

<b>Người báo cáo:</b>	Nguyễn Trung Chính	<b>Tài liệu:</b>	TUT02.02
<b>Ngày:</b>	1/30/2006	<b>Trang:</b>	1/18

# Tutorial 02.02

**Gửi đến:** picvietnam@googlegroups.com

**Nội dung:** Hàm Delay

MICROSOFT WORD

## Tóm tắt:

Tutorial post lên picvietnam, topic "PIC16F877A TỪ DỄ TỚI KHÓ" thuộc luồng "CO BẢN VỀ VI ĐIỀU KHIỂN VÀ PIC" với nội dung:

- Vài nét sơ lược về mục đích và tác dụng của chương trình delay.
- Phân tích source code một số chương trình delay.
- Tập trung phân tích, khai thác chương trình delay của Nigel như một dạng chương trình delay được chuẩn hóa.

Tutorial này sử dụng khá nhiều các kiến thức trong tutorial của Nigel.

## 1. Vài nét sơ lược về chương trình delay.

### 1.1. Chu kì xung clock và chu kì lệnh

Trong phần này ta sẽ bàn đến một vài kiến thức cơ sở phục vụ cho việc viết chương trình delay. Cụ thể là tìm hiểu về chu kì xung clock và chu kì lệnh trong vi điều khiển PIC.

Ta đã biết để vi điều khiển hoạt động được cần phải cung cấp một nguồn xung clock từ bên ngoài. Đối với vi điều khiển PIC, nguồn xung clock có thể là một mạch dao động RC đơn giản, một thạch anh,...Tất nhiên, yêu cầu của nguồn xung clock phải là càng ổn định càng tốt.

Thông thường, nguồn xung sử dụng cho vi điều khiển nói chung và PIC nói riêng là thạch anh với các ưu điểm giá thành không cao, khá ổn định và rất thuận tiện trong việc tính toán, thiết kế mạch ứng dụng và chương trình cho vi điều khiển. Trong bài này, ta cũng sử dụng thạch anh làm nguồn xung cho vi điều khiển.

Mỗi thạch anh có một tần số dao động cố định, ta gọi tần số đó là  $f_0$ , thông thường  $f_0$  có các tần số 4 MHz, 10 MHz, 20 MHz, ... Tùy theo mỗi loại vi điều khiển mà yêu cầu đối với  $f_0$  có thể khác nhau. **Đối với vi điều khiển PIC16F877A, tần số dao động  $f_0$  phải nhỏ hơn hoặc bằng 20 MHz**, đây cũng là tần số hoạt động tối đa mà đa số các vi điều khiển PIC thuộc dòng mid-range có khả năng đáp ứng được. Chu kì dao động của thạch anh ta gọi là  $t_0$  và được tính theo công thức:

$$t_0 = 1/f_0 \quad (1)$$

Rất cơ bản! Không có gì cần chú thích thêm cho công thức này.

Người báo cáo:	Nguyễn Trung Chính	Tài liệu:	TUT02.02
Ngày:	1/30/2006	Trang:	2/18

Ta cũng đã biết rằng có hai lối kiến trúc dùng để tổ chức một vi điều khiển, đó là kiến trúc Von-Neuman và kiến trúc Havard. Vi điều khiển PIC được tổ chức theo lối kiến trúc Havard. Ta không đi sâu vào các lối kiến trúc này, mà chỉ cần biết rằng *với lối kiến trúc Havard, mỗi lệnh sẽ được thực thi xong trong một khoảng thời gian là một chu kì lệnh. Khoảng thời gian này luôn cố định và phụ thuộc vào chu kì của xung clock.*

Ta có một “định nghĩa” mang tính ... đại khái như sau: *chu kì lệnh của vi điều khiển PIC là khoảng thời gian mà vi điều khiển PIC thực thi xong một lệnh. Ta gọi thời gian của một chu kì lệnh là  $t_i$ .*

Để thực thi xong một lệnh, vi điều khiển PIC cần đến 4 chu kì xung clock. Như vậy thời gian thực thi xong một lệnh sẽ được tính:

$$t_i = 4t_0 \quad (2)$$

Thay công thức (1) vào công thức (2) ta có được công thức tính thời gian của một lệnh (một chu kì lệnh) như sau:

$$t_i = 4/f_0 \quad (3)$$

*Ví dụ:* nếu ta sử dụng thạch anh loại 4 MHz thì thời gian thực thi một lệnh của vi điều khiển là:

$$t_i = 4/(4 \times 10^6) = 1 \mu s$$

Để thuận tiện cho việc tính toán và thiết kế chương trình delay, ta sẽ sử dụng loại thạch anh 4 MHz cho mạch ứng dụng, vì như các bạn đã thấy, thời gian thực thi một lệnh của vi điều khiển lúc đó là 1  $\mu s$ . Quá chẵn!

