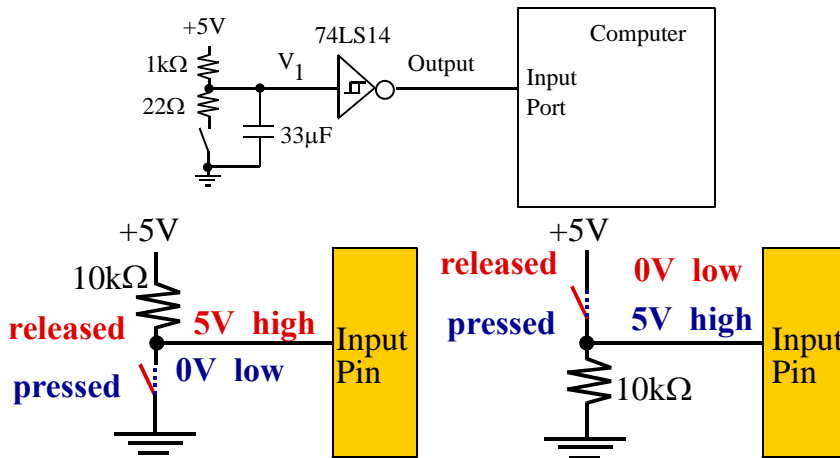
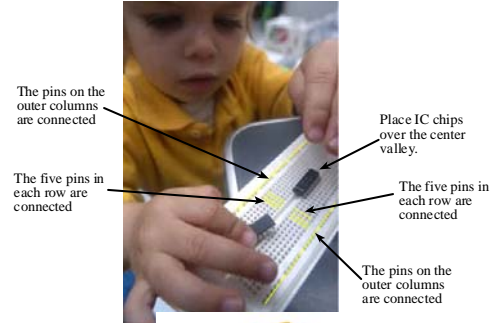


Recap

Addition and subtraction set CCR bits
 Subtraction used for conditional branching

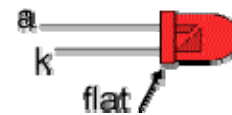
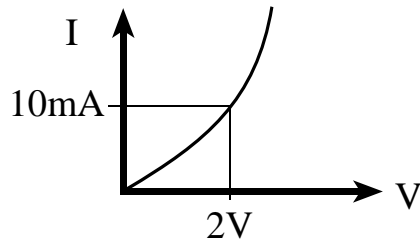
Overview

Switch interfacing
 LED interfacing
 Introduction to Lab 2
 Running on the real 9S12 board
 if-then statements reviewed

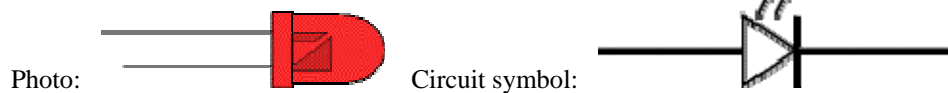


Show actual switch interface and voltages with the DVM

LED interfacing

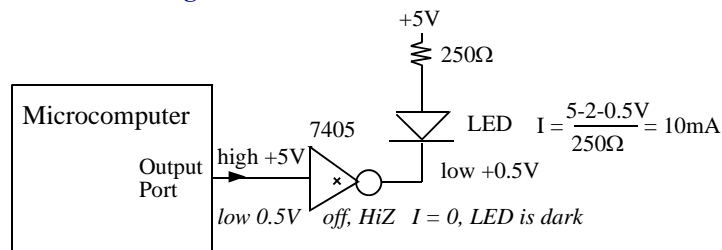


LED brightness = power = V*I



LEDs emit light when an electric current passes through them. LEDs have polarity, meaning current must pass from anode to cathode to activate. The anode is labeled **a** or **+**, and cathode is labeled **k** or **-**. The cathode is the short lead and there may be a slight flat spot on the body of round LEDs. Thus, the anode is the longer lead

LED interfacing



Ohm's Law through the resistor $V = I \cdot R$

$$R = \frac{5\text{ V} - (\text{LED V}) - (\text{output low voltage of the 7405})}{\text{desired LED current}}$$

$$= \frac{(5 - 2 - 0.5\text{V})}{0.01\text{ A}} = 250\Omega$$

solid state relay**optical sensors****fiberoptic cable**

Show actual LED interface and voltages with the DVM

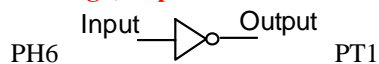
Show S12C32.htm<http://users.ece.utexas.edu/~valvano/S12C32.htm>**Hardware Setup**

Cut paper with pin names

Gently place 9S12DP512 system into protoboard

Move switch on 9S12DP512 in LOAD mode

Cable from PC to docking module

In TExaSNeed **microcomputer** and **program** filesNeed an **IO** file to first simulate system**Redesign, implement and test the NOT gate**

- 1) Design- data flow graph, flowchart, pseudocode
- 2) Implement in TExaS, debug it
- 3) Build the hardware as needed, check it, then check it again
- 4) Power applied to embedded system, reset button
- 5) Switch to Real 9S12 mode and debug it again

In Real 9S12 mode, show

Reset, Assemble (also downloads), Look at registers, Look at global variables

Single step, Change memory address, Run, Halt, Reset, Breakpoint

- 6) place 9S12DP512 in RUN mode

Power applied to embedded system, reset button

It's running at 8 MHz

Redesign making it a friendly NOT gate**Branch operations (review)**

```

bcc place      ;go if C=0
bcs place      ;go if C=1
beq place      ;go if Z=1
bne place      ;go if Z=0
bmi place      ;go if N=1
bpl place      ;go if N=0
bvc place      ;go if V=0
bvs place      ;go if V=1
bra place      ;go always
brn place      ;go never
jmp place      ;go always, ext addr

```

> < ≥ conditional branch instructions must follow a subtract compare or test instruction, such as

```

suba subb sbca sbcb subd
cba cmpa cmpb cpd cpx cpy
tsta tstb tst

```

signed branches, branch if

bge *place* **greater than or equal to**
 if $(N \wedge V) = 0$
 i.e., $(\sim N \cdot V + N \cdot \sim V) = 0$
bgt *place* **greater than**
 if $(Z + N \wedge V) = 0$
 i.e., $(Z + \sim N \cdot V + N \cdot \sim V) = 0$
ble *place* **less than or equal to**
 if $(Z + N \wedge V) = 1$
 i.e., $(Z + \sim N \cdot V + N \cdot \sim V) = 1$
blt *place* **less than**
 if $(N \wedge V) = 1$
 i.e., $(\sim N \cdot V + N \cdot \sim V) = 1$

unsigned branches, branch if

bhs *place* **greater than or equal to**
 if $C = 0$, same as **bcc**
bhi *place* **greater than**
 if $C + Z = 0$
blo *place* **less than**
 if $C = 1$, same as **bcs**
bls *place* **less than or equal to**
 if $C + Z = 1$

it is important to know

- precision (e.g., 8-bit, 16-bit)
- format (e.g., unsigned, signed)

It takes three steps

1. read the first value into a register
2. compare the first value with the second value
3. conditional branch

When testing for equal or not equal

- doesn't matter whether signed or unsigned
- still matters if 8-bit or 16-bit
- doesn't matter about load and compare order

The bottom line

Think of current and voltage when interfacing
Switches bounce when touched and when released
LEDs are fast, but our eyes are slow
Double-check wiring before turning on power
Is it 8-bit or 16-bit data?
Are they signed or unsigned numbers?