

MỤC LỤC

LỜI NÓI ĐẦU	1
CHƯƠNG 1: TỔNG QUAN VỀ CÁC PHẦN TỬ	2
1.1. TỔNG QUAN VỀ PIC 16F877A.....	2
1.1.1. Sơ đồ khối và bảng mô tả chức năng các chân của PIC16F877A	3
1.1.2. Tổ chức bộ nhớ	10
1.1.2.1. Tổ chức của bộ nhớ chương trình	11
1.1.2.2. Tổ chức bộ nhớ dữ liệu	11
1.1.2.3. Các thanh ghi mục đích chung.....	11
1.1.2.4. Các thanh ghi chức năng đặc biệt	13
1.1.2.5. Các thanh ghi trạng thái	13
1.1.3. Các cổng của PIC 16F877A.....	14
1.1.3.1. PORTA và thanh ghi TRISA	14
1.1.3.2. PORTB và thanh ghi TRISB.....	15
1.1.3.3. PORTC và thanh ghi TRISC.....	17
1.1.3.4. PORTD và thanh ghi TRISD	20
1.1.3.5. PORTE và thanh ghi TRISE	20
1.1.4. Hoạt động của định thời	22
1.1.4.1. Bộ định thời TIMER0	22
1.1.4.2. Bộ định thời TIMER1	24
1.1.4.3. Bộ định thời TIMER2	25
1.2. THIẾT BỊ HIỂN THỊ LCD	27
1.2.1. Hình dáng kích thước.	27
1.2.2. Các chân chức năng.	28
1.2.3. Sơ đồ khối của HD44780.....	29
1.2.4. Tập lệnh của LCD.....	33
1.2.5. Đặc tính của các chân giao tiếp.	38
CHƯƠNG 2: THIẾT KẾ HỆ THỐNG ĐIỀU KHIỂN MOTOR.....	39
2.1. SƠ ĐỒ KHỐI.....	39
2.2. THIẾT KẾ CÁC KHỐI.....	39
2.2.1. Bộ xử lý.....	39
2.2.2. Khối hiển thị	40

2.2.3. <i>Ma trận phím</i> .	41
2.2.3.1 Cấu tạo .	41
2.2.3.2. nguyên lý quét phím.....	42
2.2.5. <i>Khối công suất động cơ</i> .	43
2.2.5.1. Motor DC	43
2.2.5.2. Mạch cầu H	43
2.2.5. <i>Khối nguồn</i>	44
2.3. SƠ ĐỒ MẠCH HỆ THỐNG	45
CHƯƠNG 3: PHẦN MỀM ĐIỀU KHIỂN	46
3.1. LƯU ĐỒ THUẬT TOÁN	46
3.2. CHƯƠNG TRÌNH.....	48
KẾT LUẬN	55
TÀI LIỆU THAM KHẢO	57

LỜI NÓI ĐẦU

Ngày nay, với những ứng dụng của khoa học kỹ thuật tiên tiến, thế giới của chúng ta đã và đang một ngày thay đổi, văn minh và hiện đại hơn. Sự phát triển của kỹ thuật điện tử đã tạo ra hàng loạt những thiết bị với các đặc điểm nổi bật như sự chính xác cao, tốc độ nhanh, gọn nhẹ là những yếu tố rất cần thiết góp phần cho hoạt động của con người đạt hiệu quả cao.

Điện tử đang trở thành một ngành khoa học đa nhiệm vụ. Điện tử đã đáp ứng được những đòi hỏi không ngừng từ các lĩnh vực Công – Nông – Lâm – Ngư nghiệp cho đến các nhu cầu cần thiết trong hoạt động đời sống hằng ngày.

Nhắc đến Điện tử không thể không nhắc đến Kỹ thuật vi điều khiển. Kỹ thuật vi điều khiển đã trở nên quen thuộc trong ngành kỹ thuật điện tử và cả trong các ứng dụng đời thường. Hầu hết các dây truyền tự động lớn và các sản phẩm dân dụng ta đều thấy sự xuất hiện của vi điều khiển. Vi điều khiển được nhà sản xuất tích hợp rất nhiều các nhiều tính năng với các bộ ngoại vi được tích hợp ngay trên vi điều khiển, cùng với khả năng xử lý nhiều hoạt động phức tạp, tất cả được tích hợp trên một con chip nhỏ gọn, chính vì vậy sẽ có nhiều thuận lợi hơn trong thiết kế board, khi đó board mạch sẽ nhỏ gọn và đẹp hơn dễ thiết kế hơn rất nhiều. Cùng với sự phát triển của khoa học kỹ thuật là sự phát triển của vi điều khiển và các ứng dụng của nó trong kỹ thuật. Chính vì vậy em đã lựa chọn đề tài: THẾT KẾ HỆ THỐNG MỞ CỬA TỰ ĐỘNG BẰNG MÃ, và vận dụng những kiến thức về vi điều khiển mà em đã được học để hoàn thành đề tài

Trong quá trình làm đồ án tốt nghiệp, do sự hạn chế về thời gian, tài liệu và trình độ có hạn nên không tránh khỏi có thiếu sót. Em rất mong được sự đóng góp ý kiến của thầy cô trong hội đồng và các bạn để đồ án tốt nghiệp của em được hoàn thiện hơn.

Chương 1: TỔNG QUAN VỀ CÁC PHẦN TỬ

1.1. TỔNG QUAN VỀ PIC 16F877A

Thông thường có 4 họ vi điều khiển 8 bit chính là 6811 của Motorola, 8051 của Intel, z8 của Xilog và Pic 16 của Microchip Technology. Mỗi một loại trên đây đều có một tập lệnh và thanh ghi riêng duy nhất, nên chúng thường không tương thích lẫn nhau. Ngoài ra cũng có những bộ vi điều khiển 16 bit và 32 bit được sản xuất bởi các hãng khác nhau. Với tất cả những bộ vi điều khiển khác nhau thì tiêu chuẩn để lựa chọn là:

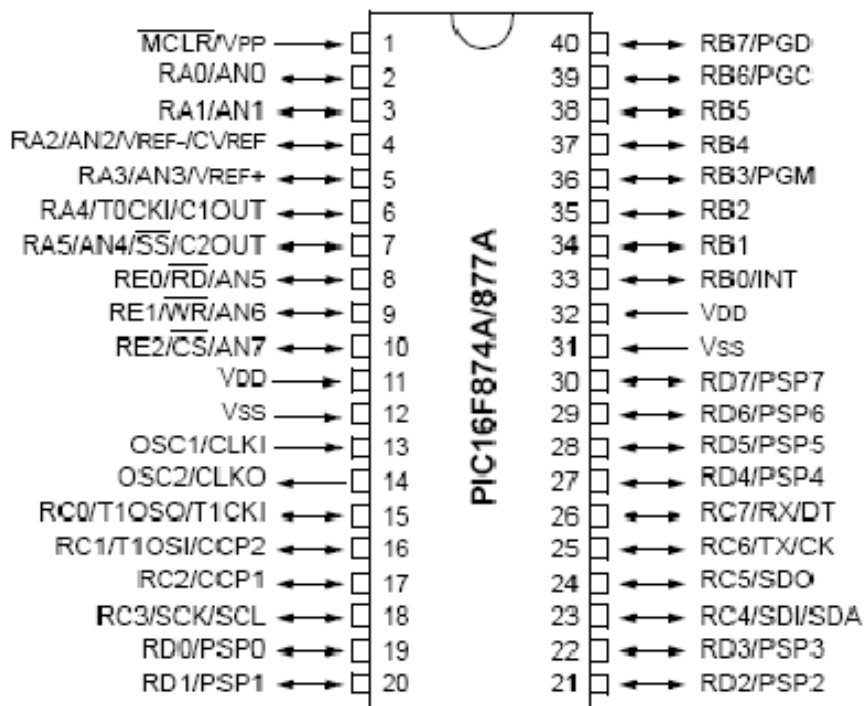
* Đáp ứng được nhu cầu tính toán của bài toán một cách hiệu quả, đầy đủ chức năng cần thiết và thấp nhất về mặt giá thành. Trong khi phân tích các nhu cầu của một dự án dựa trên bộ vi điều khiển chúng ta phải biết bộ vi điều khiển nào là 8 bit, 16 bit hay 32 bit có thể đáp ứng tốt nhất nhu cầu của bài toán một cách hiệu quả. Những tiêu chuẩn đó là:

- Tốc độ: tốc độ lớn nhất mà vi điều khiển hỗ trợ là bao nhiêu.
 - Kiểu đóng vỏ: Đóng vỏ kiểu DIP 40 chân hay QFP. Đây là yêu cầu quan trọng xét về không gian, kiểu lắp ráp và tạo mẫu thử cho sản phẩm cuối cùng.
 - Công suất tiêu thụ: Điều này đặc biệt khắc khe đối với các sản phẩm dùng pin, ắc quy.
 - Dung lượng bộ nhớ Rom và Ram trên chip.
 - Số chân vào ra và bộ định thời trên chip.
 - Khả năng dễ dàng nâng cấp cho hiệu suất cao hoặc giảm công suất tiêu thụ.
 - Giá thành cho một đơn vị: Điều này quan trọng quyết định giá thành sản phẩm mà một bộ vi điều khiển được sử dụng.
- *) Có sẵn các công cụ phát triển phần mềm như các trình biên dịch, trình hợp ngữ và gỡ rối.

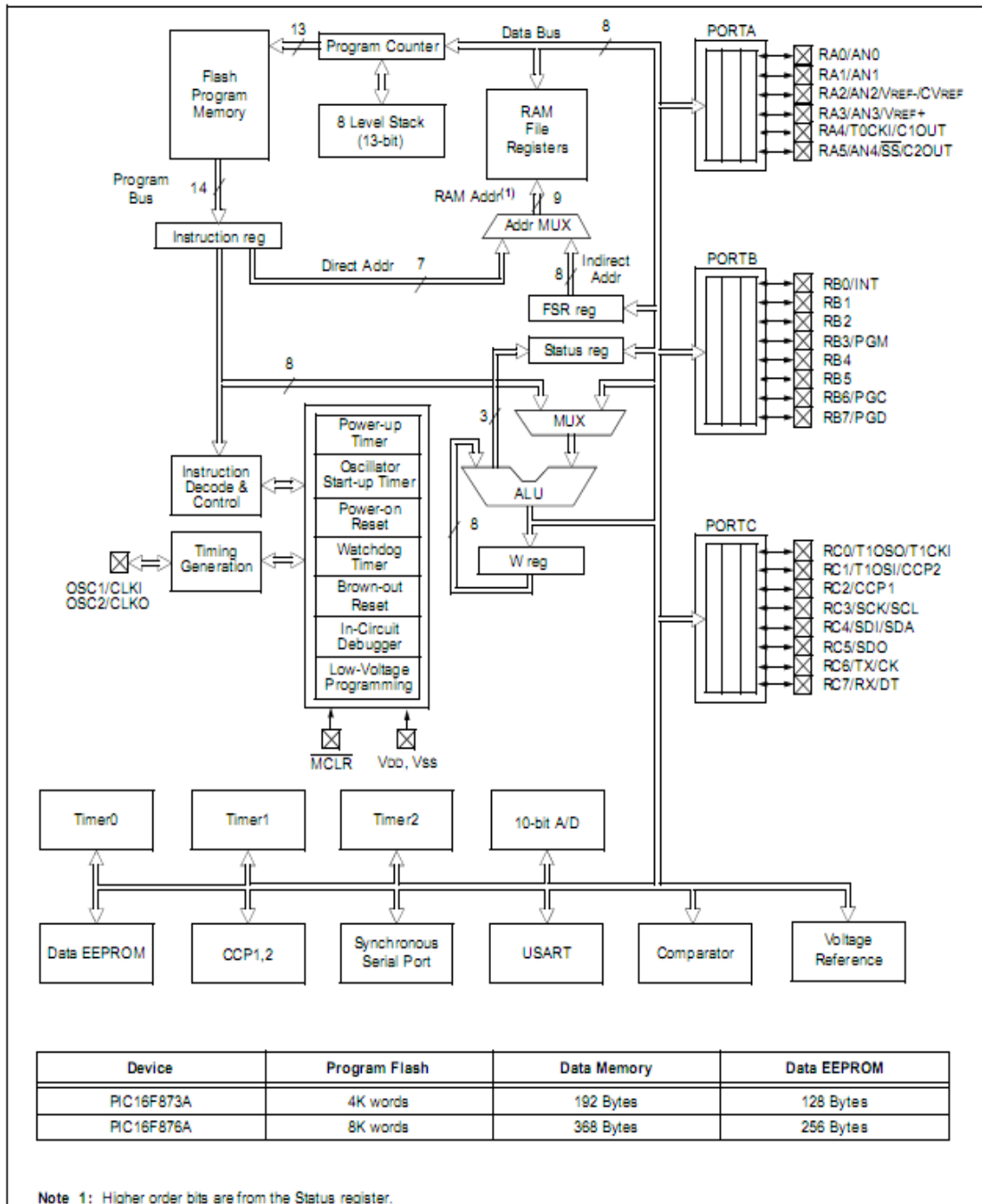
*) Nguồn các bộ vi điều khiển sẵn có nhiều và tin cậy. Khả năng sẵn sàng đáp ứng về số lượng trong hiện tại tương lai.

Hiện nay các bộ vi điều khiển 8 bit họ 8051 là có số lượng lớn nhất các nhà cung cấp đa dạng như Intel, Atmel, Philip... Nhưng về mặt tính năng và công năng thì có thể xem PIC vượt trội hơn rất nhiều so với 89 với nhiều module được tích hợp sẵn như ADC10 BIT, PWM 10 BIT, PROM 256 BYTE, COMPARATER, VERF COMPARATER, một đặc điểm nữa là tất cả các vi điều khiển PIC sử dụng thì đều có chuẩn PI tức chuẩn công nghiệp thay vì chuẩn PC (chuẩn dân dụng). Ngoài ra PIC còn được rất nhiều nhà sản xuất phần mềm tạo ra các ngôn ngữ hỗ trợ cho việc lập trình ngoài ngôn ngữ Assembly ra còn có thể sử dụng ngôn ngữ C thì sử dụng CCSC, HTPIC hay sử dụng Basic thì có MirkoBasic... và còn nhiều chương trình khác nữa để hỗ trợ cho việc lập trình bên cạnh ngôn ngữ kinh điển là assembler. Nên trong đề tài này tôi lựa chọn sử dụng vi điều khiển PIC làm bộ điều khiển chính, và ở đây là PIC16F877A.

1.1.1. Sơ đồ khối và bảng mô tả chức năng các chân của PIC16F877A



Hình 1.1. PIC 16F877A



Hình 1.2. Sơ đồ khối của PIC16F877A

Bảng mô tả chức năng các chân của PIC16F877A

Pin Name	DIP Pin#	PLCC Pin#	QFT Pin#	I/O/P Type	Buffer Type	Description
OSC1/CLKIN	13	14	30	1	ST/CMOS(4)	Đầu vào của xung dao động thạch anh/ngõ vào xung clock ngoại
OSC2/CLKOUT	1	2	18	O	-	Đầu ra của xung dao động thạch anh. Nối với thạch anh hay cộng hưởng trong chế độ dao động của thạch anh. Trong chế độ RC, ngõ ra của chân OSC2.
\overline{MCLR}/V_{pp}	1	2	18	I/P	ST	Ngõ vào của Master Clear(Reset) hoặc ngõ vào điện thế được lập trình. Chân này cho phép tín hiệu Reset thiết bị tác động ở mức thấp.
RA0/AN0	2	3	19	I/O	TTL	PORTA là port vào ra hai chiều. RA0 có thể làm ngõ vào tương tự thứ 0.
RA1/AN1	3	4	20	I/O	TTL	RA1 có thể làm ngõ vào tương tự thứ 1
RA2/AN2/VREF –	4	5	21	I/O	TTL	RA2 có thể làm ngõ vào tương tự 2 hoặc điện áp chuẩn tương tự âm.

