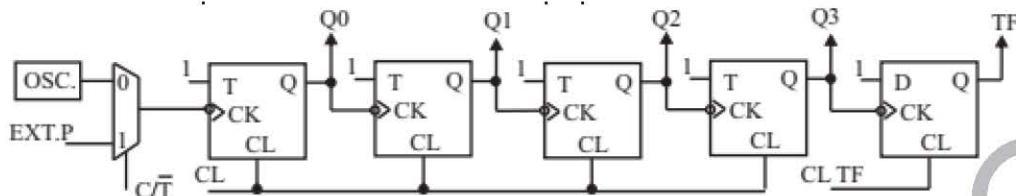


## CHƯƠNG 7: BỘ ĐỊNH THÌ(TIMERS)

### 7.1 GIỚI THIỆU CHUNG

Bộ định thi được ứng dụng tạo thời gian định thi, thời gian trễ(delay time), tạo hàm, dạng sóng, đếm các sự kiện, biến cố...

Hình 7.1 minh họa sơ đồ khối cơ bản của bộ định thi/dếm 4 bit.



Hình 7.1: Sơ đồ khối cơ bản bộ định thi/dếm 4 bit

Xung CK kích mạch đếm có thể chọn từ mạch dao động chuẩn OSC, hoặc từ xung đếm ngoài EXT.P. Tín hiệu C/T lựa chọn nguồn xung CK cho mạch đếm, C/T=0 chọn CK=OSC. Mô thức(mode) định thi, C/T=1 chọn CK=EXT.P. Mô thức đếm từ xung ngoài. Ban đầu khi mới khởi động, ngõ ra mạch đếm được xóa Q3Q2Q1Q0=0000. Khi có xung CK, mạch đếm tăng 1 giá trị, tới xung CK thứ 15 ngõ ra mạch đếm Q3Q2Q1Q0=1111. Khi có xung CK thứ 16 mạch đếm bị tràn, ngõ ra trở về 0000, đồng thời cờ báo tràn TF được đặt TF=1, báo mạch đếm bị tràn.

Hệ VDK AVR có từ 1 đến 6 bộ định thi trong chip(onchip) ký hiệu Timer0, Timer1, ... Timer5. ATmega324P có 3 bộ định thi ký hiệu Timer0, Timer1, Timer2. Với Timer0, Timer2 8 bit, Timer1 16 bit.

Để đơn giản trong trình bày, ta quy ước gọi AVR ATmega324P bằng ký hiệu MCU324P và các bộ định thi/dếm tương ứng là Timer0, Timer1, Timer2.

Hoạt động của các Timer trong MCU324P được quản lý và điều khiển bởi các thanh ghi nằm trong vùng địa chỉ I/O như tóm tắt trong Bảng 7.1

Bảng 7.1: Tóm tắt các thanh ghi Timer

Timer	Địa chỉ MEM	Địa chỉ I/O	Ký hiệu thanh ghi	Truy xuất BIT	Chức năng
Timer0	0x44	0x24	TCCR0A	Có	Điều khiển Timer0 A
	0x45	0x25	TCCR0B	Có	Điều khiển Timer0 B
	0x46	0x26	TCNT0	Không	Bộ đếm Timer0
	0x47	0x27	OCR0A	Không	So sánh ngõ ra Timer0 A
	0x48	0x28	OCR0B	Không	So sánh ngõ ra Timer0 B
	0X35	0X15	TIFR0	Có	Cờ báo ngắt Timer0
	0x6E		TIMSK0	Có	Che ngắt Timer0
Timer1	0x80		TCCR1A	Có	Điều khiển Timer1 A
	0x81		TCCR1B	Có	Điều khiển Timer1 B
	0x82		TCCR1C	Có	Điều khiển Timer1 C
	0x84		TCNT1L	Không	Bộ đếm Timer1 byte thấp
	0x85		TCNT1H	Không	Bộ đếm Timer1 byte cao
	0x86		ICR1L	Không	Bắt ngõ vào Timer1 byte thấp
	0x87		ICR1H	Không	Bắt ngõ vào Timer1 byte cao
	0x88		OCR1AL	Không	So sánh ngõ ra Timer1 A byte thấp
	0x89		OCR1AH	Không	So sánh ngõ ra Timer1 A byte cao
	0x8A		OCR1BL	Không	So sánh ngõ ra Timer1 B byte thấp
	0x8B		OCR1BH	Không	So sánh ngõ ra Timer1 B byte cao
	0X36	0X16	TIFR1	Có	Cờ báo ngắt Timer1
Timer2	0x6F		TIMSK1	Có	Che ngắt Timer1
	0xB0		TCCR2A	Có	Điều khiển Timer2 A
	0xB1		TCCR2B	Có	Điều khiển Timer2 B
	0xB2		TCNT2	Không	Bộ đếm Timer2
	0xB3		OCR2A	Không	So sánh ngõ ra Timer2 A

	0xB4	OCR2B	Không	So sánh ngõ ra Timer2 B
	0X37	0X17	TIFR2	Cò báo ngắt Timer2
	0x70		TIMSK2	Che ngắt Timer2
	0xB6		ASSR	Trạng thái bắt đồng bộ
	0x43	0x23	GTCCR	Điều khiển Timer chung

Các Timer có 4 đến 5 mô thức làm việc chính: bình thường(Normal),xóa Timer theo kết quả so sánh (CTC:Clear Timer on Compare Match),điều chế độ rộng xung nhanh(FPWM:Fast PWM),điều chế độ rộng xung hiệu chỉnh pha(PCPWM:Phase Correct PWM),điều chế độ rộng xung hiệu chỉnh pha và tần số(PFCPWM:Phase and Frequency Correct PWM).Xung CK cho Timer làm việc có thể lập trình nhiều tần số khác nhau,hoặc lấy từ bên ngoài (chỉ có Timer0 và Timer1).Đặc biệt các mô thức đều có thể tạo tín hiệu ra chân port lập trình được,phục vụ mục đích tạo sóng,tạo tín hiệu chỉ báo,kích khởi ngõ ra...

Trong phần sau ta sẽ tìm hiểu sơ đồ khối và các mô thức hoạt động của các Timer,lập trình hợp ngữ và C cho các mô thức hoạt động của các Timer.Phần sau cùng sẽ trình bày ứng dụng Timer làm bộ chuyển đổi số sang tương tự DAC và tạo sóng tam giác.

## 7.2 Sơ đồ khối các Timer

### 7.2.1 Mô tả tóm tắt sơ đồ khối các Timer

Hình 7.2,7.3,7.4 lần lượt là các sơ đồ khối của Timer0,Timer1,Timer2.

Ta ký hiệu n=0,1,2 thay cho các chỉ số của các Timern,các thanh ghi Timer0 và Timer2 8 bit,các thanh ghi Timer1 16 bit.Các ký hiệu trên sơ đồ khối:

- BOTTOM: Timer đạt BOTTOM khi giá trị bộ đếm=0x00
- MAX: Timer đạt MAX khi giá trị bộ đếm=0xFF(8 bit) hay 0xFFFF(16 bit)
- TOP: Timer đạt TOP khi bộ đếm có giá trị cao nhất trong mô thức hoạt động.Giá trị cao nhất có thể bằng MAX hay được đặt trước phụ thuộc vào mô thức(xem chi tiết phân sau)

Thanh ghi TCNTn là bộ đếm của Timern.Khi Timern tràn,nghĩa là thanh ghi TCNTn=0xFF→0x00 hay 0xFFFF→0x0000,cò TOVn được đặt lên 1 chỉ báo Timern tràn.

Thanh ghi OCRnA,OCRnB chứa giá trị so sánh ngõ ra của kênh A và B hoạt động độc lập nhau.Cò OCFnA và cò OCFnB lần lượt đặt lên 1 khi bộ đếm tới giá trị đặt trong OCRnA và OCRnB.

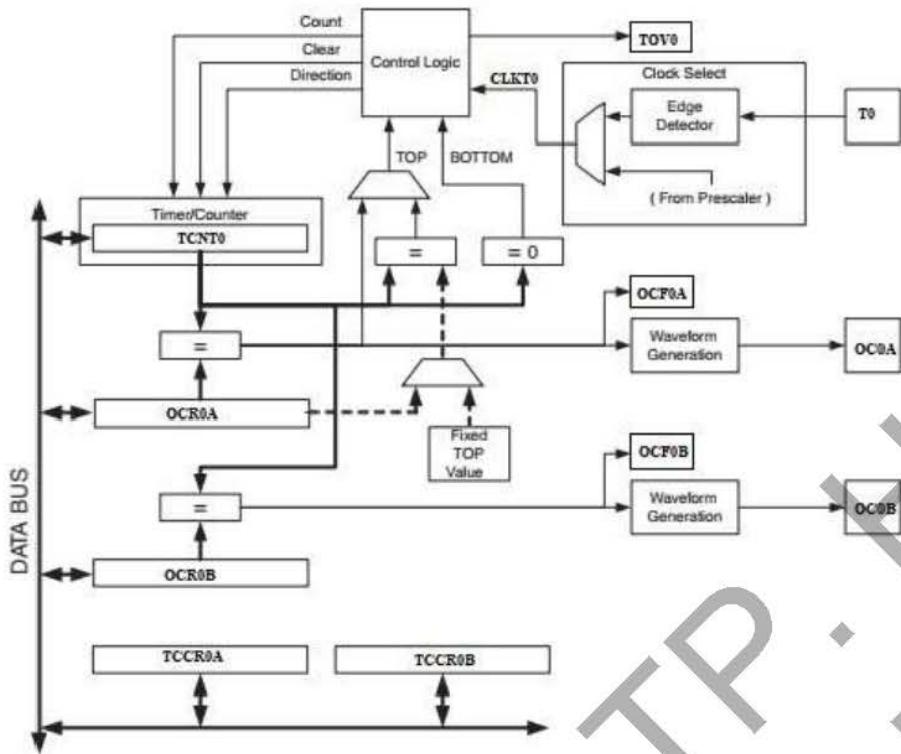
Nguồn tạo xung CK CLKTn có thể được chọn từ nguồn xung dao động nội CLKIO qua bộ chia đặt trước(Prescaler),hoặc từ ngõ I/O ngoài Tn.Chức năng đếm từ xung ngoài chỉ có trong Timer0 và Timer1,riêng Timer2 thay thế chức năng đếm xung ngoài bằng chức năng đếm thời gian thực,bằng cách gắn một bộ thạch anh Xtal f=32768Hz ở 2 ngõ TOSC1 và TOSC2.

Các ngõ ra OCnA,OCnB tạo chuyển biến logic hoặc chuỗi xung ngõ ra trong các mô thức có khai báo tạo sóng ngõ ra.

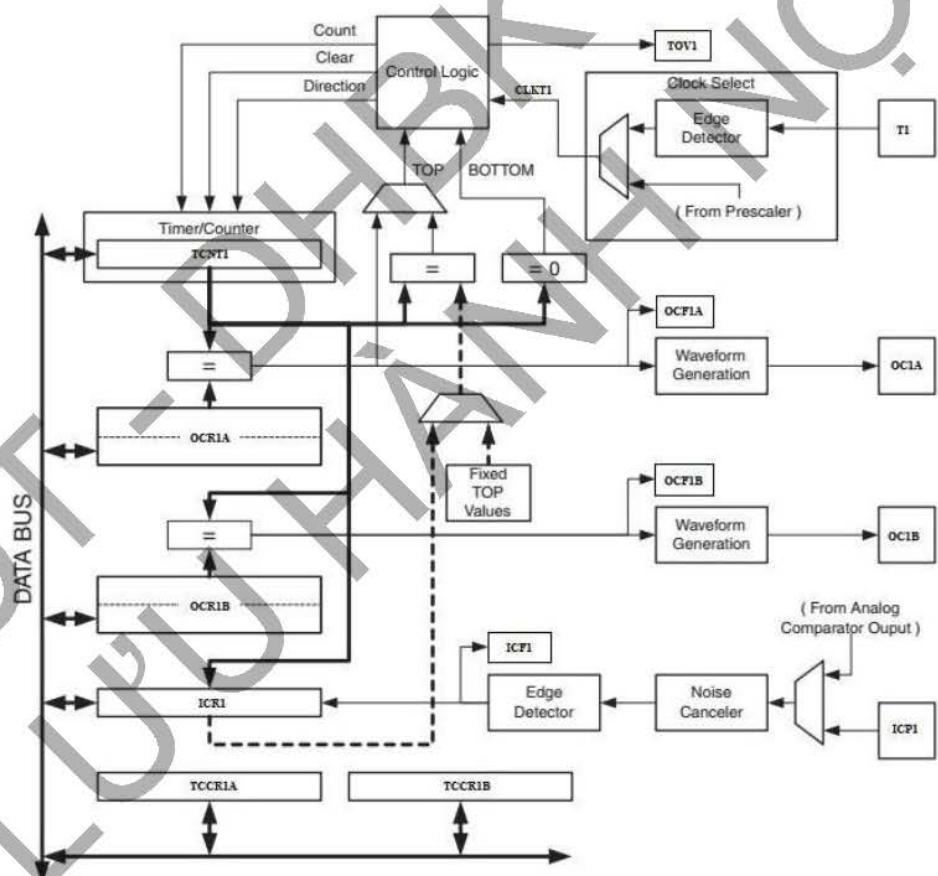
Timer1 có thêm thanh ghi ICR1 bắt giá trị bộ đếm ngõ vào được kích khởi bởi ngõ ICP1 hoặc ngõ ra bộ so sánh tương tự(Analog comparator),hoặc ICR1 làm việc như mô thức xóa Timer theo kết quả so sánh.Cò ICF1 đặt lên 1 khi ICR1 bắt được giá trị bộ đếm hoặc khi bộ đếm đạt tới giá trị đặt trong ICR1.

### ❖ Câu hỏi ôn tập

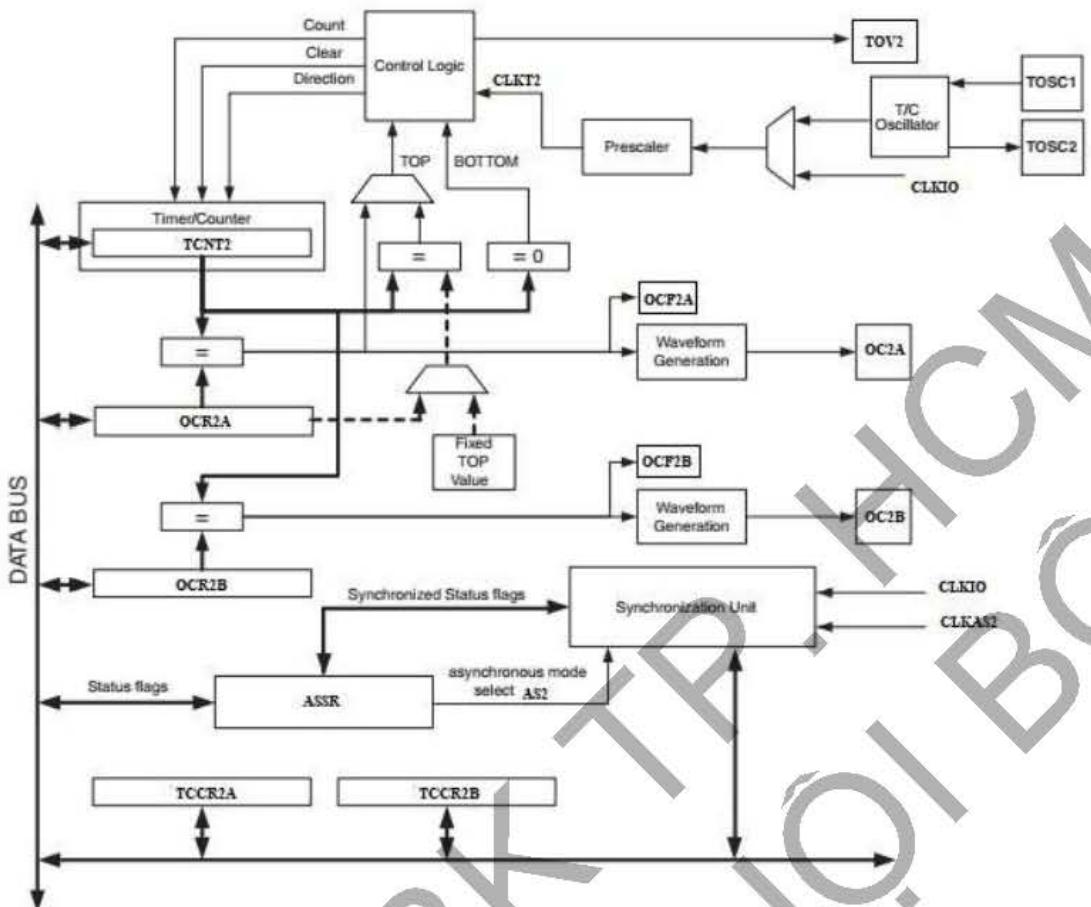
1. Cho biết tên bộ đếm và thanh ghi cò báo ngắt Timer0.
2. Khi nào Timer2 tràn,cò nào chỉ báo tràn Timer2.
3. Cho biết chức năng thanh ghi OCR0A/B.Khi nào cò OCF0A/B=1?
4. Cho biết chức năng thanh ghi ICR1.Khi nào cò ICF1=1?
5. Timer nào có thể nhận xung đếm từ nguồn bên ngoài?



Hình 7.2: Sơ đồ khái Timer0



Hình 7.3: Sơ đồ khái Timer1



Hình 7.4: Sơ đồ khái niệm Timer2

### 7.2.2 Nguồn tạo xung CK Timer

Hình 7.5 là sơ đồ khái niệm nguồn tạo xung CK cho các Timer. Trong hình 7.5 cũng minh họa sơ đồ khái niệm hoạt động của mô thức bình thường và xóa Timer theo kết quả so sánh.

Với Timer0 và Timer1, việc chọn nguồn tạo xung CLKTn do 3 bit CSn2, CSn1, CSn0 thuộc thanh ghi TCCRnB quyết định. Trường hợp chọn định kỳ (Timer), xung CLKIO tạo từ mạch dao động nội qua bộ chia đặt trước (Prescaler) cho các ngõ ra CLK/N, với  $N=1, 8, 64, 256, 1024$ . Tổ hợp CSn2:CSn0=000 cấm xung CK vào bộ đếm làm Timer dừng, các tổ hợp từ 001 đến 101 chọn ngõ ra từ bộ chia đặt trước (Prescaler) theo hệ số chia N như trên. Tổ hợp CSn2:CSn0=110/111 chọn đếm sự kiện (Counter), xung CKTn từ ngõ I/O Tn đưa vào, chọn CSn2:CSn0=110 kích cảnh xuống, 111 kích cảnh lên. Lưu ý là Timer0 và Timer1 sử dụng cùng bộ chia đặt trước nên có hệ số chia N giống nhau.

Với Timer2 bộ chia đặt trước có 7 hệ số chia  $N=1, 8, 32, 64, 128, 256, 1024$ . CS22:CS20 chỉ có chức năng dừng hoặc chọn hệ số chia N cho xung CLKT2. Việc điều khiển chọn xung ngõ vào do bit AS2 thuộc thanh ghi ASSR quyết định. Khi AS2=0, Timer2 chọn xung ngõ vào là CLKIO từ mạch dao động nội. Khi AS2=1, xung ngõ vào từ mạch dao động tạo bởi Xtal gắn vào 2 ngõ TOSC1 và TOSC2.

Việc chọn nguồn tạo xung CK Timer sẽ được trình bày chi tiết ở phần sau.

Để dễ hiểu và áp dụng Timer, ta khảo sát trước hai mô thức cơ bản của Timer là mô thức bình thường và mô thức kết quả so sánh ngõ ra. Các mô thức còn lại sẽ trình bày trong phần tạo sóng hoặc trong phần hoạt động từng Timer cụ thể.

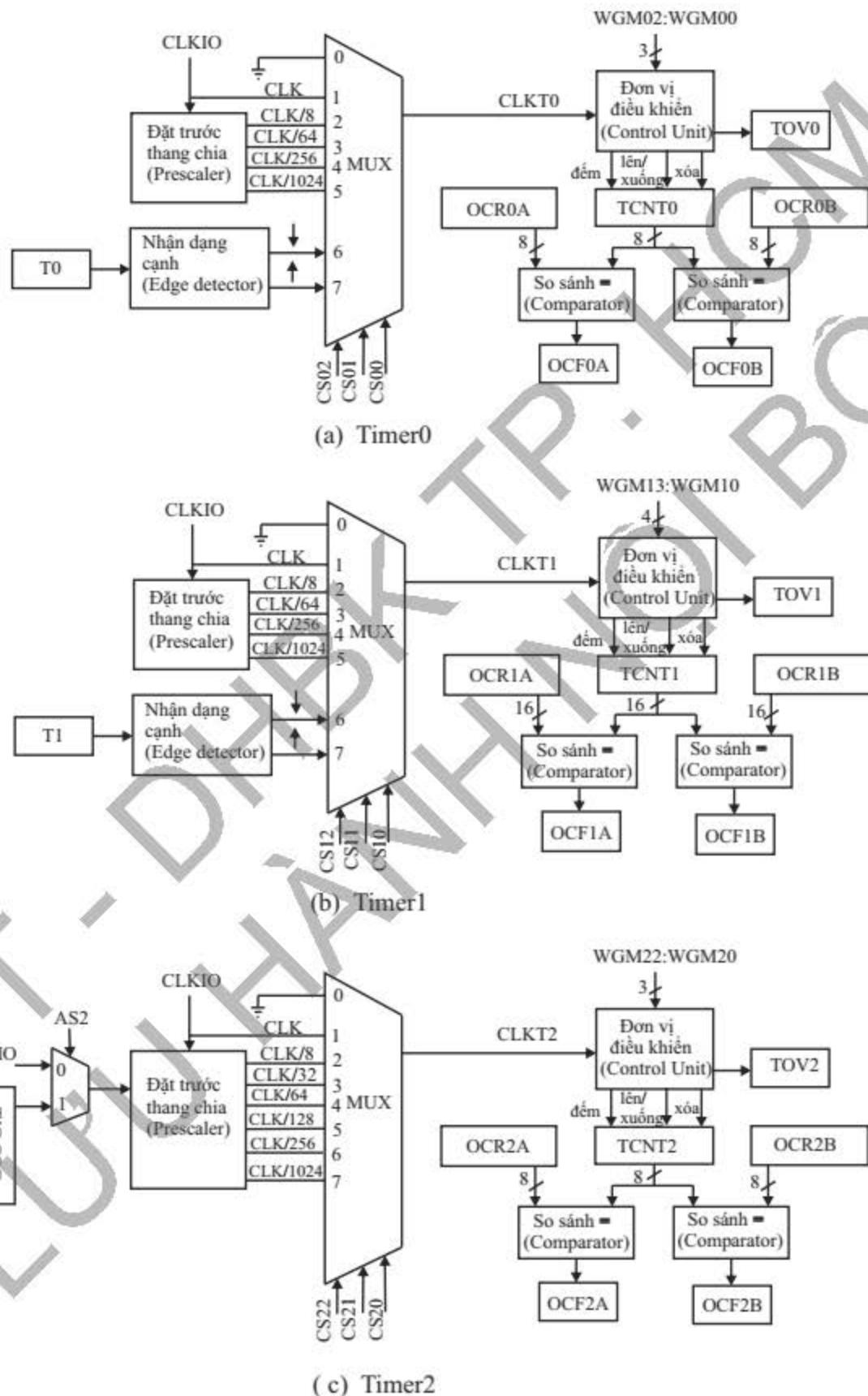
### 7.2.3 Mô thức bình thường(NOR:Normal)

Xem lại sơ đồ khái niệm 7.5.

Trong mô thức NOR, bộ đếm TCNTn đếm từ giá trị đặt trước bằng phần mềm. Mỗi lần có xung CLKTn, bộ đếm TCNTn đếm lên tăng 1 đơn vị cho đến giá trị MAX (MAX=0xFF với Timer0, Timer2 và MAX=0xFFFF với Timer1). Khi có thêm 1 xung đếm nữa, TCNTn bị tràn trả về 0x00 (0x0000), lúc này phần cứng sẽ đặt cờ báo tràn TOVn=1 chỉ báo Timer n đã bị tràn. Bộ đếm tiếp tục đếm lên từ 0 nếu ta không nạp lại giá trị đặt trước.

Hoạt động mô thức NOR sẽ được trình bày chi tiết trong phần khảo sát hoạt động các Timer.

- Lưu ý: Phần cứng chỉ đặt cờ TOV<sub>n</sub>=1 khi Timer n tràn mà không xóa TOV<sub>n</sub>, trừ trường hợp có khai báo ngắt Timer n mô thức NOR! Do đó ta phải xóa cờ TOV<sub>n</sub> bằng phần mềm để có thể nhận dạng Timer n tràn lần kế tiếp.



Hình 7.5: Sơ đồ khởi nguồn tạo xung CK các Timer và mô thức bình thường, kết quả so sánh ngõ ra.

#### 7.2.4: Mô thức xóa Timer theo kết quả so sánh ngõ ra(CTC: Clear Timer on Compare Match)

Xem lại hình 7.5.

Mỗi Timer có 2 thanh ghi OCRnA và OCRnB đặt giá trị so sánh làm việc độc lập tương ứng 2 kênh A và B.Khi bộ đếm TCNTn đếm lên đến giá trị bằng giá trị đặt trong thanh ghi OCRnA hoặc OCRnB,xung đếm kế tiếp làm phần cứng sẽ đặt cờ OCFnA hoặc OCFnB lên 1 chỉ báo giá trị đếm bằng giá trị tham chiếu đặt trước.

Với Timer0,Timer2 quy định thanh ghi OCRnA chứa giá trị TOP,do đó khi bộ đếm TCNTn đếm đến giá trị bằng OCRnA sẽ đạt giá trị TOP,xung đếm kế tiếp sẽ xóa bộ đếm TCNTn về 0 và đặt cờ OCFnA=1.Cho nên nếu muốn kênh B đáp ứng kết quả so sánh,ta phải đặt giá trị trong OCRnB không lớn hơn giá trị đặt trong OCRnA!

Với Timer1,quy định giá trị TOP chính là giá trị đặt trong thanh ghi OCR1A hoặc ICR1.

Việc chọn các mô thức quyết định bởi các bit WGMn3:WGMn0 thuộc 2 thanh ghi TCCRnA, TCCRnB.

Hoạt động mô thức CTC sẽ được trình bày chi tiết trong phần khảo sát hoạt động các Timer.

➤ **Lưu ý:** - *Phần cứng chỉ đặt cờ OCFn=1 khi Timern đạt kết quả so sánh mà không xóa OCFn, trừ trường hợp có khai báo ngắt Timern mô thức CTC!Do đó ta phải xóa cờ OCFn bằng phần mềm để có thể nhận dạng kết quả so sánh lần kế tiếp.*

- *Trong mô thức Bình thường(NOR) cờ OCFnA/B vẫn được đặt lên 1 bằng phần cứng khi bộ đếm TCNTn đếm đến giá trị bằng giá trị đặt trong thanh ghi OCRnA/B!*

#### ❖ Câu hỏi ôn tập

- Khi muốn dừng Timer phải làm như thế nào?
- Cho Fosc=8Mhz,với bộ chia đặt trước Timer0/1 cho các ngõ ra có thời gian 1 xung đếm(CLKT0/1) bằng bao nhiêu?
- Thanh ghi nào chọn mô thức hoạt động cho Timer
- Cờ TOV1=1 khi nào?Cờ OCF1A=1 khi nào?
- Trong mô thức CTC của Timer0 giá trị so sánh đặt ở thanh ghi nào,gía trị TOP đặt ở đâu?

### 7.3 Lập trình hoạt động Timer0

#### 7.3.1 Các thanh ghi Timer0

##### 1. Thanh ghi TCNT0(Timer/Counter Register): Thanh ghi định thời

Bit	7	6	5	4	3	2	1	0	
0x26 (0x46)	TCNT0[7:0]								TCNT0
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

Hình 7.6: Thanh ghi TCNT0

Thanh ghi TCNT0 là bộ đếm 8 bit,địa chỉ I/O=0x26,có thể đọc ghi,không truy xuất bit được.

Mỗi lần có xung đếm CLKT0,nội dung thanh ghi TCNT0 tăng 1.Giá trị MAX của TCNT0 bằng 0xFF.Xung đếm tiếp theo làm TCNT0 tràn và trở về 0x00.

##### 2. Thanh ghi TCCR0A(Timer/Counter Control Register A): Thanh ghi Điều khiển Timer0

Bit	7	6	5	4	3	2	1	0	
0x24 (0x44)	COM0A1	COM0A0	COM0B1	COM0B0	-	-	WGM01	WGM00	TCCR0A
Read/Write	R/W	R/W	R/W	R/W	R	R	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

Hình 7.7: Thanh ghi TCCR0A

Thanh ghi TCCR0A địa chỉ I/O=0x24,có thể đọc ghi và truy xuất bit được,cài đặt logic các ngõ ra OC0A,OC0B trong các mô thức tạo sóng và kết hợp với thanh ghi TCCR0B cài đặt mô thức hoạt động cho Timer0.

- **Bit 7:6 COM0A1:COM0A0(Compare Match Output A Mode): Bit kết quả ngõ ra A**

Hai bit này cài đặt trạng thái logic ngõ ra OC0A trong các mô thức tạo sóng(sẽ trình bày chi tiết trong phần tạo sóng)

- **Bit 5:4 COM0B1:COM0B0 (Compare Match Output B Mode): Bit kết quả ngõ ra B**

Hai bit này cài đặt trạng thái logic ngõ ra OC0B trong các mô thức tạo sóng(sẽ trình bày chi tiết trong phần tạo sóng)

- Bit 2:1 WGM01:WGM00 (Waveform Generation Mode): Bit mô thức tạo sóng**

Hai bit này kết hợp với bit WGM02 thuộc thanh ghi TCCR0B cài đặt các mô thức hoạt động của Timer0,sẽ trình bày trong mô tả thanh ghi TCCR0B.

### 3. Thanh ghi TCCR0B(Timer/Counter Control Register B): Thanh ghi B điều khiển Timer0

Bit	7	6	5	4	3	2	1	0	TCCR0B
0x25 (0x45)	FOC0A	FOC0B	-	-	WGM02	CS02	CS01	CS00	
Read/Write	W	W	R	R	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

**Hình 7.8:** Thanh ghi TCCR0B

Thanh ghi TCCR0B có địa chỉ I/O=0x25,truy xuất bit được,đọc ghi 4 bit thấp và chỉ ghi hai bit cao 7 và 6,cài đặt tần số xung CLKT0 cho bộ đếm,kết hợp với thanh ghi TCCR0A cài đặt các mô thức hoạt động của Timer0,điều khiển trực tiếp các ngõ ra OC0A,OC0B .

- Bit 7 FOC0A(Force Output Compare A): Bit ép so sánh ngõ ra A**

Bit này chỉ tác động khi Timer0 làm việc ở mô thức NOR,CTC.Khi ghi FOC0A=1,sẽ ép kết quả so sánh xảy ra(tương đương như giá trị bộ đếm bằng OCR0A)tác động lên khôi tạo sóng,ngõ ra OC0A sẽ thay đổi logic theo cài đặt COM0A1:COM0A0.

- Bit 6 FOC0B(Force Output Compare B): Bit ép so sánh ngõ ra B**

Bit này chỉ tác động khi Timer0 làm việc ở mô thức NOR,CTC.Khi ghi FOC0B=1,sẽ ép kết quả so sánh xảy ra(tương đương như giá trị bộ đếm bằng OCR0B)tác động lên khôi tạo sóng,ngõ ra OC0B sẽ thay đổi logic theo cài đặt COM0B1:COM0B0.

➤ **Lưu ý:** - Trong các mô thức PWM,khi ghi vào TCCR0B nên xóa FOC0A,FOC0B để tương hợp với các chip trong tương lai.

- Khi đọc các bit FOC0A,FOC0B sẽ cho giá trị 0.

- Bit 3 WGM02(Waveform Generation Mode): Bit mô thức tạo sóng**

Bit WGM02 kết hợp với bit WGM01,WGM00 thuộc TCCR0A cài đặt các mô thức hoạt động của Timer0 như Bảng 7.2.

**Bảng 7.2:** Các mô thức hoạt động của Timer0

Mode	WGM02	WGM01	WGM00	Mô thức hoạt động	TOP	OCR0x cập nhật	TOV0 đặt=1
0	0	0	0	NOR	0xFF	Tức thời	MAX
1	0	0	1	PCPWM	0xFF	TOP	BOTTOM
2	0	1	0	CTC	OCR0A	Tức thời	MAX
3	0	1	1	FPWM	0xFF	BOTTOM	MAX
4	1	0	0	Dự trữ	-	-	-
5	1	0	1	PCPWM	OCR0A	TOP	BOTTOM
6	1	1	0	Dự trữ	-	-	-
7	1	1	1	FPWM	OCR0A	BOTTOM	TOP

- NOR(Normal):bình thường

- CTC(Clear Timer on Compare Match):xóa Timer theo kết quả ngõ ra

- FPWM(Fast Pulse Width Modulation):điều chế độ rộng xung nhanh

- PCPWM(Phase Correct Pulse Width Modulation):điều chế độ rộng xung hiệu chỉnh pha

- Cột TOP thể hiện giá trị định bộ đếm TCNT0 đếm tới

- Cột OCR0x cập nhật thể hiện thời điểm tại đó các thanh ghi OCR0A/OCR0B cập nhật giá trị mới

- Cột TOV0 đặt=1 thể hiện thời điểm tại đó cờ TOV0 được đặt lên 1.

Ví dụ trong mô thức CTC WGM02:WGM00=010, TOP=giá trị đặt trong OCR0A,OCR0A/OCR0B có thể cập nhật giá trị mới tức thời,cờ TOV0 =1 khi Timer0 đếm tới MAX=0xFF.Hay nói cách khác,trường hợp này phải đặt OCR0A=0xFF mới có thể có cờ TOV0=1!

- Bit 2:1:0 CS02:CS01:CS00(Clock Select): Bit chọn xung Clock**

Ba bit này cài đặt nguồn tạo xung CK cho bộ đếm Timer0 như mô tả trong Bảng 7.3

**Bảng 7.3:** Lựa chọn nguồn tạo xung CK Timer0

CS02	CS01	CS00	Tần số xung CLKTO
0	0	0	Không có xung CK(Timer0 dừng)
0	0	1	CLKIO(N=1:không qua bộ chia đặt trước)
0	1	0	CLKIO/8(N=8:từ bộ chia đặt trước)
0	1	1	CLKIO/64(N=64:từ bộ chia đặt trước)
1	0	0	CLKIO/256(N=256:từ bộ chia đặt trước)
1	0	1	CLKIO/1024(N=1024:từ bộ chia đặt trước)
1	1	0	Nguồn CK ngoài từ chân T0,tác động cạnh xuống
1	1	1	Nguồn CK ngoài từ chân T0,tác động cạnh lên

- Để Timer0 dừng đếm ta đặt CS02:CS00=000

- Để Timer0 làm việc định thì,xung CK Timer lấy từ nguồn xung dao động nội,ta đặt CS2:CS0=001 đến 101 tương ứng với hệ số chia N=1,8,64,256,1024 từ xung CLKIO.

Với MCU324P cho dao động nội Fosc=8Mhz,không lập trình chia 8(cầu chì CLKDIV=1) FCLKIO=Fosc=8Mhz.Nếu CS02:CS00=001,N=1,một xung CLKTO dài 1/8Mhz=0.125μs.Nếu đặt CS02:CS00=101,N=1024,một xung CLKTO=0.125x1024=128μs.

- Nếu cho Timer0 làm việc như bộ đếm sự kiện,cài đặt CS02:CS00=110 hoặc 111,xung đếm ngoài đưa vào chân T0=PB0.Timer0 đếm lên 1 đơn vị khi có cạnh xuống nếu CS02:CS00=110,hoặc khi có cạnh lên nếu CS02:CS00=111.

➤ *Lưu ý: Khi khai báo CS02:CS00=110,111,chân T0=PB0 tự động chuyển thành ngõ vào nhận xung đếm từ bên ngoài,cho dù trước đó ta không khai báo PB0 là ngõ vào!*

**Ví dụ 7.1:** Phải đặt giá trị các thanh ghi TCCR0A,TCCR0B bằng bao nhiêu khi muốn khai báo Timer0 hoạt động ở mô thức:

- (a) Bình thường(NOR),không chia tần số
- (b) CTC,hệ số chia tần số N=256
- (c) CTC,đếm xung ngoài tác động cạnh xuống

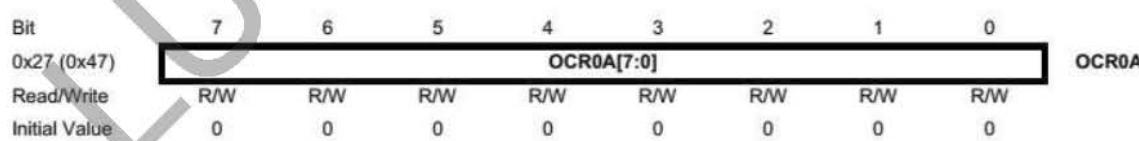
**Giải:**

(a) Timer0 làm việc mô thức NOR WGM02:WGM00=000,không chia tần số CK N=1 CS02:CS00=001.Giá trị nạp cho các thanh ghi TCCR0A,TCCR0B như sau:  
TCCR0A=0b00000000=0x00  
TCCR0B=0b00000001=0x01

(b) Timer0 làm việc mô thức CTC WGM02:WGM00=010,chia tần số CK N=256 CS02:CS00=100.Giá trị nạp cho các thanh ghi TCCR0A,TCCR0B như sau:  
TCCR0A=0b00000010=0x02  
TCCR0B=0b00000100=0x04

(c) Timer0 làm việc mô thức CTC,xung CK ngoài tác động cạnh xuống WGM02:WGM00=010, CS02:CS00=110.Giá trị nạp cho các thanh ghi TCCR0A,TCCR0B như sau:  
TCCR0A=0b00000010=0x02  
TCCR0B=0b00000110=0x06

#### 4. Thanh ghi OCR0A(Output Compare Register A): Thanh ghi so sánh ngõ ra A



**Hình 7.9:** Thanh ghi OCR0A

Thanh ghi OCR0A có địa chỉ I/O=0x27,có thể đọc ghi,không truy xuất bit được.Thanh ghi OCR0A chứa giá trị tham chiếu để bộ đếm TCNT0 so sánh trong mô thức CTC và các mô thức PWM kênh A.

➤ *Lưu ý: Trong mô thức CTC và một số mô thức PWM,gía trị đặt trong OCR0A chính là giá trị TOP của bộ đếm Timer0.*

#### 5. Thanh ghi OCR0B(Output Compare Register B): Thanh ghi so sánh ngõ ra B

Bit	7	6	5	4	3	2	1	0	
0x28 (0x48)	OCR0B[7:0]								OCR0B
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

Hình 7.10: Thanh ghi OCR0B

Thanh ghi OCR0B có địa chỉ I/O=0x28,có thể đọc ghi,không truy xuất bit được.Thanh ghi OCR0B chứa giá trị tham chiếu để bộ đếm TCNT0 so sánh trong mô thức CTC và các mô thức PWM kênh B.

➤ **Lưu ý:** Trong các mô thức CTC và một số mô thức PWM,do TOP=gia trị đặt trong OCR0A,nên giá trị đặt trong OCR0B không lớn hơn OCR0A ,việc so sánh ở kênh B mới thực hiện được!

## 6. Thanh ghi TIFR0(Timer/Counter 0 Interrupt Flag Register): Thanh ghi cờ ngắt Timer0

Bit	7	6	5	4	3	2	1	0	
0x15 (0x35)	-	-	-	-	-	OCF0B	OCF0A	TOV0	TIFR0
Read/Write	R	R	R	R	R	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

Hình 7.11: Thanh ghi TIFR0

Thanh ghi TIFR0 có địa chỉ I/O=0x15,đọc ghi và truy xuất bit được.Thanh ghi TIFR0 chứa 3 cờ báo cho các mô thức tương ứng.

- **Bit 2 OCF0B(Timer/Counter 0 Output Compare B Match Flag): Cờ báo kết quả so sánh ngõ ra B Timer0**

Bit cờ OCF0B được đặt lên 1 bằng phần cứng khi bộ đếm TCNT0 có giá trị bằng giá trị đặt trong thanh ghi OCR0B.Cờ OCF0B được xóa bằng phần cứng khi MCU chuyển điều khiển đến vector ngắt nếu có khai báo cho phép ngắt từ nguồn ngắt tác động bởi cờ này(xem chi tiết chương 10).Hoặc ta phải xóa cờ OCF0B bằng cách ghi 1 vào bit này.

- **Bit 1 OCF0A(Timer/Counter 0 Output Compare A Match Flag): Cờ báo kết quả so sánh ngõ ra A Timer0**

Bit cờ OCF0A được đặt lên 1 bằng phần cứng khi bộ đếm TCNT0 có giá trị bằng giá trị đặt trong thanh ghi OCR0A.Cờ OCF0A được xóa bằng phần cứng khi MCU chuyển điều khiển đến vector ngắt nếu có khai báo cho phép ngắt từ nguồn ngắt tác động bởi cờ này(xem chi tiết chương 10).Hoặc ta phải xóa cờ OCF0A bằng cách ghi 1 vào bit này.

- **Bit 0 TOV0(Timer/Counter 0 Over Flow Flag): Cờ báo tràn Timer0**

Bit cờ TOV0 được đặt lên 1 bằng phần cứng khi bộ đếm TCNT0 tràn từ 0xFF sang 0x00 trong mô thức NOR và CTC,ngoài ra còn tùy thuộc vào các mô thức khác như mô tả trong Bảng 7.2.Cờ TOV0 được xóa bằng phần cứng khi MCU chuyển điều khiển đến vector ngắt nếu có khai báo cho phép ngắt từ nguồn ngắt tác động bởi cờ này(xem chi tiết chương 10).Hoặc ta phải xóa cờ TOV0 bằng cách ghi 1 vào bit này.

➤ **Lưu ý:** Trong họ AVR,ngoại trừ các ghi chú đặc biệt,các bit được định nghĩa là cờ báo được phần cứng đặt lên 1 khi thỏa điều kiện nào đó,và phần cứng tự động xóa cờ ngay sau khi thực hiện 1 tác vụ nào đó.Ta không thể ghi 0 bằng phần mềm vào bit cờ để xóa cờ như các MCU khác.Nếu muốn xóa cờ bằng phần mềm,ta phải ghi 1 vào bit cờ tương ứng!

## 7. Thanh ghi TIMSK0(Timer/Counter 0 Interrupt Mask Register):Thanh ghi che ngắt Timer0

Bit	7	6	5	4	3	2	1	0	
(0x6E)	-	-	-	-	-	OCIE0B	OCIE0A	TOIE0	TIMSK0
Read/Write	R	R	R	R	R	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

Hình 7.12: Thanh ghi TIMSK0

Thanh ghi TIMSK0 có địa chỉ MEM=0x6E(I/O mở rộng),đọc ghi và truy xuất bit được.Thanh ghi TIMSK0 chứa 3 bit cấm/cho phép các nguồn ngắt tràn Timer0 hoặc kết quả so sánh ngõ ra.

- Bit 2 OCIE0B(Timer/Counter 0 Output Compare Match B Interrupt Enabe): Bit cho phép ngắt kết quả so sánh ngõ ra B Timer0**  
Khi đặt bit OCIE0B=1, đồng thời bit I thuộc thanh ghi SREG cũng đặt lên 1, ngắt kết quả so sánh ngõ ra B Timer0(COMP0B Interrupt) được cho phép. Khi bộ đếm TCNT0 đếm đến giá trị bằng giá trị đặt trong thanh ghi OCR0B, cờ OCF0B được phần cứng đặt lên 1, chính là tín hiệu báo ngắt COMP0B. MCU chuyển điều khiển đến vector ngắt COMP0B và phần cứng xóa cờ OCF0B.
- Bit 1 OCIE0A(Timer/Counter 0 Output Compare Match A Interrupt Enabe): Bit cho phép ngắt kết quả so sánh ngõ ra A Timer0**  
Khi đặt bit OCIE0A=1, đồng thời bit I thuộc thanh ghi SREG cũng đặt lên 1, ngắt kết quả so sánh ngõ ra A Timer0(COMP0A Interrupt) được cho phép. Khi bộ đếm TCNT0 đếm đến giá trị bằng giá trị đặt trong thanh ghi OCR0A, cờ OCF0A được phần cứng đặt lên 1, chính là tín hiệu báo ngắt COMP0A. MCU chuyển điều khiển đến vector ngắt COMP0A và phần cứng xóa cờ OCF0A.
- Bit 0 TOIE0(Timer/Counter 0 Over Flow Interrupt Enabe): Bit cho phép ngắt tràn Timer0**  
Khi đặt bit TOIE0=1, đồng thời bit I thuộc thanh ghi SREG cũng đặt lên 1, ngắt Timer0 tràn (TOV0 Interrupt) được cho phép. Khi bộ đếm TCNT0 tràn từ 0xFF sang 0x00 trong các mô thức NOR, CTC, hoặc tùy theo các mô thức khác như mô tả trong Bảng 7.2, cờ TOV0 được phần cứng đặt lên 1, chính là tín hiệu báo ngắt TOV0. MCU chuyển điều khiển đến vector ngắt TOV0 và phần cứng xóa cờ TOV0.

Các trường hợp ngắt Timer0 sẽ được trình bày chi tiết ở chương 10.

