ECE331 Microcomputers (KEH) May 15, 2008
Test #2 – 100 Points Maximum (2 periods Max)
Open Textbook and Microcontroller Interfacing Topics Notes
Dept. of Electrical and Computer Engineering
Rose-Hulman Institute of Technology

Name: Solution **CM Box:**_____

1) **(28 Points, 2 points per blank)**
Assembly Language Program: Interrupt-driven White Noise Generator
The hardwired circuit shown below in Figure P1 is a 31-bit right-shift register consisting of D flip-flops FF0 – FF30, whose input is formed by EXCLUSIVE OR-ing the outputs of FF2 and FF30. This sequence generator produces a "maximum length" pseudorandom binary sequence (PRBS) that will not repeat until 2^{31}-1 clock pulses have elapsed. The system output may be taken from the output of any flip-flop in the shift register. The (normally closed) PRESET pushbutton is used to start the shift register in the state of all 1's, since the state of all 0's is the one state that is *not* allowed in a maximal length pseudorandom binary sequence generator, and so we must not let this circuit start in the all 0's state. When clocked at 20 kHz, the sequence will take (2^{31}-1)/20000/60/60 = 29.8 hours to repeat itself! Thus the binary output is a rather random sequence of 0's and 1's! If this circuit drives a loudspeaker, it will produce white noise that might be used as a sleep aid.

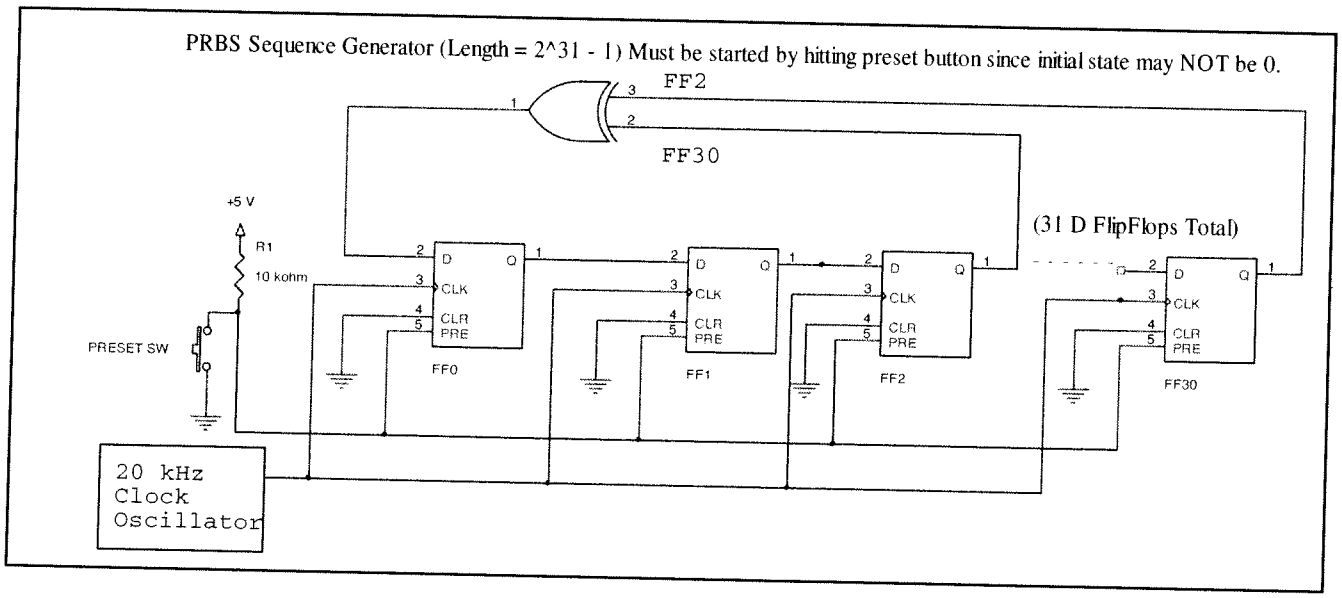


Figure P1. Pseudorandom Binary Sequence Generator
Below is an assembly-language program written for a Freescale 9S12C32 (specifically for our CSM12C32 module) that emulates this hardwired white noise generator in software as an interrupt routine. The calling program sets up the **RTI interrupt** to interrupt at a 15.625 kHz rate (instead of the 20 kHz in the hardware circuit above), and it also enables interrupts before it falls into an idle loop. The interrupt routine implements the rest of the system shown in Fig. P1. *Note that this software emulation should behave exactly like the hardware system in Fig. P1, except it is clocked at a 15.625 kHz instead of 20 kHz.*

