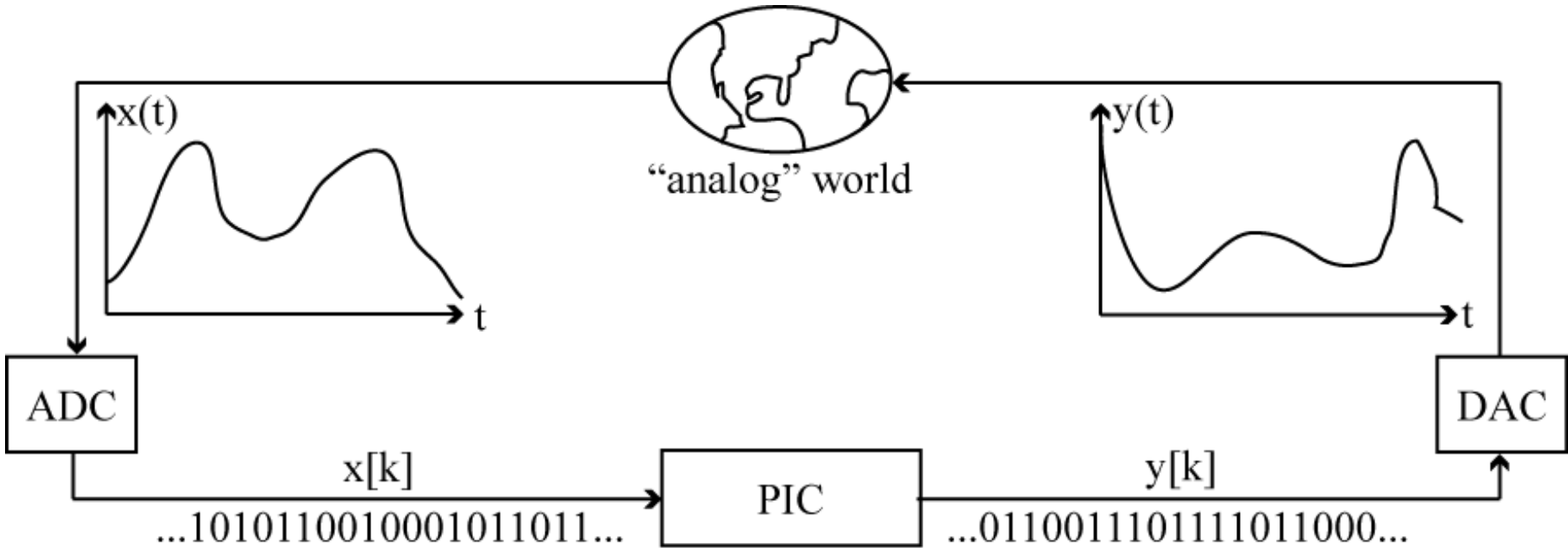


Digital Signal Processing



Analog-to-Digital Converter (ADC) converts an input analog value to an output digital representation.

This digital data is processed by a microprocessor and output to a Digital-to-Analog Converter (DAC) the converts an input binary value to an output voltage $v_{0.6}$

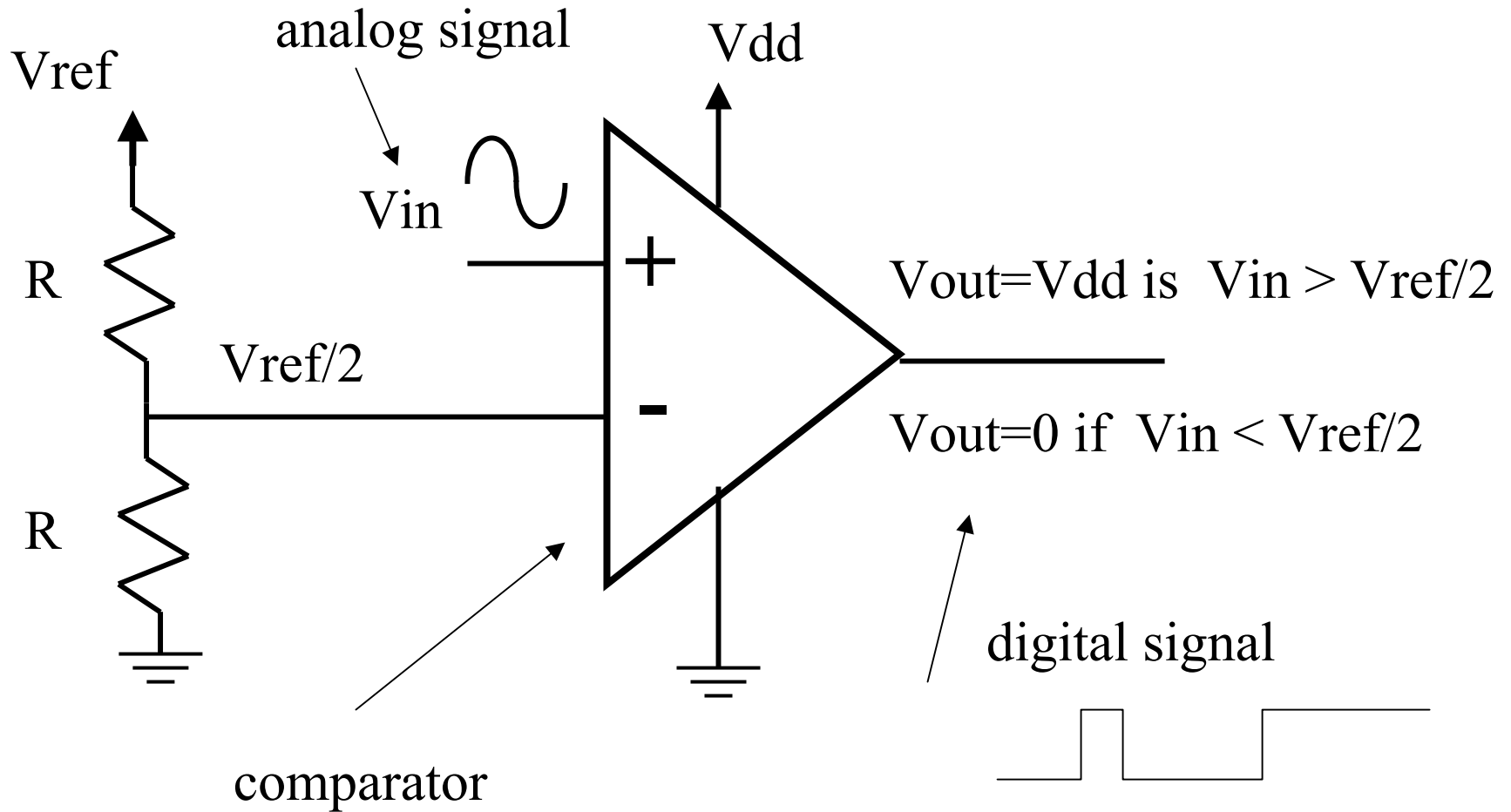
Applications

- Audio
 - Speech recognition
 - special effects (reverb, noise cancellation, etc)
- Video
 - Filtering
 - Special effects
 - Compression
- Data logging

Vocabulary

- ADC (Analog-to-Digital Converter) – converts an analog signal (voltage/current) to a digital value
- DAC (Digital-to-Analog Converter) – converts a digital value to an analog value (voltage/current)
- Sample period – for ADC, time between each conversion
 - Typically, samples are taken at a fixed rate
- Vref (Reference Voltage) – analog signal varies between 0 and Vref, or between +/- Vref
- Resolution – number of bits used for conversion (8 bits, 10 bits, 12 bits, 16 bits, etc).
- Conversion Time – the time it takes for a analog-to-digital conversion

A 1-bit ADC



ADC Resolution

For an N-bit ADC, the smallest input voltage that can be resolved is 1 LSB, or:

$$1/2^N * (V_{ref+} - V_{ref-})$$

Where V_{ref+} is the positive reference voltage and V_{ref-} is the negative reference voltage.

We will use $V_{ref-} = 0$ V, and refer to V_{ref+} as simply V_{ref} , so this simplifies to

$$1/2^N * V_{ref}.$$

For $V_{ref} = 4$ V, and $N = 4$, what is 1 LSB?

$$1/2^4 * 4 \text{ V} = 1/16 * 4 \text{ V} = 0.25 \text{ V}.$$

ADC, DAC Equations

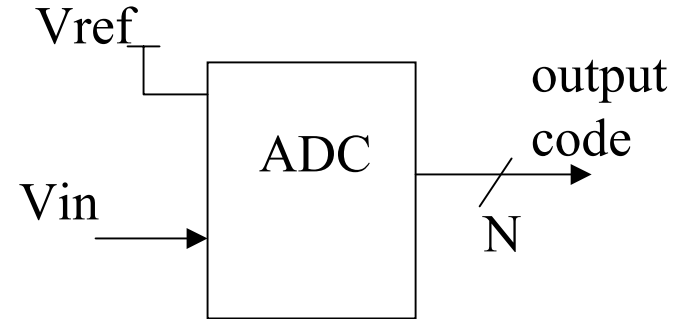
ADC: V_{in} = input voltage, V_{ref+} = reference voltage, $V_{ref-} = 0$ V.

N = number of bits of precision

$$V_{in} / V_{ref} * 2^N = \text{output_code}$$

$$\text{output_code} / 2^N * V_{ref} = V_{in}$$

$$1 \text{ LSB} = V_{ref} / 2^N$$



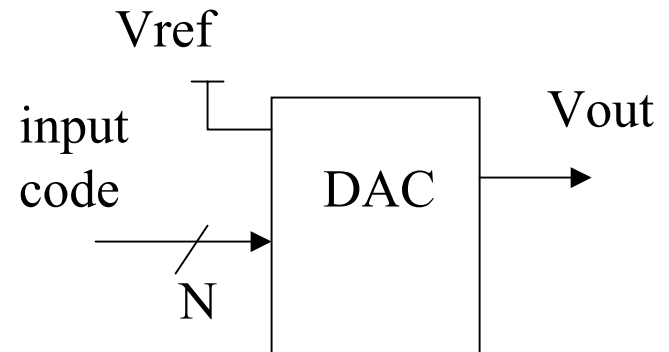
DAC: V_{out} = output voltage, V_{ref} = reference voltage,

N = number of bits of precision

$$V_{out} / V_{ref} * 2^N = \text{input_code}$$

$$\text{input_code} / 2^N * V_{ref} = V_{out}$$

$$1 \text{ LSB} = V_{ref} / 2^N$$



Sample ADC, DAC Computations

If $V_{ref} = 5V$, and a 10-bit A/D output code is 0x12A, what is the ADC input voltage?

$$\begin{aligned} V_{in} &= \text{output_code} / 2^N * V_{ref} = (0x12A) / 2^{10} * 5 \text{ V} \\ &= 298 / 1024 * 5 \text{ V} = 1.46 \text{ V (ADC } V_{in}) \end{aligned}$$