## 1.2. Mục đích và tác dụng của chương trình delay

Như ta đã thấy ở mục 1.1, thời gian thực thi lệnh của một vi điều khiển là rất nhanh so với tốc độ cảm nhận sự vật hiện tượng của con người. Điều này gây nhiều khó khăn cho việc “giao tiếp” giữa con người với một vi điều khiển cũng như khó khăn trong việc cảm nhận bằng giác quan kết quả các thao tác của một vi điều khiển.

Ví dụ, ta dùng vi điều khiển để điều khiển một LED chớp tắt liên tục. Với thao tác này vi điều khiển chỉ cần hai chu kì lệnh là hoàn tất một chu kì chớp tắt, và thời gian của mỗi chu kì sẽ là 2  $\mu s$  (khi sử dụng thạch anh 4 MHz), và trong một giây, LED sẽ chớp tắt 500000 lần. Trong khi mắt người chỉ có thể nhận biết được 24 hình ảnh trong một giây. Điều này có nghĩa là, một người ngoài hành tinh, với con mắt có tốc độ xử lí hình ảnh nhanh hơn, khi chứng kiến hiện tượng trên sẽ nói rằng: “Eh, người trái đất, tôi thấy có cái gì đó đang *chớp tắt*”. Còn người trái đất, với tốc độ xử lí hình ảnh của mắt là 24 hình trong 1 giây, khi chứng kiến hiện tượng trên sẽ nói rằng: “Không, người ngoài hành tinh, tôi thấy *nó sáng liên tục* đó chứ!”.

<b>Người báo cáo:</b>	Nguyễn Trung Chính	<b>Tài liệu:</b>	TUT02.02
<b>Ngày:</b>	1/30/2006	<b>Trang:</b>	3/18

Như vậy, làm sao để mắt người cảm nhận được LED đang chớp tắt, cách duy nhất là phải giảm số lần chớp tắt trong 1 giây nhỏ hơn 24, các thao tác để vi điều khiển hiển thị cho con người thấy được hiện tượng trên lần lượt sẽ là:

- Bật LED sáng lên
- Chờ một chút cho tới khi mắt nhận được hình ảnh LED sáng.
- Tắt LED
- Chờ một chút cho tới khi mắt nhận được hình ảnh LED tắt.
- Lặp lại các thao tác trên.

Như ta đã biết, do vi điều khiển không có cái lệnh gọi là “chờ một chút”, cho nên khái niệm chương trình delay mới được phát sinh để thực hiện quá trình chờ đó.

*Có thể nói chương trình delay đóng một vai trò quan trọng trong các thao tác hiển thị. Bên cạnh đó, chương trình delay còn có vai trò quan trọng trong việc giao tiếp với các thiết bị khác, khi mà tốc độ xử lý của vi điều khiển và các thiết bị không đồng nhất. Ngoài ra, ta còn sử dụng chương trình delay trong nhiều tình huống thực tế cần ra lệnh cho vi điều khiển phải chờ.*

## 2. Xây dựng chương trình delay

### 2.1. Các lệnh sử dụng cho chương trình delay

Ngoài các lệnh đã được đề cập đến trong bài 1, ta cần sử dụng thêm các lệnh sau cho chương trình delay:

#### **Lệnh DECFSZ**

*Cú pháp:* DECFSZ      thanh\_ghi,noi\_den

Lệnh 1

Lệnh 2

*Tác dụng:* Giảm giá trị chứa trong tham số “thanh\_ghi” và so sánh với 0.

- Nếu giá trị sau khi giảm khác 0, lệnh 1 được thực thi.
- Nếu giá trị sau khi giảm bằng 0, lệnh 1 không được thực thi và được thay bằng lệnh NOP (không làm gì cả).

Tham số “noi\_den” dùng để xác định nơi lưu giá trị thanh ghi “thanh\_ghi” sau khi giảm. Khi không sử dụng tham số “noi\_den”, trình biên dịch sẽ mặc định là kết quả được chứa trong thanh ghi W.

- Nếu tham số “noi\_den” bằng 1, kết quả được chứa trong thanh ghi “thanh\_ghi”.
- Nếu tham số “noi\_den” bằng 0, kết quả được chứa trong thanh ghi W.

Người báo cáo:	Nguyễn Trung Chính	Tài liệu:	TUT02.02
Ngày:	1/30/2006	Trang:	4/18

## Lệnh RETURN

Cú pháp: RETURN

Tác dụng: trở về chương trình chính từ chương trình con.

## Lệnh RETLW

Cú pháp: RETLW tham\_so ( $0 \leq \text{tham\_so} \leq 255$ )

Tác dụng: trở về chương trình chính từ chương trình con với giá trị tham\_so được chứa trong thanh ghi W.