Please note that this program is intended to run on our CSM12C32 lab modules, and that these modules employ a 16 MHz ceramic resonator, which sets the OSCCLK rate to 16 MHz. Note from Fig. 6.15 in

the Huang Text that OSCCLK has nothing to do with the PLL that forms the bus clock, thus the RTI interrupt rate does ***NOT*** depend upon the bus clock rate, as set by the PLL.

Each RTI interrupt corresponds to a single clock pulse in the hardwired circuit of Fig. P1. Note that four byte-sized variables (SHR3, SHR2, SHR1, and SHR0) are used to implement the 31-bit shift register, where SHR3 represents FF0 – FF7; SHR2 represents FF8 – FF15; etc. The system output is taken from FF30, and is driven out on I/O pin PM0. If a loudspeaker is connected to PM0, we will hear the broadband white noise as a steady "hiss".

Fill in the missing blanks in this assembly-language program.

```
; ECE331 White Noise
; PRBS.ASM - Generates 2^31-1 bit long PRBS (pseudorandom binary
;            sequence with a 15.625 kHz clock rate). Uses RTI interrupt.
;
            XDEF WHITENOISE
            ABSENTRY WHITENOISE
            INCLUDE 'mc9s12c32.inc'
            ORG $800
SHR3:       ds.b 1
SHR2:       ds.b 1
SHR1:       ds.b 1
SHR0:       ds.b 1
TEMP:       ds.b 1
            ORG $4000
WHITENOISE: lds #$1000
            bset DDRM,1
            bclr PTM,1

            ;Next two lines simulate depression of PRESET SW in Fig. P1
            movw #$FFFF,SHR3  ;***BLANK #1 ****
            movw #$FFFF,SHR1  ;***BLANK #2 ****

            ;Divide 16MHZ OSCCLK to get
            ;RTI interrupts at 15.625 kHz rate (Text Table 6.4 and Text Fig. 6.12)
            movb #$10,RTICTL  ;***BLANK #3 ****

            bset CRGINT,#$80  ;***BLANK #4 ****
            movb #$80,CRGFLG  ;Clear RTI interrupt flag
            CLI               ;***BLANK #5 ****
loop_here_forever:
                              bra   loop_here_forever
;*********Here ends the main program "WHITENOISE"
WHITENOISEISR:
            CLR TEMP
            BRCLR SHR3,%00100000,FF2NOTSET
            MOVB #1,TEMP
FF2NOTSET:  CLRA
            BRCLR SHR0,#1,FF30NOTSET ;***BLANK #6 ****
            LDAA #1
FF30NOTSET: EORA TEMP                ;***BLANK #7 ****
            RORA (or LSRA, ASRA)     ;***BLANK #8 ****
            ROR SHR3
            ROR SHR2
            ROR SHR1
            ROR SHR0
```

See Table 6.4 & Fig 6.16

See Fig. 6.11 and Fig 6.11

Rotate result in LSB of A into carry Flag!

```
ldaa  SHRØ                         ;***BLANK #9 ****
  RORA  (LSRA, ASRA)               ;***BLANK #10 ****
STAA PTM ;Send Bit #30 out to PM0

;Relax the RTI interrupt flag
movb  #$80, CRGFLG                 ;***BLANK #11 ****
  RTI                              ;***BLANK #12 ****
;************************************************************
;*    Initialize Reset Vector and TC0 Interrupt Vector      *
;************************************************************
  ORG $FFFE
  dc.w     WHITENOISE        ;Make reset vector point to
                             ;entry point of WHITENOISE program
  ORG  # FFFØ                      ;***BLANK #13 ****
  dc.w  WHITENOISEISR              ;***BLANK #14 ****
```

(Rotate Bit 30 into posn)

(RTI) Erroneous Comment, my error !!

