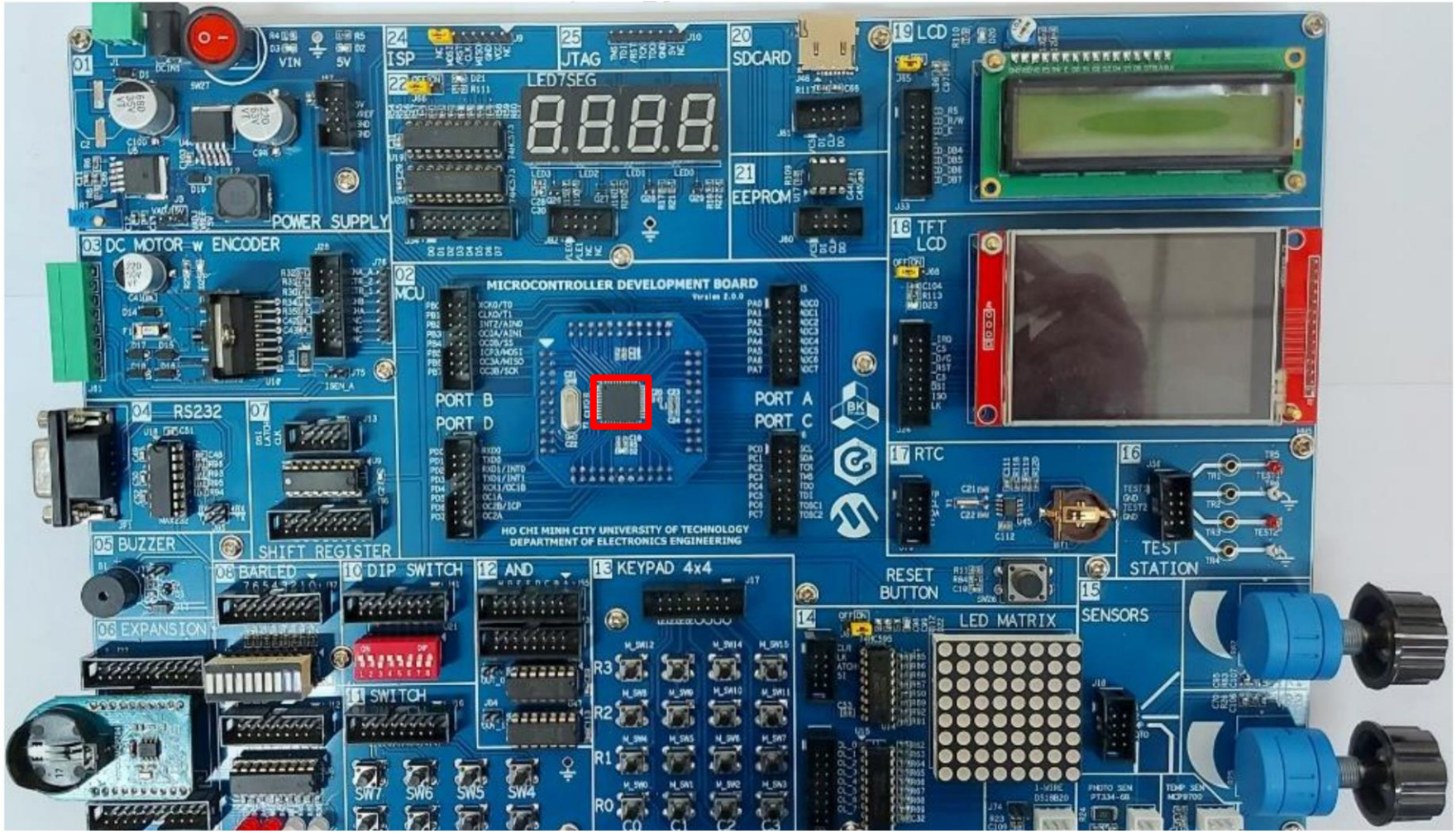
Chương 6
Giao tiếp ngoại vi

Tài liệu tham khảo:

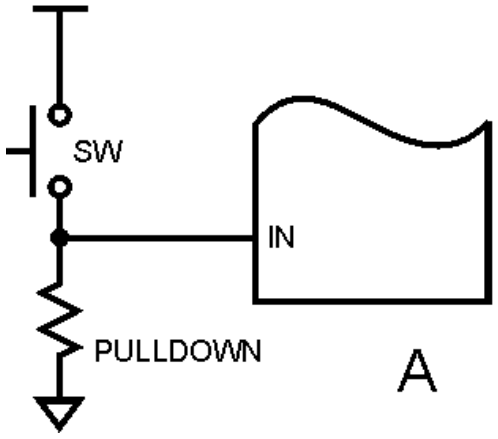
1. Chương 6: Giao tiếp ngoại vi (Giáo trình VXL)
2. Muhammad Ali Mazidi, *AVR Microcontroller and Embedded Systems: Using Assembly and C, Pearson New International Edition*, 2014.
3. Datasheet ATmega324P
4. https://nicerland.com/avr/
5. http://www.hocavr.com/
6. https://www.youtube.com/watch?v=Fr2K9pzec8g&list=PLgwJf8NK-2e55CdbY_WnY6pejPHoojCkJ

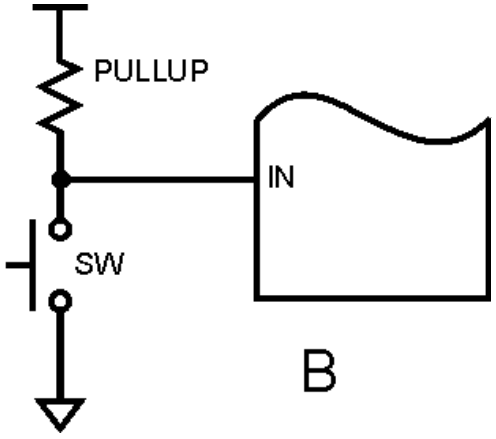
Kit thí nghiệm – ATmega324PA

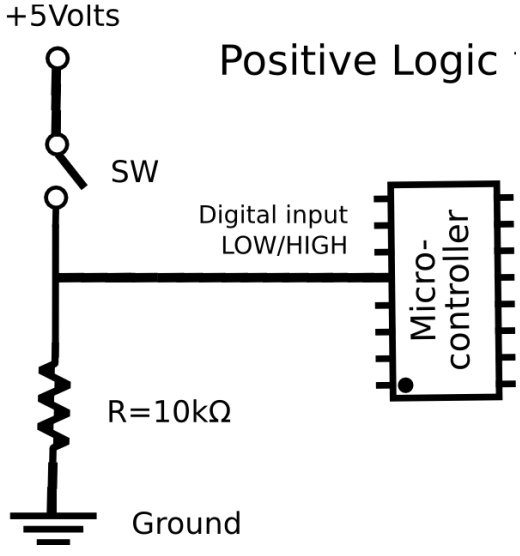


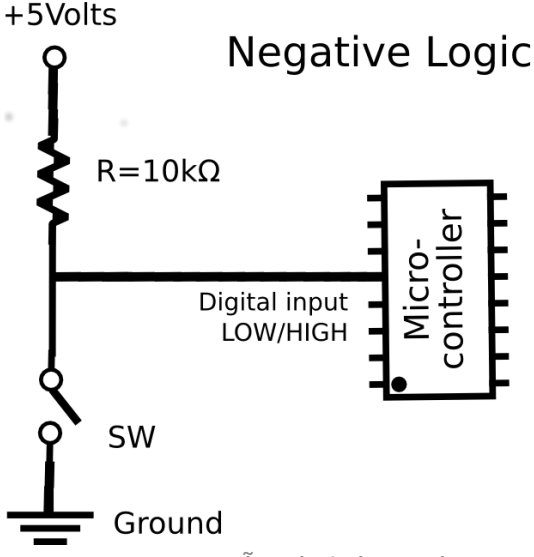
6.1 Giao tiếp với nút nhấn, LED đơn
6.2 Giao tiếp với bàn phím ma trận (Keypad)
6.3 Giao tiếp với LED 7 đoạn
6.4 Giao tiếp với LCD (Liquid Crystal Display) 16x2
6.5 Giao tiếp IC đệm 3 trạng thái 74HC244
6.6 Giao tiếp với thanh ghi dịch 74HC595

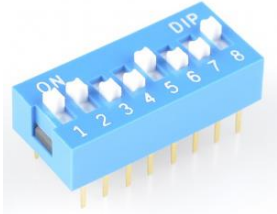
6.1 Một số dạng kết nối với nút nhấn, Switch





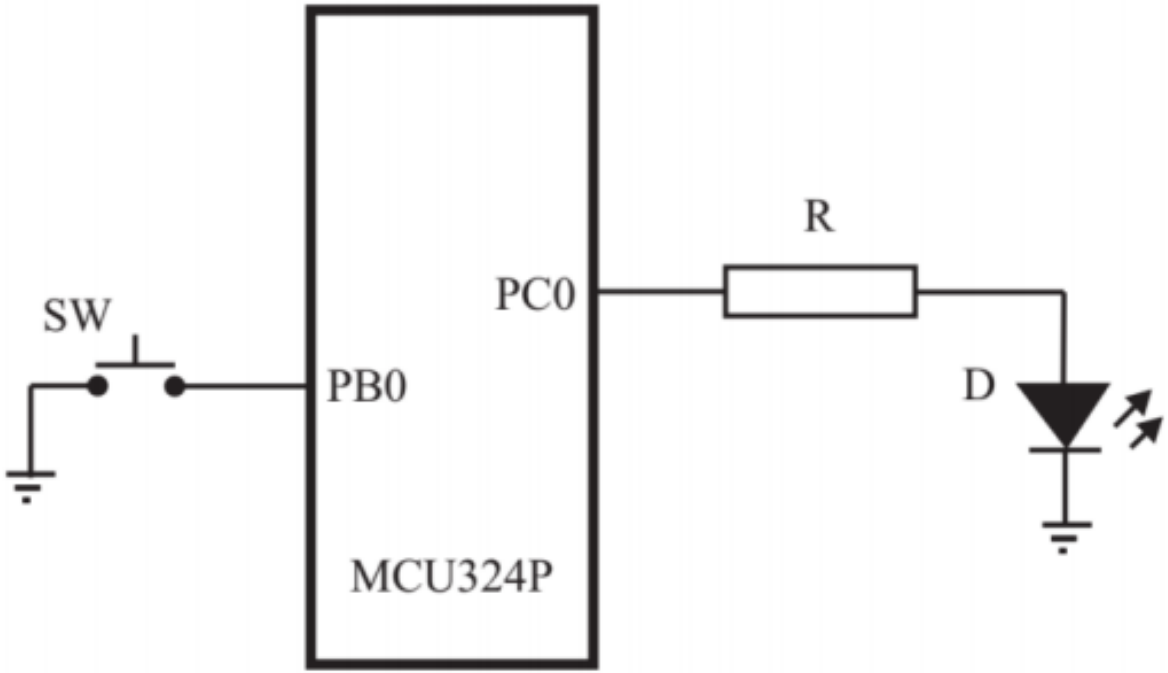






6.1 Giao tiếp với nút nhấn, LED đơn

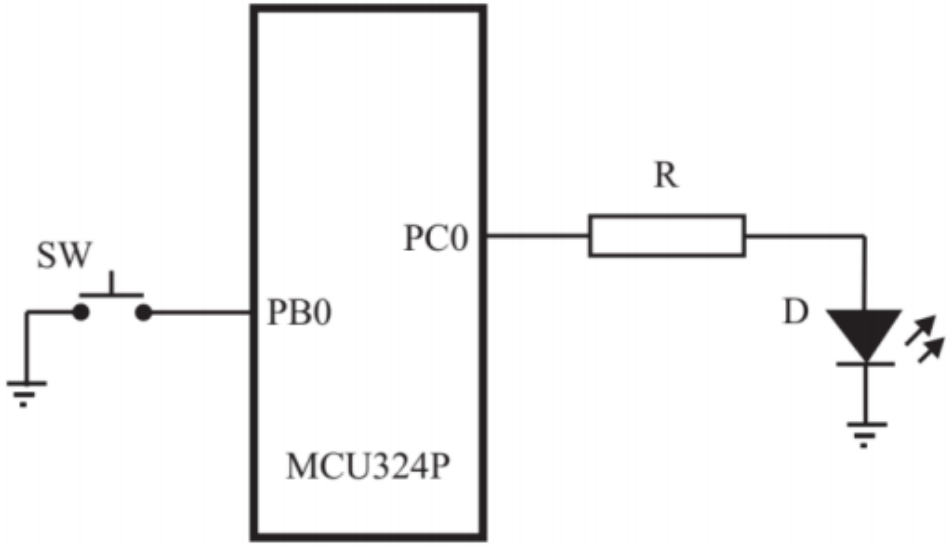
Ví dụ 1: Kết nối SW với một chân port làm ngõ vào, MCU324P nhận dạng SW nhấn tạo mức 0 ngõ vào sẽ thực hiện rẽ nhánh chuyển điều khiển chương trình.



6.1 Giao tiếp với nút nhấn, LED đơn

Đoạn chương trình nhận dạng SW nhấn như sau:

```
        CBI     DDRB,0          ;khai báo PB0 là ngõ vào
        SBI     PORTB,0         ;khai báo điện trở kéo lên ngõ PB0
WAIT:   SBIC    PINB,0          ;bỏ qua lệnh kế tiếp nếu SW nhấn PB0=0
        RJMP    WAIT            ;SW không nhấn PB0=1 lặp vòng lại
;----- Thực thi đoạn chương trình khi SW nhấn
        …
```

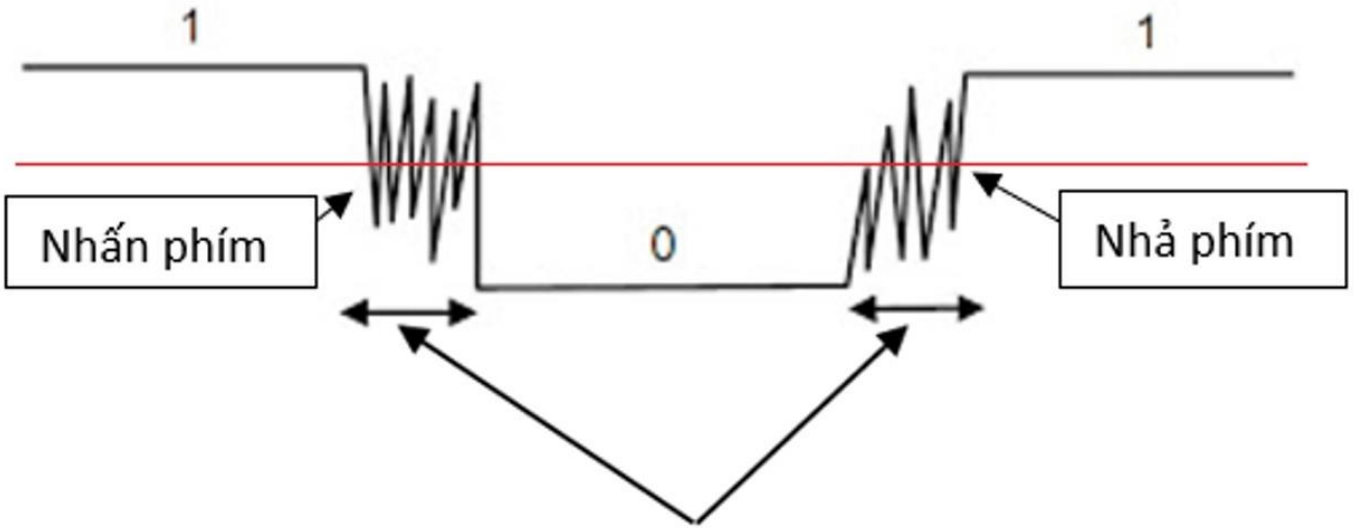


6.1 Giao tiếp với nút nhấn, LED đơn

Ví dụ 2: Kết nối SW với một chân port làm ngõ vào, MCU324P nhận dạng SW nhấn tạo mức 0 ngõ vào sẽ thực hiện bật sáng đèn LED. Giả sử trước đó đèn LED tắt.

```
        .ORG    0
        CBI     DDRB,0      ;khai báo PB0 là ngõ vào
        SBI     PORTB,0     ;khai báo điện trở kéo lên ngõ PB0
        SBI     DDRC,0      ;khai báo PC0 là ngõ ra
        CBI     PORTC,0     ;tắt LED
WAIT:   SBIC    PINB,0      ;bỏ qua lệnh kế tiếp nếu SW nhấn PB0=0
        RJMP    WAIT        ;SW không nhấn PB0=1 lặp vòng lại
        SBI     PORTC,0     ;bật LED
HERE:   RJMP    HERE        ;kết thúc chương trình (dừng tại chỗ)
```

6.1 Giao tiếp với nút nhấn (có chống rung phím)

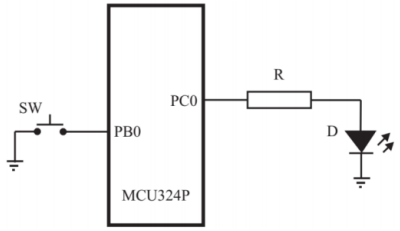


6.1 Chống rung phím bằng chương trình delay

Lưu ý: Thời gian delay có thể từ 10ms đến 20ms tùy loại SW để chống rung hiệu quả nhất!

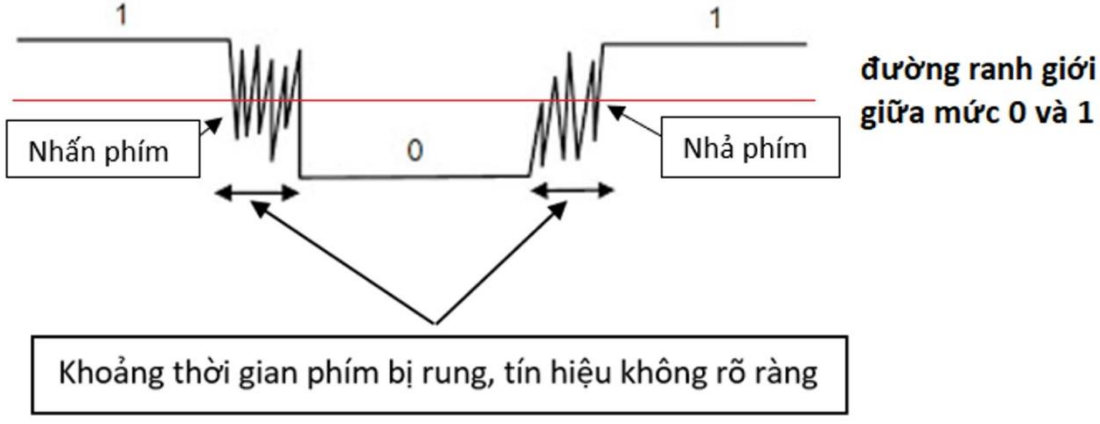
Ví dụ 3: Kết nối SW với một chân port làm ngõ vào, MCU324P nhận dạng SW nhấn tạo mức 0 ngõ vào sẽ thực hiện **đảo trạng thái** đèn LED (ON ↔ OFF). Giả sử trước đó đèn LED tắt.

```
        .ORG    0               ;địa chỉ bắt đầu chương trình sau khi reset
        RJMP    MAIN            ;nhảy đến chương trình chính
        .ORG    0x40            ;địa chỉ bắt đầu chương trình chính
MAIN:   LDI     R16,HIGH(RAMEND);đưa stack lên vùng địa chỉ cao
        OUT     SPH,R16
        LDI     R16,LOW(RAMEND)
        OUT     SPL,R16
        CBI     DDRB,0          ;khai báo PB0 là ngõ vào
        SBI     PORTB,0         ;khai báo điện trở kéo lên ngõ PB0
        SBI     DDRC,0          ;khai báo PC0 là ngõ ra
        CBI     PORTC,0         ;PC0=0 LED tối
```



6.1 Chống rung phím bằng chương trình delay

```
WAIT:   SBIC    PINB,0          ;bỏ qua lệnh kế tiếp nếu SW nhấn PB0=0
        RJMP    WAIT            ;SW không nhấn PB0=1 lặp vòng lại
        RCALL   DELAY_20MS      ;delay 20ms (bỏ qua rung phím khi nhấn)
        SBIC    PINB,0          ;đọc lại trạng thái SW (xem SW có thật sự được nhấn)
        RJMP    WAIT            ;lặp vòng lại nếu mức 1 (nếu SW vẫn bị rung sau delay)
        LDI     R17,0x01        ;R17=01H
        IN      R18,PORTC       ;đọc PORTC
        EOR     R18,R17         ;đảo bit PC0
        OUT     PORTC,R18       ;xuất ra PORTC
        RJMP    WAIT            ;lặp vòng lại từ đầu
```



6.1 Chống rung phím bằng chương trình delay

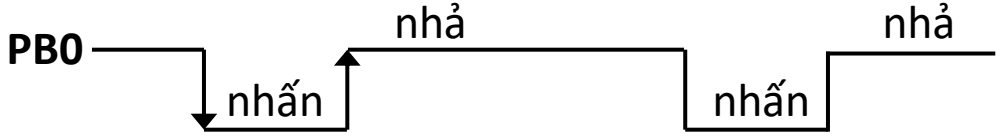
```
;----- DELAY_20MS (Fosc = 8MHz, CKDIV8 = 1)
DELAY_20MS:
        LDI     R21,160         ;1MC
L1:     LDI     R20,250         ;1MC
L2:     DEC     R20             ;1MC
        NOP                     ;1MC
        BRNE    L2              ;2/1MC
        DEC     R21             ;1MC
        BRNE    L1              ;2/1MC
        RET                     ;4MC
```

- Tổng thời gian delay (gần đúng) = 4 x 160 x 250 x 0.125µs = 20000µs = 20ms

☞ Cho biết hiện tượng gì xảy ra nếu phím được nhấn giữ (không được nhả ra)?