Đồ án tốt nghiệp

RA3/AN3/VREF +	5	6	22	I/O	TTL	RA3 có thể làm ngõ vào tương tự 3 hoặc điện áp chuẩn tương tự dương.
RA4/T0CKI	6	7	23	I/O	ST	RA4 có thể làm ngõ vào xung clock cho bộ định thời Timer0.
RA5/ \overline{SS} /AN4	7	8	24	I/O	TTL	RA5 có thể làm ngõ vào tương tự thứ 4
RB0/INT	33	36	8	I/O	TTL/ST(1)	PORTB là port hai chiều. RB0 có thể làm chân ngắt ngoài
RB1	34	37	9	I/O	TTL	
RB2	35	38	10	I/O	TTL	
RB3/PGM	36	39	11	I/O	TTL	RB3 có thể làm ngõ vào của điện thế được lập trình ở mức thấp.
RB4	37	41	14	I/O	TTL	Interrupt-on-change pin. Interrupt-on-change pin. Interrupt-on-change pin hoặc In-Circuit Debugger pin .
RB5	38	42	15	I/O	TTL	
RB6/PGC	39	43	16	I/O	TTL/ST(2)	
RB7/PGD	40	44	17	I/O	TTL/ST(3)	

						Serial programming clock. Interrupt-on-change pin hoặc In-Circuit Debugger pin . Serial programming data .
RC0/T1OSO/T1CKI	15	16	32	I/O	ST	PORTC là port vào ra hai chiều. RC0 có thể là ngõ vào của bộ dao động Timer1 hoặc ngõ xung clock cho Timer1
RC1/T1OSI/CCP2	16	18	35	I/O	ST	RC1 có thể là ngõ vào của bộ dao động Timer1 hoặc ngõ vào Capture2/ngõ ra compare2/ngõ vào PWM2.
RC2/CCP1	17	19	36	I/O	ST	RC2 có thể ngõ vào capture1/ngõ ra compare1/ngõ vào PWM1
RC3/SCK/SCL	18	20	37	I/O	ST	RC3 có thể là ngõ vào xung
RC4/SDI/SDA	23	25	42	I/O	ST	Clock đồng bộ nội tiếp/ngõ ra trong cả hai chế độ SPI và I2C RC4 có thể là dữ liệu

Đồ án tốt nghiệp

						bên trong SPI(chế độ SPI) hoặc dữ liệu I/O(chế độ I ² C).
RC5/SDO	24	26	43	I/O	ST	RC5 có thể là dữ liệu ngoài SPI(chế độ SPI)
RC6/TX/CK	25	27	44	I/O	ST	RC6 có thể là chân truyền không đồng bộ USART hoặc đồng bộ với xung đồng hồ
RC7/RX/DT	26	29	1	I/O	ST	RC7 có thể là chân nhận không đồng bộ USART hoặc đồng bộ với dữ liệu.
RD0/PSP0	19	21	38	I/O	ST/TTL(3)	PORTD là port vào ra hai chiều hoặc là parallel slave port khi giao tiếp với bus của bộ vi xử lý.
RD1/PSP1	20	22	39	I/O	ST/TTL(3)	
RD2/PSP2	21	23	40	I/O	ST/TTL(3)	
RD3/PSP3	22	24	41	I/O	ST/TTL(3)	
RD4/PSP4	27	30	2	I/O	ST/TTL(3)	
RD5/PSP5	28	31	3	I/O	ST/TTL(3)	
RD6/PSP6	29	32	4	I/O	ST/TTL(3)	
RD7/PSP7	30	33	5	I/O	ST/TTL(3)	
RE0/ \overline{RD} /AN5	8	9	25	I/O	ST/TTL(3)	<p>PORTE là port vào ra hai chiều.</p> <p>RE0 có thể điều khiển việc đọc parrallel slave port hoặc là ngoc vào tương tự thứ 5.</p>

RE1/ \overline{WR} /AN6	9	10	26	I/O	ST/TTL(3)	RE1 có thể điều khiển việc ghi parallel slave port hoặc là ngõ vào tương tự thứ 6.
RE2/ \overline{CS} /AN7	10	11	27	I/O	ST/TTL(3)	RE2 có thể điều khiển việc chọn parallel slave port hoặc là ngõ vào tương tự thứ 7
V _{SS} V _{DD}	12, 31	13, 34	7, 28	P P		Cung cấp nguồn dương cho các mức logic và những chân I/O.
NC		1,17, 28, 40	12,1 3 33, 4			Những chân này không được nối bên trong và nó được để trống

Ghi chú: I = input; O = output; I/O = input/output; P = power

- = Not used; TTL = TTL input; ST = Schmitt Trigger input

1. Là vùng đệm có ngõ vào Trigger Schmitt khi được cấu hình như ngắt ngoài.

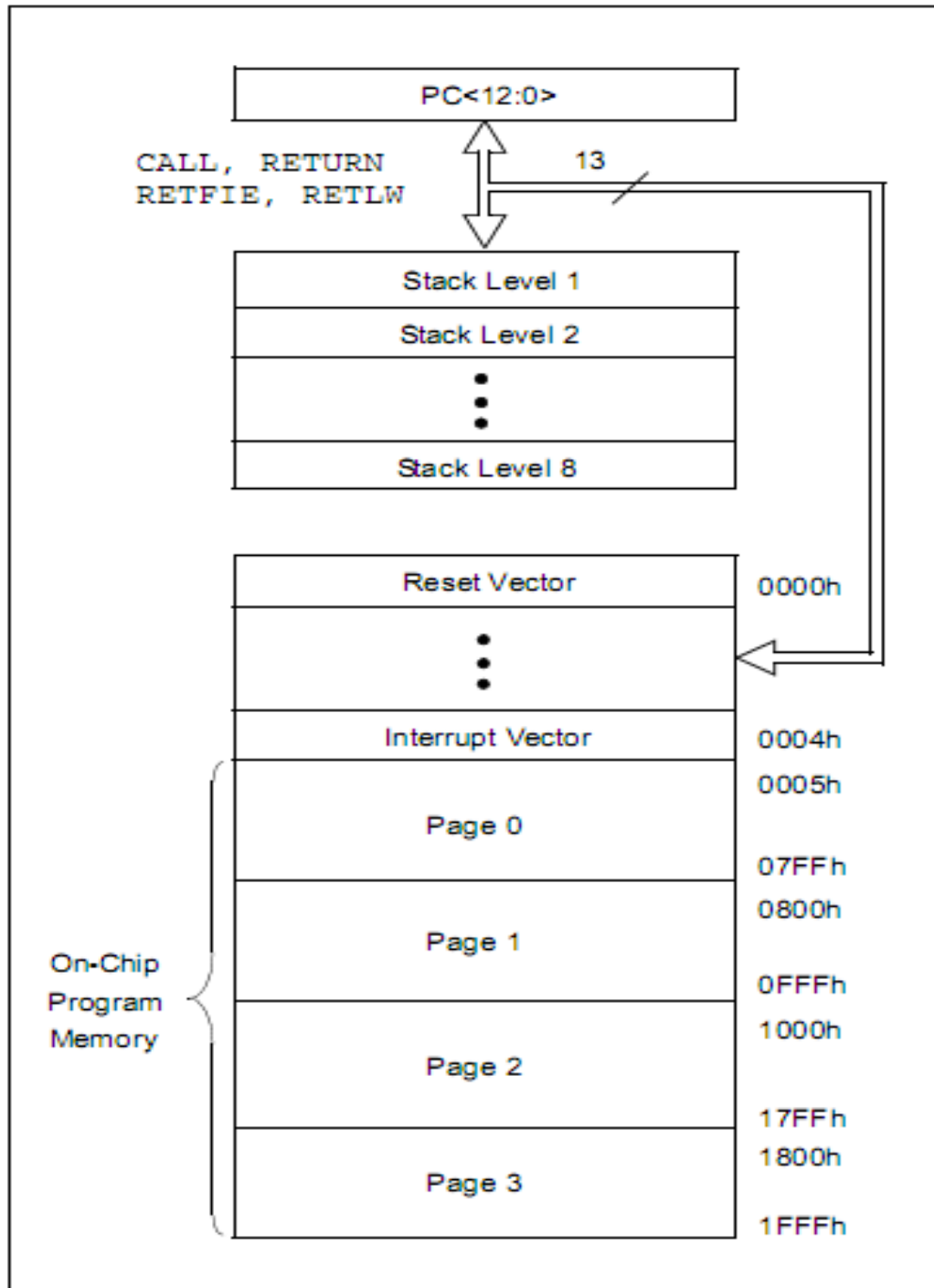
2. Là vùng đệm có ngõ vào Trigger Schmitt khi được sử dụng trong chế độ 9 Serial Programming.

3. Là vùng đệm có ngõ vào Trigger Schmitt khi được cấu hình như ngõ vào ra mục đích chung và là ngõ vào TTL khi sử dụng trong chế độ Parallel Slave Port (cho việc giao tiếp với các bus của bộ vi xử lý).

4. Là vùng đệm có ngõ vào Trigger Schmitt khi được cấu hình trong chế độ dao động RC và một ngõ vào CMOS khác.

1.1.2. Tổ chức bộ nhớ

Có 2 khối bộ nhớ trong các vi điều khiển họ PIC16F87X, bộ nhớ chương trình và bộ nhớ dữ liệu, với những bus riêng biệt để có thể truy cập đồng thời.



Hình 1.3. Ngăn xếp và bản đồ bộ nhớ chương trình PIC16F877A

1.1.2.1. Tổ chức của bộ nhớ chương trình

Các vi điều khiển họ PIC16F877A có bộ đếm chương trình 13 bit có khả năng định vị không gian bộ nhớ chương trình lên đến 8Kb. Các IC PIC16F877A có 8Kb bộ nhớ chương trình FLASH, các IC PIC16F873/874 chỉ có 4 Kb. Vector RESET đặt tại địa chỉ 0000h và vector ngắt tại địa chỉ 0004h.

1.1.2.2. Tổ chức bộ nhớ dữ liệu

Bộ nhớ dữ liệu được chia thành nhiều dãy và chứa các thanh ghi mục đích chung và các thanh ghi chức năng đặc biệt. BIT RP1 (STATUS <6>) và RP0 (STATUS <5>) là những bit dùng để chọn các dãy thanh ghi.

RP1:RP0	Bank
00	0
01	1
10	2
11	3

Chiều dài của mỗi dãy là 7Fh (128 byte). Phần thấp của mỗi dãy dùng để chứa các thanh ghi chức năng đặc biệt. Trên các thanh ghi chức năng đặc biệt là các thanh ghi mục đích chung, có chức năng như RAM tĩnh. Thường thì những thanh ghi đặc biệt được sử dụng từ một dãy và có thể được ánh xạ vào những dãy khác để giảm bớt đoạn mã và khả năng truy cập nhanh hơn.

1.1.2.3. Các thanh ghi mục đích chung

Các thanh ghi này có thể truy cập trực tiếp hoặc gián tiếp thông qua thanh ghi FSG (File Select Register).

						File Address	
Indirect addr. ^(*)	00h	Indirect addr. ^(*)	80h	Indirect addr. ^(*)	100h	Indirect addr. ^(*)	180h
TMR0	01h	OPTION_REG	81h	TMR0	101h	OPTION_REG	181h
PCL	02h	PCL	82h	PCL	102h	PCL	182h
STATUS	03h	STATUS	83h	STATUS	103h	STATUS	183h
FSR	04h	FSR	84h	FSR	104h	FSR	184h
PORTA	05h	TRISA	85h		105h		185h
PORTB	06h	TRISB	86h	PORTB	106h	TRISB	186h
PORTC	07h	TRISC	87h		107h		187h
PORTD ^(*)	08h	TRISD ^(*)	88h		108h		188h
PORTE ^(*)	09h	TRISE ^(*)	89h		109h		189h
PCLATH	0Ah	PCLATH	8Ah	PCLATH	10Ah	PCLATH	18Ah
INTCON	0Bh	INTCON	8Bh	INTCON	10Bh	INTCON	18Bh
PIR1	0Ch	PIE1	8Ch	EEDATA	10Ch	EECON1	18Ch
PIR2	0Dh	PIE2	8Dh	EEADR	10Dh	EECON2	18Dh
TMR1L	0Eh	PCON	8Eh	EEDATH	10Eh	Reserved ⁽²⁾	18Eh
TMR1H	0Fh		8Fh	EEADRH	10Fh	Reserved ⁽²⁾	18Fh
T1CON	10h		90h		110h		190h
TMR2	11h	SSPCON2	91h		111h		191h
T2CON	12h	PR2	92h		112h		192h
SSPBUF	13h	SSPAD	93h		113h		193h
SSPCON	14h	SSPSTAT	94h		114h		194h
CCPR1L	15h		95h		115h		195h
CCPR1H	16h		96h		116h		196h
CCP1CON	17h		97h	General Purpose Register 16 Bytes	117h	General Purpose Register 16 Bytes	197h
RCSTA	18h	TXSTA	98h		118h		198h
TXREG	19h	SPBRG	99h		119h		199h
RCREG	1Ah		9Ah		11Ah		19Ah
CCPR2L	1Bh		9Bh		11Bh		19Bh
CCPR2H	1Ch		9Ch		11Ch		19Ch
CCP2CON	1Dh		9Dh		11Dh		19Dh
ADRESH	1Eh	ADRESL	9Eh		11Eh		19Eh
ADCON0	1Fh	ADCON1	9Fh		11Fh		19Fh
	20h		A0h		120h		1A0h
General Purpose Register 96 Bytes		General Purpose Register 80 Bytes		General Purpose Register 80 Bytes		General Purpose Register 80 Bytes	
			EFh		16Fh		1EFh
		accesses 70h-7Fh	F0h	accesses 70h-7Fh	170h	accesses 70h - 7Fh	1F0h
Bank 0	7Fh	Bank 1	FFh	Bank 2	17Fh	Bank 3	1FFh

Hình 1.4. Các thanh ghi của PIC16F877A

1.1.2.4. Các thanh ghi chức năng đặc biệt

Các thanh ghi chức năng đặc biệt (Special Function Register) được sử dụng bởi CPU và các bộ nhớ ngoại vi để điều khiển các hoạt động được yêu cầu của thiết bị. Những thanh ghi này có chức năng như RAM tĩnh. Danh sách những thanh ghi này được trình bày ở bảng dưới. Các thanh ghi chức năng đặc biệt có thể chia thành hai loại: phần trung tâm (CPU) và phần ngoại vi.

1.1.2.5. Các thanh ghi trạng thái

	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	\overline{TO}	\overline{PD}	Z	DC	C
bit 7								bit 0
bit 7	IRP: Register Bank Select bit (used for indirect addressing) 1 = Bank 2, 3 (100h-1FFh) 0 = Bank 0, 1 (00h-FFh)							
bit 6-5	RP1:RP0: Register Bank Select bits (used for direct addressing) 11 = Bank 3 (180h-1FFh) 10 = Bank 2 (100h-17Fh) 01 = Bank 1 (80h-FFh) 00 = Bank 0 (00h-7Fh) Each bank is 128 bytes.							
bit 4	\overline{TO}: Time-out bit 1 = After power-up, CLRWDI instruction or SLEEP instruction 0 = A WDT time-out occurred							
bit 3	\overline{PD}: Power-down bit 1 = After power-up or by the CLRWDI instruction 0 = By execution of the SLEEP instruction							
bit 2	Z: Zero bit 1 = The result of an arithmetic or logic operation is zero 0 = The result of an arithmetic or logic operation is not zero							
bit 1	DC: Digit carry/borrow bit (ADDWF, ADDLW, SUBLW, SUBWF instructions) (for borrow, the polarity is reversed) 1 = A carry-out from the 4th low order bit of the result occurred 0 = No carry-out from the 4th low order bit of the result							
bit 0	C: Carry/borrow bit (ADDWF, ADDLW, SUBLW, SUBWF instructions) 1 = A carry-out from the Most Significant bit of the result occurred 0 = No carry-out from the Most Significant bit of the result occurred							

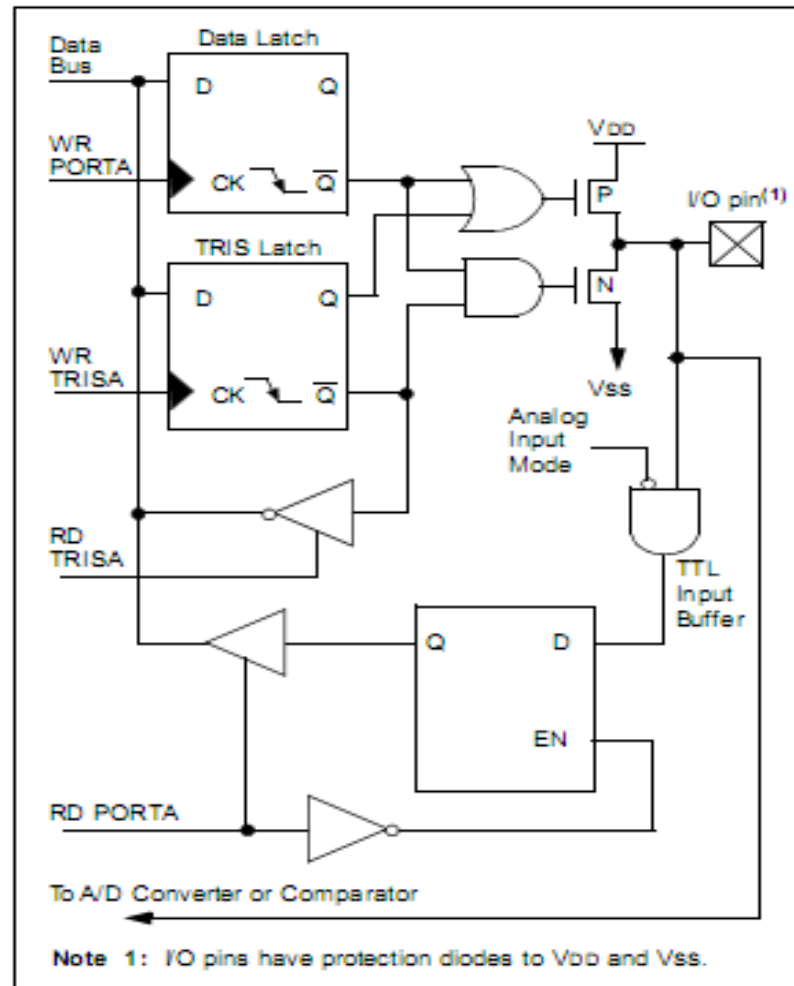
Hình 1.5. Thanh ghi trạng thái (địa chỉ 03h, 83h, 103h, 183h)

Thanh ghi trạng thái chứa các trạng thái số học của bộ ALU, trạng thái RESET và những bits chọn dãy thanh ghi cho bộ nhớ dữ liệu. Thanh ghi trạng thái có thể là đích cho bất kì lệnh nào, giống như những thanh ghi khác. Nếu thanh ghi trạng thái là đích cho một lệnh mà ảnh hưởng đến các cờ Z, DC hoặc C, và sau đó những bit này sẽ được vô hiệu hoá. Những bit này có thể

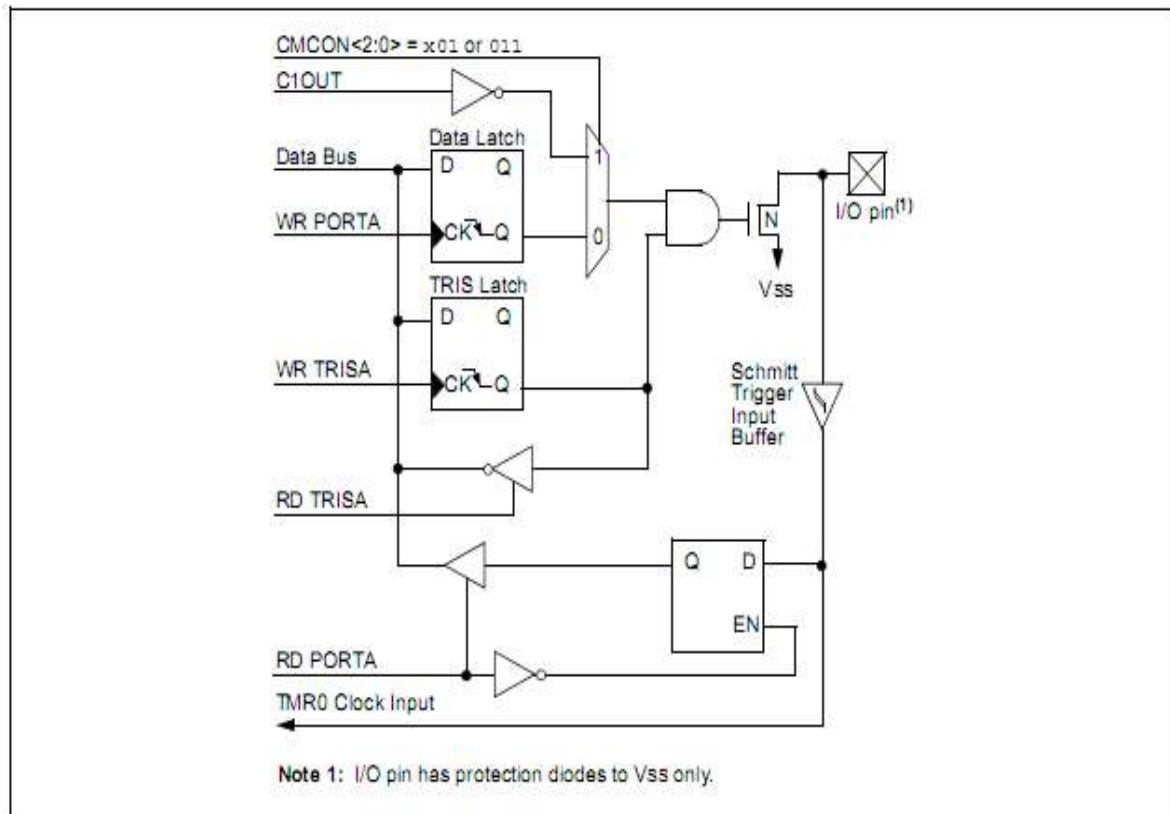
đặt hoặc xoá tùy theo trạng thái logic của thiết bị. Hơn nữa hai bit \overline{TO} và PD thì không cho phép ghi, vì vậy kết quả của một tập lệnh mà thanh ghi trạng thái là đích có thể khác hơn dự định. Ví dụ, CLR STATUS sẽ xoá 3 bit cao nhất và đặt bit Z. Lúc này các bits của thanh ghi trạng thái là 000u u1uu (u = unchanged). Chỉ có các lệnh BCF, BSF, SWAPF và MOVWF được sử dụng để thay đổi thanh ghi trạng thái, bởi vì những lệnh này không làm ảnh hưởng đến các bit Z, DC hoặc C từ thanh ghi trạng thái. Đối với những lệnh khác thì không ảnh hưởng đến những bits trạng thái này.