From Text, Table 6.1 (p. 228) (Look up "interrupts" in index)

2) ***LCD Multiplexing (14 points)***

a. (1 pts) A custom LCD display for a new product has 127 segments that must be individually controlled (turned on or off). If we choose to use ¼ multiplexing on this display, implying 4 back plane signals are needed, what is the total number of wires (back plane wires *plus* front plane wires) that must be connected to this display?

$\# FP's = \frac{127}{4} = 31\frac{3}{4} = 32$

$\# BP's = 4$ $\qquad 32+4 = \boxed{36}$

Total # Wires = 36

b. (1 pt) Repeat Part A for 1/3 multiplexing. Total # Wires = 46

$\# FP's = \frac{127}{3} = 42\frac{1}{3} = 43 \qquad \# BP's = 3$

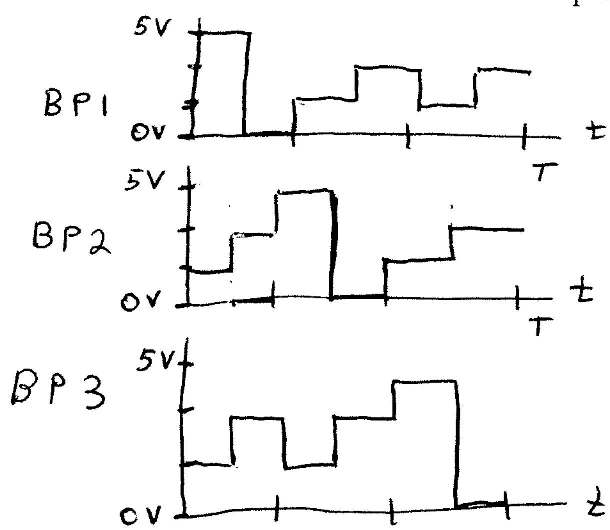
c. (1 pt) Repeat Part A for 1/10 multiplexing. Total # Wires = 23

$\# FP's = \frac{127}{10} = 12.7 = 13 \qquad \# BP's = 10$

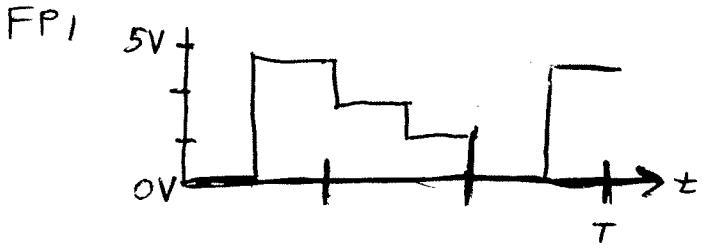
$13 + 10 = \boxed{23}$

Note that 1/10 multiplexing is Not really appropriate for This number of segments, as The # of BP's required makes total # wires quite large !!

d. (2 pts) For the case of 1/3 LCD multiplexing, there are 3 back plane signals, BP1, BP2, BP3. Assume that Vcc = 5 V, so the waveform voltage levels are 5 V, 3.333 V, 1.666 V, and 0 V. Sketch one frame of each of the three backplane signals.

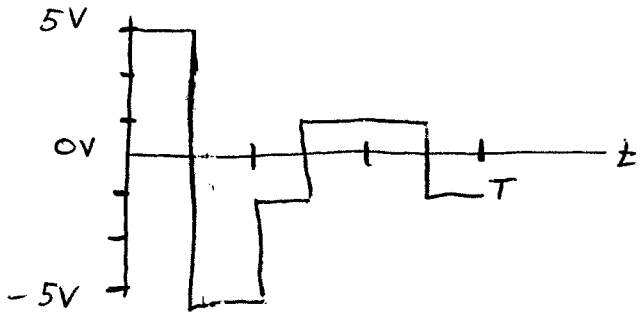


e. (2 pts) Sketch one frame of a single front plane signal, FP1, where the segments that pass over BP1, BP3 are to be ON, and the segment that passes over BP2 is to be OFF.



