

Chương 4

Vi điều khiển PIC16F877A

PIC là viết tắt của *Programable Intelligent Computer* do hãng General Instrument, sau là Microchip Technology chế tạo. Dòng chip đầu tiên là PIC1650 được thiết kế để dùng làm các thiết bị ngoại vi cho vi điều khiển CP1600. Sau đó, chip PIC1650 được phát triển và hình thành nên dòng vi điều khiển PIC ngày nay. Tại thị trường công nghệ Việt Nam, bên cạnh các họ vi điều khiển khác như 8051, Motorola 68HC, AVR, ARM,...PIC là một họ vi điều khiển phổ biến bởi các nguyên nhân:

- Có thể tìm mua dễ dàng với giá thành không đắt.
- Có nhiều tính năng khi hoạt động độc lập.
- Số lượng người dùng tại Việt Nam ngày càng tăng kéo theo sự thuận lợi trong quá trình tìm hiểu, sử dụng.
- Các tính năng của vi điều khiển PIC không ngừng được phát triển.

Tùy theo từng nhu cầu phát triển hệ thống, họ vi điều khiển PIC có các loại 8-bit, 16-bit, 32-bit hoặc ds-PIC chuyên dụng cho xử lý tín hiệu số. Trong lĩnh vực điều khiển, tự động hóa và trong các bài toán điện tử không xử lý lượng dữ liệu lớn, 8-bit PIC là loại phổ biến nhất. Một số loại vi điều khiển PIC 8-bit gồm:

PIC12xxxx: độ dài lệnh 12 bit

PIC16xxxx: độ dài lệnh 14 bit

PIC18xxxx: độ dài lệnh 16 bit

Ngôn ngữ lập trình cho PIC rất đa dạng gồm ngôn ngữ lập trình cấp thấp MPLAB được cung cấp miễn phí bởi nhà sản xuất Microchip, các ngôn ngữ lập trình cấp cao C, Basic, Pascal và một số ngôn ngữ lập trình được phát triển riêng cho PIC như PICBasic hoặc MikroBasic.

Mạch nạp cho PIC cũng rất đa dạng. Người dùng có thể sử dụng các mạch nạp được cung cấp bởi nhà sản xuất là hãng Microchip như PICSTART plus, MPLAB ICD2, MPLAB PM3,...hoặc mạch nạp được phát triển bởi một bên thứ 3 như P16PRO40 của Nigel hoặc tự lắp ráp mạch nạp một cách khá đơn giản, dễ dàng.

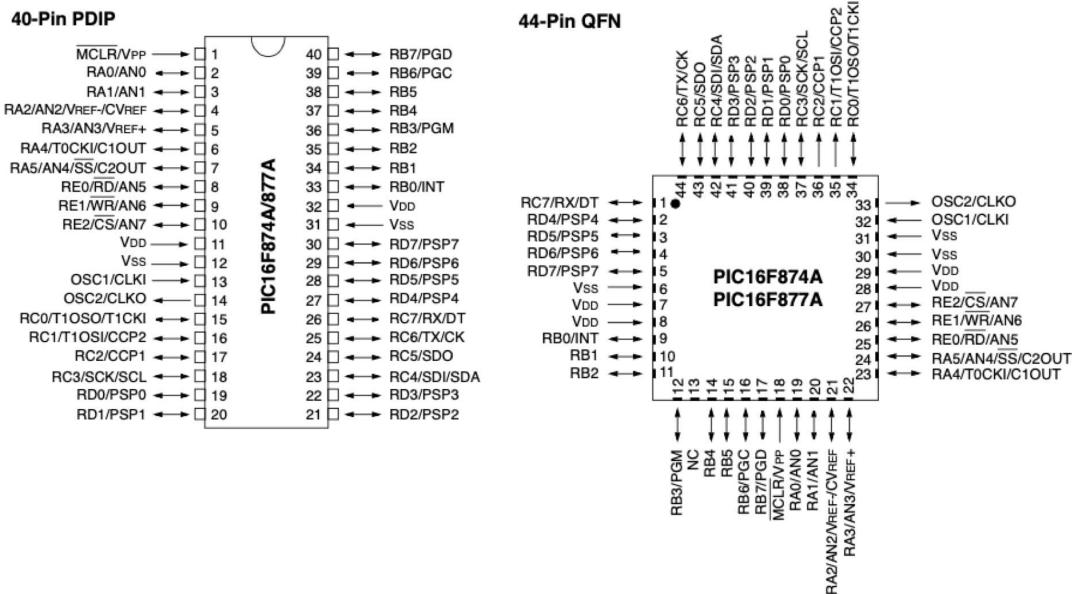
Dể người đọc tiếp cận họ vi điều khiển PIC, các phần nội dung tiếp theo dưới đây sẽ đi sâu tìm hiểu về PIC16F877A, một con chip rất phổ biến và dễ dàng tìm mua trên thị trường.

4.1 Chức năng cơ bản vi điều khiển PIC16F877A

Vi điều khiển PIC16F877A là vi điều khiển thuộc họ PIC16Fxxx với tập lệnh gồm 35 lệnh. Mỗi lệnh có độ dài 14 bit được thực thi trong một chu kỳ xung clock. Tốc độ hoạt động tối đa cho phép là 20 MHz tương ứng với một chu kỳ lệnh 200 ns. PIC16F877A có bộ nhớ chương trình 8K x 14 bit, bộ nhớ dữ liệu 368 x 8 byte RAM và bộ nhớ dữ liệu EEPROM với dung lượng 256 x 8 byte. Số cổng ngoại vi (IO Port) là 5 với 33 chân vào/ra dữ liệu (I/O pin). PIC16F877A thường được đóng gói dưới dạng 2 hàng chân hoặc 4 hàng chân (Hình 4.1).

Các đặc tính ngoại vi cơ bản của PIC16F877A gồm có:

- Timer0: 8 bit timer/counter với bộ chia trước 8 bit.
- Timer1: 16-bit timer/counter với bộ chia trước có thể hoạt động trong chế độ Sleep với sự tác động từ xung nhịp ngoài.
- Timer2: 8-bit timer/counter với bộ chia trước prescaler, bộ chia sau postscaler và giá trị đếm có thể đặt trước.



Hình 4.1: Sơ đồ chân của PIC16F877A

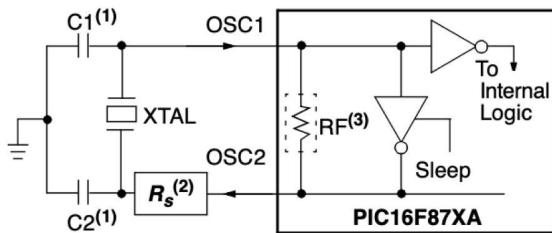
-
- 02 bộ Capture/Compare/PWM.
 - Các giao thức truyền thông nối tiếp SSP như SPI, I2C.
 - Giao thức nối tiếp USART với 9 bit địa chỉ.
 - Giao tiếp song song PSP với các chân điều khiển RD, WR, CS ở bên ngoài.
 - 08 kênh chuyển đổi tương tự - số ADC 10 bit.
 - Chế độ Brown-out reset BOR.
 - Bộ nhớ Flash với khả năng ghi/xoá được 100.000 lần.
 - Bộ nhớ EEPROM với khả năng ghi/xoá được 1.000.000 lần.
 - Dữ liệu bộ nhớ EEPROM lưu trữ lên tới 40 năm.
 - Tự nạp chương trình điều khiển bằng phần mềm.
 - Đọc/ghi chương trình thời gian thực.
 - Watchdog timer với bộ dao động trong.
 - Chức năng bảo mật mã chương trình.

Mạch dao động cho PIC16F877A

Với vi xử lý hiện đại mạch xung nhịp của CPU thường lấy từ 3 nguồn: Dao động thạch anh, dao động RC và dao động từ nguồn ngoài.

Dao động thạch anh: Bộ tạo dao động sử dụng thạch anh bên ngoài, phụ thuộc vào loại thạch anh sử dụng sẽ tạo ra tần số dao động có độ chính xác phù hợp. Sai số của tần số dao động tính theo đơn vị ppm là số xung sai lệch trên 1 triệu xung phát ra. Ví dụ thạch anh có độ sai lệch 20 ppm có nghĩa là trong 1 triệu xung mạch dao động tạo ra cho phép sai lệch không quá 20 xung. Có nhiều vi xử lý cho phép 2 nguồn dao động thạch anh độc lập (nguồn dao động tốc độ cao 20 MHz dùng cho CPU thực hiện lệnh và 32,678 KHz dùng cho đồng hồ thời gian thực hoặc chế độ tiết kiệm năng lượng của CPU).

Dao động RC: Bộ tạo dao động dùng mạch RC bên trong vi xử lý. Với mục đích giảm số lượng linh kiện bên ngoài chip, người dùng có thể dùng mạch dao động được thiết kế bên trong chip, việc này cho phép giảm giá thành và tăng tính ổn định của hệ thống (không phải lo ngại thạch anh bên ngoài hỏng hoặc mạch bên ngoài bị tác động nhiều). Người sử dụng có thể cài đặt được tần số dao động xung nhịp theo ý muốn trong mà nhà sản xuất quy định (trong khoảng từ 32 KHz đến 8 MHz). Tuy nhiên giải pháp này có nhược điểm là độ chính xác dao động không cao, để nâng cao độ chính xác nhà sản xuất cho phép hiệu chuẩn tần số dao động, nhưng thông thường sai số tốt nhất đạt khoảng 0,2 %.



Ranges Tested:			
Mode	Freq.	OSC1	OSC2
XT	455 kHz	68-100 pF	68-100 pF
	2.0 MHz	15-68 pF	15-68 pF
	4.0 MHz	15-68 pF	15-68 pF
HS	8.0 MHz	10-68 pF	10-68 pF
	16.0 MHz	10-22 pF	10-22 pF

Osc Type	Crystal Freq.	Cap. Range C1	Cap. Range C2
LP	32 kHz	33 pF	33 pF
	200 kHz	15 pF	15 pF
XT	200 kHz	47-68 pF	47-68 pF
	1 MHz	15 pF	15 pF
	4 MHz	15 pF	15 pF
HS	4 MHz	15 pF	15 pF
	8 MHz	15-33 pF	15-33 pF
	20 MHz	15-33 pF	15-33 pF

Crystals Used			
32 kHz	Epson C-001R32.768K-A	± 20 PPM	
200 kHz	STD XTL 200.000KHz	± 20 PPM	
1 MHz	ECS ECS-10-13-1	± 50 PPM	
4 MHz	ECS ECS-40-20-1	± 50 PPM	
8 MHz	EPSON CA-301 8.000M-C	± 30 PPM	
20 MHz	EPSON CA-301 20.000M-C	± 30 PPM	

Resonators Used:			
2.0 MHz	Murata Erie CSA2.00MG	± 0.5%	
4.0 MHz	Murata Erie CSA4.00MG	± 0.5%	
8.0 MHz	Murata Erie CSA8.00MT	± 0.5%	
16.0 MHz	Murata Erie CSA16.00MX	± 0.5%	

Hình 4.2: Mạch tạo dao động thạch anh cho PIC16F877A

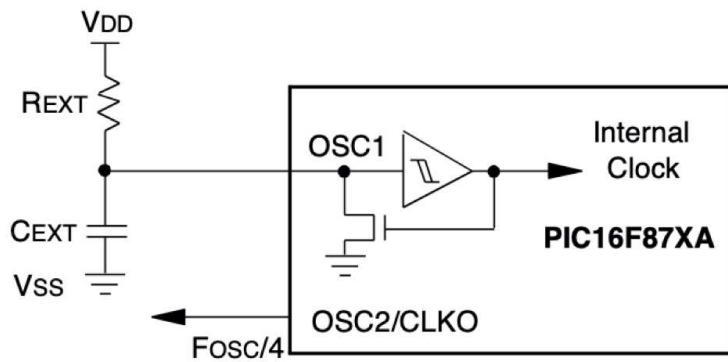
Đao động từ nguồn ngoài: Vì xử lý có thể nhận xung nhịp từ một mạch tạo xung khác từ bên ngoài.

Khi mới được cấp nguồn, bộ tạo dao động được kích hoạt. Tuy nhiên ở giai đoạn đầu, tần số dao động chưa ổn định có thể ảnh hưởng đến sự hoạt động của vi xử lý. Do đó nhà sản xuất đưa ra 1 cơ chế tạo ra 1 khoảng thời gian trước để chắc chắn tần số dao động đã ổn định sau đó mới cấp xung cho hệ thống. Cơ chế này gọi là bộ định thời gian khởi động tạo dao động (OST- Oscillator Start-up), thời gian này có thể thay đổi bởi người lập trình để phù hợp với từng bài toán ứng dụng.

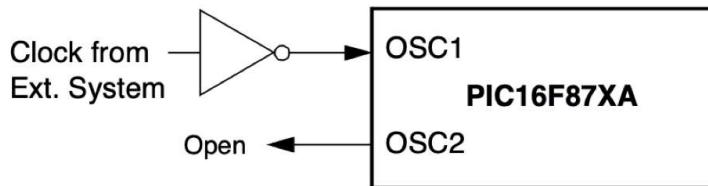
Ngoài ra, do tính chất quan trọng của xung nhịp cấp cho vi xử lý, hệ thống còn có bộ kiểm soát chất lượng của xung nhịp. Trong trường hợp phát hiện ra sự bất thường sẽ tự động chuyển sang sử dụng bộ tạo dao động bên trong, đồng thời báo cho phần mềm lập trình thông qua thanh ghi cờ. Nhờ đó người lập trình có thể đưa thuật toán khởi động lại bộ tạo dao động ngoài và chuyển người cấp xung từ dao động bên trong sang dao động bên ngoài đã được khắc phục.

Các chế độ reset

Khi hoạt động vi xử lý có thể bị reset trong các trường hợp sau:



Recommended values: $3 \text{ k}\Omega \leq R_{\text{EXT}} \leq 100 \text{ k}\Omega$
 $C_{\text{EXT}} > 20 \text{ pF}$



Hình 4.3: Mạch tạo dao động RC và từ nguồn bên ngoài cho PIC16F877A

- Khi bật nguồn (Power-on reset): Khi bật nguồn, điện áp cấp cho vi xử lý tăng từ 0 đến điện áp định mức. Mạch POR sẽ giữ vi xử lý ở trạng thái Reset cho đến khi điện áp nguồn tăng vượt quá 1 giá trị thiết kế trước. Ngoài ra, tín hiệu Reset được kéo dài 1 khoảng thời gian đủ để vi xử lý Reset 1 cách chắc chắn nhờ bộ định thời bật nguồn (Power-up Timer).

- Nhận tín hiệu reset từ bên ngoài (MCLR reset): Chân bên ngoài (\overline{MCLR}) là một chân tín hiệu có thể nhận một tín hiệu từ bên ngoài để Reset vi xử lý. Việc cài đặt tính năng này được thực hiện bằng phần mềm. Khi vi xử lý nhận được tín hiệu ở mức logic phù hợp (\overline{MCLR} ở mức 0) thì vi xử lý sẽ bị Reset.

- Nguồn cấp mất ổn định (Brown-out reset): Nguồn cấp đóng vai trò quan trọng tới sự hoạt động ổn định của vi xử lý. Cấu tạo vi xử lý hiện đại có cơ chế kiểm soát điện áp nguồn khi hệ thống đang hoạt động. Trong trường hợp phát hiện sự bất thường (điện áp nguồn nhỏ hơn một điện áp đặt trước (V_{BOR})) hệ thống có thể báo cho phần mềm lập trình và bắt đầu tiến trình Reset vi xử lý. Khối BOR có 4 chế độ hoạt động:

BOR luôn được kích hoạt.

BOR không hoạt động khi ở chế độ ngủ (Sleep).

BOR được điều khiển bằng phần mềm.

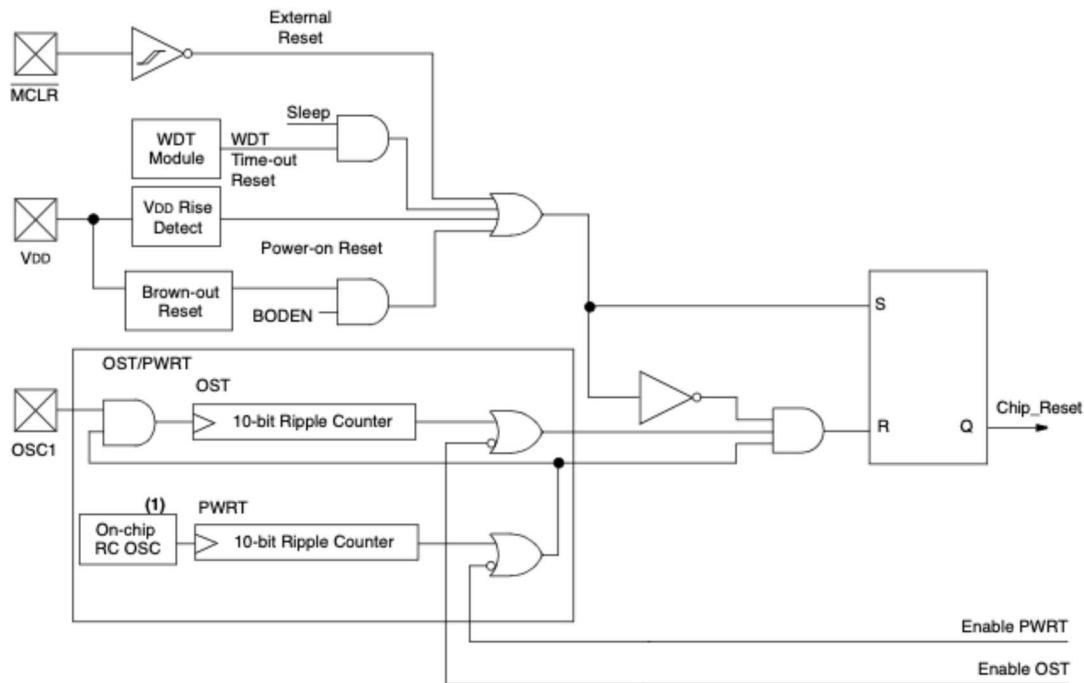
BOR luôn không hoạt động.

Để đảm bảo vi xử lý không bị Reset nhầm, điện áp nguồn trước khi đưa vào bộ xử lý BOR được lọc chống nhiễu. Đồng thời giá trị điện áp nguồn cũng có thể cài đặt bởi người lập trình.

- Watchdog-timer reset (WDT): Trong trường hợp WatchDog Timer bị tràn (chương trình có lỗi) sẽ dẫn đến vi xử lý bị Reset.

- Reset bằng phần mềm: Người dùng có thể sử dụng câu lệnh Reset trong chương trình để khởi động lại vi xử lý.

- Tràn ngăn xếp (Stack overflow): Khi số tầng của ngăn xếp đã đầy (đạt giá trị max mà nhà sản xuất Chip cho phép (với vi xử lý PIC16F18877, độ lớn vùng nhớ ngăn xếp là 8) mà chương trình vẫn tiếp tục gọi trình con (tiếp tục đầy địa chỉ vào ngăn xếp) thì hệ thống sẽ bị reset.
- Tràn bộ đệm ngăn xếp (Stack Underflow): Khi bộ nhớ ngăn xếp đã rỗng, nếu chương trình tiếp tục có lệnh lấy



Hình 4.4: Các chế độ reset của PIC16F877A

dữ liệu từ ngăn xếp ra thì hệ thống sẽ bị reset. Tình huống reset tràn ngăn xếp và tràn bộ đếm ngăn xếp cần được cho phép bởi bit số 12 của vùng nhớ cấu hình chip CONFIG2

- Chế độ nạp chương trình cho chip vi xử lý: Khi sử dụng bộ nạp trình cho Chip, vi xử lý được Reset để chuyển sang chế độ nạp trình.

4.2 Tổ chức bộ nhớ

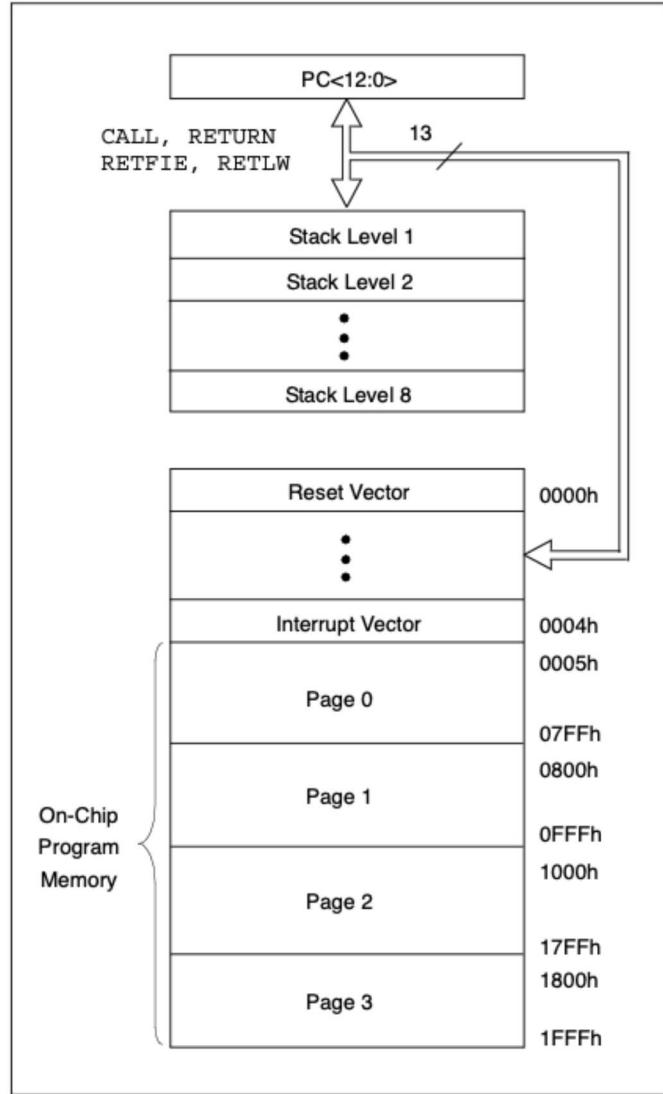
Với cấu trúc Harvard, vi xử lý sử dụng đường BUS dữ liệu và địa chỉ cho bộ nhớ chương trình và dữ liệu riêng biệt, nên cho phép truy nhập trực tiếp bộ nhớ chương trình từ vùng không gian dữ liệu khi chương trình đang chạy. Cấu trúc bộ nhớ của vi điều khiển PIC16F877A bao gồm bộ nhớ chương trình (Program memory) và bộ nhớ dữ liệu (Data memory)

4.2.1 Bộ nhớ chương trình

Bộ nhớ chương trình bao gồm vùng nhớ cấu hình chip (Configuration Words) và bộ nhớ chương trình dạng Flash. Bộ nhớ chương trình của vi điều khiển PIC16F877A có dung lượng 8K word, mỗi word có độ dài 14 bit tương ứng với độ dài của thanh ghi lệnh. Bộ nhớ chương trình được phân thành nhiều trang từ page0 - page3. Do mỗi lệnh được mã hoá trong 1 word nên chương trình nạp cho PIC16F877A tối đa có 8192 lệnh.

Để mã hoá được địa chỉ của 8K word bộ nhớ chương trình, PIC16F877A được thiết kế có bộ đếm chương trình có dung lượng 13 bit ($PC<12:0>$). Khi bị reset, bộ đếm chương trình sẽ chỉ tới địa chỉ 0000h (Reset vector). Khi có ngắt xảy ra, bộ đếm chương trình sẽ chỉ tới địa chỉ ngắt 0004h (Interrupt vector).

Bộ nhớ chương trình không bao gồm bộ nhớ stack. Bộ nhớ stack của PIC16F877A không được được địa chỉ hoá bởi bộ đếm chương trình. Bộ nhớ Stack của PIC16F877A có dung lượng chứa được 8 địa chỉ và hoạt động theo cơ chế xoay vòng. Khi có giá trị thứ 9 cần được lưu trữ, nó sẽ ghi đè lên giá trị đầu tiên. Lưu ý là không có một cờ hiệu nào được thiết kế để chỉ trạng thái của stack. Do đó ta không thể biết khi nào bộ nhớ này bị tràn. Thiết kế của 16F877A không cho phép người dùng tác động vào bộ nhớ stack. Mọi hoạt động của bộ nhớ stack đều do CPU điều khiển. Do vậy, tài liệu này không đề cập nhiều tới vùng nhớ này.



Hình 4.5: Vùng nhớ dữ liệu của PIC16F877A

4.2.2 Bộ nhớ dữ liệu

Bộ nhớ dữ liệu của PIC là bộ nhớ EEPROM được chia thành nhiều bank. Đối với PIC16F877A, bộ nhớ dữ liệu được chia thành 4 bank. Mỗi bank có dung lượng 128 byte, bao gồm các thanh ghi có chức năng đặc biệt SFG (Special Function Register) nằm ở vùng địa chỉ thấp và các thanh ghi mục đích chung GPR (General Purpose Register) nằm ở vùng địa chỉ còn lại trong bank. Các thanh ghi SFR thường xuyên được sử dụng sẽ được đặt ở tất cả các bank của bộ nhớ dữ liệu giúp thuận tiện trong quá trình truy xuất và làm giảm bớt lệnh của

chương trình. Sơ đồ cụ thể của bộ nhớ dữ liệu được thể hiện trên hình 4.6.