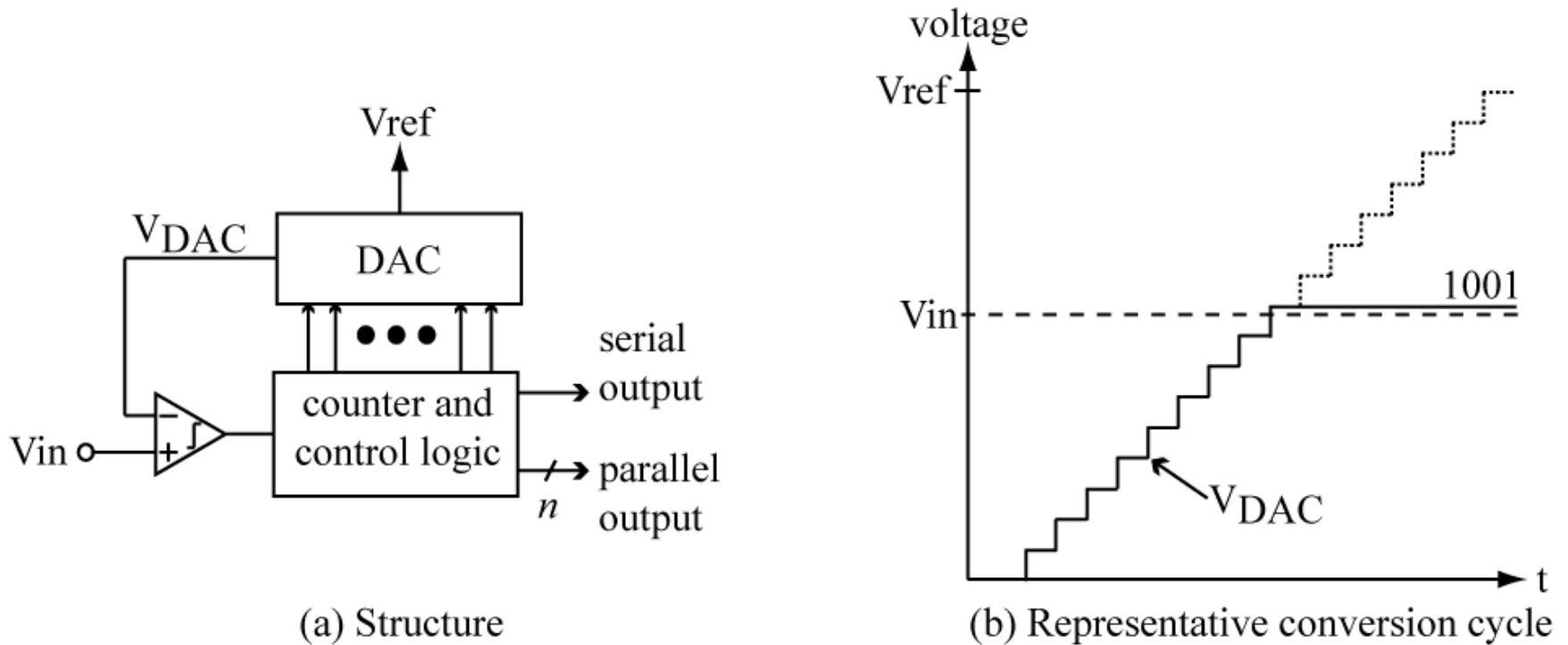
If $V_{ref} = 5V$, and an 8-bit DAC input code is 0xA9, what is the DAC output voltage?

$$\begin{aligned} V_{out} &= \text{input_code} / 2^N * V_{ref} = (0xA9) / 2^8 * 5 \text{ V} \\ &= 169 / 256 * 5 \text{ V} = 3.3 \text{ V (DAC } V_{out}) \end{aligned}$$

If $V_{ref} = 4V$, and an 8-bit A/D input voltage is 2.35 V, what is the ADC output code?

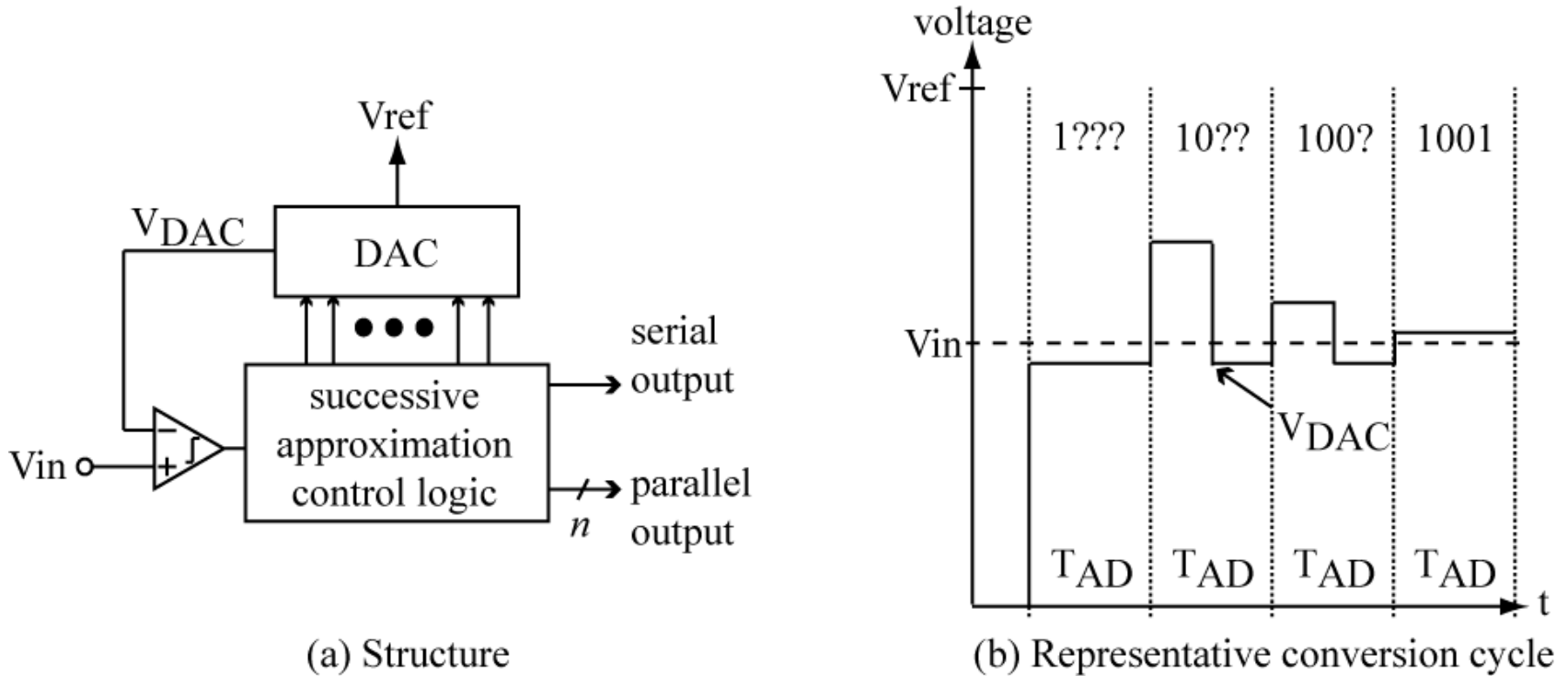
$$\begin{aligned} \text{output code} &= V_{in} / V_{ref} * 2^N = 2.35 \text{ V} / 4 \text{ V} * 2^8 \\ &= .5875 * 256 = 150.4 = 150 = 0x96 \text{ (ADC output code)} \end{aligned}$$

Counter Ramp ADC



Control logic use a counter to apply successive codes 0,1,2,3,4... to DAC (Digital-to-Analog Converter) until DAC output is greater than V_{in} . This is SLOW, and have to allocate the worst case time for each conversion, which is 2^N clock cycles for an N-bit ADC.

Successive Approximation ADC



Initially set V_{DAC} to $\frac{1}{2} V_{ref}$, then see if V_{in} higher or lower than V_{DAC} . If $> \frac{1}{2} V_{ref}$, then next guess is between V_{ref} and $\frac{1}{2} V_{ref}$, else next guess is between $\frac{1}{2} V_{ref}$ and GND. Do this for each bit of the ADC. Takes N clock cycles.

Successive Approximation Example

Given a 4-bit Successive Approximation ADC, and $V_{ref} = 4 \text{ V}$.

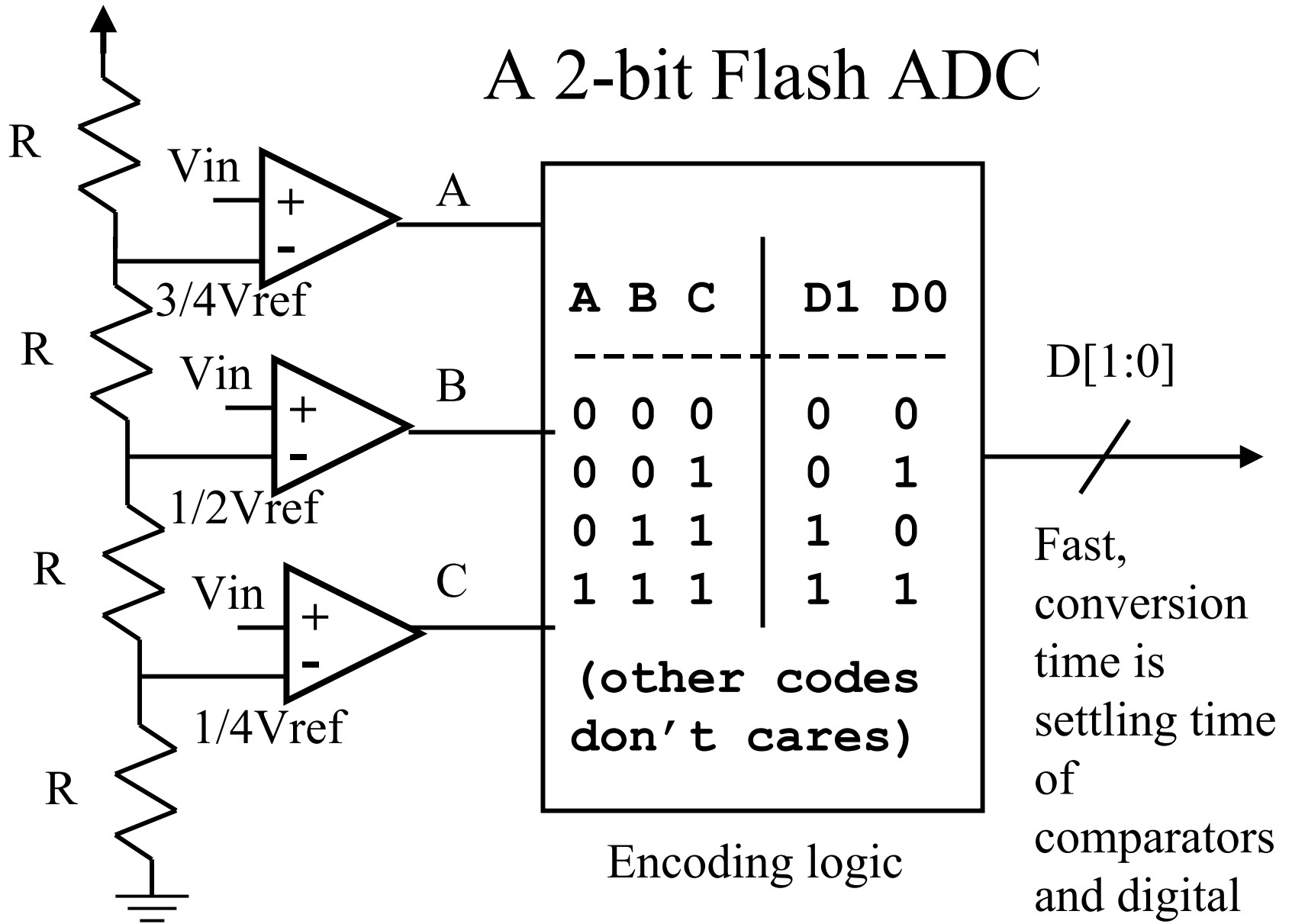
Let $V_{in} = 3.14159 \text{ V}$. Clear DAC input to $0b0000$.

