

CHƯƠNG 6: GIAO TIẾP NGOẠI VI

6.1 GIỚI THIỆU CHUNG

Trong chương này sẽ khảo sát các phương pháp giao tiếp với các thiết bị ngoại vi cơ bản qua các port IO như nút nhấn(SW),bàn phím(keypad),mở rộng các ngõ vào/ra song song bằng các cổng đệm 3 trạng thái,mạch chốt,bộ nhớ ngoài,giải mã địa chỉ,thanh ghi dịch(shift register),các bộ hiển thị LED đơn,LED dãy(barLED),LED 7 đoạn(7 segment LED),LED ma trận(matrix LED),các bộ hiển thị LCD ký tự(alpha numeric LCD),LCD đồ họa(graphic LCD),mở rộng phần truy xuất EEPROM.

Chương này sẽ trình bày chi tiết thiết kế phần cứng,thiết kế phần mềm qua giải thuật chương trình và lập trình hợp ngữ,lập trình C với mỗi thiết bị ngoại vi.

Các ví dụ trong chương này được mặc định nguồn cấp cho AVR ATmega324P $V_{cc}=+5V$,tần số dao động chủ $F_{osc}=8Mhz$ theo danh định.Để đơn giản ta dùng ký hiệu MCU324P thay cho AVR ATmega324P.

Trước khi vào phần thiết kế chi tiết,ta cần phải hiểu rõ các thông số của AVR MCU324P và các thiết bị ngoại vi cần giao tiếp,thông qua data sheet(xem chi tiết ở phần phụ lục)

1. Các thông số cơ bản của AVR MCU324P

- Nguồn cấp điện V_{cc} : $V_{cc}=1.8 - 5.5V$
- Tần số dao động F_{osc} : $F_{osc}=0 - 20Mhz$ chọn $F_{osc}=8Mhz$
- Chu kỳ máy MC: $1MC=T_{osc} = 125ns$
- Điện áp ngõ vào mức 0 max V_{ILmax}
- Điện áp ngõ vào mức 1 min V_{IHmin}
- Điện áp ngõ ra mức 0 max V_{OLmax}
- Điện áp ngõ ra mức 1 min V_{OHmin}
- Dòng ngõ vào(rising) mức 0 I_{IL}
- Dòng ngõ vào(rising) mức 1 I_{IH}
- Dòng ngõ ra max $I_{OH/Lmax}$
- Tổng các dòng I_{OL}/I_{OH} max portB,D
- Tổng dòng I_{OL}/I_{OH} max portA,C

Chọn $V_{cc}=5V$

$V_{ILmax}=0.1V_{cc}$	$V_{ILmax}=0.5V$
$V_{IHmin}=0.7V_{cc}$	$V_{IHmin}=3.5V$
$V_{OLmax}=0.9V$	$I_{OL}=20mA$
$V_{OHmin}=4.2V$	$I_{OH}=-20mA$
$I_{IL}=-1\mu A(V_{IL})$	
$I_{IH}=1\mu A(V_{IH})$	
$I_{OH/Lmax}=-/+40mA$	
100mA	
100mA	

2. Các thông số DC(tĩnh) của thiết bị ngoại vi như $V_{ILmax},V_{IHmin},V_{OLmax},V_{OHmin},I_{IL},I_{IH},I_{OL},I_{OH}$

3. Các thông số AC(động) của thiết bị ngoại vi như tốc độ,thời gian đáp ứng,giản đồ định thi...

Ví dụ 6.1: Xét điều kiện giao tiếp trực tiếp IC chốt D 8 bit 74HC475 với MCU324P như hình 6.1a

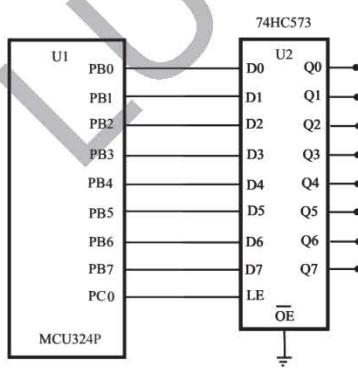
Giải:

Theo data sheet IC 74HC573,các thông số DC với $V_{cc}=5V$ như sau:

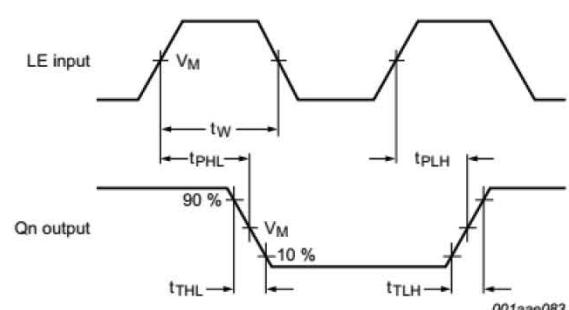
$$V_{ILmax}=0.3V_{cc}=1.5V, V_{IHmin}=0.7V_{cc}=3.5V, V_{OLmax}=0.4V, V_{OHmin}=4V, I_{Ilmax}=\pm 1\mu A$$

So sánh ngõ ra MCU324P với ngõ vào 74HC573 cho các thông số DC đều đạt yêu cầu:

	MCU324P	74HC573
Áp mức 0	$V_{OLmax}=0.9V$	$V_{ILmax}=1.5V$
Áp mức 1	$V_{OHmin}=4.2V$	$V_{IHmin}=3.5V$
Dòng mức 0	$I_{OL}=20mA$	$I_{IL}=-1\mu A$
Dòng mức 1	$I_{OH}=-20mA$	$I_{IH}=1\mu A$



(a)



(b)

Hình 6.1: (a) Sơ đồ giao tiếp 74HC573;(b) Giản đồ định thi 74HC573

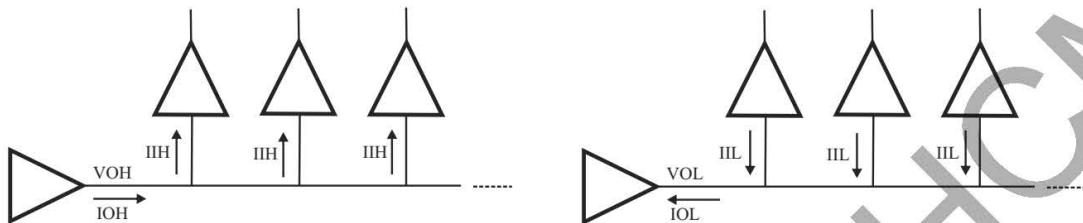
Các thông số AC 74HC573: $t_w=24\text{ns}$, $t_{pd}=t_{PHL}=t_{PLH}=45\text{ns}$

Trong chương trình ta tạo xung mở chốt trên LE:

SBI PORTC,0 ; 1MC
CBI PORTC,0 ; 1MC

Trên PC0=LE sẽ xuất hiện 1 xung mức 1 trong thời gian 1MC=125ns> t_w , t_{pd} . Do đó sau khi thực thi 2 lệnh trên ngõ ra Qn của 74HC573 sẽ hoàn toàn đáp ứng và xác lập giá trị mới.

4. Khả năng tỏa nhánh ngõ ra(fanout)



Hình 6.2: Khả năng tỏa nhánh ngõ ra

Xét ngõ ra một IC kết nối với n ngõ vào thiết bị ngoại vi
Ngõ ra mức 1 VOH:

$$\bullet I_{OH} = \sum I_{IH} \quad (6.1a)$$

Ngõ ra mức 0 VOL:

$$\bullet I_{OL} = \sum I_{IL} \quad (6.1b)$$

Số ngõ vào thiết bị ngoại vi tối đa n_{max} thỏa mãn (6.1a,b)

Ví dụ 6.2: Ngõ vào một thiết bị ngoại vi có thông số $IIL=400\mu\text{A}$, $IHH=40\mu\text{A}$. Tính số ngõ vào thiết bị ngoại vi tối đa có thể kết nối với 1 ngõ ra của MCU324P?

Giải:

MCU324P có $IOL/H=20\text{mA}$ (điều kiện test theo data sheet)

Số ngõ vào max(VOH)= $IOH/IHH=20/0.04=500$

Số ngõ vào max(VOL)= $IOL/IIL=20/0.4=50$

Vậy số ngõ vào tối đa của thiết bị ngoại vi kết nối với 1 ngõ ra MCU324P : $n_{max}=50$

➤ **Lưu ý:** Công thức tính khả năng tỏa nhánh ngõ ra chỉ xét về mặt DC (tĩnh). Trong thực tế do ảnh hưởng của các điện dung ngõ vào sẽ làm trễ đáp ứng chuyển biến mức logic, dẫn đến làm chậm đáp ứng thời gian. Do đó, ta phải giảm số ngõ vào kết nối so với tính toán!

❖ Câu hỏi ôn tập

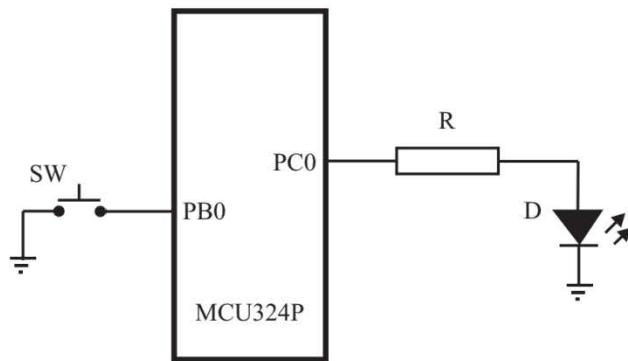
1. IC 74LS00 cấp nguồn 5V có các thông số : $VIL_{max}=0.8\text{V}$, $VIH_{min}=2\text{V}$, $VOL_{max}=0.4\text{V}$, $VOH_{min}=2.7\text{V}$. Xét giao tiếp IC 74LS00 trường hợp là ngõ vào và ngõ ra đối với MCU324P?
2. Một thiết bị ngoại vi có $IIL=-4\text{mA}$, $IHH=0.4\text{mA}$. Tính số thiết bị tối đa có thể ghép với MCU324P, cho dòng ngõ ra chân port MCU32 là 20mA.
3. Trong hình ví dụ 6.1b, giả sử $t_{wmin}=200\text{ns}$. Viết đoạn lệnh tạo xung LE phù hợp, cho $Fosc=8\text{Mhz}$.
4. Trong hình ví dụ 6.1b, giả sử $t_{pd}=t_{PHL}=t_{PLH}=300\text{ns}$. Viết đoạn lệnh tạo xung LE phù hợp, cho $Fosc=8\text{Mhz}$.
5. Một thiết bị ngoại vi hoạt động với $VIH_{min}=3\text{V}$, $IHH=60\text{mA}$. Có thể ghép trực tiếp ngõ vào thiết bị ngoại vi này với chân port MCU324P được không? Nếu không ghép trực tiếp được, đưa ra phương án ghép khác phù hợp để thiết bị ngoại vi hoạt động được.

6.2 Giao tiếp nút nhấn(SW), bàn phím(keypad)

Hình 6.3 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.

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 lập vòng lại
----- Thực thi đoạn chương trình khi SW nhấn



Hình 6.3: Giao tiếp MCU324P với SW

Để ngõ vào PB0 có mức logic rõ ràng, ta khai báo ngõ PB0 có điện trở kéo lên. Chương trình lập vòng xét trạng thái chân PB0 cho đến khi có SW nhấn PB0=0 sẽ thoát qua lệnh RJMP chuyển đến thực thi lệnh kế tiếp.

Do SW nhấn là tiếp điểm cơ khí nên sẽ có hiện tượng rung phím khi nhấn làm mạch nhận dạng hiệu nhầm nhấn/nhả SW nhiều lần! Do đó phải có mạch chống rung phím bằng phần cứng như mạch RS FF, mạch mono stable hay one-shot. Có thể thực hiện chống rung phím bằng phần mềm bằng cách gọi chương trình con delay vài chục ms khi 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 lập vòng lại
        RCALL    DELAY_10MS;delay 10ms
        SBIC     PINB,0   ;đọc lại trạng thái SW
        RJMP     WAIT     ;lập vòng lại nếu mức 1
;----- Thực thi đoạn chương trình khi SW nhấn
```

Ví dụ 6.3: Từ hình 6.3 viết một chương trình mỗi lần nhấn SW đèn LED D sẽ lờn lượt sáng /tối. Chương trình có chống rung SW.

Giải:

Trước tiên ta tính phân cực cho LED D. Với LED đơn đường kính 3mm, các giá trị danh định điện áp và dòng điện phân cực thuận là 2V và 10mA. LED sáng khi PC0=1 #VOH=5V, dòng IO danh định MCU324P 20mA nên đủ khả năng lái trực tiếp LED sáng với dòng 10mA.

$$R = (V_{OH} - V_D) / I_D \quad (6.2)$$

$$R = (5 - 2) / 10 = 300\Omega \rightarrow \text{Chọn } R = 330\Omega$$

❖ Chương trình hợp ngữ VD6.3:

```
.INCLUDE <M324PDEF.INC> ;file định nghĩa các ký hiệu sử dụng cho MCU324P
.ORG 0 ;địa chỉ bắt đầu chương trình sau khi reset
        RJMP MAIN ;nhảy đến 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 output
        CBI PORTC,0 ;PC0=0 LED tối
.WAIT: SBIC PINB,0 ;bỏ qua lệnh kế tiếp nếu SW nhấn PB0=0
        RJMP WAIT ;SW không nhấn PB0=1 lập vòng lại
        RCALL DELAY_10MS;delay 10ms
        SBIC PINB,0 ;đọc lại trạng thái SW
```

```

RJMP  WAIT      ;lập vòng lại nếu mức 1
LDI   R17,0X01  ;R17=01H
IN    R18,PORTC;đọc PORTC
EOR   R18,R17   ;đảo bit PC0
OUT   PORTC,R18;xuất ra PORTC
RJMP  WAIT      ;lập vòng lại từ đầu

```

;----- DELAY_10MS

DELAY_10MS:

```

LDI   R21,80   ;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

```

- Lệnh đầu tiên là nhảy đến địa chỉ ký hiệu nhãn MAIN tại 0x40 để dự trù vùng địa chỉ thấp cho các vector ngắn(sẽ trình bày chi tiết trong chương 10)

- Đầu chương trình phải đưa vùng stack lên vùng SRAM địa chỉ cao nhất RAMEND=0x08FF (RAMEND đã được định nghĩa trong file M324PDEF.INC)

➤ *Với MCU324P khi mới reset hệ thống, thanh ghi SP đã trở vào địa chỉ định=0x08FF, nên các lệnh nạp địa chỉ định cho SP là không cần thiết. Tuy nhiên để mang tính tổng quát, ta vẫn ghi các lệnh khởi động vùng stack, để khi áp dụng cho các MCU khác trong họ AVR có reset khởi động vùng stack không phải ở định sẵn đúng!*

- Khai báo PB0 là input có điện trở kéo lên, PC0 là output, khởi động PC0=0 LED tối
- Lập vòng kiểm tra SW nhấn PB0=0, khi có SW nhấn gọi chương trình con delay 10ms chống rung SW, sau đó kiểm tra lại SW nếu còn nhấn sẽ lặp lượt kích LED sáng/tối
- LED sáng/tối PC0=1/0 bằng cách đọc trạng thái PC0 và EXOR với bit 1
- Chương trình con DELAY_10MS tạo trễ 10ms bằng 2 vòng lặp L2 250 lần và L1 80 lần. Có thể tính gần đúng thời gian delay như sau:

Vòng L2: $(1+1+2) \times 250 = 1000\text{MC}$

Vòng L1: 80

Tổng thời gian delay = $1000 \times 80 \times 0.125\mu\text{s} = 10000\mu\text{s} = 10\text{ms}$

➤ Lưu ý: Thời gian delay có thể thay đổi tùy loại SW để chống rung hiệu quả nhất!

❖ Chương trình C VD6.3:

```

#include<avr/io.h>
void delay_ms(unsigned int n) // hàm(chương trình con) delay ms
{
  unsigned int i;
  for(i=0;i<=n;i++); // Td=6xn MC
}
int main()
{
  DDRB=0xfe ; //PB0 input
  PORTB=0x01 ; //R kéo lên PB0
  DDRC=0xff ; //PC0 output
  PORTC=0xfe ; //PC0=0
  while(1) //lập vòng vô hạn
  {
    loop: while(PINB&0x01); //chờ PB0=0 SW nhấn
    delay_ms(13333); //delay 10ms
    if((PINB&0x01)==1) //PB0=1 SW chưa nhấn
      goto loop ; //lập vòng chờ SW nhấn
  }
}

```

```

        else
            PORTC^=0x01;      //SW nhấn đảo bit PC0
    }
}

```

- Khi biên dịch chương trình con delay_ms trên sang hợp ngữ, tính gần đúng thời gian lặp vòng:
 $T_d = 6MC_{xn}$ (6.4)

Để tạo thời gian delay 10ms :

$$n = (T_d / 6MC) = 10000 / (6 \times 0.125) = 13333$$

➤ Người đọc tự tham khảo trình biên dịch sang hợp ngữ từ C trong file .lss(listing) suy ra (6.4)

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 lập vòng đọc lại từ đầu.

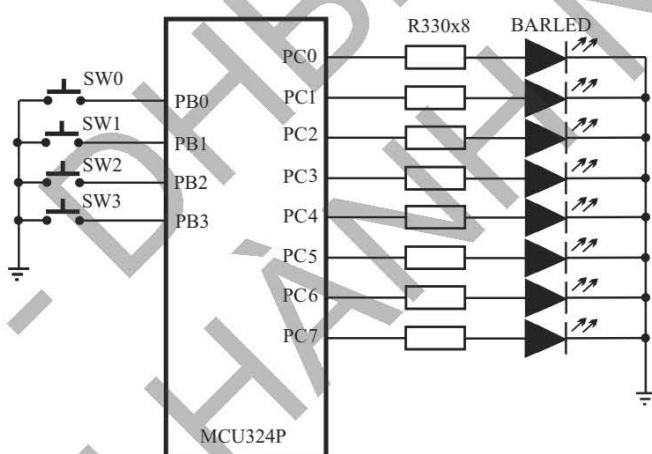
Ví dụ 6.4: Viết chương trình đọc trạng thái các SW và hiển thị dãy LED(bar LED) nếu có 1 SW nhấn/nhả như sau(có chống rung SW):

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

Giải:

Dãy LED gồm nhiều LED đơn ghép thẳng hàng chung trong một chip, phân cực từng LED như LED đơn. Trong hình 6.4 dãy LED gồm 8 LED, trong trường hợp cả 8 LED đều sáng, dòng cung cấp từ PORTC bằng $10 \times 8 = 80mA$ nhỏ hơn tổng dòng IO PORTC max = $100mA$, nên có thể ghép trực tiếp PORTC với dãy LED.

Chương trình sẽ kiểm tra SW nhấn/nhả 50 lần để xác định trạng thái SW, trả về mã SW nhấn tương ứng, và sẽ hiển thị dãy LED sáng theo mode mã SW.



Hình 6.4: Mạch điện ví dụ 6.4

❖ Chương trình hợp ngữ VD6.4:

```

INCLUDE <M324PDEF.INC>
EQU OUTPORT=PORTC
.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 R16,0XF0
      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 output
        LDI    R16,0X00
        OUT    PORTC,R16
WAIT_0:   LDI R16,50
BACK1:   RCALL GET_KEY
        BRCC  WAIT_0
        DEC   R16
        BRNE  BACK1
        PUSH  R17
WAIT_1:   LDI R16,50
BACK2:   RCALL GET_KEY
        BRCS  WAIT_1
        DEC   R16
        BRNE  BACK2
        POP   R17
        CPI   R17,0
        BREQ MD_0
        CPI   R17,1
        BREQ MD_1
        CPI   R17,2
        BREQ MD_2
        CPI   R17,3
        BREQ MD_3
        RJMP  WAIT_0
MD_0:    LDI R18,0X00      ;hiển thị mode 0
        OUT  OUTPORT,R18     ;dãy LED tối
        RJMP
        WAIT_0
MD_1:    LDI R18,0X0F      ;hiển thị mode 1
        OUT  OUTPORT,R18     ;sáng 4 LED trọng số thấp
        RJMP
        WAIT_0
MD_2:    LDI R18,0XF0      ;hiển thị mode 2
        OUT  OUTPORT,R18     ;sáng 4 LED trọng số cao
        RJMP
        WAIT_0
MD_3:    LDI R18,0XFF      ;hiển thị mode 3
        OUT  OUTPORT,R18     ;sáng cả dãy LED
        RJMP  WAIT_0
;
;
```

;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     ;cắt 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
        RJMP EXIT
CHK_KEY: ROR  R19
        BRCC KEY_CODE    ;quay phải qua C tìm vị trí SW nhấn
        DEC   R20
        BRNE CHK_KEY      ;C=0 có SW nhấn
                           ;SW không nhấn giảm vị trí SW
                           ;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	; C=1 có SW nhấn
EXIT:	RET	

❖ **Chương trình C VD6.4:**

```
#include <avr/io.h>
#define import PINB      // định nghĩa PORTB=input
#define outport PORTC    // định nghĩa PORTC=output
unsigned char key_flg ;// cờ báo có SW nhấn=1
unsigned char key_code,tam,i ;
void GET_KEY();        // khai báo hàm nhận dạng SW nhấn
int main()             // bắt đầu chương trình
{
    DDRB=0xf0 ;//khai báo PB0-PB3 là ngõ vào
    PORTB=0x0f ;//khai báo điện trở kéo lên ngõ PB0-PB3
    DDRC=0xff ;//khai báo PC là output
    outport=0x00 ;//PC=0 LED tối
    while(1)      // lập vòng vô hạn
    {
        for ( i=50;i>=1;i--) // đọc trạng thái SW nhấn 50 lần
        {
            GET_KEY(); // đọc trạng thái SW
            if(key_flg==0) //cờ báo=0 SW chưa nhấn
                i=50 ;// đọc lại từ đầu
        }
        tam = key_code ;//gán mã SW vào tam
        for ( i=50;i>=1;i--) // đọc trạng thái SW nhả 50 lần
        {
            GET_KEY(); // đọc trạng thái SW
            if (key_flg==1) //cờ báo=1 SW chưa nhả
                i=50 ;// đọc lại từ đầu
        }
        key_code =tam ;//lấy lại mã SW
        switch(key_code)
        {
            case(0): //mã SW=0
            {
                outport=0x00 ;//tối dãy LED
                break;
            }
            case(1): //mã SW=1
            {
                outport=0x0f ;//sáng 4 LED trọng số thấp
                break;
            }
            case(2): //mã SW=2
            {
                outport=0xf0 ;//sáng 4 LED trọng số cao
                break;
            }
            case(3): //mã SW=3
            {
                outport=0xff; //sáng cả dãy LED
            }
        }
    }
}
```

```
        break;
    }
}
void GET_KEY()           // định nghĩa hàm nhận dạng trạng thái SW
{
    unsigned char col,buf,j;// khai báo các biến sử dụng trong hàm
    col = 0x0e             // mã quét vị trí SW 0
    key_code=0x04           // số SW
    j=0x04;                // đếm số vị trí SW
    buf=import & 0x0f       // đọc trạng thái SW, che các bit thấp
    if( buf == 0x0f )        // buf=0fh không có SW nhận
        key_flg = 0          // xóa cờ báo
    else
    {
        key_flg=0;           ;
        while ( (key_flg == 0) &&( j != 0 ) ) // lập vòng khi cờ báo=0 và quét chưa hết 4 SW
        {
            if(buf == col)      // đúng vị trí SW đang quét
            {
                key_code=key_code - j // tính mã SW
                key_flg = 1          // đặt cờ báo=1
            }
            else
            {
                j--                // vị trí SW kế tiếp
                col=col<<1         // quét vị trí SW kế tiếp
                col++                // đặt lại LSB=1
                col&=0x0f            // mã quét vị trí SW kế tiếp
            }
        }
    }
}
```

- Trình biên dịch C AVR không có định nghĩa BIT nên trong hàm GET_KEY() xử lý như sau:
 1. Thay cờ C bằng biến key_flg(byte)
 2. Nhận dạng vị trí SW nhận bằng cách so sánh biến buf(trạng thái đọc từ import)với biến col (mã vị trí SW)lần lượt là 00001110,00001101,00001011 và 00000111.Mỗi lần dịch trái col, LSB=0 nên phải tăng col thêm 1 và xóa 4 bit cao.

Hình 6.5 minh họa MCU324P giao tiếp bàn phím(keypad)16 phím, chia thành 4 hàng và 4 cột lần lượt kết nối với 4 bit cao và 4 bit thấp của PORTB.Ta xem ví dụ 6.5.

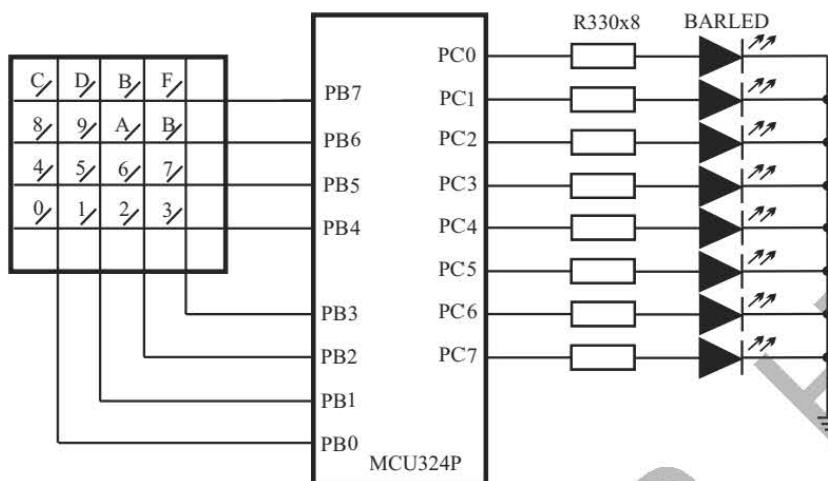
Ví dụ 6.5: Từ sơ đồ hình 6.5, viết một chương trình nhận dạng khi nhấn một phím sẽ hiển thị mã ASCII tương ứng với mã phím.chống rung phím khi nhấn và nhả.

Giải:

- ❖ Ý tưởng giải thuật nhận dạng phím nhấn
 - 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 khi xuất mã quét cột 1110 ,PB0=0,nhấn phím 4 làm PB5=0,đọc về trạng thái hàng 1101, tương ứng vị trí cột 0 vị trí hàng 4.Suy ra mã phím=0+4=4

- Áp dụng ví dụ 6.4 chống rung phím khi nhấn/nhả,nhận dạng phím nhấn



Hình 6.5: Giao tiếp bàn phím 16 phím

❖ Chương trình hợp ngữ VD6.5:

```

INCLUDE <M324PDEF.INC>
.EQU OUTPORT=PORTC
.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 R16,0X0F
      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
      RJMP KEY_RD ;ctc đọc phím
START: RCALL HEX_ASC ;chuyển từ mã Hex sang mã ASCII
      OUT OUTPORT,R18 ;hiển thị bar LED
      RJMP START ;lặp vòng từ đầu
;-----;
;HEX_ASC chuyển từ mã Hex sang mã ASCII
;Input R17=mã Hex,Output R18=mã ASCII
;-----;
HEX_ASC: CPI R17,0X0A ;ký tự <$0A?
      BRCS NUM ;C=1 nhỏ hơn
      LDI R18,0X37 ;ký tự chữ
      RJMP CHAR ;
NUM:  LDI R18,0X30 ;ký tự số
CHAR: ADD R18,R17 ;cộng thêm $30
      RET
;-----;

```

```

;KEY_RD đọc trạng thái phím
;Chống rung phím khi nhấn/nhả 50 lần
;Sử dụng GET_KEY16 nhận dạng phím 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 lập lại
        DEC   R16       ;đếm số lần nhận dạng phím
        BRNE BACK1     ;lập vòng cho đủ số lần đếm
        PUSH  R17       ;xác nhận phím nhấn,cắt 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     ;lập 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:
        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
        RJMP EXIT      ;phím chưa nhấn,C=0
        ;thoát
CHK_KEY: SUBI R17,4      ;tính vị trí cột
        NEG   R17       ;bù 2 lấy số dương
        SWAP R19       ;đảo sang 4 bit thấp mã hàng
        LDI  R20,4      ;quét 4 hàng
SCAN_ROW:
        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
        DEC   R20
        BRNE SCAN_ROW;quét hết 4 hàng
        CLC
        RJMP EXIT      ;không có phím nhấn C=0
        ;thoát
SET_FLG: SEC               ;có phím nhấn C=1

```

EXIT: RET

- Ta gom đoạn chương trình nhận dạng phím nhấn và chống rung phím khi nhấn/nhả thành chương trình con KEY_RD.Lưu ý là chỉ thoát khỏi chương trình con này khi có phím nhấn!
- Trong chương trình con GET_KEY16 sau khi xuất mã quét cột, đọc lại trạng thái hàng 2 lần để chờ Port I/O đáp ứng ổn định!

❖ **Chương trình C VD6.5:**

```
#include <avr/io.h>
#define import PINB           //định nghĩa import=PINB
#define outport PORTC         // định nghĩa outport=PORTC
unsigned char key_flg ;      // cờ báo có phím nhấn=1
unsigned char key_code,asc;   // mã phím(hex),mã ASCII của phím
void KEY_RD() ;//khai báo hàm đọc phím
void GET_KEY16() ;// khai báo hàm nhận dạng phím nhấn
unsigned char HEX_ASC(unsigned char x); //khai báo hàm chuyển mã Hex sang mã ASCII
int main() // bắt đầu chương trình
{
    DDRB=0x0f ;//PB0-PB3 output,PB4-PB7 input
    PORTB=0xff ;//điện trở kéo lên PORTB
    DDRC=0xff ;//PORTC output
    outport=0x00 ;//tối đa LED
    while(1) // lập vòng vô hạn
    {
        KEY_RD() ;// gọi hàm đọc phím
        outport=HEX_ASC(asc) ;//gọi hàm chuyển mã Hex sang mã ASCII
    }
}
void KEY_RD() //định nghĩa hàm đọc phím
{
    unsigned char tam,i ; //khai báo các biến sử dụng trong hàm
    for ( i=50;i>=1;i--) // đọc trạng thái phím nhấn 50 lần
    {
        GET_KEY16() ;// đọc trạng thái phím
        if(key_flg==0) // cờ báo=0 không có phím nhấn
            i=50 ;// đọc lại từ đầu
    }
    tam = key_code ;//gán mã phím vào tam
    for ( i=50;i>=1;i--) // đọc trạng thái phím nhả 50 lần
    {
        GET_KEY16() ;// đọc trạng thái phím
        if(key_flg==1) // cờ báo=1 phím chưa nhả
            i=50 ;// đọc lại từ đầu
    }
    key_code =tam ;// lấy lại mã phím
}
void GET_KEY16() // định nghĩa hàm nhận dạng trạng thái phím
{
    unsigned char col,row,buf,j;// khai báo các biến sử dụng trong hàm
    col = 0xfe ;// mã quét cột 0
    j=0x04 ;// đếm 4 cột
    key_flg = 0 ;// xóa cờ báo
    while ( (key_flg == 0) &&(j != 0) ) //lập vòng khi cờ báo=0 và quét chưa hết 4 cột
    {
        PORTB = col ;// xuất mã quét cột ra outport
```

```

buf = col          ;// cất mã quét cột
row = import & 0xf0 ;// đọc hàng, che các bit hàng lấy mã hàng
row = import & 0xf0 ;// đọc lại hàng, che các bit hàng lấy mã hàng
if ( row == 0xf0 ) // row=f0h không có phím nhấn
{
    col = buf << 1 ;// dịch trái tạo mã quét cột kế tiếp
    col++
    j--               ;// đếm số cột đã quét
}
else               // col khác f0h có phím nhấn
{
    key_flg = 0     ;// xóa cờ báo phím nhấn
    j = 4-j         ;// tính vị trí cột có phím nhấn
    row = row>>4 & 0x0f ;// chuyển mã hàng sang 4 bit thấp
    col=0x0e        ;// mã hàng đầu tiên
    buf = 4          ;// đếm dịch hàng
    while ( (buf != 0) &&(key_flg == 0) ) // lập vòng khi chưa dịch hết 4 hàng
    {
        if ( row!=col )      // không đúng mã hàng có phím nhấn
        {
            buf--           ;// đếm số lần dịch mã hàng
            col=col<<1       ;// dịch sang mã hàng kế tiếp
            col++             ;// đặt LSB=1
            col&=0x0f          ;// xóa 4 bit cao
            j++                ;// tăng vị trí hàng thêm 4
            j++                ;
            j++                ;
            j++                ;
        }
        else               // đúng là hàng có phím nhấn
        {
            key_code = j ;// mã phím=j
            key_flg=1       ;// đặt cờ báo có phím nhấn
        }
    }
}
unsigned char HEX_ASC(unsigned char x) //hàm chuyển mã Hex sang mã ASCII
{
    if(key_code<0x0a)
        x=key_code+0x30 ;// ký tự số
    else
        x=key_code+0x37 ;// ký tự chữ
    return(x);
}

```

- Cũng giống như ví dụ 6.4, biến key_flg(byte) thay cho cờ C báo có phím nhấn
- Lưu ý về các điều kiện thoát khỏi vòng lặp while!



Câu hỏi ôn tập

1. Viết một đoạn lệnh hợp ngữ khai báo chân PD4 là ngõ vào.
2. Viết một đoạn lệnh hợp ngữ khai báo 4 bit cao/4 bit thấp PORTA lần lượt là ngõ vào/ngõ ra
3. Viết một đoạn lệnh khai báo PORTD là ngõ ra

- Vẽ sơ đồ kết nối 1 SW với PB0 sao cho SW nhấn=mức 1,SW nhả=mức 0 và kết nối LED đơn với PB1 sao cho LED sáng khi PB1=0.
- Từ sơ đồ câu 4 viết một chương trình hợp ngữ và C thực hiện liên tục mỗi lần SW nhấn/nhả(có chép rung SW)LED lờn lượt sáng tối.

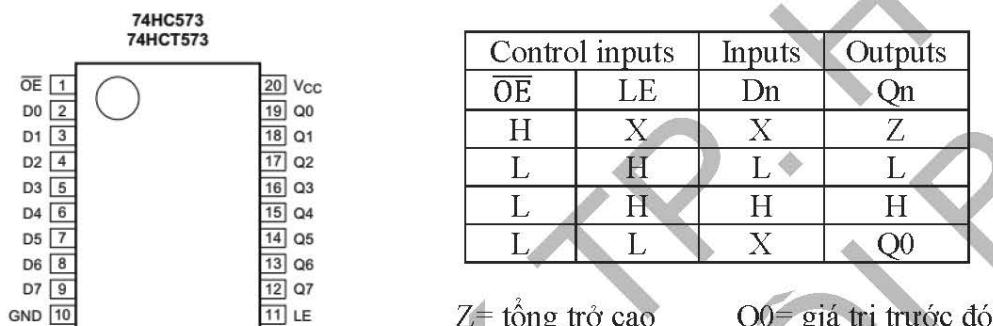
6.3 Mở rộng giao tiếp cổng ngoại vi

MCU324P có 4 PORTA,B,C,D mỗi PORT có 8 bit,tổng cộng có 32 chân giao tiếp ngoại vi.

Trong trường hợp số cổng ngoại vi cần giao tiếp nhiều hơn 32 ngõ hoặc phải sử dụng các chân ngoại vi của MCU324P cho các chức năng khác như ADC,giao tiếp cổng nối tiếp,xung CK...,ta phải thực hiện mở rộng cổng ngoại vi cho MCU324P.

Phần sau đây sẽ trình bày các phương pháp mở rộng cổng ngoại vi giao tiếp song song và nối tiếp như giao tiếp các IC chốt D,cổng đếm 3 trạng thái,bộ nhớ SRAM,giải mã địa chỉ,thanh ghi dịch...

6.3.1 Giao tiếp IC chốt D 8 bit 74HC573,IC đếm 3 trạng thái 74HC244



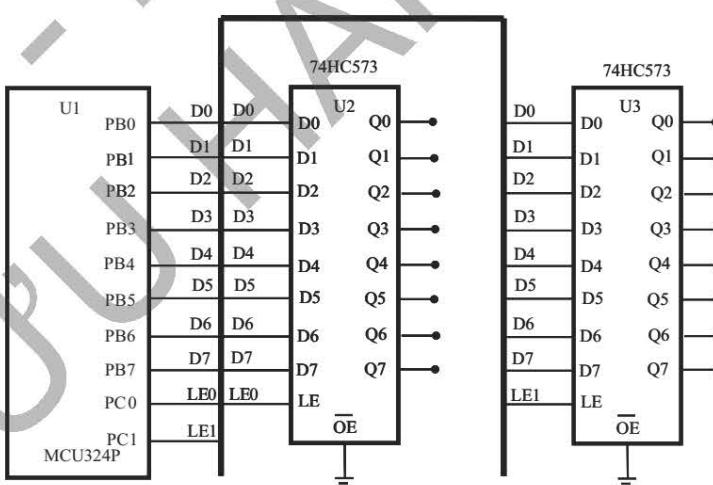
Hình 6.6: Sơ đồ chân IC 74HC573 và bảng sự thật

Từ bảng sự thật IC 74HC573,tóm tắt hoạt động như sau:

- OE điều khiển cấm/cho phép ngõ ra: OE=1 Qn=tổng trờ cao,OE=0 cho phép xuất ngõ ra
- LE điều khiển mở chốt với OE=0: LE=1 Qn=Dn mở chốt, LE=0 Qn=Q0 khóa chốt

Hình 6.7 là sơ đồ mở rộng 1 port thành 2 port ngoại vi 16 bit.Thay vì kết nối trực tiếp 2 port MCU324P,ta sử dụng 2 IC 74HC573 làm mạch phân đường(DEMUX),chỉ cần PORTB 8 bit data và 2 ngõ điều khiển PC0,PC1 lờn lượt tạo tín hiệu mở chốt chọn xuất data ra U2 hoặc U3.

Các chỉ tiêu kết nối phần cứng IC74HC573 với MCU324P đều phù hợp theo ví dụ 6.1

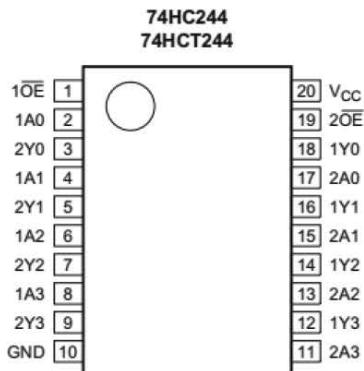


Hình 6.7: Mở rộng giao tiếp ngoại vi xuất sử dụng IC 74HC573

Để xuất data ra U2 hoặc U3 chỉ cần viết đoạn lệnh:

LDI	R16,0XFF	
OUT	DDRB,R16	;PORTB output
SBI	DDRC,0	;PC0 output
SBI	DDRC,1	;PC1 output
CBI	PORTC,0	;PC0=0 khóa chốt U2

CBI	PORTC,1	;PC1=0 khóa chốt U3
...		
OUT	PORTB,R17	;R17 chứa data xuất ra U2
SBI	PORTC,0	;PC0=1 mở chốt U2 xuất data ra U2
CBI	PORTC,0	;PC0=0 khóa chốt U2
OUT	PORTB,R18	;R18 chứa data xuất ra U3
SBI	PORTC,1	;PC1=1 mở chốt U3 xuất data ra U3
CBI	PORTC,1	;PC1=0 khóa chốt U3
...		



Inputs(i=1,2;n=0..3)		Outputs
iOE	iAn	iYn
H	X	Z
L	L	L
L	H	H

Thời gian truyền đạt An → Yn $t_{pd}=33\text{ns}$
Thời gian cho phép OE → Yn $t_{en}=45\text{ns}$

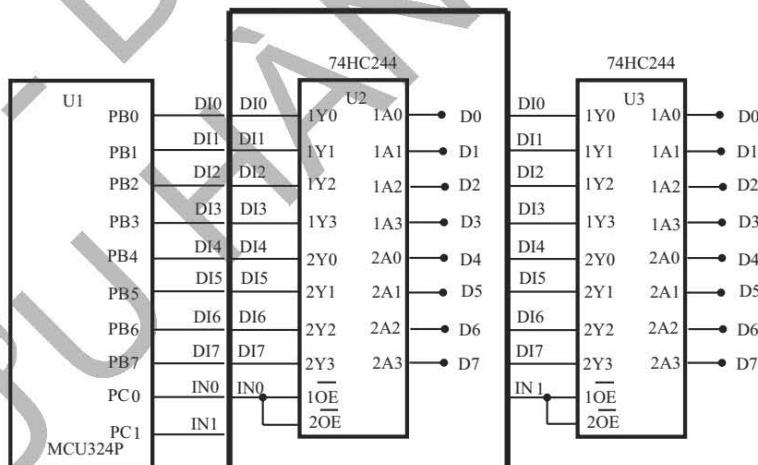
Hình 6.8: Sơ đồ chân IC 74HC244 và bảng sự thật

Từ bảng sự thật IC 74HC244, tóm tắt hoạt động như sau:

- OE cấm/cho phép xuất $\overline{OE}=0$ cấm $\overline{OE}=1$ cho phép xuất $Y_n=A_n$
- Tham khảo các thông số tĩnh và động của IC 74HC244 (xem data sheet ở phần phụ lục) cho phép kết nối trực tiếp các chân IC với các chân PORT MCU324P.

Hình 6.9 là sơ đồ mở rộng port ngõ vào 16 bit dùng 2 IC 74HC244 làm việc như bộ dồn kênh (MUX), điều khiển chọn ngõ vào U2 hoặc U3 bằng tín hiệu IN0 hoặc IN1 tích cực mức thấp.

➤ Lưu ý: Do các ngõ ra của các IC 74HC244 nối chung với nhau theo dạng wired-AND nên bắt buộc phải có ngõ điều khiển 3 trạng thái ngõ ra!



Hình 6.9: Mở rộng giao tiếp ngoại vi nhập sử dụng IC 74HC244

Để nhập data từ U2 hoặc U3 chỉ cần viết đoạn lệnh:

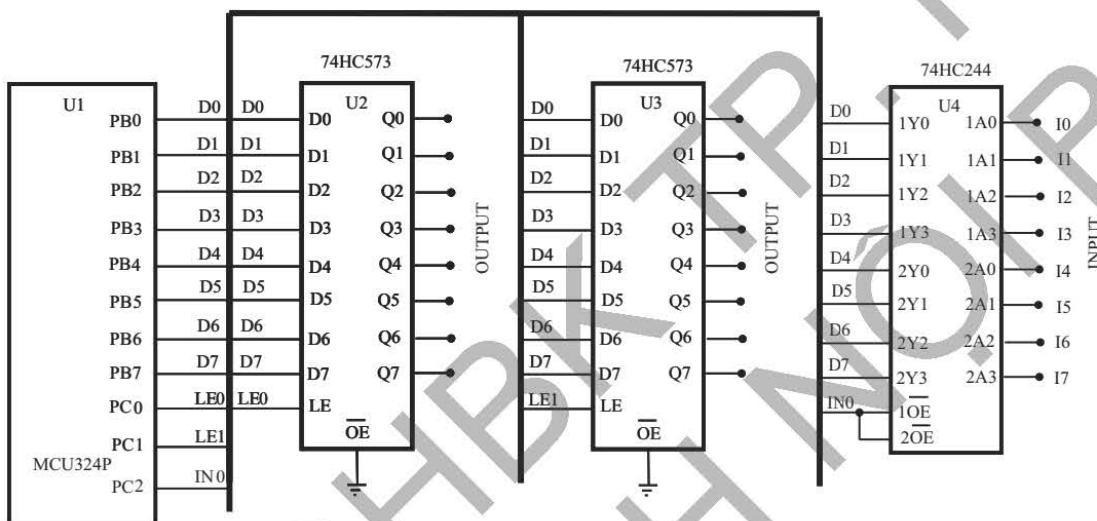
LDI	R16,0X00	
OUT	DDRB,R16	;PORTB input
SBI	DDRC,0	;PC0 output
SBI	DDRC,1	;PC1 output
SBI	PORTC,0	;PC0=1 khóa đếm U2
SBI	PORTC,1	;PC1=1 khóa đếm U3
...		

CBI	PORTC,0	;PC0=0 mở đệm U2 cho phép xuất data
IN	R17,PINB	;R17 chứa data nhập từ U2
SBI	PORTC,0	;PC0=1 khóa đệm U2
CBI	PORTC,1	;PC1=0 mở đệm U3 cho phép xuất data
IN	R18,PINB	;R18 chứa data nhập từ U3
SBI	PORTC,1	;PC1=1 khóa đệm U3

...

Từ hình 6.7 và 6.9 có thể mở rộng thêm cổng ngoại vi, chỉ cần thêm ngõ điều khiển. Ở phần sau sẽ trình bày mạch giải mã địa chỉ trong trường hợp cần mở rộng giao tiếp nhiều ngoại vi, tiết kiệm thêm các ngõ điều khiển mở cổng.

Ví dụ 6.6: Sơ đồ hình 6.10 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.



Hình 6.10: Sơ đồ ví dụ 6.6

❖ Chương trình hợp ngữ VD6.6:

