Lab 2: Interfacing with a Sensor Device on an Embedded Computer System

Due Date: See the course schedule web page.

Objective

- Learn how to interface sensors with embedded computer systems
- Understand bus protocols
- Understand the operation of parallel ports

Description

By now you should have a working sensor device that can measure the light intensity in the surrounding environment. Next, we would like to connect it to an Atom based embedded computer system to process the sensory data. In general, there are many possible ways to connect a customized device to an embedded computer, depending on the available interfaces on the computer board and the sensor device. Examples include serial communication port, parallel port, USB, General Purpose I/O (GPIO), and even Ethernet. In this lab, we choose to use the parallel port of the embedded computer and design a customized interface protocol to acquire sensor data[1].

Our customized interface protocol governs the information exchange between the embedded computer and sensor device. At the conceptual level, the embedded computer sends commands to the sensor device, which responds with its data or status. At the bus connection level, your sensor device is connected to the parallel port, which provides data bus and control signals for our customized bus protocol. The details of the interface protocol are explained as follows.

Command/Response Protocol

(1) On the Atom based embedded computer, a user application on Linux issues commands to the sensor device.

```
/* user commands */
#define MSG_RESET      0x0    /* reset the sensor to initial state */
#define MSG_PING       0x1    /* check if the sensor is working
properly */
```

[1] While GPIO pins can be used in this lab instead of the parallel port, we choose the parallel port primarily because (1) it is easy to wire up the connections, (2) the parallel port remains universal in embedded computer systems, and (3) the access to parallel ports from Linux kernel is well documented.

```
#define MSG_GET         0x2     /* obtain the most recent ADC result */
```
Command **MSG_RESET** will reset the state of the sensor. Command **MSG_PING** checks if the sensor device is operational. Command **MSG_GET** asks the sensor device to return the most recent ADC value and its timestamp.

(2) On the sensor device, the PIC microcontroller responds to the user application (on Atom) with the following messages:

```
/* Sensor Device Responses */
#define MSG_ACK         0xE     /* acknowledgement to the commands */
#define MSG_NOTHING     0xF     /* reserved */
```

Usually a **MSG_ACK** message is replied from the sensor device after a command is received and corresponding actions are performed. Message **MSG_NOTHING** is a reserved message which should not appear in normal operations. **MSG_ACK** message follows the actual ADC and RTC values if the command from the embedded computer is **MSG_GET**. An example command/response transaction between the sensor and the embedded computer is as follows.

To obtain the most recent ADC result:

1. The user application (on Atom) sends **MSG_GET** to the sensor;
2. The sensor will output three 4-bit nibbles for ADC value and 8-byte RTC value to the embedded system, trigger by the "Strobe" signal from the parallel port; the user application (on x86) reads the three nibbles in sequence;
3. The sensor responds with **MSG_ACK** to tell the user application that it finishes reporting a pair of 10-bit ADC and timestamp.

Bus Protocol

The PIC microcontroller on the sensor device is connected to the embedded computer through the parallel port. A viable pin assignment is shown in Figure 1 although there can be certainly other possible assignments. Over the parallel port, six signal lines (D3-D0, Strobe and Ack*) are available for us as illustrated in Figure 2. The data bus (D3-D0) is **4-bit**. Control signals include **Strobe** and **Ack***.

However, note that **Ack*** is **NOT** used in this lab, instead it is to be used in Lab 3 for interrupt generation. That is, in Lab 2 we solely rely on **Strobe** signal to synchronize the information exchange between the sensor device and the embedded computer.

