

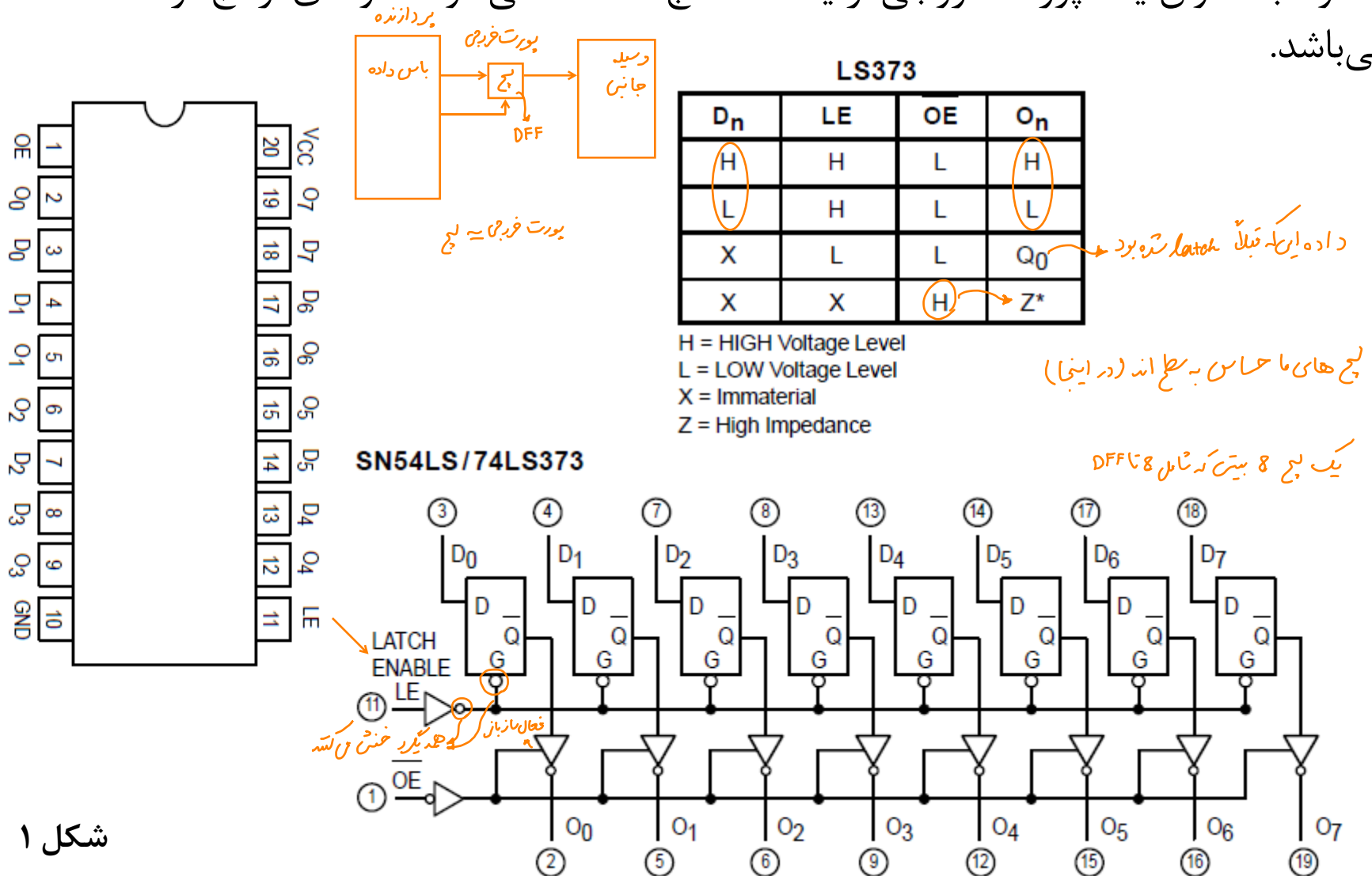
روش‌های پایه‌ی ورودی/خروجی

1

ریزپردازنده ۱
محمد مهدی همایون پور
دانشکده مهندسی کامپیوتر، دانشگاه صنعتی امیرکبیر
بهار ۱۳۹۵

پورت خروجی

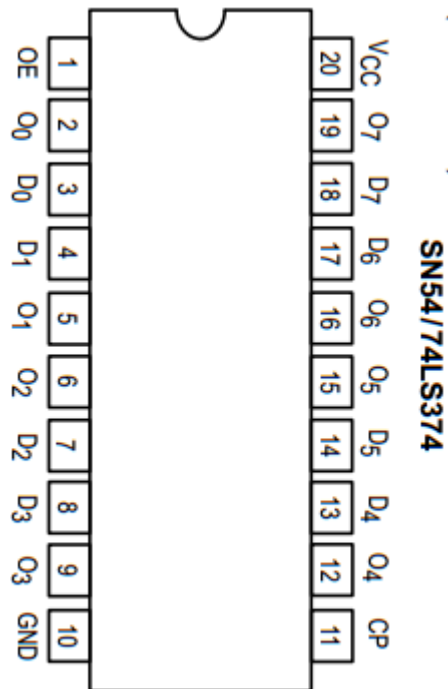
معمولا به عنوان یک پورت خروجی از یک عدد لچ استفاده می‌شود. نمونه‌ای از لچ تراشه 74LS373 می‌باشد.



شکل ۱

پورت خروجی

معمولا به عنوان یک پورت خروجی از یک عدد لچ استفاده می شود.
نمونه ای از لچ تراشه 74LS374 می باشد.

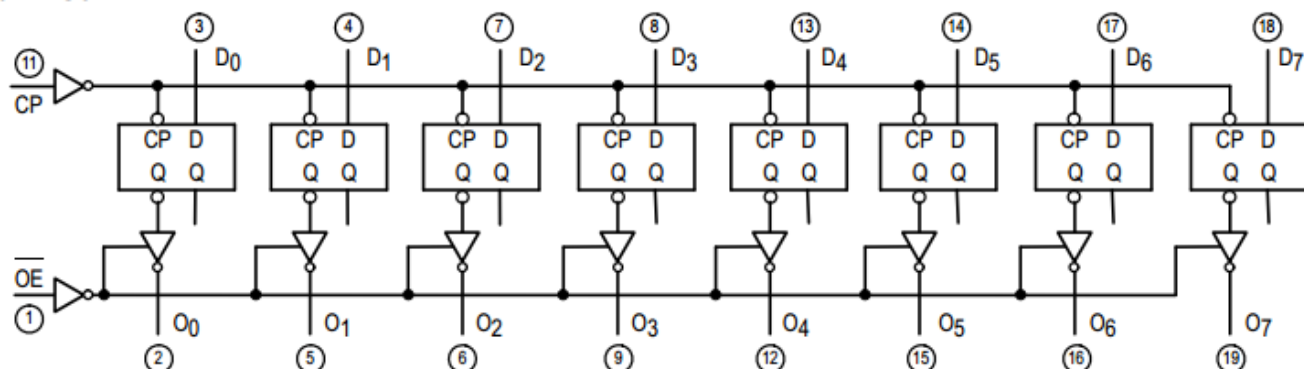


حالت به لچ

LS374

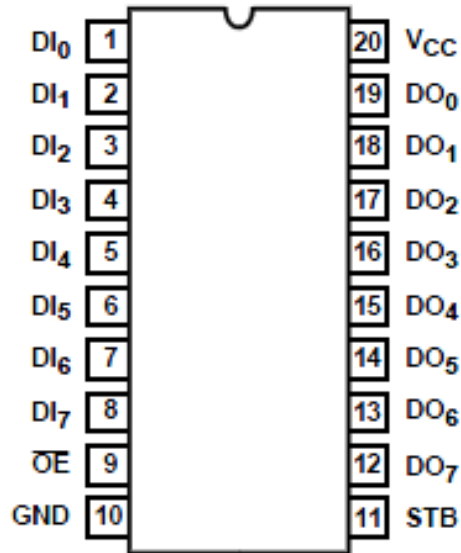
D_n	LE	OE	O_n
H		L	H
L		L	L
X	X	H	Z*

SN54LS/74LS374



شکل ۲

پورت خروجی

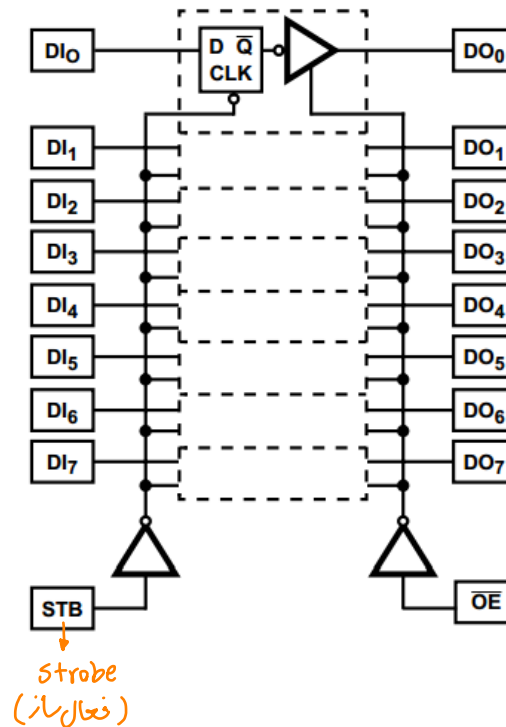


نمونه دیگری از لچ تراشه 82C82 است که یک تراشه CMOS Octal Latching Bus Driver به شکل زیر می باشد:

TRUTH TABLE

STB	\overline{OE}	DI	DO
X	H	X	Hi-Z
H	L	L	L
H	L	H	H
↓	L	X	↑

H = Logic One
 L = Logic Zero
 X = Don't Care
 ↑ = Latched to Value of Last Data
 Hi-Z = High Impedance
 ↓ = Neg. Transition



Strobe
 (فعال ساز)

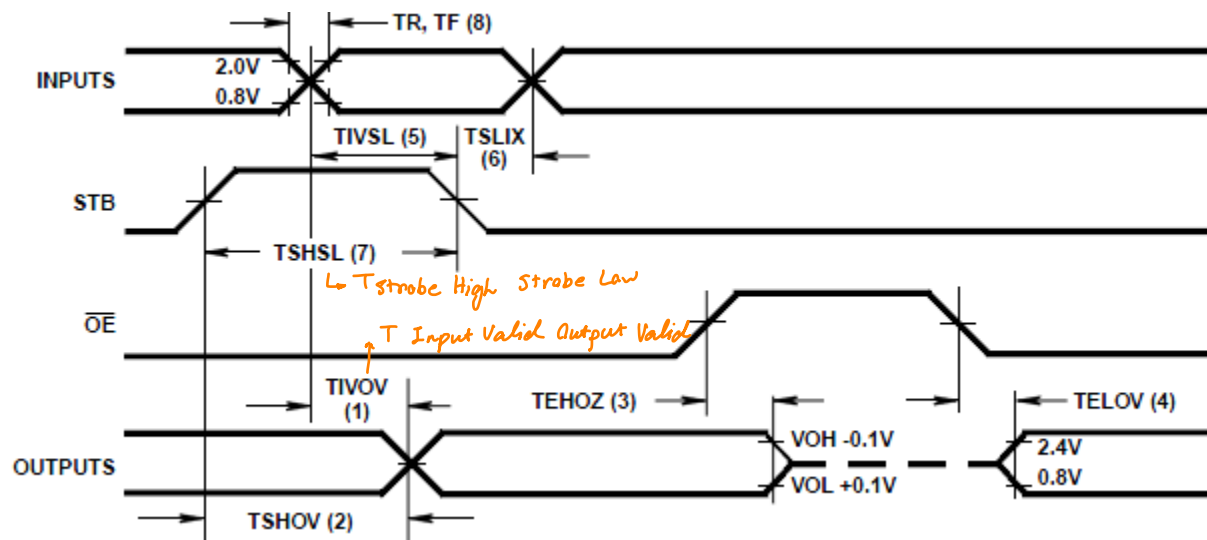
PIN	DESCRIPTION
DI ₀ -DI ₇	Data Input Pins
DO ₀ -DO ₇	Data Output Pins
STB	Active High Strobe
\overline{OE}	Active Low Output Enable

شکل ۳

مشخصات زمانی تراشه 82C82

SYMBOL	PARAMETER	MIN	MAX	UNITS	TEST CONDITIONS
(1) TIOV	Propagation Delay Input to Output	-	35	ns	Notes 2, 3
(2) TSHOV	Propagation Delay STB to Output	-	55	ns	Notes 2, 3
(3) TEHOZ	Output Disable Time	-	35	ns	Notes 2, 3
(4) TELOV	Output Enable Time	-	50	ns	Notes 2, 3
(5) TIVSL	Input to STB Setup Time	0	-	ns	Notes 2, 3
(6) TSLIX	Input to STB Hold Time	25	-	ns	Notes 2, 3
(7) TSHSL	STB High Time	25	-	ns	Notes 2, 3
(8) TR, TF	Input Rise/Fall Times	-	20	ns	Notes 2, 3

* { (5) TIVSL Input valid to Strobe Low
 (6) TSLIX چه مدت داده در ورودی بماند
 (7) TSHSL چه مدت داده در خروجی بماند



شکل ۴

پارامترهای زمانی مهم مربوط به لچ 82C82

پارامترهای زمانی مهم مربوط به لچ 82C82 عبارتند از:

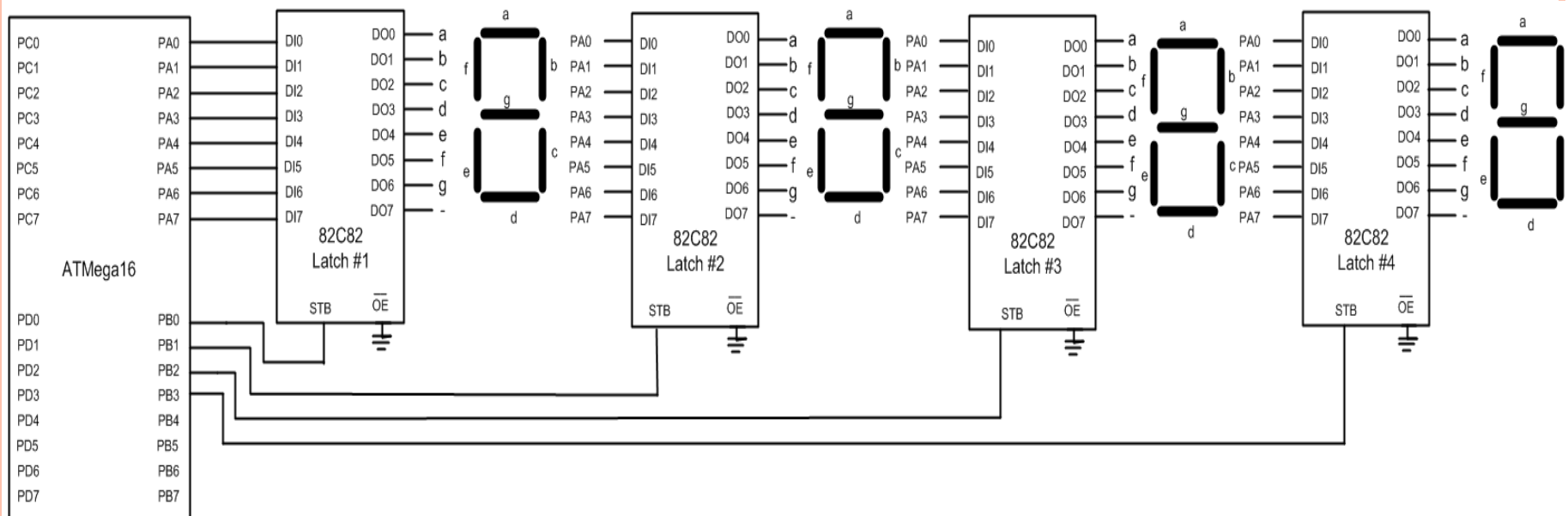
- tSHSL : زمان High بودن STB
- tIVSL : فاصله بین زمان گذاشتن داده تا لبه پایین رونده سیگنال STB
- tSLIX : فاصله بین زمان پایین رفتن پالس STB تا انتهای زمان معتبر بودن داده