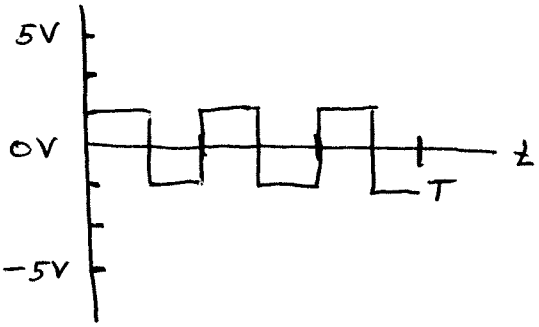
f. (2 pts) Sketch one frame of the voltage waveform $Vseg_{11}$, which represents the voltage across the "turned on" segment that lies between BP1 and FP1. ($Vseg_{11}$ = BP1 voltage – FP1 voltage). Use the FP1 voltage waveform from Part e above.

$V_{seg_{11}} = BP1 - FP1$



g. (2 pts) Sketch one frame of the voltage across the "turned off" segment that lies between BP2 and FP1, $Vseg_{21}$. ($Vseg_{21}$ = BP2 voltage – FP1 voltage). Use the FP1 voltage waveform from Part e above

$V_{seg_{21}} = BP_2 - FP_1$



h. (2 pts) For the case of 1/3 LCD multiplexing, find the RMS value of the $Vseg_{11}$ waveform, which corresponds to the waveform of a turned **ON** segment, and also the RMS value of the $Vseg_{21}$ voltage waveform, which corresponds to a turned **OFF** segment. *Hint: Recall that in the class notes, it was shown (in Figure 7.21) that for the case of ¼ multiplexing, the RMS voltage across a segment that is ON is Vrmson = 2.899 V,rms; and the RMS voltage across a segment that is OFF is Vrmsoff = 1.67 V, rms. Show your calculations in the space below.*

$$(V_{SEG11})_{RMS} = \sqrt{\frac{1}{T}\left[5^2\left(\frac{T}{3}\right) + \left(\frac{5}{3}\right)^2\left(\frac{2T}{3}\right)\right]} = 3.191\ V_{,rms}$$

$$(V_{SEG21})_{RMS} = \sqrt{\frac{1}{T}\left[\left(\frac{5}{3}\right)^2 T\right]} = 1.667\ V_{,rms}$$

RMS value of $Vseg_{11}$ = 3.191 V,rms RMS value of $Vseg_{21}$ = 1.667 V,rms

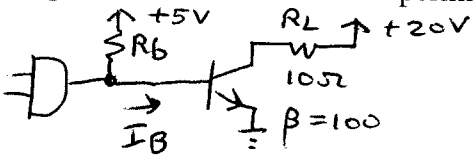
F. (1 pt) Based upon comparing the results for ¼ and 1/3 multiplexing,

(a) which multiplexing method requires fewer connections? 1/4

(b) which multiplexing method yields higher contrast? 1/3 ← Since 3.191V > 2.899V

3) (5 pts) A power NPN BJT with beta = 100 is used to switch a 10 ohm, 20 V resistive load using the upper left circuit of Slide #57. (Assume Vbe(on) = 0.7 V and Vce(sat) = 0 V.)

(a) (4 pts) What is the maximum permissible value of Rb?



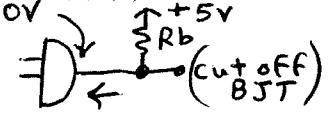
$$20 - \left(\frac{5-0.7}{R_B}\right)(100)\cdot 10\Omega = 0V$$

(I_B, β, R_L, $V_{CE(sat)}$)

$$\Rightarrow R_B = \boxed{215\Omega}$$

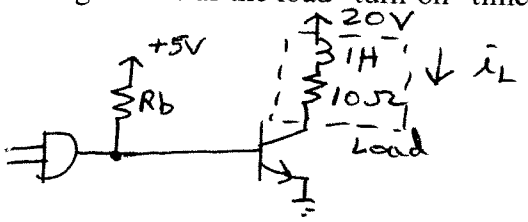
$(V_{CE})_{SAT}$ =0V

(b) (1) How much current must the open-collector driving gate be able to sink? (Assume Vcesat = 0V.)



$$I_{sink} = \frac{5V - 0V}{215\Omega} = \boxed{23.26\ mA}$$

4) (4 points) Imagine that the 10 ohm resistive load of Problem 3 is replaced by an inductive load that may be modeled as a 1.0 H inductance and a 10 ohm resistance. How long would it take (assuming that the open-collector driving gate output voltage has been LOW for a long time, and then it suddenly rises HIGH. How long after that will the load current reach 90% of its final value (1.8 Amperes)? You might regard this as the load "turn-on" time.



