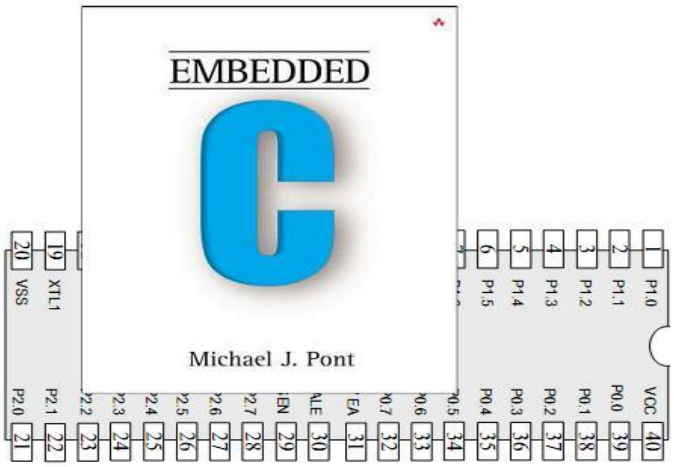
Lập Trình Hệ Thống Nhúng
10 tuần học ngôn ngữ c



Michael J. Pont
University of Leicester

Thông tin chi tiết:
http://www.le.ac.uk/engineering/mjp9/

Bản dịch đầu tay không thể thiếu những sai sót. Rất mong nhận được sự góp ý của các bạn.
Mọi thắc mắc về bản dịch xin vui lòng gửi tới :

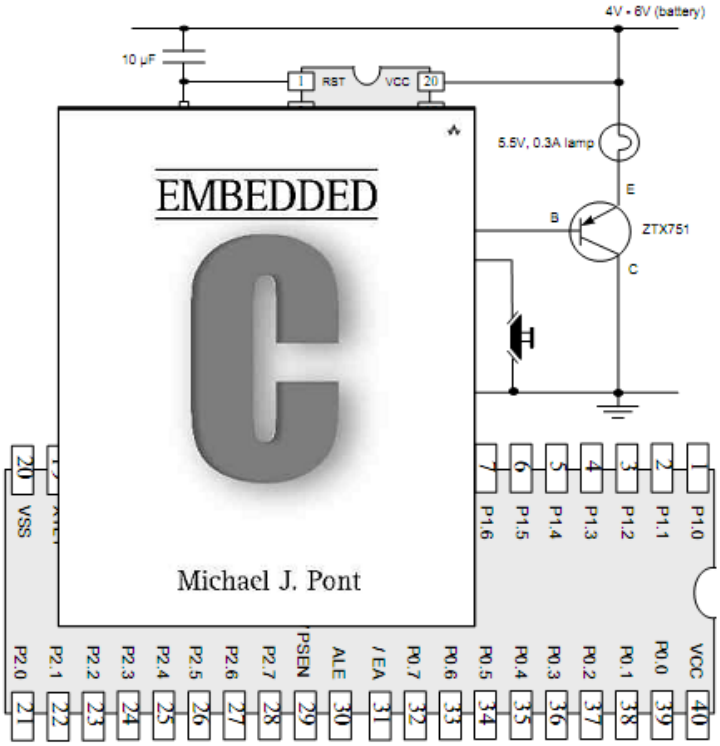
doanhau198@gmail.com

tranthien.pro@gmail.com

Atienganh.blogspot.com

Mục lục

Chương 1: Xin chào"thế giới Nhúng"



Khái quát kiến thức :

Chương mở đầu này sẽ:

• Cung cấp khái quát về khóa học này

• Giới thiệu VĐK 8051

• Cấu trúc phần mềm "siêu vòng lặp"

• Mô tả cách sử dụng các chân điều khiển

• Bạn tạo ra hàm trễ như thế nào? Và tại sao bạn cần nó?

Khái quát về khóa học này

Khóa học này có liên quan đến sự thực hiện của phần mềm (và một lượng nhỏ phần cứng) cho việc xây dựng hệ thống nhúng sử dụng 1 VĐK.

Các bộ vi xử lý được kiểm tra chi tiết là họ 8051 (Bao gồm cả thiết bị chuẩn và thiết bị nhỏ).

Tất cả lập trình ngôn ngữ C.

Kết thúc khóa học …

Kết thúc khóa học, bạn có thể:

1. Thiết kế phần mềm cho bộ xử lý nhúng ứng dụng đơn, trên cơ sở nhỏ,tiêu chuẩn công nghiệp.

2. Thực hiện thiết kế trên bằng cách sử dụng ngôn ngữ lập trình bậc cao, hiện đại

3. Bắt đầu hiểu các vấn đề về độ tin cậy,an toàn,làm thế nào để thiết kế phần mềm và chương trình quyết định có thể có một tác động tích cực hoặc tiêu cực lên một phân vùng.

Sách tham khảo

Trong suốt khóa học bạn có thể tham khảo cuốn sách sau:

Embedded C
by Michael J. Pont (2002)

Addison-Wesley
[ISBN: 0-201-79523X]

Thông tin chi tiết tại:

http://www.engg.le.ac.uk/books/Pont/ec51.htm

:

Tại sao dùng ngôn ngữ C?

• Đây là ngôn ngữ ở mức giữa, với các tính năng của bậc cao: hỗ trợ cho các chức năng và các modul, và các tính năng bậc thấp: truy cập vào phân cứng thông qua con trỏ

• Hiệu quả cao.

• Phổ biến và dễ hiểu.

• Thậm chí những người phát triển máy tính để bàn chỉ sử dụng được Java và C++ cũng dễ dàng hiểu cú pháp C.

• Tốt, trình biên dịch đã có cho mọi vi xử lý nhúng, (8-bit to 32-bit or more) .

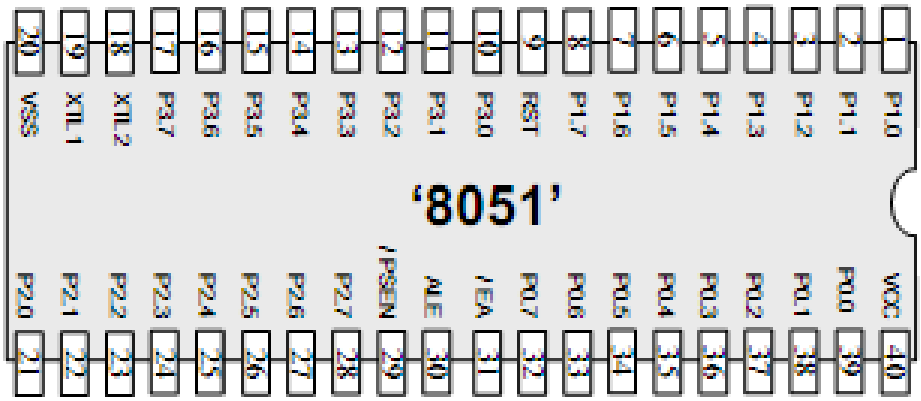
• Sách, các khóa đào tạo, code ví dụ và các trang wed thảo luận rộng rãi trên toàn thế giới.

Nói chung ngôn ngữ C có thể không là một ngôn ngữ hoàn hảo để phát triển hệ thống nhúng, nhưng nó là một sự lựa chọn tốt (và không chắc có một ngôn ngữ hoàn hảo được tạo ra.:-d).

Cần biết những gì?

• Trong suốt khóa học này, nó sẽ giả định rằng bạn đã có kinh nghiệm lập trình. Ví dụ - Java or C++.
• Mọi người đều có nền tảng về lập trình rồi thì việc học C quả thật đơn giản

The 8051 microcontroller



Tính năng điển hình của 8051:

• 32 chân I/O

• Bộ nhớ trong (RAM) - 256 bytes.

• Nâng cấp lên tới 64 kbytes of ROM (Bộ nhớ flash)

• 3 timer/counter 16-bit

• 9 chế độ ngắt (2 ngắt ngoài) với 2 mức ưu tiên.

• Nguồn tiêu thụ thấp.

Các loại khác trong họ 8051 thích hợp cho mọi thứ từ hệ thống ôtô và hàng không đến điều khiển TV.

Cấu trúc "siêu vòng lặp"

Vấn đề:

Môi trường phần mềm tối thiểu bạn cần để tạo 1 chương trình C là gì?

Giải quyết:

```
void main(void)
    {
    /* chuan bi cho nhiem vu X  */
    X_Init();
    while(1)  /* vong lap mai mai  */
        {
        X();      /* thuc hien nhiem vu X  */
        }
    }
```

Điều quan trọng là: "siêu vòng lặp", hoặc 'vòng lặp vô cùng' là một điều cần thiết vì chúng ta không có hệ thống điều hành để quay trở về gia trị ban đầu, chương trình của chúng ta sẽ duy trì vòng lặp cho đến khi ngắt nguồn.

Ưu nhược điểm của siêu vòng lặp”

☺ Ưu điểm chính của vòng lặp chính là sự đơn giản. Điều này giúp chúng ta dễ dàng xây dựng, gỡ lỗi, kiểm tra và duy trì .

☺ Vòng lặp có hiệu quả cao: có tài nguyên phần cứng tối thiểu.

☺ Vòng lặp có tính di động cao .(rất phong phú và đa dạng)

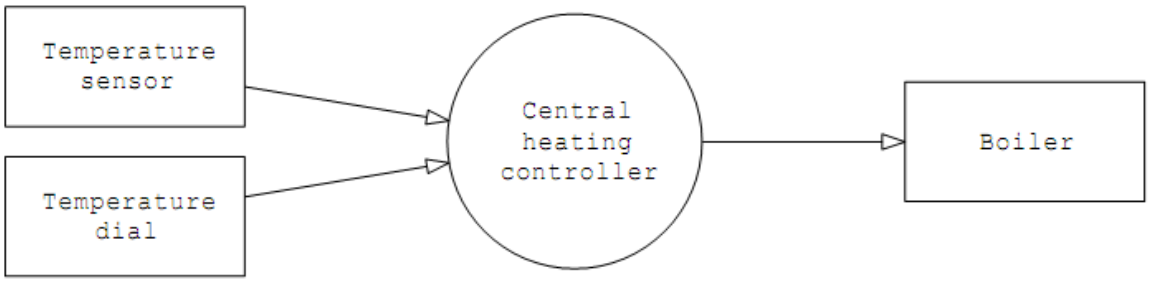
Nhược điểm:

- Nếu chương trình của bạn yêu cầu tính chính xác cao (ví dụ bạn cần có dữ liệu chính xác tới 2ms), thì khung vòng lặp này không cung cấp chính xác hay sự linh hoạt theo yêu cầu.
- Các vòng lặp hoạt động ở chế độ nguồn đầy đủ.(full power). Điều này có thể là không cần thết ở tất cả các chương trình, và có thể ảnh hưởng tới hệ thống nguồn tiêu thụ.

[Chúng ta sẽ thấy ở chương 6, người lập trình sẽ giải quyết vấn đề này.]

.

Ví dụ: Trung tâm điều khiển nhiệt độ




```
void main(void)
    {
    /* Khoi tao he thong  */
    C_HEAT_Init();

    while(1)  /* vong lap mai mai  */
        {
            /* tim kiem nhiet do nguoi dung cai dat thong qua dao dien nguoi        dung*/
        C_HEAT_Get_Required_Temperature();

        /* tim kiem nhiet do phong hien tai ..thong qua sensor nhiet do  */
        C_HEAT_Get_Actual_Temperature();

        /* dieu chinh bep lua theo yeu cau  */
        C_HEAT_Control_Boiler();
        }
    }
```

Đọc (và viết) các chân

Vấn đề

Làm thế nào để viết phần mềm để đọc và viết từ các chân của 8VĐK?

Bối cảnh:

8051 chuẩn có 4 cổng 8-bit.

Tất cả các chân đều dung được cả 2 chiều: input và out put.

SFRs các chân

Điều khiển các chân của 8051 thông qua phần mềm là được biết đến như là các thanh ghi đặc biệt "special function registers" (SFRs).

Về mặt vật lý, SFR là một vùng bộ nhớ trong RAM nội:

• P0 tại địa chỉ 0x80

• P1 tại địa chỉ 0x90

• P2 tại địa chỉ 0xA0

• P3 tại địa chỉ 0xB0

Chú ý: 0x là định dạng số ở dạng thập lục phân – xem cuốn Embedded C, Chapter 2.

SFRs và các chân

Một SFR phần đầu một tập tin cho họ 8051 sẽ so dạng:

```
sfr P0      =  0x80;
sfr P1      =  0x90;
sfr P2      =  0xA0;
sfr P3      =  0xB0;
```

Khai báo các biến SFR định danh, chúng ta có thể viết tới các port một cách đơn giản. Ví dụ, chúng ta có thể gửi dữ liệu tới Port 1 như sau:

```
unsigned char Port_data;

Port_data  =  0x0F;

P1  = Port_data;          /* viet  00001111 toi Port  1  */
```

Cũng như vậy, chúng ta có thể đọc từ Port 1 như sau:

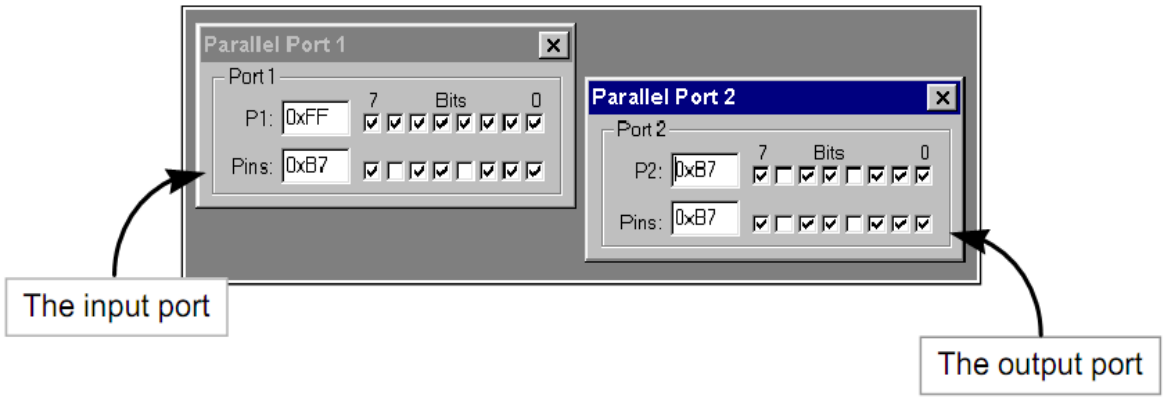
```
unsigned char Port_data;

P1  =  0xFF;              /* thiet lap che do doc  */
Port_data  = P1;          /* doc du lieu  */
```

Cách tạo và sử dụng biến sbit

Để viết tới 1 chân đơn của vđk (p1_1…) Chúng ta có thể dùng Keil C để dễ dàng biên dịch và kiểm soát lỗi.

```
#define LED_PORT P3

#define LED_ON  0
#define LED_OFF  1

sbit Warning_led  = LED_PORT^0;  /* LED duoc ket noi toi pin  3.0  */

Warning_led  = LED_ON;
...  /* delay  */
Warning_led  = LED_OFF;
...  /* delay  */
Warning_led  = LED_ON;
...  /* etc  */
```

Ví dụ: Đọc và viết bytes




```
void main  (void)
   {
   unsigned char Port1_value;

   /* cai dat P1 de doc  */
   P1  =  0xFF;

   while(1)
     {
     /* doc tu P1  */
     Port1_value  = P1;

     /* sao chep gia tri toi P2  */
     P2  = Port1_value;
     }
   }
```

Tạo hàm trễ bằng phần mềm

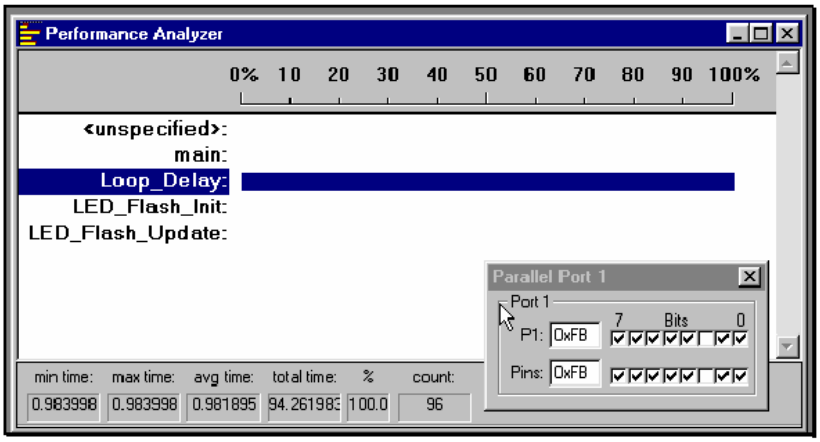
Vấn đề

Làm thế nào để tạo hàm trễ mà không dùng tới phần cứng (timer):

Giải quyết

```
Loop_Delay()
   {
   unsigned int x,y;

   for  (x=0; x  <=  65535; x++)
      {
      y++;
      }
   }


Longer_Loop_Delay()
   {
   unsigned int x, y, z;

   for  (x=0; x<=65535; x++)
      {
      for  (y=0; y<=65535; y++);
         {
         z++;
         }
      }
}
```

Sử dụng máy phân tích để kiểm tra hàm trễ



Ưu nhược điểm của hàm delays

💻 Tạo ra thời gian trễ ngắn.

☺ Không yêu cầu timer (của phần cứng)

☺ Làm việc với bất kì VĐK nào.

Nhược điểm:

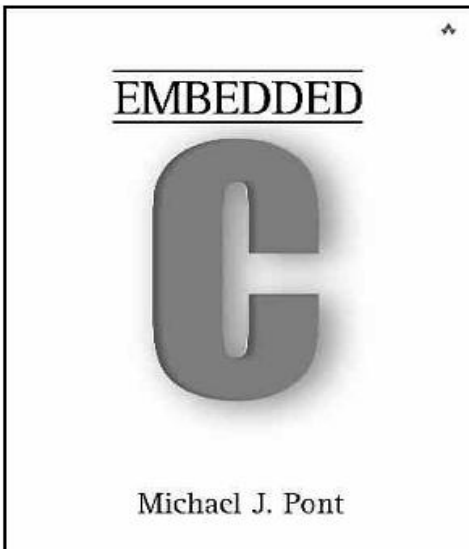
Rất khó để tạo thời gian trễ chính xác.

Vòng lặp phải được điều chỉnh lại nếu bạn sử dụng VĐK khác, thay đổi tần số, hoặc thậm chí thay đổi tối ưu hóa trình biên dịch.

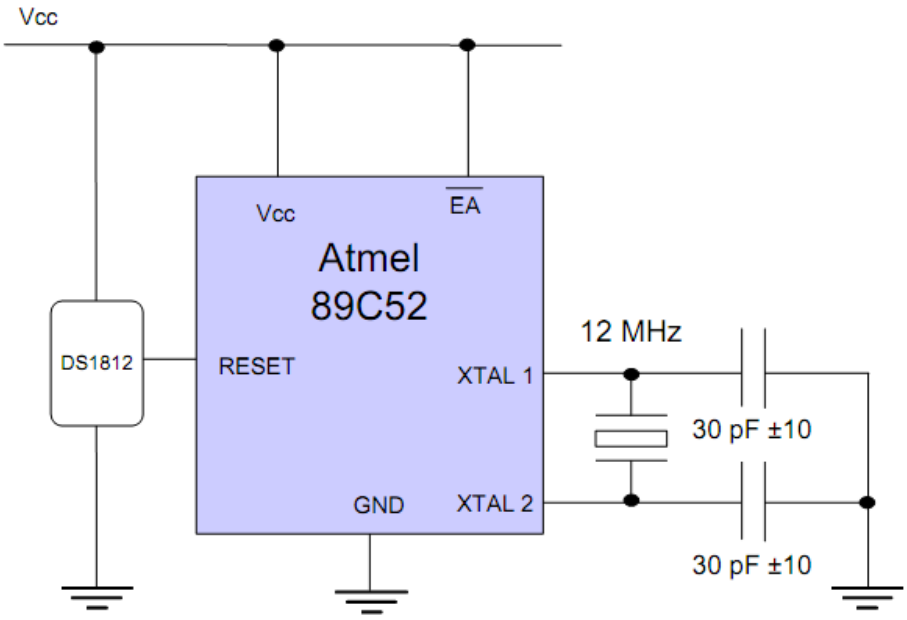
Chuẩn bị cho chương tiếp theo

Bạn giả lập phần cứng để thử những bài đã học được. Điều này tạo cho bạn có khả năng để tập trung vào khía cạnh phần mềm của hệ thống nhúng of embedded systems, mà không cần quan tâm đến vấn đề phần cứng.

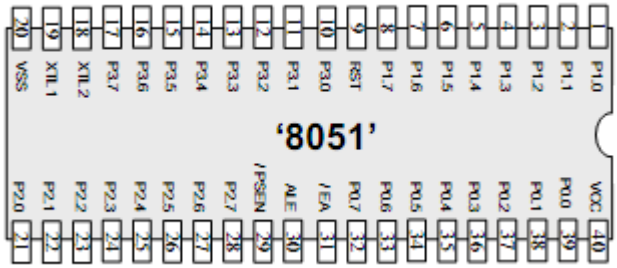
Trong chương tiếp theo chúng tôi sẽ chuẩn bị để tạo ra hệ thống thử nghiệm đầu tiên của bạn trên phần cứng thực tế.



Vui lòng đọc chương 1, 2, 3 trước khi đọc chương tiếp theo


Vcc
Vcc
EA
Atmel
89C52
DS1812
RESET
XTAL 1
GND
XTAL 2
12 MHz
30 pF ±10
30 pF ±10

Nhắc lại: The 8051 microcontroller



Tính năng điển hình của 8051:

• 32 cổng vào ra I/O.

• Bộ nhớ trong (RAM) - 256 bytes.

• Nâng lên 64 kbytes với bộ nhớ của ROM (usually flash)

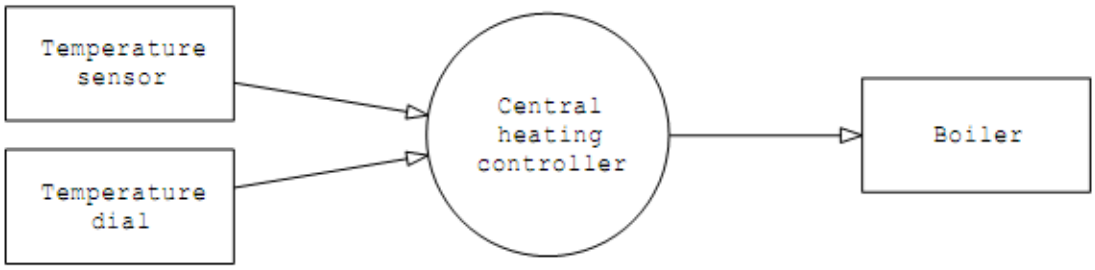
• 3 Timer 16bit/ counter

• 9 chế độ ngắt với 2 chế độ ngắt trong với 2 chế độ ưu tiên .

• Low-power Idle and Power-down modes.

Các loại khác của họ 8051 rất thích hợp cho tất cả mọi thứ từ hệ thống ôtô và hàng không đến điều khiển TV.

Nhắc lại: Trung tâm điều khiển lò sưởi




```
void main(void)
   {
   /* khoi tao he thong  */
   C_HEAT_Init();

   while(1)  /* vong lap mai mai  */
      {
      /* tim kiem nhiet do yeu cau thong qua giao
      dien nguoi dung */
      C_HEAT_Get_Required_Temperature();

      /* Nhiet do phong hien nay la….(thong qua gao
      dien nguoi dung */
      C_HEAT_Get_Actual_Temperature();

      /* dieu chinh bep ga theo yeu cau */
      C_HEAT_Control_Boiler();
      }
   }
```

Tổng quan :

Chương này sẽ:

• Giới thiệu kĩ thuật mà bạn cần để xây dựng một hệ thống nhúng thực tế đầu tiên của bạn (trên bo thử).

Chính xác, chúng ta sẽ xem xét:

• Mạch dao động

• Mạch Reset

• Mạch LEDs

Dao động phần cứng

• Tất cả máy tính kĩ thuật số đều được điều khiển bởi hình thức mạch dao động.

• Mạch này là nhịp tim của hệ thống và rất quan trọng để hệ thống làm việc chính xác.

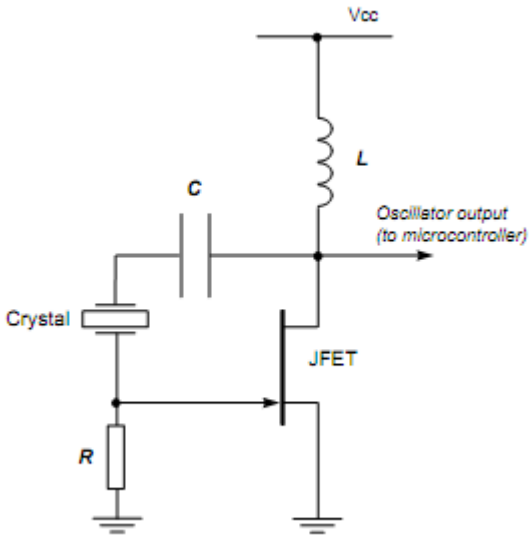
Ví dụ:

• Nếu dao động bị lỗi hệ thống sẽ ngừng hoạt động.

• Nếu dao động không đều thì bất kì thời gian tính toán nào cũng sẽ bị sai lệch.

Thạch anh dao động

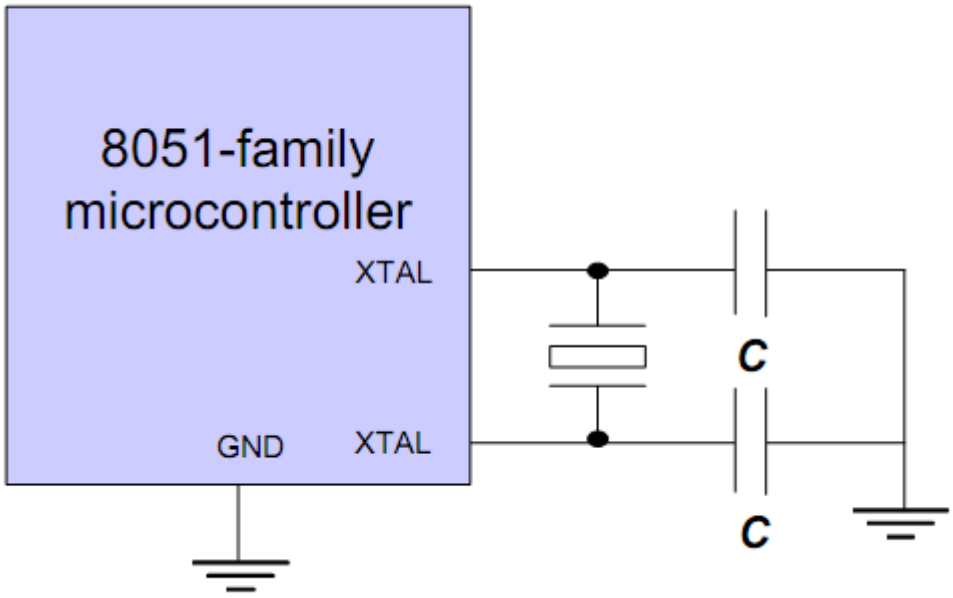
Thạch anh được sử dụng rất phổ biến để tạo ra các mạch dao động.



• Để tạo ra dao động hầu hết các thành phần bao gồm đều nằm trên chính VĐK

• Người sử dụng thiết bị này thường cung cấp thạch anh và hai tụ nhỏ để hoàn thành một bộ dao động.

Kết nối thạch anh tớ VDDK như thế nào?



Trong trường hợp không có thông tin cụ thể, giá trị của tụ =30pF sẽ thực hiện tốt trong mọi trường hợp.

Tần số dao động và thời gian chu kì máy

• Trong các thành viên họ 8051 ban đầu, 1 chu kì máy có khoảng 12 dao động.

• Các thế hệ sau như Infineon C515C, một chu kì máy có khoảng 6 dao động, trong các thiế bị gần đây như Dallas 89C420, chỉ 1 dao dộng trong mỗi chu kì máy.

• Kết quả là các thành viên thế hệ sau hoạt động cùng tần số nhưng thực hiện nhiều việc và nhanh hơn.

Giữ tần số càng thấp càng tốt

Many developers select an oscillator Nhiều nhà phát triển chọn một tần số cộng hưởng, đó là ở gần hoặc là giá trị cao nhất được hỗ trợ bởi thiết bị đặc biệt.

Điều này có thể là một sai lầm:

• Many Nhiều chương trình không yêu cầu cao về hiệu suất mà một loại 8051 hiện đại có thể cung cấp.

• Các nhiễu điện từ (EMI) sẽ tăng lên cùng với tần số dao động.

• Trong hầu hết các họ 8051 hiện đại (CMOS-based), có một mối quan hệ gần gũi giwuax tần số dao động và mạch nguồn cung cấp. Kết quả là, bằng cách sử dụng tần số thấp nhất cần thiết có thể sẽ giảm mức tiêu thụ điện năng: điều này rất hữu ích trong nhiều ứng dụng.

• Khi truy cập các thiết bị ngoại vi tần số thấp như LCD,bộ nhớ ngoài , chương trình và phần cứng thiết kế có thể đơn giản hóa- và chi phí cho các thiết bị ngoại vi, như bộ nhớ chốt có thể được giảm xuống – (nếu chip hoạt động chậm hơn).

Nói chung, bạn nên hoạt động tần số thấp nhất có thể tương thích với thực hiện các nhu cầu cho các ứng dụng của bạn.

Vấn đề ổn định

• Một yếu tố quan trọng trong việc lựa chọn một bộ dao động cho hệ thống dao động của bạn vấn đề ổn định dao động. Trong hầu hết các trường hợp,bộ dao dộng ổn định trong khoảng :'±20 ppm': '20 phần trên triệu.(percent part million).

Thạch anh tinh thể tiêu chuẩn được đánh giá trong khoảng từ ±10 → ±100 ppm, và như vậy có thể đạt tới 5 → 50 Phút/năm.

Cải thiện sự ổn định của tần số dao động

• Nếu bạn muốn một hệ thống điều khiển nhúng nói chung có thạch anh giữ thời gian chính xác, bạn có thể chọn cách là giữ thiết bị ở lò nướng hoặc tủ lạnh (nơi có nhiệt độ ổn định) , và tinh chỉnh phần mềm để giữ thời gian chính xác.(tuy nhiên điều này hiếm khi thực tế.

• 'Temperature Compensated Crystal Oscillators' (TCXOs) – Nhiệt độ bù dao động thạch anh- là có sẵn , cung cấp trong một gói phần mềm sử fujng dễ dàng- 1 thạch anh dao động, một mạch điện bù lại sự thay đổi nhiệt độ.

TCXOs có giá khoảng $100.00 .

• Một thực tế là để xác định nhiệt độ, tần số đặc trưng cho sự lựa chọn thạch anh của bạn, và bao gồm cả thông tin chương trình của bạn.

Giá của một cảm biến nhiệt độ nhỏ khoảng ~$2.00), Bạn có thể theo dõi nhiệt độ và điều chỉnh thời gian theo yêu cầu.

Ưu nhược điểm

☺ Thạc anh dao động là ổn định,. Thông thường khoảng ±20-100 ppm = ±50 phút/năm

- Phần lớn các thiết kế cơ sở của 8051 đều dựa trên mạch dao động thạch anh .
- Thạch anh có chi phí hợp lí cho việc phổ biến tần số.
- Các thành phần bổ sung chỉ yêu cầu 2 tụ nhỏ (30pF)

Nhưng:

Tinh thể dao động dễ bị rung, nên độ ổn định của tinh thể phụ thuộc vào độ tuổi.

Bộ cộng hưởng gốm

Khái quát ưu nhược điểm

☺ Rẻ hơn tinh thể.

☺ Mạnh về vật lý: ít bị tổn thương bởi rung động thiết bị,hoặc vỏ thiết bị.

☺ Nhiều cộn hưởng chứa được xây dựng trong tụ điện., và có thể sử dụng mà không cần bất kì thiết bị ngoại vi nào.

💻 Cỡ nhỏ, khoảng ½ tinh thể dao động.

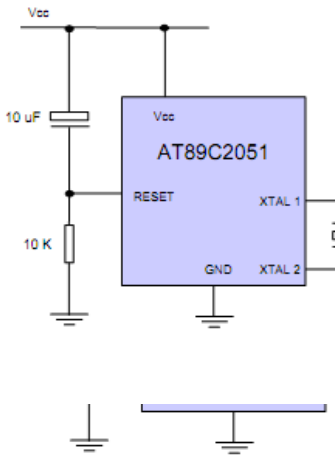
Nhưng

- Sự ổn định tương đối thấp: nói chung không thích hợp sử dụng khi yêu cầu chính xác thời gian.
- Độ chính xác thường khoảng ±5000 ppm = ±2500 phút/năm (lên tới 50 phút/tuần).

Reset phần cứng

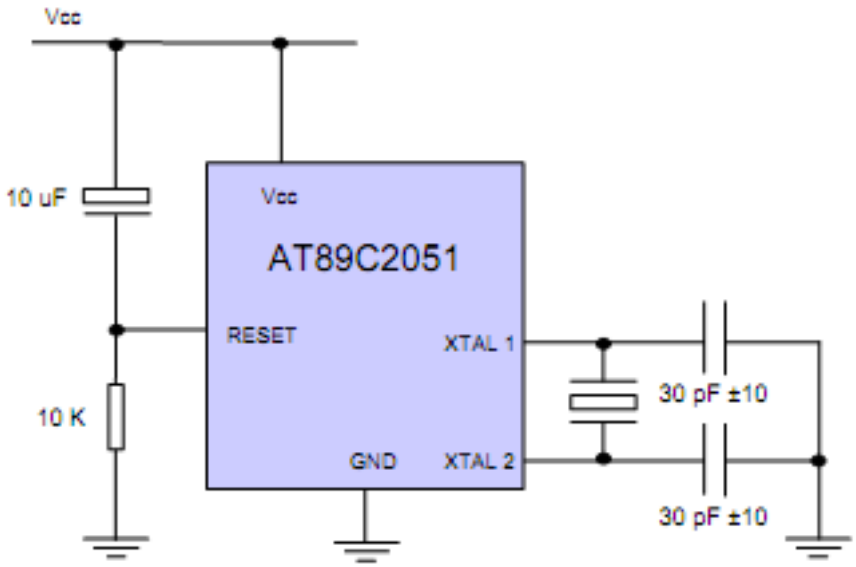
• Quá trình bắt đầu khởi động của VĐK là một quá trình không bình thường.

• Các phần cứng nằm bên dưới là phức tạp và nhỏ,
Nhà sản xuất đã xác định, (‘reset routine’)thói quen xác lập lại phải được chạy để đặt phần cứng này thành trạng thái thích hợp trước khi nó thực hiện một chương trình của người dùng. Việc reset mất một khoảng thời gian và yêu cầu bộ dao động của VĐK hoạt động.



