Embedded Systems Laboratory manual

edited by Zsolt Molnar

Óbuda University

Kálmán Kandó Faculty of Electrical Engineering

2015.

Contents

Preface

This manual is written for the Embedded Systems Laboratory, Kálmán Kandó Faculty of Electrical Engineering, Óbuda University. The course tries to cover the great part of the topics of higher level electronics. The term 'embedded system' is interpreted more widely than it is usual. The topics of the experiments cover the supplementary electronics for embedded systems, the analog and digital areas too. As a continuation of this course, the Signal and Image Processing Laboratory will help the students to get a deeper insight into development of the microcontroller, DSP and FPGA design.

The topics of this manual discuss partly analog circuits, as operational amplifiers, active filters, analog switches, voltage regulators, analog control loops. These circuits are the integral parts of the complex embedded systems, just like the mixed circuits as the digital to analog and analog to digital converters. The microcontroller simulation topic is an introduction to higher level microcontroller, DSP and FPGA design.

In the experiments there are not only the tasks for experiments, but there are the summation of the minimum required theoretical knowledge, examples, questions for preparing to the written tests, and links to help to extend your knowledge. We included the necessary data sheet pieces, so the student don't have to spend time from the short course time searching data sheets. It is important to note, that to get wider and complex knowledge in the topics of this manual, it is necessary to gather further knowledge from books, from the Internet, and from further, individually conducted experiments.

And last, but not least, I want to say special thanks to Dr. Joseph Kohut, who motivated me, helped me lots of advices, and has undertook the revision of this manual.

July 22, 2015 Zsolt Molnar

Experiment 1.

Stability and frequency compensation of operational amplifiers

Objectives

- To learn how to maintain the stability of the operational amplifiers with external compensation.
- To learn the measure of low- and high-level signal performance parameters of amplifiers.

Components

Component	Description	Number of items	Comment
uA748	operational amplifier	1	or equivalent (e.g. LM748)
1 kΩ	resistor	2	
10 kΩ	resistor	1	
2 pF	capacitor	1	ceramic
3 pF	capacitor	1	ceramic
5 pF	capacitor	1	ceramic
10 pF	capacitor	1	ceramic
30 pF	capacitor	1	ceramic
100 nF	capacitor	2	ceramic, optional (if needed)

Instruments

Items	Comment
Power supply	dual
Digital multimeter	
Function generator	
Digital oscilloscope	dual channel

Background

The operational amplifier

The operational amplifier (opamp or OPA) is a DC-coupled high-gain electronic voltage amplifier with a differential input and, usually, a single-ended output. As an electronic device, operational amplifier is an integrated circuit, which contains multiple gain stages. An ideal op-amp is usually considered to have the following properties:

- infinite open-loop gain (A_O)
- infinite input impedance (R_{IN}), and so zero input current
- zero input offset voltage (V_O)
- infinite voltage range available at the output
- infinite bandwidth with zero phase shift and infinite slew rate (SR)
- zero output impedance (R_{OUT})
- zero noise
- infinite Common-mode rejection ratio (CMRR)
- infinite Power supply rejection ratio.

Naturally, the real operational amplifiers have no ideal parameters. Some examples of the parameters above in the data sheet of LM741:

- A_O ~ 50.000 (vs. ∞)
- R_{IN} ~ 6 MΩ (vs. ∞)
- V_O ~ ±10 mV (vs. 0)
- Output voltage swing: ~±12 V @ V_{SUPPLY} = ±15 V (vs. infinite voltage range at the output)
- …

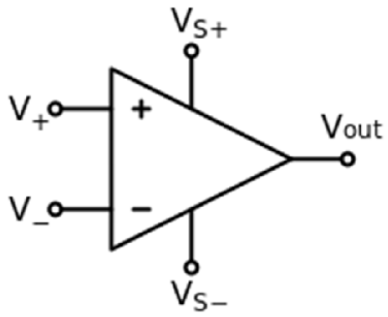


Figure 1. Electronic symbol of the operational amplifier: Vs+ and Vs- are the supply voltages, '+' is the non-inverting, '-' is the inverting input, on the right side is the output of the circuit

In amplifiers built with operational amplifiers, we use negative feedback.

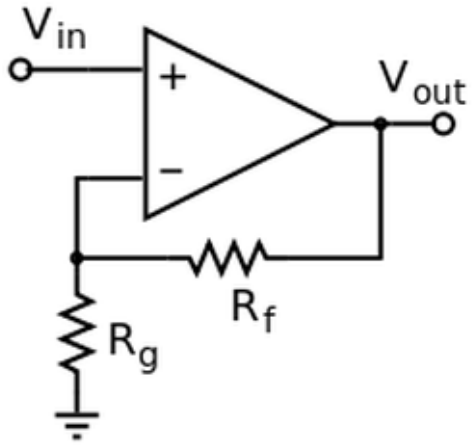


Figure 2. Typical negative feedback example: we are feeding back a proportional amount of V_{OUT} to the inverting input.

Frequency compensation

There is no stability problem (system is stable), if the internal amplifier stages of operational amplifier and the feedback circuit not causes more than 180° of phase shift, because the nature of the feedback remains negative. If the nature of feedback becomes positive, the noise of the circuit will be amplified, and will be positively fed back to the input, which is a self-increasing process.

Note 1: If the voltage amplification (A_V) of amplifier circuit is less than 1, there is no stability problem at any phase shift, this phenomenon occurs only, if the amplification is greater or equal than 1. So in the next, we examine (and discuss) only that cases, when $A_V \geq 1$.)

The stability decreases, when the phase shift of the amplifier closes to 180° (or -180°), and the oscillating tendency increases. The difference between -180° and the actual phase shift at the gain of 1, is the phase margin (Φ_m). Rule of thumb: minimum 45° of phase margin required for stabile operation at the gain of 1. It is essential to maintain $\Phi_m > 45°$, and we can use several techniques to ensure this.

Most operational amplifiers have internal frequency compensation, so these amplifiers are stable at any voltage amplification and at any frequency, if the external feedback circuitry has no significant phase shift. But there are many operational amplifiers, which have no internal compensation, these are indicated as 'with external frequency compensation'. Although the internally compensated amplifiers are easy to use, the externally compensated amplifiers allow to design an amplifier stage with optimal parameters, e.g. higher cut-off frequency, higher slew rate, or lower rise time.

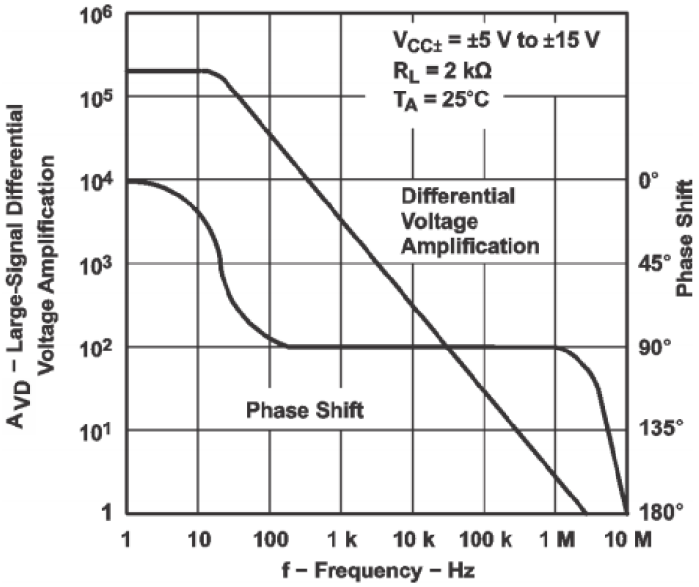


Figure 3. Large signal differential voltage amplification and phase shift vs frequency at data sheet of internally compensated type TL071 opamp

On the figure above, we can see that the pole at about 15 Hz causes 90° phase shift at the frequency about tenfold of the frequency of the pole. Each single pole causes 90° phase shift, so, if there is a second pole, at the frequency tenfold of the frequency of the second pole the total phase shift will be about -180°. If at this frequency the $A_{VD} \geq 1$, there will be oscillation. (At this operational amplifier $\Phi_m \sim 75°$, because the phase shift at $A_{VD} = 1$ is ~105°, and $\Phi_m \sim 180° - 105° = 75°$.)

On the Figure 4, at gain of 1 (at $20 \cdot \log(A\beta) = 0\ dB$) the phase shift is -155°, the $\Phi_m = 180° - 155° = 25°$, this value is less than the minimum required 45° so the amplifier will be unstable. If we create a new dominant pole in the system at ω_D, the Bode-diagram will be modified as it seen on the Figure 5. The amplifier is now stable due to the increased phase margin. This method is one of the possible methods of frequency compensation, called dominant pole compensation. It can be simply done by placing one external capacitor (called compensating capacitor) between the specified pins of integrated opamps without internal frequency compensation. This capacitor modifies the Miller-capacitance of a gain stage, so modifies the frequency-behavior of the amplifier.

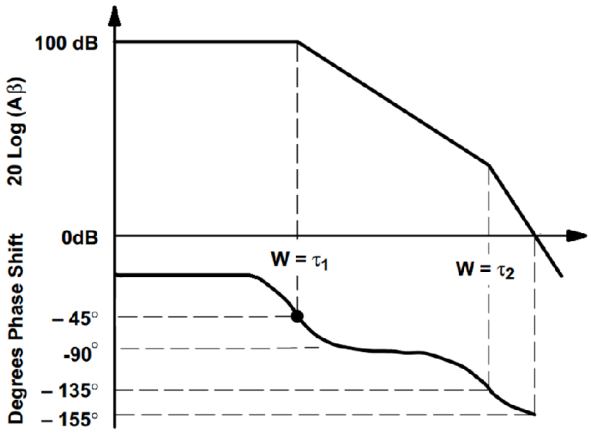


Figure 4. Possible Bode-plot of an unstable amplifier

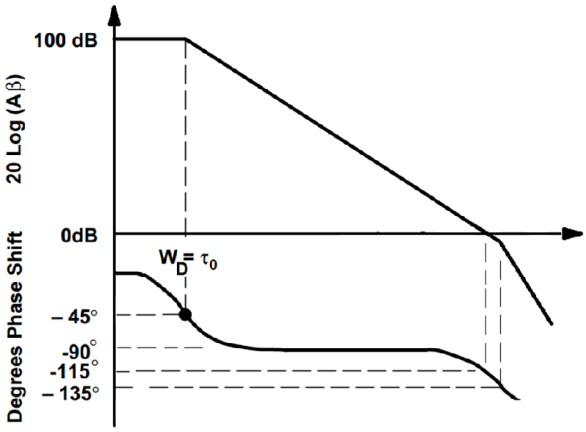


Figure 5. Modified Bode-plot with a transferred dominant pole at ω_D

Amplifier signal performance parameters

One part of the performance parameters below must be measured at low level, and the other part of them must be measured at high level. It is important to measure these parameters at correct voltage level!

'Low-level' means that the output level is significantly lower than the supply voltages, so the output level in this case must be lower than 1 V_{pp}. (We can measure the low-level parameters at e. g. 10 mV_{pp} level, but the noise will make more difficult to specify them.)

'High-level' means, that the output level is near to the supply voltages, so the output level in this case must be about 20 V_{pp}. We don't suppose to exceed this value, because the maximum output swing is about ±12 V (24 V_{pp}). You can measure at slightly lower value, but the minimum is 10 V_{pp}.

Note, in the laboratory we usually use ±15 V supply voltage, so this value was considered at the definitions of 'low-level' and 'high-level'.

High-level parameters

Slew Rate (SR)

The slew rate is the maximum possible rate of change of the operation amplifier output voltage.

$$SR = \max\left|\frac{dV_{out}(t)}{dt}\right| \left[\frac{V}{s}\right].$$

(The dimension is usually $\left[\frac{V}{\mu s}\right]$ for practical reasons.)

This parameter must be measured with square wave input signal. On the output response signal we choose a linear section, then measure its horizontal (Δt) and vertical (ΔV_{out}) projection.

$$SR = \frac{\Delta V_{out}}{\Delta t}$$

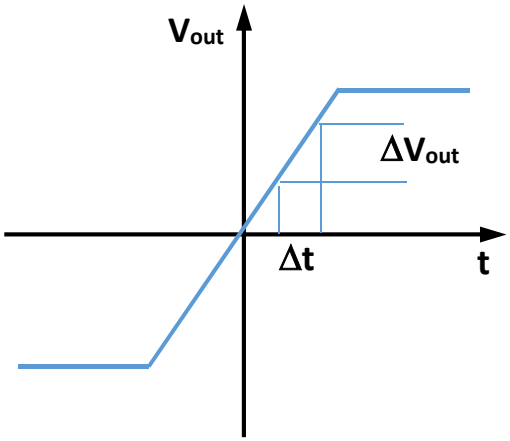


Figure 6. Slew rate measurement

Full power bandwidth (f_P)

The operational amplifier's slew-rate limiting can cause nonlinear distortion in sinusoidal waveforms. When this nonlinear distortion appears, the shape of the output signal will change. This parameter must be measured with sinusoidal input signal.

At a given output level we increase the input signal frequency from a relatively low value (e.g. 1 kHz), to that value, where the shape of the output signal changes. This frequency is the f_P, full power bandwidth. The measurement of f_P recommended to repeat several times, because this method is very subjective. (E. g. at V_{out} = 20 V_{pp} and C_C = 3 pF one measures 85 kHz, other 105 kHz at same amplifier, because difficult to realize the slight changes of sinusoidal signal's shape.)

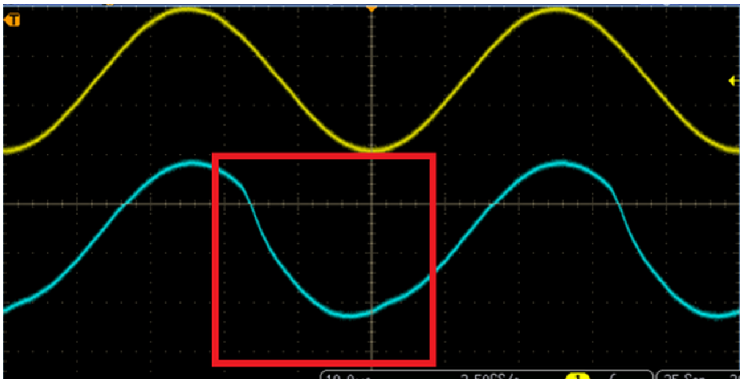


Figure 7. Full power bandwidth measurement: the shape of the output signal changes at and over f_P

In the data sheet, we can find the diagram of large signal frequency response. If at a given frequency we want to determine the maximum undistorted output swing, or inversely, at a given output swing we want to determine the maximum frequency of undistorted signal transmission, we can use this diagram. E.g. at 100 kHz and C_C (C_1) = 3 pF the maximum undistorted output swing is about ±7 V.

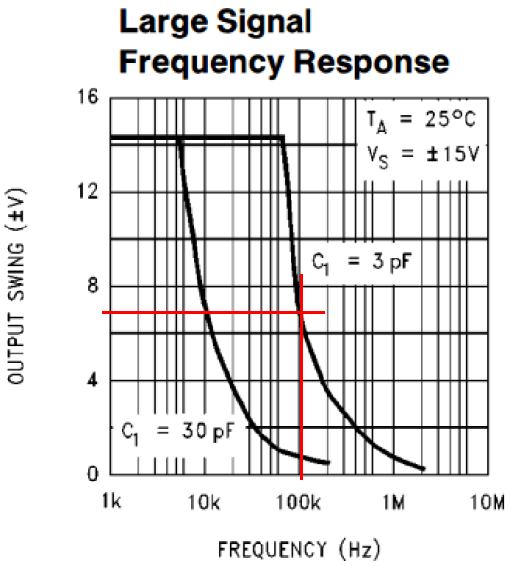


Figure 8. Determining maximal undistorted output swing

Low-level parameters

Rise and fall time (t_r, t_f)

The output response to a voltage step at the input will be not the same shape as the input signal. Rise time is the time taken by a signal to change from a specified low value to a specified high value. Fall time is the time taken by a signal to change from a specified high value to a specified low value. Typically, in analog electronics, these values are 10% and 90% of the step height. It is important to note, that the values must be determined from the steady states! This parameter must be measured with square wave input signal.

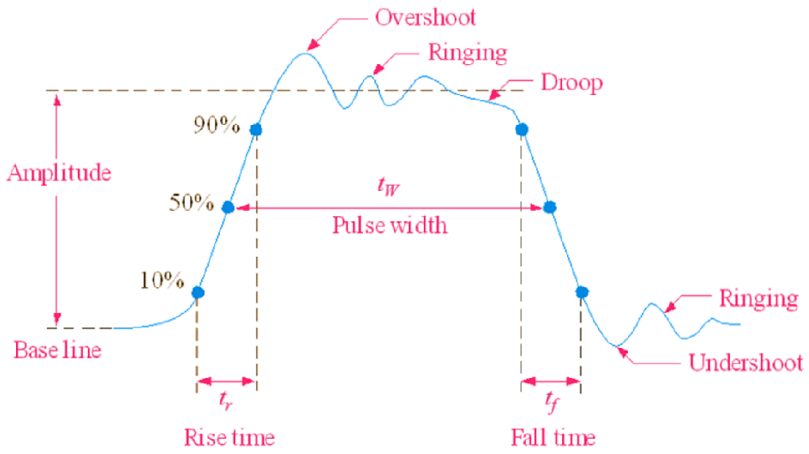


Figure 9. General output pulse response and pulse parameters

Cutoff frequency (f_C)

The cutoff frequency (or corner frequency) is the frequency either above or below which the voltage output of a circuit has fallen to a given proportion of the voltage in the passband. Typically this proportion is the 70.7% of passband voltage (-3 dB attenuation). This parameter must be measured with sinusoidal input signal.

Note, the DC amplifiers, as the amplifiers in this experiment, have only high cutoff frequency. Low cutoff frequency is 0 Hz (DC).

We can measure the cutoff frequency as following: we need to change the signal frequency from the passband, to that value, where the output level falls to the 70.7% of passband voltage. (Because of the 0 Hz low cutoff frequency of the currently examined amplifier, we can start at arbitrarily low frequency, e. g. 100 Hz.)

(Recording the Bode-diagram of a circuit, including determining the cutoff frequency, will be more precisely described in the Experiment 2.)

Extract of the data sheet of the operational amplifier type uA748

General Description

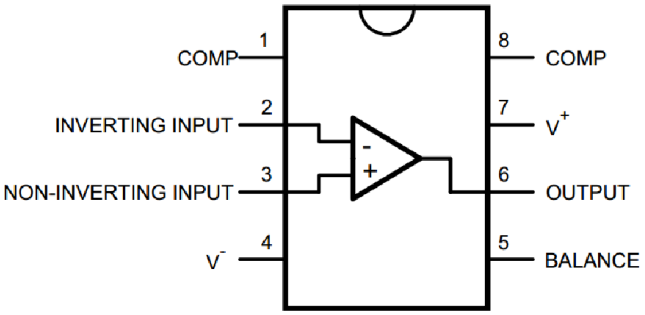
The LM748 is a general purpose operational amplifier with external frequency compensation. The unity-gain compensation specified makes the circuit stable for all feedback configurations, even with capacitive loads. It is possible to optimize compensation for best high frequency performance at any gain. As a comparator, the output can be clamped at any desired level to make it compatible with logic circuits. The LM748C is specified for operation over the 0 °C to 70 °C temperature range.

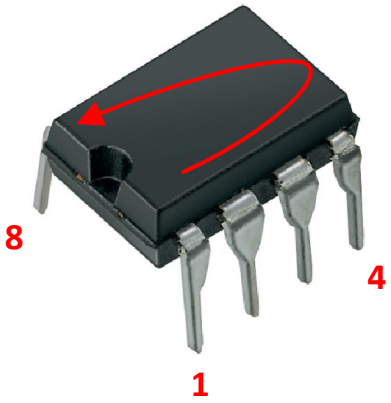
Features

- Frequency compensation with a single 30 pF capacitor
- Operation from ±5 V to ±20 V
- Continuous short-circuit protection
- Operation as a comparator with differential inputs as high as ±30 V
- No latch-up when common mode range is exceeded

Connection Diagram

Dual-In-Line (DIL) package





Absolute Maximum Ratings

If Military/Aerospace specified devices are required, please contact the National Semiconductor Sales Office/Distributors for availability and specifications.

Supply Voltage	±22V
Power Dissipation (Note 1)	500 mW
Differential Input Voltage	±30V
Input Voltage (Note 2)	±15V
Output Short-Circuit Duration (Note 3)	
Operating Temperature Range: LM748C	0°C to +70C
Storage Temperature Range	−65°C to +150°C
Lead Temperature (Soldering, 10 sec.)	+300°C

Electrical Characteristics (Note 4)

Parameter	Conditions	Min	Typ	Max	Units
Input Offset Voltage	$T_A = 25°C$, $R_S \leq 10\ k\Omega$		1.0	5.0	mV
Input Offset Current	$T_A = 25°C$		40	200	nA
Input Bias Current	$T_A = 25°C$		120	500	nA
Input Resistance	$T_A = 25°C$	300	800		kΩ
Supply Current	$T_A = 25°C$, $V_S = \pm 15V$		1.8	2.8	mA
Large Signal Voltage Gain	$T_A = 25°C$, $V_S = \pm 15V$ $V_{OUT} = \pm 10V$, $R_L \geq 2\ k\Omega$	50	160		V/mV
Input Offset Voltage	$R_S \leq 10\ k\Omega$			6.0	mV
Average Temperature Coefficient of Input Offset Voltage	$R_S \leq 50\Omega$		3.0		µV/°C
	$R_S \leq 10\ k\Omega$		6.0		µV/°C
Input Offset Current	$T_A = 0°C$ to $+70°C$			300	nA
	$T_A = -55°C$ to $+125°C$			500	nA
Input Bias Current	$T_A = 0°C$ to $+70°C$			0.8	µA
	$T_A = -55°C$ to $+125°C$			1.5	µA
Supply Current	$T_A = +125°C$, $V_S = \pm 15V$		1.2	2.25	mA
	$T_A = -55°C$ to $+125°C$		1.9	3.3	mA
Large Signal Voltage Gain	$V_S = \pm 15V$, $V_{OUT} = \pm 10V$ $R_L \geq 2\ k\Omega$	25			V/mV
Output Voltage Swing	$V_S = \pm 15V$, $R_L = 10\ k\Omega$	±12	±14		V
	$V_S = \pm 15V$, $R_L = 2\ k\Omega$	±10	±13		V
Input Voltage Range	$V_S = \pm 15V$	±12			V
Common-Mode Rejection Ratio	$R_S \leq 10\ k\Omega$	70	90		dB
Supply Voltage Rejection Ratio	$R_S \leq 10\ k\Omega$	77	90		dB

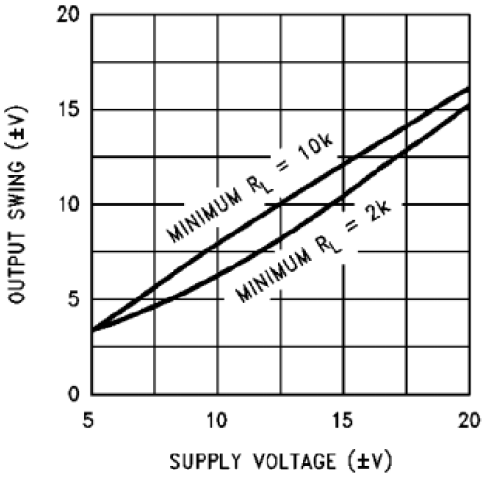
Note 1: For operating at elevated temperatures, the device must be derated based on a maximum junction to case thermal resistance of 45°C per watt, or 150°C per watt junction to ambient. (See Curves).

Note 2: For supply voltages less than ±15V, the absolute maximum input voltage is equal to the supply voltage.

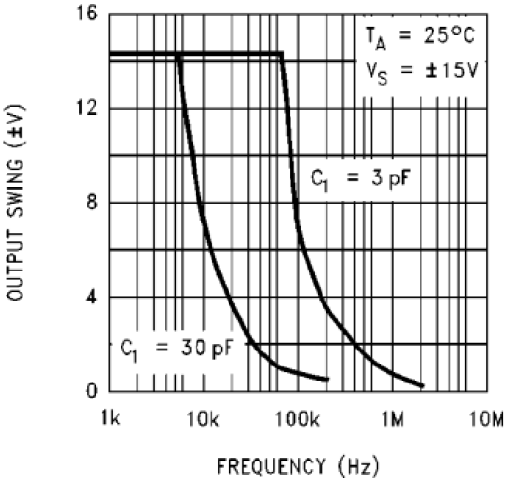
Note 3: Continuous short circuit is allowed for case temperatures to +125°C and ambient temperatures to +70°C.

Note 4: These specifications apply for $\pm 5V \leq V_S \leq +15V$ and $0°C \leq T_A \leq +70°C$, unless otherwise specified.

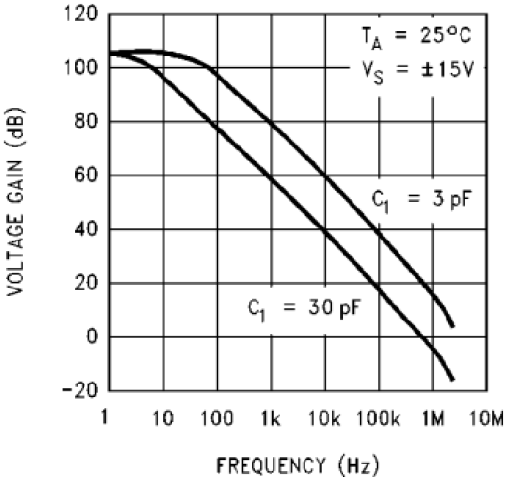
Output Swing



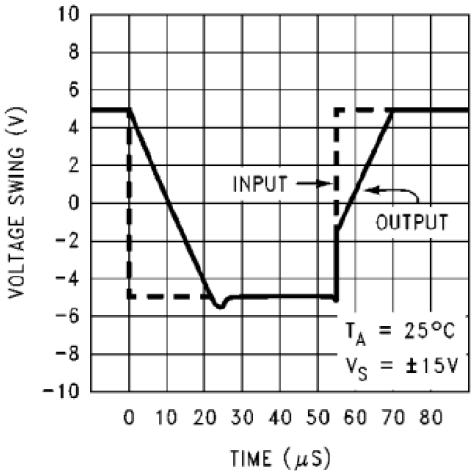
Large Signal Frequency Response



Open Loop Frequency Response



Voltage Follower Pulse Response



Part 1.

Measure the output signal of the uncompensated uA748 opamp at $A_V = 1$ and V_{IN} = GND! Then apply the compensation capacitor according to the uA748 data sheet, and examine the output signal!

Procedure

Build a non-inverting amplifier with $A_V = 1$, so-called voltage follower!

Connection diagram can be seen below. Before you begin to build the circuit, write the pin numbers from the data sheet to the diagram! Power supply is ±15 V, input voltage must be 0 V (short to GND), and C_C compensation capacitor is 0 pF (open terminals). (It is recommended to insert two filter capacitors (100 nF) to the ground, near to the supply pins.)

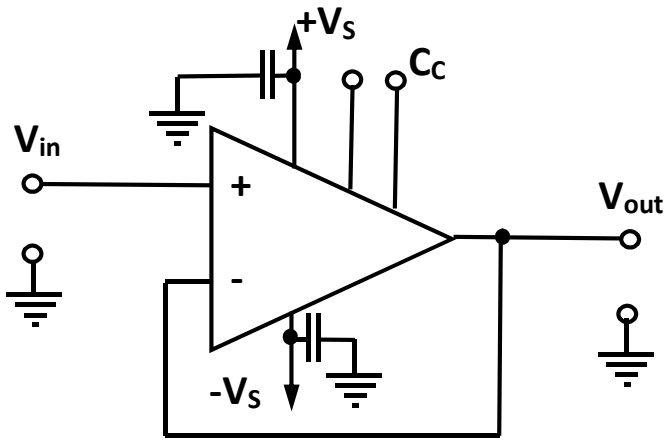


Figure 10. Voltage follower with the optional capacitors on the supply pins

Measure the output signal by oscilloscope and by DMM. (The DMM is in DC voltage mode, and the range is minimized (or auto-range)). Draw the signal on the oscilloscope! (If the signal can't be triggered, use the STOP mode of the oscilloscope!) Write the value measured by the DMM. (Although the AC signal on the output has significant amplitude, the value measured by the DMM can be near to 0 V.)

Then apply the selected compensator capacitor, see the figure above! The output signal must be near to 0 V in AC and DC too. The oscillation ceases, and the output produces only the DC offset voltage. Compare the measured value versus the value in the data sheet.

Part 2.

Examine the signal performance parameters of an amplifier with $|A_V| \approx 10$. These parameters are: SR, f_P, t_r, t_f, and f_c. Draw the waveform of the input and output signals in every case! Modify the compensating capacitor value to examine the effect of it.

Procedure

Modify the connection of the built amplifier to set A_V to +11! Use resistors 1 kΩ and 10 kΩ! (If you prefer, you can build an inverting amplifier with A_V = -10 using same resistor values.) The compensating capacitor value is specified as 3 pF for this amplification. Now, you must examine the output signal by only the oscilloscope.

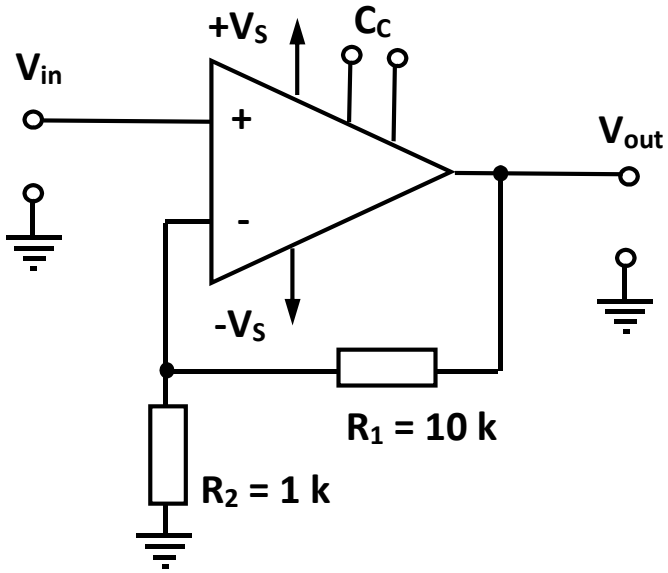


Figure 11. Amplifier, A_V = +11

If you finished the measurements at C_C = 3 pF, change C_C to 10 pF. This is the overcompensated state. Measure all of the parameters above, and draw the waveforms!

If you have enough time, repeat the process at C_C = 30 pF.

If don't have more time, reduce the C_C to that minimal value, where the amplifier is stable (e.g. 2 pF), and repeat measurement of all of the parameter values!

After you measured all of the parameters at a minimum 3 different C_C, fill this table, and draw the conclusions!

C_C	SR	f_P	t_r	t_f	f_c
2 pF					
3 pF					
10 pF					
30 pF					

Table 1. Measured parameters at various C_C

Questions

1. What is the difference in the parameters of the internally and externally frequency compensated operational amplifiers?
2. Describe the dominant pole compensation, draw the Bode-diagrams!
3. Define the next parameters, and show how you can measure them!
    a. SR
    b. f_P
    c. t_r, t_f
    d. f_c
4. How do the parameters above depend on the value of C_C?

Useful links:

http://web.mit.edu/6.101/www/reference/op_amps_everyone.pdf

http://ocw.mit.edu/resources/res-6-010-electronic-feedback-systems-spring-2013/textbook/MITRES_6-010S13_chap13.pdf

http://ocw.mit.edu/resources/res-6-010-electronic-feedback-systems-spring-2013/textbook/MITRES_6-010S13_chap05.pdf

http://www.ti.com/lit/ds/symlink/lm748.pdf

Experiment 2.

Active filters

Objectives

- To learn how to measure the Bode-diagram of a circuit.
- To study the behavior of active filters.

Components

Component	Description	Number of items	Comment
TL081	operational amplifier	1	or equivalent
10 kΩ	resistor	2	
2 nF	capacitor	1	ceramic
100 nF	capacitor	1	ceramic
10 kΩ	trimmer potentiometer	1	

Instruments

Items	Comment
Power supply	dual
Digital multimeter	
Function generator	
Digital oscilloscope	dual channel

Background

The filters

A filter is a device that passes (usually electrical) signals at certain frequencies or frequency ranges while preventing the passage of others. Note, that filters can be implemented digitally too, but in this manual we pay attention only to analog filters.

Active filter uses active components such as an amplifier. Amplifiers included in a filter design can be used to improve the performance and predictability of a filter, while avoiding the need for inductors. The parameters of the active filters are determined by their circuit topologies, and the component values used. The shape of the response, the Q (quality factor), and the tuned frequency can be set with inexpensive resistors and capacitors. There are many circuit topologies, e. g. Sallen-Key, multiple feedback (MFB), state variable, etc.

Using active elements has some limitations, e. g.: Certain circuit topologies may be impractical if no DC path is provided for bias current to the amplifier elements. Power handling capability is limited by the amplifier stages.

- Limited bandwidth
- Limited output voltage
- Noise
- Power consumption
- Need of DC path for bias currents.

Active filters can implement the same transfer functions as passive filters. Common transfer functions are:

- High-pass filter – attenuation of frequencies below their cut-off point
- Low-pass filter – attenuation of frequencies above their cut-off point
- Band-pass filter – attenuation of frequencies both above and below those they allow to pass.
- Band-stop filter (Notch filter) – attenuation of certain frequencies while allowing all others to pass.

Combinations are possible. Ideal filter response curves are shown below.

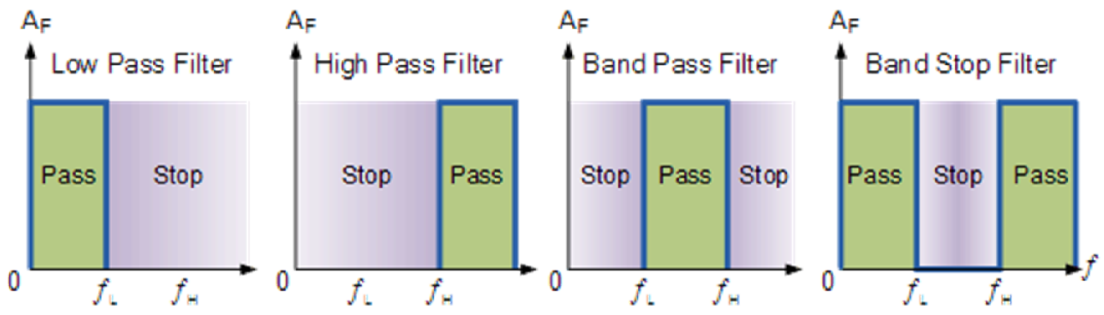


Figure 12. Ideal filter responses of basic filter types

The non-ideal responses have definite transition range (response not changes sharply), have no zero response in the stop band, and may have amplification errors in the pass band.

The Bode plot

The Bode plot is a graph of the frequency response of a system. It is usually a combination of a Bode magnitude plot and a Bode phase plot. The Bode magnitude plot expresses the magnitude of the frequency response, and the Bode phase plot, expresses the phase shift. Both quantities are plotted against a horizontal axis proportional to the logarithm of frequency. Because of the base-10 logarithm, on the horizontal axis the tenfold frequency values are equidistant. E. g. the distance on the X axis between 1 kHz and 10 kHz is equal the distance between 10 kHz and 100 kHz. These tenfold ratios are called decade (D). The magnitude plot's vertical axis is usually in decibel [dB], the scale is linear in decibel. The phase plot's vertical axis is in degree [°], and linear.

The decibel is a logarithmic unit used to express the ratio between two values of a physical quantity, often power or amplitude. In this manual this unit is always used to express amplitude ratios. If the ratio is A, this ratio in dB (a) can be calculated:

$$a = 20 \cdot lgA$$

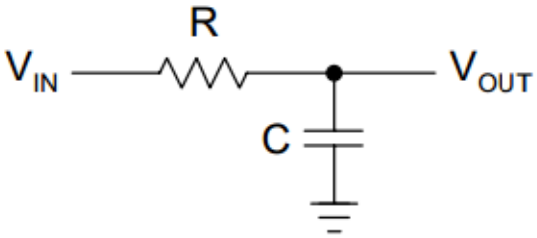
Some examples:

A	a [dB]
100	40
10	20
1	0
0.707	-3
0.1	-20
0.01	-40

Table 2. Examples – ratios and decibel values

Example of a Bode plot

The one of the simplest filters is a 1st order low pass filter made of 2 components, a resistor and a capacitor. This circuit has a cut-off frequency at f_c.



$$\tau = R \cdot C$$

$$f_c = \frac{1}{2 \cdot \pi \cdot R \cdot C}$$

Figure 13. Passive low pass filter (1st order)

This circuit has the Gain and Phase function as following (ω = 2πf):

$$Gain = |H(\omega)| = \left|\frac{V_{out}(j\omega)}{V_{in}(j\omega)}\right| = \frac{1}{\sqrt{1 + (\omega \cdot R \cdot C)^2}}$$

$$Phase = -arctg(\omega RC)$$

- At very low frequencies, where $\omega RC \ll 1$, the Gain is about 1. In this range the Phase is about 0 too.
- If the frequency increases, the denominator increases likewise, and at $\omega RC = 1$, the Gain is $\frac{1}{\sqrt{2}} \sim 0.707$, and the Phase is $-\frac{\pi}{4}$ or -45°.
- At high frequencies, where $\omega RC \gg 1$, the Gain $\rightarrow 0$, and the Phase $\rightarrow -\frac{\pi}{2}$ or -90°.

The Gain and Phase functions of the frequency of this circuit can be seen below.

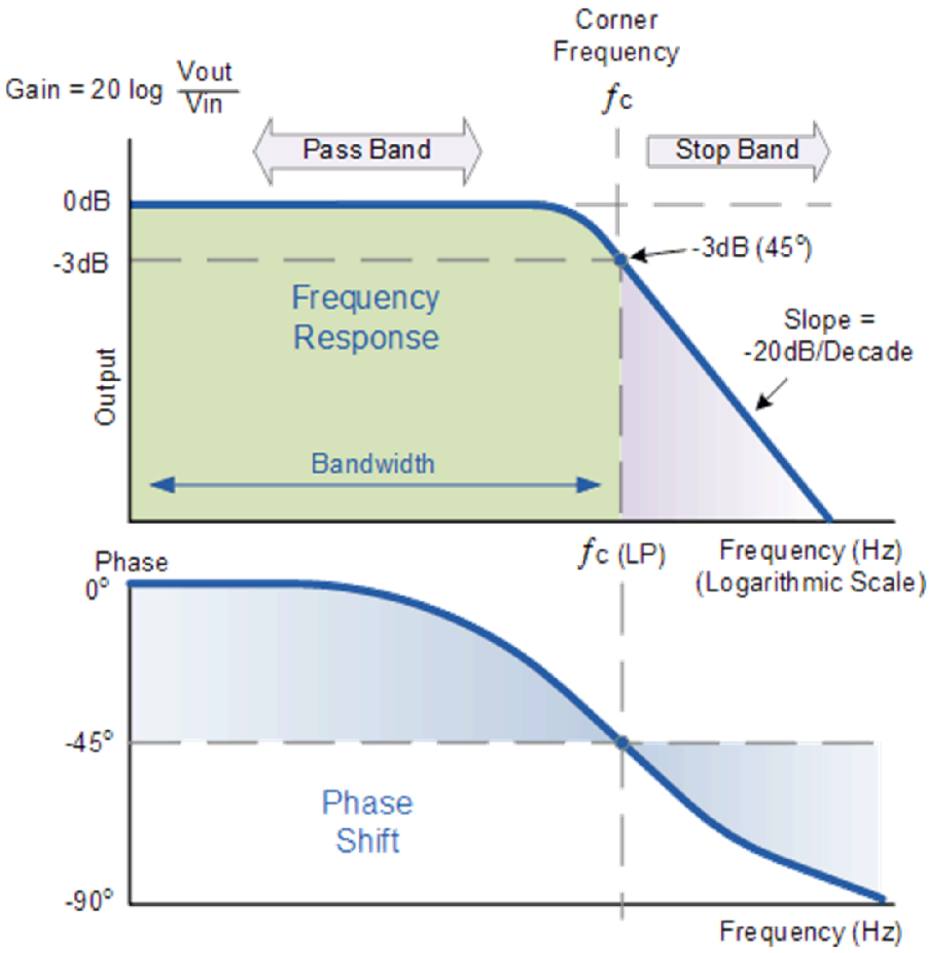


Figure 14. The Gain and Phase functions of frequency of the 1st order passive low-pass filter

How to measure precisely the Bode plot?

When you measure the Bode plot of a circuit, it's wrong way to measure at uniform distance frequencies. You must select the frequencies wisely.

If you know the approximate behavior of the circuit to be measured (e. g. nature of the filter function, planned cut-off frequency, etc.), you must clarify these, because the circuit elements are non-ideal. They have parasitic parameters, or they have not the exact values. (If you don't, first you have to define the behavior of the circuit, you have to take a pre-measurement.)

So, in most cases you can calculate the planned cut-off frequency or frequencies, and you can see the planned characteristics. Then you must determine the real cut-off point(s), and the real characteristics.

The recommended method is presented on a simple RC filter as on the Figure 13. . The test signal is sinusoidal, and we measure the input and output signals by oscilloscope. If the circuit is active, the output level must be low-level: the amplitude can't exceed 1 V_p. If the circuit is passive, we can measure at a higher level. At every point we must record the output amplitude (the record of input amplitude is recommended too), and the phase shift between input and output signals.

Let the calculated cut-off frequency named f_{C_C}. This value is not exactly same as the real value, so we have to determine the real value. Because the circuit is a low-pass filter, we start from the frequency about 1.5…2 decade lower, where the attenuation is nearly 1:1 (0 dB). Then we try to find the real cut-off frequency, f_{C_R}: we begin to increase the frequency, and continue until the output amplitude will fall to 70.7%, or -3 dB. We found f_{C_R}. Then record two points one decade upper and lower to the f_{C_R}. (We can check whether this points meet with expectations, e. g. the input and output amplitude ratio in the point at $10 \cdot f_{C_R}$ is about 0.1 or -20 dB.) Then we examine the amplitude and phase of the input and output signals between $0.1 \cdot f_{C_R}$ and $10 \cdot f_{C_R}$ at least in 4 points. (This is necessary to discover existing enhancements or ripples in the transfer function.) If there is an enhancement or ripple, measure at additional points! At last, we can draw the amplitude and phase diagrams.

Extract of the operational amplifier type TL081 data sheet

General Description

The TL08xx JFET-input operational amplifier family is designed to offer a wider selection than any previously developed operational amplifier family. Each of these JFET-input operational amplifiers incorporates well-matched, high-voltage JFET and bipolar transistors in a monolithic integrated circuit. The devices feature high slew rates, low input bias and offset currents, and low offset-voltage temperature coefficient. Offset adjustment and external compensation options are available within the TL08x family.

Absolute Maximum Ratings(1)

over operating free-air temperature range (unless otherwise noted)

			TL08_C TL08_AC TL08_BC	TL08_I	TL084Q	TL08_M	UNIT
V_{CC+}	Supply voltage(2)		18	18	18	18	V
V_{CC-}			–18	–18	–18	–18	
V_{ID}	Differential input voltage(3)		±30	±30	±30	±30	V
V_I	Input voltage(2)(4)		±15	±15	±15	±15	V
	Duration of output short circuit(5)		Unlimited	Unlimited	Unlimited	Unlimited	
	Continuous total power dissipation		See Dissipation Rating Table				
T_A	Operating free-air temperature range		0 to 70	–40 to 85	–40 to 125	–55 to 125	°C
θ_{JA}	Package thermal impedance(6)(7)	D package (8-pin)	97	97		97	°C/W
		D package (14-pin)	86	86		86	
		N package (14-pin)	76	76		80	
		NS package (14-pin)	80			76	
		P package	85	85		85	
		PS package	95	95		95	
		PW package (8 pin)	149			149	
		PW package (14 pin)	113	113		113	
	Operating virtual junction temperature		150	150	150	150	°C
T_C	Case temperature for 60 seconds	FK package				260	°C
	Lead temperature 1,6 mm (1/16 inch) from case for 10 seconds	J or JG package				300	°C
T_{stg}	Storage temperature range		–65 to 150	–65 to 150	–65 to 150	–65 to 150	°C

(1) Stresses beyond those listed under "absolute maximum ratings" may cause permanent damage to the device. These are stress ratings only, and functional operation of the device at these or any other conditions beyond those indicated under "recommended operating conditions" is not implied. Exposure to absolute-maximum-rated conditions for extended periods may affect device reliability.
(2) All voltage values, except differential voltages, are with respect to the midpoint between V_{CC+} and V_{CC-}.
(3) Differential voltages are at IN+, with respect to IN−.
(4) The magnitude of the input voltage must never exceed the magnitude of the supply voltage or 15 V, whichever is less.
(5) The output may be shorted to ground or to either supply. Temperature and/or supply voltages must be limited to ensure that the dissipation rating is not exceeded.
(6) Maximum power dissipation is a function of $T_{J(max)}$, θ_{JA}, and T_A. The maximum allowable power dissipation at any allowable ambient temperature is $P_D = (T_{J(max)} - T_A)/\theta_{JA}$. Operating at the absolute maximum T_J of 150°C can affect reliability.
(7) The package thermal impedance is calculated in accordance with JESD 51-7.