1. First guess, DAC input = $0b1000 = 8$, so $V_{dac} = 8/2^4 * 4 \text{ V} = 8/16 * 4 \text{ V} = 2 \text{ V}$.
 $V_{dac} (2 \text{ V}) < V_{in} (3.14159 \text{ V})$, so guess of '1' for MSb of DAC was correct.
2. Set next bit of DAC to '1', DAC input = $0b1100 = 12$, so $V_{dac} = 12/16 * 4 = 3 \text{ V}$.
 $V_{dac} (3 \text{ V}) < V_{in} (3.14159 \text{ V})$, so guess of '1' for bit2 of DAC was correct.
3. Set next bit of DAC to '1', DAC input = $0b1110 = 14$, so $V_{dac} = 14/16 * 4 = 3.5 \text{ V}$.
 $V_{dac} (3.5 \text{ V}) > V_{in} (3.14159 \text{ V})$, so guess of '0' for bit1 of DAC was incorrect.
Reset this bit to '0'.
4. Set last bit of DAC to '1', DAC input = $0b1101 = 13$, so $V_{dac} = 13/16 * 4 = 3.25 \text{ V}$.
 $V_{dac} (3.25 \text{ V}) > V_{in} (3.14159 \text{ V})$, so guess of '0' for bit0 of DAC was incorrect.
Reset this bit to '0'.

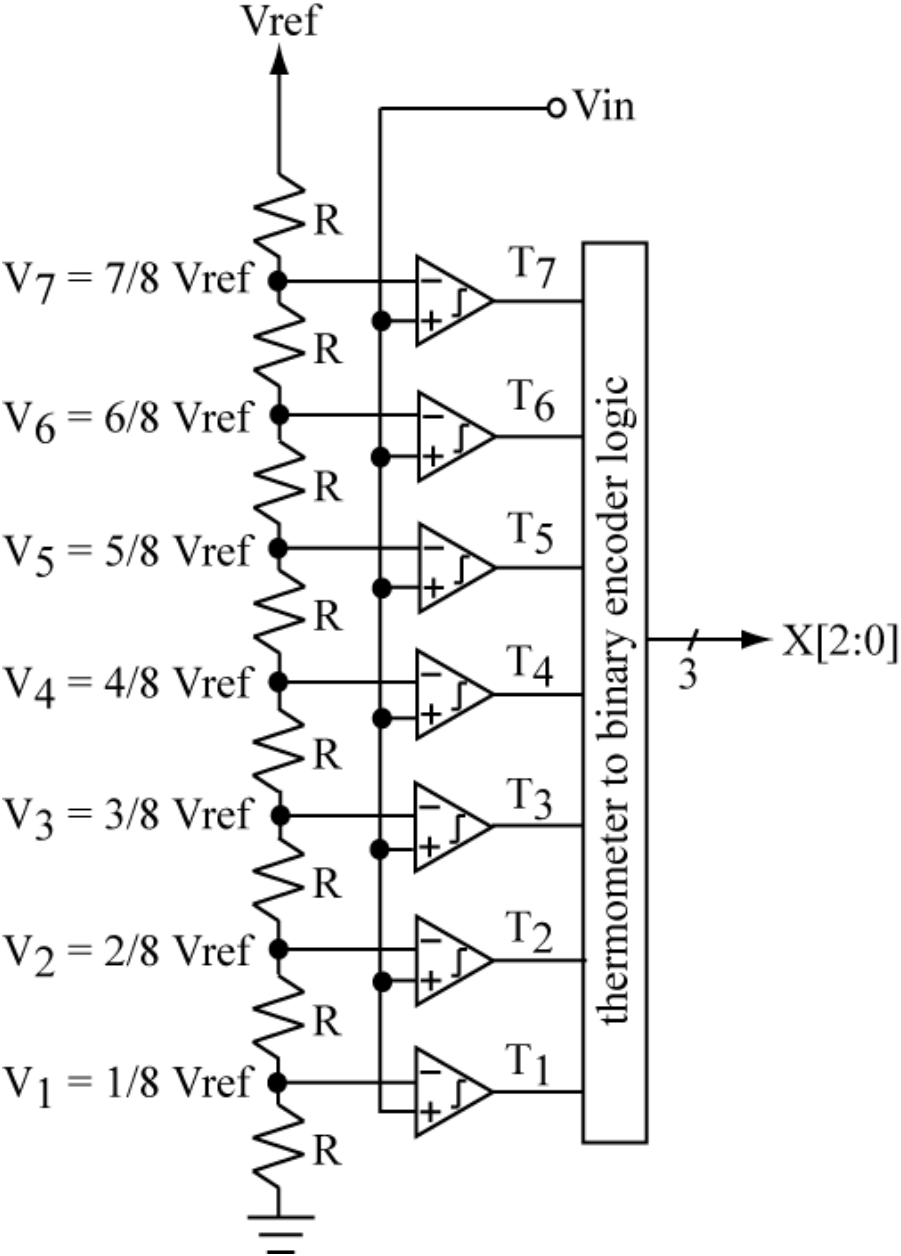
Final ADC output code is $0b1100$.

Check result: output code = $V_{in}/V_{ref} * 2^N = 3.14159/4 * 16 = 12.57 = 12$ (truncated).

A 2-bit Flash ADC



A 3-bit Flash ADC



ADC Architecture Summary

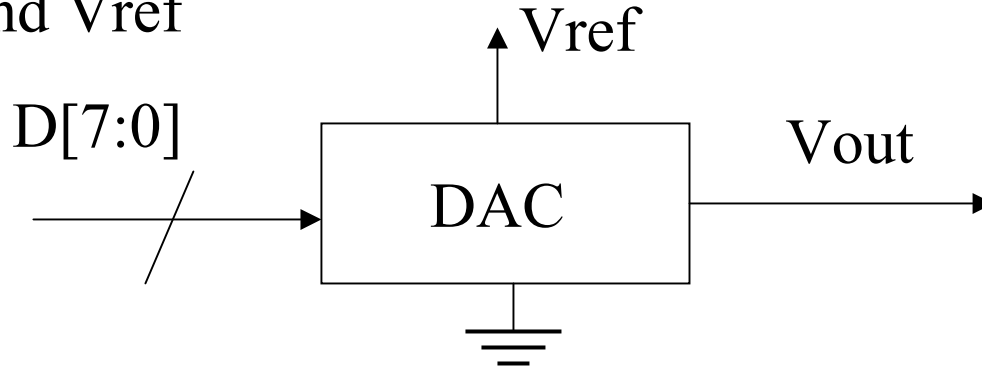
- *Flash* ADCs
 - Fastest possible conversion time
 - Requires the most transistors of any architecture
 - N-bit converter requires 2^N-1 comparators.
 - Commercially available flash converters up to 12 bits.
 - Conversion done in one clock cycle
- *Successive approximation* ADCs
 - Use only one comparator
 - Take one clock cycle per bit
 - High precision (16-bit converters are available)

Commercial ADCs

- Key timing parameter is *conversion time* – how long does it take to produce a digital output once a conversion is started
- Up to 16-bit ADCs available
- Separated into fast/medium/low speed families
 - Serial interfaces common on medium/low speed ADCs
- For high-precision ADCs, challenge is keeping system noise from affecting conversion
 - Assume a 16-bit DAC, and a 4.1 V reference, then 1 LSB = $4.1/2^{16} = 62 \mu\text{V}$.

Digital-to-Analog Conversion

For a particular binary code, output a voltage between 0 and V_{ref}



Assume a DAC that uses an unsigned binary input code, with $0 < V_{out} < V_{ref}$. Then

$$D = 0000\ 0000 \quad V_{out} = 0V$$

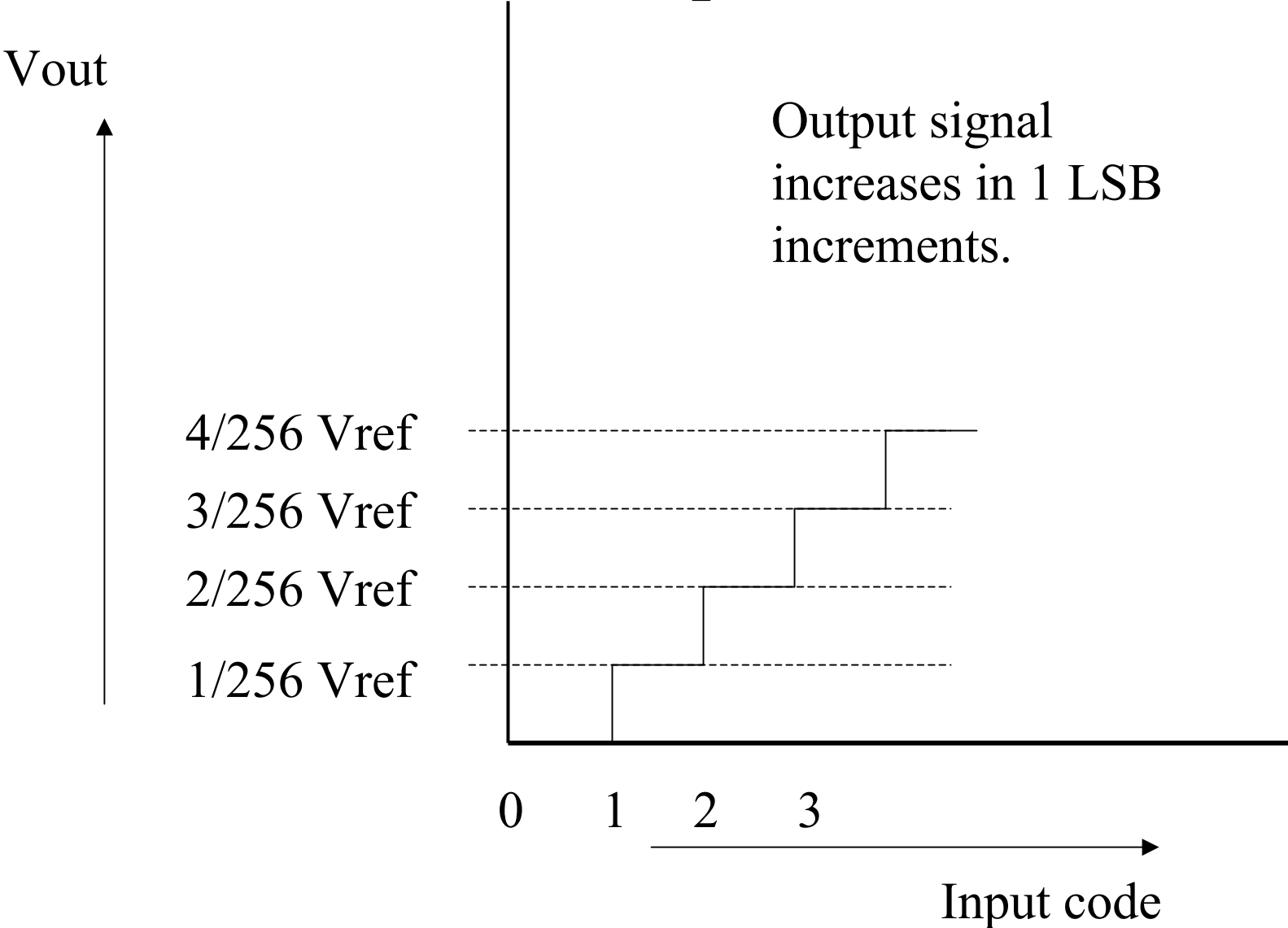
$$D = 0000\ 0001 \quad V_{out} = V_{ref}(1/256) \quad (\text{one LSB})$$