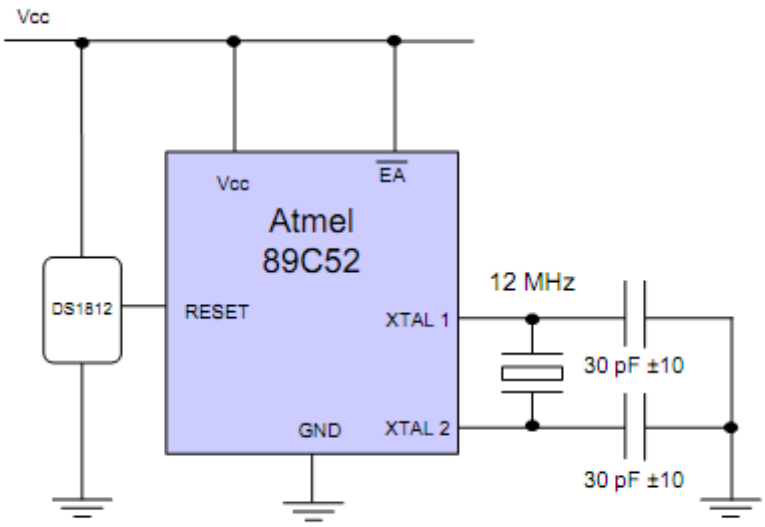
• 1 mạch reset là cách đơn giản nhất để diều khiên reset thường xuyên.

Ví dụ:



Mạch reset mạnh mẽ hơn

Ví dụ: (dùng ds1812)



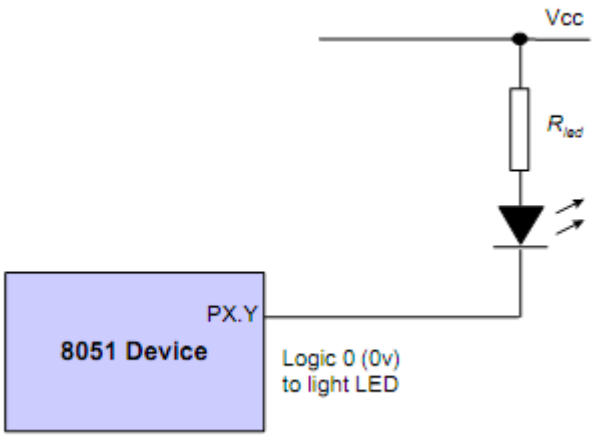
Tải DC (điện 1 chiều)

• Các chân cổng của VĐK 8051 có thể cài đặt ở giá trị 0V hoặc 5V (hoặc trong 3V, 0V hoặc 3V)dưới sự điều khiển của phần mềm.

• Mỗi cổng thông thường xuất ra tín hiệu có dòng ~10mA.

• Tổng dòng điện trên VĐK xuất ra khoảng 70mA hoặc nhỏ hơn.

LEDs



$$R_{led} = \frac{V_{cc} - V_{diode}}{I_{diode}}$$

Kết nối trực tiếp led đơn tới 1 chân VĐK .

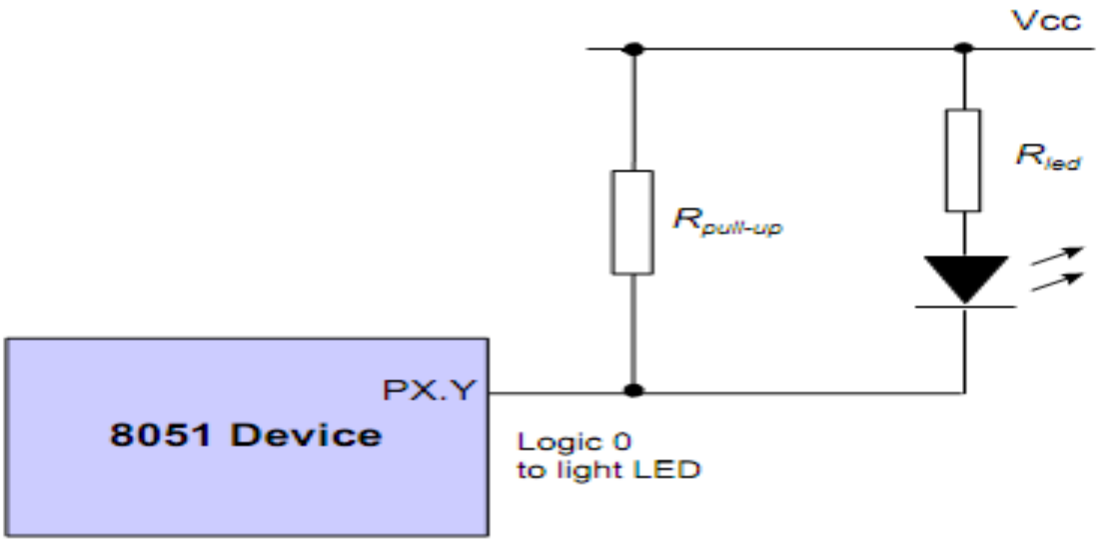
• Nguồn điện áp, V_{cc} = 5V,

• LED điện áp thuận, V_{diode} = 2V,

• Yêu cầu dòng điện, I_{diode} = 15 mA (Lưu ý rằng các thông tin về LED được chỉ rõ trong datasheet).

Giá trị điện trở cần thiết R= 200Ω.

Dùng điện trở kéo (Use of pull-up resistors)



Để thích nghi với mạch mà kết nối với chân vđk không có điện trở kéo ta cần phải mắc thêm điện trở kéo bên ngoài.

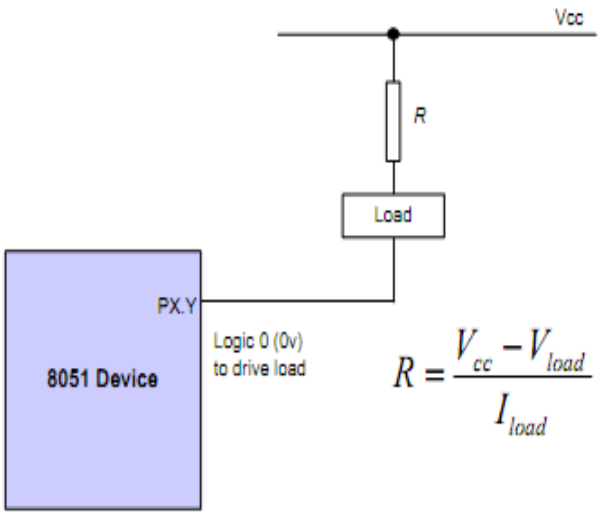
Giá trị điện trở kéo thích hợp vào khoảng 1K → 10K.

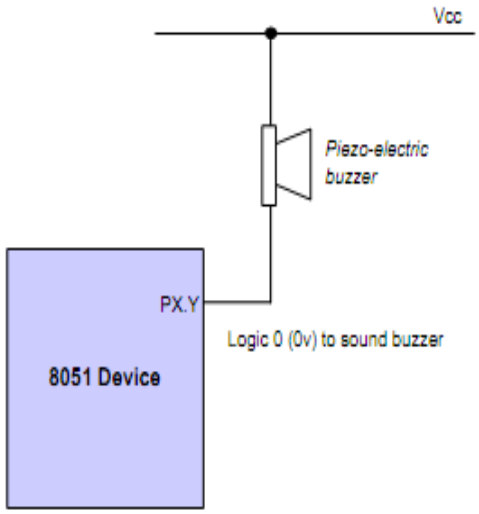
Điều này áp dụng cho mọi ví dụ trong sách.

Lưu ý:

Điều này thường xảy ra ở Port 0 (xem chương 3 để biết them chi tiết).

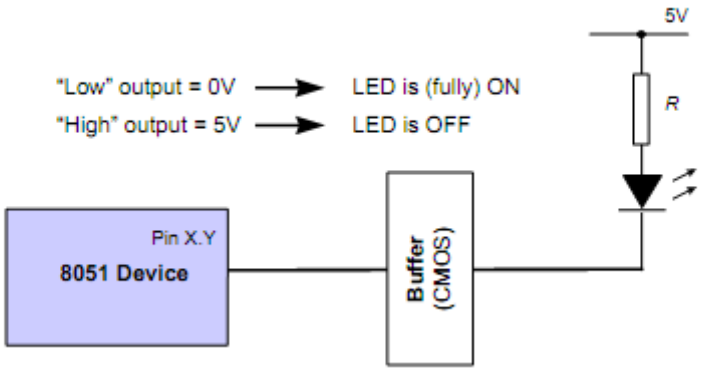
Lai một tải điện áp thấp mà không cần bộ đệm



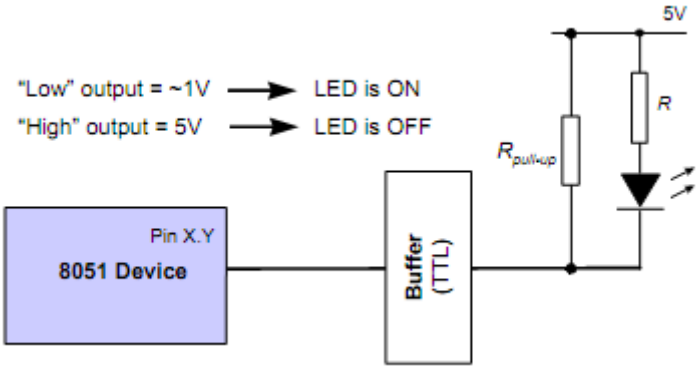


Xem cuốn " PATTERNS FOR TIME-TRIGGERED EMBEDDED SYSTEMS", p.115 (NAKED LOAD)

Dùng 1 IC đệm



Dùng IC đệm CMOS.



Dùng IC đệm TTL.

Có thể sử dụng IC đệm CMOS trong bất cứ mạch đệm nào của bạn. Tham khảo thêm tài liệu về mạch logic số.

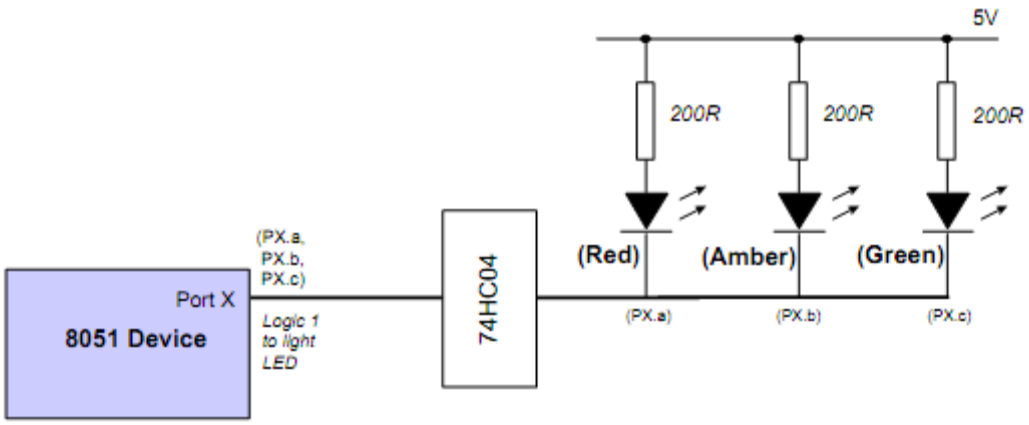
Xem cuốn"PATTERNS FOR TIME-TRIGGERED EMBEDDED SYSTEMS", p.118 (IC BUFFER)

Ví dụ: Đệm 3 LED với IC 74HC04

Ví dụ này chỉ dung một IC 74HC04 đệm cho 3 LEDs. Như đã thảo luận trong cách giải quyết đệm với CMOS chúng tôi không yêu cầu điện trở kéo lên với các bộ đệm HC (CMOS).

Trong trường hợp này chúng ta giả định rằng các led được điều khiển ở dòng ~15mA mỗi led trong tổng 50mA của bộ đệm.

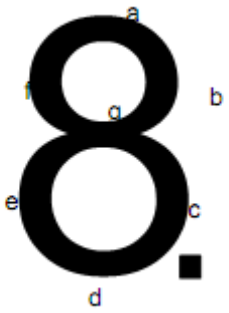
$$R_{led} = \frac{V_{cc} - V_{diode}}{I_{diode}} = \frac{5V - 2V}{0.015A} = 200\Omega$$



Xem cuốn "PATTERNS FOR TIME-TRIGGERED EMBEDDED SYSTEMS", p.123

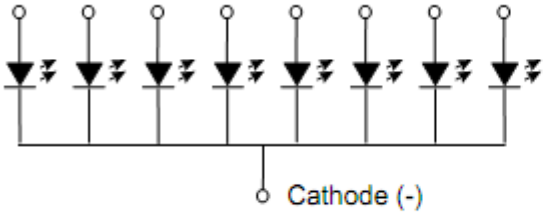
LED nhiều thanh là gì?

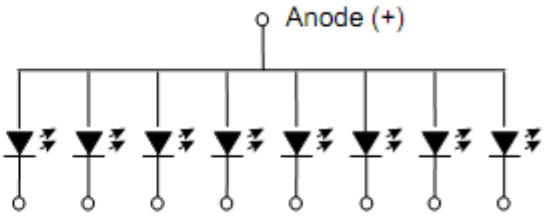
.



Nhiều đèn led được sắp xếp như một đoạn nhiều thanh hiển thị : sự hiển thị của 7 thanh led tạo thành led hiển thị 7 thanh

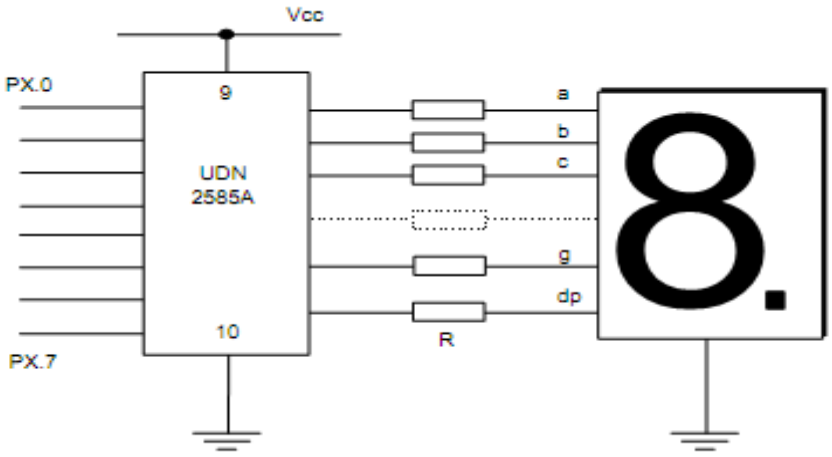
Hiển thị được bố trí như sau thường được gọi là “catot chung” hoặc “anot chung”.





Yêu cầu dòng điện qua mỗi led khoảng 2mA (led hiển thị nhỏ) tới 60mA (Led hiển thị to, 100mm hoặc hơn).

Điều khiển Led đơn

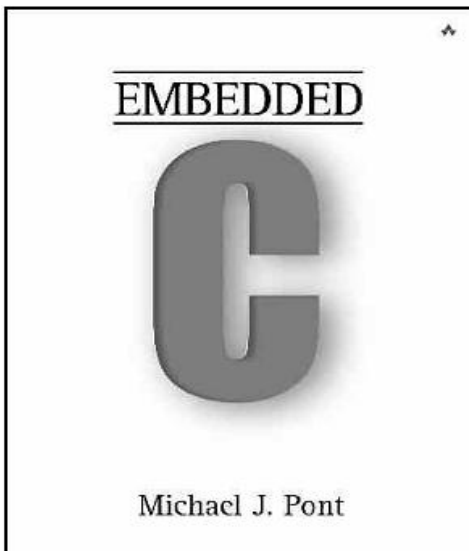


Trong trường hợp này yêu cầu có mạch đệm hoặc IC điều khiển giữa chân của VĐK và LED.

• Ví dụ, chúng ta có thể dùng UDN2585A.

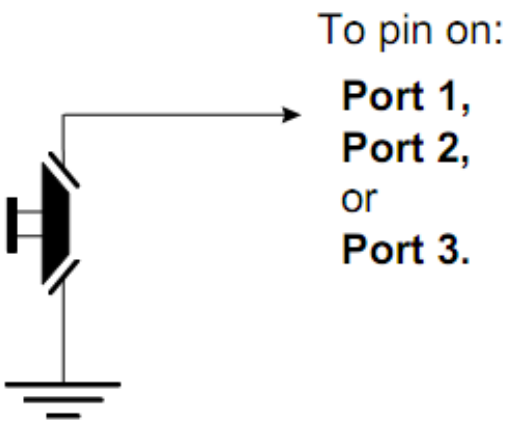
Mỗi kênh trong bộ đệm này có thể cùng một lúc dòng điện len tới 120 mA (lên tới 25V): Điều này là đủ cho một màn hình LED rất to.

• Lưu ý rằng đây là bộ đệm đảo ngược (nguồn điện). Logic 0 sẽ cho ra logic 1.



Chuẩn bị bài cho chương tiếp

Chương 3: Đọc nút bấm

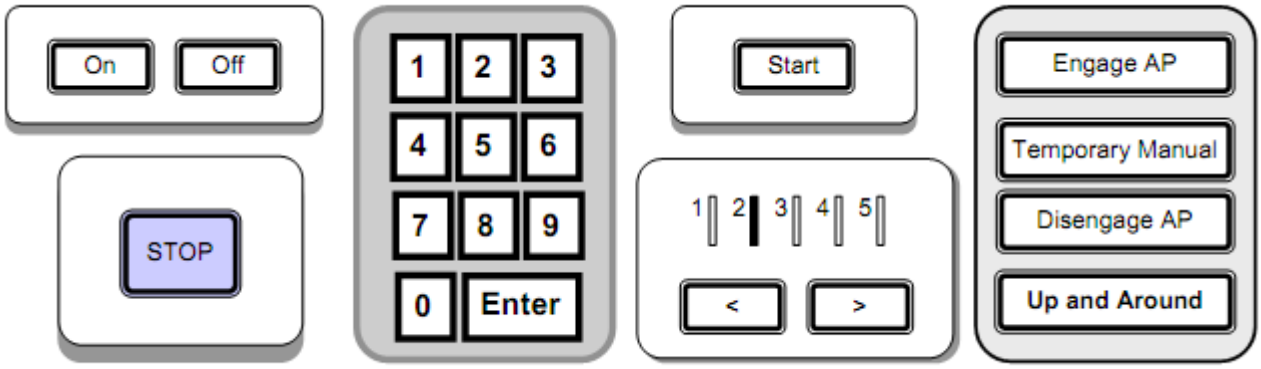


Giới thiệu

• Các hệ thống nhúng thường sử dụng các thiết bị chuyển mạch giống như một phần giao diện người dùng.

• Quy luật chung này áp dụng cho hầu hết các hệ thống điều khiển từ xa như gara ôtô, và nhiều nhất là hệ thống lái xe tự động.

• Dù hệ thống bạn tạo như thế nào, bạn vẫn cần có 1 giao diện chuyển đổi đáng tin cậy.



Trong chương này chúng ta xem xét làm thế nào để đọc tín hiệu từ nút bấm cơ khí tới hệ thống nhúng.

Trước khi xem xét chính các công tắc, chúng ta hãy xem xét quá trình đọc trạng thái từ chân vđk.

Nhắc lại: Phương pháp cơ bản đọc tín hiệu từ chân vđk.

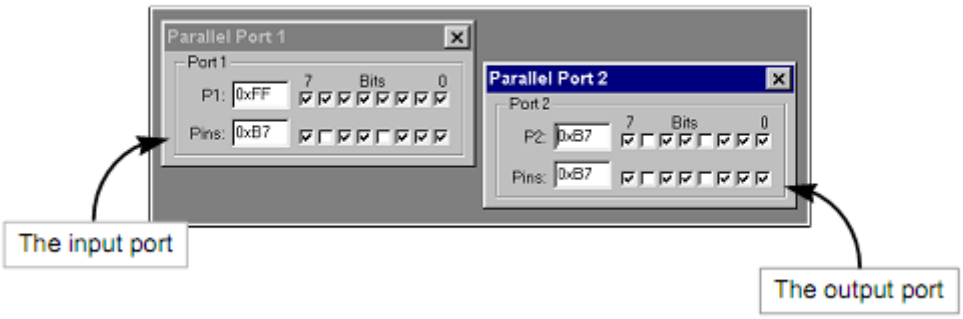
Chúng ta có thể gửi một vài dữ liệu vào Port 1 như sau:

```
sfr P1  =  0x90;          /* Tap tin tieu de thong thuong  */

P1  =  0x0F;              /* viet 00001111 toi Port  1  */
```

Trong 1 cùng cách chính xác, chúng ta có thể đọc từ Port 1 như sau :

```
unsigned char Port_data;

P1  =  0xFF;              /* Đặt Port 1 ở chế độ doc   */
Port_data  = P1;          /* doc tu P1  */
```

Ví dụ: Đọc và Viết từng bytes .




```
void main  (void)
    {
    unsigned char Port1_value;

    /* Phai thiet lap P1 de doc  */
    P1  =  0xFF;

    while(1)
       {
       /* doc gia tri P1  */
       Port1_value  = P1;

       /* chep gia tri toi  P2  */
       P2  = Port1_value;
       }
    }
```

Ví dụ: Đọc và viết bits (đơn giản)

```
/*---------------------------------------------------------*-

     Bits1.C  (v1.00)

-*---------------------------------------------------------*/

#include  <Reg52.H>

sbit Switch_pin  = P1^0;
sbit LED_pin  = P1^1;

/*  ... ..... */

void main  (void)
      {
      bit x;

      /* thiet lap chuyen doi pin de doc  */
      Switch_pin  =  1;

      while(1)
         {
         x  = Switch_pin;            /* doc Pin  1.0  */
         LED_pin  = x;               /* viet toi Pin  1.1  */
         }
      }

/*---------------------------------------------------------*-
   ---- END OF FILE  ----------------------------------
-*---------------------------------------------------------*/
```

Người lập trình có khinh nghiệm về "C" nên lưu ý dòng này:

```
sbit Switch_pin  = P1^0;
sbit LED_pin  = P1^1;
```

Ở đây chúng ta truy cập được vào 2 chân cổng P1 thông qua việc sử dụng 1 khai báo biến.

Ví dụ: Đọc và viết bits (phiên bản chung)

Sử dụng 6 phép toán với toán tử bit:

Phép toán	Mô tả
&	AND
|	OR
^	XOR
<<	Dịch bit sang trái
>>	Dịch bit sang phải
~	Đảo trạng thái

A	B	A **AND** B	A **OR** B	A **XOR** B
0	0	0	0	0
0	1	0	1	1
1	0	0	1	1
1	1	1	1	0

```
/* Chuong trinh may tinh  - minh hoa viec dung toan tu bit  */

#include  <stdio.h>

void Display_Byte(const unsigned char);

/* ... ..... */

int main()
      {
      unsigned char x  =  0xFE;
      unsigned int y  =  0x0A0B;

      printf("%-35s","x");
      Display_Byte(x);

      printf("%-35s","1s complement  [~x]");
      Display_Byte(~x);

      printf("%-35s","Bitwise AND  [x &  0x0f]");
      Display_Byte(x &  0x0f);

      printf("%-35s","Bitwise OR  [x  |  0x0f]"); Display_Byte(x
      |  0x0f);

      printf("%-35s","Bitwise XOR  [x  ^  0x0f]");
      Display_Byte(x  ^  0x0f);

      printf("%-35s","Left shift,  1 place  [x  <<=  1] "); Display_Byte(x  <<=
      1);

      x  =  0xfe;  /* quay tro lai x de * gia tri ban dau  */
      printf("%-35s","Right shift,  4 places  [x  >>=  4]"); Display_Byte(x  >>=
      4);

      printf("\n\n");

      printf("%-35s","Display MS byte of unsigned int y");
      Display_Byte((unsigned char)  (y  >>  8));

      printf("%-35s","Display LS byte of unsigned int y");
      Display_Byte((unsigned char)  (y &  0xFF));

      return  0;
      }
```

```
/* ---------------------------------------------------------- */

void Display_Byte(const unsigned char CH)
     {
     unsigned char i, c  = CH;
     unsigned char Mask  =  1  <<  7;

     for  (i  =  1; i  <=  8; i++)
          {
          putchar(c & Mask  ? '1'  : '0'); c  <<=  1;
          }

     putchar('\n');
     }
```

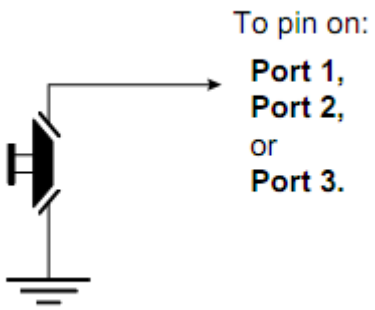
x	11111110
1s complement [~x]	00000001
Bitwise AND [x & 0x0f]	00001110
Bitwise OR [x | 0x0f]	11111111
Bitwise XOR [x ^ 0x0f]	11110001
Left shift, 1 place [x <<= 1]	11111100
Right shift, 4 places [x >>= 4]	00001111

Display MS byte of unsigned int y	00001010
Display LS byte of unsigned int y	00001011

```
    /*-------------------------------------------------------*-

    doc va viet du lieu vao tung chan(pin) cua cac cong.

    Luu y: 2 chan tren cung 1 port 1

-*-------------------------------------------------------*/

#include  <reg52.H>

void Write_Bit_P1(const unsigned char, const bit); bit
Read_Bit_P1(const unsigned char);

/*  ... ..... */

void main  (void)
    {
    bit x;

    while(1)
       {
       x  = Read_Bit_P1(0);          /* doc Port  1, Pin  0  */
       Write_Bit_P1(1,x);            /* viet to Port  1, Pin  1  */
       }
    }

/*  ------------------------------------------------------------  */

void Write_Bit_P1(const unsigned char PIN, const bit VALUE)
    {
    unsigned char p  =  0x01;             /*  00000001  */

    /* Dich trai voi so thich hop  */
    p  <<= PIN;

    /* neu muon muc logic 1 out o pin nay  */
    if (VALUE  ==  1)
        {
        P1  |= p;        /* toan tu OR  */
        return;
        }

    /* neu muon muc logic 0 out o pin nay  */
    p = ~p;         /* dao bit  */
    P1 &= p;  /* toan tu AND              */
}

/*  ------------------------------------------------------------  */

bit Read_Bit_P1(const unsigned char PIN)
    {
    unsigned char p  =  0x01;             /*  00000001  */
```

```
    /* Dich trai  */
    p  <<= PIN;

    /* viet 1 bit vao pin  (de cai dat che do doc)  */
    Write_Bit_P1(PIN,  1);

    /* doc pin  (AND) va tro ve gia tri   */
    return  (P1 & p);
    }

/*---------------------------------------------------------------*-
   ---- END OF FILE  ------------------------------------
-*---------------------------------------------------------------*/
```

Sự cần thiết của điện trở kéo



Phần cứng này hoạt động như sau:

• Khi nút bấm được mở, không có sự tác động vào các chân(pin) của các port. Một điện trở trong trên cổng kéo các chân của vđk lên mức 1 (thường là 5V). Nên chúng ta đọc các chân(pin) sẽ thấy giá trị là 1.

• Khi nút bấm được nhấn, các chân(pin) sẽ về mức 0V. khi chúng ta đọc các chân có giá trị 0.

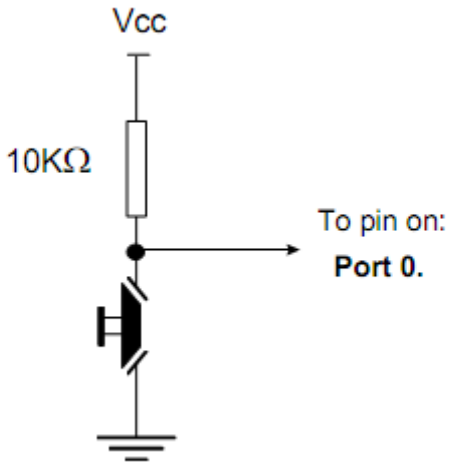
Sự cần thiết của điện trở kéo (P1,P2,P3)

Chúng ta đã tóm tắt điện trở kéo ở chương 2.

Có điện trở kéo:

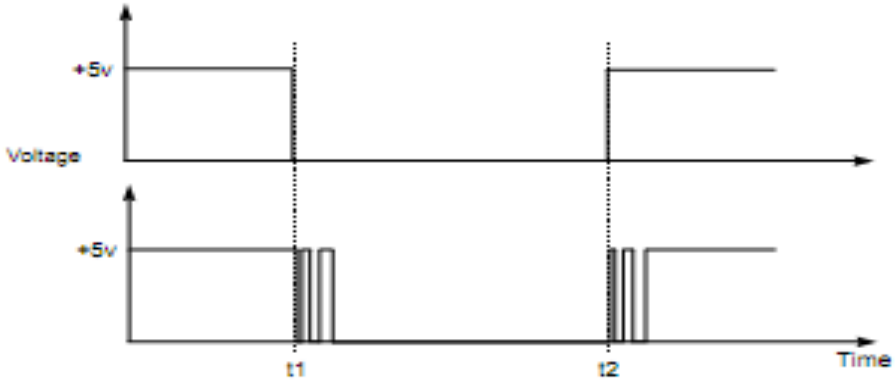
Không có điện trở kéo:

Sự cần thiết của điện trở kéo lên(P0)



Xử lý với nút bấm nảy lên

Trong thực tế các nút bấm cơ khí thường nảy lên (có nghĩa là ON rùi OFF trong một thời gian rất ngắn) sau khi nút bấm đóng hoặc mở. (đây gọi là rung cơ khí)



Mỗi lần nhảy tương đương với một lần bấm là đối với 1 nút bấm lý tưởng. Vì vậy nếu không thiết kế phần mềm phù hợp, điều nà có thể phát sinh một vài vấn đề, nhất là:

• Thay vì đọc ‘A’ từ bàn phím, chúng ta có thể đọc ‘AAAAA’.

• Đếm số lần công tắc được nhấn là vô cùng khó khăn.

• Nếu nút bấm được nhấn 1 lần và nhả ngay sau đó, việc nhảy phím diễn ra giống như ta lại nhấn nút bâm lần nữa.

Việc tạo ra phần mềm để kiểm tra giá trị đầu vào là rất đơn giản:

1. Chúng ta có thể đọc các chân phù hợp.

2. Nếu chúng ta nghĩ rằng nút bấm được nhấn, chúng ta chờ thêm 20ms và sau đó đọc lại chân lần nữa.

3. Nếu đọc lần 2 xác nhận đã đọc lầ 1 thì ta giả định rằng nút bấm đã được nhấn..

Lưu ý rằng con số ‘20 ms’ chỉ phù hợp cho một số trường hợp.

Ví dụ:Đọc nút bấm (code cơ bản)

Code dọc nút bấm này chủ yếu chúng ta muốn thực hiện các công việc như:

• Điều khiển motor khi nút bấm được nhấn.

• Chuyển đổi ánh sang đèn khi nút bấm được nhấn.

• Kích hoạt máy bơm khi nút bấm được nhấn.

Các hoạt động nay có thể sử dụng các công tắc điện mà không cần dung đến vđk, tuy nhiên chúng ta dùng vđk sẽ thích hợp cho điều khiển nhiều công tăc phức tạp . Ví dụ:

• Điều khiển motor khi nút bấm đcược nhấn.
  Điều kiện: Không phải bảo đảm an toàn tại chỗ, thay vào đó là tiếng rè trong 2s.

• Chuyển đổi ánh sang đèn khi nút bấm được nhấn.
  Điều kiện: tiết kiệm điện, br qua các yêu cầu để bật sáng ban ngày.

• Kích hoạt máy bơm khi nút bấm được nhấn
  Điều kiện: Nếu các bể chứa nước chủ yếu là dưới 300lit, Không bơm: thay vào đó bắt đầu dự trữ bơm và rút nước từ các bể khẩn cấp.

```
/*-------------------------------------------------------------*-

    Switch_read.C  (v1.00)


    Mot chuong trinh doc nut bam don gian cho  8051.
    - doc (phong nhay phim) nut bam vao Pin  1^0
    - neu nut bam duoc nhan thay thi doi Port  3 output

-*-------------------------------------------------------------*/

#include  <Reg52.h>

/* Connect switch to this pin  */
sbit Switch_pin  = P1^0;

/* hien thi trang thai nut bam o port 3  */
#define Output_port P3

/* tro ai gia tri Switch_Get_Input()  */
#define SWITCH_NOT_PRESSED  (bit)  0
#define SWITCH_PRESSED  (bit)  1

/* ham chuc nang  */
void SWITCH_Init(void);
bit    SWITCH_Get_Input(const unsigned char DEBOUNCE_PERIOD); void
DISPLAY_SWITCH_STATUS_Init(void);
void DISPLAY_SWITCH_STATUS_Update(const bit);
void DELAY_LOOP_Wait(const unsigned int DELAY_MS);
```

```
 /*------------------------------------------------------------  */
void main(void)
      {
      bit Sw_state;

      /* ham khoi tao  */
      SWITCH_Init();
      DISPLAY_SWITCH_STATUS_Init();

      while(1)
            {
            Sw_state  = SWITCH_Get_Input(30);

            DISPLAY_SWITCH_STATUS_Update(Sw_state);
            }
      }

/*-------------------------------------------------------------*-

   SWITCH_Init()

   Ham khoi tao cho swicth

-*-------------------------------------------------------------*/
void SWITCH_Init(void)
      {
      Switch_pin  =  1;  /* dung pin nay cho input  */
      }
```

```
/*---------------------------------------------------------*-

   SWITCH_Get_Input()

   Doc va phong chong nhay co khi nhu sau:

   1. Neu nut bam khong nhan, tro ve SWITCH_NOT_PRESSED.

   2. neu nut bam duoc nhan, cho ham DEBOUNCE_PERIOD  (in ms).
        Sau do:
        a. Neu nut bam khong duoc nhan trong thoi gian dai, tro ve SWITCH_NOT_PRESSED.
        b. neu nut bam van duoc nhan, tro ve  SWITCH_PRESSED

   xem Switch_Wait.H de biet thm chi tiet ve gia tri ham tre.

-*---------------------------------------------------------*/
bit SWITCH_Get_Input(const unsigned char DEBOUNCE_PERIOD)
     {
     bit Return_value  = SWITCH_NOT_PRESSED;

     if  (Switch_pin  ==  0)
          {
          /* nut bam duoc nhan  */

          /* phong chong nhay phim – cho phim ...  */
          DELAY_LOOP_Wait(DEBOUNCE_PERIOD);

          /* kiem tra nut nhan lan nua  */
          if  (Switch_pin  ==  0)
               {
               Return_value  = SWITCH_PRESSED;
               }
          }

     /* ay gio qay tro ve gia tri nut nhan  */
     return Return_value;
     }
```

```
/*---------------------------------------------------------*-

   DISPLAY_SWITCH_STATUS_Init()

   Ham khoi tao  DISPLAY_SWITCH_STATUS.( Trang thai cua nut bam)

-*---------------------------------------------------------*/
void DISPLAY_SWITCH_STATUS_Init(void)
     {
     Output_port  =  0xF0;
     }

/*---------------------------------------------------------*-

   DISPLAY_SWITCH_STATUS_Update()

   Ham don gia hien thi du lieu  (SWITCH_STATUS) tren led ket
   noi voi cac pin  (Output_Port)

-*---------------------------------------------------------*/
void DISPLAY_SWITCH_STATUS_Update(const bit SWITCH_STATUS)
     {
     if  (SWITCH_STATUS  == SWITCH_PRESSED)
          {
          Output_port  =  0x0F;
          }
     else
          {
          Output_port  =  0xF0;
          }
     }
```

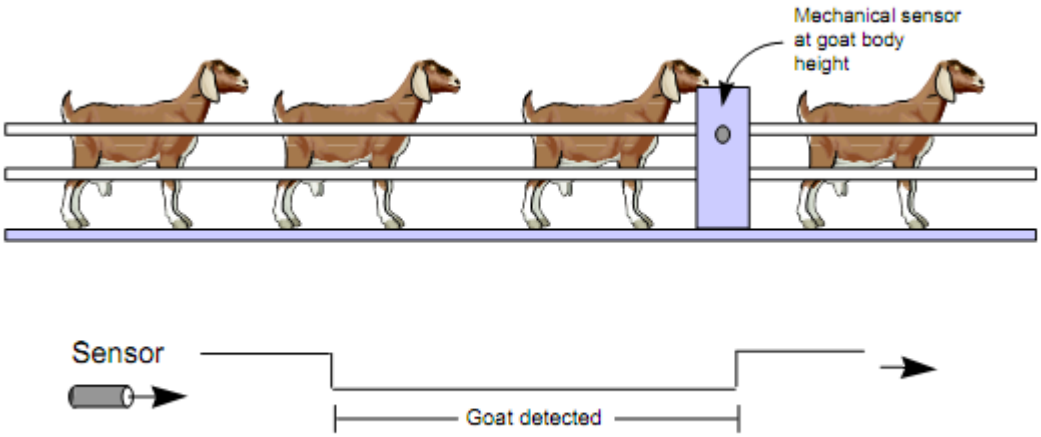
```
/*---------------------------------------------------------*-

   DELAY_LOOP_Wait()

   Thoi giant re la khac nhau voi cac tham so.

   Tham so la,  khoang thoi giant re trong 1ms, tren  12MHz  8051  (12 dao
   dong trong 1 chu ki may).

   Ban can phai dieu chinh choc ac ung dung cua ban!

-*---------------------------------------------------------*/
void DELAY_LOOP_Wait(const unsigned int DELAY_MS)
      {
      unsigned int x, y;

      for  (x  =  0; x  <= DELAY_MS; x++)
            {
            for  (y  =  0; y  <=  120; y++);
            }
      }
```

Ví dụ: Đếm những chú cừu

• Với code đơn giản trong ví dụ trước, vấn đề có thể xảy ra bất cứ khi nào một nút bấm(công tắc) được nhấn trong khoảng thời gian dài hơn thời gian trễ.

