



BỘ MÔN ĐIỆN TỬ
Department of Electronics

TÀI LIỆU THÍ NGHIỆM VI XỬ LÝ

CHƯƠNG 1 TỔNG QUAN VỀ BỘ THÍ NGHIỆM

1.1 TỔ CHỨC TÀI LIỆU HƯỚNG DẪN

Kit thí nghiệm vi xử lý là bộ thí nghiệm được thiết kế dựa trên họ vi điều khiển MCS-51. Tài liệu hướng dẫn thí nghiệm này giúp người sử dụng tiếp cận với các kiến thức cơ bản về vi điều khiển 8051 nhanh chóng hơn. Tài liệu thí nghiệm bao gồm tài liệu hướng dẫn sử dụng kit thí nghiệm, các bài thí nghiệm, và một số mã nguồn để tham khảo.

Tài liệu hướng dẫn sẽ giới thiệu các thành phần của kit thí nghiệm, được tổ chức thành các phần như sau:

- ✚ **Lý thuyết cơ bản:** phần này sẽ tóm tắt sơ lược các kiến thức lý thuyết có liên quan đến bài thí nghiệm.
- ✚ **Thiết kế phần cứng:** nội dung của phần này sẽ giúp người sử dụng nắm được chi tiết về sơ đồ và cách thức thiết kế phần cứng của kit thí nghiệm. Người sử dụng cần hiểu rõ các nội dung được đề cập trong phần này. Các thiết kế phần cứng này hoàn toàn có thể ứng dụng trong thực tế.
- ✚ **Phần mềm giao tiếp:** phần này sẽ giúp người sử dụng nắm được các kỹ thuật để xây dựng phần mềm đáp ứng yêu cầu của bài thí nghiệm. Các nội dung được đề cập trong phần này cũng sẽ rất hữu dụng trong thực tế.

Mỗi bài thí nghiệm được tổ chức thành các phần như sau:

- ✚ **Mục tiêu:** giúp người học nắm được mục tiêu cụ thể của bài thí nghiệm.
- ✚ **Yêu cầu:** phần này sẽ đưa ra yêu cầu cụ thể của bài thí nghiệm.
- ✚ **Hướng dẫn:** phần này nêu một số hướng dẫn để SV có thể lập trình dễ dàng hơn
- ✚ **Kiểm tra:** giúp người sử dụng đánh giá mức độ đạt được các mục tiêu của bài thí nghiệm, đồng thời gợi ý một số hiệu chỉnh nhằm làm phong phú nội dung thí nghiệm.

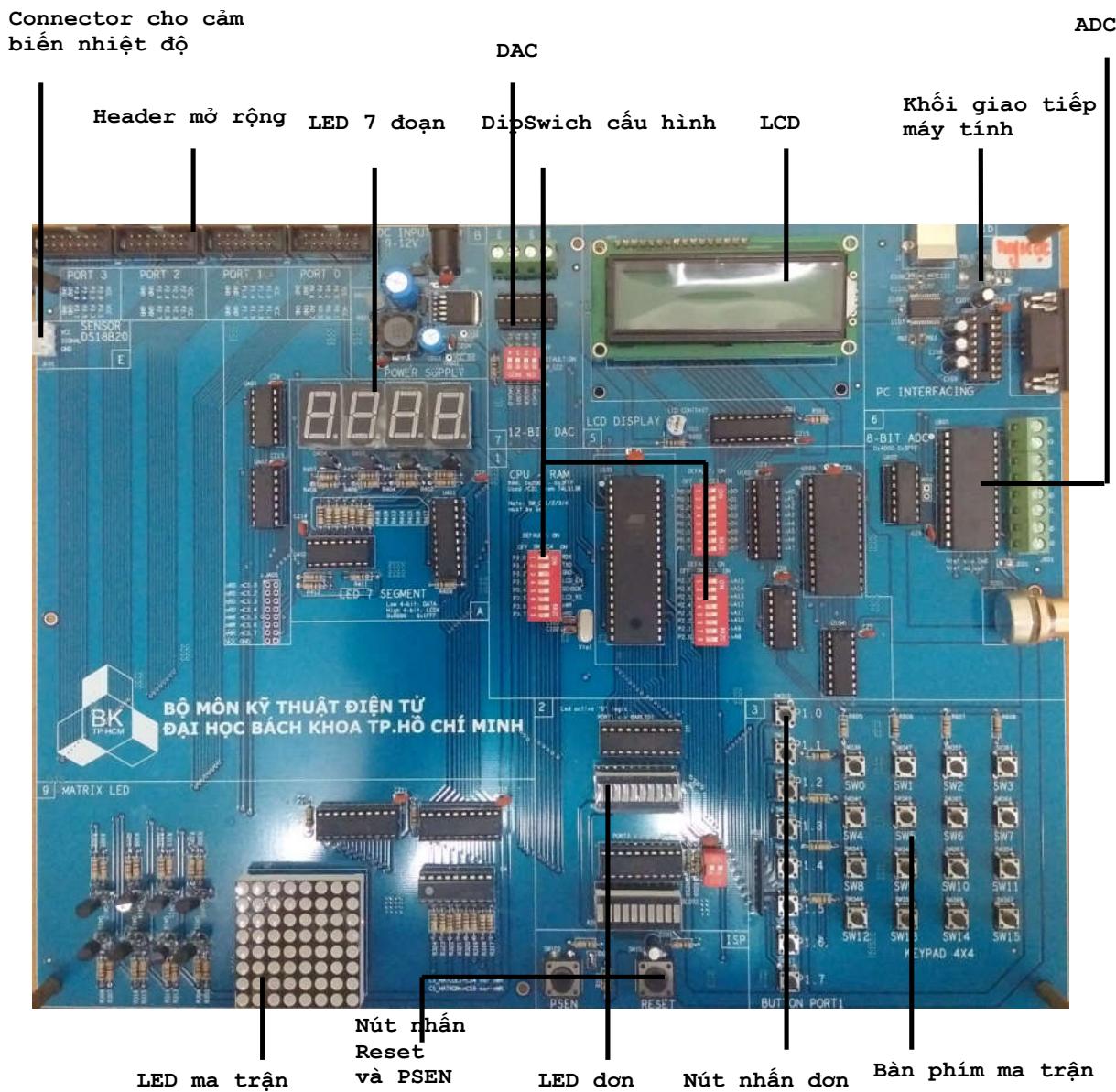
HƯỚNG DẪN THÍ NGHIỆM VI XỬ LÝ

Chú ý:

Người học cần xem trước nội dung thí nghiệm và chuẩn bị sẵn chương trình tại nhà để có thể tận dụng tốt thời gian thí nghiệm. Nếu phát hiện có sai sót hay thắc mắc, người học có thể báo trực tiếp Giảng Viên hướng dẫn hoặc email về địa chỉ buiquocbao@hcmut.edu.vn

1.2 TỔNG QUAN KIT THÍ NGHIỆM

Kit thí nghiệm có hình dạng và các khối cơ bản như Hình 1.



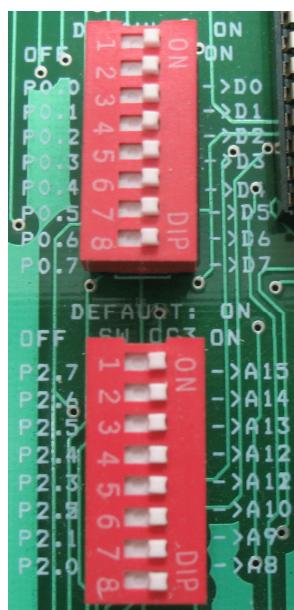
Hình 1 Tổng quan kit thí nghiệm vi xử lý

HƯỚNG DẪN THÍ NGHIỆM VI XỬ LÝ

Chú ý rằng trên kit có 2 nút nhấn RESET để reset lại chương trình và PSEN dùng để đưa chip vi điều khiển về chế độ lập trình, tuy nhiên trong chương trình thí nghiệm ta sẽ dùng chương trình monitor để thực thi chương trình ngay trên RAM (xếp chồng bộ nhớ chương trình và dữ liệu), vì vậy nút nhấn PSEN không được dùng đến.

1.2.1 Các DIP-SW8 cấu hình

Bộ thí nghiệm vi xử lý được thiết kế để có thể sử dụng một cách linh hoạt. Trên EME-MC8 có 4 DIP-SW8 SW_CC1, SW_CC2, SW_CC3, SW_CC4 nằm xung quanh DIP40 gắn vi điều khiển cho phép cấu hình bộ thí nghiệm.



Hình 2 Các dip switch cấu hình

Một phía của DIP-SW được nối đến 4 port (P0 đến P3) của 8051. Phía còn lại nối đến tín hiệu có tên được ghi trên board mạch. Mục đích của các DIP-SW8 cấu hình này là cho phép ngắn mạch hoặc hở mạch tín hiệu với port của 8051. Cụ thể là khi SW được đặt ở vị trí ON, hai tín hiệu được nối. Khi SW đặt ở vị trí OFF, hai tín hiệu hở mạch. Như vậy, khi SW tại một vị trí bật ON thì port của 8051 được nối với tín hiệu có tên tương ứng. Ví dụ khi SW thứ 0 của DIP-SW8 SW_CC2 bật ON thì có nghĩa là bit P1.0 (bit thứ 0 của Port1) được nối đến tín hiệu **DAC_nCS**.

Port	Chức năng
P0	Bus dữ liệu D0-D7

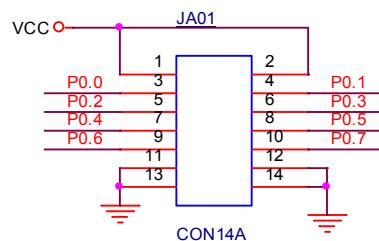
HƯỚNG DẪN THÍ NGHIỆM VI XỬ LÝ

	Byte thấp của bus địa chỉ A0-A7
P2	Byte cao của bus địa chỉ A8-A15
P1.0	/CS của DAC
P1.1	SCK của giao tiếp SPI
P1.2	SDI của giao tiếp SPI
P1.3	/LD của DAC
P3.0	RXD
P3.1	TXD
P3.4	Tín hiệu giao tiếp 1-Wire bus Ngõ ra của DS18S20
P3.3	Tín hiệu Enable của LCD
P3.2	Chọn chế độ Single-Step
P3.5	Tín hiệu RS của LCD
P3.6	/WR
P3.7	/RD

Bảng 1 Kết nối các port với tín hiệu khi DIP-SW8 cấu hình được bật ON

1.2.2 CÁC CONNECTOR MỞ RỘNG PORT

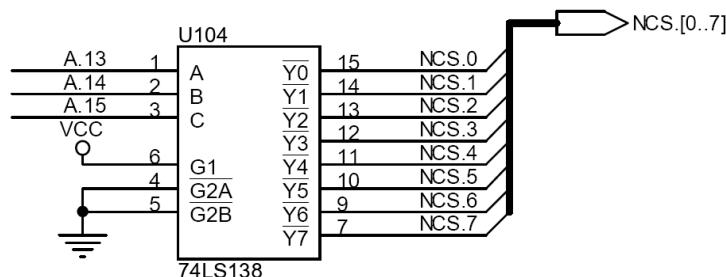
Các port được nối trực tiếp đến header mở rộng có sẵn trên board là **JA01** (Port0), **JA02** (Port1), **JA03** (Port2), **JA04** (Port3). Người sử dụng có thể dùng các header này cho mục đích kết nối kit với board mở rộng.



Hình 3 Connector mở rộng Port 0

HƯỚNG DẪN THÍ NGHIỆM VI XỬ LÝ

Mạch giải mã địa chỉ trên kit được thiết kế dùng vi mạch giải mã 74x138. Sơ đồ thiết kế như hình sau



Hình 4 Sơ đồ mạch giải mã địa chỉ

Bản đồ bộ nhớ được sắp xếp như sau

STT	Bộ nhớ và Ngoại vi	Địa chỉ truy xuất	Ghi chú
1	Bộ nhớ RAM ngoài	2000h – 3FFFh	Chứa dữ liệu và chương trình của người sử dụng khi dùng với EME-MON51 (nCS1)
2	Chốt ‘573 của khối led 7 đoạn	0000h – 1FFFh	Chỉ ghi, (nCS0)
3	Ra lệnh bắt đầu chuyển đổi ADC và chốt kênh cần chuyển đổi	4000h – 5FFFh	Chỉ ghi (nCS2)
4	Đọc 8 bit dữ liệu từ ADC	4000h – 5FFFh	Chỉ đọc (nCS2)
5	Điều khiển chốt ‘573 chốt 8 bit dữ liệu của khối LCD	6000h – 7FFFh	Chỉ ghi (nCS3)
7	Điều khiển chốt ‘573 chốt 8 bit dữ liệu của khối led matrix cột	8000 – 9FFFh	Chỉ ghi (nCS4)
8	Điều khiển chốt ‘573 chốt 8 bit dữ liệu của khối led matrix hàng	A000h – BFFFh	Chỉ ghi (nCS5)

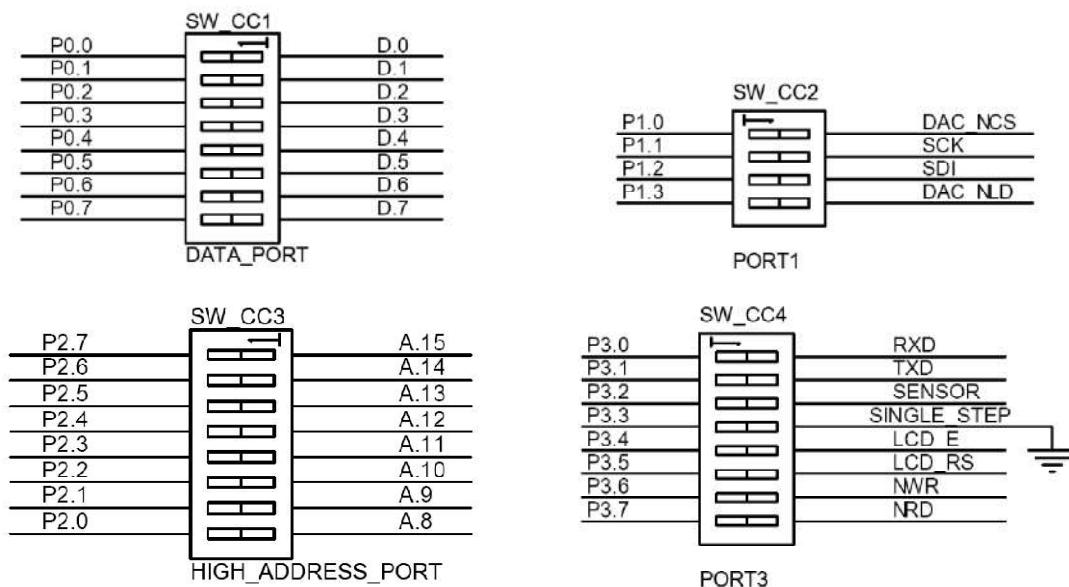
Bảng 2 Bản đồ bộ nhớ của RAM và các ngoại vi

1.3 SỬ DỤNG KIT THÍ NGHIỆM

1.3.1 Cấu hình kit thí nghiệm và kết nối máy tính

1.3.1.1 Cài đặt các DIP-SW8 và jumper cấu hình:

- Bật ON tất cả các SW của 2 DIP-SW8 cấu hình SW_CC1 và SW_CC3. Điều này cho phép 8051 hoạt động trong **chế độ 3 bus** (bus dữ liệu, bus địa chỉ, và bus điều khiển). Trong chế độ 3 bus, Port0 đóng vai trò là 8-bit dữ liệu đồng thời là 8-bit địa chỉ thấp, Port2 đóng vai trò là 8-bit địa chỉ cao tạo nên bus địa chỉ 16-bit.



Hình 5 Sơ đồ kết nối khối DIP-SW8 cấu hình

- Bật ON 4 SW thứ 0, 1, và 6, 7 của DIP-SW8 cấu hình SW_CC4. Điều này cho phép dùng các bit P3.0, P3.1 làm tín hiệu giao tiếp nối tiếp với máy tính qua cổng COM (P3.0 = RXD, P3.1 = TXD), và các bit P3.6, P3.7 được dùng là các tín hiệu điều khiển trong **chế độ 3 bus** (P3.6 = nWR, P3.7 = nRD). Bảng sau chỉ ra vị trí mặc định của các switch trên DIP-SW8 cấu hình. Trên thực tế thí nghiệm, tùy theo ngoại vi nào được sử dụng mà người lập trình cần bật ON các switch của các ngoại vi tương ứng.

Port	Vị trí mặc định	Chức năng
P0	ON	Bus dữ liệu D0-D7 Byte thấp của bus địa chỉ A0-A7

HƯỚNG DẪN THÍ NGHIỆM VI XỬ LÝ

P2	ON	Byte cao của bus địa chỉ A8-A15
P1.0	OFF	/CS của DAC
P1.1	OFF	SCK của giao tiếp SPI
P1.2	OFF	SDI của giao tiếp SPI
P1.3	OFF	/LD của DAC
P3.0	ON	RXD
P3.1	ON	TXD
P3.2	OFF	Tín hiệu giao tiếp 1-Wire bus Ngõ ra của DS18S20
P3.3	OFF	Chọn chế độ Single-Step
P3.4	ON	Tín hiệu Enable của LCD
P3.5	ON	Tín hiệu RS của LCD
P3.6	ON	/WR
P3.7	ON	/RD

Bảng 3 Cài đặt DIP-SW8 cấu hình để hoạt động trong chế độ 3 bus

1.3.1.2 Kết nối bộ thí nghiệm với máy tính.

Có hai cách để kết nối bộ thí nghiệm với máy tính.

- Kết nối cổng USB của kit thí nghiệm với cổng USB của máy tính
- Kết nối cổng COM của kit thí nghiệm với cổng COM của máy tính hoặc với cổng USB của máy tính thông qua một cáp chuyển USB sang serial.
- Sau khi kết nối kit thí nghiệm với máy tính, ta có thể giao tiếp với kit bằng cổng COM (trong trường hợp kết nối qua cổng USB, trong device manager sẽ hiển thị một cổng COM ảo)
- Trên máy tính, khởi động chương trình Hercules, chọn tab serial và cài đặt các thông số như sau: baud 19200, data size 8, parity None, Hand shake OFF. Chương trình này sẽ giúp người sử dụng giao tiếp với kit thí nghiệm thông qua phần mềm monitor MON51 đã được tích hợp sẵn.
- Cấp nguồn cho bộ thí nghiệm.

- Bắt đầu tiên trình thí nghiệm.

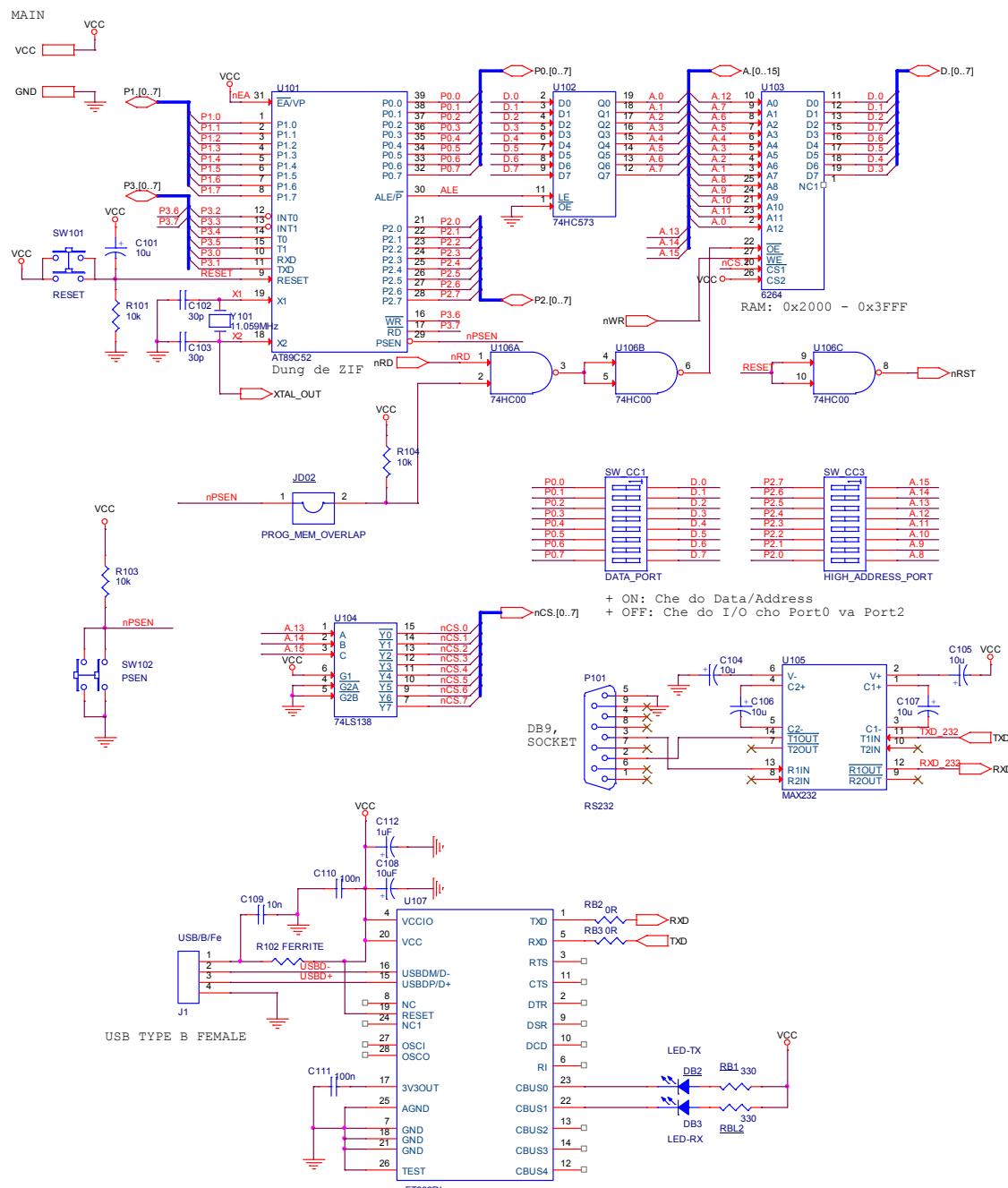
1.4 HOẠT ĐỘNG CỦA KIT THÍ NGHIỆM

Sơ đồ khối CPU của kit thí nghiệm như hình vẽ. CPU được sử dụng là AT89S52, là vi điều khiển trong họ 8051 có dung lượng bộ nhớ trong là 8K.

Chương trình MON-51 được nạp sẵn ở bộ nhớ flash bên trong CPU, bắt đầu từ địa chỉ 0000H. Khi reset chương trình MON-51 sẽ chạy, và giao tiếp máy tính thông qua cổng nối tiếp. Mã máy sau khi biên dịch sẽ được truyền từ máy tính xuống thông qua cổng COM giao tiếp với cổng nối tiếp của 8051, và được chương trình MON-51 ghi vào RAM bắt đầu từ địa chỉ 2000H. Sau khi kết thúc quá trình ghi chương trình vào RAM, người học sẽ ra lệnh cho chương trình monitor thực hiện một lệnh nhảy đến địa chỉ 2000H.

Vì bộ nhớ chương trình nội của AT89S52 là 8K, vì vậy khi thực hiện lệnh nhảy đến 2000H CPU sẽ bắt đầu đọc chương trình ngoài, khi đó tín hiệu PSEN sẽ tích cực. Trên kit thí nghiệm, vùng nhớ dữ liệu và vùng nhớ chương trình tại tầm địa chỉ 2000H-3FFFH được xếp chồng lên nhau bằng cách AND hai tín hiệu PSEN và RD lại với nhau. Vì vậy, CPU sẽ có thể thực thi được chương trình đã được ghi vào RAM trước đó.

HƯỚNG DẪN THÍ NGHIỆM VI XỬ LÝ



Hình 6 Sơ đồ khối CPU-RAM-giải mã địa chỉ-giao tiếp máy tính

1.5 VIẾT CHƯƠNG TRÌNH ASSEMBLY VỚI KEIL UVISION 4

Các hệ thống vi xử lý hoặc vi điều khiển đều cần có một phần mềm (chương trình) để điều khiển hoạt động của nó. Chương trình này được giữ trong bộ nhớ chương trình (program memory) của MCU. Ở cấp thấp nhất, chương trình trong hệ thống là các bit nhị phân thường được gọi là mã máy.

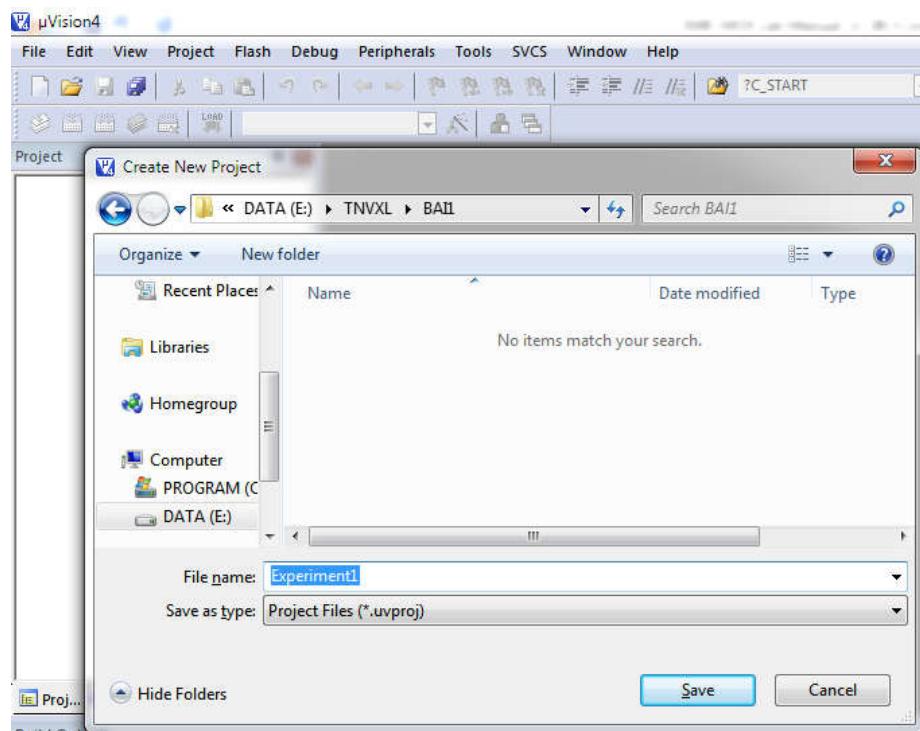
HƯỚNG DẪN THÍ NGHIỆM VI XỬ LÝ

Tuy nhiên, người lập trình rất khó để thao tác với các bit nhị phân. Trong thực tế, các chương trình sẽ được viết trên máy tính bằng hợp ngữ (assembly) hoặc các ngôn ngữ cấp cao khác như C/C++, Basic,... Các chương trình này sẽ cần phải qua bước biên dịch, liên kết để chuyển sang dạng mã máy phù hợp với loại MCU đang dùng. Công cụ để thực hiện các bước này được gọi là chương trình dịch hợp ngữ (assembler), chương trình biên dịch (compiler), và chương trình liên kết (linker). Mỗi loại MCU thường có một chương trình dịch hợp ngữ của riêng nó.

Trong tài liệu thí nghiệm này, người lập trình có thể sử dụng chương trình Keil uVision 4, với trình biên dịch C51. Bản dùng thử có thể được tải tại www.keil.com, cho phép biên dịch các chương trình assembly và C với giới hạn kích thước chương trình là 2KB. Ngoài ra cũng có thể dùng chương trình biên dịch miễn phí SDCC (tại <http://sdcc.sourceforge.net/>). Đây cũng là một bộ công cụ rất hữu ích cho người lập trình.

Để tạo một project với Keil uVision 4 ta theo các bước sau:

- Khởi động chương trình Keil uVision 4
- Chọn Project-New Project. Chọn thư mục phù hợp và gõ tên project vào cửa sổ Create New Project. Chọn Save như ở Hình 7.