```

INCLUDE <M324PDEF.INC>
.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
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
;
```

❖ Chương trình C VD6.6:

```

#include <avr/io.h>
#define import PINB //định nghĩa import=PINB
#define ioset DDRB // định nghĩa ioset=DDRB
#define outport PORTB //định nghĩa outport=PORTB
#define cont PORTC //định nghĩa cont=PORTC
const char dis_io=0xf4 ;//mã khóa các ngoại vi
const char en_u2=0xf5 ;//mã mở U2
const char en_u3=0xf6 ;//mã mở U3
const char en_u4=0xf0 ;//mã mở U4
unsigned char HEX_ASC1(unsigned char x) ;//khai báo hàm HEX_ASC
unsigned char tam,asc;
int main()
{
    DDRC=0x07 ;//định nghĩa PC0,PC1,PC2 là output
    cont=dis_io ;//khóa các ngoại vi
    while(1)
    {
        ioset=0x00 ;//khai báo port input
        cont=en_u4 ;//mở U4
        tam=import ;//nhập data từ U4
        cont=dis_io ;//khóa U4
        asc=tam&0x0f;//che 4 bit cao
        ioset=0xff ;//khai báo port output
        outport=HEX_ASC1(asc); //chuyển sang mã ASCII và xuất ra U2
    }
}
;
```

```

cont=en_u2 ;//mở U2
cont=dis_io ;//khóa U2
asc=tam>>4 ;//dịch data sang 4 bit thấp
outport=HEX_ASC1(asc); //chuyển sang mã ASCII và xuất ra U3
cont=en_u3 ;//mở U3
cont=dis_io ;//khóa U3
}
}

unsigned char HEX_ASC1(unsigned char x) //hàm chuyển mã Hex sang mã ASCII
{
    if(x<0x0a)
        x=x+0x30 ;//ký tự số
    else
        x=x+0x37 ;//ký tự chữ
    return(x);
}

```

- Do trình biên dịch C không có khai báo bit nên ta khai báo các byte hằng số để đặt/xóa các bit điều khiển ngoại vi tương ứng!

❖ Câu hỏi ôn tập

- Trong hình 6.10 ví dụ 6.6, giải thích tại sao phải kết nối \overline{OE} của U2, U3 xuống GND và \overline{OE} của U4 đến chân điều khiển từ MCU324P.
- Trong ví dụ 6.6, để cho phép xuất data ra U2 đoạn lệnh sau có đúng không và giải thích kết quả:


```

SBI    PORTC,0
OUT    OUTPORT,R17
CBI    PORTC,0
      
```
- Trong ví dụ 6.6, để nhập data từ U4, đoạn lệnh sau có đúng không và giải thích kết quả:


```

IN     R17,IMPORT
CBI    PORTC,2
SBI    PORTC,2
      
```
- Giả sử thời gian cho phép xuất của U4 $t_{OE}=400\text{ns}$, viết lại đoạn lệnh dọa data từ U4 theo ví dụ 6.6 phù hợp, cho $Fosc=8\text{Mhz}$.
- Vẽ sơ đồ giao tiếp MCU324P với IC 74HC245. Viết 1 đoạn chương trình đọc data sau đó xuất data đều giao tiếp qua IC này

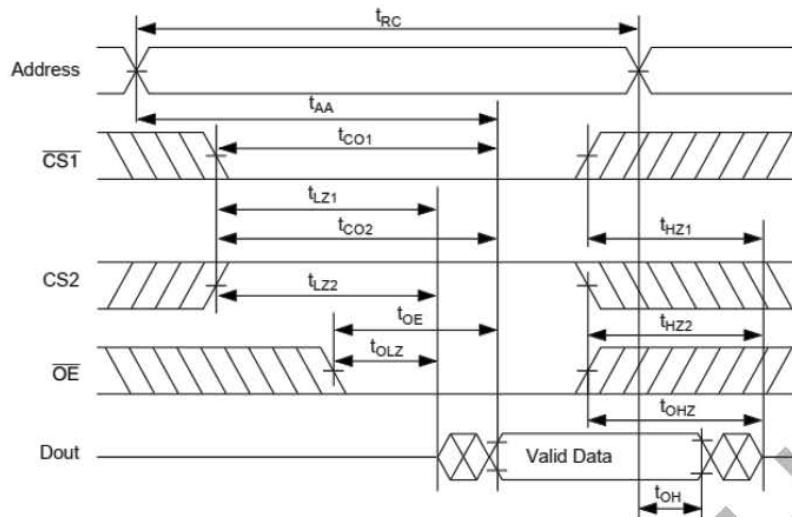
6.3.2 Giao tiếp bộ nhớ SRAM song song

MCU324P có 2KB SRAM và 1KB EEPROM onchip để lưu trữ data. Trong các ứng dụng cần dung lượng bộ nhớ lớn, MCU324P có thể giao tiếp với bộ nhớ ngoài để tăng thêm dung lượng lưu trữ data. Các bộ nhớ ngoài dung lượng lớn hàng MB trở lên hiện nay thường có dạng giao thức nối tiếp SPI hay I2C (thẻ nhớ, bộ nhớ EEPROM nối tiếp...) có thể kết nối trực tiếp với các cổng nối tiếp phù hợp của MCU324P (xem chương 8 cổng nối tiếp). Trong phần sau sẽ trình bày phương pháp giao tiếp với bộ nhớ SRAM song song tiêu biểu như IC SRAM HM6264 8KB.

Tham khảo data sheet HM6264 cho điện áp các mức logic và khả năng tòa nhánh đều phù hợp kết nối trực tiếp được với MCU324P. Ta xem đặc tính AC qua các giản đồ xung định thì đọc và ghi bộ nhớ.

NC	1	28	Vcc	WE	CST	CS2	\overline{OE}	Mode	I/O pin	V _{cc} current	Note
A12	2	27	\overline{WE}	x	H	x	x	Not selected (power down)	High Z	I_{SB}, I_{SB1}	
A7	3	26	CS2	x	x	L	x		High Z	I_{SB}, I_{SB1}	
A6	4	25	A8	H	L	H	H	Output disabled	High Z	I_{CC}	
A5	5	24	A9	L	H	L	L	Read	Dout	I_{CC}	Read cycle
A4	6	23	A11	x	x	x	x	Write	Din	I_{CC}	Write cycle 1
A3	7	22	\overline{OE}	x	x	x	x		Din	I_{CC}	Write cycle 2
A2	8	21	A10								
A1	9	20	CS1								
A0	10	19	I/O8								
I/O1	11	18	I/O7								
I/O2	12	17	I/O6								
I/O3	13	16	I/O5								
V _{ss}	14	15	I/O4								

Hình 6.11: Sơ đồ chân IC HM6264 và bảng sự thật



Hình 6.12: Định thì chu kỳ đọc với $\overline{WE} = 1$

▪ Chu kỳ đọc bộ nhớ

Từ bảng sự thật và giản đồ xung định thì chu kỳ đọc, trình tự xung ở các ngõ như sau:

1. Đặt địa chỉ truy xuất ô nhớ lên bus địa chỉ A0..A12
2. Đặt tín hiệu điều khiển chọn chip $\overline{CS1} = 0, CS2=1$
3. Tạo xung tích cực mức thấp trên chân \overline{OE}
4. Data ứng với ô nhớ cần truy xuất xuất hiện ở ngõ IO

Ta lưu ý các giá trị thời gian trong chu kỳ đọc như Bảng 6.1, theo data sheet HM6264-15

Bảng 6.1: Thời gian chu kỳ đọc HM6264-15

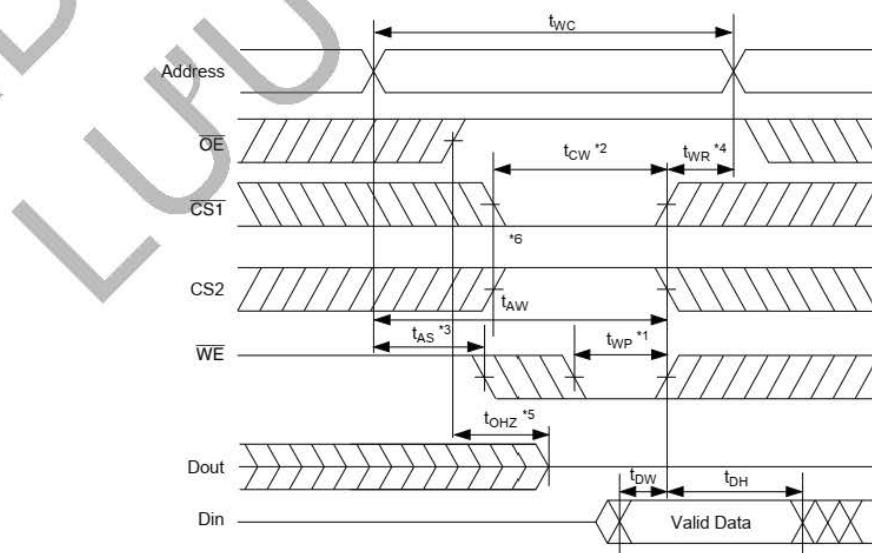
Thông số	Ký hiệu	min(ns)	max(ns)
Thời gian chu kỳ đọc	t _{RC}	150	-
Thời gian truy xuất từ địa chỉ	t _{AA}	-	150
Thời gian chọn chip đến ngõ ra	t _{OC1,2}	-	150
Thời gian \overline{OE} tích cực đến ngõ ra ổn định	t _{OE}	-	70
Thời gian ngõ ra ổn định đến địa chỉ thay đổi	t _{OH}	10	-

Thiết kế chương trình đọc bộ nhớ phải đảm bảo các thời gian định thì tương ứng không nhỏ hơn các thông số trong Bảng 6.1.

▪ Chu kỳ ghi bộ nhớ

Từ bảng sự thật và giản đồ xung định thì chu kỳ ghi, trình tự xung ở các ngõ như sau:

1. Đặt địa chỉ truy xuất ô nhớ lên bus địa chỉ A0..A12
2. Đặt tín hiệu điều khiển chọn chip $\overline{CS1} = 0, CS2=1$
3. Tạo xung tích cực mức thấp trên chân \overline{WE}
4. Đặt data cần ghi vào ô nhớ ở ngõ IO



Hình 6.13: Định thì chu kỳ ghi với $\overline{OE} = 1$

Ta lưu ý các giá trị thời gian trong chu kỳ ghi như Bảng 6.2, theo data sheet HM6264-15

Bảng 6.2: Thời gian chu kỳ ghi HM6264-15

Thông số	Ký hiệu	min(ns)	max(ns)
Thời gian chu kỳ ghi	t_{RC}	150	-
Thời gian địa chỉ ổn định đến kết thúc ghi	t_{AW}	100	-
Thời gian chọn chip đến kết thúc ghi	t_{CW}	100	-
Thời gian độ rộng xung ghi	t_{WP}	90	-
Thời gian data ổn định đến kết thúc ghi	t_{DW}	50	-

Thiết kế chương trình ghi bộ nhớ phải đảm bảo các thời gian định thì tương ứng không nhỏ hơn các thông số trong Bảng 6.2.

Ví dụ 6.7: Thiết kế mạch MCU324P giao tiếp với IC nhớ HM6264 theo yêu cầu sau:

- PORTA là hợp kênh(MUX) địa chỉ byte thấp và data ký hiệu AD0..AD7. Trong phần đầu chu kỳ bus, một xung điều khiển ALE tích cực mức 1 chọn PORTA=địa chỉ byte thấp. Phần sau chu kỳ bus PORTA=data 8 bit
 - PORTB là địa chỉ byte cao ký hiệu A8..A15 trong cả chu kỳ bus
 - Các tín hiệu điều khiển RD, WR tích cực mức 0 để đọc hoặc ghi data ở phần sau chu kỳ bus.
- Viết một chương trình minh họa việc đọc/ghi data với HM6264 sao cho chu kỳ bus là ngắn nhất. Vẽ giàn đồ xung định thì chu kỳ bus đọc/ghi theo Tosc.

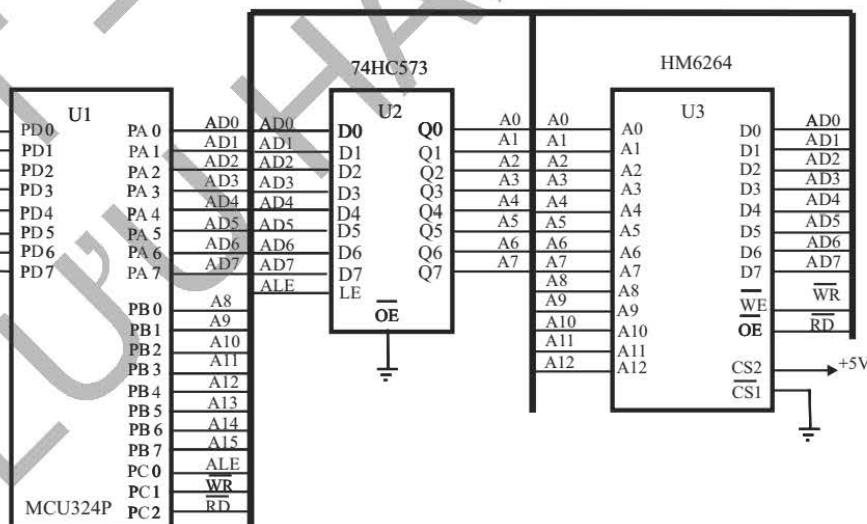
Giải:

- Để tiết kiệm chân port, PORTA làm việc như bộ MUX địa chỉ byte thấp và data AD0..AD7, ta sử dụng IC chốt D 8 bit 74HC573 và tín hiệu LE=ALE(ALE=Address Latch Enable) tích cực mức cao chốt giữ địa chỉ byte thấp A0..A7 trong phần đầu chu kỳ bus
- Kết hợp với PORTB xuất địa chỉ byte cao A8..A15 tạo thành 16 bit địa chỉ A0..A15 truy xuất tối đa $2^{16}=64KB$ địa chỉ ô nhớ
- Trong phần sau chu kỳ bus PORTA là port data D0..D7, truy xuất đọc hoặc ghi bộ nhớ tùy tín hiệu điều khiển /RD hay /WR tích cực mức thấp

Từ phân tích trên, ta có sơ đồ kết nối phần cứng như hình 6.14

Chương trình minh họa giao tiếp bộ nhớ thực hiện như sau:

- Đọc chuỗi ký tự kết thúc bằng mã NULL=\$00 lưu trong bộ nhớ chương trình(flash ROM) lần lượt ghi từng ký tự vào bộ nhớ ngoài địa chỉ đầu ST_MEM(từ \$0000-\$1FFF)
- Sau khi kết thúc ghi, lần lượt đọc từng ký tự lưu trong bộ nhớ ngoài vừa ghi, xuất ra PORTD, thời gian giữa 2 lần xuất ký tự là 1s, kết thúc đọc bộ nhớ hoặc xuất ký tự sau khi đọc được mã NULL



Hình 6.14: Thiết kế phần cứng sơ đồ mạch ví dụ 6.7

❖ Chương trình hợp ngữ VD6.7

```
.INCLUDE <M324PDEF.INC>
.EQU INPORT=PIN_A          ;ký hiệu port input
```

```

.EQU OUTPORT=PORTA      ;ký hiệu port output
.EQU IOSET=DDRA          ;ký hiệu thanh ghi hướng data
.EQU ADRPORT=PORTB       ;địa chỉ byte cao bộ nhớ
.EQU ST_MEM=$1000         ;địa chỉ đầu bộ nhớ ngoài
.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           ;PC0,PC1,PC2 output
      OUT DDRC,R16
      LDI R16,0X06           ;PC0=ALE,PC1=/WR,PC2=/RD
      OUT PORTC,R16          ;khóa các tín hiệu điều khiển
      LDI R16,0XFF
      OUT DDRD,R16           ;PortD output
      LDI ZH,HIGH(TABLE<<1)  ;con trỏ Z chứa địa chỉ đầu bảng tra
      LDI ZL,LOW(TABLE<<1)
      LDI R19,HIGH(ST_MEM)    ;R19,R18 chứa địa chỉ đầu bộ nhớ ngoài
      LDI R18,LOW(ST_MEM)

AGAIN: LPM R20,Z+          ;lấy data từ bảng tra TABLE
      RCALL MEM_WR           ;ghi ra bộ nhớ
      CPI R20,0X00
      BREQ OUT_MEM            ;tăng con trỏ địa chỉ bộ nhớ
      MOVW R24,R18
      ADIW R24,1
      MOVW R18,R24
      RJMP AGAIN              ;tiếp tục ghi data
      OUT_MEM: LDI R19,HIGH(ST_MEM)
      LDI R18,LOW(ST_MEM)
      OUT_DAT: RCALL MEM_RD   ;đọc data
      OUT PORTD,R20            ;xuất data ra PORTD
      RCALL DELAY_1S           ;delay 1s
      CPI R20,0X00
      BREQ START               ;lặp vòng lại từ đầu
      MOVW R24,R18
      ADIW R24,1
      MOVW R18,R24
      RJMP OUT_DAT             ;tiếp tục đọc data
      ;-----  

      ;MEM_RD đọc data từ bộ nhớ
      ;Sử dụng R17 nạp data cấu hình port
      ;Input R19,R18 chứa byte cao và byte thấp địa chỉ cần truy xuất
      ;Output R20 chứa data đọc từ bộ nhớ
      ;-----  

MEM_RD: LDI R17,0XFF
      OUT IOSET,R17            ;IO port output địa chỉ
      OUT DDRB,R17              ;addr.port output
      OUT OUTPORT,R18           ;xuất địa chỉ byte thấp
      SBI PORTC,0                ;tạo xung ALE mức 1
      CBI PORTC,0
      OUT ADRPORT,R19           ;xuất địa chỉ byte cao
      LDI R17,0X00
      OUT IOSET,R17            ;IO port input data

```

```

CBI    PORTC,2      ;xuất xung /RD mức thấp
IN     R20,INPORT   ;đọc data từ bộ nhớ
SBI    PORTC,2      ;kết thúc xung /RD
RET

```

```

;-----  

;MEM_WR ghi data ra bộ nhớ  

;Sử dụng R17 nạp data cấu hình port  

;Input R19,R18 chứa byte cao và byte thấp địa chỉ cần truy xuất  

;Input R20 chứa data ghi ra bộ nhớ  

;-----  


```

```

MEM_WR: LDI    R17,0XFF
        OUT   IOSET,R17    ;IO port output địa chỉ
        OUT   DDRB,R17    ;addr.port output
        OUT   OUTPORT,R18  ;xuất địa chỉ byte thấp
        SBI   PORTC,0      ;tạo xung ALE mức 1
        CBI   PORTC,0
        OUT   ADRPORT,R19  ;xuất địa chỉ byte cao
        CBI   PORTC,1      ;xuất xung /WR mức thấp
        OUT   OUTPORT,R20  ;đọc data từ bộ nhớ
        SBI   PORTC,1      ;kết thúc xung /WR
        RET

```

;---- DELAY_1S

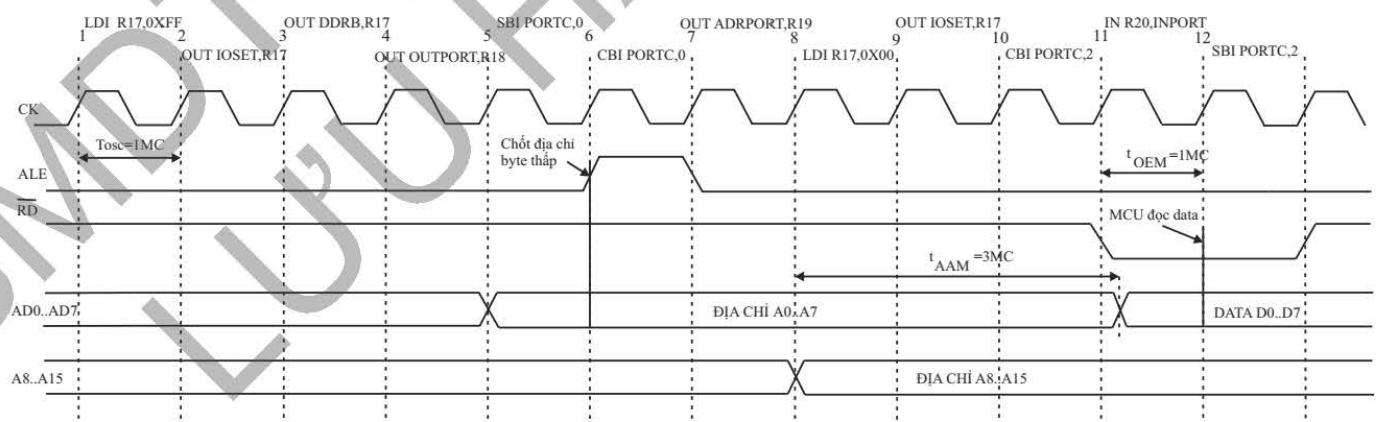
```

DELAY_1S: LDI    R23,32
        L:    LDI    R22,250  ;1MC
        L1:   LDI    R21,250  ;1MC
        L2:   DEC    R21      ;1MC
              NOP
              BRNE  L2      ;2/1MC
              DEC    R22      ;1MC
              BRNE  L1      ;2/1MC
              DEC    R23
              BRNE  L
              RET    ;4MC

```

TABLE: .DB "0123456789",0X00

■ Vẽ và phân tích giản đồ xung định thì chu kỳ đọc bộ nhớ



Hình 6.15: Giản đồ xung định thì chu kỳ đọc trong chương trình con RD_MEM

Từ chương trình con MEM_RD thiết lập giản đồ xung như sau:

- Tính số chu kỳ máy tương ứng từng lệnh
- Xem các ngõ IO đáp ứng ngay sau khi đủ thời gian thực hiện lệnh

Từ hình 6.15 ta thấy một chu kỳ bus đọc bộ nhớ kéo dài 12MC. Với $F_{osc}=8MHz$, $1MC=0.125\mu s$, thời gian chu kỳ đọc bộ nhớ $t_{RCM}=12 \times 0.125=1.5\mu s$. Các thông số định thi khác: $t_{AAM}=375ns$, $t_{OEM}=125ns$, thỏa yêu cầu Bảng 6.1.

Nếu kể cả lệnh gọi chương trình con mất 3MC và lệnh RET 4MC,sẽ mất thêm 7MC.Thời gian chu kỳ đọc bộ nhớ $=19 \times 0.125=2.375\mu s$!

Để giảm thời gian truy xuất 7MC,ta thay chương trình con bằng macro định nghĩa như sau:

.MACRO MEM_RD

```

LDI    @0,0XFF
OUT   IOSET,@0      ;IO port output địa chỉ
OUT   DDRB,@0       ;addr.port output
OUT   OUTPORT,@1    ;xuất địa chỉ byte thấp
SBI   PORTC,0        ;tạo xung ALE mức 1
CBI   PORTC,0
OUT   ADRPORT,@2    ;xuất địa chỉ byte cao
LDI    @0,0X00
OUT   IOSET,@0      ;IO port input data
CBI   PORTC,2        ;xuất xung /RD mức thấp
IN    @3,IMPORT      ;đọc data từ bộ nhớ
SBI   PORTC,2        ;kết thúc xung /RD

```

.ENDMACRO

Khi gọi macro thay cho lệnh RJMP RD_MEM:

MEM_RD R17,R18,R19,R20

Trình biên dịch sẽ thay lệnh gọi macro bằng đoạn chương trình định nghĩa macro ở trên và lần lượt thay các thông số hình thức @0,@1,@2,@3 bằng R17,R18,R19,R20 theo thứ tự lệnh gọi macro.

- Lưu ý: Trong hình 6.14 chỉ giao tiếp 1 IC HM6264 nên kết nối các chân chọn chip luôn tích cực.
- Các PORTA và PORTB vẫn có thể sử dụng cho các chức năng khác, chỉ cần giữ các chân điều khiển ALE=0,/RD=1,/WR=1!
- ❖ Người đọc tự viết chương trình C VD6.7 và vẽ giản đồ xung định thì ghi bộ nhớ từ chương trình con MEM_WR

❖ Câu hỏi ôn tập

1. Trong hình 6.14 ví dụ 6.7,tại sao phải có U2?
2. Nếu không có U2 phải sử dụng bao nhiêu chân port MCU324P giao tiếp với IC HM6264?
3. Trong chương trình hợp ngữ VD6.7,tại sao phải viết 2 dòng lệnh thứ 8 và 9:

```

LDI    R17,0X00
OUT   IOSET,R17

```

4. Theo hình 6.15,hai dòng lệnh ở chu kỳ xung CK thứ 10 và 11 có thể hoán vị nhau được không?
Giải thích?
5. Theo hình 6.14,cho biết vùng địa chỉ làm việc của bộ nhớ ngoài HM6264.

6.3.3 Mạch giải mã địa chỉ

Theo hình 6.14,MCU324P có khả năng giao tiếp tối đa với bộ nhớ 16 đường địa chỉ tương đương dung lượng 64KB.Ta có thể sử dụng 1 IC SRAM 62512 dung lượng 64KBx8 bit.

Vẫn đề đặt ra là nếu sử dụng 8 IC HM6264 để tạo bộ nhớ tổng dung lượng 64KBx8 bit,việc kết nối và chọn chip thực hiện như thế nào? Vì cả 8 chip đều có 13 đường địa chỉ trùng nhau từ A0..A12?

Để chọn đúng địa chỉ ô nhớ của chip,phải có tín hiệu chọn chip tích cực với chip được chọn và không tích cực với các chip khác.Do đó,ta phải thiết kế thêm mạch giải mã địa chỉ chọn chip

Ta xem ví dụ 6.8 về thiết kế mạch giải mã địa chỉ.

Ví dụ 6.8: Thiết kế mạch giải mã địa chỉ chọn chip để MCU324P giao tiếp với bộ nhớ ngoài dung lượng tổng 64KB như sơ đồ hình 6.14,sử dụng 8 IC HM6264,

Giải:

Tổng dung lượng bộ nhớ $64KB=2^{16}$,nên cần 16 đường địa chỉ A0..A15

IC HM6264 có dung lượng $8KB=2^{13}$,cần 13 đường địa chỉ A0..A12

Do đó,ta sử dụng các đường địa chỉ cao để phân biệt chọn chip

Trước tiên,ta thiết lập bảng phân vùng địa chỉ cho từng chip

- Cột đầu tiên liệt kê ký hiệu các tín hiệu chọn chip tích cực mức thấp là các ngõ ra của mạch giải mã địa chỉ
- Cột thứ hai chọn vùng địa chỉ chọn chip, ký hiệu địa chỉ đầu và địa chỉ cuối
- Cột thứ ba liệt kê ký hiệu các đường địa chỉ là các ngõ vào mạch giải mã địa chỉ, mức logic của địa chỉ đầu và địa chỉ cuối tương ứng

Bảng 6.3: Bảng phân vùng địa chỉ chọn 8 IC HM6264

Tín hiệu chọn chip	Vùng địa chỉ(H)	A15	A14	A13	A12	A11	A10	A9	A8	A7	A6	A5	A4	A3	A2	A1	A0
$\overline{CS0}$	0000	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	1FFF	0	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1
$\overline{CS1}$	2000	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0
	3FFF	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1
$\overline{CS2}$	4000	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	5FFF	0	1	0	1	1	1	1	1	1	1	1	1	1	1	1	1
$\overline{CS3}$	6000	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0
	7FFF	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
$\overline{CS4}$	8000	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	9FFF	1	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1
$\overline{CS5}$	A000	1	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0
	BFFF	1	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1
$\overline{CS6}$	C000	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	DFFF	1	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1
$\overline{CS7}$	E000	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0
	FFFF	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

Từ Bảng 6.3, ta thấy các đường địa chỉ thấp A0..A12 có mức logic có thể là 0 hay 1 nên tùy định. Ta loại các đường địa chỉ này và rút gọn lại bảng phân vùng địa chỉ như Bảng 6.4

Bảng 6.4: Bảng phân vùng địa chỉ rút gọn

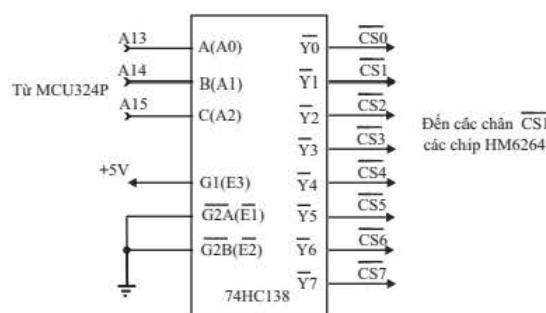
Tín hiệu chọn chip	A15	A14	A13
$\overline{CS0}$	0	0	0
$\overline{CS1}$	0	0	1
$\overline{CS2}$	0	1	0
$\overline{CS3}$	0	1	1
$\overline{CS4}$	1	0	0
$\overline{CS5}$	1	0	1
$\overline{CS6}$	1	1	0
$\overline{CS7}$	1	1	1

Cuối cùng

Bảng 6.4 chính là bảng sự thật mô tả hoạt động của mạch giải mã địa chỉ có 3 ngõ vào A15,A14,A13 và 8 ngõ ra $\overline{CS0} \dots \overline{CS7}$ tích cực mức thấp.

Từ Bảng 6.4, ta áp dụng phương pháp thiết kế hệ tổ hợp để thiết kế mạch giải mã địa chỉ.

Bảng 6.4 cũng chính là bảng sự thật của IC giải mã 3 sang 8 74HC138(xem data sheet ở phần phụ lục). Ta sử dụng IC 74HC138 thiết kế mạch giải mã địa chỉ như hình 6.16.



Hình 6.16: Mạch giải mã địa chỉ ví dụ 6.7 sử dụng IC 74HC138

Ta có thể áp dụng phương pháp giải mã địa chỉ kết hợp với đặc tính truy xuất đọc/ghi để mở rộng giao tiếp các thiết bị ngoại vi tương tự như ví dụ 6.7 và 6.8.

❖ Thiết kế mạch giải mã địa chỉ

Các bước thực hiện thiết kế mạch giải mã địa chỉ như sau:

1. Lập bảng phân vùng địa chỉ chọn chip cho các thiết bị ngoại vi:
 - Xác định chiều dài (dung lượng) của mỗi vùng địa chỉ chọn chip, địa chỉ đầu, địa chỉ cuối
 - Đặc tính truy xuất đọc và/hoặc ghi của thiết bị ngoại vi là các ngõ vào
 - Mức tích cực tín hiệu chọn chip ngõ ra
 - Tính tổng chiều dài vùng địa chỉ, suy ra số địa chỉ cần sử dụng
 - Nên chọn các vùng địa chỉ bằng hoặc tỉ lệ bội số với nhau, mạch điện sẽ đơn giản hơn
2. Từ bảng phân vùng địa chỉ chi tiết ở bước 1, loại bỏ các địa chỉ thấp có cùng logic tùy định, lọc ra bảng phân vùng địa chỉ rút gọn chỉ gồm các địa chỉ cao
3. Từ bảng phân vùng địa chỉ rút gọn và các tín hiệu truy xuất đọc/ghi, áp dụng thiết kế hệ tổ hợp cho mạch giải mã địa chỉ

➤ Lưu ý: Để đảm bảo đáp ứng thời gian truy xuất theo cách định rõ xung định thì, thời gian đáp ứng hay truyền đạt của mạch giải mã địa chỉ phải càng ngắn càng tốt!

Ví dụ 6.8b: Thiết kế mạch điện MCU324P giao tiếp với các thiết bị ngoại vi gồm bộ nhớ 1 IC HM6264, 1 IC đệm ngõ vào 74HC244, 2 IC chốt ngõ ra 74HC573.

Giải:

Ta sử dụng phương pháp giao tiếp bus địa chỉ, data và giải mã địa chỉ cho thiết kế mạch.

Lập bảng phân vùng địa chỉ các thiết bị ngoại vi:

- Chọn vùng địa chỉ chọn chip mỗi thiết bị ngoại vi 8KB
- U3=HM6264 chọn chip $\overline{CS0}$, truy xuất $\overline{RD}, \overline{WR}$ tương ứng $\overline{OE}, \overline{WE}$ trên chip
- U4=74HC573 chọn chip $CS1 = LE$, truy xuất \overline{WR} tạo xung $LE=1$
- U5=74HC573 chọn chip $CS2 = LE$, truy xuất \overline{WR} tạo xung $LE=1$
- U6=74HC244 chọn chip $\overline{CS3} = \overline{OE}$, truy xuất \overline{RD} tạo xung $\overline{OE}=0$

Bảng 6.5: Bảng phân vùng địa chỉ ví dụ 6.8

Tín hiệu chọn chip	Đặc tính truy xuất	Vùng địa chỉ(H)	A15 A14 A13 A12 A11 A10 A9 A8 A7 A6 A5 A4 A3 A2 A1 A0
$\overline{CS0}$ (U3)	$\overline{RD}, \overline{WR}$	0000 1FFF	0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
$CS1$ (U4)	\overline{WR}	2000 3FFF	0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0
$CS2$ (U5)	\overline{WR}	4000 5FFF	0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0
$\overline{CS3}$ (U6)	\overline{RD}	6000 7FFF	0 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0

Từ Bảng 6.5, ta loại được các đường địa chỉ từ A0..A12. Như vậy mạch giải mã địa chỉ sẽ có các ngõ vào địa chỉ A13, A14, A15, các ngõ vào truy xuất $\overline{RD}, \overline{WR}$, các ngõ ra $\overline{CS0}, \overline{CS1}, \overline{CS2}, \overline{CS3}$.

Để đơn giản thiết kế, ta chọn IC 74HC138. Từ Bảng 6.5 rút gọn, ta có:

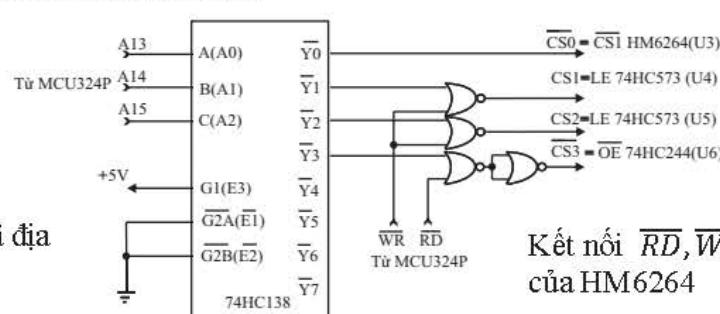
$$\overline{CS0} = \overline{Y0} \cdot (\overline{RD} + \overline{WR})$$

$$CS1 = \overline{Y1} \cdot \overline{WR} = \overline{Y1} + \overline{WR}$$

$$CS2 = \overline{Y2} \cdot \overline{WR} = \overline{Y2} + \overline{WR}$$

$$CS3 = \overline{Y3} \cdot \overline{RD} = Y3 + RD$$

Sơ đồ mạch giải mã địa chỉ như hình 6.17.



Kết nối $\overline{RD}, \overline{WR}$ đến $\overline{OE}, \overline{WE}$ của HM6264

Hình 6.17: Mạch giải mã địa chỉ ví dụ 6.8b

❖ Câu hỏi ôn tập

- Một bộ nhớ dung lượng 28KBx8 bit cần bao nhiêu đường địa chỉ truy xuất?
- Trong hình 6.17 ví dụ 6.8b,tại sao ngõ ra $\overline{CS0}$ không tổ hợp với các ngõ $\overline{RD}, \overline{WR}$ mà nối thẳng đến chân $\overline{CS1}$ của IC HM6264
- Trong hình 6.17,nếu các ngõ ra $\overline{Y1}, \overline{Y2}, \overline{Y3}$ kết nối thẳng đến các chân LE0,LE1 của U2,U3 và \overline{OE} của U4,mạch sẽ hoạt động ra sao?
- Theo Bảng 6.5,A15=0 với mọi trường hợp,thay vì kết nối chân C của IC 74HC138 với A15 ta hở chân A15 từ MCU324P và nối chân C xuống GND.Vùng địa chỉ làm việc của các chip còn đúng không?Giải thích tại sao?
- Lập lại câu 4,thêm thay đổi hở chân $\overline{G2A}$ và kết nối với chân A15 từ MCU324P.

6.3.4 Giao tiếp thanh ghi dịch IC 74HC595,IC74HC165

Một phương pháp mở rộng port giao tiếp ngoại vi là giao tiếp với thanh ghi dịch.MCU324P chỉ cần cơ bản 2 ngõ giao tiếp,một ngõ tạo xung clock,một ngõ dịch data nối tiếp ra/vào thanh ghi dịch.

Hình 6.18 giới thiệu sơ đồ chân và bảng sự thật IC 74HC595 ghi dịch vào/nối tiếp/ra song song 8 bit.Cấu hình IC gồm tầng ghi dịch,tầng lưu trữ,tầng ra 3 trạng thái(xem chi tiết data sheet ở phần phụ lục).

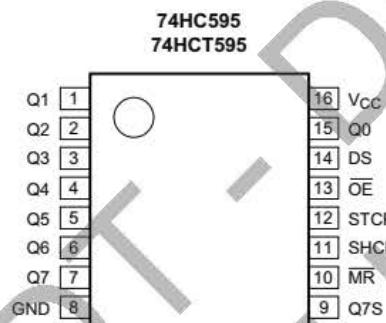
Các thông số đặc tính động IC74HC595:

- Thời gian truyền đạt max $t_{pdmax}=53\text{ns}$
- Thời gian cầm/cho phép xuất $t_{dis}=t_{en}=45\text{ns}$
- Chu kỳ ghi dịch/nạp (tiêu chuẩn) $t_w=22\text{ns}$

Với $1MC=125\text{ns}$,một lệnh của MCU324P hoàn toàn đáp ứng các thông số trên

Từ bảng sự thật hình 6.18,tóm tắt các mô thức hoạt động thông dụng của IC 74HC595 như sau:

- Reset thanh ghi và ngõ ra: $\overline{MR} = 0, \overline{OE} = 0, STCP = \uparrow$
- Ghi dịch,ngõ ra không đổi: $\overline{MR} = 1, \overline{OE} = 0, SHCP = \uparrow$ DS=DI(data in),Q7S=Q6S,Qn=Q0
- Cho phép xuất: $\overline{MR} = 1, \overline{OE} = 0, STCP = \uparrow$ Qn=QnS(data out)
- Ghi dịch,xuất ngõ ra: $\overline{MR} = 1, \overline{OE} = 0, SHCP = STCP = \uparrow$ DS=DI ,Qn=QnS,Q7S=Q6S
(ngõ ra song song bằng giá trị ghi dịch trước đó)

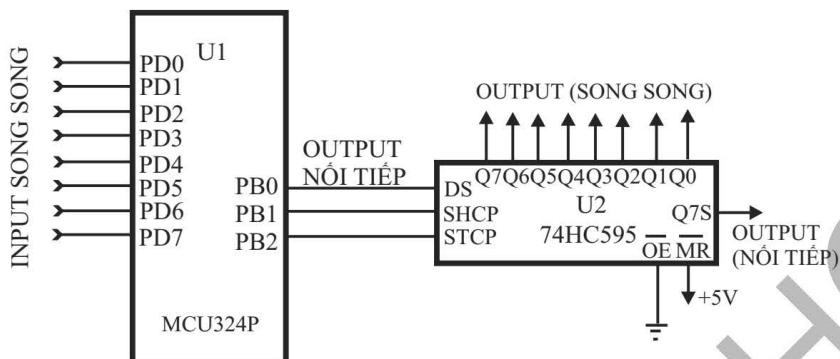


Ký hiệu	Mô tả
Q0,Q1,...,Q7	Ngõ ra song song
Q7S	Ngõ ra nối tiếp
DS	Ngõ vào nối tiếp
MR	Ngõ vào reset chính tích cực mức 0
SHCP	Ngõ vào xung clock ghi dịch
STCP	Ngõ vào xung clock lưu trữ
OE	Ngõ vào cho phép xuất tích cực mức 0

Ngõ vào điều khiển				Ngõ vào	Ngõ ra		Chức năng
SHCP	STCP	OE	MR	DS	Q7S	Qn	
X	X	L	L	X	L	Q0	Reset thanh ghi ngõ ra không đổi
X	\uparrow	L	L	X	L	L	Reset thanh ghi và ngõ ra
X	X	H	L	X	L	Z	Reset thanh ghi ngõ ra tổng trở cao
\uparrow	X	L	H	DI	Q6S	Q0	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	\uparrow	L	H	X	Q0	QnS	Mode xuất data,ngõ ra các tầng ghi dịch được phép xuất ngõ ra song song
\uparrow	\uparrow	L	H	DI	Q6S	QnS	Mode ghi dịch và xuất data ngõ ra bằng giá trị ghi dịch trước đó

Hình 6.18: Sơ đồ chân và bảng sự thật IC 74HC595

Hình 6.19 trình bày sơ đồ kết nối thanh ghi dịch IC 74HC595, mở rộng port xuất data từ nối tiếp chuyển sang song song. Để đơn giản, ta không sử dụng điều khiển reset và cho phép xuất. Ta có thể mở rộng thêm nhiều ngõ ra bằng cách kết nối ngõ Q7S tầng trước với ngõ SHCP tầng sau và nối chung các ngõ STCP.



Hình 6.19: Giao tiếp thanh ghi dịch IC 74HC595

Ví dụ 6.9: Từ sơ đồ hình 6.19 viết một chương trình nhập data từ PORTD, xuất ra dưới dạng nối tiếp giao tiếp với thanh ghi dịch IC 74HC595.

❖ Chương trình hợp ngữ VD6.9