File Address	File Address	File Address	File Address
Indirect addr. ⁽¹⁾	00h	Indirect addr. ⁽¹⁾	100h
TMR0	01h	OPTION_REG	101h
PCL	02h	PCL	102h
STATUS	03h	STATUS	103h
FSR	04h	FSR	104h
PORTA	05h	TRISA	105h
PORTB	06h	TRISB	106h
PORTC	07h	TRISC	107h
PORTD ⁽¹⁾	08h	TRISD ⁽¹⁾	108h
PORTE ⁽¹⁾	09h	TRISE ⁽¹⁾	109h
PCLATH	0Ah	PCLATH	10Ah
INTCON	0Bh	INTCON	10Bh
PIR1	0Ch	PIE1	10Ch
PIR2	0Dh	PIE2	10Dh
TMR1L	0Eh	PCON	10Eh
TMR1H	0Fh		10Fh
T1CON	10h		110h
TMR2	11h	SSPCON2	111h
T2CON	12h	PR2	112h
SSPBUF	13h	SSPADD	113h
SSPCON	14h	SSPSTAT	114h
CCPR1L	15h		115h
CCPR1H	16h		116h
CCP1CON	17h		General Purpose Register
RCSTA	18h	TXSTA	117h
TXREG	19h	SPBRG	118h
RCREG	1Ah		16 Bytes
CCPR2L	1Bh		119h
CCPR2H	1Ch	CMCON	11Ah
CCP2CON	1Dh	CVRCON	11Bh
ADRESH	1Eh	ADRESL	11Ch
ADCN0	1Fh	ADCON1	11Dh
	20h		11Eh
General Purpose Register			11Fh
96 Bytes			120h
		General Purpose Register	
		80 Bytes	
		EFh	
		F0h	
		FFh	
	7Fh	accesses 70h-7Fh	
Bank 0	Bank 1	Bank 2	Bank 3
		General Purpose Register	
		80 Bytes	
		EFh	
		F0h	
		FFh	
		accesses 70h - 7Fh	
		16Fh	
		170h	
		17Fh	
		General Purpose Register	
		80 Bytes	
		EFh	
		F0h	
		FFh	
		accesses 70h - 7Fh	
		1FFh	

Hình 4.6: Các thanh ghi đa mục đích của PIC16F877A

Các thanh ghi chức năng đặc biệt SFR

Đây là các thanh ghi được sử dụng bởi CPU hoặc được dùng để thiết lập và điều khiển các khối chức năng được tích hợp bên trong vi điều khiển. Các thanh ghi SFR được phân làm 2 loại: thanh ghi SFR liên quan đến các chức năng bên trong CPU và thanh ghi SFR dùng để thiết lập và điều khiển các khối chức năng bên ngoài (ADC, PWM,...) Phần dưới đây sẽ liệt kê sơ lược các thanh ghi dùng

STATUS REGISTER (ADDRESS 03h, 83h, 103h, 183h)

R/W-0	R/W-0	R/W-0	R-1	R-1	R/W-x	R/W-x	R/W-x
IRP	RP1	RP0	TO	PD	Z	DC	C
bit 7							bit 0

OPTION_REG REGISTER (ADDRESS 81h, 181h)

R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1
RBPU	INTEDG	T0CS	T0SE	PSA	PS2	PS1	PS0
bit 7							bit 0

INTCON REGISTER (ADDRESS 0Bh, 8Bh, 10Bh, 18Bh)

R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-x
GIE	PEIE	TMR0IE	INTE	RBIE	TMR0IF	INTF	RBIF
bit 7							bit 0

PIE1 REGISTER (ADDRESS 8Ch)

R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
PSPIE ⁽¹⁾	ADIE	RCIE	TXIE	SSPIE	CCP1IE	TMR2IE	TMR1IE
bit 7							bit 0

PIR1 REGISTER (ADDRESS 0Ch)

R/W-0	R/W-0	R-0	R-0	R/W-0	R/W-0	R/W-0	R/W-0
PSPIF ⁽¹⁾	ADIF	RCIF	TXIF	SSPIF	CCP1IF	TMR2IF	TMR1IF
bit 7							bit 0

PIE2 REGISTER (ADDRESS 8Dh)

U-0	R/W-0	U-0	R/W-0	R/W-0	U-0	U-0	R/W-0
—	CMIE	—	EEIE	BCLIE	—	—	CCP2IE
bit 7							bit 0

PIR2 REGISTER (ADDRESS 0Dh)

U-0	R/W-0	U-0	R/W-0	R/W-0	U-0	U-0	R/W-0
—	CMIF	—	EEIF	BCLIF	—	—	CCP2IF
bit 7							bit 0

Hình 4.7: Các thanh ghi điều khiển chức năng bên trong.

để thiết lập và điều khiển các khối chức năng bên trong. Các thanh ghi điều khiển các khối chức năng bên ngoài sẽ được đề cập chi tiết trong các phần trình bày các khối chức năng tương ứng.

Thanh ghi STATUS (03h, 83h, 103h, 183h) là thanh ghi chứa kết quả thực hiện phép toán của ALU, trạng thái reset và các bit chọn bank cần truy xuất trong bộ nhớ dữ liệu.

Thanh ghi OPTION_REG (81h, 181h) cho phép đọc và ghi, cho phép điều khiển chức năng pull-up của các chân trong PORTB, xác lập các tham số về xung tác động, cạnh tác động của ngắt ngoại vi và bộ đếm Timer0.

Thanh ghi INCONT (0Bh, 8Bh, 10Bh, 18Bh) là thanh ghi cho phép đọc/ghi, chứa các bit điều khiển và các cờ hiệu khi ngắt Timer0 tràn, ngắt ngoại vi RB0/INT và ngắt Interrupt-on-change tại các chân của PORTB.

Thanh ghi PIE1 và PIR1: Thanh ghi PIE1 (8Ch) chứa các bit điều khiển chi tiết các ngắt của khối chức năng ngoại vi. Thanh ghi PIR1 (0Ch) chứa cờ ngắt của các khối chức năng tương ứng thanh ghi PIE1.

Thanh ghi PIE2 và PIR2: Tương tự như cặp thanh ghi PIE1 và PIR1, cặp thanh ghi PIE2 (8Dh) và PIR2 (0Dh) chứa các bit điều khiển và bit cờ ngắt của ngắt ngoại vi các khối chức năng CCP2, SSP bus, ngắt của bộ so sánh và ngắt ghi vào bộ nhớ EEPROM.

4.3 Các cổng ngoại vi - IO Port

Cổng xuất nhập (I/O port) chính là phương tiện mà vi điều khiển dùng để tương tác với thế giới bên ngoài. Sự tương tác này rất đa dạng và thông qua quá trình tương tác đó, chức năng của vi điều khiển được thể hiện một cách rõ ràng. Một cổng xuất nhập của vi điều khiển bao gồm nhiều chân (I/O pin). Số lượng chân I/O của mỗi cổng I/O có thể khác nhau tùy thuộc vào cách bố trí và chức năng của từng cổng I/O của vi điều khiển

PART A

PORTA (RPA) bao gồm 6 I/O chân hay còn được gọi là pin. Mỗi chân này có thể được định nghĩa là các chân vào hoặc ra để phục vụ việc xuất/nhập dữ liệu. Việc định nghĩa là chân vào (input) hay chân ra (output) được điều khiển bởi thanh ghi TRISA tại địa chỉ 85h. Muốn xác định chức năng vào tại một chân trong PORTA là input, ta “set” bit điều khiển tương ứng với chân này trong thanh ghi TRISA. Ngược lại, muốn xác lập chân đó là output, ta “clear” bit tương ứng

trong thanh ghi TRISA. Thao tác này được sử dụng hoàn toàn tương tự đối với các cổng port khác (đối với PORTA là TRISA, đối với PORTB là TRISB, và đối với PORTC là TRISC,...). Ngoài ra, cổng PORTA còn là ngõ vào của bộ chuyển đổi ADC hoặc điện áp tham chiếu $V_{ref+/-}$, đầu vào/ra của bộ so sánh, đầu vào xung clock cho Timer0 hay các chân đồng bộ cho cổng truyền tín hiệu nối tiếp. Các chức năng cụ thể liên quan tới cổng PORTA sẽ được trình bày tại những phần sau.

Các thanh ghi chức năng SFR liên quan tới PORTA bao gồm:

- PORTA (05h): chứa các giá trị các chan trong PORTA
- TRISA (85h): điều khiển xuất/nhập
- CMCON (9Ch): thanh ghi điều khiển bộ so sánh
- CVRCON (9Dh): thanh ghi điều khiển bộ so sánh điện áp
- ADCON1 (9Fh): thanh ghi điều khiển bộ ADC

TABLE 4-1: PORTA FUNCTIONS

Name	Bit#	Buffer	Function
RA0/AN0	bit 0	TTL	Input/output or analog input.
RA1/AN1	bit 1	TTL	Input/output or analog input.
RA2/AN2/VREF-/CVREF	bit 2	TTL	Input/output or analog input or VREF- or CVREF.
RA3/AN3/VREF+	bit 3	TTL	Input/output or analog input or VREF+.
RA4/T0CKI/C1OUT	bit 4	ST	Input/output or external clock input for Timer0 or comparator output. Output is open-drain type.
RA5/AN4/SS/C2OUT	bit 5	TTL	Input/output or analog input or slave select input for synchronous serial port or comparator output.

Legend: TTL = TTL input, ST = Schmitt Trigger input

TABLE 4-2: SUMMARY OF REGISTERS ASSOCIATED WITH PORTA

Address	Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Value on: POR, BOR	Value on all other Resets
05h	PORTA	—	—	RA5	RA4	RA3	RA2	RA1	RA0	--0x 0000	--0u 0000
85h	TRISA	—	—	PORTA Data Direction Register						--11 1111	--11 1111
9Ch	CMCON	C2OUT	C1OUT	C2INV	C1INV	CIS	CM2	CM1	CM0	0000 0111	0000 0111
9Dh	CVRCON	CVREN	CVROE	CVRR	—	CVR3	CVR2	CVR1	CVR0	000- 0000	000- 0000
9Fh	ADCON1	ADFM	ADCS2	—	—	PCFG3	PCFG2	PCFG1	PCFG0	00-- 0000	00-- 0000

Legend: x = unknown, u = unchanged, - = unimplemented locations read as '0'. Shaded cells are not used by PORTA.

PART B

PARTB (RPB) gồm 8 chân vào ra I/O và thanh ghi điều khiển tương ứng là TRISB. Ngoài chức năng làm chân xuất nhập dữ liệu, một số chân của PARTB còn được sử dụng trong quá trình nạp cho vi điều khiển với các chế độ nạp khác nhau. Bên cạnh đó, PARTB còn liên quan tới ngắn ngoại vi và bộ nhớ Timer0.

PORTB còn được tích hợp chức năng điện trở kéo lên được điều khiển bởi chương trình.

Các thanh ghi SFR liên quan tới PORTB bao gồm:

- PORTB (05h, 106h): chứa các giá trị các chân trong PORTB
- TRISB (86h, 186h): điều khiển xuất/nhập
- OPTION_REG (81, 181h): điều khiển ngắt ngoại vi và bộ Timer0

TABLE 4-3: PORTB FUNCTIONS

Name	Bit#	Buffer	Function
RB0/INT	bit 0	TTL/ST ⁽¹⁾	Input/output pin or external interrupt input. Internal software programmable weak pull-up.
RB1	bit 1	TTL	Input/output pin. Internal software programmable weak pull-up.
RB2	bit 2	TTL	Input/output pin. Internal software programmable weak pull-up.
RB3/PGM ⁽³⁾	bit 3	TTL	Input/output pin or programming pin in LVP mode. Internal software programmable weak pull-up.
RB4	bit 4	TTL	Input/output pin (with interrupt-on-change). Internal software programmable weak pull-up.
RB5	bit 5	TTL	Input/output pin (with interrupt-on-change). Internal software programmable weak pull-up.
RB6/PGC	bit 6	TTL/ST ⁽²⁾	Input/output pin (with interrupt-on-change) or in-circuit debugger pin. Internal software programmable weak pull-up. Serial programming clock.
RB7/PGD	bit 7	TTL/ST ⁽²⁾	Input/output pin (with interrupt-on-change) or in-circuit debugger pin. Internal software programmable weak pull-up. Serial programming data.

Legend: TTL = TTL input, ST = Schmitt Trigger input

TABLE 4-4: SUMMARY OF REGISTERS ASSOCIATED WITH PORTB

Address	Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Value on: POR, BOR	Value on all other Resets
06h, 106h	PORTB	RB7	RB6	RB5	RB4	RB3	RB2	RB1	RB0	xxxx xxxx	uuuu uuuu
86h, 186h	TRISB	PORTB Data Direction Register								1111 1111	1111 1111
81h, 181h	OPTION_REG	RBU	INTEDG	T0CS	T0SE	PSA	PS2	PS1	PS0	1111 1111	1111 1111

Legend: x = unknown, u = unchanged. Shaded cells are not used by PORTB.

POR C

PORTC (RPC) gồm 8 chân I/O và thanh ghi điều khiển xuất nhập tương ứng là TRISC.Ngoài chức năng là các chân của các cổng vào/ra, PORTC còn chứa các chân chức năng của bộ so sánh, bộ Timer1, bộ điều chế độ rộng xung PWM và các chân điều khiển giao tiếp nối tiếp I2C, SPI, SSP, USART.

Các thanh ghi SFR liên quan tới PORTC bao gồm:

- PORTC (07h): chứa các giá trị các chân trong PORTC
- TRISC (87h): điều khiển xuất/nhập

TABLE 4-5: PORTC FUNCTIONS

Name	Bit#	Buffer Type	Function
RC0/T1OSO/T1CKI	bit 0	ST	Input/output port pin or Timer1 oscillator output/Timer1 clock input.
RC1/T1OSI/CCP2	bit 1	ST	Input/output port pin or Timer1 oscillator input or Capture2 input/ Compare2 output/PWM2 output.
RC2/CCP1	bit 2	ST	Input/output port pin or Capture1 input/Compare1 output/ PWM1 output.
RC3/SCK/SCL	bit 3	ST	RC3 can also be the synchronous serial clock for both SPI and I ² C modes.
RC4/SDI/SDA	bit 4	ST	RC4 can also be the SPI data in (SPI mode) or data I/O (I ² C mode).
RC5/SDO	bit 5	ST	Input/output port pin or Synchronous Serial Port data output.
RC6/TX/CK	bit 6	ST	Input/output port pin or USART asynchronous transmit or synchronous clock.
RC7/RX/DT	bit 7	ST	Input/output port pin or USART asynchronous receive or synchronous data.

Legend: ST = Schmitt Trigger input

TABLE 4-6: SUMMARY OF REGISTERS ASSOCIATED WITH PORTC

Address	Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Value on: POR, BOR	Value on all other Resets
07h	PORTC	RC7	RC6	RC5	RC4	RC3	RC2	RC1	RC0	xxxx xxxx	uuuu uuuu
87h	TRISC	PORTC Data Direction Register								1111 1111	1111 1111

Legend: x = unknown, u = unchanged

TABLE 4-7: PORTD FUNCTIONS

Name	Bit#	Buffer Type	Function
RD0/PSP0	bit 0	ST/TTL ⁽¹⁾	Input/output port pin or Parallel Slave Port bit 0.
RD1/PSP1	bit 1	ST/TTL ⁽¹⁾	Input/output port pin or Parallel Slave Port bit 1.
RD2/PSP2	bit2	ST/TTL ⁽¹⁾	Input/output port pin or Parallel Slave Port bit 2.
RD3/PSP3	bit 3	ST/TTL ⁽¹⁾	Input/output port pin or Parallel Slave Port bit 3.
RD4/PSP4	bit 4	ST/TTL ⁽¹⁾	Input/output port pin or Parallel Slave Port bit 4.
RD5/PSP5	bit 5	ST/TTL ⁽¹⁾	Input/output port pin or Parallel Slave Port bit 5.
RD6/PSP6	bit 6	ST/TTL ⁽¹⁾	Input/output port pin or Parallel Slave Port bit 6.
RD7/PSP7	bit 7	ST/TTL ⁽¹⁾	Input/output port pin or Parallel Slave Port bit 7.

Legend: ST = Schmitt Trigger input, TTL = TTL input

Note 1: Input buffers are Schmitt Triggers when in I/O mode and TTL buffers when in Parallel Slave Port mode.

TABLE 4-8: SUMMARY OF REGISTERS ASSOCIATED WITH PORTD

Address	Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Value on: POR, BOR	Value on all other Resets
08h	PORTD	RD7	RD6	RD5	RD4	RD3	RD2	RD1	RD0	xxxx xxxx	uuuu uuuu
88h	TRISD	PORTD Data Direction Register								1111 1111	1111 1111
89h	TRISE	IBF	OBF	IBOV	PSPMODE	—	PORTE Data Direction Bits			0000 -111	0000 -111

Legend: x = unknown, u = unchanged, - = unimplemented, read as '0'. Shaded cells are not used by PORTD.

PORT D

PORTD (RPD) gồm 8 chân I/O và thanh ghi điều khiển xuất nhập tương ứng là TRISD. PORTD còn là cổng xuất dữ liệu của chuẩn giao tiếp song song PSP (Parallel Slave Port).

Các thanh ghi SFR liên quan tới PORTD bao gồm:

- PORTD (08h): chứa các giá trị các chân trong PORTD
- TRISD (88h): điều khiển xuất/nhập
- TRISE (89h): điều khiển xuất/nhập PORTE và chuẩn giao tiếp PSP

PORT E

TABLE 4-9: PORTE FUNCTIONS

Name	Bit#	Buffer Type	Function
RE0/ \overline{RD} /AN5	bit 0	ST/TTL ⁽¹⁾	I/O port pin or read control input in Parallel Slave Port mode or analog input: \overline{RD} 1 = Idle 0 = Read operation. Contents of PORTD register are output to PORTD I/O pins (if chip selected).
RE1/ \overline{WR} /AN6	bit 1	ST/TTL ⁽¹⁾	I/O port pin or write control input in Parallel Slave Port mode or analog input: \overline{WR} 1 = Idle 0 = Write operation. Value of PORTD I/O pins is latched into PORTD register (if chip selected).
RE2/ \overline{CS} /AN7	bit 2	ST/TTL ⁽¹⁾	I/O port pin or chip select control input in Parallel Slave Port mode or analog input: \overline{CS} 1 = Device is not selected 0 = Device is selected

Legend: ST = Schmitt Trigger input, TTL = TTL input

Note 1: Input buffers are Schmitt Triggers when in I/O mode and TTL buffers when in Parallel Slave Port mode.

TABLE 4-10: SUMMARY OF REGISTERS ASSOCIATED WITH PORTE

Address	Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Value on: POR, BOR	Value on all other Resets
09h	PORTE	—	—	—	—	—	RE2	RE1	RE0	---- -xxx	---- -uuu
89h	TRISE	IBF	OBF	IBOV	PSPMODE	—	PORTE Data Direction bits			0000 -111	0000 -111
9Fh	ADCON1	ADFM	ADCS2	—	—	PCFG3	PCFG2	PCFG1	PCFG0	00-- 0000	00-- 0000

Legend: x = unknown, u = unchanged, - = unimplemented, read as '0'. Shaded cells are not used by PORTE.

PORTE (PRE) chỉ gồm 3 chân I/O và thanh ghi điều khiển xuất nhập tương ứng là TRISE. Các chân của PORTE có ngõ vào analog. Bên cạnh đó, PORTE còn là các chân điều khiển của chuẩn giao tiếp PSP.

Các thanh ghi SFR liên quan tới PORTE bao gồm:

- PORTD (09h): chứa các giá trị các chân trong PORTE
- TRISD (89h): điều khiển xuất/nhập và xác lập các thông số cho chuẩn giao tiếp PSP.

-
- ADCON1 (9Fh): thanh ghi điều khiển khối ADC

Mục đích sử dụng các PORT

Cổng vào số - Digital Input: Như phần giới thiệu chi tiết ở trên, mỗi chân của các cổng từ PORTA đến PORTE đều có thể được dùng làm cổng vào số. Các chân tín hiệu này được sử dụng để nhận tín hiệu có mức Logic từ các thiết bị như phím bấm, công tắc hoặc tín hiệu mô tả trạng thái hoạt động của thiết bị ghép nối. Thông thường, bộ vi điều khiển sẽ được lập trình để tạo ra một chu kỳ để đọc các tín hiệu này. Chu kỳ này phụ thuộc vào nhu cầu về tần xuất cập nhật thông tin từ bên ngoài và tốc độ thực hiện lệnh của vi điều khiển.

Khi có nhu cầu đọc dữ liệu nhanh gần như tức thời, kỹ thuật ngắn được sử dụng. Ngắn được dùng trong kiểu nhập dữ liệu này thường là ngắn ngoài INTE tại cổng RB0/INT hoặc ngắn RBIE.

Cổng ra số - Digital Output: Cũng sử dụng các chân của vi điều khiển như cổng vào số, các chân này có thể được định dạng thành cổng ra số. Khi đó, tại các chân ở các cổng tương ứng, tín hiệu logic được đưa ra để tác động vào các thiết bị ngoại vi như các led báo (công suất thấp), các relay điều khiển (công suất cao) hoặc truyền tín hiệu điều khiển sang các thiết bị khác. Tương tự như cổng vào số, phần mềm của vi điều khiển sẽ quản lý trực tiếp tín hiệu ra và có tần số quét phụ thuộc vào tốc độ của vi xử lý.

Cổng vào tương tự - Analog Input: Trong trường hợp tín hiệu vào là tín hiệu dạng tương tự, PIC16F877A cung cấp 8 cổng vào. Các cổng này được kết nối tới khối chức năng chuyển đổi tương tự - số ADC. Trong một vài trường hợp, các tín hiệu tương tự có thể được kết nối với khối chức năng so sánh (Comparation) được tích hợp bên trong Chip.

Cổng ra tương tự - Analog Output: Khi tín hiệu đưa ra cần mô tả một tín hiệu tương tự, người dùng có thể sử dụng chức năng điều chế độ rộng xung PWM và kết nối thêm phần mạch khuếch đại công suất.

Cổng vào/ra truyền thông: Các cổng này được kết nối với khối phần cứng truyền thông bên trong như SPI, USART, I2C,...

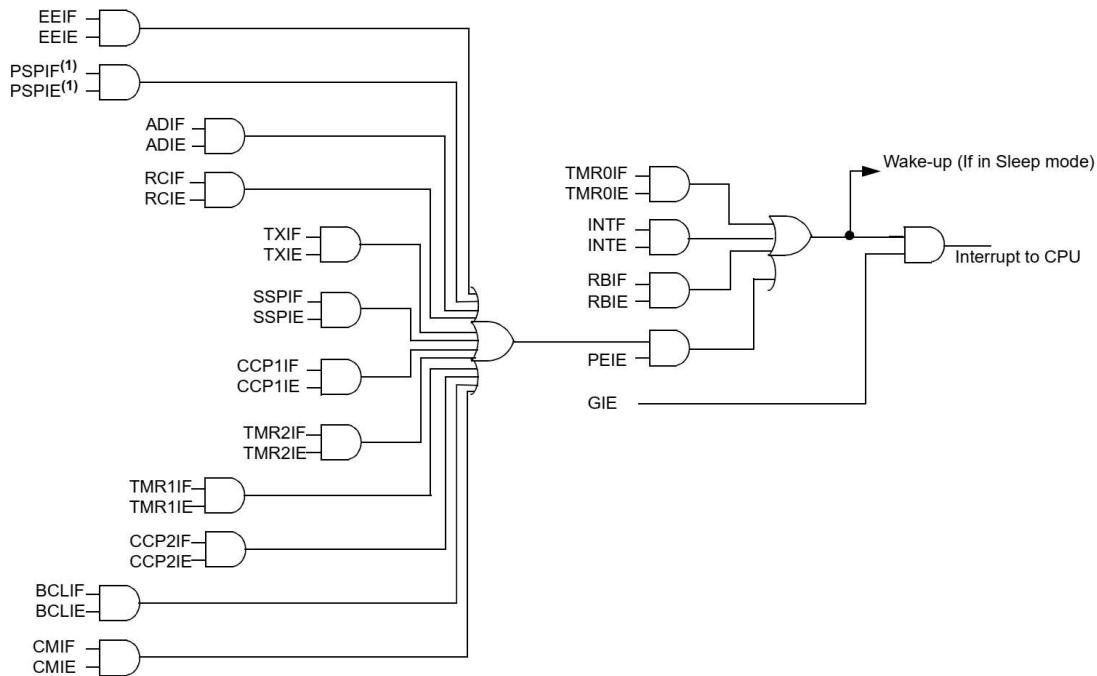
4.4 Ngắn và điều khiển ngắn

Ngắn là một phản ứng của bộ vi xử lý đối với một sự kiện cần được phần mềm chú ý. Khi điều kiện ngắn diễn ra và nếu được chấp nhận, bộ vi xử lý sẽ phản ứng bằng cách tạm dừng các hoạt động hiện tại của nó, lưu trạng thái của nó và

thực hiện chương trình ngắn (trình xử lý ngắn) tương ứng với điều kiện ngắn. Sự gián đoạn này là tạm thời và bộ vi xử lý sẽ tiếp tục các hoạt động bình thường sau khi trình xử lý ngắn kết thúc. Các sự kiện ngắn (hay còn được gọi là các tín hiệu ngắn) có thể được phát ra để đáp ứng với các sự kiện phần cứng - ngắn cứng hoặc phần mềm - ngắn mềm. Với mỗi loại vi xử lý, số lượng các ngắn cứng và ngắn mềm bị giới hạn bởi kiến trúc của máy.

PIC16F877A được thiết kế có 15 nguồn tạo ra hoạt động ngắn và được điều khiển bởi thanh ghi INTCON. Trong thanh ghi này, bit GIE - Global Interrupt Enable sẽ cho phép các ngắn hoạt động. Bên cạnh đó, mỗi ngắn còn có một bit điều khiển và cờ ngắn riêng. Khi điều kiện ngắn xảy ra, các cờ ngắn vẫn được xác lập bình thường bất chấp trạng thái của GIE. Tuy nhiên, hoạt động ngắn chỉ được diễn ra phụ thuộc vào bit GIE và các bit điều khiển tương ứng. Ngoài bit điều khiển GIE, thanh ghi INTCON còn chứa các bit cho phép ngắn ngoại vi PEIE và bit điều khiển và bit cờ của ngắn RB0/INT và TMR0. Với các ngắn ngoại vi, các bit điều khiển ngắn nằm trong thanh ghi PIE1 và PIE2 còn các bit cờ nằm trong thanh ghi PIR1 và PIR2.

Trong thời điểm xử lý đáp ứng ngắn, chỉ duy nhất một chương trình ngắn được thực thi. Khi đó, bit GIE tự động được reset về 0, đĩa chỉ lệnh kế tiếp của chương



Hình 4.8: Sơ đồ logic tất cả các ngắn trong PIC16F877A

trình chính được cất vào bộ nhớ Stack và bộ đếm chương trình sẽ chỉ đến địa chỉ 0004h. Đây là địa chỉ vector ngắn, nơi chứa chương trình ngắn. Lệnh RETFIE được dùng để thoát khỏi chương trình ngắn và quay trở về chương trình chính, đồng thời bit GIE cũng sẽ được set để cho phép các ngắn hoạt động trở lại. Các bit cờ của các ngắn cần được xóa bằng chương trình trước khi cho phép ngắn tiếp tục hoạt động trở lại.

