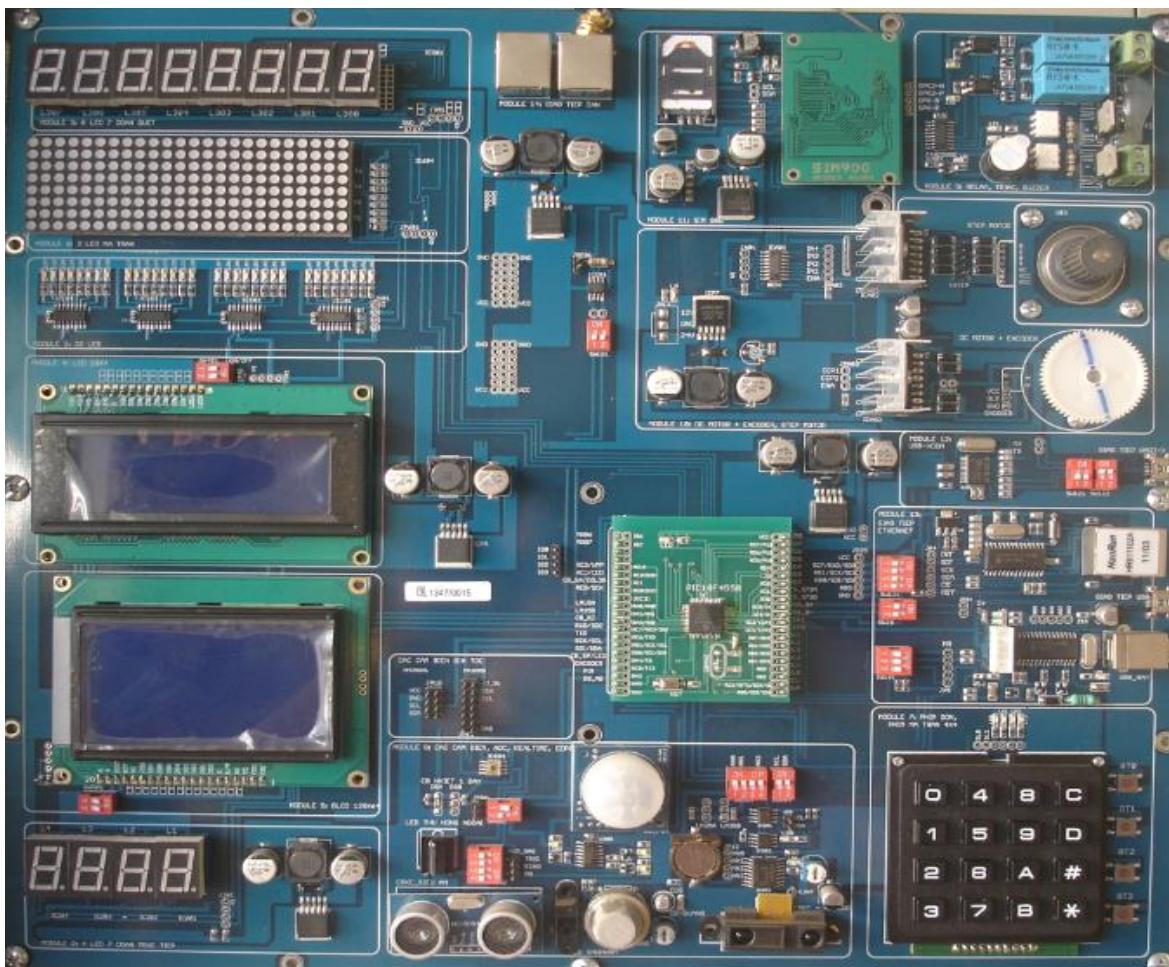


ĐẠI HỌC SƯ PHẠM KỸ THUẬT TP. HỒ CHÍ MINH
KHOA ĐIỆN – ĐIỆN TỬ
BỘ MÔN ĐIỆN TỬ – CÔNG NGHIỆP



GIÁO TRÌNH THỰC HÀNH

VI ĐIỀU KHIỂN PIC



**NGUYỄN ĐÌNH PHÚ
PHAN VÂN HOÀN
TRƯƠNG NGỌC ANH**

TP HCM, 8/2017

LỜI NÓI ĐẦU

Thực hành Vi xử lý hoặc vi điều khiển là môn học cơ sở cho tất cả các ngành Kỹ Thuật Điện – Điện Tử, Điện Tử - Truyền Thông, Tự Động Điều Khiển, Kỹ Thuật Máy Tính và Cơ Điện Tử, sau khi đã học xong lý thuyết với mục đích tiếp cận thực tế lập trình điều khiển và ứng dụng của vi điều khiển từ cơ bản đến nâng cao nhằm ôn tập các kiến thức đã học, rèn luyện các kỹ năng lập trình, kỹ năng gỡ rối, kiểm tra, phân tích, suy luận, đánh giá.

Các bài thực hành được biên soạn theo kit thực hành vi điều khiển PIC 18F4550 với cấu hình phần cứng giao tiếp với nhiều ngoại vi như led đơn dùng thanh ghi dịch, led 7 dùng thanh ghi dịch, led 7 đoạn quét, phím đơn, phím ma trận, LCD, GLCD, led ma trận 2 màu xanh đỏ, động cơ bước và động cơ DC, cùng nhiều loại cảm biến khác nhau như cảm biến nhiệt độ LM35, cảm biến nhiệt 1 dây, cảm biến khoảng cách dùng siêu âm, dùng hồng ngoại, cảm biến chuyển động, ...

Các bài thực hành được biên soạn từ dễ đến khó, từ đơn giản đến phức tạp về phần cứng lẫn phần mềm. Qua từng chương thì mức độ khó về phần cứng và phần mềm sẽ tăng dần do tính chất kết hợp. Mỗi chương có rất nhiều bài thực hành và bài tập, mục đích của từng bài rõ ràng, cùng một vấn đề nhưng được viết ở nhiều dạng khác nhau nhằm giúp người học có thể hiểu dễ dàng và nhanh chóng.

Các phần cứng có liên quan đều được trình bày lại chi tiết, rõ ràng, có lưu đồ giải thuật, có tính toán và giải thích rõ ràng.

Tài liệu được biên soạn thành 10 chương:

Chương 1: Cấu hình kit thực hành vi điều khiển.

Chương 2: Sử dụng phần mềm CCS và PICKIT.

Chương 3: Thực hành module 1: 32 led đơn dùng thanh ghi dịch 74HC595, nút nhấn, bàn phím ma trận.

Chương 4: Thực hành module 2: 4 led 7 đoạn anode chung dùng thanh ghi dịch 74HC595, timer/counter.

Chương 5: Thực hành module 3: 8 led 7 đoạn anode chung kết nối theo phương pháp quét dùng thanh ghi dịch MBI5026, timer/counter.

Chương 6: Thực hành module 4: LCD 20X4, GLCD 128X64 dùng thanh ghi dịch MBI5026.

Chương 7: Thực hành chuyển đổi tương tự sang số ADC, các cảm biến.

Chương 8: Thực hành module 5: Real time DS13B07, ADC-DAC PCF 8591, Eeprom nối tiếp AT24C256 theo chuẩn I2C và các cảm biến.

Chương 9: Thực hành module 6: động cơ bước, động cơ DC và encoder, PWM, điều khiển PID.

Chương 10: Thực hành module 7: điều khiển led ma trận 2 màu.

Nội dung chương 1 trình bày chi tiết về cấu hình kết nối phần cứng của kit thực hành vi điều khiển PIC 18F4550. Do kit được thiết kế theo tiêu chí không còn dùng dây bus để kết nối giữa port của vi điều khiển với các đối tượng điều khiển nên có dùng thêm nhiều IC mở rộng port như IC chốt 74HC573, thanh ghi dịch nối tiếp sang song song 8 bit 74HC595 và 16 bit MBI5026, người học cần phải hiểu rõ vai trò, chức năng của các IC này. Mỗi module đều có trình bày sơ đồ kết nối phần cứng và có giải thích. Người học cần phải hiểu rõ kết nối phần cứng mới khai thác hết được các tính năng độc lập điều khiển đơn giản cũng như kết hợp nhiều module để điều khiển cho một chức năng lớn. Đây là chương quan trọng trình bày hết mọi nguyên lý kết nối phần cứng, phải hiểu rõ chúng thì mới hiểu được các thư viện và chương trình.

Nội dung chương 2 trình bày chi tiết về cách sử dụng phần mềm để lập trình các chương trình cho vi điều khiển PIC, cách biên dịch, cách tìm lỗi và phần mềm để nạp code file HEX vào bộ nhớ của vi điều khiển PIC.

Nội dung chương 3 thực hành điều khiển module 32 led đơn, nút nhấn, bàn phím ma trận, nội dung trình bày rất chi tiết, từng bước, như giới thiệu các hàm đã viết trong thư viện và giải thích chức năng các hàm, các biến và các tên đã định nghĩa, ... có rất nhiều hàm đã viết sẵn nhằm giúp bạn thực hiện các ứng dụng điều khiển một cách nhanh chóng. Sau đó bắt đầu viết các chương trình mẫu điều khiển 32 led từ đơn giản đến nâng cao, có nhiều cách viết và có nhiều bài tập để luyện tập dựa vào các bài mẫu. Người học cần hiểu các bài mẫu để làm các bài tập vì đó là các bài mở rộng nhằm ứng dụng các kiến thức đã học. Tiếp theo là các chương trình giao tiếp với nút nhấn đơn, bài đơn giản, bài cho thấy hiện tượng dội của phím nhấn và cách chống dội. Cuối cùng là giao tiếp với bàn phím ma trận.

Nội dung chương 4 thực hành điều khiển module 4 led 7 đoạn dùng 4 IC thanh ghi dịch mở rộng. Nội dung được trình bày rất chi tiết, từng bước như giới thiệu các hàm đã viết trong thư viện và giải thích chức năng các hàm, các biến và các tên đã định nghĩa liên quan đến module 4 led 7 đoạn, ... có rất nhiều hàm đã viết sẵn nhằm giúp bạn thực hiện các ứng dụng điều khiển một cách nhanh chóng.

Sau đó bắt đầu viết các chương trình mẫu điều khiển 4 led 7 đoạn từ đơn giản đến nâng cao. Kết hợp với các bộ định thời timer và counter và nguồn phát hiện xung khi có sản phẩm đi qua để thực hành các chương trình đếm xung ngoại. Kết hợp với module 32 led đơn, nút nhấn và bàn phím ma trận để thực hiện yêu cầu điều khiển phức tạp hơn.

Nội dung chương 5 thực hành điều khiển module 8 led 7 đoạn kết nối theo phương pháp quét, dùng IC thanh ghi dịch mở rộng. Nội dung trình bày rất chi tiết, từng bước như giới thiệu các hàm đã viết trong thư viện và giải thích chức năng các hàm, các biến và các tên đã định nghĩa liên quan đến module 8 led 7 đoạn, ... Có rất nhiều hàm đã viết sẵn nhằm giúp bạn thực hiện các ứng dụng điều khiển một cách nhanh chóng.

Sau đó bắt đầu viết các chương trình mẫu điều khiển 8 led 7 đoạn từ đơn giản đến nâng cao, từ 2 led mở rộng dần đến 8 led. Kết hợp với các bộ định thời timer và counter để đếm thời gian làm động hồ hiển thị giờ phút giây, kết hợp với nút nhấn để chỉnh thời gian. Kết hợp với module 32 led đơn, 4 led 7 đoạn, nút nhấn và bàn phím ma trận để thực hiện yêu cầu điều khiển phức tạp hơn.

Nội dung chương 6 thực hành điều khiển module LCD 20x4, GLCD 128x64 dùng IC thanh ghi dịch mở rộng. Nội dung trình bày đầy đủ phần lý thuyết LCD, GLCD, tập lệnh của LCD, GLCD. Nguyên lý khởi tạo và hiển thị text, hiển thị hình ảnh. Từng bước giới thiệu các hàm đã viết trong thư viện và giải thích chức năng các hàm, các biến và các tên đã định nghĩa liên quan đến module LCD và GLCD. Có rất nhiều hàm đã viết sẵn nhằm giúp bạn thực hiện các ứng dụng điều khiển một cách nhanh chóng.

Tiếp theo phần viết các chương trình mẫu điều khiển hiển thị trên LCD, GLCD từ đơn giản đến nâng cao. Kết hợp với module 32 led đơn, 4 led 7 đoạn, nút nhấn và bàn phím ma trận để thực hiện yêu cầu điều khiển phức tạp hơn.

Nội dung chương 7 thực hành phần chuyển đổi ADC tích hợp trong vi điều khiển PIC 18F4550. Trình bày đầy đủ nguyên lý hoạt động của ADC, các lệnh liên quan đến ADC. Trình bày lý thuyết cảm biến LM35, mạch giao tiếp các cảm biến với vi điều khiển, cách tính toán độ phân giải.

Tiếp theo là phần viết các chương trình mẫu đọc và chuyển đổi nhiệt độ hiển thị trên led 7 đoạn, trên LCD, trên GLCD từ đơn giản đến nâng cao.

Thực hiện tương tự cho cảm biến đo khoảng cách dùng sóng hồng ngoại và cảm biến nhiệt một dây và cảm biến đo khoảng cách dùng sóng siêu âm.

Kết hợp với module 32 led đơn, 4 led 7 đoạn, nút nhấn và bàn phím ma trận để thực hiện yêu cầu điều khiển phức tạp hơn, tổng hợp nhiều chương trình điều khiển với nhiều chức năng.

Nội dung chương 8 thực hành phần giao tiếp vi điều khiển PIC 18F4550 với các thiết bị ngoại vi qua chuẩn I2C. Trình bày đầy đủ nguyên lý hoạt động của các thiết bị ngoại vi như IC thời gian thực DS13B07, IC ADC-DAC PCF8591, IC nhớ nối tiếp AT24C256.

Tiếp theo là phần viết các chương trình mẫu đọc khởi tạo, cài đặt thời gian các và chuyển đổi nhiệt độ hiển thị trên led 7 đoạn, trên LCD, trên GLCD từ đơn giản đến nâng cao.

Kết hợp với module 32 led đơn, 4 led 7 đoạn, nút nhấn và bàn phím ma trận để thực hiện yêu cầu điều khiển phức tạp hơn, tổng hợp nhiều chương trình điều khiển với nhiều chức năng.

Nội dung chương 9 thực hành phần giao tiếp vi điều khiển PIC 18F4550 với động cơ bước và động cơ DC thông qua IC giao tiếp công suất L298. Trình bày đầy đủ nguyên lý hoạt động của động cơ bước, động cơ DC, sơ đồ mạch giao tiếp, nguyên lý điều khiển cơ bản và nguyên lý điều khiển điều chế xung PWM.

Tiếp theo là phần viết các chương trình mẫu điều khiển động cơ từ cơ bản đến nâng cao, điều khiển ổn định tốc độ theo giải thuật PID.

Kết hợp với các module khác để thực hiện yêu cầu điều khiển phức tạp hơn, tổng hợp nhiều chương trình điều khiển với nhiều chức năng.

Nội dung chương 10 thực hành điều khiển module led ma trận 2 màu dùng IC thanh ghi dịch mở rộng. Nội dung trình bày đầy đủ phần lý thuyết mạch giao tiếp giữa vi điều khiển với 3 led ma trận. Nguyên lý quét cột để hiển thị kí tự, nguyên lý quét hàng để đáp ứng yêu cầu mở rộng, cách tìm mã kí tự.

Tiếp theo là phần viết các chương trình mẫu điều khiển hiển thị từng kí tự trên led ma trận từ đơn giản đến nâng cao.

Kết hợp với các module khác để thực hiện yêu cầu điều khiển phức tạp hơn, tổng hợp nhiều chương trình điều khiển với nhiều chức năng.

Tài liệu và bộ thực hành này sẽ giúp người học thực hành rất nhanh, hiệu quả cao, không mất nhiều thời gian lãng phí như các bộ thực hành trước đây. Sau khi kết thúc các bạn sẽ có kỹ năng tay nghề cao, kỹ năng làm việc thành thạo và nắm vững kiến thức, giúp bạn tự tin cho việc làm trong tương lai.

Trong quá trình biên soạn không thể tránh được các sai sót nên rất mong các bạn đọc đóng góp xây dựng . Mọi ý kiến đóng góp xin gửi về nhóm tác giả đại diện theo địa chỉ phund@hcmute.edu.vn hoặc phu_nd@yahoo.com.

Nhóm tác giả xin cảm ơn các bạn bè đồng nghiệp đã đóng góp nhiều ý kiến, xin cảm ơn người thân trong gia đình cho phép nhóm tác giả có nhiều thời gian biên soạn tài liệu này.

Nguyễn Đình Phú

Phan Văn Hoàn

Trương Ngọc Anh

DANH SÁCH HÌNH

Hình 1- 1. Bộ thực hành vi điều khiển.....	2
Hình 1- 2. Sơ đồ khói kit vi điều khiển PIC18F4550 giao tiếp với 6 module ngoại vi.....	4
Hình 1- 3. Sơ đồ chân và sơ đồ khói của IC chốt 74573.....	4
Hình 1- 4. Sơ đồ khói thanh ghi 74HC595.....	7
Hình 1- 5. Sơ đồ chi tiết bên trong thanh ghi 74HC595.....	8
Hình 1- 6. Sơ đồ khói thanh ghi MBI5026.....	9
Hình 1- 7. Sơ đồ giao tiếp port D của PIC 18F4550 với 2 IC chốt mở rộng.....	10
Hình 1- 8. Sơ đồ nguyên lý module 32 led đơn.....	11
Hình 1- 9. Sơ đồ nguyên lý module 4 led 7 đoạn.....	12
Hình 1- 10. Sơ đồ nguyên lý module 8 led 7 đoạn quét.....	13
Hình 1- 11. Sơ đồ nguyên lý module LCD 20×4.....	15
Hình 1- 12. Sơ đồ nguyên lý module GLCD 128×64.....	16
Hình 1- 13. Sơ đồ nguyên lý module phím nhấn đơn và phím ma trận.....	17
Hình 1- 14. Sơ đồ nguyên lý module thu phát hồng ngoại tạo xung.....	17
Hình 1- 15. Sơ đồ nguyên lý module điều khiển động cơ bước và động cơ DC.....	18
Hình 1- 16. Sơ đồ nguyên lý module điều khiển Relay, Triac, tải 220V AC.....	18
Hình 1- 17. Sơ đồ nguyên lý module giao tiếp I2C và các cảm biến tương tự.....	20
Hình 1- 18. Sơ đồ nguyên lý module cảm biến hồng ngoại PIR.....	20
Hình 1- 19. Sơ đồ nguyên lý module 2 cảm biến nhiệt DS18B20.....	21
Hình 1- 20. Sơ đồ nguyên lý module kết nối Encoder của motor.....	21
Hình 1- 21. Sơ đồ nguyên lý module cảm biến led thu, siêu âm, khí ga, màu.....	21
Hình 1- 22. Sơ đồ nguyên lý mạch điều khiển 3 led ma trận.....	22
Hình 1- 23. Sơ đồ nguyên lý mạch giao tiếp máy tính bằng cổng USB qua IC chuyển đổi.....	23
Hình 1- 24. Sơ đồ nguyên lý mạch giao tiếp SIM900.....	24
 Hình 2- 1. Biểu tượng phần mềm CCS.....	26
Hình 2- 2. Giao diện phần mềm CCS.....	26
Hình 2- 3. Tạo file mới.....	27
Hình 2- 4. Đặt tên file mới và đường dẫn.....	27
Hình 2- 5. Màn hình soạn thảo.....	28
Hình 2- 6. Màn hình thay đổi font.....	28

Hình 2- 7. Màn hình của chương trình chớp tắt 4 led.....	29
Hình 2- 8. Màn hình thông báo kết quả biên dịch thành công.	30
Hình 2- 9. Màn hình phần mềm nạp PICKIT2.....	31
 Hình 3- 1. Sơ đồ nguyên lý giao tiếp vi điều khiển với module 32 led đơn.	35
Hình 3- 2. Lưu đồ điều khiển 8 led chớp tắt.....	39
Hình 3- 3. Lưu đồ điều khiển 8 led sáng dần rồi tắt dần từ phải sang trái.	41
Hình 3- 4. Lưu đồ chương trình con điều khiển 32 led sáng tắt dần từ phải sang trái cho lệnh if.....	49
Hình 3- 5. Sơ đồ nguyên lý giao tiếp vi điều khiển với 4 nút nhấn đơn.....	52
Hình 3- 6. Lưu đồ điều khiển 8 led sáng tắt bằng 2 nút ON và OFF.....	54
Hình 3- 7. Lưu đồ điều khiển 8 led sáng tắt bằng 2 nút ON, OFF và INV.	55
Hình 3- 8. Lưu đồ điều khiển 8 led sáng tắt bằng 2 nút ON, OFF và INV – chống dội.....	57
Hình 3- 9. Sơ đồ nguyên lý giao tiếp vi điều khiển với ma trận 16 phím.	59
Hình 3- 10. Lưu đồ quét ma trận 16 phím.	60
 Hình 4- 1. Sơ đồ nguyên lý giao tiếp vi điều khiển với module 4 led 7 đoạn.....	76
Hình 4- 2. Lưu đồ điều khiển module 4 led 7 đoạn hiển thị 4 số 0, 1, 2, 3.	78
Hình 4- 3. Lưu đồ điều khiển đếm từ 0 đến 9.	79
Hình 4- 4. Sơ đồ khôi timer T0: chế độ 8 bit.	81
Hình 4- 5. Sơ đồ khôi timer T0: chế độ 16 bit.	82
Hình 4- 6. Hình ảnh led thu phát HY860F và sơ đồ nguyên lý.	82
Hình 4- 7. Sơ đồ nguyên lý module thu phát hồng ngoại tạo xung.	83
 Hình 5- 1. Sơ đồ nguyên lý giao tiếp vi điều khiển với module 8 led 7 đoạn.....	93
Hình 5- 2. Lưu đồ điều khiển 8 led sáng từ 0 đến 7 trên module 8 led 7 đoạn.	96
Hình 5- 3. Sơ đồ khôi của timer T1.	100
 Hình 6- 1. Hình ảnh mặt trước của LCD.	120
Hình 6- 2. Hình ảnh mặt sau của LCD.	121
Hình 6- 3. Sơ đồ khôi của bộ điều khiển LCD.....	122
Hình 6- 4. Dạng sóng điều khiển của LCD.	126
Hình 6- 5. Hiển thị 32 ký tự ở LCD 16×2	129
Hình 6- 6. Hiển thị 80 ký tự ở LCD 20×4.....	130

Hình 6- 7. Trình tự khởi tạo LCD.....	131
Hình 6- 8. Giao tiếp vi điều khiển PIC với LCD qua thanh ghi dịch MBI5026.....	132
Hình 6- 9. Lưu đồ điều khiển LCD hiển thị 4 chuỗi kí tự.....	138
Hình 6- 10. Lưu đồ đếm phút giây hiển thị trên LCD.....	142
Hình 6- 11. Lưu đồ đếm sản phẩm hiển thị trên LCD.....	144
Hình 6- 12. Lưu đồ quét phím ma trận hiển thị trên LCD.....	146
Hình 6- 13. Lưu đồ hiển thị mã tự tạo trái tim trên LCD.....	150
Hình 6- 14. Hình ảnh các đoạn cho số lớn trên LCD.....	152
Hình 6- 15. Tìm mã của các đoạn cho số lớn trên LCD.....	153
Hình 6- 16. Hình ảnh các số lớn từ 0 đến 9 trên LCD.....	153
Hình 6- 17. LCD tính theo địa chỉ hàng và cột.....	154
Hình 6- 18. Hình ảnh của GLCD.....	158
Hình 6- 19. Cấu trúc IC ST7920.....	159
Hình 6- 20. Tung độ hiển thị GDRAM và địa chỉ tương ứng của GLCD.....	163
Hình 6- 21. Vẽ lại hình GDRAM theo khối bộ nhớ.....	164
Hình 6- 22. Bộ nhớ GDRAM của GLCD có kích thước 32×256	164
Hình 6- 23. Bộ nhớ GDRAM của GLCD 32×256 bố trí theo kích thước 64×128	164
Hình 6- 24. Bộ nhớ GDRAM của GLCD 32×256 chi tiết.....	165
Hình 6- 25. Sơ đồ nguyên lý module GLCD 128×64	168

Hình 7- 1. Sơ đồ khói của ADC PIC18F4550.....	188
Hình 7- 2. Hình ảnh cảm biến LM35CZ.....	191
Hình 7- 3. Sơ đồ nguyên lý module ADC và các cảm biến tương tự.....	191
Hình 7- 4. Sơ đồ nguyên lý module điều khiển Relay, Triac, tải 220V AC.....	192
Hình 7- 5. Lưu đồ chuyển đổi ADC kênh thứ 0 để đọc nhiệt độ từ cảm biến LM35.....	194
Hình 7- 6. Lưu đồ chuyển đổi ADC kênh thứ 0 để đọc nhiệt độ và so sánh điều khiển buzzer.....	195
Hình 7- 7. Hình cảm biến khoảng cách GP2D12.....	202
Hình 7- 8. Sơ đồ khói bên trong cảm biến khoảng cách GP2D12.....	202
Hình 7- 9. Dạng sóng của cảm biến khoảng cách GP2D12.....	203
Hình 7- 10. Đồ thị chuyển đổi của cảm biến khoảng cách GP2D12.....	204
Hình 7- 11. Hình cảm biến siêu âm và góc phát thu sóng.....	209
Hình 7- 12. Giản đồ thời gian của cảm biến HC-SR04.....	209

Hình 7- 13. Sơ đồ nguyên lý cảm biến khoảng cách siêu âm.	210
Hình 7- 14. Các dạng cảm biến DS18B20.	213
Hình 7- 15. Sơ đồ khối của cảm biến DS18B20.	213
Hình 7- 16. Cấu trúc 2 thanh ghi lưu nhiệt độ của cảm biến DS18B20.	214
Hình 7- 17. Cấu trúc 2 thanh ghi lưu nhiệt độ báo động của cảm biến DS18B20.	215
Hình 7- 18. Sơ đồ cấp nguồn cho cảm biến DS18B20.	216
Hình 7- 19. Cấu trúc mã 64 bit của cảm biến DS18B20.	216
Hình 7- 20. Tổ chức bộ nhớ ROM của cảm biến DS18B20.	216
Hình 7- 21. Byte thanh ghi điều khiển của cảm biến DS18B20.	217
Hình 7- 22. Lưu đồ các lệnh liên quan đến ROM của cảm biến DS18B20.	219
Hình 7- 23. Lưu đồ của các lệnh chức năng của cảm biến DS18B20.	221
Hình 7- 24. Dạng sóng của xung reset và xung hiện diện.	223
Hình 7- 25. Dạng sóng khe thời gian đọc/ghi.	224
Hình 7- 26. Chi tiết dạng sóng khe thời gian đọc mức 1 của thiết bị chủ.	225
Hình 7- 27. Thu ngắn các khoảng thời gian.	225
Hình 7- 28. Mạch đo nhiệt độ dùng 2 cảm biến DS18B20.	226
 Hình 8- 1. Hệ thống các thiết bị giao tiếp theo chuẩn I2C.	232
Hình 8- 2. Quá trình chủ ghi dữ liệu vào tóm tắt.	234
Hình 8- 3. Quá trình chủ đọc dữ liệu từ tóm tắt.	234
Hình 8- 4. Sơ đồ chân DS1307.	234
Hình 8- 5. Sơ đồ kết nối vi điều khiển với DS1307.	235
Hình 8- 6. Tổ chức bộ nhớ của DS1307.	236
Hình 8- 7. Tổ chức các thanh ghi thời gian.	236
Hình 8- 8. Cấu trúc bên trong DS1307.	237
Hình 8- 9. Giao tiếp vi điều khiển các thiết bị theo chuẩn I2C.	238
Hình 8- 10. Lưu đồ đồng hồ số dùng DS1307.	241
Hình 8- 11. Sơ đồ khối IC PCF8591.	253
Hình 8- 12. Sơ đồ khối IC PCF8591.	255
Hình 8- 13. Cấu trúc thanh ghi điều khiển của IC PCF8591.	256
Hình 8- 14. Cấu trúc khối DAC của IC PCF8591.	257
Hình 8- 15. Công thức điện áp ra và đặc tính chuyển đổi.	258
Hình 8- 16. Dạng sóng chuyển đổi D/A.	258

Hình 8- 17. Dạng sóng truyền dữ liệu của IC PCF8591.....	259
Hình 8- 18. Dạng sóng chuyển đổi A/D ở chế độ đơn cực.	259
Hình 8- 19. Dạng sóng chuyển đổi A/D ở chế độ ngõ vào vi sai.....	260
Hình 8- 20. Địa chỉ của IC PCF8591.....	260
Hình 8- 21. Tín hiệu đọc ghi.	261
Hình 8- 22. Chế độ ghi dữ liệu vào IC PCF8591.....	261
Hình 8- 23. Chế độ đọc dữ liệu từ IC PCF8591.....	261
Hình 8- 24. Mối quan hệ giữa sự thay đổi ánh sáng và điện trở.	265
Hình 8- 25. Hình ảnh cảm biến chuyển động.....	267
Hình 8- 26. Phạm vi nhận biết của cảm biến chuyển động.	268
Hình 8- 27. Sơ đồ chân và tên các chân của cảm biến chuyển động.	268
Hình 8- 28. Sơ đồ kết nối tải của cảm biến chuyển động.	268
Hình 8- 29. Sơ đồ của cảm biến chuyển động PIR với vi điều khiển.	269
Hình 8- 30. Sơ đồ chân và tên các chân của IC eeprom AT24C256.	272
Hình 8- 31. Sơ đồ khói của IC Eeprom AT24C256.....	273
Hình 8- 32. Tổ chức địa chỉ của IC Eeprom AT24C256.....	273
Hình 8- 33. Ghi theo byte của IC Eeprom AT24C256.....	274
Hình 8- 34. Ghi theo trang của IC Eeprom AT24C256.....	274
Hình 8- 35. Đọc byte hiện tại của IC Eeprom AT24C256.	274
Hình 8- 36. Đọc byte ngẫu nhiên của IC Eeprom AT24C256.	275
Hình 8- 37. Đọc nhiều byte liên tục của IC Eeprom AT24C256.....	275
 Hình 9- 1. Động cơ bước.	280
Hình 9- 2. Sơ đồ nguyên lý động cơ bước.	280
Hình 9- 3. Hình dạng IC L298.	281
Hình 9- 4. Sơ đồ nguyên lý IC L298.	281
Hình 9- 5. IC L298 điều khiển 1 động cơ bước.	282
Hình 9- 6. Mạch giao tiếp động cơ bước và động cơ DC.	283
Hình 9- 7. Lưu đồ điều khiển động cơ bước thuận bằng 2 nút ON, OFF.....	287
Hình 9- 8Động cơ DC trong BTN.....	291
Hình 9- 9. Encoder gắn trong sau động cơ.	291
Hình 9- 10. IC L298 điều khiển 1 động cơ DC dùng 1 cầu H.	292
Hình 9- 11. IC L298 điều khiển 1 động cơ DC dùng 2 cầu H.	292

Hình 9- 12. Mạch giao tiếp động cơ bước và động cơ DC.....	293
Hình 9- 13. Dạng sóng điều chế độ rộng xung.....	297
Hình 9- 14. Sơ đồ khối của PWM PIC.	298
Hình 9- 15. Dạng sóng PWM	299
Hình 10- 1. Sơ đồ nguyên lý giao tiếp vi điều khiển với 3 led ma trận.....	316
Hình 10- 2. Lưu đồ điều khiển hiển thị 4 ký tự trên 3 led ma trận.	324

DANH SÁCH BẢNG

Bảng 1- 1. Tên các chân của IC 74HC573.	4
Bảng 1- 2. Bảng trạng thái IC 74HC573.	5
Bảng 1- 3. Bảng trạng thái IC 74HC595.	7
Bảng 1- 4. Bảng trạng thái IC MBI5026.	9
Bảng 1- 5. Các tín hiệu mở rộng của IC chốt 74HC573_A.....	10
Bảng 1- 6. Các tín hiệu mở rộng của IC chốt 74HC573_B.....	10
Bảng 1- 7. Bảng mã led 7 đoạn loại anode chung.....	12
Bảng 1- 8. Bảng dữ liệu điều khiển quét lần lượt các transistor.	14
 Bảng 4- 1. Các thanh ghi có liên quan đến timer T0.	82
 Bảng 5- 1. Bảng dữ liệu điều khiển quét lần lượt các transistor.	94
 Bảng 6- 1. Các chân của LCD.....	120
Bảng 6- 2. Các lệnh điều khiển LCD.....	122
Bảng 6- 3. Mã lệnh Function set.	124
Bảng 6- 4. Mã lệnh Display control.	124
Bảng 6- 5. Mã lệnh Clear Display.....	125
Bảng 6- 6. Mã lệnh entry mode.....	125
Bảng 6- 7. Mã lệnh entry mode.....	125
Bảng 6- 8. Cho biết các thông số thời gian của LCD.	127
Bảng 6- 9. Bảng mã ASCII.....	127
Bảng 6- 10. Địa chỉ vùng nhớ hiển thị DDRAM.	128
Bảng 6- 11. Địa chỉ từng kí tự của LCD 20x4.	130
Bảng 6- 12. Thiết lập địa chỉ CGRAM.	148
Bảng 6- 13. Vùng nhớ CGRAM.	148
Bảng 6- 14. Mã của trái tim.....	149
Bảng 6- 15. Các chân của GLCD.....	158
Bảng 6- 16. Các font chữ 8x16 của GLCD.	162
Bảng 6- 17. Địa chỉ gói mã của font 16x16.....	162
Bảng 6- 18. Tập lệnh điều khiển cơ bản của GLCD.....	165

Bảng 7- 1. Giá trị tại các tọa độ chính.	204
Bảng 7- 2. Độ phân giải và thời gian chuyển đổi	217
Bảng 8- 1. Tên và chức năng từng chân của IC PCF8591.	253
Bảng 9- 1. Các trạng thái điều khiển động cơ bước theo bước đú	283
Bảng 9- 2. Các trạng thái điều khiển động cơ bước dạng nữa bước	284
Bảng 9- 3. Các trạng thái điều khiển động cơ DC.	293
Bảng 10- 1. Mã quét cột cho led màu đỏ sáng, xanh tắt	317
Bảng 10- 2. Mã quét cột cho led màu xanh sáng, đỏ tắt	317
Bảng 10- 3. Mã quét cột cho led màu xanh và đỏ đều sáng thành màu cam	318
Bảng 10- 4. Mã các kí tự mẫu A, B, C, D.	318
Bảng 10- 5. Dùng phần mềm tìm mã ta được bảng mã ma trận của các kí tự	319
Bảng 10- 6. Mã quét hàng	327
Bảng 10- 7. Bảng thứ tự các cột màu đỏ	327
Bảng 10- 8. Bảng thứ tự các cột màu xanh	328

MỤC LỤC

GIÁO TRÌNH THỰC HÀNH	i
<i>LỜI NÓI ĐẦU</i>	ii
DANH SÁCH HÌNH	vii
DANH SÁCH BẢNG	xiii
MỤC LỤC	xv
Chương 1: CẤU HÌNH KÍT THỰC HÀNH VI ĐIỀU KHIỂN	1
1.1 GIỚI THIỆU BỘ KÍT THỰC HÀNH	2
1.2 SO ĐỒ NGUYÊN LÝ KÍT VI ĐIỀU KHIỂN	4
1.2.1 Hệ thống vi điều khiển PIC18F4550 giao tiếp với các module ngoại vi	4
1.2.2 Module 1: Giao tiếp với Led đơn	10
1.2.3 Module 2: 4 led 7 đoạn trực tiếp	11
1.2.4 Module 3: 8 led 7 đoạn quét	13
1.2.5 Module 4: LCD 20×4	14
1.2.6 Module 5: GLCD 128×64	15
1.2.7 Module 6: phím đơn và phím ma trận 4×4	16
1.2.8 Module 7: thu phát hồng ngoại tạo xung	17
1.2.9 Module 8: giao tiếp relay, buzzer và động cơ	17
1.2.10 Module 9: giao tiếp ADC, thời gian thực, bộ nhớ Eeprom và các cảm biến	19
1.2.11 Module 10: giao tiếp các cảm biến nhiệt, cảm biến khoảng cách, led thu	20
1.2.12 Module 11: giao tiếp led ma trận	22
1.2.13 Module 12: giao tiếp máy tính bằng USB qua IC chuyển đổi RS232	23
1.2.14 Module 13: giao tiếp SIM900-TE-C-V1.02	23
Chương 2: SỬ DỤNG PHẦN MỀM CCS VÀ PICKIT	25
2.1 GIỚI THIỆU	26
2.2 HƯỚNG DẪN SỬ DỤNG PHẦN MỀM CCS	26
2.2.1 Biên soạn chương trình cơ bản	26
2.2.2 Nạp chương trình vào bộ nhớ vi điều khiển	30
Chương 3: THỰC HÀNH MODULE 1–32 LED ĐƠN DÙNG THANH GHI DỊCH 74HC595, NÚT NHẤN, BÀN PHÍM MA TRẬN	33
3.1 CÁC CHƯƠNG TRÌNH ĐIỀU KHIỂN 32 LED ĐƠN	34
3.1.1 Mạch giao tiếp vi điều khiển với module 32 led đơn	34
3.1.2 Các hàm điều khiển module 32 led đơn	35

3.1.3	Các chương trình điều khiển 32 led đơn dùng lệnh for	38
3.1.4	Các chương trình điều khiển 32 led đơn dùng lệnh if.....	48
3.2	CÁC CHƯƠNG TRÌNH ĐIỀU KHIỂN 32 LED ĐƠN VÀ NÚT NHẤN	51
3.2.1	Giới thiệu	51
3.2.2	Mạch điện giao tiếp vi điều khiển với 4 nút nhấn đơn.....	52
3.2.3	Định nghĩa tên các nút nhấn trong thư viện	52
3.2.4	Các chương trình điều khiển dùng các nút nhấn đơn	53
3.2.5	Mạch điện giao tiếp vi điều khiển với ma trận phím	58
3.2.6	Các chương trình điều khiển dùng ma trận phím	60
3.3	CHƯƠNG TRÌNH THU VIỆN CHO MODULE 32 LED ĐƠN.....	64
3.4	CÁC CÂU HỎI ÔN TẬP	73
<i>Chương 4: THỰC HÀNH MODULE 2 – 4 LED 7 ĐOẠN ANODE DÙNG THANH GHI DỊCH 74HC595, COUNTER/TIMER</i>		75
4.1	CÁC CHƯƠNG TRÌNH ĐIỀU KHIỂN 4 LED 7 ĐOẠN	76
4.1.1	Mạch điện giao tiếp vi điều khiển với module 4 led 7 đoạn.....	76
4.1.2	Các hàm điều khiển module 4 led 7 đoạn.....	77
4.1.3	Các chương trình điều khiển 4 led 7 đoạn	78
4.2.1	Tóm tắt timer/counter T0	81
4.2.2	Cảm biến thu phát hồng ngoại HY860F.....	82
4.2.3	Mạch điện dùng T0 đếm xung ngoại	83
4.2.4	Các ứng dụng đếm xung ngoại dùng counter T0	83
4.3	KẾT HỢP ĐIỀU KHIỂN 32 LED VÀ ĐÉM XUNG DÙNG COUNTER T0	87
4.4	CÁC CÂU HỎI ÔN TẬP	89
<i>Chương 5: THỰC HÀNH MODULE 3 – 8 LED 7 ĐOẠN ANODE CHUNG KẾT NỐI THEO PP QUÉTDÙNG THANH GHI DỊCH MBI5026</i>		91
5.1	MẠCH ĐIỆN GIAO TIẾP VI ĐIỀU KHIỂN VỚI 8 LED 7 ĐOẠN QUÉT	92
5.1.1	Mạch điện giao tiếp vi điều khiển với module 8 led 7 đoạn.....	92
5.1.2	Các hàm điều khiển module 8 led 7 đoạn quét	94
5.1.3	Các chương trình hiển thị 8 led 7 đoạn quét.....	95
5.2	CÁC CHƯƠNG TRÌNH ĐÉM THỜI GIAN HIỂN THỊ 8 LED	99
5.2.1	Đếm giây dùng delay.....	99
5.2.2	Đếm giây dùng timer T1 định thời	100
5.2.3	Sử dụng thêm các hàm điều khiển 8 led quét.....	102
5.3	KẾT HỢP 3 MODULE	113
5.4	CÁC CÂU HỎI ÔN TẬP	116

<i>Chương 6: THỰC HÀNH MODULE 4 – LCD 20x4, GLCD 128x64</i>	119
6.1 LÝ THUYẾT LCD	120
6.1.1 Giới thiệu LCD	120
6.1.2 Sơ đồ chân của LCD	120
6.1.3 Bộ điều khiển LCD và các vùng nhớ	121
6.1.4 Các lệnh điều khiển LCD	122
6.1.5 Các hoạt động đọc ghi LCD.....	126
6.1.6 Mã ASCII.....	127
6.1.7 Vùng nhớ hiển thị DDRAM.....	128
6.1.8 Lưu đồ khởi tạo LCD	130
6.2 LCD TRONG KIT THỰC HÀNH.....	131
6.2.1 Mạch giao tiếp vi điều khiển với LCD.....	131
6.2.2 Dữ liệu và hàm điều khiển LCD và GLCD	133
6.3 CÁC CHƯƠNG TRÌNH SỬ DỤNG VÙNG NHỚ DDRAM CỦA LCD	134
6.3.1 Các ứng dụng hiển thị ký tự, dịch chuỗi trên LCD	134
6.3.2 Đếm thời gian và đếm xung ngoại hiển thị trên LCD	141
6.3.3 Các ứng dụng kết hợp bàn phím ma trận và LCD	145
6.4 CÁC ỨNG DỤNG VÙNG NHỚ CGRAM CỦA LCD	148
6.4.1 Vùng nhớ CGRAM.....	148
6.4.2 Cách tìm mã ký tự mong muốn.....	148
6.4.3 Các chương trình ứng dụng các ký tự - tự tạo trên LCD	149
6.4.4 Tạo mã 7 đoạn kích thước lớn trên LCD	151
6.5 LÝ THUYẾT GLCD	157
6.5.1 Giới thiệu GLCD	157
6.5.2 Sơ đồ chân của GLCD	158
6.5.3 IC điều khiển GLCD ST7920	159
6.5.4 Mô tả chức năng ic điều khiển GLCD ST7920	160
6.5.5 Các lệnh điều khiển GLCD	165
6.6 CÁC BÀI THỰC HÀNH DÙNG GLCD.....	168
6.6.1 Sơ đồ mạch của GLCD	168
6.6.2 Các bài thực hành hiển thị ký tự trên GLCD	168
6.6.3 Các bài thực hành hiển thị hình ảnh trên GLCD	179
6.7 KẾT HỢP NHIỀU MODULE.....	181
<i>Chương 7: THỰC HÀNH CHUYỂN ĐỔI TƯƠNG TỰ SANG SỐ - ADC, CÁC CẢM BIẾN</i>	187
7.1 CHUYỂN ĐỔI TƯƠNG TỰ SANG SỐ CỦA PIC.....	188

7.1.1	Khảo sát ADC của PIC 18F4550	188
7.1.2	Tập lệnh c cho khối ADC của PIC	189
7.2	ĐO NHIỆT ĐỘ DÙNG CẢM BIẾN LM35	190
7.2.1	Cảm biến LM35	190
7.2.2	Mạch điện giao tiếp vi điều khiển PIC với cảm biến LM35.....	191
7.2.3	Đo nhiệt độ dùng cảm biến LM35 hiển thị trên 4 led 7 đoạn	192
7.3	ĐO KHOÁNG CÁCH DÙNG CẢM BIẾN GP2D12.....	201
7.3.1	Cảm biến GP2D12	201
7.3.2	Đo khoảng cách hiển thị trên 4 led 7 đoạn	205
7.4	ĐO KHOÁNG CÁCH DÙNG CẢM BIẾN SIÊU ÂM HC-SR 04.....	208
7.4.1	Cảm biến HC-SR 04	208
7.4.2	Cảm biến HC-SR 04 trong kit thực hành.....	209
7.4.3	Đo khoảng cách dùng HCSR-04 hiển thị trên 4 led 7 đoạn	210
7.5	CẢM BIẾN NHIỆT 1 DÂY.....	212
7.5.1	Cảm biến nhiệt DS18B20.....	212
7.5.2	Hình ảnh và tên các chân của cảm biến	212
7.5.3	Sơ đồ khối của cảm biến DS18B20	213
7.5.4	Hoạt động của cảm biến DS18B20	214
7.5.5	Hoạt động cảnh báo quá nhiệt của cảm biến DS18B20	215
7.5.6	Cáp nguồn cho cảm biến DS18B20	215
7.5.7	Mã 64 bit của cảm biến DS18B20.....	216
7.5.8	Bộ nhớ rom của cảm biến DS18B20.....	216
7.5.9	Thanh ghi định cấu hình của cảm biến DS18B20	217
7.5.10	Trình tự hoạt động của cảm biến DS18B20	217
7.5.11	Các lệnh hoạt động của cảm biến DS18B20.....	218
7.5.12	Các dạng tín hiệu của chuẩn 1 dây.....	222
7.5.13	Trình tự khởi động – xung reset và xung hiện diện.....	222
7.6	CÁC CHƯƠNG TRÌNH ĐO NHIỆT ĐỘ DÙNG CẢM BIẾN DS18B20	225
7.6.1	Mạch điện giao tiếp vi điều khiển với cảm biến DS18B20.....	225
7.6.2	Đo nhiệt độ dùng cảm biến DS18B20 hiển thị trên led 7 đoạn	226
Chương 8:	THỰC HÀNH MODULE 5 – REAL TIME DS13B07, ADC–DAC PCF8591, EEPROM NỐI TIẾP AT24C256 THEO CHUẨN I2C VÀ CÁC CẢM BIẾN.....	231
8.1	LÝ THUYẾT I2C.....	232
8.1.1	Giới thiệu	232
8.1.2	Tổng quan về truyền dữ liệu chuẩn I2C	232

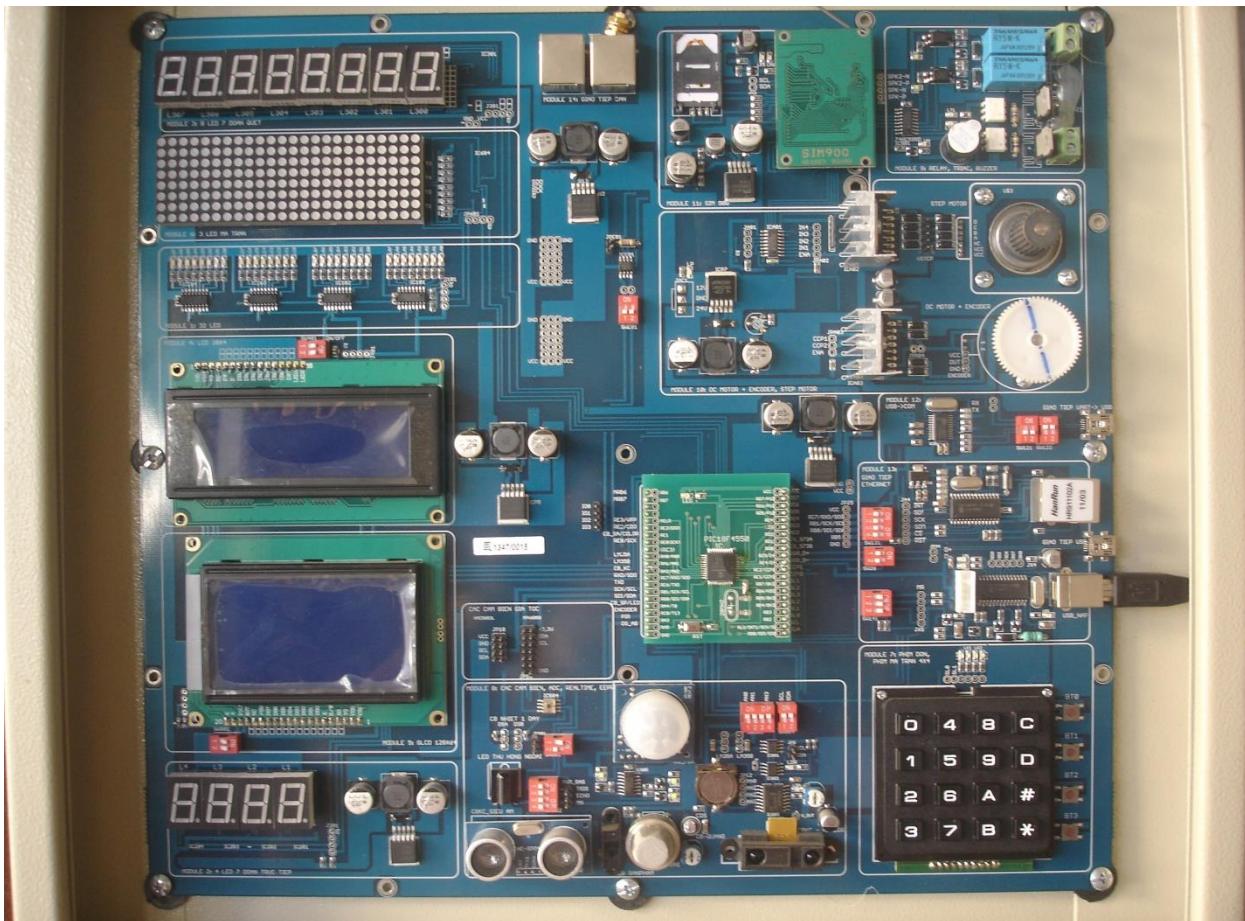
8.2 KHẢO SÁT IC REALTIME DS13B07	234
8.2.1 Khảo sát IC DS1307.....	234
8.2.2 Sơ đồ giao tiếp vi điều khiển với DS1307	237
8.2.3 Các chương trình giao tiếp DS13B07	238
8.3 KHẢO SÁT IC PCF8591	252
8.3.1 Giới thiệu IC PCF8591	252
8.3.2 Cấu hình IC PCF8591	252
8.3.3 Ứng dụng IC PCF8591	253
8.3.4 Sơ đồ chân IC PCF8591.....	253
8.3.5 Sơ đồ khởi IC PCF8591	254
8.3.6 Địa chỉ ngoài IC PCF8591	255
8.3.7 Thanh ghi điều khiển của IC PCF8591.....	255
8.3.8 Chuyển đổi D/A của IC PCF8591.....	257
8.3.9 Chuyển đổi A/D của IC PCF8591.....	258
8.3.10 Điện áp tham chiếu của IC PCF8591	260
8.3.11 Bộ dao động của IC PCF8591.....	260
8.3.12 Địa chỉ thanh ghi của IC PCF8591.....	260
8.3.13 Giao tiếp vi điều khiển với IC PCF8591.....	261
8.3.14 Các chương trình giao tiếp ic PCF 8591	261
8.4 CẢM BIẾN QUANG TRỞ.....	264
8.4.1 Khảo sát quang trở	264
8.4.2 Vi điều khiển giao tiếp với quang trở	265
8.4.3 Các chương trình điều khiển dùng quang trở.....	265
8.5 CẢM BIẾN CHUYỂN ĐỘNG PIR.....	267
8.5.1 Khảo sát cảm biến chuyển động PIR	267
8.5.2 Vi điều khiển giao tiếp với cảm biến PIR.....	268
8.5.3 Các chương trình điều khiển dùng cảm biến PIR.....	269
8.6 DAC IC PCF8591 ĐIỀU KHIỂN ĐÈN.....	270
8.7 IC EEPROM AT24C256	271
8.7.1 Khảo sát IC nhớ EEPROM AT24C256	271
8.7.2 Mạch điện giao tiếp với IC nhớ EEPROM AT24C256.....	275
8.7.3 Các chương trình ghi đọc IC nhớ EEPROM AT24C256	275
Chương 9: THỰC HÀNH MODULE 6 – ĐỘNG CƠ BUỚC, ĐỘNG CƠ DC VÀ ENCODER, PWM, ĐIỀU KHIỂN PID	279
9.1 ĐIỀU KHIỂN ĐỘNG CƠ BUỚC.....	280

9.1.1	Giới thiệu động cơ bước	280
9.1.2	IC giao tiếp công suất điều khiển L298	280
9.1.3	Mạch giao tiếp điều khiển động cơ bước.....	282
9.1.4	Mã điều khiển động cơ bước	283
9.1.5	Các biến và hàm điều khiển Relay, Triac, Buzzer, động cơ	284
9.1.6	Các bài thực hành điều khiển động cơ bước.....	285
9.2	ĐIỀU KHIỂN ĐỘNG CƠ DC, ENCODER.....	290
9.2.1	Giới thiệu động cơ DC, ENCODER.....	290
9.2.2	Mạch giao tiếp điều khiển động cơ DC	292
9.2.3	Dữ liệu điều khiển động cơ DC.....	293
9.2.4	Các bài thực hành điều khiển động cơ DC.....	293
9.3	ĐIỀU KHIỂN THAY ĐỔI TỐC ĐỘ ĐỘNG CƠ DC PWM, ENCODER.....	296
9.3.1	Giới thiệu PWM (PULSE WIDTH MODULATION).....	296
9.3.2	Cấu trúc khối điều chế độ rộng xung PWM của PIC.....	297
9.3.3	Tính chu kỳ xung PWM	299
9.3.4	Tính hệ số chu kỳ xung PWM	299
9.3.5	Các lệnh điều khiển PWM.....	299
9.3.6	Tính toán điều khiển tốc độ động cơ bằng PWM	300
9.3.7	Các bài thực hành thay đổi tốc độ động cơ dc bằng PWM.....	301
9.4	ĐIỀU KHIỂN KẾT HỢP 2 ĐỘNG CƠ BUỚC VÀ DC.....	311
9.5	ĐIỀU KHIỂN ỒN ĐỊNH TỐC ĐỘ ĐỘNG CƠ DC	311
9.3.1	Giới thiệu PID	311
9.3.2	Điều khiển động cơ dùng PID.....	311
Chương 10:	CÁC BÀI THỰC HÀNH MODULE 7 – ĐIỀU KHIỂN LED MA TRẬN 2 MÀU	315
10.1	SƠ ĐỒ MẠCH GIAO TIẾP VI ĐIỀU KHIỂN VỚI LED MA TRẬN	316
10.2	MÃ KÍ TỰ CHO LED MA TRẬN DẠNG QUÉT CỘT	316
10.3	CÁC CHƯƠNG TRÌNH ĐIỀU KHIỂN LED MA TRẬN QUÉT CỘT	320
10.4	MÃ KÍ TỰ CHO LED MA TRẬN DẠNG QUÉT HÀNG	326
10.5	CÁC CHƯƠNG TRÌNH ĐIỀU KHIỂN LED MA TRẬN QUÉT HÀNG	336
TÀI LIỆU THAM KHẢO		340

Chương 1: CẤU HÌNH KIT THỰC HÀNH VI ĐIỀU KHIỂN

1.1 GIỚI THIỆU BỘ KIT THỰC HÀNH

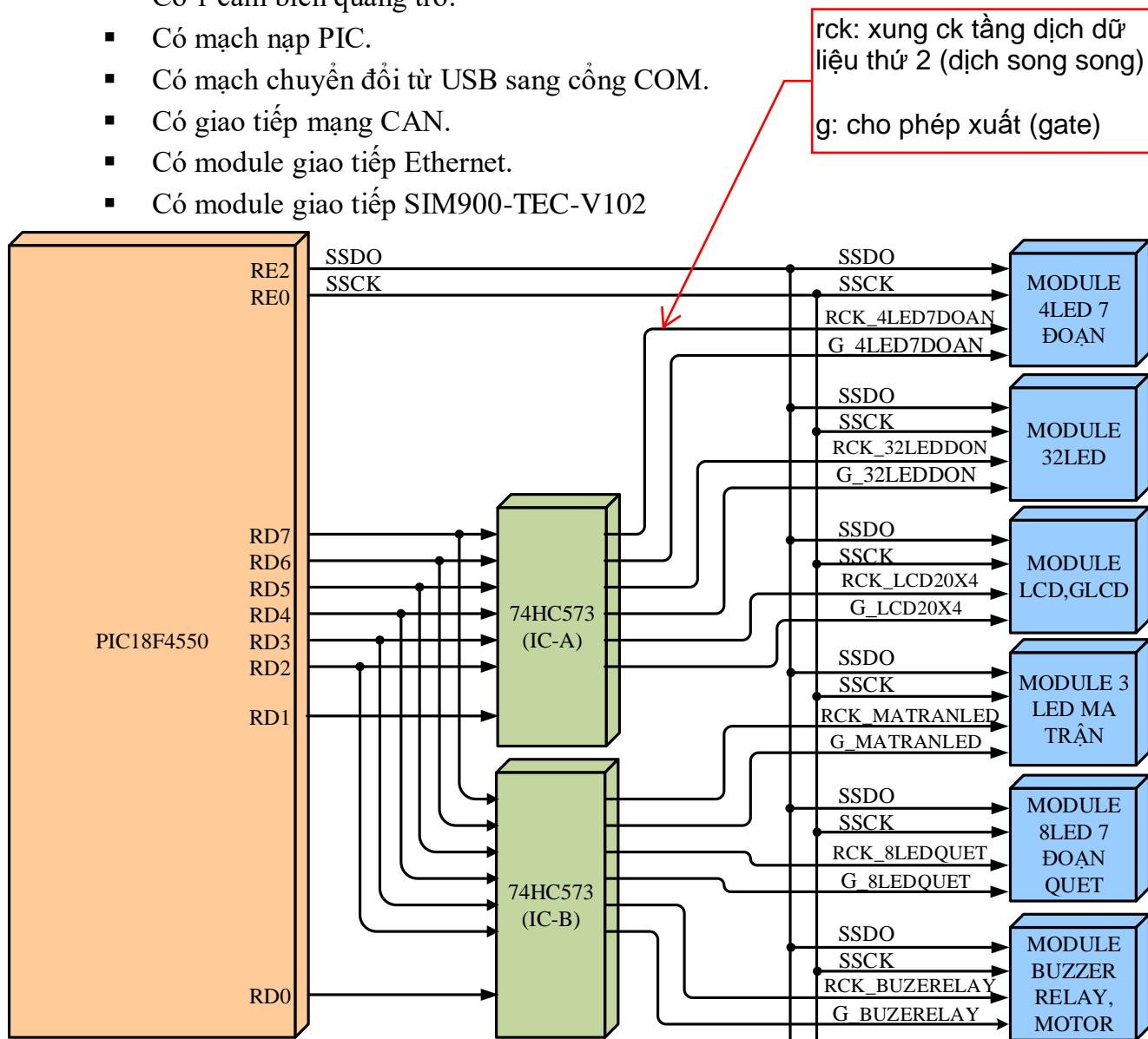
Chương này giới thiệu bộ kit thực hành vi điều khiển tương đối đầy đủ các yêu cầu phần cứng như hình 1-1, có thể giúp bạn thực hành và tự nghiên cứu, tự học.



Hình 1-1. Bộ thực hành vi điều khiển.

- ✓ Bộ thực hành vi điều khiển có thể sử dụng cho nhiều loại vi điều khiển bằng cách thay đổi socket.
- ✓ Các module thực hành gồm có:
 - Có 32 led đơn giao tiếp dùng IC mở rộng 74HC595.
 - Có 4 led 7 đoạn kết nối trực tiếp giao tiếp dùng IC mở rộng 74HC595.
 - Có 8 led 7 đoạn kết nối theo phương pháp quét thông qua IC MBI5026.
 - Có 3 led ma trận 8x8 hai màu xanh và đỏ dùng IC MBI5026.
 - Có 1 LCD 20x4 kết nối thông qua IC MBI5026.
 - Có 1 GLCD 128x64 kết nối thông qua IC MBI5026.
 - Có 4 phím đơn kết nối trực tiếp.
 - Bàn phím ma trận gồm 16 phím.
 - Có 2 relay và 2 triac.
 - Có 1 buzzer.
 - Có mạch công suất điều khiển động cơ bước và 1 động cơ bước.

- Có mạch công suất điều khiển động cơ DC và 1 động cơ DC, có encoder.
- Chuyển đổi tín hiệu tương tự sang số (ADC) đa hợp 4 kênh.
- Chuyển đổi tín hiệu số sang tín hiệu tương tự (DAC).
- Có 2 cảm biến LM35.
- Có 2 cảm biến nhiệt theo chuẩn 1 dây DS18B20.
- Có 1 cảm biến PIR nhận biết chuyển động.
- Có 1 cảm biến khí gas/khói.
- Có 2 cảm biến khoảng cách.
- Có giao tiếp đồng hồ thời gian thực nối tiếp DS13B07.
- Có 1 bộ nhớ EEPROM giao tiếp theo chuẩn I2C.
- Có 1 mạch thu phát xung.
- Có cảm biến gia tốc.
- Có 1 cảm biến quang trở.
- Có mạch nạp PIC.
- Có mạch chuyển đổi từ USB sang cổng COM.
- Có giao tiếp mạng CAN.
- Có module giao tiếp Ethernet.
- Có module giao tiếp SIM900-TEC-V102



Hình 1-2. Sơ đồ khói kit vi điều khiển PIC18F4550 giao tiếp với 6 module ngoại vi.

1.2 SƠ ĐỒ NGUYÊN LÝ KIT VI ĐIỀU KHIỂN

Bộ thực hành vi điều khiển có thể sử dụng nhiều loại vi điều khiển khác nhau, tuy nhiên, tài liệu chỉ trình bày cho loại vi điều khiển PIC18F4550.

1.2.1 Hệ thống vi điều khiển PIC18F4550 giao tiếp với các module ngoại vi

Sơ đồ khói của kit dùng vi điều khiển PIC18F4550 giao tiếp với 6 module ngoại vi như hình 1-2.

Kit vi điều khiển có nhiều module ngoại vi. Để có thể giao tiếp hết các module ngoại vi thì vi điều khiển dùng thêm 2 IC mở rộng port là 74HC573, được phân biệt là IC-A và IC-B.

Sơ đồ khói giúp bạn đọc dễ tiếp cận, sau đó bạn đọc có thể xem sơ đồ nguyên lý chi tiết ở hình tiếp theo.

Trong sơ đồ khói hình 1-2 thì port D có 6 bit nối song song với 2 IC chốt 74HC573 để tạo 12 ngõ ra độc lập để điều khiển 6 module giao tiếp.

Hai bit RD1 và RD0 dùng để điều khiển ngõ vào chốt dữ liệu của 2 IC chốt.

