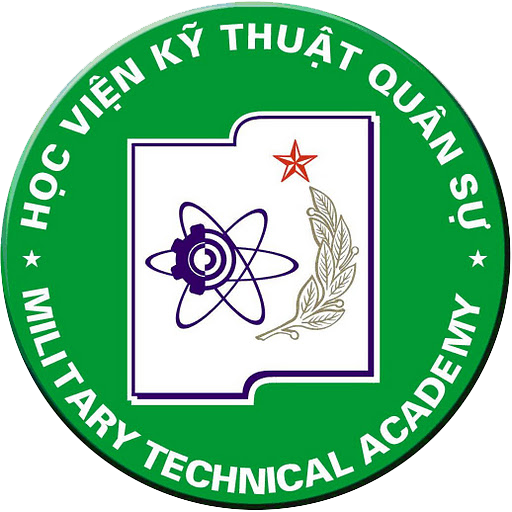
**HỌC VIỆN KỸ THUẬT QUÂN SỰ**

KHOA KĨ THUẬT ĐIỀU KHIỂN

----------------------



**ĐỒ ÁN XỬ LÝ TIN**

**Đề tài** : Thiết kế hệ thống đo nhiệt độ , độ ẩm của nhà kính thu nhỏ , hiển thị lên LCD và điện thoại qua đó điều khiển các thông số của nhà kính , truyền dẫn qua Bluetooth

**Sinh viên gồm :**

1. Chu Phúc Công - 17150281

2. Đỗ Thị Hà - 17150295

3. Đoàn Quang Trường - 17150379

**Lớp:** ĐTVT16A

**Giáo viên hướng dẫn:** Lê Hải Nam

**Hà Nội, 11/2021**

**GIỚI THIỆU CHUNG**

Xã hội ngày càng phát triển về mặt khoa học công nghệ , máy móc đang dần thay thế cho sức lao động của con người . Nhu cầu sử dụng các hệ thống thiết bị thông minh ngày càng phổ biến . Vì vậy nhóm em đã thiết kế hệ thống nhà kính thu nhỏ sử dụng vi điều khiển PIC16f877a

Đề tài gồm 3 chương:

Chương I : Tổng quan về hệ thống.

Chương II : Thiết kế phần cứng.

Chương III: Mô phỏng và thiết kế phần mềm.

1. **Tổng quan về hệ thống**
2. Các chế độ hoạt động

a.Chế độ bình thường

- Đo nhiệt độ - độ ẩm , hiển thị lên LCD và điều khiển qua nút bấm

b.Chế độ

* **2, Các linh kiện sử dụng**
* Vi điều khiển PIC16F887A
* Cảm biến nhiệt độ - độ ẩm DHT22
* Modul bluetooth HC-05
* Led đơn (tín hiệu đèn)
* Một số các linh kiện khác: nút bấm, traristor npn,dây dẫn,tụ điện,điện trở,relay,...

**3, Nguyên lý hoạt động**

Mạch đo nhiệt độ - độ ẩm dựa trên vi điều khiển PIC16F877A thông qua cảm biến độ ẩm – nhiệt độ DHT22 . Khi tín hiệu nhệt độ - độ ẩm từ modul cảm biến DHT22 trả về vi điều khiển PIC16F877A , các giá trị hiển thị trên LCD và màn hình điện thoại từ đó cảm biến so sánh các giá trị đo được với các ngưỡng ta đặt ban đầu . Nếu nhiệt độ đo được nhỏ hơn ngưỡng dưới thì đèn sợi đốt bật , khi lớn hơn ngưỡng trên thì động cơ quạt bật . Nếu độ ẩm đo được nhỏ hơn ngưỡng dưới thì động cơ phun sương bật , khi lớn hơn ngưỡng trên thì đèn báo hiệu bật .

Trong trường hợp hệ thống lỗi , thì t chuyển sang chế độ điều khiển thủ công qua công tác và nút nhấn .

**II, Thiết kế phần cứng**

LCD

(Hiện thoại )

Sơ đồ khối

Động cơ quạt gió

Khối điều khiển (PIC16F877A)

Điện thoại

(Hiển thị )

Modul Bluetooth HC-05

Động cơ phun sương

Đèn sợi đốt

DHT22

Đèn báo hiệu

Nguồn cấp

**Vi điều khiển PIC16F877A**

**A.Sơ lược về vi điều khiển PIC 16F887A**

* Sử dụng công nghệ tích hợp cao RISC CPU.
* Người dùng có thể lập trình với 35 câu lệnh đơn giản.
* Tất cả các câu lệnh thự hiện trong một chu kỳ, lệnh ngoại trừ một số câu lệnh riêng rẽ nhánh thực hiện trong hai chu kỳ lệnh.
* Tốc độ hoạt động:
  + Xung đồng hồ là DC -20MHz.
  + Chu kỳ lệnh thực hiện trong 200ns.
* Bộ nhớ chương trình Flash 8Kx14 words.
* Bộ nhớ SRam 368x8 bytes.
* Bộ nhớ EFPROM 256x8 bytes.
* Số port I/O 35 port

*Khả năng của PIC*

* Khả năng ngắt
* Ngăn nhớ Stack được phân chia làm 8 mức
* Truy cập bộ nhớ bằng địa chỉ trực tiếp hoặc gián tiếp
* Nguồn khởi động lại (POR)
* Bộ tạo thời gian PWRT, bộ tạo dao động OST
* Bộ đếm xung thời gian WDT với nguồn dao động trên chip (nguồn dao động RC) đáng tin cậy
* Có mã chương trình bảo vệ
* Phương thức cất giữ Sleep
* Thiết kế toàn tĩnh
* Dải điện thế hoạt động 2V 5,5V
* Dòng điện sử dụng 25mA

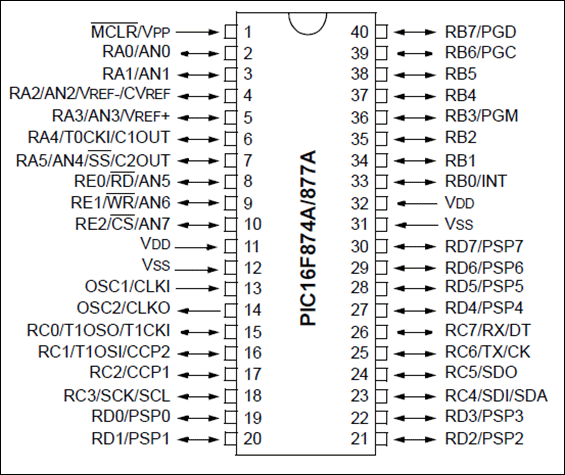
*Các tính năng nổi bật của thiết bị ngoại vi trên chip:*

* TIMER0: 8 bit của bộ định thời, bộ đếm với hệ số tỷ lệ trước.
* TIMER1: 16 bit của bộ định thời, bộ đếm với tỉ số tỉ lệ trước, có khả năng tăng trong khi ở chế độ Sleep qua xung đồng hồ cung cấp bên ngoài.
* TIMER2: 8 bit của bộ định thời, bộ đếm với 8 bit của hệ số tỉ lệ trước, hệ số tỉ lệ sau.
* Bộ chuyển đổi tín hiệu số sang tín hiệu tương tự với 10 bit.
* Cổng truyền thông tin nối tiếp SSP với SPI phương thức chủ.