Lưu ý rằng trong quá trình thực thi chương trình đáp ứng ngắn, chỉ có giá trị của thanh ghi bộ đếm chương trình được cất trong Stack. Một số thanh ghi quan trọng khác sẽ không được cất và có thể bị thay đổi trong quá trình thực thi chương trình đáp ứng ngắn. Do vậy, người sử dụng cần xử lý vấn đề này bằng chương trình.

Đối với các ngắn ngoại vi như ngắn từ chân INT hay ngắn phát hiện sự thay đổi trạng thái của các chân tín hiệu cổng PORTB, việc xác định ngắn nào xảy ra cần 3 hoặc 4 chu kỳ lệnh tùy thuộc vào thời điểm xảy ra ngắn.

INTCON REGISTER (ADDRESS 0Bh, 8Bh, 10Bh, 18Bh)

R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-x
GIE	PEIE	TMR0IE	INTE	RBIE	TMR0IF	INTF	RBIF
bit 7							bit 0

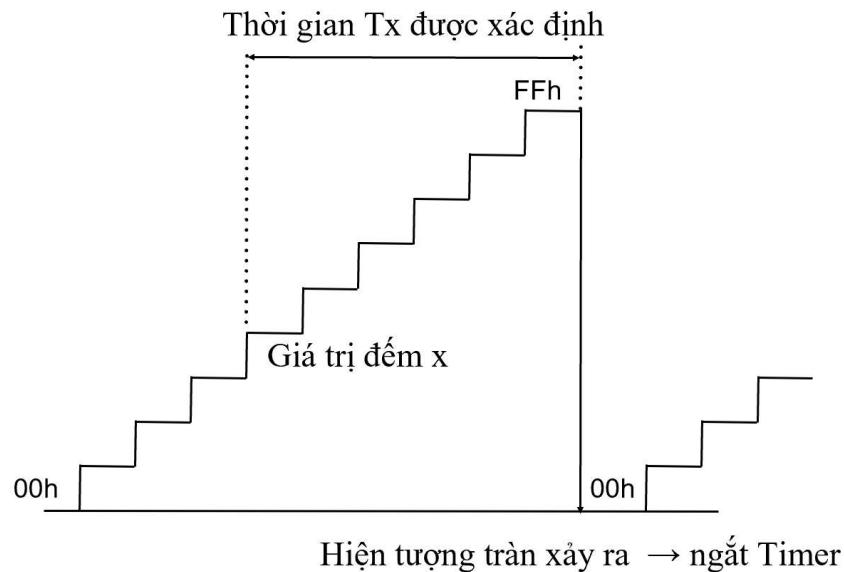
Thanh ghi INTCON quản lý 3 ngắn gồm ngắn Timer0 (TRM0IE), ngắn từ bên ngoài INT (RB0/INT) và ngắn phát hiện sự thay đổi trạng thái các chân cổng PORTB (RBIE). Trong ngắn Timer0, khi thanh ghi TRM0 xảy ra hiện tượng tràn số từ $FFh \rightarrow 00h$, bit cờ tương ứng của ngắn Timer0 - TRM0IF sẽ được set lên 1. Khi đó chương trình ngắn tương ứng với ngắn Timer0 sẽ được gọi tới nếu như bit cho phép ngắn Timer 0 (bit TMR0IE) được set (Chi tiết về ngắn Timer0 sẽ được trình bày chi tiết trong mục 4.5 tiếp theo phần này). Ngắn INT là ngắn có tín hiệu ngắn dựa trên sự thay đổi trạng thái của chân RB0/INT. Sự kiện ngắn có thể được phát hiện dựa trên sự thay đổi tín hiệu tại sườn lên/sườn xuống của tín hiệu đi vào chân RB0/INT (chân số 33 của PIC16F877A định dạng 2 hàng chân) tùy thuộc vào giá trị của bit số 6 trong thanh ghi OPTION_REG (bit INTEDG). Ngắn RBIE là ngắn được dùng để phát hiện sự thay đổi của các chân tín hiệu PORTB<7:4>, được điều khiển bởi bit RBIE. Cờ ngắn tương ứng là bit RBIF.

4.5 Bộ định thời Timer

Timer (bộ định thời) là một chức năng được tích hợp trên bộ vi điều khiển. Chức năng chính của bộ định thời là xác định ra một khoảng thời gian nhất định tùy thuộc vào ứng dụng của người sử dụng hoặc sử dụng khối Timer để tạo ra một sự kiện ngắn. Khối chức năng Timer bao gồm 1 thanh ghi chứa giá trị đếm Timer và một thanh ghi điều khiển hoạt động. Giá trị đếm trong thanh ghi Timer thường là đếm tăng từ giá trị nhỏ nhất là 00h hoặc từ một giá trị được cài đặt đến giá trị lớn nhất của nó (với bộ đếm 8 bit thì giá trị lớn nhất là FFh). Khi bộ đếm chuyển trạng thái từ FFh về 00h, bộ đếm được gọi là tràn đồng thời, một bit cờ tương ứng với Timer đó sẽ được set lên 1 với mục đích báo hiệu. Bit cờ này được sử dụng như là một tín hiệu sự kiện ngắn Timer. Sau khi tràn, bộ đếm lại tiếp tục đếm từ giá trị nhỏ nhất là 00h. Giá trị đếm trong thanh ghi Timer tăng theo một chu kỳ thời gian (chu kỳ máy) phụ thuộc vào dao động thạch anh cấp cho vi điều khiển. Để định ra một khoảng thời gian T_x , một giá trị đếm thích hợp x cần được nạp vào trong thanh ghi đếm. Khi đó

$$T_x = (FFh - x) * T_{\text{chu kỳ máy}}$$

Bit cờ ngắn của Timer cũng như các bit được dùng để xác định các chế độ hoạt động của Timer thường nằm trong thanh ghi điều khiển. Ngoài chế độ hoạt động



Hình 4.9: Nguyên lý hoạt động của ngắn trong vi điều khiển

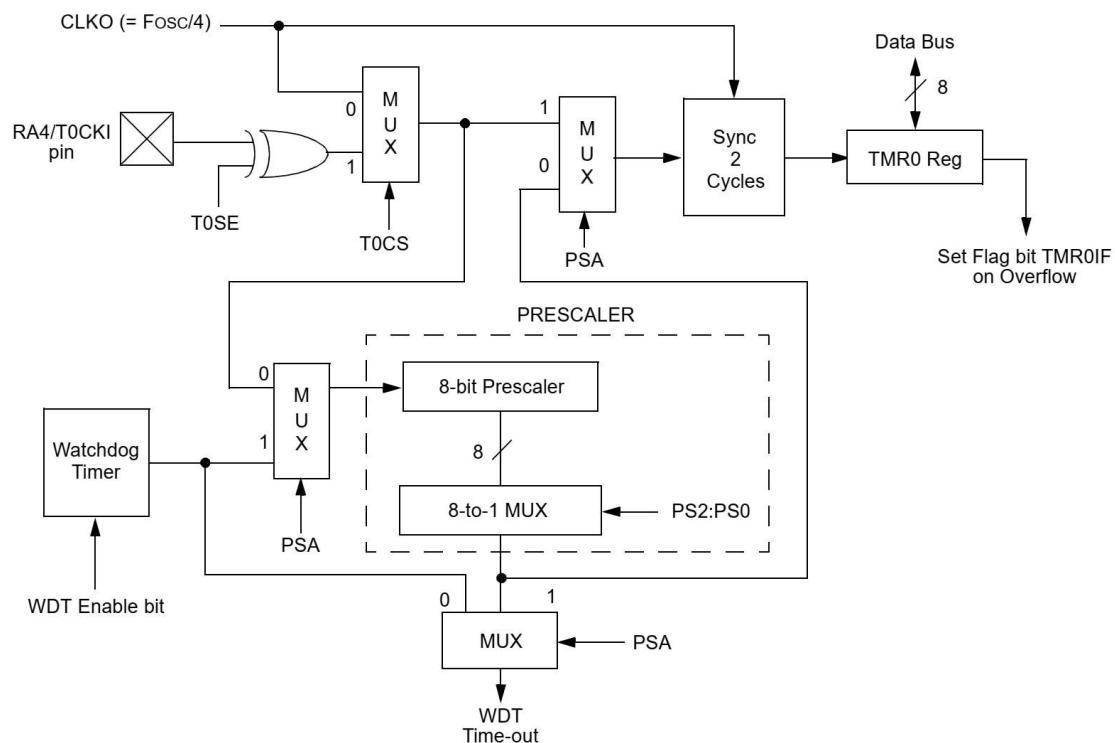
là bộ định thời Timer, khôi phục năng Timer còn có thể hoạt động như làm một bộ đếm sự kiện Counter. Trong trường hợp này, giá trị của thanh ghi Timer sẽ tăng tương ứng với một sự kiện (sườn lê hoặc sườn xuống) tại một chân đầu vào.

TIMER0

Timer0 là một trong 3 bộ định thời của vi điều khiển PIC16F877A. Đây là bộ đếm 8 bit được kết nối với một bộ chia tần (Prescaler) 8 bit. Cấu trúc của bộ Timer cho phép nó hoạt động ở chế độ định thời Timer với chu kỳ máy bằng 4 lần chu kỳ thạch anh dao động hoặc hoạt động trong chế độ bộ đếm Counter với xung đếm được đưa vào chân RA4/T0CKI. Ngắt Timer0 sẽ xuất hiện khi thanh ghi đếm Timer0 (TMR0) bị tràn. Để sử dụng Timer0, bit TMR0IE (INTCON<5>) cần được set lên 1 để cho phép ngắt Timer0 tác động. Khi TMR0IE = 0, ngắt Timer0 không hoạt động.

Chế độ định thời Timer

Để sử dụng Timer0 ở chế độ định thời gian, bit T0SC (OPTION_REG<5>) cần bị xóa về 0. Khi đó, giá trị thanh ghi TMR0 sẽ tăng theo từng chu kỳ máy.



Hình 4.10: Cấu trúc của Timer0

OPTION_REG REGISTER (ADDRESS 81h, 181h)

Một chu kỳ máy trong PIC16F877A bằng 4 lần chu kỳ xung nhịp. Khi giá trị thanh ghi TMR0 tràn, ngắt Timer0 sẽ xuất hiện. Thanh ghi TMR0 cho phép ta ghi/xóa để ấn định thời điểm ngắt Timer0 xuất hiện một cách linh động.

Ché độ bộ đếm Counter

Trong chế độ bộ đếm Counter, bit T0SC được set lên 1. Khi đó, xung tác động lên bộ đếm được lấy từ chân RA4/T0CKI. Bit T0SE (OPTION_REG<4>) cho phép lựa chọn sườn xung tác động vào bộ đếm. Khi T0SE = 0, bộ đếm sẽ tăng khi gấp sườn lên của xung đến chân RA4. Khi T0SE = 1, bộ đếm sẽ tăng tương ứng với sườn xuống của xung.

Khi Timer0 tràn

Khi thanh ghi TMR0 tràn, bit TMR0IF (INTCON<2>) được set. Đây chính là cờ ngắt của Timer0. Cờ ngắt này phải được xóa bằng chương trình trước khi bộ đếm bắt đầu thực hiện lại quá trình đếm. Lưu ý, ngắt Timer0 không thể "đánh thức" vi điều khiển từ chế độ sleep.

Sử dụng bộ chia tần Prescaler

Thực tế khi sử dụng Timer0, trong trường hợp dao động thạch anh là 20MHz, thời gian T_x lớn nhất mà Timer0 định ra được là $51.2\mu s$. Để có thể định ra được khoảng thời gian lớn hơn, một bộ chia tần được sử dụng. Giả sử với tỉ lệ chia tần là 1:2, cứ mỗi 2 xung nhịp clock, giá trị của TMR0 mới tăng lên 1 đơn vị. Do đó, khi dùng Timer0 ở bộ chia tần 1:256, thời gian tối đa của T_x có thể đạt được là hơn 13 ms. Để sử dụng bộ chia tần Prescaler, các bit điều khiển trong thanh ghi OPTION REG cần được định nghĩa tương ứng.

- **PSA** - Prescaler Assignment bit: là bit lựa chọn có/không sử dụng bộ Prescaler. Khi PSA = 0, bộ chia tần sẽ sử dụng tần số thạch anh, chia theo tỉ lệ và đưa vào khối thanh ghi TMR0. Khi PSA = 1, bộ chia tần sử dụng tần số tạo ra từ bộ WDT (Watchdog Timer). WDT là một khối chức năng định thời được sử dụng để thực hiện một nhiệm vụ tuần tự. Qua đó, WDT có thể phát hiện ra sự bất thường trong hoạt động của bộ vi điều khiển. Đối với PIC16F877A, WDT là một bộ dao động RC hoạt động độc lập với mạch dao động và tín hiệu dao động từ chân OSC1/CLK1. Thậm chí, WDT vẫn hoạt động ngay cả trong chế độ sleep. Chu kỳ T_{WDT} trải dài từ 7 đến 33 ms với điều kiện điện áp nguồn

PS2:PS0	Tỉ lệ chia tần Timer0	Tỉ lệ chia WDT
000	1:2	1:1
001	1:4	1:2
010	1:8	1:4
011	1:16	1:8
100	1:32	1:16
101	1:64	1:32
110	1:128	1:64
111	1:256	1:128

cung cấp V_{DD} là 5V và nhiệt độ môi trường từ $-40^{\circ}C$ đến $+85^{\circ}C$. Ở nhiệt độ phòng $+25^{\circ}C$, $T_{WDT} = 18$ ms nhưng cũng có tài liệu sử dụng giá trị 16 ms. Lưu ý các con số trên chưa được nhà sản xuất kiểm chứng, do đó cần được đo kiểm lại khi chạy mạch thực tế.

- **PS2:PS0** - Prescaler Rate Select bit: là bit được sử dụng để lựa chọn tỉ lệ chia tần tương ứng

Sử dụng Timer0

1. Tính toán giá trị x nạp vào TMR0 cần thiết để tạo ra trễ $T_{trễ}$ mong muốn

$$x = 256 - \frac{T_{trễ} * f_{osc}}{4 * Prescalar}$$

2. Thiết lập các bit PS2:PS0 trong thanh ghi OPTION_REG tương ứng với hệ số Prescaler mong muốn

3. Xóa bit PSA để xác nhận sử dụng tần số dao động thạch anh trong việc định thời.

4. Xóa bit T0SC để sử dụng Timer0 ở chế độ Timer.

5. Nạp giá trị x vào TMR0.

6. Kích hoạt Timer0 hoạt động bằng cách set bit TMR0IE.

7. Kích hoạt ngắt Timer bằng cách set các bit GIE và PEIE (Nếu dùng ngắt)

TIMER1

Timer1 là bộ định thời 16 bit. Giá trị đếm của Timer1 được lưu trong 2 thanh ghi 8 bit là TMR1H và TMR2L. Bit điều khiển Timer1 là bit TMR1IE (bit số 0 của thanh ghi PIE1) và cờ ngắt tương ứng là TMR1IF (bit số 0 của thanh ghi PIR1). Tương tự như Timer0, Timer1 cũng có 2 chế độ hoạt động: chế độ định thời Timer và chế độ đếm. Hoạt động của Timer1 được điều khiển bởi thanh ghi

T1CON: TIMER1 CONTROL REGISTER (ADDRESS 10h)

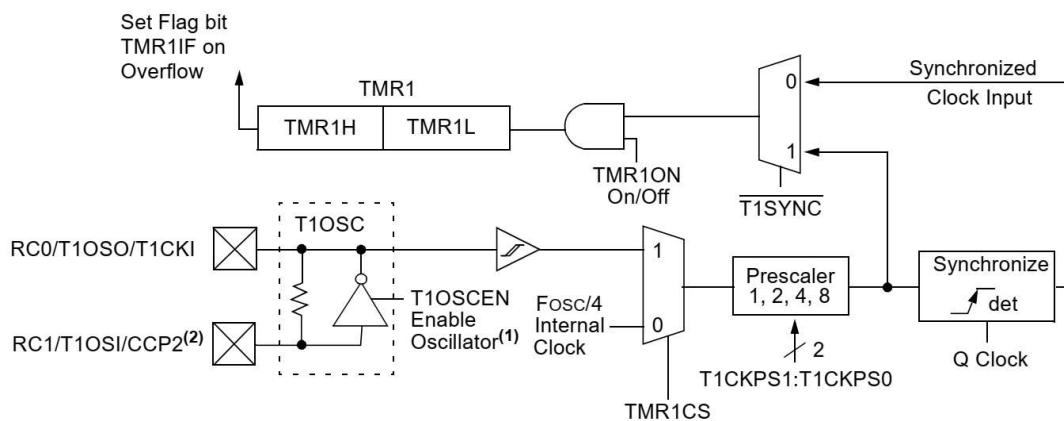
T1CON.

Chế độ định thời Timer

Để sử dụng Timer1 ở chế độ định thời timer, bit điều khiển TMR1IE cần được set lên 1 và bit lựa chọn TMR1CS bị xóa về 0. Khi đó, giá trị 16 bit trong 2 thanh ghi $\langle TMR1H:TMR1L \rangle$ sẽ tăng lên theo mỗi chu kỳ máy (tương đương 4 chu kỳ xung nhịp). Quá trình đếm có thể được tạm dừng/tiếp tục tăng tùy thuộc vào bit điều khiển TMR1ON. Khi TMR1ON = 0, bộ định thời dừng đếm, và giá trị $\langle TMR1H:TMR1L \rangle$ sẽ tiếp tục tăng khi TMR1ON = 1. Khi giá trị đếm đạt trạng thái tràn, bit cờ TMR1IF sẽ được set lên 1 và chương trình ngắt Timer1 sẽ được gọi tới.

Ché đô đếm Counter

Trong chế độ bộ đếm, giá trị trong $\langle TMR1H:TMR1L \rangle$ sẽ tăng khi có tín hiệu đếm đồng thời bit TMR1CS = 1 và TMR1ON = 1. Khi Timer1 hoạt động trong chế độ đếm xung từ bên ngoài, giá trị của thanh ghi đếm tăng theo sườn lên của tín hiệu đếm. Tuy nhiên trước đó, vì điều khiển phải nhận được tín hiệu sườn xuống tại ngõ vào của tín hiệu đếm. Tín hiệu đếm được lấy từ chân RC1/T1OSO/CCP2 khi bit T1OSCEN = 1, hoặc được lấy từ chân RC0/T1OSO/T1CKI khi bit T1OSCEN = 0



Hình 4.11: Cấu trúc của Timer1

Ở chế độ đếm đồng bộ ($\overline{T1SYNC} = 0$), tín hiệu đếm được đồng bộ với một tín hiệu xung nhịp ở bên trong vi điều khiển. Quá trình đồng bộ được diễn ra sau bộ chia tần Prescaler. Trong chế độ Sleep, bộ đếm Timer1 không hoạt động.

Ở chế độ đếm không đồng bộ ($\overline{T1SYNC} = 1$), tín hiệu đếm không được đồng bộ với xung clock bên trong. Trong trường hợp này, Timer1 vẫn hoạt động ngay cả trong chế độ Sleep và ngắt do Timer1 tạo ra có khả năng đánh thức vi điều khiển. Ở chế độ không đồng bộ, Timer1 không thể được sử dụng để làm nguồn xung clock cho khối CCP. Do vậy, tín hiệu đếm chỉ được lấy từ chân RC0/T1OSO/T1CKI của vi điều khiển.

Bộ chia tần trong Timer1

Cũng giống như đối với Timer0, Timer1 cũng có bộ chia tần được điều khiển bởi 2 bit T1CKPS1:T1CKPS0 trong thanh ghi T1CON

T1CKPS1:T1CKPS0	Tỉ lệ chia tần Timer1
00	1:1
01	1:2
10	1:4
11	1:8

Bộ dao động Timer1

PIC16F877A có thể kết hợp một mạch thạch anh bên ngoài giữa chân T1OSI (input) và T1OSO (output) để tạo tích hợp một bộ dao động năng lượng thấp, và được điều khiển bởi bit T1OSCEN (T1CON<3>). Bộ dao động hoạt động với tần số lớn nhất là 200 KHz và có thể hoạt động ngay cả khi vi điều khiển đang trong chế độ Sleep. Tần số hoạt động thông thường là 32 KHz, thường được sử dụng để tạo ra các trễ phần mềm để vi điều khiển hoạt động trong giai đoạn khởi động. Người dùng cũng có thể kết hợp để mạch dao động ở các tần số khác như bảng dưới đây

Tần số thạch anh	C1	C2
32 Khz	33pF	33pF
100 KHz	15pF	15pF
200 KHz	15pF	15pF

Sử dụng Timer1

1. Tính toán giá trị x nạp vào TMR1 cần thiết để tạo ra trễ $T_{trễ}$ mong muốn

$$x = 65536 - \frac{T_{trễ} * f_{osc}}{4 * Prescalar}$$

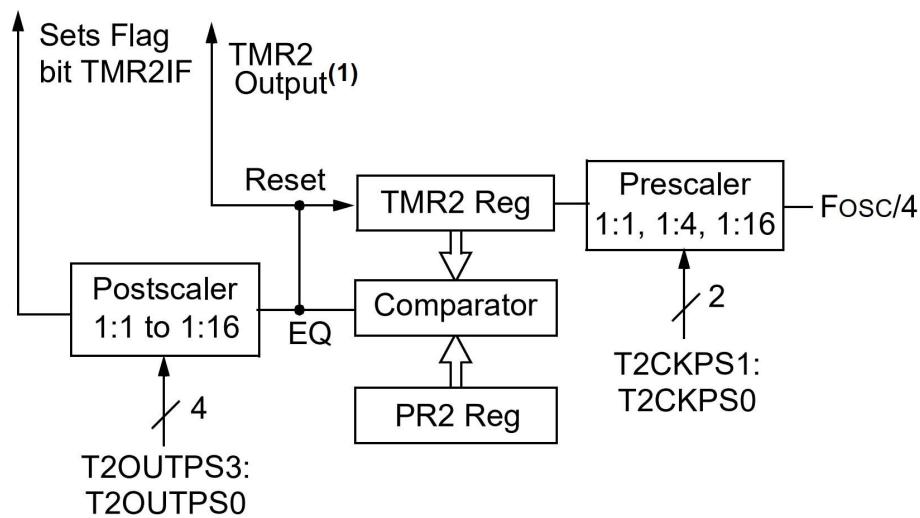
2. Thiết lập các bit T1CKPS1:T1CKPS0 trong thanh ghi T1CON tương ứng với hệ số Prescaler mong muốn
 3. Xóa bit TMR1CS để xác nhận sử dụng tần số dao động thạch anh trong việc định thời.
 4. Thiết lập chế độ đồng bộ hoặc không đồng bộ.
 5. Nạp giá trị x vào TMR1H và TMR1L..
 6. Kích hoạt Timer0 hoạt động bằng cách set bit TMR1ON và TMR1IE.
 7. Kích hoạt ngắt Timer bằng cách set các bit GIE và PEIE (Nếu dùng ngắt)

TIMER2

Timer2 là bộ định thời 8 bit được hỗ trợ bởi hai bộ chia tần Prescaler và Postscaler. Thanh ghi chứa giá trị đếm của Timer2 là TMR2. Bit cho phép giá trị trong TMR2 tăng, tức là cho phép Timer2 hoạt động là TMR2ON (bit T2CON<2>). Bit cho phép ngắt Timer2 tác động là TMR2IE (PIE<1>) và cờ ngắt tương ứng là TMR2IF (PIR1<1>).

T2CON: TIMER2 CONTROL REGISTER (ADDRESS 12h)

U-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
—	TOUTPS3	TOUTPS2	TOUTPS1	TOUTPS0	TMR2ON	T2CKPS1	T2CKPS0	bit 0



Hình 4.12: Cấu trúc của Timer2

Bộ chia tần Postscaler		Bộ chia tần Prescaler	
TOUTPS3:TOUTPS0	Hệ số chia	T2CKPS1:T2CKPS0	Hệ số chia
0000	1:1	00	1:1
0001	1:2	01	1:4
0010	1:3	1x	1:16
...	...		
1111	1:16		

Bộ chia tần Prescaler và Postscaler

Timer2 có 2 bộ chia tần được sử dụng để xác định thời gian hoạt động của bộ định tần. Để sử dụng các bộ chia tần này, người dùng sẽ xác định các giá trị tương ứng của các bit TOUTPS3:TOUTPS0 của bộ Postscaler và T2CKPS1:T2CKPS0 trong thanh ghi điều khiển Timer2 - T2CON. Ngõ xung vào của bộ Timer2 (có tần số bằng với chu kỳ máy và bằng tần số thạch anh chia 4) được đưa qua bộ Prescaler với các tỉ số chia là 1:1, 1:4 hoặc 1:16 được điều khiển bởi các bit T2CKPS1:T2CKPS0. Ngõ ra của Timer2 được đưa qua bộ chia tần Postscaler với các mức chia từ 1:1 đến 1:16 tùy theo giá trị tương ứng trong 4 bit T2OUTPS3:T2OUTPS0. Ngõ ra của Postscaler đóng vai trò quyết định trong việc điều khiển cờ ngắt.

Hoạt động của Timer2 còn được hỗ trợ bởi thanh ghi PR2. Giá trị đếm trong thanh ghi TMR2 sẽ tăng từ 00h đến giá trị chứa trong PR2, sau đó reset về 00h. Khi reset, thanh ghi PR2 được nhận giá trị mặc định là FFh.

Ngoài ra, đầu ra của Timer2 còn được đưa vào khôi module SSP. Do đó, Timer2 còn được sử dụng để tạo ra xung đồng bộ clock cho khôi giao tiếp SSP.

