Massachusetts Institute of Technology
Department of Electrical Engineering and Computer Science
6.115 Microprocessor Project Laboratory Spring 2024

Laboratory 2
Timing and Controlling Events in the Physical World

Issued: February 27, 2024 Due: March 12, 2024

GOALS: Understand and modify the monitor code running on your R31JP board
Understand the structure of an Intel Hex file
Learn to use interrupts and timers
Connect a peripheral chip to your R31JP
Take control of a physical system (LED lamp) using your R31JP
Learn about the PSoC "Programmable System on Chip" and the C programming language

PRIOR to starting this lab:
Read: Scherz "Operational Amplifiers," Intel 8254 data sheet (skim), anything you didn't finish reading from Lab 1.
Note that pages 2-21 – 2-24 in the MCS-51 Microcontroller User's manual tell you how many machine cycles per instruction. PSoC background (see the 6.115 website).

ON-LINE CODE SUBMISSION: Please **electronically submit** your **final** ASCII text assembly language (.asm) file of the lamp code you write in Exercise 7 to the course website by 1pm on March 12, 2024.

EXERCISE 1: Check out more MINMON monitor code capability on your R31JP board

Currently, you have hopefully used your MINMON "D" and "G" commands.

In this exercise, we'll add "read and write" capabilities to MINMON. Use the currently free "R" and "W" command letters. These commands will be **very** useful for the remainder of the term, especially when testing new chips in your 6.115 kit.
Please do the following:

- Add a "read" command to MINMON. At the MINMON prompt, you should be able to type a command that reads the hexadecimal contents of a byte in external memory on the R31JP board. This hex byte should be printed on the PC screen. A typical interaction might look like this in a Hyperterminal window:

```
* R9000
FF
```

In this example, we've chosen to read the contents of memory location 9000. The R31JP board has read the contents of this memory location (FF in this example) and printed it on the PC terminal screen.

- Add a "write" command to MINMON. This command should load a hex byte into a specified location in external memory. A typical interaction might look like this in a Hyperterminal window:

```
* W9000=FF
*
```

In this example, we're loading the memory at location 9000 with the byte FF (hex).

- You can test your new MINMON commands by first using the existing MINMON on your R31JP board to download your new test monitor code to RAM, and then executing the test code in RAM by

flipping the MON/RUN switch. Test your code carefully. Read and write to several locations in memory to make sure you understand the behavior of your code. Burn your finished monitor code into your code EEPROM using the 6.115 chip programmer you saw during the laboratory/kit familiarization lecture. You will not be able to use W to write to every location in external memory. Be prepared to explain why at your check-off.

EXERCISE 2: Reverse assemble an Intel hex file

Please examine the Intel Hex file shown in the text box below. This file was produced by AS31 from a text file containing an 8051 assembly language program.

```
:10000000740085E090D29280F75061727479206F0D
:0B0010006E20696E20362E3131352144
:00000001FF
```

Please do the following:

- "Reverse assemble" the program. That is, read the hex file, above, and write down the assembly language file that, assembled with AS31, would produce the file above.
- Assemble your assembly language file using AS31 and make sure that it produces the hex file in the text box.
- Explain what the program does.
- The assembly language file contained a secret message. If you are the **first** person (who has not already won a gift card) to correctly **e-mail** the message to Professor Leeb, **exactly**, then claim a $10 Amazon gift card from Professor Leeb.

EXERCISE 3: A simple calculator

Let's take your new MINMON for a test drive. Please do the following:

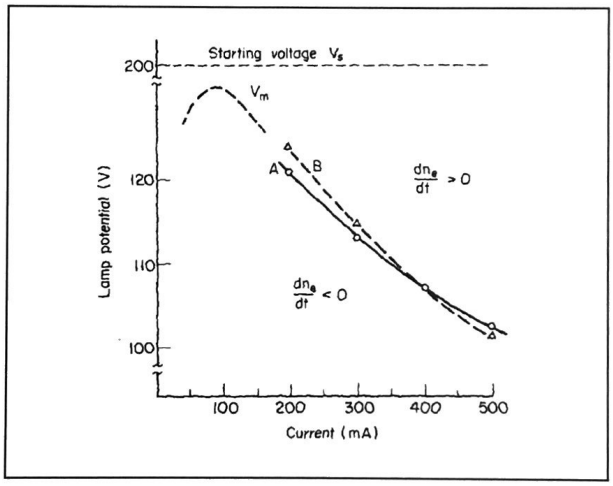
- Write a program that reads two bytes from sequential locations in external memory, e.g., 9000h and 9001h. The program should compute the sum, difference, product, and the quotient of the two numbers. Store the results, i.e., sum, difference, 16-bit product, and quotient and remainder in sequential external memory locations without overwriting the original two bytes. Make the program easy to use repeatedly. It should be executed using the MINMON "G" command, **not** by toggling the MON/RUN switch. That is, the R31JP board should stay in MON mode (red LED lit) at all times. You should enter the operands at 9000h and 9001h using your new "W" command. Complete the calculations using the "G" command, and read results from memory using your "R" command.

EXERCISE 4: Get comfortable with the fluorescent lamp

A key goal of 6.115 is to become familiar with the general technique of using an embedded controller to run a physical system. In this laboratory you'll use your R31JP board to control the operation of a fluorescent lamp. A fluorescent lamp is a sealed glass tube with electrodes at either end. During the manufacturing process, the tube is coated with a material called a phosphor, the white powder you may

have noticed on the walls of a broken tube. The tube is evacuated of air, and refilled with mercury gas at a low pressure. In operation, an electrical arc or discharge fills the tube in response to a voltage difference between the electrodes. The arc excites the mercury gas, creating charged plasma that emits ultraviolet light, which stimulates the phosphors and makes them glow.

A fluorescent lamp is very different from an incandescent lamp. A fluorescent lamp bulb cannot simply be "plugged in" to the wall. It requires a special circuit called a "ballast". The reason that the lamp cannot simply be "plugged in" to a voltage source is indicated by the V-I characteristic for a typical fluorescent lamp, shown in the figure below (from "Electric Discharge Lamps" by Waymouth). When the bulb is "cold," i.e., off and not producing any light, it exhibits high impedance between its electrodes. A high voltage must be applied across the lamp to initiate an arc. This is the "Starting voltage" shown in the figure. Once the arc fires or "strikes", the gas in the tube begins to ionize. Once the gas ionizes, the



impedance of the lamp is relatively low compared to the cold tube. This is indicated by the "warm" or "struck" or "lit" V-I characteristics in the figure, which show the bulb performance for two different tube temperatures (40 Celsius for curve A, 23 Celsius for curve B). These curves show loci of equilibrium V-I points once the bulb has struck. If the starting voltage is maintained across the tube, an enormous current would eventually flow into the tube until something broke, e.g., a fuse blew, the filaments in the lamp broke, a wire melted, etc. So, once the bulb has "struck", the terminal voltage must be reduced to a point on the equilibirium curves that will produce the desired current and illumination.