```

INCLUDE <M324PDEF.INC>
.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=0
      CBI PORTB,2 ;CK xuất=0
START: IN  R17,PIND ;đọc data từ PORTD
      RCALL SHIFT_OUT;gọi ctc ghi dịch
      RJMP START
;
;  
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 ;R18 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 ck dịch
BIT_0:  CBI PORTB,0 ;xuất bit 0
SH_CK:  SBI PORTB,1 ;xuất xung ck dịch
        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
        CBI PORTB,2
        ROL R17 ;dịch lần cuối phục hồi nội dung R17

```

❖ **Chương trình C VD6.9:**

```
#include <avr/io.h>
#define import PINB      //định nghĩa import=PINB
#define outport PORTB    //định nghĩa outport=PORTB
void SHIFT_OUT(unsigned char x) ;//khai báo hàm SHIFT_OUT
unsigned char tam,asc;
int main()
{
    DDRD=0x00 ;//định nghĩa PORTD input
    DDRB=0X07 ;//PB0..PB2 output
    PORTB=0Xf8 ;//các xung ck=0
    while(1)
    {
        tam=import ;//đọc dat từ PORTD
        SHIFT_OUT(tam) ;//gọi hàm ghi dịch
    }
}
void SHIFT_OUT(unsigned char dat)
{
    unsigned char tam,i;
    tam=0x80 ;//bắt đầu xét bit 7
    for (i=1;i<=8;i++) //lặp vòng 8 lần
    {
        if((dat&tam)==tam) //bit xét=1?
        {
            PORTB=0xf9; //bit xét=1,PB0=1
            PORTB=0xfb; //xuất xung ck dịch ra PB1
            PORTB=0xf9;
        }
        else
        {
            PORTB=0xf8; //bit xét=0,PB0=0
            PORTB=0xfa; //xuất xung ck dịch ra PB1
            PORTB=0xf8;
        }
        tam=tam>>1; //xét bit kế tiếp
    }
    PORTB=0xfc; //xuất xung ck xuất data ngõ ra
    PORTB=0xf8;
}
```

➤ Lưu ý: Chương trình C xử lý bit không thuận lợi bằng hợp ngữ!!!

Để chuyển data ngõ vào song song sang nối tiếp và nhập vào MCU324P, ta sử dụng IC ghi dịch vào song song ra nối tiếp 74HC165.

➤ Người đọc tự tham khảo data sheet IC74HC165 và thiết kế giao tiếp MCU324P với IC74HC165

❖ **Câu hỏi ôn tập**

1. Trong sơ đồ hình 6.19, nếu muốn mở rộng thêm 8 ngõ ra song song, phải kết nối thêm IC 74HC595 như thế nào?
2. Trong sơ đồ hình 6.19, nếu kết nối chân STCP chung với chân SHCP, mạch sẽ làm việc ra sao?
3. Từ hình 6.19, nếu muốn điều khiển ngõ ra Q0..Q7 ở mức tổng trở cao khi ghi dịch data và cho phép xuất ngay sau khi dịch xong data, phải kết nối mạch như thế nào?

4. Trong chương trình con SHIFT_OUT, tính thời gian ghi dịch 8 bit cho đến thời điểm bắt đầu xuất cạnh lén xung STCP.
5. Trong chương trình con SHIFT_OUT, thay vì đặt data ra chân PB0 trước và xuất xung CK dịch ra chân PB1 sau, ta có thể thực hiện bằng đoạn lệnh sau được không, ví dụ dịch xuất ra bit 1:

```
LDI    R17,0XF3
OUT    PORTB,R17
LDI    R17,0XF0
OUT    PORTB,R17
```

Giải thích lý do.

6.4 Giao tiếp hiển thị LED 7 đoạn. LED ma trận

Trong phần này ta sẽ khảo sát các dạng giao tiếp với bộ hiển thị LED 7 đoạn chủ yếu hiển thị ký tự số và LED ma trận hiển thị điểm theo đồ họa.

6.4.1 Giao tiếp hiển thị LED 7 đoạn

Ta xem lại phần cấu tạo, hoạt động và cách tạo mã lái LED 7 đoạn đã trình bày ở chương 0.

Bảng 6.6 tóm tắt mã 7 đoạn lái LED 7 đoạn AC (anode chung) hiển thị số Hex, kết nối các chân đoạn với các bit data tương ứng.

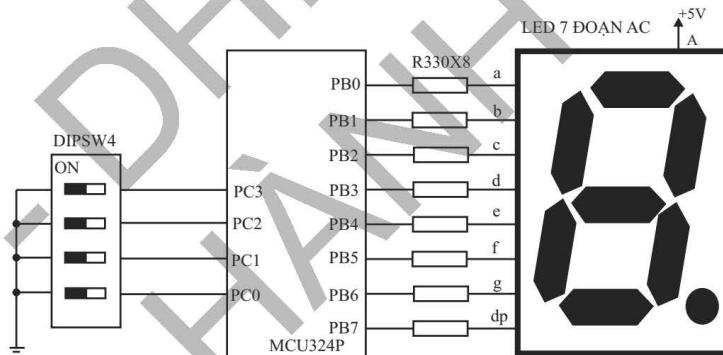
Bảng 6.6: Mã 7 đoạn lái LED 7 đoạn AC

Đoạn	dp	g	f	e	d	c	b	a
Data	D7	D6	D5	D4	D3	D2	D1	D0
Ký tự	0	1	2	3	4	5	6	7
Mã 7 đoạn (H)	C0	F9	A4	B0	99	92	82	F8

Ký tự	0	1	2	3	4	5	6	7
Mã 7 đoạn (H)	C0	F9	A4	B0	99	92	82	F8

Ví dụ 6.10: Sơ đồ mạch giao tiếp LED 7 đoạn AC như hình 6.20. Viết một chương trình đọc data từ DIPSW4 và hiển thị mã Hex của data ra LED 7 đoạn AC.

Giải:



Hình 6.20: Mạch điện MCU324P giao tiếp trực tiếp LED 7 đoạn AC

DIPSW4 gồm 4 SW trong một chip, gạt SW sang vị trí ON là đóng SW, vị trí còn lại là SW. PortB lái trực tiếp các chân đoạn do dòng ngõ ra các chân port có thể đạt 20mA, chỉ cần phân cực các đoạn sáng danh định 10mA, 2V như LED đơn.

❖ Chương trình hợp ngữ VD6.10

```
.INCLUDE <M324PDEF.INC>
.EQU OUTPORT=PORTB      ;ký hiệu port output
.EQU IOSETB=DDRB         ;ký hiệu thanh ghi hướng data
.ORG 0
.RJMP MAIN
.ORG 0X40
MAIN: LDI R16,HIGH(RAMEND) ;đưa stack lên vùng d/c cao
      OUT SPH,R16
      LDI R16,LOW(RAMEND)
```

```

        OUT    SPL,R16
        LDI    R16,0XFF          ;PORTB output
        OUT    IOSETB,R16
        LDI    R16,0XF0          ;PC0..PC3 input
        OUT    DDRC,R16          ;
        LDI    R16,0X0F          ;R kéo lên PC0..PC3
        OUT    PORTC,R16
START:   IN     R17,PINC      ;đọc data từ DIPSW4
        ANDI   R17,0X0F          ;che 4 bit thấp
        LDI    ZH,HIGH(TAB_7SA<<1); Z trả 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
        OUT    OUTPORT,R17        ;xuất mã 7 đoạn
        RJMP   START
TAB_7SA: .DB    0XC0,0XF9,0XA4,0XB0,0X99,0X92,0X82,0XF8,0X80,0X90,0X88,0X83
        .DB    0XC6,0XA1,0X86,0X8E

```

- Phần chủ yếu của chương trình là khai báo bảng tra mã 7 đoạn, bắt đầu từ địa chỉ TAB_7SA là mã 7 đoạn ký tự 0, sau đó theo thứ tự tăng đến F.
- Tra mã 7 đoạn bằng cách lấy địa chỉ nền TAB_7SA cộng thêm offset chính là data đọc từ DIPSW4.

❖ Chương trình C VD6.10

```

#include <avr/io.h>
#define outport PORTB //định nghĩa outport=PINB
#define iosetb DDRB // định nghĩa ioset=DDRB
unsigned char tab_7sa[]={0xc0,0xf9,0xa4,0xb0,0x99,0x92,0x82,0xf8,0x80,0x90,0x88,0x83,
                        0xc6,0xa1,0x86,0x8e};//bảng tra mã 7 đoạn
unsigned char tam;
int main()
{
    iosetb=0xff //định nghĩa port=output
    DDRC=0xf0 //định nghĩa PC0..PC3=input
    PORTC=0x0f //R kéo lên PC0..PC3
    while(1)
    {
        tam=PINC&0x0f //đọc data từ DIPSW4,lấy 4 bit thấp
        outport=tab_7sa[tam];//xuất mã 7 đoạn ứng với data đọc từ DIPSW4
    }
}

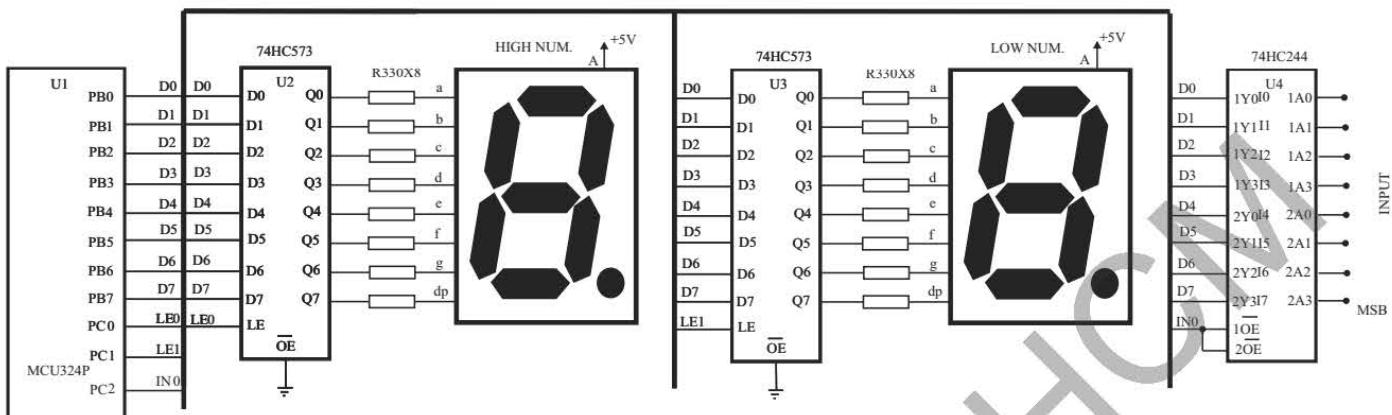
```

- Trong chương trình có khai báo dãy(array) tab_7sa chứa data là mã 7 đoạn mặc định trong vùng data. Trình biên dịch C sẽ tự động chuyển dãy data này vào bộ nhớ SRAM khi mới khởi động (thông thường đặt tại địa chỉ 0x100 của SRAM). Do đó khi truy xuất data từ dãy, trình biên dịch C thực hiện truy xuất dãy bình thường, không quan tâm kiểu bộ nhớ cần truy xuất như trong trình hợp ngữ!
- Trường hợp hiển thị nhiều LED 7 đoạn, ta có thể sử dụng mở rộng port dùng phương pháp mạch chốt D 8 bit (phương pháp chốt), minh họa qua ví dụ 6.11.

Ví dụ 6.11: Từ ví dụ 6.6 hình 6.10, kết nối 2 LED 7 đoạn AC lần lượt với các ngõ ra U2 và U3 tương ứng. Viết một chương trình đọc byte data từ U4, hiển thị dưới dạng số Hex ra 2 LED 7 đoạn, trọng số cao xuất ra U2, trọng số thấp xuất ra U3.

Giải:

Sơ đồ kết nối như hình 6.21.Về cơ bản chương trình tương tự như ví dụ 6.6, chỉ cần thay chương trình con chuyển sang mã ASCII bằng chương trình con chuyển sang mã 7 đoạn



Hình 6.21: Sơ đồ hiển thị mở rộng port phương pháp chốt

❖ Chương trình hợp ngữ VD6.11:

```

INCLUDE <M324PDEF.INC>
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

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
       SWAP R17 ;hoán vị sang 4 bit thấp
       LDI R16,0XFF
       OUT IOSET,R16 ;khai báo PORT output
       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
       CBI PORTC,0 ;khóa U2
       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
       CBI PORTC,1 ;khóa U3

```

RJMP START

```
;-----  
;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 trả 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
```

```
;-----  
TAB_7SA: .DB 0XC0,0XF9,0XA4,0XB0,0X99,0X92,0X82,0XF8,0X80,0X90,0X88,0X83  
        .DB 0XC6,0XA1,0X86,0X8E
```

❖ Chương trình C VD6.11:

```
#include <avr/io.h>  
#define import PINB      //định nghĩa import=PINB  
#define ioset DDRB       // định nghĩa ioset=DDRB  
#define outport PORTB    //định nghĩa outport=PORTB  
#define cont PORTC       //định nghĩa cont=PORTC  
const char dis_io=0xf4 ;//mã khóa các ngoại vi  
const char en_u2=0xf5 ;//mã mở U2  
const char en_u3=0xf6 ;//mã mở U3  
const char en_u4=0xf0 ;//mã mở U4  
unsigned char get_7seg(unsigned char x) ;//khai báo hàm HEX_ASC  
unsigned char tam,i;  
int main()  
{  
    DDRC=0x07      ;//định nghĩa PC0,PC1,PC2 là output  
    cont=dis_io     ;//khóa các ngoại vi  
    while(1)  
    {  
        ioset=0x00  ;//khai báo port input  
        cont=en_u4 ;//mở U4  
        tam=import ;//nhập data từ U4  
        cont=dis_io ;//khóa U4  
        i=tam>>4   ;//dịch data sang 4 bit thấp  
        ioset=0xff   ;//khai báo port output  
        outport=get_7seg(i) ;//chuyển sang mã ASCII và xuất ra U2  
        cont=en_u2 ;//mở U2  
        cont=dis_io ;//khóa U2  
        i=tam&0x0f;//che 4 bit cao  
        outport=get_7seg(i) ;//chuyển sang mã ASCII và xuất ra U3  
        cont=en_u3 ;//mở U3  
        cont=dis_io ;//khóa U3  
    }  
}  
unsigned char get_7seg(unsigned char x) //hàm chuyển mã 7 đoạn  
{  
    unsigned char tab_7sa[]={0xc0,0xf9,0xa4,0xb0,0x99,0x92,0x82,0xf8,0x80,0x90,0x88,  
                            0x83,0xc6,0xa1,0x86,0x8e};//bảng tra mã 7 đoạn
```

```

x=tab_7sa[x] ;//lấy mã 7 đoạn
return(x) ;
}

```

Từ ví dụ 6.11,với phương pháp chốt hiển thị LED 7 đoạn,cứ hiển thi thêm 1 LED 7 đoạn phải thêm 1 IC chốt D 8 bit và 1 ngõ điều khiển LE mở chốt.Ví dụ nếu muốn thiết kế bộ hiển thi 4 LED 7 đoạn,ta phải cần 4 IC chốt D 8 bit và 1 port 8 bit xuất data,4 ngõ điều khiển mở chốt,tổng cộng cần 12 đường vào/ra của MCU324P.Ta có thể kết hợp thêm mạch giải mã/phân đường từ 2 sang 4 tạo ngõ ra 4 đường tín hiệu chốt,ngõ vào chỉ cần 2 đường vào/ra của MCU324P,tổng cộng chỉ cần 10 đường vào/ra của MCU324P.

Một phương pháp khác tiết kiệm mạch chốt khi mở rộng bộ hiển thị nhiều ký tự gọi là phương pháp quét.Lợi dụng hiện tượng lưu ảnh mắt người,tại một thời điểm chỉ cấp nguồn cho 1 LED sáng,các LED còn lại tối,cứ thế tuần tự cho từng LED và sau đó lập vòng lại.Để đảm bảo mắt người có cảm giác các LED đều sáng liên tục,tần số quét lập vòng không nhỏ hơn 50Hz là đạt.

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

Để LED sáng tương đương như phân cực DC(phương pháp chốt),phải có dòng trung bình qua mỗi đoạn bằng 10mA và điện áp thuận 2V.Tùy đây suy ra dòng định qua 1 đoạn:

$$I_{SEG} = n \times 10 \text{ (mA)} \quad (6.5)$$

với n= tổng số LED 7 đoạn cần hiển thị

Ví dụ 6.12: 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 cát 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 1port và hiển thị giá trị thập phân tương ứng ra bộ hiển thị.

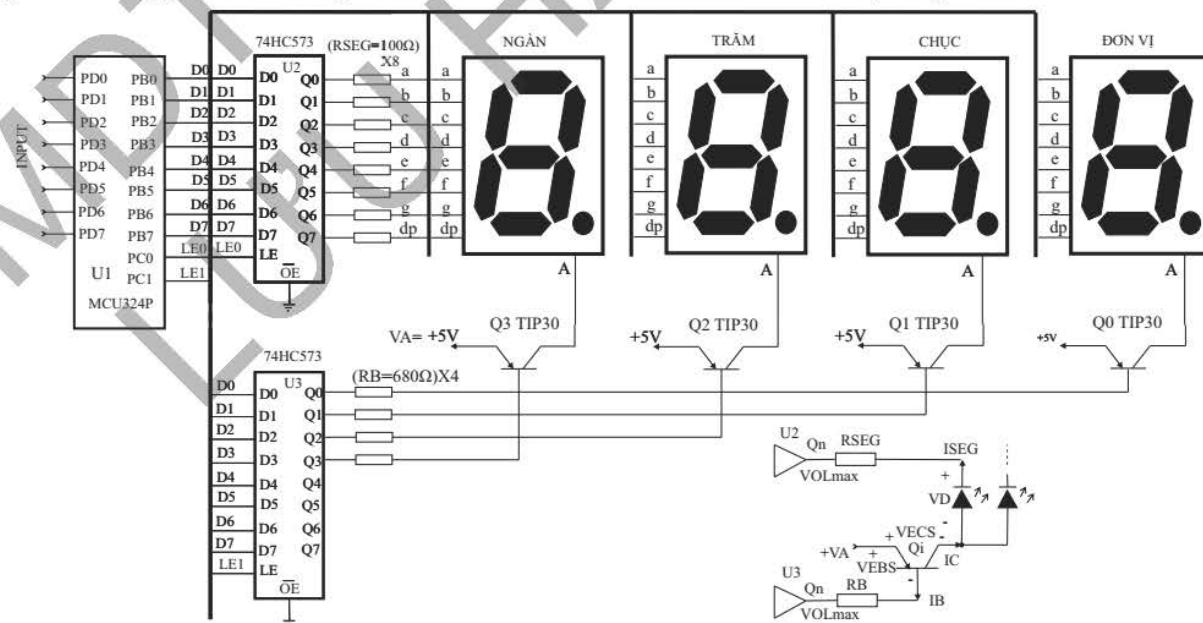
Giải:

Hình 6.22 giới thiệu sơ đồ thiết kế bộ hiển thi 4 LED AC dùng phương pháp quét

- U1 MCU324P giao tiếp mở rộng port với 2 IC chốt U2 và U3,PORTB xuất data 8 bit, PC0,PC1 xuất các tín hiệu điều khiển mở chốt
- U2 IC74HC573 chốt data xuất ra là mã 7 đoạn AC lái các chân đoạn,LE0=PC0 tín hiệu mở chốt
- U3 IC74HC573 chốt data xuất ra 4 bit thấp là mã quét lái Anode LED7 đoạn,mức 0 tích cực
- Q0..Q3 TIP30 4BJT pnp cấp nguồn Anode LED 7 đoạn lái dòng từng LED

Do n=4 nên theo (6.5) dòng định qua đoạn $I_{SEG}=4 \times 10=40\text{mA}$.Trong trường hợp xấu nhất,ngõ ra U2 lái cả 4 LED có cùng 1 đoạn sáng,dòng ngõ ra U2 $IOLmax=40\text{mA}$ DC vượt quá khả năng chịu đựng tối đa là 35mA .Như vậy ta phải thêm vào mỗi ngõ ra U2 một mạch đệm khuếch đại dòng.

Để có thể kết nối trực tiếp ngõ ra U2 với chân đoạn,ta hạ dòng qua chân đoạn xuống còn 5mA , chấp nhận sáng yếu hơn.Dòng $I_{SEG}=20\text{mA},IOLmax=20\text{mA}$ nên U2 đáp ứng được!



Hình 6.22: Sơ đồ thiết kế mạch quét hiển thị 4 LED AC

Tính trong trường hợp xấu nhất các đoạn đều sáng,dòng đỉnh tổng qua Anode LED=8x20=160mA. Ngõ ra IC 74HC573 không thể lái trực tiếp Anode LED được,do đó ta phải thêm BJT khuếch đại dòng Q0..Q3 lái các Anode LED.Mức 0 ngõ ra của U3 sẽ làm các BJT dẫn bão hòa,V_{ECS}≈0.3V,nguồn 5V sẽ được cấp cho Anode LED tương ứng.

Tính điện trở hạn dòng chân đoạn(xem hình 6.22): Cho U2 có V_{OLmax}=0.7V(theo data sheet)

$$R_{SEG} = (V_A - V_{ECS} - V_D - V_{OLmax})/I_{SEGP} \quad (6.6)$$

$$R_{SEG} = (5 - 0.3 - 2 - 0.7)/0.02 = 100\Omega$$

Tính các điện trở phân cực B cho Q0..Q3: Chọn Q0..Q3=TIP30

Các BJT dẫn bão hòa cho $\beta_{min}=40$,V_{ECS}=0.3V,V_{EBS}=0.8V(xem data sheet TIP30)

Trường hợp sáng cả 8 đoạn I_{Cmax}=20x8=160mA:

Để BJT bão hòa phải thỏa mãn điều kiện:

$$\beta I_B > I_C \quad (6.7)$$

$$\beta \frac{V_A - V_{EBS} - V_{OLmax}}{R_B} > I_C \quad (6.8)$$

$$R_B < \beta \frac{V_A - V_{EBS} - V_{OLmax}}{I_C} \quad (6.9)$$

$$R_B < 40 \frac{5 - 0.8 - 0.7}{0.16} = 875\Omega \rightarrow \text{Chọn } R_B = 680\Omega$$

$$I_B = \frac{3.5}{680} = 5.14\text{mA}$$

74HC573 có dòng I_{OLmax}=35mA>5.14mA nên lái được I_B BJT

❖ Chương trình hợp ngữ VD6.12

```

INCLUDE <M324PDEF.INC>
.EQU OUTPORT=PORTB
.EQU INPORT=PIND
.EQU SR_ADR=0X100
.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 DDRB,R16
      LDI R16,0X00
      OUT DDRD,R16
      LDI R20,0X00
      LDI R21,0X00
      RCALL SCAN_4LA
START: IN R17,INPORT ;đọc data
       RCALL BIN8_BCD ;etc chuyển sang số BCD
       RCALL SCAN_4LA ;etc quét 4 LED
       RJMP START
;-----
;SCAN_4LA hiển thị 4 LED AC bằng phương pháp quét
;Input: R21,R20=số BCD nén(ngàn-chục),(trăm đơn vị)
;Sử dụng etc BCD_UNP tách số BCD nén thành không nén

```

;Sử dụng ctc DELAY_1MS,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
LDI	XH,HIGH(SR_ADR);X	trỏ địa chỉ đầu SRAM
LDI	XL,LOW(SR_ADR)	
LOOP:	LDI	R17,0XFF ;làm tối các đèn
	OUT	OUTPORT,R17
	SBI	PORTC,1 ;mở U3
	CBI	PORTC,1 ;khóa U3
	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
	CBI	PORTC,0 ;khóa U2
	OUT	OUTPORT,R19 ;xuất mã quét anode LED
	SBI	PORTC,1 ;mở U3
	CBI	PORTC,1 ;khóa U3
	RCALL	DELAY_1MS ;tạo trễ 1ms sáng đèn
	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	

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 ;cắt 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 ;cắt vào SRAM,tăng địa chỉ SRAM
MOV R17,R21 ;thực hiện tương tự với số BCD nén cắt trong R21
ANDI R17,0X0F
ST X+,R17
MOV R17,R21
SWAP R17
ANDI R17,0X0F
ST X+,R17
RET

;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

;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 đầu lên cao

OR R20,R16 ;dán dư số phép chia lần 2 vào 4 bit thấp

MOV R21,R17 ;R21=dư số sau cùng

RET

;DIV_8_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 chia 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

;

DELAY_1MS:

LDI R16,8 ;1MC sử dụng R16

MOV R15,R16 ;1MC nạp data cho R15

LDI R16,250 ;1MC sử dụng R16

L1: MOV R14,R16 ;1MC nạp data cho R14

L2: DEC R14

;1MC

NOP

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

- Đã mang tính tổng quát, chương trình con BIN8_BCD chuyển từ số nhị phân 8 bit sang số BCD 3 digit,kết quả là số BCD nén cắt trong R21,R20.

- Trong chương trình con SCAN_4LA,để thuận tiện tạo vòng lặp 4 lần tương ứng quét 4 đèn,tà sử dụng chương trình con BCD_UNP tách số BCD nén thành không nén.Chương trình con này có thể không cần thiết nếu từ chương trình con BIN8_BCD cho kết quả là số BCD không nén,như trong chương trình C thực hiện ở phần sau!

- Có thể tùy chỉnh thời gian trễ trong ctc DELAY_1MS để điều chỉnh độ sáng đèn
- Chương trình C VD6-12**

```
#include <avr/io.h>
#define import PIND //định nghĩa import=PIND
#define outport PORTB //định nghĩa outport=PORTB
#define cont PORTC //định nghĩa cont=PORTC
const char dis_io=0xfc;//mã khóa các ngoại vi
const char en_u2=0xfd;//mã mở U2
const char en_u3=0xfe;//mã mở U3
const int sr_adr=0x200;//địa chỉ đầu SRAM
void SCAN_4LA() //khai báo hàm quét 4 LED AC
void BIN8_BCD(unsigned char x) //khai báo hàm chuyển đổi nhị phân sang BCD
unsigned char get_7seg(unsigned char x) // khai báo hàm chuyển mã 7 đoạn
void delay_ms(unsigned int n) //khai báo hàm delay 1ms
unsigned char i,tam,*p;
int main()
{
    DDRC=0x03 //định nghĩa PC0,PC1,PC2 là output
    cont=dis_io //khóa các ngoại vi
    DDRB=0xFF //PORTB output
    DDRD=0x00 //PORTD input
    p=sr_adr //trỏ địa chỉ đầu vùng nhớ SRAM
    for(i=1;i<=4;i++) //khởi động vùng nhớ
    {
        *p=0x00 //xóa ô nhớ
        p++ //tăng địa chỉ ô nhớ
    }
    p=sr_adr //trỏ địa chỉ đầu vùng nhớ SRAM
    SCAN_4LA() //hiển thị 0
    while(1)
    {
        tam=import //nhập data từ PORTD
        BIN8_BCD(tam); //chuyển từ nhị phân sang BCD
        p=sr_adr //trỏ địa chỉ đầu vùng nhớ
        SCAN_4LA() //hiển thị giá trị nhập
    }
}
void BIN8_BCD(unsigned char x) //hàm chuyển mã nhị phân 8 bit sang BCD 3 digit
{
    p=sr_adr //trỏ địa chỉ đầu bộ nhớ
    *p=tam%10 //cắt số dư vào ô nhớ
    tam=tam/10 //chia 10 và gán số bị chia=thương số
    p++ //tăng địa chỉ ô nhớ
    *p=tam%10 //cắt số dư lần 2
    tam=tam/10 //chia lần 2
    p++ ;
    *p=tam //cắt số dư sau cùng
}
void SCAN_4LA()
{
    unsigned char scan_a,led_code,j;
    scan_a=0xfe //mã quét đèn đầu
    for (j=1;j<=4;j++)
    {

```

```

        outport=0xff;//xuất mã tắt đèn cắt nguồn anode LED
        cont=en_u3 ;//mở chốt U3
        cont=dis_io ;//khóa chốt U3
        led_code=*p;//lấy số BCD trong bộ nhớ
        p++          ;//tăng con ntrô bộ nhớ
        outport=get_7seg(led_code); //chuyển sang mã 7 đoạn và xuất ra port
        cont=en_u2 ;//mở chốt U2
        cont=dis_io ;//khóa chốt U2
        outport=scan_a;//xuất mã lái anode
        cont=en_u3 ;//mở chốt U3
        cont=dis_io ;//khóa chốt U3
        delay_ms(1333) ;//delay 1ms
        scan_a=(scan_a<<1)| 0x01;//quét đèn kế tiếp
    }
}

void delay_ms(unsigned int n)// ctc delay ms
{
    unsigned int k;
    for(k=0;k<=n;k++); // Td=6xn MC
}