1.1.3. Các cổng của PIC 16F877A

1.1.3.1. PORTA và thanh ghi TRISA



Hình 1.6. Sơ đồ khối của chân RA3:RA0 và RA5

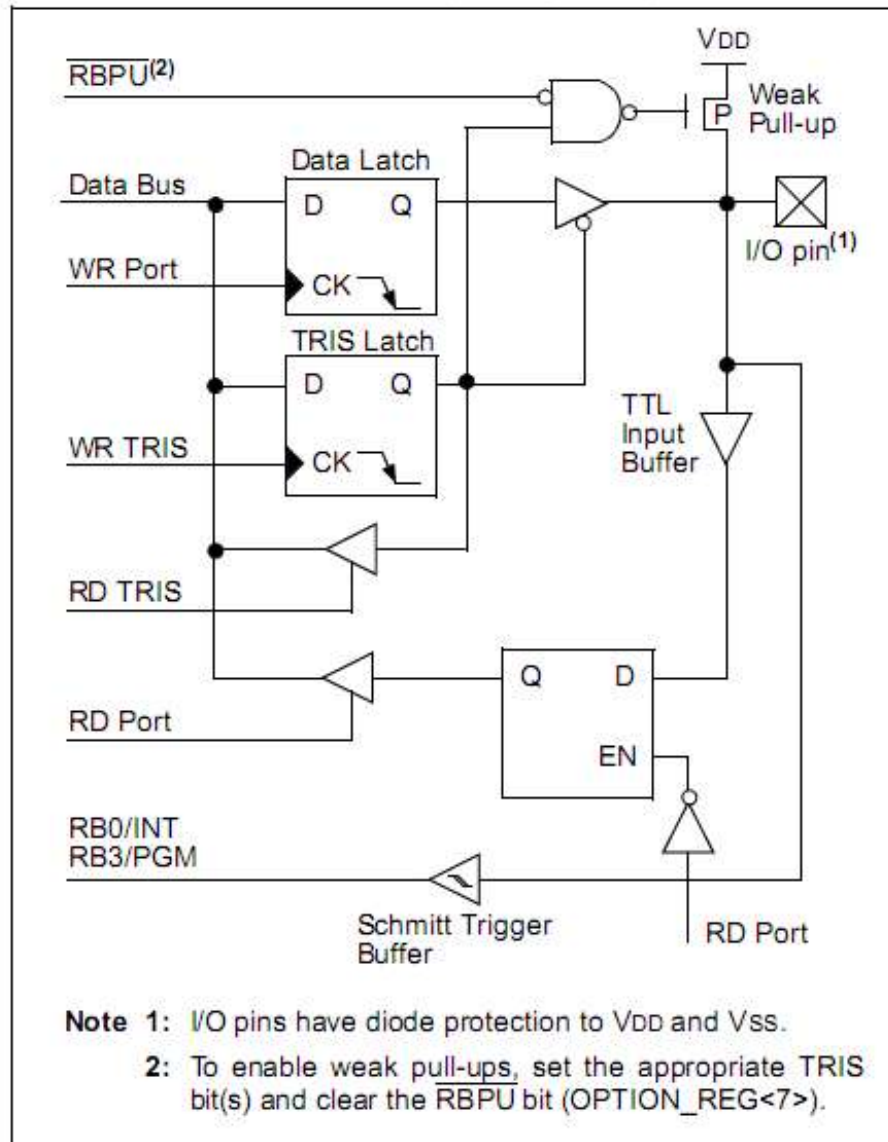


Hình 1.7. Sơ đồ khối của chân RA4/T0CKI

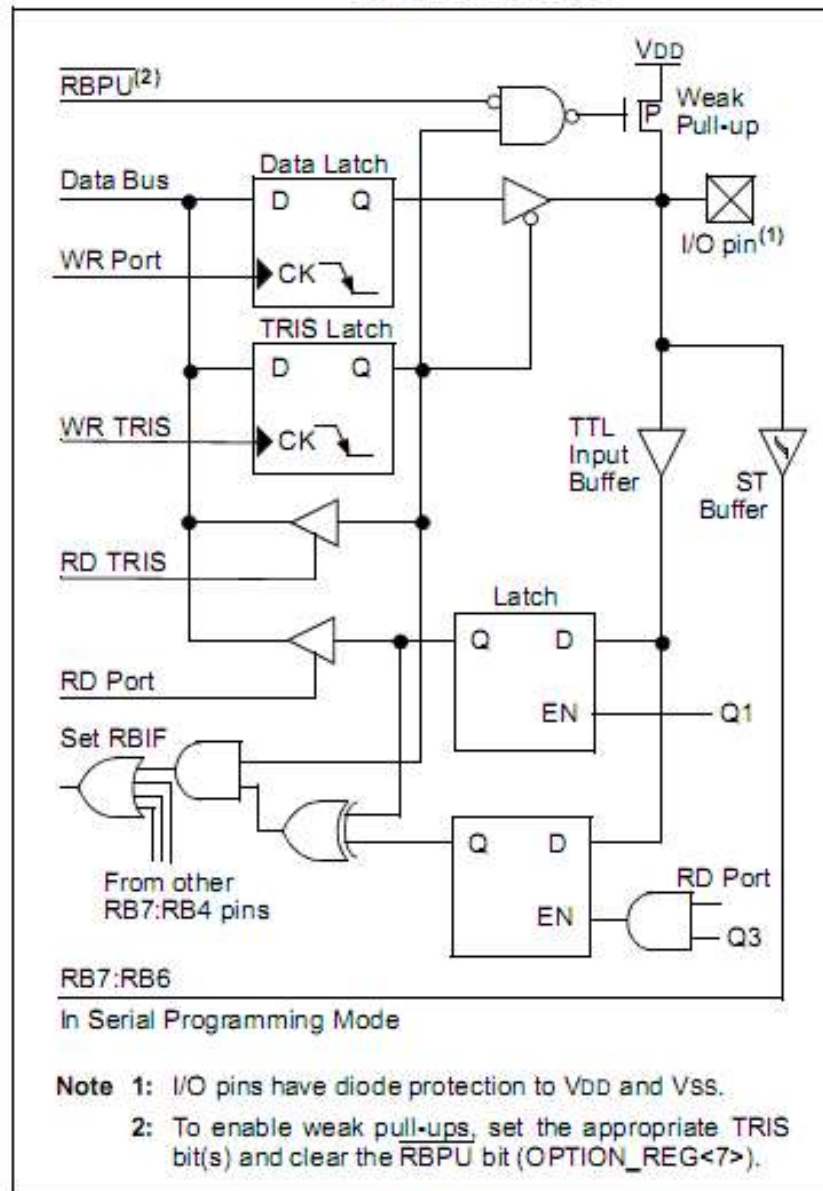
1.1.3.2. PORTB và thanh ghi TRISB

PORTB có độ rộng 8 bit, là port vào ra hai chiều. Ba chân của PORTB được đa hợp với chức năng lập trình mức điện thế thấp (Low Voltage Programming): RB3/PGM, RB6/PGC và RB7/PGD. Mỗi chân của PORTB có một điện trở kéo bên trong. Một bit điều khiển có thể mở tắt cả những điện trở kéo này lên. Điều này được thực hiện bằng cách xoá bit \overline{RBPU} (OPTION_REG<7>). Những điện trở này bị cấm khi có một Power-on Reset. Bốn chân của PORTB: RB7 đến RB4 có một ngắt để thay đổi đặc tính. Chỉ những chân được cấu hình như ngõ vào mới có thể gây ra ngắt này. Những chân vào (RB7:RB4) được so sánh với giá trị được chốt trước đó trong lần đọc cuối cùng của PORTB. Các kết quả không phù hợp ở ngõ ra trên chân RB7:RB4 được OR với nhau để phát ra một ngắt Port thay đổi RB với cờ ngắt là RBIF (INTCON<0>). Ngắt này có thể đánh thức thiết bị từ trạng thái nghỉ (SLEEP). Trong thủ tục phục vụ ngắt người sử dụng có thể xoá ngắt theo cách sau:

- a) Đọc hoặc ghi bất kì lên PORTB. Điều này sẽ kết thúc điều kiện không hoà hợp.
- b) Xoá bit cờ RBIF.



Hình 1.8. Sơ đồ khối các chân RB3:RB0

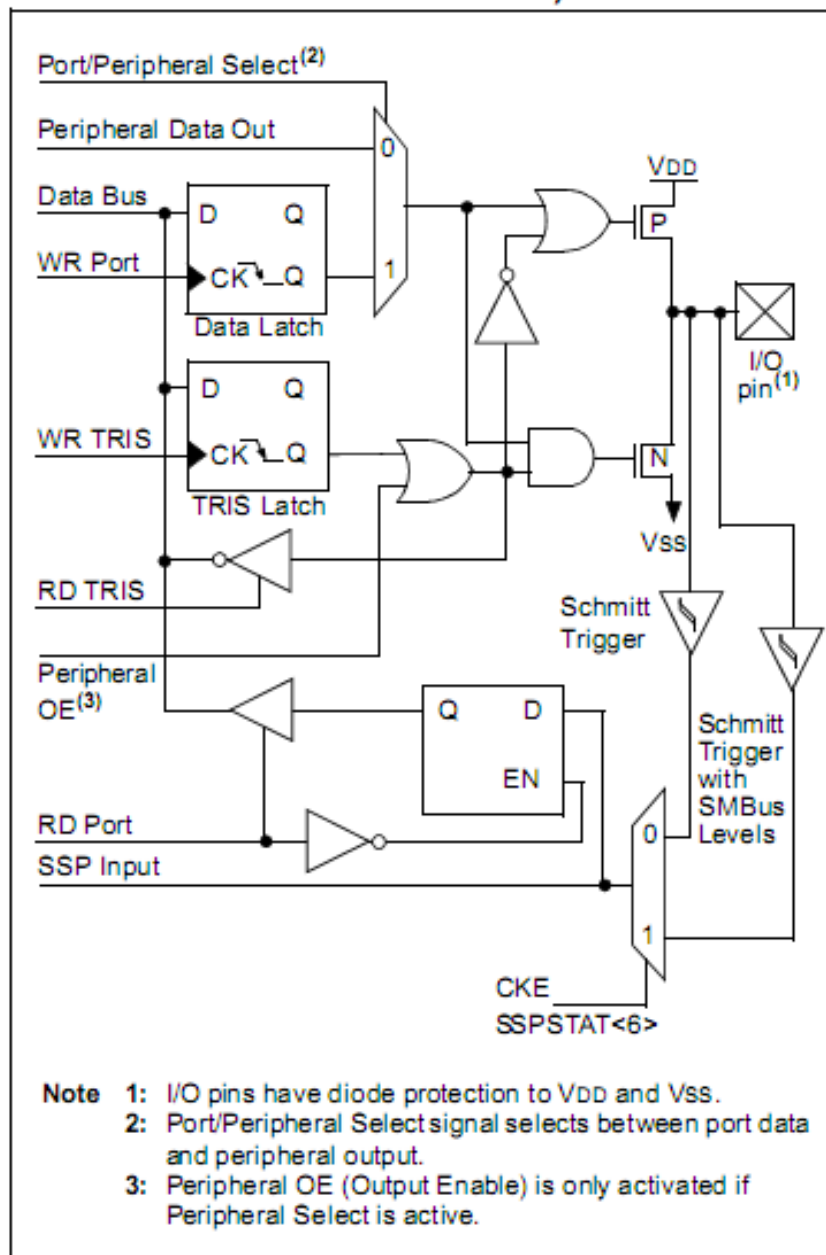


Hình 1.9. Sơ đồ khối các chân RB7:RB4

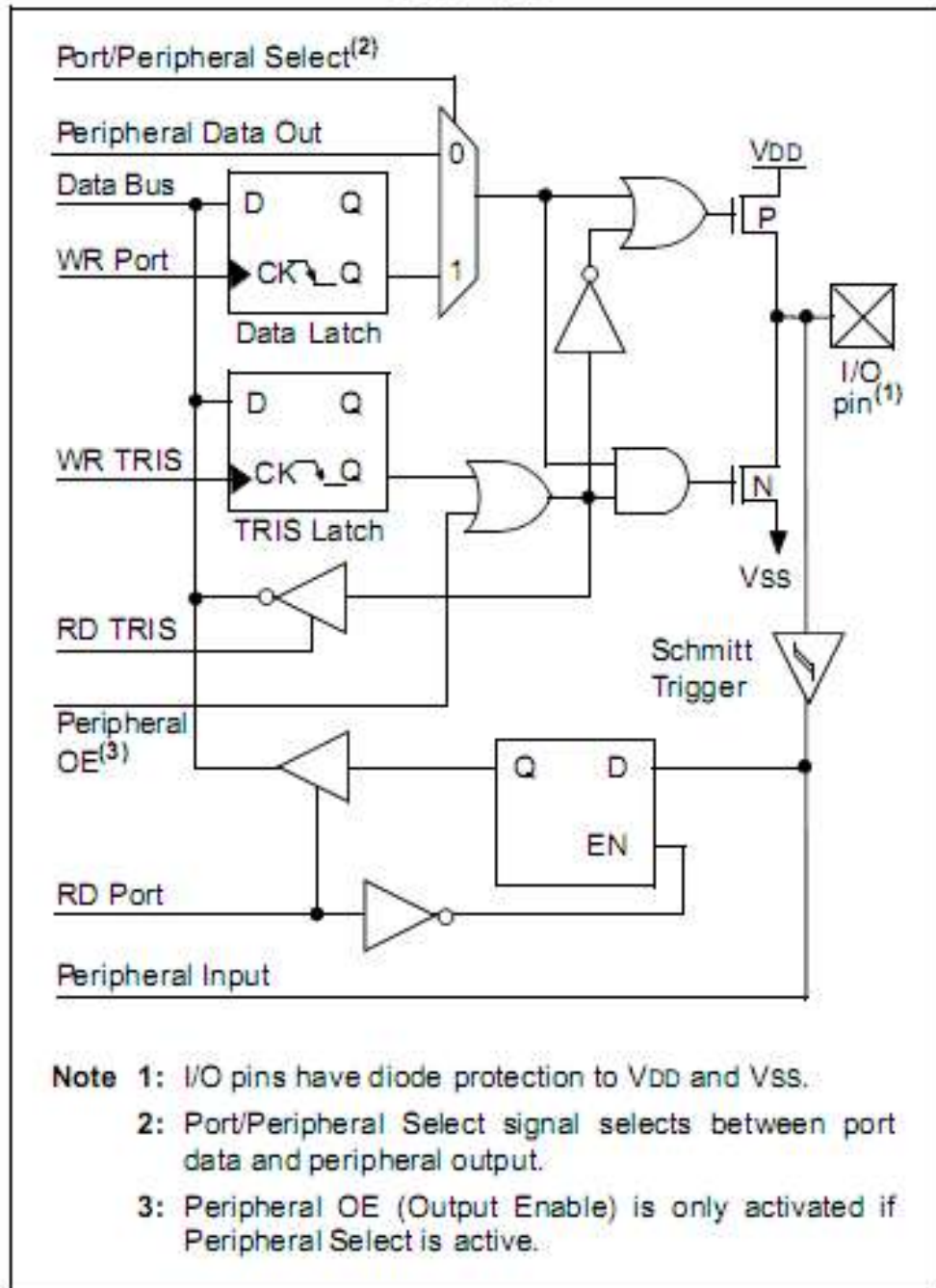
1.1.3.3. PORTC và thanh ghi TRISC

PORTC có độ rộng là 8 bit, là port hai chiều. Thanh ghi dữ liệu trực tiếp tương ứng là TRISC. Cho tất cả các bit của TRISC là 1 thì các chân tương ứng ở PORTC là ngõ vào. Cho tất cả các bit của TRISC là 0 thì các chân tương ứng ở PORTC là ngõ ra. PORTC được đa hợp với vài chức năng ngoại vi, những chân của PORTC có đệm Trigger Schmitt ở ngõ vào. Khi bộ I2C được cho phép, chân 3 và 4 của PORTC có thể cấu hình với mức I2C bình thường, hoặc với mức SMBus bằng cách sử dụng bit CKE (SSPSTAT<6>).

Khi những chức năng ngoại vi được cho phép, chúng ta cần phải quan tâm đến việc định nghĩa các bits của TRIS cho mỗi chân của PORTC. Một vài thiết bị ngoại vi ghi đè lên bit TRIS thì tạo nên một chân ở ngõ ra, trong khi những thiết bị ngoại vi khác ghi đè lên bit TRIS thì sẽ tạo nên một chân ở ngõ vào. Khi những bit TRIS ghi đè bị tác động trong khi thiết bị ngoại vi được cho phép, những lệnh đọc thay thế ghi (BSF, BCF, XORWF) với TRISC là nơi đến cần phải được tránh. Người sử dụng cần phải chỉ ra vùng ngoại vi tương ứng để đảm bảo cho việc đặt TRIS bit là đúng.