## 2.2. Thuật toán cho chương trình delay

Ta đã biết ở phần 1, *chương trình delay là chương trình dùng để ra lệnh cho vi điều khiển ... “chờ một chút”* (tạm thời định nghĩa một cách ... đại khái như vậy). *Điều này cũng đồng nghĩa với việc ra lệnh cho vi điều khiển làm một công việc vô nghĩa nào đó trong một khoảng thời gian do ta quyết định.*

Trong tập lệnh của vi điều khiển PIC, ta có lệnh NOP. Lệnh này có tác dụng ra lệnh cho vi điều khiển ... không làm gì cả, và thời gian thực thi lệnh này cũng là 1 chu kỳ lệnh. Như vậy, ta có cần thiết phải xây dựng thuật toán cho chương trình delay, vì chỉ cần ... “NOP” liên tục là xong? Hoàn toàn không đơn giản như vậy, vì khi đó ta sẽ gặp phải các vấn đề sau:

- Thứ nhất, cái thuật toán có vẻ ... không bình thường.
- Thứ hai, viết chương trình như vậy thì rất mỏi tay (muốn ra lệnh cho vi điều khiển chờ 1 ms, bạn phải viết đi viết lại cái lệnh NOP ... 1000 lần nếu sử dụng loại thạch anh 4 MHz).
- Thứ ba, dung lượng bộ nhớ chương trình bị phí phạm một cách ... quá đáng.

Rõ ràng là ta không thể viết chương trình delay theo cách đó. Và việc khắc phục tất cả các nhược điểm nêu trên cũng là *các tiêu chí đặt ra cho một chương trình delay, đó là: ngắn gọn và thuận tiện cho việc sử dụng.*

*Một phương pháp thường sử dụng để viết các chương trình delay là cho vi điều khiển ... nhảy tới nhảy lui mấy cái label.* Tuy nhiên để kiểm soát được thời gian delay do chương trình tạo ra, ta cần tính toán các giá trị trong chương trình một cách phù hợp.

Sau đây ta sẽ đi sâu vào các thuật toán dùng để viết chương trình delay này.

### 2.2.1 Thuật toán 1

Trong thuật toán này ta sử dụng lệnh DECFSZ để xây dựng chương trình delay.

Đoạn chương trình 1: xét một đoạn code như sau

Người báo cáo:	Nguyễn Trung Chính	Tài liệu:	TUT02.02
Ngày:	1/30/2006	Trang:	5/18

```

MOV LW    d'20'      ; đưa giá trị 20 vào thanh ghi W
MOV WREG delay-reg  ; delay-reg <- 20
loop
DECFSZ    delay-reg,0 ; giảm giá trị trong thanh ghi delay-reg
                        ; và so sánh với 0, kết quả chứa trong W
GOTO     loop        ; nếu giá trị thanh ghi "delay_reg" khác 0
                        ; thì nhảy tới label "loop"
; các lệnh tiếp theo sau đoạn chương trình delay sau
; khi giá trị trong thanh ghi "delay-reg" đã giảm về 0.

```

*Đoạn chương trình delay được thể hiện trong vòng lặp "loop". Ta thấy lệnh "DECFSZ ..." cần một chu kì lệnh để thực thi, lệnh "GOTO ..." cần 2 chu kì lệnh, khi đó giá trị trong thanh ghi "delay-reg" sẽ bị giảm đi một đơn vị. Như vậy để giá trị trong thanh ghi "delay-reg" giảm một đơn vị, ta cần (1 + 2) = 3 chu kì lệnh và quãng thời gian cần thiết để giá trị trong thanh ghi "delay-reg" giảm một đơn vị sẽ là 3t<sub>i</sub> (t<sub>i</sub> như đã đề cập đến trong phần trên là thời gian của một chu kì lệnh).*

Trong ví dụ trên, do ta đưa vào thanh ghi delay-reg giá trị 20 cho nên số lần giảm giá trị thanh ghi "delay-reg" sẽ là (20 + 1) = 21. Ta có thể tính được thời gian delay T do đoạn chương trình trên tạo ra sẽ là:

$$T = 3 \times (20 + 1) \times t_i$$

Ví dụ, nếu ta sử dụng loại thạch anh 4 MHz thì một chu kì lệnh sẽ có thời gian t<sub>i</sub> = 1 μs, do đó đoạn chương trình trên sẽ tạo ra khoảng thời gian delay:

$$T = 3 \times (20 + 1) \times 1 \mu s = 63 \mu s$$

Một cách tổng quát, ta có thể suy ra được công thức tính thời gian delay cho đoạn chương trình trên như sau:

$$T = 3 \times (N + 1) t_i \quad (4)$$

Trong đó N là giá trị đưa vào thanh ghi "delay-reg".

Đến đây ta đã có thể hình dung được một cách sơ lược cách tính toán thời gian delay T của một chương trình delay. *Thời gian T này sẽ phụ thuộc vào cấu trúc giải thuật chương trình delay và thời gian một chu kì lệnh t<sub>i</sub>.*

<b>Người báo cáo:</b>	Nguyễn Trung Chính	<b>Tài liệu:</b>	TUT02.02
<b>Ngày:</b>	1/30/2006	<b>Trang:</b>	6/18

Một điểm cần chú ý thông thường các thanh ghi ta sử dụng là thanh ghi 8 bit, cho nên giá trị tối đa có thể đưa vào một thanh ghi là 255. Vậy thời gian delay lớn nhất mà đoạn chương trình delay trên có thể tạo ra là:

$$T_{\max} = 3 \times (255 + 1) t_i$$

Muốn tạo thời gian delay lâu hơn, ta phải tăng số lượng các vòng lặp lên. Đoạn chương trình sau minh họa cách tăng số lượng vòng lặp cho chương trình delay:

**Đoạn chương trình 2:**

```

MOV LW    d'255'
MOV WRF  delay-reg1    ; đưa giá trị 255 vào thanh ghi "delay-reg1"
loop
DECFSZ   delay-reg1,0  ; giảm giá trị thanh ghi "delay-reg1" và so sánh với 0
GOTO     loop1         ; nếu chưa bằng 0 nhảy tới label "loop1"
GOTO     next          ; nếu đã bằng 0 chương trình delay hoàn tất
loop1
MOV LW    d'255'
MOV WRF  delay-reg2    ; đưa vào thanh ghi "delay-reg2" giá trị 255
DECFSZ   delay-reg2,0  ; giảm giá trị thanh ghi "delay-reg2" và so sánh với 0
MOV WRF  loop1         ; nếu chưa bằng 0 thì nhảy đến label "loop1"
GOTO     loop          ; nếu bằng 0 thì nhảy đến label "loop"
next
.....

```

Ta xét đoạn chương trình từ label "loop1" trước. Đoạn chương trình này tương tự như đoạn chương trình 1, cho nên cách tính thời gian delay trong đoạn chương trình này không có gì thay đổi. Giá trị N trong công thức 4 sẽ tương ứng với giá trị N<sub>2</sub> đưa vào thanh ghi "delay-reg2" (255). Ta gọi T<sub>2</sub> là thời gian delay do đoạn chương trình này tạo ra thì T<sub>2</sub> sẽ được tính như sau:

$$T_2 = 3 \times (N_2 + 1) t_i \quad (5)$$

Khi giá trị trong thanh ghi "delay-reg2" giảm về 0 thì các lệnh từ label "loop" được thực thi. Ở thời điểm này giá trị trong thanh ghi "delay-reg1" sẽ giảm đi một đơn vị và tiếp tục thực thi vòng lặp "loop1". Như vậy sau một khoảng thời gian T<sub>2</sub>, giá trị trong thanh ghi "delay-reg1" sẽ giảm đi một đơn vị, và nếu ta gọi N<sub>1</sub> là giá trị đưa vào thanh ghi "delay-reg1" thì số lần giảm giá trị trong thanh ghi "delay-reg1" sẽ là (N<sub>1</sub> + 1). Như vậy thời gian delay T do đoạn chương trình 2 tạo ra là:

$$T = (N_1 + 1) T_2 = 3 \times (N_1 + 1) \times (N_2 + 1) \times t_i \quad (6)$$

Dựa theo các giá trị đưa vào trong đoạn chương trình 2 ta có thể tính được thời gian delay do đoạn chương trình trên tạo ra như sau:

Người báo cáo:	Nguyễn Trung Chính	Tài liệu:	TUT02.02
Ngày:	1/30/2006	Trang:	7/18

$$T = 3 \times (255+1) \times (255+1) t_i = 196608 t_i$$

Nếu sử dụng loại thạch anh 4 MHz thì thời gian delay do đoạn chương trình trên tạo ra là 196608  $\mu$ s.

Như vậy, tùy vào thời gian delay cần thiết và tùy vào loại thạch anh sử dụng trong mạch mà ta có thể đưa các giá trị  $N_1$  và  $N_2$  vào các thanh ghi "delay-reg1" và "delay-reg2" một cách thích hợp dựa vào công thức (6).

*Ví dụ: tính toán các giá trị đưa vào thanh ghi "delay-reg1" và "delay-reg2" để thời gian delay do đoạn chương trình 2 tạo ra là 90 ms. Giả sử ta đang sử dụng loại thạch anh 4 MHz.*

Ta giải bài toán như sau: do loại thạch anh ta sử dụng có tần số 4 MHz nên  $t_i = 1 \mu$ s. Do đó ta có

$$(N_1+1) \times (N_2+1) = T/3t_i = 90 \times 10^{-3} / (3 \times 1 \times 10^{-6}) = 30 \times 10^3$$

Nếu chọn giá trị đưa vào thanh ghi "delay-reg2" là  $N_2 = 199$  thì giá trị  $N_1$  đưa vào thanh ghi "delay-reg1" sẽ là:

$$N_1 = 30 \times 10^3 / (199+1) - 1 = 149$$

*Một điểm cần chú ý là bên cạnh việc thỏa mãn công thức (6), các giá trị  $N_1$  và  $N_2$  phải thỏa mãn điều kiện:*

$$0 < N_1 < 256 \text{ và } 0 < N_2 < 256 \quad (7)$$

Thuật toán trên cho phép ta giải quyết khá triệt để các vấn đề dành cho một chương trình delay. Tuy nhiên, *nhược điểm của thuật toán trên là: trong trường hợp cần nhiều thời gian delay khác nhau, ta phải viết nhiều chương trình delay khác nhau tương ứng.* Thuật toán 2 cho chương trình delay được phát triển dựa trên thuật toán 1 cho phép khắc phục nhược điểm trên và sẽ được trình bày cụ thể ở phần tiếp theo.