SYMBOL	PARAMETER	MIN	MAX	UNITS	TEST CONDITIONS
(1) TIVOV	Propagation Delay Input to Output	-	35	ns	Notes 2, 3
(2) TSHOV	Propagation Delay STB to Output	-	55	ns	Notes 2, 3
(3) TEHOZ	Output Disable Time	-	35	ns	Notes 2, 3
(4) TELOV	Output Enable Time	-	50	ns	Notes 2, 3
(5) TIVSL	Input to STB Setup Time	0	-	ns	Notes 2, 3
(6) TSLIX	Input to STB Hold Time	25	-	ns	Notes 2, 3
(7) TSHSL	STB High Time	25	-	ns	Notes 2, 3
(8) TR, TF	Input Rise/Fall Times	-	20	ns	Notes 2, 3

اتصال ۴ لچ به میکروکنترلر و نمایش دهنده‌های ۷ قطعه‌ای

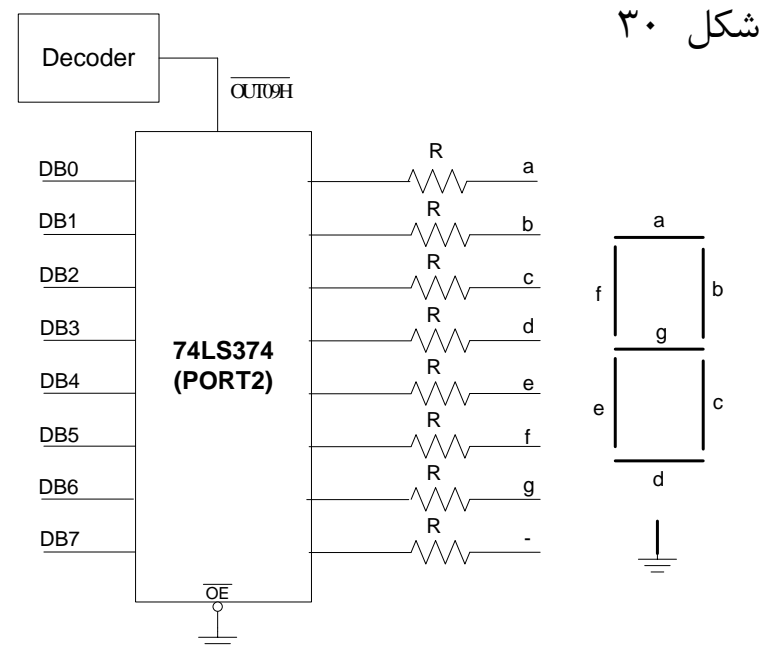
شکل ۵

کینبار داده‌ها را در حافظه می‌کنیم بعد با فعال کردن STB به 7-seg‌ها را در نمایش می‌دهیم
STB‌ها با هم فعال نمی‌کنیم چون نمی‌خواهیم به تاندم یک عدد در نمایش بدهند



اتصال نمایش دهنده ۷ قطعه‌ای به ریزپردازنده ۸۰۸۶

نام قطعه									معادل شانزدهی
رقم	-	a	b	c	d	e	f	g	
0	0	1	1	1	1	1	1	0	7EH
1	0	0	1	1	0	0	0	0	30H
2	0	1	1	0	1	1	0	1	6DH
3	0	1	1	1	1	0	0	1	79H
4	0	0	1	1	0	0	1	1	33H
5	0	1	0	1	1	0	1	1	5BH
6	0	1	0	1	1	1	1	1	5FH
7	0	1	1	1	0	0	0	0	70H
8	0	1	1	1	1	1	1	1	7FH
9	0	1	1	1	1	0	1	1	7BH
A	0	1	1	1	0	1	1	1	77H
b	0	0	0	1	1	1	1	1	1FH
c	0	0	0	0	1	1	1	0	0EH
d	0	0	1	1	1	1	0	1	3DH
E	0	1	0	0	1	1	1	1	4FH
F	0	1	0	0	0	1	1	1	47H
خاموش	0	0	0	0	0	0	0	0	00H



نمایش رقم ۸ بر روی نمایش دهنده

Table7Seg: db 7EH, 30H, 6DH, 79H, 33H, 5BH, 5FH, 70H, 7FH, 7BH, 77H, 1FH, 0EH, 3DH, 4FH, 47H, 00H



برنامه ارتباط با ۴ عدد پورت خروجی (لچ) متصل به نمایش دهنده های ۷ قطعه ای

; Show Number 1000 on four 7-Segments, 7 Segments are Common Cathode

Data in R16, R17, R18, R19

```
LDI      R16, 0x06; Value:1
LDI      R17, 0x3F, Value:0
LDI      R18, 0x3F, Value:0
LDI      R19, 0x3F, Value:0
CALL     OutputWrite
```

OutputWrite:

```
LDI      R20, 0xFF
OUT      DDRA, R20      ; PORTA is Output
OUT      DDRB, R20      ; PORTB is Output
```

```
OUT      PORTA, R16      ; Value on Port A
LDI      R20, 0x01
OUT      PORTB, R20      ; Latch1 Strobe High
LDI      R20, 0x00
OUT      PORTB, R20      ; Latch1 Strobe Low 2*62.5 ns > TSHSL=35ns    TSHSL=STB High Time
; 4*62.5 ns > TIVSL=0 ns    TIVSL=Input To STB Set Time
; (NOP and next instruction time)=2*62.5ns > TSLIX=25ns    TSLIX=Input To STB Hold Time
```

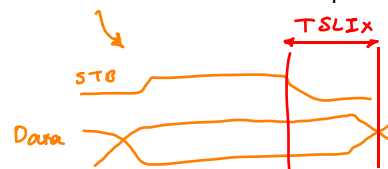
لازم نبود، حرفاً
برای خیال راحتی:

```
NOP
OUT      PORTA, R17      ; Value on Port A
LDI      R20, 0x02
OUT      PORTB, R20      ; Latch2 Strobe High
LDI      R20, 0x00
OUT      PORTB, R20      ; Latch2 Strobe Low
NOP
OUT      PORTA, R18      ; Value on Port A
LDI      R20, 0x04
OUT      PORTB, R20      ; Latch3 Strobe High
LDI      R20, 0x00
OUT      PORTB, R20      ; Latch3 Strobe Low
NOP
OUT      PORTA, R19      ; Value on Port A
LDI      R20, 0x08
OUT      PORTB, R20      ; Latch4 Strobe High
LDI      R20, 0x00
OUT      PORTB, R20      ; Latch4 Strobe Low
RET
```

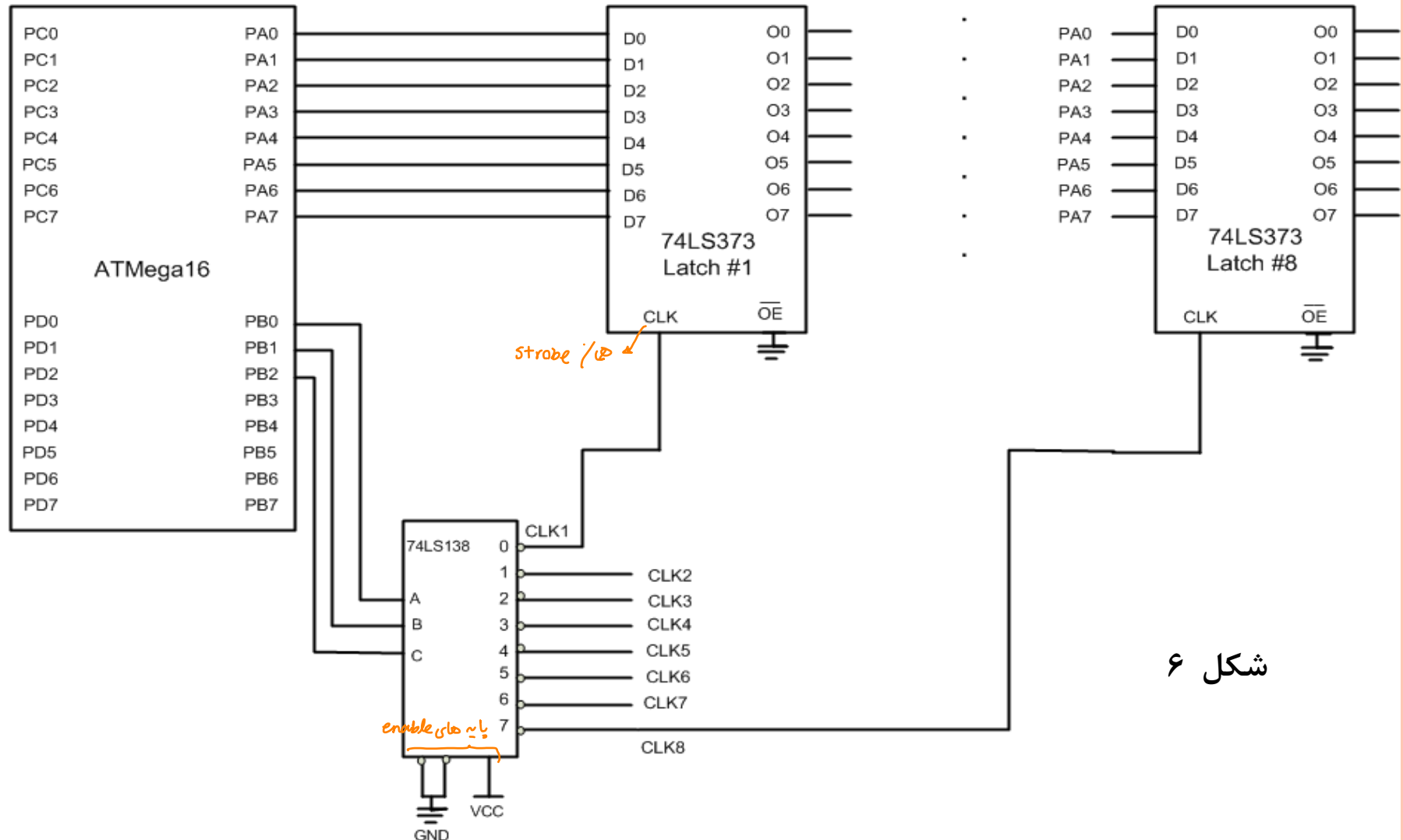
در لایه پایین ردم داده منتقل می شود

در دستور ۲

۲۵

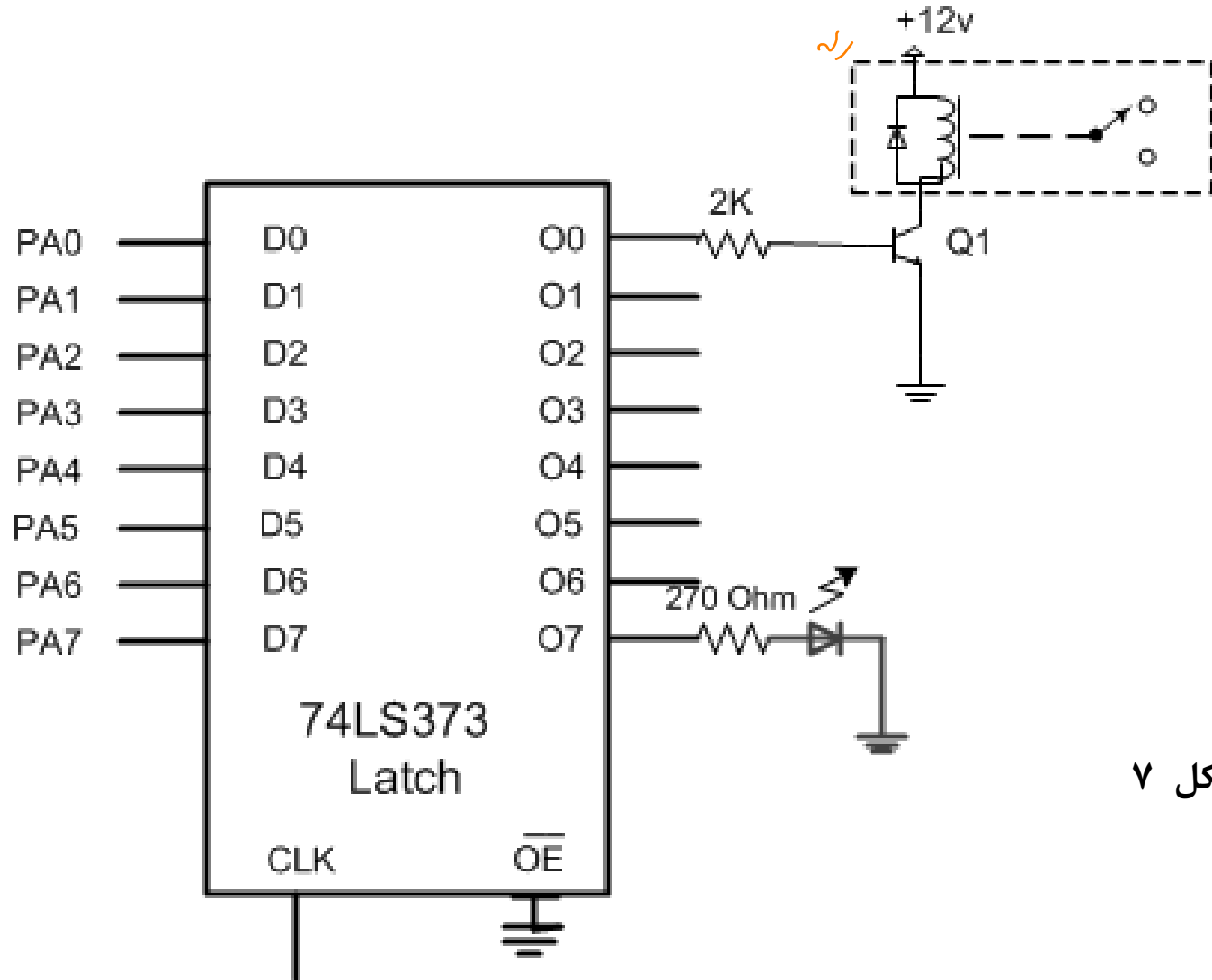


اتصال ۸ لچ ۷۴۳۷۳ به میکروکنترلر



شکل ۶

اتصال LED و رله به خروجی لچ



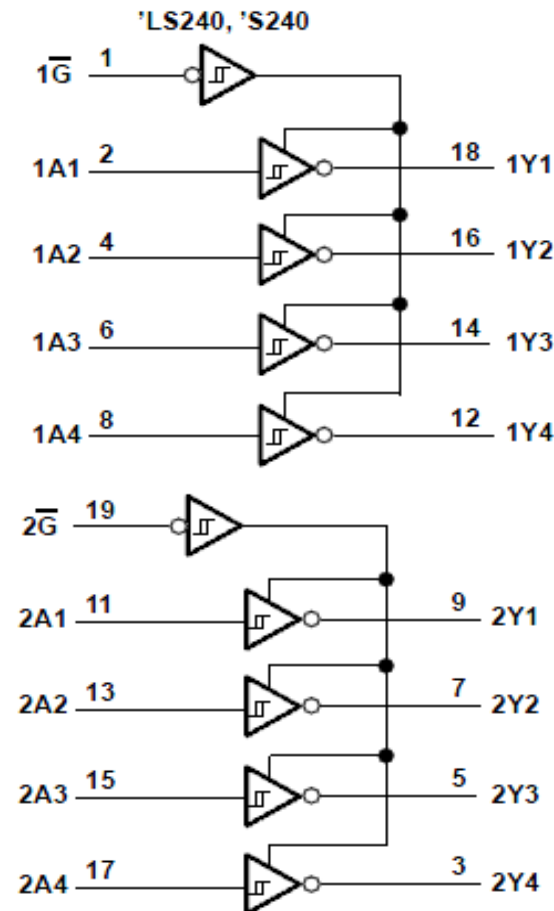
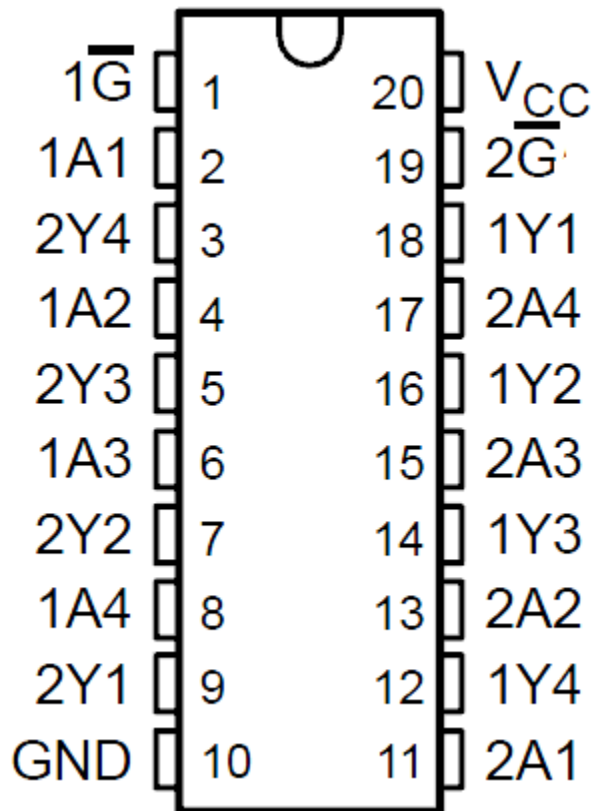
شکل ۷

تفاوت latch و buffer :
 در بافر داده لچ نمی‌شود؛ فقط زمانی که فعال است داده را عبور می‌دهد.
 در latch داده لچ می‌شود و منتظر کلاک می‌ماند تا داده بعدی را لچ کند.