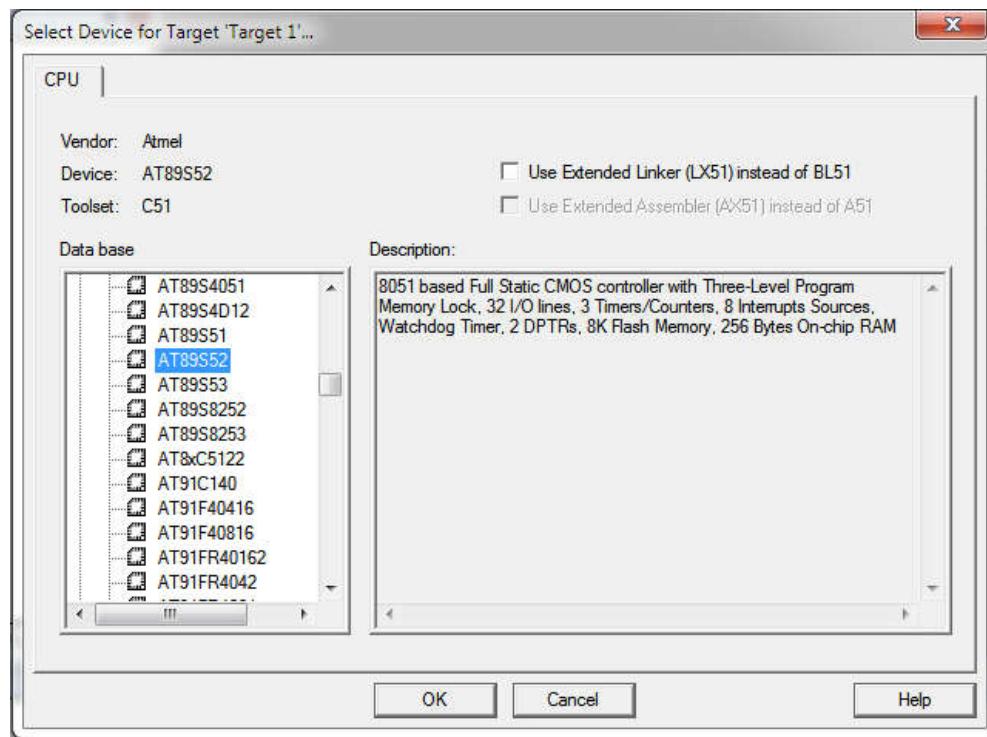


Hình 7 Tạo project mới với uVision 4

HƯỚNG DẪN THÍ NGHIỆM VI XỬ LÝ

Trong cửa sổ Select Device, chọn Atmel-AT89S52. Đây là CPU sử dụng trên kit thí nghiệm.

Click OK.



Hình 8 Chọn vi điều khiển cho project

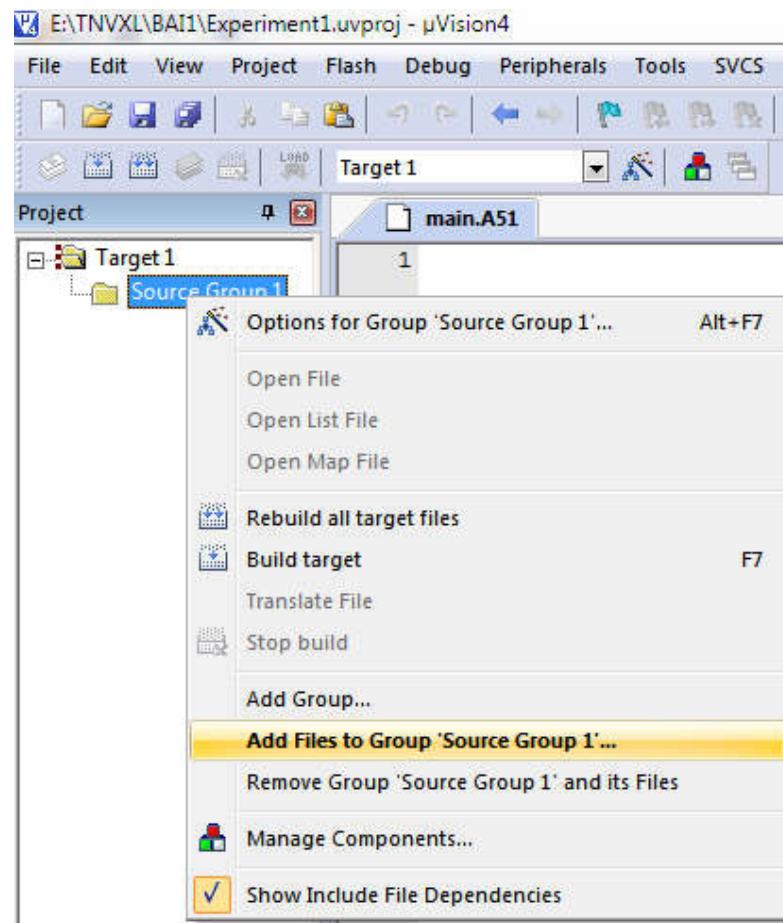
Khi chương trình hiện cửa sổ hỏi: “Copy ‘STARTUP.A51’ to project folder and Add File To Project”, chọn NO

Chọn File-New để tạo một file text mới.

Chọn File-Save để lưu file này với tên file phù hợp và đuôi là .A51

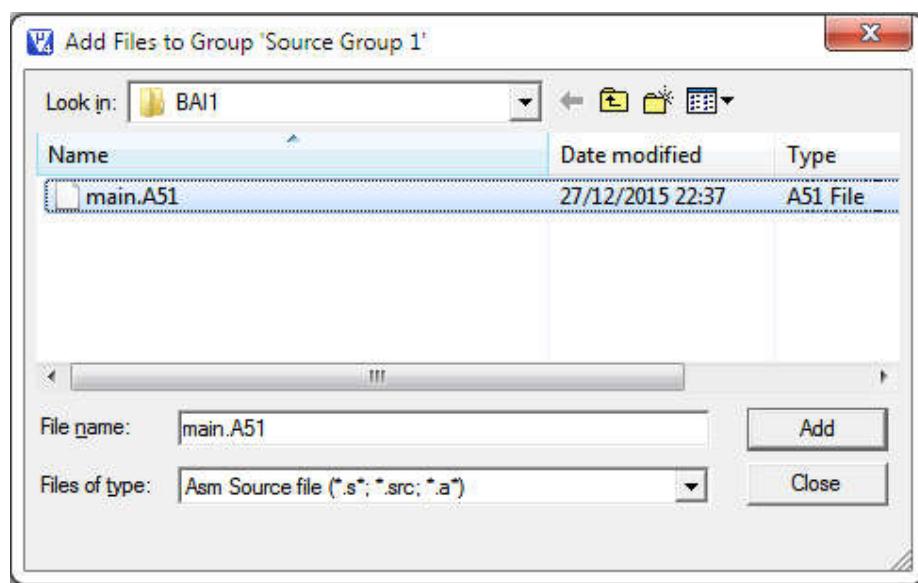
Click chuột phải vào tab ‘Source Group 1’ chọn Add files to Group ‘Source Group 1’.

HƯỚNG DẪN THÍ NGHIỆM VI XỬ LÝ



Hình 9 Thêm file vào project

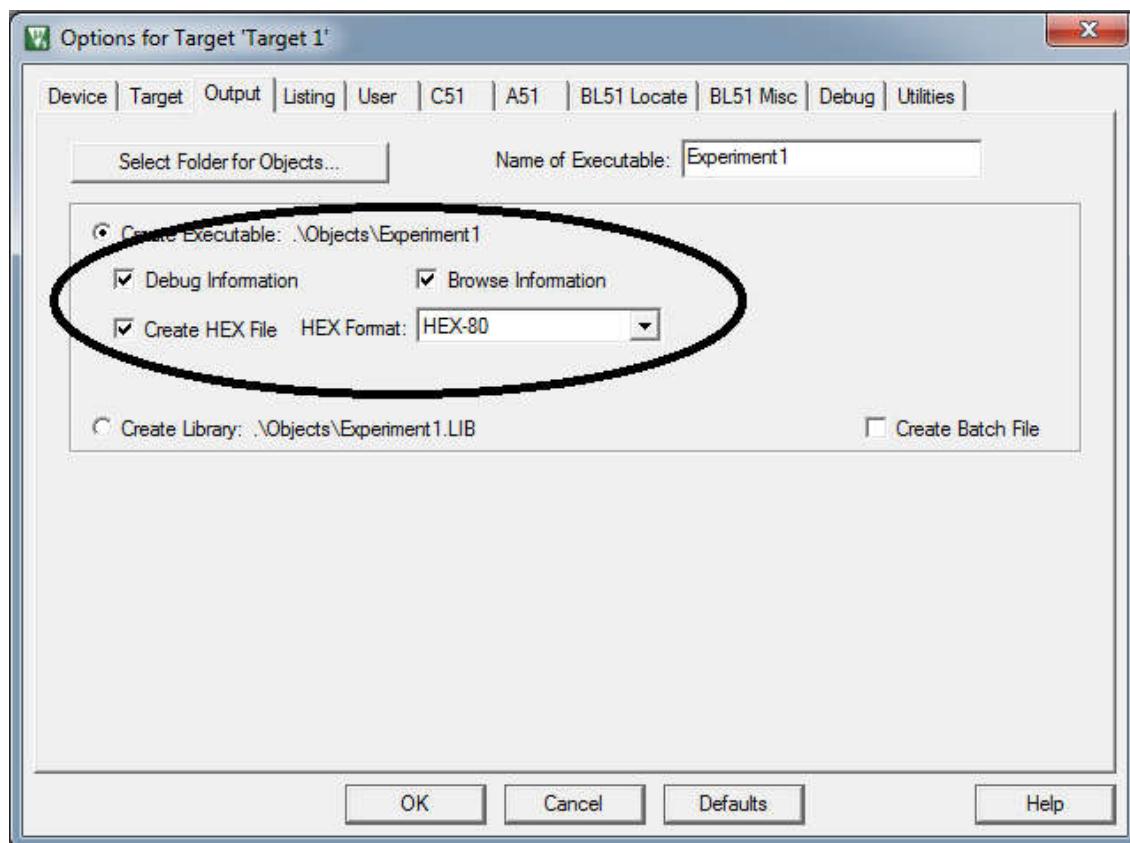
Chọn Files of Type là Asm Source file. Chọn file vừa tạo và click Add



Hình 10 Chọn file assembly để thêm vào project

HƯỚNG DẪN THÍ NGHIỆM VI XỬ LÝ

Chọn Project-Option for Target ‘Target 1’, chọn Tab Output, chọn Create Hex File.



Hình 11 Cấu hình chương trình biên dịch để tạo file hex

Giả sử chương trình được viết có tên **main.A51**. Kit thí nghiệm sử dụng chương trình monitor để download mã nguồn chương trình xuống địa chỉ 2000H, vì vậy chương trình người sử dụng phải nằm trong vùng địa chỉ từ 2000h đến 3FFFh để có thể chạy được trên kit. Chính vì vậy, người sử dụng phải dùng dẫn xuất *ORG* tại đầu chương trình.

```
ORG      2000h
; phần thân chương trình.
END
```

Hoặc sử dụng mẫu sau cho chương trình

```
; +-----+
; Constant declares
; use EQU directive to define constants
; +-----+
```

HƯỚNG DẪN THÍ NGHIỆM VI XỬ LÝ

```
; +-----+
; User DATA memory is internal data memory
; use DS directive to define variables
; +-----+
        DSEG AT 30H

; +-----+
; User PROGRAM memory at 2000h
; +-----+
        CSEG AT 2000h
        jmp start

; +-----+
; Interrupt Vector table
; Write from here
; +-----+
; +-----+
; Main program
; +-----+
start:
; +-----+
; Other defines
; +-----+
        end
```

HƯỚNG DẪN THÍ NGHIỆM VI XỬ LÝ

The screenshot shows the Keil uVision4 IDE interface. The top menu bar includes File, Edit, View, Project, Flash, Debug, Peripherals, Tools, and SVCS. Below the menu is a toolbar with various icons. The main window is divided into several panes: a 'Project' pane on the left showing a file tree with 'Build (F7)' selected; a 'Target 1' pane above the code editor; a 'main.A51' code editor pane containing assembly code:

```
1 ORG 2000H
2 MAIN:
3 CPL P1.0
4 JMP MAIN
5 END
```

Below the code editor is a 'Build Output' pane displaying the results of the build process:

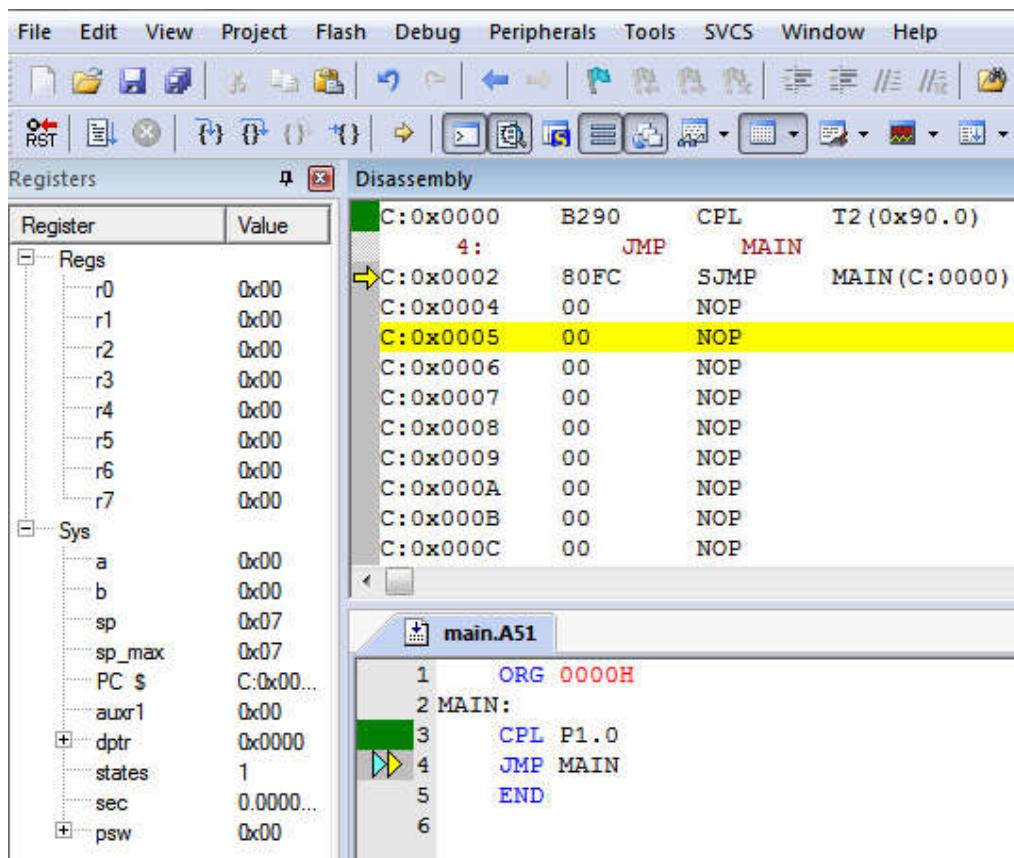
```
Build target 'Target 1'
linking...
Program Size: data=8.0 xdata=0 code=8196
creating hex file from "Experiment1"...
"Experiment1" - 0 Error(s), 0 Warning(s).
```

Hình 12 Biên soạn và biên dịch chương trình

Chọn Project-Build Target. Nếu chương trình không có lỗi, chương trình sẽ biên dịch thành công với số lỗi (error) là 0, đồng thời file kết quả biên dịch sẽ được tạo ra với đuôi .hex.

Người lập trình có thể mô phỏng chương trình bằng cách sử dụng simulator có sẵn của Keil uVision như ở **Hình 13**, hoặc sử dụng chương trình Proteus để nạp file hex tạo ra khi biên dịch vào thiết kế để thử nghiệm trước. Trong trường hợp này, người lập trình cấu hình cho chương trình bắt đầu từ địa chỉ 0 bằng cách sử dụng chỉ dẫn ORG 0000H ở đầu chương trình.

HƯỚNG DẪN THÍ NGHIỆM VI XỬ LÝ



Hình 13 Mô phỏng sử dụng uVision

1.6 LẬP TRÌNH C VỚI UVISION 4

Trong lập trình cho vi điều khiển, hợp ngữ thường được sử dụng khi cần viết những chương trình có kích thước nhỏ, hoặc những đoạn chương trình đòi hỏi tốc độ thực thi nhanh, có thể tính toán chính xác thời gian thực thi.

Nhược điểm lớn của chương trình hợp ngữ là chúng chỉ có thể sử dụng cho đúng một loại CPU riêng biệt. Ví dụ một chương trình hợp ngữ cho 8051 không thể sử dụng cho PIC được, người lập trình phải mất thời gian học tập lệnh của vi điều khiển mới và chuyển đổi chương trình cũ qua. Mặt khác, khi cần viết những chương trình phức tạp, đặc biệt trong trường hợp đòi hỏi nhiều người cùng tham gia viết chương trình, lập trình hoàn toàn bằng hợp ngữ là rất khó khăn.

Với sự phát triển của công nghệ vi mạch, ngày càng nhiều các họ vi điều khiển khác nhau ra đời, với dung lượng bộ nhớ lớn. Do đó, vấn đề tối ưu hóa kích thước chương trình trở nên không còn quá trọng yếu. Thay vào đó, người lập trình đòi hỏi phải có khả năng học lập

HƯỚNG DẪN THÍ NGHIỆM VI XỬ LÝ

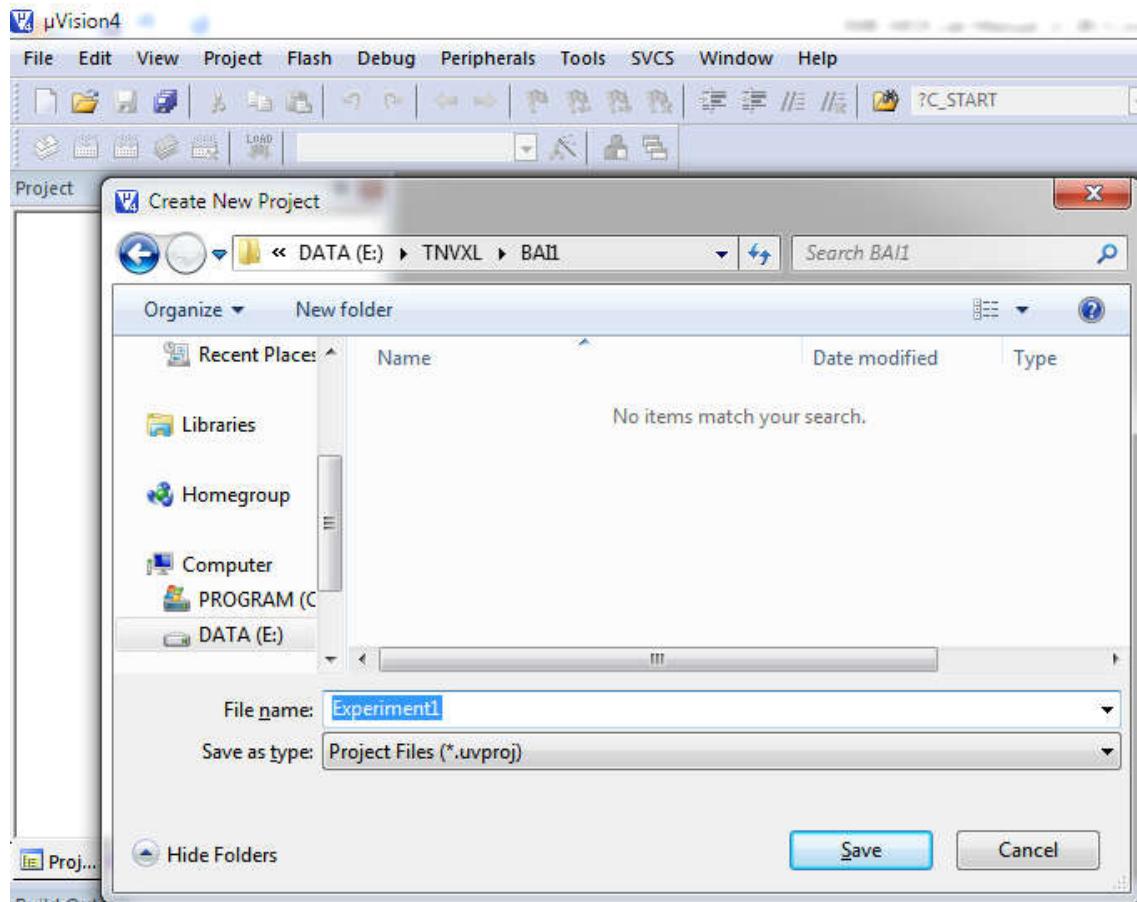
trình cho một vi điều khiển mới một cách nhanh chóng, có khả năng viết các chương trình phức tạp, và những mã nguồn cũ có thể dễ dàng sử dụng lại cho các họ vi điều khiển mới.

Ngôn ngữ C, với các đặc điểm như là ngôn ngữ có cấu trúc, có khả năng tương tác phần cứng cao, mềm dẻo và có cộng đồng sử dụng rộng lớn, là ngôn ngữ thích hợp để lập trình cho các vi điều khiển.

1.6.1 Tạo project sử dụng ngôn ngữ C

Để tạo một project với Keil uVision 4 ta theo các bước sau:

- Khởi động chương trình Keil uVision 4
- Chọn Project-New Project. Chọn thư mục phù hợp và gõ tên project vào cửa sổ Create New Project. Chọn Save

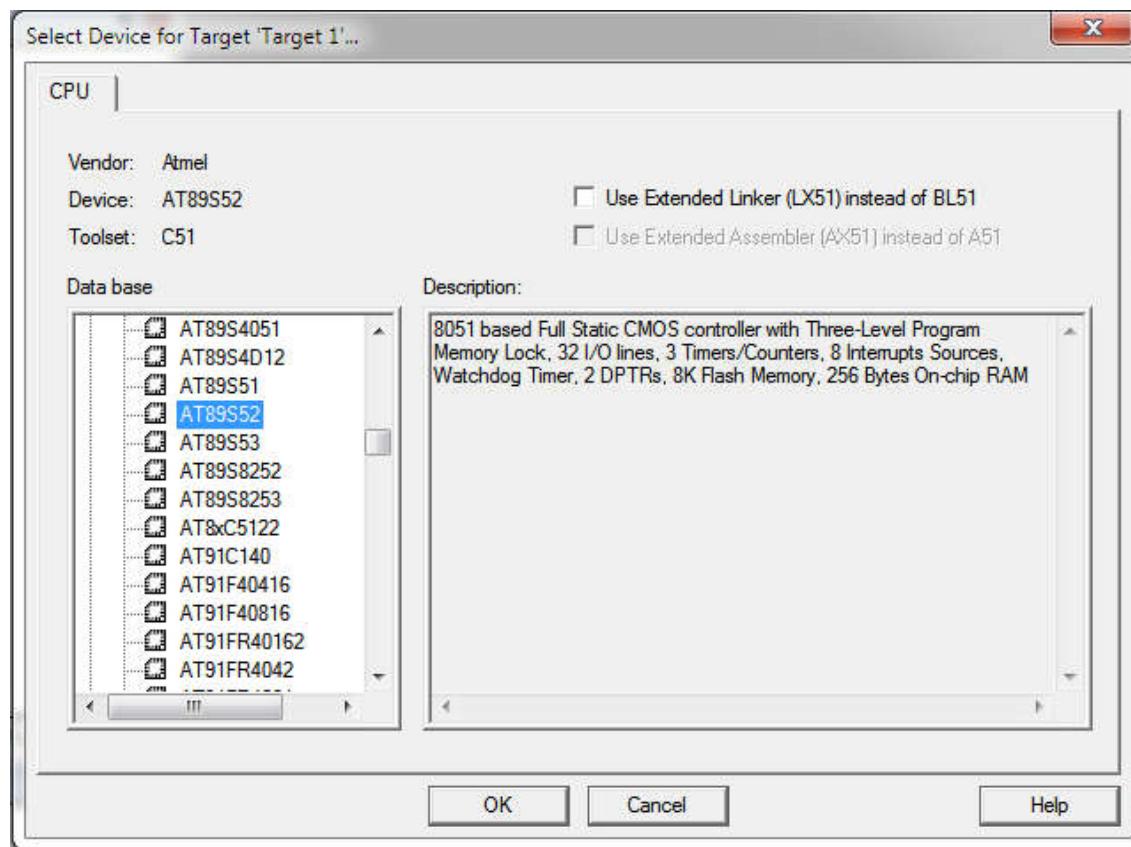


Hình 14 Tạo project với uVision 4

Trong cửa sổ Select Device, chọn Atmel-AT89S52. Đây là CPU sử dụng trên kit thí nghiệm. Click OK.

HƯỚNG DẪN THÍ NGHIỆM VI XỬ LÝ

Khi chương trình hiện cửa sổ hỏi: “Copy ‘STARTUP.A51’ to project folder and Add File To Project”, chọn YES



Hình 15 Chọn vi điều khiển

Mở file STARTUP.A51, tìm đến dòng " CSEG AT 0", sửa lại thành CSEG AT 2000H. Với cách làm này chương trình sẽ được biên dịch bắt đầu từ 2000H

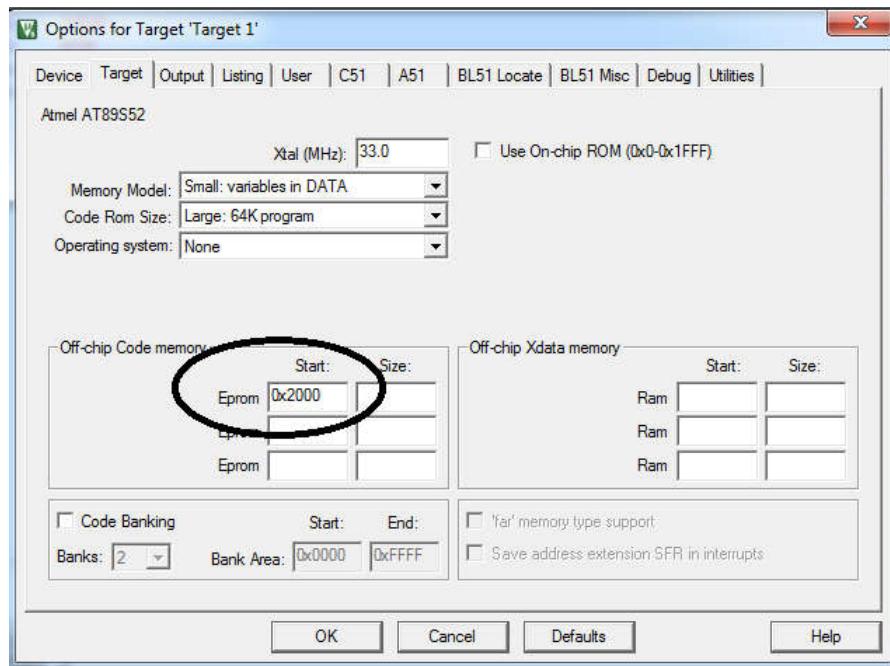
HƯỚNG DẪN THÍ NGHIỆM VI XỬ LÝ

The screenshot shows the assembly code for the startup program. The code defines various SFR symbols (ACC, B, SP, DPL, DPH) with initial values. It then defines the ?C_C51STARTUP segment as CODE and the ?STACK segment as IDATA. The entry point ?C_STARTUP is defined as LJMP to STARTUP1. The code concludes with an RSEG directive back to the ?C_C51STARTUP segment.

```
106 ; Standard SFR Symbols
107 ACC DATA 0EOH
108 B DATA 0FOH
109 SP DATA 81H
110 DPL DATA 82H
111 DPH DATA 83H
112
113 NAME ?C_STARTUP
114
115
116 ?C_C51STARTUP SEGMENT CODE
117 ?STACK SEGMENT IDATA
118
119 RSEG ?STACK
120 DS 1
121
122 EXTRN CODE (?C_START)
123 PUBLIC ?C_STARTUP
124
125 CSEG AT 2000H
126 ?C_STARTUP: LJMP STARTUP1
127
128 RSEG ?C_C51STARTUP
129
```

Hình 16 Chính lại địa chỉ bắt đầu chương trình trong startup code

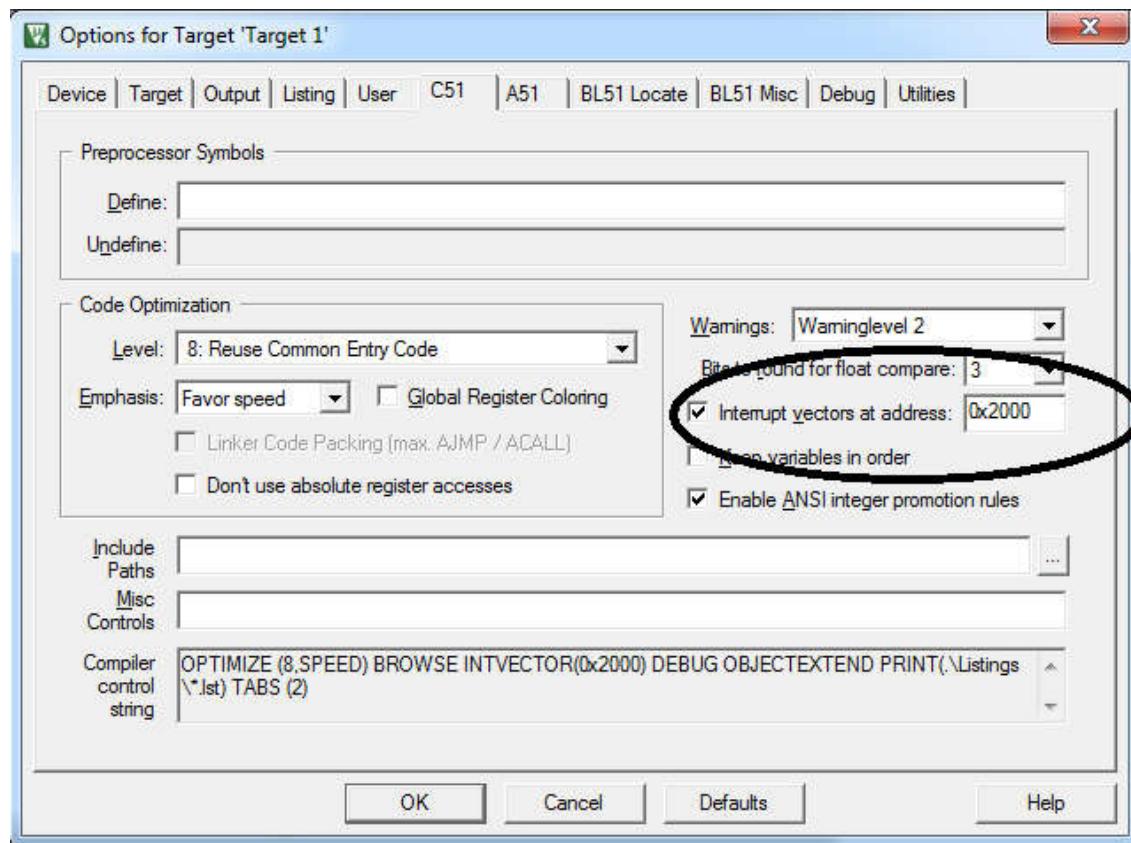
Chọn Project-Option for Target, chọn tab Target, trong cửa sổ Off-chip Code Memory, gõ số 0x2000 vào ô Start



Hình 17 Cấu hình địa chỉ bắt đầu chương trình

HƯỚNG DẪN THÍ NGHIỆM VI XỬ LÝ

Chọn tab C51, kích chọn "Interrupt vector table at address" và gõ giá trị 0x2000 vào textbox bên cạnh



Hình 18 Chính địa chỉ cho bảng vector ngắt

Lưu ý:

Ba bước trên là để trình biên dịch tạo ra file thực thi bắt đầu tại địa chỉ 2000H, để có thể thực hiện được chương trình trên kit thí nghiệm.