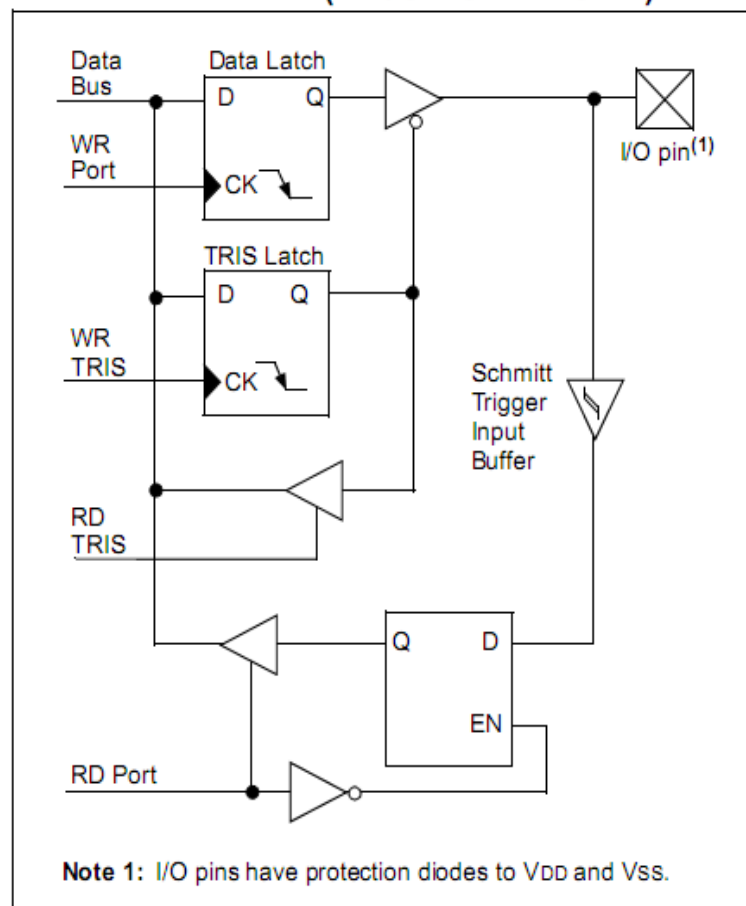
Hình 1.10. Sơ đồ khối của các chân RC<4:3>



Hình 1.11. Sơ đồ khối của các chân RC<2:0> và RC<7:5>

1.1.3.4. PORTD và thanh ghi TRISD

PORTD là port 8 bit với đệm Trigger Schmitt ở ngõ vào. Mỗi chân có thể được cấu hình riêng lẻ như một ngõ vào hoặc ngõ ra. PORTD có thể được cấu hình như port của bộ vi xử lý rộng 8 bit (parallel slave port) bằng cách đặt bit điều khiển PSPMODE (TRISE <4>). Trong chế độ này, đệm ở ngõ vào là TTL.



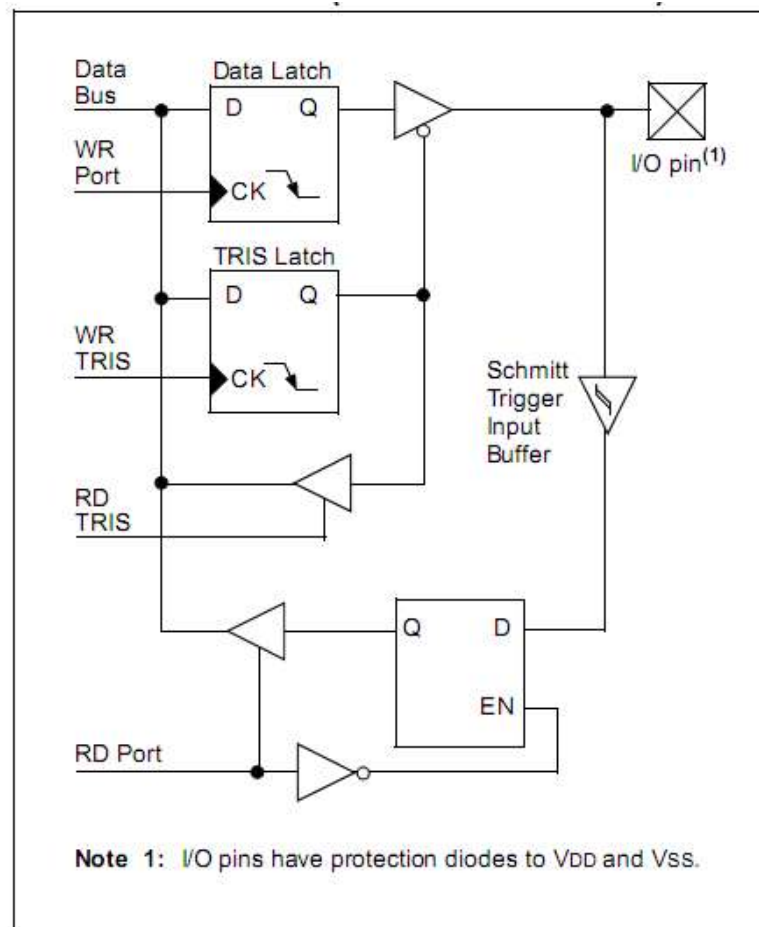
Hình 1.12. Sơ đồ khối của PORTD (trong chế độ là port I/O)

1.1.3.5. PORTE và thanh ghi TRISE

PORTE có ba chân (RE0/RD/AN5, RE1/WR/AN6, và RE2/CS/AN7) mỗi chân được cấu hình riêng lẻ như những ngõ vào hoặc những ngõ ra. Những chân này có đệm Trigger Schmitt ở ngõ vào. Những chân của PORTE đóng vai trò như những ngõ vào điều khiển vào ra cho Port của vi xử lý khi bit PSMODE (TRISE <4>) được đặt. Trong chế độ này, người sử dụng cần phải chắc chắn rằng những bit TRISE <2:0> được đặt, và chắc rằng những

chân này được cấu hình như những ngõ vào số. Cũng bảo đảm rằng ADCON1 được cấu hình cho vào ra số. Trong chế độ này, những đệm ở ngõ vào là TTL.

Những chân của PORTE được đa hợp với những ngõ vào tương tự, Khi được chọn cho ngõ vào tương tự, những chân này sẽ đọc giá trị "0". TRISE điều khiển hướng của những chân RE chỉ khi những chân này được sử dụng như những ngõ vào tương tự. Người sử dụng cần phải giữ những chân được cấu hình như những ngõ vào khi sử dụng chúng như những ngõ vào tương tự.



Hình 1.13. Sơ đồ khối của PORTE (trong chế độ I/O port)

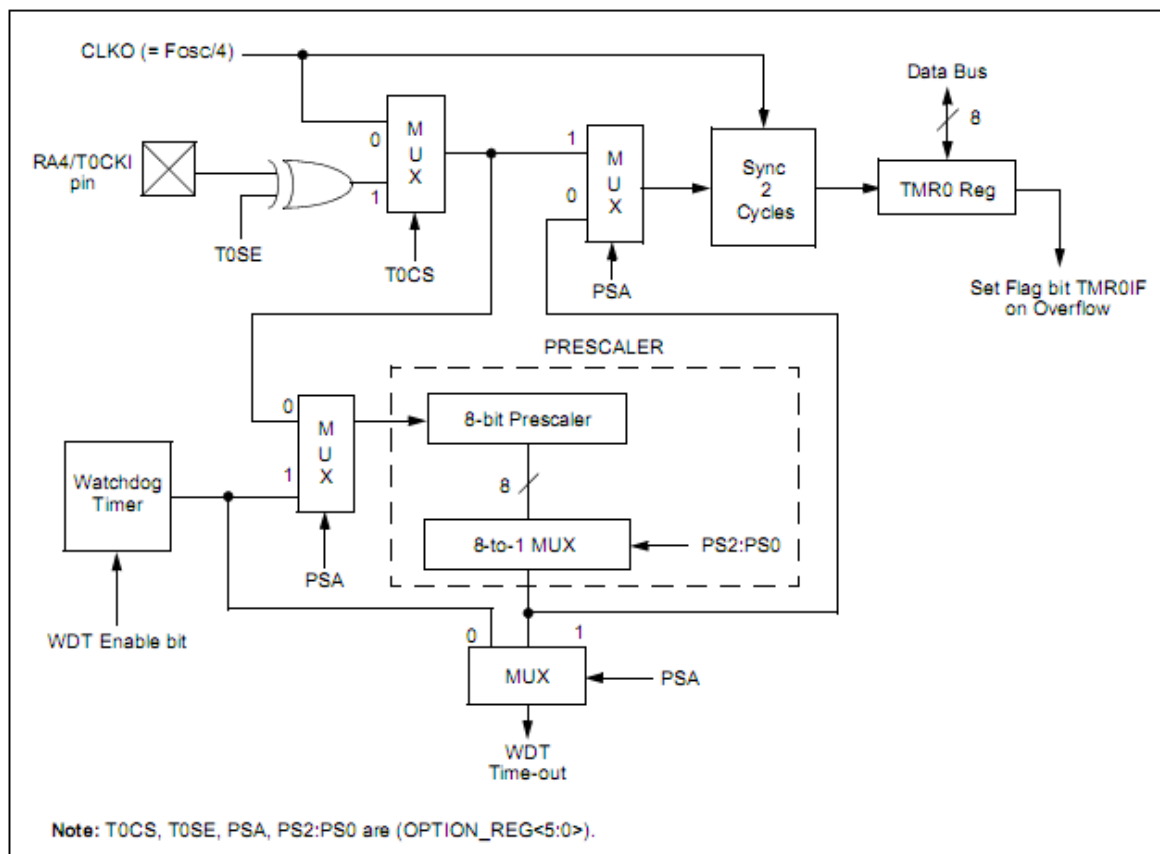
1.1.4. Hoạt động của định thời

1.1.4.1. Bộ định thời TIMER0

Bộ định thời/bộ đếm Timer0 có các đặc tính sau:

- Bộ định thời / bộ đếm 8 bit
- Cho phép đọc và ghi
- Bộ chia 8 bit lập trình được bằng phần mềm
- Chọn xung clock nội hoặc ngoại
- Ngắt khi có sự tràn từ FFh đến 00h
- Chọn sườn cho xung clock ngoài

Sơ đồ khối của bộ định thời Timer0 và bộ chia dùng chung với WDT được đưa ra trong hình 1.14.



Hình 1.14. Sơ đồ khối của bộ định thời Timer0 và bộ chia dùng chung với WDT

Chế độ định thời (Timer) được chọn bằng cách xoá bit T0CS (OPTION_REG<5>). Trong chế độ định thời, bộ định thời Timer0 sẽ tăng dần sau mỗi chu kì lệnh (không có bộ chia). Nếu thanh ghi TMR0 được ghi thì sự tăng sẽ bị ngăn lại sau hai chu kì lệnh.

Chế độ đếm (Counter) được chọn bằng cách xoá bit T0CS (OPTION_REG<5>). Trong chế độ đếm, Timer0 sẽ tăng dần ở mỗi cạnh lên xuống của chân RA4/T0CKI. Sự tăng sườn được xác định bởi bit Timer0 Source Edge Select, T0SE (OPTION_RE<4>). Bộ chia chỉ được dùng chung qua lại giữa bộ định thời Timer0 và bộ định thời Watchdog. Bộ chia không cho phép đọc hoặc ghi

Ngắt Timer0

Ngắt TMR0 được phát ra khi thanh ghi TMR0 tràn từ FFh đến 00h. Sự tràn này sẽ đặt bit T0IF (INTCON<2>). Ngắt này có thể được giấu đi bằng cách xoá bit T0IE (INTCON<5>). Bit T0IF cần phải được xoá trong chương trình bởi thủ tục phục vụ ngắt của bộ định thời Timer0 trước khi ngắt này được cho phép lại.

Sử dụng Timer0 với xung clock ngoài

Khi bộ chia không được sử dụng, clock ngoài đặt vào thì giống như bộ chia ở ngõ ra. Sự đồng bộ của chân T0CKI với clock ngoài được thực hiện bằng cách lấy mẫu bộ chia ở ngõ ra trên chân Q2 và Q4. Vì vậy thực sự cần thiết để chân T0CKI ở mức cao trong ít nhất 2 chu kỳ máy và ở mức thấp trong ít nhất 2 chu kỳ máy.

Bộ chia

Thiết bị PIC16F87X chỉ có một bộ chia mà được dùng chung bởi bộ định thời TIMER0 và bộ định thời Watchdog. Bộ chia có các Hệ số chia dùng cho Timer0 hoặc bộ WDT. Các hệ số này không có khả năng đọc và khả năng viết. Để chọn hệ số chia xung vào Timer0 hoặc cho bộ WDT ta tiến hành xoá hoặc đặt bit PSA của thanh ghi OPTION_REG<3>.

Những bit PS2, PS1, PS0 của thanh ghi OPTION_REG<2:0> dùng để xác lập các hệ số chia.

1.1.4.2. Bộ định thời TIMER1

Bộ định thời TIMER1 là một bộ định thời/bộ đếm 16 bit gồm hai thanh ghi TMR1H (Byte cao) và TMR1L (byte thấp) mà có thể đọc hoặc ghi. Cặp thanh ghi này tăng số đếm từ 0000h đến FFFFh và báo tràn sẽ xuất hiện khi có sự chuyển số đếm từ FFFFh xuống 0000h. Ngắt, nếu được phép có thể phát ra khi có số đếm tràn và được đặt ở bit cờ ngắt TMR1IF. Ngắt có thể được phép hoặc cấm bằng cách đặt hoặc xoá bit cho phép ngắt TMR1IE.

Bộ định thời Timer1 có thể được cấu hình để hoạt động một trong hai chế độ sau:

- Định thời một khoảng thời gian (timer)
- Đếm sự kiện (Counter)

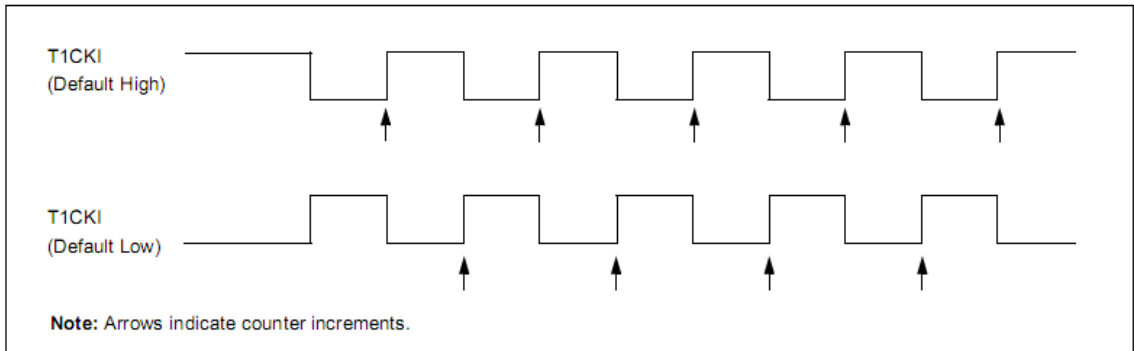
Việc lựa chọn một trong hai chế độ được xác định bằng cách đặt hoặc xoá bit điều khiển TMR1ON.

----	----	T1CKPS1	T1CKPS0	T1OSCEN	T1SYNC	TMR1CS	TMR1ON	
Bit7								Bit0

- | | |
|---------|--|
| Bit 7-6 | Không được định nghĩa |
| Bit 5-4 | bit chọn bộ chia clock cho timer1 |
| Bit 3 | bit điều khiển cho phép bộ dao động Timer1 |
| Bit 2 | bit điều khiển clock ngoài Timer |
| Bit 1 | bit chọn nguồn clock cho Timer1 |
| Bit 0 | bit điều khiển hoạt động của Timer1 |

Chế độ Timer

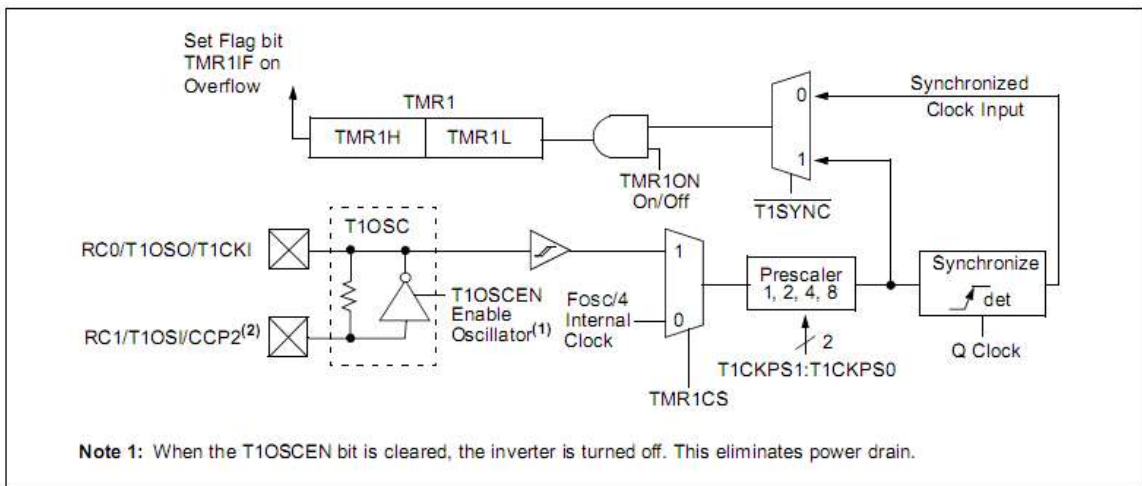
Chế độ Timer được chọn bằng cách xoá TMR1CS. Trong chế độ này, Nguồn clock đặt vào Timer là mạch dao động $F_{OSC}/4$. Bit điều khiển đồng bộ không bị tác động vì clock ngoài luôn luôn đồng bộ.



Hình 1.15. Cảnh tăng timer1

Chế độ counter

Trong chế độ này, bộ định thời tăng số đếm qua clock ngoài. Việc tăng xảy ra sau mỗi sườn lên của xung clock ngoài. Bộ định thời phải có một sườn lên trước khi việc đếm bắt đầu.