پورت ورودی

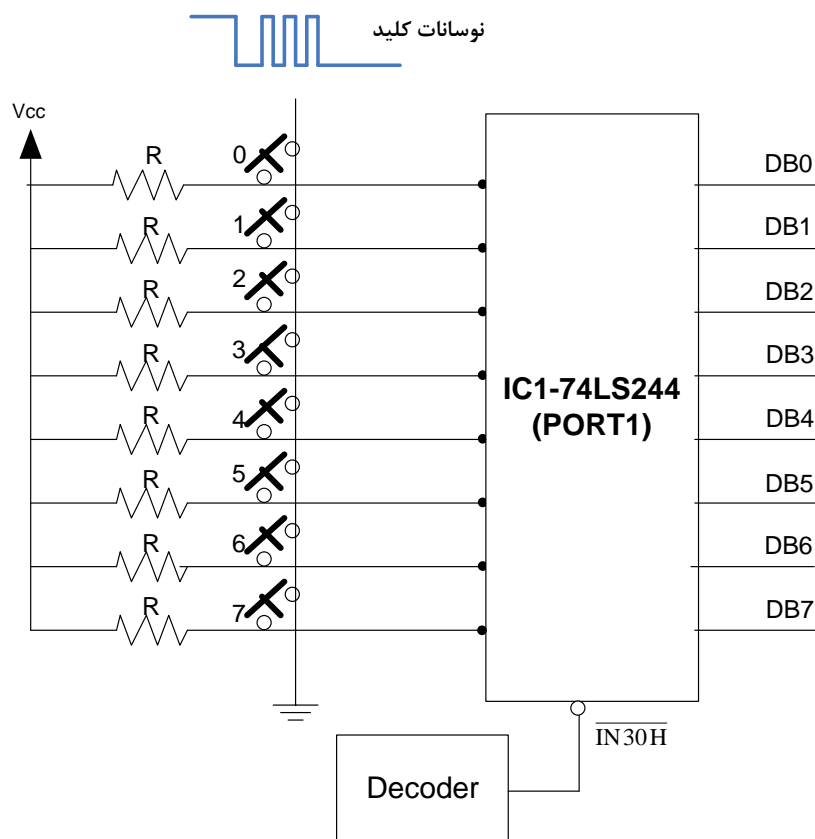
معمولاً به عنوان یک پورت ورودی از یک عدد بافر استفاده می‌شود. نمونه‌ای از بافر تراشه با مشخصات زیر است:

OCTAL BUFFERS AND LINE DRIVERS WITH 3-STATE OUTPUTS



شکل ۸

طراحی کیبورد سطری متصل به ریزپردازنده ۸۰۸۶



```

Loop1:  in al,      PORT1
        cmp      al, 0FFH
        jz       Loop1
        call    Delays20ms
        in      al, PORT1
        mov     cl, 07H
Loop2:  sal      al
        jnc     Label1
        dec     cl
        jnz     Loop2
Label1: mov     al, cl
    
```

```

DelayShort:
Again1:  mov  cx, LoopCounter1
        nop
        loop Again1
        ret
    
```

```

DelayLong:
Again2:  mov  bx, LoopCounter2
        call DelayShort
        dec  bx
        jnz  Again2
        ret
    
```

شکل ۲۷



برنامه خواندن کیبورد توسط ATmega16

; Detect the Pressed Key and return its number in R0

; Port Address Valid to data valid in PIN register > tPHL (74ls138) + tPZL (74244) + 1.5 Clocks (port Pd)= 41ns+30ns+1.5 Clocks

; Port Address Valid to data valid in PIN register > 41ns+30ns+1.5*62.5=164.75ns which is less than 3 clock pulses (each clock=62.5ns)

CALL BufferRead

BufferRead:

LDI R24, 0x00
OUT DDRA, R24 ; PORTA is Input
LDI R24, 0xFF
OUT DDRB, R24 ; PORTB is Output

LDI R24, 0x00 ;
OUT PORTB, R24 ;

NOP
NOP
NOP

LOOP1: LDI R20, 0x8 ; R20 will finally contain the No. of the pressed Key
IN R16, PINA ; Read Value from Input Buffer #1
CMP R16, 0xFF
BREQ LOOP1 ; If R16=0xFF means that no Key was Pressed

LOOP2: RCALL Delay20ms ; Call a 20ms Delay if any key was pressed
DEC R20; ;

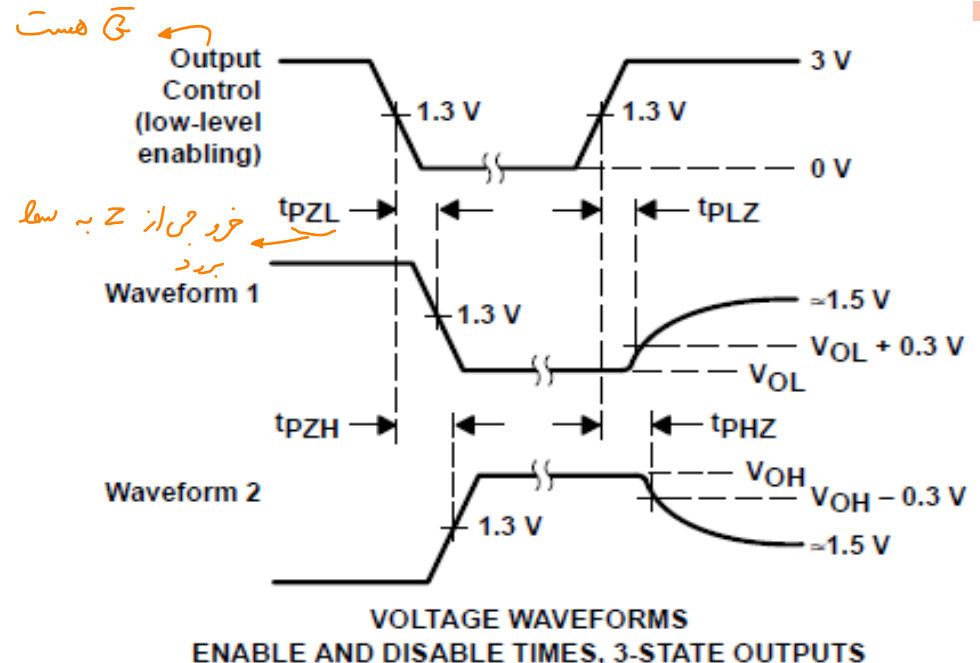
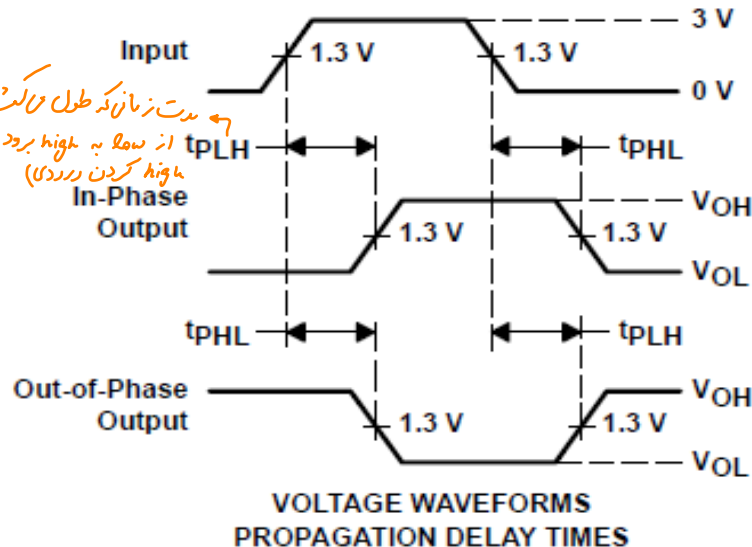
LSL R16 ; Shift left the Value read from Keyboard
BRCC LOOP3 ; Branch if Carry Flag is Cleared, so the pressed Key is detected
RJMP LOOP2

LOOP3: MOV R0, R20; ; Now R0 Contains the No. of pressed key

تراشه 74LS244

PARAMETER	TEST CONDITIONS	'LS240			'LS241, 'LS244			UNIT
		MIN	TYP	MAX	MIN	TYP	MAX	
t_{PLH}	$R_L = 667 \Omega, C_L = 45 \text{ pF}$		9	14		12	18	ns
t_{PHL}			12	18		12	18	
t_{PZL}	$R_L = 667 \Omega, C_L = 45 \text{ pF}$		20	30		20	30	ns
t_{PZH}			15	23		15	23	
t_{PLZ}	$R_L = 667 \Omega, C_L = 5 \text{ pF}$		10	20		10	20	ns
t_{PHZ}			15	25		15	25	

شکل ۹



پارامترهای زمانی مهم در خصوص 74LS244

- پارامترهای زمانی مهم 74LS244 عبارتند از:
- t_{PLH} : تاخیر انتشار برای تغییر وضعیت خروجی از Low به High با فرض اینکه پایه‌های Enable از قبل فعال بوده باشند.
 - t_{PHL} : تاخیر انتشار برای تغییر وضعیت خروجی از High به Low با فرض اینکه پایه‌های Enable از قبل فعال بوده باشند.
 - t_{PZL} : تاخیر انتشار برای رفتن خروجی از حالت شناور به وضعیت Low
 - t_{PZH} : تاخیر انتشار برای رفتن خروجی از حالت شناور به وضعیت High

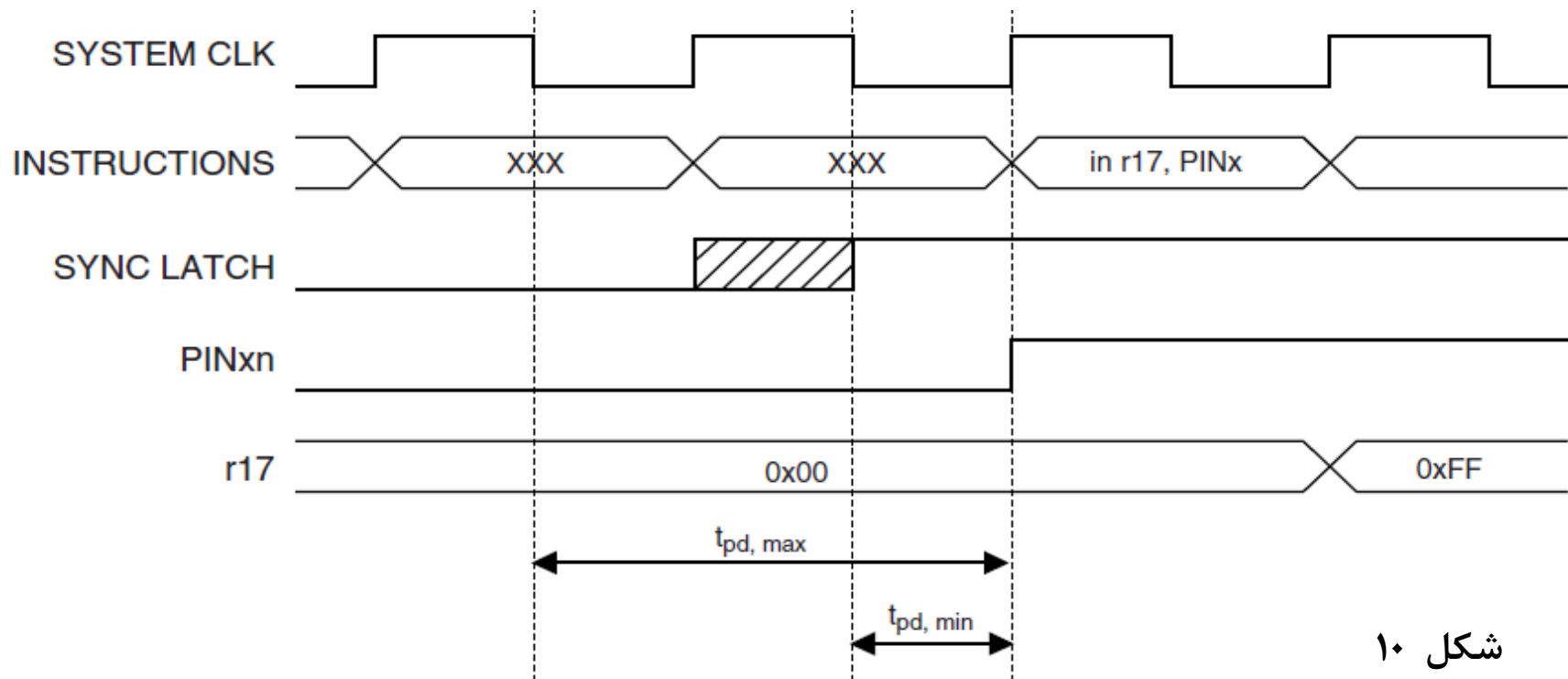
PARAMETER	TEST CONDITIONS	'LS240			'LS241, 'LS244			UNIT
		MIN	TYP	MAX	MIN	TYP	MAX	
t_{PLH}	$R_L = 667 \Omega, C_L = 45 \text{ pF}$		9	14		12	18	ns
t_{PHL}			12	18		12	18	
t_{PZL}	$R_L = 667 \Omega, C_L = 45 \text{ pF}$		20	30		20	30	ns
t_{PZH}			15	23		15	23	
t_{PLZ}	$R_L = 667 \Omega, C_L = 5 \text{ pF}$		10	20		10	20	ns
t_{PHZ}			15	25		15	25	

بعد از آنکه دستور زیر را بنویسیم 😊
 $t_{dl} + \max(t_{pZL}, t_{pZH}) + 1.5 \text{ clk}$
 فعال کردن پایه \rightarrow فعال کردن پایه تا اعمال وضعیت کلیه پایه ها به R0
 \hookrightarrow decoder (تاخیر باز)

