The University of Texas at Arlington

Lecture 10
DAC: Digital-to-Analog Conversion

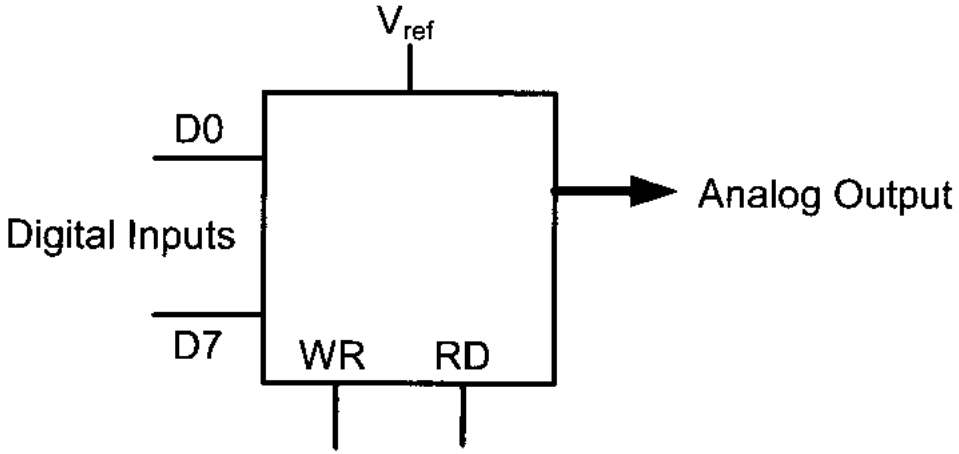
CSE 3442/5442

Embedded Systems 1

Based heavily on slides by Dr. Gergely Záruba and Dr. Roger Walker

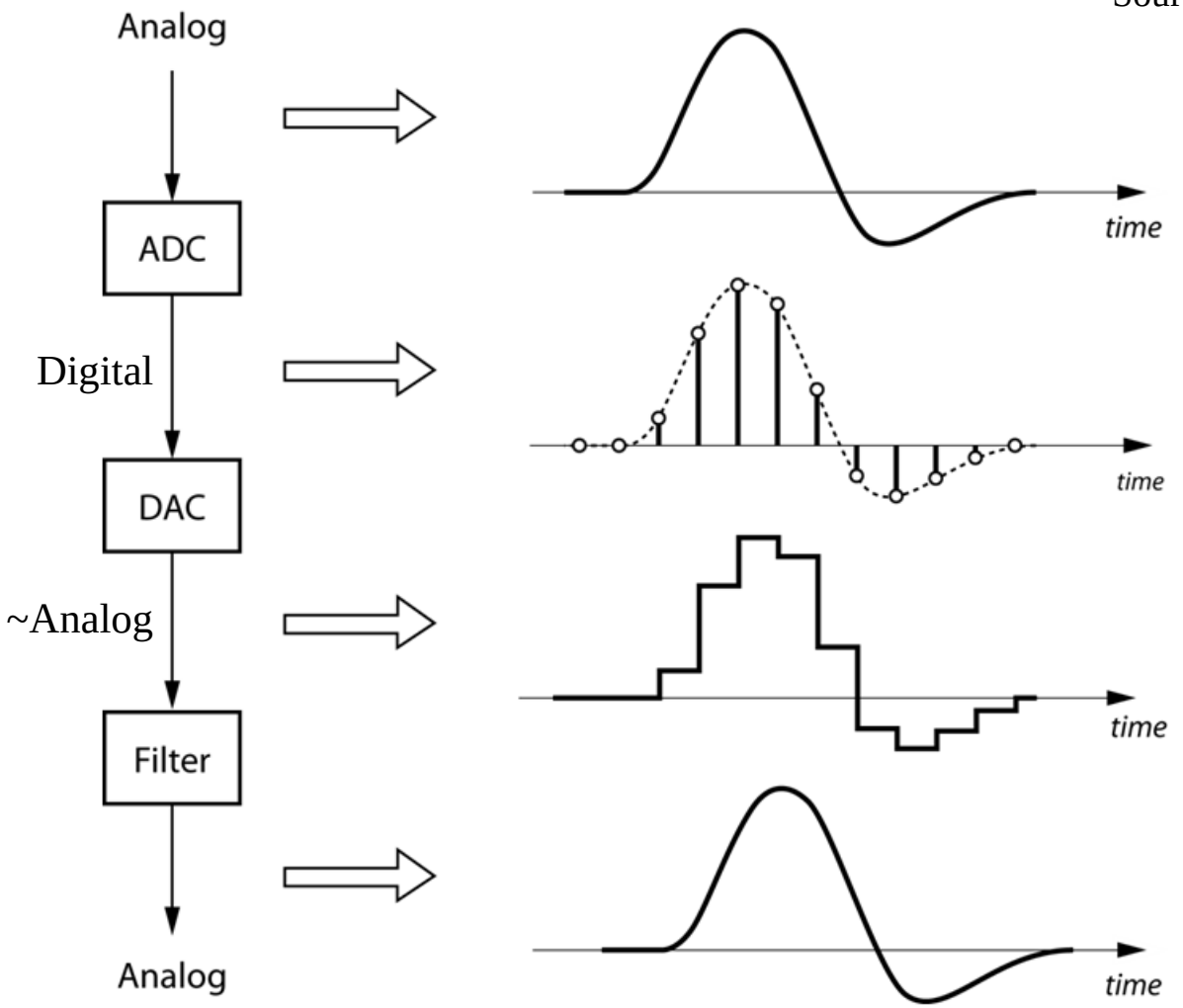
Digital to Analog Conversion

- Unlike an ADC, a DAC takes a **digital value** and converts it into a **quasi-analog voltage**
- Usually the conversion is linear
  - Equal step size
- PIC18s have no built-in DACs (use standalone ICs)
- A DAC can have a **parallel** input or a **serial** input



Reconstructing a Signal

Source: http://www.nutaq.com/

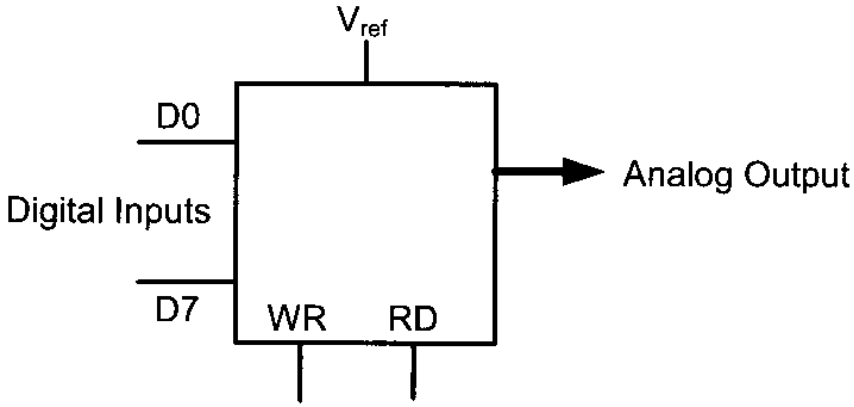


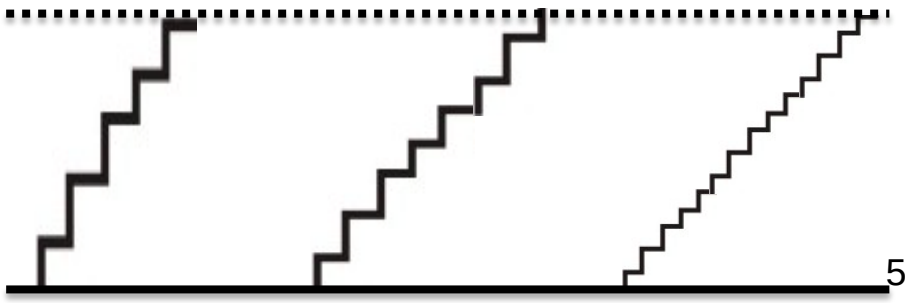
Applications

- **ADC** (sensor input)
  - Audio (microphone)
  - Temperature
  - Weight/Force
  - Light
  - Electricity
  - Air Pressure
  - Water Flow Rate
- **DAC** (actuator output)
  - Audio (speaker)
  - Motor Speed
  - Valve
  - Cooling System
  - Light
  - Video (some)
  - Fuel Injector