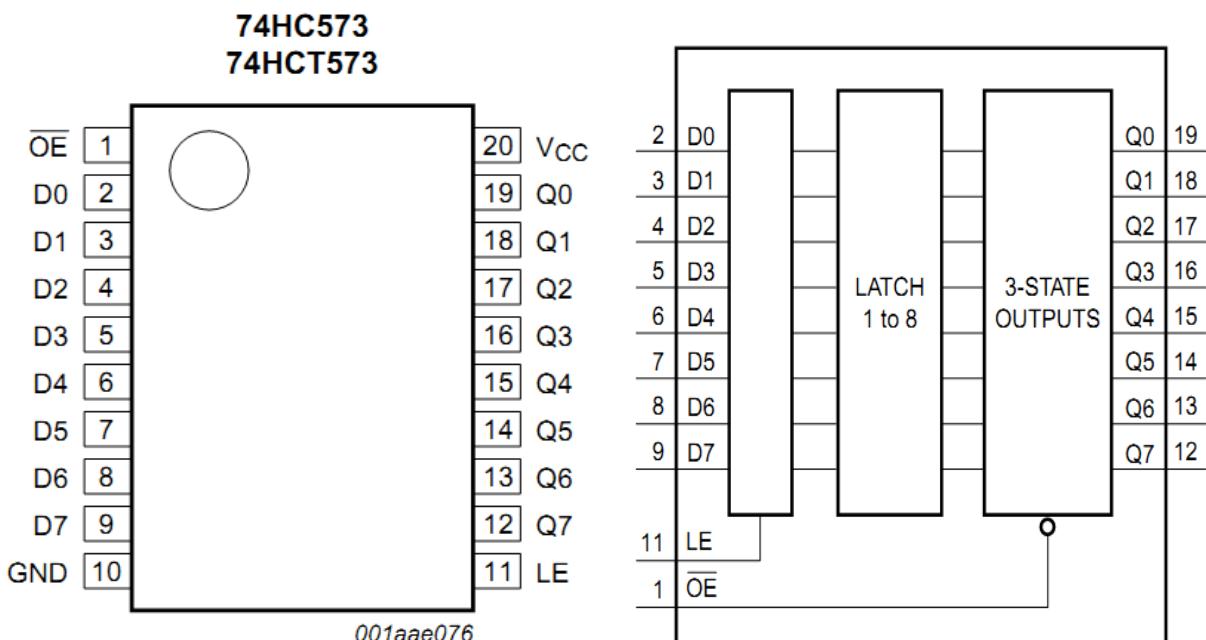
Để hiểu rõ cách thức chốt dữ liệu thì ta phải tìm hiểu nguyên lý hoạt động của IC chốt.

a. **IC chốt 74HC573:**

Sơ đồ kí hiệu và bảng trạng thái IC chốt như hình 1-3.

IC chốt 74HC573 có 8 ngõ vào nhận dữ liệu D₇-D₀ và 8 ngõ ra Q₇-Q₀.

Tín hiệu điều khiển xuất dữ liệu (OE: output enable) tích cực mức thấp cho phép đưa dữ liệu từ mạch chốt ở bên trong ra ngoài, nếu không cho phép thì các ngõ ra sẽ ở trạng thái tổng trở cao.

**Hình 1-3. Sơ đồ chân và sơ đồ khói của IC chốt 74573.****Bảng 1-1. Tên các chân của IC 74HC573.**

Symbol	Pin	Description
\overline{OE}	1	3-state output enable input (active LOW)
D[0:7]	2, 3, 4, 5, 6, 7, 8, 9	data input
GND	10	ground (0 V)
\overline{LE}	11	latch enable input (active HIGH)
Q[0:7]	19, 18, 17, 16, 15, 14, 13, 12	3-state latch output
V _{CC}	20	supply voltage

Tín hiệu điều khiển chốt (LE: latch enable):

Khi ở mức 1 thì cho phép thông dữ liệu: dữ liệu đưa đến 8 ngõ vào D₇-D₀ sẽ được lưu vào mạch chốt bên trong và nếu được phép thì đưa ra bên ngoài Q₇-Q₀.

Khi ở mức 0 thì không cho phép thông dữ liệu ngõ vào D₇-D₀ lưu vào mạch chốt, dữ liệu lưu trong mạch chốt chính là dữ liệu trước khi hạ tín hiệu chốt về mức 0.

Bảng 1-2. Bảng trạng thái IC 74HC573.

Table 3. Function table^[1]

Operating mode	Control		Input	Internal latches	Output
	OE	LE			
Enable and read register (transparent mode)	L	H	L	L	L
			H	H	H
Latch and read register	L	L	I	L	L
			h	H	H
Latch register and disable outputs	H	L	I	L	Z
			h	H	Z

Có thể nói ngắn gọn là khi tín hiệu LE ở mức 1 thì cho phép thông dữ liệu từ ngõ vào đến ngõ ra, khi LE ở mức 0 thì tín hiệu bị chặn lại và các ngõ ra lưu lại dữ liệu trước khi hạ chốt.

b. Hoạt động của 2 IC chốt 74HC573 trong hệ thống:

Trong sơ đồ khái hình 1-2 thì hai tín hiệu OE đều được nối GND để luôn cho phép xuất dữ liệu ra, hai bit RD0 và RD1 dùng để điều khiển 2 tín hiệu LE của 2 IC chốt.

Bình thường thì 2 tín hiệu chốt ở mức logic 0, khi đó không IC nào được phép nhận dữ liệu.

Khi gởi dữ liệu ra 6 bit từ RD2 đến RD7 thì cả 2 IC chốt đều có dữ liệu ở đầu vào.

Muốn IC chốt nào nhận dữ liệu 6 bit này thì ta tiến hành mở chốt cho IC đó, IC còn lại không bị ảnh hưởng.

Tương tự bạn tiến hành xuất dữ liệu ra chốt dữ liệu lại cho IC còn lại.

Kết quả ta có thể có 12 đường tín hiệu khác nhau ở ngõ ra của 2 IC chốt.

c. Chức năng của 12 đường trong hệ thống:

12 tín hiệu ngõ ra của 2 IC chốt chia thành 6 cặp, mỗi cặp 2 tín hiệu có tên chung là RCK và G, còn phần tiếp theo của tên chỉ cho biết đối tượng điều khiển.

Vì điều khiển PIC dùng 2 tín hiệu được đặt tên là SSDO và SSCK giao tiếp song song với tất cả 6 module và 2 tín hiệu này có chức năng dịch chuyển dữ liệu nối tiếp từ vi điều khiển đến các module.

Trong sơ đồ khối hình 1-2 thì mỗi module giao tiếp sử dụng 4 tín hiệu: SSDO, SSCK, RCK_x và G_x, với x là tên gắn với chức năng của từng module tương ứng.

Các module ngoại vi sử dụng các thanh ghi dịch 74HC595 và MBI5026 có chức năng chuyển dữ liệu từ nối tiếp sang song – song để điều khiển các thiết bị ở ngõ ra.

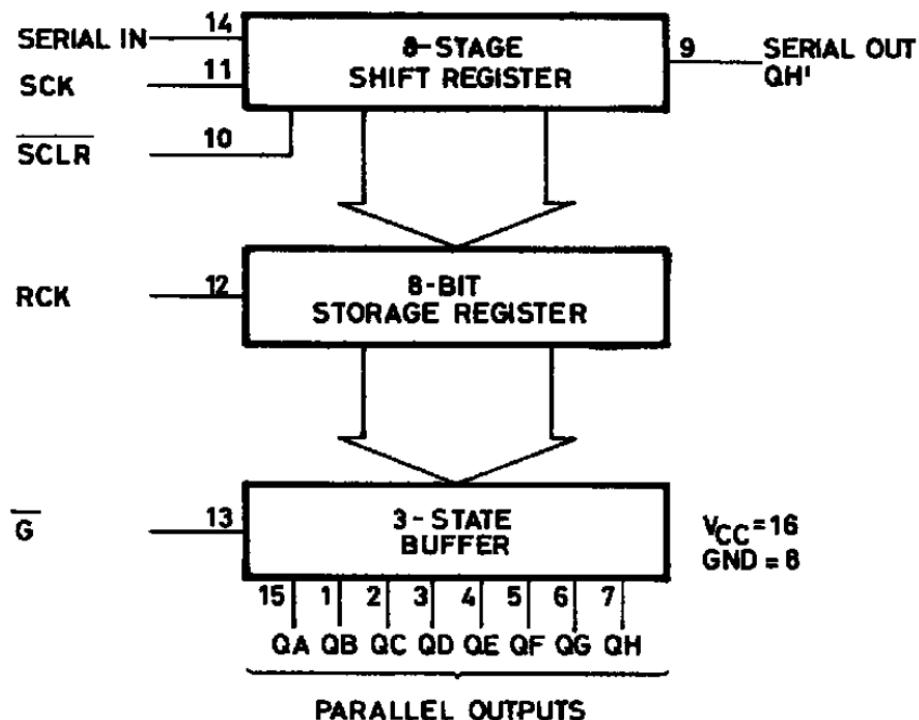
Kit vi điều khiển được thiết kế theo kiểu giao tiếp này với mục đích mở rộng nhiều tín hiệu giao tiếp để đủ tín hiệu điều khiển tất cả các ngoại vi, không sử dụng dây bus nối giữa các port với các module như các kit vi điều khiển trước đây, nhờ đó khắc phục được các nhược điểm đứt dây, kết nối sai port, sai thứ tự tín hiệu, ...

Ta cần phải tìm hiểu nguyên lý hoạt động của các IC thanh ghi dịch 74HC595 và MBI5026.

d. **Hoạt động của IC 74HC595:**

Sơ đồ kí hiệu và bảng trạng thái IC 74HC595 như hình 1-4. IC 74HC595 có 2 tầng thanh ghi 8 bit: tầng thanh ghi dịch (8-bit Stage Shift Register) và tầng thanh ghi lưu trữ (8-bit Store Register).

- Chân tín hiệu SERIAL_IN (SI) là ngõ vào nhận dữ liệu nối tiếp.
- Chân tín hiệu SCK dùng để nhận xung clock để đẩy dữ liệu vào thanh ghi dịch.
- Chân tín hiệu SCLR dùng để xóa dữ liệu trong thanh ghi dịch nối tiếp. Tín hiệu này tích cực mức 0 và không dùng thì nối với mức 1.
- Chân tín hiệu QH' dùng để kết nối với thanh ghi dịch tiếp theo.
- Tín hiệu RCK dùng để nạp dữ liệu từ thanh ghi dịch bên trong sang thanh ghi lưu trữ và nếu cho phép thì xuất dữ liệu ra ngoài.
- Chân tín hiệu G dùng để mở bộ đệm 3 trạng thái xuất tín hiệu ra ngoài.

*Hình 1-4. Sơ đồ khối thanh ghi 74HC595.**Bảng 1-3. Bảng trạng thái IC 74HC595.*

INPUTS					OUTPUT
SI	SCK	SCLR	RCK	G	
X	X	X	X	H	QA THRU QH OUTPUTS DISABLE
X	X	X	X	L	QA THRU QH OUTPUTS ENABLE
X	X	L	X	X	SHIFT REGISTER IS CLEARED
L	---	H	X	X	FIRST STAGE OF S.R. BECOMES "L" OTHER STAGES STORE THE DATA OF PREVIOUS STAGE, RESPECTIVELY
H	---	H	X	X	FIRST STAGE OF S.R. BECOMES "H" OTHER STAGES STORE THE DATA OF PREVIOUS STAGE, RESPECTIVELY
X	---	H	X	X	STATE OF S.R IS NOT CHANGED
X	X	X	---	X	S.R. DATA IS STORED INTO STORAGE REGISTER
X	X	X	---	X	STORAGE REGISTER STATE IS NOT CHANGED

X: DON'T CARE

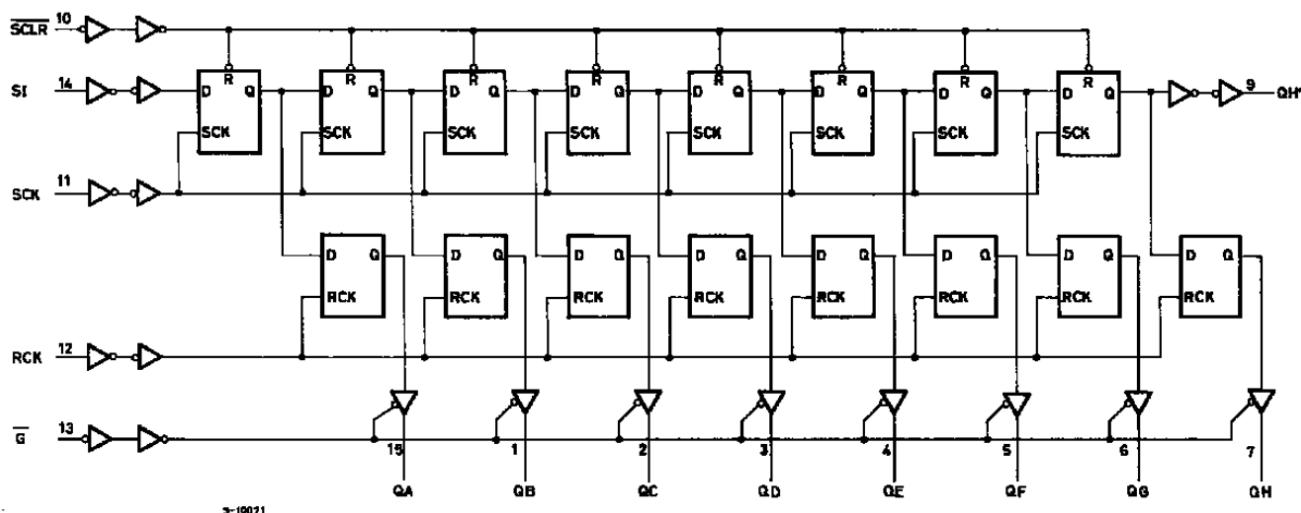
Chức năng 2 tầng thanh ghi là khi dịch chuyển dữ liệu sẽ xảy ra ở tầng thứ nhất không làm ảnh hưởng tầng thứ 2 và không ảnh hưởng đến ngõ ra, chờ cho dịch chuyển xong dữ liệu mới cho phép xuất dữ liệu ra ngoài.

Sơ đồ bên trong của IC thanh ghi dịch 74HC595 như hình 1-5.

Trong sơ đồ trên ta thấy có 2 tầng thanh ghi dùng flip flop D, tầng thứ nhất được điều khiển bởi 3 tín hiệu SCLR, SI và SCK.

Dữ liệu ngõ ra của 8 flip flop D được nối đến 8 ngõ vào của 8 flip flop D ở tầng thứ 2. 8 bit này thay đổi ở bên trong không làm ảnh hưởng đến 8 ngõ ra Q khi chưa có xung RCK.

Khi có xung RCK thì 8 bit dữ liệu được đẩy đến 8 ngõ ra Q tương ứng của 8 flip flop ở tầng thứ 2 và nếu được phép bởi tín hiệu G thì dữ liệu sẽ xuất ra ngoài.



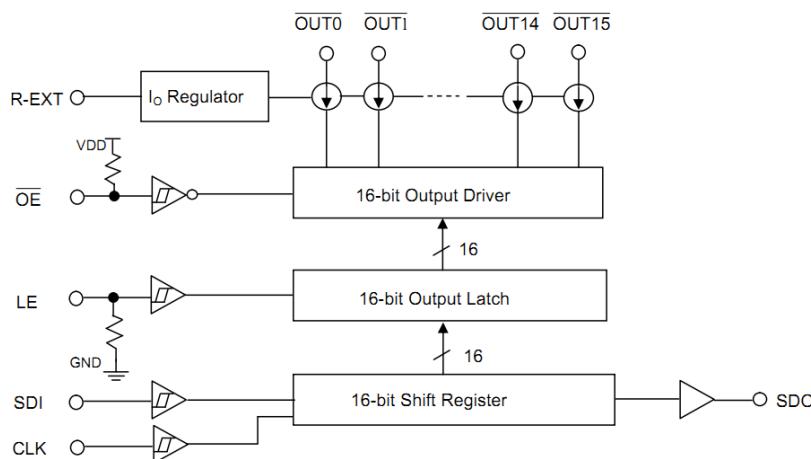
Hình 1-5. Sơ đồ chi tiết bên trong thanh ghi 74HC595.

e. *Hoạt động của IC MBI5026:*

Sơ đồ khối của vi mạch MBI5026 và bảng trạng thái hoạt động của IC như hình 1-6:

MBI5026 là thanh ghi dịch 16 bit có thể nhận dòng từ 5mA đến 90mA có thể thiết lập được bằng điện trở gắn thêm bên ngoài. Áp có thể hoạt động lên đến 36V phù hợp với các tải công suất như relay, solenoid, ...

- Chân tín hiệu SDI là ngõ vào nhận dữ liệu nối tiếp.
- Chân tín hiệu CLK dùng để nhận xung clock để đẩy dữ liệu vào thanh ghi dịch.
- Chân tín hiệu LE dùng để giống như RCK để nạp dữ liệu từ tầng 1 sang tầng 2.
- Chân tín hiệu SDO dùng để kết nối với thanh ghi dịch tiếp theo.
- Chân tín hiệu OE dùng để mở bộ đệm 3 trạng thái xuất tín hiệu ra ngoài.
- Chân tín hiệu R-EXT dùng để giới hạn dòng tải chạy vào, khi dùng thì nối với một đầu của điện trở và đầu còn lại của điện trở nối mass.
- 16 tín hiệu ngõ ra tích cực mức thấp, chỉ nhận dòng vào chứ không cấp dòng ra.



Hình 1-6. Sơ đồ khối thanh ghi MBI5026.**Bảng 1-4. Bảng trạng thái IC MBI5026.**

CLK	LE	OE	SDI	OUT0 ... OUT7 ... OUT15	SDO
	H	L	D _n	D _n D _{n-7} D _{n-15}	D _{n-15}
	L	L	D _{n+1}	No Change	D _{n-14}
	H	L	D _{n+2}	D _{n+2} D _{n-5} D _{n-13}	D _{n-13}
	X	L	D _{n+3}	D _{n+2} D _{n-5} D _{n-13}	D _{n-13}
	X	H	D _{n+3}	Off	D _{n-13}

f. Hoạt động của vi điều khiển với các thanh ghi và IC chốt:

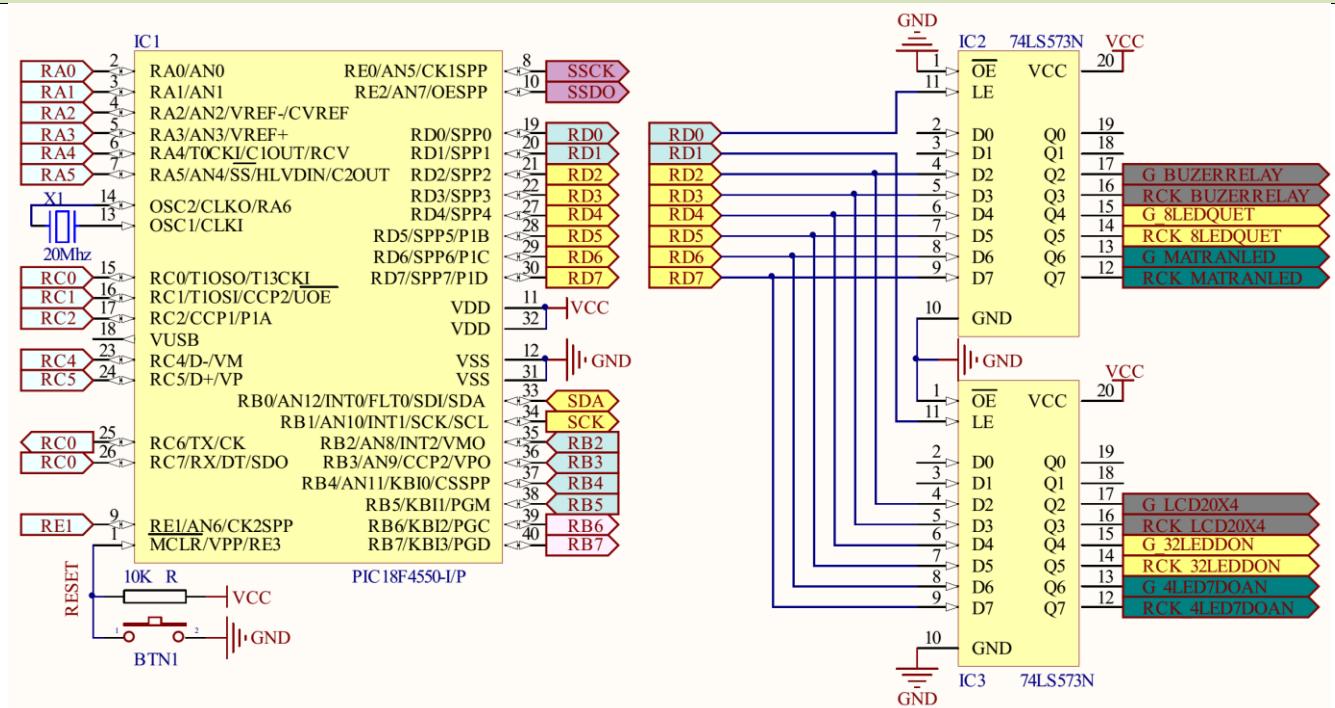
Qua khảo sát 2 loại IC thanh ghi dịch thì khi giao tiếp với vi điều khiển chỉ cần dùng 4 tín hiệu như đã nêu là: SDI, SCK, RCK, G đối với IC 74HC595 và SDI, CLK, LE, OE đối với IC MBI5026, các tín hiệu theo thứ tự thì chỉ khác tên, còn chức năng thì giống nhau.

Hoạt động của vi điều khiển đẩy dữ liệu ra thanh ghi dịch như sau:

- Vi điều khiển dịch dữ liệu nối tiếp ra chân RE2 và được đặt tên SSDO và xuất xung nhịp ở chân RE0 và được đặt tên SSCK để đẩy dữ liệu nối tiếp vào các thanh ghi dịch 74HC595 và MBI5026. Hai tên SSDI và SSCK có thêm chữ S cho khác với 2 tên SDO là SCK là do phần mềm đã định nghĩa cho chuẩn truyền dữ liệu SPI.
- Sau khi dịch xong toàn bộ dữ liệu thì vi điều khiển tạo 1 xung RCK_x để nạp dữ liệu từ thanh ghi dịch sang thanh ghi lưu trữ trong các thanh ghi dịch.
- Vi điều khiển tiến hành điều khiển chân G_x sang mức tích cực để cho phép xuất dữ liệu từ thanh ghi lưu trữ ra ngoài để điều khiển thiết bị.

Sơ đồ nguyên lý giao tiếp vi điều khiển với 2 IC chốt như hình 1-7:

Trong sơ đồ thì tín hiệu OE của IC chốt không dùng nên nối mức 0 để luôn cho phép xuất dữ liệu từ mạch chốt bên trong đến các ngõ ra Q.

**Hình 1-7. Sơ đồ giao tiếp port D của PIC 18F4550 với 2 IC chốt mở rộng.****Bảng 1-5. Các tín hiệu mở rộng của IC chốt 74HC573_A.**

TT	TÊN TÍN HIỆU	TÍN HIỆU ĐIỀU KHIỂN	MODULE NGOẠI VI
1	RCK_4LED7DOAN	Điều khiển tín hiệu RCK các thanh ghi dịch 74595	4 led 7 đoạn trực tiếp
2	G_4LED7DOAN	Điều khiển tín hiệu G các thanh ghi dịch 74595	
3	RCK_32LEDDON	Điều khiển tín hiệu RCK các thanh ghi dịch 74595	32 led đơn
4	G_32LEDDON	Điều khiển tín hiệu G các thanh ghi dịch 74595	
5	RCK_LCD20X4	Điều khiển tín hiệu RCK thanh ghi dịch MBI5026	LCD 20X4
6	G_LCD20X4	Điều khiển tín hiệu G thanh ghi dịch MBI5026	

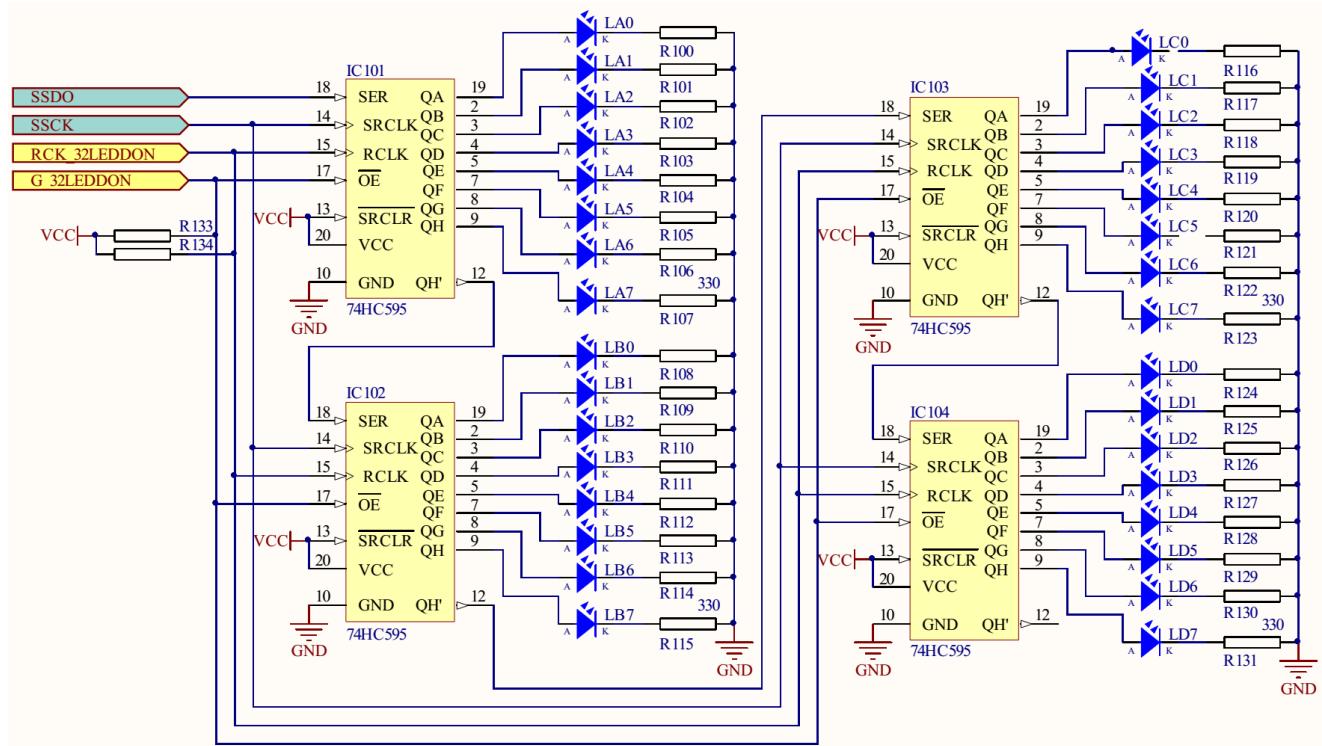
Bảng 1-6. Các tín hiệu mở rộng của IC chốt 74HC573_B.

TT	TÊN TÍN HIỆU	TÍN HIỆU ĐIỀU KHIỂN	MODULE NGOẠI VI
1	RCK_MATRANLED	Điều khiển tín hiệu RCK các thanh ghi dịch MBI5026	3 Led ma trận
2	G_MATRANLED	Điều khiển tín hiệu G các thanh ghi dịch MBI5026	
3	RCK_8LEDQUET	Điều khiển tín hiệu RCK thanh ghi dịch MBI5026	8 led 7 đoạn quét
4	G_8LEDQUET	Điều khiển tín hiệu G thanh ghi dịch MBI5026	
5	RCK_BUZERRELAY	Điều khiển tín hiệu RCK thanh ghi dịch 74595	Relay, buzzer, motor
6	G_BUZERRELAY	Điều khiển tín hiệu G thanh ghi dịch 74595	

1.2.2 Module 1: Giao tiếp với Led đơn

Một trong những ứng dụng đơn giản làm quen với lập trình là điều khiển 32 led đơn với các hiệu ứng quảng cáo từ đơn giản đến phức tạp. Để có thể điều khiển nhiều led sáng thì kit sử dụng IC thanh ghi dịch dữ liệu từ nối tiếp sang song song để điều khiển led như hình 1-8.

Vì điều khiển giao tiếp với module 32 led thông qua 4 IC thanh ghi dịch 74HC595 bằng 4 tín hiệu SSDO, SSCK, RCK_32LEDDON và G_32LEDDON.



Hình 1-8. Sơ đồ nguyên lý module 32 led đơn.

Mức 1 led sáng, mức 0 led tắt.

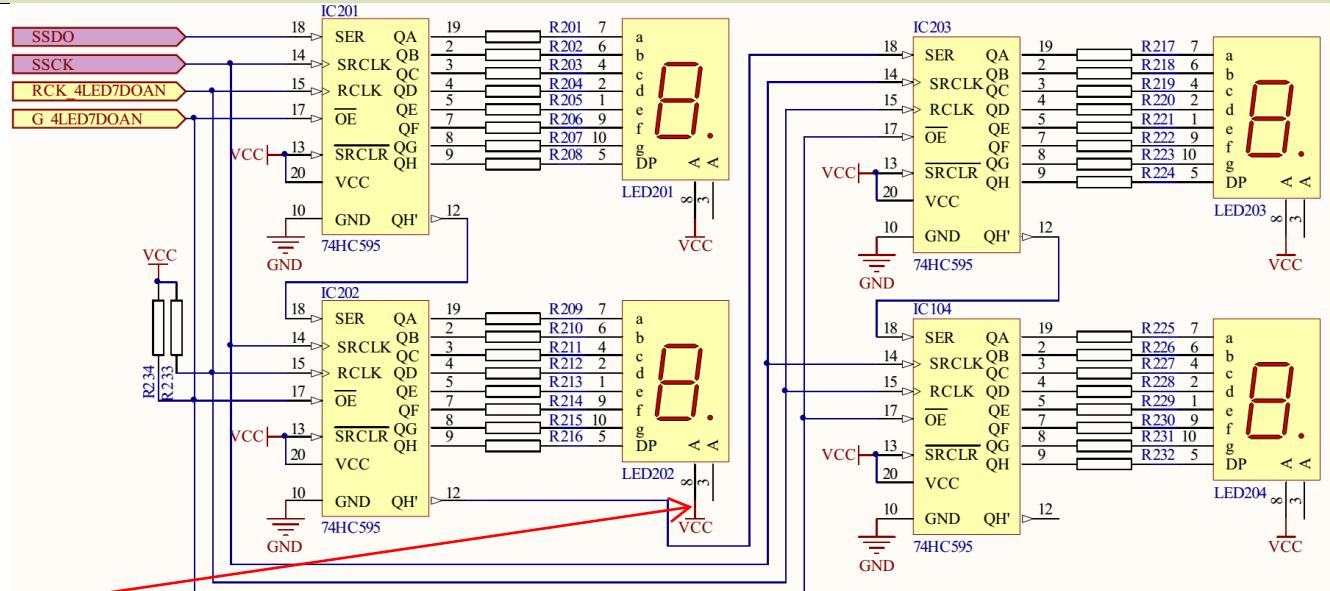
1.2.3 Module 2: 4 led 7 đoạn trực tiếp

Led 7 đoạn dùng để hiển thị số trong nhiều ứng dụng như máy đếm sản phẩm, máy đếm tiền, đèn giao thông, ...

Trong kit này có giao tiếp với 4 led 7 đoạn trực tiếp dùng 4 IC thanh ghi dịch 74HC595 có sơ đồ như hình 1-9.

Vì điều khiển giao tiếp với module 4 led 7 đoạn thông qua 4 IC thanh ghi dịch 74HC595 bằng 4 tín hiệu SSDO, SSCK, RCK_4LED7DOAN và G_4LED7DOAN.

Mức 0 led sáng, mức 1 led tắt.



LED ANODE CHUNG
 '0': SÁNG
 '1': TẮT

Hình 1-9. Sơ đồ nguyên lý module 4 led 7 đoạn.

Bảng 1-7. Bảng mã led 7 đoạn loại anode chung.

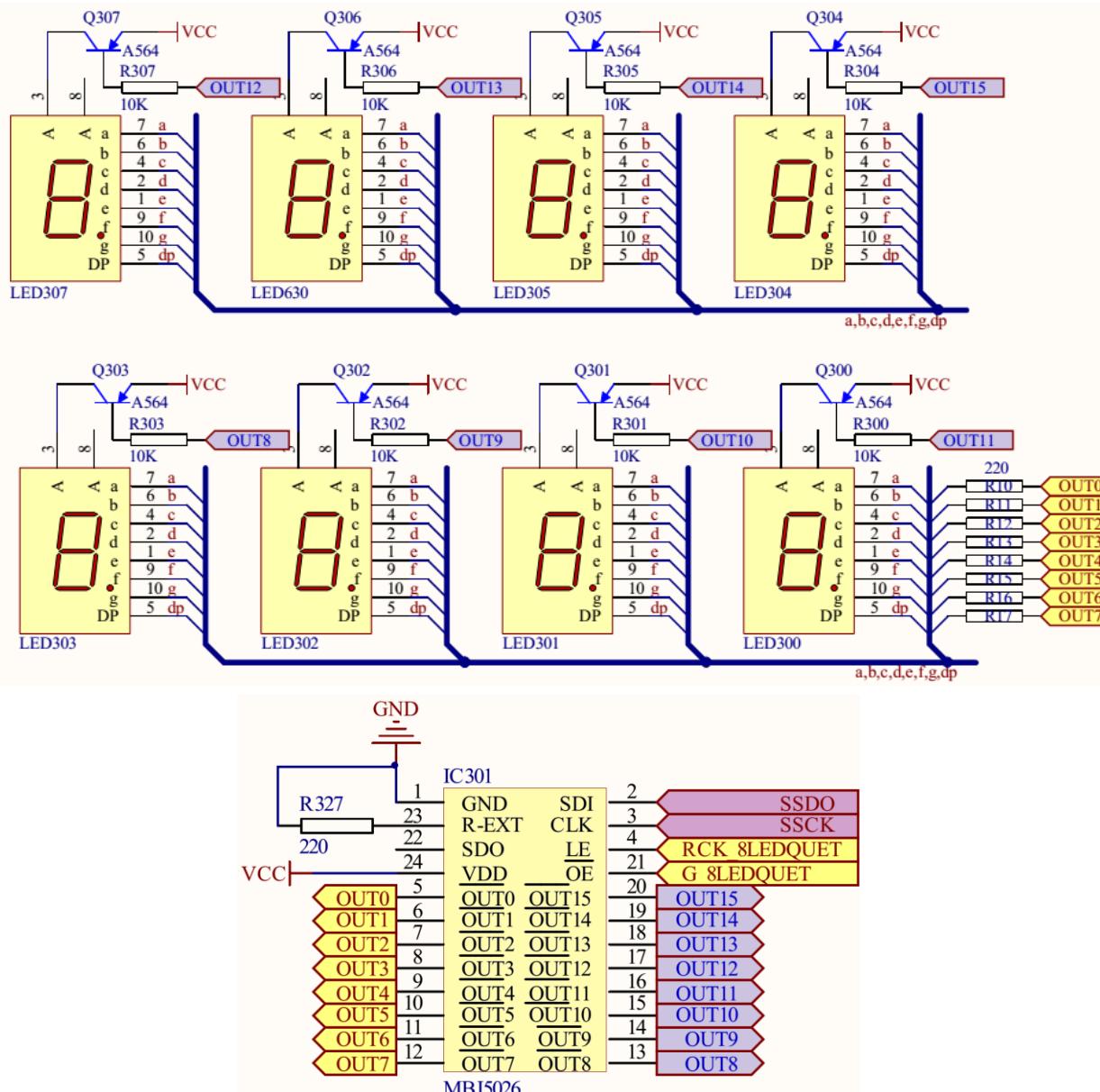
BIT 0

Số hex	dp	g	f	e	d	c	b	a	Mã số hex
0	1	1	0	0	0	0	0	0	C0
1	1	1	1	1	1	0	0	1	F9
2	1	0	1	0	0	1	0	0	A4
3	1	0	1	1	0	0	0	0	B0
4	1	0	0	1	1	0	0	1	99
5	1	0	0	1	0	0	1	0	92
6	1	0	0	0	0	0	1	0	82
7	1	1	1	1	1	0	0	0	F8
8	1	0	0	0	0	0	0	0	80
9	1	0	0	1	0	0	0	0	90
A	1	0	0	0	1	0	0	0	88
B	1	0	0	0	0	0	1	1	83
C	1	1	0	0	0	1	1	0	C6
D	1	0	1	0	0	0	0	1	A1
E	1	0	0	0	0	1	1	0	86

F	1	0	0	0	1	1	1	0	8E
---	---	---	---	---	---	---	---	---	----

1.2.4 Module 3: 8 led 7 đoạn quét

Phần này trình bày phương pháp điều khiển 8 led bằng phương pháp quét. Trong kit này có giao tiếp với 8 led 7 đoạn trực tiếp dùng 1 IC thanh ghi dịch 16 bit MBI 5026 có sơ đồ như hình 1-10.



Hình 1-10. Sơ đồ nguyên lý module 8 led 7 đoạn quét.

Vì điều khiển giao tiếp với module 8 led 7 đoạn quét thông qua IC thanh ghi dịch MBI 5026 bằng 4 tín hiệu SSDO, SSCK, RCK_8LEDQUET và G_8LEDQUET.

Mức 0 led sáng, mức 1 led tắt.

Trong dữ liệu 16 bit thì 8 bit xuất ra đầu tiên sẽ điều khiển transistor, bit thứ 0 điều khiển led thứ 0 bên phải, bit thứ 1, điều khiển led 1, tương tự bit thứ 7 điều khiển led 7 tận cùng bên trái.

Bit bằng 0 thì transistor dẫn led sáng, bit bằng 1 thì transistor tắt làm led tắt.

8 bit xuất ra sau cùng là mã 7 đoạn hiển thị trên led.

Khi điều khiển 8 led 7 đoạn quét thì phải gửi 2 byte: 1 byte quét led và 1 byte mã 7 đoạn.

IC thanh ghi dịch MBI 5026 dạng cực thu để hở nên phải dùng điện trở kéo lên để điều khiển transistor, các đoạn của led cùng với điện trở hạn dòng đóng vai trò là điện trở kéo lên.

Bảng 1-8. Bảng dữ liệu điều khiển quét lần lượt các transistor.

TT led sáng	Mã quét 8 led Chỉ cho 1 transistor dẫn để 1 led sáng									Mã số hex
	7	6	5	4	3	2	1	0		
7	1	1	1	1	1	1	1	0		FE
6	1	1	1	1	1	1	0	1		FD
5	1	1	1	1	1	0	1	1		FB
4	1	1	1	1	0	1	1	1		F7
3	1	1	1	0	1	1	1	1		EF
2	1	1	0	1	1	1	1	1		DF
1	1	0	1	1	1	1	1	1		BF
0	0	1	1	1	1	1	1	1		7F

1.2.5 Module 4: LCD 20×4

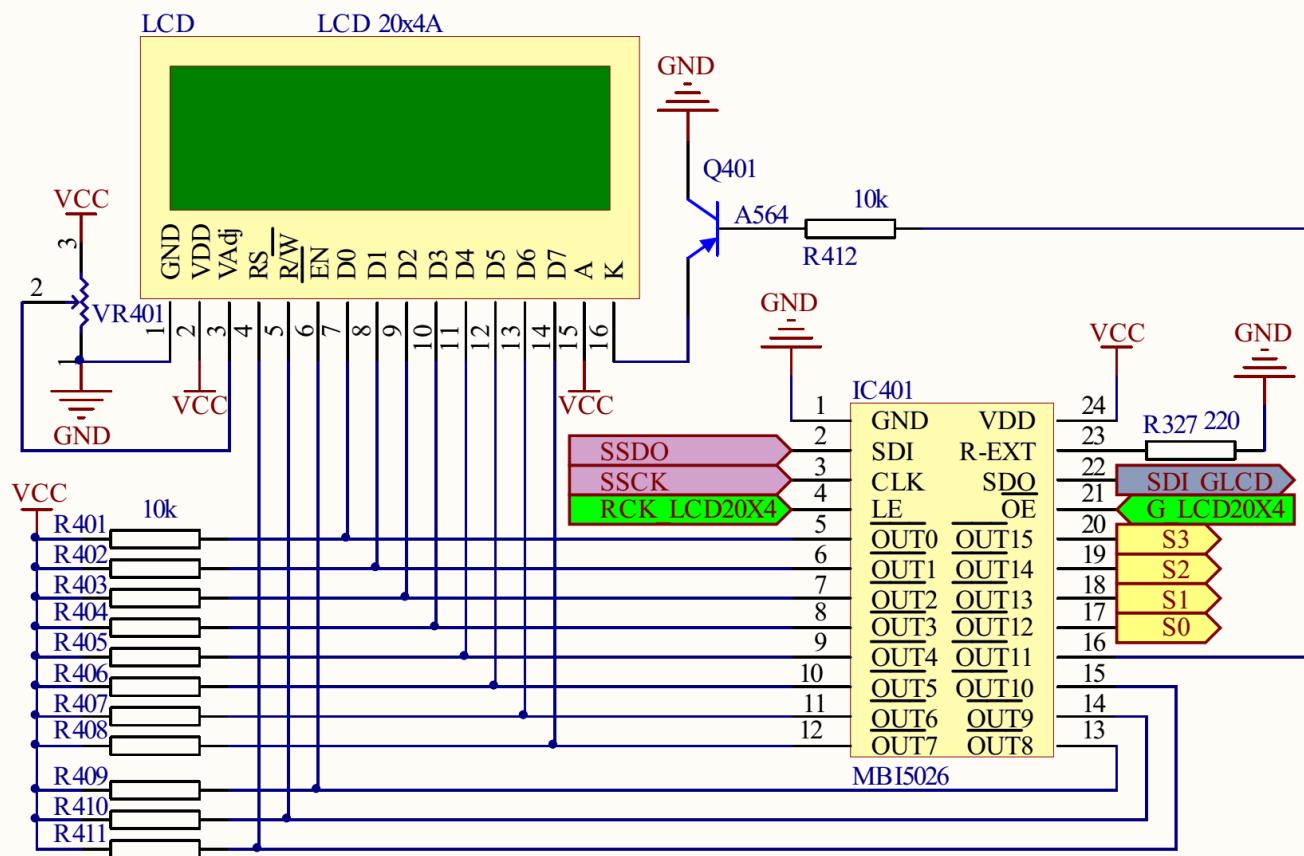
Module LCD dùng để hiển thị nhiều ký tự mã ASCII, kit vi điều khiển có tích hợp LCD 20×4 để có thể hiển thị được 4 hàng và mỗi hàng có 20 ký tự, sơ đồ mạch giao tiếp vi điều khiển với LCD qua IC thanh ghi dịch MBI5026 có sơ đồ như hình 1-11.

MBI có 16 ngõ ra kết nối điều khiển LCD như sau:

- Ngõ ra thứ 0 đến bit thứ 7 dùng giao tiếp dữ liệu với LCD.
- Ngõ ra thứ 8, 9, 10, 11 là các tín hiệu E, RW, RS và điều khiển transistor nguồn của đèn nền. Mức 0 làm transistor dẫn cho đèn nền sáng, mức 1 hoặc trạng thái tống trở cao làm transistor tắt, đèn nền tắt.
- Bit thứ 12 đến 15 giao tiếp với cảm biến màu sắc.

Trong thư viện đã định nghĩa sẵn các bit để điều khiển LCD.

Ngõ ra dữ liệu của IC MBI 5026 sẽ nối đến ngõ vào của IC MBI 5026 điều khiển GLCD.



Hình 1-11. Sơ đồ nguyên lý module LCD 20×4.

1.2.6 Module 5: GLCD 128×64

Module GLCD dùng để hiển thị hình ảnh và ký tự, sơ đồ mạch giao tiếp vi điều khiển với GLCD qua IC MBI5026 có sơ đồ như hình 1-12.

MBI có 16 ngõ ra kết nối điều khiển GLCD như sau:

Ngõ ra từ thứ 0 đến thứ 7 làm 8 bit giao tiếp dữ liệu.

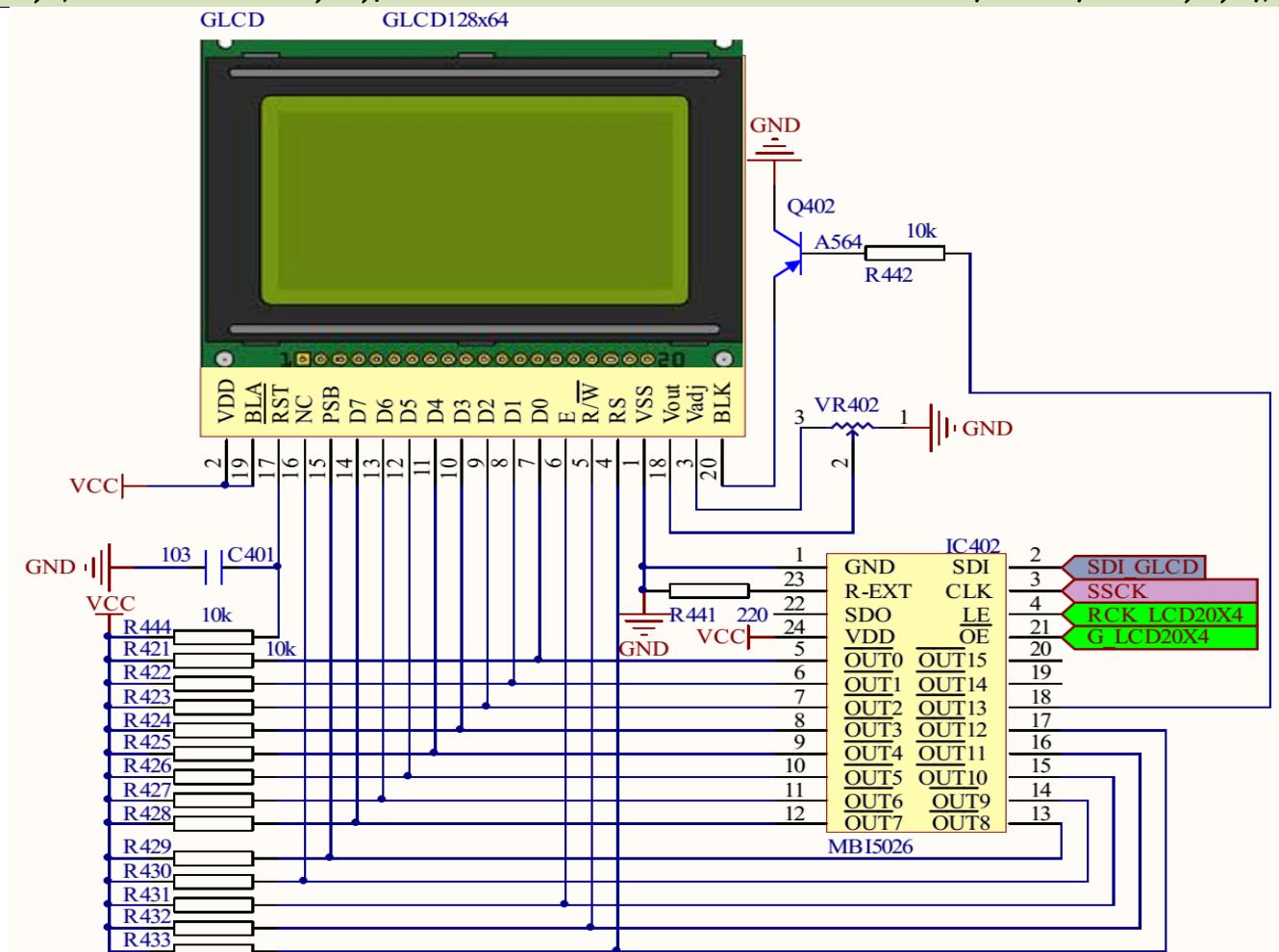
Hai ngõ ra thứ 8, 9 làm 2 tín hiệu chọn chip CS1, CS2.

Ba ngõ ra tiếp theo 10, 11, 12 làm 3 bit điều khiển E, R/W, RS.

Ngõ ra thứ 13 làm bit điều khiển transistor mở nguồn cho GLCD khi cần sử dụng. Mức 0 làm transistor dẫn, đèn anode sáng, mức 1 làm led tắt.

Do LCD và GLCD nối tiếp nhau nên có 3 chế độ sử dụng;

- Chỉ điều khiển LCD
- Chỉ điều khiển GLCD
- Điều khiển cả 2 LCD và GLCD.



Hình 1-12. Sơ đồ nguyên lý module GLCD 128×64.

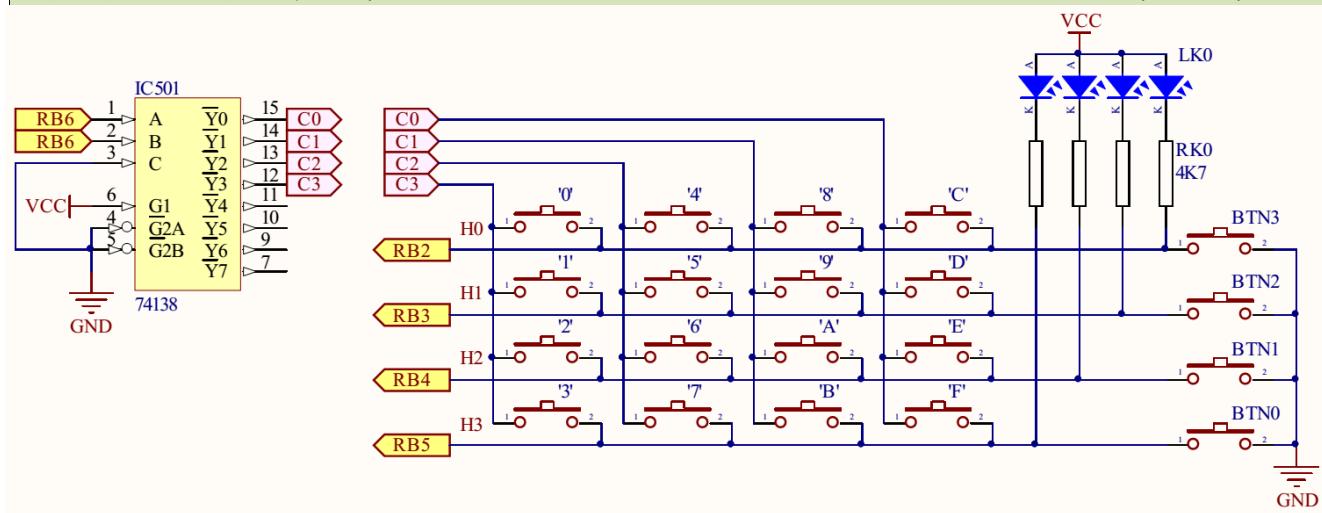
1.2.7 Module 6: phím đơn và phím ma trận 4×4

Để làm quen với cách thức giao tiếp bàn phím thì kit có thiết kế giao tiếp 4 phím nhấn đơn và phím ma trận 4×4 được 16 phím có sơ đồ nguyên lý như hình 1-13.

Vì điều khiển giao tiếp với module 4 nút nhấn đơn BT0, BT1, BT2, BT3 sử dụng 4 bit RB2, RB3, RB4 và RB5. Các tín hiệu đều có điện trở kéo lên nguồn và có led, khi nhấn thì led sáng mức logic là 0, khi không nhấn thì mức logic là 1, led tắt.

Phím ma trận 4×4 sử dụng 4 bit RB2, RB3, RB4 và RB5 đóng vai trò là hàng (H0, H1, H2, H3) có dùng điện trở và led kéo lên nguồn. Hai bit RB6, RB7 qua IC giải mã tạo thành 4 ngõ ra làm 4 cột C0, C1, C2 và C3.

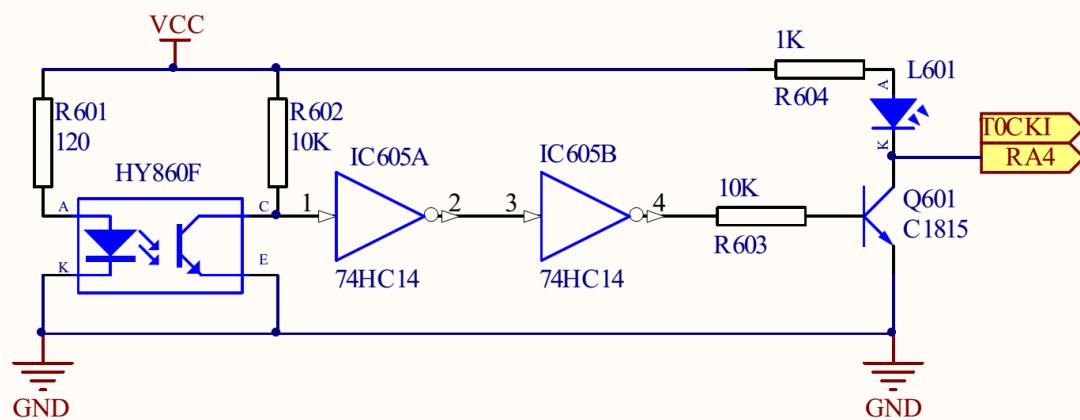
Tổ hợp 4 hàng và 4 cột dùng để quét bàn phím ma trận.



Hình 1-13. Sơ đồ nguyên lý module phím nhấn đơn và phím ma trận.

1.2.8 Module 7: thu phát hồng ngoại tạo xung

Kit có tích hợp module thu phát hồng ngoại tạo xung khi có sản phẩm che chắn giữa led phát và led thu, có led sáng tắt để báo hiệu, ngõ ra của xung đưa đến timer/counter để thực hiện các bài đếm xung ngoại dùng counter. Sơ đồ mạch như hình 1-14.



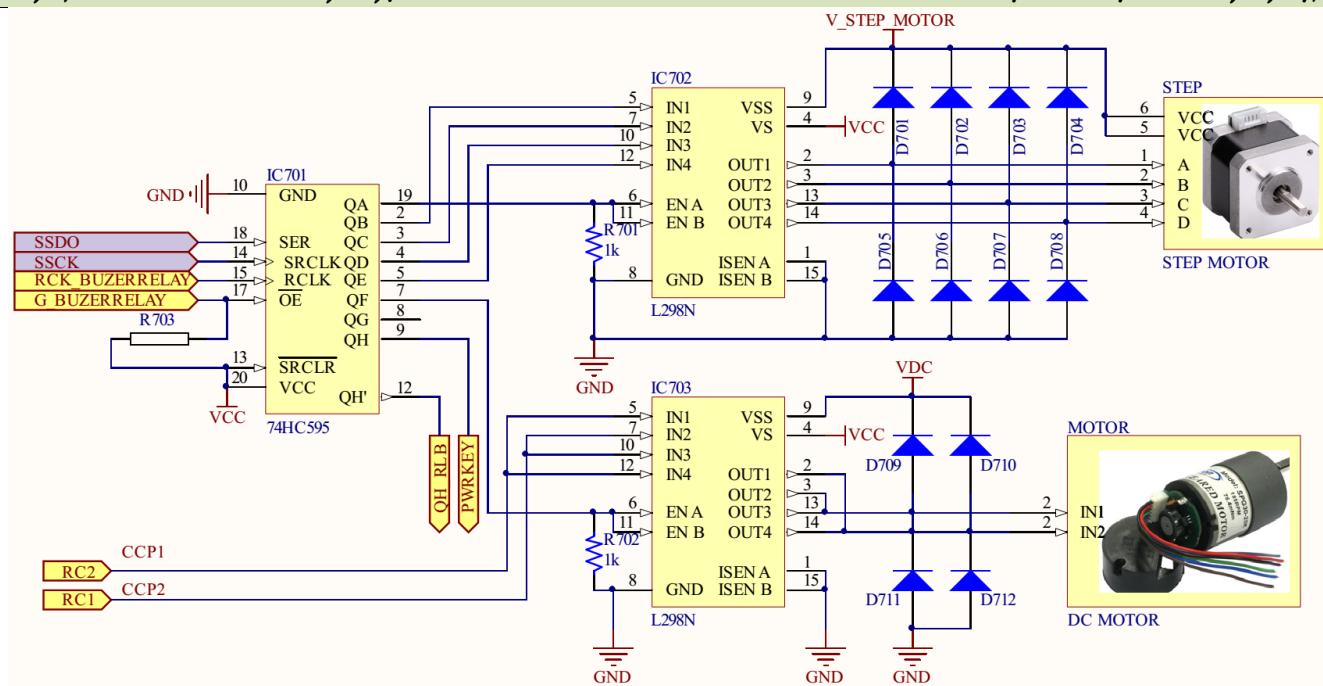
Hình 1-14. Sơ đồ nguyên lý module thu phát hồng ngoại tạo xung.

1.2.9 Module 8: giao tiếp relay, buzzer và động cơ

Kit có tích hợp module giao tiếp điều khiển động cơ bước và động cơ DC, Triac, Relay, Buzzer thông qua 2 thanh ghi dịch 74HC595.

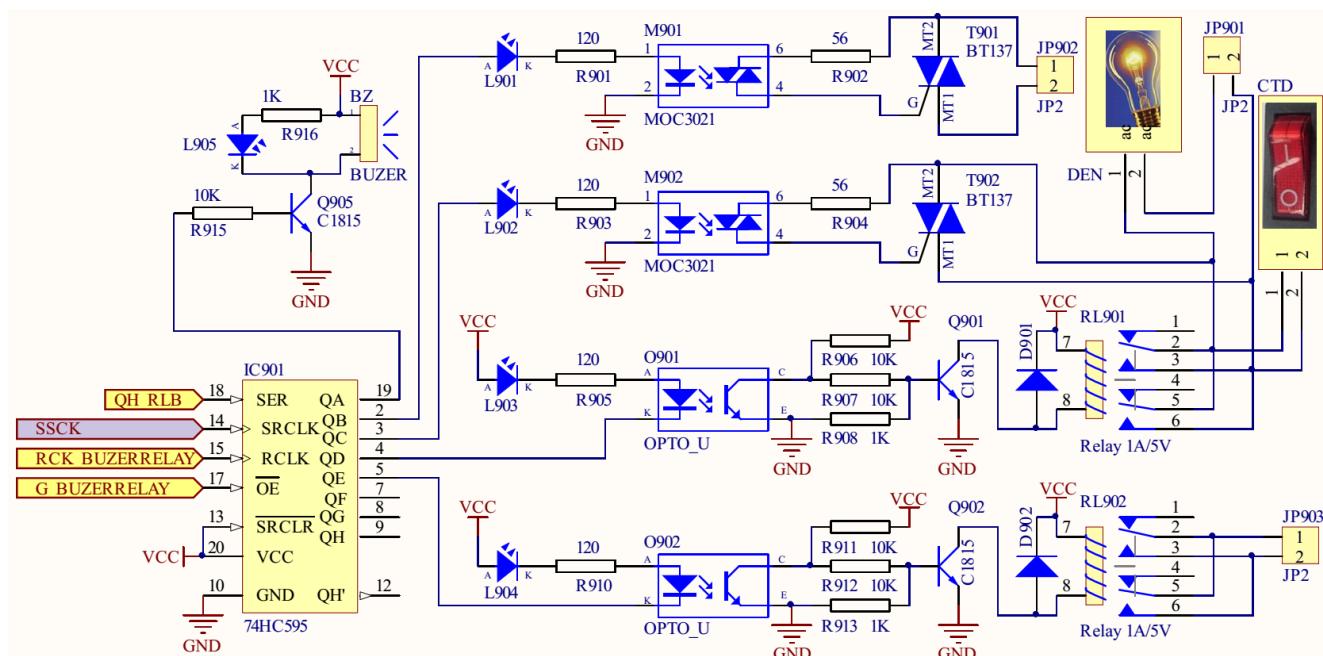
IC thanh ghi dịch 74HC595 thứ nhất có sơ đồ mạch như hình 1-15.

- Năm ngõ ra từ QA đến QE giao tiếp với IC công suất L298 để điều khiển động cơ bước.
- Ngõ ra QF để giao tiếp với IC công suất L298 để điều khiển cho phép động cơ DC cùng với 2 tín hiệu CCP1 và CCP2 của vi điều khiển để điều khiển động cơ: chạy, ngừng, thay đổi tốc độ.
- Ngõ ra QH để giao tiếp điều khiển nguồn của SIM900.
- Ngõ ra QH* để chuyển dữ liệu sang IC 74HC595 thứ 2.

**Hình 1-15. Sơ đồ nguyên lý module điều khiển động cơ bước và động cơ DC.**

Động cơ DC sử dụng nguồn 24V, động cơ bước dùng nguồn 6.7V nên sử dụng nguồn điều chỉnh bằng LM2576-ADJ, áp được điều chỉnh cho phù hợp với động cơ bước.

IC thanh ghi dịch 74HC595 thứ hai có sơ đồ mạch như hình 1-16.

**Hình 1-16. Sơ đồ nguyên lý module điều khiển Relay, Triac, tải 220V AC.**

- Ngõ ra QA điều khiển tắt mở Buzzer, mức 1 làm Buzzer kêu, mức 0 làm Buzzer tắt.
- Ngõ ra QB, QC điều khiển tắt mở 2 opto triac MOC3021 để điều khiển triac công suất, ngõ ra triac nối với domino.

- Ngõ ra QD, QE điều khiển tắt mở 2 relay để điều khiển relay công suất, ngõ ra relay nối với domino.

Chú ý: trong sơ đồ thì có dùng 1 Relay và 1 Triac nối với tải là bóng đèn 25W qua một contact (CTD) để điều khiển đèn sáng hay tắt làm gia tăng nhiệt độ cho các cảm biến nhiệt.

Ở các bài đo nhiệt độ dùng cảm biến nếu muốn thấy sự thay đổi nhiệt độ thì mở công tắc đèn (CTD) làm đèn sáng để tăng nhiệt cho đèn nhiệt độ mong muốn, 4 cảm biến nhiệt được đặt gần bóng đèn để hấp thụ nhiệt.

Ở các bài đo nhiệt độ dùng cảm biến tự động tắt khi quá nhiệt và tự động mở khi nhiệt độ thấp hơn thì ta sử dụng Relay để điều khiển bóng đèn.

Động cơ DC có gắn Encoder để tạo xung và xung đưa về counter T1.

1.2.10 Module 9: giao tiếp ADC, thời gian thực, bộ nhớ Eeprom và các cảm biến

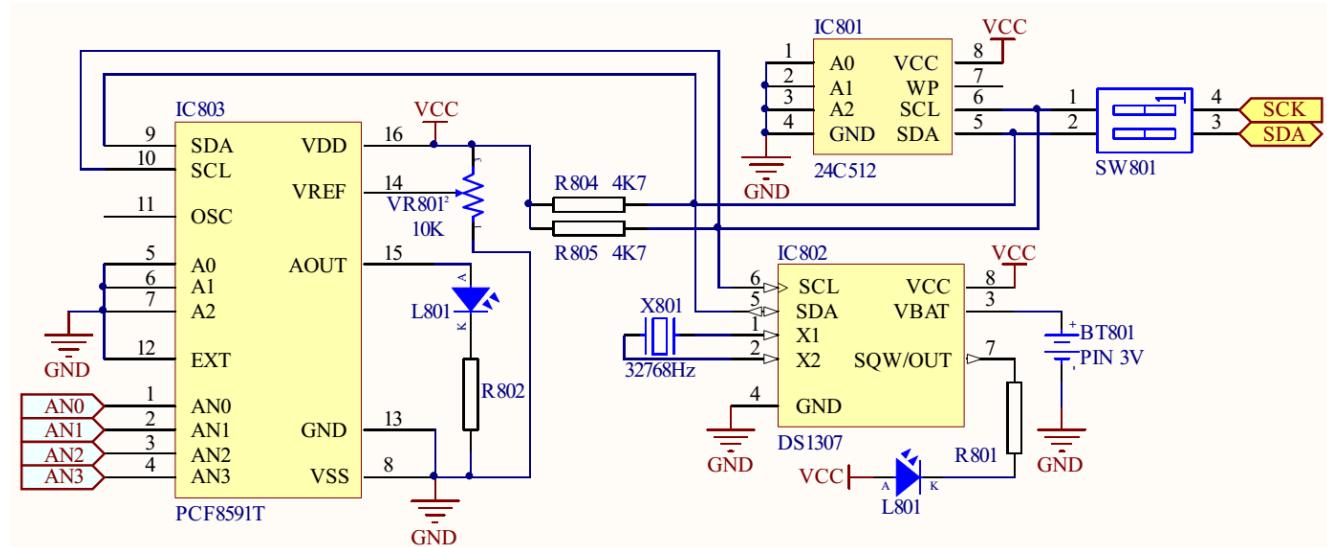
Kit có tích hợp các thiết bị giao tiếp với vi điều khiển theo chuẩn I2C bao gồm: ADC 8 bit PCF8591T, IC thời gian thực DS1307 và IC nhớ Eeprom AT24C512 có sơ đồ mạch như hình 1-17. Các tín hiệu của I2C là SDA và SCL sẽ nối với SDA và SCL của vi điều khiển thông qua switch SW801 và có 2 điện trở kéo lên nguồn.