IN, PINA

خواندن مقدار موجود بر روی یک پایه

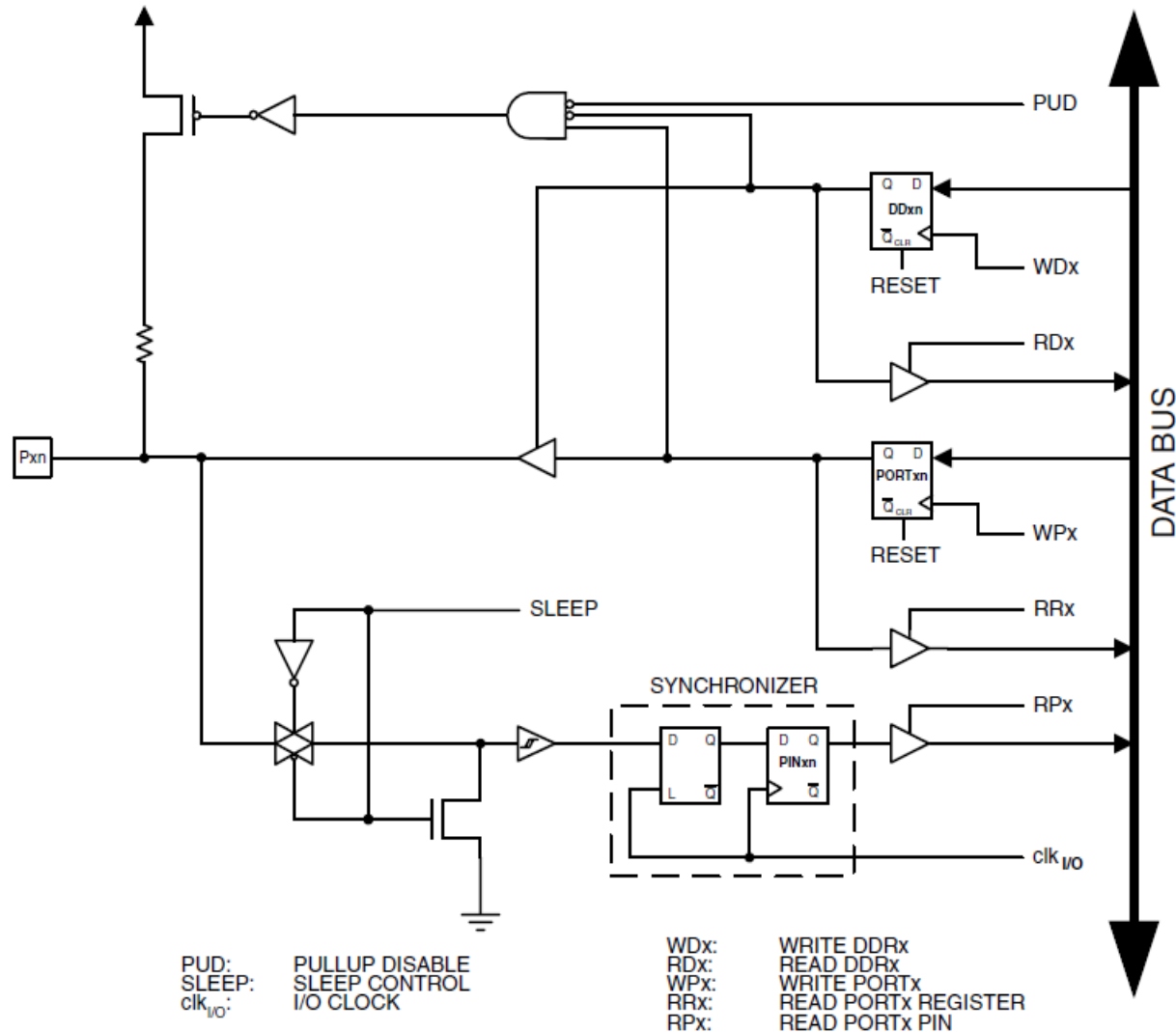
نمودار زیر، زمان بندی همگام سازی را هنگام خواندن یک مقدار قرار گرفته از خارج از میکروکنترلر بر روی پایه میکروکنترلر نشان می دهد.



شکل ۱۰

حداکثر به میزان ۱.۵ پالس ساعت زمان لازم است تا داده ورودی در ثبات PIN پورت قرار گیرد.

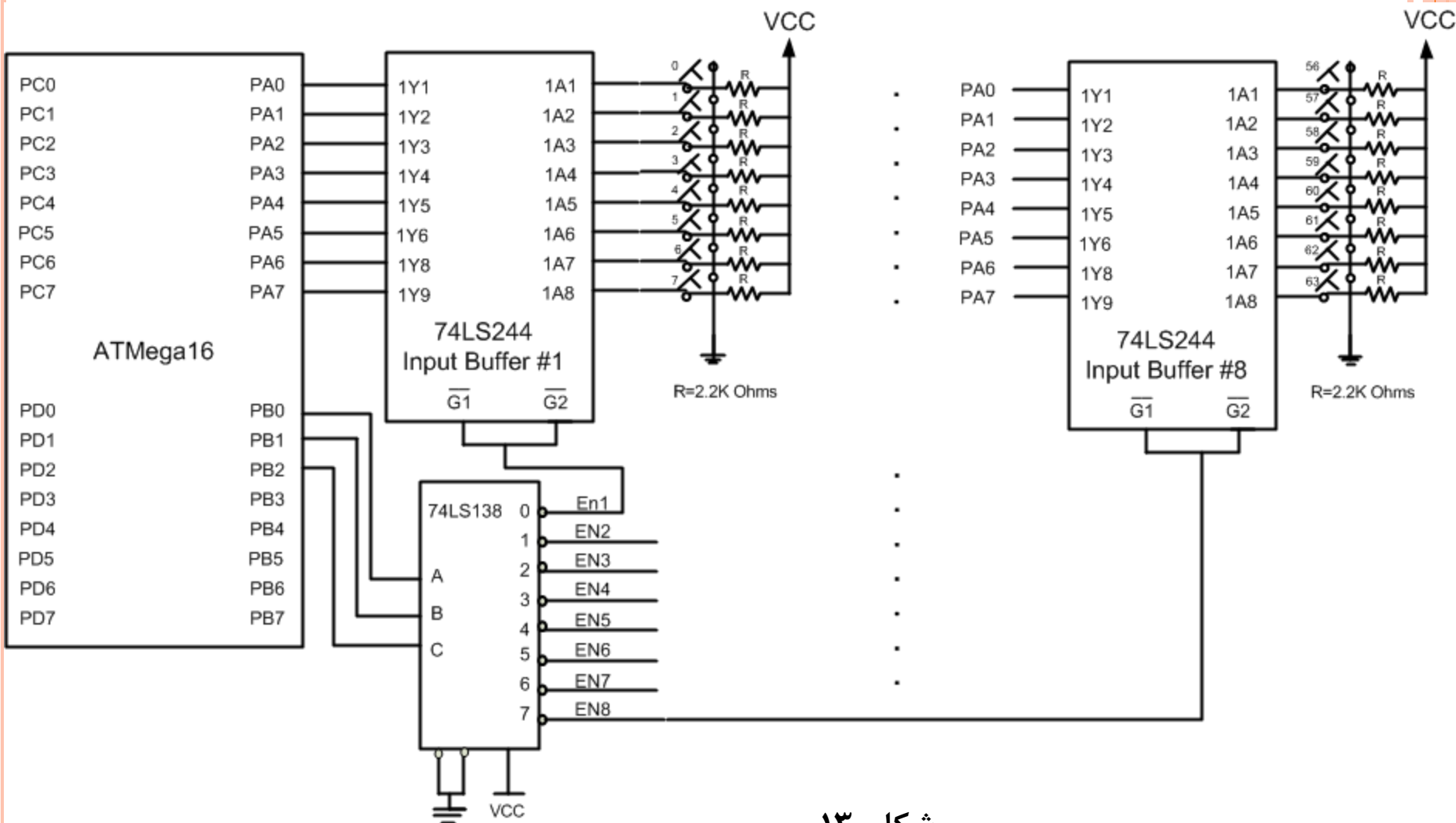
درگاه‌ها به عنوان ورودی/خروجی رقمی



شکل ۱۱

19

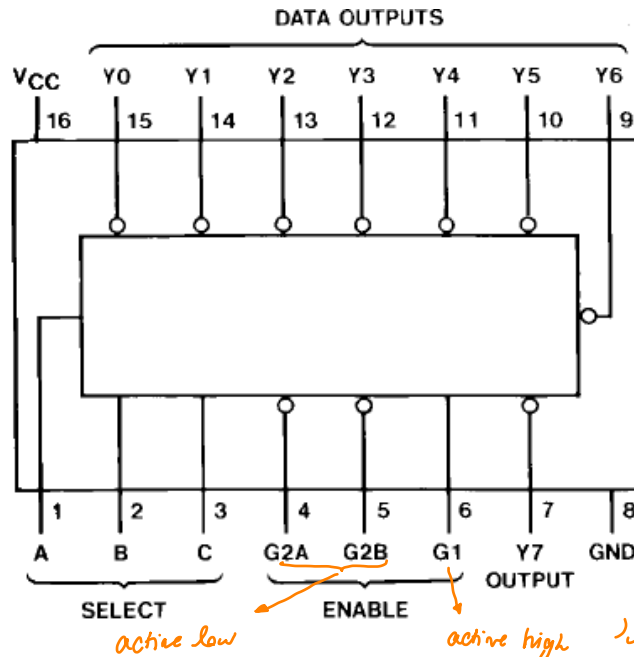
ارتباط میکروکنترلر با ۸ عدد پورت ورودی (بافر)



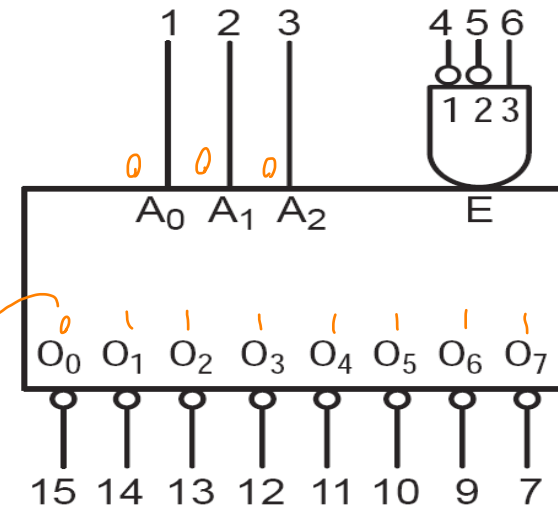
شکل ۱۳

74LS138: 1 of 8 decoder

شکل ۱۴



active-low
خروجی است



باید بهترین (t_{PLH} و t_{PHL}) در نظر بگیریم (خروجی وقت جواب عرضی ندارد)
هر دو اتفاق میفتد

AC CHARACTERISTICS ($T_A = 25^\circ\text{C}$)

Symbol	Parameter	Levels of Delay	Limits			Unit
			Min	Typ	Max	
t_{PLH} t_{PHL}	Propagation Delay Address to Output	2 2		13 27	20 41	ns
t_{PLH} t_{PHL}	Propagation Delay E_1 or E_2 Enable to Output	2 2		12 21	18 32	ns
t_{PLH} t_{PHL}	Propagation Delay E_3 Enable to Output	3 3		17 25	26 38	ns

پارامترهای زمانی مهم در خصوص 74LS138

پارامترهای زمانی مهم 74LS138 عبارتند از:

- t_{PLH} : تاخیر انتشار از زمان گذاشتن آدرس تا تغییر وضعیت خروجی از High به Low با فرض اینکه پایه‌های Enable از قبل فعال بوده باشند.
- t_{PHL} : تاخیر انتشار از زمان گذاشتن آدرس تا تغییر وضعیت خروجی از Low به High با فرض اینکه پایه‌های Enable از قبل فعال بوده باشند.

AC CHARACTERISTICS ($T_A = 25^\circ\text{C}$)

Symbol	Parameter	Levels of Delay	Limits			Unit
			Min	Typ	Max	
t_{PLH} t_{PHL}	Propagation Delay Address to Output	2 2		13 27	20 41	ns
t_{PLH} t_{PHL}	Propagation Delay E_1 or E_2 Enable to Output	2 2		12 21	18 32	ns
t_{PLH} t_{PHL}	Propagation Delay E_3 Enable to Output	3 3		17 25	26 38	ns



برنامه ارتباط با ۸ عدد پورت ورودی (بافر) و استفاده از دیکودر

; Read Values from Input ports. Values Returned in R16 to R23

; 74LS138: tPLH=20ns, tPLH: Decoder Propagation Delay Address to Output High

; 74LS138 : tPHL=41ns, tPHL: Decoder Propagation Delay Address to Output LOW (should wait so that all non-selected buffers are disabled)

; 74LS244: tPZL=30ns, tPZH=23ns,

; tPZL: Buffer output high Z to Low, tPZH: Buffer output high Z to High

; tPZL > tPZH, so tPZL is more critical

; Port Address Valid to data valid in PIN register > tPHL (74ls138) + tPZL (74244) + 1.5 Clocks (port Pd)= 41ns+30ns+1.5 Clocks

; Port Address Valid to data valid in PIN register > 41ns+30ns+1.5*62.5=164.75ns which is about 3 clocks

CALL BufferRead

BufferRead:

```
LDI R24, 0x00
OUT DDRA, R24 ; PORTA is Input
LDI R24, 0xFF
OUT DDRB, R24 ; PORTB is Output
```

```
LDI R24, 0x00
OUT PORTB, R24 ;
NOP
NOP
NOP
```

```
IN R16, PINA ; Read Value from Input Buffer #1
;Port Address Valid to data valid in PIN register  $\cong$  3 NOPs = 3Clocks> 164.75ns
```

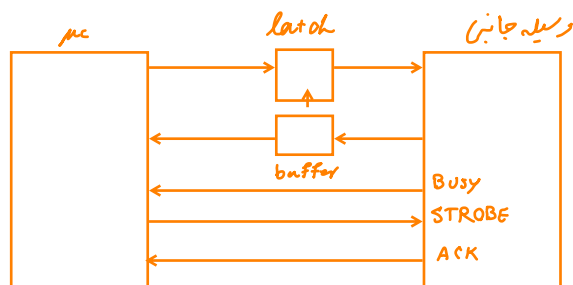
```
LDI R24, 0x01 ;
OUT PORTB, R24 ;
NOP
NOP
NOP
IN R17, PINA ; Read Value from Input Buffer #2
```

بافر #۱ را انتخاب کردم

مدیریت ارتباط ریزپردازنده با وسایل جانبی از طریق درگاه‌ها

- در هنگام انتقال باید ریزپردازنده با سرعت دستگاه جانبی همگام شود. برخی دستگاه‌های جانبی مثل چاپگرها نمی‌توانند داده را به سرعت ریزپردازنده دریافت کنند.
- از طرف دیگر دستگاه‌های دیگری مانند دیسک‌های سخت ممکن است داده را با سرعتی بیش از ریزپردازنده درخواست کنند.
- برای جلوگیری از تلف شدن داده لازم است هر دوی این حالات به خوبی کنترل شوند.
- بدون توجه به نوع پورت I/O مورد استفاده، موازی یا سریال، یک استراتژی برای همگام‌سازی و کنترل جریان داده از

طریق درگاه‌ها بین پردازنده و وسایل جانبی لازم است.



چگونه اطلاع داریم که وسیله جانبی داده دریافت؟ نیاز به یک روش handshake هست
BUSY: اگر یک بایر؛ وسیله جانبی نمی‌تونه داده دریافت کنه / STROBE: خط که بر وسیله آن وسیله جانبی
سیگنال میگیره براش داده جدید latch کرده / ACK = بر وسیله آن میگیره سیگنال وسیله جانبی داده رو برداشته
سه روش معمول کنترل و همگام نمودن جریان داده در پورت‌ها:

- I/O برنامه‌ریزی شده (Programmed I/O) با استفاده از روش سرکشی (Polling)

- وقفه (Interrupt)

- دسترسی مستقیم حافظه و وسایل جانبی به یکدیگر (DAM (Direct Memory Access)

I/O برنامه ریزی شده

- در روش سرکشی (Pooling) ریزپردازنده مرتباً می بایست به وسیله جانبی سرکشی کند و آمادگی او برای ارسال یا دریافت داده را بررسی کند. اینکار تمام وقت CPU را می گیرد.

- یک چاپگر با سرعت ۱۰۰ کاراکتر بر ثانیه قادر به چاپ هر کاراکتر در مدت ۱۰ میلی ثانیه است.

- اما فرض کنید روالی برای ارسال NoOfBytes بایت داده به چاپگر به صورت زیر در نظر گرفته شود:

LD R16, NoOfBytes ; No of bytes to be printed.

LD R20, 0x00 ; R20 as a counter

LOOP1: LD R21, Z+ ; Read data from memory [2]

OUT PORTA, R21 ; Send to Printer [1]

INC R20 [1]

SUB R20, R16 [1]

BRNE LOOP1 ; Check if Z Flag is 0 [1/2]

دفعه آخر حلقه، یک کلاک پالس طولی می کشد

- اعداد نشان داده شده در براکت بیانگر تعداد کلاک هایی است که هر دستور به طول می انجامد.

- با استفاده از این روتین ارسال هر کاراکتر به پرینتر به ۷ پالس کلاک نیاز دارد.

- 29 اگر زمان هر پالس ساعت 62.5ns باشد، برای ارسال هر کاراکتر به پرینتر زمان لازم است و لذا در هر ثانیه میکروکنترلر

می تواند ۲۲۸۵۷ کاراکتر را برای پرینتر ارسال نماید، در حالیکه چاپگر در هر ثانیه تنها می تواند ۱۰۰ کاراکتر را دریافت و چاپ نماید.