6.1 Giao tiếp với nút nhấn có chống rung và chống giữ phím

Ví dụ 4: Kết nối SW với một chân port làm ngõ vào, MCU324P nhận dạng SW nhấn tạo mức 0 ngõ vào sẽ thực hiện **đảo trạng thái** đèn LED (ON ↔ OFF). Giả sử trước đó đèn LED tắt.




```
        .ORG    0
        CBI     DDRB,0          ;khai báo PB0 là ngõ vào
        SBI     PORTB,0         ;khai báo điện trở kéo lên ngõ PB0
        SBI     DDRC,0          ;khai báo PC0 là ngõ ra
        CBI     PORTC,0         ;tắt LED
WAIT_0: SBIC    PINB,0          ;chờ nhấn phím.
        RJMP    WAIT_0          ;nếu không nhấn thì tiếp tục chờ.
        RCALL   DELAY_20MS      ;chống rung phím khi nhấn.
WAIT_1: SBIS    PINB,0          ;chờ nhả phím. Chưa nhả thì tiếp tục chờ.
        RJMP    WAIT_1
        RCALL   DELAY_20MS      ;chống rung phím khi nhả.
        LDI     R17,0x01
        IN      R18,PORTC
        EOR     R18,R17         ;đảo trạng thái PC0
        OUT     PORTC,R18
        RJMP    WAIT_0          ;lặp vòng lại từ đầu
```

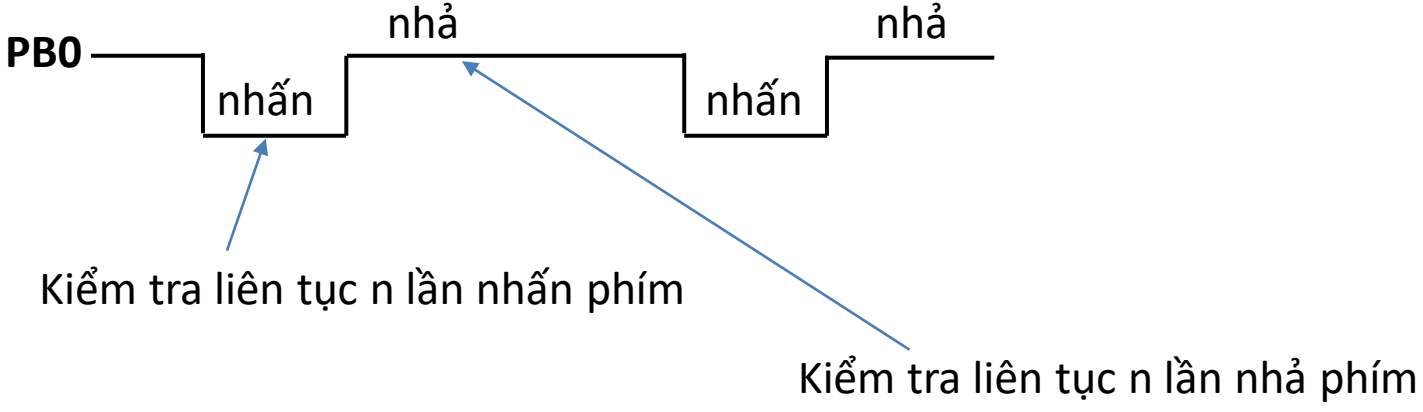
```
;----- DELAY_20MS (Fosc = 8MHz, CKDIV8 = 1)
DELAY_20MS:
        LDI     R21,160         ;1MC
L1:     LDI     R20,250         ;1MC
L2:     DEC     R20             ;1MC
        NOP                     ;1MC
        BRNE    L2              ;2/1MC
        DEC     R21             ;1MC
        BRNE    L1              ;2/1MC
        RET                     ;4MC
```

- Tổng thời gian delay (gần đúng) = 4 x 160 x 250 x 0.125µs = 20000µs = 20ms

Lưu ý: Thời gian delay có thể từ 10ms đến 20ms tùy loại SW để chống rung hiệu quả nhất!

6.1 Chống rung phím bằng cách đọc nhiều lần trạng thái phím

- Một phương pháp chống rung SW là lặp vòng n lần đọc trạng thái SW xác định SW nhấn liên tục và lặp vòng n lần đọc trạng thái SW xác định SW nhả liên tục sẽ xác nhận SW có nhấn/nhả. Nếu chỉ 1 lần đọc không đúng trạng thái SW nhấn/nhả sẽ lặp vòng đọc lại từ đầu.

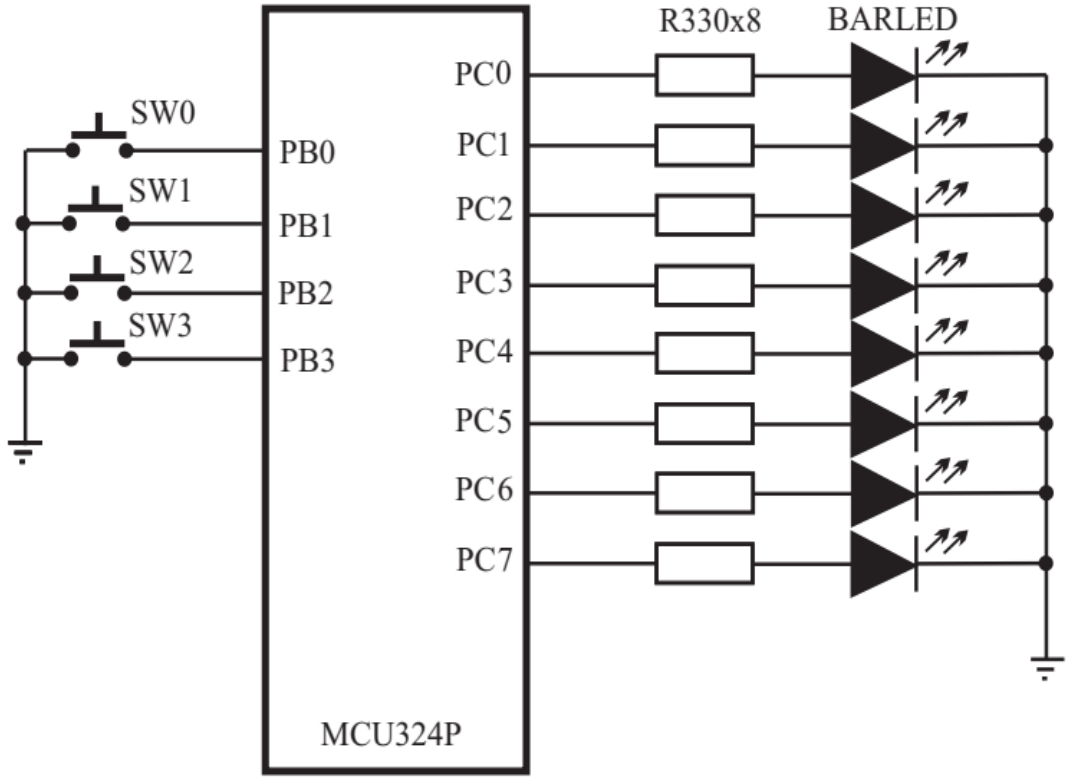


6.1 Chống rung phím bằng cách đọc nhiều lần trạng thái phím

Ví dụ 5: Viết chương trình đọc trạng thái các SW và hiển thị dãy LED (bar LED) nếu có 1 SW nhấn/nhả như sau (có chống rung SW):

- SW0 : tối toàn bộ dãy LED
- SW1 : sáng 4 LED trọng số thấp
- SW2 : sáng 4 LED trọng số cao
- SW3 : sáng toàn bộ dãy LED

Lưu ý: nếu có nhiều hơn 2 phím được nhấn thì chương trình chỉ thực hiện cho phím nào được nhấn trước!



6.1 Chống rung phím bằng cách đọc nhiều lần trạng thái phím

```
        .EQU    OUTPORT=PORTC
        .ORG    0
        RJMP    MAIN
        .ORG    0x40
MAIN:   LDI     R16,HIGH(RAMEND);đưa stack lên vùng địa chỉ cao
        OUT     SPH,R16
        LDI     R16,LOW(RAMEND)
        OUT     SPL,R16
        LDI     R16,0xF0
        OUT     DDRB,R16        ;khai báo PB0-PB3 là ngõ vào
        LDI     R16,0x0F
        OUT     PORTB,R16       ;khai báo điện trở kéo lên ngõ PB0-PB3
        LDI     R16,0xFF
        OUT     DDRC,R16        ;khai báo PORTC là ngõ ra
        LDI     R16,0x00
        OUT     PORTC,R16       ;PORTC=0: LED tối
```

.EQU Ký hiệu = Biểu thức/Hằng số
.DEF Ký hiệu = Thanh ghi GPRs
Xem thêm Chương 4: Hợp ngữ Assembly

```
WAIT_0:  LDI    R16,50    ;số lần nhận dạng SW nhấn
BACK1:   RCALL  GET_KEY   ;gọi ctc nhận dạng SW
         BRCC   WAIT_0    ;C=0: SW chưa nhấn lặp lại
         DEC    R16       ;đếm số lần nhận dạng SW
         BRNE   BACK1     ;lặp vòng cho đủ số lần đếm
         PUSH   R17       ;xác nhận SW nhấn,cất mã SW
WAIT_1:  LDI    R16,50    ;số lần nhận dạng SW nhả
BACK2:   RCALL  GET_KEY   ;gọi ctc nhận dạng SW
         BRCS   WAIT_1    ;C=1: SW chưa nhả
         DEC    R16       ;đếm số lần nhận dạng SW
         BRNE   BACK2     ;lặp vòng cho đủ số lần đếm
         POP    R17       ;xác nhận SW nhả lấy lại mã SW
         CPI    R17,0     ; mã SW = 0?
         BREQ   MD_0      ;hiển thị dãy LED mode 0
         CPI    R17,1     ; mã SW = 1?
         BREQ   MD_1      ;hiển thị dãy LED mode 1
         CPI    R17,2     ;mã SW = 2
         BREQ   MD_2      ;hiển thị dãy LED mode 2
         CPI    R17,3     ;mã SW = 3?
         BREQ   MD_3      ;hiển thị dãy LED mode 3
         RJMP   WAIT_0    ;lặp vòng lại từ đầu
```

Cờ C	T. Thái phím
C = 0	Chưa nhấn
C = 1	Có nhấn

Chống rung phím

```
MD_0:   LDI     R18,0x00        ;hiển thị mode 0
        OUT     OUTPORT,R18     ;dãy LED tối
        RJMP    WAIT_0          ;lặp vòng từ đầu
MD_1:   LDI     R18,0x0F        ;hiển thị mode 1
        OUT     OUTPORT,R18     ;sáng 4 LED trọng số thấp
        RJMP    WAIT_0          ;lặp vòng từ đầu
MD_2:   LDI     R18,0xF0        ;hiển thị mode 2
        OUT     OUTPORT,R18     ;sáng 4 LED trọng số cao
        RJMP    WAIT_0          ;lặp vòng từ đầu
MD_3:   LDI     R18,0xFF        ;hiển thị mode 3
        OUT     OUTPORT,R18     ;sáng cả dãy LED
        RJMP    WAIT_0          ;lặp vòng từ đầu
```

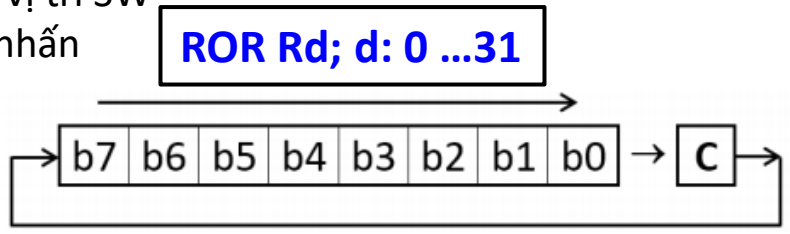
- SW0 = 0 (mode 0): tối toàn bộ dãy LED
- SW1 = 0 (mode 1): sáng 4 LED trọng số thấp
- SW2 = 0 (mode 2): sáng 4 LED trọng số cao
- SW3 = 0 (mode 3): sáng toàn bộ dãy LED

```
;---------------------------------------
;GET_KEY đọc trạng thái các SW,
;Trả về R17= mã SW và C=1 nếu có SW nhấn
;Trả về C=0 nếu SW chưa nhấn
;---------------------------------------
GET_KEY:  LDI     R17,4      ;R17 chứa số vị trí SW
          MOV     R20,R17    ;cất số SW vào R20
          IN      R19,PINB   ;đọc SW
          ANDI    R19,0x0F   ;che 4 bit thấp
          CPI     R19,0x0F   ;xem có SW nhấn?
          BRNE    CHK_KEY    ;R19 khác 0FH: có SW nhấn
NO_KEY:   CLC                ;Trả về C=0: SW chưa nhấn
          RJMP    EXIT       ;thoát
CHK_KEY:  ROR     R19        ;quay phải qua C tìm vị trí SW nhấn
          BRCC    KEY_CODE   ;C=0 (hay bit PBi = 0): có SW được nhấn
          DEC     R20        ;SW không nhấn, giảm vị trí SW
          BRNE    CHK_KEY    ;lặp vòng xét đến hết số vị trí SW
          RJMP    NO_KEY     ;thoát khi không có SW nhấn
KEY_CODE: SUB     R17,R20    ;R17=mã SW
          SEC                ;Trả về C=1: có SW nhấn
EXIT:     RET
```

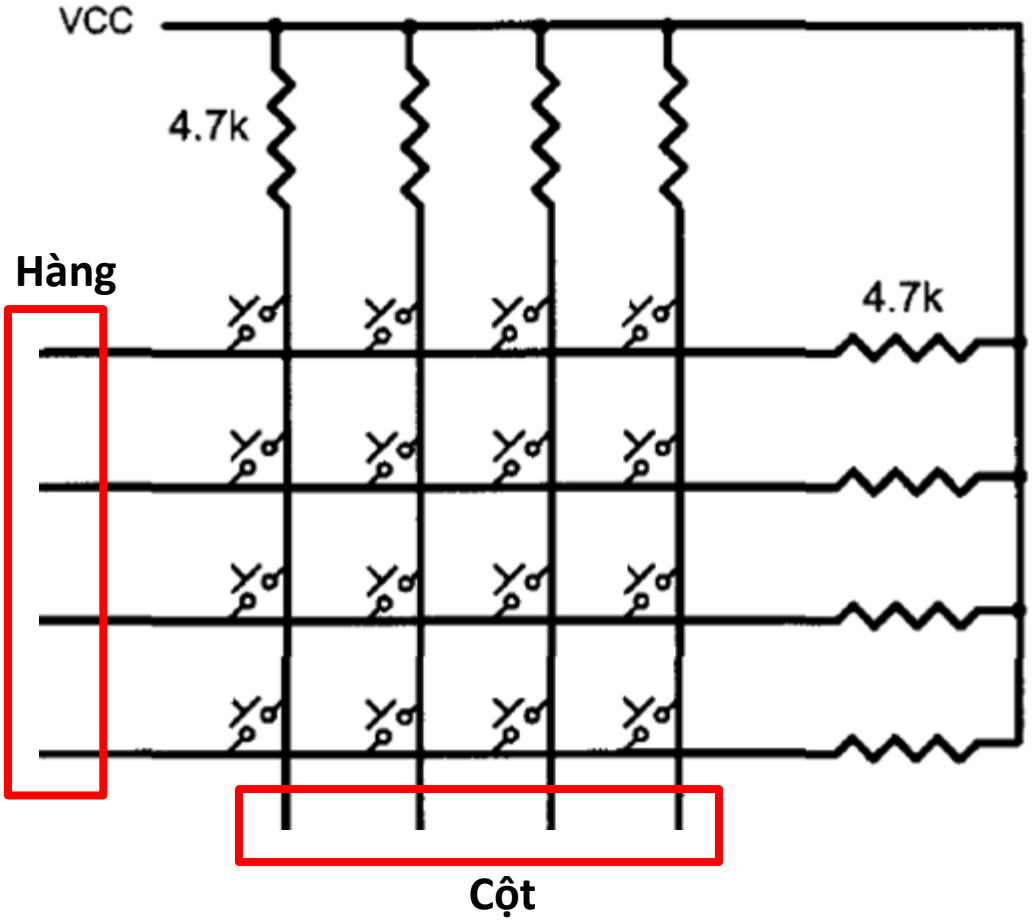
KQ trả về

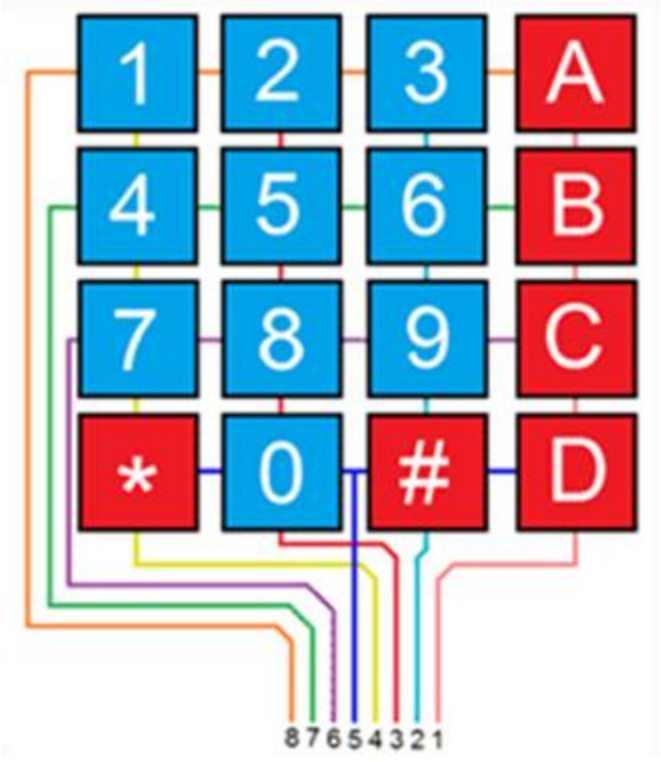
Phím được nhấn	Mã phím
SW0	R17 = 0
SW1	R17 = 1
SW2	R17 = 2
SW3	R17 = 3

Cờ C	T. Thái phím
C = 0	Chưa nhấn
C = 1	Có nhấn

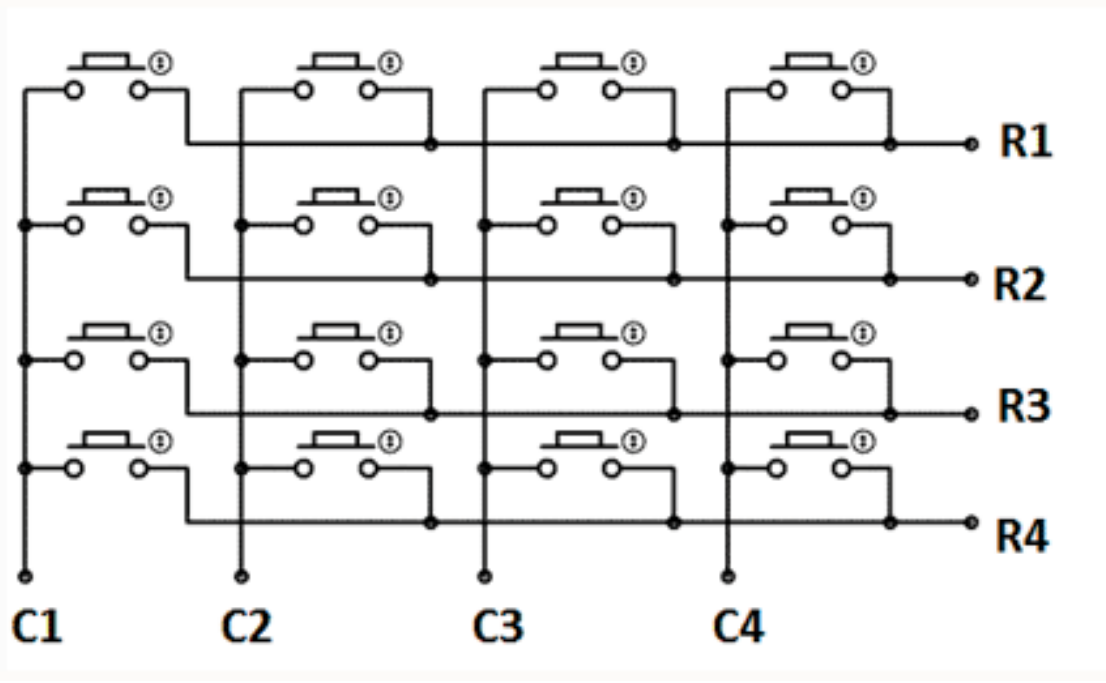


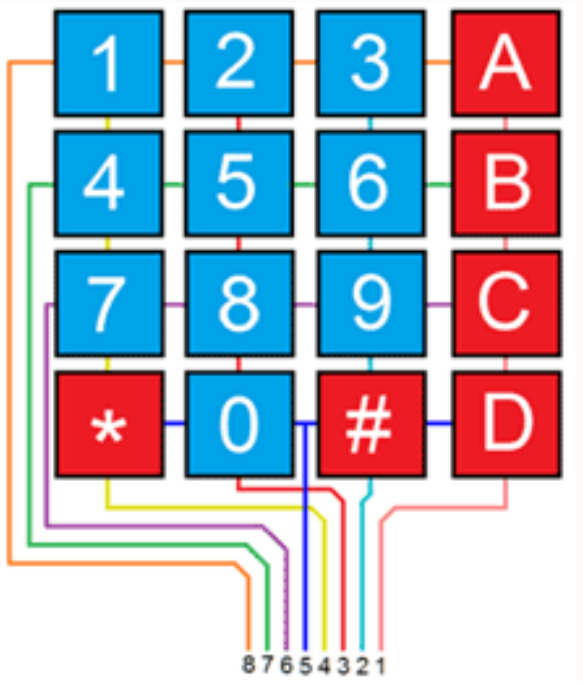
6.2 Giao tiếp với bàn phím ma trận (Keypad)