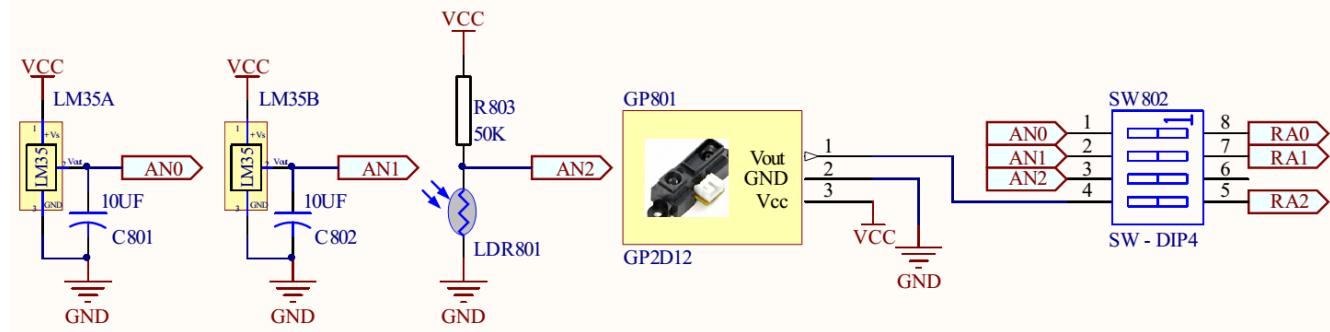
IC thời gian thực có nguồn pin để khi mất điện IC vẫn hoạt động. Ngõ ra SQW/OUT dạng cực thu để hở và được nối với led.

IC PCF8591T có 4 kênh ADC từ AN0 đến AN3. Có 2 cảm biến LM35 nối với kênh AN0 và AN1, cảm biến quang nối với kênh AN2 và cảm biến khí ga ở phần sau thì nối với kênh AN3.

Hai cảm biến nhiệt LM35 và cảm biến khoảng cách GP2D12 còn nối với ADC của vi điều khiển PIC 18F4550 thông qua switch SW802.

PCF8951T còn có 1 kênh DAC với ngõ ra là AOUT nối với led và có biến trờ VR801 để chỉnh độ phân giải cho khối ADC và DAC.





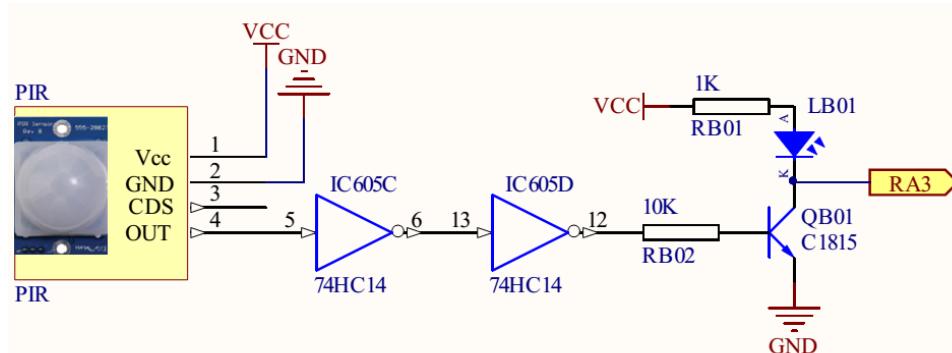
Hình 1-17. Sơ đồ nguyên lý module giao tiếp I2C và các cảm biến tương tự.

Cảm biến nhiệt LM35 có thêm 1 tụ điện ở ngõ ra để giảm bớt độ nhạy để đo cho ổn định hơn. Cảm biến ánh sáng có điện trở nối lên nguồn có giá trị từ $10k\Omega$ trở lên.

Các cảm biến trên đều là cảm biến có điện áp ra dạng tương tự.

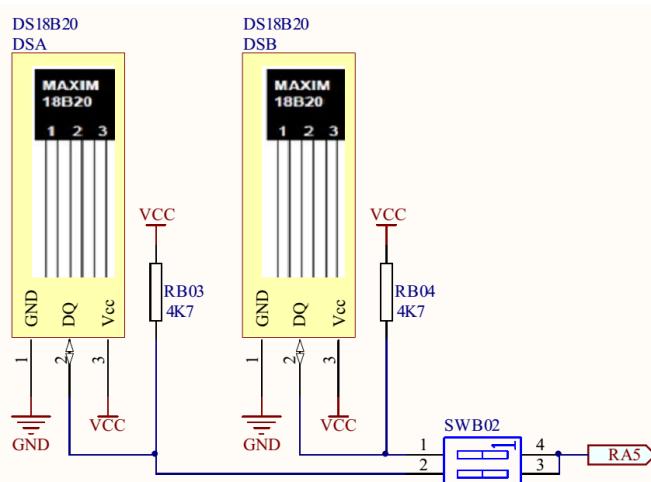
1.2.11 Module 10: giao tiếp các cảm biến nhiệt, cảm biến khoảng cách, led thu

Kit có tích hợp cảm biến phát hiện người chuyển động PIR và có led báo hiệu dùng để làm mạch báo trộm nối với chân tín hiệu AN3/RA3.



Hình 1-18. Sơ đồ nguyên lý module cảm biến hồng ngoại PIR.

Có 2 cảm biến nhiệt tích hợp ADC bên trong xuất nhập tín hiệu bằng 1 dây còn gọi là cảm biến 1 dây (one wire) DS18B20. Do có 2 cảm biến nên phân biệt là A và B, có pin-header 2 chân và có switch SWB02 để có thể nối 1 hoặc 2 cảm biến với 1 chân RA5. Sơ đồ như hình 1-19.



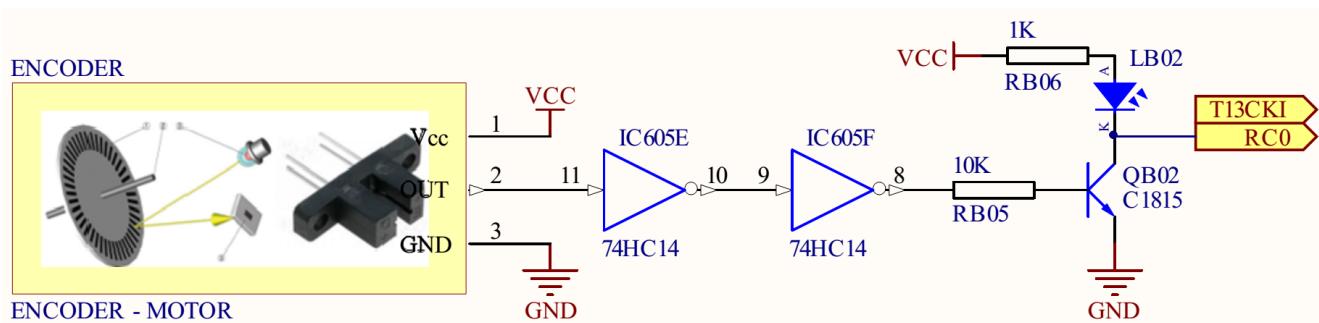
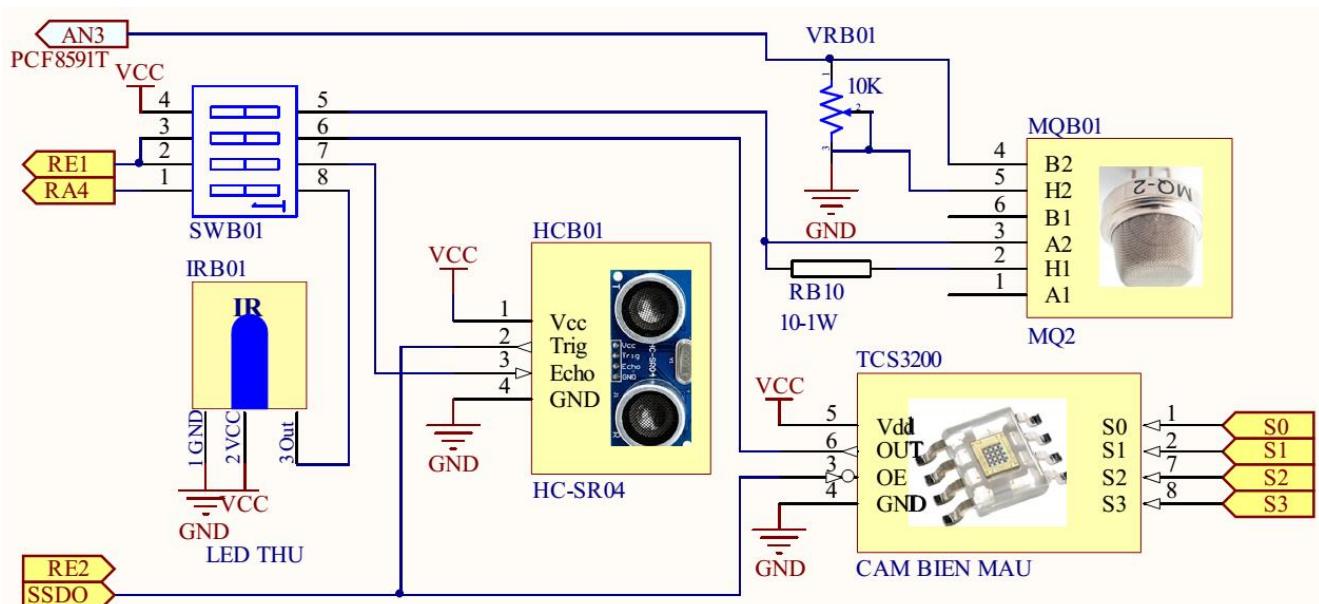
Hình 1-19. Sơ đồ nguyên lý module 2 cảm biến nhiệt DS18B20.

Có các bài thực hành cho 2 cảm biến như sau:

Bài thực hành chỉ dùng 1 trong 2 cảm biến: Với bài dạng này thì không cần biết và không cần đọc mã bộ nhớ của cảm biến vì chỉ có 1 cảm biến. Khi thực hành ở dạng này thì chuyển 1 switch bất kỳ của SWB02 sang vị trí ON, cái còn lại ở vị trí OFF.

Bài thực hành dùng cả 2 cảm biến: với bài dạng này thì cần phải biết code của bộ nhớ của 2 cảm biến, khi truy xuất cảm biến nào thì phải gởi địa chỉ của cảm biến đó mới truy xuất được. Khi thực hành ở dạng này thì chuyển cả 2 switch của SWB02 sang vị trí ON.

Motor DC của kit có gắn sẵn Encoder và được đưa vào mạch Schmitt trigger để hiệu chỉnh tín hiệu rồi đưa vào chân RC0/T1CK13 như hình 1-20.

**Hình 1-20. Sơ đồ nguyên lý module kết nối Encoder của motor.****Hình 1-21. Sơ đồ nguyên lý module cảm biến led thu, siêu âm, khí ga, màu.**

Kit có cảm biến Gas, khi dùng thì chuyển SWB01(4) sang ON để mở nguồn, khi không dùng thì chuyển sang OFF vì dòng tiêu thụ của cảm biến lớn. Ngõ ra của cảm biến được đưa đến khối ADC PCF8591T chân AN3.

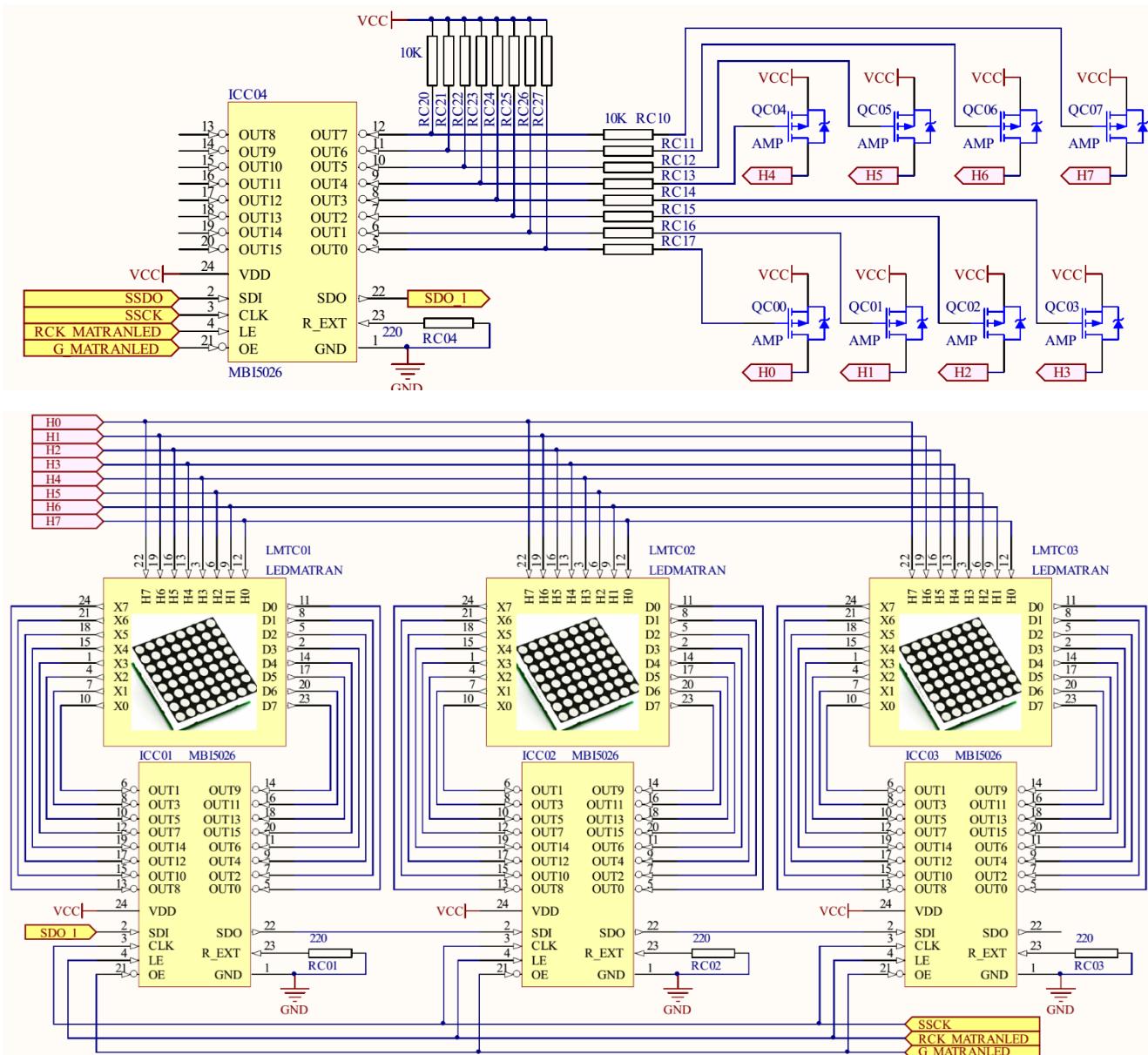
Có led thu hồng ngoại có thể dùng remote để điều khiển từ xa. Khi dùng thì chuyển SWB01(1) sang ON, khi không dùng thì chuyển sang OFF.

Có 1 cảm biến khoảng cách dùng sóng siêu âm cho kết quả khoảng cách tỉ lệ với tần số xung. Khi dùng thì chuyển SWB02(2) sang ON, SWB02(3) sang OFF, khi không dùng thì chuyển sang OFF.

Có 1 cảm biến màu sắc. Khi dùng thì chuyển SWB02(3) sang ON, SWB02(1) sang OFF, khi không dùng thì chuyển sang OFF.

1.2.12 Module 11: giao tiếp led ma trận

Một trong những ứng dụng phổ biến trong quảng cáo là thông tin được hiển thị trên led ma trận, để giúp người học hiểu được nguyên lý điều khiển led ma trận như thế nào thì trong hệ thống có thiết kế giao tiếp với 3 led ma trận 8x8 hai màu xanh và đỏ. Sơ đồ nguyên lý trình bày ở hình 1-22.



Hình 1-22. Sơ đồ nguyên lý mạch điều khiển 3 led ma trận.

Mỗi led ma trận 8x8 2 màu có tổng cộng 24 tín hiệu điều khiển gồm 8 tín hiệu anode chung và 8 tín hiệu cathode chung cho màu đỏ và 8 tín hiệu cathode chung cho màu xanh.

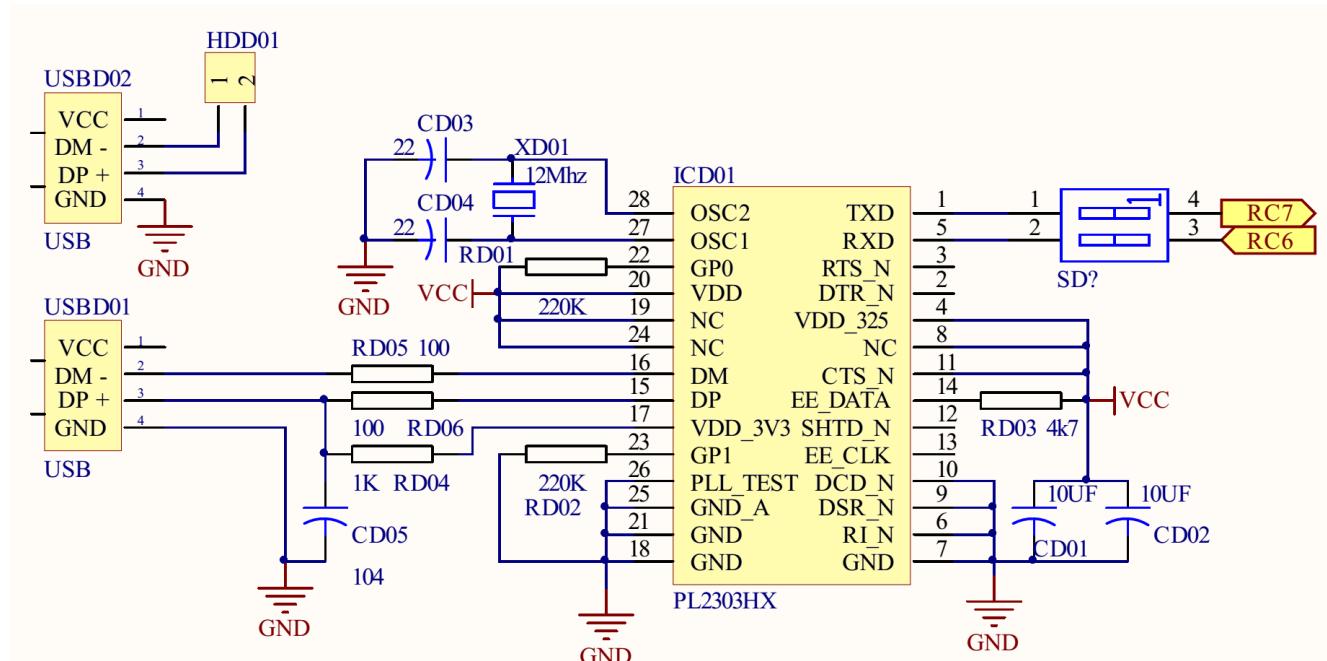
Tín hiệu cathode màu xanh và màu đỏ của mỗi led sẽ do một IC MBI 5026 điều khiển, 3 led sẽ dùng 3 IC MBI 5026.

Tín hiệu anode chung của 3 led được nối chung với nhau và trong sơ đồ được đặt tên là H0 đến H7. Do sử dụng phương pháp quét nên phải khuếch đại dòng bằng transistor Mosfet AMP 4935, tín hiệu từ vi điều khiển được chuyển từ nối tiếp sang song song lấy ra ở 8 ngõ ra từ out0 đến out7 điều khiển 8 transistor để điều khiển anode của 3 led ma trận.

4 IC MBI 5026 được mắc nối tiếp đường dữ liệu SSDO, SSDO của vi điều khiển sẽ kết nối với SDI của IC C04, ngõ ra dữ liệu nối tiếp SDO của IC C01 kết nối với ngõ vào SDI của IC C02, tương tự cho các IC còn lại. Các tín hiệu còn lại SSCK, RCK_MATRANLED, G_MATRANLED thì nối song song.

1.2.13 Module 12: giao tiếp máy tính bằng USB qua IC chuyển đổi RS232

Kit có tích hợp mạch chuyển đổi chuẩn giao tiếp USB thành RS232 do IC PL2303HX để thực hiện các bài truyền dữ liệu giữa máy tính và vi điều khiển, sơ đồ nguyên lý trình bày ở hình 1-23.

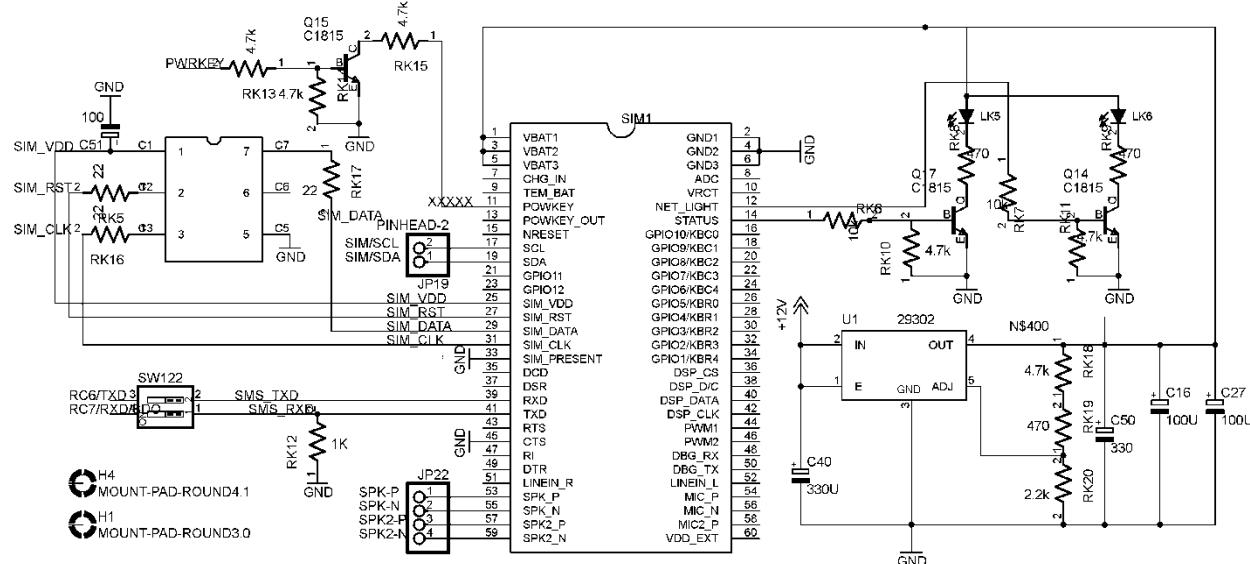


Hình 1-23. Sơ đồ nguyên lý mạch giao tiếp máy tính bằng cổng USB qua IC chuyển đổi.

Cổng USBD01 dùng để giao tiếp với máy tính theo chuẩn RS232, cổng USBD02 nối với pin-header HDD01 để thực hiện các bài giao tiếp máy tính với vi điều khiển có tích hợp chuẩn giao tiếp USB như PIC18F4550.

1.2.14 Module 13: giao tiếp SIM900-TE-C-V1.02

Kit có tích hợp mạch giao tiếp module SIM900-TE-C-V1.02, sơ đồ nguyên lý trình bày ở hình 1-24.

**Hình 1-24. Sơ đồ nguyên lý mạch giao tiếp SIM900.**

Vì điều khiển giao tiếp với SIM thông qua 3 tín hiệu, 1 tín hiệu điều khiển khởi tạo SIM có tên là PWRKEY, 2 tín hiệu giao tiếp truyền dữ liệu theo chuẩn UART là RXD, TXD thông qua SW122 nối đến 2 tín hiệu RC6/TXD và RC7/RXD.

Chương 2: SỬ DỤNG PHẦN MỀM CCS VÀ PICKIT

2.1 GIỚI THIỆU

Phần mềm lập trình cho vi điều khiển PIC có nhiều chương trình khác nhau, tài liệu này trình bày cách sử dụng phần mềm CCS.

Nếu máy tính bạn sử dụng chưa cài phần mềm CCS thì tiến hành copy chương trình CCS vào máy tính và cài đặt theo hướng dẫn của phần mềm.

Biểu tượng phần mềm CCS sau khi cài xong xuất hiện trên desktop như hình sau:



Hình 2-1. Biểu tượng phần mềm CCS.

2.2 HƯỚNG DẪN SỬ DỤNG PHẦN MỀM CCS

Sau khi cài đặt xong thì tiến hành soạn thảo và biên dịch chương trình theo trình tự sau:

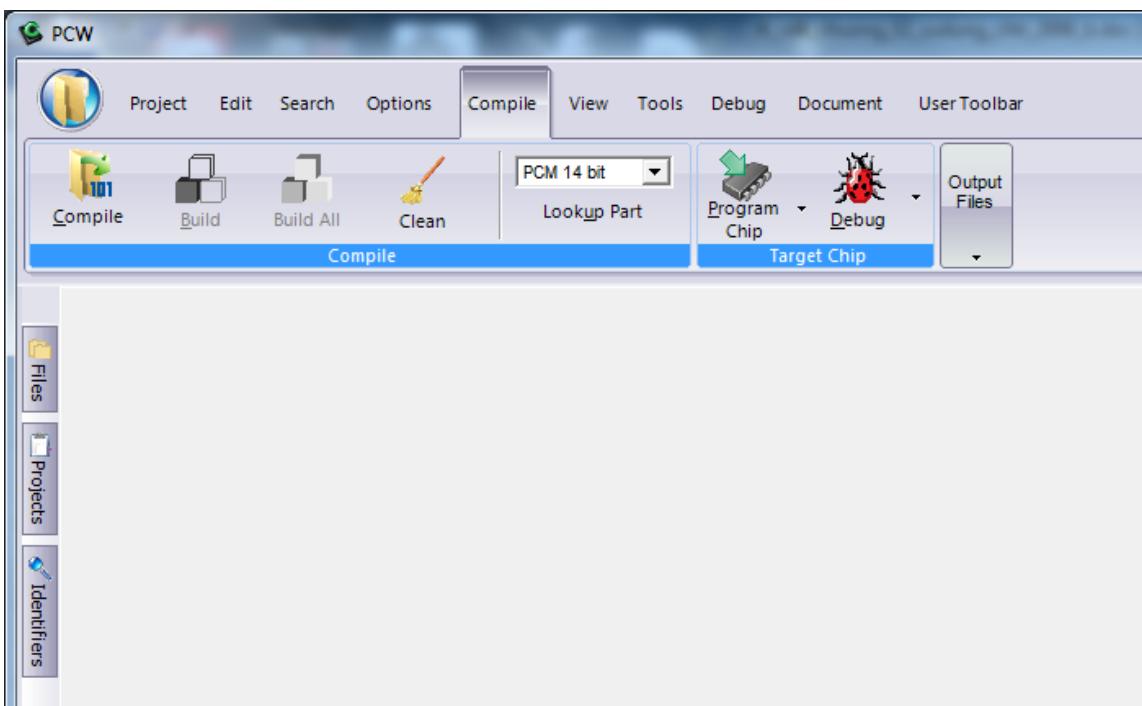
2.2.1 Biên soạn chương trình cơ bản

Bước 1: Nên tạo 1 thư mục để lưu các chương trình lập trình cho vi điều khiển.

Tài liệu này sử dụng thư mục “D:\TH_VDK_PIC_N1_SANG_THU_2”

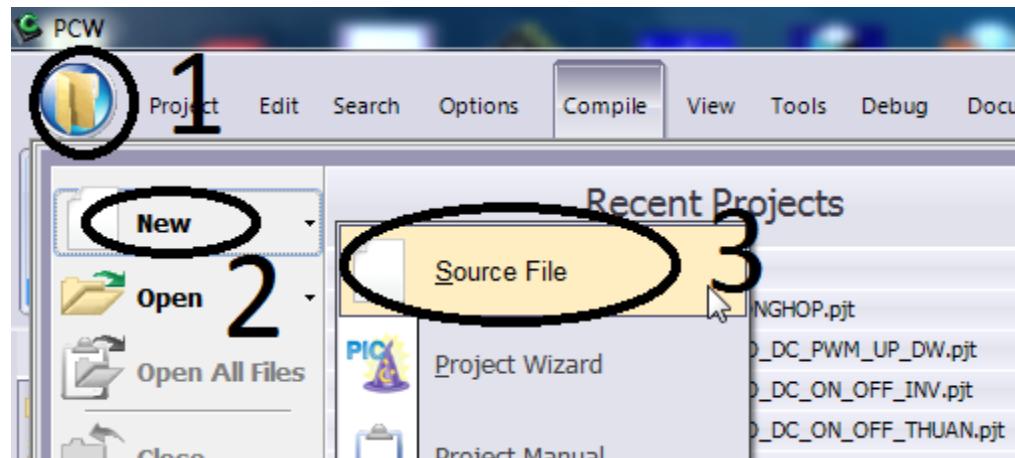
Bước 2: Khởi tạo chương trình CCS, giao diện xuất hiện như hình 2-2.

Thường thì phần mềm sẽ khởi tạo và mở một chương trình soạn thảo sau cùng hoặc mở một ứng dụng có trong phần mềm.



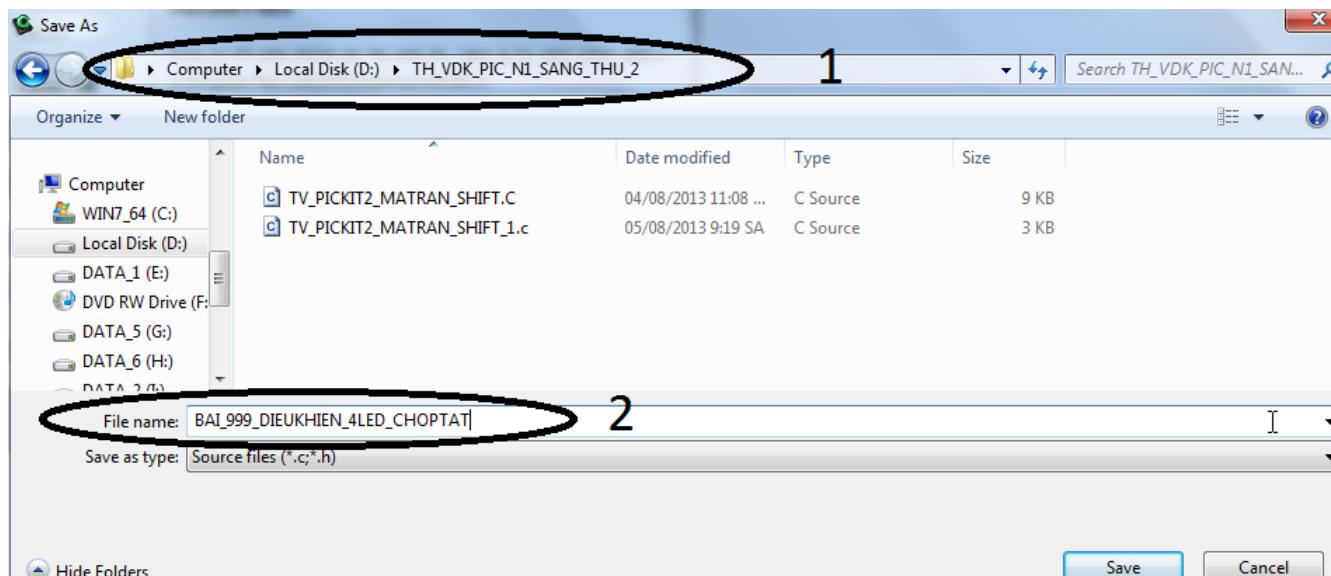
Hình 2-2. Giao diện phần mềm CCS.

Bước 3: Tiến hành chọn biểu tượng open, rồi chọn mục “New” và chọn “New source” theo thứ tự 1, 2, 3 như trong hình 2-3:



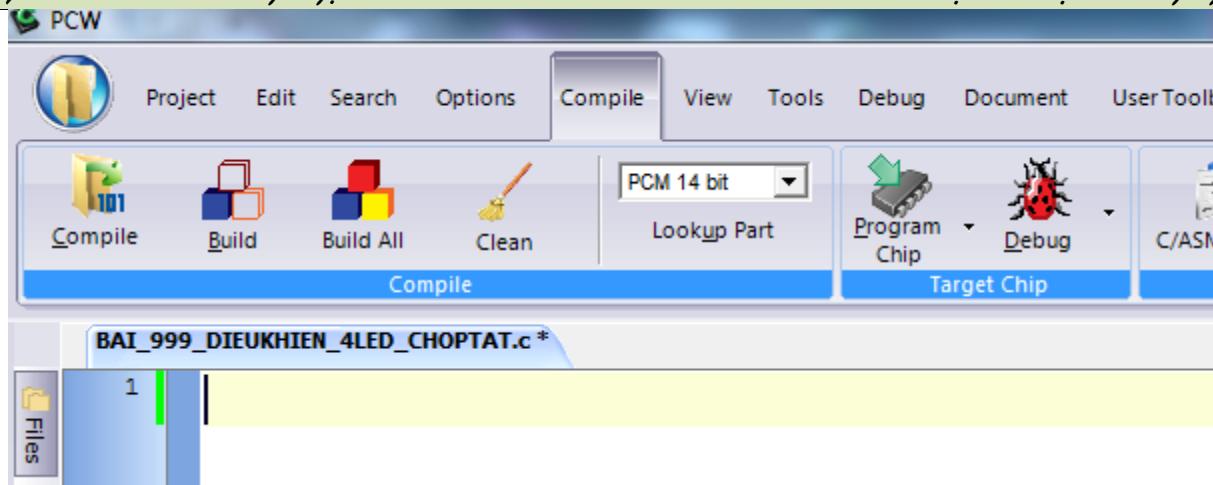
Hình 2-3. Tạo file mới.

Chọn new và chọn thư mục “D:\TH_VDK_PIC_N1_SANG_THU_2” để lưu file nguồn và đánh tên file theo thứ tự 1, 2 như trong hình 2-4:

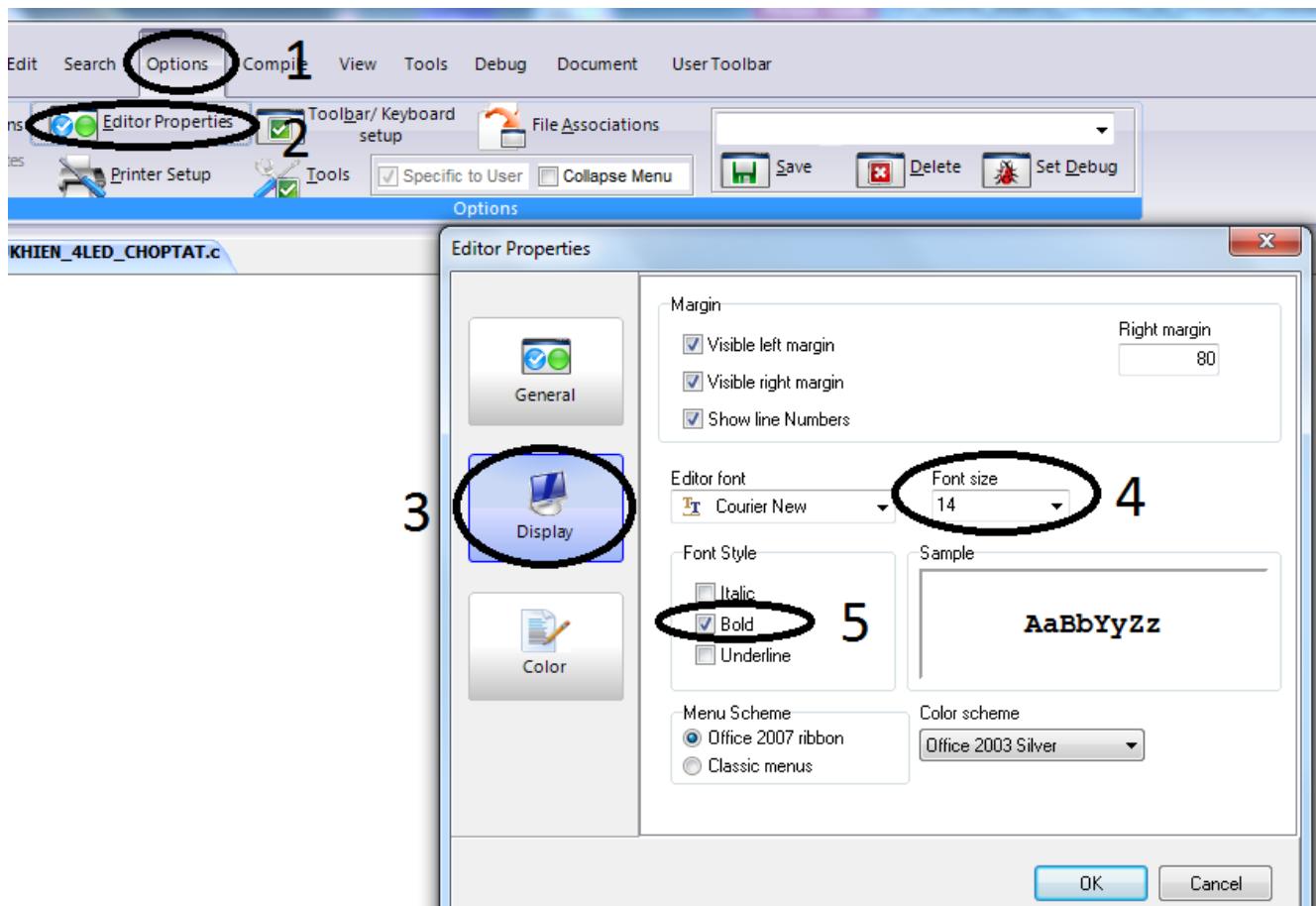


Hình 2-4. Đặt tên file mới và đường dẫn.

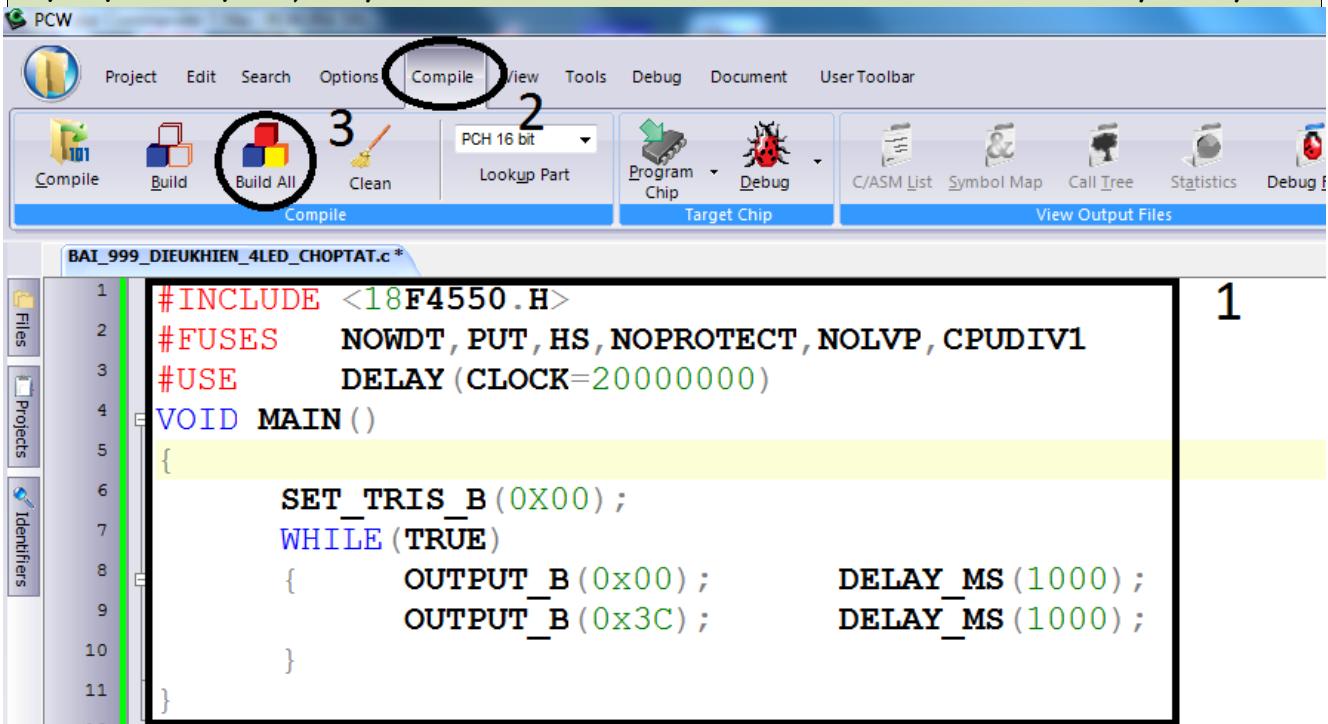
Giao diện sau xuất hiện như hình 2-5 và trong file chưa có gì:

**Hình 2-5. Màn hình soạn thảo.**

Bước 4: điều chỉnh font chữ cho phù hợp: tiến hành chọn mục “Option” (số 1), chọn “editor Properties” (số 2), chọn tiếp “Display” (số 3), chọn tiếp “Font size” (số 4) cho phù hợp và nên chọn là 14 và chọn mục Bold (số 5), tất cả theo trình tự từ 1 đến 5.

**Hình 2-6. Màn hình thay đổi font.**

Bước 5: sau khi điều chỉnh xong thì tiến hành biên soạn chương trình chép tắt đơn giản như hình như hình 2-7:



Hình 2-7. Màn hình của chương trình chớp tắt 4 led.

Tiến hành đánh các lệnh của chương trình trong hình vuông vào (số 1).

Sau khi đánh xong thì nhấn tổ hợp “ctrl” và “S” để lưu file.

Chương trình này có chức năng chớp tắt 4 led nối với 4 bit: bit thứ 2, 3, 4 và 5 của portB – xem hình 1-12 của chương 1. Kit vi điều khiển sử dụng portB giao tiếp với bàn phím ma trận và có kết nối 4 led để báo hiệu – khi không nhấn thì đèn tắt, khi nhấn thì đèn sáng, ta sử dụng 4 led này để thực hiện bài lập trình đầu tiên cho đơn giản.

Trong chương trình thì hành đầu tiên là thư viện của vi điều khiển PIC18F4550, file này chính là file gốc của phần mềm CCS.

Hàng lệnh thứ 2 là khai báo cấu hình: không sử dụng bộ định thời giám sát (No Watchdog Timer: NOWDT), sử dụng bộ định thời kéo dài thời gian reset vi điều khiển thêm 72ms để chờ nguồn điện ổn định sau khi vi điều khiển được mở nguồn (Power Up Timer: PUT), chọn bộ dao động tần số cao từ 4MHz đến 20MHz (High Speed: HS), không bảo vệ code (No protect), không chọn chế độ lập trình điện áp thấp (NOLVP: no low voltage program).

Hàng thứ 3 là khai báo tần số tụ thanh anh sử dụng là 20MHz.

Ba hàng trên thường là cố định cho tất cả các chương trình, khi viết chương trình khác thì nên copy từ chương trình đã có sang.

Hàng thứ 4 là khai báo chương trình chính.

Các lệnh tiếp theo là của chương trình chính đều nằm trong cặp dấu ngoặc nhọn.

Lệnh “SET_TRIS_B (0x00);” có chức năng định cấu hình cho portB là port xuất dữ liệu ra điều khiển led.

Lệnh WHILE(TRUE) là lệnh lặp có điều kiện và điều kiện cho ở đây luôn đúng và những gì trong vòng lặp while này sẽ thực hiện từ lệnh đầu tiên đến lệnh cuối cùng rồi lặp lại cho đến khi nào mất điện thì thôi – nên thường gọi là vòng lặp vô tận.

Trong vòng lặp while này có 4 lệnh: lệnh “OUTPUT_B(0x00);” là gán portB bằng 00H – kết quả lệnh này làm 8 ngõ ra của portB bằng 0, do led tích cực mức 0 và chỉ có 4 led nên 4 led sáng.

Lệnh tiếp theo là delay 1 giây để ta nhìn thấy led sáng.

Lệnh “OUTPUT_B(0x3C);” là gán portB bằng 3CH – kết quả lệnh này làm 4 ngõ ra của portB bằng 1 làm led tắt.

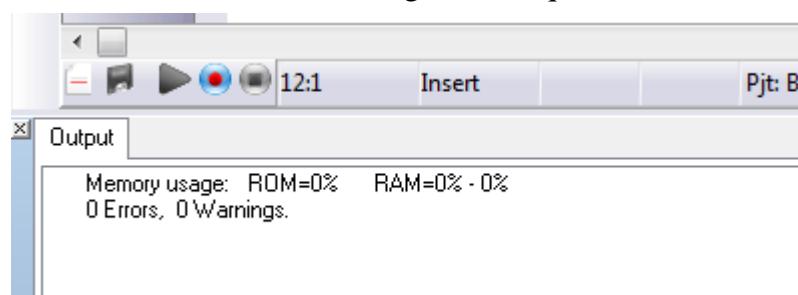
Lệnh tiếp theo là delay 1 giây để ta nhìn thấy led tắt.

Vòng lặp while đã thực hiện xong hết các lệnh và bây giờ sẽ quay lại thực hiện tiếp.

Ngôn ngữ lập trình CCS không phân biệt chữ hoa hay chữ thường.

Tiến hành biên dịch bằng cách chọn tab “Compile” (số 2) và chọn “Build All” (số 3) – xem hình 2-7 hoặc nhấn phím F9.

Sau khi biên dịch sẽ xuất hiện của sổ thông báo kết quả biên dịch như hình 2-8.



Hình 2-8. Màn hình thông báo kết quả biên dịch thành công.

Do lập trình đúng nên khi biên dịch thông báo “0 error ...” và cho biết phần trăm bộ nhớ đã sử dụng.

Nếu sai cú pháp thì biên dịch không thành, phải hiệu chỉnh cho hết lỗi.

Trong thông báo còn cho biết 2 thông tin dung lượng bộ nhớ ROM và RAM đã sử dụng.

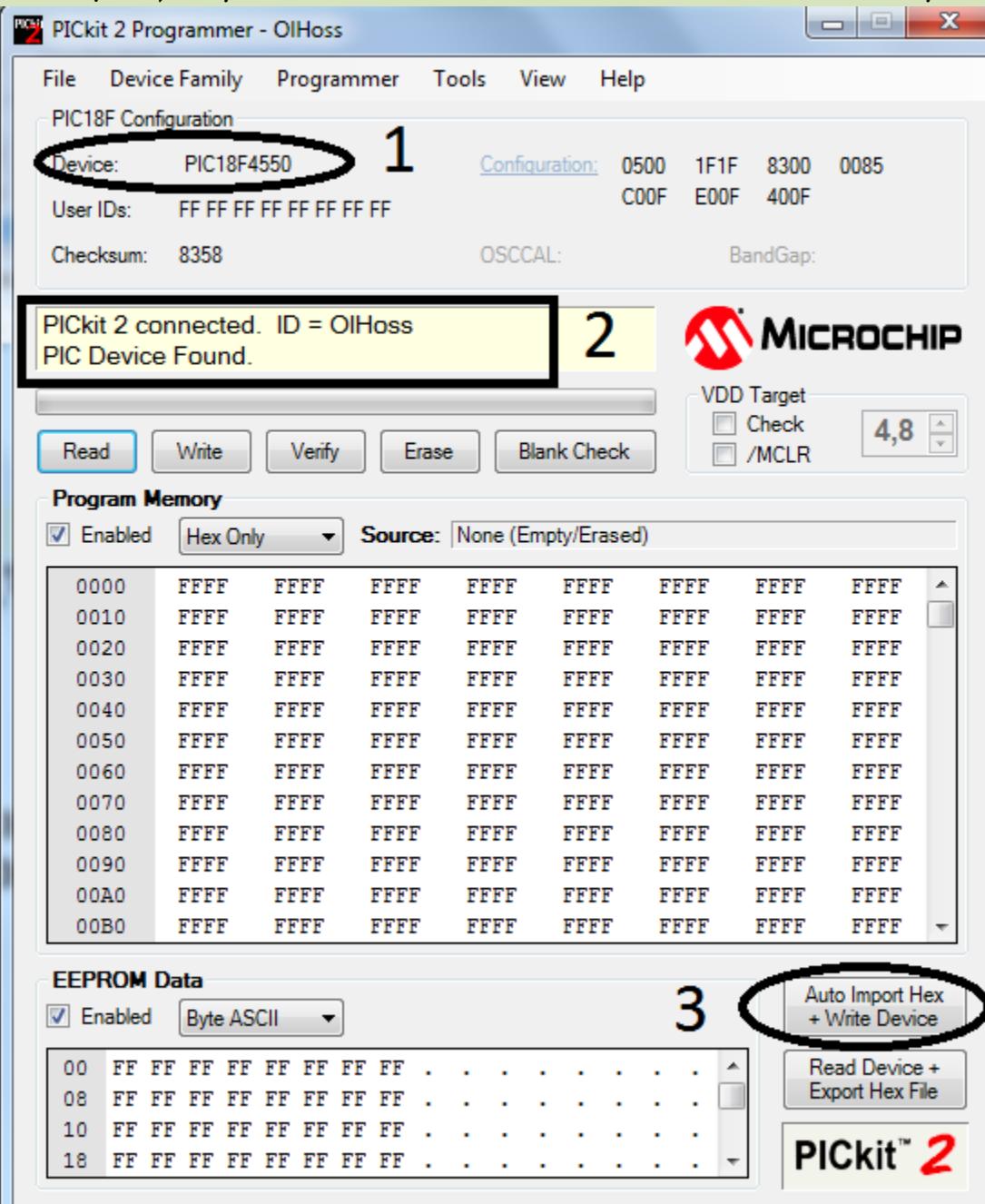
Có thể tắt màn hình thông báo bằng cách nhấn phím “ESC”.

Sau khi biên dịch xong thì sẽ được file chứa các số hex để nạp vào bộ nhớ của vi điều khiển.

Bước theo sau sẽ hướng dẫn nạp file hex vào bộ nhớ của vi điều khiển.

2.2.2 Nạp chương trình vào bộ nhớ vi điều khiển

Bước 1: Mở chương trình: Kit sử dụng mạch nạp của “PICKIT2”, tiến hành cài PICKIT2, kết nối cáp USB bộ thực hành vi điều khiển với máy tính và mở chương trình PICKIT2, sau khi kết nối thành công thì trên màn hình sẽ hiển thị loại CPU đang sử dụng và thông báo kết nối thành công theo trình tự 1, 2 như trong hình 2-9.

**Hình 2-9. Màn hình phần mềm nạp PICKIT2.**

Bước 2: Mở file hex để nạp: sau khi chọn xong thiết bị thì nhập chuột vào nút nhấn ở vị trí số 3 trong hình 2-9.

Một màn hình mới xuất hiện và bạn hãy chọn đường dẫn chứa file hex của chương trình biên soạn ở trên rồi chọn file cần nạp và bấm open thì phần mềm sẽ tiến hành nạp.

Sau khi nạp xong sẽ thông báo và vi điều khiển sẽ chạy đúng chương trình điều khiển 4 led chớp tắt.

Chương 3: THỰC HÀNH

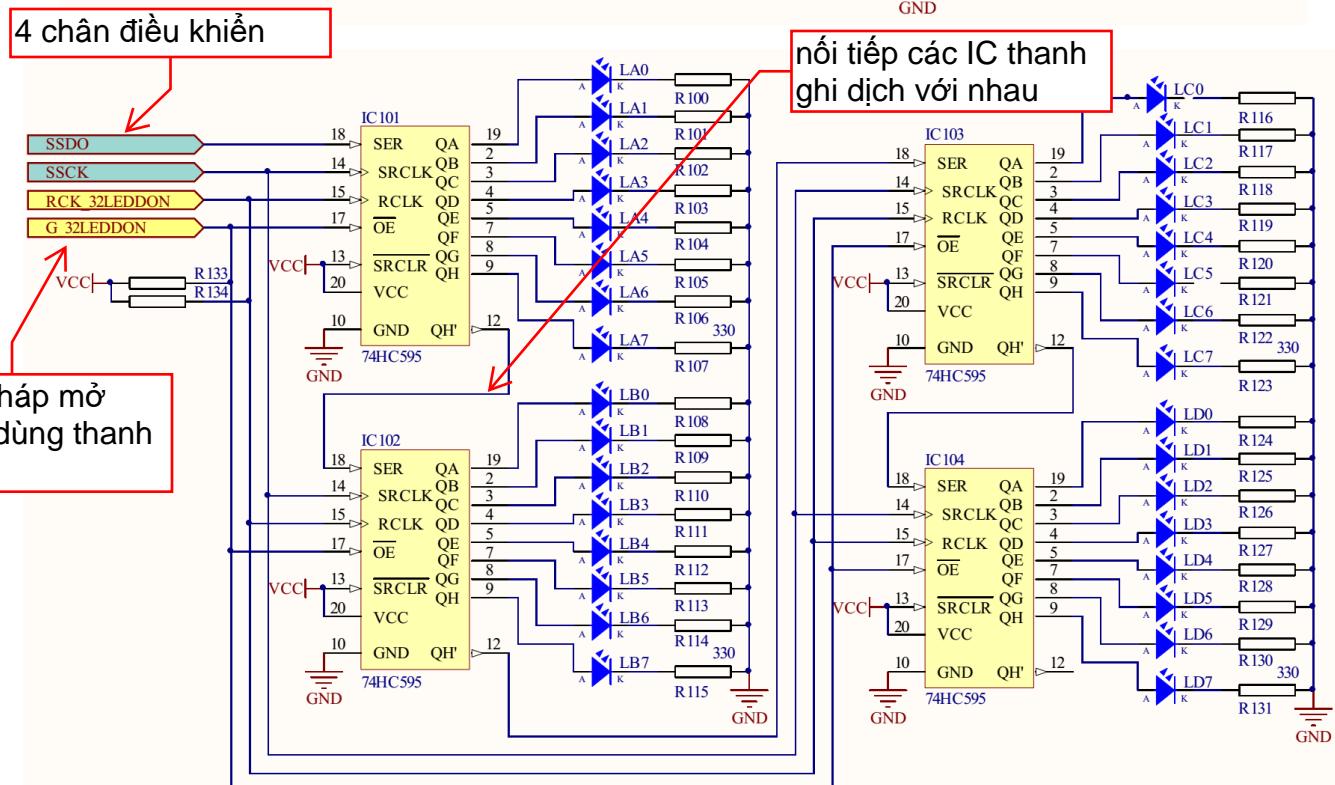
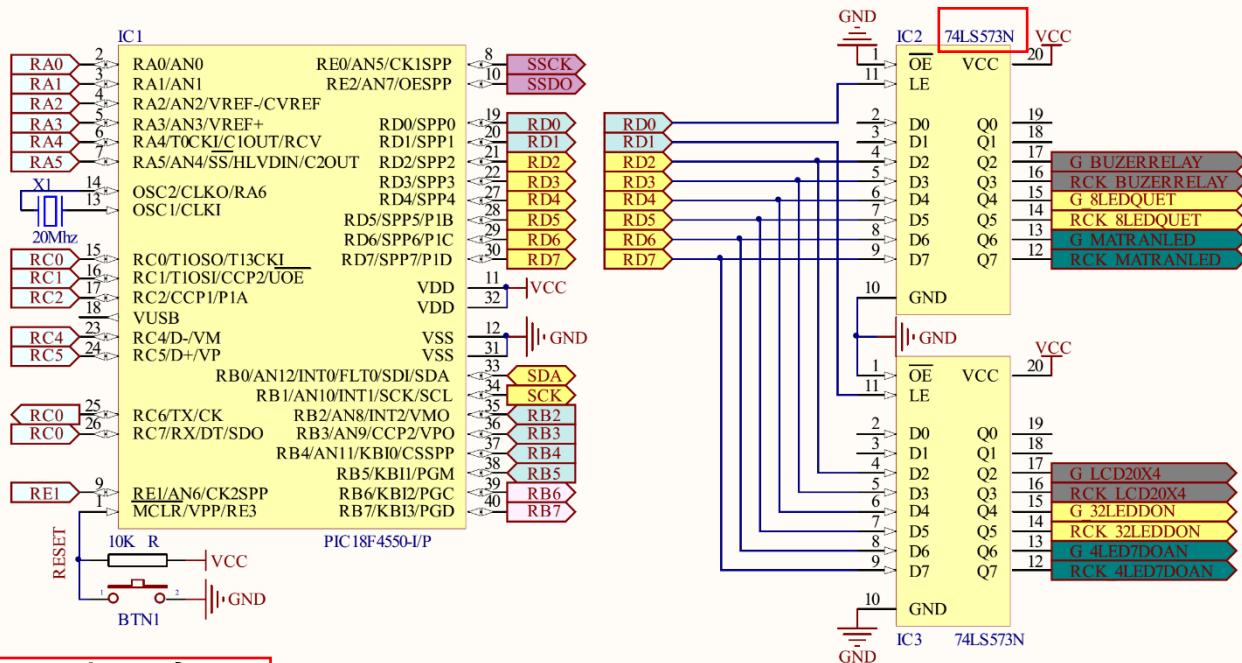
MODULE 1- 32 LED ĐƠN DÙNG THANH GHI DỊCH 74HC595, NÚT NHẤN, BÀN PHÍM MA TRẬN

3.1 CÁC CHƯƠNG TRÌNH ĐIỀU KHIỂN 32 LED ĐƠN

3.1.1 Mạch giao tiếp vi điều khiển với module 32 led đơn

Phần này thực hành các bài giao tiếp với 32 led đơn từ các bài đơn giản đến các bài phức tạp và tổng hợp nhiều bài lại với nhau.

Mạch điện giao tiếp vi điều khiển với module 32 led đơn như hình 3-1.



Hình 3-1. Sơ đồ nguyên lý giao tiếp vi điều khiển với module 32 led đơn.

Trong mạch sử dụng 4 IC thanh ghi 74HC595 để mở rộng port điều khiển 32 led đơn.

Phản vi điều khiển PIC giao tiếp với IC thanh ghi dịch

Có 4 tín hiệu điều khiển là:

- SSDO dùng để dịch dữ liệu nối tiếp.
- SSCK dùng để cấp xung clock.
- RCK_32LEDDON dùng điều khiển nạp dữ liệu song song từ bên trong IC thanh ghi dịch ra bên ngoài để điều khiển led đơn.
- G_32LEDDON dùng để cho phép xuất dữ liệu.

Phản IC thanh ghi dịch giao tiếp với 32 led

Mỗi IC thanh ghi dịch 74HC595 có 8 bit kết nối với 8 led, 4 IC điều khiển được 32 led.

Số lượng byte dữ liệu điều khiển module 32 led là 4 byte.

Khả năng mở rộng

Trong kit thực hành dùng 4 IC để điều khiển 32 led, sau này các ứng dụng thực tế cần nhiều led hơn thì bạn kết nối thêm IC thanh ghi nối tiếp vào một cách dễ dàng.

Chú ý nếu nhiều IC thanh ghi dịch thì phải tính toán fan-out cho phù hợp.

3.1.2 Các hàm điều khiển module 32 led đơn

Do sử dụng IC chốt để rộng thêm tín hiệu điều khiển các khói và các thanh ghi để mở rộng điều khiển các đối tượng led đơn, led 7 đoạn, LCD, nên chương trình điều khiển sẽ phức tạp hơn so với phương pháp dùng port điều khiển trực tiếp.

Để giúp các bạn hiểu được hệ thống thì ta cần khảo sát các hàm đã viết trong thư viện. Nên xem thêm các định nghĩa trong thư viện.

Hàm thứ 101: khởi tạo port và IC chốt:

```
VOID SET_UP_PORT_IC_CHOT()
{
    SET_TRIS_D(0x00);
    SET_TRIS_E(0x00);
    SET_TRIS_A(0x10);
    OUTPUT_D(0xFF);
    TIN_HIEU_DK_74573_A=0xFF;
    TIN_HIEU_DK_74573_B=0xFF;
    CHOT_IC_74573_A_GOI_DU_LIEU;
    CHOT_IC_74573_B_GOI_DU_LIEU;
    RBDC=0;
}
```

Khi bắt đầu lập trình thì tại chương trình chính bạn phải gọi hàm khởi tạo các port điều khiển bao gồm định hướng vào ra cho các port, khởi tạo dữ liệu cho các IC chốt, gửi dữ liệu ra các IC chốt để cảm nhận cả các module không được phép xuất dữ liệu. Tắt relay, triac, buzzer và các động cơ.

Trong chương trình chính ngay từ bắt đầu phải luôn gọi hàm này để định cấu hình.

Hàm thứ 102: là hàm xuất dữ liệu từng byte ra các thanh ghi dịch như sau:

```
void xuat_1byte(unsigned int8 x)
{
    unsigned int8 sb;
    #bit msb = x.7
    for (sb=0;sb<8;sb++)
    {
        output_bit(ssdo,msb);
        output_low(ssck);
        output_high(ssck);
        x= x<<1;
    }
}
```

Dịch bit cao đầu tiên

Khi gọi hàm này thì truyền tham số 1 byte cho biến X, trong hàm có định nghĩa 1 bit MSB là bit thứ 7 của biến X.

Vòng lặp for sẽ thực hiện xuất dữ liệu của bit MSB ra bit SSDO.

Hai lệnh tiếp theo làm bit SSCK xuống mức logic 0 rồi lên lại mức 1 có nghĩa là đã tạo ra 1 xung ck.

Kết hợp cả hai đồng nghĩa với việc ta đã xuất được 1 bit dữ liệu ra các thanh ghi dịch và các thanh ghi dịch đều nhận được bit này.

Ta xoay dữ liệu X sang trái để đẩy bit thứ 6 lên thay cho bit thứ 7 và vòng lặp for này thực hiện lại và kết quả bit thứ 6 cũng đã dịch ra các thanh ghi dịch.

Tương tự cho đến khi vòng lặp for kết thúc thì ta đã dịch được 1 byte ra các thanh ghi.

Hàm thứ 103: là các hàm có liên quan:

#DEFINE	CHO_IC_74573_A_GOI_DU_LIEU	OUTPUT_HIGH(ENABLE_573A)
#DEFINE	CHOT_IC_74573_A_GOI_DU_LIEU	OUTPUT_LOW(ENABLE_573A)
#DEFINE	CHO_IC_74573_B_GOI_DU_LIEU	OUTPUT_HIGH(ENABLE_573B)
#DEFINE	CHOT_IC_74573_B_GOI_DU_LIEU	OUTPUT_LOW(ENABLE_573B)

Mỗi định nghĩa tương đương với 1 hàm (thực ra là 1 lệnh) – dùng câu định nghĩa để nói lên chức năng cho dễ hiểu.