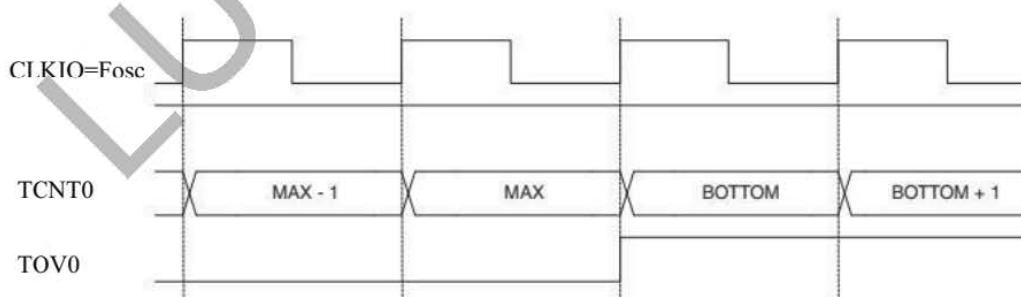
#### ❖ Câu hỏi ôn tập

- Các cờ TOV0, OCF0A, OCF0B thuộc thanh ghi nào?
- Viết đoạn lệnh lập vòng kiểm tra cờ TOV0, xóa cờ TOV0 và thoát nếu TOV0=1.
- Viết đoạn lệnh lập vòng kiểm tra cờ OCF0A, xóa cờ OCF0A và thoát nếu OCF0A=1.
- Giá trị nạp TCCR0A và TCCR0B bằng bao nhiêu nếu muốn cho Timer0 chạy mô thức CTC, xung CLKTO=32μs?
- Cho TCCR0A=0x02, TCCR0B=0x02, OCR0A=145. Timer0 bắt đầu khởi động đếm mất bao lâu cờ OCF0A=1? Trong trường hợp nào cờ OCF0B=1. Giải thích tại sao?
  - OCR0B=136
  - OCR0B=\$95

#### 7.3.2 Hoạt động Timer0 mô thức bình thường(NOR: Normal)

Mô thức NOR thường được sử dụng trong các ứng dụng định giờ, tạo thời gian trễ(delay time) theo thời gian chuẩn hàng số từ bộ dao động chủ, hoặc đếm các sự kiện bên ngoài.

Trong mô thức NOR, cài đặt WGM02:WGM00=000, cài đặt CS02:CS00 chọn tần số xung CLKTO. Mỗi lần có 1 xung CLKTO, Timer0 đếm lên 1 đơn vị cho đến giá trị MAX=0xFF. Xung đếm kế tiếp làm Timer0 tràn và bộ đếm trở về BOTTOM=0x00, đồng thời cờ TOV0 đặt lên 1 bằng phần cứng. Hình 7-13 minh họa giản đồ xung mô thức NOR với CKIO=Fosc.



Hình 7.13: Giản đồ xung Timer0 mô thức NOR, CLKIO=Fosc

#### ❖ Các bước lập trình Timer0 trong mô thức NOR

1. Nạp giá trị đặt trước vào thanh ghi TCNT0
2. Cài đặt mô thức NOR bằng cách đặt WGM02:WGM00=000,cài đặt tần số xung CLKT0 bằng cách chọn tổ hợp CS02:CS00 để có hệ số chia N.Thực hiện bằng cách nạp các giá trị thích hợp vào 2 thanh ghi TCCR0A,TCCR0B.Khi ta cài đặt CS02:CS00 khác 000 Timer0 bắt đầu đếm lên.
3. Liên tục kiểm tra cờ TOV0 bằng vòng lặp để phát hiện khi nào TOV0=1 là Timer0 tràn, thoát khỏi vòng lặp.
4. Ngừng Timer0 bằng cách nạp CS02:CS00=000.
5. Xóa cờ TOV0 bằng cách ghi 1 vào bit TOV0,để phát hiện lần tràn kế tiếp.
6. Lặp lại bước 1 nếu có yêu cầu.

❖ Trong các ví dụ sau,ta quy ước CLKIO=Fosc=8Mhz trừ trường hợp đặc biệt có ghi chú cụ thể.Hệ số chia N chọn ngõ ra bộ chia đặt trước tạo xung đếm CLKTn=Fosc/N.

**Ví dụ 7.2:** Chương trình hợp ngữ sau đây sử dụng Timer0 tạo chuỗi xung đối xứng trên PB1.Giải thích hoạt động của chương trình và thử tính chu kỳ xung ngõ ra PB1.

```

.EQU P_OUT=1
.ORG 0
RJMP MAIN
.ORG 0X40
MAIN: LDI R16,HIGH(RAMEND) ;đưa stack lên đỉnh SRAM
      OUT SPH,R16
      LDI R16,LOW(RAMEND)
      OUT SPL,R16
      LDI R16,(1<<P_OUT) ;đặt PB1 output
      OUT DDRB,R16
      LDI R17,0X00
      OUT TCCR0A,R17
      LDI R17,0X00
      OUT TCCR0B,R17
START: RCALL DELAY
      IN R17,PORTB
      EOR R17,R16
      OUT PORTB,R17
      RJMP START
;
-----;
DELAY: LDI R17,$A0
      OUT TCNT0,R17
      LDI R17,0X01
      OUT TCCR0B,R17
WAIT:  IN R17,TIFR0
      SBRS R17,TOV0
      RJMP WAIT
      OUT TIFR0,R17
      LDI R17,0X00
      OUT TCCR0B,R17
      RET

```

### Giải:

Phản giải thích từng dòng lệnh chương trình như trên,sau đây ta sẽ giải thích hoạt động của Timer0 mô thức NOR tạo xung.

1. Bước đầu tiên khởi động TCCR0A=0x00,TCCR0B=0x00 cài đặt Timer0 mode NOR,chưa chạy
2. Gọi ctc DELAY cho Timer0 chạy trong khoảng thời gian nhất định,sau đó đảo bit PB1 tạo chuỗi xung ngõ ra đối xứng,độ rộng xung bằng thời gian Timer0 chạy(chính xác là thời gian thực thi ctc DELAY)
3. Trong ctc DELAY,đầu tiên nạp bộ đếm TCNT0 giá trị đặt trước 0xA0

4. Cho Timer0 bắt đầu chạy với CLKTO=CLKIO,hệ số chia N=1,bằng cách nạp TCCR0B=0x01
  5. Đọc trạng thái cờ TOV0 và lập vòng chờ cho đến khi cờ TOV0=1 báo Timer0 tràn
  6. Khi cờ TOV0=1,ghi lại nội dung R17 vào TIFR0,tương ứng bit TOV0=1 để xóa cờ TOV0
  7. Nạp TCCR0B=0x00 dừng Timer0 và thoát khỏi ctc DELAY
- Như vậy thời gian Timer0 chạy tính từ giá trị 0xA0 đến lúc tràn bằng 0x00(cờ TOV0=1),được tính bằng cách lấy số xung đếm được nhân cho chu kỳ CK Timer0(CLKTO=0.125μs):

Số xung đếm được=0x00 - 0xA0=0x60=96

Thời gian Timer0 chạy=96x0.125=12μs

Vậy chu kỳ xung vuông đối xứng ngõ ra PB1:  $T=12 \times 2=24\mu s$

#### ❖ Tính giá trị đặt trước nạp cho bộ đếm

Từ trên ta suy ra công thức tổng quát tính thời gian delay khi Timer0 chạy模式 NOR như sau:

Đặt  $n$  là số xung đếm được từ giá trị đặt trước đến khi bộ đếm tràn,như vậy  $-n$  chính là giá trị đặt trước nạp cho TCNT0,TCLKTO=ToscxN là chu kỳ 1 xung đếm Timer0.

Thời gian delay  $T_d$  được tính:

$$T_d = n \times T_{CLKTO} = n \times T_{osc} \times N \quad (7.1)$$

Từ (7.1) có thể suy ra giá trị đặt trước  $n$ (là số âm) nạp cho TCNT0 để tạo thời gian delay  $T_d$ .

**Ví dụ 7.3:** Tính giá trị đặt trước cho Timer0 khi muốn tạo thời gian delay 1ms.Viết đoạn lệnh khởi động Timer0 chạy với thời gian delay trên.

**Giải:**

Do TCNT0 8 bit nên giá trị khởi động  $n=0x00-0xFF$

$T_{osc}=0.125\mu s$ ,ta tính  $T_{CLKTO}$  theo hệ số chia N và n để có  $T_d=1000\mu s$  cho kết quả như sau:

N	$T_{CLKTO}(\mu s)$	$n=T_d/T_{CLKTO}$
1	0.125	8000
8	1	1000
64	8	125
256	32	31.25
1024	128	7.8125

Vậy ta chọn hệ số chia N=64 và n=125 tương ứng số âm Hex 0x83

Đoạn lệnh khởi động Timer0 như sau:

```
...
LDI    R16,0X83      ;nạp bộ đếm giá trị đặt trước
OUT    TCNT0,R16
LDI    R16,0X00      ;Timer0 mode NOR
OUT    TCCR0A,R16
LDI    R16,0X03      ;Timer0 mode NOR,N=64
OUT    TCCR0B,R16
...
```

#### ❖ Tính chính xác thời gian trễ tạo bởi Timer

Trong ví dụ 7.2,việc tính chu kỳ xung ra  $T=24\mu s$  chỉ là gần đúng,vì để ngõ ra PB1 đảo bit phải kể đến thời gian thực thi các lệnh như gọi chương trình con,khởi động các thanh ghi,chờ cờ TOV0=1,xóa cờ TOV0,dừng Timer0,RET,thời gian thực thi lệnh đảo bit...Như vậy chu kỳ xung thực sự sẽ lớn hơn 24μs!

Ví dụ sau sẽ minh họa việc tính chính xác chu kỳ xung do Timer tạo ra.

**Ví dụ 7.4:** Viết một chương trình tạo chuỗi xung vuông đối xứng tần số 500Hz ở ngõ ra PB1.Dựa trên chương trình đã viết,tính chính xác lại chu kỳ xung ra và nêu phương pháp hiệu chỉnh để đạt kết quả chính xác nhất có thể cho ví dụ này và ví dụ 7.2.

**Giải:**

Chuỗi xung vuông đối xứng f=500Hz,T=2ms,nên độ rộng xung  $T/2=1ms$ .Dựa vào ví dụ 7.3,ta lập trình Timer0 chạy mode NOR,hệ số chia N=64,gia trị đặt trước =-125.

Sử dụng lại chương trình ví dụ 7.2,chỉ cần thay đổi ctc DELAY thành DELAY\_1MS và cài đặt các thanh ghi Timer0 theo các thông số trên.

#### ❖ Chương trình hợp ngữ

```
.EQU  P_OUT=1
.ORG 0
```

	RJMP	MAIN	
	.ORG	0X40	
MAIN:	LDI	R16,HIGH(RAMEND)	;đưa stack lên đỉnh SRAM
	OUT	SPH,R16	
	LDI	R16,LOW(RAMEND)	
	OUT	SPL,R16	
	LDI	R16,(1<<P_OUT)	;đặt PB1 output
	OUT	DDRB,R16	
	LDI	R17,0X00	;Timer0 mode NOR
	OUT	TCCR0A,R17	
	LDI	R17,0X00	;Timer0 mode NOR,dừng
	OUT	TCCR0B,R17	
START:	RCALL	DELAY_1MS	
	IN	R17,PORTE	;delay tạo độ rộng xung 1ms
	EOR	R17,R16	1MC
	OUT	PORTE,R17	1MC
	RJMP	START	1MC
			2MC
-----			
DELAY_1MS:	LDI	R17,(-125)	1MC
	OUT	TCNT0,R17	1MC
	LDI	R17,0X03	1MC
	OUT	TCCR0B,R17	1MC
WAIT:	IN	R17,TIFR0	1MC
	SBRS	R17,TOV0	2/1MC
	RJMP	WAIT	2MC
	OUT	TIFR0,R17	1MC
	LDI	R17,0X00	1MC
	OUT	TCCR0B,R17	1MC
	RET		4MC

- Thời gian thực thi các lệnh trong chương trình con kể cả RET là 16MC
- Thời gian thực thi chương trình chính cho đến khi xong lệnh đảo bit(EOR) là 8MC

Như vậy thời gian cộng thêm để tính độ rộng xung là:  $24 \times 0.125 = 3\mu s$

Thời gian này nhỏ hơn thời gian 1 xung đếm bằng  $8\mu s$  nên sai số không đáng kể!

Bây giờ xét lại ví dụ 7.2, độ rộng xung tính chính xác sẽ là:  $T_p = 12 + 3 = 15\mu s$

Chu kỳ xung tính lại:  $T_c = 15 \times 2 = 30\mu s$

Sai số so với giá trị tính ban đầu:  $e = [(30 - 24)/24]100\% = 25\% !!!$

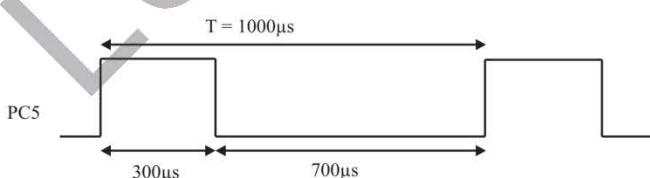
Nếu muốn tạo xung vuông đối xứng  $T=24\mu s$  chính xác như ban đầu và vẫn giữ các lệnh như trên, ta phải giảm bớt giá trị đặt trước cho bộ đếm Timer0.

Trong ví dụ 7.2 cài đặt CLKT0=Fosc nên 1 chu kỳ xung đếm=1MC=0.125μs .

Số xung phải giảm bằng:  $3/0.125 = 24 = \$18$

Do đó giá trị đặt trước  $n = \$A0 + \$18 - \$B8$  tương ứng số xung đếm đến tràn Timer0 là  $96 - 24 = 72$

**Ví dụ 7.5:** Viết một chương trình hợp ngữ/C tạo chuỗi xung vuông tần số 1Khz, chu kỳ nhiệm vụ(CKNV) 30% xuất ra chân PC5.Ban đầu bỏ qua thời gian thực thi các lệnh,sau đó tính chính xác và hiệu chỉnh lại sao cho đạt kết quả tốt nhất.



Hình 7.14: Xung ngõ ra PC5 ví dụ 7.5

**Giải:**

Ta chọn hệ số chia N sao cho tạo độ rộng xung chính xác đến mức có thể như Bảng 7.4

**Bảng 7.4:** Tóm tắt hệ số chia, độ rộng xung và sai số tương ứng ví dụ 7.5

N	T <sub>CLKT0(μs)</sub>	T <sub>p1=300μs</sub> n1=T <sub>p1/T<sub>CLKT0</sub></sub>	Sai số làm tròn e1(μs)	T <sub>p2=700μs</sub> n2=T <sub>p2/T<sub>CLKT0</sub></sub>	Sai số làm tròn e2(μs)
1	0.125	2400	-	5600	-
8	1	300	-	700	-
64	8	37.5	0.5x8=4	87.5	0.5x8=4
256	32	9.375	0.375x32=12	21.875	0.125x32=4
1024	128	2.343	0.343x128=44	5.468	0.468x128=60

Từ bảng 7.4 cho thấy nếu chọn N=64,với độ rộng xung Tp1=300μs giá trị đặt trước -n1=-37,với độ rộng xung Tp2=700μs giá trị đặt trước -n2=-87,sai số làm tròn thấp nhất là 4+4=8μs.

#### ❖ Chương trình hợp ngữ ví dụ 7.5

```

.EQU P_OUT=5 ;gán ký hiệu P_OUT=5
.EQU TP_H=-37 ;giá trị đặt trước mức 1
.EQU TP_L=-87 ;giá trị đặt trước mức 0
.ORG 0
RJMP MAIN
.ORG 0X40
MAIN: LDI R16,HIGH(RAMEND) ;đưa stack lên đỉnh SRAM
      OUT SPH,R16
      LDI R16,LOW(RAMEND)
      OUT SPL,R16
      LDI R16,(1<<P_OUT) ;đặt PC5 output
      OUT DDRC,R16
      LDI R17,0X00 ;Timer0 mode NOR
      OUT TCCR0A,R17
      LDI R17,0X00 ;Timer0 mode NOR, dừng
      OUT TCCR0B,R17
START: SBI PORTC,P_OUT ;output=1 1MC
       LDI R17,TP_H ;nạp TCNT0=TP_H 1MC
       RCALL DELAY_T0 ;ctc chạy Timer0 3MC
       CBI PORTC,P_OUT ;output=0 1MC
       LDI R17,TP_L ;nạp TCNT0=TP_L 1MC
       RCALL DELAY_T0 ; 3MC
       RJMP START ;lặp vòng lại 2MC
;
DELAY_T0: OUT TCNT0,R17 ;
          LDI R17,0X03 ;Timer0 chạy,hệ số chia N=64 1MC
          OUT TCCR0B,R17 ;
WAIT:   IN  R17,TIFR0 ;đọc thanh ghi cờ Timer0 1MC
        SBRS R17,TOV0 ;chờ cờ TOV0=1 báo Timer0 tràn 2/1MC
        RJMP WAIT ;cờ TOV0=0 tiếp tục chờ 2MC
        OUT TIFR0,R17 ;nạp lại bit TOV0=1 xóa cờ TOV0 1MC
        LDI R17,0X00 ;dừng Timer0 1MC
        OUT TCCR0B,R17 ; 1MC
        RET ; 4MC
;
```

- Về cơ bản cấu trúc chương trình tương tự như ví dụ 7.2,chỉ cài đặt lại hệ số chia N và hiệu chỉnh chương trình con DELAY lại một chút.

- Trong chương trình chính phân ra 2 thời đoạn: thời đoạn mức 1 đặt bit P\_OUT=1,nạp TCNT0=TP\_H và gọi ctc chạy Timer0 DELAY\_T0.Thời đoạn mức 0 xóa bit P\_OUT=0,nạp TCNT0=TP\_H và gọi ctc chạy Timer0 DELAY\_T0.Sau đó lặp vòng lại.

Bây giờ ta tính chính xác TpH và TpL,CLKT0=8μs,1MC=0.125μs

$$TpH=37x8+5x0.125(ct\ chính)+15x0.125(ctc)=298.5\mu s \rightarrow e1=298.5-300=-1.5\mu s \#0.5\%$$

$$TpL=87x8+7x0.125(ct \text{ chính})+15x0.125(ctc)=698.75\mu s \rightarrow e1=698.75-700=-1.25\mu s \# 0.18\%$$

Sai số nhỏ hơn 1 xung đếm nên không cần hiệu chỉnh!

Trong trường hợp muốn chính xác ta thêm các lệnh giả chỉ kéo dài thời gian như lệnh NOP hoặc các lệnh lập vòng không làm ảnh hưởng đến kết quả của chương trình bù vào số thời gian thiêu như đã tính. Phần này xem như bài tập dành cho người đọc!

### ❖ Chương trình C ví dụ 7.5

```
#include <avr/io.h>
#define outport PORTC      // định nghĩa PORTC=output
const char p_out=5        ;// ký hiệu giá trị p_out
const char Tp_H=-37,Tp_L=-87;//giá trị đặt trước mức 1 và mức 0
void delay_T0()           ;// khai báo hàm delay_T0
int main()                 // bắt đầu chương trình
{
    DDRC |=(1<<p_out)    ;/khai báo p_out output
    TCCR0A=0x00            ;//Timer0 mode NOR
    TCCR0B=0x00            ;//Timer0 mode NOR,dừng Timer0
    while(1)                // lập vòng vô hạn
    {
        outport |=(1<<p_out)  ;//p_out=1
        TCNT0=Tp_H           ;//nạp TCNT0=Tp_H
        delay_T0()             ;//chạy Timer0
        outport&=~(1<<p_out)   ;//p_out=0
        TCNT0=Tp_L             ;//nạp TCNT0=Tp_L
        delay_T0()             ;//chạy Timer0
    }
}
//-----
void delay_T0()           //hàm delay T0 chạy Timer0
{
    TCCR0B=0x03            ;/hệ số chia N=64,chạy Timer0
    while(!(TIFR0&(1<<TOV0)));//chờ cờ TOV0=1
    TIFR0 |=(1<<TOV0) ;//xóa cờ TOV0
    TCCR0B=0x00            ;//dừng Timer0
}
```

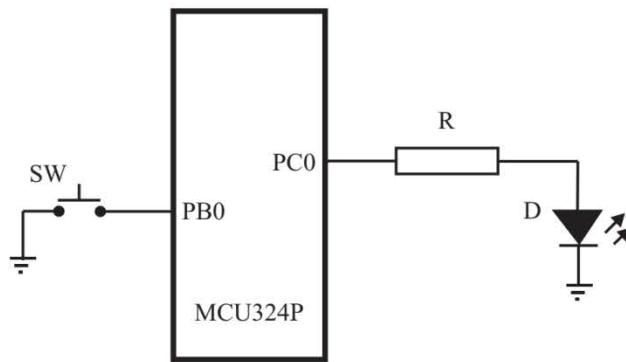
- Chạy mô phỏng chương trình C cho kết quả:  $TpH=299.75\mu s$ ,  $TpL=698.5\mu s$

### ❖ Tạo thời gian trễ tối thiểu đến tối đa sử dụng Timer

Trong ví dụ 7.2, ta thấy để tạo thời gian delay sử dụng Timer phải mất trên 10 chu kỳ máy tương đương hơn 1 $\mu s$  với  $CLKT0=1MC=0.125\mu s$  cho các lệnh điều khiển Timer. Do đó, thời gian delay tạo bởi Timer phải từ vài chục  $\mu s$  trở lên để giảm thiểu sai số. Tuy nhiên ta có thể tăng tần số Fosc lên tối đa 20Mhz để giảm thời gian CLKIO, nhưng sẽ ảnh hưởng đến tốc độ làm việc của toàn hệ thống!

Với  $Fosc=8Mhz$ , 1 xung CLKT0 dài nhất với hệ số chia  $N=1024$  bằng  $128\mu s$ . Bộ đếm TCNT0 8 bit có thể đếm tối đa 256 xung mới tràn. Như vậy thời gian dài nhất để Timer0 đếm tràn bằng  $128x256=32768\mu s$ . Tuy nhiên, ta có thể tăng thêm thời gian delay bằng cách thêm 1 bộ đếm phụ đếm số lần tràn của Timer0, như minh họa trong ví dụ 7.6.

**Ví dụ 7.6:** Từ sơ đồ mạch hình 7.15, viết một chương trình thực hiện cứ mỗi lần nhấn và nhả SW (có chông rung SW), LED sáng trong 1s.



Hình 7.15: Sơ đồ mạch ví dụ 7.6

**Giải:**

Việc nhấn và nhả SW có chia rung SW đã trình bày ở chương 6. Ta sử dụng Timer0 tạo thời gian delay 1s đặt PC0=1 khi nhận dạng có SW nhấn/nhả tại PB0.

Chọn hệ số chia  $N=1024$ , CLKT0=128 $\mu$ s, số xung Timer0 đếm được để tạo thời gian delay 1s:  
 $nT=1000000/128=7812.5 \approx 252 \times 31 = 7812$

Ta chọn giá trị đặt trước cho bộ đếm TCNT0=-252=\$04 và đếm 31 lần tràn Timer0 là đủ 1s.

#### ❖ Chương trình hợp ngữ ví dụ 7.6

```

.DEF COUNT=R19          ;COUNT=bộ đếm số lần tràn Timer0
.EQU LED_OUT=0          ;gán ký hiệu LED_OUT=0
.EQU SW=0                ;gán ký hiệu SW=0
.EQU TP=-252             ;giá trị đặt trước Timer0
.EQU NUM=31              ;số lần tràn Timer0
.ORG 0
RJMP MAIN
.ORG 0X40

MAIN: LDI R16,HIGH(RAMEND) ;đưa stack lên đỉnh SRAM
      OUT SPH,R16
      LDI R16,LOW(RAMEND)
      OUT SPL,R16
      LDI R16,(1<<LED_OUT) ;đặt PC0 output
      OUT DDRC,R16
      LDI R16,0x00            ;đặt PB0 input
      OUT DDRB,R16
      SBI PORTB,SW            ;diện trở kéo lên PB0
      LDI R17,0X00
      OUT TCCR0A,R17          ;Timer0 mode NOR
      LDI R17,0X00
      OUT TCCR0B,R17          ;Timer0 mode NOR,dừng
      CBI PORTC,LED_OUT        ;tắt LED
      LDI COUNT,NUM            ;nap giá trị đếm số lần tràn Timer0
START: LDI R18,50           ;lập vòng liên tục 50 lần kiểm tra SW nhấn
BACK:  SBIC PINB,SW          ;kiểm tra SW nhấn
      RJMP START
      DEC R18
      BRNE BACK
      LDI R18,50
BACK1: SBIS PINB,SW          ;SW nhấn,lập vòng lại từ đầu
      RJMP BACK1
      DEC R18
      BRNE BACK2
      SBI PORTC,LED_OUT        ;đếm số lần lập vòng SW nhấn
      RCALL DELAY_1S            ;thoát khi SW nhấn liên tục đủ số vòng lập
      ;lập vòng liên tục 50 lần kiểm tra SW nhả
      ;kiểm tra SW nhả
      ;SW nhấn,lập vòng lại từ đầu
      ;đếm số lần lập vòng SW nhả
      ;thoát khi SW nhả liên tục đủ số vòng lập
      ;sáng LED
      ;etc chạy Timer0 delay 1s

```

```

CBI    PORTC,LED_OUT      ;tắt LED
RJMP   START              ;lặp vòng lại
;-----  

DELAY_1S: LDI R17,TP  

        OUT TCNT0,R17      ;  

        LDI R17,0X05        ;Timer0 chạy,hệ số chia N=1024  

        OUT TCCR0B,R17  

WAIT:   IN  R17,TIFR0      ;đọc thanh ghi cờ Timer0  

        SBRS R17,TOV0        ;chờ cờ TOV0=1 báo Timer0 tràn  

        RJMP WAIT            ;cờ TOV0=0 tiếp tục chờ  

        OUT TIFR0,R17        ;nạp lại bit TOV0=1 xóa cờ TOV0  

        LDI R17,0X00        ;dừng Timer0  

        OUT TCCR0B,R17  

        DEC COUNT           ;đếm số lần tràn Timer0  

        BRNE DELAY_1S        ;chưa đủ,tiếp tục chạy Timer0  

        LDI COUNT,NUM        ;đủ 1s,nạp lại số đếm và thoát  

        RET

```

#### ❖ Chương trình C ví dụ 7.6

```

#include <avr/io.h>
#define import PINB          //định nghĩa PORTB=input
#define outport PORTC        // định nghĩa PORTC=output
const char led_out=0,sw=0  // ký hiệu giá trị led_out,sw
const char Tp=-252,Num=31 //giá trị đặt trước bộ đếm và số lần báo tràn
unsigned char count,i;
void delay_1s()           // khai báo hàm delay 1s
int main()                // bắt đầu chương trình
{
    DDRB&=~(1<<sw)      //khai báo ngõ SW input
    PORTB |= (1<<sw)      //điện trở kéo lên ngõ SW
    DDRC |=(1<<led_out)   //khai báo ngõ led_out output
    TCCR0A=0x00            //Timer0 mode NOR
    TCCR0B=0x00            //Timer0 mode NOR,dừng Timer0
    outport &=~(1<<led_out) //tắt LED
    count=Num               //đặt số đếm
    while(1)                // lặp vòng vô hạn
    {
        for ( i=50;i>=1;i--) // đọc trạng thái SW nhấn 50 lần
        {
            if(import&(1<<sw)) //SW=1 chưa nhấn
                i=50             // đọc lại từ đầu
        }
        for ( i=50;i>=1;i--) // đọc trạng thái SW nhả 50 lần
        {
            if(!(import&(1<<sw))) //SW=0 chưa nhả
                i=50             // đọc lại từ đầu
        }
        outport |=(1<<led_out) //sáng LED
        delay_1s()              //delay 1s
        outport&=~(1<<led_out) //tắt LED
    }
}
//-----
void delay_1s()           //hàm delay 1s sử dụng Timer0

```

```

{
do
{
    TCNT0=Tp      ;//nạp Timer0 giá trị đặt trước
    TCCR0B=0x05  ;//hệ số chia N=1024, chạy Timer0
    while(!(TIFR0&(1<<TOV0)));//chờ cờ TOV0=1
    TIFR0 |=(1<<TOV0); //xóa cờ TOV0
    TCCR0B=0x00  ;//dừng Timer0
    count--        ;//đếm số lần tràn
}
while(count!=0)    ;//lặp vòng cho đủ số lần tràn
count=Num          ;//nạp lại số đếm
}

```

#### ❖ Câu hỏi ôn tập

- Để tạo thời gian delay đúng 8ms, giá trị đặt trước cho TCNT0 và hệ số chia N bằng bao nhiêu?
- Viết đoạn lệnh khởi động Timer0 mô thức NOR, tạo delay 100μs.
- Cho Fosc=1Mhz, Timer0 có thể tạo delay tối đa bao nhiêu ms (không sử dụng thanh ghi phụ)?
- Viết đoạn lệnh lập vòng kiểm tra cờ TOV0, nếu TOV0=1 xóa TOV0 và thoát theo cách khác với trình bày trong các ví dụ trên.
- Để xóa cờ TOV0, ta thực hiện lệnh theo một trong hai cách sau:

Cách 1: ORI TIFR0,1

Cách 2: OUT TIFR0,1

Cho nhận xét về 2 cách trên.

#### 7.3.3 Hoạt động Timer0 mô thức xóa Timer theo kết quả so sánh

(CTC: Clear Timer on Compare Match)

Mô thức CTC cũng thường được ứng dụng trong định thời, tạo thời gian delay, đếm sự kiện từ xung ngoài, và nhất là tạo sóng với độ rộng xung, tần số thay đổi (sẽ trình bày chi tiết trong phần tạo sóng).

Trong mô thức CTC, cài đặt WGM02:WGM00=010, cài đặt CS02:CS00 chọn tần số xung CLKT0.

Mô thức CTC có 2 kênh A và B làm việc độc lập, mỗi kênh có thanh ghi riêng OCR0A và OCR0B chứa giá trị tham chiếu để bộ đếm TCNT0 so sánh. Ban đầu TCNT0 đếm từ 0x00 lên đến khi giá trị bộ đếm bằng giá trị trong thanh ghi OCR0A hoặc OCR0B, sẽ đáp ứng kết quả so sánh bằng. Ở xung CLKT0 tiếp theo, cờ OCF0A hoặc OCF0B được phần cứng đặt lên 1 chỉ báo nội dung bộ đếm bằng giá trị so sánh tương ứng trong OCR0A hoặc OCR0B. Do quy định giá trị TOP bằng nội dung OCR0A nên khi đạt kết quả so sánh kênh A, cờ OCF0A=1, bộ đếm TCNT0 bị xóa về 0 và tiếp tục đếm lên.

Hình 7-16 minh họa giàn đồ xung định thời Timer0 hoạt động mô thức CTC ở kênh A và B.

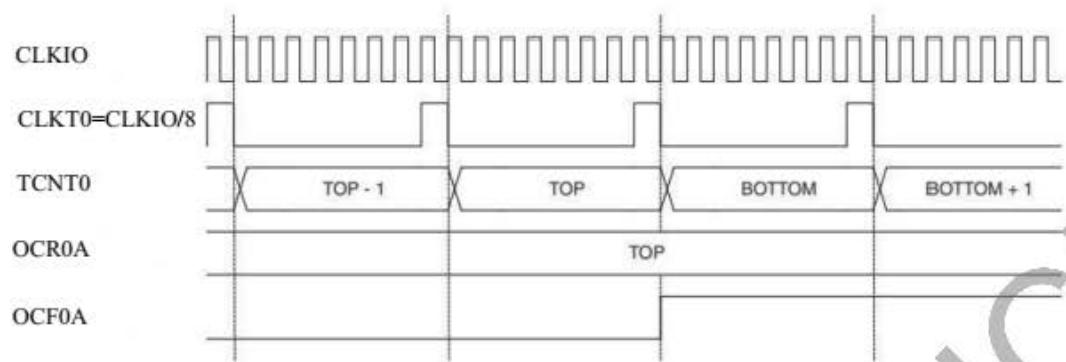
- **Lưu ý:**
  - Cờ OCF0x=1 khi TCNT0 đếm được OCR0x+1 xung
  - Do TOP=OCR0A nên để đạt kết quả so sánh kênh B, cờ OCF0B=1, giá trị đặt trong OCR0B phải không được lớn hơn giá trị đặt trong OCR0A
  - Cờ TOV0 đặt lên 1 chỉ khi đặt OCR0A=MAX=0xFF

#### ❖ Các bước lập trình Timer0 trong mô thức CTC

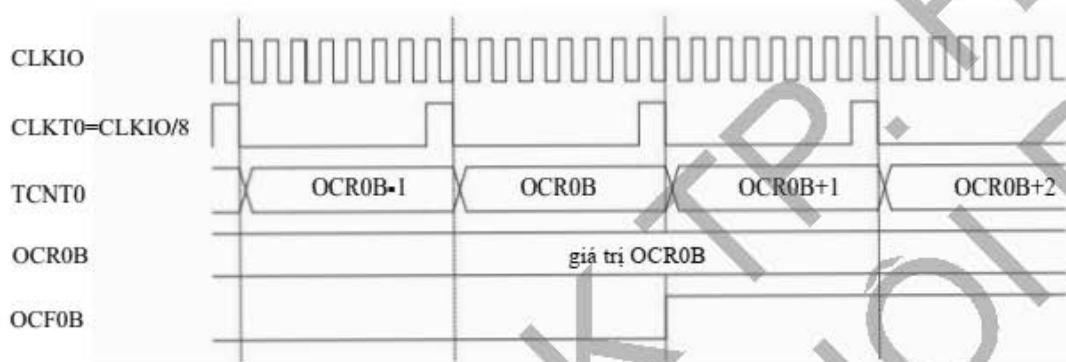
- Nạp giá trị đặt trước ( $n - 1$ ) vào thanh ghi OCR0x( $x=A, B$ ),  $n$  cũng chính là số xung đếm
- Cài đặt mô thức CTC bằng cách đặt WGM02:WGM00=010, cài đặt tần số xung CLKT0 bằng cách chọn tổ hợp CS02:CS00 để có hệ số chia N. Thực hiện bằng cách nạp các giá trị thích hợp vào 2 thanh ghi TCCR0A, TCCR0B. Khi ta cài đặt CS02:CS00 khác 000 Timer0 bắt đầu đếm lên.
- Liên tục kiểm tra cờ OCF0x bằng vòng lặp để phát hiện khi nào OCF0x=1 là đạt kết quả so sánh bằng, thoát khỏi vòng lặp.
- Xóa cờ OCF0x bằng cách ghi 1 vào bit OCF0x, để phát hiện lần báo kết quả so sánh kế tiếp.
- Lặp lại bước 1 nếu có yêu cầu.

- Từ hoạt động mô thức CTC ta thấy mô thức CTC tỏ ra chính xác hơn mô thức NOR trong các ứng dụng tạo thời gian delay, nhất là thời gian ngắn. Do trong mô thức CTC không cần mất thời gian dừng Timer

dễ nạp lại giá trị đặt trước cho bộ đếm, Timer đếm liên tục không dừng nên bù trừ thời gian chờ kiểm tra cờ OCF0x!



(a) Giản đồ xung định thi mô thức CTC Timer0 kênh A



(b) Giản đồ xung định thi mô thức CTC Timer0 kênh B  
OCR0B <= OCR0A = TOP

**Hình 7.16:** Giản đồ xung định thi Timer0 mô thức CTC

Ta viết lại chương trình ví dụ 7.2 sử dụng mô thức CTC.

**Ví dụ 7.7:** Lặp lại ví dụ 7.2, tạo chuỗi xung vuông đối xứng ra PB1, chu kỳ 24 $\mu$ s sử dụng Timer0 mô thức CTC. Dựa vào chương trình đã viết, tính chính xác lại chu kỳ xung ra.

**Giải:**

Theo ví dụ 7.2, số xung đếm được để Timer0 tràn:  $n = \$00 - \$A0 = \$60$

Do đó, giá trị đặt trước cho  $OCR0A = \$60 - 1 = \$5F$

Chương trình hợp ngữ như sau:

```
.EQU P_OUT=1
.ORG 0
RJMP MAIN
.ORG 0X40
MAIN: LDI R16,HIGH(RAMEND) ;đưa stack lên đỉnh SRAM
      OUT SPH,R16
      LDI R16,LOW(RAMEND)
      OUT SPL,R16
      LDI R16,(1<<P_OUT) ;đặt PB1 output
      OUT DDRB,R16
      LDI R17,$5F ;giá trị so sánh
      OUT OCR0A,R17
      LDI R17,0X02 ;Timer0 mode CTC
      OUT TCCR0A,R17
      LDI R17,0X01 ;Timer0 chạy, hệ số chia N=1
      OUT TCCR0B,R17
START: IN  R17,TIFR0 ;đọc thanh ghi cờ Timer0          1MC
       SBRs R17,OCF0A ;chờ cờ OCF0A=1 báo kết quả so sánh kênh A 2/1MC
       RJMP START ;cờ OCF0A=0 tiếp tục chờ 2MC
```

OUT	TIFR0,R17	;nạp lại bit OCF0A=1 xóa cờ OCF0A	1MC
IN	R17,PORTB	;đọc PortB	1MC
EOR	R17,R16	;đảo bit PB1	1MC
OUT	PORTB,R17	;xuất ra PortB	1MC
RJMP	START	;lặp vòng lại	2MC

- Trong chương trình này ta chỉ cần nạp OCR0A giá trị cần so sánh và cho Timer0 chạy mô thức CTC, N=1 ở đầu chương trình.Sau đó chỉ cần kiểm tra cờ OCF0A=1 và xóa cờ,đảo bit ngõ ra P\_OUT

- Do độ rộng xung ngắn chỉ 12μs,nên ta chuyển đoạn lệnh kiểm tra cờ thành chương trình chính và tính thời gian thực thi các lệnh từ lúc nhận dạng cờ OCF0A=1 đến khi đảo bit ngõ ra xong phải nhỏ hơn thời đoạn xung 12μs.Ở đoạn lệnh trên dài 11MC=11x0.125=1.375μs,vẫn nằm trong thời đoạn xung 12μs nên độ rộng xung ngõ ra vẫn chính xác.

**Ví dụ 7.8:** Tim tần số xung ngõ ra PB5 trong chương trình sau:

- (a) Bỏ qua thời gian thực thi các lệnh kiểm tra cờ và đảo bit
- (b) Tính chính xác độ rộng xung ra

.EQU	P_OUT=5		
.ORG	0		
RJMP	MAIN		
.ORG	0X40		
MAIN:	LDI R16,HIGH(RAMEND)	;đưa stack lên đỉnh SRAM	
	OUT SPH,R16		
	LDI R16,LOW(RAMEND)		
	OUT SPL,R16		
	LDI R16,(1<<P_OUT)	;đặt PB5 output	
	OUT DDRB,R16		
	LDI R17,89	;giá trị so sánh	
	OUT OCR0A,R17		
	LDI R17,0X02	;Timer0 mode CTC	
	OUT TCCR0A,R17		
	CLR R17		
	OUT PORTB,R17	;R17=0X00	
START:	LDI R20,0X03	;xuất ra PortB	
	OUT TCCR0B,R20		
AGAIN:	IN R20,TIFR0	;Timer0 chạy,hệ số chia N=64	1MC
	SBRS R20,OCF0A		1MC
	RJMP AGAIN	;chờ cờ OCF0A=1 báo kết quả so sánh kênh A	2/1MC
	OUT TIFR0,R20	;cờ OCF0A=0 tiếp tục chờ	2MC
	EOR R17,R16	;nạp lại bit OCF0A=1 xóa cờ OCF0A	1MC
	OUT PORTB,R17	;đảo bit PB5	1MC
	RJMP START	;xuất ra PortB	1MC
		;lặp vòng lại	2MC

**Giải:**

(a) CLKT0=0.125x64=8μs,số xung bộ đếm để đạt kết quả so sánh OCF0A=1 n=89+1=90.  
Thời gian độ rộng xung ngõ ra không tính các lệnh điều khiển: Td=90x8=720μs.

$$\text{Tần số xung ra: } f_o = 10^6 / (720 \times 2) = 694.4 \text{Hz}$$

(b) Nếu tính các lệnh điều khiển và lập vòng sẽ mất 12MC=12x0.125=1.5μs < Td=720μs.Do đó độ rộng xung ngõ ra vẫn là Td=720μs.

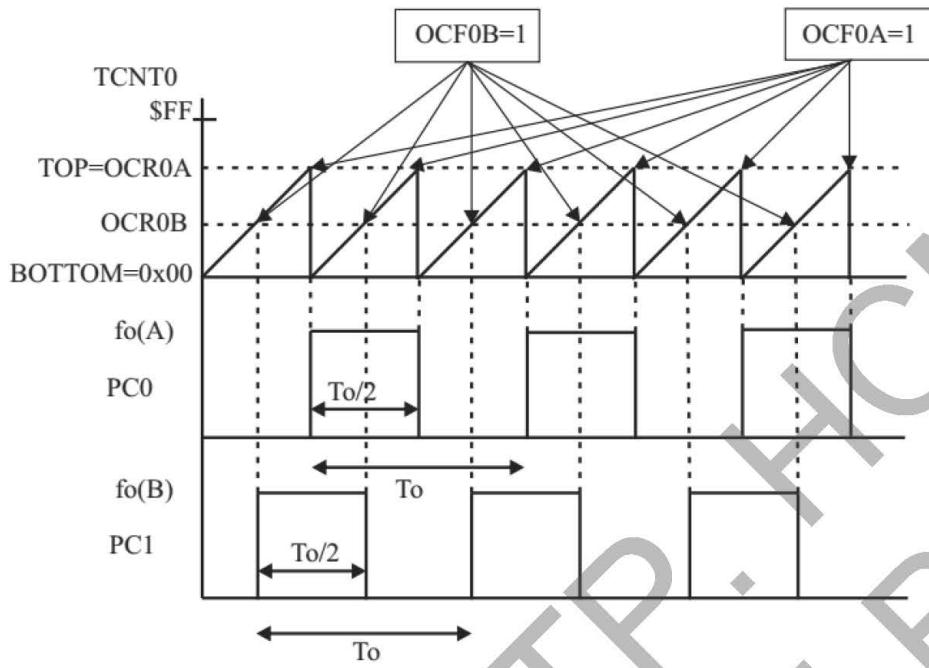
Ví dụ sau minh họa việc sử dụng đồng thời 2 kênh A,B trong mô thức CTC và sự thay đổi linh động của tần số xung đếm Timer0 qua bộ chia đặt trước.

**Ví dụ 7.9:** Thiết kế một bộ tổng hợp tần số tạo 2 chuỗi xung vuông đối xứng vuông pha tần số lập trình được  $f_o(A) = f_o(B) = 10 \text{Khz}/n$ ,  $n = 1 \div 255$ .

**Giải:**

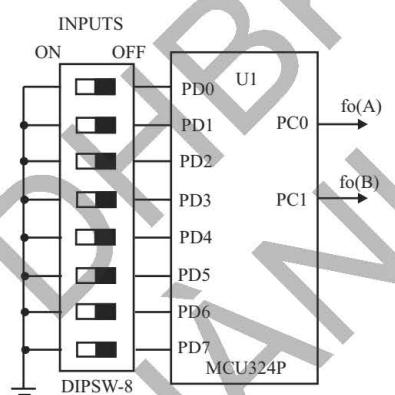
- Tần số xung ra sẽ được quyết định bởi hệ số chia N cho CLKT0 và giá trị TOP= OCR0A

- Để tạo 2 xung ra vuông pha ta đặt  $OCR0B=OCR0A/2$ , đáp ứng xung ra như hình 7.17



**Hình 7.17:** Dạng xung 2 ngõ ra vuông pha Timer0 mô thức CTC2 kênh A và B

Sơ đồ thiết kế như hình 7.18.