Electrical Characteristics

$V_{CC\pm}$ = ±15 V (unless otherwise noted)

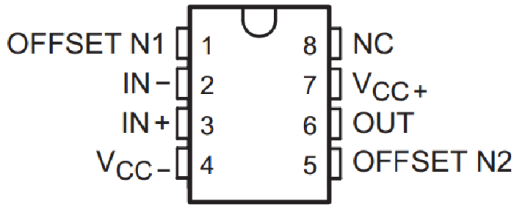
PARAMETER		TEST CONDITIONS	T_A[1]	TL081C TL082C TL084C			TL081AC TL082AC TL084AC			TL081BC TL082BC TL084BC			TL081I TL082I TL084I			UNIT
				MIN	TYP	MAX	MIN	TYP	MAX	MIN	TYP	MAX	MIN	TYP	MAX	
V_{IO}	Input offset voltage	V_O = 0, R_S = 50 Ω	25°C		3	15		3	6		2	3		3	6	mV
			Full range			20			7.5			5			9	
α_{VIO}	Temperature coefficient of input offset voltage	V_O = 0, R_S = 50 Ω	Full range		18			18			18			18		µV/°C
I_{IO}	Input offset current[2]	V_O = 0	25°C		5	200		5	100		5	100		5	100	pA
			Full range			2			2			2			10	nA
I_{IB}	Input bias current[2]	V_O = 0	25°C		30	400		30	200		30	200		30	200	pA
			Full range			10			7			7			20	nA
V_{ICR}	Common-mode input voltage range		25°C	±11	−12 to 15		±11	−12 to 15		±11	−12 to 15		±11	−12 to 15		V
V_{OM}	Maximum peak output voltage swing	R_L = 10 kΩ	25°C	±12	±13.5		±12	±13.5		±12	±13.5		±12	±13.5		V
		R_L ≥ 10 kΩ	Full range	±12			±12			±12			±12			
		R_L ≥ 2 kΩ		±10	±12		±10	±12		±10	±12		±10	±12		
A_{VD}	Large-signal differential voltage amplification	V_O = ±10 V, R_L ≥ 2 kΩ	25°C	25	200		50	200		50	200		50	200		V/mV
			Full range	15			15			25			25			
B_1	Unity-gain bandwidth		25°C		3			3			3			3		MHz
r_i	Input resistance		25°C		10^{12}			10^{12}			10^{12}			10^{12}		Ω
CMRR	Common-mode rejection ratio	V_{IC} = V_{ICR}min, V_O = 0, R_S = 50 Ω	25°C	70	86		75	86		75	86		75	86		dB
k_{SVR}	Supply-voltage rejection ratio ($\Delta V_{CC\pm}/\Delta V_{IO}$)	V_{CC} = ±15 V to ±9 V, V_O = 0, R_S = 50 Ω	25°C	70	86		80	86		80	86		80	86		dB
I_{CC}	Supply current (each amplifier)	V_O = 0, No load	25°C		1.4	2.8		1.4	2.8		1.4	2.8		1.4	2.8	mA
V_{O1}/V_{O2}	Crosstalk attenuation	A_{VD} = 100	25°C		120			120			120			120		dB

(1) All characteristics are measured under open-loop conditions with zero common-mode voltage, unless otherwise specified. Full range for TA is 0°C to 70°C for TL08_C, TL08_AC, TL08_BC and −40°C to 85°C for TL08_I.

(2) Input bias currents of an FET-input operational amplifier are normal junction reverse currents, which are temperature sensitive, as shown in Figure 17. Pulse techniques must be used that maintain the junction temperature as close to the ambient temperature as possible.

Connection diagram

TL081, TL081A, TL081B
D, P, OR PS PACKAGE
(TOP VIEW)



NC − No internal connection

Part 1.

Measure the Bode plot of a simple 1st order passive RC low-pass filter!

Procedure

Build a 1st order passive RC low-pass filter! R = 10 kΩ, C = 2 nF. Connection diagram can be seen below.

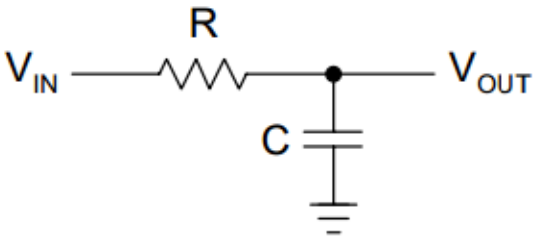


Figure 15. 1st order passive RC low-pass filter

Calculate f_{C_C}! $f_{C_C} =$ .....

Measure the real f_{C_R}, and measure at the frequencies below in the table! (You can perform tests with high level signal.) Then draw the Bode plot (amplitude and phase)!

	0.01 f_{CC}	0.1 f_{CR}	0.2 f_{CR}	0.5 f_{CR}	f_{CR}	2 f_{CR}	5 f_{CR}	10 f_{CR}
V_{in} [V]								
V_{out} [V]								
φ [°]								

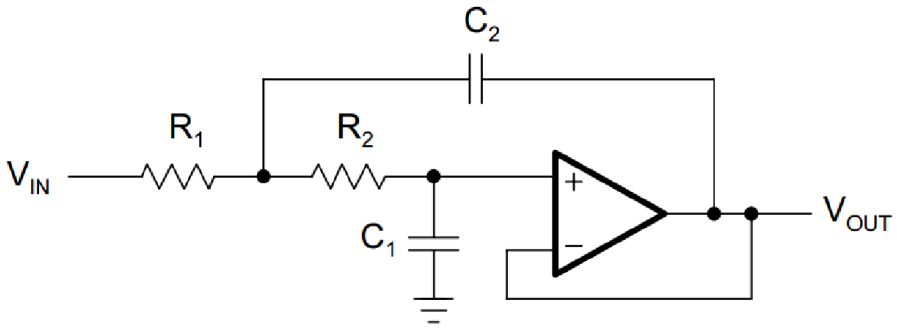
Table 3. Measured values of the passive RC filter

Part 2.

Measure the Bode plot of a 2nd order active low-pass filter!

Procedure

Build the circuit below! $R_1 = R_2 = 10$ kΩ, $C_1 = 2$ nF, $C_2 = 100$ nF.



. Figure 16 2nd order active unity-gain Sallen-Key low-pass filter

Calculate f_{C_C} and Q!

$$f_{C_C} = \frac{1}{2 \cdot \pi \sqrt{R_1 \cdot C_1 \cdot R_2 \cdot C_2}} = \text{.....}$$

$$Q = \frac{\sqrt{R_1 \cdot C_1 \cdot R_2 \cdot C_2}}{C_1 \cdot (R_1 + R_2)} = \text{.....}$$

Measure at the points you determined, and write the measured values below to the table! If it is necessary, expand the table! This is an active circuit, you must perform tests with low level signal! Be careful, because the relatively high Q factor, there is a peak in the transfer near to the f_C! Draw the Bode plot!

V_{in} [V]								
V_{out} [V]								
φ [°]								

Table 4. Measured values of the active RC filter

Part 3.

Measure the Bode plot of an all-pass filter! The all-pass filter is a special circuit, which passes all frequencies equally in gain, but changes the phase relationship between various frequencies.

Procedure

Build the circuit below! R_1 = 10 kΩ, C = 2 nF, R = 10 kΩ trimmer potentiometer.

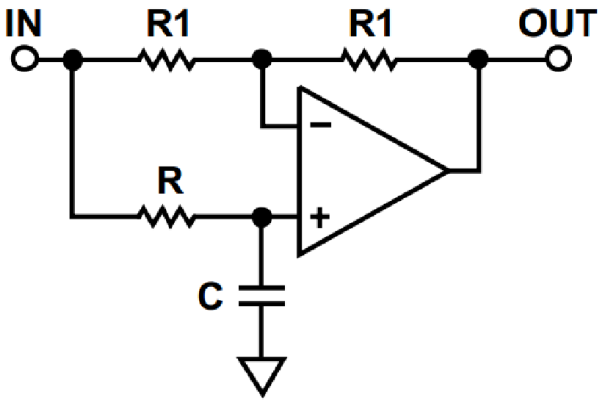


Figure 17. All-pass filter

If R is set and fixed (to e. g. 5 kΩ), the phase shift of the input signal changes from 0° at DC to -180° at high frequency. At the cut-off frequency of the R-C low pass filter at the non-inverting input, the phase shift is -90°.

If the input frequency is fixed, the phase shift varies according to the set value of R. Note, that the range of the phase shift is the function of the frequency.

Set the frequency to 16 kHz, amplitude is about 1 V_p. Check the phase shift, if the R changes! Write the experiences!

Fix the trimmer value at the middle (at about 5 kΩ), then measure at the frequencies you determined, and write the measured values below to the table! Draw the Bode plot!

V_{in} [V]								
V_{out} [V]								
φ [°]								

Table 5. Measured values of the all-pass filter

Questions

1. What is the Bode plot? How can it be measured?
2. Explain the main filter responses!
3. How to specified low-pass/high-pass/band-pass/band-stop filters?
4. Draw an active filter, and determine it's cut-off frequency! Sketch it's Bode plot!
5. How does the all-pass filter work?

Useful links:

http://web.mit.edu/6.101/www/reference/op_amps_everyone.pdf

http://www.ti.com/lit/ml/sloa088/sloa088.pdf

http://www.ti.com/lit/ds/symlink/tl084.pdf

http://sim.okawa-denshi.jp/en/OPseikiLowkeisan.htm

http://www.analog.com/designtools/en/filterwizard/#/type

http://www.ti.com/lsds/ti/analog/webench/webench-filters.page

Experiment 3.

Analog switches

Objectives

- To study the behavior of analog switch elements.
- To learn how to measure the static and dynamic parameters of a reed relay.

Components

Component	Description	Number of items	Comment
DIP05-1A72	reed relay	1	or equivalent
BC182	bipolar transistor	1	or equivalent (NPN)
TL081	operational amplifier	1	or equivalent
CD4052B	analog multiplexer	1	
1 kΩ	resistor	2	
10 kΩ	resistor	1	
100 kΩ	resistor	2	
1 MΩ	resistor	1	
100 Ω/5W	resistor	1	or 51 Ω/5W
470 Ω	trimmer potentiometer	1	

Instruments

Items	Comment
Power supply	dual
Digital multimeter	
Function generator	
Digital oscilloscope	dual channel

Background

The analog switches

Analog switches and multiplexers have become the essential components in the design of electronic systems which require the ability to control and select a specified transmission path for analog signals. These devices are used in a wide variety of applications including multichannel data acquisition systems, process control, instrumentation, video systems, etc.

Mainly the following components are used in/for analog switches:

- Reed relays
- Bipolar transistors
- JFETs
- MOSFETs
- CMOS devices.

Mechanical relays are not used for analog switches, their main function is to switch high current or voltage while ensure galvanic isolation. In most cases they are not suitable to switch low-level analog signals, because the contacts are not designed for it. On the contacts an oxide layer is formed by the environmental oxigen, which isolates the contacts on low voltages, even the contacts look mechanically closed.

The table below contains the brief comparison of mechanical relays, reed relays and semiconductor switches.

Specification	Mechanical relay	Reed relay	Semiconductor switch
Switching time	~10 ms	~1 ms	<100 µs
Life expectancy (mechanical)	10^6 cycles	10^{10} cycles	unlimited
Life expectancy (electric)	10^5 cycles	10^9 cycles	unlimited
Minimal load	50 mW	no load required	no load required
Insulation resistance	10^9 Ω	10^{12} Ω	10^9 Ω
Insertion loss	low	low	high
Overload sensitivity	not sensitive	sensitive	very sensitive
Linearity	DC to MHz	DC to GHz	signal distorsion
Galvanic isolation	yes, air gap	yes, air gap	no
Leakage current	no	no	yes
Power consumption	~100 mW	~10 mW	<<1 mW

The ideal analog switch has no on-resistance, infinite off-impedance and zero time delay, and can handle large signal and common-mode voltages. But the world is not ideal…

Reed relays

The reed relay uses an electromagnet to control one or more reed switches. The contacts are made of magnetic material and the electromagnet acts directly on them without requiring an armature to move them. The contacts are protected from corrosion, and are usually plated with gold, which has appropriate resistivity and corrosion-resistant. As the moving parts are small and lightweight, reed relays can switch much faster than mechanical relays with armatures. They are mechanically simple, resulting high reliability and long life.

Basically the reed switches are sealed in a long, narrow glass tube. The glass envelope may contain one or multiple reed contacts. For this type the user must provide an electromagnet (a coil around), which operates the switch or switches. There are reed relays, which have built-in electromagnet, and are integrated in a case, typically SIL or DIL.

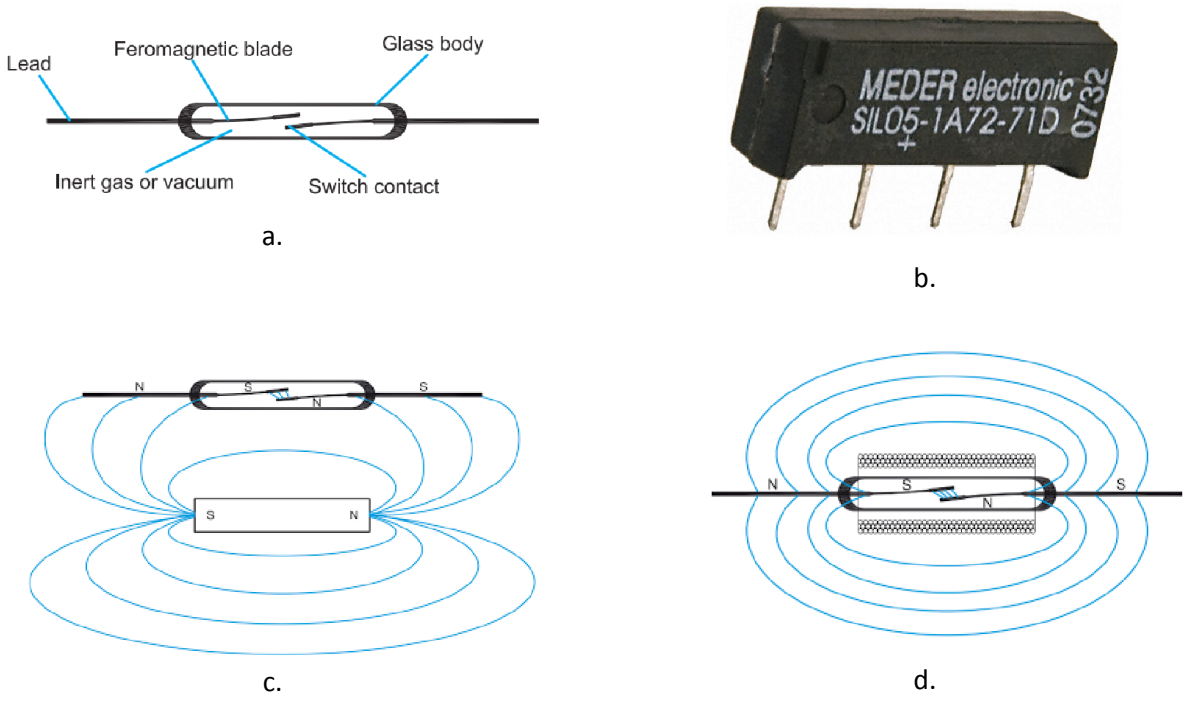


Figure 18. Reed switch (a.), SIL reed relay (b.), reed switch in a magnetic field (c.) and reed switch in a solenoid's magnetic field (d.)

Bipolar transistors

First, it should be noted, that the bipolar transistors are not really suitable for analog switches. However they are very important switch elements, used in many applications, so we take a short summary. Below, you can see the V_{CE}-I_C characteristics of a NPN bipolar transistor.

The areas of operation for a bipolar transistor switch are known as the saturation region and the cut-off region. This means we use the transistor as a switch by driving it back and forth between its “fully-OFF’ (cut-off) and “fully-ON’ (saturation) regions as shown above. The purple shaded area at the bottom of the curves represents the “cut-off’ region while the blue area to the left represents the “saturation’ region of the transistor.

In the cut-off region the operating conditions of the transistor are zero or negative input base current (I_B), zero output collector current (I_C) and maximum collector voltage (V_{CE}) which resulting that no current flowing through the device. Therefore the transistor is switched “fully-OFF’.

In the saturation region high amount of base current is applied, resulting the depletion layer being as small as possible and maximum current flowing through the transistor. V_{CE} drops to a minimum, therefore the transistor is switched “fully-ON’.

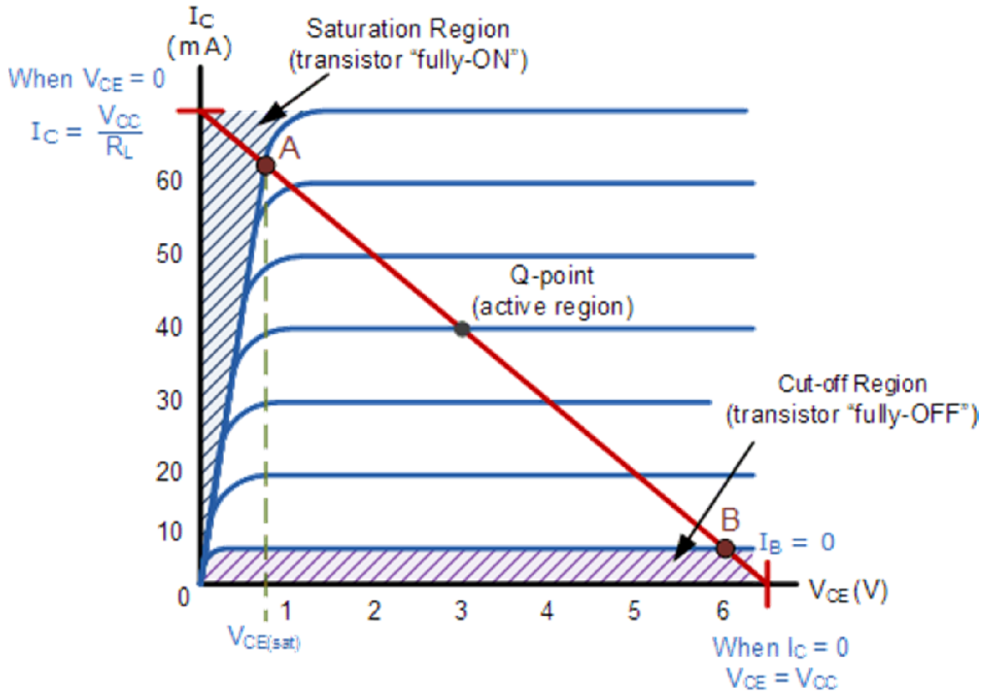


Figure 19. NPN transistor V_{CE}-I_C characteristics

In these operating points either the current or the voltage is near to zero, therefore the power dissipation is approximately zero (the power dissipation of the transistor can be calculated as: $PD = V_{CE} \cdot I_C$). It is important, that the operating voltage of the load (at the collector) can be much higher, than the control voltage on the base, so this circuit is suitable for voltage level matching. There are many integrated circuits, which has an output stage made of bipolar transistor (or MOSFET), and the collector (or drain) point wired as output. These outputs are called open-collector (or open-drain) outputs.

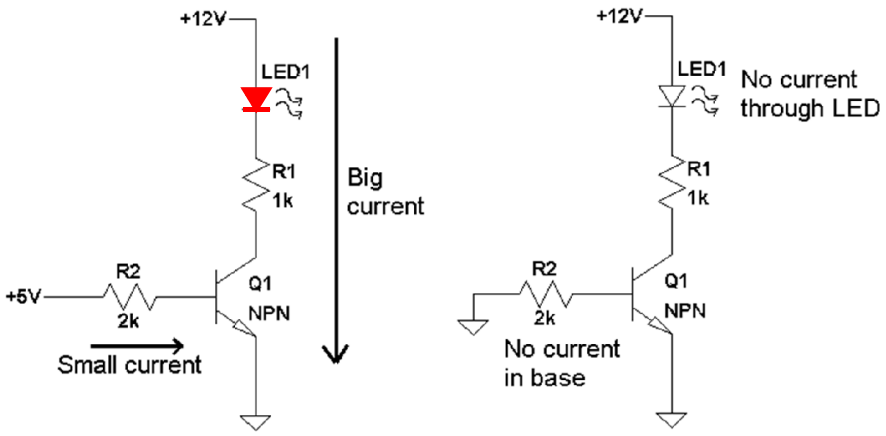


Figure 20. Example of NPN transistor as a power switch

JFETs

JFETs was commonly used for analog switches in interval 1980…90, but due to their relatively difficult circuitry and high switched-on resistivity only rarely used as analog signal switch. Today for special applications such as video switching and multiplexing are the JFETs and complementary bipolar processes used, because the high performance characteristics required are not attainable with CMOS.

MOSFETs

Single MOSFETs are used only for power switches. If they are used for analog signal switches, in most cases they are integrated in pairs: one N-channel, and one P-channel MOSFET form a CMOS (complementer MOS) switch. If a single MOSFET were used as an analog signal switch, there arises a problem the switched on-resistance varies with the V_{GS}, which varies due to the change in the input signal. The switch is bidirectional; the source and drain electrodes are interchangeable (while operating, the side with the lowest $V_{I/O}$ is the source). On a picture below you can see a single (N-type) MOSFET switch, and its r_{DSON}-(V_I-V) characteristics, where V is the gate voltage.

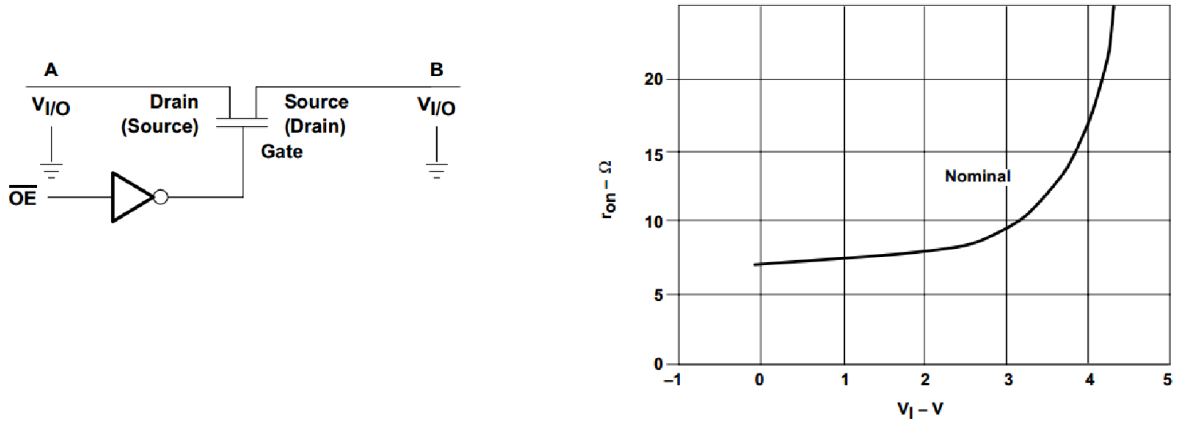


Figure 21. Single N-type MOSFET switch and its r_{ON} characteristics

If we use the CMOS configuration, where the switch consist of a single n-channel transistor in parallel with a single p-channel MOSFET, the resulting switched-on resistance characteristics will be much flatter, than in the previous case. A flat r_{ON} is especially important if $V_{I/O}$ signals must swing from rail to rail.

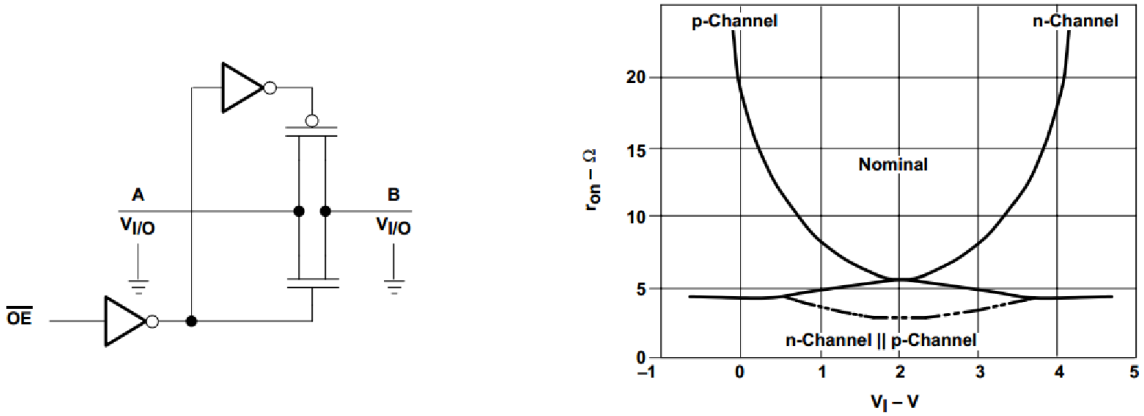


Figure 22. CMOS switch and its resultant r_{ON} characteristics

The CMOS switches have other errors too: the non-zero r_{ON}, which mainly causes errors in low resistance networks. If the network has high impedance, the errors are caused by the leakage currents and parasitic capacitances.

The leakage currents mainly consist of the sub-threshold voltage current (when the channel is closed) and of the gate current (through the oxide layer). These currents can be modeled by a single current generator, shown on the figure below. When the MOSFET is on, the current can flow through the load, and through the input to the ground. So the current is divided between to paths, and the current flowing to the load is determined by the resistance ratio of the R_{LOAD} and the (R_G + R_{ON}). If the MOSFET is off, the leakage current can flow only through the load, causing voltage error on the output.

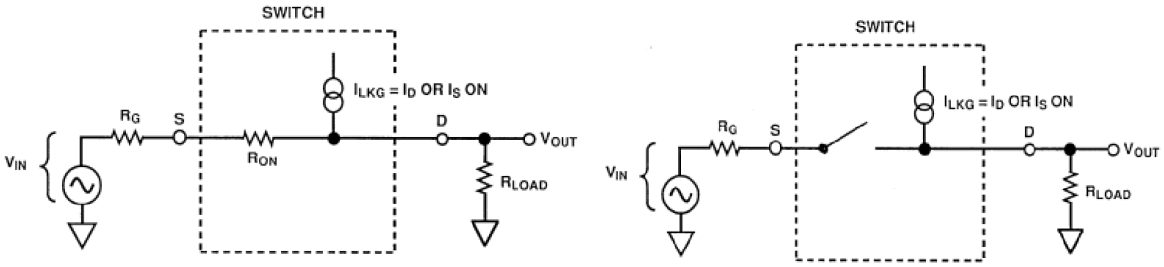


Figure 23. Leakage current modelling

The parasitic capacitances affect the AC performance of CMOS switches. Additional external capacitances will further degrade performance. These capacitances affect feedthrough, crosstalk and system bandwidth. C_{DS} (drain-to-source capacitance), C_D (drain-to ground capacitance), and C_{LOAD} all work in conjunction with R_{ON} and R_{LOAD} to form the overall transfer function. The series-pass capacitance, C_{DS}, not only affects the response in the on-state, it degrades the feedthrough performance of the switch during its off state. When the switch is off, C_{DS} couples the input signal to the output load. The circuits modeling the parasitic behavior of MOSFET switches are shown below.

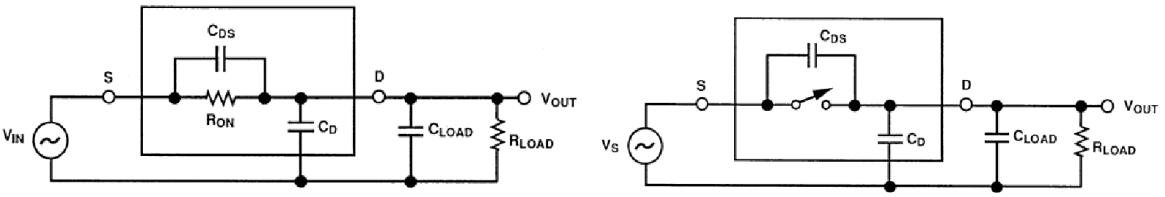


Figure 24. Parasitic capacitance modeling

The common switch functions like single switches, multiple switches, multiplexers and demultiplexers are mostly formed by CMOS switches integrated in a single circuit.

Measurement of the basic and performance parameters of a reed relay

If we examine the data sheet of a reed relay, we can found many specified parameters. There are static and dynamic parameters, some of them characterize the coil, others the contact.

Main static parameters are:

- Coil resistance
- Pull-in voltage
- Drop-out voltage
- Static contact resistance

Main dynamic parameters are:

- Operate time (switch-on time)
- Release time

In the next, we describe briefly, how to measure these parameters. Determine the coil resistance is very simple: use a DMM in resistance measurement mode.

For measuring pull-in voltage (and drop-out voltage) we use the circuit below.

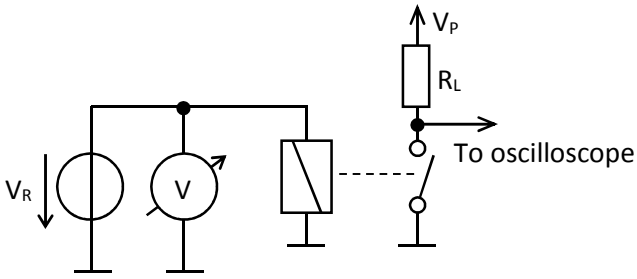


Figure 25. Circuit for measuring pull-in and drop-out voltages

The voltages of V_R and V_P are different, while V_R will be changed between 0 V and the nominal voltage of the reed coil, the V_P is fixed. The value of the V_R is measured by a voltmeter, determining the exact value of the pull in and drop-out voltages. The point between R_L and the switch is observed by an oscilloscope. If the switch is open, on that point will be the V_P, however, if the switch is closed, there will be 0 V.

Pull-in voltage

The pull-in voltage is the coil voltage required to pull-in the relay contacts. So, we must raise the coil voltage (V_R) from 0 V, and watch if the contact closes. If the output goes to 0 V, we must record the value from the voltmeter. It is recommended to measure this point three times, and values shall be averaged.

Drop-out voltage

The drop-out voltage is the coil voltage at which a relay's contacts will dropout. To measure this voltage, we must reduce the coil voltage (V_R) from nominal voltage, and watch if the contact opens. If the output goes to V_P, we must record the value from the voltmeter. It is recommended to measure this point three times, and values shall be averaged.

Contact resistance

The contact resistance is the total electrical resistance of a pair of closed contacts measured at their associated contact terminals. Contact resistance is determined by measuring the voltage drop across the contacts ($V_{CONTACT}$) using the appropriate test current ($I_{CONTACT}$) shown below.

First, the nominal coil voltage must apply. Then, as the switch closed, on R_L will flow a current:

$$I_{CONTACT} = \frac{V_P}{R_L}$$

and the contact resistance will be:

$$R_{CONTACT} = \frac{V_{CONTACT}}{I_{CONTACT}}$$

Because of the low $V_{CONTACT}$ it is essential to measure this voltage very close to the pins of the reed relay switch, if it is possible, directly on the pins. This way, the wiring resistance and other additional error will be ignored.

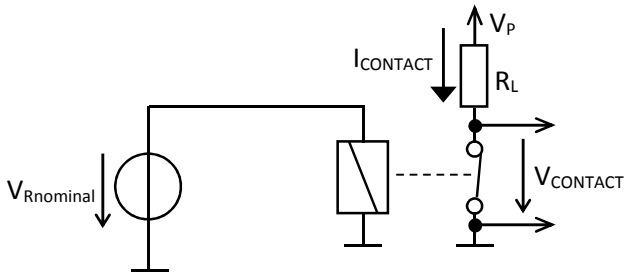


Figure 26. Circuit for measuring contact resistance

Operate time

The operate time is the time that elapses between the instant power is applied to a relay coil and the moment the contacts have closed. In case the relay has several contacts, the duration of the operate time shall be considered to end when the last contact has closed. Bounce time is not included in the operate time of a relay unless otherwise specified. Bounce is the intermittent opening and closing of contacts caused by vibration or shock resulting from the collision of the relay's moving parts.

For measuring dynamic parameters, we must switch on and off periodically the contacts, so a square wave must to apply to the coil of the reed relay. The simplest way to ensure this square wave at a suitable power level is using a bipolar NPN transistor, as shown below.

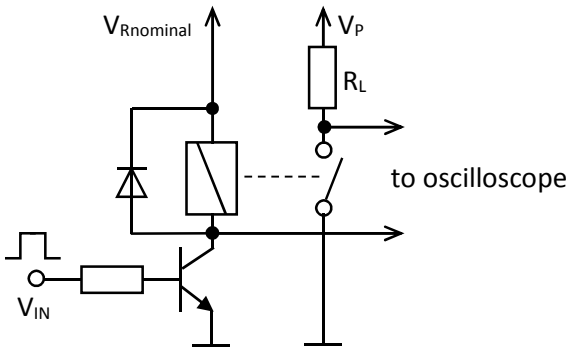


Figure 27. Circuit for measuring dynamic parameters

If the V_{IN} has a rising edge, on the collector of the transistor will be a falling edge, and the relay switches on. So the voltage at the output (the common point of the R_L and the switch) occurs a falling edge. The time between the falling edge on the collector and the falling edge at the output will be the operate time.

The diode on the solenoid has an important function. While the transistor is switched on, a current flows through the solenoid, and a magnetic field is built up. If the transistor closes, the current flow stops to the ground. But the magnetic field (and magnetic energy) remains there, and starts to collapse. The energy can be neither created nor be destroyed, but it can change form, so the magnetic energy stored in the solenoid forces the current flow to continue, and a voltage spike in reverse direction will be created. This voltage spike is higher, if the switch-off of the transistor is quicker. The spike can have a very high amplitude, even hundreds of volts, and can destroy the transistor. The diode on the solenoid can shunt this spike, and protects the transistor.

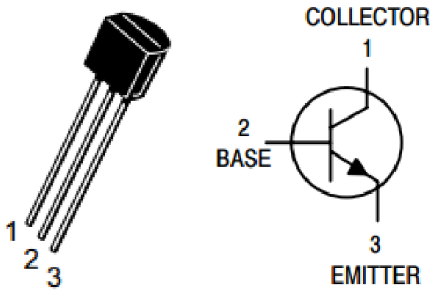
Release time

For an SPST (single pole single throw) relay, the release time concludes at the opening of the NO (normally open) contacts. For an SPDT (single pole double throw) relay, the release time is the time that elapses between the instant a relay coil is de-energized, and closure of the NC (normally closed) contacts. Release bounce time is not included in the release time of a relay.

Figure 28. SPST and SPDT switches

This parameter can be measured in a circuit described at the measurement of the operate time. If the V_{IN} has a falling edge, on the collector of the transistor will be a rising edge, and the relay switches off. So the voltage at the output (the common point of the R_L and the switch) occurs a rising edge too. The time between the rising edge on the collector and the rising edge at the output will be the release time.

Pinout diagram of BC182



The reed relay type MEDER DIP05-1A72-12D data sheet

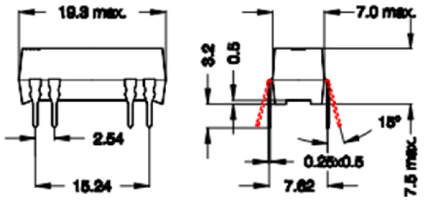
MEDER electronic AG
Friedrich-List-Strasse 6
78234 Engen-Welschingen

Phone.: +49 (0) 7733 / 9487-0
Fax: +49 (0) 7733 / 9487-32
eMail: info@meder.com
Internet: www.meder.com

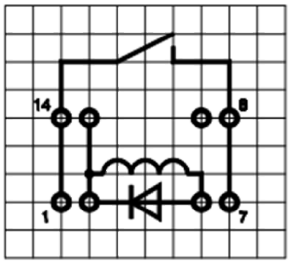
type: DIP05-1A72-12D

part number: 3205100112

dimensions (tolerance ± 0,1mm)

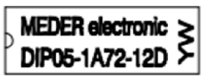


layout 12D **pitch** 2,54 / **top view**



marking

Type
data-code EN 60062



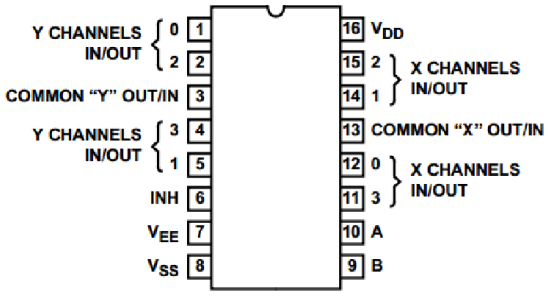
coil data	condition	Min.	Typ.	Max.	unit
coil resistance	at 20°C	450		550	Ω
nominal voltage			5,0		VDC
pull-in voltage				3,5	VDC
drop-out voltage		0,75			VDC
coil voltage	at 20°C			22,0	VDC
coil voltage	at 60°C			14,0	VDC
nominal power	determined with nominal voltage and rated current		50		mW

contact data (Form /Dry)					
contact material		Rhodium			
rated power	each combination of the switching voltage and current must not exceed the given rated power			15	W
switching voltage				200	VDC
switching current				1,0	A
carry current				1,25	A
static contact resistance	initial values measured with 1,4 x $AT_{pull-in}$			150	mΩ
Insulation resistance	RH Ω 45%	10^{10}			Ω
breakdown voltage		250			VDC
capacitance	without test coil			0,3	pF

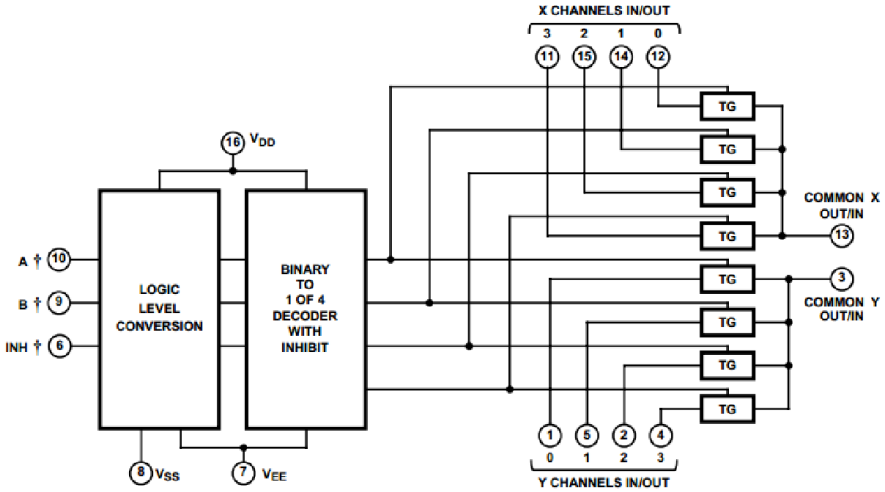
relay data					
insulation resistance coil-contact		10^{11}			Ω
insulation voltage coil-contact		1,5			kVDC
shock	½ sine wave, duration 11ms			150	g
vibration	50 – 2000Hz			10	g
operate time including bounce	measured at 1,4 x $AT_{pull-in}$		0,5		ms
release time			0,1		ms

general data					
operating temperature		-20		70	°C
storing temperature		-35		95	°C
soldering temperature	10 sec. at			260	°C
cleaning		fully sealed			
material of case		mineral-filled epoxy			
material of pins		Cu-alloy tinned			

Pinout diagram, truth table and block diagram of CD4052B MUX/DEMUX



CD4052B			
INHIBIT	B	A	"ON" CHANNEL(S)
0	0	0	0x, 0y
0	0	1	1x, 1y
0	1	0	2x, 2y
0	1	1	3x, 3y
1	X	X	None



Part 1.

Measure the static parameters of a DIP reed relay!

Procedure

First, plug the reed relay into the breadboard, and measure the resistance of the coil directly using a DMM! Check if it is meets the data sheet! Then calculate the operating power of the reed relay at nominal coil voltage!

Then build the circuit shown on Figure 25! Use the one side of the power supply to maintain V_P, fixed to 5 V, and use the other side to set the V_R voltage! While working, the V_R prohibited to exceed 5 V! It is important, that the MEDER DIP05-1A72-12D contains an integrated free-run diode, which must be in a reverse direction, i. e. its cathode must connect to the V_R! R_L be a power resistor!

Measure the pull-in voltage and drop-out voltage of the relay three times! Average the results! Check if it is meets the data sheet!

Then build the circuit shown on Figure 26! In this configuration, $V_{Rnominal}$ = 5 V, V_P = 5 V, both fixed. The contact is always closed, and on the R_L flows about 50 mA. Measure the voltage on the switch, putting the probes to the pins of the reed relay! If the voltage is too small, raise the V_P to 10 V! Calculate the resistance of the contact, check if it is meets the data sheet!

Part 2.

Measure the dynamic parameters of a DIP reed relay!

Procedure

Build the circuit shown on Figure 27! $V_{Rnominal}$ = V_P = 5 V, fixed. Vin is a 5 V amplitude square wave signal, low level is 0 V, high level is 5 V, frequency is about 100 Hz. Measure operate and release times! Be careful! Falling edge on the collector and on the contact means that the relay operates, and at rising edges releases! Check if the measured parameters meet the data sheet!

Then increase the frequency of V_{in} to 500 Hz, and examine the bounce! Draw it! Remember that all the contacts (switches, buttons, relay contacts) produces bounce, and must be ensured by debouncing! (E. g. if a push button connects to a digital input of a microcontroller. Why?)

Then increase the frequency, and search that point, when the relay shuts off. Examine how the bounce time and the on-time ratio changes, as the shut off point approaches!

Part 3.

Homework:

Design a PGA (programmable gain amplifier)! Use a TL081 opamp, a CD4052 analog multiplexer and resistors (values can be found in the 'Components' table at the beginning of this chapter)! Gains are: x0.01, x0.1, x1, x10, all inverting. The maximum of VDD-VEE voltage is 20 V, according to the data sheet of CD4052. In our application be VDD = 5 V, VEE = -5 V.

Examine the built circuit!

Procedure

Build the circuit! Selector inputs can be controlled by applying 0 V or VDD to them. You can use the switches of the demo board, if the logic voltage input of the board is used! Select several gain settings, and measure the gains and the high cutoff frequencies at these settings! Describe, how can use this circuit in a system! Think, how the autorange function can be implemented with this PGA and a microcontroller with an integrated ADC!

Questions

1. Compare a reed relay and a CMOS switch as analog switches!
2. Which are, and how to measure the static parameters of a reed relay?
3. Which are, and how to measure the dynamic parameters of a reed relay?
4. What is the PGA? What components can build a PGA?
5. What are the similarities and differences (and what parameters of the switching elements are important) if you want to operate a DC motor with a CMOS driver, or want to multiplex low level analog signals with CMOS switches?

Useful links:

http://www.analog.com/media/en/training-seminars/tutorials/MT-088.pdf

http://www.ti.com/lit/an/szza030/szza030.pdf

https://www3.panasonic.biz/ac/e_download/control/relay/common/catalog/mech_eng_term.pdf?f_cd=401551

Experiment 4.

Simulation of logic circuits

Objectives

- To study the behavior of combinatorial and sequential logic circuits by simulation.
- To learn how to use basic simulator functions.