چاپگر موازی

جدول زیر سیگنال‌های چاپگر موازی معمولی را توصیف می‌کند.

یک کابل هادی با ۱۶ رشته سیم هادی برای چنین ارتباطی لازم است.

پین‌های ۱۹ تا ۳۰ به بدنه‌ی چاپگر متصل می‌شوند و برای شیلد کردن هر کدام از سیم‌های سیگنال به کار می‌روند.

توصیف	جهت	سیگنال	پین بازگشت	پین سیگنال
پالس STROBE برای خواندن داده‌ی ورودی. طول پالس در ترمینال دریافت باید بیش از 0.5 میکروثانیه باشد. سطح سیگنال معمولاً HIGH است و خواندن داده در سطح LOW این سیگنال انجام می‌شود.	In	\overline{STROBE}	19	1
این سیگنال‌ها به ترتیب اطلاعات اولین تا هشتمین بیت داده‌ی موازی را نشان می‌دهند. هر سیگنال به ازای 1 در سطح HIGH و به ازای 0 در سطح LOW قرار می‌گیرد.	In	DATA 1	20	2
	In	DATA 2	21	3
	In	DATA 3	22	4
	In	DATA 4	23	5
	In	DATA 5	24	6
	In	DATA 6	25	7
	In	DATA 7	26	8
	In	DATA 8	27	9
پالس به طول تقریبی 0.5 میکروثانیه بیان می‌دارد که داده دریافت شده و چاپگر آماده‌ی دریافت داده‌ی بعدی است.	Out	\overline{ACKNLG}	28	10

چاپگر موازی

توصیف	جهت	سیگنال	پین بازگشت	پین سیگنال
سطح HIGH بیان می کند که چاپگر نمی تواند داده ای دریافت کند که در موارد زیر اتفاق می افتد:				
• در حین دریافت داده	Out	BUSY	29	11
• در حین عملیات چاپ				
• در وضعیت OFF-LINE				
• در وضعیت خطای چاپگر				
سطح HIGH بیان می کند که کاغذ چاپگر تمام شده است.	Out	PE	30	12
این سیگنال بیان می کند که چاپگر در وضعیت خواسته شده است.	Out	SLCT	-	13
قرار دادن این سیگنال در سطح LOW باعث می شود که بعد از چاپ به طور خودکار کاغذ یک خط به پایین رود.	In	\overline{AUTO} $\overline{FEED XT}$	-	14
بدون استفاده	-	NC	-	15
سطح منطقی زمین	-	0V	-	16
زمین بدنه ی چاپگر که مستقل از سطح منطقی زمین است.	-	CHASIS -GND	-	17
بدون استفاده	-	NC	-	18
سیگنال بازگشت زوج های به هم تابیده شده که در سطح LOW قرار می گیرند.	-	GND	-	19 تا 30

چاپگر موازی

توصیف	جهت	سیگنال	پین بازگشت	پین سیگنال
قرار گرفتن این سیگنال در سطح LOW کنترلر چاپگر را به وضعیت اولیه‌اش بازنشانی می‌کند و بافر آن را پاک می‌کند. عرض این پالس در گیرنده باید بیش از 50 میکروثانیه باشد.	In	\overline{INIT}	-	31
در وضعیت‌های زیر سطح این سیگنال LOW می‌شود:	Out	\overline{ERROR}	-	32
<ul style="list-style-type: none"> وضعیت اتمام کاغذ وضعیت OFF-LINE وضعیت خطا 				
همانند پین‌های 19 تا 30	-	GND	-	33
بدون استفاده	-	NC	-	34
از طریق مقاومت $4.7K\Omega$ به +5v متصل می‌شود.			-	35
ورود داده به چاپگر تنها زمانی امکان‌پذیر است که این سیگنال در سطح LOW قرار گیرد.	In	$\overline{SLCT IN}$	-	36

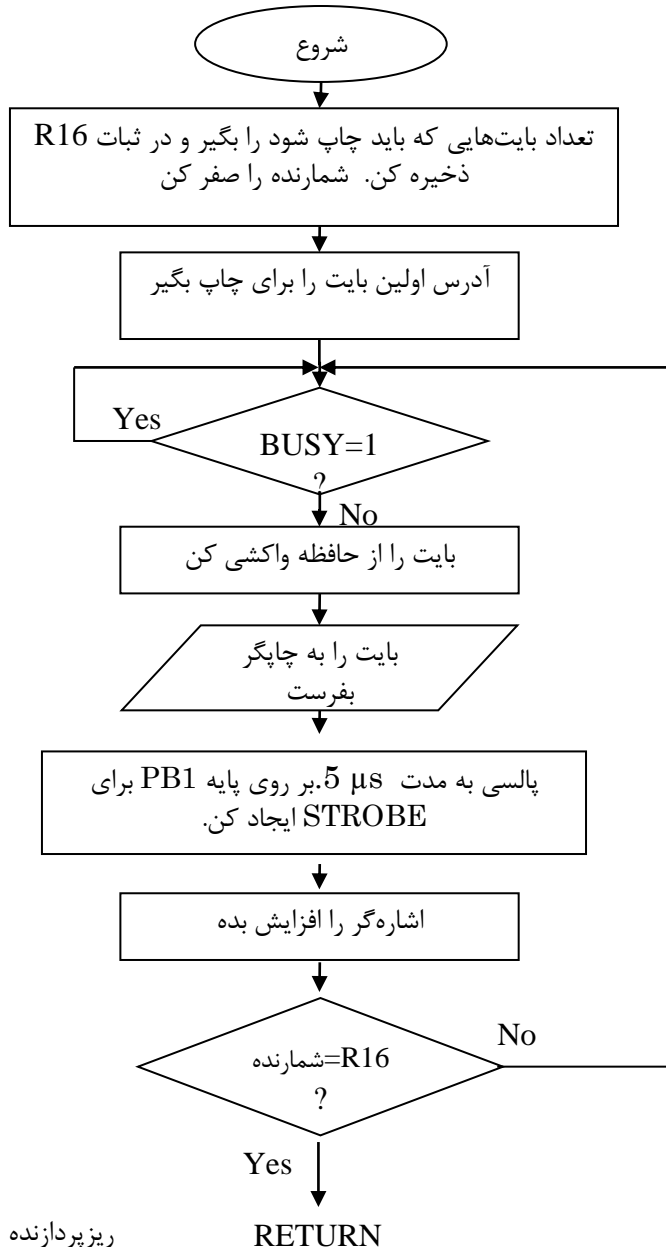
چاپگر موازی

- هشت سیم داده به نام‌های DATA1 تا DATA8 وجود دارند. وقتی سیگنال \overline{STROBE} به مدت 0.5 میکروثانیه یا بیشتر در سطح LOW قرار گیرد، چاپگر داده موجود بر پین‌های داده را لچ می‌کند.
- دو سیگنال کنترلی دیگر نیز برای چاپگر فراهم شده‌اند که BUSY و \overline{ACKNLG} نام‌گذاری شده‌اند. با توجه به جدول BUSY یک سیگنال ACTIVE-HIGH است که بیان می‌کند چاپگر مشغول چاپ یک کاراکتر است، خطایی پیش آمده است یا در وضعیت OFF-LINE قرار دارد.
- \overline{ACKNLG} یک پالس ACTIVE-LOW است که بعد از دریافت و چاپ یک کاراکتر از طرف چاپگر اعمال می‌شود. به تفاوت این دو سیگنال توجه کنید که BUSY یک سیگنال حساس به سطح است در حالیکه \overline{ACKNLG} حساس به لبه است.
- سیگنال‌های BUSY، \overline{ACKNLG} و \overline{STROBE} مجموعه‌ای از سیگنال‌های Handshaking را بین چاپگر و CPU تشکیل می‌دهند. CPU دست خود را از طریق سیگنال \overline{STROBE} دراز می‌کند و می‌گوید "داده حاضر است"؛ چاپگر با سیگنال \overline{ACKNLG} به او پاسخ می‌دهد و می‌گوید "آن را دریافت کردم، می‌توانی داده‌ی بعدی را برایم بفرستی".

روش سرکشی (ادامه)

- شکل ۱۴ فرآیند لازم برای انتقال داده از کامپیوتر به چاپگر را نشان می‌دهد. در این برنامه فرض شده که داده‌های آماده‌ی چاپ در خانه‌های متوالی در یک بخش از حافظه SRAM میکروکنترلر (که اصطلاحاً بافر گفته می‌شود) قرار گرفته‌اند. اشاره‌گری به ابتدای این بافر اشاره می‌کند و شمارنده‌ای تعداد بایت‌هایی که باید برای چاپ فرستاده شوند را شمارش می‌کند.
- بلوک تصمیم‌گیری “ $BUSY = 1?$ ” یک حلقه‌ی سرکشی را تشکیل می‌دهد که در آن CPU مرتباً پرچم BUSY چاپگر را بررسی می‌کند و زمانی که در سطح LOW قرار گرفت CPU داده‌ی بعدی را آماده کرده و به چاپگر می‌فرستد و اگر داده‌های دیگری همچنان باقی مانده باشد مجدداً در حلقه‌ی سرکشی قرار می‌گیرد.
- قبل از نوشتن برنامه‌ای که عملیات تشریح شده را انجام دهد، سخت افزار لازم برای این ارتباط را فراهم می‌کنیم.

روش سرکشی (ادامه)



شکل ۱۴

فلوچارت ارسال داده برای پرینتز همراه با handshaking

روش سرکشی (ادامه)

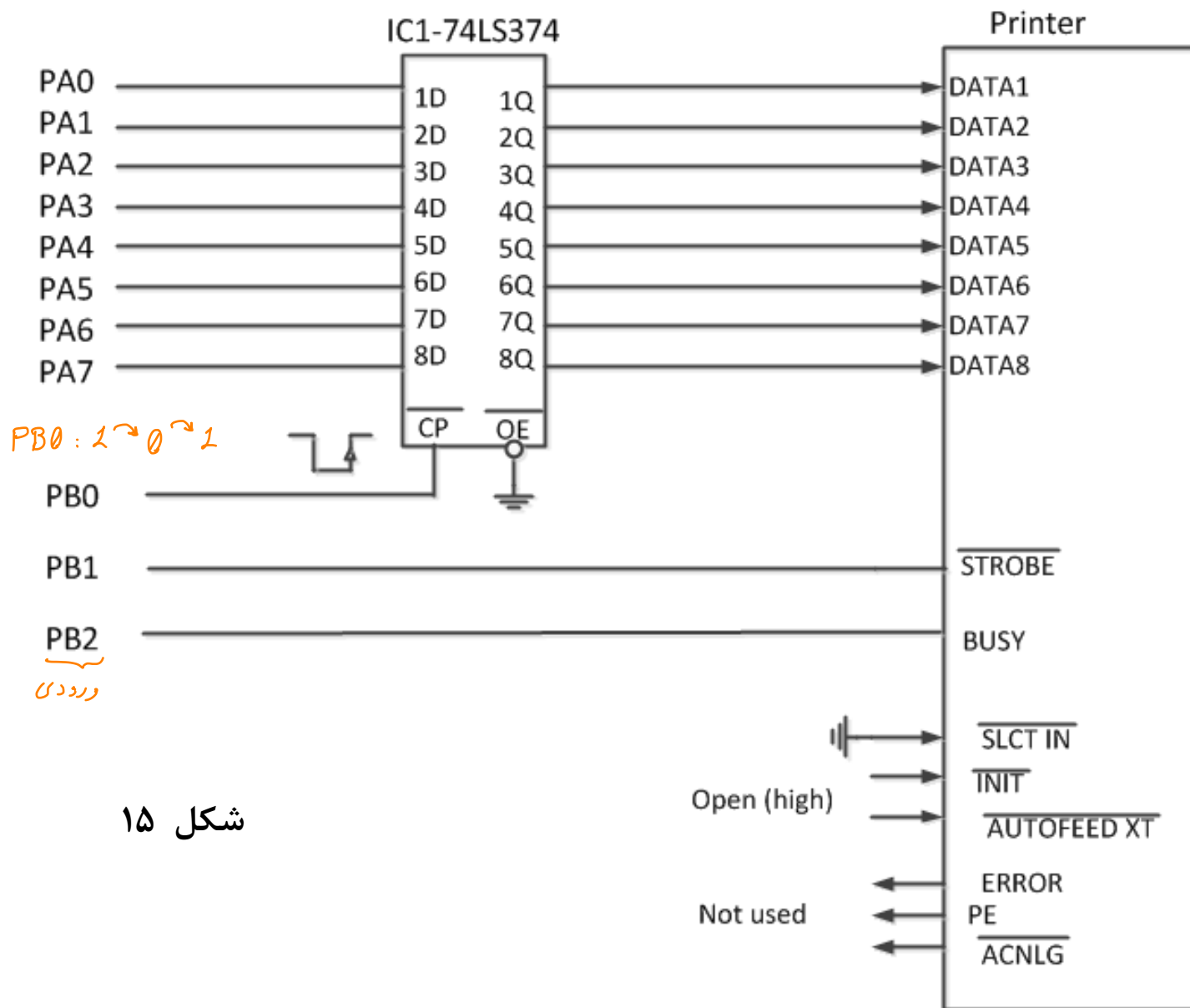
در شکل ۱۵ مدار مربوطه نشان داده شده است.

یک لچ هشت بیتی برای نگه داشتن داده در پین‌های ورودی چاپگر به کار رفته است. کلاک این لچ از سیگنال DB0 درگاه B می‌آید. سیگنال PB2 سیگنال BUSY را چک می‌کند و PB1 سیگنال STROBE را تولید می‌کند.

استفاده از سیگنال BUSY برای همگام کردن بین ریزپردازنده و وسیله جانبی است.

چون انتقال داده به چاپگر تحت کنترل نرم‌افزار انجام می‌شود، این روش را **I/O برنامه‌ریزی شده** یا روش سرکشی گویند.

روش سرکشی (ادامه)

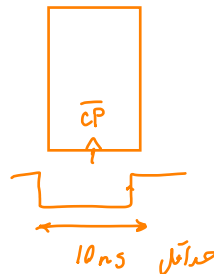


شکل ۱۵



برنامه ارتباط با پرینتر

; Read 16 Values from Address 0100H and write them to the printer.
 ; R30=01, R31=00H (Z=0x100)
 ; R16: Number of characters to be printed
 ; T1=input to latch STB setup time=10ns ; STB means CP input
 ; T2=Input to latch STB Hold time=25ns ; T3=STB Low time=25ns



Printer:	CALL	Printer	
	LDI	R24, 0xFFH	
	OUT	DDRA, R24	; PORTA is Output
	LDI	R24, 0x03H	; PB0, PB1 output, PB2 input
	OUT	DDRB, R24	;
	LDI	R20, 0x00	; R20 as a counter for printed bytes
:			
LOOP1:	<i>pooling</i> ← SBIC	PINB, 02	; Skip if PB2 (Busy signal) is clear (Ready) [1/3] +
	RJMP	LOOP1	; Pooling of BUSY signal [3] +
	LD	R21, Z+	; [2] *
	OUT	PORTA, R21	; <i>clk latch</i> [1] *
	CBI	PORTB, 0	; CP=0 [1]
	SBI	PORTB, 0	; CP=1 T1, T2 and T3 times are satisfied [1]
	CBI	PORTB, 1	; STROBE=0 0.5Us=8*62.5ns [1]
	NOP		[1]
	NOP		[1]
	NOP		[1]
	NOP		[1]
	NOP		[1]
	NOP		[1]
	NOP		[1]
	NOP		[1]
	SBI	PORTB, 01	; STROBE=1 [1]
	INC	R20	; [1]
	CP	R20, R16	; [1]
	BRNE	LOOP1	; Check if Z Flag is 0 [1/2]
	RET		

روش سرکشی (ادامه)

• در برنامه‌ی نوشته شده، سه دستور با علامت “+” مشخص شده‌اند که حلقه‌ی سرکشی را تشکیل می‌دهند و اعداد نوشته شده در براکت بیانگر تعداد کلاک‌های لازم برای اجرای این دستورات است که جمعا ۴ کلاک (62.5ns) برای کلاک (16MHz) برای حلقه‌ی سرکشی مورد نیاز است (۲۵۰ نانوثانیه).