**Hình 7.18:** Sơ đồ mạch ví dụ 7.8

- DIPSW-8 ngõ vào đặt hệ số  $n$  ( $n=1 \div 255$ ) chia tần số

- Ngõ ra xung  $fo(A)=PC0, fo(B)=PC1$

- Ý tưởng giải thuật chương trình:

- Nhập giá trị  $n$  từ PortD, tính độ rộng xung tương ứng và suy ra hệ số chia  $N$  bộ chia đặt trước sao cho đạt sai số thấp nhất và giá trị  $TOP < 255$ , ta được Bảng 7.5:

**Bảng 7.5:** Tóm tắt tầm giá trị ngõ vào  $n$ , chu kỳ xung  $To$ , hệ số chia  $N$  và giá trị  $TOP$

$n$	$To=100xn(\mu s)$	$To/2=50xn(\mu s)$	$N$	$CLKT0(\mu s)$	$TOP+1=(To/2)/CLKT0$
1 - 5	100 - 500	50 - 250	8	1	50 - 250
6 - 40	600 - 4000	300 - 2000	64	8	37 - 250
41 - 160	4100 - 16000	2050 - 8000	256	32	64 - 250
161 - 255	16100 - 25500	8050 - 12750	1024	128	62 - 99

- Như vậy theo bảng 7.5, tùy thuộc vào  $n$  ta sẽ đặt hệ số chia  $N$  và tính ra giá trị  $TOP$  tương ứng

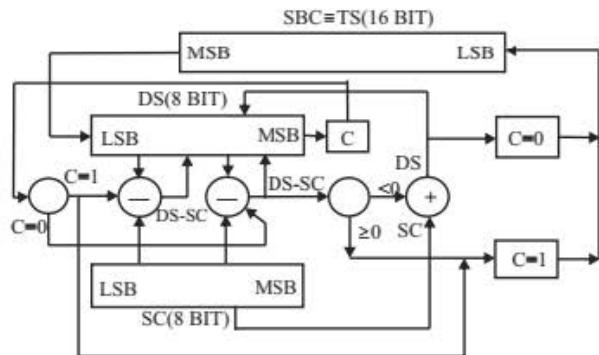
- Đặt  $OCR0A=TOP$  và  $OCR0B=(TOP+1)/2-1$ , cho Timer0 chạy mô thức CTC tạo xung ra PC0, PC1

- Để tính  $TOP+1$  ta chia làm 2 bước:

1. Tính  $To/2=50xn$  (16 bit)

2. Gọi chương trình con chia 16 bit cho 8 bit tính  $TOP+1=(To/2)/CLKT0$

- Giải thuật chương trình con DIV16\_8 chia số nhị phân 16 bit cho 8 bit như hình 7.19



**Hình 7.19:** Lưu đồ phép chia số nhị phân 16 bit cho 8 bit

- Số bị chia(SBC) cũng là thương số(TS)16 bit,số chia(SC:8 bit),dư số(DS:8bit)

1. Nạp SBC,SC và DS=0x00 vào các thanh ghi
2. Dịch trái SBC vào DS,MSB SBC dịch vào LSB DS,MSB DS dịch qua C
3. Nếu C=1,trừ DS cho SC cất vào DS,đặt C=1,chuyển qua bước 5,nếu C=0 tiếp bước 4
4. Trừ DS cho SC cất vào DS,nếu kết quả trừ dương đặt C=1,nếu kết quả âm xóa C=0 và cộng kết quả với SC trả lại DS như trước.
5. Nạp bit C vào LSB SBC
6. Lặp lại bước 2 16 lần cho kết quả TS(16 bit),DS(8 bit)

#### ❖ Chương trình hợp ngữ ví dụ 7.9

```

.DEF OPD1_L=R20
.DEF OPD1_H=R21
.DEF OPD2=R22
.DEF OPD3=R23
.DEF COUNT=R19
.EQU F0_A=0
.EQU F0_B=1
.EQU TP=50
.ORG 0
RJMP MAIN
.ORG 0X40
MAIN: LDI R16,HIGH(RAMEND) ;đưa stack lên đỉnh SRAM
      OUT SPH,R16
      LDI R16,LOW(RAMEND)
      OUT SPL,R16
      LDI R16,(1<<F0_A)|(1<<F0_B);đặt F0_A,F0_B output
      OUT DDRC,R16
      LDI R16,0X00          ;PortD input
      OUT DDRD,R16
      LDI R16,0xFF          ;điện trở kéo lên PortD
      OUT PORTD,R16
      LDI R17,0X02          ;Timer0 mode CTC
      OUT TCCR0A,R17
      CLR R17              ;R17=0X00
      OUT PORTC,R17         ;xuất ra PortC
START: RCALL GET_SCA
      RCALL CTC_AB
      RJMP START
;
;GET_SCA đọc giá trị n từ PortD
;So sánh n với 4 tầm (0÷5):CS2:0=2,(6÷40):CS2:0=3,(41÷160):CS2:0=4,(>160):CS2:0=5
;Tính giá trị TOP+1 và suy ra giá trị đặt OCR0A,OCR0B

```

;Input: PIND  
;Output: TCCR0B,OCR0A,OCR0B  
;Sử dụng R16,R17,ctc DIV16\_8

-----  
GET\_SCA: IN R17,PIND ;đọc n  
          CPI R17,6 ;n<6 tầm 1,CLKT0=1μs  
          BRCC CHK1 ;n≥6 kiểm tra tiếp  
          LDI R16,TP ;nạp hệ số đơn vị To/2  
          MUL R17,R16 ;TOP+1=nx(To/2)  
          MOV OPD1\_L,R0 ;cắt TOP+1 vào OPD1\_L  
          LDI R16,0X02 ;CS02:CS00=010  
          OUT TCCR0B,R16 ;Timer0 mode CTC,N=8  
          RJMP LD\_OCR0 ;nạp giá trị cho OCR0x  
CHK1:    CPI R17,161 ;n<161?  
          BRCC CHK3 ;n≥161 tầm 4  
          CPI R17,41 ;n<41 tầm 2  
          BRCC CHK2 ;n≥41 tầm 3  
          LDI OPD2,8 ;CLKT0=8μs  
          LDI R16,0X03 ;CS2:0=011  
          OUT TCCR0B,R16 ;Timer0 mode CTC,N=64  
          RJMP TOP\_CAL  
CHK2:    LDI OPD2,32 ;CLKT0=32μs  
          LDI R16,0X04 ;CS2:0=100  
          OUT TCCR0B,R16 ;Timer0 mode CTC,N=256  
          RJMP TOP\_CAL  
CHK3:    LDI OPD2,128 ;CLKT0=128μs  
          LDI R16,0X05 ;CS2:0=101  
          OUT TCCR0B,R16 ;Timer0 mode CTC,N=1024  
TOP\_CAL: LDI R16,TP ;nạp hệ số đơn vị To/2  
          MUL R17,R16 ;R1:R0=nx(To/2)  
          MOVW OPD1\_L,R0 ;chuyển kết quả nhân vào SBC  
          RCALL DIV16\_8 ;ctc chia nx(To/2)/CLKT0  
LD\_OCR0: MOV R17,OPD1\_L ;cắt TS=TOP+1  
          DEC R17 ;trừ bớt 1  
          OUT OCR0A,R17 ;nạp TOP vào OCR0A  
          LSR OPD1\_L ;chia 2 TOP+1  
          DEC OPD1\_L ;trừ bớt 1  
          OUT OCR0B,OPD1\_L;nạp vào OCR0B  
          RET  
-----

;DIV16\_8 chia số nhị phân 16 bit OPD1 cho 8 bit OPD2  
;Input: OPD1\_H,OPD1\_L=SBC(GPR16-31)  
;        OPD2=SC(GPR0-31)  
;Output:OPD1\_H,OPD1\_L=thương số  
;        OPD3=DS(GPR0-31)  
;Sử dụng COUNT(GPR16-31)

-----  
DIV16\_8: LDI COUNT,16 ;COUNT=đếm 16  
          CLR OPD3 ;xóa dư số  
SH\_NXT: CLC ;C=0=bit thương số  
          LSL OPD1\_L ;dịch trái SBC L,bit0=C=thương số  
          ROL OPD1\_H ;quay trái SBC H,C=bit7  
          ROL OPD3 ;dịch bit7 SBC H vào dư số  
          BRCS OV\_C ;tràn bit C=1,chia được  
          SUB OPD3,OPD2 ;trừ dư số với số chia

	BRCC	GT_TH	;C=0 chia được	
	ADD	OPD3,OPD2	;C=1 không chia được,không trừ	
	RJMP	NEXT		
OV_C:	SUB	OPD3,OPD2	;trừ dư số với số chia	
GT_TH:	SBR	OPD1_L,1	;chia được,thương số=1	
NEXT:	DEC	COUNT	;đếm số lần dịch SBC	
	BRNE	SH_NXT	;chưa đủ tiếp tục dịch bit	
		RET		
-----				
	;CTC_AB kiểm tra cờ OCF0A,OCF0B			
	;Đảo bit F0_A hoặc F0_B và xóa cờ tương ứng nếu cờ báo =1			
	;Input: R16,R17			
-----				
CTC_AB:	IN	R17,TIFR0	;đọc thanh ghi cờ Timer0	1MC
	SBRC	R17,OCF0B	;cờ OCF0B=0 kiểm tra kênh A	2/1MC
	RJMP	B_CHAN	;cờ OCF0B=1 xử lý kênh B	2MC
	SBRS	R17,OCF0A	;cờ OCF0A=1 xử lý kênh A	2/1MC
	RJMP	CTC_AB	;cờ OCF0A=0 tiếp tục chờ	2MC
	OUT	TIFR0,R17	;nạp lại bit OCF0A=1 xóa cờ OCF0A	1MC
	IN	R17,PORTC	;đọc PortC	1MC
	LDI	R16,(1<<F0_A)	;đặt bit vị trí F0_A=1	1MC
	EOR	R17,R16	;đảo bit F0_A	1MC
	OUT	PORTC,R17	;xuất ra PortC	1MC
	RJMP	EXIT_CTC	;thoát	2MC
B_CHAN:	OUT	TIFR0,R17	;nạp lại bit OCF0B=1 xóa cờ OCF0B	1MC
	IN	R17,PORTC	;đọc PortC	1MC
	LDI	R16,(1<<F0_B)	;đặt bit vị trí F0_B=1	1MC
	EOR	R17,R16	;đảo bit F0_B	1MC
	OUT	PORTC,R17	;xuất ra PortC	1MC
EXIT_CTC:	RET			4MC

- Trong ctc GET\_SCA,với n=(1÷5),CLKT0=1,nên TOP+1=nxTp=nx50,do đó không cần tính phép chia sẽ bị mất thời gian chia,làm tổng thời gian thực thi việc đảo bit lớn hơn độ rộng xung,chu kỳ xung ngõ ra không chính xác.Trong các trường hợp CLKT0>1μs phải sử dụng ctc DIV16\_8.

- Phần tính tổng thời gian thực thi việc đảo bit ngõ ra xem như bài tập dành cho người đọc

#### ❖ Chương trình C ví dụ 7.9

```
#include <avr/io.h>
#define outport PORTC
#define import PIND
const char f0_A=0,f0_B=1
const char Tp=50
void get_sea()
void ctc_AB()
unsigned int tam
int main()
{
    DDRC |=(1<<f0_A)|(1<<f0_B) ;//khai báo f0_A/B output
    DDRD=0X00 ;//khai báo portD input
    PORTD=0Xff ;//điện trở kéo lên PortD
    TCCR0A=0x02 ;//Timer0 mode CTC
    outport=0x00 ;//xóa các ngõ ra
    while(1) ;//lặp vòng vô hạn
}
```

```

        get_sca()           ;//tính hệ số chia N và giá trị nạp OCR0x
        ctc_AB()            ;//điều khiển xuất xung ngõ ra f0_A/B
    }
}

//-----
void get_sca()          //hàm hệ số chia N và giá trị nạp OCR0x
{
    if(inport<6)         //n<6 CLKT0=1
    {
        tam=Tp*inport   ;//tam=nx(To/2)/CLKT0
        OCR0A= tam - 1  ;//tính giá trị nạp OCR0A
        OCR0B=tam/2 - 1 ;//tính giá trị nạp OCR0B
        TCCR0B=0x02      ;//hệ số chia N=8,CLKT0=1μs,chạy Timer0
    }
    else if(inport>160)  //n>160 CLKT0=128
    {
        tam=Tp*inport/128 ;//tam=nx(To/2)/CLKT0
        OCR0A= tam - 1  ;//tính giá trị nạp OCR0A
        OCR0B=tam/2 - 1 ;//tính giá trị nạp OCR0B
        TCCR0B=0x05      ;//hệ số chia N=1024,CLKT0=128μs,chạy Timer0
    }
    else if(inport>40)   //160>=n>40 CLKT0=32
    {
        tam=Tp*inport/32 ;//tam=nx(To/2)/CLKT0
        OCR0A= tam - 1  ;//tính giá trị nạp OCR0A
        OCR0B=tam/2 - 1 ;//tính giá trị nạp OCR0B
        TCCR0B=0x04      ;//hệ số chia N=256,CLKT0=32μs,chạy Timer0
    }
    else                  //40>=n>5 CLKT0=8
    {
        tam=Tp*inport/8  ;//tam=nx(To/2)/CLKT0
        OCR0A= tam - 1  ;//tính giá trị nạp OCR0A
        OCR0B=tam/2 - 1 ;//tính giá trị nạp OCR0B
        TCCR0B=0x03      ;//hệ số chia N=64,CLKT0=8μs,chạy Timer0
    }
}
//-----
void ctc_AB()           //hàm ctc_AB xóa cờ OCF0A/B khi đặt/đảo bit ngõ ra f0_A/B
{
    while(!(TIFR0&((1<<OCF0A)|(1<<(OCF0B)))));//chờ cờ OCF0A/B=1
    if(TIFR0&(1<<OCF0B)) //cờ OCF0B=1
    {
        TIFR0 |=(1<<OCF0B); //xóa cờ OCF0B
        outport^=(1<<f0_B); //đảo bit f0_B
    }
    else if(TIFR0&(1<<OCF0A)) //cờ OCF0A=1
    {
        TIFR0 |=(1<<OCF0A); //xóa cờ OCF0A
        outport^=(1<<f0_A); //đảo bit f0_A
    }
}

```

- Chương trình C viết hàm get\_sca() tính giá trị nạp OCR0x khá đơn giản.Tuy nhiên ta sẽ không kiểm soát được tổng thời gian thực thi việc đảo bit,phải xem trình biên dịch chéo qua hợp ngữ trong file.lss hoặc chạy thực nghiệm mô phỏng.

## ❖ Câu hỏi ôn tập

1. Giải thích tại sao sử dụng mô thức CTC chính xác hơn mô thức NOR trong các ứng dụng định thời nhất là thời gian ngắn khoảng hàng chục  $\mu s$ .
2. Đặt OCR0A=0x64, Timer0 đếm bao nhiêu xung bộ đếm trở về 0 trong mô thức CTC?
3. Viết một đoạn lệnh khởi động Timer0 chạy mô thức CTC, tạo thời gian delay 1.2ms.
4. Viết một đoạn lệnh khởi động Timer0 chạy mô thức CTC, tạo 2 chuỗi xung vuông đối xứng tần số 50Hz, lệch pha nhau 60°.
5. Xem lại chương trình hợp ngữ ví dụ 7.9, tính chính xác thời gian đảo bit 2 ngõ ra f0\_A và f0\_B.

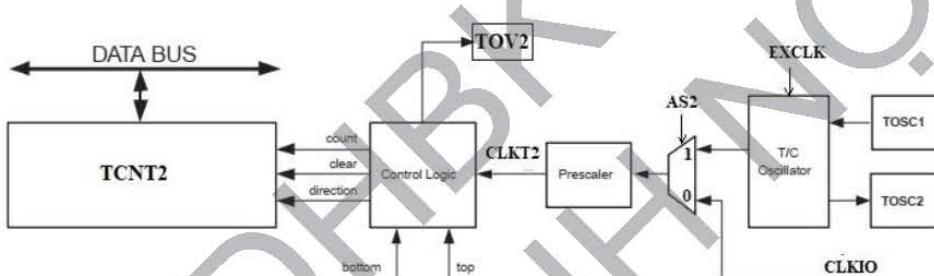
## 7.4 Lập trình hoạt động Timer2

### 7.4.1 Các thanh ghi Timer2

Timer2 là bộ Timer 8 bit. Các thanh ghi Timer2 như TCNT2, TCCR2A, TCCR2B, OCR2A, OCR2B, TIMSK2, TIFR2 có chức năng hoàn toàn giống như Timer0, chỉ khác Timer0 các điểm sau đây:

1. Timer2 không có nguồn tạo xung CLKT2 từ ngõ ngoài T0 như Timer0. Thay vào đó, Timer2 ở mô thức chọn nguồn CLKT2 ngoài, xung đếm lấy từ bộ dao động thạch anh gắn vào ngõ TOSC1(PC6) và TOSC2(PC7), hoặc tín hiệu dao động ngoài đưa vào ngõ TOSC1.
2. Các bit CS2:CS0 chỉ có chức năng chọn hệ số chia N từ bộ chia đặt trước. Việc chọn nguồn xung CLKT2 nhờ vào bit AS2 thuộc thanh ghi ASSR, và việc chọn giữa dao động từ thạch anh gắn vào ngõ TOSC1, TOSC2 hay dao động ngoài đưa vào ngõ TOSC1 do bit EXCLK thuộc thanh ghi ASSR quyết định.

Hình 7.20 minh họa sơ đồ khối nguồn tạo xung CLKT2.



Hình 7.20: Sơ đồ khối nguồn tạo xung CLKT2

- Lưu ý: - Tần số dao động chủ Fosc tối thiểu phải lớn hơn 4 lần tần số dao động ngõ TOSC1(TOSC2)
  - Bộ dao động thiết kế tối ưu với thạch anh  $Fxtal = 32768Hz = 2^{15} Hz$  làm mạch đồng hồ, do có thể kết hợp bộ chia đặt trước và bộ đếm có đặt trước chia  $2^{15}$  tạo xung 1s chính xác!

Phản sau đây ta chỉ mô tả các thanh ghi Timer2 có chức năng khác với Timer0, còn các thanh ghi có chức năng hoàn toàn giống như Timer0 xem lại ở mục 7.3.1, chỉ cần đổi chỉ số Timer 0 thành chỉ số 2.

- Lưu ý: Ngoài trừ 2 thanh ghi TIFR2 và GTCCR có địa chỉ nằm trong vùng I/O, truy xuất bằng các lệnh IN, OUT, các thanh ghi còn lại nằm trong vùng I/O mở rộng, địa chỉ MEM lớn hơn 0x60, nên phải truy xuất như SRAM bằng các lệnh LDS, STS!

### 1. Thanh ghi TCCR2B(Timer/Counter Control Register B)

Bit (0xB1)	7	6	5	4	3	2	1	0	TCCR2B
Read/Write	W	W	R	R	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

Hình 7.21: Thanh ghi TCCR2B

Thanh ghi TCCR2B có địa chỉ MEM=0xB1, truy xuất bit được, đọc ghi 4 bit thấp và chỉ ghi hai bit cao 7 và 6, cài đặt tần số xung CLKT2 cho bộ đếm, kết hợp với thanh ghi TCCR2A cài đặt các mô thức hoạt động của Timer2, điều khiển trực tiếp các ngõ ra OC2A, OC2B.

- Bit 7 FOC2A(Force Output Compare A)

Chức năng tương tự như bit FOC0A Timer0

- Bit 6 FOC2B(Force Output Compare B)**  
Chức năng tương tự như bit FOC0B Timer0
- Bit 3 WGM22(Waveform Generation Mode)**  
Bit WGM22 kết hợp với bit WGM21,WGM20 thuộc thanh ghi TCCR2A có chức năng tương tự như 3 bit WGM02,WGM01,WGM00 trong Timer0.
- Bit 2:1:0 CS22:CS21:CS20 (Clock Select)**  
Ba bit này cài đặt nguồn tạo xung CK cho bộ đếm Timer2 như mô tả trong Bảng 7.7

Bảng 7.7: Lựa chọn nguồn tạo xung CK Timer2

CS22	CS21	CS20	Tần số xung CLKT2
0	0	0	Không có xung CK(Timer2 dừng)
0	0	1	CLKIO(N=1:không qua bộ chia đắt trước)
0	1	0	CLKIO/8(N=8:từ bộ chia đắt trước)
0	1	1	CLKIO/32(N=32:từ bộ chia đắt trước)
1	0	0	CLKIO/64(N=64:từ bộ chia đắt trước)
1	0	1	CLKIO/128(N=128:từ bộ chia đắt trước)
1	1	0	CLKIO/256(N=256:từ bộ chia đắt trước)
1	1	1	CLKIO/1024(N=1024:từ bộ chia đắt trước)

- Để Timer2 dừng đếm ta đặt CS22:CS20=000
- Với CS22:CS20=001 đến 111 chọn 7 hệ số chia tương ứng N=1,8,32,64,128,256,1024.Bộ chia đắt trước chia cả xung CK Timer2 lấy từ dao động nội hay dao động bên ngoài ngõ TOSC1(TOSC2)

## 2. Thanh ghi ASSR(Asynchronous Status Register): Thanh ghi trạng thái bắt đồng bộ

Bit (0xB6)	7	6	5	4	3	2	1	0	ASSR
Read/Write	-	EXCLK	AS2	TCN2UB	OCR2AUB	OCR2BUB	TCR2AUB	TCR2BUB	
Initial Value	R	R/W	R/W	R	R	R	R	R	

Hình 7.22: Thanh ghi ASSR

Thanh ghi ASSR có địa chỉ MEM=0xB6, truy xuất bit được đọc/ghi bit 5,6 và chỉ đọc các bit còn lại.Thanh ghi ASSR có chức năng khai báo chọn nguồn tạo xung CK Timer2 và chỉ báo trạng thái cập nhật nội dung các thanh ghi TCNT2,OCR2A/B,TCCR2A/B trong mô thức chọn xung CLKT2 ngoài, còn gọi là mô thức bắt đồng bộ.

- Bit 6 EXCLK(Enable External Clock Input): Bit cho phép ngõ vào xung Clock ngoài**  
Ghi bit EXCLK=1 chọn dao động ngoài đưa vào ngõ TOSC1,ngõ TOSC2 để trống.Nếu bit này bằng 0 chọn dao động từ thạch anh nối ngõ TOSC1 và TOSC2.Việc chọn bit EXCLK phải được thực hiện trước khi Timer2 chạy mô thức bắt đồng bộ và chỉ có ý nghĩa khi đặt bit AS2=1 chọn dao động ngoài cho Timer2.
- Bit 5 AS2(Asynchronous Timer/Counter2): Bit Timer2 bắt đồng bộ**  
Khi xóa AS2=0,Timer2 chọn xung CK từ dao động nội CLKIO.Khi đặt AS2=1,Timer2 chọn xung CK từ ngõ ngoài qua thạch anh nối ngõ TOSC1 và TOSC2 hoặc dao động ngoài đưa vào ngõ TOSC1,do bit EXCLK quyết định.Khi chuyển trạng thái bit AS2,nội dung các thanh ghi TCNT2,TCCR2A/B,OCR2A/B có thể bị sai cần phải cập nhật lại!
- Bit 4 TCN2UB(Timer/Counter2 Update Busy): Cờ báo bận cập nhật Timer2**  
Khi ghi TCNT2 trong mô thức bắt đồng bộ,thực chất là chỉ mới ghi vào thanh ghi đếm của TCNT2,phần cứng đặt bit TCN2UB=1 báo bận.Cho đến khi TCNT2 cập nhật mới nội dung từ thanh ghi đếm,phần cứng xóa bit TCN2UB chỉ báo TCNT2 đã cập nhật giá trị mới ghi.
- Bit 3 OCR2AUB(Output Compare Register2 A Update Busy): Cờ báo bận cập nhật thanh ghi OCR2A**  
Khi ghi OCR2A trong mô thức bắt đồng bộ,thực chất là chỉ mới ghi vào thanh ghi đếm của OCR2A,phần cứng đặt bit OCR2AUB=1 báo bận.Cho đến khi OCR2A cập nhật mới nội dung từ thanh ghi đếm,phần cứng xóa bit OCR2AUB=0 chỉ báo OCR2A đã cập nhật giá trị mới ghi.
- Bit 2 OCR2BUB(Output Compare Register2 B Update Busy): Cờ báo bận cập nhật thanh ghi OCR2B**

Khi ghi OCR2B trong mô thức bắt đồng bộ, thực chất là chỉ mới ghi vào thanh ghi đệm của OCR2B, phần cứng đặt bit OCR2BUB=1. Cho đến khi OCR2B cập nhật mới nội dung từ thanh ghi đệm, phần cứng xóa bit OCR2BUB=0 chỉ báo OCR2B đã cập nhật giá trị mới ghi.

- **Bit 1 TCR2AUB(Timer/Counter Control Register2 A Update Busy): Cờ báo bận cập nhật thanh ghi TCCR2A**

Khi ghi TCCR2A trong mô thức bắt đồng bộ, thực chất là chỉ mới ghi vào thanh ghi đệm của TCCR2A, phần cứng đặt bit TCR2AUB=1 báo bận. Cho đến khi TCCR2A cập nhật mới nội dung từ thanh ghi đệm, phần cứng xóa bit TCR2AUB=0 chỉ báo TCCR2A đã cập nhật giá trị mới ghi.

- **Bit 0 TCR2BUB(Timer/Counter Control Register2 B Update Busy): Cờ báo bận cập nhật thanh ghi TCCR2B**

Khi ghi TCCR2B trong mô thức bắt đồng bộ, thực chất là chỉ mới ghi vào thanh ghi đệm của TCCR2B, phần cứng đặt bit TCR2BUB=1 báo bận. Cho đến khi TCCR2B cập nhật mới nội dung từ thanh ghi đệm, phần cứng xóa bit TCR2BUB=0 chỉ báo TCCR2B đã cập nhật giá trị mới ghi.

➤ **Lưu ý:**

- Các thanh ghi TCNT2, OCR2A/B, TCCR2A/B đều có thanh ghi đệm. Trong mô thức bắt đồng bộ, việc ghi vào các thanh ghi trên thực chất là ghi vào thanh ghi đệm tương ứng trước, và sau đó ít nhất 2 chu kỳ xung dao động chủ các thanh ghi mới cập nhật nội dung ghi từ thanh ghi đệm.
- Hoạt động ghi bất kỳ thanh ghi nào trong 5 thanh ghi trên trong khi có 1 bit cờ báo bận có thể sẽ làm sai nội dung các thanh ghi này. Do đó khi ghi nội dung mới vào bất kỳ một trong 5 thanh ghi trên trong mô thức bắt đồng bộ, phải chờ cờ báo bận tương ứng xóa về 0 mới tiếp tục ghi thanh ghi khác.
- Khi đọc TCNT2, giá trị đọc chính là nội dung bộ đếm Timer2, còn nếu đọc OCR2A/B, TCCR2A/B giá trị đọc là nội dung của thanh ghi đệm tương ứng.

### 3. Thanh ghi GTCCR(General Timer/Counter Control Register): Thanh ghi điều khiển chung Timer

Bit	7	6	5	4	3	2	1	0	
0x23 (0x43)	TSM	-	-	-	-	-	PSRASY	PSRSYNC	GTCCR
Read/Write	R/W	R	R	R	R	R	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

Hình 7.23: Thanh ghi GTCCR

Thanh ghi GTCCR có địa chỉ I/O=0x23, đọc/ghi và truy xuất bit được. Thanh ghi GTCCR điều khiển các Timer reset và hoạt động đồng bộ.

- **Bit 7 TSM(Timer/Counter Synchronous mode): Bit đồng bộ Timer**

Bit TSM=1 tác động các Timer hoạt động mô thức đồng bộ. Trong mô thức này, các bit PSRASY, PSRSYNC được giữ nguyên mức 1 để duy trì trạng thái reset bộ chia đặt trước. Điều này đảm bảo các Timer dừng và có thể cài đặt lại cấu hình hoạt động mà không gây lỗi. Khi xóa bit TSM=0, phần cứng sẽ xóa các bit PSRASY, PSRSYNC=0, Timer sẽ chạy ngay tức thì.

- **Bit 1 PSRASY(Prescale Reset Timer/Counter2): Bit reset bộ chia đặt trước Timer2**

Bit PSRASY=1 reset bộ chia đặt trước Timer2. Bình thường phần cứng sẽ xóa bit PSRASY=0 ngay lập tức. Nếu Timer2 đang hoạt động mô thức bắt đồng bộ, bit này vẫn giữ mức 1 cho đến khi bộ chia đặt trước reset mới xóa về 0. Bit PSRASY vẫn giữ nguyên mức 1 nếu bit TSM=1 để đảm bảo trạng thái bộ chia đặt trước reset và Timer2 dừng, cho đến khi TSM=0, phần cứng sẽ xóa bit này về 0.

- **Bit 0 PSRSYNC(Prescale Reset Timer/Counter(0/1)): Bit reset bộ chia đặt trước Timer0,1**

Bit PSRSYNC=1 reset bộ chia đặt trước Timer0/1. Bình thường phần cứng sẽ xóa bit PSRSYNC=0 ngay lập tức. Bit PSRSYNC vẫn giữ nguyên mức 1 nếu bit TSM=1 để đảm bảo trạng thái bộ chia đặt trước reset và Timer0/2 dừng, cho đến khi TSM=0, phần cứng sẽ xóa bit này về 0.

➤ **Lưu ý:** Timer0 và Timer1 sử dụng chung bộ chia đặt trước, nên reset bộ chia đặt trước sẽ tác động đến cả 2 Timer0/1

## ❖ Câu hỏi ôn tập

1. Cho biết chức năng các bit AS2 và EXCLK.
2. Khi đặt AS2=1,EXCLK=0,bộ thạch anh fXtal=32768Hz gắn ở 2 chân TOSC1,TOSC2, CS22:CS20=011,một xung đếm Timer2 dài bao lâu?
3. Cho biết chức năng các cờ TCN2AUB,OCR2AUB,OCR2BUB,TCR2AUB,TCR2BUB.
4. Khi Timer2 chạy mô thức bất đồng bộ(AS2=1),làm sao để biết bộ đếm đã cập nhật giá trị mới
5. Xem đoạn lệnh sau đây:

```
WAIT: SBIS    TIFR2,OCF2A  
        RJMP    WAIT  
        CBI     TIFR2,OCF2A  
        ...  
        LDI     R17,99  
        OUT    OCR2A,R17  
        ...
```

Cho biết đoạn lệnh trên có bị lỗi không? Giải thích chi tiết lỗi nếu có.

### 7.4.2 Hoạt động Timer2 mô thức NOR và CTC

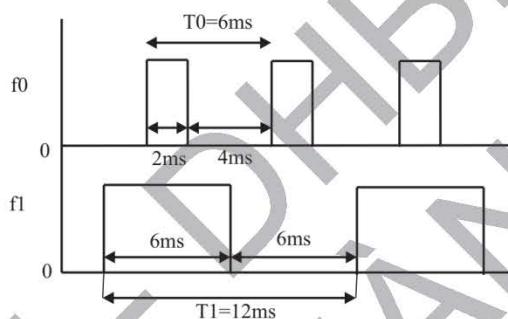
Timer2 hoạt động mô thức NOR và CTC hoàn toàn giống như Timer0.

Trong mô thức NOR,khi thanh ghi TCNT2 đếm từ giá trị đặt trước đến TOP=0xFF và tràn về 0x00,cờ TOV2 đặt lên 1 bằng phần cứng và xóa về 0 bằng phần mềm.

Trong mô thức CTC,TCNT2 đếm từ 0 đến giá trị bằng nội dung thanh ghi OCR2A/B,cờ OCF2A/B đặt lên 1 bằng phần cứng và xóa về 0 bằng phần mềm.Ở xung CLKT2 kế tiếp,OCR2A chưa giá trị TOP nên khi cờ OCF2A=1,TCNT2 trở về 0.

Ví dụ sau minh họa Timer2 hoạt động cả 2 mô thức NOR và CTC.

**Ví dụ 7.10:** Thiết kế mạch tạo 2 chuỗi xung vuông có dạng như Hình 7.24.



Hình 7.24: Dạng sóng ngõ ra ví dụ 7.10

**Giải:**

Để có dạng sóng ngõ ra như Hình 7.24,ta áp dụng Timer2 chạy luân phiên 2 mô thức CTC và NOR như minh họa ở hình 7.25.

Theo hình 7.25,trong thời đoạn xung ra f0 mức 0,Timer2 chạy mô thức CTC giá trị TOP=OCR2A tương ứng thời đoạn 4ms.Khi đạt kết quả so sánh cờ OCF2A=1,xóa cờ OCF2A=0,đặt f0=1 và chuyển Timer2 chạy mô thức NOR,đồng thời nạp TCNT2 giá trị đặt trước để Timer2 đếm đến tràn trong thời đoạn 2ms.Cho đến khi Timer2 tràn cờ TOV2=1,xóa cờ TOV2,xóa f0=0,chuyển sang mô thức CTC như ban đầu.

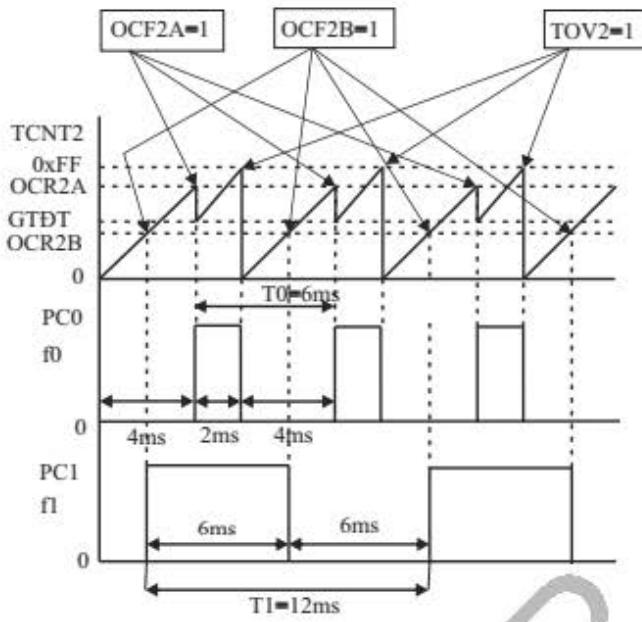
Chọn hệ số chia N=128 tương ứng CS22:CS20=101,thời gian 1 xung CLKT2=128x0.125=16μs.

Tính giá trị đặt trong OCR2A trong thời đoạn 4ms:  $TOP+1=4000/16=250$ ,suy ra  $OCR2A=249$

Cũng với xung CLKT2=16μs,thời đoạn 2ms có số xung đếm tràn là 125,suy ra giá trị đặt trước (GTDT) cho TCNT2=-125#131 xung đếm.Khi Timer2 đếm trong mô thức NOR,vẫn xảy ra kết quả so sánh bằng kênh A,cờ OCF1A=1.Do đó,trong mô thức NOR ta chỉ kiểm tra cờ TOV2 đến khi tràn,và xóa cờ TOV2 cùng với cờ OCF1A trước khi chuyển sang mô thức CTC.

Xung ngõ ra f1 tạo thành từ kết quả so sánh kênh B,đặt trước OCR2B nhỏ hơn OCR2A.Cờ OCF2B=1 chỉ khi chạy mô thức CTC,do trong mô thức NOR giá trị TCNT2 luôn lớn hơn giá trị đặt trong OCR2B.Thực hiện đảo bit ngõ ra khi cờ OCF2B=1 và xóa cờ OCF2B.

Xung ra f1 độ rộng 6ms,thời điểm cờ OCF2B=1 bằng phân nửa thời điểm cờ OCF2A=1,suy ra giá trị đặt  $OCR2B=(TOP+1)/2 - 1=124$ .



Hình 7.25: Timer2 chạy luân phiên mô thức CTC và NOR

❖ Giải thuật chương trình ví dụ 7.10

1. Khởi động nạp OCR2A=249,OCR2B=124, ngõ ra xung f0 PC0=0, ngõ ra xung f1 PC1=0
2. Cho Timer2 chạy mô thức CTC, hệ số chia N=128
3. Kiểm tra nếu cờ OCF2B=1 xóa cờ OCF2B và đảo bit PC1.
4. Nếu cờ OCF2B=0, kiểm tra nếu cờ OCF2A=1 xóa cờ OCF2A, đặt PC0=1, chuyển sang bước 5.  
Nếu cờ OCF2A=0 lặp lại bước 3.
5. Nạp TCNT2=-125, cho Timer2 chạy mô thức NOR, N=128.
6. Lặp vòng kiểm tra cho đến khi cờ TOV2=1 xóa cờ TOV2 và OCF1A, xóa PC0=0, trở về bước 2.

❖ Chương trình hợp ngữ ví dụ 7.10

```

EQU F0=0
EQU F1=1
ORG 0
RJMP MAIN
ORG 0X40
MAIN: LDI R16,HIGH(RAMEND); đưa stack lên vùng đ/c cao
       OUT SPH,R16
       LDI R16,LOW(RAMEND)
       OUT SPL,R16
       LDI R16,(1<<F0)|(1<<F1)
       OUT DDRC,R16      ;khai báo F0,F1 là output
       CBI PORTC,F0        ;F0=0
       CBI PORTC,F1        ;F1=0
       LDI R18,(1<<F1)     ;đặt vị trí bit f1=1
       LDI R16,249          ;giá trị đặt OCR2A
       STS OCR2A,R16
       LDI R16,124          ;giá trị đặt OCR2B
       STS OCR2B,R16
START: LDI R16,0X02        ;Timer2 mode CTC
       STS TCCR2A,R16
       LDI R16,0X05        ;N=128, chạy Timer2 mode CTC
       STS TCCR2B,R16
WAIT0: IN  R17,TIFR2      ;đọc cờ OCF2B=1
       SBRS R17,OCF2B       ;cờ OCF2B=1 xử lý kênh B

```

	RJMP	WAIT1	;kiểm tra cờ OCF2A
	OUT	TIFR2,R17	;xóa cờ OCF2B
	IN	R17,PORTC	;đọc PortC
	EOR	R17,R18	;đảo bit f1
	OUT	PORTC,R17	;xuất ra PortC
WAIT1:	IN	R17,TIFR2	;đọc cờ OCF2A=1
	SBRS	R17,OCF2A	;cờ OCF2A=1 xử lý kênh A
	RJMP	WAIT0	;cờ OCF2A=0 quay về kiểm tra cờ OCF2B
	OUT	TIFR2,R17	;xóa cờ OCF2A
	SBI	PORTC,F0	;đặt f0=1
	LDI	R16,0X00	;chuyển Timer2 mode NOR
	STS	TCCR2A,R16	;
	LDI	R16,-125	;nạp Timer2 giá trị đặt trước
	STS	TCNT2,R16	
WAIT2:	IN	R17,TIFR2	;đọc cờ TOV2
	SBRS	R17,TOV2	;cờ TOV2=1 xử lý tràn
	RJMP	WAIT2	;chờ cờ TOV2=1
	OUT	TIFR2,R17	;xóa cờ TOV2
	CBI	PORTC,F0	;xóa f0=0
	RJMP	START	;lặp vòng lại từ đầu

- Trong chương trình trên, do thời gian 1 xung CLK2=16μs khá dài, nên ta không cần dừng và xóa Timer2 khi chuyển mô thức, và có thể cập nhật ghi tức thời các thanh ghi TCNT2, TCCR2A/B khi Timer2 làm việc ở các mô thức đồng bộ.

➤ **Lưu ý:** *Truy xuất thanh ghi TIFR2 bằng lệnh IN/OUT, các thanh ghi TCNT2, OCR2A/B, TCCR2A/B bằng lệnh LDS/STS!!!*

### ❖ Chương trình C ví dụ 7.10

```
#include <avr/io.h>
#define outport PORTC // định nghĩa PORTC=output
const char f0=0,f1=1 // ký hiệu vị trí bit các ngõ ra
int main() // bắt đầu chương trình
{
    DDRC |=(1<<f0)|(1<<f1); //khai báo f0,f1 output
    outport=0x00 //xóa các ngõ ra
    OCR2A=249 //giá trị đặt OCR2A
    OCR2B=124 //giá trị đặt OCR2B
    while(1) // lặp vòng vô hạn
    {
        TCCR2A=0X02 //Timer2 mode CTC
        TCCR2B=0X05 //hệ số chia N=128, chạy Timer2
        while(!(TIFR2&((1<<OCF2A)(1<<(OCF2B))))) //chờ cờ OCF2A/B=1
        if(TIFR2&(1<<OCF2B)) //cờ OCF2B=1
        {
            TIFR2 |=(1<<OCF2B) //xóa cờ OCF2B
            outport^=(1<<f1) //đảo bit f1
        }
        else if(TIFR2&(1<<OCF2A)) //cờ OCF2A=1
        {
            TIFR2 |=(1<<OCF2A) //xóa cờ OCF2A
            outport|(1<<f0) //đặt bit f0
            TCCR2A=0X00 //Timer2 mode NOR
            TCNT2=-125 //nạp giá trị đặt trước bộ đếm
        }
    }
}
```

```

        while(!(TIFR2&(1<<TOV2)));//chờ cờ TOV2=1
        TIFR2 |=(1<<TOV2)      ;//xóa cờ TOV2
        outport&= ~(1<<f0)      ;//xóa bit f0
    }
}
}

```

### 7.4.3 Hoạt động Timer2 mô thức bắt đồng bộ(xung CLKT2 ngoài)

Khi đặt bit AS2=1,AS2 thuộc thanh ghi ASSR,Timer2 chuyển sang mô thức bắt đồng bộ.Xung CLKT2 lấy từ xung dao động bộ thạch anh gắn vào ngõ TOSC1(PC6) và TOSC2(PC7) hoặc từ nguồn xung dao động ngoài đưa vào ngõ TOSC1(TOSC2 để trống).Bit EXCLK thuộc thanh ghi chọn nguồn xung dao động ngoài.Bit EXCLK=1 chọn nguồn dao động từ bộ thạch anh,bit EXCLK=0 chọn nguồn dao động đưa vào chân TOSC1.Bộ đếm dao động nội được thiết kế làm việc tối ưu ở tần số fXtal=32768Hz!

Bộ chia đặt trước vẫn làm việc khi Timer2 chạy mô thức bắt đồng bộ.

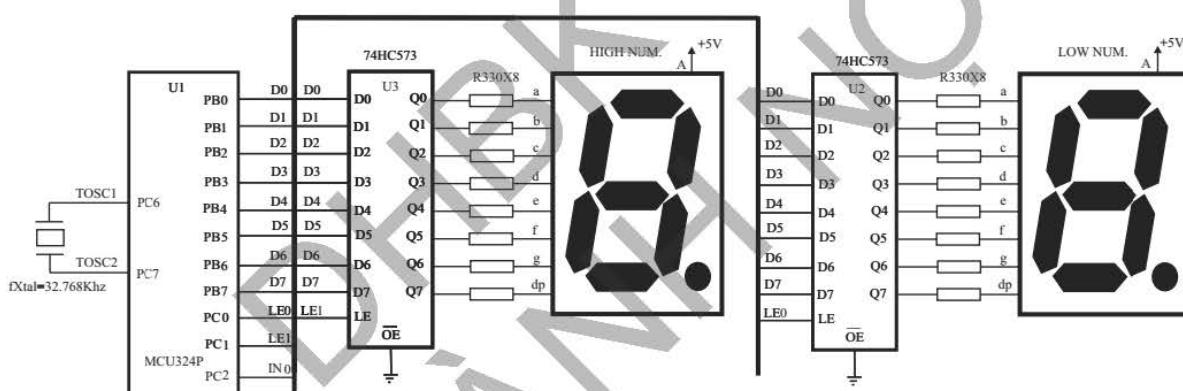
Khi chuyển Timer2 sang mô thức bắt đồng bộ,nội dung các thanh ghi TCNT2,OCR2A/B, TCCR2A/B có thể bị lỗi,nên cần cập nhật lại trước khi chạy Timer2.

➤ **Lưu ý:** Trong mô thức bắt đồng bộ,khi ghi vào bất kỳ 1 trong 5 thanh ghi trên phải kiểm tra cờ báo bận trong ứng thuộc thanh ghi ASSR,cho đến khi cờ báo bận xóa về 0 mới có thể ghi tiếp thanh ghi khác.

**Ví dụ 7.11:** Thiết kế một đồng hồ đếm giây từ 0 - 60s, hiển thị ra 2 LED 7 đoạn AC.

**Giải:**

Về thiết kế phần cứng,ta áp dụng lại sơ đồ ví dụ 6.11.Hình 7.26 là sơ đồ thiết kế ví dụ 7.11.



Hình 7.26: Sơ đồ thiết kế ví dụ 7.11

Ta áp dụng Timer2 chạy mô thức bắt đồng bộ AS2=1,gắn bộ thạch anh fXtal=32768Hz vào 2 chân TOSC1 và TOSC2,EXCLK=0.

Với fXtal=32768Hz=2<sup>15</sup>Hz,ta chọn Timer2 chạy mô thức CTC hệ số chia N=256,CS22:CS20=110, giá trị đặt OCR2A=127.Suy ra thời gian đạt kết quả so sánh OCF2A=1: Td=2<sup>15</sup>/(256x128)=1s.

Sử dụng bộ đếm số lần cờ OCF2A=1,tà được bộ đếm giây.Khi bộ đếm tới 60s,xóa bộ đếm về 0.

### ❖ Chương trình hợp ngữ ví dụ 7.11

.DEF	SEC_CT=R20
.EQU	OUTPORT=PORTB
.EQU	IOSET=DDRB
.ORG	0
RJMP	MAIN
.ORG	0X40
MAIN:	LDI R16,HIGH(RAMEND);đưa stack lên vùng đ/c cao
OUT	SPH,R16
LDI	R16,LOW(RAMEND)
OUT	SPL,R16
LDI	R16,0X03

	OUT	DDRC,R16	;khai báo PC0,PC1 là output
	CBI	PORTC,0	;khóa ngõ ra U2
	CBI	PORTC,1	;khóa ngõ ra U3
	LDI	R16,0XFF	
	OUT	IOSET,R16	;khai báo Port output
	LDI	R16,(1<<AS2)	;Timer2 chạy bắt đồng bộ
	STS	ASSR,R16	
	LDI	R16,127	;nạp giá trị OCR2A
	STS	OCR2A,R16	
WAIT0:	LDS	R16,ASSR	;đọc cờ OCR2AUB
	SBRC	R16,OCR2AUB	;tiếp tục khi cờ OCR2AUB=0
	RJMP	WAIT0	
	LDI	R16,0X02	;nạp giá trị TCCR2A, Timer2 mode CTC
	STS	TCCR2A,R16	
WAIT1:	LDS	R16,ASSR	;đọc cờ TCR2AUB
	SBRC	R16,TCR2AUB	;tiếp tục khi cờ TCR2AUB=0
	RJMP	WAIT1	
	LDI	R16,0X06	;nạp giá trị TCCR2B, N=256
	STS	TCCR2B,R16	
WAIT2:	LDS	R16,ASSR	;đọc cờ TCR2BUB
	SBRC	R16,TCR2BUB	;tiếp tục khi cờ TCR2BUB=0
	RJMP	WAIT2	
	CLR	SEC_CT	;xóa bộ đếm s
START:	MOV	R17,SEC_CT	;chuyển vào R17
	RCALL	BIN8_BCD2	;ctc chuyển số nhị phân thành BCD 2 digit
	RCALL	DISP_SEC	;ctc hiển thị s
	RCALL	DELAY_T2	;ctc đếm s
	RJMP	START	
;			
DELAY_T2:	IN	R17,TIFR2	;đọc cờ OCF2A
	SBRS	R17,OCF2A	;tiếp tục khi cờ OCF2A=1
	RJMP	DELAY_T2	
	OUT	TIFR2,R17	;xóa cờ OCF2A
	INC	SEC_CT	;bộ đếm s tăng 1
	CPI	SEC_CT,60	;tới 60s?
	BRNE	EXIT_DL	;chưa thoát
	CLR	SEC_CT	;xóa bộ đếm s
EXIT_DL:	RET		
;			
	;BIN8_BCD2 chuyển số nhị phân 8 bit sang số BCD 2 digit		
	;Input R17=số nhị phân 8 bit		
	;Output R17 số BCD nén		
	;Sử dụng ctc DIV8_8,R16=10 số chia		
;			
BIN8_BCD2:	LDI	R16,10	;R16=số chia
	RCALL	DIV8_8	;ctc chia 2 số nhị phân 8 bit
	SWAP	R17	;chuyển thương số=dư số sau cùng lên 4 bit cao
	OR	R17,R16	;dán dư số phép chia lần 1 vào 4 bit thấp
	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=đư số
          MOV R17,R15  ;R17=thương số
          RET
;
```

```

DISP_SEC: PUSH R17    ;cất data
          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
          SWAP R17        ;hoán vị sang 4 bit thấp
          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
          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
;
```

```

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

- Trong mô thức bắt đồng bộ, AS2=1, mỗi lần ghi vào các thanh ghi OCR2A, TCCR2A/B, ta phải kiểm tra các cờ báo bận OCR2AUB, TCR2AUB, TCR2BUB tương ứng, cho đến khi các cờ này xóa về 0 mới ghi vào thanh ghi khác.

- Chương trình con DEAY\_T2 chờ cờ OCF2A=1, xóa cờ OCF2A, tăng 1 bộ đếm giây(s) cho đến 60s xóa bộ đếm giây(s) về 0.

- Các chương trình con còn lại áp dụng từ ví dụ 6.11.

