1 Small Changes to SiLab Code (using built-in ports) (1 point)

Here is the first of the SiLabs programs: a tiny loop, along with some initializations. These initializations are explained in Classnotes μ 1.

; bitflip.a51 this blinks LED--but at a rate too fast to see unless run in single-step
$NOSYMBOLS ; keeps listing short..
$INCLUDE (C:\MICRO\8051\RAISON\INC\c8051f410.inc)

ORG 0 ; tells assembler the address at which to place this code
; Disable the WDT.
ANL PCA0MD, NOT(040h) ; Clear Watchdog Enable bit

; Enable the Port I/O Crossbar
MOV XBR1, #40h ; Enable Crossbar

SETB P0.0 ; start with LED off (it’s active low), just to make it predictable

FLIPIT: CPL P0.0 ; flip LED

SJMP FLIPIT

END

**Your Task**

In order to demonstrate that you recognize how this very simple program works, please modify it as follows (you can just pencil in your changes, above):

- use two LED’s, at port bits P0.0 and P0.1
- whenever one of them turns ON, let the other almost-immediately turn OFF—so that as they blink they trade roles

Don’t worry about the fact that this program would be useless, running at full speed. Assume that we will single step it (and perhaps later will patch in a delay).
2  Byte In, Byte Out (3 points, total)

Show how to wire an 8-bit ADC and 8-bit DAC to an 8051, in two different setups—one using external buses, the other using built-in ports.

The ADC is of the simplest sort: it always has a good value available, and starts itself as soon as it finishes a conversion (as your Lab ADC did, a couple of labs ago). The DAC also is simple, including no input register.

2.1  …Use external buses (2 points)

Put both ADC and DAC at Port 1, as defined by the I/O decoding of the “Big Board,” and using the data bus.

2.2  …Use internally-defined ports (1 point)

Do the same task, but this time use the 8051’s built-in ports 0 and 1.1 Put ADC at 0, DAC at 1.

3  An Input Switch: Hardware (2 points)

two versions)

3.1  …Use external buses

Again assume you have available the I/O decoding of the “Big Board.” Use this to let the computer sense the level of a switch connected to bit 0 of port 1. This is “port 1” using the external buses, not “port 1” as an internally-decoded controller port. The switch is single-throw (the simple kind). Don’t worry about bounce.

1These ports are available on the SiLabs controller, and would be available on the Dallas part if it were not wired to buses; in our Dallas wiring, the buses occupy port 2.
3.1.1 …Use built-in port

Let the computer sense the level of a switch connected to bit 0 of port 1. This is “port 1” as an internally-decoded controller port: P1.0.

4 Two Output Devices: Hardware (4 points, total)

4.1 Add an LED output to your computer (two versions) (2 points total)

4.1.1 …Use external buses (1 point)

Again assume you have available the I/O decoding of the “Big Board.” Let the computer light an LED (and keep it lit till changed under program control) by sending a High to bit 0 of port 1. This is “port 1” using the external buses, not “port 1” as an internally-decoded controller port. Assume the LED drops 2V and wants 1.5mA.

4.1.2 …Use built-in port (1 point)

… and draw the LED drive if it is done at an 8051 built-in port: P1.0. Choose one design that works for both a Dallas part and a SiLabs part. Their specs are shown in a page at the end of this HW. As you look at these specs, you’ll be confronted with a lot of information that you don’t want. What you’re hunting for is information that will let you decide whether to sink or source current in the LED. Explain your choice, briefly, referring to the specs.

4.2 Add a heater drive output to your computer (2 points)

Again assume you have available the I/O decoding of the “Big Board.” Let the computer control a heater (an inductive device that passes 1A when powered by a +10V supply). The computer should turn this on by sending a High to bit 1 of port 1. Show this in two versions:

… Use external buses, and use built-in port
Show this using bit 1 of the decoded “port 1” using the external buses, and a second design that would use the built-in port P1.1

5 Pushbutton Response Hardware 3 points, total

Show hardware that would allow the lab computer to test whether two pushbuttons (tested independently) had been pressed. The pushbuttons are the simple sort of switch: SPST. They ground a line, or leave it floating. Show how to feed in those two signals, which we’ll call BUTTON_A and BUTTON_B—for lack of a smarter idea. We’d like you to show the interfacing hardware in stages:

5.1 Debounce the two switches 1 point

Show the debouncer once; for the second switch just draw a box labelled ’debouncer.’