• Đây là mối quan tâm lớn, bởi vì trong nhiều trường hợp, người sử dụng sẽ nhấn trong it nhất 500ms (hoặc cho đến khi họ nhận được phản hồi rằng nút bấm đã được nhấn). Kết quả là người dùng gõ chữ ”hello” trên man hình có thể thấy:
“HHHHHHHHHeeeeeeeeellllllllllllllloooooooooo”.

Một hậu quả là các mã code này không thích hợp cho ứng dụng mà chúng ta cần đếm số lần công tắc được nhấn .

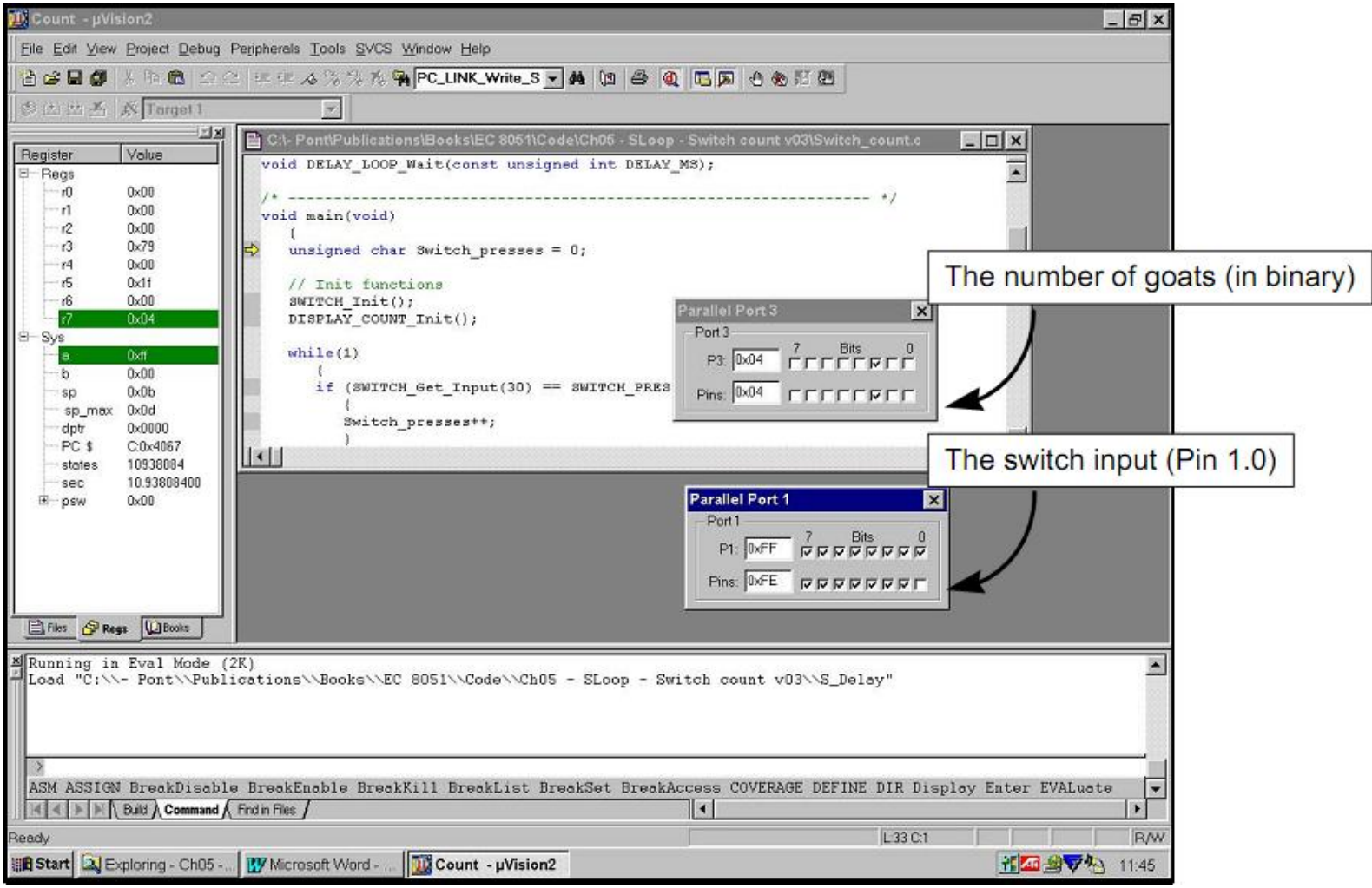


Nếu chúng ta thử dung code này cho ví dụ trước, cảm biến sẽ không cho chúng ta đếm số cừu nhưng thay vào đó là thời gian chú cừu đi qua sensor.

```
/*-------------------------------------------------------*-

    mot chuong trinh "dem cuu voi"  8051...

-*-------------------------------------------------------*/

#include  <Reg52.h>

/* ket noi toi pin nay  */
sbit Switch_pin  = P1^0;

/* hien thi so dem o port nay  */
#define Count_port P3

/* tro ve gia tri Switch_Get_Input()  */
#define SWITCH_NOT_PRESSED  (bit)  0
#define SWITCH_PRESSED  (bit)  1

/* ham chuc nang  */
void SWITCH_Init(void);
bit    SWITCH_Get_Input(const unsigned char DEBOUNCE_PERIOD); void
DISPLAY_COUNT_Init(void);
void DISPLAY_COUNT_Update(const unsigned char);
void DELAY_LOOP_Wait(const unsigned int DELAY_MS);

/*  ------------------------------------------------------------  */
void main(void)
    {
    unsigned char Switch_presses  =  0;

    /* Init functions  */
    SWITCH_Init();
    DISPLAY_COUNT_Init();

    while(1)
        {
        if  (SWITCH_Get_Input(30)  == SWITCH_PRESSED)
            {
            Switch_presses++;
            }

        DISPLAY_COUNT_Update(Switch_presses);
        }
    }
```

```
/*---------------------------------------------------------*/

void SWITCH_Init(void)
      {
      Switch_pin  =  1;  /* dung pin nay cho input  */
      }

/*---------------------------------------------------------*-

    SWITCH_Get_Input()

    Doc va chong nhay co khi nhu sau:

    1. Neu nut bam khong nhan, tro ve SWITCH_NOT_PRESSED.

    2. neu nut bam duoc nhan, cho ham DEBOUNCE_PERIOD  (in ms).
          Sau do:
          a. Neu nut bam khong duoc nhan trong thoi gian dai, tro ve SWITCH_NOT_PRESSED.
          b. neu nut bam van duoc nhan, tro ve  SWITCH_PRESSED

    xem Switch_Wait.H de biet thm chi tiet ve gia tri ham tre.

-*---------------------------------------------------------*/
bit SWITCH_Get_Input(const unsigned char DEBOUNCE_PERIOD)
      {
      bit Return_value  = SWITCH_NOT_PRESSED;

      if  (Switch_pin  ==  0)
            {
            /* nut bam duoc nhan  */

            /* chong nhay phim  - just wait...  */
            DELAY_LOOP_Wait(DEBOUNCE_PERIOD);

            /* kiem tra lai nut bam  */
            if  (Switch_pin  ==  0)
                  {
                  /* doi den khi nut bam duoc nha ra.  */
                  while  (Switch_pin  ==  0);
                  Return_value  = SWITCH_PRESSED;
                  }
            }

      /* cuoi cung quay tro ve gia tri ban dau  */
      return Return_value;
      }
```

```
/*---------------------------------------------------------*-

   DISPLAY_COUNT_Init()

   Ham khoi tao hien thi gia tri dem DISPLAY COUNT .

-*---------------------------------------------------------*/
void DISPLAY_COUNT_Init(void)
      {
      Count_port  =  0x00;
      }

/*---------------------------------------------------------*-

   DISPLAY_COUNT_Update()

   Ham don gian de hien thi ra cac led  (Count_Port)

-*---------------------------------------------------------*/
void DISPLAY_COUNT_Update(const unsigned char COUNT)
      {
      Count_port  = COUNT;
      }

/*---------------------------------------------------------*-

   DELAY_LOOP_Wait()

   Thoi giant re la khac nhau voi cac tham so.

   Tham so la,  khoang thoi giant re trong 1ms, tren  12MHz  8051  (12 dao
   dong trong 1 chu ki may).

   Ban can phai dieu chinh cho cac ung dung cua ban!

-*---------------------------------------------------------*/
void DELAY_LOOP_Wait(const unsigned int DELAY_MS)
      {
      unsigned int x, y;

      for  (x  =  0; x  <= DELAY_MS; x++)
            {
            for  (y  =  0; y  <=  120; y++);
            }
      }
```



Kết luận

Mã code nút bấm vừa trình bày và thảo luận trong các chương trước giúp chúng ta thực hiện được 2 việc :

• Thực hiện một hoạt động khi nút bấm được nhấn.

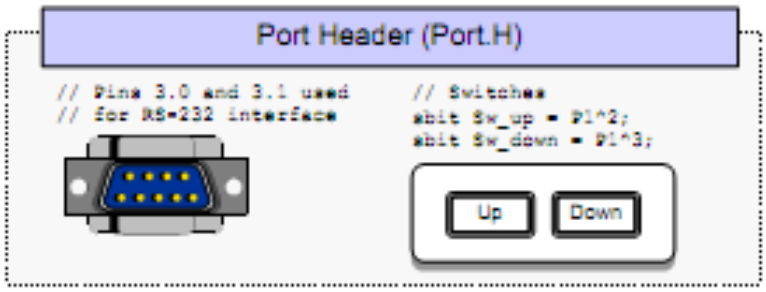
• Trả lời sự việc là người sử dụng nhấn và sau đó nút bấm và sau đó nhả.

Trong cả 2 trường hợp chúng tôi đã minh họa làm thế nào để chống rung(nhảy) nút bấm.

Chuẩn bị cho chương tiếp theo

Trong chương tiếp theo, chung tôi chuyển sự chú về phương pháp để các bạn có thể sử dụng lại mã code cho các đề tài tiếp theo.

Chương 4: Thêm cấu trúc vào code của bạn



Giới thiệu

Chúng ta sẽ làm 3 việc chính trong chương này:

1. Chúng ta mô tả làm cách nào để áp dụng lập trình hướng đối tượng với ngôn ngữ C.cho phép tạo ra các thư viện mã code để bạn dễ dàng lắp vào các đề tài lập trình nhúng khác.

2. Chúng ta mô tả làm cách nào để tạo và sử dụng các file "header". Các tập tin nay gói gọn trong khía cạnh từ khóa của môi trường phần cứng, giống như các loại vi xử lý được sử dụng, bộ dao động tần số và số chu kì yêu cầu thực hiện mỗi lệnh. Điều này giúp ta dễ dàng tới cổng có mã của vi xử lý khác.

3. Chúng ta sẽ mô tả làm cách nào để tạo và sử dụng một file 'Port Header': tập hợp tất cả chi tiết của các cổng truy cập từ toàn bộ hệ thống. Giống như file "header", nó có rất nhiều tính năng quan trọng.

Chúng ta sẽ sử dụng 3 phương pháp này trong code ví dụ ở các chương sau.

Ngôn ngữ chung	Ngôn ngữ ví dụ
-	Code máy
First-Generation Language (1GL)	Assembly Language.
Second-Generation Languages (2GLs)	COBOL, FORTRAN
Third-Generation Languages (3GLs)	C, Pascal, Ada 83
Fourth-Generation Languages (4GLs)	C++, Java, Ada 95

Graham đã nói[1]:

> Cụm từ "hướng đối tượng"('object-oriented)' đã trở thành gần như đồng nghĩa với sự hiện đại, tốt đẹp và giá trị trong công nghệ thông tin.

Jalote đã nói:

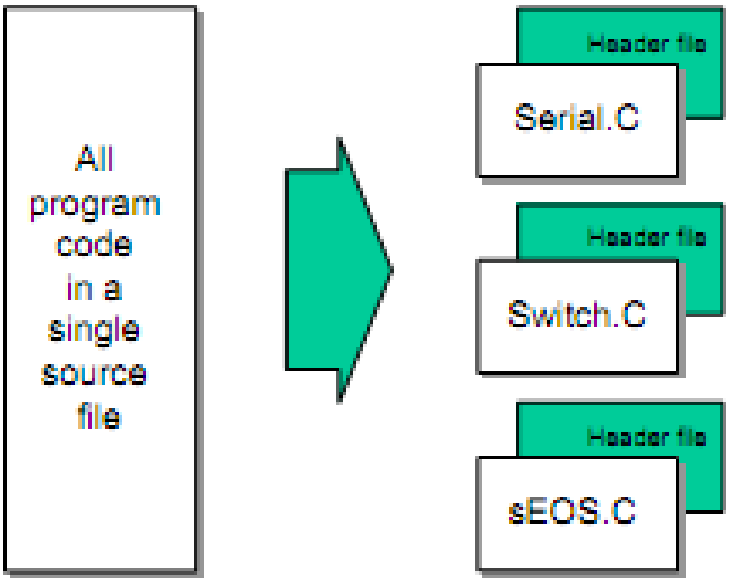
> "Một lợi thế chính của việc sử dụng định hướng hướng đối tượng là nó làm cho dễ dàng sản xuất và hiểu thiết kế hơn.

Ngôn ngữ hướng đối tượng không có sẵn trong các ứng dụng nhỏ, bởi vì chi phí cho việc sử dụng các tính năng trong ngôn ngữ này khá cao.

1 Graham, I. (1994) "Object-Oriented Methods," (2nd Ed.) Addison-Wesley. Page 1.

2 Jalote, P. (1997) "An Integrated Approach to Software Engineering", (2nd Ed.) Springer-Verlag. Page 273.

Nó có thể tạo ra các ‘file-based-classes’(file cơ sở, các lớp) trong C mà không cần một bộ nhớ đáng kể..



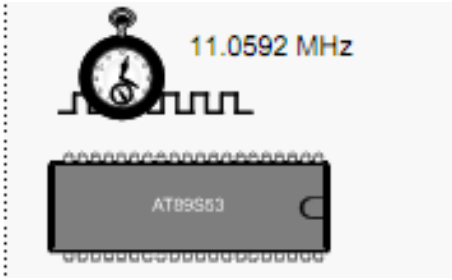
Ví dụ lập trình hướng đối tượng với c (O-O C)

```
/*-------------------------------------------------------------*-

     PC_IO.H  (v1.00)


     Xem  PC_IO.C de biet them chi tiet

-*-------------------------------------------------------------*/

#ifndef  _PC_IO_H
#define  _PC_IO_H

/*  ------ hang so cong khai  ----------------------------------  */

/* Gia tri tro ve boi  PC_LINK_Get_Char_From_Buffer neu khong co san trong bo
dem  */
#define PC_LINK_IO_NO_CHAR  127

/*  ------ Khai bao ham mau  -------------------------  */

void PC_LINK_IO_Write_String_To_Buffer(const char* const); void
PC_LINK_IO_Write_Char_To_Buffer(const char);

char PC_LINK_IO_Get_Char_From_Buffer(void);

/* phai thuong xuyen goi cac ham nay...  */
void PC_LINK_IO_Update(void);

#endif

/*-------------------------------------------------------------*-
    ---- END OF FILE  ------------------------------------
-*-------------------------------------------------------------*/
```

```
/*-------------------------------------------------------------*-

     PC_IO.C  (v1.00)


     [chua day du  - - xem  EC Chap  9 de hoan thien thu vien] -*-----------------------------------

--------------------*/

#include "Main.H"
#include "PC_IO.H"

/*  ------ dinh nghia cac bien cong khai ------------------------  */

tByte In_read_index_G;              /* du lieu trong bo dem da duoc “doc”  */
tByte In_waiting_index_G;           /* du lieu trong bo dem cho de doc  */

tByte Out_written_index_G;  /* du lieu trong bo dem da duoc viet  */ tByte
Out_waiting_index_G;  /* du lieu trong bo dem cho de viet */

/*  ------ ham mau rieng  ------------------------  */

static void PC_LINK_IO_Send_Char(const char);

/*  ------ hang so rieng  ---------------------------------  */

/* nhan  bo dem length */
#define RECV_BUFFER_LENGTH  8

/* truyen bo dem length  */
#define TRAN_BUFFER_LENGTH  50

#define XON      0x11
#define XOFF  0x13

/*  ------ cac bien rieng  ---------------------------------  */

static tByte Recv_buffer[RECV_BUFFER_LENGTH];
static tByte Tran_buffer[TRAN_BUFFER_LENGTH];

/*-------------------------------------------------------------*/
void PC_LINK_IO_Update(...)
     {

     }
```

```
/*-------------------------------------------------------*/
void PC_LINK_IO_Write_Char_To_Buffer(...)
      {

      }

/*-------------------------------------------------------*/
void PC_LINK_IO_Write_String_To_Buffer(...)
      {

      }

/*-------------------------------------------------------*/
char PC_LINK_IO_Get_Char_From_Buffer(...)
      {

      }

/*-------------------------------------------------------*/
void PC_LINK_IO_Send_Char(...)
      {

      }
```

Project Header (Main.H)




```
#include  <AT89S53.H>

#define OSC_FREQ  (11059200UL)

typedef unsigned char tByte;
```

```
/*---------------------------------------------------------*-

     Main.H  (v1.00)

-*---------------------------------------------------------*/

#ifndef  _MAIN_H
#define  _MAIN_H

/*-------------------------------------------------
     Chung ta can chinh sua doan nay cho moi project
   ------------------------------------------------  */

/* bao gom cac tap tin cua loai vi dieu khien thich hop  */
#include  <reg52.h>

/* tan so dao dong  (in Hz) e.g.  (11059200UL)  */
#define OSC_FREQ  (12000000UL)

/* so dao dong theo huong  dan  (12, etc)
      12  - ho  8051  /  8052 va phan lon  nhieu thiet bi khac nhau cua Philip.
      4  - Dallas  320,  520 etc.
        1  - Dallas  420, etc.  */
#define OSC_PER_INST  (12)

/*  -------------------------------------------------
     khong nen them cac phan duoi
   ------------------------------------------------  */

/* dinh ghia theo loai bien (xem Chap  5)        */
typedef unsigned char tByte;
typedef unsigned int    tWord;
typedef unsigned long tLong;

/* ham ngat  (xem Chap  7)                    */
#define INTERRUPT_Timer_0_Overflow  1
#define INTERRUPT_Timer_1_Overflow  3
#define INTERRUPT_Timer_2_Overflow  5

#endif

/*---------------------------------------------------------*-
   ---- END OF FILE  ---------------------------------
-*---------------------------------------------------------*/
```

Các tiêu đề thiết bị

```
/*-------------------------------------------------------
   REG515C.H

   File tieu de  Infineon C515C

   Copyright  (c)  1995-1999 Keil Elektronik GmbH    All rights reserved.
-------------------------------------------------------*/


/*      bo chuyen doi A/D           */
sfr     ADCON0  =  0xD8;


/*      he thong ngat               */
sfr     IEN0            =  0xA8;


/*      Ports       */
sfr     P0              =  0x80;
sfr     P1              =  0x90;
sfr     P2              =  0xA0;
sfr     P3              =  0xB0;
sfr     P4              =  0xE8;
sfr     P5              =  0xF8;
sfr     P6              =  0xDB;
sfr     P7              =  0xFA;


/*      Serial Channel              */
sfr     SCON            =  0x98;


/*      Timer0  / Timer1            */
sfr     TCON            =  0x88;


/*      CAP/COM Unit  / Timer2  */
sfr     CCEN            =  0xC1;
```

Tần số dao động và dao động theo hướng dẫn

```
/* tan so dao dong  (in Hz) e.g.  (11059200UL)  */
#define OSC_FREQ  (12000000UL)

/* so dao dong theo huong dan  (12, etc)
      12  - ho  8051  /  8052 va phan lon cac thiet bi khac cua philip.
      4  - Dallas  320,  520 etc.
        1  - Dallas  420, etc.  */
#define OSC_PER_INST  (12)
```

Chúng ta giải thích làm thế nào đê sử dụng các thông tin này:

• Tạo trễ (Embedded C, Chapter 6),

• Kiểm soát thời gian trong hệt thống (Chapter 7), and,

• Kiểm soát tốc độ (baud) truyền trong giao tiếp kiểu serial (Chapter 9).

Dữ liệu các loại biến thông thường

```
typedef unsigned char tByte;
typedef unsigned int    tWord;
typedef unsigned long tLong;
```

Trong C các từ khóa "typedef" cho phép chúng ta cung cấp bí danh cho các kiểu dữ liệu. Như vậy, trong project này chúng ta nhìn thấy mã như thế này:

tWord Temperature;

thay vì:

unsigned int Temperature;

Lí do chính sử dụng typedef là để đơn giản hóa và thúc đẩy việc sử dụng dữ liệu không âm.(unsigned)

• 8051 không hỗ trợ mã số và mã code số học phụ là cần thiết để thao tác dữ liệu : điều này làm giảm tốc độ chương trình và làm tăng kích thước chương trình.

• Sử dụng toán tử bit chỉ dung cho các dữ liệu kiểu unsigned: dung biến kiểu 'typedef' làm giảm khả năng lập trình và vô tình áp dụng toán tử này vào dữ liệu signed.

Cuối cùng, giống như chương trình máy tính, sử dụng từ khóa " *typedef*" là cách để dễ dàng để lắp ráp tới code của bạn sử dụng các vi xử lí khác.

Interrupts

Như đã lưu ý trong "Embedded C" Chapter 2, Ngắt là một thành phần quan trọng trong các hệ thông nhúng.

Những dòng sau trong tiêu đề "Project Header" có tác dụng làm cho bạn dễ dàng sử dụng hàm ngắt trong các dự àn của bạn.

```
#define INTERRUPT_Timer_0_Overflow  1
#define INTERRUPT_Timer_1_Overflow  3
#define INTERRUPT_Timer_2_Overflow  5
```

Chúng ta sẽ mô tả làm cách nào để dung hàm này dễ dàng trong " Embedded C, Ch. 7."

Tóm tắt: tại sao sử dụng Project Header(tiêu đề dự án)?

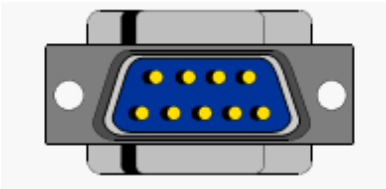
Việc sử dụng PROJECT HEADER làm cho code của bạn trở nên hay hơn, bởi vì bất cứ ai sử dụng project của bạn đều biết nơi để tìm kiếm thông tin. Chẳng hạn như mô hình mẫu của VĐK và tần số dao động đòi hỏi sự thực hiện của phần mềm.

Việc sử dụng 1 project header có thể làm cho code của bạn dễ dàng di động, bằng cách đặt một số dữ liệu quan trọng phụ thuộc vào VĐK ở một nơi: nếu bạn thay đổi bộ vi xử lý hoặc dao động sử dụng sau đó- trong nhiều trường hợp – bạn sẽ chỉ cần thay đổi project header(tiêu đề dự án)

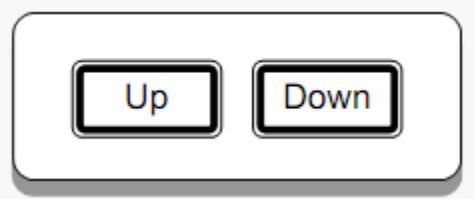
The Port Header (Port.H)

Port Header (Port.H)

```
// chân  3.0 và 3.1 đc sử dụng
// cổng RS-232
```



```
// công tắc
  sbit Sw_up  = P1^2;
sbit Sw_down  = P1^3;
```



Tập tin Port Header rất đơn giản để chúng ta hiểu và áp dụng.

Để ý rằng, Ví dụ, chúng ta có 3 file C trong một profect (A, B,
C) mỗi file yêu cầu truy cập tới 1 hoặc nhiều hơn 1 chân cảu VĐK,

File A có thể bao gồm:

```
/* File A  */
sbit Pin_A  = P3^2;
```

File B có thể bao gồm:

```
/* File B  */
#define Port_B  = P0;
```

File C có thể bao gồm:

```
/* File C  */
sbit Pin_C  = P2^7;
```

Trong phiên bản của mã code này, tất cả các port đòi hỏi truy cập trải rộng qua nhiều file.

Có rất nhiều thuận lợi thu được bằng cách tích hợp tất cả các cổng truy cập trong một file tiêu đề "port.h" :

```
/*  ----- Port.H  -----  */

/* dinh nghia cong truy cap B  */
#define Port_B  = P0;

/* dinh nghia cong truy cap A  */
sbit Pin_A  = P3^2;

/* dinh nghia cong truy cap C  */
sbit Pin_C  = P2^7;
```

```
/*-------------------------------------------------------------*-

      Port.H  (v1.01)


    'Port Header'  (xem Chap  5) cho projwct DATA_ACQ  (xem Chap  9)

-*-------------------------------------------------------------*/

#ifndef  _PORT_H
#define  _PORT_H

#include "Main.H"

/*  ------ Menu_A.C  ----------------------------------------  */ /* su dung toan bo cong port 1
va 2 cho du lieu thu duoc */
#define Data_Port1 P1
#define Data_Port2 P2


/*  ------ PC_IO.C  ----------------------------------------  */ /* chan 3.0 va 3.1 dung giao tiep

RS-232  */

#endif

/*-------------------------------------------------------------*-
    ---- END OF FILE  ------------------------------------
-*-------------------------------------------------------------*/
```

Cấu trúc lại ví dụ :"Hello World" .

```
/*-------------------------------------------------------------*-

     Main.H  (v1.00)

-*-------------------------------------------------------------*/

#ifndef  _MAIN_H
#define  _MAIN_H

/*-------------------------------------------------------
     Ban can sua phan nay cho moi project khac nhau
    -------------------------------------------------------  */

/* dat cac file tieu de thich hop cho vdk tai day  */
#include  <reg52.h>

/* tan so dao dong  (Hz) e.g.  (11059200UL)  */
#define OSC_FREQ  (12000000UL)

/* so dao dong trong huong dan  (12, etc)
      12  - hoc 8051 va phan lon cac vdk thuoc ho philip etc.
      4  - Dallas  320,  520 etc.
        1  - Dallas  420, etc.  */
#define OSC_PER_INST  (12)

/*  -------------------------------------------------------
     khong nen sua phan khai bao sau
    -------------------------------------------------------  */

/* kieu du lieu  (xem Chap  5)           */
typedef unsigned char tByte;
typedef unsigned int    tWord;
typedef unsigned long tLong;

/* khai bao ham ngat (xem Chap  7)       */
#define INTERRUPT_Timer_0_Overflow  1
#define INTERRUPT_Timer_1_Overflow  3
#define INTERRUPT_Timer_2_Overflow  5

#endif
```

```
/*-------------------------------------------------------------*-

     Port.H  (v1.00)


     'Port Header' for project HELLO2  (xem Chap  5)

-*-------------------------------------------------------------*/

#ifndef  _PORT_H
#define  _PORT_H

/*  ------ LED_Flash.C  ---------------------------------------  */

/* ket noi led toi chan nay, kem theo tro thich hop  */
sbit LED_pin  = P1^5;

#endif

/*-------------------------------------------------------------*-
   ---- END OF FILE  ------------------------------------
-*-------------------------------------------------------------*/
```

```
/*-------------------------------------------------------*-

    Main.C  (v1.00)


    Chuong trinh test cho 8051 "Hello Embedded World".

-*-------------------------------------------------------*/

#include "Main.H"
#include "Port.H"

#include "Delay_Loop.h"
#include "LED_Flash.h"

void main(void)
    {
    LED_FLASH_Init();

    while(1)
        {
        /* Thay doi trang thai led  (OFF to ON,)  */
        LED_FLASH_Change_State();

        /* Tre khoang 1000 ms  */
        DELAY_LOOP_Wait(1000);
        }
    }

/*-------------------------------------------------------*-
   ---- END OF FILE  ----------------------------------
-*-------------------------------------------------------*/
```

```
*-------------------------------------------------------------*-

     LED_flash.H  (v1.00)


     - Xem LED_flash.C de biet them chi tiet.

-*-------------------------------------------------------------*/

#ifndef  _LED_FLASH_H
#define  _LED_FLASH_H

/*  ------ ham chuc nang mau  ------------------------  */

void LED_FLASH_Init(void);
void LED_FLASH_Change_State(void);

#endif

/*-------------------------------------------------------------*-
   ---- END OF FILE  ------------------------------------
-*-------------------------------------------------------------*/
```

```
/*---------------------------------------------------------*-

    LED_flash.C  (v1.00)


    Kiem tra ham sang led don gian.

-*---------------------------------------------------------*/

#include "Main.H"
#include "Port.H"

#include "LED_flash.H"

/*  ------ dinh nghia bien rieng  ----------------------  */

static bit LED_state_G;

/*---------------------------------------------------------*-

   LED_FLASH_Init()

   Chuan bi thay doi trang thai led .

-*---------------------------------------------------------*/
void LED_FLASH_Init(void)
      {
      LED_state_G = 0;
      }
```

```
/*---------------------------------------------------------*-

   LED_FLASH_Change_State()

   Thay doi trang thai cua LED tren 1 chan out danh nghia cua vdk

-*---------------------------------------------------------*/
void LED_FLASH_Change_State(void)
     {
     /* thay doi trang thai led tu OFF → ON    */
     if (LED_state_G == 1)
          {
          LED_state_G = 0;
          LED_pin = 0;
          }
     else
          {
          LED_state_G = 1;
          LED_pin = 1;
          }
     }

/*---------------------------------------------------------*-
   ---- END OF FILE  ----------------------------------
-*---------------------------------------------------------*

     /*---------------------------------------------------------*-

     Delay_Loop.H  (v1.00)


     - Xem Delay_Loop.C de biet them chi tiet.

-*---------------------------------------------------------*/

#ifndef  _DELAY_LOOP_H
#define  _DELAY_LOOP_H

/*  ------ ham mau chuc nang  -------------------------  */
void DELAY_LOOP_Wait(const tWord DELAY_MS);

#endif

/*---------------------------------------------------------*-
   ---- END OF FILE  ----------------------------------
-*---------------------------------------------------------*/
```

```
/*---------------------------------------------------------*-

    Delay_Loop.C  (v1.00)


    Tao tre phan mem don gian trong 1 vong lap.

-*---------------------------------------------------------*/

#include "Main.H"
#include "Port.H"

#include "Delay_loop.h"

/*---------------------------------------------------------*-

   DELAY_LOOP_Wait()

   Thoi giant re voi bien la thong so .

   Thong so o day la thoi giant re ms tren  12MHz  8051  (12 osc cycles).

   Ban can sua lai thoi gian cho dung chuong trinh cua ban

-*---------------------------------------------------------*/
void DELAY_LOOP_Wait(const tWord DELAY_MS)
     {
     tWord x, y;

     for  (x  =  0; x  <= DELAY_MS; x++)
          {
          for  (y  =  0; y  <=  120; y++);
          }
     }

/*---------------------------------------------------------*-
   ---- END OF FILE  ----------------------------------
-*---------------------------------------------------------*/
```