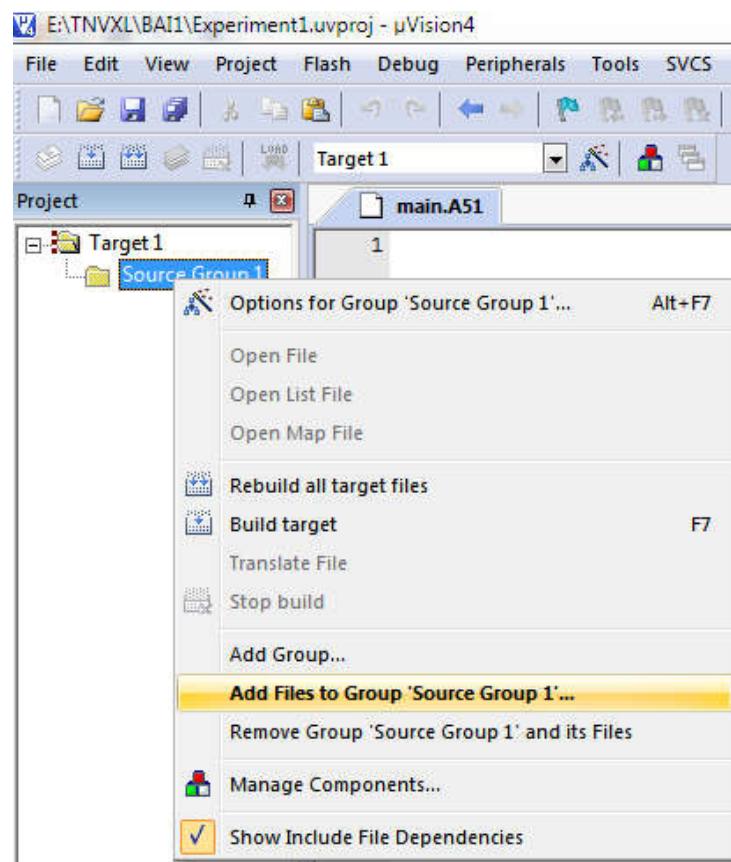
Nếu người lập trình muốn tạo ra file hex để có thể mô phỏng trên Proteus hoặc chạy trên phần cứng của mình từ địa chỉ đầu là 0, thì không cần ba bước trên.

Chọn File-New để tạo một file text mới.

Chọn File-Save để lưu file này với tên file phù hợp và đuôi là .c, ví dụ main.c

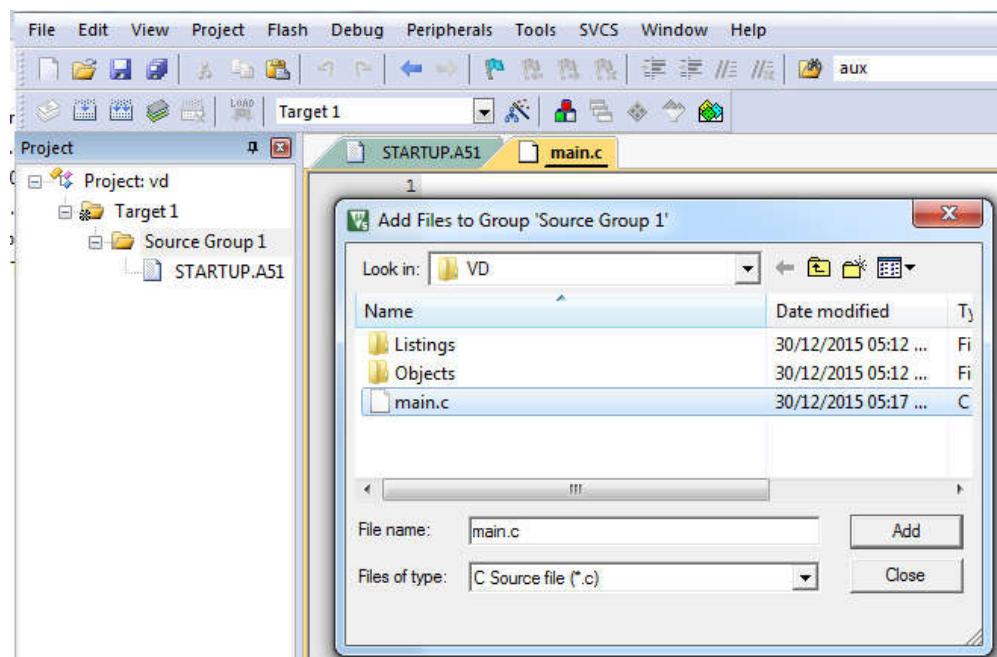
Click chuột phải vào tab 'Source Group 1' chọn Add files to Group 'Source Group 1'.

HƯỚNG DẪN THÍ NGHIỆM VI XỬ LÝ



Hình 19 Thêm file nguồn vào project

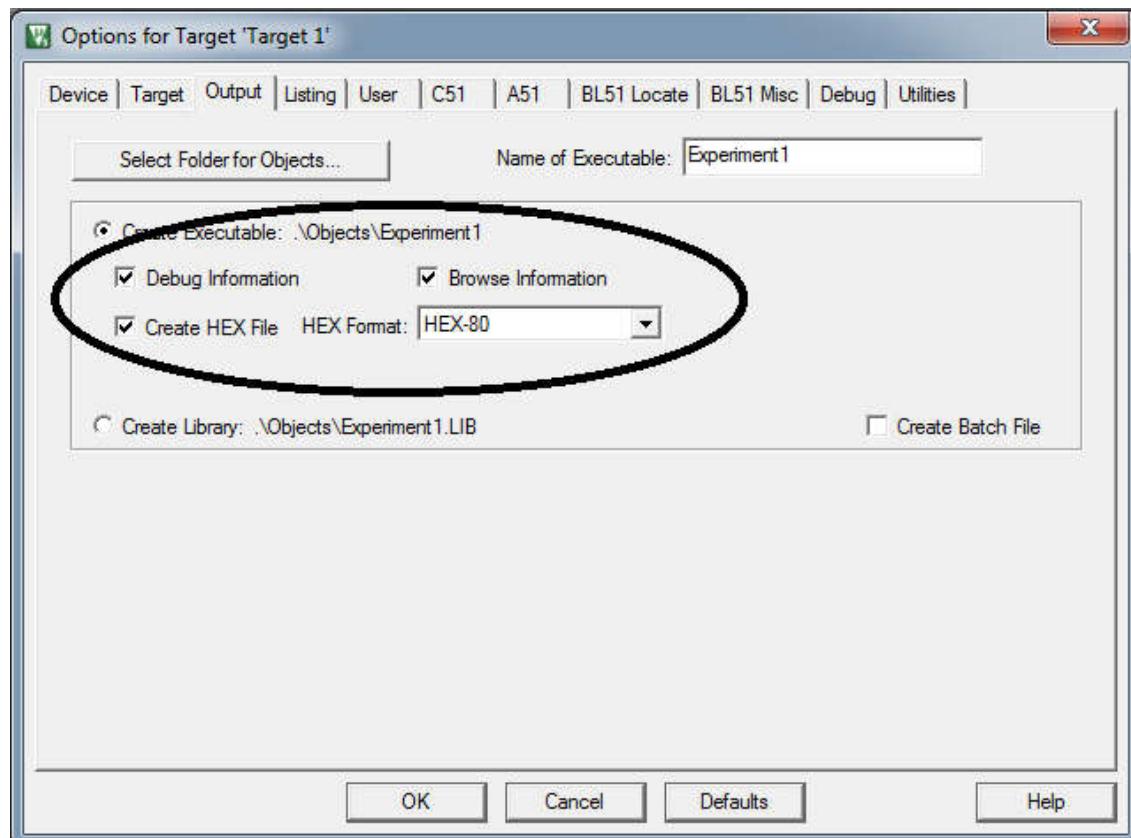
Chọn Files of Type là C Source file. Chọn file vừa tạo và click Add



Hình 20 Thêm file .c vào project

HƯỚNG DẪN THÍ NGHIỆM VI XỬ LÝ

Chọn Project-Option for Target ‘Target 1’, chọn Tab Output, chọn Create Hex File như trong Hình 21. Lưu ý là tên hexfile được cho trong ô Name of Executable, ta có thể sửa tên file này theo ý muốn.



Hình 21 Cấu hình chương trình biên dịch để tạo file hex

Sau khi tạo project, ta có thể bắt đầu viết chương trình trong file main.c. Một chương trình C cho 8051 có cấu trúc cơ bản như Hình 22.

Đầu file main.c là các khai báo #include để báo cho trình xử lý tiền biên dịch thêm các file header (.h) vào chương trình. Sau đó là các khai báo macro, khai báo biến toàn cục, các mô tả hàm.

Mỗi project phải chứa đúng 1 hàm **main**. Sau khi reset, chương trình sẽ thực hiện các hàm khởi động trong module startup và gọi hàm **main**.

Các định nghĩa hàm được đặt ở sau hàm **main**.

Sau khi hoàn tất chương trình, ta biên dịch bằng cách chọn Project-Build Target hoặc nhấn phím tắt F7. Nếu chương trình biên dịch thành công, file Experiment1.hex sẽ được tạo ra.

HƯỚNG DẪN THÍ NGHIỆM VI XỬ LÝ

The screenshot shows the µVision IDE interface. The title bar reads "C:\Users\S-TEK\Desktop\TNVXL TEST\VD\vd.uvproj - µVision". The menu bar includes File, Edit, View, Project, Flash, Debug, Peripherals, Tools, SVCS, Window, and Help. The toolbar has various icons for file operations like Open, Save, and Build. A tab bar at the top shows "Target 1" with three tabs: "STARTUP.A51" (selected), "main.c", and "AT89X51.H". The main workspace displays the "main.c" code:

```
1 /*
2 Declare all header files here
3 */
4 #include <AT89X51.H>
5 /*
6 Declare all macro heres
7 */
8 #define LED P1_0
9 /*
10 Declare all global variable here
11 */
12 char DisplayVal;
13 /*
14 Declare all function prototypes here
15 */
16 void delay( unsigned int us);
17
18 // Main function. All project has 1 main function
19 void main(void)
20 {
21     DisplayVal = 0;
22     while (1)
23     {
24         P1 = DisplayVal;
25         delay (10000);
26         DisplayVal++;
27     }
28 }
29 void delay( unsigned int us)
30 {
31     unsigned int i;
32     for (i=us;i>0;i--)
33 }
```

Hình 22 Ví dụ về một chương trình C cho 8051

1.7 DOWNLOAD CHƯƠNG TRÌNH XUỐNG KIT VÀ THỰC THI

Trong thực tế, với file kết quả biên dịch **.hex**, người lập trình cần sử dụng các **bộ lập trình** (programmer) để có thể ghi chương trình này vào bộ nhớ chương trình của hệ thống. Bộ nhớ chương trình thường được tích hợp sẵn trong MCU hoặc nằm trong ROM của hệ thống. Một số bộ lập trình thông dụng có thể tìm được là LabTool của Advantech, hoặc SuperPro của Xeltek. Ngoài ra cũng có một số bộ lập trình đơn giản hơn, phù hợp với nhu cầu học tập của sinh viên.

HƯỚNG DẪN THÍ NGHIỆM VI XỬ LÝ

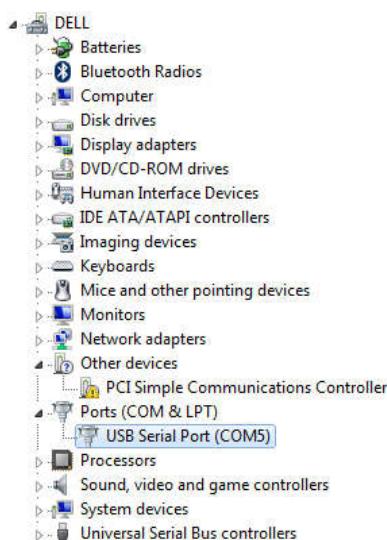
Tuy nhiên, với kit thí nghiệm vi xử lý này, người lập trình không cần có bộ lập trình độc lập vì trên kit đã được tích hợp sẵn chương trình **monitor**. Đây là một đoạn chương trình nhỏ đã được ghi sẵn vào hệ thống, cho phép người sử dụng dùng máy tính để giao tiếp với kit thí nghiệm thông qua cổng COM. Người sử dụng có thể ra lệnh cho monitor ghi chương trình của mình vào bộ nhớ chương trình và sau đó thực thi nó. Bộ nhớ chương trình trong trường hợp này thường là một bộ nhớ có khả năng ghi đọc bình thường ví dụ như RAM. Monitor sẽ ghi chương trình của người lập trình vào trong RAM như dữ liệu, sau đó sẽ ra lệnh thực thi đoạn mã này như bộ nhớ chương trình. Kỹ thuật này được gọi là **chồng phủ vùng nhớ** (overlay), trong đó địa chỉ của bộ nhớ chương trình và bộ nhớ dữ liệu được thiết kế trùng nhau. Ngoài ra monitor còn có nhiều lệnh khác phục vụ trong việc gỡ rối chương trình.

Để có thể sử dụng nhanh MON51 trên kit thí nghiệm người sử dụng cần có một chương trình giao tiếp cổng nối tiếp trên máy tính. Trong tài liệu này người học sử dụng chương trình Hercules, là một chương trình miễn phí có thể tải về tại www.hw-group.com.

Chương trình này cho phép kết nối với cổng COM của máy tính.

- Dùng dây nối cổng COM của máy tính với cổng RS232 trên kit thí nghiệm, hoặc nối cổng USB của máy tính vào cổng USB của kit thí nghiệm.
- Mở nguồn của kit thí nghiệm

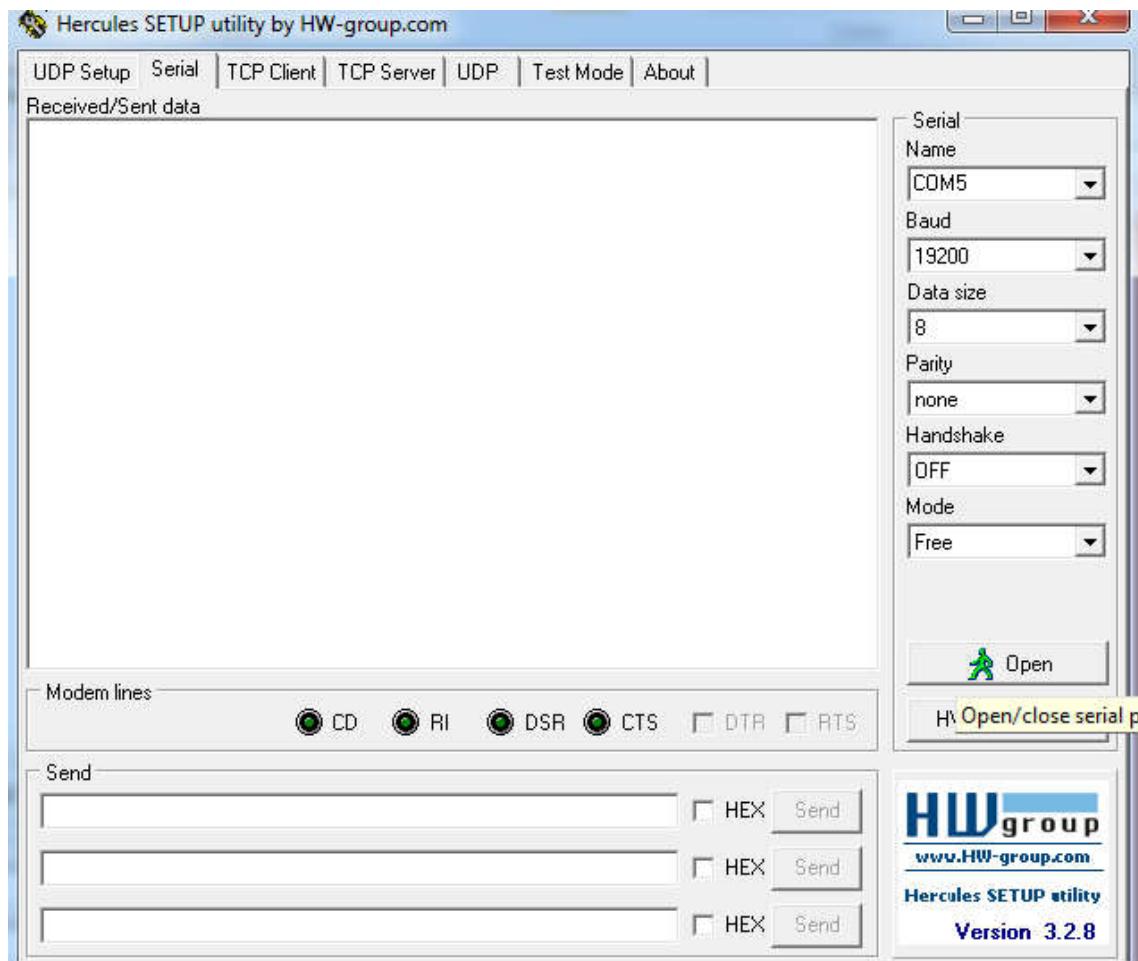
Khi kết nối kit với máy tính qua cổng USB, máy tính sẽ có thể giao tiếp với kit qua một cổng COM ảo. Số hiệu của cổng COM ảo này có thể được xem trên Device Manager



Hình 23 Tìm số hiệu cổng COM

HƯỚNG DẪN THÍ NGHIỆM VI XỬ LÝ

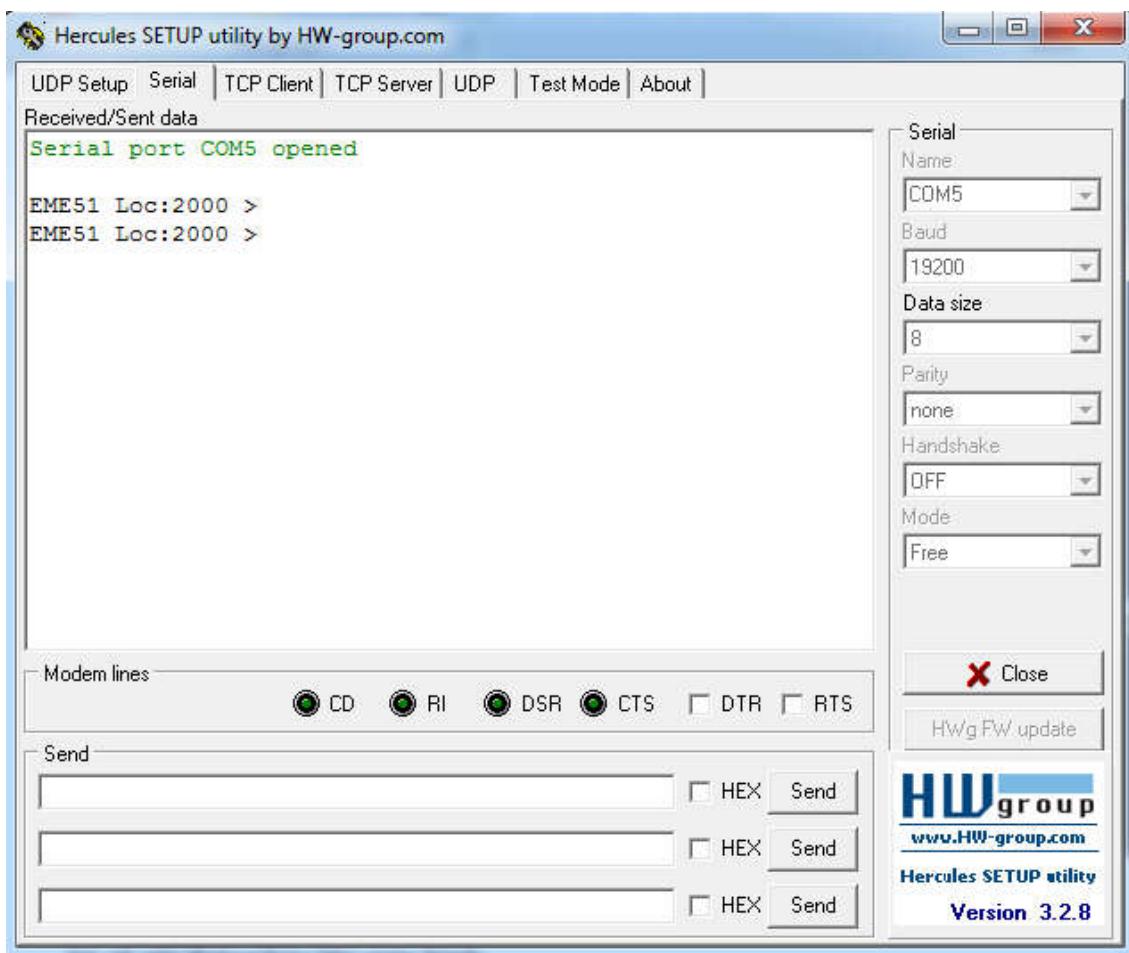
- Kết nối chương trình Hercules với cổng COM, cấu thông số cổng COM là 19200bps, 8 bit, 1 stop bit, không có bắt tay. Sau đó, click Open



Hình 24 Cấu hình cổng COM

- Click chuột trái vào ô Received/Send Dada và nhấn Enter. Chương trình MON 51 sẽ gửi thông báo lên màn hình

HƯỚNG DẪN THÍ NGHIỆM VI XỬ LÝ

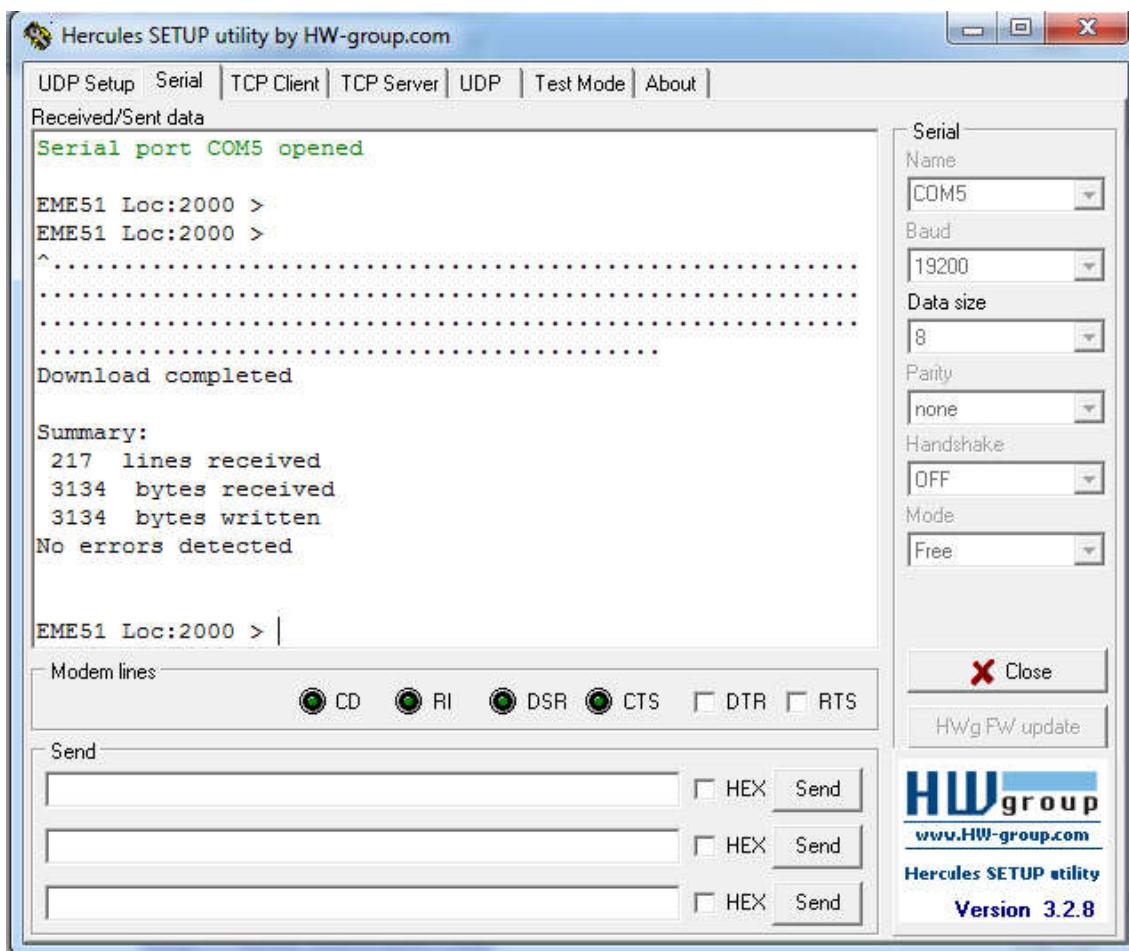


Hình 25 Mở cổng COM

Sau khi đã biên dịch thành công chương trình và nhận được file .hex, người sử dụng cần dùng các thao tác sau để có thể thực thi chương trình.