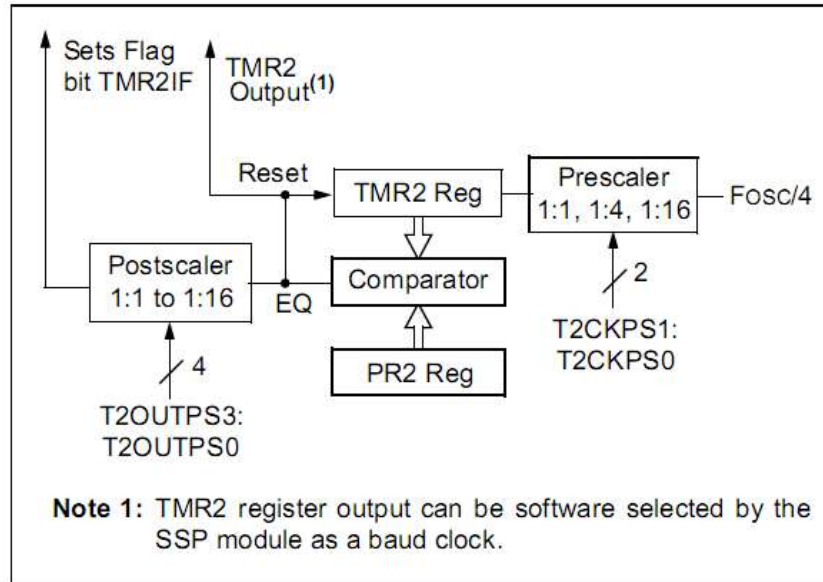
Hình 1.16. Sơ đồ khối bộ định thời timer1

1.1.4.3. Bộ định thời TIMER2

Bộ định thời TIMER2 là bộ định thời 8 bit với một bộ đếm và một bộ potscaler. Nó thường dùng chung với bộ CCP trong chế độ PWM (sẽ được đề cập ở phần sau). Thanh ghi TMR2 có thể đọc hoặc ghi và được xoá khi có bất kì tín hiệu reset nào của thiết bị

Bộ định thời TIMER2 có một thanh ghi chu kỳ 8 bit, PR2. Bộ định thời tăng số đếm lên từ 00h đến giá trị được ghi trong thanh ghi TR2 và sau đó reset lại giá trị 00h trong chu kỳ kế tiếp. PR2 là thanh ghi có thể đọc hoặc ghi.

Giá trị trùng hợp trong thanh ghi TMR2 được đi qua bộ postscaler 4 bit để phát ra một ngắt TMR2 (được đặt ở bit cờ ngắt TMR2IF). Bộ định thời TIMER2 có thể được tắt (không hoạt động) bằng cách xoá bit điều khiển TMR2ON để giảm thiểu công suất tiêu tán nguồn.



Hình 1.17. Sơ đồ khối của TIMER2

U-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 7							bit 0

- bit 7 **Unimplemented:** Read as '0'
- bit 6-3 **TOUTPS3:TOUTPS0:** Timer2 Output Postscale Select bits
 - 0000 = 1:1 postscale
 - 0001 = 1:2 postscale
 - 0010 = 1:3 postscale
 -
 -
 -
 - 1111 = 1:16 postscale
- bit 2 **TMR2ON:** Timer2 On bit
 - 1 = Timer2 is on
 - 0 = Timer2 is off
- bit 1-0 **T2CKPS1:T2CKPS0:** Timer2 Clock Prescale Select bits
 - 00 = Prescaler is 1
 - 01 = Prescaler is 4
 - 1x = Prescaler is 16

Legend:			
R = Readable bit	W = Writable bit	U = Unimplemented bit, read as '0'	
- n = Value at POR	'1' = Bit is set	'0' = Bit is cleared	x = Bit is unknown

Hình 1.18. T2CON: Thanh ghi điều khiển Timer2 (địa chỉ 12h)

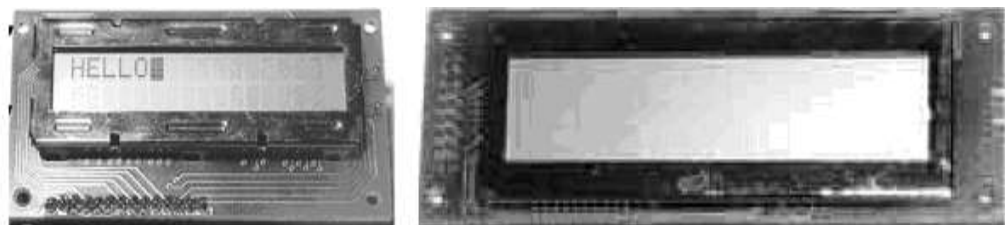
Một đặc điểm khác của vi điều khiển Pic16F877A là có bộ dao động chủ trên chip điều, nó sẽ giúp tránh được những sai số không cần thiết trong việc tạo xung dao động, vi điều khiển Pic16F877A có khả năng tự Reset bằng bộ WDT, và có thêm 256 byte EEPROM. Nhưng giá thành của Pic đắt hơn so với 8051.

1.2. THIẾT BỊ HIỂN THỊ LCD

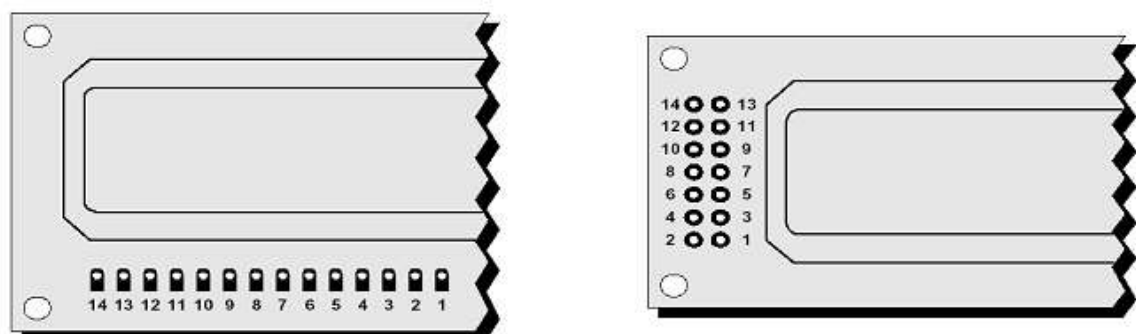
Ngày nay, thiết bị hiển thị LCD (Liquid Crystal Display) được sử dụng trong rất nhiều các ứng dụng của VĐK. LCD có rất nhiều ưu điểm so với các dạng hiển thị khác như nó có khả năng hiển thị kí tự đa dạng, trực quan (chữ, số và kí tự đồ họa), dễ dàng đưa vào mạch ứng dụng theo nhiều giao thức giao tiếp khác nhau, tốn rất ít tài nguyên hệ thống và giá thành rẻ ... Trong đề tài này tôi sử dụng HD44780 của Hitachi, một loại thiết bị hiển thị LCD rất thông dụng ở nước ta.

1.2.1. Hình dáng kích thước.

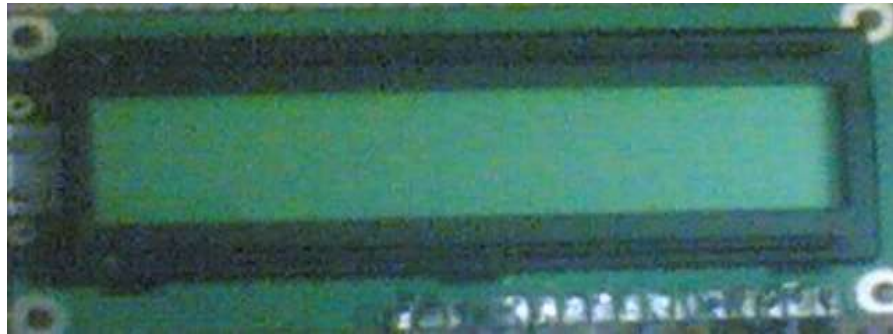
Có rất nhiều loại LCD với nhiều hình dáng và kích thước khác nhau, trên hình 1.19. là hai loại LCD thông dụng.



Hình 1.19. Hình hai loại LCD thông dụng



Hình 1.20. Sơ đồ chân của LCD



Hình 1.21. LCD loại DM 1602A.

Khi sản xuất LCD, nhà sản xuất đã tích hợp chip điều khiển (HD44780) bên trong lớp vỏ và chỉ đưa các chân giao tiếp cần thiết. Các chân này được đánh số thứ tự và đặt tên như hình 1.20.

1.2.2. Các chân chức năng.

Bảng 3.1. Các chân chức năng của HD44780.

Chân số	Tên	Chức năng
1	Vss	Chân nối đất cho LCD, khi thiết kế mạch ta nối chân này với GND của mạch điều khiển.
2	Vdd	Chân cấp nguồn cho LCD, khi thiết kế mạch ta nối chân này với 5V của mạch điều khiển.
3	Vo	Chân này dùng để điều chỉnh độ tương phản của LCD.
4	RS	Chân chọn thanh ghi (Register select). Nối chân RS với logic “0” (GND) hoặc logic “1” (Vcc) để chọn thanh ghi. + Logic “0”: Bus DB0-DB7 sẽ nối với thanh ghi lệnh IR của LCD (ở chế độ “ghi” - write) hoặc nối với bộ đếm địa chỉ của LCD (ở chế độ “đọc” - read) + Logic “1”: Bus DB0-DB7 sẽ nối với thanh ghi dữ liệu DR bên trong LCD.
5	RW	Chân chọn chế độ đọc/ghi (Read/Write). Nối chân R/W với logic “0” để LCD hoạt động ở chế độ ghi, hoặc nối với logic “1” để LCD ở chế độ đọc.
6	E	Chân cho phép (Enable). Sau khi các tín hiệu được đặt lên bus DB0-DB7, các lệnh chỉ được chấp nhận khi có 1 xung cho phép của chân E. + Ở chế độ ghi: Dữ liệu ở bus sẽ được LCD chuyển vào

		(chấp nhận) thanh ghi bên trong nó khi phát hiện một xung (low-to-high transition) của tín hiệu chân E. + Ở chế độ đọc: Dữ liệu sẽ được LCD xuất ra DB0-DB7 khi phát hiện sườn lên (low-to-high transition) ở chân E và được LCD giữ ở bus đến khi nào chân E xuống mức thấp.
7÷14	DB0÷ DB7	8 đường của bus dữ liệu dùng để trao đổi thông tin với MPU. Có 2 chế độ sử dụng 8 đường bus này: + Chế độ 8 bit: Dữ liệu được truyền trên cả 8 đường, với bit MSB là bit DB7. + Chế độ 4 bit: Dữ liệu được truyền trên 4 đường từ DB4 tới DB7, bit MSB là DB7.
15	A	15 là Catot, điện áp khoảng $U_{ak}=4,2V$
16	K	Chân nối đất của đèn Back light

1.2.3. Sơ đồ khối của HD44780.

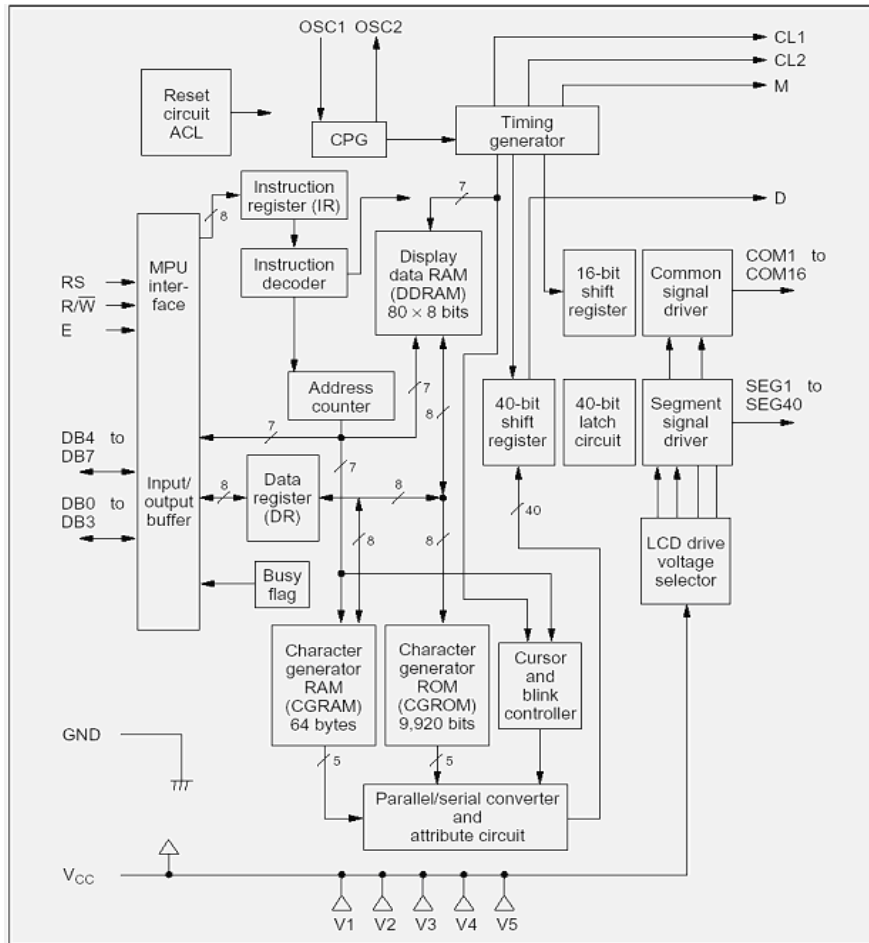
Để hiểu rõ hơn chức năng các chân và hoạt động của chúng, ta tìm hiểu sơ qua chip HD44780 thông qua các khối cơ bản của nó.

*) Các thanh ghi:

Chip HD44780 có 2 thanh ghi 8 bit quan trọng là: Thanh ghi lệnh IR (Instructor Register) và thanh ghi dữ liệu DR (Data Register).

- Thanh ghi IR: Để điều khiển LCD, người dùng phải “ra lệnh” thông qua tám đường bus DB0-DB7. Mỗi lệnh được nhà sản xuất LCD đánh địa chỉ rõ ràng. Người dùng chỉ việc cung cấp địa chỉ lệnh bằng cách nạp vào thanh ghi IR. Nghĩa là, khi ta nạp vào thanh ghi IR một chuỗi 8 bit, chip HD44780 sẽ tra bảng mã lệnh tại địa chỉ mà IR cung cấp và thực hiện lệnh đó.

VD: Lệnh “hiển thị màn hình” có địa chỉ lệnh là 00001100 (DB7...DB0)



Hình 1.22. Sơ đồ khối của HD44780.

- Thanh ghi DR: Thanh ghi DR dùng để chứa dữ liệu 8 bit để ghi vào vùng RAM, DDRAM hoặc CGRAM (ở chế độ ghi) hoặc dùng để chứa dữ liệu từ 2 vùng RAM này gửi ra cho MPU (ở chế độ đọc). Nghĩa là, khi MPU ghi thông tin vào DR, mạch nội bên trong chip sẽ tự động ghi thông tin này vào DDRAM hoặc CGRAM. Hoặc khi thông tin về địa chỉ được ghi vào IR, dữ liệu ở địa chỉ này trong vùng RAM nội của HD44780 sẽ được chuyển ra DR để truyền cho MPU. Vậy bằng cách điều khiển chân RS và R/W chúng ta có thể chuyển qua lại giữa 2 thanh ghi này trong khi giao tiếp với MPU. Bảng 3.2. tóm tắt lại các thiết lập đối với hai chân RS và R/W theo mục đích giao tiếp.

Bảng 3.2. Bảng chức năng chân RS và R/W theo mục đích sử dụng.

RS	RW	Ý nghĩa
0	0	Ghi vào thanh ghi IR để ra lệnh cho LCD (VD: cần display clear, ...)
0	1	Đọc cờ bận ở DB7 và giá trị của bộ đếm địa chỉ ở DB0-DB6
1	0	Ghi vào thanh ghi DR
1	1	Đọc dữ liệu từ DR

*) Cờ báo bận BF (Busy Flag):

Khi thực hiện các hoạt động bên trong chip, mạch nội bên trong cần một khoảng thời gian để hoàn tất. Khi đang thực thi các hoạt động bên trong chip như thế, LCD bỏ qua mọi giao tiếp với bên ngoài và bật cờ BF (thông qua chân DB7 khi có thiết lập RS=0, R/W=1) lên để báo cho MPU biết nó đang “bận”. Dĩ nhiên, khi xong việc, nó sẽ đặt cờ BF lại mức 0.

*) Bộ đếm địa chỉ AC (Address Counter):

Như trong sơ đồ khối, thanh ghi IR không trực tiếp kết nối với vùng RAM (DDRAM và CGRAM) mà thông qua bộ đếm địa chỉ AC. Bộ đếm này lại nối với 2 vùng RAM theo kiểu rẽ nhánh. Khi một địa chỉ lệnh được nạp vào thanh ghi IR, thông tin được nối trực tiếp cho 2 vùng RAM nhưng việc chọn lựa vùng RAM tương tác đã được bao hàm trong mã lệnh. Sau khi ghi vào (hoặc đọc từ) RAM, bộ đếm AC tự động tăng lên (hoặc giảm đi) 1 đơn vị và nội dung của AC được xuất ra cho MPU thông qua DB0-DB6 khi có thiết lập RS=0 và R/W=1 (xem bảng 3.2). Lưu ý: Thời gian cập nhật AC không được tính vào thời gian thực thi lệnh mà được cập nhật sau khi cờ BF lên mức cao (not busy), cho nên khi lập trình hiển thị, bạn phải delay một khoảng tADD khoảng 4 μ S-5 μ S (ngay sau khi BF=1) trước khi nạp dữ liệu mới.

*) Vùng RAM hiển thị DDRAM (Display Data RAM):

Đây là vùng RAM dùng để hiển thị, nghĩa là ứng với một địa chỉ của RAM là một ô kí tự trên màn hình và khi bạn ghi vào vùng RAM này một mã

8 bit, LCD sẽ hiển thị tại vị trí tương ứng trên màn hình một kí tự có mã 8 bit mà bạn đã cung cấp như hình 1.23.

Display position (digit)	1	2	3	4	5	79	80
DDRAM address (hexadecimal)	00	01	02	03	04	4E	4F

Figure 2 1-Line Display

Display position	1	2	3	4	5	39	40
DDRAM address (hexadecimal)	00	01	02	03	04	26	27
	40	41	42	43	44	66	67

Hình 1.23. Mối liên hệ giữa địa chỉ của DDRAM và vị trí hiển thị của LCD.

Vùng RAM này có 80x8 bits nhớ, nghĩa là chứa được 80 kí tự mã 8 bits. Những vùng RAM còn lại không dùng cho hiển thị có thể dùng như vùng RAM đa mục đích. Lưu ý là để truy cập vào DDRAM, ta phải cung cấp địa chỉ cho AC theo mã HEX.

*) Vùng ROM chứa kí tự CGROM (Character Generator ROM):

Vùng ROM này dùng để chứa các mẫu kí tự loại 5x8 hoặc 5x10 điểm ảnh/kí tự, và định địa chỉ bằng 8 bit. Tuy nhiên, nó chỉ có 208 mẫu kí tự 5x8 và 32 mẫu kí tự kiểu 5x10 (tổng cộng là 240 thay vì 256 mẫu kí tự). Người dùng không thể thay đổi vùng ROM này.

