Reading: Chapter 7, Supplemental

Quiz 3

1. Your Name
2. How many pins of the parallel port do we use on our parallel port controller and what do they do?
3. Can diodes allow current to flow in either direction?
4. What does a pull down resistor do?
Today’s Topics

- Examine the parallel port controller schematic
- Examples of parallel port input - hardware
- Addressing the parallel port
- An example of BASIC code
- Addressing the parallel port - Windows
- An example of C code
- Pulse width modulation - saving current using computer control
- Input code examples

The New Board you Made . . .

- Data register bits 0 through 7, address 0x378
- Includes the added input on the parallel port
- Pin 13 is read at bit bit 28 of the control register of the parallel port
- Control register port address 0x37A
Addressing the Parallel Port

• Historically, one would address input/output ports “nearly directly,” with a specific address (from 0 to x7FF)
• Data Register = port 0x378 of the PC
• In Basic, "OUT &H378" sends an 8-bit value to the printer port. The data sent is hexadecimal:
  OUT &H378, &H0F (binary 00001111)

• Control Register = port 0x37A - it’s 32 bits wide
• In Basic, "IN &H37A" reads an 8-bit value from the printer port. The data sent is hexadecimal:
  STUFF=IN &H37A

REM Program to make Stiquito walk with a tripod gait. This REM assumes that the upper nibble controls one tripod, and the REM lower nibble controls the other. We allow the nitinol to REM rest after it is activated.

    REM "OUT &H378" sends an 8-bit value to the printer port. The REM data sent is hexadecimal.
    
    DELAY = 14000
    10 OUT &H378, &HF0 : REM &HF0 is binary 11110000
         FOR x = 1 TO DELAY : NEXT x

    OUT &H378, 0
    FOR x = 1 TO DELAY : NEXT x

    OUT &H378, &H0F : REM &H0F is binary 00001111
         FOR x = 1 TO DELAY : NEXT x

    OUT &H378, 0
    FOR x = 1 TO DELAY : NEXT x
Some More Code

Oh, to end when a key is pressed:
REM If a key on the keyboard was pressed
REM then end. Otherwise, blink some more!
a$ = INKEY$
IF a$ = "" THEN GOTO 10
END

In C? We’ll investigate this. There is no standard way to
do this, it depends on the compiler.

Using the Parallel Port - Windows

• With Windows, you may not have to address ports at
such a low level. This is the case with Windows
NT/2000.
• First, you have to link in some code (a windows
executable model is a “dll” - filename.dll)
• Next, you need to ensure you have some libraries
available (dllportio.h, led.h, windows.h)
• We have created a file, led.c, which will handle how to
access the ports directly. He have given you a “high
level” interface.
• You will need to compile some modules, and link them
together. This is done with a makefile (see the website)
So how do we Program the Parallel Port?

• We have an Application Programming Interface (API) called DLPORTIO (DriverLINX Port I/O)
• Functions are described in dlportio.h
• Use DlPortReadPortXXXX(PORT_ADDR) to read information
• Use DLPortWritePortXXXX(PORT_ADDR) to write information
• Three versions of each:
  • UCHAR = unsigned char = byte = 8 bits
  • USHORT = unsigned short = 16 bits
  • ULONG = unsigned long = 32 bits

The Software Architecture of our System

Your Program
main

LED Interface
led_on
led_off

Button Interface
is_button_down

DLPORTIO
DlPortReadPortUchar
DlPortReadPortUshort
DlPortReadPortUlong
DlPortWritePortUchar
DlPortWritePortUshort
DlPortWritePortUlong

led.h
led.c
button.h
button.c
dlportio.h
dlportio.lib
Recap of C

Some skills - building a byte with several bits set:
#define LED_D1_M 0x01
#define LED_D2_M 0x02
#define LED_D3_M 0x04
#define LED_D4_M 0x08
#define LED_D5_M 0x10
#define LED_D6_M 0x20
#define LED_D7_M 0x40
#define LED_D8_M 0x80

unsigned char sendbyte;

sendbyte=LED_D5_M | LED_D6_M | LED_D7_M | LED_D8_M;

sendbyte is now 0xF0, or binary 11110000

Some more C

To end when a key is pressed:
/* If a key on the keyboard was pressed
 then end. Otherwise, blink some more! */

In C? We’ll investigate this.

Need some sleep?
Sleep(1000); /* sleep for 1000 ms */
Writing the LED’s Through the Parallel Port

Data register format:

<table>
<thead>
<tr>
<th>Bit</th>
<th>7</th>
<th>6</th>
<th>5</th>
<th>4</th>
<th>3</th>
<th>2</th>
<th>1</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>x378</td>
<td>D8</td>
<td>D7</td>
<td>D6</td>
<td>D5</td>
<td>D4</td>
<td>D3</td>
<td>D2</td>
<td>D1</td>
</tr>
<tr>
<td>Read/Write</td>
<td>W</td>
<td>W</td>
<td>W</td>
<td>W</td>
<td>W</td>
<td>W</td>
<td>W</td>
<td>W</td>
</tr>
</tbody>
</table>

Bit 7..0 - D8 - D1: LED On/Off Control
These bits control the LED’s at locations D8 - D1 on the board. A value of ‘1’ will turn on an LED. A value of ‘0’ will turn off an LED.