$I_i = 0\ A$

$i_L = I_F - (I_F - I_i)\, e^{-t/\tau}$

$I_F = \frac{20V}{10\Omega} = 2A$

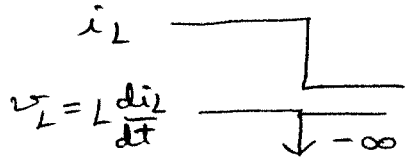
$\tau = L/R = \frac{1H}{10\Omega} = 0.1\ s$

$i_L = 2(1 - e^{-10t})$

$2(1 - e^{-10t_x}) = 0.9(2A) = 1.8A \Rightarrow t_x =$ **0.230 sec**

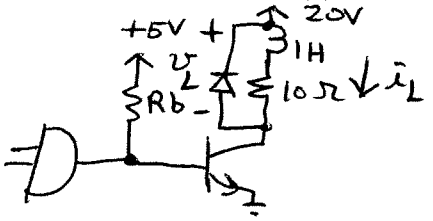
5) (12 pts) Imagine that the driving gate output voltage of the circuit in Problem 4 is suddenly changed from HIGH to LOW.

a) (2) What serious problem will occur? (Explain using $v_L = L di_L/dt$)



∞ Negative voltage spike adds in series with power supply and **Burns out BJT!**

b) (2) Redraw the circuit showing how a ***single*** fast-acting diode may be added to this circuit to solve the problem of Part 5(a).



When BJT ON, diode OFF ($v_L > 0$:)

Now when BJT turns off, $v_L < 0 \Rightarrow$ diode **ON**

$\Rightarrow i_L$ circulates through diode until current dissipates gradually.

c) (4) For the circuit of 5(b), determine how long it will take for the load current to decay from its full value down to 10% of this value (0.2 A). You might regard this as the "load turn off time".

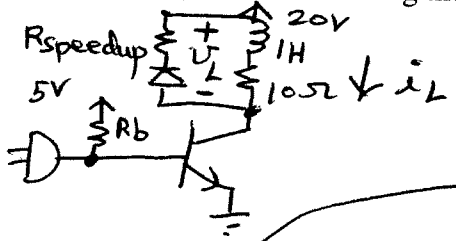
$I_i = 2A,\ I_F = 0A$

$\Rightarrow i_L = 2e^{-10t}$

$2e^{-10t_x} = 0.2 \Rightarrow t_x =$ **0.230 sec**

d) (2) How could you make this turn off time faster? Redraw the circuit showing how one additional resistor "Rspeedup" might be added, so the load will turns off faster, without affecting the load current of the load turn-on time.

Now $\tau = \frac{L}{R_L + R_{speedup}}$



e) (2) What is the problem if you make Rspeedup too large? As usual, there is an engineering tradeoff.

Initial current through Rspeedup is 2A → BJT voltage vs. Turn off speed

If Rspeedup is too large, $v_L(0) = -2A\,(R_{speedup}) =$ Very Large (−) Value

⇒ BJT may still be in danger of Blowing Out!

(8 pts) Using only rising-edge sensitive D flip-flops (with D, CLK, CLR, Q and Q\ pins) and assorted inverters and logic gates, design a circuit that will derive the ***2x resolution*** CW output waveform shown in Fig. 6 from the A and B input waveforms. (See arrow below) YOU NEED NOT DERIVE THE CCW output waveform. ***Be sure to label you're A and B inputs and your CW output.***