Ví dụ: cấu trúc lại ví dụ đếm những chú dê.

```
/*-------------------------------------------------------------*-

     Main.H  (v1.00)

-*-------------------------------------------------------------*/

#ifndef  _MAIN_H
#define  _MAIN_H

/*-------------------------------------------------
     Ban can them phan nay ho moi project
   ------------------------------------------------  */

/* phai khai bao file tieu de header thich hop cho vdk o day  */
#include  <reg52.h>

/* tan so dao dong  (in Hz) e.g.  (11059200UL)  */
#define OSC_FREQ  (12000000UL)

/* so dao dong trong moi huong dan  (12, etc)
      12  - ho  8051  /  8052 va phan lon cac ic khac cua philip
      4  - Dallas  320,  520 etc.
        1  - Dallas  420, etc.  */
#define OSC_PER_INST  (12)

/*  -------------------------------------------------
     nen sua cac phan duoi day
   ------------------------------------------------  */

/* kieu du lieu                              */
typedef unsigned char tByte;
typedef unsigned int    tWord;
typedef unsigned long tLong;

/*  dinh nghia ham ngat                       */
#define INTERRUPT_Timer_0_Overflow  1
#define INTERRUPT_Timer_1_Overflow  3
#define INTERRUPT_Timer_2_Overflow  5

#endif
```

```
/*-------------------------------------------------------------*-

     Port.H  (v1.00)


     'Port Header' cho project GOATS2  (xem Chap  5)

-*-------------------------------------------------------------*/

#ifndef  _PORT_H
#define  _PORT_H

/*  ------ Switch_Wait.C  ---------------------------------------  */ /* ket noi toi pin nay  */
sbit Switch_pin  = P1^0;

/*  ------ Display_count.C  -------------------------------------  */ /* ma nhi phan ra o port 3 */
#define Count_port P3

#endif

/*-------------------------------------------------------------*-
   ---- END OF FILE  ------------------------------------
-*-------------------------------------------------------------*/
```

```
/*---------------------------------------------------------*-

    Main.C  (v1.00)


    Chuong trinh dem cho 8051.

-*---------------------------------------------------------*/

#include "Main.H"
#include "Port.H"

#include "Switch_wait.H"
#include "Display_count.H"

/*  ------------------------------------------------------  */
void main(void)
    {
    tByte Switch_presses  =  0;

    /* ham khoi tao*/
    SWITCH_Init();
    DISPLAY_COUNT_Init();

    while(1)
        {
        if  (SWITCH_Get_Input(30)  == SWITCH_PRESSED)
            {
            Switch_presses++;
            }

        DISPLAY_COUNT_Update(Switch_presses);
        }
    }

/*---------------------------------------------------------*-
   ---- END OF FILE  ----------------------------------
-*---------------------------------------------------------*/
```

```
/*-------------------------------------------------------------*-

     Switch_wait.H  (v1.00)


     - Xem Switch_wait.C de biet them chi tiet

-*-------------------------------------------------------------*/

#ifndef  _SWITCH_WAIT_H
#define  _SWITCH_WAIT_H

/*  ------ hang so chung  ----------------------------------  */ /* quay tro ve gia tri
Switch_Get_Input()  */
#define SWITCH_NOT_PRESSED  (bit)  0
#define SWITCH_PRESSED  (bit)  1

/*  ------ ham mau chuc nang chung  -------------------------  */
void SWITCH_Init(void);
bit    SWITCH_Get_Input(const tByte DEBOUNCE_PERIOD);

#endif

/*-------------------------------------------------------------*-
    ---- END OF FILE  ------------------------------------
-*-------------------------------------------------------------*/
```

```
/*-------------------------------------------------------------*-

    Switch_Wait.C  (v1.00)

    Thu vien don gian cho phong chong rung phim.

    luu y: khong thoi gin cua chuc nang nay a bien o muc cao!

-*-------------------------------------------------------------*/

#include "Main.H"
#include "Port.H"

#include "Switch_wait.h"
#include "Delay_loop.h"

/*-------------------------------------------------------------*-

   SWITCH_Init()

   Cai dat chuc nang cho thu vien nut bam.

-*-------------------------------------------------------------*/
void SWITCH_Init(void)
     {
     Switch_pin  =  1;  dung pin nay cho tin hieu vao  */
     }
```

```
/*-------------------------------------------------------------*-

   SWITCH_Get_Input()

   Doc va chong rung phim co khi nhu sau:

   1. neu nut bam chua duoc nhan quay tro ve SWITCH_NOT_PRESSED.

   2. neu nut bam duoc nhan, cho ham chong rung phim

   DEBOUNCE_PERIOD  (in ms).

-*-------------------------------------------------------------*/
bit SWITCH_Get_Input(const tByte DEBOUNCE_PERIOD)
     {
     bit Return_value  = SWITCH_NOT_PRESSED;

     if (Switch_pin == 0)
          {
          /* nut bam duoc nhan  */

          /* chong rung phim..cho (wait)  */
          DELAY_LOOP_Wait(DEBOUNCE_PERIOD);

          /* Kiem tra nut bam lan nua  */
          if (Switch_pin == 0)
               {
               /* Cho den khi nut bam nay len */
               while (Switch_pin == 0);
               Return_value  = SWITCH_PRESSED;
               }
          }

     /* cuoi cung quay tro ve gia tri ban dau  */
     return Return_value;
     }

/*-------------------------------------------------------------*-
   ---- END OF FILE  ----------------------------------
-*-------------------------------------------------------------*/
```

```
/*------------------------------------------------------------*-

    Display_count.H  (v1.00)


    - Xem Display_count.C de biet them chi tiet

-*------------------------------------------------------------*/

#ifndef  _DISPLAY_COUNT_H
#define  _DISPLAY_COUNT_H

/*  ------ Ham mau chuc nang-------------------------  */
void DISPLAY_COUNT_Init(void);
void DISPLAY_COUNT_Update(const tByte);

#endif

/*------------------------------------------------------------*-
   ---- END OF FILE  ------------------------------------
-*------------------------------------------------------------*/
```

```
/*-------------------------------------------------------------*-

      Display_count.C  (v1.00)


-*-------------------------------------------------------------*/

#include "Main.H"
#include "Port.H"

#include "Display_Count.H"

/*-------------------------------------------------------------*-

    DISPLAY_COUNT_Init()

    Khoi tao thu vien ham hien thi.

-*-------------------------------------------------------------*/
void DISPLAY_COUNT_Init(void)
      {
      Count_port  =  0x00;
      }

/*-------------------------------------------------------------*-

    DISPLAY_COUNT_Update()

    Ham hien thi don gian gia tri dem tren led qua cac cong

-*-------------------------------------------------------------*/
void DISPLAY_COUNT_Update(const tByte COUNT)
      {
      Count_port  = COUNT;
      }

/*-------------------------------------------------------------*-
    ---- END OF FILE  ----------------------------------
-*-------------------------------------------------------------*/
```

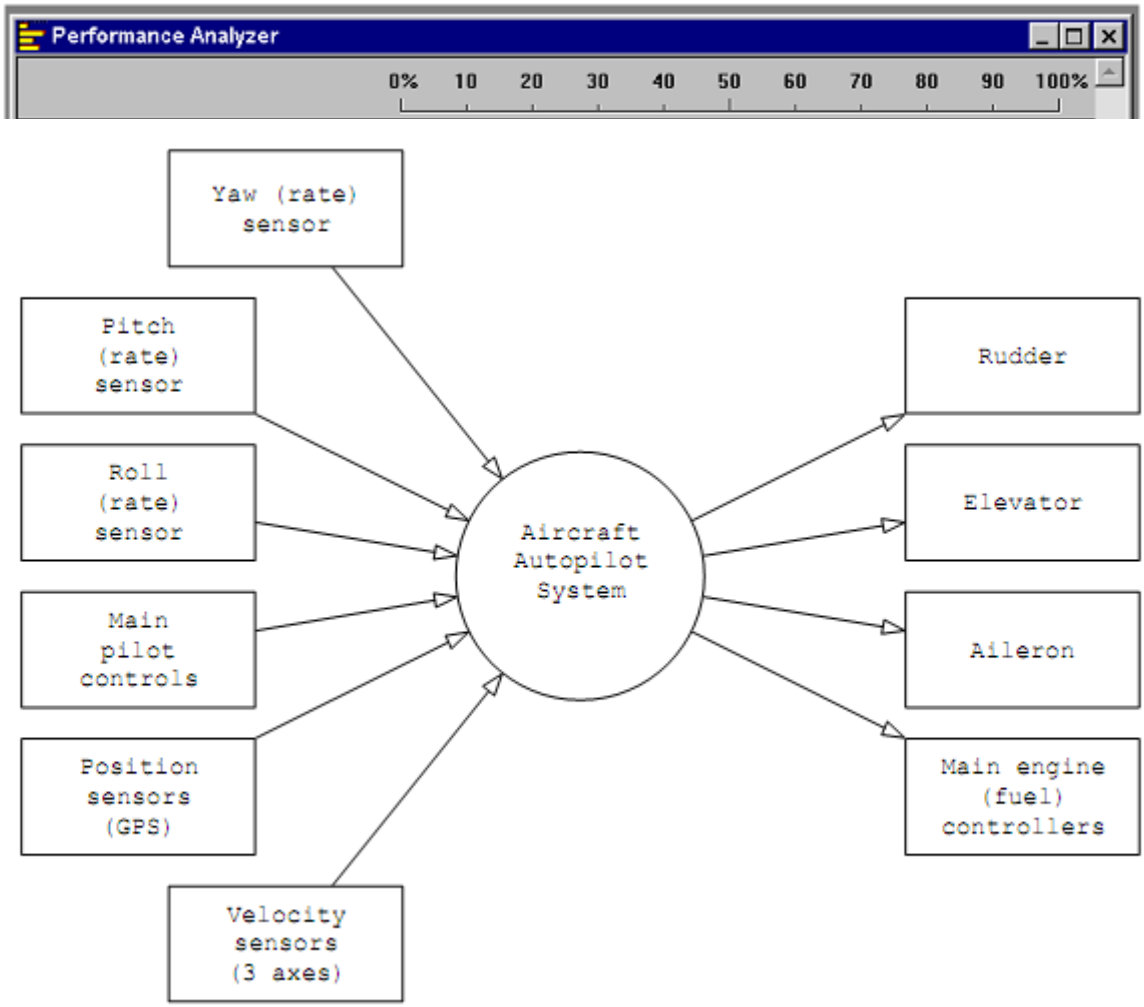
```
/*-------------------------------------------------------------*-

     Delay_Loop.H  (v1.00)


     - Xem Delay_Loop.C de biet them chi tiet.

-*-------------------------------------------------------------*/

#ifndef  _DELAY_LOOP_H
#define  _DELAY_LOOP_H

/*  ------ ham chuc nang chung  -------------------------  */
void DELAY_LOOP_Wait(const tWord DELAY_MS);

#endif

/*-------------------------------------------------------------*-
   ---- END OF FILE  ------------------------------------
-*-------------------------------------------------------------*/
```

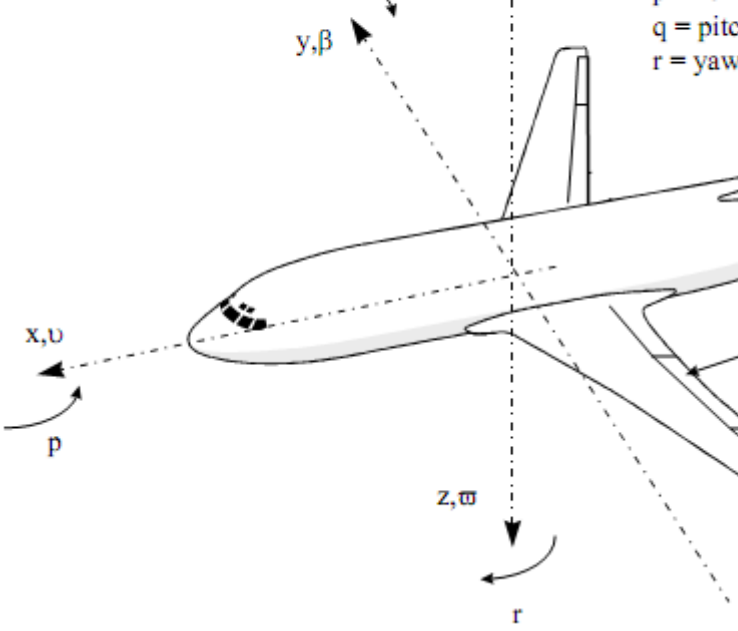
```
/*---------------------------------------------------------*-

    Delay_Loop.C  (v1.00)


    Tao ham tre phan mem do gian.

-*---------------------------------------------------------*/

#include "Main.H"
#include "Port.H"

#include "Delay_loop.h"

/*---------------------------------------------------------*-

   DELAY_LOOP_Wait()

   Thoi giant re voi cac thong so.

   Thong so tre la ms, tan so  12MHz voi 8051  (12 osc cycles).

   Ban can sua laic ho dung voi chuong trinh cua ban!

-*---------------------------------------------------------*/
void DELAY_LOOP_Wait(const tWord DELAY_MS)
     {
     tWord x, y;

     for (x = 0; x <= DELAY_MS; x++)
          {
          for (y = 0; y <= 120; y++);
          }
     }

/*---------------------------------------------------------*-
   ---- END OF FILE  ----------------------------------
-*---------------------------------------------------------*/
```

Chuẩn bị cho chương tiếp theo

Vui lòng đọc trước chương 5 trước khi chúng ta cùng nghiên cứu

Chương 5: Thời gian thực

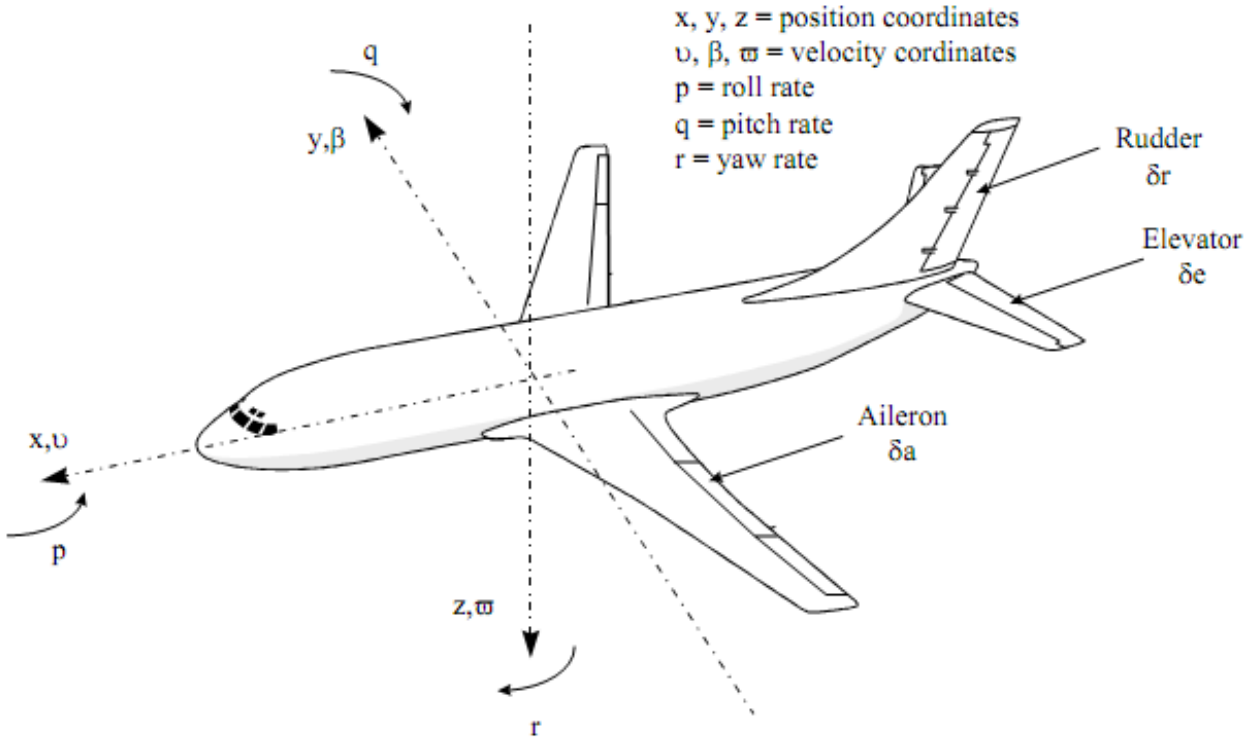




Giới thiệu

Trong chuyên đề này, chúng ta bắt đầu xem xét xác thời gian.

Vấn đề này khá quan trọng vì nhiều hệ thống nh thực.



```
bit SWITCH_Get_Input(const tByte DEBOUNCE_PERIOD)
    {
    tByte Return_value  = SWITCH_NOT_PRESSED;

    if (Switch_pin == 0)
        {
        /* nhan nut bam  */

        /* Chong rung phim  - just wait...  */
        DELAY_LOOP_Wait(DEBOUNCE_PERIOD);  /* POTENTIAL PROBLEM  */

        /* kiem tra nut bam lan nua  */
        if (Switch_pin == 0)
            {
            /* cho den khi nt bam duoc nha tay  */
            while (Switch_pin == 0);
            Return_value  = SWITCH_PRESSED;
            }
        }

    /* cuoi cung tro ve gia tri nut bam  */
    return Return_value;
    }
```

Vấn đề đầu tiên sự chống rung phím.không biết chắc lúc nào nút bấm được nhấn. Vì hàm trế là thực hiện bằng phần mềm có thể không chính xác được thời gian.

Vấn đề thứ 2 thậm chí còn nghiêm trọng hơn, trong một hệ thống với các đặc trưng thời gian thực: chúng ta làm cho hệ thống chờ đợi vô thời hạn cho sự nhả ra cảu nút bấm.

Chúng ta sẽ giải quyết vấn đề này trong chương này.

Tạo trễ phần cứng

Tất cả các thành viên họ 8051 đều có it nhất 16 bit timer/counter(như đã biết là timer 0 và timer1).

Timer này có thể sử dụng với thời gian chính xác.

```
/* Cau hinh Timer  0 nhu mot  timer 16-bit                    */
TMOD &=  0xF0;  /* Xoa tat ca bit T0 (T1 trai khong doi)  */
TMOD  |=  0x01;  /* cai dat bít T0  (T1 trai khong doi)                                  */

ET0  =  0;       /* khong cho phep ngat  */

/* gia tri cho tre  50 ms */
TH0  =  0x3C;       /* Timer  0 gia tri dau (byte cao)  */
TL0  =  0xB0;       /* Timer  0 gia tri dau  ( Byte thap)  */

TF0  =  0;                    /* xoa co tran  */
TR0  =  1;                    /* bat dau Timer  0  */

while  (TF0  ==  0);  /* choc ho den khi timer 0 tran  (TF0  ==  1)  */

TR0  =  0;                    /* dung Timer  0  */
```

Bây giờ chúng ta xem nó hoạt động thế nào……

Thanh ghi TCON SFR

Bit	7 (msb)	6	5	4	3	2	1	0 (lsb)
NAME	**TF1**	**TR1**	**TF0**	**TR0**	**IE1**	**IT1**	**IE0**	**IT0**

TF1 Cờ tràn Timer 1

Cài đặt bởi cờ tràn Timer 1.
(Xóa bởi phần cứng nếu vi xử lý ngắt thông thường .)

TR1 bit điều khiển Timer 1

Cài đặt/ xóa bởi phần mềm để Timer 1 ‘ON’ or ‘OFF’.

TF0 Cờ tràn Timer 0

Xóa bởi phần cứng trên cờ tràn Timer 0
(Xóa bởi phần cứng nếu vi xử lý ngắt thông thường.)

TR0 bit điều khiển Timer 0

Cài đặt/ xóa bởi phần mềm để Timer 0 ‘ON’ or ‘OFF’.

Lưu ý rằng cờ báo tràn của timer có thể dung để tạo ra hàm ngắt. Chúng tôi se không sử dụng hàm này trong mã code trễ phàn cứng.

Để vô hiệu hóa hàm ngắt, ta có thể dung lệnh C :

```
ET0 = 0;  /* khong cho phep ngat  (Timer  0)  */
ET1 = 0;  /* khong cho phep ngat  (Timer  1)  */
```

Thanh ghi TMOD SFR

Bit	7	6	5	4	3	2	1	0
NAME	Gate	C / T	M1	M0	Gate	C / T	M1	M0
	Timer 1				Timer 0			

Mode 1 (M1 = 0; M0 = 1)

16-bit timer/counter (nạp lại chế độ bình thường))

Mode 2 (M1 = 1; M0 = 0)

8-bit timer/counter (chế độ 8bit tự nạp lại))

GATE Gating control

Khi cài đặt, timer/counter "x" is có hiệu lực khi "INT x" ở mức cao và "TRx" điều khiển bit được cài đặt. Khi xóa timer "x" có hiệu lực bất cứ khi nào bit điều khiên "TRx" được cài đặt ("TRx" =1)

C / T Counter or timer chọn bit

Cài đặt counter hoạt động truy cập (đầu vào từ chân "Tx").
Xóa timer hoạt động truy cập (dầu vào từ hệ thống đồng hồ nội bộ)).

Hai thanh ghi bổ sung

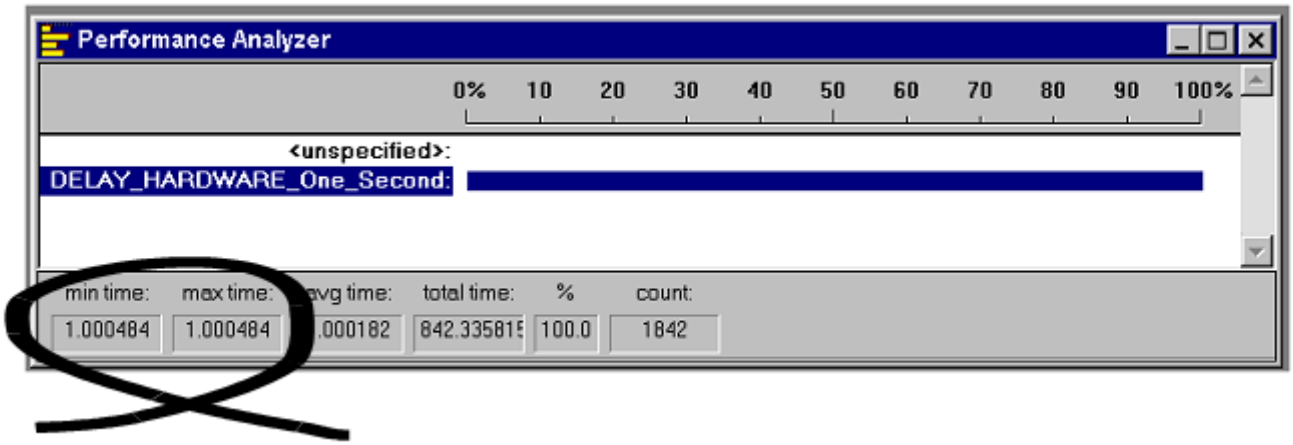
Trước khi chúng ta xem xét phần cứng này tạo ra hàm trễ như thế nào, chúng ta cần nhận thấy rằng có 2 thanh ghi bổ sung liên kết tới mỗi timer: Chúng ta gọi là TL0 and TH0, and TL1 and TH1.

Ví dụ: Generating a precise 50 ms delay

```
/*-------------------------------------------------------------*-

    Hardware_Delay_50ms.C  (v1.00)


    Kiem tra co se tre phan cung.

-*-------------------------------------------------------------*/

#include  <reg52.h>

sbit LED_pin  = P1^5;
bit LED_state_G;

void LED_FLASH_Init(void);
void LED_FLASH_Change_State(void);

void DELAY_HARDWARE_One_Second(void);
void DELAY_HARDWARE_50ms(void);

/*... ..........................................................................................................*/

void main(void)
    {
    LED_FLASH_Init();

    while(1)
        {
        /* thay doi trang thai led  (OFF to ON, or vice versa)  */
        LED_FLASH_Change_State();

        /* Tre khoang  1000 ms  */
        DELAY_HARDWARE_One_Second();
        }
    }
```

```
/*---------------------------------------------------------*-

   LED_FLASH_Init()

   Chuan bi cho ham thay doi trang thai led- xem ben duoi

-*---------------------------------------------------------*/
void LED_FLASH_Init(void)
     {
     LED_state_G = 0;
     }

/*---------------------------------------------------------*-

   LED_FLASH_Change_State()

-*---------------------------------------------------------*/
void LED_FLASH_Change_State(void)
     {
     /* doi trang thai ON-OFF  */
     if (LED_state_G == 1)
          {
          LED_state_G = 0;
          LED_pin = 0;
          }
     else
          {
          LED_state_G = 1;
          LED_pin = 1;
          }
     }
```

```
/*---------------------------------------------------------*-

   DELAY_HARDWARE_One_Second()

   Tre phan cung trong khoang   1000 ms.

   *** Gia thiet 12MHz  8051  (12 osc cycles)  ***

-*---------------------------------------------------------*/
void DELAY_HARDWARE_One_Second(void)
      {
      unsigned char d;

      /* goi ham  DELAY_HARDWARE_50ms() 20 lan  */
      for (d = 0; d < 20; d++)
            {
            DELAY_HARDWARE_50ms();
            }
      }

/*---------------------------------------------------------*-

   DELAY_HARDWARE_50ms()

   *** gia thiet 12MHz  8051  (12 osc cycles)  ***

-*---------------------------------------------------------*/
void DELAY_HARDWARE_50ms(void)
      {
      /* cau hinh  Timer  0 nhu mot  16-bit timer                    */
      TMOD &=  0xF0;  /* xoa tat ca bit  T0  (T1 trai khong doi)  */
      TMOD  |=  0x01;  /* cai dat bit yeu cau  T0 (T1 trai khong doi)                    */

      ET0  =  0;              /* cam ngat  */

      /* gia tri tre  50 ms delay  */
      TH0  =  0x3C;       /* Timer  0 gia tri ban dau ( Byte cao)  */
      TL0  =  0xB0;       /* Timer  0 gia tri ban dau  ( Byte thap)  */

      TF0  =  0;                    /* xoa co tran  */
      TR0  =  1;                    /* bat dau timer  0  */

      while  (TF0  ==  0);  /* choc ho den khi  Timer  0 tran  (TF0  ==  1)  */

      TR0  =  0;                    /* dung Timer  0  */
      }
```



Trong trường hợp này chúng ta giả thiết tần số dao động cho vđk là 12 MHz

Chúng ta yêu cầu trễ 50s, vì vậy thời gian yêu cầu :

$$\frac{50\ \text{ms}}{1000\ \text{ms}} \times 1000000 = 50000 \text{ đơn vị.}$$

Timer tràn khi bộ đếm tăng đến giá trị 65535.

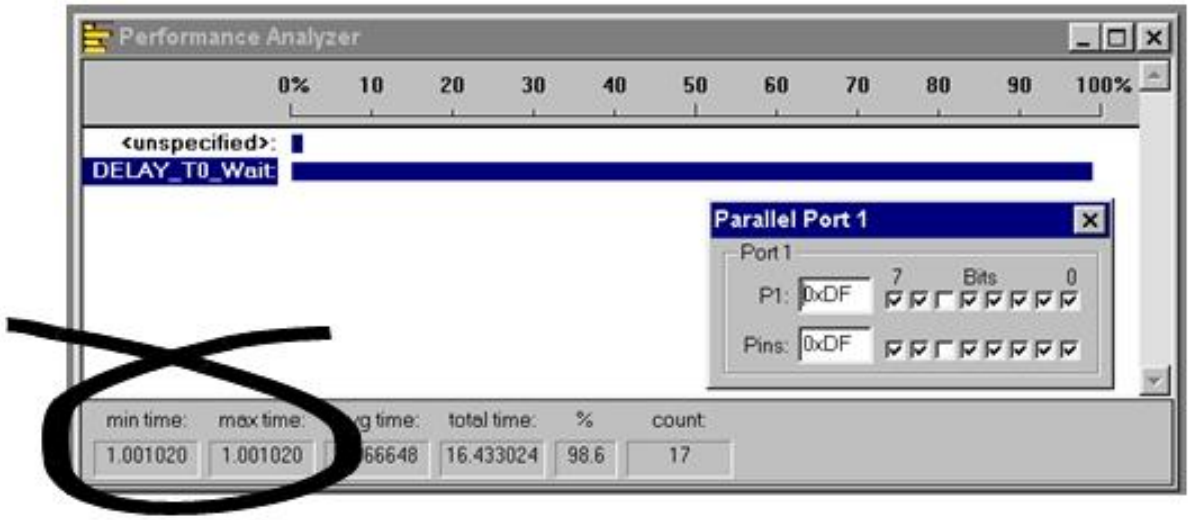
Như vậy, giá trị đầu cần nạp để trễ 50 ms delay là:

65536 - 50000 = 15536 (decimal) = 0x3CB0

Ví dụ: tạo hàm trễ phần cứng

```
/*---------------------------------------------------------*-

    Main.C  (v1.00)


    Nhay led voi ham tre phan cung  (T0).

-*---------------------------------------------------------*/

#include "Main.H"
#include "Port.H"

#include "Delay_T0.h"
#include "LED_Flash.h"

void main(void)
    {
    LED_FLASH_Init();

    while(1)
        {
        /* thay doi trang thai led  (OFF to ON,)  */
        LED_FLASH_Change_State();

        /* tre khoang 1000 ms  */
        DELAY_T0_Wait(1000);
        }
    }

/*---------------------------------------------------------*-
   ---- END OF FILE  ----------------------------------
-*---------------------------------------------------------*/
```

```
#define PRELOAD01       (65536 - (tWord)(OSC_FREQ / (OSC_PER_INST * 1020)))
#define PRELOAD01H  (PRELOAD01 / 256)
#define PRELOAD01L  (PRELOAD01 % 256)

/*-------------------------------------------------------------*/

void DELAY_T0_Wait(const tWord N)
    {
    tWord ms;

    /* cấu hình Timer  0 như một  16-bit timer                  */
    TMOD &= 0xF0;  /* xoa bit  T0  (T1 left unchanged)  */
    TMOD |= 0x01;  /* cai dat bit yeu cau T0  (T1 left unchanged)                  */

    ET0 = 0;            /* cam ngat  */

    /* tre 1 ms moi vong lap                                                */
    for (ms = 0; ms < N; ms++)
        {
        TH0 = PRELOAD01H;
        TL0 = PRELOAD01L;

        TF0 = 0;                    /* xoa co tran  */
        TR0 = 1;                    /* bat dau timer  0  */

        while (TF0 == 0);  /* cho den khi timer 0 tran  (TF0 == 1)  */

        TR0 = 0;                    /* ngung Timer  0  */
        }
    }
```


Performance Analyzer
0%
10
20
30
40
50
60
70
80
90
100%
<unspecified>:
DELAY_T0_Wait:
Parallel Port 1
Port 1
P1:
0xDF
7
Bits
0
Pins:
0xDF
min time:
max time:
g time:
total time:
%
count:
1.001020
1.001020
66648
16.433024
98.6
17

Sự cần thiết cho cơ chế 'timeout' (thời gian trễ) - Ví dụ

Philips 8Xc552 là một thiết bị mở rộng 8051 với một số thiết bị ngoại vi on-chip trong đó bao gồm cả ADC 8-kênh 10-bi. Philips cung cấp một chương trình ứng dụng (AN93017) mô tả làm thế nào để sử dụng tính năng này của vđk.

Chương trình này bao gồm đoạn mã sau:

```
/* chờ cho đến khi ADC chuyển đổi xong  (checking ADCI)  */
while  ((ADCON & ADCI)  ==  0);
```

Mã code như vậy có thể không đáng tin cậy, bởi vì có một số trường hợp mà ứng dụng có thể bị treo. Điều này xảy ra vì một trong những lý do sau :

• Nếu ADC khởi tạo không chính xác, chúng ta không thể chắc chắn rằng một chuyển đổi có thể được thực hiện.

• Nếu ADC phải chịu một điện áp đầu vào quá nhiều, sau đó có thể nó sẽ không hoạt động.

• Nếu biến ADCON or ADCI không khởi tạo đúng, Chúng có thể không hoạt động theo yêu cầu.

Ví dụ của Philips không có ý định 'sản xuất' code. Thật không may, tuy nhiên code này đã phổ biến trong nhiều chương trình nhúng.

Hai giải pháp giải quyết vấn đề: Loop timeouts và hardware timeouts.

(ngưng trễ dạng vòng lặp và ngưng trễ phần cứng)

Tạo loop timeouts

Cơ sở của loop timeout:

```
tWord Timeout_loop = 0;


while (++Timeout_loop);
```

Original ADC code:

```
/* Doi cho den khi chuyen doi ket thuc (checking ADCI) */
while ((ADCON & ADCI) == 0);
```

Modified version, with a loop timeout:

```
tWord Timeout_loop = 0;

/* doi cho den khi chuyen doi ket thuc

 (checking ADCI) –vong thoi giant tre don gian */
while (((ADCON & ADCI) == 0) && (++Timeout_loop != 0));
```

Note that this alternative implementation is also useful:

```
while (((ADCON & ADCI) == 0) && (Timeout_loop != 0))
    {
    Timeout_loop++;       /* ngat su dung phan cung ao */
    }
```

```
/*-------------------------------------------------------------*-

     TimeoutL.H  (v1.00)


     Thoi giant tre don gian voi cac thanh vien ho 8051

     * nhung gia tri nay khong chinh xac –ban phai lap vao he thong cua ban*

-*-------------------------------------------------------------*/

#ifndef  _TIMEOUTL_H
#define  _TIMEOUTL_H

/*  ------ Public constants  ---------------------------------  */

/* bien nay thay doi thoi gian lap
     Gia  tri nay co cac khoang thoi giant re tren  8051,  12 MHz,  12 Osc
     / cycle

     *** phai chinh xac dung voi chuong trinh cua ban  ***

     *** thoi gian khac nhau voi cac thiet lap toi uu hoa trinh

     bien dich  *** */

/* tWord  */
#define LOOP_TIMEOUT_INIT_001ms  65435
#define LOOP_TIMEOUT_INIT_010ms  64535
#define LOOP_TIMEOUT_INIT_500ms  14535
/* tLong  */
#define LOOP_TIMEOUT_INIT_10000ms  4294795000UL

#endif

/*-------------------------------------------------------------*-
   ---- END OF FILE  ------------------------------------
-*-------------------------------------------------------------*/
```

Ví dụ: kiểm tra loop timeouts

```
/*-------------------------------------------------------*-

     Main.C  (v1.00)

-*-------------------------------------------------------*/

#include  <reg52.H>

#include "TimeoutL.H"

/* dinh nghia kieu bien  (xem Chap  5) */
typedef unsigned char tByte;
typedef unsigned int    tWord;
typedef unsigned long tLong;

/* khai bao ham  */
void Test_Timeout(void);

/*-------------------------------------------------------*/
void main(void)
     {
     while(1)
          {
          Test_Timeout();
          }
     }

/*-------------------------------------------------------*/
void Test_Timeout(void)
     {
     tLong Timeout_loop  = LOOP_TIMEOUT_INIT_10000ms;

     /* ngung tre vong lap don gian...  */
     while  (++Timeout_loop  !=  0);
     }
```

Ví dụ: nút bấm tin cậy hơn

```
bit SWITCH_Get_Input(const tByte DEBOUNCE_PERIOD)
    {
    tByte Return_value  = SWITCH_NOT_PRESSED;
    tLong Timeout_loop  = LOOP_TIMEOUT_INIT_10000ms;

    if (Switch_pin == 0)
        {
        /* nut bam duoc nhan  */

        /* chong rung  - chi can doi...  */
        DELAY_T0_Wait(DEBOUNCE_PERIOD);

        /* khiem tra nut bam lan nua  */
        if (Switch_pin == 0)
            {
            /* doi cho den khi nut bam nay len
                (voi vong lap thoi gian tre  -  10 s)  */
            while ((Switch_pin == 0) && (++Timeout_loop != 0));

            /* kiem tra vong lap thoi gian tre  */
            if (Timeout_loop == 0)
                {
                Return_value  = SWITCH_NOT_PRESSED;
                }
            else
                {
                Return_value  = SWITCH_PRESSED;
                }
            }
        }

    /* bay gio quay tro ve gia tri nut bam  */
    return Return_value;
    }
```

Tạo hardware timeouts

```
/* cau hinh Timer  0 nhu mot  16-bit timer                    */
TMOD &=  0xF0;  /* xoa bit T0  (T1 trai khong doi)  */
TMOD  |=  0x01;  /* cai dat bit yeu cau T0  (T1 trai khong doi                    */

ET0  =  0;              /* cam ngat  */

/* thoi giant tre don gian khoang 10ms  */
TH0  = PRELOAD_10ms_H;  /* Xem Timeout.H for PRELOAD de biet them  */
TL0  = PRELOAD_10ms_L;
TF0  =  0;  /* xoa co tran  */
TR0  =  1;  /* bat dau timer  */

while  (((ADCON & ADCI)  ==  0) &&  !TF0);
```

```
/*-------------------------------------------------------------*-

     TimeoutH.H  (v1.00)

-*-------------------------------------------------------------*/

#ifndef  _TIMEOUTH_H
#define  _TIMEOUTH_H

/*  ------ hang so chung  ----------------------------------  */

/* gia tri Timer T_ su dung don giant hoi gian tre (phan cung )  */
      - Timers 16-bit, che do nap lai binh thuong('one shot').  */

      Luu y:  Cac che do dinh thoi phai kiem tra bang tay cho hinh xac .  */

/* dinh nghia ban dau  Timer  0  / Timer  1 gia tri de tre 50us  */
#define T_50micros  (65536  -  (tWord)((OSC_FREQ  /
26000)/(OSC_PER_INST)))
#define T_50micros_H  (T_50micros  /  256)
#define T_50micros_L  (T_50micros  %  256)



/* dinh nghia ban dau  Timer  0  / Timer  1 gia tri tre  ~10 ms*/
#define T_10ms          (65536  -  (tWord)(OSC_FREQ  /  (OSC_PER_INST  *  100)))
#define T_10ms_H  (T_10ms  /  256)
#define T_10ms_L          (T_10ms  %  256)

/* dinh nghia ban dau  Timer  0  / Timer  1 gia tri tre  ~15 ms  */
#define T_15ms          (65536  -  (tWord)(OSC_FREQ  /  (OSC_PER_INST  *  67)))
#define T_15ms_H  (T_15ms  /  256)
#define T_15ms_L          (T_15ms  %  256)

/* dinh nghia ban dau  Timer  0  / Timer  1 gia tri tre  ~20 ms  */
#define T_20ms          (65536  -  (tWord)(OSC_FREQ  /  (OSC_PER_INST  *  50)))
#define T_20ms_H  (T_20ms  /  256)
#define T_20ms_L          (T_20ms  %  256)

/* dinh nghia ban dau  Timer  0  / Timer  1 gia tri tre  ~50 ms  */
#define T_50ms          (65536  -  (tWord)(OSC_FREQ  /  (OSC_PER_INST  *  20)))
#define T_50ms_H  (T_50ms  /  256)
#define T_50ms_L  (T_50ms  %  256)

#endif
```

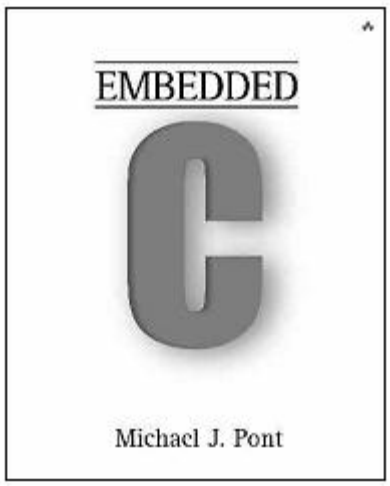
Kết luận

Việc trễ và thời gian chờ xem xét trong chương này sử dụng rất rộng rãi trong các chương trình nhúng.