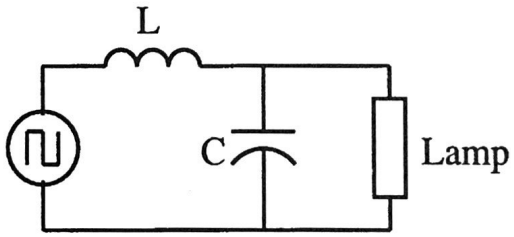
Reaching a stable equilibrium point on the struck V-I curve is tricky. If you examine the curves in the figure carefully, you'll notice that, for useful current ranges where illumination is produced (between 100 and 500 mA) an **increase** in terminal voltage corresponds to a **decrease** in terminal current, and vice-versa. Roughly speaking, this happens because, as the current decreases in the tube, the number of charged carriers in the tube also decreases, decreasing the conductivity of the plasma column in the tube. So a higher voltage is needed to maintain the lower current! Increasing the current, on the other hand, increases the conductivity of the plasma. A lower voltage is required to sustain the higher current.

This odd behavior makes it difficult to keep the lamp stable on the V-I equilibrium curve. Imagine, for example, that we plug the lamp into a fixed voltage source and that, somehow, the lamp is struck and a perfect voltage is applied that will keep the lamp on the equilibrium curve. Now, imagine a slight, inevitable disturbance that momentarily increases the current in the bulb. This disturbance could be a slight change in exterior temperature, for example. The voltage across the tube remains fixed, but now we are "off" the equilibrium curve, with a larger number of charge carriers in the tube compared to before

the disturbance. Off the equilibrium curve, this voltage will push yet more current into the bulb, further increasing the conductivity. If the voltage remains unchanged, the bulb enters a "runaway" condition where the current increases until something breaks.

A ballast circuit is used to start the lamp and keep it at a stable equilibrium point once struck. One easy way to solve the stability problem would be to put the lamp in series with a conventional linear resistor. For a suitable value of resistance, this series combination would appear to have an overall positive linear resistance characteristic. We could then apply a high voltage to strike the lamp/resistor combination, and lower the voltage to run at a stable equilibrium point in steady state. The problem with this plan is that the resistor dissipates power. This creates heat in the lamp fixture and also decreases the efficiency of the lamp.

A practical ballast circuit might look something like this:



In this ballast, a square wave produced by the input voltage source drives a resonant circuit that consists of an inductor, a capacitor, and the lamp. When the lamp is cold and off, we can approximate it as an open circuit. In this case, the square wave source is driving an LC resonant circuit. If the square wave is at the resonant frequency, and the parasitic resistances in the inductor and capacitor are small, a huge voltage will be generated across the inductor and capacitor. The capacitor voltage will strike the lamp. Once the lamp is struck, it has a relatively low impedance. The capacitor is in parallel with this low impedance, and the resonant quality (Q) of the circuit is reduced dramatically. Roughly, you can imagine that the lamp "shorts out" the capacitor, leaving the circuit as an inductor in series with the lamp. The inductor serves as the "ballast" impedance that limits the current through the lamp. The square wave voltage in the circuit above must be large enough to drive the steady state current (between 100 and 500 mA) into the inductor.

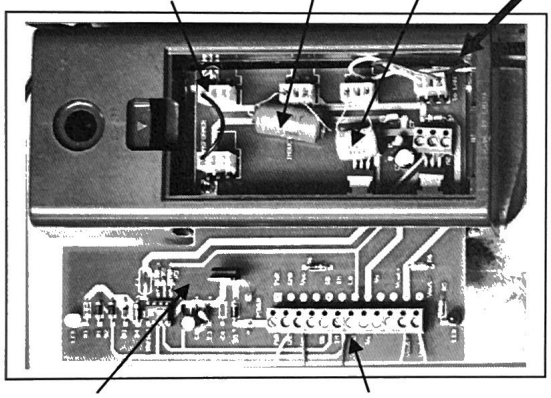
An AC voltage is used for at least two reasons. First, AC permits "lossless" reactive elements to serve as ballast impedance. Specifically, an inductor can be used instead of a resistor to limit the current through the lamp. If we used a DC input voltage, the inductor would have essentially zero impedance, and would not serve to limit current. Second, the bulb lasts longer if the two filaments occasionally swap roles as electrode emitter and receiver (cathode and anode).

If the resonant circuit has a "high Q" (low-loss) and is driven near its resonant frequency, the capacitor voltage will be large enough to strike the lamp. This presents an interesting engineering economics/manufacturing problem when making a lamp ballast. To make sure that the lamp will strike in a range of customer operating environments, the LC tank should be tightly tuned with low loss. This will create a strong "peak" in capacitor voltage at the resonant frequency. Unfortunately, this approach also means that the frequency of the square wave must be tightly controlled to be very near the resonant frequency. Therefore, either the L and C values must be tightly controlled and unchanging, or the square

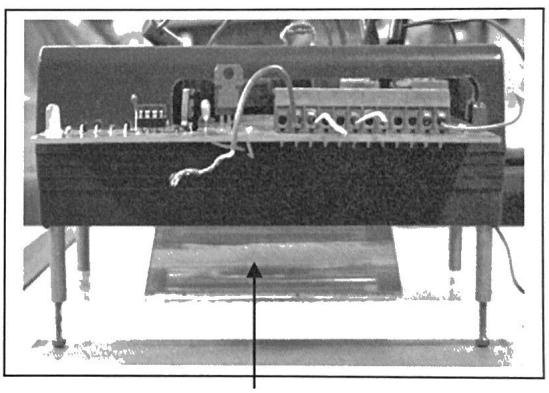
wave frequency must adapt to changes or tolerances in the L and C values. Precision power-level inductors and capacitors are expensive, and incompatible with the production of cheap ballasts.

In this laboratory, we will explore the problem of making a lamp ballast that will strike reliably in the presence of poorly controlled values of L and C. We'll begin this exercise by examining a low power fluorescent lamp and ballast. Physically, your fluorescent lamp fixture will look like the unit in the pictures shown below. These fixtures have been hand-built by the 6.115 staff. ***They are to be treated like fine china!*** Please make sure that you have had one of the staff familiarize you with the operation of the lamp before you use it, so that you do not injure yourself or destroy it. ***High voltages, in the hundreds-of-volts range, can be generated by this lab station.*** You must understand its operation fully before you begin to design and build with it.

Top View:



Front View:



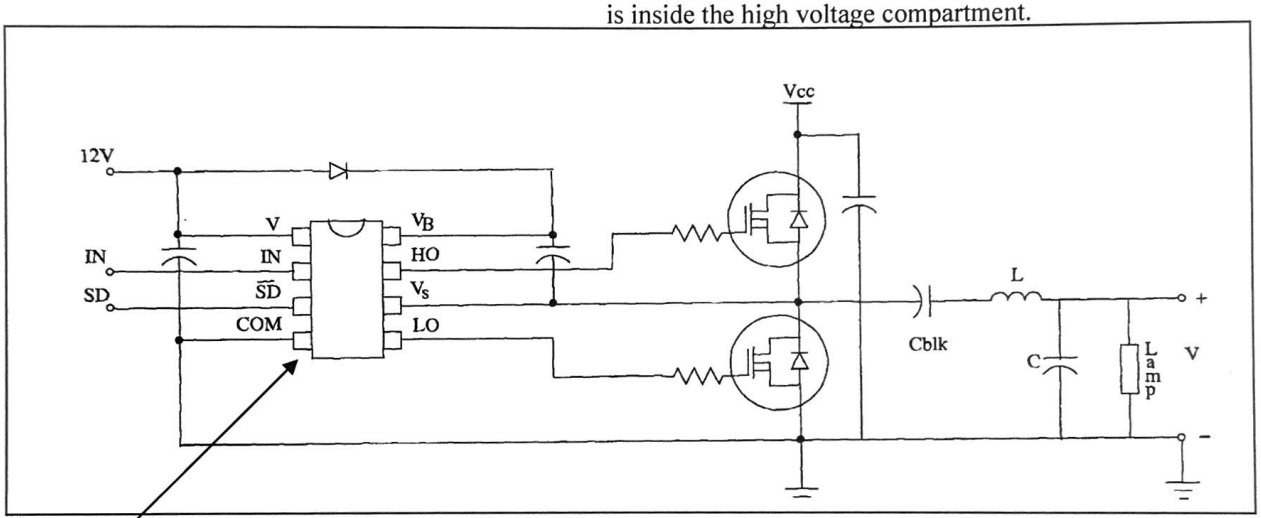
Your fixture should include a ballast circuit board, plastic case, stand with a mirror, and a wall transformer ("DC adaptor") that plugs into the side of the plastic case. Any components inside the high voltage compartment in the top of the plastic case (where you will plug in an inductor, capacitor, and jumper) can be at lethal high voltage when the setup is connected to power. Do **NOT touch** any of the components in the high voltage compartment when any power is applied to the station. Remove power before changing components in the high voltage compartment. If you think that a component needs to be changed or checked, find a staff member to help you. You will also need a variable laboratory power supply that can create an output between 0 and 50 volts and a TTL (0-5 volt) signal generator.

The signals at the signal connector (outside the high voltage compartment) are relatively safe, but also should not be touched when power is applied to the setup. You will make all of your connections to the signal connector.

You should never reach into the high voltage compartment. Dress your wires carefully so that nothing falls into the compartment.

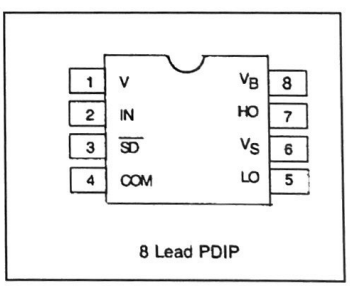
Ok, so what's in the box? Here's a simplified circuit diagram of your lamp ballast:

Everything on this half of the page, under this text, is inside the high voltage compartment.



This chip (above this text) is the IR2104 from International Rectifier. It is located outside of the high voltage compartment on your setup.

You can find out more about the IR2104 by examining its spec sheet on the International Rectifier web site, www.irf.com. The pinout of the IR2104 is:

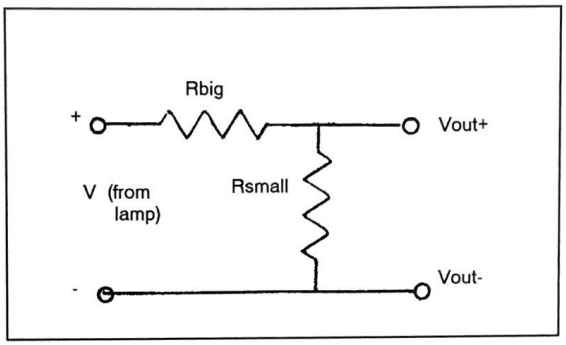


There are a few details in the box that are missing from this circuit diagram. For example, the 12 volts to the IR2104 chip actually comes from the DC adaptor (that little black cube you plug into the wall) and through a three terminal regulator. Also, there are some "bells and whistles" not shown on the circuit diagram. There are two power-indicator LED's on the circuit board. The red LED lights up when Vcc voltage is supplied. The yellow LED lights up when +12 V (from the DC adaptor) is supplied to the IR2104. These circuit details will be helpful in the lab, but will not be relevant for your understanding of how the ballast works to operate the fluorescent lamp, so they have been removed for clarity.

The signal connector on the ballast circuit board connects to many points in this circuit. We will be interested in 5 of these connections. The first is GND. **The GND connection on the signal connector must be connected to the ground on your lab kit, lab power supply, and anything else you may want to connect to the lamp fixture.** The second is Vcc. The Vcc tie point should be connected to the positive output of your variable laboratory power supply. This is the voltage at the top of the MOSFET totem pole shown in the circuit diagram. The third is IN. The IN tie point connects to the pin labeled IN on the IR2104 chip. This tie point should be connected to the signal output of a signal generator. The signal generator must be configured to generate a TTL square wave, that is, a square wave oscillating between 0 and 5 volts. Make certain that the square wave is oscillating between the proper levels

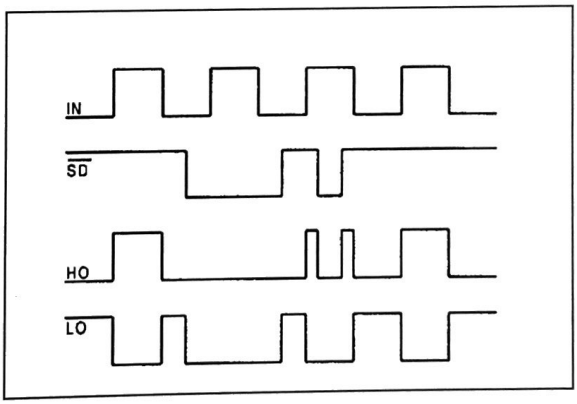
BEFORE you connect the signal generator to the lamp fixture. Improper levels, for example, negative voltage or too large a square wave, could destroy the lamp fixture.

On the ballast circuit board, the lamp voltage V passes through a resistor divider as shown below:



The resistor Rbig is 300 Kohms and Rsmall is approximately 30 Kohms. The output voltage Vout will be about a tenth of the input voltage, V. The signals Vout+ and Vout- are the fourth and fifth tie points of interest on the signal connector. The point Vout- is essentially identical to the tie point GND.

The key to understanding the ballast circuit is to understand how the IR2104 works. A timing diagram for the IR2104 is shown below:



On the 6.115 ballast circuit board, the shutdown pin, #SD on the IR2104, is wired high, i.e., the chip is always enabled. When the IN signal is high, the HO signal is also high, activating the top MOSFET in the ballast circuit. This drives the L-C-Lamp circuit with a high voltage, Vcc. When the IN signal is low, the bottom MOSFET in the ballast circuit is activated, grounding the input of the L-C-Lamp circuit. In short, the entire circuit can be thought of as a square wave voltage source oscillating between levels of zero volts and Vcc volts. The capacitor Cblk has a large value compared to C. The capacitor Cblk looks like a very low impedance at typical resonant frequencies for the ballast circuit (20-40 kHz). However, the DC component of the square wave produced by the IR2104 and MOSFETs appears across Cblk. The remainder of the resonant circuit, i.e., the L-C-Lamp combination, is effectively driven by a square wave of voltage oscillating between Vcc/2 and –Vcc/2. You control the frequency of this oscillation by varying the frequency of the TTL square wave that you apply to the IN pin on the signal connector.