unsigned char get_7seg(unsigned char x) //hàm chuyển mã 7 đoạn
{
    unsigned char tab_7sa[]={0xc0,0xf9,0xa4,0xb0,0x99,0x92,0x82,0xf8,0x80,0x90,0x88,
                           0x83,0xc6,0xa1,0x86,0x8e}; //bảng tra mã 7 đoạn
    x=tab_7sa[x] ;//lấy mã 7 đoạn
    return(x) ;
}

```

- Hàm BIN8_BCD cho kết quả là số BCD không nén cắt trong các ô nhớ SRAM địa chỉ từ 0x200 đến 0x203.Phần tính toán phép chia sử dụng C đơn giản hơn hợp ngữ rất nhiều!

- Hàm SCAN_4LA không cần phải chuyển sang số BCD không nén.

❖ Câu hỏi ôn tập

- Từ Bảng 6.6,ghi ra bảng mã 7 đoạn cho LED 7 đoạn Cathode chung(CC).Từ Hình 6.20 và ví dụ 6.10,thiết kế mạch MCU324P giao tiếp với LED 7 đoạn CC
- Trong hình 6.21 và ví dụ 6.11,nêu ra phương pháp hiển thị dấu thập phân bằng phần cứng hoặc phần mềm.
- Từ sơ đồ hình 6.22,có thể mở rộng hiển thị tối đa bao nhiêu LED 7 đoạn?Tính dòng trung bình ngõ ra IC74HC573 phải nhận và dòng đỉnh mỗi BJT cấp cho Anode LED trong trường hợp xấu nhất,cho phân cực dòng trung bình qua mỗi đoạn 5mA.
- Với kết quả từ câu 3,IC 74HC573 có thể lái trực tiếp chân đoạn được không?Nếu không,vẽ lại mạch lái chân đoạn phù hợp.
- Từ chương trình con SCAN_4LA,tính thời gian chu kỳ quét các đèn tính từ lúc bắt đầu và kết thúc vòng lặp LOOP ,và thời gian 1 đèn sáng trong chu kỳ quét.

6.4.2 Giao tiếp hiển thị LED ma trận 8x8

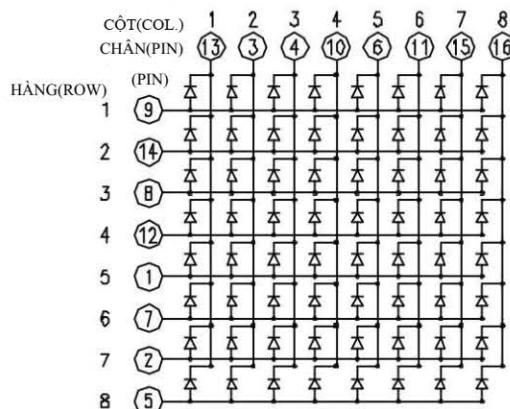
Bộ hiển thị dùng LED 7 đoạn chủ yếu chỉ hiển thị chữ số.Để có thể hiển thị đầy đủ ký tự đồ họa ta dùng bộ hiển thị ma trận LED,gồm các đèn LED ma trận điểm 8x8 ghép lại.

Sau đây ta tìm hiểu tóm tắt cấu tạo,cách phân cực tạo điểm sáng và cách tạo bộ font ký tự đồ họa từ LED ma trận 8x8.

Hình 6.23 mô tả cấu tạo của LED ma trận 8x8 tiêu biểu.Mỗi LED đơn tương đương một điểm sáng,do đó để phân cực điểm sáng ta chỉ cần phân cực thuận LED điểm tương ứng.Ví dụ muốn điểm hàng i cột j sáng,ta đặt hàng i lên mức 1 và cột j mức 0 và phải có điện trở hạn dòng ở hàng i hoặc cột j.Điện áp và dòng điện phân cực mỗi LED điểm 2V/5-10mA.

Để hiển thị LED ma trận, ta phải dùng phương pháp quét tương tự như mục 6.4.1 ở trên. Giả sử ta áp dụng phương pháp xuất data ra hàng và xuất mã quét cột. Tại một thời điểm ta xuất data ra 8 hàng và đặt cột đầu tiên có giá trị 0, thực hiện tương tự như vậy cho 7 cột kế tiếp. Như vậy, chu kỳ quét sáng mỗi cột là 1/8 chu kỳ quét toàn bộ đèn. Với chu kỳ quét không lớn hơn 20ms, mắt người có cảm giác cả đèn sáng liên tục.

➤ *Lưu ý: Đặt hàng và cột là do ta tự quy ước. Ví dụ nếu ta quay hình 6.23 sang trái 90 độ sẽ chuyển hàng thành cột và cột thành hàng!*



Hình 6.23: Cấu tạo LED ma trận 8x8

❖ Phương pháp thiết lập bộ mã font ký tự

HÀNG	\$0	\$7E	\$11	\$11	\$11	\$7E	\$0	\$0
D0								
D1								
D2								
D3								
D4								
D5								
D6								
D7								
CỘT	D0	D1	D2	D3	D4	D5	D6	D7
MÃ CỘT	\$FE	\$FD	\$FB	\$F78	\$EF	\$DF	\$BF	\$7F

Hình 6.24: Thiết lập mã font ký tự A

Hình 6.24 trình bày phương pháp thiết lập bộ mã font ký tự cho LED ma trận 8x8. Ta vẽ một hình vuông 8 hàng 8 cột chia thành 64 ô vuông, tương ứng 64 điểm LED. Ta gán các bit trọng số cho hàng và cột như hình. Tô màu các ô cần sáng và thiết lập data hàng tương ứng từng cột theo quy tắc ô có tô màu là mức 1 và không tô màu là mức 0 (do quy ước hàng mức 1 LED sáng). Ví dụ cột D1 trên hình 6.24 có tô màu hàng D1 đến D6 nên data hàng này là 0b01111110=\$7E. Nếu bit D1 của cột mức 0, 6 điểm ứng với vị trí hàng D1 đến D6 sẽ sáng ở cột D1. Mã quét cột sẽ lần lượt có 1 bit mức 0, các bit còn lại mức 1 nên sẽ có 8 tổ hợp mã cố định \$FE,\$FD,\$FB,\$F7,\$EF,\$DF,\$BF,\$7F

Từ hình 6.24, ta có bộ mã font ký tự A như Bảng 6.7, trong đó bộ mã font chính là data hàng.

Bảng 6.7: Bộ mã font ký tự A theo mã quét cột

Cột	0	1	2	3	4	5	6	7
Mã cột	\$FE	\$FD	\$FB	\$F78	\$EF	\$DF	\$BF	\$7F0
Mã font	\$00	\$7E	\$11	\$11	\$11	\$7E	\$00	\$00

Bảng 6.8 trình bày bộ mã font ký tự mã ASCII nhìn thấy được từ \$20 đến \$7F và mã con trả \$80 theo phương pháp thiết lập ở trên.

Bảng 6.8: Bộ mã font ký tự mã ASCII từ \$20-\$80

0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00	;" "=\$20
0x00,0x00,0x00,0x4F,0x00,0x00,0x00,0x00	;" ! "=\$21
0x00,0x00,0x07,0x00,0x07,0x00,0x00,0x00	;" " "=\$22

0x00,0x14,0x7F,0x14,0x7F,0x14,0x00,0x00	;" # "=\$23
0x00,0x24,0x2A,0x7F,0x2A,0x12,0x00,0x00	;" \$ "=\$24
0x00,0x23,0x13,0x08,0x64,0x62,0x00,0X00	;" % "=\$25
0x00,0x36,0x49,0x55,0x22,0x50,0x00,0x00	;" & "=\$26
0x00,0x00,0x05,0x03,0x00,0x00,0x00,0x00	;" ' "=\$27
0x00,0x00,0x1C,0x22,0x41,0x00,0x00,0x00	;" ("=\$28
0x00,0x00,0x41,0x22,0x1C,0x00,0x00,0x00	;") "=\$29
0x00,0x14,0x08,0x3E,0x08,0x14,0x00,0x00	;" * "=\$2A
0x00,0x08,0x08,0x3E,0x08,0x08,0x00,0x00	;" + "=\$2B
0x00,0x00,0x50,0x30,0x00,0x00,0x00,0x00	;" ; "=\$2C
0x00,0x08,0x08,0x08,0x08,0x00,0x00,0x00	;" - "=\$2D
0x00,0x00,0x60,0x60,0x00,0x00,0x00,0x00	;" . "=\$2E
0x00,0x20,0x10,0x08,0x04,0x02,0x00,0x00	;" / "=\$2F
0x00,0x3E,0x51,0x49,0x45,0x3E,0x00,0x00	;" 0 "=\$30
0x00,0x00,0x42,0x7F,0x40,0x00,0x00,0x00	;" 1 "=\$31
0x00,0x42,0x61,0x51,0x49,0x46,0x00,0x00	;" 2 "=\$32
0x00,0x21,0x41,0x45,0x4B,0x31,0x00,0x00	;" 3 "=\$33
0x00,0x18,0x14,0x12,0x7F,0x10,0x00,0x00	;" 4 "=\$34
0x00,0x27,0x45,0x45,0x45,0x39,0x00,0x00	;" 5 "=\$35
0x00,0x3C,0x4A,0x49,0x49,0x30,0x00,0x00	;" 6 "=\$36
0x00,0x01,0x01,0x79,0x05,0x03,0x00,0x00	;" 7 "=\$37
0x00,0x36,0x49,0x49,0x49,0x36,0x00,0x00	;" 8 "=\$38
0x00,0x06,0x49,0x49,0x29,0x1E,0x00,0x00	;" 9 "=\$39
0x00,0x00,0x36,0x36,0x00,0x00,0x00,0x00	;" : "=\$3A
0x00,0x00,0x56,0x36,0x00,0x00,0x00,0x00	;" ; "=\$3B
0x00,0x08,0x14,0x22,0x41,0x00,0x00,0x00	;" < "=\$3C
0x00,0x14,0x14,0x14,0x14,0x14,0x00,0x00	;" = "=\$3D
0x00,0x00,0x41,0x22,0x14,0x08,0x00,0x00	;" > "=\$3E
0x00,0x02,0x01,0x51,0x09,0x06,0x00,0x00	;" ? "=\$3F
0x00,0x32,0x49,0x79,0x41,0x3E,0x00,0x00	;" @ "=\$40
0x00,0x7E,0x11,0x11,0x11,0x7E,0x00,0x00	;" A "=\$41
0x00,0x41,0x7F,0x49,0x49,0x36,0x00,0x00	;" B "=\$42
0x00,0x3E,0x41,0x41,0x41,0x22,0x00,0x00	;" C "=\$43
0x00,0x41,0x7F,0x41,0x41,0x3E,0x00,0x00	;" D "=\$44
0x00,0x7E,0x49,0x49,0x49,0x49,0x00,0x00	;" E "=\$45
0x00,0x7F,0x09,0x09,0x09,0x01,0x00,0x00	;" F "=\$46
0x00,0x3E,0x41,0x41,0x49,0x7A,0x00,0x00	;" G "=\$47
0x00,0x7F,0x08,0x08,0x08,0x7F,0x00,0x00	;" H "=\$48
0x00,0x00,0x41,0x7F,0x41,0x00,0x00,0x00	;" I "=\$49
0x00,0x20,0x40,0x41,0x3F,0x01,0x00,0x00	;" J "=\$4A
0x00,0x7F,0x08,0x14,0x22,0x41,0x00,0x00	;" K "=\$4B
0x00,0x7F,0x40,0x40,0x40,0x40,0x00,0x00	;" L "=\$4C
0x00,0x7F,0x02,0x0C,0x02,0x7F,0x00,0x00	;" M "=\$4D
0x00,0x7F,0x06,0x08,0x30,0x7F,0x00,0x00	;" N "=\$4E
0x00,0x3E,0x41,0x41,0x41,0x3E,0x00,0x00	;" O "=\$4F
0x00,0x7F,0x09,0x09,0x09,0x06,0x00,0x00	;" P "=\$50
0x00,0x3E,0x41,0x51,0x21,0x5E,0x00,0x00	;" Q "=\$51
0x00,0x7F,0x09,0x19,0x29,0x46,0x00,0x00	;" R "=\$52
0x00,0x26,0x49,0x49,0x49,0x32,0x00,0x00	;" S "=\$53

0x00,0x01,0x01,0x7F,0x01,0x01,0x00,0x00	;" T "=\$54
0x00,0x3F,0x40,0x40,0x40,0x3F,0x00,0x00	;" U "=\$55
0x00,0x1F,0x20,0x40,0x20,0x1F,0x00,0x00	;" V "=\$56
0x00,0x7F,0x20,0x18,0x20,0x7F,0x00,0x00	;" W "=\$57
0x00,0x63,0x14,0x08,0x14,0x63,0x00,0x00	;" X "=\$58
0x00,0x07,0x08,0x70,0x08,0x07,0x00,0x00	;" Y "=\$59
0x00,0x61,0x51,0x49,0x45,0x43,0x00,0x00	;" Z "=\$5A
0x00,0x00,0x7F,0x41,0x41,0x00,0x00,0x00	;" ["=\$5B
0x00,0x02,0x04,0x08,0x10,0x20,0x00,0x00	;" \ "=\$5C
0x00,0x00,0x41,0x41,0x7F,0x00,0x00,0x00	;"] "=\$5D
0x00,0x04,0x02,0x01,0x02,0x04,0x00,0x00	;" ^ "=\$5E
0x00,0x40,0x40,0x40,0x40,0x40,0x00,0x00	;" - "=\$5F
0x00,0x01,0x02,0x04,0x00,0x00,0x00,0x00	;" \ "=\$60
0x00,0x20,0x54,0x54,0x54,0x78,0x00,0x00	;" a "=\$61
0x00,0x7F,0x48,0x44,0x44,0x38,0x00,0x00	;" b "=\$62
0x00,0x38,0x44,0x44,0x44,0x28,0x00,0x00	;" c "=\$63
0x00,0x38,0x44,0x44,0x48,0x7F,0x00,0x00	;" d "=\$64
0x00,0x38,0x54,0x54,0x54,0x18,0x00,0x00	;" e "=\$65
0x00,0x00,0x08,0x7E,0x09,0x02,0x00,0x00	;" f "=\$66
0x00,0xC,0x52,0x52,0x4C,0x3E,0x00,0x00	;" g "=\$67
0x00,0x7F,0x08,0x04,0x04,0x78,0x00,0x00	;" h "=\$68
0x00,0x00,0x44,0x7D,0x40,0x00,0x00,0x00	;" i "=\$69
0x00,0x20,0x40,0x44,0x3D,0x00,0x00,0x00	;" j "=\$6A
0x00,0x00,0x7F,0x10,0x28,0x44,0x00,0x00	;" k "=\$6B
0x00,0x00,0x41,0x7F,0x40,0x00,0x00,0x00	;" l "=\$6C
0x00,0x7C,0x04,0x78,0x04,0x78,0x00,0x00	;" m "=\$6D
0x00,0x7C,0x08,0x04,0x04,0x78,0x00,0x00	;" n "=\$6E
0x00,0x38,0x44,0x44,0x44,0x38,0x00,0x00	;" o "=\$6F
0x00,0x7E,0xC,0x12,0x12,0xC,0x00,0x00	;" p "=\$70
0x00,0xC,0x12,0x12,0xC,0x7E,0x00,0x00	;" q "=\$71
0x00,0x7C,0x08,0x04,0x04,0x08,0x00,0x00	;" r "=\$72
0x00,0x58,0x54,0x54,0x54,0x64,0x00,0x00	;" s "=\$73
0x00,0x04,0x3F,0x44,0x40,0x20,0x00,0x00	;" t "=\$74
0x00,0x3C,0x40,0x40,0x3C,0x40,0x00,0x00	;" u "=\$75
0x00,0x1C,0x20,0x40,0x20,0x1C,0x00,0x00	;" v "=\$76
0x00,0x3C,0x40,0x30,0x40,0x3C,0x00,0x00	;" w "=\$77
0x00,0x44,0x28,0x10,0x28,0x44,0x00,0x00	;" s "=\$78
0x00,0x1C,0xA0,0xA0,0x90,0x7C,0x00,0x00	;" y "=\$79
0x00,0x44,0x64,0x54,0x4C,0x44,0x00,0x00	;" z "=\$7A
0x00,0x00,0x08,0x36,0x41,0x00,0x00,0x00	;" { "=\$7B
0x00,0x00,0x00,0x7F,0x00,0x00,0x00,0x00	;" "=\$7C
0x00,0x00,0x41,0x36,0x08,0x00,0x00,0x00	;" } "=\$7D
0x00,0x02,0x01,0x02,0x04,0x02,0x00,0x00	;" ~ "=\$7E
0x00,0xFF,0xFF,0xFF,0xFF,0x00,0x00,0x00	;" "=\$7F
0x00,0xC0,0xC0,0xC0,0xC0,0xC0,0x00,0x00	; cursor=\$80

Ví dụ 6.13: Thiết kế mạch MCU324P giao tiếp với LED ma trận 8x8. Viết chương trình nhập mã ASCII hiển thị được và hiển thị ký tự mã ASCII ra LED ma trận 8x8.

Giải:

Mạch thiết kế như hình 6.25

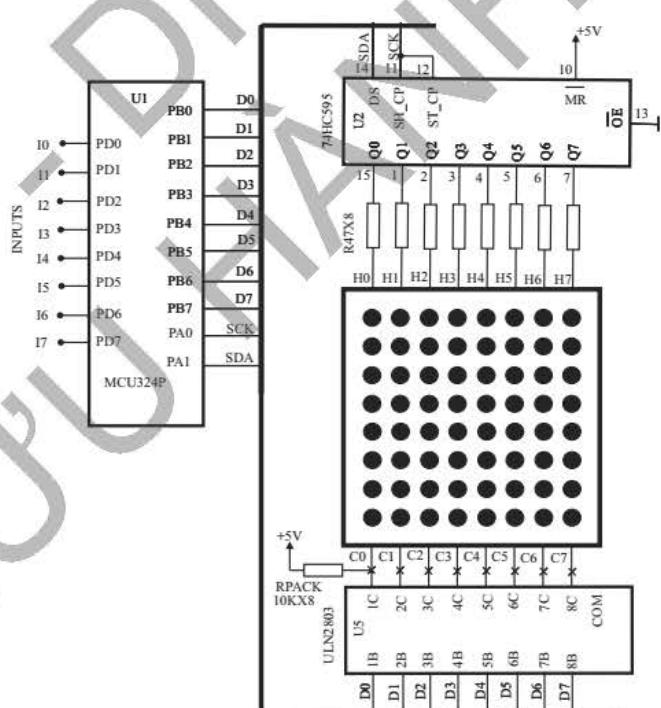
- U1=MCU324P sử dụng PORTB vào/ra giao tiếp ngoại vi,PA0,PA1 giao tiếp thanh ghi dịch
- U2=74HC595 thanh ghi dịch chuyển data hàng từ nối tiếp sang song song. Để đơn giản ta nối chung 2 chân SHCP và STCP cho hoạt động mô thức ghi dịch và xuất data. Data hàng là mã font ký tự xuất nối tiếp từng byte theo cột, SCK=PA0 xung CK dịch và xuất ngõ ra, SDA=PA1 ngõ vào nối tiếp nhận data từ MCU dịch ra MSB trước
- U5=ULN2803 8 cổng đèn đảo dòng cao cực thu hở,lái cột.Ta mắc thêm RPACK 10Kx8 ngõ ra U5 kéo lên nguồn để đảm bảo logic 1 ở cột tắt hắt LED không bị sáng rò!

➤ Lưu ý với mô thức hoạt động ghi dịch và xuất data của 74HC595 như trên, data xuất ở ngõ ra là data ngõ ra tầng ghi dịch trước đó $Q7=Q7S, Q6=Q6S, \dots Q1=Q1S, Q0=Q0S$, nên phải ghi dịch 9 xung CK byte data mới xuất hiện đủ ở các ngõ ra Qi !

❖ Tính toán phân cực LED ma trận (Hình 6.26)

- Theo quy ước hàng tích cực mức 1,cột tích cực mức 0 LED điểm sáng
- Nếu chọn dòng trung bình lái 1 LED điểm=5mA, dòng đỉnh tương ứng=5x8=40mA vượt quá dòng chịu đựng ngõ ra IC 74HC595 Iomax=35mA.Như vậy ta phải thêm mạch đèn nâng dòng không đảo cho ngõ ra(sử dụng BJT mắc CC hay MOSFET mắc CD). Để đơn giản ta hạ dòng trung bình qua LED điểm còn 2.5mA(thông thường phân cực LED siêu sáng tầm 2 đến 3mA DC là đạt). Như vậy trong trường hợp xấu nhất,dòng trung bình ngõ ra U2 bằng dòng đỉnh qua LED điểm = $2.5 \times 8 = 20$ mA <35mA nên U2 có thể lái trực tiếp hàng được.
- Tính trường hợp cực đại 8 LED điểm trên cùng một cột sáng, dòng đỉnh qua cột= $8 \times 20 = 160$ mA nên ngõ ra MCU không thể lái trực tiếp cột được! Do đó,ta sử dụng mạch đèn nâng dòng U5=ULN2803 lái cột(Xem data sheet ở phần phụ lục)
- ULN2803 tích hợp 8 mạch BJT mắc darlington ngõ ra đảo cực thu hở,có các thông số cơ bản:
 - Ngõ vào tương hợp mức TTL
 - Điện áp ngõ ra max 50V
 - Dòng ngõ ra max 500mA

➤ Do mã quét cột xuất từ MCU phải qua cổng đảo U5 lái cột nên mã quét cột ở Bảng 6.7 phải đảo lại!

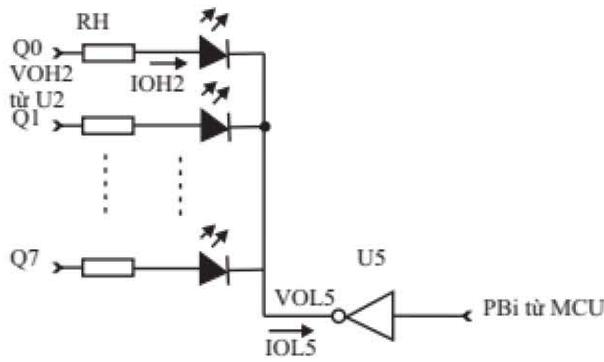


Hình 6.25: Sơ đồ thiết kế ví dụ 6.13

- Để đảm bảo điểm sáng đều, ta phân cực điện trở hạn dòng trên hàng như hình 6.26
- Tính các giá trị điện trở hạn dòng:

$$RH = (VOH2min - VD - VOL5max) / IOH2(\text{đỉnh}) \quad (6.10)$$

$$RH = (4 - 2 - 1) / 20 = 50\Omega \rightarrow \text{Chọn } RH = 47\Omega$$



Hình 6.26: Phân cực cho các LED điểm và khả năng tách nhánh của U5

- ❖ Ý tưởng giải thuật tạo mã font ký tự từ mã ASCII nhập vào
 - Dùng phương pháp tra bảng(lookup table), đặt bảng mã font ký tự theo Bảng 6.8 trong bộ nhớ chương trình(Flash ROM) tại địa chỉ bắt đầu FONT_TAB(0x200)
 - Tám byte data đầu tiên bắt đầu tại địa chỉ ký hiệu TAB_CHR=FONT_TAB chính là mã font ký tự dấu cách(space bar) có mã ASCII=\$20,tám byte tiếp theo ứng với “!”=\$21,cứ thế tiếp tục...
 - Như vậy tính từ địa chỉ nền TAB_CHR,mã font ký tự \$20 bắt đầu từ địa chỉ offset=0,mã font ký tự \$21 bắt đầu từ địa chỉ offset= 8,mã font ký tự \$22 bắt đầu từ địa chỉ offset=16...
 - Do một địa chỉ của bộ nhớ chương trình gồm 2 byte,nên mỗi bộ mã font ký tự cách nhau 4 địa chỉ. Từ đó ta suy ra công thức tổng quát xác định địa chỉ offset hay địa chỉ bắt đầu bộ mã font ký tự tương ứng mã ASCII cần tra ký hiệu ASC:

$$\text{Offset} = (\text{ASC}-\$20) \times 4 \quad (6.11)$$

$$\text{Địa chỉ bắt đầu bộ mã font ký tự} = \text{TAB_CHR} + \text{offset} \quad (6.12)$$

➤ Để truy xuất địa chỉ flash ROM ta phải dịch trái 1 bước địa chỉ tính được ở (6.12)

❖ Chương trình hợp ngữ ví dụ 6.13

Để tiện cho các ví dụ sau,trước tiên ta thiết lập file thư viện **FONT_CHR.INC** chứa bộ mã font ký tự đã thiết lập như Bảng 6.8.File **FONT_CHR.INC** được soạn thảo bằng bất kỳ phần mềm soạn thảo văn bản nào(ví dụ Notepad trong Windows),và được lưu trong cùng thư mục với file chương trình nguồn để trình biên dịch dò tìm được khi liên kết(không cần phải ghi cụ thể đường dẫn đến thư mục chứa file)

- Trong file **FONT_CHR.INC** có khai báo ký hiệu FONT_TAB là địa chỉ bắt đầu đặt bộ mã font ký tự,và ký hiệu TAB_CHR trả địa chỉ bắt đầu.
- Trong chương trình nguồn sử dụng chỉ dẫn INCLUDE gọi file thư viện này:

.INCLUDE "FONT_CHR.INC"; Sử dụng dấu "" theo quy định cú pháp

➤ Phải gán giá trị cho FONT_TAB trước khi gọi file thư viện font ký tự trên

```
;;
;FONT_CHR.INC chứa bộ mã font ký tự
;FONT_TAB=địa chỉ bắt đầu trong bộ nhớ chương trình
;TAB_CHR= trả địa chỉ nền đầu tiên của vùng nhớ mã font ký tự
;;
        .ORG  FONT_TAB          ;đặt địa chỉ đầu bảng tra
        TAB_CHR:                   ;ký hiệu nhãn địa chỉ đầu bảng tra
        .DB   0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00  ;," "=00H(offset quy đổi)
        .DB   0x00,0x00,0x00,0x4F,0x00,0x00,0x00,0x00  ;," ! "=01H
        .DB   0x00,0x00,0x07,0x00,0x07,0x00,0x00,0x00  ;," " "=02H
        .DB   0x00,0x14,0x7F,0x14,0x7F,0x14,0x00,0x00  ;," # "=03H
        .DB   0x00,0x24,0x2A,0x7F,0x2A,0x12,0x00,0x00  ;," $ "=04H
        .DB   0x00,0x23,0x13,0x08,0x64,0x62,0x00,0X00  ;," % "=05H
        .DB   0x00,0x36,0x49,0x55,0x22,0x50,0x00,0x00  ;," & "=06H
        .DB   0x00,0x00,0x05,0x03,0x00,0x00,0x00,0x00  ;," ' "=07H
        .DB   0x00,0x00,0x1C,0x22,0x41,0x00,0x00,0x00  ;," ( "=08H
        .DB   0x00,0x00,0x41,0x22,0x1C,0x00,0x00,0x00  ;," ) "=09H
        .DB   0x00,0x14,0x08,0x3E,0x08,0x14,0x00,0x00  ;," * "=0AH
        .DB   0x00,0x08,0x08,0x3E,0x08,0x08,0x00,0x00  ;," + "=0BH
```

.DB	0x00,0x00,0x50,0x30,0x00,0x00,0x00,0x00	,” ; “=0CH
.DB	0x00,0x08,0x08,0x08,0x08,0x00,0x00,0x00	,” - “=0DH
.DB	0x00,0x00,0x60,0x60,0x00,0x00,0x00,0x00	,” . “=0EH
.DB	0x00,0x20,0x10,0x08,0x04,0x02,0x00,0x00	,” / “=0FH
.DB	0x00,0x3E,0x51,0x49,0x45,0x3E,0x00,0x00	,” 0 “=10H
.DB	0x00,0x00,0x42,0x7F,0x40,0x00,0x00,0x00	,” 1 “=11H
.DB	0x00,0x42,0x61,0x51,0x49,0x46,0x00,0x00	,” 2 “=12H
.DB	0x00,0x21,0x41,0x45,0x4B,0x31,0x00,0x00	,” 3 “=13H
.DB	0x00,0x18,0x14,0x12,0x7F,0x10,0x00,0x00	,” 4 “=14H
.DB	0x00,0x27,0x45,0x45,0x45,0x39,0x00,0x00	,” 5 “=15H
.DB	0x00,0x3C,0x4A,0x49,0x49,0x30,0x00,0x00	,” 6 “=16H
.DB	0x00,0x01,0x01,0x79,0x05,0x03,0x00,0x00	,” 7 “=17H
.DB	0x00,0x36,0x49,0x49,0x49,0x36,0x00,0x00	,” 8 “=18H
.DB	0x00,0x06,0x49,0x49,0x29,0x1E,0x00,0x00	,” 9 “=19H
.DB	0x00,0x00,0x36,0x36,0x00,0x00,0x00,0x00	,” : “=1AH
.DB	0x00,0x00,0x56,0x36,0x00,0x00,0x00,0x00	,” ; “=1BH
.DB	0x00,0x08,0x14,0x22,0x41,0x00,0x00,0x00	,” < “=1CH
.DB	0x00,0x14,0x14,0x14,0x14,0x00,0x00,0x00	,” = “=1DH
.DB	0x00,0x00,0x41,0x22,0x14,0x08,0x00,0x00	,” > “=1EH
.DB	0x00,0x02,0x01,0x51,0x09,0x06,0x00,0x00	,” ? “=1FH
.DB	0x00,0x32,0x49,0x79,0x41,0x3E,0x00,0x00	,” @ ”=20H
.DB	0x00,0x7E,0x11,0x11,0x11,0x7E,0x00,0x00	,” A “=21H
.DB	0x00,0x41,0x7F,0x49,0x49,0x36,0x00,0x00	,” B “=22H
.DB	0x00,0x3E,0x41,0x41,0x41,0x22,0x00,0x00	,” C “=23H
.DB	0x00,0x41,0x7F,0x41,0x41,0x3E,0x00,0x00	,” D ”=24H
.DB	0x00,0x7E,0x49,0x49,0x49,0x49,0x00,0x00	,” E “=25H
.DB	0x00,0x7F,0x09,0x09,0x09,0x01,0x00,0x00	,” F “=26H
.DB	0x00,0x3E,0x41,0x41,0x49,0x7A,0x00,0x00	,” G “=27H
.DB	0x00,0x7F,0x08,0x08,0x08,0x7F,0x00,0x00	,” H “=28H
.DB	0x00,0x00,0x41,0x7F,0x41,0x00,0x00,0x00	,” I “=29H
.DB	0x00,0x20,0x40,0x41,0x3F,0x01,0x00,0x00	,” J “=2AH
.DB	0x00,0x7F,0x08,0x14,0x22,0x41,0x00,0x00	,” K “=2BH
.DB	0x00,0x7F,0x40,0x40,0x40,0x40,0x00,0x00	,” L “=2CH
.DB	0x00,0x7F,0x02,0x0C,0x02,0x7F,0x00,0x00	,” M “=2DH
.DB	0x00,0x7F,0x06,0x08,0x30,0x7F,0x00,0x00	,” N “=2EH
.DB	0x00,0x3E,0x41,0x41,0x41,0x3E,0x00,0x00	,” O “=2FH
.DB	0x00,0x7F,0x09,0x09,0x09,0x06,0x00,0x00	,” P “=30H
.DB	0x00,0x3E,0x41,0x51,0x21,0x5E,0x00,0x00	,” Q ”=31H
.DB	0x00,0x7F,0x09,0x19,0x29,0x46,0x00,0x00	,” R “=32H
.DB	0x00,0x26,0x49,0x49,0x49,0x32,0x00,0x00	,” S “=33H
.DB	0x00,0x01,0x01,0x7F,0x01,0x01,0x00,0x00	,” T “=34H
.DB	0x00,0x3F,0x40,0x40,0x40,0x3F,0x00,0x00	,” U “=35H
.DB	0x00,0x1F,0x20,0x40,0x20,0x1F,0x00,0x00	,” V “=36H
.DB	0x00,0x7F,0x20,0x18,0x20,0x7F,0x00,0x00	,” W “=37H
.DB	0x00,0x63,0x14,0x08,0x14,0x63,0x00,0x00	,” X “=38H
.DB	0x00,0x07,0x08,0x70,0x08,0x07,0x00,0x00	,” Y “=39H
.DB	0x00,0x61,0x51,0x49,0x45,0x43,0x00,0x00	,” Z “=3AH
.DB	0x00,0x00,0x7F,0x41,0x41,0x00,0x00,0x00	,” [“=3BH
.DB	0x00,0x02,0x04,0x08,0x10,0x20,0x00,0x00	,” \ “=3CH
.DB	0x00,0x00,0x41,0x41,0x7F,0x00,0x00,0x00	,”] “=3DH
.DB	0x00,0x04,0x02,0x01,0x02,0x04,0x00,0x00	,” ^ “=3EH
.DB	0x00,0x40,0x40,0x40,0x40,0x40,0x00,0x00	,” _ “=3FH
.DB	0x00,0x01,0x02,0x04,0x00,0x00,0x00,0x00	,” \ “=40H
.DB	0x00,0x20,0x54,0x54,0x54,0x78,0x00,0x00	,” a ”=41H
.DB	0x00,0x7F,0x48,0x44,0x44,0x38,0x00,0x00	,” b “=42H

```

.DB 0x00,0x38,0x44,0x44,0x44,0x28,0x00,0x00 ;" c "=43H
.DB 0x00,0x38,0x44,0x44,0x48,0x7F,0x00,0x00 ;" d "=44H
.DB 0x00,0x38,0x54,0x54,0x54,0x18,0x00,0x00 ;" e "=45H
.DB 0x00,0x00,0x08,0x7E,0x09,0x02,0x00,0x00 ;" f "=46H
.DB 0x00,0x0C,0x52,0x52,0x4C,0x3E,0x00,0x00 ;" g "=47H
.DB 0x00,0x7F,0x08,0x04,0x04,0x78,0x00,0x00 ;" h "=48H
.DB 0x00,0x00,0x44,0x7D,0x40,0x00,0x00,0x00 ;" i "=49H
.DB 0x00,0x20,0x40,0x44,0x3D,0x00,0x00,0x00 ;" j "=4AH
.DB 0x00,0x00,0x7F,0x10,0x28,0x44,0x00,0x00 ;" k "=4BH
.DB 0x00,0x00,0x41,0x7F,0x40,0x00,0x00,0x00 ;" l "=4CH
.DB 0x00,0x7C,0x04,0x78,0x04,0x78,0x00,0x00 ;" m "=4DH
.DB 0x00,0x7C,0x08,0x04,0x04,0x78,0x00,0x00 ;" n "=4EH
.DB 0x00,0x38,0x44,0x44,0x44,0x38,0x00,0x00 ;" o "=4FH
.DB 0x00,0x7E,0x0C,0x12,0x12,0x0C,0x00,0x00 ;" p "=50H
.DB 0x00,0x0C,0x12,0x12,0x0C,0x7E,0x00,0x00 ;" q "=51H
.DB 0x00,0x7C,0x08,0x04,0x04,0x08,0x00,0x00 ;" r "=52H
.DB 0x00,0x58,0x54,0x54,0x54,0x64,0x00,0x00 ;" s "=53H
.DB 0x00,0x04,0x3F,0x44,0x40,0x20,0x00,0x00 ;" t "=54H
.DB 0x00,0x3C,0x40,0x40,0x3C,0x40,0x00,0x00 ;" u "=55H
.DB 0x00,0x1C,0x20,0x40,0x20,0x1C,0x00,0x00 ;" v "=56H
.DB 0x00,0x3C,0x40,0x30,0x40,0x3C,0x00,0x00 ;" w "=57H
.DB 0x00,0x44,0x28,0x10,0x28,0x44,0x00,0x00 ;" x "=58H
.DB 0x00,0x1C,0xA0,0xA0,0x90,0x7C,0x00,0x00 ;" y "=59H
.DB 0x00,0x44,0x64,0x54,0x4C,0x44,0x00,0x00 ;" z "=5AH
.DB 0x00,0x00,0x08,0x36,0x41,0x00,0x00,0x00 ;" { "=5BH
.DB 0x00,0x00,0x00,0x7F,0x00,0x00,0x00,0x00 ;" | "=5CH
.DB 0x00,0x00,0x41,0x36,0x08,0x00,0x00,0x00 ;" } "=5DH
.DB 0x00,0x02,0x01,0x02,0x04,0x02,0x00,0x00 ;" ~ "=5EH
.DB 0xFF,0xFF,0xFF,0xFF,0x00,0x00,0x00,0x00 ;" " =5FH
.DB 0xC0,0xC0,0xC0,0xC0,0xC0,0xC0,0xC0,0xC0 ; cursor=60H

```

Bắt đầu chương trình:

```

INCLUDE <M324PDEF.INC>
.EQU FONT_TAB=0X200 ;đặt địa chỉ đầu bảng tra mã font ký tự trong Flash ROM
.INCLUDE "FONT_CHR.INC"; gọi file mã font ký tự
.EQU OUTPORT=PORTB
.EQU IOSETB=DDRB
.EQU INPORT=PORTD
.EQU INPORT_DR=DDRD
.EQU DAT_IN=PIND
.EQU SCK=0
.EQU SDA=1
.EQU SHIFT=PORTA
.EQU SHIFT_DR=DDRA
.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 SHIFT_DR,R16 ;khai báo PA0,PA1 là output
      CBI SHIFT,SCK ;SCK=0

```

	CBI	SHIFT,SDA	;SDA=0
	LDI	R16,0XFF	
	OUT	IOSETB,R16	;PORTB=output
	LDI	R16,0	
	OUT	OUTPORT,R16	;tắt toàn bộ đèn
	OUT	INPORT_DR,R16	;PortD input
	LDI	R16,0XFF	
	OUT	INPORT,R16	
START:	IN	R17,DAT_IN	;nhập data từ input
	LDI	ZH,HIGH(TAB_CHR)	;Z trỏ địa chỉ đầu bảng tra bộ mã font ký tự
	LDI	ZL,LOW(TAB_CHR)	
	SUBI	R17,\$20	;lấy offset bằng cách trừ \$20
	LDI	R16,4	;và nhân cho 4
	MUL	R17,R16	
	ADD	R30,R0	;cộng offset vào địa chỉ nền đầu bảng tra
	ADC	R31,R1	
	CLC		;chuẩn bị quay qua C
	ROL	R30	;quay trái ZL qua C
	ROL	R31	;quay trái ZH qua C tạo địa chỉ nền truy xuất bộ nhớ CT
	LDI	R19,8	;đếm 8 lần quét
	LDI	R18,0X01	;mã quét cột 0
LOOP:	CLR	R16	;
	OUT	OUTPORT,R16	;xóa toàn bộ các cột
	LPM	R17,Z+	;lấy mã font ký tự, tăng địa chỉ con trỏ
	RCALL	SHO_8	;xuất mã font ký tự
	OUT	OUTPORT,R18	;xuất mã quét cột
	RCALL	DELAY_1MS	;tạo trễ 1ms
	CLC		;xóa C chuẩn bị quay
	ROL	R18	;quay trái tạo mã quét cột kế tiếp
	DEC	R19	;đếm số lần quét cột
	BRNE	LOOP	;thoát khi đủ 8 lần quét
	RJMP	START	

;

;SHO_8 dịch trái 8 bit data cất trong R17 ra ngõ SDA

;Input: R17

;Output SDA nối tiếp MSB trước

;Phải dịch và xuất 9 lần mới đúng vị trí hàng(xem lại hình 6.18)

;

SHO_8:	LDI	R16,9	;đếm 9 lần dịch
SH_LOOP:	ROL	R17	;quay trái qua C byte thấp C=b7,b0=C
	BRCC	BIT_0	;C=0 nhảy đến BIT_0
	SBI	SHIFT,SDA	;dịch bit 7=1 ra SDA
	RJMP	NEXT	;nhảy đến NEXT
BIT_0:	CBI	SHIFT,SDA	;dịch bit 7=0 ra SDA
NEXT:	SBI	SHIFT,SCK	;tạo xung CK
	CBI	SHIFT,SCK	
	DEC	R16	;đếm số lần dịch
	BRNE	SH_LOOP	;dịch đủ 9 lần
	RET		

DELAY_1MS:	LDI	R16,8	;1MC sử dụng R16
	MOV	R15,R16	;1MC nạp data cho R15
	LDI	R16,250	;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

❖ Chương trình C VD6-13

Để tiện áp dụng cho những ví dụ sau, trước tiên ta tạo thư viện tên **font_chr.h** chứa bộ mã font ký tự đã thiết lập như trên.

- File thư viện **font_chr.h** phải đặt cùng thư mục file chương trình nguồn, trình biên dịch mới liên kết được. Trong chương trình nguồn ta sử dụng chỉ dẫn include gọi hàm thư viện này:

```
#include "font_chr.h" //Sử dụng dấu “ “ theo quy định cú pháp!
```

//Hàm thư viện file chr.h

0x00,0x7F,0x09,0x09,0x09,0x01,0x00,0x00 //”F “=26H
0x00,0x3E,0x41,0x41,0x49,0x7A,0x00,0x00 //” G “=27H
0x00,0x7F,0x08,0x08,0x08,0x7F,0x00,0x00 //” H “=28H
0x00,0x00,0x41,0x7F,0x41,0x00,0x00,0x00 //” I “=29H
0x00,0x20,0x40,0x41,0x3F,0x01,0x00,0x00 //” J “=2AH
0x00,0x7F,0x08,0x14,0x22,0x41,0x00,0x00 //” K “=2BH
0x00,0x7F,0x40,0x40,0x40,0x40,0x00,0x00 //” L “=2CH
0x00,0x7F,0x02,0x0C,0x02,0x7F,0x00,0x00 //” M “=2DH
0x00,0x7F,0x06,0x08,0x30,0x7F,0x00,0x00 //” N “=2EH
0x00,0x3E,0x41,0x41,0x41,0x3E,0x00,0x00 //” O “=2FH
0x00,0x7F,0x09,0x09,0x09,0x06,0x00,0x00 //” P “=30H
0x00,0x3E,0x41,0x51,0x21,0x5E,0x00,0x00 //” Q “=31H
0x00,0x7F,0x09,0x19,0x29,0x46,0x00,0x00 //” R “=32H
0x00,0x26,0x49,0x49,0x49,0x32,0x00,0x00 //” S “=33H
0x00,0x01,0x01,0x7F,0x01,0x01,0x00,0x00 //” T “=34H
0x00,0x3F,0x40,0x40,0x40,0x3F,0x00,0x00 //” U “=35H
0x00,0x1F,0x20,0x40,0x20,0x1F,0x00,0x00 //” V “=36H
0x00,0x7F,0x20,0x18,0x20,0x7F,0x00,0x00 //” W “=37H
0x00,0x63,0x14,0x08,0x14,0x63,0x00,0x00 //” X “=38H
0x00,0x07,0x08,0x70,0x08,0x07,0x00,0x00 //” Y “=39H
0x00,0x61,0x51,0x49,0x45,0x43,0x00,0x00 //” Z “=3AH
0x00,0x00,0x7F,0x41,0x41,0x00,0x00,0x00 //” [“=3BH
0x00,0x02,0x04,0x08,0x10,0x20,0x00,0x00 //” \“=3CH
0x00,0x00,0x41,0x41,0x7F,0x00,0x00,0x00 //”] “=3DH
0x00,0x04,0x02,0x01,0x02,0x04,0x00,0x00 //” ^ “=3EH
0x00,0x40,0x40,0x40,0x40,0x40,0x00,0x00 //” - “=3FH
0x00,0x01,0x02,0x04,0x00,0x00,0x00,0x00 //” \“=40H
0x00,0x20,0x54,0x54,0x54,0x78,0x00,0x00 //” a “=41H
0x00,0x7F,0x48,0x44,0x44,0x38,0x00,0x00 //” b “=42H
0x00,0x38,0x44,0x44,0x44,0x28,0x00,0x00 //” c “=43H
0x00,0x38,0x44,0x44,0x48,0x7F,0x00,0x00 //” d “=44H
0x00,0x38,0x54,0x54,0x54,0x18,0x00,0x00 //” e “=45H
0x00,0x00,0x08,0x7E,0x09,0x02,0x00,0x00 //” f “=46H
0x00,0x0C,0x52,0x52,0x4C,0x3E,0x00,0x00 //” g “=47H
0x00,0x7F,0x08,0x04,0x04,0x78,0x00,0x00 //” h “=48H
0x00,0x00,0x44,0x7D,0x40,0x00,0x00,0x00 //” i “=49H
0x00,0x20,0x40,0x44,0x3D,0x00,0x00,0x00 //” j “=4AH
0x00,0x00,0x7F,0x10,0x28,0x44,0x00,0x00 //” k “=4BH
0x00,0x00,0x41,0x7F,0x40,0x00,0x00,0x00 //” l “=4CH
0x00,0x7C,0x04,0x78,0x04,0x78,0x00,0x00 //” m “=4DH
0x00,0x7C,0x08,0x04,0x04,0x04,0x78,0x00,0x00 //” n “=4EH
0x00,0x38,0x44,0x44,0x44,0x38,0x00,0x00 //” o “=4FH
0x00,0x7E,0x0C,0x12,0x12,0x0C,0x00,0x00 //” p “=50H
0x00,0x0C,0x12,0x12,0x0C,0x7E,0x00,0x00 //” q “=51H
0x00,0x7C,0x08,0x04,0x04,0x08,0x00,0x00 //” r “=52H
0x00,0x58,0x54,0x54,0x54,0x64,0x00,0x00 //” s “=53H
0x00,0x04,0x3F,0x44,0x40,0x20,0x00,0x00 //” t “=54H
0x00,0x3C,0x40,0x40,0x3C,0x40,0x00,0x00 //” u “=55H
0x00,0x1C,0x20,0x40,0x20,0x1C,0x00,0x00 //” v “=56H
0x00,0x3C,0x40,0x30,0x40,0x3C,0x00,0x00 //” w “=57H
0x00,0x44,0x28,0x10,0x28,0x44,0x00,0x00 //” s “=58H
0x00,0x1C,0xA0,0xA0,0x90,0x7C,0x00,0x00 //” y “=59H

```

0x00,0x44,0x64,0x54,0x4C,0x44,0x00,0x00 //;" z "=5AH
0x00,0x00,0x08,0x36,0x41,0x00,0x00,0x00 //;" { "=5BH
0x00,0x00,0x00,0x7F,0x00,0x00,0x00,0x00 //;" | "=5CH
0x00,0x00,0x41,0x36,0x08,0x00,0x00,0x00 //;" } "=5DH
0x00,0x02,0x01,0x02,0x04,0x02,0x00,0x00 //;" ~ "=5EH
0x00,0xFF,0xFF,0xFF,0xFF,0xFF,0x00,0x00 //;" "=5FH
0x00,0xC0,0xC0,0xC0,0xC0,0xC0,0x00,0x00 // cursor=60H
};//kết thúc bảng tra bộ mã font ký tự

```

Chương trình C bắt đầu từ đây

```

#include<avr/io.h>
#include "font_chr.h"//Gọi hàm thư viện font ký tự
#define dat_in PIND //định nghĩa dat_in=PIND
#define import PORTD //định nghĩa import=PORTD
#define import_dr DDRD //định nghĩa import_dr=DDRD
#define outport PORTB //định nghĩa outport=PORTB
#define iosetb DDRB //định nghĩa iosetb=định hướng PORTB
#define shift PORTA //định nghĩa shift=PORT
#define shift_dr DDRA //định nghĩa shift_dr=DDRA
const char SCK=0,SDA=1;//ký hiệu vị trí bit SCK,SDA
void SHO_8(unsigned char dat) ;//khai báo hàm dịch 6 bit
void delay_ms(unsigned int n) ;//khai báo hàm delay 1ms
unsigned char i,tam,col ;
unsigned int offset ;
int main()
{
    shift_dr=0x03 ;//định nghĩa SCK,SDA là output
    shift=0xfc ;//SCK=0,SDA=0
    import_dr=0x00 ;//import=input
    import=0xff ;//điện trở kéo lên import
    iosetb=0xff ;//PortB output
    outport=0x00 ;//xóa đèn
    while(1)
    {
        tam=dat_in ;//nhập data từ import
        offset=(tam-0x20)*8;//lấy offset
        col=0x01 ;//mã quét cột 0
        for(i=0;i<=7;i++)
        {
            outport=0x00 ;//xóa toàn bộ đèn
            tam=TAB_CHR[offset+i];//lấy mã font ký tự
            SHO_8(tam) ;//dịch mã font ký tự ra U2
            outport=col ;//xuất mã quét cột
            delay_ms(1333) ;//trễ 1ms
            col=col<<1 ;//tạo mã quét kế tiếp
        }
    }
}
//-----
void SHO_8(unsigned char dat)
{
    unsigned char temp,j;
    temp=0x80 ;//bắt đầu xét bit 7

```

```

for(j=1;j<=9;j++) //lặp vòng 9 lần(xem lại hình 6.18)
{
    if((dat&temp)==temp) //bit xét=1?
        shift=shift|(1<<SDA); //bit xét=1,SDA=1
    else
        shift=shift&(~(1<<SDA));//SDA=0
    shift=shift|(1<<SCK); //xuất xung dịch CK
    shift=shift&(~(1<<SCK));
    temp=temp>>1; //xét bit kế tiếp
}
}