Reading the LED’s Through the Parallel Port

Control register format:

<table>
<thead>
<tr>
<th>Bit</th>
<th>7</th>
<th>6</th>
<th>5</th>
<th>4</th>
<th>3</th>
<th>2</th>
<th>1</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x37C</td>
<td>D8</td>
<td>D7</td>
<td>D6</td>
<td>D5</td>
<td>D4</td>
<td>D3</td>
<td>D2</td>
<td>D1</td>
</tr>
<tr>
<td>Read/Write</td>
<td>R</td>
<td>R</td>
<td>R</td>
<td>R</td>
<td>R</td>
<td>R</td>
<td>R</td>
<td>R</td>
</tr>
</tbody>
</table>

Bit 7..0 - D8 - D1: LED State
These bits indicate the state of the LED’s at locations D8 - D1 on the board. A value of ‘1’ means the LED is on. A value of ‘0’ means the LED is off.
#include <windows.h>
#include "led.h"

int main(void)
{
    /* Turn on LED's D5 - D8 for 1 second */
    led_on(LED_D5_M | LED_D6_M | LED_D7_M | LED_D8_M);
    Sleep(1000);

    /* Turn off LED's D5 - D8 for 1 second */
    led_off(LED_D5_M | LED_D6_M | LED_D7_M | LED_D8_M);
    Sleep(1000);

    return 1;
} /* end - main() */

Using the LED Functions - led.c

Control register format:

```
<table>
<thead>
<tr>
<th>Bit</th>
</tr>
</thead>
<tbody>
<tr>
<td>7</td>
</tr>
<tr>
<td>6</td>
</tr>
<tr>
<td>5</td>
</tr>
<tr>
<td>4</td>
</tr>
<tr>
<td>3</td>
</tr>
<tr>
<td>2</td>
</tr>
<tr>
<td>1</td>
</tr>
<tr>
<td>0</td>
</tr>
</tbody>
</table>
```

- R: Read
- W: Write

Bit 4 - B0: Button State

This bit indicates the state of the button. A value of ‘1’ means the button is being pressed. A value of ‘0’ means the button is not being pressed.
Input on the Parallel Port - C

The call to read from the Parallel Port:
`DlPortReadPortUchar(PORT_ADDR)` (in dlportio.h)

How do you check a specific bit?
```c
if ((DlPortReadPortUchar(PORT_ADDR) & MASK) == MASK)
{
    /* The bit(s) are set */
}
```

Motivation for Pulse Width Modulation

• We have a need to limit the amount of current that Stiquito uses (save battery life, run “cooler”)

![Diagram of percent of length that actuator contracts and apparent current](image)
Implementing Pulse Width Modulation

• All this means is that you should not keep the LEDs (or nitinol wires) “ON” for the entire time. Turn them off every so often.
• The exact amount of time depends on how you built Stiquito (every robot is different).

A code snippet for PWM - BASIC

REM High frequency pulses initially contract actuators
FOR a = 1 TO 20
    OUT &H378, &HF0              : REM &HF0 is binary 11110000
    FOR x = 1 TO 100 : NEXT x
    OUT &H378, 0
    FOR x = 1 TO 100 : NEXT x
NEXT a

REM Low frequency pulses maintain actuator contraction
FOR a = 1 TO 80
    OUT &H378, &HF0              : REM &HF0 is binary 11110000
    FOR x = 1 TO 100 : NEXT x
    OUT &H378, 0
    FOR x = 1 TO 800 : NEXT x
NEXT a
A code snippet for PWM - C

/* A PWM code fragment for one LED */
For(i==0;i<5;i++) {    /* loop for about 200 ms */
    led_on(LED_D8_M);
    Sleep(20);
    led_off(LED_D8_M);
    Sleep(20);
}

For(i==0;i<8;i++){    /* loop for about 800 ms */
    led_on(LED_D8_M);
    Sleep(20);
    led_off(LED_D8_M);
    Sleep(80);
}

What’s next?

• Next week we will examine controlling the Stiquito robot, and creating the “best gait”
• You can prepare by building your Stiquito tether (to connect to the Parallel Port Controller)
• You will use the code you create for lab 3 to help make Stiquito walk efficiently
Lab 3 . . .

• Monday’s Lab #3 - Use your parallel port controller, write software to do the following:
  • Flash each LED, one at a time, for one second, continuously, until a key is pressed. Start at bit 0, progress to bit 7.
  • Flash four LEDs (one nibble) for one second, then all off for one second, then flash the other four LEDs (other nibble) for one second, then all off for one second. Repeat until a keystroke is pressed.
  • Repeat the previous flashing, but this time, use pulse width modulation. During the first 0.2 seconds of the 1.0 second on time, turn the LEDs on for 20 ms, then off for 20 ms. During the remaining 0.8 seconds of on time, turn the LED on for 20 ms, then off for 80 ms.
  • Implement the function is_button_down() in button.c. It reads the parallel port control register. The function returns TRUE if the button is being pressed, FALSE otherwise.

Lab 3 - more

• Use the is_button_down() function to determine the state of the button. Print “on” to the screen if the function returns TRUE.

• Hint: Over the weekend, read Chapter 7 again carefully. Use information from the Chapter and from this lecture to write your code.

• ALSO make some measurements on lab equipment. I will set up two lab stations for measuring such things as current, voltage, resistance, and frequency.