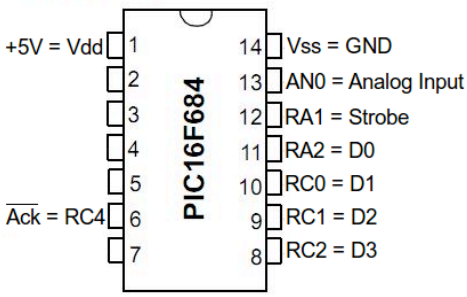


Figure 1. PIN Assignment of PIC

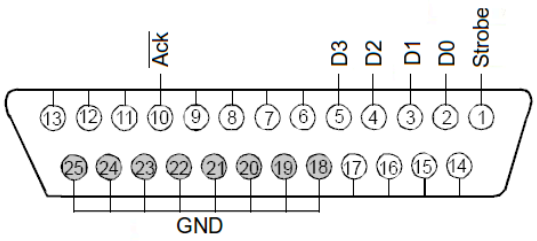
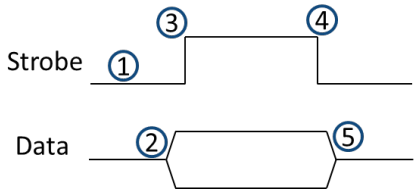


Figure 2. Pin Assignment on Parallel Port



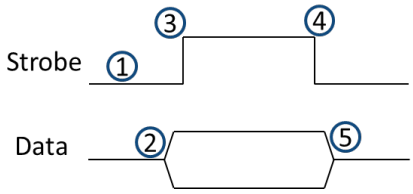
WRITE operation
(from computer's perspective)

Figure 3. Write operation on the bus

The steps (in Figure 3) involved in the write operation are as follows.

(1) The computer pulls the Strobe signal low. The PIC microcontroller gets ready to read 4-bit command message.
(2) The computer outputs a command on the data bus
(3) The computer raises the Strobe signal to indicate the command is ready on the bus. The PIC microcontroller starts reading the value (i.e. a command) from the bus. The computer maintains the value for at least 10ms.
(4) The computer ends the write operation by pulling the Strobe signal to low again. By this time, the PIC should have already finished reading the value.
(5) The computer stops putting the command on the bus. The write operation concludes.

Note that the parallel port of the computer should be in low impedance mode when outputting the command, while the PIC should put the data pins in high impedance mode since it is receiving the value.



READ operation
(from computer's perspective)

Figure 4. Read operation on the bus

Figure 4 illustrates the five steps involved in a read operation from the computer (although the timing diagram has the same pattern as Figure 3, there are differences in the steps):

(1) The computer pulls the Strobe signal low to get ready for read operation

(2) The PIC microcontroller on the sensor device outputs a 4-bit value (either MSG_ACK or portion of ADC value) when it sees a low on the strobe line.
(3) The computer raises the Strobe signal and starting reading the value from the data bus. The PIC checks the Strobe signal and learns that the computer has started reading the value.
(4) The computer pulls the Strobe signal low again to indicate that the value has been read.
(5) The PIC microcontroller sees the Strobe pulled low and stops outputting the 4-bit value.

Note that the parallel port should be high impedance mode and the PIC should put its data lines to low impedance mode during the read operation. The read operation can be repeated for multiple times (e.g. three times for obtaining a 10-bit ADC value).

What You Need to Accomplish in this Lab

1. Design a user space application to directly access the parallel port and control the Strobe signals and data bus as described in Figures 3 and 4.
2. Wire up the sensor device with the parallel port following the pin assignment shown in Figures 1 and 2.
3. Design firmware for the PIC microcontroller on the sensor device to support the read and write operations as described in Figures 3 and 4.
4. On the Atom embedded system, display the 10-bit ADC result and 8-byte RTC value obtained through the MSG_GET command.
5. (Optionally) display the commands received on a LCD module

Deliverables

A zipped file containing
1. Schematic of the design (in both native and pdf formats)
2. Source code
3. Reports

References

[1] PIC18F45K20 datasheet.
http://ww1.microchip.com/downloads/en/devicedoc/41303g.pdf
[2] Parallel port pin assignment and programming. Available online.
[3] Peter Anderson, Use of a PC Printer Port for Control and Data Acquisition,
http://et.nmsu.edu/~etti/fall96/computer/printer/printer.html