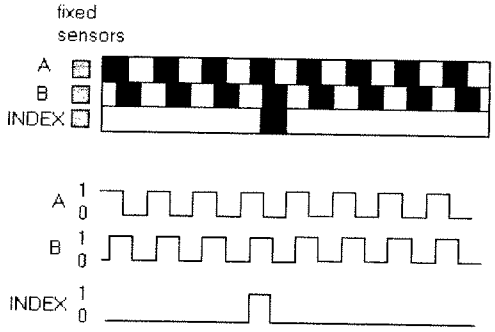


Fig 5. Incremental encoder disk track patterns

→This one

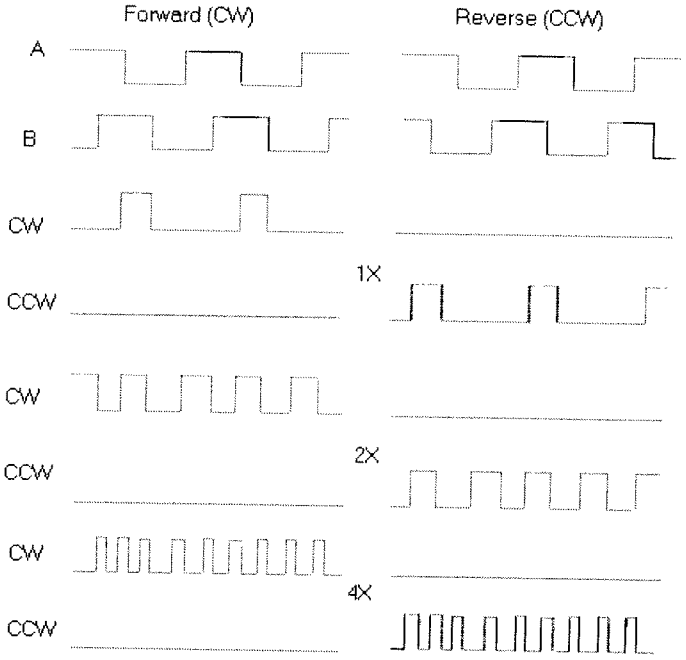
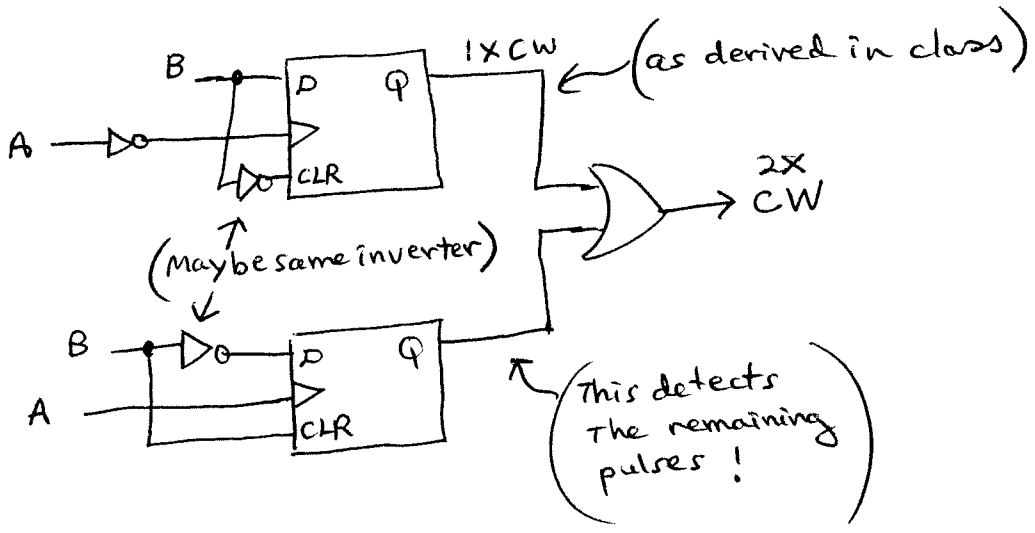


Fig 6. Quadrature direction sensing and resolution enhancement. (CW = clockwise, CCW= counter-clockwise)