$$D = 0000\ 0010 \quad V_{out} = V_{ref}(2/256)$$

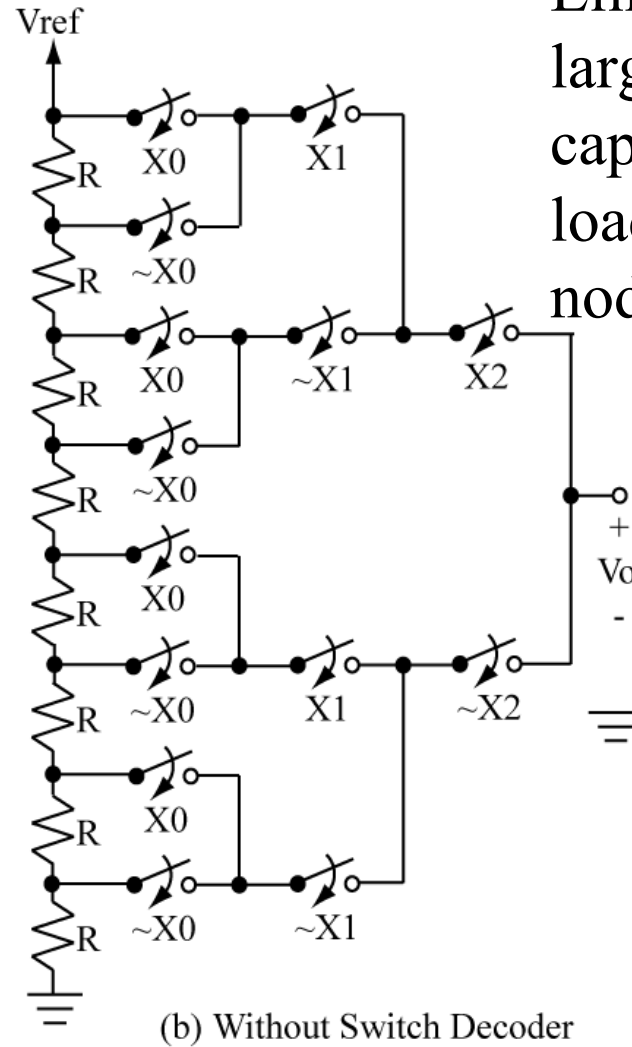
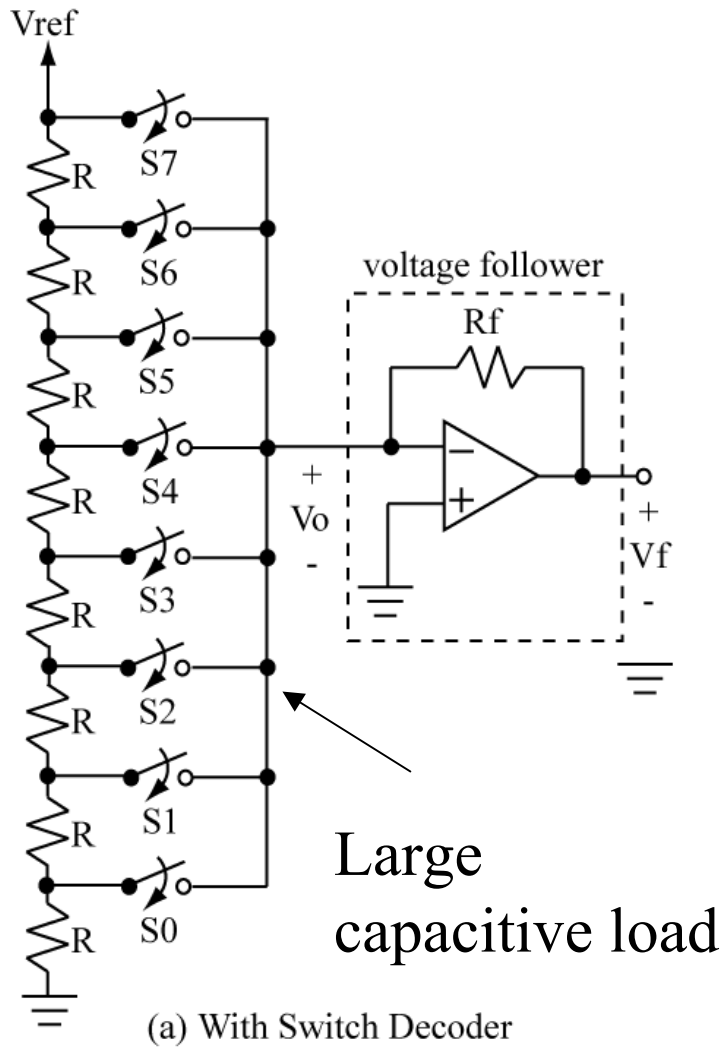
...

$$D = 1111\ 1111 \quad V_{out} = V_{ref}(255/256) \quad (\text{full scale})$$

DAC Output Plot



Flash DAC



Eliminates large capacitive load at one node.

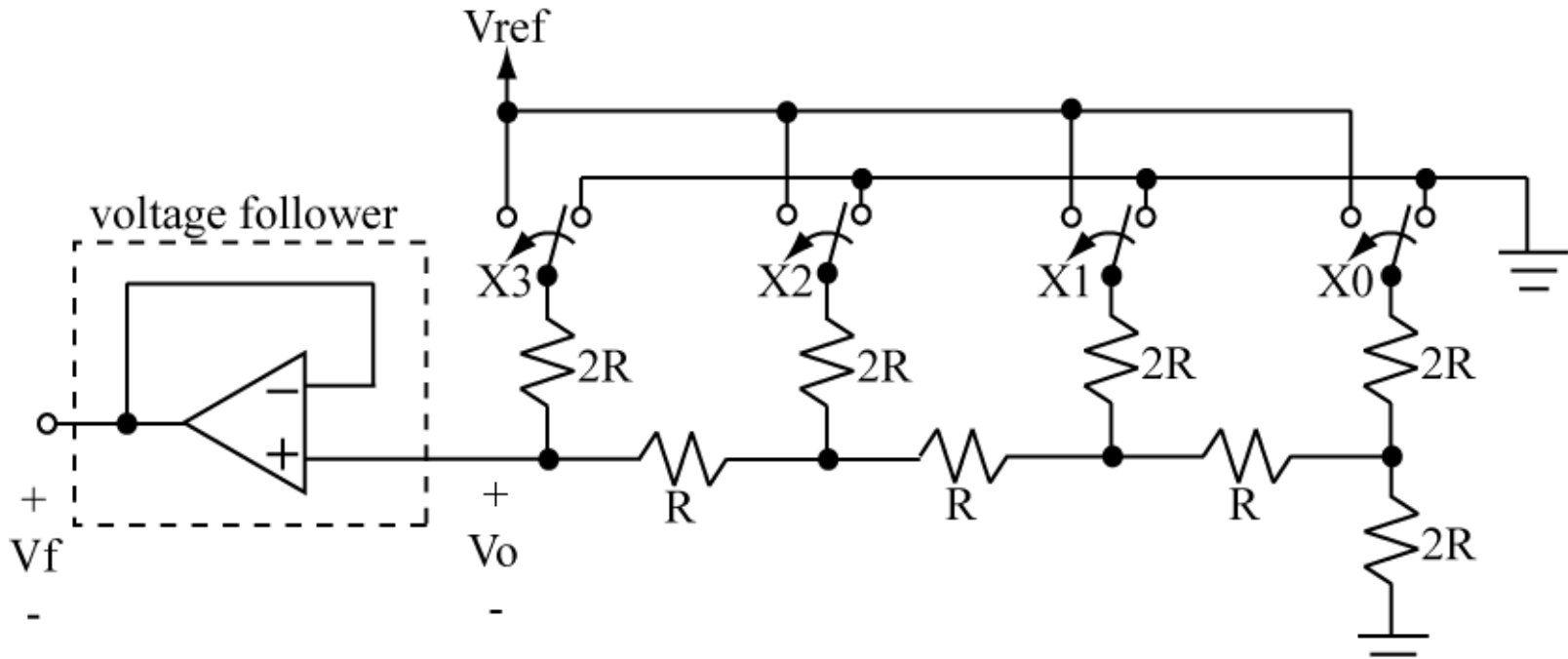
N-bit DAC requires 2^N resistors!

V 0.6

Copyright Thomson/Delmar Learning 2005. All Rights Reserved.

17

R-2R Ladder DAC



Resistor ladder divides the V_{ref} voltage to a binary weighted value 4-bit value, with the 4-bits equal to $X_3 X_2 X_1 X_0$

If the switch X_n is connected to V_{ref} , then that bit value is '1', if the switch X_n is not connected to V_{ref} , then that bit value is '0'.

Majority of DACs use this architecture as requires far less resistors than flash DACs.