5.2 Feed the debounced signals to the computer: External Bus 1 point

Put the signals on data lines D0 and D1 of port 1 (as defined by the I/O decoder, the ’139 that you added in Lab 19).

5.3 Feed the debounced signals to the computer: 8051 Port 3 1 point

Use the 8051’s internally-defined PORT3, bits 0 and 1.
6 Pushbutton Response Software 5 points, total

6.1 Code to test the external bus buttons 3 points

Assuming that the buttons are wired through the external bus, show code (assembly language, not hex codes!) that would allow the lab computer to test whether two pushbuttons (tested independently) had been pressed. If BUTTON_A is pressed, do a branch subroutine (ACALL, for example) to DOA and don’t test BUTTON_B; if BUTTON_A is not pressed but BUTTON_B is pressed, ACALL DOB. If neither is pressed, fall out of the program (don’t worry about what happens next).

6.2 Code to test the internally-defined PORT3 buttons 2 points

Assuming that the buttons are wired through the 8051’s internally-defined PORT3, show code (assembly language, not hex codes!) that would allow the lab computer to test whether two pushbuttons (tested independently) had been pressed. If BUTTON_A is pressed, do a branch subroutine (ACALL, for example) to DOA and don’t test BUTTON_B; if BUTTON_A is not pressed but BUTTON_B is pressed, ACALL DOB. If neither is pressed, fall out of the program (don’t worry about what happens next).
7  PWM: Sawtooth brightness control changed to Triangle (3 points, total)

For a change, here we’ll ask Big Board people to scrutinize a SiLabs lab. Here is an excerpt from SiLabs Lab $\mu$ 3, illustrating the controller’s digital PWM hardware (“Pulse Width Modulation:” varying of duty cycle or fraction of a cycle a binary signal spends at HIGH).

7.1  PWM, Analog and Digital Versions

The technique of varying drive to a load by varying duty cycle on an output pin is quite easy to implement on a controller. The hardware required is almost nil, in contrast to what is required for continuous variation of voltage or current, which requires a DAC. PWM can be done entirely in software. But the ’410 offers PWM in an easier form. The PCA timer can be used to count an 8- or 16-bit value, and to set a bit when the count exceeds a reference value. Thus in purely digital form it can mimic the analog PWM method, in which a waveform ramps (sawtooth or triangle) and triggers a comparator when ramp value exceeds an analog reference value. Here is a circuit sketch from Lab Op Amps 3, to refresh your recollection of the scheme.

Figure 1: Analog PWM circuit of Lab Op Amps 3
Here is the digital equivalent, which the ’410 uses to implement an 8-bit PWM. When $Count = Reference$, the output bit goes High; when $Count$ rolls over, output bit goes Low.

Figure 2: ’410 implements PWM in digital form
The lab includes code that generates a ramp to drive a PWM device (gradually increasing the duty cycle of the signal on a pin that drives an LED):

```assembly
UP: ACALL DELAY
    MOV PCA0CPH0, A ; this register, PCA0PH0, is a comparison register against which a continually-running ramp is compared.
    INC A ; When ramp exceeds PCA0PH0 value, output (to LED) goes high
    SJMP UP
```

7.2 Preliminary question: sawtooth? (0.5 points)

Explain in a few words why the code shown above produces a sawtooth duty-cycle.² You may find the SiLabs µ3 writeup helpful. It’s a little fuller than what we present here.