• از آنجایی که چاپگر به مدت زمان بیشتری برای انجام چاپ نیاز دارد این حلقه بارها تکرار می‌شود تا چاپگر آماده‌ی دریافت کاراکتر بعدی شود. در این مثال CPU باید حلقه را $10\text{ms}/250\text{ns}=40000$ بار تکرار کند.

• حلقه‌ی سرکشی راندمان پایینی دارد چون CPU مدت زمان زیادی را باید منتظر چاپگر باشد تا آماده‌ی دریافت کاراکتر بعدی شود در حالیکه واکنشی بایت جدید و تحویل دادن هر کاراکتر به پرینتر (دستورالعمل‌های مشخص شده با علامت *) به $3 \times 62.5\text{ns}=187.5\text{ns}$ زمان نیاز دارد.

• در این مدت CPU دستور دیگری جز سرکشی بیت وضعیت چاپگر را انجام نمی‌دهد. اگر CPU کار دیگری جز چاپ این کاراکترها نداشته باشد، این مساله اهمیتی ندارد ولی اگر بخواهیم CPU را در یک سیستم Multi tasking راه‌اندازی کنیم که در آن CPU چندین کار را با هم انجام می‌دهد، آنگاه روش سرکشی برای کار با دستگاه‌های جانبی کندی چون چاپگر مناسب نیست.

روش سرکشی (ادامه)

- ATmega16 می‌تواند با سرعت خیلی بیشتری از آنچه در مثال چاپگر مشاهده شد، داده را ارسال کند.

- برای محاسبه‌ی این نرخ حداکثر فرض می‌کنیم دستگاه جانبی بعد از یک بار اجرای حلقه‌ی سرکشی آماده‌ی دریافت داده‌ی بعدی است.

- در این مثال ۲۲ کلاک برای هر بار بررسی سیگنال BUSY، ارسال داده جدید، تولید پالس STROBE، افزایش اشاره‌گر و بررسی آن است که در یک سیستم با کلاک 16MHz به $22 \times 62.5\text{ns} = 1375\text{ns} = 1.375\mu\text{s}$ زمان نیاز دارد و لذا نرخ ارسال داده ۷۲۷۲

کاراکتر بر ثانیه خواهد بود.

روش سرکشی (ادامه)

- یک سیستم مبتنی بر ریزپردازنده نوعا چندین دستگاه جانبی دارد که شامل فلاپی دیسک، چاپگر، مودم، ترمینال نمایشگر ویدئو و ... است.

- استفاده از روش سرکشی لازم می‌دارد که برای هر کدام از این دستگاه‌ها یک سیگنال BUSY/READY موجود باشد.

- شکل زیر سیگنال‌های BUSY/READY برای هشت دستگاه جانبی را نشان می‌دهد که از طریق یک پورت هشت بیتی خوانده می‌شود.

Video Terminal	Modem	DAC	ADC	Plotter	Daisy Wheel Printer	Line Printer	Floppy disk
7	6	5	4	3	2	1	0

شکل ۱۶- سیگنال‌های BUSY/READY برای هشت دستگاه جانبی

- زمانیکه چندین دستگاه جانبی به ریزپردازنده متصل می‌شوند زمان پاسخ‌گویی به آنها مهم می‌شود.
- زمان پاسخ‌گویی از لحظه‌ای است که برای یک وسیله خط BUSY آن در حالت LOW قرار می‌گیرد تا زمانیکه CPU به آن سرویس‌دهی می‌کند.



برنامه ارتباط با پرینتر

; Polling of 8 devices
; Busy Line of all 8 devices are connected to PB

```
LOOP1:      SBIS      PINB, 0      ; Skip next inst. if FD is not Ready      [1/3]
            RCALL     FD           ;                               [3]
            SBIS      PINB, 1      ; Skip next inst. if LP is not Ready      [1/3]
            RCALL     LP           ;                               [3]
            SBIS      PINB, 2      ; Skip next inst. if DWP is not Ready      [1/3]
            RCALL     DWP          ;                               [3]
            SBIS      PINB, 3      ; Skip next inst. if PLOT is not Ready      [1/3]
            RCALL     PLOT         ;                               [3]
            SBIS      PINB, 4      ; Skip next inst. if ADC is not Ready      [1/3]
            RCALL     ADC          ;                               [3]
            SBIS      PINB, 5      ; Skip next inst. if DAC is not Ready      [1/3]
            RCALL     DAC          ;                               [3]
            SBIS      PINB, 6      ; Skip next inst. if MOD is not Ready      [1/3]
            RCALL     MOD          ;                               [3]
            SBIS      PINB, 7      ; Skip next inst. if TERM is not Ready      [1/3]
            RCALL     TERM         ;                               [3]
            JMP       LOOP1
```

I/O وقفه‌گرا

- مشکل واقعی خود روش سرکشی است.
- اگرچه پیاده‌سازی سخت‌افزاری و نرم‌افزاری آن ساده است ولی راندمان پایینی در استفاده از منابع کامپیوتری دارد.
- زمانیکه چندین دستگاه جانبی وجود دارد این روش ممکن است کاملاً نارضایت‌بخش باشد.
- روش‌های بر پایه‌ی وقفه و DMA جایگزین مناسب روش سرکشی هستند.

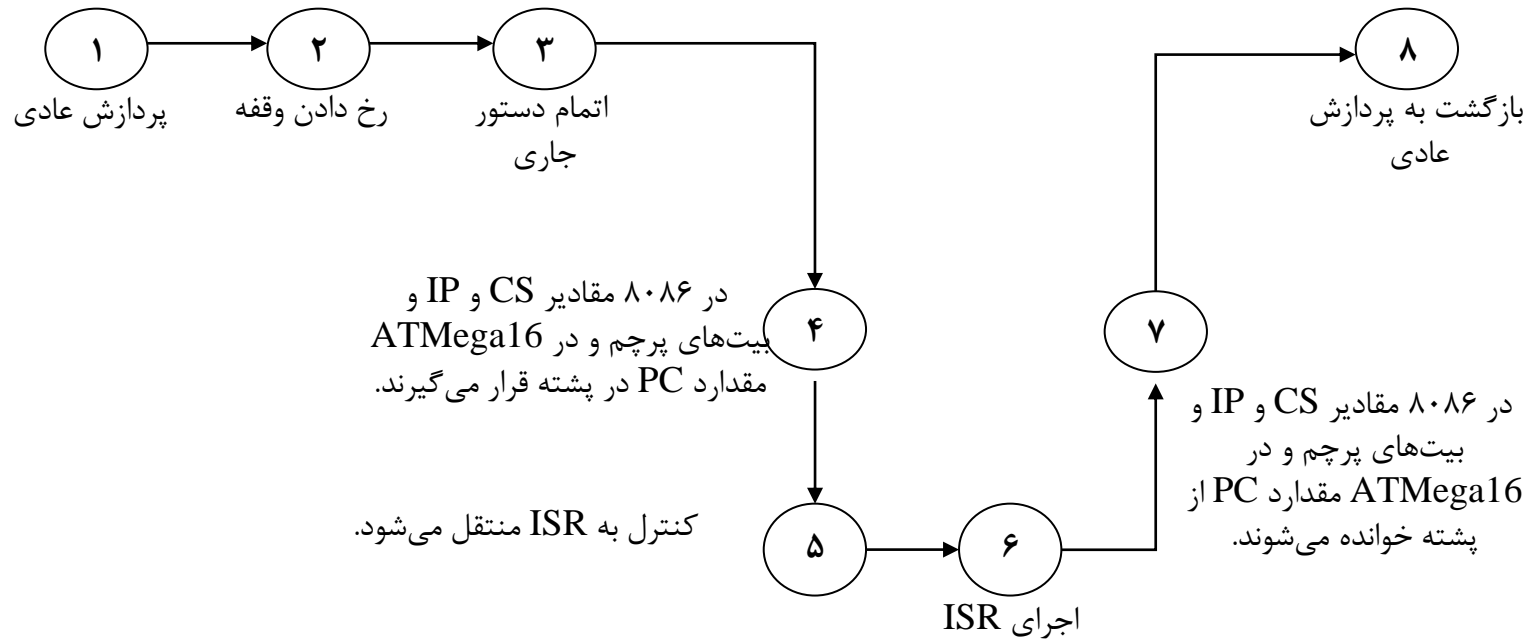
I/O وقفه‌گرا (ادامه)

- مشکل اساسی در برقراری ارتباط بین ریزپردازنده و دستگاه جانبی در این است که پروسسور نمی‌داند کی دستگاه آماده است.
- روش سرکشی برای حل این مشکل پیشنهاد چک کردن مداوم بیت وضعیت را داد که مشکلات آن را بررسی کردیم.
- بهتر است دستگاه جانبی به CPU اطلاع دهد که آماده است و سپس CPU شروع به سرویس‌دهی آن دستگاه کند. این اساس وقفه در سیستم‌های مبتنی بر ریزپردازنده است.
- در انتهای اجرای هر دستوری CPU خط وقفه را چک می‌کند و اگر این ورودی و بیت فعال‌ساز وقفه فعال باشند، کنترل برنامه به مکان خاصی از حافظه که روتین سرویس وقفه (Interrupt Service Routine:ISR) نامیده می‌شود منتقل می‌شود.

I/O وقفه‌گرا (ادامه)

- شکل ۱۷ پاسخ CPU به یک وقفه را نشان می‌دهد.
- در مرحله ۱ فرض شده CPU پردازش عادی خود را انجام می‌دهد.
- در مرحله ۲ دستگاه جانبی وقفه تولید کرده است. بعد از اتمام دستورالعمل جاری در مرحله ۳، در ۸۰۸۶ محتوای رجیسترهای CP، IP و بیت‌های پرچم و در ATmega16 محتوای PC در پشته قرار می‌گیرند (مرحله ۴) و مرحله ۵ انتقال کنترل برنامه به ISR است.
- بعد از اجرای روتین وقفه در مرحله ۶، آنچه به پشته فرستاده شده بود از پشته بازیابی می‌شود و مرحله ۷ به اتمام می‌رسد. در مرحله ۸ پردازش عادی CPU ادامه می‌یابد.
- اگر فرض کنیم $1.25\mu s$ زمان برای پاسخ‌دهی به وقفه و فراهم کردن داده برای دستگاه جانبی لازم باشد و نیز نرخ دریافت داده‌ی چاپگر ۱۰۰ کاراکتر در ثانیه باشد ($1s/100=10000\mu s$) باشد آنگاه $9998.75\mu s$ برای CPU وقت باقی می‌ماند تا به پردازش عادی خود بپردازد. بدین ترتیب CPU می‌تواند چندین عمل را همزمان انجام دهد.

I/O وقفه‌گرا (ادامه)



شکل ۱۷- فرآیند انجام شونده بعد از رخ دادن وقفه

انواع وقفه‌ها در ۸۰۸۶

جدول ۱

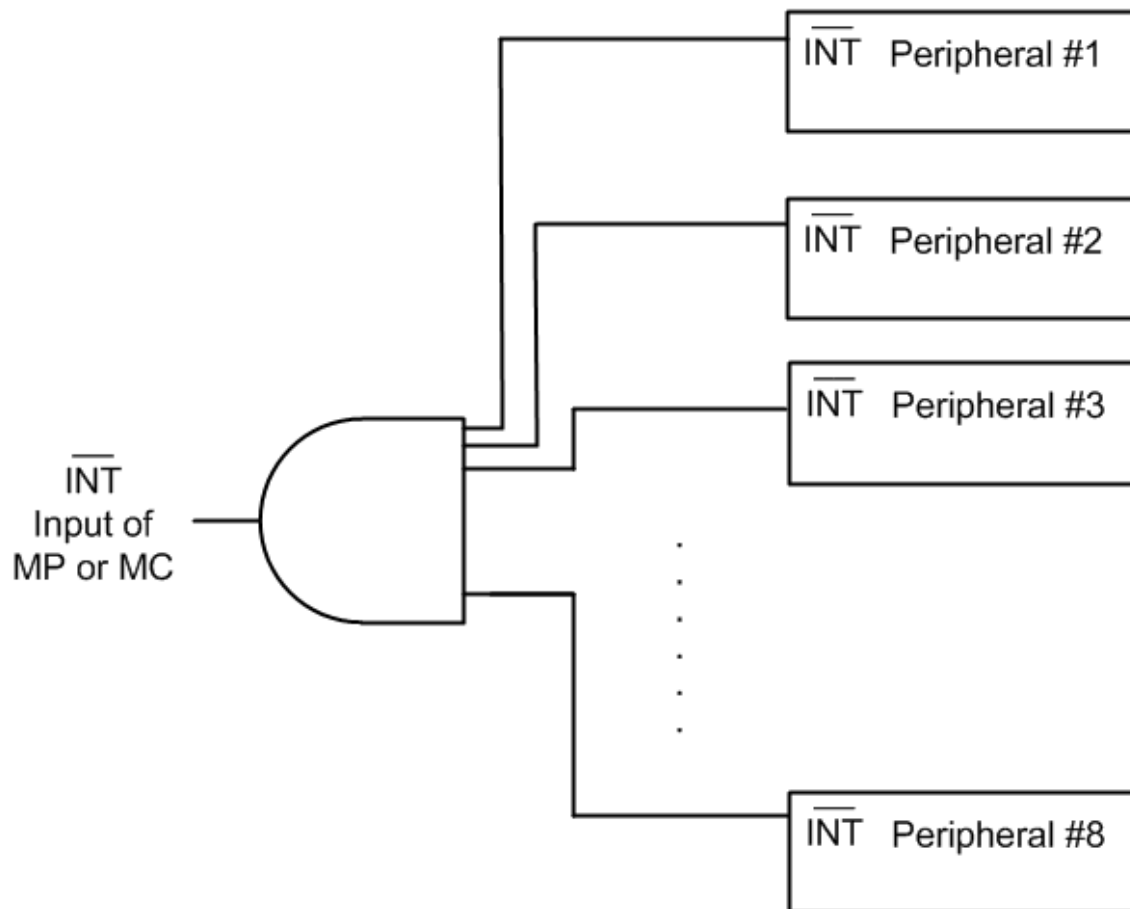
نام وقفه	نحوه‌ی راه‌اندازی	قابل پوشش؟	چگونگی تحریک	اولویت	سیگنال تصدیق وقفه	آدرس جدول بردار	تأخیر وقفه
NMI	سخت‌افزار خارجی	خیر	لبه‌ی پالس، حداقل ۲ پالس پالس‌ساعت	2	ندارد	00008H-0000BH	دستور جاری + ۵۱ (پالس‌ساعت)
INTR	سخت‌افزار خارجی	بلی از طریق IF	سطح بالا تا زمانی که تأیید شود	3	\overline{INTA}	$n^* \times 4$	دستور جاری + ۶۱ (پالس‌ساعت)
int n	داخلی، نرم-افزاری	خیر	ندارد	1	ندارد	$n \times 4$	۵۱ (پالس‌ساعت)
int 3	داخلی، نرم-افزاری	خیر	ندارد	1	ندارد	0000CH-0000FH	۵۲ (پالس‌ساعت)
into	داخلی، نرم-افزاری	بلی از طریق OF	ندارد	1	ندارد	00010H-00013H	۵۳ (پالس‌ساعت)
تقسیم بر 0	داخلی CPU	خیر	ندارد	1	ندارد	00000H-00003H	۵۱ (پالس‌ساعت)
تک‌گامی	داخلی CPU	بلی از طریق TF	ندارد	4	ندارد	00004H-00007H	۵۱ (پالس‌ساعت)

- همه‌ی انواع وقفه محتوای ثابت‌های CP و IP و نیز بیت‌های پرچم را در پشته قرار می‌دهند، به علاوه محتوای IF و TF نیز پاک می‌شود.
- n^* یک عدد ۸ بیتی است که حین پالس دوم INTA از طریق باس داده خوانده شده و توسط وسیله وقفه دهنده تامین می‌شود.
- چنانچه دستوری بنام INT اجرا شود وقفه نوع ۳ یعنی INT 3 اجرا می‌شود.