Sample DAC Computations

If $V_{ref} = 5V$, and the 8-bit input code is $0x8A$, what is the DAC output voltage?

$$\begin{aligned} \text{input_code}/2^N * V_{ref} &= (0x8A)/2^8 * 5 \text{ V} \\ &= 138/256 * 5 \text{ V} = 2.70 \text{ V (Vout)} \end{aligned}$$

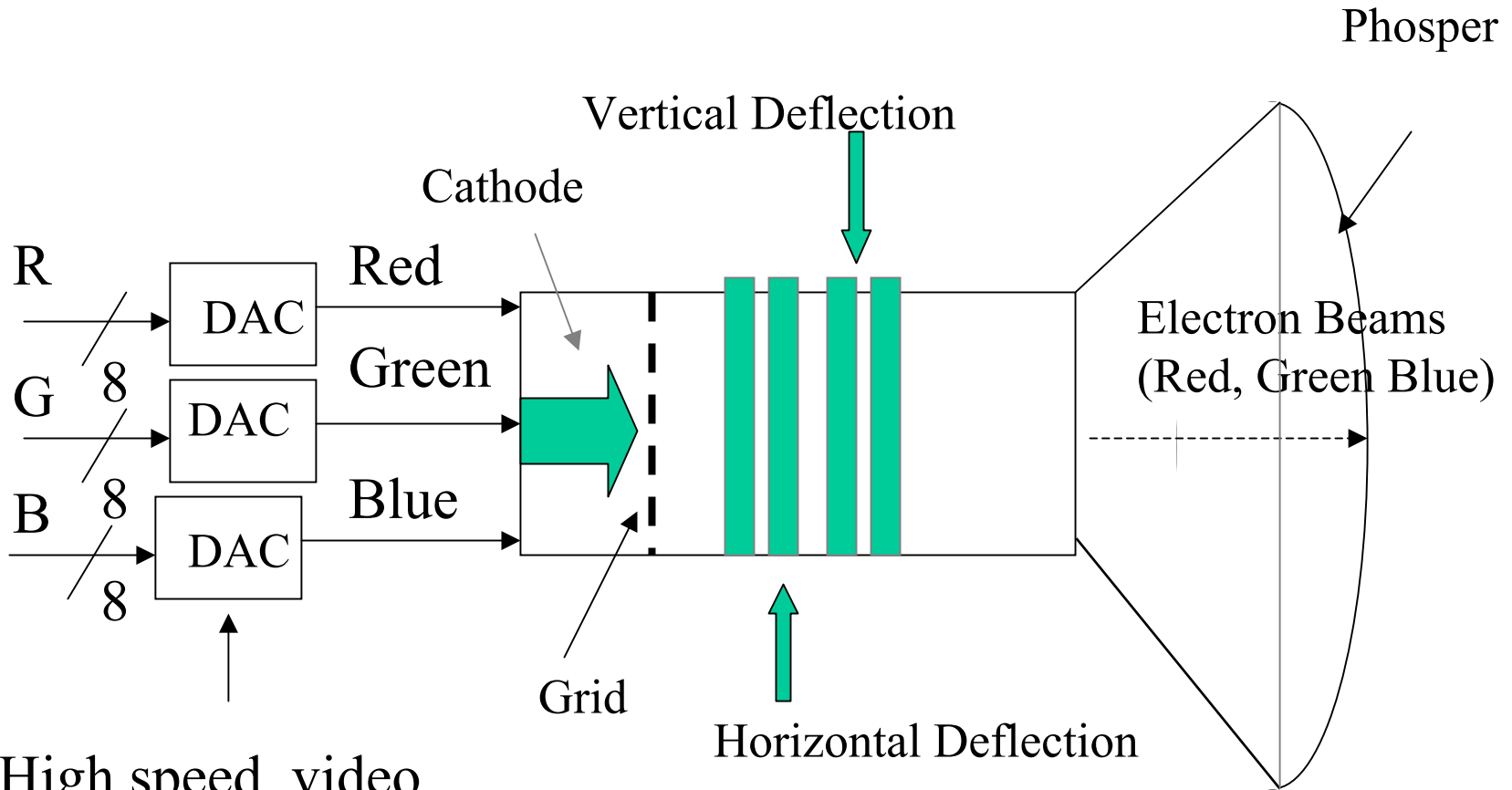
If $V_{ref} = 4V$, and the DAC output voltage is 1.25 V , what is the 8-bit input code?

$$\begin{aligned} V_{out}/ V_{ref} * 2^N &= 1.25 \text{ V}/4 \text{ V} * 2^8 \\ &= 0.3125 * 256 = 80 = 0x50 \text{ (input_code)} \end{aligned}$$

Commercial DACs

- Either voltage or current DACs
 - Current DACs require an external operational amplifier to convert to voltage
- Precision up to 16-bits
- Key timing parameter is *settling time* - amount of time it takes to produce a stable output voltage once the input code has changed
- We will use an 8-bit voltage DAC with an I2C interface from Maxim semiconductor

DAC Application



High speed video
DACs produce RGB
signals for color CRT

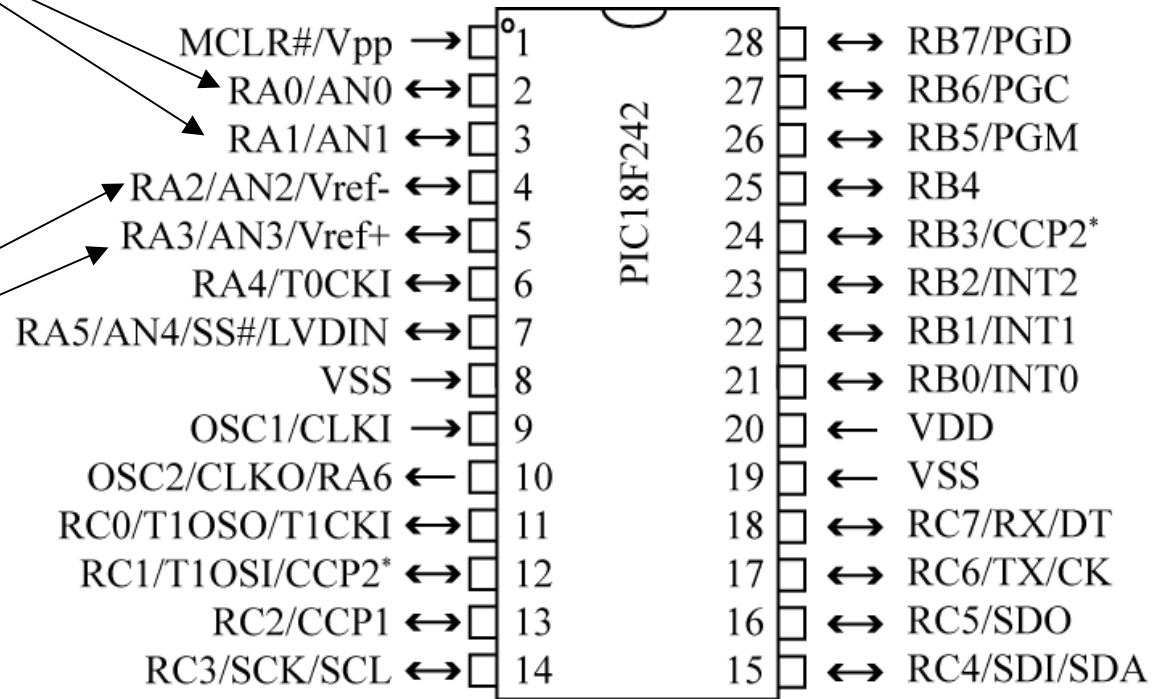
PIC18Fxx2 ADC

- PIC18F242 has onboard ADC
 - Successive approximation
 - 10 bit resolution
 - Reference voltage can be V_{dd} or separate voltage
 - Multiple input (more than one input channel)
 - Clock source for ADC is either a divided F_{osc} , or an internally generated clock. The ADC clock period (T_{ad}) cannot be less than 1.6 μs
- Total conversion time is $10 * T_{ad} + T_{aq}$ (acquisition)
 - T_{aq} is approximately 20 μs ; acquisition time is the amount of time input capacitor requires to charge up to input voltage if the analog input channel is changed.
 - So a 20 Mhz F_{osc} , $T_{osc} = .05 \mu s$, so $32T_{osc} = 1.6 \mu s$; conversion time = $10 * 1.6 \mu s + 20 \mu s = 36 \mu s$.

Input Pins

Analog input channels (AN0, AN1, AN4)

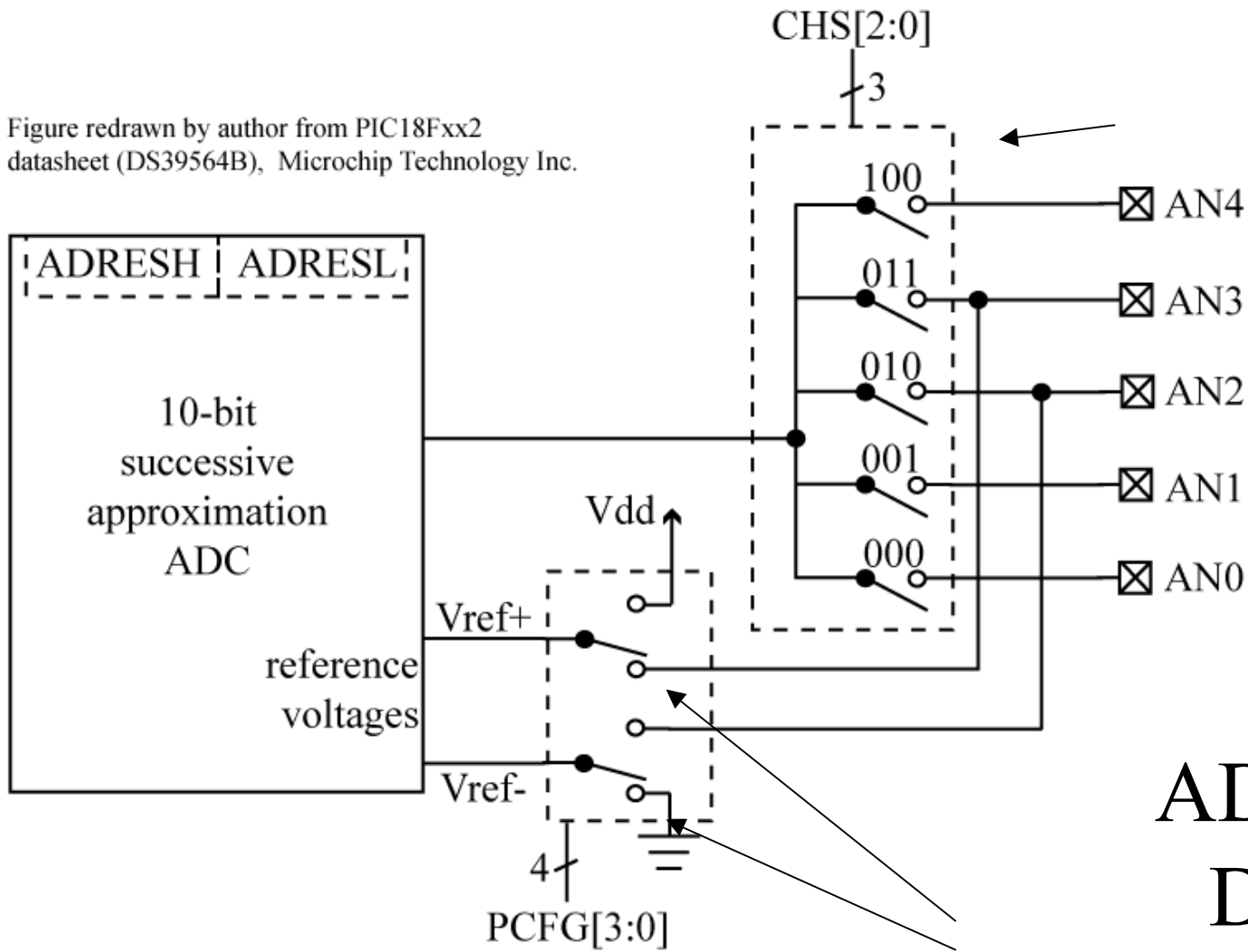
Can be analog
input channels
or Vref+/Vref-



*RB3 is the alternate pin for the CCP2 pin multiplexing

Figure redrawn by author from PIC18Fxx2 datasheet (DS39564B), Microchip Technology Inc.

Figure redrawn by author from PIC18Fxx2
datasheet (DS39564B), Microchip Technology Inc.