Tính toán giá trị x nạp vào TMR2. Để tạo ra trễ $T_{trễ}$ mong muốn, giá trị x cần nạp vào TMR2 được tính là

$$x = (PR2 + 1) - \frac{T_{trễ} * f_{osc}}{4 * Prescalar * Postscaler}$$

Ví dụ: Tạo ra trễ 10 ms

Trong ví dụ này, Timer0 được sử dụng. Khoảng trễ 10 ms được chia ra thành 10 khoảng, mỗi khoảng 1 ms, kiểm soát bởi biến đếm Count. Nếu muốn tạo ra khoảng thời gian trễ lớn hơn, ta có thể thay đổi giá trị nạp vào biến Count tương ứng. Khoảng thời gian 1 ms được tạo ra bởi bộ định thời Timer0. Với giả thiết tần số thạch anh là 20MHz, bộ chia tần 1:32 thì giá trị cần nạp vào TMR0 là 100. Đoạn code chương trình được minh họa dưới đây.

```

#include<pic.h>
int Count=0;
void main(void)
{
    TMRO=100;           // Khoi tao TMRO
    T0CS=0;             // Lua chon che do Timer

    T0SE=0;             // Kich hoat the suon len
    PSA=0;              // Lua chon bo chia tan Prescaler
    PS0=0;
    PS1=0;              // Bo chia tan 1:32
    PS2=1;
    while(1)
    {
        while(!T0IF);      //dem tu 100 - 256 roi T0IF=1
        T0IF=0;             //Reset co overflow flag
        TMRO=100;
        Count++;            //Tang gia tri dem len 1 don vi
        if(Count==10)
        {
            Count=0;        } // Khoi tao lai bo dem
    }
}

```

4.6 Khối Capture/Compare/PWM

Khối chức năng Capture/Compare/PWM (CCP) là khối thực hiện các chức năng bắt giữ (Capture), so sánh (Compare) và điều chế độ rộng xung (Pulse Width Modulation - PWM). Các chức năng này hoạt động dựa vào các bộ định thời Timer1 và Timer2. PIC16F877A cung cấp 2 bộ CCP gồm CCP1 và CCP2 có thể hoạt động độc lập. Mỗi khối có 1 thanh ghi giá trị 16 bit được ghép từ 2 thanh ghi 8 bit CCPR1H:CCPR1L và CCPR2H:CCPR2L. Ngoài ra, vi điều khiển có

CCPx	CCPy	Tác động
Capture	Capture	Dùng chung nguồn xung clock từ Timer1
Capture	Compare	Tạo sự kiện đặc biệt làm xóa Timer1
Compare	Compare	Tạo sự kiện đặc biệt làm xóa Timer1
PWM	PWM	Sử dụng chung tần số clock từ Timer2 và cùng chịu tác động của ngắt Timer2
PWM	Capture	Không ảnh hưởng, hoạt động độc lập
PWM	Compare	Không ảnh hưởng, hoạt động độc lập

thêm các thanh ghi điều khiển tương ứng là CCP1CON và CCP2CON để điều khiển hoạt động của từng khối CCP. Trong một số trường hợp, CCP1 và CCP2 có khả năng kết hợp với nhau để tạo ra các sự kiện đặc biệt hoặc tác động lên Timer1 và Timer2.

CCP1CON REGISTER/CCP2CON REGISTER (ADDRESS 17h/1Dh)

U-0	U-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
—	—	CCPxX	CCPxY	CCPxM3	CCPxM2	CCPxM1	CCPxM0

Thanh ghi điều khiển khối CCP

CCPxX và **CCPxY**: là 2 bit được sử dụng để thiết lập độ rộng xung cần điều chế (duty cycle).

CCPxM3:CCPxM0 là bit xác lập các chế độ hoạt động của khối CCP

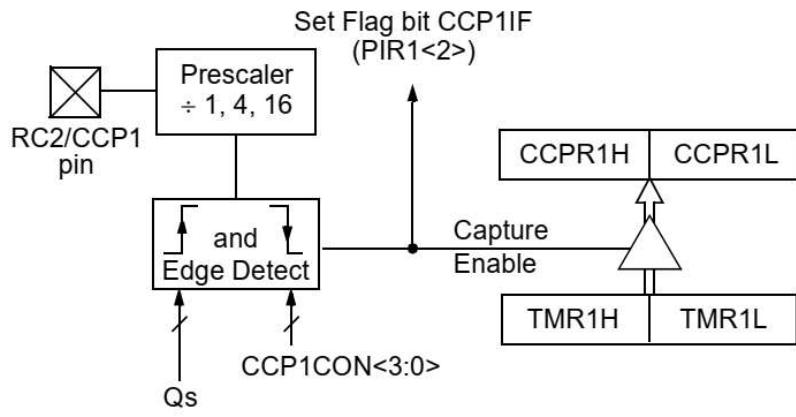
CCPxM3:CCPxM0	Chế độ hoạt động
0000	Khối CCP không hoạt động
0100	Chế độ Capture, mỗi sườn xuống
0101	Chế độ Capture, mỗi sườn lên
0110	Chế độ Capture, tại mỗi sườn lên các xung thứ 4
0111	Chế độ Capture, tại mỗi sườn lên các xung thứ 16
1000	Chế độ so sánh, set đầu ra lên 1 khi bằng nhau
1001	Chế độ so sánh, xóa đầu ra lên 1 khi bằng nhau
1010	Chế độ so sánh, kích hoạt ngắn khi bằng nhau
1011	Chế độ so sánh, tạo ra sự kiện đặc biệt khi bằng nhau
11xx	Chế độ PWM

CHẾ ĐỘ CAPTURE

Trong chế độ Capture, giá trị trong thanh ghi CCPR1H:CCPR1L sẽ "bắt" (xác định) giá trị trong thanh ghi TMR1 khi có sự kiện tác động vào chân RC2/CCP1. Sự kiện tác động được định nghĩa là 1 trong những sự kiện sau:

- Tất cả các sườn lên
- Tất cả các sườn xuống
- Tất cả các sườn lên thứ 4
- Tất cả các sườn lên thứ 16

Việc định nghĩa các sự kiện là thế nào được xác định bởi các bit CCPxM3:CCPxM0. Khi việc capture được diễn ra, cờ ngắn tương ứng sẽ được thiết lập (cờ CCP1IF

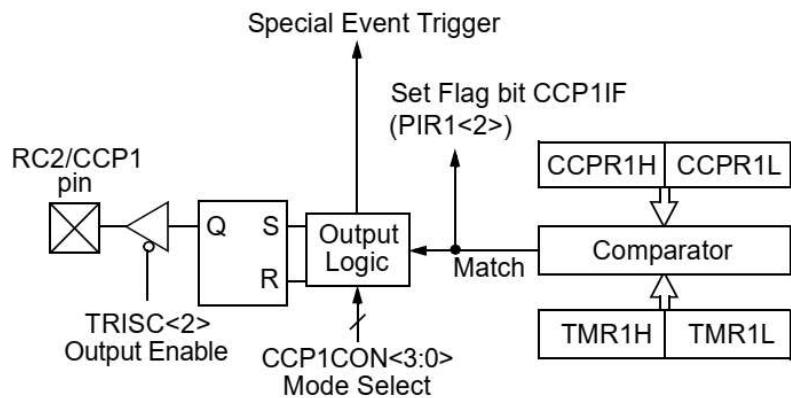


Hình 4.13: Hoạt động bộ Capture

hoặc CCP2IF). Cờ ngắn này có thể được xóa bởi phần mềm. Nếu hành động capture mới được diễn ra trước khi giá trị trong thanh ghi CCPRx được đọc, giá trị cũ sẽ bị ghi đè bởi giá trị mới.

Để chức năng Capture hoạt động tốt, các chân RC2/CCP1 và RC1/CCP2 cần được khởi tạo là các chân vào input (xác định tại TRISC). Ngoài ra, Timer1 phải hoạt động trong chế độ Timer hoặc bộ đếm đồng bộ. Trường hợp Timer1 hoạt động trong chế độ đếm không đồng bộ, chức năng Capture có thể không hoạt động. Khi thoát khỏi chế độ capture, người dùng cần xóa bit CCPxIE và CCPxIF trước khi thay đổi chế độ hoạt động để tránh các ngắt bị gọi tới một cách không mong muốn.

CHẾ ĐỘ COMPARE



Hình 4.14: Hoạt động bộ Compare

Khi hoạt động ở chế độ Compare, giá trị trong thanh ghi CCPRx sẽ thường xuyên được so sánh với giá trị trong thanh ghi TMR1H:TMR1L. Khi 2 giá trị đó bằng nhau, chân tín hiệu ra RC2/CCP1 sẽ được thay đổi trạng thái (đưa lên mức cao, đưa xuống mức thấp hoặc giữ nguyên trạng thái). Đồng thời, cờ ngắt CCP1IF cũng được set, chương trình ngắt tương ứng sẽ được gọi tới nếu ngắt CCP1IE đang ở mức 1. Sự thay đổi trạng thái của chân đầu ra có thể được điều khiển bởi các bit CCPxM3:CCPxM0.

Tương tự như chế độ Capture, Timer1 cần được xác định ở chế độ định thời hoặc chế độ đếm đồng bộ. Ngoài ra khi ở chế độ Capture, CCP có khả năng tạo ra sự kiện đặc biệt (Special event trigger) làm reset giá trị thanh ghi TMR1 và khởi động bộ chuyển đổi ADC. Điều này cho phép người dùng điều khiển giá trị thanh ghi TMR1 một cách linh động hơn.

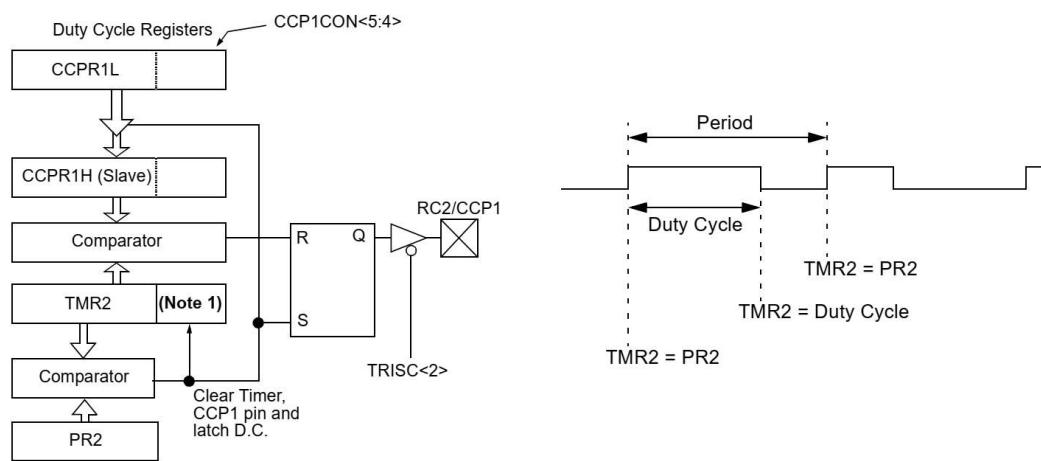
CHẾ ĐỘ PWM

Trong chế độ điều chế độ rộng xung (PWM), vi điều khiển sẽ tạo ra tín hiệu PWM có độ phân giải 10 bit và đưa ra chân CCP. Do đó, các chân CCP cần được xác định là output bằng cách xóa bit tương ứng trong thanh ghi TRISC.

Dầu ra của bộ PWM được xác định bởi khoảng thời gian cơ sở time base (period) và khoảng thời gian mà đầu ra ở trạng thái mức logic cao (duty cycle). Tần số của tín hiệu PWM là nghịch đảo của chu kì.

Chu kỳ PWM

Chu kì PWM là khoảng thời gian được xác định bởi giá trị nạp trong thanh



Hình 4.15: Hoạt động bộ điều chế độ rộng xung PWM

ghi PR2. Chu kỳ này được xác định bởi công thức

$$\text{Chu kỳ PWM} = [(PR2) + 1] * 4 * T_{\text{thạch anh}} * \text{TMR2 Prescale}$$

Tần số PWM được định nghĩa là nghịch đảo của chu kỳ PWM. Khi TMR2 bằng với giá trị PR2, 3 sự kiện sau đây diễn ra tại chu kỳ liền kề ngay đó:

- TMR2 bị xóa.

- Chân CCP1 được set. Nếu thời gian duy trì trạng thái ở mức cao là 0%, chân tín hiệu CCP sẽ không được set.

- Thời gian duy trì PWM sẽ được lấy từ CCPRR1L sang CCPR1H

Thời gian duy trì PWM - duty cycle

Thời gian duy trì PWM là giá trị 10 bit được lưu trữ đặc biệt trong thanh ghi CCPR1L và 2 bit CCP1CON<5:4>. Trong đó thanh ghi CCPR1L lưu trữ 8 bit có trọng số cao và CCP1CON<5:4> lưu trữ 2 bit có trọng số thấp. Giá trị thời gian duy trì được tính theo công thức dưới đây

$$PWM_{\text{duty cycle}} = (CCPR1L : CCP1CON < 5 : 4 >) * T_{\text{thạch anh}} * \text{TMR2 Prescale Value}$$

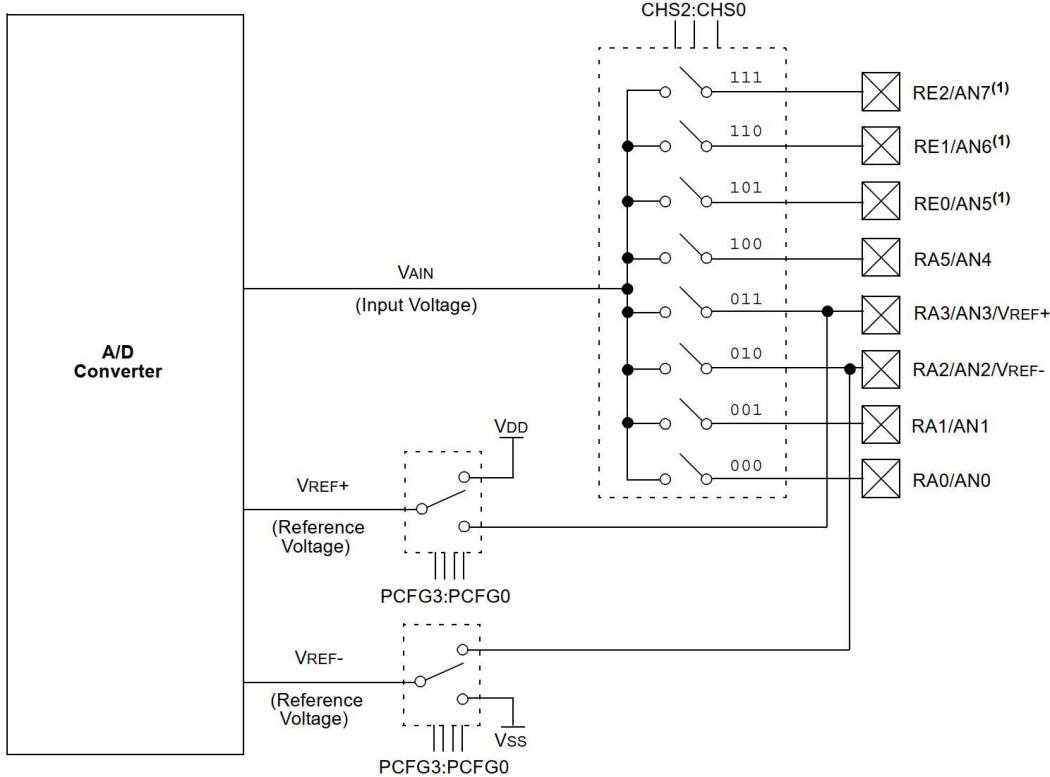
Giá trị trong CCPR1L và CCP1CON<5:4> có thể được cập nhật tại thời điểm bất kỳ, nhưng giá trị thời gian duy trì duty cycle không được nạp sang thanh ghi CCPR1H cho tới thời điểm giá trị trong TMR2 và PR2 bằng nhau. Trong chế độ PWM, thanh ghi CCPR1H là thanh ghi chỉ đọc.

Khởi tạo bộ PWM

1. Xác định chu kỳ PWM bằng cách nạp giá trị tương ứng cho PR2.
2. Xác định thời gian duy trì bằng cách nạp cho CCPR1L và CCP1CON<5:4>
3. Định nghĩa chân đầu ra CCP là output bằng cách xóa thanh ghi TRISC.
4. Xác định hệ số Prescale cho thanh ghi TMR2 và kích hoạt hoạt động cho Timer2
5. Cấu hình khối CCP1 để PWM hoạt động.

4.7 Chuyển đổi ADC

ADC (Analog to Digital Converter) là bộ chuyển đổi tín hiệu giữa 2 dạng tương tự và số. PIC16F877A có 8 ngõ vào tương tự (kí hiệu AN) tại các chân RA4:RA0 và RE2:RE0. Hiệu điện thế tham chiếu V_{REF} (dương/âm) có thể lựa chọn mặc định là điện áp được cấp vào 2 chân nguồn V_{DD} và V_{SS} hoặc được cấu hình là



Hình 4.16: Cấu trúc của khối ADC

điện áp đưa vào 2 chân RA3 và RA2. Kết quả chuyển đổi từ tín hiệu tương tự sang số là 10 bit được lưu trong 2 thanh ghi ADRESH:ADRESL. Khi không sử dụng chức năng ADC, 2 thanh ghi này có thể được sử dụng như là thanh ghi lưu trữ thông thường.

Thanh ghi điều khiển ADCON0 - điều khiển hoạt động khối ADC.

ADCS1:ADCS0 kết hợp với bit **ADCS2** trong thanh ghi ADCON1 được sử dụng để lựa chọn tần số trích mẫu cho khối AD.

CHS2:CHS0 là các bit chọn kênh đầu vào analog cho bộ chuyển đổi ADC.

GO/DONE là bit trạng thái của bộ ADC.

$GO/DONE = 1$, bộ ADC đang thực hiện chuyển đổi ADC.

$GO/DONE = 0$, quá trình chuyển đổi không diễn ra (hoặc đã thực hiện xong).

ADON là bit cho phép bộ ADC hoạt động.

ADCON1 <ADCS2>	ADCON0 <ADCS1:ADCS0>	Tần số trích mẫu
0	00	$F_{thạch anh}/2$
0	01	$F_{thạch anh}/8$
0	10	$F_{thạch anh}/32$
0	11	Tần số tính từ bộ đao động RC bên trong
1	00	$F_{thạch anh}/4$
1	01	$F_{thạch anh}/16$
1	10	$F_{thạch anh}/64$
1	11	Tần số tính từ bộ đao động RC bên trong

CHS2:CHS0	Chọn kênh đầu vào analog
000	Kênh 0 - AN0
110	Kênh 1 - AN1
010	Kênh 2 - AN2
011	Kênh 3 - AN3
100	Kênh 4 - AN4
101	Kênh 5 - AN5
110	Kênh 6 - AN6
111	Kênh 7 - AN7

ADON = 1, khối ADC bắt đầu làm việc.

ADON = 0, khối ADC ngừng hoạt động. Khi đó, khối ADC không
được cung cấp nguồn

ADCON0 REGISTER (ADDRESS 1Fh)

R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	U-0	R/W-0
ADCS1	ADCS0	CHS2	CHS1	CHS0	GO/DONE	—	ADON
bit 7							bit 0

Thanh ghi điều khiển ADCON1 - điều khiển các cổng cho khối ADC.

ADFM là các bit định dạng kết quả lưu dữ liệu của đầu ra ADC.

ADFM = 1: dữ liệu 10 bit lưu trong 10 bit phía phải của 16 bit
ADRESH:ADRESL, các bit còn lại là 0.

ADFM = 0: dữ liệu 10 bit lưu trong 10 bit phía trái của 16 bit
ADRESH:ADRESL, các bit còn lại là 0.

PCFG <3:0>	AN 7	AN 6	AN 5	AN 4	AN 3	AN 2	AN 1	AN 0	V_{REF+}	V_{REF-}	C/V
0000	A	A	A	A	A	A	A	A	V_{DD}	V_{SS}	8/0
0001	A	A	A	A	V_{REF+}	A	A	A	AN3	V_{SS}	7/1
0010	D	D	D	A	A	A	A	A	V_{DD}	V_{SS}	5/0
0011	D	D	D	A	V_{REF+}	A	A	A	AN3	V_{SS}	4/1
0100	D	D	D	D	A	D	A	A	V_{DD}	V_{SS}	3/0
0101	D	D	D	D	V_{REF+}	D	A	A	AN3	V_{SS}	2/1
011x	D	D	D	D	D	D	D	D	-	-	0/0
1000	A	A	A	A	V_{REF+}	V_{REF-}	A	A	AN3	AN2	6/2
1001	D	D	A	A	A	A	A	A	V_{DD}	V_{SS}	6/0
1010	D	D	A	A	V_{REF+}	A	A	A	AN3	V_{SS}	5/1
1011	D	D	A	A	V_{REF+}	V_{REF-}	A	A	AN3	AN2	4/2
1100	D	D	D	A	V_{REF+}	V_{REF-}	A	A	AN3	AN2	3/2
1101	D	D	D	D	V_{REF+}	V_{REF-}	A	A	AN3	AN2	2/2
1110	D	D	D	D	D	D	D	A	V_{DD}	V_{SS}	1/0
1111	D	D	D	D	V_{REF+}	V_{REF-}	D	A	AN3	AN2	1/2

ADCS2: là bit kết hợp với 2 bit ADCS1:ADCS0 ở thanh ghi ADCON0 nhằm xác định tần số trích mẫu ADC.

PCFG3:PCFG0: các bit xác định cấu hình cổng, số lượng cổng đầu vào ADC.

ADCON1 REGISTER (ADDRESS 9Fh)

R/W-0	R/W-0	U-0	U-0	R/W-0	R/W-0	R/W-0	R/W-0
ADFM	ADCS2	—	—	PCFG3	PCFG2	PCFG1	PCFG0

bit 7

bit 0

Các bước thực hiện chuyển đổi ADC

1. Cấu hình cho khối ADC:

- Cấu hình lựa chọn các cổng vào analog, điện áp tham chiếu và cổng vào ra số tại các bit PCFG3:PCFG0 trong thanh ghi ADCON1.

- Chọn đầu vào analog tại thanh ghi ADCON0.
- Chọn tần số trích mẫu của chuyển đổi ADC (ADCON0).
- Set bit ADON để khởi ADC hoạt động.

2. Cấu hình ngắt A/D nếu sử dụng bao gồm:

- Xóa bit ADIF.
- Set bit ADIE.

-
- Set bit PEIE.
 - Set bit GIE.
- Chờ một khoảng thời gian, tối thiểu $37\mu s$ để bộ ADC được cập nhật.
 - Bắt đầu chuyển đổi bằng cách set bit GO/\overline{DONE} của thanh ghi ADCON0.
 - Chờ khối ADC thực hiện chuyển đổi xong bằng cách kiểm tra:
 - trạng thái của bit GO/\overline{DONE} . Nếu bit này bị xóa về 0 thì quá trình chuyển đổi đã xong, hoặc
 - trạng thái của cờ chuyển đổi ADC (bit ADIF).
 - Đọc kết quả trong 2 thanh ghi $<ADRESH:ADRESL>$, xóa bit ADIF nếu cần
 - Quay lại bước 1 hoặc bước 2 nếu cần thiết để chuẩn bị cho quá trình chuyển đổi vòng tiếp theo.

Chuyển đổi ADC trong chế độ Sleep

Khối chuyển đổi ADC có thể hoạt động ngay cả trong chế độ Sleep. Để thực hiện được điều này, nguồn xung nhịp trích mẫu cần được gán là nguồn dao động RC (ADCS1:ADCS0=11). Khi nguồn dao động RC được chọn, khối ADC sẽ chờ trong 1 chu kỳ lệnh trước khi bắt đầu thực hiện chuyển đổi. Việc cài đặt này sẽ cho phép lệnh SLEEP hoạt động. Khi việc chuyển đổi thực hiện xong, bit GO/\overline{DONE} sẽ bị xóa và kết quả chuyển đổi được chuyển vào thanh ghi ADRESH:ADRESL. Khi ngắt ADC được kích hoạt, vi điều khiển sẽ thoát khỏi chế độ SLEEP. Nếu ngắt ADC không được kích hoạt, khối ADC sẽ bị ngắt nguồn cho dù bit ADON vẫn được set.

4.8 Khối so sánh tương tự

Khối so sánh tương tự bao gồm 2 bộ so sánh với các đầu vào được tích hợp từ các chân RA0 đến RA3, và đầu ra là chân RA4 và RA5 của vi điều khiển. Ngoài ra, người dùng có thể sử dụng điện áp tham chiếu được tạo ra bởi module tạo điện áp so sánh (mục 4.9) để làm đầu vào input cho khối so sánh tương tự này. Khi giá trị điện áp đầu vào tương tự V_{IN+} nhỏ hơn đầu vào tương tự V_{I-+} , đầu ra của bộ so sánh được xác định ở mức thấp (mức 0). Ngược lại, giá trị của đầu ra được xác lập ở mức logic cao (mức 1). Hoạt động của khối so sánh tương tự được điều khiển bởi thanh ghi CMCON có cấu hình dưới đây

C2OUT và **C1OUT** là 2 bit kết quả của bộ so sánh. Giá trị của nó là 1 nếu đầu vào tương ứng $V_{IN+} > V_{IN-}$ và bằng 0 nếu ngược lại $V_{IN+} < V_{IN-}$.

CMCON REGISTER

R-0	R-0	R/W-0	R/W-0	R/W-0	R/W-1	R/W-1	R/W-1
C2OUT	C1OUT	C2INV	C1INV	CIS	CM2	CM1	CM0
bit 7				bit 0			

C2INV và **C1INV** là 2 bit nghịch đảo của C2OUT và C1OUT tương ứng.

CIS là bit xác định cấu hình đầu vào input của bộ so sánh trong trường hợp giá trị CM2:CM0 = 110. Người đọc có thể xem phần cấu hình bộ so sánh ngay sau đây để có thêm thông tin chi tiết.

CM2:CM0 là các bit xác định cấu hình sử dụng bộ so sánh (hình 4.17)

So sánh với điện áp tham chiếu

Khối so sánh điện áp thường được cấu hình để so sánh 2 điện áp tương tự với nhau. Hai điện áp này được đưa vào các chân V_{IN+} và V_{IN-} tương ứng. Điện áp so sánh phải có giá trị nằm trong khoảng từ điện áp nối đất V_{SS} đến điện áp nguồn V_{DD} . Thông thường các tín hiệu vào đều được lấy từ bên ngoài. Tuy nhiên trong trường hợp 3 bit CM<2:0> có giá trị 110 thì vi điều khiển sẽ có thể nhận tín hiệu đi vào chân V_{IN+} từ đầu ra của khối tạo điện áp so sánh.

Tín hiệu ngắt so sánh tương tự

Trong ngắt ADC, vi điều khiển sẽ xác lập cờ ngắt ADIF ngay khi quá trình chuyển đổi ADC được thực hiện xong để làm tín hiệu tác động ngắt đến vi điều khiển. Hoàn toàn tương tự như vậy, mỗi khi có sự thay đổi tại bất cứ đầu ra bộ so sánh (CMON<7:6>), phần mềm sẽ phát hiện và set bit cờ ngắt so sánh CMIF (PIR2<6>). Khi đó chương trình ngắt so sánh tương tự sẽ được gọi tới nếu bit cho phép ngắt CMIE (PIE2<6>), các bit GIE, PEIE đều được set là 1.

