Práctica 13a

Objetivo

Demostrar el conocimiento necesario para implementar y ejecutar un código en lenguaje c y ensamblador, así mismo acceder a los puertos de la tarjeta T-Juino por medio de las bibliotecas.

Desarrollo

Realice lo siguiente.

1. Instalar arduino

   https://www.arduino.cc/en/software
   https://downloads.arduino.cc/arduino-1.8.19-linux64.tar.xz

   sudo ./install.sh

2. Cargar el archivo ppi8255.hex a la placa T-Juino

   Desde Windows
   C:\Program Files (x86)\Arduino\hardware\tools\avr/bin/avrdude -CC:\Program Files (x86)\Arduino\hardware\tools\avr/etc/avrdude.conf -v -patmega1280 -carduino -PCOM10 -b57600 -D -Uflash:w:C:\Users\"NOMBRE DE USUARIO DE COMPUTADORA"\AppData\Local\Temp\arduino_build_692634/sketch_may08a.ino.hex:i

   Desde Linux lite
   sudo /home/linuxlite/Downloads/arduino-1.8.19/hardware/tools/avr/bin/avrdude -C/home/linuxlite/Downloads/arduino-1.8.19/hardware/tools/avr/etc/avrdude.conf -v -patmega1280 -carduino -P/dev/ttyUSB0 -b57600 -D -Uflash:w:/home/linuxlite/Desktop/Folder_compartido_linux_lite/OAC_Ports/ppi8255.hex:i

3. Tomando de ejemplo el archivo test_ppi.c con la placa T-Juino conectada y corriendo.

   Compilarlo
   gcc -m32 -o test_ppi test_ppi.c libppi.a
   Ejecutarlo
   sudo ./test_ppi

4. Poner en 0 al bit 0 del puerto A y ver el resultado en el puerto B.

5. Agregar captura de pantalla del resultado de la ejecución impreso en terminal.

6. Apoyándose en el ejemplo anterior y en el archivo P13.asm de la práctica 13a crear un archivo P13bC.c donde estarán las funciones y el código necesario para hacer lo siguiente:

a) **setBitPort(dato,numBit,puerto)**: utiliza _setBit en un dato, modificando el bit señalado por numBit, y saca al puerto ese dato modificado para posteriormente leer ese puerto y retornar esa lectura.

b) **clearBitPort(dato,numBit,puerto)**: utiliza _clearBit en un dato, modificando el bit señalado por numBit, y saca al puerto ese dato modificado para posteriormente leer ese puerto y retornar esa lectura.

c) **notBitPort(dato,numBit,puerto)**: utiliza _notBit en un dato, modificando el bit señalado por numBit, y saca al puerto ese dato modificado para posteriormente leer ese puerto y retornar esa lectura.

d) **testBitPort(numBit,puerto)**: utiliza _testBit leyendo el bit señalado por numBit en un dato leído del puerto y retorna esa lectura.

e) Para probar el funcionamiento se deberá hacer lo siguiente:

1. Comenzar con un dato que será 0B3h el cual es leído del puerto A.
2. Posteriormente activar el bit 3, sacarlo al puerto B y retornarlo para imprimirlo.
3. El dato resultante del paso 2 desactivarle los bits 1, 4 y 7, sacarlo al puerto B y retornarlo para imprimirlo.
4. El dato resultante del paso 3 invertirle los bits 2, 3, 4, 5 y 7, sacarlo al puerto B y retornarlo para imprimirlo.
5. El dato resultante del paso 4 revisar el estado de los bits 1, 4 y 7, sacarlo al puerto B y retornarlo para imprimirlo.

f) Mostrar en terminar como hexadecimal cada cambio que se va realizando por cada modificación de bit y con leds en la salida del puerto B.

Conclusiones y comentarios Dificultades en el desarrollo Referencias