Trong chương tiếp theo chúng ta xem xét phần mềm khác cấu thành nhiều chương trình nhúng: hệ điều hành..

Chuẩn bị cho chương tiếp theo

Vui lòng xem trước chương 6



Chương 6: Tạo ra một hệ điều hành nhúng

Xác định tốc độ dòng chảy dựa vào dòng xung

Hệ thống tiệt trùng sữa

```
void main(void)
    {
    /* chuẩn bị chạy hàm X */
    X_Init();

    while(1) /* 'mãi mãi' (Vòng lặp vô tận) */
        {
        X();     /* chạy hàm X */
        }
    }
```

Một giới hạn riêng của kiểu kiến trúc này đó là rất khó để thi hành hàm X() ở những khoảng thời gian chính xác: như chúng ta sẽ thấy , đó là một mặt hạn chế đáng kể.

Chú ý đến một loạt các yêu cầu được thu thập từ một dãy các dự án nhúng(trong kiểu kiến trúc không đặc biệt):

Tốc độ hiện tại của vật chuyển động phải được đo ở những khoảng thời gian 0.5s.

Kết quả hiển thị phải được lặp lại 40 lần trong 1 giây.

Các cài đặt mới phải được áp dụng trong mỗi khoảng thời gian 0.5 giây

Sự thay đổi về tần số thời gian phải được áp dụng 20 lần mỗi giây.

Dữ liệu dao động phải được lấy mẫu 1000 lần trong một giây.

Dữ liệu miền tần số phải được phân loại 20 lần mỗi giây.

Bàn phím phải được quét trong mỗi 200ms.

Các nút điều khiển chính phải kết nối với tất cả các nút khác một lần trên một giây.

Các cài đặt điều chỉnh mới phải được tính toán mỗi 0.5 giây.

Các cảm biến phải được lấy mẫu một lần trong một giây.

Trên thực tế , rất nhiều hệ thống nhúng sẽ phải chấp nhận kiểu ‘hàm chu kì’ này.

```
void main(void)
    {

    Init_System();

    while(1) /* 'mãi mãi' (Vòng lặp vô tận) */
        {
        X();                /* Gọi hàm này (Tồn tại 10 ms) */
        Delay50ms(); /* Trễ 50 ms */
        }
    }
```

Điều này sẽ tốt nếu:

1. Nếu chúng ta biết chính xác thời gian tồn tại của hàm X()và

2. Khoảng thời gian này không bao giờ biến thiên.

Các ngắt Timer cơ bản (Lõi của một hệ điều hành nhúng)

```
#define INTERRUPT_Timer_2_Overflow 5
...

void main(void)
    {
    Timer_2_Init();         /* Cài đặt Timer 2 */

    EA = 1;                 /* Cho phép toàn bộ các ngắt hoặt động */

    while(1);               /* vòng lặp vô tận rỗng */
    }

void Timer_2_Init(void)
    {
    /* Timer 2 là một Timer 16 bit , có thể tự động nạp lại khi bị tràn giá trị

        Code này (Phù hợp với 8051/52) giả sử hệ thống dao động với tần số 12Mhz.Như vậy
        Timer 2 có độ phân giải là 1.000 µs

        Nạp lại giá trị FC18 (hex) = 64536 (Thập phân)
        Timer (16-bit) sẽ tràn khi tới giá trị 65536 (Thập phân)
        Vì vậy,với những cài đặt sau, timer sẽ tràn sau mỗi khoảng thời gian 1 ms */
    T2CON = 0x04;       /* Nạp giá trị thanh ghi T2CON */

    TH2     = 0xFC;     /*  Load  T2  high  */
    RCAP2H= 0xFC;       /*  Load  T2  reload capt. reg. high byte */
    TL2     = 0x18;     /*  Load  T2  low byte */
    RCAP2L= 0x18;       /*  Load  T2  reload capt. reg. low byte */

    /* Ngắt timer 2 đã được cho phép, và ISR sẽ được gọi đến bất cứ khi nào Timer này tràn –
    xem bên dưới. */
    ET2 = 1;

    /* Bắt đầu chạy Timer 2 */
    TR2 = 1;
    }

void X(void) interrupt INTERRUPT_Timer_2_Overflow
    {
    /* ISR này sẽ được gọi sau mỗi khoảng thời gian 1ms */
    /* Đặt những code cần thiết ở đây */
    }
```

Chương trình con phục ngắt (ISR)

Ngắt được tạo ra bởi việc ngắt của Timer 2 , gọi ISR như sau:

```
/* ------------------------------------------------------------ */
void X(void) interrupt INTERRUPT_Timer_2_Overflow
    {
    /* ISR này sẽ được gọi sau mỗi khoảng thời gian 1ms */
    /* Đặt những code cần thiết ở đây */
    }
```

Mối liên lạc giữa hàm này và sự tràn của Timer được tạo ra bằng cách sử dụng từ khóa Keil interrupt:

void X(void) interrupt INTERRUPT_Timer_2_Overflow

…cộng với #define sau chỉ dẫn:

#define INTERRUPT_Timer_2_Overflow 5

Ngắt	Địa chỉ	Số hiệu IE
Reset hệ thống	0x00	-
Ngắt ngoài 0	0x03	0
Tràn Timer 0	0x0B	1
Ngắt ngoài 1	0x13	2
Tràn Timer 1	0x1B	3
Port nối tiếp	0x23	4
Tràn Timer 2	0x2B	5

Timer tự động nạp lại

```
/* Nạp trước các giá trị để trễ 50ms */
TH0 = 0x3C;          /* Timer 0: Giá trị khởi tạo ( Byte cao) */
TL0 = 0xB0;          /* Timer 0: Giá trị khởi tạo (Byte thấp) */

TF0 = 0;             /* Xóa cờ tràn */
TR0 = 1;             /* Bắt đầu timer 0 */

while (TF0 == 0); /* Lặp đến khi Timer o tràn (TF0 == 1) */

TR0 = 0;             /* dừng Timer 0 */
```

Đối với hệ điều hành, chúng ta có thêm các yêu cầu nhỏ sau:

chúng ta yêu cầu một chuỗi các ngắt ở những khoảng thời gian chính xác định trước.

Chúng ta muốn tạo ra các ngắt này mà không without tạo ra các tải đáng kể trên CPU.

Timer 2 phù hợp với những yêu cầu chính xác.

Trong trường hợp này, timer 2được nạp lại bằng việc sử dụng nội dung của thanh ghi ‘thu thập’(chú ý rằng tên của các thanh ghi này có một chút thay đổi nhỏ tùy vào hãng sản xuất):

```
RCAP2H = 0xFC;      /* Nạp vào giá trị nạp lại byte cao */
RCAP2L = 0x18;      /* Nạp vào giá trị nạp lại byte thấp*/
```

Khả năng tự động nạp lại đảm bảo rằng timer này sẽ giữ nguyên các giá trị yêu cầu ở những khoảng thời gian chính xác 1ms,với rất ít phần mềm được tạo mà không có bất kì sự can thiệp nào từ phía người dùng.

```
void main(void)
    {
    Init_System();

    while(1) /* Vòng lặp vô tận */
        {
        X();                /* Gọi hàm này (Duy trì trong 10ms) */
        Delay_50ms(); /*Trễ 50 ms */
        }
    }
```

Trong trường hợp này:

Chúng ta sử dụng một vòng lặp vô tận và hàm trễ

Chúng ta gọi hàm X() sau mỗi khoảng thời gian xấp xỉ 60ms .

Chúng ta sẽ cùng xem một cách làm tốt hơn cách này …

Giới thiệu về sEOS

Xem cuốn "Embedded C", Chương 7

```
/*-------------------------------------------------------*-

    Main.c (v1.00)

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

    Minh họa cho sEOS chạy một tác vụ giả .

-*-------------------------------------------------------*/

#include "Main.H"
#include "Port.H"
#include "Simple_EOS.H"

#include "X.H"

/* ------------------------------------------------ */

void main(void)
    {
    /* Chuẩn bị cho một tác vụ giả */
    X_Init();

    /* Cài đặt EOS đơn giản (khoảng thời gian tick 60 ms) */
    sEOS_Init_Timer2(60);

    while(1) /* Vòng lặp vô tận */
        {
        /* Đưa ra chế độ nghỉ để tiết kiệm năng lượng */
        sEOS_Go_To_Sleep();
        }
    }

/*-------------------------------------------------------*-
   ---- KẾT THÚC FILE -------------------------------
-*-------------------------------------------------------*/
```

```
/*---------------------------------------------------*-

    Simple_EOS.C (v1.00)

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

    File chính cho hệ điều hành đơn giản (Simple Embedded Operating System: sEOS).

    Phiên bản chứng thực với tác vụ giả  X().

-*---------------------------------------------------*/

#include "Main.H"
#include "Simple_EOS.H"

/* Header cho tác vụ giả                */
#include "X.H"

/*---------------------------------------------------*-

   sEOS_ISR()

   Gọi ra một cách định kì bởi việc tràn Timer 2:
   Xem hàm sEOS_Init_Timer2() cho các định thời chi tiết.

-*---------------------------------------------------*/
sEOS_ISR() interrupt INTERRUPT_Timer_2_Overflow
    {
    /* Phải tự reset cờ T2 */
    TF2 = 0;

    /*===== Mã người dùng – Bắt đầu ============================== */

    /* Gọi tác vụ giả ở đây */
    X();

    /*===== Mã người dùng –Hết ============================== */
    }
```

```
/*------------------------------------------------------*-

   sEOS_Init_Timer2()

-*------------------------------------------------------*/
void sEOS_Init_Timer2(const tByte TICK_MS)
     {
     tLong Inc;
     tWord Reload_16;
     tByte Reload_08H, Reload_08L;

     /* Timer 2 là một Timer 16 bit, tự động nạp lại khi bị tràn */
     T2CON = 0x04;        /* Nạp giá trị thanh ghi T2CON */

     /* Lượng gia tăng timer yêu cầu (Lớn nhất 65536) */
     Inc = ((tLong)TICK_MS * (OSC_FREQ/1000)) /
               (tLong)OSC_PER_INST;

     /* 16-bit nạp lại giá trị */
     Reload_16 = (tWord) (65536UL - Inc);

     /* 8-bit nạp lại giá trị (cao & thấp) */
     Reload_08H = (tByte)(Reload_16 / 256);
     Reload_08L = (tByte)(Reload_16 % 256);

     /* được sử dụng để kiểm tra sự định thời (trong mô hình) */
     /*P2 = Reload_08H; */
     /*P3 = Reload_08L; */

     TH2    = Reload_08H;    /* Nạp T2 Byte cao */
     RCAP2H = Reload_08H;    /* Nạp T2 Giá trị nạp lại(cao) */
     TL2    = Reload_08L;    /* Nạp T2 byte thap */
     RCAP2L = Reload_08L;    /* Nạp T2 giá trị nạp lại(thấp) */

     /* ngắt Timer 2 đã được cho phép, và ISR sẽ bị gọi bất cứ khi nào timer tràn. */
     ET2 = 1;

     /* bắt đầu chạy timer 2 */
     TR2 = 1;

     EA = 1;                  /* cho phép ngắt toàn bộ */
     }
```

```
/*---------------------------------------------------*-

   sEOS_Go_To_Sleep()

   hệ điều hành này đưa vào chế độ 'nghỉ'giữa các xung nhịp để tiết kiệm năng lượng. xung
   nhịp tiếp theo sẽ trả lại chế độ bình thường cho bộ xử lý

-*---------------------------------------------------*/
void sEOS_Go_To_Sleep(void)
     {
     PCON |= 0x01;        /* chế độ nghỉ (họ 8051) */
     }

/*---------------------------------------------------*-
   ---- KẾT THÚC FILE -------------------------------
-*---------------------------------------------------*/
```

```
/*-------------------------------------------------------*-

    X.C (v1.00)

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

    Tác vụ giả để giới thiệu về sEOS.

-*-------------------------------------------------------*/

#include "X.H"

/*-------------------------------------------------------*-

   X_Init()

   Hàm bắt đầu.

-*-------------------------------------------------------*/
void X_Init(void)
     {
     /* tác vụ giả bắt đầu... */
     }

/*-------------------------------------------------------*-

   X()

   Tác vụ giả được gọi từ việc ngắt.

-*-------------------------------------------------------*/
void X(void)
     {
     /* tác vụ giả... */
     }

/*-------------------------------------------------------*-
   ---- KẾT THÚC FILE ------------------------------
-*-------------------------------------------------------*/
```

Tác vụ, hàm và lập trình

Khi đề cập đến các hệ thống nhúng, bạn sẽ thường xuyên nghe thấy cụm từ‘thiết kế nhiệm vụ’, ‘nhiệm vụ chu kì’ và
‘hệ xử lý đa nhiệm’.

Trong trường hợp này, thuật ngữ ‘tác vụ’ luôn được sử dụng để chỉ một hàm mà hàm đó được thực thi trong một chu kì chuẩn.

Trong trường hợp của sEOS, các tác vụ sẽ được thực thi bằng cách sử dụng các hàm được gọi từ dịch vụ ngắt của timer.

Cài đặt khoảng thời gian ‘tick’

Trong hàm main(), chúng ta có thể thấy rằng khoảng thời gian tick của hệ thống đã được tự động hóa ở mức độ lớn:

```
/* Cài đặt EOS đơn giản (khoảng thời gian tick 60 ms) */
sEOS_Init_Timer2(60);
```

Trong ví dụ này, một khoảng thời gian 60ms được sử dụng: Điều này có nghĩa là ISR (hàm ‘cập nhật’) ở lõi của sEOS sẽ được gọi ra sau mỗi 60ms:

```
/*------------------------------------------------------*-

   sEOS_ISR()

   Gọi ra định kì khi Timer 2 tràn:
   Xem hàm sEOS_Init_Timer2() cho định thời chi tiết.

-*------------------------------------------------------*/
sEOS_ISR() interrupt INTERRUPT_Timer_2_Overflow
    {
    …
    }
```

Tự động điều khiển thời gian đã đạt được bằng cách bộ tiền xử lý C, và các thông tin bao gồm ở header file
(Main.H):

```
/* Xung nhịp / tần số cộng hưởng (Hz) ví dụ 11059200 */
#define OSC_FREQ (12000000UL)

/* Số lượng dao động trên một chương trình (12, v.v) */
...
#define OSC_PER_INST (12)
```

Thông tin này được sử dụng tiếp theo để tính toán Timer yêu cầu được nạp lại giá trị trong Simple_EOS.C như sau:

```
    /* lượng gia tăng Timer yêu cầu (lớn nhất là 65536) */
    Inc = ((tLong)TICK_MS * (OSC_FREQ/1000)) / (tLong)OSC_PER_INST;

    /* 16-bit nạp lại giá trị */
    Reload_16 = (tWord) (65536UL - Inc);

    /* 8-bit nạp lại giá trị (cao & thấp) */
    Reload_08H = (tByte)(Reload_16 / 256);
    Reload_08L = (tByte)(Reload_16 % 256);

…

    TH2   = Reload_08H;   /* Nạp T2 Byte cao */
    RCAP2H= Reload_08H;   /* Nạp T2 (giá trị cần nạp lại(byte cao)) */
    TL2   = Reload_08L;   /* Nạp T2 Byte thấp */
    RCAP2L= Reload_08L;   /* Nạp T2 (giá trị cần nạp lại(byte thấp)) */
```

Nếu sử dụng bộ dao động 12MHz, thì có thể định thời chính xác trong khoảng xấp xỉ từ 1 ms đến 60 ms .

Nếu sử dụng những tần số khác (ví dụ 11.0592 MHz), thì định thời
chính xác chỉ thu được trong mốt số khoảng giới hạn.

Nếu bạn phát triển một ứng dụng yêu cầu định thời chính xác, bạn phải tự kiểm tra kết quả định thời.

```
/* sử dụng để kiểm tra sự định thời một cách không tự động (trong mô phỏng) */
P2 = Reload_08H;
P3 = Reload_08L;
```

Tiết kiệm năng lượng

Sử dụng sEOS, chúng ta có thể giảm bớt lượng tiêu thụ của ứng dụng bằng cách sử dụng chế độ ngủ khi bộ xử lý thực thi xong ngắt IRS.

Điều này đạt được bằng cách sử dụng hàm sEOS_Go_To_Sleep():

```
/*-------------------------------------------------------*-

   sEOS_Go_To_Sleep()

   Hệ điều hành này đưa vào 'chế độ nghỉ' giữa các xung nhịp để tiết kiệm năng lượng.xung
   nhịp tiếp theo sẽ trả lại chế độ bình thường cho hệ thống.

-*-------------------------------------------------------*/
void sEOS_Go_To_Sleep(void)
    {
    PCON |= 0x01;        /* nhập vào chế độ nghỉ (họ 8051) */
    }
```

Chú ý rằng bộ xử lý sẽ tự động trả lại chế độ ‘bình thường’ khi Timer tiếp theo ngắt (sinh ra một ngắt).

Linh kiện	Bình thường	Nghỉ	Power Down
Atmel 89S53	11 mA	2 mA	60 A

Sử dụng sEOS trong dự án của chính bạn

Khi sử dụng sEOS trong các ứng dụng của chính bạn, bạn sẽ cần đến cả bản sao của các file Simple_EOS.C và Simple_EOS.H trong dự án của bạn: file .C sẽ cần được chỉnh sửa cho phù hợp với yêu cầu của bạn trong phần minh họa dưới đây:

```
sEOS_ISR() interrupt INTERRUPT_Timer_2_Overflow
    {
    /* phải tự reset cờ nhớ T2 */
    TF2 = 0;

    /*===== USER CODE - Begin ============================== */

    /* thêm hàm của bạn (TASK) ở đây... */

    /*===== USER CODE - End ================================ */
    }
```

Phương pháp này có linh hoạt không?

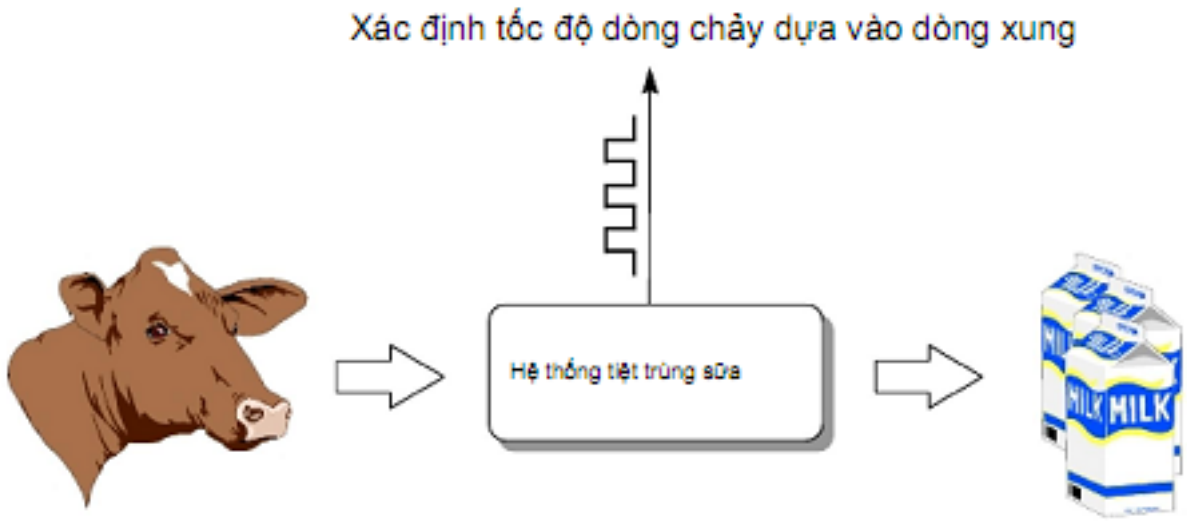
Sự có mặt của một Timer trên chip, được sử dụng để tạo ra ngắt bằng cách này không có nghĩa là chỉ giới hạn cho họ 8051.: hầu như tất cả các bộ xử lý dùng cho các hệ thống nhúng đều có các Timer, có thể được sử dụng theo một kiểu cách rât giống các miêu tả ở đây.

Ví dụ, các bộ Timer tương tự cũng có trên các vi điều khiển 8-bit khác (Họ vi điều khiển PIC,họ vi điều khiển của Motorola

HC08), và cũng có trên các thiết bị 16-bit khác (ví dụ như họ Infineon

C167) và cả trên các bộ xử lý 32-bit (ví dụ như họ ARM,họ Motorola MPC500).

Ví dụ: Tiệt trùng sữa



```
/*-------------------------------------------------------------*-

    Port.H (v1.00)

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

   File tiêu đề Port cho ví dụ về hệ thống tiệt trùng sữa
   ("Embedded C" Chương 7)

-*-------------------------------------------------------------*/

/* ------ Pulse_Count.C --------------------------------- */

/* kết nối đầu vào xung với chân này-được hãm bằng phần mềm */
sbit Sw_pin = P3^0;

/* kết nối báo động với chân này (cài và nếu xung ở dưới mức cho phép) */
sbit Alarm_pin = P3^7;

/* ------ Bargraph.C ----------------------------------- */

/* Bargraph hiển thị lên các chân này (8 chân nối này có thể được phân bố theo các cách
khác nhau, nếu bị yêu cầu). */
sbit Pin0 = P1^0;
sbit Pin1 = P1^1;
sbit Pin2 = P1^2;
sbit Pin3 = P1^3;
sbit Pin4 = P1^4;
sbit Pin5 = P1^5;
sbit Pin6 = P1^6;
sbit Pin7 = P1^7;


/*-------------------------------------------------------------*-
   ---- END OF FILE ------------------------------------
-*-------------------------------------------------------------*/
```

```
/*-------------------------------------------------------------*-

    Main.c (v1.00)

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

    Ví dụ về hệ thống tiệt trùng sữa.

-*-------------------------------------------------------------*/

#include     "Main.H"
#include     "Port.H"
#include     "Simple_EOS.H"
#include     "Bargraph.H"

#include "Pulse_Count.H"

/* ------------------------------------------------------------ */

void main(void)
    {
    PULSE_COUNT_Init();
    BARGRAPH_Init();

    /* cài đặt simple EOS (khoảng thời gian tick 30ms) */
    sEOS_Init_Timer2(30);

    while(1) /* vòng lặp vô tận */
        {
        /* đưa vào chế độ nghỉ để tiết kiệm năng lượng */
        sEOS_Go_To_Sleep();
        }
    }

/*-------------------------------------------------------------*-
   ---- END OF FILE ------------------------------------
-*-------------------------------------------------------------*/
```

```
/*-------------------------------------------------------*-

    Simple_EOS.C (v1.00)

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

    File chính cho hệ điều hành nhúng đơn giản(sEOS) sử dụng 8051.

    -- phiên bản này để kiểm tra tốc độ dòng chảy của sữa.

-*-------------------------------------------------------*/

#include "Main.H"
#include "Simple_EOS.H"

#include "Pulse_count.H"


/*-------------------------------------------------------*-

   sEOS_ISR()

   Được gọi ra một cách đình kì bởi việc tràn timer 2:
   Xem hàm sEOS_Init_Timer2()để biết chi tiết định thời.

-*-------------------------------------------------------*/
sEOS_ISR() interrupt INTERRUPT_Timer_2_Overflow
    {
    /* phai tu reset cờ nhớ T2 */
    TF2 = 0;

    /*===== USER CODE - Begin ================================== */

    /* Gọi hàm 'cập nhật' ở đây */
    PULSE_COUNT_Update();

    /*===== USER CODE - End ==================================== */
    }
```

```
/*------------------------------------------------------------*-

   sEOS_Init_Timer2()

   ...

-*------------------------------------------------------------*/
void sEOS_Init_Timer2(const tByte TICK_MS)
     {
     tLong Inc;
     tWord Reload_16;
     tByte Reload_08H, Reload_08L;

     /* Timer 2 là một Timer 16 bit,tự động nạp lại giá trị khi bị tràn */
     T2CON = 0x04;        /* nạp thanh ghi điều khiển T2 */

     /* độ lớn yêu cầu(lớn nhất là 65536) */
     Inc = ((tLong)TICK_MS * (OSC_FREQ/1000)) / (tLong)OSC_PER_INST;

     /* giá trị 16 bit nạp lại */
     Reload_16 = (tWord) (65536UL - Inc);

     /* giá trị 8 bit nạp lại (cao & thấp) */
     Reload_08H = (tByte)(Reload_16 / 256);
     Reload_08L = (tByte)(Reload_16 % 256);

     /* được sử dụng để tự kiểm tra định thời (trong mô phỏng) */
     /*P2 = Reload_08H; */
     /*P3 = Reload_08L; */

     TH2    = Reload_08H;    /* Load T2 high byte */
     RCAP2H = Reload_08H;    /* Load T2 reload capt. reg. high byte */
     TL2    = Reload_08L;    /* Load T2 low byte */
     RCAP2L = Reload_08L;    /* Load T2 reload capt. reg. low byte */

     /* ngắt Timer 2 đã được cho phép, ISR sẽ bị gọi bất cứ khi nào Timer tràn. */
     ET2 = 1;

     /* bắt đầu chạy timer 2*/
     TR2 = 1;

     EA = 1;                  /* cho phep toan bo cac ngat */
     }
```

```
/*---------------------------------------------------------*-

   sEOS_Go_To_Sleep()

   hệ điều hành sẽ đưa vào 'chế độ nghỉ' giữa các tick đồng hồ
   để tiết kiệm năng lượng. tick đồng hồ tiếp theo sẽ trả lại trạng thái
   bình thường cho bộ xử lý.

-*---------------------------------------------------------*/
void sEOS_Go_To_Sleep(void)
    {
    PCON |= 0x01;        /* đưa vào chế độ nghỉ (họ 8051) */
    }

/*---------------------------------------------------------*-
   ---- END OF FILE ----------------------------------
-*---------------------------------------------------------*/
```

```
/*-------------------------------------------------------------*-

    Pulse_Count.C (v1.00)

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

    Đếm xung từ một thiết bị ngắt cơ khí hay thiết bị tương tự.
                                                          ___
    ___

-*-------------------------------------------------------------*/

#include "Main.H"
#include "Port.H"

#include "Bargraph.H"
#include "Pulse_Count.H"


/* ------ Private function prototypes ------------------------ */
void PULSE_COUNT_Check_Below_Threshold(const tByte);

/* ------ khai báo biến dùng chung ---------------------- */
/* dữ liệu được hiển thị */
extern tBargraph Data_G;

/* ------ định nghĩa biến dùng chung -------------------------- */
/* chỉ set sau khi sườn xuống được phát hiện */
bit Falling_edge_G;

/* ------ định nghĩa biến riêng ------------------------ */
/* liên tục kiểm tra tín hiệu xung */
/* (chú ý: không thể có một dãy các bit...) */
static bit Test4, Test3, Test2, Test1, Test0;

static tByte Total_G = 0;
static tWord Calls_G = 0;

/* ------ hằng số riêng ---------------------------------- */

/* cho phép thay đổi mức logic mà phần cứng không thay đổi */
#define HI_LEVEL (0)
#define LO_LEVEL (1)
```

```
/*-------------------------------------------------------------*-

   PULSE_COUNT_Init()

   Hàm khởi động cho thư viện thiết bị ngắt.

-*-------------------------------------------------------------*/
void PULSE_COUNT_Init(void)
     {
     Sw_pin = 1; /* sử dụng chân này cho đầu vào */

     /* các dấu hiệu */
     Test4 = LO_LEVEL;
     Test3 = LO_LEVEL;
     Test2 = LO_LEVEL;
     Test1 = LO_LEVEL;
     Test0 = LO_LEVEL;
     }

/*-------------------------------------------------------------*-

   PULSE_COUNT_Check_Below_Threshold()

   Kiểm tra xem bộ đếm xung có thấp hơn giá trị ngưỡng không.
   Nếu đúng, hệ thống báo động sẽ kêu.

-*-------------------------------------------------------------*/
void PULSE_COUNT_Check_Below_Threshold(const tByte THRESHOLD)
     {
     if (Data_G < THRESHOLD)
          {
          Alarm_pin = 0;
          }
     else
          {
          Alarm_pin = 1;
          }
     }
```

```
/*-------------------------------------------------------------*-

   PULSE_COUNT_Update()

   Đây là hàm ngắt chính.

   Nó lên được gọi sau mỗi 30ms
   (để cho phép thời gian hãm đặc trưng là 20ms).

-*-------------------------------------------------------------*/
void PULSE_COUNT_Update(void)
     {
     /* xóa cờ timer */
     TF2 = 0;

     /* trộn các kết quả kiểm định */
     Test4 = Test3;
     Test3 = Test2;
     Test2 = Test1;
     Test1 = Test0;

     /* thu thập các kết quả mới nhất */
     Test0 = Sw_pin;

     /* kết quả        yêu cầu:
          Test4 ==     HI_LEVEL
          Test3 ==     HI_LEVEL
          Test1 ==     LO_LEVEL
          Test0 ==     LO_LEVEL */

     if ((Test4 == HI_LEVEL) &&
          (Test3 == HI_LEVEL) &&
          (Test1 == LO_LEVEL) &&
          (Test0 == LO_LEVEL))
          {
          /* sườn xuống được phát hiện */
          Falling_edge_G = 1;
          }
     else
          {
          /* mặc định */
          Falling_edge_G = 0;
          }
```

```
    /* tính trung bình 45 lần gọi tác vụ
         - số đếm lớn nhất trên một chu kì này là 9 xung
           if (++Calls_G < 45) */

    /* 450 được sử dụng ở đây mang mục đích kiểm tra (trong mô phỏng)
         [Bởi vì có một giới hạntốc độ nên bạn có thể mô phỏng bằng tay
         ...] */
    if (++Calls_G < 450)
         {
         Total_G += (int) Falling_edge_G;
         }
    else
         {
         /* cập nhật hiển thị */
         Data_G = Total_G; /* lớn nhất là 9 */
         Total_G = 0;
         Calls_G = 0;
         PULSE_COUNT_Check_Below_Threshold(3);
         BARGRAPH_Update();
         }
    }

/*-------------------------------------------------------------*-
   ---- END OF FILE ----------------------------------------
-*-------------------------------------------------------------*/
```

```
/*-------------------------------------------------------*-

    Bargraph.h (v1.00)

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

    - xem Bargraph.c để biết thêm chi tiết.

-*-------------------------------------------------------*/

#include "Main.h"

/* ------ khai báo kiểu dữ liệu chung ---------------------- */

typedef tByte tBargraph;

/* ------ khai báo hàm nguyên mẫu -------------------------- */

void BARGRAPH_Init(void);
void BARGRAPH_Update(void);

/* ------ khai báo hằng số --------------------------------- */

#define BARGRAPH_MAX (9)
#define BARGRAPH_MIN (0)

/*-------------------------------------------------------*-
   ---- END OF FILE ----------------------------------
-*-------------------------------------------------------*/
```

```
/*-------------------------------------------------------*-

    Bargraph.c (v1.00)

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

    Thư viện bargraph đơn giản.

-*-------------------------------------------------------*/

#include "Main.h"
#include "Port.h"

#include "Bargraph.h"

/* ------ khai báo biến chung ----------------------- */

/* dữ liệu được hiển thị */
tBargraph Data_G;

/* ------ các hằng riêng ---------------------------------- */

#define BARGRAPH_ON (1)
#define BARGRAPH_OFF (0)

/* ------ các biến riêng ---------------------------------- */

/* các biến lưu trữ ngưỡng này được sử dụng để cập nhật hiển thị */
static tBargraph M9_1_G;
static tBargraph M9_2_G;
static tBargraph M9_3_G;
static tBargraph M9_4_G;
static tBargraph M9_5_G;
static tBargraph M9_6_G;
static tBargraph M9_7_G;
static tBargraph M9_8_G;
```

```
*-------------------------------------------------------*-

   BARGRAPH_Init()

   Chuẩn bị cho hiển thị bargraph.

-*-------------------------------------------------------*/
void BARGRAPH_Init(void)
     {
     Pin0 = BARGRAPH_OFF;
     Pin1 = BARGRAPH_OFF;
     Pin2 = BARGRAPH_OFF;
     Pin3 = BARGRAPH_OFF;
     Pin4 = BARGRAPH_OFF;
     Pin5 = BARGRAPH_OFF;
     Pin6 = BARGRAPH_OFF;
     Pin7 = BARGRAPH_OFF;

     /* sử dụng một tỉ lệ tuyến tính để hiển thị dữ liệu
          Nhớ rằng: *9* trạng thái đầu ra có thể thực hiện
          - làm tất cả các phép toán một lần */
     M9_1_G = (BARGRAPH_MAX - BARGRAPH_MIN) / 9;
     M9_2_G = M9_1_G * 2;
     M9_3_G = M9_1_G * 3;
     M9_4_G = M9_1_G * 4;
     M9_5_G = M9_1_G * 5;
     M9_6_G = M9_1_G * 6;
     M9_7_G = M9_1_G * 7;
     M9_8_G = M9_1_G * 8;
     }
```

```
/*-------------------------------------------------------------*-

   BARGRAPH_Update()

   Cập nhật hiển thị bargraph.

-*-------------------------------------------------------------*/
void BARGRAPH_Update(void)
     {
     tBargraph Data = Data_G - BARGRAPH_MIN;

     Pin0  =  ((Data   >=  M9_1_G)  ==  BARGRAPH_ON);
     Pin1  =  ((Data   >=  M9_2_G)  ==  BARGRAPH_ON);
     Pin2  =  ((Data   >=  M9_3_G)  ==  BARGRAPH_ON);
     Pin3  =  ((Data   >=  M9_4_G)  ==  BARGRAPH_ON);
     Pin4  =  ((Data   >=  M9_5_G)  ==  BARGRAPH_ON);
     Pin5  =  ((Data   >=  M9_6_G)  ==  BARGRAPH_ON);
     Pin6  =  ((Data   >=  M9_7_G)  ==  BARGRAPH_ON);
     Pin7  =  ((Data   >=  M9_8_G)  ==  BARGRAPH_ON);
     }

/*-------------------------------------------------------------*-
   ---- END OF FILE ----------------------------------
-*-------------------------------------------------------------*/
```

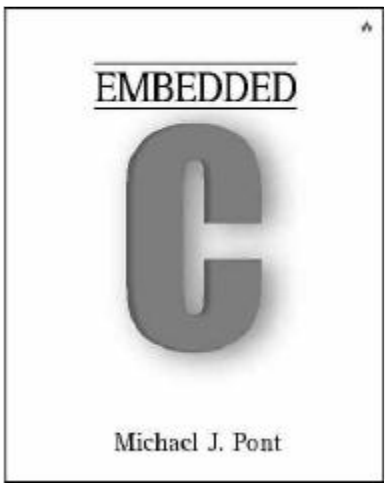
Kết luận

Hệ điều hành đơn giản (‘sEOS’) được giới thiệu trong chuyên đề này áp đặt một tải xử lý rất nhỏ, nhưng dù sao cũng rất mềm dẻo và hữu ích.