6.2 Giao tiếp với bàn phím ma trận (Keypad)



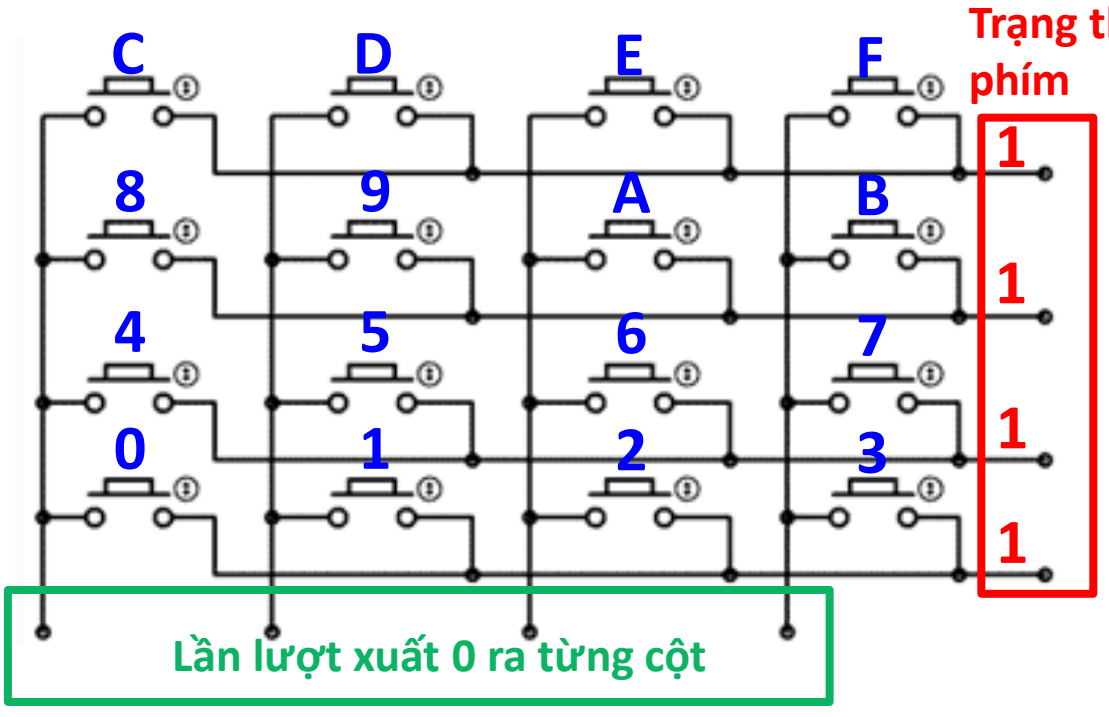


❖ Nhận dạng phím nhấn bằng cách **quét cột**

- Ngõ vào ở 4 hàng: bình thường mức 1.
- **Lần lượt xuất mức 0 ra từng cột.**
- **Đọc trạng thái từng hàng.** Nếu có phím nhấn kết nối với cột đang ở mức 0, hàng tương ứng sẽ bị kéo xuống mức 0.
- Từ mã quét vị trí cột và vị trí hàng ở mức 0 suy ra **mã phím bằng cách cộng vị trí cột và hàng**:
  - Vị trí cột: 0, 1, 2, 3
  - Vị trí hàng: 0, 4, 8, C

Trạng thái phím:

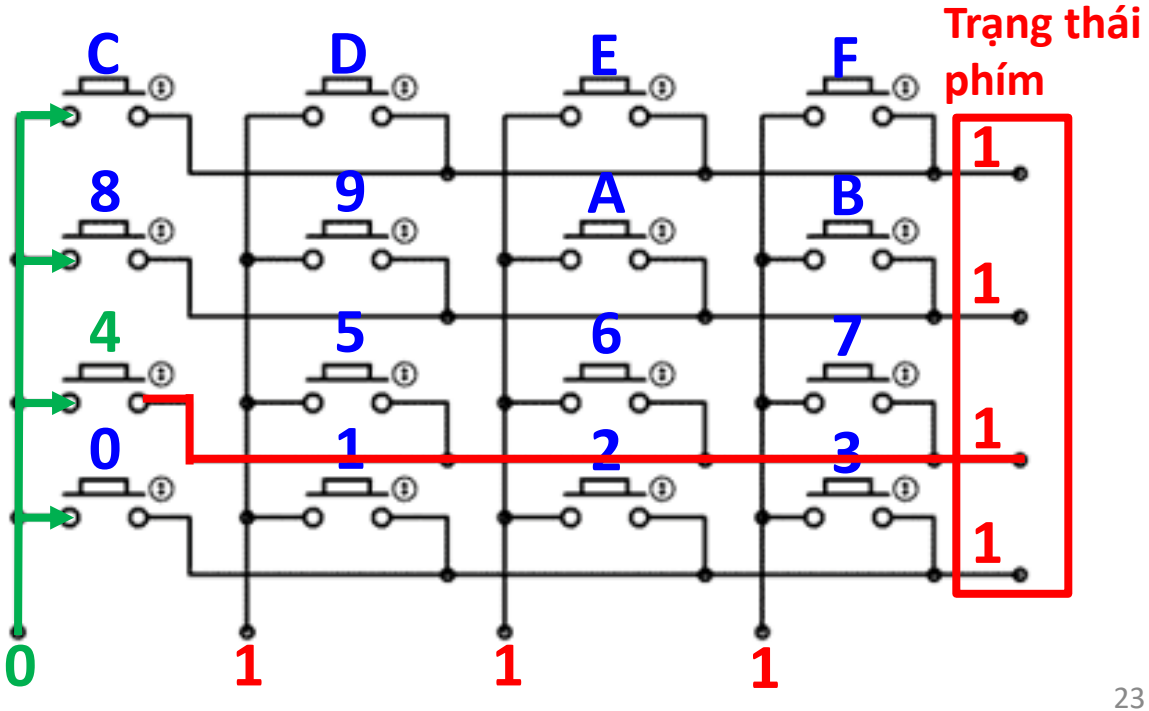
- 0: nhấn
- 1: nhả



❖ Nhận dạng phím nhấn bằng cách **quét cột**

- Ngõ vào ở 4 hàng: bình thường mức 1.
- **Lần lượt xuất mức 0 ra từng cột.**
- **Đọc trạng thái từng hàng.** Nếu có phím nhấn kết nối với cột đang ở mức 0, hàng tương ứng sẽ bị kéo xuống mức 0.
- Từ mã quét vị trí cột và vị trí hàng ở mức 0 suy ra **mã phím bằng cách cộng vị trí cột và hàng**:
  - Vị trí cột: 0, 1, 2, 3
  - Vị trí hàng: 0, 4, 8, C

Ví dụ 6: nhận dạng phím 4.
0: nhấn
1: nhả

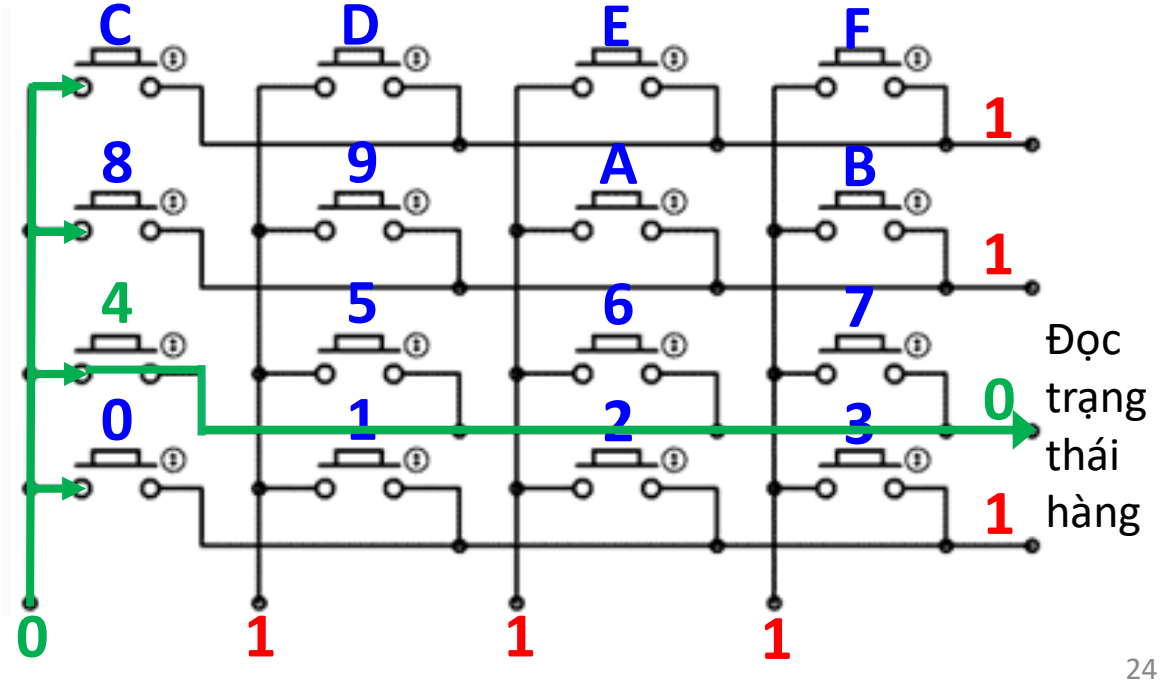


❖ Nhận dạng phím nhấn bằng cách **quét cột**

- Ngõ vào ở 4 hàng: bình thường mức 1.
- **Lần lượt xuất mức 0 ra từng cột.**
- **Đọc trạng thái từng hàng.** Nếu có phím nhấn kết nối với cột đang ở mức 0, hàng tương ứng sẽ bị kéo xuống mức 0.
- Từ mã quét vị trí cột và vị trí hàng ở mức 0 suy ra **mã phím bằng cách cộng vị trí cột và hàng**:
  - Vị trí cột: 0, 1, 2, 3
  - Vị trí hàng: 0, 4, 8, C

Ví dụ 6: nhận dạng phím 4.

Mã phím = vị trí cột + vị trí hàng
= 0 + 4
= 4

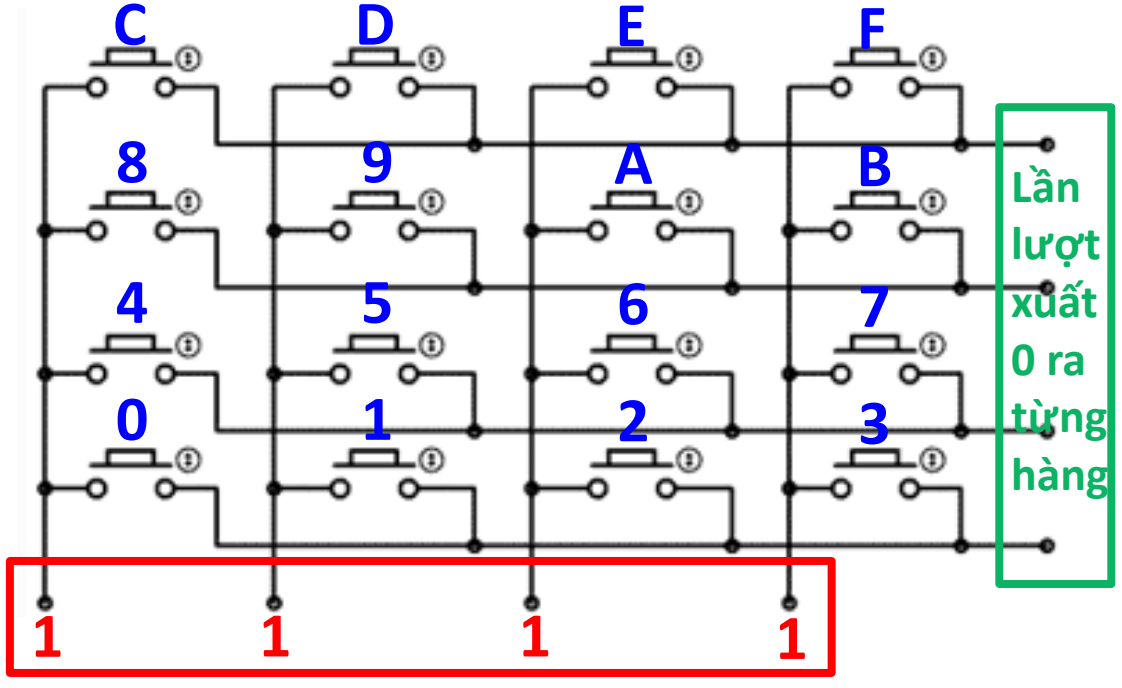


❖ Nhận dạng phím nhấn bằng cách **quét hàng**

- Ngõ vào ở 4 cột: bình thường mức 1.
- **Lần lượt xuất mức 0 ra từng hàng.**
- **Đọc trạng thái từng cột.** Nếu có phím nhấn kết nối với hàng đang ở mức 0, cột tương ứng sẽ bị kéo xuống mức 0.
- Từ mã quét vị trí hàng và vị trí cột ở mức 0 suy ra **mã phím bằng cách cộng vị trí hàng và cột**:
  - Vị trí cột: 0, 1, 2, 3
  - Vị trí hàng: 0, 4, 8, C

Trạng thái phím:

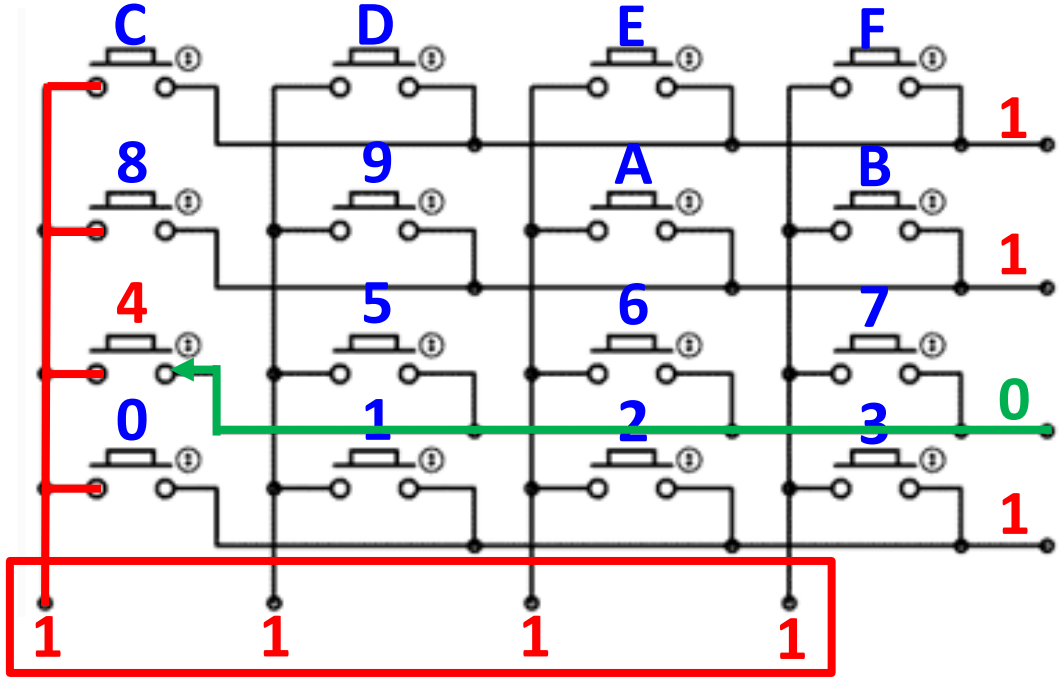
- 0: nhấn
- 1: nhả



❖ Nhận dạng phím nhấn bằng cách **quét hàng**

- Ngõ vào ở 4 cột: bình thường mức 1.
- **Lần lượt xuất mức 0 ra từng hàng.**
- **Đọc trạng thái từng cột.** Nếu có phím nhấn kết nối với hàng đang ở mức 0, cột tương ứng sẽ bị kéo xuống mức 0.
- Từ mã quét vị trí hàng và vị trí cột ở mức 0 suy ra **mã phím bằng cách cộng vị trí hàng và cột**:
  - Vị trí cột: 0, 1, 2, 3
  - Vị trí hàng: 0, 4, 8, C

Ví dụ 7: nhận dạng phím 4.
0: nhấn
1: nhả

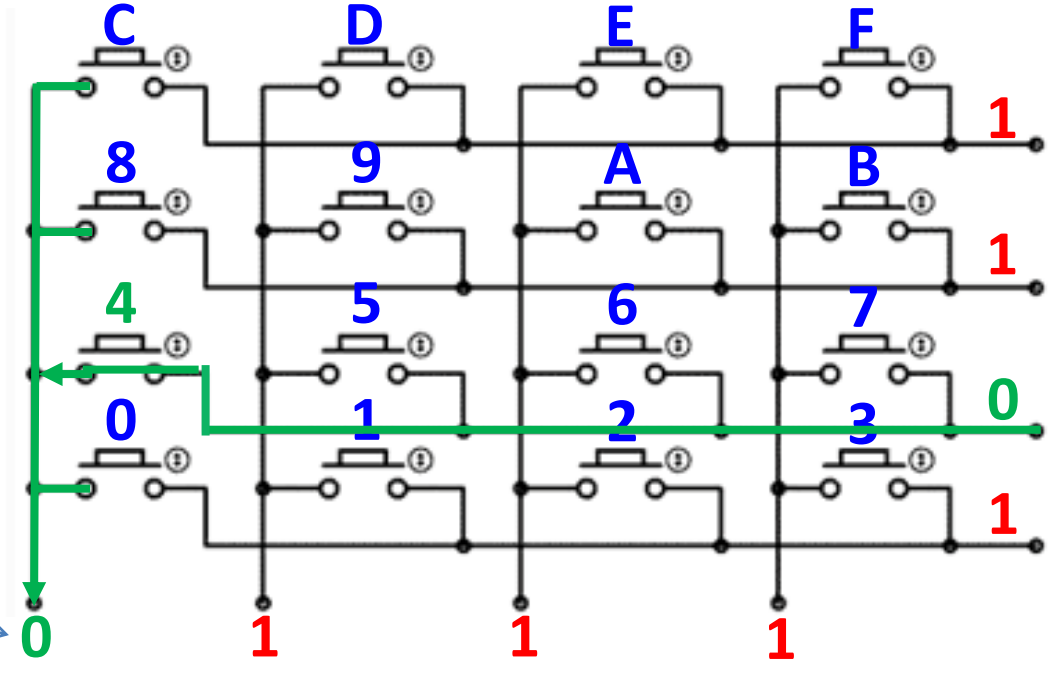


❖ Nhận dạng phím nhấn bằng cách **quét hàng**

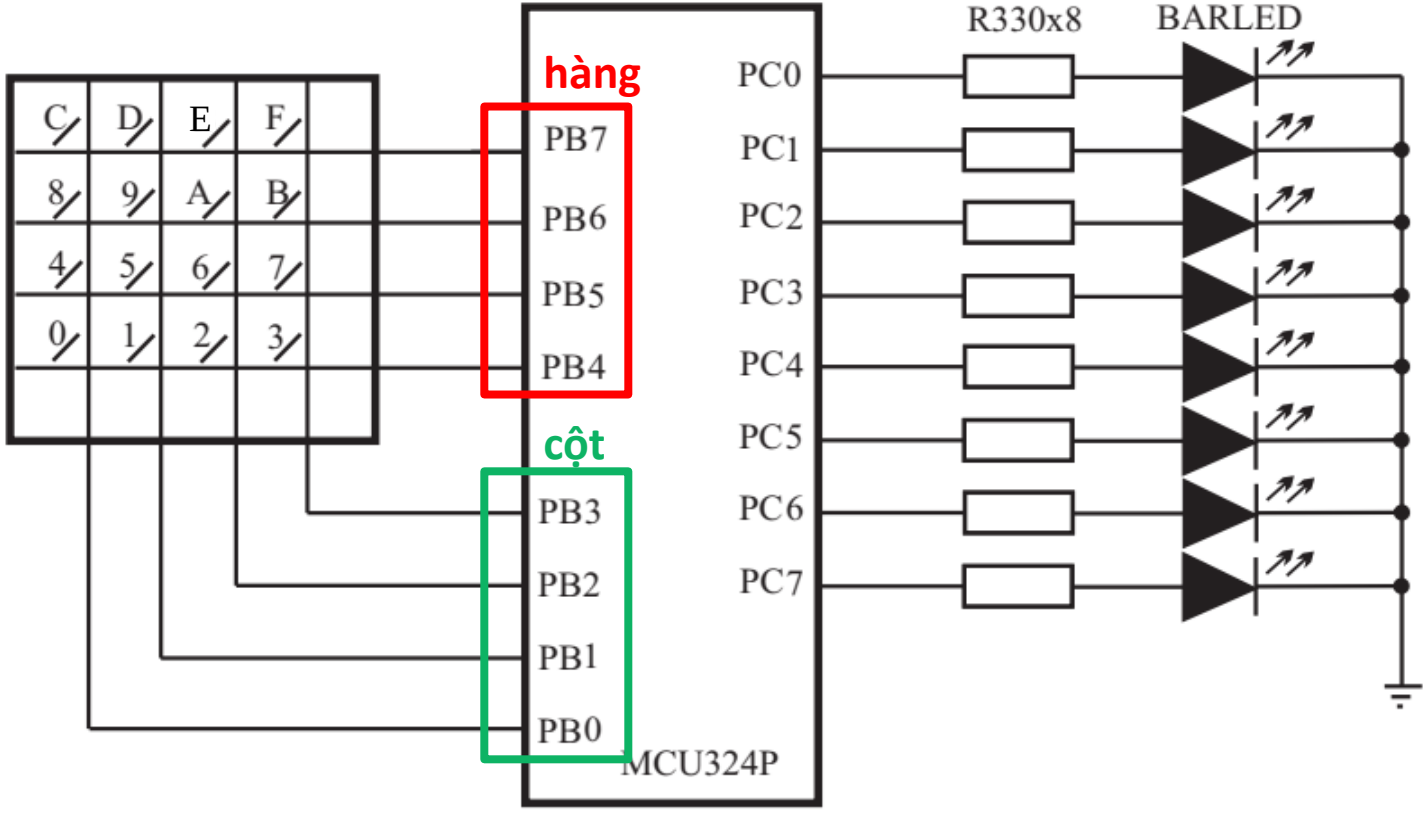
- Ngõ vào ở 4 cột: bình thường mức 1.
- **Lần lượt xuất mức 0 ra từng hàng.**
- **Đọc trạng thái từng cột.** Nếu có phím nhấn kết nối với hàng đang ở mức 0, cột tương ứng sẽ bị kéo xuống mức 0.
- Từ mã quét vị trí hàng và vị trí cột ở mức 0 suy ra **mã phím bằng cách cộng vị trí hàng và cột**:
  - Vị trí cột: 0, 1, 2, 3
  - Vị trí hàng: 0, 4, 8, C

Ví dụ 7: nhận dạng phím 4.