### ❖ Chương trình C ví dụ 7.11

```

#include <avr/io.h>
#define ioSet DDRB      // định nghĩa ioSet=DDRB
#define outport PORTB   //định nghĩa outport=PORTB
#define cont PORTC      //định nghĩa cont=PORTC
const char le0=0,le1=1 ;//ký hiệu ngõ chốt U2,U3
void delay_T2()       ;//khai báo hàm đếm giây
;
```

```

void bin8_bcd2()          ;//khai báo hàm chuyển số nhị phân 8 bit sang BCD 2 digit
void disp_sec()           ;//khai báo hàm hiển thị giây
unsigned char get_7seg(unsigned char x) ;//khai báo hàm chuyển mã 7 đoạn AC
unsigned char sec_ct,tam,i;
int main()
{
    DDRC=0x03           ;//định nghĩa PC0,PC1 là output
    cont&=~((1<<le0)|(1<<le1)) ;//khóa U2,U3
    ioset=0xff           ;//PortB output
    ASSR|=(1<<AS2)      ;//Timer2 mode bắt đồng bộ
    OCR2A=127            ;//giá trị đặt OCR2A
    while(ASSR&(1<<OCR2AUB)) ;//chờ cờ OCR2AUB=0
    TCCR2A=0x02           ;//Timer2 mode CTC
    while(ASSR&(1<<TCR2AUB)) ;//chờ cờ TCR2AUB=0
    TCCR2B=0x06            ;//hệ số chia N=256
    while(ASSR&(1<<TCR2BUB)) ;//chờ cờ TCR2BUB=0
    sec_ct=0               ;//xóa bộ đếm giây
    while(1)
    {
        tam=sec_ct ;//lấy giá trị đếm
        bin8_bcd2() ;//chuyển sang BCD
        disp_sec() ;//hiển thị giây
        delay_T2() ;//đếm giây
    }
}
void delay_T2()
{
    while(!(TIFR2&(1<<OCF2A))) ;//chờ cờ OCF2A=1
    TIFR2|=(1<<OCF2A) ;//xóa cờ OCF2A
    sec_ct++             ;//tăng 1 bộ đếm giây
    if(sec_ct==60)
        sec_ct=0          ;//xóa bộ đếm giây khi tới 60s
}
void bin8_bcd2()
{
    unsigned char x;
    x=tam%10             ;//x= dư số tam/10
    tam=(tam/10)<<4       ;//thượng số tam/10 dịch lên 4 bit cao
    tam=tam | x           ;//dán dư số vào 4 bit thấp
}
void disp_sec()
{
    i=tam&0x0f;//che 4 bit cao
    outport=get_7seg(i) ;//chuyển sang mã ASCII và xuất ra U2
    cont|=(1<<le0)      ;//mở U2
    cont&=~(1<<le0)      ;//khóa U2
    i=tam>>4             ;//dịch data sang 4 bit thấp
    outport=get_7seg(i) ;//chuyển sang mã ASCII và xuất ra U3
    cont|=(1<<le1)      ;//mở U3
    cont&=~(1<<le1)      ;//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)      ;
}

```

## ❖ Câu hỏi ôn tập

- Cho biết về chức năng Timer2 khác Timer0 ở điểm nào?
- Xem đoạn lệnh sau và cho biết Timer2 chạy mô thức nào, thời gian 1 xung đếm dài bao lâu:

```

...
LDI    R20,$02
STS    TCCR2A,R20
LDI    R20,$05
STS    TCCR2B,R20

```

- Cho Timer2 chạy mô thức CTC, đặt OCR1A=0xFF. Khi nào cờ TOV2=1?
- Trong ví dụ 7.11, tại sao mỗi lần ghi vào các thanh ghi thuộc Timer2 phải đọc thanh ghi ASSR?
- Xem đoạn lệnh sau và cho biết nguồn xung đếm Timer lấy từ đâu:

```

...
LDI    R17,0XC0
STS    ASSR,R17

```

## 7.5 Lập trình hoạt động Timer1

### 7.5.1 Các thanh ghi Timer1

Timer1 là bộ Timer 16 bit, các thanh ghi TCNT1, OCR1A, OCR1B, ICR1 độ dài 16 bit, chia thành 2 thanh ghi 8 bit TCNT1H, TCNT1L, OCR1AH, OCR1AL, OCR1BH, OCR1BL, ICR1H, ICR1L tương ứng. Ta sẽ tìm hiểu chi tiết các thanh ghi Timer1.

#### 1. TCNT1H, TCNT1L(Timer/Counter1)

Bit	7	6	5	4	3	2	1	0	
(0x85)	TCNT1[15:8]								TCNT1H
(0x84)	TCNT1[7:0]								TCNT1L
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

Hình 7.27: Thanh ghi TCNT1H, TCNT1L

Thanh ghi TCNT1 gồm 2 thanh ghi 8 bit TCNT1H (byte cao) và TCNT1L (byte thấp) là bộ đếm 16 bit, địa chỉ MEM=0x85, 0x84 tương ứng, có thể đọc ghi, không truy xuất bit được.

Mỗi lần có xung đếm CLKT1, nội dung thanh ghi TCNT1 tăng 1 từ byte thấp. Giá trị MAX của TCNT1 bằng 0xFFFF. Xung đếm tiếp làm TCNT1 tràn và trở về 0x0000.

Để đảm bảo việc ghi/đọc byte cao và byte thấp đồng thời trong quá trình CPU truy xuất các thanh ghi 16 bit, byte cao không được truy xuất trực tiếp mà chỉ truy xuất thông qua thanh ghi đệm 8 bit TEMP (xem chi tiết phần sau).

#### 2. Thanh ghi TCCR1A(Timer/Counter Control Register A)

Bit	7	6	5	4	3	2	1	0	
(0x80)	COM1A1	COM1A0	COM1B1	COM1B0	-	-	WGM11	WGM10	TCCR1A
Read/Write	R/W	R/W	R/W	R/W	R	R	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

Hình 7.28: Thanh ghi TCCR1A

Thanh ghi TCCR1A địa chỉ MEM=0x80, có thể đọc ghi và truy xuất bit được, cài đặt logic các ngõ ra OC1A, OC1B trong các mô thức tạo sóng và kết hợp với thanh ghi TCCR1B cài đặt mô thức hoạt động cho Timer1.

- **Bit 7:6 COM1A1:COM1A0 (Compare Match Output A Mode)**

Hai bit này cài đặt trạng thái logic ngõ ra OC1A trong các mô thức tạo sóng (sẽ trình bày chi tiết trong phần tạo sóng)

- **Bit 5:4 COM1B1:COM1B0 (Compare Match Output B Mode)**

Hai bit này cài đặt trạng thái logic ngõ ra OC1B trong các mô thức tạo sóng(sẽ trình bày chi tiết trong phần tạo sóng)

- Bit 2:1 WGM11:WGM10 (Waveform Generation Mode)**

Hai bit này kết hợp với bit WGM13,WGM12 thuộc thanh ghi TCCR1B cài đặt các mô thức hoạt động của Timer1,sẽ trình bày trong mô tả thanh ghi TCCR1B.

### 3. Thanh ghi TCCR1B(Timer/Counter Control Register B)

Bit (0x81)	7	6	5	4	3	2	1	0	TCCR1B
Read/Write	ICNC1	ICES1	-	WGM13	WGM12	CS12	CS11	CS10	
Initial Value	R/W	R/W	R	R/W	R/W	R/W	R/W	R/W	

Hình 7.29: Thanh ghi TCCR1B

Thanh ghi TCCR1B có địa chỉ MEM=0x81,đọc/ghi và truy xuất bit được cài đặt tần số xung CLK1 cho bộ đếm,kết hợp với thanh ghi TCCR1A cài đặt các mô thức hoạt động của Timer1,cài đặt dạng xung bắt ngõ vào(input capture)Timer1.

- Bit 7 ICNC1(Input Capture Noise Canceler): Triệt nhiễu bắt ngõ vào**

Bit này có chức năng lọc nhiễu tín hiệu bắt ngõ vào ICP(PD6).Khi ghi ICNC1=1, CPU sẽ lấy mẫu tín hiệu bắt ngõ vào ICP khi chuyển biến, liên tục 4 lần để xác định mới chấp nhận có tác động bắt ngõ vào.Do đó việc bắt ngõ vào có triệt nhiễu phải mất 4 chu kỳ dao động!

- Bit 6 ICES1(Input Capture Edge Select): Chọn cạnh xung bắt ngõ vào**

Bit này lựa chọn cạnh xung kích khởi bắt ngõ vào ở chân ICP(PC6).Khi ghi ICES1=1,xung kích cạnh lên,khi ghi ICES1=0,xung kích cạnh xuống.Khi có cạnh xung kích ở ngõ vào ICP,nội dung bộ đếm TCNT1 được chép vào thanh ghi ICR1,đồng thời phần cứng đặt cờ ICF1=1 chỉ báo dữ liệu đã được chép từ bộ đếm sang ICR1.Cờ ICF1=1 cũng tạo ngắt bắt ngõ vào tương ứng,nếu ngắt này được cho phép,và phần cứng tự động xóa cờ ICF1 khi CPU chuyển đến vector ngắt tương ứng(xem chi tiết trong chương 10).Hoặc ta phải xóa cờ ICF1 bằng phần mềm bằng cách ghi bit 1 vào vị trí cờ này.

- Bit 4,3 WGM13,WGM12(Waveform Generation Mode)**

Bit WGM13,WGM12 kết hợp với bit WGM11,WGM10 thuộc TCCR1A cài đặt các mô thức hoạt động của Timer1 như Bảng 7.8.

Bảng 7.8: Các mô thức hoạt động của Timer1

MODE	WGM13	WGM12	WGM11	WGM10	MÔ THỨC HOẠT ĐỘNG	TOP	Thời điểm cập nhật OCR1x	TOV1=1 ở điểm
0	0	0	0	0	Normal	0xFFFF	Tức thời	MAX
1	0	0	0	1	PWM, Phase Correct, 8-bit	0x00FF	TOP	BOTTOM
2	0	0	1	0	PWM, Phase Correct, 9-bit	0x01FF	TOP	BOTTOM
3	0	0	1	1	PWM, Phase Correct, 10-bit	0x03FF	TOP	BOTTOM
4	0	1	0	0	CTC	OCR1A	Tức thời	MAX
5	0	1	0	1	Fast PWM, 8-bit	0x00FF	BOTTOM	TOP
6	0	1	1	0	Fast PWM, 9-bit	0x01FF	BOTTOM	TOP
7	0	1	1	1	Fast PWM, 10-bit	0x03FF	BOTTOM	TOP
8	1	0	0	0	PWM, Phase and Frequency Correct	ICR1	BOTTOM	BOTTOM
9	1	0	0	1	PWM, Phase and Frequency Correct	OCR1A	BOTTOM	BOTTOM
10	1	0	1	0	PWM, Phase Correct	ICR1	TOP	BOTTOM
11	1	0	1	1	PWM, Phase Correct	OCR1A	TOP	BOTTOM
12	1	1	0	0	CTC	ICR1	Tức thời	MAX
13	1	1	0	1	Dự trữ	-	-	-
14	1	1	1	0	Fast PWM	ICR1	BOTTOM	TOP
15	1	1	1	1	Fast PWM	OCR1A	BOTTOM	TOP

- Có 15 mô thức hoạt động và mô thức 13 dự trữ: mô thức 0 NOR,mô thức CTC có 2 lựa chọn: mô thức CTC4 giá trị TOP bộ đếm đặt trong thanh ghi OCR1A,mô thức CTC12 giá trị TOP bộ đếm đặt trong thanh ghi ICR1.Các mô thức còn lại sẽ trình bày trong phần tạo sóng.

- Cột TOP thể hiện giá trị định bộ đếm TCNT1 đếm tới
- Cột thời điểm OCR1x cập nhật thể hiện thời điểm tại đó các thanh ghi OCR1A/OCR1B cập nhật giá trị mới
- Cột TOV1 =1 ở điểm thể hiện thời điểm tại đó cờ TOV1 được đặt lên 1.

Ví dụ trong mô thức CTC4WGM13: WGM10=100, TOP=giá trị đặt trong OCR1A,OCR1A/OCR1B có thể cập nhật giá trị mới tức thời,cờ TOV1 =1 khi Timer1 đếm tới MAX=0xFFFF.Hay nói cách khác,trường hợp này phải đặt OCR1A=0xFFFF mới có thể có cờ TOV1=1!

- **Bit 2:1:0 CS02:CS01:CS00(Clock Select)**

Ba bit này cài đặt nguồn tạo xung CK cho bộ đếm Timer1 như mô tả trong Bảng 7.9

**Bảng 7.9:** Lựa chọn nguồn tạo xung CK Timer1

CS02	CS01	CS00	Tần số xung CLKT0
0	0	0	Không có xung CK(Timer0 dừng)
0	0	1	CLKIO(N=1:không qua bộ chia đặt trước)
0	1	0	CLKIO/8(N=8:từ bộ chia đặt trước)
0	1	1	CLKIO/64(N=64:từ bộ chia đặt trước)
1	0	0	CLKIO/256(N=256:từ bộ chia đặt trước)
1	0	1	CLKIO/1024(N=1024:từ bộ chia đặt trước)
1	1	0	Nguồn CK ngoài từ chân T1,tác động cạnh xuống
1	1	1	Nguồn CK ngoài từ chân T1,tác động cạnh lên

- *Lưu ý: Khi khai báo CS02:CS00=110,111, chân T1=PB1 tự động chuyển thành ngõ vào nhận xung đếm từ bên ngoài,cho dù trước đó ta không khai báo PB1 là ngõ vào!*

#### 4. Thanh ghi TCCR1C(Timer/Counter Control Register C): Thanh ghi điều khiển Timer C

Bit (0x82)	7	6	5	4	3	2	1	0	
Read/Write	FOC1A	FOC1B	-	-	-	-	-	-	TCCR1C
Initial Value	R/W	R/W	R	R	R	R	R	R	

**Hình 7.31:** Thanh ghi TCCR1C

Thanh ghi TCCR1C địa chỉ MEM=0x82, đọc/ghi và truy xuất bit được.Thanh ghi này chứa 2 bit điều khiển ngõ ra phát xung OC1A/B của Timer1.

- **Bit 7 FOC1A(Force Output Compare A): Bit ép so sánh ngõ ra A**

Bit này chỉ tác động khi Timer1 làm việc ở mô thức NOR,CTC.Khi ghi FOC1A=1,sẽ ép kết quả so sánh xảy ra(tương đương như giá trị bộ đếm bằng OCR1A)tác động lên khối tạo sóng.ngõ ra OC1A sẽ thay đổi logic theo cài đặt COM1A1:COM1A0.

- **Bit 6 FOC1B(Force Output Compare B): Bit ép so sánh ngõ ra B**

Bit này chỉ tác động khi Timer1 làm việc ở mô thức NOR,CTC.Khi ghi FOC1B=1,sẽ ép kết quả so sánh xảy ra(tương đương như giá trị bộ đếm bằng OCR1B)tác động lên khối tạo sóng.ngõ ra OC1B sẽ thay đổi logic theo cài đặt COM1B1:COM1B0.

- *Lưu ý:* - Trong các mô thức PWM,khi ghi vào TCCR1B nên xóa FOC1A,FOC1B để tương hợp với các chip trong tương lai.  
- Khi đọc các bit FOC1A,FOC1B sẽ cho giá trị 0.

#### 5. Thanh ghi OCR1AH,OCR1AL(Output Compare Register 1 A)

Bit (0x89)	7	6	5	4	3	2	1	0	
(0x88)	OCR1A[15:8]								OCR1AH
Initial Value	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	OCR1AL
Read/Write	0	0	0	0	0	0	0	0	

**Hình 7.32:** Thanh ghi OCR1AH,OCR1AL

Thanh ghi OCR1A 16 bit gồm 2 thanh ghi 8 bit OCR1AH(byte cao) và OCR1AL(byte thấp),địa chỉ MEM=0x89,0x88 tương ứng,có thể đọc ghi,không truy xuất bit được.Thanh ghi OCR1A chứa giá trị tham chiếu để bộ đếm TCNT1 so sánh trong mô thức CTC và các mô thức PWM kênh A.

Để đảm bảo việc ghi/đọc byte cao và byte thấp đồng thời trong quá trình CPU truy xuất các thanh ghi 16 bit, byte cao không được truy xuất trực tiếp mà chỉ truy xuất thông qua thanh ghi đệm 8 bit TEMP(xem chi tiết phần sau).

➤ *Lưu ý: Trong mô thức CTC4 và PWM, giá trị đặt trong OCR1A chính là giá trị TOP của bộ đếm Timer1.*

## 6. Thanh ghi OCR1BH,OCR1BL(Output Compare Register 1 B)

Bit	7	6	5	4	3	2	1	0	
(0x8B)	OCR1B[15:8]								OCR1BH
(0x8A)	OCR1B[7:0]								OCR1BL
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

Hình 7.33: Thanh ghi OCR1BH,OCR1BL

Thanh ghi OCR1B 16 bit gồm 2 thanh ghi 8 bit OCR1BH(byte cao) và OCR1BL(byte thấp), địa chỉ MEM=0x8B,0x8A tương ứng,có thể đọc ghi,không truy xuất bit được.Thanh ghi OCR1B chứa giá trị tham chiếu để bộ đếm TCNT1 so sánh trong mô thức CTC và các mô thức PWM kênh B.

Để đảm bảo việc ghi/đọc byte cao và byte thấp đồng thời trong quá trình CPU truy xuất các thanh ghi 16 bit, byte cao không được truy xuất trực tiếp mà chỉ truy xuất thông qua thanh ghi đệm 8 bit TEMP(xem chi tiết phần sau).

➤ *Lưu ý: Trong mô thức CTC4 và các mô thức PWM,do TOP=giá trị đặt trong OCR1A,nên giá trị đặt trong OCR1B không lớn hơn OCR1A ,việc so sánh ở kênh B mới thực hiện được!*

## 7. Thanh ghi ICR1BH,ICR1BL(Input Capture Register 1 ): Thanh ghi bắt ngõ vào Timer1

Bit	7	6	5	4	3	2	1	0	
(0x87)	ICR1[15:8]								ICR1H
(0x86)	ICR1[7:0]								ICR1L
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

Hình 7.34: Thanh ghi ICR1H,ICR1L

Thanh ghi ICR1 16 bit gồm 2 thanh ghi 8 bit ICR1H(byte cao) và ICR1L(byte thấp), địa chỉ MEM=0x87,0x86 tương ứng,có thể đọc ghi,không truy xuất bit được.Tại mỗi thời điểm có tín hiệu bắt ngõ vào kích khởi từ chân ICP(PD6) hoặc ngõ ra bộ so sánh tương tự,nội dung bộ đếm TCNT1 được chép vào thanh ghi ICR1.Thanh ghi ICR1 còn chứa giá trị TOP trong mô thức CTC12 và các mô thức PWM.Để đảm bảo việc ghi/đọc byte cao và byte thấp đồng thời trong quá trình CPU truy xuất các thanh ghi 16 bit, byte cao không được truy xuất trực tiếp mà chỉ truy xuất thông qua thanh ghi đệm 8 bit TEMP(xem chi tiết phần sau).

## 8. Thanh ghi TIFR1(Timer/Counter 1 Interrupt Flag Register)

Bit	7	6	5	4	3	2	1	0	
0x16 (0x36)	-	-	ICF1	-	-	OCF1B	OCF1A	TOV1	TIFR1
Read/Write	R	R	R/W	R	R	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

Hình 7.35: Thanh ghi TIFR1

Thanh ghi TIFR1 có địa chỉ I/O=0x16, đọc ghi và truy xuất bit được.Thanh ghi TIFR1 chứa 4 cờ báo cho các mô thức tương ứng.

- **Bit 5 ICF1(Timer/Counter 1 Input Capture Flag): Cờ báo bắt ngõ vào Timer1**

Khi sự kiện bắt ngõ vào xảy ra, phần cứng đặt cờ ICF1=1 chỉ báo nội dung bộ đếm đã chép qua thanh ghi ICR1.Trường hợp thanh ghi ICR1 chứa giá trị TOP,cờ ICF1=1 khi bộ đếm TCNT1 đạt đến giá trị TOP.Cờ ICF1 được xóa bằng phần cứng khi MCU chuyển điều khiển đến vector ngắt nếu có khai báo cho phép ngắt từ nguồn ngắt tác động bởi cờ này(xem chi tiết chương 10).Hoặc ta phải xóa cờ ICF1 bằng cách ghi 1 vào bit này.

- **Bit 2 OCF1B: Timer/Counter 1 Output Compare B Match Flag**

Bit cờ OCF1B được đặt lên 1 bằng phần cứng khi bộ đếm TCNT1 có giá trị bằng giá trị đặt trong thanh ghi OCR1B.Cờ OCF1B được xóa bằng phần cứng khi MCU chuyển điều khiển đến vector ngắt nếu có khai báo cho phép ngắt từ nguồn ngắt tác động bởi cờ này(xem chi tiết chương 10).Hoặc ta phải xóa cờ OCF1B bằng cách ghi 1 vào bit này.

- Bit 1 OCF1A: Timer/Counter 1 Output Compare A Match Flag**

Bit cờ OCF1A được đặt lên 1 bằng phần cứng khi bộ đếm TCNT1 có giá trị bằng giá trị đặt trong thanh ghi OCR1A.Cờ OCF1A được xóa bằng phần cứng khi MCU chuyển điều khiển đến vector ngắt nếu có khai báo cho phép ngắt từ nguồn ngắt tác động bởi cờ này(xem chi tiết chương 10).Hoặc ta phải xóa cờ OCF1A bằng cách ghi 1 vào bit này.

- Bit 0 TOV1: Timer/Counter 1 Over Flow Flag**

Bit cờ TOV1 được đặt lên 1 bằng phần cứng khi bộ đếm TCNT1 tràn từ 0xFFFF sang 0x0000. trong các mô thức NOR,CTC,ngoài ra còn tùy thuộc vào các mô thức khác như mô tả trong Bảng 7.8.Cờ TOV1 được xóa bằng phần cứng khi MCU chuyển điều khiển đến vector ngắt nếu có khai báo cho phép ngắt từ nguồn ngắt tác động bởi cờ này(xem chi tiết chương 10).Hoặc ta phải xóa cờ TOV1 bằng cách ghi 1 vào bit này.

## 9. Thanh ghi TIMSK1: Timer/Counter 1 Interrupt Mask Register

Bit (0x6F)	7	6	5	4	3	2	1	0	TIMSK1
Read/Write	-	-	R/W	R	R	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

**Hình 7.36:** Thanh ghi TIMSK1

Thanh ghi TIMSK1 có địa chỉ MEM=0x6F, đọc ghi và truy xuất bit được.Thanh ghi TIMSK1 chứa 4 bit cầm/cho phép các nguồn ngắt tràn Timer1,kết quả so sánh ngõ ra và bắt ngõ vào.

- Bit 5 ICIE1(Timer/Counter 1 Input Capture Interrupt Enabe): Bit cho phép ngắt bắt ngõ vào Timer1**

Khi đặt bit ICIE1=1, đồng thời bit I thuộc thanh ghi SREG cũng đặt lên 1,ngắt bắt ngõ vào Timer1(Input Capture Interrupt)được cho phép.Khi có tín hiệu bắt ngõ vào ICR1 được chép giá trị TCNT1,hoặc mô thức Timer CTC12 bộ đếm đạt đến giá trị TOP=ICR1,cờ ICF1 được phần cứng đặt lên 1,chính là tín hiệu báo ngắt bắt ngõ vào.MCU chuyển điều khiển đến vector ngắt bắt ngõ vào và phần cứng xóa cờ ICF1.

- Bit 2 OCIE1B: Timer/Counter 1 Output Compare Match B Interrupt Enabe**

Khi đặt bit OCIE1B=1, đồng thời bit I thuộc thanh ghi SREG cũng đặt lên 1,ngắt kết quả so sánh ngõ ra B Timer1(COMP1B Interrupt)được cho phép.Khi bộ đếm TCNT1 đếm đến giá trị bằng giá trị đặt trong thanh ghi OCR1B,cờ OCF1B được phần cứng đặt lên 1,chính là tín hiệu báo ngắt COMP1B.MCU chuyển điều khiển đến vector ngắt COMP1B và phần cứng xóa cờ OCF1B.

- Bit 1 OCIE1A: Timer/Counter 1 Output Compare Match A Interrupt Enabe**

Khi đặt bit OCIE1A=1, đồng thời bit I thuộc thanh ghi SREG cũng đặt lên 1,ngắt kết quả so sánh ngõ ra A Timer1(COMP1A Interrupt)được cho phép.Khi bộ đếm TCNT1 đếm đến giá trị bằng giá trị đặt trong thanh ghi OCR1A,cờ OCF1A được phần cứng đặt lên 1,chính là tín hiệu báo ngắt COMP1A.MCU chuyển điều khiển đến vector ngắt COMP1A và phần cứng xóa cờ OCF1A.

- Bit 0 TOIE1: Timer/Counter 1 Over Flow Interrupt Enabe**

Khi đặt bit TOIE1=1, đồng thời bit I thuộc thanh ghi SREG cũng đặt lên 1,ngắt Timer1 tràn (TOV1 Interrupt)được cho phép.Khi bộ đếm TCNT1 tràn từ 0xFFFF sang 0x0000 trong các mô thức NOR,CTC,hoặc tùy thuộc các mô thức khác như mô tả trong Bảng 7.8,cờ TOV1 được phần cứng đặt lên 1,chính là tín hiệu báo ngắt TOV1.MCU chuyển điều khiển đến vector ngắt TOV1 và phần cứng xóa cờ TOV1.

Các trường hợp ngắt Timer1 sẽ được trình bày chi tiết ở chương 10.

### ❖ Câu hỏi ôn tập

1. Ngoại trừ độ dài thanh ghi 16 bit,Timer1 có các thanh ghi nào khác hoặc thêm bớt chức năng so với Timer0?

2. Timer1 có bao nhiêu mô thức CTC, mô tả chi tiết cài đặt các bit WGM13:WGM10 và giá trị TOP tương ứng.
3. Cho biết các chức năng thanh ghi ICR1.
4. Cờ ICF1=1 khi nào?
5. Tín hiệu bắt ngõ vào (input capture) lấy từ đâu? Làm sao để tín hiệu bắt ngõ vào tác động cạnh xuông?

### 7.5.2 Hoạt động Timer1 mô thức bình thường(NOR: Normal)

Timer1 hoạt động mô thức NOR tương tự như Timer0.

Trong mô thức NOR, cài đặt WGM13:WGM10=000, cài đặt CS02:CS00 chọn tần số xung CLKT1. Mỗi lần có 1 xung CLKT1, Timer1 đếm lên 1 đơn vị cho đến giá trị MAX=0xFFFF. Xung đếm kế tiếp làm Timer1 tràn và bộ đếm trở về BOTTOM=0x0000, đồng thời cờ TOV1 đặt lên 1 bằng phần cứng.

**Ví dụ 7.12:** Sử dụng Timer1 tạo chuỗi xung vuông đối xứng độ rộng 1s xuất ra PC0.

**Giải:**

Ta chọn mô thức NOR, WGM13:WGM10=000.

Lập bảng tính số xung đếm n theo hệ số chia N của bộ chia đặt trước như Bảng 7.10

**Bảng 7.10:** Số xung đếm n theo hệ số chia N thời gian 1s

N	T <sub>CLKT1(μs)</sub>	Số xung đếm n=1s/T <sub>CLKT1</sub>
1	0.125	8,000,000
8	1	1,000,000
64	8	125,000
256	32	31,250
1024	128	976.6

Theo bảng 7.10, ta chọn N=256, giá trị đặt trước cho bộ đếm Timer1=-31250.

#### ❖ Chương trình hợp ngữ ví dụ 7.12

```

.EQU P_OUT=0
.EQU TF=-31250 ;TF=giá trị đặt trước
.ORG 0
RJMP MAIN
.ORG 0X40
MAIN: LDI R16,HIGH(RAMEND) ;đưa stack lên đỉnh SRAM
      OUT SPH,R16
      LDI R16,LOW(RAMEND)
      OUT SPL,R16
      LDI R16,(1<<P_OUT) ;đặt P_OUT output
      OUT DDRC,R16
      LDI R17,0X00 ;Timer1 mode NOR
      STS TCCR1A,R17
      LDI R17,0X00 ;Timer1 mode NOR, dừng
      STS TCCR1B,R17
START: RCALL DELAY_1S ;delay tạo độ rộng xung 1s
       IN R17,PORTC ;đọc PortC
       EOR R17,R16 ;đảo bit P_OUT
       OUT PORTC,R17 ;xuất ra PortC
       RJMP START ;lặp vòng lại
-----
DELAY_1S: LDI R17,HIGH(TF) ;nạp TCNT1H
          STS TCNT1H,R17
          LDI R17,LOW(TF) ;nạp TCNT1L
          STS TCNT1L,R17
          LDI R17,0X04 ;Timer2 chạy, hệ số chia N=256

```

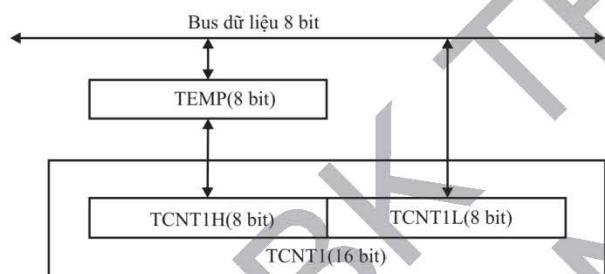
STS	TCCR1B,R17	
WAIT:	IN R17,TIFR1	;đọc thanh ghi cờ Timer1
	SBRS R17,TOV1	;chờ cờ TOV1=1 báo Timer1 tràn
	RJMP WAIT	;cờ TOV1=0 tiếp tục chờ
	OUT TIFR1,R17	;nạp lại bit TOV1=1 xóa cờ TOV1
	LDI R17,0X00	;dừng Timer1
	STS TCCR1B,R17	
	RET	

#### ❖ Vấn đề truy xuất thanh ghi 16 bit

Trong Timer1 có các thanh ghi 16 bit như TCNT1,ICR1,OCR1A/B được phân thành 2 thanh ghi 8 bit để phù hợp với việc truy xuất trên bus dữ liệu 8 bit. Do đó để truy xuất thanh ghi 16 bit phải thực hiện truy xuất 2 lần. Ví dụ sau cho ta thấy việc truy xuất kiểu này có thể gây sai dữ liệu.

Giả sử nội dung bộ đếm TCNT1=\$12FF. Ta đọc byte thấp trước cát vào R20, nội dung R20 là \$FF. Tiếp theo ta đọc byte cao cát vào R21, đồng thời điểm này bộ đếm tăng 1 nên byte cao cát vào R21 bằng \$13. Như vậy nội dung đọc bộ đếm trong R21:R20 bằng \$13FF, thay vì đúng là \$1300!

Để giải quyết vấn đề này, AVR có thanh ghi bộ đếm byte cao ký hiệu là TEMP. Việc truy xuất byte cao không thực hiện trực tiếp vào thanh ghi như byte thấp mà phải thông qua thanh ghi TEMP. Hình 7.37 minh họa việc truy xuất bộ đếm TCNT1 16 bit qua thanh ghi TEMP.



**Hình 7.37:** Truy xuất bộ đếm TCNT1 16 bit byte cao qua thanh ghi TEMP

Khi ghi byte cao TCNT1H, MCU sẽ ghi vào TEMP trước, chờ đến khi ghi byte thấp TCNT1L, MCU đồng thời thực hiện việc ghi dữ liệu byte thấp vào TCNT1L và dữ liệu từ TEMP vào TCNT1H. Như vậy khi ghi dữ liệu 16 bit vào TCNT1, phải ghi byte cao trước, sau đó ghi byte thấp sau.

Ta xem lại đoạn chương trình ví dụ 7.12, để nạp giá trị TF vào TCNT1 ta thực hiện theo trình tự:

LDI	R17,HIGH(TF)	;nạp byte cao TF
STS	TCNT1H,R17	;byte cao TF cát vào TEMP
LDI	R17,LOW(TF)	;nạp byte thấp TF
STS	TCNT1L,R17	;byte thấp TF cát vào TCNT1L, chuyển TEMP vào TCNT1H

Khi đọc bộ đếm TCNT1 thì ngược lại, ta đọc dữ liệu byte thấp vào TCNT1L trước, đồng thời điểm này dữ liệu byte cao cũng được đọc vào TEMP. Khi đọc byte cao vào TCNT1H, dữ liệu byte cao được chuyển từ TEMP sang TCNT1H.

Theo ví dụ ở trên, ta phải thực hiện đọc bộ đếm theo trình tự sau, TCNT1=\$12FF:

LDS	R20,TCNT1L	;R20=TCNT1L=\$FF, TEMP=TCNT1H=\$12
LDS	R21,TCNT1H	;R21=TEMP=\$12

Việc truy xuất các thanh ghi ICR1, OCR1A/B cũng thực hiện giống như trên.

#### ❖ Câu hỏi ôn tập

- Trong mô thức NOR, nếu muốn tạo thời gian delay 1.5ms sử dụng Timer1, giá trị đặt trước bộ đếm và hệ số chia N bằng bao nhiêu cho kết quả chính xác nhất?
- Viết đoạn lệnh khởi động Timer1 theo nhu cầu 1.
- Xem đoạn lệnh sau và cho biết sau bao lâu cờ TOV1=1:

```

...
LDI   R17,$F6
STS   TCNT1H,R17
LDI   R17,$3C
STS   TCNT1L,R17

```

LDI	R17,0
STS	TCCR1A,R17
LDI	R17,0X04
STS	TCCR1B,R17

4. Cho  $F_{osc}=1\text{Mhz}$ , không sử dụng bộ đếm phụ, Timer1 có thể tạo trễ dài nhất bao lâu?  
 5. Giả sử Timer1 đang chạy,  $CLKT1=0.125\mu\text{s}$ . Cho biết nội dung R21:R20 sau khi thực hiện đoạn lệnh như sau:

...	
LDI	R18,\$34
STS	TCNT1H,R18
LDI	R18,\$FE
STS	TCNT1L,R18
NOP	
NOP	
LDS	R21,TCNT1H
LDS	R20,TCNT1L

### 7.5.3 Hoạt động Timer1 mô thức xóa Timer theo kết quả so sánh (CTC: Clear Timer on Compare Match)

Về cơ bản, Timer1 hoạt động mô thức CTC cũng tương tự như Timer0. Tuy nhiên Timer1 có 2 lựa chọn cho mô thức CTC là CTC4 hoặc CTC12.

Trong mô thức CTC4, đặt WGM13:WGM10=0100, giá trị TOP của bộ đếm đặt trong thanh ghi OCR1A. Hoạt động kết quả so sánh kênh A và B hoàn toàn tương tự như Timer0.

Còn trong mô thức CTC12, đặt WGM13:WGM10=1100, giá trị TOP của bộ đếm đặt trong thanh ghi ICR1. Khi bộ đếm TCNT1 đạt đến giá trị đặt trong OCR1A/B, cờ OCF1A/B = 1 tương ứng. Sau đó bộ đếm tiếp tục đếm tới giá trị TOP đặt trong ICR1 và trở về 0, cờ ICF1=1 tương ứng. Tất nhiên để kết quả so sánh xảy ra ở kênh A hoặc B, giá trị đặt trong OCR1A/B không lớn hơn ICR1.

**Ví dụ 7.13:** Từ sơ đồ hình 7.38, viết một chương trình thực hiện LED0 sáng chớp tắt 5 lần và LED1 sáng chớp tắt 4 lần trong 2s, sau đó cả 2 đèn đều tắt trong 2s, LED2 sáng trong 2s và tắt trong 2s, lập lại chu kỳ sáng chớp tắt trở lại như trên.

**Giải:**

Ta chọn Timer1 chạy mô thức CTC12. Điều khiển thời đoạn 2s bằng giá trị TOP, chọn hệ số chia  $N=256$  tương ứng  $T_{CLK}=32$  theo như Bảng 7.10, ta tính được  $TOP+1=2,000,000/32=62500$ , suy ra  $ICR1=TOP=62499$ .

Chọn kết quả so sánh kênh A điều khiển LED0, chu kỳ sáng chớp tắt  $LED0=2/5=0.4\text{s}$ , thời đoạn sáng=0.2s, suy ra  $OCR1A+1=200,000/32=6250=TP0$ . Trong 2s từ lúc bộ đếm TCNT1 đếm từ 0 đến  $TOP=62499$  giá trị đặt trong OCR1A thay đổi tương ứng 10 lần mỗi lần tăng thêm TP0 như Bảng 7.11.

**Bảng 7.11:** Thay đổi giá trị nạp OCR1A trong 2s

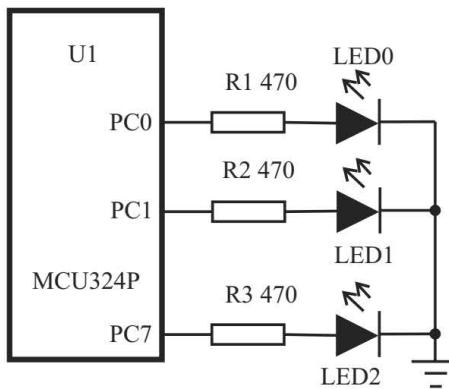
Lần nạp	1	2	3	4	5	6	7	8	9	10
OCR1A	TP0-1	2TP0-1	3TP0-1	4TP0-1	5TP0-1	6TP0-1	7TP0-1	8TP0-1	9TP0-1	10TP0-1
PC0	1	0	1	0	1	0	1	0	1	0

Chọn kết quả so sánh kênh B điều khiển LED1, chu kỳ sáng chớp tắt  $LED0=2/4=0.5\text{s}$ , thời đoạn sáng=0.25s, suy ra  $OCR1B+1=250,000/32=7812=TP1$ . Trong 2s từ lúc bộ đếm TCNT1 đếm từ 0 đến  $TOP=62499$  giá trị đặt trong OCR1B thay đổi tương ứng 8 lần mỗi lần tăng thêm TP1 như Bảng 7.12.

**Bảng 7.12:** Thay đổi giá trị nạp OCR1A trong 2s

Lần nạp	1	2	3	4	5	6	7	8
OCR1B	TP1-1	2TP1-1	3TP1-1	4TP1-1	5TP1-1	6TP1-1	7TP1-1	8TP1-1
PC1	1	0	1	0	1	0	1	0

Chọn Timer1 đạt giá trị TOP, cờ ICF1=1 sau mỗi 2s, điều khiển lái LED2



Hình 7.38: Sơ đồ mạch ví dụ 7.13

❖ Chương trình hợp ngữ ví dụ 7.13

```

.DEF COUNT0=R19
.DEF COUNT1=R20
.DEF OPD2=R21
.DEF OPD1L=R24
.DEF OPD1H=R25
.EQU LED0=0
.EQU LED1=1
.EQU LED2=7
.EQU TF=62499 ;TF=gia trị đặt trước ICR1
.EQU TP0=6250 ;TP0=gia trị bước đặt trước OCRIA
.EQU TP1=7812 ;TP1=gia trị bước đặt trước OCR1B
.ORG 0
RJMP MAIN
.ORG 0X40
MAIN:
LDI R16,HIGH(RAMEND) ;đưa stack lên đỉnh SRAM
OUT SPH,R16
LDI R16,LOW(RAMEND)
OUT SPL,R16
LDI R16,(1<<LED0)|(1<<LED1)|(1<<LED2) ;đặt LED0,LED1,LED2 output
OUT DDRC,R16
CBI PORTC,LED0 ;tắt LED0
CBI PORTC,LED1 ;tắt LED1
CBI PORTC,LED2 ;tắt LED2
LDI R16,HIGH(TF) ;nạp ICR1
STS ICR1H,R16
LDI R16,LOW(TF)
STS ICR1L,R16
LDI R16,0X00 ;Timer1 mode CTC12
STS TCCR1A,R16
LDI R16,0B00011100 ;Timer1 mode CTC12,N=256, chạy Timer1
STS TCCR1B,R16
SBIS TIFR1,ICF1 ;đọc thanh ghi cờ Timer1
RJMP LED2_CHK ;cờ ICF1=0 kiểm tra trạng thái LED2
SBI TIFR1,ICF1 ;xóa cờ ICF1
LDI R16,(1<<LED2) ;đảo bit LED2
RCALL OUT_P ;gọi ctc đảo bit
LED2_CHK: SBIS PORTC,LED2 ;LED2=1 cho phép xuất LED0,LED1
RJMP CL_LED ;LED2=0 cấm xuất LED0,LED1
WAIT1: SBIS TIFR1,OCF1A ;đọc thanh ghi cờ Timer1
RJMP WAIT2 ;cờ OCF1A=0 kiểm tra cờ OCF1B

```

```

SBI    TIFR1,OCF1A      ;xóa cờ OCF1A
LDI    R16,(1<<LED0)   ;đảo bit LED0
RCALL OUT_P
INC    COUNT0             ;tăng hệ số nhân kênh A
LDI    OPD1H,HIGH(TP0)   ;tính giá trị nạp OCR1A
LDI    OPD1L,LOW(TP0)
MOV    OPD2,COUNT0
RCALL MUL16_8
SBIW   OPD1H:OPD1L,1
STS    OCR1AH,OPD1H
STS    OCR1AL,OPD1L
CPI    COUNT0,10
BRCS   WAIT2
LDI    COUNT0,0
SBIS   TIFR1,OCF1B
RJMP   WAIT0
SBI    TIFR1,OCF1B
LDI    R16,(1<<LED1)
RCALL OUT_P
INC    COUNT1             ;tăng hệ số nhân kênh B
LDI    OPD1H,HIGH(TP1)
LDI    OPD1L,LOW(TP1)
MOV    OPD2,COUNT1
RCALL MUL16_8
SBIW   OPD1H:OPD1L,1
STS    OCR1BH,OPD1H
STS    OCR1BL,OPD1L
CPI    COUNT1,8
BRCS   NEXT
LDI    COUNT1,0
NEXT: RJMP   WAIT0
CL LED: CBI    PORTC,LED0  ;tắt LED0
          CBI    PORTC,LED1  ;tắt LED1
          LDI    COUNT0,0     ;xóa hệ số nhân kênh A
          LDI    COUNT1,0     ;xóa hệ số nhân kênh B
          LDI    R16,0
          STS    OCR1AH,R16   ;xóa OCR1A
          STS    OCR1AL,R16
          STS    OCR1BH,R16
          STS    OCR1BL,R16
          RJMP   WAIT0
          ;OUT_P đảo bit PORTC xác định bởi R16
OUT P:  IN     R17,PORTC
        EOR   R17,R16
        OUT   PORTC,R17
        RET
;
;MUL16_8 nhân số nhị phân 16 bit cho 8 bit,kết quả 24 bit
;Inputs: OPD1H:OPD1L số bị nhân 16 bit,OPD2 số nhân 8 bit
;Outputs: OPD2 byte cao,OPD1H:OPD1L 2 byte thấp kết quả
;Sử dụng R2,R3
;
MUL16_8: MUL   OPD1L,OPD2      ;nhân byte thấp số bị nhân(SBN) với số nhân(SN)

```

```

MOVW R2,R0          ;chuyển tích byte thấp(1) vào R3:R2
MUL    OPD1H,OPD2   ;nhân byte cao SBN với SN cho tích byte cao(2)
ADD    R3,R0          ;cộng byte thấp tích (2) với byte cao tích(1)
MOV    OPD1H,R3      ;chuyển byte giữa kết quả vào OPD1H
CLR    OPD2          ;xóa OPD2
ADC    OPD2,R1      ;cộng byte cao tích(2) với số nhớ
MOV    OPD1L,R2      ;chuyển byte thấp kết quả tích vào OPD1L
RET

```

- Timer1 chạy mô thức CTC12, TOP=ICR1, cứ sau mỗi 2s(còn ICF1=1) đảo bit báo trạng thái LED2, đồng thời kiểm tra nếu LED2=0 xóa các thanh ghi OCR1A/B, biến count0/1, tắt các LED0/1, nếu LED2=1 sẽ kiểm tra cờ OCF1A/B.
- Với kênh A, sau mỗi 0.2s đạt kết quả so sánh(còn OCF1A=1), đảo bit LED0 và cập nhật giá trị mới tăng thêm TP=6250 cho thanh ghi OCR1A. Sau 10 lần cập nhật OCR1A đủ 2s.
- Với kênh B, sau mỗi 0.25s đạt kết quả so sánh(còn OCF1B=1), đảo bit LED1 và cập nhật giá trị mới tăng thêm TP=7812 cho thanh ghi OCR1B. Sau 8 lần cập nhật OCR1B đủ 2s.
- Khi khởi động hoặc khi LED2=1 mới bắt đầu kiểm tra cờ OCF1A/B, giá trị OCR1A/B=0, bộ đếm TCNT1 đếm từ 0 lên nên xảy ra kết quả so sánh bằng kênh A/B, cờ OCF1A/B=1, OCR1A/B bắt đầu nạp giá trị đặt trước đầu tiên.
- Chương trình con OUT\_P đảo bit ngõ ra PORTC theo nội dung R16.
- Chương trình con MUL16\_8 nhân số nhị phân 16 bit cho 8 bit, kết quả tích số dài 24 bit. Để thuận tiện nên chọn cặp thanh ghi OPD1H:OPD1L=R25:R24 để thực hiện lệnh trừ 1 word và nạp dữ liệu tức thời.

➤ **Lưu ý:** *Ở lệnh xóa đọc và xóa các cờ, ví dụ đọc và xóa cờ ICF1 ta không sử dụng dạng :*

```

WAIT0:IN  R17,TIFR1
        SBRS  R17,ICF1
        RJMP  WAIT0
        OUT   TIFR1,R17

```

*Bởi vì có thời điểm cả 3 cờ ICF1, OCF1A, OCF1B đều bằng 1, nên đoạn lệnh trên sẽ xóa cả 3 cờ !*

#### ❖ Chương trình C ví dụ 7.13

```

#include <avr/io.h>
#define outport PORTC      // định nghĩa PORTC=output
const char led0=0,led1=1,led2=7; // ký hiệu vị trí bit các ngõ ra
const int Tf=62499           // giá trị đặt trước ICR1
const int Tp0=6250            // giá trị mỗi bước tăng cộng thêm vào OCR1A
const int Tp1=7812             // giá trị mỗi bước tăng cộng thêm vào OCR1B
unsigned char count0,count1; // bắt đầu chương trình
int main()                  // bắt đầu chương trình
{
    DDRC |=(1<<led0)|(1<<led1)|(1<<led2); // khai báo led0,led1,led2 output
    outport=0x00                         // xóa các ngõ ra
    count0=0                            // xóa hệ số nhân kênh A
    count1=0                            // xóa hệ số nhân kênh B
    ICR1=Tf                            // giá trị đặt ICR1
    TCCR1A=0x00                         // Timer1 mode CTC12
    TCCR1B=0b00011100                   // Timer1 mode CTC12,N=256
    while(1)                            // lập vòng vô hạn
    {
        if(TIFR1&(1<<ICF1))           // cờ ICF1=1?
        {
            TIFR1 &=(1<<ICF1)       // chỉ xóa cờ ICF1
            PORTC^=(1<<led2)         // đảo bit led2
        }
        if((PORTC&(1<<led2))==0)
        {

```

- Phát biểu xóa các cờ thực hiện như trên chỉ xóa một cờ duy nhất, như đã lưu ý ở phần hợp ngữ.