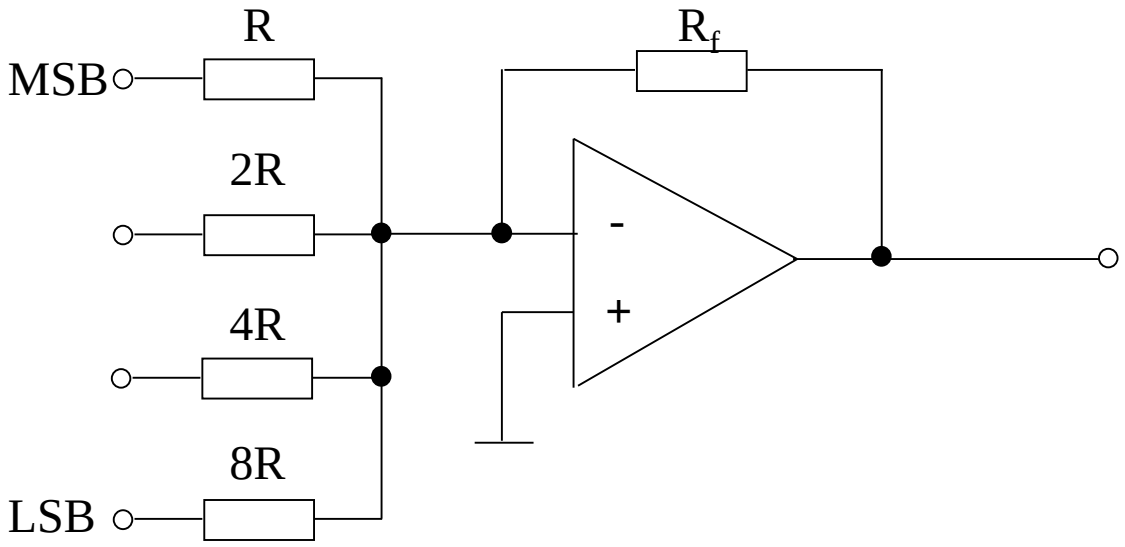
DAC Resolution

- Similar to an ADC's **resolution**
- More bits (inputs) == More precise output
- Most are 8, 10, or 12 bits
  - 8: 256 discrete voltage levels for output
  - 10: 1024 discrete voltage levels for output
  - 12: 4096 discrete voltage levels for output



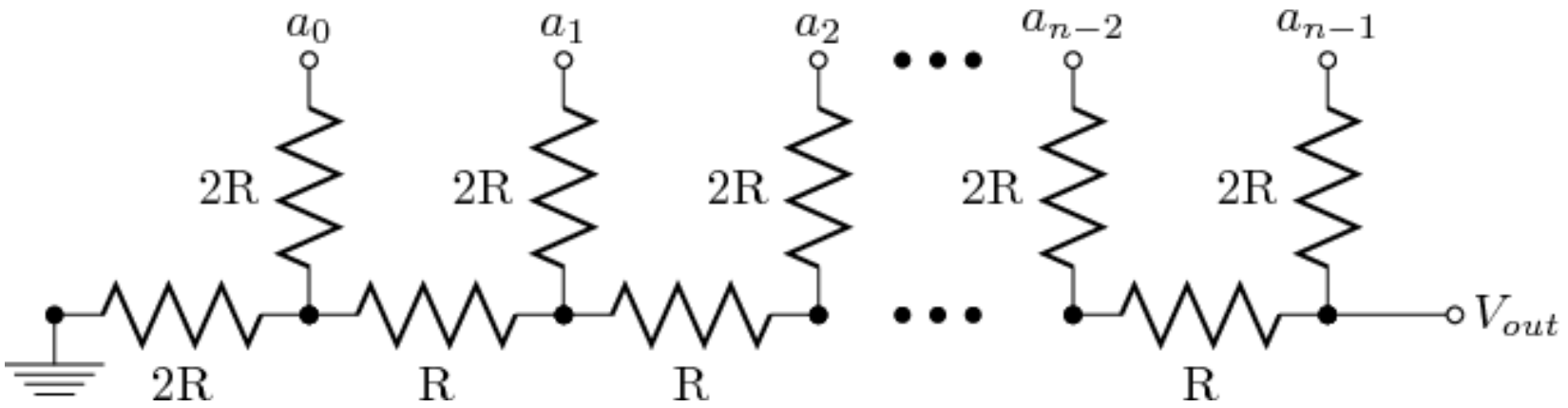


Buffered Binary Resistor Network DAC

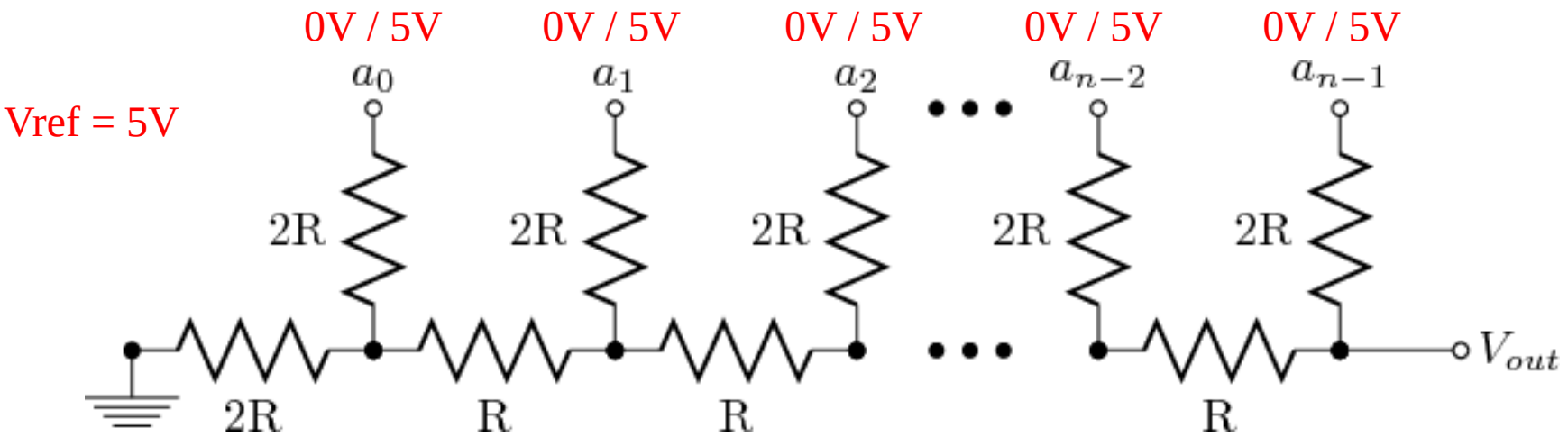


- Seldom used when more than 6 bits
- Consider the design of an 8-bit DAC if the smallest resistor has resistance R
  - what would be the value of the largest resistor?
  - what would be the tolerance of the smallest resistor?
  - input resistors could load the output differently
- Very difficult to manufacture very accurate resistors over this range