Mã phím = vị trí hàng + vị trí cột
= 4 + 0
= 4



Ví dụ 8: Cho sơ đồ kết nối MCU324P giao tiếp bàn phím (keypad) 16 phím như bên dưới. Sử dụng phương pháp **quét cột**, viết chương trình nhận dạng khi nhấn một phím sẽ hiển thị mã HEX tương ứng với mã phím và xuất ra bar LED. Có chống rung phím khi nhấn và nhả.



```
        .EQU    OUTPORT=PORTC
        .ORG    0
        RJMP    MAIN
        .ORG    0X40
MAIN:   LDI     R16,HIGH(RAMEND);đưa stack lên vùng địa chỉ cao
        OUT     SPH,R16
        LDI     R16,LOW(RAMEND)
        OUT     SPL,R16
        LDI     R16,0X0F
        OUT     DDRB,R16        ;khai báo PB4-PB7 ngõ vào, PB0-PB3 ngõ ra
        LDI     R16,0XFF
        OUT     PORTB,R16       ;khai báo điện trở kéo lên ngõ PB0-PB7
        LDI     R16,0XFF
        OUT     DDRC,R16        ;khai báo PORTC là output
        LDI     R16,0X00
        OUT     PORTC,R16       ;PORTC=0 LED tối
START:  RCALL   KEY_RD          ;ctc đọc phím
        OUT     OUTPORT,R17     ;hiển thị mã phím ra bar LED
        RJMP    START           ;lặp vòng từ đầu
```

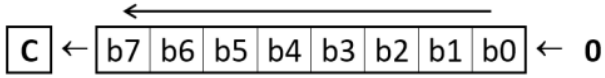
```
;---------------------------------------
;KEY_RD đọc trạng thái phím
;Chống rung phím khi nhấn/nhả 50 lần
;Sử dụng GET_KEY16 nhận dạng phím nhấn: R17 = mã phím, C = 0: chưa nhấn, C = 1: có nhấn
;Chỉ thoát khi có phím nhấn!!!
;---------------------------------------
KEY_RD:  LDI     R16,50        ;số lần nhận dạng phím nhấn
BACK1:   RCALL   GET_KEY16     ;gọi ctc nhận dạng phím
         BRCC    KEY_RD        ;C=0 phím chưa nhấn lặp lại
         DEC     R16           ;đếm số lần nhận dạng phím
         BRNE    BACK1         ;lặp vòng cho đủ số lần đếm
         PUSH    R17           ;xác nhận phím nhấn,cất mã phím
WAIT_1:  LDI     R16,50        ;số lần nhận dạng phím nhả
BACK2:   RCALL   GET_KEY16     ;gọi ctc nhận dạng phím
         BRCS    WAIT_1        ;C=1 phím chưa nhả
         DEC     R16           ;đếm số lần nhận dạng phím
         BRNE    BACK2         ;lặp vòng cho đủ số lần đếm
         POP     R17           ;xác nhận phím nhả lấy lại mã phím
         RET
```

```
;---------------------------------------
;GET_KEY16 đọc trạng thái các phím,
;Trả về R17= mã phím và C=1 nếu có phím nhấn
;Trả về C=0 nếu phím chưa nhấn
;---------------------------------------
GET_KEY16:
LDI     R17,4       ;R17 đếm số lần quét cột
LDI     R20,0XFE    ;bắt đầu quét cột 0
SCAN_COL:           ;hoặc NOP → delay 1MC
OUT     PORTB,R20   ;
IN      R19,PINB    ;đọc trạng thái hàng
IN      R19,PINB    ;đọc lại trạng thái hàng
ANDI    R19,0XF0    ;che 4 bit cao lấy mã hàng
CPI     R19,0XF0    ;xem có phím nhấn?
BRNE    CHK_KEY     ;R19 khác F0H: có phím nhấn
LSL     R20         ;quét cột kế tiếp
INC     R20         ;đặt LSB=1
DEC     R17
BRNE    SCAN_COL    ;tiếp tục quét hết số cột
CLC                 ;phím chưa nhấn,C=0
RJMP    EXIT        ;thoát
```

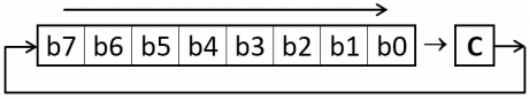
Vị trí hàng	0	4	8	C
Vị trí cột	0	1	2	3

```
CHK_KEY:
SUBI    R17,4       ;tính vị trí cột
NEG     R17         ;bù 2 (số ≤ 0) → số dương
SWAP    R19         ;đảo sang 4 bit thấp mã hàng
LDI     R20,4       ;quét 4 hàng → tìm vị trí hàng…
SCAN_ROW:           ; tìm vị trí hàng có phím được nhấn
ROR     R19         ;quay phải mã hàng qua C tìm bit 0
BRCC    SET_FLG     ;C=0 đúng vị trí hàng có phím nhấn
INC     R17   ;không đúng hàng, tăng vị trí hàng thêm 4
INC     R17
INC     R17
INC     R17
DEC     R20
BRNE    SCAN_ROW    ;quét hết 4 hàng
SET_FLG:SEC         ;có phím nhấn C=1
EXIT:   RET
```

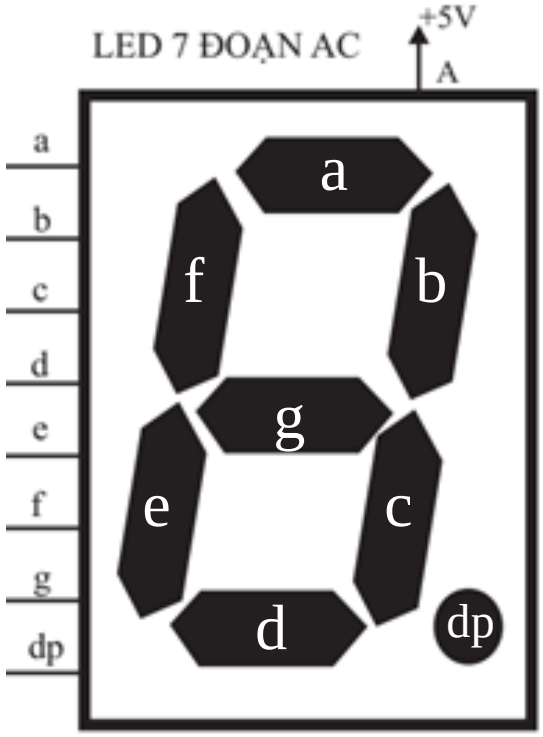
LSL Rd;



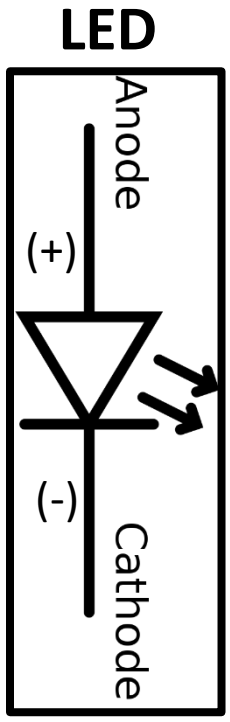
ROR Rd;

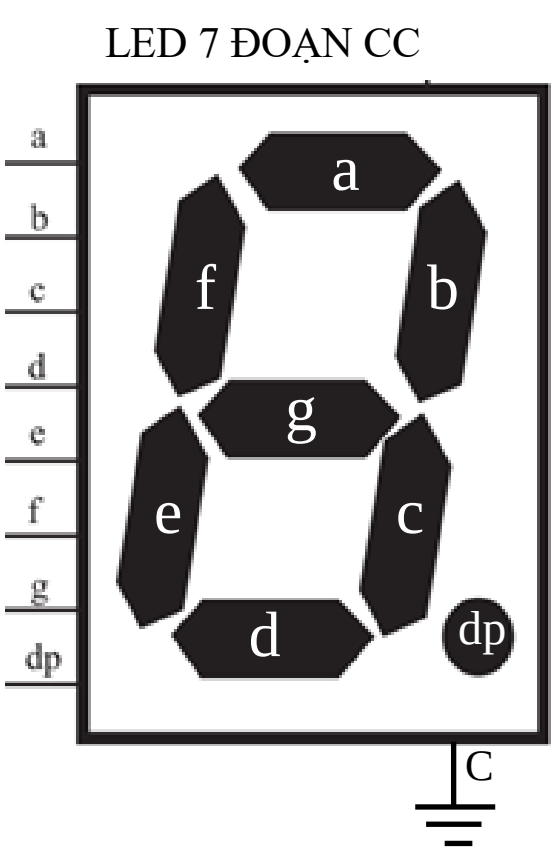


6.3 Giao tiếp với LED 7 đoạn



LED 7 đoạn Anode chung

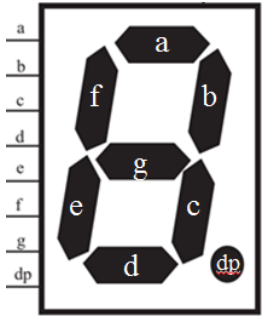




LED 7 đoạn Cathode chung

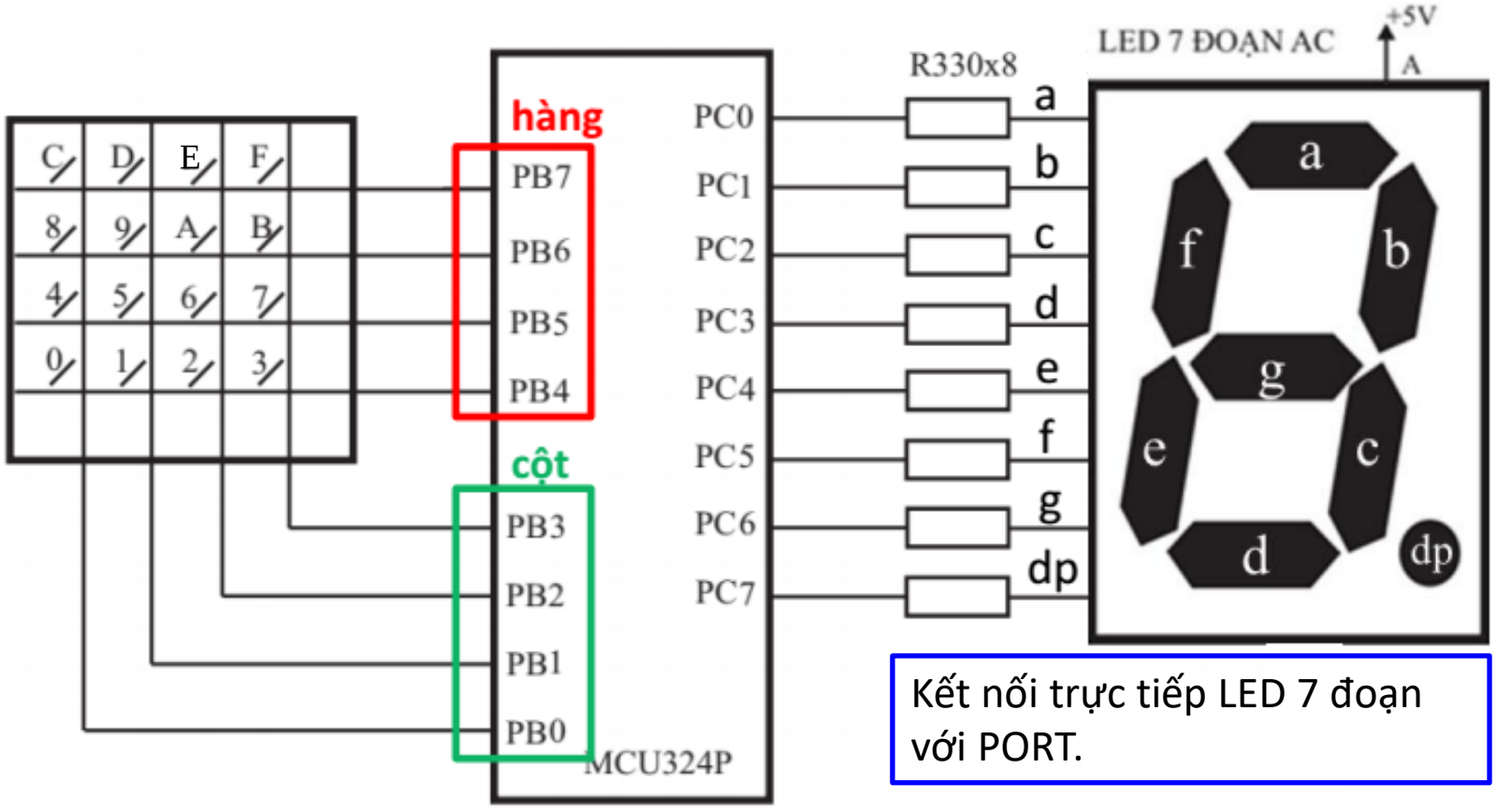
6.3 Giao tiếp với LED 7 đoạn

	MSB							LSB
Đoạn	dp	g	f	e	d	c	b	a
Data	D7	D6	D5	D4	D3	D2	D1	D0



Ký tự (HEX)	Anode chung	Cathode chung	Ký tự (HEX)	Anode chung	Cathode chung
0	C0	3F	8	80	7F
1	F9	06	9	90	6F
2	A4	5B	A	88	77
3	B0	4F	b	83	7C
4	99	66	C	C6	39
5	92	6D	d	A1	5E
6	82	7D	E	86	79
7	F8	07	F	8E	71

Ví dụ 9: Cho sơ đồ kết nối MCU324P giao tiếp bàn phím (keypad) 16 phím như bên dưới. Sử dụng phương pháp **quét cột**, viết chương trình nhận dạng khi nhấn một phím sẽ hiển thị mã HEX tương ứng với mã phím và xuất ra LED 7 đoạn Anode chung. Có chống rung phím khi nhấn và nhả.



```
        .EQU    OUTPORT=PORTC
        .ORG    0
        RJMP    MAIN
        .ORG    0X40
MAIN:   LDI     R16,HIGH(RAMEND);đưa stack lên vùng địa chỉ cao
        OUT     SPH,R16
        LDI     R16,LOW(RAMEND)
        OUT     SPL,R16
        LDI     R16,0X0F
        OUT     DDRB,R16        ;khai báo PB4-PB7 ngõ vào, PB0-PB3 ngõ ra
        LDI     R16,0XFF
        OUT     PORTB,R16       ;khai báo điện trở kéo lên ngõ PB0-PB7
        LDI     R16,0XFF
        OUT     DDRC,R16        ;khai báo PORTC là output
        LDI     R16,0XFF
        OUT     PORTC,R16       ;PORTC=FFH: tắt LED 7 đoạn
START:  RCALL   KEY_RD          ;ctc đọc phím
        RCALL   SEG_LED         ;hiển thị mã phím (HEX) ra LED 7 đoạn
        RJMP    START           ;lặp vòng từ đầu
```

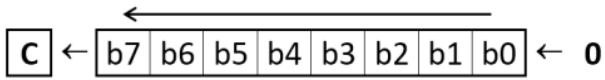
```
;---------------------------------------
;KEY_RD đọc trạng thái phím
;Chống rung phím khi nhấn/nhả 50 lần
;Sử dụng GET_KEY16 nhận dạng phím nhấn: R17 = mã phím, C = 0: chưa nhấn, C = 1: có nhấn
;Chỉ thoát khi có phím nhấn!!!
;---------------------------------------
KEY_RD:  LDI     R16,50          ;số lần nhận dạng phím nhấn
BACK1:   RCALL   GET_KEY16       ;gọi ctc nhận dạng phím
         BRCC    KEY_RD          ;C=0 phím chưa nhấn lặp lại
         DEC     R16             ;đếm số lần nhận dạng phím
         BRNE    BACK1           ;lặp vòng cho đủ số lần đếm
         PUSH    R17             ;xác nhận phím nhấn,cất mã phím
WAIT_1:  LDI     R16,50          ;số lần nhận dạng phím nhả
BACK2:   RCALL   GET_KEY16       ;gọi ctc nhận dạng phím
         BRCS    WAIT_1          ;C=1 phím chưa nhả
         DEC     R16             ;đếm số lần nhận dạng phím
         BRNE    BACK2           ;lặp vòng cho đủ số lần đếm
         POP     R17             ;xác nhận phím nhả lấy lại mã phím
         RET
```

```
;---------------------------------------
;GET_KEY16 đọc trạng thái các phím,
;Trả về R17= mã phím và C=1 nếu có phím nhấn
;Trả về C=0 nếu phím chưa nhấn
;---------------------------------------
GET_KEY16:
LDI     R17,4       ;R17 đếm số lần quét cột
LDI     R20,0XFE    ;bắt đầu quét cột 0
SCAN_COL:           ;hoặc NOP → delay 1MC
OUT     PORTB,R20   ;
IN      R19,PINB    ;đọc trạng thái hàng
IN      R19,PINB    ;đọc lại trạng thái hàng
ANDI    R19,0XF0    ;che 4 bit cao lấy mã hàng
CPI     R19,0XF0    ;xem có phím nhấn?
BRNE    CHK_KEY     ;R19 khác F0H, có phím nhấn
LSL     R20         ;quét cột kế tiếp
INC     R20         ;đặt LSB=1
DEC     R17
BRNE    SCAN_COL    ;tiếp tục quét hết số cột
CLC                 ;phím chưa nhấn,C=0
RJMP    EXIT        ;thoát
```

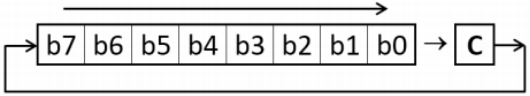
Vị trí hàng	0	4	8	C
Vị trí cột	0	1	2	3

```
CHK_KEY:
SUBI    R17,4       ;tính vị trí cột
NEG     R17         ;bù 2 (số ≤ 0) → số dương
SWAP    R19         ;đảo sang 4 bit thấp mã hàng
LDI     R20,4       ;quét 4 hàng → tìm vị trí hàng…
SCAN_ROW:           ; tìm vị trí hàng có phím được nhấn
ROR     R19         ;quay phải mã hàng qua C tìm bit 0
BRCC    SET_FLG     ;C=0 đúng vị trí hàng có phím nhấn
INC     R17   ;không đúng hàng, tăng vị trí hàng thêm 4
INC     R17
INC     R17
INC     R17
DEC     R20
BRNE    SCAN_ROW    ;quét hết 4 hàng
SET_FLG:SEC         ;có phím nhấn C=1
EXIT:   RET
```

LSL Rd;



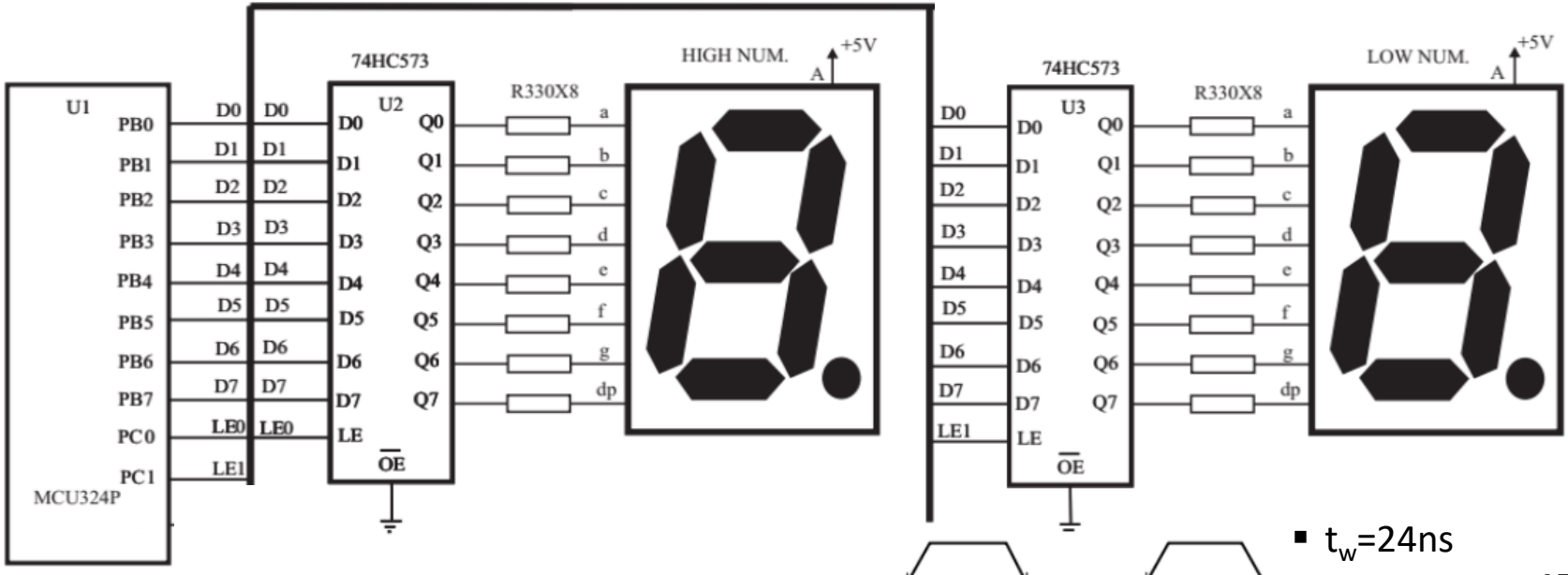
ROR Rd;



```
;----------------------------------------
;SEG_LED: tra bảng mã LED 7 đoạn Anode chung
;R17: mã HEX của ký tự hiển thị ra LED 7 đoạn
;----------------------------------------
SEG_LED: LDI     ZH,HIGH(TAB_7SA<<1)     ;Z trỏ địa chỉ đầu bảng tra mã 7 đoạn
         LDI     ZL,LOW(TAB_7SA<<1)      ;trong flash ROM
         ADD     R30,R17                 ;cộng offset vào ZL (ZL = R30)
         LDI     R17,0
         ADC     R31,R17                 ;cộng carry vào ZH (ZH = R31)
         LPM     R17,Z                   ;lấy mã 7 đoạn
         OUT     OUTPORT,R17             ;xuất mã 7 đoạn
         RET
TAB_7SA: .DB     0XC0,0XF9,0XA4,0XB0,0X99,0X92,0X82,0XF8,0X80,0X90,0X88,0X83
         .DB     0XC6,0XA1,0X86,0X8E
```

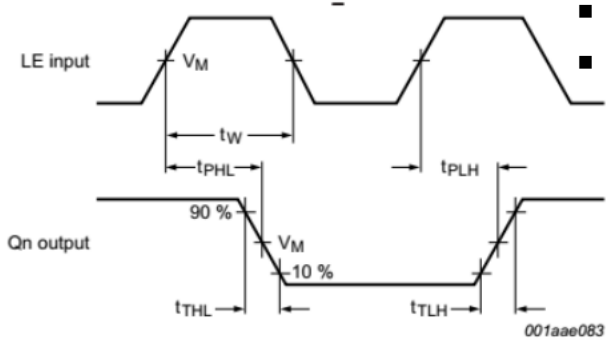
Giao tiếp với nhiều hơn 1 LED 7 đoạn

Ví dụ 10: Cho sơ đồ kết nối MCU324P giao với 2 LED 7 đoạn Anode chung như bên dưới. Viết chương trình hiện số HEX 12 ra 2 LED tương ứng.