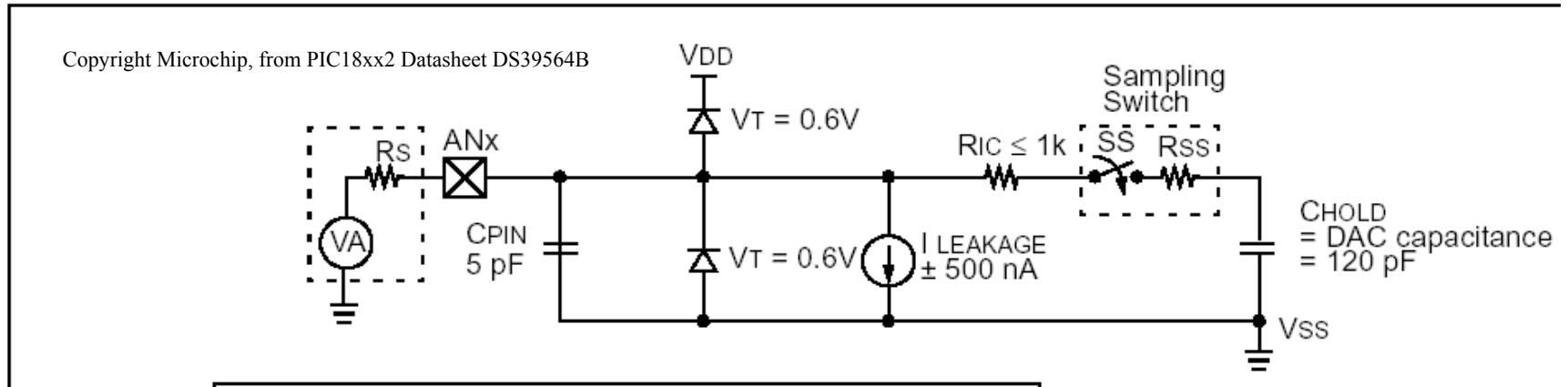
Channel
select
analog
mux.

ADC Block Diagram

Vref+/Vref- select

Acquisition Time

FIGURE 11-2: ANALOG INPUT MODEL



Acquisition time is the time required for the analog input voltage to be sampled by the input capacitor if the analog channel is changed.

On the PIC18Fxx2, the acquisition time is $20 \mu\text{s}$.

PIC18Fxx2 ADC Port Configuration Bits

PCFG[3:0]	AN4	AN3	AN2	AN1	AN0	V _{REF+}	V _{REF-}
00x0	A	A	A	A	A	V _{DD}	V _{SS}
00x1	A	V _{REF+}	A	A	A	AN3	V _{SS}
0100	D	A	D	A	A	V _{DD}	V _{SS}
0101	D	V _{REF+}	D	A	A	AN3	V _{SS}
011x	D	D	D	D	D	---	---
1x00	A	V _{REF+}	V _{REF-}	A	A	AN3	AN2
1001	A	A	A	A	A	V _{DD}	V _{SS}
1010	A	V _{REF+}	A	A	A	AN3	V _{SS}
1011	A	V _{REF+}	V _{REF-}	A	A	AN3	AN2
1101	D	V _{REF+}	V _{REF-}	A	A	AN3	AN2
1110	D	D	D	D	A	V _{DD}	V _{SS}
1111	D	V _{REF+}	V _{REF-}	D	A	AN3	AN2

Bits PCFG[3:0] are in register ADCON1[3:0].

Values shown are for the PIC18F2x2.

Commonly used value as this sets AN0 to be analog input, other A port bits as digital inputs, and Vref+ = V_{DD}, Vref- = V_{SS}

PIC18Fxx2 ADC Configuration Registers

Name	SFR (bit)	Comments
ADON	ADCON0 [0]	0 = ADC is powered off 1 = ADC is powered up
GO/DONE#	ADCON0 [2]	0 = A/D conversion not in progress 1 = conversion in progress (set this bit to start ADC conversion)
CHS [2:0]	ADCON0 [5:3]	ADC channel select bits 000 = AN0 001 = AN1 010 = AN2 011 = AN3 100 = AN4 Selects which ADC input is being converted
ADCS [2:0]	ADCON1 [6]:ADCON0 [7:6]	ADC conversion clock select bits (selects clock source for ADC successive approximation cycles)
PCFG [3:0]	ADCON1 [3:0]	ADC port configuration control bits (selects number of analog channels and ADC references)
ADFM	ADCON1 [7]	0 = left justified in ADRESH:ADRESL 1 = right justified in ADRESH:ADRESL
ADIE	PIE1 [6]	ADC interrupt enable
ADIP	IPR1 [6]	ADC interrupt priority select
ADIF	PIR1 [6]	ADC interrupt interrupt flag

PIC18Fxx2 ADC Conversion Clock Selection

ADCON1 [6] ADCS2	ADCON0 [7:6] ADCS [1:0]	A/D Clock
0	00	$F_{OSC}/2$
0	01	$F_{OSC}/8$
0	10	$F_{OSC}/32$
x	11	$F_{ADC RC}$ (internal ADC oscillator)
1	00	$F_{OSC}/4$
1	01	$F_{OSC}/16$
1	10	$F_{OSC}/64$

ADC clock must have period that is $> 1.6 \mu s$.

If internal oscillator is chosen, then ADC clock is guaranteed to meet period of constrain of $1.6 \mu s$.

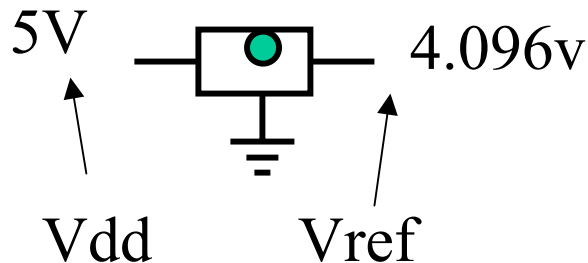
For any other clock source selection, you **MUST** compute the period and ensure that it is $> 1.6 \mu s$, or incorrect operation may result.

Voltage References

Stability of voltage reference is critical for high precision conversions.

We will use V_{dd} as our voltage reference for convenience, but will be throwing away at least two bits of precision due to V_{dd} fluctuations.

Example Commercial voltage reference: 2.048v, 2.5v , 3v, 3.3v, 4.096v, 5v (Maxim 6029)

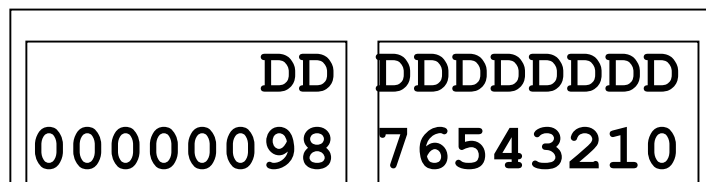


Key parameter for a voltage is stability over temperature operating range. Need this to be less than $\frac{1}{2}$ of a LSB value.

PIC18Fxx2 ADC Registers

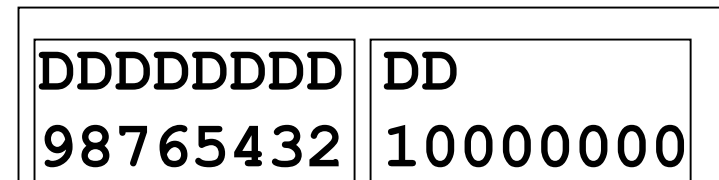
- ADCON0, ADCON1 – configuration registers
 - ADCON1 used to configure port A for analog/digital inputs, voltage reference
 - ADCON0 used for clock selection, analog input selection, start/finish conversion status.
- ADRESH, ADRESL -10-bit results returns in two registers
 - 10-bit result can be configured to be left or right justified.

ADRESH : ADRESL



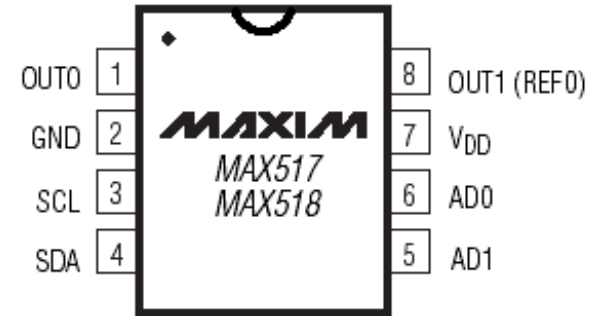
Right justified

ADRESH : ADRESL



Left justified

MAXIM 517 DAC



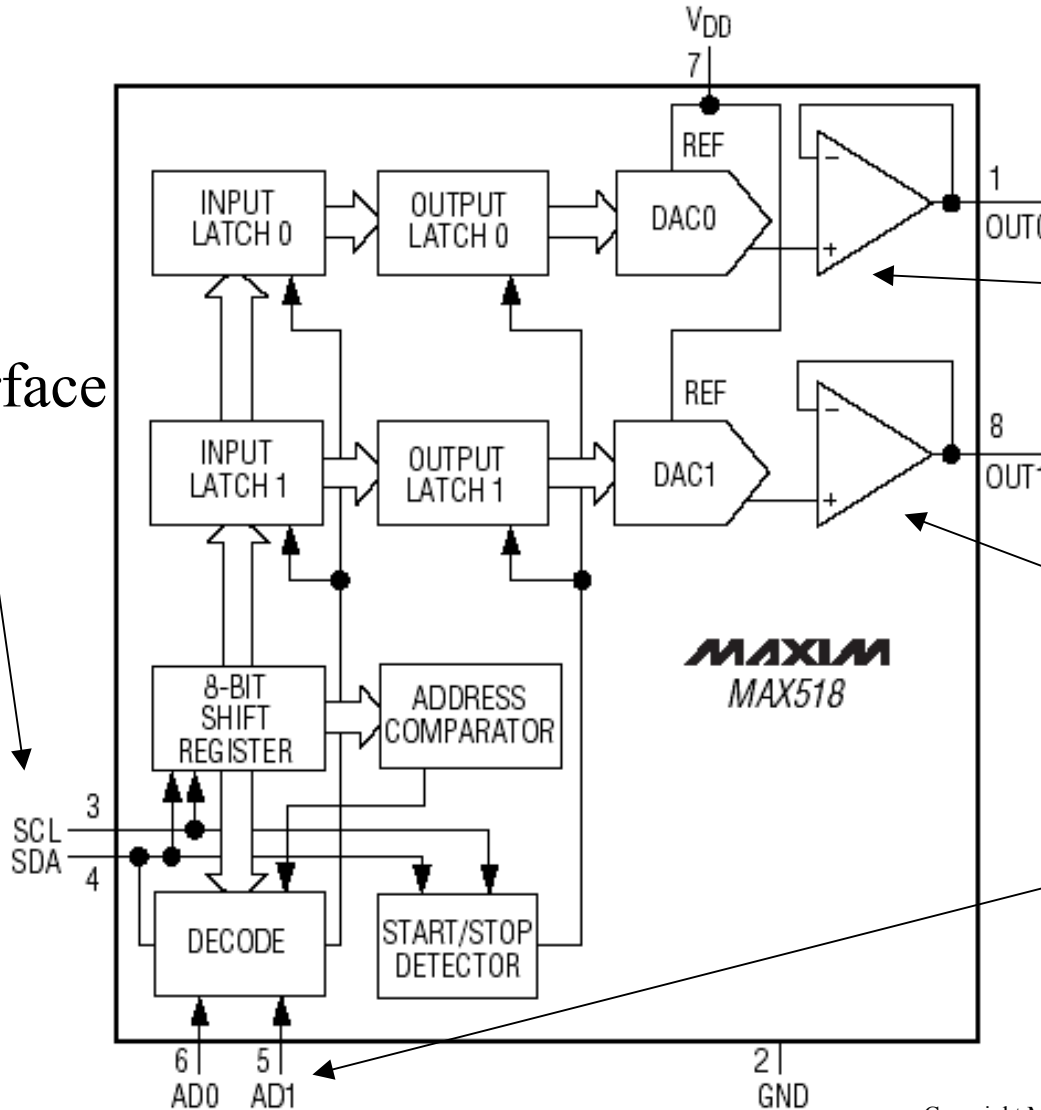
DIP/SO

R/2R DAC

Not present on Max517, V_{ref} instead.