Table 2 Example of Correspondence between EPROM Address Data and Character Pattern (5 × 8 Dots)

EPROM Address											Data					
A11	A10	A9	A8	A7	A6	A5	A4	A3	A2	A1	A0	O4	O3	O2	O1	O0
												1	0	0	0	0
												1	0	0	0	0
												1	0	1	1	0
												1	1	0	0	1
												1	0	0	0	1
												1	0	0	0	1
												1	1	1	1	0
0	1	1	0	0	0	1	0					0	0	0	0	0
												1	0	0	0	0
												1	0	0	1	0
												1	0	1	0	0
												1	0	1	1	0
												1	1	0	0	0
												1	1	0	1	0
												1	1	1	0	0
												1	1	1	1	0

Hình 1.24. Mối liên hệ giữa địa chỉ của ROM và dữ liệu tạo mẫu kí tự.

*) Vùng RAM chứa kí tự đồ họa CGRAM (Character Generator RAM):

Như trên bảng mã kí tự, nhà sản xuất dành vùng có địa chỉ byte cao là 0000h để người dùng có thể tạo các mẫu kí tự đồ họa riêng. Tuy nhiên dung lượng vùng này rất hạn chế: Ta chỉ có thể tạo 8 kí tự loại 5x8 điểm ảnh, hoặc 4 kí tự loại 5x10 điểm ảnh. Để ghi vào CGRAM, xem hình 1.24.

1.2.4. Tập lệnh của LCD.

Trước khi tìm hiểu tập lệnh của LCD, sau đây là một vài chú ý khi giao tiếp với LCD:

* Tuy trong sơ đồ khối của LCD có nhiều khối khác nhau, nhưng khi lập trình điều khiển LCD ta chỉ có thể tác động trực tiếp được vào 2 thanh ghi DR và IR thông qua các chân DBx, và ta phải thiết lập chân RS, R/W phù hợp để chuyển qua lại giữa 2 thanh ghi này. (xem bảng 3.2)

Table 5 Relationship between CGRAM Addresses, Character Codes (DDRAM) and Character Patterns (CGRAM Data)

For 5 × 8 dot character patterns

Character Codes (DDRAM data)								CGRAM Address								Character Patterns (CGRAM data)																
7	6	5	4	3	2	1	0	5	4	3	2	1	0	7	6	5	4	3	2	1	0											
High				Low				High				Low				High				Low												
0 0 0 0 * 0 0 0								0 0 0								0 0 0	*	*	*	1	1	1	1	0	Character pattern (1)							
																0 0 1	*	*	*	1	0	0	0	1								
																0 1 0	*	*	*	1	0	0	0	1								
																0 1 1	*	*	*	1	1	1	1	0								
																1 0 0	*	*	*	1	0	1	0	0								
																1 0 1	*	*	*	1	0	0	1	0								
																1 1 0	*	*	*	1	0	0	0	1								
																1 1 1	*	*	*	0	0	0	0	0								
0 0 0 0 * 0 0 1								0 0 1								0 0 0	*	*	*	1	0	0	0	1	Character pattern (2)							
																0 0 1	*	*	*	0	1	0	1	0								
																0 1 0	*	*	*	1	1	1	1	1								
																0 1 1	*	*	*	0	0	1	0	0								
																1 0 0	*	*	*	1	1	1	1	1								
																1 0 1	*	*	*	0	0	1	0	0								
																1 1 0	*	*	*	0	0	1	0	0								
																1 1 1	*	*	*	0	0	0	0	0								
0 0 0 0 * 1 1 1								1 1 1								0 0 0	*	*	*	Cursor position												
																0 0 1	*	*	*													
																1 0 0	*	*	*													
																1 0 1	*	*	*													
																1 1 0	*	*	*													
																1 1 1	*	*	*													
																0 0 0	*	*	*													
																0 0 1	*	*	*													

Hình 1.25. Mối liên hệ giữa địa chỉ của CGRAM, dữ liệu CGARM, và mã kí tự.

* Với mỗi lệnh, LCD cần một khoảng thời gian để hoàn tất, thời gian này có thể khá lâu đối với tốc độ của MPU, nên ta cần kiểm tra cờ BF hoặc đợi (delay) cho LCD thực thi xong lệnh hiện hành mới có thể ra lệnh tiếp theo.

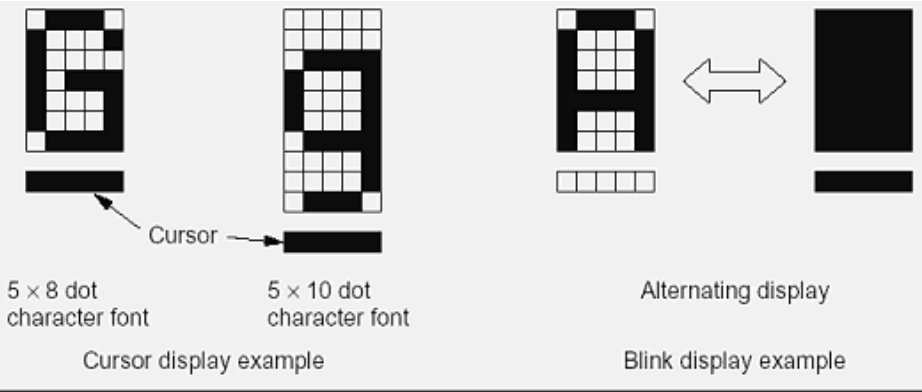
* Địa chỉ của RAM (AC) sẽ tự động tăng (giảm) 1 đơn vị, mỗi khi có lệnh ghi vào RAM. (Điều này giúp chương trình gọn hơn)

* Các lệnh của LCD có thể chia thành 4 nhóm như sau:

- Các lệnh về kiểu hiển thị. VD : Kiểu hiển thị (1 hàng/2 hàng), chiều dài dữ liệu (8 bit/4 bit), ...
- Chỉ định địa chỉ RAM nội.
- Nhóm lệnh truyền dữ liệu trong RAM nội.
- Các lệnh còn lại .

Bảng 3.3. Tập lệnh của LCD.

Tên lệnh	Hoạt động	Thời gian chạy																																																						
Clear Display	<p>Mã lệnh: DBx = DB7 DB6 DB5 DB4 DB3 DB2 DB1 DB0 DBx = 0 0 0 0 0 0 0 1</p> <p>Lệnh Clear Display (xóa hiển thị) sẽ ghi một khoảng trống (mã hiển thị kí tự 20H) vào tất cả ô nhớ trong DDRAM, sau đó trả bộ đếm địa chỉ AC=0, trả lại hiển thị gốc nếu nó bị thay đổi, nghĩa là: Tắt hiển thị, con trỏ dời về góc trái (hàng đầu tiên), chế độ tăng AC.</p>																																																							
Return home	<p>Mã lệnh: DBx = DB7 DB6 DB5 DB4 DB3 DB2 DB1 DB0 DBx = 0 0 0 0 0 0 1 *</p> <p>Lệnh Return home trả bộ đếm địa chỉ AC về 0, trả lại kiểu hiển thị gốc nếu nó bị thay đổi. Nội dung của DDRAM không thay đổi.</p>	1.52 ms																																																						
Entry mode set	<p>Mã lệnh: DBx = DB7 DB6 DB5 DB4 DB3 DB2 DB1 DB0 DBx = 0 0 0 0 0 1 [I/D] [S]</p> <p>I/D: Tăng (I/D=1) hoặc giảm (I/D=0) bộ đếm địa chỉ hiển thị AC 1 đơn vị mỗi khi có hành động ghi hoặc đọc vùng DDRAM. Vị trí con trỏ cũng di chuyển theo sự tăng giảm này.</p> <p>S: Khi S=1 toàn bộ nội dung hiển thị bị dịch sang phải (I/D=0) hoặc sang trái (I/D=1) mỗi khi có hành động ghi vùng DDRAM. Khi S=0: không dịch nội dung hiển thị. Nội dung hiển thị không dịch khi đọc DDRAM hoặc đọc/ghi vùng CGRAM.</p> <div style="display: flex; justify-content: space-around; align-items: flex-start;"> <div style="text-align: center;"> <p>Figure 3 1-Line by 8-Character Display Example</p> <table border="1" style="border-collapse: collapse; margin: 0 auto;"> <tr><td>Display position</td><td>1</td><td>2</td><td>3</td><td>4</td><td>5</td><td>6</td><td>7</td><td>8</td></tr> <tr><td>DDRAM address</td><td>00</td><td>01</td><td>02</td><td>03</td><td>04</td><td>05</td><td>06</td><td>07</td></tr> </table> </div> <div style="text-align: center;"> <p>Figure 5 2-Line by 8-Character Display Example</p> <table border="1" style="border-collapse: collapse; margin: 0 auto;"> <tr><td>Display position</td><td>1</td><td>2</td><td>3</td><td>4</td><td>5</td><td>6</td><td>7</td><td>8</td></tr> <tr><td>DDRAM address</td><td>00</td><td>01</td><td>02</td><td>03</td><td>04</td><td>05</td><td>06</td><td>07</td></tr> <tr><td>For shift left</td><td>01</td><td>02</td><td>03</td><td>04</td><td>05</td><td>06</td><td>07</td><td>08</td></tr> <tr><td>For shift right</td><td>4F</td><td>00</td><td>01</td><td>02</td><td>03</td><td>04</td><td>05</td><td>06</td></tr> </table> </div> </div> <p style="text-align: center;">Hình 3.7. Hoạt động dịch trái và dịch phải nội dung hiển thị</p>	Display position	1	2	3	4	5	6	7	8	DDRAM address	00	01	02	03	04	05	06	07	Display position	1	2	3	4	5	6	7	8	DDRAM address	00	01	02	03	04	05	06	07	For shift left	01	02	03	04	05	06	07	08	For shift right	4F	00	01	02	03	04	05	06	37μs
Display position	1	2	3	4	5	6	7	8																																																
DDRAM address	00	01	02	03	04	05	06	07																																																
Display position	1	2	3	4	5	6	7	8																																																
DDRAM address	00	01	02	03	04	05	06	07																																																
For shift left	01	02	03	04	05	06	07	08																																																
For shift right	4F	00	01	02	03	04	05	06																																																

<p>Display on/off control</p>	<p>Mã lệnh: DBx = DB7 DB6 DB5 DB4 DB3 DB2 DB1 DB0 DBx = 0 0 0 0 1 [D] [C] [B]</p> <p>D: Hiện thị màn hình khi D=1 và ngược lại. Khi tắt hiển thị, nội dung DDRAM không thay đổi. C: Hiện thị con trỏ khi C=1 và ngược lại. Vị trí và hình dạng con trỏ, xem hình 3.8. B: Nhấp nháy kí tự tại vị trí con trỏ khi B=1 và ngược lại. Xem thêm hình 8. về kiểu nhấp nháy. Chu kì nhấp nháy khoảng 409,6ms khi mạch dao động nội LCD là 250kHz.</p>  <p style="text-align: center;">Figure 12 Cursor and Blinking</p> <p style="text-align: center;">Hình 3.8. Kiểu con, kiểu kí tự và nhấp nháy kí tự</p>	<p>37μs</p>															
<p>Cursor or display shift</p>	<p>Mã lệnh: DBx = DB7 DB6 DB5 DB4 DB3 DB2 DB1 DB0 DBx = 0 0 0 1 [S/C] [R/L] * *</p> <p>Lệnh Cursor or display shift dịch chuyển con trỏ hay dữ liệu hiển thị sang trái mà không cần hành động ghi/đọc dữ liệu. Khi hiển thị kiểu 2 dòng, con trỏ sẽ nhảy xuống dòng dưới khi dịch qua vị trí thứ 40 của hàng đầu tiên. Dữ liệu hàng đầu và hàng 2 dịch cùng một lúc. Chi tiết sử dụng xem bảng sau:</p> <table border="1" data-bbox="395 1666 1241 1899"> <thead> <tr> <th>S/C</th> <th>R/L</th> <th>Hoạt động</th> </tr> </thead> <tbody> <tr> <td>0</td> <td>0</td> <td>Dịch vị trí con trỏ sang trái (Nghĩa là giảm AC một đơn vị).</td> </tr> <tr> <td>0</td> <td>1</td> <td>Dịch vị trí con trỏ sang phải (Tăng AC lên 1 đơn vị).</td> </tr> <tr> <td>1</td> <td>0</td> <td>Dịch toàn bộ nội dung hiển thị sang trái, con trỏ cũng dịch theo.</td> </tr> <tr> <td>1</td> <td>1</td> <td>Dịch toàn bộ nội dung hiển thị sang phải, con trỏ cũng dịch theo.</td> </tr> </tbody> </table>	S/C	R/L	Hoạt động	0	0	Dịch vị trí con trỏ sang trái (Nghĩa là giảm AC một đơn vị).	0	1	Dịch vị trí con trỏ sang phải (Tăng AC lên 1 đơn vị).	1	0	Dịch toàn bộ nội dung hiển thị sang trái, con trỏ cũng dịch theo.	1	1	Dịch toàn bộ nội dung hiển thị sang phải, con trỏ cũng dịch theo.	<p>37μs</p>
S/C	R/L	Hoạt động															
0	0	Dịch vị trí con trỏ sang trái (Nghĩa là giảm AC một đơn vị).															
0	1	Dịch vị trí con trỏ sang phải (Tăng AC lên 1 đơn vị).															
1	0	Dịch toàn bộ nội dung hiển thị sang trái, con trỏ cũng dịch theo.															
1	1	Dịch toàn bộ nội dung hiển thị sang phải, con trỏ cũng dịch theo.															
<p>Function</p>	<p>Mã lệnh: DBx = DB7 DB6 DB5 DB4 DB3 DB2 DB1 DB0</p>																

set	<p style="text-align: center;">DBx = 0 0 1 [DL] [N] [F] * *</p> <p>DL: Khi DL=1, LCD giao tiếp với MPU bằng giao thức 8 bit (từ bit DB7 đến DB0). Ngược lại, giao thức giao tiếp là 4 bit (từ bit DB7 đến bit DB0). Khi chọn giao thức 4 bit, dữ liệu được truyền/nhận 2 lần liên tiếp với 4 bit cao gửi/nhận trước, 4 bit thấp gửi/nhận sau.</p> <p>N: Thiết lập số hàng hiển thị. Khi N=0: hiển thị 1 hàng, N=1: hiển thị 2 hàng.</p> <p>F: Thiết lập kiểu kí tự. Khi F=0: kiểu kí tự 5x8 điểm ảnh, F=1: kiểu kí tự 5x10 điểm ảnh.</p> <p>* Chú ý:</p> <ul style="list-style-type: none"> • Chỉ thực hiện thay đổi Function set ở đầu chương trình. Và sau khi được thực thi 1 lần, lệnh thay đổi Function set không được LCD chấp nhận nữa ngoại trừ thiết lập chuyển đổi giao thức giao tiếp. • Không thể hiển thị kiểu kí tự 5x10 điểm ảnh ở kiểu hiển thị 2 hàng. 	37μs
Set CGRAM address	<p>Mã lệnh: DBx = DB7 DB6 DB5 DB4 DB3 DB2 DB1 DB0 DBx= 0 1 [ACG][ACG][ACG][ACG][ACG][ACG]</p> <p>Lệnh này ghi vào AC địa chỉ của CGRAM. Kí hiệu [ACG] chỉ 1 bit của chuỗi dữ liệu 6 bit. Ngay sau lệnh này là lệnh đọc/ghi dữ liệu từ CGRAM tại địa chỉ đã được chỉ định.</p>	37μs
Set DDRAM address	<p>Mã lệnh: DBx = DB7 DB6 DB5 DB4 DB3 DB2 DB1 DB0 DBx = 1 [AD] [AD] [AD] [AD] [AD] [AD] [AD]</p> <p>Lệnh này ghi vào AC địa chỉ của DDRAM, dùng khi cần thiết lập tọa độ hiển thị mong muốn. Ngay sau lệnh này là lệnh đọc/ghi dữ liệu từ DDRAM tại địa chỉ đã được chỉ định. Khi ở chế độ hiển thị 1 hàng, địa chỉ có thể từ 00H đến 4FH. Khi ở chế độ hiển thị 2 hàng, địa chỉ từ 00h đến 27H cho hàng thứ nhất, và từ 40h đến 67h cho hàng thứ 2.</p>	37μs
Read BF and address	<p>Mã lệnh: DBx = DB7 DB6 DB5 DB4 DB3 DB2 DB1 DB0 DBx= [BF] [AC] [AC] [AC] [AC] [AC] [AC] [AC]</p> <p>(RS=0, R/W=1)</p> <p>Như đã đề cập trước đây, khi cờ BF bật, LCD đang làm việc và lệnh tiếp theo (nếu có) sẽ bị bỏ qua nếu cờ BF chưa về mức thấp. Cho nên, khi lập trình điều khiển, bạn phải kiểm tra cờ BF trước khi ghi dữ liệu vào LCD. Khi đọc cờ BF, giá trị của AC cũng được xuất ra các bit [AC]. Nó là địa chỉ của CG hay DDRAM là tùy thuộc vào lệnh trước đó.</p>	0μs
Write	Mã lệnh: DBx = DB7 DB6 DB5 DB4 DB3 DB2 DB1 DB0	

ata to CG or DDRAM	$DBx = [Write\ data] \quad (RS=1, R/W=0)$ Khi thiết lập $RS=1, R/W=0$, dữ liệu cần ghi được đưa vào các chân DBx từ mạch ngoài sẽ được LCD chuyển vào trong LCD tại địa chỉ được xác định từ lệnh ghi địa chỉ trước đó (lệnh ghi địa chỉ cũng xác định luôn vùng RAM cần ghi). Sau khi ghi, bộ đếm địa chỉ AC tự động tăng/giảm 1 tùy theo thiết lập Entry mode. Lưu ý là thời gian cập nhật AC không tính vào thời gian thực thi lệnh.	$37\mu s$ t_{ADD} $4\mu s$
Read data from CG or DDRAM	Mã lệnh: $DBx = DB7\ DB6\ DB5\ DB4\ DB3\ DB2\ DB1\ DB0$ $DBx = [Read\ data] \quad (RS=1, R/W=1)$ Khi thiết lập $RS=1, R/W=1$, dữ liệu từ CG/DDRAM được chuyển ra MPU thông qua các chân DBx (địa chỉ và vùng RAM đã được xác định bằng lệnh ghi địa chỉ trước đó). Sau khi đọc, AC tự động tăng/giảm 1 tùy theo thiết lập Entry mode, tuy nhiên nội dung hiển thị không bị dịch bất chấp chế độ Entry mode.	$37\mu s$ t_{ADD} $4\mu s$

1.2.5. Đặc tính của các chân giao tiếp.

LCD sẽ bị hỏng nghiêm trọng, hoặc hoạt động sai lệch nếu bạn vi phạm khoảng đặc tính điện sau đây:

Bảng 3.4. Đặc tính điện làm việc điển hình.