- Đảm bảo là chương trình thí nghiệm được biên dịch với địa chỉ đầu là từ 2000h đến 3FFFh. Giả sử file *experiment1.hex* được biên dịch với địa chỉ là 2000h.
- Kích chuột phải, chọn Send File - Send File ... hoặc nhấn tổ hợp phím Ctrl-O.
- Di chuyển đến thư mục chứa file .hex cần thử nghiệm và chọn file này.
- Cửa sổ chọn file đóng lại, quá trình truyền file bắt đầu. Khi kết thúc truyền, Hercules sẽ cho thấy số lượng byte truyền được và truyền có thành công hay không.

HƯỚNG DẪN THÍ NGHIỆM VI XỬ LÝ



Hình 26 Download chương trình xuống RAM

- Nếu kết quả truyền đã thành công, người sử dụng có thể gõ phím J trên bàn phím, gõ địa chỉ của chương trình vừa truyền xuống (cụ thể là 2000h) và gõ Enter để thực thi chương trình.
- Để thoát khỏi chương trình người sử dụng và quay trở lại chương trình monitor, nhấn nút reset trên kit thí nghiệm.

CHƯƠNG 2 THÍ NGHIỆM VỚI NÚT NHẤN VÀ LED ĐƠN

2.1 LÝ THUYẾT CƠ BẢN

Trong thực tế, led đơn và nút nhấn là hai phương pháp giao tiếp người sử dụng đơn giản và cũng rất hiệu quả. Người sử dụng có thể dùng nút nhấn tác động vào hệ thống điều khiển để ra lệnh cho hệ thống. Ngược lại, led đơn có thể được dùng để hiển thị trạng thái hoạt động bên trong của hệ thống. Led đơn đặc biệt hữu dụng trong việc giúp người thiết kế gỡ rối chương trình. Ví dụ người lập trình có thể thay đổi tốc độ nhấp nháy của led để chỉ ra các lỗi khác nhau trong chương trình.

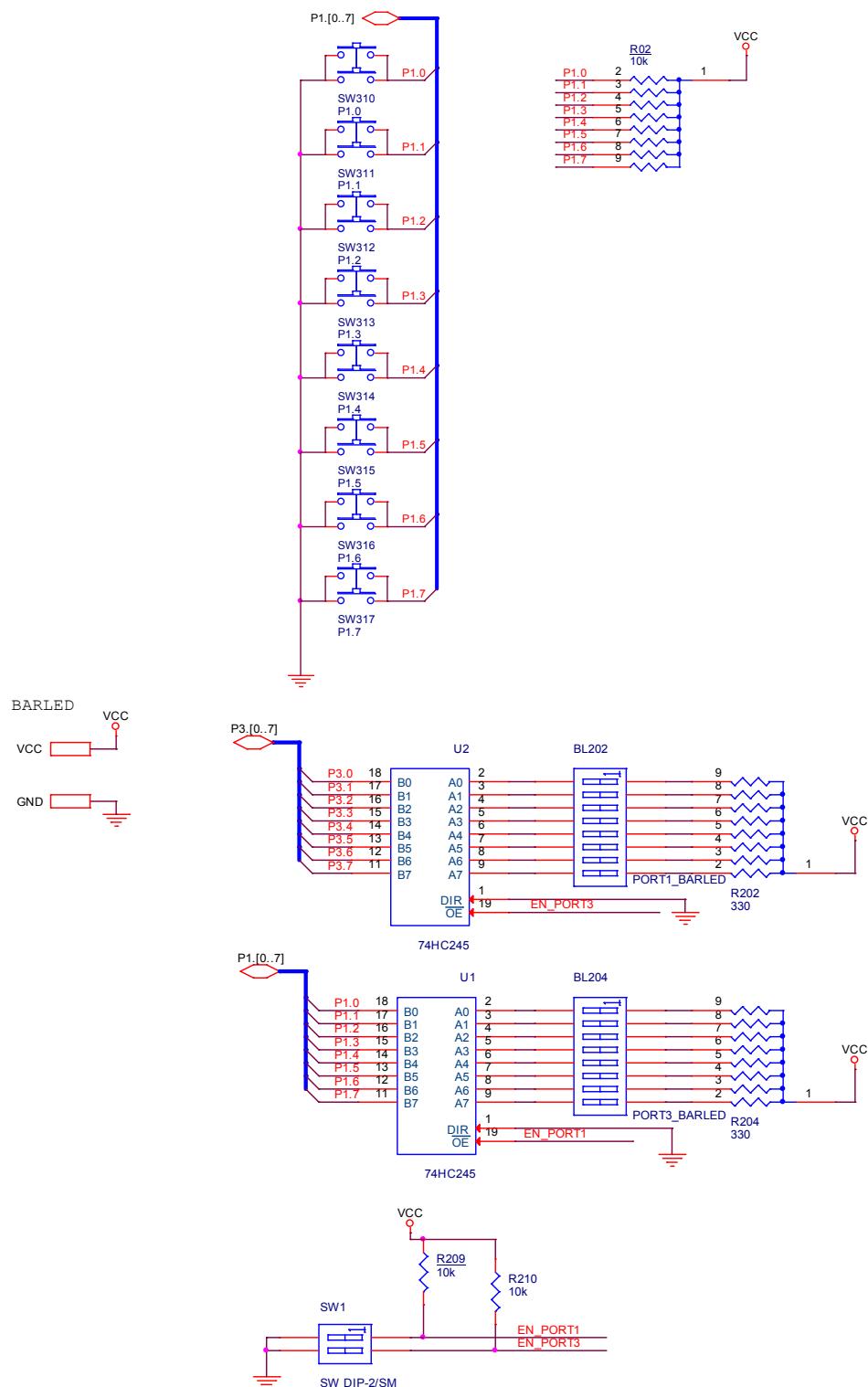
2.2 THIẾT KẾ PHẦN CỨNG

Các nút nhấn và led đơn có thể được nối trực tiếp đến các port I/O trên vi điều khiển như Hình 27. Trên kit thí nghiệm có 8 nút nhấn đơn kết nối đến PORT1 và 2 led thanh (bar led), mỗi bar led được nối đến port 1 và port 3 của 8051 thông qua IC 74HC245.

Lưu ý:

Vì chương trình được download xuống RAM và thực thi trên RAM, do đó khi chạy chương trình vi xử lý làm việc ở chế độ giao tiếp bộ nhớ ngoài. Để giao tiếp bộ nhớ ngoài, hai tín hiệu RD và WR được CPU sử dụng, vì vậy chương trình không thể sử dụng hai chân P3.6 và P3.7 như là I/O port.

HƯỚNG DẪN THÍ NGHIỆM VI XỬ LÝ



Hình 27 Sơ đồ giao tiếp led thanh và nút nhấn đơn

2.3 LẬP TRÌNH TRUY XUẤT I/O PORT

Về mặt lập trình, vì 8051 sử dụng chế độ I/O ánh xạ bộ nhớ nên để truy xuất đến các port của 8051 ở chế độ I/O Port, người lập trình có thể đọc hoặc ghi trực tiếp vào các thanh ghi port trong vùng thanh ghi chức năng đặc biệt (SFR) có địa chỉ từ 80h đến 7Fh. Cụ thể là thanh ghi điều khiển Port0 là 80h, Port1 là 90h, Port2 là 0A0h, và Port3 là 0B0h.

Khi truy xuất đến các thanh ghi này, người lập trình dùng phương pháp định địa chỉ trực tiếp. Như vậy, lệnh nào trong tập lệnh có hỗ trợ phương pháp định địa chỉ trực tiếp thì đều có thể dùng để truy xuất I/O Port.

```
MOV      A, P1          ; đọc P1 vào thanh ghi A  
SETB    P1.1           ; led tại P1.1 tắt  
MOV      C, P1.0        ; đọc trạng thái nút nhấn tại P1.0
```

Ví dụ như để đọc giá trị của chân P1.0 và xuất ra chân P3.0, vì nút nhấn và led đơn được nối trực tiếp đến các bit port để sử dụng chức năng I/O port nên người lập trình chỉ cần truy xuất đến 2 bit P1.0 và P3.0. Giá trị đọc được từ nút nhấn cũng như giá trị xuất ra led chỉ có độ rộng là 1 bit nên người lập trình nên sử dụng cờ C để chứa giá trị này.

```
MOV      C, P1.0        ; đọc nút nhấn  
MOV      P3.0, C         ; xuất ra led
```

2.4 LẬP TRÌNH TẠO ĐỘ TRỄ DÙNG CÂU LỆNH

Trong thực tế, để điều khiển các ngoại vi, các thao tác điều khiển luôn cần phải tuân theo một trình tự nhất định với khoảng thời gian giữa chúng là xác định. Do đó, các chương trình con tạo trễ luôn là một phần rất quan trọng của chương trình điều khiển.

Ví dụ như ta cần bật/tắt 1 LED để báo hiệu là chương trình đang hoạt động. Nếu LED bật/tắt quá nhanh, ta sẽ có cảm giác là LED luôn sáng. Vì vậy, giữa 2 lần sáng/tắt cần có 1 thời gian trễ để mắt người có thể cảm nhận được, thường thời gian này là 1 s.

Có nhiều cách để có thể viết được chương trình con tạo trễ. Phần thí nghiệm này giúp người lập trình nắm được cách thức tạo trễ dùng các câu lệnh tạo thành vòng lặp. Mỗi vi điều khiển luôn sử dụng một tín hiệu xung clock để đồng bộ các hoạt động trong hệ thống (8051 trên kit sử dụng clock với tần số 11.059MHz). Một câu lệnh được thực thi sẽ cần một số xung clock xác định thường được đo bằng chu kỳ máy (một chu kỳ máy của 8051 mất 12 xung clock). Như vậy, một câu lệnh được thực thi sẽ tiêu hao một khoảng thời gian xác định. Ví dụ lệnh

HƯỚNG DẪN THÍ NGHIỆM VI XỬ LÝ

MOV của 8051 có thể mất 1 hoặc 2 chu kỳ máy, tức là 12 hoặc 24 xung clock, nghĩa là khoảng 1us hoặc 2us (bỏ qua sai số).

Trong tập lệnh của vi điều khiển luôn có lệnh *NOP*. Lệnh này thường được dùng chỉ để tiêu tốn 1 chu kỳ máy mà không thực thi thao tác gì cả. Do đó, để tạo ra thời gian trễ ngắn, người lập trình có thể dùng vài lệnh *NOP*. Trong trường hợp thời gian tạo trễ dài, cần nhiều chu kỳ máy, người lập trình có thể dùng vòng lặp để tạo trễ

MOV R7, #n

DJNZ R7, \$

Vòng lặp này sử dụng $(2n+1)$ chu kỳ máy, với n có độ rộng 1 byte nên số chu kỳ máy tối đa là $(2*256+1)=513$ chu kỳ máy. Trong thực tế, người lập trình có thể tính gần đúng là $2n$ chu kỳ máy. Với thời gian dài hơn, có thể lồng nhiều vòng lặp vào nhau

MOV R7, #n

LOOP: MOV R6, #m

DJNZ R6, \$

DJNZ R7, LOOP

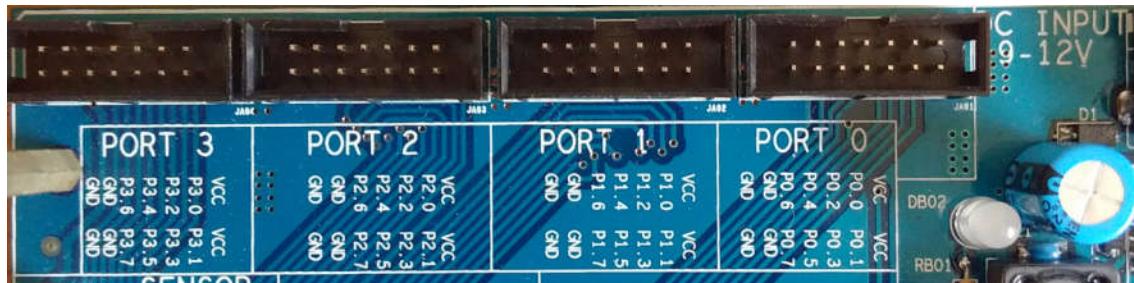
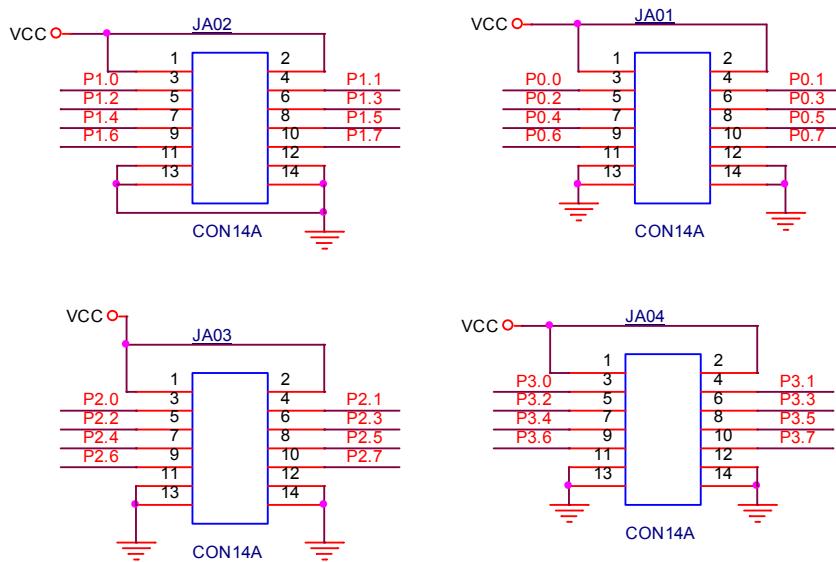
Vòng lặp này mất $(2m+3)*n+1$ chu kỳ máy, và tính gần đúng là $2mn$ chu kỳ. Lưu ý là khi tính gần đúng thì vòng lặp bên trong phải có số lần lặp lớn thì mới đảm bảo được sai số nhỏ.

2.5 HƯỚNG DẪN ĐO THỬ NGHIỆM VỚI OSCILLOSCOPE:

Trên kit có 4 connector nối vào 4 port như Hình 28. Để đo dạng sóng trên từng chân, ta sử dụng oscilloscope và dây nối đôi được cung cấp kèm theo kit thí nghiệm.

Kết nối một dây vào GND, dây còn lại vào chân port cần đo. Sử dụng dây probe của oscilloscope kết nối vào các đầu dây còn lại và chỉnh oscilloscope để quan sát dạng sóng.

HƯỚNG DẪN THÍ NGHIỆM VI XỬ LÝ



Hình 28 Các connector mở rộng

CHƯƠNG 3 THÍ NGHIỆM VỚI TIMER

3.1 LÝ THUYẾT CƠ BẢN

Trong bài thí nghiệm đầu tiên, vòng lặp lệnh được dùng để tiêu hao thời gian của vi điều khiển. Tuy nhiên, cách sử dụng vòng lặp lệnh đòi hỏi nhiều công sức tinh chỉnh để có được độ trễ chính xác. Mặt khác, khi sử dụng vòng lặp lệnh sẽ tiêu tốn thời gian hoạt động của CPU. Để tạo ra các khoảng thời gian chính xác người ta thường dùng ngoại vi timer.

Timer (bộ định giờ) thường được dùng để đo khoảng thời gian. Thực chất Timer là một bộ đếm với xung đếm có tần số cố định và biết trước. Khi xác định được số xung đếm và biết chu kỳ của xung thì có thể tính được thời gian. Timer của 8051 có thể hoạt động ở chế độ 8-bit hoặc 16-bit. Tùy theo số xung đếm cần nhiều hay ít mà có thể cấu hình Timer ở chế độ tương ứng.

3.2 LẬP TRÌNH SỬ DỤNG TIMER

Timer là ngoại vi on-chip của 8051, để cấu hình và điều khiển ngoại vi này, người lập trình có thể truy xuất vào các thanh ghi tương ứng trong vùng SFR (địa chỉ 80h đến FFh). Đôi với Timer đó là các thanh ghi *TMOD*, *THx*, *TLx*, *TCON*. Trong đó, *THx* và *TLx* chứa giá trị hiện thời của bộ đếm xung. Ví dụ, để cấu hình Timer0 hoạt động ở chế độ 8-bit

MOV TMOD, #02h

Để khởi tạo giá trị đầu cho Timer0

MOV TH0, #HIGH(-50000) ; giá trị khởi động là -50000

MOV TL0, #LOW(-50000)

Để chờ hết khoảng thời gian đã định trước, người lập trình có thể quan sát cờ *TFx*. Cờ này sẽ lên 1 mỗi khi Timerx bị tràn, tức là giá trị bộ đếm xung quay trở lại giá trị 0 (zero)

JNB TF0, \$; chờ Timer0 tràn sau 50000 chu kỳ máy

Nếu timer hoạt động ở chế độ 16 bit, sau khi timer tràn ta phải nạp lại giá trị đầu cho timer.

Khi Timer chạy ở chế độ 2, thanh ghi đếm *TLx* sẽ được tự nạp lại khi timer tràn, vì vậy chương trình không cần dừng timer để nạp lại giá trị đầu.

Ví dụ, để cấu hình Timer0 hoạt động ở chế độ 8-bit:

MOV TMOD, #02h

HƯỚNG DẪN THÍ NGHIỆM VI XỬ LÝ

Để khởi tạo giá trị đầu cho Timer0

```
MOV      TH0, #(-50) ; giá trị khởi động là -50  
MOV      TL0, #(-50)
```

Để khởi động timer, set cờ TR

```
SETB    TR0
```

Để chờ hết khoảng thời gian đã định trước, người lập trình có thể quan sát cờ TFx . Cờ này sẽ lên 1 mỗi khi Timerx bị tràn, tức là giá trị bộ đếm xung quay trở lại giá trị 0 (zero)

```
JNB      TF0, $      ; chờ Timer0 tràn sau 50000 chu kỳ máy
```

Để tạo xung tuần hoàn, ta chỉ cần đảo chân port và xóa cờ tràn TF. Lưu ý là timer chạy độc lập với CPU, có nghĩa là trong khi CPU đang thực thi câu lệnh, timer vẫn chạy

Xung có thể được tạo ra như sau:

LOOP:

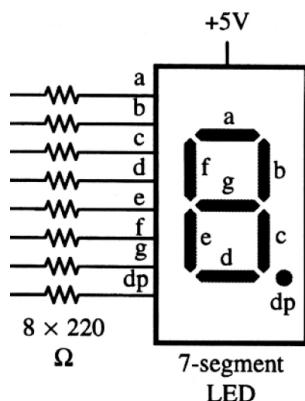
```
JNB      TF0,$  
CPL      P1.0  
CLR      TF0  
SJMP     LOOP
```

CHƯƠNG 4 THÍ NGHIỆM HIỂN THỊ DÙNG LED 7 ĐOẠN

4.1 LÝ THUYẾT CƠ BẢN

Led 7 đoạn thường được dùng để hiển thị các chữ số hoặc chữ cái đơn giản. Giao tiếp này cho phép người không rành về kỹ thuật cũng có thể sử dụng hệ thống thông qua việc đọc các thông tin hiển thị trên led. Ví dụ nhiều led 7 đoạn có thể được dùng để hiển thị số điện thoại tại các buồng gọi điện thoại công cộng, hoặc các giá trị giây tại các trục đèn giao thông.

Cấu tạo của led 7 đoạn gồm 8 led đơn được nối chung cực anode (dạng led anode chung) hoặc nối chung cực cathode (dạng led cathode chung). Các đoạn led đơn này được đặt tên là a, b, c, d, e, f, g và dấu chấm dp. Để hiển thị một giá trị lên led 7 đoạn, cần cấp điện áp lên chân chung và các tín hiệu điều khiển đoạn tương tự như điều khiển các led đơn.

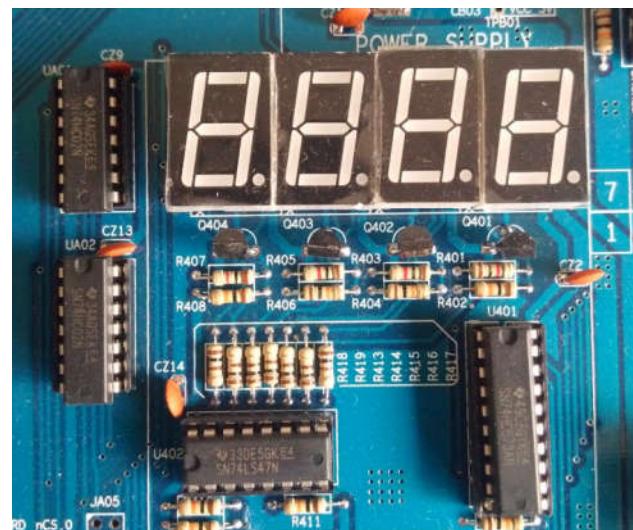
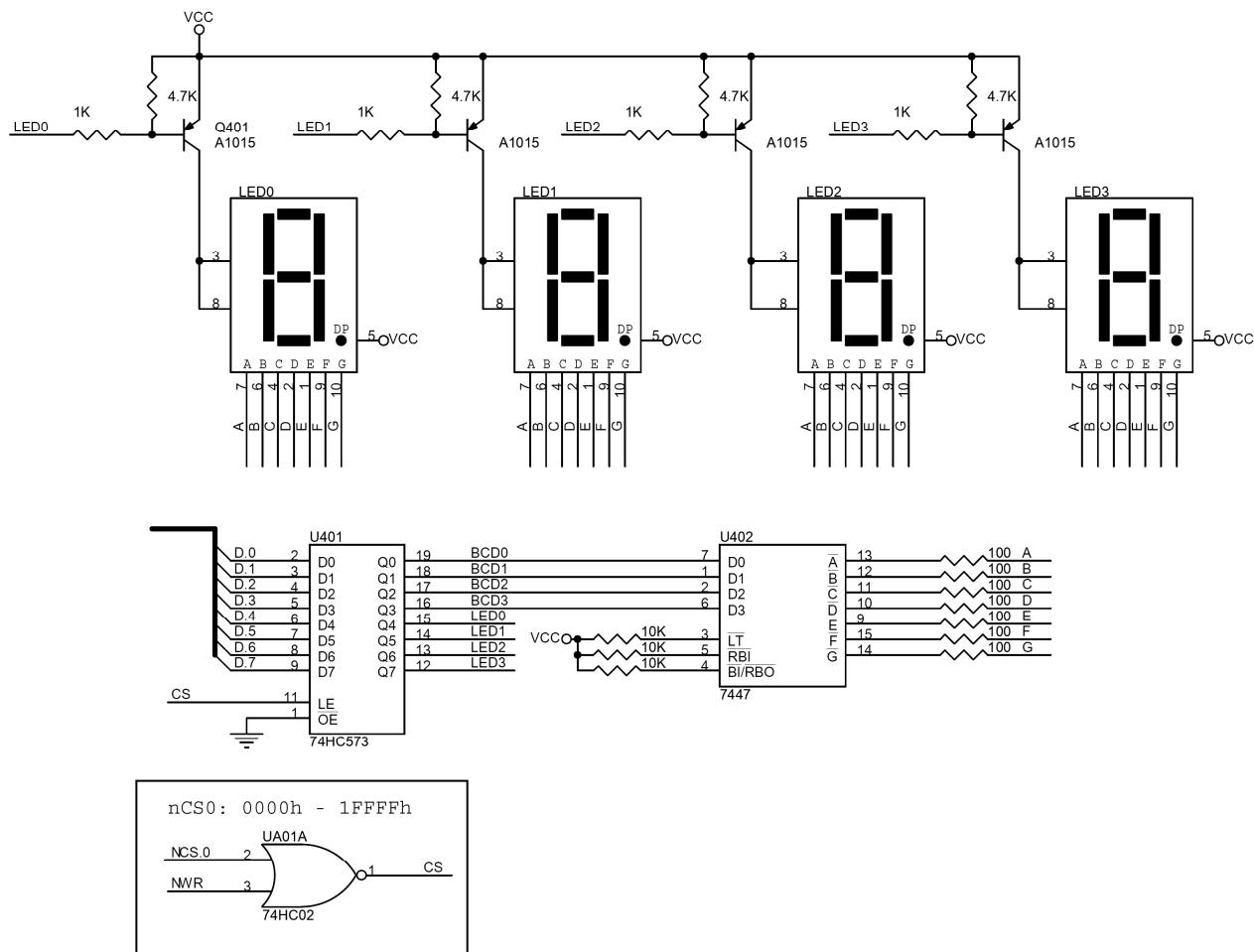


Hình 29 Led 7 đoạn dạng anode chung

Để hiển thị một giá trị số lên led 7 đoạn, người lập trình cần xuất các giá trị để điều khiển các led a, b, ..., g và dp. Tuy nhiên, dữ liệu trong các hệ thống vi xử lý thường tồn tại dưới dạng nhị phân, dạng này không thể trực tiếp hiển thị lên led 7 đoạn. Do đó cần phải thực hiện chuyển đổi từ mã biểu diễn nhị phân sang mã biểu diễn lên led 7 đoạn. Việc chuyển đổi này có thể thực hiện bằng phần cứng với vi mạch chuyển mã hoặc dùng phần mềm (phương pháp tra bảng, look-up table).

HƯỚNG DẪN THÍ NGHIỆM VI XỬ LÝ

4.2 THIẾT KẾ PHẦN CỨNG



Hình 30 Sơ đồ thiết kế khối led 7 đoạn

HƯỚNG DẪN THÍ NGHIỆM VI XỬ LÝ

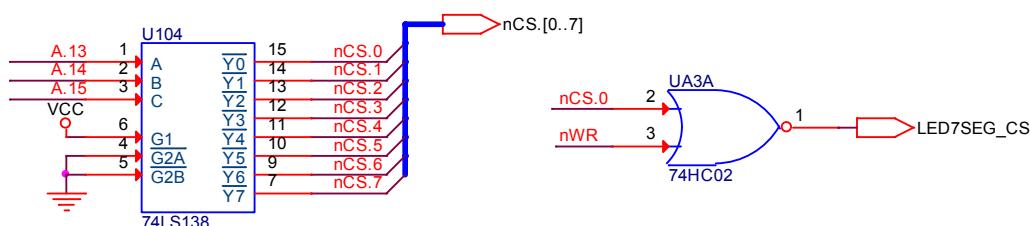
Khối led 7 đoạn gồm có 4 led anode chung dùng vi mạch 74x47 để chuyển từ mã BCD (một dạng mã nhị phân trong hệ thống vi xử lý) sang mã 7 đoạn (mã cho phép hiển thị lên led 7 đoạn). **Khối led 7 đoạn được thiết kế để hoạt động với cơ chế 3 bus.** Trong cơ chế này, Port0 và Port2 được dùng để làm bus dữ liệu và bus địa chỉ. Hai tín hiệu đọc ghi của bus điều khiển nằm trên Port3. Mỗi ngoại vi hoặc bộ nhớ trong chế độ 3 bus sẽ được gán địa chỉ thông qua mạch giải mã địa chỉ. Dữ liệu cần được ghi ra một vi mạch chốt 74x573 vì bus dữ liệu thông thường ở trạng thái Hi-Z.

Trong sơ đồ thiết kế khối led 7 đoạn ở trên, 4 bit thấp của bus dữ liệu sẽ được dùng để chứa mã BCD của số cần hiển thị, 4 bit cao chứa tín hiệu điều khiển khóa của mỗi led. Giá trị BCD sẽ đi qua vi mạch chuyển mã 74x47, ngõ ra của vi mạch này là các tín hiệu lái các đoạn A đến G của led 7 đoạn dạng anode chung. Các tín hiệu dữ liệu và điều khiển được lấy từ ngõ ra của vi mạch chốt ‘573. Đó là do bus dữ liệu của MCU có tính Hi-Z, nên khi hết chu kỳ truy xuất, các đường dữ liệu sẽ mất hết giá trị. Vì mạch chốt ‘573 giúp đảm bảo dữ liệu vẫn tiếp tục tồn tại sau khi MCU không còn truy xuất đến khối led (MCU phải liên tục hiển thị từng led sau một khoảng thời gian nhất định, thường là vài ms). Tín hiệu cho phép chốt ‘573 tích cực mức cao và được tạo ra bằng cách NOR tín hiệu giải mã địa chỉ nCS0 từ 74x138 với tín hiệu cho phép ghi nWR. Địa chỉ của ‘573 này là 0000h.