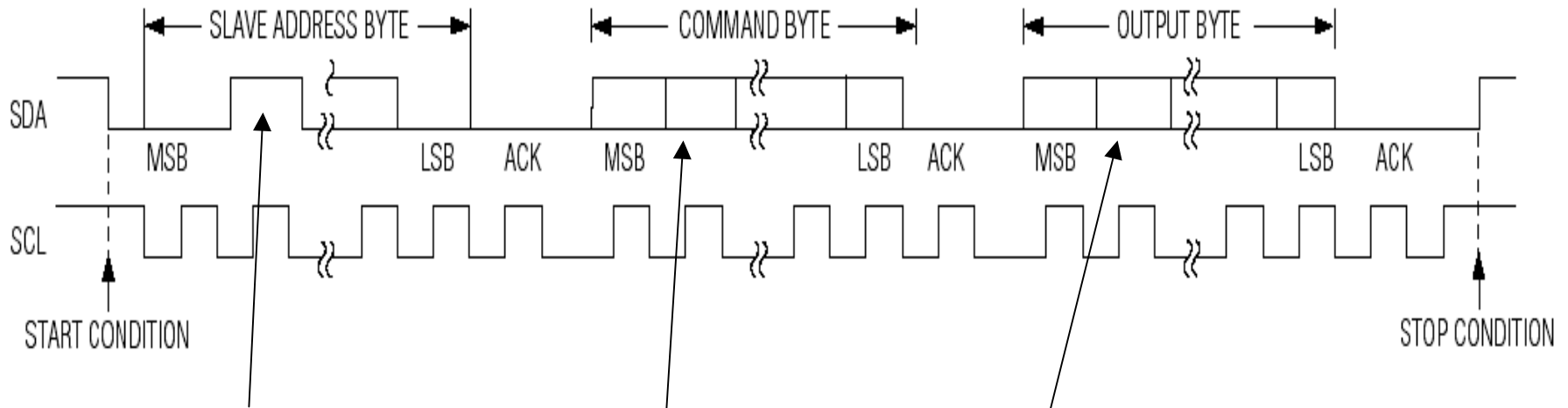
Personalizes device address

I2C interface



Max517 I2C Transaction

Copyright Maxim, from MAX518/517 Datasheet 19-0293



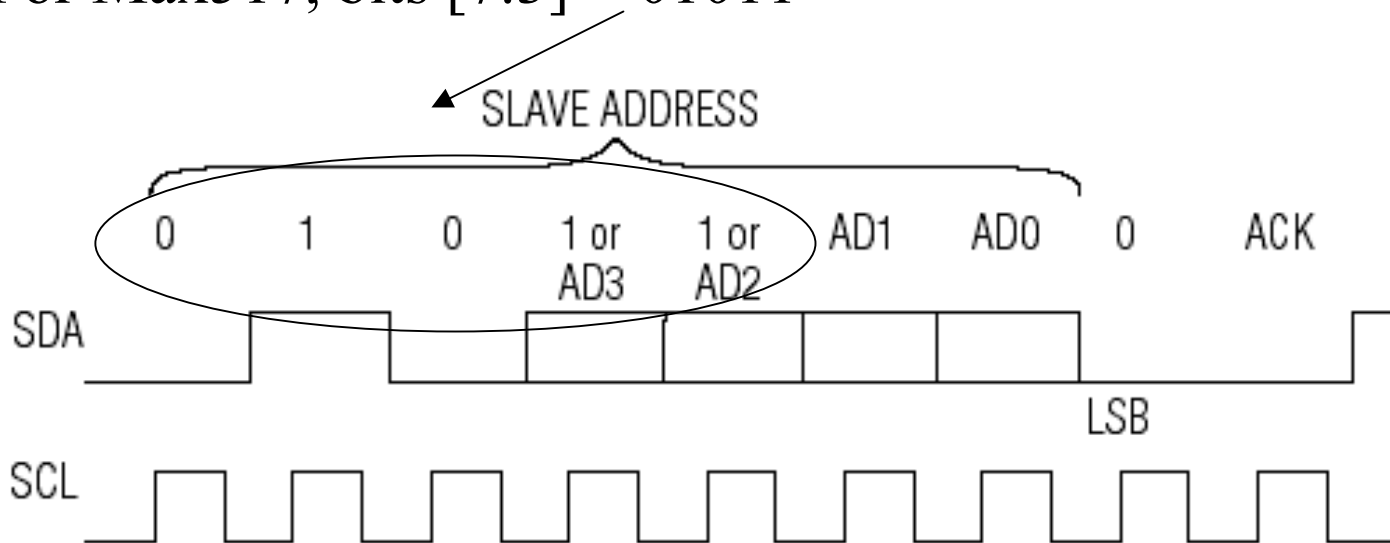
First byte: Device address

Second Byte: DAC
command byte

Third Byte:
output byte to
DAC

Device Address Format

For Max517, bits [7:3] = 01011

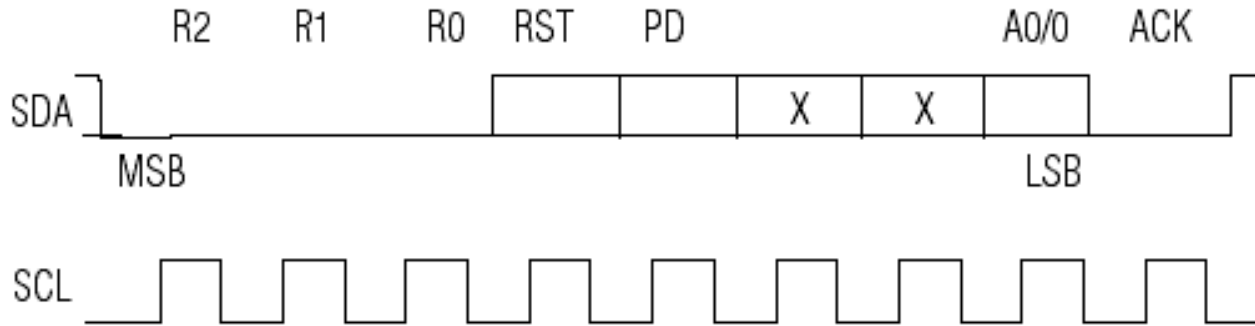


SLAVE ADDRESS BITS AD0, AD1, AD2, AND AD3 CORRESPOND TO THE LOGIC STATE OF THE ADDRESS INPUT PINS.

If AD1:AD0 tied to gnd then address is: 01011000 = 0x58

Command Format

Only command byte we will use for Max517 will be 00000000 = 0x00 as this does a write to DAC0.



R2, R1, R0: RESERVED BITS. SET TO 0.

RST: RESET BIT, SET TO 1 TO RESET ALL DAC REGISTERS.

PD: POWER-DOWN BIT. SET TO 1 TO PLACE THE DEVICE IN THE 4 μ A SHUTDOWN MODE. SET TO 0 TO RETURN TO THE NORMAL OPERATIONAL STATE.

A0: ADDRESS BIT. DETERMINES WHICH DAC'S INPUT LATCH RECEIVES THE 8 BITS OF DATA IN THE NEXT BYTE. SET TO 0 FOR MAX517.

ACK: ACKNOWLEDGE BIT. THE MAX517/MAX518/MAX519 PULLS SDA LOW DURING THE 9TH CLOCK PULSE.

X: DONT CARE.

Copyright Maxim, from MAX518/517 Datasheet 19-0293

Timing

Copyright Maxim, from MAX518/517 Datasheet 19-0293

DYNAMIC PERFORMANCE					
Voltage Output Slew Rate		Positive and negative	MAX51 _C	2.0	V/ μ s
			MAX51 _E	1.4	
			MAX51 _M	1.0	
Output Settling Time		To 1/2 LSB, 10k Ω and 100pF load (Note 8)	6	μ s	
Digital Feedthrough		Code = 00 hex, all digital inputs from 0V to VDD	5	nV-s	

Max517 DAC has a 6 μ s settling time.

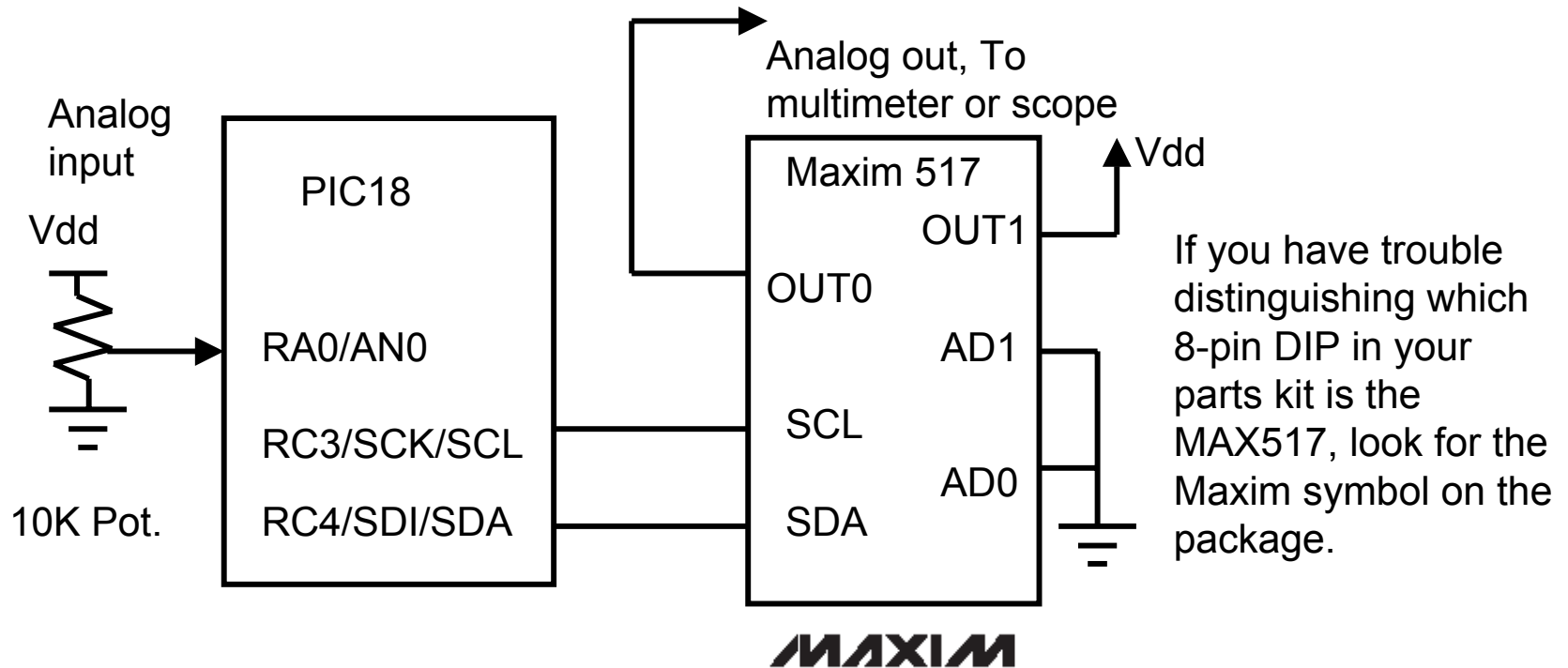
Requires 3 bytes over I2C bus to write a new value.