//-----
void delay_ms(unsigned int n)// ctc delay ms
{
    unsigned int k;
    for(k=0;k<=n;k++); // Td=6xn MC
}

```

- Ván đè truy xuất bằng tra bộ mã font ký tự tương tự như giải thích ở ví dụ 6.10,trình biên dịch C mặc định kiểu bộ nhớ truy xuất là data(SRAM)và khai báo dãy là char tính bằng byte,nên tính offset cách nhau 8 byte!
- Thuận lợi của chương trình C là truy xuất bộ mã font ký tự đơn giản,nhưng phải tốn bộ nhớ SRAM làm vùng đệm chứa data và không xác định được địa chỉ đầu vùng đệm cần truy xuất!

Trường hợp cần mở rộng bộ hiển thị LED ma trận,ta thực hiện mở rộng theo hàng và giữ nguyên cột.

Để tiết kiệm chân port trong trường hợp mở rộng nhiều hàng,ta vẫn áp dụng giao tiếp hàng băng thanh ghi dịch như ví dụ 6.13.Tương ứng với mỗi bộ hiển thị LED ma trận 8x8,có một IC ghi dịch vào nối tiếp ra song song 8 ngõ lái hàng,chỉ cần 2 tín hiệu điều khiển CK và data in từ MCU324P.

Ví dụ 6.14: Thiết kế mạch MCU324P lái bộ hiển thị gồm 2 LED ma trận 8x8,sử dụng thanh ghi dịch lái hàng, và một SW chọn mode hiển thị.Viết chương trình thực hiện các mode hiển thị như sau:

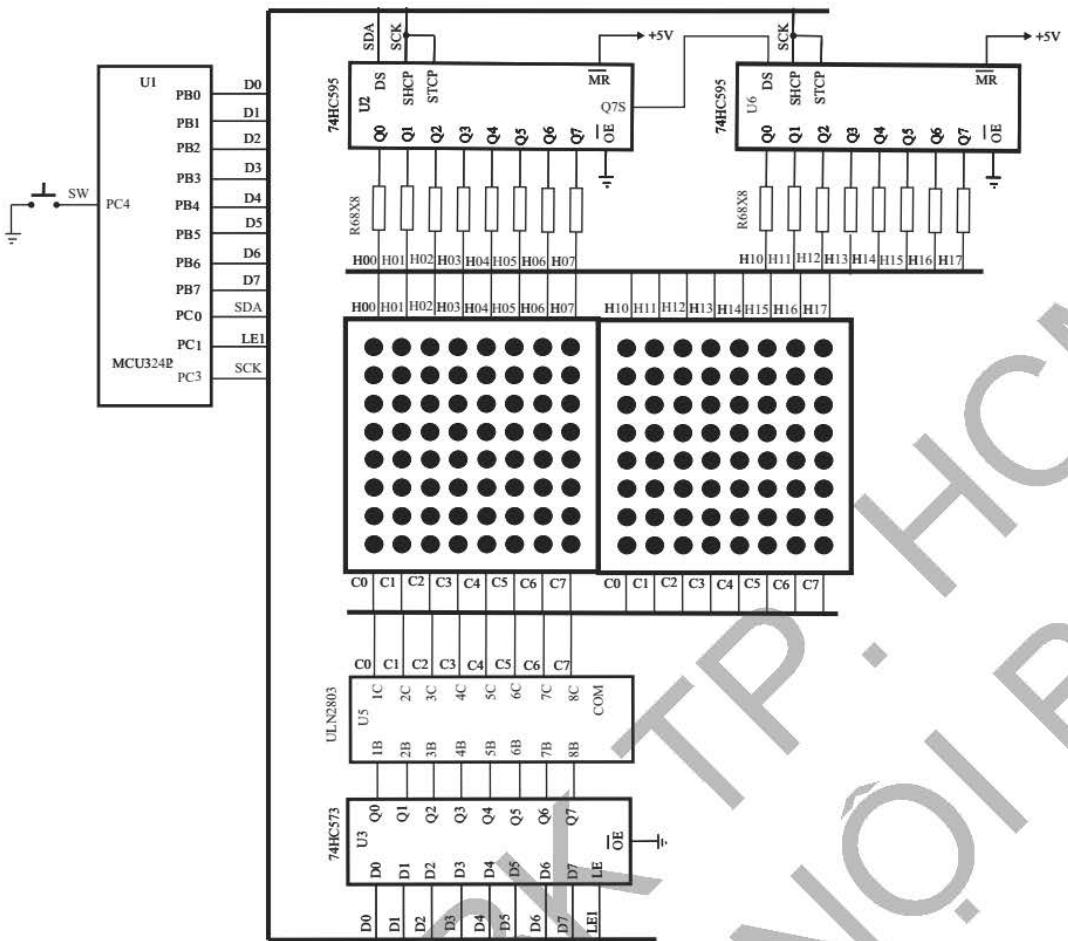
- SW=1: hiển thị 2 ký tự AB đứng yên
- SW=0: hiển thị chạy chữ 2 ký tự AB từ trái sang phải,ký tự B xuất hiện trước

Giải:

- Từ sơ đồ hình 6.25 ta thêm IC ghi dịch U6=74HC595 lái hàng của bộ hiển thị LED ma trận 8x8 thứ hai.Ta có sơ đồ thiết kế như hình 6.27.
- U2 là thanh ghi dịch xuất mã hàng bộ hiển thị đầu 8 bit thấp.MCU324P xuất data và xung CK cho U2 qua ngõ SDA=PC0 và SCK=PC3.U6 kết nối giống như U2,lấy data vào từ Q7S U2(ngõ ra nối tiếp U2) và chung xung CK với U2.
- Để có thể mở rộng sử dụng PortB,ta thêm U3 IC74HC573 chốt mã quét cột lái U5,điều khiển mở chốt bằng LE1=PC1
- Dòng qua hàng vẫn không đổi,kết nối các cột cùng trọng số chung nên dòng qua cột tăng gấp đôi bằng $160 \times 2 = 320$ mA.IC U5 ULN2803 có dòng IOLmax=500mA vẫn chịu được.Trong trường hợp số cột kết nối chung nhiều hơn,ta phải mắc thêm IC ULN2803 chia dòng tải,hoặc thay bằng các BJT hoặc MOSFET công suất cao chịu dòng tải lớn.

❖ **Ý tưởng giải thuật chương trình**

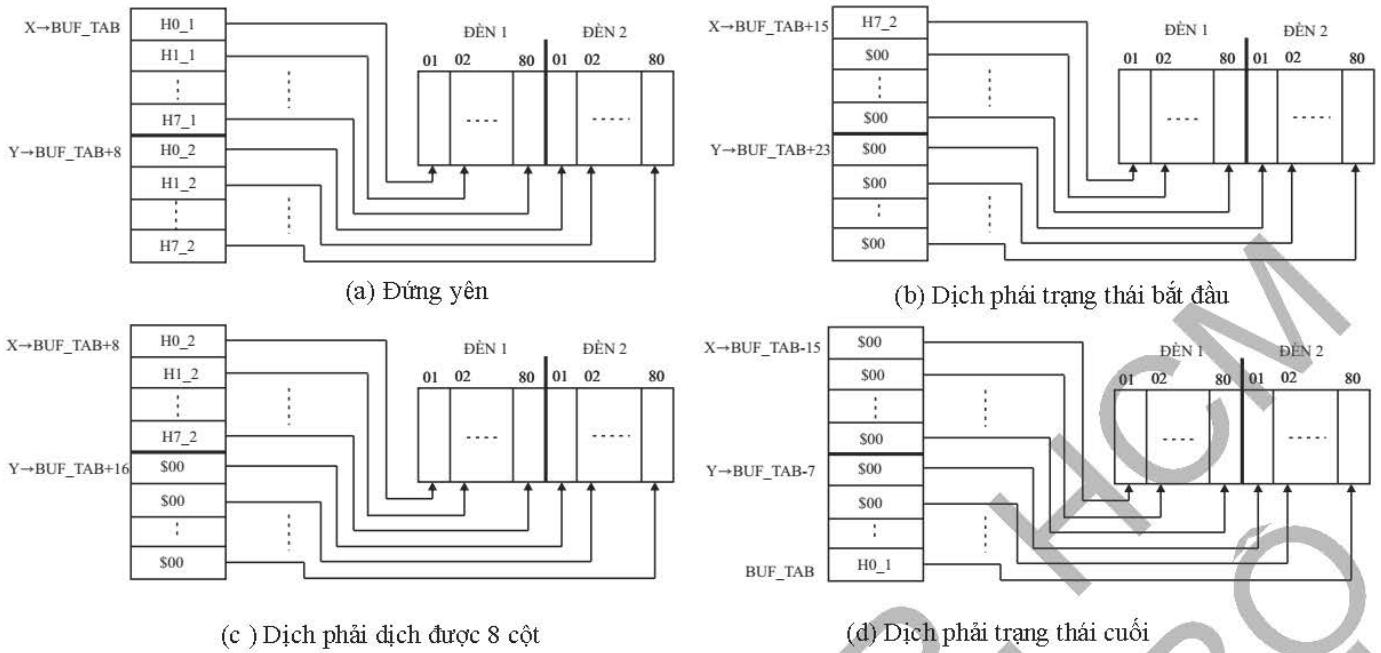
- Tổng quát ta gán các ký hiệu port,các bit điều khiển và các giá trị hằng để dễ theo dõi,kiểm tra và thay đổi thông số.
- Đặt UNI_COL=8 : số cột quét trong 1 đèn hay bằng số byte mã hàng 1 ký tự
- Đặt TOT_COL=16: tổng số cột quét trong toàn bộ đèn
- Đầu chương trình chuyển mã font các ký tự cần hiển thị từ Flash ROM sang SRAM
- Việc hiển thị các mode dựa vào địa chỉ truy xuất SRAM



Hình 6.27: Sơ đồ thiết kế ví dụ 6.14

Hình 6.28 minh họa các mô thức hiển thị và trạng thái quét tương ứng

- Do kết nối quét song song 2 cột cùng trọng số nên phải có 2 con trỏ địa chỉ truy xuất hiển thị 2 byte của 2 ký tự cùng trọng số tương ứng, ghép thành 16 bit xuất nối tiếp ra thanh ghi dịch
 - Hình 6.28a hiển thị mô thức 2 ký tự đứng yên, truy xuất các mã hàng của 2 ký tự và xuất ra đèn theo các vị trí cột tích cực mức 0 tương ứng
 - Hình 6.28b hiển thị mô thức dịch phải ký tự, bắt đầu hiển thị mã hàng byte thứ 8 ký tự 2 ở cột đầu tiên, các cột còn lại tối
 - Hình 6.28c hiển thị mô thức dịch phải ký tự, dịch được 8 cột, đèn 1 hiển thị trọn ký tự 2, đèn 2 tối
 - Dịch phải tiếp 8 cột nữa, hiển thị trọn 2 ký tự như hình 6.28a
 - Hình 6.28d hiển thị mô thức dịch phải ký tự, hiển thị trạng thái cuối, cột cuối hiển thị mã hàng byte đầu của ký tự 1. Sau đó lặp vòng lại từ đầu.

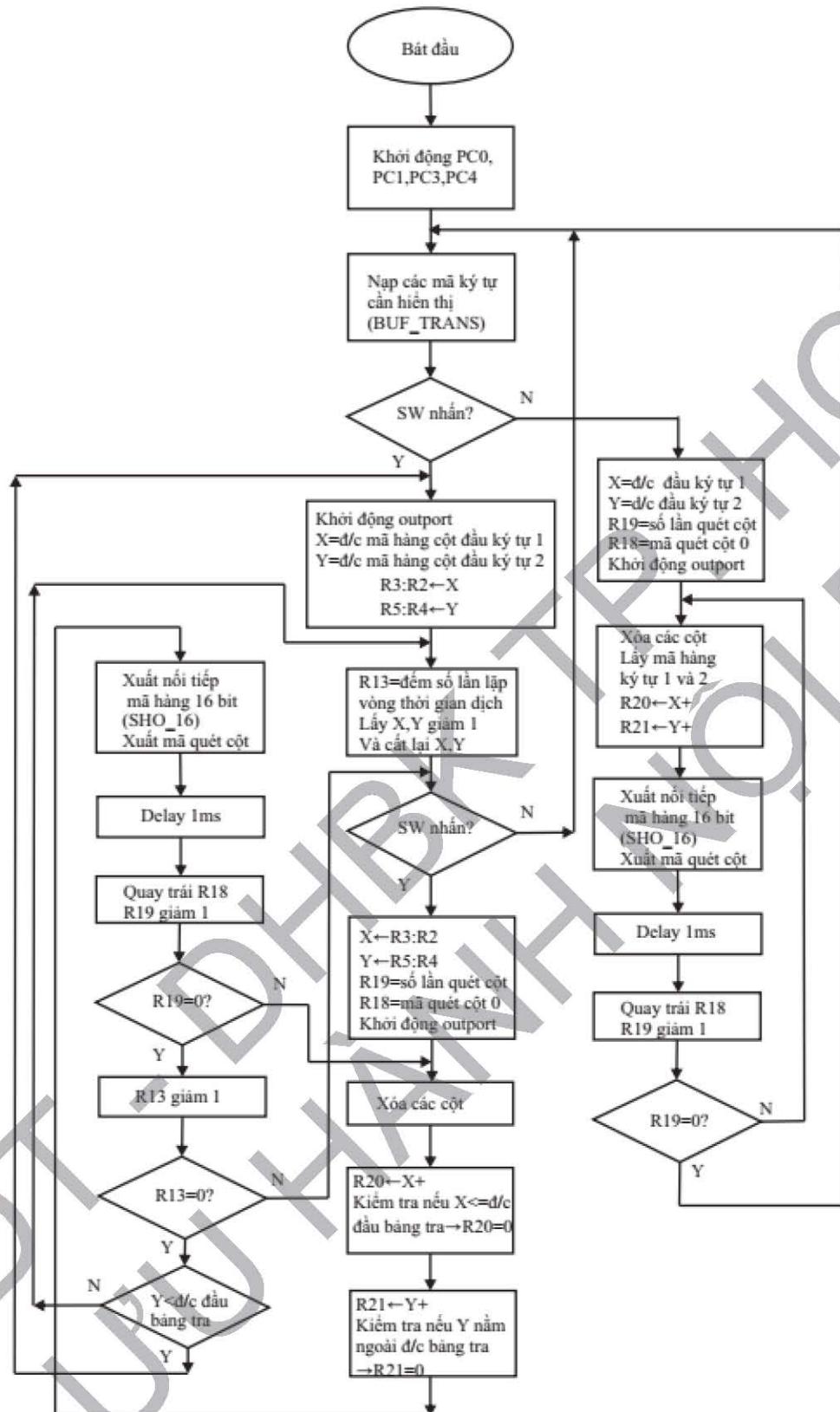


Hình 6.28: Các mô thức hiển thị và truy xuất data từ bảng tra trong SRAM

Hình 6.29 trình bày lưu đồ giải thuật chi tiết chương trình ví dụ 6.14.

- Trong mô thức hiển thị đứng yên SW không nhấn, mỗi lần quét cột, 2 con trỏ X, Y truy xuất 2 byte mã hàng của 2 ký tự ứng với cột hiển thị, xuất nối tiếp 16 bit ra port, thanh ghi dịch chuyển thành song song lái các hàng của đèn tương ứng.

- Trong mô thức hiển thị dịch phải SW nhấn, con trỏ X khởi động tại địa chỉ cuối bảng tra+1, con trỏ Y bằng X cộng thêm số byte mã hàng 1 ký tự(8 byte). Mỗi lần dịch phải, khởi động lại X và Y giảm 1 địa chỉ. Thay đổi tốc độ hay thời gian dịch bằng cách thay đổi số lần lập vòng 1 chu kỳ quét 8 cột của bảng đèn (R13 đếm số lần lập vòng). Lập vòng lại chu kỳ dịch mới khi Y nhỏ hơn địa chỉ đầu bảng tra.



Hình 6.29: Lưu đồ giải thuật chi tiết ví dụ 6.14

❖ Chương trình hợp ngữ ví dụ 6.14

```

INCLUDE <M324PDEF.INC>
.EQU FONT_TAB=0X200 ;đặt địa chỉ đầu bảng tra mã font ký tự trong Flash ROM
.INCLUDE "FONT_CHR.INC"; gọi file mã font ký tự
.EQU OUTPORT=PORTB
.EQU INPORT=PINB
.EQU IOSETB=DDRB

```

```

.EQU CONT=PORTC          ;PORTC điều khiển
.EQU CONT_DR=DDRC         ;hướng truyền PORTC
.EQU CONT_OUT=PORTC       ;PORTC=output
.EQU CONT_IN=PINC         ;PORTC input
.EQU SDA=0                ;bit SDA
.EQU LE1=1                ;bit LE1
.EQU IN0=2                ;bit IN0
.EQU SCK=3                ;bit SCK
.EQU SW=4                ;bit SW
.EQU TAB_BUF=0x200        ;địa chỉ đầu bảng tra trong SRAM
.EQU T_SHIFT=5            ;số lần lặp vòng tạo thời gian dịch
.EQU UNI_COL=8             ;số cột quét của 1 đèn=số byte hiển thị ký tự
.EQU TOT_COL=16           ;tổng số cột quét toàn bảng đèn
.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 R16,0XF
OUT CONT_DR,R16            ;khai báo PC0,PC1,PC2 là output,PC4 input
CBI CONT,SDA               ;SDA=PC0=0
CBI CONT,LE1                ;LE1=PC1=0 khóa ngõ ra U3
CBI CONT,SCK               ;SCK=PC3=0 đặt CK=0
SBI CONT,SW                ;R kéo lên SW=PC4
START: LDI R17,'A'          ;nạp ký tự A
LDI XH,HIGH(TAB_BUF);X trỏ địa chỉ đầu bảng tra SRAM
LDI XL,LOW(TAB_BUF)
RCALL BUF_TRANS            ;chuyển mã font ký tự từ Flash ROM vào SRAM
LDI R17,'B'                ;nạp ký tự B
RCALL BUF_TRANS            ;chuyển tiếp mã font ký tự
SBIS CONT IN,SW             ;SW không nhấn chạy mode đứng yên
RJMP START_R                ;SW nhấn chạy mode dịch phải
LDI XH,HIGH(TAB_BUF)      ;X trỏ địa chỉ đầu ký tự 1
LDI XL,LOW(TAB_BUF)
LDI YH,HIGH(TAB_BUF+UNI_COL);Y trỏ địa chỉ đầu ký tự 2
LDI YL,LOW(TAB_BUF+UNI_COL)
LDI R19,UNI_COL              ;đếm 8 lần quét
LDI R18,0X01                ;mã quét cột 0
LDI R16,0xFF
OUT IOSETB,R16              ;PORTB=output
LOOP: CLR R16                ;
OUT OUTPORT,R16             ;xóa toàn bộ các cột
SBI CONT,LE1                ;mở U3
CBI CONT,LE1                ;khóa U3
LD R20,X+                  ;R20=mã hàng ký tự 1,tăng con trỏ hàng ký tự 1
LD R21,Y+                  ;R21=mã hàng ký tự 2,tăng con trỏ hàng ký tự 2
RCALL SHO_16                ;xuất nối tiếp 2 byte mã hàng ký tự 1 và 2
OUT OUTPORT,R18             ;xuất mã quét cột
SBI CONT,LE1                ;mở U3
CBI CONT,LE1                ;khóa U3
RCALL DELAY_1MS             ;tạo trễ 1ms
CLC                         ;xóa C chuẩn bị quay
ROL R18                     ;quay trái tạo mã quét cột kế tiếp

```

DEC	R19	;đếm số lần quét cột
BRNE	LOOP	;thoát khi đú 8 lần quét
RJMP	START	
START_R:	LDI R16,0XFF	
OUT	IOSETB,R16 ;PORTB=output	
LDI	XH,HIGH(TAB_BUF+TOT_COL)	;X trỏ địa chỉ mã hàng cột đầu đèn 1
LDI	XL,LOW(TAB_BUF+TOT_COL)	
LDI	YH,HIGH(TAB_BUF+TOT_COL+UNI_COL);Y trỏ địa chỉ mã hàng cột đầu đèn 2	
LDI	YL,LOW(TAB_BUF+TOT_COL+UNI_COL)	
MOVW	R3:R2,XH:XL	;cắt X
MOVW	R5:R4,YH:YL	;cắt Y
ST_SCAN:	LDI R16,T_SHIFT	;đếm số vòng lặp tạo thời gian dịch
MOV	R13,R16	;R13=số vòng lặp
MOVW	XH:XL,R3:R2	;lấy con trỏ địa chỉ mã hàng cột đầu đèn 1
MOVW	YH:YL,R5:R4	;lấy con trỏ địa chỉ mã hàng cột đầu đèn 2
SBIW	XH:XL,1	;giảm X 1 địa chỉ
MOVW	R3:R2,XH:XL	;cắt X
SBIW	YH:YL,1	;giảm Y 1 địa chỉ
MOVW	R5:R4,YH:YL	;cắt Y
AGAIN:	SBIC CONT_IN,SW	;thoát nếu SW không nhấn
RJMP	START	;trở về đầu chương trình
LDI	R18,0X01	;đặt lại mã quét cột đầu
MOVW	XH:XL,R3:R2	;lấy lại con trỏ địa chỉ mã hàng cột đầu đèn 1
MOVW	YH:YL,R5:R4	;lấy lại con trỏ địa chỉ mã hàng cột đầu đèn 2
ST_SHIFT:	LDI R19,UNI_COL	;đếm 8 lần quét
LOOP_R:	CLR R16	
OUT	OUTPORT,R16	;xóa toàn bộ các cột
SBI	CONT,LE1	;mở U3
CBI	CONT,LE1	;khóa U3
LD	R20,X+	;R20=mã hàng ký tự 1,tăng con trỏ X
CPI	XH,HIGH(TAB_BUF);so sánh byte cao X với địa chỉ đầu bảng tra	
BRCS	LT_MINX	;nhỏ hơn xóa R20
BRNE	CHECK_Y	;lớn hơn chấp nhận R20
CPI	XL,LOW(TAB_BUF);bằng,so sánh byte thấp	
BRCS	LT_MINX	;nhỏ hơn xóa R20
BRNE	CHECK_Y	;lớn hơn chấp nhận R20
LT_MINX:		
CLR	R20	;bằng hoặc nhỏ hơn R20=0
CHECK_Y:		
LD	R21,Y+	;R21=mã hàng ký tự 2,tăng con trỏ Y
CPI	YH,HIGH(TAB_BUF+TOT_COL);so sánh byte cao Y với địa chỉ cuối bảng tra+1	
BRCS	LOW_LIM	;nhỏ hơn xét địa chỉ thấp
BRNE	LT_MINY	;lớn hơn xóa R21
CPI	YL,LOW(TAB_BUF+TOT_COL);bằng,so sánh byte thấp	
BRCS	LOW_LIM	;nhỏ hơn xét địa chỉ thấp
BRNE	LT_MINY	;lớn hơn xóa R21
LOW_LIM:		
CPI	YH,HIGH(TAB_BUF);so sánh byte cao Y với địa chỉ đầu bảng tra	
BRCS	LT_MINY	;nhỏ hơn xóa R21
BRNE	SHIFT_OUT	;lớn hơn chấp nhận R21
CPI	YL,LOW(TAB_BUF);bằng,so sánh byte thấp	
BRCS	LT_MINY	;nhỏ hơn xóa R21
BRNE	SHIFT_OUT	;lớn hơn chấp nhận R21
LT_MINY:		
CLR	R21	;bằng hoặc nhỏ hơn R21=0

SHIFT_OUT:

```

    RCALL SHO_16          ;xuất nối tiếp 2 byte mã hàng ký tự 1 và 2
    OUT   OUTPORT,R18     ;xuất mã quét cột
    SBI   CONT,LE1         ;mở U3
    CBI   CONT,LE1         ;khóa U3
    RCALL DELAY_1MS       ;tạo trễ 1ms
    CLC
    ROL   R18              ;xóa C chuẩn bị quay trái
    DEC   R19              ;quay trái tạo mã quét cột kế tiếp
    BRNE  LOOP_R           ;đếm số lần quét cột
    DEC   R13              ;đếm số vòng lặp tạo thời gian dịch
    BRNE  AGAIN             ;dịch tiếp khi chưa đủ thời gian
    CPI   YH,HIGH(TAB_BUF-1);so sánh Y với địa chỉ đầu bảng tra mã hàng-1
    BRNE  DIF               ;không bằng ,kiểm tra nhỏ hơn
    CPI   YL,LOW(TAB_BUF-1)
    BRNE  DIF               ;không bằng kiểm tra nhỏ hơn
    RJMP  START_R           ;bằng quay lại từ đầu
    DIF:  BRCC  ST_SCAN      ;còn lớn hơn tiếp tục
    RJMP  START_R           ;nhỏ hơn quay lại từ đầu
;
```

BUF_TRANS:

```

    LDI   R19,UNI_COL        ;R19 đếm số byte 1 bộ mã font ký tự
    LDI   ZH,HIGH(TAB_CHR);Z trả địa chỉ đầu bảng tra bộ mã font ký tự
    LDI   ZL,LOW(TAB_CHR)
    SUBI R17,$20            ;lấy offset bằng cách trừ $20
    LDI   R16,4              ;và nhân cho 4
    MUL   R17,R16
    ADD   R30,R0              ;cộng offset vào địa chỉ nền đầu bảng tra
    ADC   R31,R1
    CLC
    ROL   R30              ;chuẩn bị quay qua C
    ROL   R31              ;quay trái ZL qua C
    ROL   R31              ;quay trái ZH qua C tạo địa chỉ nền truy xuất bộ nhớ CT
;
```

GET_CODE:

```

    LPM   R17,Z+            ;lấy mã font ký tự,tăng địa chỉ con trả
    ST    X+,R17             ;cắt mã hàng vào SRAM
    DEC   R19              ;đếm số byte cắt
    BRNE GET_CODE           ;thoát khi đủ 8 byte mã hàng
    RET
;
```

;SHO_16 dịch trái 16 bit data cắt trong R21,R20 ra ngõ SDA
;Input: R21,R20 R21 byte cao
;Output SDA nối tiếp MSB trước
;Phải dịch và xuất 17 lần mới đúng vị trí hàng(xem lại hình 6.18)

SHO_16: LDI	R17,17	;đếm 17 lần dịch
SH_LOOP: ROL	R20	;quay trái qua C byte thấp C=b7,b0=C
ROL	R21	;quay trái qua C byte cao C=b7,b0=C
BRCC	BIT 0	;C=0 nhảy đến BIT 0
SBI	CONT,SDA	;dịch bit 7=1 byte cao ra SDA
RJMP	NEXT	;nhảy đến NEXT
BIT 0: CBI	CONT,SDA	;dịch bit 7=0 byte cao ra SDA
NEXT: SBI	CONT,SCK	;tạo xung CK
CBI	CONT,SCK	
DEC	R17	;đếm số lần dịch
BRNE	SH_LOOP	;dịch đủ 17 lần

RET

;

DELAY_1MS:

LDI	R16,8	;1MC sử dụng R16
MOV	R15,R16	;1MC nạp data cho R15
LDI	R16,250	;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

❖ Chương trình C VD6-14

```
#include <avr/io.h>
#include "font_chr.h" //Gọi hàm thư viện font ký tự
#define import PINB //định nghĩa import=PINB
#define outport PORTB //định nghĩa outport=PORTB
#define iosetb DDRB //định nghĩa iosetb=định hướng PORTB
#define cont PORTC //định nghĩa cont=PORTC
#define cont_dr DDRC //định nghĩa cont_dr=DDRC
#define cont_in PINC //định nghĩa cont_in=PINC đọc SW=PC4
#define serial_out PORTC //định nghĩa serial_out=PORTC serial out=PC0
const char dis_io=0xf4 ;//mã khóa các ngoại vi,SCK=0,SDA=0 b11110100
const char en_u3=0xf6 ;//mã mở U3=b11110110
const char ck_high=0x08;//PC3=1 b00001000
const char ck_low=0xf7 ;//PC3=0 b11110111
const char shift_bit1=0x01;//PC0=1
const char shift_bit0=0xfe;//PC0=0
const char SW_open=0x10;//PC4=1
const unsigned int TAB_BUF=0x420,T_SHIFT=5,UNI_COL=8,TOT_COL=16;
void SHO_16() //khai báo hàm ghi dịch 16 bit
void delay_ms(unsigned int n) //khai báo hàm delay 1ms
unsigned char i,tam,col,*px,*py;
unsigned int offset,buf16,tam_px,tam_py,sh_count ;
int main()
{
    cont_dr=0x0f //định nghĩa PC0,PC1,PC3 =output,PC4=input
    cont=dis_io //khóa các ngoại vi,SCK=0,SDA=0,R kéo lên PC4
    while(1)
    {
        START:
        px=TAB_BUF; //px trỏ địa chỉ đầu bảng tra(TAB_BUF=0x420)
        tam='A' //nhập ký tự 1
        offset=(tam-0x20)*8 //lấy offset
        for(i=0;i<=7;i++) //lấy 8 byte mã hàng
        {
            *px=TAB_CHR[offset+i];//lấy mã hàng ký tự A
            px++; //tăng con trỏ địa chỉ
        }
        tam='B' //nhập ký tự 2
        offset=(tam-0x20)*8 //lấy offset
```

```

for(i=0;i<=7;i++)
{
    *px=TAB_CHR[offset+i];//lấy mã hàng ký tự B
    px++;
}
if(cont_in&SW_open)      //kiểm tra SW nhấn?
{
    px=TAB_BUF;          //px trỏ địa chỉ hàng đầu ký tự 1
    py=TAB_BUF+UNI_COL;//py trỏ địa chỉ hàng đầu ký tự 2
    col=0x01              //PORTB=outport
    for(i=0;i<=7;i++)     //quét 8 cột
    {
        outport=0x00       ;//xóa toàn bộ đèn
        cont=en_u3          ;//mở U3
        cont=dis_io          ;//khóa U3
        buf16=*py++;         ;//lấy mã hàng ký tự 2
        buf16=(buf16<<8)*px++;//ghép với mã hàng ký tự 1 thành 16 bit
        SHO_16();            //dịch 16 bit nối tiếp ra port
        outport=col           ;//xuất mã quét cột
        cont=en_u3          ;//mở U3
        cont=dis_io          ;//khóa U3
        delay_ms(1333)       ;//trễ 1ms
        col=col<<1           ;//tạo mã quét kế tiếp
    }
}
else
{
    START_R:             //SW nhấn hiển thị mode dịch phải
    iosetb=0xff;          //khai báo port output
    px=TAB_BUF+TOT_COL;//px trỏ địa chỉ cuối bảng tra+1
    py=TAB_BUF+TOT_COL+UNI_COL;//py trỏ địa chỉ hơn px 8 byte
    tam_px=px;            //cắt px
    tam_py=py;            //cắt py
    while(py>=TAB_BUF)   //lặp vòng khi py>=địa chỉ đầu bảng tra
    {
        px=tam_px--;      //giảm px 1 dịch phải
        py=tam_py--;      //giảm py 1 dịch phải
        for(sh_count=1;sh_count<=T_SHIFT;sh_count++)//lặp vòng thời gian dịch
        {
            if(cont_in&SW_open) // kiểm tra SW nhấn?
                goto START;    //SW không nhấn thoát về START
            else               //SW nhấn chạy mode dịch phải
            {
                px=tam_px;      //lấy lại px
                py=tam_py;      //lấy lại py
                col=0x01;        //mã quét cột 0
                for(i=0;i<=7;i++) //lặp vòng quét 8 cột
                {
                    outport=0x00 ;//xóa toàn bộ đèn
                    cont=en_u3 ;//mở U3
                    cont=dis_io ;//khóa U3
                    buf16=*py++ ;//lấy mã hàng ký tự 2,tăng py thêm 1
                    if((TAB_BUF >= py)|(TAB_BUF+TOT_COL<=py))//py ngoài
                        buf16=0x0000;//vùng bảng tra,xóa mã hàng ký tự 2
                    buf16=(buf16<<8)*px++;//lấy mã hàng ký tự 2 ghép thành 16 bit
                }
            }
        }
    }
}

```

- Khi khởi động, trình biên dịch C sẽ tự động chuyển dãy TAB_CHR[] từ Flash ROM sang SRAM địa chỉ đầu đặt tại 0x0100 đến 0x0407 (với MCU324P) dài 776 byte.
 - Để dễ theo dõi khi chạy mô phỏng, ta đặt địa chỉ đầu buffer chứa mã hàng 2 ký tự TAB_BUF giá trị cụ thể. Trong chương trình đặt TAB_BUF=0x0420 nằm ngoài vùng chứa mã ký tự như trên và dữ trữ các biến
 - Ta tạo bảng tra mã font ký tự lái bảng đèn (2 ký tự A và B) có địa chỉ đầu TAB_BUF=0x420, sử dụng 2 con trỏ px và py lần lượt trỏ địa chỉ mã hàng của ký tự 1 và ký tự 2 xuất ra cùng vị trí cột.

❖ Câu hỏi ôn tập

1. Từ quy tắc thiết lập font mã ký tự như hình 6.24, thiết lập bộ mã font ký tự cho các ký hiệu α , β .
 2. Cũng từ hình 6.14, nhưng áp dụng cách quét hàng và xuất mã cột tương ứng theo hàng, thiết lập bộ mã font ký tự chữ A,B.
 3. Từ sơ đồ hình 6.25, nếu phân cực dòng trung bình một điểm sáng bằng 5mA, U2 vẫn còn lái trực tiếp các chân hàng của bộ hiển thị được không? Nếu không lái được, hãy đưa ra cách thiết kế khác.
 4. Nếu áp dụng cách thiết lập mã quét hàng như câu 2, phải điều chỉnh lại sơ đồ hình 6.25 như thế nào cho phù hợp điều kiện dòng điện?

5. Từ bộ mã font ký tự như câu 2, viết một đoạn chương trình xuất ký tự A ra bộ hiển thị LED ma trận 8x8 theo sơ đồ hình 6.25(giả sử đã điều chỉnh phù hợp điều kiện dòng điện)

6.5 Giao tiếp bộ hiển thị LCD ký tự,LCD đồ họa

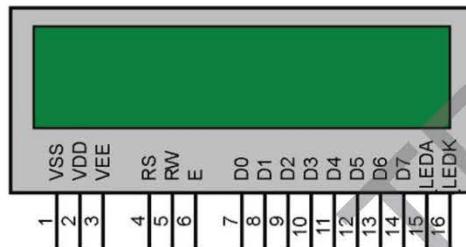
Để có thể hiển thị đa dạng ký tự và hình ảnh, dễ dàng giao tiếp và lập trình, tiết kiệm năng lượng nhất là đối với thiết bị cầm tay, người ta thường sử dụng LCD ký tự hay LCD đồ họa làm bộ hiển thị.

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.

Trong phần này chủ yếu ta sẽ nghiên cứu cách giao tiếp với LCD ký tự 16x2 16 ký tự/2 dòng(IC lái HD44780) và LCD đồ họa 128x64 điểm hàngxcolumn (IC lái KS0108).

6.5.1 Giao tiếp LCD ký tự 16x2

Hình 6.30 là sơ đồ chân bảng hiển thị LCD 16x2 có 2 dòng và 16 ký tự/dòng.



Hình 6.30: Sơ đồ chân LCD 16x2

Bảng 6.9 mô tả chức năng chân LCD 16x2

Bảng 6.9: Chức năng các chân bảng đèn LCD 16x2

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=1: 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

Bảng đèn LCD 16x2 ký tự được lái bởi IC HD44780. Về cấu hình và hoạt động chi tiết, người đọc có thể tham khảo data sheet của IC HD44780 trong phần phụ lục. Trong phần sau chỉ tóm tắt một số yêu cầu hoạt động cơ bản của LCD 16x2 trích từ data sheet IC HD44780.

Bảng đèn LCD ký tự có 8 chân data D0..D7 và 3 chân điều khiển RS,RW,E kết nối với thiết bị lái. Hai chân A và K của LED nền bảng đèn, muốn sáng nền chỉ cần phân cực thuận AK hạn dòng bằng điện trở 4.7 đến 10Ω/0.5W.

1. Các chân điều khiển RS,RW,E

LCD ký tự có 2 thanh ghi: thanh ghi lệnh(IR) và thanh ghi data(DR), phân biệt bằng RS.Khi MCU truy xuất LCD,nếu RS=0 data chính là mã lệnh sẽ chuyển đến IR,nếu RS=1 data là mã ASCII chuyển đến DR hiển thị ký tự tương ứng ra LCD

- Tóm tắt giao tiếp giữa MCU với LCD
 - Ghi lệnh: RS=0,RW=0,E=1,D7-D0=mã lệnh
 - Đọc lệnh: RS=0,RW=1,E=1,D7=BF:BF=1 LCD bận,BF=0 LCD rỗi
 - Ghi data: RS=1,RW=0,E=1:D7-D0= data
 - Đọc data: RS=1,RW=1,E=1:D7-D0=data

2. Giao tiếp data 8 bit hay 4 bit

LCD ký tự cho phép giao tiếp 2 dạng data:

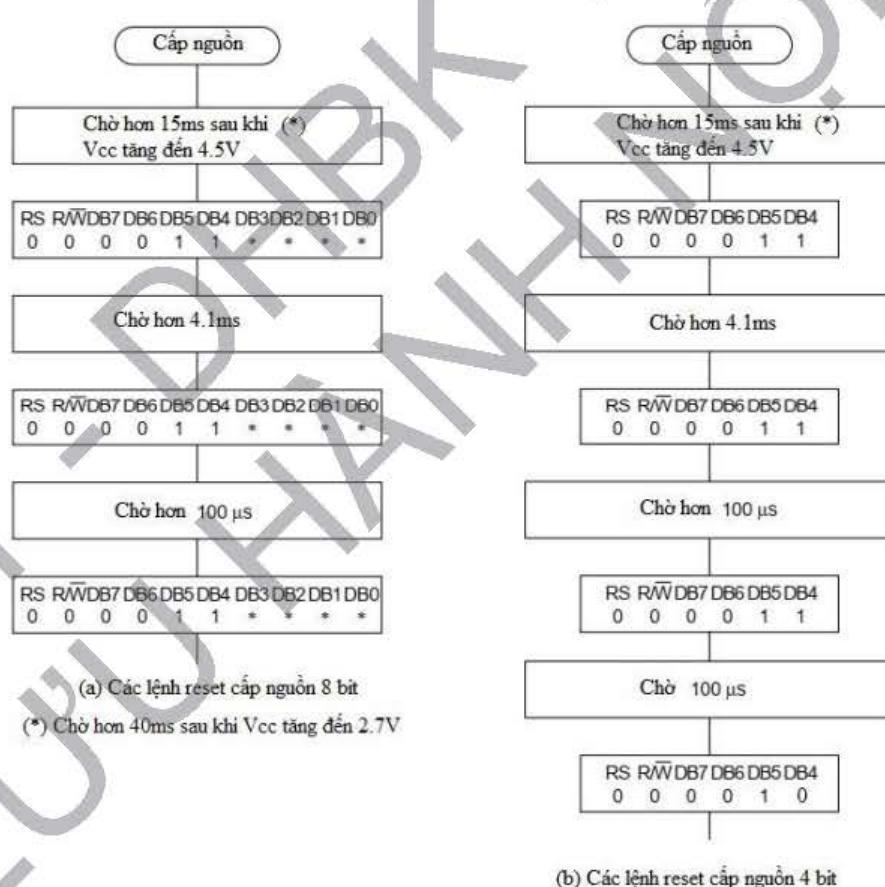
- Giao tiếp 8 bit: kết nối 8 chân D0..D7 với 8 chân port tương ứng của MCU.Mỗi lệnh giao tiếp chỉ cần truy xuất 1 byte kết hợp 3 tín hiệu điều khiển
- Giao tiếp 4 bit: kết nối 4 chân D4..D7 với 4 chân port tương ứng của MCU.Mỗi lệnh giao tiếp cần truy xuất 2 lần,mỗi lần 4 bit kết hợp với 3 tín hiệu điều khiển

➤ *Truy xuất 4 bit cao trước, chờ 100μs(xem phần định thời truy xuất), truy xuất tiếp 4 bit thấp*

3. Reset nguồn khởi động LCD

Trong trường hợp mới cấp nguồn cho LCD, để đạt điều kiện reset khởi động nguồn bên trong mạch, chỉ cần chờ từ 100ms trở lên sau khi nguồn Vcc đạt 4.5V là đạt yêu cầu.

Trong trường hợp không kiểm soát được việc reset khởi động nguồn cấp cho LCD và MCU, nên thêm phần reset khởi động nguồn cho LCD như hình 6.31, bằng cách ghi các lệnh cho LCD tùy vào cách giao tiếp 8 hay 4 bit.



Hình 6.31: Các lệnh reset cấp nguồn cho LCD ký tự

4. Khởi động LCD

Khởi động LCD: thường thực hiện 4 lệnh ghi mã lệnh sau ra LCD như hình 6.32

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

Hình 6.32: Bốn mã lệnh khởi động LCD

- **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
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 chép,B=0 ký tự và con trỏ không chép
- **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

Ví dụ 6.15: Ghi các mã lệnh khởi động LCD 2 dòng font 5x8;xóa màn hình;màn hình on xóa con trỏ;con trỏ dịch phải,địa chỉ DDRAM tăng 1 khi ghi data,màn hình không dịch.

Giải:

Function set: 2 dòng font 5x8	mã lệnh=38H
Clear display: xoá 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

5. Ghi data ra LCD(CG/DDRAM)

- Data là mã ASCII hiển thị ký tự theo data sheet(xem Table 4,5 data sheet HD44780)
- Địa chỉ CG/DDRAM tăng/giảm 1 tùy theo lệnh entry mode set

6. Đọc trạng thái cờ báo bận và địa chỉ

- BF=1:LCD bận không nhận lệnh
- BF=0: LCD rỗi sẵn sàng nhận lệnh
- AAAAAAA:địa chỉ CG/DDRAM của lệnh trước đó

➤ *Thay vì đọc cờ BF,có thể delay từ 50μs đến 1.6ms tùy loại lệnh ,chờ LCD thực hiện xong lệnh(xem định thi lệnh trong data sheet)*

		RS R/W DB7 DB6 DB5 DB4 DB3 DB2 DB1 DB0										
Write data to CG or DDRAM	Code	<table border="1" style="margin-left: auto; margin-right: auto;"> <tr><td>1</td><td>0</td><td>D</td><td>D</td><td>D</td><td>D</td><td>D</td><td>D</td><td>D</td><td>D</td></tr> </table> <p style="text-align: center;">← Higher order bits Lower order bits →</p>	1	0	D	D	D	D	D	D	D	D
1	0	D	D	D	D	D	D	D	D			
		RS R/W DB7 DB6 DB5 DB4 DB3 DB2 DB1 DB0										
Read busy flag and address	Code	<table border="1" style="margin-left: auto; margin-right: auto;"> <tr><td>0</td><td>1</td><td>BF</td><td>A</td><td>A</td><td>A</td><td>A</td><td>A</td><td>A</td><td>A</td></tr> </table> <p style="text-align: center;">← Higher order bit Lower order bit →</p>	0	1	BF	A	A	A	A	A	A	A
0	1	BF	A	A	A	A	A	A	A			

Hình 6.33: Các lệnh ghi data ra CG/DDRAM và đọc cờ báo bận và địa chỉ

➤ **Địa chỉ truy xuất CG/DDRAM hay vị trí hiển thị ký tự/con trỏ trên màn hình LCD 16x2:**

- Dòng 1: mã lệnh \$80 đến \$8F: vị trí 1 đến 16

- Dòng 2: mã lệnh \$C0 đến \$CF: vị trí 1 đến 16

Ví dụ muốn đặt con trỏ ở vị trí thứ 4 hàng 1, ta ghi mã lệnh \$83 ra LCD. Muốn dịch con trỏ đến vị trí thứ 11 dòng 2 ta ghi mã lệnh \$CA ra LCD.

➤ **Định thời truy xuất lệnh**

- Theo data sheet HD44780, với tần số dao động nội tiều chuẩn $fosc=270\text{Khz}$, thời gian truy xuất các lệnh tối đa là $37\mu\text{s}$, trừ 2 lệnh Return home(\$02) và Clear display(\$01) dài 1.52ms .

- Do đó thay vì kiểm tra cờ BF để biết LCD có bận không, ta có thể gọi chương trình con delay $100\mu\text{s}$, ngoại trừ sau 2 lệnh Return home và Clear display delay 2ms , trước khi truy xuất LCD

Ta lưu ý các thông số thời gian từ hình 6.34 như sau:

- Chu kỳ E(chu kỳ cho phép GLCD) $t_{cycEmin}=500\text{ns}$

- Độ rộng xung E mức cao $PW_{EHmin}=230\text{ns}$

- Thời gian đặt địa chỉ(từ lúc đặt RS,R/W đến cạnh lên của E) $t_{ASmin}=40\text{ns}$

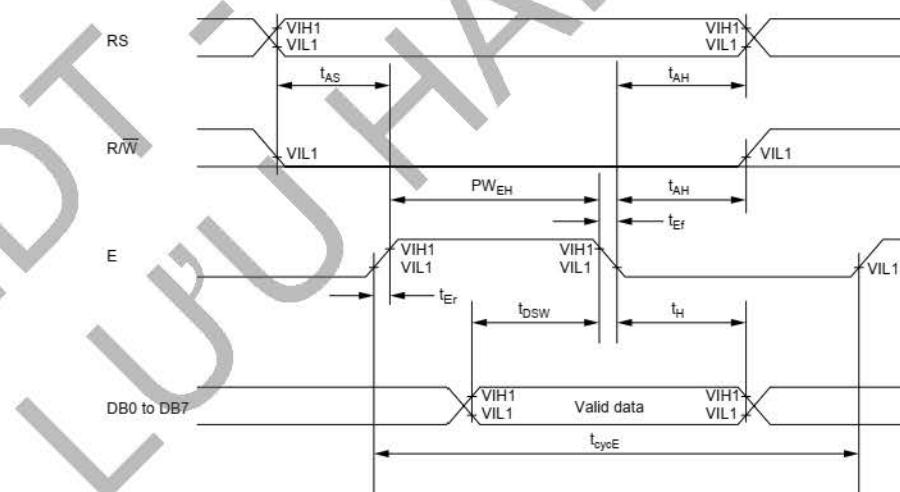
- Thời gian giữ địa chỉ(từ lúc cạnh xuống của E đến khi RS,R/W thay đổi) $t_{AHmin}=10\text{ns}$

- Thời gian đặt data(ghi)(từ lúc đặt data ghi đến cạnh xuống của E) $t_{DSWmin}=80\text{ns}$

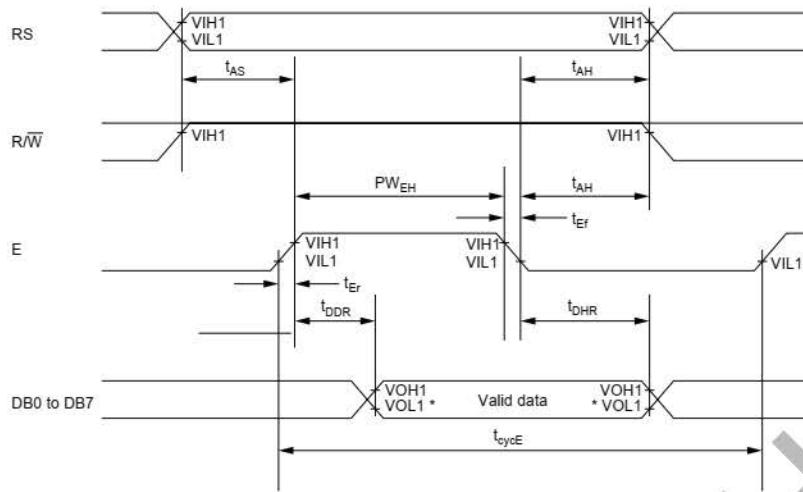
- Thời gian giữ data(ghi)(từ lúc cạnh xuống của E đến khi data thay đổi) $t_{Hmin}=10\text{ns}$

- Thời gian trễ data(đọc)(từ lúc cạnh lên của E đến khi data xuất hiện) $t_{DDRmax}=160\text{ns}$

- Thời gian giữ data(đọc)(từ lúc cạnh xuống của E đến khi data thay đổi) $t_{DHRmin}=5\text{ns}$



(a) Giản đồ xung định thời ghi LCD ký tự



(b) Giản đồ xung định thì đọc LCD ký tự

Hình 6.34: Giản đồ xung định thì ghi/đọc LCD ký tự

- Từ các thông số thời gian trên, ta lưu ý tính thời gian xuất các tín hiệu điều khiển theo chu kỳ máy MC như sau ($F_{osc}=8\text{Mhz}, 1\text{MC}=125\text{ns}$):

- Thời gian giữa 2 lần truy xuất LCD ký tự > 500ns # 4MC
- Thời gian tín hiệu E mức 1/mức 0 > 230ns # 2MC
- Thời gian từ lúc đặt các tín hiệu RS,R/W đến khi E chuyển từ 0 lên 1 > 40ns # 1MC
- Thời gian đặt data ghi ra bus data LCD đến khi E chuyển từ 1 xuống 0 > 80ns # 1MC
- Thời gian chờ đọc data từ lúc E chuyển từ 0 lên 1 > 160ns # 2MC

❖ Tóm tắt các mã lệnh điều khiển LCD ký tự

Bảng 6.10 tóm tắt các mã lệnh điều khiển LCD ký tự thường hay sử dụng

Bảng 6.10: Tóm tắt một số mã lệnh thông dụng (điều khiển RS=0, RW=0)

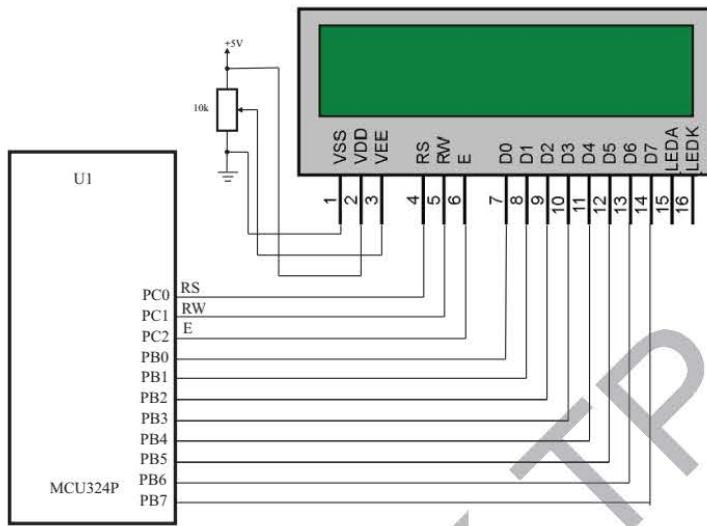
Mã(Hex)	Thực hiện
01	Xóa màn hình(Clear display)
02	Trở về vị trí đầu dòng(Return home)
04	Dịch con trỏ sang trái(khi ghi/đọc data)
05	Dịch màn hình sang phải(khi ghi/đọc data)
06	Dịch con trỏ sang phải(khi ghi/đọc data)
07	Dịch màn hình sang trái(khi ghi/đọc data)
08	Tắt màn hình,tắt con trỏ
0A	Tắt màn hình,hiện con trỏ
0C	Hiện màn hình,tắt con trỏ
0E	Hiện màn hình,không chớp ký tự chỉ bởi con trỏ
0F	Hiện màn hình,chớp ký tự chỉ bởi con trỏ
10	Dịch con trỏ sang trái (không thay đổi nội dung DDRAM)
14	Dịch con trỏ sang phải (không thay đổi nội dung DDRAM)
18	Dịch màn hình sang trái(không thay đổi nội dung DDRAM)
1C	Dịch màn hình sang phải(không thay đổi nội dung DDRAM)
80	Chuyển con trỏ về đầu dòng 1
C0	Chuyển con trỏ về đầu dòng 2
38	Đặt chức năng giao tiếp 8 bit,2 dòng,5X8 dots
28	Đặt chức năng giao tiếp 4 bit cao,2 dòng,5X8 dots

Ví dụ 6.16: Hình 6.35 là sơ đồ MCU324P giao tiếp trực tiếp với LCD 16x2 giao tiếp 8 bit. Các thông số DC của LCD đều phù hợp mức logic với MCU324P(xem data sheet HD44780). Viết một chương trình điều khiển LCD 16x2 thực hiện các công việc sau:

- Reset cấp nguồn cho LCD

- Khởi động LCD như ví dụ 6.15, viết thành chương trình con
- Hiển thị ra LCD 2 dòng ký tự như sau

Xin chao!
 Mon Vi xu ly



Hình 6.35: Sơ đồ thiết kế LCD giao tiếp 8 bit trực tiếp với MCU324P

Giải:

- Sử dụng PORTB giao tiếp 8 bit với LCD
- Các tín hiệu điều khiển RS=PC0,RW=PC1,E=PC2
- Để reset cấp nguồn LCD ta thực hiện các bước lệnh như hình 6.31a
- Chương trình con khởi động LCD giao tiếp 8 bit đặt mode làm việc như ví dụ 6.15

Function set: 2 dòng font 5x8	mã lệnh=38H
Clear display: xoá 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
- Để hiển thị 2 dòng chữ ta sử dụng bảng tra ghi mã ASCII tương ứng trong Flash ROM,dòng 1 kết thúc bằng mã \$0D(ký tự xóng dòng),dòng 2 kết thúc bằng mã \$00(mã NULL)
- Để điều khiển xuất 2 dòng chữ ra LCD theo mẫu trên,ta đếm số ký tự từng dòng và vị trí cần hiển thị.Dòng đầu có 9 ký tự kể cả dấu cách,bắt đầu hiển thị từ vị trí thứ 4(mã \$83).Dòng thứ hai có 12 ký tự,bắt đầu hiển thị từ vị trí thứ 3(mã \$C2)

❖ Chương trình hợp ngữ VD6.16