→ **Dùng 2 IC chốt 8 bit 74HC573**

→ **Phương pháp chốt LED**



- t_w=24ns
- $t_{pd}=t_{PHL}=t_{PLH}$=45ns

```
        .EQU    OUTPORT=PORTB
        .EQU    IOSET=DDRB
        .ORG    0
        RJMP    MAIN
        .ORG    0X40
MAIN:   LDI     R16,HIGH(RAMEND)        ;đưa stack lên vùng đ/c cao
        OUT     SPH,R16
        LDI     R16,LOW(RAMEND)
        OUT     SPL,R16
        LDI     R16,0X03
        OUT     DDRC,R16                ;khai báo PC0,PC1 là output
        CBI     PORTC,0                 ;khóa ngõ ra U2
        CBI     PORTC,1                 ;khóa ngõ ra U3
        LDI     R16,0xFF
        OUT     IOSET,R16               ;khai báo PORT output
```

```
LDI     R17,0X12; xuất 12 ra 2 LED
PUSH    R17       ;cất data
SWAP    R17       ;hoán vị sang 4 bit thấp
ANDI    R17,0X0F ;che 4 bit thấp data
RCALL   GET_7SEG ;lấy mã 7 đoạn
OUT     OUTPORT,R17 ;xuất mã 7 đoạn
SBI     PORTC,0 ;mở U2 (2MC)
CBI     PORTC,0 ;khóa U2 (2MC)
POP     R17       ;phục hồi data
ANDI    R17,0X0F ;che 4 bit thấp
RCALL   GET_7SEG ;lấy mã 7 đoạn
OUT     OUTPORT,R17 ;xuất mã 7 đoạn
SBI     PORTC,1 ;mở U3 (2MC)
CBI     PORTC,1 ;khóa U3 (2MC)
HR:
RJMP    HR
```

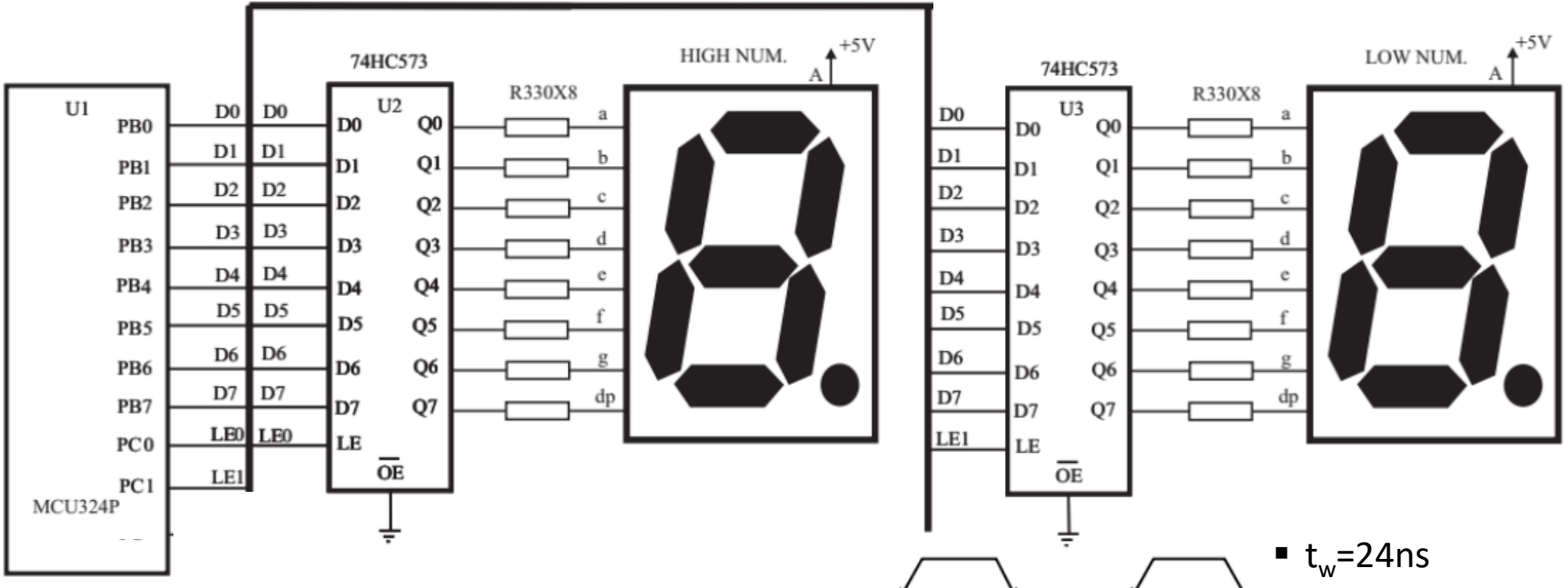
2

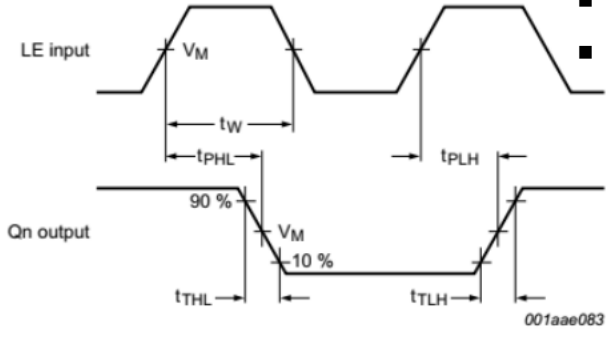
Xem mô tả IC 74HC573 ở ví dụ 6.1 giáo trình VXL.

3

```
;---------------------------------------
;GET_7SEG tra mã 7 đoạn từ data đọc vào
;Input R17=mã Hex,Output R17=mã 7 đoạn
;---------------------------------------
GET_7SEG:
;Z trỏ địa chỉ đầu bảng tra mã 7 đoạn
LDI     ZH,HIGH(TAB_7SA<<1)
LDI     ZL,LOW(TAB_7SA<<1) ;trong flash ROM
ADD     R30,R17   ;cộng offset vào ZL
LDI     R17,0
ADC     R31,R17   ;cộng carry vào ZH
LPM     R17,Z     ;lấy mã 7 đoạn
RET
;---------------------------------------
TAB_7SA:
.DB 0XC0,0XF9,0XA4,0XB0,0X99,0X92,0X82,0XF8
.DB 0X80,0X90,0X88,0X83, 0XC6,0XA1,0X86,0X8E
```

Ví dụ 11: Cho sơ đồ kết nối MCU324P giao với 2 LED 7 đoạn Anode chung như bên dưới. Viết chương trình đếm lên từ 00 đến 99 rồi quay lại 00 →01 → …→99 →00… , giá trị bộ đếm tăng lên 1 cứ sau 0.5s. Sử dụng Fosc = 8MHz, CKDIV8 = 1.





- t_w=24ns
- t_{pd}=t_{PHL}=t_{PLH}=45ns

→ **Dùng 2 IC chốt 8 bit 74HC573**

→ **Phương pháp chốt LED**

```
        .EQU    OUTPORT=PORTB
        .EQU    IOSET=DDRB
        .ORG    0
        RJMP    MAIN
        .ORG    0X40
MAIN:   LDI     R16,HIGH(RAMEND)        ;đưa stack lên vùng đ/c cao
        OUT     SPH,R16
        LDI     R16,LOW(RAMEND)
        OUT     SPL,R16
        LDI     R16,0X03
        OUT     DDRC,R16                ;khai báo PC0,PC1 là output
        CBI     PORTC,0                 ;khóa ngõ ra U2
        CBI     PORTC,1                 ;khóa ngõ ra U3
        LDI     R16,0xFF
        OUT      IOSET,R16              ;khai báo PORT output
```

```
AGAIN:
LDI     R18,0     ;đếm hàng đơn vị
LDI     R19,0     ;đếm hàng chục
LP:
MOV     R17,R19
RCALL   GET_7SEG ;lấy mã 7 đoạn
OUT     OUTPORT,R17 ;xuất mã 7 đoạn
SBI     PORTC,0 ;mở U2 (hàng chục)
CBI     PORTC,0 ;khóa U2
DV:
MOV     R17,R18
RCALL   GET_7SEG ;lấy mã 7 đoạn
OUT     OUTPORT,R17 ;xuất mã 7 đoạn
SBI     PORTC,1 ;mở U3 (hàng đơn vị)
CBI     PORTC,1 ;khóa U3
RCALL   DL;tạo trễ 0.5s
```

2

```
INC     R18;tăng hàng đơn vị lên 1
CPI     R18,10
BREQ    CHUC ;đơn vị = 10
RJMP    DV
CHUC:
LDI     R18,0
INC     R19 ;tăng hàng chục lên 1
CPI     R19,10
BREQ    AGAIN
RJMP    LP
```

3

```
;----------------------------------------
;GET_7SEG tra mã 7 đoạn từ data đọc vào
;Input R17=mã Hex,Output R17=mã 7 đoạn
;----------------------------------------
GET_7SEG:
;Z trỏ địa chỉ đầu bảng tra mã 7 đoạn
LDI     ZH,HIGH(TAB_7SA<<1)
LDI     ZL,LOW(TAB_7SA<<1) ;trong flash
ROM
ADD     R30,R17    ;cộng offset vào ZL
LDI     R17,0
ADC     R31,R17   ;cộng carry vào ZH
LPM     R17,Z      ;lấy mã 7 đoạn
RET
```

4

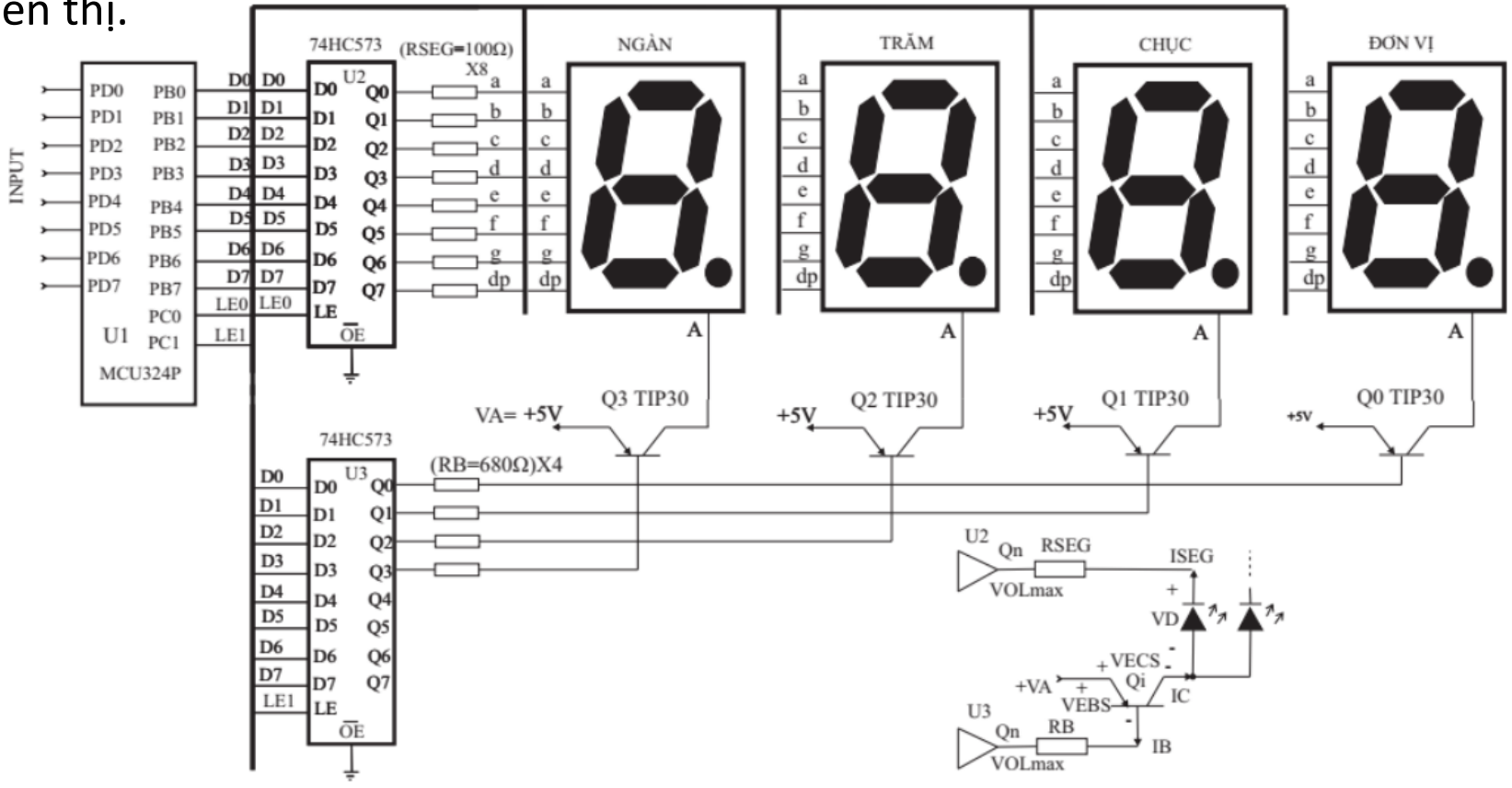
```
;----------------------------------------
;Chương trình tạo trễ 0.5s, Fosc = 8MHz,
;CKDIV8 = 1
DL:     LDI     R22,16
LP3:    LDI     R21,250
LP2:    LDI     R20,250
LP1:    NOP
        DEC     R20
        BRNE    LP1
        DEC     R21
        BRNE    LP2
        DEC     R22
        BRNE    LP3
        RET
```

5

```
;----------------------------------------
TAB_7SA:
.DB 0XC0,0XF9,0XA4,0XB0,0X99,0X92,0X82,0XF8,0X80,0X90
```

6

Ví dụ 12: Thiết kế mạch MCU324P giao tiếp với bộ hiển thị 4 LED 7 đoạn AC dùng **phương pháp quét**. Viết một chương trình con hiển thị các số BCD nén cất trong các thanh ghi R21 và R20 (R21=ngàn-trăm, R20=chục-đơn vị). Viết một chương trình đọc số nhị phân 8 bit từ 1 port và hiển thị giá trị thập phân tương ứng ra bộ hiển thị.



❖ Phương pháp quét LED.

- Dựa trên hiện tượng lưu ảnh của mắt.
- Tại một thời điểm chỉ cấp nguồn cho 1 LED sáng, các LED còn lại tắt, cứ thế tuần tự cho từng LED và sau đó lặp vòng lại.
- Để đảm bảo mắt người có cảm giác các LED đều sáng liên tục, tần số quét lặp vòng **không nhỏ hơn 50Hz** (người lập trình điều chỉnh tần số quét cho phù hợp thực tế).
- Với phương pháp quét, mạch lái bộ hiển thị chỉ cần 1 IC chốt mã 7 đoạn và mạch giải mã sang n đường kích mạch lái khuếch đại dòng điện lần lượt cấp nguồn cho n LED tương ứng.

1

```
        .EQU    OUTPORT=PORTB
        .EQU    INPORT=PIND
        .EQU    SR_ADR=0X100
        .ORG    0
        RJMP    MAIN
        .ORG    0X40
MAIN:   LDI     R16,HIGH(RAMEND)
        OUT     SPH,R16
        LDI     R16,LOW(RAMEND)
        OUT     SPL,R16
        LDI     R16,0X03
        OUT     DDRC,R16 ;khai báo PC0,PC1 là output
        CBI     PORTC,0 ;khóa ngõ ra U2
        CBI     PORTC,1 ;khóa ngõ ra U3
```

2

```
        LDI     R16,0XFF
        OUT     DDRB,R16
        LDI     R16,0X00
        OUT     DDRD,R16
        LDI     R20,0X00
        LDI     R21,0X00
        RCALL   SCAN_4LA
START:  IN      R17,INPORT;đọc data
        ;ctc chuyển sang số BCD
        RCALL   BIN8_BCD
        ;ctc quét 4 LED
        RCALL   SCAN_4LA
        RJMP    START
```

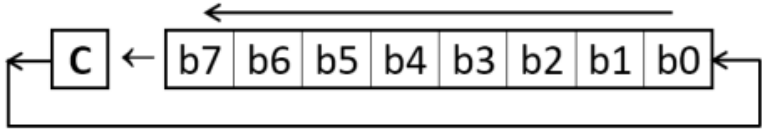
3

```
;-----------------------------------------
;SCAN_4LA hiển thị 4 LED AC bằng phương pháp quét
;Input: R21,R20=số BCD nén(ngàn-trăm),(chục-đơn vị)
;Sử dụng ctc BCD_UNP tách số BCD nén thành không nén
;Sử dụng ctc DELAY_US,GET_7SEG
;Sử dụng R17,R18,R19,X
;-----------------------------------------
SCAN_4LA:
RCALL   BCD_UNP          ;ctc chuyển số BCD nén thành không nén
LDI     R18,4            ;R18 đếm số lần quét LED
LDI     R19,0XFE         ;mã quét anode bắt đầu LED0 (LED đơn vị)
LDI     XH,HIGH(SR_ADR)  ;X trỏ địa chỉ đầu SRAM (ví dụ: SR_ADR = 0x100)
LDI     XL,LOW(SR_ADR)
LOOP:
LDI     R17,0X0F         ;tắt các khóa BJT (tắt tất cả các đèn LED)
OUT     OUTPORT,R17
SBI     PORTC,1          ;mở U3 (IC chốt điều khiển khóa BJT)
CBI     PORTC,1          ;khóa U3 (IC chốt điều khiển khóa BJT)
```

4

```
LD      R17,X+          ;nạp số BCD từ SRAM
RCALL   GET_7SEG        ;lấy mã 7 đoạn
OUT     OUTPORT,R17     ;xuất mã 7 đoạn
SBI     PORTC,0         ;mở U2 (IC chốt data xuất ra LED 7 đoạn)
CBI     PORTC,0         ;khóa U2 (IC chốt data xuất ra LED 7 đoạn)
OUT     OUTPORT,R19     ;xuất mã quét anode LED
SBI     PORTC,1         ;mở U3 (IC chốt điều khiển khóa BJT)
CBI     PORTC,1         ;khóa U3 (IC chốt điều khiển khóa BJT)
RCALL   DELAY_US        ;tạo trễ 1ms sáng đèn (thời gian quét LED)
SEC                     ;C=1 chuẩn bị quay trái
ROL     R19             ;mã quét anode kế tiếp
DEC     R18             ;đếm số lần quét
BRNE    LOOP            ;thoát khi quét đủ 4 lần
RET
```

ROL Rd



5

```
BCD_UNP:
LDI     XH,HIGH(SR_ADR)  ;X trỏ địa chỉ đầu SRAM
LDI     XL,LOW(SR_ADR)
MOV     R17,R20          ;lấy số BCD nén trọng số thấp
ANDI    R17,0X0F         ;lấy số BCD thấp
ST      X+,R17           ;cất vào SRAM, tăng địa chỉ SRAM
MOV     R17,R20          ;lấy lại số BCD
SWAP    R17              ;hoán vị 2 số BCD
ANDI    R17,0X0F         ;lấy số BCD cao
ST      X+,R17           ;cất vào SRAM, tăng địa chỉ SRAM
MOV     R17,R21          ;thực hiện tương tự với số BCD nén cất trong R21
ANDI    R17,0X0F
ST      X+,R17
MOV     R17,R21
SWAP    R17
ANDI    R17,0X0F
ST      X+,R17
RET
```

Chương trình con BCD_UNP: có chức năng chuyển 2 số **BCD nén** trong thanh ghi R20 (chục_đơn vị) và R21 (ngàn_trăm) thành 4 số **BCD không nén**. Kết quả tương ứng với hàng đơn vị, hàng chục, hàng trăm, hàng ngàn được lưu lần lượt trong 4 ô nhớ SRAM có địa chỉ: SR_ADR, SR_ADR + 1, SR_ADR + 2, SR_ADR + 3.

6

```
;---------------------------------------
;GET_7SEG tra mã 7 đoạn từ data đọc vào
;Input R17=mã Hex,Output R17=mã 7 đoạn
;---------------------------------------
GET_7SEG:
LDI     ZH,HIGH(TAB_7SA<<1)     ;Z trỏ địa chỉ đầu bảng tra mã 7 đoạn
LDI     ZL,LOW(TAB_7SA<<1)      ;trong flash ROM
ADD     R30,R17                 ;cộng offset vào ZL
LDI     R17,0
ADC     R31,R17                 ;cộng carry vào ZH
LPM     R17,Z                   ;lấy mã 7 đoạn
RET
```

7

```
;----------------------------------------
;BIN8_BCD chuyển số nhị phân 8 bit sang số BCD 3 digit
;Input R17=số nhị phân 8 bit
;Output R21,R20=số BCD nén,R21 trọng số cao
;Sử dụng ctc DIV8_8,R16=10 số chia
;----------------------------------------
BIN8_BCD:
CLR     R20     ;xóa các thanh ghi kết quả
CLR     R21
LDI     R16,10  ;R16=số chia
RCALL   DIV8_8  ;ctc chia 2 số nhị phân 8 bit
MOV     R20,R16 ;R20=dư số phép chia đầu
LDI     R16,10
RCALL   DIV8_8
SWAP    R16     ;chuyển dư số phép chia lần 2 lên 4 bit cao
OR      R20,R16 ;dán dư số phép chia lần 2 vào 4 bit cao của R20
MOV     R21,R17 ;R21=thương số sau cùng.
RET
```

8

```
;----------------------------------------
;DIV8_8 chia 2 số Hex 8 bit
;Input R17= số bị chia,R16=số chia
;Output R17=thương số,R16=dư số
;Sử dụng R15
;----------------------------------------
DIV8_8:
        CLR     R15     ;R15=thương số
GT_DV:  SUB     R17,R16 ;trừ số bị chi cho số chia
        BRCS    LT_DV   ;C=1 không chia được
        INC     R15     ;tăng thương số thêm 1
        RJMP    GT_DV   ;thực hiện tiếp
LT_DV:  ADD     R17,R16 ;lấy lại dư số
        MOV     R16,R17 ;R16=dư số
        MOV     R17,R15 ;R17=thương số
        RET
```

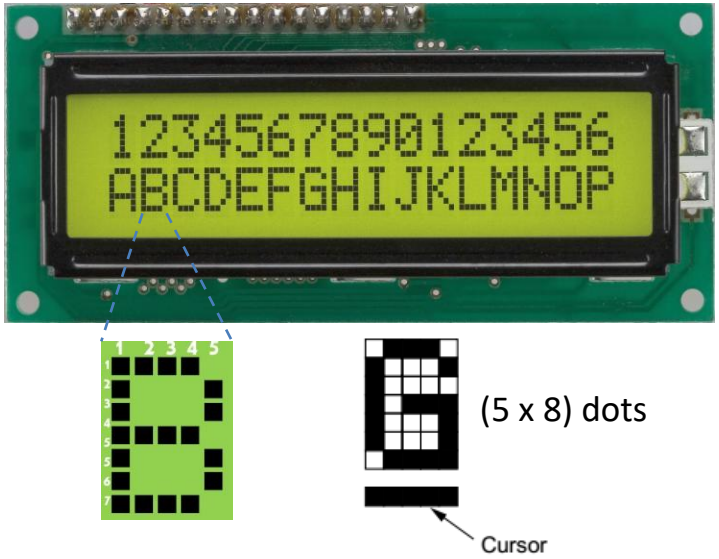
9

```
;-----------------------------------------------------
;DELAY_US tạo thời gian trễ Td = R16 x 100 (µs) (Fosc=8MHz, CKDIV8 = 1)
;Input:R16 hệ số nhân thời gian trễ 1 đến 255
;-----------------------------------------------------
DELAY_US:
          LDI     R16,10          ;tạo trễ 1ms
          MOV     R15,R16         ;1MC nạp data cho R15
          LDI     R16,200         ;1MC sử dụng R16
L1:       MOV     R14,R16         ;1MC nạp data cho R14
L2:       DEC     R14             ;1MC
          NOP                     ;1MC
          BRNE    L2              ;2/1MC
          DEC     R15             ;1MC
          BRNE    L1              ;2/1MC
          RET                     ;4MC
;-----------------------------------------------------
TAB_7SA: .DB 0XC0,0XF9,0XA4,0XB0,0X99,0X92,0X82,0XF8,0X80,0X90,0X88,0X83
          .DB 0XC6,0XA1,0X86,0X8E
```

6.4 Giao tiếp với LCD (Liquid Crystal Display) 16 x 2

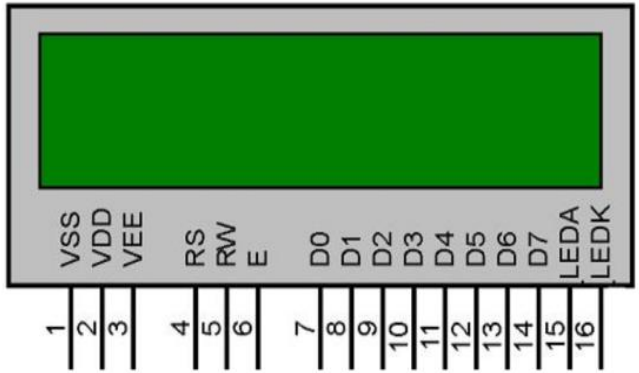
- Để có thể hiển thị đa dạng ký tự và hình ảnh, thường sử dụng LCD ký tự hay LCD đồ họa làm bộ hiển thị.
- LCD ký tự (Alphanumeric LCD) chuyên dùng hiển thị các ký tự mã ASCII và một số ký tự đặc biệt như bảng chữ cái Hy Lạp, ký hiệu toán, ký hiệu điều khiển…
- LCD đồ họa (Graphic LCD) có cấu trúc ma trận điểm nên có thể hiển thị cả ký tự và hình ảnh.