## ❖ Câu hỏi ôn tập

1. Cho  $TCCR1A=0x00, TCCR1B=0x82$ , đặt giá trị TOP ở đâu?
  2. Viết đoạn lệnh khởi động Timer1 tạo thời gian delay 25ms với sai số thấp nhất.
  3. So sánh hoạt động và ứng dụng của 2 mô thức CTC4, CTC12.
  4. Cho  $TCCR1A=0x00, TCCR1B=0x3C, ICR1=0x05DC, OCR1A=1200, OCR1B=1350$ . Khi Timer1 chạy, các cờ nào có thể đặt lên 1.
  5. Trong câu 4, nếu đặt  $OCR1A=0xFFFF$  cờ TOV1 có thể đặt lên 1 không?

#### 7.5.4 Timer1 làm việc bắt ngoặt vào (Input Capture)

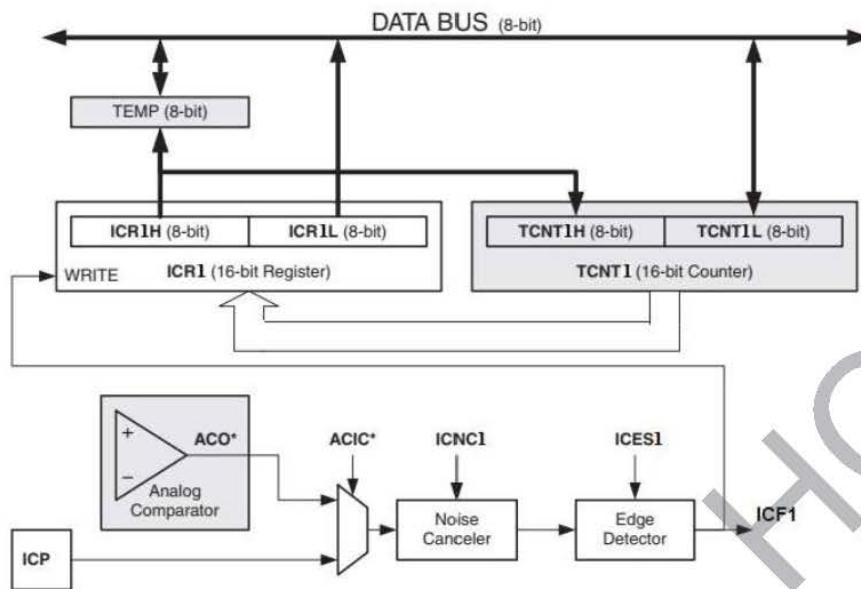
Timer1 làm việc bắt ngõ vào(Input Capture) Timer1 có thêm chức năng bắt ngõ vào, thực hiện khi có tín hiệu kích khởi từ chân ICP(PD6) hay từ ngõ ra mạch so sánh tương tự ACO, nội dung bộ đếm TCNT1 được chép qua thanh ghi ICR1 đồng thời phần cứng đặt cờ ICF1=1 chỉ báo tương ứng. Chức năng này được ứng dụng trong việc đo độ rộng, chu kỳ xung vuông, hoặc nhận dạng và đánh dấu thời điểm phát hiện 1 sự kiện từ bên ngoài...

Hình 7.39 trình bày sơ đồ khối hoạt động bắt ngõ vào, các khối tô màu xám không thuộc khối bắt ngõ vào.

### **1. Nguồn tín hiệu kích khởi bắt nguồn vào**

Tín hiệu kích khởi bắt ngõ vào được chọn từ chân ICP(PC6) hoặc từ ngõ ra ACO của bộ so sánh tương tự(xem chương 9).Việc chọn nguồn tín hiệu kích khởi do bit ACIC thuộc thanh ghi ACSR quyết định (xem chương 9).Bit ACIC=0 chọn ngõ ICP,ACIC=1 chọn ngõ ACO.

Tín hiệu kích khởi bất ngờ vào tác động cạnh lén hoặc cạnh xuống điều khiển bởi bit ICES1 thuộc thanh ghi TCCR1B. ICES1=1 chọn kích khởi cạnh lén, ICES1=0 chọn kích khởi cạnh xuống.



**Hình 7.39:** Sơ đồ khái hoạt động bắt ngõ vào

Trường hợp muốn sử dụng bộ triệt nhiễu bắt ngõ vào, ta đặt bit ICEN1=1, ICEN1 thuộc thanh ghi TCCR1B. Tín hiệu bắt ngõ vào được lấy mẫu 4 lần liên tiếp để xác định có chuyển biến cạnh đúng mới thực hiện bắt ngõ vào. Do đó khi sử dụng triệt nhiễu, đáp ứng cập nhật thanh ghi ICR1 sẽ trễ 4 chu kỳ dao động kể từ thời điểm xuất hiện cạnh xung tác động ngõ vào.

➤ *Lưu ý: Khi khai báo Timer1 ở các mô thức sử dụng thanh ghi ICR1=TOP, bit ICEN1, ICES1 không ảnh hưởng và các ngõ ICP/ACO không hoạt động như ngõ vào kích khởi*

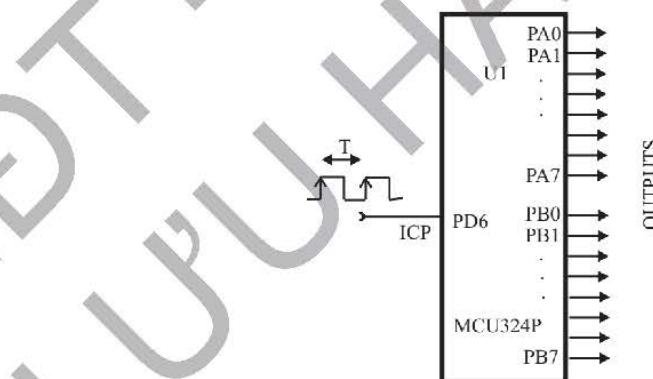
## 2. Hoạt động thanh ghi ICR1

Khi có cạnh xung kích khởi, MCU lập tức chép nội dung bộ đếm TCNT1 vào thanh ghi ICR1, đồng thời phần cứng đặt cờ ICF1=1. Trường hợp có cho phép ngắt bắt ngõ vào, phần cứng tự động xóa cờ ICF1 khi MCU chuyển điều khiển đến vector ngắt bắt ngõ vào (xem chương 10). Hoặc ta phải xóa cờ ICF1 bằng cách ghi bit 1 vào vị trí bit cờ ICF1.

Khi áp dụng Timer1 bắt ngõ vào, để tránh sai sót ta chỉ nên đọc thanh ghi ICR1 và đọc byte thấp trước, byte cao sau như quy định truy xuất thanh ghi 16 bit.

**Ví dụ 7.14:** Viết một chương trình đo chu kỳ xung vuông  $T=10\mu s - 50000\mu s$ . Xuất giá trị nhị phân đo được ra PORTB byte cao và PORTA byte thấp.

**Giải:**



**Hình 7.40:** Mạch đo chu kỳ xung ví dụ 7.14

Ta cho Timer1 chạy mô thức NOR, giá trị MAX=65535 nên để đo tới giá trị  $50000\mu s$  xung CLKT1 phải bằng  $1\mu s$ . Độ rộng xung đếm này cũng đáp ứng được tầm đo tối thiểu  $10\mu s$ . Do đó ta sẽ khai báo Timer1 mô thức NOR, N=8, chọn xung bắt ngõ vào kích khởi cạnh lên.

Cho tín hiệu cần đo chu kỳ vào ngõ ICP(PD6)

Khi Timer1 chạy thực hiện:

- Chờ cờ ICF1=1 và xóa cờ ICF1, cát giá trị ICR1 lần 1=ICR1(1)
- Lặp lại bước trên, cát giá trị ICR1 lần 2=ICR1(2)
- Trừ giá trị ICR1(2) cho ICR1(1) được thời gian chu kỳ tín hiệu vào ngõ ICP.

### ❖ Chương trình hợp ngữ ví dụ 7.14

```

.ORG 0
RJMP MAIN
.ORG 0X40
MAIN: LDI R16,HIGH(RAMEND) ;đưa stack lên đỉnh SRAM
      OUT SPH,R16
      LDI R16,LOW(RAMEND)
      OUT SPL,R16
      LDI R16,0XFF
      OUT DDRA,R16          ;PortA output
      OUT DDRB,R16          ;PortB output
      LDI R16,0X00
      STS TCCR1A,R16         ;Timer1 mode NOR
      LDI R16,0B01000010     ;Timer1 mode NOR,N=8,bắt ngõ vào cạnh lên
      STS TCCR1B,R16
      WAIT0: IN R17,TIFR1
              SBRS R17,ICF1
              RJMP WAIT0
              OUT TIFR1,R17
              LDS R18,ICR1L
              LDS R19,ICR1H
              WAIT1: IN R17,TIFR1
                      SBRS R17,ICF1
                      RJMP WAIT1
                      OUT TIFR1,R17
                      LDS R20,ICR1L
                      LDS R21,ICR1H
                      SUB R20,R18
                      SBC R21,R19
                      OUT PORTA,R20
                      OUT PORTB,R21
                      RJMP WAIT0

```

;

;đọc thanh ghi cờ Timer1

;chờ cờ ICF1=1 báo kết quả so sánh ICR1

;chờ cờ ICF1=1

;xóa cờ ICF1

;cắt thời điểm lấy mẫu cạnh lên đầu tiên

;

;đọc thanh ghi cờ Timer1

;chờ cờ ICF1=1 báo kết quả so sánh ICR1

;chờ cờ ICF1=1

;xóa cờ ICF1

;cắt thời điểm lấy mẫu cạnh lên tiếp theo

;

;trừ thời điểm sau với thời điểm đầu

;

;xuất byte thấp chu kỳ ra PortA

;xuất byte cao chu kỳ ra PortB

### ❖ Chương trình C ví dụ 7.14

```

#include <avr/io.h>
unsigned int temp0,temp1;
int main()
{
    DDRA |=0xff           // bắt đầu chương trình
    DDRB |=0xff
    TCCR1A=0x00
    TCCR1B=0x42
    while(1)               // lập vòng vô hạn
    {
        while(!(TIFR1&(1<<ICF1))) ;//cờ ICF1=1?
        TIFR1 |=1<<ICF1       ;//xóa cờ ICF1
        temp0=ICR1             ;//thời điểm cạnh xung đầu tiên
        while(!(TIFR1&(1<<ICF1))) ;//cờ ICR1=1?
        TIFR1 |=1<<ICF1       ;//xóa cờ ICR1
        temp1=ICR1             ;//thời điểm cạnh xung tiếp theo
        temp1=temp1-temp0      ;//tính chu kỳ
        PORTA=temp1            ;//xuất byte thấp ra PortA
        PORTB=temp1>>8        ;//xuất byte cao ra PortB
    }
}

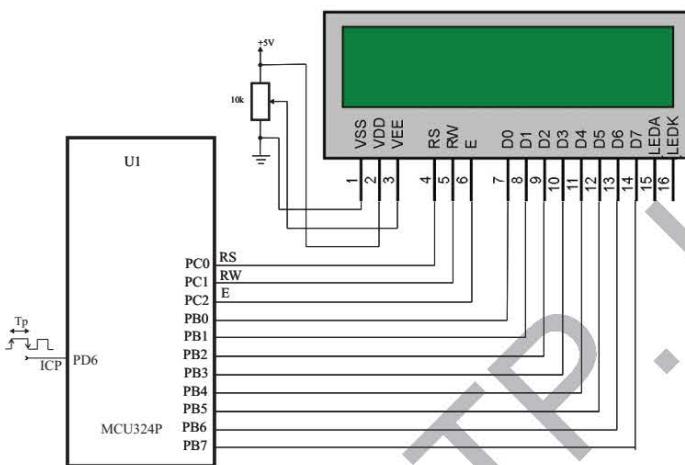
```

}

- Người đọc thử chạy chương trình hợp ngữ và C ví dụ 7.14 xem có thể đo chu kỳ xung tối thiểu bao nhiêu và cho nhận xét.

**Ví dụ 7.15:** Thiết kế mạch đo độ rộng xung từ 10 $\mu$ s - 50000 $\mu$ s, hiển thị giá trị đo ra LCD ký tự 16x2.

**Giải:**



**Hình 7.41:** Sơ đồ thiết kế mạch đo độ rộng xung ví dụ 7.15

Ta áp dụng giao tiếp LCD ký tự 16x2 trong ví dụ 6.16 chương 6. Cho tín hiệu cần đo độ rộng xung vào chân ICP(PD6). Áp dụng như ví dụ 7.14, cho Timer1 chạy mô thức NOR, N=8. Để đo độ rộng xung, đầu tiên ta chọn xung bắt ngõ vào kích khởi cảnh lên và sau đó kích khởi cảnh xuống. Hiệu số giữa thời điểm lấy mẫu sau và trước chính là độ rộng xung.

Màn hình LCD trình bày hiển thị như sau:

Do do rong xung:  
Tp( $\mu$ s)= 50000

#### ❖ Chương trình hợp ngữ ví dụ 7.15

```

.DEF OPD1_L=R20
.DEF OPD1_H=R21
.DEF OPD2=R22
.DEF OPD3=R23
.DEF COUNT=R18
.EQU OUTPORT=PORTB      ;PORTB data
.EQU INPORT=PINB
.EQU IOSETB=DDRB
.EQU CONT=PORTC          ;PORTC điều khiển
.EQU CONT_DR=DDRC         ;
.EQU CONT_IN=PINC         ;
.EQU RS=0                 ;bit RS
.EQU RW=1                 ;bit RW
.EQU E=2                  ;bit E
.EQU NULL=$00              ;mã kết thúc
.EQU BCD_BUF=0X200
.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 cấm 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
LDI    R16,2               ;delay 200μs
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
                           ;khi nhập ký tự,màn hình không dịch
                           ;ctc khởi động LCD 8 bit
-----;
RCALL INIT_LCD8

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
LDI    R17,$80             ;con trỏ bắt đầu ở đầu dòng 1
RCALL CURS_POS          ;xuất lệnh ra LCD
LDI    ZH,HIGH(MSG1<<1)  ;Z trỏ đầu bảng tra MSG1
LDI    ZL,LOW(MSG1<<1)
RCALL MSG_DISP            ;ghi MSG1 ra LCD
LDI    R17,$C0             ;con trỏ bắt đầu ở đầu dòng 2
RCALL CURS_POS          ;xuất lệnh ra LCD
LDI    ZH,HIGH(MSG2<<1)  ;Z trỏ đầu bảng tra MSG2
LDI    ZL,LOW(MSG2<<1)
RCALL MSG_DISP            ;ghi MSG2 ra LCD
LDI    R16,0X00
STS   TCCR1A,R16          ;Timer1 mode NOR
LDI    R16,0B01000010      ;Timer1 mode NOR,N=8,bắt ngõ vào cạnh lên
STS   TCCR1B,R16
SBI   TIFR1,ICF1          ;xóa cờ ICF1 chuẩn bị đo

```

START:

WAIT0:	IN R17,TIFR1	;đọc thanh ghi cờ Timer1
	SBRS R17,ICF1	;chờ cờ ICF1=1 báo kết quả so sánh ICR1
	RJMP WAIT0	;chờ cờ ICF1=1
	OUT TIFR1,R17	;xóa cờ ICF1
	LDS R18,ICR1L	;cắt thời điểm lấy mẫu cạnh lên đầu tiên
	LDS R19,ICR1H	
	LDI R16,0B00000010	;Timer1 mode NOR,N=8,bắt ngõ vào cạnh xuống
	STS TCCR1B,R16	;
WAIT1:	IN R17,TIFR1	;đọc thanh ghi cờ Timer1
	SBRS R17,ICF1	;chờ cờ ICF1=1 báo kết quả so sánh ICR1
	RJMP WAIT1	;chờ cờ ICF1=1
	OUT TIFR1,R17	;xóa cờ ICF1
	LDS R20,ICR1L	;cắt thời điểm lấy mẫu cạnh lên tiếp theo
	LDS R21,ICR1H	
	LDI R16,0B01000010	;Timer1 mode NOR,N=8,bắt ngõ vào cạnh lên
	STS TCCR1B,R16	;
	SUB R20,R18	;trừ thời điểm sau với thời điểm đầu
	SBC R21,R19	
	RCALL BIN16_BCD5DG	;chuyển số nhị phân sang BCD
	LDI R17,\$C8	;con trỏ bắt đầu ở đầu vị trí 9 dòng 2
	RCALL CURS_POS	;xuất lệnh ra LCD
	RCALL NUM_DISP	;hiển thị độ rộng xung
	RJMP START	,lặp vòng đo lại

-----  
;BIN16\_BCD5DG chuyển đổi số nhị phân 16 bit sang số BCD 5 digit  
;Inputs: OPD1\_H=R21:OPD1\_L=R20 chứa số nhị phân 16 bit  
;Outputs: BCD\_BUF:BCD\_BUF+4:địa chỉ SRAM chứa 5 digit BCD từ cao đến thấp  
;Sử dụng R17,COUNT,X,etc DIV16\_8  
;-----

#### BIN16\_BCD5DG:

	LDI XH,HIGH(BCD_BUF);X trỏ địa chỉ đầu buffer BCD	
	LDI XL,LOW(BCD_BUF)	
	LDI COUNT,5 ;đếm số byte bộ nhớ	
	LDI R17,0X00 ;nạp giá trị 0	
LOOP_CL:	ST X+,R17 ;xóa buffer bộ nhớ	
	DEC COUNT ;đếm đủ 5 byte	
	BRNE LOOP_CL ;nạp SC	
DIV_NXT:	LDI OPD2,10 ;chia số nhị phân 16 bit cho 10	
	RCALL DIV16_8 ;cắt số dư vào buffer	
	ST -X,OPD3 ;thương số=0?	
	CPI OPD1_L,0 ;khác 0 chia tiếp	
	BRNE DIV_NXT ;hiển thị 5 ký tự	
	RET	

-----  
;NUM\_DISP hiển thị 5 ký tự địa chỉ đầu BCD\_BUF trong SRAM  
;(BCD\_BUF)=digit cao nhất,(BCD\_BUF+4)=digit thấp nhất  
;Sử dụng R16,R17,COUNT,X,etc DELAY\_US,etc OUT\_LCD  
;-----

NUM_DISP:	LDI COUNT,5 ;hiển thị 5 ký tự	
	LDI XH,HIGH(BCD_BUF);X trỏ địa chỉ đầu buffer	
	LDI XL,LOW(BCD_BUF)	
DISP_NXT:	LD R17,X+ ;lấy số BCD từ buffer,tăng địa chỉ buffer kế tiếp	
	LDI R16,0X30 ;chuyển sang mã ASCII	
	ADD R17,R16	
	LDI R16,1 ;chờ 100μs	

```

        RCALL DELAY_US
        SBI    CONT,RS      ;RS=1 ghi data hiển thị LCD
        RCALL OUT_LCD       ;ghi ký tự ra LCD
        DEC    COUNT         ;hiển thị đủ 5 ký tự
        BRNE  DISP_NXT
        RET

;-----;
;DIV16_8 chia số nhị phân 16 bit OPD1 cho 8 bit OPD2
;Input: OPD1_H,OPD1_L= SBC(GPR16-31)
;      OPD2=SC(GPR0-31)
;Output:OPD1_H,OPD1_L=thương số
;      OPD3=DS(GPR0-31)
;Sử dụng COUNT(GPR16-31)
;-----;

DIV16_8: LDI    COUNT,16   ;COUNT=đếm 16
          CLR    OPD3      ;xóa dư số
SH_NXT:  CLC
          LSL    OPD1_L     ;dịch trái SBC L,bit0=C=thương số
          ROL    OPD1_H     ;quay trái SBC H,C=bit7
          ROL    OPD3      ;dịch bit7 SBC H vào dư số
          BRCS  OV_C       ;tràn bit C=1,chia được
          SUB   OPD3,OPD2   ;trừ dư số với số chia
          BRCC  GT_TH      ;C=0 chia được
          ADD   OPD3,OPD2   ;C=1 không chia được,không trừ
          RJMP  NEXT
OV_C:   SUB   OPD3,OPD2   ;trừ dư số với số chia
GT_TH:  SBR   OPD1_L,1   ;chia được,thương số=1
NEXT:   DEC   COUNT      ;đếm số lần dịch SBC
        BRNE  SH_NXT     ;chưa đủ tiếp tục dịch bit
        RET

;-----;
;CURS_POS đặt con trỏ tại vị trí có địa chỉ trong R17
;Input: R17=$80 -$8F dòng 1,$C0-$CF dòng 2
;      R17= địa chỉ vị trí con trỏ
;Sử dụng R16,etc DEAY_US,OUT_LCD
;-----;

CURS_POS: LDI   R16,1     ;chờ 100μs
          RCALL DELAY_US
          CBI    CONT,RS      ;RS=0 ghi lệnh
          RCALL OUT_LCD
          RET

;-----;
;MSG_DISP hiển thị chuỗi ký tự kết thúc bằng mã NULL đặt trong Flash ROM
;Input: Z chứa địa chỉ đầu chuỗi ký tự
;Output: hiển thị chuỗi ký tự ra LCD tại vị trí con trỏ hiện hành
;Sử dụng R16,R17,etc DELAY_US,OUT_LCD
;-----;

MSG_DISP: LPM   R17,Z+     ;lấy mã ASCII ký tự từ Flash ROM
          CPI   R17,NULL     ;kiểm tra ký tự kết thúc
          BREQ  EXIT_MSG     ;ký tự NULL thoát
          LDI   R16,1         ;chờ 100μs
          RCALL DELAY_US
          SBI   CONT,RS      ;RS=1 ghi data hiển thị LCD
          RCALL OUT_LCD     ;ghi mã ASCII ký tự ra LCD
          RJMP  MSG_DISP     ;tiếp tục hiển thị ký tự

```

```

EXIT_MSG:    RET
;-----  

;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  

              CBI    CONT,RS          ;RS=0 ghi lệnh  

              MOV    R17,R18          ;R18=Function set  

              RCALL OUT_LCD          ;  

              LDI    R16,1           ;chờ 100µs  

              RCALL DELAY_US  

              CBI    CONT,RS          ;RS=0 ghi lệnh  

              MOV    R17,R19          ;R19=Clear display  

              RCALL OUT_LCD          ;  

              LDI    R16,20          ;chờ 2ms sau lệnh Clear display  

              RCALL DELAY_US  

              CBI    CONT,RS          ;RS=0 ghi lệnh  

              MOV    R17,R20          ;R20=Display on/off control  

              RCALL OUT_LCD          ;  

              LDI    R16,1           ;chờ 100µs  

              RCALL DELAY_US  

              CBI    CONT,RS          ;RS=0 ghi lệnh  

              MOV    R17,R21          ;R21=Entry mode set  

              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  

              BRNE   L2             ;2/1MC  

              DEC    R15           ;1MC  

              BRNE   L1             ;2/1MC  

              RET  

              .ORG   0X0200          ;4MC
;  

MSG1: .DB "Do do rong xung:",$00
MSG2: .DB "Tp(", $E4, "s)=", $00      ;$E4 mã ký tự µ

```

- Trong lần lấy mẫu giá trị cạnh xuống,sau khi cắt nội dung ICR1 phải chuyển trạng thái kích cạnh lên ngay,sau đó mới xử lý việc hiển thị ra LCD.

- Trong phần xử lý hiển thị ra LCD,tu áp dụng ví dụ 6.16 chương 6.Để dễ tham khảo,tu viết thành các chương trình con tiện dụng như sau:

- CURS\_POS đặt con trỏ tại vị trí cần hiển thị.Địa chỉ con trỏ đặt trong R17 theo quy ước  
Dòng 1: từ \$80 đến \$8F vị trí 1 đến 16  
Dòng 2: từ \$C0 đến \$CF vị trí 1 đến 16
- MSG\_DISP hiển thị chuỗi ký tự mã ASCII cắt trong Flash ROM.Chuỗi ký tự được định nghĩa bằng chỉ dẫn DB,có đặt nhãn ký hiệu địa chỉ đầu chuỗi,kết thúc chuỗi bằng mã NULL=\$00
- NUM\_DISP hiển thị 5 số BCD cắt trong SRAM địa chỉ đầu BCD\_BUF chứa digit cao nhất. Khi hiển thị ra LCD phải chuyển sang mã ASCII bằng cách cộng thêm 0x30
- BIN16\_BCD5DG chuyển số nhị phân 16 bit sang BCD 5 digit bằng cách liên tục chia thương số cho 10 cho đến khi thương số=0,cắt các dư số vào buffer,digit thấp vào địa chỉ cao.Phần này có áp dụng etc DIV16\_8 chia số nhị phân 16 bit cho 8 bit ở ví dụ 7.9.Để thuận tiện nên định nghĩa OPD1\_H:OPD1\_L=R21:R20 chứa giá trị độ rộng xung.
- Các chương trình con còn lại xem lại ví dụ 6.16

## ❖ Chương trình C VD7.15

```
#include <avr/io.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
const char RS=0,RW=1,E=2 ;//các ký hiệu điều khiển LCD
const char NULL=0x00 ;//ký hiệu mã kết thúc
void CURS_POS(unsigned char x) ;//khai báo hàm đặt vị trí con trỏ ra LCD
void MSG_DISP(unsigned char *pf) ;//khai báo hàm hiển thị chuỗi ký tự
void BIN16_BCD5DG() ;//khai báo hàm chuyển đổi số nhị phân sang BCD
void NUM_DISP() ;//khai báo hàm hiển thị số BCD ra LCD
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 MSG1[]="Do do rong xung:\x00"; //\x00 ký hiệu mã ASCII $00
unsigned char MSG2[]="Tp(\xe4""s)= \x00"; //\xe4 ký hiệu µ
unsigned char BCD_BUF[5];//khai báo buffer chứa số BCD
unsigned char i,tam,*p;
unsigned int temp0,temp1;
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&=~(1<<RS); //truy xuất lệnh
    tam=0x30 //mã lệnh $30 lần 1
    OUT_LCD(tam);
    delay_ms(5600); //delay 4.2ms
    cont&=~(1<<RS); //truy xuất lệnh
    tam=0x30 //mã lệnh $30 lần 2
```

```

OUT_LCD(tam) ;
delay_ms(2673) ;//delay 0.2ms
cont&=~(1<<RS);//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
delay_ms(133) ;//delay 0.1ms
cont&=~(1<<RS);//truy xuất lệnh
tam=0x01 ;//mã lệnh xóa màn hình
OUT_LCD(tam) ;
delay_ms(2667) ;//delay 2ms sau lệnh clear display
CURS_POS(0x80) ;//đặt con trỏ vị trí đầu dòng 1
MSG_DISP(p=MSG1);//hiển thị chuỗi ký tự MSG1
CURS_POS(0xc0) ;//đặt con trỏ vị trí đầu dòng 2
MSG_DISP(p=MSG2);//hiển thị chuỗi ký tự MSG2
TCCR1A=0x00 ;//Timer1 mode NOR
TCCR1B=0x42 ;//Timer1 mode NOR,N=8,xung kích cạnh lên
while(1) // lặp vòng vô hạn
{
    TIFR1 |=(1<<ICF1) ;//xóa cờ ICF1 chuẩn bị đếm
    while(!(TIFR1&(1<<ICF1))) ;//cờ ICF1=1?
    TIFR1 |=(1<<ICF1) ;//xóa cờ ICF1
    temp0=ICR1 ;//thời điểm cạnh lên xung
    TCCR1B=0x02 ;//Timer1 mode NOR,N=8,xung kích cạnh xuống
    while(!(TIFR1&(1<<ICF1))) ;//cờ ICR1=1?
    TIFR1 |=(1<<ICF1) ;//xóa cờ ICR1
    temp1=ICR1 ;//thời điểm cạnh xuống xung kế tiếp
    TCCR1B=0x42 ;//Timer1 mode NOR,N=8,xung kích cạnh lên
    temp1=temp1-temp0
    BIN16_BCD5DG() ;//chuyển số nhị phân sang BCD
    CURS_POS(0xc8) ;//đặt vị trí con trỏ dòng 2
    NUM_DISP() ;//hiển thị độ rộng xung
}
}

//-----
void BIN16_BCD5DG() //hàm chuyển đổi số nhị phân 16 bit sang BCD 5 digit
{
    for(i=0;i<=4;i++) ;//xóa buffer BCD 5 ô nhớ
    BCD_BUF[i]=0 ;
    i=4 ;//khởi động ô nhớ thứ 5
    do
    {
        BCD_BUF[i]=temp1%10;//cắt dư số phép chia 10 vào buffer
        i-- ;//giảm địa chỉ buffer
        temp1=temp1/10; //lấy thương số phép chia
    }
    while(temp1!=0) ;//thoát khi thương số=0
}

//-----
void NUM_DISP() //hàm hiển thị 5 digit
{
    for(i=0;i<=4;i++) //lặp vòng 5 lần
    {
        tam=BCD_BUF[i]+0x30 ;//chuyển số BCD sang mã ASCII
        delay_ms(133) ;//delay 0.1ms
    }
}

```

```

        cont |=(1<< RS) ;//truy xuất data
        OUT_LCD(tam) ;//ghi ký tự hiển thị ra LCD
    }
}

//-----
void CURS_POS(unsigned char x) //hàm đặt vị trí con trỏ tại địa chỉ x
{
    delay_ms(133) ;//delay 0.1ms
    cont&=~(1<<RS);//truy xuất lệnh
    OUT_LCD(x) ;
}

//-----
void MSG_DISP(unsigned char *pf)//hàm hiển thị chuỗi ký tự kết thúc bằng mã NULL trả bởi pf
{
    tam=*pf ;//lấy mã ASCII từ chuỗi
    while(tam!=NULL) //ký tự NULL?
    {
        delay_ms(133) ;//delay 0.1ms
        cont |=(1<< RS) ;//truy xuất data
        OUT_LCD(tam) ;//ghi ký tự hiển thị ra LCD
        tam=*&pf ;//lấy mã ASCII từ chuỗi,tăng địa chỉ kế tiếp
    }
}

//-----
void INIT_LCD8(char dat1,char dat2,char dat3,char dat4)
{
    delay_ms(133) ;//delay 0.1ms
    cont&=~(1<<RS) ;//truy xuất lệnh
    tam=dat1 ;//mã lệnh function set
    OUT_LCD(tam) ;
    delay_ms(133) ;//delay 0.1ms
    cont&=~(1<<RS) ;//truy xuất lệnh
    tam=dat2 ;//mã lệnh clear display
    OUT_LCD(tam) ;
    delay_ms(2667) ;//delay 2ms sau lệnh clear display
    cont&=~(1<<RS) ;//truy xuất lệnh
    tam=dat3 ;//mã lệnh display control on/off
    OUT_LCD(tam) ;
    delay_ms(133) ;//delay 0.1ms
    cont&=~(1<<RS) ;//truy xuất lệnh
    tam=dat4 ;//mã lệnh entry mode set
    OUT_LCD(tam) ;
}

//-----
void OUT_LCD(unsigned char val)
{
    output=val;
    cont |=(1<<E);
    cont&=~(1<<E);
}

//-----
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

1. Nguồn tín hiệu bắt ngõ vào có thể lấy từ đâu? Chức năng bit ACIC?
2. Khai báo tín hiệu bắt ngõ vào tác động cạnh xoong và triệt nhiễu tín hiệu bắt ngõ vào như thế nào?
3. Làm thế nào biết nội dung bộ đếm đã được chép qua thanh ghi ICR1?
4. Trong ví dụ 7.14, có thể đo chu kỳ xung tối thiểu bao nhiêu  $\mu s$ ?
5. Trong ví dụ 7.15, có thể đo độ rộng xung tối thiểu bao nhiêu  $\mu s$ ?

## 7.6 Lập trình đếm sự kiện

Cấu hình Timer0 và Timer1 cho phép hoạt động mô thức đếm sự kiện từ xung đếm bên ngoài.

Xem lại hình 7.5, T0(PB0) là ngõ vào xung đếm cho Timer0 và T1(PB1) là ngõ vào xung đếm cho Timer1. Mô thức NOR và CTC vẫn áp dụng cho xung đếm ngoài, chỉ khác là không sử dụng bộ chia đặt trước được, do phải đặt CSn2:CSn0=110 hay 111 để nhận xung từ ngõ Tn kích cạnh xoong hay cạnh lên tương ứng.

Khi cài đặt CSn1:CSn0=110 hay 111, chân Tn tự động chuyển thành ngõ vào nhận xung CK ngoài.

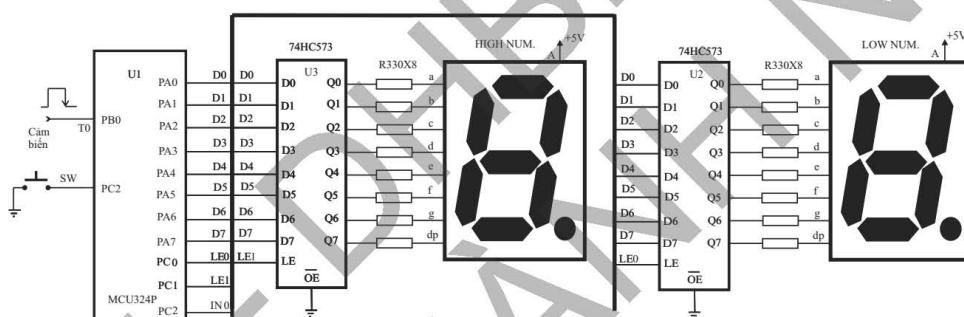
Mô thức đếm sự kiện ứng dụng trong đo đặc nhận dạng xung bên ngoài như đếm sản phẩm, đo vận tốc, lưu lượng, dung tích..., với các cảm biến cho ngõ ra dưới dạng xung đếm.

**Ví dụ 7.16:** Thiết kế mạch đếm sản phẩm hoạt động như sau: Khi nhấn SW bắt đầu đếm từ 0, mỗi lần có 1 sản phẩm đi ngang qua, cảm biến đếm phát ra 1 xung. Số sản phẩm đếm được đến 90 thì dừng, sau đó nhấn SW bắt đầu đếm lại từ đầu. Hiển thị liên tục số sản phẩm đếm được ra 2 LED 7 đoạn.

**Giải:**

Yêu cầu cầu mạch đếm đến 90 nên ta sử dụng Timer0 mô thức đếm sự kiện, CTC kênh A, xung CK ngoài kích cạnh xoong, ngõ ra xung cảm biến đếm đưa vào chân T0(PB0). Áp dụng sơ đồ ví dụ 7.11 hiển thị 2 LED 7 đoạn AC. Do phải sử dụng chân PB0 nên ta chuyển sang PortA giao tiếp với LED thay cho PortB.

Hình 7.42 trình bày sơ đồ mạch đếm sản phẩm ví dụ 7.16



Hình 7.42: Sơ đồ mạch đếm sản phẩm ví dụ 7.16

## ❖ Chương trình hợp ngữ ví dụ 7.16

```

.MAIN:
    .EQU OUTPORT=PORTA
    .EQU IOSET=DDRA
    .EQU SW=2
    .ORG 0
    RJMP MAIN
    .ORG 0X40
    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,PC2 input
    CBI PORTC,0        ;khóa ngõ ra U2
    CBI PORTC,1        ;khóa ngõ ra U3
    SBI PORTC,SW       ;điện trở kéo lên PC2
    LDI R16,0xFF

```

```

        OUT    IOSET,R16      ;khai báo Port output
        CBI    DDRB,0         ;khai báo PB0=T0 input
        SBI    PORTB,0        ;điện trở kéo lên PB0
        LDI    R16,90          ;nạp giá trị OCR0A
        OUT    OCR0A,R16      ;Timer0 mode CTC
        LDI    R16,0X02        ;TCCR0A,R16
        OUT    TCCR0A,R16      ;khởi động xóa hiển thị
        LDI    R17,0            ;ctc chuyển số nhị phân thành BCD 2 digit
        RCALL BIN8_BCD2       ;ctc số đếm
        RCALL DISP_SEC        ;-----



;-----START:-----



        LDI    R16,0X00        ;dùng Timer0
        OUT    TCCR0B,R16
        OUT    TCNT0,R16        ;xóa bộ đếm
        LDI    R16,50          ;kiểm tra SW nhấn 50 lần liên tục
        SBIC  PINC,SW
        RJMP  BACK0
        DEC   R16
        BRNE  WAIT0
        LDI    R16,50          ;kiểm tra sw nhả 50 lần liên tục
        SBIS  PINC,SW
        RJMP  BACK1
        DEC   R16
        BRNE  WAIT1
        LDI    R16,0X06        ;đếm xung ngoài ngõ T0,kích cạnh xuống
        OUT    TCCR0B,R16
        IN    R17,TCNT0        ;đọc số đếm
        RCALL BIN8_BCD2       ;ctc chuyển số nhị phân thành BCD 2 digit
        RCALL DISP_SEC        ;ctc số đếm
        IN    R17,TIFR0        ;đọc cờ OCF0A
        SBRS  R17,OCF0A        ;tiếp tục khi cờ OCF0A=1
        RJMP  WAIT2
        OUT    TIFR0,R17        ;xóa cờ OCF0A
        RJMP  START
        ;-----



;-----BIN8_BCD2 chuyển số nhị phân 8 bit sang số BCD 2 digit
;Input R17=số nhị phân 8 bit
;Output R17 số BCD nén
;Sử dụng ctc DIV8_8,R16=10 số chia
;-----



BIN8_BCD2:  LDI    R16,10      ;R16=số chia
             RCALL DIV8_8      ;ctc chia 2 số nhị phân 8 bit
             SWAP  R17          ;chuyển thương số=dư số sau cùng lên 4 bit cao
             OR    R17,R16        ;dán dư số phép chia lần 1 vào 4 bit thấp
             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

```

```

LT_DV: RJMP GT_DV ;thực hiện tiếp
        ADD R17,R16 ;lấy lại dư số
        MOV R16,R17 ;R16=đư số
        MOV R17,R15 ;R17=thương số
        RET

;-----;
;DISP_SEC hiển thị số BCD nén ra 2 LED 7 đoạn AC
;Phương pháp xuất data ra mạch chốt 8 bit
;Input: R17=số BCD nén
;Sử dụng ctc GET_7SEG
;-----;

DISP_SEC: PUSH R17 ;cất data
           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
           SWAP R17 ;hoán vị sang 4 bit thấp
           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
           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

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

```

#### ❖ Chương trình C ví dụ 7.16

```

#include <avr/io.h>
#define ioset DDRA // định nghĩa ioset=DDRA
#define outport PORTA //định nghĩa outport=PORTA
#define cont PORTC //định nghĩa cont=PORTC
#define cont_in PINC //định nghĩa cont_in=PINC
const char le0=0,le1=1,sw=2 ;//ký hiệu ngõ chốt U2,U3,SW
void bin8_bcd2() ;//khai báo hàm chuyển số nhị phân 8 bit sang BCD 2 digit
void disp_sec() ;//khai báo hàm hiển thị số đếm
unsigned char get_7seg(unsigned char x) ;//khai báo hàm HEX_ASC
unsigned char sec_ct,tam,i;
int main()
{

```

```

DDRC=0x03      ;//định nghĩa PC0,PC1 là output,PC2 input
cont&=~((1<<le0)|(1<<le1)) ;//khóa U2,U3
cont |=(1<<sw)    ;//điện trở kéo lên ngõ sw
ioset=0xff      ;//PortA output
DDRB=~(1<<PB0)  ;//PB0=T0 input
PORTB=(1<<PB0)  ;//điện trở kéo lên ngõ PB0
OCR0A=90        ;//giá trị đặt OCR0A
TCCR0A=0x02      ;//Timer0 mode CTC
tam=0            ;//khởi động xóa hiển thị
bin8_bcd2()      ;//chuyển sang BCD
disp_sec()        ;//hiển thị số đếm
while(1)
{
    TCCR0B=0x00 ;//Timer0 dừng
    TCNT0=0      ;//xóa bộ đếm
    for(i=50;i>=1;i++)//kiểm tra sw nhấn 50 lần liên tục
    {
        if(cont_in&(1<<sw))
            i=50   ;
    }
    for(i=50;i>=1;i++)//kiểm tra sw nhả 50 lần liên tục
    {
        if(!(cont_in&(1<<sw)))
            i=50   ;
    }
    TCCR0B=0x06      ;//Timer0 mode NOR,xung CK ngoài kích cạnh xuống
    while(!(TIFR0&(1<<OCF0A))) //cờ OCF0A=0 tiếp tục
    {
        tam=TCNT0;//lấy giá trị đếm
        bin8_bcd2() ;//chuyển sang BCD
        disp_sec()  ;//hiển thị số đếm
    }
    TIFR0|=(1<<OCF0A) ;//xóa cờ OCF0A
}
//-----
void bin8_bcd2()
{
    unsigned char x;
    x=tam%10          ;//x= dư số tam/10
    tam=(tam/10)<<4   ;//thương số tam/10 dịch lên 4 bit cao
    tam=tam | x        ;//dán dư số vào 4 bit thấp
}
//-----
void disp_sec()
{
    i=tam&0x0f        ;//che 4 bit cao
    outport=get_7seg(i) ;//chuyển sang mã ASCII và xuất ra U2
    cont|=(1<<le0)    ;//mở U2
    cont&=~(1<<le0)   ;//khóa U2
    i=tam>>4         ;//dịch data sang 4 bit thấp
    outport=get_7seg(i) ;//chuyển sang mã ASCII và xuất ra U3
    cont|=(1<<le1)    ;//mở U3
    cont&=~(1<<le1)   ;//khóa U3
}

```

```

//-----
unsigned char get_7seg(unsigned char x) //hàm chuyển mã 7 đoạn
{
    unsigned char tab_7sa[]={0xe0,0xf9,0xa4,0xb0,0x99,0x92,0x82,0xf8,0x80,0x90,0x88,
                           0x83,0xc6,0xa1,0x86,0x8e};//bảng tra mã 7 đoạn
    x=tab_7sa[x] ;//lấy mã 7 đoạn
    return(x) ;
}

```

**Ví dụ 7.17:** Thiết kế mạch đo dung tích chất lỏng làm việc như sau:

- Khi cho máy bơm chất lỏng qua bộ đo dung tích,cảm biến dung tích là bộ ghép quang điện (optical encoder)phát 100 xung 1 vòng quay tương ứng 0.5 lít.
- Một nút nhấn SW1(START/STOP) bắt đầu khởi động máy bơm và đo dung tích chất lỏng hoặc dừng hẳn máy bơm.Nút nhấn thứ hai SW2(PAUSE/CONTINUE)điều khiển tạm dừng bơm hoặc bơm tiếp.Máy bơm tối đa đến 100 lít thì tự động dừng bơm,chờ nhấn SW1 khởi động lại từ đầu.
- Liên tục hiển thị dung tích bơm được đến hàng 0.01 lit ra màn hình LCD ký tự thể hiện như sau:

Do dung tích: CENTLIT= 10000
---------------------------------

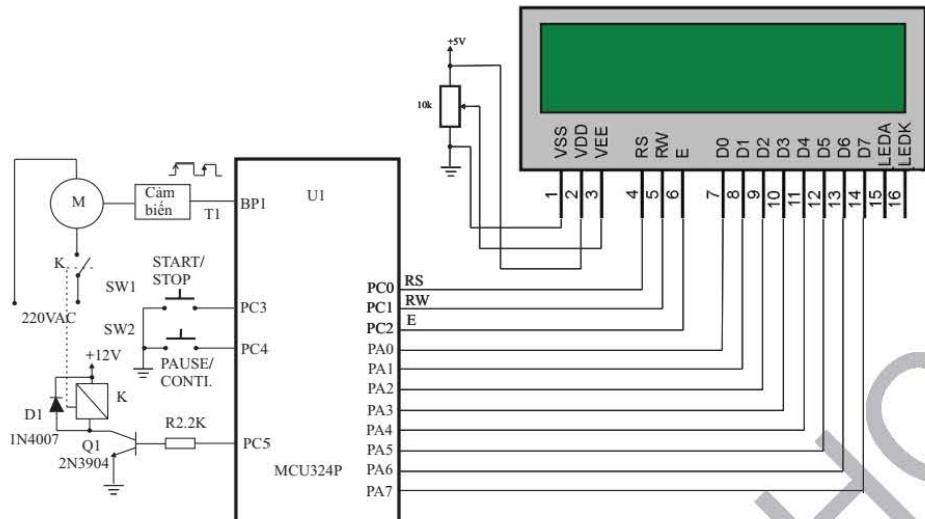
**Giải:**

Từ đặc tính cảm biến,ta suy ra 1 xung cảm biến phát ra tương ứng  $0.5/100=5\text{ml}$ .Như vậy để đếm đến tối đa 100 lít tương ứng 20,000 xung.Do đó ta sử dụng Timer1 mô thức đếm sự kiện,CTC4,xung đếm đưa vào ngõ T1(PB1),cho tác động cạnh lên và giá trị đặt OCR1A=20000.

Sơ đồ thiết kế như hình 7.43,áp dụng ví dụ 7.15.Do phải sử dụng chân T1=PB1 nhận xung từ cảm biến nên ta chuyển PortB sang PortA kết nối bus dữ liệu LCD.SW1(START/STOP)và SW2(PAUSE/CONTINUE)lần lượt kết nối PC3 và PC4.Ngõ ra PC5 lái Q1 dẫn/tắt bão hòa đóng/cắt rờ le K cấp nguồn cho máy bơm làm việc theo yêu cầu điều khiển.

❖ Ý tưởng giải thuật chương trình

1. Khởi động LCD,khởi động Timer1 mô thức CTC4,đếm xung ngoài tác động cạnh lên,nạp OCR1A=20000
2. Xóa bộ đếm hiển thị khởi động
3. Chờ nhấn SW1 đặt M\_CONT=1 chạy máy bơm
4. Đọc bộ đếm,chia 2 giá trị đếm ra c1,hiển thị số đo dung tích
5. Kiểm tra nếu cờ OCF1A=1 báo đếm tới 10000c1 dừng bơm xóa M\_CONT=0,chờ nhấn SW1 trở về bước 2,nếu cờ OCF1A=0 tiếp bước 6.
6. Kiểm tra SW,xét các trường hợp:
  - Nếu nhấn SW1 dừng bơm và chờ nhấn SW1 lần nữa trở về bước 2
  - Nếu nhấn SW2 luân phiên đảo trạng thái dừng bơm/bơm tiếp và trở về bước 4
  - Nếu không nhấn SW nào cả,trở về bước 4



Hình 7.43: Sơ đồ mạch đo&điều khiển dung tích

❖ Chương trình hợp ngữ ví dụ 7.17