```

INCLUDE <M324PDEF.INC>
.EQU OUTPORT=PORTB      ;PORTB data
.EQU INPORT=PINB
.EQU IOSETB=DDRB
.EQU CONT=PORTC          ;PORTC điều khiển
.EQU CONT_DR=DDRC          ;
.EQU CONT_OUT=PORTC          ;
.EQU CONT_IN=PINC          ;
.EQU RS=0                  ;bit RS
.EQU RW=1                  ;bit RW
.EQU E=2                   ;bit E
.EQU CR=$0D                ;mã xuống dòng
.EQU NULL=$00              ;mã kết thúc
  
```

```

.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 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 cám LCD
      LDI R16,0XFF
      OUT IOSETB,R16     ;khai báo outport
      LDI R16,250          ;delay 25ms
      RCALL DELAY_US      ;ctc delay 100μsxR16
      LDI R16,250          ;delay 25ms
      RCALL DELAY_US      ;ctc delay 100μsxR16
      CBI CONT,RS          ;RS=0 ghi lệnh
      LDI R17,$30          ;mã lệnh=$30 lần 1
      RCALL OUT_LCD       ;ctc ghi ra LCD
      LDI R16,42            ;delay 4.2ms
      RCALL DELAY_US
      CBI CONT,RS
      LDI R17,$30          ;mã lệnh=$30 lần 2
      RCALL OUT_LCD       ;delay 200μs
      LDI R16,2
      RCALL DELAY_US
      CBI CONT,RS
      LDI R17,$30          ;mã lệnh=$30 lần 3
      RCALL OUT_LCD
      LDI R18,$38          ;Function set 2 dòng font 5x8
      LDI R19,$01          ;Clear display
      LDI R20,$0C          ;display on,con trỏ off
      LDI R21,$06          ;Entry mode set dịch phải con trỏ,DDRAM tăng 1 đ/c
      START: LDI R16,1        ;khi nhập ký tự,màn hình không dịch
      RCALL INIT_LCD8      ;ctc khởi động LCD 8 bit
      LDI R16,1            ;chờ 100μs
      RCALL DELAY_US
      CBI CONT,RS          ;RS=0 ghi lệnh
      LDI R17,$01          ;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,$83          ;con trỏ bắt đầu ở dòng 1 vị trí thứ 4
      RCALL OUT_LCD
      LDI ZH,HIGH(TAB<<1) ;Z trỏ đầu bảng tra ký tự
      LDI ZL,LOW(TAB<<1)  ;lấy mã ASCII ký tự từ Flash ROM
      LINE1: LPM R17,Z+      ;kiểm tra ký tự xuống dòng?
      CPI R17,CR            ;ký tự CR xuống dòng
      BREQ DOWN             ;chờ 100μs
      LDI R16,1
      RCALL DELAY_US
      SBI CONT,RS          ;RS=1 ghi data hiển thị LCD

```

```

        RCALL OUT_LCD          ;ghi mã ASCII ký tự ra LCD
        RJMP  LINE1            ;tiếp tục hiển thị dòng 1
DOWN:   LDI    R16,1           ;chờ 100µs

        RCALL DELAY_US         ;RS=0 ghi lệnh
        CBI   CONT,RS           ;con trỏ bắt đầu ở dòng 2 vị trí thứ 3
        LDI    R17,$C2           ;
        RCALL OUT_LCD          ;lấy mã ASCII ký tự từ Flash ROM
LINE2:  LPM   R17,Z+           ;kiểm tra ký tự kết thúc
        CPI    R17,NULL          ;ký tự NULL thoát
        BREQ  WAIT              ;chờ 100µs
        LDI    R16,1
        RCALL DELAY_US
        SBI    CONT,RS
        RCALL OUT_LCD
        RJMP  LINE2
WAIT:   LDI    R18,80          ;RS=1 ghi data hiển thị LCD
AG1:    LDI    R16,250          ;ghi mã ASCII ký tự ra LCD
        RCALL DELAY_US          ;tiếp tục hiển thị dòng 2
        DEC    R18              ;lập vòng delay 80x25ms=2s
        BRNE  AG1               ;delay 25ms
        RJMP  START

;-----;INIT_LCD8 khởi động LCD ghi 4 byte mã lệnh
;Function set:R18=$38 2 dòng font 5x8
;Clear display:R19=$01 xóa màn hình
;Display on/off control:R20=$0C màn hình on,con trỏ off
;Entry mode set:R21=$06 dịch phải con trỏ ,đ/c DDRAM tăng 1 khi ghi data
;-----;INIT_LCD8: LDI    R16,1           ;chờ 100µs
        RCALL DELAY_US          ;RS=0 ghi lệnh
        CBI   CONT,RS           ;R18=Function set
        MOV    R17,R18           ;
        RCALL OUT_LCD          ;chờ 100µs
        LDI    R16,1
        RCALL DELAY_US          ;RS=0 ghi lệnh
        CBI   CONT,RS           ;R19=Clear display
        MOV    R17,R19           ;chờ 2ms sau lệnh Clear display
        RCALL OUT_LCD
        LDI    R16,20
        RCALL DELAY_US          ;RS=0 ghi lệnh
        CBI   CONT,RS           ;R20=Display on/off control
        MOV    R17,R20           ;
        RCALL OUT_LCD          ;chờ 100µs
        LDI    R16,1
        RCALL DELAY_US          ;RS=0 ghi lệnh
        CBI   CONT,RS           ;R21=Entry mode set
        MOV    R17,R21           ;chờ 100µs
        RCALL OUT_LCD
        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,PWEH=2MC=250ns,tDSW=3MC=375ns
RET

;-----
;DELAY_US tạo thời gian trễ =R16x100μs(Fosc=8Mhz)
;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

- Đầu chương trình là phần reset cấp nguồn cho mode giao tiếp LCD 8 bit như hình 6.31 gồm các lệnh: delay 50ms→ghi mã lệnh \$33→delay 4.2ms→ghi mã lệnh \$33→delay 200μs→ghi mã lệnh \$33
- Tiếp theo là chương trình con INIT_LCD8 khởi động các mode làm việc cho LCD theo 4 mã lệnh: R18=Function set,R19=Clear display,R20=Display on/off control,R21=Entry mode set
- Chương trình con OUT_LCD ghi data ra LCD,đặt RW=0,nếu RS=0 là ghi lệnh,RS=1 là ghi ký tự hiển thị ra LCD.Xuất xung E cho phép LCD độ rộng thỏa điều kiện yêu cầu.
- Trước khi truy xuất,ta gọi chương trình con delay 100μs,ngoại trừ sau khi thực hiện lệnh xóa màn hình delay 2ms,đảm bảo hơn thời gian LCD thực hiện xong lệnh trước đó!
- Đầu tiên chương trình xóa màn hình,sau đó lấy data là mã ASCII từ bảng tra TAB và hiển thị ra LCD từ vị trí con trỏ thứ 4 hàng 1(ghi mã lệnh \$83 trước đó),kiểm tra nếu là ký tự xuống dòng(\$0D)sẽ ghi lệnh xuống dòng đặt con trỏ tại vị trí thứ 3 hàng 2(mã lệnh \$C2).Tiếp tục lấy ký tự ghi ra LCD và kiểm tra nếu là mã kết thúc(\$00) sẽ ngừng ghi,chờ 2s và quay lại từ đầu.

❖ Chương trình C VD6.16

```
#include <avr/io.h>
#define import PINB //định nghĩa import=PINB
#define outport PORTB //định nghĩa outport=PORTB
#define iosetb DDRB //định nghĩa iosetb=định hướng PORTB
#define cont PORTC //định nghĩa cont=PORTC
#define cont_dr DDRC //định nghĩa cont_dr=DDRC
#define cont_in PINC //định nghĩa cont_in=PINC đọc SW=PC4
const char RS_0=0xfe ;//RS=PC0=0 truy xuất lệnh b11111110
const char RS_1=0x01 ;//RS=PC0=1 truy xuất data b00000001
const char RW_0=0xfd ;//RW=PC1=0 truy xuất ghi b11111101
const char RW_1=0x02 ;//RW=PC1=1 truy xuất đọc b00000010
const char E_0=0xfb ;//E=PC2=0 cầm LCD b11111011
const char E_1=0x04 ;//E=PC2=1 cho phép LCD b00000100
const char CR=0xd ;//ký hiệu mã xuống dòng
const char NULL=0x00 ;//ký hiệu mã kết thúc
void OUT_LCD(unsigned char val) ;//khai báo hàm ghi mã lệnh/data ra LCD
void INIT_LCD8(char dat1,char dat2,char dat3,char dat4); //khai báo hàm khởi động LCD
void delay_ms(unsigned int n) ;//khai báo hàm delay ms
unsigned char TAB[]="Xin chao!\rMon Vi xu ly\x00"; //\r,\x00 ký hiệu mã ASCII $0D,$00
```

```

unsigned char i,tam;
int main()
{
    cont_dr=0x07 //định nghĩa PC0,PC1,PC2 =output
    cont=0x00 // E=0,RW=0,RS=0 cầm LCD truy xuất ghi
    iosetb=0xff //khai báo PORTB=output
    delay_ms(33333); //delay 25ms
    delay_ms(33333); //delay 25ms
    cont=cont&RS_0;//truy xuất lệnh
    tam=0x30 //mã lệnh $30 lần 1
    OUT_LCD(tam);
    delay_ms(5600); //delay 4.2ms
    cont=cont&RS_0;//truy xuất lệnh
    tam=0x30 //mã lệnh $30 lần 2
    OUT_LCD(tam);
    delay_ms(2673) //delay 0.2ms
    cont=cont&RS_0;//truy xuất lệnh
    tam=0x30 //mã lệnh $30 lần 1
    OUT_LCD(tam);
    INIT_LCD8(0x38,0x01,0x0c,0x06); //khởi động LCD theo mode như trên
    while(1)
    {
        delay_ms(133) //delay 0.1ms
        cont=cont&RS_0;//truy xuất lệnh
        tam=0x01 //mã lệnh xóa màn hình
        OUT_LCD(tam);
        delay_ms(2667) //delay 2ms sau lệnh clear display
        cont=cont&RS_0;//truy xuất lệnh
        tam=0x83 //mã lệnh dời con trỏ đến vị trí 4 hàng 1
        OUT_LCD(tam);
        i=0 //chỉ số dây ký tự TAB
        tam=TAB[i] //lấy mã ASCII từ bảng tra
        while(tam!=CR) //ký tự CR?
        {
            delay_ms(133) //delay 0.1ms
            cont=cont | RS_1;//truy xuất data
            OUT_LCD(tam) //ghi ký tự hiển thị ra LCD
            i++ //ký tự kế tiếp
            tam=TAB[i] //lấy mã ASCII từ bảng tra
        }
        delay_ms(133) //delay 0.1ms
        cont=cont&RS_0 //truy xuất lệnh
        tam=0xc2 //mã lệnh dời con trỏ đến vị trí 3 hàng 2
        OUT_LCD(tam);
        i++ //trỏ đến ký tự đầu dòng 2
        tam=TAB[i] //lấy mã ASCII từ bảng tra
        while(tam!=NULL)
        {
            delay_ms(133) //delay 0.1ms
            cont=cont | RS_1;//truy xuất data
            OUT_LCD(tam) //ghi ký tự hiển thị ra LCD
            i++ ;
            tam=TAB[i] //lấy mã ASCII từ bảng tra
        }
    }
}

```

```

        for(i=1;i<=50;i++)      //lập vòng delay 50x40ms=2s
            delay_ms(53333); //delay 40ms
    }
}

void INIT_LCD8(char dat1,char dat2,char dat3,char dat4)
{
    delay_ms(133)          ;//delay 0.1ms
    cont=cont&RS_0           ;//truy xuất lệnh
    tam=dat1                ;//mã lệnh function set
    OUT_LCD(tam)            ;
    delay_ms(133)          ;//delay 0.1ms
    cont=cont&RS_0           ;//truy xuất lệnh
    tam=dat2                ;//mã lệnh clear display
    OUT_LCD(tam)            ;
    delay_ms(2667)          ;//delay 2ms sau lệnh clear display
    cont=cont&RS_0           ;//truy xuất lệnh
    tam=dat3                ;//mã lệnh display control on/off
    OUT_LCD(tam)            ;
    delay_ms(133)          ;//delay 0.1ms
    cont=cont&RS_0           ;//truy xuất lệnh
    tam=dat4                ;//mã lệnh entry mode set
    OUT_LCD(tam)            ;
}

void OUT_LCD(unsigned char val)
{
    outport=val;
    cont=cont | E_1;
    cont=cont&E_0;
}

void delay_ms(unsigned int n) // ctc delay ms
{
    unsigned int k;
    for(k=0;k<=n;k++); // Td=6xn MC
}

```

❖ Trường hợp giao tiếp 4 bit

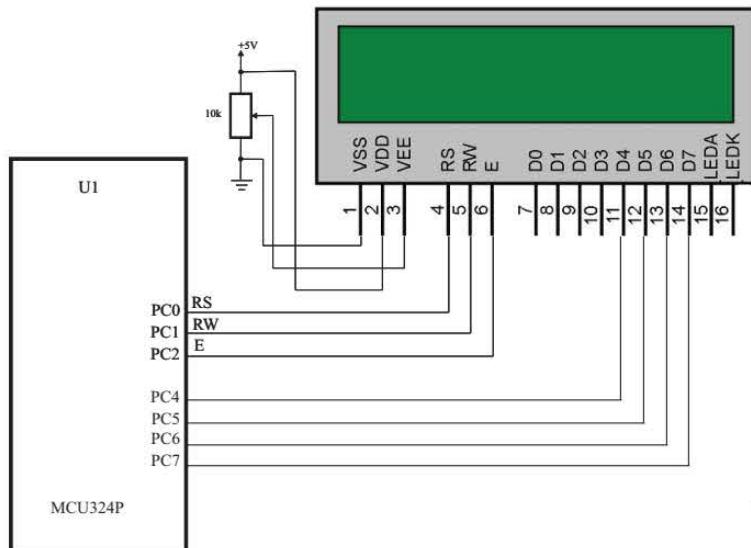
- Ta kết nối 4 chân D7,D6,D5,D4 của LCD với 4 chân port MCU324P,kết nối 3 chân điều khiển như giao tiếp 8 bit

- Khi truy xuất mã lệnh hoặc data,tà chia byte lệnh/data thành 2 lệnh,mỗi lệnh gồm 4 bit kết hợp với các tín hiệu điều khiển,truy xuất 4 bit cao trước.

➤ *Lưu ý là giữa 2 lần truy xuất lệnh 4bit phải có thời gian chờ 100μs!*

Hình 6.36 là sơ đồ giao tiếp MCU324P với LCD ký tự 16x2 theo kiểu giao tiếp 4 bit,chỉ cần 1 Port I/O của MCU324P kết nối 4 chân data D4..D7 và 3 chân điều khiển RS,RW,E của LCD ký tự.Mỗi lần truy xuất lệnh,MCU324P lấy 4 bit data chuyển lên 4 bit cao và ghép với 3 bit thấp điều khiển E,RW,RS xuất data ngõ ra và tín hiệu điều khiển LCD.

Ví dụ 6.17: Từ sơ đồ hình 6.36,lập lại chương trình như ví dụ 6.16 có thêm phản hiệu ứng màn hình như sau: sau khi hiển thị chuỗi ký tự như ví dụ 6.16 trong vòng 2s,chuỗi ký tự sẽ dịch chuyển về phía trái cho đến hết màn hình,sau đó chuỗi ký tự tiếp tục dịch chuyển về phía phải cho đến hết màn hình và lập vòng lại từ đầu.



Hình 6.36: Giao tiếp LCD ký tự 16X2 ghép 4 bit

❖ Chương trình hợp ngữ VD6.17

```

INCLUDE <M324PDEF.INC>
EQU LCD=PORTC ;PORTC giao tiếp BUS LCD16x2
.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 xuống dòng
.EQU NULL=$00 ;mã kết thúc
.ORG 0
ORG 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 R16,0XFF
      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 cầm LCD
      LDI R16,250 ;delay 25ms
      RCALL DELAY_US ;ctc delay 100μsxR16
      LDI R16,250 ;delay 25ms
      RCALL DELAY_US ;ctc delay 100μsxR16
      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
      LDI R16,42 ;delay 4.2ms
      RCALL DELAY_US
      CBI LCD,RS
      LDI R17,$30 ;mã lệnh=$30 lần 2
      RCALL OUT_LCD4 ;delay 200μs
      LDI R16,2
      RCALL DELAY_US

```

```

CBI    LCD,RS
LDI    R17,$32
RCALL OUT_LCD4_2
LDI    R18,$28
LDI    R19,$01
LDI    R20,$0C
LDI    R21,$06
RCALL INIT_LCD4
;mã lệnh=$32
;Function set 2 dòng font 5x8, mode 4 bit
;Clear display
;display on, con trỏ off
;Entry mode set dịch phải con trỏ, DDRAM tăng 1 đ/c
;khi nhập ký tự, màn hình không dịch
;ctc khởi động LCD 4 bit

-----
START: CBI    LCD,RS
LDI    R17,$01
RCALL OUT_LCD4_2
LDI    R16,20
RCALL DELAY_US
CBI    LCD,RS
LDI    R17,$83
RCALL OUT_LCD4_2
LDI    ZH,HIGH(TAB<<1)
LDI    ZL,LOW(TAB<<1)
;RS=0 ghi lệnh
;xóa màn hình
;chờ 2ms sau lệnh Clear display
;RS=0 ghi lệnh
;con trỏ bắt đầu ở dòng 1 vị trí thứ 4
;Z trỏ đầu bảng tra ký tự
;lấy mã ASCII ký tự từ Flash ROM
;kiểm tra ký tự xuống dòng?
;ký tự CR xuống dòng
;RS=1 ghi data hiển thị LCD
;ghi mã ASCII ký tự ra LCD
;tiếp tục hiển thị dòng 1
;chờ 100µs

LINE1: LPM   R17,Z+
CPI   R17,CR
BREQ  DOWN
SBI   LCD,RS
RCALL OUT_LCD4_2
RJMP  LINE1
;RS=0 ghi lệnh
;con trỏ bắt đầu ở dòng 2 vị trí thứ 3
;lấy mã ASCII ký tự từ Flash ROM
;kiểm tra ký tự xuống dòng?
;ký tự CR xuống dòng
;RS=1 ghi data hiển thị LCD
;ghi mã ASCII ký tự ra LCD
;tiếp tục hiển thị dòng 2
;chờ 100µs

DOWN:  LDI   R16,1
RCALL DELAY_US
CBI   LCD,RS
LDI   R17,$C2
RCALL OUT_LCD4_2
;RS=0 ghi lệnh
;con trỏ bắt đầu ở dòng 2 vị trí thứ 3
;lấy mã ASCII ký tự từ Flash ROM
;kiểm tra ký tự xuống dòng?
;ký tự CR xuống dòng
;RS=1 ghi data hiển thị LCD
;ghi mã ASCII ký tự ra LCD
;tiếp tục hiển thị dòng 2
;chờ 100µs

LINE2: LPM   R17,Z+
CPI   R17,NULL
BREQ  WAIT
SBI   LCD,RS
RCALL OUT_LCD4_2
RJMP  LINE2
;RS=0 ghi lệnh
;con trỏ bắt đầu ở dòng 2 vị trí thứ 3
;lấy mã ASCII ký tự từ Flash ROM
;kiểm tra ký tự xuống dòng?
;ký tự NULL thoát
;RS=1 ghi data hiển thị LCD
;ghi mã ASCII ký tự ra LCD
;tiếp tục hiển thị dòng 2
;chờ 100µs

WAIT:  LDI   R18,80
AG1:   LDI   R16,250
RCALL DELAY_US
DEC   R18
BRNE AG1
LDI   R19,14
LDI   R17,$18
CBI   LCD,RS
RCALL OUT_LCD4_2
;lập vòng dịch trái màn hình
;mã lệnh dịch trái màn hình, DDRAM giữ nguyên
;RS=0 ghi lệnh
;lập vòng chờ 10x25=0.25s
;chờ 25ms

WAIT2: LDI   R18,10
AG2:   LDI   R16,250
RCALL DELAY_US
DEC   R18
BRNE AG2
DEC   R19
BRNE WAIT2
LDI   R19,30
;lập vòng dịch phải màn hình
;mã lệnh dịch phải màn hình
WAIT3: LDI   R17,$1C

```

```

CBI    LCD,RS           ;RS=0 ghi lệnh
RCALL OUT_LCD4_2
LDI    R18,10
AG3:  LDI    R16,250
      RCALL DELAY_US
      DEC   R18
      BRNE AG3
      DEC   R19
      BRNE WAIT3
      RJMP  START

;-----;
;INIT_LCD4 khởi động LCD ghi 4 byte mã lệnh theo giao tiếp 4 bit
;Function set:R18=$28 2 dòng font 5x8 giao tiếp 4 bit
;Clear display:R19=$01 xóa màn hình
;Display on/off LCDrol:R20=$0C màn hình on,con trỏ off
;Entry mode set:R21=$06 dịch phái con trỏ ,đ/c DDRAM tăng 1 khi ghi data
;RS=bit0=0,RW=bit1=0
;-----;

INIT_LCD4: CBI    LCD,RS           ;RS=0: ghi lệnh
            MOV    R17,R18          ;R18=Function set
            RCALL OUT_LCD4_2        ;ghi 1 byte data ra LCD
            MOV    R17,R19          ;R19=Clear display
            RCALL OUT_LCD4_2        ;chờ 2ms sau lệnh Clear display
            LDI    R16,20          ;R20=Display LCDrol on/off
            RCALL DELAY_US
            MOV    R17,R20
            RCALL OUT_LCD4_2
            MOV    R17,R21          ;R21=Entry mode set
            RCALL OUT_LCD4_2
            RET

;-----;
;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

;-----;
;OUT_LCD4_2 ghi 1 byte mã lệnh/data ra LCD
;chia làm 2 lần ghi 4bit
;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: LDI   R16,1           ;chờ 100us
             RCALL DELAY_US
             IN    R16,LCD          ;đọc PORT LCD
             ANDI R16,(1<<RS)     ;lọc bit RS
             PUSH R16               ;cắt R16
             PUSH R17               ;cắt 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
POP R17          ;phục hồi R17
POP R16          ;phục hồi R16
SWAP R17          ;đảo 4 bit
ANDI R17,$F0      ;lấy 4 bit thấp chuyển thành cao
OR R17,R16        ;ghép bit RS
RCALL OUT_LCD4    ;ghi ra LCD
RET

```

;-----
;DELAY_US tạo thời gian trễ =R16x100μs(Fosc=8Mhz)
;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

- Về cơ bản giải thuật chương trình tương tự như giao tiếp 8 bit ở ví dụ 6.16, chỉ khác ở các phần reset cấp nguồn, lệnh function set và chương trình con xuất mã lệnh/data ra LCD
- Phần reset cấp nguồn giao tiếp 4 bit như hình 6.31b theo trình tự chờ 50ms, ghi mã lệnh \$3, chờ 4.2ms, ghi mã lệnh \$3, chờ 200μs, ghi mã lệnh \$3, chờ 100μs và mã lệnh \$2.
- Chương trình con INIT_LCD4 khởi động LCD với lệnh function set bằng mã lệnh \$28 đặt cấu hình 4 bit, các lệnh còn lại giống như ví dụ 6.16.
- Chương trình con OUT_LCD4 2 tách byte mã lệnh/data ra 2 nhóm 4 bit, mỗi nhóm 4 bit được đưa lên trọng số cao và ghép với 3 bit điều khiển E,RW,RS ở trọng số thấp, xuất nối tiếp ra port, với nhóm 4 bit cao truyền trước.

❖ Chương trình C VD6-17

```

#include <avr/io.h>
#define lcd PORTC //định nghĩa lcd=PORTC
#define lcd_dr DDRC //định nghĩa lcd_dr=DDRC
#define lcd_in PINC //định nghĩa lcd_in=PINC
const char RS=0,RW=1,E=2 ;//ký hiệu vị trí bit RS,RW,E
const char CR=0xd ;//ký hiệu mã xuống dòng
const char NULL=0x00 ;//ký hiệu mã kết thúc
void OUT_LCD4(unsigned char val) ;//khai báo hàm ghi ra LCD
void OUT_LCD4_2(unsigned char val) ;//khai báo hàm ghi mã lệnh/data ra LCD
void INIT_LCD4(char dat1,char dat2,char dat3,char dat4); //khai báo hàm khởi động LCD
void delay_ms(unsigned int n) ;//khai báo hàm delay ms
unsigned char TAB[]="Xin chao!\rMon Vi xu ly\x0"; //\r,\x0 ký hiệu mã ASCII $0D,$00
unsigned char i,n,tam,RS_val;
int main()
{
    lcd_dr=0xff ;//port lcd output
    lcd=0xf0 ;// E=0,RW=0,RS=0,
}

```

```

delay_ms(33333)      ;//delay 25ms
delay_ms(33333)      ;//delay 25ms
lcd&=~(1<<RS)      ;//truy xuất lệnh RS=0
tam=0x30              ;//mã lệnh $30 lần 1
OUT_LCD4(tam)        ;
delay_ms(5600)        ;//delay 4.2ms
lcd&=~(1<<RS)      ;//truy xuất lệnh RS=0
tam=0x30              ;//mã lệnh $30 lần 2
OUT_LCD4(tam)        ;
delay_ms(266)         ;//delay 0.2ms
tam=0x32              ;//mã lệnh $32
lcd&=~(1<<RS)      ;//truy xuất lệnh RS=0
OUT_LCD4_2(tam)      ;
INIT_LCD4(0x28,0x01,0x0c,0x06);//khởi động LCD theo mode như trên
while(1)
{
    delay_ms(133)      ;//delay 0.1ms
    tam=0x01            ;//mã lệnh xóa màn hình
    lcd&=~(1<<RS)      ;//truy xuất lệnh RS=0
    OUT_LCD4_2(tam)    ;
    delay_ms(2667)     ;//delay 2ms sau lệnh clear display
    tam=0x83            ;//mã lệnh dời con trỏ đến vị trí 4 hàng 1
    lcd&=~(1<<RS)      ;//truy xuất lệnh RS=0
    OUT_LCD4_2(tam)    ;
    i=0                ;//chỉ số dây ký tự TAB
    tam=TAB[i]          ;//lấy mã ASCII từ bảng tra
    while(tam!=CR)      //ký tự CR?
    {
        lcd |=(1<<RS) ;//RS=1 truy xuất data
        OUT_LCD4_2(tam) ;//ghi ký tự hiển thị ra LCD
        i++               ;//ký tự kế tiếp
        tam=TAB[i]          ;//lấy mã ASCII từ bảng tra
    }
    tam=0xc2            ;//mã lệnh dời con trỏ đến vị trí 3 hàng 2
    lcd&=~(1<<RS)      ;//truy xuất lệnh RS=0
    OUT_LCD4_2(tam)    ;
    i++               ;//trỏ đến ký tự đầu dòng 2
    tam=TAB[i]          ;//lấy mã ASCII từ bảng tra
    while(tam!=NULL)
    {
        lcd |=(1<<RS) ;//RS=1 truy xuất data
        OUT_LCD4_2(tam) ;//ghi ký tự hiển thị ra LCD
        i++               ;
        tam=TAB[i]          ;//lấy mã ASCII từ bảng tra
    }
    for(i=1;i<=50;i++)   //lặp vòng delay 50x40ms=2s
        delay_ms(53333); //delay 40ms
    for(n=1;n<=14;n++)  //lặp vòng số lần dịch trái
    {
        tam =0x18          ;//dịch trái màn hình,DDRAM giữ nguyên
        lcd&=~(1<<RS)      ;//truy xuất lệnh RS=0
        OUT_LCD4_2(tam)    ;
        for(i=1;i<=10;i++)  //delay 10x25ms=250ms
            delay_ms(33333); //delay 25ms
    }
}

```

```

        }
        for(n=1;n<=30;n++) //lập vòng số lần dịch phải
        {
            tam =0x1C ;//dịch phải màn hình,DDRAM giữ nguyên
            lcd&=~(1<<RS) ;//truy xuất lệnh RS=0
            OUT_LCD4_2(tam) ;
            for(i=1;i<=10;i++) //delay 10x25ms=250ms
                delay_ms(33333); //delay 25ms
        }
    }
}

//-----
void INIT_LCD4(char dat1,char dat2,char dat3,char dat4)//hàm khởi động LCD 4 bit
{
    lcd&=~(1<<RS) ;//truy xuất lệnh RS=0
    tam=dat1 ;//mã lệnh function set
    OUT_LCD4_2(tam) ;
    tam=dat2 ;//mã lệnh clear display
    OUT_LCD4_2(tam) ;
    delay_ms(2667) ;//delay 2ms sau mã lệnh Clear
    tam=dat3 ;//mã lệnh display control on/off
    OUT_LCD4_2(tam) ;
    tam=dat4 ;//mã lệnh entry mode set
    OUT_LCD4_2(tam) ;
}
//-----
void OUT_LCD4(unsigned char val)
{
    lcd=val;
    lcd |=(1<<E);
    lcd&=~(1<<E);
}
//-----
void OUT_LCD4_2(unsigned char chr_val)
{
    unsigned char chr1,chr2;
    delay_ms(133) ;
    chr1=lcd&(1<<RS) ;//lọc lấy bit RS
    chr2=chr_val&0xf0 ;//lấy 4 bit cao
    chr2 |=chr1 ;//ghép bit RS
    OUT_LCD4(chr2);//truy xuất bus LCD
    delay_ms(133) ;
    chr2=chr_val<<4;//dịch lên 4 bit cao
    chr2&=0xf0 ;//lấy 4 bit cao
    chr2 |=chr1 ;//ghép bit RS
    OUT_LCD4(chr2);//truy xuất bus LCD
}
//-----
void delay_ms(unsigned int n)// ctc delay ms
{
    unsigned int k;
    for(k=0;k<=n;k++); // Td=6xn MC
}

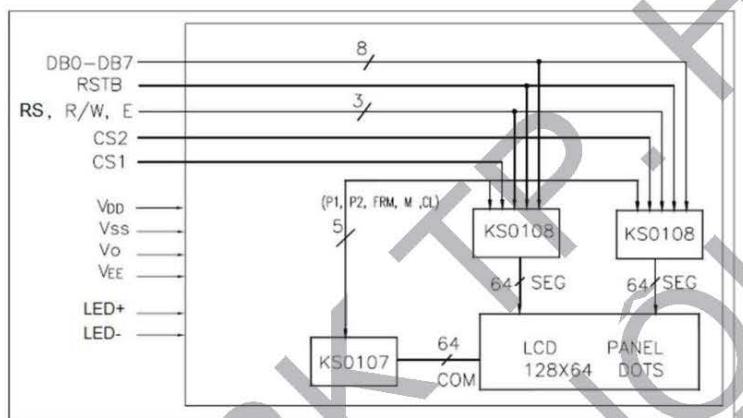
```

❖ Câu hỏi ôn tập

- Ghi lại các mã lệnh khởi động LCD ký tự làm việc với cấu hình 1 dòng, font 5x10, giao tiếp 4 bit, hiện con trỏ không chớp và con trỏ dịch phải khi ghi ký tự
- Trong ví dụ 6.16 thay vì sử dụng các chương trình con delay để chờ LCD rồi, ta kiểm tra cờ báo bận của LCD. Viết chương trình con đọc cờ báo bận từ LCD.
- Từ chương trình con OUT_LCD ví dụ 6.16, vẽ giản đồ xung các tín hiệu RS, RW, E data và tính các thông số thời gian theo như hình 6.34, so sánh với data sheet.
- Áp dụng các chương trình con trong ví dụ 6.16 và sơ đồ hình 6.35, viết một đoạn chương trình ghi ký tự số 9 ra LCD ký tự tại vị trí thứ 5 dòng 2 (giả sử LCD đã được khởi động cấu hình)
- Áp dụng các chương trình con trong ví dụ 6.17 và sơ đồ hình 6.36, viết một đoạn chương trình ghi ký tự A ra LCD ký tự tại vị trí thứ 8 dòng 1 (giả sử LCD đã được khởi động cấu hình)

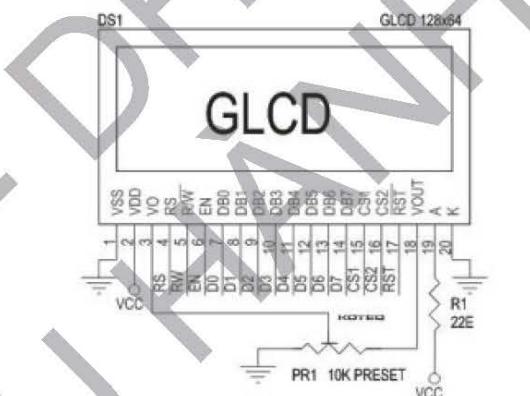
6.5.2 Giao tiếp Đèn LCD đồ họa(Graphical LCD):

Hình 6.37 là sơ đồ cấu trúc 1 panel LCD đồ họa 128x64 điểm, sử dụng mạch lái IC KS0108.