Định nghĩa thứ 1 có chức năng mở chốt cho IC chốt 74HC573A (làm tín hiệu enable_573A lên mức 1) để cho phép các tín hiệu xuất ra điều khiển các thanh ghi có liên quan nhưng chưa xuất.

Định nghĩa thứ 2 có chức năng đóng chốt 74HC573A lại (làm tín hiệu enable_573A xuống mức 0) và không còn cho phép xuất tín hiệu.

Tương tự cho IC chốt còn lại.

#DEFINE	MO_32_LED_DON	G_32A=0;
#DEFINE	TAT_32_LED_DON	G_32A=1;

Hai định nghĩa này có chức năng cho phép và không cho phép chân G của module 32 led. Tương tự cho các module khác.

Hàm thứ 104: là các hàm xuất dữ liệu điều khiển ra IC chốt:

```
void mo_ic_74573_a_thong_dl()
{
    output_d(0x00);
    output_bit(g_32leddon,g_32a);
    output_bit(g_4led7doan,g_4a);
    output_bit(g_lcd20x4,g_lcda);
    cho_ic_74573_a_goi_du_lieu;
}

void mo_ic_74573_b_thong_dl()
{
    output_d(0x00);
    output_bit(g_matranled,g_mtb);
    output_bit(g_8ledquet,g_8b);
    output_bit(g_buzerelay,g_rbdcb);
    cho_ic_74573_b_goi_du_lieu;
}
```

Có 2 hàm cho 2 IC chốt.

Lệnh thứ nhất xuất dữ liệu 0x00 ra port D để đưa các tín hiệu điều khiển về mức 0 để không cho phép. Tiếp hành gởi các bit điều khiển cho phép G_x tương ứng ra các bit tương ứng của port D (chỉ có 3 bit cho 3 chân G). Gọi hàm cho phép thông dữ liệu.

Trong các hàm thì sau khi cho phép thông dữ liệu xong thì lập tức đóng chốt lại.

Các hàm trên là hàm dùng chung cho tất cả các module, tiếp theo là các hàm điều khiển module 32 led đơn.

Hàm thứ 301: là hàm xuất dữ liệu 4 byte ra module 32 led đơn:

```
void xuat_32led_don_4byte(unsigned int8 bld3,bld2,bld1,bld0)
{
    xuat_1byte(bld3);
    xuat_1byte(bld2);
    xuat_1byte(bld1);
    xuat_1byte(bld0);

    mo_ic_74573_a_thong_dl();
    output_high(rck_32leddon);
    output_low(rck_32leddon);
    mo_32_led_don;
    chot_ic_74573_a_goi_du_lieu;
}
```

Hàm này có chức năng xuất 4 byte dữ liệu ra module 32 led. Khi gọi hàm này thì phải truyền 4 tham số byte dữ liệu.

Sau đó chương trình này sẽ gọi hàm truyền từng byte ra thanh ghi dịch. Thực hiện cho hết 4 byte.

Tiến hành mở chốt để thông các tín hiệu (RCK_32LED, G_32LED), rồi tạo xung RCK_32LED để đẩy dữ liệu đã lưu trong thanh ghi dịch sang thanh ghi lưu trữ, kết hợp với việc đã cho phép (G_32LED) nên dữ liệu sẽ xuất ra ngoài điều khiển 32 led sáng tùy thuộc vào dữ liệu của 4 byte.

Tiến hành cho phép module 32 led đơn (điều khiển chân G_32LED).

Sau đó đóng chốt lại để không làm ảnh hưởng khi truy xuất các đối tượng khác.

Hàm thứ 302: là hàm xuất dữ liệu 2 word 16 bit ra module 32 led đơn:

```
void xuat_32led_don_2word(unsigned int16 wld1, unsigned int16 wld0)
{
    unsigned int8 b3,b2,b1,b0;
    b3 = wld1>>8;
    b2 = wld1;
    b1 = wld0>>8;
    b0 = wld0;
    xuat_32led_don_4byte(b3,b2,b1,b0);
}
```

Hàm này dùng để xuất 2 word dữ liệu 16 bit ra module 32 led đơn. Hai dữ liệu 16 bit được tách thành 4 byte rồi sau đó gọi hàm xuất 4 byte là xong. Mục đích giúp bạn viết chương trình cho nhanh và gọn.

Hàm thứ 303: là hàm xuất dữ liệu 1 double word 32 bit ra module 32 led đơn:

```
void xuat_32led_don_1dw(unsigned int32 dwld)
{
    unsigned int16 wd1,wd0;
    wd1 = dwld>>16;
    wd0 = dwld;
    xuat_32led_don_2word(wd1,wd0);
}
```

Hàm này dùng để xuất 1 double word dữ liệu 32 bit ra module 32 led đơn. Dữ liệu 32 bit được tách thành 2 word 16 bit rồi gọi hàm xuất 2 word, hàm xuất 2 word sẽ gọi hàm xuất 4 byte.

Hàm thứ 304: là hàm xuất dữ liệu 1 byte ra module 32 led đơn:

```
void xuat_32led_don_1byte(unsigned int8 b0)
{
    xuat_32led_don_4byte(0,0,0,b0);
}
```

Hàm này dùng để xuất 1 byte dữ liệu ra module 32 led đơn. Gọi hàm xuất 4 byte bao gồm byte cần xuất và 3 byte bằng 0.

Như đã nêu, các hàm phức tạp và nhiều hàm liên quan như những gì vừa trình bày, các phần đã nêu và giải thích để giúp bạn hiểu rõ hệ thống, nhưng các bạn yên tâm, khi viết chương trình thì rất đơn giản.

Kit thực hành này mới có khả năng đáp ứng được nhiều ý tưởng sáng tạo so với các kit thực hành trước đây bị hạn chế về phần cứng

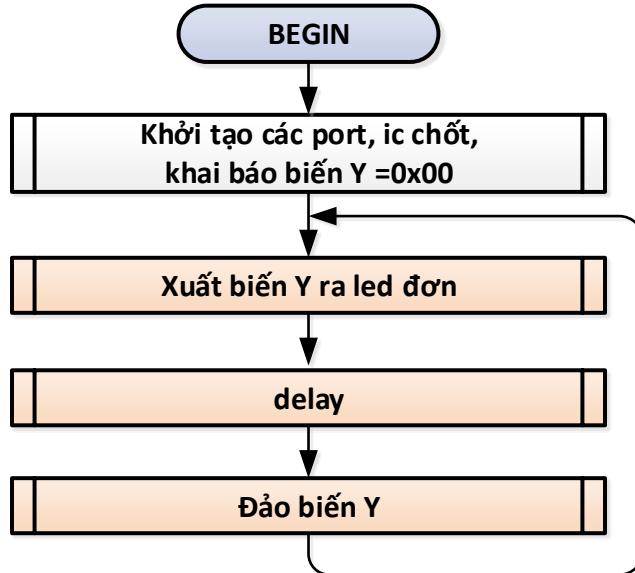
3.1.3 Các chương trình điều khiển 32 led đơn dùng lệnh for

Phần này thực hành các bài điều khiển led dùng vòng lặp for, các chương trình đơn giản dễ hiểu.

Bài mẫu 301. Chương trình điều khiển 8 LED chớp tắt.

Lưu tên file là “BAI_301_CHOPTAT_8LED”.

- Mục đích: biết cách viết chương trình điều khiển led đơn chớp tắt.
- Lưu đồ:



Hình 3- 2. Lưu đồ điều khiển 8 led chớp tắt.

- Giải thích lưu đồ:

Bắt đầu là khởi tạo các port và điều khiển IC chốt để không cho phép xuất dữ liệu ra các thiết bị ngoại vi. Khai báo và gán biến Y bằng 0x00 tương ứng số nhị phân 8 bit.

Tiến hành xuất dữ liệu của biến Y ra led đơn và dữ liệu 8 bit 0 sẽ làm 8 led tắt. Gọi hàm delay với thời gian tùy ý. Đảo biến Y đang là 8 bit 0 thì sẽ trở thành 8 bit 1 hay 0xFF, quay trở lại xuất dữ liệu của biến Y ra led đơn và làm 8 led sáng. Sau đó 3 công việc này được lặp đi lặp lại vô điều kiện nên trong lưu đồ không có kí hiệu kết thúc end.

- Chương trình:

```
#include <tv_pickit2_shift_1.c>
unsigned int8 y;
void main()
{
    set_up_port_ic_chot();
    y=0;
    while(true)
    {
        xuat_32led_don_4byte(0,0,0,y);
        delay_ms(200);
        y= ~y;
    }
}
```

copy các thư viện vào chung 1 thư mục chứa file chương trình đang làm

chớp tắt 5 lần
for (5) --> 10

tắt hết 3s
quay lại từ đầu

}

- e. Tiến hành biên dịch và nạp.
- f. Quan sát kết quả: nếu kết quả không đúng yêu cầu thì kiểm tra lại chương trình.
- g. Giải thích chương trình:

Hàng thứ 1: “`#include <tv_pickit2_shift_1.c>`”: khai báo thư viện viết sẵn chứa các khai báo: gồm khai báo thư viện vi điều khiển, khai báo ADC, khai báo cấu hình vi điều khiển, khai báo dao động, khai báo mã 7 đoạn và các chương trình con xuất dữ liệu ra các module.

Hàng thứ 2: “`unsigned int8 y;`”: khai báo biến Y là 8 bit không dấu.

Chương trình chính thực hiện lệnh:

Hàng lệnh “`set_up_port_ic_chot();`”: gọi hàm định cấu hình cho port D điều khiển 2 IC chốt và tiến hành chốt 2 IC. Như đã giải thích ở trên.

Lệnh “`y=0;`”: có chức năng gán dữ liệu cho biến y bằng 0 để điều khiển 8 led tắt.

Lệnh “`while(true)`”: có chức năng thực hiện các lệnh trong vòng lặp vô tận.

Lệnh “`xuat_32led_don_4byte(0,0,0,y);`”: có chức năng xuất 4 byte dữ liệu gồm byte dữ liệu y và 3 byte 0 ra 32 led, hàm này đã viết sẵn trong thư viện khai báo: “`tv_pickit2_shift_1.c`”.

Lệnh “`delay_ms(200);`”: có chức năng delay để nhìn thấy led sáng hoặc tắt.

Lệnh “`y = ~y;`”: có chức năng đảo dữ liệu của biến y từ 00 thành FF để điều khiển led sáng.

Vòng lặp while thực hiện lại từ đầu.

Tóm lại: chương trình rất đơn giản gồm khai báo thư viện, chương trình gọi hàm khởi tạo port, IC chốt, vòng lặp while gọi hàm xuất dữ liệu ra module 32 led, thay đổi dữ liệu điều khiển rồi lại xuất dữ liệu thay đổi ra và cứ thế lặp lại.

h. Câu hỏi:

Bài tập 302. Hãy hiệu chỉnh chương trình để điều khiển 16 LED chớp tắt.

Lưu tên file là “BAI_302_CHOPTAT_16LED”.

Bài tập 303. Hãy hiệu chỉnh chương trình để điều khiển 32 LED chớp tắt.

Lưu tên file là “BAI_303_CHOPTAT_32LED”.

Bài mẫu 304. Chương trình điều khiển 8 LED sáng dần và tắt dần phải sang trái.

Lưu tên file là “BAI_304_STD_PST_8LED”.

- a. Mục đích: biết cách viết chương trình điều khiển 8 led sáng dần và tắt dần phải sang trái.
- b. Lưu đồ: như hình 3-3.

c. Giải thích lưu đồ:

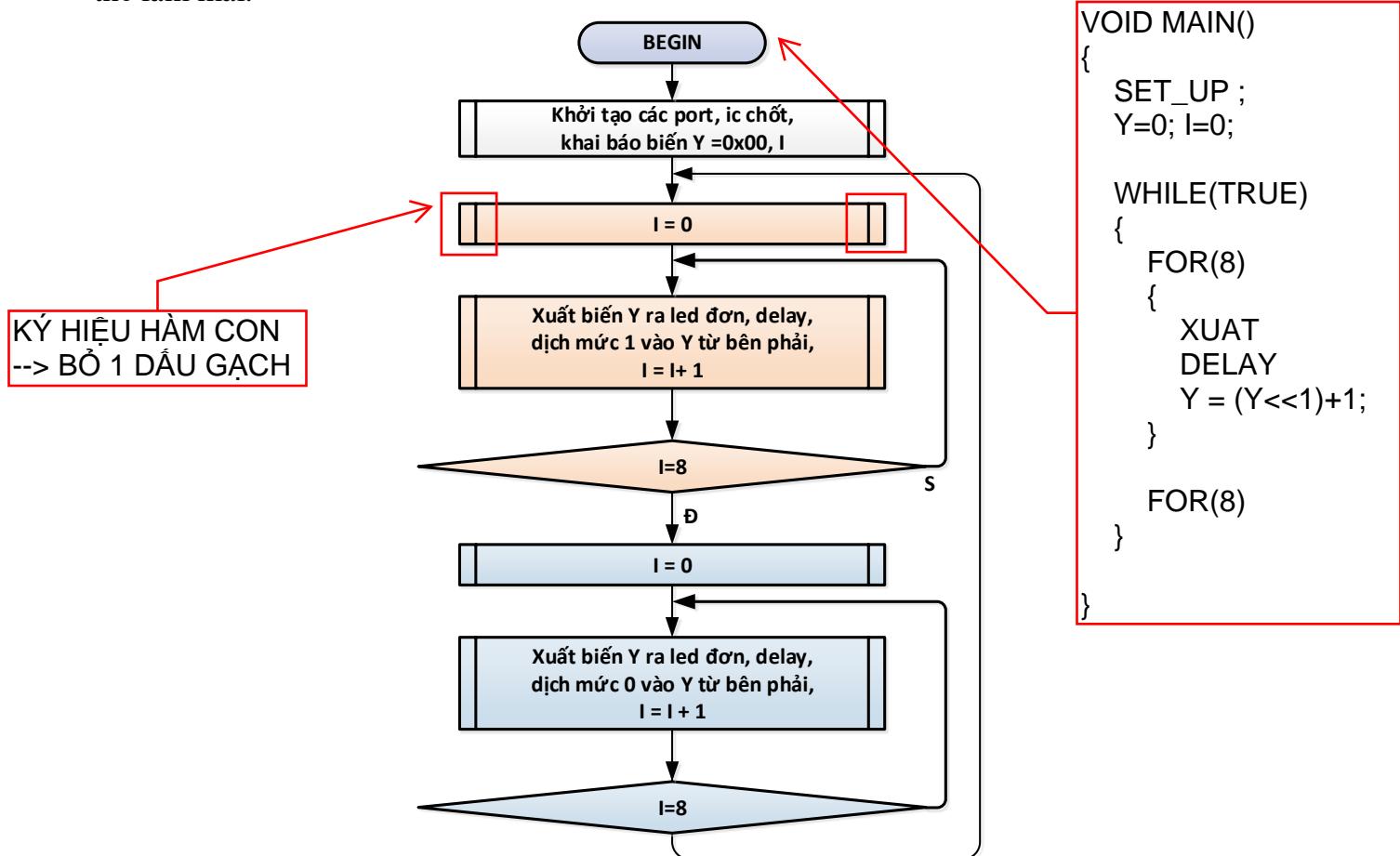
Bắt đầu là khởi tạo các port và điều khiển IC chót để không cho phép xuất dữ liệu ra các thiết bị ngoại vi. Khai báo và gán biến Y bằng 0x00 tương ứng số nhị phân 8 bit. Khai báo biến L.

Tiến hành xuất dữ liệu của biến Y ra led đơn và dữ liệu 8 bit 0 sẽ làm 8 led tắt, gọi hàm delay với thời gian tùy ý, dịch mức 1 vào bên phải của biến Y, tăng biến đếm I, kiểm tra xem bằng 8 hay chưa. Nếu chưa bằng quay trở lại xuất dữ liệu của biến Y ra led đơn sẽ làm 1 led sáng.

Tương tự như vậy sau 8 lần thì 8 led sáng dần lên từ phải sang trái và khi đó biến đèn I bằng 8 thì vòng lặp thứ nhất kết thúc.

Công việc cũng thực hiện tương tự như vòng lặp thứ 2 nhưng dữ liệu dịch vào là mức 0 sẽ làm 8 led tắt dần từ phải sang trái.

Sau 8 lần thì vòng lặp thứ 2 cũng kết thúc và quay trở lại vòng lặp thứ nhất làm lại và cứ thế làm mãi.



Hình 3- 3. Lưu đồ điều khiển 8 led sáng dần rồi tắt dần từ phải sang trái.

d. Chương trình:

```
#include <tv_pickit2_shift_1.c>
unsigned int8 y, i;
```

```

void main()
{
    set_up_port_ic_chot();
    y=0; // Tắt hết
    while(true)
    {
        Y=0;
        for(i=0;i<8;i++) // Sáng dần
        {
            xuat_32led_don_4byte(0,0,0,y);
            delay_ms(200);
            y= (y<<1)+1;
        }
        Y=0xFF;
        for(i=0;i<8;i++) // Tắt dần
        {
            xuat_32led_don_4byte(0,0,0,y);
            delay_ms(200);
            y= (y<<1);
        }
    }
}

```

- e. Tiến hành biên dịch và nạp.
f. Quan sát kết quả: nếu kết quả không
g. Giải thích chương trình: sinh viên

BT1:
SÁNG DÀN 8 LED T-P
SD 8 LED P-T

BT2:
SD 5 LED T-P
TAT DAN 7 LED P-T

Bài mẫu 305. Chương trình điều khiển 16 LED sáng dần và tắt dần phải sang trái.

Lưu tên file là “BAI_305_STD_PST_16LED”.

- a. Mục đích: biết cách mở rộng cho 16 led.
b. Lưu đồ: sinh viên hãy tự viết.
c. Chương trình:

```

#include <tv_pickit2_shift_1.c>
unsigned int16 y;
unsigned int8 i;

void main()
{
    set_up_port_ic_chot();
    y=0;
    while(true)
    {
        for(i=0;i<16;i++)
        {
            xuat_32led_don_2word(0,y); // không xuất 16 led giữa được --> xuat...4byte (0, y>>8, y,0);
            delay_ms(300); // Tách 16 bit --> 2 x 8 bit
            y= (y<<1)+1; // di chuyển các 8 bit cao về 8 bit thấp --> lấy, gán vào biến 8 bit
        }
    }
}

```

```

for(i=0;i<16;i++)
{
    xuat_32led_don_2word(0,y);
    delay_ms(300);
    y= (y<<1);
}

```

BAI 306: (Y,Y)

- d. Tiến hành biên dịch và nạp.
e. Quan sát kết quả: nếu kết quả không đúng yêu cầu thì kiểm tra lại chương trình.
f. Giải thích chương trình: **Chương trình có 2 vòng lặp for:**

- Vòng lặp for thứ nhất cho 16 led từ led thứ 0 đến led thứ 15 sáng dần.
- Vòng lặp for thứ hai cho 16 led từ led thứ 0 đến led thứ 15 tắt dần.

Trong chương trình này sử dụng 1 biến 16 bit để điều khiển 16 led sáng dần và tắt dần.

- g. Câu hỏi: hãy cho biết sự khác nhau của hàm xuất dữ liệu ra 32 led của bài 305 và bài 304.

Bài tập 306. Hãy viết chương trình điều khiển 32 LED chia làm 2 nhóm 16 led sáng dần và tắt dần từ trái sang phải cùng lúc.

Lưu tên file là “BAI_306_STD_TSP_16LED_2NHOM”

2 nhóm 16 led --> cùng số led (dùng chung vòng lặp for)
sáng giống nhau, cùng lúc --> xuất cùng 1 giá trị ra 2 bên --> 2word(y,y)

Bài mẫu 307. Chương trình điều khiển 32 LED sáng dần và tắt dần phải sang trái.

Lưu tên file là “BAI_307_STD_PST_32LED”.

- a. Mục đích: biết cách viết chương trình điều khiển 32 led sáng tắt dần.
b. Lưu đồ: sinh viên hãy tự viết.
c. Chương trình:

```

#include <tv_pickit2_shift_1.c>
unsigned int8 i;
unsigned int32 y;

void main()
{
    set_up_port_ic_chot();
    y=0;
    while(true)
    {
        for(i=0;i<32;i++)
        {
            xuat_32led_don_1dw(y);
            delay_ms(100);
            y= (y<<1)+1;
        }
        for(i=0;i<32;i++)
    }
}

```

```

    {
        xuat_32led_don_1dw(y);
        delay_ms(100);
        y= (y<<1);
    }
}

```

- d. Tiến hành biên dịch và nạp.
- e. Quan sát kết quả: nếu kết quả không đúng yêu cầu thì kiểm tra lại chương trình.
- f. Giải thích chương trình:
- g. Câu hỏi: hãy cho biết sự khác nhau của 4 hàm xuất dữ liệu ra 32 led của 4 bài.

Bài mẫu 308. Chương trình điều khiển 32 LED sáng dần và tắt dần phải sang trái rồi trái sang phải.

Lưu tên file là “BAI_308_SDTA_PST_TSP_32LED”.

- a. Mục đích: biết các viết chương trình điều khiển 32 led sáng tắt dần 2 chiều.
- b. Lưu đồ:
- c. Chương trình:

```

#include <tv_pickit2_shift_1.c>
unsigned int8      i;
unsigned int32     y;

void main()
{
    set_up_port_ic_chot();
    y=0;
    while(true)
    {
        for(i=0;i<32;i++)
        {
            xuat_32led_don_1dw(y);
            delay_ms(30);
            y= (y<<1)+1;
        }
        for(i=0;i<32;i++)
        {
            xuat_32led_don_1dw(y);
            delay_ms(30);
            y= (y<<1);
        }
        for(i=0;i<32;i++)
        {
            xuat_32led_don_1dw(y);
            delay_ms(30);
            y= (y>>1)+0x80000000;
        }
    }
}

```

giải thuật sáng dần --> bài giảng

```

        }
        for(i=0;i<32;i++)
        {
            xuat_32led_don_1dw(y) ;
            delay_ms(30) ;
            y= (y>>1) ;
        }
    }
}

```

- d. Tiến hành biên dịch và nạp.
- e. Quan sát kết quả: nếu kết quả không đúng yêu cầu thì kiểm tra lại chương trình.
- f. Giải thích chương trình: Chương trình có 4 vòng lặp for, vòng for thứ nhất điều khiển 32 led sáng dần từ phải sang trái, vòng for thứ 2 điều khiển 32 led tắt dần từ phải sang trái, vòng for thứ 3 điều khiển 32 led sáng dần từ trái sang phải, vòng for thứ 4 điều khiển 32 led tắt dần từ phải sang trái.

Chú ý: Khi dịch phải sang trái thì cộng với 0x01 hay 1 cũng giống nhau nhưng khi dịch từ trái sang phải thì nếu 8 bit ta sẽ cộng với dữ liệu nhị phân 8 bit là : 1000_0000 hay 0x80, nếu 16 bit thì cộng với 1000_0000_0000_0000 hay 0x8000, tương tự cho 32 bit là 0x80000000.

Bài mẫu 309. Chương trình điều khiển 32 LED tổng hợp nhiều chương trình gồm:

1. Điều khiển 32 LED chớp tắt 5 lần.
2. Điều khiển 32 LED sáng dần và tắt dần phải sang trái 2 lần.
3. Điều khiển 32 LED chớp tắt 5 lần.
4. Điều khiển 32 LED sáng dần và tắt dần trái sang phải 2 lần.

Lưu tên file là “BAI_309_TONGHOP_32LED”.

- a. Mục đích: biết cách viết chương trình tổng hợp từ các chương trình đã viết.
- b. Lưu đồ: sinh viên hãy tự viết.
- c. Chương trình:

```

#include <tv_pickit2_shift_1.c>
unsigned int8 i,j;
unsigned int32 y;

void sang_tat_32led_5lan()
{
    for(j=0;j<5;j++)
    {
        xuat_32led_don_4byte(0,0,0,0);
        delay_ms(100);
        xuat_32led_don_4byte(0xff,0xff,0xff,0xff);
        delay_ms(100);
    }
}

```

```

void sang_tat_dan_pst_32led_2lan()
{
    for(j=0;j<2;j++)
    {
        for(i=0;i<32;i++)
        {
            xuat_32led_don_1dw(y);
            delay_ms(30);
            y= (y<<1)+1;
        }
        for(i=0;i<32;i++)
        {
            xuat_32led_don_1dw(y);
            delay_ms(30);
            y= (y<<1);
        }
    }
}

```

các vòng for lồng vào nhau --> thì biến chứa số lần phải khác nhau,
nếu không lồng vào nhau thì có thể dùng chung

```

void sang_tat_dan_tsp_32led_2lan()
{
    for(j=0;j<2;j++)
    {
        for(i=0;i<32;i++)
        {
            xuat_32led_don_1dw(y);
            delay_ms(30);
            y= (y>>1)+0x80000000;
        }
        for(i=0;i<32;i++)
        {
            xuat_32led_don_1dw(y);
            delay_ms(30);
            y= (y>>1);
        }
    }
}

```

```

void main()
{
    set_up_port_ic_chot();
    y=0x00000000;
    while(true)
    {
        sang_tat_32led_5lan();
        sang_tat_dan_pst_32led_2lan();
        sang_tat_32led_5lan();
        sang_tat_dan_tsp_32led_2lan();
    }
}

```

viết hàm con cho ct chính gọn gàng

thực hiện nhiều lần bằng cách goi

- d. Tiến hành biên dịch và nạp.
 - e. Quan sát kết quả: nếu kết quả không đúng yêu cầu thì kiểm tra lại chương trình.
 - f. Giải thích chương trình:

Bài mẫu 310. Chương trình điều khiển 32 LED tổng hợp:

Thêm vào bài trên chương trình:

5. Điều khiển 32 LED chớp tắt 5 lần.
 6. Điều khiển 32 LED sáng dần và tắt dần từ ngoài vào 2 lần.
 7. Điều khiển 32 LED chớp tắt 5 lần.
 8. Điều khiển 32 LED sáng dần và tắt dần trừ trong ra 2 lần.

Lưu tên file là “BAI_310_TONGHOP_32LED”.

- a. Mục đích: biết cách mở rộng thêm các chương trình.
 - b. Lưu đồ: sinh viên hãy tự viết.
 - c. Chương trình: thêm vào dưới phần khai báo biến các chương trình con bên dưới:

```
unsigned int16 zt,zp;
void sang_tat_dan_ngoaivao_32led_21an()
{
    for(j=0;j<2;j++)
    {
        for(i=0;i<16;i++)
        {
            xuat_32led_don_2word(zp,zt);
            delay_ms(30);
            zp= (zp<<1)+1;
            zt= (zt>>1)+0x8000;
        }
        for(i=0;i<16;i++)
        {
            xuat_32led_don_2word(zp,zt);
            delay_ms(30);
            zp= (zp<<1);
            zt= (zt>>1);
        }
    }
}
```

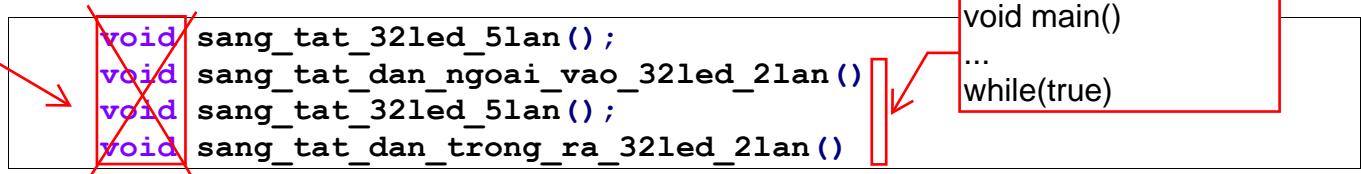
```
void sang_tat_dan_trong_ra_32led_2lan()
{
    for(j=0;j<2;j++)
    {
        for(i=0;i<16;i++)
        {
            xuat_32led_don_2word(zp,zt);
            delay_ms(30);
            zp= (zp>>1)+0x8000; zt= (zt<<1)+1;
        }
    }
}
```

```

        }
        for(i=0;i<16;i++)
        {
            xuat_32led_don_2word(zp,zt);
            delay_ms(30);
            zp= (zp>>1);
            zt= (zt<<1);
        }
    }
}

```

Ở vòng lặp while trong chương trình chính thì tiến hành đánh thêm vào 4 hàm, gọi các hàm con vừa viết như sau:



3.1.4 Các chương trình điều khiển 32 led đơn dùng lệnh if

Các chương trình dùng lệnh for thì không có vấn đề gì đối với từng chương trình hoặc nhiều chương trình điều khiển led đơn thực hiện một cách tuần tự, tuy nhiên khi hệ thống thực hiện nhiều chương trình với nhiều chức năng khác nhau với yêu cầu đáp ứng nhanh thì các chương trình điều khiển dùng vòng lặp for sẽ không đáp ứng được.

Ví dụ chương trình con điều khiển 32 led sáng tắt dần từ phải sang trái dùng vòng lặp for với thời gian delay là 1 giây thì để thực hiện xong 1 chu kỳ điều khiển của chương trình này phải mất hết 64 giây (32 giây cho sáng dần, 32 giây cho tắt dần). Trong lúc thực hiện chương trình này nếu ta có nhấn phím để ngừng chương trình này và chuyển sang chương trình khác thì cũng vô tác dụng vì chương trình chưa kết thúc. Nếu dùng nút nhấn để chuyển chương trình thì phải nhấn và giữ trong thời gian 64 giây thì mới có khả năng đáp ứng được.

Một ví dụ khác là vừa điều khiển 32 sáng tắt dần với thời gian trễ 1 giây vừa đếm sản phẩm hiển thị trên 4 led 7 đoạn: với cách viết dùng vòng lặp for thì phải đợi xong 1 chu kỳ điều khiển 32 led thì mới cập nhật kết quả đếm 1 lần – trong lúc đang điều khiển 32 led nếu có sản phẩm thì counter cũng đếm nhưng không hiển thị ra led được.

Vậy làm sao để thực hiện nhiều chương trình với đáp ứng nhanh và liên tục?

Đáp ứng nhanh và liên tục có nghĩa là giả sử đang điều khiển led thứ 3 sáng dần mà ta nhấn phím thì nó phát hiện và xử lý ngay hoặc khi phát hiện có xung thì nó cập nhật kết quả ngay và sau đó tiếp tục điều khiển led thứ 4 sáng dần ...

Để thực hiện được điều này thì có nhiều cách:

- Cách thứ 1 là vẫn dùng vòng lặp for nhưng can thiệp vào chương trình delay để kiểm tra phím nhấn, xung đếm của counter, ... Cách này không hay.
- Cách thứ 2 là viết không dùng vòng lặp. Các chương trình bây giờ viết theo cách khác và mỗi lần gọi thì phải kiểm tra đang thực hiện đến đâu và làm tiếp công việc và chỉ thực hiện 1 công việc đơn giản rồi thoát ngay.

Ví dụ chương trình điều khiển 32 led sáng dần và tắt dần viết theo cách này thì viết như thế nào?

Với chương trình này thì chu kỳ thực hiện ta biết là 64 lần: 32 lần cho led sáng dần và 32 lần cho led tắt dần, phải dùng 1 biến đếm để đếm số lần thực hiện, dữ liệu cho chương trình này là độc lập.

Mỗi khi chương trình này được gọi thì nó sẽ kiểm tra xem biến đếm nếu nhỏ hơn 32 thì đang ở phần làm led sáng dần thì ta tiến hành xuất dữ liệu ra module 32 led và dịch mức 1 vào rồi delay theo ý muốn, tăng biến đếm lên 1 và sau đó thoát.

Giả sử thời gian delay vẫn là 1 giây thì chương trình này khi được gọi thì sau 1 giây sẽ thoát chứ không phải chờ đến 64 giây như cách dùng vòng lặp for.

Cứ thế sau 32 lần gọi, 32 led đã sáng dần hết rồi thì bây giờ kiểm tra xem biến đếm nhỏ hơn 64 hay không, nếu đúng thì làm tắt dần 1 led, delay, tăng biến đếm rồi thoát.

Cứ thế cho đến khi biến đếm bằng 64 thì xong 1 chu kỳ thì ta cho biến đếm bằng 0 lại để bắt đầu chu kỳ tiếp theo.

Tóm lại chương trình có n công việc theo trình tự thời gian thì ta viết sao cho mỗi lần gọi thì chỉ thực hiện 1 công việc, có biến giám sát công việc đang thực hiện.

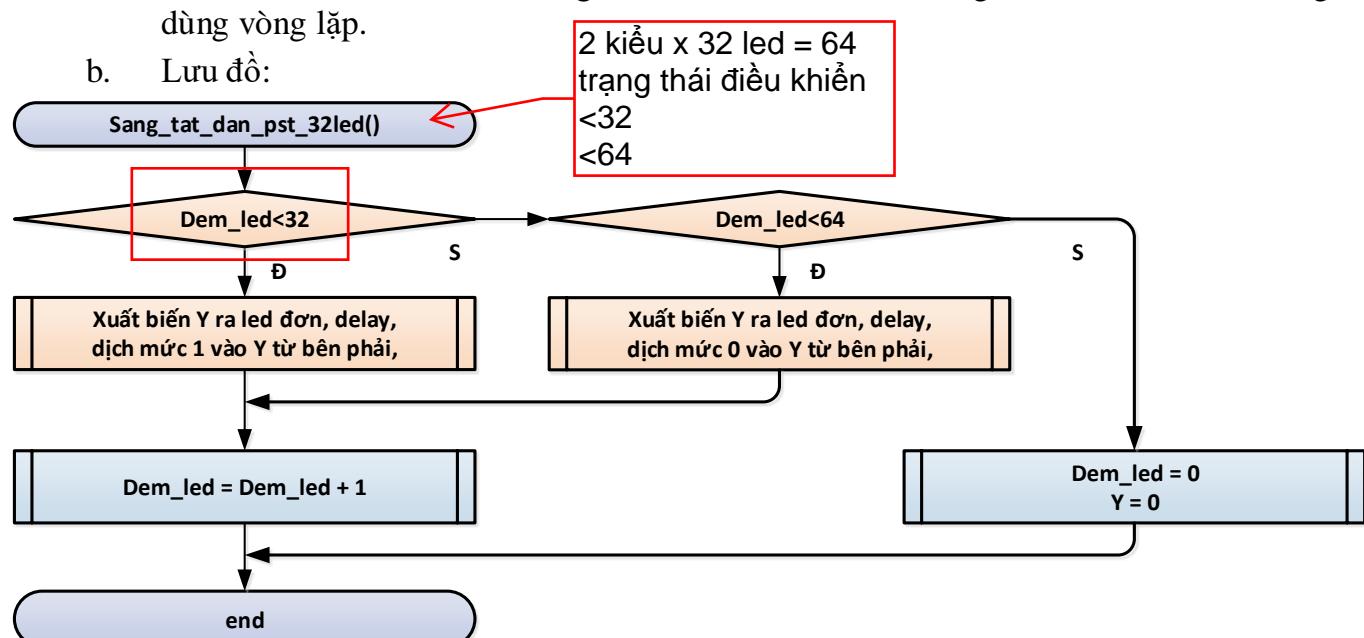
Với cách viết này thì mới có khả năng đáp ứng được một hệ thống điều khiển lớn được.

Bài mẫu 311. Hãy viết chương trình điều khiển 32 led sáng dần tắt dần từ phải sang trái viết không dùng vòng lặp, chỉ dùng lệnh if.

Lưu tên file là “BAI_311_STD_PST_32LED_NO_LOOP”.

a. Mục đích: biết các viết chương trình điều khiển 32 led sáng tắt dần theo cách không dùng vòng lặp.

b. Lưu đồ:



Hình 3-4. Lưu đồ chương trình con điều khiển 32 led sáng tắt dần từ phải sang trái cho lệnh if.

c. Giải thích lưu đồ:

Lưu đồ này chỉ trình bày cho chương trình con, không viết lưu đồ cho chương trình chính. Chương trình chính chỉ cần gọi chương trình con này là xong.

Khi gọi chương trình con này thì kiểm tra biến dem_led nếu nhỏ hơn 32 có nghĩa là còn nằm trong vòng lặp làm 32 led sáng dần lên và như vậy tiếp tục làm 32 led sáng dần lên thêm, mỗi lần thêm 1 led thì tăng biến dem_led.

Đến khi sáng đủ 32 led thì điều kiện nhỏ hơn 32 sẽ sai vì biến dem_led bây giờ bằng 32 và sẽ chuyển sang kiểm tra điều kiện nhỏ hơn 64, điều kiện này đúng thì khi đó sẽ thực hiện công việc tương tự nhưng làm 32 led tắt dần, cho đến khi tắt hết thì biến dem_led bằng 64 và khi đó sẽ reset các giá trị về lại từ đầu để bắt đầu một chu kỳ mới.

Mỗi lần xử lý xong 1 led sáng dần hoặc tắt dần thì chương trình sẽ kết thúc, trở lại chương trình chính để làm các công việc khác.

d. Chương trình:

```
#include <tv_pickit2_shift_1.c>
unsigned int8 dem_led;
unsigned int32 y;
void delay_tuy_y(unsigned int8 dl2)
{
    unsigned int8 dl;
    for (dl=0;dl<dl2;dl++) delay_ms(10);
}
void sang_tat_dan_pst_32led(unsigned int8 dl)
{
    if (dem_led<32) // 0-31: 32 trạng thái sáng dần
    {
        xuat_32led_don_1dw(y);
        y= (y<<1)+1;
        delay_tuy_y(dl);
    }
    else if (dem_led<64) // if((dem_led>=32) && (dem_led<64))
    {
        xuat_32led_don_1dw(y);
        y= (y<<1);
        delay_tuy_y(dl);
    }
    else // 32-63: 32 tt tắt dần
    {
        dem_led = 0;
        y = 0;
    }
    dem_led++;
}
void main()
{
    set_up_port_ic_chot();
    y=0; // dem_led=1;
    while(true)
}
```

```

{
    sang_tat_dan_pst_32led(2);
    //cac yeu cau dieu khien khac
}

```

- e. Tiến hành biên dịch và nạp.
- f. Quan sát kết quả: nếu kết quả không đúng yêu cầu thì kiểm tra lại chương trình.
- g. Giải thích chương trình: trong vòng lặp while ta gọi chương trình sáng tắt dần phải sang trái 32 led và truyền tham số delay là 2 cho chương trình con.
Chương trình con kiểm tra biến đếm led nếu nhỏ hơn 32 thì tiến hành: xuất dữ liệu ra 32 led, dịch bit 1 vào, delay, tăng biến đếm led lên 1 rồi thoát.
Khi biến đếm led lớn hơn 32 và nhỏ hơn 64 thì điều khiển led tắt dần: xuất dữ liệu ra 32 led, dịch mức 0 vào, delay và tăng biến đếm led rồi thoát.
Cho đến khi bằng 64 thì reset lại biến đếm led và Y để bắt đầu lại chu kỳ mới.
Truyền tham số delay giúp ta linh động hơn do có nhiều chương trình cần delay dài hơn thì tham số lớn hơn, ví dụ chương trình chớp tắt thì phải delay lớn chứ delay nhỏ ta không nhìn thấy được.
Sau khi gọi chương trình điều khiển 32 led sáng tắt dần thì chương trình chỉ xử lý 1 led và thoát trở lại ngay chương trình chính và trong chương trình chính ta có thể thực hiện các chương trình điều khiển khác thì nó đáp ứng ngay, ... Sau khi hết các yêu cầu điều khiển trong vòng lặp while thì vòng lặp while sẽ quay lại từ đầu. Nếu gọi lại chương trình điều khiển 32 led sáng tắt dần thì chương trình điều khiển 32 led sẽ thực hiện tiếp và cứ thế, ...

Trong bài này chưa có yêu cầu điều khiển khác, chủ yếu là chỉ cho bạn cách viết. Sau này khi học thêm nhiều module khác thì các yêu cầu điều khiển sẽ được đưa vào.

Bài tập 312. Hãy viết chương trình điều khiển 32 led sáng dần tắt dần từ phải sang trái, trái sang phải dùng lệnh if – không dùng lệnh for.
Lưu tên file là “BAI_312_STD_PST_TSP_32LED”.

Bài tập 313. Hãy viết lại bài 310 chỉ dùng lệnh if – không dùng lệnh for.
Lưu tên file là “BAI_314_TONGHOP_32LED”.

3.2 CÁC CHƯƠNG TRÌNH ĐIỀU KHIỂN 32 LED ĐƠN VỊ

3.2.1 Giới thiệu

Nút nhấn, bàn phím dùng để giao tiếp giữa con người và mạch điện tử để điều khiển, ví dụ: bàn phím máy tính, bàn phím điện thoại, bàn phím máy bán xăng dầu dùng nhập số tiền cần bán, số lít cần bán ... máy giặt tự động có bàn phím để chỉnh chế độ giặt, chọn mức nước ...

Có 2 dạng giao tiếp vi điều khiển với bàn phím, nút nhấn:

- Hệ thống ít phím: ví dụ điều khiển động cơ bằng 3 phím: Start, Stop, Inv, đồng hồ có 3 đến 4 phím để chỉnh thời gian.
- Hệ thống nhiều phím: bàn phím máy tính, bàn phím điện thoại, ...

3.2.2 Mạch điện giao tiếp vi điều khiển với 4 nút nhấn đơn

Phần này thực hành các bài giao tiếp vi điều khiển với 4 nút nhấn đơn để thực hiện các yêu cầu điều khiển như điều khiển led tắt mở, điều khiển đảo chiều led và các ứng dụng sau này đều có liên quan đến nút nhấn.

Mạch điện giao tiếp vi điều khiển với các nút nhấn như hình 3-5.

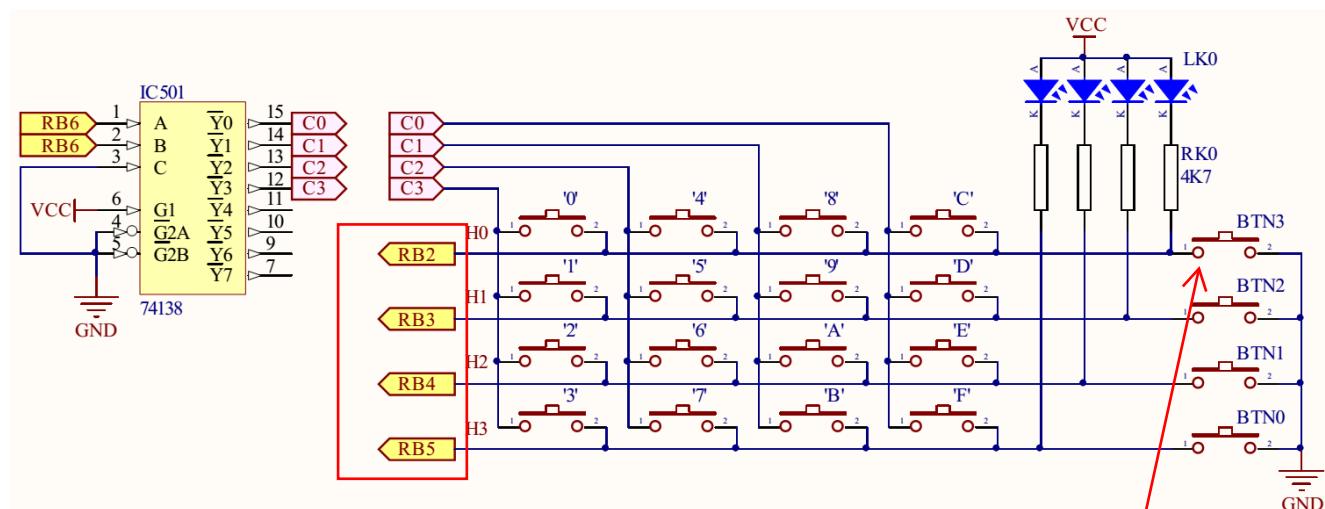
Vì điều khiển dùng 4 bit của portB giao tiếp với 4 nút nhấn đơn, các nút nhấn còn lại của bàn phím ma trận thì chưa cần quan tâm đến.

Các nút nhấn bình thường hở nên trạng thái tín hiệu ở mức logic ‘1’, khi nhấn thì ngắn mạch nối mass làm mức logic về ‘0’. Khi nhả phím thì về lại bình thường mức logic ‘1’.

Để kiểm tra có nhấn hay không thì ta kiểm tra trạng thái nếu bằng ‘1’ thì không nhấn phím, bằng ‘0’ thì có nhấn.

Trong mạch có 4 đèn báo hiệu : bình thường thì 4 đèn tắt, khi nhấn 1 phím thì đèn tương ứng sáng.

Khi đèn sáng thì chắc chắn phím nhấn còn tốt. Nhiều bạn lập trình sai thường đỗ lỗi cho nút nhấn hỏng.



Hình 3-5. Sơ đồ nguyên lý giao tiếp vi điều khiển với 4 nút nhấn đơn.

3.2.3 Định nghĩa tên các nút nhấn trong thư viện

Kit thực hành có 4 nút nhấn đơn có tên là BT0 đến BT3 portB.

Nhấn thì vi điều khiển nhận **mức 0**
(led sáng)

Không nhấn: mức 1

Trong thư viện ta có định nghĩa sẵn nhiều tên khác nhau cho các nút nhấn để phù hợp với các bài thực hành, ví dụ ta nhấn ON thì đèn sáng, nhấn nút OFF thì đèn tắt hay hơn là nhấn BT0 đèn sáng và BT1 thì đèn tắt. Khi viết hàm ta đặt tên hàm là PHIM_ON và PHIM_OFF nó rõ ràng hơn là PHIM_BT0 và PHIM_BT1. Khi thấy hàm PHIM_ON nó gợi cho ta chức năng là

mở ngay, còn khi thấy hàm PHIM_BT0 thì ta phải đọc nội dung mới biết nó làm gì khi mà ta viết nhiều phím và chương trình đã viết lâu rồi nay đọc lại sẽ mất nhiều thời gian.

Sau đây là các định nghĩa về 4 nút nhấn:

#DEFINE BT0 PIN_B5	PIN_B2	Định nghĩa tên cho nút nhấn
#DEFINE BT1 PIN_B4		Có sẵn trong thư viện TV_PICKIT2...
#DEFINE BT2 PIN_B3		
#DEFINE BT3 PIN_B2		
#DEFINE ON BT0		Nếu muốn đặt tên khác: viết trong chương trình của mình
#DEFINE OFF BT1		
#DEFINE INV BT2		
#DEFINE UP BT0		#define SS PIN_B5
#DEFINE MOD BT1		hoặc
#DEFINE DW BT2		#define SS BT0
#DEFINE CLR BT3		
#DEFINE STOP BT3		Thứ tự phím trên bộ TN có thể đảo ngược so với khai báo thì ta có thể thay đổi thư viện hoặc nhấn phím theo thực tế
#DEFINE ON1 BT0		
#DEFINE OFF1 BT1		
#DEFINE ON2 BT2		
#DEFINE OFF2 BT3		

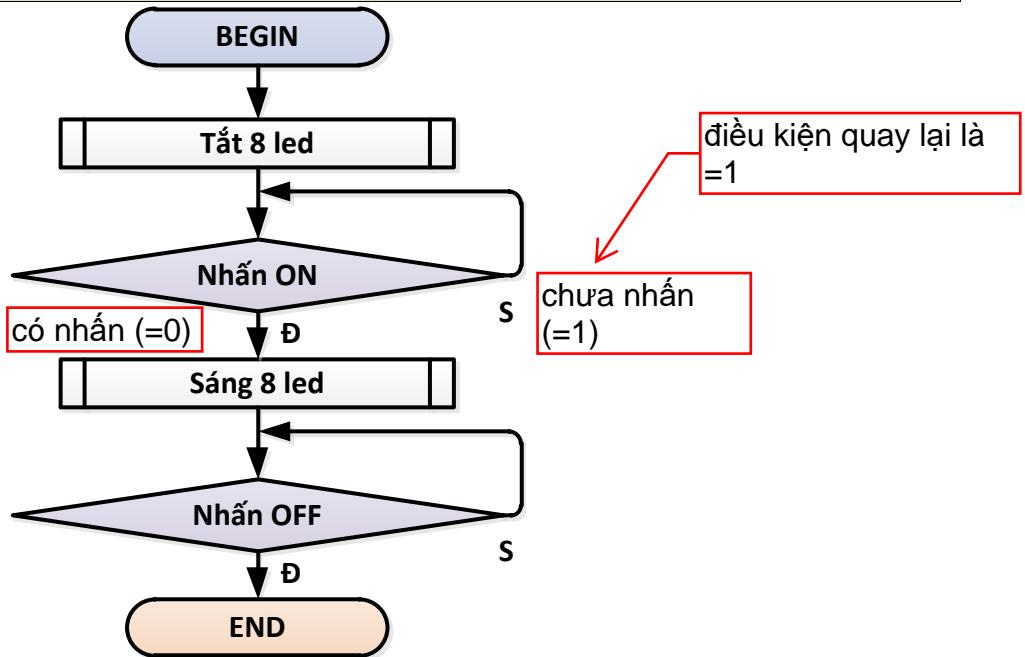
Việc định nghĩa này giúp ta thống nhất và khi viết chương trình không phải định nghĩa nữa.

3.2.4 Các chương trình điều khiển dùng các nút nhấn đơn

Bài mẫu 321. Chương trình điều khiển 8 LED bằng 2 nút ON và OFF. Khi có điện thì led tắt, khi nhấn ON thì 8 led sáng, khi nhấn OFF thì 8 led tắt.

Lưu tên file là “BAI_321_ON_OFF_8LED”

- Mục đích : Biết cách lập trình kiểm tra nút nhấn và điều khiển.
- Lưu đồ:

**Hình 3-6. Lưu đồ điều khiển 8 led sá**

Thiết lập port B có 4 ngõ vào (=1) là RB2 - RB5

(theo phần cứng trang 52)

những ngõ còn lại thì tùy theo phần cứng, dùng làm ngõ ra (=0)

$$\begin{array}{ll} 0011 & 1100 = 0x3C \\ 8421 & 8421 \end{array}$$

c. Chương trình:

```

#include <tv_pickit2_shift_1.c>
void main()
{
    set_up_port_ic_chot();
    set_tris_b(0x3C);
    xuat_32led_don_4byte(0,0,0,0);
    while(true)
    {
        while(input(bt0));
        xuat_32led_don_4byte(0,0,0,0xff);

        while(input(bt1));
        xuat_32led_don_4byte(0,0,0,0);
    }
}

```

while(input(bt0) == 1) {}

Vòng lặp kiểm tra trạng thái nút nhấn,

nếu chưa nhấn (=1) thì quay lại kiểm tra tiếp cho đến khi nhấn (=0) thì thoát vòng lặp

d. Tiến hành biên dịch và nạp.

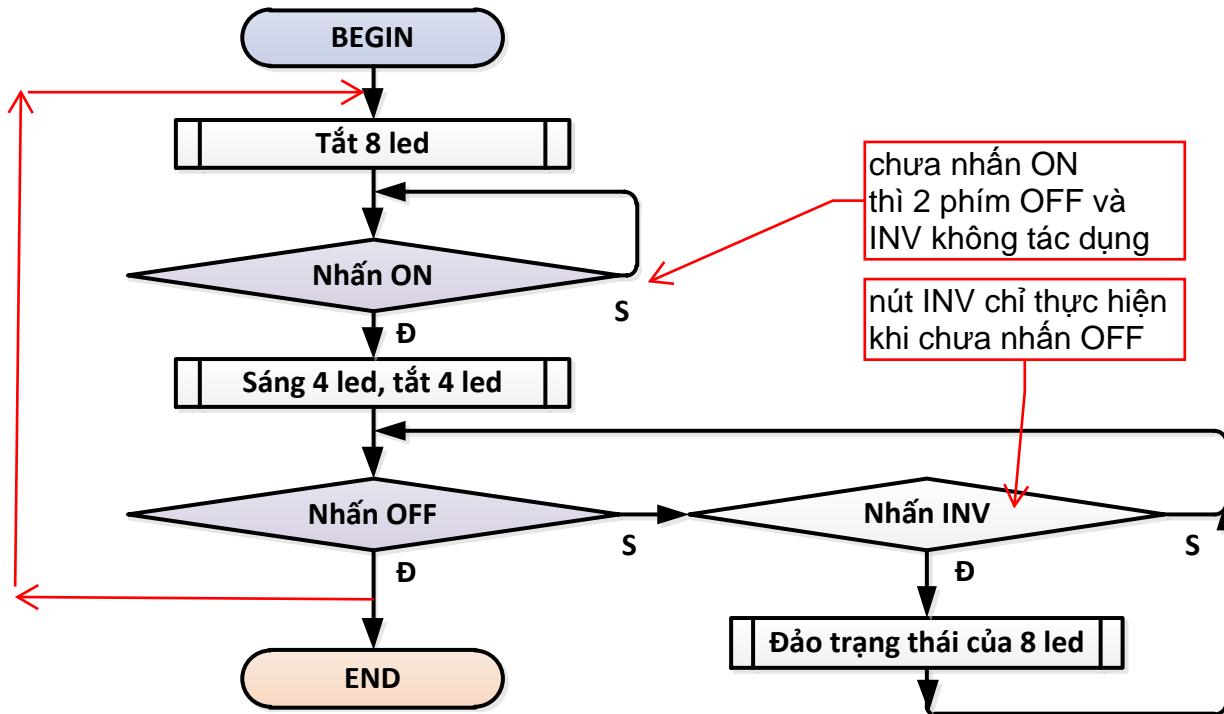
e. Quan sát kết quả: sau khi nạp xong chương trình thì nhấn nút ON sẽ làm 8 led sáng, nhấn nút OFF sẽ làm 8 led tắt, nếu kết quả không đúng yêu cầu thì kiểm tra lại chương trình.

f. Giải thích chương trình:

Bài mẫu 322. Chương trình điều khiển 8 LED bằng 3 nút ON, OFF, INV. Khi có điện thì 8 led tắt, khi nhấn ON thì 4 led sáng, khi nhấn OFF thì led tắt, khi nhấn INV thì 4 led sáng thành tắt, 4 led tắt thành sáng.

Lưu tên file là “BAI_322_ON_OFF_INV_8LED”

- a. Mục đích: điều khiển để thấy hiện tượng dội gây ra khi nhấn nút trong điều khiển.
 b. Lưu đồ:



Hình 3-7. Lưu đồ điều khiển 8 led sáng tắt bằng 2 nút ON, OFF và INV.

- c. Chương trình:

```

#include <tv_pickit2_shift_1.c>
unsigned int8 y;
void main()
{
    set_up_port_ic_chot();
    set_tris_b(0x3c);
    y=0;
    xuat_32led_don_4byte(0,0,0,0);
    while(true)
    {
        while(input(on));
        y=0x0f;
        xuat_32led_don_4byte(0,0,0,y);
        do
        {
            if (!input(inv))
            {
                y=~y;
                xuat_32led_don_4byte(0,0,0,y);
            }
        }while(input(off));
        xuat_32led_don_4byte(0,0,0,0);
    }
}

```

vòng lặp do/while sẽ thực hiện khi chưa nhấn OFF

Annotations:

- Red box: chờ nhấn ON (Waiting for ON press)
- Red box: if(input(INV) == 0) (if input(INV) == 0)
- Red box: đoạn chương trình thực hiện khi nhấn INV (=0) (Program segment executed when INV is pressed (=0))
- Red box: input(off) == 1 --> chưa nhấn OFF (input(off) == 1 --> not pressing OFF)

d. Tiến hành biên dịch và nạp.

e. Quan sát kết quả: Sau khi nạp xong chương trình thì nhấn nút ON sẽ làm 4 led sáng, nhấn nút OFF sẽ làm 4 led tắt, nhấn nút INV sẽ làm 4 led sáng thành tắt, 4 led tắt thành sáng, nếu kết quả không đúng yêu cầu thì kiểm tra lại chương trình.

f. Giải thích chương trình:

Trong chương trình này khi chạy thực tế thì do tốc độ của vi điều khiển quá nhanh, nên khi nhấn đảo chiều thì do thời gian nhấn phím dài nên vi điều khiển thực hiện đảo led liên tục, ta sẽ nhìn thấy 8 led sáng luôn cho đến khi ta buông phím, hoặc ta nhấn nhanh thì trạng thái đảo của led không thể xác định rõ ràng.

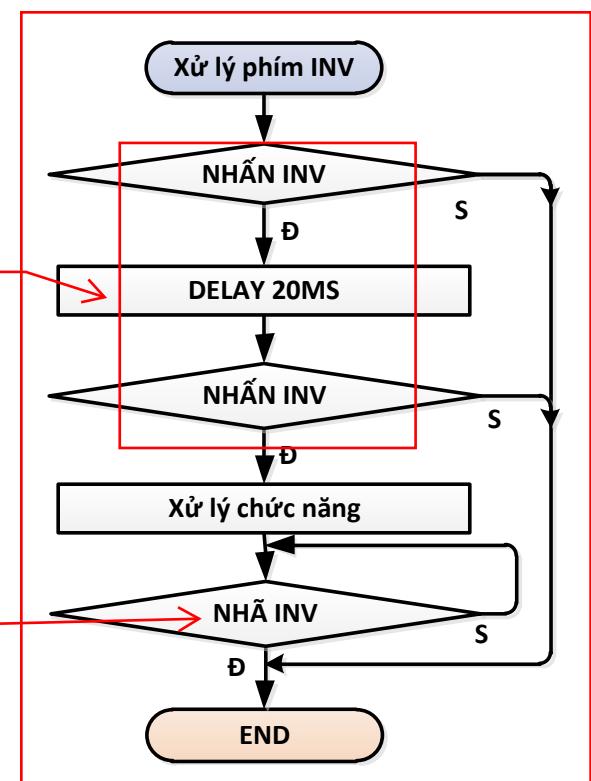
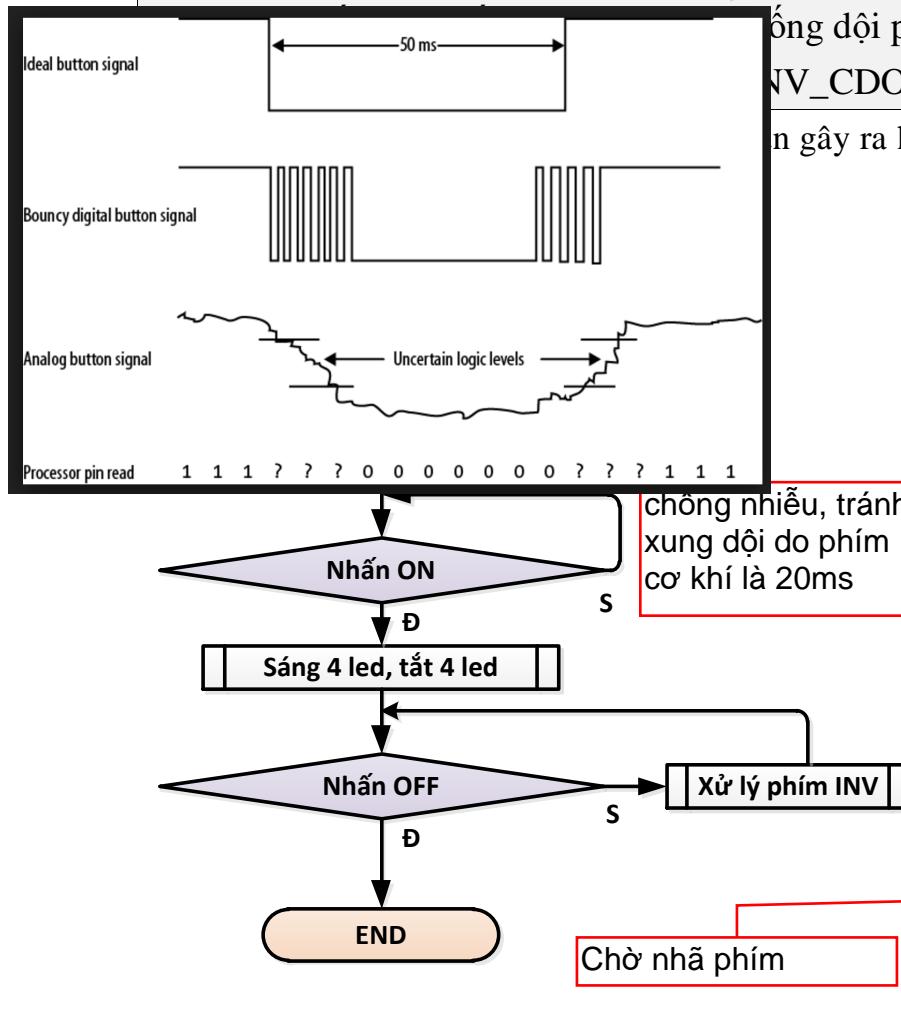
Để điều khiển chính xác thì phải chống dội phím nhấn, rồi xử lý chức năng và kiểm tra buông phím.

Trong bài điều khiển chỉ có phím INV gây ra ảnh hưởng nên lưu đồ và chương trình xử lý phần chống dội và chờ buông phím INV theo sau.

Bài mẫu 323. Chương trình điều khiển 8 LED bằng 3 nút ON, OFF, INV. Khi có điện

“INV_CDOI_8LED”

n gây ra hiện tượng dội, biết cách lập trình chống



Hình 3-8. Lưu đồ điều khiển 8 led sáng tắt bằng 2 nút ON, OFF và INV – chống dội.

c. Chương trình:

```
#include    <tv_pickit2_shift_1.c>
unsigned int8 y;

void phim_inv()
{
    if (!input(inv)) // kiểm tra phím có nhấn lần thứ 1
    {
        delay_ms(20); // delay trì hoãn khoảng thời gian dội của phím
        if (!input(inv)) // kiểm tra phím có nhấn lần 2
        {
            y=~y;
            xuat_32led_don_4byte(0,0,0,y);
            while(!input(inv)); // chờ nhả phím
        }
    }
}

void main()
{
    set_up_port_ic_chot();
    set_tris_b(0x3c);
    y=0x00;
    xuat_32led_don_4byte(0,0,0,0);
    while(true)
    {
        while(input(on));
        y=0x0f;
        xuat_32led_don_4byte(0,0,0,y);
        do{ phim_inv(); }
        while(input(off));
        xuat_32led_don_4byte(0,0,0,0);
    }
}
```

d. Tiến hành biên dịch và nạp.

e. Quan sát kết quả: trong bài này thì chống dội và chờ nhả phím thì mới thực hiện.

f. Giải thích chương trình:

Bài tập 324. Dùng vi điều khiển giao tiếp với 16 led đơn và 3 nút nhấn được đặt tên là UP, DW, CLR. Khi cấp điện thì 16 led tắt, khi nhấn UP thì led sáng dần lên từ phải sang trái – mỗi lần nhấn thì 1 led sáng, khi nhấn DW thì led tắt dần theo chiều ngược lại, khi nhấn CLR thì xóa hết.

Lưu tên file là “BAI_324_16LED_UP_DW_CLR_1”.

Nhấn mới thực hiện -> ta xử lý ngay trong hàm nút nhấn

Bài tập 325. Giống bài 324 nhưng nút nhấn CLR có thêm chức năng đảo chiều sáng và tắt của led: ví dụ khi nhấn UP thì sáng dần từ phải sang trái, sau khi nhấn CLR thì khi nhấn UP led sẽ sáng dần theo chiều từ trái sang phải, tương tự cho nút nhấn DW.