R-2R Resistor Ladder DAC

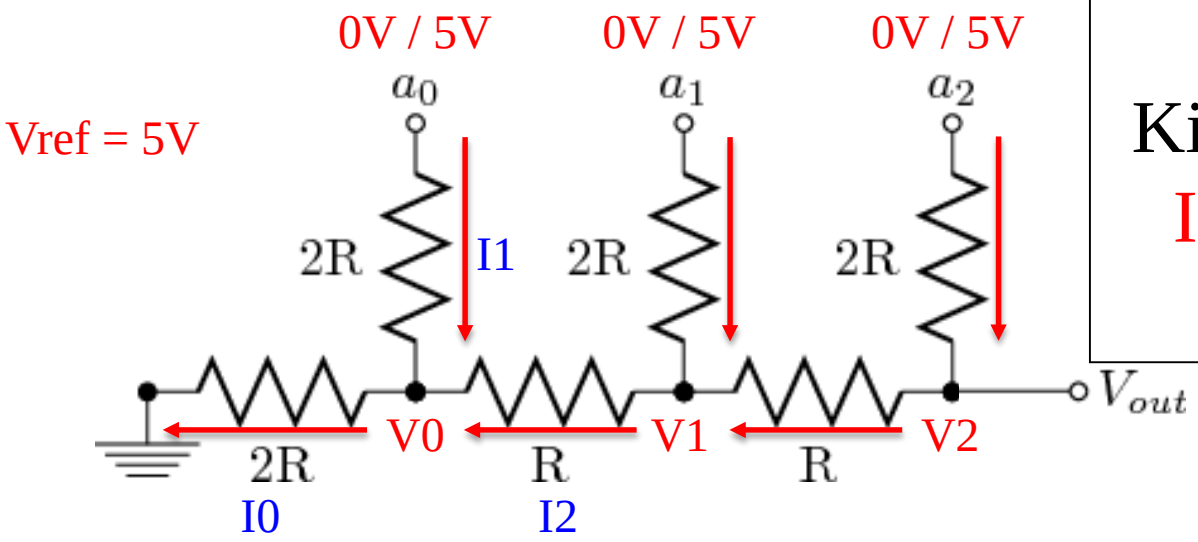


R-2R Resistor Ladder DAC



$$\text{Vout} = \text{Vref} \times \frac{VAL}{2^N}$$

R-2R Resistor Ladder DAC
3-bit Example



Nodal Analysis

Kirchhoff's current law

$I_{in} = I_{out}$ per junction

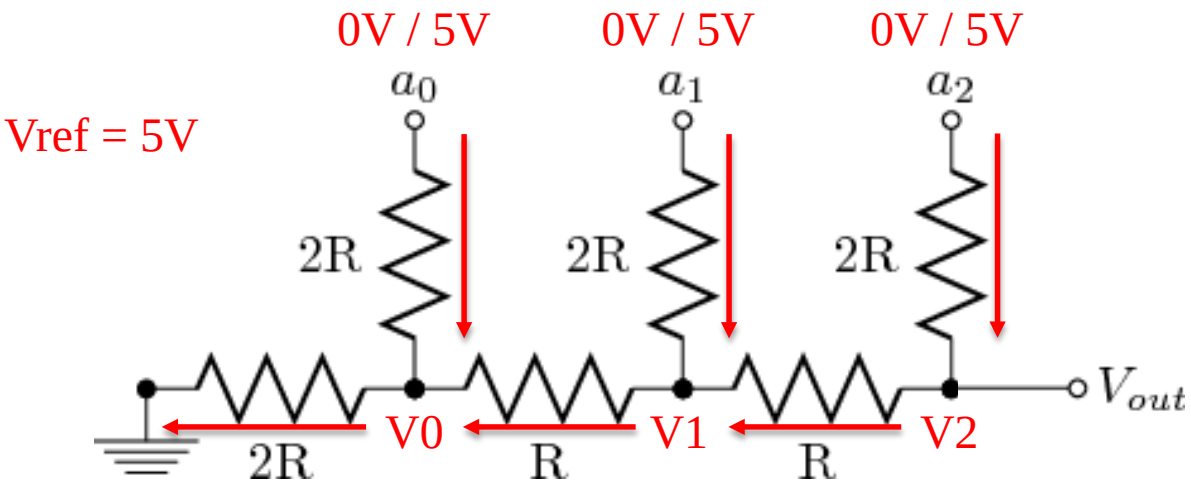
$V = IR$

For V_0: $I_0 = I_1 + I_2$

$$I_0 = \frac{V_0 - 0V}{2R} \qquad I_1 = \frac{a_0 - V_0}{2R} \qquad I_2 = \frac{V_1 - V_0}{R}$$

→ Use $\frac{V_0 - 0V}{2R} = \frac{a_0 - V_0}{2R} + \frac{V_1 - V_0}{R}$ and V_1, V_2 equations to solve for V_2 (V_{out}) with only a_0, a_1, and a_2 left

R-2R Resistor Ladder DAC
3-bit Example



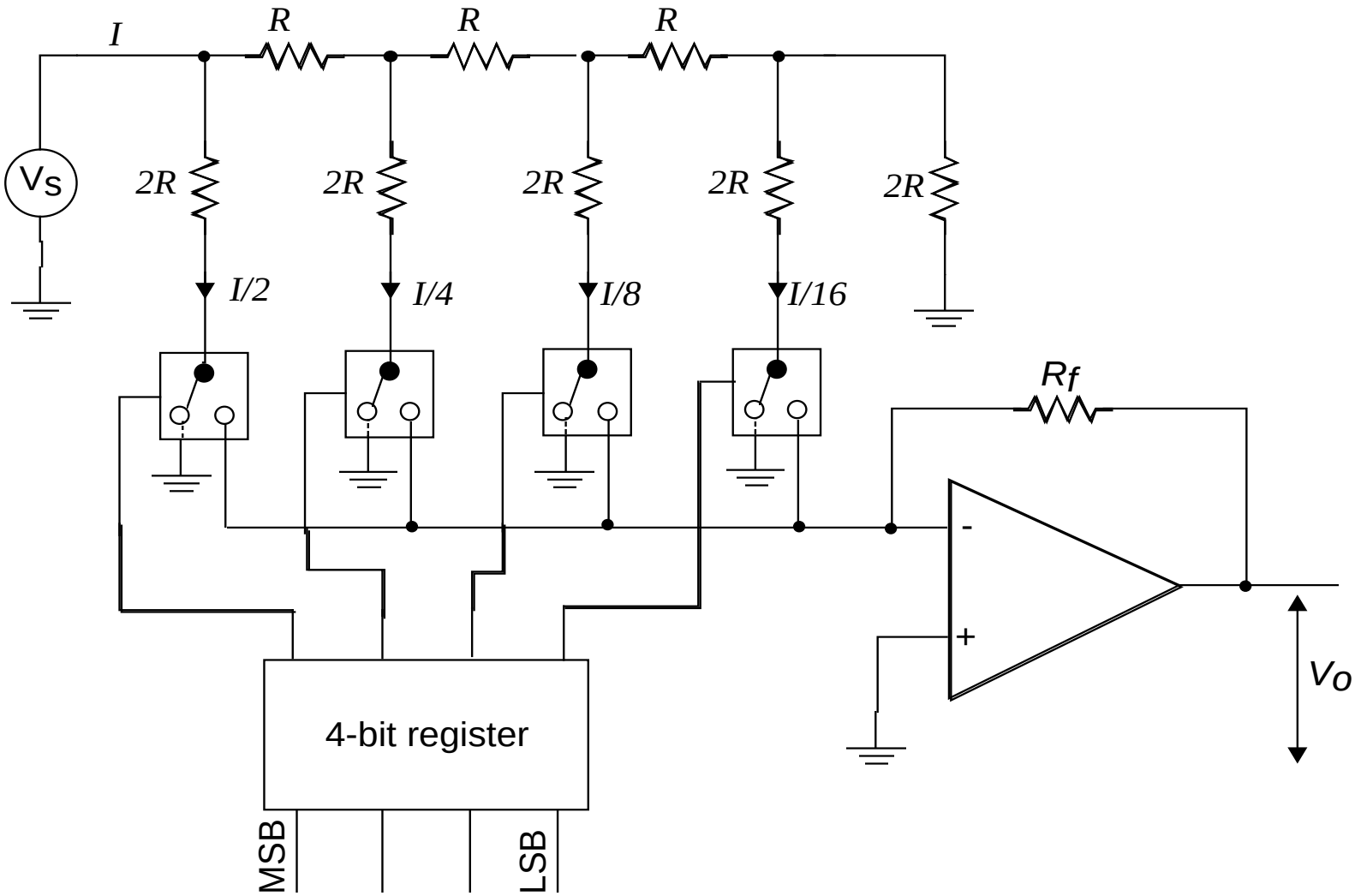
VAL	Vout
000	0.0 V
001	0.625 V
010	1.25 V
011	1.875 V
100	2.5 V
101	3.125 V
110	3.75 V
111	4.375 V