## 2.2.2 Thuật toán 2

Các bạn có thể dễ dàng nhận ra đây là thuật toán cho chương trình delay được sử dụng trong tutorial của Nigel. Phần này sẽ phân tích cụ thể giải thuật và source code của đoạn chương trình delay này. Và để thể hiện thái độ tôn trọng tác giả, tutorial này vẫn giữ nguyên mã lệnh như trong tutorial của Nigel.

*Giả sử ta đang sử dụng loại thạch anh 4 MHz.* Ta xét đoạn code sau:

Đoạn chương trình 3:

```

MOV LW    d'90'
MOV WFP   count1      ; đưa giá trị 90 vào thanh ghi count1
d1
MOV LW    d'199'
```

<b>Người báo cáo:</b>	Nguyễn Trung Chính	<b>Tài liệu:</b>	TUT02.02
<b>Ngày:</b>	1/30/2006	<b>Trang:</b>	8/18

```

MOVWF    counta    ; đưa giá trị 199 vào thanh ghi counta
MOVLW    d'1'
MOVWF    countb    ; đưa giá trị 1 vào thanh ghi countb
delay_0
DECFSZ   counta,1  ; giảm giá trị trong thanh ghi counta và so sánh với 0
GOTO     $+2        ; nếu chưa bằng 0, nhảy tới lệnh "GOTO delay_0"
DECFSZ   countb,1  ; nếu bằng 0, giảm giá trị trong thanh ghi countb
GOTO     delay_0    ; countb sau khi giảm có giá trị bằng 0 nên lệnh này
                  ; không được thực thi
DECFSZ   count1,1  ; giảm giá trị trong thanh ghi count1
GOTO     d1         ; nhảy về label d1
-----        ; các lệnh tiếp theo của chương trình chính sau đoạn
                  ; chương trình delay

```

Trước tiên ta lưu ý đến lệnh "GOTO \$+2". Lệnh này có tác dụng nhảy tới lệnh thứ hai kể từ dòng lệnh "GOTO \$+2", tức là nhảy đến lệnh "GOTO delay\_0". Hoàn toàn tương tự ta có thể dùng lệnh có cấu trúc tương tự để nhảy đến bất cứ dòng lệnh nào trong chương trình thông qua việc thay thế hằng số sau dấu "\$".

Ta xét đoạn code bắt đầu từ label "delay\_0" trước. Lệnh DECFSZ counta,1" mất một chu kì lệnh để thực thi. Nếu giá trị chứa trong thanh ghi counta chưa bằng 0 thì lệnh "GOTO \$+2" được thực thi. Lệnh này mất hai chu kì lệnh. Tiếp theo, lệnh "GOTO delay\_0" được thực thi. Lệnh này cũng mất hai chu kì lệnh. Sau đó, giá trị trong thanh ghi counta tiếp tục được giảm. Đến đây ta nhận thấy rằng, để giảm một giá trị trong thanh ghi counta, ta mất hết 5 chu kì lệnh (1 chu kì lệnh cho lệnh DECFSZ counta,1", 2 chu kì lệnh cho lệnh "GOTO \$+2" và 2 chu kì lệnh cho lệnh "GOTO delay\_0"), và do giá trị đưa vào thanh ghi counta là 199 nên thời gian cần thiết để thanh ghi counta giảm hết giá trị về 0 là:

$$T_a = 5 \times (199 + 1) \times t_i$$

Do ta đang sử dụng loại thạch anh 4 MHz nên  $T_a$  sẽ mang giá trị 1000  $\mu$ s hay 1 ms.

Khi giá trị trong thanh ghi counta bằng 0, lệnh "GOTO \$+2" sẽ không được thực thi mà thay vào đó là lệnh NOP, tiếp theo lệnh "DECFSZ countb,1" sẽ được thực thi. Ta thấy giá trị đưa vào thanh ghi countb là 1 nên sau khi giảm countb sẽ bằng 0 nên lệnh "GOTO delay\_0" sẽ được thay bằng lệnh NOP và tiếp theo, lệnh "DECFSZ count1,1" sẽ

<b>Người báo cáo:</b>	Nguyễn Trung Chính	<b>Tài liệu:</b>	TUT02.02
<b>Ngày:</b>	1/30/2006	<b>Trang:</b>	9/18

được thực thi. Sau đó chương trình quay trở về label “d1” để thực hiện việc nạp lại các giá trị cho thanh ghi counta, countb và tiếp tục thực thi đoạn code từ label “delay\_0”.