Lưu tên file là “BAI_325_16LED_UP_DN_CLR_2”.

Bài tập 326. Hãy lập trình dùng vi điều khiển giao tiếp với 32 led đơn và 3 nút nhấn được đặt tên là UP, DN, CLR.

Hệ thống điều khiển 32 led với các chương trình như sau:

1. Chớp tắt 32 led.
2. Sáng tắt dần 32 led từ phải sang trái.
3. Sáng tắt dần 32 led từ trái sang phải.
4. Sáng tắt dần 32 led từ ngoài vào.
5. Sáng tắt dần 32 led từ trong ra.
- ~~6. Sáng dần 32 led từ phải sang trái.~~
- ~~7. Sáng dần 32 led từ trái sang phải.~~

Khi cấp điện mặc nhiên vi điều khiển chạy chương trình 1.

Khi nhấn UP thì chuyển sang chương trình kế theo chiều tăng - khi đến chương trình số 7 thì ngừng.

Khi nhấn DN thì chuyển sang chương trình kế theo chiều giảm - khi đến chương trình số 1 thì ngừng.

Khi nhấn CLR thì quay về chương trình số 1.

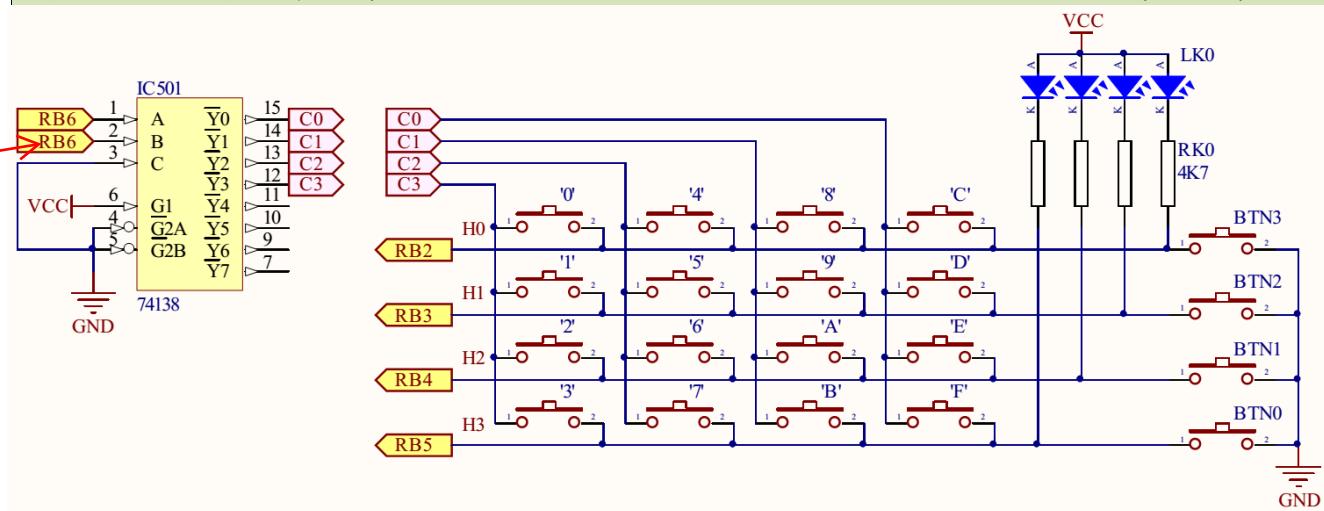
Chú ý là khi nhấn thì mạch phải chuyển ngay lập tức.

Lưu tên file là “BAI_326_TONGHOP_UP_DN_CLR” để lưu tất cả các file của project.

3.2.5 Mạch điện giao tiếp vi điều khiển với ma trận phím

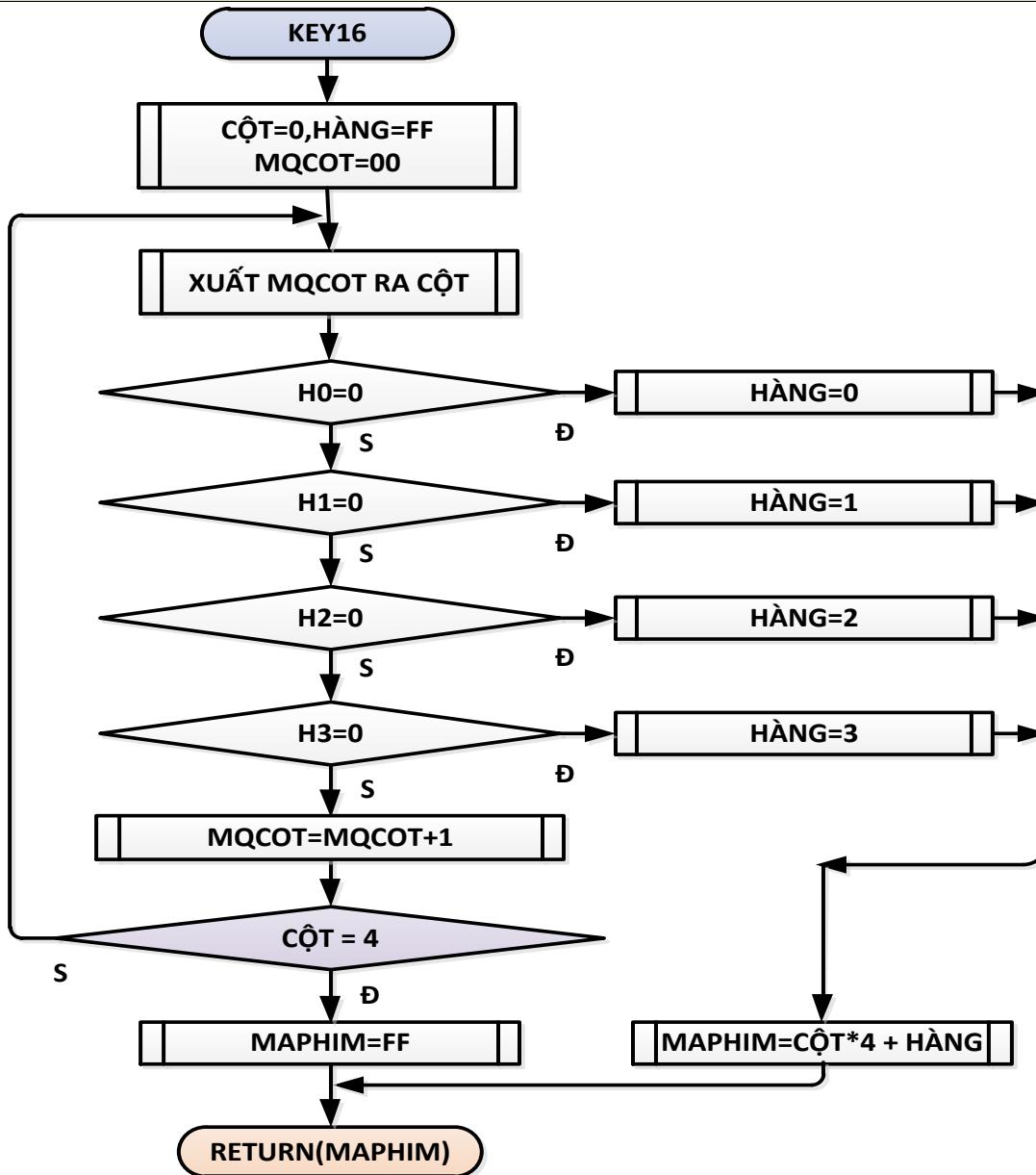
Phần này thực hành các bài giao tiếp vi điều khiển với ma trận bàn phím 4x4.

Mạch điện giao tiếp vi điều khiển với các nút nhấn như hình 3-9.



Hình 3-9. Sơ đồ nguyên lý giao tiếp vi điều khiển với ma trận 16 phím.

Vì điều khiển dùng 6 bit của portB giao tiếp với ma trận 4x4: 4 bit RB2 đến RB5 đóng vai trò là hàng, 2 bit RB6 và RB7 xuất ra số nhị phân có 4 trạng thái 00, 01, 10, 11 qua mạch giải mã 2 đường sang 4 đường ngõ ra tích cực mức 0 sẽ tạo ra 4 trạng thái tương ứng là 1110, 1101, 1011, 0111 dùng để quét 4 cột. Lưu đồ quét ma trận phím:



Hình 3-10. Lưu đồ quét ma trận 16 phím.

3.2.6 Các chương trình điều khiển dùng ma trận phím

Bài mẫu 331. Tạo file thư viện quét phím để dễ dàng sử dụng về sau.

Trong thư viện có 2 hàm quét phím và chống dội, bạn chỉ dùng 1 trong 2 hàm.

Hàm thứ nhất, khi nhấn phím xong mới tiến hành chống dội chờ bạn phải buông phím đã nhấn thì mã phím đó mới được chấp nhận và tiến hành xử lý mã cho phím đó. Ta có thể xem giống như mạch đếm tích cực cạnh xuống của xung clock vậy.

Hàm thứ hai, khi ta nhấn phím thì mạch sẽ xử lý ngay cho dù ta còn nhấn phím, sau đó chương trình chống dội mới thực hiện. Ta có thể xem giống như mạch đếm tích cực cạnh lên của xung clock vậy.

Chú ý: Khi viết thư viện thì bạn không được biên dịch vì chương trình chưa hoàn chỉnh, khi biên dịch thì sẽ báo lỗi.

Lưu tên file thư viện quét phím là “TV_PICKIT2_SHIFT_KEY4X4_138”

```

const unsigned char maquetkey[4] = {0x3f, 0x7f, 0xbf, 0xff};
unsigned int8 mpt1=0, mpt2=0;
unsigned int key_nhan()
{
    signed int8 maphim, hang, cot;
    maphim = hang = 0xff;
    for(cot=0; cot<4; cot++)
    {
        output_b(maquetkey[cot]);
        if (!input(pin_b2)) {hang = 0; break;}
        else if (!input(pin_b3)) {hang = 1; break;}
        else if (!input(pin_b4)) {hang = 2; break;}
        else if (!input(pin_b5)) {hang = 3; break;}
    }
    if (hang != 0xff) maphim = cot*4 + hang;
    return (maphim);
}
//xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
//ham tich cuc canh len
//xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
unsigned int key_4x4_up()
{
    mpt1 = key_nhan();
    if (mpt1 != 0xff)
    {
        delay_ms(20);
        mpt1 = key_nhan();
        do{mpt2 = key_nhan();} while (mpt2 == mpt1);
    }
    return (mpt1);
}
//xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
//ham tich cuc canh xuong
//xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
unsigned int key_4x4_dw()
{
    mpt1=key_nhan();
    if (mpt1!=0xff)
    {
        if (mpt1!=mpt2)
        {
            return (mpt1); //phím mới
        }
    }
}

```

```

        mpt2=mpt1;
    }
    else
    {
        delay_ms(1);
        mpt1=key_nhan();
        do{ mpt2=key_nhan(); }
        while (mpt2!=mpt1);
        return(mpt1); //thoat vi trung ma phím da nhan
        mpt2=mpt1;
    }
}
else
{
    return(mpt1); //thoat vi khong nhan
    mpt2=mpt1;
}
}
}

```

Bài mẫu 332. Chương trình quét phím ma trận 4x4, khi nhấn phím nào thì mã của phím hiển thị ở các led đơn dạng số nhị phân.

Khi nhả phím thì mã mới xuất hiện trên led.

Lưu tên file là “BAI_332_MATRANPHIM_LEDON”

- Mục đích: biết cách viết thư viện, biết cách lập trình giao tiếp bàn phím ma trận khi nhấn thì hiển thị mã phím nhị phân trên led.
- Lưu đồ: sinh viên hãy tự viết.
- Chương trình:

```

#include <tv_pickit2_shift_1.c>
#include <tv_pickit2_shift_key4x4_138.c> //khai báo thư viện
signed int8 mp; //quét phím ma trận
void main()
{
    set_up_port_ic_chot();
    set_tris_b(0x3c); //trang 59, thiết lập 4
    //ngõ vào cho hàng RB2 - RB5
    while(true)
    {
        mp = key_4x4_dw(); //kiểm tra phím
        if (mp != 0xff) xuat_32led_don_4byte(0,0,0,mp);
    }
}

```

- Tiến hành biên dịch và nạp.
- Quan sát kết quả: Bạn tiến hành của phím sẽ hiển thị trên led đó Sau đó bạn hãy hiệu chỉnh lại

```

MP = KEY_4X4_DW(); //KT PHÍM MA TRẬN
IF(MP!=0xFF) //CÓ PHÍM NHẤN THÌ KHÁC 0xFF
{
    //XỬ LÝ CHỨC NĂNG CỦA CÁC PHÍM
    IF(MP==0) Y=0xFF;
    IF(MP==7) Y = 0;
    XUAT_32_LED_DON_4BYTE(0,0,0,Y);
}

```

`key_4x4_up () ;`" rồi biên dịch và nạp. Tiến hành nhấn phím thì khi bạn nhấn, mã phím sẽ có ngay lập tức.

- f. Giải thích chương trình: Hãy giải thích chương trình con quét bàn phím ma trận.

Bài mẫu 333. Chương trình điều khiển 8 led với yêu cầu: khi nhấn phím số ‘1’ thì 8 led sáng, khi nhấn phím số ‘0’ thì 8 led tắt. Phím số ‘0’ và ‘1’ là của bàn phím ma trận.

Lưu tên file là “BAI_333_MATRANPHIM_8LED_ON_OFF

- a. Mục đích: biết ứng dụng bàn phím ma trận để điều khiển.
- b. Lưu đồ: sinh viên hãy tự viết.
- c. Chương trình:

```
#include <tv_pickit2_shift_1.c>
#include <tv_pickit2_shift_key4x4_138.c>
signed int8 mp;
void main()
{
    set_up_port_ic_chot();
    set_tris_b(0x3c);
    while(true)
    {
        do {mp= key_nhan();} while(mp!=1);
        xuat_32led_don_4byte(0,0,0,0xff);
        do {mp= key_nhan();} while(mp!=0);
        xuat_32led_don_4byte(0,0,0,0);
    }
}
```

KHÔNG NHẤN PHÍM
THÌ CHỜ

- d. Tiến hành biên dịch và nạp.
- e. Quan sát kết quả: khi nhấn phím ‘1’ thì 8 led sáng, nhấn phím ‘0’ thì 8 led tắt, những led còn lại không có tác dụng.
- f. Giải thích chương trình: hãy cho biết tại sao trong chương trình lại gọi hàm key_nhan();

Bài tập 334. Hãy viết chương trình điều khiển 8 led (có số thứ tự từ led0 đến led7).

Yêu cầu: 8 nút nhấn từ 0 đến 7 có chức năng ON: Khi nhấn phím số ‘0’ thì led0 sáng, tương tự cho đến số 7 thì led7 sáng.

8 nút nhấn từ A đến F có chức năng OFF: khi nhấn phím số ‘8’ thì led0 tắt, tương tự cho đến số F thì led7 tắt.

Lưu tên file là “BAI_334_MATRANPHIM_8LED_16_BTN_ON_OFF”.

~~Bài tập 335.~~ Hãy làm lại bài 3-26 nhưng sử dụng 7 phím từ 0 đến 7 của bàn phím ma trận, không dùng UP, DW và CLR.

Lưu tên file là “BAI_335_7CT_7BTN”.

3.3 CHƯƠNG TRÌNH THƯ VIỆN CHO MODULE 32 LED ĐƠN

Để tiện lợi cho các bài thực hành kết hợp nhiều module và tránh việc lặp lại nên các chương trình điều khiển 32 led đơn được viết ở trong thư viện.

Bài tập 336. Viết thư viện các bài điều khiển 32led.

Lưu tên file là “TV_PICKIT2_SHIFT_32_LED_DON.C”.

Thư viện này lưu các bài điều khiển 32 led viết theo cách khác là sử dụng lệnh if thay cho lệnh for.

Dùng lệnh if tuy có dài dòng hơn nhưng hay hơn dùng lệnh for, bạn hãy tìm hiểu thử hay hơn như thế nào.

```
unsigned int32      y=0,yp_chay=1,yt_chay=0x80000000,y_codinh=0;
unsigned int16 zt,zt_chay_t=0x8000,zt_chay_p=1,zt_codinh=0;
unsigned int16 zp,zp_chay_t=0x8000,zp_chay_p=1,zp_codinh=0;

unsigned int16 tgdlay=0;
unsigned int8 dem_led=0;
signed i32=32,j32=0,ttct_td=1,i16=16,j16=0;

unsigned int32
za=1,ya=1,xa=0xfffffffffe,zb=0x80000000,yb=0x80000000,xb=0x7fffffff;
signed ia=1,ja=1;
//xxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
void reset_toan_bo_bien()
{
    ttct_td++;
    y=0;          yp_chay=1;          yt_chay=0x80000000;          y_codinh=0;
    zt_chay_t=0x8000;      zt_chay_p=1;      zt_codinh=0;
    zp_chay_t=0x8000;      zp_chay_p=1;      zp_codinh=0;
    zt=zp=0;
    i32=32;j32=0;i16=16,j16=0;
    dem_led=0;

    za=1;      ya=1;      xa=0xfffffffffe;      ia=1;      ja=1;
    zb=0x80000000;      yb=0x80000000;      xb=0x7fffffff;
}

//xxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
void tang_dem_led_delay_1(unsigned int8 dl)
{
    unsigned int8 i;
    for (i=0; i<dl;i++)
        hien_thi_8led_7doan_quet_all();
```

quét led 2 lần (nhiều lần)
--> để tạo thời gian delay
(led quét kg bị tắt)

```

        dem_led++;
    }
//xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
void tang_dem_led_delay_0(unsigned int8 dl)
{
    if (dl>0)      delay_ms(dl);
    dem_led++;
}
//xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
void delay_chon(unsigned int8 dl,chondelay)
{
    if (!chondelay) tang_dem_led_delay_0(dl);
    else           tang_dem_led_delay_1(dl);
}
//xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
void diem_sang_dich_trai_mat_dan_32led(unsigned int8 dl,chondelay)
{
    if(ia<33)
    {
        if(ja>0)
        {
            zb = (zb << 1);
            y = xb|zb;
            xuat_32led_don_1dw(y);
            delay_chon(dl,chondelay);
            ja--;
        }
        else
        {
            ia++;
            ja=ia;
            yb = yb>>1;
            xb = xb>>1;
            zb = yb;
        }
    }
    else reset_toan_bo_bien();
}
//xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
void diem_sang_dich_phai_mat_dan_32led(unsigned int8 dl,chondelay)
{
    if(ia<33)
    {
        if(ja>0)
        {
            za = (za >> 1);
            y = xa|za;
            xuat_32led_don_1dw(y);
            delay_chon(dl,chondelay);
            ja--;
        }
    }
}

```

2,1

1 --> !1 --> 0 (!TRUE --> FALSE --> KHÔNG TH.HIỆN IF --> TH ELSE

2

```

        }
        else
        {
            ia++;
            ja=ia;
            ya = ya<<1;
            xa = xa <<1;
            za = ya;
        }
    }
    else reset_toan_bo_bien();
}
//xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
void dao_dl_32bit_va_xuat()
{
    y=~y;
    xuat_32led_don_1dw(y);
}
//xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
void sang_tat_32led(unsigned int8 dl,chondelay)
{
    if (tgdlay==1)          xuat_32led_don_1dw(y);
    else if (tgdlay==50)     dao_dl_32bit_va_xuat();
    else if (tgdlay==100)    dao_dl_32bit_va_xuat();
    else if (tgdlay==150)    dao_dl_32bit_va_xuat();
    else if (tgdlay==200)
    {
        ttct_td++;   y=0;      tgdlay=0;
    }
    tgdlay++;
    delay_chon(dl,chondelay);
}
//xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
void sang_tat_dan ngoai_vao_32led(unsigned int8 dl,chondelay)
{
    if (dem_led<16)
    {
        xuat_32led_don_2word(zp,zt);
        zp= (zp<<1)+1;
        zt= (zt>>1)+0x8000;
        delay_chon(dl,chondelay);
    }
    else if (dem_led<32)
    {
        xuat_32led_don_2word(zp,zt);
        zp= (zp<<1);
        zt= (zt>>1);
        delay_chon(dl,chondelay);
    }
    else reset_toan_bo_bien();
}

```

2, 1
dl = 2
chondelay = 1

2,1

```

}

//xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
void sang_tat_dan_trong_ra_32led(unsigned int8 dl,chondelay)
{
    if (dem_led<16)
    {
        xuat_32led_don_2word(zp,zt);
        zp= (zp>>1)+0x8000;
        zt= (zt<<1)+1;
        delay_chon(dl,chondelay);
    }
    else if (dem_led<32)
    {
        xuat_32led_don_2word(zp,zt);
        zp= (zp>>1);
        zt= (zt<<1);
        delay_chon(dl,chondelay);
    }
    else reset_toan_bo_bien();
}
//xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
void sang_tat_dan_phai_sang_trai_2x16led(unsigned int8 dl,chondelay)
{
    if (dem_led<16)
    {
        xuat_32led_don_2word(zp,zp);
        zp= (zp>>1)+0x8000;
        delay_chon(dl,chondelay);
    }
    else if (dem_led<32)
    {
        xuat_32led_don_2word(zp,zp);
        zp= (zp>>1);
        delay_chon(dl,chondelay);
    }
    else reset_toan_bo_bien();
}
//xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
void sang_tat_dan_trai_sang_phai_2x16led(unsigned int8 dl,chondelay)
{
    if (dem_led<16)
    {
        xuat_32led_don_2word(zt,zt);
        zt= (zt<<1)+1;
        delay_chon(dl,chondelay);
    }
    else if (dem_led<32)
    {
        xuat_32led_don_2word(zt,zt);
        zt= (zt<<1);
    }
}

```

```
        delay_chon(dl, chondelay);
    }
    else reset_toan_bo_bien();
}
//xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
void sang_tat_dan_pst_32led(unsigned int8 dl, chondelay)
{
    if (dem_led<32)
    {
        xuat_32led_don_1dw(y);
        y= (y<<1)+1;
        delay_chon(dl, chondelay);
    }
    else if (dem_led<64)
    {
        xuat_32led_don_1dw(y);
        y= (y<<1);
        delay_chon(dl, chondelay);
    }
    else reset_toan_bo_bien();
}
//xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
void sang_tat_dan_tsp_32led(unsigned int8 dl, chondelay)
{
    if (dem_led<32)
    {
        xuat_32led_don_1dw(y);
        y= (y>>1)+0x80000000;
        delay_chon(dl, chondelay);
    }
    else
    if (dem_led<64)
    {
        xuat_32led_don_1dw(y);
        y= (y>>1);
        delay_chon(dl, chondelay);
    }
    else reset_toan_bo_bien();
}
//xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
void diem_sang_di_chuyen_pst_32led(unsigned int8 dl, chondelay)
{
    if (dem_led==0)
    {
        y=1;
        xuat_32led_don_1dw(y);
        y= (y<<1);
        delay_chon(dl, chondelay);
    }
    else if (dem_led<33)
```

```

{
    xuat_32led_don_1dw(y);
    y= (y<<1);
    delay_chon(dl,chondelay);
}
else reset_toan_bo_bien();
}

//xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
void diem_sang_di_chuyen_tsp_32led(unsigned int8 dl,chondelay)
{
    if (dem_led==0)
    {
        y=0x80000000;
        xuat_32led_don_1dw(y);
        y= (y>>1);
        delay_chon(dl,chondelay);
    }
    else if (dem_led<33)
    {
        xuat_32led_don_1dw(y);
        y= (y>>1);
        delay_chon(dl,chondelay);
    }
    else reset_toan_bo_bien();
}
//xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
void sang_don_pst_32led(unsigned int8 dl,chondelay)
{
    if(i32>0)
    {
        if(j32<i32)
        {
            y = y_codinh|yp_chay;
            xuat_32led_don_1dw(y);
            delay_chon(dl,chondelay);
            yp_chay = yp_chay <<1;
            j32++;
        }
        if (j32==i32)
        {
            i32--;
            j32=0;
            y_codinh=y;
            yp_chay = 1;
        }
    }
    else reset_toan_bo_bien();
}
//xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
void sang_don_tsp_32led(unsigned int8 dl,chondelay)

```

```

{
    if(i32>0)
    {
        if(j32<i32)
        {
            y = y_codinh|yt_chay;
            xuat_32led_don_1dw(y);
            delay_chon(dl,chondelay);
            yt_chay = yt_chay >>1;
            j32++;
        }
        if (j32==i32)
        {
            i32--;
            j32=0;
            y_codinh=y;
            yt_chay = 0x80000000;
        }
    }
    else reset_toan_bo_bien();
}
//xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
void sang_don_tnt_32led(unsigned int8 dl,chondelay)
{
    if(i16>0)
    {
        if(j16<i16)
        {
            zp = zp_codinh|zp_chay_p;
            zt = zt_codinh|zt_chay_t;
            xuat_32led_don_2word(zt,zp);
            delay_chon(dl,chondelay);

            zp_chay_p = zp_chay_p <<1;
            zt_chay_t = zt_chay_t >>1;
            j16++;
        }
        if (j16==i16)
        {
            i16--;
            j16=0;
            zt_codinh=zt;    zp_codinh=zp;
            zp_chay_p = 1;   zt_chay_t = 0x8000;
        }
    }
    else reset_toan_bo_bien();
}
//xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
void sang_don_ttr_32led(unsigned int8 dl,chondelay)
{

```

```

if(i16>0)
{
    if(j16<i16)
    {
        zp = zp_codinh|zp_chay_t;
        zt = zt_codinh|zt_chay_p;
        xuat_32led_don_2word(zt,zp);
        delay_chon(dl,chondelay);
        zp_chay_t = zp_chay_t >>1;
        zt_chay_p = zt_chay_p <<1;
        j16++;
    }
    if (j16==i16)
    {
        i16--;
        j16=0;
        zt_codinh=zt;
        zp_codinh=zp;
        zp_chay_t = 0x8000;    zt_chay_p = 1;
    }
}
else reset_toan_bo_bien();
}

```

Bài mẫu 337. Giải bài tập 326. Do biết các bạn viết chưa được nên đây là bài giải cho các bạn tham khảo.

Lưu tên file là “BAI_337_TONGHOP_UP_DN_CLR” để lưu tất cả các file của project.

- Mục đích: Biết cách lập trình kết hợp các bài điều khiển 32 led giao với nhiều chương trình khác nhau dùng nút nhấn đơn để điều khiển. Biết cách viết để đáp ứng thời gian tốt nhất, biết sự khác biệt giữa cách lập trình dùng vòng lặp for và if.
- Lưu đồ: Sinh viên hãy tự viết.
- Chương trình:

```

#include    <tv_pickit2_shift_1.c>
signed ttct=1;
#include    <tv_pickit2_shift_32led_don.c>

void phim_up()
{
    if (!input(up) &&(ttct<7) )
    {
        delay_ms(10);
        if (!input(up))
        {
            ttct++;    ↙
            while(!input(up));
            reset_toan
        }
    }
}

```

```

        }
    }
//xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
void phim_dw()
{
    if (!input(dw) && (ttct>1))
    {
        delay_ms(10);
        if (!input(dw))
        {
            ttct--; ↗
            while(!input(dw));
        }
    }
}
//xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
void phim_clr()
{
    if (!input(clr)&&(ttct>1))
    {
        ttct=1;
        reset_toan_bo_bien();
    }
}
//xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
void main()
{
    set_up_port_ic_chot();
    set_tris_b(0x3c);
    while(true)
    {
        xuat_4led_7doan_1so(ma7doan[ttct]);
        if (ttct==1) sang_tat_32led(10,0);
        if (ttct==2) sang_tat_dan_pst_32led(10,0);
        if (ttct==3) sang_tat_dan_tsp_32led(10,0);
        if (ttct==4) sang_tat_dan ngoai_vao_32led(10,0);
        if (ttct==5) sang_tat_dan_trong_ra_32led(10,0);
        if (ttct==6) sang_don_pst_32led(10,0);
        if (ttct==7) sang_don_tsp_32led(10,0);
        phim_dw();
        phim_up();
        phim_clr();
    }
}

```

- d. Tiến hành biên dịch và nạp.
- e. Quan sát kết quả: Nếu kết quả không đúng yêu cầu thì kiểm tra lại chương trình.
- f. Giải thích chương trình: Hãy đọc hiểu và giải thích.

Bài mẫu 338. Giống bài trên nhưng chạy tự động nhiều chương trình.

Lưu tên file là “BAI_338_QUANGCAO” để lưu tất các file của project.

- Mục đích: Biết cách viết chương trình chạy tự động.
- Lưu đồ: Sinh viên hãy tự viết.
- Chương trình:

```
#include    <tv_pickit2_shift_1.c>
#include    <tv_pickit2_shift_321ed_don.c>

void main()
{
    set_up_port_ic_chot();
    while(true)
    {
        if (ttct_td==1)    sang_tat_321ed(10,0);
        if (ttct_td==2)    sang_tat_dan_pst_321ed(10,0);
        if (ttct_td==3)    sang_tat_dan_tsp_321ed(10,0);
        if (ttct_td==4)    sang_tat_dan ngoai_vao_321ed(10,0);
        if (ttct_td==5)    sang_tat_dan trong_ra_321ed(10,0);
        if (ttct_td==6)    sang_don_pst_321ed(10,0);
        if (ttct_td==7)    diem_sang_dich_trai_mat_dan_321ed(10,0);
        if (ttct_td==8)    sang_don_tsp_321ed(10,0);
        if (ttct_td==9)    diem_sang_dich_phai_mat_dan_321ed(10,0);
        if (ttct_td==10)   sang_tat_dan_trai_sang_phai_2x16led(40,0);
        if (ttct_td==11)   sang_tat_dan_phai_sang_trai_2x16led(40,0);
        if (ttct_td==12)   diem_sang_di_chuyen_pst_321ed(40,0);
        if (ttct_td==13)   diem_sang_di_chuyen_tsp_321ed(40,0);
        if (ttct_td==14)   sang_don_tnt_321ed(40,0);
        if (ttct_td==15)   sang_don_ttr_321ed(40,0);
        if (ttct_td>15)   ttct_td=1;
        xuat_4led_7doan_2so (ma7doan[ttct_td/10],ma7doan[ttct_td%10]);
    }
}
```

0 -- 1 ???

- Tiến hành biên dịch và nạp.
- Quan sát kết quả: Nếu kết quả không đúng yêu cầu thì kiểm tra lại chương trình.
- Giải thích chương trình: Hãy đọc hiểu và giải thích.

3.4 CÁC CÂU HỎI ÔN TẬP

Câu 3-1. Hãy vẽ sơ đồ mạch giao tiếp 4 nút nhấn với vi điều khiển PIC 18F4550 của kit.

Câu 3-2. Hãy vẽ sơ đồ mạch giao tiếp bàn phím ma trận với vi điều khiển PIC 18F4550 của kit.

Câu 3-3. Hãy vẽ lưu đồ quét bàn phím ma trận 4x4.

Câu 3-4. Hãy vẽ dạng sóng dội khi nhấn phím và lưu đồ chống dội nút nhấn.

Câu 3-5. Hãy vẽ mạch điện giao tiếp vi điều khiển với 32 led đơn.

Câu 3-6. Hãy giải thích nguyên lý vi điều khiển gửi dữ liệu ra 32 led đơn.

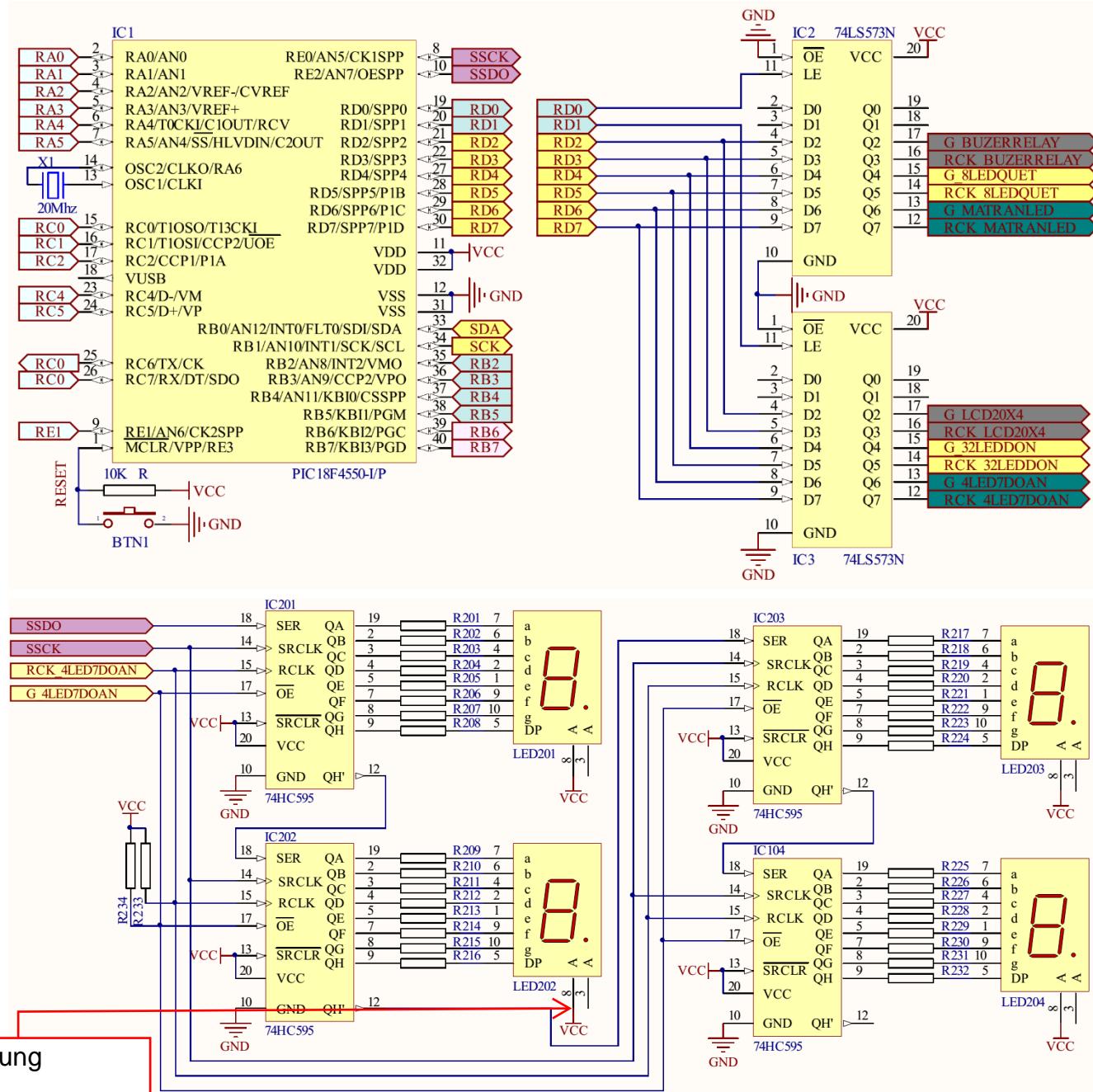
***Chương 4: THỰC HÀNH
MODULE 2 – 4 LED 7 ĐOẠN
ANODE DÙNG THANH GHI
DỊCH 74HC595,
COUNTER/TIMER***

4.1 CÁC CHƯƠNG TRÌNH ĐIỀU KHIỂN 4 LED 7 ĐOẠN

4.1.1 Mạch điện giao tiếp vi điều khiển với module 4 led 7 đoạn

Phần này thực hiện các bài giao tiếp với 4 led 7 đoạn để thực hiện các bài đếm thời gian, đếm sản phẩm, hiển thị các thông tin là các số thập phân.

Mạch điện giao tiếp vi điều khiển với module 4 led 7 đoạn như hình 4-1.



Hình 4-1. Sơ đồ nguyên lý giao tiếp vi điều khiển với module 4 led 7 đoạn.

Trong mạch sử dụng 4 IC thanh ghi 74HC595 để mở rộng port điều khiển 4 led 7 đoạn.

Phản vi điều khiển PIC giao tiếp với IC thanh ghi dịch

Có 4 tín hiệu điều khiển là:

- SSDO dùng để dịch dữ liệu nối tiếp.
- SSCK dùng để cấp xung clock.
- RCK_4LED7DOAN dùng điều khiển nạp dữ liệu song song từ bên trong IC thanh ghi dịch ra bên ngoài để điều khiển led 7 đoạn.
- G_4LED7DOAN dùng để cho phép xuất dữ liệu.

Phản IC thanh ghi dịch giao tiếp với 4 led 7 đoạn

Mỗi IC 74HC595 có 8 bit kết nối với 1 led 7 đoạn, 4 IC điều khiển được 4 led 7 đoạn.

Số lượng byte dữ liệu điều khiển module 4 led 7 đoạn là 4 byte.

Khả năng mở rộng

Trong kit thực hành dùng 4 IC để điều khiển 4 led, sau này các ứng dụng thực tế cần nhiều led hơn thì bạn cứ kết nối thêm IC thanh ghi nối tiếp vào một cách dễ dàng.

4.1.2 Các hàm điều khiển module 4 led 7 đoạn

Module 4 led 7 đoạn giống module 32 led đơn chung các phần cơ bản như khởi tạo port, IC chốt, đóng chốt, mở chốt, ... phần này chỉ nêu các hàm xuất dữ liệu ra module 4 led 7 đoạn.

Hàm thứ 401: xuất dữ liệu 4 byte ra 4 led:

```
void xuat_4led_7doan_4so(unsigned int b1743,b1742,b1741,b1740)
{
    xuat_1byte(b1740);
    xuat_1byte(b1741);
    xuat_1byte(b1742);
    xuat_1byte(b1743);

    mo_ic_74573_a_thong_dl();
    output_low(rck_4led7doan);
    output_high(rck_4led7doan);
    mo_4_led_7doan;
    chot_ic_74573_a_goi_du_lieu;
}
```

Hàm này xuất 4 byte ra điều khiển 4 led 7 đoạn tương ứng, hàm này giống hàm xuất 4 byte ra module 32 led đơn. Khi gọi hàm này phải có đầy đủ 4 byte.

Hàm thứ 402: xuất dữ liệu 1 byte ra 1 led:

```
void xuat_4led_7doan_1so(unsigned int b1740)
{
    xuat_4led_7doan_4so(0xFF,0xFF,0xFF,b1740);
```

Trong một số ứng dụng chỉ dùng 1 led thì chỉ cần xuất 1 byte ra module 4 led, nếu gọi hàm số 1 thì phải kèm theo 3 byte 0xFF để tắt 3 led không dùng.

Để đơn giản thì ta dùng hàm này và chỉ cần truyền 1 byte nhưng thực tế hàm này sẽ tự động chèn thêm 3 byte 0xFF để đủ 4 byte.

Tương tự cho 2 hàm tiếp theo.

Hàm thứ 403: xuất dữ liệu 2 byte ra 2 led:

```
void xuat_4led_7doan_2so(unsigned int b1741,b1740)
{
    xuat_4led_7doan_4so(0xff,0xff,b1741,b1740);
}
```

Hàm thứ 404: xuất dữ liệu 3 byte ra 3 led:

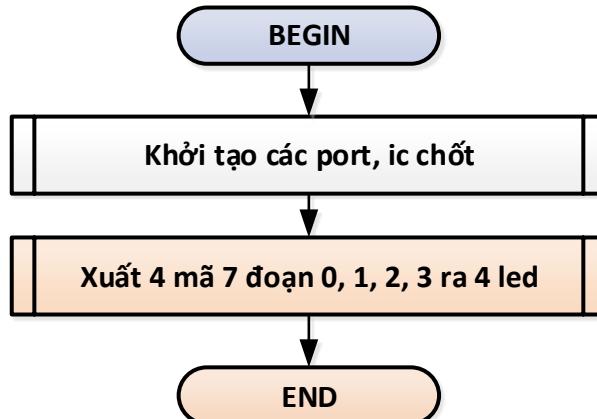
```
void xuat_4led_7doan_3so(unsigned int b1742,b1741,b1740)
{
    xuat_4led_7doan_4so(0xff,b1742,b1741,b1740);
}
```

4.1.3 Các chương trình điều khiển 4 led 7 đoạn

Bài mẫu 401. Chương trình điều khiển 4 LED 7 đoạn hiển thị 4 số từ 0 đến 3.

Lưu tên file là “BAI_401_HIENTHI_4SO”.

- Mục đích: biết điều khiển 4 led 7 đoạn hiển thị 4 số.
- Lưu đồ:



Hình 4-2. Lưu đồ điều khiển module 4 led 7 đoạn hiển thị 4 số 0, 1, 2, 3.

- Chương trình:

```
#include <tv_pickit2_shift_1.c>
void main()
{
    set_up_port_ic_chot();
    xuat_4led_7doan_4so(0xb0, 0xa4, 0xf9, 0xc0);
    while(true);
}
```

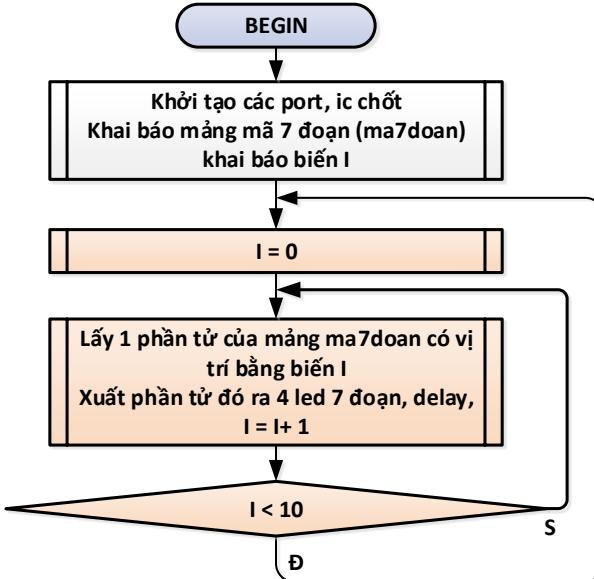
- Tiến hành biên dịch và nạp.
- Quan sát kết quả: 4 led 7 đoạn sẽ hiển thị 4 số: 3, 2, 1, 0.

TRANG 12, SỐ 3,
LED BÊN TRÁI
NGOÀI CÙNG

Bài mẫu 402. Chương trình điều khiển 1 LED 7 đoạn đếm từ 0 đến 9.

Lưu tên file là “BAI_402_DEM_0_9”.

- Mục đích: Biết cách lập trình bài đếm thời gian 1 số hiển thị trên 1 led 7 đoạn.
- Giải thuật:

**Hình 4- 3. Lưu đồ điều khiển đếm từ 0 đến 9.**

c. Chương trình:

```
#include <tv_pickit2_shift_1.c>
signed int8 i;
void main()
{
    set_up_port_ic_chot();
    while(true)
    {
        for (i=0;i<10;i++)
        {
            xuat_4led_7doan_1so(ma7doan[i]);
            delay_ms(500);
        }
    }
}
```

- d. Tiến hành biên dịch và nạp.
e. Quan sát kết quả: Sẽ có 1 led 7 đoạn đếm từ 0 đến 9.
f. Giải thích chương trình: Bạn hãy giải thích chương trình.

Bài mẫu 403. Chương trình điều khiển 2 LED 7 đoạn đếm từ 00 đến 99.

Lưu tên file là “BAI_403_DEM_00_99”.

- a. Mục đích: Biết cách lập trình bài đếm thời gian 2 số hiển thị trên 2 led 7 đoạn.
b. Lưu đồ: Sinh viên hãy tự viết.
c. Chương trình:

```
#include <tv_pickit2_shift_1.c>
signed int8 i;
void main()
{
```

```

set_up_port_ic_chot();
while(true)
{
    for (i=0;i<99;i++)
    {
        xuat_4led_7doan_2so(ma7doan[i/10],ma7doan[i%10]);
        delay_ms(200);
    }
}

```

LẤY MÃ 7 ĐOẠN
CỦA SỐ ĐÉM

CHIA LẤY PHẦN
NGUYÊN - CHỤC

CHIA LẤY PHẦN DƯ
ĐƠN VỊ
TÁCH SỐ

- d. Tiến hành biên dịch và nạp.
- e. Quan sát kết quả: sẽ có 2 led 7 đoạn đếm từ 00 đến 99.
- f. Giải thích chương trình: bạn hãy giải thích chương trình.

Bài tập 404. Hãy viết chương trình điều khiển 2 led của 4 LED 7 đoạn đếm giây từ 00 đến 59, thời gian delay là 1000ms. Nếu muốn chạy nhanh thì giảm delay.

Lưu tên file là “BAI_404_DEM_GIAY_00_59”. for 0 <60 ++

2 for : 0-59 58-1

Bài tập 405. Hãy viết chương trình điều khiển 2 led của 4 LED 7 đoạn đếm từ 00 đến 59, rồi đếm về 00, rồi lặp lại, thời gian delay là 200ms.

Lưu tên file là “BAI_405_DEM_00_59_00”.

số 59 và số 0 có thời
gian delay gấp đôi
các số khác

Bài tập 406. Hãy viết chương trình điều khiển 2 led của 4 LED 7 đoạn đếm từ 00 đến 59, với thời gian delay là 200ms, khi bằng 59 thì ngừng lại khoảng 3 giây i=0; xuống 00, với thời gian trễ là 200ms, khi bằng 00 thì ngừng 3 giây rồi lặp do

Sử dụng lệnh do while.

Lưu tên file là “BAI_406_DEM_00_59_3S_00_3S”.

```

{
    ... //nội dung lặp
    i++;
}
while(i<60);

```

Bài mẫu 407. Chương trình điều khiển 3 LED 7 đoạn đếm từ 000 đến 999.

Lưu tên file là “BAI_407_DEM_000_999”.

- a. Mục đích: Biết cách lập trình bài đếm thời gian 3 số hiển thị trên 3 led 7 đoạn.
- b. Lưu đồ: sinh viên hãy tự viết.
- c. Chương trình:

```

#include <tv_pickit2_shift_1.c>
signed int16 i;
void main()
{
    set_up_port_ic_chot();
    while(true)
    {

```

```

for (i=0;i<1000;i++)
{
    xuat_4led_7doan_3so (ma7doan[i/100],
    ma7doan[i/10%10],ma7doan[i%10]);
    delay_ms (50);
}
}

```

$123/10 = 12$ du 3
 $12/10 = 1$ du 2

- d. Tiến hành biên dịch và nạp.
- e. Quan sát kết quả: sẽ có 3 led 7 đoạn đếm từ 000 đến 999.
- f. Giải thích chương trình: bạn hãy giải thích chương trình.

Bài tập 408. Hãy viết chương trình điều khiển 4 LED 7 đoạn đếm từ 0000 đến 9999.

Lưu tên file là “BAI_408_DEM_0000_9999”.

Gợi ý: mở rộng giới hạn cho vòng lặp for, hiệu chỉnh phần giải mã hiển thị.

Bài tập 409. Hãy viết chương trình điều khiển 4 LED 7 đoạn đếm **phút giây**.

Lưu tên file là “BAI_409_DEM_PHUT_GIAY”.

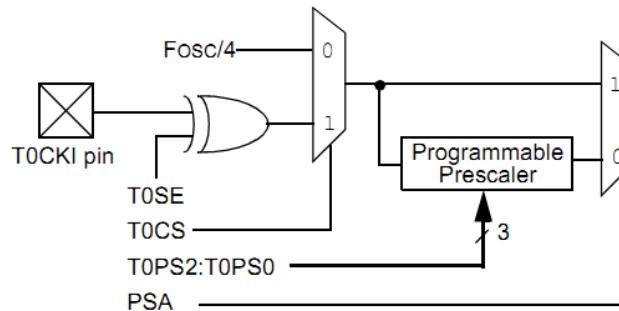
4.2 TIMER VÀ COUNTER CỦA VI ĐIỀU KHIỂN PIC

Vì điều khiển PIC 18F4550 có 4 timer T0, T1, T2 và T3.

T0, T1, T3 là timer/counter 16 bit, cả 3 đều có bộ chia trước. T2 là timer 8 bit không có bộ chia trước và chia sau phục vụ cho các ứng dụng đặc biệt.

4.2.1 Tóm tắt timer/counter T0

Phần này thực hiện đếm xung ngoại dùng timer T0. Vì nguồn xung từ cảm biến nghiệm đã cấp cho Timer T0 nên ta chỉ khảo sát timer T0.



```

2 for --> if
for(phut)
{ for(giay)
  {
  }
}
giay=0;
while(true)
{
  xuat
  delay
  giay++;
  if(giay==60)
  {
    giay = 0;
    phut++;
    if....
  }
}

```

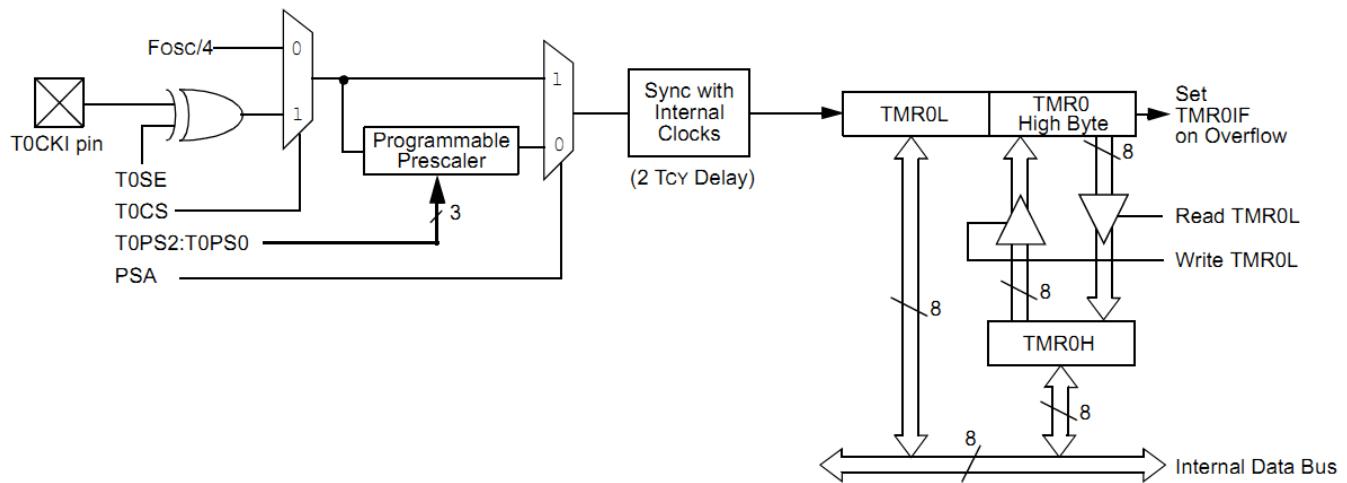
Note: Upon Reset, Timer0 is enabled in 8-bit mode with clock input from T0CKI maximum prescale.

Hình 4-4. Sơ đồ khối timer T0: chế độ 8 bit.

Vì điều khiển PIC 18F4550 có timer T0 có thể hoạt động ở 2 chế độ 8 bit hoặc 16 bit. Chế độ 8 bit thì giống họ vi điều khiển PIC16.

Về nguyên lý thì những gì của timer T0 ở họ PIC16 thì đều có thể thực hiện ở họ PIC18. Phần này không trình bày chi tiết mà chỉ tóm tắt.

Sơ đồ khôi timer T0 hoạt động ở chế độ 8 bit như hình 4-4 và 16 bit ở hình 4-5.



Note: Upon Reset, Timer0 is enabled in 8-bit mode with clock input from T0CKI maximum prescale.

Hình 4-5. Sơ đồ khôi timer T0: chế độ 16 bit.

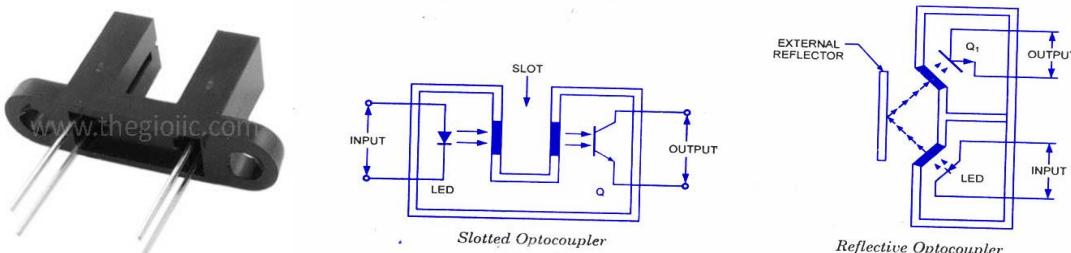
Bảng 4-1. Các thanh ghi có liên quan đến timer T0.

Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Reset Values on page
TMR0L	Timer0 Register Low Byte								54
TMR0H	Timer0 Register High Byte								54
INTCON	GIE/GIEH	PEIE/GIEL	TMR0IE	INT0IE	RBIE	TMR0IF	INT0IF	RBIF	53
INTCON2	RBPU	INTEDG0	INTEDG1	INTEDG2	—	TMR0IP	—	RBIP	53
T0CON	TMR0ON	T08BIT	T0CS	T0SE	PSA	T0PS2	T0PS1	T0PS0	54
TRISA	—	TRISA6 ⁽¹⁾	TRISA5	TRISA4	TRISA3	TRISA2	TRISA1	TRISA0	56

Các lệnh liên quan đến timer được trình bày ở tài liệu lý thuyết, các bit điều khiển liên quan đến timer thì có thể xem thêm trong thư viện 18F4550.h.

4.2.2 Cảm biến thu phát hồng ngoại HY860F

Để đếm xung ngoại thì nguồn xung đếm có thể tạo ra ở nhiều dạng khác nhau, nhưng trong bộ thí nghiệm thì sử dụng led thu phát hồng ngoại để nhận biết một sản phẩm đi qua để nó gần với thực tế ứng dụng bên ngoài.



Hình 4-6. Hình ảnh led thu phát HY860F và sơ đồ nguyên lý.

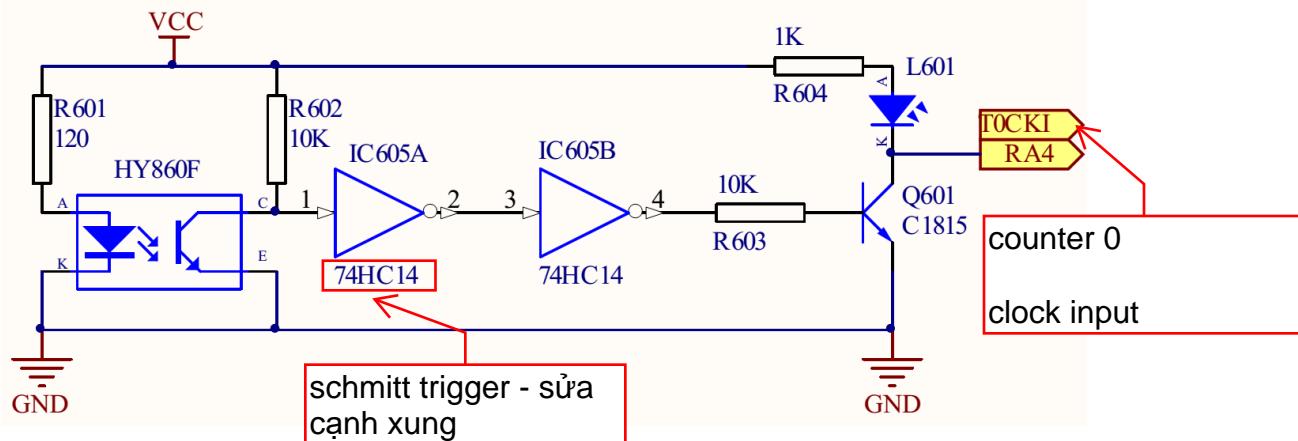
BTN sử dụng led thu phát HY860F có hình dạng và sơ đồ nguyên lý như hình 4-6.

Có 1 led phát ra ánh sáng hồng ngoại và 1 transistor thu.

Trong các robot có các động cơ và có gắn encoder thì có sử dụng dạng cảm biến này.

4.2.3 Mạch điện dùng T0 đếm xung ngoại

Khi sử dụng HY860F thì ta phải kết nối thêm điện trở và IC như hình sau:



Hình 4-7. Sơ đồ nguyên lý module thu phát hồng ngoại tạo xung.

Led phát sẽ nối với điện trở lên nguồn 5V để led luôn hoạt động phát ra ánh sáng hồng ngoại. Transistor có ngõ ra nối với điện trở và nối lên nguồn, đóng vai trò là điện trở tải để lấy tín hiệu.

Tín hiệu ngõ ra được nối với 2 cổng not của IC 74HC14 có chức năng đếm và schmitt trigger sửa dạng xung cho vuông và sau đó có thể đưa trực tiếp đến ngõ vào counter T0 hoặc đưa qua transistor để cho phép kết nối nhiều tín hiệu chung với nhau dạng cực thu để hở.

Khi không có sản phẩm nào che giữa led phát và led thu thì transistor được phân cực và ngõ ra transistor bằng 0, qua 2 cổng not thì cũng bằng 0, điều khiển transistor tắt, ngõ ra xung RA4/T0CKI bằng 1, led không có dòng nên tắt.

Khi bạn che ánh sáng giữa led phát và transistor thì transistor tắt, ngõ ra lên 1, qua 2 cổng not cũng bằng 1, điều khiển transistor dẫn. Ngõ ra xung RA4/T0CKI bằng 0, led có dòng nên led sáng.

Mỗi lần 1 sản phẩm đi qua thì tạo ra 1 xung, xung sẽ đưa đến counter T0 để đếm và hiển thị trên led.

4.2.4 Các ứng dụng đếm xung ngoại dùng counter T0

Phần này thực hành các bài đếm xung ngoại dùng counter T0 và hiển thị kết quả đếm trên module 4 led 7 đoạn.

Bài mẫu 411. Chương trình đếm sản phẩm hiển thị trên 4 led 7 đoạn dùng counter T0, giới hạn đếm từ 0 đến 100. Khi bằng 101 thì quay về 1.

Lưu tên file là “BAI_411_DEM_XUNG_T0”.

- Mục đích: Biết cách lập trình bài đếm xung ngoại dùng counter T0, hiển thị led 7 đoạn.

b. Lưu đồ: sinh viên hãy tự viết.

c. Chương trình:

```
#include <tv_pickit2_shift_1.c>
unsigned int8 t0;
void main()
{
    set_up_port_ic_chot();
    setup_timer_0 (t0_ext_l_to_h | t0_div_1| t0_8_bit);
    set_timer0(0);
    while(true)
    {
        t0 = get_timer0(); // đọc gt của counter
        xuat_4led_7doan_3so(ma7doan[t0/100],
        ma7doan[t0/10%10],ma7doan[t0%10]);
        if (t0>=101) set_timer0(1); // so sánh giá trị cuối
    }
}
```

t0: counter 0
ext: external (ngoài)
l to h: cạnh lên

tỷ lệ 1:1 - 1 xung vào - đếm 1 đơn vị

đặt số đếm ban đầu

mặc định 16 bit
8 bit (0-255)

so sánh giá trị cuối

d. Tiến hành biên dịch và nạp.

e. Quan sát kết quả: dùng giấy che giữa led phát và led thu thì mạch cảm biến hồng ngoại sẽ tạo ra xung, có đèn báo hiệu sáng lên rồi tắt tương ứng với xung, kết quả giá trị trên led sẽ tăng lên 1.

f. Giải thích chương trình: gồm có khởi tạo IC chốt, khởi tạo T0 chế độ đếm xung ngoại, hệ số chia là 1, chế độ 8 bit và giá trị bắt đầu là 0.

Vòng lặp while thực hiện đọc giá trị đếm được của T0 đếm giải mã hiển thị và so sánh giới hạn để kết lặp lại. Bài này đơn giản giúp các bạn dễ hiểu chính vì thế cần phải thực hiện tiếp để cải tiến cho tốt hơn qua các bài tiếp theo.

Bài mẫu 412. Chương trình đếm sản phẩm hiển thị trên 4 led 7 đoạn dùng counter T0, giới hạn đếm từ 0 đến 100. Khi bằng 101 thì quay về 1 và có xóa số 0 vô nghĩa.

Lưu tên file là “BAI_412_DEM_XUNG_T0_XOA_SO0_VN”.

a. Mục đích: Biết cách xóa số 0 vô nghĩa.

b. Lưu đồ: sinh viên hãy tự viết.

c. Chương trình:

```
#include <tv_pickit2_shift_1.c>
unsigned int8 t0;
unsigned int8 donvi, chuc, tram;
void giao_ma_hien_thi (unsigned int16 tam)
{
    donvi = ma7doan[tam %10];
    chuc = ma7doan[tam/10%10];
    tram = ma7doan[tam/100];
    if (tram==0xc0)
    {
        tram=0xff; // mã 7 đoạn số 0
    }
}
```

GIẢI MÃ

xóa số 0 vô nghĩa
(có cũng được mà kg
có cũng được)
105
005

```

        if (chuc==0xc0) chuc=0xff;
    }
    xuat_4led_7doan_3so(tram,chuc,donvi);
}
void main()
{
    set_up_port_ic_chot();
    setup_timer_0 (t0_ext_l_to_h | t0_div_1|t0_8_bit);
    set_timer0(0);
    while(true)
    {
        t0=get_timer0();
        giao_ma_hien_thi (t0);
        if (t0>=101) set_timer0(1);
    }
}

```

HIỂN THỊ nén tách riêng với giải mã

4 SỐ

BIẾN SỐ, CÀI ĐẶT
GT ĐÊM

- d. Tiến hành biên dịch và nạp.
- e. Quan sát kết quả: giống bài trước nhưng thực hiện cài tiến là xóa số 0 vô nghĩa.
- f. Giải thích chương trình:

Bài mẫu 413. Giống bài 412 nhưng chỉ giải mã hiển thị khi kết quả đếm của counter thay đổi.

Lưu tên file là “BAI_413_DEM_XUNG_T0_XOA_SO0_VN_CT”.

- a. Mục đích: Biết cách lập trình tối ưu để tiết kiệm thời gian, tránh làm những việc không cần thiết.
- b. Lưu ý: sinh viên hãy tự viết.
- c. Chương trình:

```

#include <tv_pickit2_shift_1.c>
unsigned int8      t0, t0_tam;
void main()
{
    set_up_port_ic_chot();
    setup_timer_0 (t0_ext_l_to_h | t0_div_1|t0_8_bit);
    set_timer0(0);
    t0_tam = t0 = 0;
    xuat_4led_7doan_giaima_xoa_so0(t0);
    while(true)
    {
        t0 = get_timer0();
        if (t0!=t0_tam)
        {
            t0_tam = t0;
            xuat_4led_7doan_giaima_xoa_so0(t0);
            if (t0>=101) set_timer0(1);
        }
    }
}

```

CHO PHÉP ĐÊM

so sánh gt hiện tại là t0 và
gt trước đó t0_tam

chỉ xuất 1 giá trị

d. Tiến hành biên dịch và nạp.

e. Quan sát kết quả: giống bài trước nhưng thực hiện cài tiến là xóa số 0 vô nghĩa.

Qua các bài thực hành của module này ta thấy muốn hiển thị một dữ liệu thành các số thập phân thì ta phải tách hàng đơn vị, hàng chục, hàng trăm và hàng ngàn, sau đó lấy mã 7 đoạn tương ứng và tiến hành xóa số 0 vô nghĩa và cuối cùng xuất ra led để hiển thị.