**B. Khảo sát vi điều khiển PIC16F877A của hãng Microchip**

**1. Sơ đồ chân của PIC16F877A**

**a. Sơ đồ chân**

**a. Sơ đồ chân**

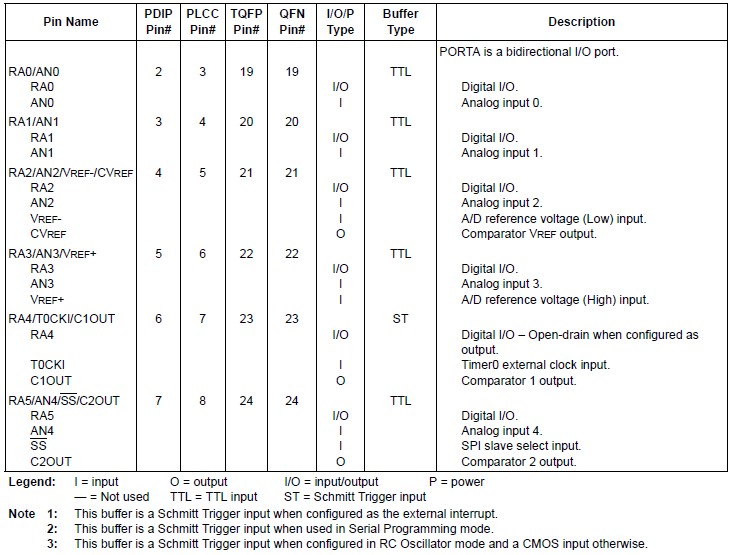
**b. Một số thông tin về vi điều khiển PIC16F877A**

**Port A**: PortA (RA0  RA5) có số chân từ chân số 2 đến chân số 7.

PortA (RPA) bao gồm 6 I/O pin. Đây là các chân “hai chiều” (bidirectional pin), nghĩa là có thể xuất và nhập được. Chức năng I/O này được điều khiển bởi thanh ghi TRISA (địa chỉ 85h). Bên cạnh đó PortA còn là ngõ ra của bộ ADC , bộ so sánh ngõ vào analog , ngõ vào xung clock của Timer0 và ngõ vào của bộ giao tiếp MSSP ( Master Synchronous Serial Port)

Các thanh ghi trong SRF liên quan đến PORTA bao gồm:

* PORTA (địa chỉ 05h): chứa giá trị của pin trong PORTA.
* TRISA (địa chỉ 85h): điều khiển xuất nhập.
* CMCON (địa chỉ 9Ch): thanh ghi điều khiển bộ so sánh.
* CVRCON (địa chỉ 9Dh): thanh ghi điều khiển bộ so sánh điện áp.
* ADCON1 (địa chỉ 9Eh): thanh ghi điều khiển bộ ADC.

Chức năng của các chân PortA

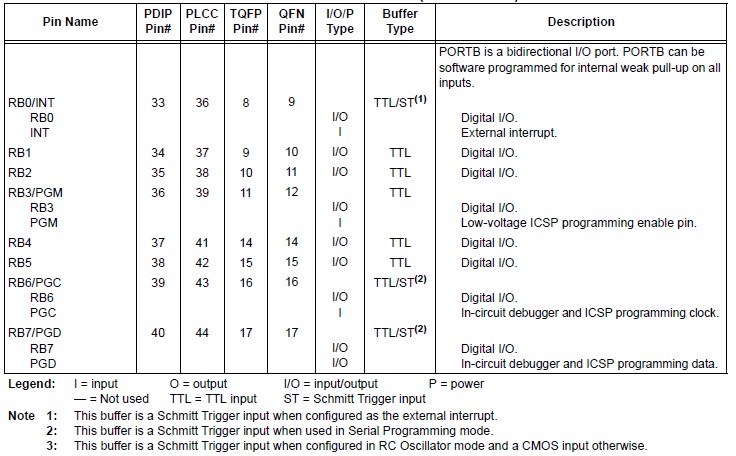
**Port B**: PortB (RB0  RB7) có số chân từ chân số 33 đến chân số 40.

PORTB (RPB) gôm̀ 8 pin I/O. thanh ghi điêù khiên̉ xuất nhâp̣ tương ứng là TRISB. Bên cạnh đó một sô ́ chân cuả PORTB coǹ được sử dụng trong quá trình nạp chương trình cho vi điều khiển với các chê ́ đô ̣ nạp khác nhau. PORTB còn liên quan đến ngắn ngoại vi Timer0 ̣

Chức năng cụ thể của từng chân trong PORTB như sau:

Cać thanh ghi SFR liên quan đến PORTB bao gồm:

* PORTB ( địa chỉ 06h, 106h ): chưá cać giá tri ̣ pin trong PORTB.
* TRISB ( điạ chỉ 86h, 186h ): thanh ghi điêù khiên̉ xuât́ nhâp̣ .
* OPTION\_REG ( địa chỉ 81h, 181h ): điêù khiên̉ ngắt ngoại vi và bộ Timer0.



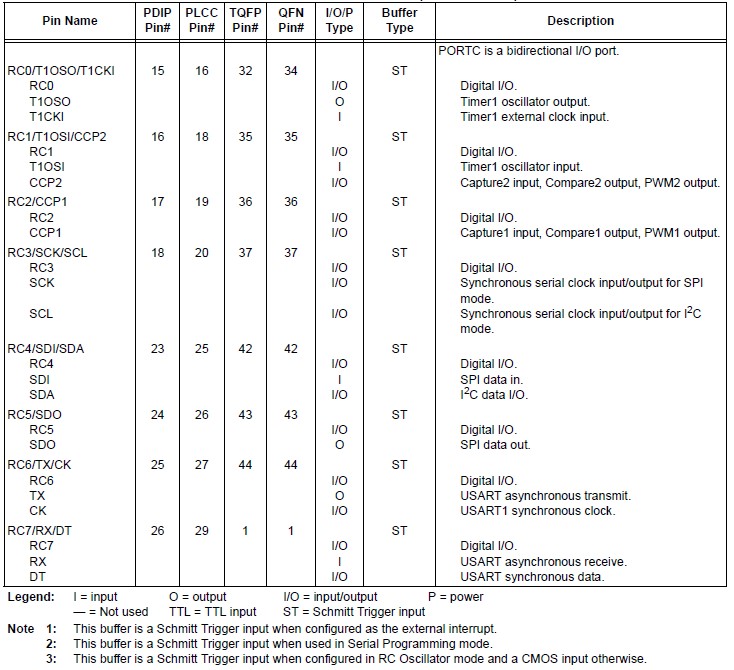
**Port C**: PortC( RC0  RC7) có số chân từ chân số 15 đến chân số 18 và chân số 23 đến chân số 26.

PortC (RPC) gồm 8 pin I/O. Thanh ghi điều khiển xuất nhập tương ứng là TRISC. Bên cạnh đó PORTC còn chứa các chân chức năng của bộ so sánh, bộ Timer1, bộ PWM và các chuẩn giao tiếp nối tiếp I2C, SPI, SSP, USART

Chức năng cụ thể của các chân trong PortC như sau :

Các thanh ghi điều khiển liên quan đến PortC gồm :

* PORTC ( địa chỉ 07h): chứa giá tri ̣ pin trong PORTC.
* TRISC ( điạ chi ̉ 87h ): thanh ghi điêù khiên̉ xuất nhập .