Nodal Analysis

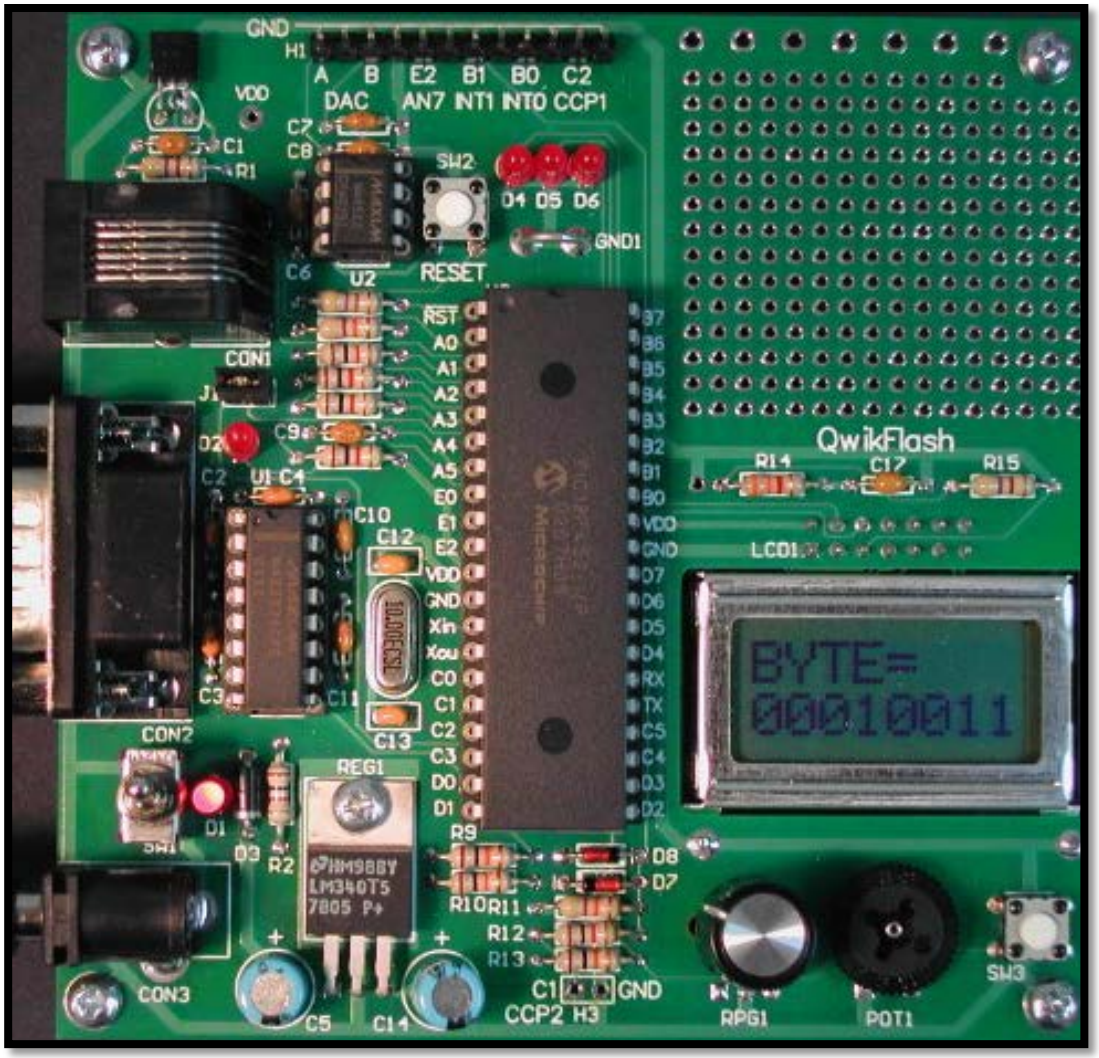
$$\text{Vout} = \frac{1}{8}a_0 + \frac{1}{4}a_1 + \frac{1}{2}a_2$$

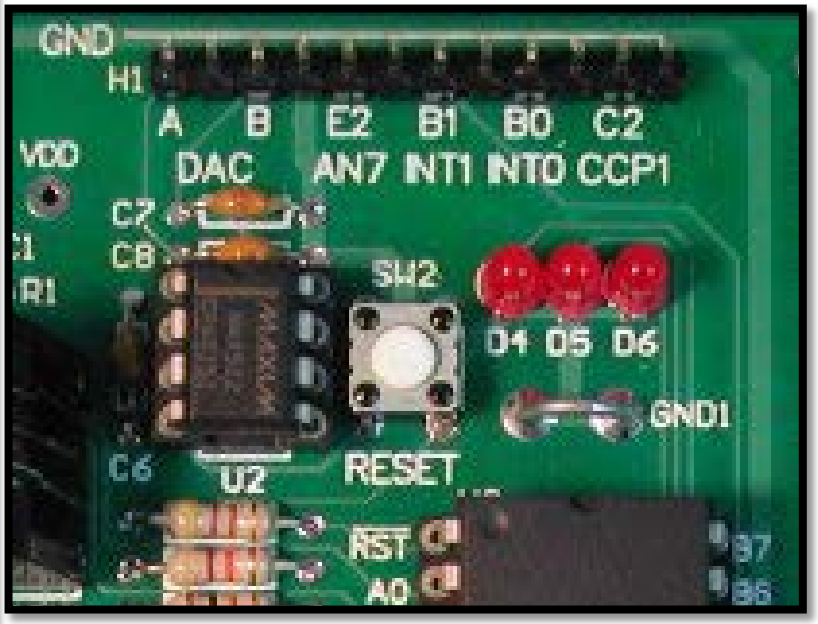
$$\text{Vout} = \text{Vref x } \frac{VAL}{2^N}$$

