UNIVERSITAT POLITÈCNICA DE CATALUNYA
BARCELONATECH
Enginyeria de Sistemes TIC

Generador de Morse

Dispositius Programables — Enginyeria de Sistemes TIC

Jordi Bonet Francisco del Águila

14 de novembre de 2022

Índex

1 Objectiu

L'objectiu d'aquesta pràctica és dissenyar un transmissor de codi Morse.

2 Consideracions prèvies

2.1 Components

Els elements que intervenen són:

1. Plataforma Arduino.
2. Placa *protoboard*
3. Trossos de cable prim per interconnectar la *protoboard* i l'Arduino.
4. 2 polsadors.
5. Sondes de voltatge, per tal de visualitzar els diferents senyals a l'oscil · loscopi.
6. També es podria usar un sistema amplificador, amb altaveus, per tal d'escoltar el senyal generat, però això no ens permet assegurar que generem un codi Morse correcte. Sempre caldrà visualitzar els senyals a l'oscil·loscopi per assegurar que la durada dels senyals és correcta.

Un cop al laboratori, us proporcionarem dos polsadors per grup.

2.2 Morse

Una descripció simple del codi Morse la teniu aprofitant la pràctica que heu fet a l'assignatura de Circuits i Sistemes Lineals. Essencialment podem recordar que el codi Morse internacional defineix cinc elements:

- El punt, que té una durada d'una unitat de temps. Usarem el símbol *1*.
- La ratlla, amb una durada de tres unitats de temps, és a dir, llargada triple que el punt. Usarem el símbol *111*.
- La pausa (o silenci) entre elements (punt o ratlla) d'un caràcter (lletra o nombre), amb una durada d'una unitat de temps. Usarem el símbol *0*.
- La pausa entre caràcters, amb una durada de tres unitats de temps. Usarem el símbol *000*.
- La pausa entre paraules, amb una durada de set unitats de temps. Usarem el símbol *0000000*.

2.3 Estructuració del programa

El disseny amb dos polsadors es basa en què un polsador genera un *punt* i l'altre una *ratlla*. D'aquesta manera la seqüència de pulsacions en cadascun dels polsadors determinarà la seqüència de *punts* i *ratlles*.

Una possible estructura del programa consisteix en una fase d'inicialització dels registres o variables que ens calguin, seguida d'un bucle principal encarregat de detectar quin dels polsadors s'ha premut. En funció del polsador premut cal fer una crida a la subrutina que genera el *punt* o bé a la que genera la *ratlla*.

En endavant considerem la *unitat de temps* de durada $t_u = 150\,\text{ms}$. La subrutina **punt** ha de consistir en la generació d'un to d'1 kHz durant t_u seguit d'una pausa de durada t_u. la subrutina **ratlla** ha de consistir en la generació d'un to d'1 kHz durant $3t_u$ seguit d'una pausa de durada t_u. La implementació de les pauses entre caràcters, de durada $3t_u$, i entre paraules, de durada $7t_u$, les deixem a càrrec de l'operador que actua sobre els polsadors.

El codi del fitxer *p4-exemple.s*, que apareix a continuació, us pot ser útil per generar aquest programa.

```
.set DDRB_o , 0x4
.equ PORTB_o , 0x5
DDRD_o = 0x0a
PORTD_o = 0x0b
TCCR0A_o = 0x24
TCCR0B_o = 0x25
OCR0A_o = 0x27

.global main

waitabit: ldi r19,41
wait3: ldi r18,0xFF
wait2: ldi r17,0xFF
wait1: subi r17,0x01
    brne wait1
```

```
        subi r18,0x01
        brne wait2
        subi r19,0x01
        brne wait3
        ret

main:   ldi r16,0b01000000
        out DDRD_o,r16
        ldi r16,0b00000000
        out PORTD_o,r16
        ldi r16,124
        out OCR0A_o,r16
        ldi r16,0b01000010
        out TCCR0A_o,r16
        ldi r16,0b00000011
        out TCCR0B_o,r16

loop:   call waitabit
        in r20,TCCR0A_o
        cbr r20,0b01000000
        out TCCR0A_o,r20
        call waitabit
        in r20,TCCR0A_o
        sbr r20,0b01000000
        out TCCR0A_o,r20
        rjmp loop
```

Aquest codi utilitza el perifèric *timer_0* dels AVR. La informació detallada d'aquest perifèric es troba a l'apartat 15 d'[ATmega328p]. Aquest perifèric es pot configurar de moltes maneres. Té dos modes de generació PWM, un mode de comptador simple i un mode comparador.

Tingueu en compte que us pot ajudar molt el fet de veure el resum de registres de entrada / sortida que intervenen en la configuració de qualsevol perifèric. Aquest resum es troba al final del capítol on es descriu cada perifèric.

Aquest perifèric, com pràcticament la totalitat de la resta, també es pot fer servir per a generar interrupcions. De moment no farem cap utilització de les interrupcions, per tant, podeu excloure les explicacions relacionades amb les interrupcions, així com els registres associats a les interrupcions: *TIMSK0* i *TIFR0*.

3 Estudi previ

En cap dels següents apartats podeu usar el perifèric *timer_0*.

TASCA PRÈVIA 1 Escriviu una subrutina de nom **to1** que generi un to d'1 kHz durant t_u.

TASCA PRÈVIA 2 Escriviu una subrutina de nom **sl1** que generi una pausa de t_u.

TASCA PRÈVIA 3 Escriviu una subrutina de nom **punt** que, aprofitant les anterior subrutines **to1** i **sl1**, generi el *punt* de Morse.

TASCA PRÈVIA 4 Escriviu una subrutina de nom **ratlla** que, aprofitant les anterior subrutines **to1** i **sl1**, generi la *ratlla* de Morse.

TASCA PRÈVIA 5 Escriviu en el fitxer *p4-codi1.s* el programa de codi Morse que generi un *punt* o una *ratlla* en funció del polsador premut. Definiu els pins d'entrada dels dos polsadors i el pin de sortida on es generarà el senyal de Morse.

En els següents apartats ha d'intervenir el perifèric *timer_0*.

TASCA PRÈVIA 6 Inseriu en el fitxer *p4-exemple.s* els comentaris necessaris per tal d'explicar amb tot detall què fan les línies del codi.

TASCA PRÈVIA 7 Modifiqueu el fitxer *p4-codi1.s* aprofitant els recursos que s'usen en el fitxer *p4-exemple.s*. Si cal, re-definiu els pins. Anomeneu el fitxer modificat *p4-codi2.s*.

TASCA PRÈVIA 8 Penseu com es podria detectar en qualsevol instant la situació anòmala en què els dos polsadors estan simultàniament premuts. Modifiqueu el fitxer *p4-codi2.s* per tal que quan es produeixi aquesta simultaneïtat no s'enviï cap codi Morse i s'indiqui que es tracta d'una anomalia encenent el LED de la placa Arduino. Anomeneu el fitxer modificat *p4-codi3.s*.

4 Treball pràctic

El treball al laboratori consisteix en la comprovació 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
```

4.1 Tasques

La verificació dels codis es realitzarà observant els senyals a l'oscil·loscopi. Cal fer mesures de totes les durades significatives del senyal.

TASCA 9 Verifiqueu que heu comprès què fa el codi del *p4-exemple.s*.

TASCA 10 Comproveu el correcte funcionament de *p4-codi1.s*.

TASCA 11 Comproveu el correcte funcionament de *p4-codi2.s*.

TASCA 12 Comproveu el correcte funcionament de *p4-codi3.s*.

Referències

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