بعضی از وقفه‌های ۸۰۸۶

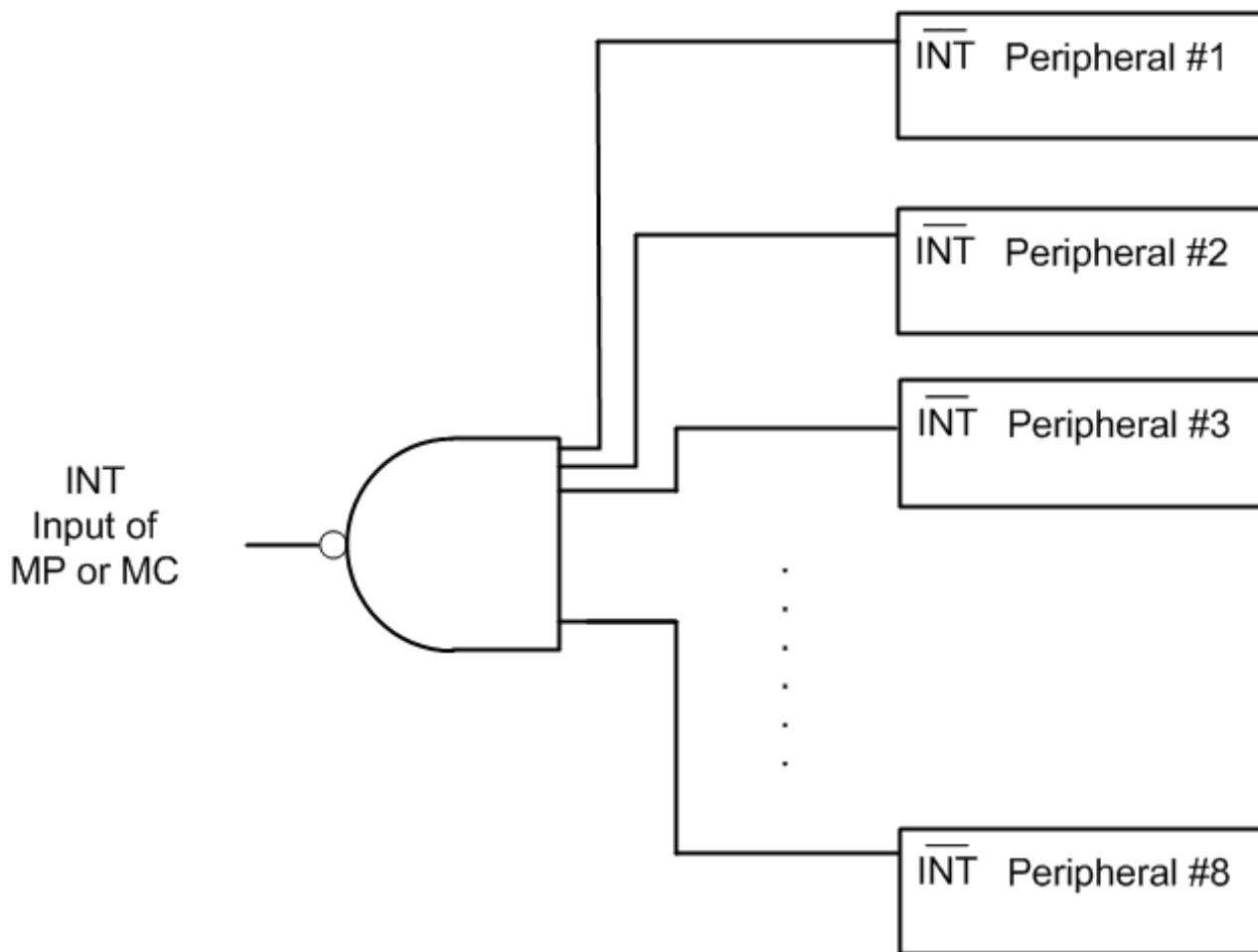
- چنانچه تقسیم بر صفر رخ دهد وقفه نوع 0 یعنی وقف INT 0 رخ می‌دهد.
- پردازش تک گام وقفه نوع 1 است.
- چنانچه پایه NMI فعال شود وقفه نوع 2 یعنی INT 2 اجراء می‌شود.
- چنانچه دستوری بنام INT اجرا شود وقفه نوع 3 یعنی INT 3 اجرا می‌شود.
- INTO وقفه نوع 4 است یعنی INT4 اجرا می‌شود.

دریافت تقاضای وقفه چند وسیله جانبی و تولید یک سیگنال وقفه



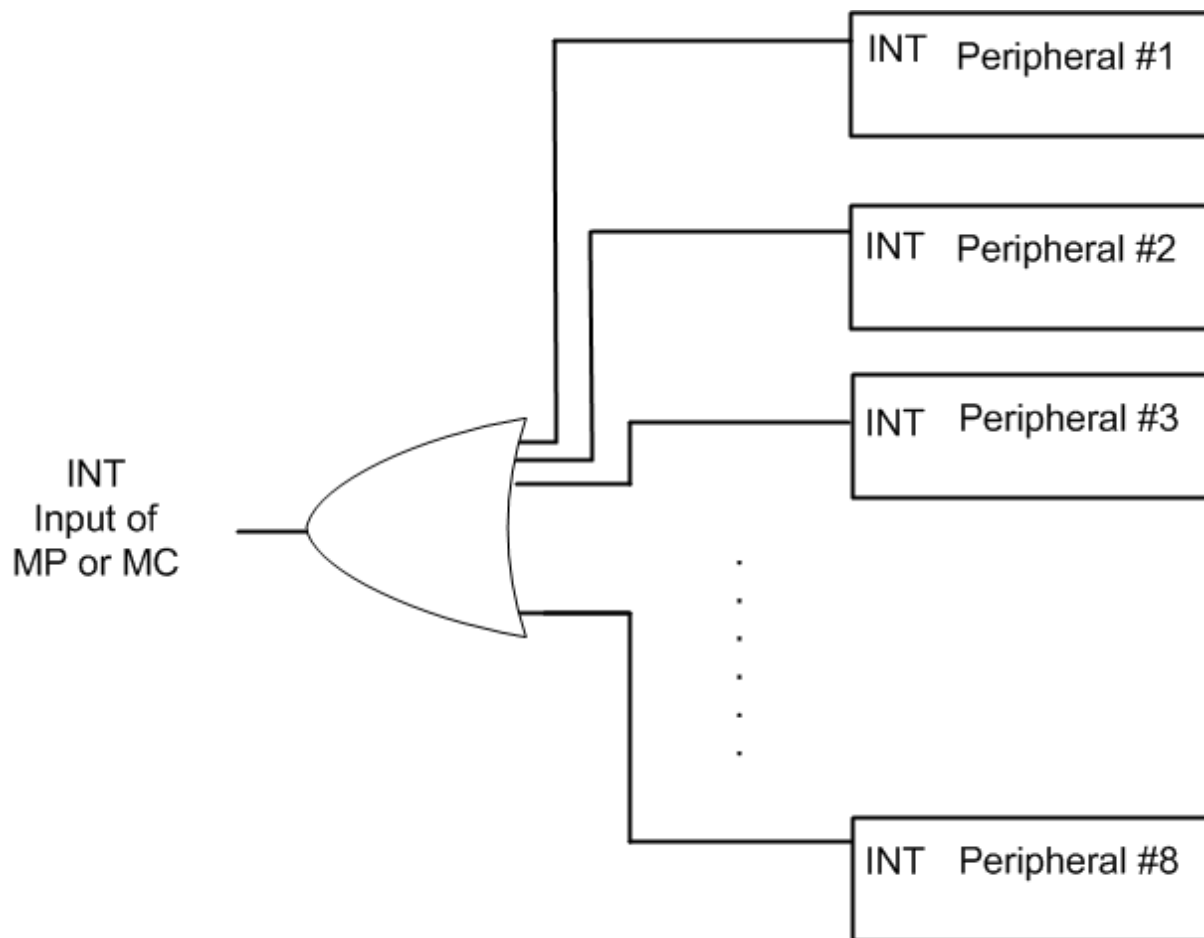
شکل ۱۸- ترکیب خروجی تقاضای وقفه وسایل جانبی از نوع active low و تبدیل آنها به یک خروجی جهت اعمال به ورودی وقفه active low پردازنده

دریافت تقاضای وقفه چند وسیله جانبی و تولید یک سیگنال وقفه



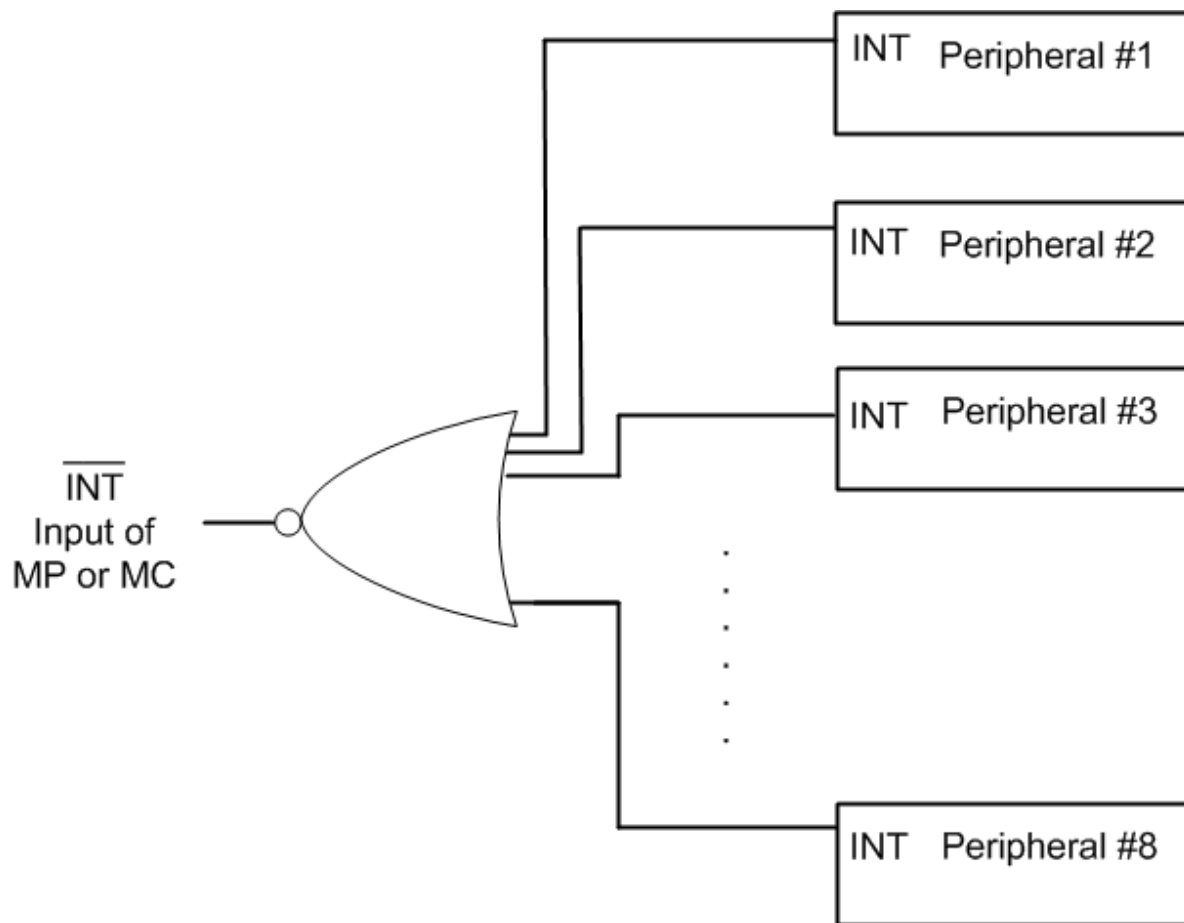
شکل ۱۹- ترکیب خروجی تقاضای وقفه وسایل جانبی از نوع active low و تبدیل آنها به یک خروجی جهت اعمال به ورودی وقفه active high پردازنده

دریافت تقاضای وقفه چند وسیله جانبی و تولید یک سیگنال وقفه



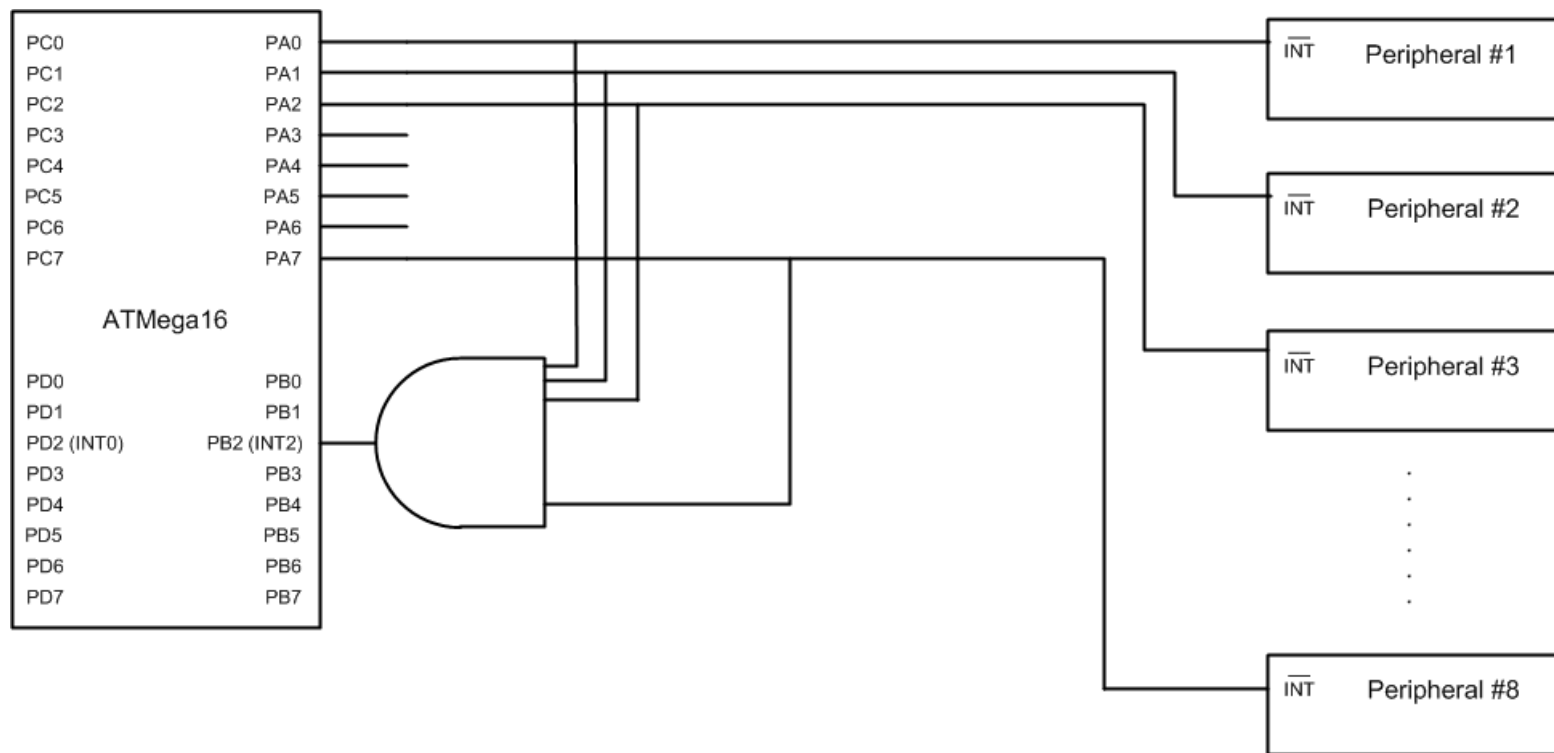
شکل ۲۰- ترکیب خروجی تقاضای وقفه وسایل جانبی از نوع active high و تبدیل آنها به یک خروجی جهت اعمال به ورودی وقفه active high پردازنده

دریافت تقاضای وقفه چند وسیله جانبی و تولید یک سیگنال وقفه



شکل ۲۱- ترکیب خروجی تقاضای وقفه وسایل جانبی از نوع active high و تبدیل آنها به یک خروجی جهت اعمال به ورودی وقفه active low پردازنده

شناسایی وسیله جانبی وقفه دهنده و ارائه سرویس به آن

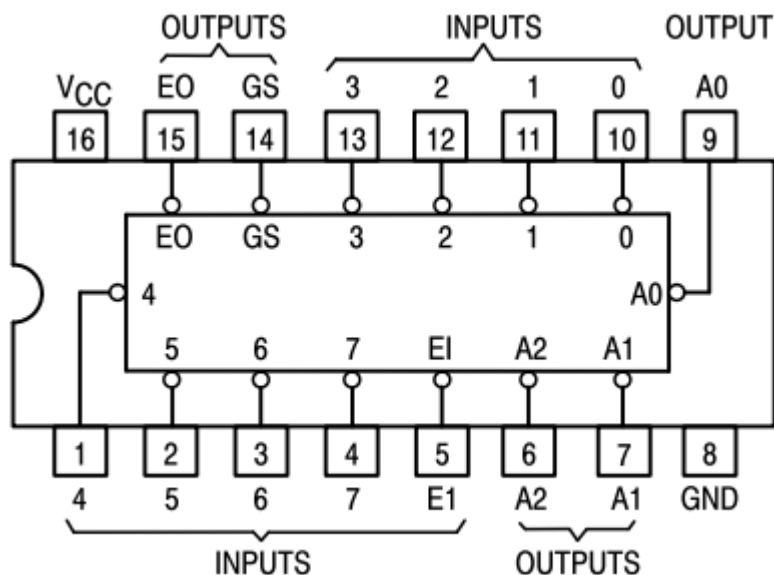


شکل ۲۲- ترکیب خروجی تقاضای وقفه وسایل جانبی از نوع active low و تبدیل آنها به یک خروجی جهت اعمال به ورودی وقفه active low پردازنده و اتصال کلیه خروجی‌های تقاضای وقفه به یکی از پورت‌های پردازنده

- پس از دریافت وقفه روی پایه INT2، در روتین وقفه بیت‌های پورت A را یکی یکی چک می‌کنیم تا وسیله وقفه دهنده شناسایی شود.
- در اینجا اگر دو وسیله همزمان تقاضای وقفه کنند، می‌توان آنها را به راحتی شناسایی و یکی یکی سرویس‌دهی کرد.