Bản chất đơn giản của sEOS cũng cung cấp các lợi ích khác.Điều đó có nghĩa là người phát triển có thể tự mình nhanh chóng cho một ‘hệ điều hành’ trên một con vi điều khiển.
Điều này cũng có nghĩa rằng người kĩ sư có thể nhanh chóng thích nghi các yêu cầu của của các ứng dụng đặc biệt.

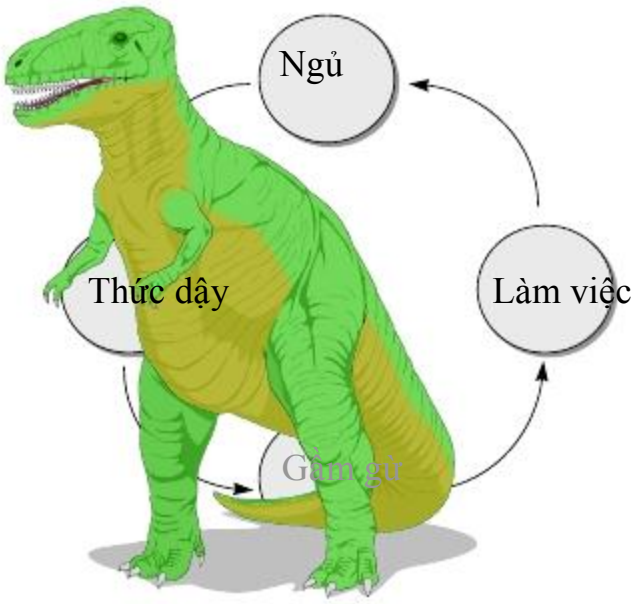
Có lẽ tác dụng không mong muốn nhất của kiểu ‘hệ điều hành đơn giản’ là không giống với một ‘hệ điều hành thời gian thực’ truyền thống – tự nó trở thành một phần của hệ thống, hơn là tạo ra một đoạn mã riêng biệt.

Chuẩn bị cho chuyên đề tiếp theo



Xin đọc **Chương 8**
Trước khi bắt đầu chuyên đề này

Chương 7: Hệ thống đa trạng thái và dãy hàm



Lời giới thiệu

Có hai loại hệ thống đa trạng thái:

Đa trạng thái (được định thời)
Trong hệ thống này, sự chuyển tiếp giữa các trạng thái **chỉ** phụ thuộc vào hành trình của thời gian.

Ví dụ như, hệ thống sẽ bắ đầu ở trạng thái A,thực thi nhiều lần hàm FunctionA(), trong 10s. Sau đó chuyển sang trạng thái B, thực thi hàm FunctionB() trong 5s.Sau đó chuyển đến trạng thái A va kết thúc quá trình.

Một hệ thống điều khiển đèn giao thông là một ví dụ như vậy.

Đa trạng thái (nhận tín hiệu vào & định thời)
Đó là một kiểu hệ thống khá phổ biến,ở đó sự chuyển trạng thái (và chế độ của từng trạng thái)của hệ thống sẽ được quết định bởi cả thời gian và các đầu vào của hệ thống.

Ví dụ, hệ thống chỉ chuyển từ trạng thái A sang trạng thái B nếu mốt đầu vào nào đó nhận được một tín hiệu trong khoảng thời gian X giây.

Hệ thống tự động lái thảo luận ở đầu chuyên đề có thể minh họa cho vấn đề này, cũng có thể ví dụ như hệ thống điều khiển cho một máy giặt, hay một hệ thống cảnh báo người lạ xâm nhập.

Để trọn vẹn, chúng ta sẽ tiếp tục đề cập thêm các khả năng:

Đa trạng thái (có đầu vào)

Đó là một kiểu hệ thống tương đối hiếm gặp, ở đây sự chuyển trạng thái (và chế độ của từng trạng thái) **chỉ** được quyết định bởi các đầu vào của hệ thống.

Ví dụ,hệ thống chỉ chuyển từ trạng thái A sang trạng thái B nếu có tín hiệu đầu vào.nó sẽ vẫn ở trạng thái A nếu chưa có tín hiệu vào.

Một hệ thống như thế không có khái niệm về thời gian, và vì thế không có cách nào tạm ngưng hoặc tương tự như thế. Chúng ta sẽ không đề cập đến các hệ thống như thế trong khóa học này.

Trong chuyên đề này, chúng ta sẽ tính toán xem làm cách nào để các kiểu kiến trúc đa trạng thái (Thời gian)và các kiểu kiến trúc đa trạng thái (đầu vào / Thời gian) có thể thi hành trong C.

Thi hành một hệ thống đa trạng thái (định thời gian)

Chúng ta có thể miêu các kiểu kiến trúc đa trạng thái,điều khiển thời gian như sau:

Hệ thống sẽ hoặt động ở hai (hoặc hơn 2 trạng thái).

Mỗi trạng thái có thể kèm theo một hay nhiều hơn lời gọi hàm.

Sự chuyển giao giữa các trạng thái sẽ được điều khiển bằng hành trình thời gian.

Sự chuyển giao giữa các trạng thái cũng có thể gồm nhiều lời gọi hàm.

Chú ý rằng, theo thứ tự giảm dần nhiệm vụ duy trì xảy ra tiếp theo, trạng thái hệ thống không nên có tên tùy ý, nhưng nên – nơi có thể– phản ánh trạng thái của một vật chất được quan sát bởi người sử dụng hoặc chính nhà phát triển.

Cũng xin chú ý rằng trạng thái hệ thống sẽ thường xuyên được biểu hiện bởi câu lệnh switch trong hệ điều hành ISR.

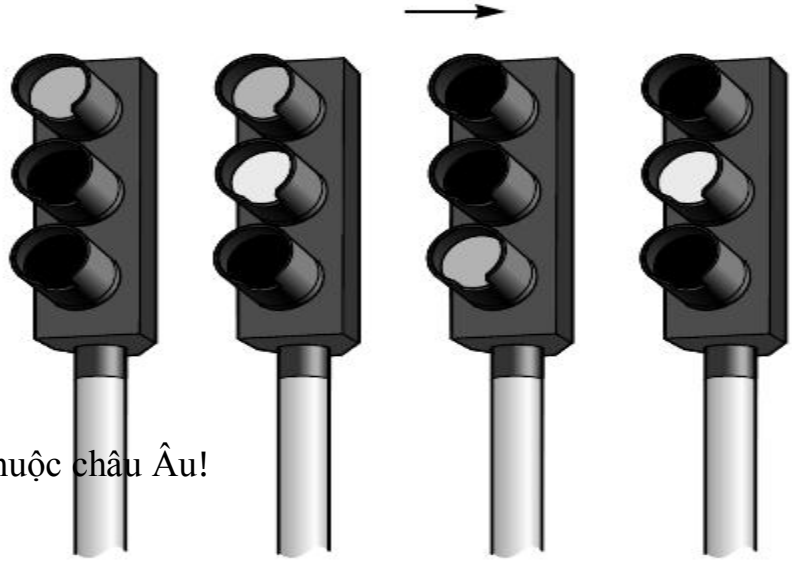
Ví dụ: dãy đèn giao thông

Thời gian

Đỏ

vàng

xanh



Chú ý:
Kiểu điều khiển này thuộc châu Âu!

Trong trường hợp này, các trạng thái khác nhau rất dễ dàng để nhận ra:

Đỏ

Đỏ-vàng

xanh

vàng

Trong mã này, chúng ta sẽ biểu diễn các trạng thái như sau:

```
/* các trạng thái có thể có của hệ thống */
typedef enum {RED, RED_AND_AMBER, GREEN, AMBER} eLight_State;
```

chúng ta sẽ lưu trữ các khoảng thời gian trong mỗi trạng thái như sau:

```
/* (tính theo giây) */
#define RED_DURATION 20
#define RED_AND_AMBER_DURATION 5
#define GREEN_DURATION 30
#define AMBER_DURATION 5
```

Trong trường hợp đơn giản, chúng ta không cần các lời gọi hàm ở (hoặc giữa) các trạng thái của hệ thống: các chế độ yêu cầu sẽ được thi hành trực tiếp thông qua điều khiển các chân (ở cuối hệ thống).

Ví dụ:

```
case RED:
        {
        Red_light = ON;
        Amber_light = OFF;
        Green_light = OFF;

    ...
```

```
/*------------------------------------------------------------*-

    Main.c (v1.00)
   ----------------------------------------------------
    Ví dụ về đèn giao thông.
-*------------------------------------------------------------*/

#include "Main.H"
#include "Port.H"
#include "Simple_EOS.H"

#include "T_Lights.H"
/* ------------------------------------------------------------ */
void main(void)
    {
    /* chuan bi chay day den giao thong */
    TRAFFIC_LIGHTS_Init(RED);

    /* cài đặt simple EOS (tick 50 ms) */
    sEOS_Init_Timer2(50);

    while(1) /* vòng lặp vô tận */
        {
        /* cho vào chế độ nghỉ để tiết kiệm năng lượng */
        sEOS_Go_To_Sleep();
        }
    }

/*------------------------------------------------------------*-
   ---- END OF FILE ------------------------------------
-*------------------------------------------------------------*/
```

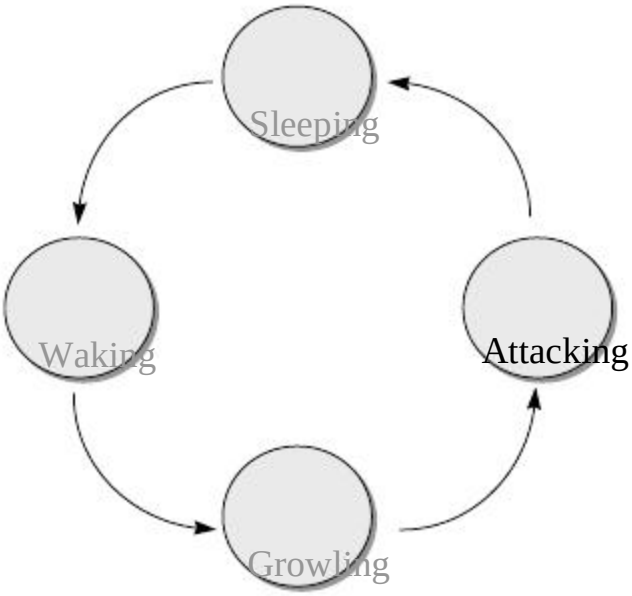
```
/*-------------------------------------------------------*-

     T_Lights.H (v1.00)
    ----------------------------------------------------
     - xem T_Lights.C để biết thêm chi tiết.
-*-------------------------------------------------------*/

#ifndef _T_LIGHTS_H
#define _T_LIGHTS_H

/* ------ khai báo dữ liệu chung ---------------------- */

/* các trạng thái có thể có của hệ thống */
typedef enum {RED, RED_AND_AMBER, GREEN, AMBER} eLight_State;

/* ------ các hàm nguyên mẫu chung -------------------------- */

void TRAFFIC_LIGHTS_Init(const eLight_State);
void TRAFFIC_LIGHTS_Update(void);

#endif

/*-------------------------------------------------------*-
   ---- END OF FILE ----------------------------------
-*-------------------------------------------------------*/
```

```
/*-------------------------------------------------------------*-

    T_lights.C (v1.00)

-*-------------------------------------------------------------*/

#include "Main.H"
#include "Port.H"

#include "T_lights.H"

/* ------ hằng số riêng ---------------------------------- */

/* dễ dàng thay đổi giá trị logic ở đây */
#define ON 0
#define OFF 1

/* các thời gian trong mỗi (4) trạng thái sáng của hệ thống
    (thời gian được tính bằng giây) */
#define RED_DURATION 20
#define RED_AND_AMBER_DURATION 5
#define GREEN_DURATION 30
#define AMBER_DURATION 5


/* ------ các biến riêng ---------------------------------- */

/* trạng thái của hệ thống */
static eLight_State Light_state_G;

/* thời gian trong các trạng thái đó */
static tLong Time_in_state;

/* được sử dụng bởi sEOS */
static tByte Call_count_G = 0;

/*-------------------------------------------------------------*-

   TRAFFIC_LIGHTS_Init()

   Chuẩn bị cho các hoạt động của đèn giao thông.

-*-------------------------------------------------------------*/
void TRAFFIC_LIGHTS_Init(const eLight_State START_STATE)
    {
    Light_state_G = START_STATE;        /* lựa chọn trạng thái đầu */
    }
```

```
/*-------------------------------------------------------------*-

   TRAFFIC_LIGHTS_Update()

   Phải được gọi một lần/một giây.

-*-------------------------------------------------------------*/
void TRAFFIC_LIGHTS_Update(void)
    {
    switch (Light_state_G)
        {
        case RED:
            {
            Red_light = ON;
            Amber_light = OFF;
            Green_light = OFF;

            if (++Time_in_state == RED_DURATION)
                {
                Light_state_G = RED_AND_AMBER;
                Time_in_state = 0;
                }

            break;
            }

        case RED_AND_AMBER:
            {
            Red_light = ON;
            Amber_light = ON;
            Green_light = OFF;

            if (++Time_in_state == RED_AND_AMBER_DURATION)
                {
                Light_state_G = GREEN;
                Time_in_state = 0;
                }

            break;
            }
```

```
        case GREEN:
            {
            Red_light = OFF;
            Amber_light = OFF;
            Green_light = ON;

            if (++Time_in_state == GREEN_DURATION)
                {
                Light_state_G = AMBER;
                Time_in_state = 0;
                }

            break;
            }

        case AMBER:
            {
            Red_light = OFF;
            Amber_light = ON;
            Green_light = OFF;

            if (++Time_in_state == AMBER_DURATION)
                {
                Light_state_G = RED;
                Time_in_state = 0;
                }

            break;
            }
        }
    }

/*-------------------------------------------------------------*-
   ---- END OF FILE ----------------------------------
-*-------------------------------------------------------------*/
```

Ví dụ: khủng long máy



Các trang thái hê thống

Ngủ:
Khủng long bất động ,nhưng hiển nhiên là vẫn 'thở'. Các tiếng ngáy phát ra không đều hay những chuyển động nhỏ trong thời gian này sẽ thêm phần thú vị cho khán giả.

Thức dậy:
Khủng long sẽ bắt đầu thức giấc,mí mắt bắt đầu hé mở , và bắt đầu thở mạnh hơn.

Gầm gừ:
Mắt đột nhiên mở và khủng long sẽ phát ra tiếng gầm lớn. tiếp theo là một số chuyển động và tiếp tục gầm.

Tấn công:
Các chuyển động nhanh ngẫu nhiên hướng về phía khán giả, gây ra các tiếng động (you should be able to hear this from the next floor in
the museum).

```
typedef enum {SLEEPING, WAKING, GROWLING, ATTACKING} eDinosaur_State;

/* thời gian của các trạng thái có thể có */
/* (thời gian tính bằng giây) */
#define SLEEPING_DURATION 255
#define WAKING_DURATION 60
#define GROWLING_DURATION 40
#define ATTACKING_DURATION 120
```

```
/*-------------------------------------------------------*-
    Dinosaur.C (v1.00)
   ----------------------------------------------------
    Trình diễn kiến trúc đa trạng thái (Timed):
    Hệ thống điều khiển khủng long.

  -*-------------------------------------------------------*/

#include "Main.h"
#include "Port.h"

#include "Dinosaur.h"

/* ------ khai báo biến riêng ---------------------- */
/* các trạng thái có thể có của hệ thống */
typedef
enum {SLEEPING, WAKING, GROWLING, ATTACKING} eDinosaur_State;

/* ------ hàm nguyên mẫu riêng ------------------------ */
void DINOSAUR_Perform_Sleep_Movements(void);
void DINOSAUR_Perform_Waking_Movements(void);
void DINOSAUR_Growl(void);
void DINOSAUR_Perform_Attack_Movements(void);

/* ------ các hằng số riêng ---------------------------------- */
/* thời gian trong mỗi trạng thái
    (thời gian tính bằng giây) */
#define SLEEPING_DURATION 255
#define WAKING_DURATION 60
#define GROWLING_DURATION 40
#define ATTACKING_DURATION 120

/* ------ biến riêng---------------------------------- */
/* trạng thái hiện tại của hệ thống */
static eDinosaur_State Dinosaur_state_G;

/* thời gian trong trạng thái đó */
static tByte Time_in_state_G;

/* được sử dụng bởi sEOS */
static tByte Call_count_G = 0;
```

```
/*---------------------------------------------------------*-
   DINOSAUR_Init()
-*---------------------------------------------------------*/
void DINOSAUR_Init(void)
     {
     /* trạng thái ban đầu */
     Dinosaur_state_G = SLEEPING;
     }

/*---------------------------------------------------------*-

   DINOSAUR_Update()

   Phải được ghi theo lịch trình 1 lần/giây (từ sEOS ISR).

-*---------------------------------------------------------*/
void DINOSAUR_Update(void)
     {
     switch (Dinosaur_state_G)
          {
          case SLEEPING:
               {
               /* gọi hàm có lien quan */
               DINOSAUR_Perform_Sleep_Movements();

               if (++Time_in_state_G == SLEEPING_DURATION)
                    {
                    Dinosaur_state_G = WAKING;
                    Time_in_state_G = 0;
                    }

               break;
               }

          case WAKING:
               {
               DINOSAUR_Perform_Waking_Movements();

               if (++Time_in_state_G == WAKING_DURATION)
                    {
                    Dinosaur_state_G = GROWLING;
                    Time_in_state_G = 0;
                    }

               break;
               }
```

```
        case GROWLING:
            {
             /* gọi hàm có liên quan */
             DINOSAUR_Growl();

            if (++Time_in_state_G == GROWLING_DURATION)
                {
                Dinosaur_state_G = ATTACKING;
                Time_in_state_G = 0;
                }

            break;
            }

        case ATTACKING:
            {
             /* gọi hàm có liên quan */
             DINOSAUR_Perform_Attack_Movements();

            if (++Time_in_state_G == ATTACKING_DURATION)
                {
                Dinosaur_state_G = SLEEPING;
                Time_in_state_G = 0;
                }

            break;
            }
        }
}
```

```
/*-------------------------------------------------------------*/
void DINOSAUR_Perform_Sleep_Movements(void)
      {
      /* chi la gioi thieu... */
      P1 = (tByte) Dinosaur_state_G;
      P2 = Time_in_state_G;
      }

/*-------------------------------------------------------------*/
void DINOSAUR_Perform_Waking_Movements(void)
      {
      /* chỉ là giới thiệu... */
      P1 = (tByte) Dinosaur_state_G;
      P2 = Time_in_state_G;
      }

/*-------------------------------------------------------------*/
void DINOSAUR_Growl(void)
      {
      /* chỉ là giới thiệu... */
      P1 = (tByte) Dinosaur_state_G;
      P2 = Time_in_state_G;
      }

/*-------------------------------------------------------------*/
void DINOSAUR_Perform_Attack_Movements(void)
      {
      /* chỉ là giới thiệu... */
      P1 = (tByte) Dinosaur_state_G;
      P2 = Time_in_state_G;
      }

/*-------------------------------------------------------------*-
   ---- END OF FILE ----------------------------------
-*-------------------------------------------------------------*/
```

Thi hành một hệ thống đa trạng thái (đầu vào/định thời(Input/Timer))

Hệ thống sẽ hoạt động ở 2 hay nhiều trạng thái.

Mỗi trạng thái có thể có một hay nhiều hơn lời gọi hàm.

Sự thay đổi giữa các trạng thái có thể được điều khiển bởi hành trình thời gian, bởi đầu vào hệ thống, hoặc bởi cả hai yếu tố trên.

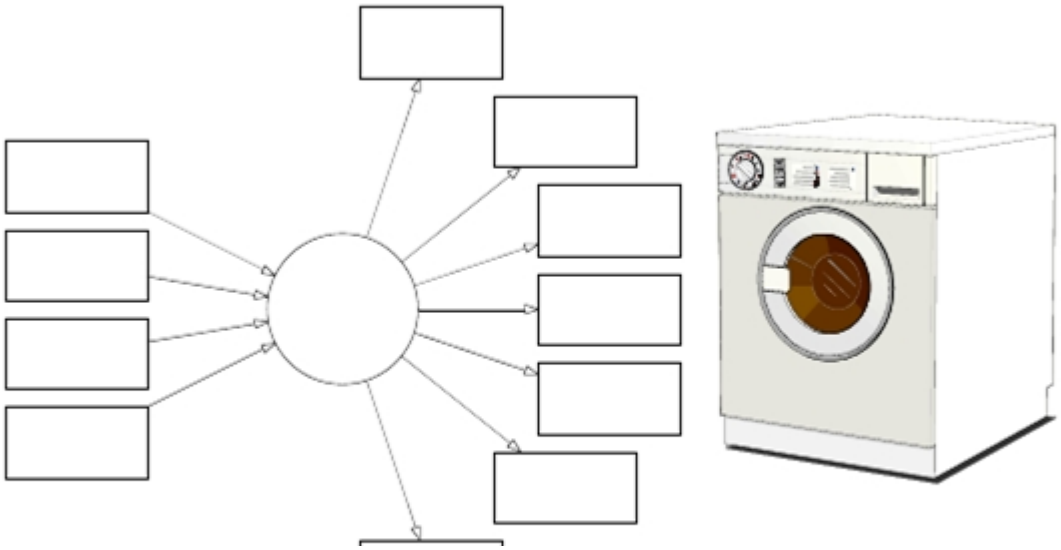
Sự thay đổi giữa các trạng thái cũng có thể chứa nhiều lời gọi hàm.

Thi hành trạng thái nghỉ

Hãy xem xét các yêu cầu của hệ thống dưới đây(không chính thống):

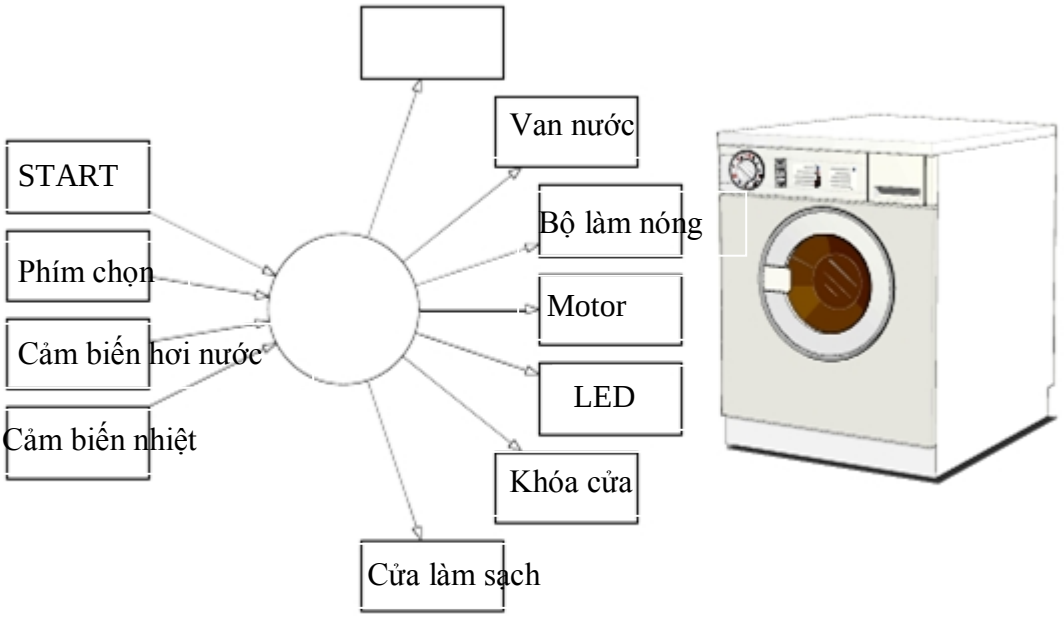
Bơm sẽ chạy trong 10 giây.Nếu trong khoảng thời gian này, không có chất lỏng chảy ra khỏi bể chứa thì bơm sẽ đóng lại và có tiếng cảnh báo'mực nước thấp'.nếu có chất lỏng, bơm sẽ chạy thêm 45s nữa hoặc sẽ chảy đến khi nào cảm biến mực nước cao được kích hoạt (whichever is first).

Sau khi cửa trước được mở, mật khẩu sẽ được nhập vào bảng điều khiển trong vòng 30 giây nếu không sẽ có tiếng báo động



động của van thì kết
ế độ bằng tay.

ạng thái khác.



Dưới đây là các miêu tả ngắn gọn các hoạt động mong muốn của hệ thống:

1. Ngươi dùng chọn một chế độ giặt (ví dụ ‘len’, ‘Cotton’) trên phím chọn.

2. Người dùng bấm nút ‘Start’.

3. Khóa cửa đã khóa.

4. Van nước đã mở, cho phép chảy vào khu vực giặt.

5. Nếu chương trình bao gồm cả làm sạch thì mở cửa làm sạch.khi làm sạch xong, cửa làm sạch được đóng lại.

6. Khi ‘mức nước đầy’ được phát hiện thì van nước đóng lại.

7. Nếu chương trình giặt bao gồm chế độ nước ấm,thì bộ làm nóng nước được bật . khi nước đạt tới nhiệt độ chính xác thì tắt bộ làm nóng nước.

8. Mô tơ của máy được bật để quay trống(nơi để giặt).sau đó motor sẽ trải qua một chuỗi các chuyển động,bao gồm cả tiến và lùi(ở các cấp tốc độ khác nhau) để giặt quần áo. (các chuyển động chính xác của motor được quết định bởi người sử dụng đã chọn chế đồ như thế nào) .Ở cuối quá tình giặt, motor sẽ dừng.

9. Bơm được bật để xả nước, khi chỗ giặt(drum) chảy ra hết thì bơm tắt

Kiểu kiến trúc Input / Timed mà chúng ta thảo luận ở đây không chi duy nhất cho 'hàng trắng(white goods)' (ví dụ như máy giặt).

Ví dụ, một dãy các sự kiện được sử dụng để nâng bánh đỡ(landing-gear) trong máy bay chở hàng sẽ được điều khiển một cách tương tự.Trong trường hợp này, các kiểm tra quan trọng (ví dụ 'WoW' - 'Weight on Wheels') sẽ được sử dụng để xác định xem là máy bay đang ở dưới đất hay trên trời:các kiểm tra này sẽ hoàn thành trước khi công việc bắt đầu.

Hồi tiếp từ các cửa khác nhau và các cảm biến 'bánh đỡ' sẽ dùng để đảm bảo rằng từng bước của hoạt động chính xác.

```
/*-------------------------------------------------------*-

    Washer.C (v1.01)

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

    Kết cấu đa trạng thái cho bộ điều khiển máy giặt.

-*-------------------------------------------------------*/

#include "Main.H"
#include "Port.H"

#include "Washer.H"

/* ------ khai báo dữ liệu riêng ---------------------- */

/* các trạng thái có thể */
typedef enum {INIT, START, FILL_DRUM, HEAT_WATER,
                    WASH_01, WASH_02, ERROR} eSystem_state;

/* ------ các hàm nguyên mẫu riêng ------------------------ */

tByte   WASHER_Read_Selector_Dial(void);
bit     WASHER_Read_Start_Switch(void);
bit     WASHER_Read_Water_Level(void);
bit     WASHER_Read_Water_Temperature(void);

void    WASHER_Control_Detergent_Hatch(bit);
void    WASHER_Control_Door_Lock(bit);
void    WASHER_Control_Motor(bit);
void    WASHER_Control_Pump(bit);
void    WASHER_Control_Water_Heater(bit);
void    WASHER_Control_Water_Valve(bit);

/* ------ hằng số riêng ---------------------------------- */

#define OFF 0
#define ON 1

#define MAX_FILL_DURATION (tLong) 1000
#define MAX_WATER_HEAT_DURATION (tLong) 1000

#define WASH_01_DURATION 30000
```

```
/* ------ các biến riêng --------------------------------- */

static eSystem_state System_state_G;

static tWord Time_in_state_G;

static tByte Program_G;

/* 10 chương trình khác nhau được trợ giúp
     Mỗi loại có thể (hoặc không)dùng để làm sạch */
static tByte Detergent_G[10] = {1,1,1,0,0,1,0,1,1,0};

/* mỗi loại có thể (hoặc không)dùng để làm nóng nước */
static tByte Hot_Water_G[10] = {1,1,1,0,0,1,0,1,1,0};

/* ------------------------------------------------------------ */
void WASHER_Init(void)
     {
     System_state_G = INIT;
     }
```

```
/* ------------------------------------------------------------ */
void WASHER_Update(void)
    {
    /* gọi ra  lần/1 giây */
    switch (System_state_G)
        {
        case INIT:
            {
            /* chỉ để giới thiệu */
            Debug_port = (tByte) System_state_G;

            /* cài đặt trạng thái đầu */
            /* Motor tắt */
            WASHER_Control_Motor(OFF);

            /* bơm tắt */
            WASHER_Control_Pump(OFF);

            /* máy sưởi tắt */
            WASHER_Control_Water_Heater(OFF);

            /* van được đóng */
            WASHER_Control_Water_Valve(OFF);

            /* chờ (vô hạn định) đến khi nút START được bấm */
            if (WASHER_Read_Start_Switch() != 1)
                {
                return;
                }

            /* bắt đầu bấm nút
                -> đọc từ bàn phím chọn */
            Program_G = WASHER_Read_Selector_Dial();

             /* thay đổi trạng thái */
            System_state_G = START;
             break;
            }
```

```
case START:
    {
    /* chỉ để mô phỏng */
    Debug_port = (tByte) System_state_G;

    /* đóng cửa */
    WASHER_Control_Door_Lock(ON);

    /* bắt đầu cho đầy drum */
    WASHER_Control_Water_Valve(ON);

    /* tung ra chế độ làm sạch (nếu bình thường) */
    if (Detergent_G[Program_G] == 1)
        {
        WASHER_Control_Detergent_Hatch(ON);
        }

    /* sẵn sàng tới trạng thái tiếp theo */
    System_state_G = FILL_DRUM;
    Time_in_state_G = 0;

    break;
    }
```

```
case FILL_DRUM:
    {
    /* chỉ có tính chất giới thiệu */
    Debug_port = (tByte) System_state_G;

    /* ở lại trạng thái này đến khi drum đầy
        CHÚ Ý: thời gian nghỉ bao gồm cả ở đây */
    if (++Time_in_state_G >= MAX_FILL_DURATION)
        {
        /* sẽ làm đầy drum... */
        System_state_G = ERROR;
        }

    /* kiểm tra mực nước*/
    if (WASHER_Read_Water_Level() == 1)
        {
        /* Drum đầy */

        /* chương trình có yêu cầu làm nóng nước không? */
        if (Hot_Water_G[Program_G] == 1)
            {
            WASHER_Control_Water_Heater(ON);

            /* sẵn sàng chuyển trạng thái */
            System_state_G = HEAT_WATER;
            Time_in_state_G = 0;
            }
        else
            {
            /* chỉ sử dụng nước lạnh */
            /* sẵn sàng tới trạng thái tiếp theo */
            System_state_G = WASH_01;
            Time_in_state_G = 0;
            }
        }
    break;
    }
```

```
case HEAT_WATER:
    {
    /* chỉ để giới thiệu */
    Debug_port = (tByte) System_state_G;

    /* giữ nguyên trạng thái đến khi nước nóng
        CHÚ Ý: thời gian nghỉ bao gồm cả ở đây */
    if (++Time_in_state_G >= MAX_WATER_HEAT_DURATION)
        {
        /* nước sẽ được làm ấm... */
        System_state_G = ERROR;
        }

    /* kiểm tran nhiệt độ của nước */
    if (WASHER_Read_Water_Temperature() == 1)
        {
        /* nước ở mức nhiệt yêu cầu */
        /* sẵn sàng tới trạng thái tiếp theo */
        System_state_G = WASH_01;
        Time_in_state_G = 0;
        }

    break;
    }
```

```
        case WASH_01:
            {
            /* chỉ để minh họa */
            Debug_port = (tByte) System_state_G;

            /* tất cả chương trình giặt đều chứa WASH_01
                Drum được quay chậm để đảm bảo quần áo được ướt hoàn toàn */
            WASHER_Control_Motor(ON);

            if (++Time_in_state_G >= WASH_01_DURATION)
                {
                System_state_G = WASH_02;
                Time_in_state_G = 0;
                }

            break;
            }

        /* các bước giặt còn lại bị bỏ qua ở đây ... */

        case WASH_02:
            {
            /* chỉ để minh họa */
            Debug_port = (tByte) System_state_G;

            break;
            }

        case ERROR:
            {
            /* chỉ để minh họa */
            Debug_port = (tByte) System_state_G;

            break;
            }
        }
}
```

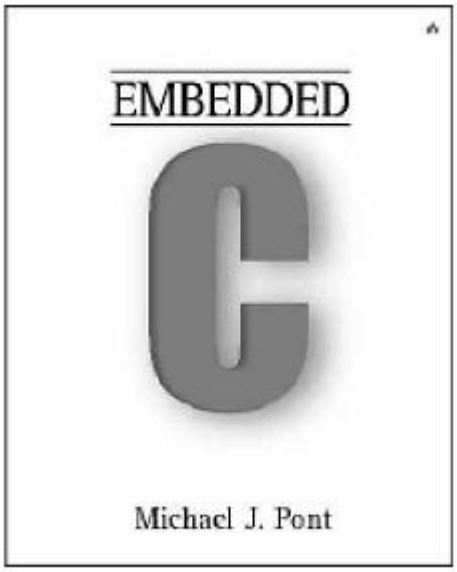
```
/* ------------------------------------------------------------ */
tByte WASHER_Read_Selector_Dial(void)
    {
    /* code của người sử dụng ở đây */

    return 0;
    }

/* ------------------------------------------------------------ */
bit WASHER_Read_Start_Switch(void)
    {
    /* đơn giản để giới thiệu ... */

    if (Start_pin == 0)
        {
        /* nút bắt đầu bấm */
        return 1;
        }
    else
        {
        return 0;
        }
    }

/* ------------------------------------------------------------ */
bit WASHER_Read_Water_Level(void)
    {
    /* User code here... */

    return 1;
    }

/* ------------------------------------------------------------ */
bit WASHER_Read_Water_Temperature(void)
    {
    /* code của người sử dụng ở đây... */

    return 1;
    }

/* ------------------------------------------------------------ */
void WASHER_Control_Detergent_Hatch(bit State)
    {
    bit Tmp = State;
    /* code của người sử dụng ở đây... */
    }
```

Kết luận

Chuyên đề này thảo luận sự thi hành của các hệ thống đa trạng thái (timed và input/timed), được sử dụng để kết nối vơi một hệ điều hành, chuyên đề này được trình bày trong cuốn "Embedded C"

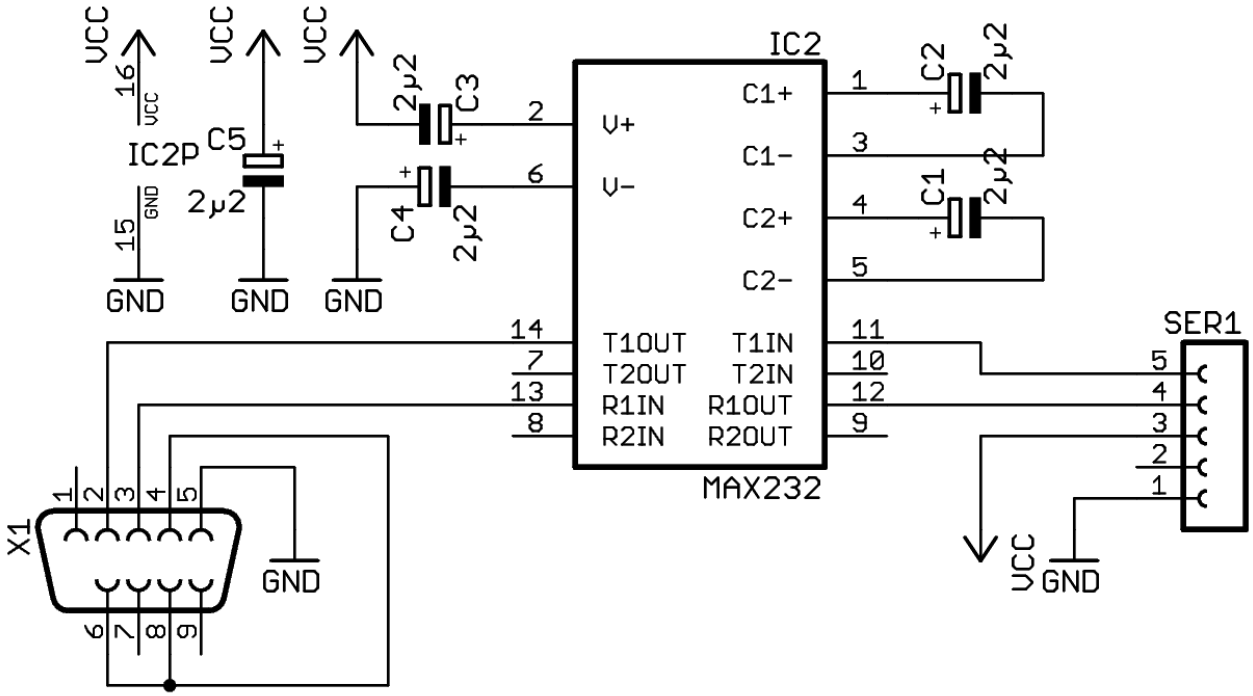
Chương 7, các kiểu kiến trúc mềm dẻo này sẽ được sử dụng rộng rãi trong các hệ thống nhúng.