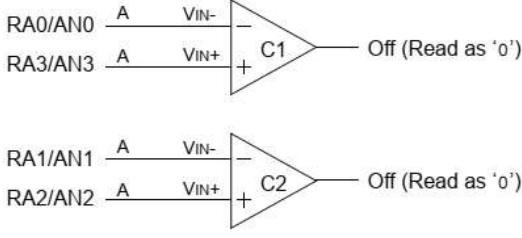
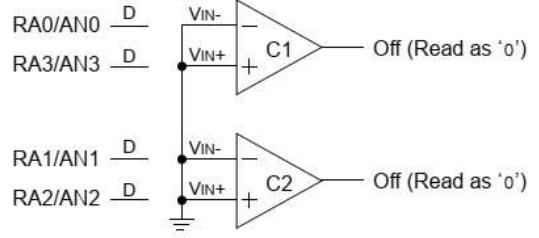
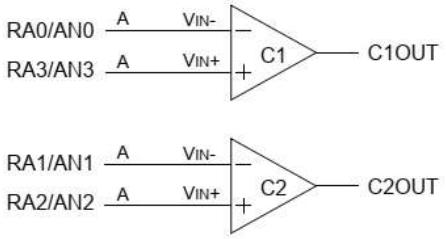
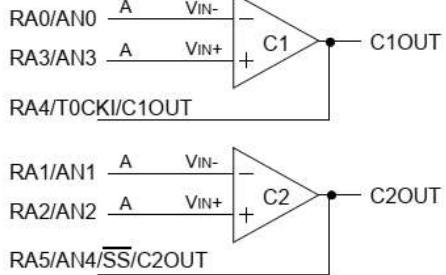
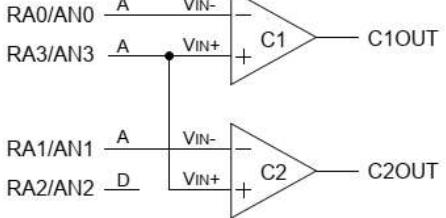
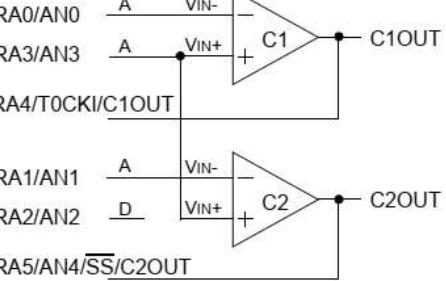
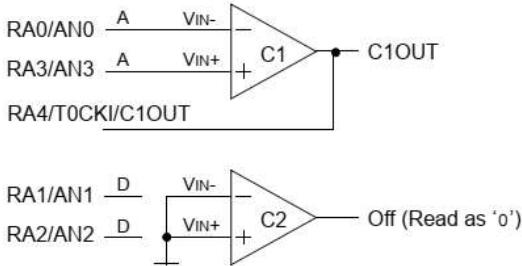
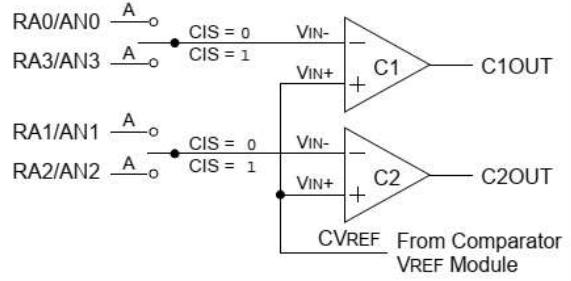
Trong chương trình ngắt tương tự, người dùng cần thiết phải xóa tín hiệu ngắt bằng 1 trong 2 phương thức sau:

- Đọc hoặc ghi vào thanh ghi CMCON, hoặc
- Xóa bit CMIF.

Hoạt động trong chế độ Sleep

Trong chế độ Sleep, nếu bộ so sánh vẫn được tích cực, hoạt động của bộ so sánh và ngắt bộ so sánh tương tự vẫn có khả năng hoạt động. Tác động ngắt bộ so sánh sẽ có thể đánh thức hoạt động của vi điều khiển. Tuy nhiên, bộ vi điều khiển có khả năng bị rơi vào trạng thái khởi động lại vì công suất dòng tiêu thụ cho bộ so sánh lớn hơn công suất dòng tiêu thụ trong chế độ Sleep. Chính vì

COMPARATOR I/O OPERATING MODES

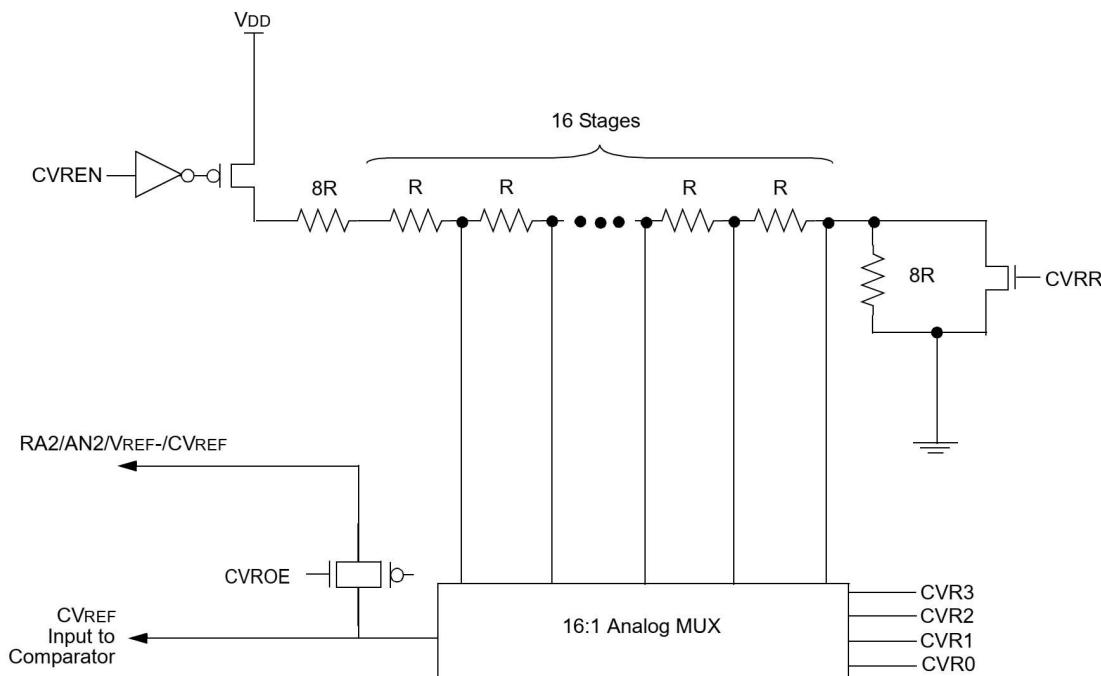
Comparators Reset CM2:CM0 = 000  <p>Off (Read as '0')</p>	Comparators Off (POR Default Value) CM2:CM0 = 111  <p>Off (Read as '0')</p>
Two Independent Comparators CM2:CM0 = 010  <p>C1OUT</p> <p>C2OUT</p>	Two Independent Comparators with Outputs CM2:CM0 = 011  <p>C1OUT</p> <p>RA4/T0CKI/C1OUT</p> <p>C2OUT</p> <p>RA5/AN4/SS/C2OUT</p>
Two Common Reference Comparators CM2:CM0 = 100  <p>C1OUT</p> <p>C2OUT</p>	Two Common Reference Comparators with Outputs CM2:CM0 = 101  <p>C1OUT</p> <p>RA4/T0CKI/C1OUT</p> <p>C2OUT</p> <p>RA5/AN4/SS/C2OUT</p>
One Independent Comparator with Output CM2:CM0 = 001  <p>C1OUT</p> <p>RA4/T0CKI/C1OUT</p> <p>Off (Read as '0')</p>	Four Inputs Multiplexed to Two Comparators CM2:CM0 = 110  <p>C1OUT</p> <p>C2OUT</p> <p>CVREF From Comparator VREF Module</p>

Hình 4.17: Các cấu hình sử dụng của bộ so sánh tương tự

thế, trước khi đưa vi điều khiển vào chế độ Sleep, người dùng nên tắt/khóa bộ so sánh bằng cách đặt giá trị của 3 bit CM_{2:0} là 111.

4.9 Khối tạo điện áp tham chiếu

Khối tạo điện áp tham chiếu làm một mạch gồm 16 điện trở mắc nối tiếp với nhau (hình 4.18) có khả năng tạo ra điện áp tham chiếu có độ chính xác cao cung cấp cho bộ so sánh điện áp tương tự trong chế độ 110. Khi lập trình cho thanh ghi điều khiển CVRCON, người dùng có thể thay đổi được giá trị điện áp tham chiếu đầu ra. Có thể thấy ngay trên mạch nguyên lý, bộ tạo điện áp tham chiếu có thể tạo ra 2 dải giá trị điện áp đầu ra tùy thuộc vào giá trị logic của bit CVRR và cũng có thể được tách ra khỏi điện áp nguồn cung cấp V_{DD} khi cần giảm công suất hoạt động của vi điều khiển. Tín hiệu đầu ra của bộ tạo điện áp tham chiếu có thể được đưa ra chân RA2/AN2/ V_{REF-}/CV_{REF} . Khi đó, điện áp ra có thể được dùng như một tín hiệu chuyển đổi số/tương tự D/A đơn giản (thường dùng để test). Hoặc, tín hiệu ra được đưa trực tiếp vào đầu vào của bộ so sánh tương tự.



Hình 4.18: Bộ tạo điện áp tham chiếu

CVRCON CONTROL REGISTER (ADDRESS 9Dh)

R/W-0	R/W-0	R/W-0	U-0	R/W-0	R/W-0	R/W-0	R/W-0
CVREN	CVROE	CVRR	—	CVR3	CVR2	CVR1	CVR0

CVREN: bit cho phép khối tạo điện áp tham chiếu hoạt động. Khi CVREN = 1, khối được cấp nguồn. Ngược lại khi CVREN = 0, mạch không được cấp nguồn.

CVROE: bit cho phép tín hiệu được đưa tới đầu ra. Khi CVROE = 1, tín hiệu ra được đưa tới chân RA2/AN2/ V_{REF-}/CV_{REF} .

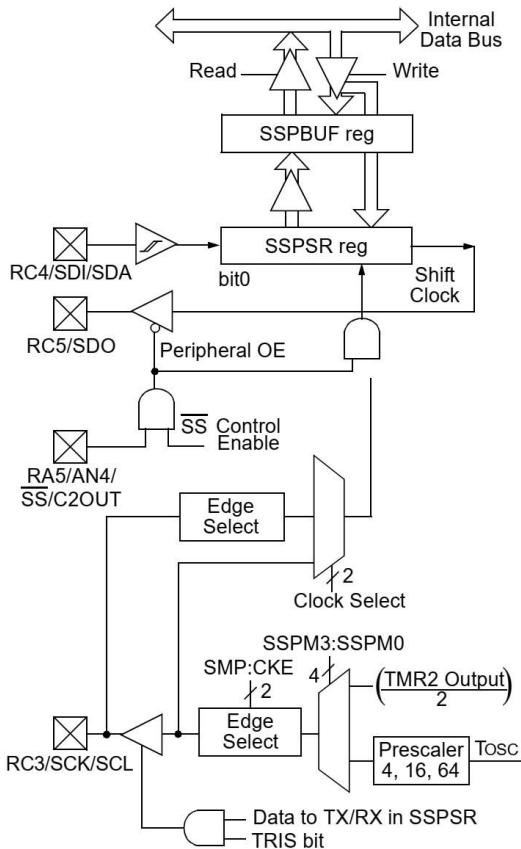
CCRR: bit lựa chọn khoảng điện áp ra. Nếu CVRR = 1, điện áp ra có giá trị trong khoảng từ 0 đến 0,75 giá trị điện áp V_{DD} với bước tính là $V_{DD}/24$. Nếu CVRR = 0, điện áp ra trong khoảng từ 0,25 đến 0,75 giá trị của điện áp V_{DD} với bước tính là $V_{DD}/32$.

CVR3:CVR0: các bit xác định giá trị đầu ra của bộ tạo điện áp tham chiếu.

Giá trị CVRR	Điện áp tham chiếu đầu ra CV_{REF}
0	$\frac{CVR<3:0>}{24} V_{DD}$
1	$\frac{1}{4}V_{DD} + \frac{CVR<3:0>}{32} V_{DD}$

4.10 Giao tiếp truyền thông

Trong hoạt động của vi điều khiển, ngoài chức năng đọc và xử lý các tín hiệu số hay tín hiệu tương tự, các ứng dụng còn cần tới khả trao đổi thông tin giữa các thiết bị. Để thực hiện điều này, PIC16F877A được trang bị các giao thức để truyền/nhận dữ liệu nối tiếp SPI (Serial Peripheral Interface), giao thức truyền thông I2C (Inter-Integrated Circuit), giao thức truyền/nhận không đồng bộ USART (Universal Synchronous and Asynchronous Receiver-Transmitter), và giao thức truyền song song. Trong phạm vi của tài liệu này, chúng tôi sẽ trình bày về các giao thức truyền nối tiếp SPI, I2C và USART bởi vì đây là những giao thức quen thuộc và được dùng trong các mạch điện tử và các ứng dụng điều khiển và tự động hóa.



Hình 4.19: Khối giao thức SPI

Giao thức SPI

SPI là giao thức truyền thông nối tiếp được sử dụng để thực hiện giao tiếp điểm-điểm giữa các thiết bị. Cấu hình của khối giao thức SPI được trình bày trong hình 4.19. Trong đó, giao thức sử dụng 3 chân tín hiệu SDO, SDI, SCK cho giao thức cơ bản và thêm chân tín hiệu \overline{SS} cho giao thức trong chế độ slave

- RC5/SDO: Serial Data Out.
- RC4/SDI/SDA: Serial Data In.
- RC3/SCK/SCL: Serial Clock
- RA5/AN4/ \overline{SS} /C2OUT: Slave Select.

Các thanh ghi điều khiển hoạt động trong giao thức SPI

Để cấu hình cho giao thức SPI, người dùng cần tương tác với 4 thanh ghi: thanh ghi điều khiển SSPCON1, thanh ghi trạng thái SSPSTAT, thanh ghi đệm SSPBUF và thanh ghi dịch SSPSR.

SSPCON1: MSSP CONTROL REGISTER 1 (SPI MODE) (ADDRESS 14h)

| R/W-0 |
|-------|-------|-------|-------|-------|-------|-------|-------|
| WCOL | SSPOV | SSPEN | CKP | SSPM3 | SSPM2 | SSPM1 | SSPM0 |

WCOL: bit phát hiện xung đột truyền dữ liệu.

WCOL = 0: không có xung đột.

WCOL = 1: có xung đột (thanh ghi SSPBUF được nạp dữ liệu khi nội dung cũ chưa truyền đi xong)

SSPOV: bit chỉ thị quá trình nhận bị tràn

SSPOV = 0: không tràn

SSPOV = 1: SSPBUF đã nhận dữ liệu mới khi mà dữ liệu cũ vẫn chưa được xử lý.

SSPEN: bit cho phép/không cho phép sử dụng cổng SPI.

CKP: bit lựa chọn phân cực clock.

CKP = 1: trạng thái nghỉ của xung nhịp ở mức cao.

CKP = 0: trạng thái nghỉ của xung nhịp ở mức thấp.

SSPM3:SSPM0: bit lựa chọn chế độ hoạt động

SSPM3:SSPM0	Chế độ
0101	Chế độ Slave, clock lấy từ chân SCK, không dùng chân SS
0100	Chế độ Slave, clock lấy từ chân SCK, có dùng chân SS
0011	Chế độ Master, clock lấy từ TMR2/2
0010	Chế độ Master, clock = F _{thạch anh} /64
0001	Chế độ Master, clock = F _{thạch anh} /16
0000	Chế độ Master, clock = F _{thạch anh} /4

SSPSTAT: MSSP STATUS REGISTER (SPI MODE) (ADDRESS 94h)

R/W-0	R/W-0	R-0	R-0	R-0	R-0	R-0	R-0
SMP	CKE	D/A	P	S	R/W	UA	BF

SMP: bit xác định thời điểm lấy mẫu. Bit này bị xóa trong chế độ slave. Trong chế độ master, SMP = 1(hoặc 0), dữ liệu input được lấy mẫu tại thời điểm cuối(hoặc thời điểm giữa) của đầu ra dữ liệu.

CKE: bit chọn thời điểm truyền SPI.

CKE = 1: việc truyền diễn ra khi xung nhịp chuyển từ trạng thái tích cực sang trạng thái nghỉ.

CKE = 0: việc truyền diễn ra khi xung nhịp chuyển từ trạng thái nghỉ sang trạng thái tích cực.

D/A: bit chọn dữ liệu/địa chỉ, chỉ dùng trong chế độ I2C.

P: bit dừng - stop bit, chỉ dùng trong chế độ I2C.

S: bit bắt đầu - start bit, , chỉ dùng trong chế độ I2C.

R/W: bit đọc/ghi, chỉ dùng trong chế độ I2C.

UA: cập nhật chế độ địa chỉ, chỉ dùng trong chế độ I2C.

BF: bit chỉ thị trạng thái bộ đệm buffer. Khi SSPBUF đầy, BF = 1.

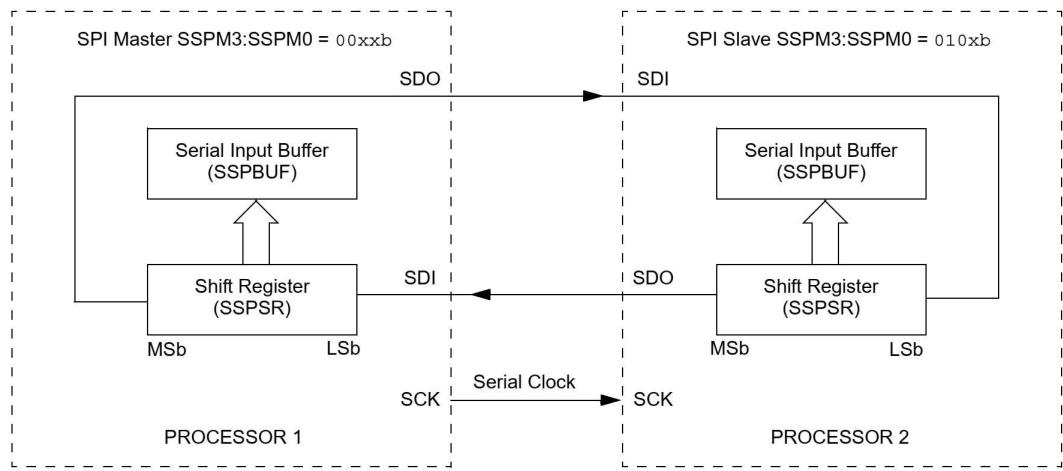
Nguyên tắc hoạt động của giao thức SPI

Khi giao thức SPI được khởi tạo, tùy theo các giá trị của thanh ghi SSPCON và SSPSTAT, vi điều khiển sẽ xác lập chế độ hoạt động của nó cũng như các đặc tính của các chế độ. Các đặc tính này gồm:

- Chế độ master với SCK là output.
- Chế độ slave với SCK là input.
- Xác định trạng thái nghỉ của SCK
- Xác định thời điểm trích mẫu của dữ liệu input.
- Xác định sườn tác động của clock.
- Xác định tần số clock.
- Chế độ chọn slave.

Hình 4.20 mô tả kết nối giữa hai vi điều khiển trong giao thức SPI. Chân tín hiệu SDO của vi điều khiển này được nối với chân SDI của vi điều khiển kia và ngược lại. Với chân SCK, tín hiệu được truyền từ vi điều khiển master sang vi điều khiển slave.

Khi hoạt động, các bit dữ liệu trong thanh ghi dịch SSPSR sẽ được truyền từ vi điều khiển này sang vi điều khiển kia và ngược lại bắt đầu từ bit có trọng số cao. Khi toàn bộ các bit trong thanh ghi dịch SSPSR được truyền đi và nhận về hết, dữ liệu sẽ được chuyển sang thanh ghi bộ đệm SSPBUF. Ngay sau đó, bit BF (SSPSTAT<0>) sẽ được set lên 1 và cờ ngắt truyền dữ liệu SSPIF cũng được set. Việc chuyển dữ liệu từ thanh ghi dịch SSPSR sang thanh ghi bộ đệm SSPBUF cho phép việc truyền dữ liệu kế tiếp được thực hiện ngay trước khi dữ liệu vừa đọc về được vi điều khiển xử lý. Trong khi dữ liệu đang được truyền/nhận giữa



Hình 4.20: Kết nối trong giao thức SPI

các vi điều khiển, việc thay đổi nội dung của thanh ghi đệm SSPBUF sẽ không được thực hiện (không thể thay đổi được SSPBUF trong lúc này). Nếu có lệnh thay đổi dữ liệu, bit WCOL sẽ được set thông báo trạng thái xung đột đã diễn ra. Người dùng sẽ phải xóa bit WCOL này khi quá trình truyền/nhận dữ liệu kết thúc.

Tùy thuộc vào mục đích của việc truyền dữ liệu, hệ thống có thể ở 1 trong 3 trạng thái truyền/nhận dữ liệu sau:

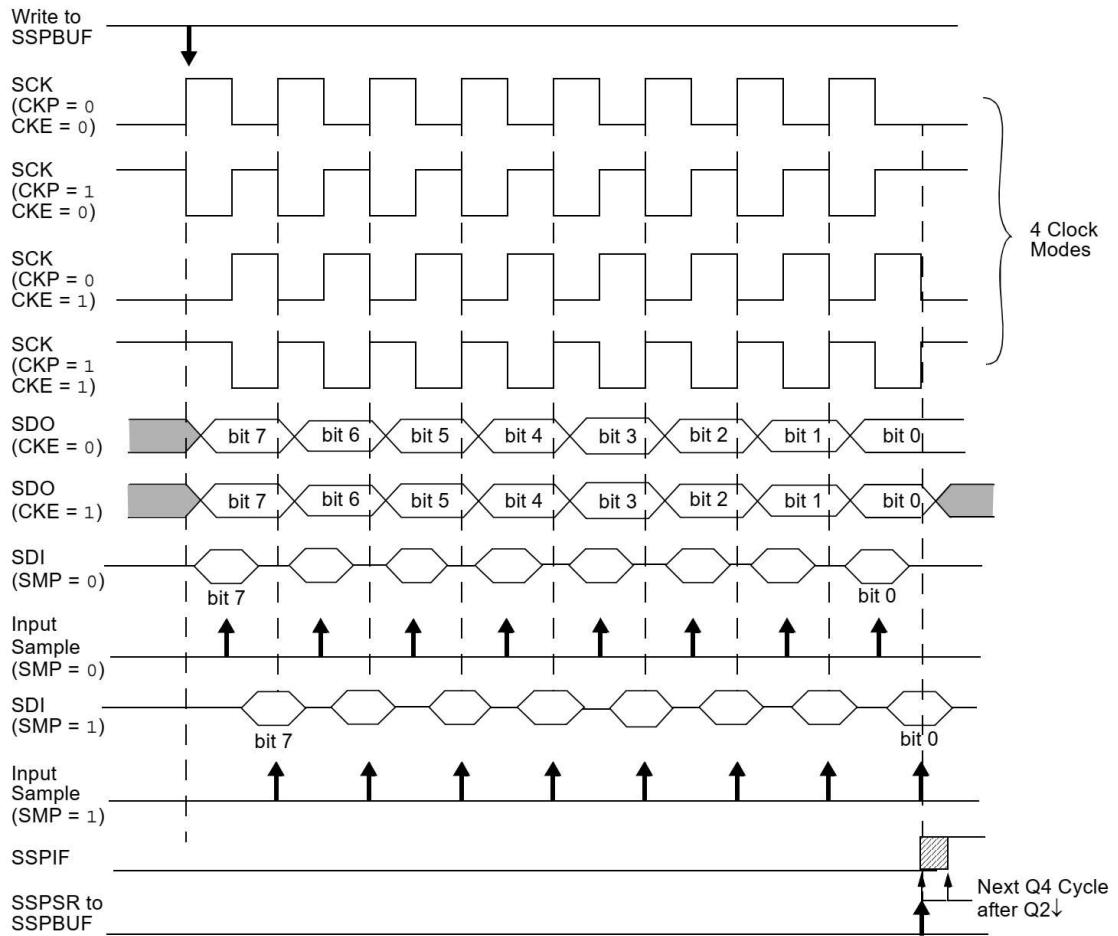
- master truyền dữ liệu - slave truyền "dữ liệu giả".
- master truyền dữ liệu - slave truyền dữ liệu.
- master truyền "dữ liệu giả" - slave truyền dữ liệu.

Chế độ master

Vi điều khiển master khởi tạo quá trình truyền/nhận dữ liệu bất kì lúc nào bởi nó kiểm soát xung nhịp SCK. Điều này có nghĩa là vi điều khiển master sẽ quyết định khi nào vi điều khiển slave có quyền trao đổi dữ liệu.