Cũng theo sơ đồ trên, vì tín hiệu đoạn A đến F của các led 7 đoạn được nối chung nên tất cả các led 7 đoạn đều nhận được cùng một dữ liệu khi MCU truy xuất đến. Tuy nhiên chỉ có duy nhất 1 led 7 đoạn được phép hiển thị bằng cách mở khóa BJT cho led đó. Tín hiệu mức 0 tại cực B của BJT A1015 sẽ làm cho BJT dẫn bão hòa và cực C của BJT sẽ có giá trị gần 5V (gần bằng điện thế tại cực E) cung cấp nguồn 5V cho led 7 đoạn.

4.3 GIAO TIẾP KHỐI LED 7 ĐOẠN

Nhu đã phân tích ở trên, khối Led 7 đoạn được thiết kế như là 1 ngoại vi dạng ánh xạ bộ nhớ, với tín hiệu chọn địa chỉ là CS0 như Hình 31.



Hình 31 Tín hiệu chọn chip cho khối LED 7 đoạn

HƯỚNG DẪN THÍ NGHIỆM VI XỬ LÝ

Một ngoại vi hoặc bộ nhớ dữ liệu ngoài (off-chip) được thiết kế để hoạt động với cơ chế 3 bus có thể được truy xuất bằng câu lệnh *MOVX*. Câu lệnh này chứa các thông tin gồm địa chỉ của ngoại vi hoặc ô nhớ cần truy xuất, dạng lệnh là đọc hoặc ghi.

MOVX A, @DPTR ; đọc ngoại vi tại địa chỉ trong DPTR vào A

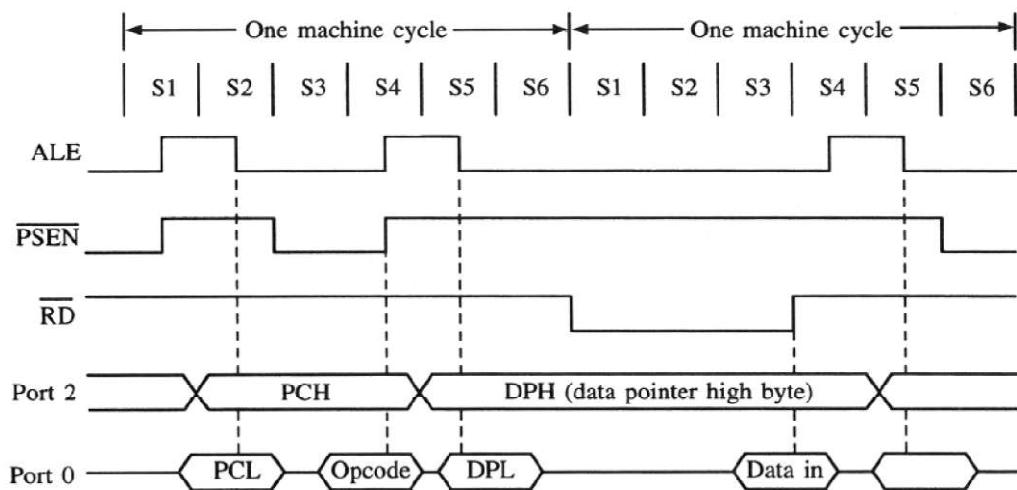
Khi câu lệnh đọc được thực thi, 8051 sẽ thực hiện các bước sau:

- Đặt địa chỉ cần đọc lên bus địa chỉ A0-A15 (tức là Port0 và Port2), giá trị này sẽ làm cho tín hiệu giải mã địa chỉ tương ứng được phép tích cực.
- Ra lệnh đọc bằng tín hiệu điều khiển nRD (bit P3.7).
- Đưa dữ liệu đọc được từ bên ngoài vào thanh ghi A thông qua bus dữ liệu (Port0).

MOVX @DPTR, A ; ghi A ra ngoại vi tại địa chỉ trong DPTR

Khi câu lệnh ghi được thực thi, 8051 sẽ thực hiện các bước sau:

- Đặt địa chỉ cần ghi lên bus địa chỉ A0-A15, giá trị này sẽ làm cho tín hiệu giải mã địa chỉ tương ứng được phép tích cực.
- Đặt dữ liệu trong thanh ghi A lên bus dữ liệu.
- Ra lệnh ghi bằng tín hiệu nWR (bit P3.6).



Hình 32 Giản đồ định thời của lệnh đọc/ghi ngoại vi trong chế độ 3 bus

Điều quan trọng trong việc hiển thị một giá trị lên led 7 đoạn là cần phải chuyển biểu diễn của giá trị tính toán trong MCU (thường là dạng nhị phân) sang dạng hiển thị được lên led 7 đoạn (thường gọi là mã 7 đoạn).

Với phương pháp giải mã phần cứng, giá trị cần ghi ra vi mạch chuyển mã 74x47 phải là mã BCD. Như vậy trong chương trình, người lập trình cần chuyển từ giá trị nhị phân cần hiển thị sang dạng mã BCD. Ví dụ, để hiển thị giá trị 25h (tức là giá trị 37 thập phân), người lập trình cần chuyển giá trị 25h này thành hai số BCD là 3 và 7 (thường được biểu diễn trong hệ thống là 37h). Sau đó lần lượt ghi ra led 7 đoạn tương ứng. Hàm thực hiện việc chuyển đổi này thường được gọi là BIN2BCD (Binary to BCD).

Để hiển thị được nhiều led 7 đoạn, người thiết kế có thể dùng một trong hai phương pháp chủ yếu. Thứ nhất là phương pháp quét led (4 led 7 đoạn của EME-MC8 được hiển thị bằng phương pháp quét led). Trong phương pháp này một led 7 đoạn sẽ được hiển thị trong một khoảng thời gian nhất định sau đó sẽ chuyển sang hiển thị led 7 đoạn khác. Chu trình này được lặp liên tục với tốc độ hiển thị tổng cộng của các led 7 đoạn phải đảm bảo lớn hơn 24 hình/s (tức là tối đa là 40ms cho một lần hiển thị tất cả các led). Để có thể dùng phương pháp quét led, tín hiệu điều khiển các đoạn A đến G của các led được nối chung. Tuy nhiên các led sẽ không hiển thị dữ liệu giống nhau vì tín hiệu cấp nguồn cho led sẽ được điều khiển thông qua một khóa dùng BJT như sơ đồ mạch.

Phương pháp hiển thị thứ hai là phương pháp chốt led. Lúc này nguồn cấp cho mỗi led luôn được mở, tuy nhiên mỗi led sẽ có các tín hiệu điều khiển đoạn A đến G riêng biệt và thường là ngõ ra của một vi mạch chốt ví dụ như 74x573. Phương pháp này đòi hỏi phải có một số lượng vi mạch chốt bằng với số lượng led 7 đoạn. Muốn hiển thị lên led, người lập trình chỉ cần ghi mã hiển thị của led ra vi mạch chốt tương ứng.

4.4 HIỂN THỊ MỘT SỐ TỪ 0-9 LÊN 1 LED 7 ĐOẠN

4.4.1 Sử dụng hợp ngữ

Như đã thấy ở thiết kế phần cứng, để có thể hiển thị một số lên led 7 đoạn, người lập trình cần xuất giá trị BCD của số cần hiển thị ra 4 bit thấp của vi mạch ‘573, 4 bit cao sẽ được dùng để chọn led nào được phép sáng. Giả sử ta cần hiển thị lên led 7 đoạn ngoài cùng bên phải (LED 0) , như vậy, giá trị của 4 bit cao sẽ là 1110 để làm tín hiệu LED0 bằng 0, kích dẫn BJT Q401. Nếu ta muốn hiển thị số 5, 4 bit thấp sẽ có giá trị là 0101. Giá trị này sẽ được IC giải mã chuyển từ BCD sang mã 7 đoạn để hiển thị LED. Vậy, giá trị cần xuất ra là 11100101B hay là E0H.

Vì ‘573 được thiết kế để giao tiếp với MCU thông qua cơ chế 3 bus nên người lập trình cần dùng lệnh *MOVX* dạng ghi, trong đó *DPTR* trả đến địa chỉ 0000h.

HƯỚNG DẪN THÍ NGHIỆM VI XỬ LÝ

Đoạn chương trình sau sẽ hiển thị số 5 lên Led 7 đoạn số 0.

```
MOV      A, #0E0h      ; hiển thị 5 lên led 7 đoạn SỐ 0  
MOV      DPTR, #0000h ; trỏ đến '573 của khối led 7 đoạn  
MOVX    @DPTR, A      ; ghi giá trị ra '573
```

Để dễ hơn trong việc ghi 10 giá trị từ 0 đến 9 ra led 7 đoạn, người lập trình có thể sử dụng phương pháp tra bảng để tận dụng khả năng dùng vòng lặp, trong đó nội dung bảng tra chính là 10 giá trị sẽ lần lượt được xuất ra '573

TABLE:

DB 0E0h, 0E1h, 0E2h, ...

Câu lệnh sau cho phép tra lấy nội dung trong bảng

```
MOV      DPTR, #TABLE ; trỏ đến bảng tra  
MOV      A, #0          ; thứ tự của phần tử cần lấy trong bảng  
MOVC    A, @A+DPTR ; A = 0E0h
```

4.4.2 Sử dụng ngôn ngữ C

Như đã nói ở trên, khối LED 7 đoạn được thiết kế như là một ngoại vi ở địa chỉ 0x0000 - 0x1FFF ở vùng nhớ dữ liệu ngoài của 8051. Để truy xuất đến vùng nhớ này ta sử dụng con trỏ như sau:

```
1 #include <AT89X51.H>  
2 void main(void)  
3 {  
4     unsigned char xdata *p;  
5     p=0x000;  
6     *p = 0xE5;  
7     while (1)  
8     {  
9     }  
10  
11 }  
12 }
```

Hình 33 Sử dụng con trỏ để giao tiếp LED 7 đoạn

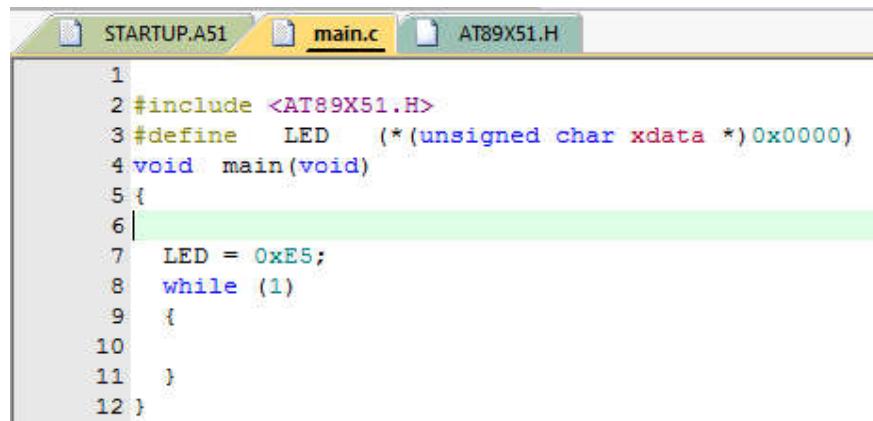
Với cách viết này, con trỏ p là một biến con trỏ chỉ đến vùng nhớ dữ liệu ngoài vì nó được khai báo với từ khóa xdata .

HƯỚNG DẪN THÍ NGHIỆM VI XỬ LÝ

Ta gán giá trị 0x0000 vào p để làm nó trỏ đến địa chỉ 0x0000 ở vùng nhớ ngoài.

Câu lệnh `*p = 0xE5` sẽ xuất giá trị 0xE5 ra địa chỉ 0x0000 ở vùng nhớ ngoài, tương tự như các câu lệnh hợp ngữ ở phần trên.

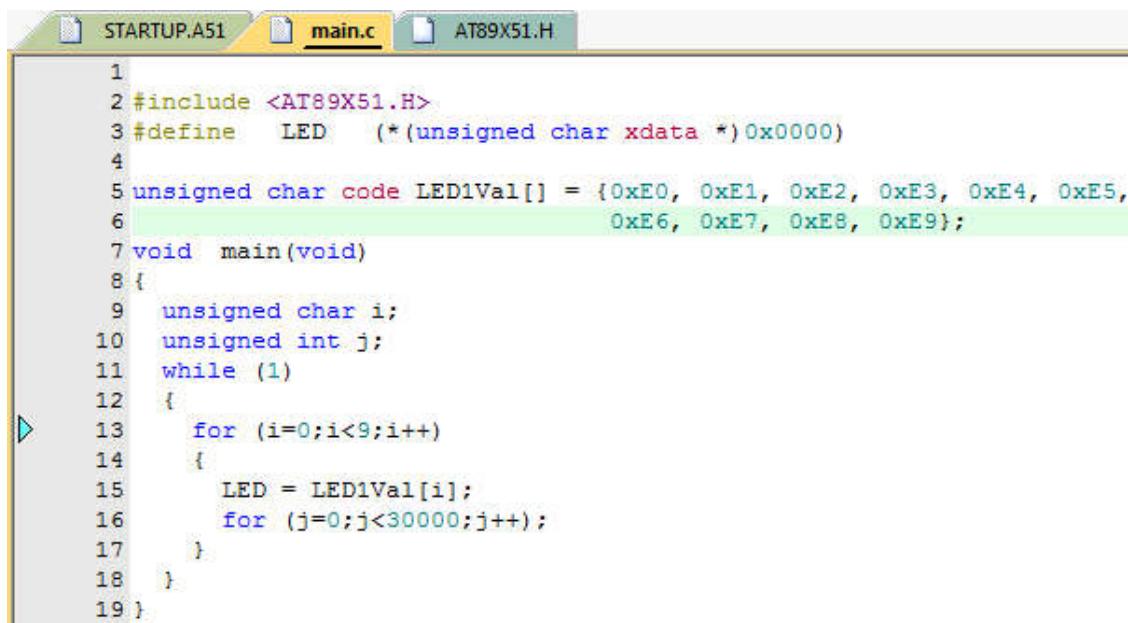
Ngoài cách truy cập thông qua con trỏ, ta có thể định nghĩa một macro chỉ đến ô nhớ 0x0000 và dùng nó để xuất giá trị ra LED như sau



```
1
2 #include <AT89X51.H>
3 #define LED (*(unsigned char xdata *)0x0000)
4 void main(void)
5 {
6
7 LED = 0xE5;
8 while (1)
9 {
10
11 }
12 }
```

Hình 34 Định nghĩa một ngoại vi ánh xạ bộ nhớ

Để khai báo bảng hằng số trong bộ nhớ chương trình và tra bảng này như ở phần trên, ta khai báo 1 mảng với từ khóa "code" và truy cập thông qua chỉ số như ví dụ sau:



```
1
2 #include <AT89X51.H>
3 #define LED (*(unsigned char xdata *)0x0000)
4
5 unsigned char code LED1Val[] = {0xE0, 0xE1, 0xE2, 0xE3, 0xE4, 0xE5,
6                                 0xE6, 0xE7, 0xE8, 0xE9};
7 void main(void)
8 {
9     unsigned char i;
10    unsigned int j;
11    while (1)
12    {
13        for (i=0;i<9;i++)
14        {
15            LED = LED1Val[i];
16            for (j=0;j<30000;j++);
17        }
18    }
19 }
```

Hình 35 Thực hiện bảng tra dùng ngôn ngữ C

4.5 HIỂN THỊ ĐỒNG THỜI GIÁ TRỊ SỐ LÊN CẢ 4 LED 7 ĐOẠN

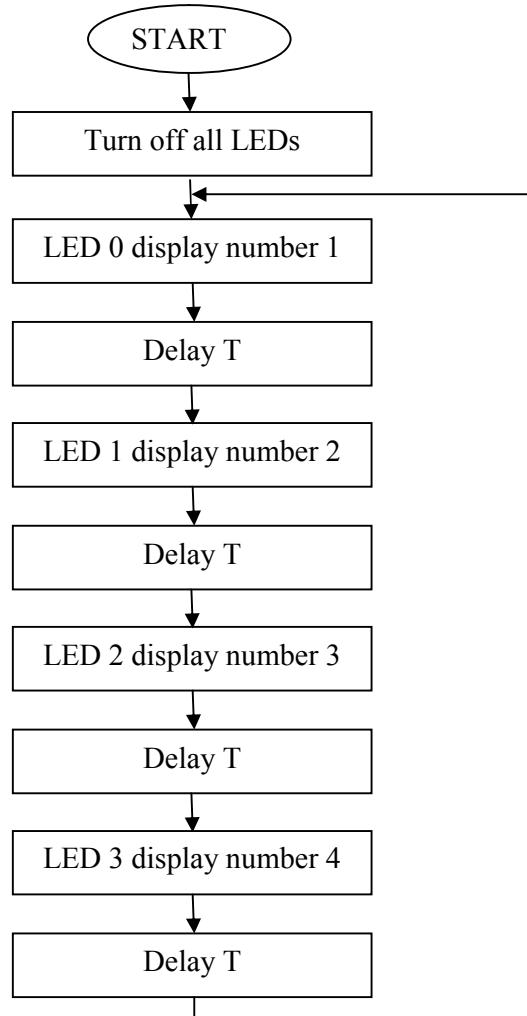
Thiết kế của kit thí nghiệm không cho phép hiển thị 4 số khác nhau lên cả 4 led một cách đồng thời. Để có thể nhìn thấy 4 giá trị cùng 1 lúc ta dùng phương pháp quét LED.

HƯỚNG DẪN THÍ NGHIỆM VI XỬ LÝ

Ví dụ như ta cần hiển thị 4 số 1234 ra 4 LED. Để làm được điều này, ta sẽ cho LED0 hiển thị số 4 trong một thời gian T, sau đó tắt LED0 và cho LED1 hiển thị số 3 cũng trong thời gian T. Tương tự cho LED2 và LED3.

Như vậy, các LED sẽ lần lượt sáng/tắt. Trong 1 giây số lần sáng của mỗi LED sẽ là $1s/4T$.

Như vậy, quá trình hiển thị 4 số 1234 lên LED 7 đoạn như sau:



Hình 36 Lưu đồ chương trình quét LED

Nhờ vào hiệu ứng lưu ảnh của mắt người, nếu LED sáng tắt đủ nhanh thì mắt người sẽ không nhận ra hiện tượng chớp nháy. Theo lý thuyết, nếu số lần LED sáng tắt lớn hơn 24 lần trong 1 giây thì mắt người coi như LED sáng liên tục. Như vậy, thời gian T càng nhỏ thì mắt người càng thấy hình ảnh ổn định.

Cách thức để ghi giá trị ra 1 led tương tự như trong thí nghiệm 1. Tuy nhiên, tín hiệu cho phép led sẽ lần lượt được tích cực để hiển thị 4 giá trị lên 4 led 7 đoạn khác nhau. Như vậy, 4

HƯỚNG DẪN THÍ NGHIỆM VI XỬ LÝ

bit cao sẽ lần lượt có các giá trị là 0111, 1011, 1101, 1110; và 4 bit thấp sẽ chứa mã BCD của số cần hiển thị ra led (tức là 0001, 0010, 0011, 0100). Người lập trình cũng có thể sử dụng phương pháp tra bảng để lấy giá trị xuất ra LED.

TABLE:

DB 071h, 0B2h, 0D3h, 0E4h

CHƯƠNG 5 THÍ NGHIỆM HIỂN THỊ DÙNG LCD

5.1 LÝ THUYẾT CƠ BẢN

Để có thể hiển thị thông tin linh hoạt và tiết kiệm năng lượng, hệ thống có thể sử dụng module LCD. Có nhiều loại module LCD, trong đó thông dụng là loại hiển thị 2 hàng 16 ký tự. Module LCD có thể được dùng để hiển thị các thông tin dạng ký tự. Vì được tích hợp sẵn bộ lái LCD nên việc điều khiển module LCD tương đối đơn giản. Các tín hiệu điều khiển module LCD gần giống với các tín hiệu của một MCU hoạt động theo cơ chế 3 bus.

Module LCD đã được thiết kế chuẩn để cho phép ta có thể giao tiếp với LCD do một hãng bất kỳ sản xuất với điều kiện là các LCD có sử dụng cùng IC điều khiển HD44780. Phần lớn các module LCD sử dụng giao tiếp 14 chân trong đó có 8 đường dữ liệu, 3 đường điều khiển và 3 đường cấp nguồn. Kết nối được bố trí dưới dạng 1 hàng 14 chân hoặc 2 hàng 7 chân.

Các chân 1 và 2 là các chân cấp nguồn Vss, Vdd. Chân 3 Vee là chân điều khiển độ tương phản của màn hình. Chân 4 là đường RS, đây là chân điều khiển lệnh. Khi RS = 0 thì dữ liệu ghi vào LCD được hiểu là các lệnh, dữ liệu đọc từ LCD được hiểu là trạng thái của nó. Chân 5 là đường điều khiển đọc ghi R/nW, mức thấp sẽ cho phép ghi vào LCD, mức cao cho phép đọc ra từ LCD. Chân 6 là đường điều khiển cho phép E. Các chân còn lại chứa dữ liệu 8-bit vào hoặc ra LCD.

Chân số	Tên	Chức năng
1	V _{SS}	Đất
2	V _{DD}	Cực + của nguồn điện
3	V _{EE}	Tương phản (contrast)
4	RS	Register Select (Chọn thanh ghi)
5	R/W	Read/Write
6	E	Cho phép (Enable)
7	D0	Bit 0 của dữ liệu
8	D1	Bit 1 của dữ liệu

HƯỚNG DẪN THÍ NGHIỆM VI XỬ LÝ

9	D2	Bit 2 của dữ liệu
10	D3	Bit 3 của dữ liệu
11	D4	Bit 4 của dữ liệu
12	D5	Bit 5 của dữ liệu
13	D6	Bit 6 của dữ liệu
14	D7	Bit 7 của dữ liệu

5.1.1 Chức năng các chân của LCD

Module LCD được điều khiển thông qua một tập lệnh chuẩn. Bảng sau tóm tắt các lệnh điều khiển LCD.

Command (Lệnh)	RS	RW	Nhị phân								Hex
			D7	D6	D5	D4	D3	D2	D1	D0	
NOP (No operation = không làm gì cả)	0	0	0	0	0	0	0	0	0	0	00
Clear display (xóa hiển thị)	0	0	0	0	0	0	0	0	0	1	01
Display & Cursor home (hiển thị và đặt cursor ở góc trái phía trên)	0	0	0	0	0	0	0	0	1	x	02 hoặc 03
Character Entry mode (Chế độ nhập ký tự)	0	0	0	0	0	0	0	1	I/D	S	04 đến 07
Display On/Off & Cursor	0	0	0	0	0	0	1	D	U	B	08 đến 0F

HƯỚNG DẪN THÍ NGHIỆM VI XỬ LÝ

(Tắt mở hiển thị và cursor)																			
Display/Cursor Shift (Dịch curson/hiển thị)	0	0	0	0	0	1	D/C	R/L	x	x	10 đến 1F								
Function set (Đặt chức năng)	0	0	0	0	1	8/4	2/1	10/7	x	x	20 đến 3F								
Set CGRAM address (Đặt địa chỉ CGRAM)	0	0	0	1	A	A	A	A	A	A	40 đến 7F								
Set DDRAM address (Đặt địa chỉ DDRAM)	0	0	1	A	A	A	A	A	A	A	80 đến FF								
Busy Flag & Addr (Cờ bộ đếm và bộ đếm địa chỉ)	0	1	BF	Bộ đếm địa chỉ															
Read Data (Đọc dữ liệu từ CGRAM hoặc DDRAM)	1	0	Dữ liệu đọc																
Write Data (Ghi dữ liệu vào CGRAM hoặc DDRAM)	1	1	Dữ liệu ghi																

Chú thích:

I/D: 1 = Increment *, 0=Decrement

R/L: 1 = Right shift, 0 = Left shift

HƯỚNG DẪN THÍ NGHIỆM VI XỬ LÝ

S: 1 = Display shift on, 0 = Display shift off * 8/4: 1 = 8 bit interface *, 0 = 4 bit interface
 D: 1 = Display on, 0 = Diaplay off * 2/1: 1 = ché độ 2 hàng, 0 = ché độ 1 hàng *
 U: 1 = Cursor underline on, 0=Underline off * 10/7: 1=5x10 dot format, 0=5x7 dot format *
 B: 1 = Cursor blink on, 0 = cursor blink off *
 D/C: 1 = Display shift, 0 = cursor move x = don't care * = đặt ban đầu

5.1.2 Tóm tắt tập lệnh điều khiển LCD

Các dữ liệu được hiển thị lên LCD dựa vào một tập ký tự chuẩn. Các dữ liệu 8-bit sẽ được chuyển thành ký tự hiển thị dựa trên bảng ký tự chuẩn này.

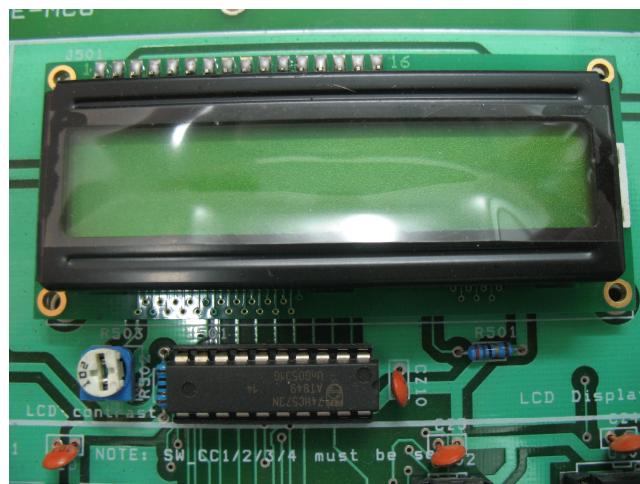
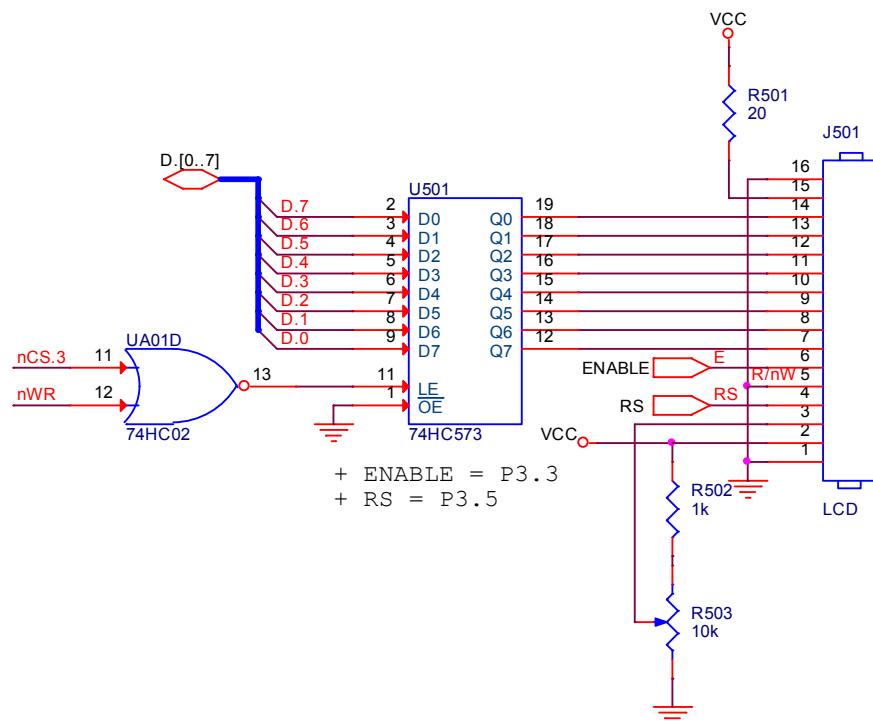
xxxx	Lower 4 Bits	Upper 4 Bits	0000	0001	0010	0011	0100	0101	0110	0111	1000	1001	1010	1011	1100	1101	1110	1111
xxxx0000		CG RAM (1)	Ø	ø	P	’P					-	‐	Ξ	Ξ	ø	p		
xxxx0001	(2)		!	!1	A	Q	a	q			■	■	?	?	4	ä	q	
xxxx0010	(3)		”	”2	B	R	b	r			‘	‘	イ	イ	ツ	×	β	θ
xxxx0011	(4)		#	#3	C	S	c	s			„	„	ウ	ウ	テ	€	ω	
xxxx0100	(5)		\$	\$4	D	T	d	t			、	、	エ	エ	ト	ト	μ	¤
xxxx0101	(6)		%	%5	E	U	e	u			・	・	オ	オ	ナ	।	Ը	Ը
xxxx0110	(7)		&	&6	F	U	f	v			ヲ	ヲ	カ	カ	ニ	Յ	ρ	Σ
xxxx0111	(8)		’	’7	G	W	g	w			ヰ	ヰ	ヰ	ヰ	ヰ	ヰ	ヰ	ヰ
xxxx1000	(1)		((8	H	X	h	x			イ	イ	ク	ク	ネ	リ	յ	չ
xxxx1001	(2)))9	I	Y	i	y			Ղ	Ղ	Ղ	Ղ	Ղ	Ղ	Ղ	Ղ
xxxx1010	(3)		*	*:	J	Z	j	z			エ	エ	コ	コ	ハ	レ	յ	չ
xxxx1011	(4)		+	+:	K	L	k	l			オ	オ	サ	サ	ヒ	□	¤	¤
xxxx1100	(5)		,	,<	L	¥	1	l			ヰ	ヰ	シ	シ	フ	ワ	¤	¤
xxxx1101	(6)		-	=M	M	J	m)			ュ	ュ	ス	ス	ヘ	ն	մ	մ
xxxx1110	(7)		.	.>	N	^	n	+			զ	զ	է	է	տ	՞	՞	՞
xxxx1111	(8)		/	/?	O	_	o	+			ս	ս	Ս	Ս	Ր	Պ	օ	՞

Bảng 4 Bảng ký tự chuẩn cho LCD

HƯỚNG DẪN THÍ NGHIỆM VI XỬ LÝ

Định thì giao tiếp các tín hiệu điều khiển của LCD có thể được xem thêm trong mô tả kỹ thuật của vi mạch HD44780.