Như vậy việc đưa giá trị 1 vào thanh ghi countb thực chất chỉ là để thực hiện quá trình chuyển tiếp mỗi khi thanh ghi counta giảm về 0. Và đoạn code từ label “delay\_0” thực chất là để tạo ra thời gian delay *gần đúng 1ms* do ta đã bỏ qua một số chu kì lệnh trong bước chuyển tiếp (lưu ý một lần nữa là ta đang sử dụng loại thạch anh 4 MHz), sau đó giá trị trong thanh ghi count1 được giảm 1 đơn vị. Vòng lặp cứ tiếp tục cho đến khi giá trị trong thanh ghi count1 được giảm về 0. Khi đó lệnh “GOTO d1” không được thực thi nữa và quá trình tạo thời gian delay kết thúc, các lệnh tiếp theo trong chương trình chính sẽ tiếp tục được thực thi.

Đến đây ta có thể nhận thấy rằng cứ mỗi 1 ms thì giá trị trong thanh ghi count1 sẽ giảm đi 1 đơn vị. Do đó, muốn tạo ra bất cứ một thời gian delay nào là bội số của 1 ms, ta chỉ việc đưa giá trị tương ứng vào thanh ghi count1. Trong ví dụ ở đoạn chương trình 3, do ta đưa vào thanh ghi count1 giá trị 90 nên thời gian delay sẽ là 90 ms. Hoàn toàn tương tự cho việc tạo ra thời gian delay 10 ms, 50 ms, 100 ms, 150 ms, 200 ms, ...ta cũng dễ dàng nhận thấy là thời gian delay tối đa do đoạn chương trình trên tạo ra là 255 ms. Với các thao tác thông thường dành cho vi điều khiển, có thể nói đây là thời gian delay đủ lớn để ta có thể sử dụng.

Thuật toán 2 tuy dài hơn và sử dụng nhiều thanh ghi hơn so với thuật toán 1 nhưng nó có nhiều ưu điểm hơn thuật toán 1 do tính linh động và dễ sử dụng của nó. Ta có thể sử dụng đoạn chương trình delay này như một chương trình delay mẫu cho việc xây dựng các ứng dụng cho vi điều khiển PIC.

Trong trường hợp sử dụng loại thạch anh có tần số cao hơn, ta có thể kết hợp hai thuật toán 1 và 2 để tạo ra thời gian delay mong muốn.

### 3. Ứng dụng

Trong các phần trên, ta đã có thể hình dung được mục đích, tác dụng và một số giải thuật cho việc xây dựng một chương trình delay. Bây giờ là lúc sử dụng các kiến thức đó cho các ứng dụng cụ thể.

- Ứng dụng 1:

Ta sẽ phát triển ứng dụng đầu tiên cho chương trình delay từ mạch nguyên lí và chương trình đã được xây dựng trong bài 1. *Trong bài 1, ta đã thực hiện viết xuất các giá trị ra PORTB và kiểm chứng bằng các LED gắn vào PORTB. Bây giờ ta sẽ viết chương trình cho tất cả các LED gắn vào PORTB chớp tắt sau mỗi khoảng thời gian 100 ms.*

Giải thuật cho chương trình chắc cũng không có gì phải đáng bàn, các bước thực hiện lần lượt sẽ là:

<b>Người báo cáo:</b>	Nguyễn Trung Chính	<b>Tài liệu:</b>	TUT02.02
<b>Ngày:</b>	1/30/2006	<b>Trang:</b>	10/18

- Bật tất cả các LED ở PORTB
- Delay 100 ms
- Tắt tất cả các LED ở PORTB
- Delay 100 ms
- Lặp lại các thao tác trên

Chương trình sẽ được viết như sau:

**Chương trình 2.1:**

```

=====
; WWW.PICVIETNAM.COM
; Lap trinh:          NGUYEN TRUNG CHINH
; Ngay bat dau:      23 thang 01 nam 2006
; Ngay hoan thanh:   23 thang 01 nam 2006
; Kiem tra chuong trinh: Doan Hiep, Doan Minh Dang,
;                   picvietnam@googlegroups.com
; Ngaykiem tra:
; Su dung vi dieu khien Microchip:    PIC16F877A
    title          "chuongtrinh2-1.asm"
    processor      16f877a
    include        <p16f877a.inc>
    __CONFIG      _CP_OFF & _WDT_OFF & _BODEN_OFF & _PWRTE_ON &
                 _XT_OSC & _WRT_OFF & _LVP_OFF & _CPD_OFF

; Cap nhat va bo sung:
; Mo ta chuong trinh:    Chuong trinh dung de dieu khien tat ca cac LED gan vao
;                   PORTB chop tat lien tục sau moi khoang thoi gian 100 ms.
;
;                   Khong su dung chuong trinh con
; Mo taphan cung:       8 LED duoc gan vao PORTB thong qua cac dien tro, cac
;                   thanhphan di kem bao gom thach anh, mach reset va nguon
=====