Please do and answer the following:

- Connect your lamp fixture to the necessary lab equipment. First, set the lab power supply to zero volts and turn it off. Second, connect your DC adaptor to the lamp box, and plug the DC adaptor into the wall. Then, connect in this order, the grounds of your scope, lab kit, and lab supply to GND. Next, check your TTL square wave source with a scope to make sure that it does not go below 0 volts or above +5 volts, and then connect the TTL square wave source to IN. Connect the lab power supply to Vcc. Finally, connect a scope probe tip to Vout+. The resonant frequency of your fixture should be somewhere between 20 and 40 kHz. Set the signal generator above this range, at 60 kHz. Now, turn on the lab supply and set it to a low voltage for Vcc, approximately 3 volts. You can check the square wave produced by the two MOSFETs by putting a scope probe on the Vs pin on the signal connector. The pin Vs on the signal connector is connected to the drain of the lower MOSFET, and also the source of the upper MOSFET. At 3 volts, Vcc will be too low to provide enough excitation to strike the bulb. So, for this first experiment, the bulb should remain off at all times. Slowly lower the frequency of the square wave driving IN, noting the amplitude and shape of the voltage waveform between Vout+ and Vout-. Explain what you see. What is the shape of the voltage? Why? What is the amplitude of the voltage? At what frequency does the amplitude peak? How is this frequency related to the values of L and C in your circuit? What happens if you take the frequency below 20 kHz? (Don't go below 10 kHz.)
- Now, set the signal generator so that you are driving IN with a square wave frequency at the resonant frequency of your fixture. Increase the lab power supply voltage so that Vcc rises from 3 volts to 40 volts. You should see the lamp flicker briefly ("glow discharge"), and then strike and glow brightly. If the lamp does not glow after 5 or 10 seconds, turn off the lab power supply and check your setup with a staff member! Get comfortable with the difference between glow discharge (a dimly lit, flickering bulb) and clean ignition and steady state light output.
- Leave everything as is, but turn-off the lab power supply, leaving it set to 40 volts. After a few seconds, turn the lab power supply on again, suddenly applying 40 volts to Vcc, noting the behavior of the voltage waveform between Vout+ and Vout-. Explain what you see in terms of the behavior of the bulb and its light output.

EXERCISE 5: Make some square waves

Ultimately, we want the R31JP board to find the resonant frequency of the lamp ballast automatically. Because the L and C components in the fixture may be unknown to you or have fairly large tolerances in value, you will not be able to simply program the R31JP to produce a square wave of a precise, needed frequency. The R31JP will need to produce a range of square wave frequencies, selecting the correct frequency at which to operate the lamp. For this exercise, we'll examine different ways to make square waves with the microcontroller. Use a Port 1 pin as your square wave output. You should not remove your 74C922 and keypad, you'll need it later. Just be sure to deactivate the 74C922 (put it in tristate mode) so that the chip is not attempting to drive a Port 1 pin that you are simultaneously attempting to drive with the microcontroller.

Please do and answer the following:

- Write a program that reads a byte from external memory. Count down from this number to zero. Every time you reach zero, toggle the state of the Port 1 pin, then begin counting from the number to zero again. In this way, you should be able to make a program that causes the microcontroller to provide a variable frequency square wave output. You should not have to recompile the program to change the frequency, i.e., the frequency should be controllable from MINMON. What is the highest

frequency you can achieve? The lowest? What frequency resolution can you achieve, i.e, how finely can you adjust the frequency? Could you strike your lamp with the frequencies you make this way?

- Repeat this exercise, but this time use a two-byte count instead of a single byte. You will be graded for elegance, i.e., use the smallest amount of code and register space you can to do this. How does the 16-bit count effect your highest achievable frequency? Lowest? Resolution? Could you strike your lamp with this code? Again, make changes possible using MINMON so that the code does not have to be reassembled to change frequencies.
- Write another program that generates variable frequency square waves. However, instead of using a count down loop, this time use the microcontroller's internal timer 0 configured in Mode 2, the auto re-load mode. Use a software timer interrupt to change the state of the Port 1 pin. Keep your code, particularly the interrupt service routine, as lean as possible to ensure fast execution. You will need to use the MON/RUN switch on your R31JP board to execute this program because interrupts are involved; that is, you cannot use the MINMON "G" command to run the program. Explain why. Nevertheless, if you are clever, you can make the frequency controllable by writing to a memory location using the MINMON "W" command before flipping the MON/RUN switch to RUN. This will avoid excessive compiling. What is the fastest frequency you can achieve? Slowest? Resolution? Could you strike your lamp with this code?
- For the three pieces of code you just wrote (byte count, word count, and interrupt driven count), carefully analyze the timing of your code (number of machine cycles per instruction) and explain your observed highest and lowest frequencies and frequency resolution in terms of your code analysis.

EXERCISE 6: Understand the 8254 Counter/Timer chip

In this exercise we'll hook up a "peripheral" to the R31JP, the 8254 counter/timer chip. This chip will let us make square waves with very fine frequency control. Technically, you have already hooked up a peripheral to your R31JP, the 74C922 keypad controller we used in Lab 1. However, we connected the 74C922 to Port 1 with an "I/O mapped" connection scheme that uses a dedicated processor port. The 8052 microcontroller does not provide convenient support for simultaneously wiring lots of I/O mapped peripherals directly to the microcontroller through ports. So, in this exercise, we will explore a "memory mapped" peripheral connection. The 8254 will be addressed in the same manner as a memory location in RAM.

The 8254 is one chip in Intel's "8000-series" peripherals. The 8000 family of peripherals provides many different functions, and we will take advantage of some of them this term. For example, there are 8000-series chips that provide 8-bit parallel output ports like Port 1 on your R31JP (the 8255), serial ports, extra memory, or combinations of these. The 8254 provides a bank of three counter/timers that are similar in function to the two timers on the 8052 microcontroller in your R31JP board. Read the 8254 specification sheet. The 8254 is typical of a design philosophy that reappears throughout the 8000-peripherals. If you master the hook-up and programming of the 8254, you'll find using the rest of the 8000-series relatively straightforward.

The 8254 contains four internal byte-wide registers that can be individually selected through two address lines on the chip, labeled A0 and A1 in the spec sheet. The 8254 can be configured to provide a number of different counting and timing operations. You can select the 8254 configuration by programming an 8-bit internal command register. The other three registers in the 8254 provide count or timing information for each of the three counter/timers in the 8254. Wire your 8254 into the memory address "hole" that can be addressed using the XIOSELECT line coming from the R31JP. Connect address lines A0 and A1 from the R31JP board to the A0 and A1 lines on the 8254. Use the XIOSELECT line from the Xilinx PLD on the R31JP to drive the chip select line on the 8254. Select appropriate lines from the R31JP to connect to the RD and WR pins on the 8254. For this laboratory, you will use Mode 3 of the

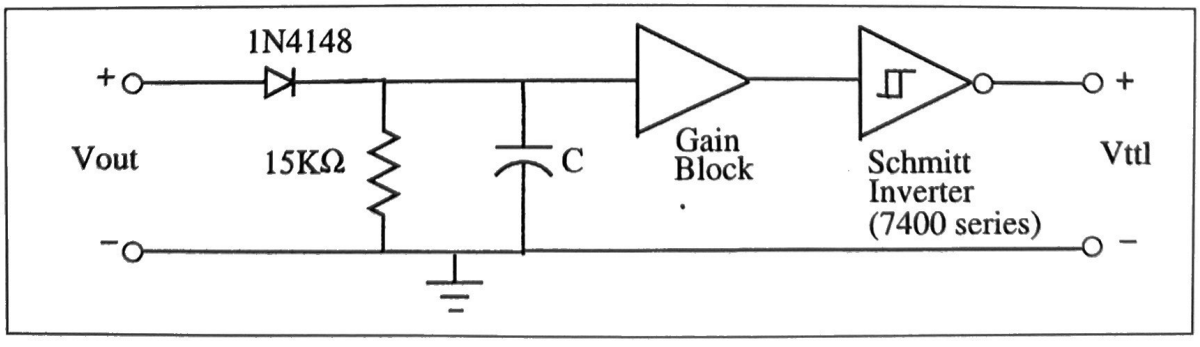
8254 to make a square wave at the frequency of your choice. Notice that you will need to hook a TTL crystal clock to one of the CLK pins on the 8254.