5.2 THIẾT KẾ PHẦN CỨNG

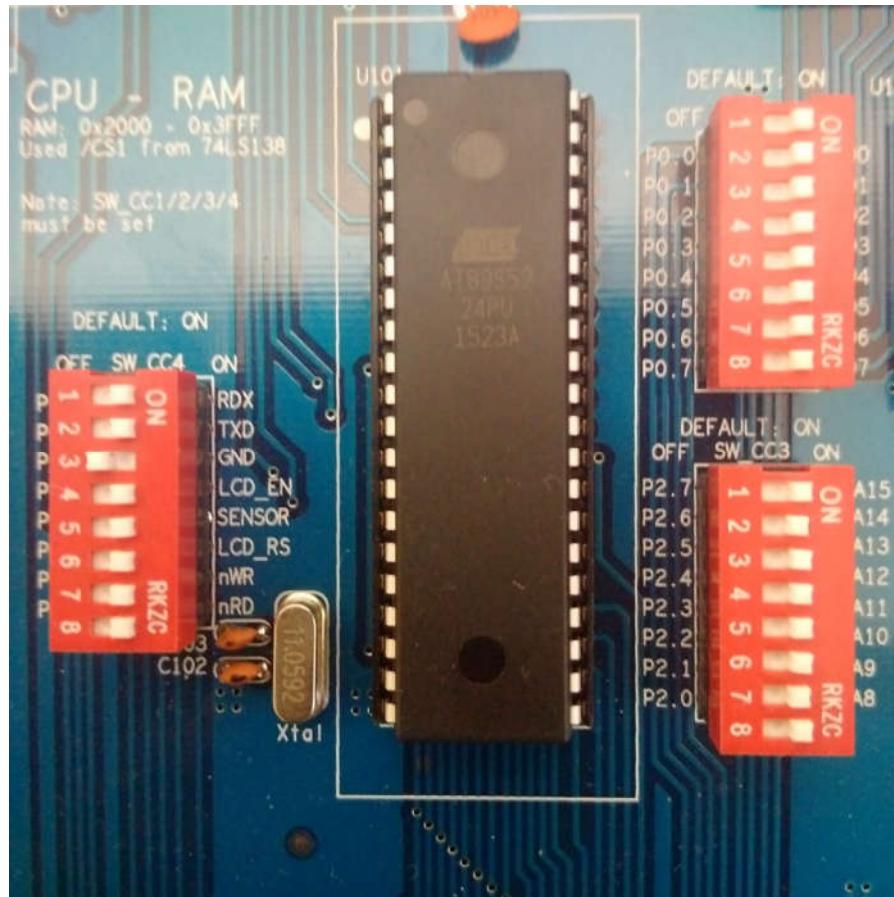


Hình 37 Sơ đồ thiết kế giao tiếp LCD

LCD được thiết kế để hoạt động với cơ chế 3 bus, tức là phải bật ON **SW_CC1** và **SW_CC3**. 8 đường dữ liệu D0-D7 chứa dữ liệu hiển thị hoặc lệnh điều khiển LCD được ghi vào LCD thông qua vi mạch chốt 74x573, trong đó tín hiệu cho phép chốt CS được tạo ra từ mạch giải mã địa chỉ.

HƯỚNG DẪN THÍ NGHIỆM VI XỬ LÝ

Hai tín hiệu điều khiển E và RS của LCD được kết nối đến P3.3 và P3.5 của 8051. Người sử dụng phải bật ON hai switch này trên **SW_CC4** để có thể điều khiển LCD. Như vậy, **SW_CC4** phải bật ON tổng cộng 6 switch tương ứng với các bit 0, 1, 4, 5, 6, 7 của Port 3, tức là các tín hiệu RXD, TXD, E, RS, nWR, nRD



Hình 38 Cài đặt SW_CC4 của Port 3 để điều khiển LCD

5.3 LẬP TRÌNH GIAO TIẾP LCD

Vì giao tiếp thông qua cơ chế 3 bus nên người lập trình cần dùng câu lệnh *MOVX* phối hợp với việc điều khiển hai tín hiệu E (P3.4) và RS (P3.5) để truy xuất LCD theo đúng định thi ghi. Để điều khiển LCD, trước tiên cần khởi động và đặt cấu hình cho LCD. Việc này được thực hiện bằng cách gửi một số lệnh khởi động đến LCD. Lệnh thứ nhất mà ta gửi báo cho LCD biết ta sẽ giao tiếp với nó thông qua bus dữ liệu 4-bit hay 8-bit. Ta cũng chọn font ký tự là 5x8. Điều này được thực hiện bằng cách gửi lệnh 38h đến LCD. Lưu ý là đường RS phải giữ ở mức thấp để báo cho LCD biết đang nhận lệnh. Tiếp theo ta cần gửi lệnh 0Eh. Lệnh này dùng để bật LCD và tắt con trỏ ký tự. Byte thứ 3 được gửi thêm để cài đặt một số tham số hoạt động của LCD. Ví dụ ta có thể gửi lệnh 06h để ra lệnh cho con trỏ tự động dịch phải mỗi

khi ta gửi một ký tự hiển thị cho nó. Lệnh *MOVX* có thể được dùng để truy xuất đến module LCD với *DPTR* chứa địa chỉ của thao tác cụ thể.

Để đảm bảo module LCD hoàn tất một thao tác điều khiển, người lập trình có thể dùng một trong hai phương pháp. Cách thứ nhất là sử dụng chương trình con tạo trễ để chờ module hoàn thành lệnh hiện thời. Thời gian chờ cụ thể của module LCD cần phải xem cụ thể trong mô tả kỹ thuật của vi mạch HD44780. Cách thứ hai là sau mỗi lần truy xuất đến module LCD, người lập trình cần liên tục đọc trạng thái của module LCD và kiểm tra bit thứ 7 của byte trạng thái nhận được. Nếu bit này là 1 thì LCD vẫn còn đang bận, chương trình cần tiếp tục chờ. Nếu bit này là 0 thì LCD đã hoàn tất thao tác hiện thời và sẵn sàng cho thao tác điều khiển khác.

```
MOVX    A, @DPTR ; đọc trạng thái LCD  
JB      ACC.7, wait ; nếu busy thì nhảy đến wait  
...     ; nếu không thì tiếp tục
```

5.4 VIẾT CHƯƠNG TRÌNH HIỂN THỊ CHUỖI LÊN LCD

Khối LCD được kết nối với 8051 thông qua cơ chế 3 bus, trong đó 8 đường dữ liệu của LCD được nối đến bus dữ liệu của MCU, 3 đường điều khiển được nối đến ngõ ra của khối giải mã địa chỉ. Như vậy lệnh *MOVX* sau có thể được dùng để đọc hoặc ghi LCD, trong đó *DPTR* chứa địa chỉ của thao tác tương ứng

```
MOVX    A, @DPTR ; lệnh đọc LCD  
MOVX    @DPTR, A ; lệnh ghi LCD
```

Giả sử ta cần ghi chuỗi "Hello" lên LCD. Trước tiên cần khởi động LCD theo mô tả ở phần trên. Sau đó, lần lượt ghi mã của các ký tự 'H', 'e', 'l', 'l', 'o' đến LCD theo bảng ký tự của LCD. **Lưu ý** là sau mỗi lần truy xuất module LCD cần chờ cho thao tác hiện thời được hoàn tất trước khi bắt đầu thao tác khác.

Để dễ dàng hơn trong việc lấy các ký tự và xuất ra LCD, người lập trình có thể dùng phương pháp tra bảng như sau

Message: *DB* *'Hello'*

Nên thiết kế chương trình bằng cách sử dụng các chương trình con để có thể sử dụng lại. Ví dụ, một chương trình con có tên *lcd_init* có nhiệm vụ khởi động module LCD, chương trình

HƯỚNG DẪN THÍ NGHIỆM VI XỬ LÝ

con *write_char* có nhiệm vụ ghi 1 byte dữ liệu hiển thị đến LCD, chương trình con *write_command* có nhiệm vụ ghi 1 byte lệnh đến LCD.

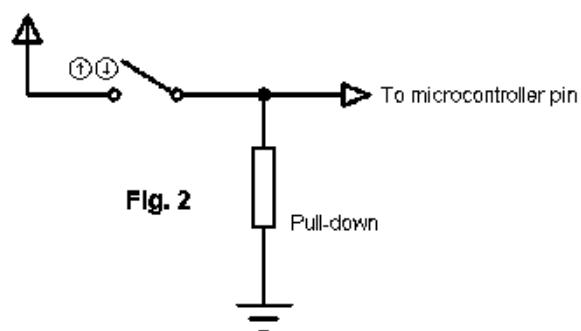
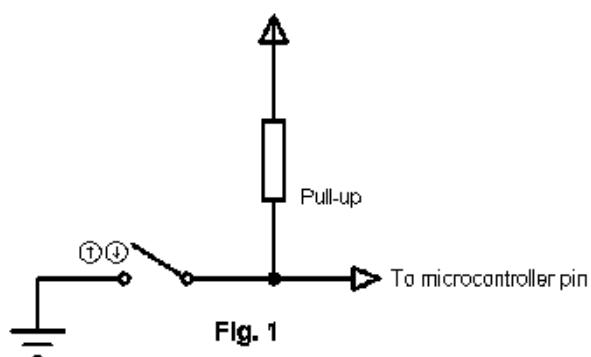
Mã nguồn chương trình mẫu hiển thị một chuỗi lên LCD có thể download trên trang web của bộ môn điện tử.

CHƯƠNG 6 THÍ NGHIỆM VỚI CÔNG TẮC ĐƠN VÀ BÀN PHÍM MA TRẬN

6.1 LÝ THUYẾT CƠ BẢN

Công tắc (Switch) là linh kiện cơ khí được dùng để nối 2 hoặc nhiều tiếp điểm lại với nhau khi có tác động. Thông thường Switch có 2 tiếp điểm (Switch đơn).

Để giao tiếp switch đơn có hai cách giao tiếp như sau.



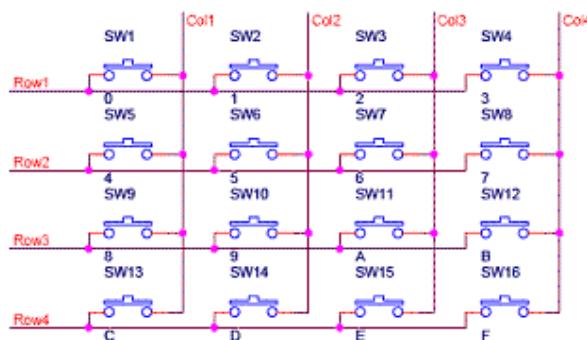
Hình 39 Giao tiếp phím nhấn

Ở cách phía trên, một điện trở kết nối nguồn VCC và 1 đầu của switch, đầu này lấy tín hiệu ra đưa vào chân vi điều khiển, đầu còn lại nối với GND. Khi Switch không đóng, tín hiệu ra sẽ là HIGH (logic 1). Khi Switch đóng, tín hiệu ra sẽ là LOW (logic 0). Cách kết nối này là sử dụng điện trở kéo lên (pull-up resistor)

Cách giao tiếp còn lại thì hoàn toàn ngược lại, dùng điện trở kéo xuống (pull-down resistor).

HƯỚNG DẪN THÍ NGHIỆM VI XỬ LÝ

Trên bàn phím ma trận, các phím nhấn được tổ chức thành dạng ma trận, ở đó một đầu của phím nối vào các hàng, đầu còn lại nối vào các cột như Hình 40.

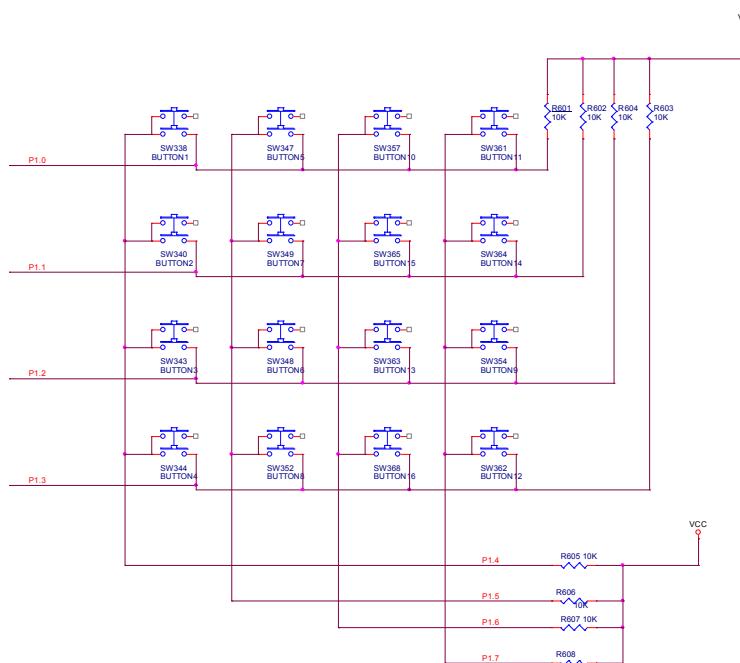


Hình 40 Tô chức bàn phím ma trận

Với cách kết nối trên, khi ta nhấn một phím thì hàng và cột sẽ được kết nối với nhau. Từ số hàng và cột trên ta có thể suy ra vị trí phím được nhấn.

6.2 THIẾT KẾ PHẦN CỨNG

Trên kit thí nghiệm, bàn phím ma trận gồm có 16 phím được tổ chức thành 4 hàng, bốn cột và kết nối tới PORT 1.



Hình 41 Kết nối bàn phím ma trận

HƯỚNG DẪN THÍ NGHIỆM VI XỬ LÝ

Như trên Hình 41, bốn hàng của bàn phím kết nối vào P1.0, P1.1, P1.2 và P1.3. Bốn cột kết nối vào P1.4, P1.5, P1.6 và P1.7.

6.3 PHẦN MỀM GIAO TIẾP

Để giao tiếp bàn phím ma trận ta sử dụng kỹ thuật quét phím. Ta có thể quét phím theo hàng hoặc theo cột. Sau đây trình bày quá trình quét phím theo cột.

Đầu tiên, ta xuất giá trị 1 ra các chân giao tiếp hàng (P1.0 đến P1.3) để có thể sử dụng các chân này làm input.

Cho cột 0 (P1.4) bằng 0 và 3 cột còn lại lên 1. Nếu như có bất kỳ phím nào trên cột 0 được nhấn, hàng tương ứng sẽ được kết nối vào cột 0 và sẽ có giá trị là 0. Như vậy ở lần quét này ta biết được có phím nào trên cột 0 được nhấn.

Tương tự, Cho cột 1 (P1.5) bằng 0 và 3 cột còn lại lên 1. Nếu như có bất kỳ phím nào trên cột 0 được nhấn, hàng tương ứng sẽ được kết nối vào cột 1 và sẽ có giá trị là 0. Như vậy ở lần quét này ta biết được có phím nào trên cột 1 được nhấn.

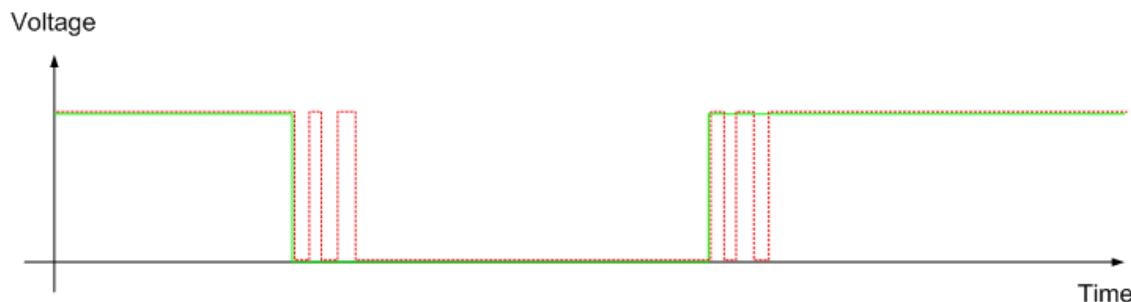
Làm tương tự đối với cột 2 và cột 3.

Như vậy, một chu kỳ quét phím sẽ quét qua lần lượt 4 cột, và phát hiện được có phím nhấn trên toàn bộ các phím trên ma trận phím.

Thời gian quét hết 4 cột là rất nhanh so với tốc độ nhấn và nhả phím của tay người, vì vậy đảm bảo khi bất kỳ phím nào được nhấn, quá trình quét phím sẽ phát hiện được.

6.4 RUNG PHÍM VÀ CHỐNG RUNG

Với phím nhấn cơ khí, tiếp điểm cơ khí sau khi được nhấn hay nhả sẽ bị rung động. Do đó, khi nhấn hay nhả phím sẽ tạo ra một chuỗi các xung thay vì một xung đơn như ở Hình 42.

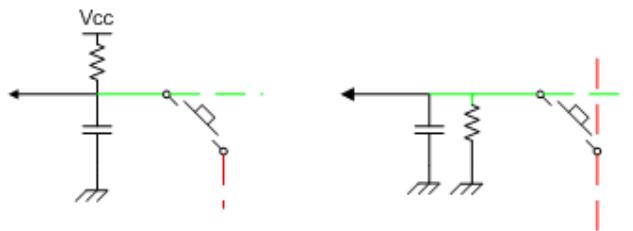


Hình 42 Phím đơn bị rung khi nhấn hay nhả

Khi đó, nếu ta dùng tín hiệu này để xác định số lần phím nhấn thì kết quả sẽ bị sai lệch. Để có kết quả đúng ta cần chống rung (debounce).

HƯỚNG DẪN THÍ NGHIỆM VI XỬ LÝ

Có hai cách để chống rung là chống rung bằng phần cứng và chống rung bằng phần mềm.



Schematic 1

Schematic 2

Hình 43 Chống rung bằng phần cứng

Hình 43 trình bày cách chống rung bằng phần cứng, sử dụng một tụ điện khoảng 0.1 uF mắc như trong hình. Tụ này phối hợp với điện trở kéo lên hoặc kéo xuống sẽ lọc đi các xung hẹp sinh ra do quá trình rung phím.

Cách thứ hai là sử dụng phần mềm để chống rung.

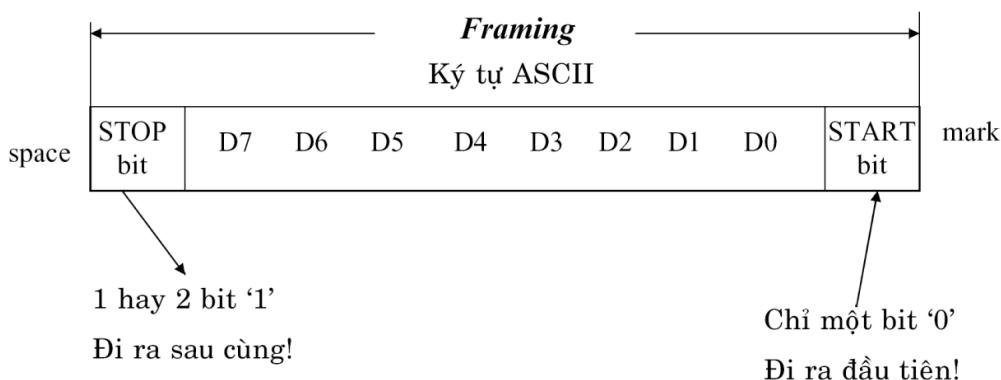
Cách chống rung đơn giản nhất là đọc tín hiệu ra hai lần liên tiếp sau một khoảng thời gian trễ thích hợp, nếu 2 lần này mà kết quả đọc được giống nhau thì ta coi là kết quả đúng, nếu kết quả là sai thì ta lại đọc lần thứ 3 cũng sau thời gian T, nếu kết quả giống như lần đọc thứ 2 thì coi như kết quả lần này là đúng, còn nếu không thì lại lặp lại quá trình. Thời gian T thường được chọn là 50ms.

Ta có thể tăng độ chính xác bằng cách giảm thời gian T và tăng số lần đọc lên. Ví dụ phím được coi là trả về giá trị 1 nếu như ta đọc được liên tiếp 20 lần cùng giá trị 1 với mỗi lần đọc cách nhau 1ms.

CHƯƠNG 7 THÍ NGHIỆM GIAO TIẾP QUA CỔNG NỐI TIẾP

7.1 LÝ THUYẾT CƠ BẢN

Truyền thông nối tiếp là một phần rất quan trọng trong các hệ thống xử lý. Đối với 8051, việc truyền nối tiếp được thực hiện thông qua ngoại vi cổng nối tiếp. Ngoại vi này cho phép người lập trình giao tiếp với bên ngoài thông qua giao thức truyền nối tiếp 8 bit dữ liệu, 1 stop bit. Tốc độ truyền có thể lập trình được bằng phần mềm.



7.1.1 Khung truyền nối tiếp bất đồng bộ

Khi không có dữ liệu gửi thì đường tín hiệu duy trì ở trạng thái cao (trạng thái Mark). Bắt đầu của một ký tự dữ liệu được chỉ bởi mức thấp trong thời gian 1 bit. Bit này được gọi là bit bắt đầu (Start bit). Rồi sau đó các bit dữ liệu được gửi ra trên đường tín hiệu lần lượt từng bit một (bắt đầu với LSB). Từ dữ liệu có thể 5, 6, 7, hoặc 8 bit và có thể theo sau là bit kiểm tra chẵn lẻ P (Parity bit). Tiếp theo các bit dữ liệu và P (nếu có sử dụng kiểm tra chẵn lẻ), đường tín hiệu được trả về mức cao trong ít nhất thời gian 1 bit để giúp nhận biết kết thúc ký tự. Bit này còn gọi là bit dừng (Stop bit), một số hệ thống cũ có thể sử dụng 2 bit dừng.

Thuật ngữ tốc độ baud dùng để chỉ tốc độ dữ liệu nối tiếp được truyền. Tốc độ baud được định nghĩa là $1/(thời gian giữa những chuyển tiếp tín hiệu)$. Thí dụ: Nếu tín hiệu thay đổi cứ sau 3.33 ms thì tốc độ baud là $1/3.33ms = 300$ bd (hay baud). Chú ý là tốc độ này tổng quát thì khác với tốc độ định nghĩa theo bps (bits/giây). Các tốc độ baud thông dụng là 300, 600, 1200, 2400, 4800, 9600, và 19200 baud (hiện nay các số này còn cao hơn nữa, thường giới hạn với truyền bất đồng bộ là 100000 baud).

Bộ thí nghiệm EME-MC8 đã sử dụng cổng nối tiếp này để giao tiếp với chương trình HyperTerminal trên máy tính, từ đó cho phép người dùng có thể sử dụng các chức năng của chương trình monitor.

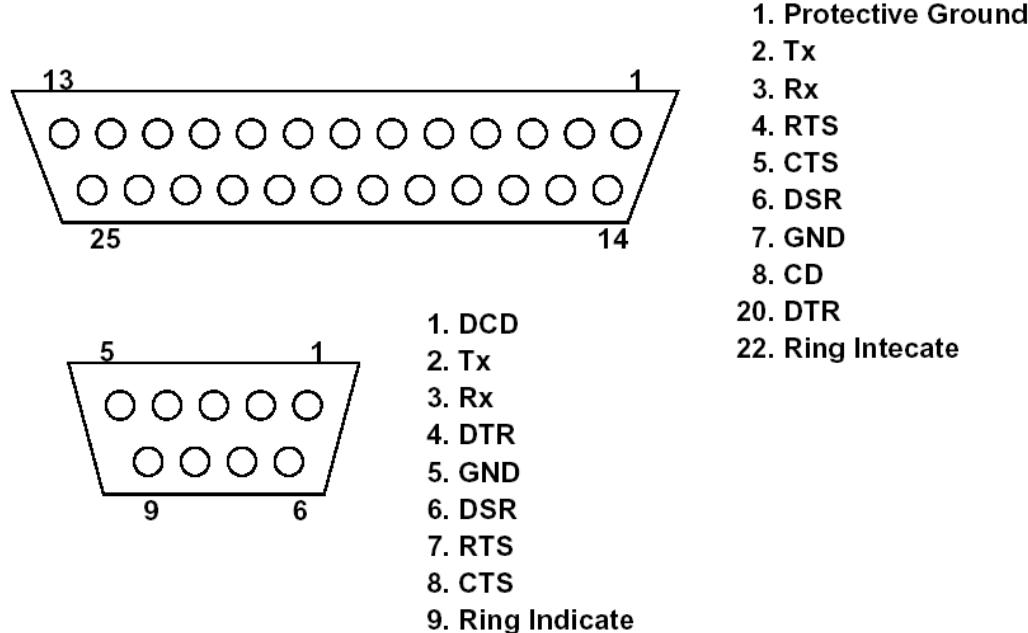
HƯỚNG DẪN THÍ NGHIỆM VI XỬ LÝ

Trong thực tế, cổng nối tiếp của 8051 có thể giao tiếp với cổng RS-232 trên máy tính thông qua một vi mạch chuyển đổi từ TTL sang RS-232 như MAX232. Việc truyền thông tin chỉ cần 3 dây TXD, RXD, và GND nếu không dùng bắt tay bằng phần cứng.

7.2 THIẾT KẾ PHẦN CỨNG

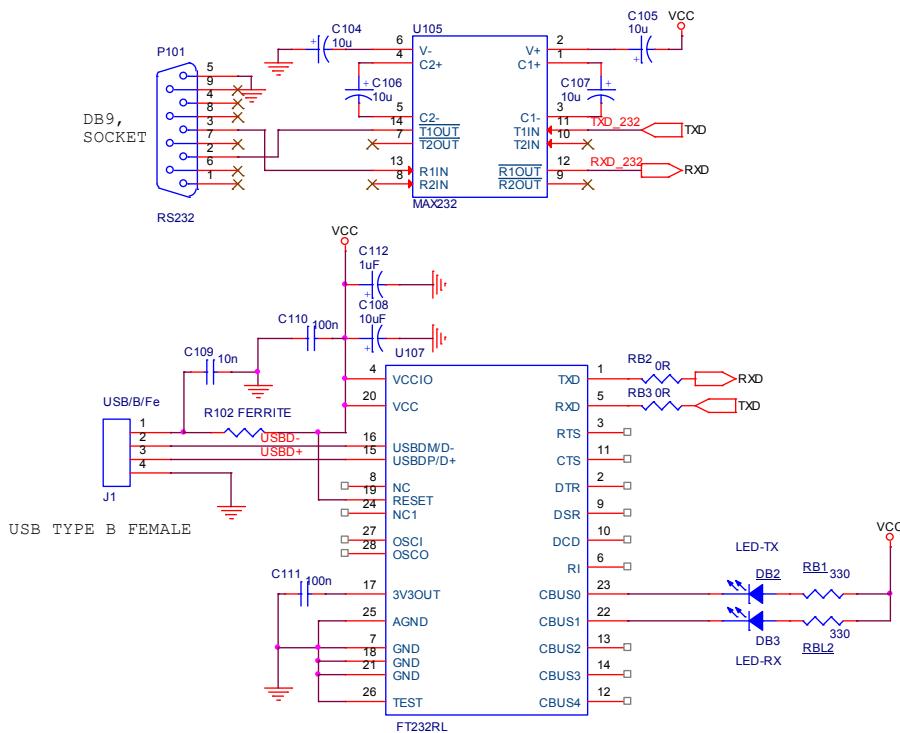
Trên kit thí nghiệm, cổng nối tiếp của 8051 được nối đến cổng DB9 trên mạch thông qua vi mạch chuyển đổi từ TTL sang RS-232 (MAX232) và nối với cổng USB qua IC chuyển đổi FTDI232. Điều này cho phép kết nối DB9 với cổng COM của máy tính hoặc cổng USB của kit với cổng USB của máy tính để giao tiếp máy tính với serial port của 8051.