Đã thực hành và đã hiểu, để giúp các bạn không phải viết lại thì trong thư viện đã xây dựng sẵn hàm có chức năng này.

Hàm thứ 405: Hiển thị dữ liệu 16 bit ra module 4 led 7 đoạn:

```
unsigned char donvi4, chuc4, tram4, ngan4;
void xuat_4led_7doan_giaima_xoa_so0 (unsigned int16 tam)
{
    donvi4 = ma7doan[tam % 10];
    chuc4 = ma7doan[tam / 10 % 10];
    tram4 = ma7doan[tam / 100 % 10];
    ngan4 = ma7doan[tam / 1000 % 10];
    if (ngan4==0xc0)
    {
        ngan4=0xff;
        if (tram4==0xc0)
        {
            tram4=0xff;
            if (chuc4==0xc0)     chuc4=0xff;
        }
    }
    xuat_4led_7doan_4so (ngan4, tram4, chuc4, donvi4);
}
```

Hàm thực hiện chức năng như đã trình bày. Khi sử dụng chỉ cần gọi hàm và trao thông số 16 bit.

Viết sẵn hàm này giúp bạn tránh sự lặp lại và làm chương trình ngắn gọn. Với cách này thì sau này kết hợp nhiều module thì chương trình sẽ ngắn gọn.

Hàm thứ 406: Hiển thị dữ liệu 2 byte thành 4 số ở module 4 led 7 đoạn:

```
void xuat_4led_7doan_2so_4led (unsigned int8 bh,b1)
{
    xuat_4led_7doan_4so (ma7doan[bh/10%10],ma7doan[bh%10],
                          ma7doan[b1/10%10],ma7doan[b1%10]);
}
```

Hai byte cần hiển thị trong phạm vi từ 00 đến 99 sẽ được hiển thị trên 4 led, 1 số 2 led.

Bài tập 414. Hãy viết chương trình đếm sản phẩm hiển thị trên 4 led 7 đoạn dùng counter T0, giới hạn đếm từ 240 đến 255. Khi bằng 255 thì quay về 240.

Đúng ra là đếm từ 0 nhưng để đếm 255 thì bạn phải tạo tới 255 xung sẽ mất nhiều thời gian nên để nhanh chóng ta bắt đầu đếm từ 240 để thêm 15 xung nữa là đạt 255 và 1 xung nữa thì về lại 240. Mục đích là cho thấy mạch đếm ở chế độ 8 bit.

Lưu tên file là “BAI_414_DEM_XUNG_T0_240_255”.

Bài tập 415. Chương trình đếm sản phẩm hiển thị trên 4 led 7 đoạn dùng counter T0, giới hạn đếm từ 250 đến 300. Khi bằng 301 thì quay về 250. Giống bài 414 nhưng chế độ đếm 16 bit. Xóa bỏ bit **T0_8_BIT** trong lệnh khởi tạo T0.

Lưu tên file là “BAI_415_DEM_XUNG_T0”.

```
IF(NGAN==  
{  
    IF(TRAM==  
    {  
        IF(CHUC==  
    }  
}
```

Chú ý: Để kiểm tra giới hạn đếm tạo xung đếm thì nên bắt đầu giá trị đếm từ 250.

Bài tập 416. Chương trình đếm sản phẩm hiển thị trên 4 led 7 đoạn dùng counter T0, giới hạn đếm từ 0 đến 9999. Khi bằng 10000 thì quay về 1 và có **xóa số 0** vô nghĩa. Lưu tên file là “BAI_416_DEM_XUNG_T0”.

SETUP_TIMER_0(T0_OFF);

copy hàm chống dội
phím bài 323

Bài tập 417. Giống bài 413 nhưng có thêm 2 nút nhấn ON, OFF có chức năng cho phép đếm và **ngừng đếm**. Khi đang ngừng đếm thì 32 led đơn tắt, có xung cũng không đếm. Nhấn ON thì cho phép đếm và 32 led sáng.

Lưu tên file là “BAI_417_DEM_XUNG_T0_START_STOP_32LED”.

4.3 KẾT HỢP ĐIỀU KHIỂN 32 LED VÀ ĐÉM XUNG DÙNG COUNTER T0

Phần này thực hành kết hợp 2 module gồm module 32 led đơn và 4 led 7 đoạn.

Bài mẫu 421. Kết hợp bài 413 và bài 338 337

Vừa đếm sản phẩm vừa điều khiển 32 led đơn sáng nhiều chương trình.

Lưu tên file là “BAI_421_DEM_XUNG_T0_QUANG_CAO_32LED”.

- Mục đích: Biết cách lập trình kết hợp 2 module: 32 led đơn với 4 led 7 đoạn.
- Lưu đồ: sinh viên hãy tự viết.
- Chương trình:

```
#include <tv_pickit2_shift_1.c>  
unsigned int8 ttct=1, t0, t0_tam;  
#include <tv_pickit2_shift_32led_don.c>  
  
void phim_up()  
{  
    if (!input(up) && (ttct<7) )  
    {  
        delay_ms(10);  
        if (!input(up))  
        {  
            ttct++;  
            while(!input(up));  
        }  
    }  
}  
  
void phim_dw()
```

```

{
    if (!input(dw) && (ttct>1))
    {
        delay_ms(10);
        if (!input(dw))
        {
            ttct--;
            while(!input(dw));
        }
    }
}

void phim_clr()
{
    if (!input(clr)&&(ttct>1))    ttct=1;
}

void main()
{
    set_up_port_ic_chot();
    setup_timer_0 (t0_ext_l_to_h | t0_div_1|t0_8_bit);
    set_timer0(0);
    set_tris_b(0x3c);
    t0_tam = t0 = 0;
    xuat_4led_7doan_giaima_xoa_so0(t0);
    while(true)
    {
        t0=get_timer0();
        if (t0!=t0_tam)
        {
            t0_tam = t0;
            xuat_4led_7doan_giaima_xoa_so0(t0);
            if (t0>=101)    set_timer0(1); //số phía sau
        }
        if (ttct==1)    sang_tat_32led(10,0);
        if (ttct==2)    sang_tat_dan_pst_32led(10,0);
        if (ttct==3)    sang_tat_dan_tsp_32led(10,0);
        if (ttct==4)    sang_tat_dan ngoai_vao_32led(10,0);
        if (ttct==5)    sang_tat_dan_trong_ra_32led(10,0);
        if (ttct==6)    sang_don_pst_32led(10,0);
        if (ttct==7)    sang_don_tsp_32led(10,0);

        phim_dw();
        phim_up();
        phim_clr(); //xuat...4so(TT_CT, 0xFF, CH_T0, DV_T0);
    }
}

```

- d. Tiến hành biên dịch và nạp.
- e. Quan sát kết quả: vừa đếm sản phẩm vừa điều khiển 32 led chạy nhiều chương trình.
- f. Giải thích chương trình: tổng hợp 2 chương trình đã viết.

Chú ý: có sử dụng hàm xuất 4 led đoạn có luôn giải mã hiển thị và xóa số 0.

337 (không tự động) --> 338

~~Bài tập 422.~~ Kết hợp bài 413 và bài 326 nhưng chương trình chạy tự động.

Lưu tên file là “BAI_422_DEM_XUNG_T0_QUANG_CAO_32LED”.

copy bài 332 - ma trận phím

Bài tập 423. Hãy viết lưu đồ và chương trình kết hợp bàn phím ma trận với 4 led 7 đoạn có chức năng khi bắt đầu thì hiển thị 4 dấu chấm, nếu nhấn phím số nào thì mã 7 đoạn của phím đó dịch vào bên phải các dữ liệu trước đó dịch sang trái.

Lưu tên file là “BAI_423_KEY_4X4_4LED_DICH”.

trái - phải
phải - trái
4 led
2 led giữa

424. Hãy viết lưu đồ và chương trình giống bài 423 như chỉ cho dịch các số từ 0 đến 9. Phím C có chức năng xóa về lại từ đầu. Phím B có chức năng undo tối đa 4 cấp.

Lưu tên file là “BAI_424_KEY_4X4_4LED_DICH_UNDO4”.

4.4 CÁC CÂU HỎI ÔN TẬP

Câu 4-1. Hãy vẽ sơ đồ mạch giao tiếp 4 led quét với các IC 74HC595.

Câu 4-2. Hãy vẽ sơ đồ mạch giao tiếp IC 74HC595 của module 4 led với vi điều khiển và IC chốt.

Câu 4-3. Ở bài mẫu 401 bạn hãy cho biết khi thực hiện hàm gởi dữ liệu thì chúng sẽ xuất hiện ở đâu trong sơ đồ mạch phần cứng và có chức năng gì? Dòng điện chạy từ đâu đến đâu?

Câu 4-4. Hãy cho biết ưu điểm của các hàm 401, 402, 403 và 404.

Câu 4-5. Hãy cho biết các thông số của timer T0.

Câu 4-6. Hãy vẽ sơ đồ khói của timer T0 và giải thích chức năng các kí hiệu và các bit có trong sơ đồ.

Câu 4-7. Hãy cho biết các thanh ghi liên quan đến timer T0.

Câu 4-8. Hãy cho biết tên và chức năng từng bit của thanh ghi T0CON.

Câu 4-9. Hãy mở file thư viện 18f4550.h và xem các định nghĩa có liên quan đến các thông số của timer T0. Tại sao người ta định nghĩa hằng số T0_DIV_1 có giá trị là 8? Tương tự cho các hằng số khác.

Câu 4-10. Hãy viết lại hàm thư viện 405 và giải thích.

Câu 4-11. Hãy vẽ sơ đồ mạch tạo xung và giải thích nguyên lý tạo xung.

Câu 4-12. Hãy cho biết ưu điểm của bài 413.

Chương 5: THỰC HÀNH

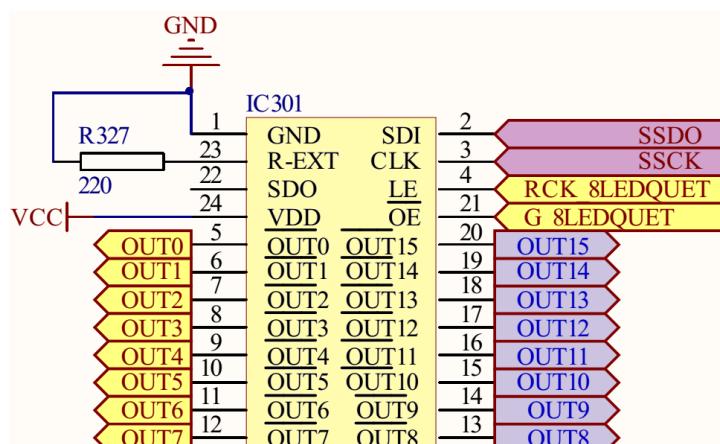
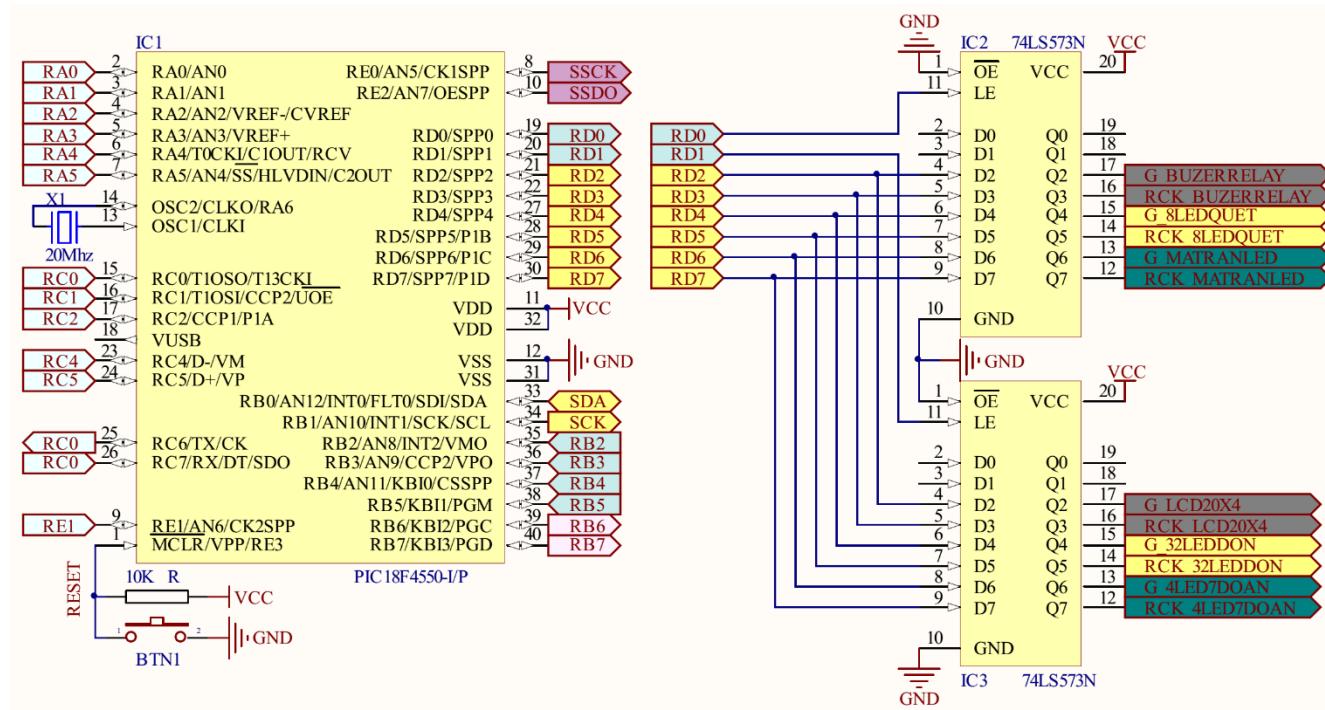
MODULE 3 – 8 LED 7 ĐOẠN ANODE CHUNG KẾT NỐI THEO PP QUÉTDÙNG THANH GHI DỊCH MBI5026

5.1 MẠCH ĐIỆN GIAO TIẾP VI ĐIỀU KHIỂN VỚI 8 LED 7 ĐOẠN QUÉT

5.1.1 Mạch điện giao tiếp vi điều khiển với module 8 led 7 đoạn

Phần này thực hành các bài giao tiếp với 8 led 7 đoạn theo phương pháp quét dùng thanh ghi dịch 16 bit MBI5026.

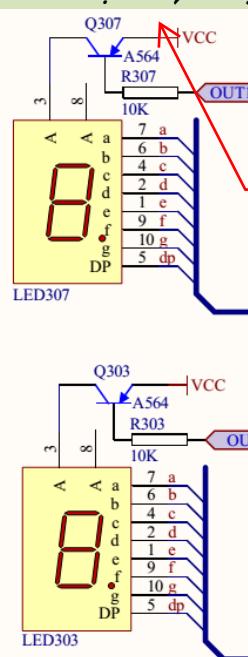
Mạch điện giao tiếp vi điều khiển với module 8 led 7 đoạn như hình 5-1.



Thanh ghi dịch 16 bit
= 2 IC dịch 8 bit 74HC595

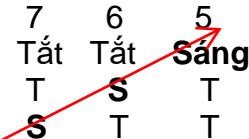
so sánh trang số 76, 2 thanh ghi dịch --> 8 led
Phương pháp **quét** (**Multiplexer - đa hợp**)

Đại Học Sư Phạm Kỹ Thuật



8 chân a,b,...,dp nối chung với nhau
chân cấp nguồn (Anode) được điều khiển riêng biệt

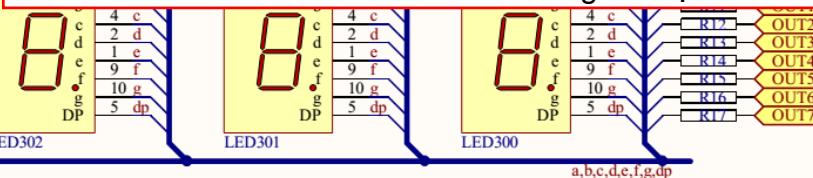
--> Khi điều khiển, tại 1 thời điểm chỉ có 1 led sáng (vì nối chung)



--> ta muốn thấy cùng 1 lúc --> thực hiện chu kỳ điều khiển này nhiều lần trong 1s (vài trăm, ngàn lần /1s)

--> mỗi led sẽ sáng tắt vài trăm, ngàn lần /1s

--> tính chất lưu ảnh của mắt --> led sáng liên tục



Hình 5-1. Sơ đồ nguyên lý giao tiếp vi điều khiển với module 8 led 7 đoạn.

Trong mạch sử dụng 1 IC thanh ghi MBI5026 để mở rộng port điều khiển 8 led 7 đoạn.

Phản vi điều khiển PIC giao tiếp với IC thanh ghi dịch

Có 4 tín hiệu điều khiển là:

- SSDO dùng để dịch dữ liệu nối tiếp.
- SSCK dùng để cấp xung clock.
- RCK_8LEDQUET dùng điều khiển nạp dữ liệu song song từ bên trong IC thanh ghi dịch ra bên ngoài để điều khiển led.
- G_8LEDQUET dùng để cho phép xuất dữ liệu.

Phản IC thanh ghi dịch giao tiếp với 8 led 7 đoạn quét

IC thanh ghi dịch MBI5026 có 16 bit ngõ ra kết nối với 8 đoạn của 8 led (các đoạn nối chung với nhau) và điều khiển 8 transistor để điều khiển led sáng hay tắt.

Số lượng byte dữ liệu điều khiển module 8 led 7 đoạn theo phương pháp quét là 2 byte: 1 byte cho mã đoạn và 1 byte mã quét điều khiển 8 transistor.

Byte mã quét sẽ có 1 bit bằng 0 và 7 bit còn lại bằng 1 để cho phép 1 led sáng và sau khi dịch ra thì xuất hiện ở 8 ngõ ra từ out8 đến out15 để điều khiển 1 transistor dẫn, 7 transistor còn lại tắt.

Byte mã 7 đoạn cần hiển thị sau khi dịch ra thì xuất hiện ở 8 ngõ ra từ out0 đến out7 để điều khiển 8 đoạn gồm a, b, c, d, e, f, g, dp tích cực mức thấp.

Kết hợp lại thì sẽ có 1 led sáng số thập phân.

Giải thích về dòng điện thì dòng điện sẽ chạy từ nguồn Vcc vào cực E của 1 transistor dẫn rồi tách làm 2 dòng: dòng thứ nhất chạy ra khỏi cực B chạy về IC MBI 5026 – dòng này gọi là

dòng phân cực, dòng thứ 2 chạy ra khỏi cực C chạy vào đầu anode chung của 1 led 7 đoạn và chạy ra các đoạn có mức tích cực là mức 0 và chạy về IC MBI5026.

Sau khi cho 1 led sáng trong 1 thời gian ngắn ta sẽ điều khiển led kế sáng và tiếp tục cho đến led thứ 8 rồi lặp lại led thứ 1.

Chu kỳ quét phải lớn hơn đáp ứng của mắt người để ta quan sát không thấy led chập chờn.

Bảng 5-1. Bảng dữ liệu điều khiển quét lần lượt các transistor.

TT led sáng	Mã quét 8 led Chỉ cho 1 transistor dẫn để 1 led sáng									Mã số hex
	7	6	5	4	3	2	1	0		
7	1	1	1	1	1	1	1	0		FE
6	1	1	1	1	1	1	0	1		FD
5	1	1	1	1	1	0	1	1		FB
4	1	1	1	1	0	1	1	1		F7
3	1	1	1	0	1	1	1	1		EF
2	1	1	0	1	1	1	1	1		DF
1	1	0	1	1	1	1	1	1		BF
0	0	1	1	1	1	1	1	1		7F

Khả năng mở rộng

Trong kit thực hành dùng 1 IC để điều khiển 8 led, sau này các ứng dụng thực tế cần nhiều led hơn thì bạn cứ kết nối thêm IC thanh ghi nối tiếp vào một cách dễ dàng. Nhưng do phương pháp quét nếu mở rộng nhiều và nếu chu kỳ quét lớn thì led sẽ mờ hay nháy nháy.

5.1.2 Các hàm điều khiển module 8 led 7 đoạn quét

Module 8 led 7 đoạn quét gồm các hàm xuất dữ liệu ra như sau:

Hàm thứ 501: xuất 2 byte ra module 8 led quét để hiển thị 1 led:

```
void xuat_8led_7doan_quet_1(unsigned int ma, so_hthi)
{
    xuat_1byte(~ma);
    xuat_1byte(~so_hthi);

    mo_ic_74573_b_thong_dl();
    output_high(rck_8ledquet);
    output_low(rck_8ledquet);
```

```

mo_8_led_quet;
chot_ic_74573_b_goi_du_lieu;
}

```

Do IC MBI5026 là thanh ghi dịch đảo nên khi xuất thì cả 2 byte đều phải đảo mức logic. Byte mã quét led xuất trước, byte hiển thị xuất sau. Các lệnh còn lại giống các hàm đã nêu.

Hàm thứ 502: xuất 2 byte ra module 8 led quét để tắt led:

```

void xuat_8led_7doan_quet_tat_led()
{
    xuat_8led_7doan_quet_1(0xff,0xff);
}

```

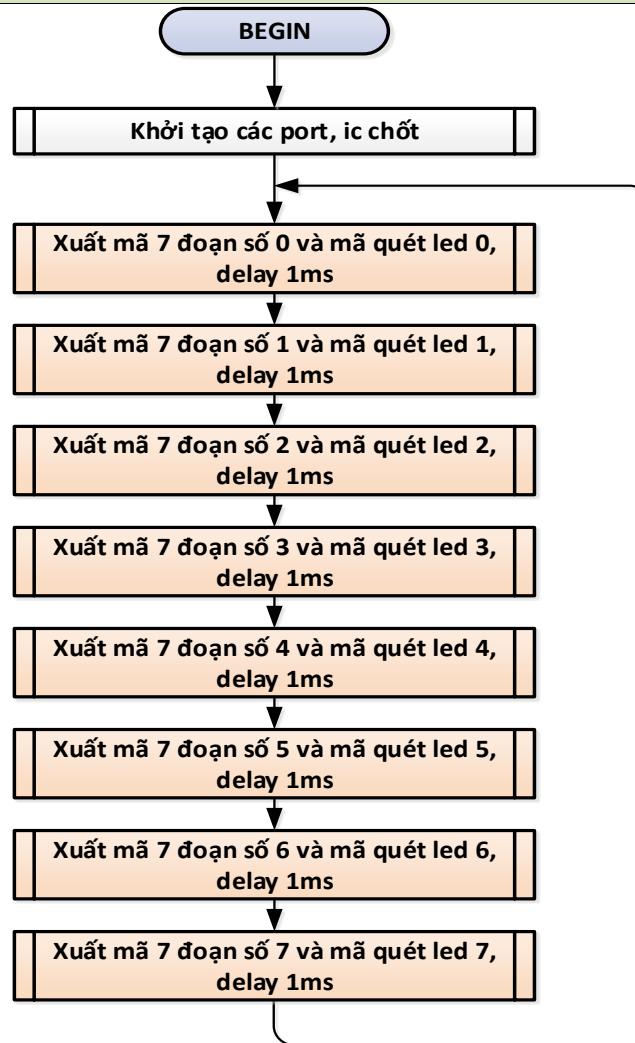
5.1.3 Các chương trình hiển thị 8 led 7 đoạn quét

Phần này thực hành các bài điều khiển 8 led 7 đoạn kết nối theo phương pháp quét để giúp bạn hiểu nguyên lý quét led 7 đoạn. Thấy được hiện tượng quét nhanh và chậm, biết cách làm cho vấn đề phức tạp trở nên đơn giản hơn.

Bài mẫu 501. Chương trình điều khiển 8 LED 7 đoạn hiển thị 8 số từ 7 đến 0.

Lưu tên file là “BAI_501_HIENTHI_8SO”.

- Mục đích: biết cách viết chương trình điều khiển led 7 đoạn theo phương pháp quét.
- Lưu đồ:



Hình 5-2. Lưu đồ điều khiển 8 led sáng từ 0 đến 7 trên module 8 led 7 đoạn.

c. Chương trình:

```

#include <tv_pickit2_shift_1.c>
unsigned int16 tg_delay;
void main()
{
    set_up_port_ic_chot();
    tg_delay=1;
    while(true)
    {
        xuat_8led_7doan_quet_1(0x7f, 0xc0); //0
        delay_ms(tg_delay);

        xuat_8led_7doan_quet_1(0xbff, 0xf9); //1
        delay_ms(tg_delay);

        xuat_8led_7doan_quet_1(0xdf, 0xa4); //2
        delay_ms(tg_delay);
    }
}
  
```

chọn led --> cấp
nguồn cho led --> cho
transistor dẫn (xuất
mức 0 - vì PNP)
0111 1111
(led phải)

mã 7 đoạn của số
đếm

```

xuat_8led_7doan_quet_1(0xef, 0xb0);           //3
delay_ms(tg_delay);

xuat_8led_7doan_quet_1(0xf7, 0x99);           //4
delay_ms(tg_delay);

xuat_8led_7doan_quet_1(0xfb, 0x92);           //5
delay_ms(tg_delay);

xuat_8led_7doan_quet_1(0xfd, 0x82);           //6
delay_ms(tg_delay);

xuat_8led_7doan_quet_1(0xfe, 0xf8);           //7
delay_ms(tg_delay);
}
}

```

- d. Tiến hành biên dịch và nạp.
- e. Quan sát kết quả: 8 led sẽ hiển thị 8 số từ 0 đến 7. Tiến hành thay đổi thông số thời gian delay (`tg_delay`) bằng 10, bằng 100 và bằng 1000 để thấy mỗi lần chỉ có 1 led sáng.
- f. Câu hỏi: Hãy cho biết thời gian sáng và thời gian tắt của mỗi led là bao nhiêu của mỗi lần quét?

Bài tập 502. Chương trình điều khiển hiển thị 2 số 1 và 0 trên 2 led ở vị trí bên phải.
Lưu tên file là “BAI_502_HIENTHI_10”.

Hãy so sánh cường độ sáng của bài này và bài 501.

Bài mẫu 503. Chương trình điều khiển 8 LED 7 đoạn hiển thị 8 số từ 7 đến 0, **cách 2**.
Lưu tên file là “BAI_503_HIENTHI_8SO_CACH2”.

- a. Mục đích: biết cách viết chương trình điều khiển led 7 đoạn theo phương pháp quét đơn giản hơn, dễ nhớ hơn, dễ sử dụng hơn.
- b. Lưu đồ: sinh viên hãy tự viết.
- c. Chương trình:

```

#include <tv_pickit2_shift_1.c>
unsigned int16 tg_delay;
void main()
{
    set_up_port_ic_chot();
    tg_delay=1;
    while(true)
    {
        xuat_8led_7doan_quet_2(0, 0xc0);           //0
        delay_ms(tg_delay);

        xuat_8led_7doan_quet_2(1, 0xf9);           //1
    }
}

```

thay mã 0xbf = số thứ
tự led

```

        delay_ms(tg_delay);

        xuat_8led_7doan_quet_2(2, 0xa4); //2
        delay_ms(tg_delay);

        xuat_8led_7doan_quet_2(3, 0xb0); //3
        delay_ms(tg_delay);

        xuat_8led_7doan_quet_2(4, 0x99); //4
        delay_ms(tg_delay);

        xuat_8led_7doan_quet_2(5, 0x92); //5
        delay_ms(tg_delay);

        xuat_8led_7doan_quet_2(6, 0x82); //6
        delay_ms(tg_delay);

        xuat_8led_7doan_quet_2(7, 0xf8); //7
        delay_ms(tg_delay);
    }
}

```

- d. Tiến hành biên dịch và nạp.
- e. Quan sát kết quả: 8 led sẽ hiển thị 8 số từ 0 đến 7.
- f. Giải thích chương trình: cách 2 sẽ giúp bạn được điều gì, hãy giải thích chương trình con xuất “`xuat_8led_7doan_quet_2(ttlled,dlht);`” và các chương trình con có liên quan trong thư viện.

Bài mẫu 504. Chương trình điều khiển 8 LED 7 đoạn hiển thị 8 số từ 7 đến 0, cách 3.

Lưu tên file là “BAI_504_HIENTHI_8SO_CACH3”.

- a. Mục đích: biết cách viết chương trình điều khiển led 7 đoạn theo phương pháp quét dạng vòng lặp và sử dụng mảng để lưu dữ liệu hiển thị cho 8 led, chương trình ngắn gọn hơn.
- b. Lưu đồ: sinh viên hãy tự viết.
- c. Chương trình:

```
#include <tv_pickit2_shift_1.c>
unsigned int8 led7dq[8]={0xc0,0xf9,0xa4,0xb0,0x99,0x92,0x82,0xf8};
unsigned int8 i;
void hien_thi_8led()
{
    for(i=0;i<8;i++)
    {
        xuat_8led_7doan_quet_2(i, led7dq[i]);
        delay_us(100);
        xuat_8led_7doan_quet_tat_led();
    }
}
```

vị trí led

mã 7 đoạn
của số đếm

- Quét led:
Xuất mã 7 đoạn
Cấp nguồn cho led
Delay (đảm bảo 1s đạt được vài trăm đến ngàn lần, kg được thực hiện nhanh quá --> led không đáp ứng kịp)
Tắt hết led (chống lem - do ảnh hưởng giá trị của led kế lén led trước)

```

}
void main()
{
    set_up_port_ic_chot();
    while(true)
    {
        hien_thi_8led();
    }
}

```

- d. Tiến hành biên dịch và nạp.
- e. Quan sát kết quả: 8 led sẽ hiển thị 8 số từ 0 đến 7.
- f. Giải thích chương trình: hãy giải thích chương trình, cách 3 sẽ giúp bạn được điều gì?

5.2 CÁC CHƯƠNG TRÌNH ĐÉM THỜI GIAN HIỂN THỊ 8 LED

Phần này thực hành các bài điều đếm thời gian nhưng hiển thị trên module 8 led 7 đoạn quét.

Sử dụng timer T1 định thời báo ngắt để đếm chính xác về thời gian.

5.2.1 Đếm giây dùng delay

Bài mẫu 511. Chương trình đếm giây từ 00 đến 59 hiển thị trên 2 led bên phải, **không cần chỉnh xác về thời gian.**

Lưu tên file là “BAI_511_DEM_GIAY”.

- a. Mục đích: biết cách viết chương trình điều khiển led 7 đoạn quét đếm giây.
- b. Lưu đồ: sinh viên hãy tự viết.
- c. Chương trình:

```

#include <tv_pickit2_shift_1.c>
signed int8      giay;
unsigned int16   i;
void hienthi_delay()
{
    for (i=0;i<200;i++)
    {
        xuat_8led_7doan_quet_2(0,ma7doan [giay %10]);
        delay_us(200);
        xuat_8led_7doan_quet_tat_led();
        xuat_8led_7doan_quet_2(1,ma7doan [giay/10]);
        delay_us(200);
        xuat_8led_7doan_quet_tat_led();
    }
}
void main()
{
    set_up_port_ic_chot();
    while(true)
    {
}

```

```

        for(giay =0; giay <60; giay++) hienthi_delay();
    }
}

```

- d. Tiến hành biên dịch và nạp.
- e. Quan sát kết quả: đếm từ 00 đến 59. Điều chỉnh giá trị của biến I trong chương trình hiển thị delay sẽ thay đổi thời gian đếm, giá trị nhỏ đếm nhanh, giá trị lớn đếm chậm.
- f. Giải thích chương trình:
- g. Câu hỏi: Hãy cho biết chức năng của hàm tắt led? Nếu bạn không biết thì hãy bỏ nó và quan sát led thật kỹ giữa 2 chương trình để tìm sự khác biệt và giải thích. Hãy cho biết nếu cũng yêu cầu này mà thực hiện ở module 4 led 7 đoạn trực tiếp thì như thế nào?

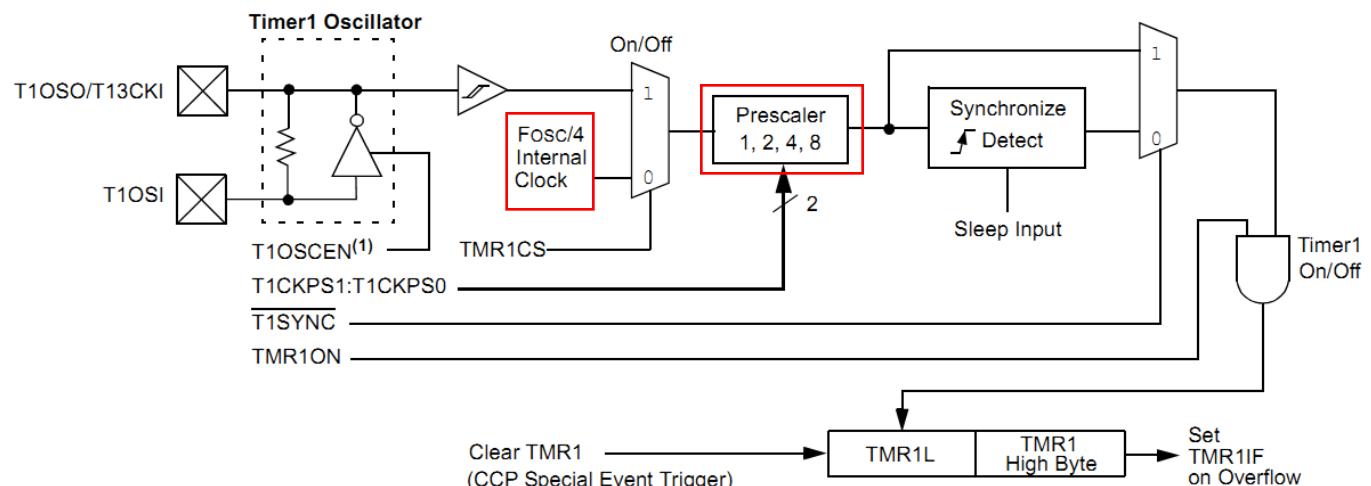
5.2.2 Đếm giây dùng timer T1 định thời

Ở bài đếm giây trên, nếu muốn đếm chính xác thì rất khó để tính toán:

- Thứ nhất ta không biết được thời gian thực hiện 1 lệnh trong ngôn ngữ C là bao nhiêu.
- Thứ hai là khi mở rộng đếm nhiều thông số như giờ phút giây thì các chu kỳ thực hiện không đồng đều nên dẫn đến sai số.

Để đếm chính xác thì ta giao công việc đó cho timer đếm xung nội với các chu xung nội là hằng số và ổn định, chính xác. Khi đó nhiệm vụ của vi điều khiển bây giờ là quét led và kiểm tra xem đủ thời gian 1 giây chưa, nếu chưa thì tiếp tục quét led, nếu đủ thì tăng giây và kiểm tra giới hạn.

Do có sử dụng timer T1 nên phần này có trình bày lại sơ đồ của timer T1 như hình 5-3:



Hình 5- 3. Sơ đồ khái niệm của timer T1.

Bài mẫu 512. Chương trình đếm giây chính xác dùng timer T1 báo ngắn.

Lưu tên file là “BAI_512_DEM_GIAY_CX”.

BÀI MẪU CHUẨN

Mục đích: biết sử dụng timer T1 định thời và ngắn để đếm giây chính xác hiển thị 8 led quét.

- b. Lưu đồ: sinh viên hãy tự viết.

c. Chương trình:

```

#include <tv_picket2_shift_1.c>
signed int8      giay, bdn;

#include "avr/io.h"
#include "avr/interrupt.h"

void interrupt_timer1()
{
    bdn++;
    set_timer1(3036);
}

void giao_ma_gan_cho_8led_quet()
{
    led_7dq[0] = ma7doan [giay %10];
    led_7dq[1] = ma7doan [giay/10];
}

void main()
{
    set_up_port_ic_chot();
    setup_timer_1(t1_internal | t1_div_by_8);
    set_timer1(3036);
    enable_interrupts(global);
    enable_interrupts(int_timer1);

    giay =0;
    bdn=0;
    while(true)
    {
        giao_ma_gan_cho_8led_quet();
        if (bdn<10) hien_thi_8led_7doan_quet();
        else
        {
            bdn = bdn-10;
            giay++;
            if (giay==60) giay =0;
        }
        giay--;
        if(giay==0) giay = 12;
    }
}

```

timer 1 tràn (16 bit) --> ngắt
số lớn nhất + 1
 $0-65535 + 1 = 65536$

Số xung cần đếm = TG cần có/T
 $= 1s/T = 625.000 xung > 65535 xung$
 $T = 1/(TA/4/8) = 1/625.000$
--> đếm nhiều lần: 62.500 xung x 10 lần
Số đếm bắt đầu = số tràn - số xung cần đếm
 $= 65536 - 62500 = 3036$

//khởi tạo ngắt timer 1

chưa đủ 10 lần (1s)

chỉ quét led để h.thị

đủ 1s (10 lần)

trường hợp timer đếm nhiều hơn 10 lần --> sẽ tính cho lần sau

bdn = 0;

d. Tiến hành biên dịch và nạp.

e. Quan sát kết quả: đếm giây chính xác từ 00 đến 59.

f. Giải thích chương trình:

Khởi tạo timer1 đếm xung nội có tần số **20MHz qua bộ chia 4 còn 5MHz**. Sử dụng bộ chia trước có hệ số chia là 8 nên xung vào bộ đếm với tần số còn lại là **5MHz/8 = 0.625MHz** hay bằng **625,000Hz**.

Tần số 625,000Hz có nghĩa là trong thời gian 1 giây sẽ có 625,000 xung hay đếm 625,000 xung sẽ tương đương 1 giây.

Sử dụng timer 1 để đếm nhưng timer T1 chỉ có 16 bit và giới hạn đếm lớn nhất là 65536 xung thì tràn, vậy ta thiết lập timer đếm 62500 thì tràn và cho đếm 10 lần.

Mỗi lần tràn thì tăng biến đếm ngắn và nếu biến đếm ngắn bằng 10 thì được 625,000 xung sẽ tương đương giây.

Thiết lập giá trị đếm bắt đầu cho timer1 là 3036 nên timer T1 đếm 62500 xung sẽ tràn và báo ngắn.

Khi ngắn xảy ra thì giá trị của timer T1 được khởi tạo lại và đếm tiếp rồi báo lần tràn tiếp theo.

Muốn thay đổi thời gian đếm cho nhanh để kiểm tra kết quả chương trình thì có thể thay đổi giá trị “T1_DIV_BY_8” bằng “T1_DIV_BY_1” hoặc thay đổi thêm giá trị biến đếm ngắn BDN bằng 2 và hiệu chỉnh lệnh BDN = BDN - 2.

- g. Câu hỏi: Tại sao các chu kỳ thực hiện không đồng đều? Hãy giải thích tại sao không so sánh BDN bằng 10 và khi bằng thì gán bằng 0 mà lại làm phức tạp như trong chương trình.

5.2.3 Sử dụng thêm các hàm điều khiển 8 led quét

Ở mục 5.1.2 đã giới thiệu 1 số hàm điều khiển 8 led 7 đoạn quét cơ bản để bạn hiểu và thực hành các bài, sau khi đã làm quen thì từ nay về sau để tiện lợi và nhanh chóng, không mất thời gian để viết lại thì trong thư viện đã viết sẵn 1 số hàm như sau:

Hàm thứ 503: xuất 2 byte ra module 8 led quét để hiển thị 1 led theo thứ tự:

```
const unsigned char ttledquet[8]=
{0x7f,0xbff,0xdff,0xef,0xff7,0xfb,0xfd,0xfe};
void xuat_8led_7doan_quet_2(unsigned int thutuled,so_hthi)
{
    xuat_1byte(~(ttledquet[thutuled]));
    xuat_1byte(~so_hthi);

    mo_ic_74573_b_thong_dl();
    output_high(rck_8ledquet);
    output_low(rck_8ledquet);
    mo_8_led_quet;
    chot_ic_74573_b_goi_du_lieu;
}
```

Để sáng 1 led 7 đoạn thì phải gởi mã quét cho transistor tương ứng dẫn (logic 0), 7 transistor còn lại tắt (logic 1), các mã quét cho ở bảng và đã viết trong chương trình. Các mã quét này khó nhớ và dễ nhầm lẫn khi đánh máy ví dụ mã quét là 0xFE có 1 bit bằng 0 để cho phép 1 led sáng nhưng bạn đánh nhầm thành 0xEE thì khi đó có 2 bit bằng 0 sẽ làm 2 led sáng cùng số và ảnh hưởng đến led khác.

Để không còn nhầm lẫn thì ta quy định 8 led quét có số thứ tự từ 0 đến 7 từ phải sang. Khi muốn sáng led thứ 3 thì bạn gọi hàm trên ra và byte thứ nhất là số 3, byte còn lại là mã 7 đoạn. Trong chương trình sẽ lấy byte mã quét tương ứng số led thứ 3 xuất ra thanh ghi dịch cùng với mã 7 đoạn sẽ làm led sáng tương ứng.

Vậy cái khó và phức tạp đã trở nên đơn giản. Hàm này sử dụng ở bài 503.

Hàm thứ 504: quét 8 led sáng với 8 byte dữ liệu có kiểm tra:

```
unsigned char led_7dq[8]={0xff,0xff,0xff,0xff,0xff,0xff,0xff,0xff} ;
unsigned int8 tt8led=0;
void hien_thi_8led_7doan_quet()
{
    for(tt8led=0;tt8led<8;tt8led++)
        if (led_7dq[tt8led] !=0xff)
        {
            xuat_8led_7doan_quet_2(tt8led, led_7dq[tt8led]);
            delay_us(100);
            xuat_8led_7doan_quet_tat_led();
        }
}
```

Để tiện lợi thì trong thư viện đã xây dựng hàm hiển thị 8 số trên 8 led quét.

Hàng đầu tiên định nghĩa mảng **led_7dq** có 8 byte và gán bằng 0xFF tương ứng với 8 led 7 đoạn.

Khi bạn muốn dùng hàm này thì phải gán giá trị cần hiển thị cho led tương ứng từ 0 đến 7.

Khi gọi hàm này thì vòng lặp for thực hiện 8 lần nhưng chương trình sẽ kiểm tra nếu mã gởi ra led để hiển thị khác 0xFF thì mới tiết hành xuất dữ liệu ra để hiển thị, nếu bằng 0xFF thì quét led tiếp theo.

Việc so sánh này sẽ đáp ứng các yêu cầu như ta đếm giây chỉ cần hiển thị 2 led thì chương trình này chỉ quét và hiển thị đúng 2 led, thời gian sẽ ngắn lại, led sáng rõ đẹp hơn.

Hàm này có thể hiển thị từ 1 led đến 8 led, rất linh động.

Từ nay về sau bạn muốn sử dụng thì chỉ cần gán dữ liệu và gọi hàm là xong.

Hàm thứ 505: quét 8 led sáng với 8 byte dữ liệu – không kiểm tra:

```
void hien_thi_8led_7doan_quet_all()
{
    for(tt8led=0;tt8led<8;tt8led++)
    {
        xuat_8led_7doan_quet_2(tt8led, led_7dq[tt8led]);
        delay_us(100);
        xuat_8led_7doan_quet_tat_led();
    }
}
```

Hàm này khác hàm trên là quét hiển thị hết cho dù dữ liệu là 0xFF.

Hàm trên có 1 khuyết điểm ở 1 số bài ví dụ bài đếm từ 0 đến 99999 có xóa số 0 vô nghĩa.

Khi đếm 1 số từ 0 đến 9 thì chương trình quét có 1 led sáng nên led sáng rõ, 100% chu kỳ.

Khi đếm 2 số từ 10 đến 99 thì chương trình quét có 2 led sáng nên led sáng giảm phân nữa, thời gian sáng cho mỗi led là 50% chu kỳ.

Khi đếm 3 số từ 100 đến 999 thì chương trình quét có 3 led sáng, thời gian sáng cho mỗi led còn 33% chu kỳ.

Khi đếm 4 số từ 1000 đến 9999 thì chương trình quét có 4 led sáng, thời gian sáng cho mỗi led còn 25% chu kỳ.

Khi đếm 5 số từ 10000 đến 99999 thì chương trình quét có 4 led sáng, thời gian sáng cho mỗi led còn 20% chu kỳ.

Vậy thời gian sáng không đồng đều khi quan sát rất khó chịu nên ta sử dụng hàm quét sáng hết cho chúng đồng đều.

Bài tập 513. Hãy viết chương trình đếm phút giây chính xác.

Lưu tên file là “BAI_513_DEM_PHUT_GIAY”.

Bài tập 514. Hãy viết chương trình đếm giờ phút giây chính xác.

Lưu tên file là “BAI_514_DEM_GIO_PHUT_GIAY”.

Bài mẫu 515. Hãy viết chương trình đếm giờ phút giây chính xác có thêm 3 nút UP, DW và MOD để chỉnh thời gian. Nút MOD chọn chế độ chỉnh giây, nhấn nữa chỉnh phút, nhấn nữa thì chỉnh giờ, nhấn nữa thì không cho chỉnh. Nhấn UP thì chỉnh tăng, nhấn DW thì chỉnh giảm.

Lưu tên file là “BAI_515_GIO_PHUT_GIAY_KEY”.

- Mục đích: biết viết chương trình đồng hồ số chính xác hiển thị 8 led quét và cách lập trình để chỉnh thời gian.
- Lưu đồ: sinh viên hãy tự viết.
- Chương trình:

```
#include      <tv_pickit2_shift_1.c>
signed int8    gio,phut,giay,bdn;
unsigned int8   gia_tri_mod,i;

#int_timer1
void interrupt_timer1()
{
    bdn++;
    set_timer1(3036);
}
void giao_ma_gan_cho_8led_quet()
{
    led_7dq[0]= ma7doan [giay %10];
    led_7dq[1]= ma7doan [giay /10];
    led_7dq[3]= ma7doan [phut %10];
    led_7dq[4]= ma7doan [phut /10];
    led_7dq[6]= ma7doan [gio %10];
    led_7dq[7]= ma7doan [gio /10];

    if      (gia_tri_mod==1)  led_7dq[6]=led_7dq[6] & 0x7f;
```

```

else if (gia_tri_mod==2) led_7dq[3]=led_7dq[3] & 0x7f;
else if (gia_tri_mod==3) led_7dq[0]=led_7dq[0] & 0x7f;

}

void delay_hienthi()
{
    for(i=0;i<100;i++) hien_thi_8led_7doan_quet();
}

void phim_mod()
{
    if (!input(mod))
    {
        delay_hienthi();
        if(!input(mod))
        {
            gia_tri_mod++;
            if (gia_tri_mod>=4) gia_tri_mod=0;
            gai_ma_gan_cho_8led_quet();
            do{ hien_thi_8led_7doan_quet(); } while(!input(mod));
        }
    }
}

void phim_up()
{
    if (!input(up))
    {
        delay_hienthi();
        if(!input(up))
        {
            switch (gia_tri_mod)
            {
                case 1: if (gio==23) gio=0;
                          else gio++;
                          break;
                case 2: if (phut==59) phut=0;
                          else phut++;
                          break;
                case 3: if (giay==59) giay=0;
                          else giay++;
                          break;
                default: break;
            }
            gai_ma_gan_cho_8led_quet();
            do{ hien_thi_8led_7doan_quet(); } while(!input(up));
        }
    }
}

```

hiển thị dấu chấm khi điều chỉnh thời gian

chờ nhã phím có quét led để led không tắt khi chưa nhã phím

```

}

void phim_dw()
{
    if (!input(dw))
    {
        delay_hienthi();
        if(!input(dw))
        {
            switch (gia_tri_mod)
            {
                case 1: if (gio==0)      gio=23;
                          else          gio--;
                          break;
                case 2: if (phut==0)     phut=59;
                          else          phut--;
                          break;
                case 3: if (giay==0)    giay=59;
                          else          giay--;
                          break;
                default: break;
            }
            gai_ma_gan_cho_8led_quet();
            do{ hien_thi_8led_7doan_quet(); } while(!input(dw));
        }
    }
}

//xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
void main()
{
    set_up_port_ic_chot();
    setup_timer_1(t1_internal | t1_div_by_8);
    set_timer1(3036);
    enable_interrupts(global);
    enable_interrupts(int_timer1);
    giay =0x00; phut=0;    gio=0;    gia_tri_mod=0;
    while(true)
    {
        gai_ma_gan_cho_8led_quet();
        if(bdn<10)
        {
            hien_thi_8led_7doan_quet();
            phim_mod();
            phim_up();
            phim_dw();
        }
        else
        {
            bdn = bdn-10;
        }
    }
}

```

```

        giay++;
        if (giay==60)
        {
            giay =0;
            phut++;
            if (phut==60)
            {
                phut=0;
                gio++;
                if (gio==24) gio =0;
            }
        }
    }
}

```

- d. Tiến hành biên dịch và nạp.
- e. Quan sát kết quả: đếm giờ phút giây hiển thị trên 8 led, có 2 led tắt. Khi nhấn phím MOD thì dấu chấm giờ sáng để báo hiệu cho phép chỉnh giờ, nhấn MOD lần nữa thì chỉnh phút, lần nữa chỉnh giây và lần nữa thì thoát khỏi chế độ chỉnh.
Khi cho phép chỉnh đối tượng nào thì nếu nhấn UP thì đối tượng đó tăng lên 1, nhấn DW thì giảm.
- f. Giải thích chương trình: trong bài này có sử dụng chương trình hiển thị 8 led đã viết sẵn trong thư viện có tên là: HIEN THI 8LED_7DOAN_QUET, chương trình con này giống bài 504. Dữ liệu hiển thị được khai báo nằm trong mảng có tên là LED_7DQ có 8 phần tử tương ứng với 8 led 7 đoạn. Bạn muốn sử dụng chương trình con này thì phải gán dữ liệu hiển thị tương ứng.
Trong chương trình con GIAI_MA_GAN_CHO_8LED_QUET xử lý giải mã, dấu chấm gán cho mảng.
Có 3 chương trình kiểm tra phím nhấn UP, DW và MOD: kiểm tra có nhấn phím thì xử lý và chống dội. Trong chương trình chống dội ở chương 3, ta là chờ nhả phím ra thì mới thoát và dùng lệnh while, ở bài này ta dùng hàm lệnh do while để khi chưa nhả phím thì nó sẽ gọi chương trình hiển thị để cho 8 led sáng liên tục.
- h. Câu hỏi: Hãy cho biết nếu dùng lệnh while chờ nhả phím thì sẽ xuất hiện hiện tượng gì khi nhấn phím mà không nhả phím.

Bài mẫu 516. Cải tiến bài 515.

Lưu tên file là “BAI_516_GIO_PHUT_GIAY_KEY_CAITIEN”.

- Mục đích: Tối ưu hơn chương trình đã viết.
- Lưu đồ: sinh viên hãy tự viết.
- Chương trình:

```

#include      <tv_pickit2_shift_1.c>
signed int8    gio,phut,giay,bdn,gia_tri_mod;

int_timer1

```

```

void interrupt_timer1()
{
    bdn++;
    set_timer1(3036);
}

void giao_ma_gan_cho_8led_quet()
{
    led_7dq[0]= ma7doan [giay %10];
    led_7dq[1]= ma7doan [giay /10];
    led_7dq[3]= ma7doan [phut %10];
    led_7dq[4]= ma7doan [phut /10];
    led_7dq[6]= ma7doan [gio %10];
    led_7dq[7]= ma7doan [gio /10];

    if      (gia_tri_mod==1) led_7dq[6]=led_7dq[6] & 0x7f;
    else if (gia_tri_mod==2) led_7dq[3]=led_7dq[3] & 0x7f;
    else if (gia_tri_mod==3) led_7dq[0]=led_7dq[0] & 0x7f;
}

void phim_mod()
{
    if (phim_bt1(200))
    {
        gia_tri_mod++;
        if (gia_tri_mod>=4) gia_tri_mod=0;
    }
}

void phim_up()
{
    if (phim_bt0(200))
    {
        switch (gia_tri_mod)
        {
            case 1:   if (gio==23)     gio=0;
                       else           gio++;
                       break;
            case 2:   if (phut==59)   phut=0;
                       else           phut++;
                       break;
            case 3:   if (giay==59)   giay=0;
                       else           giay++;
                       break;
            default: break;
        }
    }
}

//xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
void phim_dw()
{
}

```

```

if (phim_bt2(200))
{
    switch (gia_tri_mod)
    {
        case 1: if (gio==0) gio=23;
                  else gio--;
                  break;
        case 2: if (phut==0) phut=59;
                  else phut--;
                  break;
        case 3: if (giay==0) giay=59;
                  else giay--;
                  break;
        default: break;
    }
}
//xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
void main()
{
    set_up_port_ic_chot();
    setup_timer_1(t1_internal | t1_div_by_8);
    set_timer1(3036);
    enable_interrupts(global);
    enable_interrupts(int_timer1);
    giay =0x00;
    phut=0;
    gio=0;
    gia_tri_mod=0;
    while(true)
    {
        giao_ma_gan_cho_8led_quet();
        if (bdn<10)
        {
            hien_thi_8led_7doan_quet();
            phim_mod(); phim_up(); phim_dw();
        }
        else
        {
            bdn = bdn-10;
            giay++;
            if (giay==60)
            {
                giay =0;
                phut++;
                if (phut==60)
                {
                    phut=0;
                    gio++;
                    if (gio==24) gio =0;
                }
            }
        }
    }
}

```

```
    }  
    }  
}
```

- d. Quan sát kết quả: giống như bài 515.
 - e. Giải thích chương trình:
 - f. Câu hỏi: Hãy tìm sự khác nhau giữa hai bài 515 và 516 để biết các cải tiến là gì?

Bài mẫu 517. Cải tiến bài 516.

Lưu tên file là “BAI_517_GIO_PHUT_GIAY_KEY_CAITIEN”.

- a. Mục đích: Tối ưu hóa nữa chương trình đã viết.
 - b. Lưu đồ: sinh viên hãy tự viết.
 - c. Chương trình:

```

#include <tv_pickit2_shift_1.c>
signed int8 gio,phut,giay,bdn,gia_tri_mod,dem_tg_exit=0;
#define _timer1
void interrupt_timer1()
{
    bdn++;
    set_timer1(3036);
}

void giao_ma_gan_cho_8led_quet()
{
    led_7dq[0]= ma7doan [giay %10];
    led_7dq[1]= ma7doan [giay /10];
    led_7dq[3]= ma7doan [phut %10];
    led_7dq[4]= ma7doan [phut /10];
    led_7dq[6]= ma7doan [gio %10];
    led_7dq[7]= ma7doan [gio /10];
    if (gia_tri_mod==1) led_7dq[6]=led_7dq[6]
    else if (gia_tri_mod==2) led_7dq[3]=led_7dq[3]
    else if (gia_tri_mod==3) led_7dq[0]=led_7dq[0]
}

void tat_2led_chinh()
{
    if (gia_tri_mod==1) {led_7dq[7]=0xff; led_7dq[6]=0xff;}
    else if (gia_tri_mod==2) {led_7dq[4]=0xff; led_7dq[3]=0xff;}
    else if (gia_tri_mod==3) {led_7dq[1]=0xff; led_7dq[0]=0xff;}
}

//xxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
void phim_mod()
{
}

```

led_7dq[0] - led ngoài cùng bên phải
led_7dq[0] = ma7doan[T0%10];
...

hiển thị dấu chấm
0111 1111

& 0x7f;
& 0x7f;
& 0x7f;

tắt led, nháy nháy

```

if (phim_bt1(200))
{
    gia_tri_mod++;
    if (gia_tri_mod>=4) gia_tri_mod=0;
    dem_tg_exit=0;
}
//xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
void phim_up()
{
    if (phim_bt0(200))
    {
        dem_tg_exit=0;
        switch (gia_tri_mod)
        {
            case 1: if (gio==23) gio=0;
                      else gio++;
                      break;
            case 2: if (phut==59) phut=0;
                      else phut++;
                      break;
            case 3: if (giay==59) giay=0;
                      else giay++;
                      break;
            default: break;
        }
        giao_ma_gan_cho_8led_quet();
    }
}
//xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
void phim_dw()
{
    if (phim_bt2(200))
    {
        dem_tg_exit=0;
        switch (gia_tri_mod)
        {
            case 1: if (gio==0) gio=23;
                      else gio--;
                      break;
            case 2: if (phut==0) phut=59;
                      else phut--;
                      break;
            case 3: if (giay==0) giay=59;
                      else giay--;
                      break;
            default: break;
        }
        giao_ma_gan_cho_8led_quet();
    }
}

```

```

}

//xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

void main()
{
    set_up_port_ic_chot();
    setup_timer_1(t1_internal | t1_div_by_8);
    set_timer1(3036);
    enable_interrupts(global);
    enable_interrupts(int_timer1);
    giay =0;
    phut=0;
    gio=0;
    gia_tri_mod=0;
    gai_ma_gan_cho_8led_quet();
    while(true)
    {
        if      (bdn<10)
        {
            if (gia_tri_mod!=0)
            {
                if ((bdn==0)&&(input(bt0))&& (input(bt2)))
                    tat_2led_chinh();
                else if (bdn==5)  gai_ma_gan_cho_8led_quet();
            }
            hien_thi_8led_7doan_quet_all();
            phim_mod();
            phim_up();
            phim_dw();
        }
        else
        {
            bdn = bdn-10;
            dem_tg_exit++;
            if (dem_tg_exit==20)  gia_tri_mod=0;
            giay++;
            if (giay==60)
            {
                giay =0;
                phut++;
                if (phut==60)
                {
                    phut=0;
                    gio++;
                    if (gio==24) gio =0;
                }
            }
            gai_ma_gan_cho_8led_quet();
        }
    }
}

```

xử lý nhập nháy led,
xử lý phần mã, phần giải mã

0,5s: bdn = 0-4
0,5s: bdn = 5-9

ta không chỉnh tg
trong vòng 20s thì sẽ
thoát chế độ

- d. Tiến hành biên dịch và nạp.
 - e. Quan sát kết quả:
 - f. Giải thích chương trình:
 - g. Câu hỏi: Hãy tìm sự khác nhau giữa hai bài 516 và 517 để biết các cải tiến là gì?

Bài tập 518. Hãy viết chương trình đếm xung ngoại dùng counter T0 hiển thị kết quả đếm trên 8 led quét, giới hạn đếm từ 00 đến 99. Có nút clear khi nhấn thì xóa về 00, có nút start_stop.

Lưu tên file là “BAI_518_DEM_XUNG_T0_SSC”.

5.3 KẾT HỢP 3 MODULE

Phần này thực hành kết hợp 3 module gồm module 32 led đơn, 4 led 7 đoạn và 8 led 7 đoạn quét.

Bài mẫu 521. Kết hợp bài 517 và bài 422 để thực hiện chức năng: vừa đếm thời gian giờ phút giây hiển thị 8 led 7 đoạn quét, vừa đếm sản phẩm hiển thị ở module 4 led và điều khiển 32 led đơn sáng nhiều chương trình.

Lưu tên file là “BAI_521_3MODULE_123”.

- a. Mục đích: Biết cách lập trình kết hợp 3 module: 32 led đơn với 4 led 7 đoạn và 8 led quét.
 - b. Lưu đồ: sinh viên hãy tự viết.
 - c. Chương trình:

```
#include      <tv_pickit2_shift_1.c>
signed int8    gio,phut,giay,bdn,gia_tri_mod,dem_tg_exit=0;
unsigned int8   t0,t0_tam;
void delay_quet_8led(unsigned int8 dl);
#include      <tv_pickit2_shift_32led_don.c>

#int_timer1
void interrupt_timer1()
{
    bdn++;
    set_timer1(3036);
}
void gai_ma_gan_cho_8led_quet()
{
    led_7dq[0]= ma7doan [giay %10];
    led_7dq[1]= ma7doan [giay /10];
    led_7dq[3]= ma7doan [phut %10];
    led_7dq[4]= ma7doan [phut /10];
    led_7dq[6]= ma7doan [gio %10];
    led_7dq[7]= ma7doan [gio /10];
}
void tat_2led_chinh()
```

```

{
    if      (gia_tri_mod==1)    {led_7dq[7]=0xff; led_7dq[6]=0xff;}
    else if (gia_tri_mod==2)    {led_7dq[4]=0xff; led_7dq[3]=0xff;}
    else if (gia_tri_mod==3)    {led_7dq[1]=0xff; led_7dq[0]=0xff;}
}

void phim_mod()
{
    if (phim_bt1(50))
    {
        gia_tri_mod++;
        if (gia_tri_mod>=4) gia_tri_mod=0;
        dem_tg_exit=0;
    }
}

void phim_up()
{
    if (phim_bt0(20))
    {
        dem_tg_exit=0;
        switch (gia_tri_mod)
        {
            case 1: if (gio==23) gio=0;
                      else gio++;
                      break;
            case 2: if (phut==59) phut=0;
                      else phut++;
                      break;
            case 3: if (giay==59) giay=0;
                      else giay++;
                      break;
            default: break;
        }
        giao_ma_gan_cho_8led_quet();
    }
}

void phim_dw()
{
    if (phim_bt2(20))
    {
        dem_tg_exit=0;
        switch (gia_tri_mod)
        {
            case 1: if (gio==0) gio=23;
                      else gio--;
                      break;
            case 2: if (phut==0) phut=59;
                      else phut--;
        }
    }
}

```

```

        break;
    case 3: if (giay==0) giay=59;
              else      giay--;
              break;
    default: break;
}
gai_ma_gan_cho_8led_quet();
}

void main()
{
    set_up_port_ic_chot();
    setup_timer_1(t1_internal | t1_div_by_8);
    set_timer1(3036);
    enable_interrupts(global);
    enable_interrupts(int_timer1);
    bdn=0;
    giay =0x00;
    phut=0;
    gio=0;
    gia_tri_mod=0;

    setup_timer_0 (t0_ext_l_to_h | t0_div_1|t0_8_bit);
    set_timer0(0);
    t0_tam = t0 = 0;
    xuat_4led_7doan_giaima_xoa_so0(t0);
    gai_ma_gan_cho_8led_quet();
    while(true)
    {
        if(bdn<10)
        {
            if (gia_tri_mod!=0)
            {
                if ((bdn==0)&&(input(bt0))&& (input(bt2)))
                    tat_2led_chinh();
                else if (bdn==5) gai_ma_gan_cho_8led_quet();
            }
            hien_thi_8led_7doan_quet_all();
            phim_mod();
            phim_up();
            phim_dw();
            t0=get_timer0();
            if (t0!=t0_tam)
            {
                t0_tam = t0;
                xuat_4led_7doan_giaima_xoa_so0(t0);
                if (t0>=101) set_timer0(1);
            }
            if (ttct_td==1) sang_tat_32led(2,1);
        }
    }
}

```

xem trong thư viện 32
led đơn
Tr. 64 - 66

```

        if (ttct_td==2) sang_tat_dan_pst_32led(5,1);
        if (ttct_td==3) sang_tat_dan_tsp_32led(5,1);
        if (ttct_td==4) sang_tat_dan ngoai_vao_32led(5,1);
        if (ttct_td==5) sang_tat_dan_trong_ra_32led(5,1);
        if (ttct_td==6) sang_don_pst_32led(5,1);
        if (ttct_td==7) diem_sang_dich_trai_mat_dan_32led(5,1);
        if (ttct_td==8) sang_don_tsp_32led(5,1);
        if (ttct_td==9) diem_sang_dich_phai_mat_dan_32led(5,1);

        if (ttct_td==10)
            sang_tat_dan_trai_sang_phai_2x16led(5,1);
        if (ttct_td==11)
            sang_tat_dan_phai_sang_trai_2x16led(5,1);
        if (ttct_td==12) diem_sang_di_chuyen_pst_32led(5,1);
        if (ttct_td==13) diem_sang_di_chuyen_tsp_32led(5,1);
        if (ttct_td==14) sang_don_tnt_32led(5,1);
        if (ttct_td==15) sang_don_ttr_32led(5,1);
        if (ttct_td>15) ttct_td=1;
    }
    else
    {
        bdn = bdn-10;
        dem_tg_exit++;
        if (dem_tg_exit==20) gia_tri_mod=0;
        giay++;
        if (giay==60)
        {
            giay =0;
            phut++;
            if (phut==60)
            {
                phut=0;
                gio++;
                if (gio==24) gio =0;
            }
        }
        giao_ma_gan_cho_8led_quet();
    }
}
}

```

- d. Tiến hành biên dịch và nạp.
- e. Quan sát kết quả: đồng hồ ở 8 led 7 đoạn, đếm sản phẩm và điều khiển 32 led chạy nhiều chương trình.
- f. Giải thích chương trình: hãy giải thích chương trình.

