Enginyeria de Sistemes TIC

Dispositius Programables

Pràctica 5 - Comunicació Serie

Francisco del Águila López

Octubre 2012
Escola Politècnica Superior d'Enginyeria de Manresa
Universitat politècnica de Catalunya

1 Objectiu

L'objectiu d'aquesta pràctica és permetre la comunicació entre l'Arduino i l'ordinador personal.

2 Introducció teòrica

2.1 Comunicació serie asíncrona

Una de les formes que tenen els computadors per comunicar-se els uns amb els altres és amb un perifèric anomenat **USART** [USART]. Aquest perifèric permet una comunicació de tipus serie, per tant fa servir dues línies de comunicació, una per a la recepció i l'altra per a la transmissió. Aquest tipus de comunicació generalment es fa en mode asíncron, per tant no hi ha cap tipus de rellotge que reguli la mateixa comunicació. En una comunicació serie, la informació en forma de bits viatgen un darrera de l'altra, per aquest motiu la quantitat de línies és mínima.

En el cas de l'Arduino la interfície de comunicació USART que es fa servir per comunicar amb l'ordinador personal està formada per la mateixa interfície USB. El port USB queda configurat emulant el comportament de una USART per tant, no tenim la necessitat d'interconnectar l'Arduino amb l'ordinador amb cap altre cable més que el mateix de l'USB que fem servir per alimentar i per programar.

2.2 Programari de comunicació

L'ordinador que es connecta amb l'Arduino necessita d'un terminal que permeti la interacció entre l'usuari i el dispositiu amb el que es comunica. La funció d'aquest terminal és permetre que tot el que l'usuari tecleja sigui transmès al dispositiu connectat i també permet que allò que el dispositiu envia cap a l'ordinador pugui ser visualitzat per la pantalla. Una eina d'aquest tipus podria ser **picocom**.

Per instal·lar aquesta aplicació en una distribució tipus Debian s'ha d'executar la comanda

```
sudo aptitude install picocom
```

Per saber com funciona aquesta aplicació es pot fer ús del seu manual: *man picocom*.

2.3 Mòdul USART

La majoria d'AVR, i en particular l'ATmega328p que fem servir a la plataforma Arduino conté un mòdul USART configurable de diferents maneres. La descripció detallada d'aquest mòdul es troba a l'apartat 20, pàg 178 del manual de referència de l'ATmega328p [ATmega328p].

La configuració que es fa servir per a la comunicació amb l'ordinador és en mode asíncron, amb un rellotge intern i sense habilitar el mode de multiprocessador.

La velocitat a la que poden anar els bits, al igual que el format del missatge són paràmetres configurables tant per part de l'Arduino com per part de l'ordinador. La comunicació serà possible si els dos dispositius tenen la mateixa configuració. En general aquesta comunicació per defecte queda definida per 8 bits de dades, 1 bit de parada i sense paritat. Per tant aquesta serà la configuració que es farà servir tant a l'Arduino com en l'ordinador en cas que no es digui el contrari.

2.4 Codificació ASCII

Un estàndard molt simple per poder codificar en binari els caràcters d'un text és la codificació [ASCII]. Aquesta codificació en el seu format més simple consisteix en una taula de 7 bits. Amb 7 bits només es poden codificar 128 possibles valors per tant el conjunt de possibles caràcters és bastant reduït. Ja que la majoria de sistemes digitals treballen amb unitats de 8 bits (1Byte), la taula ASCII també està definida amb 8 bits. Això augmenta el nombre de caràcters al doble (256 valors), quedant definida la taula ASCII estesa. El sistemes Linux es pot fer servir la comanda *man asccii* per veure els valors.

La codificació binària que es fa servir per transmetre un caràcter a través d'una comunicació basada en un dispositiu USART és la codificació ASCII. Per exemple, la transmissió de la lletra **A** queda codificada amb els bits **0x41**.

3 Exemple de programa

El següent programa, exemple.S, implementa una comunicació serie de manera que l'Arduino es manté a l'espera fin que rep un byte. Quan rep aquest byte simplement es limita a tornar-lo a transmetre. Aquest tipus de comportament és el que s'anomena fer un eco.

```
.set DDRB_o , 0x4
.equ PORTB_o , 0x5
PORTD_o = 0x0b
DDRD_o = 0x0a

UDR0 = 0xc6
UBRR0H = 0xc5
UBRR0L = 0xc4
UCSR0C = Oxc2
UCSR0B = 0xC1
UCSR0A = 0xC0

.global main

/* rutina de recepció de bytes, el valor es recull al registre r16 */
rx:     lds     r16,UCSR0A
        sbrs    r16,7
        rjmp    rx
        lds     r16,UDR0
        ret

/* rutina de transmissió de byte, el valor a transmetre està al registre r16 *
tx:     lds     r17,UCSR0A
        sbrs    r17,5
        rjmp    tx
        sts     UDR0,r16
        ret

main:
        /* set baud rate a 9600*/
        ldi     r16, 0
        sts     UBRR0H,r16
        ldi     r16, 103
        sts     UBRR0L,r16

        /* set frame format */
        /* el valor de reset dels registres ja és correcte:
asíncron, sense paritat, 1 stop, 8 dades,
```

```
velocitat normal, comunicació no multiprocessor
però assegurem el que volem*/
        ldi     r16, 0b00100000
        sts     UCSR0A,r16

        ldi     r16, 0b00000110
        sts     UCSR0C,r16

        /* enable rx, tx, sense interrupcions */
        ldi     r16, 0b00011000
        sts     UCSR0B,r16

        /* configuració dels pins */
        ldi     r16,0b00000010
        out     DDRD_o,r16

loop:
        call    rx
        call    tx
        rjmp    loop
```