Chân của DB25 (chân số)	Chân của DB9 (chân số)	Viết tắt	Tên đầy đủ
2	3	TD	Transmit Data
3	2	RD	Receive Data
4	7	RTS	Request To Send
5	8	CTS	Clear To Send
6	6	DSR	Data Set Ready
7	5	SG	Signal Ground
8	1	CD	Carrier Detect
20	4	DTR	Data Terminal Ready
22	9	RI	Ring Indicator



HƯỚNG DẪN THÍ NGHIỆM VI XỬ LÝ

Hai tín hiệu dùng cho giao tiếp là TxD và RxD trên Port 3 của 8051. Như vậy, người lập trình cần bật ON hai switch này (của SW_CC4) để có thể dùng cổng nối tiếp.



Hình 44 Sơ đồ kết nối cổng nối tiếp và cài đặt switch tương ứng

7.3 PHẦN MỀM GIAO TIẾP

Để truyền nhận dữ liệu thông qua cổng nối tiếp, người lập trình chỉ cần khởi động nó bằng các thao tác: truy xuất thanh ghi SMOD để cài đặt chế độ làm việc; truy xuất Timer 1 để điều khiển tốc độ baud (EME-MC8 sử dụng thạch anh 11.0592MHz, cho phép tạo tốc độ baud với sai số là 0%).

MOV SMOD, #52h ; khởi động ở chế độ 2

MOV TH1, #... ; cài đặt tốc độ

Để có thể ghi hoặc đọc dữ liệu từ cổng nối tiếp, người lập trình cần truy xuất thanh ghi SBUF bằng các chương trình con

JNB TI, \$; chờ phát xong byte trước đó

CLR TI ; chuẩn bị phát byte kế

MOV SBUF, A ; phát

HƯỚNG DẪN THÍ NGHIỆM VI XỬ LÝ

Hoặc

JNB RI, \$; chờ nhận xong 1 byte

CLR RI ; nhận byte kế

MOV A, SBUF ; byte kế nằm trong A

CHƯƠNG 8 THÍ NGHIỆM LẬP TRÌNH NGẮT CHO 8051

8.1 LÝ THUYẾT CƠ BẢN

Ngắt là tính năng rất quan trọng của vi xử lý, cho phép chương trình phục vụ một ngoại vi khi cần mà không cần phải mất thời gian hỏi vòng (polling) các ngoại vi.

Với 8051, khi xảy ra sự kiện ngắt, ví dụ như timer tràn, chương trình sẽ quay về một địa chỉ cố định trong bộ nhớ chương trình, địa chỉ này được gọi là vector ngắt của ngoại vi. Từ địa chỉ này, chương trình thực hiện một lệnh nhảy đến địa chỉ của chương trình phục vụ ngắt.

Interrupt	Source	Vector Address
System Reset	RST	0000H
External 0	IE0	0003H
Timer 0	TF0	000BH
External 1	IE1	0013H
Timer 1	TF1	001BH
Serial Port	RI or TI	0023H

Để cho phép ngắt, người lập trình phải cho phép bit tương ứng trong thanh ghi IE

IE : Interrupt Enable Register (Bit Addressable)

If the bit is 0, the corresponding interrupt is disabled. If the bit is 1, the corresponding interrupt is enabled.

EA	-	-	ES	ET1	EX1	ET0	EX0
----	---	---	----	-----	-----	-----	-----

EA	IE.7	Disables all interrupts. If EA = 0, no interrupt will be acknowledged. If EA = 1, interrupt source is individually enable or disabled by setting or clearing its enable bit.
-	IE.6	Not implemented, reserved for future use*.
-	IE.5	Not implemented, reserved for future use*.
ES	IE.4	Enable or disable the Serial port interrupt.
ET1	IE.3	Enable or disable the Timer 1 overflow interrupt.
EX1	IE.2	Enable or disable External interrupt 1.
ET0	IE.1	Enable or disable the Timer 0 overflow interrupt.
EX0	IE.0	Enable or disable External Interrupt 0.

Người lập trình cũng có thể chọn mức ưu tiên cho ngắt thông qua thanh ghi IP

IP : Interrupt Priority Register (Bit Addressable)

If the bit is 0, the corresponding interrupt has a lower priority and if the bit is 1 the corresponding interrupt has a higher priority.

-	-	-	PS	PT1	PX1	PT0	PX0
---	---	---	----	-----	-----	-----	-----

-	IP.7	Not implemented, reserved for future use*.
-	IP.6	Not implemented, reserved for future use*.
-	IP.5	Not implemented, reserved for future use*.
PS	IP.4	Defines the Serial Port interrupt priority level.
PT1	IP.3	Defines the Timer 1 Interrupt priority level.
PX1	IP.2	Defines External Interrupt priority level.
PT0	IP.1	Defines the Timer 0 interrupt priority level.
PX0	IP.0	Defines the External Interrupt 0 priority level.

8.2 LẬP TRÌNH SỬ DỤNG NGẮT

Một chương trình assembly sử dụng ngắt, ví dụ ngắt timer 0 sẽ có cấu trúc như sau:

The screenshot shows a hex editor window titled "vd.a51". The assembly code is as follows:

```
1 ORG 0000H ;RESET VECTOR
2 LJMP MAIN
3 ORG 000BH ;TIMERO INTERRUPT VECTOR
4 LJMP TO_ISR
5 ORG 0030H
6 MAIN:
7 MOV TMOD, #20H ; CONFIGURE TIMER 0 AT MODE 2
8 MOV TH0, #-200
9 MOV IE, #82H ;ENABLE TIMER 0 INTERRUPT
10 SETB TR0
11 JMP $
12 TO_ISR:
13 CPL P1.0
14 RETI
15 END
```

Hình 45 Lập trình ngắt dùng assembly

Khi timer 0 tràn, chương trình sẽ quay về địa chỉ 000BH, từ đó nhảy đến TO_ISR, sau khi gặp lệnh RETI thì quay về lệnh kế tiếp từ thời điểm thực thi ngắt.

LẬP TRÌNH SỬ DỤNG NGẮT TRÊN KIT THÍ NGHIỆM

Trên kit thí nghiệm, bảng vector ngắt được chương trình monitor dời về địa chỉ 2000H. Do đó khi lập trình, ta phải dời bảng vector ngắt lên địa chỉ 2000H như sau

HƯỚNG DẪN THÍ NGHIỆM VI XỬ LÝ

```
vd.a51
1 ORG      2000H ;RESET VECTOR
2 LJMP    MAIN
3 ORG      200BH ;TIMERO INTERRUPT VECTOR
4 LJMP    T0_ISR
5 ORG      2030H
6 MAIN:
7 MOV TMOD, #20H ; CONFIGURE TIMER 0 AT MODE 2
8 MOV TH0, #200
9 MOV IE, #82H ;ENABLE TIMER 0 INTERRUPT
10 SETB   TR0
11 JMP    $
12 T0_ISR:
13 CPL    P1.1
14 RETI
15 END
```

Hình 46 Lập trình ngắt dùng assembly, với bảng vector ngắt tại 2000H

Nếu sử dụng C để lập trình, ta chỉnh các thông số cấu hình cho project như đã nói ở phần lập trình C và lập trình bình thường

```
STARTUP.A51 main.c AT89X51.H
1 #include <AT89X51.H>
2 void initHardware(void);
3
4 void main(void)
5 {
6     initHardware();
7     while(1);
8 }
9 void initHardware()
10 {
11     TMOD = 0X20;
12     TH0 = -200;
13     IE = 0X82;
14     TR0 = 1;
15 }
16 void T0ISR (void) interrupt 1 using 0
17 {
18     P1_0 = ~P1_0;
19 }
```

Hình 47 Lập trình ngắt dùng C

Dòng 15: "void T0ISR (void) interrupt 1 using 0" có ý nghĩa khai báo hàm T0ISR là trình phục vụ ngắt cho ngắt Timer 0 (ngắt số 1 trong bảng sau) và sử dụng bank thanh ghi 0

HƯỚNG DẪN THÍ NGHIỆM VI XỬ LÝ

Number	Interrupt	Symbol
0	External 0	EX0
1	Timer0	IT0
2	External1	EX1
3	Timer1	IT1
4	Serial	ES
5	Timer2	ET2



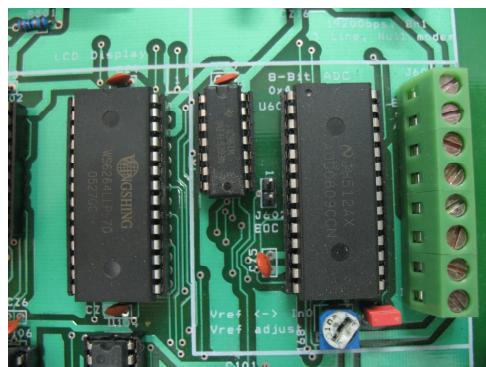
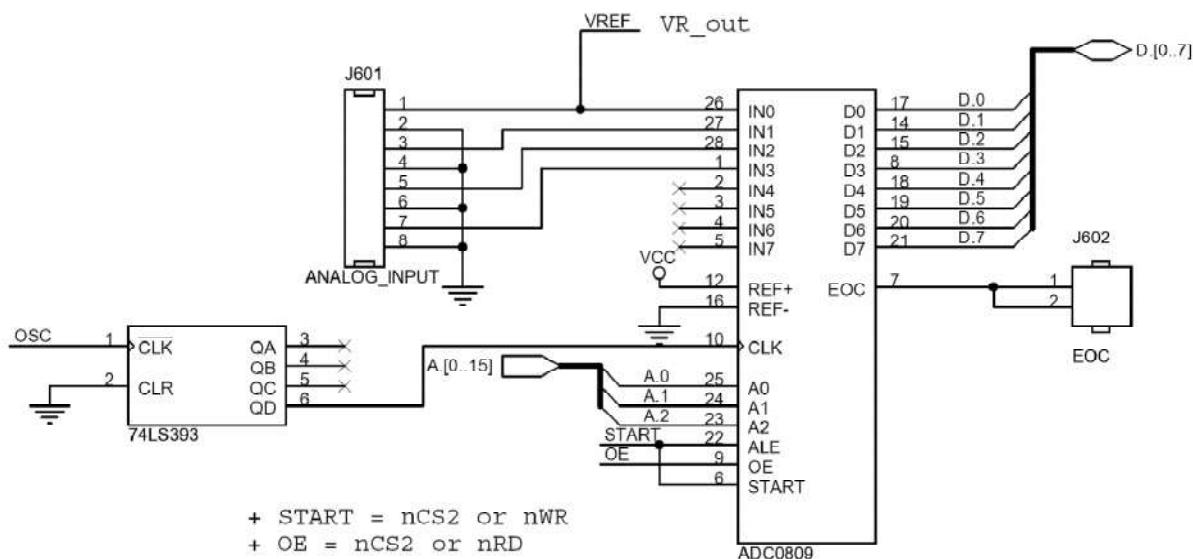
CHƯƠNG 9 THÍ NGHIỆM ĐIỀU KHIỂN ADC

9.1 LÝ THUYẾT CƠ BẢN

Vi mạch ADC được sử dụng để chuyển đổi các tín hiệu từ dạng tương tự sang dạng số. Cũng cần phải nói thêm về tín hiệu tương tự. Đó là các tín hiệu liên tục trong miền thời gian và biên độ. Trong thực tế, hầu hết các tín hiệu vật lý đều tồn tại dưới dạng tương tự. Muốn xử lý được các tín hiệu này bằng các hệ thống số thì cần phải chuyển đổi các tín hiệu tương tự này sang dạng số và vi mạch ADC (Analog to Digital Converter) thực hiện công việc đó.

9.2 THIẾT KẾ PHẦN CỨNG

Kit thí nghiệm sử dụng vi mạch chuyển đổi ADC0809. Sơ đồ kết nối như hình sau



Hình 48 Sơ đồ kết nối với ADC0809

Có thể thấy là ADC0809 được thiết kế để giao tiếp với 8051 thông qua cơ chế 3 bus. Tín hiệu giải mã địa chỉ cho ADC được lấy ra từ mạch giải mã địa chỉ 74x138. Các tín hiệu điều khiển

chủ yếu là START và OE đều được tổ hợp từ tín hiệu giải mã địa chỉ nCS2. Tần số chuyển đổi của ADC (tín hiệu CLK) được chia ra từ tín hiệu dao động của 8051 bằng IC đếm 74x393. Tín hiệu EOC (tích cực khi đã chuyển đổi xong) được nối đến header để sử dụng khi cần).

ADC0809 có thể chuyển đổi lên đến 8 kênh. Trong đó kênh 0 đã được nối sẵn đến một biến trả để người sử dụng có thể thử nghiệm hoạt động của ADC một cách nhanh chóng. Để chọn kênh, 3 đường địa chỉ thấp của bus địa chỉ (A0, A1, A2) đã được sử dụng. Như vậy, người lập trình có thể truy xuất đến 8 địa chỉ thấp nhất của nCS2 để truy xuất đến 8 kênh này.

9.3 PHẦN MỀM GIAO TIẾP

Vì ADC0809 được thiết kế để hoạt động với cơ chế 3 bus và tín hiệu giải mã địa chỉ là từ chân nCS2 nên người lập trình cần dùng các lệnh truy xuất không gian bộ nhớ dữ liệu ngoài để truy cập ADC, cụ thể là lệnh *MOVX*.

Các bước cụ thể gồm: thứ nhất, người lập trình cần yêu cầu ADC0809 thực hiện chuyển đổi một kênh cụ thể bằng cách tích cực tín hiệu START và 3 đường địa chỉ (dùng tín hiệu ALE). Với kết nối phần cứng này, khi tín hiệu START tích cực thì đường ALE cũng tích cực. Như vậy chỉ cần dùng một lệnh ghi ra ngoại vi tại địa chỉ tương ứng là được

MOV DPTR, #nCS2 ; kết hợp địa chỉ ADC với địa chỉ kênh

MOVX @DPTR, A ; ra lệnh chuyển đổi kênh tương ứng

Sau khi chờ một khoảng thời gian (thường lấy khoảng 100us), hoặc chờ mức tích cực trên chân EOC, người lập trình cần đọc từ ADC bằng cách tích cực chân OE. Điều này cũng được thực hiện bằng lệnh đọc ngoại vi

MOVX A, @DPTR

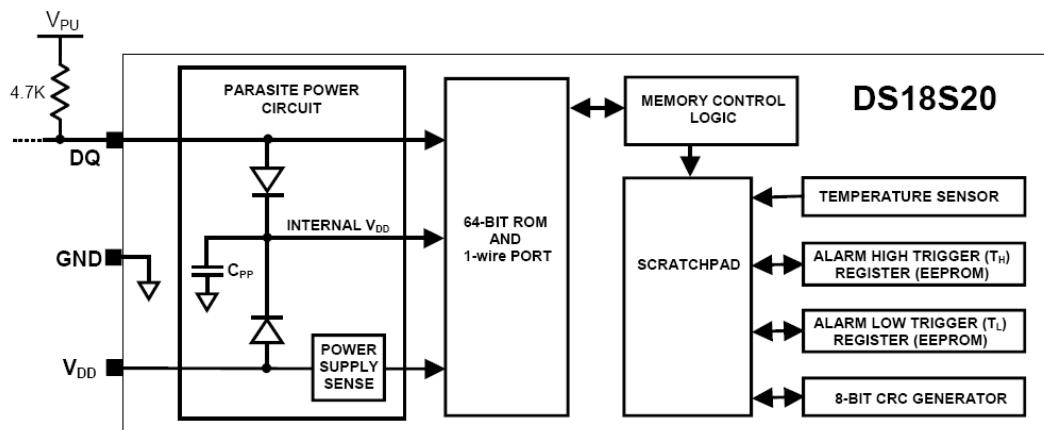
CHƯƠNG 10 THÍ NGHIỆM GIAO TIẾP CẢM BIẾN NHIỆT ĐỘ

10.1 LÝ THUYẾT CƠ BẢN

DS18S20 là vi mạch số đo độ C 9-bit với chức năng cảnh báo khi nhiệt độ vượt quá một ngưỡng trên hoặc dưới lập trình được. Vi mạch thông tin trên bus 1 dây (1-Wire bus), tức là chỉ cần 1 chân port để thông tin giữa cảm biến và MCU. Tầm đo của vi mạch trong khoảng -55°C đến +125°C với độ chính xác tối đa là 0.5°C. Bên cạnh đó, DS18S20 có thể được cấp nguồn trực tiếp thông qua đường dữ liệu.

Mỗi DS18S20 có chứa một mã nối tiếp 64-bit duy nhất, điều đó cho phép nhiều vi mạch có thể cùng hoạt động trên bus 1 dây. DS18S20 gồm 3 chân: GND, DQ, và V_{DD}, trong đó DQ là chân vào và ra của dữ liệu.

Giao thức bus 1 dây (1-Wire) là một giao thức được Dallas định nghĩa. Đường điều khiển cần có điện trở kéo lên vì tất cả các vi mạch đều được kết nối đến bus thông qua 1 cổng 3 trạng thái hoặc cực máng hở (chân DQ). Trong hệ thống bus này, MCU (master) nhận dạng và định địa chỉ các vi mạch trên bus bằng mã địa chỉ 64-bit. Vì mỗi vi mạch có một mã nhận dạng 64-bit duy nhất nên gần như là không giới hạn số vi mạch kết nối trên bus.



10.2 Sơ đồ khối DS18S20

Chức năng cốt lõi của DS18S20 là chuyển đổi trực tiếp nhiệt độ thành tín hiệu số. Ngoài ra cảm biến nhiệt có độ phân giải 9-bit, tương ứng 0.5°C mỗi bước. DS18S20 khởi động ở trạng thái nghỉ công suất thấp, để khởi động đo nhiệt độ và chuyển đổi tương tự sang số, MCU (master) phải thực hiện lệnh Convert T [44h]. Sau khi chuyển đổi, kết quả được lưu vào thanh ghi nhiệt độ 2-byte trong bộ nhớ và DS18S20 trở lại trạng thái nghỉ.

HƯỚNG DẪN THÍ NGHIỆM VI XỬ LÝ

Dữ liệu ngõ ra của DS18S20 được hiệu chỉnh theo độ C. Dữ liệu nhiệt độ được lưu trữ dưới dạng số bù 2 16-bit (mở rộng dấu). Bit dấu (S) chỉ nhiệt độ dương (S=0) hay âm (S=1).

LS Byte	bit 7	bit 6	bit 5	bit 4	bit 3	bit 2	bit 1	bit 0
	2^6	2^5	2^4	2^3	2^2	2^1	2^0	2^{-1}
MS Byte	bit 15	bit 14	bit 13	bit 12	bit 11	bit 10	bit 9	bit 8
	S	S	S	S	S	S	S	S

Định dạng thanh ghi nhiệt độ

TEMPERATURE	DIGITAL OUTPUT (Binary)	DIGITAL OUTPUT (Hex)
+85.0°C*	0000 0000 1010 1010	00AAh
+25.0°C	0000 0000 0011 0010	0032h
+0.5°C	0000 0000 0000 0001	0001h
0°C	0000 0000 0000 0000	0000h
-0.5°C	1111 1111 1111 1111	FFFFh
-25.0°C	1111 1111 1100 1110	FFCEh
-55.0°C	1111 1111 1001 0010	FF92h

*The power-on reset value of the temperature register is +85°C

Mối quan hệ giữa dữ liệu số và nhiệt độ

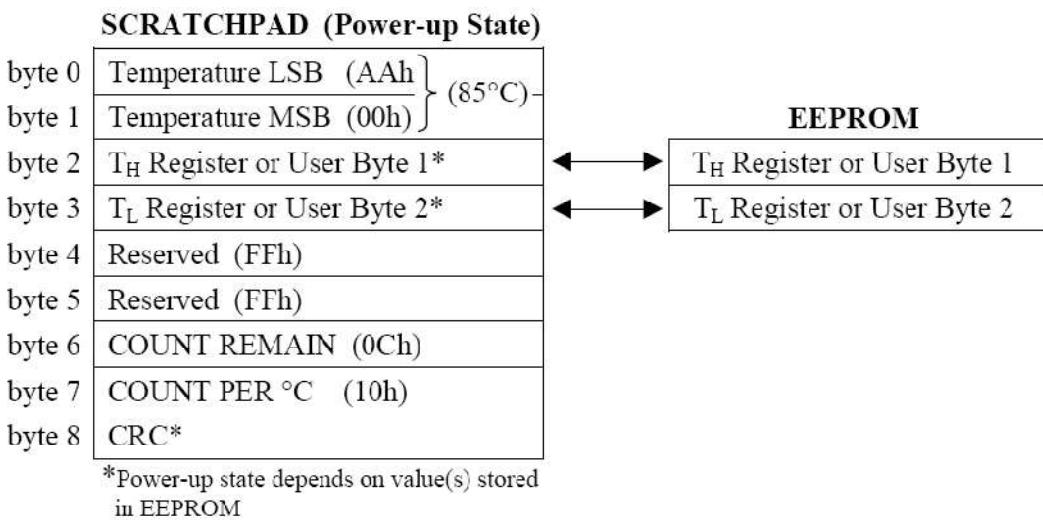
Sau khi DS18S20 thực hiện chuyển đổi nhiệt độ, giá trị này được so sánh với giá trị nhiệt độ cảnh báo được lưu trong 2 thanh ghi 8-bit T_H và T_L . Tuy nhiên chỉ các bit từ 8 đến 1 trong thanh ghi nhiệt độ được so sánh. Nếu nhiệt độ ngoài ngưỡng, tức là cao hơn T_H hoặc thấp hơn T_L , thì một điều kiện cảnh báo xuất hiện và cờ cảnh báo được đặt lên bên trong DS18S20. Nếu nhiệt độ trở lại bình thường thì điều kiện cảnh báo bị tắt ở lần chuyển đổi nhiệt độ kế tiếp.

bit 7	bit 6	bit 5	bit 4	bit 3	bit 2	bit 1	bit 0
S	2^6	2^5	2^5	2^5	2^2	2^1	2^0

Định dạng thanh ghi T_H và T_L

Bộ nhớ DS18S20 được tổ chức như sau

HƯỚNG DẪN THÍ NGHIỆM VI XỬ LÝ



Bản đồ bộ nhớ DS18S20

Bộ nhớ DS18S20 bao gồm bộ nhớ RAM và bộ nhớ EEPROM để lưu trữ thanh ghi cảnh báo T_H và T_L. Byte 0 và 1 của bộ nhớ RAM chứa LSB và MSB của thanh ghi nhiệt độ. Các byte này là chỉ đọc. Byte 2 và 3 dùng để truy xuất thanh ghi T_H và T_L. Byte 4 và 5 được dành riêng để sử dụng bên trong vi mạch. Byte 6 và 7 chứa các thanh ghi COUNT REMAIN và COUNT PER °C, được dùng để tính các kết quả phân giải mở rộng (xem thêm trong mô tả kỹ thuật của DS18S20, phần OPERATION – MEASURING TEMPERATURE). Byte 8 là byte chỉ đọc và chứa CRC của byte từ 0 đến 7 của bộ nhớ RAM.

Dữ liệu được viết vào byte 2 và 3 của RAM sử dụng lệnh Write Scratchpad [4Eh]; dữ liệu được phát với LSB của byte 2. Bộ nhớ RAM có thể được đọc sử dụng lệnh Read Scratchpad [BEh] sau khi viết dữ liệu. Khi đọc bộ nhớ RAM, dữ liệu được phát với LSB của byte 0. Để chuyển T_H và T_L từ RAM vào bộ nhớ EEPROM, MCU (master) phải thực thi lệnh Copy Scratchpad [48h].

Hệ thống bus 1 dây (1-Wire bus)

Hệ thống này sử dụng một master (chủ của bus) để điều khiển một hoặc nhiều slave (tớ của bus). DS18S20 luôn là slave. Khi chỉ có một slave, hệ thống được gọi là “single-drop”, khi có nhiều slave, hệ thống được gọi là “multi-drop”. Tất cả dữ liệu và lệnh đều được phát với LSB trước tiên.

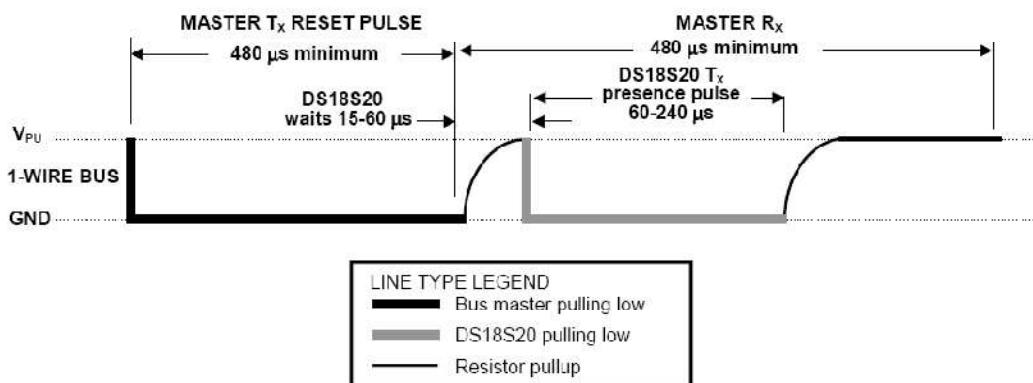
Mỗi thiết bị giao tiếp với đường dữ liệu thông qua công 3 trạng thái hoặc cực máng hở. Khả năng này cho phép thiết bị giải phóng đường dữ liệu khi thiết bị không phát dữ liệu đến bus, do đó bus có sẵn để các thiết bị khác sử dụng. Bus 1 dây cần một điện trở kéo lên khoảng 5k,

HƯỚNG DẪN THÍ NGHIỆM VI XỬ LÝ

do đó trạng thái nghỉ của bus là mức cao. Nếu bus được giữ ở mức thấp nhiều hơn 480μs, các thiết bị trên bus sẽ bị reset.

Chuỗi thao tác để truy xuất DS18S20 như sau:

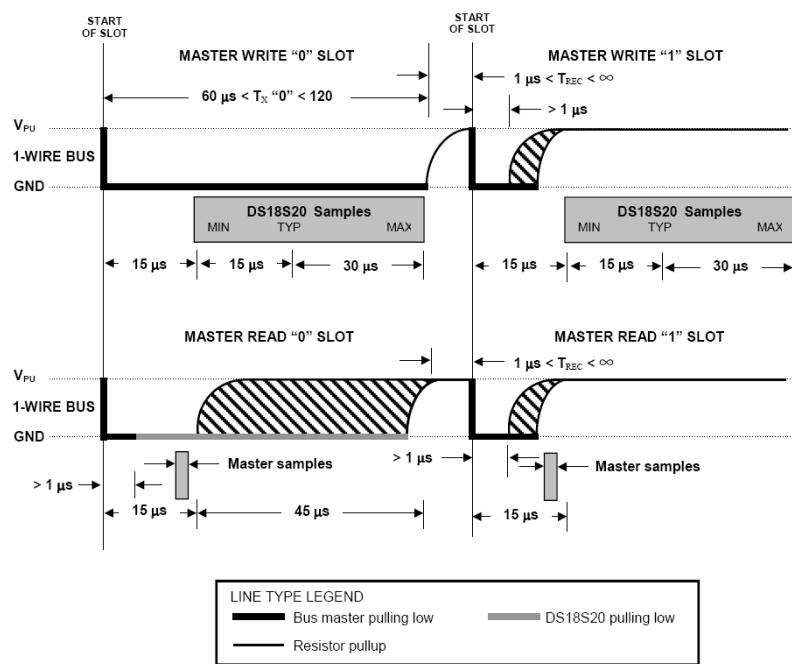
- Khởi động: bao gồm một xung reset được phát bởi master, và xung có mặt được phát bởi slave.