```

<b>Người báo cáo:</b>	Nguyễn Trung Chính	<b>Tài liệu:</b>	TUT02.02
<b>Ngày:</b>	1/30/2006	<b>Trang:</b>	11/18

```

;-----
; Khai tao cac bien
;-----
count1      EQU  0x20      ; cac bien dung cho doan chuong trinh delay
counta      EQU  0x21
countb      EQU  0x22

;=====
;CHUONG TRINH CHINH
;=====

      ORG      0x000
      GOTO    start

start

;-----
; Khai tao PORTB
;-----

      BCF     STATUS,RP1
      BSF     STATUS,RP0      ; chon BANK 1

      CLRF   TRISB           ; PORTB <- output

      BCF     STATUS,RP0      ; chon BANK 0

;-----
; Vong lap chinh
;-----
loop
      MOVLW  0xFF
      MOVWF  PORTB           ; bat tat ca cac LED o PORTB

      MOVLW  d'100'         ; doan chuong trinh tao thoi gian delay 100 ms
      MOVWF  count1

```

<b>Người báo cáo:</b>	Nguyễn Trung Chính	<b>Tài liệu:</b>	TUT02.02
<b>Ngày:</b>	1/30/2006	<b>Trang:</b>	12/18

d1\_1

```

MOVLW    d'199'
MOVWF    counta
MOVLW    d'1'
MOVWF    countb

```

delay\_01

```

DECFSZ   counta,1
GOTO     $+2
DECFSZ   countb,1
GOTO     delay_01
DECFSZ   count1,1
GOTO     d1_1           ; het doan chuong trinh delay

CLRF     PORTB         ; tat cac LED o PORTB

MOVLW    d'100'        ; doan chuong trinh delay 100 ms
MOVWF    count1

```

d1\_2

```

MOVLW    d'199'
MOVWF    counta
MOVLW    d'1'
MOVWF    countb

```

delay\_02

```

DECFSZ   counta,1
GOTO     $+2
DECFSZ   countb,1

```

<b>Người báo cáo:</b>	Nguyễn Trung Chính	<b>Tài liệu:</b>	TUT02.02
<b>Ngày:</b>	1/30/2006	<b>Trang:</b>	13/18

```

GOTO    delay_02
DECFSZ  count1,1
GOTO    d1_2           ; het doan chuong trinh delay 1 ms

GOTO    loop           ; tro ve vong lap chinh cua chuong trinh
END

```

Với chương trình trên, ta có thể quan sát các hiện tượng do vi điều khiển tạo ra ở PORTB thông qua các LED. Do thời gian delay là 100 ms cho nên trong 1 giây trạng thái của LED sẽ thay đổi 10 lần. Điều này cho phép ta quan sát được bằng mắt thường. bây giờ, các bạn hãy thử giảm thời gian delay xuống nhỏ dần (giảm giá trị đưa vào thanh ghi count1), để xem hiện tượng gì sẽ xảy ra. Khi thời gian delay giảm đến một giá trị nào đó, ta sẽ có cảm giác rằng các LED không còn chớp tắt nữa, mà sẽ sáng một cách liên tục.

Ta dễ dàng nhận thấy một nhược điểm trong chương trình trên là phải viết đi viết lại chương trình delay đến hai lần, và một lần nữa, vấn đề về dung lượng bộ nhớ chương trình được đặt ra. Một giải pháp để khắc phục nhược điểm trên, đó là *chương trình con*.

*Một chương trình con có thể tạm hiểu là một đoạn code nào đó được lặp đi lặp lại nhiều lần trong chương trình chính, và thay vì phải viết đi viết lại đoạn code đó nhiều lần, ta tổ chức đoạn code đó thành một chương trình con và gọi đoạn code đó từ chương trình chính thông qua lệnh "CALL". Một chương trình con sẽ bắt đầu bằng 1 label và kết thúc bằng lệnh RETURN hoặc lệnh RETLW.*

Có thể nói chương trình con giúp ta có nhiều phương án hơn trong việc tổ chức một chương trình viết cho vi điều khiển.

Bây giờ, ta sẽ tổ chức lại chương trình 2.1 thành một chương trình mới bằng cách sử dụng chương trình con cho đoạn code tạo thời gian delay 100 ms. Chương trình mới sẽ được viết như sau:

### Chương trình 2.2:

---

```

; WWW.PICVIETNAM.COM
; Lap trinh:           NGUYEN TRUNG CHINH
; Ngay bat dau:       23 thang 01 nam 2006
; Ngay hoan thanh:   23 thang 01 nam 2006

```

<b>Người báo cáo:</b>	Nguyễn Trung Chính	<b>Tài liệu:</b>	TUT02.02
<b>Ngày:</b>	1/30/2006	<b>Trang:</b>	14/18

```

; Kiem tra chuong trinh:   Doan Hiep, Doan Minh Dang,
;
;                           picvietnam@googlegroups.com
; Ngaykiem tra:
; Su dung vi dieu khien Microchip:   PIC16F877A
;
; title   "chuongtrinh2-2.asm"
; processor 16f877a
; include <p16f877a.inc>
;
; __CONFIG _CP_OFF & _WDT_OFF & _BODEN_OFF & _PWRTE_ON &
;          _XT_OSC & _WRT_OFF & _LVP_OFF & _CPD_OFF
;
; Cap nhat va bo sung:
;
; Mo ta chuong trinh:   Chuong trinh dung de dieu khien tat ca cac LED gan vao
;
;                           PORTB chop tat lien tục sau moi khoang thoi gian 100 ms.
;
;                           Co su dung chuong trinh con
;
; Mo ta phan cung:   8 LED duoc gan vao PORTB thong qua cac dien tro, cac
;
;                           thanh phan di kem bao gom thach anh, mach reset va nguon
;
;=====
;-----
; Khoi tao cac bien
;-----
count1          EQU 0x20          ; cac bien dung cho doan chuong trinh delay
counta          EQU 0x21
countb          EQU 0x22
;=====
;CHUONG TRINH CHINH
;=====
ORG             0x000
GOTO            start
start