Components

None

Instruments

None

Other

Items	Comment
PC	with installed B2 Spice A/D v4 Pro

Background

The simulation

The circuit simulation programs use mathematical models to replicate the behavior of an actual electronic device or a circuit. Simulating a circuit's behavior prior to building it can greatly improve its efficiency, allow to catch potentially costly 'bugs', as well as to provide additional insights into the circuit's performance by exploring various 'what-if' scenarios. Such simulations are particularly important when routine circuit prototyping using a breadboard is not easily available, e.g. as in a design of integrated circuit devices. More generally, most new circuits under development, except for the simplest kind, can benefit from such electronic computer-aided design (ECAD). There is a large number of circuit simulators available both commercially and for free. Depending on which circuit type needs to be simulated, a particular code may be geared towards treatment of either analog or digital signals. However, most of the present-day ECAD programs support some form of mixed-mode - a mode that allows to simulate both analog and digital circuits.

The B2 Spice software allows for the simulation of digital, analog, and hybrid circuits. However, this manual is only concerned with the simulation of digital circuits. This tutorial is designed to provide an introduction to B2 Spice. This tutorial describes only a small portion the B2 Spice capabilities. Circuit simulation is a valuable design and test tool. Simulating circuits before they are implemented can save development time by flushing out errors in the circuit. Simulation can also be used to test portions of circuits that are not directly testable by any other means. The validity of the simulation process is dependent upon the how well the simulation matches the output of a circuit if the circuit was actually implemented. If the models used in the simulation process accurately represent the actual device characteristics, the simulation output will match the actual output. In other words, the validity of the simulation is based upon the quality of the models used. The B 2 Spice simulator allows you to change model parameters to more accurately represent the devices used in your circuit.

In the following we present how to test simple circuits in B2 Spice, using the digital simulation mode. The examples in the next, only the most important, and useful tools and options are described. Additional knowledge can be collected during the laboratory exercises, from the examples in the B2 Spice Help and by self-study.

Example – Simulating a combinatorial logic circuit

Launch the B2 Spice program. (The associated icon should appear on the Windows desktop. If the B2 Spice icon is not present, then launch the program from the Start menu.) The resulting display should appear similar to the window shown in figure below.

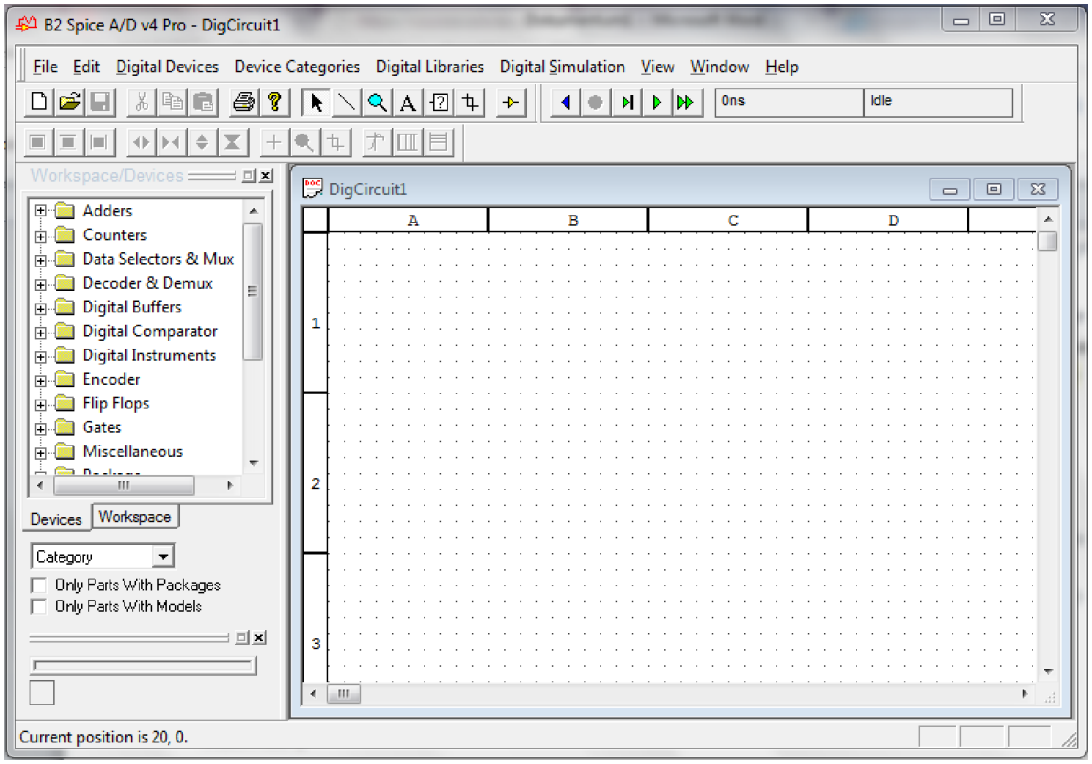


Figure 29. B2 starting display

After the B2 Spice is launched, switch to digital mode (*File → Switch To Digital Mode*). If you can't find this menu item, the simulator is already in digital mode. The work window is recommended to maximize. Note, this program is rather old, and not fully compatible with new operating systems. If the program crashes, the B2 Spice tries to recover your work, but rather please save your work regularly!

Build the circuit shown below!

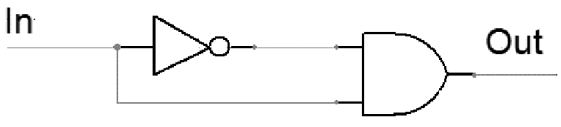


Figure 30. Pulse narrowing circuit

A step-by step description of the procedure used to build this circuit is provided in the next.

1. From the File menu, select Switch to Digital Mode.

2. From the *Digital Devices* menu, select an inverter and place on the workspace.
3. From the *Digital Devices* menu, select an AND gate and place on the workspace.
4. From the *Digital Devices* menu, select an Output Port gate and place on the workspace.
5. Hit the space bar to select another Output Port and place on the workspace. Although the circuit shown in the figure above only has one output, placing a second Output Port in the design is useful to view the inverter output in the subsequent timing diagram.
6. From the *Digital Devices* menu, select an Input Port and place on the workspace.
7. Use the selection tool or the wire tool to connect the various circuit elements. Your final circuit should appear as shown in figure below.

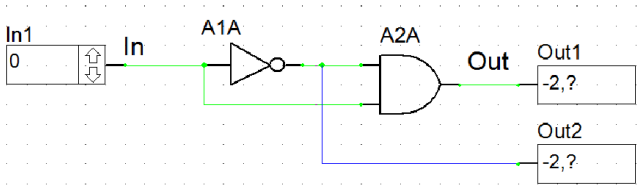


Figure 31. The built pulse narrowing circuit in B2 Spice

8. Verify that the circuit is wired properly by selecting the *Check for Wiring Errors* option under the *Digital Simulation* menu. Correct any errors that are detected. (If your circuit appears to be correct but the simulator tells you there are wiring errors that you can't seem to find or get rid of, close the simulator and re-launch. This should make these "errors' go away.)
9. Change the circuit element names! Place the cursor over element names 'In1', 'Out1', and 'Out2'. Double-click the left mouse button and change the circuit element names to 'A', 'Z', and 'Inv_out', respectively. Your circuit should now look similar to the circuit shown in the figure below.

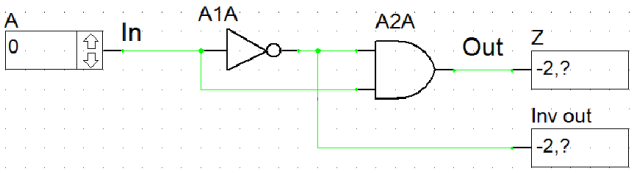


Figure 32. The pulse narrowing circuit with named input and outputs

10. View the Trace Window! Select the *Show Timing Diagram* option under the *View* pull-down menu. This bring ups a window similar to what is shown in figure below. Right click in the window, and feel free to change the colors or order of signals if you desire.

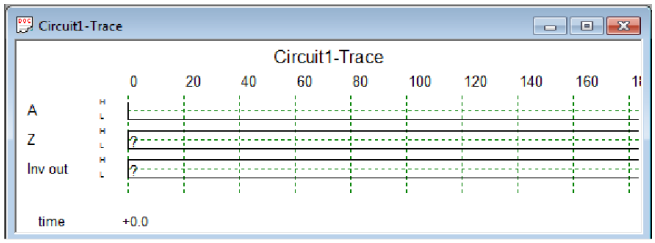


Figure 33. The trace window

11. Simulate the circuit! Use the *Step* button to start the circuit simulation. Use the *Selection Tool* to change the input stimulus on the Input Port device. The trace window should look similar to the window shown below. Change the input stimulus and then take a few steps.

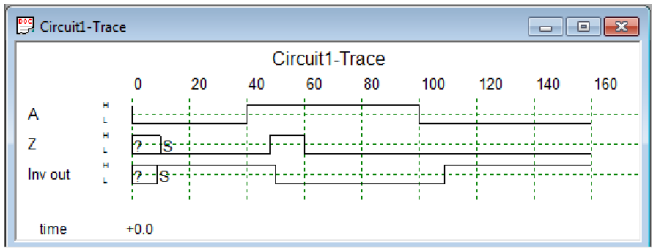


Figure 34. Simulation results in the trace window

12. Change the default propagation delay times! The models of the devices you used in your circuit have many properties associated with them. You can change these properties to more accurately represent the circuit you are testing. Listed below is a procedure that shows how to change the default propagation delay times. The propagation delay times are one of many characteristics associated with the individual device models.
    - Right-click on the inverter.
    - Select *Edit Part Properties…* → *Edit Model…* → *Edit Pin Properties*.
    - Select the output pin from the dialog window on the left. This brings the T_{phl} and T_{plh} times to the windows on the right.
    - Select the *Custom* radio button.
    - Change the T_{plh} time to 4 ns and the T_{phl} time to 5 ns.
    - Select the *Accept* button.
    - Select OKs to remove the three dialog boxes from the screen.
13. Change the simulation step size! Often, finer resolution in the output timing diagram is required to enable you to locate certain aspects of your circuit simulation output. Changing the simulation step size allows you to control certain aspects of your simulation output.
    - Under the *Digital Simulation* menu, select the *Digital Simulation Options…* and enter a *Step Size* of 10 ns. Make sure the schematic window is highlighted when implementing this step.

14. Re-simulate the circuit! The circuit is now ready to be simulated using the new simulation parameters.
    - Under the *Digital Simulation* menu, select *Reset Simulation*.
    - Perform another simulation on the circuit using the directions above.
    - The trace window should appear similar to the former simulation.
15. Use a test vector file to test the circuit! Make a sample test vector file: *Digital Simulation → Create Test Vector File…* Save the generated file, then open from *File* menu (*Open Text File*)! The content will be similar to the following:

Time	**A**
0.0	**0**
40.0	**1**
100.0	**0**
200.0	**1**

16. The test vector file is mostly used to test combinatorial logic circuits. The content in the 'Time' and input 'A' column may vary according to your previous stimulus. In the Time column there are the moments of the input changes in nanoseconds, and in the column A the new logic values. Change the time and input values, then save the file! Then reset the simulation, and select *Digital Simulation → Process Test Vector File*! See the trace window!
17. See the timing table for this simulation! This table can find: *View → Show Timing Table*
18. Then try how to use command files! Command files are for testing the circuits with many inputs automatically and allow to run the simulation for certain period of time. *Click Digital Simulation → Create Command File*. Open the created command file! The content is similar to the following:

Reset
Set A 0 0
Set A 1 40
Set A 0 100
Set A 1 200
Go 200

19. The command file is useful to test sequential logic circuits, for example shift registers, counters etc. if there is a defined clock source. The command file language consists of three simple commands:
    - *Reset*, which resets the time to zero and initializes all of the components.
    - *Go <t>*, which advances time by t nanoseconds. Run has the same effect as Go. Note, the running time defined by Go or Run command must be longer or equal

than the maximum of defined times in the Set commands (see next list item), or else there will be an error message.

- *Set <c> <v> <t>*, which sets the input named c to the hexadecimal value v at time t. The Set commands do not have to be ordered by time in the file. If you want to set an input to 0 or 1 immediately, use Set <c> <v> command, which sets the input named c to the hexadecimal value v at the current simulation time.
- *Rem* or ';' can be used at the beginning of a line that is a comment.

20. Change the command file, then save, and watch the simulation results in the trace window or in the timing table. After reset the simulation, the command file can be executed by selecting *Digital Simulation → Do Command File*!

Simulating a sequential logic circuit

In the following, we give a brief description, how to test a sequential circuit. The tested circuit is the 74192 (synchronous 4-bit BCD up/down counter). Its logic symbol and time sequence from the catalog can be seen below. Before testing this circuit, go to the *Digital Simulation → Digital Simulation Options*, and uncheck all the items under *Error Handling* options! If in the next, you have any problems simulating sequential circuits, try to set in this window the *Default Initial Conditions* to *Random (1,r) and (0,s)*, and reset the simulation!

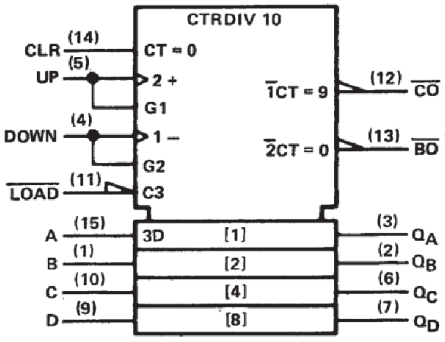
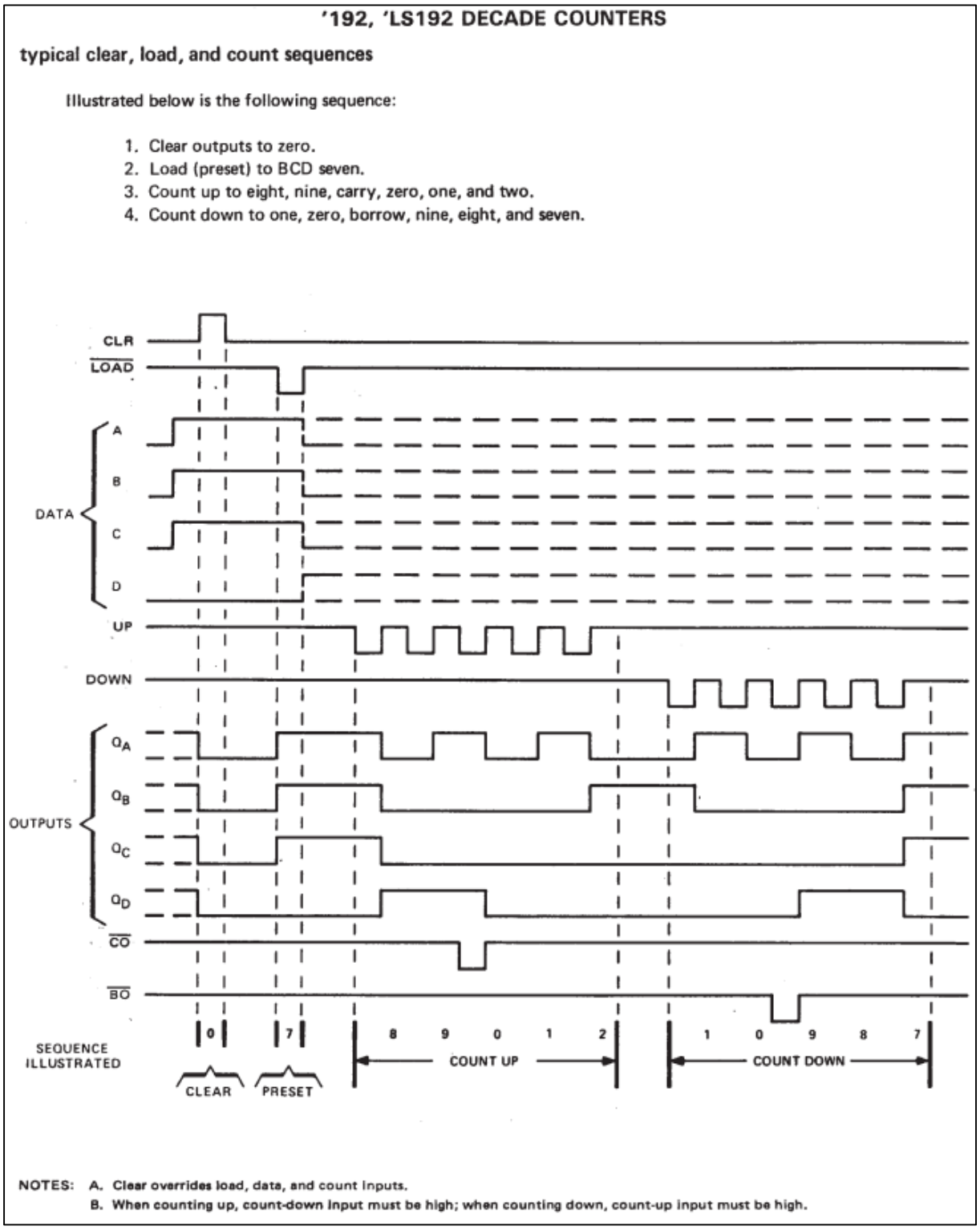


Figure 35. 74192 symbol

The pin functions are:

- ct=0: (CLR), resets the outputs QA…QD
- 2+, g1: (UP), clock for counting up, or enable for counting down
- 1-, g2: (DOWN), clock for counting down, or enable for counting up.
- c3: (LOAD), loads the value on A…D inputs
- $\overline{1}$ct=9: ($\overline{\text{CARRY}}$), carry, shows the 9→0 transition on the output
- 2ct=0: ($\overline{\text{BORROW}}$), borrow, shows the 0→9 transition on the output
- 4 pins below on the left: (A, B, C, D), data input for loading
- 4 pins below on the right: (QA, QB, QC, QD), counter output

- The numbers 1, 2, 4, and 8 in parenthesis inside the symbol show the digit values of the registers.



36. Figure 74192 time sequence

We will test the next functions of the counter:

1. Resetting.
2. Counting up, and the functionality of the $\overline{\text{CARRY}}$ pin.
3. Counting down, and the functionality of the $\overline{\text{BORROW}}$ pin.
4. Loading a value from the input.

To perform the tests above, we need the next devices:

- Basic devices:
  - 4 pieces of digital input port (*Digital Devices → Input Port*)
  - 3 pieces of digital output port (*Digital Devices → Output Port*)
  - 1 piece of clock source (*Digital Devices → Clock*)
- Components for the multiplexer to change the counting direction:
  - 2 pieces of dual input OR gate (74LS32)
  - 1 piece of inverter gate (74LS04)

  The last two component model you can find on the left side, in the devices listing, and gates group.
- For simplifying the schematic, we use buses. For creating buses:
  - 1 piece of combiner to combine 4 separated line to a bus (*Digital Devices → Combiner (4->1)*),
  - 1 piece of splitter to split a bus to 4 individual line (*Digital Devices → Splitter (1->4)*).

Put these devices to the workspace, then name them, and create wiring, as seen below!

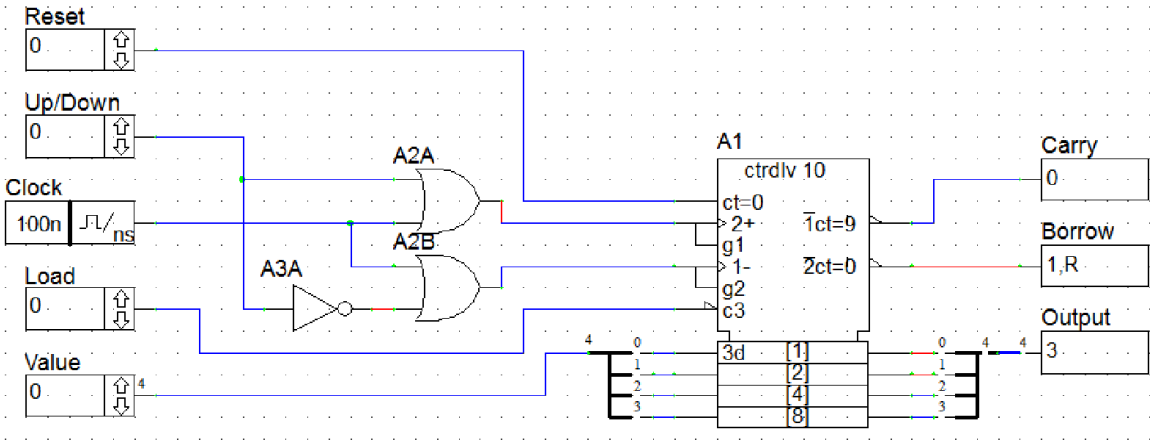


Figure 37. 74192 test layout

Use the models signed by ‘D’, if possible. These are the pure digital models. Be careful, the port names must be set exactly, otherwise the command file in the next cannot be executed! The Input port named Value is a 4-bit port, set the # of bits field as shown below!

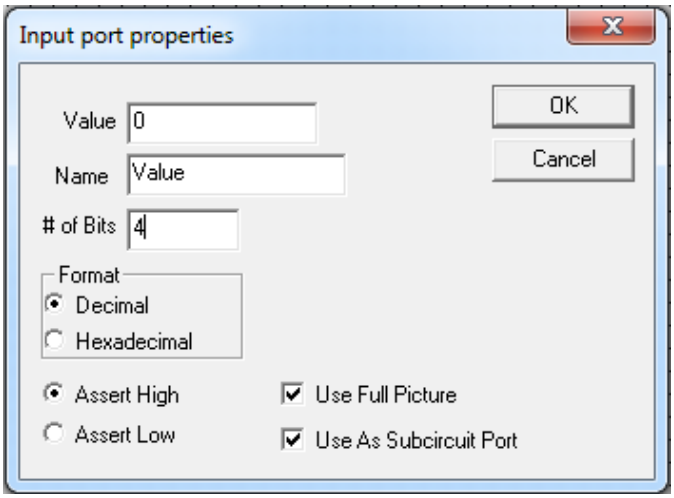


Figure 38. Input port properties window

Generate the command file, and modify as following:

; Resetting the simulator
Reset

; Setting the initial values, and counting up
Set Reset 1 0
Set Up/Down 0 0
Set Load 1 0

; Releasing from reset at 100 ns
Set Reset 0 100

; Running for 1000ns
Go 1000

; Counting down
Set Up/Down 1 1250
Go 500

; Testing parallel load function
Set Value 4 1800
Set Load 0 1850
Set Load 1 1900
Go 1000

; This is the end of the test.
; Total running time is (1000 + 500 + 1000) ns

Process the command file, and see the results!

Simulating a PLD

PLDs (Programmable Logic Devices) are for implementing application specific logics. In B2 Spice, the user can design simple PLDs to realize for simulation these special components. Although, in the device's list of the B' Spice we can find BCD to 7 segment decoder, but now we will implement this function in a PLD. The PLD will be controlled from the counter above, and the outputs will be visualized on a 7 segment decoder.

First, place a PLD from the menu: *Digital Devices → Programmable Device*, and a 7 segment display from the *Devices* window, *Digital Instruments* group (*Seven Seg Display*). For connecting to our counter designed previously, we need a splitter (*Splitter (1->4)*). If you place a programmable device, it is a small square, until it is not configured. For creating the data file of the PLD, create an empty file, and copy the contents of the next list.

IN0	IN1	IN2	IN3	/	a	b	c	d	e	f	g
0	0	0	0	/	1	1	1	1	1	1	0
1	0	0	0	/	0	1	1	0	0	0	0
0	1	0	0	/	1	1	0	1	1	0	1
1	1	0	0	/	1	1	1	1	0	0	1
0	0	1	0	/	0	1	1	0	0	1	1
1	0	1	0	/	1	0	1	1	0	1	1
0	1	1	0	/	1	0	1	1	1	1	1
1	1	1	0	/	1	1	1	0	0	0	0
0	0	0	1	/	1	1	1	1	1	1	1
1	0	0	1	/	1	1	1	1	0	1	1
0	1	0	1	/	0	0	0	0	0	0	0
1	1	0	1	/	0	0	0	0	0	0	0
0	0	1	1	/	0	0	0	0	0	0	0
1	0	1	1	/	0	0	0	0	0	0	0
0	1	1	1	/	0	0	0	0	0	0	0
1	1	1	1	/	0	0	0	0	0	0	0

The first row is a header, defines the inputs and outputs of the PLD. The next rows contain the input combinations and output responses to them. The segments to be activated must be at high level, and for the codes above 9 the all outputs are 0, for blank display. Every item in a row are separated by a TAB, and every rows are closed by an ENTER. Save the file. To configure

the PLD, double-click on it, then select *Set Data File*, and select the saved data file. If you close the window by clicking to OK, and the file was error-free, the PLD will be expanded, and the input and output pins will be on the left and right side of the square symbol.

Connect the elements as you see below, then check the correct operation of the circuit!

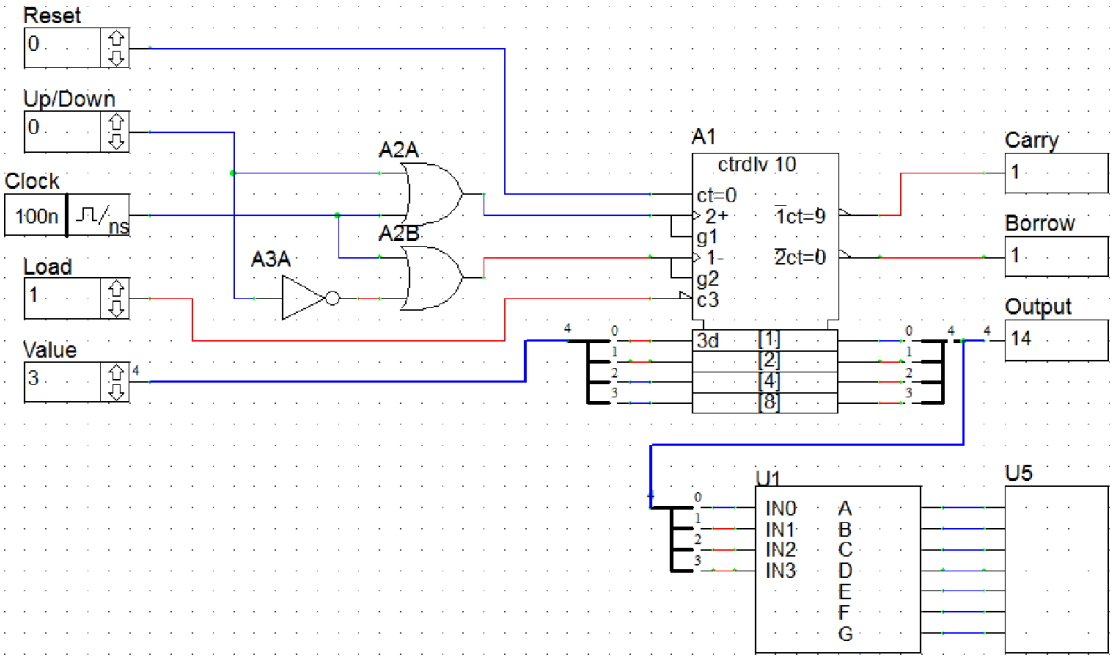


Figure 39. The designed circuit with 7 segment decoder and display

Part 1.

Before you start the individual work, you must perform the sample tasks described above in the section 'Example – Simulating a combinatorial logic circuit'.

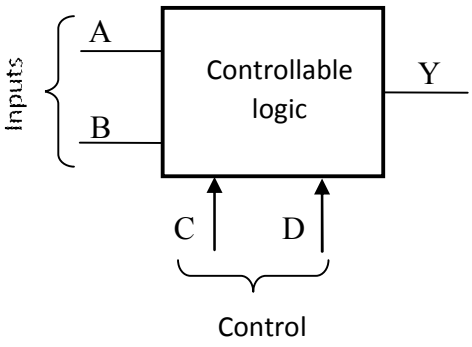
(Design a controllable logic! Simulate the implemented circuit! If the simulation indicates errors, for example bad design or logic hazard, fix it!

(In digital logic, a hazard in a system is an undesirable effect caused by either a deficiency in the system or external influences. Logic hazards are manifestations of a problem in which changes in the input variables do not change the output correctly due to some form of delay caused by logic elements. This results in the logic not performing its function properly. The three different most common kinds of hazards are usually referred to as static, dynamic and function hazards. A static hazard is the situation where, when one input variable changes, the output changes momentarily before stabilizing to the correct value. A dynamic hazard is the possibility of an output changing more than once as a result of a single input change.

Hazards are temporary problems, as the logic circuit will eventually settle to the desired function. Therefore, in synchronous designs, it is standard practice to register the output of a circuit before it is being used in a different clock domain or routed out of the system, so that hazards do not cause any problems. If that is not the case, however, it is imperative that hazards be eliminated as they can have an effect on other connected systems.)

Procedure

The symbol of the controllable logic and its function table are given below.



C	D	Y
0	0	A+B
0	1	A⊕B
1	0	$\overline{A \cdot B}$
1	1	Not defined

Figure 40. The controllable logic to be designed

Generate the Karnough-map, and recognize the appropriate groups! One of the possible grouping is shown below.

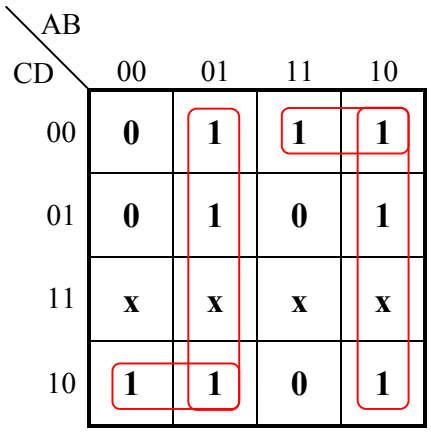


Figure 41. The controllable logic's Karnough map with an example of grouping

This grouping gives one of the minimal number of physical logic gates. The Boolean equation read from this map is:

$$Y = \overline{A} \cdot B + A \cdot \overline{B} + A \cdot \overline{C} \cdot \overline{D} + \overline{A} \cdot C \cdot \overline{D}$$

Implement this equation by 74XX logic gates, and simulate manually! Then use test vector file! Examine the output in the trace window! There are some glitches due to logic hazard. Eliminate these hazards!

You can use the test vector file listed below.

Time	A	B	C	D
0.0	0	0	0	0
100.0	0	1	0	0
200.0	1	0	0	0
300.0	1	1	0	0
400.0	0	0	0	1
500.0	0	1	0	1
600.0	1	0	0	1
700.0	1	1	0	1
800.0	0	0	1	0
900.0	0	1	1	0
1000.0	1	0	1	0
1100.0	1	1	1	0
1200.0	0	0	1	1
1300.0	0	1	1	1
1400.0	1	0	1	1
1500.0	1	1	1	1
1600.0	0	0	0	0

Steps for simulation:

1. Reset the simulator! *Digital Simulation → Reset Simulation*.
2. Set the simulation step to 100 ns! *Digital Simulation → Digital Simulation Options*, *Step Size* = 100. Don't exit!
3. Set the propagation delay of gates! *Default Propagation Delays*, *Typical*. Set *Override Device Propagation Delays*, then OK.
4. Run the test vector file! *Digital Simulation → Process Test Vector File*.
5. The results can be examined in the timing diagram. *View → Show Timing Diagram*. To zoom or walk on the timing diagram use the third row of the icons in the B2 Spice menu icons field.

The results are shown below. Where are the logic hazards? Why are they? How to eliminate them?

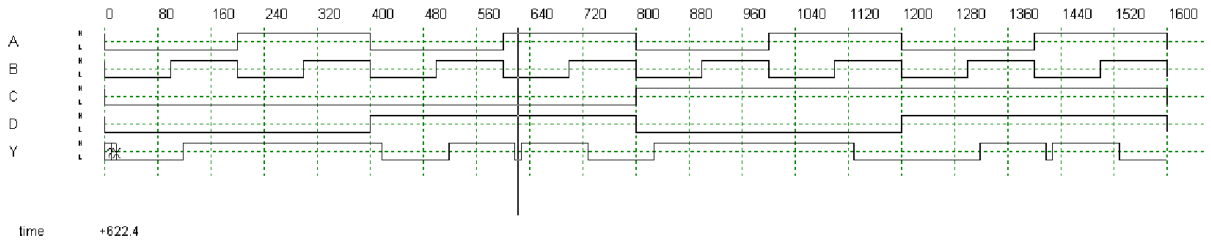


Figure 42. The output of the controllable logic with hazards

Part 2.

Design a 4 bit shift register from JK flip-flops! At reset, on the first flip-flop's output let be H, and on the others L. Test the circuit!

Procedure

Use the 7476 JK flip-flop! It has dynamic inputs J and K, and static inputs Preset and Clear. Use static inputs to maintain the initial states! The function table of 7476 is on the figure below.

Inputs					Outputs	
PR	**CLR**	**CLK**	**J**	**K**	**Q**	$\overline{\mathbf{Q}}$
L	H	X	X	X	H	L
H	L	X	X	X	L	H
L	L	X	X	X	H (Note 1)	H (Note 1)
H	H	⎍	L	L	Q_0	$\overline{Q}_0$
H	H	⎍	H	L	H	L
H	H	⎍	L	H	L	H
H	H	⎍	H	H	Toggle	

Figure 43. Function table of 7476 MS JK flip-flop

Part 3.

Design a pseudo-random generator from the shift register built previously! Test the circuit! Write a list of the values on the output as a hexadecimal number for one cycle!

Procedure

We can use the linear-feedback shift register (LFSR) method. Use a feedback with a 2 input XOR gate (7486) from the third and fourth bits of the shift register! Use a combiner from 4 bits to a bus, and an output port to examine the output! If on all the outputs of the shift register are 0 at reset, the contents of the shift register remains all 0. So you must set at least one output to 1 to start the pseudo-random number generation.

Part 4.

Design a shortened cycle counter from 74192, details are given by the instructor. If there is no other request, the counter let count from 3 to 7, then return to 3. Test the circuit!

Procedure

The 74192 is a counter with parallel load, so we can prepare the value of 3 to start the counting from this value. The load is necessary, if the counter leaves the value of 7, so we must create (to combine from output bits) a LOAD signal, if the counter reaches 8. In that moment, the load will be active, and the prepared value (3) will be the new output instead of 8. As the value of Q_D is zero before reaching 8 (0101, 0110, 0111 $\rightarrow$ 1000), so we need only the QD to control the LOAD input. But load is active low (and the outputs are active high), so we need an inverter to maintain the proper signal transition. Note, that the value on the output for a short but perceptible time will be 8, due to the propagation delays. So this solution is not professional, but used in practice.

Questions

1. What are the advantages of the simulation? And what are the advantages of building a circuit?
2. What is logic hazard, and what types are known?
3. What are the functions of 74192 counter?
4. How to build a simple pseudo-random generator?
5. What simulation modes and tools were mentioned in this manual?

Useful links:

http://www.ti.com/lit/ds/symlink/sn74192.pdf

http://www.ti.com/lit/an/scta036a/scta036a.pdf

http://www.ti.com/lit/ml/sdyz001a/sdyz001a.pdf

http://www2.elo.utfsm.cl/~lsb/elo211/aplicaciones/katz/chapter3/chapter03.doc4.html

Experiment 5.

Digital to analog converters

Objectives

- To study the operation of basic DAC topologies.
- Gathering experience how to connect digital and analog world.
- To study, how to measure static DAC parameters.

Components

None

Instruments

Items	Comment
Power supply	dual
Digital multimeter	
Function generator	
Digital oscilloscope	dual channel
DAC demo panel	

Background

The data converters

There are two basic type of converters, digital-to-analog (DAC or D/A) and analog-to digital (ADC or A/D). Their purpose is fairly straightforward. In the case of DACs, they output an analog voltage that is in a proportion to a reference voltage, the proportion based on the digital word applied. In the case of the ADC, a digital representation of the analog voltage that is applied to the ADCs input, is outputted, the representation is proportional to a reference voltage. In both cases the digital word is almost always based on a binary weighted proportion. The digital input or output is arranged in words of varying widths, referred to as bits, typically anywhere from 6 bits to 24 bits.

Two major features of the converters are: the resolution and the accuracy. It is important to note, that the accuracy and resolution are different. The resolution of a converter is the number of bits in its digital word. The accuracy is the number of those bits that meet the specifications. For instance, a DAC might have 16 bits of resolution, but might only be monotonic to 14 bits. This means that the assured accuracy of the DAC will be no better than 14 bits. This is not to say that the other bits are irrelevant. With further processing, typically filtering, often the accuracy can be improved. While these terms are similar and sometimes used interchangeably, the distinction between them should be remembered.

The digital to analog converters

The integrated DACs typically have the converter itself and a collection of support circuitry built into the chip. In this experiment, we will examine only basic parts of various converters.

On the figure below, you can see the scheme of the DAC. The DAC gets the binary number on the digital input, and converts it to an analog voltage at the output. The DAC must has a reference voltage (V_{REF}), which will be multiplied by the binary number on the digital input. The V_{REF} determines the full scale (FS), and it equal to it. Of course, the DAC needs power supply, between V_{DD} and V_{SS}.

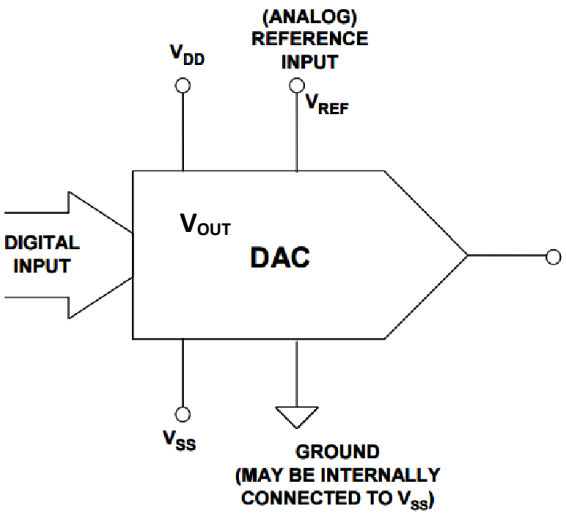


Figure 44. Simple D/A converter

The output voltage of the N-bit DAC is (where N is the number of the bits of the DAC):

$$V_{OUT} = \frac{V_{REF}}{2^N} . (B_{N-1} \cdot 2^{N-1} + B_{N-2} \cdot 2^{N-2} + \cdots + B_0 \cdot 2^0)$$

For example, if the N=3, there are 3 bits on the digital input. The MSB is the B2, the LSB is the B0. They binary weight is equal to 2^i, where the "i' is its index (2, 1 or 0). So, if the Digital input is "011', from MSB to LSB, the VOUT is:

$$V_{OUT} = \frac{V_{REF}}{2^3} . (B_2 \cdot 2^2 + B_1 \cdot 2^1 + B_0 \cdot 2^0) = \frac{V_{REF}}{8} . (0 \cdot 4 + 1 \cdot 2 + 1 \cdot 1) = V_{REF} \cdot \frac{3}{8}$$

For the values from "000' to "111' are in the table below.

Digital input value (MSB to LSB)	Output voltage
000	V_{REF}·(0/8) = 0·V_{REF} = 0
001	V_{REF}·(1/8) = 0.125·V_{REF}
010	V_{REF}·(2/8) = 0.25·V_{REF}
011	V_{REF}·(3/8) = 0.375·V_{REF}
100	V_{REF}·(4/8) = 0.5·V_{REF}
101	V_{REF}·(5/8) = 0.625·V_{REF}
110	V_{REF}·(6/8) = 0.75·V_{REF}
111	V_{REF}·(7/8) = 0.875·V_{REF}

Table 6. 3-bit DAC output values

Note, that the FS (V_{REF}) is never reached. Representing the relationship of the input values and output voltages, we get the transfer function below:

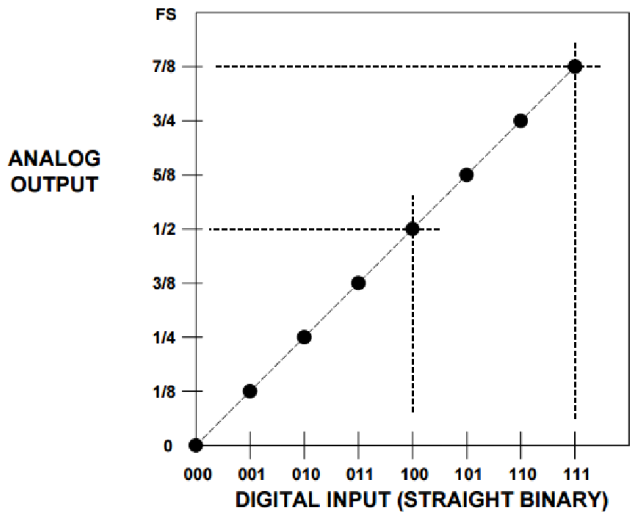


Figure 45. Transfer function for ideal unipolar 3-bit DAC

Static errors of D/A converters

In the following, we will discuss shortly the static errors of D/A converters. Static errors, that are the errors that affect the accuracy of the converter when it is converting static (DC) signals, can be completely described by just four terms. These are offset error, gain error, integral nonlinearity and differential nonlinearity. Each can be expressed in LSB units or sometimes as a percentage of the FS.

Offset error

The offset error as shown in the figure below is defined as the difference between the nominal and actual offset points. For a DAC this point is the step value when the digital input is zero. This error affects all codes by the same amount and can usually be compensated for by a trimming process. If trimming is not possible, this error is referred to as the zero-scale error.

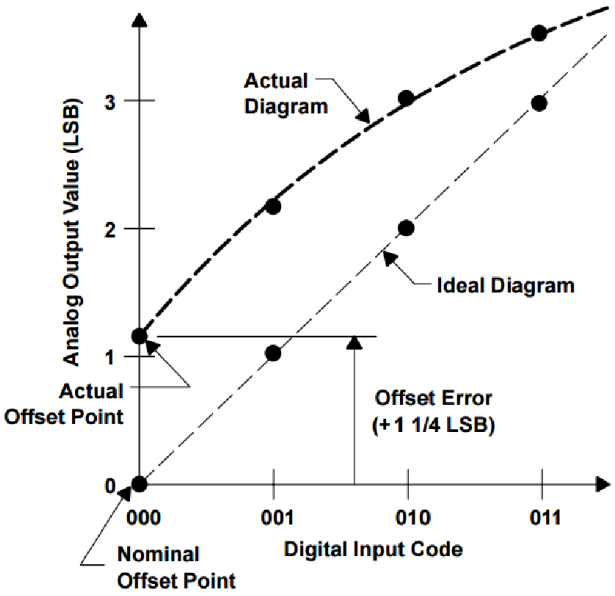


Figure 46. Offset error of a 3-Bit DAC

The gain error

The gain error shown in figure below is defined as the difference between the nominal and actual gain points on the transfer function after the offset error has been corrected to zero. For a DAC, the gain point is the step value when the digital input is full scale. This error represents a difference in the slope of the actual and ideal transfer functions and as such corresponds to the same percentage error in each step. This error can also usually be adjusted to zero by trimming.

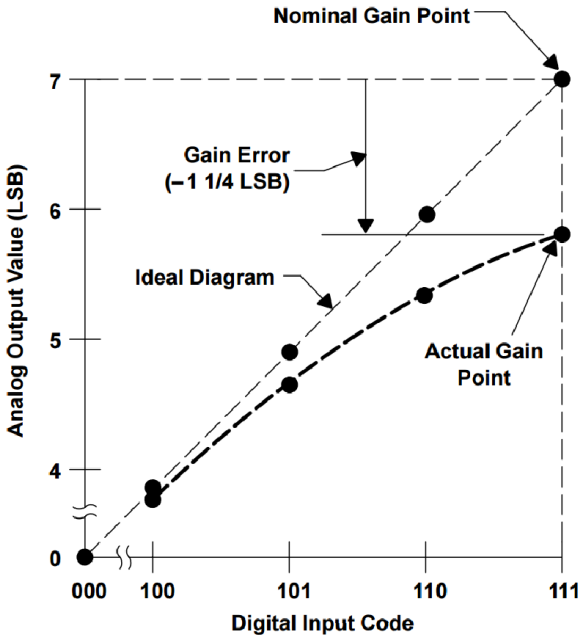


Figure 47. Gain error of a 3-bit DAC (after correction of the offset error)

Differential nonlinearity (DNL) error

The differential nonlinearity error shown in the figure below (sometimes seen as simply differential linearity) is the difference between an actual step height for a DAC, and the ideal value of 1 LSB. Therefore if the step width or height is exactly 1 LSB, then the differential nonlinearity error is zero. If the DNL exceeds 1 LSB, there is a possibility that the converter can become non-monotonic. This means that the magnitude of the output gets smaller value for an increased magnitude of the input.