R-2R Resistor Ladder DAC

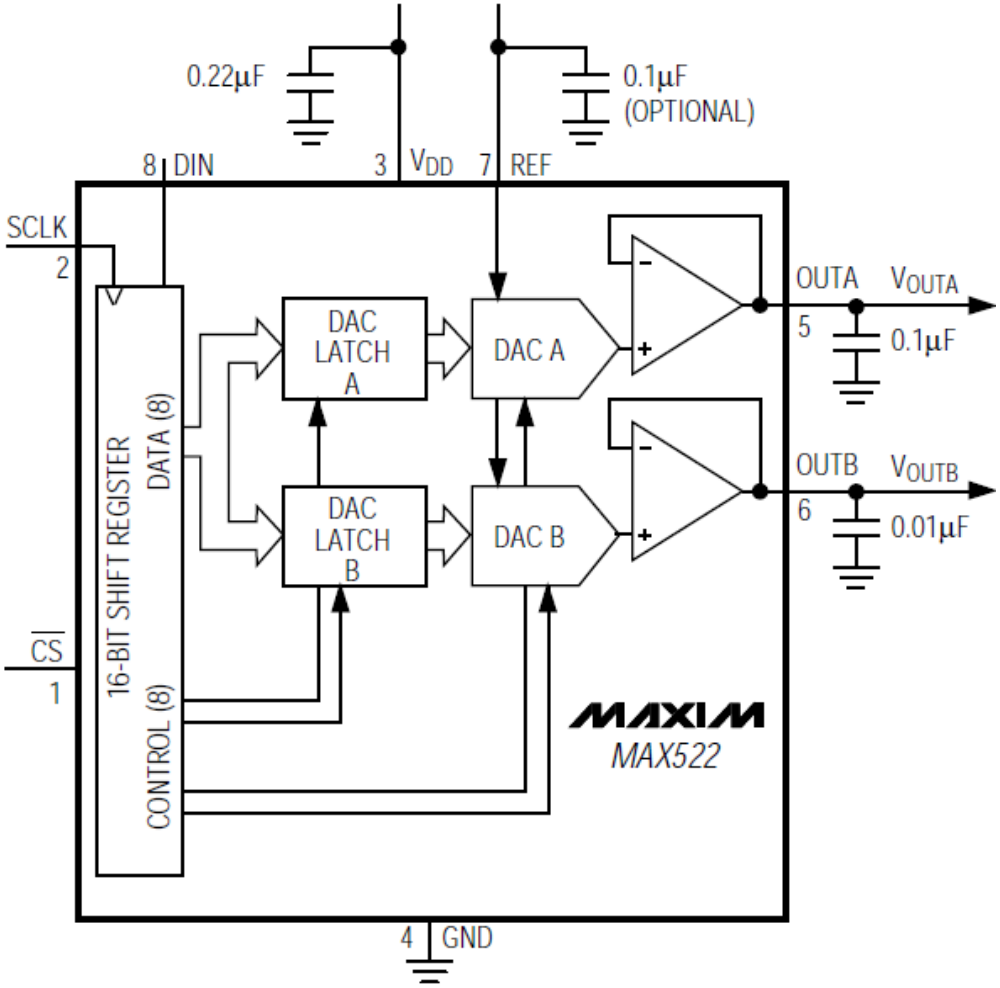


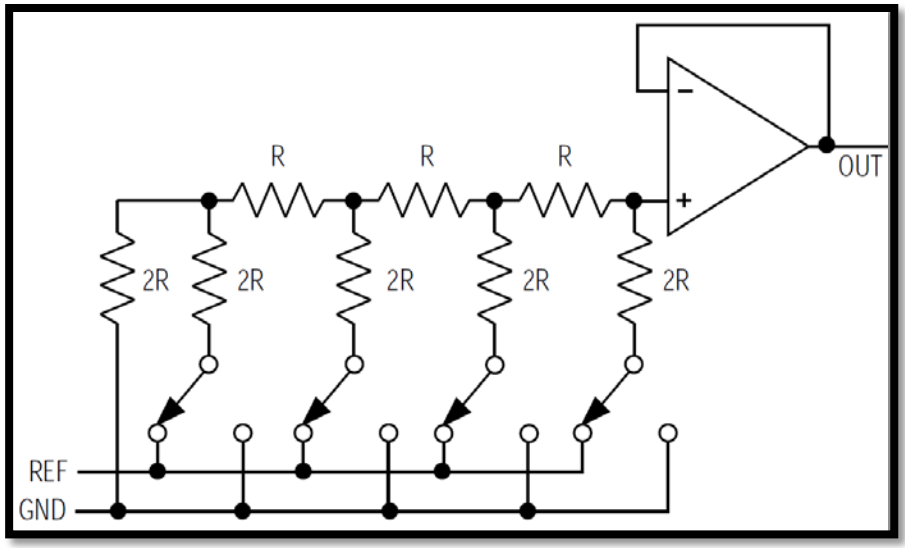
DAC on the QwikFlash



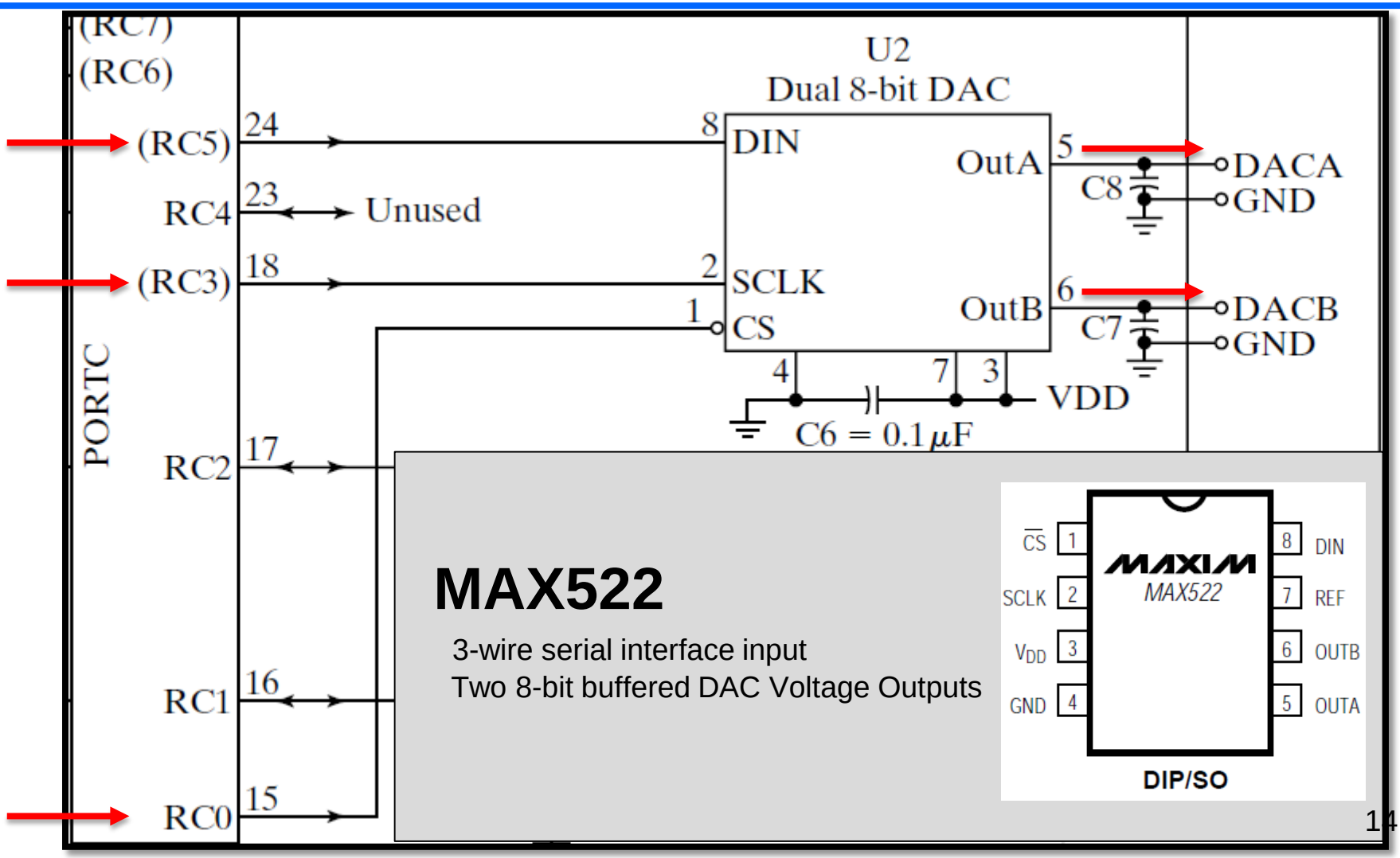


MAX522

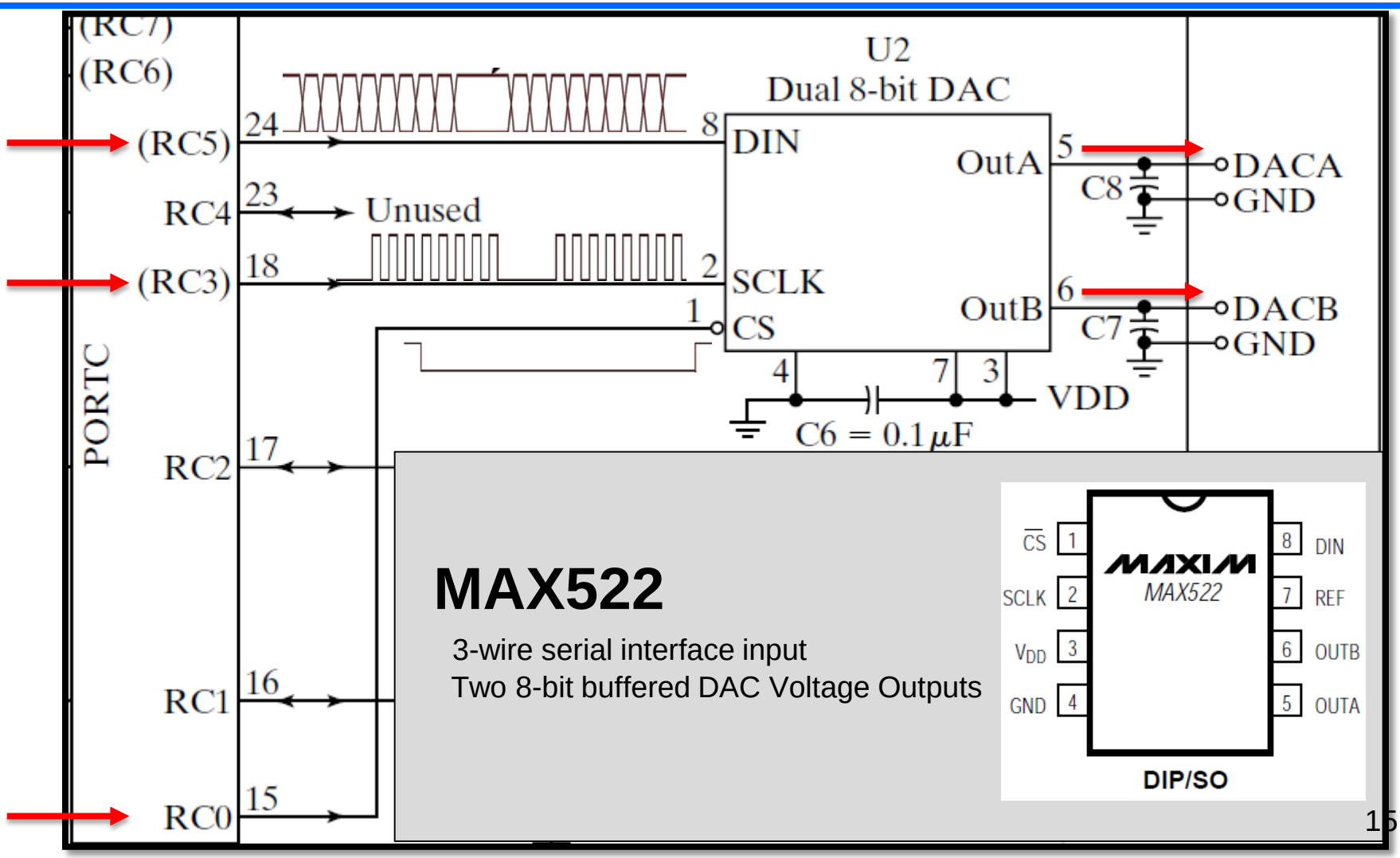

0.22µF
0.1µF
(OPTIONAL)
8 DIN
3 VDD
7 REF
SCLK
2
DAC
LATCH
A
DAC A
OUTA
VOUTA
5
0.1µF
DAC
LATCH
B
DAC B
OUTB
VOUTB
6
0.01µF
16-BIT SHIFT REGISTER
DATA (8)
CONTROL (8)
CS
1
MAXIM
MAX522
4 GND