```

<b>Người báo cáo:</b>	Nguyễn Trung Chính	<b>Tài liệu:</b>	TUT02.02
<b>Ngày:</b>	1/30/2006	<b>Trang:</b>	15/18

;-----

; Khai tạo PORTB

;-----

BCF STATUS,RP1

BSF STATUS,RP0 ; chọn BANK 1

CLRF TRISB ; PORTB <- output

BCF STATUS,RP0 ; chọn BANK 0

;-----

; Vòng lặp chính

;-----

loop

MOVLW 0XFF

MOVWF PORTB ; bật tất cả các LED ở PORTB

CALL delay\_100ms ; gọi chương trình con tạo thời gian delay 100 ms

CLRF PORTB ; tắt các LED ở PORTB

CALL delay\_100ms ; gọi chương trình con tạo thời gian delay 100 ms

GOTO loop ; trở về vòng lặp chính của chương trình

=====

; CHUONG TRINH CON

=====

delay\_100ms ; label bắt đầu chương trình con

MOVLW d'100' ; đoạn code cho chương trình con delay 100 ms

MOVWF count1

d1

MOVLW d'199'

MOVWF counta

MOVLW d'1'

MOVWF countb



<b>Người báo cáo:</b>	Nguyễn Trung Chính	<b>Tài liệu:</b>	TUT02.02
<b>Ngày:</b>	1/30/2006	<b>Trang:</b>	17/18

```

GOTO    delay
delay
MOVWF   count1
d1
MOVLW   d'199'
MOVWF   counta
MOVLW   d'1'
MOVWF   countb
delay_0
DECFSZ  counta,1
GOTO    $+2
DECFSZ  countb,1
GOTO    delay_0
DECFSZ  count1,1
GOTO    d1
RETURN                                     ;ket thuc chuong trinh con, tro ve chuong trinh chinh

```

Như ta đã biết ở phần 2, với thuật toán 2, muốn thay đổi thời gian delay cho chương trình delay, ta chỉ việc thay đổi giá trị đưa vào thanh ghi count1. Ở đây ta cũng làm thao tác tương tự. Đoạn code từ label "delay" được giữ nguyên không thay đổi. Khi gọi chương trình con delay\_100ms, giá trị 100 sẽ được đưa vào thanh ghi W, sau đó nhảy tới label "delay" để đưa giá trị đó vào thanh ghi "count1" để tiếp tục thực hiện việc tạo thời gian delay. Các thao tác được tiến hành tương tự như khi gọi chương trình con delay\_200ms và lúc đó giá trị được đưa vào thanh ghi W sẽ là 200.

Hoàn toàn tương tự ta có thể tạo ra một loạt những chương trình delay 1 ms, 2 ms, 5, ms, ... để sử dụng một cách dễ dàng tùy theo yêu cầu về chương trình delay của ứng dụng cụ thể.

Đến đây xem như ta đã phát triển một cách khá hoàn thiện về các giải pháp cho chương trình delay thông qua việc xây dựng chương trình con delay và hiểu được cách tạo nhiều thời gian delay khác nhau trong cùng một chương trình mà không cần phải viết đi viết lại nhiều chương trình delay.

Bây giờ là lúc rút ra một vài kết luận trước khi kết thúc bài 2 và chuẩn bị cho bài 3.

<b>Người báo cáo:</b> Nguyễn Trung Chính	<b>Tài liệu:</b> TUT02.02
<b>Ngày:</b> 1/30/2006	<b>Trang:</b> 18/18

#### 4. Kết luận

Chương trình delay đóng một vai trò khá quan trọng trong việc phát triển các ứng dụng cho vi điều khiển. Chương trình delay được sử dụng nhiều trong các thao tác hiển thị và trong các ứng dụng cần ra lệnh cho vi điều khiển phải chờ.

Các thuật toán dùng để xây dựng chương trình delay phải thỏa mãn các tiêu chí ngắn gọn và thuận tiện cho việc sử dụng, đồng thời giúp ta kiểm soát được thời gian delay do đoạn chương trình tạo ra.

Thời gian delay do chương trình delay tạo ra sẽ phụ thuộc vào giải thuật sử dụng cho chương trình delay và loại thạch anh sử dụng cho vi điều khiển.

Chương trình con giúp ta có được nhiều phương án tổ chức một chương trình ứng dụng một cách linh hoạt, gọn gàng và dễ hiểu hơn.

Hết bài 2!