Chuẩn bị cho chuyên đề tiếp theo



Xin đọc **Chương 9**
Trước khi bắt đầu chuyên đề tiếp theo

Chương 8: Sử dụng giao diện nối tiếp



Giới thiệu tổng quát về chuyên đề

Chuyên đề này sẽ:

Thảo luận về kết nối dữ liệu chuẩnDRS-232

Xem xét việc sử dụng RS-232 để truyền dữ liệu(và nhận dữ liệu)từ các máy tính(và các thiết bị tương tự) như thế nào.

Điều này hữu ích:

Trong các ứng dụng thu nhận dữ liệu.

Trong các ứng dụng điều khiển(gửi đến bộ điều khiển các tham số).

Cho sự chỉnh lý nói chung.

'RS-232' là gì?

Năm 1997, hiệp hội công nghiệp viễn thông(Telecommunications Industry Association) chính thức tung ra thị trường một giao thức truyền thông nối tiếp với tên TIA-232 phiên bản F, đã trở nên nổi tiếng hơn 'RS-232' (xuất hiện lần đầu tiên vào những năm 1960 với tên 'Recommended Standard'). Các chuẩn tương tự (V.28) cũng được công bố bởi International Telecommunications Union (ITU) và bơi CCITT (The Consultative Committee International Telegraph and Telephone).

Chuẩn 'RS-232' bao gồm các khía cạnh sau:

Giao thức này được sử dụng cho việc truyền dữ liệu.

Điện áp được sử dụng trên các đường truyền tín hiệu.

Các đầu nối được sử dụng để kết nối các thiết bị với nhau.

Nói chung,chuẩn này được sử dụng nhiều và rộng rãi, tốc độ truyền dữ liệu nằm trong khoảng từ 115 đến 330 kbits /s (115 / 330 k baud). Khoảng cách truyền dữ liệu là 15 m hoặc hơn.

Chú ý rằng RS-232 là một chuẩn kết nối cùng mức (peer-to-peer).

Giao thức RS-232 cơ bản

RS-232 là một giao thức hướng kí tự. Đó là, gửi một khối 8-bit dữ liệu.Để truyền phát một byte dữ liệu trên kết nối RS-232 ,thông thường chúng ta mã hóa thông tin như sau:

Gửi một bit ‘Start’.

Gửi dữ liệu (8 bits).

Gửi một bit ‘Stop’ (hay nhiều hơn).

Chú ý: UART sẽ chịu trách nhiệm về những chi tiết này!

Truyền dữ liệu không đồng bộ và tốc độ baud

RS-232 sử dụng một giao thức không đồng bộ.

Cuối của cả hai kết nối đều có một đồng hồ nội,
chạy ở cùng một tốc độ. Dữ liệu (trong trường hợp của RS-232,
bit 'Start') sau đó được sử dụng để đồng bộ các đồng hồ (nếu cần thiết) để đảm dữ liệu được truyền thành công.

RS-232 thường hoạt ở một giải(giới hạn) tốc độ baud.

Tiêu biểu là: 75, 110, 300, 1200, 2400, 4800, 9600,
14400, 19200, 28800, 33600, 56000, 115000 và
330000 (hiếm khi).

9600 là lựa chọn rất an toàn vì nó đươc hỗ trợ rất rộng rãi.

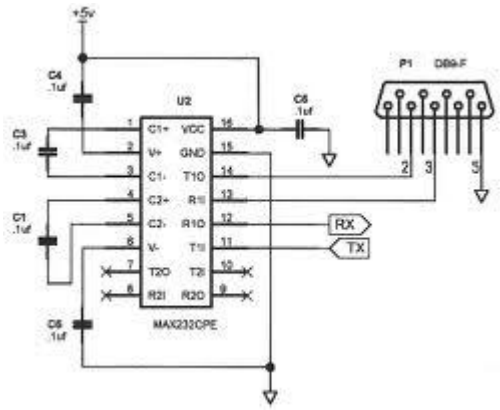
RS-232 các mức điện áp

Các mức ngưỡng được sử dụng để nhận là +3 V và -3 V and the lines are inverted.

Điệ áp lớn nhất cho phép là +/- 15V.

Chú ý rằng các mức điện áp này không thể kết nối trực tiếp với các chân của vi điều khiển(vài kiểu giao diện phần cứng sẽ được yêu cầu).

Ví dụ, Max232 và Max233 được sử dụng rất phổ biến và rông rãi ở các dòng chip điều khiển.



Sử dụng Max 233(như RS-232) để truyền nhận.

Cấu trúc phần mềm

Giả sử chúng ta muốn truyền dữ liệu tới một máy PC với tốc độ baud tiêu chuẩn là 9600(nghĩa là 9600 bit trên một giây). Truyền từng byte dữ liệu, tính cả bit start và bit stop là 10 bit thông tin (giả sử rằng một bit stop đơn được sử dụng). Theo một kết quả thì mỗi byte truyền đi mất xấp xỉ 1ms.

Ví du: Giả sử chúng ta muốn gửi thông tin sau tới PC:

Current core temperature is 36.678 degrees

…thì việc gửi 42 kí tự này sẽ mất hơn 40 ms để hoàn thành .Điều này thường thường là một khoảng thời gian dài, không thể chấp nhận.

Cách rõ ràng để giải quyết vần đề này là tăng tốc độ baud. Tuy nhiên cách này rất ít khi được sử dụng (và nó không thực sự giải quyết được vấn đề trên).

Một cách giải quết tốt hơn là ghi tất dữ liệu trên một vùng đệm của vi điều Khiển.Nội dung của vùng đệm này sau đó sẽ được gửi tới vi điều khiển (luôn là một bit một lần) sử dụng một tác vụ đều đặn, định kì.

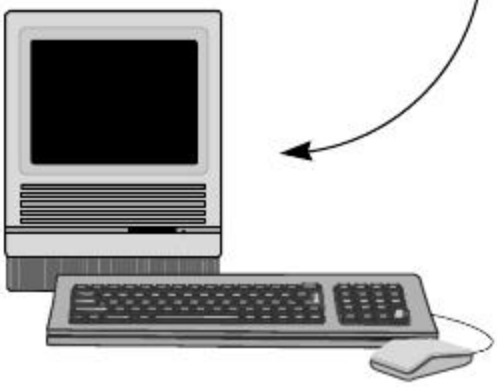
Tổng quan

Current cor e temperature
is 36. 678 degrees

Tất cả kí tự được ghi ngay lập tức được ghi vào bộ đệm

(hoạt động rất nhanh)

Bộ đệm(buffer)



Trình lập lịch sẽ gửi một
kí tự tới PC sau mỗi 10 ms
(ví dụ)

Sử dụng U(S)ART trên chip cho kết nối RS-232

UART là song công hoàn toàn,nghĩa là nó có thể truyền và nhận đồng thời một lúc.

Nó cũng có thể đệm lúc thu, nghĩa là nó có thể thu và giữ một byte trong khi byte thứ hai được nhận

Port nối tiếp có thể thực hiện 4 chế độ (một chế độ đồng bộ và ba chế độ không đồng bộ).

Trước tiên chúng ta xét chế độ 1.

Trong chế độ này, 10 bit được truyền đi(thông qua chân TxD) hay nhận (thông qua chân RxD): một bit start (0), 8 data bits và một bit stop (1).

Các thanh ghi port nối tiếp

Bộ điều khiển port nối tiếp và bộ ghi trạng thái chứa trong thanh ghi SCON. Thanh ghi này chứa các bit chọn chế độ (và các bit ngắt,
TI và RI: không được sử dụng ở đây).

SBUF là một thanh ghi đệm truyền nhận ở cổng nối tiếp.

Ghi lên SBUF để nạp thanh ghi truyền nhận và bắt đầu truyền nhận.

```
SBUF = 0x0D;      /* Output CR */
```

Đọc SBUF bằng cách truy cập vào thanh ghi nhận riêng rẽ:.

```
/* doc du lieu tu UART                    */
Data = SBUF;
```

Tạo ra tốc độ baud

Chúng ta chủ yếu đề cập cách sử dụng port nối tiếp ở chế độ.

Trong chế độ này, tốc độ baud được xác định tốc độ tràn của timer 1 hay Timer 2.

Chúng ta sẽ tập trung vào Timer một để sinh ra tốc độ baud.

Tốc độ baud được sinh ra bởi tốc độ tràn của Timer 1 và giá trị của SMOD như sau:

$$\text{Baud rate (Mode 1)} = \frac{2^{SMOD} \times Frequency_{oscillator}}{32 \times Instructions_{cycle} \times (256 - TH1)}$$

Trong đó:

SMOD	Là tốc bit 'độ baud kép' trên thanh ghi PCON
Frequency oscillator	Là tần số dao động
Instructionscycle	Là số lượng quy trình máy trên vòng lặp (ví dụ: 12,6)
*TH*1	là giá trị nạp lại cho Timer 1

Chú ý rằng timer này chế độ 8 bit tự động nạp lại giá trị và sự tạo thành ngắt đó nên được vô hiệu.

Tại sao lại sử dụng thạch anh 11.0592 MHz?

Rất quan trọng khi hiểu rằng không thể tạo ra các tốc độ baud chuẩn (ví dụ: 9600) sử dụng Timer 1 (hay Timer 2),trừ phi bạn sử dụng bộ dao động thạch anh 11.0592 .

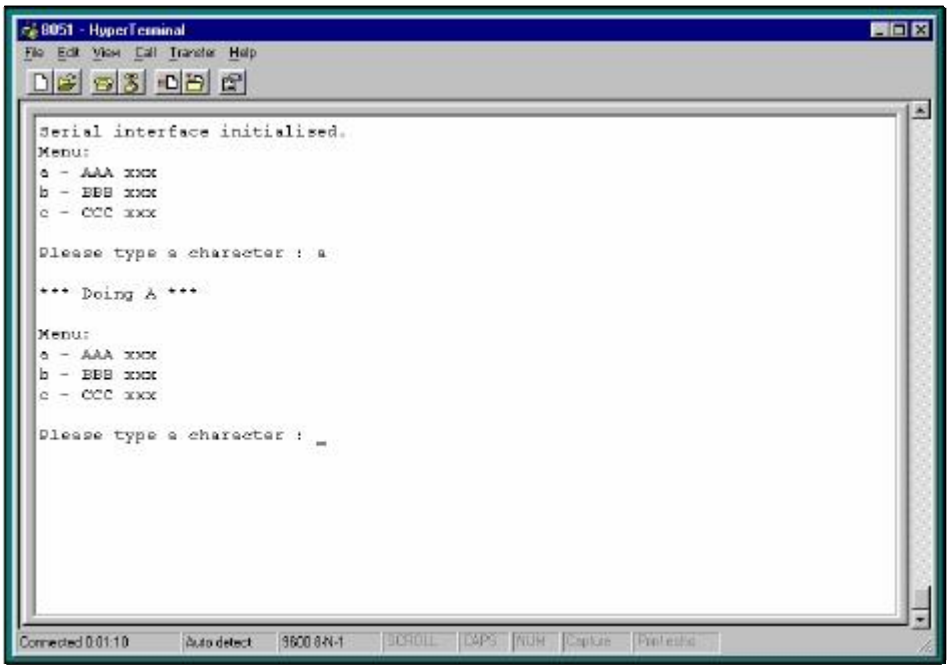
Ghi nhớ: đó là một giao thức không động bộ, và dựa vào sự hoạt động chính xác trên cả hai kết nối đang làm việc với cùng tốc độ baud. Trên thực tế, bạn có thể làm việc với một tỗ độ baud khác nhau ở cuối cả hai kết **tới 5%**, nhưng không được lớn hơn.

Mặc dù sai số, nhưng đó luôn luôn là nguyên tắc tốt để có được tốc độ baud xấp xỉ giá trị chuẩn bởi vì, **trong phạm vi này**, có thể có những biến đổi giữa các bộ dao động trong PC và hệ thống nhúng.

Cũng chú ý rằng,sử dụng các dạng dao động thạch anh (hơn thiết bị cộng hưởng bằng gốm) là rất cần thiết khi kết nối với các đường dẫn nối tiếp , không đồng bộ (như RS-232, RS-485, hay CAN): bộ dao động gốm không đủ ổn định cho mục đích này.

Phần mềm trên PC

Nếu máy tính của bạn đang chạy hệ điều hành Windows (95, 98, NT, 2000),thì một lựa chọn đơn giản mà hiệu quả là ứng dụng‘Hyperterminal’.



Về lệnh printf()?

Thường thường, chúng tôi khuyến cáo không nên sử dụng hàm thư viện chuẩn "printf()", vì:

Hàm này gửi dữ liệu ngay lập tức tới UART. Theo một kết quả, thời gian truyền nhận quá lâu dẫn đến sử dụng không an toàn (trong một ứng dụng lập trình có tính chất hợp tác), và

hầu hết sự thi hành của hàm printf() không liên quan chặt chẽ với thời gian nghỉ, vì vậy việc sử dụng hàm này có thể dẫn đến 'treo' nếu có lỗi xảy ra.

RS-232 và 8051: Bao gồm cả điểm mạnh và điểm yếu

Sự hỗ trợ RS-232 là một phần của 8051: các ứng dụng dựa vào RS-232 rất linh động.

RS-232 rất thường gặp: mỗi PC đều có một hoặc nhiều hơn cổng kết nối RS-232.

Dữ liệu có thể truyền xa đến 30 m (100 ft) (đối với các chip truyền nhận hiện đại).

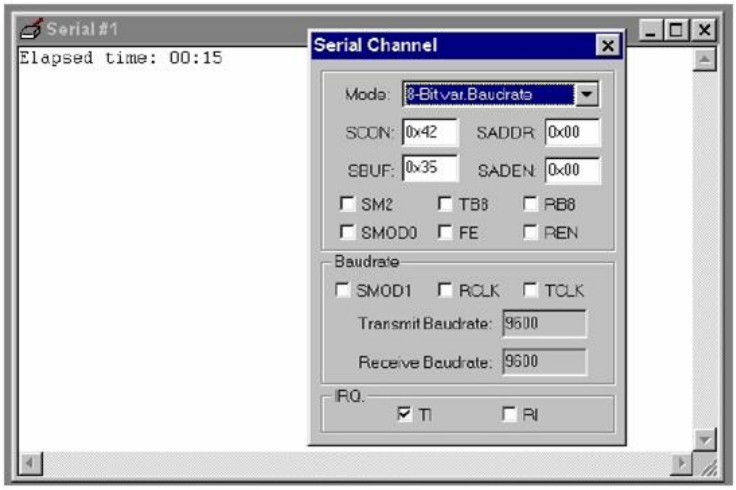
Bởi vì có phần cứng hỗ trợ nên RS-232 có mức đột tải thấp.

NHƯNG:

RS-232 là một giao thức bình đẳng (ví dụ: không giống với RS-485): bạn Chỉ có thể kết nối trực tiếp (đồng thời) một vi điều khiển với một PC.

RS-232 có rất ít (hoặc không có) phần kiểm tra lỗi ở các mức phần cứng (không giống với CAN): nếu bạn muốn chắc chắn rằng dữ liệu nhận ở máy tính là hợp lệ thì bạn cần tiến hành kiểm tra bằng phần mềm.

Ví dụ: Hiển thị thời gian trên một máy tính



```
/*------------------------------------------------------------*-

    Main.c (v1.00)

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

    RS-232 (thời gian trôi qua) ví dụ - sEOS.

-*------------------------------------------------------------*/

#include "Main.H"
#include "Port.H"
#include "Simple_EOS.H"

#include "PC_O_T1.h"
#include "Elap_232.h"

/* ------------------------------------------------------------ */

void main(void)
    {
    /* cài đặt tốc độ baud là 9600 */
    PC_LINK_O_Init_T1(9600);

    /* chuẩn bị cho hệ thống đo thời gian chạy */
    Elapsed_Time_RS232_Init();

    /* cài đặt simple EOS (tick 5ms) */
    sEOS_Init_Timer2(5);

    while(1) /* vòng lặp vô tận */
        {
        sEOS_Go_To_Sleep();       /* đưa vào chế độ nghỉ để tiết kiệm năng lượng */
        }
    }

/*------------------------------------------------------------*-
   ---- END OF FILE ----------------------------------
-*------------------------------------------------------------*/
```

```
/*-------------------------------------------------------------*-

    Elap_232.C (v1.00)

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

    Hàm thư viện đơn giản cho việc theo dõi thời gian trôi qua
    Phiên bản giơi thiệu dùng để hiển thị thời gian trên màn hình máy
    tính qua kết nối RS232.

-*-------------------------------------------------------------*/

#include "Main.h"
#include "Elap_232.h"
#include "PC_O.h"

/* ------ định nghĩa các biến chung ----------------------- */

tByte Hou_G;
tByte Min_G;
tByte Sec_G;

/* ------ khi báo các biến chung ----------------------- */

/* xem Char_Map.c */
extern const char code CHAR_MAP_G[10];

/*-------------------------------------------------------------*-
   Elapsed_Time_RS232_Init()

   Hàm khởi đầu cho thư viện hiển thị thời gian trên máy tính qua kết nối RS-232.

-*-------------------------------------------------------------*/
void Elapsed_Time_RS232_Init(void)
    {
    Hou_G = 0;
    Min_G = 0;
    Sec_G = 0;
    }
```

```
/*-------------------------------------------------------*/

void Elapsed_Time_RS232_Update(void)
    {
    char Time_Str[30] = "\rElapsed time:                    ";

    if (++Sec_G == 60)
        {
        Sec_G = 0;

        if (++Min_G == 60)
            {
            Min_G = 0;

            if (++Hou_G == 24)
                {
                Hou_G = 0;
                }
            }
        }

    Time_Str[15] = CHAR_MAP_G[Hou_G / 10];
    Time_Str[16] = CHAR_MAP_G[Hou_G % 10];

    Time_Str[18] = CHAR_MAP_G[Min_G / 10];
    Time_Str[19] = CHAR_MAP_G[Min_G % 10];

    Time_Str[21] = CHAR_MAP_G[Sec_G / 10];
    Time_Str[22] = CHAR_MAP_G[Sec_G % 10];

    /* chúng ta sự dụng dữ liệu "giây" để bật hoạc tắt dấu hai chấm
        (giữa giờ và phút) */
    if ((Sec_G % 2) == 0)
        {
        Time_Str[17] = ':';
        Time_Str[20] = ':';
        }
    else
        {
        Time_Str[17] = ' ';
        Time_Str[20] = ' ';
        }

    PC_LINK_O_Write_String_To_Buffer(Time_Str);
    }
```

```
/*-------------------------------------------------------------*-

   PC_LINK_O_Init_T1()

   Phiên bản này sử dụng T1 để tạo ra tốc độ baud.

   Sử dụng phần cứng UART của 8051

-*-------------------------------------------------------------*/
void PC_LINK_O_Init_T1(const tWord BAUD_RATE)
     {
     PCON &= 0x7F;    /* Set bit SMOD về không (không nhân đôi các tốc độ baud) */

     /*     Bộ nhận vô hiệu dữ liệu 8-bit
, 1 bit start, 1 stop bit stop, baud thay đổi */
     SCON = 0x42;

     TMOD |= 0x20;      /* T1 chế độ 2, 8-bit tự nạp lại giá trị */

     TH1 = (256 - (tByte)((((tLong)OSC_FREQ / 100) * 3125)
                / ((tLong) BAUD_RATE * OSC_PER_INST * 1000)));

     TL1 = TH1;
     TR1 = 1;      /* chạy Timer */
     TI = 1;       /* gửi kí tự đầu tiên (mô hình) */

     /* cài đặt các bộ đệm để đọc và ghi dữ liệu */
     Out_written_index_G = 0;
     Out_waiting_index_G = 0;

     /* không cho phép ngắt hoạt động */
     ES = 0;
     }
```

```
/*-------------------------------------------------------------*/

void PC_LINK_O_Update(void)
    {
    /* nói về các byte truyền nhận ở đây.
        Có dữ liệu nào sẵn sàng gửi chưa? */
    if (Out_written_index_G < Out_waiting_index_G)
        {
        PC_LINK_O_Send_Char(Tran_buffer[Out_written_index_G]);

        Out_written_index_G++;
        }
    else
        {
        /* không có dữ liệu để gửi – reset chỉ số đệm */
        Out_waiting_index_G = 0;
        Out_written_index_G = 0;
        }
    }

/*-------------------------------------------------------------*/

void PC_LINK_O_Write_String_To_Buffer(const char* const STR_PTR)
    {
    tByte i = 0;

    while (STR_PTR[i] != '\0')
        {
        PC_LINK_O_Write_Char_To_Buffer(STR_PTR[i]);
        i++;
        }
    }
```

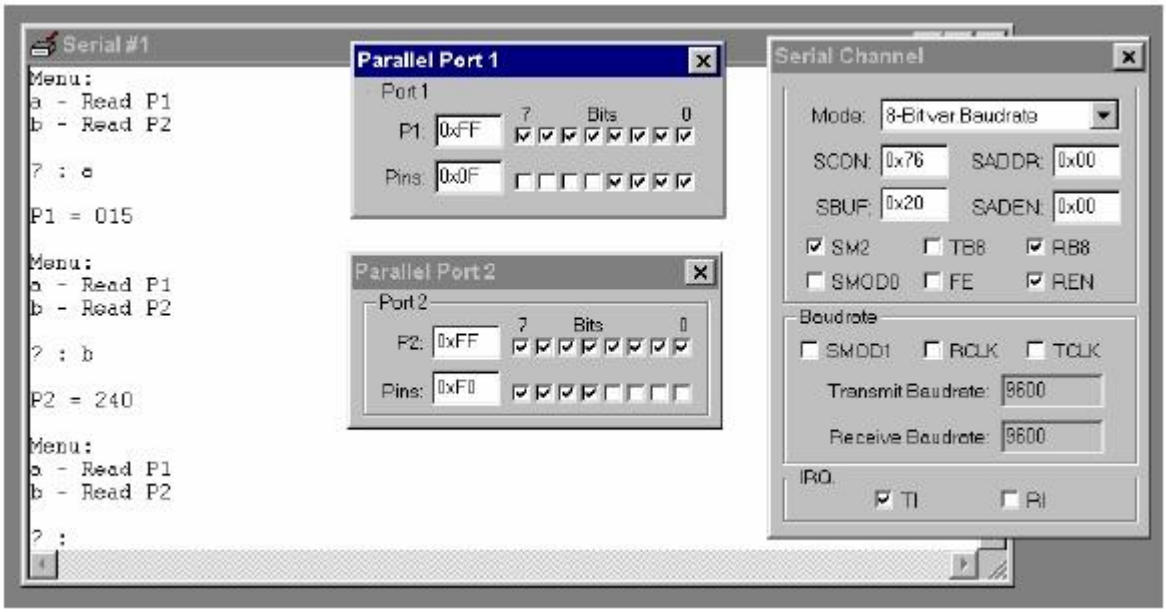
```
/*-------------------------------------------------------------*/
void PC_LINK_O_Write_Char_To_Buffer(const char CHARACTER)
    {
    /* ghi dữ liệu lên bộ đệm chỉ khi nào có chỗ trống
        (không có lỗi báo cáo trong thư viện đơn giản này...) */
    if (Out_waiting_index_G < TRAN_BUFFER_LENGTH)
        {
        Tran_buffer[Out_waiting_index_G] = CHARACTER;
        Out_waiting_index_G++;
        }
    }

/*-------------------------------------------------------------*/
void PC_LINK_O_Send_Char(const char CHARACTER)
    {
    tLong Timeout1 = 0;

    if (CHARACTER == '\n')
        {
        Timeout1 = 0;
        while ((++Timeout1) && (TI == 0));

        if (Timeout1 == 0)
            {
            /* UART không có đáp ứng lỗi
                Không có lỗi báo cáo trong thư viện đơn giản này... */
            return;
            }

        TI = 0;
        SBUF = 0x0d;       /* đầu ra CR         */
        }

    Timeout1 = 0;
    while ((++Timeout1) && (TI == 0));

    if (Timeout1 == 0)
        {
        /* UART did not respond - error
            No error reporting in this simple library... */
        return;
        }

    TI = 0;

    SBUF = CHARACTER;
    }
```

```
sEOS_ISR() interrupt INTERRUPT_Timer_2_Overflow
    {
    TF2 = 0;      /* tụ reset cờ nhớ T2                                  */

    /*===== USER CODE - Begin ================================== */
    /* gọi hàm cập nhật mỗi 5ms */
    PC_LINK_O_Update();

    /* ISR này bị gọi sau mỗi 5 ms
        - nhu cầu cập nhật thời gian sau mỗi giây */
    if (++Call_count_G == 200)
        {
        /* cập nhật thời gian */
        Call_count_G = 0;

        /* gọi hàm cập nhật */
        Elapsed_Time_RS232_Update();
        }
    /*===== USER CODE - End ==================================== */
    }
```

Ví dụ: sự thu nhận tín hiệu

Trong đoạn này,chúng tôi sẽ đưa ra một ví dụ về hệ thống thu nhận tín hiệu đơn giản với cấu trúc **serial-menu** .

trong trường hợp này, sử dụng menu, người dùng cần xác định rõ trạng thái của các chân đầu Port 1 hay Port 2:



.

```
void MENU_Command_Processor(void)
    {
    char Ch;

    if (First_time_only_G == 0)
        {
        First_time_only_G = 1;
        MENU_Show_Menu();
        }

    /* kiểm tra các đầu vào của người dùng */
    PC_LINK_IO_Update();

    Ch = PC_LINK_IO_Get_Char_From_Buffer();

    if (Ch != PC_LINK_IO_NO_CHAR)
        {
        MENU_Perform_Task(Ch);
        MENU_Show_Menu();
        }
    }


void MENU_Show_Menu(void)
    {
    PC_LINK_IO_Write_String_To_Buffer("Menu:\n");
    PC_LINK_IO_Write_String_To_Buffer("a - Read P1\n");
    PC_LINK_IO_Write_String_To_Buffer("b - Read P2\n\n");
    PC_LINK_IO_Write_String_To_Buffer("? : ");
    }
```

```
void MENU_Perform_Task(char c)
    {
    PC_LINK_IO_Write_Char_To_Buffer(c);          /* phản hồi menu lựa chọn */
    PC_LINK_IO_Write_Char_To_Buffer('\n');

    /* chạy tác vụ */
    switch (c)
        {
        case 'a':
        case 'A':
            {
            Get_Data_From_Port1();
            break;
            }

        case 'b':
        case 'B':
            {
            Get_Data_From_Port2();
            break;
            }
        }
    }

void Get_Data_From_Port1(void)
    {
    tByte Port1 = Data_Port1;
    char String[11] = "\nP1 = XXX\n\n";

    String[6] = CHAR_MAP_G[Port1 / 100];
    String[7] = CHAR_MAP_G[(Port1 / 10) % 10];
    String[8] = CHAR_MAP_G[Port1 % 10];

    PC_LINK_IO_Write_String_To_Buffer(String);
    }

void Get_Data_From_Port2(void)
    {
    tByte Port2 = Data_Port2;
    char String[11] = "\nP2 = XXX\n\n";

    String[6] = CHAR_MAP_G[Port2 / 100];
    String[7] = CHAR_MAP_G[(Port2 / 10) % 10];
    String[8] = CHAR_MAP_G[Port2 % 10];

    PC_LINK_IO_Write_String_To_Buffer(String);
    }
```

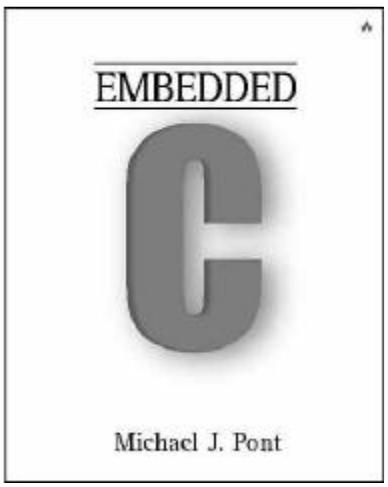
```
sEOS_ISR() interrupt INTERRUPT_Timer_2_Overflow
    {
    TF2 = 0;      /*       phải tự reset cờ T2                          */

    /*===== USER CODE - Begin ================================== */
    /* gọi MENU_lệnh_của bộ điều khiển mỗi 5ms */
    MENU_Command_Processor();

    /*===== USER CODE - End ==================================== */
    }
```

Kết luận

Trong chuyên đề này, chúng ta minh họa cách sử dụng giao diện nối tiếp trên vi điều khiển 8051.

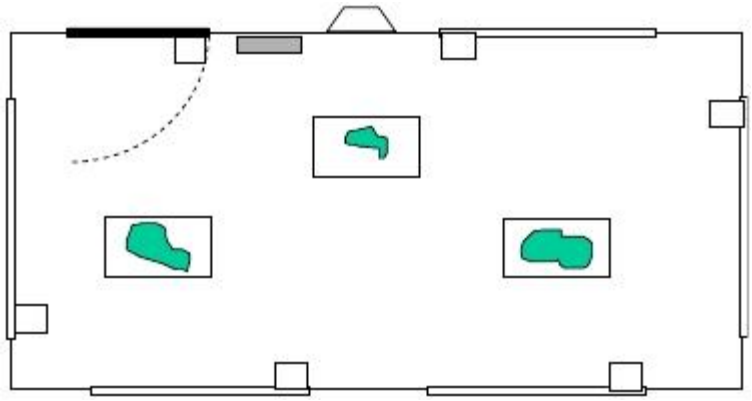
Trong chuyên đề tiếp, chúng ta sẽ sử dụng nghiên cứu trường hợp để minh họa các kĩ thuật đã thảo luận ở đây được sử dụng trong các ứng dụng thực tế như thế nào.