Hình 6.37: Sơ đồ cấu trúc panel LCD đồ họa 128x64

Sơ đồ chân panel thông dụng (*) được mô tả như hình 6.38 và Bảng 6.11 mô tả chức năng các chân.



Hình 6.38: Sơ đồ chân panel LCD đồ họa(GLCD) 128x64

Bảng 6.11: Chức năng các chân panel LCD 128x64 hình 6.38

Chân số	Ký hiệu	Kết nối ngoài(I/O)	Mô tả chức năng
1	V _{SS}	Nguồn	GND
2	V _{DD}	Nguồn	Nguồn logic (+5V)
3	V ₀	Nguồn điều chỉnh	Nguồn tạo độ tương phản ($\approx -3.3V$)(**)
4	RS(D/I)	MCU / (I)	Chọn thanh ghi: RS=1; data, RS=0: lệnh
5	R/W	MCU / (I)	Tín hiệu đọc/ghi: R/W=1 đọc, R/W=0: ghi
6	E(EN)	MCU / (I)	Tín hiệu cho phép làm việc, kích cảnh xuống
7 - 14	DB0 - DB7	MCU / (I/O)	Bus data 2 chiều
15	CS1	MCU / (I)	Tín hiệu chọn chíp tích cực thấp, chọn $\frac{1}{2}$ màn hình trái
16	CS2	MCU / (I)	Tín hiệu chọn chíp tích cực thấp, chọn $\frac{1}{2}$ màn hình phải
17	/RST	MCU / (I)	Tín hiệu reset tích cực thấp
18	V _{EE}	Nguồn	Nguồn âm lái LCD (-3÷-10V) (**)

19	LED+	Nguồn	Anode đèn nền(+5V)
20	LED-	Nguồn	Cathode đèn nền GND

(*) Các panel GLCD khác có thể chân ra khác với cùng ký hiệu chức năng chân

(**) Nguồn âm VEE được tạo ra từ bên trong mạch phân cực chân V_O qua VR 10K như hình 6.38

1. Tổ chức địa chỉ DDRAM truy xuất điểm ảnh LCD đồ họa

Gồm 2 IC KS0108 lái 128x64 điểm ảnh(xem chi tiết data sheet IC KS0108)

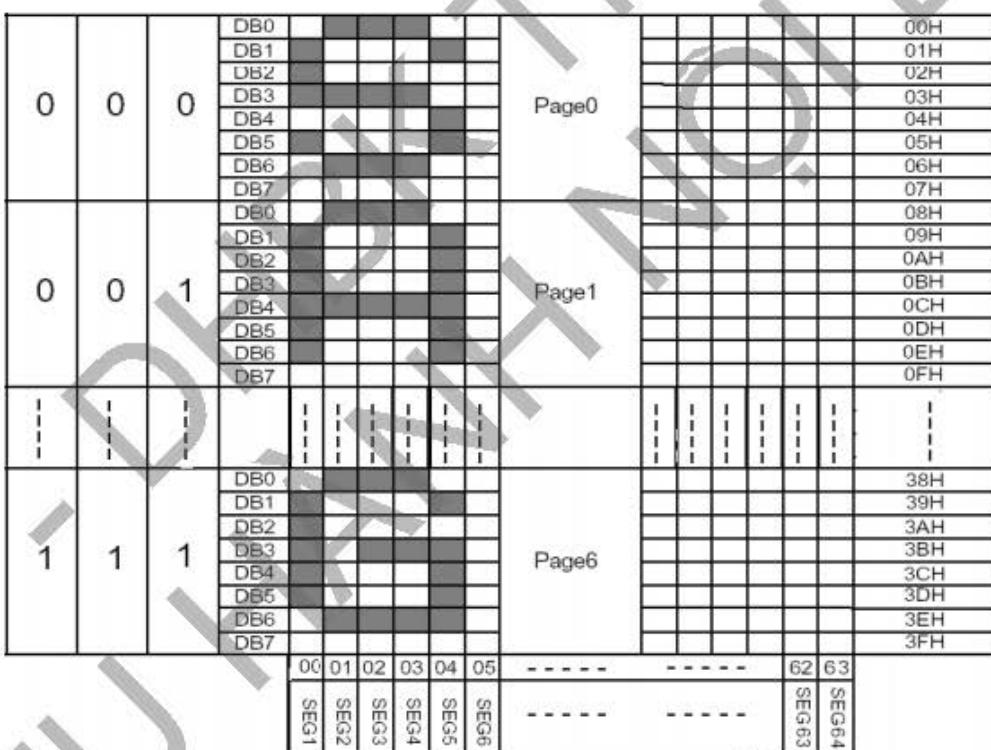
- Mỗi IC KS0108 lái 64x64 điểm ảnh=4096 điểm ảnh, tổ chức địa chỉ DDRAM lái điểm ảnh chia thành trang/cột như hình 6.39

- Có 8 trang gán địa chỉ từ 000 – 111(trang 0 – 7)
- Mỗi trang có 8 hàng gán địa chỉ từ 00H -3FH(64 hàng)
- Data ứng với mỗi điểm ảnh gán theo trọng số 8 hàng địa chỉ từ thấp đến cao trong 1 trang
Ví dụ trang 0 gán data DB0-DB7 trong ứng hàng địa chỉ từ 00H-07H
- 64 cột gán địa chỉ từ 00H-3FH điều khiển chung cho cả 8 trang

Như vậy để truy xuất 1 điểm ảnh, phải xác định địa chỉ hàng/cột và bit data được gán tương ứng!

Truy xuất 1 byte data chỉ cần quan tâm địa chỉ trang và cột.

Ví dụ để hiển thị vạch thẳng đứng đầu tiên chữ A như hình 6.39 ở đầu hàng trang 1, xuất data=7EH ra bus data LCD (mức 1=sáng, mức 0=tối) ghi vào DDRAM tại địa chỉ trang 1 là 001 và cột địa chỉ 00H



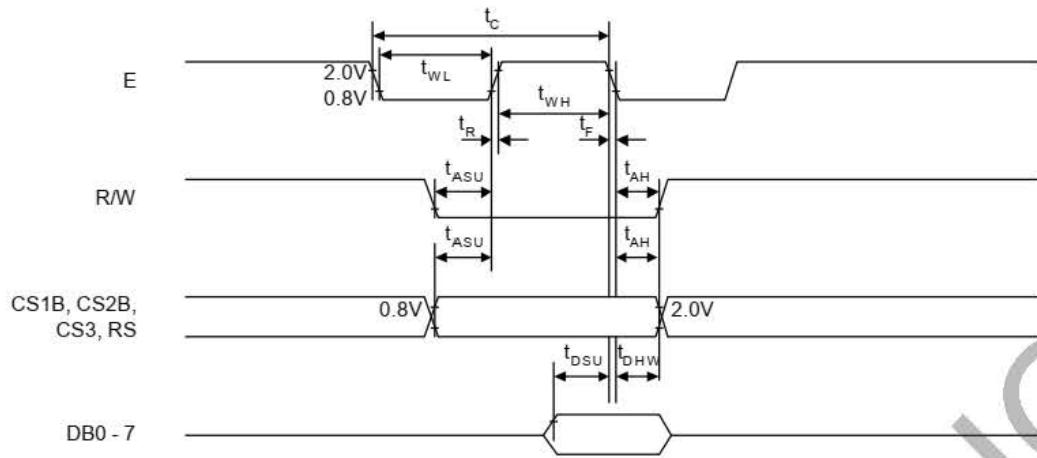
Hình 6.39: Tổ chức địa chỉ DDRAM theo trang,dòng,cột lái điểm ảnh

2. Truy xuất data

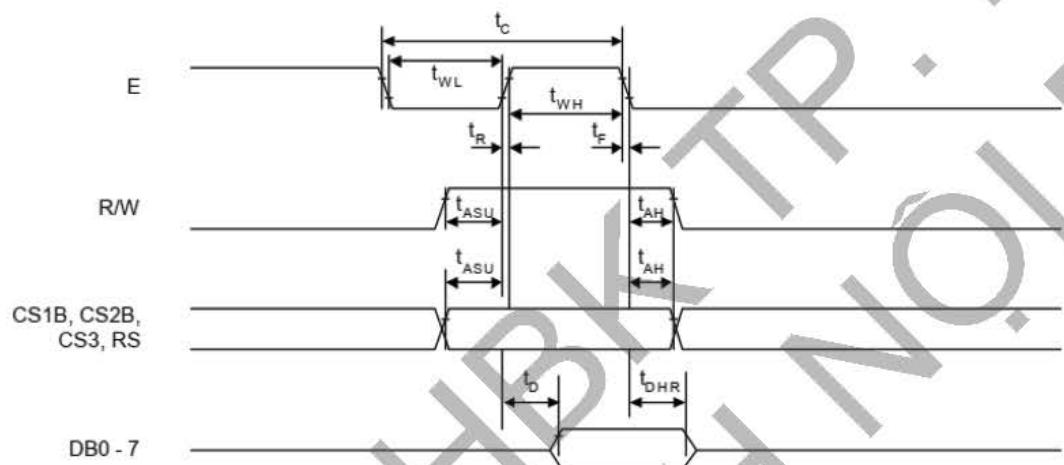
Hình 6.40a là giàn đồ xung truy xuất ghi data ra GLCD

Ta thấy hoạt động các chân điều khiển tương tự như LCD ký tự.

- Tín hiệu R/W mức thấp báo ghi
- Tín hiệu RS báo ghi lệnh(RS=0) hay ghi data(RS=1)
- Tín hiệu CS1/CS2 tích cực mức 0 chọn truy xuất ½ màn hình trái/phải
- Đặt data lên DB0-DB7
- Tín hiệu E tích cực cạnh xuống ghi data vào GLCD



(a) Giản đồ xung định thì ghi GLCD



(b) Giản đồ xung định thì đọc GLCD

Hình 6.40: Giản đồ xung định thì ghi/đọc GLCD

Hình 6.40b là giản đồ xung định thì đọc GLCD, trình tự tương tự như ghi chỉ khác xung R/W=1. Khi tín hiệu E tích cực cạnh xuống GLCD sẽ xuất data DB0..DB7 ra bus data cho phép đọc về.

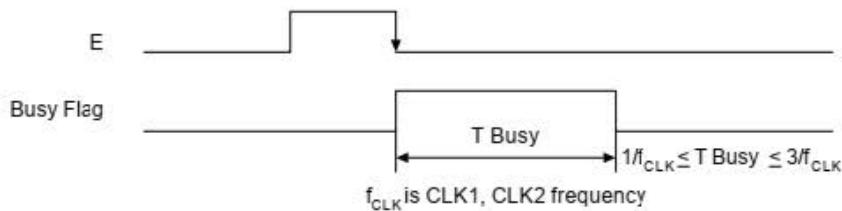
Ta lưu ý các thông số thời gian từ hình 6.40 như sau:

- Chu kỳ E(chu kỳ cho phép GLCD) $t_{Cmin}=1000ns$
- Độ rộng xung E mức cao $t_{WHmin}=450ns$
- Độ rộng xung E mức thấp $t_{WLmin}=450ns$
- Thời gian đặt địa chỉ(từ lúc đặt CS1,CS2,RS,R/W đến cạnh lên của E) $t_{ASUmin}=140ns$
- Thời gian giữ địa chỉ(từ lúc cạnh xuống của E đến khi CS1,CS2,RS,R/W thay đổi) $t_{AHmin}=10ns$
- Thời gian đặt data(ghi)(từ lúc đặt data ghi đến cạnh xuống của E) $t_{DSUmin}=200ns$
- Thời gian giữ data(ghi)(từ lúc cạnh xuống của E đến khi data thay đổi) $t_{DHWmin}=10ns$
- Thời gian trễ data(đọc)(từ lúc cạnh lên của E đến khi data xuất hiện) $t_{Dmax}=320ns$
- Thời gian giữ data(đọc)(từ lúc cạnh xuống của E đến khi data thay đổi) $t_{DHRmin}=20ns$

❖ Từ các thông số thời gian trên, ta lưu ý tính thời gian xuất các tín hiệu điều khiển theo chu kỳ máy MC như sau(Fosc=8Mhz, 1MC=125ns):

- Thời gian giữa 2 lần truy xuất GLCD $> 1000ns \# 8MC$
- Thời gian tín hiệu E mức 1/mức 0 $> 450ns \# 4MC$
- Thời gian từ lúc đặt các tín hiệu CS1,CS2,RS,R/W đến khi E chuyển từ 0 lên 1 $> 140ns \# 2MC$
- Thời gian đặt data ghi ra bus data GLCD đến khi E chuyển từ 0 lên 1 $> 200ns \# 2MC$
- Thời gian chờ đọc data từ lúc E chuyển từ 0 lên 1 $> 320ns \# 3MC$

Hình 6.41 minh họa thời gian chờ trong khi GLCD bận.



Hình 6.41: Giản đồ xung định thời báo bận của GLCD

Ngay sau khi có cạnh xuống ở chân E, GLCD đặt cờ báo bận DB7=1 trong thời gian T_{busy} để thực hiện công việc. Sau thời gian tối đa khoảng $3T_{CLK}$, LCD xóa cờ báo bận báo sẵn sàng giao tiếp bên ngoài trở lại.

Với $f_{CLK}=300\text{KHz} \rightarrow T_{busy}=3T_{CLK} \approx 10\mu\text{s}$, f_{CLK} là tần số xung clock nội của GLCD.

3. Reset

- Tín hiệu reset tích cực mức thấp thời gian tối thiểu $1\mu\text{s}$
- Sau khi reset màn hình bị xóa, địa chỉ hiển thị chuyển về đầu màn hình trái (điểm ảnh dòng 0, cột 0). Khi chân RST=0 chỉ có thể truy xuất lệnh đọc trạng thái để kiểm tra DB7=bận, DB4=reset (xem Bảng 6.12)
- Có thể reset bằng mềm qua các lệnh điều khiển, chân RESET kết nối mức 1!

4. Tập lệnh giao tiếp LCD đồ họa

Tập lệnh giao tiếp LCD đồ họa tóm tắt trong Bảng 6.12.

Bảng 6.12: Tóm tắt các lệnh giao tiếp GLCD

LỆNH	RS	R/W	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0	Chức năng
Màn hình mở/tắt	0	0	0	0	1	1	1	1	1	0/1	Điều khiển màn hình mở/tắt, trạng thái trong và data DDRAM không đổi DB0=0/1: tắt/mở
Đặt địa chỉ cột (địa chỉ Y)	0	0	0	1	Địa chỉ Y(0 - 63)						Đặt địa chỉ Y trong bộ đếm địa chỉ Y
Đặt trang (địa chỉ X)	0	0	1	0	1	1	1	Trang (0 - 7)			Đặt địa chỉ X trong thanh ghi địa chỉ X
Hiển thị đầu hàng (địa chỉ Z)	0	0	1	1	Hiển thị đầu hàng(0 - 63)						Chỉ báo DDRAM hiển thị ở đầu màn hình
Đọc trạng thái	0	1	Bận	0	on/off	Reset	0	0	0	0	Trạng thái đọc : Bận: 0/1=rồi/bận on/off: 0/1=màn hình mở/tắt Reset: 0/1=bình thường/reset
Ghi data hiển thị	1	0	Data ghi								Ghi data DB0..DB7 ra DDRAM, sau lệnh ghi địa chỉ Y tự động tăng thêm 1
Đọc data hiển thị	1	1	Data đọc								Đọc data DB0..DB7 từ DDRAM ra bus data

Từ bảng Bảng 6.12, ta rút ra các lệnh thường sử dụng giao tiếp GLCD như sau:

- ❖ **Truy xuất lệnh:** RS=0, CS1=0 truy xuất 1/2 màn hình trái, CS2=0 truy xuất 1/2 màn hình phải

Mã lệnh	Chức năng	Truy xuất
\$3E	Tắt màn hình	Ghi R/W=0
\$3F	Mở màn hình	Ghi R/W=0
\$40 – \$7F	Đặt địa chỉ cột(Y) từ 0 đến 63	Ghi R/W=0
\$B8 – \$BF	Đặt địa chỉ trang(X) từ 0 đến 7	Ghi R/W=0
\$C0 – \$FF	Đặt địa chỉ đầu hàng từ 0 đến 63	Ghi R/W=0

- ❖ **Truy xuất data:** RS=1,CS1=0 truy xuất 1/2 màn hình trái,CS2=0 truy xuất 1/2 màn hình phải

R/W	DB7 – DB0	Truy xuất
0	Ghi data ra DDRAM,sau mỗi lệnh ghi địa chỉ cột(Y) tăng thêm 1	Ghi data
1	Đọc data từ DDRAM ra bus data	Đọc data

- ❖ **Đọc trạng thái LCD:** RS=0,R/W=1, CS1=0:truy xuất ½ màn hình trái,CS2=0: truy xuất ½ màn hình phải

DB7=BẬN DB7=1: GLCD bận, DB7=0: GLCD rỗi

DB5=on/off DB5=1: tắt màn hình, DB5=0: mở màn hình

DB4=RESET DB4=1: reset, DB4=0: bình thường

➤ Sau khi ghi lệnh/data ra LCD chờ LCD báo rỗi bằng một trong 2 cách sau:

- Đọc trạng thái LCD như trên, kiểm tra DB7=0 là LCD rỗi sẵn sàng giao tiếp

- Delay khoảng 10µs tương ứng thời gian dài nhất của T_{busy}(F_{CLK}=300KHz, tần số CK nội GLCD)

5. Thiết lập bộ mã font ký tự cho GLCD

Trong mục 6.4.2 đã trình bày phương pháp thiết lập bộ mã font ký tự cho LED ma trận 8x8, kết quả như Bảng 6.8. Ta chọn ma trận 8x8 điểm ảnh để hiển thị 1 ký tự, điểm ảnh sáng mức 1, tối mức 0.

Như vậy với tổ chức địa chỉ theo 8 trang(8 hàng/trang) và 64 cột của DDRAM, hiển thị 1 ký tự cần 1 trang và 8 cột, toàn màn hình sẽ hiển thị được 8 hàng ký tự, mỗi hàng tối đa 16 ký tự(1/2 màn hình trái 8 ký tự, 1/2 màn hình phải 8 ký tự)

HÀNG	\$00	\$7E	\$11	\$11	\$11	\$7E	\$00	\$00
TRANG	D0							
CỘT	0	1	2	3	4	5	6	7
D0								
D1								
D2								
D3								
D4								
D5								
D6								
D7								

Hình 6.41: Thiết lập bộ mã font ký tự A theo địa chỉ trang và cột

Với cách chọn trên, ta có thể áp dụng Bảng 6.8 làm bộ mã font ký tự hiển thị ra GLCD.

- ❖ **Giải thuật hiển thị 1 ký tự**

- Từ mã ASCII ký tự, tra bảng suy ra mã font ký tự tương ứng gồm 8 byte

- Đặt địa chỉ trang (X) muốn hiển thị ký tự (từ 0 đến 7), mã lệnh \$B8 – \$BF

- Đặt địa chỉ cột (Y) bắt đầu hiển thị ký tự = 8n, với n=0 đến 7, mã lệnh \$40 – \$7F

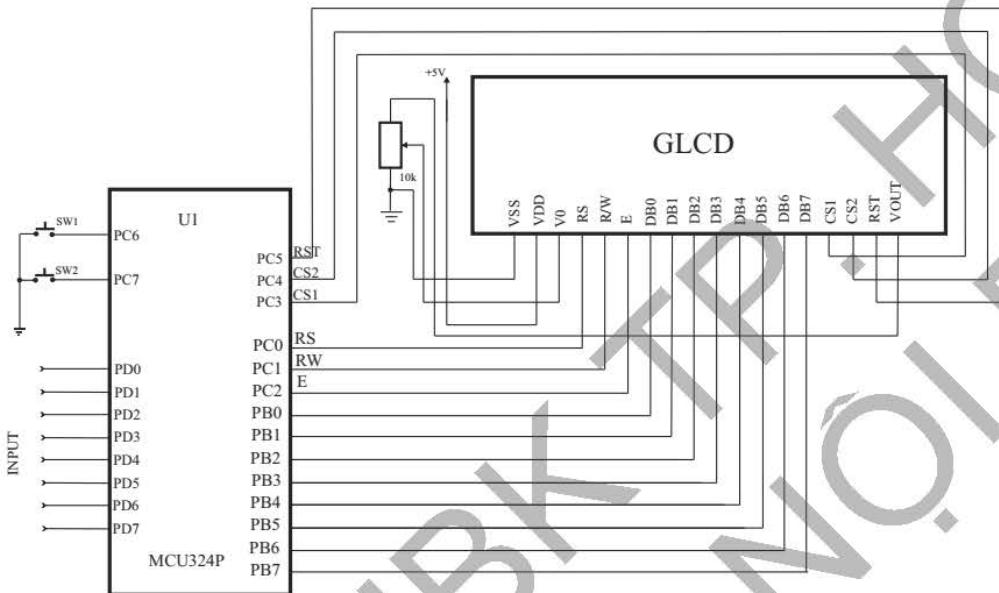
- Lần lượt ghi 8 byte data là mã font ký tự ra bus data (địa chỉ cột tự động tăng 1 sau mỗi lần ghi)

- Trước mỗi lần ghi data, đọc data từ GLCD kiểm tra bit D7 hoặc chờ tối thiểu 10µs chờ GLCD rỗi mới thực hiện ghi data.

Ví dụ 6.18: Hình 6.42 là sơ đồ MCU324P giao tiếp trực tiếp với GLCD, mô tả như sau:

- PORTD nhập data đặt từ DIPSW8 là mã ASCII từ \$20 - \$7F để hiển thị ký tự tương ứng ra GLCD
- SW1=PC6 khi nhấn=0, cho phép nhập data từ PORTD

- SW2=PC7=CR(mã ASCII=\$0D), mã xuống dòng(theo dòng ký tự)
- PORTB kết nối bus data DB0 - DB7 của GLCD
- Các chân điều khiển GLCD: RS=PC0,R/W=PC1,E=PC2,/CS1=PC3,/CS2=PC4,RST=PC5
- Viết một chương trình hiển thị ký tự là mã ASCII từ \$20 - \$7F ra GLCD, thực hiện như sau:
 - Ký tự được đặt từ DIPSW8 và nhập từ PORTD khi nhấn nút SW1
 - Xuống dòng ký tự mới khi hiển thị đủ 16 ký tự ở dòng trên hoặc nhấn nút SW2
 - Nếu ở dòng cuối(dòng 8) hiển thị đủ 16 ký tự, nhập tiếp ký tự mới màn hình sẽ bị xóa và hiển thị ký tự mới ở đầu dòng 1
 - Nếu ở dòng cuối, nhấn nút SW2 màn hình sẽ bị xóa và trở về hiển thị từ đầu dòng 1
 - Luôn hiển thị con trỏ tại vị trí chờ hiển thị ký tự



Hình 6.42: Giao tiếp trực tiếp GLCD

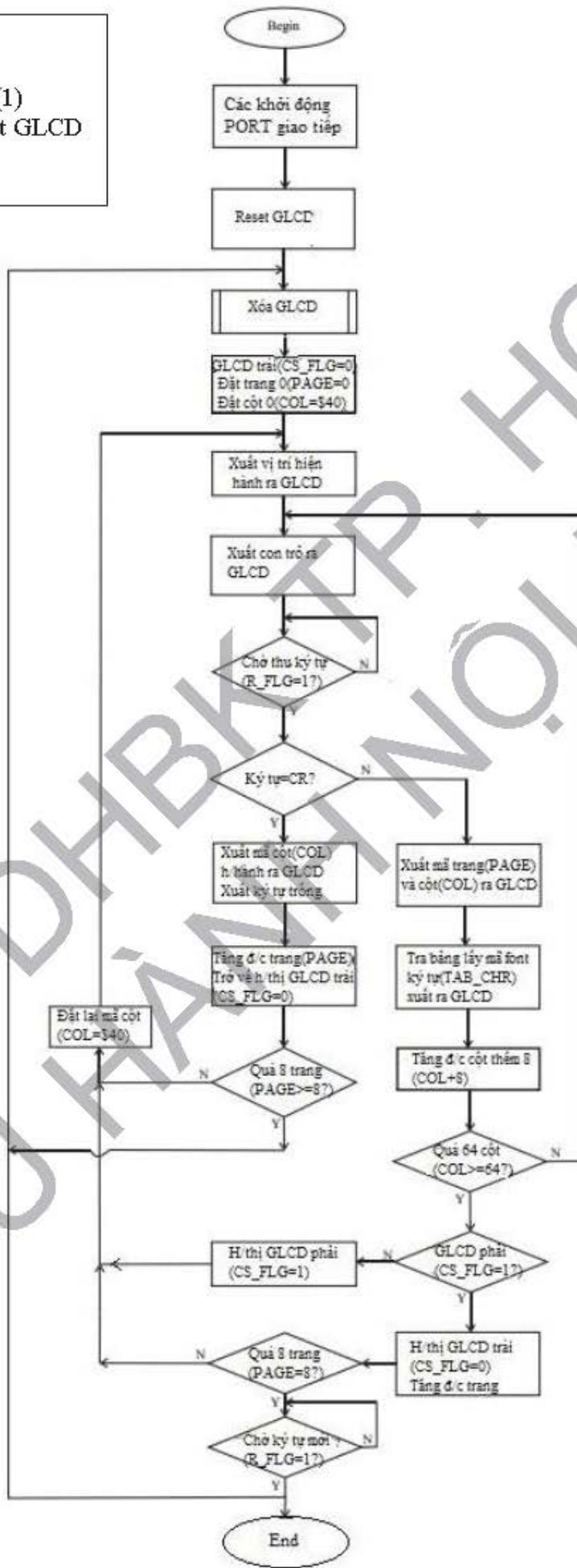
Giải:

- Tóm tắt ý tưởng giải thuật chương trình ví dụ 6.18:
 - Khởi động reset GLCD, gọi chương trình con CLEAR xóa tất cả DDRAM
 - Hiển thị con trỏ tại đầu dòng 1 màn hình trái, chờ thu ký tự
 - Nếu ký tự là mã hiển thị từ \$20 - \$7F, chuyển đổi ra mã ký tự tra bảng và ghi 8 byte data bộ mã font ký tự tương ứng ra GLCD, tăng bộ đếm cột 8 đơn vị, hiển thị con trỏ tại vị trí kế tiếp(lưu ý là sau mỗi lần xuất data ra LCD, bộ đếm địa chỉ cột (Y)DDRAM của LCD tự động tăng 1)
 - Nếu hiển thị đủ 8 ký tự trên 1 dòng của màn hình trái, chuyển qua hiển thị tiếp cùng dòng với màn hình phải tương tự như trên
 - Nếu hiển thị đủ 16 ký tự trên 1 dòng, chuyển con trỏ xuống đầu dòng kế tiếp ở màn hình trái(tăng trang)
 - Nếu thu ký tự CR, chuyển con trỏ xuống đầu dòng kế tiếp như trên
 - Trường hợp ở dòng thứ 8, nếu hiển thị đủ 16 ký tự, nhập thêm ký tự mới sẽ xóa màn hình quay về hiển thị ký tự mới tại đầu dòng 1 ở màn hình trái.
 - Trường hợp ở dòng thứ 8, nếu thu ký tự CR sẽ xóa màn hình, quay về đặt con trỏ tại đầu dòng 1 ở màn hình trái.
- Sử dụng chương trình con WAIT delay khoảng 10μs chờ LCD rồi!

Hình 6.43 trình bày lưu đồ chi tiết giải thuật chương trình ví dụ 6.18.



- PAGE= địa chỉ trang
 - COL= địa chỉ cột
 - R_FLG= cờ báo thu ký tự(1)
 - CS_FLG= cờ báo truy xuất GLCD trái/phải(0/1)



Hình 6.43: Lưu đồ giải thuật chi tiết ví dụ 6.18

❖ Chương trình hợp ngữ VD6.18

```

.INCLUDE <M324PDEF.INC>
.EQU FONT_TAB=0X200 ;đặt địa chỉ đầu bảng tra mã font ký tự trong Flash ROM
.INCLUDE "FONT_CHR.INC"; gọi file mã font ký tự
.DEF COUNT=R18           ;COUNT=bộ đếm
.DEF BUF_DAT=R19          ;BUF_DAT=chứa mã lệnh/data
.DEF FLG_REG=R20          ;FLG_REG=chứa các cờ báo
.DEF PAGE_P=R21          ;PAGE_P=vị trí địa chỉ trang
.DEF COL=R22              ;COL=địa chỉ cột
.EQU OUTPORT=PORTB        ;PORTB data bus GLCD
.EQU INPORT=PINB
.EQU IOSETB=DDRB
.EQU CONT=PORTC           ;PORTC điều khiển
.EQU CONT_DR=DDRC
.EQU CONT_OUT=PORTC
.EQU CONT_IN=PINC
.EQU DAT=PORTD             ;PORTD data input
.EQU IOSETD=DDRD
.EQU DAT_IN=PIND
.EQU RS=0                  ;RS=bit0
.EQU RW=1                  ;RW=bit1
.EQU E=2                   ;E=bit2
.EQU CS1=3                 ;CS1=bit3
.EQU CS2=4                 ;CS2=bit4
.EQU RST=5                 ;RST=bit5
.EQU SW1=6                 ;SW1=bit6 nhập ký tự
.EQU SW2=7                 ;SW2=bit7 CR xuống dòng
.EQU R_FLG=0                ;cờ báo thu ký tự bit0 thuộc FLG_REG
.EQU CS_FLG=1               ;cờ báo chọn 1/2 màn hình bit1 thuộc FLG_REG
.EQU CR=$0D                 ;mã xuống dòng
.EQU NULL=$00               ;mã kết thúc
.EQU CURSOR=$80             ;mã con trỏ
.ORG 0
.RJMP MAIN
.ORG 0X40
MAIN: LDI R16,HIGH(RAMEND)    ;đưa stack lên vùng đ/c cao
OUT SPH,R16
LDI R16,LOW(RAMEND)
OUT SPL,R16
LDI R16,0X3F
OUT CONT_DR,R16            ;PC0..PC5 output,PC6,PC7 input
LDI R16,$D8
OUT CONT,R16               ;PC0=RS=0,PC1=R/W=0 ghi,PC2=E=0 cấm GLCD
                           ;PC3=CS1=1,PC4=CS2=1 không chọn màn hình
                           ;PC5=RST=0 reset GLCD,PC6=PC7=1 đặt R kéo lên
                           ;chờ reset 10μs(>1μs)
                           ;kết thúc reset
RCALL WAIT
SBI CONT,RST
LDI R16,0XFF
OUT IOSETB,R16             ;khai báo outport
LDI R16,0X00
OUT IOSETD,R16             ;khai báo data port input
LDI R16,0XFF
OUT DAT,R16                ;R kéo lên port input
START: CBR FLG_REG,(1<<CS_FLG);CS_FLG=bit1=0 báo h/thị màn hình trái
       RCALL CLEAR             ;xóa toàn bộ màn hình và DDRAM

```

```

LDI PAGE_P,0 ;PAGE_P=R21 địa chỉ trang bắt đầu
LDI COL,$40 ;COL=R22 địa chỉ cột bắt đầu
;-----;
;Xử lý hiển thị con trỏ ra GLCD
;-----;
START1: LDI BUF_DAT,CURSOR;nạp ký tự con trỏ
        RCALL OUT_CHR ;hiển thị ký tự ra GLCD
;-----;
;Xử lý chờ đọc ký tự ctc GET_CHR trả về
;mã ký tự trong BUF_DAT và R_FLG=1 nếu có SW nhấn
;-----;
WAIT_CHR: RCALL GET_CHR ;đọc ký tự
          SBRS FLG_REG,R_FLG ;R_FLG=1 bỏ qua lệnh kế tiếp
          RJMP WAIT_CHR ;R_FLG=0 chờ đọc ký tự
          CPI BUF_DAT,CR ;data đọc vào=CR?
          BREQ LINE_FD ;ký tự CR xuống dòng
;-----;
;Xử lý hiển thị ký tự ra GLCD,BUF_DAT chứa mã ký tự đọc vào
;-----;
          CPI BUF_DAT,$20 ;ký tự đọc vào phải thỏa $20<=R19<=$7F
          BRCS START1 ;không thỏa trả về chờ đọc ký tự mới
          CPI BUF_DAT,$80 ;hiển thị ký tự ra GLCD
          BRCC START1 ;tăng cột đến vị trí hiển thị kế tiếp
          RCALL OUT_CHR ;hết 64 cột?
          LDI R17,8 ;chuyển sang 1/2 màn hình phải
          ADD R17, COL ;cắt địa chỉ cột cập nhật
          CPI R17,$80 ;lặp vòng chờ đọc ký tự mới
          BRCC SW_DISP
          MOV COL,R17
          RJMP START1
;-----;
;Xử lý xuống dòng
;-----;
LINE_FD: LDI BUF_DAT,$20 ;nạp ký tự trống
          RCALL OUT_CHR ;xuất ra GLCD xóa con trỏ tại vị trí hiện hành
          INC PAGE_P ;tăng đến trang kế tiếp
          CBR FLG_REG,(1<<CS_FLG);CS_FLG=0 báo chọn 1/2 màn hình trái
          CPI PAGE_P,8 ;quá 8 trang?
          BRCS SKIP_P ;chưa, xuống dòng
          RJMP START ;trở về đầu màn hình
SKIP_P: LDI COL,$40 ;đặt lại địa chỉ cột đầu
          RJMP START1 ;tiếp tục đọc ký tự
;-----;
;Xử lý hiển thị hết 64 cột
;-----;
SW_DISP: SBRS FLG_REG,CS_FLG ;CS_FLG=1 bỏ qua lệnh kế tiếp
          RJMP R_DISP ;chuyển qua hiển thị 1/2 màn hình phải
          CBR FLG_REG,(1<<CS_FLG);CS_FLG=0 báo trả về hiển thị 1/2 màn hình trái
          INC PAGE_P ;tăng đến địa chỉ trang kế tiếp
          CPI PAGE_P,8 ;quá 8 trang?
          BRNE SKIP_P ;chưa, tiếp tục đọc ký tự
WAIT_CHR1: RCALL GET_CHR ;chờ đọc ký tự tiếp
          SBRS FLG_REG,R_FLG ;R_FLG=1 bỏ qua lệnh kế tiếp
          RJMP WAIT_CHR1 ;R_FLG=0 chờ đọc ký tự
          RJMP START ;trở về đầu màn hình

```

R_DISP: SBR FLG_REG,(1<<CS_FLG);CS_FLG=1 báo hiển thị 1/2 màn hình phải
RJMP SKIP_P ;tiếp tục đọc ký tự

;-----
;GET_FADDR tính địa chỉ nền truy xuất mã font ký tự trong Flash ROM
;Input R17=mã ASCII ký tự
;Z= TAB_CHR+(R17-\$20)*4
;Dịch trái Z tạo địa chỉ nền

;-----
GET_FADDR:

LDI	ZH,HIGH(TAB_CHR);Z trả địa chỉ đầu bảng tra bộ mã font ký tự
LDI	ZL,LOW(TAB_CHR)
SUBI	R17,\$20 ;lấy offset bằng cách trừ \$20
LDI	R16,4 ;và nhân cho 4
MUL	R17,R16
ADD	R30,R0 ;cộng offset vào địa chỉ nền đầu bảng tra
ADC	R31,R1
LSL	R30 ;dịch trái ZL bit7→ C
ROL	R31 ;quay trái ZH qua C tạo địa chỉ nền truy xuất bộ nhớ CT
RET	

;-----
;OUT_CHR ghi mã font ký tự ra LCD tại địa chỉ PAGE_P và COL hiện hành
;Input PAGE_P,COL đ/c trang và cột hiện hành,BUF_DAT=mã ký tự ASCII
;Bit RS=0/1 ghi mã lệnh/data ra GLCD
;Sử dụng ece GET_ADDR tính đ/c nền truy xuất data trong Flash ROM
;Sử dụng ece OUT_GLCD ghi lệnh/data ra GLCD
;Sử dụng R17 nạp data,COUNT bộ đếm

;-----
OUT_CHR: MOV R17,PAGE_P

ORI	R17,\$B8 ;R17=địa chỉ trang hiện hành
CBI	CONT,RS ;truy xuất lệnh
RCALL	OUT_GLCD ;ghi ra GLCD
MOV	R17,COL ;R17= địa chỉ cột hiện hành
CBI	CONT,RS ;truy xuất lệnh
RCALL	OUT_GLCD ;ghi ra GLCD
MOV	R17,BUF DAT ;R17=mã ký tự
RCALL	GET_FADDR ;tính địa chỉ bắt đầu mã font ký tự trong Flash ROM
LDI	COUNT,8 ;đếm xuất data tương ứng 8 cột

AGAIN: LPM R17,Z+ ;lấy mã font ký tự từ Flash ROM
SBI CONT,RS ;truy xuất data hiển thị ra GLCD
RCALL OUT_GLCD ;hiển thị ký tự ra GLCD
DEC COUNT ;đếm số lần xuất
BRNE AGAIN ;lập vòng đủ 8 lần

RET

;-----
;OUT_GLCD ghi lệnh/data ra GLCD tùy vào bit RS=0/1
;Input R17=mã lệnh/data
;CS_FLG=0/1: truy xuất 1/2 màn hình trái/phải

;-----
OUT_GLCD:

LDI	R16,20
RCALL	WAIT ;chờ 10μs
SBRS	FLG_REG,CS_FLG ;CS_FLG=1 bỏ qua lệnh kế tiếp,1/2 màn hình phải
RJMP	L_DISP ;CS_FLG=0 1/2 màn hình trái
SBI	CONT,CS1 ;cảm 1/2 màn hình trái
CBI	CONT,CS2 ;2MC,cho phép 1/2 màn hình phải

```

        RJMP    EN_GLCD      ;2MC
L_DISP: SBI     CONT,CS2   ;cầm 1/2 màn hình phải
        CBI     CONT,CS1   ;2MC, cho phép 1/2 màn hình trái
EN_GLCD: RCALL   OUT_BUS   ;ghi ra bus data GLCD
        SBI     CONT,CS1   ;cầm cả màn hình
        SBI     CONT,CS2
        RET

;-----;
;OUT_BUS ghi lệnh/data ra GLCD
;Sử dụng R17
;-----;

OUT_BUS: OUT    OUTPORT,R17  ;1MC, ghi data/lệnh ra GLCD
        SBI     CONT,E       ;2MC, E=1 cho phép GLCD
        NOP
        NOP
        NOP
        CBI     CONT,E       ;2MC, E=0 cầm GLCD
        RET

;-----;
;CLEAR khởi động xóa toàn bộ màn hình bằng cách ghi $00 ra DDRAM
;Bắt đầu từ trang 0,xóa từ cột 0 đến cột 63
;Lặp vòng như trên cho đến hết 8 trang
;-----;

CLEAR: LDI     PAGE_P,0    ;bắt đầu trang 0
CLOOP1: LDI     R17,$B8    ;lấy đ/c trang
        OR      R17,PAGE_P
        CBI     CONT,RS    ;RS=0 mã lệnh
        CBI     CONT,CS1   ;cho phép cả 2 1/2 màn hình
        CBI     CONT,CS2
        LDI     R16,20
        RCALL   WAIT      ;chờ 10µs
        RCALL   OUT_BUS   ;ghi đ/c trang ra GLCD
        LDI     R17,$40    ;bắt đầu cột 0
        LDI     R16,20
        RCALL   WAIT      ;chờ 10µs
        RCALL   OUT_BUS   ;ghi đ/c cột ra GLCD
        LDI     COUNT,64   ;đếm số cột
        CLOOP2: CLR    R17
        SBI     CONT,RS    ;R17=$00
        LDI     R16,20
        RCALL   WAIT
        RCALL   OUT_BUS   ;xóa DDRAM tại đ/c đặt
        DEC    COUNT      ;đếm số lần xóa
        BRNE   CLOOP2    ;lặp vòng cho đến hết 64 cột
        INC    PAGE_P    ;tăng đ/c trang kế tiếp
        CPI    PAGE_P,8   ;hết 8 trang?
        BRNE   CLOOP1    ;lặp vòng lại nếu chưa hết 8 trang
        CBI     CONT,CS1   ;cầm cả màn hình
        CBI     CONT,CS2
        RET

;-----;
;GET_CHR nhận dạng đọc nút nhấn có chống rung
;SW1 nhấn đọc data từ port input vào cát trong BUF_DAT
;SW2 nhấn BUF DAT=CR ký tự xuống dòng
;Sử dụng ctc GET_KEY2 đọc trạng thái nút nhấn