R
R
R
2R
2R
2R
2R
2R
OUT
REF
GND

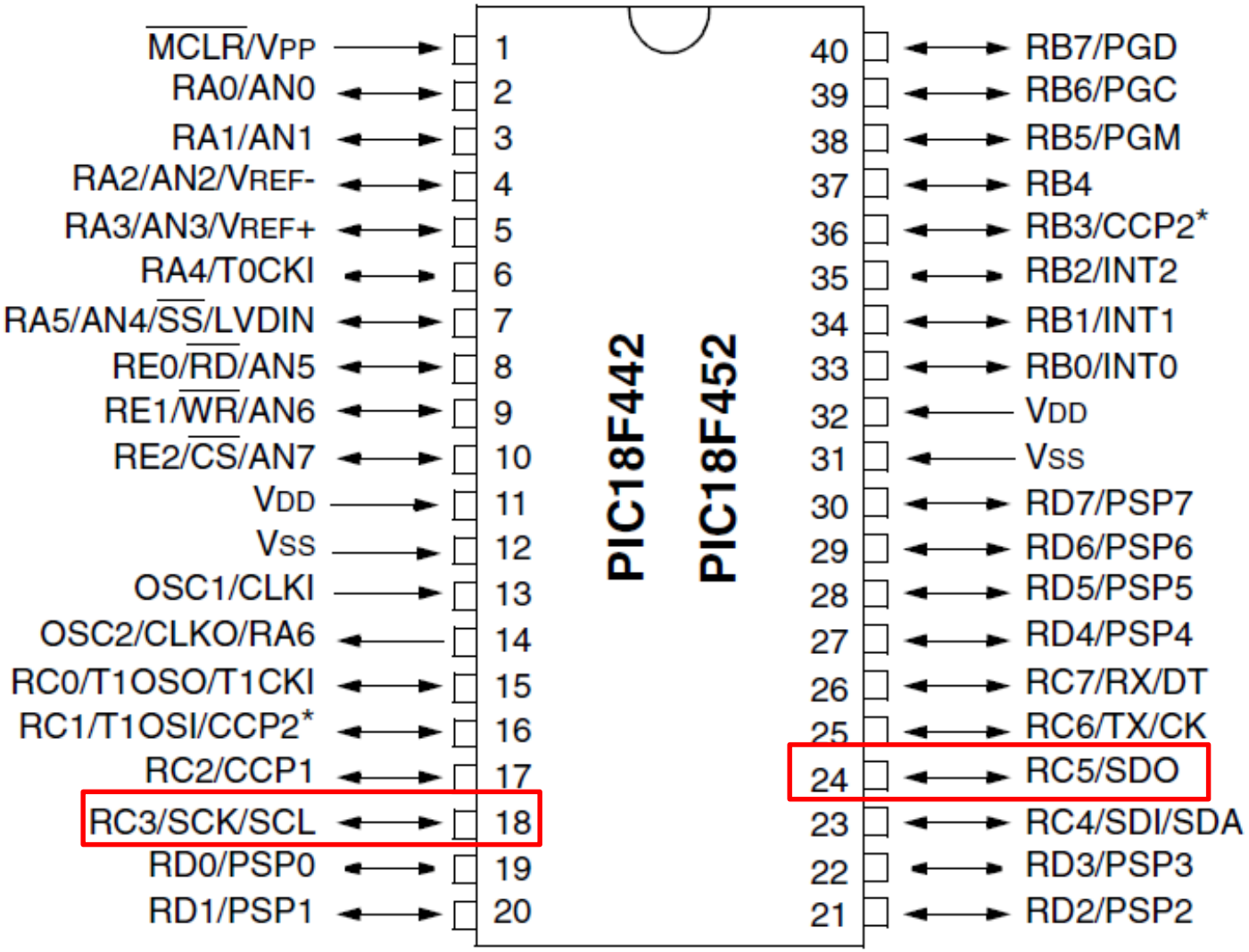
DAC on the QwikFlash



DAC on the QwikFlash



MSSP Module
SPI Connections



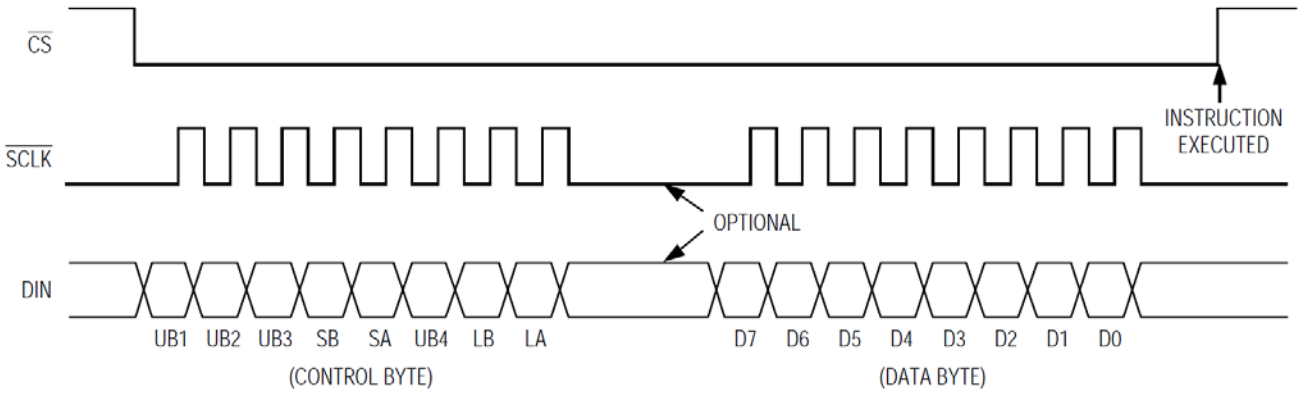
MAX522 DAC

- 3-wire serial Interface
  - !CS: needs to be **kept low** when feeding in data
  - **Rising edge** will modify behavior of DAC
  - SCLK: every **rising edge** data is clocked in (<5MHz)
  - DIN: Data line
- Each command is 16 bits long
- For PIC interfacing, we can program it directly or use serial communication peripherals

DATA BITS	B0*	DAC Data Bit 0 (LSB)
	B1	DAC Data Bit 1
	B2	DAC Data Bit 2
	B3	DAC Data Bit 3
	B4	DAC Data Bit 4
	B5	DAC Data Bit 5
	B6	DAC Data Bit 6
	B7	DAC Data Bit 7 (MSB)
CONTROL BITS	LA	Load Reg DAC A, Active High
	LB	Load Reg DAC B, Active High
	UB4	Uncommitted Bit 4
	SA	Shut Down DAC A, Active High
	SB	Shut Down DAC B, Active High
	UB3	Uncommitted Bit 3
	UB2	Uncommitted Bit 2
	UB1**	Uncommitted Bit 1

* Clocked in last.
**Clocked in first.