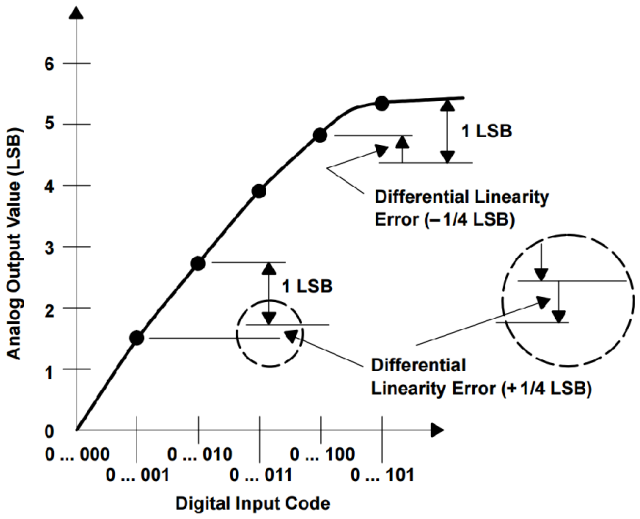


Figure 48. Differential linearity error of a DAC

Integral Nonlinearity (INL) Error

The integral nonlinearity error shown in figure below (sometimes seen as simply linearity error) is the deviation of the values on the actual transfer function from a straight line. This straight line can be either a best straight line which is drawn so as to minimize these deviations

or it can be a line drawn between the end points of the transfer function once the gain and offset errors have been nullified. The second method is called end-point linearity and is the usual definition adopted since it can be verified more directly. For a DAC they are measured at each step. The name integral nonlinearity derives from the fact that the summation of the differential nonlinearities from the bottom up to a particular step, determines the value of the integral nonlinearity at that step.

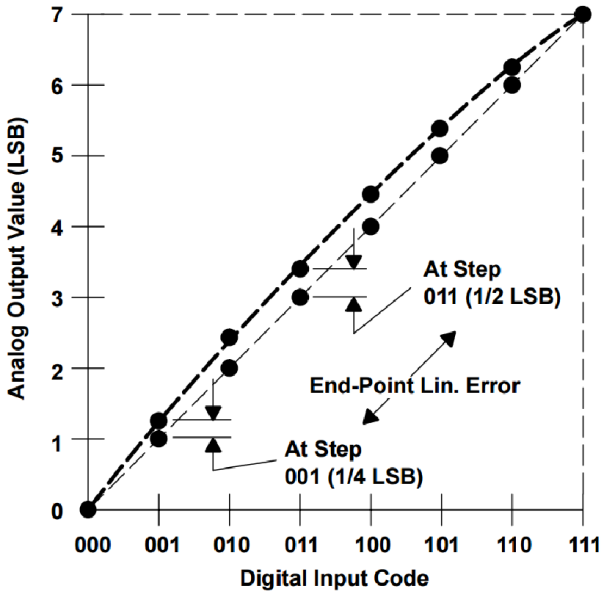


Figure 49. End-point linearity error of a 3-bit DAC (offset error and gain error are zero)

Absolute Accuracy (Total) Error

The absolute accuracy or total error of a DAC is shown in figure below. It is the maximum value of the difference between an analog output value and the point determined by its digital input code value. It includes offset, gain, and integral linearity errors.

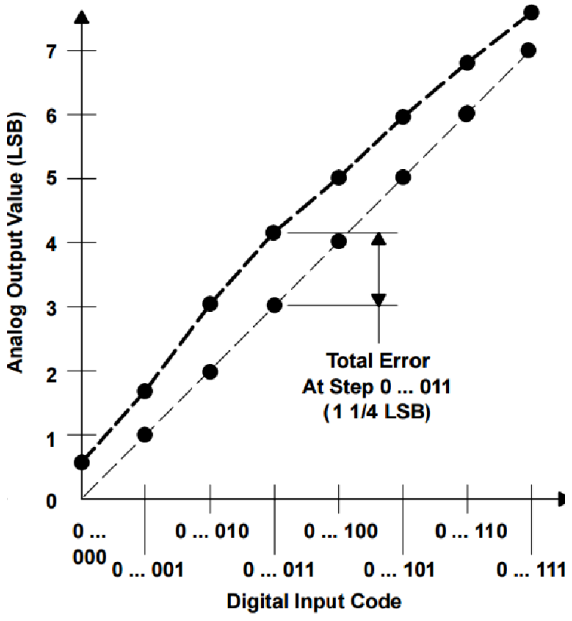


Figure 50. Total error of a 3-bit DAC

A few DAC topologies

There are many DAC topologies, we can examine only a few of them. Further topologies can be found at the useful links.

The current mode binary weighted DAC

The current mode binary weighted DAC consists of N current sources (N the number of the bits of the DAC), of which source currents are binary weighted. If MSB has the current value of I, the next bit has I/2, and so on: the LSB has a current of $I/2^{(N-1)}$. If a bit at the input is 1, the output current of the DAC will increase by the current of the actual bit, and if the bit is 0, the output current will not be influenced. The topology is shown below.

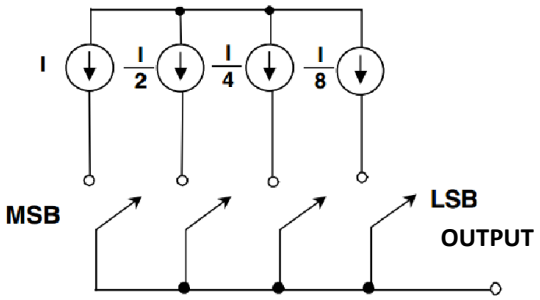


Figure 51. Current-mode binary-weighted DAC

The output current may convert to voltage by an active current to voltage converter. This topology is rarely used, because difficult to fabricate in integrated circuits, due to the large current ratios for high resolution.

The R-2R ladder DAC

The R-2R ladder DAC uses only two different resistor values, and their ratio is 1:2. This topology is very popular, because an N-bit DAC requires only 2N resistors, and they are quite easily trimmed in integrated circuits. The R-2R ladder (see figure below) has a very interesting behavior: if we examine the network with grounded resistors at the lower side, and we look "into' the circuit at the junctions to left, the resultant resistance will be always R, regardless the size of the ladder.

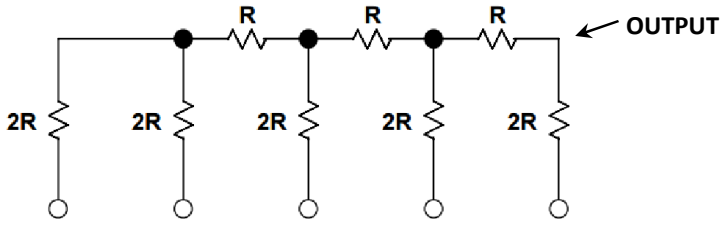


Figure 52. The R-2R ladder

There are many possibilities to use the R-2R ladder as a DAC, one of them is shown on the figure below. The part framed with red is the R-2R ladder, and the part framed with blue is the analog switch section.

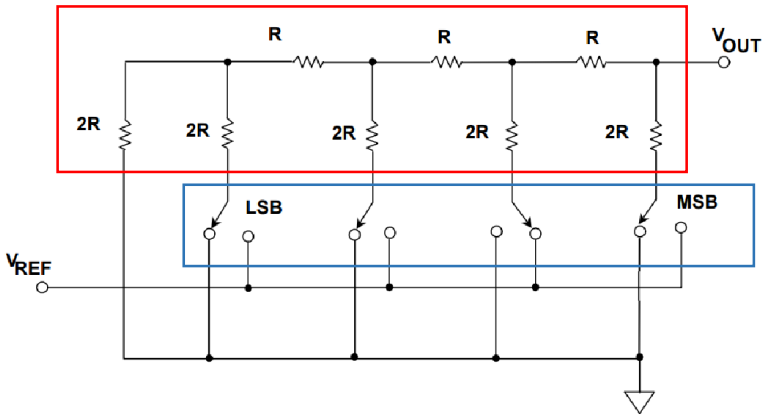


Figure 53. DAC, which consists of an R-2R ladder

At the lower side of the 2R resistors there are switches, which can switch either GND or V_{REF} to these points. The V_{OUT} will be

$$V_{OUT} = MSB \cdot \frac{V_{REF}}{2^1} + (MSB - 1) \cdot \frac{V_{REF}}{2^2} \ldots + LSB \cdot \frac{V_{REF}}{2^N},$$

if the ladder has N stages (and there is a 2R resistor at the end to the ground). The resultant resistance at the output will be always R, because the V_{REF} and GND points are considered equipotential in this analysis. We can use this circuit as a current source to the ground, in this case we need a current-to-voltage converter at the output, and $I_{OUT} = V_{OUT}/R$, the output resistance of the source is R. The current to voltage converter can be implemented by an opamp and a resistor.

PWM DAC

In digital systems it is very simple to produce a PWM (Pulse Width Modulated) signal (see figure below). The data is represented by the ratio of the ON time to the period (also known as the duty cycle). A given ON time corresponds to an average DC voltage, which is linearly proportional to the duty cycle. For implementing the PWM DAC, either the PWM period is fixed and the duty cycle is varied or vice versa. In most cases, the PWM DAC is implemented with fixed PWM period and variable duty cycles.

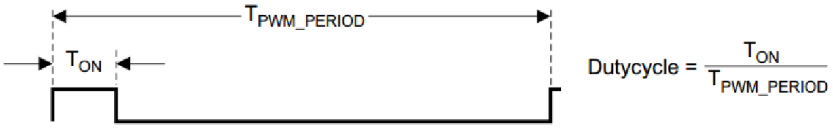


Figure 54. The PWM signal, and the definition of duty cycle

On passing the PWM signal through a low-pass filter (LPF), a DC voltage with reasonable ripple is generated. The low pass filter can be implemented by a simple R-C circuit, or by a given order active filter. If a high order filter is used, the response time of the output increases.

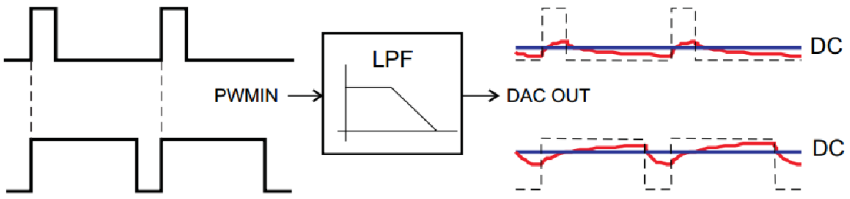


Figure 55. Filtered PWM signal

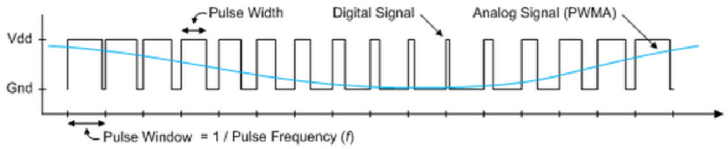


Figure 56. An analog signal generated by PWM DAC and LPF

The PWM signal can be obtained, if we use a digital (magnitude) comparator. To the input A we apply a constant number, which determines the duty cycle, and to the input B we apply a counter output, which determines the PWM period. On the output A<B will be a PWM signal, which will have a high level, while the counter's output on the B inputs is lower than the value on A inputs. So, the duty cycle is proportional to the input value A.

DAC application

There are many DAC applications, because many digital systems need to produce analog signals or controlling analog circuits/devices. The figure below shows a simple DAC application, in which the I_{OUT} is set by the 13-bit DAC. The I_{OUT} is:

$$I_{OUT} = \frac{V_{REF}}{R} \cdot \frac{N}{2^{13}}$$

where N is the DAC input code. This application gives the possibility to set the analog output current with relatively high resolution by – for example – a microcontroller.

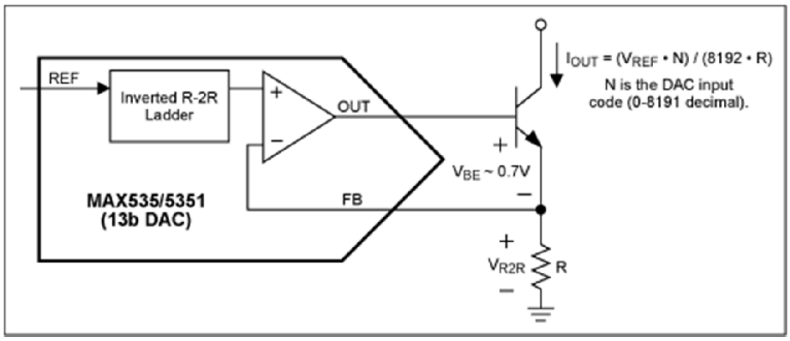
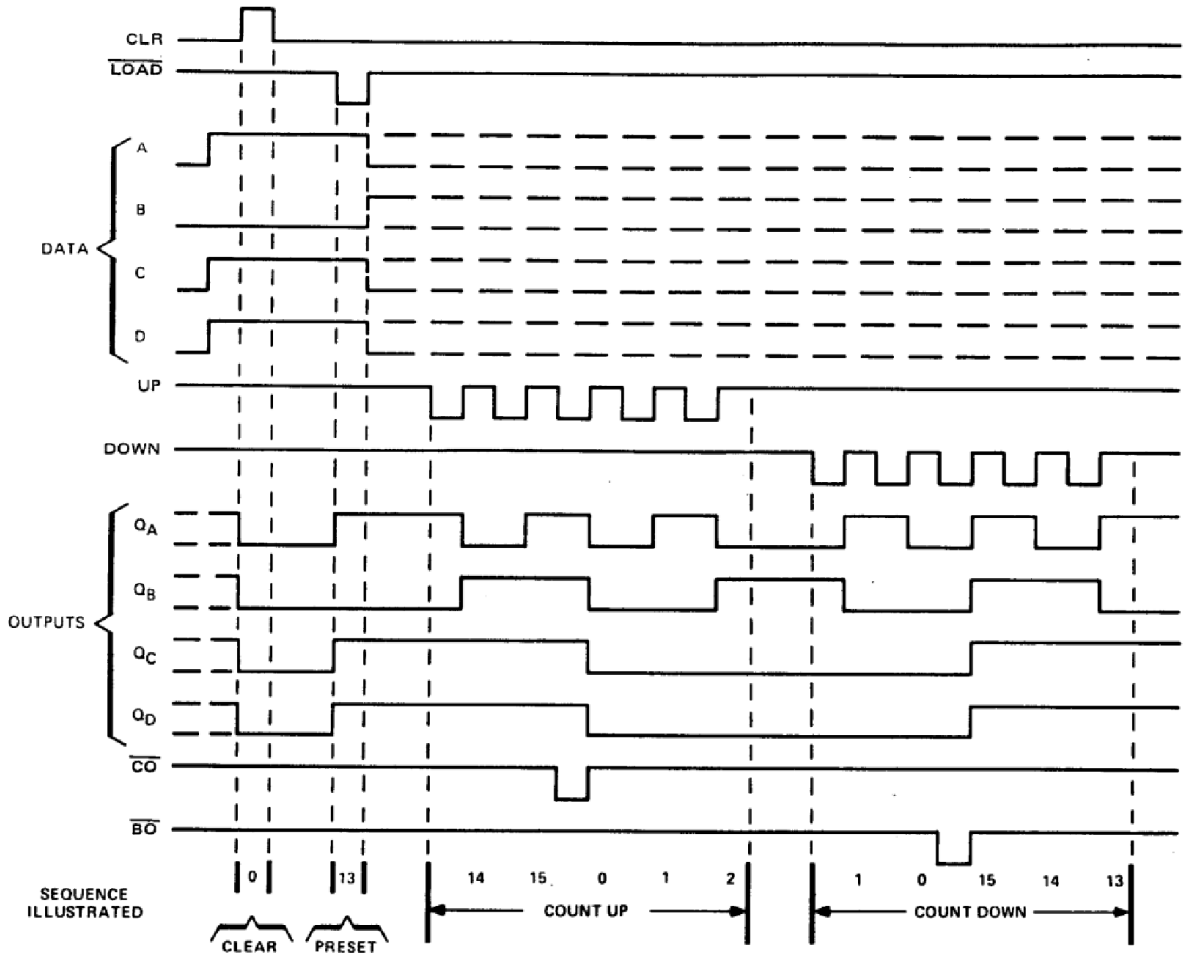


Figure 57. DAC controlled current source

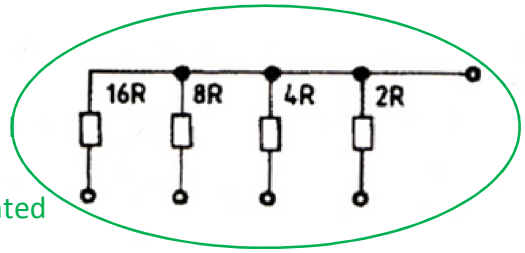
Count sequences of the 74193 binary counter



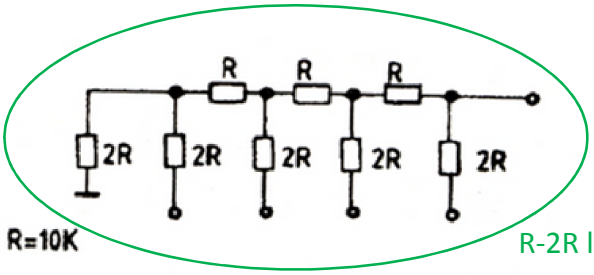
Function table of the 7485 magnitude comparator. The cascading inputs are not used in this manual.

COMPARING INPUTS				CASCADING INPUTS			OUTPUTS		
A3, B3	A2, B2	A1, B1	A0, B0	A > B	A < B	A = B	A > B	A < B	A = B
A3 > B3	X	X	X	X	X	X	H	L	L
A3 < B3	X	X	X	X	X	X	L	H	L
A3 = B3	A2 > B2	X	X	X	X	X	H	L	L
A3 = B3	A2 < B2	X	X	X	X	X	L	H	L
A3 = B2	A2 = B2	A1 > B1	X	X	X	X	H	L	L
A3 = B3	A2 = B2	A1 < B1	X	X	X	X	L	H	L
A2 = B3	A2 = B2	A1 = B1	A0 > B0	X	X	X	H	L	L
A3 = B3	A2 = B2	A1 = B1	A0 < B0	X	X	X	L	H	L
A3 = B3	A2 = B2	A1 = B1	A0 = B0	H	L	L	H	L	L
A3 = B3	A2 = B2	A1 = B1	A0 = B0	L	H	L	L	H	L
A3 = B3	A2 = B2	A1 = B1	A0 = B0	X	X	H	L	L	H
A3 = B3	A2 = B2	A1 = B1	A0 = B0	H	H	L	L	L	L
A3 = B3	A2 = B2	A1 = B1	A0 = B0	L	L	L	H	H	L

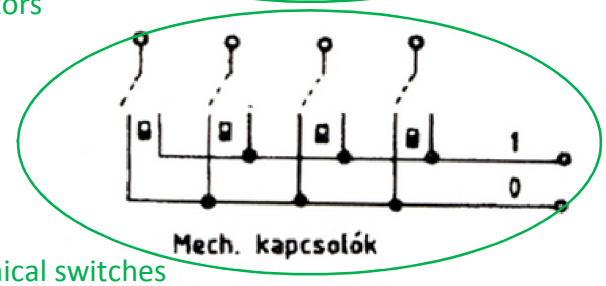
The circuits of the DAC demo panel



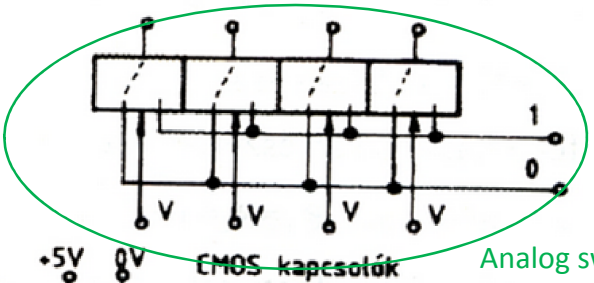
Binary weighted resistors



R-2R ladder

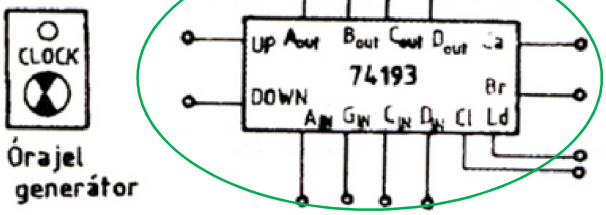


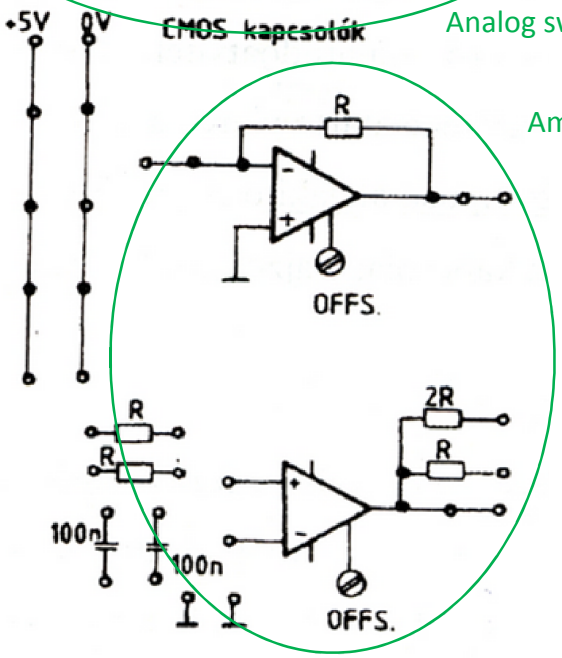
Mechanical switches



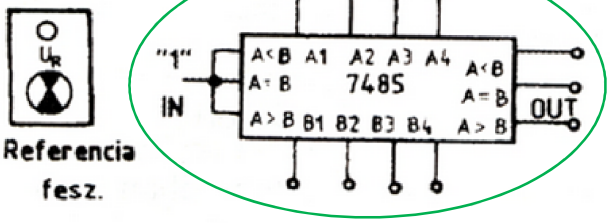
Analog switches

Binary up/down counter





Amplifiers



Magnitude comparator

Part 1.

Verify all functions of the demo panel!

Procedure

Apply the power supply, then verify the analog circuits:

- Verify the operation of the operational amplifiers as a follower! Adjust the offset voltage to 0 V!
- Verify the operation of the reference source, set it to -1.6 V!

Then verify the digital and mixed signal circuits:

- Verify the operation of the clock source by an oscilloscope!
- Verify the operation of the CMOS switches! The control voltage is 0/5 V, the analog voltage at the inputs can't exceed ±5 V! (For example apply DC voltage to the common points, and check if this voltage appears and disappears on the '1' and '0' lines. If a CMOS switch's V control line is 0, the switch closes the line '0' to the output, if V is 1, the line '1' will be switched to the output.)
- Verify the operation of the 74193 counter! Apply the clock to the UP clock input, set LOAD and DOWN to high and CLR to low! Examine the outputs! Then apply the clock to DOWN, and high level to UP input, and examine the outputs!
- Verify the operation of the magnitude comparator! Apply 0 to all A inputs, and all B inputs. The output A=B must be high, other outputs must be low! Then apply 1 to all B inputs, only the A<B output will be high. Then change the control of A and B, now only the A>B will be high.

If something does not operate, report it to the instructor!

Part 2.

Build a current mode binary weighted DAC, and examine it's operation!

Procedure

Build a current mode binary weighted DAC, which is shown on the figure below. Use the mechanical switches! Write the output values in a table, while input changes 0000 to 1111 in 16 steps. Calculate the gain error and the (integral) linearity error in % and LSB!

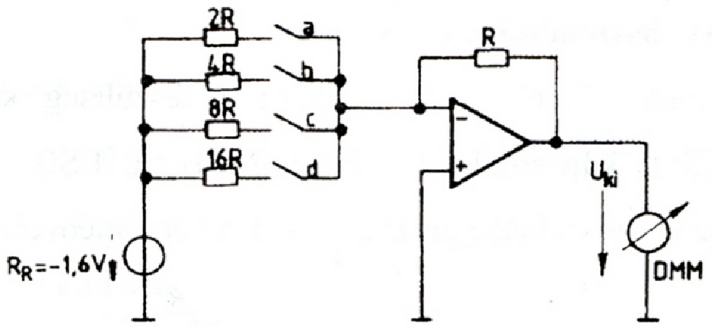


Figure 58. Current mode, binary weighted DAC

Change the mechanical switches to CMOS switches! The CMOS switches will be controlled by the 74193 counter in UP counting mode. Examine the DAC output on the oscilloscope, make sure the correct triggering for stable display! Draw the waveform of the signal!

Part 3.

Build an R-2R ladder DAC, and examine its operation!

Procedure

Build an R-2R ladder DAC with an operational amplifier, as you can see below! Use mechanical switches! Write the output values in a table, while input changes 0000 to 1111 in 16 steps. Calculate the gain error and the (integral) linearity error in % and LSB!

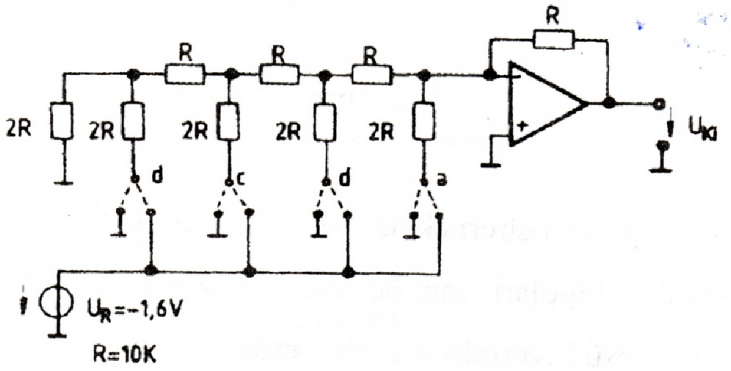


Figure 59. The R-2R ladder DAC

Then examine the DAC as a multiplier! Change U_R to a sinusoidal signal from a function generator, U_P = 1 V, f = 1 kHz. Apply various input codes, and examine the output!

Change the mechanical switches to CMOS switches! Use U_R = -1.6 V DC! The CMOS switches will be controlled by the 74193 counter in UP counting mode. Search glitches at the transients! Draw a glitch waveform!

Facultative task: Use the DAC as a modulator! U_R is a sinusoidal signal to be modulated, U_P = 1 V, f = 2 kHz, the modulator signal is the 'ramp' produced by the counter. Examine the output signal! (The f_{CLOCK} not need to be too high.)

Part 4. (Facultative task)

Build a PWM DAC, and examine its operation!

Procedure

Build a PWM DAC, using 74193, 7485, and a low pass filter built of an opamp and R and C elements! The output voltage let be V_{OUT} = IN·100 mV, where IN is the input code value! Examine the output signal before and after filtering at various input codes!

Questions

1. What is the function of the D/A converters? Explain their transfer function!
2. Which are the static errors of the D/A converters? Specify them!
3. Explain the schematic and the operation of a current mode binary weighted DAC!
4. Draw the schematic of an R-2R ladder DAC, and explain, how it can be used as a DAC?
5. What is the principle of the PWM DAC? How does it work?
6. How does work the magnitude comparator? How can it be used as the element of a PWM DAC?

Useful links:

http://www.electronics.teipir.gr/personalpages/papageorgas/download/mcu_embedded/TI_MSP430_led_SOLAR/Industrial%20Process%20Control%205_DAC%20architecture.pdf

http://www.uio.no/studier/emner/matnat/ifi/INF4420/v13/undervisningsmateriale/inf4420_v13_09_dataconverters_print.pdf

http://www.analog.com/library/analogDialogue/archives/39-06/Chapter%202%20Sampled%20Data%20Systems%20F.pdf

http://www.analog.com/media/en/training-seminars/tutorials/MT-015.pdf

Experiment 6.

Voltage regulators

Objectives

- To become familiar with the basic voltage converters, such as linear regulator, switching mode regulator and switched capacitance regulator.
- To get to know, how to measure basic voltage regulator parameters, and signals in the switching mode voltage converters.

Components

Component	Description	Number of items	Comment
LM7805	voltage regulator	1	or equivalent
LM2575-5.0	voltage regulator	1	or equivalent
ICL7660	voltage converter	1	or equivalent
1N5819	Schottky diode	1	or equivalent
1N4148	diode	2	or equivalent
330 µH	inductor	1	
10 µF	capacitor	2	electrolytic, min. 25 V
100 µF	capacitor	1	electrolytic, min. 16 V
330 µF	capacitor	1	electrolytic, min. 10 V
100 nF	capacitor	1	ceramic
330 nF	capacitor	1	ceramic
0.1 Ω	resistor	1	
100 Ω	resistor	1	wire wound

Instruments

Items	Comment
Power supply	dual
Digital multimeter	
Digital oscilloscope	dual channel

Background

Power supplies

All electronic systems require power supplies to operate. This part of the system design is often overlooked, but, if the power supply is not fully suitable for the system, its performance will be degraded. The power supply converts one form of electrical energy to another. For example at the input there is 230 V AC, and at the output 5 V DC. The load sinks current to operate, and this current is ensured by the power supply, and the power supply sinks current from its input, from the energy source.

Although there are DC and AC power supplies (depending on their output voltage), in this manual we focus on the DC supplies. The DC power supplies supply a voltage of fixed polarity (either positive or negative) to its load. The DC to DC supplies are powered from a DC source (for example from a battery), and AC to DC supplies are powered from AC source (for example from the 230 V power line). If the input power is AC, it must to convert to DC by rectification. In the figure below, you can see the block diagram of a typical AC to DC supply.

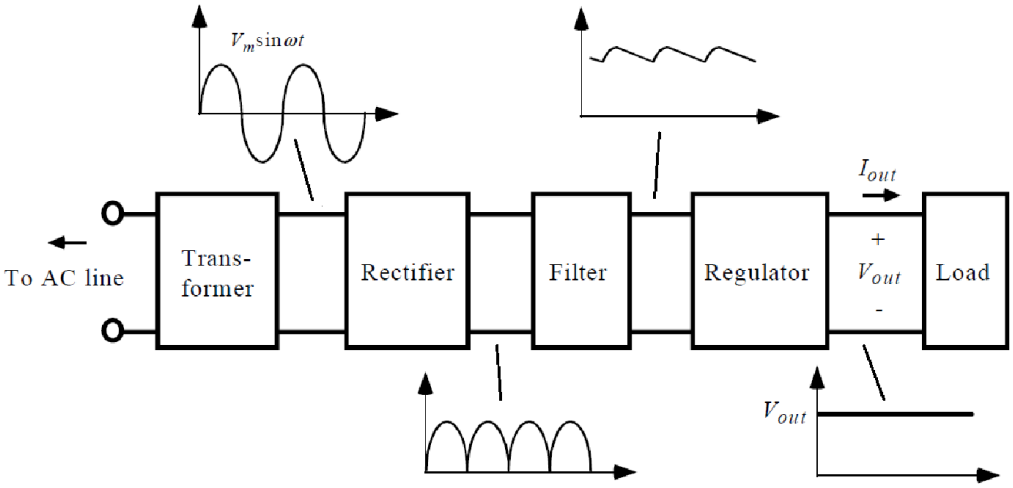


Figure 60. The typical AC to DC power supply

The transformer reduces the line voltage to an appropriate level. The rectifier converts the AC voltage to DC voltage, and the filter smoothes the voltage from the rectifier. At the filter's output there is a DC voltage with a certain degree of ripple, but this voltage is not suitable for most circuits to be powered. This voltage must be regulated by the regulator, which stabilizes the output voltage – despite many effects – to an appropriate level to the load. These effects are, for example: change of the input voltage, of the load current, or of the temperature, etc…

The regulator may be linear (analog/dissipative) or switching mode. Linear regulators process the input power directly, with all active power conversion components operating in their linear operating regions. In switching regulators, the input power is converted to pulses before processing, by components that operate predominantly in non-linear modes (e.g., transistors spend most of their time in cutoff or saturation). Power is 'lost' (converted to heat) when components operate in their linear regions and, consequently, switching converters are usually more efficient than linear converters, because their components spend less time in linear operating regions.

The advantages of the linear regulators are:

- low noise
- low price
- low dropout voltage achievable
- low complexity

and the advantages of the switching mode regulators are:

- high efficiency
- smaller size at same output power

Because the AC line transformer is massive and bulky, it is undesirable. In the switched mode power supplies (SMPS, see figure below) first step is the rectification and filtering. Then this high level DC voltage will be converted to high frequency AC pulses, and will be transformed. If the frequency is higher, the transformer can be smaller and lighter. Then the transformed voltage is rectified, filtered and stabilized. Note, that the DC output remains galvanically independent from the input, because of the used high frequency transformer and optoisolator.

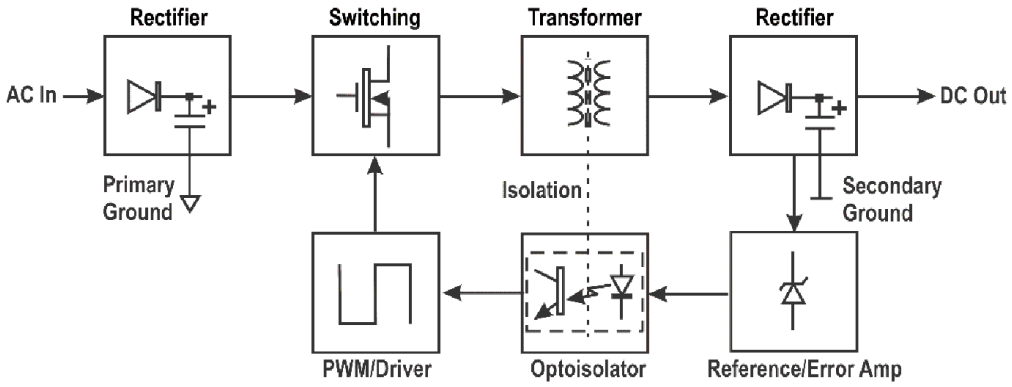


Figure 61. Modern switched mode power supply

Voltage regulators

Linear voltage regulator

The functional diagram of the linear voltage regulator can be seen on the figure below. It operates by using a voltage-controlled current source to force a fixed voltage to appear at the regulator's output terminal.

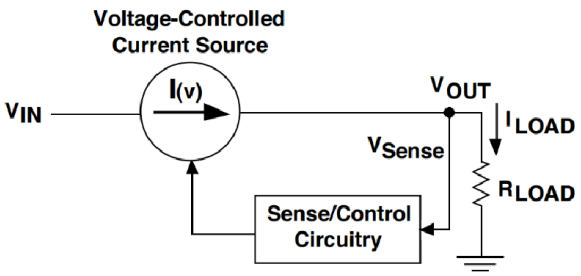


Figure 62. Linear voltage regulator

The control circuitry must monitor (sense) the output voltage, and adjust the current source (as required by the load) to hold the output voltage at the desired value. The design limit of the current source defines the maximum load current the regulator can source and still maintain regulation. The output voltage is controlled using a feedback loop, which requires some type of compensation to assure loop stability. Most linear regulators have built-in compensation, and are completely stable without external components. Some regulators (like Low-Dropout types), do require some external capacitance connected from the output lead to ground to assure regulator stability. Another characteristic of any linear regulator is that it requires a finite amount of time to 'correct' the output voltage after a change in load current demand. This 'time lag' defines the characteristic called transient response, which is a measure of how fast the regulator returns to steady-state conditions after a change of the load.

The operation of the control loop in a typical linear regulator will be detailed using the simplified schematic diagram in the figure below (the function of the control loop is similar in all of the linear regulator types).

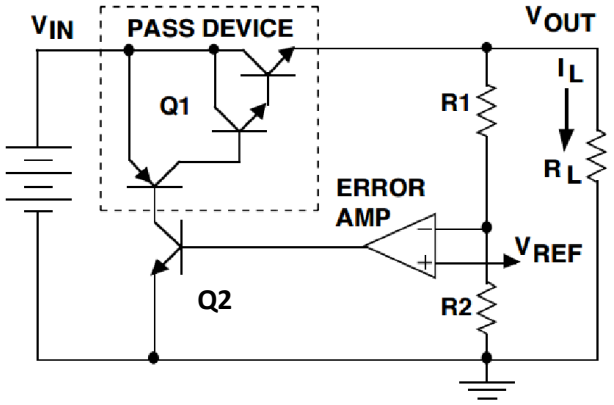


Figure 63. Simplified schematic diagram of a linear voltage regulator

The pass device (Q1) in this regulator is made up of an NPN Darlington-transistor, driven by a PNP transistor. The current flowing out the emitter of the pass transistor (which is also the load current IL) is controlled by Q2 and the voltage error amplifier. The current through the R1, R2 resistive divider is assumed to be negligible compared to the load current. The reference voltage source can be a simple Zener diode, or a precision voltage reference device. The feedback loop which controls the output voltage is obtained by using R1 and R2 to 'sense' the output voltage, and applying this sensed voltage to the inverting input of the voltage error amplifier. The non-inverting input is tied to a reference voltage, that results: the error amplifier will constantly adjust its output voltage (and the current through Q1) to force the voltages at its inputs to be equal to each other. The feedback loop action continuously holds the regulated output at a fixed value which is a multiple of the reference voltage (as set by R1 and R2), regardless of changes in load current. The output voltage is:

$$V_{OUT} = V_{REF} \cdot \frac{R_1 + R_2}{R_2}$$

The changing output voltage is sensed through R1 and R2 and appears as an 'error signal' at the input of the error amplifier, causing it to correct the current through Q1. Note, that this circuit is the simplified version of the circuit can be found in 78xx series integrated 3-terminal voltage regulator family.

The power dissipation of a linear voltage regulator can be calculated as:

$$P_D = (V_{IN} - V_{OUT}) \cdot I_L,$$

if the power consumption of the circuits of the regulator is neglected. So, at a constant I_L, the P_D is proportional to the difference of the input and output voltages. If we want to reduce this value, we have to decrease the input voltage. The difference between the input and output voltages is named as dropout (or simply drop) voltage, and it has a minimal specified value, so the input voltage (and the power dissipation) can't be reduced to any degree.

The efficiency can be calculated as following (if we neglect the loss of the regulator itself):

$$\eta = \frac{P_{OUTPUT}}{P_{OUTPUT} + P_{LOSS}} = \frac{V_{OUT} \cdot I_{OUT}}{V_{OUT} \cdot I_{OUT} + (V_{IN} - V_{OUT}) \cdot I_{OUT}} = \frac{V_{OUT}}{V_{IN}}.$$

This function is shown on the figure below.

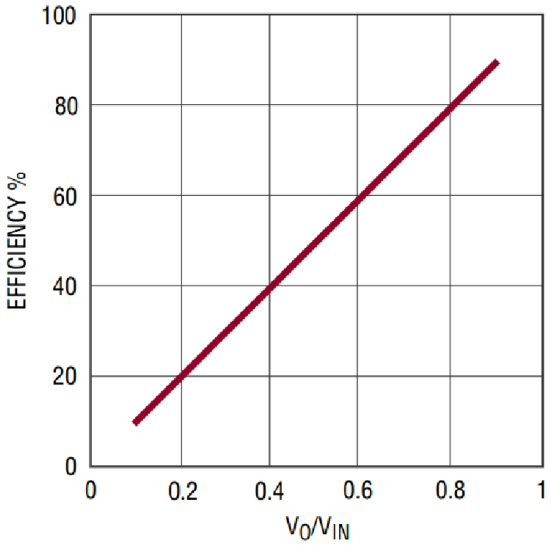


Figure 64. Maximum efficiency of a linear regulator

For example, if we use a voltage regulator which provides 5 V at the output, and the input voltage is 10 V, the efficiency will be 50%. If output current is 500 mA, the output power will be 2.5 W, and the voltage regulator will dissipate (will generate heat) another 2.5 W. It's not a too good ratio…

Integrated voltage regulators can contain several protection circuits. Most important is the overcurrent (or overload) protection, which reduces the output voltage, if the nominal output current is exceeded, thus, this protection protects the pass device. In most of the integrated linear voltage regulators has a 'foldback' current protection circuit, which means that both the output current and the output voltage will be reduced in case of overload. The prime purpose

of foldback current limiting in linear power supplies is to keep the output transistor within its safe power dissipation limit. An another important protection is the overheating protection, which protects the circuit from high temperatures due to dissipation. If the temperature rises over the allowed limit, the output current will be limited to protect the device.

Linear voltage regulators are manufactured in a wide variety. The input and output voltages and output currents are rather diversified. There are fixed and variable output voltage versions. Some of them are designed for regulating positive voltage, and others for regulating negative voltage. There are normal and LDO (low dropout) regulators: the LDO regulators are designed for operating at very low dropout voltages, as low as 50 mV. In the catalogs we can find regulators, which have an enable pin to switch on and off the output voltage, and so on…

A simple, typical linear voltage regulator is the 78xx family mentioned above. Its typical application is shown below. The regulator requires only two additional component, a capacitor to its input, and an another to its output.

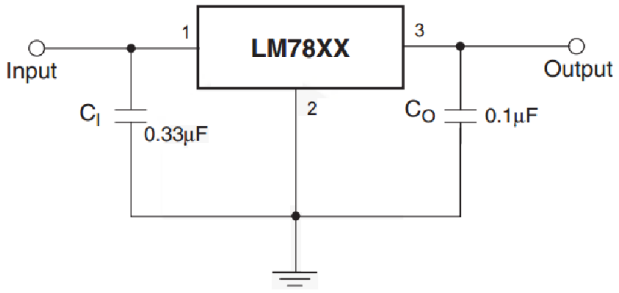


Figure 65. Typical application of the 78xx voltage regulator family

Switching mode voltage regulator

The most basic switcher topologies require only one transistor which is essentially used as a switch, one diode, one inductor, a capacitor across the output, and for practical but not fundamental reasons, another capacitor across the input terminals. A practical converter, however, requires a control section comprised of several additional elements, such as a voltage reference, error amplifier, comparator, oscillator, and switch driver, and may also include optional features like current limiting and shutdown capability.

Depending on the power level, modern IC switching regulators may integrate the entire converter except for the main magnetic element(s) (usually a single inductor) and the input/output capacitors. Often, a diode, the one which is an essential element of basic switcher topologies, cannot be integrated either. In any case, the complete power conversion for a switcher cannot be as integrated as is a linear regulator. The requirement of a magnetic element means that system designers should not be inclined to think of switching regulators as simply “drop in’ solutions. However, the switching regulators are very flexible, depending on their topology, they able to give not only lower voltage at the output than the input voltage. If it is necessary, the output voltage can be higher than the input voltage, or can has inverted polarity. The step-down, step-up and inverting regulators are provided in a huge range from the manufacturers. It is not a rare situation, that the modern switching mode regulator has the efficiency is over 90% in a wide operating range.

In this manual we only pay attention to step-down converters, which are used to down-convert a DC voltage to a lower DC voltage of the same polarity. It is named as Buck converter, and shown on the figure below.

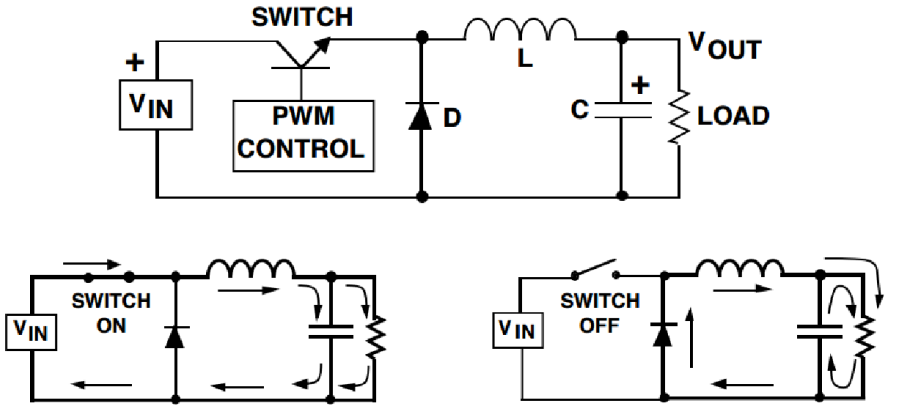


Figure 66. Buck converter, and the current paths at the closed and opened switch states

The lower diagrams show the current flow paths (shown as the heavy lines) when the switch is on and off. When the switch turns on, the input voltage is connected to the inductor. The difference between the input and output voltages is then forced across the inductor, causing current through the inductor to increase. During the on time, the inductor current flows into both the load and the output capacitor (the capacitor charges during this time. When the switch is turned off, the input voltage applied to the inductor is removed. However, since the current in an inductor can not change instantly, the voltage across the inductor will adjust to hold the current constant. The input end of the inductor is forced negative in voltage by the decreasing current, eventually reaching the point where the diode is turned on. The inductor current then flows through the load and back through the diode. The capacitor discharges into the load during the off time, contributing to the total current being supplied to the load (the total load current during the switch off time is the sum of the inductor and capacitor current). The shape of the current flowing in the inductor is similar to the figure below.

