**Assignment 6 I2C Interface with Wii Nunchuk – CSE 325, Fall 2010**

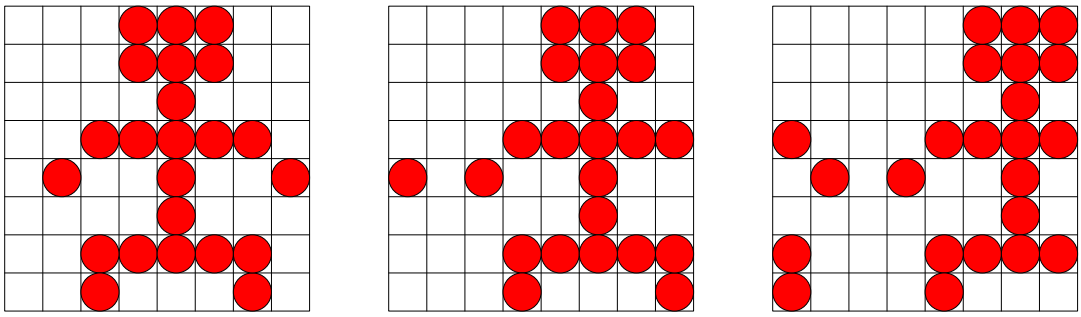
Many embedded applications need to read data from sensors. It is common that an interface driver should be built as a part of execution environment to support the applications. In the assignment, you are given a Wii Nunchuk device which consists of a STMicroelectronics 3-Axis MEMS inertia sensor and an analog joystick to detect physical movement and position. The Nunchuk device has an I2C interface and can report movement and position data upon a request. Your task is to design and implement a driver to read the data from Nunchuk and to use the driver to control the display on the 8X8 LED matrix.

The driver implements a single function *read_Nunchuk(*buf)* for MCF 5211 microprocessor. The function reads 6 byte data from Nunchuk and saves the bytes in the memory location pointed by *buf*. When needed, the *read_Nunchuk* may have to initialize necessary MCF 5211 interface peripherals, IO pins, and Nunchuk before a read operation can be carried out.

The functions should run on the M5211DEMO and the project board. The connections between MCF5211 and the Nunchuk adapter should be wired according to the following table.

| Nunchuk | | MCF5211 |
|---|---|---|
| Data | ←→ | SDA |
| Clock | ← | SCK |
| GND | | GND |
| Vcc | | Vcc |

You will modify your program for Assignment 5 to use the driver for controlling the rotation of pattern displayed on the 8X8 LED matrix. Depending on the position of Nunchuk joystick in +X, -X, +Y, and -Y, the patterns rotate right, left, up, or down respectively. For instance, if the joystick is in +X position, the display is changed from (a) to (b), i.e. the pattern is right-rotated. If the joystick is held in +X position for a while, the display is further changed from (b) to (c). When the Nunchuk buttons Z or C are pressed, the original pattern, i.e., (a), is displayed.

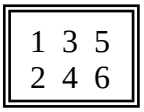


(a) (b) (c)

While your program uses the joystick inputs to control the rotation of the display patterns, delays and thresholds should be added properly to enable reasonable responses. For instance, holding in +X position for a small time interval should not result in two consecutive rotations. Also, a small +X move may not be considered as an input for a right rotation.

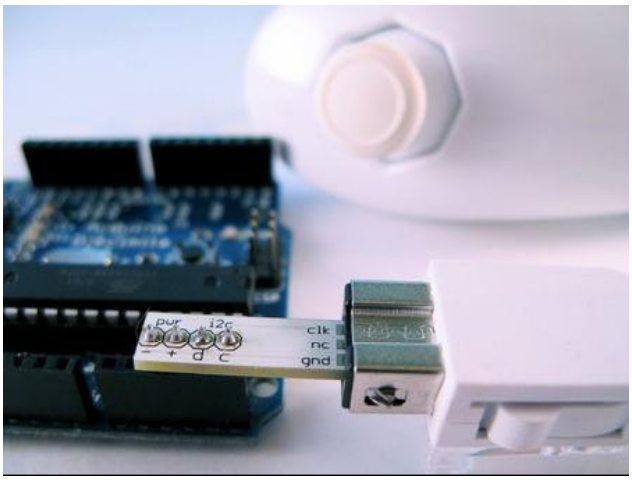
## Wii Nunchuk Interface:

The Wii Nunchuk uses a proprietary connector. The connector front end looks like below. The connections are as follows:




1 = +3V (3V recommended but works at 5V)
2 = Clock
3 = N/A
4 = N/A
5 = Data
6 = Gnd

An adapter is used to connect the Wii Nunchuk to any processor which is as shown.




The order of signals from left to right from the image above is ( Gnd – Vcc – Data – Clock).

## Frequency:

The frequency used to communicate with Nunchuk is 100KHz.

## Handshaking/Initialization/Read operation:

To communicate with the Nunchuk, we must send a handshake signal. So first send 2 bytes "0x40, 0x00" to initialize the Nunchuk. Then send one byte "0x00" each time you request data from the Nunchuck. The data from the Nunchuk will come back in 6 byte chunks.

## I2C Address:

The I2C slave address of Nunchuk is 0x52.

## Default Data:

The default values (6 bytes) expected from Nunchuk are as follows. For X- and Y-axis values in the first rows, Min (Full Left) means the analog stick of the Wii Nunchuk is the to the extreme left along the axis. Medium (Center) means the stick is the default position and Max (Full right) means the stick is in the extreme right position.

| Value in example | Description | Values of sample Nunchuk |
|---|---|---|
| 0x7E | X-axis value of the analog stick | Min(Full Left):0x1E / Medium(Center):0x7E / Max(Full Right):0xE1 |
| 0x7B | Y-axis value of the analog stick | Min(Full Down):0x1D / Medium(Center):0x7B / Max(Full Right):0xDF |
| 0xAF | X-axis acceleration value | Min(at 1G):0x48 / Medium(at 1G):0x7D / Max(at 1G):0xB0 |
| 0x80 | Y-axis acceleration value | Min(at 1G):0x46 / Medium(at 1G):0x7A / Max(at 1G):0xAF |
| 0x7A | Z-axis acceleration value | Min(at 1G):0x4A / Medium(at 1G):0x7E / Max(at 1G):0xB1 |
| 0x3B | Button state (Bits 0/1) / acceleration LSB | Bit 0: "Z"-Button (0 = pressed, 1 = released) / Bit 1: "C" button (0 = pressed, 1 = released) / Bits 2-3: X acceleration LSB / Bits 4-5: Y acceleration LSB / Bits 6-7: Z acceleration |

When you read Nunchuk data, there is a coding scheme on the data. To decode the data, please calculate the data as:

Exact data = (Reading data XOR 0x17) + 0x17

Also the value for pressing/releasing buttons should be

```
case 0:  // Z is pressed        00
case 1:  // C is pressed        01
case 2:  // Both C and Z        10
case 3:  // Neither             11
```

Here is a pretty good resource on how to decode. http://www.robotshop.ca/content/PDF/inex-zx-nunchuck-datasheet.pdf