Trong chế độ master, dữ liệu được truyền/nhận ngay khi thanh ghi SSPBUF được chép dữ liệu từ SSPSR. Nếu vi điều khiển master chỉ nhận dữ liệu, chân truyền dữ liệu (SDO) có thể được khóa (disabled) bởi phần mềm Chế độ slave. Với mỗi byte dữ liệu nhận được, dữ liệu sẽ được nạp vào SSPBUF như một byte thông thường (bit BF và SSPIF được set)

Hình 4.21 mô tả dạng tín hiệu của quá trình truyền/nhận dữ liệu của vi điều khiển trong chế độ master. Khi tín hiệu CKE được set, dữ liệu trên chân SDO



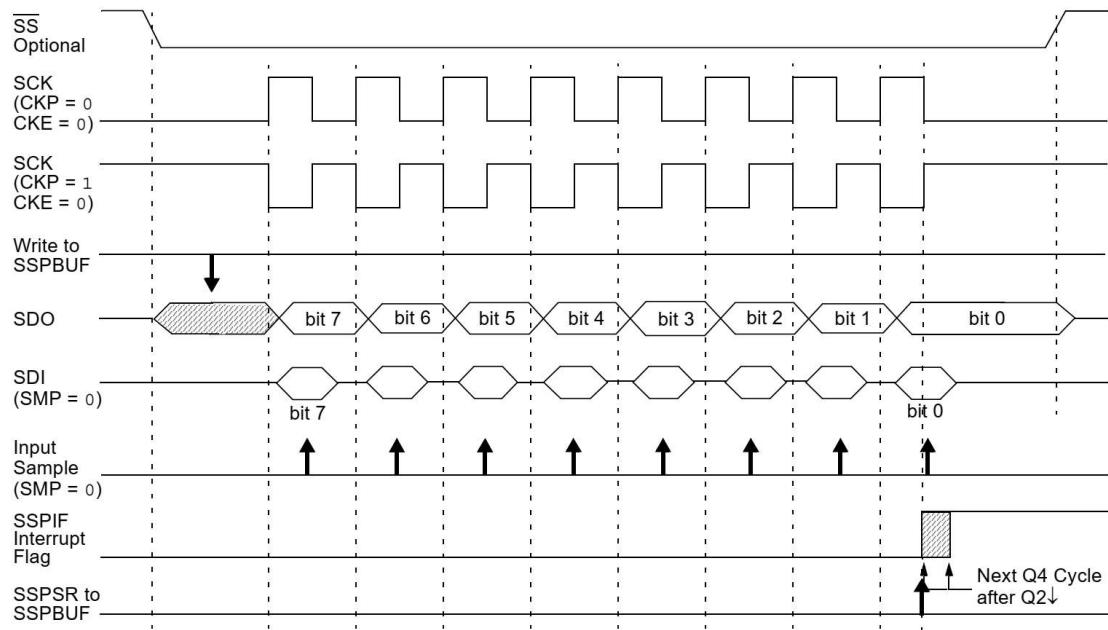
Hình 4.21: Giản đồ thời gian của các tín hiệu trong giao thức SPI chế độ master

sẽ có hiệu lực lực trước khi gấp sườn xung đầu tiên của SCK. Sự khác biệt của thời gian lấy mẫu (đọc) của tín hiệu input phụ thuộc vào giá trị của bit SMP. Khi toàn bộ dữ liệu được truyền/nhận, thanh ghi SSPBUF sẽ được nạp.

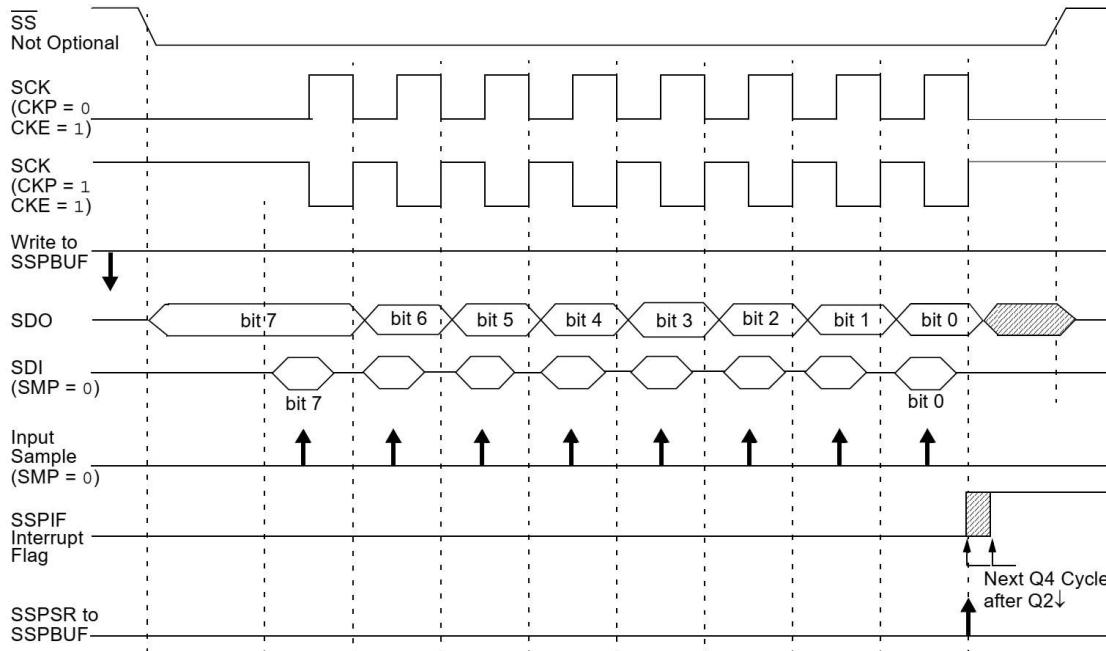
Chế độ Slave

Trong chế độ Slave, dữ liệu truyền/nhận phụ thuộc vào xung nhịp đưa vào chân SCK. Khi bit dữ liệu cuối cùng được chốt, cờ SSPIF sẽ được set. Ngoài ra, vi điều khiển vẫn có thể truyền/nhận dữ liệu trong chế độ Sleep. Khi nhận được đủ 1 byte, vi điều khiển sẽ được đánh thức.

Chân tín hiệu \overline{SS} được tích cực (mức logic 0) sẽ cho phép vi điều khiển hoạt động ở chế độ đồng bộ. Khi tín hiệu tại chân \overline{SS} ở mức thấp, quá trình truyền/n-



Hình 4.22: Giản đồ thời gian của các tín hiệu trong giao thức SPI chế độ slave với $CKE=0$



Hình 4.23: Giản đồ thời gian của các tín hiệu trong giao thức SPI chế độ slave với $CKE=1$

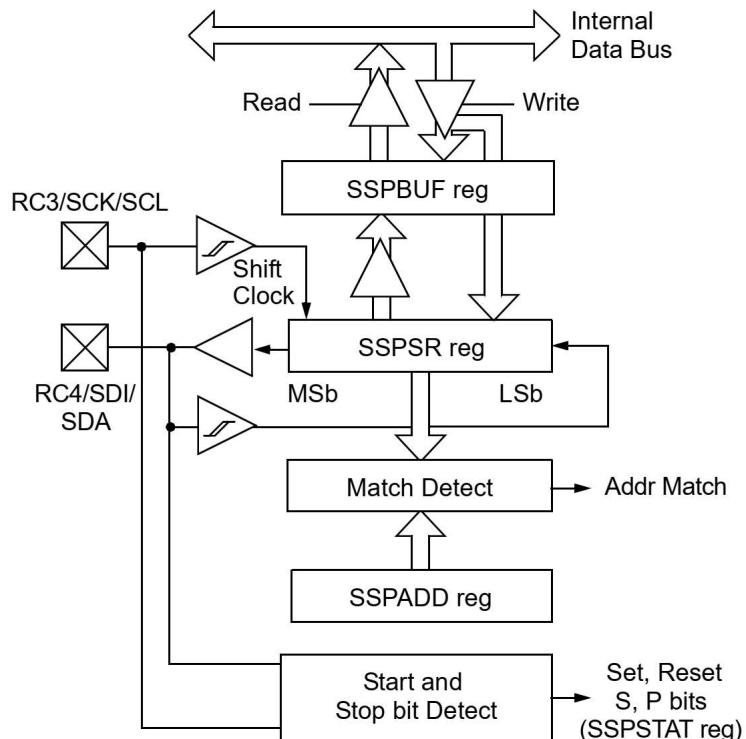
hận được kích hoạt và chân tín hiệu SDO có thể xuất được dữ liệu ra. Khi tín hiệu tại \overline{SS} ở mức cao, chân SDO không thể thay đổi được trạng thái logic dù đang trong quá trình truyền/nhận dữ liệu và trở thành một cổng ra bị thả nổi. Khi đó, điện trở pull-up/pull-down cần được thiết kế tùy thuộc vào từng ứng dụng.

Cũng giống như trong chế độ Master, giá trị của bit CKE cũng sẽ ảnh hưởng đến thời điểm có ý nghĩa của các bit truyền/nhận và giá trị của SMP quyết định thời điểm đọc/ghi của các bit dữ liệu (hình 4.22 và 4.23).

Giao thức I2C

PIC16F877A cũng tích hợp giao thức truyền thông I2C với 2 chế độ Master và Slave có kết nối với ngắt. Giao thức I2C sử dụng 2 chân để truyền/nhận dữ liệu (hình 4.24):

- RC3/SCK/SCL: xung nhịp truyền nối tiếp (SCL).
- RC4/SDI/SDA: dữ liệu truyền nối tiếp (SDA).



Hình 4.24: Khối giao thức I2C

Dể điều khiển hoạt động của giao thức truyền/nhận I2C, vi điều khiển sẽ sử dụng tới 6 thanh ghi bao gồm: hai thanh ghi điều khiển SSPCON1 và SSPCON2, thanh ghi trạng thái SSPSTAT, thanh ghi đệm SSPBUF, thanh ghi dịch SSPSR và thanh ghi địa chỉ SSPADD.

SSPCON1: MSSP CONTROL REGISTER 1 (I²C MODE) (ADDRESS 14h)

| R/W-0 |
|-------|-------|-------|-------|-------|-------|-------|-------|
| WCOL | SSPOV | SSPEN | CKP | SSPM3 | SSPM2 | SSPM1 | SSPM0 |

WCOL: bit phát hiện xung đột.

SSPOV: bit xác định quá trình nhận bị tràn.

SSPEN: bit cho phép truyền đồng bộ

CKP: bit điều khiển giải phóng SCK dùng trong chế độ slave.

SSPM3:SSPM0: bit xác định chế độ truyền nối tiếp đồng bộ.

SSPM3:SSPM0	Chế độ
1111	I2C chế độ Slave, 10 bit địa chỉ, cho phép ngắn Start và Stop bit
1110	I2C chế độ Slave, 7 bit địa chỉ, cho phép ngắn Start và Stop bit
1011	I2C chế độ điều khiển master firmware
1000	I2C chế độ master, clock = $F_{thạch anh}/(4 * (SSPADD + 1))$
0111	I2C chế độ slave, 10 bit địa chỉ
0110	I2C chế độ slave, 7 bit địa chỉ

SSPCON2: MSSP CONTROL REGISTER 2 (I²C MODE) (ADDRESS 91h)

R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
GCEN	ACKSTAT	ACKDT	ACKEN	RCEN	PEN	RSEN	SEN

GCEN: bit cho phép kết nối chung.

GCEN = 1: cho phép ngắn khi nhận dữ liệu từ địa chỉ 0000h ở SSPSR.

GCEN = 0: không cho phép

ACKSTAT: bit báo trạng thái truyền

ACKSTAT = 1: trạng thái không nhận được từ slave.

ACKSTAT = 0: trạng thái đã nhận được từ slave.

ACKDT: bit báo nhận được dữ liệu.

ACKDT = 1: chưa nhận được dữ liệu.

ACKDT = 0: đã nhận được dữ liệu.

ACKEN: bit cho phép nhận chuỗi bit (chế độ nhận Master)

ACKEN = 1 : Khởi tạo nhận chuỗi trên chân SDA và SCL và truyền bit dữ liệu ACKDT.

ACKEN = 0: không truyền chuỗi dữ liệu.

RCEN: bit cho phép nhận (chỉ chế độ Master)

RCEN = 1: cho phép chế độ nhận I2C

RCEN = 0: không cho phép nhận

PEN: bit cho phép điều kiện dừng.

PEN = 1: Khởi tạo điều kiện dừng trên chân SDA và SCL. Tự động xóa bởi phần cứng.

PEN = 0: Không có điều kiện dừng.

RSEN: bit cho phép lặp lại điều kiện bắt đầu (chế độ master)

RSEN = 1: khởi tạo chế độ lặp lại điều kiện bắt đầu trên chân SDA và SCL.

RSEN = 0: không cho phép.

SEN: bit cho phép điều kiện bắt đầu/kéo dài

Chế độ Master:

SEN = 1: Khởi tạo điều kiện bắt đầu trên chân SDA và SCL

SEN = 0: không khởi tạo

Chế độ Slave:

SEN = 1: kéo dài xung nhịp clock cho quá trình truyền và nhận của slave

SEN = 0: kéo dài xung nhịp clock chỉ cho quá trình truyền của slave.

SSPSTAT: MSSP STATUS REGISTER (I²C MODE) (ADDRESS 94h)

R/W-0	R/W-0	R-0	R-0	R-0	R-0	R-0	R-0
SMP	CKE	D/A	P	S	R/W	UA	BF

SMP: bit điều khiển tốc độ quay.

SMP = 1: không cho phép điều khiển tốc độ ở chế độ tốc độ tiêu chuẩn 100KHz và 1 MHz

SMP = 0: Cho phép điều khiển tốc độ ở chế độ high-speed 400 kHz.

CKE: bit lựa chọn SMBus

CKE = 1: cho phép.

CKE = 0: không cho phép.

D/A: bit lựa chọn dữ liệu/địa chỉ.

P: bit dừng - stop bit.

P = 1: chỉ ra rằng stop bit được phát hiện.

P = 0: stop bit không được phát hiện.

S: bit bắt đầu.

S = 1: Start bit được phát hiện.

S = 0: Stop bit không được phát hiện.

R/W: bit xác định đọc/ghi.

UA: cập nhật địa chỉ trong chế độ slave 10 bit

UA = 1: chỉ ra người dùng cần cập nhật địa chỉ trong thanh ghi SSPADD

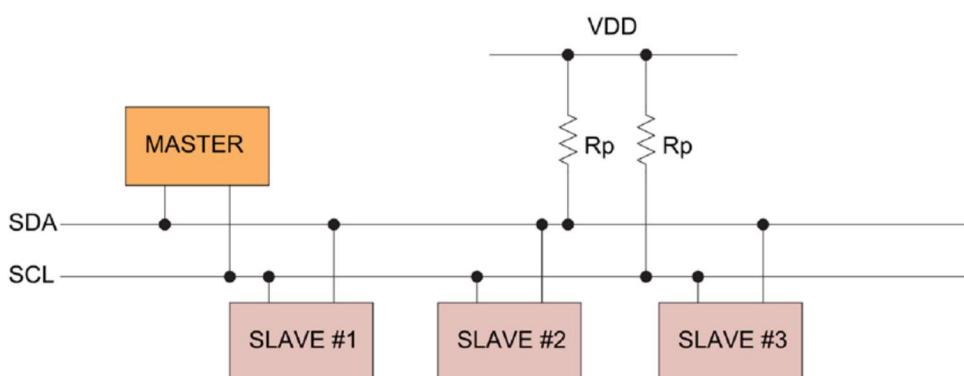
UA = 0: địa chỉ không cần cập nhật.

BF: bit chỉ trạng thái đệm đã đầy

Kết nối và nguyên lý hoạt động của giao thức I²C

Chuẩn giao thức I²C cũng có 2 chế độ là Master và slave được kết nối với ngắt.

Kết nối giữa các thiết bị dùng giao thức I²C được mô tả trên hình 4.25 trong đó sử dụng 2 chân RC3/SCL và RC4/SDA. Mỗi dây SDA hoặc SCL đều được nối với điện áp dương của nguồn thông qua một điện trở pull-up do các chân giao tiếp có dạng cực máng hở. Giá trị của điện trở này tùy thuộc từng thiết bị mà có giá trị từ 1k đến 4k7.



Hình 4.25: Khối giao thức I²C

Mỗi thiết bị được xác định bởi một địa chỉ duy nhất. I2C quy định hệ thống sẽ có 1 thiết bị Master và các thiết bị khác là Slave. Mỗi Slave được phân biệt với nhau bởi địa chỉ riêng. Xung nhịp SCL do Master làm chủ khi muốn giao tiếp với Slave. Khi đó, Master sẽ gửi byte địa chỉ kèm theo xung nhịp SCL; byte này khớp với Slave nào thì Slave đó sẽ phát ra một xung ACK (chấp nhận) để báo rằng nó có thể giao tiếp với Master. Việc truyền/nhận sẽ bắt đầu bởi Start bit và dừng bởi Stop bit.

Thanh ghi SSPBUF chứa dữ liệu sẽ được truyền đi hoặc nhận được và đóng vai trò là thanh ghi đệm cho thanh ghi dịch SSPSR. Thanh ghi SSPADD chứa địa chỉ của thiết bị ngoại vi cần truy xuất dữ liệu của I2C trong chế độ Slave. Khi hoạt động trong chế độ Master, thanh ghi này chứa giá trị tạo ra tốc độ baud cho xung clock dùng để truyền nhận dữ liệu. Trong quá trình truyền/nhận dữ liệu, sau khi nhận được 1 byte dữ liệu hoàn chỉnh, thanh ghi SSPIR sẽ chuyển dữ liệu vào thanh ghi SSPBUF. Thanh ghi SSPIR không đọc/ghi được trực tiếp. Quá trình truy xuất phải được thực hiện thông qua thanh ghi SSPBUF. Trong quá trình truyền dữ liệu, dữ liệu cần truyền khi đưa vào thanh ghi SSPBUF sẽ được đồng thời đưa vào thanh ghi SSPIR.

Chế độ Slave

Để vi điều khiển hoạt động trong chế độ Slave, chân SDA và SCL (TRISC<4:3>) cần được xác lập là input. Khi đó, vi điều khiển Slave sẽ được điều khiển bởi một thiết bị khác thông qua địa chỉ. Khi địa chỉ truyền đến Slave là trùng khớp hoặc khi dữ liệu đã được nhận xong, vi điều khiển sẽ tạo ra xung \overline{ACK} để báo hiệu kết thúc truyền/nhận; giá trị trong thanh ghi SSPIR sẽ được đưa vào thanh ghi SSPBUF. Tuy nhiên, xung \overline{ACK} sẽ không được tạo ra nếu trong một các trường hợp sau xảy ra:

(1) bit BF (SSPSTAT<0>) báo hiệu bộ đệm buffer đầy được set trước khi quá trình truyền nhận xảy ra.

(2) bit SSPOV (SSPCON<6>) được set trước khi quá trình truyền nhận xảy ra (SSPOV được set trong trường hợp khi 1 byte khác được nhận vào trong khi dữ liệu trong SSPBUF trước đó vẫn chưa được lấy ra)

Trong các trường hợp trên, thanh ghi SSPIR sẽ không đưa giá trị vào thanh ghi SSPBUF, nhưng bit SSPIF(PIR<3>) sẽ được set. Để quá trình truyền/nhận dữ liệu được tiếp tục, người dùng cần đọc dữ liệu từ thanh ghi SSPBUF vào trước. Khi đó, bit BF sẽ tự động xóa còn bit SSPOV phải được xóa bằng chương trình. Khi MSSP được kích hoạt, nó sẽ chờ tín hiệu để bắt đầu hoạt động. Khi

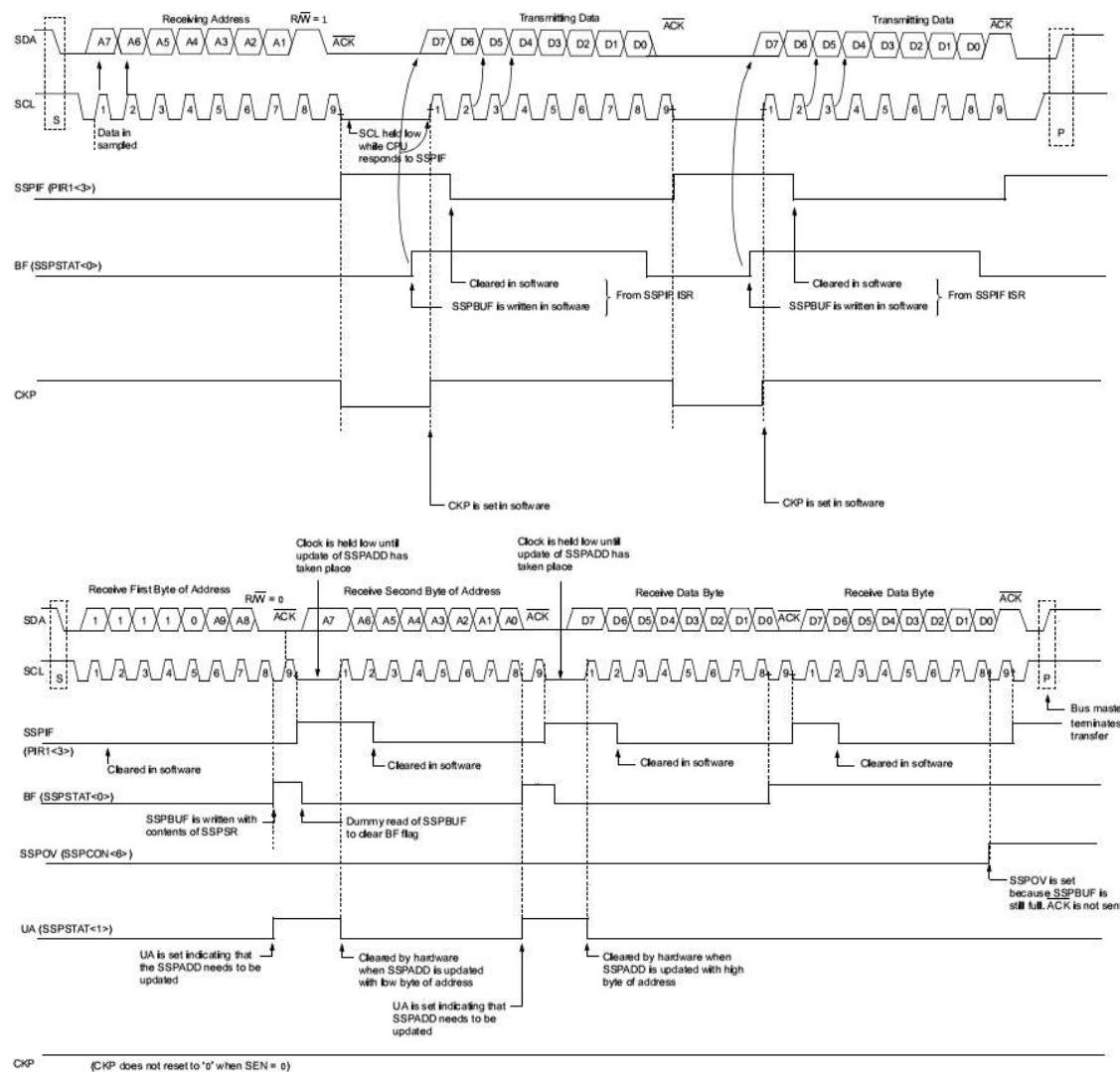
nhận được tín hiệu bắt đầu hoạt động(sườn xuống đầu tiên của SDA), dữ liệu 8 bit sẽ được dịch vào thanh ghi SSPSR. Các bit đưa vào sẽ được lấy mẫu tại sườn lên của xung clock. Giá trị nhận được từ thanh ghi SSPSR sẽ được so sánh với giá trị trong thanh ghi SSPADD tại sườn xuống của xung clock thứ 8. Nếu kết quả so sánh bằng nhau, tức là I2C master chỉ định đối tượng giao tiếp là vi điều khiển đang ở chế độ Slave, bit BF và SSPOV sẽ được xóa về 0 và gây ra tác động sau:

1. Giá trị trong thanh ghi SSPSR được đưa vào thanh ghi SSPBUF.
2. bit BF được set.
3. Một xung được tạo ra.
4. Cờ ngắt SSPIF được set tại cạnh xuống thứ 9 của xung clock.

Trường hợp chế độ 10 bit địa chỉ, vi điều khiển cần nhận vào 10 bit địa chỉ để so sánh. Do đó, bit SSPSTAT<2> cần được xóa về 0 để cho phép nhận 2 byte địa chỉ. Byte đầu tiên có định dạng "11110 A9 A8 0". Trong đó A9 và A8 là 2 bit MSB của 10 bit địa chỉ. Byte thứ 2 là 8 bit địa chỉ còn lại. Quá trình nhận dạng 10 bit địa chỉ diễn ra như sau, trong đó các bước 7-9 xảy ra trong quá trình truyền:

1. Đầu tiên, 2 bit MSB của 10 bit địa chỉ được nhận trước. Bit SSPIF, BF, và UA (SSPSTAT<1>) được set. (Byte đầu tiên có định dạng "11110 A9 A8 0")
2. Cập nhật vào 8 bit địa chỉ thấp của thanh ghi SSPADD, bit UA sẽ được xóa bởi vi điều khiển để khởi tạo xung clock ở chân SCL sau khi quá trình cập nhật hoàn tất.
3. Đọc quá trị thanh ghi SSPBUF (bit BF được xóa) và cờ ngắt SSPIF được xóa.
4. Nhận 8 bit địa chỉ cao, bit SSPIF, BF và UA được set.
5. Cập nhật 8 bit địa chỉ đã nhận. Nếu địa chỉ là khớp (address match), xung clock ở chân SCL được khởi tạo và bit UA được set.
6. Đọc giá trị thanh ghi SSPBUF (bit BF được xóa) và xóa cờ ngắt SSPIF.
7. Nhận tín hiệu Start.
8. Nhận byte địa chỉ cao (bit SSPIF và BF được set).
9. Đọc giá trị trong thanh ghi SSPBUF (bit BF được xóa) và xóa cờ ngắt SSPIF.