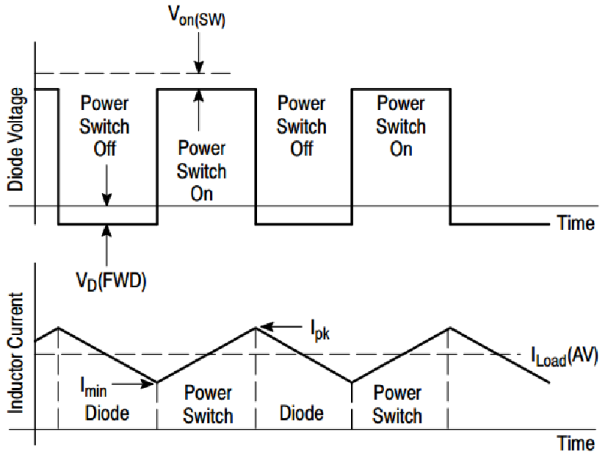


Figure 67. The diode voltage and the inductor current at a Buck regulator

As explained, the current through the inductor ramps up when the switch is on, and ramps down when the switch is off. The DC load current from the regulated output is the average

value of the inductor current. The peak-to-peak difference in the inductor current waveform is referred to as the inductor ripple current, and the inductor is typically selected large enough to keep this ripple current less than 20 % to 30 % of the rated DC current. Of course, for the voltage regulation a feedback is necessary, which controls the switching times according to the difference between the actual and desired output voltage.

A typical Buck regulator is the LM2575, which is a part of the SIMPLE SWITCHER product family. The LM2575 is rated for 1 A of continuous load current. The maximum input voltage for the part is 40 V (60 V for the 'HV' versions), with both adjustable and fixed output voltages available. The basic Buck regulator application circuit of this component is shown below. At fixed-output versions (as at the part used in this experiment), the R2 and R1 resistors are integrated into the regulator.

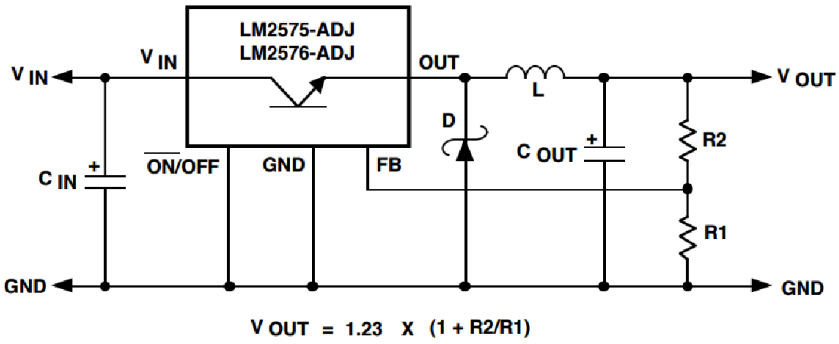


Figure 68. Typical application of the LM2575 Buck converter

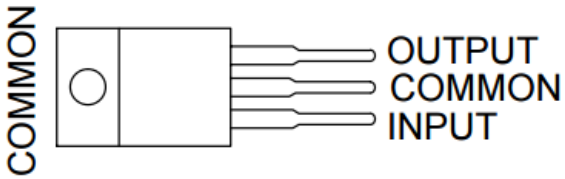
LM7805 electrical characteristics (from its data sheet)

electrical characteristics at specified virtual junction temperature, V_I = 10 V, I_O = 500 mA (unless otherwise noted)

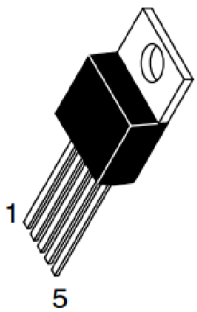
PARAMETER	TEST CONDITIONS	T_J†	μA7805C			UNIT
			MIN	TYP	MAX	
Output voltage	I_O = 5 mA to 1 A, $P_D \le 15$ W, V_I = 7 V to 20 V	25°C	4.8	5	5.2	V
		0°C to 125°C	4.75		5.25	
Input voltage regulation	V_I = 7 V to 25 V	25°C		3	100	mV
	V_I = 8 V to 12 V			1	50	
Ripple rejection	V_I = 8 V to 18 V, f = 120 Hz	0°C to 125°C	62	78		dB
Output voltage regulation	I_O = 5 mA to 1.5 A	25°C		15	100	mV
	I_O = 250 mA to 750 mA			5	50	
Output resistance	f = 1 kHz	0°C to 125°C		0.017		Ω
Temperature coefficient of output voltage	I_O = 5 mA	0°C to 125°C		–1.1		mV/°C
Output noise voltage	f = 10 Hz to 100 kHz	25°C		40		μV
Dropout voltage	I_O = 1 A	25°C		2		V
Bias current		25°C		4.2	8	mA
Bias current change	V_I = 7 V to 25 V	0°C to 125°C			1.3	mA
	I_O = 5 mA to 1 A				0.5	
Short-circuit output current		25°C		750		mA
Peak output current		25°C		2.2		A

† Pulse-testing techniques maintain the junction temperature as close to the ambient temperature as possible. Thermal effects must be taken into account separately. All characteristics are measured with a 0.33-μF capacitor across the input and a 0.1-μF capacitor across the output.

LM78XX pinout



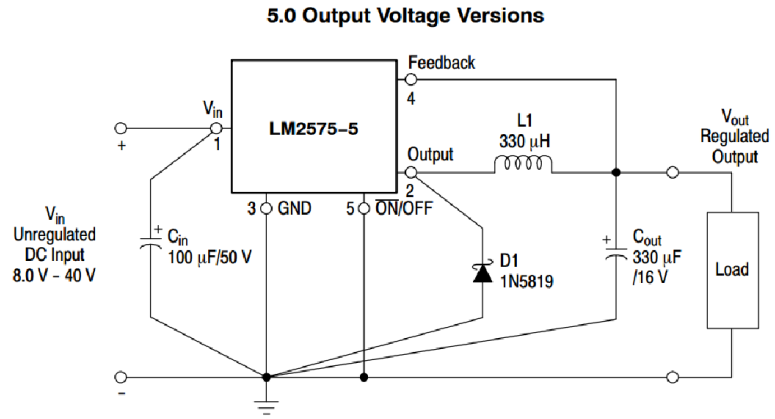
LM2575 fixed output voltage version - pinout



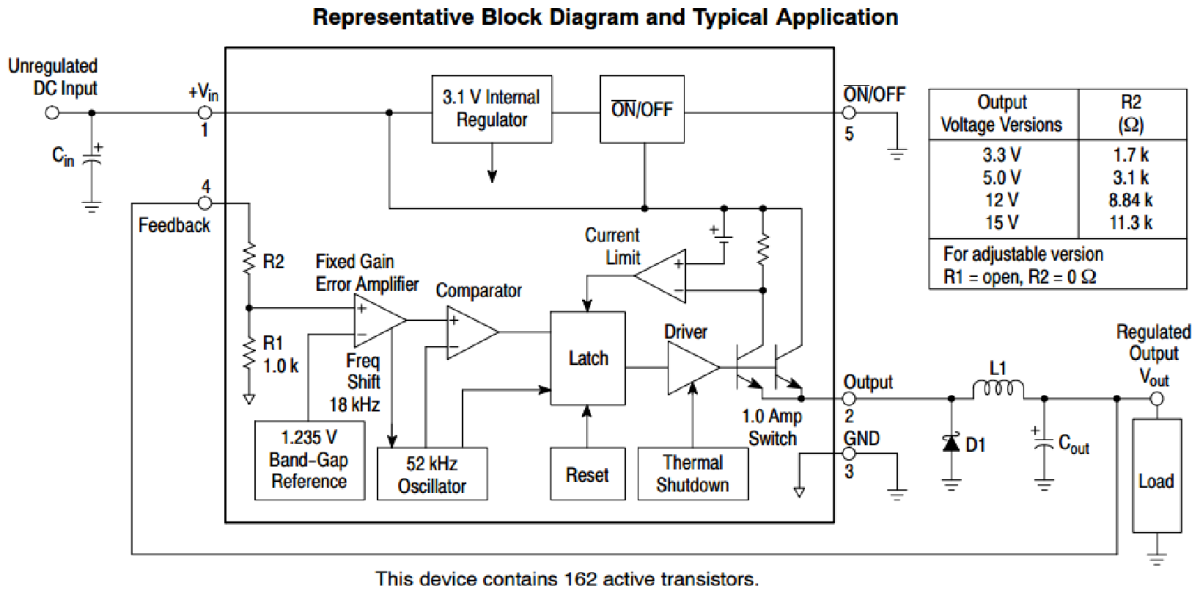
PIN FUNCTION DESCRIPTION

Pin	Symbol	Description (Refer to Figure 1)
1	V_{in}	This pin is the positive input supply for the LM2575 step–down switching regulator. In order to minimize voltage transients and to supply the switching currents needed by the regulator, a suitable input bypass capacitor must be present (C_{in} in Figure 1).
2	Output	This is the emitter of the internal switch. The saturation voltage V_{sat} of this output switch is typically 1.0 V. It should be kept in mind that the PCB area connected to this pin should be kept to a minimum in order to minimize coupling to sensitive circuitry.
3	GND	Circuit ground pin. See the information about the printed circuit board layout.
4	Feedback	This pin senses regulated output voltage to complete the feedback loop. The signal is divided by the internal resistor divider network R2, R1 and applied to the non–inverting input of the internal error amplifier. In the Adjustable version of the LM2575 switching regulator this pin is the direct input of the error amplifier and the resistor network R2, R1 is connected externally to allow programming of the output voltage.
5	$\overline{ON}$/OFF	It allows the switching regulator circuit to be shut down using logic level signals, thus dropping the total input supply current to approximately 80 μA. The input threshold voltage is typically 1.4 V. Applying a voltage above this value (up to $+V_{in}$) shuts the regulator off. If the voltage applied to this pin is lower than 1.4 V or if this pin is connected to ground, the regulator will be in the "on" condition.

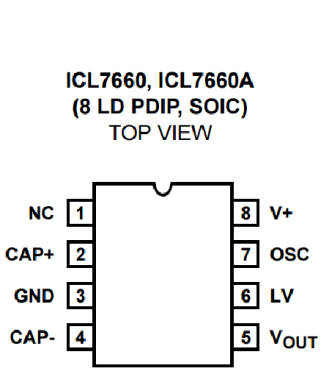
LM2575-5.0 typical application

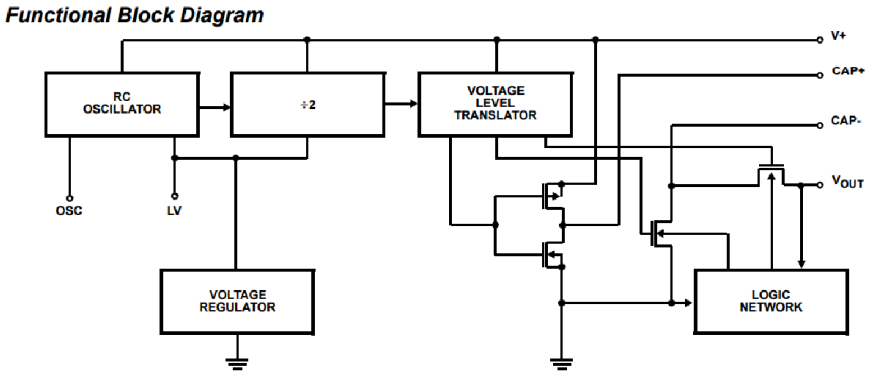


LM2575 fixed output voltage version - block diagram

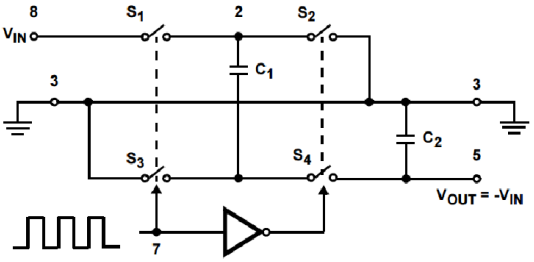


ICL 7660 pinout and functional block diagram

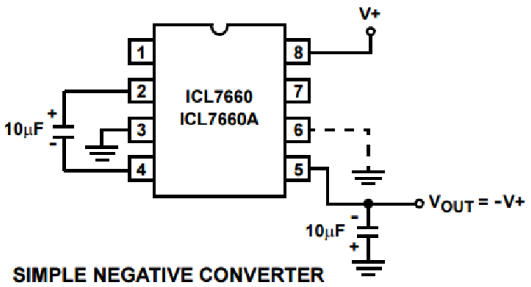




ICL7660 negative voltage converter application

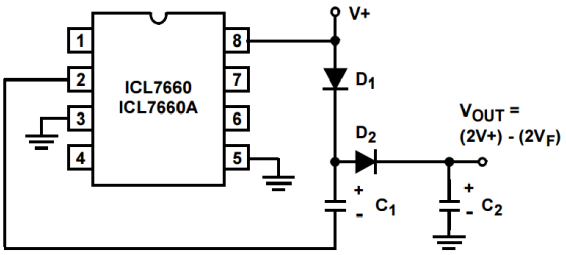


IDEALIZED NEGATIVE VOLTAGE CONVERTER



SIMPLE NEGATIVE CONVERTER

ICL7660 voltage doubler application



POSITIVE VOLT DOUBLER

Absolute Maximum Ratings

Supply Voltage
- ICL7660 . . . . . +10.5V
- ICL7660A . . . . . +13.0V

LV and OSC Input Voltage . . . . . -0.3V to (V+ +0.3V) for V+ < 5.5V
(Note 2) . . . . . (V+ -5.5V) to (V+ +0.3V) for V+ > 5.5V
Current into LV (Note 2) . . . . . 20µA for V+ > 3.5V
Output Short Duration ($V_{SUPPLY} \leq 5.5V$) . . . . . Continuous

ICL7660 electrical specifications

Electrical Specifications ICL7660 and ICL7660A, V+ = 5V, T_A = 25°C, C_{OSC} = 0, Test Circuit Figure 11 Unless Otherwise Specified

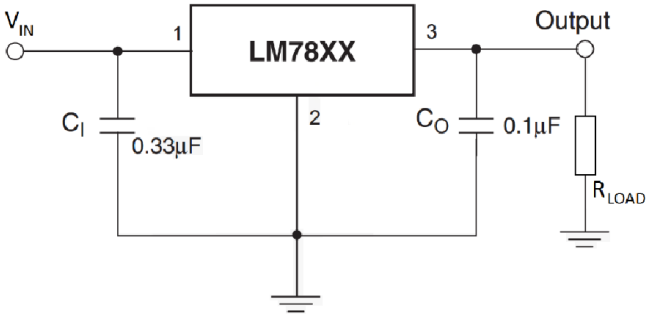
PARAMETER	SYMBOL	TEST CONDITIONS	ICL7660			ICL7660A			UNITS
			MIN	TYP	MAX	MIN	TYP	MAX	
Supply Current	I+	$R_L = \infty$	-	170	500	-	80	165	µA
Supply Voltage Range - Lo	V_L+	MIN ≤ T_A ≤ MAX, R_L = 10kΩ, LV to GND	1.5	-	3.5	1.5	-	3.5	V
Supply Voltage Range - Hi	V_H+	MIN ≤ T_A ≤ MAX, R_L = 10kΩ, LV to Open	3.0	-	10.0	3	-	12	V
Output Source Resistance	R_{OUT}	I_{OUT} = 20mA, T_A = 25°C	-	55	100	-	60	100	Ω
		I_{OUT} = 20mA, 0°C ≤ T_A ≤ 70°C	-	-	120	-	-	120	Ω
		I_{OUT} = 20mA, -55°C ≤ T_A ≤ 125°C	-	-	150	-	-	-	Ω
		I_{OUT} = 20mA, -40°C ≤ T_A ≤ 85°C	-	-	-	-	-	120	Ω
		V^+ = 2V, I_{OUT} = 3mA, LV to GND 0°C ≤ T_A ≤ 70°C	-	-	300	-	-	300	Ω
		V+ = 2V, I_{OUT} = 3mA, LV to GND, -55°C ≤ T_A ≤ 125°C	-	-	400	-	-	-	Ω
Oscillator Frequency	f_{OSC}		-	10	-	-	10	-	kHz
Power Efficiency	P_{EF}	R_L = 5kΩ	95	98	-	96	98	-	%
Voltage Conversion Efficiency	$V_{OUT\ EF}$	$R_L = \infty$	97	99.9	-	99	99.9	-	%
Oscillator Impedance	Z_{OSC}	V+ = 2V	-	1.0	-	-	1	-	MΩ
		V = 5V	-	100	-	-	-	-	kΩ

Part 1.

Build a linear regulator with 7805, and examine its operation!

Procedure

First, build the circuit shown below. R_{LOAD} = 100 Ω, V_{IN}=12 V. Measure the output voltage by DMM, and examine it by the oscilloscope! Repeat the previous action with R_{LOAD} = ∞. Then remove the capacitors, and repeat these steps again!



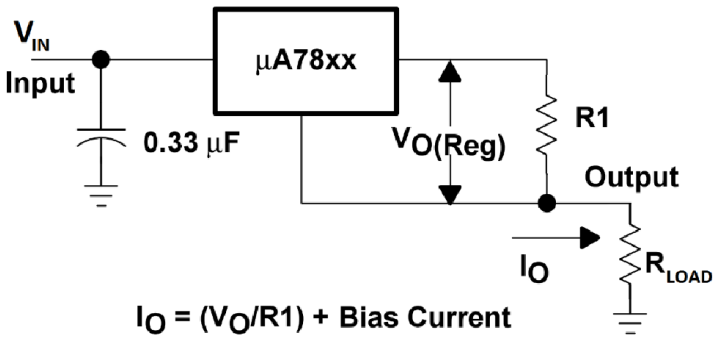
Measure the minimal dropout voltage at R_{LOAD} = 100 Ω! Reduce the input voltage to that level, where the output voltage decreases 2 % to the output voltage measured at V_{IN} = 12 V.

Part 2.

Build a current source with the 7805, and examine its operation!

Procedure

Build a current source with the 7805, as you can see on the figure below! R_1 =100 Ω, so output current is about 50 mA. (Bias current of 7805 is typically 4.2 mA, see data sheet). Measure the output current at R_{LOAD}=0 Ω and using another load!



How can you determine the maximum value of load resistance by calculation and by measurement?

Part 3.

Build a buck converter with LM2575, and examine its operation!

Procedure

Build a buck converter with LM2575T-5.0, according to LM2575-5.0 typical application at the data sheet section of this experiment! R_{LOAD} = 100 Ω. Measure the output voltage with DMM and oscilloscope! The oscilloscope must be in AC coupling mode, and with about 10 mV/DIV range!

Using the second channel of the oscilloscope measure the switching signal on the pin 2 of the circuit! Draw these signals in correct phase!

Measure the inductor current! For this step insert a low resistance resistor (for example 0.1 Ω) in series to the inductor, and measure the voltage on it by the oscilloscope! The GND of the oscilloscope must be on the right pin of the resistor (nearer to the output), and the channel input must be on the left side (nearer to the IC). Important: in that case the other channel of the oscilloscope **must not connect to the circuit** because of the danger of the short circuit to the ground! Draw the signal (the level of it will be very low: in case it cannot be measured, increase the output current by reducing the load resistance to 50 Ω).

Part 3.

Examine the CMOS switched capacitance converter type ICL7660!

Procedure

Build a simple negative voltage converter using the LMC7660. V_{IN} = 8 V, according to the ICL7660 negative voltage converter application at the data sheet section of this experiment. Examine its operation! Then build a voltage doubler. Diodes are type 1N4148. Examine its operation! In both cases examine the operation with $R_{LOAD} = \infty$, and R_{LOAD} = 1 kΩ.

Questions

1. Draw the block diagram of a typical AC to DC power supply! Explain its operation!
2. What kind of regulators do you know? What are the advantages of them?
3. Explain the operation of the linear regulators!
4. Draw the simplified schematic diagram of a linear voltage regulator! How does it work?
5. Explain the operation of the switched mode buck converter!
6. What is the dropout voltage?
7. How can you calculate the efficiency of a regulator?

Useful links:

http://cds.linear.com/docs/en/application-note/AN140fa.pdf

http://www.analog.com/library/analogDialogue/archives/43-09/EDCh%209%20power.pdf

http://www.ti.com/lit/an/snva558/snva558.pdf

http://www.ti.com/lit/an/snva559/snva559.pdf

Experiment 7.

Analog to digital converters

Objectives

- To study the operation of basic ADC topologies.
- Gathering experience how to connect digital and analog world.
- To study, how to define and measure static ADC parameters.

Components

None

Instruments

Items	Comment
Power supply	dual
Digital multimeter	
Function generator	
Digital oscilloscope	dual channel
ADC demo panel	

Background

The analog-to-digital and digital-to-analog conversion

As we described in Experiment 5. (D/A converters), in our days it is essential to convert analog signals to digital, and vice versa at high performance. The most of complex systems are digital, but the signals of the "real world' we have to process are analog. Suppose, that you want to take a digital photo of your holiday. You press the shutter button, the analog signal of the CCD (which converts the analog light intensity – originated from the object – to electrical charge, but practically to say to analog voltage) is generated, but this analog signal is not really simply to post-process and store in the memory. In the camera there is an ADC, which converts all pixel voltages of the CCD array to digital number. Note, that all pixel contains 3 sub-pixels (red, green and blue). If the picture has 1920 x 1080 pixels (really it is not a high resolution), it means, that 1920 x 1080 x 3 = 6,220,800 different voltages must be digitized by an ADC. The digitized information must be very precise, because the photo must have high quality (colors, intensities…). And, last but not least the conversion must be executed very quickly! And it is only one example…

Analog-to-digital converters (ADCs) translate analog quantities – which are typical of most phenomena in the 'real world' – to digital values, used in information processing, computing, data transmission, and control systems. Digital-to-analog converters (DACs) are used in transforming transmitted or stored data, or the results of digital processing, back to 'real-world' variables for control, information display, or further analog processing. The relationships between inputs and outputs of DACs and ADCs are shown in the figure below.

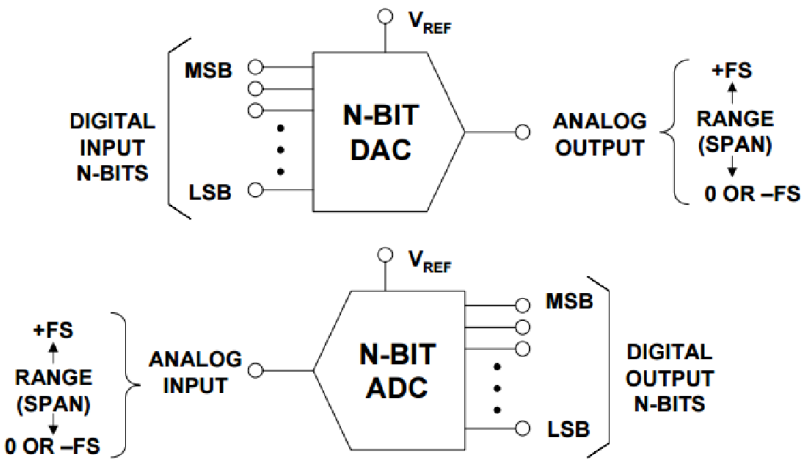


Figure 69. DAC and ADC input and output definitions

The transfer function of an ideal 3-bit ADC is shown in figure below. There is a range of analog input voltage over which the ADC will produce the same output code; this range is the quantization uncertainty and is equal to 1 LSB. Note that the width of the transition regions between adjacent codes is zero for an ideal ADC (because it has infinite resolution). In practice, however, there is always transition noise associated with these levels, and therefore the width

is non-zero. It is customary to define the analog input corresponding to a given code by the code center which lies halfway between two adjacent transition regions (illustrated by the black dots in the diagram). This requires that the first transition region occur at ½ LSB. The full-scale analog input voltage is defined by 7/8 FS, (FS – 1 LSB).

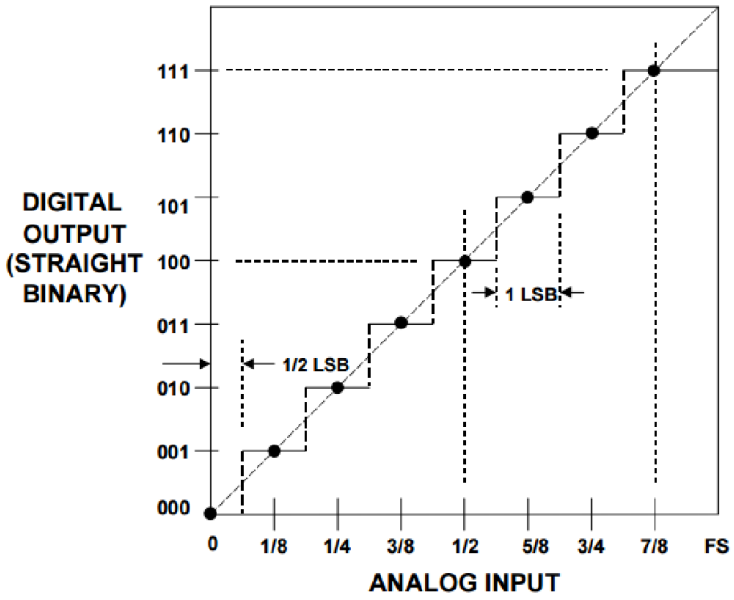


Figure 70. Transfer function for ideal unipolar 3-bit ADC

In many systems, it is desirable to represent both positive and negative analog quantities with binary codes. Either offset binary, twos complement, ones complement, or sign-magnitude codes will accomplish this, but offset binary and twos complement are by far the most popular codes. For offset binary, the zero signal value is assigned the code 1000…0, where "…' means additional zeroes, according to the number of the bits of the ADC. The sequence of codes is identical to that of straight binary. The only difference between a straight and offset binary system is the half-scale offset associated with analog signal.

The offset binary output code for a bipolar 3-bit ADC as a function of its analog input is shown in the next figure. Note that zero analog input defines the center of the mid-scale code 100. The most negative input voltage is generally defined by the 001 code (–FS + 1 LSB), and the most positive by 111 (+FS – 1 LSB). The 000 output code is available for use if desired, but makes the output non-symmetrical about zero and complicates the mathematics.

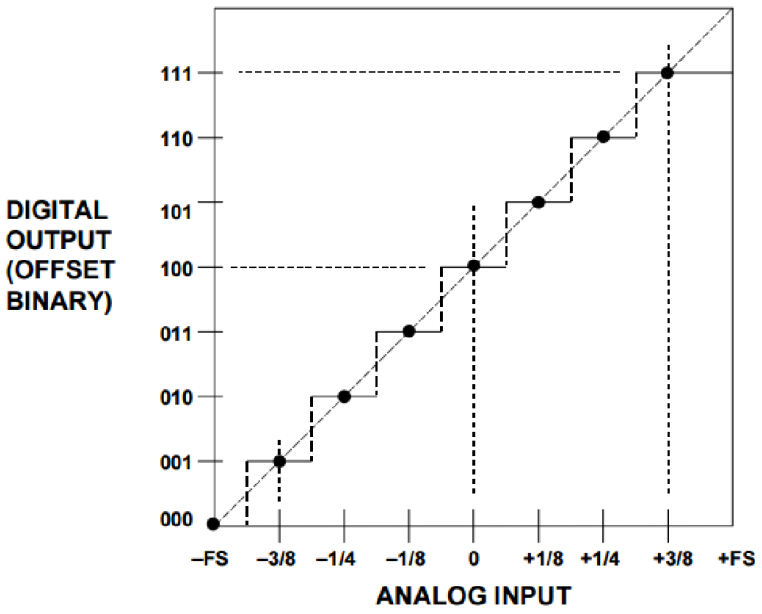


Figure 71. Transfer function for ideal bipolar 3-bit ADC

The unipolar converters, has only a single polarity analog output port. These are the simplest type, but bipolar converters are generally more useful in real-world applications.

There are two types of bipolar converters: the simpler is merely a unipolar converter with an accurate 1 MSB of negative offset (and many converters are arranged so that this offset may be switched in and out so that they can be used as either unipolar or bipolar converters at will), but the other, known as a sign-magnitude converter is more complex, and has N bits of magnitude information and an additional bit which corresponds to the sign of the analog signal. Sign-magnitude DACs are quite rare, and sign-magnitude ADCs are found mostly in digital voltmeters (DVMs). The unipolar, offset binary, and sign-magnitude representations are shown in the next figure.

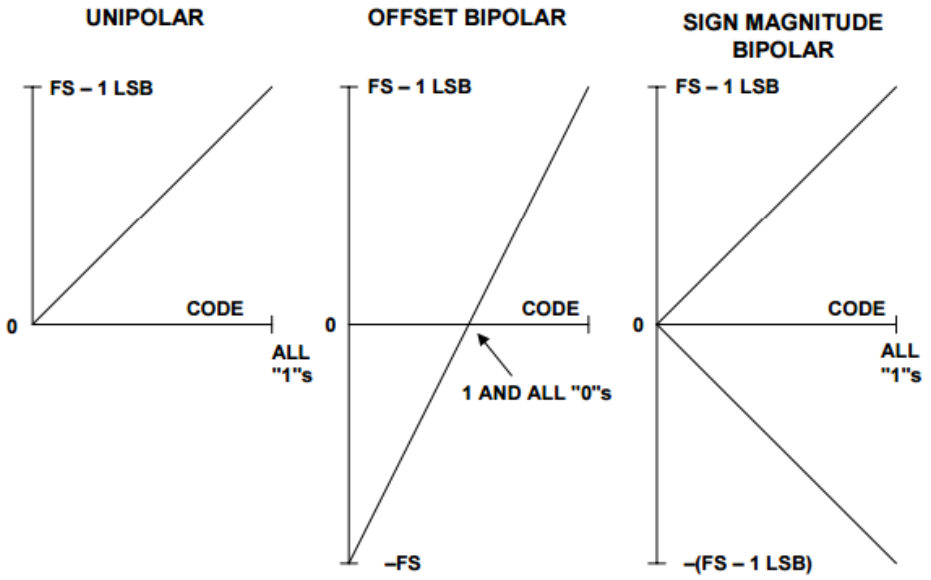


Figure 72. Unipolar and bipolar converters

Static errors of D/A converters

The most important thing to remember about both DACs and ADCs is, that either the input or output is digital, and therefore the signal is quantized. That is, an N-bit word represents one of 2^N possible states, and therefore an N-bit DAC (with a fixed reference) can have only 2^N possible analog outputs, and an N-bit ADC can have only 2^N possible digital outputs. The analog signals will be generally voltages or currents. The resolution of data converters may be expressed in several different ways: the weight of the Least Significant Bit (LSB), parts per million of full-scale (ppm FS), millivolts (mV), etc. Different devices (even from the same manufacturer) will be specified differently, so converter users must learn to translate between the different types of specifications to compare devices successfully. Note, at the 24-bit ADC (FS = 10 V), the LSB is about 600 nV, while the Johnson noise in a 10 kHz BW of a 2.2 kΩ resistor @ 25 °C has a same value! It is useful to remember: 10-bits and 10 V FS yields an LSB of 10mV, 1000 ppm, or 0.1%. All other values may be calculated by powers of 2. (For example: 10 V FS, 12-bits resolution yields an LSB of about 10 mV/$2^{(12-10)}$ = 2.5 mV.)

RESOLUTION N	2^N	VOLTAGE (10V FS)	ppm FS	% FS	dB FS
2-bit	4	2.5 V	250,000	25	– 12
4-bit	16	625 mV	62,500	6.25	– 24
6-bit	64	156 mV	15,625	1.56	– 36
8-bit	256	39.1 mV	3,906	0.39	– 48
10-bit	1,024	9.77 mV (10 mV)	977	0.098	– 60
12-bit	4,096	2.44 mV	244	0.024	– 72
14-bit	16,384	610 µV	61	0.0061	– 84
16-bit	65,536	153 µV	15	0.0015	– 96
18-bit	262,144	38 µV	4	0.0004	– 108
20-bit	1,048,576	9.54 µV (10 µV)	1	0.0001	– 120
22-bit	4,194,304	2.38 µV	0.24	0.000024	– 132
24-bit	16,777,216	596 nV	0.06	0.000006	– 144

Table 7. The size of LSB at different resolutions

Before we can consider the various architectures used in data converters, it is necessary to consider the performance to be expected, and the specifications which are important. In the following we will consider the main definition of static errors and specifications used for data converters. The first applications of data converters were in field of measurement and control, where the exact timing of the conversion was usually unimportant, and the data rate was slow. In such applications, the DC specifications of converters are important, but timing and AC specifications are not. Today many, if not most, converters are used in sampling and signal reconstruction systems where AC specifications are critical (and DC ones may not be). The figure below shows the ideal transfer characteristics for a 3-bit unipolar ADC. The input to an ADC is analog and is not quantized, but its output is quantized. The transfer characteristic therefore consists of eight horizontal steps. When considering the offset, gain and linearity of

an ADC we consider the line joining the midpoints of these steps – often referred to as the code centers. Digital full-scale (all '1's) corresponds to 1 LSB below the analog full-scale (FS).

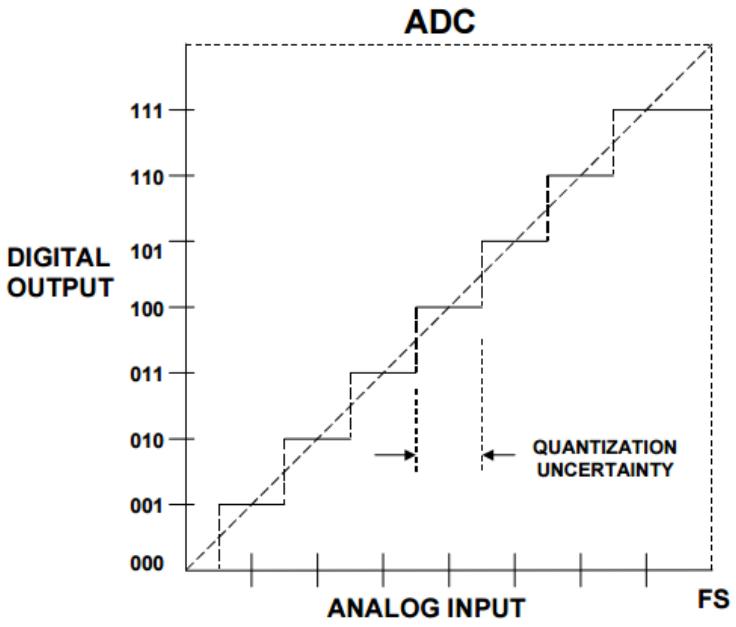


Figure 73. Quantization uncertainity

The (ideal) ADC transitions take place at ½ LSB above zero, and thereafter every LSB, until 1½ LSB below analog full-scale. Since the analog input to an ADC can take any value, but the digital output is quantized, there may be a difference of up to ½ LSB between the actual analog input and the exact value of the digital output. This is known as the quantization error or quantization uncertainty as shown in figure above. Note, that in AC (sampling) applications this quantization error gives rise to quantization noise, which is not discussed in this manual.

The four DC errors in a data converter are offset error, gain error, and two types of linearity error (differential and integral). Offset and gain errors are analogous to offset and gain errors in amplifiers as shown below for a bipolar input range. (Though offset error and zero error, which are identical in amplifiers and unipolar data converters, are not identical in bipolar converters and should be carefully distinguished.) The transfer characteristics of both DACs and ADCs may be expressed as a straight line given by D = K + GA, where D is the digital code, A is the analog signal, and K and G are constants. In a unipolar converter, the ideal value of K is zero; in an offset bipolar converter it is –1 MSB. The offset error is the amount by which the actual value of K differs from its ideal value. The gain error is the amount by which G differs from its ideal value, and is generally expressed as the percentage difference between the two, although it may be defined as the gain error contribution (in mV or LSB) to the total error at full-scale. These errors can usually be trimmed by the data converter user. Note, however, that amplifier offset is trimmed at zero input, and then the gain is trimmed near to full-scale. The trim algorithm for a bipolar data converter is not so straightforward.

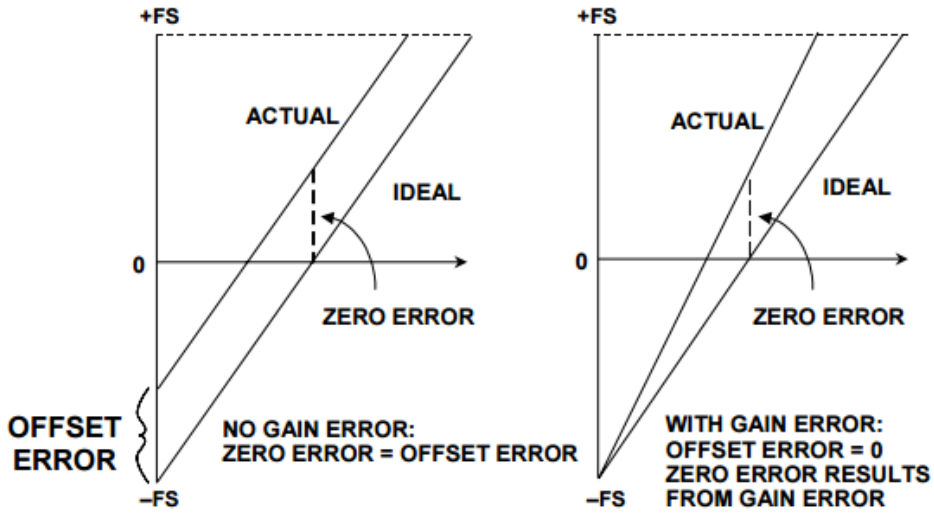


Figure 74. Bipolar ADC offset and gain error

The integral linearity error of a converter is also analogous to the linearity error of an amplifier, and is defined as the maximum deviation of the actual transfer characteristic of the converter from a straight line, and is generally expressed as a percentage of full-scale (but may be given in LSBs). For an ADC, the most popular convention is to draw the straight line through the mid-points of the codes, or the code centers. There are two common ways of choosing the straight line: end point and best straight line as shown in figure below.

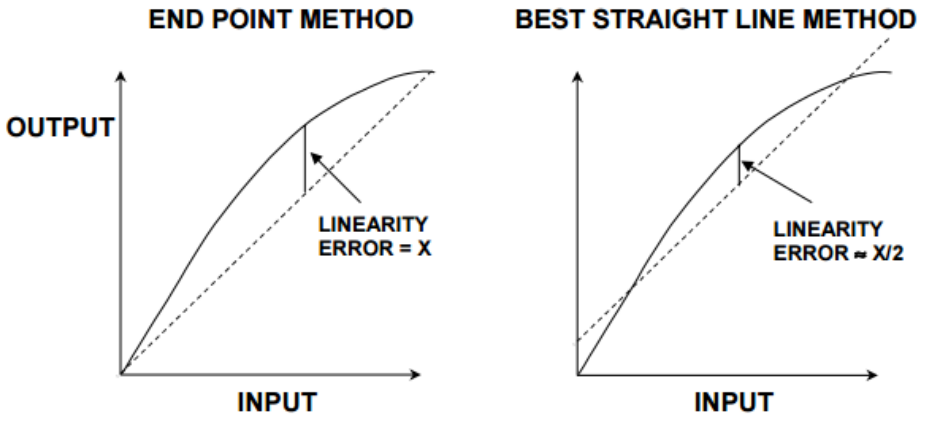


Figure 75. Methods of defining integral linearity errors

The other type of converter nonlinearity is differential nonlinearity (DNL). This relates to the linearity of the code transitions of the converter. In the ideal case, a change of 1 LSB in digital code corresponds to a change of exactly 1 LSB of analog signal. In a DAC, a change of 1 LSB in digital code produces exactly 1 LSB change of analog output, while in an ADC there should be exactly 1 LSB change of analog input to move from one digital transition to the next. Differential linearity error is defined as the maximum amount of deviation of any quantum (or LSB change) in the entire transfer function from its ideal size of 1 LSB.

In case the change in analog signal corresponding to 1 LSB digital change is more or less than 1 LSB, there is said to be a DNL (Differential Non-Linearity) error. The DNL error of a converter

is normally defined as the maximum value of DNL to be found at any transition across the range of the converter. Figure below shows the non-ideal transfer functions for a DAC and an ADC and shows the effects of the DNL error.

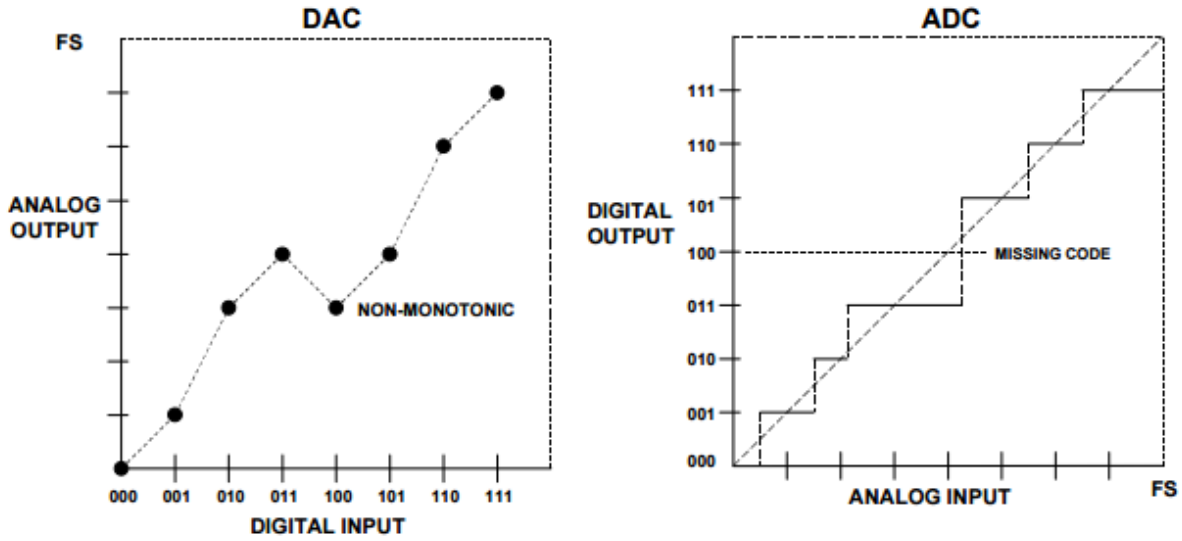


Figure 76. Transfer Functions for Non-Ideal 3-Bit DAC and ADC

On the next figure, the DNL of an ADC is examined more closely on an expanded scale. ADCs can be non-monotonic, but a more common result of excess DNL in ADCs is missing codes. Missing codes in an ADC are as objectionable as non-monotonicity in a DAC.