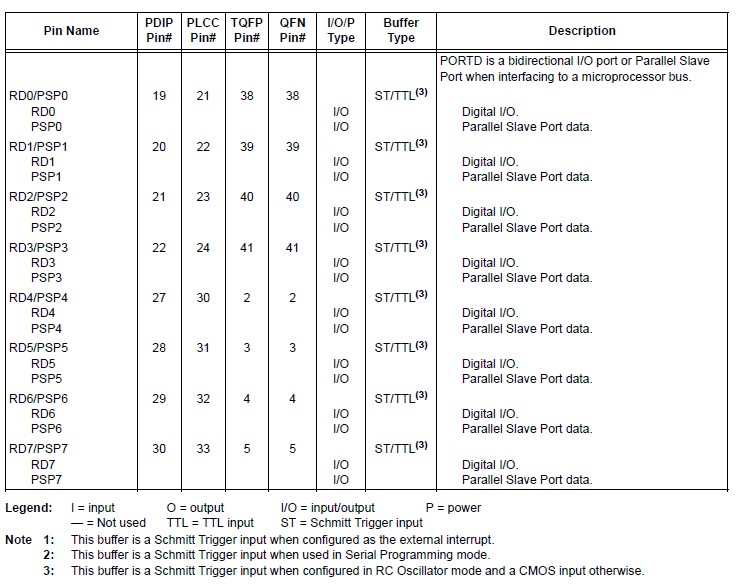
**Port D**: PortD (RD0  RD7) có số chân từ chân số 33 đến chân số 40.

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

Chưć năng cụ thê ̉ các chân trong PORTD như sau:

Cać thanh ghi liên quan đêń PORTD bao gôm̀ :

* PORTD : chứa giá trị các pin trong PORTD.
* TRISD: thanh ghi điều khiển xuất nhập .



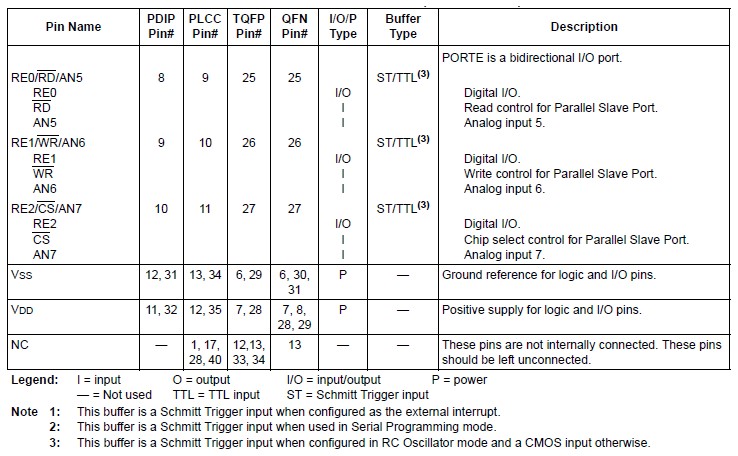
**Port E**: PortE (RE0  RE2) có số chân từ chân số 19 đến chân số 22 và chân số 27 đến chân 30.

PortE (RPE) gồm 3 chân I/O. 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.

Chưć năng cu ̣ thê ̉ cuả cać chân trong PORTE như sau :

Cać thanh ghi liên quan đêń PORTE bao gồm :

* PORTE : chưá giá tri ̣ pin trong PORTE.
* TRISE : Điều khiển xuất nhập và xác lập các thông số cho chuẩn giao tiếp PSP .
* ADCON1 : thanh ghi điều khiển khối ADC.



**1.1 Tổ chức các bộ nhớ.**

Cấu trúc của bộ nhớ vi điều khiển PIC16F877A bao gồm 2 bộ nhớ:

* Bộ nhớ chương trình (Programmemory).
* Bộ nhớ dữ liệu (Data memory).

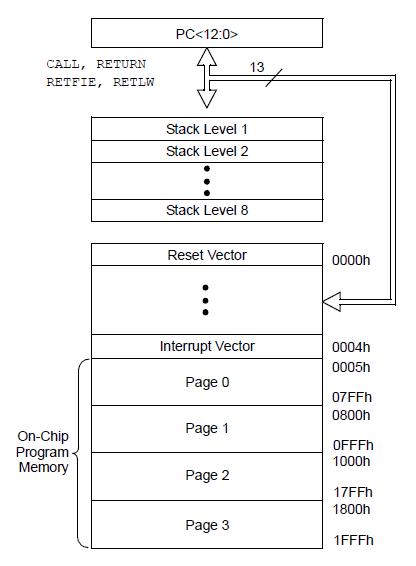
**a. Bộ nhớ chương trình (Programmemory).**

Bộ nhớ chương trình của vi điều khiển PIC16F877A là bộ nhớ Flash dung lượng bộ nhớ 8K được phân chia thành nhiều trang (từ 03). Như vậy bộ nhớ chương trình có khả năng chứa được 8\*1024=8192 câu lệnh.

Để mã hóa được địa chỉ của 8K bộ nhớ chương trình, bộ đếm chương trình có dung lượng 13bit.

Khi vi điều khiển được Reset, bộ đếm chương trình chỉ đến địa chỉ 0004h(Interrupt vector).

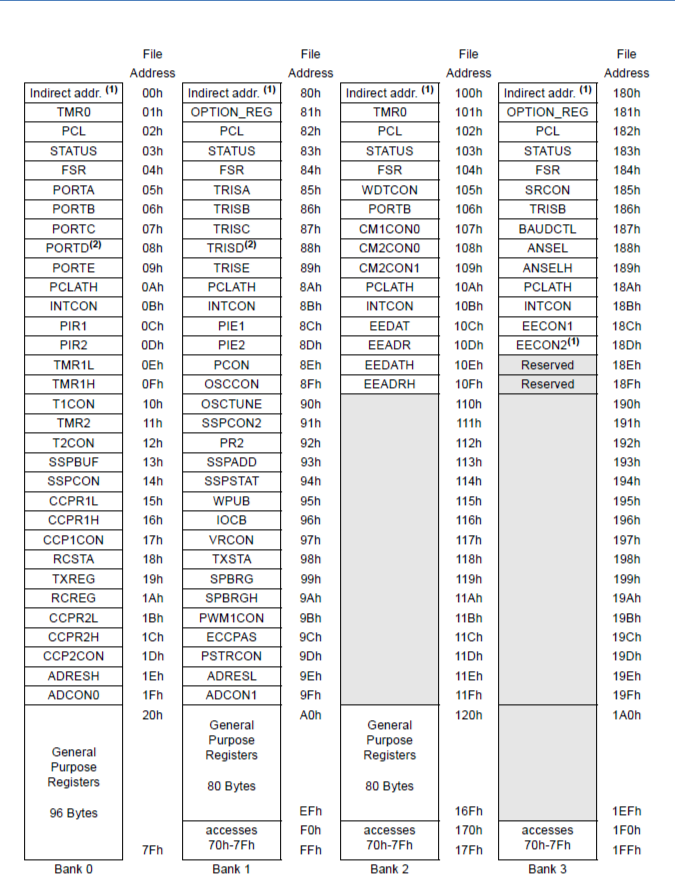
Bộ nhớ chương trình không bao gồm bộ nhớ stack và không được địa chỉ hóa bởi bộ đếm chương trình.