Chân cấp nguồn (Vcc-GND)	Min:-0.3V , Max+7V
Các chân ngõ vào (DBx,E,...)	Min:-0.3V , Max:(Vcc+0.3V)
Nhiệt độ hoạt động	Min:-30C , Max:+75C
Nhiệt độ bảo quản	Min:-55C , Max:+125C

Đặc tính điện làm việc điển hình: (Đo trong điều kiện hoạt động $V_{cc} = 4.5V$ đến $5.5V$, $T = -30$ đến $+75C$).

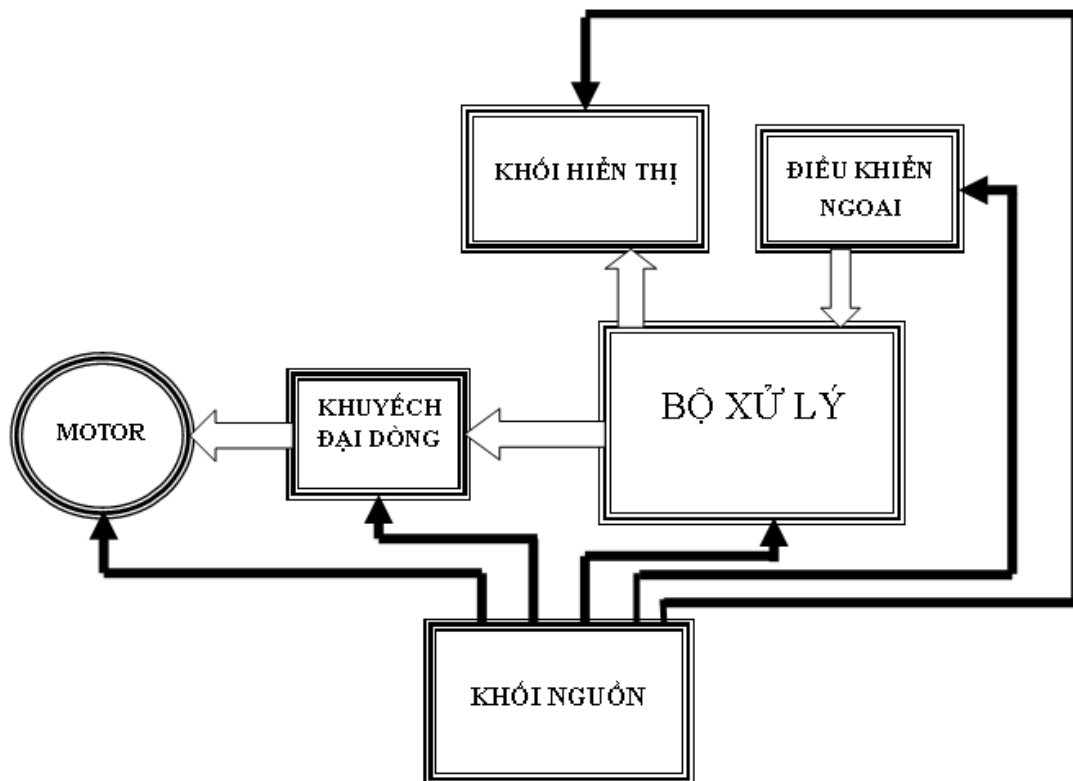
Bảng 3.5. Miền làm việc bình thường.

Chân cấp nguồn Vcc-GND	2.7V đến 5.5V
Điện áp vào mức cao V_{IH}	2.2V đến Vcc
Điện áp vào mức thấp V_{IL}	-0.3V đến 0.6V
Điện áp ra mức cao (DB0-DB7)	Min 2.4V (khi $I_{OH} = -0.205mA$)
Điện áp ra mức thấp (DB0-DB7)	Max 0.4V (khi $I_{OL} = 1.2mA$)
Dòng điện ngõ vào (input leakage current) I_{LI}	-1uA đến 1uA (khi $V_{IN} = 0$ đến Vcc)
Dòng điện cấp nguồn I_{CC}	350uA(typ.) đến 600uA
Tần số dao động nội f_{OSC}	190kHz đến 350kHz (điển hình là 270kHz)

Chương 2: THIẾT KẾ HỆ THỐNG ĐIỀU KHIỂN MOTOR

2.1. Sơ đồ khối

Với yêu cầu của đề tài là thiết kế hệ thống điều khiển mở cửa bằng mật mã bằng cách sử dụng pic vi điều khiển để điều khiển chiều động cơ, em xin đưa ra sơ đồ khối như sau:

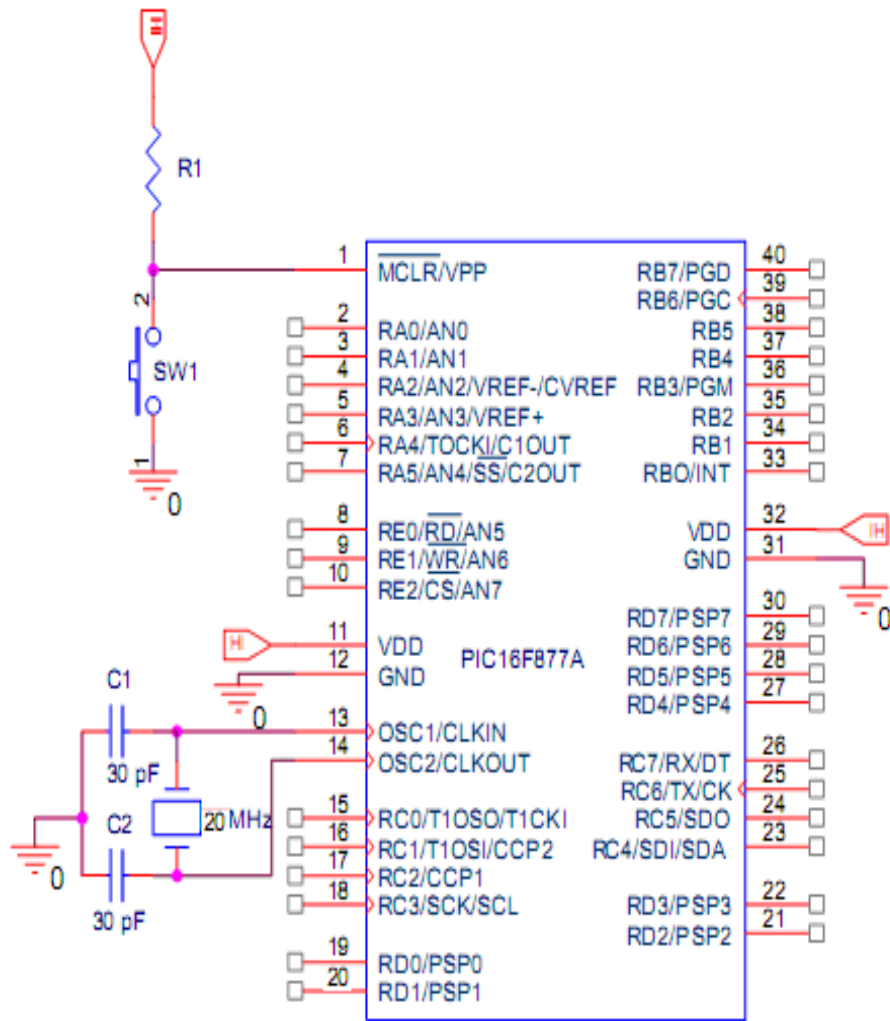


Hình 2.1. Sơ đồ khối hệ thống điều khiển đóng mở cửa

2.2. Thiết kế các khối

2.2.1. Bộ xử lý

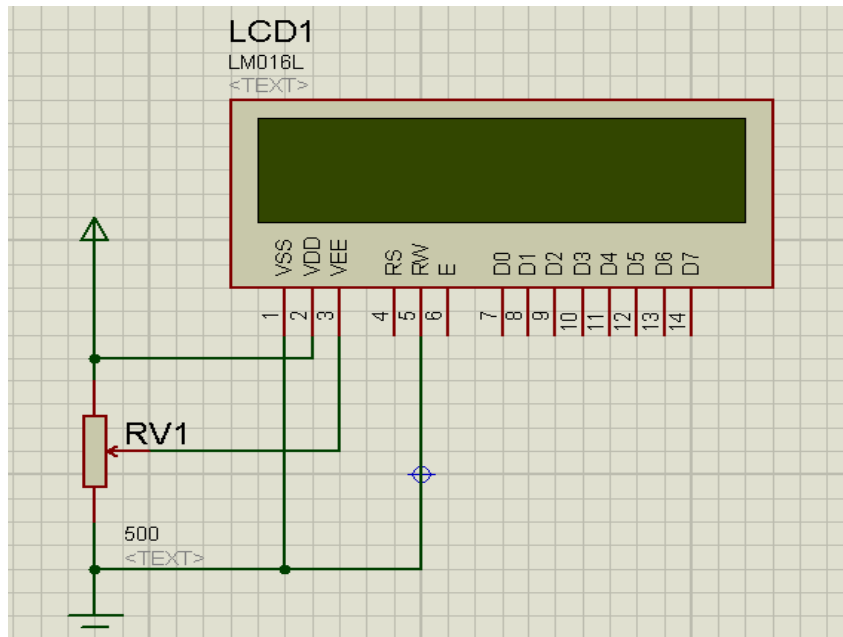
Bộ xử lý làm nhiệm vụ nhận tín hiệu điều khiển từ bàn phím ngoài (ma trận phím 4x4), điều khiển cho động cơ quay theo chiều thuận hay chiều ngược, ở đây em sử dụng vi điều khiển PIC16F877A. Đây là vi điều khiển có 40 chân, với 5 cổng vào ra là Port A (RA0÷RA5), Port B (RB0÷RB7), Port C (RC0÷RC7), Port D (RD0÷RD7), Port E (RE0÷RE2). Nó có 8K Flash ROM và 368 Bytes RAM.



Hình 2.2. Sơ đồ nguyên lý của PIC16F877A trong mạch

2.2.2. Khởi hiển thị

Để thuận tiện cho việc hiển thị kí tự và chế độ cài đặt trạng thái điều khiển, ở em đây sử dụng LCD_DM 1602A.



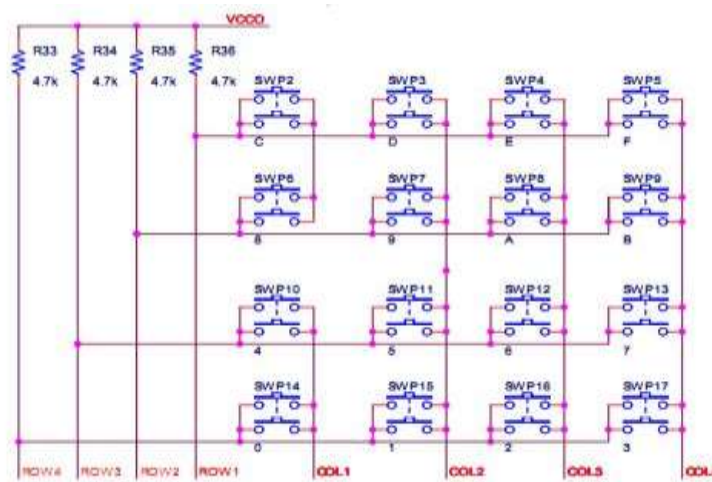
Hình 2.3. Sơ đồ nguyên lý của LCD1602

LCD1602 là loại 2 dòng, 16 kí tự, sử dụng nguồn nuôi thấp (từ 2, 5 đến 5V). Có thể hoạt động ở hai chế độ 4 bit hoặc 8 bit (trong đề tài này em sử dụng chế độ 4 bit).

2.2.3. Ma trận phím .

2.2.3.1 Cấu tạo .

Về cơ bản thì ma trận phím dựa trên một ma trận hàng và cột. Sự kiện một phím được nhấn hoặc thả có thể được ghi lại bằng cách điều khiển và đọc trạng thái của các hàng và các cột trong ma trận phím.



Hình 2.4. Sơ đồ nguyên lý ma trận phím 4x4.

2.2.3.2. nguyên lý quét phím.

Thuật toán quét phím được sử dụng là lần lượt tìm hàng và tìm cột (hoặc ngược lại) Khi tìm hàng, các hàng sẽ được đặt làm đầu vào, các cột được đặt làm đầu ra mức thấp. Sau đó kiểm tra các hàng xem có hàng nào ở mức thấp hay không (có phím nào bấm gây ra nối với cột hay không)? Sau khi xác định được hàng sẽ đặt các cột làm đầu vào, hàng vừa tìm được làm đầu ra mức thấp. Việc kiểm tra được tiến hành với các cột. Sau khi xác định được hàng và cột sẽ suy ra phím được bấm.

+ Đầu tiên cho các hàng ở mức 1 và các cột ở mức 0

+ Kiểm tra xem nào hàng nào được nhấn (khi nút được nhấn) tức là được nhận tín hiệu từ các cột

+ Nếu mà có 1 hàng bất kỳ được nhấn sau đó ta lại chuyển giá trị 0 vào hàng được nhấn và giá trị 1 vào các cột. Khi đó cột lại nhận nhiệm vụ làm tín hiệu vào. Nếu mà 1 trong các cột được nhấn (khi nút được nhấn) trong các trường hợp này thì cho ra các giá trị tương ứng.

Ma trận phím 4x4 gồm có 16 phím nhấn, trong đó em sử dụng các phím nhấn từ 0 → 9 làm các phím số dùng để nhập mật khẩu hay thay đổi mật khẩu.



Hình 2.5. Hình ảnh thực của ma trận phím

Các phím A,B,C là các phím dùng để thiết lập chế độ cài đặt và điều khiển cho động cơ hoạt động theo chiều thuận hay ngược.

2.2.5. Khối công suất động cơ.

2.2.5.1. Motor DC

Ở đây em sử dụng loại động cơ DC, như hình 2.5



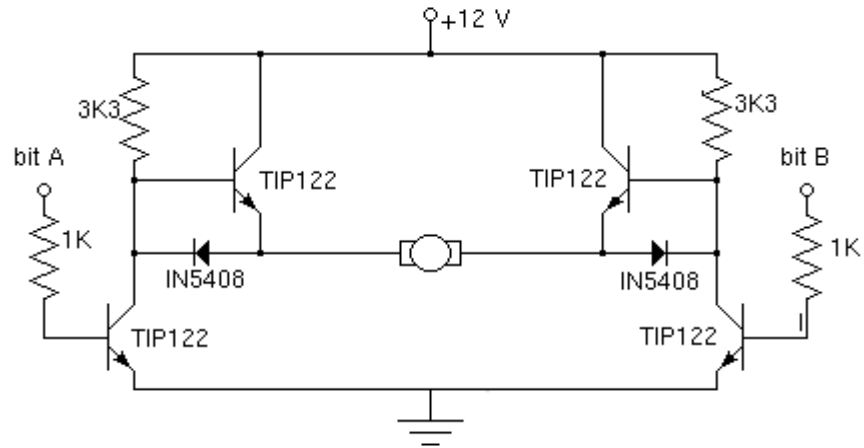
Hình 2.6. Động cơ DC

2.2.5.2. Mạch cầu H

Mạch cầu H là được gọi là mạch cầu H vì nó được cấu tạo bởi 4 transistor hay là Fet. Đôi khi mạch cầu H cũng được cấu tạo bởi 2 transistor hay Fet. Tác dụng của transistor và Fet là các van đóng mở dẫn dòng điện từ nguồn xuống tải với công suất lớn. Tín hiệu điều khiển các van là tín hiệu nhỏ (điện áp hay dòng điện) và cho dẫn dòng và điện áp lớn để cung cấp cho tải.

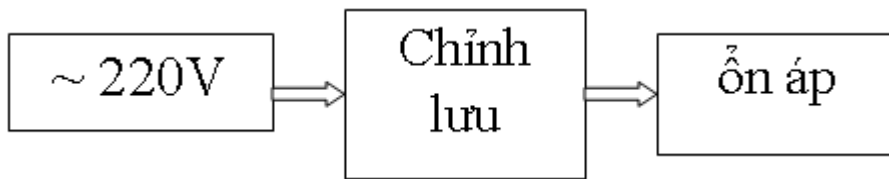
Tín hiệu đầu ra của vi điều khiển là nhỏ hơn 5V (do các điều chế PWM) mà điều khiển động cơ cần dòng điện và điện áp lớn. Các van điều khiển hay các chân điều khiển chỉ cần tín hiệu nhỏ (Điện áp hay dòng điện) là mở khóa (Transistor) dẫn dòng cho tải.

Mạch cầu H có thể đảo chiều dòng điện qua tải nên thế nó hay được dùng trong các mạch điều khiển động cơ DC . Đối với mạch điều khiển động cơ thì mạch cầu H có thể đảo chiều động cơ quá là đơn giản. Chỉ cần mở khóa các van đúng chiều mà mình muốn. Nên thế trong đề tài nay em sử dụng mạch cầu H và các Transistor được sử dụng là Tip 122 .