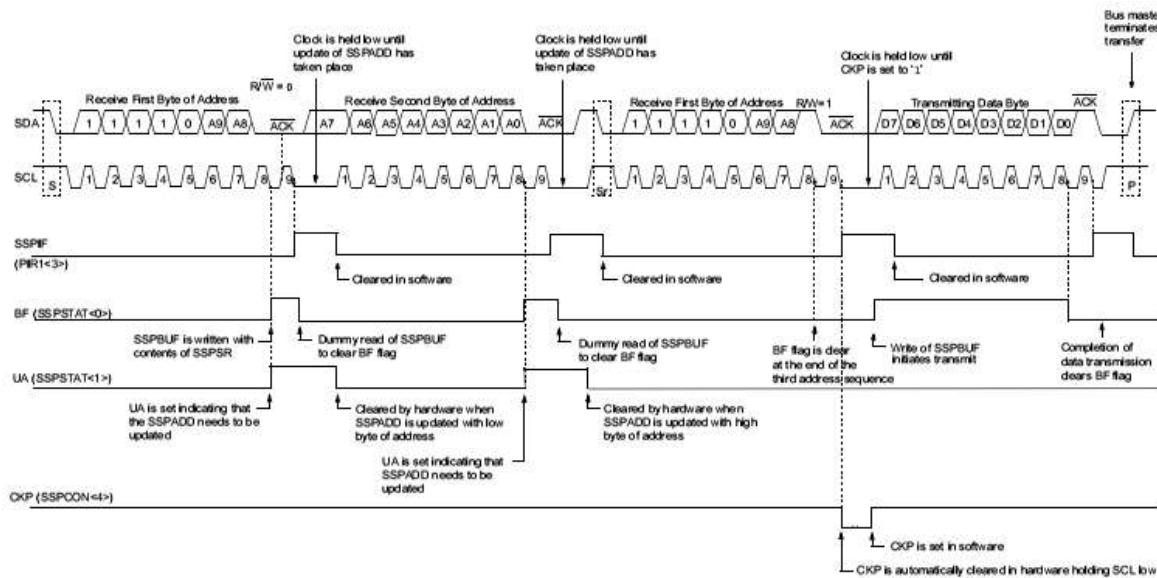
Quá trình nhận dữ liệu: Khi bit R/W của byte địa chỉ bị xóa và địa chỉ đúng (address match), bit R/W của thanh ghi SSPSTAT được xóa về 0. Địa chỉ nhận được nạp vào thanh ghi SSPBUF và đường SDA được giữ ở mức thấp (\overline{ACK}). Khi địa chỉ nhận tồn tại, tín hiệu \overline{ACK} được duy trì thêm 1 xung nhịp và dữ liệu sẽ được truyền ngay sau đó. Sau khi dữ liệu được truyền, trạng thái tràn được xác lập bởi bit BF và SSPOV được xác lập. Đồng thời, ngắt MSSP được thiết lập bởi bit SSPIF. Bit SSPIF chỉ được xóa bởi phần mềm. Nếu bit SEN tích cực (SSPCON $<0>$ =1), xung clcok RC3/SCK/SCL sẽ bị giữ ở mức thấp mỗi khi dữ



Hình 4.26: Quá trình nhận I²C

liệu truyền. Muốn khởi tạo lại xung clock, ta cần set bit CKP. Điều này làm cho hiện tượng tràn dữ liệu không xảy ra

Quá trình truyền dữ liệu: Khi bit R/\overline{W} của byte địa chỉ được set và địa chỉ đúng (address match), bit R/\overline{W} của thanh ghi SSPSTAT được set lên 1. Địa chỉ nhận được nạp vào thanh ghi SSPBUF. Sau đó, xung \overline{ACK} được tạo ra, xung clock ở chân RC3/SCK/SCL được đưa xuống mức thấp bắt chấp trạng thái của bit SEN. Khi đó I2C Master sẽ không được đưa xung clock vào I2C Slave cho đến khi dữ liệu ở thanh ghi SSPSR ở trạng thái sẵn sàng cho quá trình truyền dữ liệu (dữ liệu đưa vào thanh ghi SSPBUF sẽ đồng thời được đưa vào thanh ghi SSPSR). Tiếp theo cho phép xung ở pin RC3/SCK/SCL bằng cách set bit CKP (SSPCON<4>). Từng bit của byte dữ liệu sẽ được dịch ra ngoài tại mỗi cạnh xuống của xung clock. Như vậy dữ liệu sẽ sẵn ở ngõ ra khi xung clock ở mức logic cao, giúp cho I2C Master nhận được dữ liệu tại mỗi cạnh lên của xung clock. Tại cạnh lên xung clock thứ 9, dữ liệu đã được dịch hoàn toàn vào I2C Master, xung sẽ được tạo ra ở I2C Master, đồng thời pin SDA sẽ được giữ ở mức logic cao. Trong trường hợp xung \overline{ACK} được chốt bởi I2C Slave, thanh ghi SSPSTAT sẽ được reset. I2C Slave sẽ chờ tín hiệu của bit Start để tiếp tục truyền byte dữ liệu tiếp theo (đưa byte dữ liệu tiếp theo vào thanh ghi SSPBUF) và set bit CKP. Ngắt MSSP xảy ra khi một byte dữ liệu kết thúc quá trình truyền, bit SSPIF được set tại cạnh xuống của xung clock thứ 9 và phải được xóa bằng chương



Hình 4.27: Quá trình truyền I2C

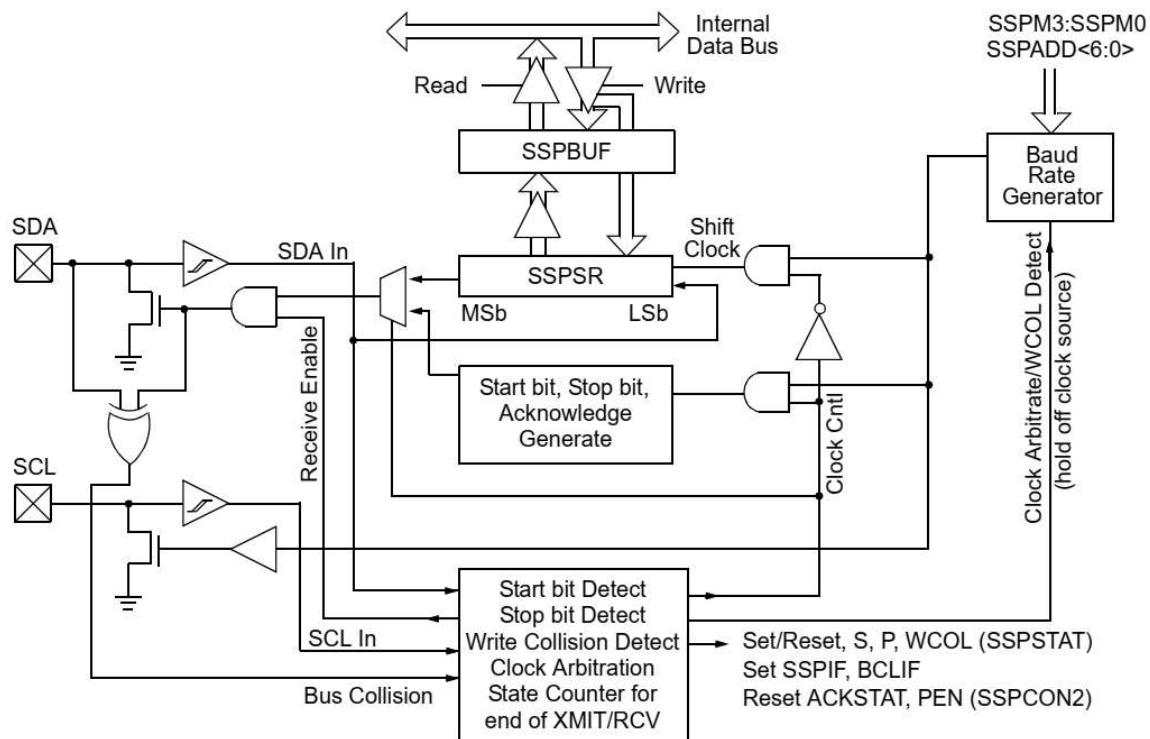
trình để đảm bảo sẽ được set khi byte dữ liệu tiếp theo truyền xong.

Chế độ Master

Chế độ I2C Master được xác lập bằng cách đưa các giá trị thích hợp vào các bit SSPM của thanh ghi SSPCON và set bit SSPEN. Ở chế độ Master, các chân SCK và SDA sẽ được điều khiển bởi phần cứng của MSSP. I2C Master đóng vai trò tích cực trong quá trình giao tiếp và điều khiển các I2C Slave thông qua việc chủ động tạo ra xung giao tiếp và các điều kiện Start, Stop khi truyền nhận dữ liệu. Một byte dữ liệu có thể được bắt đầu bằng điều kiện Start, kết thúc bằng điều kiện Stop hoặc bắt đầu và kết thúc với cùng một điều kiện khởi động lặp lại (Repeated Start Condition).

Hoạt động của I2C chế độ Master

Vì điều khiển Master tạo ra tất cả các xung clock và các điều kiện Start và Stop. Quá trình truyền kết thúc khi điều kiện Stop hoặc điều kiện Repeat Start



Hình 4.28: Sơ đồ khái niệm của I2C Master

xảy ra. Điều kiện Repeat Start xảy ra tương ứng với sự bắt đầu của việc truyền dữ liệu kế tiếp. Khi đó đường truyền I2C sẽ không được giải phóng.

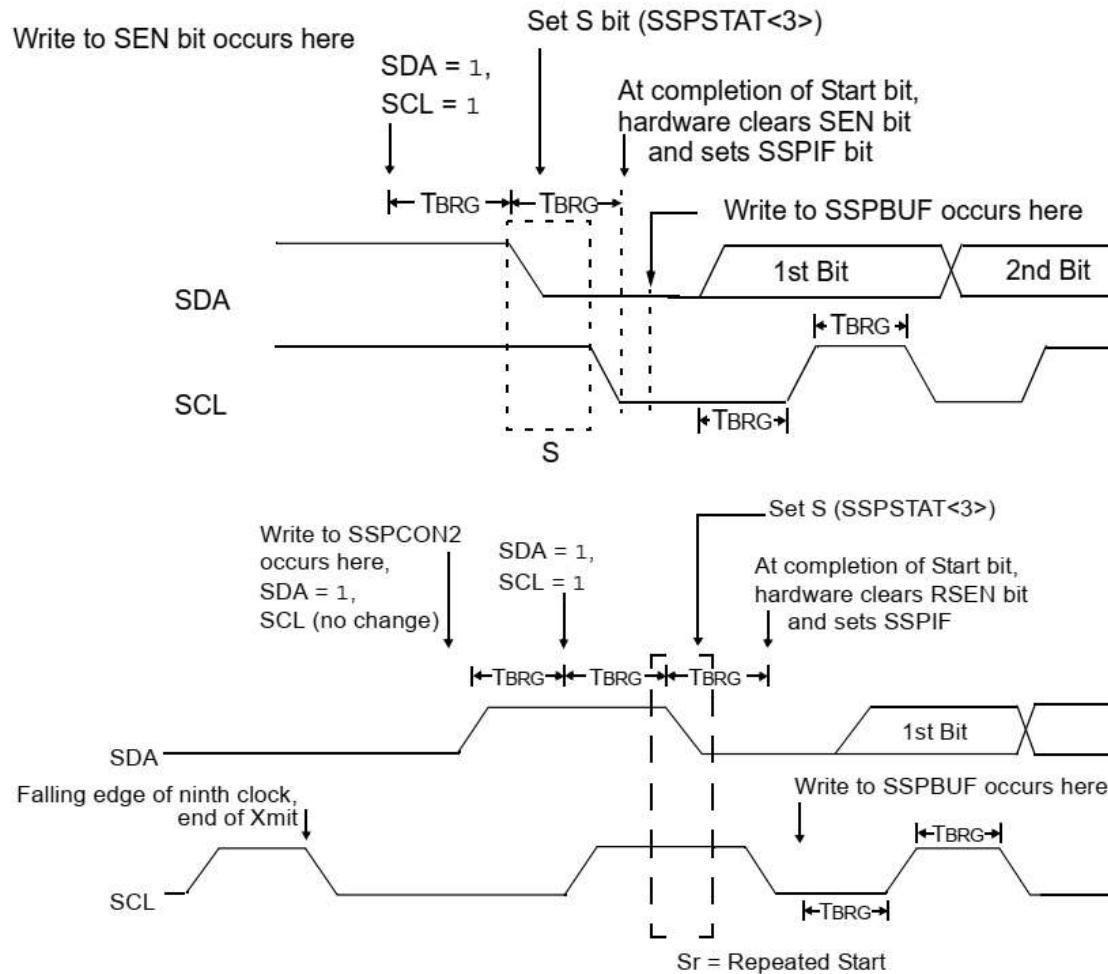
Trong chế độ truyền Master, dữ liệu được đưa ra trên chân SDA trong khi xung clock được đưa ra trên chân SCL. Byte truyền đầu tiên chứa địa chỉ của Slave (7 bit) và bit R/\overline{W} . Trong trường hợp này, bit $R/\overline{W} = 0$. Dữ liệu truyền đi sẽ là 8 bit liên tiếp. Sau mỗi byte truyền, bit \overline{ACK} sẽ được nhận. Các điều kiện Start và Stop là các điều kiện chỉ ra sự bắt đầu và kết thúc quá trình truyền. Trong trường hợp bit $R/\overline{W} = 1$, đây là quá trình nhận dữ liệu của vi điều khiển Master.

Khởi tạo xung nhịp

Xung nhịp clock được tạo ra trong chế độ Master được hình thành từ khối tạo xung nhịp (Baund rate generator), giá trị ấn định tần số xung clock nối tiếp được lấy từ 7 bit thấp của thanh ghi SSPADD. Khi dữ liệu được đưa vào thanh ghi SSPBUF, bit BF được set và BRG tự động đếm ngược về 0 và dừng lại, chân SCL được giữ nguyên trạng thái trước đó. Khi dữ liệu tiếp theo được đưa vào, BRG sẽ cần một khoảng thời gian TBRG tự động reset lại giá trị để tiếp tục quá trình đếm ngược. Mỗi vòng lệnh (có thời gian T_{CY}), BRG sẽ giảm giá trị 2 lần. Các giá trị cụ thể của tần số xung nối tiếp do BRG tạo ra được liệt kê trong bảng sau:

F_{CY}	$F_{CY} * 2$	Giá trị BRG	F_{SCL}
10 MHz	20 MHz	19h	400 Khz
10 MHz	20 MHz	20h	312.5 KHz
10 MHz	20 MHz	3Fh	100 KHz
4 MHz	8 MHz	0Ah	400 KHz
4 MHz	8 MHz	0Dh	308 KHz
4 MHz	8 MHz	28h	100 KHz
1 MHz	2 MHz	03h	333 KHz
1 MHz	2 MHz	0Ah	100 KHz
1 MHz	2 MHz	00h	1 MHz

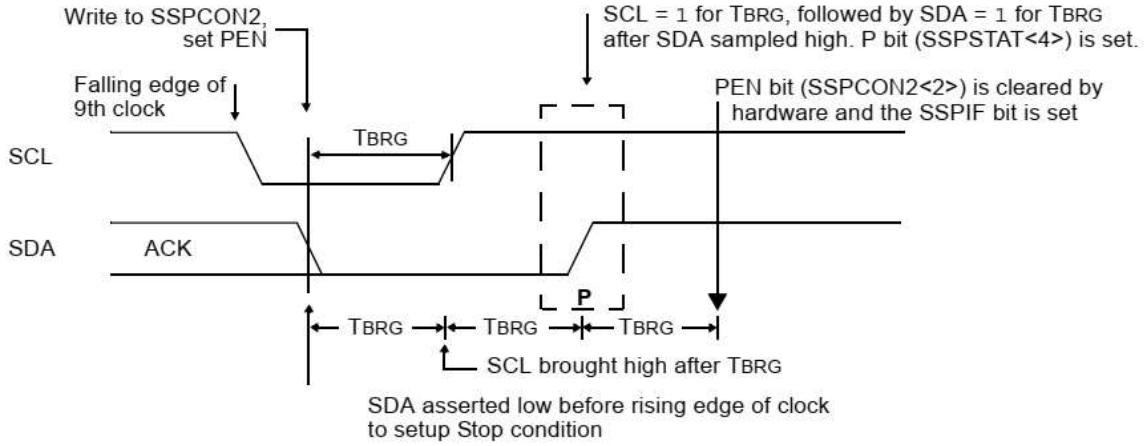
Trong đó giá trị BRG là giá trị được lấy từ 7 bit thấp của thanh ghi SSPADD. Do I2C ở chế độ Master mode, thanh ghi SSPADD sẽ không được sử dụng để chứa địa chỉ, thay vào đó chức năng của SSPADD là thanh ghi chứa giá trị của BRG. Để tạo được điều kiện Start, trước hết cần đưa hai pin SCL và SDA lên mức logic cao và bit SEN (SSPCON2<0>) phải được set. Khi đó BRG sẽ tự động đọc giá trị 7 bit thấp của thanh ghi SSPADD và bắt đầu đếm. Sau khoảng thời gian TBRG, pin SDA được đưa xuống mức logic thấp. Trạng thái pin SDA ở mức



Hình 4.29: Giản đồ thời gian I2C Master trong quá trình tạo điều kiện Start và Start liên tục

logic thấp và pin SCL ở mức logic cao chính là điều kiện Start của I2C Master mode. Khi đó bit S (SSPSTAT<3>) sẽ được set. Tiếp theo BRG tiếp tục lấy giá trị từ thanh ghi SSPADD để tiếp tục quá trình đếm, bit SEN được tự động xóa và cờ ngắt SSPIF được set. Trong trường hợp pin SCL và SDA ở trạng thái logic thấp, hoặc là trong quá trình tạo điều kiện Start, pin SCL được đưa về trạng thái logic thấp trước khi pin SDA được đưa về trạng thái logic thấp, điều kiện Start sẽ không được hình thành, cờ ngắt BCLIF sẽ được set và I2C sẽ ở trạng thái tạm ngưng hoạt động (Idle).

Tín hiệu Stop sẽ được đưa ra pin SDA khi kết thúc dữ liệu bằng cách set bit PEN (SSPCON2<2>). Sau cạnh xuống của xung clock thứ 9 và với tác động của



Hình 4.30: Giản đồ thời gian I₂C Master trong quá trình tạo điều kiện Stop

bit điều khiển PEN, pin SDA cũng được đưa xuống mức thấp, BRG lại bắt đầu quá trình đếm. Sau một khoảng thời gian TBRG, pin SCL được đưa lên mức logic cao và sau một khoảng thời gian TBRG nữa pin SDA cũng được đưa lên mức cao. Ngay tại thời điểm đó bit P (SSPSTAT<4>) được set, nghĩa là điều kiện Stop đã được tạo ra. Sau một khoảng thời gian TBRG nữa, bit PEN tự động được xóa và cờ ngắt SSPIF được set.

Để tạo được điều kiện Start lặp lại liên tục trong quá trình truyền dữ liệu, trước hết cần set bit RSEN (SSPCON2<1>). Sau khi set bit RSEN, pin SCL được đưa xuống mức logic thấp, pin SDA được đưa lên mức logic cao, BRG lấy giá trị từ thanh ghi SSPADD vào để bắt đầu quá trình đếm. Sau khoảng thời gian TBRG, pin SCL cũng được đưa lên mức logic cao trong khoảng thời gian TBRG tiếp theo. Trong khoảng thời gian TBRG kế tiếp, pin SDA lại được đưa xuống mức logic thấp trong khi SCL vẫn được giữ ở mức logic cao. Ngay thời điểm đó bit S (SSPSTAT<3>) được set để báo hiệu điều kiện Start được hình thành, bit RSEN tự động được xóa và cờ ngắt SSPIF sẽ được set sau một khoảng thời gian TBRG nữa. Lúc này địa chỉ của I₂C Slave có thể được đưa vào thanh ghi SSPBUF, sau đó ta chỉ việc đưa tiếp địa chỉ hoặc dữ liệu tiếp theo vào thanh ghi SSPBUF mỗi khi nhận được tín hiệu \overline{ACK} từ I₂C Slave, I₂C Master sẽ tự động tạo tín hiệu Start lặp lại liên tục cho quá trình truyền dữ liệu liên tục. Cần chú ý là bất cứ một trinh tự nào sai trong quá trình tạo điều kiện Start lặp lại sẽ làm cho bit BCLIF được set và I₂C được đưa về trạng thái “Idle”.

Xét quá trình truyền dữ liệu, xung clock sẽ được đưa ra từ pin SCL và dữ liệu được đưa ra từ pin SDA. Byte dữ liệu đầu tiên phải là byte địa chỉ xác định I₂C

Slave cần giao tiếp và bit (trong trường hợp này = 0). Đầu tiên các giá trị địa chỉ sẽ được đưa vào thanh ghi SSPBUF, bit BF tự động được set lên 1 và bộ đếm tạo xung clock nối tiếp BRG (Baud Rate Generator) bắt đầu hoạt động. Khi đó từng bit dữ liệu (hoặc địa chỉ và bit) sẽ được dịch ra ngoài theo từng cạnh xuống của xung clock sau khi cạnh xuống đầu tiên của pin SCL được nhận diện (điều kiện Start), BRG bắt đầu đếm ngược về 0. Khi tất cả các bit của byte dữ liệu được đã được đưa ra ngoài, bộ đếm BRG mang giá trị 0. Sau đó, tại cạnh xuống của xung clock thứ 8, I2C Master sẽ ngưng tác động lên pin SDA để chờ đợi tín hiệu từ I2C Slave (tín hiệu xung). Tại cạnh xuống của xung clock thứ 9, I2C Master sẽ lấy mẫu tín hiệu từ pin SDA để kiểm tra xem địa chỉ đã được I2C Slave nhận dạng chưa, trạng thái được đưa vào bit ACKSTAT (SSPCON2<6>). Cũng tại thời điểm cạnh xuống của xung clock thứ 9, bit BF được tự động clear, cờ ngắt SSPIF được set và BRG tạm ngưng hoạt động cho tới khi dữ liệu hoặc địa chỉ tiếp theo được đưa vào thanh ghi SSPBUF, dữ liệu hoặc địa chỉ sẽ tiếp tục được truyền đi tại cạnh xuống của xung clock tiếp theo.

Xét quá trình nhận dữ liệu ở chế độ I2C Master mode. Trước tiên ta cần set bit cho phép nhận dữ liệu RCEN (SSPCON2<3>). Khi đó BRG bắt đầu quá trình đếm, dữ liệu sẽ được dịch vào I2C Master qua pin SDA tại cạnh xuống của pin SCL. Tại cạnh xuống của xung clock thứ 8, bit cờ hiệu cho phép nhận RCEN tự động được xóa, dữ liệu trong thanh ghi SSPSR được đưa vào thanh ghi SSPBUF, cờ hiệu BF được set, cờ ngắt SSPIF được set, BRG ngưng đếm và pin SCL được đưa về mức logic thấp. Khi đó MSSP ở trạng thái tạm ngưng hoạt động để chờ đợi lệnh tiếp theo. Sau khi đọc giá trị thanh ghi SSPBUF, cờ hiệu BF tự động được xóa. Ta còn có thể gửi tín hiệu \overline{ACK} bằng cách set bit ACKEN (SSPCON2<4>).

Giao thức USART

USART (Universal Synchronous Asynchronous Receiver Transmitter) là một trong hai chuẩn giao tiếp nối tiếp. USART còn được gọi là giao diện giao tiếp nối tiếp nối tiếp SCI (Serial Communication Interface). Người dùng có thể sử dụng giao thức này để giao tiếp với các thiết bị ngoại vi, với các vi điều khiển khác hoặc với máy tính. Các dạng của giao diện USART ngoại vi bao gồm:

- Không đồng bộ (Asynchronous - full duplex)
- Đồng bộ master (half-duplex)
- Đồng bộ slave (half-duplex)

Hai pin dùng cho giao diện này là RC6/TX/CK và RC7/RX/DT, trong đó

RC6/TX/CK dùng để truyền xung clock (baud rate) và RC7/RX/DT dùng để truyền data. Trong trường hợp này ta phải set bit TRISC<7:6> và SPEN (RC-STA<7>) để cho phép giao diện USART.

Hai thanh ghi điều khiển quá trình truyền/nhận là TXSTA và RCSTA
Thanh ghi TXSTA

TXSTA: TRANSMIT STATUS AND CONTROL REGISTER (ADDRESS 98h)

R/W-0	R/W-0	R/W-0	R/W-0	U-0	R/W-0	R-1	R/W-0
CSRC	TX9	TXEN	SYNC	—	BRGH	TRMT	TX9D

bit 7

bit 0

CSRC: bit chọn nguồn clock.

Chế độ không đồng bộ: không cần quan tâm.

Chế độ đồng bộ:

CSRC = 1 Chế độ Master, xung clock lấy từ bộ BRG.

CSRC = 0 Chế độ Slave, xung clock lấy từ bên ngoài.

TX9: bit cho phép truyền 9 bit

TX9 = 1 truyền dữ liệu 9 bit.

TX9 = 0 truyền dữ liệu 8 bit.

TXEN: bit cho phép truyền

TXEN = 1 cho phép truyền.

TXEN = 0 không cho phép truyền.

SYNC: bit chọn chế độ truyền

SYNC = 1 truyền đồng bộ.

SYNC = 0 truyền không đồng bộ.

BRGH: bit chọn truyền (không đồng bộ) tốc độ cao

BRGH = 1 tốc độ cao.

BRGH = 0 tốc độ thấp.

TRMT: bit trạng thái thanh ghi dịch-truyền

TRMT = 1 thanh ghi TSR không có dữ liệu.

TRMT = 0 thanh ghi TSR có dữ liệu.

TX9D: bit chứa dữ liệu thứ 9 khi truyền 9 bit.

Thanh ghi RCSTA

RCSTA: RECEIVE STATUS AND CONTROL REGISTER (ADDRESS 18h)

R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R-0	R-0	R-x
SPEN	RX9	SREN	CREN	ADDEN	FERR	OERR	RX9D
bit 7						bit 0	

SPEN: bit cho phép cổng nối tiếp.

SPEN = 1 cho phép cổng USART (chân RC7/RX/DT và RC6/TX/CK).

SPEN = 0 không cho phép giao tiếp USART

RX9: bit cho phép nhận dữ liệu 9 bit.

RX9 = 1 cho phép nhận dữ liệu 9 bit.

RX9 = 0 nhận dữ liệu 8 bit.

SREN: bit cho phép nhận dữ liệu 1 byte đơn.

Chế độ USART không đồng bộ: không cần quan tâm.

Chế độ USART Master đồng bộ:

SREN = 1 cho phép nhận dữ liệu 1 byte (8bit hoặc 9 bit).

SREN = 0 Không cho phép nhân dữ liệu 1 byte.

CREN: bit cho phép nhận dữ liệu liên tiếp.

Chế độ USART không đồng bộ:

CREN = 1 cho phép nhân chuỗi dữ liệu liên tục.

CREN = 0 Không cho phép nhân chuỗi dữ liệu

Ché đô đồng bô:

CREN = 1 cho phép nhân dữ liệu cho tới khi xóa bit CREN

CREN = 0 Không cho phép nhân chuỗi dữ liệu.

ADDEN: Bit cho phép xác nhận địa chỉ. Chế độ USART không đồng bộ 9 bit

ADDEN = 1 cho phép xác nhận địa chỉ. Khi RSR<8> =1 thì ngắt được cho phép thực thi và giá trị trong buffer được nhận vào.