Bảng bộ nhớ chương trình và các ngăn xếp

Hình 1.3 Bộ nhớ chương trình 16F8887

**b.Bộ nhớ dữ liệu (Data memory).**

Bộ nhớ dữ liệu của PIC là bộ nhớ EEPROM được chia thành nhiều bank. Đối với PIC16F887 chia thành 4 bank. Mỗi bank có dung lượng chứa 128 byte, bao gồm các thanh ghi có chức năng đặc biệt SFG (Spencial Function Register) nằm ở các vùng địa chỉ thấp và các thanh ghi mục đích chung GPR (General Purpose Register) nằm ở các vùng địa chỉ còn lại trong bank. Các thanh ghi SFG thường xuyên được sử dụng sẽ được đặt ở tất cả các bank của bộ nhớ dữ liệu giúp truy suất và làm giảm bớt lệnh của chương trình. Bộ nhớ dữ liệu của PIC 16F887

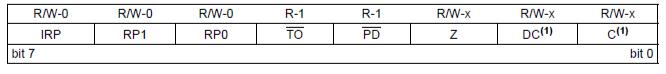
Hình 1.4 Bộ nhớ data 16F887

**1.1.3 Các thanh ghi đặc biệt**

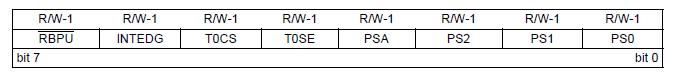
Các thanh ghi được sử dụng bởi CPU hoặc dùng đẻ thiết lập điều khiển các khối chức năng tích hợp trong vi điều khiển. Phân chia thanh SFR làm hai loại: thanh ghi SFR liên quan đến các chức năng bên trong CPU và thanh ghi SFR dùng để thiết lập và điều khiển các khối chức năng bên ngoài.

**a. Các thanh ghi liên quan đến bên trong:**

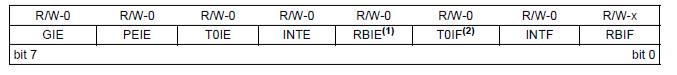
Thanh ghi SATUS (03h, 83h, 103h, 183h): thanh ghi chứa kết quả thực hiện phép toán của khối ALU, trạng thái RESET và cấc bit chọn bank cùng truy suất trong bộ nhớ dữ liệu.



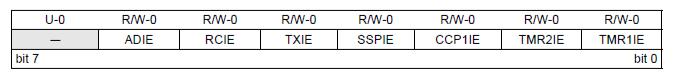
Thanh ghi OPTION\_REG (81h, 181h): thanh ghi này cho phép đọc và ghi, cho phép điều khiển các chức năng puled ma trận-up của các chân PORTB, xác lập các tham số xung tác động, cạnh tác động của ngắt ngoại vi và bộ đếm Timer0



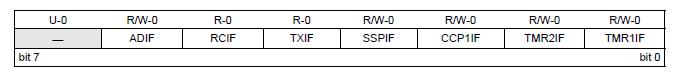
Thanh ghi INTCON (0Bh, 8Bh, 10Bh, 18Bh): thanh ghi cho phép đọc và ghi, chứa các bit điều khiển và các bít cờ hiệu khi Time0 tràn, ngắt ngoại vi RB0/INT và ngắt interrupt-on-change tại các chân của PORTB.



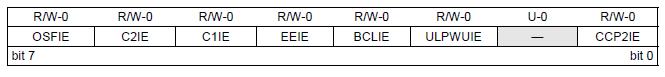
Thanh ghi PIE1 (8Ch): chứa các bit điều khiển chi tiết các ngắt của khối chức năng ngoại vi



Thanh ghi PIR1 (0Ch) chứa cờ ngắt của các khối chức năng ngọai vi, các ngắt này được cho phép bởi các bit điều khiển chứa trong thanh ghi PEI1.



Thanh ghi PIE2 (8Dh): chứa các bit điều khiển các ngắt của các khối chức năng CCP2, SSP bus, ngắt của bộ so sánh và ngắt ghi vào bộ nhớ EEPROM



Thanh ghi PIR2 (0Dh): chứa các cờ ngắn của của các khối chức năng ngoại vi các ngắt này được cho phép bởi các bit điều khiển chứa trong thanh ghi PIE2

**1.1.4 Các cổng xuất nhập của PIC( I/O)**

PIC16F887 tất cả có 35 chân I/O mục đích thông thường (GPIO: General Purpose Input Ouput) có thể được sử dụng. Tùy theo những thiết bi ngoại vi được chọn mà một vài chân không thể sử dụng ở chức năng GPIO. Thông thường, khi một thiết bị ngoai vi được chọn, những chân liên quan của thiết bị ngoại vi không được sử dụng ở chức năng GPIO.35 chân được chia thành 5 port:

* PortA chia làm 8 chân.
* PortB chia làm 8 chân.
* PortC chia làm 8 chân.
* PortD chia làm 8 chân.
* PortE chia làm 3 chân.

Mỗi port được điều khiển bởi 2 thanh ghi 8-bit, thanh ghi Port và thanh ghi Tris. Thanh ghi Tris được sử dụng để điều khiển port nhập hay xuất. Mỗi bit của Tris sẽ điều khiển mỗi chân của port đó, nếu giá trị bit là 1 thì chân liên quan là nhập, ngược lại nếu giá trị bit là 0 thì chân liên quan là xuất. Thanh ghi Port được sử dụng để chứa các giá trị của port liên quan. Mỗi bit của thanh ghi Port chứa giá trị của chân liên quan.