LCD ký tự 16 x 2
(sử dụng IC lái HD44780)



LCD đồ họa 128 x 64 dots
(sử dụng IC lái KS0108)

6.4 Giao tiếp với LCD 16 x 2



- **Các chân điều khiển: RS, RW, E.**
  - RS (Register Select):
    - RS = 0: truy xuất thanh ghi lệnh (IR)
    - RS = 1: truy xuất thanh ghi data (DR)
  - RW (Read/Write):
    - $R/\overline{W}$ = 1: Chế độ đọc (Read)
    - $R/\overline{W}$ = 0: Chế độ ghi (Write)
  - E (Enable): chân cho phép
    - E = 1 → 0 (cạnh xuống): cho phép
    - E = 0: không cho phép
- **Các chân data/command: D0 – D7.**

Chân	Ký hiệu	Chức năng	Mô tả
1	VSS	Nguồn	GND
2	VDD	Nguồn	Nguồn(5V)
3	VEE	Điều khiển	Điều chỉnh độ tương phản,thường mắc qua chiết áp thay đổi từ 0 -5V
4	RS	Điều khiển	RS=0: truy xuất lệnh RS=1: truy xuất data
5	RW	Điều khiển	RW=1: truy xuất đọc RW=0: truy xuất ghi
6	E	Điều khiển	E=↓: cho phép truy xuất LCD E=0: cấm truy xuất LCD
7 -14	D0 - D7	Data/ Command	RS=1: data RS=0: lệnh
15	LEDA	Anode LED	Anode LED nền
16	LEDK	Cathode LED	Cathode LED nền

6.4 Giao tiếp với LCD 16 x 2

Một số mã lệnh thông dụng (điều khiển RS=0, $R/\overline{W} = 0$)

Mã (HEX)	Chức năng	Thời gian thực thi
01	Xóa màn hình (Clear display) (tự về vị trí đầu dòng 1)	1.52ms
02	Trở về vị trí đầu dòng (Return home)	1.52ms
04	Dịch con trỏ sang trái (khi ghi/đọc data)	37µs
05	Dịch màn hình sang phải (khi ghi/đọc data)	37µs
06	Dịch con trỏ sang phải (khi ghi/đọc data)	37µs
07	Dịch màn hình sang trái (khi ghi/đọc data)	37µs
08	Tắt màn hình, tắt con trỏ	37µs
0A	Tắt màn hình, hiện con trỏ	37µs
0C	Hiện màn hình, tắt con trỏ	37µs
0E	Hiện màn hình,không chớp ký tự chỉ bởi con trỏ	37µs
0F	Hiện màn hình, chớp ký tự chỉ bởi con trỏ	37µs

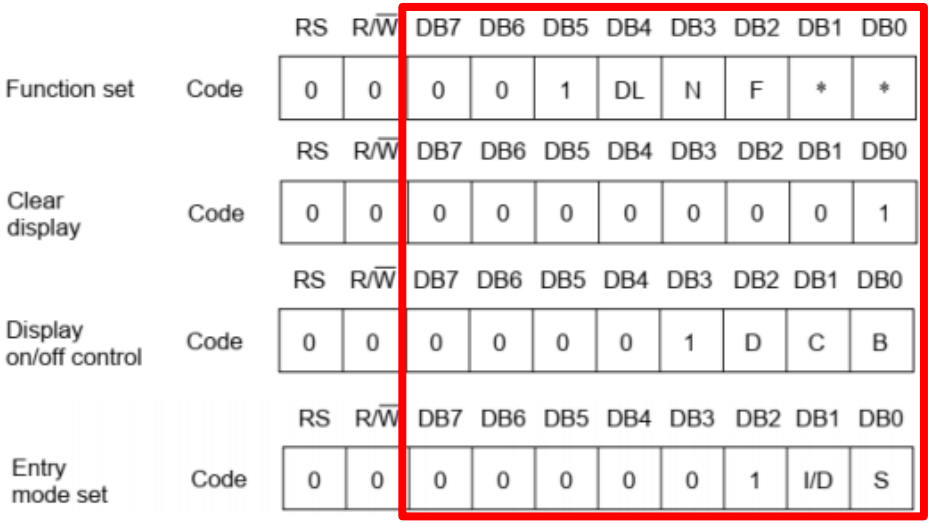
Thời gian thực thi ứng với tần số f_{osc} = 270KHz (Xem thêm trong datasheet HD44780)

6.4 Giao tiếp với LCD 16 x 2

Một số mã lệnh thông dụng (điều khiển RS=0, $R/\overline{W} = 0$)

Mã (HEX)	Chức năng	Thời gian thực thi
10	Dịch con trỏ sang trái (không thay đổi nội dung DDRAM)	37µs
14	Dịch con trỏ sang phải (không thay đổi nội dung DDRAM)	37µs
18	Dịch màn hình sang trái (không thay đổi nội dung DDRAM)	37µs
1C	Dịch màn hình sang phải (không thay đổi nội dung DDRAM)	37µs
28	Đặt chức năng **giao tiếp 4 bit cao**, 2 dòng, 5 x 8 dots	37µs
38	Đặt chức năng **giao tiếp 8 bit**, 2 dòng, 5 x 8 dots	37µs
80	Chuyển con trỏ về đầu dòng 1	37µs
C0	Chuyển con trỏ về đầu dòng 2	37µs

Đọc/ghi bộ nhớ DDRAM/CGRAM: **37µs** (ứng với tần số f_{osc} = 270KHz)



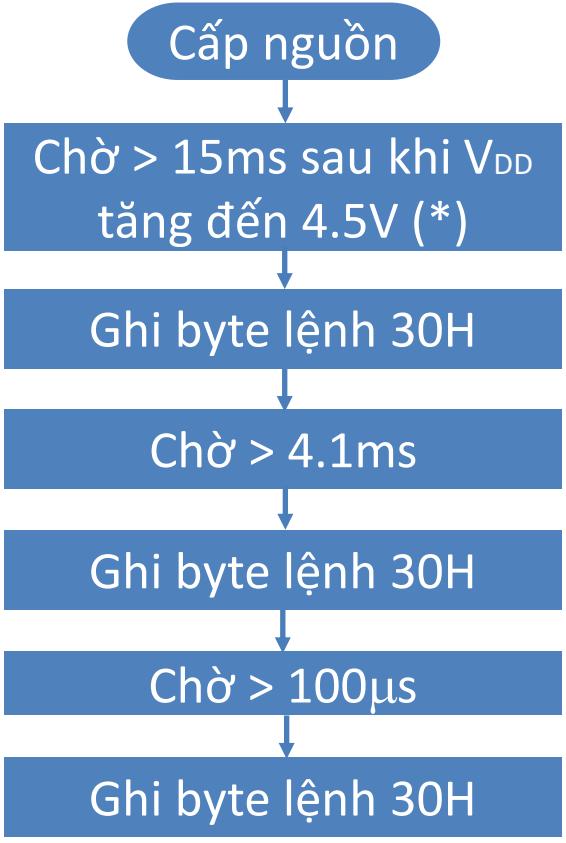
		RS	R/$\overline{W}$	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0
Function set	Code	0	0	0	0	1	DL	N	F	*	*
Clear display	Code	0	0	0	0	0	0	0	0	0	1
Display on/off control	Code	0	0	0	0	0	0	1	D	C	B
Entry mode set	Code	0	0	0	0	0	0	0	1	I/D	S

(*: tùy định)

Giải thích ý nghĩa một số lệnh. Xem thêm trong Chương 6 giáo trình VXL và datasheet HD44780.

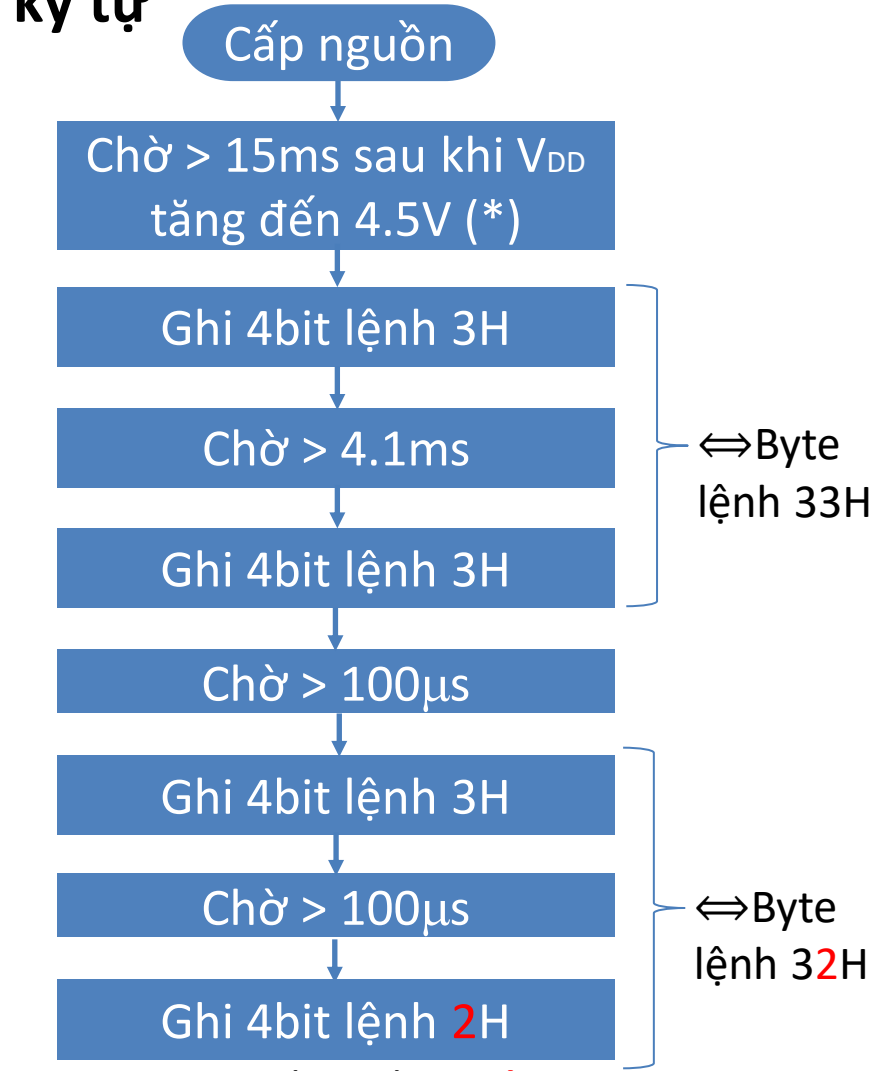
- **Lệnh 1: Function set:** đặt cấu hình làm việc LCD kết nối 8/4 bit (DL), số dòng (N), ma trận 5x8 hay 5x10 dots (F).
  - DL=1: giao tiếp data 8 bit D7-D0, DL=0 data 4 bit D7-D4
  - N=1: đặt 2 dòng, N=0: đặt 1 dòng
  - F=1: font 5x10 (N=0), F=0: font 5x8
- **Lệnh 2: Clear display:** xóa toàn bộ màn hình và con trỏ tự động dời về đầu dòng 1.
  - Ghi mã lệnh 01H
- **Lệnh 3: Display on/off control:** điều khiển màn hình và con trỏ.
  - D=1 màn hình on, D=0 màn hình off
  - C=1 con trỏ on, C=0 con trỏ off
  - B=1 ký tự và con trỏ chớp, B=0 ký tự và con trỏ không chớp
- **Lệnh 4: Entry mode set:** chọn mode dịch con trỏ/màn hình.
  - I/D=1/0: tăng/giảm địa chỉ DDRAM thêm/bớt 1 khi truy xuất DDRAM
  - S=1: dịch màn hình sang phải (I/D=0) hay trái (I/D=1), S=0: không dịch màn hình

Các lệnh reset cấp nguồn cho LCD ký tự



Các lệnh reset cấp nguồn **chế độ 8 bit**

(*) Sử dụng nguồn 3V: Chờ > 40ms sau khi VDD tăng đến 2.7V



Các lệnh reset cấp nguồn **chế độ 4 bit**

Các lệnh khởi động cho LCD ký tự (sau khi reset cấp nguồn)

- **Function set:** 2 dòng, font 5x8 dots
  - Chế độ 8 bit: mã lệnh=38H
  - Chế độ 4 bit: mã lệnh=28H
- **Clear display:** xóa màn hình mã lệnh=01H
- **Display on/off control:** màn hình on, xóa con trỏ mã lệnh=0CH
- **Entry mode set:** con trỏ dịch phải,địa chỉ DDRAM tăng 1 khi ghi data, màn hình không dịch. mã lệnh=06H

Phân biệt giữa mã lệnh (command) và dữ liệu (data)

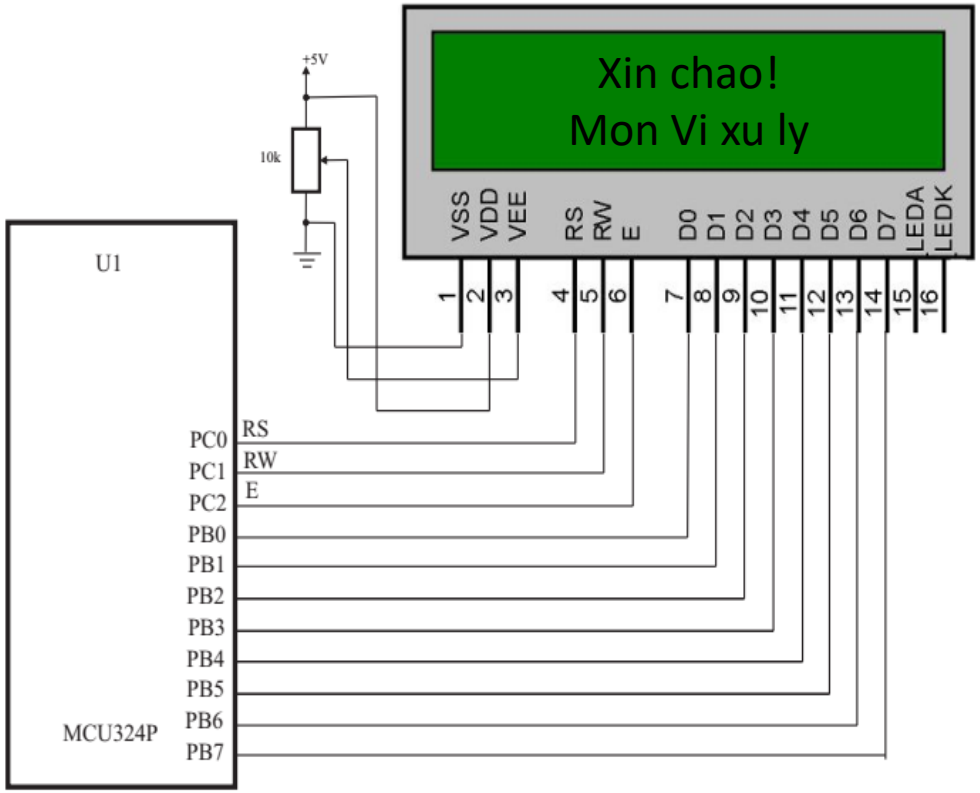
- Ghi lệnh xuống LCD: chọn thanh ghi lệnh (IR) với **RS = 0**
- Ghi dữ liệu xuống LCD (hiển thị lên màn hình LCD): chọn thanh ghi dữ liệu (DR) với **RS = 1.** Lưu ý: dữ liệu hiển thị lên LCD là **mã ASCII** của ký tự tương ứng.
- Trong cả 2 trường hợp trên, tạo xung **cạnh xuống** ở chân E để cho phép LCD bắt đầu thực thi.

❖ Một số lưu ý:

- **Giao tiếp 8 bit:** kết nối 8 chân D0...D7 với 8 chân port tương ứng của MCU. Mỗi lệnh giao tiếp chỉ cần truy xuất 1 byte kết hợp 3 tín hiệu điều khiển (RS, RW, E).
- **Giao tiếp 4 bit:** kết nối 4 chân D4...D7 với 4 chân port tương ứng của MCU. Mỗi lệnh giao tiếp cần truy xuất 2 lần, mỗi lần 4 bit kết hợp với 3 tín hiệu điều khiển (RS, RW, E). Truy xuất 4 bit cao trước, chờ > 100µs, truy xuất tiếp 4 bit thấp.
- Thay vì đọc cờ BF (Busy Flag), có thể delay từ 50µs đến 1.6ms tùy loại lệnh, chờ LCD thực hiện xong lệnh.