7.3 Triangle (2.5 points)

Show code that would apply a triangular rather than sawtooth duty cycle to the LED.

```assembly
UP_DOWN: ACALL DELAY ; loop that will ramp value of PCA0CPH0 up, then down, then up..., endlessly
```

²In this digital version of PWM there is no literal sawtooth waveform generated; we are likening the digital behavior to the analog. If it helps, imagine the waveform as what you would see if you fed the digital values in COUNT to a DAC.
8 Port Specifications: Dallas and SiLabs 8051’s

DC ELECTRICAL CHARACTERISTICS

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Symbol</th>
<th>Min</th>
<th>Typ</th>
<th>Max</th>
<th>Units</th>
</tr>
</thead>
<tbody>
<tr>
<td>Supply Voltage (Notes 1, 2, 3)</td>
<td>VDD</td>
<td>4.5</td>
<td>5.0</td>
<td>5.5</td>
<td>V</td>
</tr>
<tr>
<td>Power-Up Pull-Up (Notes 2, 4)</td>
<td>VDD</td>
<td>4.2</td>
<td>4.35</td>
<td>4.5</td>
<td>V</td>
</tr>
<tr>
<td>Reset Trip Point (Min Operating Voltage) (Notes 2, 3, 4)</td>
<td>VDD</td>
<td>3.55</td>
<td>4.125</td>
<td>4.35</td>
<td>V</td>
</tr>
<tr>
<td>Supply Current, Active Mode (Note 5)</td>
<td>Icc</td>
<td>75</td>
<td>110</td>
<td>mA</td>
<td></td>
</tr>
<tr>
<td>Supply Current, Idle Mode at 33MHz (Note 6)</td>
<td>Icc</td>
<td>42</td>
<td>59</td>
<td>mA</td>
<td></td>
</tr>
<tr>
<td>Supply Current, Idle Mode (Note 7)</td>
<td>Icc</td>
<td>1</td>
<td>100</td>
<td>μA</td>
<td></td>
</tr>
<tr>
<td>Supply Current, Idle Mode, Bandgap Disabled (Note 7)</td>
<td>Icc</td>
<td>159</td>
<td>359</td>
<td>μA</td>
<td></td>
</tr>
<tr>
<td>Input Low Level (Note 2)</td>
<td>VIL</td>
<td>-0.3</td>
<td>0.8</td>
<td>V</td>
<td></td>
</tr>
<tr>
<td>Input High Level (Note 2)</td>
<td>VIH</td>
<td>2.0</td>
<td>2.0</td>
<td>V</td>
<td></td>
</tr>
<tr>
<td>Input High Level XTAL and RST (Note 2)</td>
<td>VIL</td>
<td>3.5</td>
<td>0.0</td>
<td>V</td>
<td></td>
</tr>
<tr>
<td>Output Low Voltage, Port 1 and 3 at VDD = 3.3mV (Note 2)</td>
<td>VILH</td>
<td>0.15</td>
<td>0.45</td>
<td>V</td>
<td></td>
</tr>
<tr>
<td>Output Low Voltage, Port 0 and 2, ALE, PSEN at VSS = 3.3mV (Note 2)</td>
<td>VIL2</td>
<td>0.15</td>
<td>0.45</td>
<td>V</td>
<td></td>
</tr>
<tr>
<td>Output High Voltage, Port 1, 2, and 3 at VDD = -50mV (Notes 2, 6)</td>
<td>VILH</td>
<td>2.4</td>
<td>2.4</td>
<td>V</td>
<td></td>
</tr>
<tr>
<td>Output High Voltage, Port 0, 1, 2, ALE, PSEN, RD, WR in Bus Mode at VSS = -50mV (Notes 2, 10)</td>
<td>VILH</td>
<td>2.4</td>
<td>2.4</td>
<td>V</td>
<td></td>
</tr>
<tr>
<td>Output High Voltage, RST at VSS = -0.4mV (Note 2, 11)</td>
<td>VILH</td>
<td>2.4</td>
<td>2.4</td>
<td>V</td>
<td></td>
</tr>
<tr>
<td>Input Low Current, Port 1, 2, and 3 at 0V</td>
<td>IIL</td>
<td>-50</td>
<td>50</td>
<td>μA</td>
<td></td>
</tr>
<tr>
<td>Transition Current from 1 to 0, Port 1, 2, 3 and 2V (Note 12)</td>
<td>Ic</td>
<td>550</td>
<td>550</td>
<td>μA</td>
<td></td>
</tr>
<tr>
<td>Input Leakage Current, Port 5 in I/O Mode and TX (Note 13)</td>
<td>IL</td>
<td>-10</td>
<td>+10</td>
<td>μA</td>
<td></td>
</tr>
<tr>
<td>Input Current, Port 0 in Bus Mode (Note 14)</td>
<td>IIL</td>
<td>-200</td>
<td>+200</td>
<td>μA</td>
<td></td>
</tr>
<tr>
<td>RST Pullup Resistance (Note 13)</td>
<td>RST</td>
<td>50</td>
<td>120</td>
<td>200</td>
<td>kΩ</td>
</tr>
</tbody>
</table>