**2, Các vấn đề về Timer**

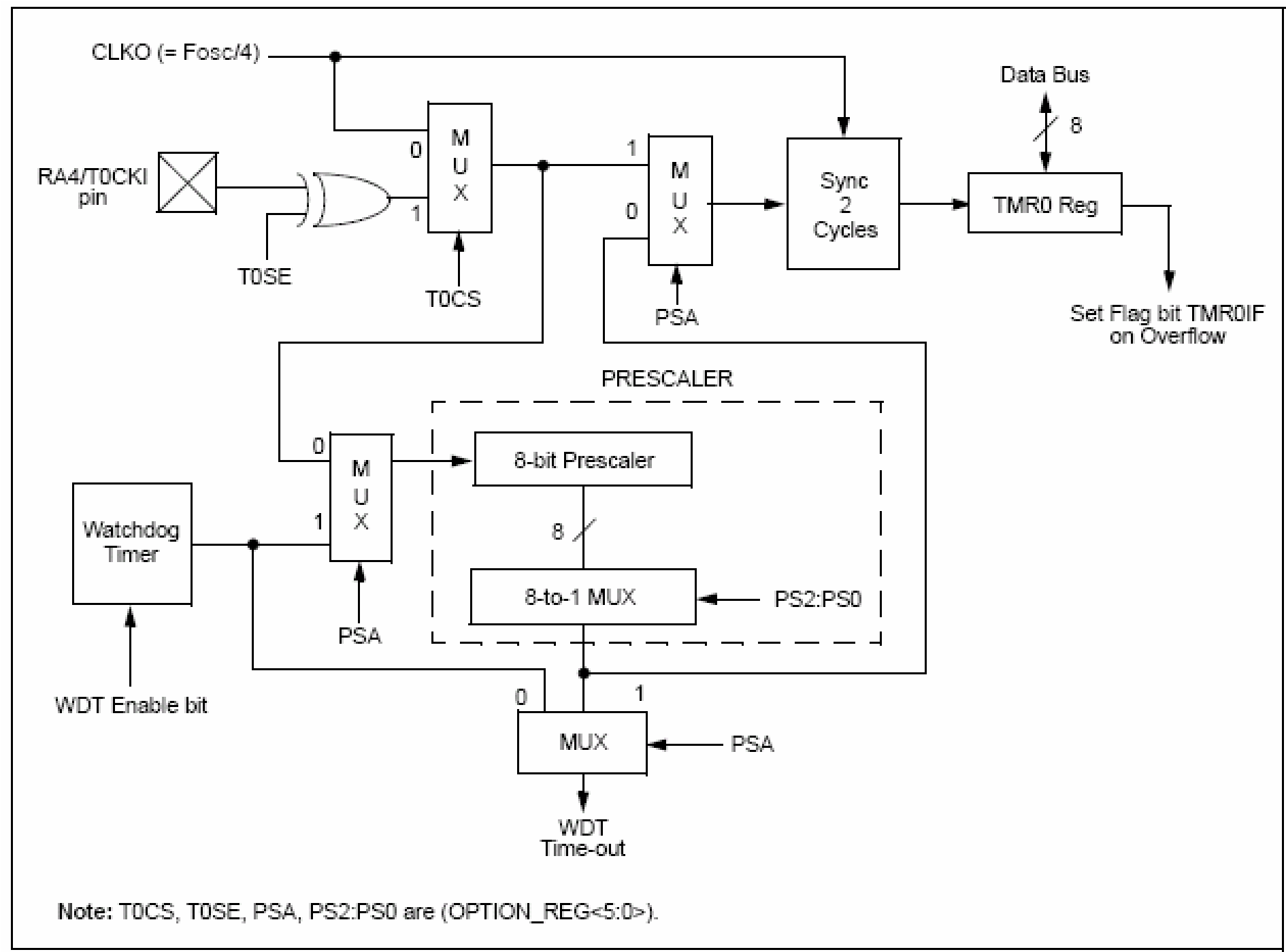
PIC16F877A có tất cả 3 timer : timer0 (8 bit), timer1 (16 bit) và timer2 (8 bit).

**2.1, Timer0**

**a, Khát quát**

Là bộ định thời hoặc bộ đếm có những ưu điểm sau:

* Là timer/counter 8 bit.
* Có thể đọc và ghi giá trị đếm của timer/counter.
* Có bộ chia trước 8 bit cho phép lập trình lựa chọn hệ số chia bằng phần mềm.
* Cho phép lựa chọn nguồn xung clock bên trong hoặc bên ngoài.
* Phát sinh ngắt khi bị tràn từ FFH về 00H.
* Cho phép lựa chọn tác động xung CK cạnh lên hoặc cạnh xuống .



**b, Hoạt động của Timer 0:**

Timer 0 có thể hoạt động như một bộ định thời hoặc một bộ đếm.Việc chọn bộ định thời hoặc bộ đếm có thể được xác lập bằng việc xoá hoặc đặt bít TOCS của thanh ghi OPTION\_REG<5>.

Nếu dùng hệ số chia xung đầu vào thì xoá bit PSA của thanh ghi OPTION\_REG<3>.

Trong chế độ bộ định thời được lựa chọn bởi việc xoá bit TOCS (OPTION REG<5>), nó sẽ được tăng giá trị sau một chu kỳ lệnh nếu không chọn hệ số chia xung đầu vào.Và giá trị của nó được viết tới thanh ghi TMR0.

Khi dùng xung clock bên ngoài cho bộ định thời Timer0 và không dùng hệ số chia clock đầu vào Timer0 thì phải đáp ứng các điều kiện cần thiết để có thể hoạt động đó là phải bảo đảm xung clock bên ngoài có thể đồng bộ với xung clock bên trong (TOSC).

Hệ số chia dùng cho Timer 0 hoặc bộ WDT. Các hệ số nay không có khả năng đọc và khả năng viết. Để chọn hệ số chia xung cho bộ tiền định của Timer0 hoặc cho bộ WDT ta tiến hành xoá hoặc đặt bít PSA của thanh ghi OPTION\_REG<3>

Những bít PS2, PS1, PS0 của thanh ghi OPTION\_REG<2:0> dùng để xác lập các hệ số chia.

Bộ tiền định có giá trị 1:2 chẳng hạn, có nghĩa là : bình thường không sử dụng bộ tiền định của Timer0 (đồng nghĩa với tiền định tỉ lệ 1:1) thì cứ khi có tác động của 1 xung clock thì timer0 sẽ tăng thêm một đơn vị. Nếu sử dụng bộ tiền định 1:4 thì phải mất 4 xung clock thì timer0 mới tăng thêm một đơn vị. Vô hình chung, giá trị của timer0 (8 bit) lúc này không còn là 255 nữa mà là 255\*4=1020.

c, **Ngắt của bộ Timer0**

Ngắt của bộ Timer 0 được phát sinh ra khi thanh ghi TMR0 bị tràn tức từ FFh quay về 00h.Khi đó bít T0IF của thanh ghi INTCON<2> sẽ được đặt. Bit này phải được xóa bằng phần mềm nếu cho phép ngắt bit T0IE của thanh ghi INTCON<5> được set. Timer0 bị dừng hoạt ở chế độ SLEEP ngắt Timer 0 không đánh thức bộ xử lý ở chế độ SLEEP.

**D, Các thanh ghi liên quan đến Timer0 bao gồm:**

***Thanh ghi OPTION\_REG : điều khiển hoạt động của Timer0***

Cấu trúc thanh ghi OPTION\_REG REGISTER điều khiển hoạt động của Timer0



* bit 5 TOCS lựa chọn nguồn clock

1=Clock ngoài từ chân T0CKI

0=Clock trong Focs/4

* bit 4 T0SE bit lựa chon sườn xung clock

1=Timer 0 tăng khi chân T0CKI tử cao xuống thấp (sườn xuống)

0=Timer 0 tăng khi chân T0CKI tử thấp lên cao (sườn xuống)

* bit 3 PSA bit gán bộ chia xung đầu vào

1=gán bộ chia Prescaler cho WDT

0=gán bộ chia Prescaler cho Timer 0

* bit 2:0 PS2:PS1 lựa chọn hệ số chia hệ số xung theo bảng sau:

|  |  |  |
| --- | --- | --- |
| PS2:PS0 | Timer0 | 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 |

Thanh ghi TMR0 đại chỉ 01h và 101h : chứa giá trị của bộ định thời Timer0

Thanh ghi INTCON : cho phép ngắt hoạt động

Thanh ghi chứa các bit điều khiển và các bít cờ hiệu khi timer0 bị tràn, ngắt ngoại vi RB0/INT và ngắt interrupt\_on\_change tại các chân của PORTB.



Cấu trúc thanh ghi INTCON cho phép ngắt Timer0 hoạt động

* Bit 7 GIE Global Interrupt Enable bit

GIE = 1 cho phép tất cả các ngắt.

GIE = 0 không cho phép tất cả các ngắt.