Please do and answer the following:

- The XIOSELECT pin is active (low) for access to memory addresses in the range FE00h through FEFFh. How many different address locations in this range select the control register on your 8254 chip? What 8254 register is located at FE00h? What 8254 registers are located at FE01h, FE02h, and FE03h, respectively?
- For this question, imagine that we could drive the CLK0 pin either from a 1 MHz crystal or a 10 MHz crystal clock. Which choice gives the lowest possible 8254 output frequency on OUT0 in Mode 3? Which choice gives the highest possible output frequency? Which crystal provides the best output resolution around 30 kHz (where our fluorescent light might operate)? That is, suppose the 8254 was operating in Mode 3 and producing an output square wave close to 30 kHz. Then suppose we change the "count" loaded into the timer register by adding or subtracting 1 from the count. Which input crystal, 1 MHz or 10 MHz, would provide the smallest change in output frequency?
- For this part, connect the 8254 CLK0 pin to a 10 MHz TTL crystal oscillator. Use your new MINMON read and write commands to configure the 8254 for Mode 3 operation. Produce a square wave as close to the resonant frequency of your fluorescent lamp as you can, and use your 8254 output to strike the lamp. Recall the correct sequence for striking the lamp. First, create the square wave you need on the input pin of the lamp. Make sure that the wall transformer is connected and providing power to the lamp. Then turn on the lab power supply that provides the 40 volts to the inverter in the lamp ballast. NOTE: You do **not** need to write or assemble a program for this part. You should be able to "take control" of your 8254 using interactive commands in MINMON. Make sure that you understand and can use the 8254 correctly **before** you hook up to the lamp ballast!

EXERCISE 7: Control the lamp

Now we will develop circuitry and a control program that allows the microcontroller to automatically ignite the lamp. Use an 8254 chip with a 10 MHz crystal to produce the square wave to drive the lamp fixture. To automatically determine a proper drive frequency, the microcontroller will need some sort of feedback. So, we will first construct a circuit that produces a signal with information about the status of the lamp. A useful feedback circuit might look like this:



The left-most input port of the feedback circuit should be connected to the Vout+ and Vout- pins on the signal connector of the lamp ballast board. The diode and RC filter rectify and smooth the AC lamp voltage to provide a DC voltage level across the capacitor. This DC level indicates the amplitude of the lamp voltage sinusoid. Anticipating lamp frequencies from 10 kHz to 50 kHz, an RC pair with a time constant of between 1 to 15 milliseconds will provide adequate filtering. The gain block should be a

noninverting and adjustable gain that can vary between 2 and 50. Our prototype used an LM358 op-amp and potentiometer to make this adjustable gain block. The Schmitt inverter is one gate from a 74LS14 TTL package. The signal Vttl should be connected to a Port 1 pin on the microcontroller. Make sure that your feedback circuit and the R31JP share a common ground connection! Ground any unused 74LS14 inputs!

Please do and answer the following:

- Design and build your feedback circuit. Select an appropriate value for C. Design a variable, adjustable gain block.
- Connect your feedback circuit to the lamp fixture, i.e., the left-most port of the feedback circuit, to Vout+ and Vout- on the signal connector of the lamp. Do not yet connect Vttl to your microcontroller. Instead, use a scope to monitor Vttl so you can understand how the feedback circuit works. Set your lab power supply to provide the lamp fixture with 40 volts. Strike your lamp using a signal generator or the 8254. If you play with the gain block, varying the gain from low to high and back, you should see the Schmitt trigger output change. The output will be a logic HIGH for low gains, and a logic LOW for high gains. Now, with the lamp lit, set your gain block to the lowest gain possible. The Schmitt trigger output should be a logic HIGH. Adjust the gain block to the largest gain possible without letting the Schmitt output go LOW. That is, the gain should be as large as possible so that the output of the Schmitt gate remains HIGH when the lamp is on.
- Play with the signal generator frequency. Start high, around 50 or 60 kHz. Watch Vttl. You should notice that, significantly above resonance, the lamp voltage is low and the Schmitt output is a logic HIGH. As you lower the drive frequency, approaching resonance, the lamp voltage will increase. Eventually, the Schmitt output should change states, going LOW when the lamp voltage is high enough. This transition or negative edge tells you that you are nearing the resonant frequency that will result in a high lamp voltage. When you near resonance and the lamp strikes, the lamp voltage should drop suddenly, and the Schmitt output will become a logic HIGH. This transition or rising edge tells you that the lamp has struck.
- Write a program that automatically strikes the lamp. It should start at a high frequency, known to be above the lamp resonance, e.g., 50 or 60 kHz. The program should then step the frequency down fairly quickly, checking to see when you are near the "striking zone" by looking for a LOW Schmitt output. Once you enter the striking zone, step down in frequency more slowly and with fine resolution. Check at each frequency step to see if the Schmitt output has popped back up to HIGH, indicating that the lamp has struck. Once the lamp has struck, the microcontroller should "lock in" or maintain that drive frequency.
- Your program and feedback circuit should allow the microcontroller to automatically strike the lamp even if the resonant frequency moves around. Your microcontroller should be able to find the new resonant frequency and strike the lamp over a reasonable range of frequencies, e.g., 20 kHz to 50 kHz. We will check your code for elegance and efficiency. For example, your code should strike the lamp as quickly as is reasonably possible. Excessive delays, such as taking 10 or 20 seconds to light the lamp, would **not** be acceptable in a commercial product.