$\text{ADDEN} = 0$ không cho phép xác nhận địa chỉ, các byte dữ liệu được nhận vào và bit thứ 9 có thể được sử dụng như là bit chẵn lẻ parity.

FERR: Bit xác định có lỗi framing

FERR = 1 có lỗi Framing trong quá trình truyền/nhận

FERR = 0 không có lỗi.

OERR: Bit xác định có lỗi Overrun

OERR = 1 có lỗi Overrun

OERR = 0 không có lỗi.

RX9D: Bit chứa dữ liệu thứ 9 của dữ liệu truyền/nhận

Bộ xác định tốc độ truyền USART

Bộ xác định tốc độ truyền BRG (Baud Rate Generator) là bộ xác lập chế độ truyền 8 bit hỗ trợ cả chế độ truyền USART đồng bộ và không đồng bộ. Thanh ghi SPBRG điều khiển chu kỳ của một bộ định thời 8 bit. Trong chế độ không đồng bộ, bit BRGH (TXSTA<2>) cũng điều khiển tốc độ truyền. Trong chế độ truyền đồng bộ, bit BRGH không được sử dụng. Trong đó X là giá trị trong thanh ghi SPBRG, có giá trị từ 0 đến 255

SYNC	BRGH = 0 (Tốc độ chậm)	BRGH = 1 (Tốc độ cao)
0	Không đồng bộ Tốc độ truyền = $f_{thạch anh}/64(X + 1)$	Tốc độ truyền = $f_{thạch anh}/16(X + 1)$
1	Đồng bộ Tốc độ truyền = $f_{thạch anh}/4(X + 1)$	N/A

Tốc độ truyền trong chế độ không đồng bộ với BRGH = 0

Baud Rate (K)	$F_{OSC} = 20MHz$			$F_{OSC} = 16MHz$		
	KBaud	% sai lệch	SPBRG	KBaud	% sai lệch	SPBRG
0.3	-	-	-	-	-	-
1.2	1.221	1.75	255	1.202	0.17	129
2.4	2.404	0.17	129	2.404	0.17	64
9.6	9.766	1.73	31	9.615	1.73	15
19.2	19.531	1.72	15	19.231	1.72	7
28.8	31.250	8.51	9	27.778	8.51	4
33.6	34.722	3.34	8	35.714	6.99	4
57.6	62.500	8.51	4	62.500	9.58	2
HIGH	1.221	-	255	0.977	-	255
LOW	312.500	-	0	250.000	-	0

Tốc độ truyền trong chế độ không đồng bộ với BRGH = 0

Baud Rate (K)	$F_{OSC} = 10MHz$			$F_{OSC} = 3.6864MHz$		
	KBaud	% sai lệch	SPBRG	KBaud	% sai lệch	SPBRG
0.3	-	-	-	0.3	0	191
1.2	1.221	0.17	129	1.2	0	47
2.4	2.404	0.17	64	2.4	0	23
9.6	9.766	1.73	15	9.6	0	5
19.2	19.531	1.72	7	19.2	0	2
28.8	31.250	8.51	4	28.8	0	1
33.6	31.250	6.99	4	-	-	-
57.6	52.083	9.58	2	57.6	0	0
HIGH	0.610	-	255	0.225	-	255
LOW	156.250	-	0	57.6	-	0

Tốc độ truyền trong chế độ không đồng bộ với BRGH = 1

Baud Rate (K)	$F_{OSC} = 20MHz$			$F_{OSC} = 16MHz$		
	KBaud	% sai lệch	SPBRG	KBaud	% sai lệch	SPBRG
0.3	-	-	-	-	-	-
1.2	-	-	-	-	-	-
2.4	-	-	-	-	-	-
9.6	9.615	0.16	129	9.615	0.16	103
19.2	19.231	0.16	64	19.231	0.16	51
28.8	29.070	0.94	42	29.412	2.13	33
33.6	33.784	0.55	36	33.333	0.79	29
57.6	59.524	3.34	20	58.824	2.13	16
HIGH	4.883	-	255	3.906	-	255
LOW	1250.000	-	0	1000.000	-	0

USART không đồng bộ

Trong chế độ truyền không đồng bộ, khối chức năng USART sử dụng cấu hình giao thức NRZ (Non-Return-to-Zero), nghĩa là các bit truyền đi sẽ bao gồm 1 bit Start, 8 hay 9 bit dữ liệu (thông thường là 8 bit) và 1 bit Stop. Bit LSB sẽ được

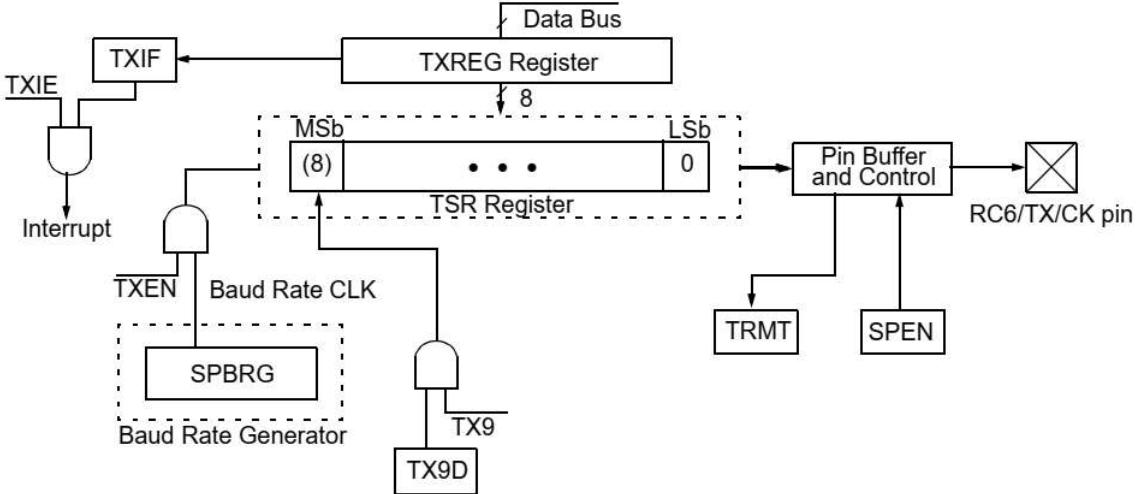
Tốc độ truyền trong chế độ không đồng bộ với BRGH = 1

Baud Rate (K)	$F_{OSC} = 10MHz$			$F_{OSC} = 3.6864MHz$		
	KBaud	% sai lệch	SPBRG	KBaud	% sai lệch	SPBRG
0.3	-	-	-	-	-	-
1.2	-	-	-	1.2	0	191
2.4	2.441	1.71	255	2.4	0	95
9.6	9.615	0.16	64	9.6	0	23
19.2	19.531	1.72	31	19.2	0	11
28.8	28.409	1.36	21	28.8	0	7
33.6	32.895	2.10	18	32.9	2.04	6
57.6	56.818	1.36	10	57.6	0	3
HIGH	2.441	-	255	0.9	-	255
LOW	625.000	-	0	230.4	-	0

truyền đi trước. Các khối truyền và nhận data độc lập với nhau sẽ dùng chung tần số tương ứng với tốc độ baud cho quá trình dịch dữ liệu (tốc độ baud gấp 16 hay 64 lần tốc độ dịch dữ liệu tùy theo giá trị của bit BRGH), và để đảm bảo tính hiệu quả của dữ liệu thì hai khối truyền và nhận phải dùng chung một định dạng dữ liệu. Chế độ truyền không đồng bộ được thực hiện bởi việc xóa bit SYNC (TXTSTA<4>). Khi cài đặt giao tiếp không đồng bộ cần chú ý những vấn đề sau:

- Tốc độ truyền BRG.
- Mạch trích mẫu.
- Khối truyền không đồng bộ.
- Khối nhận không đồng bộ

Khối truyền USART không đồng bộ: Thành phần quan trọng nhất của khối truyền dữ liệu là thanh ghi dịch dữ liệu TSR (Transmit Shift Register). Thanh ghi TSR sẽ lấy dữ liệu từ thanh ghi đệm dùng cho quá trình truyền dữ liệu TXREG. Dữ liệu cần truyền phải được đưa trước vào thanh ghi TXREG. Ngay sau khi bit Stop của dữ liệu cần truyền trước đó được truyền xong, dữ liệu từ thanh ghi TXREG sẽ được đưa vào thanh ghi TSR, thanh ghi TXREG bị rỗng, ngắt xảy ra và cờ hiệu TXIF (PIR1<4>) được set. Ngắt này được điều khiển bởi bit TXIE (PIE1<4>). Cờ hiệu TXIF vẫn được set bất chấp trạng thái của bit TXIE hay tác động của chương trình (không thể xóa TXIF bằng chương



Hình 4.31: Khối truyền USART không đồng bộ

trình) mà chỉ reset về 0 khi có dữ liệu mới được đưa vào thanhh ghi TXREG.

Trong khi cờ hiệu TXIF đóng vai trò chỉ thị trạng thái thanh ghi TXREG thì cờ hiệu TRMT (TXSTA<1>) có nhiệm vụ thể hiện trạng thái thanh ghi TSR. Khi thanh ghi TSR rỗng, bit TRMT sẽ được set. Bit này chỉ đọc và không có ngắt nào được gắn với trạng thái của nó. Một điểm cần chú ý nữa là thanh ghi TSR không có trong bộ nhớ dữ liệu và chỉ được điều khiển bởi CPU.

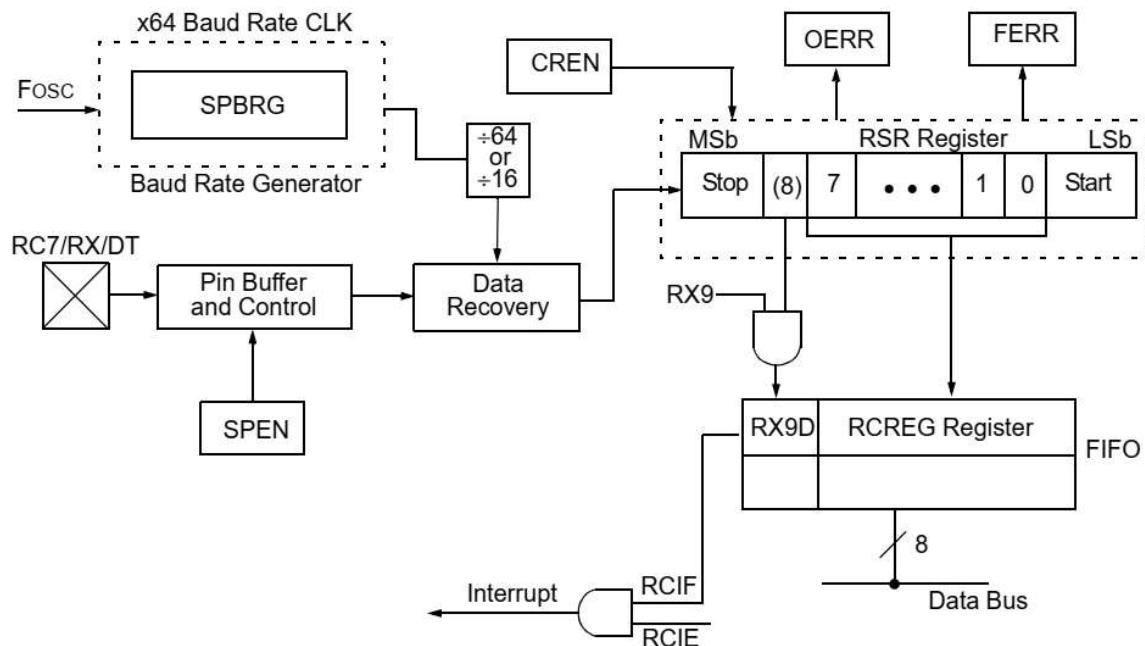
Khối truyền dữ liệu được cho phép hoạt động khi bit TXEN (TXSTA<5>) được set. Quá trình truyền dữ liệu chỉ thực sự bắt đầu khi đã có dữ liệu trong thanh ghi TXREG và xung truyền baud được tạo ra. Khi khởi truyền dữ liệu được khởi động lần đầu tiên, thanh ghi TSR rỗng. Tại thời điểm đó, dữ liệu đưa vào thanh ghi TXREG ngay lập tức được load vào thanh ghi TSR và thanh ghi TXREG bị rỗng. Lúc này ta có thể hình thành một chuỗi dữ liệu liên tục cho quá trình truyền dữ liệu. Trong quá trình truyền dữ liệu nếu bit TXEN bị reset về 0, quá trình truyền kết thúc, khối truyền dữ liệu được reset và pin RC6/TX/CK chuyển đến trạng thái high-impedance. Trong trường hợp dữ liệu cần truyền là 9 bit, bit TX9 (TXSTA<6>) được set và bit dữ liệu thứ 9 sẽ được lưu trong bit TX9D (TXSTA<0>). Nên ghi bit dữ liệu thứ 9 vào trước, vì khi ghi 8 bit dữ liệu vào thanh ghi TXREG trước có thể xảy ra trường hợp nội dung thanh ghi TXREG sẽ được load vào thanh ghi TSG trước, như vậy dữ liệu truyền đi sẽ bị sai khác so với yêu cầu. Tóm lại, để truyền dữ liệu theo giao diện USART bất đồng bộ, ta cần thực hiện tuần tự các bước sau:

1. Tạo xung truyền baud bằng cách đưa các giá trị cần thiết vào thanh

ghi RSBRG và bit điều khiển mức tốc độ baud BRGH.

2. Cho phép cổng giao diện nối tiếp nối tiếp bắt đồng bộ bằng cách clear bit SYNC và set bit PSEN.
3. Set bit TXIE nếu cần sử dụng ngắt truyền.
4. Set bit TX9 nếu định dạng dữ liệu cần truyền là 9 bit.
5. Set bit TXEN để cho phép truyền dữ liệu (lúc này bit TXIF cũng sẽ được set).
6. Nếu định dạng dữ liệu là 9 bit, đưa bit dữ liệu thứ 9 vào bit TX9D.
7. Đưa 8 bit dữ liệu cần truyền vào thanh ghi TXREG.
8. Nếu sử dụng ngắt truyền, cần kiểm tra lại các bit GIE và PEIE (thanh ghi INTCON).

Khối nhận USART không đồng bộ: Dữ liệu được đưa vào từ chân RC7/RX/DT sẽ kích hoạt khối phục hồi dữ liệu. Khối phục hồi dữ liệu thực chất là một bộ dịch dữ liệu có tốc độ cao và có tần số hoạt động gấp 16 lần hoặc 64 lần tần số baud. Trong khi đó tốc độ dịch của thanh nhận dữ liệu sẽ bằng với tần số baud hoặc tần số của bộ tạo dao động.



Hình 4.32: Khối nhận USART không đồng bộ

Bit điều khiển cho phép khôi nhận dữ liệu là bit RCEN (RCSTA<4>). Thành phần quan trọng nhất của khôi nhận dữ liệu là thsnh ghi nhận dữ liệu RSR (Receive Shift Register). Sau khi nhận diện bit Stop của dữ liệu truyền tới, dữ liệu nhận được trong thanh ghi RSR sẽ được đưa vào thanh ghi RCGER, sau đó cờ hiệu RCIF (PIR1<5>) sẽ được set và ngắt nhận được kích hoạt. Ngắt này được điều khiển bởi bit RCIE (PIE1<5>). Bit cờ hiệu RCIF là bit chỉ đọc và không thể được tác động bởi chương trình. RCIF chỉ reset về 0 khi dữ liệu nhận vào ở thanh ghi RCREG đã được đọc và khi đó thanh ghi RCREG rỗng. Thanh ghi RCREG là thanh ghi có bộ đệm kép (double-buffered register) và hoạt động theo cơ chế FIFO (First In First Out) cho phép nhận 2 byte và byte thứ 3 tiếp tục được đưa vào thanh ghi RSR. Nếu sau khi nhận được bit Stop của byte dữ liệu thứ 3 mà thanh ghi RCREG vẫn còn đầy, cờ hiệu báo tràn dữ liệu (Overrun Error bit) OERR(RCSTA<1>) sẽ được set, dữ liệu trong thanh ghi RSR sẽ bị mất đi và quá trình đưa dữ liệu từ thanh ghi RSR vào thanh ghi RCREG sẽ bị gián đoạn.

Trong trường hợp này cần lấy hết dữ liệu ở thanh ghi RSREG vào trước khi tiếp tục nhận byte dữ liệu tiếp theo. Bit OERR phải được xóa bằng phần mềm và thực hiện bằng cách clear bit RCEN rồi set lại. Bit FERR (RCSTA<2>) sẽ được set khi phát hiện bit Stop dựa dữ liệu nhận vào. Bit dữ liệu thứ 9 sẽ được đưa vào bit RX9D (RCSTA<0>). Khi đọc dữ liệu từ thanh ghi RCREG, hai bit FERR và RX9D sẽ nhận các giá trị mới. Do đó cần đọc dữ liệu từ thanh ghi RCSTA trước khi đọc dữ liệu từ thanh ghi RCREG để tránh bị mất dữ liệu. Tóm lại, khi sử dụng giao diện nhận dữ liệu USART bắt đồng bộ cần tiến hành tuần tự các bước sau:

1. Thiết lập tốc độ baud (đưa giá trị thích hợp vào thanh ghi SPBRG và bit BRGH).
2. Cho phép cổng giao tiếp USART bắt đồng bộ (clear bit SYNC và set bit SPEN).
3. Nếu cần sử dụng ngắt nhận dữ liệu, set bit RCIE.
4. Nếu dữ liệu truyền nhận có định dạng là 9 bit, set bit RX9.
5. Cho phép nhận dữ liệu bằng cách set bit CREN.
6. Sau khi dữ liệu được nhận, bit RCIF sẽ được set và ngắt được kích hoạt (nếu bit RCIE được set).
7. Đọc giá trị thanh ghi RCSTA để đọc bit dữ liệu thứ 9 và kiểm tra xem quá trình nhận dữ liệu có bị lỗi không.

-
8. Đọc 8 bit dữ liệu từ thanh ghi RCREG.
 9. Nếu quá trình truyền nhận có lỗi xảy ra, xóa lỗi bằng cách xóa bit CREN.
 10. Nếu sử dụng ngắt nhận cần set bit GIE và PEIE (thanh ghi INTCON).

USART đồng bộ

Giao diện USART đồng bộ được kích hoạt bằng cách set bit SYNC. Cổng giao tiếp nối tiếp vẫn là hai chân RC7/RX/DT, RC6/TX/CK và được cho phép bằng cách set bit SPEN. USART cho phép hai chế độ truyền nhận dữ liệu là Master mode và Slave mode. Master mode được kích hoạt bằng cách set bit CSRC (TXSTA<7>), Slave mode được kích hoạt bằng cách clear bit CSRC. Điểm khác biệt duy nhất giữa hai chế độ này là Master mode sẽ lấy xung clock đồng bộ từ bộ tao xung baud BRG còn Slave mode lấy xung clock đồng bộ từ bên ngoài qua chân RC6/TX/CK. Điều này cho phép Slave mode hoạt động ngay cả khi vi điều khiển đang ở chế độ sleep.

Truyền USART đồng bộ chế độ Master

Tương tự như giao diện USART không đồng bộ, thành phần quan trọng nhất của hối truyền dữ liệu là thanh ghi dịch TSR (Transmit Shift Register). Thanh ghi này chỉ được điều khiển bởi CPU. Dữ liệu đưa vào thanh ghi TSR được chứa trong thanh ghi TXREG. Cờ hiệu của khối truyền dữ liệu là bit TXIF (chỉ thị trạng thái thanh ghi TXREG), cờ hiệu này được gắn với một ngắt và bit điều khiển ngắt này là TXIE. Cờ hiệu chỉ thị trạng thái thanh ghi TSR là bit TRMT. Bit TXEN cho phép hay không cho phép truyền dữ liệu. Các bước cần tiến hành khi truyền dữ liệu qua giao diện USART đồng bộ Master mode:

1. Tạo xung truyền baud bằng cách đưa các giá trị cần thiết vào thanh ghi RSBRG và bit điều khiển mức tốc độ baud BRGH.
2. Cho phép cổng giao diện nối tiếp nối tiếp đồng bộ bằng cách set bit SYNC, PSEN và CSRC.
3. Set bit TXIE nếu cần sử dụng ngắt truyền.
4. Set bit TX9 nếu định dạng dữ liệu cần truyền là 9 bit.
5. Set bit TXEN để cho phép truyền dữ liệu.
6. Nếu định dạng dữ liệu là 9 bit, đưa bit dữ liệu thứ 9 vào bit TX9D.
7. Đưa 8 bit dữ liệu cần truyền vào thanh ghi TXREG.

-
- Nếu sử dụng ngắt truyền, cần kiểm tra lại các bit GIE và PEIE (thanh ghi INTCON).

Nhận USART đồng bộ chế độ Master

Cấu trúc khối truyền dữ liệu là không đổi so với giao diện không đồng bộ, kể cả các cờ hiệu, ngắt nhận và các thao tác trên các thành phần đó. Điểm khác biệt duy nhất là giao diện này cho phép hai chế độ nhận dữ liệu, đó là chỉ nhận 1 word dữ liệu (set bit SCEN) hay nhận một chuỗi dữ liệu (set bit CREN) cho tới khi ta clear bit CREN. Nếu cả hai bit đều được set, bit điều khiển CREN sẽ được ưu tiên. Các bước cần tiến hành khi nhận dữ liệu bằng giao diện USART đồng bộ Master mode:

- Thiết lập tốc độ baud (đưa giá trị thích hợp vào thanh ghi SPBRG và bit BRGH).
- Cho phép cổng giao tiếp USART bắt đồng bộ (set bit SYNC, SPEN và CSRC).
- Clear bit CREN và SREN.
- Nếu cần sử dụng ngắt nhận dữ liệu, set bit RCIE.
- Nếu dữ liệu truyền nhận có định dạng là 9 bit, set bit RX9.
- Nếu chỉ nhận 1 word dữ liệu, set bit SREN, nếu nhận 1 chuỗi word dữ liệu, set bit CREN.
- Sau khi dữ liệu được nhận, bit RCIF sẽ được set và ngắt được kích hoạt (nếu bit RCIE được set).
- Đọc giá trị thanh ghi RCSTA để đọc bit dữ liệu thứ 9 và kiểm tra xem quá trình nhận dữ liệu có bị lỗi không.
- Đọc 8 bit dữ liệu từ thanh ghi RCREG.
- Nếu quá trình truyền nhận có lỗi xảy ra, xóa lỗi bằng cách xóa bit CREN.
- Nếu sử dụng ngắt nhận cần set bit GIE và PEIE (thanh ghi INTCON).

Truyền USART đồng bộ chế độ Slave

Quá trình này không có sự khác biệt so với chế độ Master khi vi điều khiển hoạt động ở chế độ bình thường. Tuy nhiên khi vi điều khiển đang ở trạng thái sleep, sự khác biệt được thể hiện rõ ràng. Nếu có hai word dữ liệu được đưa vào thanh ghi TXREG trước khi lệnh sleep được thực thi thì quá trình sau sẽ xảy ra:

-
- Word dữ liệu đầu tiên sẽ ngay lập tức được đưa vào thanh ghi TSR để truyền đi.
 - Word dữ liệu thứ hai vẫn nằm trong thanh ghi TXREG.
 - Cờ hiệu TXIF sẽ không được set.
 - Sau khi word dữ liệu đầu tiên đã dịch ra khỏi thanh ghi TSR, thanh ghi TXREG tiếp tục truyền word thứ hai vào thanh ghi TSR và cờ hiệu TXIF được set.
 - Nếu ngắt truyền được cho phép hoạt động, ngắt này sẽ đánh thức vi điều khiển và nếu toàn bộ các ngắt được cho phép hoạt động, bộ đếm chương trình sẽ chỉ tới địa chỉ chương trình ngắt (0004h).

Các bước cần tiến hành khi truyền dữ liệu bằng giao diện USART đồng bộ Slave mode:

1. Set bit SYNC, SPEN và clear bit CSRC.
2. Clear bit CREN và SREN.
3. Nếu cần sử dụng ngắt, set bit TXIE.
4. Nếu định dạng dữ liệu là 9 bit, set bit TX9.
5. Set bit TXEN.
6. Đưa bit dữ liệu thứ 9 vào bit TX9D trước (nếu định dạng dữ liệu là 9 bit).
7. Đưa 8 bit dữ liệu vào thanh ghi TXREG.
8. Nếu ngắt truyền được sử dụng, set bit GIE và PEIE (thanh ghi INTCON).

Nhận USART đồng bộ chế độ Slave

Sự khác biệt của Slave mode so với chế độ Master chỉ thể hiện rõ ràng khi vi điều khiển hoạt động ở chế độ sleep. Ngoài ra chế độ Slave mode không quan tâm tới bit SREN. Khi bit CREN (cho phép nhận chuỗi dữ liệu) được set trước khi lệnh sleep được thực thi, 1 word dữ liệu vẫn được tiếp tục nhận, sau khi nhận xong bit thanh ghi RSR sẽ chuyển dữ liệu vào thanh ghi RCREG và bit RCIF được set. Nếu bit RCIE (cho phép ngắt nhận) đã được set trước đó, ngắt sẽ được thực thi và vi điều khiển được “đánh thức”, bộ đếm chương trình sẽ chỉ đến địa chỉ 0004h và chương trình ngắt sẽ được thực thi.

Các bước cần tiến hành khi nhận dữ liệu bằng giao diện USART đồng bộ Slave mode:

1. Cho phép cổng giao tiếp USART bắt đồng bộ (set bit SYNC, SPEN

-
- clear bit CSRC).
2. Nếu cần sử dụng ngắt nhận dữ liệu, set bit RCIE.
 3. Nếu dữ liệu truyền nhận có định dạng là 9 bit, set bit RX9.
 4. Set bit CREN để cho phép quá trình nhận dữ liệu bắt đầu.
 5. Sau khi dữ liệu được nhận, bit RCIF sẽ được set và ngắt được kích hoạt (nếu bit RCIE được set).
 6. Đọc giá trị thanh ghi RCSTA để đọc bit dữ liệu thứ 9 và kiểm tra xem quá trình nhận dữ liệu có bị lỗi không.
 7. Đọc 8 bit dữ liệu từ thanh ghi RCREG.
 8. Nếu quá trình truyền nhận có lỗi xảy ra, xóa lỗi bằng cách xóa bit CREN.
 9. Nếu sử dụng ngắt nhận cần set bit GIE và PEIE (thanh ghi INTCON).