At 400Khz, one bit time = 2.5 μ s.

Each byte is 8 bits + 1 ACK.

So 27 bits * 2.5 μ s = 67.5 μ s not counting software overhead. So we are limited by I2C bus speed, not by the DAC settling time.

Testing the ADC and DAC

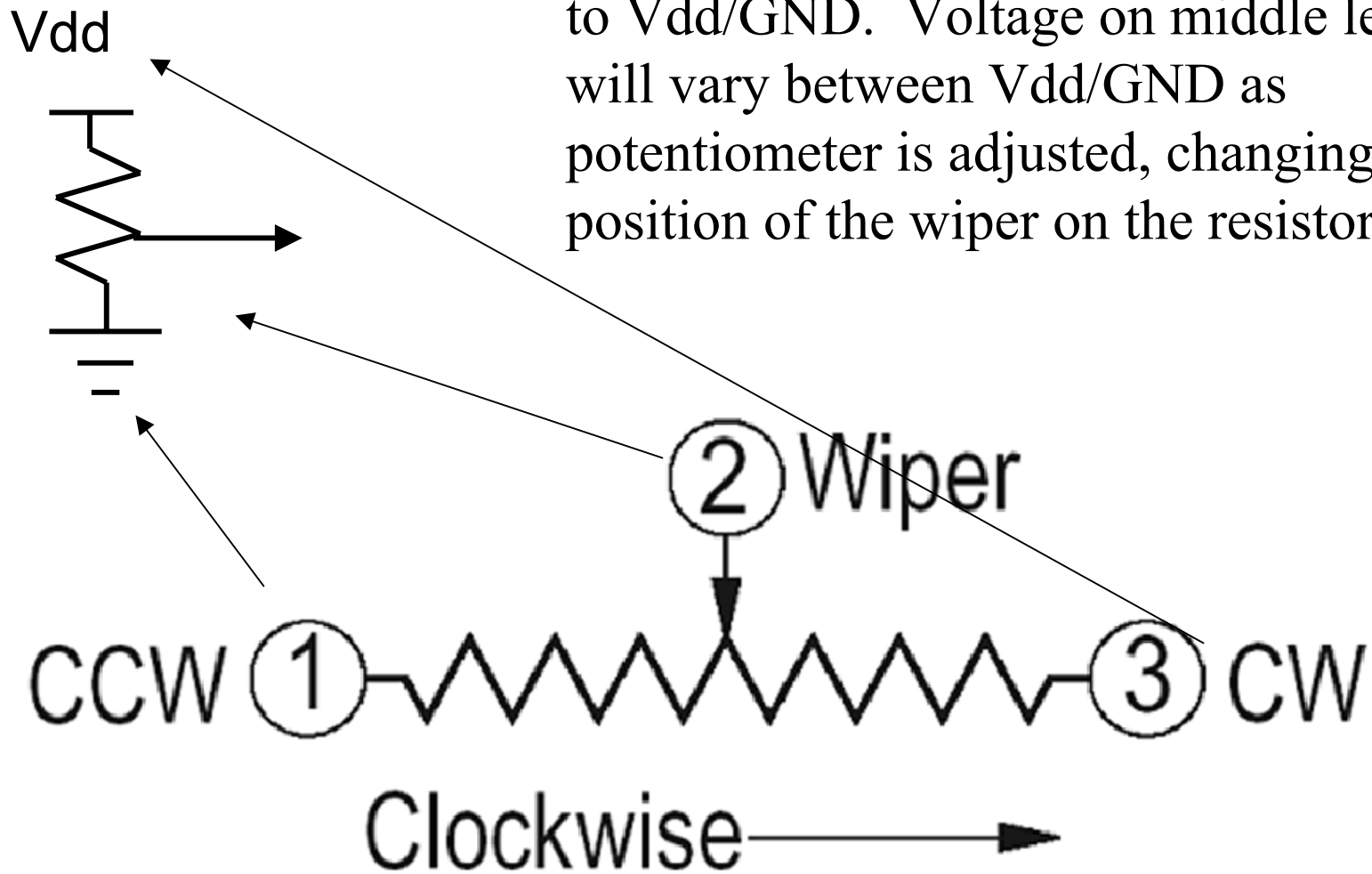


This diagram assumes that 10K pullups are already on the SCL/SDA lines from the previous lab.

Read the voltage from the potentiometer via the PIC18Fxx2 ADC, write this digital value to the DAC. The DAC output voltage should match the potentiometer voltage.

Potentiometer

A variable resistor. Tie outer two legs to Vdd/GND. Voltage on middle leg will vary between Vdd/GND as potentiometer is adjusted, changing the position of the wiper on the resistor.



dactest.c

ADC Configuration

```
// ADC Setup
TRISA = 0xFF;    // all bits of portA inputs

// ADC Setup
TRISA = 0xFF;    // all bits input
//ADCON1 = 0x8E, ADCON0 = 0x80 by bit assignments below
ADFM = 1;        // right justification
// A0 analog, others digital, Vref+ = VDD, Vref- = Vss
PCFG3 = 1; PCFG2 = 1; PCFG1 = 1; PCFG0 = 0;
ADCS2 = 0; ADCS1 = 1; ADCS0 = 0; // ADC clock = Fsoc/32
CHS2 = 0; CHS1 = 0; CHS0 = 0;    // select channel 0

ADON = 1; // turn on ADC

printf("ADC is configured!!!");
pcrlf();
```

dactest.c (cont.)

```
int adc_value;
while(1) {
    GODONE = 1; // start conversion
    // wait for end of conversion
    while (GODONE);

    // read result
    adc_value = 0;
    adc_value = adc_value | (ADRESH << 8);
    adc_value = adc_value | (ADRESL);
    printf("%x", adc_value);
    pcrLf();

    dac_value = ((adc_value >> 2) & 0x00ff);

    // now write to DAC
    update_dac(dac_value);
}
```

Read from A/D, print

This is a 10-bit value!!!

Only write upper 8 bits to DAC

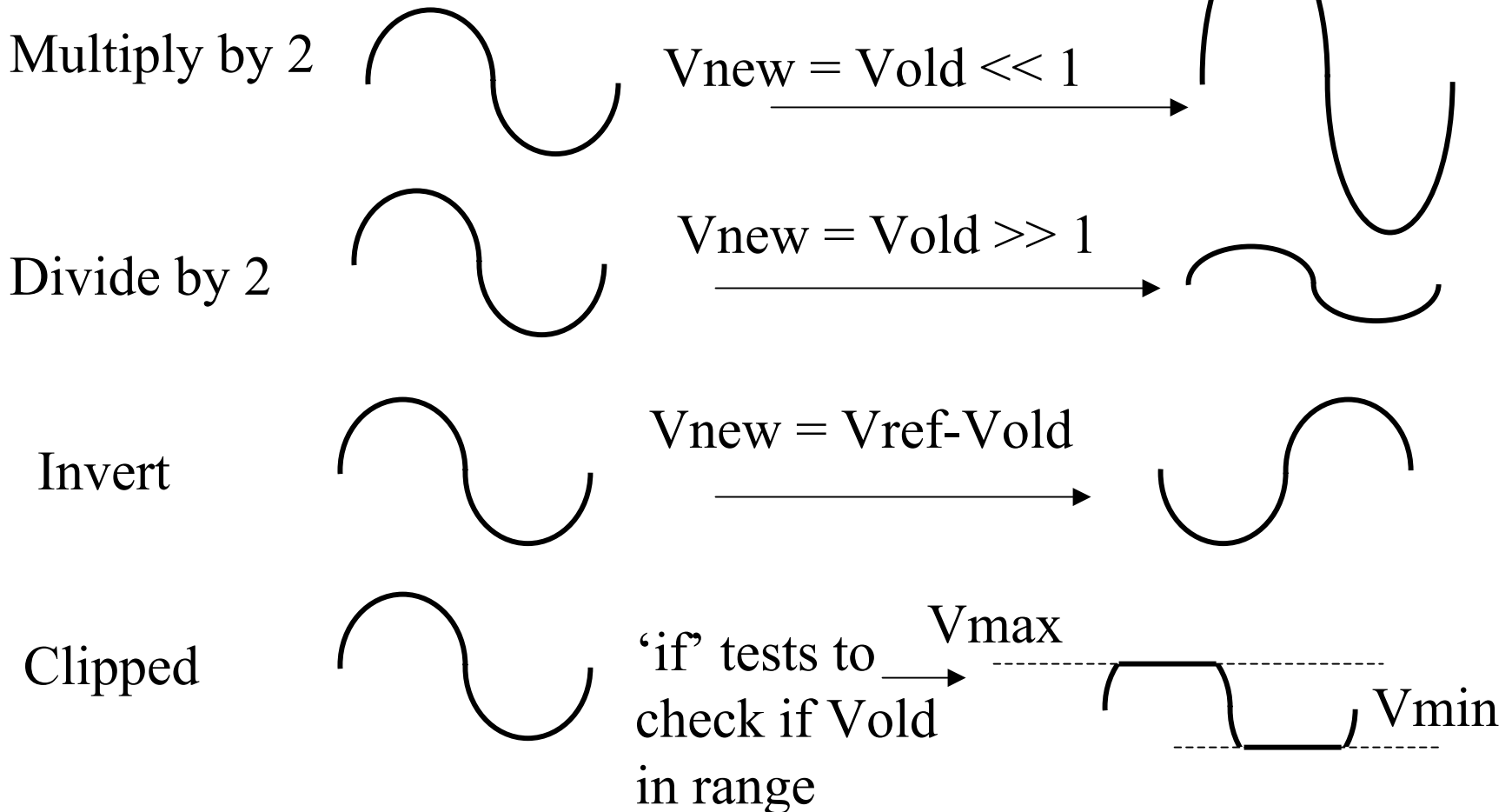
dactest.c (cont.)

```
#define DAC 0x58          /* I2C DAC 01011000 */

void update_dac(unsigned char val) {
    i2c_start()
    i2c_put(DAC);         ← device address byte
    i2c_put(0x00);       ← DAC command byte
    i2c_put(val);        ← DAC output byte
    i2c_stop();
}
```

Modifications to dactst.c for Lab #10

Modify *dactest.c* to provide four functions:



What do you have to know?

- Vocabulary
- DAC R/2N architecture
- ADC Flash, Successive approximation architectures
- PIC18Fxx2 A/D
 - How to configure
 - Acquisition, Conversion time
 - How to start do conversion, read result
- Max517 DAC usage