EXERCISE 8: Make sure you understand the 8051 hardware

As you know, we have provided you with a microcontroller board, the R31JP, which eliminates the tedium of wiring up six or seven chips to make a minimum system before you can play. However, it's important that you understand the hardware and the general features and details of microcontroller hardware. When you use microcontrollers to design a product, it's very likely that you will have to construct your own minimum system. In lecture, we'll look not only at the R31JP but also another single board computer (SBC) based on the 8051 microcontroller shown on the last two pages of this lab. One

page shows the "processor core", the other shows the "memory subsytem" connected to the address bus and data bus provided by the processor core. Use these schematics to test your understanding of the 8051 architecture. Please look at the system shown on the last two pages, and answer the following questions (bring this sheet to your checkoff):

- How long (in seconds) is one machine cycle on this SBC: _____

- The memory subsystem contains 3 memory chips: Two ROMS labeled C1 and C3 in the schematic, and one RAM labeled D3 in the schematic. Which chip would most profitably contain MINMON (circle your answer):

C1 C3 D3 None of these

- Given the SBC as connected in the attached schematics, after a power-on reset, the first code-store fetch will be from (circle all which apply):

Internal 8051 Rom C1 C3 D3 None of these

- The MINMON command "G8000" could be used to execute a program or routine on this SBC given only the chips and interconnect shown on the attached schematics:

TRUE FALSE

- The MINMON command "G2000" would execute a program stored in (circle all which apply):

Internal 8051 Rom C1 C3 D3 None of these

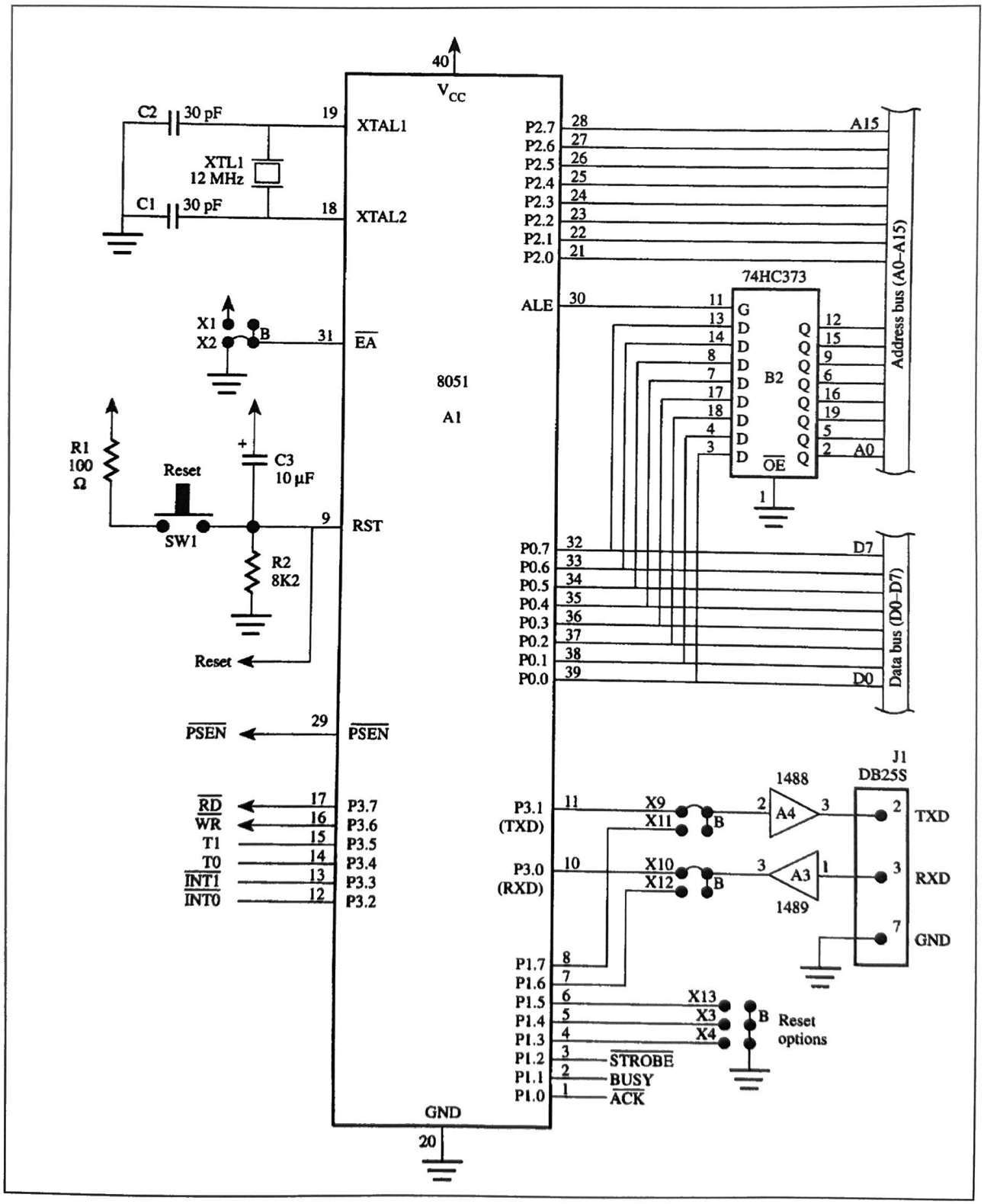
- Given the SBC as connected in the attached schematics, the 8051 instruction "MOVC A, @A+DPTR" could be used to read data bytes stored in (circle all which apply):

Internal 8051 Rom C1 C3 D3 None of these

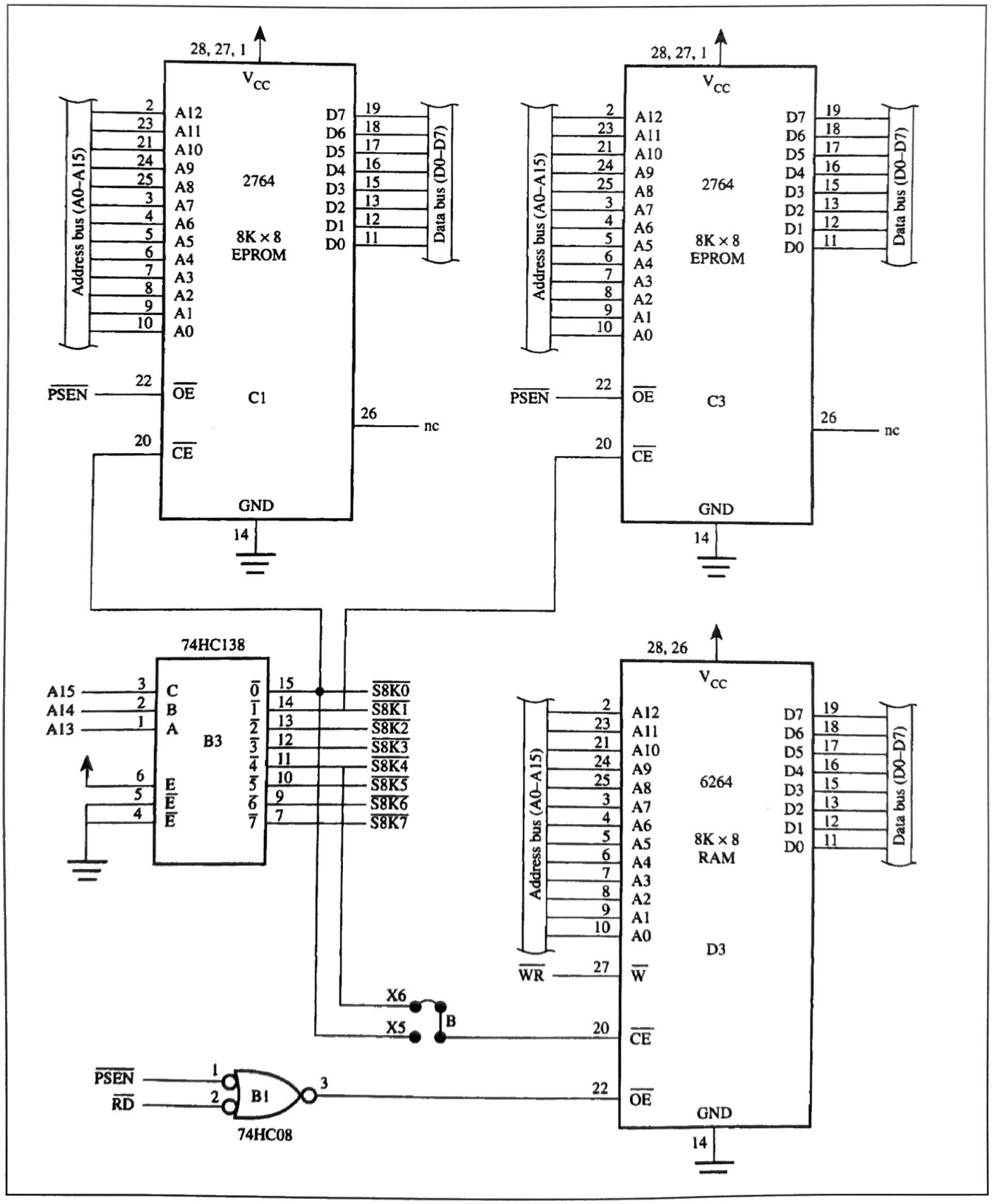
- Given the SBC as connected in the attached schematics, the 8051 instruction "MOVX A, @DPTR" could be used to read data bytes stored in (circle all which apply):