4 Estudi previ

1. Llegiu detalladament l'apartat 20 de [ATmega328p]
2. Descriviu detalladament què fa exemple.S
3. Modifiqueu *exemple.S* de manera que la comunicació quedi definida a una velocitat de 115kbps, amb paritat, 1 bit de parada, 7 bits de dades i el mode de velocitat normal (no doble).
4. Fes un programa que quan detecti que s'ha polsat la lletra l o L s'encengui el led de l'Arduino i quan rebi qualsevol altre dada s'apagui. Fes servir la configuració de velocitat, bits de dades, paritat i stop del programa *exemple.S*. Anomena'l *prac5_1.S*.
5. Fes un programa que quan detecti que s'ha polsat la lletra n o N respongui amb els caràcters corresponents als nombres 0,1,2,3,4,5,6,7,8,9. Quan es rebi qualsevol altre valor, respongui amb el caràcter N. Anomena'l *prac5_2.S*.
6. Fes un programa que quan detecti exactament la seqüència de lletres *"led"* encengui el led, amb qualsevol altre seqüència el led s'ha de mantenir apagat. La resposta de l'Arduino cap al ordinador a cada pulsació hauria de ser el nombre de lletres que queden per teclejar fins aconseguir la seqüència *"led"*. Implementeu aquest programa com una màquina d'estats. Dibuixeu el graf corresponent, definiu els

diferents estats, declareu qui mantindrà el valor de l'estat del sistema. Anomena'l *prac5_ 3.S.*

5 Treball pràctic

El treball al laboratori consisteix en la comprovació pràctica de les tasques de l'estudi previ.

5.1 Assemblat del codi font

Per assemblar el nostre codi font s'ha de cridar a la comanda

```
avr-gcc -mmcu=atmega328p -o prova.elf prova.S
```

on el paràmetre **-mmcu** indica el model de microcontrolador que estem fent servir, **-o** indica el fitxer final generat (format *elf*) i l'últim paràmetre és directament el fitxer amb el codi font.

Un paràmetre que pot ser d'utilitat és **-E**, en aquest cas es realitza l'assemblat estrictament. Pot ser d'utilitat per analitzar possibles errors.

```
avr-gcc -mmcu=atmega328p -o prova.asm -E prova.S
```

Per convertir el format *elf* a *ihex*, la comanda és

```
avr-objcopy --output-target=ihex prova.elf prova.hex
```

Una possibilitat molt important és el procés de des-assemblat. En aquest cas, a partir d'un fitxer executable de tipus *elf* s'obté un fitxer en codi font. Òbviament la informació extra que conté el codi font original ha desaparegut. La comanda és

```
avr-objdump -S prova.elf > prova.disam
```

5.2 Transferència dels fitxers executables (Avrdude)

Quan es connecta l'Arduino al ordinador pel port USB, s'ha de comprovar que s'ha detectat el port de comunicació i per tant està reconegut pel sistema. Això es pot comprovar amb l'ordre *dmesg*. A les línies finals hauria d'aparèixer un nou dispositiu amb identificació **ttyACM0.**

Per comprovar que l'Arduino està operatiu i disposat a acceptar ordres, la comanda és

```
avrdude -c arduino -P /dev/ttyACM0 -p m328p
```

on els paràmetres indiquen que el tipus de programador que fa servir l'*avrdude* és el que inclou directament el kit Arduino, que el port de comunicacions és el nou port USB detectat i que el dispositiu amb el que el treballa és un ATmega328p. Aquests paràmetres es poden consultar en el manual *man avrdude*.

Per fer les transferències dels fitxers binaris es fa servir l'opció **-U**(consulteu el manual). Els paràmetres d'aquesta opció són la memòria a la que s'accedeix, si es fa lectura o escriptura, el fitxer que es vol transferir, el format del fitxer. Un exemple per llegit la memòria de programa i crear un fitxer Intel Hex amb el seu contingut seria

```
avrdude -c arduino -P /dev/ttyACM0 -p m328p -U flash:r:prova.hex:i
```

5.3 Tasques

1. Assembla *exemple.S* i comprova el seu funcionament.
2. Comprova el correcte funcionament de *prac5_1.S*.
3. Comprova el correcte funcionament de *prac5_2.S*.
4. Comprova el correcte funcionament de *prac5_3.S*.

Referències

[Ard] Arduino UNO - http://arduino.cc/en/Main/ArduinoBoardUno

[ATmega328p] Atmel. ATmega328P datasheet. http://www.atmel.com/dyn/resources/prod_documents/doc8271.pdf

[AtmAss] Atmel Assembler documentation - http://www.atmel.com/atmel/acrobat/doc1022.pdf

[Instr] Joc d'instruccions dels AVR - http://www.atmel.com/atmel/acrobat/doc0856.pdf

[AS] Manual de referència del Assemblador del GNU - http://sourceware.org/binutils/docs/as/

[ihex] Format de fitxer intel HEX - http://en.wikipedia.org/wiki/Intel_HEX

[Hardvard] Arquitectura de tipus Hardvard - http://en.wikipedia.org/wiki/Harvard_architecture

[Endian] Ordre de disposició dels bytes - http://en.wikipedia.org/wiki/Endianness

[USART] Dispositiu USART http://en.wikipedia.org/wiki/Universal_asynchronous_receiver/transmitter

[ASCII] Taula de caràcters ASCII http://en.wikipedia.org/wiki/ASCII