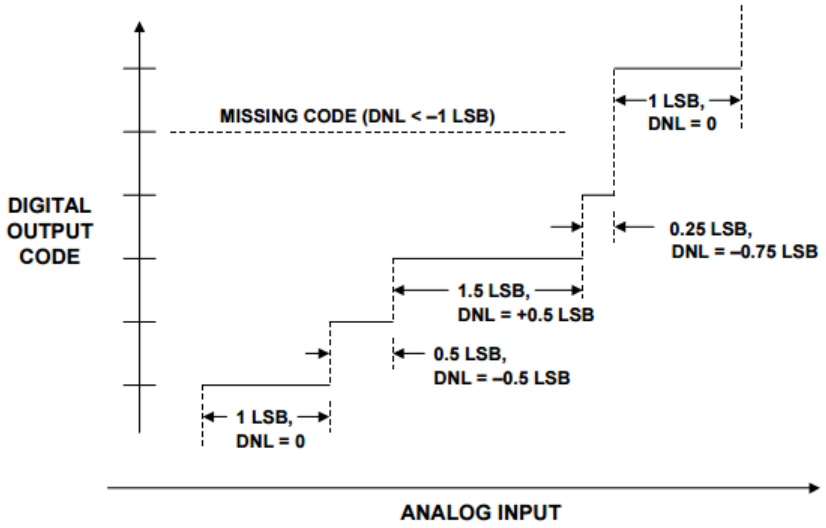


Figure 77. Missing code caused by DNL, ADC remains monotonic

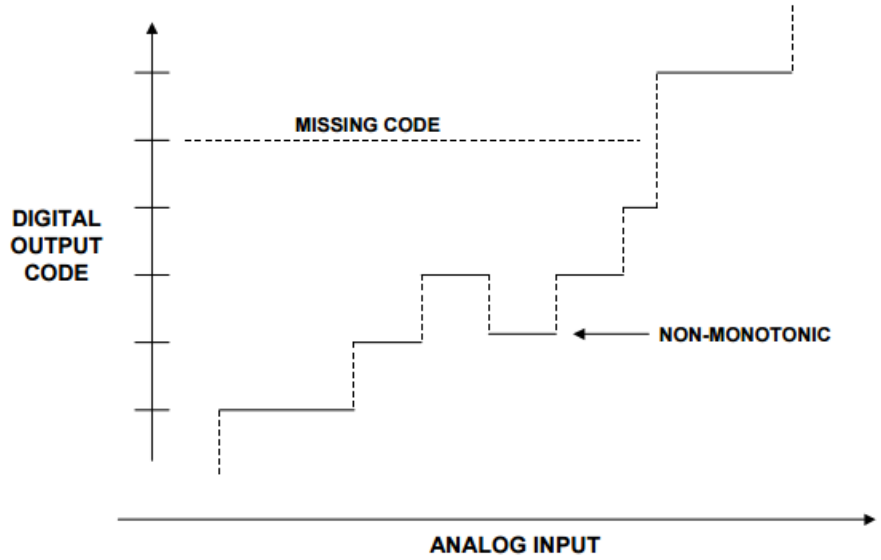


Figure 78. Non-monotonic ADC with missing code

Sampling

Some ADC is able to digitize changing (AC) signals, but most of them require constant input voltage during analog to digital conversion. Certain types of ADCs produce incorrect output code, if the input voltage does not remain constant during conversion (for example SAR ADC). Most ADCs today have a built-in sample-and-hold function, thereby allowing to process AC signals. This type of ADC is referred to as sampling ADC.

To process AC signals, a sample-and-hold function is added as shown in figure below. The ideal SHA (Sample-and-Hold Amplifier) is a switch driving a hold capacitor followed by a high input impedance buffer. The input impedance of the buffer must be high enough so that the capacitor is discharged by less than 1 LSB during the hold time. The SHA samples the signal in the sample mode, and holds the signal constant during the hold mode. Timing is adjusted so that the encoder performs the conversion during the hold time. A sampling ADC can therefore process fast signals – the upper frequency limitation is determined by the SHA aperture jitter, bandwidth, distortion, etc., not the encoder. In the example shown, a good sample-and-hold could acquire the signal in 2 µs, allowing a sampling frequency of 100 kSPs (kilosamples per second, kS/s), and the capability of processing input frequencies up to 50 kHz. (The values will be explained later.)

It is important to understand a subtle difference between a true sample-and-hold amplifier (SHA) and a track-and-hold amplifier (T/H, or THA). Strictly speaking, the output of a sample-and-hold is not defined during the sample mode, however the output of a track-and-hold tracks the signal during the sample or track mode. In practice, the function is generally implemented as a track-and-hold, and the terms track-and-hold and sample-and-hold are often used interchangeably. The waveforms shown in the next figure are those associated with a track-and-hold.

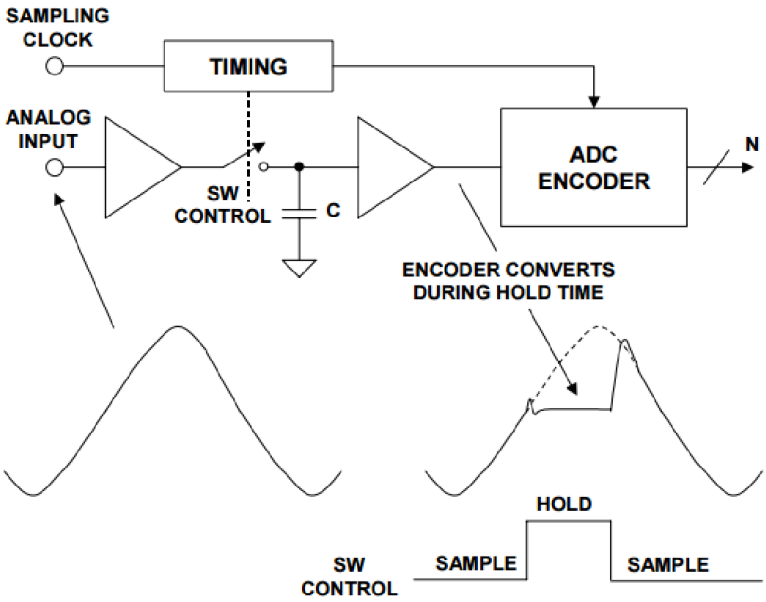


Figure 79. Sample-and-Hold Function Required for Digitizing AC Signals

At sampling, there is a very important rule: the Nyquist-rule. A continuous analog signal is sampled at discrete intervals, $t_s = 1/f_s$ (f_s is the sampling frequency), which must be carefully chosen to ensure an accurate representation of the original analog signal. Simply stated, the Nyquist-rule requires that the sampling frequency must be at least twice the highest frequency contained in the signal, otherwise some information about the signal will be lost. If the sampling frequency is less than twice the maximum analog signal frequency, a phenomena known as aliasing will occur. The Nyquist-rule: a signal with a maximum frequency f_a must be sampled at a rate $f_s > 2 \cdot f_a$ or information about the signal will be lost because of aliasing. Aliasing occurs whenever $f_s < 2 \cdot f_a$. The aliasing can be studied in time and frequency domain, but we only will take a short overview in time domain, only at sinusoidal input voltage, which does not contain other frequency components than its base frequency, f_{in}.

If the sampling frequency, f_s is higher, than the double of the frequency f_{in}, then after the sampling we will have more than 2 samples per period of input voltage. Using appropriate reconstruction technique (called interpolation), the original input signal can be restored. The reconstruction technique is simple (e.g. using a SHA unit), the higher number of samples required per period of input signal to restore it at an acceptable quality. If at reconstruction we only use only simple lines to join the signal's points, at least 10…20 samples required to produce an acceptable output signal. Using more complex interpolation technique (e.g. using a high-order low-pass filter), may be enough for example 2.5 samples per period…

On the figure below we can see, if there are many points sampled per period of input signal, the shape of the signal easily recoverable. If there are less points, the signal shape will be distorted, but if the Nyquist-rule is kept, the frequency of the output signal is same as the sampled signal.

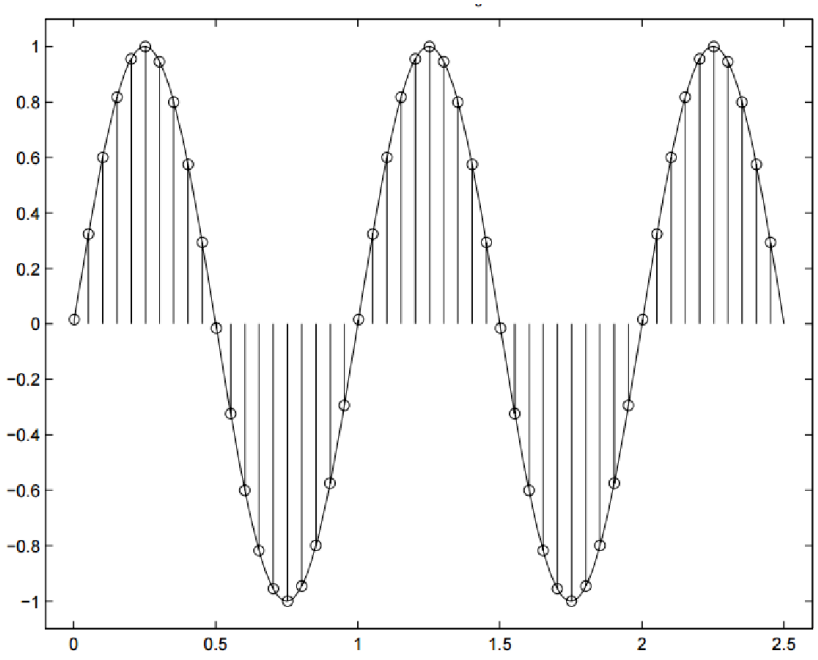


Figure 80. Sampling a sinusoid signal at a high rate (20 samples per period)

If the sampling frequency is insufficient (does not meet the Nyquist-rule), the restored signal's frequency will not be equal than the original signal frequency. Its frequency will be lower, than expected. This phenomenon is called aliasing, as shows the figure below. This problem occurs in that case too, when the input signal contains overtones (harmonics), and although the Nyquist-rule is OK at the base frequency, but does not meet on higher overtones. An appropriate input low-pass filter (in the front of the sampling unit) can limit the upper frequency of the signal, and helps to avoid aliasing.

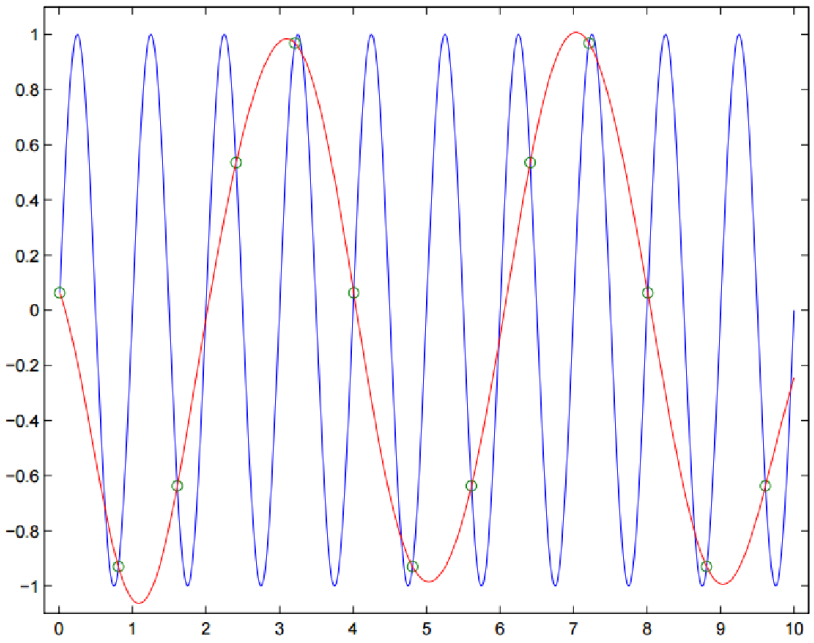


Figure 81. Sampling a sinusoid signal at a too slow rate (about 1.25 sample per period)

Sampled data system

An outline of a sampled data system containing ADC and DAC can be seen on the figure below.

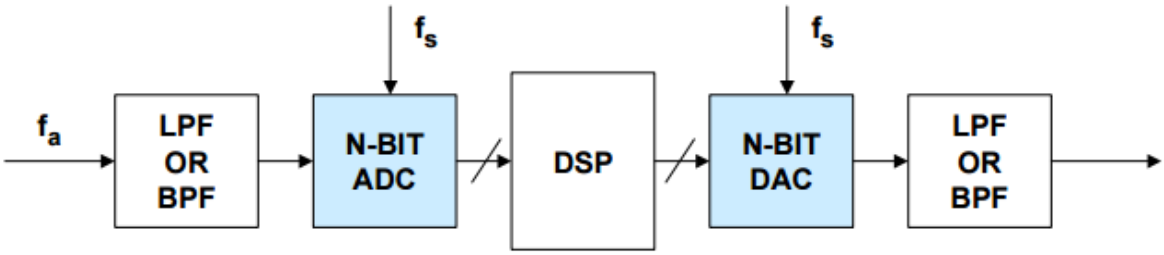


Figure 82. Sampled Data System

The input frequency range is limited by the input (anti-aliasing) low-pass filter, to less, than $f_s/2$. The ADC digitizes the band-limited input signal, and passes the digital output values to a special computing unit (DSP – Digital Signal Processor), which performs the necessary operations on the digitized data flow. Finished processing the signal, DSP passes the digital data to the DAC, which converts numbers of the data flow to analog voltage (or current). Because the output voltage has steps due to the DAC's sampling frequency, this steps must be smoothed by an output low-pass filter. In the system may be many additional elements, but the elements on the figure are utilized in most of the systems.

A few ADC topologies

The tracking ADC

The tracking ADC architecture (shown on the figure below) continually compares the input signal with a reconstructed representation of the input signal. The up/down counter is controlled by the comparator output. If the analog input exceeds the DAC output, the counter counts up until they are equal. If the DAC output exceeds the analog input, the counter counts down until they are equal.

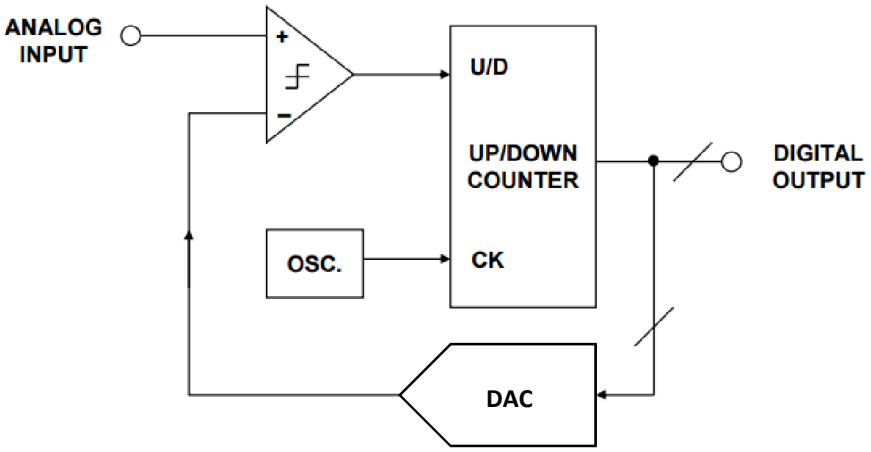


Figure 83. Tracking ADC

It is evident that if the analog input changes slowly, the counter can follow it, and the digital output will remain close to its correct value (see figure below). If the analog input suddenly undergoes a large step change, it will be many hundreds or thousands of clock cycles before the output is again valid. The tracking ADC therefore responds correctly to slowly changing signals, but results large errors at a quickly changing signal.

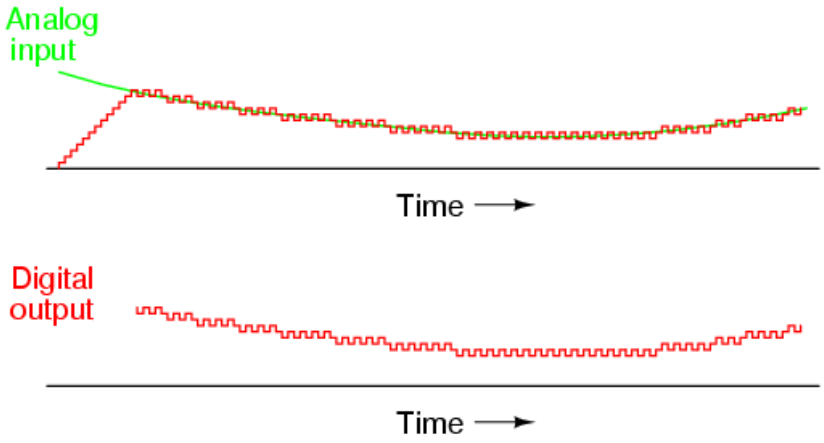


Figure 84. Tracking ADC time diagram

The simple analysis above ignores the behavior of the ADC in case the analog input and DAC output are nearly equal. This will depend on the exact nature of the comparator and counter. If the comparator is a simple one, the DAC output will cycle by 1 LSB round the analog input level, and the digital output will, of course, do the same—there will be 1 LSB of flicker. Note that the output in such a case steps every clock cycle, irrespective of the exact value of the analog input. A more satisfactory, but more complex arrangement would be to use a window comparator with a 1-2 LSB wide window. When the DAC output is too high or too low, the system behaves as in the previous description, but if the DAC output is within the window, the counter stops cycling. Tracking ADCs are not very common. Their slow step response makes them unsuitable for many applications, but they do have one asset: their output is continuously available. Most ADCs perform conversions: i.e., on receipt of a 'start convert' command (which may be internally generated), they perform a conversion and, after a delay, a result becomes available. Providing that the analog input changes slowly, the correct output of a tracking ADC is always available. Another valuable characteristic of tracking ADCs is that a fast transient on the analog input causes the output to change only one count. This is very useful in noisy environments. Notice the similarity between a tracking ADC and a successive approximation ADC. Replacing the up/down counter with SAR logic yields the architecture for a successive approximation ADC.

The successive approximation (SAR) ADC

The successive approximation ADC has been the mainstay of data acquisition systems for many years. The basic successive approximation ADC is shown in the figure below. It performs conversions on command. In order to process AC signals, SAR ADCs must have an input sample-and-hold amplifier (SHA) to keep the signal constant during the conversion cycle.

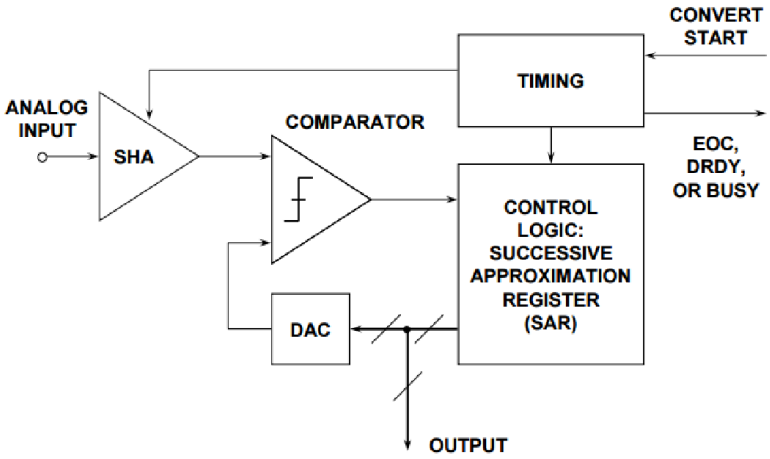


Figure 85. Successive approximation ADC

On the assertion of the CONVERT START command, the sample-and-hold (SHA) is placed in the hold mode, and the internal DAC is set to midscale (half of FS). The comparator determines whether the SHA output is above or below the DAC output, and the result (bit 1, the most significant bit of the conversion) is stored in the successive approximation register (SAR). The DAC is then set either to ¼ scale or ¾ scale (depending on the value of bit 1), and the comparator makes the decision for bit 2 of the conversion. The result is stored in the register, and the process continues. The example of the timing diagram of a SAR ADC is shown below.

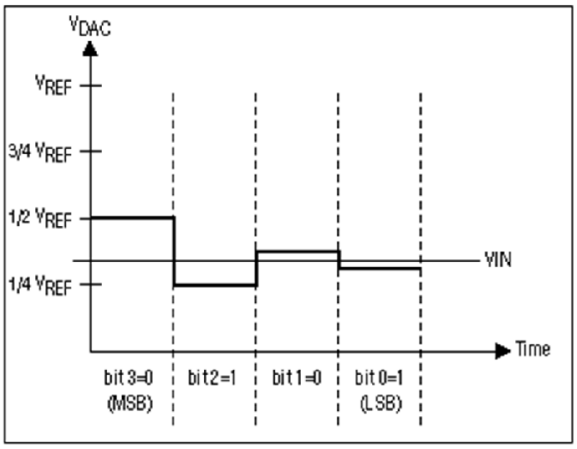


Figure 86. SAR ADC timing diagram

Timing of a typical SAR ADC conversion is the following. First, there is a signal, which indicates, and controls the sampling ($\overline{\text{CONVST}}$). Its rising edge starts the conversion. The end of conversion is generally indicated by an end-of-convert (EOC), data-ready (DRDY), or a busy signal (actually, not-BUSY indicates end of conversion). The polarities and name of this signals may be different for different SAR ADCs, but the fundamental concept is the same. At the beginning of the conversion interval, the signal goes high (or low) and remains in that state until the conversion is completed, at which time it goes low (or high). The trailing edge is generally an indication of valid output data, but data sheet should be carefully studied—in some ADCs additional delay is required before the output data is valid. Timing of the conversion is shown below.

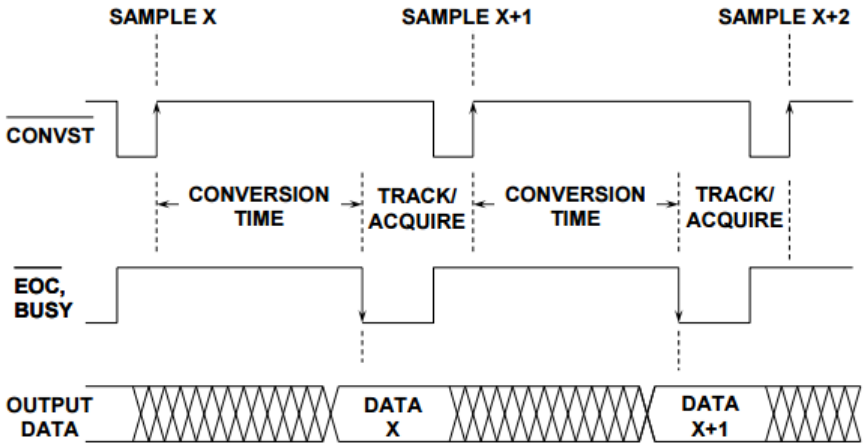
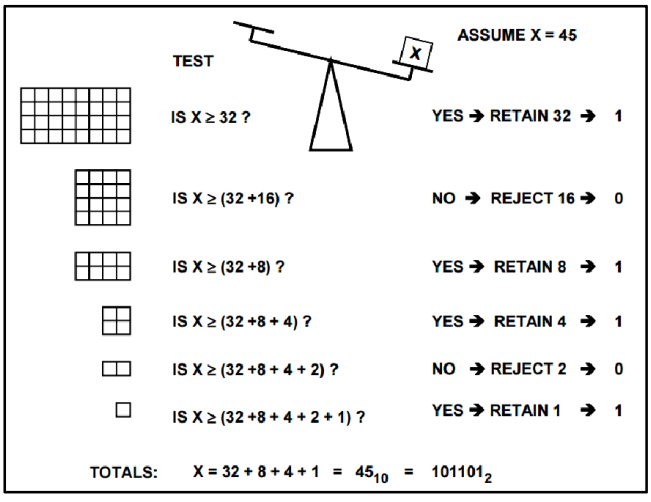


Figure 87. SAR ADC conversion timing

An N-bit conversion takes N steps. It would seem on superficial examination that a 16-bit converter would have twice the conversion time of an 8-bit one, but this is not the case. In an 8-bit converter, the DAC must settle to 8-bit accuracy before the bit decision is made, whereas in a 16-bit converter, it must settle to 16-bit accuracy, which takes a lot longer (about double the time). Because of their architecture, SAR ADCs generally allow single-shot conversion at any repetition rate from DC to the converter's maximum conversion rate. Notice that the overall accuracy and linearity of the SAR ADC is determined primarily by the internal DAC. Until recently, most precision SAR ADCs used laser-trimmed thin-film DACs to achieve the desired accuracy and linearity. The thin-film resistor trimming process adds cost, and the thin-film resistor values may be affected when subjected to the mechanical stresses of packaging. For these reasons, switched capacitor (or charge-redistribution) DACs have become popular in newer SAR ADCs.

The figures below try to illustrate, how the conversion takes place.



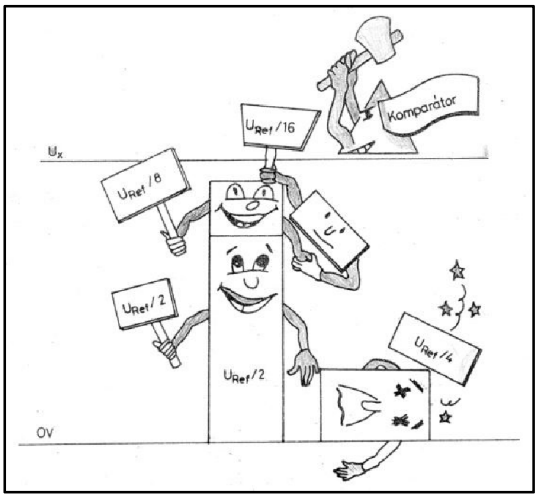


Figure 88. SAR ADC operation

One of the possible SAR structure is shown on the figure below. (Try to find, how it works!)

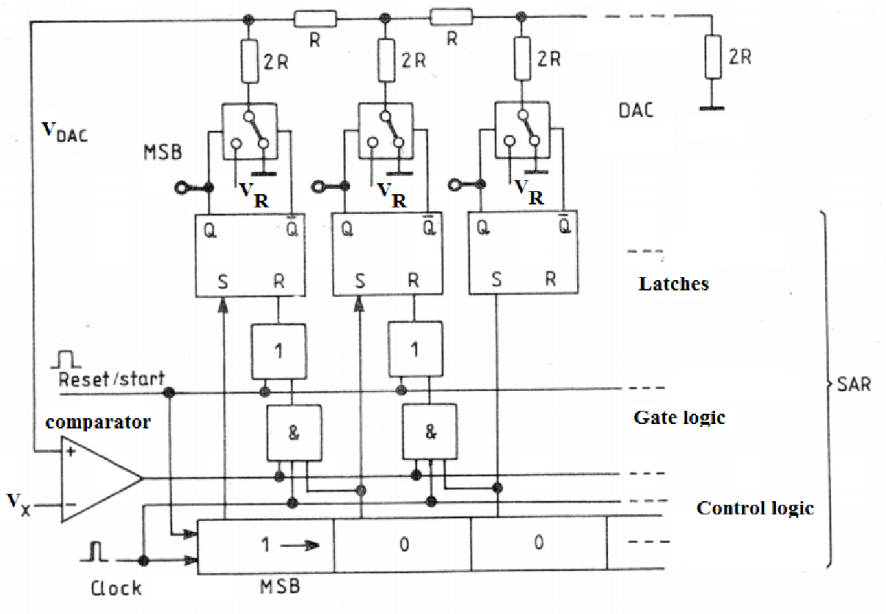


Figure 89. SAR ADC structure

The voltage-to frequency (VFC) ADC

The voltage-to-frequency converter (VFC) is an oscillator whose frequency is proportional to a control voltage. The VFC/counter ADC is monotonic and free of missing codes, integrates noise, and can consume very little power. It is also very useful for telemetry applications, since the VFC, which is small, cheap and low-powered can be mounted on the experimental subject (patient, wild animal, artillery shell, etc.) and communicate with the counter by a telemetry link as shown in the figure below.

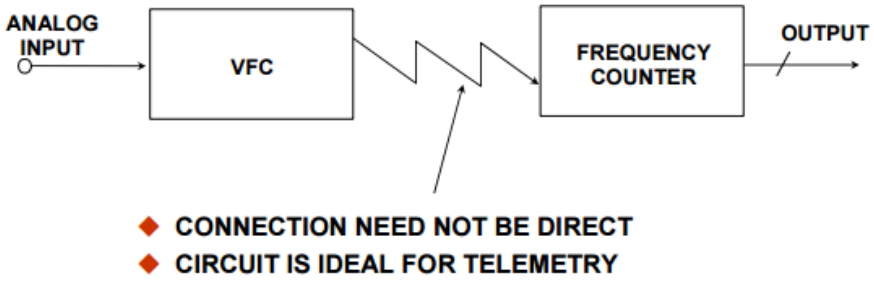


Figure 90. VFC ADC application

One possible implementation of VFC ADC is shown below. The input voltage is inverted by a voltage inverter, so at the analog switch there is the input voltage on the one side, and the inverted input voltage on the other side. While the integrator – following the switch – integrates the $+V_X$, the voltage at the output of the integrator will go lower. When this voltage exceeds the V_R value at the inverting input of the comparator at the bottom, the RS FF will be set, and the output Q will be high. This level controls the switch to conduct the $-V_X$ to the integrator, and the integrator output voltage will rise. When the voltage level at the non-inverting input of the upper comparator will be higher than GND, the RS FF will be reset, and

the output will be low. This switches the $+V_X$ to the integrator, and the process described above will continue. If higher the V_X, the integrator output will change faster, so the time between set and reset will be shorter. In other words: higher the input voltage, higher the output frequency.

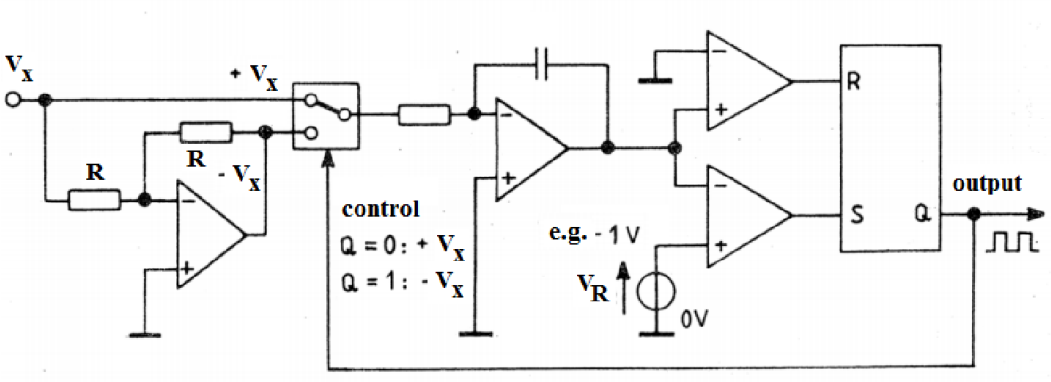


Figure 91. VFC ADC implementation

ADC applications

The applications of the analog-to-digital converters are very wide, they can be found in the cheap electronic devices, and in the most expensive ones. Performance of them varies in a wide range, if we only focus to the resolution, the ADCs are produced from typically 8 to 24 bits, but if you survey the offer from the manufacturers, you can find 6-bit and 32-bit ADC too.

The figure below shows a part of block diagram of an ECG (electrocardiograph) equipment: in the orange block there are two ADCs in the signal chain. The orange square represents an integrated circuit, called ECG Analog Front End, which contains several analog and digital functions including the two integrated ADCs. Functions of this circuit (analog switch states in the MUX, amplification of the buffers, sampling rates of the ADCs) can be set from the external microcontroller. The output data from the ADCs is also processed by the external microcontroller or DSP (digital signal processor). Typical resolution of these ADCs is 16 or 24 bit, the speed of conversion is about 10 kSps (kilosamples per second, i.e. 10.000 samples per second).

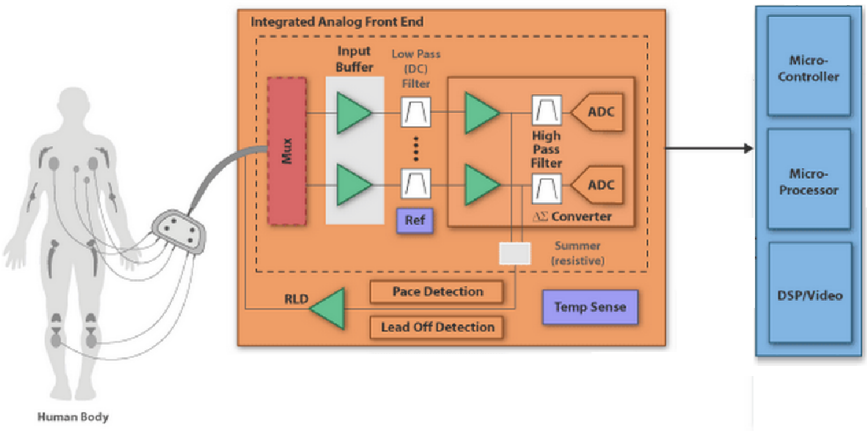
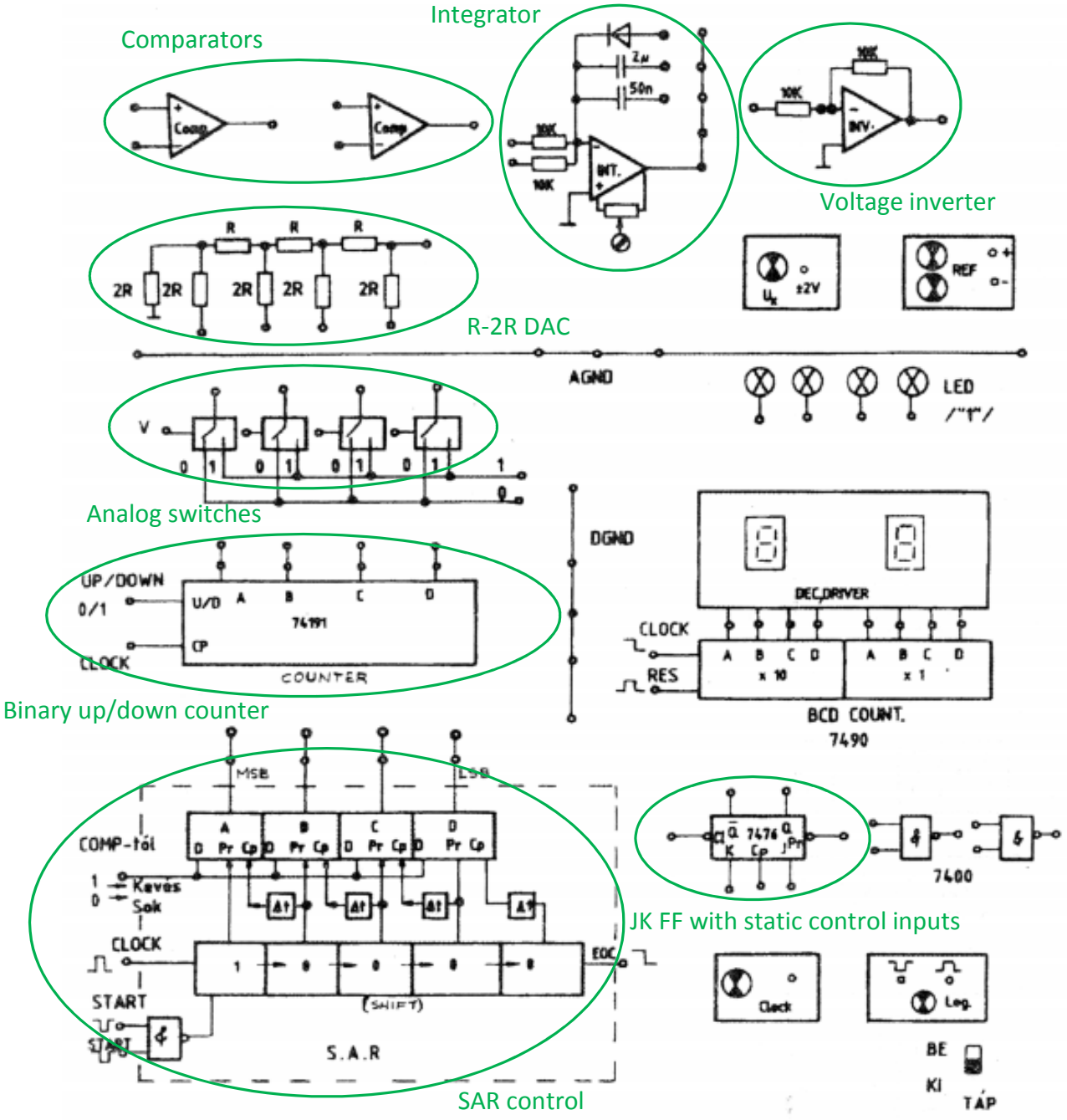


Figure 92. ADC application: ADC integrated as a part of an ECG Front End

The circuits of the ADC demo panel



Part 1.

Verify all functions of the demo panel!

Procedure

Apply the power supply, then verify the analog circuits:

- Verify the operation of the operational amplifiers!
    - Check if the comparator output changes with changes of the input voltage: connect AGND to the inverting input, and apply input voltage from the ±2 V source. The output can be checked by oscilloscope, it must be "leap' between the two power supply voltages, if the input voltage changes its sign.
    - Check the integrator! Make the connection between the free end of the 2 µF capacitor, and the amplifier output! Check this output by oscilloscope: if one of the input resistor is connected to the negative reference, the output rises relatively slightly to the positive supply line, and if the input voltage is altered to the positive reference (put the banana plug from REF- to REF+), the output falls slightly to the negative supply line. The time constant can be calculated as the following: $\tau = R \cdot C$. (It is the time required to charge the capacitor through the resistor, by ~63% of the difference between the initial value and final value or discharge the capacitor to ~37%.)
    - Check the inverting amplifier! Apply input voltage to it, and check the output voltage!
- Verify the operation of the reference source, it has two outputs, one is a negative, and one is positivevoltage, related to the ground (AGND)!

Then verify the digital and mixed signal circuits:

- Verify the operation of the clock source by an oscilloscope!
- Verify the operation of the CMOS switches! The control voltage is 0/5 V, **the analog voltage at the inputs can't exceed ±5 V!** (For example apply DC voltage to the common points, and check if this voltage appears and disappears on the '1' and '0' lines. If a CMOS switch's V control line is 0, the switch closes the line '0' to the output, if V is 1, the line '1' will be switched to the output.)
- Verify the operation of the 74191 counter! Apply the clock to the CP clock input, set U/D to high! Examine the outputs on the LEDs! Then set U/D to low, and, and examine the outputs!
- Verify the operation of the 7476 JK FF and the 7400 NAND gates
- The operation of the SAR logic will be checked during the experiment, and the BCD counters are not checked, because they are not used in this experiment.

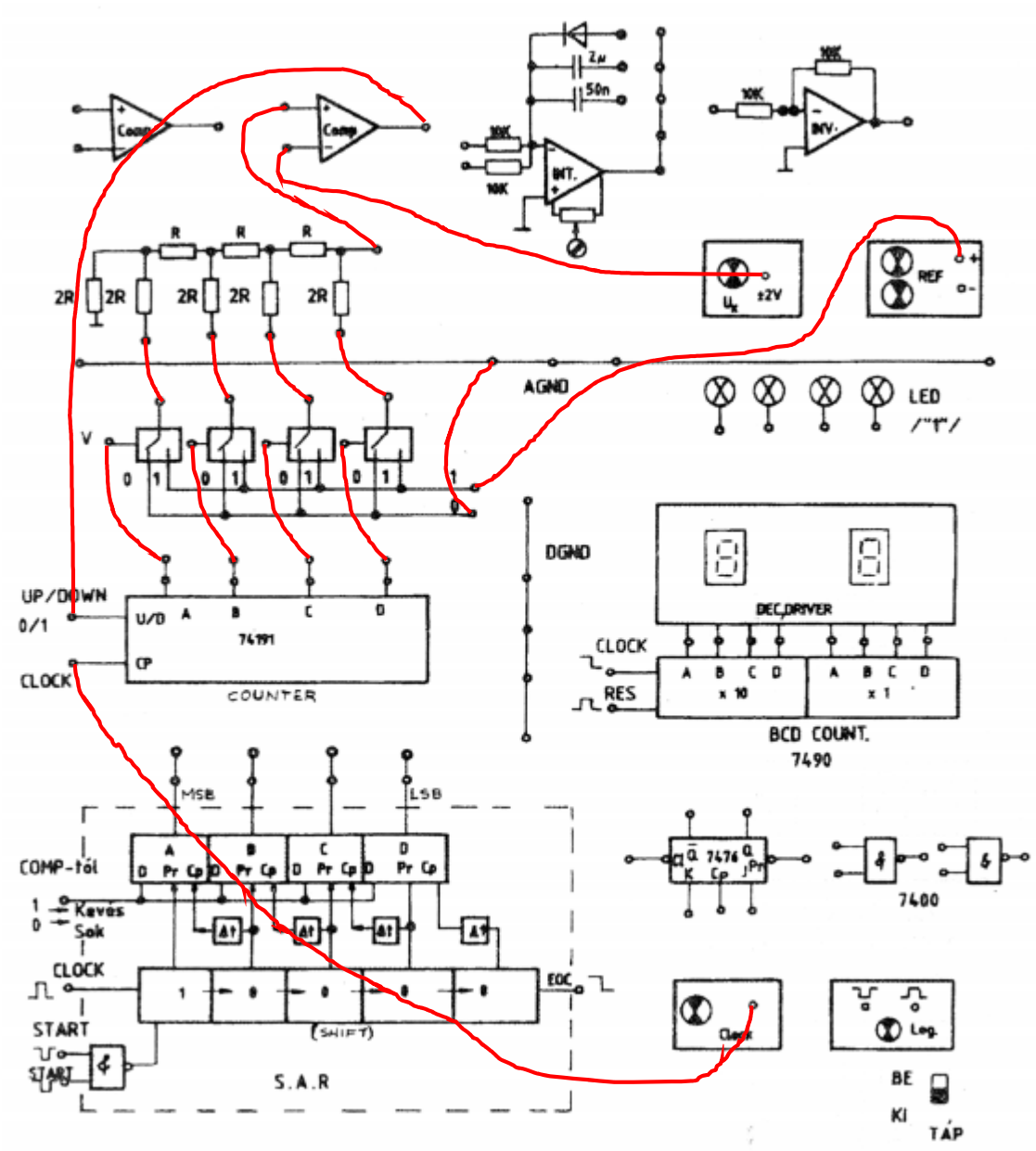
If there is something problem of the above, report it to the instructor!

Part 2.

Build a tracking ADC, and examine its operation!

Experiment

Build a tracking (also known as 'servo') ADC! See the diagram below!



When the input voltage (Ux) is higher, than the DAC voltage at the comparator's non-inverting input, the output of the comparator will be at low level, which forces the 74191 counter to count up. The counter output drives the analog switch group. If a high level is applied to the

switch control input (named V), the switch puts the reference voltage to the corresponding DAC input, or if the V is low, the switch puts GND to this input. So if the counter counts up, the DAC output voltage will rise, if the counter counts down, the DAC output voltage will fall. If the DAC voltage exceeds the input voltage (Ux), the comparator output will go to high level, and the counter turns to count down… So the DAC output voltage (and therefore the digital value at the counter output) will always follow the input voltage.

Examine the operation of this ADC, while input voltage is a DC voltage. Measure the DAC voltage and the input voltage by oscilloscope, try, how the DAC voltage follows the input voltage. Examine the operation of the comparator! Then apply a low frequency AC signal (f < 100 Hz) to the input (instead of Ux), and see the DAC output and input voltage on the scope. The input voltage must be in the range of 0 V … + 1.5 V! Then change the input signal's frequency to f > 1 kHz, examine the output!

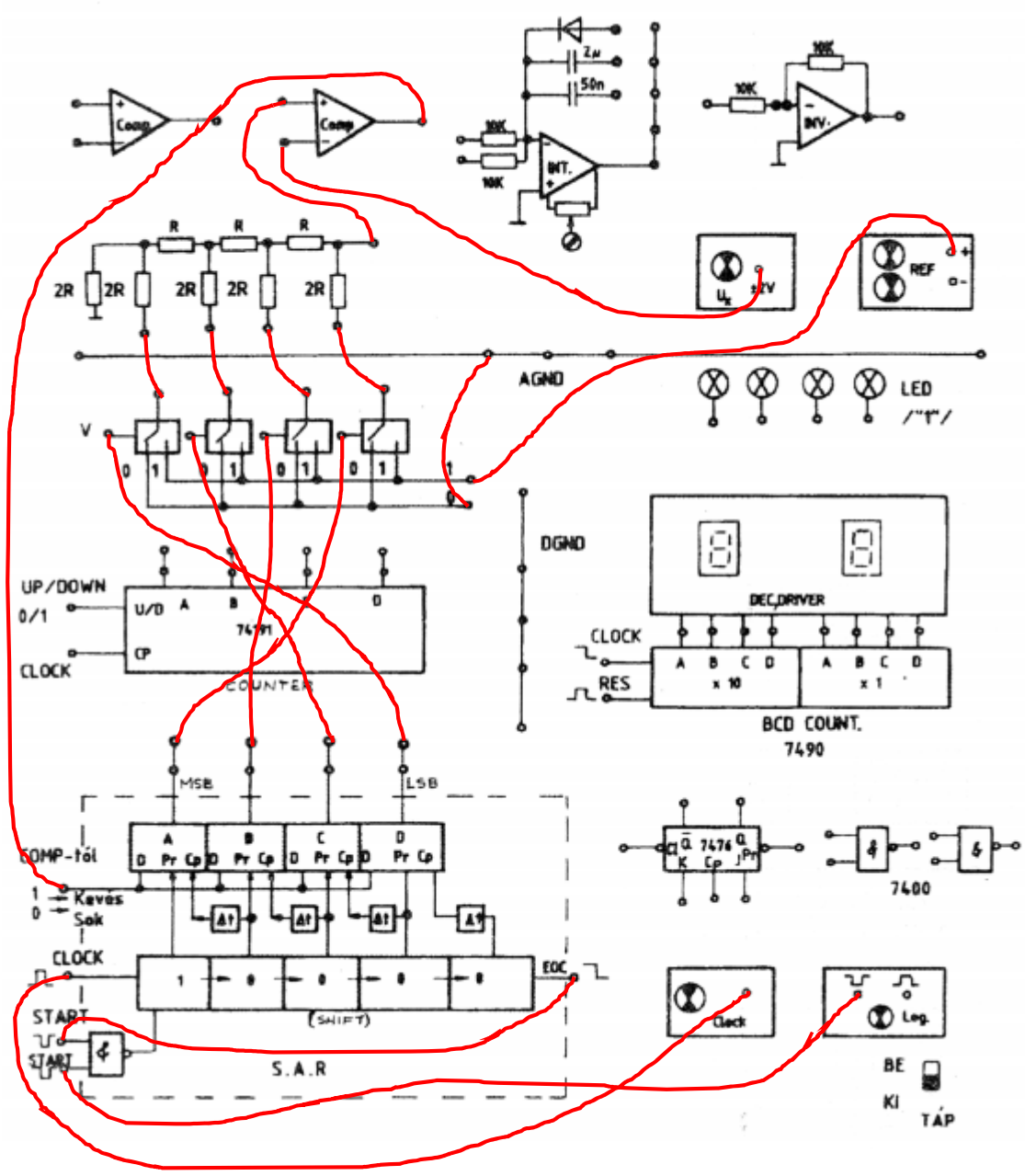
Part 3.