5.4 CÁC CÂU HỎI ÔN TẬP

Câu 5-1. Hãy vẽ sơ đồ mạch giao tiếp 8 led quét với IC MBI5026.

- Câu 5-2.** Hãy vẽ sơ đồ mạch giao tiếp IC MBI5026 của module 8 led với vi điều khiển và IC chốt.
- Câu 5-3.** Hãy giải thích 2 dữ liệu trong bài 501 là 0×FE và 0xC0 khi thực hiện hàm gởi dữ liệu thì chúng sẽ xuất hiện ở đâu trong sơ đồ mạch phần cứng và có chức năng gì. Dòng điện chạy từ đâu đến đâu?
- Câu 5-4.** Hãy cho biết ưu điểm của bài 503, bài 504.
- Câu 5-5.** Hãy so sánh bài đếm giây 511 và 404.
- Câu 5-6.** Hãy vẽ lưu đồ bài đếm giây 511.
- Câu 5-7.** Hãy vẽ lưu đồ đếm giờ phút giây dựa vào bài 511. Và cho biết tại sao cách này không chính xác.
- Câu 5-8.** Hãy vẽ lưu đồ bài đếm giây 512. Và cho biết tại sao cách này chính xác.
- Câu 5-9.** Hãy trình bày cách tính toán thời gian delay 1 giây dùng timer T1.
- Câu 5-10.** Hãy vẽ lưu đồ bài đồng hồ số 515.
- Câu 5-11.** Hãy vẽ lưu đồ bài đồng hồ số 516.
- Câu 5-12.** Hãy vẽ lưu đồ bài đồng hồ số 517.

Chương 6: THỰC HÀNH

MODULE 4 – LCD 20×4, GLCD 128×64

6.1 LÝ THUYẾT LCD

6.1.1 Giới thiệu LCD

Ở các phần giao tiếp với led 7 đoạn có hạn chế vì chỉ hiển thị được các số từ 0 đến 9 hoặc số hex từ 0 đến F – không thể nào hiển thị được các thông tin kí tự khác, nhưng chúng sẽ được hiển thị đầy đủ trên LCD.

LCD có rất nhiều dạng phân biệt theo kích thước từ vài kí tự đến hàng chục kí tự, từ 1 hàng đến vài chục hàng. Ví dụ LCD 16×2 có nghĩa là có 2 hàng, mỗi hàng có 16 kí tự. LCD 20×4 có nghĩa là có 4 hàng, mỗi hàng có 20 kí tự.

LCD 40×4 như hình 6-1:



Hình 6-1. Hình ảnh mặt trước của LCD.

6.1.2 Sơ đồ chân của LCD

LCD có nhiều loại và số chân của chúng cũng khác nhau nhưng có 2 loại phổ biến là loại 14 chân và loại 16 chân. Sự khác nhau là các chân nguồn cung cấp cho đèn nền, còn các chân điều khiển thì không thay đổi, khi sử dụng loại LCD nào thì phải tra datasheet của chúng để biết rõ các chân.

Sơ đồ chân của LCD như bảng 6-1.

Bảng 6-1. Các chân của LCD.

Chân số	Tên chân	Input/output	Chức năng tín hiệu
1	VSS	Power	GND
2	VDD	Power	+5V
3	VO	Analog	Contrast Control
4	RS	Input	Register Select. H: data signal, L: instruction signal
5	R/W	Input	Read/Write. H: read mode, L: write mode
6	E	Input	Enable (strobe)
7	D0	I/O	Data (LSB)

8	D1	I/O	Data
9	D2	I/O	Data
10	D3	I/O	Data
11	D4	I/O	Data
12	D5	I/O	Data
13	D6	I/O	Data
14	D7	I/O	Data (MSB)
15	LED_A	input	Backlight Anode
16	LED_K	input	Backlight Cathode

Trong 16 chân của LCD được chia ra làm 4 dạng tín hiệu như sau:

Các chân cấp nguồn: Chân số 1 là chân nối mass (0V), chân thứ 2 là Vdd nối với nguồn +5V. Chân thứ 3 dùng để chỉnh contrast thường nối với biến trở, chỉnh cho đèn khi thấy được kí tự thì ngừng, trong bộ thực hành thì đã chỉnh rồi.

Các chân điều khiển: Chân số 4 là chân RS dùng để điều khiển lựa chọn thanh ghi. Chân R/W dùng để điều khiển quá trình đọc và ghi. Chân E là chân cho phép dạng xung chốt.

Các chân dữ liệu D7-D0: Chân số 7 đến chân số 14 là 8 chân dùng để trao đổi dữ liệu giữa thiết bị điều khiển và LCD.

Các chân LED_A và LED_K: Chân số 15, 16 là 2 chân dùng để cấp nguồn cho đèn nền để có thể nhìn thấy vào ban đêm.

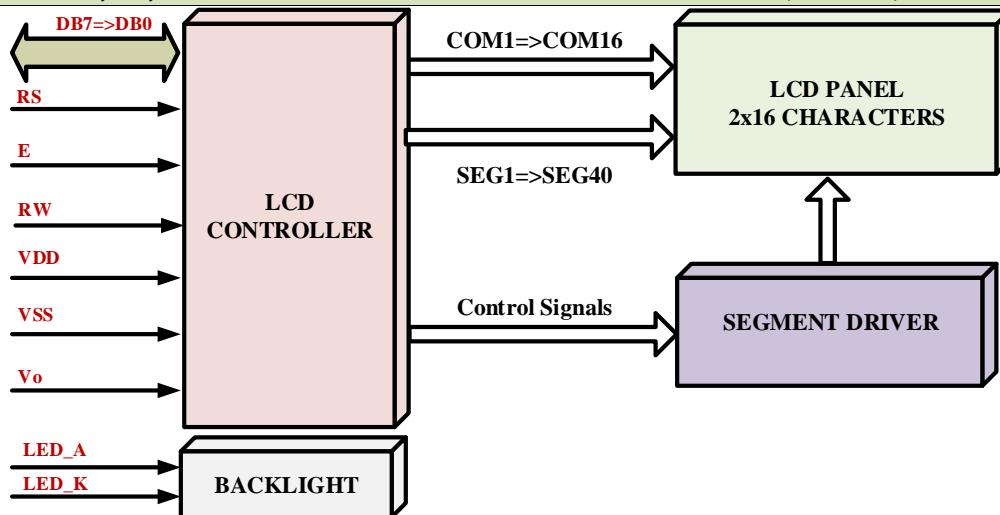
6.1.3 Bộ điều khiển LCD và các vùng nhớ

Để điều khiển LCD thì có các IC chuyên dùng được tích hợp bên dưới LCD có mã số 447801 đến các IC 447809 như trong hình 6-2.



Hình 6-2. Hình ảnh mặt sau của LCD.

Sơ đồ khối của bộ điều khiển LCD như hình sau:

**Hình 6-3. Sơ đồ khái niệm của bộ điều khiển LCD**

Sơ đồ khái niệm gồm có 4 phần: bộ điều khiển LCD, bảng kí tự LCD, bộ thúc tín hiệu các đoạn, đèn nền.

Bộ điều khiển LCD có ba vùng nhớ nội, mỗi vùng có chức năng riêng. Bộ điều khiển phải khởi động trước khi truy cập bất kỳ vùng nhớ nào.

a. Bộ nhớ DDRAM

Bộ nhớ chứa dữ liệu để hiển thị (Display Data RAM: DDRAM) lưu trữ những mã kí tự để hiển thị lên màn hình. Mã kí tự lưu trữ trong vùng DDRAM sẽ tham chiếu với từng bitmap kí tự được lưu trữ trong CGROM đã được định nghĩa trước hoặc đặt trong vùng do người sử dụng định nghĩa.

b. Bộ phát kí tự ROM - CGROM

Bộ phát kí tự ROM (Character Generator ROM: CGROM) chứa các kiểu bitmap cho mỗi kí tự được định nghĩa trước mà LCD có thể hiển thị, như được trình bày bảng mã ASCII.

Mã kí tự lưu trong DDRAM cho mỗi vùng kí tự sẽ được tham chiếu đến một vị trí trong CGROM.

Ví dụ, mã kí tự số hex 0x53 lưu trong DDRAM được chuyển sang dạng nhị phân 4 bit cao là DB[7:4] = “0101” và 4 bit thấp là DB[3:0] = “0011” chính là kí tự chữ ‘S’ sẽ hiển thị trên màn hình LCD.

c. Bộ phát kí tự RAM – CGRAM

Bộ phát kí tự RAM (Character Generator RAM: CG RAM) cung cấp vùng nhớ để tạo ra 8 kí tự tùy ý. Mỗi kí tự gồm 5 cột và 8 hàng.

6.1.4 Các lệnh điều khiển LCD

Tập lệnh của LCD

Bảng 6-2. Các lệnh điều khiển LCD.

LỆNH	RS	RW	D7	D6	D5	D4	D3	D2	D1	D0	Mô tả	clock
------	----	----	----	----	----	----	----	----	----	----	-------	-------

(1) NOP	0	0	0	0	0	0	0	0	0	0	No operation	0
(2) Clear display	0	0	0	0	0	0	0	0	0	1	Write "20H" to DDRAM and set DDRAM address counter to "00H".	1.52ms
(3) Cursor home	0	0	0	0	0	0	0	0	1	0	Sets address counter to zero, returns shifted display to original position. DDRAM contents remain unchanged.	39μs
(4) Entry mode set	0	0	0	0	0	0	0	1	I/D	S	Sets cursor move direction, and specifies automatic shift.	39μs
(5) Display control	0	0	0	0	0	0	1	D	C	B	Turns display (D), cursor on/off (C) or cursor blinking (B).	39μs
(6) Cursor display shift	0	0	0	0	0	1	S/C	R/L	0	0	Set cursor moving and display shift control bit, and the direction, without changing DDRAM data.	39μs
(7) Function set	0	0	0	0	1	DL	N	F	0	0	Set interface data length (DL : 4-bit/8-bit), numbers of display line (N : 1-line/2-line), display font type(F : 5 X 8 dots/ 5 X 11 dots)	39μs
(8) Set CGRAM addr	0	0	0	1	AC5	AC4	AC3	AC2	AC1	AC0	Set CGRAM address in address counter	39μs

(9) Set DDRAM addr	0	0	1	AC6	AC5	AC4	AC3	AC2	AC1	AC0	Set DDRAM address in address counter	39μs
(10) Read Busy Flag and Address	0	1	BF	AC6	AC5	AC4	AC3	AC2	AC1	AC0	Whether during internal operation or not can be known by reading BF. The contents of address counter can also be read.	0μs
(11) Write data to RAM	1	0	D7	D6	D5	D4	D3	D2	D1	D0	Write data to CGRAM or DDRAM	43μs
(12) Read data from RAM	1	1	D7	D6	D5	D4	D3	D2	D1	D0	Read data from CGRAM or DDRAM	43μs

Lệnh thiết chức năng “Function set”: lệnh này dùng để thiết lập chức năng giao tiếp.

Bảng 6-3. Mã lệnh Function set.

LỆNH	RS	RW	D7	D6	D5	D4	D3	D2	D1	D0	Số hex
(7) Function set	0	0	0	0	1	DL	N	F	0	0	
			0	0	1	1	1	0	0	0	0x38

Bit DL (data length) = 1 thì cho phép giao tiếp 8 đường data D7 ÷ D0, nếu bằng 0 thì cho phép giao tiếp 4 đường D7 ÷ D4.

Bit N (number of line) = 1 thì cho phép hiển thị 2 hàng, nếu bằng 0 thì cho phép hiển thị 1 hàng.

Bit F (font) = 1 thì cho phép hiển thị với ma trận 5×8 , nếu bằng 0 thì cho phép hiển thị với ma trận 5×11 .

Các bit cao còn lại là hàng số không đổi.

Trong thư viện của LCD sẽ định nghĩa lệnh này như sau:

#DEFINE FUNCTION_SET 0x38

Lệnh điều khiển hiển thị “Display Control”:

Bảng 6-4. Mã lệnh Display control.

LỆNH	RS	RW	D7	D6	D5	D4	D3	D2	D1	D0	Số hex
(5) Display control	0	0	0	0	0	0	1	D	C	B	

				0	0	0	0	1	1	0	0	0x0C
--	--	--	--	---	---	---	---	---	---	---	---	------

Bit D: cho phép LCD hiển thị thì D = 1, không cho hiển thị thì bit D = 0.

Bit C: cho phép con trỏ hiển thị thì C = 1, không cho hiển thị con trỏ thì bit C = 0.

Bit B: cho phép con trỏ nhấp nháy thì B = 1, không cho con trỏ nhấp nháy thì bit B = 0.

Với các bit như trên thì để hiển thị phải cho D = 1, 2 bit còn lại tùy chọn, trong thư viện thì cho 2 bit đều bằng 0, không cho phép mở con trỏ và nhấp nháy, nếu bạn không thích thì chỉnh lại.

#DEFINE LCD_DISPLAY_CONTROL 0x0C

Lệnh xoá màn hình “Clear Display”:

Bảng 6-5. Mã lệnh Clear Display.

LỆNH	RS	RW	D7	D6	D5	D4	D3	D2	D1	D0	Số hex
(2) Clear display	0	0	0	0	0	0	0	0	0	1	0x01

Khi thực hiện lệnh này, vi điều khiển của LCD sẽ ghi kí tự khoảng trắng “ ” vào tất cả các vùng nhớ hiển thị và bộ đếm địa chỉ được xoá về 0. Vì phải ghi 128 kí tự vào 128 byte ô nhớ DDRAM nên mất 1,52ms. Khá nhiều so với các lệnh khác.

Lệnh thiết lập lối vào “Entry mode set”:

Bảng 6-6. Mã lệnh entry mode.

LỆNH	RS	RW	D7	D6	D5	D4	D3	D2	D1	D0	Số hex
(4)Entry mode set	0	0	0	0	0	0	0	1	I/D	S	
			0	0	0	0	0	1	1	0	0x06

Lệnh này dùng để thiết lập lối vào cho các kí tự hiển thị.

Bit I/D = 1 thì con trỏ tự động tăng lên 1 mỗi khi có 1 byte dữ liệu ghi vào bộ hiển thị, khi I/D = 0 thì con trỏ sẽ tự động giảm đi 1 mỗi khi có 1 byte dữ liệu ghi vào bộ hiển thị.

Bit S = 1 thì cho phép dịch chuyển dữ liệu mỗi khi nhận 1 byte hiển thị.

Trong thư viện có định nghĩa lệnh này như sau:

#DEFINE LCD_ENTRY_MODE 0x06

Lệnh thiết lập lối vào “Cursor/Display shift”:

Bảng 6-7. Mã lệnh entry mode.

LỆNH	RS	RW	D7	D6	D5	D4	D3	D2	D1	D0	Số hex
(6) Cursor display shift	0	0	0	0	0	1	S/C	R/L	0	0	
Cho phép dịch trái			0	0	0	1	1	0	0	0	0x18
Cho phép dịch phải			0	0	0	1	1	1	0	0	0x1C

Lệnh này dùng để điều khiển dịch màn hình và di chuyển con trỏ:

Bit SC: SC = 1 cho phép dịch chuyển, SC = 0 thì không cho phép.

Bit RL xác định hướng dịch chuyển: RL = 1 thì dịch phải, RL = 0 thì dịch trái. Nội dung bộ nhớ DDRAM vẫn không đổi.

Vậy khi cho phép dịch thì có 2 tùy chọn: dịch trái và dịch phải.

Trong thư viện thì có định nghĩa lệnh này như sau:

```
#DEFINE LCD_SHIFT_LEFT      0x18
#DEFINE LCD_SHIFT_RIGHT     0x1C
```

Lệnh di chuyển con trỏ về đầu màn hình “Cursor Home”: khi thực hiện lệnh này thì bộ đếm địa chỉ được xoá về 0, phần hiển thị trở về vị trí gốc đã bị dịch trước đó. Nội dung bộ nhớ RAM hiển thị DDRAM không bị thay đổi.

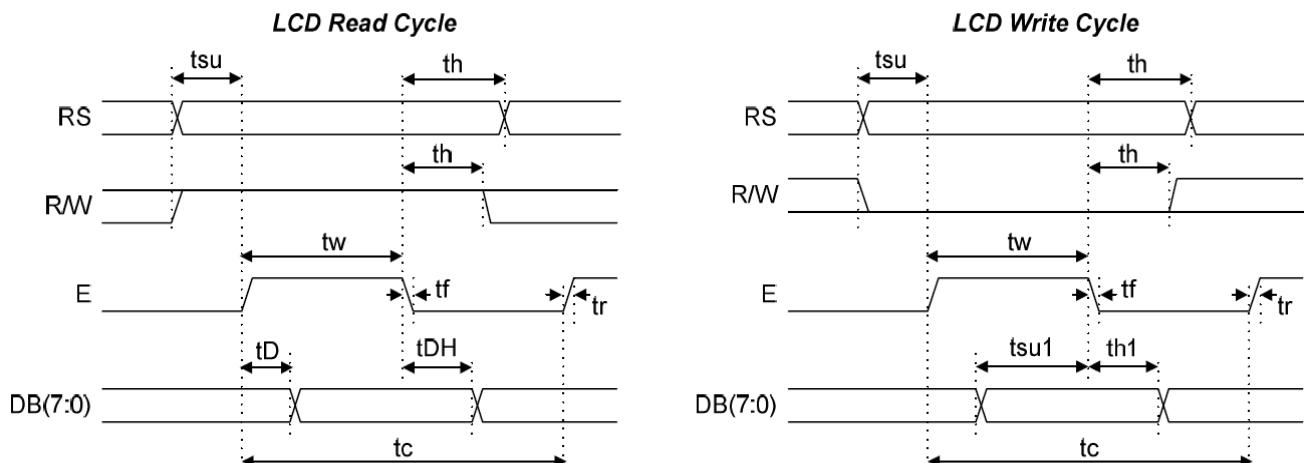
Lệnh thiết lập địa chỉ cho bộ nhớ RAM phát kí tự “Set CGRAM Addr”: lệnh này dùng để thiết lập địa chỉ cho bộ nhớ RAM phát kí tự.

Lệnh thiết lập địa chỉ cho bộ nhớ RAM hiển thị “Set DDRAM Addr”: lệnh này dùng để thiết lập địa chỉ cho bộ nhớ RAM lưu trữ các dữ liệu hiển thị.

Hai lệnh cuối cùng là lệnh đọc và lệnh ghi dữ liệu LCD.

6.1.5 Các hoạt động đọc ghi LCD

Có 2 chế độ hoạt động là đọc và ghi LCD, hai dạng sóng tương ứng như hình 6-4.



Hoạt động đọc:

- Điều khiển tín hiệu RS.
- Điều khiển tín hiệu R/W lên mức 1.
- Điều khiển tín hiệu E lên mức cao để cho phép.
- Đọc dữ liệu từ bus dữ liệu DB7÷DB0 (data bus).
- Điều khiển tín hiệu E về mức thấp.

Hoạt động ghi:

- Điều khiển tín hiệu RS.

- Điều khiển tín hiệu R/W xuống mức 0.
- Điều khiển tín hiệu E lên mức cao để cho phép.
- Xuất dữ liệu ra bus dữ liệu DB7÷DB0 (data bus).
- Điều khiển tín hiệu E về mức thấp.

Các thông số thời gian cho ở bảng 6-8.

Bảng 6-8. Cho biết các thông số thời gian của LCD.

Parameter	Symbol	Min	Max	Unit	Test Pin
Enable cycle time	tc	500		ns	E
Enable High pulse width	tw	220		ns	E
Enable rise/fall time	tr, tf		25	ns	E
RS, R/W setup time	tsu	40		ns	RS, R/W
RS, R/W hold time	th	10		ns	RS, R/W
Read data output delay	tD	60	120	ns	DB0-DB7
Read data hold time	tDH	20		ns	DB0-DB7
Write data setup time	tsu1	40		ns	DB0-DB7
Write data hold time	th1	10		ns	DB0-DB7

6.1.1 Mã ASCII

LCD sử dụng mã ASCII để hiển thị các ký tự, các số, các ký hiệu, ... có tổng cộng là 256 ký tự. Mã ASCII như ở bảng 6-9.

Trong các bài thực hành hiển thị các ký tự và các bài đếm thì ta hiển thị các con số từ 0 đến 9 và tra trong bảng thì ta thấy các con số sẽ có mã từ 0x30 đến 0x39.

Để hiển thị trên Led 7 đoạn các giá trị đếm từ giá trị số nhị phân thì ta phải chuyển sang số BCD hàng đơn vị, hàng chục, hàng trăm, ... rồi lấy mã 7 đoạn tương ứng.

Để hiển thị trên LCD từ giá trị số nhị phân thì ta phải chuyển sang số BCD rồi sau đó cộng thêm với 0x30 mới hiển thị được trên LCD.

Bảng 6-9. Bảng mã ASCII.

0011 0000 = 0x30

	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	Ø P									— Ø Ø Ø Ø Ø Ø			
xxxx0001	(2)			! 1 A Q a q						¤ ¤ ¤ ¤ ¤ ¤							
xxxx0010	(3)			" 2 B R b r						‘ ‘ ‘ ‘ ‘ ‘							
xxxx0011	(4)			# 3 C S c s						’ ’ ’ ’ ’ ’							
xxxx0100	(5)			\$ 4 D T d t						、 、 、 、 、 、							
xxxx0101	(6)			% 5 E U e u						・ 、 、 、 、 、 、							
xxxx0110	(7)			& 6 F U f v						ヲ カ ニ ヽ ヽ							
xxxx0111	(8)			* 7 G W g w						ア キ フ ラ ク							
xxxx1000	(1)			(8 H X h x						イ ク ホ リ ク							
xxxx1001	(2)) 9 I Y i y						カ テ ノ ル ル							
xxxx1010	(3)			* ; J Z j z						エ コ ハ レ ジ							
xxxx1011	(4)			+ ; K C k <						オ サ ハ ボ							
xxxx1100	(5)			, < L ¥ l l						フ ハ フ ハ フ							
xxxx1101	(6)			- = M J m >						ユ ハ ハ ハ ハ							
xxxx1110	(7)			. > N ^ n +						ミ ハ ハ ハ ハ							
xxxx1111	(8)			/ ? O _ o +						ウ ハ ハ ハ ハ							

6.1.2 Vùng nhớ hiển thị DDRAM

Trong chip điều khiển LCD, đã tích hợp đầy đủ bảng mã ASCII, muốn hiển thị kí tự nào thì ta tiến hành gởi mã của kí tự đó ra vùng nhớ DDRAM, khi đó vi điều khiển của LCD sẽ lấy mã ma trận của kí tự đó đem ra hiển thị trên màn hình.

Trong tập lệnh LCD thì có lệnh thiết lập vùng nhớ để hiển thị là lệnh thứ 9 Set DDRAM addr. Vùng nhớ này chứa mã để hiển thị, trong lệnh có 7 bit địa chỉ AC6 ÷ AC0

Bảng 6-10. Địa chỉ vùng nhớ hiển thị DDRAM.

	Set DDRAM address in address counter								Không tính D7 = 1	Tính D7 = 1
	D7	D6	D5	D4	D3	D2	D1	D0		
(9) Set DDRAM addr	1	AC6	AC5	AC4	AC3	AC2	AC1	AC0	Số hex	Số hex
Địa chỉ đầu	1	0	0	0	0	0	0	0	0x00	0x80

	1	0	0	0	0	0	1	0x01	0x81
	1					...			
Địa chỉ cuối cùng	1	1	1	1	1	1	1	0x7F	0xFF

Nếu tính luôn bit lệnh D7 = 1 thì địa chỉ vùng nhớ này từ 0x80 đến 0xFF có tổng cộng 128 byte có thể chứa được 128 ký tự để hiển thị.

Vùng nhớ này được chia ra làm 2:

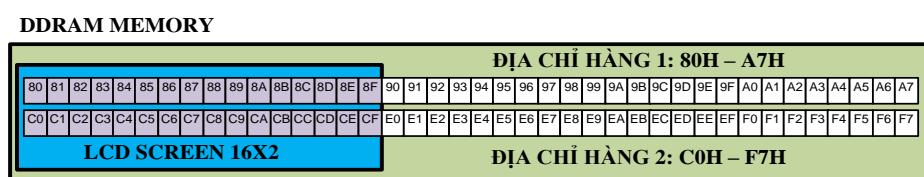
- Vùng 1: các ô nhớ có địa chỉ từ 0x80 đến 0xBF sẽ lưu các ký tự hiển thị hàng 1 của LCD.
- Vùng 2: các ô nhớ có địa chỉ từ 0xC0 đến 0xFF sẽ lưu các ký tự hiển thị hàng 2 của LCD.

Các nhà chế tạo LCD sản xuất ra nhiều loại LCD khác nhau, ví dụ LCD 16×2 có 2 hàng và mỗi hàng hiển thị được 16 ký tự, khi đó:

Hàng 1 chỉ hiển thị được 16 ký tự có địa chỉ từ 0x80 đến 0x8F, từ 0x90 đến 0xBF nằm ẩn bên trong, muốn hiển thị thì phải dùng lệnh dịch.

Hàng 2 chỉ hiển thị được 16 ký tự có địa chỉ từ 0xC0 đến 0xCF, từ 0xB0 đến 0xFF nằm ẩn bên trong, muốn hiển thị thì phải dùng lệnh dịch.

Xem hình sau:



Hình 6-5. Hiển thị 32 ký tự ở LCD 16×2

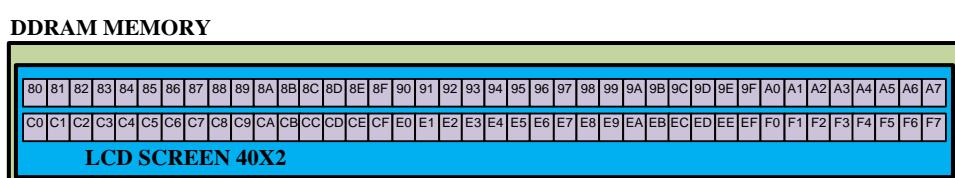
Bộ thí nghiệm vi điều khiển dùng LCD 20×4, có 4 hàng và mỗi hàng 20 ký tự. Nhưng nếu theo bố trí bộ nhớ thì thông số 20x4 sẽ trở thành 40×2 có nghĩa là có 2 hàng và mỗi hàng có 40 ký tự.

Hàng 1 chỉ hiển thị được 40 ký tự có địa chỉ từ 0x80 đến 0xA7, từ 0xA8 đến 0xBF nằm ẩn bên trong, muốn hiển thị thì phải dùng lệnh dịch.

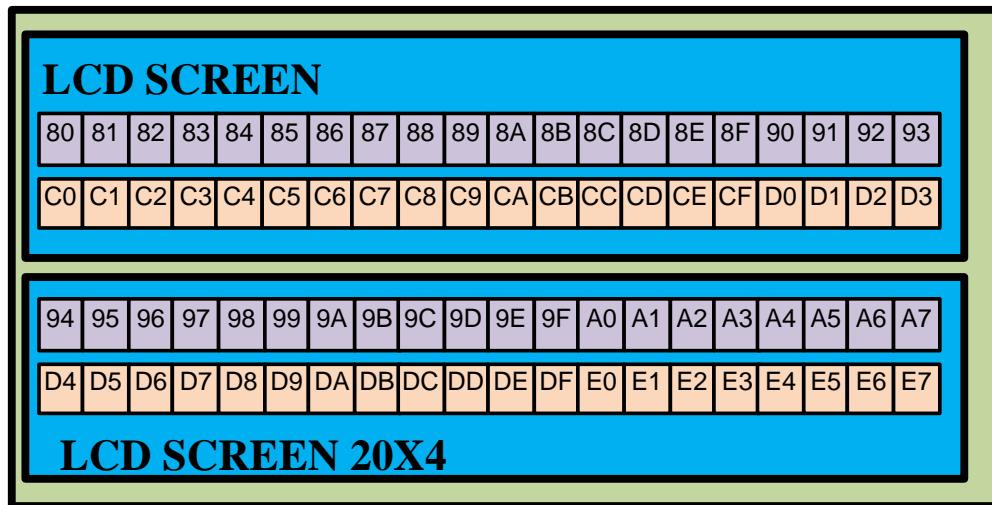
Hàng 2 chỉ hiển thị được 16 ký tự có địa chỉ từ 0xC0 đến 0xE7, từ 0xE8 đến 0xFF nằm ẩn bên trong, muốn hiển thị thì phải dùng lệnh dịch.

Khi chế tạo 2 hàng và mỗi hàng 40 ký tự thì LCD sẽ dài và mỏng nên nhà chế tạo bố trí lại thành 20×4, khi đó bạn sẽ thấy hàng 1 và hàng 3, hàng 2 và hàng 4 liên tục nhau.

Xem hình sau:



DDRAM MEMORY



Hình 6-6. Hiển thị 80 ký tự ở LCD 20×4.

Địa chỉ chi tiết từng ký tự trên từng hàng cho ở bảng sau.

Bảng 6-11. Địa chỉ từng ký tự của LCD 20×4.

ĐC	80	81	82	83	84	85	86	87	88	89	8A	8B	8C	8D	8E	8F	90	91	92	93
(Hàng 1)																				
ĐC	C0	C1	C2	C3	C4	C5	C6	C7	C8	C9	CA	CB	CC	CD	CE	CF	D0	D1	D2	D3
(Hàng 2)																				
ĐC	94	95	96	97	98	99	9A	9B	9C	9D	9E	9F	A0	A1	A2	A3	A4	A5	A6	A7
(Hàng 3)																				
ĐC	D4	D5	D6	D7	D8	D9	DA	DB	DC	DD	DE	DF	E0	E1	E2	E3	E4	E5	E6	E7
(Hàng 4)																				

Trong LCD còn vùng nhớ CGRAM được trình bày ở phần sau.

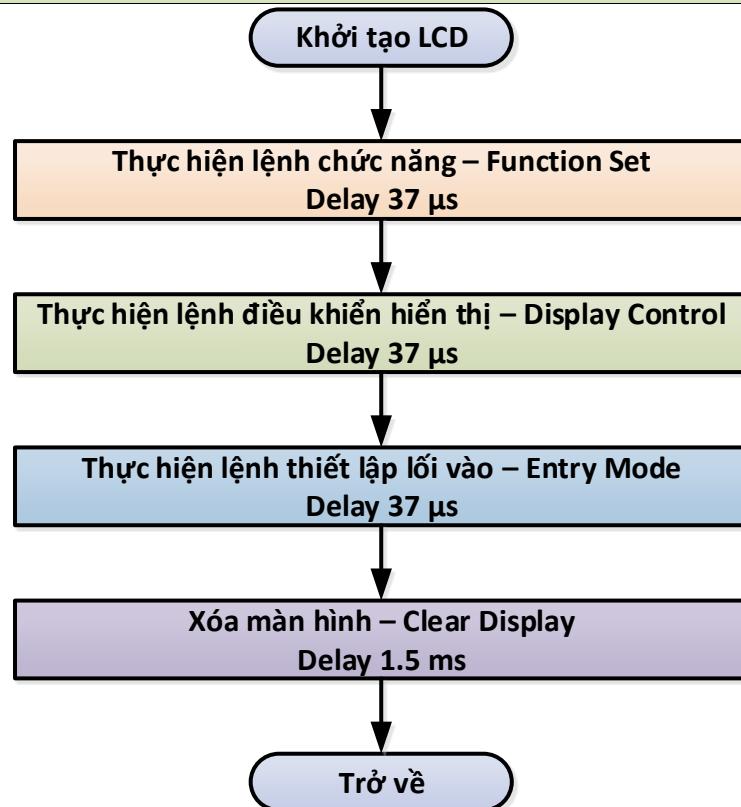
6.1.3 Lưu đồ khởi tạo LCD

Khi sử dụng LCD để hiển thị các ký tự thì ta phải khởi tạo LCD. Trình tự khởi tạo LCD được thể hiện qua hình 6-7.

Sau khi cấp điện thì phải đợi 20ms sau đó thì tiến hành thực hiện lệnh khởi tạo Function Set – là lệnh định cấu hình giao tiếp bus dữ liệu bao nhiêu bit nên phải thực hiện đầu tiên.

Tương tự cho các lệnh còn lại và chú ý thời gian, đây là trình tự do nhà sản xuất cung cấp.

Sau khi khởi tạo xong thì mới gởi dữ liệu ra LCD để hiển thị.

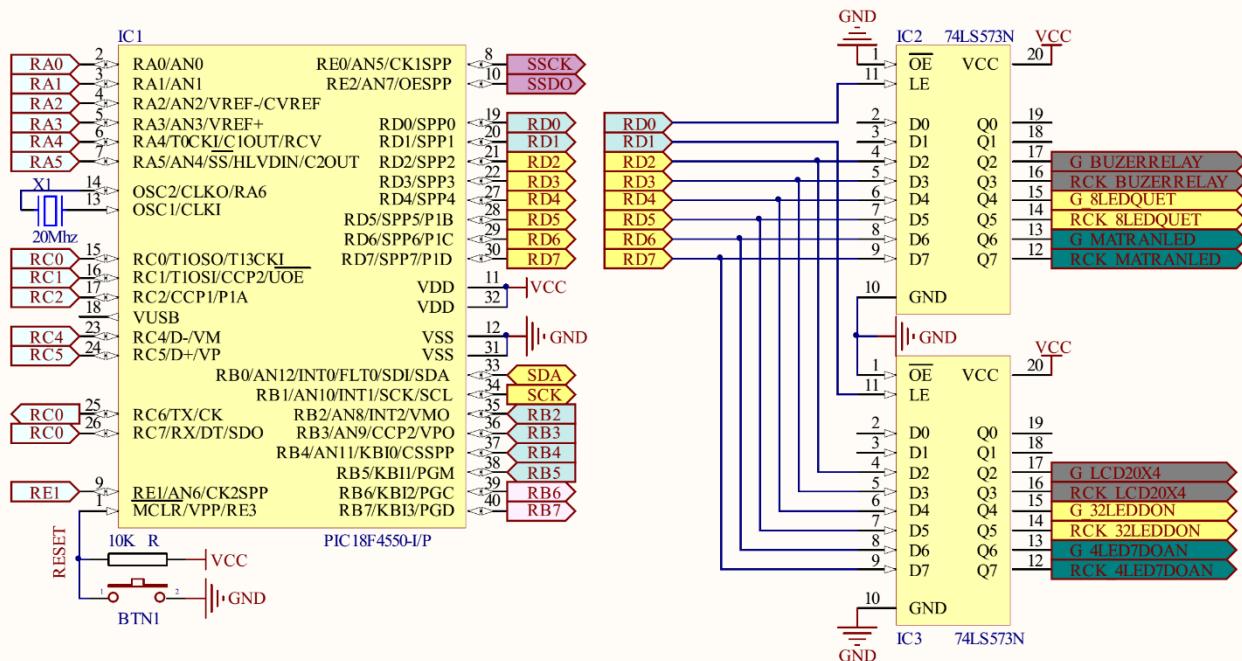


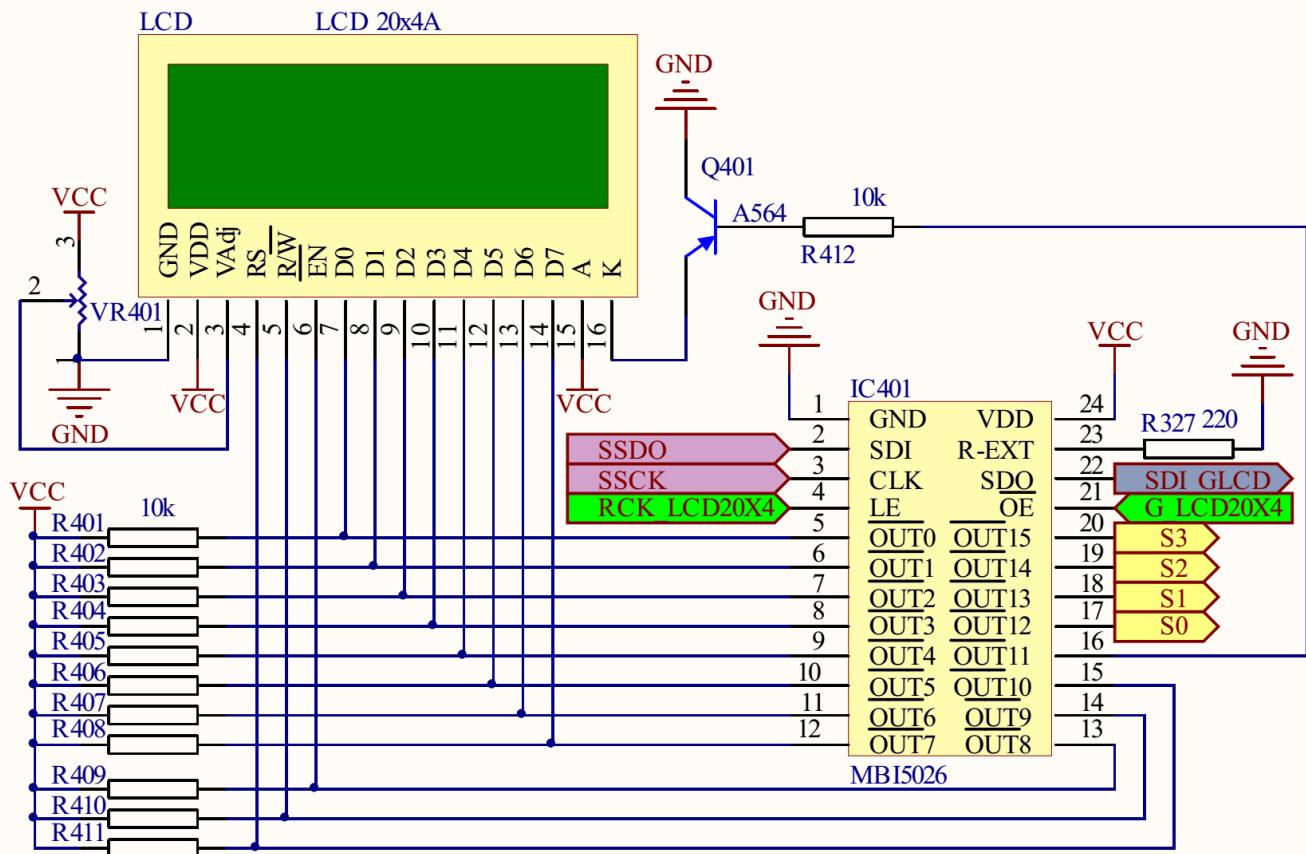
Hình 6-7. Trình tự khởi tạo LCD.

6.2 LCD TRONG KIT THỰC HÀNH

6.2.1 Mạch giao tiếp vi điều khiển với LCD

Trong phần này sẽ trình bày phần giao tiếp vi điều khiển PIC18F4550 với LCD như hình 6-8.





Hình 6-8. Giao tiếp vi điều khiển PIC với LCD qua thanh ghi dịch MBI5026.

Như đã trình bày, trong bộ thực hành sử dụng IC thanh ghi MBI5026 để mở rộng port điều khiển LCD.

Phản vi điều khiển PIC giao tiếp với IC thanh ghi dịch MBI5026:

Có 4 tín hiệu điều khiển là:

- SSDO dùng để dịch dữ liệu nối tiếp.
- SSCK dùng để cấp xung clock.
- RCK_LCD20X4 dùng điều khiển nạp dữ liệu song song từ bên trong IC thanh ghi dịch MBI5026 ra bên ngoài để điều khiển LCD.
- G_LCD20X4 dùng để cho phép xuất dữ liệu.

Phản IC thanh ghi dịch MBI5026 giao tiếp với LCD:

Có 16 tín hiệu ngõ ra với các chức năng như sau:

- Từ out0 đến out7: giao tiếp với bus 8 dữ liệu của LCD theo chế độ 8 bit (vẫn có thể thực hành được chế độ 4 bit).
- Ba tín hiệu out8, out9, out10 làm các tín hiệu điều khiển LCD tương ứng với E, RW và RS.
- Tín hiệu out11 làm tín hiệu điều khiển đóng ngắt transistor cho phép tắt mờ đèn nền để tiết kiệm nguồn năng lượng (P) và tăng tuổi thọ của đèn nền khi không sử dụng.

Mức logic 0 làm đèn nền sáng, mức 1 hoặc khi ngõ ra ở trạng thái tổng trở cao thì đèn nền tắt.

- Bốn tín hiệu S0 đến S3 dùng để giao tiếp IC cảm biến ánh sáng.
- Ngõ ra SDO nối sang một IC MBI 5026 khác để điều khiển GLCD và sẽ trình bày ở phần GLCD.

6.2.2 Dữ liệu và hàm điều khiển LCD và GLCD

Theo kết nối phần cứng đã trình bày thì dữ liệu điều khiển LCD gồm có 2 byte:

- Byte thứ 1 (LCDDATA) dùng để tải dữ liệu.
- Byte thứ 2 (LCDCONTROL) dùng để tải 4 tín hiệu điều khiển E, RW, RS, P (không tính đến 4 tín hiệu S0 đến S3).

Trong kit thực hành còn có module GLCD chung các tín hiệu điều khiển với LCD và cũng dùng IC MBI 5026, phần cứng giao tiếp, dữ liệu điều khiển cũng tương tự như LCD. Có 2 byte để điều khiển GLCD.

- Byte thứ 1 (GLCDDATA) dùng để tải dữ liệu.
- Byte thứ 2 (GLCDCONTROL) dùng để tải 6 tín hiệu điều khiển sẽ được trình bày chi tiết ở GLCD.

Vậy có tổng cộng 4 byte dữ liệu khi giao tiếp với module LCD, GLCD.

Trình tự gọi dữ liệu:

- GLCDCONTROL
- GLCDDATA
- LCDCONTROL
- LCDDATA

Vì điều khiển giao tiếp với cả 2 module LCD và GLCD nên phát sinh các trường hợp thực hành như sau:

- Chỉ thực hành các bài điều khiển LCD.
- Chỉ thực hành các bài điều khiển GLCD.
- Kết hợp cả 2 module LCD và GLCD.

Trong thư viện đã khai báo 4 byte dữ liệu để điều khiển LCD và GLCD như sau:

```
unsigned int8 lcddata=0;
unsigned int8 lcdcontrol=0;
unsigned int8 glcddata=0;
unsigned int8 glcdcontrol=0;
```

Các hàm gọi dữ liệu ra LCD, GLCD và các biến để lưu các dữ liệu và các tín hiệu điều khiển như sau:

Hàm thứ 601: xuất 4 byte ra module LCD và GLCD:

```
void xuat_glcd_lcd()
{
    xuat_1byte(glcdcontrol);
    xuat_1byte(glcddata);
```

```

xuat_1byte(lcdcontrol);
xuat_1byte(lcddata);

mo_ic_74573_a_thong_dl();
output_high(rck_lcd20x4);
output_low(rck_lcd20x4);
mo_glcd_lcd;
chot_ic_74573_a_goi_du_lieu;
}

```

Hàm này gởi lần lượt 4 byte ra 2 IC thanh ghi dịch.

Hàm thứ 602: xuất 4 byte ra module LCD còn dữ liệu GLCD thì không đổi:

```

void xuat_lcd20x4(unsigned int8 lcd_signal,lcd_ins_hthi)
{
    lcdcontrol = ~lcd_signal;
    lcddata    = ~lcd_ins_hthi;
    xuat_glcd_lcd();
}

```

Nếu bạn chỉ điều khiển 1 module LCD thì nên dùng hàm này.

Hàm thứ 603: xuất 4 byte ra module GLCD còn dữ liệu LCD thì không đổi:

```

void xuat_glcd128x64(unsigned int8 glcd_signal,glcd_ins_hthi)
{
    glcdcontrol = ~glcd_signal;
    glcddata    = ~glcd_ins_hthi;
    xuat_glcd_lcd();
}

```

Nếu bạn chỉ điều khiển 1 module GLCD thì nên dùng hàm này.

6.3 CÁC CHƯƠNG TRÌNH SỬ DỤNG VÙNG NHỚ DDRAM CỦA LCD

Trong LCD có 2 vùng nhớ thì phần này sử dụng vùng nhớ DDRAM, phần sau sẽ dùng vùng nhớ CGRAM còn lại.

6.3.1 Các ứng dụng hiển thị ký tự, dịch chuỗi trên LCD

Thư viện LCD. Chương trình thư viện điều khiển LCD 20x4 đã cho trong thư mục của bạn có tên là “TV_PICKIT2_SHIFT_LCD.c”.

Bạn có thể sử dụng cho các bài thực hành và **nên đọc hiểu**.

a. Chương trình:

```

unsigned int8 lcd_control;
#bit lcd_p = lcd_control.3
#bit lcd_rs = lcd_control.2
#bit lcd_rw = lcd_control.1
#bit lcd_e = lcd_control.0

#define lcd_function_set      0x38
#define lcd_display_control   0x0c

```

```

#define lcd_clear_display      0x01
#define lcd_entry_mode        0x06
#define lcd_shift_left         0x18
#define lcd_shift_right        0x1c
                                         câu trúc LCD (Tr.130)

#define lcd_addr_line1         0x80
#define lcd_addr_line2         0xc0
#define lcd_addr_line3         0x94
#define lcd_addr_line4         0xd4

void lcd_xuat_8bit(int8 lcd_data_x)
{
    lcd_e=1; xuat_lcd20x4(lcd_control,lcd_data_x);
    lcd_e=0; xuat_lcd20x4(lcd_control,lcd_data_x);
}
void lcd_command(int8 lcd_data_x)
{
    lcd_rs = 0;
    lcd_xuat_8bit(lcd_data_x);
    delay_us(20);
}
void lcd_data(int8 lcd_data_x)
{
    lcd_rs = 1;
    lcd_xuat_8bit(lcd_data_x);
    delay_us(20);
}

void setup_lcd()
{
    lcd_e = 0;
    lcd_rw = 0;
    lcd_p =0;
    lcd_command(lcd_function_set);
    delay_us(40);
    lcd_command(lcd_display_control);
    delay_us(40);
    lcd_command(lcd_clear_display);
    delay_ms(2);
    lcd_command(lcd_entry_mode);
    delay_us(40);
}

const unsigned char lcd_so_x[10][6] =
{
    0,1,2,4,3,5,                                // so 0
    1,2,32,3,7,3,                               // so 1
    6,6,2,4,3,3,                               // so 2
    6,6,2,3,3,5,                               // so 3
    7,3,7,32,32,7,                            // so 4
}

```

```

7,6,6,3,3,5,                                // so 5
0,6,6,4,3,5,                                // so 6
1,1,7,32,32,7,                                // so 7
0,6,2,4,3,5,                                // so 8
0,6,2,3,3,5};                                // so 9

const unsigned char lcd_ma_8doan[] =
{
    0x07,0x0f,0x1f,0x1f,0x1f,0x1f,0x1f,0x1f, //doan f - 0
    0x1f,0x1f,0x1f,0x00,0x00,0x00,0x00,0x00, //doan a - 1
    0x1c,0x1e,0x1f,0x1f,0x1f,0x1f,0x1f,0x1f, //doan b - 2
    0x00,0x00,0x00,0x00,0x00,0x1f,0x1f,0x1f, //doan d - 3
    0x1f,0x1f,0x1f,0x1f,0x1f,0x1f,0x0f,0x07, //doan e - 4
    0x1f,0x1f,0x1f,0x1f,0x1f,0x1f,0x1e,0x1c, //doan c - 5
    0x1f,0x1f,0x1f,0x00,0x00,0x1f,0x1f,0x1f, //doan g+d-6
    0x1f,0x1f,0x1f,0x1f,0x1f,0x1f,0x1f,0x1f}; //doan i -7

void lcd_goto_xy(signed int8 x, signed int8 y)
{
    const unsigned int8 lcd_vitri[]={0x80,0xc0,0x94,0xd4};
    lcd_command(lcd_vitri[x]+y);
}

void xxx()
{
    signed int8 x;
    x= lcd_so_x[0][0];
    x= lcd_ma_8doan[0];
}

```

b. Giải thích chương trình:

Phản thứ nhất: là định nghĩa dữ liệu điều khiển LCD:

```

unsigned int8 lcd_control;
#bit lcd_p = lcd_control.3
#bit lcd_rs = lcd_control.2
#bit lcd_rw = lcd_control.1
#bit lcd_e = lcd_control.0

#define lcd_function_set      0x38
#define lcd_display_control   0x0c
#define lcd_clear_display     0x01
#define lcd_entry_mode        0x06
#define lcd_shift_left         0x18
#define lcd_shift_right        0x1c

#define lcd_addr_line1         0x80
#define lcd_addr_line2         0xc0
#define lcd_addr_line3         0x94
#define lcd_addr_line4         0xd4

```

các mã lệnh trong
datasheet

Hàng thứ nhất là khai báo biến điều khiển LCD “LCD_CONTROL” – nên gắn tên LCD cho các dữ liệu để dễ quản lý và không nhầm lẫn khi có nhiều đối tượng trong một hệ thống lớn.

4 hàng tiếp theo sau là định nghĩa 4 bit điều khiển LCD, trong đó có 1 bit điều khiển nguồn cung cấp cho LCD có tên là LCD_P.

Tiếp theo là định nghĩa các mã lệnh điều khiển của LCD gồm có 6 lệnh thường dùng.

Cuối cùng là định nghĩa các địa chỉ bắt đầu của từng hàng hiển thị trên LCD.

Giá trị của các mã lệnh và địa chỉ đã trình bày ở trên.

Phần thứ hai: là hàm khởi tạo LCD.

```
void setup_lcd ()
{
    lcd_e = 0;
    lcd_rw = 0;
    lcd_p = 0;
    lcd_command(lcd_function_set);
    delay_us(40);
    lcd_command(lcd_display_control);
    delay_us(40);
    lcd_command(lcd_clear_display);
    delay_ms(2);
    lcd_command(lcd_entry_mode);
    delay_us(40);
}
```

4 lệnh đầu tiên thực hiện các mức logic ban đầu cho các tín hiệu điều khiển LCD và mở nguồn cho đèn backlight sáng.

Mã lệnh đầu tiên gửi đến LCD là LCD_FUNCTION_SET. Hàm được gọi là LCD_COMMAND.

Sau đó delay 39μs thì ta chọn 40 cũng được.

Tương tự cho các mã lệnh còn lại.

Phần thứ ba: là 3 hàm xuất dữ liệu ra LCD.

```
void lcd_xuat_8bit(int8 lcd_data_x)
{
    lcd_e=1; xuat_lcd20x4(lcd_control, lcd_data_x);
    lcd_e=0; xuat_lcd20x4(lcd_control, lcd_data_x);
}
void lcd_command(int8 lcd_data_x)
{
    lcd_rs = 0;
    lcd_xuat_8bit(lcd_data_x);
    delay_us(20);
}
void lcd_data(int8 lcd_data_x)
{
    lcd_rs = 1;
    lcd_xuat_8bit(lcd_data_x);
```

```

delay_us(20);
}

```

Hàm “LCD_COMMAND” có chức năng gửi mã lệnh điều khiển ra LCD.

Hàm “LCD_DTA” có chức năng gửi dữ liệu ra LCD để hiển thị.

Hai hàm này khác nhau ở bit điều khiển LCD_RS.

Cả 2 hàm dùng chung 1 hàm xuất dữ liệu “LCD_XUAT_8BIT”.

Theo giản đồ thời gian, khi xuất dữ liệu ra thì tín hiệu LCD_E phải lên mức 1 được thực hiện bởi 2 lệnh đầu tiên “**LCD_E=1; XUAT_LCD20X4(LCD_CONTROL, LCD_DATA_X);**”

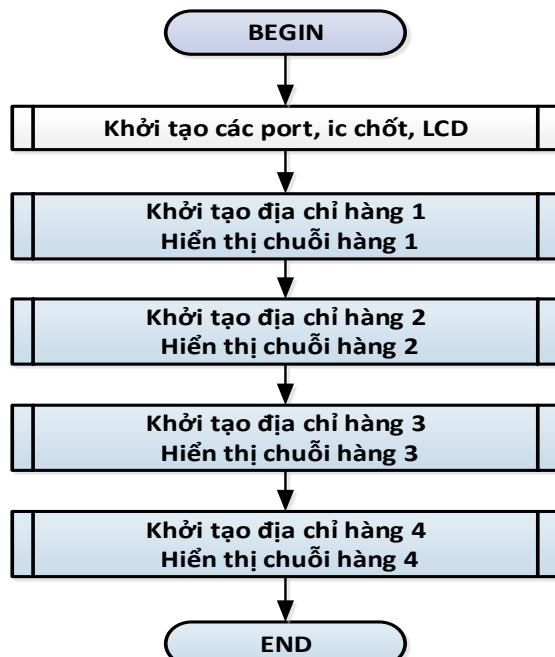
Sau đó, giữ nguyên dữ liệu nhưng điều khiển tín hiệu LCD_E xuống mức logic 0 để chốt dữ liệu hay báo cho vi điều khiển của LCD biết là có dữ liệu mới được thực hiện bằng 2 lệnh còn lại “**LCD_E=0; XUAT_LCD20X4(LCD_CONTROL, LCD_DATA_X);**”

Các hàm còn lại phục vụ cho phần sau.

Bài mẫu 601. Chương trình điều khiển LCD 20x4 hiển thị 4 hàng kí tự. Bạn có thể thay đổi nội dung hiển thị ở 4 hàng tùy thích nhưng chỉ trong phạm vi 20 kí tự cho mỗi hàng.

Lưu tên file là “BAI_601_LCD_HIENTHI_4HANG”

- Mục đích: Biết khởi tạo LCD, biết hiển thị chuỗi kí tự trên các hàng của LCD.
- Lưu đồ:



Hình 6-9. Lưu đồ điều khiển LCD hiển thị 4 chuỗi kí tự.

- Chương trình:

```

#include <tv_pickit2_shift_1.c>
#include <tv_pickit2_shift_lcd.c>

unsigned char hang1[]={ "1hien thi lcd 20x4** " };

```

```

unsigned char hang2[]{"2dai hoc su pham kt  "};
unsigned char hang3[]{"3bo thi nghiem vdk  "};
unsigned char hang4[]{"0123456789abcdefghi "};
signed int8 i;

void main()
{
    set_up_port_ic_chot();
    setup_lcd();

    lcd_command(lcd_addr_line1);
    for (i=0;i<20;i++) { lcd_data(hang1[i]); }

    lcd_command(lcd_addr_line2);
    for (i=0;i<20;i++) { lcd_data(hang2[i]); }

    lcd_command(lcd_addr_line3);
    for (i=0;i<20;i++) { lcd_data(hang3[i]); }

    lcd_command(lcd_addr_line4);
    for (i=0;i<20;i++) { lcd_data(hang4[i]); }

    while(true);
}

```

để trống -> nghĩa là không giới hạn số ký tự

"KDD SPKT" : 8 ký tự

4 chuỗi ứng với 4 hàng của LCD
20x4 : 20 cột (ký tự), 4 hàng

số ký tự cần hiển thị

- d. Tiến hành biên dịch và nạp.
- e. Quan sát kết quả: hiển thị nội dung 4 hàng đã khai báo trong chương trình.
- f. Giải thích chương trình:

Bài mẫu 602. Chương trình điều khiển LCD 20x4 hiển thị 4 hàng kí tự sử dụng thư viện LCD. Bạn có thể thay đổi nội dung hiển thị ở 4 hàng tùy thích nhưng chỉ trong phạm vi 20 kí tự cho mỗi hàng.

Lưu tên file là “BAI_602_LCD_HIENTHI_4HANG”

- a. Mục đích: Biết khởi tạo LCD, biết hiển thị chuỗi kí tự trên các hàng của LCD.
- b. Lưu đồ: sinh viên hãy tự viết.
- c. Chương trình:

```

#include <tv_pickit2_shift_1.c>
#include <tv_pickit2_shift_lcd.c>
void main()
{
    set_up_port_ic_chot();
    setup_lcd();

    lcd_command(lcd_addr_line1);
    lcd_data("1hien thi lcd 20x4** ");

    lcd_command(lcd_addr_line2);
    lcd_data("2dai hoc su pham kt ");
}

```

không cần đếm bao nhiêu ký tự, nhưng phải <20, nếu quá thì sẽ hiển thị qua hàng khác (1-->3, 2-->4) --> xem cấu trúc bộ nhớ hiển thị trang 130

lcd_command (gửi lệnh), lcd_data (gửi dữ liệu)
vd: xác định vị trí (lệnh)
nội dung hiển thị (dữ liệu)
phân biệt được lệnh hay dữ liệu là nhờ chân RS. (RS=0:lệnh, RS=1: dữ liệu)

```

lcd_command(lcd_addr_line3);
lcd_data("3bo thi nglem vdk   ");

lcd_command(lcd_addr_line4);
lcd_data("0123456789abcdefgij");

while(true);
}

```

- d. Tiến hành biên dịch và nạp.
- e. Quan sát kết quả: hiển thị nội dung 4 hàng đã khai báo trong chương trình.
- f. Giải thích chương trình: hãy so sánh 2 bài 601 và 602.

<p>Bài tập 603. Hãy viết chương trình hiển thị nội dung 4 hàng với 4 góc của 4 hàng. Lưu tên file là “BAI_603_LCD.c”</p>	<p>xđ vị trí --> hiển thị</p> <pre> lcd_command(0x80); lcd_data("A"); lcd_command(0x93); lcd_data("B"); ... </pre>	<p>20×4 hiển thị 4 kí tự A, B, C, D ở</p>
---	---	---

Bài mẫu 604. Chương trình điều khiển LCD 20×4 hiển thị 2 hàng kí tự và dịch trái sau khoảng thời gian trễ 1 giây.

Lưu tên file là “BAI_604_LCD_HT_2HANG_DICH_TRAI”

- a. Mục đích: Biết dịch chuỗi kí tự trên các hàng của LCD.
- b. Lưu đồ: sinh viên hãy tự viết.
- c. Chương trình:

```

#include <tv_pickit2_shift_1.c>
#include <tv_pickit2_shift_lcd.c>

void main()
{
    set_up_port_ic_chot();
    setup_lcd();
    lcd_command(lcd_addr_line1);
    lcd_data("khoa dien - dien tu ");
    lcd_command(lcd_addr_line2);
    lcd_data("bm dientucong nglep");
    delay_ms(1000);
    while(true)
    {
        lcd_command(lcd_shift_left);
        delay_ms(1000);
    }
}

```

- d. Tiến hành biên dịch và nạp.

- e. Quan sát kết quả: hiển thị 2 hàng chuỗi và sau đó dịch chuyển xuống hai hàng dưới.
f. Giải thích chương trình:

Bài tập 605. Hãy hiệu chỉnh bài trên để dịch phải sau khoảng thời gian trễ 1 giây.

Lưu tên file là “BAI_605_LCD_HT_2HANG_DICH_PHAI”; lcd_command(lcd_shift_right);

Bài tập 606. Hãy hiệu chỉnh bài trên để dịch trái từ 2 hàng trên, xuống 2 hàng dưới thì dịch phải trở lại hai hàng trên lặp lại, thời gian trễ 1 giây.

Lưu tên file là “BAI_606_LCD_HT_2HANG_DICH_2C”

for(20 lần)
{ dịch trái }

for(20 lần)
{ dịch phải }

6.3.2 Đếm thời gian và đếm xung ngoại hiển thị trên LCD

Bài mẫu 611. Chương trình đếm giây hiển thị trên LCD 20x4

Lưu tên file là “BAI_611_LCD_DEM_GIAY”

- a. Mục đích: Biết đếm và giải mã số HEX sang mã ASCII hiển thị trên LCD.
b. Lưu đồ: sinh viên hãy tự viết.
c. Chương trình:

```
#include <tv_pickit2_shift_1.c>
#include <tv_pickit2_shift_lcd.c>
signed int8 giay;
void lcd_hienthi_dongho()
{
    lcd_goto_xy(0,18);
    delay_us(20);
    lcd_data(giay/10+0x30);
    lcd_data(giay%10+0x30);
}
void main()
{
    set_up_port_ic_chot();
    setup_lcd();
    lcd_goto_xy(0,0);
    lcd_data("dong ho:");
    giay = 0;
    while(true)
    {
        lcd_hienthi_dongho();
        delay_ms(1000);
        giay++;
        if (giay==60) giay=0;
    }
}
```

0x30: mã ASCII số 0
0x31: 1
(bảng mã ASCII - tr.127)

Giải mã: tách số, lấy mã ASCII (số đếm + 0x30)

0,18
x,y
x: hàng, y: cột
--> xđ vị trí

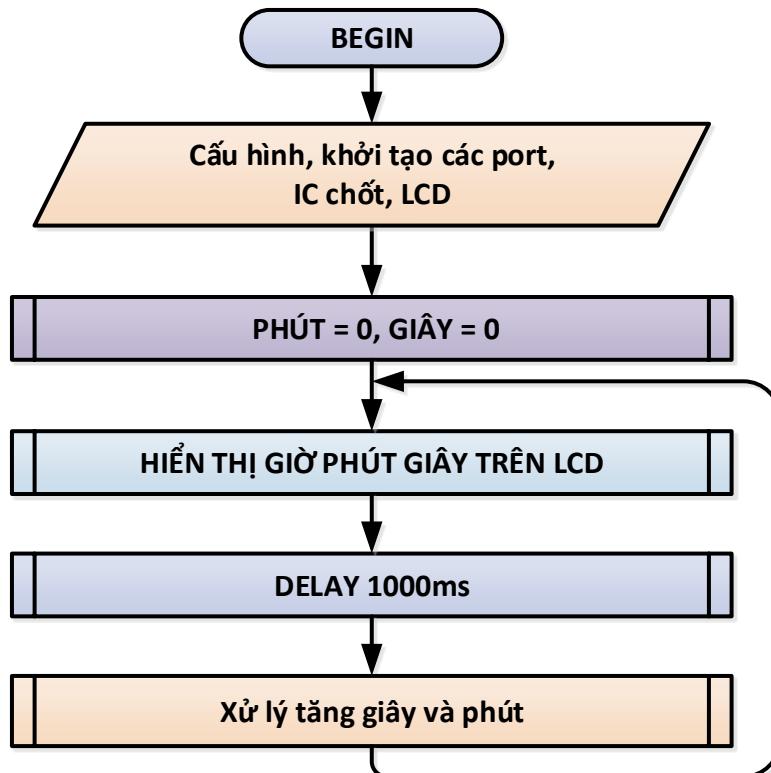
hiển thị từ trái --> phải
chục --> đơn vị

- d. Tiến hành biên dịch và nạp.
e. Quan sát kết quả: giây sẽ đếm và hiển thị ở hàng 1, gần vị trí thứ 18 và 19 của LCD.
f. Giải thích chương trình:

Bài mẫu 612. Chương trình đếm phút giây hiển thị trên LCD 20x4

Lưu tên file là “BAI_612_LCD_DEM_PHUT_GIAY”

- Mục đích: Biết đếm phút giây và giải mã số HEX sang mã ASCII hiển thị trên LCD.
- Lưu đồ:

**Hình 6- 10. Lưu đồ đếm phút giây hiển thị trên LCD.**

- Chương trình:

```

#include <tv_pickit2_shift_1.c>
#include <tv_pickit2_shift_lcd.c>
signed int phut,giay;

void lcd_hienthi_dongho()
{
    lcd_goto_xy(0,15);
    lcd_data(phut/10+0x30); lcd_data(phut%10+0x30);
    lcd_data(' ');
    lcd_data(giay/10+0x30); lcd_data(giay%10+0x30);
}

void main()
{
    set_up_port_ic_chot();
    setup_lcd();
    lcd_goto_xy(0,0);
}
  
```

```

lcd_data("dong ho:");
giay = 0; phut = 0;
while(true)
{
    hienthi_lcd();
    delay_ms(1000);
    giay++;
    if (giay==60)
    {
        giay=0; phut++;
        if (phut==60) phut=0;
    }
}

```

{ phut = 0;
gio++;
if(gio==24) gio = 0;
}

//khởi tạo
gio = 23;
phut = 59;
giay=50;

- d. Tiến hành biên dịch và nạp.
- e. Quan sát kết quả: đếm phút và giây, có hiệu chỉnh thời gian đếm cho nhanh để kiểm tra.
- f. Giải thích chương trình:

Bài tập 613. Hãy viết chương trình đếm giờ phút giây hiển thị trên LCD 20x4.

