Chapter 10. Synchronous Serial Communication and Keyboard Connection

1. Synchronous Communication

As its name implies, synchronous communication takes place between a transmitter and a receiver operating on synchronized clocks. In a synchronous system, the communication partners have a short conversation before data exchange begins. In this conversation, they align their clocks and agree upon the parameters of the data transfer, including the time interval between bits of data. Any data that falls outside these parameters will be assumed to be either in error or a placeholder used to maintain synchronization. (Synchronous lines must remain constantly active in order to maintain synchronization, thus the need for placeholders between valid data.) Once each side knows what to expect of the other, and knows how to indicate to the other whether what was expected was received, then communication of any length can commence.

Even though 16F877's USART module provides hardware enabled synchronous master/slave mode of serial communication, we opt to a software enabled approach. It's because the built-in serial port will be connected to a host PC for hex code download. Of course, that same port can be used for other serial device, it would be cumbersome to connect and disconnect a code. Moreover, we will connect another serial device, like a keyboard or mouse, in the example of this chapter, therefore, software approach will give us more freedom of adding additional serial device.

An application of this chapter is to connect a keyboard (eventually two keyboards) and one LCD to the 16F877 in order to let two persons of hearing or speaking disability communicate by typing and reading. The keyboard we are going to connect is the most common type, IBM AT or PS/2 keyboard. The keyboard communicates with PC in synchronous serial communication.

AT type keyboard has 5 pins while PS/2 type keyboard has 6 pins. As illustrated below, for both types of keyboard, there are total 4 signals: +5V power signal, ground, and a CLOCK line, and a DATA line.

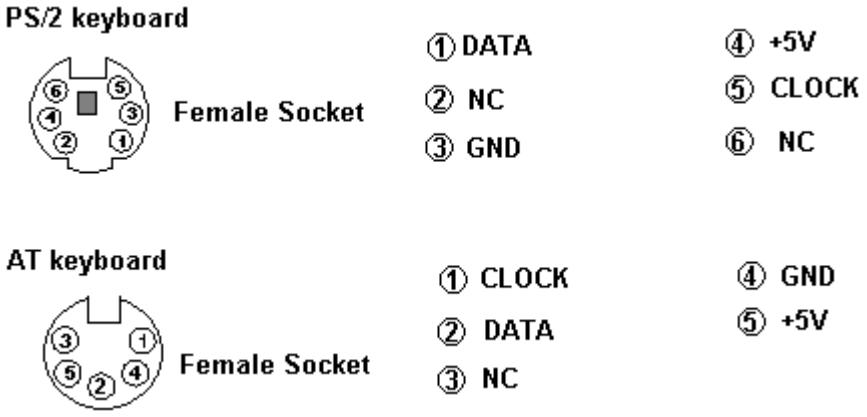

Fig. 75 PS/2 and AT type keyboard

2. IBM AT- or PS/2 – type Keyboard Protocol

The protocol between the keyboard and PC is the most important subject we have to understand in our example application.

The PS/2 mouse and keyboard implement a bidirectional synchronous serial protocol. The bus is "idle" when both lines are high (open-collector). This is the only state where the keyboard/mouse is allowed begin transmitting data. The host has ultimate control over the bus and may inhibit communication at any time by pulling the Clock line low.

The keyboard always generates the clock signal. This is done by a keyboard controlling microcontroller inside the keyboard.

The Data and Clock lines are both open-collector with pull-up resistors to +5V. An "open-collector" interface has two possible state: low, or high impedance. In the "low" state, a transistor pulls the line to ground level. In the "high impedance" state, the interface acts as an open circuit and doesn't drive the line low or high.

The first thing we have to know is how the keyboard controller chip send data to a host (PC or 17F877 in our case). As mentioned above there are two signals from the keyboard: CLOCK and DATA. DATA is sent only when synchronized with the CLOCK. When the keyboard is idle, without any key pressed, both CLOCK and DATA are remained pulled up High. When a key is pressed in the keyboard, both the CLOCK pulse and DATA pulse are transmitted from the keyboard. Through the DATA, strings of byte data are generated. The clock pulses are generated during the data transmission through the CLOCK line.

One thing we have to remember is that a single key stroke does not generate only a byte of data: it generates usually 3 bytes of data and, but other keys generate 5 bytes of data. The list of byte data generated by each individual key is called Keyboard Scan Codes. This discussion follows.

Let's continue our discussion on keyboard protocol. A byte data from the keyboard is sent in a frame consisting of 11 bits. The frame consists, in the following order, of:

1 Start bit (Low),
8-bit data (LSB first, as usual),
1 Odd Parity bit, and
1 Stop bit (High).

The width of the data bit is about 70μs. The frame is synchronized with 11 clock pulses of 70μs with about 40% duty cycle. Namely, the clock pulse's width is 70μs and it's High is about 307μs and its Low for 40μs. As indicated below, a host can sample (or monitor) each bit of the frame at the falling edge of the clock pulse. If we allow a short transition time of High-to-Low change, it would be safe to sample after around 5 μs of the High-to-Low transition of the clock.

To read the frame using 16F877, we need two I/O ports configured as inputs for CLOCK and DATA lines. First we monitor the CLOCK line for transition from High to Low. When it