```

.DEF OPD1 L=R20
.DEF OPD1 H=R21
.DEF OPD2=R22
.DEF OPD3=R23
.DEF COUNT=R18
.DEF FLAG REG=R19
.EQU OUTPORT=PORTA ;PORTA giao tiếp bus data LCD
.EQU INPORT=PINA
.EQU IOSETB=DDRA
.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 SW1=3 ;ký hiệu SW1
.EQU SW2=4 ;ký hiệu SW2
.EQU M_CONT=5 ;ký hiệu M_CONT
.EQU SW_FLG=0 ;ký hiệu cờ báo SW nhấn
.EQU NULL=$00 ;mã kết thúc
.EQU TOPA=20000 ;giá trị đặt OCR1A
.EQU BCD_BUF=0X200
.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 R16,0B00100111
      OUT CONT DR,R16 ;khai báo PC0,PC1,PC2,PC5 output,PC3,PC4 input
      LDI R16,0B00011000 ;RS=0,RW=0,E=0,M_OUT=0,điện trở kéo lên PC3,PC4
      OUT CONT,R16
      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
LDI    R16,2 ;delay 200μs

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
;khi nhập ký tự,màn hình không dịch
;ctc khởi động LCD 8 bit

RCALL INIT_LCD8 ;-----

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
LDI    R17,$80 ;con trỏ bắt đầu ở đầu dòng 1
RCALL CURS_POS ;xuất lệnh ra LCD
LDI    ZH,HIGH(MSG1<<1) ;Z trỏ đầu bằng tra MSG1
LDI    ZL,LOW(MSG1<<1)
RCALL MSG_DISP ;ghi MSG1 ra LCD
LDI    R17,$C0 ;con trỏ bắt đầu ở đầu dòng 2
RCALL CURS_POS ;xuất lệnh ra LCD
LDI    ZH,HIGH(MSG2<<1) ;Z trỏ đầu bằng tra MSG2
LDI    ZL,LOW(MSG2<<1)
RCALL MSG_DISP ;ghi MSG2 ra LCD
LDI    R16,HIGH(TOPA) ;giá trị đặt OCR1A
STS    OCR1AH,R16
LDI    R16,LOW(TOPA)
STS    OCR1AL,R16
LDI    R16,0X04 ;Timer1 mode CTC4 TOP=OCR1A
STS    TCCR1A,R16
LDI    R16,0B00000111 ;Timer1 nhận CK ngoài,kích cạnh lên
STS    TCCR1B,R16 ;
LDI    R17,0 ;xóa bộ đếm

START: LDI    R17,0 ;chuyển số nhị phân sang BCD
STS    TCNT1H,R17 ;con trỏ bắt đầu ở đầu vị trí 9 dòng 2
STS    TCNT1L,R17 ;xuất lệnh ra LCD
RCALL BIN16_BCD5DG
LDI    R17,$C8
RCALL CURS_POS

```

WAIT_SW:	RCALL NUM_DISP	;hiển thị số đo
	RCALL GET_SW	;đọc SW
	SBRS FLAG_REG,SW_FLG	;cờ SW_FLG=1 báo có SW nhấn
	RJMP WAIT_SW	;chờ nhấn SW
	CPI R17,1	;SW1 nhấn?
	BRNE WAIT_SW	;không phải đọc SW lại
	SBI CONT,M_CONT	;mở máy bơm
LOOP_CT:	LDS R20,TCNT1L	;đọc bộ đếm
	LDS R21,TCNT1H	
	LSR R21	;chia 2 số đếm
	ROR R20	
	RCALL BIN16 BCD5DG	;chuyển số nhị phân sang BCD
	LDI R17,\$C8	;con trỏ bắt đầu ở đầu vị trí 9 dòng 2
	RCALL CURS_POS	;xuất lệnh ra LCD
	RCALL NUM_DISP	;hiển thị độ rộng xung
	IN R17,TIFR1	;đọc cờ OCF1A
	SBRS R17,OCF1A	;cờ OCF1A=1 xử lý dừng bơm
	RJMP SW_CHK	;kiểm tra SW
	OUT TIFR1,R17	;xóa cờ OCF1A
	RJMP STOP_M	;xử lý dừng bơm
	RCALL GET_SW	;đọc SW
SW_CHK:	SBRS FLAG_REG,SW_FLG	;xử lý khi có SW nhấn
	RJMP LOOP_CT	;không có SW nhấn tiếp tục bơm
	CPI R17,1	;nhấn SW1?
	BRNE SW2_CHK	;kiểm tra SW2
	CBI CONT,M_CONT	;dừng bơm
STOP_M:	RCALL GET_SW	;đọc SW
	SBRS FLAG_REG,SW_FLG	
	RJMP STOP_M	
	CPI R17,1	;chờ nhấn SW1 lần nữa
	BRNE STOP_M	
	RJMP START	;trở về trạng thái khởi động
	CPI R17,2	;SW2 nhấn?
SW2_CHK:	BRNE LOOP_CT	;không phải tiếp tục bơm
	IN R17,CONT	;đọc bit điều khiển bơm
	LDI R16,(1<<M_CONT)	
	EOR R17,R16	;đảo bit tạm dừng/chạy tiếp máy bơm
	OUT CONT,R17	
	RJMP LOOP_CT	;lặp vòng đo lại

-----  
;GET\_SW đọc trạng thái SW1,SW2 có chống rung  
;Trả về R17 chứa mã SW1=1 hoặc mã SW2=2 và cờ SW\_FLG=1 nếu có SW nhấn  
;Trả về cờ SW\_FLG=0 nếu không có SW nhấn  
;Sử dụng R16,R17,cờ SW\_FLG thuộc thanh ghi FLAG\_REG  
;

GET_SW:	CBR FLAG_REG,(1<<SW_FLG);xóa cờ báo nhấn SW
BACK0:	LDI R16,50 ;kiểm tra SW nhấn 50 lần liên tục
WAIT0:	IN R17,CONT_IN
	ANDI R17,(1<<SW1) (1<<SW2);che bit SW1,SW2
	CPI R17,(1<<SW1) (1<<SW2);kiểm tra SW nhấn?
	BREQ EXIT_SW ;không nhấn thoát
	DEC R16 ;có nhấn tiếp tục
	BRNE WAIT0 ;
	PUSH R17 ;cắt mã SW
BACK1:	LDI R16,50 ;kiểm tra sw nhấn 50 lần liên tục

```

WAIT1:    IN      R17,CONT_IN
          ANDI   R17,(1<<SW1)|(1<<SW2)
          CPI    R17,(1<<SW1)|(1<<SW2)
          BRNE  BACK1 ;chờ nhả SW
          DEC    R16
          BRNE  WAIT1
          POP    R17      ;phục hồi mã SW
          CPI    R17,(1<<SW2) ;SW1=0 nhấn,SW2=1 không nhấn
          BRNE  SW2_CODE ;không phải kiểm tra mã SW2
          LDI    R17,1    ;gán giá trị mã SW1
          RJMP  SET_FLG ;báo cờ nhấn SW
SW2_CODE: CPI    R17,(1<<SW1) ;SW2=0 nhấn,SW1=1 không nhấn
          BRNE  EXIT_SW ;không phải thoát
          LDI    R17,2    ;gán giá trị mã SW2
SET_FLG:  SBR   FLAG_REG,(1<<SW_FLG);đặt cờ báo nhấn SW
EXIT_SW: RET
;

;BIN16_BCD5DG chuyển đổi số nhị phân 16 bit sang số BCD 5 digit
;Inputs: OPD1_H=R21:OPD1_L=R20 chứa số nhị phân 16 bit
;Outputs: BCD_BUF:BCD_BUF+4:địa chỉ SRAM chứa 5 digit BCD từ cao đến thấp
;Sử dụng R17,COUNT,X,etc DIV16_8
;

BIN16_BCD5DG:
          LDI    XH,HIGH(BCD_BUF);X trỏ địa chỉ đầu buffer BCD
          LDI    XL,LOW(BCD_BUF)
          LDI    COUNT,5 ;đếm số byte bộ nhớ
          LDI    R17,0X00 ;nạp giá trị 0
LOOP_CL:  ST     X+,R17 ;xóa buffer bộ nhớ
          DEC    COUNT ;đếm đủ 5 byte
          BRNE  LOOP_CL
          LDI    OPD2,10 ;nạp SC
DIV_NXT:  RCALL DIV16_8 ;chia số nhị phân 16 bit cho 10
          ST     -X,OPD3 ;cắt số dư vào buffer
          CPI    OPD1_L,0 ;thương số=0?
          BRNE  DIV_NXT ;khác 0 chia tiếp
          RET
;

;NUM_DISP hiển thị 5 ký tự địa chỉ đầu BCD_BUF trong SRAM
;(BCD_BUF)=digit cao nhất,(BCD_BUF+4)=digit thấp nhất
;Sử dụng R16,R17,COUNT,X,etc DELAY_US,etc OUT_LCD
;

NUM_DISP: LDI    COUNT,5      ;hiển thị 5 ký tự
          LDI    XH,HIGH(BCD_BUF);X trỏ địa chỉ đầu buffer
          LDI    XL,LOW(BCD_BUF)
DISP_NXT: LD     R17,X+ ;lấy số BCD từ buffer,tăng địa chỉ buffer kế tiếp
          LDI    R16,0X30 ;chuyển sang mã ASCII
          ADD    R17,R16
          LDI    R16,1    ;chờ 100μs
          RCALL DELAY_US
          SBI    CONT,RS ;RS=1 ghi data hiển thị LCD
          RCALL OUT_LCD ;ghi ký tự ra LCD
          DEC    COUNT ;hiển thị đủ 5 ký tự
          BRNE  DISP_NXT
          RET
;

```

```

;DIV16_8 chia số nhị phân 16 bit OPD1 cho 8 bit OPD2
;Input: OPD1_H,OPD1_L= SBC(GPR16-31)
;      OPD2=SC(GPR0-31)
;Output:OPD1_H,OPD1_L=thương số
;      OPD3=DS(GPR0-31)
;Sử dụng COUNT(GPR16-31)
;-----
DIV16_8: LDI    COUNT,16      ;COUNT=đếm 16
          CLR    OPD3        ;xóa dư số
SH_NXT:  CLC              ;C=0=bit thương số
          LSL    OPD1_L       ;dịch trái SBC L,bit0=C=thương số
          ROL    OPD1_H       ;quay trái SBC H,C=bit7
          ROL    OPD3        ;dịch bit7 SBC H vào dư số
          BRCS  OV_C         ;tràn bit C=1,chia được
          SUB   OPD3,OPD2     ;trừ dư số với số chia
          BRCC  GT_TH        ;C=0 chia được
          ADD   OPD3,OPD2     ;C=1 không chia được,không trừ
          RJMP  NEXT         ;đến bước sau
OV_C:   SUB   OPD3,OPD2     ;trừ dư số với số chia
GT_TH:  SBR   OPD1_L,1      ;chia được,thương số=1
NEXT:   DEC   COUNT        ;đếm số lần dịch SBC
          BRNE  SH_NXT       ;chưa đủ tiếp tục dịch bit
          RET
;-----
;CURS_POS đặt con trỏ tại vị trí có địa chỉ trong R17
;Input: R17=$80 -$8F dòng 1,$C0-$CF dòng 2
;      R17= địa chỉ vị trí con trỏ
;Sử dụng R16,ctc DEAY_US,OUT_LCD
;-----
CURS_POS: LDI   R16,1        ;chờ 100µs
          RCALL DELAY_US
          CBI   CONT,RS       ;RS=0 ghi lệnh
          RCALL OUT_LCD
          RET
;-----
;MSG_DISP hiển thị chuỗi ký tự kết thúc bằng mã NULL đặt trong Flash ROM
;Input: Z chứa địa chỉ đầu chuỗi ký tự
;Output: hiển thị chuỗi ký tự ra LCD tại vị trí con trỏ hiện hành
;Sử dụng R16,R17,ctc DELAY_US,OUT_LCD
;-----
MSG_DISP: LPM   R17,Z+        ;lấy mã ASCII ký tự từ Flash ROM
          CPI   R17,NULL      ;kiểm tra ký tự kết thúc
          BREQ  EXIT_MSG      ;ký tự NULL thoát
          LDI   R16,1           ;chờ 100µs
          RCALL DELAY_US
          SBI   CONT,RS       ;RS=1 ghi data hiển thị LCD
          RCALL OUT_LCD      ;ghi mã ASCII ký tự ra LCD
          RJMP  MSG_DISP      ;tiếp tục hiển thị ký tự
EXIT_MSG: RET
;-----
;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]-----
INIT_LCD8: LDI R16,1 ;chờ 100µs
            RCALL DELAY_US
            CBI CONT,RS ;RS=0 ghi lệnh
            MOV R17,R18 ;R18=Function set
            RCALL OUT_LCD ;
            LDI R16,1 ;chờ 100µs
            RCALL DELAY_US
            CBI CONT,RS ;RS=0 ghi lệnh
            MOV R17,R19 ;R19=Clear display
            RCALL OUT_LCD ;
            LDI R16,20 ;chờ 2ms sau lệnh Clear display
            RCALL DELAY_US
            CBI CONT,RS ;RS=0 ghi lệnh
            MOV R17,R20 ;R20=Display on/off control
            RCALL OUT_LCD ;
            LDI R16,1 ;chờ 100µs
            RCALL DELAY_US
            CBI CONT,RS ;RS=0 ghi lệnh
            MOV R17,R21 ;R21=Entry mode set
            RCALL OUT_LCD
            RET

;-----[OUT_LCD]-----
;OUT_LCD ghi mã lệnh/data ra LCD
;Input: R17 chứa mã lệnh/data
;-----[OUT_LCD]-----
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]-----
;DELAY_US tạo thời gian trễ =R16x100µs(Fosc=8Mhz)
;Input:R16 hệ số nhân thời gian trễ 1 đến 255
;-----[DELAY_US]-----
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

;-----[MSG]-----
MSG1: .DB "Đo dung tích:", $00
MSG2: .DB "CENTLIT=", $00

```

#### ❖ Chương trình C ví dụ 7.17

Phần chương trình C ví dụ 7.17 dành cho người đọc tự soạn thảo.

#### ❖ Câu hỏi ôn tập

1. Khai báo Timer0/1 mô thức đếm sự kiện bằng cách nào?
2. Các chân I/O nào là ngõ vào xung CLK Timer0/1 trong mô thức đếm sự kiện?

3. Có thể đặt hệ số chia N cho xung CLK đếm từ ngoài không,tại sao?
4. Viết một đoạn lệnh khởi động Timer1 làm bộ đếm xung ngoài,kích cạnh lên, có đặt trước đến 20000 .
5. Trong ví dụ 7.16,phần đầu chương trình có lệnh:

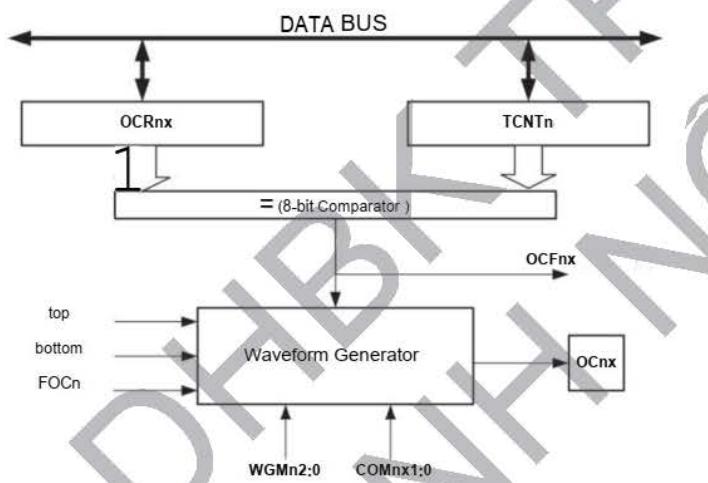
...  
 CBI DDRB,0 ;khai báo PB0=T0 input  
 Có thể bỏ lệnh này được không,tại sao?

## 7.7 Lập trình tạo sóng với Timer0,Timer2

Ở các phần trên ta chỉ sử dụng mô thức NOR,CTC và cờ báo tương ứng để tạo xung ngõ ra trên 1 bit Port bằng phần mềm.Trong cấu hình các Timer đều có khối tạo sóng ngõ ra có thể lập trình tạo thay đổi trạng thái logic các ngõ ra bằng phần cứng,**mà không cần kiểm tra các cờ báo TOVn,OCFnx**(các cờ báo lúc này để báo ngắn,hoặc để cập nhật dữ liệu).Áp dụng tạo sóng ngõ ra thay đổi tần số,độ rộng ứng dụng trong đo đặc,điều khiển....Trong phần này ta sẽ khảo sát trước phần tạo sóng với Timer0 và Timer2 do hoạt động hoàn toàn tương tự nhau.

### 7.7.1 Sơ đồ khối so sánh ngõ ra

Việc tạo sóng ngõ ra bằng phần cứng thực hiện thông qua khối so sánh ngõ ra như hình 7.44.

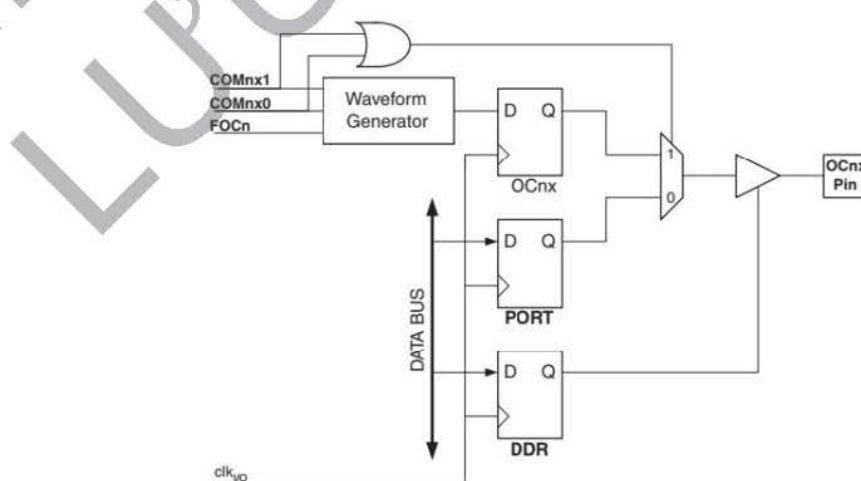


**Hình 7.44:** Sơ đồ khối so sánh ngõ ra

Khi bộ đếm TCNTn đếm đến giá trị bằng giá trị đặt trong thanh ghi OCRnx,đạt kết quả so sánh,cờ OCFnx=1 kích khởi bộ tạo sóng(Waveform Generator)làm việc.Tùy theo mô thức cài đặt bằng các bit WGMn2:0,ngõ ra chân Port OCnx sẽ thay đổi trạng thái logic theo cài đặt các bit COMnx1:0.

Với Timer0 2 chân OC0A(PB3)và OC0B(PB4)tương ứng kênh A và B.Với Timer2 chân OC2A (PD7)và OC2B(PD6)tương ứng kênh A và B.

Hình 7.45 minh họa hoạt động của khối tạo sóng ngõ ra chân OCnx.



**Hình 7.45:** Sơ đồ khối hoạt động tạo sóng ngõ ra chân OCnx

Các bit COMnx1:COMnx0 điều khiển cấm/cho phép(chọn 0/1 theo bộ MUX)ngõ ra khối tạo sóng ghép ra chân Port OCnx, đồng thời phải khai báo chân OCnx là ngõ ra qua thanh ghi DDR của Port. Trường hợp cấm phát sóng ngõ ra, chân OCnx trở thành I/O port bình thường.

Ngoài trường hợp đạt kết quả so sánh ngõ ra, bit FOCn cũng điều khiển tác động trực tiếp ngõ ra OCnx tương tự như đạt kết quả so sánh ngõ ra, mặc dù thực tế chưa đạt. Phần này sẽ trình bày chi tiết sau.

### 7.7.2 Chọn mô thức tạo sóng và trạng thái logic chân OCnx

Trong phần này ta sẽ xem chi tiết chức năng các bit WGMn2:WGMn0 và COMnx1:COMnx0.

#### 1. Bit WGMn2(bit 3 TCCRnB), WGMn1, WGMn0(bit 1,0 TCCRnA)

Các bit WGMn2:WGMn0 chọn mô thức hoạt động Timern như Bảng 7.13.

**Bảng 7.13:** Các mô thức tạo sóng Timer0, Timer2

Mô thức	WGMn2	WGMn1	WGMn0	Hoạt động	TOP	Cập nhật OCRnx	TOVn=1 tại điểm
0	0	0	0	Bình thường(NOR)	0xFF	Tức thời	MAX
1	0	0	1	PWM hiệu chỉnh pha (PCPWM)	0xFF	TOP	BOTTOM
2	0	1	0	Kết quả so sánh ngõ ra (CTC)	OCRnA	Tức thời	MAX
3	0	1	1	PWM nhanh(FPWM)	0xFF	BOTTOM	MAX
4	1	0	0	Dự trữ	-	-	-
5	1	0	1	PWM hiệu chỉnh pha (PCPWM)	OCRnA	TOP	BOTTOM
6	1	1	0	Dự trữ	-	-	-
7	1	1	1	PWM nhanh(FPWM)	OCRnA	BOTTOM	TOP

- MAX=0xFF, BOTTOM=0x00

- Cột TOP thể hiện giá trị định bộ đếm

- Cột Cập nhật OCRnx chỉ thời điểm thanh ghi OCRnx cập nhật giá trị mới

- Cột TOVn=1 tại điểm chỉ thời điểm cờ TOVn=1

Từ Bảng 7.13, ta có thể phân nhóm 4 mô thức chính của Timern là NOR, CTC, FPWM, PCPWM.

Trừ mô thức 4,6 dự trữ, các mô thức của Timern đều có thể tạo sóng ngõ ra.

#### 2. Bit COMnA1:COMnA0(bit 7,6 TCCRnA)

Các bit này cài đặt trạng thái logic ngõ ra OCnA khi đạt kết quả so sánh, tùy vào mô thức không PWM(NOR, CTC), FPWM hay PCPWM theo mô tả ở Bảng 7.14 đến 7.16

**Bảng 7.14:** Trạng thái ngõ ra OCnA mô thức NOR, CTC

COMnA1	COMnA0	Mô tả
0	0	I/O bình thường, OCnA không kết nối
0	1	Đảo bit OCnA khi đạt kết quả so sánh
1	0	Xóa OCnA=0 khi đạt kết quả so sánh
1	1	Đặt OCnA=1 khi đạt kết quả so sánh

**Bảng 7.15:** Trạng thái ngõ ra OCnA mô thức FPWM

COMnA1	COMnA0	Mô tả
0	0	I/O bình thường, OCnA không kết nối
0	1	WGMn2=0: I/O bình thường, OCnA không kết nối WGMn2=1: đảo bit OCnA khi đạt kết quả so sánh
1	0	Xóa OCnA=0 khi đạt kết quả so sánh Đặt OCnA=1 ở BOTTOM(trạng thái không đảo)
1	1	Đặt OCnA=1 khi đạt kết quả so sánh Xóa OCnA=0 ở BOTTOM(trạng thái đảo)

**Bảng 7.16:** Trạng thái ngõ ra OCnA mô thức PCPWM

COMnA1	COMnA0	Mô tả
0	0	I/O bình thường,OCnA không kết nối
0	1	WGMn2=0: I/O bình thường,OCnA không kết nối WGMn2=1: đảo bit OCnA khi đạt kết quả so sánh
1	0	Xóa OCnA=0 khi đếm lên đạt kết quả so sánh Đặt OCnA=1 khi đếm xuống đạt kết quả so sánh
1	1	Đặt OCnA=1 khi đếm lên đạt kết quả so sánh Xóa OCnA=0 khi đếm xuống đạt kết quả so sánh

### 3. Bit COMnB1:COMnB0(bit 5,4 TCCRnA)

Các bit này cài đặt trạng thái logic ngõ ra OCnB khi đạt kết quả so sánh,tùy vào mô thức không PWM(NOR,CTC), FPWM hay PCPWM theo mô tả ở Bảng 7.17 đến 7.19

Bảng 7.17: Trạng thái ngõ ra OCnB mô thức NOR,CTC

COMnB1	COMnB0	Mô tả
0	0	I/O bình thường,OCnB không kết nối
0	1	Đảo bit OCnB khi đạt kết quả so sánh
1	0	Xóa OCnB=0 khi đạt kết quả so sánh
1	1	Đặt OCnB=1 khi đạt kết quả so sánh

Bảng 7.18: Trạng thái ngõ ra OCnB mô thức FPWM

COMnB1	COMnB0	Mô tả
0	0	I/O bình thường,OCnB không kết nối
0	1	Dự trữ
1	0	Xóa OCnB=0 khi đạt kết quả so sánh Đặt OCnB=1 ở BOTTOM(trạng thái không đảo)
1	1	Đặt OCnB=1 khi đạt kết quả so sánh Xóa OCnB=0 ở BOTTOM(trạng thái đảo)

Bảng 7.19: Trạng thái ngõ ra OCnB mô thức PCPWM

COMnB1	COMnB0	Mô tả
0	0	I/O bình thường,OCnB không kết nối
0	1	Dự trữ
1	0	Xóa OCnB=0 khi đếm lên đạt kết quả so sánh Đặt OCnB=1 khi đếm xuống đạt kết quả so sánh
1	1	Đặt OCnB=1 khi đếm lên đạt kết quả so sánh Xóa OCnB=0 khi đếm xuống đạt kết quả so sánh

Ví dụ 7.18: Tìm các giá trị nạp cho các thanh ghi tương ứng:

- (a) Tạo chuỗi xung vuông đối xứng ngõ ra OC0A, chu kỳ  $250\mu s$
- (b) Timer2 chạy mô thức FPWM,OCR2A=99,ngõ ra OC2B=1,OC2A đảo bit khi đạt kết quả so sánh tương ứng.Chi thời gian 1 xung đếm  $8\mu s$ .

Giải:

- (a) Để tạo chuỗi xung vuông đối xứng chu kỳ  $T=250\mu s$ , độ rộng xung  $125\mu s$ , ta chọn Timer0 làm việc:
  - Mô thức CTC2 ,WGM02:WGM00=010
  - Giá trị nạp OCR1A=124,hệ số chia N=8 tương ứng 1 xung CLKT0= $1\mu s$ ,CS02:CS00=010
  - Đảo bit ngõ ra OC0A mỗi lần đạt kết quả so sánh, COM0A1:COM0A0=01
 Giá trị nạp TCCR0A=0b01000010 và TCCR0B=0b00000010
- (b) Timer2 chạy:
  - Mô thức FPWM,OCR2A=99,mô thức FPWM7 WGM22:WGM20=111
  - Ngõ ra OC2B=1 khi đạt kết quả so sánh ,COM2B1:COM2B0=11

- Ngõ ra OC2A đảo bit khi đạt kết quả so sánh ,COM2A1:COM2A0=01
  - Thời gian 1 xung đếm 8 $\mu$ s ,CS22:CS20=011
- Giá trị nạp TCCR2A=0b01110011 và TCCR0B=0b00001011

### 7.7.3 Lập trình Timer0/2 mô thức NOR, CTC tạo sóng ngõ ra

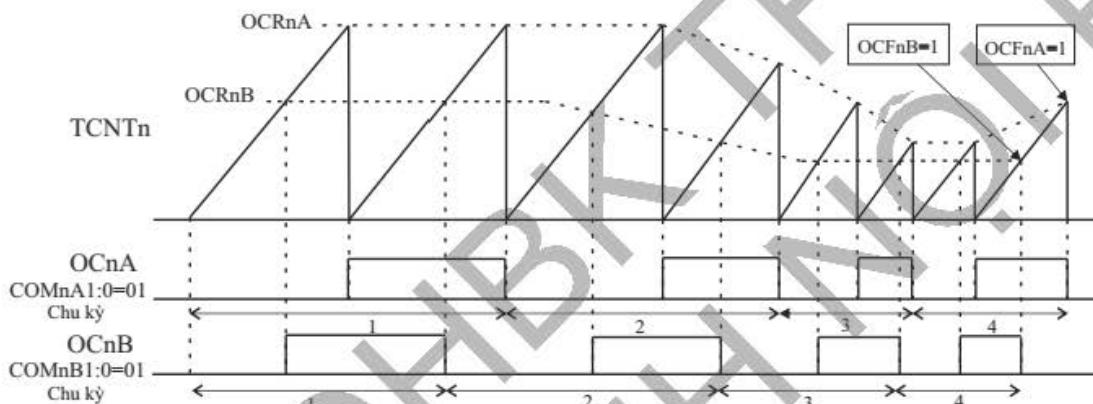
Như đã phân tích ở trên, mặc dù mô thức NOR vẫn có thể tạo sóng ngõ ra, tuy nhiên ta phải mất thời gian dừng Timer để nạp lại giá trị đặt trước cho bộ đếm, nên không thích hợp trong các ứng dụng tạo sóng!

Mô thức CTC thuận lợi hơn do không cần nạp giá trị đặt trước cho bộ đếm. Giá trị đặt trước có thể cập nhật tức thời bất kỳ thời điểm nào, chỉ cần cập nhật thanh ghi OCRnx. Mô thức CTC tạo sóng ngõ ra ứng dụng tạo xung vuông tần số thay đổi PFM(Pulse Frequency Modulation), xung đánh dấu, kích khởi định thì tạo trễ, đếm sự kiện bên ngoài....

Theo bảng 7.14 và Bảng 7.17, ta có thể cài đặt ngõ ra OCnx một trong 4 trạng thái trong mô thức NOR, CTC:

- COMnx1:COMnx0=00: OCnx=I/O không tạo sóng ra
- COMnx1:COMnx0=01: Đảo bit OCnx khi TCNTn=OCRnA/B (đếm lên)
- COMnx1:COMnx0=10: OCnx=0 khi TCNTn=OCRnA/B
- COMnx1:COMnx0=10: OCnx=1 khi TCNTn=OCRnA/B

Hình 7.46 minh họa việc tạo sóng ngõ ra OCnx khi Timer chạy mô thức CTC, đảo bit OCnx.



Hình 7.46: Mô thức CTC đảo bit tạo sóng ngõ ra OCnx

Tần số xung ngõ ra được tính tương ứng thời gian 2 lần đảo bit OCnx:

$$f_o = \frac{F_{osc}}{2N \times (OCRnA + 1)} \quad (7.2)$$

- **Lưu ý:**
- Phải khai báo chân OCnx là ngõ ra qua thanh ghi DDRx
  - Ngõ ra OCnx chỉ chuyển trạng thái khi đếm lên đạt TCNTn=OCRnx
  - Ngõ ra OCnx chuyển trạng thái hoàn toàn bằng phần cứng tùy theo cài đặt COMnx1:0

Ta viết lại ví dụ 7.7 sử dụng mô thức CTC tạo sóng ngõ ra.

**Ví dụ 7.19:** Lập lại ví dụ 7.7, tạo chuỗi xung vuông đối xứng chu kỳ 24 $\mu$ s xuất ra PB3.

**Giải:**

PB3 chính là chân OC0A, ngõ ra bộ tạo sóng Timer0 kênh A. Do đó ta chỉ cần khai báo lại thanh ghi TCCR0A có bit COM0A1:COM0A0=01 đảo bit ngõ ra và không cần kiểm tra cờ OCF0A.

#### ❖ Chương trình hợp ngữ ví dụ 7.19

```
.EQU P_OUT=3 ;ký hiệu OC0A
.ORG 0
RJMP MAIN
.ORG 0X40
MAIN: LDI R16,HIGH(RAMEND) ;đưa stack lên đỉnh SRAM
      OUT SPH,R16
```

```

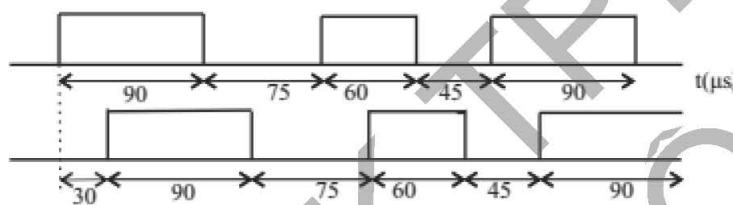
LDI    R16,LOW(RAMEND)
OUT   SPL,R16
LDI    R16,(1<<P_OUT)      ;đặt PB3 output
OUT   DDRB,R16
LDI    R17,$5F              ;giá trị so sánh
OUT   OCR0A,R17
LDI    R17,0B01000010        ;Timer0 mode CTC đảo bit OC0A
OUT   TCCR0A,R17
LDI    R17,0X01              ;Timer0 chạy,hệ số chia N=1
OUT   TCCR0B,R17
START: RJMP  START          ;lặp vòng lại

```

- So sánh ví dụ 7.7, ta thấy không cần phải kiểm tra cờ OCF0A báo TCNT0=OCR0A để thực hiện việc đảo bit bằng phần mềm, thay vào đó phần cứng đã thực hiện việc đảo bit ngõ ra OCR0A. Do đó, ta có thể tăng tần số sóng ngõ ra.

Theo công thức (7.2), tần số ngõ ra cực đại  $f_{max} = F_{osc}/2$ .

**Ví dụ 7.20:** Tạo 2 chuỗi xung ra có dạng như hình 7.47.



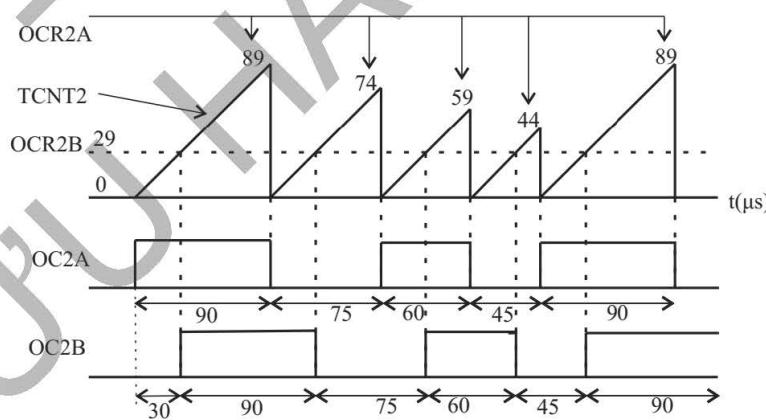
Hình 7.47: Định dạng hai chuỗi xung ví dụ 7.47

**Giải:**

Ta sử dụng Timer2 mô thức CTC2, tạo sóng ngõ ra 2 chân OC2A(PD7) và OC2B(PD6) với giá trị đặt trước cho thanh ghi OCR2A và OCR2B như hình 7.48, với hệ số chia  $N=8$  tương ứng  $CLKT2=1\mu s$ .

Chu kỳ chuỗi xung 1 gồm 4 thời đoạn: mức 1 90μs, mức 0 75μs, mức 1 60μs, mức 0 45μs. Ta chọn đảo bit ngõ ra OC2A, khởi động đặt OCR2A=20 (phải nhỏ hơn OCR2B) để khi bộ đếm đạt kết quả so sánh kênh A OC2A=1, và lần lượt thay đổi giá trị đặt trước cho OCR2A 4 giá trị 89, 74, 59, 44 khi cờ OCF2A=1.

Chu kỳ chuỗi xung 2 cũng giống như chuỗi xung 1, chỉ dài chậm hơn 30μs. Ta chọn đảo bit ngõ ra OC2B xuất xung 2, ban đầu đặt OC2B=0 và giá trị đặt trước OCR2B=29.



Hình 7.48: Sử dụng Timer2 mô thức CTC, đảo bit OC2A/B

#### ❖ Chương trình hợp ngữ ví dụ 7.20

```

.EQU P_1=7           ;ký hiệu OC2A
.EQU P_2=6           ;ký hiệu OC2B
.ORG 0
RJMP MAIN
.ORG 0X40

```

MAIN:	LDI R16,HIGH(RAMEND)	;đưa stack lên đỉnh SRAM
	OUT SPH,R16	
	LDI R16,LOW(RAMEND)	
	OUT SPL,R16	
	LDI R16,(1<<P_1) (1<<P_2)	;đặt P_1,P_2 output
	OUT DDRD,R16	
	CBI PORTD,P_1	;P_1=0
	CBI PORTD,P_2	;P_2=0
	LDI R16,4	;bộ đếm thời đoạn xung
	LDI ZH,HIGH(DAT<<1)	;Z trả địa chỉ đầu bảng giá trị nạp OCR2A
	LDI ZL,LOW(DAT<<1)	
	LDI R17,20	;giá trị khởi động cho OCR2A
	STS OCR2A,R17	
	LDI R17,29	;giá trị đặt trước cho OCR2B
	STS OCR2B,R17	
	LDI R17,0B01010010	;Timer2 mode CTC đảo bit OC2A/B
	STS TCCR2A,R17	
	LDI R17,0X02	;Timer2 chạy,hệ số chia N=8
	STS TCCR2B,R17	
START:	IN R17,TIFR2	;đọc cờ OCF2A
	SBRS R17,OCF2A	;cờ OCF2A=1 cập nhật OCR2A
	RJMP START	;chờ cờ OCF2A=1
	OUT TIFR2,R17	;xóa cờ OCF2A
	LPM R17,Z+	;cập nhật giá trị tiếp theo OCR2A
	STS OCR2A,R17	
	DEC R16	;đếm đủ 4 thời đoạn lập vòng lại
	BRNE START	
	LDI R16,4	;nạp lại bộ đếm thời đoạn
	LDI ZH,HIGH(DAT<<1)	;nạp lại địa chỉ đầu bảng tra giá trị đặt trước
	LDI ZL,LOW(DAT<<1)	
	RJMP START	;lặp vòng lại
;	DAT: .DB 89,74,59,44	;bảng tra các giá trị đặt trước cho OCR2A

- Đầu chương trình ta xóa ngõ ra OC2A/B và nạp giá trị khởi động OCR2A=20. Do đó thời đoạn này chỉ xảy ra kết quả so sánh ngõ ra kênh A và đảo bit OC2A=1,OC2B=0. Đến thời đoạn đầu tiên nạp OCR2A=89 mới bắt đầu xảy ra kết quả so sánh cả 2 kênh A và B.

- Kiểm tra cờ OCF2A=1 chỉ để cập nhật thời đoạn xung kế tiếp cho OCR2A. Trong chương 10 sử dụng ngắt so sánh kết quả ngõ ra Timer2 kênh A,khi cờ OCF2A=1 tạo ngắt tương ứng,phần cứng tự động xóa cờ OCF2A và trình phục vụ ngắt sẽ cập nhật thời đoạn xung kế tiếp,chương trình chính không cần thực hiện các công việc này!

#### ❖ Chương trình C ví dụ 7.20

```
#include <avr/io.h>
#define outport PORTD // định nghĩa PORTD=output
const char p_1=7,p_2=6 // ký hiệu vị trí bit các ngõ ra
const char dat[]={89,74,59,44};//bảng tra giá trị đặt trước OCR2B
unsigned char i;
int main() // bắt đầu chương trình
{
    DDRD |=(1<<p_1)|(1<<p_2); //khai báo p_1,p_2 output
    outport=0x00 //xóa các ngõ ra
    OCR2A=20 //giá trị khởi động OCR2A
    OCR2B=29 //giá trị đặt OCR2B
```

```

TCCR2A=0x52          ;//Timer2 mode CTC2,OC2A/B đảo bit
TCCR2B=0x02          ;//Timer2 hệ số chia N=8
while(1)              //lặp vòng vô hạn
{
    for(i=0;i<=3;i++)
    {
        while(!(TIFR2&(1<<OCF2A))) ;//chờ cờ OCF2A=1?
        TIFR2 |=(1<<OCF2A)         ;//xóa cờ OCF2A
        OCR2A=dat[i]               ;//nạp giá trị đặt trước kế tiếp
    }
}
}

```

### ❖ Bit FOCnA,FOCnB

Trong phần này ta sẽ xem chi tiết chức năng các bit FOCnA,FOCnB( bit 7,6 thanh ghi TCCRNb).

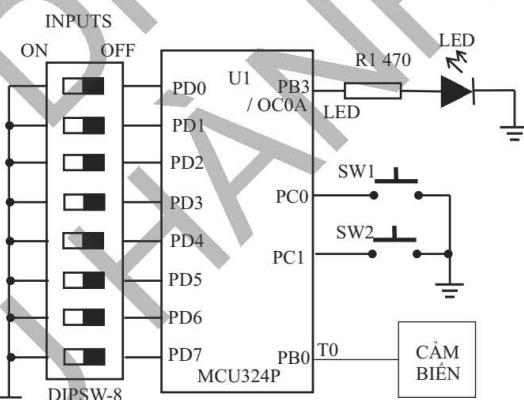
Bit FOCnA,FOCnB chỉ có tác động đối với mô thức NOR,CTC.Khi đặt FOCnA/B=1,ngõ ra OCnA/B sẽ chuyển trạng thái theo cài đặt các bit COMnx1:COMnx0,giống như trường hợp đặt kết quả so sánh ngõ ra kênh A/B(mặc dù thực tế chưa xảy ra).Trường hợp điều khiển này gọi là ép so sánh ngõ ra (Force Output Compare),không ảnh hưởng đến cờ OCFnx và bộ đếm TCNTx.

➤ **Lưu ý:** - Trường hợp  $TCNTn=OCRnx=0x00$ ,ngõ ra OCnx sẽ cập nhật trạng thái tức thời theo cài đặt COMnx1:COMnx0.

**Ví dụ 7.21:** Hình 7.49 là sơ đồ mạch đếm số người vào phòng có đặt trước.Cảm biến nhận dạng phát 1 xung mỗi khi có người vào.Cài đặt giới hạn số người vào phòng bằng DIPSW\_8 ở PortD.Khi nhấn SW1 mạch reset và bắt đầu đếm số người vào phòng cho đến số lượng bằng giá trị đặt ở DIPSW\_8,sẽ bật LED sáng đóng cửa và dừng đếm.Nhấn SW2 làm sáng/tối LED tương ứng đóng/mở cửa,cho dù mạch đang đếm hoặc dừng.Viết chương trình thực hiện các công đoạn như mô tả.

**Giải:**

Ta sử dụng Timer0 mô thức đếm sự kiện CTC2,xung CK ngoài từ cảm biến đếm kích cảnh xuống.



Hình 7.49: Sơ đồ mạch ví dụ 7.21

Giá trị đặt trước cho OCROA bằng giá trị đặt từ DIPSW\_8 trừ 1.

Sử dụng chương trình con GET\_SW đọc SW1 và SW2:

Khi nhấn SW1 thực hiện các bước:

- Đọc giá trị đặt trước nạp cho OCROA

- Khởi động Timer0 chạy mô thức đếm sự kiện CTC2,xung kích cảnh xuống,COM0A1:0=11 đặt OC0A=1 khi bộ đếm tới giá trị đặt.

- Trong khi đếm, chờ cờ OCF0A=1 dừng Timer0 và lập vòng lại từ đầu, hoặc kiểm tra SW2 nhấn gọi chương trình con tác động ép OC0A đảo bit, trở về đếm như cũ.

Khi nhấn SW2 gọi chương trình con thực hiện các bước:

- Cài đặt lại COM0A1:0=01 đảo bit OC0A

- Đặt FOC0A=1 ép OC0A đảo bit

❖ Chương trình hợp ngữ ví dụ 7.21

```

        .DEF FLAG REG=R18      ;thanh ghi chứa cờ báo SW nhấn
        .EQU CONT_IN=PINC      ;
        .EQU LED=3              ;ký hiệu OC0A
        .EQU SW FLG=0            ;ký hiệu cờ báo có SW nhấn
        .EQU SW1=0
        .EQU SW2=1
        .ORG 0
        RJMP MAIN
        .ORG 0X40

MAIN:   LDI R16,HIGH(RAMEND) ;đưa stack lên đỉnh SRAM
        OUT SPH,R16
        LDI R16,LOW(RAMEND)
        OUT SPL,R16
        LDI R16,(1<<LED)      ;đặt LED output
        OUT DDRB,R16
        CBI PORTB,LED           ;LED=0
        LDI R17,0                ;PortC input
        OUT DDRC,R17
        LDI R17,0X03             ;điện trở kéo lên PC0,PC1
        OUT PORTC,R17
        LDI R17,0                ;PortD input
        OUT DDRD,R17
        LDI R17,0XFF             ;điện trở kéo lên PortD
        OUT PORTD,R17
        START:  LDI R17,0          ;Timer0 dừng
        OUT TCCR0B,R17
        RCALL GET_SW
        SBRS FLAG_REG,SW_FLG    ;cờ SW_FLG=1 báo có SW nhấn
        RJMP START
        CPI R17,1                ;SW1 nhấn?
        BRNE SW2_CHK             ;kiểm tra SW2
        IN R17,PIND               ;đọc giá trị đặt
        SUBI R17,1                ;OCR0A=gia trị đặt -1
        OUT OCR0A,R17
        LDI R17,0B11000010         ;Timer0 mode CTC2,OC0A=1 khi so sánh bằng
        OUT TCCR0A,R17
        LDI R17,6
        OUT TCCR0B,R17

WAIT_OCF0A:
        SBIS TIFR0,OCF0A          ;chờ cờ OCF0A=1
        RJMP WAIT_SW
        SBI TIFR0,OCF0A          ;xóa cờ OCF0A
        RJMP START
        ;lặp vòng lại từ đầu
        ;đọc SW
        ;cờ SW_FLG=1 báo có SW nhấn
        ;quay về kiểm tra cờ OCF0A
        ;SW2 nhấn?
        ;không,quay về kiểm tra cờ OCF0A
        ;gọi ctc vào mode ép OC0A
        ;Timer0 thoát khỏi mode ép OC0A

WAIT_SW: RCALL GET_SW
        SBRS FLAG_REG,SW_FLG    ;cờ SW_FLG=1 báo có SW nhấn
        RJMP WAIT_OCF0A
        CPI R17,2                ;SW2 nhấn?
        BRNE WAIT_OCF0A
        RCALL SW2_FORCE
        LDI R17,6
        OUT TCCR0B,R17
        RJMP WAIT_OCF0A
        ;quay về kiểm tra cờ OCF0A
        ;SW2 nhấn?