Build a SAR ADC, and examine its operation!

Experiment

Build a SAR ADC, see the diagram below! At the start (which can be caused by START pulse or EOC falling edge), a '1' will be written into the shift register's MSB. The corresponding analog switch will apply the reference voltage to the MSB of the R-2R DAC, so the DAC output voltage will be the half of the reference voltage. If it is higher than the input voltage, the comparator's output voltage will be high: it represents, that half of the reference voltage is 'not enough', so this bit must remain 1. If the DAC output voltage is lower than the input voltage, the comparator's output voltage will be 0 V: it represents, that the half of the reference voltage is 'too much', so this bit must be reset. Then, the process will continue by the next bit, and so on, to the LSB. When the '1' from the last bit of the shift register will be shifted out, and the output will be '0', this signs, that this is the end of conversion (EOC).

Examine the operation of this ADC, the input voltage is DC. Measure the DAC voltage and the input voltage by oscilloscope, try, how follows the DAC voltage the input voltage, if it changes. The input voltage must be in the range of 0 V … + 1.5 V!

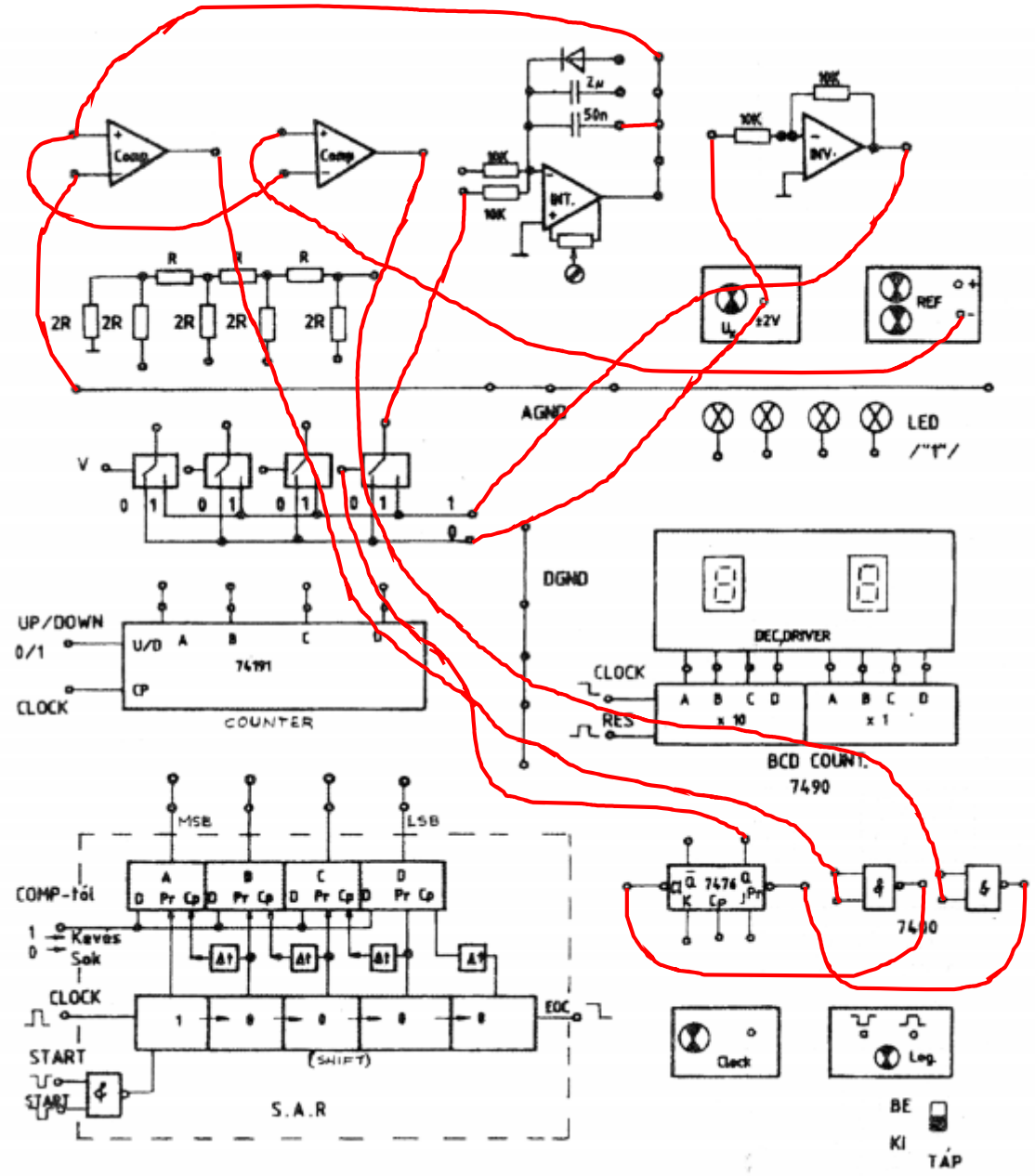


Part 4.

Build a VCF ADC, and examine its operation!

Experiment

Build a VCF ADC, see the diagram below! Examine the operation of this ADC, apply a DC voltage to the input. Measure the integrator output voltage by oscilloscope, try, how this voltage depends on the input voltage. What are the limits of the input voltage level, and how depends the output frequency on the input voltage?


Comp
Comp
2µ
50n
INT.
INV.
10K
100K
R
2R
AGND
±2V
REF
LED
/"1"/
V
DGND
UP/DOWN
0/1
CLOCK
U/D
74191
COUNTER
DEC.DRIVER
RES
x 10
x 1
BCD COUNT.
7490
MSB
LSB
COMP-tól
Kevés
Sok
START
(SHIFT)
EOC
S.A.R
7476
7400
Clock
Log.
BE
KI
TÁP

Questions

1. What is the function of the A/D converters? Explain their transfer function!
2. What are the static errors of the A/D converters? Specify them!
3. Draw a simple tracking ADC structure! How does it work?
4. Draw a simple VFC ADC structure! How does it work?
5. What are the advantages and disadvantages of the SAR ADCs?
6. How does the dual slope ADC work? What is the main advantage of this structure? (This type of ADCs is not negotiated in this manual, but it is a very important. See – for example – the page 6.73 of the document at the first link below)

Useful links:

http://www.analog.com/library/analogDialogue/archives/43-09/EDCh%206%20Converter.pdf

https://sisu.ut.ee/sites/default/files/imageprocessing/files/digitizn.pdf

http://www.analog.com/media/cn/training-seminars/design-handbooks/MixedSignal_Sect2.pdf

Experiment 8.

Speed control of a DC electric motor

Objectives

- To become familiar with the closed-loop, analog controller circuits.
- To gather experience in construction of complex circuits.

Components

Component	Description	Number of items	Comment
LM7805	voltage regulator	1	or equivalent
uA741	operational amplifier	1	or equivalent
SN74123	monostable multivibr.	1	or equivalent
SN74132	Schmitt-trigger	1	or equivalent
BC301	transistor, NPN	1	with heat sink
BC303	transistor, PNP	1	with heat sink
100 nF	capacitor	4	ceramic
47 nF	capacitor	1	ceramic
4.7 MΩ	resistor	1	
100 kΩ	resistor	1	
68 kΩ	resistor	1	
18 kΩ	resistor	1	
4.7 kΩ	resistor	1	
1.5 kΩ	resistor	1	
100 Ω	resistor	1	
6.2 Ω	resistor	3	high power, min 3 W
10 kΩ	potentiometer	1	

Instruments

Items	Comment
Power supply	dual
Digital multimeter	
Digital oscilloscope	dual channel
Motor unit	DC motor + optocoupler and its additional elements in one fitted unit

Background

Basics

In this experiment, we will build a motor speed controller circuit. In this circuit we will produce a signal, which is proportional to the speed of the motor. This signal will be compared to a reference, which maintains the set point, i. e. represents that speed, what is expected. The result of this comparison is a signal, which is the difference between the desired and the expected speed of the motor. If there is a difference between the setpoint and measured speed, our circuit will correct the speed of the motor, in order that this difference let be zero, so the required and current speed will be equal. This theory is called negative feedback control.

Control systems

On the figure below you can see a block diagram, which shows the basic elements of a feedback control system. The functional relationships between these elements are easily seen. An important factor to remember is that the block diagram represents flowpaths of control signals, but does not represent flow of energy through the system or process.

Below are various terms related along with the closed-loop block diagram. The process is the system or procedure by that a particular quantity or condition is controlled. This is also known as the controlled system.

The controller and the final control element are components required to produce the appropriate control signal applied to the controlled system. These components together are also known as the ‘controller’.

The feedback elements (measurement sensor/transmitter) are components required to identify the functional relationship among the feedback signal and the controlled outcome.

The reference point is an external signal applied to the summing point of the control system to cause the plant to generate a specified action. That signal represents the desired value of a controlled variable and is also known as the ‘setpoint’.

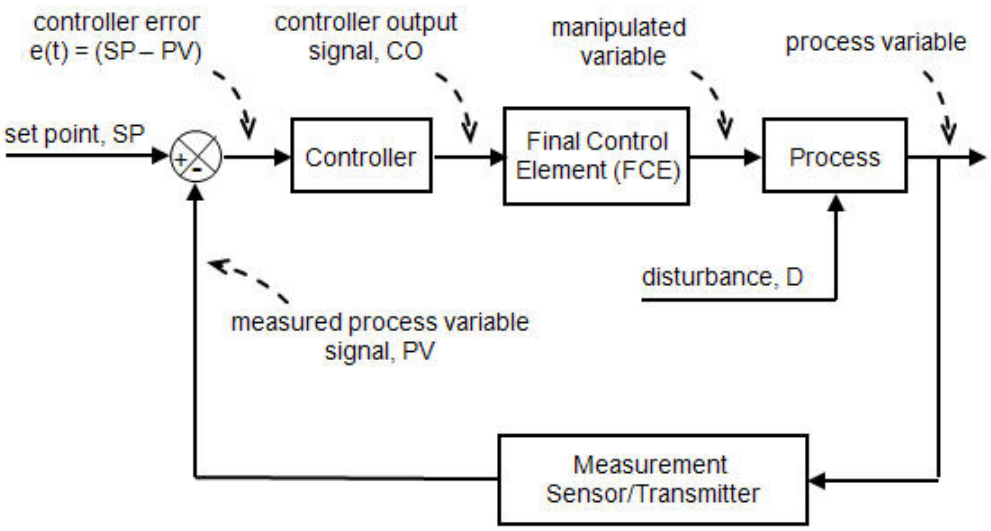


Figure 93. Closed-loop control system

In our experiment, the controlled system is a DC electric motor, and the process variable is the speed of this motor. We measure the speed by using an optical sensor, and the pulses generated by this sensor will be formed and filtered, so comes into being the measured process variable signal. The setpoint is created by a potentiometer, on its slider will be a DC voltage, which will be compared with the measured process variable signal, by an operational amplifier, which operates as a subtracter. This operational amplifier will drive a power stage, which is the control element. This power amplifier will drive the DC motor. On the figure below, you can see this control circuit, and its blocks. (This circuit can be found in a larger size at the end of this manual.)

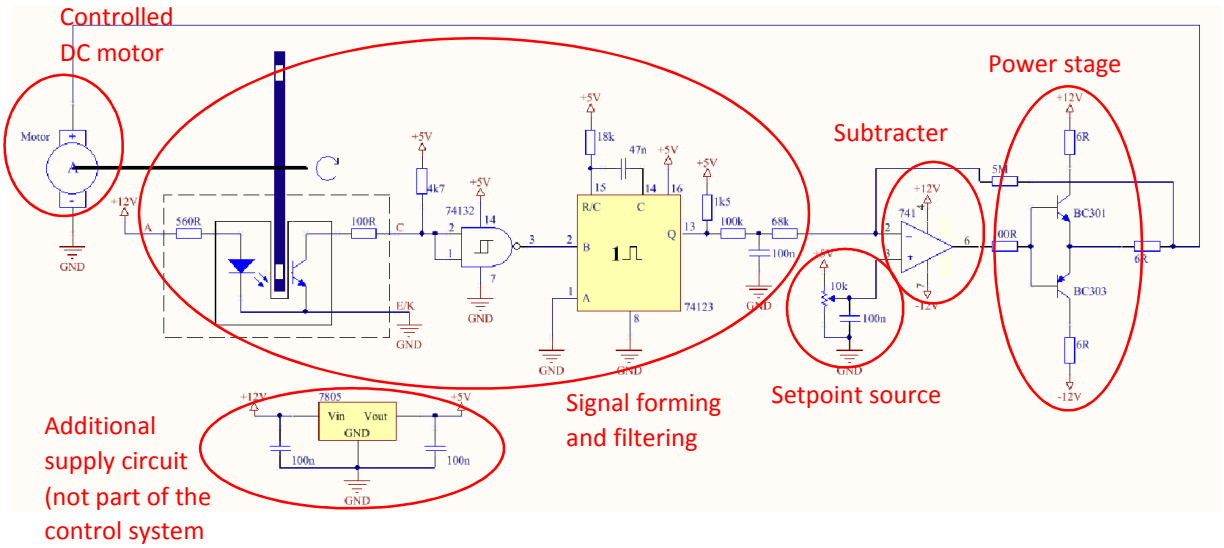


Figure 94. Motor control circuit discussed in this manual

In the next, we will build up this circuit above by subcircuits, and will perform the tests.

Part 1.

Prepare the supply voltages for the circuit!

Procedure

The circuit needs 3 supply voltage: +12 V, -12 V and +5 V. The power supply has only two channels, so he +5 V will be created from the +12 V. After connecting the power supply outputs to produce ±12 V dual supply, build the +5 V subcircuit using 7805 voltage regulator. Check if its output voltage is OK.

Part 2.

Identify the DC motor unit!

Procedure

The DC motor unit has a 5 points connector. Check, and register, which points are for the DC motor and which points are for the optocoupler.

Function	Pin number / color
Motor outlet 1	
Motor outlet 2	
Optocoupler diode – Anode	
Optocoupler diode – Cathode and transistor – Emitter	
Optocoupler transistor – Collector	

Part 3.

Check the operation of the optocoupler!

Procedure

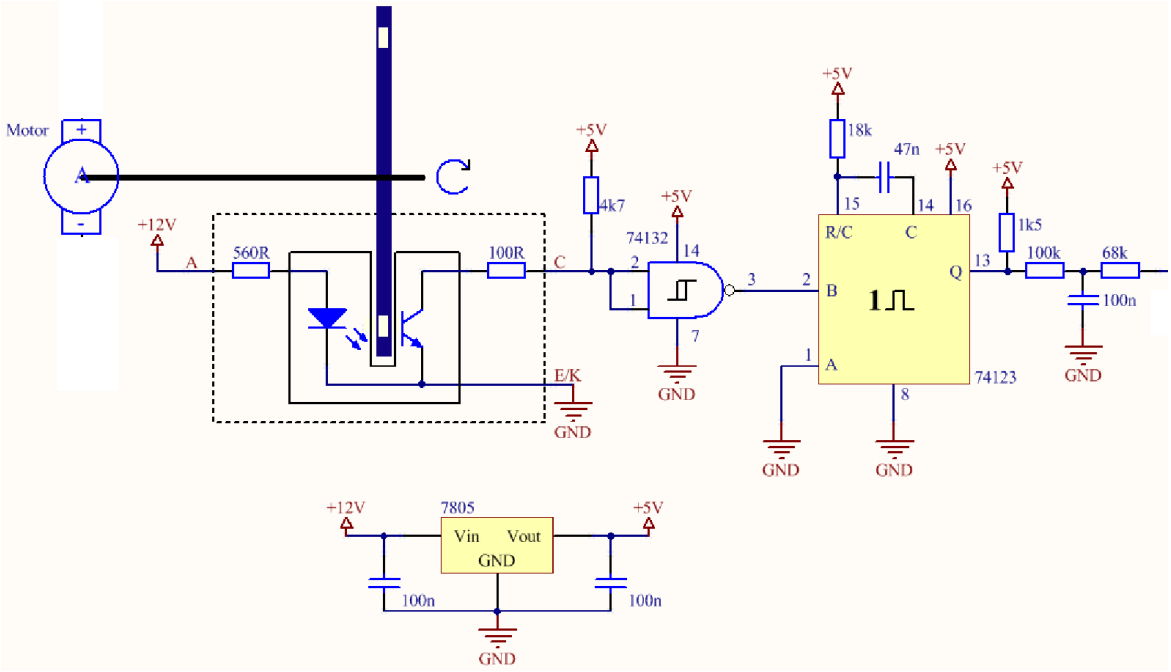
Provide the +12 V to the diode of the optocoupler. The diode emits in the infra-red zone, so its light cannot be seen. (But infra-red light is detected by – for example – the phone's camera.) Connect the +5 V to the transistor's collector in the optocoupler, via a 4.7 kΩ resistor! Check the signal by the oscilloscope at the collector, if the shaft of the motor is rotated by hand! This signal is not really suitable for further application, because its shape and amplitude varies by the speed of the shaft.

Part 4.

Build the Schmitt-trigger NAND stage with 74132, and see how is the signal shape formed!

Procedure

Build the monostable multivibrator, as it can be seen on the circuit diagram. Don't forget to put logic 0 (GND) to the input 'A'. At the output, there will be a constant-time pulse for every rising edge of the input B. Check the signal at the output, examine the input and output relation! Then build the low-pass RC filter circuit, and check if the DC voltage of the filter changes, when the shaft of the motor is rotated by hand. Now we finished building this set of subcircuits. The built circuit is below.

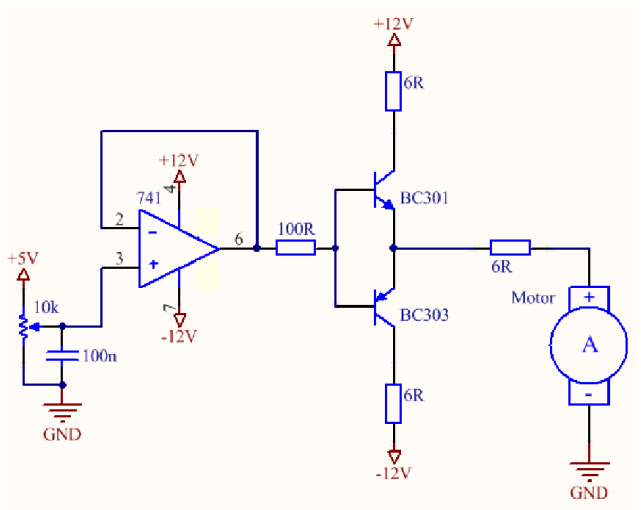


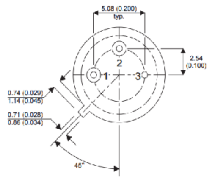
Part 5.

Build the driver circuit for the motor!

Procedure

Now, we begin to build the motor control circuit. First, we build an open-loop controller circuit, a driver. Its circuit diagram is shown below.





TO39 (TO205AD)
PINOUTS

1 – Emitter 2 – Base 3 – Collector

The resistors with value of 6 Ω are for only circuit protection. Use a heat sink (star shape) on BC301! Don't allow the voltage on the motor to exceed 4 V! On the transistor case there is the same potential as on the collector, so be careful, don't cause short circuit! Try the motor driver, if you adjust the potentiometer, the motor will rotate at various speeds! Try to stop the motor by hand, remember the force (torque), which is needed! Set the motor voltage to 1 V, 2 V, 3 V and 4 V, and measure the motor speed! The motor speed (rpm) can be calculated using the signal at the Schmitt-trigger output. (The number of the slots on the disk is 60.)

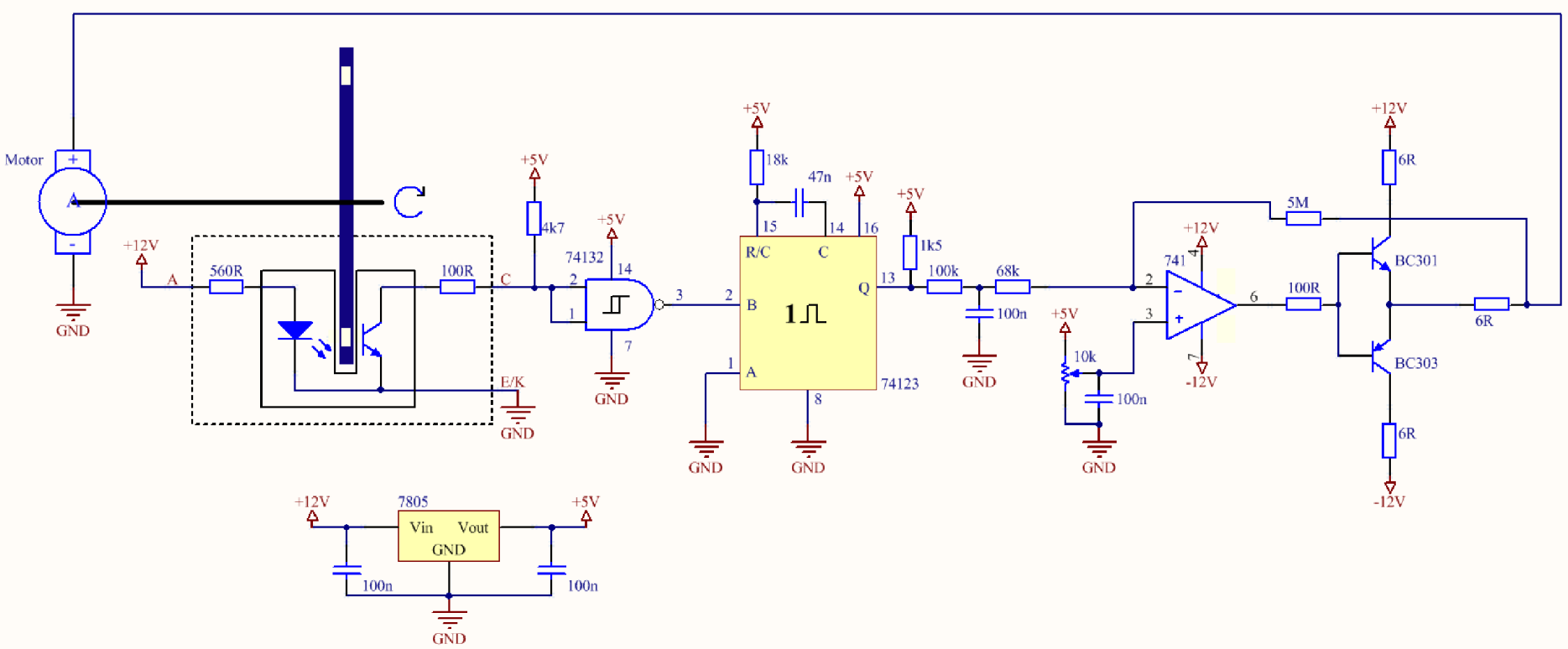
Part 6.

Modify the driver circuit for creating the final motor control circuit!

Procedure

Modify the driver circuit according to the full circuit diagram (circuit diagram at the end of this manual), and join the modified driver circuit, and the circuit built in the Part 4! Now you built up a closed-loop control system. Try the circuit, adjust the potentiometer, and if the circuit works, the motor speed will change too. Try to stop the motor again by hand, and compare the torque needed to stop it in the Part 4! Examine all the specific signals in the circuit in free-run mode, and if the motor's shaft is loaded. (This points are in minimum: monostable output, filter output, amplifier output.) Set the setpoint to 0 V, and try to rotate the shaft! Measure the current consumption of the (driver) circuit! What happened, and why?

The full circuit of the closed-loop motor speed-control system



Questions

1. Draw a simple model of closed-loop control system! Describe shortly the blocks and signals!
2. What is a difference between the open-loop and closed-loop control?
3. How does work the optical method for sensing motor speed (described in this manual)?
4. What is functionality of a Schmitt-trigger?
5. What is the functionality of a monostable multivibrator?

Useful links:

https://www.site.uottawa.ca/~rhabash/ELG4152LN01.pdf

https://www.youtube.com/watch?v=bPyh1F-_kpg

http://www.facstaff.bucknell.edu/mastascu/econtrolhtml/Intro/IntroNotes/IntroNote_VeryBasic.html

http://www.facstaff.bucknell.edu/mastascu/InstrumentationCourse/MeetingNotes/MtgControlIntro.htm

Experiment 9-10.

MPLAB IDE – Microcontroller simulation 1-2.

Objectives

- Acquire knowledge in microcontroller development tools.
- Learn, how the microcontroller software development takes place.
- Write and debug simple programs, learn the basic debugging tools.

Components

None

Instruments

None

Other

Items	Comment
PC	with installed MPLAB IDE
example source files	test1.asm, test2.asm, test3.asm

Background

Microcontroller system development

Developing software (and hardware) for a microcontroller based systems involves the use of a range of tools that can include editors, assemblers, compilers, debuggers, simulators, emulators and Flash/OTP programmers. To the beginner to microcontroller development it is often not clear how all of these different components play together in the development cycle and what are the differences between e.g. emulators and simulators.

In the following, we will try to give a short explanation of the different tools involved in the microcontroller development cycle, with a particular focus on the used tools in this experiment. First, let's see, how the development of a microcontroller system takes place.

Starting the development process, we have to recognize the task or problem to solve it. First the system must be specified. We have to do a plan – called system specification -, which contains the operational and performance specifications of a system. Completed it, we already have decided what functions will be implemented by hardware and what functions will be implemented by software. For hardware development there are several tools: for example circuit designer software, circuit simulator software, printed circuit board designer software, development boards for hardware functions, oscilloscope, logic analyzer, and so on, but in this manual we don't pay attention to hardware design, only to software design.

Coding, assembling and compiling

In most cases it is previously determined, which computer-language must be used for programming the microcontroller in the system. Assembly language is a low-level programming language, in which there is a very strong (generally one-to-one) correspondence between the language and the architecture's machine code instructions. Each assembly language is specific to a particular computer architecture, in contrast to most high-level programming languages, which are generally portable across multiple architectures. Assembly language is converted into executable machine code by a utility program referred to as an assembler. Assembly language uses a mnemonic to represent each low-level machine instruction or operation. Typical operations require one or more operands in order to form a complete instruction, and most assemblers can therefore take labels, symbols and expressions as operands to represent addresses and other constants, freeing the programmer from tedious manual calculations. C language is a high-level programming language, with strong abstraction from the details of the computer. For example, in case you add two variables, and store the result in a third variable, you don't worry about the microcontroller's data registers, status flags, and others. You must only write this: 'c = a + b;'. A C language program is easily portable between microcontroller families or types, but always requires interpreting or compiling.

The software code for a microcontroller is written in a programming language of choice (often assembly or C). This is called 'source code', written by using a standard ASCII text editor and saved as an ASCII text file. Programming in assembly language involves learning a microcontroller's specific instruction set (assembler mnemonics), but results in the most

compact and fastest code. A higher level language like C is mostly independent of a microcontroller's specific architecture, but still requires some controller specific extensions of the standard language to be able to control all of a chip's peripherals and functionality. The penalty for more portable code and faster program development is a larger code size (120%...140% compared to assembly language source code).

Next the source code needs to be translated into instructions the microcontroller can actually execute. A microcontroller's instruction set is represented by 'op codes'. Op codes are a unique sequence of bits ('0' and '1') that are decoded by the controller's instruction decode logic and then executed. Instead of writing opcodes in bits, they are commonly represented as hexadecimal numbers (whereby one hex number represents 4 bits of a byte, so it takes two hex numbers to represent 8 bits or 1 byte). For that reason a microcontroller's firmware in machine readable form is also called "hex code' and the file that stores that code 'hex file'.

Assemblers or (C-) compilers translate the human readable source code into 'hex code' that represents the machine instructions (op codes). To support modular code and reusable libraries of code, most assemblers and compilers today come with linkers and librarians. Linkers, link code modules saved in different files together into a single final program. At the same time they take care of a chip's memory allocation by assigning each instruction to a microcontroller memory addresses in such a way that different modules do not overlap. Librarians help you to manage, organize and revision control a library of re-usable code modules. Once the ASCII source code text file has been assembled (by an assembler program) or compiled (by a compiler program) and the files have been linked (with the linker), the output results in a number of files that can be used for debugging the software and programming the actual microcontroller's memory.

Debugging the microcontroller's code

For debugging, there are basically two tools, the simulator and the emulator. In integrated development systems the software component for debugging (simulating and emulating) are implemented, and can be used similarly to each other. You can run or step the program, you can set breakpoints, etc. But the simulation and the emulation are fundamentally different methods for debugging. If you use the simulator, on the PC's display you can see how the program runs, and how the registers change, where the program counter points, etc., but what you see, it doesn't happen. (In short: the operation of the algorithm can be controlled.) If you emulate the program running, you use an external emulator hardware (which in some cases is implemented in the microcontroller itself), and what you see on the PC's display, it does happen in the real world, owing to the external hardware. (In short: the operation of the full target equipment can be controlled.)

Simulators try to model the behavior of the complete microcontroller in software. Some simulators go even a step further and include the whole system (simulation of peripherals outside of the microcontroller). No matter how fast your PC, there is no simulator on the market that can actually simulate a microcontroller's behavior in real-time. Simulating external events can become a time-consuming exercise, as you have to manually create 'stimulus' files that tell the simulator what external waveforms to expect on which

microcontroller pin. A simulator can also not talk to your target system, so functions that rely on external components are difficult to verify. For that reason simulators are best suited to test algorithms that run completely within the microcontroller (like a math routine for example). They are the perfect tool to complement expensive emulators for large development teams, where buying an emulator for each developer is financially not feasible.

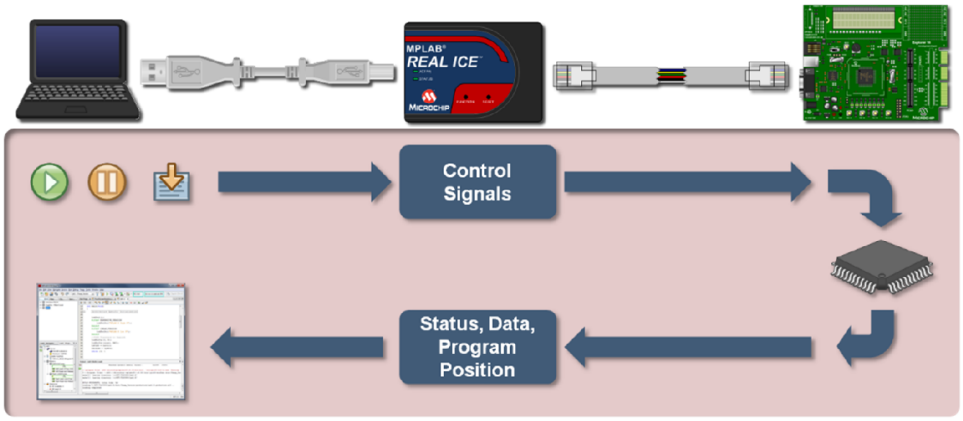


Figure 95. A microcontroller debugger system

So, the debugger is a piece of software running on the PC, which has to be tightly integrated with the emulator (hardware) that you use to validate your code. For that reason all emulator manufacturers ship their own debugger software with their tools, but also compiler manufacturers frequently include debuggers, which work with certain emulators, into their development suites.

A debugger allows you to download your code to the emulator's memory and then control all of the functions of the emulator from a PC. Common debugging features include the capability to examine and modify the microcontroller's on-chip registers, data- and program-memory, pausing or stopping program executing at defined program locations by setting breakpoints; single-stepping (execute one instruction at a time) through the code; and looking at a history of executed code (trace).

IDE

So far we've talked about several different pieces of software: text Editor, assembler and/or compiler, linker, librarian and debugger. You can easily imagine that it can become quite a time-consuming challenge to alternate back and forth between all of these programs during the debugging process (discover a bug, edit the source code, compile it again, link it again, download the modified code to the emulator, etc.). This is where an integrated development environment (IDE) comes in.

An Integrated Development Environment puts all of the previously discussed software components under one common unified user interface, so that it becomes possible to make a code change and get the modified code loaded into the emulator with a few mouse clicks, instead of dozens. A good IDE allows you for example to click on a syntax error message produced by the compiler and have the source code with the highlighted offending instruction pop up for editing in the text editor. One click of a button and the modified code gets

retranslated, linked and downloaded to the emulator. An IDE allows you to store the configuration settings for a project - like compiler switches, or what flavor of chip to emulate - so you can easily recreate a project later on. Some IDEs are flexible enough to allow you to incorporate different choices of third party tools (like compilers and debuggers), others only work with a manufacturer's own tool chain.

Other debugging tools

In case you have to debug your code and testing your application there are several different tools you can utilize that differ greatly in terms of development time spend and debugging features available. Oscilloscope, logic analyzer, digital signal generator are not only suitable for debugging the hardware, but also for debugging the software. You can use various microcontroller development boards, if you haven't yet the final hardware, you can try almost all of the functions of the software (of course in most cases you have to build some supplementary hardware). If the application requires, you can use protocol analyzers (for example for RS232, I2C, USB or CAN buses), memory programmers, and so on…

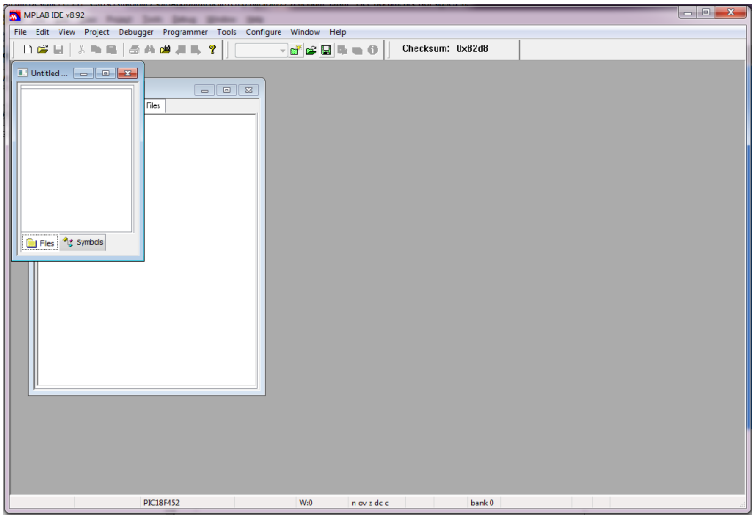
During this experiments we will only use the MPLAB IDE, its editor, assembler, linker and simulator parts, but during the 'Signal and image processing laboratory' course you will use a development board also, with several peripherals, and an in-circuit debugger (ICD3 for Microchip's microcontrollers) for 'real-world' program running.

Part 1.

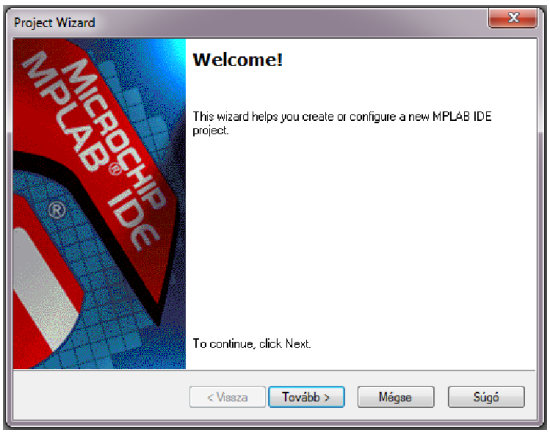
Set the MPLAB IDE for beginning the simulation, create a new project!

Procedure

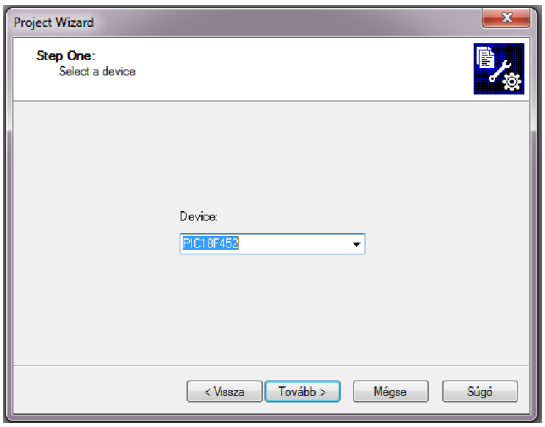
Start the MPLAB! The next window will be on the display.

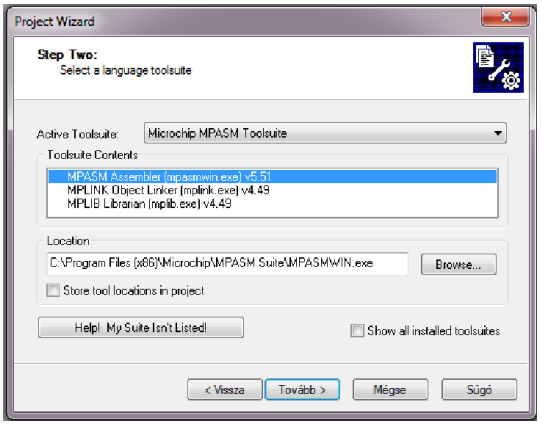


Start the *Project → Project Wizard*!

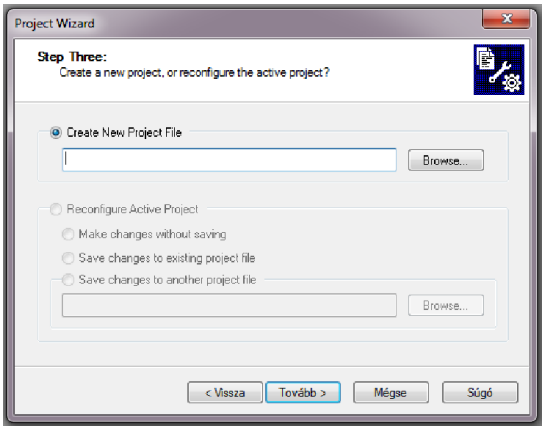


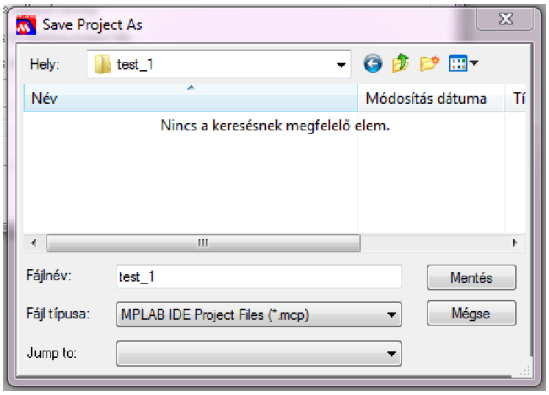
Select Device and language toolsuite!



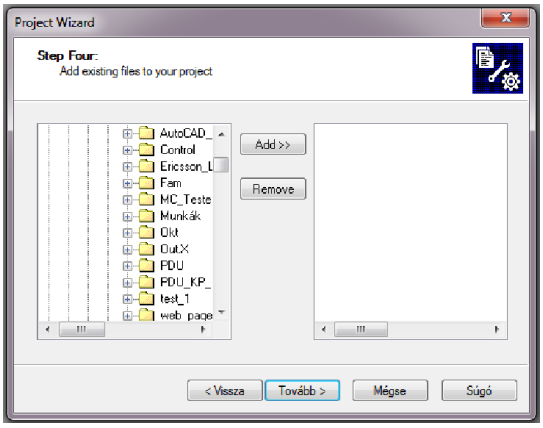


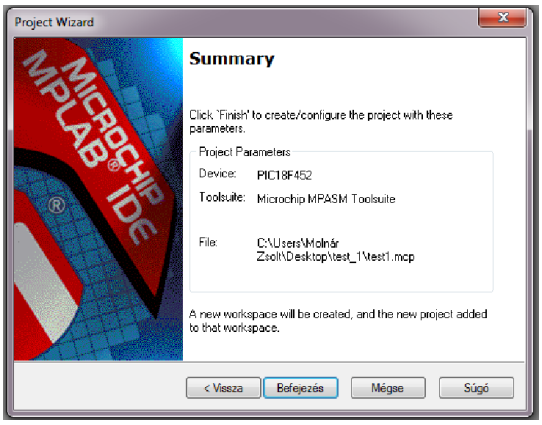
Set the project name and the path!





Since there is no source file yet, skip this step (Tovább)! The project is created!





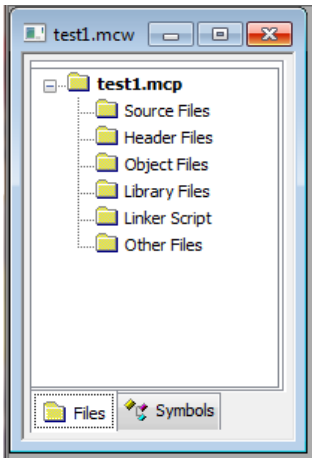
Part 2.

Simulate a simple program, which adds two unsigned 8-bit number, and stores the result in a 16-bit variable!

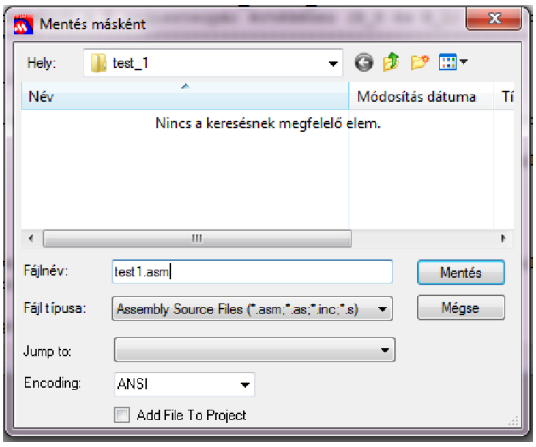
Important! Before you execute the following tasks described below, it is necessary to read and interpret the source code. Merely the loading and building the source code, then running the program is not enough to learn the basics of assembly programming and debugging.