* Bit 6 PEIE Pheripheral Interrupt Enable bit

PEIE = 1 cho phép tất cả các ngắt ngoại vi.

PEIE = 0 không cho phép tất cả các ngắt ngoại vi.

* Bit 5 TMR0IE Timer0 Overflow Interrupt Enable bit

TMR0IE = 1 cho phép ngắt Timer0.

TMR0IE = 0 không cho phép ngắt Timer0.

* Bit 4 RBIE RB0/INT External Interrupt Enable bit

RBIE = 1 cho phép tất cả các ngắt ngoại vi RB0/INT

RBIE = 0 không cho phép tất cả các ngắt ngoại vi RB0/INT

* Bit 3 RBIE RB Port change Interrupt Enable bit

RBIE = 1 cho phép ngắt RB Port change

RBIE = 0 không cho phép ngắt RB Port change

* Bit 2 TMR0IF Timer0 Interrupt Flag bit

TMR0IF = 1 thanh ghi TMR0 bị tràn (phải xóa cờ hiệu bằng chương trình).

TMR0IF = 0 thanh ghi TMR0 chưa bị tràn.

* Bit 1 INTF BR0/INT External Interrupt Flag bit

INTF = 1 ngắt RB0/INT xảy ra (phải xóa cờ hiệu bằng chương trình).

INTF = 0 ngaột RB0/INT chửa xaỷy ra.

* Bit 0 RBIF RB Port Change Interrupt Flag bit

RBIF = 1 ít nhất có một chân RB7:RB4 có sự thay đổi trạng thái. Bít này phải được xóa bằng chương trình sau khi đã kiểm tra lại các giá trị chân tại PORTB.

RBIF = 0 không có sự thay đổi trạng thái các chân RB7:RB4.

**2.2, TIMER1**

**a, Khát quát**

**Timer1** là bộ định thời 16 bit, giá trị của Timer1 sẽ được lưu trong hai thanh ghi 8 bit TMR1H:TMR1L. Cờ ngắt của Timer1 là bit TMR1IF, bit điều khiển của Timer1 là TRM1IE.Cặp thanh ghi của TMR1 sẽ tăng từ 0000h lên đến FFFFh rồi sau đó tràn về 0000h. Nếu ngắt được cho phép, nó sẽ xảy ra khi khi giá trị của TMR1 tràn từ FFFFh rồi về 0000h, lúc này TMR1IF sẽ bật lên.

**b, Timer1 có 3 chế độ hoạt động :**

* Chế độ hoạt động định thời đồng bộ: Chế độ được lựa chọn bởi bit TMR1CS. Trong chế độ này xung cấp cho Timer1 là Fosc/4, bit T1SYNC không có tác dụng.
* Chế độ đếm đồng bộ: trong chế độ này, giá trị của timer1 sẽ tăng khi có xung cạnh lên vào chân T1OSI/RC1. Xung clock ngoại sẽ được đồng bộ với xung clock nội, hoạt động đồng bộ được thực hiện ngay sau bộ tiền định tỉ lệ xung (prescaler).
* Chế độ đếm bất đồng bộ:chế độ này xảy ra khi bit T1SYNC được set. Bộ định thời sẽ tiếp tục đếm trong suốt quá trình ngủ (Sleep) của vi điều khiển và có khả năng tạo một ngắt khi bộ định thời tràn và làm cho vi điều khiển thoát khỏi trạng thái ngủ.

**C, Các thanh ghi liên quan đến Timer1 bao gồm:**

* INTCON (địa chỉ 0Bh, 8Bh, 10Bh, 18Bh): cho phép ngắt hoạt động (2 bit GIE và PEIE).
* PIR1 (địa chỉ 0Ch): chứa cờ ngắt Timer1 (TMR1IF).
* PIE1 (địa chỉ 8Ch): cho phép ngắt Timer1 (TMR1IE).

Ba thanh ghi vừa nêu trên sẽ được trình bày ở phần chương trình ngắt của PIC

* TMR1L (địa chỉ 0Eh): chứa giá trị 8 bít thấp của bộ đếm Timer1.
* TMR1H (địa chỉ 0Eh): chứa giá trị 8 bít cao của bộ đếm Timer1.

Hai thanh ghi TMR1L và TMR1H là 2 thanh ghi chứa dữ liệu 16 bit (lần lượt chứa 4 bit thấp và 4 bit cao) của bộ đếm Timer1

* T1CON (địa chỉ 10h): xác lập các thông số chi Timer



Cấu trúc thanh ghi T1CON điều khiển hoạt động của Timer1

* bit 7,6 không sử dụng
* bit 5,4 T1CKPS1: T1CKPS0 lựa chọn hệ số chia xung vào.

|  |  |
| --- | --- |
|  | |
| T1CKPS1 T1CKPS0 | |
| 00 | 1:1 |
| 01 | 1:2 |
| 10 | 1:4 |
| 11 | 1:8 |

* bit 3 T1OSCEN bit điều khiển bộ dao động Timer1

=1 Bộ dao động hoạt động

=0 Bộ dao động không hoạt động

* bit 2 bit điều khiển xung clock ngoài đồng bộ

khi TMR1CS=1

* bit2=0 có đồng clock ngoài

bit2=1 không đồng bộ clock ngoài

khi TMR1CS=0 bit này không có tác dụng

* bit 1 TMR1CS bit lựa chọn nguồn xung clock vào

TMR1CS=1 clock từ chân RC0/T1OSO/T1CKI (sườn lên)

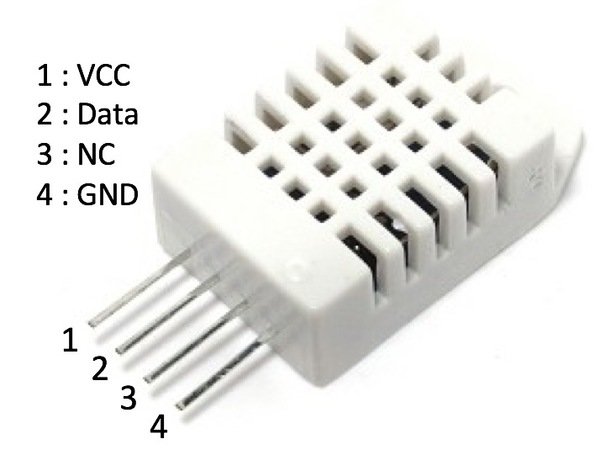
TMR1CS=0 clock trong Fosc/4

* bit 1 bit 0: bật tắt Timer

1= Timer 1 enable

0=Timer 1 Disable

**Cảm biến nhiệt độ - độ ẩm DTH22**

****

**Thông số kỹ thuật:**

* Nguồn sử dụng: 3~5VDC.
* Dòng sử dụng: 2.5mA max (khi truyền dữ liệu).
* Đo tốt ở độ ẩm 0100%RH với sai số 2-5%.
* Đo tốt ở nhiệt độ -40 to 80°C sai số ±0.5°C.
* Tần số lấy mẫu tối đa 0.5Hz (2 giây 1 lần)
* Kích thước 27mm x 59mm x 13.5mm (1.05" x 2.32" x 0.53")
* 4 chân, khoảng cách chân 0.1''.