تراشه کدگذار اولویت گذار 74148

- تراشه 74LS148 یک نوع کدگذار اولویت گذار (Priority Encoder) است.
- این تراشه را می توان برای اولویت دهی وقفه ها استفاده نمود.



شکل ۲۳- پایه های تراشه 74LS148

تراشه 74148

INPUTS									OUTPUTS				
EI	0	1	2	3	4	5	6	7	A2	A1	A0	GS	EO
H	X	X	X	X	X	X	X	X	H	H	H	H	H
L	H	H	H	H	H	H	H	H	H	H	H	H	L
L	X	X	X	X	X	X	X	L	L	L	L	L	H
L	X	X	X	X	X	X	L	H	L	L	H	L	H
L	X	X	X	X	X	L	H	H	L	H	L	L	H
L	X	X	X	X	L	H	H	H	L	H	H	L	H
L	X	X	X	L	H	H	H	H	H	L	L	L	H
L	X	X	L	H	H	H	H	H	H	L	H	L	H
L	X	L	H	H	H	H	H	H	H	H	L	L	H
L	L	H	H	H	H	H	H	H	H	H	H	L	H

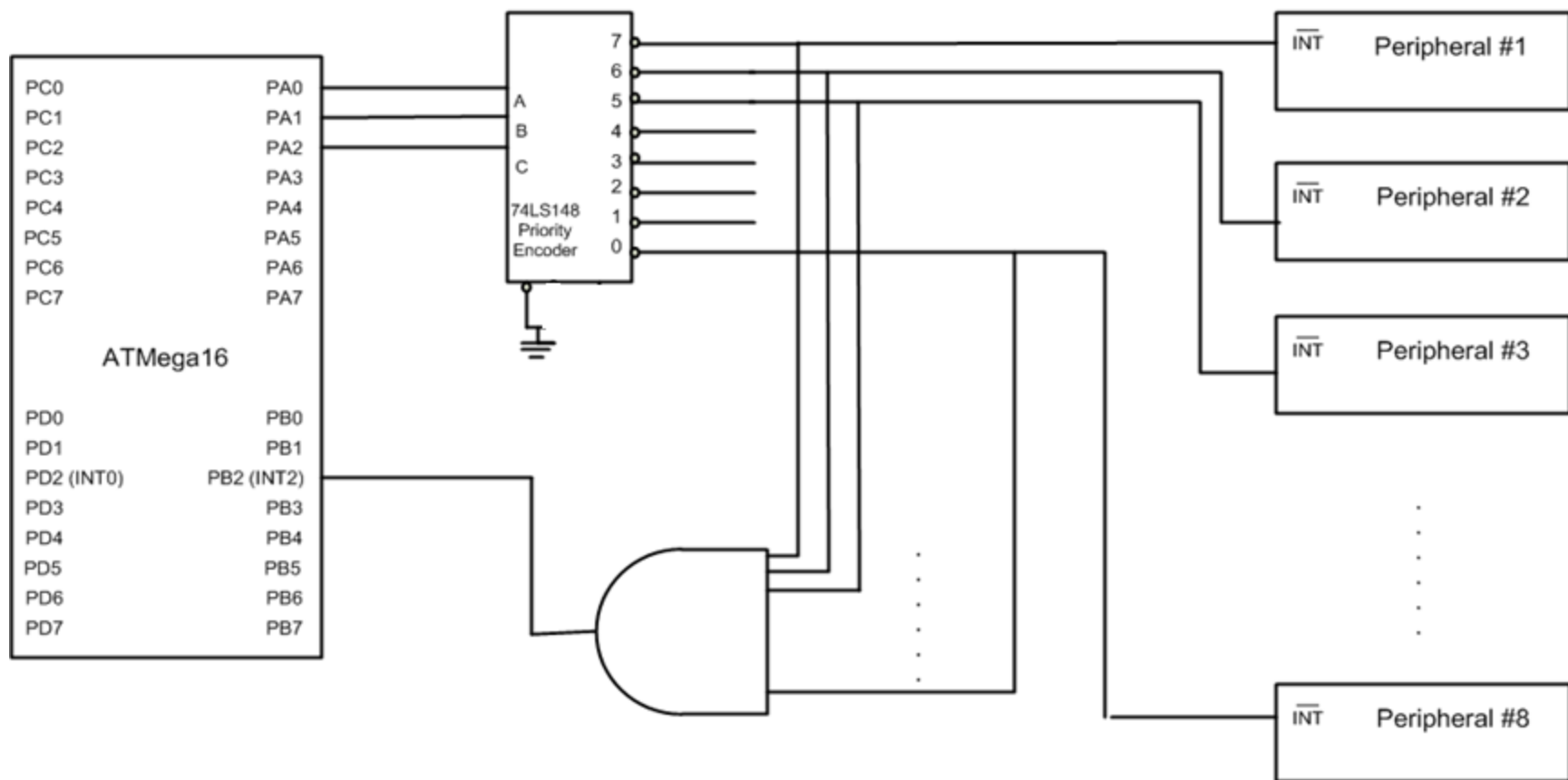
شکل ۲۴- بلوک دیاگرام عملیاتی تراشه 74LS148

پارامترهای زمانی تراشه 74148

Symbol	From (Input)	To (Output)	Waveform	Limits			Unit	Test Conditions
				Min	Typ	Max		
tPLH	1 thru 7	A0, A1, or A2	In-phase output		14	18	ns	C _L = 15 pF, R _L = 2.0 kΩ
tPHL					15	25		
tPLH	1 thru 7	A0, A1, or A2	Out-of-phase output		20	36	ns	
tPHL					16	29		
tPLH	0 thru 7	EO	Out-of-phase output		7.0	18	ns	
tPHL					25	40		
tPLH	0 thru 7	GS	In-phase output		35	55	ns	
tPHL					9.0	21		
tPLH	EI	A0, A1, or A2	In-phase output		16	25	ns	
tPHL					12	25		
tPLH	EI	GS	In-phase output		12	17	ns	
tPHL					14	36		
tPLH	EI	EO	In-phase output		12	21	ns	
tPHL					28 30	40 45		
								(LS148) (LS748)

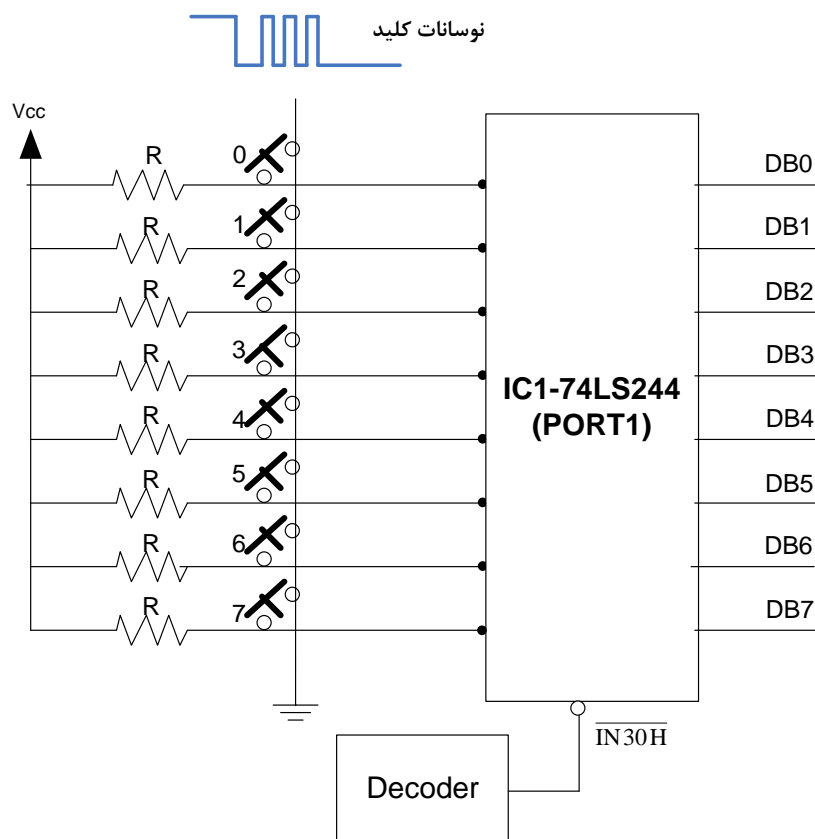
شکل ۲۵- پارامترهای زمانی تراشه 74LS148

اولویت دهی به تقاضای وقفه وسایل جانبی توسط تراشه 74148



- وسایل وقفه دهنده را به ترتیب اولویت از بالا به پایین قرار می‌دهیم.
- چنانچه دو وسیله همزمان وقفه دهند، انکودر وسیله با اولویت بیشتر را کدگذاری و به ۳ بیت کم ارزش پورت A تحویل می‌دهد. در نتیجه وسیله با اولویت بیشتر زودتر سرویس داده می‌شود.

طراحی کیبورد سطری متصل به ریزپردازنده ۸۰۸۶



```

Loop1:  in al, PORT1
        cmp al, 0FFH
        jz  Loop1
        call Delays20ms
        in  al, PORT1
        mov cl, 07H
Loop2:  sal al
        jnc Label1
        dec cl
        jnz Loop2
Label1: mov al, cl
    
```

```

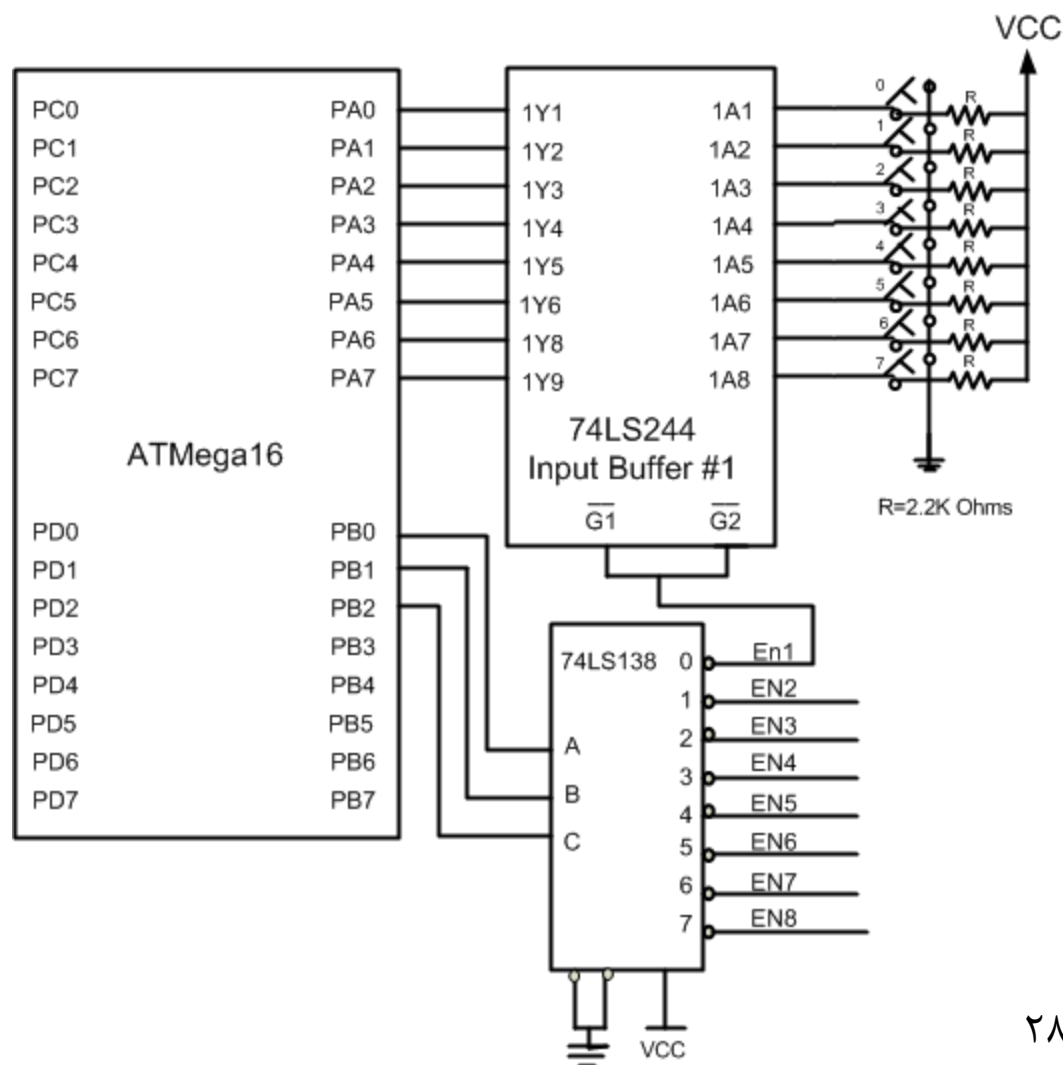
DelayShort:
Again1:  mov cx, LoopCounter1
        nop
        loop Again1
        ret
    
```

```

DelayLong:
Again2:  mov bx, LoopCounter2
        call DelayShort
        dec bx
        jnz Again2
        ret
    
```

شکل ۲۷

طراحی کیبورد سطری متصل به میکروکنترلر



شکل ۲۸



برنامه خواندن کیبورد توسط ATmega16

; Detect the Pressed Key and return its number in R0

; Port Address Valid to data valid in PIN register > tPHL (74ls138) + tPZL (74244) + 1.5 Clocks (port Pd)= 41ns+30ns+1.5 Clocks

; Port Address Valid to data valid in PIN register > 41ns+30ns+1.5*62.5=164.75ns which is less than 3 clock pulses (each clock=62.5ns)

CALL BufferRead

BufferRead:

LDI R24, 0x00
OUT DDRA, R24 ; PORTA is Input
LDI R24, 0xFF
OUT DDRB, R24 ; PORTB is Output

LDI R24, 0x00 ;
OUT PORTB, R24 ;

NOP
NOP
NOP

LOOP1: LDI R20, 0x8 ; R20 will finally contain the No. of the pressed Key
IN R16, PINA ; Read Value from Input Buffer #1
CMP R16, 0xFF
BREQ LOOP1 ; If R16=0xFF means that no Key was Pressed

LOOP2: RCALL Delay20ms ; Call a 20ms Delay if any key was pressed
DEC R20; ;

LSL R16 ; Shift left the Value read from Keyboard
BRCC LOOP3 ; Branch if Carry Flag is Cleared, so the pressed Key is detected
RJMP LOOP2

LOOP3: MOV R0, R20; ; Now R0 Contains the No. of pressed key