Ví dụ 13: Cho sơ đồ giao tiếp giữa MCU324P với LCD 16x2 **giao tiếp 8 bit** như bên dưới. Viết một chương trình điều khiển LCD 16x2 thực hiện các công việc sau:

- Viết chương trình con có chức năng reset cấp nguồn cho LCD.
- Viết chương trình con có chức khởi động LCD.
- Viết chương trình hoàn chỉnh hiển thị ra LCD 2 dòng ký tự như sau:



```
.EQU    OUTPORT=PORTB       ;PORTB data
.EQU    INPORT=PINB
.EQU    IOSETB=DDRB
.EQU    CONT=PORTC          ;PORTC điều khiển
.EQU    CONT_DR=DDRC
.EQU    CONT_IN=PINC
.EQU    RS=0                ;bit RS
.EQU    RW=1                ;bit RW
.EQU    E=2                 ;bit E
.EQU    CR=$0D              ;mã báo hiệu kết thúc dòng 1
.EQU    NULL=$00            ;mã báo hiệu kết thúc dòng 2
```

1

2

```
        .ORG    0
        RJMP    MAIN
        .ORG    0X40
MAIN:   LDI     R16,HIGH(RAMEND)        ;đưa stack lên vùng địa chỉ cao
        OUT     SPH,R16
        LDI     R16,LOW(RAMEND)
        OUT     SPL,R16
        LDI     R16,0X07
        OUT     CONT_DR,R16             ;khai báo PC0,PC1,PC2 là output
        CBI     CONT,RS                 ;RS=PC0=0
        CBI     CONT,RW                 ;RW=PC1=0  truy xuất ghi
        CBI     CONT,E                  ;E=PC2=0 cấm LCD
        LDI     R16,0XFF
        OUT     IOSETB,R16              ;khai báo outport
```

Lưu ý:

```
CBI     CONT,RS     ⇔     CBI     PORTC,0; RS = 0
CBI     CONT,RW     ⇔     CBI     PORTC,1; RW = 0
CBI     CONT,E      ⇔     CBI     PORTC,2; E = 0
```

```
        ;-------------------------------
        ;khởi tạo LCD gồm:
        ;reset cấp nguồn
        ;cấu hình LCD hoạt động chế độ 8 bit
        ;-------------------------------
        RCALL   POWER_RESET_LCD8 ;reset cấp nguồn LCD 8 bit
        RCALL   INIT_LCD8        ;ctc khởi động LCD 8 bit
        ;-------------------------------
        CBI     CONT,RS          ;RS=0 ghi lệnh
        LDI     R17,$83          ;con trỏ bắt đầu ở dòng 1 vị trí thứ 4
        RCALL   OUT_LCD
        LDI     R16,1            ;chờ 100µs
        RCALL   DELAY_US
        LDI     ZH,HIGH(TAB<<1)  ;Z trỏ đầu bảng tra ký tự
        LDI     ZL,LOW(TAB<<1)
LINE1:  LPM     R17,Z+           ;lấy mã ASCII ký tự từ Flash ROM
        CPI     R17,CR           ;kiểm tra ký tự xuống dòng?
        BREQ    DOWN             ;ký tự CR xuống dòng
        SBI     CONT,RS          ;RS=1 ghi data hiển thị LCD
        RCALL   OUT_LCD          ;ghi mã ASCII ký tự ra LCD
        LDI     R16,1            ;chờ 100µs
        RCALL   DELAY_US
        RJMP    LINE1            ;tiếp tục hiển thị dòng 1
```

```
DOWN:   CBI     CONT,RS         ;RS=0 ghi lệnh
        LDI     R17,$C2         ;con trỏ bắt đầu ở dòng 2 vị trí thứ 3
        RCALL   OUT_LCD
        LDI     R16,1           ;chờ 100µs
        RCALL   DELAY_US
LINE2:  LPM     R17,Z+          ;lấy mã ASCII ký tự từ Flash ROM
        CPI     R17,NULL        ;kiểm tra ký tự kết thúc
        BREQ    HERE            ;ký tự NULL thoát
        SBI     CONT,RS         ;RS=1 ghi data hiển thị LCD
        RCALL   OUT_LCD         ;ghi mã ASCII ký tự ra LCD
        LDI     R16,1           ;chờ 100µs
        RCALL   DELAY_US
        RJMP    LINE2           ;tiếp tục hiển thị dòng 2
HERE:   RJMP    HERE
```

```
;-----------------------------------------------------------------
;Các lệnh reset cấp nguồn LCD 8 bit
;Chờ hơn 15ms
;Ghi mã lệnh 30H lần 1, chờ ít nhất 4.1ms
;Ghi mã lệnh 30H lần 2, chờ ít nhất 100µs
;Ghi mã lệnh 30H lần 3, chờ ít nhất 100µs
;-----------------------------------------------------------------
POWER_RESET_LCD8:
        LDI     R16,200 ;delay 20ms
        RCALL   DELAY_US ;ctc delay 100µsxR16
        ;Ghi mã lệnh 30H lần 1, chờ 4.2ms
        CBI     CONT,RS ;RS=0 ghi lệnh
        LDI     R17,$30 ;mã lệnh=$30 lần 1,RS=RW=E=0
        RCALL   OUT_LCD ;ctc ghi ra LCD
        LDI     R16,42 ;delay 4.2ms
        RCALL   DELAY_US
```

5

```
;Ghi mã lệnh 30H lần 2, chờ 200µs
        CBI     CONT,RS ;RS=0 ghi lệnh
        LDI     R17,$30 ;mã lệnh=$30 lần 2
        RCALL   OUT_LCD
        LDI     R16,2 ;delay 200µs
        RCALL   DELAY_US
;Ghi mã lệnh 30H lần 3, chờ 200µs
        CBI     CONT,RS ;RS=0 ghi lệnh
        LDI     R17,$30
        RCALL   OUT_LCD
        LDI     R16,2 ;delay 200µs
        RCALL   DELAY_US
        RET
```

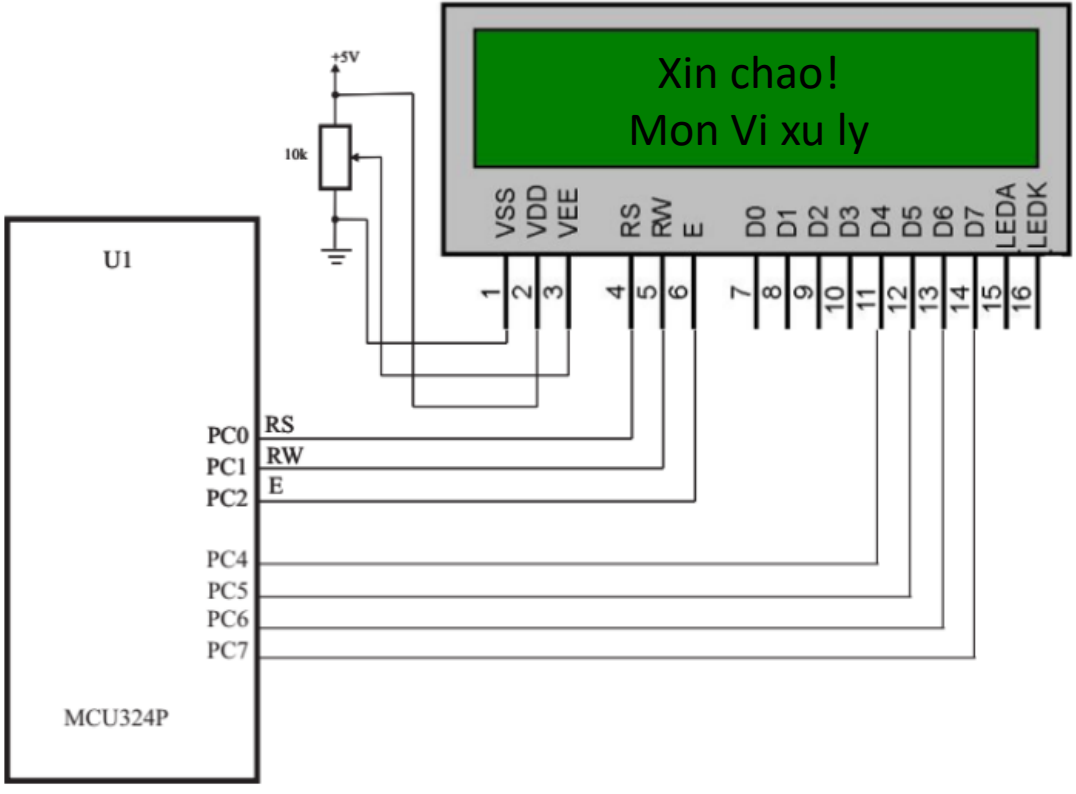
6

```
;-----------------------------------------------------------
;INIT_LCD8 khởi động LCD ghi 4 byte mã lệnh
;Function set: 0x38: 8 bit, 2 dòng, font 5x8
;Clear display: 0x01: xóa màn hình
;Display on/off control: 0x0C: màn hình on, con trỏ off
;Entry mode set: 0x06: dịch phải con trỏ, địa chỉ DDRAM tăng 1 khi ghi data
;-----------------------------------------------------------
INIT_LCD8:  CBI     CONT,RS     ;RS=0 ghi lệnh
            LDI     R17,0x38    ;chế độ giao tiếp 8 bit, 2 dòng, font 5x8
            RCALL   OUT_LCD
            LDI     R16,1       ;chờ 100µs
            RCALL   DELAY_US
            CBI     CONT,RS     ;RS=0 ghi lệnh
            LDI     R17,0x01    ;xóa màn hình
            RCALL   OUT_LCD
            LDI     R16,20      ;chờ 2ms sau lệnh Clear display
            RCALL   DELAY_US
            CBI     CONT,RS     ;RS=0 ghi lệnh
            LDI     R17,0x0C    ;màn hình on, con trỏ off
            RCALL   OUT_LCD
            LDI     R16,1       ;chờ 100µs
            RCALL   DELAY_US
            CBI     CONT,RS     ;RS=0 ghi lệnh
            LDI     R17,0x06    ;dịch phải con trỏ, địa chỉ DDRAM tăng 1 khi ghi data
            RCALL   OUT_LCD
            LDI     R16,1       ;chờ 100µs
            RCALL   DELAY_US
            RET
```

```
;-------------------------------------------------
;OUT_LCD ghi mã lệnh/data ra LCD
;Input: R17 chứa mã lệnh/data
;-------------------------------------------------
OUT_LCD:  OUT      OUTPORT,R17        ;1MC,ghi lệnh/data ra LCD
          SBI      CONT,E             ;2MC,xuất xung cho phép LCD
          CBI      CONT,E             ;2MC
          RET
;-------------------------------------------------------
;DELAY_US tạo thời gian trễ =R16x100µs(Fosc=8MHz, CKDIV8 = 1)
;Input:R16 hệ số nhân thời gian trễ 1 đến 255
;-------------------------------------------------------
DELAY_US:  MOV     R15,R16            ;1MC nạp data cho R15
          LDI      R16,200            ;1MC sử dụng R16
L1:       MOV      R14,R16            ;1MC nạp data cho R14
L2:       DEC      R14                ;1MC
          NOP                         ;1MC
          BRNE     L2                 ;2/1MC
          DEC      R15                ;1MC
          BRNE     L1                 ;2/1MC
          RET                         ;4MC
          .ORG     0X0200
;-----------------------------------------------------------
TAB:      .DB "Xin chao!",$0D,"Mon Vi xu ly",$00
```

Ví dụ 14: Cho sơ đồ giao tiếp giữa MCU324P với LCD 16x2 **giao tiếp 4 bit** như bên dưới. Viết một chương trình điều khiển LCD 16x2 thực hiện các công việc sau:

- Viết chương trình con có chức năng reset cấp nguồn cho LCD.
- Viết chương trình con có chức khởi động LCD.
- Viết chương trình hoàn chỉnh hiển thị ra LCD 2 dòng ký tự như sau:



1

```
.EQU    LCD=PORTC     ;PORTC giao tiếp bus LCD 16 x 2
.EQU    LCD_DR=DDRC
.EQU    LCD_IN=PINC
.EQU    RS=0                    ;bit RS
.EQU    RW=1                    ;bit RW
.EQU    E=2                     ;bit E
.EQU    CR=$0D                  ;mã báo hiệu kết thúc dòng 1
.EQU    NULL=$00                ;mã báo hiệu kết thúc dòng 2
```

```
        .ORG    0
        RJMP    MAIN
        .ORG    0X40
MAIN:   LDI     R16,HIGH(RAMEND)        ;đưa stack lên vùng địa chỉ cao
        OUT     SPH,R16
        LDI     R16,LOW(RAMEND)
        OUT     SPL,R16
        LDI     R16,0XFF
        OUT     LCD_DR,R16              ;khai báo PORTC là output
        CBI     LCD,RS                  ;RS=PC0=0
        CBI     LCD,RW                  ;RW=PC1=0  truy xuất ghi
        CBI     LCD,E                   ;E=PC2=0 cấm LCD
```

Lưu ý:

```
CBI     LCD,RS      ⇔     CBI     PORTC,0; RS = 0
CBI     LCD,RW      ⇔     CBI     PORTC,1; RW = 0
CBI     LCD,E       ⇔     CBI     PORTC,2; E = 0
```

```
        ;------------------------------
        ;khởi tạo LCD gồm:
        ;reset cấp nguồn
        ;cấu hình LCD hoạt động chế độ 4 bit
        ;------------------------------
        RCALL   POWER_RESET_LCD4 ;reset cấp nguồn LCD 4 bit
        RCALL   INIT_LCD4        ;ctc khởi động LCD 4 bit
        ;------------------------------
        CBI     LCD,RS           ;RS=0 ghi lệnh
        LDI     R17,$83          ;con trỏ bắt đầu ở dòng 1 vị trí thứ 4
        RCALL   OUT_LCD4_2
        LDI     R16,1            ;chờ 100µs
        RCALL   DELAY_US
        LDI     ZH,HIGH(TAB<<1)  ;Z trỏ đầu bảng tra ký tự
        LDI     ZL,LOW(TAB<<1)
LINE1:  LPM     R17,Z+           ;lấy mã ASCII ký tự từ Flash ROM
        CPI     R17,CR           ;kiểm tra ký tự xuống dòng?
        BREQ    DOWN             ;ký tự CR xuống dòng
        SBI     LCD,RS           ;RS=1 ghi data hiển thị LCD
        RCALL   OUT_LCD4_2       ;ghi mã ASCII ký tự ra LCD
        LDI     R16,1            ;chờ 100µs
        RCALL   DELAY_US
        RJMP    LINE1            ;tiếp tục hiển thị dòng 1
```

```
DOWN:   CBI     LCD,RS          ;RS=0 ghi lệnh
        LDI     R17,$C2         ;con trỏ bắt đầu ở dòng 2 vị trí thứ 3
        RCALL   OUT_LCD4_2
        LDI     R16,1           ;chờ 100µs
        RCALL   DELAY_US
LINE2:  LPM     R17,Z+          ;lấy mã ASCII ký tự từ Flash ROM
        CPI     R17,NULL        ;kiểm tra ký tự kết thúc
        BREQ    HERE            ;ký tự NULL thoát
        SBI     LCD,RS          ;RS=1 ghi data hiển thị LCD
        RCALL   OUT_LCD4_2      ;ghi mã ASCII ký tự ra LCD
        LDI     R16,1           ;chờ 100µs
        RCALL   DELAY_US
        RJMP    LINE2           ;tiếp tục hiển thị dòng 2
HERE:   RJMP    HERE
```

5

```
;-----------------------------------------------------------------
;Các lệnh reset cấp nguồn LCD 4 bit
;Chờ hơn 15ms
;Ghi 4 bit mã lệnh 30H lần 1, chờ ít nhất 4.1ms
;Ghi 4 bit mã lệnh 30H lần 2, chờ ít nhất 100µs
;Ghi byte mã lệnh 32H, chờ ít nhất 100µs sau mỗi lần ghi 4 bit
;-----------------------------------------------------------------
POWER_RESET_LCD4:
        LDI     R16,200 ;delay 20ms
        RCALL   DELAY_US ;ctc delay 100µsxR16
        ;Ghi 4 bit cao mã lệnh 30H lần 1, chờ 4.2ms
        CBI     LCD,RS ;RS=0 ghi lệnh
        LDI     R17,$30 ;mã lệnh=$30 lần 1,RS=RW=E=0
        RCALL   OUT_LCD4 ;ctc ghi ra LCD 4 bit cao
        LDI     R16,42 ;delay 4.2ms
        RCALL   DELAY_US
        ;Ghi 4 bit cao mã lệnh 30H lần 2, chờ 200µs
        CBI     LCD,RS ;RS=0 ghi lệnh
        LDI     R17,$30 ;mã lệnh=$30 lần 2
        RCALL   OUT_LCD4 ;ctc ghi ra LCD 4 bit cao
        LDI     R16,2 ;delay 200µs
        RCALL   DELAY_US
        ;Ghi byte mã lệnh 32H
        CBI     LCD,RS ;RS=0 ghi lệnh
        LDI     R17,$32
        RCALL   OUT_LCD4_2; ctc ghi 1 byte, mỗi lần 4 bit
        RET
```

```
;-------------------------------------------------------------
;INIT_LCD4 khởi động LCD ghi 4 byte mã lệnh
;Function set: 0x28: 4 bit, 2 dòng, font 5x8
;Clear display: 0x01: xóa màn hình
;Display on/off control: 0x0C: màn hình on, con trỏ off
;Entry mode set: 0x06: dịch phải con trỏ, địa chỉ DDRAM tăng 1 khi ghi data
;-------------------------------------------------------------
INIT_LCD4:  CBI         LCD,RS              ;RS=0 ghi lệnh
            LDI         R17,0x28            ;chế độ giao tiếp 4 bit, 2 dòng, font 5x8
            RCALL       OUT_LCD4_2
            ;-------------------------------------------------------------
            CBI         LCD,RS              ;RS=0 ghi lệnh
            LDI         R17,0x01            ;xóa màn hình
            RCALL       OUT_LCD4_2
            LDI         R16,20              ;chờ 2ms sau lệnh Clear display
            RCALL       DELAY_US
            ;-------------------------------------------------------------
            CBI         LCD,RS              ;RS=0 ghi lệnh
            LDI         R17,0x0C            ;màn hình on, con trỏ off
            RCALL       OUT_LCD4_2
            ;-------------------------------------------------------------
            CBI         LCD,RS              ;RS=0 ghi lệnh
            LDI         R17,0x06            ;dịch phải con trỏ, địa chỉ DDRAM tăng 1 khi ghi data
            RCALL       OUT_LCD4_2
            ;-------------------------------------------------------------
            RET
```

```
;-----------------------------------------------
;OUT_LCD4_2 ghi 1 byte mã lệnh/data ra LCD
;chia làm 2 lần ghi 4bit:  4 bit cao trước, 4 bit thấp sau
;Input: R17 chứa mã lệnh/data,R16
;bit RS=0/1:lệnh/data,bit RW=0:ghi
;Sử dụng ctc OUT_LCD4
;-----------------------------------------------
OUT_LCD4_2:
        IN      R16,LCD         ;đọc PORT LCD
        ANDI    R16,(1<<RS)     ;lọc bit RS
        PUSH    R16             ;cất R16
        PUSH    R17             ;cất R17
        ANDI    R17,$F0         ;lấy 4 bit cao
        OR      R17,R16         ;ghép bit RS
        RCALL   OUT_LCD4        ;ghi ra LCD
        LDI     R16,1           ;chờ 100us
        RCALL   DELAY_US
```

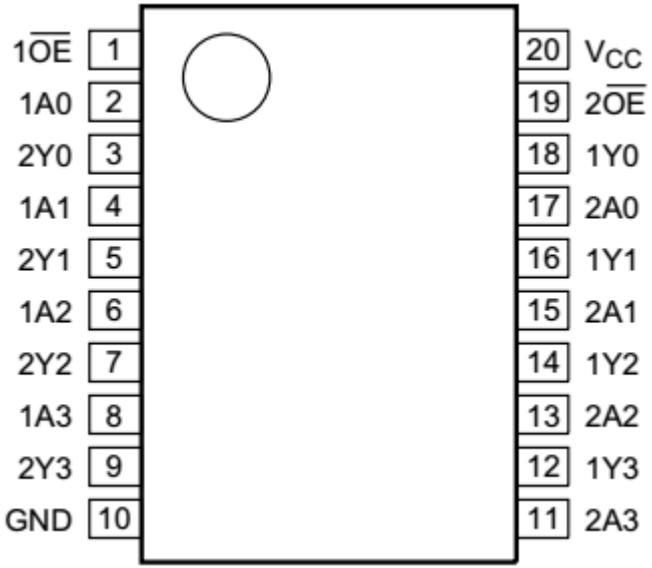
7

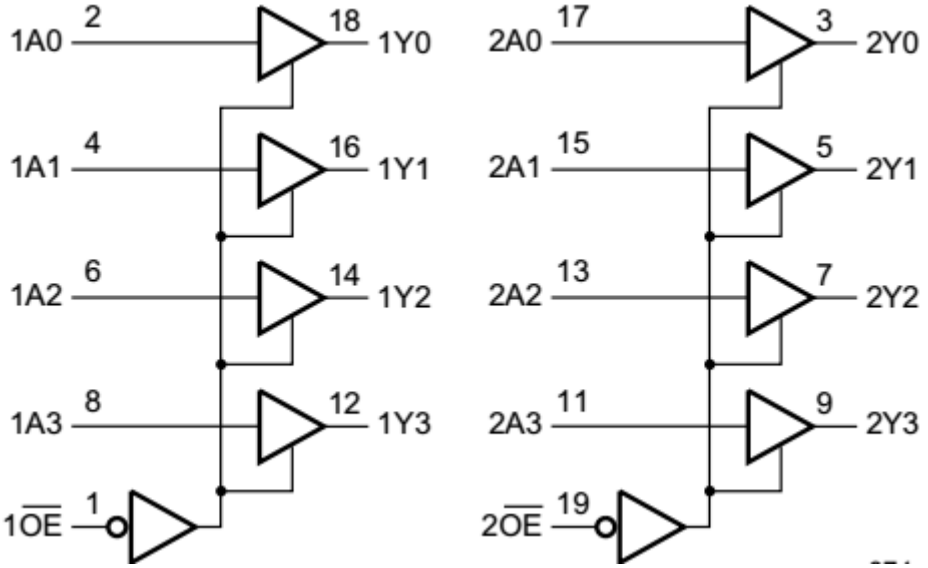
```
        POP     R17             ;phục hồi R17
        POP     R16             ;phục hồi R16
        SWAP    R17             ;đảo 4 bit
;lấy 4 bit thấp chuyển thành cao
        ANDI    R17,$F0
        OR      R17,R16  ;ghép bit RS
        RCALL   OUT_LCD4;ghi ra LCD
        LDI     R16,1    ;chờ 100us
        RCALL   DELAY_US
        RET
```

8

```
;-------------------------------------------------
;OUT_LCD4 ghi mã lệnh/data ra LCD
;Input: R17 chứa mã lệnh/data 4 bit cao
;-------------------------------------------------
OUT_LCD4: OUT       LCD,R17
          SBI       LCD,E
          CBI       LCD,E
          RET
;-------------------------------------------------------
;DELAY_US tạo thời gian trễ =R16x100µs(Fosc=8MHz, CKDIV8 = 1)
;Input:R16 hệ số nhân thời gian trễ 1 đến 255
;-------------------------------------------------------
DELAY_US:  MOV      R15,R16          ;1MC nạp data cho R15
           LDI      R16,200          ;1MC sử dụng R16
L1:        MOV      R14,R16          ;1MC nạp data cho R14
L2:        DEC      R14              ;1MC
           NOP                       ;1MC
           BRNE     L2               ;2/1MC
           DEC      R15              ;1MC
           BRNE     L1               ;2/1MC
           RET                       ;4MC
           .ORG     0X0200
;-----------------------------------------------------------
TAB:       .DB "Xin chao!",$0D,"Mon Vi xu ly",$00
```