Định thì khởi động

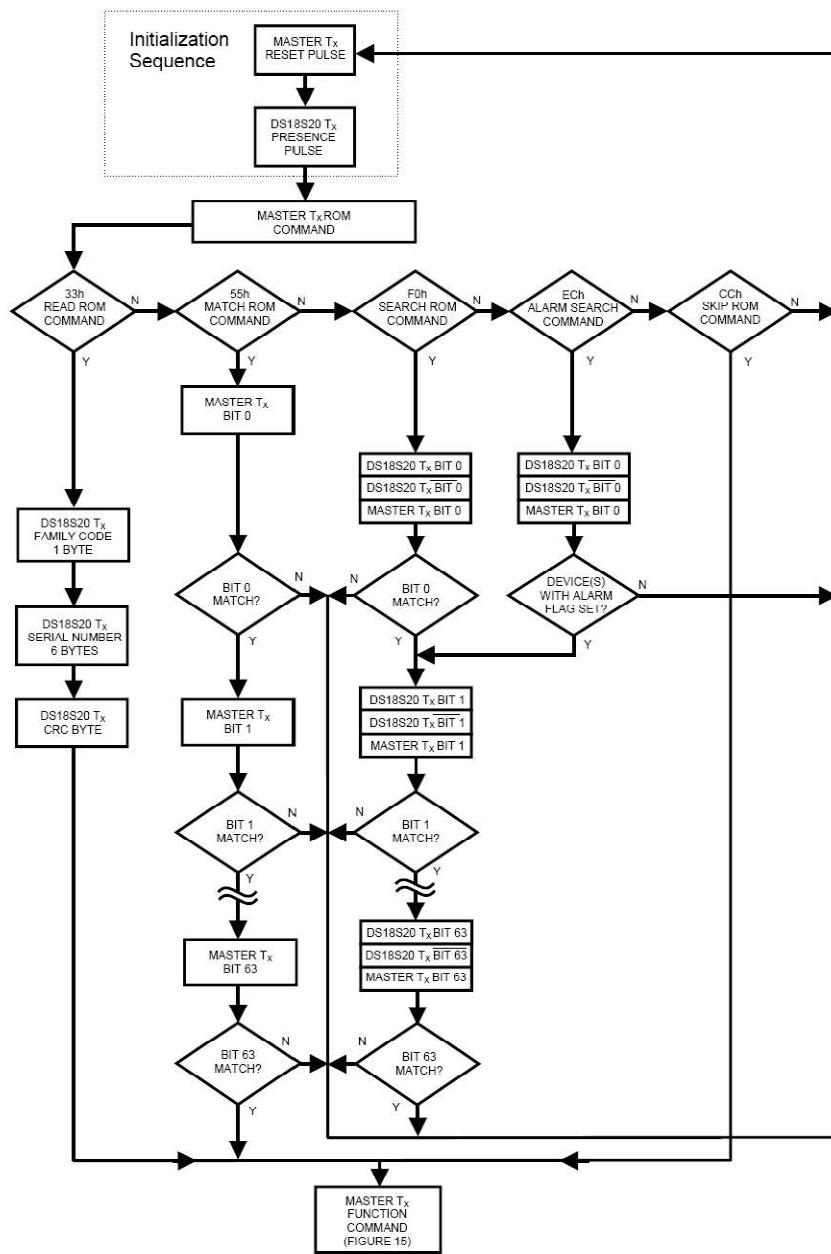
- Lệnh ROM: sau khi master phát hiện được sự có mặt của slave, nó có thể ra lệnh ROM. Các lệnh này hoạt động trên các mã 64-bit duy nhất của mỗi slave và cho phép master truy xuất đến một thiết bị cụ thể trên bus. Các lệnh này cũng cho phép master xác định loại và số lượng thiết bị trên bus. Có 5 lệnh ROM, mỗi lệnh dài 8-bit, bao gồm SEARCH ROM [F0h], READ ROM [33h], MATCH ROM [55h], SKIP ROM [CCh], ALARM SEARCH [ECh].
- Lệnh chức năng DS18S20: sau khi master đã dùng lệnh ROM để xác định được slave mà nó cần thông tin, master có thể ra các lệnh chức năng DS18S20. Các lệnh này bao gồm: CONVERT T [44h], WRITE SCRATCHPAD [4Eh], READ SCRATCHPAD [BEh], COPY SCRATCHPAD [48h], RECALL E² [B8h], READ POWER SUPPLY [B4h].

HƯỚNG DẪN THÍ NGHIỆM VI XỬ LÝ



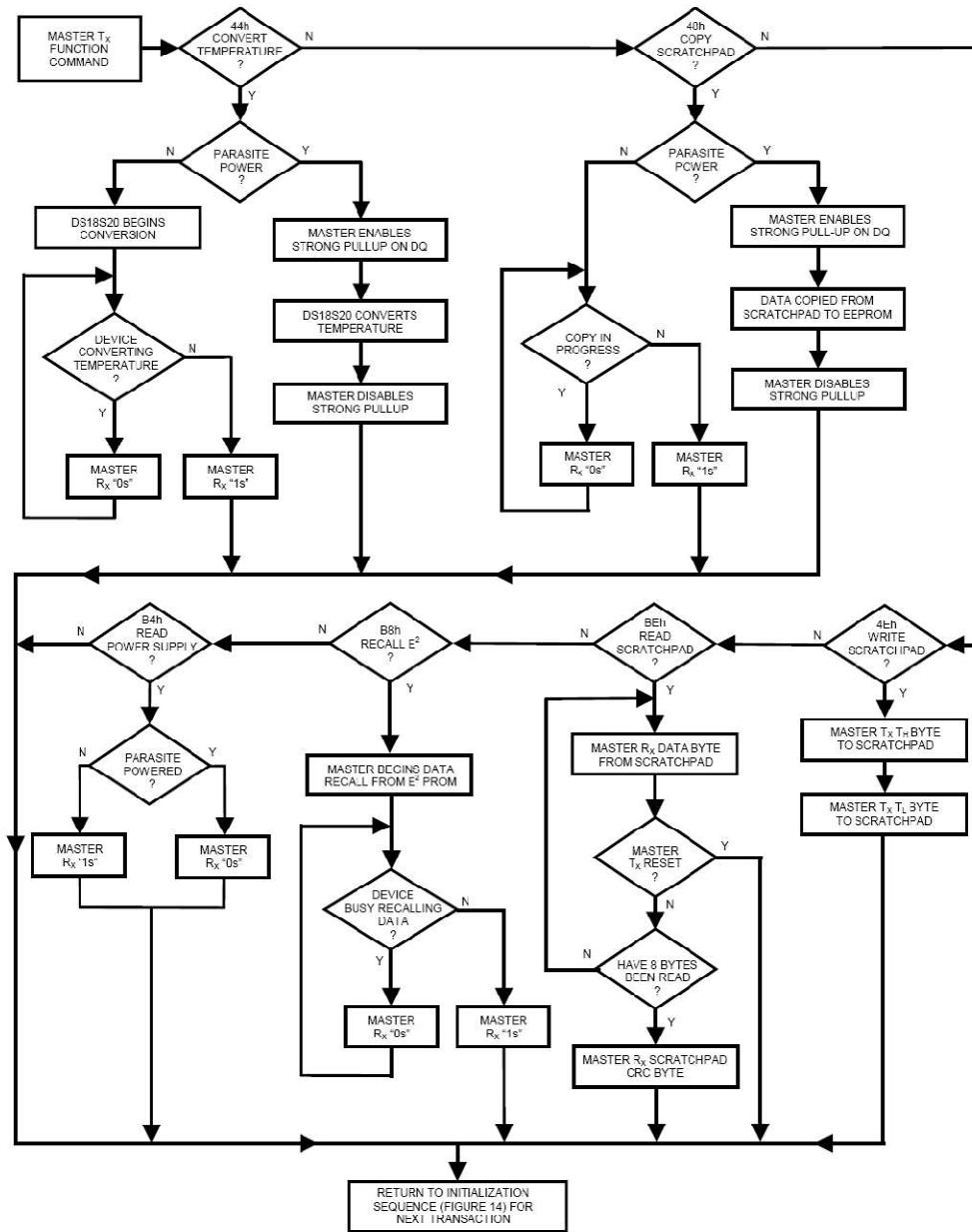
Định thi khe thời gian đọc/ghi

HƯỚNG DẪN THÍ NGHIỆM VI XỬ LÝ



Lưu đồ lệnh ROM

HƯỚNG DẪN THÍ NGHIỆM VI XỬ LÝ



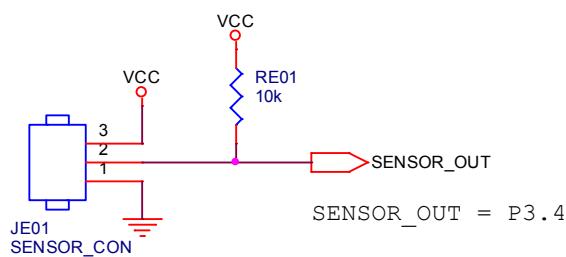
Lưu đồ lệnh chức năng DS18S20

Chuỗi thao tác này là bắt buộc cho mỗi lần truy xuất DS18S20.

10.3 THIẾT KẾ PHẦN CỨNG

Tín hiệu giao tiếp giữa 8051 và DS18S20 thông qua chuẩn 1-Wire là P3.4. Như vậy, người sử dụng cần bật ON switch này trên SW-CC4.

HƯỚNG DẪN THÍ NGHIỆM VI XỬ LÝ



Hình 49 Sơ đồ kết nối DS18B20

10.4 PHẦN MỀM GIAO TIẾP

Người lập trình cần dùng các lệnh *SETB*, *CLR* để thao tác trên bit P3.4 theo giao thức điều khiển DS18S20.

CHƯƠNG 11 THÍ NGHIỆM VỚI VI MẠCH DAC MCP4922

11.1 LÝ THUYẾT CƠ BẢN

Vi mạch chuyển đổi DAC được sử dụng để chuyển tín hiệu từ dạng số sang dạng tương tự. DAC thường được dùng để tạo ngõ ra giao tiếp giữa hệ thống số với thế giới analog bên ngoài.

Vi mạch DAC MCP4922 là vi mạch chuyển đổi số sang tương tự 12-bit dùng giao tiếp SPI của Microchip. Dưới đây là bảng chức năng chân của vi mạch

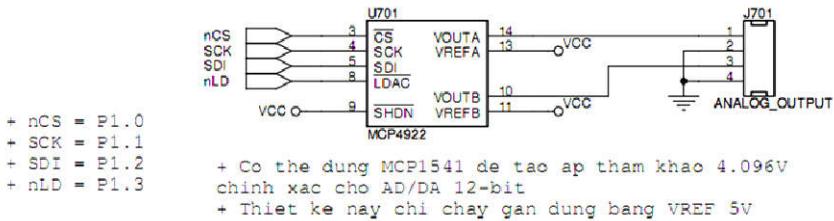
MCP4921 Pin No.	MCP4922 Pin No.	Symbol	Function
1	1	V_{DD}	Positive Power Supply Input (2.7V to 5.5V)
—	2	NC	No Connection
2	3	\overline{CS}	Chip Select Input
3	4	SCK	Serial Clock Input
4	5	SDI	Serial Data Input
—	6	NC	No Connection
—	7	NC	No Connection
5	8	LDAC	Synchronization input used to transfer DAC settings from serial latches to the output latches.
—	9	\overline{SHDN}	Hardware Shutdown Input
—	10	V_{OUTB}	DAC _B Output
—	11	V_{REFD}	DAC _D Voltage Input (AV_{SS} to V_{DD})
7	12	AV_{SS}	Analog ground
6	13	V_{HFA}	DAC _A Voltage Input (AV_{SS} to V_{DD})
8	14	V_{OUTA}	DAC _A Output

Chân V_{DD} là ngõ vào nguồn cung cấp dương, điện áp có thể thay đổi từ 2.7V đến 5.5V. Chân nCS là chân ngõ vào cho phép của vi mạch. Chân này phải được giữ ở mức thấp để cho phép vi mạch làm việc. SCK là ngõ vào clock nối tiếp. SDI là chân ngõ vào dữ liệu nối tiếp tương thích SPI. Ngõ vào nLDAC ở mức thấp sẽ cho phép chuyển thanh ghi chốt ngõ vào sang thanh ghi DAC.(chốt ngõ ra). Chân này cũng có thể nối đến GND nếu dùng cạnh lê của nCS. Chân nSHDN ở mức thấp sẽ làm cho DAC ở vào trạng thái nghỉ. Các ngõ ra DAC là V_{OUTA} và V_{OUTB} . Tín hiệu ngõ ra này thay đổi giữa AV_{SS} và V_{DD} , trong đó chân AV_{SS} là chân GND của tín hiệu analog. Các ngõ vào V_{REFA} và V_{REFB} là các ngõ vào điện áp tham khảo.

11.2 THIẾT KẾ PHẦN CỨNG

Trên kit thí nghiệm, vi mạch MCP4922 được thiết kế giao tiếp với MCU thông qua giao tiếp SPI. Các tín hiệu SCK, SDI, nLDAC, và DAC_nCS từ MCP4922 được nối đến DIP-SW4 cấu hình trước khi nối vào MCU. Do đó, cần bật ON các switch này để kết nối tín hiệu điều khiển của EME-MC8 vào module MCU tương ứng.

HƯỚNG DẪN THÍ NGHIỆM VI XỬ LÝ



Như vậy, để điều khiển các tín hiệu này, người lập trình cần truy xuất đến các bit trên port 1 bằng các lệnh thông qua thanh ghi port P1.

11.3 PHẦN MỀM GIAO TIẾP

Vì tín hiệu điều khiển MCP4922 được kết nối trực tiếp đến các bit của port 1 nên các lệnh cho phép thao tác trên thanh ghi P1. Ví dụ đoạn mã sau có thể được dùng để dịch 1 bit dữ liệu từ 8051 vào MCP4922

```
CLR      SCK          ; tạo xung clock  
MOV      SDI, C        ; dịch cờ C  
SETB    SCK          ; vào MCP4922
```

Ví dụ để xuất các mức điện áp 0V, 1V, 2.5V, 4V, và 5V ra ngõ ra của MCP4922, người lập trình có thể làm như sau:

Vì MCP4922 là DAC 12-bit với điện áp tham khảo là 5V nên độ phân giải tương ứng là $5V/4096 = 1.22mV/LSB$. Như vậy, người lập trình có thể tính ra được các giá trị nhị phân cần xuất ra MCP4922 để tạo ra các mức điện áp như yêu cầu.

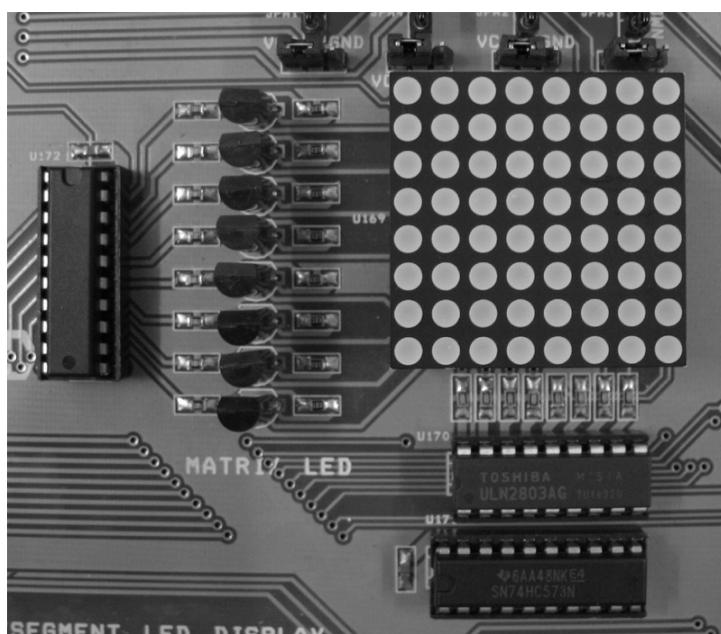
Vì độ phân giải này có thể tạo ra sai số ở ngõ ra của MCP4922 nên người thiết kế có thể dùng vi mạch tạo điện áp tham khảo chính xác MCP1541. Vi mạch này tạo điện áp 4.096V chính xác ở ngõ ra, do đó, MCP4922 sẽ có độ phân giải là 1mV/LSB.

Dùng các chương trình dịch dữ liệu vào và ra ở phần trên để ghi dữ liệu vào MCP4922.

CHƯƠNG 12 THÍ NGHIỆM VỚI LED MA TRẬN

12.1 LÝ THUYẾT CƠ BẢN

Led ma trận có thể dùng để hiển thị thông tin đến từng điểm ảnh. Do đó, led ma trận có thể biểu diễn được chữ số, chữ cái, và các hình ảnh khác. Led ma trận có thể là dạng đơn sắc (thường là màu đỏ), hoặc đa sắc. Đối với dạng đa sắc, một điểm ảnh chứa đến 2 led với màu khác nhau (ví dụ màu đỏ và màu xanh lá cây). Sử dụng phối hợp hai màu này có thể sinh ra thêm một số màu trung gian. Quá trình này có thể được thực hiện bằng cách thay đổi độ rộng xung điều khiển của từng màu (tức là thời gian sáng của từng led màu). Đối với led ma trận, cả hai chân anode và cathode của led đều được đưa ra thành tín hiệu điều khiển.



Led ma trận

Led ma trận được hiển thị bằng phương pháp quét từng hàng hoặc từng cột. Chu kỳ quét thường là 1/8 (hoặc 1/16). Điều này có nghĩa là một khung hình sẽ được hiển thị đầy đủ sau 8 (hoặc 16) lần quét hàng (hoặc cột). Khi quét led ma trận cần quan tâm đến dòng trung bình và dòng định qua led vì thông số này sẽ quyết định độ sáng của led. Ví dụ với led 8 hàng dùng phương pháp quét cột, nếu mỗi led cần dòng trung bình là 5mA để sáng thì dòng trung bình tối đa của 1 cột sẽ là $8 \text{ led} \times 5\text{mA} = 40\text{mA}$. Tuy nhiên vì led chỉ hiển thị trong 1/8 (hoặc 1/16) của chu kỳ quét nên dòng định của xung quét phải đạt đến $8 \times 40\text{mA} = 360\text{mA}$ (hoặc 720mA). Các chân của vi mạch số thường không thể cung cấp đủ dòng như thế này. Do đó các tín hiệu

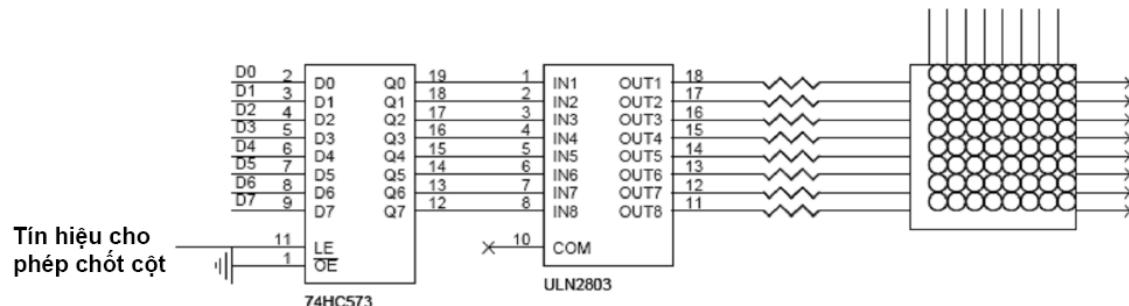
HƯỚNG DẪN THÍ NGHIỆM VI XỬ LÝ

lái led cần phải được đệm với các linh kiện dòng lớn như transistor để đảm bảo đủ dòng cho từng led.

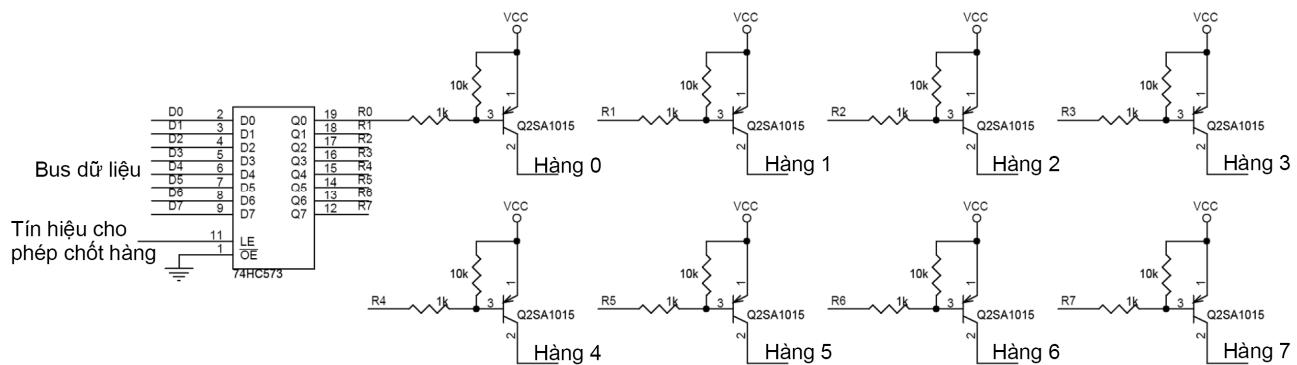
Để hiển thị được một ký tự chữ hoặc số, chương trình cần một bảng font để chuyển giữa giá trị bên trong MCU thành giá trị hiển thị lên led ma trận tương tự như với led 7 đoạn. Bảng font này có thể được ghi trực tiếp vào chương trình.

12.2 THIẾT KẾ PHẦN CỨNG

Led ma trận trên kit là led có kích thước 8x8 với phương pháp quét led là quét cột (vì mỗi cột đã được đệm với dãy transistor ULN2803 có dòng tối đa lên đến 500mA). Led có hàng là anode và cột là cathode. Thiết kế sẽ sử dụng vi mạch ULN2803 cho phép kéo dòng lên đến 500mA ở phía cột và BJT B562 ở phía hàng vì phía cột sẽ gánh dòng tối đa lên đến 8 led đồng thời. Để hiển thị một điểm ảnh thì dữ liệu xuất ra trên hàng phải là mức 0 vì khi đó khóa BJT sẽ dẫn làm cho hàng được cấp nguồn V_{CC}, dữ liệu xuất trên cột sẽ là mức 1 vì ULN2803 hoạt động như công đảo nên tín hiệu lái cột sẽ có điện áp là mức 0 (tức là gần GND) cho phép các led trên cột tương ứng được sáng.



Hình 50 Sơ đồ mạch phía cột của khối led ma trận



Hình 51 Sơ đồ mạch phía hàng của khối led ma trận

HƯỚNG DẪN THÍ NGHIỆM VI XỬ LÝ

Các tín hiệu dữ liệu hàng và cột được thiết kế theo phương pháp 3 bus. Do đó, led ma trận được điều khiển thông qua hai vi mạch chốt 74x573, một chốt dữ liệu của từng hàng và một dùng để chọn cột quét. Tín hiệu cho phép chốt dữ liệu được tổ hợp từ tín hiệu giải mã địa chỉ và tín hiệu cho phép ghi nWR. Giải mã địa chỉ được thiết kế bằng 74138 với địa chỉ cụ thể của hàng và cột có thể xem trong bản đồ bộ nhớ.

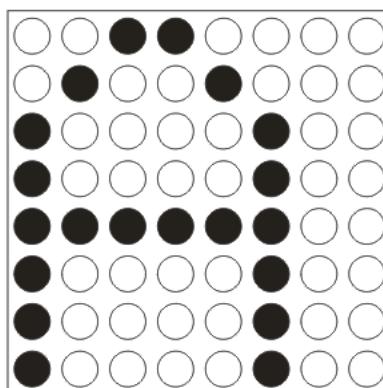
12.3 PHẦN MỀM GIAO TIẾP

Để có thể hiển thị lên led ma trận, phương pháp được chọn là quét cột. Mỗi cột sẽ được hiển thị trong một khoảng thời gian nhất định sau đó chuyển sang cột kế tiếp. Vì led ma trận có kích thước 8x8 nên xung dòng quét cột sẽ là 1/8. Để đảm bảo hiển thị rõ, toàn bộ bảng hiển thị phải được quét trong vòng trước 40ms (25 hình/s). Như vậy mỗi cột sẽ có thời gian hiển thị tối đa là 5ms (tuy nhiên nên dùng thời gian trễ là 1ms). Nếu thời gian hiển thị nhỏ hơn thì tần số quét sẽ tăng, tuy nhiên lúc này dòng trung bình có thể không đủ để led sáng tốt. Do đó không thể giảm thời gian quét xuống quá nhỏ, trung bình nên chọn vào khoảng 1-3ms (giá trị cụ thể còn tùy thuộc vào loại led).

Trước tiên, người lập trình cần chuyển đổi giá trị cần hiển thị thành dạng có thể hiển thị lên led ma trận bằng cách dùng phương pháp tra bảng để tra trên bảng font. **Lưu ý** là một dữ liệu hiển thị sẽ được chuyển thành 8 byte dữ liệu hiển thị (vì led ma trận có 8 cột).

Để hiển thị được 1 cột, người lập trình cần xuất 1 byte ra hàng (1 byte trong 8 byte tra được từ bảng font), cho phép cột đó, chờ một khoảng thời gian và chuyển sang hiển thị cột kế tiếp với qui trình tương tự. Sau khi hết 8 cột, qui trình hiển thị được lặp lại từ đầu.

Ví dụ ta muốn hiển thị ký tự chữ A lên led ma trận.



Hình 52 Chữ A trên led ma trận 8x8

HƯỚNG DẪN THÍ NGHIỆM VI XỬ LÝ

Để có thể hiển thị được chữ A lên led ma trận, trước tiên cần xác định địa chỉ của chốt hàng và cột bằng bản đồ bộ nhớ. Địa chỉ này sẽ được dùng để xuất các dữ liệu ra led.

Bước kế tiếp là chuẩn bị dữ liệu để xuất. Vì phương pháp quét là quét cột nên dữ liệu xuất ra cột sẽ lần lượt là 10000000B, 01000000B, 00100000B, 00010000B, 00001000B, 00000100B, 00000010B, 00000001B (mức tích cực của 1 cột là mức cao). Mỗi lần xuất cách nhau 1ms, sau khi đã chuẩn bị dữ liệu trên hàng. Dữ liệu xuất trên hàng sẽ được lấy ra bằng phương pháp tra bảng. Nhìn vào hình ảnh của chữ A dự định hiển thị, ta có thể xác định được các mã xuất ra 8 hàng của 1 cột lần lượt là 03h, 0EDh, 0EEh, 0EEh, 0EDh, 03h, OFFh, OFFh (đây chính là font của chữ A). Dùng dẫn xuất DB để khai báo chuỗi dữ liệu này như sau

CharA: *DB* *03h, 0EDh, 0EEh, 0EEh, 0EDh, 03h, OFFh, OFFh*

Dẫn xuất DB không phải là một lệnh của MCU. Đó là một hướng dẫn để trình biên dịch ghi sẵn các giá trị cho từng byte vào trong bộ nhớ chương trình tại địa chỉ có nhãn là *CharA*. Dẫn xuất này nên được đặt trong phân đoạn dữ liệu hoặc cuối của chương trình chính.

MCU sẽ dùng chuỗi lệnh sau để thực hiện tra bảng (look-up table)

```
MOV      A, #n          ; n = cột cần hiển thị  
MOV      DPTR, #CharA  
MOVC    A, @A+DPTR
```

Để đọc được mã của chữ A cần hiển thị tại cột thứ n. Lúc này n sẽ thay đổi từ 0 (cho cột đầu tiên) đến 7 (cho cột cuối cùng). Quá trình này sẽ được lặp liên tục.