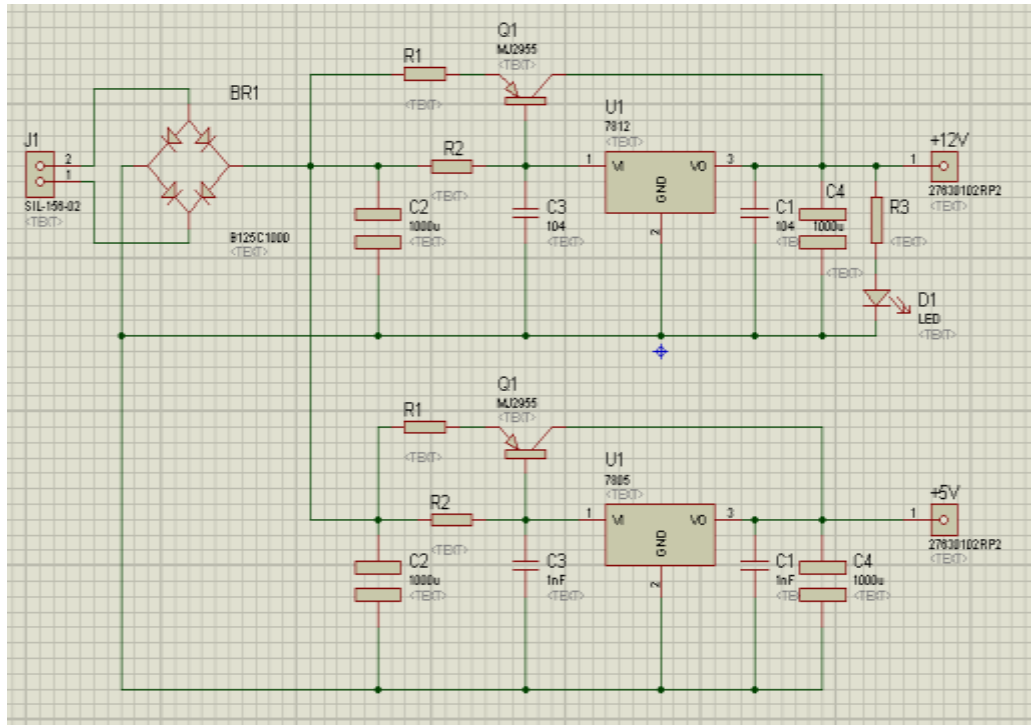
Hình 2.7. Sơ đồ nguyên lý mạch cầu H

2.2.5. Khối nguồn



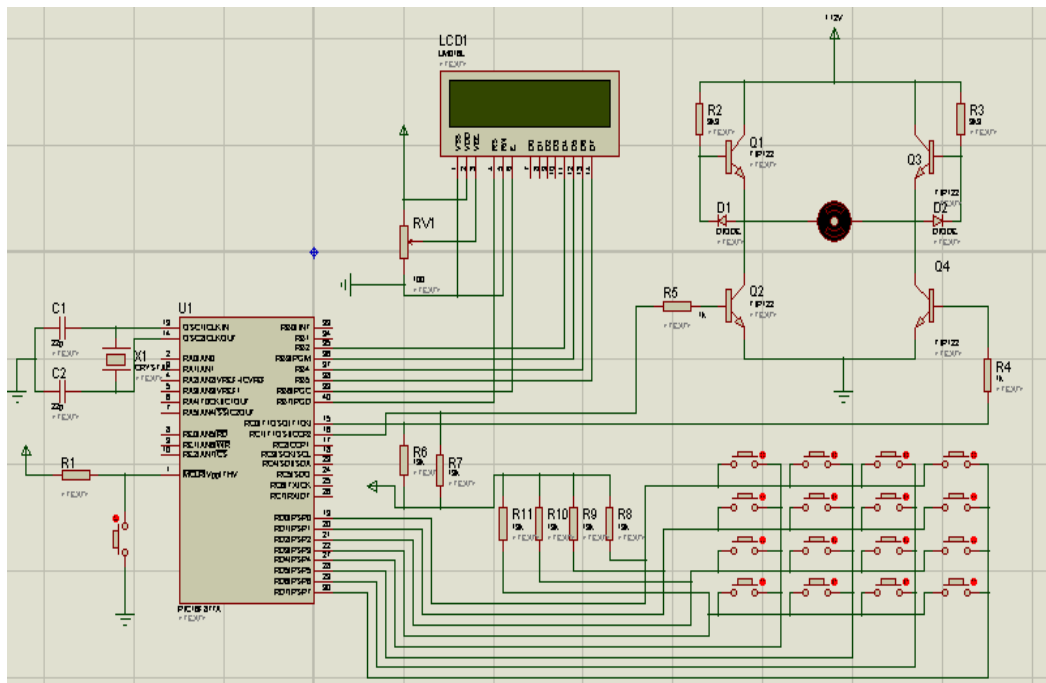
Cung cấp nguồn nuôi cho toàn bộ hệ thống.

Ở đây bộ ổn áp dùng IC 7805 , 7812 để tạo nguồn +5V cung cấp nguồn cho mạch vi điều khiển và IC 7812 để tạo nguồn +12V cấp cấp và ổn định cho mạch khuếch đại động cơ. Sơ đồ nguyên lý như hình 2.6.



Hình 2.8. Sơ đồ nguyên lý của bộ ổn áp

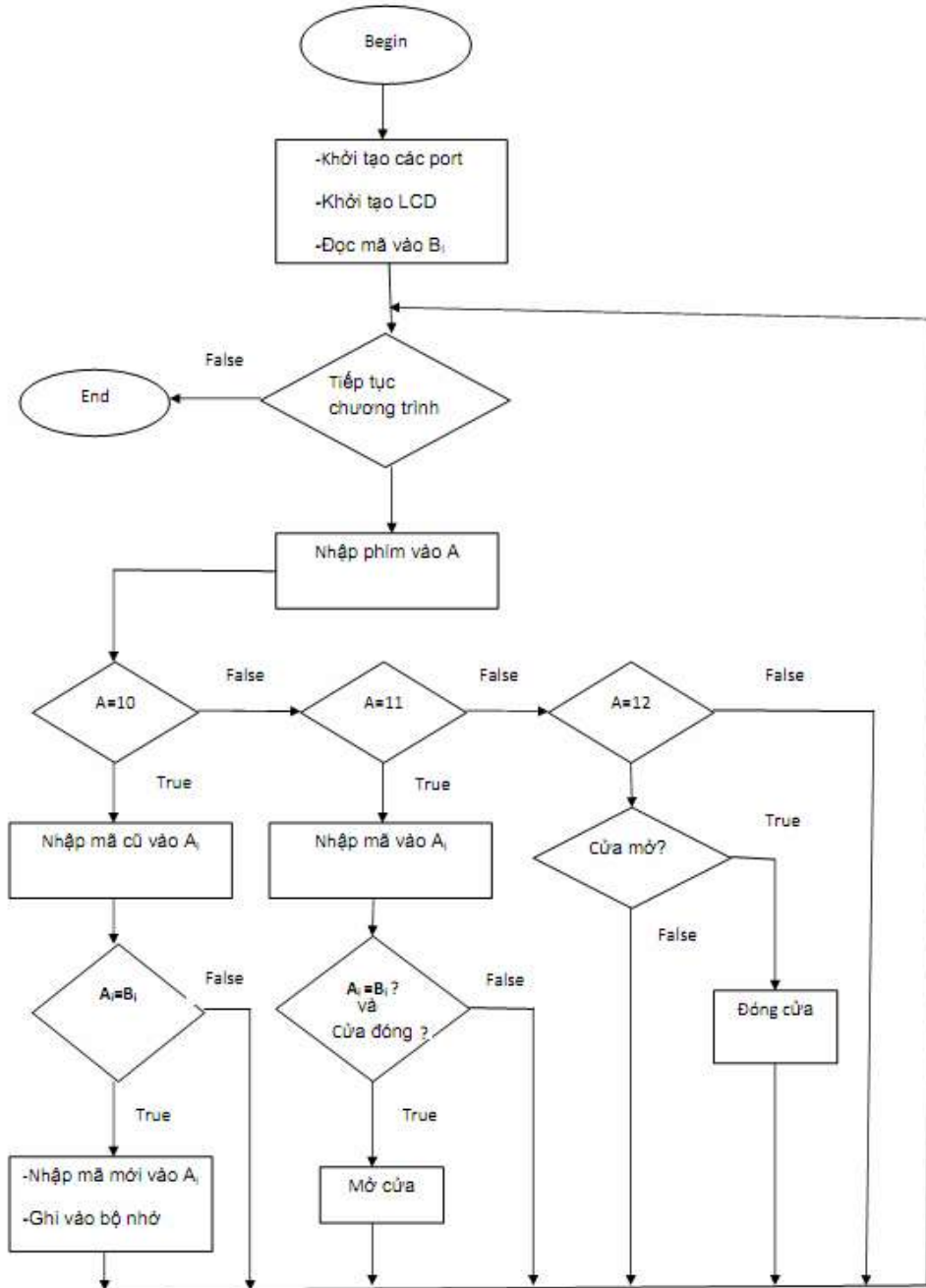
2.3. Sơ đồ mạch hệ thống



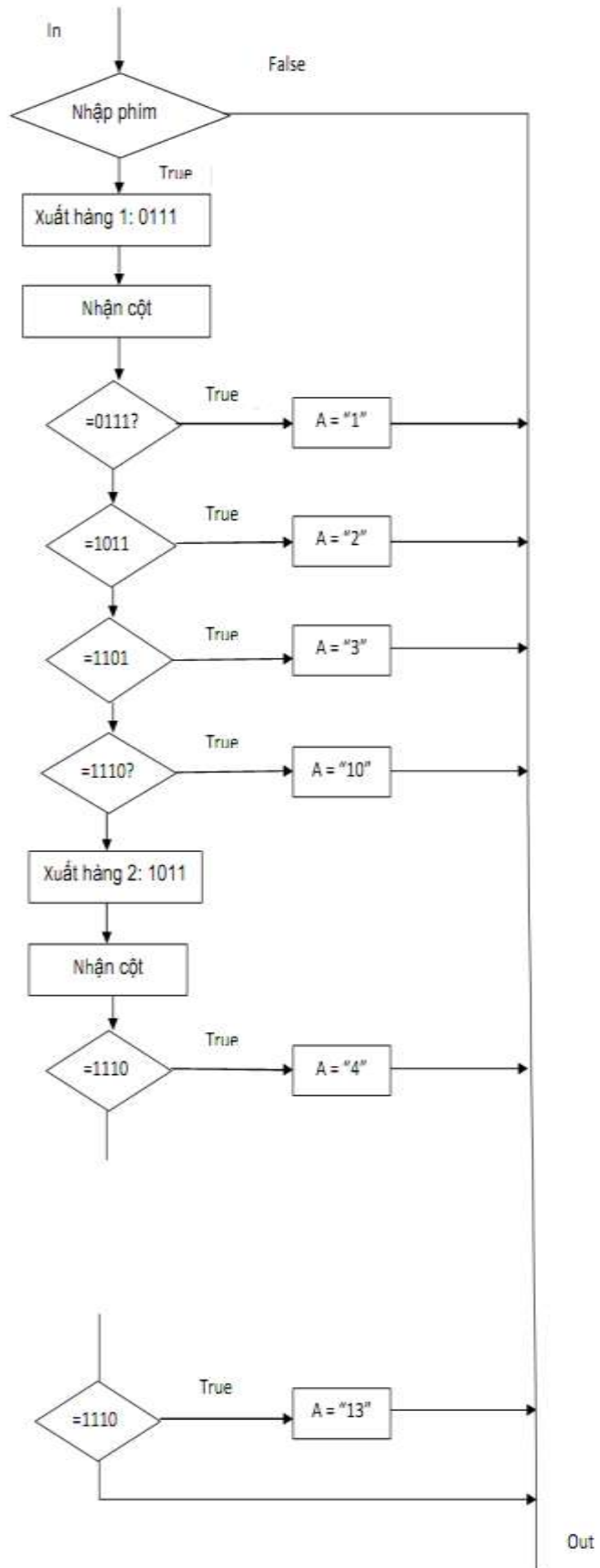
Hình 2.9. Sơ đồ nguyên lý mạch điều khiển Motor

Chương 3: PHẦN MỀM ĐIỀU KHIỂN

3.1. Lưu đồ thuật toán



Lưu đồ quét phím .



3.2. Chương trình

```
#include <16f877A.h>
#include <math.h>
#FUSES NOWDT, HS, NOPUT, NOPROTECT, NODEBUG, NOBROWNOUT,
NOLVP, NOCPD, NOWRT
#use delay(clock=20000000)
#include<lcd_ct.c>
#define Add 0
#define TM 5000
#define TK 1000
//=====
void Scan();
void OpenM();
void CloseM();
void Setting();
void InputNumber();

int8 A=0,A1=0,A2=0,A3=0,A4=0;
int8 B1,B2,B3,B4;
//unsigned int8 VarIn, VarOut;
int8 State;
//int8 Count;
int i;

void main()
{
    set_tris_D(0xF0);
    set_tris_C(0x00);
    lcd_init();
    output_C(0x00);
    delay_ms(100);
//write to eeprom
    if(read_eeprom(Add)!=128)
        {
            write_eeprom(Add+1,1);
            write_eeprom(Add+2,2);
            write_eeprom(Add+3,3);
            write_eeprom(Add+4,4);
```

```
    }
//read from eeprom
    B1=read_eeprom(Add+1);
    B2=read_eeprom(Add+2);
    B3=read_eeprom(Add+3);
    B4=read_eeprom(Add+4);
while(1)
{
    lcd_putcmd(0x80);
    printf(lcd_putchar,"Auto Open Sys ");
    lcd_putcmd(0xC0);
    printf(lcd_putchar,"          ");
    delay_ms(TK);

    Scan();
    switch(A)
    {
    case 10: lcd_putcmd(0x80);
            printf(lcd_putchar,"Setting Number ");
            delay_ms(TK);
            lcd_putcmd(0x80);
            printf(lcd_putchar,"Put Old Number ");
            delay_ms(TK);
            InputNumber();
            if((A1==B1)&&(A2==B2)&&(A3==B3)&&(A4==B4))
            {
                lcd_putcmd(0x80);
                printf(lcd_putchar,"Put New Number ");
                lcd_putcmd(0xC0);
                printf(lcd_putchar,"          ");
                Setting();
            }
            break;
    case 11: lcd_putcmd(0x80);
            printf(lcd_putchar,"Closing ..... ");
            CloseM();
            break;
    case 12: lcd_putcmd(0x80);
```

```
printf(lcd_putchar,"Put Num to Open ");
delay_ms(TK);
InputNumber();
if((A1==B1)&&(A2==B2)&&(A3==B3)&&(A4==B4))
{
    lcd_putcmd(0x80);
    printf(lcd_putchar,"Open ..... ");
    OpenM();
}
break;
}
}
}
//-----
void Scan()
{

while(1)
{
    State=0;
    output_D(0xF7);
    delay_us(50);

switch(input_D()&0xF0)
{
    case 0x70: A=1;
        State=1;
        break;
    case 0xB0: A=2;
        State=1;
        break;
    case 0xD0: A=3;
        State=1;
        break;
    case 0xE0: A=10;
        State=1;
        break;
}
}
```

```
output_D(0xFB);  
delay_us(50);
```

```
switch(input_D() & 0xF0)  
{  
    case 0x70: A=4;  
        State=1;  
        break;  
    case 0xB0: A=5;  
        State=1;  
        break;  
    case 0xD0: A=6;  
        State=1;  
        break;  
    case 0xE0: A=11;  
        State=1;  
        break;  
}
```

```
output_D(0xFD);  
delay_us(50);
```

```
switch(input_D() & 0xF0)  
{  
    case 0x70: A=7;  
        State=1;  
        break;  
    case 0xB0: A=8;  
        State=1;  
        break;  
    case 0xD0: A=9;  
        State=1;  
        break;  
    case 0xE0: A=12;  
        State=1;  
        break;  
}
```

```
output_D(0xFE);
delay_us(50);

switch(input_D() & 0xF0)
{
    case 0x70: A=14;
        State=1;
        break;
    case 0xB0: A=0;
        State=1;
        break;
    case 0xD0: A=15;
        State=1;
        break;
    case 0xE0: A=13;
        State=1;
        break;
}

if(State)
{
    LCD_putchar(A+0x30);
    delay_ms(100);
    break;
}

}

}

//-----
void OpenM()
{
    for(i=0; i<250; i=i+1)
    {
        output_C(0x01);
        delay_ms(20);
        output_C(0x00);
    }
}
```

```
    delay_ms(20);  
  }  
}
```

```
//-----
```

```
void CloseM()  
{  
  for(i=0;i<250;i=i+1)  
  {  
    output_C(0x02);  
    delay_ms(20);  
    output_C(0x00);  
    delay_ms(20);  
  }  
}
```

```
//-----
```

```
void Setting()  
{  
  LCD_putcmd(0xC6);  
  Scan();  
  B1=A;  
  delay_ms(TK);  
  
  LCD_putcmd(0xC8);  
  Scan();  
  B2=A;  
  delay_ms(TK);  
  
  LCD_putcmd(0xCA);  
  Scan();  
  B3=A;  
  delay_ms(TK);  
  
  LCD_putcmd(0xCC);  
  Scan();  
  B4=A;  
  delay_ms(TK);
```

```
//write to eeprom
write_eeprom(Add,128);
write_eeprom(Add+1,B1);
write_eeprom(Add+2,B2);
write_eeprom(Add+3,B3);
write_eeprom(Add+4,B4);

}
//-----
void InputNumber()
{
LCD_putcmd(0xC6);
Scan();
A1=A;
delay_ms(TK);

LCD_putcmd(0xC8);
Scan();
A2=A;
delay_ms(TK);

LCD_putcmd(0xCA);
Scan();
A3=A;
delay_ms(TK);

LCD_putcmd(0xCC);
Scan();
A4=A;
delay_ms(TK);
}
```


KẾT LUẬN

Đề tài **MỞ CỬA BẰNG MẬT MÃ** không phải là đề tài mới mẻ và cũng không phải đề tài lớn, nhưng qua đó em đã bổ xung được rất nhiều kinh nghiệm và kiến thức có ích cho bản thân như.

- Hiểu được phương pháp điều khiển thông qua vi điều khiển.
- Tìm hiểu thêm được về Motor .
- Xây dựng Hệ thống điều khiển cơ bản.

Và qua thời gian làm đề tài cùng với sự hướng dẫn chỉ bảo của thầy **NGUYỄN VĂN DƯƠNG** em đã học hỏi, rèn luyện được tinh thần làm việc nghiêm túc và cách thức tìm tòi, học hỏi, nghiên cứu các kiến thức mới không ngừng để bổ xung tích lũy kiến thức cho mình.

Trong quá trình làm đồ án tốt nghiệp, do sự hạn chế về thời gian, tài liệu và trình độ có hạn nên không tránh khỏi có thiếu sót. Em rất mong được sự đóng góp ý kiến của thầy cô trong hội đồng và các bạn để đồ án tốt nghiệp của em được hoàn thiện hơn.

Sau cùng em xin trân thành bày tỏ lòng biết ơn của mình đối với thầy **NGUYỄN VĂN DƯƠNG** và các thầy cô trong khoa đã giúp đỡ em hoàn thành đồ án tốt nghiệp này.

Ngày 1 tháng 11 năm 2010

Sinh viên

Phạm Quý Nghiêm

TÀI LIỆU THAM KHẢO

1. Nguyễn Tăng Cường, Phan Quốc Thắng, Cấu trúc và lập trình họ Vi Điều khiển 8051, Nhà xuất bản khoa học và Kỹ Thuật.
2. Nguyễn Mạnh Giang, Cấu trúc, lập trình ghép nối và ứng dụng của Vi Điều Khiển, nhà xuất bản Lao Động – Xã Hội.
3. Phạm Minh Hà(2004), *Kỹ thuật mạch điện tử*, Nhà xuất bản Khoa học và kỹ thuật.
4. Ngô Diệm Tập, Vi Điều Khiển trong đo lường và điều khiển tự động, Nhà xuất bản Khoa Học và Kỹ Thuật, Hà Nội.
5. Họ Vi Điều Khiển 8051, Tống Văn ON, nhà Xuất bản Lao Động và Xã Hội.
6. Các bạn có thể truy cập các trang Web rất hay của Việt Nam như :
www.dientuvietnam.net
www.picvietnam.com
www.dientuvienthong.net
www.vagam.dieukhien.net
www.duyphi.phpnet.us/index.htm