**Modul Bluetooth HC-05**

****

* Điện áp hoạt động: 3.3VDC.
* Mức điện áp chân giao tiếp: TTL tương thích 3.3VDC.
* Dòng điện khi hoạt động: khi Pairing 30 mA, sau khi pairing hoạt động truyền nhận bình thường 8 mA.
* Baudrate UART có thể chọn được: 1200, 2400, 4800, 9600, 19200, 38400, 57600, 115200.

**III, Mô phỏng và thiết kế phần mềm:**

**1, Sơ đồ thuật toán:**

Set = 1

T1++

Set = 2

T2++

Đ

Set = 5

RH2++

Set = 4

RH1++

Set = 3

S

S

S

S

Pin\_b1 = 1

Đ

Đ

Đ

Pin\_b2 = 1

RH2<RH

RH1>RH

Pin\_b4 = 1

Pin\_b3 = 1

T2<T

T1>T

**2) Code mô phỏng chương trình**

#include <16F877A.h>

#fuses HS,NOWDT,NOPROTECT,NOLVP

#use delay(clock = 20000000)

#include <lcd.c>

#use fast\_io(c)

// Connection pin between PIC16F877A and DHT22 sensor

#BIT Data\_Pin = 0x06.0 // Pin mapped to PORTD.0

#BIT Data\_Pin\_Direction = 0x86.0 // Pin direction mapped to TRISD.0

#use rs232(baud=38400,UART1,parity=N,bits=8)

char message1[] = "Temp = 00.0 C";

char message2[] = "RH = 00.0 %";

char message3[] = "Tem1 = 00.0 C";

char message4[] = "Tem2 = 00.0 C";

char message5[] = "H1 = 00.0 %";

char message6[] = "H2 = 00.0 %";

short Time\_out ;

unsigned int8 T\_byte1, T\_byte2, RH\_byte1, RH\_byte2, CheckSum ;

unsigned int16 Temp, RH,T1 = 0,T2 = 500;

unsigned int16 RH1 = 0, RH2 = 500;

int set = 0;

void nut()

{

if(input(pin\_d3)==0)

{

while(input(pin\_d3)==0)

{

}

set++;

if(set>5){set=1;}

}

}

void start\_signal(){

Data\_Pin\_Direction = 0; // Configure connection pin as output

Data\_Pin = 0; // Connection pin output low

delay\_ms(25);

Data\_Pin = 1; // Connection pin output high

delay\_us(30);

Data\_Pin\_Direction = 1; // Configure connection pin as input

}

short check\_response(){

delay\_us(40);

if(!Data\_Pin){ // Read and test if connection pin is low

delay\_us(80);

if(Data\_Pin){ // Read and test if connection pin is high

delay\_us(50);

return 1;}

}

}

unsigned int8 Read\_Data(){

unsigned int8 i, k, \_data = 0; // k is used to count 1 bit reading duration

if(Time\_out)

break;

for(i = 0; i < 8; i++){ k = 0; while(!Data\_Pin){ // Wait until pin goes high k++; if (k > 100) {Time\_out = 1; break;}

delay\_us(1);}

delay\_us(30);

if(!Data\_Pin)

bit\_clear(\_data, (7 - i)); // Clear bit (7 - i)

else{

bit\_set(\_data, (7 - i)); // Set bit (7 - i)

while(Data\_Pin){ // Wait until pin goes low

k++;

if (k > 100) {Time\_out = 1; break;}

delay\_us(1);}

}

}

return \_data;

}

void DHT22(){

delay\_ms(1000);

Time\_out = 0;

Start\_signal();

if(check\_response()){ // If there is response from sensor

RH\_byte1 = Read\_Data(); // read RH byte1

RH\_byte2 = Read\_Data(); // read RH byte2

T\_byte1 = Read\_Data(); // read T byte1

T\_byte2 = Read\_Data(); // read T byte2

Checksum = Read\_Data(); // read checksum

if(Time\_out){ // If reading takes long time

lcd\_putc('\f'); // LCD clear

lcd\_gotoxy(5, 1); // Go to column 5 row 1

lcd\_putc("Time out!");

}

else{

if(CheckSum == ((RH\_Byte1 + RH\_Byte2 + T\_Byte1 + T\_Byte2) & 0xFF)){

RH = RH\_byte1;

RH = (RH << 8) | RH\_byte2;

Temp = T\_byte1;

Temp = (Temp << 8) | T\_byte2; if (Temp > 0X8000){

message1[6] = '-';

Temp = Temp & 0X7FFF; }

else

message1[6] = ' ';

message1[7] = (Temp / 100) % 10 + 48;

message1[8] = (Temp / 10) % 10 + 48;

message1[10] = Temp % 10 + 48;

message2[7] = (RH / 100) % 10 + 48;

message2[8] = (RH / 10) % 10 + 48;

message2[10] = RH % 10 + 48;

message1[11] = 223; // Degree symbol

lcd\_putc('\f'); // LCD clear

lcd\_gotoxy(1, 1); // Go to column 1 row 1

printf(lcd\_putc,message1); // Display message1

lcd\_gotoxy(1, 2); // Go to column 1 row 2

printf(lcd\_putc, message2); // Display message2

}

else {

lcd\_putc('\f'); // LCD clear

lcd\_gotoxy(1, 1); // Go to column 1 row 1

lcd\_putc("Checksum Error!");

}

}

}

else {

lcd\_putc('\f'); // LCD clear

lcd\_gotoxy(3, 1); // Go to column 3 row 1

lcd\_putc("No response");

lcd\_gotoxy(1, 2); // Go to column 1 row 2

lcd\_putc("from the sensor");

}

}

void main(){

lcd\_init(); // Initialize LCD module

lcd\_putc('\f'); // LCD clear

lcd\_gotoxy(5,1);

lcd\_putc("Nhom 10");

while(TRUE){

nut();

// T2

message4[7] = (T2 / 100) % 10 + 48;

message4[8] = (T2 / 10) % 10 + 48;

message4[10] = T2 % 10 + 48;

message4[11] = 223;

// T1

message3[7] = (T1 / 100) % 10 + 48;

message3[8] = (T1 / 10) % 10 + 48;

message3[10] = T1 % 10 + 48;

message3[11] = 223;

// RH1

message5[7] = (RH1 / 100) % 10 + 48;

message5[8] = (RH1 / 10) % 10 + 48;

message5[10] = RH1 % 10 + 48;

//RH2

message6[7] = (RH2 / 100) % 10 + 48;

message6[8] = (RH2 / 10) % 10 + 48;

message6[10] = RH2 % 10 + 48;

switch(set){

case 1 :

{

do{

if(input(pin\_a0)==0)

{

while(input(pin\_a0)==0)

{

}

T1=T1+100;

}else if (input(pin\_a1)==0)

{

while(input(pin\_a1)==0)

{

}

T1=T1+10;

} else if(input(pin\_a2)==0)

{

while(input(pin\_a2)==0)

{

}

T1++;

}

else if(input(pin\_a3)==0)

{

while(input(pin\_a3)==0)

{

}

T1=T1-100;

}else if (input(pin\_a4)==0)

{

while(input(pin\_a4)==0)

{

}

T1=T1-10;

} else if(input(pin\_a5)==0)

{

while(input(pin\_a5)==0)

{

}

T1--;