Procedure

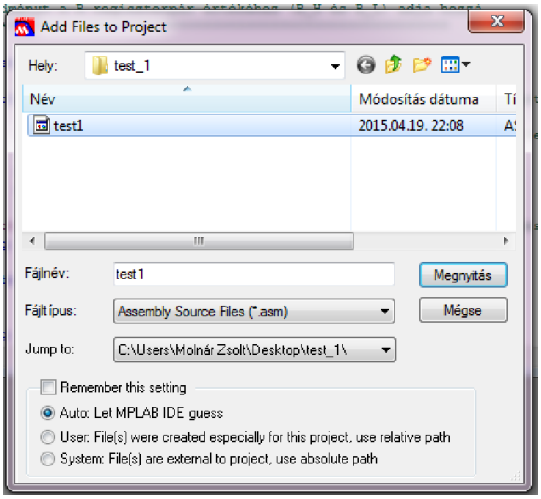
In the project window, you can see this:



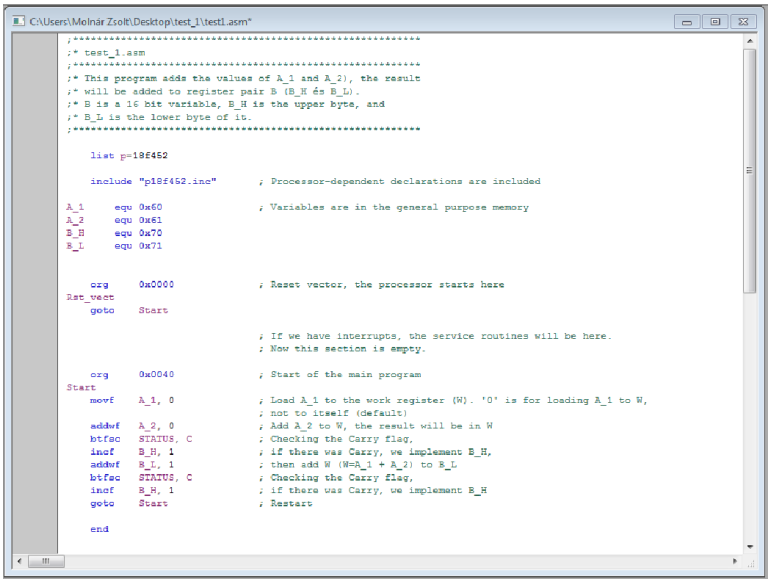
Create a new source file: *File* → *New*



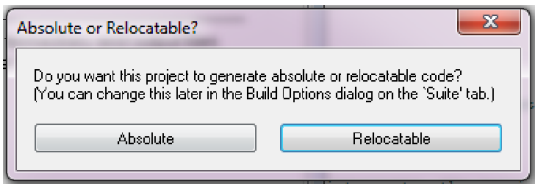
Then in the *Project Window* right click to *Source Files*, and select *Add Files*…! Select the created source, then click OK!



Copy the contents of the included test1.asm to your source file, and you can see this:



Build the project: *Project → Build All*, select *Absolute code* (see below)!



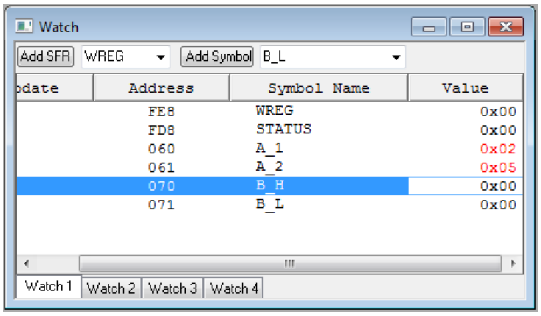
If there is all right, you can see the message: 'Build succeeded', or if there are errors, rectify them!

Select the debugger for this project: *Debugger → Select Tool → Mplab Sim*! The icon set (shown below) will be appeared.

The icons are from left to right:

- **Run** (runs the program)
- **Pause** (stops the program, but the running of the program can continue from the actual point, if you want)
- **Animate** (slow run, at every step the display will be refreshed (for example the register values)
- **Step Into** (or Single Step) (only executes one row of the program, in our case one instruction)
- **Step Over** (the subroutines are executed as a single instruction)
- **Step Out** (if the program is in a subroutine, the execution stops only when the subroutine ends)
- **Reset** (resets the processor)
- **Breakpoint** (defines a breakpoint, but it is simpler to double click on the instruction…)

Then switch on the Watch window: *View → Watch*! Add the special function registers WREG and STATUS by using Add SFR! Add the defined symbols A_1, A_2, B_H, B_L by using *Add Symbol*! Then set this values to test the program: A_1 = 0x02, A_2 = 0x05. You can modify the values by double-clicking on the actual value of the register.



Using Single Step function, check the program execution! If you on the row ‘goto Start’, modify the values of A_1 and A_2: A_1 = 0xAC, A_2 = 0x80. Run the program again step by step!

Then double click to the ‘movf’ instruction below the Start label, you set a breakpoint. If the program execution reaches this point in Run mode, the simulator stops. Try this function, and use the possibility of stopped state to modify registers, and test the programs with other values!

Part 3.

Simulate a simple program, which shows bit masking!

Procedure

Copy the contents of Test2.asm to the window containing the program. This example shows the commonly used bit masking techniques. Simulate the program in single step mode, and try other masking constants!

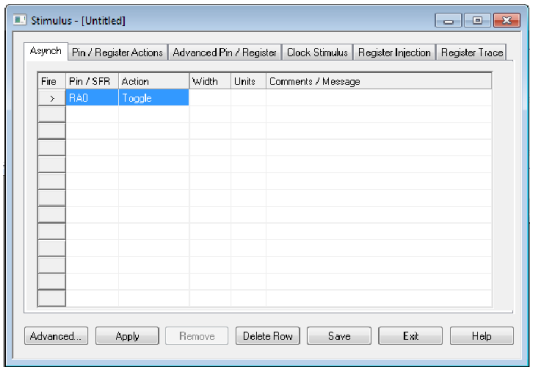
Part 4.

Simulate a simple program, which uses stimulus to simulate an external push-button!

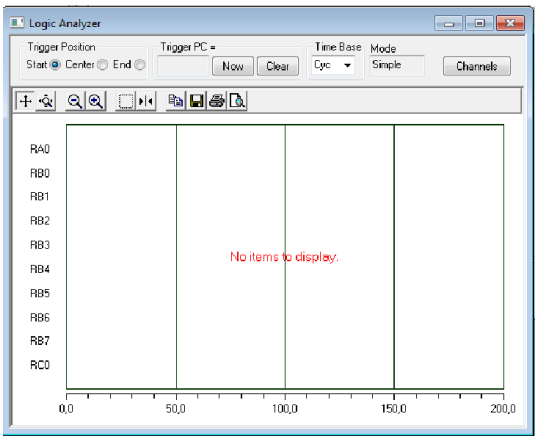
Procedure

Copy the contents of Test3.asm to the window containing the program. To simulate the program, we need to trigger the bit stepping applying high level to RA0. In the real world, it means, for example pushing a switch, or waiting a high level on a digital line. In the simulator, we must simulate this event. The tool for it the Stimulus.

Start the stimulus: *Debugger* → *Stimulus* → *New Workbook*. We use Asynchronous stimulus on RA0, so set the values below, then click *Apply*! At every ‘Fire’ the simulated level on RA0 will be inverted.



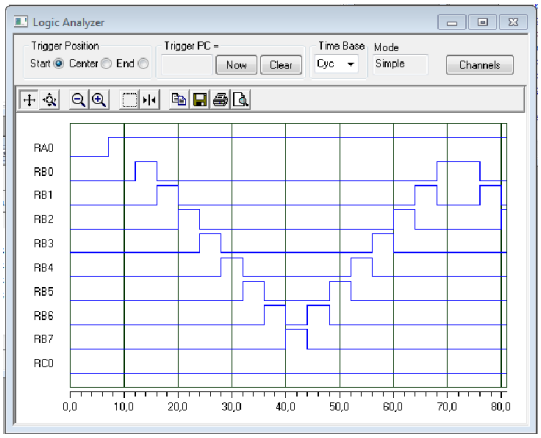
One method to see what happens is the Logic Analyzer. Start this *View* → *Simulator* → *Logic Analyzer*!



Set the channels to be examined clicking the Channels button! The examined channels are: RA0, RB0…RB7.

Then start the program execution by *Animate* function. Click Fire, and see what happens! If you want, you can stop or reset the simulation.

The simulation results are shown below.



Part 5.

Modify *Test3.asm* program for starting the stepping from left to right, and simulate it!

Modify *Test3.asm* program: stepping should be stopped immediately, when RA0 is set to zero! Continue stepping at the same output combination, when RA0 is set to one! Simulate the operation of the algorithm!

Part 6.

Write a program, which subtracts a 16 bit number (unsigned) from an other 16 bit number! Simulate the program!

Part 7.

Load back (or copy the contents) of test1.asm! Simulate this program using stimulus!

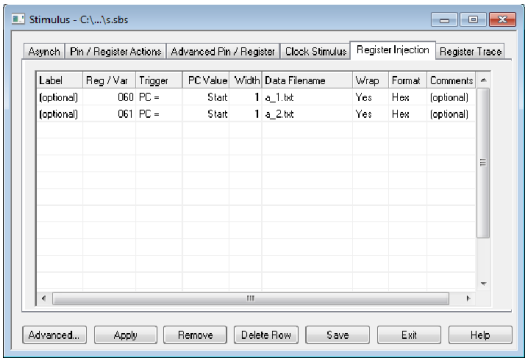
Procedure

In the Part 2, you set the values of A_1 and A_2 manually, and watched the values of B_L and B_H "manually' (by your eyes). It is more elegant, if the test is performed automatically, not to mention that case, if the test data has a huge amount.

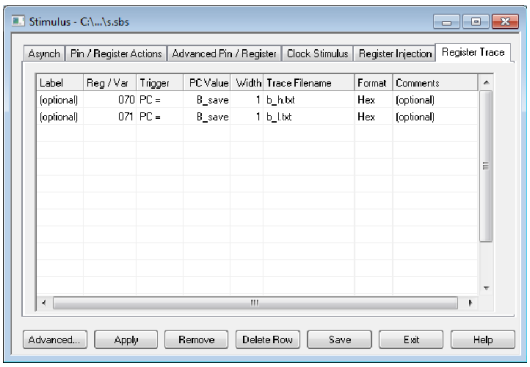
First, we must slightly modify the source. We must insert a row after Start label. We want to load A_1 and A_2 values from the stimulus, when the program execution reaches this point, but in the original source there is a use of the A_1 variable, so it can't be modified by the simulator in time. This row will be a '*nop*' instruction, which has no any effect to our program, but the simulator will be able to modify A_1 variable, before it is loaded. The other change is putting a label before (in fact: on) the row '*Goto Start*'. This is necessary for that the values of B_L and B_H can be saved at a right place (at this label), before the new values for A_1 and A_2 are loaded. Compile the program!

Next, prepare two text files for A_1 and A_2 to load the values of them. The minimal content for A_1 (as was in Part 2.): 0x02, 0xAC, and for A_2: 0x05, 0x80. Each value is in a row, and press ENTER for a new line! You can fill the files with additional values, but be careful to add the same number of the new values to each files! (During the tests writing this manual, we added one row to each files, with a value of 0x50 for A_1 and A_2 too.) Create two empty files for tracing B_L and B_H registers!

Open a new stimulus workbook, and at a *Register Injection* tab, fill the rows as the figure shows below! (60 and 61 are the memory addresses of variables A_1 and A_2.) (note, if the MPLAB doesn't allow to set some cell, first temporarily set in the *Reg/Var* column an another register, for example *PORTB*. Then fill all other cells, then modify the Reg/Var cell to the right value by writing it (60 and 61 are not selectable items)!



Then change to register tab, and fill the rows as the figure shows below! (70 and 71 are the memory addresses of variables B_H and B_L.)



Press *Apply*, and set a breakpoint to '*nop*' instruction! Run the program, and see and check the values of the trace files. Note, that the values are depended on the initial values of the B_H and B_L registers, if it is annoying, set their values to '0' before simulate the program!

If the initial values of B_H and B_L were 0x00, the results in the text files will be the next:

b_h.txt:	b_l.txt:
00	07
01	33
01	d3
01	da

Check the values! At the fourth jump to *Start* label the stimulus files for A_1 and A_2 will be read from their first values, because the value at the '*Wrap*' column of *Register Injection* stimulus is '*Yes*'.

Part 8.

Write a program, which receives a byte from the serial port, then sends back this byte incremented (byte + 1) back. Simulate the program!

Procedure

The sample program is included at the end of this manual. Use *Register Injection* function of the stimulus! RCREG will be loaded 'on demand', see the MPLAB help for stimulating the serial port.

Some parts of PIC18F452 data sheet

20.0 INSTRUCTION SET SUMMARY

The PIC18FXXX instruction set adds many enhancements to the previous PICmicro instruction sets, while maintaining an easy migration from these PICmicro instruction sets.

Most instructions are a single program memory word (16-bits), but there are three instructions that require two program memory locations.

Each single word instruction is a 16-bit word divided into an OPCODE, which specifies the instruction type and one or more operands, which further specify the operation of the instruction.

The instruction set is highly orthogonal and is grouped into four basic categories:

- **Byte-oriented** operations
- **Bit-oriented** operations
- **Literal** operations
- **Control** operations

The PIC18FXXX instruction set summary in Table 20-2 lists **byte-oriented**, **bit-oriented**, **literal** and **control** operations. Table 20-1 shows the opcode field descriptions.

Most **byte-oriented** instructions have three operands:

1. The file register (specified by 'f')
2. The destination of the result (specified by 'd')
3. The accessed memory (specified by 'a')

The file register designator 'f' specifies which file register is to be used by the instruction.

The destination designator 'd' specifies where the result of the operation is to be placed. If 'd' is zero, the result is placed in the WREG register. If 'd' is one, the result is placed in the file register specified in the instruction.

All **bit-oriented** instructions have three operands:

1. The file register (specified by 'f')
2. The bit in the file register (specified by 'b')
3. The accessed memory (specified by 'a')

The bit field designator 'b' selects the number of the bit affected by the operation, while the file register designator 'f' represents the number of the file in which the bit is located.

The **literal** instructions may use some of the following operands:

- A literal value to be loaded into a file register (specified by 'k')
- The desired FSR register to load the literal value into (specified by 'f')
- No operand required (specified by '—')

The **control** instructions may use some of the following operands:

- A program memory address (specified by 'n')
- The mode of the Call or Return instructions (specified by 's')
- The mode of the Table Read and Table Write instructions (specified by 'm')
- No operand required (specified by '—')

All instructions are a single word, except for three double-word instructions. These three instructions were made double-word instructions so that all the required information is available in these 32 bits. In the second word, the 4-MSbs are 1's. If this second word is executed as an instruction (by itself), it will execute as a NOP.

All single word instructions are executed in a single instruction cycle, unless a conditional test is true or the program counter is changed as a result of the instruction. In these cases, the execution takes two instruction cycles with the additional instruction cycle(s) executed as a NOP.

The double-word instructions execute in two instruction cycles.

One instruction cycle consists of four oscillator periods. Thus, for an oscillator frequency of 4 MHz, the normal instruction execution time is 1 µs. If a conditional test is true or the program counter is changed as a result of an instruction, the instruction execution time is 2 µs. Two-word branch instructions (if true) would take 3 µs.

Figure 20-1 shows the general formats that the instructions can have.

All examples use the format 'nnh' to represent a hexadecimal number, where 'h' signifies a hexadecimal digit.

The Instruction Set Summary, shown in Table 20-2, lists the instructions recognized by the Microchip Assembler (MPASM™).

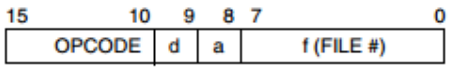
Section 20.1 provides a description of each instruction.

TABLE 20-1: OPCODE FIELD DESCRIPTIONS

Field	Description
a	RAM access bit a = 0: RAM location in Access RAM (BSR register is ignored) a = 1: RAM bank is specified by BSR register
bbb	Bit address within an 8-bit file register (0 to 7)
BSR	Bank Select Register. Used to select the current RAM bank.
d	Destination select bit; d = 0: store result in WREG, d = 1: store result in file register f.
dest	Destination either the WREG register or the specified register file location
f	8-bit Register file address (0x00 to 0xFF)
fs	12-bit Register file address (0x000 to 0xFFF). This is the source address.
fd	12-bit Register file address (0x000 to 0xFFF). This is the destination address.
k	Literal field, constant data or label (may be either an 8-bit, 12-bit or a 20-bit value)
label	Label name
mm * *+ *- +*	The mode of the TBLPTR register for the Table Read and Table Write instructions. Only used with Table Read and Table Write instructions: No Change to register (such as TBLPTR with Table reads and writes) Post-Increment register (such as TBLPTR with Table reads and writes) Post-Decrement register (such as TBLPTR with Table reads and writes) Pre-Increment register (such as TBLPTR with Table reads and writes)
n	The relative address (2's complement number) for relative branch instructions, or the direct address for Call/Branch and Return instructions
PRODH	Product of Multiply high byte
PRODL	Product of Multiply low byte
s	Fast Call/Return mode select bit. s = 0: do not update into/from shadow registers s = 1: certain registers loaded into/from shadow registers (Fast mode)
u	Unused or Unchanged
WREG	Working register (accumulator)
x	Don't care (0 or 1) The assembler will generate code with x = 0. It is the recommended form of use for compatibility with all Microchip software tools.
TBLPTR	21-bit Table Pointer (points to a Program Memory location)
TABLAT	8-bit Table Latch
TOS	Top-of-Stack
PC	Program Counter
PCL	Program Counter Low Byte
PCH	Program Counter High Byte
PCLATH	Program Counter High Byte Latch
PCLATU	Program Counter Upper Byte Latch
GIE	Global Interrupt Enable bit
WDT	Watchdog Timer
$\overline{TO}$	Time-out bit
$\overline{PD}$	Power-down bit
C, DC, Z, OV, N	ALU status bits Carry, Digit Carry, Zero, Overflow, Negative
[ ]	Optional
( )	Contents
→	Assigned to
< >	Register bit field
∈	In the set of
italics	User defined term (font is courier)

FIGURE 20-1: GENERAL FORMAT FOR INSTRUCTIONS

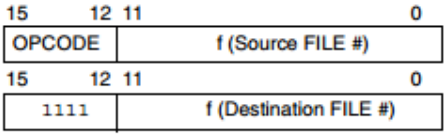
Byte-oriented file register operations — **Example Instruction**



ADDWF MYREG, W, B

d = 0 for result destination to be WREG register
d = 1 for result destination to be file register (f)
a = 0 to force Access Bank
a = 1 for BSR to select bank
f = 8-bit file register address

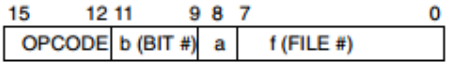
Byte to Byte move operations (2-word)



MOVFF MYREG1, MYREG2

f = 12-bit file register address

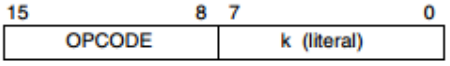
Bit-oriented file register operations



BSF MYREG, bit, B

b = 3-bit position of bit in file register (f)
a = 0 to force Access Bank
a = 1 for BSR to select bank
f = 8-bit file register address

Literal operations

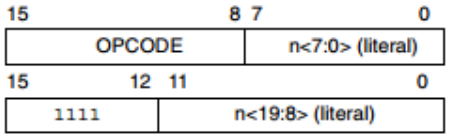


MOVLW 0x7F

k = 8-bit immediate value

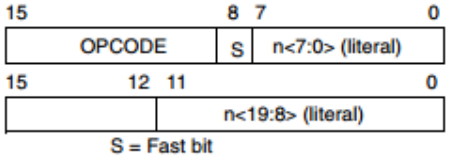
Control operations

CALL, GOTO and Branch operations

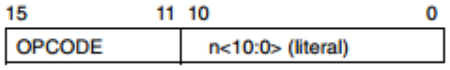


GOTO Label

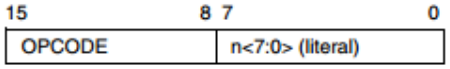
n = 20-bit immediate value



CALL MYFUNC



BRA MYFUNC



BC MYFUNC

TABLE 20-2: PIC18FXXX INSTRUCTION SET

Mnemonic, Operands		Description	Cycles	16-Bit Instruction Word MSb			LSb	Status Affected	Notes
BYTE-ORIENTED FILE REGISTER OPERATIONS									
ADDWF	f, d, a	Add WREG and f	1	0010	01da0	ffff	ffff	C, DC, Z, OV, N	1, 2
ADDWFC	f, d, a	Add WREG and Carry bit to f	1	0010	0da	ffff	ffff	C, DC, Z, OV, N	1, 2
ANDWF	f, d, a	AND WREG with f	1	0001	01da	ffff	ffff	Z, N	1,2
CLRF	f, a	Clear f	1	0110	101a	ffff	ffff	Z	2
COMF	f, d, a	Complement f	1	0001	11da	ffff	ffff	Z, N	1, 2
CPFSEQ	f, a	Compare f with WREG, skip =	1 (2 or 3)	0110	001a	ffff	ffff	None	4
CPFSGT	f, a	Compare f with WREG, skip >	1 (2 or 3)	0110	010a	ffff	ffff	None	4
CPFSLT	f, a	Compare f with WREG, skip <	1 (2 or 3)	0110	000a	ffff	ffff	None	1, 2
DECF	f, d, a	Decrement f	1	0000	01da	ffff	ffff	C, DC, Z, OV, N	1, 2, 3, 4
DECFSZ	f, d, a	Decrement f, Skip if 0	1 (2 or 3)	0010	11da	ffff	ffff	None	1, 2, 3, 4
DCFSNZ	f, d, a	Decrement f, Skip if Not 0	1 (2 or 3)	0100	11da	ffff	ffff	None	1, 2
INCF	f, d, a	Increment f	1	0010	10da	ffff	ffff	C, DC, Z, OV, N	1, 2, 3, 4
INCFSZ	f, d, a	Increment f, Skip if 0	1 (2 or 3)	0011	11da	ffff	ffff	None	4
INFSNZ	f, d, a	Increment f, Skip if Not 0	1 (2 or 3)	0100	10da	ffff	ffff	None	1, 2
IORWF	f, d, a	Inclusive OR WREG with f	1	0001	00da	ffff	ffff	Z, N	1, 2
MOVF	f, d, a	Move f	1	0101	00da	ffff	ffff	Z, N	1
MOVFF	f_s, f_d	Move f_s (source) to 1st word f_d (destination) 2nd word	2	1100 1111	ffff ffff	ffff ffff	ffff ffff	None	
MOVWF	f, a	Move WREG to f	1	0110	111a	ffff	ffff	None	
MULWF	f, a	Multiply WREG with f	1	0000	001a	ffff	ffff	None	
NEGF	f, a	Negate f	1	0110	110a	ffff	ffff	C, DC, Z, OV, N	1, 2
RLCF	f, d, a	Rotate Left f through Carry	1	0011	01da	ffff	ffff	C, Z, N	
RLNCF	f, d, a	Rotate Left f (No Carry)	1	0100	01da	ffff	ffff	Z, N	1, 2
RRCF	f, d, a	Rotate Right f through Carry	1	0011	00da	ffff	ffff	C, Z, N	
RRNCF	f, d, a	Rotate Right f (No Carry)	1	0100	00da	ffff	ffff	Z, N	
SETF	f, a	Set f	1	0110	100a	ffff	ffff	None	
SUBFWB	f, d, a	Subtract f from WREG with borrow	1	0101	01da	ffff	ffff	C, DC, Z, OV, N	1, 2
SUBWF	f, d, a	Subtract WREG from f	1	0101	11da	ffff	ffff	C, DC, Z, OV, N	
SUBWFB	f, d, a	Subtract WREG from f with borrow	1	0101	10da	ffff	ffff	C, DC, Z, OV, N	1, 2
SWAPF	f, d, a	Swap nibbles in f	1	0011	10da	ffff	ffff	None	4
TSTFSZ	f, a	Test f, skip if 0	1 (2 or 3)	0110	011a	ffff	ffff	None	1, 2
XORWF	f, d, a	Exclusive OR WREG with f	1	0001	10da	ffff	ffff	Z, N	
BIT-ORIENTED FILE REGISTER OPERATIONS									
BCF	f, b, a	Bit Clear f	1	1001	bbba	ffff	ffff	None	1, 2
BSF	f, b, a	Bit Set f	1	1000	bbba	ffff	ffff	None	1, 2
BTFSC	f, b, a	Bit Test f, Skip if Clear	1 (2 or 3)	1011	bbba	ffff	ffff	None	3, 4
BTFSS	f, b, a	Bit Test f, Skip if Set	1 (2 or 3)	1010	bbba	ffff	ffff	None	3, 4
BTG	f, d, a	Bit Toggle f	1	0111	bbba	ffff	ffff	None	1, 2

Note 1: When a PORT register is modified as a function of itself (e.g., `MOVF PORTB, 1, 0`), the value used will be that value present on the pins themselves. For example, if the data latch is '1' for a pin configured as input and is driven low by an external device, the data will be written back with a '0'.

2: If this instruction is executed on the TMR0 register (and, where applicable, d = 1), the prescaler will be cleared if assigned.

3: If Program Counter (PC) is modified or a conditional test is true, the instruction requires two cycles. The second cycle is executed as a `NOP`.

4: Some instructions are 2-word instructions. The second word of these instructions will be executed as a `NOP`, unless the first word of the instruction retrieves the information embedded in these 16-bits. This ensures that all program memory locations have a valid instruction.

5: If the Table Write starts the write cycle to internal memory, the write will continue until terminated.

TABLE 20-2: PIC18FXXX INSTRUCTION SET (CONTINUED)

Mnemonic, Operands		Description	Cycles	16-Bit Instruction Word MSb			LSb	Status Affected	Notes
CONTROL OPERATIONS									
BC	n	Branch if Carry	1 (2)	1110	0010	nnnn	nnnn	None	
BN	n	Branch if Negative	1 (2)	1110	0110	nnnn	nnnn	None	
BNC	n	Branch if Not Carry	1 (2)	1110	0011	nnnn	nnnn	None	
BNN	n	Branch if Not Negative	1 (2)	1110	0111	nnnn	nnnn	None	
BNOV	n	Branch if Not Overflow	1 (2)	1110	0101	nnnn	nnnn	None	
BNZ	n	Branch if Not Zero	2	1110	0001	nnnn	nnnn	None	
BOV	n	Branch if Overflow	1 (2)	1110	0100	nnnn	nnnn	None	
BRA	n	Branch Unconditionally	1 (2)	1101	0nnn	nnnn	nnnn	None	
BZ	n	Branch if Zero	1 (2)	1110	0000	nnnn	nnnn	None	
CALL	n, s	Call subroutine1st word	2	1110	110s	kkkk	kkkk	None	
		2nd word		1111	kkkk	kkkk	kkkk		
CLRWDT	—	Clear Watchdog Timer	1	0000	0000	0000	0100	$\overline{TO}$, $\overline{PD}$	
DAW	—	Decimal Adjust WREG	1	0000	0000	0000	0111	C	
GOTO	n	Go to address1st word	2	1110	1111	kkkk	kkkk	None	
		2nd word		1111	kkkk	kkkk	kkkk		
NOP	—	No Operation	1	0000	0000	0000	0000	None	
NOP	—	No Operation	1	1111	xxxx	xxxx	xxxx	None	4
POP	—	Pop top of return stack (TOS)	1	0000	0000	0000	0110	None	
PUSH	—	Push top of return stack (TOS)	1	0000	0000	0000	0101	None	
RCALL	n	Relative Call	2	1101	1nnn	nnnn	nnnn	None	
RESET		Software device RESET	1	0000	0000	1111	1111	All	
RETFIE	s	Return from interrupt enable	2	0000	0000	0001	000s	GIE/GIEH, PEIE/GIEL	
RETLW	k	Return with literal in WREG	2	0000	1100	kkkk	kkkk	None	
RETURN	s	Return from Subroutine	2	0000	0000	0001	001s	None	
SLEEP	—	Go into Standby mode	1	0000	0000	0000	0011	$\overline{TO}$, $\overline{PD}$	

Note 1: When a PORT register is modified as a function of itself (e.g., `MOVF PORTB, 1, 0`), the value used will be that value present on the pins themselves. For example, if the data latch is '1' for a pin configured as input and is driven low by an external device, the data will be written back with a '0'.

2: If this instruction is executed on the TMR0 register (and, where applicable, d = 1), the prescaler will be cleared if assigned.

3: If Program Counter (PC) is modified or a conditional test is true, the instruction requires two cycles. The second cycle is executed as a `NOP`.

4: Some instructions are 2-word instructions. The second word of these instructions will be executed as a `NOP`, unless the first word of the instruction retrieves the information embedded in these 16-bits. This ensures that all program memory locations have a valid instruction.

5: If the Table Write starts the write cycle to internal memory, the write will continue until terminated.

TABLE 20-2: PIC18FXXX INSTRUCTION SET (CONTINUED)

Mnemonic, Operands		Description	Cycles	16-Bit Instruction Word MSb			LSb	Status Affected	Notes
LITERAL OPERATIONS									
ADDLW	k	Add literal and WREG	1	0000	1111	kkkk	kkkk	C, DC, Z, OV, N	
ANDLW	k	AND literal with WREG	1	0000	1011	kkkk	kkkk	Z, N	
IORLW	k	Inclusive OR literal with WREG	1	0000	1001	kkkk	kkkk	Z, N	
LFSR	f, k	Move literal (12-bit) 2nd word to FSRx 1st word	2	1110 1111	1110 0000	00ff kkkk	kkkk kkkk	None	
MOVLB	k	Move literal to BSR<3:0>	1	0000	0001	0000	kkkk	None	
MOVLW	k	Move literal to WREG	1	0000	1110	kkkk	kkkk	None	
MULLW	k	Multiply literal with WREG	1	0000	1101	kkkk	kkkk	None	
RETLW	k	Return with literal in WREG	2	0000	1100	kkkk	kkkk	None	
SUBLW	k	Subtract WREG from literal	1	0000	1000	kkkk	kkkk	C, DC, Z, OV, N	
XORLW	k	Exclusive OR literal with WREG	1	0000	1010	kkkk	kkkk	Z, N	
DATA MEMORY ↔ PROGRAM MEMORY OPERATIONS									
TBLRD*		Table Read	2	0000	0000	0000	1000	None	
TBLRD*+		Table Read with post-increment		0000	0000	0000	1001	None	
TBLRD*-		Table Read with post-decrement		0000	0000	0000	1010	None	
TBLRD+*		Table Read with pre-increment		0000	0000	0000	1011	None	
TBLWT*		Table Write	2 (5)	0000	0000	0000	1100	None	
TBLWT*+		Table Write with post-increment		0000	0000	0000	1101	None	
TBLWT*-		Table Write with post-decrement		0000	0000	0000	1110	None	
TBLWT+*		Table Write with pre-increment		0000	0000	0000	1111	None	

Note 1: When a PORT register is modified as a function of itself (e.g., `MOVF PORTB, 1, 0`), the value used will be that value present on the pins themselves. For example, if the data latch is '1' for a pin configured as input and is driven low by an external device, the data will be written back with a '0'.

2: If this instruction is executed on the TMR0 register (and, where applicable, d = 1), the prescaler will be cleared if assigned.

3: If Program Counter (PC) is modified or a conditional test is true, the instruction requires two cycles. The second cycle is executed as a `NOP`.

4: Some instructions are 2-word instructions. The second word of these instructions will be executed as a `NOP`, unless the first word of the instruction retrieves the information embedded in these 16-bits. This ensures that all program memory locations have a valid instruction.

5: If the Table Write starts the write cycle to internal memory, the write will continue until terminated.

4.13 STATUS Register

The STATUS register, shown in Register 4-2, contains the arithmetic status of the ALU. The STATUS register can be the destination for any instruction, as with any other register. If the STATUS register is the destination for an instruction that affects the Z, DC, C, OV, or N bits, then the write to these five bits is disabled. These bits are set or cleared according to the device logic. Therefore, the result of an instruction with the STATUS register as destination may be different than intended.

For example, `CLRF STATUS` will clear the upper three bits and set the Z bit. This leaves the STATUS register as `000u uluu` (where `u` = unchanged).

It is recommended, therefore, that only `BCF`, `BSF`, `SWAPF`, `MOVFF` and `MOVWF` instructions are used to alter the STATUS register, because these instructions do not affect the `Z`, `C`, `DC`, `OV`, or `N` bits from the STATUS register. For other instructions not affecting any status bits, see Table 20-2.

Note: The C and DC bits operate as a $\overline{\text{borrow}}$ and $\overline{\text{digit borrow}}$ bit respectively, in subtraction.

REGISTER 4-2: STATUS REGISTER

U-0	U-0	U-0	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x
—	—	—	N	OV	Z	DC	C
bit 7							bit 0

bit 7-5 **Unimplemented:** Read as '0'

bit 4 **N:** Negative bit
This bit is used for signed arithmetic (2's complement). It indicates whether the result was negative (ALU MSB = 1).
1 = Result was negative
0 = Result was positive

bit 3 **OV:** Overflow bit
This bit is used for signed arithmetic (2's complement). It indicates an overflow of the 7-bit magnitude, which causes the sign bit (bit7) to change state.
1 = Overflow occurred for signed arithmetic (in this arithmetic operation)
0 = No overflow occurred

bit 2 **Z:** Zero bit
1 = The result of an arithmetic or logic operation is zero
0 = The result of an arithmetic or logic operation is not zero

bit 1 **DC:** Digit carry/$\overline{\text{borrow}}$ bit
For `ADDWF`, `ADDLW`, `SUBLW`, and `SUBWF` instructions
1 = A carry-out from the 4th low order bit of the result occurred
0 = No carry-out from the 4th low order bit of the result

Note: For $\overline{\text{borrow}}$, the polarity is reversed. A subtraction is executed by adding the two's complement of the second operand. For rotate (`RRF`, `RLF`) instructions, this bit is loaded with either the bit 4 or bit 3 of the source register.

bit 0 **C:** Carry/$\overline{\text{borrow}}$ bit
For `ADDWF`, `ADDLW`, `SUBLW`, and `SUBWF` instructions
1 = A carry-out from the Most Significant bit of the result occurred
0 = No carry-out from the Most Significant bit of the result occurred

Note: For $\overline{\text{borrow}}$, the polarity is reversed. A subtraction is executed by adding the two's complement of the second operand. For rotate (`RRF`, `RLF`) instructions, this bit is loaded with either the high or low order bit of the source register.

Legend:			
R = Readable bit	W = Writable bit	U = Unimplemented bit, read as '0'	
- n = Value at POR	'1' = Bit is set	'0' = Bit is cleared	x = Bit is unknown

9.0 I/O PORTS

Depending on the device selected, there are either five ports or three ports available. Some pins of the I/O ports are multiplexed with an alternate function from the peripheral features on the device. In general, when a peripheral is enabled, that pin may not be used as a general purpose I/O pin.

Each port has three registers for its operation. These registers are:

- TRIS register (data direction register)
- PORT register (reads the levels on the pins of the device)
- LAT register (output latch)

The data latch (LAT register) is useful for read-modify-write operations on the value that the I/O pins are driving.

Example program listings

```
;*********************************************************
;* test_1.asm
;*********************************************************
;* This program adds the values of A_1 and A_2), the result
;* will be added to register pair B (B_H és B_L).
;* B is a 16 bit variable, B_H is the upper byte, and
;* B_L is the lower byte of it.
;*********************************************************

    list p=18f452

    include 'p18f452.inc'        ; Processor-dependent declarations are included

A_1     equ 0x60                 ; Variables are in the general purpose memory
A_2     equ 0x61
B_H     equ 0x70
B_L     equ 0x71


    org     0x0000               ; Reset vector, the processor starts here
Rst_vect
    goto    Start

                                 ; If we have interrupts, the service routines
will be here.
                                 ; Now this section is empty.

    org     0x0040               ; Start of the main program
Start
    movf    A_1, 0               ; Load A_1 to the work register (W). '0' is for
loading A_1 to W,
                                 ; not to itself (default)
    addwf   A_2, 0               ; Add A_2 to W, the result will be in W
    btfsc   STATUS, C            ; Checking the Carry flag,
    incf    B_H, 1               ; if there was Carry, we implement B_H,
    addwf   B_L, 1               ; then add W (W=A_1 + A_2) to B_L
    btfsc   STATUS, C            ; Checking the Carry flag,
    incf    B_H, 1               ; if there was Carry, we implement B_H
    goto    Start                ; Restart

    end
```

```
;*********************************************************
;* test_2.asm
;*********************************************************
;* This program shows the basic masking, using AND, OR and XOR
;* instructions
;*********************************************************

    list p=18f452

    include 'p18f452.inc'       ; Processor-dependent declarations are included

A_1     equ 0x60                ; Variables are in the general purpose memory
A_2     equ 0x61


    org     0x0000              ; Reset vector, the processor starts here
Rst_vect
    goto    Start

                                ; If we have interrupts, the service routines
will be here.
                                ; Now this section is empty.

    org     0x0040              ; Start of the main program
Start
    movlw   0xAA                ; Load 0xAA
    andlw   0xF0                ; Set to 0 the low nibble

    movlw   0xAA                ; Load 0xAA
    iorlw   0xF0                ; Set to 1 the high nibble

    movlw   0xAA                ; Load 0xAA
    xorlw   0xF0                ; Invert all bits of the high nibble

Circle
    goto    Circle              ; Infinite loop

    end
```

```
;*********************************************************
;* test_3.asm
;*********************************************************
;* This program waits to high level on RA0, then starts a bit
;* stepping on PORTB to left. If PORTB.7 is set, the bit stepping
;* will be reversed.
;*********************************************************

    list p=18f452

    include 'p18f452.inc'       ; Processor-dependent declarations are included


    org     0x0000              ; Reset vector, the processor starts here
Rst_vect
    goto    Start

                                ; If we have interrupts, the service routines
will be here.
                                ; Now this section is empty.

    org     0x0040              ; Start of the main program
Start

    bcf     ADCON1, PCFG3       ; Configuring RA0 to digitál (default is analog)
    bsf     ADCON1, PCFG2
    bsf     ADCON1, PCFG1
```

```
    clrf    LATB                ; All bits of LATB are 0
    movlw   0x00                ; And all PORTB bits are output
    movwf   TRISB               ; so TRISB set to all 0s.
    bsf     STATUS, C           ; Set CARRY flag to 1, so at the first stepping
this value
                                ; will be on the LATB0

Check_RA0
    btfss   PORTA, 0            ; Checking RA0
    goto    Check_RA0           ; if it is 0, waiting to 1
Rotate_L                        ; If 1, starts the stepping
    rlcf    LATB, 1             ; Rotating LATB to left (through CARRY)
    btfss   LATB, 7             ; while the 7. bit set
    goto    Rotate_L            ; but if it isn't set, rotate left again
Rotate_R
    rrncf   LATB, 1             ; Rotating left
    btfss   LATB, 0             ; while the LSB set
    bra     Rotate_R            ; If it isn't set, goes to rotating left (again)
    goto    Check_RA0           ; 1 cycle is done, check RA0 again...

    end
```

```
;*************************************************************
;* test_4.asm
;*************************************************************
;* This program uses TIMER1 interrupt, and inverts PORTB0 in a TIME*10ms.
;* Program doesn't work if TIME=0! The PicDem2 Plus demo panel has a clock
;* frequency of 4MHz, so the simulation frequency is the same.
;*************************************************************

    list p=18f452

    include 'p18f452.inc'   ; Processor-dependent declarations are included

TIME        equ 0x60        ; Variables are in the general purpose memory
TIME_WORK   equ 0x70

    org 0x0000
Rst_vect                    ; Reset vector, the processor starts here
    goto    Start

    org 0x0008
Int_vect                    ; Interrupt vector. If there is at least one enabled
interrupt, and
                            ; it's interrupt flag is set to 1, the program
execution
                            ; continues at this point
                            ; Checking if interrupt source is the TMR1 overflow:
    btfss   PIE1, TMR1IE    ; if TMR1IE and TMR1IF are 1, the there is an TIMER1
IT
    reset                   ; or else it is a false interrupt calling (because no
oter enabled IT
    btfss   PIR1, TMR1IF
    reset
    bra     TMR1_ISR        ; Jumping to TIMER1 interrupt service routine

    org 0x0040
Start                       ; Start of the main program
                            ; Setting PORTB
    bcf     LATB, 0         ; clearing LATB0
;   bcf     TRISB, 0        ; Simplest way to set RB0 as output
                            ; but we are now using masking
    movlw   0xFE            ; clearing TRISB0, all other bits are unchanged
    andwf   TRISB, 1        ; So RB0 is an output
```

```
                            ; Setting Timer1
    bsf     T1CON, TMR1ON   ; TIMER1 enable
    movlw   0xD8            ; Load TIMER1 with the value ensuring the 10 ms
overflow:
    movwf   TMR1H           ; (65536-TMR1Preload)*(1/1MHz)=10 ms, so:
    movlw   0xF0            ; TMR1Preload = 65536 - 10ms*1MHz = 55536 = 0xD8F0
    movwf   TMR1L           ; Frequency is 1 MHz in the functions above, because
peripheral
                            ; clock is 1 MHz, if the clock source has the
frequency of 4 MHz
                            ; Setting interrupts
    bcf     PIR1, TMR1IF    ; clearing TIMER1 overflow flag
    bsf     PIE1, TMR1IE    ; enabling TIMER1 overflow
    bsf     INTCON, PEIE    ; enabling pheripheral interrupts
    bsf     INTCON, GIE     ; enabling globális interrupts

    movff   TIME, TIME_WORK ; copying TIME to TIME_WORK, so TIME remains
unchanged

Stop                        ; infinite loop
    goto    Stop

    org     0x0200
TMR1_ISR                    ; Timer1 interrupt service routine
    movlw   0xD8            ; loading the value calculated above to TMR1
    movwf   TMR1H
    movlw   0xF0
    movwf   TMR1L
    decfsz  TIME_WORK, 1
    goto    No_activity
    btg     LATB, 0         ; inverting RB0
    movff   TIME, TIME_WORK ; Restoring TIME_WORK from TIME
No_activity
    bcf PIR1, TMR1IF        ; Timer1 interrupt is serviced, so clearing TMR1IF
    retfie                  ; and returning from interrupt (always by 'retfie'
command!)

    end
```

```
;*********************************************************
;* test_5.asm
;*********************************************************
;* This program receives a byte from the serial port
;* then sends this byte+1 back (incremented) back. Settings are is 9600, 8, N, 1
;* but at simulation it is not important...
;*********************************************************

    list p=18f452

    include 'p18f452.inc'   ; Processor-dependent declarations are included

    org 0x0000
Rst_vect                    ; Reset vector, the processor starts here
    goto    Start

    org 0x0008
Int_vect                    ; Interrupt vector.

    btfss   PIE1, RCIE      ; USART IT enabled??
    goto    Other_Int       ; No.
    btfss   PIR1, RCIF      ; Is it an USART IT?
    goto    Other_Int       ; No.
                            ; This is an USART IT.
    movlw   06h             ; Checking the receiver errors
    andwf   RCSTA, W        ; Masking OERR and FERR
```

```
    btfss   STATUS, Z
    goto    Rcv_Error       ; There is OERR or FERR

    movf    RCREG, W        ; Reading received data
    incf    WREG            ; Incrementing data
    movwf   TXREG           ; Loading incremented data to TXREG
    goto    ISR_End         ; End of interrupt service routine.

Rcv_Error
    bcf     RCSTA, CREN     ; If there was OERR or FERR, the USART must be
    bsf     RCSTA, CREN     ; switched off and on
    goto    ISR_End         ; End of interrupt service routine.

Other_Int
    goto    Other_Int       ; There is no other interrupt source, so it is an
error,
                            ; the program remains in an infinite loop.
ISR_End
    retfie                  ; Returning from interrupt

Start                       ; Initializing the used peripherals
    org 0x0050

    bcf     TRISC, 6        ; RC6/TX pin is output

    bsf     TXSTA, BRGH     ; high bit rate
    movlw   .25             ; Loading SPBRG value (9600 bps, clock is 4 MHz)
    movwf   SPBRG           ; From data sheet: 9600 bps = 4 MHz/(16(SPBRG+1)), so
SPBRG = 25

    bsf     RCSTA, SPEN     ; Enabling rerial port
    bsf     RCSTA, CREN     ; Enabling continuous receiving
    bsf     TXSTA, TXEN     ; Enabling transmission

    bcf     PIR1, RCIF      ; Clearing receiver interrupt
    bsf     PIE1, RCIE      ; Enabling receiver interrupt
    bsf     INTCON, PEIE    ; Enabling peripheral interrupt
    bsf     INTCON, GIE     ; Enabling global interrupt

Stop                        ; Infinite  loop, waiting to interrupt
    goto    Stop

    end
```

The USART stimulus for “test5.asm’

```
wait 20 ms
30 31 32

wait 10 ms
‘HELLO! ‘

rand 10 50 ms
‘345’
```

Questions

1. What is a microcontroller? What is the difference between microprocessor and microcontroller? (If you do not have enough knowledge, search Internet, e.g. keyword: 'microprocessor vs microcontroller'!)
2. How the microcontroller system software development is performed?
3. What are the basic tools for microcontroller system development?
4. What is the main difference between simulator and emulator?
5. What are debugging tools: breakpoint, single stepping and stimulus?

Useful links:

http://ww1.microchip.com/downloads/en/DeviceDoc/51519a.pdf

http://ww1.microchip.com/downloads/en/DeviceDoc/39564c.pdf