6.5 Giao tiếp với IC đệm 3 trạng thái 74HC244





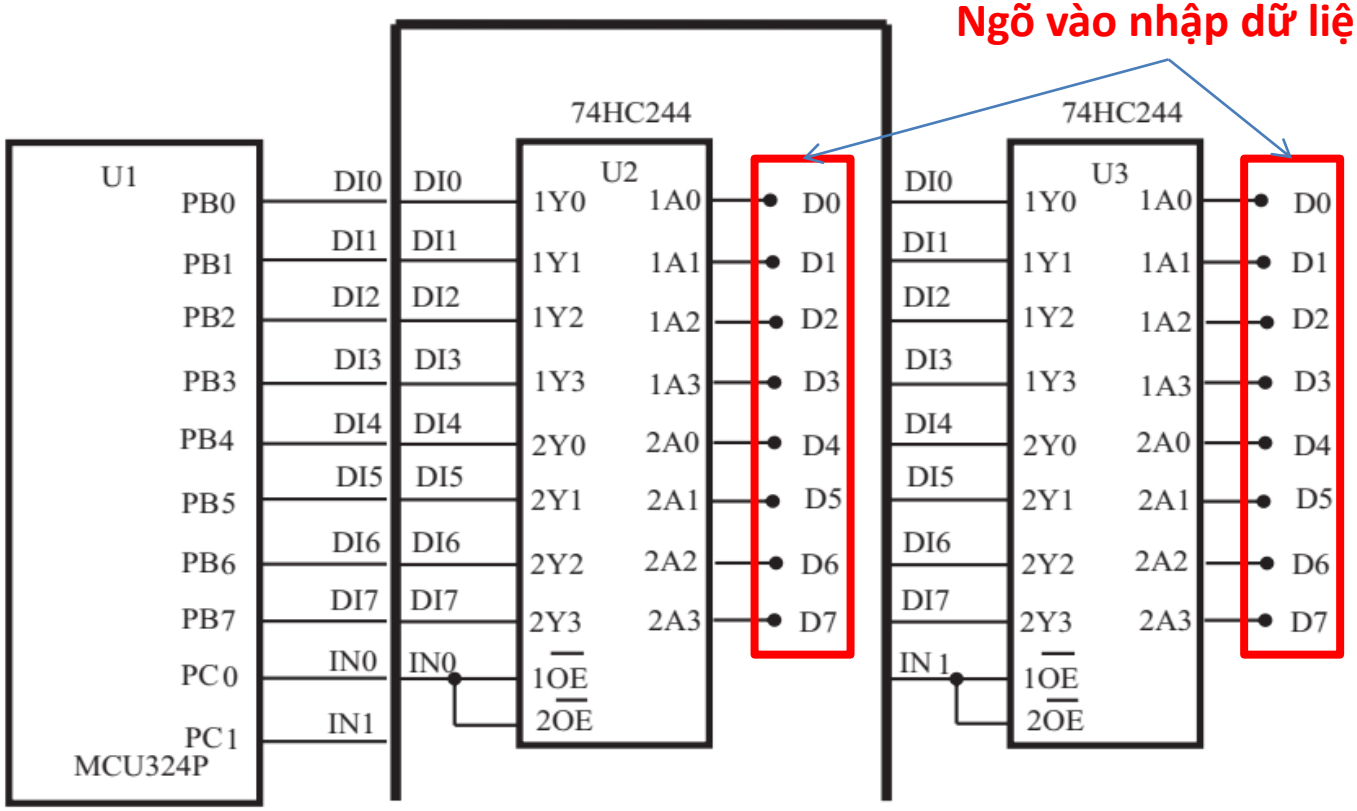
Bảng hoạt động IC 74HC244

Inputs *(i=1,2;n=0..3)*		Outputs
$i\overline{OE}$	iAn	iYn
H	X	Hi-Z
L	L	L
L	H	H

- $\overline{OE}$ cấm/cho phép xuất (tích cực thấp).
  - $\overline{OE}$ = 1 cấm: iYn=tổng trở cao (Hi-Z)
  - $\overline{OE}$ = 0 cho phép xuất: iYn=iAn

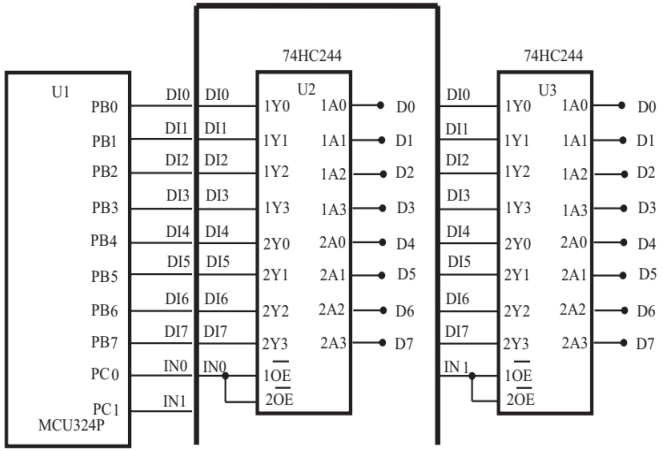
6.5 Giao tiếp với IC đệm 3 trạng thái 74HC244

Ví dụ 15: Viết đoạn chương trình cất data từ đệm U2 vào thanh ghi R17 và data từ đệm U3 vào thanh ghi R18



6.5 Giao tiếp với IC đệm 3 trạng thái 74HC244

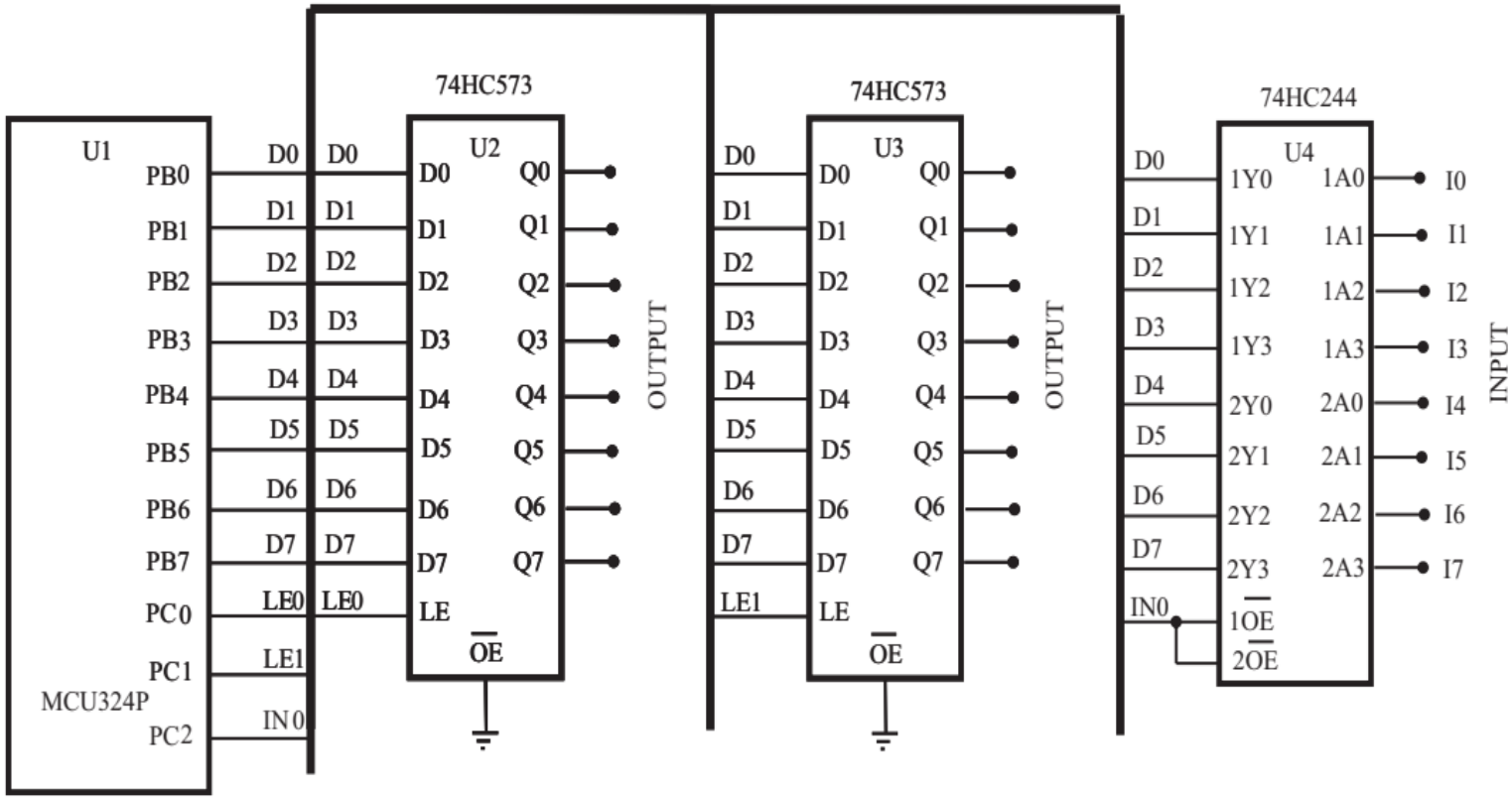
Ví dụ 15: Viết đoạn chương trình cất data từ đệm U2 vào thanh ghi R17 và data từ đệm U3 vào thanh ghi R18




```
LDI   R16, 0x00
OUT   DDRB, R16      ;PORTB là port nhập
SBI   DDRC,0         ;PC0 là ngõ xuất
SBI   DDRC,1         ;PC1 là ngõ xuất
SBI   PORTC,0        ;cấm đệm U2
SBI   PORTC,1        ;cấm đệm U3
CBI   PORTC,0        ;mở đệm U2, data đưa về Port B
IN    R17, PINB      ;cất data vào R17
SBI   PORTC,0        ;cấm đệm U2
CBI   PORTC,1        ;mở đệm U3, data đưa về Port B
IN    R18, PINB      ;cất data vào R17
SBI   PORTC,1        ;cấm đệm U3
```

6.5 Giao tiếp với IC đệm 3 trạng thái 74HC244

Ví dụ 16: Cho sơ đồ mở rộng 2 port ngõ ra và 1 port ngõ vào sử dụng 2 IC 74HC573 và 1 IC 74HC244. Viết một chương trình nhập data từ ngõ vào U4, chuyển 4 bit thấp và 4 bit cao sang mã ASCII, lần lượt xuất ra U2 và U3 tương ứng.



```
        .EQU    OUTPORT=PORTB
        .EQU    INPORT=PINB
        .EQU    IOSET=DDRB
        .ORG    0
        RJMP    MAIN
        .ORG    0X40
MAIN:   LDI     R16,HIGH(RAMEND);đưa stack lên vùng đ/c cao
        OUT     SPH,R16
        LDI     R16,LOW(RAMEND)
        OUT     SPL,R16
        LDI     R16,0X07
        OUT     DDRC,R16        ;khai báo PC0,PC1,PC2 là output
        SBI     PORTC,2         ;khóa ngõ vào U4
        CBI     PORTC,0         ;khóa ngõ ra U2
        CBI     PORTC,1         ;khóa ngõ ra U3
```

2

```
START:  LDI     R16,0X00
        OUT     IOSET,R16       ;khai báo PORT input
        CBI     PORTC,2         ;mở U4
        IN      R17,INPORT      ;đọc data
        SBI     PORTC,2         ;khóa U4
        PUSH    R17             ;cất data
        LDI     R16,0XFF
        OUT     IOSET,R16       ;khai báo PORT output
        ANDI    R17,0X0F        ;che 4 bit thấp data
        RCALL   HEX_ASC         ;chuyển sang mã ASCII
        OUT     OUTPORT,R18     ;xuất mã ASCII
        SBI     PORTC,0         ;mở U2
        CBI     PORTC,0         ;khóa U2
        POP     R17             ;phục hồi data
        SWAP    R17             ;hoán vị sang 4 bit thấp
        ANDI    R17,0X0F        ;che 4 bit thấp data
        RCALL   HEX_ASC         ;chuyển sang mã ASCII
        OUT     OUTPORT,R18     ;xuất mã ASCII
        SBI     PORTC,1         ;mở U3
        CBI     PORTC,1         ;khóa U3
        RJMP    START
```

```
;---------------------------------------
;HEX_ASC chuyển từ mã Hex sang mã ASCII
;Input R17=mã Hex,Output R18=mã ASCII
;---------------------------------------
HEX_ASC:CPI     R17,0X0A
        BRCS    NUM
        LDI     R18,0X37
        RJMP    CHAR
NUM:    LDI     R18,0X30
CHAR:   ADD     R18,R17
        RET
```

❖ **Nhận xét:**

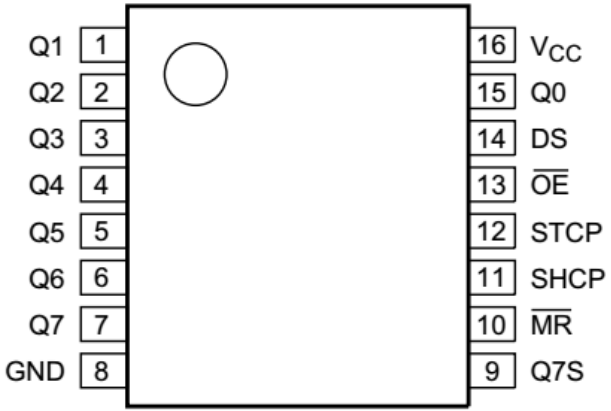
- Số HEX từ 0H đến 9H + 30H → Mã ASCII tương ứng từ 30H đến 39H.
- Số HEX từ AH đến FH + 37H → Mã ASCII tương ứng từ 41H đến 46H (tương ứng với mã ASCII của các ký tự A,….,F).

Mã HEX	0	1	2	3	4	5	6	7	8	9
Mã ASCII	30H	31H	32H	33H	34H	35H	36H	37H	38H	39H

Mã HEX	A	B	C	D	E	F
Mã ASCII	41H	42H	43H	44H	45H	46H

6.6 Giao tiếp với thanh ghi dịch 74HC595

- **Ứng dụng:** mở rộng port xuất data từ nối tiếp chuyển sang song song.



Ký hiệu	Mô tả
Q0,Q1,…,Q7	Ngõ ra song song
Q7S	Ngõ ra nối tiếp
DS	Ngõ vào nối tiếp
$\overline{MR}$	Ngõ vào reset chính tích cực mức 0
SHCP	Ngõ vào xung clock ghi dịch
STCP	Ngõ vào xung clock lưu trữ
$\overline{OE}$	Ngõ vào cho phép xuất tích cực mức 0

Ngõ vào điều khiển				Ngõ vào	Ngõ ra		Chức năng
SHCP	STCP	$\overline{OE}$	$\overline{MR}$	DS	Q7S	Qn	
X	X	L	L	X	L	NC	Reset thanh ghi, ngõ ra không đổi.
X	↑	L	L	X	L	L	Reset thanh ghi và ngõ ra.
X	X	H	L	X	L	Hi-Z	Reset thanh ghi ngõ ra tổng trở cao.
↑	X	L	H	DI	Q6S	NC	Mode ghi dịch, DI dịch vào tầng ghi dịch 0, ngõ ra tầng thấp dịch vào tầng cao, ngõ ra song song không đổi.
X	↑	L	H	X	NC	QnS	Mode xuất data, ngõ ra các tầng ghi dịch được phép xuất ngõ ra song song.
↑	↑	L	H	DI	Q6S	QnS	Mode ghi dịch và xuất data ngõ ra bằng giá trị ghi dịch trước đó.

NC = No Change: Không thay đổi

↑ Cạnh lên xung clock

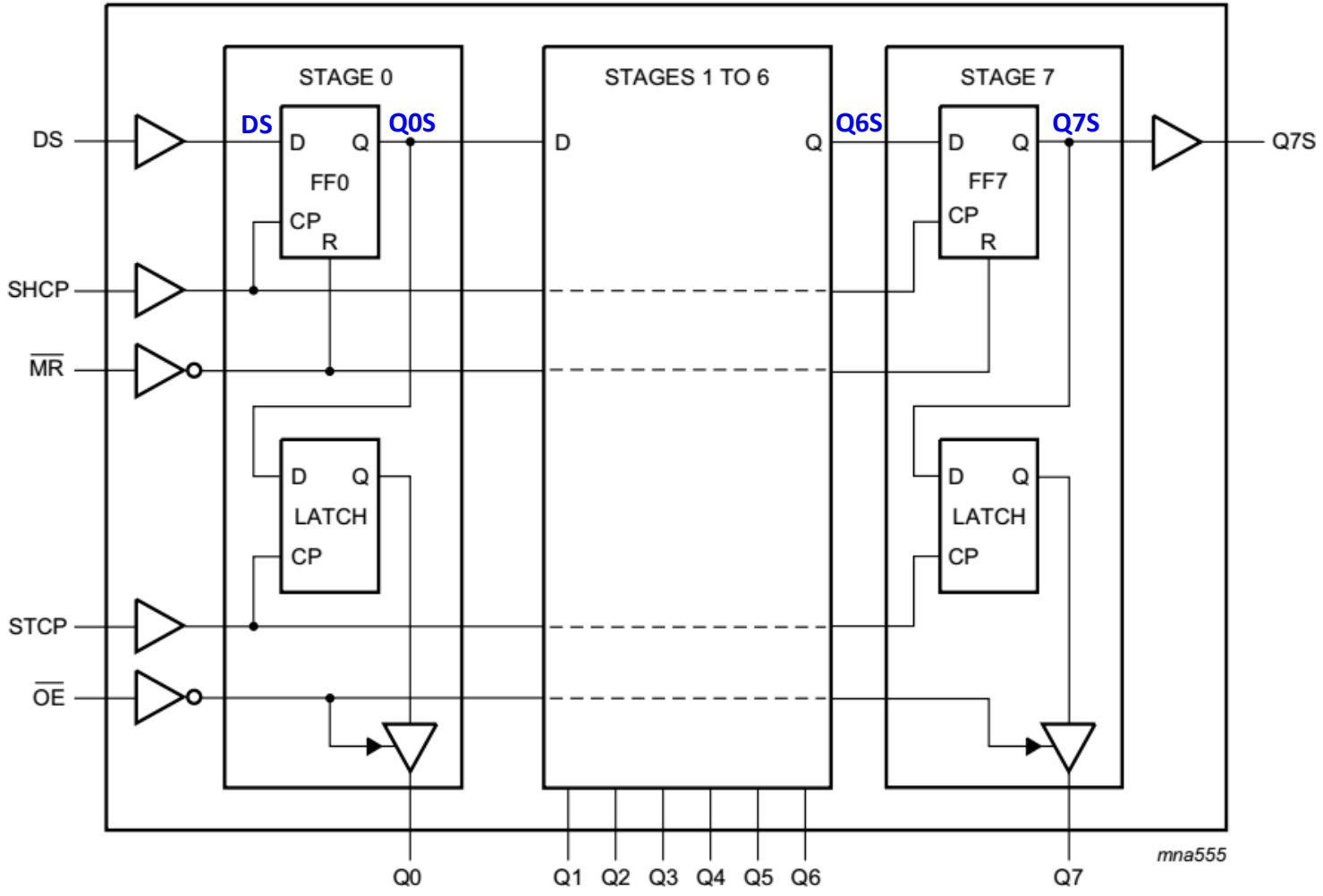
QnS = Ngõ ra thanh ghi dịch (Ngõ ra D_FFn)

Qn = Ngõ ra song song (Ngõ ra D_LATCHn)

X = tùy định

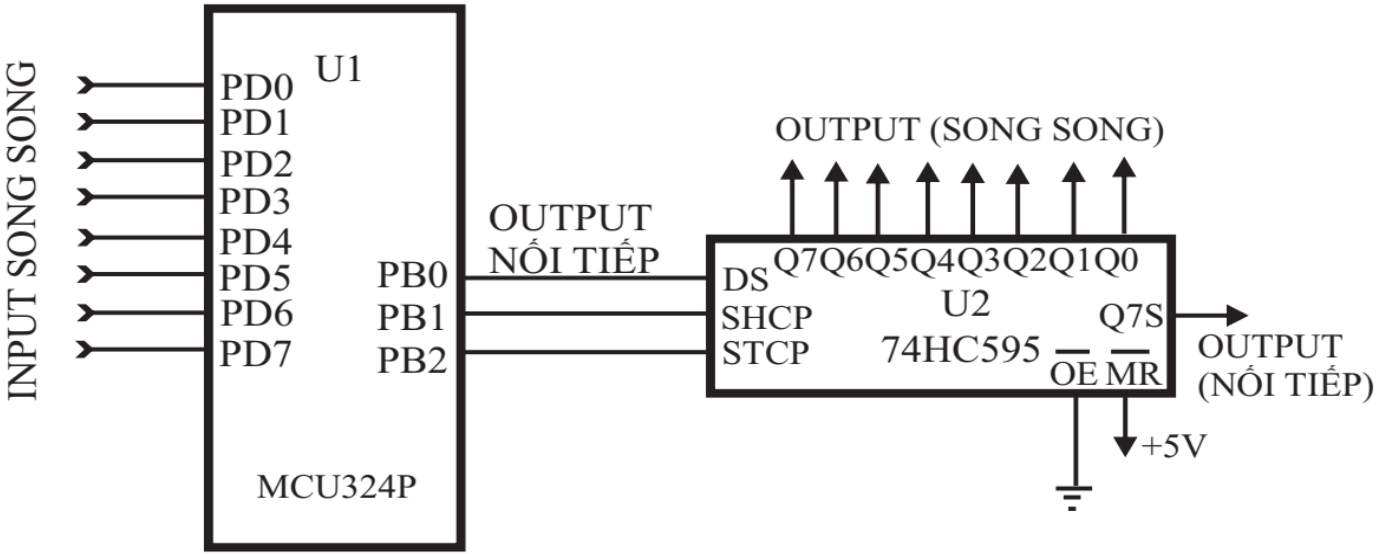
H = mức cao, L = mức thấp

6.6 Giao tiếp với thanh ghi dịch 74HC595



6.6 Giao tiếp với thanh ghi dịch 74HC595

Ví dụ 17: Cho sơ đồ kết nối giữa MCU324P và 74HC595 như bên dưới. Viết một chương trình nhập data từ PORTD, xuất ra dưới dạng nối tiếp giao tiếp với thanh ghi dịch IC 74HC595.



6.6 Giao tiếp với thanh ghi dịch 74HC595

1

```
        .ORG    0
        RJMP    MAIN
        .ORG    0X40
MAIN:   LDI     R16,HIGH(RAMEND);đưa stack lên vùng đ/c cao
        OUT     SPH,R16
        LDI     R16,LOW(RAMEND)
        OUT     SPL,R16
        LDI     R16,0X00;
        OUT     DDRD,R16        ;PortD input
        LDI     R16,0X07        ;PB0=DS,PB1=CK dịch,PB2=CK xuất output
        OUT     DDRB,R16
        CBI     PORTB,1         ;CK dịch (SHCP = PB1) =0
        CBI     PORTB,2         ;CK xuất (STCP = PB2) =0
START:  IN      R17,PIND        ;đọc data từ PORTD
        RCALL   SHIFT_OUT       ;gọi ctc ghi dịch
        RJMP    START
```

2

```
;----------------------------------
;SHIFT_OUT dịch data nối tiếp xuất ra port, MSB trước
;Input: R16 đếm số bit dịch,R17 chứa byte data cần dịch
;Output: data xuất ra PB0,R17 bảo toàn nội dung
;----------------------------------
SHIFT_OUT: LDI   R16,8      ;R16 đếm số bit dịch
SH_O:      ROL   R17        ;quay trái R17 qua C
           BRCC  BIT_0      ;C=0 xuất bit 0
           SBI   PORTB,0    ;C=1 xuất bit 1
           RJMP  SH_CK      ;nhảy đến xuất xung clock dịch (SHCP = PB1)
BIT_0:     CBI   PORTB,0    ;xuất bit 0
SH_CK:     SBI   PORTB,1    ;xuất xung clock dịch (SHCP = PB1)
           CBI   PORTB,1
           DEC   R16        ;đếm số lần dịch
           BRNE  SH_O       ;dịch tiếp cho đủ số bit
           SBI   PORTB,2    ;xuất xung data ngõ ra song song (STCP = PB2)
           CBI   PORTB,2    ;sau 8 xung clock dịch SHCP
           ROL   R17        ;dịch lần cuối (lần 9) phục hồi nội dung R17
           RET
```