}

lcd\_gotoxy(1, 1); // Go to column 1 row 1

printf(lcd\_putc,message3); // Display message1

lcd\_gotoxy(1, 2); // Go to column 1 row 1

printf(lcd\_putc,message4); // Display message1

}

while(T1>500);

break;

}

case 2:

{

do{

if(input(pin\_a0)==0)

{

while(input(pin\_a0)==0)

{

}

T2=T2+100;

}else if (input(pin\_a1)==0)

{

while(input(pin\_a1)==0)

{

}

T2=T2+10;

} else if(input(pin\_a2)==0)

{

while(input(pin\_a2)==0)

{

}

T2++;

}

else if(input(pin\_a3)==0)

{

while(input(pin\_a3)==0)

{

}

T2=T2-100;

}else if (input(pin\_a4)==0)

{

while(input(pin\_a4)==0)

{

}

T2=T2-10;

} else if(input(pin\_a5)==0)

{

while(input(pin\_a5)==0)

{

}

T2--;

}

lcd\_gotoxy(1, 1); // Go to column 1 row 1

printf(lcd\_putc,message3); // Display message1

lcd\_gotoxy(1, 2); // Go to column 1 row 1

printf(lcd\_putc,message4); // Display message1

}while(T2<501);

break;

}

case 3 :

{

do{

if(input(pin\_a0)==0)

{

while(input(pin\_a0)==0)

{

}

RH1=RH1+100;

}else if (input(pin\_a1)==0)

{

while(input(pin\_a1)==0)

{

}

RH1=RH1+10;

} else if(input(pin\_a2)==0)

{

while(input(pin\_a2)==0)

{

}

RH1++;

}

else if(input(pin\_a3)==0)

{

while(input(pin\_a3)==0)

{

}

RH1=RH1-100;

}else if (input(pin\_a4)==0)

{

while(input(pin\_a4)==0)

{

}

RH1=RH1-10;

} else if(input(pin\_a5)==0)

{

while(input(pin\_a5)==0)

{

}

RH1--;

}

lcd\_gotoxy(1, 1); // Go to column 1 row 1

printf(lcd\_putc,message5); // Display message1

lcd\_gotoxy(1, 2); // Go to column 1 row 1

printf(lcd\_putc,message6); // Display message1

}while(RH1>500);

break;

}

case 4 :

{

do{

if(input(pin\_a0)==0)

{

while(input(pin\_a0)==0)

{

}

RH2=RH2+100;

}else if (input(pin\_a1)==0)

{

while(input(pin\_a1)==0)

{

}

RH2=RH2+10;

} else if(input(pin\_a2)==0)

{

while(input(pin\_a2)==0)

{

}

RH2++;

}

else if(input(pin\_a3)==0)

{

while(input(pin\_a3)==0)

{

}

RH2=RH2-100;

}else if (input(pin\_a4)==0)

{

while(input(pin\_a4)==0)

{

}

RH2=RH2-10;

} else if(input(pin\_a5)==0)

{

while(input(pin\_a5)==0)

{

}

RH2--;

}

lcd\_gotoxy(1, 1); // Go to column 1 row 1

printf(lcd\_putc,message5); // Display message1

lcd\_gotoxy(1, 2); // Go to column 1 row 1

printf(lcd\_putc,message6); // Display message1

}while(RH2<501);

break;

}

case 5 :

{

DHT22();

puts(message1);

puts(message2);

if(RH<RH1)

{

output\_high(pin\_b3);

output\_low(pin\_b4);

}

else if(RH>RH2)

{

output\_high(pin\_b4);

output\_low(pin\_b3);

}

else

{

output\_low(pin\_b4);

output\_low(pin\_b3);

}

if(T1>Temp)

{

output\_high(pin\_b1);

output\_low(pin\_b2);

} else if(Temp>T2)

{

output\_high(pin\_b2);

output\_low(pin\_b1);

}

else

{

output\_low(pin\_b1);

output\_low(pin\_b2);

}

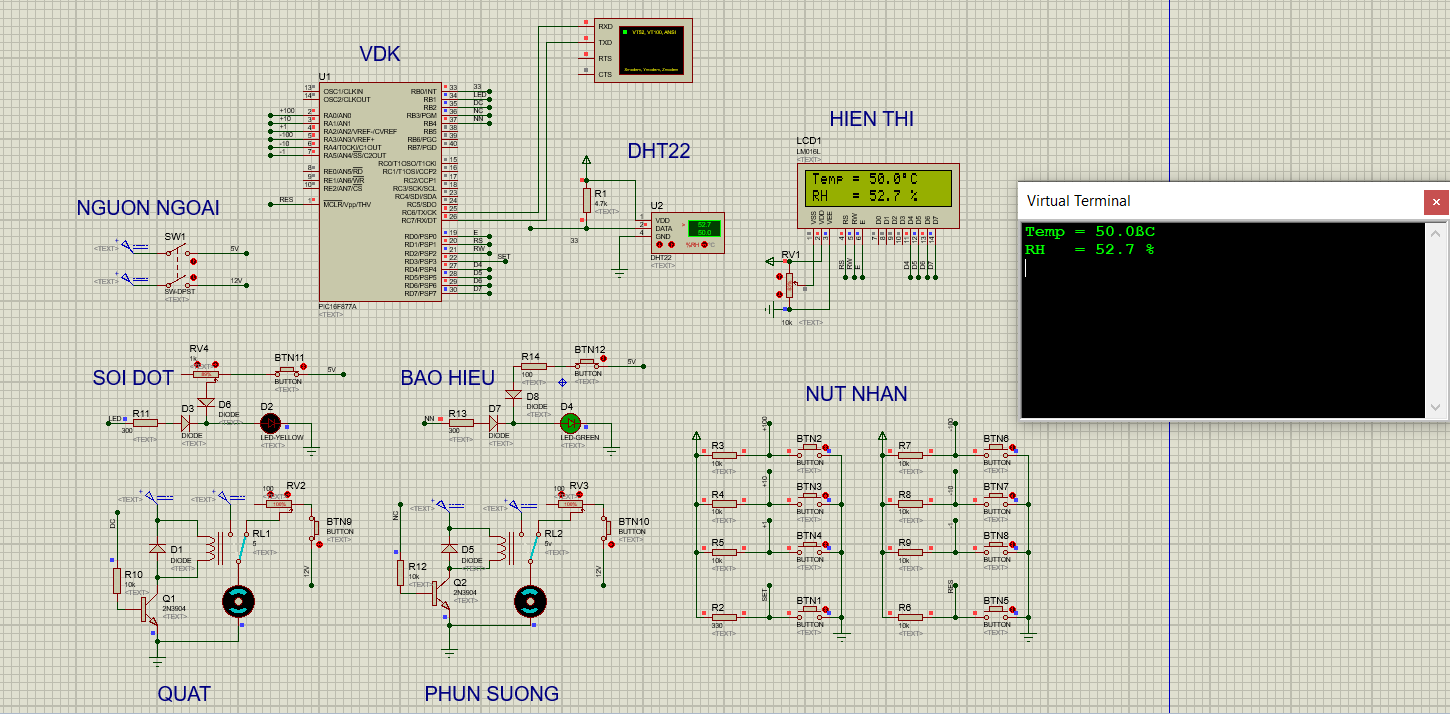
}

}

}

}

**3) Mô phỏng trên proteus**



**4. Sản phầm thực tế do nhóm thiết kế:**

