UNIVERSITAT POLITÈCNICA DE CATALUNYA
BARCELONATECH
Enginyeria de Sistemes TIC

Interrupcions

Dispositius Programables — Enginyeria de Sistemes TIC

Jordi Bonet Francisco del Águila

27 d'octubre de 2023

Índex

1 Objectiu **1**

2 Introducció teòrica **1**
2.1 Interrupcions a l'AVR 1

3 Exemple de programa **2**

4 Estudi previ **3**

5 Treball pràctic **4**
5.1 Tasques 4

1 Objectiu

L'objectiu d'aquesta pràctica és utilitzar les interrupcions del port sèrie. Cal implementar unes tasques similars a les de la pràctica anterior, però ara usant les interrupcions. Es continua requerint l'ús d'un autòmat en alguna de les tasques i continua essent d'obligat compliment haver realitzat el **diagrama d'estats** abans d'implementar l'autòmat (escriure codi).

2 Introducció teòrica

2.1 Interrupcions a l'AVR

La descripció detallada del funcionament de les interrupcions es troba als apartats 7.7 i 12 de [ATmega328p]. El concepte d'interrupció es pot entendre com una crida a una subrutina (en aquest cas rutina d'interrupció) amb la diferència que qui fa la crida no és una instrucció *call*, o altres similars com *icall* o *rcall*, que es troba dins el programa principal, sinó que és el propi maquinari perifèric de la CPU qui fa la crida a la rutina d'interrupció.

Les interrupcions es poden habilitar o deshabilitar de forma global, amb efecte sobre totes les interrupcions, siguin del tipus que siguin. Per a habilitar de forma global usem la instrucció *sei*, i per a deshabilitar de forma global *cli*. També podem habilitar o deshabilitar només algunes interrupcions de forma local. En aquest cas no disposem d'instruccions, sinó que hem de configurar alguns dels registres del mapa d'entrada / sortida associats a les interrupcions de cada mòdul. Per tal que una determinada interrupció estigui habilitada és necessari que estigui habilitada globalment i localment.

Per tal d'evitar que es produeixin interrupcions quan encara no s'ha realitzat la configuració de tot el microcontrolador cal seguir una metodologia. En arrencar el microcontrolador, les interrupcions estan globalment deshabilitades. Per tant, mentre es realitzen totes les configuracions inicials, podem assegurar que no es produirà cap interrupció. Quan hem acabat la configuració inicial, executem la instrucció *sei*. A partir d'aquest moment es poden produir interrupcions (de fet si consultem [Instr] veurem que és a partir de la instrucció que segueix *sei*).

La manera de tractar amb rutines d'interrupció en assemblador és la següent. Cal usar el símbol especial *___vector_XX*, on *XX* és la posició del vector d'interrupció que apareix a l'apartat 12.4 de [ATmega328p] [1]. Així, si considerem la interrupció de recepció del mòdul USART, usarem *___vector_18*. Aquests símbols s'han de definir com a globals, amb la directiva (*.global*, per tal que siguin visibles per al *linker*. Observeu en el codi d'exemple que el símbol especial **main**, que és on comença el nostre programa, també es defineix com a global.

3 Exemple de programa

El següent programa, *p6-exemple.s*, implementa una comunicació serie de manera que l'Arduino, un cop s'ha inicialitzat tot correctament, es manté sense fer res en un bucle infinit. Quan es produeix la recepció d'un byte, es genera una interrupció que fa que s'executi la rutina d'interrupció d'atenció a la interrupció. En aquesta rutina es llegeix el byte i es realitza una transmissió d'aquest byte, de manera que es retorna el mateix byte cap al PC. En el terminal visualitzarem el mateix byte transmès. Aquest tipus de comportament és el que s'anomena fer un eco.

```
DDRB_o = 0x4
PORTB_o = 0x5
DDRD_o = 0x0a
PORTD_o = 0x0b

UCSR0A = 0xc0
UCSR0B = 0xc1
UCSR0C = 0xc2
UBRR0L = 0xc4
UBRR0H = 0xc5
UDR0 = 0xc6

.global main
.global ___vector_18

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

/* defineixo la rutina d'interrupció
per recepció de byte a la USART */
```

[1] Observeu que la posició **0** correspon al **RESET**

```
__vector_18:
    lds r16,UDR0
    call tx
    reti

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

    /* set frame format */
    /* tot i que el valor dels registres després d'un reset ja és correcte
    (asíncron, 8 bits de dades, 1 bit de parada, sense paritat,
    velocitat normal, comunicació no multiprocessor)
    assegurem aquesta configuració escrivint el valor als registres*/
    ldi r16, 0b00100000
    sts UCSR0A, r16

    ldi r16, 0b00000110
    sts UCSR0C, r16

    /* enable rx, tx, amb interrupció de rx */
    ldi r16, 0b10011000
    sts UCSR0B,r16

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

    /* habilitem interrupcions globals */
    sei

/* el bucle principal no fa res */
loop:
    rjmp loop
    ret
```

4 Estudi previ

TASCA PRÈVIA 1

Llegiu detalladament l'apartat 7.7 i 12 de [ATmega328p]

TASCA PRÈVIA 2 Assembleu *p6-exemple.s* i a partir del fitxer *p6-exemple.elf* desassembleu el fitxer generant *p6-exemple.disam*. Descriu detalladament tot el codi que apareix justificant exactament que fa cada grup d'instruccions màquina.

TASCA PRÈVIA 3 Feu un programa que quan detecti exactament la seqüència de lletres *led* encengui el LED i quan rebi qualsevol altre seqüència l'apagui. 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: definiu els estats, dibuixeu el diagrama d'estats i declareu qui mantindrà el valor de l'estat del sistema. *L'esdeveniment que fa canviar d'estat ha de ser forçosament la recepció d'un byte.* Anomeneu aquest fitxer *p6-codi3.s*.

TASCA PRÈVIA 4 Quin avantatge pot tenir usar la interrupció de transmissió tenint en compte que el propi microcontrolador sap en quin moment vol transmetre?

TASCA PRÈVIA 5 Feu 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*.

Implementeu aquest programa com una màquina d'estats: definiu els estats, dibuixeu el diagrama d'estats i declareu qui mantindrà el valor de l'estat del sistema. *Els esdeveniments que fan canviar d'estat han de ser forçosament la recepció d'un byte i/o la finalització de transmissió d'un byte.* Anomeneu aquest fitxer *p6-codi2.s*.

5 Treball pràctic

El treball al laboratori consisteix en la comprovació pràctica de les tasques de l'estudi previ. A continuació teniu algunes comandes del *Toolchain* de GNU, imprescindibles per passar els codis al microcontrolador.

```
avr-gcc -mmcu=atmega328p -o a.elf a.s

avr-objcopy --output-target=ihex a.elf a.hex

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

5.1 Tasques

TASCA 6 Assembleu *p6-exemple.s* i comproveu el seu funcionament.

TASCA 7 Comproveu el correcte funcionament de *p6-codi3.s*.

TASCA 8 Comproveu el correcte funcionament de *p6-codi2.s*.

Referències

[ATmega328p] Atmel. ATmega328P datasheet; `https://ww1.microchip.com/downloads/en/DeviceDoc/ATmega48A-PA-88A-PA-168A-PA-328-P-DS-DS40002061B.pdf`

[Instr] Joc d'instruccions dels AVR; `http://ww1.microchip.com/downloads/en/devicedoc/atmel-0856-avr-instruction-set-manual.pdf`