SW2_CHK: CPI R17,2

```

```

BRNE START ;không, lập vòng lại từ đầu
RCALL SW2_FORCE ;gọi ctc vào mode ép OC0A
RJMP START

;-----;
SW2_FORCE: LDI R17,0B01000010 ;Timer0 mode CTC2, đảo bit OC0A
    OUT TCCR0A,R17
    LDI R17,0B10000110 ;Timer0 mode FOC0A=1, ép OC0A đảo bit
    OUT TCCR0B,R17
    RET

;-----;
;GET_SW đọc trạng thái SW1,SW2 có chống rung
;Trả về mã SW1=1 hoặc mã SW2=2 và cờ SW_FLG=1 nếu có SW nhấn
;Trả về cờ SW_FLG=0 nếu không có SW nhấn
;Sử dụng R16,R17,cờ SW_FLG thuộc thanh ghi FLAG_REG
;

GET_SW:    CBR    FLAG_REG,(1<<SW_FLG);xóa cờ báo nhấn SW
BACK0:     LDI    R16,50      ;kiểm tra SW nhấn 50 lần liên tục
WAIT0:     IN     R17,CONT_IN
            ANDI   R17,(1<<SW1)|(1<<SW2);che bit SW1,SW2
            CPI    R17,(1<<SW1)|(1<<SW2);kiểm tra SW nhấn?
            BREQ   EXIT_SW    ;không nhấn thoát
            DEC    R16        ;có nhấn tiếp tục
            BRNE   WAIT0     ;
            PUSH   R17        ;cất mã SW
BACK1:     LDI    R16,50      ;kiểm tra sw nhả 50 lần liên tục
WAIT1:     IN     R17,CONT_IN
            ANDI   R17,(1<<SW1)|(1<<SW2)
            CPI    R17,(1<<SW1)|(1<<SW2)
            BRNE   BACK1     ;chờ nhả SW
            DEC    R16
            BRNE   WAIT1     ;
            POP    R17        ;phục hồi mã SW
            CPI    R17,(1<<SW2) ;SW1=0 nhấn, SW2=1 không nhấn
            BRNE   SW2_CODE   ;không phải kiểm tra mã SW2
            LDI    R17,1       ;gán giá trị mã SW1
            RJMP   SET_FLAG  ;báo cờ nhấn SW
SW2_CODE:  CPI    R17,(1<<SW1) ;SW2=0 nhấn, SW1=1 không nhấn
            BRNE   EXIT_SW    ;không phải thoát
            LDI    R17,2       ;gán giá trị mã SW2
SET_FLAG:  SBR    FLAG_REG,(1<<SW_FLG);đặt cờ báo nhấn SW
EXIT_SW:   RET

```

#### ❖ Chương trình C ví dụ 7.21

Phần chương trình C ví dụ 7.21 dành cho người đọc tự soạn thảo.

#### ❖ Câu hỏi ôn tập

- Liệt kê các chân I/O có chức năng là ngõ ra mạch tạo sóng? Điều kiện để có các ngõ ra tạo sóng?
- Viết đoạn lệnh khởi động Timer0 chạy mô thức CTC tạo chuỗi xung ngõ ra OC0B, xung vuông đối xứng chu kỳ 2ms.
- Viết đoạn lệnh khởi động Timer2 tạo thời gian delay 1.6ms, ngay sau khi kết thúc thời gian delay chân PD7=0, PD6=1.
- Cho TCCR0A=0x42, TCCR0B=0x03, OCR0A=89, xung ra ở chân nào và tần số bao nhiêu Hz?
- Cho biết chức năng bit FOCnA/B.

#### 7.7.4 Lập trình Timer0/2 mô thức PWM nhanh(FPWM:Fast PWM) tạo sóng ngõ ra

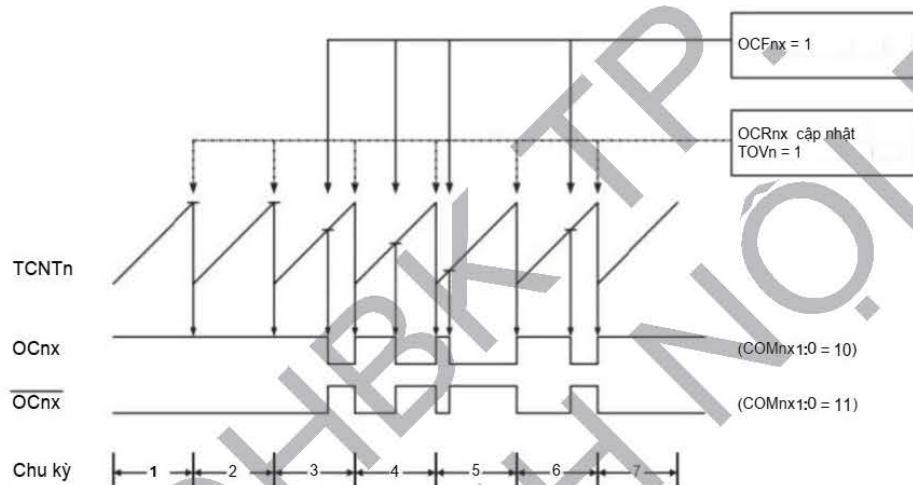
Mô thức FPWM tạo xung ngõ ra điều chế độ rộng xung tần số cao thích hợp cho các ứng dụng điều khiển PWM tần số cao.Timer0/2 cung cấp 2 mô thức FPWM gồm FPWM3,WGMn2:WGMn0=011 TOP=MAX=0xFF và FPWM7 ,WGMn2:WGMn0=111 TOP=OCRnA.

Bộ đếm bắt đầu đếm lên từ 0 cho đến TOP và ở xung đếm tiếp theo bộ đếm xóa về 0, hoạt động theo dạng một độ dốc. Trạng thái ngõ ra OCnx theo Bảng 7.15 và Bảng 7.18, tóm tắt hoạt động như sau:

- COMnx1:COMnx0=01: FPWM3 OCnx là I/O bình thường,FPWM7 OCnx đảo bit khi đạt kết quả so sánh,kênh B không sử dụng
- COMnx1:COMnx0=10:OCnx=0 khi đạt kết quả so sánh,OCnx=1 khi bộ đếm trở về BOTTOM=0 (trạng thái không đảo)
- COMnx1:COMnx0=11:OCnx=1 khi đạt kết quả so sánh,OCnx=0 khi bộ đếm trở về BOTTOM=0 (trạng thái đảo)

Còn TOVn=1 ở vị trí TOP và thanh ghi OCRnx cập nhật giá trị mới ở BOTTOM(thanh ghi OCRnx có bộ đếm kép, việc ghi vào OCRnx là ghi vào bộ đếm nên có thể ghi bất kỳ thời điểm nào)

Hình 7.50 minh họa mô thức FPWM tạo sóng ngõ ra OCnx.



Hình 7.50: Mô thức FPWM tạo sóng ngõ ra OCnx

Khi OCRnx=0, ngõ ra sẽ xuất hiện 1 gai xung hẹp khi bộ đếm đến MAX+1. Trường hợp OCRnx=TOP, OCnx=1 nếu trạng thái không đảo hoặc OCnx=0 nếu trạng thái đảo như đoạn đầu hình 7.50.

Tần số xung ngõ ra mô thức FPWM như hình 7.50 được tính:

$$f_o = \frac{F_{osc}}{N(TOP + 1)} \quad (7.3)$$

Độ rộng xung(chu kỳ nhiệm vụ) trạng thái không đảo:

$$T_{pn} = \frac{N(OCRnx + 1)}{F_{osc}} \quad (7.4)$$

Độ rộng xung(chu kỳ nhiệm vụ) trạng thái đảo:

$$T_{pin} = \frac{N(TOP - OCRnx)}{F_{osc}} \quad (7.5)$$

**Ví dụ 7.22:** Lập lại ví dụ 7.5,tạo chuỗi xung vuông tần số 1Khz,chu kỳ nhiệm vụ(CKNV)=30%.

Giải:

Theo bảng 7.4 ví dụ 7.5, ta chọn Timer0 mô thức FPWM7, với hệ số chia N=64, CLKT0=8μs, tính được giá trị TOP+1 theo công thức (7.3): TOP+1=8000Khz/(64x1Khz)=125. Suy ra OCR0A=124.

Để tạo CKNV=30% tương đương 300μs, chọn trạng thái không đảo, áp dụng công thức (7.4) tìm được OCR0B=300/8-1=36(làm tròn). Xung ngõ ra lấy ở chân OC0B(PB4).

#### ❖ Chương trình hợp ngữ ví dụ 7.22

```

.EQU P_OUT=4
.ORG 0
RJMP MAIN
.ORG 0X40
MAIN: LDI R16,HIGH(RAMEND) ;đưa stack lên đỉnh SRAM
      OUT SPH,R16
      LDI R16,LOW(RAMEND)
      OUT SPL,R16
      LDI R16,(1<<P_OUT) ;đặt P_OUT output
      OUT DDRB,R16
      LDI R16,124 ;giá trị đặt OCR0A tạo fo=1Khz
      OUT OCR0A,R16
      LDI R16,36 ;giá trị đặt OCR0B tạo CKNV 30%
      OUT OCR0B,R16
      LDI R16,0B00100011 ;Timer0 mode FPWM7,không đảo OC0B
      OUT TCCR0A,R16
      LDI R16,0B00001011 ;Timer0 mode FPWM7,hệ số chia N=64
      OUT TCCR0B,R16
HERE:  RJMP HERE

```

#### ❖ Chương trình C ví dụ 7.22

```

#include <avr/io.h>
const char p_out=4           ;// ký hiệu vị trí bit ngõ ra
int main()                   ;// bắt đầu chương trình
{
    DDRB |=(1<<p_out)       ;//khai báo p_out output
    OCR0A=124                ;//giá trị đặt OCR0A tạo fo=1Khz
    OCR0B=36                  ;//giá trị đặt OCR0B tạo CKNV 30%
    TCCR0A=0b00100011         ;//Timer0 mode FPWM7,OC0B không đảo
    TCCR0B=0b00001011         ;//Timer0 mode FPWM7,hệ số chia N=64
    while(1)                   ; // lặp vòng vô hạn
}

```

**Ví dụ 7.23:** Thiết kế mạch điều khiển tốc độ động cơ DC như hình 7.51. MCU tạo xung PWM tần số 1Khz, độ rộng xung thay đổi từ khoảng 5% - 95% chu kỳ, lái MOSFET\_N Q1 dẫn/tắt cấp nguồn +12V cho động cơ. Mỗi lần nhấn/nhả SW UP độ rộng xung tăng khoảng 2.5% và mỗi lần nhấn/nhả SW DWN độ rộng xung giảm khoảng 2.5%. Điện áp trung bình cấp cho động cơ:

$$V_{DC} = \frac{T_p}{T} V \quad (7.6)$$

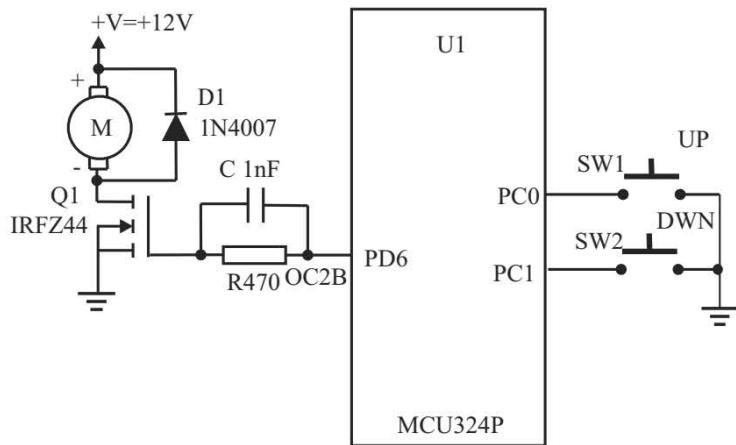
Với  $T_p$ : độ rộng xung(CKNV),  $T$ : chu kỳ xung,  $V$ : Điện áp nguồn cung cấp

Tốc độ động cơ DC tỉ lệ với điện áp cấp nguồn cho động cơ.

**Giải:**

Ta sử dụng Timer2 mô thức FPWM7 tạo xung PWM tần số fo=1Khz như ví dụ 7.22.

- Đặt TOP=OCR2A=124,tương ứng chọn hệ số chia N=64,CS20:CS00=100
- $T_{pmin}=5\%T=50\mu s$ ,tương ứng số xung đếm tối thiểu  $N_{min}=50/8=6$ (làm tròn)
- $T_{pmax}=95\%T=950\mu s$ ,tương ứng số xung đếm tối đa  $N_{max}=950/8=119$ (làm tròn)
- Mỗi bước tăng/giảm khi nhấn SW UP/DWN 2.5% $T=25\mu s$ ,tương ứng số xung thêm/bớt  $DN=3$
- Khởi động đặt  $OCR2B=N_{min}-1=5$  và thay đổi giá trị đặt từ 5 đến 118,mỗi bước cộng/trừ 3 tùy theo nhấn SW UP/DWN tương ứng.Đặt OC2B(PD6) trạng thái không đảo.



Hình 7.51: Sơ đồ mạch điều khiển tốc độ động cơ DC ví dụ 7.23

❖ Chương trình hợp ngữ ví dụ 7.23

```

.DEF FLAG_REG=R18
.EQU CONT_IN=PINC
.EQU SW_FLG=0
.EQU P_OUT=6
.EQU SW1=0
.EQU SW2=1
.EQU PMIN=5
.EQU PMAX=118
.EQU DELTA=3
.ORG 0
RJMP MAIN
.ORG 0X40
MAIN: LDI R16,HIGH(RAMEND) ;đưa stack lên đỉnh SRAM
      OUT SPHL,R16
      LDI R16,LOW(RAMEND)
      OUT SPL,R16
      LDI R16,(1<<P_OUT) ;đặt P_OUT output
      OUT DDRD,R16
      LDI R16,0 ;PortC input
      OUT DDRC,R16
      LDI R17,0X03 ;điện trở kéo lên PC0,PC1
      OUT PORTC,R17
      LDI R16,124 ;giá trị đặt OCR2A tạo fo=1Khz
      STS OCR2A,R16
      LDI R16,PMIN ;giá trị đặt OCR2B tạo CKNV 5%
      STS OCR2B,R16
      LDI R16,0B00100011 ;Timer2 mode FPWM7,không đảo OC0B
      STS TCCR2A,R16
      LDI R16,0B00001100 ;Timer2 mode FPWM7,hệ số chia N=64
      STS TCCR2B,R16
      LDI R19,DELTA
START: RCALL GET_SW ;đọc SW
      SBRS FLAG_REG,SW_FLG ;cờ SW_FLG=1 có SW nhấn
      RJMP START
      CPI R17,1 ;SW1=UP nhấn?
      BRNE SW2_CHK ;kiểm tra SW2
      LDS R17,OCR2B ;đọc OCR2B
      ADD R17,R19 ;tăng thêm 1 bước

```

	CPI	R17,PMAX	;độ rộng xung vượt PMAX?
	BRCS	UP_SP	;chưa cập nhật giá trị mới
	LDI	R17,PMAX	;giới hạn giá trị=PMAX
UP_SP:	STS	OCR2B,R17	;cập nhật OCR2B
	RJMP	START	
SW2_CHK:	CPI	R17,2	;SW2=DWN nhấn?
	BRNE	START	
	LDS	R17,OCR2B	;đọc OCR2B
	SUB	R17,R19	;giảm bớt 1 bước
	CPI	R17,PMIN	;độ rộng xung dưới PMIN?
	BRCC	DWN_SP	;chưa cập nhật giá trị mới
	LDI	R17,PMIN	;giới hạn giá trị=PMIN
DWN_SP:	STS	OCR2B,R17	;cập nhật OCR2B
	RJMP	START	
-----			
;GET_SW đọc trạng thái SW1,SW2 có chông rung			
;Trả về mã SW1=1 hoặc mã SW2=2 và cờ SW_FLG=1 nếu có SW nhấn			
;Trả về cờ SW_FLG=0 nếu không có SW nhấn			
;Sử dụng R16,R17,cờ SW_FLG thuộc thanh ghi FLAG_REG			
-----			
GET_SW:	CBR	FLAG_REG,(1<<SW_FLG);xóa cờ báo nhấn SW	
BACK0:	LDI	R16,50 ;kiểm tra SW nhấn 50 lần liên tục	
WAIT0:	IN	R17,CONT_IN	
	ANDI	R17,(1<<SW1) (1<<SW2);che bit SW1,SW2	
	CPI	R17,(1<<SW1) (1<<SW2);kiểm tra SW nhấn?	
	BREQ	EXIT_SW ;không nhấn thoát	
	DEC	R16 ;có nhấn tiếp tục	
	BRNE	WAIT0 ;	
	PUSH	R17 ;cắt mã SW	
BACK1:	LDI	R16,50 ;kiểm tra sw nhả 50 lần liên tục	
WAIT1:	IN	R17,CONT_IN	
	ANDI	R17,(1<<SW1) (1<<SW2)	
	CPI	R17,(1<<SW1) (1<<SW2)	
	BRNE	BACK1 ;chờ nhả SW	
	DEC	R16	
	BRNE	WAIT1	
	POP	R17 ;phục hồi mã SW	
	CPI	R17,(1<<SW2) ;SW1=0 nhấn,SW2=1 không nhấn	
	BRNE	SW2_CODE ;không phải kiểm tra mã SW2	
	LDI	R17,1 ;gán giá trị mã SW1	
	RJMP	SET_FLAG ;báo cờ nhấn SW	
SW2_CODE:	CPI	R17,(1<<SW1) ;SW2=0 nhấn,SW1=1 không nhấn	
	BRNE	EXIT_SW ;không phải thoát	
	LDI	R17,2 ;gán giá trị mã SW2	
SET_FLAG:	SBR	FLAG_REG,(1<<SW_FLG);đặt cờ báo nhấn SW	
EXIT_SW:	RET		

### ❖ Chương trình C ví dụ 7.23

```
#include <avr/io.h>
#define cont_in PINC
const char p_out=6,sw1=0,sw2=1      // ký hiệu vị trí bit p_out,sw1,sw2
const char sw_flg=0                  // ký hiệu vị trí bit cờ báo sw nhấn
const char Pmax=118,Pmin=5,delta=3 ;ký hiệu các giá trị độ rộng xung
void get_sw()                        //khai báo hàm đọc sw
```

```

unsigned char flag_reg,sw_code;
int main() // bắt đầu chương trình
{
    DDRD |=(1<<p_out) ;//khai báo p_out output
    DDRC=0x00 ;//khai báo PortC input
    PORTC=(1<<sw1)|(1<<sw2); //điện trở kéo lên bit sw1,sw2
    OCR2A=124 ;//giá trị đặt OCR2A tạo fo=1Khz
    OCR2B=Pmin ;//giá trị đặt OCR2B tạo CKNV 5%
    TCCR2A=0b00100011 ;//Timer2 mode FPWM7,OC0B không đảo
    TCCR2B=0b00001100 ;//Timer2 mode FPWM7,hệ số chia N=64
    while(1) // lập vòng vô hạn
    {
        do
            get_sw() ;//đọc sw
            while(!(flag_reg&(1<<sw_flg))) ;//thoát khi cờ sw_flg=1 báo có sw nhấn
            if(sw_code==1)
            {
                OCR2B=OCR2B+delta ;//tăng độ rộng xung 1 bước
                if(OCR2B>=Pmax)
                    OCR2B=Pmax ;//giới hạn độ rộng xung max
            }
            else if(sw_code==2)
            {
                OCR2B=OCR2B-delta ;//giảm độ rộng xung 1 bước
                if(OCR2B<Pmin)
                    OCR2B=Pmin ;//giới hạn độ rộng xung min
            }
        }
    //}
    void get_sw() //hàm đọc sw trả về mã sw và cờ sw_flg=1 báo có sw nhấn
    {
        unsigned char tam,i;
        flag_reg&=~(1<<sw_flg) ;//xóa cờ sw_flg
        for(i=50;i>=1;i--) //lập vòng đọc sw nhấn liên tục 50 lần
        {
            tam=cont_in&((1<<sw1)|(1<<sw2)) ;//đọc Port và che các bit sw
            if(tam==((1<<sw1)|(1<<sw2))) //không có sw nhấn thoát
                break;
        }
        if(tam!=((1<<sw1)|(1<<sw2))) //có sw nhấn tiếp tục
        {
            sw_code=tam ;//cất mã sw
            for(i=50;i>=1;i--) //lập vòng đọc sw nhả liên tục 50 lần
            {
                tam=cont_in&((1<<sw1)|(1<<sw2)) ;//đọc Port và che các bit sw
                if(tam!=((1<<sw1)|(1<<sw2)))
                    i=50 ;//lập vòng lại khi chưa nhả sw
            }
            if(sw_code==~(1<<sw2)) //sw1 nhấn,sw2 không nhấn
            {
                sw_code=1 ;//gán mã sw1=1
                flag_reg|=(1<<sw_flg) ;//đặt cờ sw_flg=1
            }
            else if(sw_code==(1<<sw1)) //sw2 nhấn,sw1 không nhấn

```

```

    {
        sw_code=2           ;//gán mã sw2=2
        flag_reg|=(1<<sw_flg) ;//đặt cờ sw_flg=1
    }
}

```

#### ❖ Câu hỏi ôn tập

- Muốn Timer2 chạy mô thức FPWM3,hệ số chia N=256,xuất xung ra OC2A/B,phải nạp TCCR2A/B giá trị bao nhiêu?
- Cho TCCR2A=0xB3,TCCR2B=0x04,OCR2A=99,OCR2B=119.Tần số xung ngõ ra OC2A/B bằng bao nhiêu?
- Cũng theo câu 3,cho biết CKNV ngõ ra OC2A/B.
- Viết đoạn lệnh khởi động Timer0 tạo xung ra tần số 400Hz,CKNV 45%,sử dụng mô thức FPWM.
- Khi muốn thay đổi độ rộng xung trong mô thức FPWM,nên ghi giá trị cập nhật vào thời điểm nào để đảm bảo không bị cập nhật sai?

#### 7.7.5 Lập trình Timer0/2 mô thức PWM hiệu chỉnh pha(PCPWM:Phase Correct PWM) tạo sóng ngõ ra

Trong mô thức PCPWM bộ đếm hoạt động theo dạng 2 độ dốc.Bộ đếm bắt đầu đếm lên từ BOTTOM đến TOP và từ TOP đếm xuống về lại BOTTOM.Mô thức PCPWM có 2 lựa chọn,PCPWM1 WGMn2:WGMn0=001 TOP=0xFF hoặc PCPWM5 WGMn2:WGMn0=101 TOP=OCrnA.

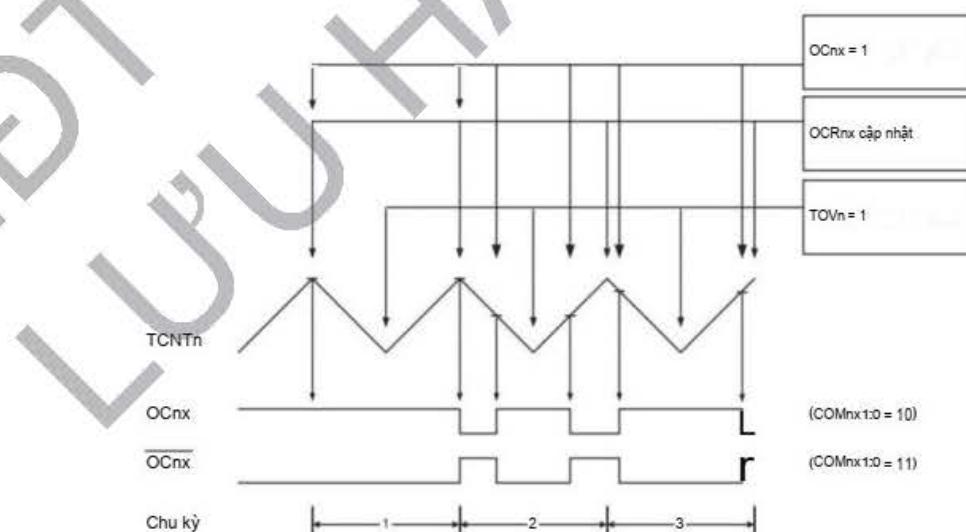
Trạng thái ngõ ra OCnx theo Bảng 7.16 và Bảng 7.19,tóm tắt hoạt động như sau:

- COMnx1:COMnx0=01: PCPWM1 OCnx là I/O bình thường,PCPWM5 OCnx đảo bit khi đạt kết quả so sánh,kênh B không sử dụng
- COMnx1:COMnx0=10:OCnx=0 khi đếm lên đạt kết quả so sánh,OCnx=1 khi đếm xuống đạt kết quả so sánh(trạng thái không đảo)
- COMnx1:COMnx0=11:OCnx=1 khi đếm lên đạt kết quả so sánh,OCnx=0 khi đếm xuống đạt kết quả so sánh(trạng thái đảo)

Cờ TOVn=1 tại BOTTOM và thanh ghi OCRnA cập nhật tại TOP.

Mô thức PCPWM có tần số xung ngõ ra thấp hơn mô thức FPWM,tuy nhiên độ rộng xung đổi xứng trong một chu kỳ nên pha tín hiệu không thay đổi khi thay đổi độ rộng xung.Do đó mô thức PCPWM thường được ứng dụng trong điều khiển tốc độ động cơ,bộ hoán năng(inverter),điều khiển AC....

Hình 7.52 minh họa dạng sóng ngõ ra OCnx ở mô thức PCPWM với COMnx1:COMnx0=10,11.



Hình 7.52: Mô thức PCPWM tạo sóng ngõ ra OCnx

Trường hợp đặt OCRnx=TOP, ngõ ra OCnx=1 ở trạng thái không đảo hoặc OCnx=0 ở trạng thái đảo. Trường hợp đặc biệt ngõ ra OCnx đảo trạng thái khi chưa đạt kết quả so sánh xảy ra ở một trong hai điều kiện sau:

- OCRnx thay đổi từ MAX như ở đầu chu kỳ 2 hình 7.52. Ngõ ra OCnx đảo bit để đảm bảo độ rộng xung đối xứng xung quanh điểm BOTTOM ở lần đếm kế tiếp.
- Bộ đếm bắt đầu đếm từ giá trị cao hơn giá trị đặt trong OCRnx nên không xảy ra kết quả so sánh, do đó OCnx đảo bit để đảm bảo như trên.

Từ hình 7.52 ta tìm được công thức tính tần số xung ngõ ra mô thức PCPWM:

$$f_{opc} = \frac{F_{osc}}{N \times 2TOP} \quad (7.7)$$

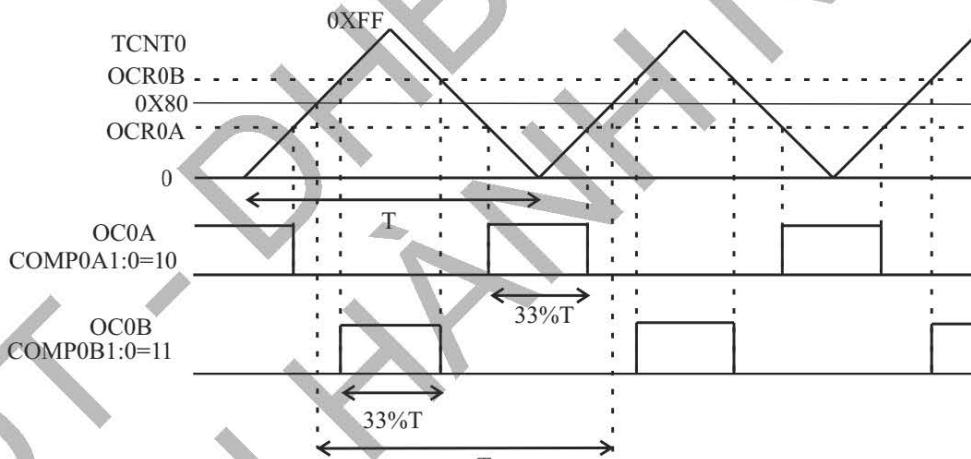
Độ rộng xung(CKNV)trạng thái không đảo:

$$T_{pnpc} = 2OCRnx \frac{N}{F_{osc}} \quad (7.8)$$

Độ rộng xung(CKNV)trạng thái không đảo:

$$T_{pinpc} = 2(TOP - OCRnx) \frac{N}{F_{osc}} \quad (7.9)$$

**Ví dụ 7.24:** Trong các ứng dụng tạo xung cho inverter, điều khiển AC, dạng xung điều khiển đối xứng như hình 7.53 thường được áp dụng. Viết một chương trình tạo 2 chuỗi xung ngõ ra OC0A(PB3) và OC0B(PB4) như hình 7.53, với tần số xung 61Hz.



Hình 7.53: Tạo 2 chuỗi xung trên OC0A, OC0B ví dụ 7.24

**Giải:**

Sử dụng Timer0 tạo 2 chuỗi xung ngõ ra trên OC0A, OC0B như hình 7.53, ta phải sử dụng mô thức PCPWM1 WGM02: WGM00=001 TOP=0xFF.

- Tần số xung ra  $f_{opc}=61\text{Hz}$ , theo công thức (7.7)  $N=8\times10^6/(61\times510)=257$   
Chọn hệ số chia  $N=256$ , tính lại  $f_{opc}=8\times10^6/(256\times510)=61.3\text{Hz}$
- Chọn kênh A trạng thái không đảo, theo công thức (7.8):  
 $OCR0A=(0.33/61)\times8\times10^6/(2\times256)=84$ (làm tròn)
- Chọn kênh B trạng thái đảo, theo công thức (7.9):  
 $OCR0B=255-(0.33/61)\times8\times10^6/(2\times256)=170$ (làm tròn)

#### ❖ Chương trình hợp ngữ ví dụ 7.24

```
.EQU P_A=3 ;ký hiệu OC0A
.EQU P_B=4 ;ký hiệu OC0B
```

```

.EQU DAT_A=84      ;giá trị đặt OCR0A
.EQU DAT_B=170      ;giá trị đặt OCR0B
.ORG 0
RJMP MAIN
.ORG 0X40
MAIN: LDI R16,HIGH(RAMEND) ;đưa stack lên đỉnh SRAM
      OUT SPH,R16
      LDI R16,LOW(RAMEND)
      OUT SPL,R16
      LDI R16,(1<<P_A)|(1<<P_B) ;đặt P_A,P_B output
      OUT DDRB,R16
      LDI R17,DAT_A      ;giá trị so sánh kênh A
      OUT OCR0A,R17
      LDI R17,DAT_B      ;giá trị so sánh kênh B
      OUT OCR0B,R17
      LDI R17,0B10110001  ;Timer0 mode PCPWM,OC0A không đảo,OC0B đảo
      OUT TCCR0A,R17
      LDI R17,0X04        ;Timer0 chạy,hệ số chia N=256
      OUT TCCR0B,R17
START: RJMP START      ;lặp vòng lại

```

#### ❖ Chương trình C ví dụ 7.24

```

#include <avr/io.h>
const char p_a=3,p_b=4      ;// ký hiệu vị trí bit ngõ ra
const char dat_a=84,dat_b=170;//ký hiệu các giá trị đặt trước
int main()                  //bắt đầu chương trình
{
    DDRB |=(1<<p_a)|(1<<p_b) ;//khai báo p_a,p_b output
    OCR0A=dat_a                ;//giá trị đặt OCR0A
    OCR0B=dat_b                ;//giá trị đặt OCR0B
    TCCR0A=0b10110001          ;//Timer0 mode PCPWM1,OC0A không đảo,OC0B đảo
    TCCR0B=4                    ;//hệ số chia N=256
    while(1)                   ; // lặp vòng vô hạn
}

```

#### ❖ Câu hỏi ôn tập

- Muốn Timer2 chạy模式 PCPWM1,hệ số chia N=256,xuất xung ra OC2A/B,phải nạp TCCR2A/B giá trị bao nhiêu?
- Cho TCCR2A=0xB1,TCCR2B=0x04,OCR2A=99,OCR2B=119.Tần số xung ngõ ra OC2A/B bằng bao nhiêu?
- Cũng theo câu 3,cho biết CKNV ngõ ra OC2A/B.
- Viết đoạn lệnh khởi động Timer0 tạo xung ra tần số 400Hz,CKNV 45%,sử dụng mode PCPWM.
- Khi muốn thay đổi độ rộng xung trong mode PCPWM,nên ghi giá trị cập nhật vào thời điểm nào để đảm bảo không bị cập nhật sai?

### 7.8 Lập trình tạo sóng với Timer1

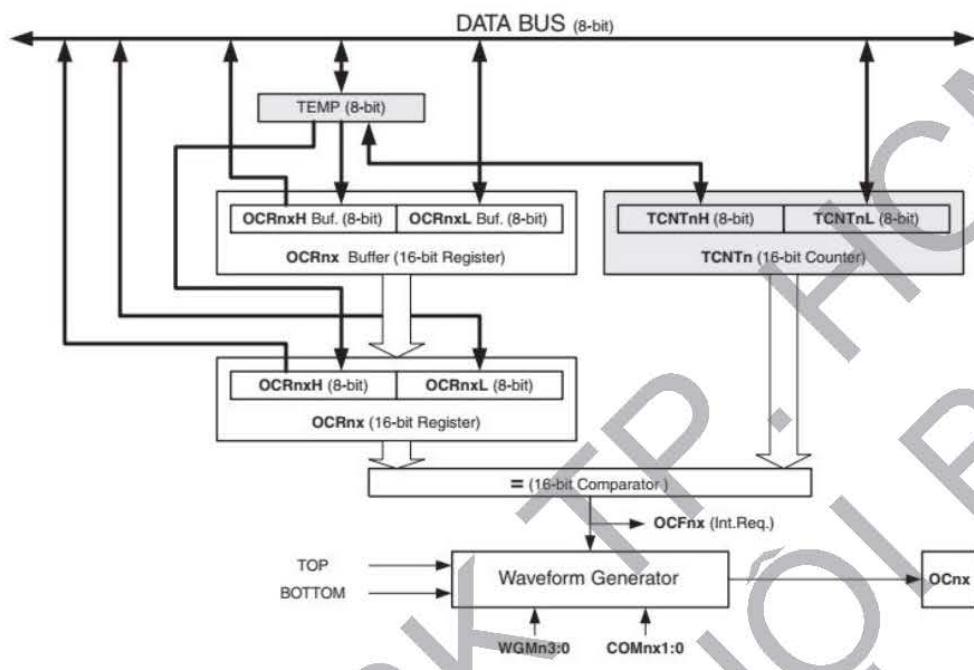
Về cơ bản việc tạo sóng với Timer1 cũng tương tự như Timer0/2.Tuy nhiên Timer1 có độ dài 16 bit và tới 15 mode tạo sóng nên sử dụng Timer1 tạo sóng sẽ đa dạng hơn nhiều!

#### 7.8.1 Sơ đồ khôi so sánh ngõ ra Timer1

Việc tạo sóng ngõ ra bằng phần cứng thực hiện thông qua khôi so sánh ngõ ra như hình 7.54.

Khi bộ đếm TCNTn đếm đến giá trị bằng giá trị đặt trong thanh ghi OCRnx,đạt kết quả so sánh,còn OCFnx=1 kích khôi bộ tạo sóng(Waveform Generator)làm việc.Tùy theo mode cài đặt bằng các bit WGMn3:0,ngõ ra chân Port OCnx sẽ thay đổi trạng thái logic theo cài đặt các bit COMnx1:0.

Với Timer1 2 chân OC1A(PD5) và OC1B(PD4) tương ứng kênh A và B.Các thanh ghi OCR1x trong Timer1 có bộ đếm kép.Ngoại trừ mô thức NOR và CTC MCU truy xuất trực tiếp OCR1x,các mô thức PWM đều truy xuất qua bộ đếm.Các thanh ghi OCR1x được cập nhật từ giá trị trong bộ đếm đồng bộ với xung CK bộ đếm tại điểm TOP hay BOTTOM để đảm bảo việc tạo xung đối xứng,tránh có gai đột biến ngõ ra.Việc truy xuất các thanh ghi OCR1x cũng phải đảm bảo theo quy tắc truy xuất thanh ghi 16 bit,ghi byte cao trước,đọc byte thấp trước.



Hình 7.54: Sơ đồ khối so sánh ngõ ra Timer1

### 7.8.2 Chọn mô thức tạo sóng và trạng thái logic chân OCnx

Trong phần này ta sẽ xem chi tiết chức năng các bit WGM13:WGM10 và COM1x1:COM1x0.

#### 1. Bit WGM13,WGM12(bit 4,3 TCCR1B),WGM11,WGM10(bit 1,0 TCCR1A)

Các bit WGM13:WGM10 chọn mô thức hoạt động Timer1 như Bảng 7.20.

Bảng 7.20: Các mô thức Timer1

MODE	WGM13	WGM12	WGM11	WGM10	MÔ THỨC HOẠT ĐỘNG	TOP	Thời điểm cập nhật OCR1x	TOV1=1 ở điểm
0	0	0	0	0	Normal	0xFFFF	Tức thời	MAX
1	0	0	0	1	PWM, Phase Correct, 8-bit	0x00FF	TOP	BOTTOM
2	0	0	1	0	PWM, Phase Correct, 9-bit	0x01FF	TOP	BOTTOM
3	0	0	1	1	PWM, Phase Correct, 10-bit	0x03FF	TOP	BOTTOM
4	0	1	0	0	CTC	OCR1A	Tức thời	MAX
5	0	1	0	1	Fast PWM, 8-bit	0x00FF	BOTTOM	TOP
6	0	1	1	0	Fast PWM, 9-bit	0x01FF	BOTTOM	TOP
7	0	1	1	1	Fast PWM, 10-bit	0x03FF	BOTTOM	TOP
8	1	0	0	0	PWM, Phase and Frequency Correct	ICR1	BOTTOM	BOTTOM
9	1	0	0	1	PWM, Phase and Frequency Correct	OCR1A	BOTTOM	BOTTOM
10	1	0	1	0	PWM, Phase Correct	ICR1	TOP	BOTTOM
11	1	0	1	1	PWM, Phase Correct	OCR1A	TOP	BOTTOM
12	1	1	0	0	CTC	ICR1	Tức thời	MAX
13	1	1	0	1	Dự trữ	-	-	-
14	1	1	1	0	Fast PWM	ICR1	BOTTOM	TOP
15	1	1	1	1	Fast PWM	OCR1A	BOTTOM	TOP

Ngoài trừ mô thức 13 dự trữ,có 15 mô thức hoạt động phân thành 5 nhóm như sau:

- Mô thức NOR 1 đã trình bày

- Mô thức CTC có 2 mô thức CTC4, CTC12 đã trình bày
- Mô thức FPWM có 5 mô thức FPWM 5, 6, 7 TOP cố định (0xFF, 0x1FF, 0x3FF), FPWM 14, 15 TOP=ICR1 hay TOP=OCR1A
- Mô thức PCPWM có 5 mô thức PCPWM 1, 2, 3 TOP cố định (0xFF, 0x1FF, 0x3FF), PCPWM 10, 11 TOP=ICR1 hay TOP=OCR1A
- Mô thức PFCPWM (PWM, Phase and Frequency Correct) có 2 mô thức PFCPWM 8, 9 TOP=ICR1 hay TOP=OCR1A

## 2. Bit COM1A1:COM1A0(bit 7,6 TCCRnA), COM1B1:COM1B0(bit 5,4 TCCRnA)

Các bit này cài đặt trạng thái logic ngõ ra OC1A/B khi đạt kết quả so sánh, tùy vào mô thức không PWM(NOR, CTC), FPWM hay PCPWM, PFCPWM theo mô tả ở Bảng 7.21 đến 7.23

**Bảng 7.21:** Trạng thái ngõ ra OC1A/B mô thức NOR, CTC

COM1A1/COM1B1	COM1A0/COM1B0	Mô tả
0	0	I/O bình thường, OC1A/B không kết nối
0	1	Đảo bit OC1A/B khi đạt kết quả so sánh
1	0	Xóa OC1A/B=0 khi đạt kết quả so sánh
1	1	Đặt OC1A/B=1 khi đạt kết quả so sánh

**Bảng 7.22:** Trạng thái ngõ ra OC1A/B mô thức FPWM

COM1A1/COM1B1	COM1A0/COM1B0	Mô tả
0	0	I/O bình thường, OC1A/B không kết nối
0	1	WGM13:0=14,15: đảo bit OC1A khi đạt kết quả so sánh, OC1B=I/O bình thường
1	0	Các trường hợp khác OC1A/B=I/O bình thường
1	1	Xóa OC1A/B=0 khi đạt kết quả so sánh
7		Đặt OC1A/B=1 ở BOTTOM(mô thức không đảo)
		Đặt OC1A/B=1 khi đạt kết quả so sánh
		Xóa OC1A/B=0 ở BOTTOM(mô thức đảo)

**Bảng 7.23:** Trạng thái ngõ ra OC1A/B mô thức PCPWM và PFCPWM

COM1A1/COM1B1	COM1A0/COM1B0	Mô tả
0	0	I/O bình thường, OCnA không kết nối
0	1	WGM13:0=9,11: đảo bit OC1A khi đạt kết quả so sánh, OC1B=I/O bình thường
1	0	Các trường hợp khác OC1A/B=I/O bình thường
1	1	Xóa OCnA=0 khi đếm lên đạt kết quả so sánh
		Đặt OCnA=1 khi đếm xuống đạt kết quả so sánh
		Đặt OCnA=1 khi đếm lên đạt kết quả so sánh
		Xóa OCnA=0 khi đếm xuống đạt kết quả so sánh

### 7.8.3 Lập trình Timer1 mô thức NOR, CTC tạo sóng ngõ ra

Tương tự như trên, mô thức NOR không thích hợp áp dụng tạo sóng ngõ ra, ta chỉ sử dụng mô thức CTC với 2 lựa chọn CTC4, CTC12.

- Mô thức CTC4 WGM13:WGM10=0100, TOP=OCR1A cập nhật tức thời, chờ TOV1=1 tại MAX (tương ứng đặt OCR1A=0xFFFF)
- Mô thức CTC12 WGM13:WGM10=1100, TOP=ICR1A cập nhật tức thời, chờ TOV1=1 tại MAX (tương ứng đặt OCR1A=0xFFFF)

Các trạng thái ngõ ra OC1A/B theo COM1x1:COM1x0 ở Bảng 7.21.

Tần số xung ngõ ra (trường hợp COM1A1:COM1A0=01) được tính như công thức (7.2):

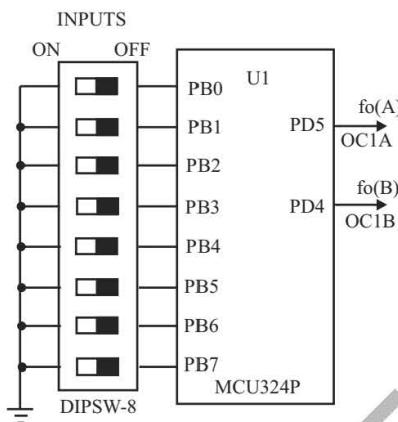
$$f_o = \frac{F_{osc}}{2N \times (TOP + 1)} \quad (7.10)$$

TOP=OCR1A (mô thức CTC4), TOP=ICR1 (mô thức CTC12)

**Ví dụ 7.25:** Thiết kế mạch tạo 2 chuỗi xung vuông đổi xứng có chu kỳ thay đổi được  $T=10xM(\mu s)M=0-255$ , trong đó cạnh lén chuỗi xung thứ 2 cách sau cạnh lén chuỗi xung 1 chính xác 20%T

**Giải:**

Sơ đồ mạch như hình 7.55, PortB nhập hệ số nhân M thay đổi chu kỳ T. Ta sử dụng Timer1 mô thức CTC12 tạo 2 chuỗi xung ngõ ra OC1A(PD5), OC1B(PD4). Để có thể tạo xung chính xác, ta chọn hệ số chia  $N=1$  ứng với  $CLKT1=0.125\mu s \# 8$  xung đếm/1μs, số xung bộ đếm tương ứng chu kỳ xung cực đại bằng  $2550 \times 8 = 20400$ . Số xung ứng với 20%Tmin bằng  $0.2 \times 10 \times 8 = 16$  luôn là số chẵn!



Hình 7.55: Sơ đồ mạch ví dụ 7.55

Ta chọn ngõ ra OC1B tạo xung cạnh lén sau xung ngõ ra OC1A 20%T.

Giá trị nạp cho ICR1=10xMx8/2 - 1=40M - 1

Giá trị nạp cho OCR1A=40M/2 - 1=20M - 1

Giá trị nạp cho OCR1B=40M/2+0.2x80M - 1=36M - 1

#### ❖ Chương trình hợp ngữ ví dụ 7.25

```

.EQU IMPORT=PINB
.EQU P_A=5 ;ký hiệu OC1A
.EQU P_B=4 ;ký hiệu OC1B
.ORG 0
RJMP MAIN
.ORG 0X40
MAIN: LDI R16,HIGH(RAMEND) ;đưa stack lên đỉnh SRAM
      OUT SPH,R16
      LDI R16,LOW(RAMEND)
      OUT SPL,R16
      LDI R16,(1<<P_A)|(1<<P_B) ;đặt P_A,P_B output
      OUT DDRD,R16
      LDI R16,0X00 ;PortB input
      OUT DDRB,R16
      LDI R16,0xFF ;điện trở kéo lên PortB
      OUT PORTB,R16
      LDI R17,0B01010000 ;Timer1 mode CTC12,OC1A,OC1B đảo bit
      STS TCCR1A,R17
      LDI R17,0B00011001 ;Timer1 chạy,mode CTC12,hệ số chia N=1
      STS TCCR1B,R17
START: RCALL PER_CHG ;ctc tính giá trị đặt trước ICR1,OCR1A/B
      RJMP START ;lặp vòng lại
;
PER_CHG: IN   R17,IMPORT ;đọc hệ số nhân M
        MOV  R18,R17 ;cắt M
        LDI  R16,40 ;tính giá trị đặt ICR1

```