Note 1: Specifications to ±40°C are guaranteed by design and not production tested.
Note 2: All voltages are referred to ground.
Note 3: The user should note that this port is not intended for power-supply rails or operating at voltages above VDD.
Note 4: All voltages are referred to ground.
Note 5: The user should note that this port is not intended for power-supply rails or operating at voltages above VDD.
Note 6: Active current is measured with a 330kHz clock source driving XTAL1, VCC = 5V. All other pins are disconnected.
Note 7: Idle mode current is measured with a 330kHz clock source driving XTAL1, VCC = 5V, RST at ground. All other pins are disconnected.
Note 8: During a 0-1 transition, a one clock cycle delay time is provided in both transition modes.
Note 9: This port is a weak address holding latch in bus mode. Peak current occurs near the input transition point of the holding latch at approximately 2V.

Table 18.1. Port I/O DC Electrical Characteristics

<table>
<thead>
<tr>
<th>Parameters</th>
<th>Conditions</th>
<th>Min</th>
<th>Typ</th>
<th>Max</th>
<th>Units</th>
</tr>
</thead>
<tbody>
<tr>
<td>Output High Voltage</td>
<td>VDD = -3mV, Port I/O push-pull</td>
<td>VDD = 0.6V</td>
<td>—</td>
<td>—</td>
<td>V</td>
</tr>
<tr>
<td>Output High Voltage</td>
<td>VDD = 70μA, Port I/O push-pull</td>
<td></td>
<td>50</td>
<td></td>
<td>mV</td>
</tr>
<tr>
<td>Output High Voltage</td>
<td>VDD = 8.5mA</td>
<td></td>
<td>460</td>
<td></td>
<td>μA</td>
</tr>
<tr>
<td>Output High Voltage</td>
<td>VDD = 4.0V</td>
<td></td>
<td>40</td>
<td></td>
<td>μA</td>
</tr>
<tr>
<td>Output High Voltage</td>
<td>VDD = 70μA</td>
<td></td>
<td>460</td>
<td></td>
<td>μA</td>
</tr>
<tr>
<td>Output High Voltage</td>
<td>VDD = 8.5mA</td>
<td></td>
<td>460</td>
<td></td>
<td>μA</td>
</tr>
<tr>
<td>Input Low Voltage</td>
<td>VDD x 0.7</td>
<td>—</td>
<td>—</td>
<td>V</td>
<td></td>
</tr>
<tr>
<td>Input Low Voltage</td>
<td>—</td>
<td>—</td>
<td>VDD x 0.3</td>
<td>—</td>
<td>V</td>
</tr>
<tr>
<td>Input Leakage Current</td>
<td>Weak Pullup Off</td>
<td>—</td>
<td>&lt;0.1</td>
<td>μA</td>
<td></td>
</tr>
<tr>
<td>Weak Pullup Impedance</td>
<td>—</td>
<td>120</td>
<td>—</td>
<td>kΩ</td>
<td></td>
</tr>
</tbody>
</table>

Figure 3: Port Specifications, Dallas and SiLabs versions of 8051

hw4apr15.tex; April 13, 2015