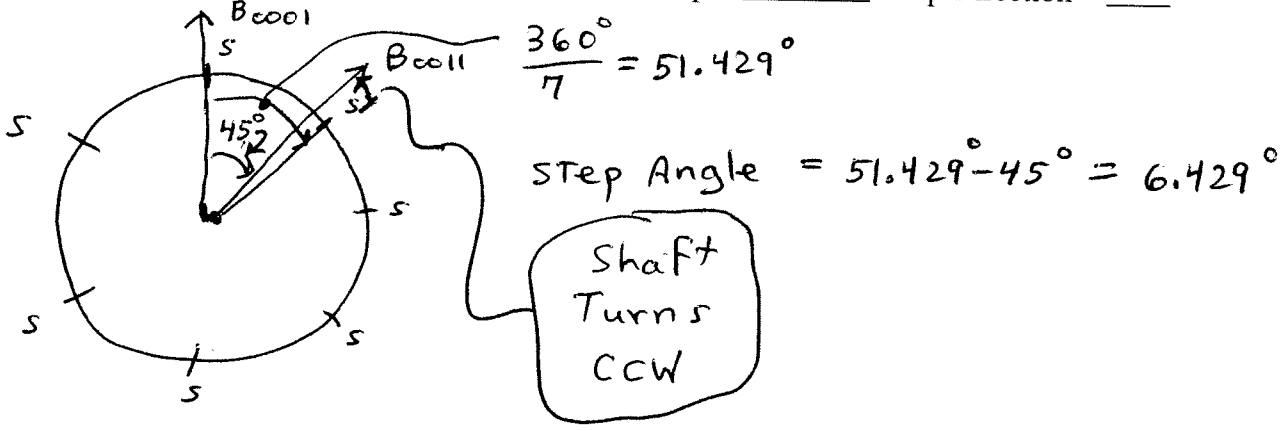
6) ***Stepping Motor (8 points)***
Referring to the stepping motor circuit diagram shown in the course notes (Slide #68), imagine that the two bottom rows of 7407/7406 inverters are removed, leaving us with just one row of 2N6427 power Darlington transistors. Then imagine that a microcontroller has PTM3 connected to the base of the left-most power Darlington, PTM2 to the next one, PTM1 to the next, and finally PTM0 to the right-most power Darlington.

a. (2 pts) List the sequence of eight 4-bit numbers that would have to be output on the low 4 bits of PORT M (in the order PTM3:PTM2:PTM1:PTM0) in order to make the magnetic field vector developed by the stepping motor step in the clockwise (CW) direction, with 8 steps per revolution (45 degrees per step). Let your first number correspond to the magnetic field pointing directly up. (Hint: you may turn on either 1 or 2 coils at a time.)

0001, 0011, 0010, 0110, 0100, 1100, 1000, 1001

b. (4 pts) Assuming a permanent magnet rotor with 7 permanent magnet poles (instead of the rotor with 3 permanent magnet poles considered on Slide #69 in the lecture notes), determine the number of steps per revolution of the shaft using the 8-value sequence of Part A. Do this by drawing, in the space provided below, the 7-pole rotor (showing only the south poles) with one of the 7 poles aligned with the initial B field. Then when the B field steps 45 degrees to its next position, determine which south pole is closest to the new position of the B field, and hence is pulled into alignment. Determine the angle through which the shaft rotates, and determine its direction (CW or CCW).

Degrees of Shaft Angle Rotation Per Step = 6.429° Step Direction = CCW



$\frac{360^\circ}{7} = 51.429^\circ$

Step Angle = $51.429^\circ - 45^\circ = 6.429^\circ$

c. (1 pt) What is the best name for the four 1N4001 power diodes in this stepping motor circuit? (circle one)
(1.) transient voltage suppression diodes 2. turn-on speedup diodes 3. turn-off speedup diodes
4. load current limiter diodes

d. (1 pt) What is the best name for the purpose of the 22-ohm resistor in this stepping motor circuit? (circle one)
1. turn-on speedup resistor (2.) turn-off speedup resistor 3. load current limiter
4. voltage transient suppression resistor

7) (2 pt) A magnetic reed switch will be most sensitive to an applied magnetic field (**B**) that is oriented in a direction that is

1. perpendicular to the reeds (2.) parallel to the reeds 3. at a 45 degree angle to the reeds

8) (2 pt) What is the purpose of the diodes in the 8 x 8 scanned keyswitch matrix discussed in the course notes?

(1.) short-circuit protection 2. over-voltage protection 3. speed up key scanning process

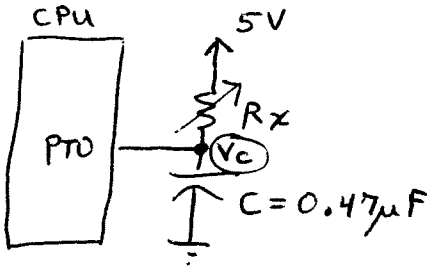
9) (7 pts) Imagine that a "poor man's A/D" circuit implemented in the C language is used to sense the value of a variable resistor Rx by connecting Rx between PT0 and Vcc = 5.0 V and a 0.47 µF capacitor between PT0 and ground. Assume that PT0 has a logic high threshold of 3.00 V. If PT0 is driven low (to 0 V) for several seconds, and then suddenly released (allowed to float), the time elapsed before a logic 1 is read on PT0 is 3.5 ms.