```

;Sử dụng R16,R17,R19=BUF_DAT

```
-----  
    GET_CHR: LDI      R16,50      ;số lần nhận dạng SW nhấn  
    BACK1:   RCALL GET_KEY2  
             BRCC  NO_SW      ;đếm số lần nhận dạng SW  
             DEC   R16        ;đếm số lần nhận dạng SW  
             BRNE  BACK1      ;lặp vòng cho đủ số lần đếm  
             PUSH  R17        ;xác nhận SW nhấn,cắt mã SW  
    WAIT_1:  LDI      R16,50      ;số lần nhận dạng SW nhả  
    BACK2:   RCALL GET_KEY2  
             BRCS  WAIT_1     ;C=1 SW chua nhả  
             DEC   R16        ;đếm số lần nhận dạng SW  
             BRNE  BACK2      ;lập vòng cho đủ số lần đếm  
             POP   R17        ;  
             CPI   R17,1      ;  
             BRNE  SW2_P      ;  
             IN    BUF_DAT,DAT_IN  
             SBR   FLG_REG,(1<<R_FLG)  
             RJMP EXIT_G  
SW2_P:   CPI   R17,0      ;  
             BRNE  NO_SW      ;  
             LDI   BUF_DAT,CR  
             SBR   FLG_REG,(1<<R_FLG)  
             RJMP EXIT_G  
NO_SW:   CBR   FLG_REG,(1<<R_FLG)  
EXIT_G:  RET  
-----
```

;GET_KEY2 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
;sử dụng R13,R17,R23

```
-----  
    GET_KEY2: LDI      R17,2      ;R17 chứa số vị trí SW  
             MOV      R13,R17    ;cắt số SW vào R1  
             IN       R23,CONT_IN ;đọc SW  
             ANDI    R23,0XC0    ;che 2 bit cao  
             CPI     R23,0XC0    ;xem có SW nhấn?  
             BRNE    CHK_KEY    ;R23 khác $C0, có SW nhấn  
NO_KEY:   CLC  
             RJMP   EXIT_KEY   ;SW chưa nhấn,C=0  
             ;thoát  
CHK_KEY:  ROL      R23        ;quay trái qua C tìm vị trí SW nhấn  
             BRCC    KEY_CODE   ;C=0 có SW nhấn  
             DEC     R13        ;SW không nhấn giảm vị trí SW  
             BRNE    CHK_KEY    ;lập vòng xét đến hết số vị trí SW  
             RJMP    NO_KEY     ;thoát khi không có SW nhấn  
KEY_CODE: SUB     R17,R13    ;R17=mã SW  
             SEC  
EXIT_KEY: RET  
-----
```

;WAIT tạo trễ µs
;Input R16=số lần lập vòng Td=R16x4MC
;Sử dụng R15

```
-----  
    WAIT:   MOV      R15,R16  
    DL1:    DEC      R15        ;1MC
```

NOP	;1MC
BRNE DL1	;2/1MC
RET	

- Trong chương trình trên sử dụng cả reset cứng qua điều khiển chân RST và reset mềm bằng chương trình con CLEAR ở đầu chương trình
- Tại địa chỉ trang và cột hiện hành, hiển thị ký hiệu con trỏ chờ nhập ký tự. Khi hiển thị ký tự đọc vào, con trỏ sẽ dịch sang vị trí kế tiếp
- Khi ký tự đọc vào là CR, sẽ xóa ký tự con trỏ tại vị trí hiện hành và xử lý xuống dòng
- Sử dụng FLG_REG=R20 làm thanh ghi chứa các cờ báo R_FLG=bit0, CS_FLG=bit1
- Chương trình con GET_CHR nhận dạng và đọc nút nhấn có chống rung SW (xem lại ví dụ 6.4)
- Chương trình con GET_FADDR từ mã ASCII ký tự đọc vào tính địa chỉ nền truy xuất bộ mã font ký tự gồm 8 byte cát trong Flash ROM như ví dụ 6.13
- Chương trình con OUT_CHR hiển thị ký tự tại vị trí hiện hành của con trỏ
- Chương trình con OUT_GLCD ghi mã lệnh/data tùy vào RS=0/1 ra 1/2 màn hình trái/phải tùy vào cờ báo CS_FLG=0/1
- Chương trình con OUT_BUS xuất data ra bus data GLCD và xuất xung E cho phép GLCD với thời gian thỏa điều kiện yêu cầu, tín hiệu RW=0 ở đầu chương trình

❖ Chương trình C VD6-18

```
#include <avr/io.h>
#include "font_chr.h"
#define import PINB //định nghĩa import=PINB
#define export PORTB //định nghĩa export=PORTB
#define ioSetb DDRB //định nghĩa ioSetb=định hướng PORTB
#define cont PORTC //định nghĩa cont=PORTC
#define cont_dr DDRC //định nghĩa cont_dr=DDRC
#define cont_in PINC //định nghĩa cont_in=PINC đọc SW1=PC6,SW2=PC7
#define dat_in PIND //định nghĩa dat_in=PIND
#define dat PORTD //định nghĩa port data
#define ioSetd DDRD //định nghĩa ioSetd=định hướng PORTD
const char RS_0=0xfe ;//RS=PC0=0 truy xuất lệnh b11111110
const char RS_1=0x01 ;//RS=PC0=1 truy xuất data b00000001
const char RW_0=0xfd ;//RW=PC1=0 truy xuất ghi b11111101
const char RW_1=0x02 ;//RW=PC1=1 truy xuất đọc b00000010
const char E_0=0xfb ;//E=PC2=0 cầm LCD b11111011
const char E_1=0x04 ;//E=PC2=1 cho phép LCD b00000100
const char CS1_0=0xf7 ;//CS1=PC3=0 cho phép 1/2 màn hình trái b11110111
const char CS1_1=0x08 ;//CS1=PC3=1 cầm 1/2 màn hình trái b00001000
const char CS2_0=0xef ;//CS2=PC4=0 cho phép 1/2 màn hình phải b11101111
const char CS2_1=0x10 ;//CS2=PC4=1 cầm 1/2 màn hình phải b00010000
const char RST_0=0xdf ;//RST=PC5=0 reset GLCD b11011111
const char RST_1=0x20 ;//RST=PC5=1 GLCD hoạt động b00100000
const char CR=0xd //ký hiệu mã xuống dòng
const char NULL=0x00 //ký hiệu mã kết thúc
const char cursor=0x80 //ký hiệu ký tự con trỏ
void CLEAR() //khai báo hàm khởi động xóa GLCD
void OUT_CHR() //khai báo hàm ghi ký tự hiển thị ra GLCD
void OUT_GLCD(unsigned char da_co) //khai báo hàm ghi mã lệnh/data ra 1/2 màn hình
void OUT_BUS(unsigned char val) //khai báo hàm ghi mã lệnh/data ra bus data GLCD
void GET_FADDR(unsigned char chr_code); //khai báo hàm tính địa chỉ nền truy xuất mã ký tự
void GET_CHR() //khai báo hàm đọc ký tự
```

```

void GET_KEY2()      ;//khai báo hàm nhận dạng SW nhấn
void delay_ms(unsigned int n) ;//khai báo hàm delay ms
unsigned char i,tam,page,page_addr,col,R_flg,CS_flg;
unsigned int offset   ;
int main()
{
    cont_dr=0x3f    ;//RS=PC0,RW=PC1,E=PC2 ,CS1=PC3,CS2=PC4,RST=PC5 output
                    //SW1=PC6,SW2=PC7 input
    cont=0xd8       ;//RS=0,RW=0 ghi,E=0 cấm,CS1=1,CS2=1 không truy xuất màn hình,
                    //RST=0 reset GLCD,R kéo lên PC6,PC7
    iosetb=0xff     ;//khai báo PORTB=output
    iosetd=0x00     ;//khai báo input port data
    dat=0xff        ;// port data R kéo lên
    delay_ms(13)   ;//delay 10us
    cont=cont | RST_1;//kết thúc reset
    while(1)
    {
        CS_flg=0      ;//chọn truy xuất 1/2 màn hình trái
        CLEAR()        ;//xóa toàn bộ màn hình(xóa DDRAM)
        page=0x00      ;//địa chỉ trang bắt đầu
        col=0x40       ;//địa chỉ cột bắt đầu
        while((page<=7) && (col<0x80))
        { //lặp vòng khi chưa hết 8 trang,64 cột
            tam=cursor      ;//mã ký tự con trỏ
            OUT_CHR()        ;//ghi ký tự hiển thị ra GLCD
            do
                GET_CHR()      ;//đọc ký tự
                while(R_flg!=1) ;//kiểm tra cờ R_flg=1?
                if(tam!=CR)    ;//kiểm tra ký tự=CR?
                {
                    while((page<=7)&&(tam>=0x20) &&(tam<=0x7F))/ký tự từ 0x20 đến 0x7f
                    { //và chưa hết 8 trang
                        OUT_CHR(tam); //ghi ký tự hiển thị ra LCD
                        col=col+8      ;//tăng đến vị trí cột hiển thị kế tiếp
                        if(col>=0x80)  //quá 64 cột?
                        {
                            //xử lý xuống dòng
                            if(CS_flg==1)//đang hiển thị 1/2 màn hình phải?
                            {
                                CS_flg=0  ;//trở về 1/2 màn hình trái
                                page++    ;//tăng đến trang kế tiếp
                                if(page>=8) //quá 8 trang?
                                {
                                    do
                                        GET_CHR()  ;
                                        while(R_flg!=1) ;//chờ nhập ký tự
                                    }
                                }
                            else //đang hiển thị 1/2 màn hình trái
                                CS_flg=1;//chuyển qua 1/2 màn hình phải
                                col=0x40;//trở về cột đầu dòng
                            } //hết phần xét quá 64 cột
                        } //hết phần while xét ký tự nhập trong tầm
                    } //hết phần if xét ký tự khác CR
                else //xử lý xuống dòng
                {

```

```

        tam=0x20    ;//mã ký tự trống
        OUT_CHR(tam); //ghi ký tự ra GLCD
        page++      ;//tăng đến trang kế tiếp
        CS_flg=0    ;//trở về 1/2 màn hình trái
        col=0x40    ;//đặt lại cột đầu khi xuống dòng
    }
}

void GET_FADDR(unsigned char chr_code)
{
    offset=(chr_code-0x20)*8    ;//tính offset để suy ra địa chỉ nền bảng tra mã font ký tự
}
void OUT_CHR()
{
    cont=cont & RS_0; //truy xuất lệnh
    page_addr=page | 0xb8 ;//đặt địa chỉ trang
    OUT_GLCD(page_addr) ;//ghi địa chỉ trang hiện hành ra GLCD
    OUT_GLCD(col)       ;//ghi địa chỉ cột hiện hành ra GLCD
    GET_FADDR(tam)      ;//tính offset địa chỉ nền tra bảng mã font ký tự
    for(i=0;i<=7;i++)
    {
        cont=cont | RS_1      ;//RS=1 truy xuất data
        tam=TAB_CHR[offset+i];//lấy mã font ký tự từ bảng tra
        OUT_GLCD(tam) ;
    }
}
void OUT_GLCD(unsigned char da_co)
{
    delay_ms(13); //chờ 10us
    if(CS_flg==0) //truy xuất 1/2 màn hình trái?
    {
        cont=cont | CS2_1 ;//khóa 1/2 màn hình phải
        cont=cont & CS1_0 ;//cho phép 1/2 màn hình trái
    }
    else
    {
        cont=cont | CS1_1 ;//khóa 1/2 màn hình trái
        cont=cont & CS2_0 ;//cho phép 1/2 màn hình phải
    }
    OUT_BUS(da_co)   ;//xuất data ra bus data GLCD
}
void OUT_BUS(unsigned char val)//xuất data ra bus data GLCD
{
    outport=val ;
    cont=cont | E_1; //E=1 cho phép xuất GLCD
    cont=cont | E_1; //tạo thời gian mức 1
    cont=cont | E_1;
    cont=cont&E_0; //E=0 cầm GLCD
}
void CLEAR() //xóa toàn bộ màn hình(xóa DDRAM)
{
    page=0x00 ;//địa chỉ trang bắt đầu
    while(page<=7)

```

```

    {
        page_addr=page |0xb8;//đặt địa chỉ trang
        col=0x40           ;//địa chỉ cột bắt đầu
        cont=cont & RS_0   ;//RS=0 truy xuất lệnh
        cont= cont & CS1_0 ;//cho phép 1/2 màn hình trái
        cont=cont &CS2_0  ;//cho phép 1/2 màn hình phải
        delay_ms(13)      ;//chờ 10us
        OUT_BUS(page_addr);//ghi địa chỉ trang ra GLCD
        delay_ms(13)      ;//chờ 10us
        OUT_BUS(col)      ;//ghi địa chỉ cột ra GLCD
        for (i=0;i<=63;i++)
        {
            tam=0 ;
            cont=cont | RS_1;//truy xuất data
            delay_ms(13) ;
            OUT_BUS(tam) ;//xóa DDRAM tại địa chỉ hiện hành
        }
        page++ ;//tăng đến trang kế tiếp
    }
    cont=cont | CS1_1 ;//cảm 1/2 màn hình trái
    cont=cont | CS2_1 ;//cảm 1/2 màn hình phải
}
void GET_CHR()//định nghĩa hàm đọc ký tự
{
    unsigned char j,key_code;
    for (j=50;j>=1;j--) // đọc trạng thái SW nhả 50 lần
        GET_KEY2() // đọc trạng thái SW
    if(R_flg!=0)
    {
        key_code=tam //gán mã SW vào tam
        for (j=50;j>=1;j--) // đọc trạng thái SW nhả 50 lần
        {
            GET_KEY2() // đọc trạng thái SW
            if (R_flg==1) // cờ báo=1 SW chưa nhả
                j=50 // đọc lại từ đầu
        }
        tam=key_code //lấy lại mã SW
        switch(tam)
        {
            case(0): //mã SW=0
            {
                tam=dat_in;// nhập ký tự từ port data input
                R_flg=1 //cờ báo có SW nhả
                break;
            }
            case(1): //mã SW=1
            {
                tam=0x0d;//ký tự CR
                R_flg=1 //cờ báo có SW nhả
                break;
            }
        }
    }
}

```

```

void GET_KEY2()//định nghĩa hàm nhận dạng SW nhấn
{
    unsigned char col_gk,buf,jgk;// khai báo các biến sử dụng trong hàm
    col_gk = 0xbff      ;// mã quét vị trí SW1 bit6
    tam=0x02          ;//số SW
    jgk=0x02          ;// đếm số vị trí SW
    buf=cont_in & 0xc0 // đọc trạng thái SW,che 2 bit cao
    if( buf == 0xc0 ) // buf=0x0c không có SW nhấn
        R_flg = 0 // xóa cờ báo
    else
    {
        R_flg=0 ;
        while ( (R_flg == 0) &&( jgk != 0 ) ) // lặp vòng khi cờ báo=0 và quét chưa hết 2SW
        {
            if(buf == (col_gk & 0xc0)) // đúng vị trí SW đang quét
            {
                tam=tam - jgk ;// tính mã SW
                R_flg = 1 // đặt cờ báo=1
            }
            else
            {
                jgk-- //vị trí SW kế tiếp
                col_gk=col_gk<<1 ;//quét vị trí SW kế tiếp
                col_gk++ //đặt LSB=1
            }
        }
    }
}

void delay_ms(unsigned int n) // ctc delay ms
{
    unsigned int k;
    for(k=0;k<=n;k++) // Td=6xn MC
}

```

- Trong chương trình C tạo các hàm có cùng tên và chức năng như chương trình hợp ngữ để người đọc dễ đối chiếu
- Hàm GET_FADDR trong chương trình C thực chất chỉ tính offset,không cần tính địa chỉ nền như chương trình hợp ngữ,vì khi khởi động bảng tra bộ mã font ký tự tự động được nạp từ Flash ROM sang SRAM(xem lại nhận xét ở ví dụ 6.13).
- Hàm GET_CHR và GET_KEY2 áp dụng từ ví dụ 6.4,chỉ khác ở điểm trong ví dụ 6.4 phải nhấn SW mới thoát khỏi hàm,trong ví dụ này hàm GET_CHR cho thoát sau khi kiểm tra 50 lần hàm GET_KEY2 nếu không có SW nhấn với R_flg=0,trả về data ký tự nhập hoặc mã CR và R_flg=1 nếu có SW nhấn.
- Cũng giống như các ví dụ trên,do chương trình C không có khai báo bit nên ta dùng 2 byte R_flg và CS_flg thay cho 2 cờ báo thu ký tự và truy xuất 1/2 màn hình trái/phải.

❖ Câu hỏi ôn tập

1. Trong ví dụ 6.18,để hiển thị ký tự ở vị trí thứ 3 dòng 5,cho biết địa chỉ trang và cột tương ứng, ký tự hiển thị phần nào của màn hình.
2. Để xuất ký tự ra GLCD như câu 1,trình tự xuất các tín hiệu điều khiển như thế nào?
3. Từ chương trình con OUT_BUS,vẽ giản đồ xung các tín hiệu RS,RW,CS1,CS2,E,data.Tính các thông số thời gian như trong giản đồ xung hình 6.39,so sánh với các thông số trong data sheet
4. Sử dụng chương trình con OUT BUS,viết một đoạn chương trình xuất 1 điểm sáng tại vị trí địa chỉ hàng 4 thuộc trang 3,cột 30 ở 1/2 màn hình phải.

5. Dựa vào cách thiết lập font ký tự như hình 6.41, thiết lập bộ mã font ký tự hình trái tim. Sử dụng chương trình con OUT_CHR trong ví dụ 6.18, hiển thị hình trái tim tại vị trí như câu 1.

6.6 Mở rộng truy xuất EEPROM

Cấu hình MCU324P có bộ nhớ EEPROM trên chip dung lượng 1KB, địa chỉ từ 0000 đến 03FFH.

Trong một số chương trình thường có cài đặt các thông số ban đầu vận hành chương trình cần lưu lại, đôi khi có thể thay đổi các thông số này. Cách tốt nhất là lưu các thông số này vào EEPROM, để đảm bảo vẫn lưu lại khi cắt điện, hoặc có thể lập trình thay đổi bất cứ khi nào cần thiết.

6.6.1 Các thanh ghi truy xuất EEPROM

Để quản lý hoạt động truy xuất EEPROM, MCU324P có 3 thanh ghi:

- EECR (EEPROM Control Register) chứa các bit điều khiển truy xuất EEPROM
- EEDR (EEPROM Data Register) chứa data truy xuất EEPROM
- EEARH-EEARL (EEPROM Address Register High-Low) chứa địa chỉ truy xuất EEPROM
- Các thanh ghi trên có địa chỉ IO nhỏ hơn 0x60 nên truy xuất bằng các lệnh IN/OUT

1. EEDR

- Muốn ghi data vào EEPROM, ta ghi data vào EEDR. Ví dụ: OUT EEDR, R17
- Muốn đọc data từ EEPROM, ta đọc data từ EEDR. Ví dụ: IN R20, EEDR

2. EEARH và EEARL

- Trước khi truy xuất data qua EEDR, ta phải đặt địa chỉ EEPROM 16 bit qua 2 thanh ghi EEARH và EEARL.
- Do dung lượng EEPROM chỉ 1KB, nên chỉ biểu diễn 10 bit địa chỉ thấp

BIT	15	14	13	12	11	10	9	8
EEARH	-	-	-	-	-	-	EEAR9	EEAR8
EEARL	EEAR7	EEAR6	EEAR5	EEAR4	EEAR3	EEAR2	EEAR1	EEAR0
BIT	7	6	5	4	3	2	1	0

BIT	7	6	5	4	3	2	1	0
EECR	-	-	EEPM1	EEPM0	EERIE	EEMPE	EEPE	EERE
BIT	7	6	5	4	3	2	1	0

Hình 6.44: Mô tả các thanh ghi EEARH-EEARL và EECR

3. EECR

- Truy xuất bit được
- EEPM1:EEPM0 chọn mô thức ghi vào EEPROM như mô tả trong Bảng 6.13.

Bảng 6.13: Chọn các mô thức ghi EEPROM

EEPM1	EEPM0	Thời gian lập trình(Fosc=8Mhz)	Hoạt động
0	0	3.4ms	Xóa và ghi
0	1	1.8ms	Chỉ xóa
1	0	1.8ms	Chỉ ghi
1	1	-	Dự trữ

Khi mới reset hệ thống, EEPM1=EEPM0=0. Khi EEPE=1, không ghi các bit này được

- EEMPE (EEPROM Master Programming Enable): EEMPE=1 trong vòng 4 chu kỳ xung CK cho phép ghi data vào EEPROM.

- Sau 4 chu kỳ xung CK, phần cứng xóa EEMPE=0 nên không thể đáp ứng ghi được cho dù đặt EEPE=1!
- - EEPE (EEPROM Programming Enable): EEPE=1 cùng với EEMPE=1 trong vòng 4 chu kỳ xung CK, data ghi vào EEPROM (qua thanh ghi EEDR) được chấp nhận.
- Khi thực thi lệnh đặt EEPE=1 xong, MCU sẽ treo 2 chu kỳ CK mới thực thi lệnh kế tiếp
- Khi ghi xong data phần cứng sẽ xóa EEPE=0
- Thời gian ghi khoảng 26368MC # 3.3ms với Fosc=8Mhz

- **EERE(EEPROM Read Enable):** với điều kiện EEPE=1,EERE=1 cho phép đọc data từ EEPROM(qua thanh ghi EEDR)
 - Khi thực thi lệnh đọc EEPROM, MCU sẽ treo 4 chu kỳ CK mới thực thi lệnh kế tiếp
 - Sau khi đọc xong data, phần cứng sẽ xóa EERE=0
- **ERIE(EEPROM Ready Interrupt Enable):** cho phép ngắt khi truy xuất EEPROM,sẽ trình bày trong chương 10.

6.6.2 Cách truy xuất EEPROM

1. Truy xuất ghi

Các bước thực hiện lệnh như sau:

1. Chờ bit EEPE=0 chỉ báo EEPROM ghi xong
2. Đặt địa chỉ truy xuất EEPROM vào EEARH:RRARL
3. Ghi data vào EEDR
4. Đặt bit EEMPE=1
5. Đặt bit EEPE=1 data được ghi vào EEPROM với địa chỉ tương ứng

Ngay sau khi thực hiện lệnh bước 4 phải thực hiện ngay lệnh bước 5 để đảm bảo việc ghi data trong vòng 4 chu kỳ CK ngay sau khi bit EEMPE=1.

Ví dụ 6.19: Viết một đoạn chương trình nhập data từ PORTD lưu vào EEPROM tại địa chỉ 0x100.

Giải:

```
.EQU EE_ADDR=0X100 ;gán ký hiệu địa chỉ EEPROM
...
LDI R16,0X00 ;PORTD input
OUT DDRD,R16
IN R17,PIND ;đọc data từ PORTD
LDI R16,HIGH(EE_ADDR) ;nạp địa chỉ EEPROM
OUT EEARH,R16
LDI R16,LOW(EE_ADDR)
OUT EEARL,R16
WAIT: SBIC EECR,EEPE ;EEPE=0 có thể ghi tiếp
      RJMP WAIT ;chờ EEPROM ghi xong
      OUT EEDR,R17 ;ghi data vào EEDR
      SBI EECR,EEMPE ;EEMPE=1 cho phép ghi
      SBI EECR,EEPE ;ghi data vào EEPROM
      ...

```

2. Truy xuất đọc

Các bước thực hiện lệnh như sau:

1. Chờ bit EEPE=0 chỉ báo EEPROM ghi xong
2. Đặt địa chỉ truy xuất EEPROM vào EEARH:RRARL
3. Đặt bit EERE=1
4. Đọc data từ EEPROM qua EEDR với địa chỉ tương ứng

Ví dụ 6.20: Tiếp theo ví dụ 6.19, đọc data từ EEPROM tại địa chỉ EE_ADDR xuất ra PORTB.

Giải:

```
...
LDI R16,0XFF
OUT DDRB,R16
WAIT1:SBIC EECR,EEPE
      RJMP WAIT1
      LDI R16,HIGH(EE_ADDR) ;nạp địa chỉ EEPROM
      OUT EEARH,R16
      LDI R16,LOW(EE_ADDR)
      OUT EEARL,R16
      SBI EECR,EERE
      IN R17,EEDR
```

6.6.3 Khởi động vùng nhớ EEPROM

Trong trường hợp cần lưu giữ data trong EEPROM, ta khởi động vùng nhớ lưu data cho EEPROM dùng chỉ dẫn .ESEG với hợp ngữ và định nghĩa thuộc tính `_attribute_` (dấu nối dưới kép) hoặc từ khóa EEME với C, thực hiện tương tự như ví dụ sau (xem lại chương 5, mục 5.2.4)

Ví dụ 6.21: Biên soạn lại chương trình ví dụ 6.11, với bảng mã 7 đoạn AC cất trong EEPROM.

Giải:

Trong ví dụ 6.11, bảng mã 7 đoạn AC được cất trong Flash ROM nên chương trình con/hàm GET_7SEG sẽ truy xuất data từ bộ nhớ chương trình.

Bây giờ ta khai báo lại bảng mã 7 đoạn AC nằm trong EEPROM và biên soạn lại chương trình con/hàm GET_7SEG truy xuất data từ EEPROM tương ứng. Phần còn lại của chương trình giống như ví dụ 6.11. Phần khai báo cất data trong EEPROM và chương trình con/hàm GET_7SEG như sau.

❖ Chương trình hợp ngữ VD6-21

```
...
.ESEG ;chỉ dẫn khởi động vùng nhớ EEPROM(tạo segment EEPROM)
.ORG 0X100 ;chỉ dẫn đặt địa chỉ bắt đầu
TAB_7SA: .DB 0XC0,0XF9,0XA4,0XB0,0X99,0X92,0X82,0XF8,0X80,0X90,0X88,0X83
        .DB 0XC6,0XA1,0X86,0X8E
        ...
;-----
GET_7SEG:
    LDI ZH,HIGH(TAB_7SA) ;Z trả địa chỉ đầu bảng tra
    LDI ZL,LOW(TAB_7SA) ;trong EEPROM
    ADD ZL,R17 ;cộng offset vào ZL
    LDI R17,0
    ADC ZH,R17
WAIT_0: SBIC EECR,EEPE ;kiểm tra EEPROM
        RJMP WAIT_0 ;EEPROM đang bận
        OUT EEARH,ZH ;ghi địa chỉ EEPROM
        OUT EEARL,ZL
        SBI EECR,EERE ;cho phép đọc
        IN R17,EEDR ;lấy mã 7 đoạn
        RET
;-----
```

- Chỉ dẫn .ESEG khởi động vùng nhớ lưu data trong segment EEPROM (mặc định đầu tiên là 0x0000)
- Sử dụng chỉ dẫn .ORG để đặt địa chỉ bắt đầu vùng nhớ tại 0x100
- Chương trình con GET_7SEG thực hiện đọc mã 7 đoạn từ bảng tra tương tự như đọc data từ EEPROM

❖ Chương trình C VD6.21

```
...
const char tab_7sa[] __attribute__ ((section(".eprom")))= {0xc0,0xf9,0xa4,0xb0,0x99,0x92,0x82,
0xf8,0x80,0x90,0x88,0x83,0xc6,0xa1,0x86,0x8e}; //khai báo bảng tra mã 7 đoạn AC trong EEPROM
...
unsigned char get_7seg(unsigned char x) //hàm chuyển mã 7 đoạn
{
    while(EECR&(1<<EEPE)) ;//chè EEPE=0
    EEAR=&tab_7sa[x] ;//nạp địa chỉ eeprom
    EECR|=(1<<EERE) ;//cho phép đọc
    x=EEDR ;//lấy mã 7 đoạn
    return(x)
}
...
```

- Trong phần khai báo kiểu data cất trong EEPROM của chuỗi tab_7sa[],sử dụng từ khóa định nghĩa thuộc tính **_attribute_ ((section(".eprom")))**

❖ **Câu hỏi ôn tập**

1. Ghi cụ thể vùng địa chỉ truy xuất EEPROM của MCU324P
2. Với Fosc=8Mhz, thời gian ghi data vào EEPROM của MCU324P mất bao lâu?
3. Thay vì chờ EEPE=0, có thể thực hiện lệnh nào khác?
4. Viết một chương trình con/hàm ghi data vào ô nhớ EEPROM, chạy thử với data= 0xAA, địa chỉ =0x0010
5. Viết một chương trình con/hàm đọc data từ ô nhớ EEPROM, chạy thử với địa chỉ 0x0010

Bài tập chương 6

❖ **Các bài tập sau thiết kế với Fosc=8Mhz, viết chương trình bằng hợp ngữ và C, chạy mô phỏng và minh họa kết quả.**

Phần 1 Các bài tập cơ bản

- 6.1 Viết đoạn chương trình khởi động PORTA nhập data có điện trở kéo lên, PORTB xuất data
- 6.2 Viết đoạn chương trình khởi động PORTB là port xuất/nhập, PORTC có 3 bit thấp xuất, 2 bit cao kế tiếp nhập. Khởi đầu PORTB xuất, các tín hiệu xuất được xóa bằng 0. Các bit port C có điện trở kéo lên
- 6.3 Từ bài tập 6.2, viết một đoạn chương trình khởi động các port, khi có tín hiệu PC3=0, PORTB xuất data cất trong R17 và các tín hiệu PC2PC1PC0=100. Khi có tín hiệu PC4=0, PC2PC1PC0=011 và data được đọc từ PORTB về cất trong R17.
- 6.4 Viết một đoạn chương trình nhận dạng khi PC0=0, xuất chuỗi ký tự kết thúc bằng mã NULL=\$00 từ bảng tra trong bộ nhớ chương trình có địa chỉ đầu ký hiệu TABLE, ra PORTB. Lặp vòng lại liên tục.
- 6.5 Lặp lại bài tập 6.4 thêm vào yêu cầu mỗi lần xuất data ra PORTB, tạo 1 xung trên PC2 thời gian mức 1 không nhỏ hơn 500ns.
- 6.6 Viết một đoạn chương trình nhận dạng khi PC0=0, đọc byte số HEX từ PORTD, tra bảng TAB_7SEG chứa mã 7 đoạn LED AC, xuất ra PORTB và PORTA mã 7 đoạn ứng với 2 ký tự HEX đọc được, số HEX cao ở PORTB và thấp ở PORTA.
- 6.7 Viết một chương trình con tên ROL_16 quay trái qua cờ C 1 từ nhị phân 16 bit cất trong 2 thanh ghi/ô nhớ ký hiệu H_BYTEx và L_BYTEx với H_BYTEx chứa byte cao.
- 6.8 Viết một chương trình con tên OFFSET_CAL tính giá trị biểu thức sau:

$$\text{OFFSET} = (\text{DAT} - \text{BASE}) * 8$$

OFFSET = thanh ghi/ô nhớ 16 bit
 DAT, BASE = thanh ghi/ô nhớ 8 bit
- 6.9 Viết một chương trình con tên DIV16_8 chia số nhị phân 16 bit cho 8 bit, sử dụng phép dịch bit và trừ (xem giải thuật phép chia nhị phân ở chương 0). Kết quả trả về thương số 16 bit và dư số 8 bit
- 6.10 Viết một chương trình con tên DIV_16 chia 2 số nhị phân 16 bit, kết quả trả về thương số và dư số phép chia dài 16 bit.
- 6.11 Viết một chương trình con tên BIN16_BCD chuyển đổi số nhị phân 16 bit sang số BCD nén 5 digit chứa trong 5 ô nhớ, vận dụng chương trình con DIV16_8 trong bài tập 6.9.
- 6.12 Vẽ sơ đồ MCU324P giao tiếp với nút nhấn SW0, SW1 và 2 LED đơn L0, L1. Viết chương trình thực hiện khi nhấn SW0, L0 sáng, L1 tối; khi nhấn SW1, L1 sáng, L0 tối. Chống rung SW bằng phần mềm.

6.13* Trong ví dụ 6.4,biên soạn lại chương trình thực hiện như sau:

- SW0 nhấn: tối dãy LED
- SW1 nhấn: sáng/tối dãy LED với thời gian sáng/tối 0.5s
- SW2 nhấn: sáng 1 LED dịch từ bit thấp đến cao và quay vòng lại,thời gian chờ dịch 0.5s
- SW3 nhấn: sáng 2 LED ở giữa dãy,sau đó sáng dần lan ra 2 biên và quay vòng lại,thời gian chờ sáng lan dần ra biên 0.5s

Thực hiện các mode sáng trên bằng chương trình con và trong thời gian sáng ở 1 mode bất kỳ,khi nhấn SW chuyển mode sáng khác phải đáp ứng ngay!

6.14 Trong ví dụ 6.5,thay dãy LED bằng LED 7 đoạn AC.Biên soạn lại chương trình thực hiện khi nhấn phím từ bàn phím sẽ hiển thị mã phím ra LED 7 đoạn AC.

6.15 Từ ví dụ 6.7 và hình 6.15,vẽ giản đồ xung định thì ghi bộ nhớ và ghi chú các thông số thời gian.

6.16 Sử dụng macro thay cho chương trình con đọc và ghi bộ nhớ,viết lại chương trình hợp ngữ ví dụ 6.7.

6.17* Trong hình 6.35 ví dụ 6.16,giả sử data đọc từ PORTD là số BCD nén cát trong thanh ghi/ô nhớ ký hiệu DAT_IN,viết chương trình đọc data từ PORTD,hiển thị giá trị đọc được ra LCD ký tự như sau (giả sử DAT_IN=26):

Nhiệt do :
T=26°C

6.18 Viết một macro/hàm ghi và macro/hàm đọc truy xuất EEPROM,với các biến là địa chỉ và data

Phần 2 Các bài tập nâng cao

6.19 Thiết kế mạch giải mã địa chỉ giao tiếp bus địa chỉ/data giữa MCU32 với các thiết bị ngoại vi có bảng phân vùng địa chỉ như sau:

I/O	Tín hiệu chọn chip	Vùng địa chỉ(H)	Truy xuất
U2= HM6264	/CS0	0000 - 1FFF	Đọc/Ghi
U3=HM6264	/CS1	2000 - 3FFF	Đọc/Ghi
U4=74HC573	CS2	4000 - 43FF	Ghi
U5=74HC573	CS3	5000 - 53FF	Ghi
U6=74HC245	/CS4	5400 -57FF	Đọc/Ghi

Chọn phương án thiết kế sao cho sơ đồ mạch giải mã địa chỉ gọn và đáp ứng nhanh nhất!

6.20 Vẽ và giải thích thiết kế sơ đồ MCU324P giao tiếp trực tiếp với bàn phím 16 phím như ví dụ 6.5, giao tiếp với bộ hiển thị 4 LED 7 đoạn AC thông qua 4 mạch chốt IC74HC573 ghép với 1 port và các tín hiệu điều khiển.

6.21 Viết một chương trình con/hàm tên DISP_LATCH4 hiển thị 4 ký tự mã HEX từ các thanh ghi/ô nhớ ký hiệu NUM_32,NUM_10 ra bộ hiển thị 4 chữ số như bài tập 6.20

6.22 Viết một chương trình con/hàm tên CHR_SHIFT lặp lượt dịch số Hex(từ 0-F) cát trong thanh ghi/ô nhớ ký hiệu CHR_IN vào 2 thanh ghi/ô nhớ ký hiệu NUM_32,NUM_10 theo như ví dụ sau:

- Khởi động NUM_32=\$00,NUM_10=\$00
- CHR_IN=1→gọi CHR_SHIFT→NUM_32=\$00,NUM_10=\$01
- CHR_IN=2→gọi CHR_SHIFT→NUM_32=\$00,NUM_10=\$12
- CHR_IN=3→gọi CHR_SHIFT→NUM_32=\$01,NUM_10=\$23

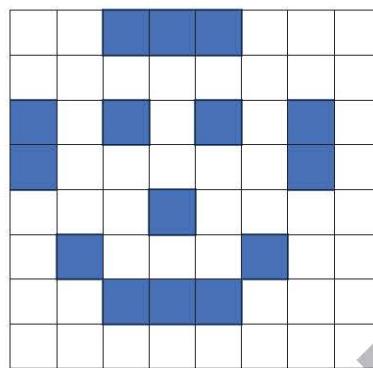
- CHR_IN=4→gọi CHR_SHIFT→NUM_32=\$12,NUM_10=\$34

-

6.23* Kết hợp bài tập 6.20 đến 6.22 viết một chương trình nhập ký tự từ bàn phím 16 phím,lần lượt hiển thị và dịch trái ký tự đã hiển thị ra bộ hiển thị 4 LED 7 đoạn AC.

Thứ viết lại chương trình con/hàm hiển thị có xóa số 0 không có nghĩa.

6.24* Thiết lập bộ mã hiển thị biểu tượng như hình sau:



Áp dụng chương trình ví dụ 6.13 và sơ đồ hình 6.25, biên soạn lại chương trình thực hiện như sau:

1. Hiển thị biểu tượng đứng yên trong 2s
2. Hiển thị dịch chuyển biểu tượng sang phải(tốc độ điều chỉnh được) cho đến hết màn hình
3. Hiển thị biểu tượng từ biên phải màn hình dịch chuyển sang trái cho đến hết biên trái màn hình
4. Hiển thị biểu tượng từ biên trên dịch chuyển xuống dưới cho đến hết biên dưới màn hình
5. Lặp vòng về bước 1

Các mode hiển thị trên viết bằng chương trình con/hàm để dễ quản lý.

6.25 Từ hình 6.27 ví dụ 6.14, vẽ sơ đồ mở rộng giao tiếp bộ hiển thị có 4 LED ma trận 8x8 vẫn dùng IC ghi dịch.Kiểm tra các điều kiện lái dòng cho bộ hiển thị đạt yêu cầu không,và đưa phương án điều chỉnh thiết kế nếu không đạt yêu cầu về lái dòng.

6.26* Từ sơ đồ thiết kế bài tập 6.25, viết chương trình hiển thị chuỗi ký tự HELLO và chuyển mode hiển thị tạo hiệu ứng màn hình tương tự như bài tập 6.23(ít nhất 2 mode)

6.27 * Chuyển ví dụ 6.16 về sơ đồ MCU giao tiếp thanh ghi dịch 74HC595, ngõ ra thanh ghi dịch lái LCD ký tự.Viết lại chương trình như ví dụ 6.16.

6.28* Từ ví dụ 6.17 và sơ đồ hình 6.36,sử dụng 1 port MCU324P giao tiếp với bàn phím 16 phím như ví dụ 6.5.Viết chương trình nhập ký tự từ bàn phím và lần lượt hiển thị ký tự ra LCD ký tự.

- Các phím số từ 0 - 9 hiển thị số tương ứng
- Phím mã \$0A hiển thị dấu chấm thập phân(chỉ hiển thị sau khi nhập 1 chữ số)
- Phím mã \$0B xóa ký tự vừa mới nhập
- Phím mã \$0C xóa toàn bộ màn hình,đưa con trỏ về đầu dòng 1
- Phím mã \$0D dịch con trỏ tới 1 ký tự
- Phím \$0E dịch con trỏ lui về 1 ký tự
- Phím \$0F xuống dòng/trở về dòng 1(nếu đang ở dòng 2)

6.29* Từ ví dụ 6.17 và sơ đồ hình 6.36, áp dụng vào mạch đếm sản phẩm hoạt động như sau:

- Cảm biến sản phẩm cho 1 xung độ rộng 10 μ s ở ngõ ra mỗi lần có 1 sản phẩm đi ngang qua
- Ngõ ra cảm biến sản phẩm kết nối với PD2 của MCU324P
- Một nút nhấn SW tích cực mức 0 reset bộ đếm về 0

Viết chương trình đếm số sản phẩm làm việc như sau:

- Sử dụng bộ đếm 16 bit dùng 2 thanh ghi/ô nhớ ký hiệu H_BYTEx(byte cao) và L_BYTEx
- Mỗi lần có cạnh xuống trên PD2,bộ đếm tăng 1 và hiển thị trên LCD ký tự trình bày như sau:

So san pham:
12345

- Trường hợp bộ đếm bị tràn mạch vẫn tiếp tục đếm và hiển thị cảnh báo bị tràn:

So san pham:
00001 tran!

- Nhấn SW sẽ reset bộ đếm về 0 cho dù bộ đếm đang hoạt động
Thử tính tốc độ đáp ứng tối đa của bộ đếm với chương trình đã thiết kế

6.30 Thiết lập bảng tra mã hiển thị chuỗi ký tự Hello! ra GLCD, phân bố điểm ảnh sao cho hiển thị được 80 - 90% toàn màn hình. Lưu ý địa chỉ trang/cột và phần màn hình trái/phải.

6.31* Từ hình 6.42 và vận dụng ví dụ 6.18 và bài tập 6.30, viết chương trình hiển thị chuỗi ký tự Hello! ra GLCD. Thử tạo hiệu ứng dịch chuyển chuỗi ký tự sang phải/trái.