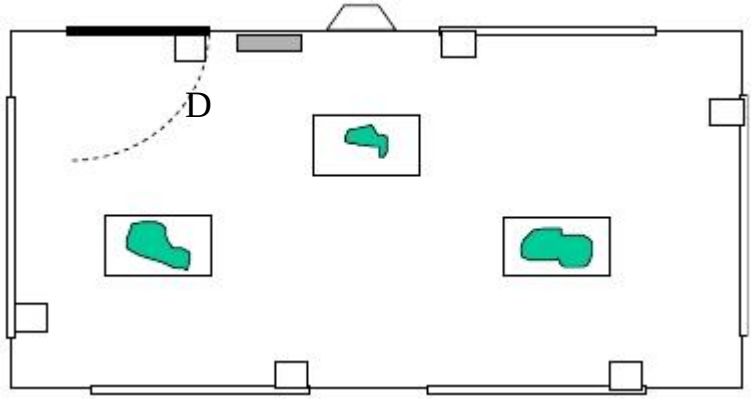
Chuẩn bị cho chuyên đề tiếp theo

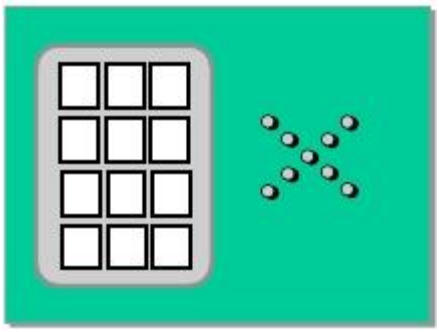


Xin đọc **Chương 10**
Trước khi bắt đầu chuyên đề tiếp theo

Chương 9: Hệ thống báo động người xâm phạm




D



Sự hoạt động của hệ thống

Khi bắt đầu kích hoạt, hệ thống ở trạng thái 'không bảo vệ' (Disarmed).

Trong trạng thái này, các cảm biến bị bỏ qua. Báo động không kêu.
Hệ thống ở trạng thái này đến khi người dùng nhập vào mật khẩu đúng qua bàn phím(trong hệ thống chứng thực của chúng ta mật khẩu là "1234"). Khi một mật khẩu đúng được nhập vào,hệ thống cho vào trạng
thái 'trang bị' (Arming) .

Trong trạng thái **'Arming'**, hệ thống đợi trong 60 s,để cho phép người dùng ra khỏi khu vực này trước khi quá trình kiểm soát bắt đầu. sau 60s,
hệ thống đưa vào trạng thái 'được trang bị'(armed).

Trong trạng thái **'Armed'**, tình trạng của các cảm biến của hệ thống
sẽ được kiểm soát. Nếu một cảm biến ở cửa sổ có tín hiệu thì hệ thống sẽ đưa ra trạng thái 'có người đột nhập'. Nếu cảm biến ở cửa có tín hiệu thì hệ thống đưa vào trạng thái 'nguôi giận' (disarming). Hoạt động của bàn phím cũng được kiểm soát: nếu một mật khẩu đúng được đánh vào, hệ thống sẽ đưa vào trạng thái'không bảo vệ' state.

Trong trạng thái **'nguôi giận'**, chúng ta giả sử rằng cửa được mở bởi người nào đó, có thể là một người sử dụng được ủy quyền.
Hệ thống sẽ ở trạng thái này tới 60 giây, sau đó, theo mặc định, nó sẽ đưa vào trạng thái có người xâm nhập . nếu trong chu kì 60s, người sử dụng nhập vào mật khẩu đúng, hệ thống sẽ đưa vào trạng thái 'không bảo vệ'.

Trong trạng thái **có người xâm nhập** , chuông báo động sẽ kêu. Hệ thống báo động sẽ kêu (vô hạn định) tới khi mât khẩu đúng được nhập vào.

.

Các thành phần phần mềm ‘khóa’ được sử dụng trong ví dụ này

chúng ta sử dụng những thành phần phần mềm sau:

Phần mềm điều khiển các chân ngoài (để kích hoạt chuông), được giới thiều trong cuốn “Embedded C” Chương 3.

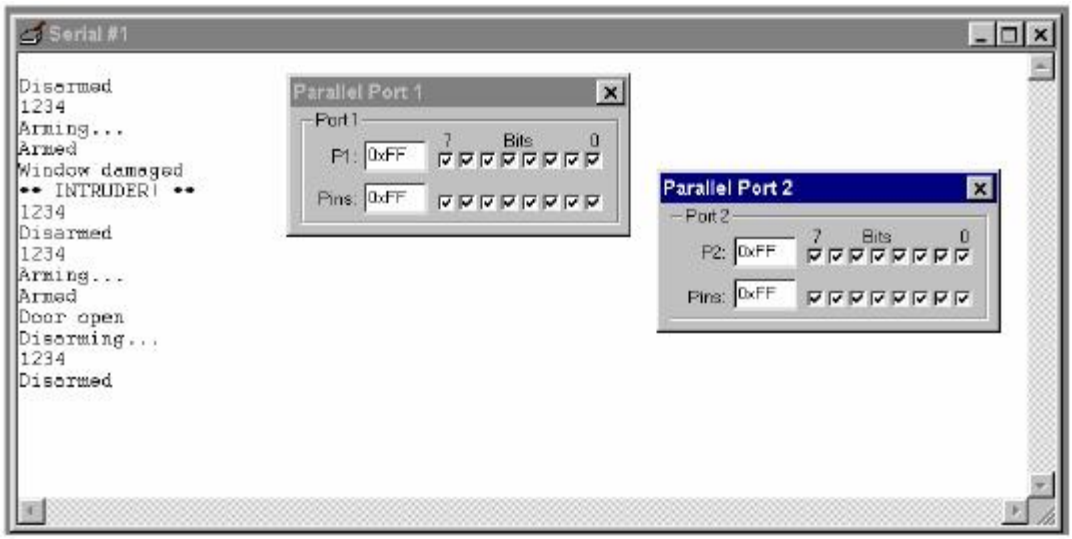
Đọc công tác, (đã thảo luận trong “Embedded C” Chương 4) để truy cập vào các đầu vào từ các cảm biến cửa và cửa sổ. Chú ý rằng, trong ví dụ đơn giản này(với mục đích minh họa) không có hãm công tác nào được thực hiện. Thành phần này có thể được thêm vào (nếu được yêu cầu) mà không hề khó khăn.

Hệ thống nhúng, sEOS, được giới thiệu trong cuốn “Embedded C” Chương 7.

Một thư viện ‘bàn phím’ , căn cứ trên một dãy các công tắc. Chú ý rằng, để đơn giản khi sử dụng thư viện bàn phím trong phần mô phỏng này, chúng ta giả sử chỉ có 8 phím bấm trong chương trình ví dụ (0 - 7). Hệ thống này chắc chắn có thể sử dụng ít nhất 10 phím bấm: việc thêm vào các phím bấm rất dễ dàng nếu được yêu cầu.

Thư viện RS-232 (từ “Embedded C” Chương 9) được sử dụng để minh họa cho sự hoạt động của chương trình. Thư viện này có thể không cần thiết trong hệ thống cuối (nhưng có thể hữu ích khi dùng để trợ giúp bảo dưỡng hệ thống).

Chạy chương trình



Phần mềm

```
/*---------------------------------------------------------*-

    Port.H (v1.00)

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

   'Port Header' (xem chuong 5)dự án INTRUDER (xem chuong 10)

-*---------------------------------------------------------*/

/* ------ Keypad.C ---------------------------------------- */

#define KEYPAD_PORT P2

sbit    K0  =  KEYPAD_PORT^0;
sbit    K1  =  KEYPAD_PORT^1;
sbit    K2  =  KEYPAD_PORT^2;
sbit    K3  =  KEYPAD_PORT^3;
sbit    K4  =  KEYPAD_PORT^4;
sbit    K5  =  KEYPAD_PORT^5;
sbit    K6  =  KEYPAD_PORT^6;
sbit    K7  =  KEYPAD_PORT^7;

/* ------ Intruder.C ---------------------------------------- */
sbit Sensor_pin = P1^0;
sbit Sounder_pin = P1^7;

/* ------ Lnk_O.C ---------------------------------------- */

/* Chân 3.0 và 3.1 được sử dụng cho giao diện kết nối RS-232 */


/*---------------------------------------------------------*-
   ---- END OF FILE ----------------------------------
-*---------------------------------------------------------*/
```

```
/*-------------------------------------------------------*-

    Main.c (v1.00)

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

    Hệ thống báo động đơn giản.

-*-------------------------------------------------------*/

#include "Main.H"
#include "Port.H"
#include "Simple_EOS.H"

#include "PC_O_T1.h"
#include "Keypad.h"
#include "Intruder.h"

/* ..................................................... */

void main(void)
    {
    /* cai dat toc do baud la 9600 */
    PC_LINK_O_Init_T1(9600);

    /* chuan bi cho ban phim */
    KEYPAD_Init();

    /* chuan bi cho bao dong xam nhap */
    INTRUDER_Init();

    /* cai simple EOS (tick 5ms) */
    sEOS_Init_Timer2(5);

    while(1) /* vong lap vo tan */
        {
        sEOS_Go_To_Sleep();       /* dua vao che do nghi de tiet kiem nang luong */
        }
    }

/*-------------------------------------------------------*-
   ---- END OF FILE ----------------------------------
-*-------------------------------------------------------*/
```

```
/*-------------------------------------------------------*-

    Intruder.C (v1.00)

-*-------------------------------------------------------*/

...

/* ------ khai bao du lieu chung ---------------------- */

/* cac trang thai co kha nang cua he thong */
typedef enum {DISARMED, ARMING, ARMED, DISARMING, INTRUDER}
                    eSystem_state;

/* ------ ham nguyen mau rieng ------------------------ */

bit     INTRUDER_Get_Password_G(void);
bit     INTRUDER_Check_Window_Sensors(void);
bit     INTRUDER_Check_Door_Sensor(void);
void    INTRUDER_Sound_Alarm(void);

...

/* ------------------------------------------------------------ */
void INTRUDER_Init(void)
    {
    /* cai dat trang thai dau cua he thong (DISARMED) */
    System_state_G = DISARMED;

    /* Set 'thoi gian trong trang thai've 0 */
    State_call_count_G = 0;

    /* xoa bo dem ban phim */
    KEYPAD_Clear_Buffer();

    /* Set co 'trang thai moi' */
    New_state_G = 1;

    /* Set hai chan cam bien de 'doc' che do */
    Window_sensor_pin = 1;
    Sounder_pin = 1;
    }
```

```
void INTRUDER_Update(void)
    {
    /* gia tang thoi gian */
    if (State_call_count_G < 65534)
        {
        State_call_count_G++;
        }

    /* goi ra sau moi 50 ms */
    switch (System_state_G)
        {
        case DISARMED:
            {
            if (New_state_G)
                {
                PC_LINK_O_Write_String_To_Buffer("\nDisarmed");
                New_state_G = 0;
                }

            /* chac chan rang canh bao da tat */
            Sounder_pin = 1;

            /* chờ mật khẩu đúng ... */
            if (INTRUDER_Get_Password_G() == 1)
                {
                System_state_G = ARMING;
                New_state_G = 1;
                State_call_count_G = 0;
                break;
                }

            break;
            }
```

```
case ARMING:
    {
    if (New_state_G)
        {
        PC_LINK_O_Write_String_To_Buffer("\nArming...");
        New_state_G = 0;
        }

    /* doi o day 60 giay (gia su khoang tick la 50 ms) */
    if (++State_call_count_G > 1200)
        {
        System_state_G = ARMED;
        New_state_G = 1;
        State_call_count_G = 0;
        break;
        }

    break;
    }
```

```
case ARMED:
    {
    if (New_state_G)
        {
        PC_LINK_O_Write_String_To_Buffer("\nArmed");
        New_state_G = 0;
        }

    /* dau tien, kiem tra cam bien cua so */
    if (INTRUDER_Check_Window_Sensors() == 1)
        {
        /* mot nguoi la duoc phat hien */
        System_state_G = INTRUDER;
        New_state_G = 1;
        State_call_count_G = 0;
        break;
        }

    /* tiep theo,kiem tra cac cam bien cua */
    if (INTRUDER_Check_Door_Sensor() == 1)
        {
        /* co the la nguoi su dung duoc uy quen – toi trang thai 'Disarming' */
        System_state_G = DISARMING;
        New_state_G = 1;
        State_call_count_G = 0;
        break;
        }

    /* cuoi cung, kiem tra mat khau dung */
    if (INTRUDER_Get_Password_G() == 1)
        {
        System_state_G = DISARMED;
        New_state_G = 1;
        State_call_count_G = 0;
        break;
        }

    break;
    }
```

```
case DISARMING:
    {
    if (New_state_G)
        {
        PC_LINK_O_Write_String_To_Buffer("\nDisarming...");
        New_state_G = 0;
        }

    /* doi o day 60s (gia su khoang thoi gian tick la 50 ms)
        De cho phep nguoi su dung nhap mat khau dung
        - sau do neu het thoi gian thi co tieng bao dong. */
    if (++State_call_count_G > 1200)
        {
        System_state_G = INTRUDER;
        New_state_G = 1;
        State_call_count_G = 0;
        break;
        }

    /* van phai kiem tra cac cam bien cua so */
    if (INTRUDER_Check_Window_Sensors() == 1)
        {
        /* mot nguoi la duoc phat hien */
        System_state_G = INTRUDER;
        New_state_G = 1;
        State_call_count_G = 0;
        break;
        }

    /* cuoi cung, kiem tra mat khau dung hay chua */
    if (INTRUDER_Get_Password_G() == 1)
        {
        System_state_G = DISARMED;
        New_state_G = 1;
        State_call_count_G = 0;
        break;
        }

    break;
    }
```

```
        case INTRUDER:
            {
            if (New_state_G)
                {
                PC_LINK_O_Write_String_To_Buffer("\n** INTRUDER! **");
                New_state_G = 0;
                }

            /* co tieng bao dong! */
            INTRUDER_Sound_Alarm();

            /* giu nguyen tieng bao dong den khi mat khau dung duoc nhap
             vao */
            if (INTRUDER_Get_Password_G() == 1)
                {
                System_state_G = DISARMED;
                New_state_G = 1;
                State_call_count_G = 0;
                }

            break;
            }
        }
}
```

```
bit INTRUDER_Get_Password_G(void)
    {
    signed char Key;
    tByte Password_G_count = 0;
    tByte i;

    /* cap nhat bo dem ban phim */
    KEYPAD_Update();

    /* co du lieu moi o bo dem ban phim khong? */
    if (KEYPAD_Get_Data_From_Buffer(&Key) == 0)
        {
        /* khong co du lieu moi – mat khau sai */
        return 0;
        }

    /* neu chung ta o do, mot phim se duoc bam*/

    /* mat bao lau ke tu khi phim cuoi cung duoc bam?
        Phai bam trong 50s (gia su  khoang thoi gian tick la 50 ms ) */
    if (State_call_count_G > 1000)
        {
        /* them 5s tinh t khi phim cuoi cung duoc bam
            - khoi dong lai xu ly dau vao */
        State_call_count_G = 0;
        Position_G = 0;
        }

    if (Position_G == 0)
        {
        PC_LINK_O_Write_Char_To_Buffer('\n');
        }

    PC_LINK_O_Write_Char_To_Buffer(Key);

    Input_G[Position_G] = Key;
```

```
/* co phai mat khau co 4 so? */
if ((++Position_G) == 4)
    {
    Position_G = 0;
    Password_G_count = 0;

    /* kiem tra mat khau */
    for (i = 0; i < 4; i++)
        {
        if (Input_G[i] == Password_G[i])
            {
            Password_G_count++;
            }
        }
    }

if (Password_G_count == 4)
    {
    /* mat khau dung */
    return 1;
    }
else
    {
    /* mat khau sai */
    return 0;
    }
}
```

```
bit INTRUDER_Check_Window_Sensors(void)
    {
    /* chi mot cam bien cua so o day
        - nang cap de dang. */
    if (Window_sensor_pin == 0)
        {
        /* nguoi la duoc phat hien... */
        PC_LINK_O_Write_String_To_Buffer("\nWindow damaged");
        return 1;
        }

    /* mac dinh */
    return 0;
    }

/* ------------------------------------------------------------ */
bit INTRUDER_Check_Door_Sensor(void)
    {
    /* cam bien cua don (truy cap hanh trinh) */
    if (Door_sensor_pin == 0)
        {
        /* co ai do mo cua... */
        PC_LINK_O_Write_String_To_Buffer("\nDoor open");
        return 1;
        }

    /* mac dinh */
    return 0;
    }

/* ------------------------------------------------------------ */
void INTRUDER_Sound_Alarm(void)
    {
    if (Alarm_bit)
        {
        /* canh bao noi voi cac chan nay */
        Sounder_pin = 0;
        Alarm_bit = 0;
        }
    else
        {
        Sounder_pin = 1;
        Alarm_bit = 1;
        }
    }
```

```
void KEYPAD_Update(void)
    {
    char Key;

    /* quet ban phim o day... */
    if (KEYPAD_Scan(&Key) == 0)
        {
        /* khong co du lieu ban phim – quay lai */
        return;
        }

    /* muon tim ra chi so 0, neu du lieu cu khong duoc doc
        (simple ~circular buffer). */
    if (KEYPAD_in_waiting_index == KEYPAD_in_read_index)
        {
        KEYPAD_in_waiting_index = 0;
        KEYPAD_in_read_index = 0;
        }

    /* nap du lieu ban phim vao bo dem */
    KEYPAD_recv_buffer[KEYPAD_in_waiting_index] = Key;

    if (KEYPAD_in_waiting_index < KEYPAD_RECV_BUFFER_LENGTH)
        {
        /* tang ma khong lam tran bo dem */
        KEYPAD_in_waiting_index++;
        }
    }


bit KEYPAD_Get_Data_From_Buffer(char* const pKey)
    {
    /* neu co du lieu moi o bo dem */
    if (KEYPAD_in_read_index < KEYPAD_in_waiting_index)
        {
        *pKey = KEYPAD_recv_buffer[KEYPAD_in_read_index];

        KEYPAD_in_read_index++;

        return 1;
        }

    return 0;
    }
```

```
bit KEYPAD_Scan(char* const pKey)
    {
    char Key = KEYPAD_NO_NEW_DATA;

    if   (K0  ==  0)  {  Key  =  '0';    }
    if   (K1  ==  0)  {  Key  =  '1';    }
    if   (K2  ==  0)  {  Key  =  '2';    }
    if   (K3  ==  0)  {  Key  =  '3';    }
    if   (K4  ==  0)  {  Key  =  '4';    }
    if   (K5  ==  0)  {  Key  =  '5';    }
    if   (K6  ==  0)  {  Key  =  '6';    }
    if   (K7  ==  0)  {  Key  =  '7';    }

    if (Key == KEYPAD_NO_NEW_DATA)
        {
        /*khong co phim nao duoc bam */
        Old_key_G = KEYPAD_NO_NEW_DATA;
        Last_valid_key_G = KEYPAD_NO_NEW_DATA;

        return 0;       /* khong co du lieu moi */
        }

    /* mot phim duoc bam: debounce boi hai lan kiem tra */
    if (Key == Old_key_G)
        {
        /*mot phim bam dung duoc phat hien */

        /* phai la mot phim bam moi – khong tu dong nap lai */
        if (Key != Last_valid_key_G)
            {
            /* phim moi! */
            *pKey = Key;
            Last_valid_key_G = Key;

            return 1;
            }
        }

    /* khong co du lieu moi */
    Old_key_G = Key;
    return 0;
    }
```

```
sEOS_ISR() interrupt INTERRUPT_Timer_2_Overflow
    {
    TF2 = 0;      /* tu reset co T2                                    */

    /*===== USER CODE - Begin ================================== */
    /* goi ham cap nhat RS-232 moi 5ms */
    PC_LINK_O_Update();

    /* ISR nay duoc goi sau moi 5 ms
        - chi muon cap nhat xem co nguoi la khong sau moi 50 ms. */
    if (++Call_count_G == 10)
        {
        /* thoi gian de cap nhat canh bao co ke dot nhap */
        Call_count_G = 0;

        /* goi ham cap nhat intruder */
        INTRUDER_Update();
        }
    /*===== USER CODE - End ==================================== */
    }
```

Nâng cấp và chỉnh sửa hệ thống

Làm thế nào để bạn thêm một bàn phím ‘thực’ ?

(xem cuốn “Patterns for Time-Triggered Embedded Systems, chương. 20)

Bạn sẽ thêm một màn hình hiển thị LCD như thế nào ?

(xem cuốn “Patterns for Time-Triggered Embedded Systems, Chương. 22)

Bạn sẽ thêm các nút (node) như thế nào ?

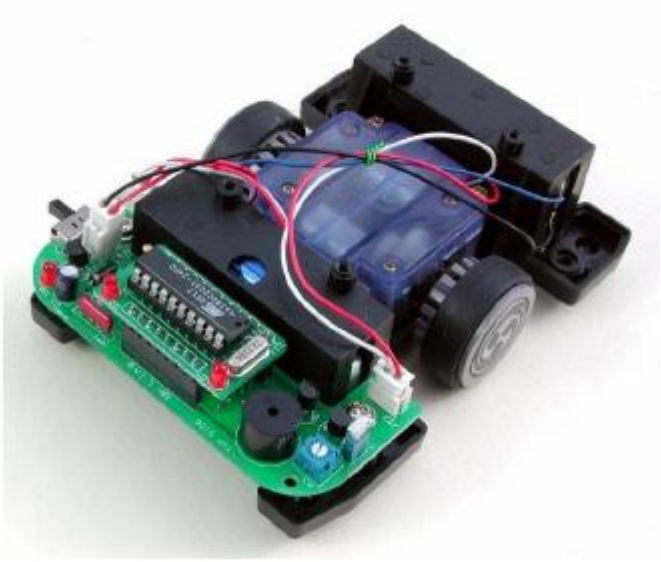
(xem cuốn “Patterns for Time-Triggered Embedded Systems, Phần F)

Kết luận

Trường hợp nghiên cứu này minh họa hầu hết các chức năng của ngôn ngữ nhúng C (như đã được thảo luận trong các chuyên đề trước đó của khóa học này).

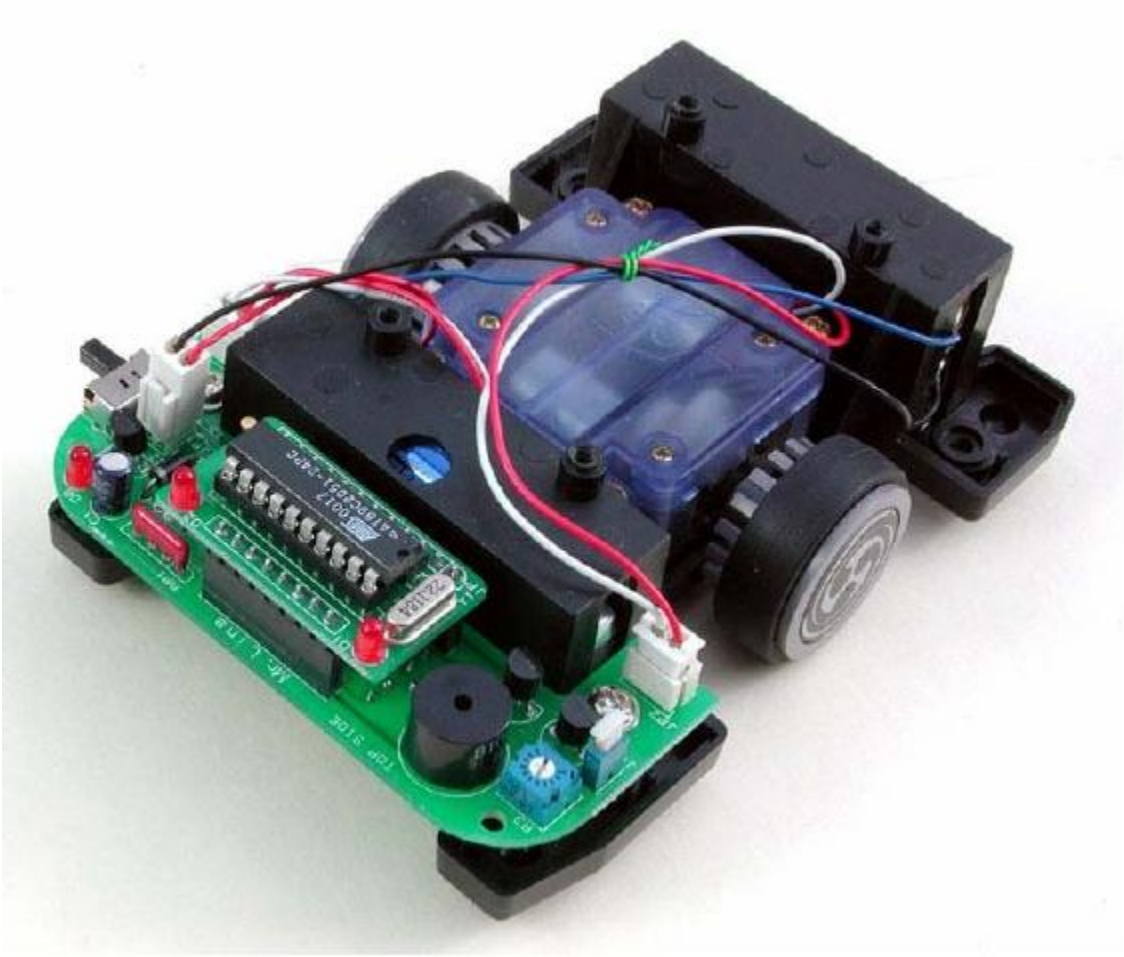
Chúng ta sẽ xem xét trường hợp nghiên cứu cuối cùng trong chuyên đề tiếp theo.

Chương 10: Điều khiển một robot di chuyển



Khái quát chung

Trong phần này, chúng ta sẽ thảo luận việc thiết kế một phần mềm để điều khiển một con robot di động nhỏ.



Robot này là “Mr Line”

Nó được sản xuất bởi “Microrobot NA”

http://www.microrobotna.com

Robot có thể làm gì?

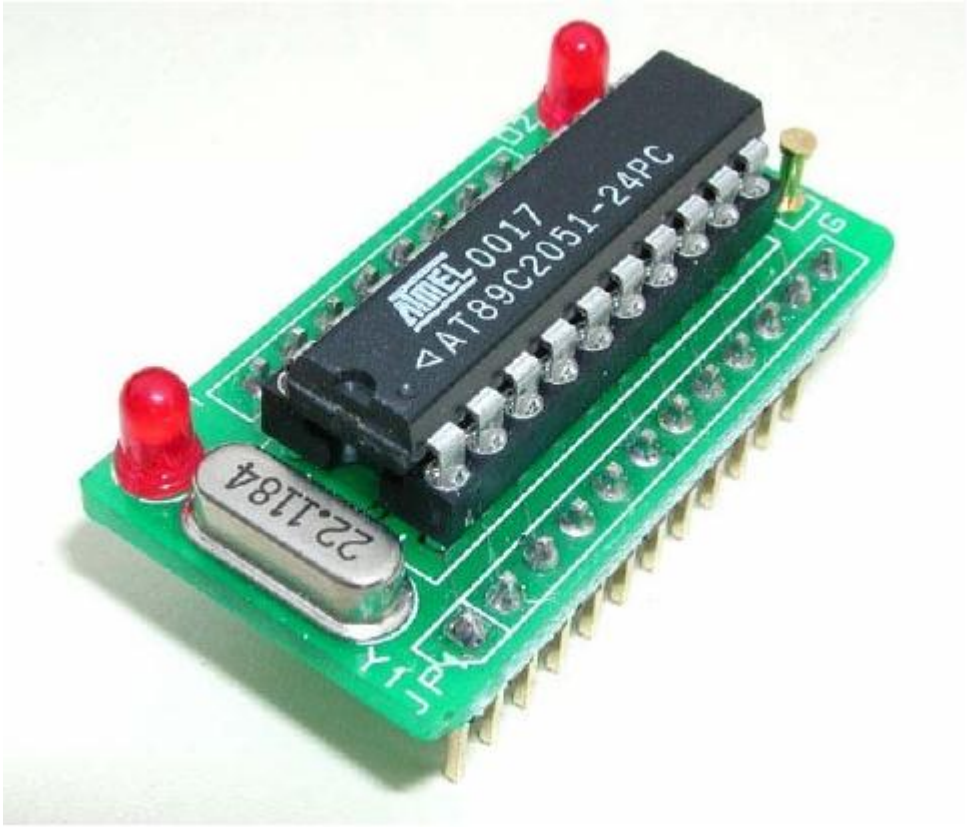
Robot có các cảm biến hồng ngoại và thiết bị phát, cho phép nó phát hiện ra vạch đen trên nền trắng và bám theo vách đen đó.

http://www.microrobotna.com

Bộ não của robot

Mr Line được điều khiển bởi một vi điều khiển họ 8051 (AT89C2051).

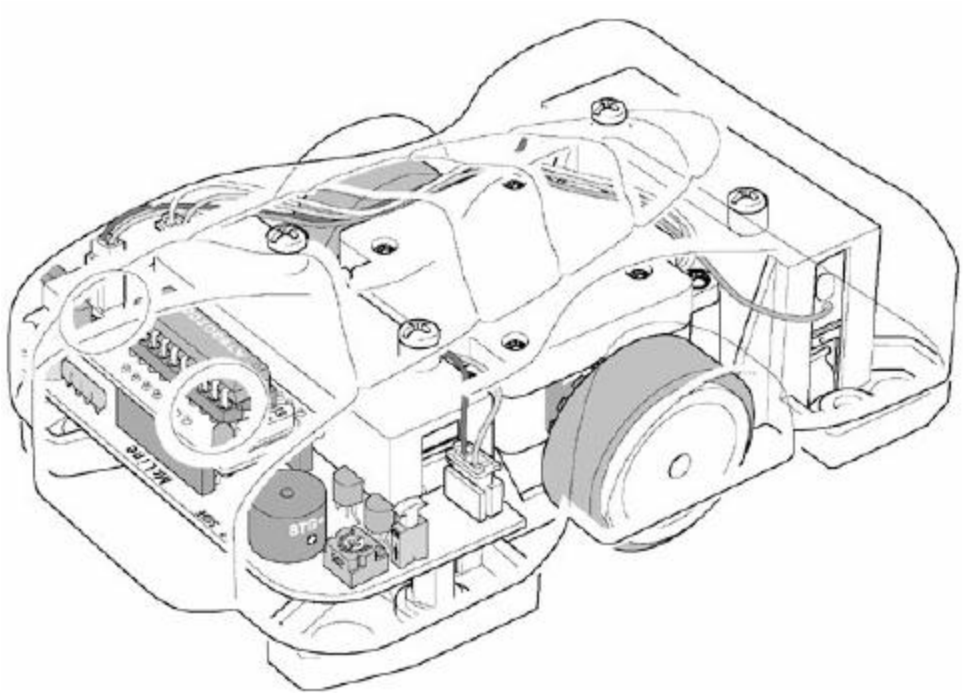
Chúng ta sẽ sử dụng một vi điều khiển có chân tương thích là AT89C4051.



http://www.microrobotna.com

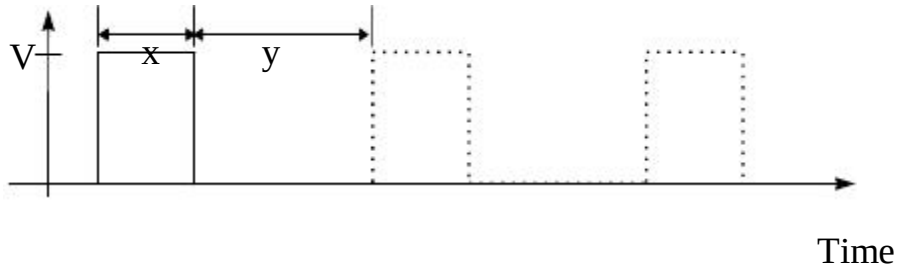
Robot chuyển động như thế nào?

Mr Line có hai motor: bằng việc điều khiển tốc độ tỉ đối của hai motor này chúng ta có thể điều chỉnh tốc độ và hướng của con robot khi nó di chuyển.



http://www.microrobotna.com

Điều chế độ rộng xung



Chế độ làm việc(%) = $\frac{x}{x \quad y}$ 100

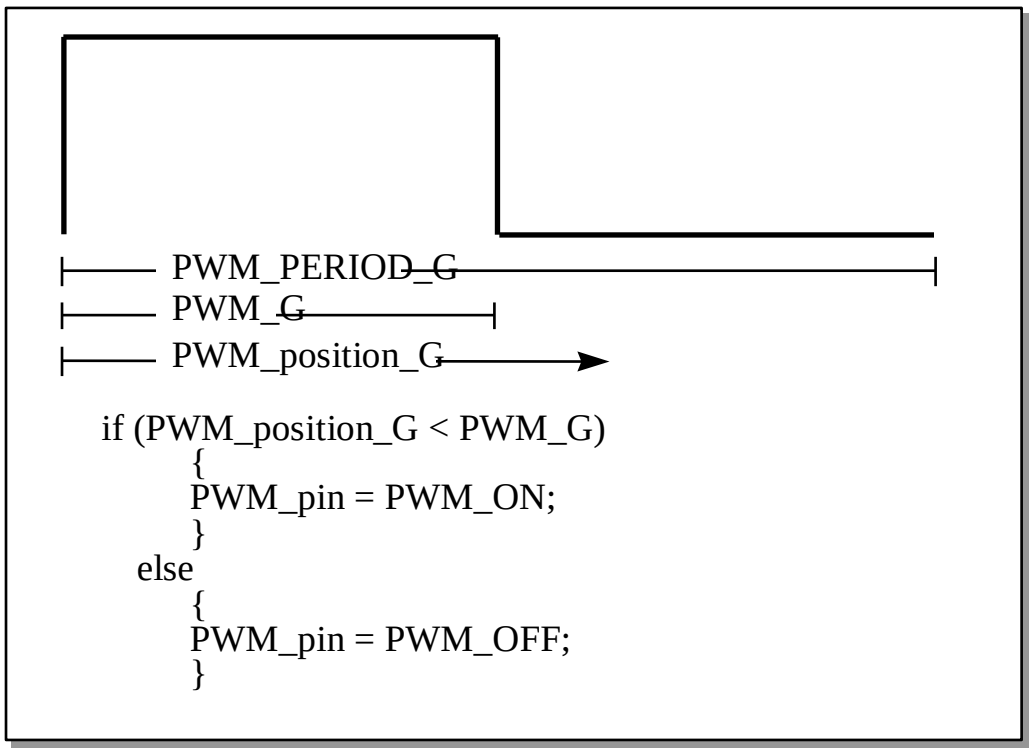
chu kì = *x* + *y*, (trong đó x và y được tính bằng giây).

Tần số = $\frac{1}{x \quad y}$,.

Chú ý:điền áp trung bình được tính băng ‘chế độ làm việc’ nhân với điện áp tái.

Xem cuốn: **“Patterns for Time-Triggered Embedded Systems”**, Chương 33

.



Xem cuốn: **"Patterns for Time-Triggered Embedded Systems"**, Chương 33

Để biết thêm về robot

Xem tại:

http://www.le.ac.uk/engineering/mjp9/robot.htm

Kết thúc

That brings us to the end of this course!

Bản dịch đầu tay không thể thiếu những sai sót. Rất mong nhận được sự góp ý của các bạn.
Mọi thắc mắc về bản dịch xin vui lòng gửi tới :

doanhau198@gmail.com

tranthien.pro@gmail.com

Atienganh.blogspot.com