```

MUL R17,R16
MOVW R24,R0
SBIW R25:R24,1
STS ICR1H,R25
STS ICR1L,R24
LDI R16,20 ;tính giá trị đặt OCR1A
MOV R17,R18
MUL R17,R16
MOVW R24,R0
SBIW R25:R24,1
STS OCR1AH,R25
STS OCR1AL,R24
LDI R16,36 ;tính giá trị đặt OCR1B
MUL R18,R16
MOVW R24,R0
SBIW R25:R24,1
STS OCR1BH,R25
STS OCR1BL,R24
RET

```

- **Lưu ý:** Khi thực hiện ghi vào các thanh ghi 16 bit của Timer1, phải ghi byte cao trước, byte thấp sau!  
Còn khi đọc thì ngược lại đọc byte thấp trước, byte cao sau!

❖ **Chương trình C ví dụ 7.25**

```

#include <avr/io.h>
#define import PINB
const char p_a=5,p_b=4 //ký hiệu vị trí bit ngõ ra
void per_chg() //khai báo hàm tính các giá trị đặt ICR1,OCR1A/B
int main() //bắt đầu chương trình
{
    DDRD |=(1<<p_a)|(1<<p_b); //khai báo p_a,p_b output
    DDRB=0x00 //PortB input
    PORTB=0xff //điện trở kéo lên PortB

    TCCR1A=0b01010000 //Timer1 mode CTC12,OC1A,OC1B đảo bit
    TCCR1B=0b00011001 //Timer1 mode CTC12,hệ số chia N=1, chạy Timer1
    while(1) //lặp vòng vô hạn
    {
        per_chg(); //gọi hàm tính các giá trị đặt ICR1,OCR1A/B
    }
}

void per_chg()
{
    unsigned char tam;
    tam=import //đọc hệ số nhân M
    ICR1=40*tam - 1 //tính giá trị đặt ICR1
    OCR1A=20*tam - 1 //tính giá trị đặt OCR1A
    OCR1B=36*tam - 1 //tính giá trị đặt OCR1B
}

```

#### 7.8.4 Lập trình Timer1 mô thức FPWM tạo sóng ngõ ra

Mô thức FPWM của Timer1 cũng tương tự so với Timer0/2. Timer1 có 5 mô thức FPWM như sau:  
- Mô thức FPWM5- 8 bit WGM13:WGM10=0101, TOP=0x00FF

- Mô thức FPWM6- 9 bit WGM13: WGM10=0110, TOP=0x01FF
- Mô thức FPWM7- 10 bit WGM13: WGM10=0111, TOP=0x03FF
- Mô thức FPWM14 WGM13: WGM10=1110, TOP=ICR1, cờ ICF1=1 tại TOP
- Mô thức FPWM15 WGM13: WGM10=1111, TOP=OCR1A, cờ OCF1A=1 tại TOP

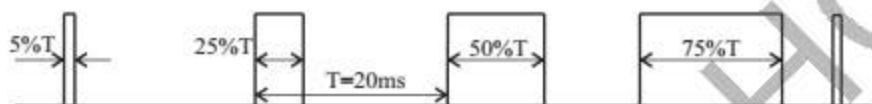
Trong cả 5 mô thức trên OCR1x cập tại BOTTOM, TOV1=1 tại TOP

Giá trị đặt trước tối thiểu cho OCR1x hay ICR1=0x0003.

Các trạng thái ngõ ra OC1A/B theo COM1x1:COM1x0 ở Bảng 7.22.

Tần số xung ngõ ra và độ rộng xung trạng thái không đảo hay đảo được tính như công thức (7.3),(7.4),(7.5).

**Ví dụ 7.26:** Viết một chương trình tạo chuỗi xung vuông tần số cố định  $f_0=50\text{Hz}$ , độ rộng xung lần lượt thay đổi 5%T, 25%T, 50%T, 75%T, với T=1/f<sub>0</sub> như hình 7.56.



Hình 7.56: Tạo chuỗi xung theo ví dụ 7.26

**Giải:**

Ta sử dụng Timer1 mô thức FPWM14, xung ngõ ra OC1A không đảo.

Chọn hệ số chia N=8,1 xung đếm dài 1μs.

- Theo công thức (7.3): TOP+1=20000x1=20000, suy ra ICR1=19999

- Theo công thức (7.4), tính được các giá trị đặt OCR1A để có độ rộng xung 5%T, 25%T, 50%T, 75%T lần lượt là 999,4999,9999,14999.

#### ❖ Chương trình hợp ngữ ví dụ 7.26

```

.EQU P_OUT=5          ;ký hiệu OC1A
.EQU P_COUNT=19999    ;ký hiệu giá trị đặt ICR1
.ORG 0
.RJMP MAIN
.ORG 0X40
MAIN: LDI R16,HIGH(RAMEND) ;đưa stack lên đỉnh SRAM
      OUT SPH,R16
      LDI R16,LOW(RAMEND)
      OUT SPL,R16
      LDI R16,(1<<P_OUT)   ;đặt P_OUT output
      OUT DDRD,R16
      LDI R16,4              ;bộ đếm số lần cập nhật giá trị đặt OCR1A
      LDI ZH,HIGH(TIME_TAB<<1);Z trả địa chỉ đầu bảng tra
      LDI ZL,LOW(TIME_TAB<<1)
      LDI R17,HIGH(P_COUNT)
      STS ICR1H,R17           ;nạp byte cao ICR1
      LDI R17,LOW(P_COUNT)
      STS ICR1L,R17           ;nạp byte thấp ICR1
      LDI R17,0B10000010      ;Timer1 mode FPWM 14,OC1A không đảo
      STS TCCR1A,R17
      LDI R17,0B00011010      ;Timer1 chạy, mode FPWM 14,hệ số chia N=8
      STS TCCR1B,R17
START: RCALL TIME_INTV ;ctc tính giá trị đặt trước OCR1A
      RJMP START             ;lặp vòng lại
;
TIME_INTV: IN  R17,TIFR1 ;đọc cờ ICF1
            SBRs R17,ICF1 ;cờ ICF1=1 xử lý tiếp
            RJMP TIME_INTV ;chờ cờ ICF1=1
            OUT TIFR1,R17 ;xóa cờ ICF1

```

```

LPM R17,Z+ ;lấy byte thấp giá trị cập nhật OCR1A
MOV R18,R17 ;cắt byte thấp
LPM R17,Z+ ;lấy byte cao giá trị cập nhật OCR1A
STS OCR1AH,R17 ;cập nhật byte cao OCR1A
STS OCR1AL,R18 ;cập nhật byte thấp OCR1A
DEC R16 ;đếm số lần cập nhật
BRNE EXIT_TIME ;thoát nếu chưa đủ số lần cập nhật
LDI R16,4 ;nạp lại bộ đếm
LDI ZH,HIGH(TIME_TAB<<1);nạp lại địa chỉ bảng tra
LDI ZL,LOW(TIME_TAB<<1)

EXIT_TIME: RET
-----
TIME_TAB: .DW 999,4999,9999,14999

```

- Do các giá trị đặt trước dài 2 byte nên phải dùng chỉ dẫn DW khai báo. Dữ liệu word cất trong bộ nhớ Flash ROM theo byte thấp địa chỉ thấp, byte cao địa chỉ cao.
- Khi đọc dữ liệu từ bảng tra và ghi vào OCR1A, phải đảm bảo ghi byte cao trước, byte thấp sau!
- Thực hiện lệnh ghi OCR1A bất kỳ thời điểm nào thực chất là ghi vào bộ đếm, MCU sẽ chỉ cập nhật vào thanh ghi OCR1A khi bộ đếm trở về BOTTOM!

#### ❖ Chương trình C ví dụ 7.26

```

#include <avr/io.h>
const char p_out=5 // ký hiệu vị trí bit ngõ ra
const int p_count=19999 //khai báo giá trị đặt ICR1
unsigned char i;
unsigned int Time_tab[]={999,4999,9999,14999};//khai báo bảng tra độ rộng xung
int main() // bắt đầu chương trình
{
    DDRD |=(1<<p_out); //khai báo p_out output
    ICR1=p_count //nạp giá trị đặt ICR1
    TCCR1A=0b10000010;//Timer1 mode FPWM 14,OC1A không đảo
    TCCR1B=0b00011010;//Timer1 mode FPWM 14,hệ số chia N=8, chạy Timer1
    while(1) // lập vòng vô hạn
    {
        for(i=0;i<=3;i++)
        {
            while(!(TIFR1&(1<<ICF1))) ;//chờ cờ ICF1=1
            TIFR1 |=(1<<ICF1) ;//xóa cờ ICF1
            OCR1A=Time_tab[i] //nạp giá trị đặt trước kế tiếp
        }
    }
}

```

#### 7.8.5 Lập trình Timer1 mô thức PCPWM tạo sóng ngõ ra

Mô thức PCPWM của Timer1 cũng tương tự so với Timer0/2. Timer1 có 5 mô thức PCPWM như sau:

- Mô thức PCWM1- 8 bit WGM13:WGM10=0001, TOP=0x00FF
- Mô thức FPWM2- 9 bit WGM13:WGM10=0010, TOP=0x01FF
- Mô thức FPWM3- 10 bit WGM13:WGM10=0011, TOP=0x03FF
- Mô thức FPWM10 WGM13:WGM10=1010, TOP=ICR1, cờ ICF1=1 tại TOP
- Mô thức FPWM11 WGM13:WGM10=1011, TOP=OCR1A, cờ OCF1A=1 tại TOP

Trong cả 5 mô thức trên OCR1x cập nhật tại TOP, TOV1=1 tại BOTTOM

Giá trị đặt trước tối thiểu cho OCR1x hay ICR1=0x0003.

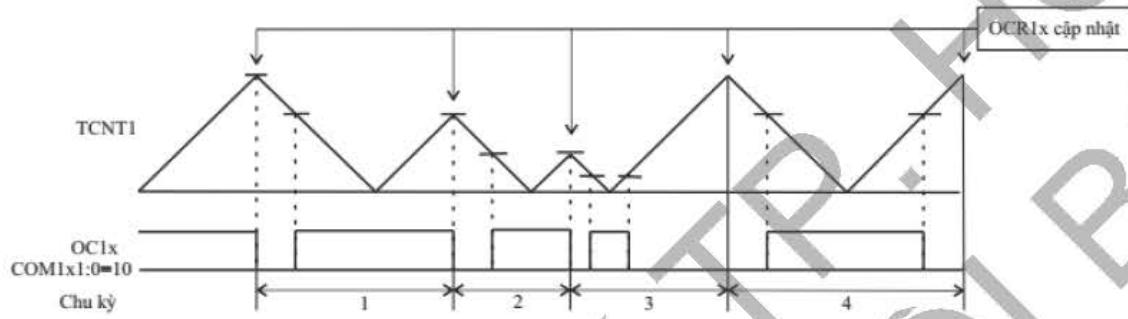
Các trạng thái ngõ ra OC1A/B theo COM1x1:COM1x0 ở Bảng 7.23.

Tần số xung ngõ ra và độ rộng xung trạng thái không đảo hay đảo được tính như công thức (7.7),(7.8),(7.9).

➤ **Lưu ý:** - Trong các mô thức PCPWM10,11 TOP=ICR1/OCR1A,khi thay đổi TOP hoặc OCR1A/B phải đảm bảo  $TOP \geq OCR1x$  mới xảy ra kết quả so sánh.

- Do  $OCR1x$  được cập nhật tại TOP, chu kỳ 1 xung được tính giữa 2 điểm TOP liên tiếp nhau, thời đoạn độ dốc đi xuống phụ thuộc vào giá trị TOP trước đó, còn thời đoạn độ dốc đi lên phụ thuộc vào giá trị TOP mới. Cho nên khi thay đổi TOP khi Timer đang chạy, việc thay đổi  $OCR1x$  có thể làm xung ngõ ra bị mất đổi xứng về mặt tần số/chu kỳ, mặc dù độ rộng xung vẫn đổi xứng quanh điểm BOTTOM.

Trong hình 7.57 minh họa trường hợp TOP và  $OCR1x$  thay đổi khi Timer đang chạy. Ở chu kỳ 3, xung ra bị mất đổi xứng tần số/chu kỳ, do  $OCR1x$  được cập nhật tại TOP. Việc này hoàn toàn có thể khắc phục bằng mô thức PFCPWM trình bày ở phần sau.



Hình 7.57: Việc thay đổi TOP khi Timer đang chạy gây ra mất đổi xứng tần số/chu kỳ

### 7.8.6 Lập trình Timer1 mô thức PFCPWM tạo sóng ngõ ra

Mô thức PCPWM của Timer1 hiệu chỉnh cả pha và tần số, có 2 mô thức PFCPWM như sau:

- Mô thức FPWM 8 WGM13:WGM10=1000, TOP=ICR1, cờ ICF1=1 tại TOP
- Mô thức FPWM 9 WGM13:WGM10=1001, TOP=OCR1A, cờ OCF1 A=1 tại TOP

Trong cả 2 mô thức trên  $OCR1x$  cập nhật tại BOTTOM, TOV1=1 tại BOTTOM

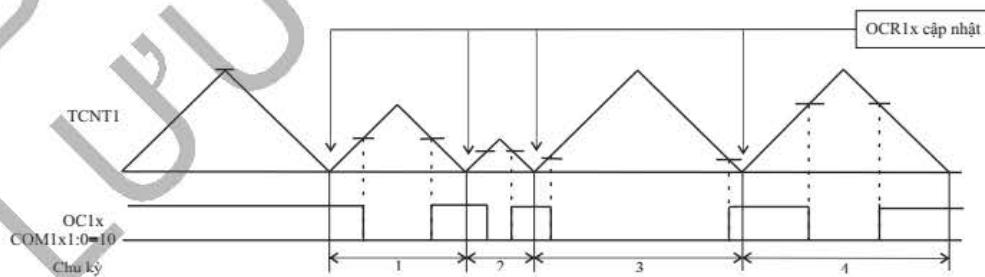
Giá trị đặt trước tối thiểu cho  $OCR1x$  hay  $ICR1=0x0003$ .

Các trạng thái ngõ ra OC1A/B theo COM1x1:COM1x0 ở Bảng 7.23.

Tần số xung ngõ ra và độ rộng xung trạng thái không đảo hay đảo được tính như công thức (7.7),(7.8),(7.9).

Mô thức PFCPWM hoạt động hoàn toàn tương tự như mô thức PCPWM, chỉ khác ở điểm thanh ghi  $OCR1x$  cập nhật tại BOTTOM. Điều này khắc phục được vấn đề mất đổi xứng tần số/chu kỳ có thể xảy ra trong mô thức PCPWM ở trên, dẫn đến hiệu chỉnh được cả pha và tần số xung ra.

Hình 7.58 minh họa tần số/chu kỳ xung ra luôn đổi xứng trong 1 chu kỳ xung là thời đoạn giữa 2 điểm BOTTOM liên tiếp nhau. Độ rộng xung(xung âm trạng thái không đảo) đổi xứng quanh điểm TOP.



Hình 7.58: Mô thức PFCPWM hiệu chỉnh được cả pha và tần số xung ra.

**Ví dụ 7.27:** Viết một chương trình tạo chuỗi xung vuông tuần hoàn gồm 4 xung có chu kỳ và độ rộng thay đổi theo như Bảng 7.24.

Bảng 7.24: Chuỗi xung ví dụ 7.27

Xung	1	2	3	4
Chu kỳ T(ms)	10	15	20	25
Dộ rộng Tp(%T)	10	25	50	75

Giải:

Ta sử dụng Timer1 mô thức PFCPWM9,hệ số chia N=8,1 xung CLKT1=1μs.

- Thanh ghi OCR1A=TOP được tính theo công thức (7.7) để tạo chu kỳ T cho bộ đếm 2 độ dốc

- Thanh ghi OCR1B tạo Tp, chọn ngõ ra OC1B(PD4) đảo,tính OCR1B theo công thức (7.9)

Ta có Bảng 7.24b tính các giá trị đặt OCR1A,OCR1B tương ứng.

Bảng 7.24b: Các giá trị đặt OCR1A/B

Xung	1	2	3	4
Chu kỳ T(ms)	10	15	20	25
OCR1A	5000	7500	10000	12500
Dộ rộng Tp(%T)	10	25	50	75
Tp(ms)	1	3.75	10	18.75
OCR1B	4500	5625	5000	3125

Do các thanh ghi OCR1A/B cập nhật tại điểm BOTTOM,nên ta chọn điểm TOP trước đó làm điểm ghi giá trị mới vào các thanh ghi OCR1A/B(thực chất là ghi vào bộ đếm),tương ứng cờ OCF1A=1.

Để có thể quan sát giản đồ xung định thì như hình 7.58,ta đánh dấu 4 chu kỳ của 4 xung tương ứng bằng cách tạo 1 tín hiệu tham chiếu đảo bit mỗi lần bộ đếm về BOTTOM,tương ứng cờ TOV1=1.

Ngoài ra còn một tín hiệu tham chiếu thứ hai,đảo bit khi bộ đếm đến TOP,bằng cách đặt COM1A1:COM1A0=01(mô thức PFCPWM9 theo bảng 7.23)

Trong chương trình ta đặt P\_A=xung tham chiếu TOP,P\_B=xung ra theo bảng 7.24b, P\_C= xung tham chiếu BOTTOM

#### ❖ Chương trình hợp ngữ ví dụ 7.27

```

.EQU P_A=5          ;ký hiệu OC1A,tham chiếu TOP
.EQU P_B=4          ;ký hiệu OC1B,xung ra
.EQU P_C=6          ;ký hiệu xung tham chiếu BOTTOM
.ORG 0
RJMP MAIN
.ORG 0X40
MAIN: LDI R16,HIGH(RAMEND) ;đưa stack lên đỉnh SRAM
      OUT SPH,R16
      LDI R16,LOW(RAMEND)
      OUT SPL,R16
      LDI R16,(1<<P_A)|(1<<P_B)|(1<<P_C);đặt P_A,P_B,P_C output
      OUT DDRD,R16
      LDI R16,4          ;bộ đếm số lần cập nhật giá trị đặt OCR1A
      LDI XH,HIGH(TOP_TAB<<1);X trả địa chỉ đầu bảng tra TOP
      LDI XL,LOW(TOP_TAB<<1)
      LDI YH,HIGH(TP_TAB<<1) ;Y trả địa chỉ đầu bảng tra giá trị đặt OCR1B
      LDI YL,LOW(TP_TAB<<1)
      LDI R19,(1<<P_C)
      LDI R17,0B01110001   ;Timer1 mode PFCPWM 9,OCR1A đảo bit,OC1B đảo
      STS TCCR1A,R17
      LDI R17,0B00010010   ;Timer1 chạy,mode PFCPWM 9,hệ số chia N=8
      STS TCCR1B,R17
START: RCALL GET_PREDAT ;ctc tính giá trị đặt trước TOP=OCR1A,OCR1B
      RJMP START           ;lặp vòng lại
;
GET_PREDAT:
      IN    R17,TIFR1       ;đọc cờ TOV1

```

SBRS	R17,TOV1	;cờ TOV1=1 xử lý tiếp
RJMP	OCR1A_CHK	;kiểm tra lại cờ OCF1A
OUT	TIFR1,R17	;xóa cờ TOV1
IN	R17,PORTD	;đọc PortD
EOR	R17,R19	;đảo bit P_C
OUT	PORTD,R17	;xuất ra PortD
<b>OCR1A_CHK:</b>		;đọc cờ OCF1A
IN	R17,TIFR1	;cờ OCF1A=1 xử lý tiếp
SBRS	R17,OCF1A	;kiểm tra cờ TOV1 lại
RJMP	GET_PREDAT	;xóa cờ OCF1B
OUT	TIFR1,R17	;Z trả địa chỉ đầu bảng tra giá trị đặt OCR1B
MOVW	ZL,YL	;lấy byte thấp giá trị cập nhật OCR1B,Z trả byte cao
LPM	R17,Z+	;cắt byte thấp
MOV	R18,R17	;lấy byte cao giá trị cập nhật OCR1B,Z trả word kế tiếp
LPM	R17,Z+	;cắt địa chỉ tra bảng OCR1B kế tiếp
MOVW	YL,ZL	;cập nhật byte cao OCR1B
STS	OCR1BH,R17	;cập nhật byte thấp OCR1B
STS	OCR1BL,R18	;Z trả địa chỉ đầu bảng tra TOP
MOVW	ZL,XL	;lấy byte thấp giá trị cập nhật TOP,Z trả byte cao
LPM	R17,Z+	;cắt byte thấp
MOV	R18,R17	;lấy byte cao giá trị cập nhật TOP,Z trả word kế tiếp
LPM	R17,Z+	;cắt địa chỉ tra bảng TOP kế tiếp
MOVW	XL,ZL	;cập nhật byte cao OCR1A
STS	OCR1AH,R17	;cập nhật byte thấp OCR1A
STS	OCR1AL,R18	;đếm số lần cập nhật
DEC	R16	;thoát nếu chưa đủ số lần cập nhật
BRNE	EXIT_PRE	;nạp lại bộ đếm
LDI	R16,4	
LDI	XH,HIGH(TOP_TAB<<1);X trả địa chỉ đầu bảng tra TOP	
LDI	XL,LOW(TOP_TAB<<1)	
LDI	YH,HIGH(TP_TAB<<1);Y trả địa chỉ đầu bảng tra giá trị đặt OCR1B	
LDI	YL,LOW(TP_TAB<<1)	

**EXIT\_PRE:** RET

---

TOP_TAB:	.DW	5000,7500,10000,12500
TP_TAB:	.DW	4500,5625,5000,3125

### ❖ Chương trình C ví dụ 7.27

```
#include <avr/io.h>
#define outport PORTD
const char p_a=5,p_b=4,p_c=6 ;// ký hiệu vị trí bit ngõ ra
unsigned char i;
unsigned int TOP_tab[]={5000,7500,10000,12500};//khai báo bảng tra TOP
unsigned int TP_tab[]={4500,5625,5000,3125};//khai báo bảng tra TP

int main() // bắt đầu chương trình
{
    DDRD |=(1<<p_a)|(1<<p_b)|(1<<p_c) ;//khai báo p_a,p_b,p_c output
    TCCR1A=0b01110001 ;//Timer1 mode PFCPWM 9,OC1A đảo bit,OC1B đảo
    TCCR1B=0b00010010 ;//Timer1 mode PFCPWM 9,hệ số chia N=8, chạy Timer1
    while(1) // lập vòng vô hạn
    {
        for(i=0;i<=3;i++)
        {
            while(!(TIFR1&(1<<TOV1))) ;//chờ cờ TOV1=1
```

```

TIFR1 |= (1<<TOV1)           ;//xóa cờ TOV1
outport^=(1<<p_c)            ;//đảo bit p_c
while(!(TIFR1&(1<<OCF1A)))   ;//chè cờ OCF1A=1
TIFR1 |= (1<<OCF1A)          ;//xóa cờ OCF1A
OCR1B=TP_tab[i]               ;//nạp giá trị mới Tp
OCR1A=TOP_tab[i]              ;//nạp giá trị mới T
}
}
}

```

- Khi viết chương trình C cần lưu ý đồng bộ việc cập nhật cặp giá trị OCR1A và OCR1B, MCU sẽ cập nhật cặp giá trị này tại điểm BOTTOM. Do đó ta chờ cờ TOV1=1 để nhận dạng điểm BOTTOM trước, MCU cập nhật OCR1A/B ghi trước đó bắt đầu chu kỳ xung hiện tại, sau đó chờ cờ OCF1A=1 để nhận dạng điểm TOP, MCU ghi giá trị mới OCR1A/B chuẩn bị chu kỳ xung kế tiếp.

#### ❖ Câu hỏi ôn tập

1. Viết một đoạn lệnh khởi động Timer1 chạy mô thức CTC tạo xung ra tần số 250Hz,CKNV 35%.
2. Lập lại câu 1 với Timer1 chạy mô thức FPWM
3. Lập lại câu 1 với Timer1 chạy mô thức PCPWM.
4. Mô thức PCPWM và PFCPWM khác nhau ở chỗ nào? Nếu muốn ghi dữ liệu cập nhật mới vào các thanh ghi OCR1x khi nhận dạng cờ TOV=1, nên chọn mô thức nào? Tại sao?
5. Trong các mô thức PWM đặt TOP=ICR1, có thể cập nhật giá trị ICR1 tại thời điểm nào thích hợp?

### 7.9 Ứng dụng Timer làm DAC PWM, tạo sóng tam giác

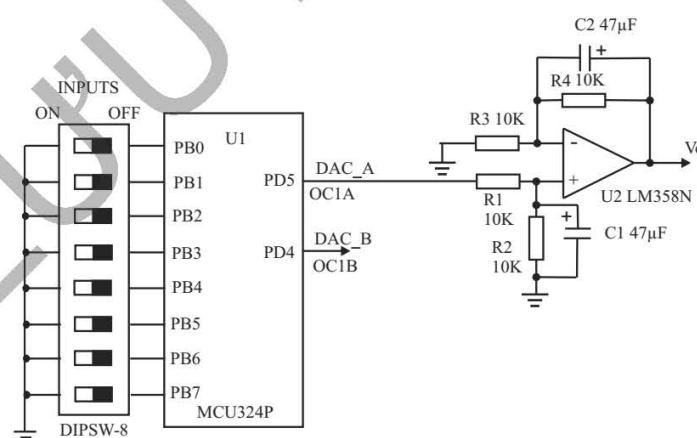
Các thiết bị ngoại vi trong đo lường, điều khiển phần lớn làm việc với tín hiệu tương tự(analog) ví dụ như đồng hồ hiển thị kim, van điện từ, động cơ điều khiển, bộ khuếch đại âm tần, loa... Do đó phần ngoại vi của MCU phải có bộ chuyển đổi tín hiệu số sang tương tự DAC(Digital Analog Converter).

Phần sau đây là các ví dụ ứng dụng Timer làm DAC PWM và tạo sóng tam giác sử dụng rộng rãi trong các ứng dụng đo lường, điều khiển...

**Ví dụ 7.28:** Thiết kế một DAC 8 bit với ngõ vào là số nhị phân 8 bit, ngõ ra là chuỗi xung PWM tần số 1Khz, độ rộng tỉ lệ với giá trị số D ngõ vào. Mạch lọc ngõ ra tạo điện áp DC 0 - 5V tỉ lệ tương ứng giá trị D.

**Giải:**

Mạch thiết kế như hình 7.59. MCU324P làm bộ DAC PWM nhận dữ liệu 8 bit D ngõ vào từ PortB minh họa bằng DIPSW-8, ngõ ra DAC\_A là tín hiệu xung PWM  $f_o = 1\text{Khz}$ , độ rộng tỉ lệ với giá trị số D. Ngõ ra DAC\_B là tín hiệu xung PWM đảo với DAC\_A tạo DAC độ dốc ngược với DAC\_A. Mạch khuếch đại vi sai (mạch trừ không đảo) và lọc U2 tạo điện áp DC ngõ ra  $V_o$  tỉ lệ với giá trị số D. Phần thiết kế mạch lọc không trình bày ở đây, người đọc có thể tham khảo thêm các tài liệu về mạch lọc tích cực dùng OPAMP.



**Hình 7.59:** Ứng dụng Timer làm DAC PWM ví dụ 7.28

Độ rộng xung ngõ ra DAC\_A được tính:

$$T_p = \frac{D}{FS} T = C \frac{N}{F_{osc}} \quad (7.11)$$

$D$ = giá trị số ngõ vào DAC

$FS=255$  giá trị toàn tam 8 bit

$T=1000Hz$  chu kỳ xung PWM

$C=số$  xung Timer đếm được trong thời gian  $T_p$

$$\frac{N}{F_{osc}} = CLK\ Timer$$

Điện áp DC ngõ ra được tính:

$$V_o = \frac{T_p}{T} V = \frac{D}{FS} V \quad (7.12)$$

$V=5V$  biên độ xung ngõ ra DAC.

Từ công thức (7.11) và (7.12), để đơn giản ta chọn CLK Timer=1μs( $N=8$ ), ta được:

$$T_p = C = \frac{D}{255} 1000 = (D \times 200)/51 \quad (7.13)$$

Sử dụng Timer1 chạy mô thức FPWM14,OC1A không đảo,OC1B đảo,hệ số chia N=8:

- Tần số fo=1000Hz, giá trị đặt ICR1=TOP=999

- Giá trị đặt OCR1A=OCR1B tính theo công thức (7.13).

### ❖ Chương trình hợp ngữ ví dụ 7.28

```

.DEF OPD1 L=R20
.DEF OPD1_H=R21
.DEF OPD2=R22
.DEF OPD3=R23
.DEF COUNT=R18
.EQU INPORT=PINB
.EQU DAC_A=5 ;ký hiệu OC1A
.EQU DAC_B=4 ;ký hiệu OC1B
.EQU P_COUNT=999 ;ký hiệu giá trị đặt ICR1
.EQU FSC=51 ;ký hiệu giá trị đầy thang FS/5
.EQU T_MUL=200 ;ký hiệu giá trị T/5
.ORG 0
RJMP MAIN
.ORG 0X40
MAIN: LDI R16,HIGH(RAMEND) ;đưa stack lên đỉnh SRAM
      OUT SPH,R16
      LDI R16,LOW(RAMEND)
      OUT SPL,R16
      LDI R16,(1<<DAC_A)|(1<<DAC_B) ;đặt các ngõ output
      OUT DDRD,R16
      LDI R16,0 ;PortB input
      OUT DDRB,R16
      LDI R16,0XFF ;điện trở kéo lên PortB
      OUT PORTB,R16
      LDI R16,T_MUL ;nạp R16 hệ số nhân
      LDI R17,HIGH(P_COUNT)
      STS ICR1H,R17 ;nạp byte cao ICR1
      LDI R17,LOW(P_COUNT)
      STS ICR1L,R17 ;nạp byte thấp ICR1
      LDI R17,0B10110010 ;Timer1 mode FPWM 14,OC1A không đảo,OC1B đảo
      STS TCCR1A,R17
      LDI R17,0B00011010 ;Timer1 chạy,mode FPWM 14,hệ số chia N=8

```

```

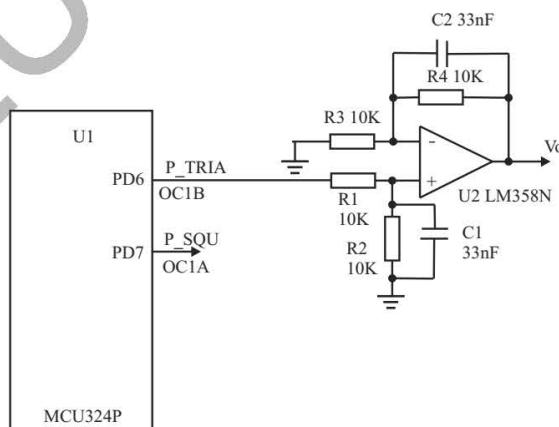
STS      TCCR1B,R17
START:   RCALL GET_DAC
        RJMP  START
;
;-----;
GET_DAC: IN     R17,INPUT
        MUL   R17,R16
        MOVW OPD1_L,R0
        LDI    OPD2,FSC
        RCALL DIV16_8
        STS   OCR1AH,OPD1_H
        STS   OCR1BH,OPD1_H
        STS   OCR1AL,OPD1_L
        STS   OCR1BL,OPD1_L
;
EXIT_DAC: RET
;
;-----;
;DIV16_8 chia số nhị phân 16 bit OPD1 cho 8 bit OPD2
;Input: OPD1_H,OPD1_L= SBC(GPR16-31)
;      OPD2=SC(GPR0-31)
;Output:OPD1_H,OPD1_L=thương số
;      OPD3=DS(GPR0-31)
;Sử dụng COUNT(GPR16-31)
;
DIV16_8: LDI    COUNT,16      ;COUNT=đếm 16
        CLR    OPD3       ;xóa dư số
SH_NXT: CLC
        LSL    OPD1_L       ;dịch trái SBC L,bit0=C=thương số
        ROL    OPD1_H       ;quay trái SBC H,C=bit7
        ROL    OPD3         ;dịch bit7 SBC H vào dư số
        BRCS  OV_C         ;trần bit C=1,chia được
        SUB   OPD3,OPD2     ;trừ dư số với số chia
        BRCC  GT_TH        ;C=0 chia được
        ADD   OPD3,OPD2     ;C=1 không chia được,không trừ
        RJMP  NEXT
OV_C:   SUB   OPD3,OPD2     ;trừ dư số với số chia
GT_TH:  SBR   OPD1_L,1      ;chia được,thương số=1
NEXT:   DEC   COUNT         ;đếm số lần dịch SBC
        BRNE  SH_NXT       ;chưa dù tiếp tục dịch bit
        RET
;

```

**Ví dụ 7.29:** Thiết kế mạch tạo sóng tam giác tần số 100Hz.

**Giải:**

Sơ đồ mạch hình 7.60 tương tự như hình 7.59, chỉ khác về các giá trị tụ điện lọc tần số cao.



**Hình 7.60:** Sơ đồ mạch ví dụ 7.29

Theo công thức (7.6) ví dụ 7.23, điện áp trung bình ngõ ra Vo tỉ lệ tuyến tính độ rộng xung Tp. Do đó trong thời đoạn 1 độ dốc lên của xung tam giác ta chia thành M xung PWM độ rộng tăng dần theo tỉ lệ :

$$T_{pi} = \Delta T \times i \quad (i = 0 \div M) \quad (7.14)$$

$T_{pi}$  = độ rộng xung thứ i

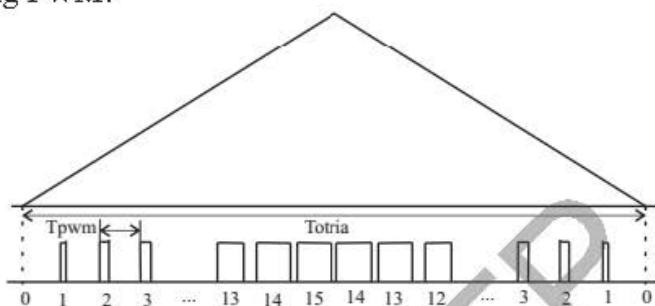
$\Delta T$  = hằng số tỉ lệ

Thời đoạn độ dốc xuống cũng chia thành M xung PWM độ rộng giảm dần theo tỉ lệ như vậy với i chạy từ M về 0. Hình 7.61 minh họa cách tạo số xung PWM trong 1 chu kỳ xung tam giác với M=15.

Gọi  $f_{PWM}$  là tần số xung PWM thay đổi độ rộng xung, tần số xung tam giác ngõ ra được tính:

$$f_{otria} = f_{PWM}/2M \quad (7.15)$$

Ta sử dụng Timer2 mô thức PCPWM5, hệ số chia N=8, CLKTF2=1μs, OC2B không đảo, OC2A đảo bit để theo dõi chu kỳ xung PWM:



Hình 7.61: Tạo số xung PWM trong 1 chu kỳ xung tam giác với M=15

- Chọn M=15, để có xung tam giác ngõ ra  $f_{otria}=100Hz$  theo công thức (7.15)  $f_{PWM}=3Khz$ .
- Theo công thức (7.7), để tạo xung PWM tần số 3Khz OCR1A=TOP=166
- Trong thời đoạn độ dốc lên dài  $T_{otria}/2=5ms$  có 15 chu kỳ xung PWM  $T_{PWM}=333\mu s$
- Theo công thức (7.14) với M=15,  $T_{PWM}=333\mu s$ , tính được  $\Delta T=22\mu s$
- Theo công thức (7.8) và (7.14) trong thời đoạn độ dốc lên xung tam giác, lần lượt xuất 15 xung PWM với độ rộng 22xi( $\mu s$ ), tương ứng giá trị đặt OCR1B=11xi ( $i=1 - 15$ ) (CLKTF2=1μs)
- Tương tự với thời đoạn độ dốc xuống xung tam giác, giá trị đặt OCR1B=11xi ( $i=14 - 0$ )
- Do các thanh ghi OCR2A/B đều cập nhật tại TOP, nên ta ghi giá trị mới cho OCR1B tại BOTTOM tương ứng cờ TOV2=1 để đồng bộ việc cập nhật OCR2B chính xác!

#### ❖ Chương trình hợp ngữ ví dụ 7.29

```

.DEF COUNT=R18
.EQU P_TRIA=6          ;ký hiệu OC2B
.EQU P_SQU=7            ;ký hiệu OC2A
.EQU P_COUNT= 166       ;ký hiệu giá trị đặt OCR2A
.EQU DELTA=11           ;ký hiệu số gia
.ORG 0
RJMP MAIN
.ORG 0X40
MAIN: LDI R16,HIGH(RAMEND) ;đưa stack lên đỉnh SRAM
      OUT SPH,R16
      LDI R16,LOW(RAMEND)
      OUT SPL,R16
      LDI R16,(1<<P_TRIA)|(1<<P_SQU) ;đặt các ngõ output
      OUT DDRD,R16
      LDI R16,DELTA          ;nạp R16 số gia
      LDI R19,0                ;R19 chứa giá trị đặt OCR2B
      LDI COUNT,15             ;bộ đếm số lần cập nhật OCR2B
      LDI R17,P_COUNT
      STS OCR2A,R17            ;nạp OCR2A
      LDI R17,0B01100001       ;Timer2 mode PCPWM5,OC2A đảo bit,OC2B không đảo
      STS TCCR2A,R17

```

```

LDI    R17,0B00001010 ;Timer1 chạy, mode PCPWM5, hệ số chia N=8
STS    TCCR2B,R17
START: RCALL GET_PULSE ;ctc tính giá trị đặt trước OCR2B
        RJMP START ;lặp vòng lại
;-----;
GET_PULSE: IN   R17,TIFR2 ;đọc cờ OCF2A
            SBRS R17,OCF2A ;cờ OCF2A=1 cập nhật OCR2B
            RJMP GET_PULSE ;xóa cờ OCF2A
            OUT  TIFR2,R17 ;cập nhật OCR2B
            ADD   R19,R16 ;đếm đủ số lần cập nhật đếm lên
            STS   OCR2B,R19
            DEC   COUNT
            BRNE GET_PULSE
            LDI   COUNT,15
CT_DWN:  IN   R17,TIFR2 ;đọc cờ OCF2A
            SBRS R17,OCF2A ;cờ OCF2A=1 cập nhật OCR2B
            RJMP CT_DWN ;xóa cờ OCF2A
            OUT  TIFR2,R17 ;cập nhật OCR2B
            SUB   R19,R16 ;đếm đủ số lần cập nhật đếm lên
            STS   OCR2B,R19
            DEC   COUNT
            BRNE CT_DWN
            LDI   COUNT,15
            RET

```

Phần chương trình C ví dụ 7.28 và 7.29 dành cho người đọc tự soạn thảo.

Một phương pháp tạo sóng tam giác khác là lần lượt xuất giá trị cập nhật OCR2B ra Port, ví dụ PortB, tương ứng mỗi chu kỳ xung PWM và sử dụng IC DAC giao tiếp PortB tạo điện áp Vo dạng tam giác. Trong chương 11 sẽ trình bày phương pháp này.

## BÀI TẬP CHƯƠNG 7

Trong các bài tập sau đây, chọn Fosc=8Mhz mặc định, trừ trường hợp cụ thể sẽ có ghi chú riêng.

### ❖ Phần cơ bản

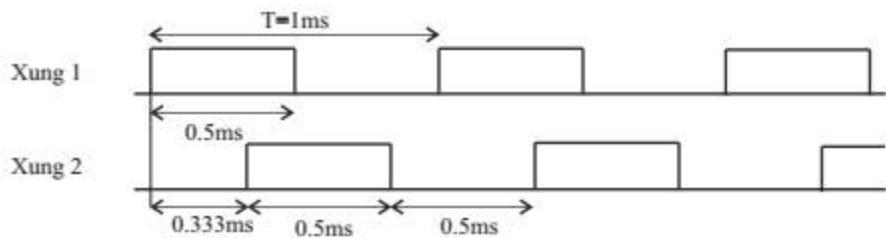
- Viết một chương trình tạo chuỗi xung vuông đối xứng ngõ ra PB0 tần số 20KHz, sử dụng Timer0:
  - Mô thức NOR
  - Mô thức CTCĐánh giá sai số xung tạo ra thực tế.
- Viết một chương trình tạo chuỗi xung vuông đối xứng ngõ ra PC0 tần số 100Hz, sử dụng Timer0:
  - Mô thức NOR
  - Mô thức CTCĐánh giá sai số xung tạo ra thực tế.
- Lập lại bài 2 với Timer2
- Lập lại bài 2 với Timer1
- Viết một chương trình tạo chuỗi xung vuông tần số 400Hz CKNV 35% ngõ ra PB5, sử dụng Timer0:
  - Mô thức NOR
  - Mô thức CTCĐánh giá sai số xung tạo ra thực tế.
- Lập lại bài 5 với Timer2.
- Lập lại bài 5 với Timer1.
- Dựa vào hình 7.15 ví dụ 7.6, viết một chương trình thực hiện khi nhấn/nhả SW LED sáng trong 2s:
  - Sử dụng Timer0 tạo thời gian delay
  - Sử dụng Timer1 tạo thời gian delay
  - Sử dụng Timer2 với bộ dao động thạch anh ngoại fXTal=32768Hz
- Vẽ sơ đồ MCU324P giao tiếp 4 SW khi nhấn tích cực mức 0 và 1 LED đơn (phản cực dòng 10mA khi LED sáng). Viết một chương trình sử dụng Timer tạo thời gian delay, thực hiện các công việc như sau:
  - Nhấn SW1 LED sáng trong 2s
  - Nhấn SW2 LED sáng chớp tắt 11 lần trong 2s
  - Nhấn SW3 LED sáng chớp tắt 21 lần trong 2s
  - Nhấn SW4 LED sáng chớp tắt 41 lần trong 2sLặp vòng liên tục các công việc trên.
- Viết một chương trình tạo chuỗi xung tuần hoàn có độ rộng thay đổi như hình BT7.10:



- Viết một chương trình tạo 2 chuỗi xung vuông đối xứng tần số f1=1Khz, tần số f2=2Khz:
  - Sử dụng Timer0 mô thức CTC xuất xung f1 ra PD0, Timer2 mô thức CTC xuất xung f2 ra PD1
  - Sử dụng Timer0 và Timer2 như câu (a) nhưng xuất xung f1 ra OC0A, xuất xung f2 ra OC2A
- Lập lại bài 11 nhưng chỉ sử dụng Timer0 (ngõ ra tùy chọn)
- Thiết kế mạch tạo tín hiệu xung PWM tần số fo=3Khz, độ rộng xung thay đổi được từ 1 - 255μs bằng cách cài đặt DIPSW-8 ở ngõ vào. Sử dụng Timer0 mô thức FPWM, ngõ ra tùy chọn.
- Lập lại bài 13, sử dụng Timer2 mô thức PCPWM.
- Dựa vào hình 7.51 ví dụ 7.23, thiết kế mạch tạo tín hiệu xung PWM tần số fo=200Hz, CKNV thay đổi từ 5% - 95% bằng cách nhấn SW ở ngõ vào, mỗi lần nhấn SW1 tăng 1%, nhấn SW2 giảm 1%. Sử dụng Timer1 mô thức FPWM, ngõ ra tùy chọn.
- Lập lại bài 15, sử dụng Timer1 mô thức PCPWM.

### ❖ Phần bài tập nâng cao

- Viết một chương trình tạo 2 chuỗi xung vuông như hình BT7.17.



Hình BT7.17: Dạng xung bài tập 7.17

18. Dựa vào bài tập 16,hình 7.53 và ví dụ 7.24, thiết kế mạch tạo 2 xung ra như hình 7.53 có tần số 50Hz, CKNV thay đổi được từ 5% - 95% bằng 2 SW nhấn như bài tập 16.

19. \*Từ bài 18, thêm 2 SW thay đổi tần số từ 5 - 50Hz, mỗi lần nhấn SW3 tăng 1Hz, nhấn SW4 giảm 1Hz. Thay đổi CKNV bằng SW1(tăng),SW2(giảm) như bài tập 18.

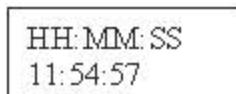
20. \*Thiết kế 1 đồng hồ hiển thị giờ, phút, giây hoạt động như sau:

- Sử dụng dao động thạch anh chuẩn  $f_{XTal}=32768\text{Hz}$

- Cài đặt thời gian bằng 2 SW:

SW1 chọn cài đặt(SETUP): nhấn lần 1 chọn giờ, lần 2 chọn phút, lần 3 chọn giây, lần 4 đếm giờ  
SW2 chọn đếm số(COUNT): nhấn tăng 1 đơn vị cho đến 59 hoặc 11 tùy cài đặt giây, phút hay giờ

- Hiển thị ra LCD ký tự 16x2 như sau:



Ở trạng thái cài đặt phải sáng chớp số giờ, phút hoặc giây đang cài đặt tương ứng.

21. Dựa vào ví dụ 7.14,7.15, thiết kế mạch đo chu kỳ độ rộng xung từ 1 - 50000 $\mu\text{s}$ :

- Một công tắc chọn đo độ rộng xung hay chu kỳ xung

- Hiển thị bằng LCD 16x2 tương tự như ví dụ 7.15, ghi chú đo độ rộng hay đo chu kỳ tương ứng.

22. \*Dựa vào ví dụ 7.16 và hình 7.42, mở rộng thêm mạch đếm sản phẩm có 2 mô thức làm việc:

- Một công tắc chọn mô thức đếm tự do/dếm đắt trước

- Đếm tự do: mạch đếm liên tục từ 0 - 99 và tràn về 0

- Đếm đắt trước: thêm 2 SW đặt trước số sản phẩm từ 1 - 99, mỗi lần nhấn SW1 tăng 1, nhấn SW2 giảm 1, bộ đếm đền giá trị đặt trước sẽ dừng đếm.

- SW khởi động xóa bộ đếm và đếm theo mô thức chọn bởi công tắc

23. \*Dựa trên ví dụ 7.17, thiết kế mạch đo tốc độ động cơ có cảm biến tốc độ là bộ ghép quang gắn cùng trực động cơ. Cảm biến phát 100 xung/vòng quay. Hiển thị tốc độ động cơ theo vòng/phút ký hiệu rpm.

24. Dựa trên ví dụ 7.23 và hình 7.51, thiết kế mạch điều khiển tốc độ động cơ DC như ví dụ 7.23 sử dụng mô thức PFCPWM.

25. \*Từ bài 7.23 và 7.24, thiết kế mạch đo và điều khiển tốc độ động cơ DC, hiển thị tốc độ ra LCD 16x2.

26. Dựa vào ví dụ 7.29, thiết kế mạch tạo sóng răng cưa tần số 3Khz.