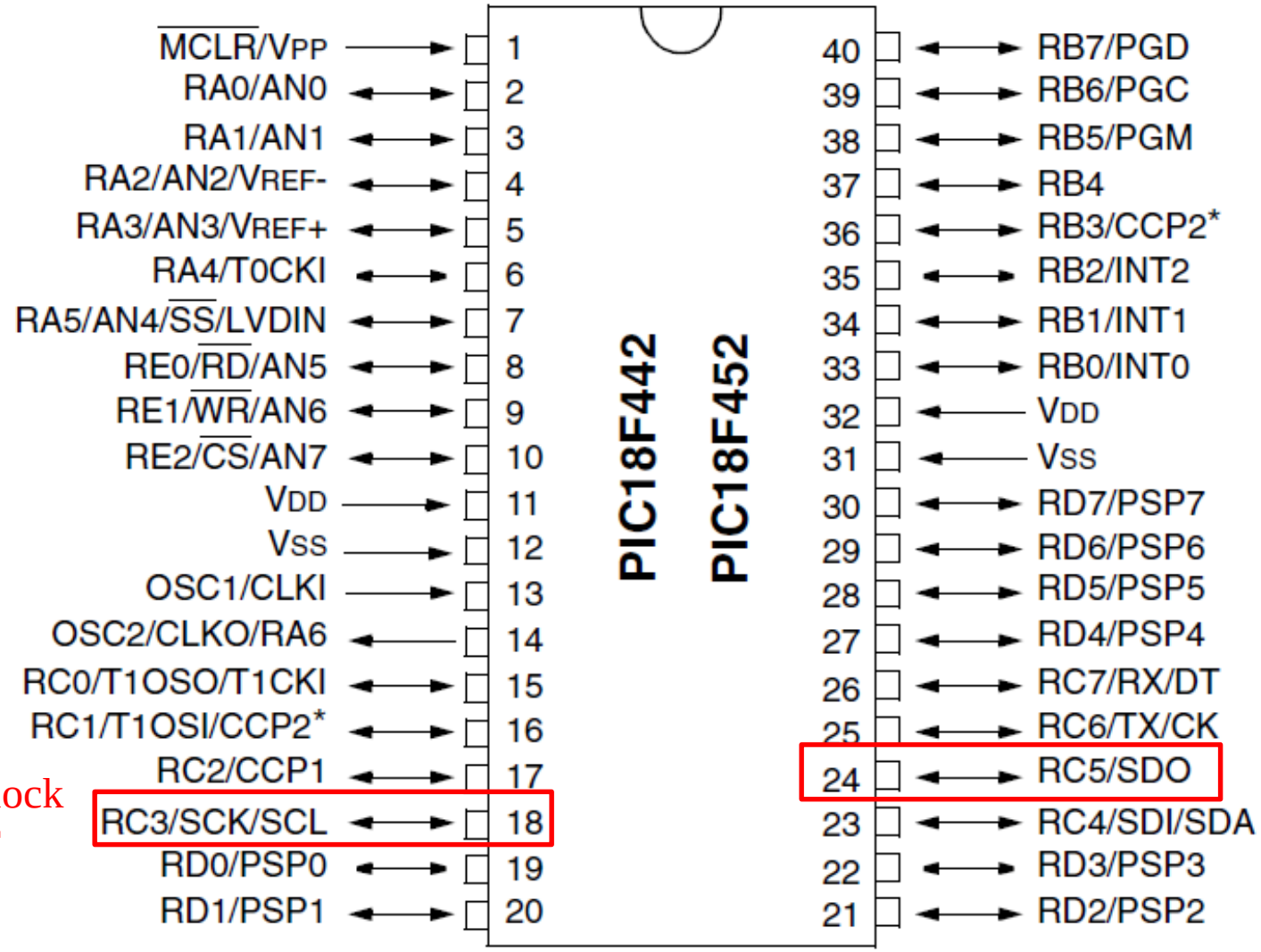
MAX522 DAC



CONTROL								DATA								FUNCTION
UB1	UB2	UB3	SB	SA	UB4	LB	LA	B7 MSB	B6	B5	B4	B3	B2	B1	B0 LSB	
X	X	1	*	*	0	0	0	X	X	X	X	X	X	X	X	No Operation to DAC Registers
X	X	1	*	*	0	0	0									Unassigned Command
X	X	1	*	*	0	1	0	8-Bit DAC Data								Load Register to DAC B
X	X	1	*	*	0	0	1	8-Bit DAC Data								Load Register to DAC A
X	X	1	*	*	0	1	1	8-Bit DAC Data								Load Both DAC Registers
X	X	1	0	0	0	*	*	X	X	X	X	X	X	X	X	All DACs Active
X	X	1	0	0	0	*	*	X	X	X	X	X	X	X	X	Unassigned Command
X	X	1	1	0	0	*	*	X	X	X	X	X	X	X	X	Shut Down DAC B
X	X	1	0	1	0	*	*	X	X	X	X	X	X	X	X	Shut Down DAC A
X	X	1	1	1	0	*	*	X	X	X	X	X	X	X	X	Shut Down All DACs

X = *Don't care.*

MSSP Module
SPI Connections



Serial Clock OUT

Serial Data OUT

SPI Mode Registers

The MSSP module has four registers for SPI mode operation. These are:

- MSSP Control Register1 (SSPCON1)
- MSSP Status Register (SSPSTAT)
- Serial Receive/Transmit Buffer (SSPBUF)
- MSSP Shift Register (SSPSR) - Not directly accessible

REGISTER 15-1: SSPSTAT: MSSP STATUS REGISTER (SPI MODE)

R/W-0	R/W-0	R-0	R-0	R-0	R-0	R-0	R-0
SMP	CKE	D/$\overline{A}$	P	S	R/$\overline{W}$	UA	BF
bit 7							bit 0

bit 7 **SMP:** Sample bit

SPI Master mode:

1 = Input data sampled at end of data output time

0 = Input data sampled at middle of data output time

SPI Slave mode:

SMP must be cleared when SPI is used in Slave mode

bit 6 **CKE:** SPI Clock Edge Select

When CKP = 0:

1 = Data transmitted on rising edge of SCK

0 = Data transmitted on falling edge of SCK

When CKP = 1:

1 = Data transmitted on falling edge of SCK

0 = Data transmitted on rising edge of SCK

bit 5 **D/$\overline{A}$:** Data/$\overline{Address}$ bit

Used in I^2C mode only

bit 4 **P:** STOP bit

Used in I^2C mode only. This bit is cleared when the MSSP module is disabled, SSPEN is cleared.

bit 3 **S:** START bit

Used in I^2C mode only

bit 2 **R/$\overline{W}$:** Read/Write bit information

Used in I^2C mode only

bit 1 **UA:** Update Address

Used in I^2C mode only

bit 0 **BF:** Buffer Full Status bit (Receive mode only)

1 = Receive complete, SSPBUF is full

0 = Receive not complete, SSPBUF is empty

REGISTER 15-2: SSPCON1: MSSP CONTROL REGISTER1 (SPI MODE)

R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
WCOL	SSPOV	SSPEN	CKP	SSPM3	SSPM2	SSPM1	SSPM0
bit 7							bit 0

bit 7 **WCOL:** Write Collision Detect bit (Transmit mode only)

1 = The SSPBUF register is written while it is still transmitting the previous word (must be cleared in software)

0 = No collision

bit 6 **SSPOV:** Receive Overflow Indicator bit

SPI Slave mode:

1 = A new byte is received while the SSPBUF register is still holding the previous data. In case of overflow, the data in SSPSR is lost. Overflow can only occur in Slave mode.The user must read the SSPBUF, even if only transmitting data, to avoid setting overflow (must be cleared in software).

0 = No overflow

Note: In Master mode, the overflow bit is not set since each new reception (and transmission) is initiated by writing to the SSPBUF register.

bit 5 **SSPEN:** Synchronous Serial Port Enable bit

1 = Enables serial port and configures SCK, SDO, SDI, and $\overline{SS}$ as serial port pins

0 = Disables serial port and configures these pins as I/O port pins

Note: When enabled, these pins must be properly configured as input or output.

bit 4 **CKP:** Clock Polarity Select bit

1 = IDLE state for clock is a high level

0 = IDLE state for clock is a low level

bit 3-0 **SSPM3:SSPM0:** Synchronous Serial Port Mode Select bits

0101 = SPI Slave mode, clock = SCK pin, $\overline{SS}$ pin control disabled, $\overline{SS}$ can be used as I/O pin

0100 = SPI Slave mode, clock = SCK pin, $\overline{SS}$ pin control enabled

0011 = SPI Master mode, clock = TMR2 output/2

0010 = SPI Master mode, clock = Fosc/64

0001 = SPI Master mode, clock = Fosc/16

0000 = SPI Master mode, clock = Fosc/4

Example – MAX522

```
unsigned char controlByte= 0;
unsigned char dataByte = 0;
SSPSTAT = 0bxxxxxxxx;  //SMP and CKE
SSPCON1 = 0bxxxxxxxx; //Enable SPI serial port

while(1)
{
    //calculations…, assign value to dataByte (8-bit data you want to send out)
    //Enable DAC (Chip Select pin on DAC)

    SSPBUF = controlByte; //MAX522 Control bits to select which output to use
     while(SSPSTATbits.BF == 0); //wait until transmission is complete

     SSPBUF = dataByte; //Assign Voltage value to DAC via buffer
     while(SSPSTATbits.BF == 0); //wait until transmission is complete

      //Disable DAC (Chip Select pin on DAC)
}
```

Questions?