A. (4 pts) Find the value of Rx.

$$V_c = 5\left(1 - e^{-\frac{t}{(0.47\mu F \cdot Rx)}}\right)$$

$$3.0V = 5\left(1 - e^{-\frac{3.5ms}{0.47\mu F \cdot Rx}}\right)$$

$$\Rightarrow R_x = 8.127\ k\Omega$$



B. (1 pts) How should the LSB of the PERT register be set in order to obtain the most accurate measurement of Rx? Explain your reasoning.

0 (disable internal pullup, since we do NOT want any other resistance in parallel with R_x)

C. (1 pts) How would you set the LSB's of the Port T data register and the Port T data direction register in order to drive PT0 to 0 V?

Port T_data_0 = 0 (set data bit to "0")

Port T_ddr_0 = 1 (drive data "0" out on PTØ pin)

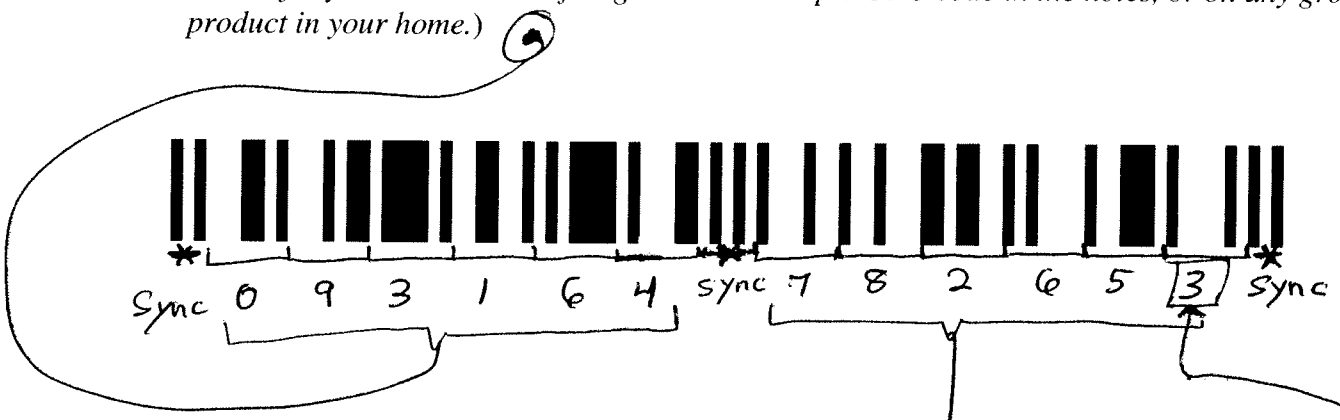
D. (1 pts) How would you set the LSB's of the Port T data register and the PORT T data direction register in order to release (float) PT0?

Port T_data_0 = x (don't care)

Port T_ddr_0 = 0 (float PT0)

10) (10 pts) **UPC Bar Code** (Used on groceries and many other consumer products, but NOT on books!)

a. (3 pts) Using the UPC encoding table found in the notes, determine the six encoded UPC digits in the left half of the bar code. Recall that Black = 1, White = 0; there are 3 SYNC patterns: 101 at each end, and 01010 in the middle. (*Hint: first make sure you can successfully decode the six left digits in the example UPC code in the notes, or on any grocery product in your home.*)



b. (3 pts) Recalling that the UPC encoding table found in the notes must have its ***black and white regions exchanged*** for the right half of the UPC code, determine the six encoded UPC digits in the right half of the bar code. (*Hint: first make sure you can successfully decode the six right digits in the example UPC code in the notes, or on any grocery product in your home.*)

c. (4 pts) The last (rightmost) digit you found in Part (b) is the UPC-A checksum digit. In the space below, show the step-by-step calculation of this checksum digit from the other preceding 11 digits. Your results must match the 12th digit you decoded above.

Checksum = $10 - [3(0+3+6+7+2+5) + (9+1+4+8+6)]$ % 10

$= 10 - 97 \,\%\, 10 = 10 - 7 = 3$

✓