Internal 8051 Rom C1 C3 D3 None of these

Single Board Computer: Processor Core



Single Board Computer: Memory Subsystem



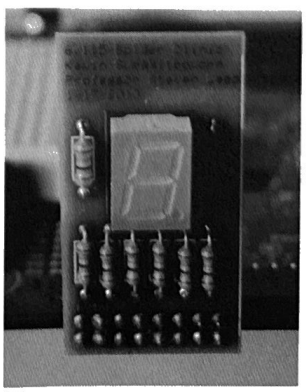
EXERCISE 9: Learn about the Cypress PSoC Big Board

In the previous lab, you started using the PSoC STICK, primarily focusing on the hardware. In this exercise, we will take a look at the PSoC Big Board and see how a PSoC is programmed in the C language with Creator. In the last decade, a new concept called "System on Chip" has begun to alter the way designers think about using microcontrollers. Cypress Semiconductor has created PSoC, a family of microcontrollers with an amazing set of hardware and software features on a single chip at relatively low prices (a few dollars per chip). In 6.115, you will have the chance to play with PSoC. On a single IC, a PSoC family microcontroller offers flash program memory, data memory, flexible digital blocks that can be used to implement all sorts of digital functions including serial ports, parallel ports, analog-to-digital converters, digital-to-analog converters, I2C, USB connectivity, and, amazingly, flexible op-amp blocks for handling analog signals as well. You decide how you want to use the hardware, and having a PSoC IC is like getting to pick from an electronics parts catalog with instant delivery. The PSoC family includes several different types of CPU cores, including a line of PSoCs with 8051 cores, and also a line of PSoCs with ARM cores (commonly used in cell-phones, for example).

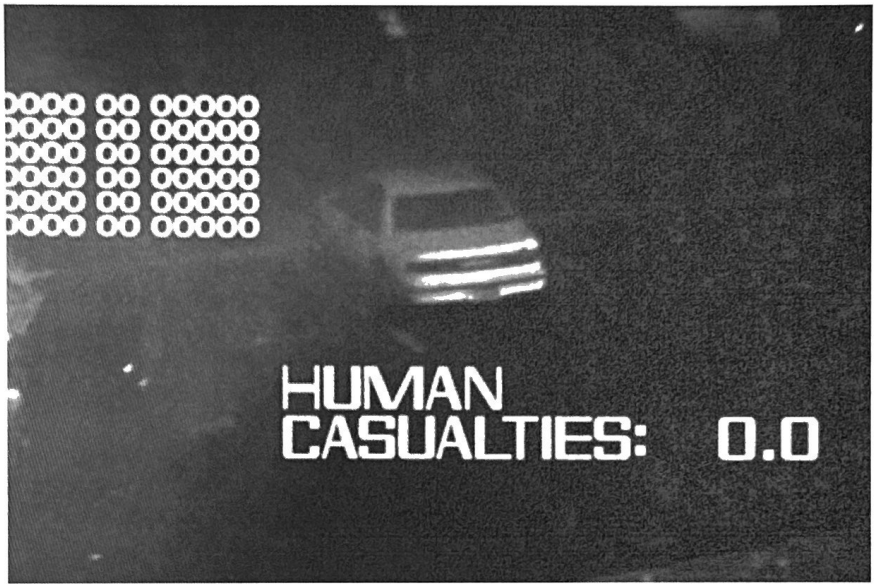
We will play with the PSoC to learn about single-chip microcontrollers and also to experience "high-level" programming in a language called "C". Cypress provides a C compiler for use with PSoC in its "PSoC Creator" software. You installed your Creator software in previous labs. This software is free, and you are welcome to install copies on your home computer.

REMEMBER: The PSoC board is expensive, and we will NOT be able to replace it if you blow it up. Do not be careless with the board. There is no reason to have an accident with it, particularly during the laboratory exercises, if you are careful. ASK if you have questions before making a mistake with the board. Some tips to note:

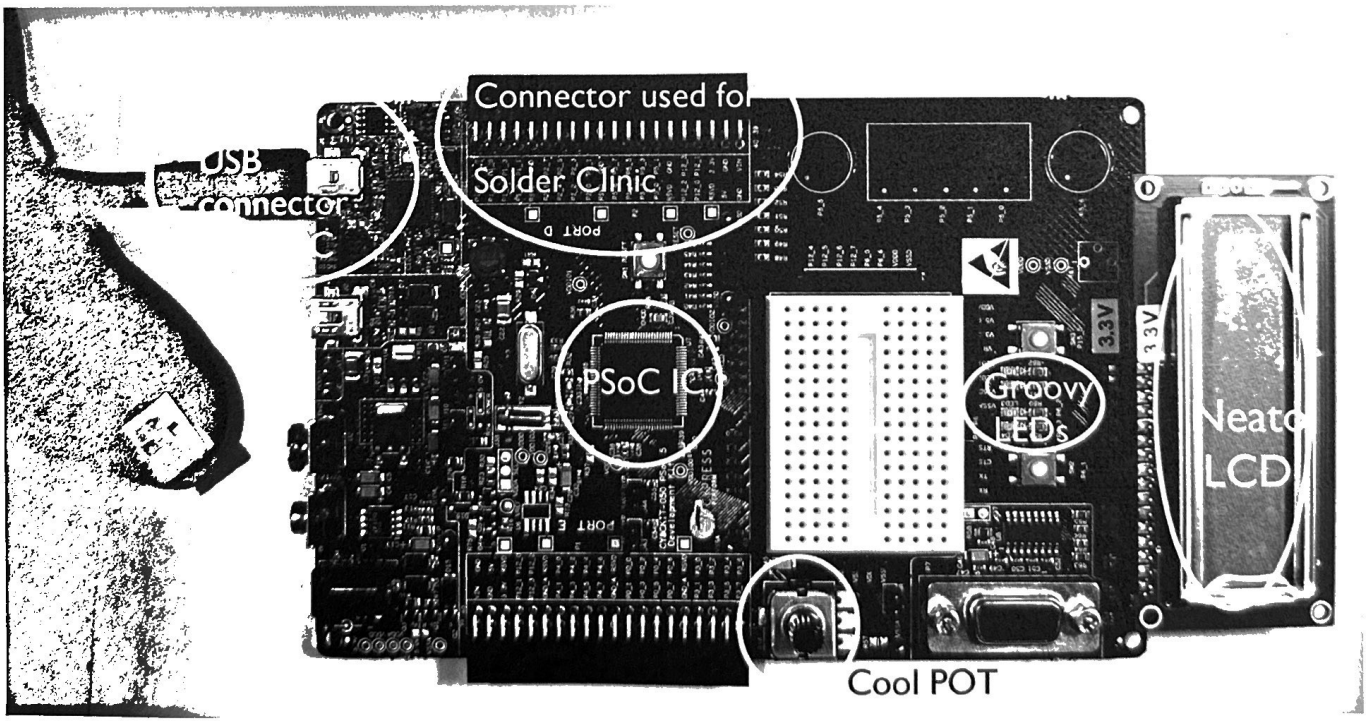
1. ONLY power the board with the USB connection to the computer. This connection also allows you to program the board using Creator. ONLY use the top, left-side USB connector shown in the picture below.
2. Do NOT connect anything to the board other than the USB connector and the DIGIT LED peripheral board we will discuss below. Ask the staff if you have any confusion about how to connect the LED peripheral board for the following lab exercises. Do NOT connect the PSoC board to your Briefcase KIT, and do not use signals from the board to drive other parts, e.g., on your KIT.
3. These are precautions. We have found that a common reason why people destroy these expensive boards is that they fail to connect grounds between the PSoC evaluation board and other IC's on a kit or elsewhere. Also, the boards have been destroyed through the application of excessive voltage. For now, all of these and many other problems can be avoided by using the evaluation board "as is," with only the USB connector and the one addition of the 6.115 DIGIT LED peripheral board. Please be careful.
4. Note that, when you open an older PSoC project, you may be asked to "update components" by Creator. Please do so.
5. A picture of your DIGIT LED board is shown to the right. Please find this board in your kit. Your board may have minor cosmetic differences (on the silkscreen), but it should look essentially like the picture to the right. **Solder this board together at a "solder clinic" on campus.**



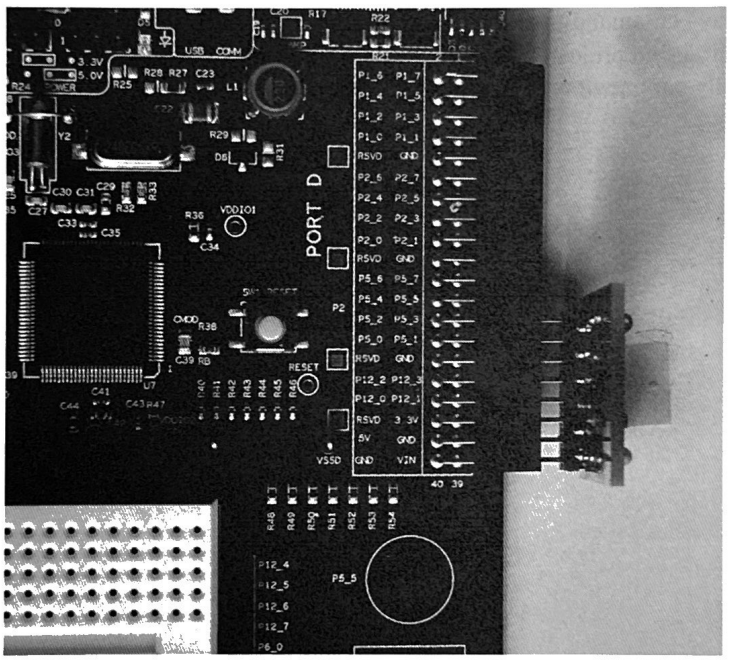
(We assume that there is no need to convince you of the importance of being able to use a digit LED display. Along with a blinking light, a digit display is one of the most important features for practically any project. Note, for example, that while the PSoC might be most suitable for a liquid-metal T1000 Terminator, even the T800 prefers digit displays, as shown in the HUD screen shot below. If you have **not** already won a gift certificate, be the first to e-mail Professor Leeb with the exact title of the immortal, must-see classic movie containing this scene and win a $10 Amazon certificate!)



A full picture of your PSoC evaluation Big Board is shown below, with area labels (the "lab PC" is your Windows 10 computer):



Here's what the Digit LED board looks like while it's being inserted into the PSoC evaluation board area labeled "Connector used for Solder Clinic." NOTE that the Digit LED board has a double row of pins, which should be inserted into the double row of socket holes on the "Connector...". Make sure that you study the pictures carefully, understand the orientations of the Big Board and Digit LED board, and ask questions as needed BEFORE you insert the Digit LED board. The picture below shows the Digit LED board in the middle of pushing it in for insertion. Make sure that your LED board is fully inserted before using the evaluation board.



The "7-segment" display on the Digit LED board is a convenient array of LEDs prepackaged to ease display applications. You can read more about this type of display here:

https://www.electronics-tutorials.ws/blog/7-segment-display-tutorial.html

Please do the following:

- Sign up for a 6.115 solder clinic appointment and build your LED peripheral board. This board is a small printed circuit board designed to plug into our PSoC evaluation boards.
- Go to the "PSoC" page on the 6.115 course website: http://web.mit.edu/6.115/www/page/psoc-information.html
- Download the "Blinky" code for PSoC creator and make sure that you remember how to use Creator. Run Blinky so that you see the "6.115" message on your Digit LED display. In the Creator software, select the correct PSoC target device (CY8C5868AXI-LP035) using the "Device Selector..." under the "Project" menu tab. Use the "Build" menu tab to build Blinky, and use the "Debug" menu tab to program the PSoC evaluation board. Make sure you are comfortable "building" and "programming" the PSoC and that you can see the LED display working. In Creator, make sure that you can find and edit the files with ".cydwr", ".c", and ".cysch" for the project. These files are, respectively, the "design wide resources," the "C-language program code," and the internal "schematic" for the project.

- Now, from the 6.115 course website, download the "Exercise 1" for the PSoC (on the webpage right under the "Blinky" project). Complete Exercise 1 by following the instructions on the file page with a ".cysch" extension for Exercise 1. Build and program the project on your PSoC board. You should be rewarded with a regularly flashing light in your bank of "Groovy LEDs" on the evaluation board.
- Modify the c-code for the Exercise 1 project so that the light flashes a pattern corresponding to "SOS" in Morse Code: Three short flashes, followed by three long flashes, followed by three short flashes, followed by a 1 second pause. The SOS should repeat indefinitely. You may find the "CyDelay" command useful in your code.
- Save your modified project. Include a phone picture of the light flashing on, and also include your carefully commented, modified SOS C-code in your lab report.