Lưu tên file là “BAI_613_LCD_DEM_GIO_PHUT_GIAY”

613 thêm bài 512 vào

512 thêm phần hiển thị LCD và phút giây

Bài tập 614. Hãy viết chương trình đếm giờ phút giây hiển thị trên LCD 20x4, sử dụng timer báo ngắt để đếm chính xác, có thêm 3 nút nhấn để chỉnh được thời gian.

Gợi ý: Sử dụng cách viết của bài đếm giờ phút giây hiển thị 8 led 7 đoạn và thực hiện hiển thị trên LCD.

Lưu tên file là “BAI_614_LCD_GIO_PHUT_GIAY”

Bài tập 615. Hãy hiệu chỉnh bài 614 vừa hiển thị trên LCD 20x4 và hiển thị trên module 8 led quét.

không có nút nhấn, LCD có thể chỉ cần hiển thị 1 lần khi đủ 1 giây, vì LCD sẽ giữ nguyên nội dung nếu ta không gửi ra thêm nd gì

Gợi ý: Sử dụng Khi hiển thị thì 8 --> thêm nút nhấn điều chỉnh và nháy nháy trên LCD khi chỉnh giây thay đổi. --> 515,516,517 --> 517 thêm phần hiển thị LCD

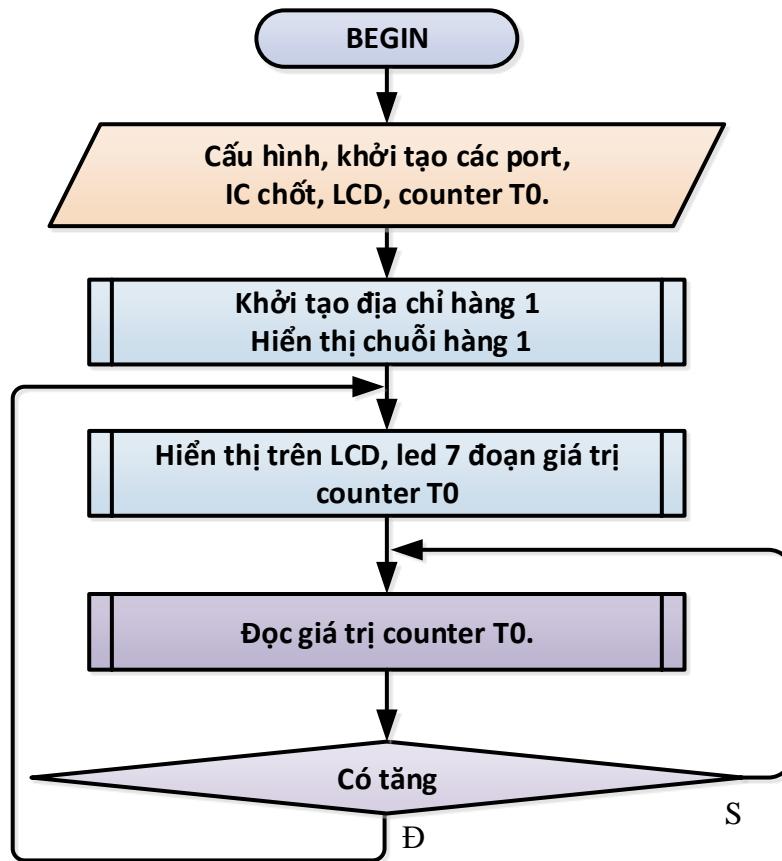
Lưu tên file là “BAI_615_LCD_8LED_GIO_PHUT_GIAY”

412

Bài mẫu 616. Chương trình đếm sản phẩm dùng counter T0 hiển thị trên LCD 20x4 ở hàng thứ 1 và hiển thị trên module 4 led 7 đoạn, giới hạn đếm từ 000 đến 100.

Lưu tên file là “BAI_616_LCD_4LED_DEM_SP_T0”

- a. Mục đích: Biết đếm xung ngoại và giải mã số HEX sang mã ASCII hiển thị trên LCD.
 b. Lưu đồ:

**Hình 6-11. Lưu đồ đếm sản phẩm hiển thị trên LCD.**

- c. Chương trình:

```

#include <tv_pickit2_shift_1.c>
#include <tv_pickit2_shift_lcd.c>
unsigned int16 t0,t0_tam;
void hienthi_lcd()
{
    lcd_goto_xy(0,17);
    lcd_data(t0/100+0x30);
    lcd_data(t0/10%10+0x30);
    lcd_data(t0%10+0x30);
}

void main()
{
    set_up_port_ic_chot();
    setup_lcd();
    lcd_goto_xy(0,0);
    lcd_data("dem san pham:");
}
  
```

```

VOID GM_LCD()
{
    DV = T0%10 + 0X30;
    CH = T0/10%10 + 0X30;
    TR = T0/100 + 0X30;

    IF(TR==0X30)
    {
        TR = 32; //0X20 - ''
        IF(CH == 0X30)
            CH = 0X20;
    }
}

VOID HT_LCD()
{
    LCD_GOTO_XY(0,0);
    LCD_DATA(TR);
    LCD_DATA(CH);
    LCD_DATA(DV);
}
  
```

```

setup_timer_0 (t0_ext_l_to_h | t0_div_1);
set_timer0(0);
t0=t0_tam=0;

while(true)
{
    t0_tam=t0;
    hienthi_lcd();
    xuat_4led_7doan_giaima_xoa_so0(t0);
    do{ t0=get_timer0();} while (t0==t0_tam);
    if (t0>=101) set_timer0(1);
}

```

thay do/while
bằng if --> 413

CHỈ HT 1 GT TRÊN 4 LED 7
ĐOẠN --> xuat..4e

thêm ct --> chỉ được
thực hiện khi kết thúc
vòng lặp do/while (khi
có xung)

- d. Tiến hành biên dịch và nạp.
- e. Quan sát kết quả: nếu kết quả không đúng yêu cầu thì kiểm tra lại chương trình.
- f. Giải thích chương trình: Trong chương trình có 2 biến lưu giá trị biến đếm, quá trình giải mã hiển thị trên LCD chỉ được thực hiện khi giá trị đếm (T0) khác với giá trị đã lưu trước đó trong biến đếm tạm (T0_TAM). Chức năng này làm giảm quá trình giải mã và hiển thị trên LCD khi giá trị không đổi, tiết kiệm được thời gian để có thể giúp vi điều khiển đáp ứng các sự kiện khác.

Bài tập 617. Hãy hiệu chỉnh bài 616 thêm phần cài giới hạn đếm bằng 3 nút UP, DW và CLR. Giới hạn cài từ 01 đến 99. Khi nhấn CLR thì giá trị cài về 01, giá trị đang đếm về 00.

Hàng 1 hiển thị giá trị cài, hàng 2 hiển thị giá trị đếm.

4 led 7 đoạn thì 2 led trái hiển thị giá trị cài, 2 led phải hiển thị giá trị đếm.

Lưu tên file là “BAI_617_LCD_4LED_DEM_SP_T0_UD”

XUAT_4LED_7DOAN_4SO(...);

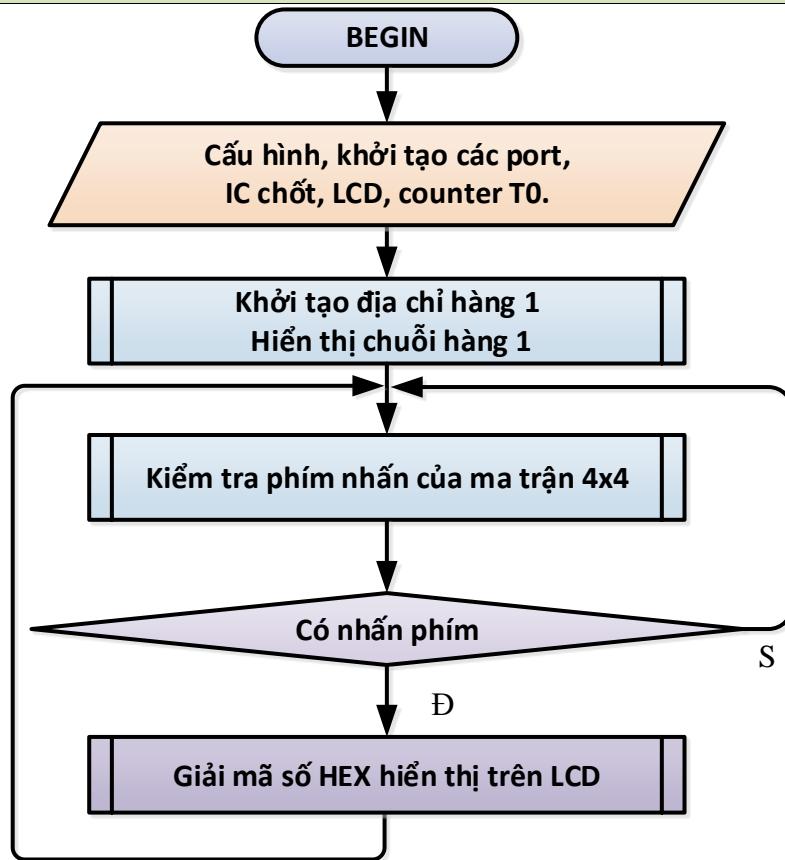
6.3.3 Các ứng dụng kết hợp bàn phím ma trận và LCD

Bài mẫu 621. Chương trình có chức năng khi nhấn phím bất kỳ của bàn phím ma trận 4x4 thì mã phím đó xuất hiện tại vị trí đầu tiên bên phải của hàng 2, ví dụ khi nhấn phím số 0 thì số 0 hiển thị, khi nhấn phím số 5 thì số 5 hiển thị, số trước đó sẽ mất đi.

Hàng 1 hiển thị thông tin: “HAY NHAN PHIM:”

Lưu tên file là “BAI_621_LCD_KEX4X4”

- a. Mục đích: Biết kết hợp bàn phím ma trận với LCD.
- b. Lưu đồ:



Hình 6-12. Lưu đồ quét phím ma trận hiển thị trên LCD.

c. Chương trình:

```

#include <tv_pikkit2_shift_1.c>
#include <tv_pikkit2_shift_lcd.c>
#include <tv_pikkit2_shift_key4x4_138.c>
signed int8 mp;
void hienthi_lcd()
{
    lcd_goto_xy(0,19);
    if(mp<10)    lcd_data(mp+0x30);
    else          lcd_data(mp+0x37);
}

void main()
{
    set_up_port_ic_chot();
    setup_lcd();

    lcd_goto_xy(0,0);
    lcd_data("hay nhan phim:");

    while(true)
    {
        do {mp= key_4x4_up();} dw
    }
}
  
```

```

        while(mp==0xFF);
        hienthi_lcd();
    }
}

```

IF(MP != 0xFF)
{
 HIENTHI_LCD();
}

- d. Tiến hành biên dịch và nạp.
- e. Quan sát kết quả: nếu kết quả không đúng yêu cầu thì kiểm tra lại chương trình.
- f. Giải thích chương trình: tại sao cộng 0x30 và 0x37?

Bài tập 622. Hãy viết chương trình có chức năng khi nhấn phím bất kỳ của bàn phím ma trận 4×4 thì mã phím đó xuất hiện tại tận cùng bên phải của hàng 2, các kí tự đã nhấn sẽ dịch sang trái. Các kí tự hàng 1 không thay đổi.

Hàng 1 hiển thị thông tin : “HAY NHAN PHIM:”

Lưu tên file là “BAI_622_LCD_KEX4X4_DICH”

Bài tập 623. Hãy viết chương trình có chức năng khi mới cấp điện thì 4 hàng hiển thị kí tự trắng, khi nhấn phím bất kỳ của bàn phím ma trận 4×4 thì mã phím đó xuất hiện tại tận cùng bên phải của hàng 1, nhấn nữa thì dịch sang trái, cho đến khi được 20 kí tự mà nhấn nữa thì xuống hàng 2, khi hàng 2 đủ 20 kí tự thì dịch xuống hàng 3 và khi hàng 3 đủ 20 thì xuống hàng 4.

Lưu tên file là “BAI_623_LCD_KEX4X4_DICH_4HANG_1234”

Bài tập 624. Hãy viết chương trình có chức năng hiển thị 2 hàng thông tin và điều khiển buzzer bằng cách khi nhấn phím A của bàn phím ma trận 4×4 thì buzzer sẽ kêu úng với thời gian Y nằm trong giới hạn từ 0 đến 9 được thay đổi bằng 10 phím số từ 0 đến 9 của bàn phím ma trận.

Khi buzzer kêu thì XXX sẽ hiển thị chữ “ON”, khi chuông hết kêu thì hiển thị chữ “OFF”. Thời gian Y mặc nhiên khi bắt đầu là 1s, nếu thời gian bằng 0 thì chuông không kêu nếu nhấn A.

Hàng 1 hiển thị thông tin: “ DIEU KHIEN BUZZER ”

Hàng 2 hiển thị thông tin: “BUZZER: XXX, TG:Y”

Lưu tên file là “BAI_624_LCD_KEX4X4_DICH”

Bài tập 625. Hãy viết lại bài 617 nhưng dùng bàn phím ma trận để cài giới hạn.

Ví dụ, muốn cài 45 thì nhấn 4 hiển thị ở hàng đơn vị, sau đó nhấn 5 thì giá trị 4 sẽ dịch sang hàng chục và 5 thành hàng đơn vị, kết quả là 45.

Nhấn phím C thì xóa về 01.

bài 423

Lưu tên file là “ BAI_625_LCD_4LED DEM_SP_T0_MATRIX_KEY ”

6.4 CÁC ỨNG DỤNG VÙNG NHỚ CGRAM CỦA LCD

6.4.1 Vùng nhớ CGRAM

Ở mục trước ta đã sử dụng LCD có chức năng hiển thị các kí tự hoặc các chuỗi kí tự thì các kí tự đó thuộc mã ASCII và đã tạo sẵn trong chip điều khiển LCD, bạn chỉ cần gởi đúng mã ASCII thì kí tự đó sẽ được hiển thị.

Phần này chúng ta sẽ sử dụng vùng nhớ còn lại là CGRAM mà nhà chế tạo LCD dành cho người dùng tự thiết lập các kí tự mong muốn.

Trong bảng tập lệnh thì có lệnh thiết lập địa chỉ CGRAM có dạng bảng như sau:

Bảng 6-12. Thiết lập địa chỉ CGRAM.

D7	D6	D5	D4	D3	D2	D1	D0	HEX	
0	1	A5	A4	A3	A2	A1	A0		
0	1	0	0	0	0	0	0	0x40	Địa chỉ đầu
0	1		
0	1	1	1	1	1	1	1	0x7F	Địa chỉ cuối

Vùng nhớ CGRAM này có địa chỉ từ 0x40 đến 0x7F cho phép lưu 64 byte dữ liệu.

Mỗi kí tự hiển thị trên LCD là một ma trận gồm 8 byte nên vùng nhớ CGRAM cho phép lưu được 8 kí tự.

Bảng 6-13. Vùng nhớ CGRAM.

Vùng thứ	Địa chỉ cuối	B7	B6	B5	B4	B3	B2	B1	B0	Địa chỉ đầu	Địa chỉ kí tự
0	0x47								0x40	0	
1	0x4F								0x48	1	
2	0x57								0x50	2	
3	0x5F								0x58	3	
4	0x67								0x60	4	
5	0x6F								0x68	5	
6	0x77								0x70	6	
7	0x7F								0x78	7	

6.4.2 Cách tìm mã kí tự mong muốn

Mỗi kí tự là một ma trận điểm 5×8 như bảng sau, muốn điểm nào sáng thì bit trong ô đó bằng 1, với hình trái tim thì byte thứ 0 là $0x0A$ – tính theo hàng ngang, tương tự cho các hàng còn lại.

Bảng 6-14. Mã của trái tim.

BYTE	BIT					HEX
	4	3	2	1	0	
0		X		X		0x0A
1	X		X		X	0x15
2	X				X	0x11
3	X				X	0x11
4		X		X		0x0A
5			X			0x04
6						0x00
7						0x00

0000 1010 = 0x0A

Cách để hiển thị trái tim:

- 1 • Khai báo mảng lưu 8 byte mã của trái tim.
- 2 • Khởi tạo địa chỉ bắt đầu của 1 trong 8 vùng ví dụ chọn $0x40$ để lưu dữ liệu vào vùng nhớ CGRAM từ $0x40$ đến $0x47$.
- 3 • Copy 8 byte dữ liệu của trái tim vào vùng nhớ CGRAM từ $0x40$ đến $0x47$.
- 4 • Gọi lệnh hiển thị kí tự thứ 0.

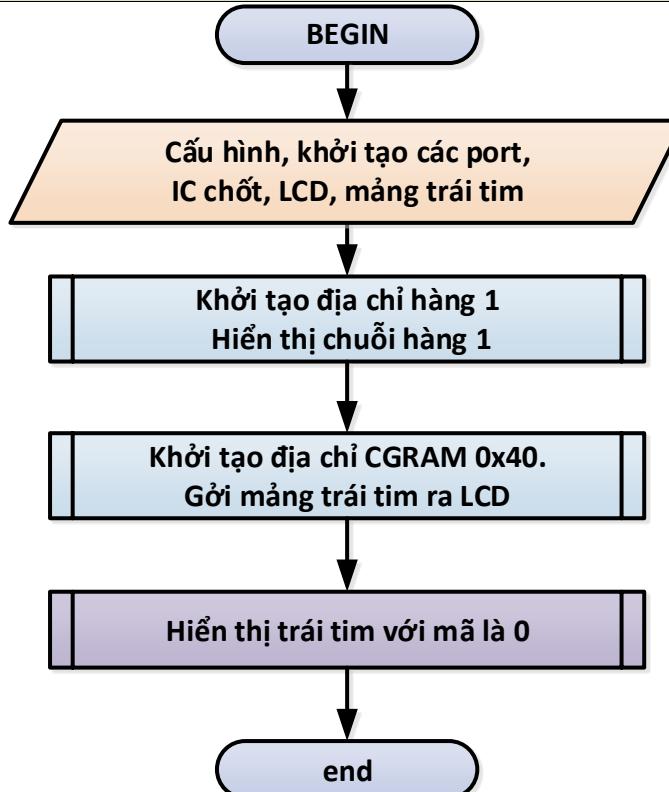
6.4.3 Các chương trình ứng dụng các kí tự - tự tạo trên LCD

Bài mẫu 631. Chương trình hiển thị kí tự hay biểu tượng tự tạo.

Hàng 1 hiển thị thông tin: “ HIEN THI TRAI TIM ”

Lưu tên file là “BAI_631_LCD_TRAITIM”

- Mục đích: biết lập trình hiển thị các kí tự mong muốn.
- Lưu đồ:

*Hình 6-13. Lưu đồ hiển thị mã tự tạo trái tim trên LCD.*

c. Chương trình:

```

#include <tv_pickit2_shift_1.c>
#include <tv_pickit2_shift_lcd.c>
const unsigned char hang2[]={0xa,0x15,0x11,0x11,0xa,0x4,0,0};
signed int8 i;

void main()
{
    set_up_port_ic_chot();
    setup_lcd();
    lcd_goto_xy(0,0);
    lcd_data(" hien thi trai tim ");
    lcd_command(0x40);
    for (i=0;i<8;i++) { lcd_data(hang2[i]); }
    lcd_goto_xy(1,0);
    lcd_data(0);
    while(true);
}

```

bước 1

bước 2 - trang 148 - ký tự thứ 0

bước 3 - copy
0x40 lưu hang2[0]
0x41 lưu hang2[1]

bước 4: xác định vị trí
gửi mã tự tạo

d. Tiến hành biên dịch và nạp.

e. Quan sát kết quả: nếu kết quả không đúng yêu cầu thì kiểm tra lại chương trình.

f. Giải thích chương trình:

Bài tập 632. Hãy hiệu chỉnh bài 631 sao cho hàng 2 hiển thị 20 trái tim.

Lưu tên file là “BAI_632_LCD_20_TRAITIM”

```
lcdgotoxy...
for(20)
{
    lcd_data.....
}
```

Bài tập 633. Hãy lập trình hiển thị 3 ký tự: trái tim và hình thoi hình vuông.

Hàng 1 hiển thị thông tin: “ HIEN THI 3 KY TU ”

Lưu tên file là “BAI_633_LCD_3KYTU”

giữa 3 hàng 2,3,4

Bài tập 634. Hãy lập trình hiển thị 8 ký tự tự tạo khác nhau mà bạn tùy ý chọn.

Hàng 1 hiển thị thông tin: “ HIEN THI 8 KY TU ”

Lưu tên file là “BAI_634_LCD_8KYTU”

Bài tập 635. Hãy lập trình hiển thị ~~trái táo~~ (Apple) chiếm 4 ma trận (mỗi 1 ma trận hiển thị 1 ký tự trên LCD).

Hàng 1 hiển thị thông tin: “ BIEU TUONG APPLE ”

Lưu tên file là “BAI_635_LCD_APPLE”

cột sóng

8 (2 hàng - 4 cột)

Bài tập 636. Hãy thêm vào bài 634 sao cho ~~trái táo~~ di chuyển từ phải sang trái rồi từ trái sang phải.

Hàng 1 hiển thị thông tin: “ BIEU TUONG APPLE ”

Lưu tên file là “BAI_636_LCD_APPLE_DICHUYEN”

//XÓA ND LCD
CLEAR --> XÓA CẢ MÀN HÌNH

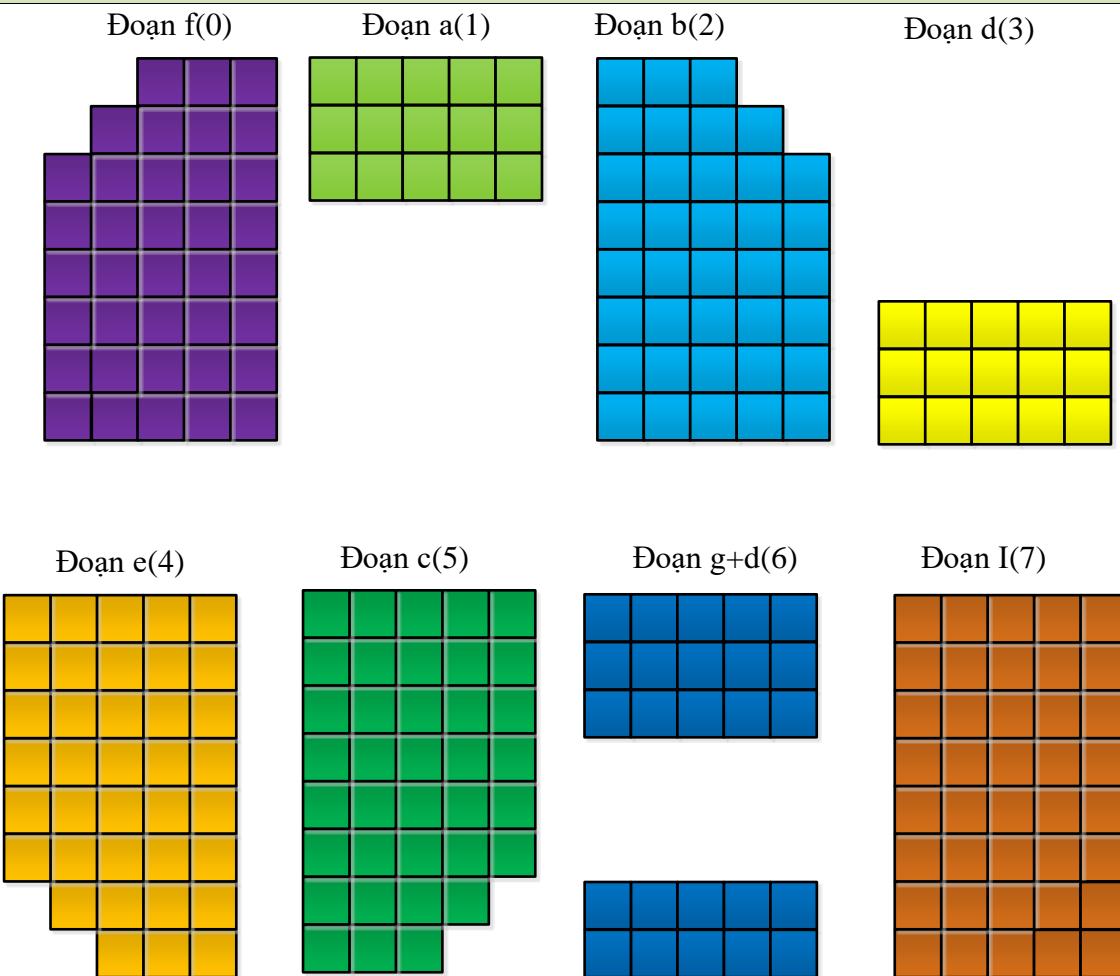
//CÁCH 2

LCD_GOTO_XY(... ,);
FOR(l=0; l<; l++)
LCD_DATA(0X20); // '' - 0X20

6.4.4 Tạo mã 7 đoạn kích thước lớn trên LCD

Bước 1: xây dựng mã cho các đoạn

Để tạo các số từ 0 đến 9 để hiển thị lớn hơn ký tự bình thường cho dễ nhìn thì cách tạo ra gần giống mã của led 7 đoạn. Các đoạn được đánh tên tương ứng là a, b, c, d, e, f, g, i

**Hình 6-14. Hình ảnh các đoạn cho số lớn trên LCD.**

Các con số thứ tự đi kèm là trình tự khai báo trong mảng.

Mã hex của các đoạn tương ứng:

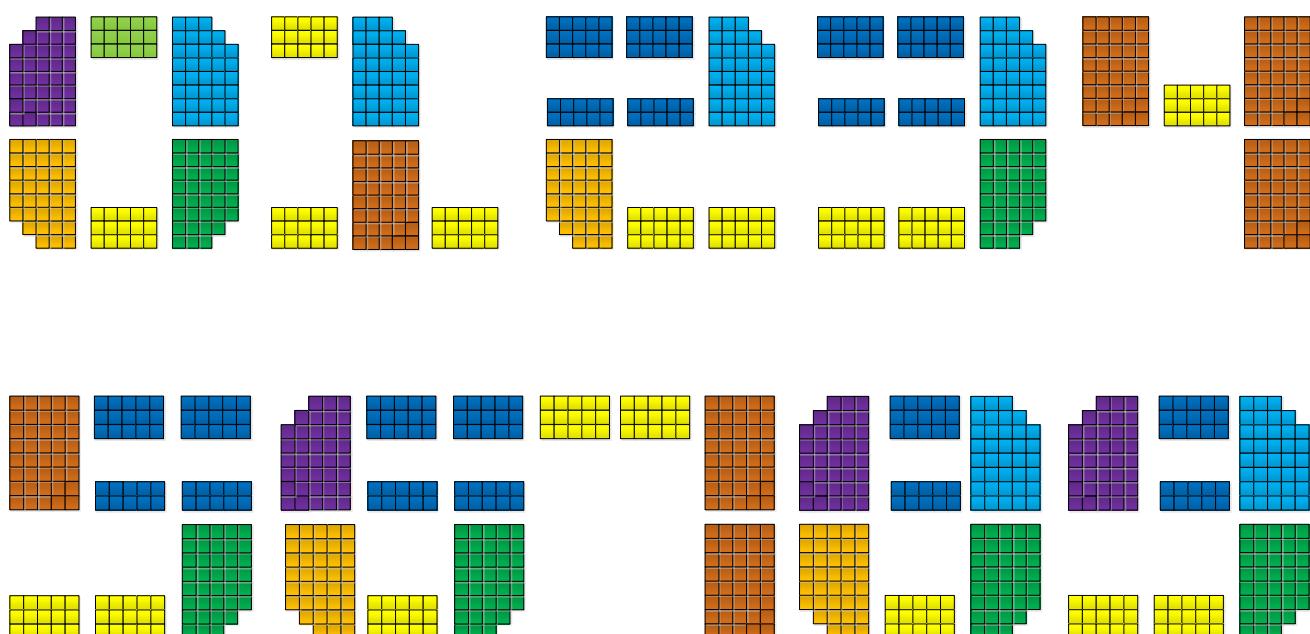
Đoạn f(0)	Đoạn a(1)	Đoạn b(2)	Đoạn d(3)																																
<table border="1"> <tr><td>0x07</td></tr> <tr><td>0x0F</td></tr> <tr><td>0x1F</td></tr> <tr><td>0x1F</td></tr> <tr><td>0x1F</td></tr> <tr><td>0x1F</td></tr> <tr><td>0x1F</td></tr> <tr><td>0x1F</td></tr> </table>	0x07	0x0F	0x1F	0x1F	0x1F	0x1F	0x1F	0x1F	<table border="1"> <tr><td>0x1F</td></tr> <tr><td>0x1F</td></tr> <tr><td>0x1F</td></tr> <tr><td>0x00</td></tr> <tr><td>0x00</td></tr> <tr><td>0x00</td></tr> <tr><td>0x00</td></tr> <tr><td>0x00</td></tr> </table>	0x1F	0x1F	0x1F	0x00	0x00	0x00	0x00	0x00	<table border="1"> <tr><td>0x1C</td></tr> <tr><td>0x1E</td></tr> <tr><td>0x0F</td></tr> <tr><td>0x1F</td></tr> <tr><td>0x1F</td></tr> <tr><td>0x1F</td></tr> <tr><td>0x1F</td></tr> <tr><td>0x1F</td></tr> </table>	0x1C	0x1E	0x0F	0x1F	0x1F	0x1F	0x1F	0x1F	<table border="1"> <tr><td>0x00</td></tr> <tr><td>0x00</td></tr> <tr><td>0x00</td></tr> <tr><td>0x00</td></tr> <tr><td>0x00</td></tr> <tr><td>0x1F</td></tr> <tr><td>0x1F</td></tr> <tr><td>0x1F</td></tr> </table>	0x00	0x00	0x00	0x00	0x00	0x1F	0x1F	0x1F
0x07																																			
0x0F																																			
0x1F																																			
0x1F																																			
0x1F																																			
0x1F																																			
0x1F																																			
0x1F																																			
0x1F																																			
0x1F																																			
0x1F																																			
0x00																																			
0x00																																			
0x00																																			
0x00																																			
0x00																																			
0x1C																																			
0x1E																																			
0x0F																																			
0x1F																																			
0x1F																																			
0x1F																																			
0x1F																																			
0x1F																																			
0x00																																			
0x00																																			
0x00																																			
0x00																																			
0x00																																			
0x1F																																			
0x1F																																			
0x1F																																			

Đoạn e(4)	Đoạn c(5)	Đoạn g+d(6)	Đoạn I(7)
0x1F	0x1F	0x1F	0x1F
0x1F	0x1F	0x1F	0x1F
0x1F	0x1F	0x1F	0x1F
0x1F	0x1F	0x1F	0x1F
0x1F	0x1F	0x1F	0x1F
0x1F	0x1F	0x1E	0x1F
0x0F	0x0F	0x1C	0x1F
0x07	0x07		

Hình 6-15. Tìm mã của các đoạn cho số lớn trên LCD.

Bước 2: Kết hợp các đoạn để xây dựng các con số từ 0 đến 9

Để sáng con số lớn thì ta kết hợp các đoạn để được các con số từ 0 đến 9, mỗi con số sẽ chiếm 6 vị trí như sau:



Hình 6-16. Hình ảnh các số lớn từ 0 đến 9 trên LCD.

Để thuận tiện cho tạo dữ liệu và lập trình, các đoạn được tính theo thứ tự: f, a, b, c, d, e, f, hoặc g+d.

Mỗi kí tự số lớn sẽ sử dụng 6 kí tự bình thường, khi tính thì tính 3 kí tự hàng thứ nhất trước rồi đến 3 kí tự hàng thứ 2 và khi lập trình cũng vậy.

Muốn sáng số 0 thì các đoạn: f, a, b, e, d, c sáng – tương ứng với số thứ tự là: 0, 1, 2, 4, 3, 5.

Muốn sáng số 1 thì các đoạn: a, b, ‘ ’, d, i, d, sáng – tương ứng với số thứ tự là: 1, 2, 0×20, 3, 7, 3.

Muốn sáng số 2 thì các đoạn: g+d, g+d, b, e, d, d sáng – tương ứng với số thứ tự là: 6, 6, 2, 4, 3, 3.

Muốn sáng số 3 thì các đoạn: g+d, g+d, b, d, d, c sáng – tương ứng với số thứ tự là: 6, 6, 2, 3, 3, 5.

Muốn sáng số 4 thì các đoạn: i, d, i, 0, 0, i sáng – tương ứng với số thứ tự là: 7, 3, 7, 0×20, 0×20, 7.

Muốn sáng số 5 thì các đoạn: i, g+d, g+d, d, d, c sáng – tương ứng với số thứ tự là: 7, 6, 6, 3, 3, 5.

Muốn sáng số 6 thì các đoạn: f, g+d, g+d, e, d, c sáng – tương ứng với số thứ tự là: 0, 6, 6, 4, 3, 5.

Muốn sáng số 7 thì các đoạn: a, a, i, 0, 0, i sáng – tương ứng với số thứ tự là: 1, 1, 7, 0×20, 0×20, 7.

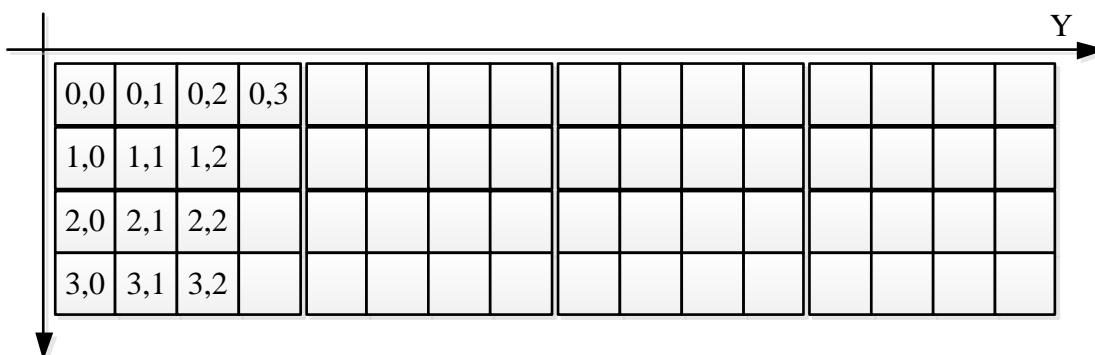
Muốn sáng số 8 thì các đoạn: f, g+d, b, e, d, c sáng – tương ứng với số thứ tự là: 0, 6, 2, 4, 3, 5.

Muốn sáng số 9 thì các đoạn: f, g+d, b, d, d, c sáng – tương ứng với số thứ tự là: 0, 6, 2, 3, 3, 5.

Có 1 số led hiển thị khoảng trắng có mã là 0×20 hay 32.

Bước 3: Thiết lập lại vị trí các kí tự trên màn hình LCD 20×4:

LCD 20×4 có 4 hàng và 20 cột có tổng cộng là 80 kí tự theo tọa độ 2 trục X và Y như hình 6-17.



Hình 6-17. LCD tính theo địa chỉ hàng và cột.

Ô có địa chỉ 0×80 bây giờ gọi là ô có tọa độ (x, y) = (0,0).

Ô có địa chỉ 0×81 bây giờ gọi là ô có tọa độ (x, y) = (0,1).

Ô có địa chỉ 0×C0 bây giờ gọi là ô có tọa độ (x, y) = (1,0).

Ô có địa chỉ 0×C1 bây giờ gọi là ô có tọa độ (x, y) = (1,1).

Tương tự cho các ô còn lại.

Để sáng 1 con số sẽ dùng 6 kí tự, mỗi hàng chiếm 3 kí tự, ví dụ muốn sáng số 0 bắt đầu tại tọa độ (x, y) = (0,0) thì 6 ô sau sẽ được sử dụng là:

3 ô hàng thứ nhất: ô (0,0), ô (0,1), ô (0,2)

3 ô hàng thứ hai: ô (1,0), ô (1,1), ô (1,2)

Mã số 0 có các đoạn là f, a, b, e, d, c sẽ hiển thị lần lượt vào 6 ô nhớ trên.

Tại tọa độ $(x, y) = (0,0)$ ta cho hiển thị đoạn thứ 1 là f. Tăng giá trị y lên 1: $(x, y) = (0, 1)$ để hiển thị đoạn thứ 2 là a. Tăng giá trị y lên 1: $(x, y) = (0, 2)$ để hiển thị đoạn thứ 3 là b.

Tăng giá trị x lên 1: $(x, y) = (1, 0)$ ta cho hiển thị đoạn thứ 4 là e. Tăng giá trị y lên 1: $(x, y) = (1, 1)$ để hiển thị đoạn thứ 5 là d. Tăng giá trị y lên 1: $(x, y) = (1, 2)$ để hiển thị đoạn thứ 6 là c.

Các giá trị tọa độ hiển thị được tính theo trục x và y nhưng địa chỉ thật được tính từ giá trị xuất phát là 0x80, 0x81, 0x82 và 0xC0, 0xC1, 0xC2.

Cách tính:

- Nếu x bằng 0 thì bắt đầu là 0x80.
- Nếu x bằng 1 thì bắt đầu là 0xC0.
- Nếu x bằng 2 thì bắt đầu là 0x94.
- Nếu x bằng 3 thì bắt đầu là 0xD4.
- Sau đó ta cộng địa chỉ trên với giá trị của y.

Bước 4: Viết chương trình:

Các mã của các đoạn đã viết trong chương trình thư viện của LCD bao gồm mảng 2 chiều cho 10 kí tự từ 0 đến 9, mỗi kí tự có 8 byte khi hiển thị sẽ tạo thành con số.

Chương trình con LCD_GOTO_XY() có chức năng di chuyển con trỏ đến tọa độ (x, y) .

Các dữ liệu này đã viết trong hàm thư viện của LCD.

Bài mẫu 641. Chương trình hiển thị kí tự số 0, mỗi con số chiếm 6 kí tự trên LCD20×4.

Lưu tên file là “BAI_641_LCD_SO_0”

- a. Mục đích: biết lập trình hiển thị con số có kích thước lớn trên màn hình LCD.
- b. Lưu đồ: sinh viên hãy tự viết.
- c. Chương trình:

```
#include <tv_pickit2_shift_1.c>
#include <tv_pickit2_shift_lcd.c>
signed int8 i;
void lcd_hienthi_so_z_toado_xy(signed int8 lcd_so, x1, y1)
{
    lcd_goto_xy(x1,y1);
    for (i=0;i<6;i++)
    {
        if (i==3)    lcd_goto_xy(x1+1,y1);
        lcd_data(lcd_so_x[lcd_so][i]);
    }
}

void main()
{
    set_up_port_ic_chot();
    setup_lcd();
}
```

```

lcd_goto_xy(0,0);
lcd_data("hien thi so 0 hang 3");

lcd_command(0x40);
for (i=0;i<64;i++) { lcd_data(lcd_ma_8doan[i]); }

lcd_hienthi_so_z_toado_xy(0,2,0);
while(true);
}

```

- d. Tiến hành biên dịch và nạp.
- e. Quan sát kết quả: nếu kết quả không đúng yêu cầu thì kiểm tra lại chương trình.
- f. Giải thích chương trình:

Bài tập 642. Hãy viết chương trình hiển thị kí tự số 0123 kích thước lớn trên LCD20×4.

Lưu tên file là “BAI_642_LCD_SO_0123”

Bài mẫu 643. Chương trình đếm từ 00 đến 99 **hiển thị số kích thước lớn.**

Hàng 1 hiển thị thông tin : “**DEM TU 00 DEN 99**”

Lưu tên file là “BAI_643_LCD_DEM_00_99”

- a. Mục đích: biết lập trình đếm giải mã hiển thị con số có kích thước lớn trên màn hình LCD.
- b. Lưu đồ: sinh viên hãy tự viết.
- c. Chương trình:

```

#include <tv_pickit2_shift_1.c>
#include <tv_pickit2_shift_lcd.c>
signed int8 i,dem;
void lcd_hienthi_so_z_toado_xy(signed int8 lcd_so, x1, y1)
{
    lcd_goto_xy(x1,y1);
    for (i=0;i<6;i++)
    {
        if (i==3) lcd_goto_xy(x1+1,y1);
        lcd_data(lcd_so_x[lcd_so][i]);
    }
}

void main()
{
    set_up_port_ic_chot();
    setup_lcd();
    lcd_goto_xy(0,0);
    lcd_data("/**dem tu 00 den 99**");
}

```

```
lcd_command(0x40);
for (i=0;i<64;i++) { lcd_data(lcd_ma_8doan[i]); }
```

KHỞI TẠO CÁC KÝ TỰ TỰ TẠO

```
lcd_hienthi_so_z_toado_xy(0,2,0);
while(true)
{
    for (dem=0; dem<100; dem++)
    {
        lcd_hienthi_so_z_toado_xy(dem/10,2,0);
        lcd_hienthi_so_z_toado_xy(dem%10,2,3);
        delay_ms(500);
    }
}
```

GIẢI MÃ VÀ HIỂN THỊ

- d. Tiến hành biên dịch và nạp.
- e. Quan sát kết quả: nếu kết quả không đúng yêu cầu thì kiểm tra lại chương trình.
- f. Giải thích chương trình:

Bài tập 644. Hãy hiệu chỉnh bài 616 đếm sản phẩm hiển thị trên led 7 đoạn và LCD nhưng sử dụng kích thước lớn trên LCD.

Hàng 1 hiển thị thông tin: “**DEM SAN PHAM ** ”

Lưu tên file là “BAI_644_LCD_DEM_SP_KTL”

6.5 LÝ THUYẾT GLCD

6.5.1 Giới thiệu GLCD

Ở phần trên ta đã sử dụng hết các khả năng của LCD để hiển thị kí tự ASCII và kí tự tự tạo.

Phần này ta sẽ sử dụng GLCD có chức năng giống LCD và thêm khả năng hiển thị hình ảnh.

GLCD có rất nhiều dạng phân biệt theo kích thước điểm ảnh và GLCD có trong kit thực hành là GLCD 128x64 có nghĩa là có 128 điểm ảnh theo hàng và 64 điểm ảnh theo cột.

GLCD 128x64 như hình 6-18:

**Hình 6-18. Hình ảnh của GLCD.**

6.5.2 Sơ đồ chân của GLCD

Sơ đồ chân của GLCD như bảng sau.

Bảng 6-15. Các chân của GLCD.

Chân số	Tên chân	Input/output	Chức năng tín hiệu
1	VSS	Power	GND
2	VDD	Power	+5V
3	VO	Analog	Contrast Control
4	RS (CS_SPI)	Input	Register Select. H: data signal, L: instruction signal
5	R/W (SDI_SPI)	Input	Read/Write. H: read mode, L: write mode
6	E (SCK_SPI)	Input	Enable (strobe)
6-14	DB0 ~ DB7	H/L	Data bit 0~7
15	PSB (CS1)	Input	Interface selection (Chip select signal for IC1) 0: serial mode 1: 8/4 bits parallel bus mode
16	NC (CS2)	Input	No connection (Chip select signal for IC2)
17	\overline{RESET}	Input	Reset signal
18	V_{out} (V_{EE})	O	Output voltage for GLCD driving
15	BLA	input	Backlight Anode
16	BLK	input	Backlight Cathode

20 chân của GLCD được chia ra thành 4 dạng tín hiệu như sau:

Các chân cấp nguồn: Chân số 1 là chân V_{ss} sẽ nối mass (0V), chân thứ 2 là V_{dd} sẽ nối với nguồn +5V. Chân thứ 3 dùng để chỉnh contrast thường nối với biến trở, biến trở có 1 đầu

nối với 0V và điện áp ra ở chân số 18 là V_{out} , chỉnh biến trở cho đến khi thấy được kí tự thì ngừng, trong bộ thực hành thì đã chỉnh rồi.

Các chân BLA và BLK: là 2 chân dùng để cấp nguồn cho đèn nền để có thể nhìn thấy vào ban đêm. Giống LCD cũng được điều khiển bằng transistor để tắt mở đèn nền.

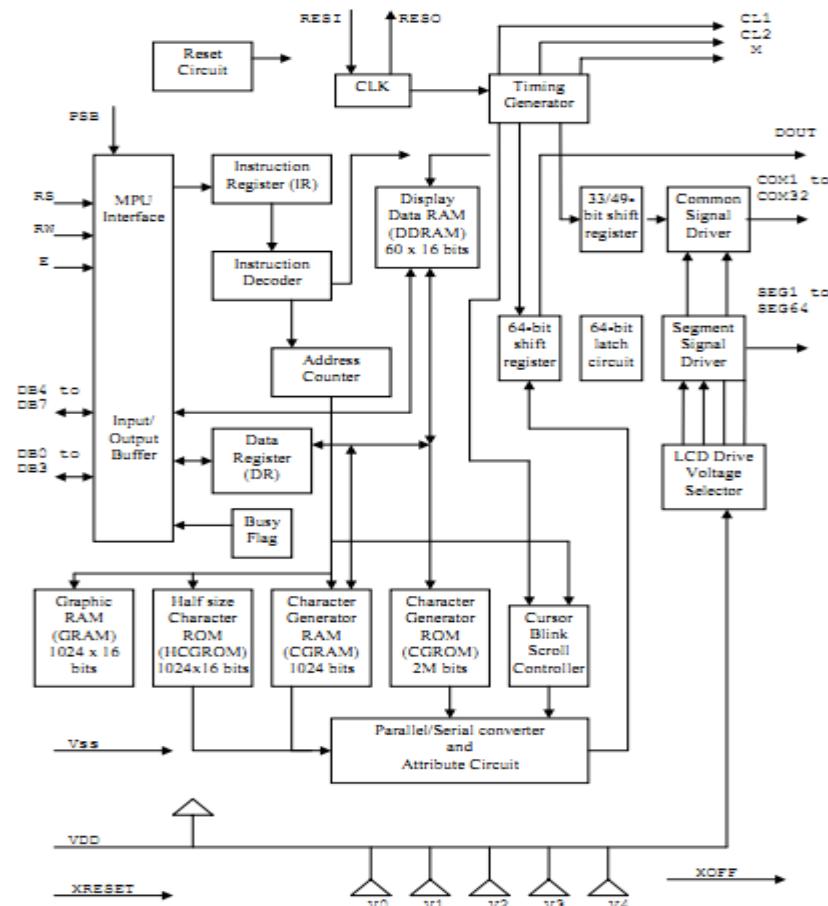
Các chân điều khiển: bao gồm tín hiệu RS dùng để điều khiển lựa chọn thanh ghi. Tín hiệu R/W dùng để điều khiển quá trình đọc và ghi. Tín hiệu E là chân cho phép dạng xung chốt. Ba tín hiệu này còn có chức năng giao tiếp chuẩn SPI ở một số GLCD. Hai tín hiệu PSB và tín hiệu reset tích cực mức thấp.

Các chân dữ liệu D7-D0: Chân số 7 đến chân số 14 là 8 chân dùng để trao đổi dữ liệu giữa thiết bị điều khiển và GLCD.

6.5.3 IC điều khiển GLCD ST7920

Để điều khiển GLCD, có các IC chuyên dùng được tích hợp bên dưới GLCD và có nhiều IC điều khiển, GLCD của kit thực hành thì dùng IC có mã số là ST7920.

Sơ đồ khối của IC ST7920 như hình 6-19.



Hình 6- 19. Cấu trúc IC ST7920.

Các khối bao gồm:

Khối giao tiếp với vi điều khiển có tên là MPU interface gồm có 5 tín hiệu RS, RW, E, PSB, RESET và 8 đường dữ liệu.

Có 2 thanh ghi gồm thanh ghi lệnh (Instruction Register) và thanh ghi dữ liệu (Data Register).

Vùng nhớ DDRAM chứa dữ liệu để hiển thị (Display Data RAM) giống LCD gồm 60 ô nhớ và mỗi ô nhớ là 16 bit (60×16 bit).

Vùng nhớ CGROM là vùng nhớ ROM phát kí tự (Character Generator ROM) có dung lượng 2M bit, vùng nhớ này giống LCD chứa sẵn các kí tự phổ biến.

Vùng nhớ CGRAM là vùng nhớ RAM phát kí tự tự tạo (Character Generator RAM) có dung lượng 1024 bit, vùng nhớ này giống LCD dành cho người dùng thiết lập các kí tự tự tạo.

Vùng nhớ HCGROM là vùng nhớ ROM phát kí tự nhưng kích thước mỗi kí tự giảm một nửa (Half Character Generator ROM) có dung lượng 1024 ô nhớ, mỗi ô 16 bit (1024×16 bit).

Vùng nhớ GRAM chứa dữ liệu hình ảnh để hiển thị (Graphic RAM) có dung lượng 1024 ô nhớ và mỗi ô nhớ chứa 16 bit (1024×16 bit).

6.5.4 Mô tả chức năng ic điều khiển GLCD ST7920

Giao tiếp với MPU

IC điều khiển ST7920 hỗ trợ 3 chế độ giao tiếp với MPU: chế độ giao tiếp song-song 8 bit, chế độ giao tiếp song – song 4 bit và chế độ giao tiếp đồng bộ nối tiếp SPI.

Tín hiệu PSB khi bằng 1 thì chọn chế độ giao tiếp song – song.

Tín hiệu PSB khi bằng 0 thì chọn chế độ giao tiếp nối tiếp SPI.

Còn chế độ giao tiếp song – song 8 bit hay 4 bit tùy thuộc vào bit DL trong lệnh thiết lập chức năng (Function set) – giống LCD.

Hai thanh ghi 8 bit DR và IR được dùng cho các hoạt động ghi và đọc của IC ST7920.

Thanh ghi dữ liệu DR dùng để truy xuất dữ liệu các vùng nhớ DDRAM/CGRAM/GRAM và IRAM với con trỏ địa chỉ lưu trong bộ đếm AC (Address Counter).

Thanh ghi lệnh IR dùng để lưu trữ lệnh từ MPU gửi đến ST7920.

Các vùng nhớ

Vùng nhớ phát kí tự CGROM 16×16 và HCGROM 8×16

IC ST7920 cung cấp CGROM lên đến 8192 font kí tự 16×16 và 126 kí tự 8×16 . Vùng nhớ này hỗ trợ cho ngôn ngữ Trung Quốc và tiếng Anh. Hai byte liên tiếp được dùng để chỉ định một kí tự 16×16 hoặc 2 kí tự 8×16 .

Các mã kí tự được ghi vào DDRAM và font tương ứng được ánh xạ từ vùng nhớ CGROM hoặc HCGROM để điều khiển hiển thị trên LCD.

Vùng nhớ chứa kí tự tự tạo CGRAM

IC ST7920 cung cấp cho người dùng khả năng tự tạo kí tự theo font mong muốn.

Vùng nhớ này có địa chỉ 6 bit nên có thể lưu được tối đa 4 mã kí tự tự tạo có kích thước 16×16 . 4 bit thấp từ 3 đến 0 dùng để định địa chỉ 16 từ dữ liệu của kí tự tự tạo, 2 bit 5 và 4 dùng để truy xuất 1 trong 4 kí tự.

4 kí tự tự tạo sẽ có mã hiển thị là 0000H, 0002H, 0004H và 0006H.

Khi tạo mã kí tự thì hàng cuối cùng thường để trắng dùng để nháy con trỏ.

Vùng nhớ DDRAM

DDRAM có 64×2 byte (64×16 bit) nên có thể lưu trữ 16 kí tự có kích thước 16×16 cho 1 hàng, hoặc 32 kí tự 8×16 cho 1 hàng. Các mã kí tự lưu trong vùng nhớ DDRAM trỏ đến vùng nhớ với font lưu trong CGROM, HCGROM và CGRAM.

ST7920 hiển thị font với độ cao phân nửa của vùng nhớ HCGROM, font do người dùng định nghĩa ở vùng nhớ CGRAM và font đầy đủ 16×16 ở vùng nhớ CGROM.

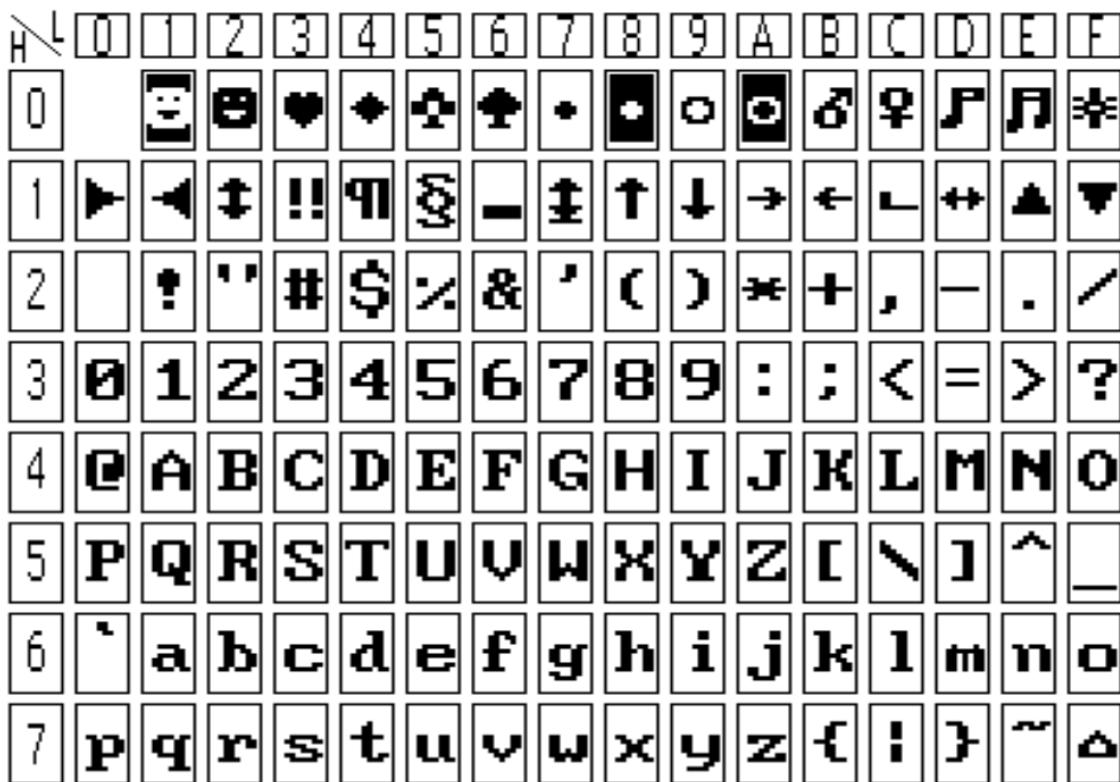
Các mã dữ liệu 02H đến 7FH được dùng cho font kí tự với độ cao giảm phân nửa.

Bảng 6-16. Các font chữ 16×16 của các vùng nhớ của GLCD.

DDRAM data (char. code)				CGRAM Addr.				CGRAM data (higher byte)				CGRAM data (lower byte)				
B15~B4	B3	B2	B1	B0	B5	B4	B3	B2	B1	B0	D1	D0	D1	D0	D1	D0
0	X	00	X	00	0	0	0	0	0	0	0	1	0	0	0	0
					0	0	1	1	1	1	1	1	0	0	1	0
					0	0	1	0	0	0	0	1	0	0	0	1
					0	0	1	0	0	0	1	0	0	0	1	0
					0	1	0	0	0	1	0	0	0	0	1	1
					0	1	0	0	0	0	1	0	0	1	1	1
					0	1	0	0	0	0	1	0	0	0	1	0
					0	1	0	1	0	1	1	1	1	0	0	0
					0	1	1	0	1	0	0	1	0	1	0	0
					0	1	1	1	0	1	0	0	1	0	0	1
					1	0	0	0	0	1	0	0	1	0	1	0
					1	0	0	1	0	0	1	0	0	0	1	0
					1	0	0	1	0	0	1	0	0	0	1	0
					1	0	1	0	0	1	0	0	0	0	1	0
					1	0	1	0	0	1	0	0	0	0	1	0
					1	1	0	0	0	1	0	0	1	0	0	0
					1	1	0	1	0	0	1	0	0	0	0	0
					1	1	1	0	0	1	0	0	1	0	0	0
					1	1	1	1	0	0	1	0	0	1	0	0
					1	1	1	1	0	0	1	0	0	0	1	0
					1	1	1	1	0	0	1	0	0	0	0	0
					1	1	1	1	0	0	1	0	0	0	0	0
					1	1	1	1	0	0	1	0	0	0	0	0
					1	1	1	1	0	0	1	0	0	0	0	0
					1	1	1	1	0	0	1	0	0	0	0	0
					1	1	1	1	0	0	1	0	0	0	0	0
					1	1	1	1	0	0	1	0	0	0	0	0
					1	1	1	1	0	0	1	0	0	0	0	0
					1	1	1	1	0	0	1	0	0	0	0	0
					1	1	1	1	0	0	1	0	0	0	0	0
					1	1	1	1	0	0	1	0	0	0	0	0
					1	1	1	1	0	0	1	0	0	0	0	0
					1	1	1	1	0	0	1	0	0	0	0	0
					1	1	1	1	0	0	1	0	0	0	0	0
					1	1	1	1	0	0	1	0	0	0	0	0
					1	1	1	1	0	0	1	0	0	0	0	0
					1	1	1	1	0	0	1	0	0	0	0	0
					1	1	1	1	0	0	1	0	0	0	0	0
					1	1	1	1	0	0	1	0	0	0	0	0
					1	1	1	1	0	0	1	0	0	0	0	0
					1	1	1	1	0	0	1	0	0	0	0	0
					1	1	1	1	0	0	1	0	0	0	0	0
					1	1	1	1	0	0	1	0	0	0	0	0
					1	1	1	1	0	0	1	0	0	0	0	0
					1	1	1	1	0	0	1	0	0	0	0	0
					1	1	1	1	0	0	1	0	0	0	0	0
					1	1	1	1	0	0	1	0	0	0	0	0
					1	1	1	1	0	0	1	0	0	0	0	0
					1	1	1	1	0	0	1	0	0	0	0	0
					1	1	1	1	0	0	1	0	0	0	0	0
					1	1	1	1	0	0	1	0	0	0	0	0
					1	1	1	1	0	0	1	0	0	0	0	0
					1	1	1	1	0	0	1	0	0	0	0	0
					1	1	1	1	0	0	1	0	0	0	0	0
					1	1	1	1	0	0	1	0	0	0	0	0
					1	1	1	1	0	0	1	0	0	0	0	0
					1	1	1	1	0	0	1	0	0	0	0	0
					1	1	1	1	0	0	1	0	0	0	0	0
					1	1	1	1	0	0	1	0	0	0	0	0
					1	1	1	1	0	0	1	0	0	0	0	0
					1	1	1	1	0	0	1	0	0	0	0	0
					1	1	1	1	0	0	1	0	0	0	0	0
					1	1	1	1	0	0	1	0	0	0	0	0
					1	1	1	1	0	0	1	0	0	0	0	0
					1	1	1	1	0	0	1	0	0	0	0	0
					1	1	1	1	0	0	1	0	0	0	0	0
					1	1	1	1	0	0	1	0	0	0	0	0
					1	1	1	1	0	0	1	0	0	0	0	0
					1	1	1	1	0	0	1	0	0	0	0	0
					1	1	1	1	0	0	1	0	0	0	0	0
					1	1	1	1	0	0	1	0	0	0	0	0
					1	1	1	1	0	0	1	0	0	0	0	0
					1	1	1	1	0	0	1	0	0	0	0	0
					1	1	1	1	0	0	1	0	0	0	0	0
					1	1	1	1	0	0	1	0	0	0	0	0
					1	1	1	1	0	0	1	0	0	0	0	0
					1	1	1	1	0	0	1	0	0	0	0	0
					1	1	1	1	0	0	1	0	0	0	0	0
					1	1	1	1	0	0	1	0	0	0	0	0
					1	1	1	1	0	0	1	0	0	0	0	0
					1	1	1	1	0	0	1	0	0	0	0	0
					1	1	1	1	0	0	1	0	0	0	0	0
					1	1	1	1	0	0	1	0	0	0	0	0
					1	1	1	1	0	0	1	0	0	0	0	0
					1	1	1	1	0	0	1	0	0	0	0	0
					1	1	1	1	0	0	1	0	0	0	0	0
					1	1</td										

Byte cao D15 đến D7 được gởi đến GLCD trước và sau đó là byte thấp từ D7 đến D0. Hãy tham khảo thêm bảng 6-16.

Bảng 6-17. Các font chữ 8x16 của GLCD.



Các ký tự có kích thước 8×16 nằm trong bảng sau:

Các font của CGRAM và CGROM chỉ có thể được hiển thị ở vị trí bắt đầu của mỗi địa chỉ như bảng 6-18.

Bảng 6-18. Địa chỉ gởi mã của font 16x16.

80	81	82	83	84	85	86	87	88	89	8A	8B	8C	8D	8E	8F
H L	H L	H L	H L	H L	H L	H L	H L	H L	H L	H L	H L	H L	H L	H L	H L
S i	t r	o n	i x		S T	7	9	2 0							
矽	創	電	子	.	.	中	文	編	碼	(正	確)		
矽	創	電	子	.	.	中	文	編	碼						

Table 4

Incorrect position

Trong hình trên thì để hiển thị ký tự kích thước 16×16 thì phải gởi 2 byte liên tiếp vào cùng 1 địa chỉ, ví dụ bạn phải gởi 2 byte cùng 1 địa chỉ 0x85 chứ không được gởi 1 byte ở địa chỉ 0x85 và 1 byte ở địa chỉ 0x86.

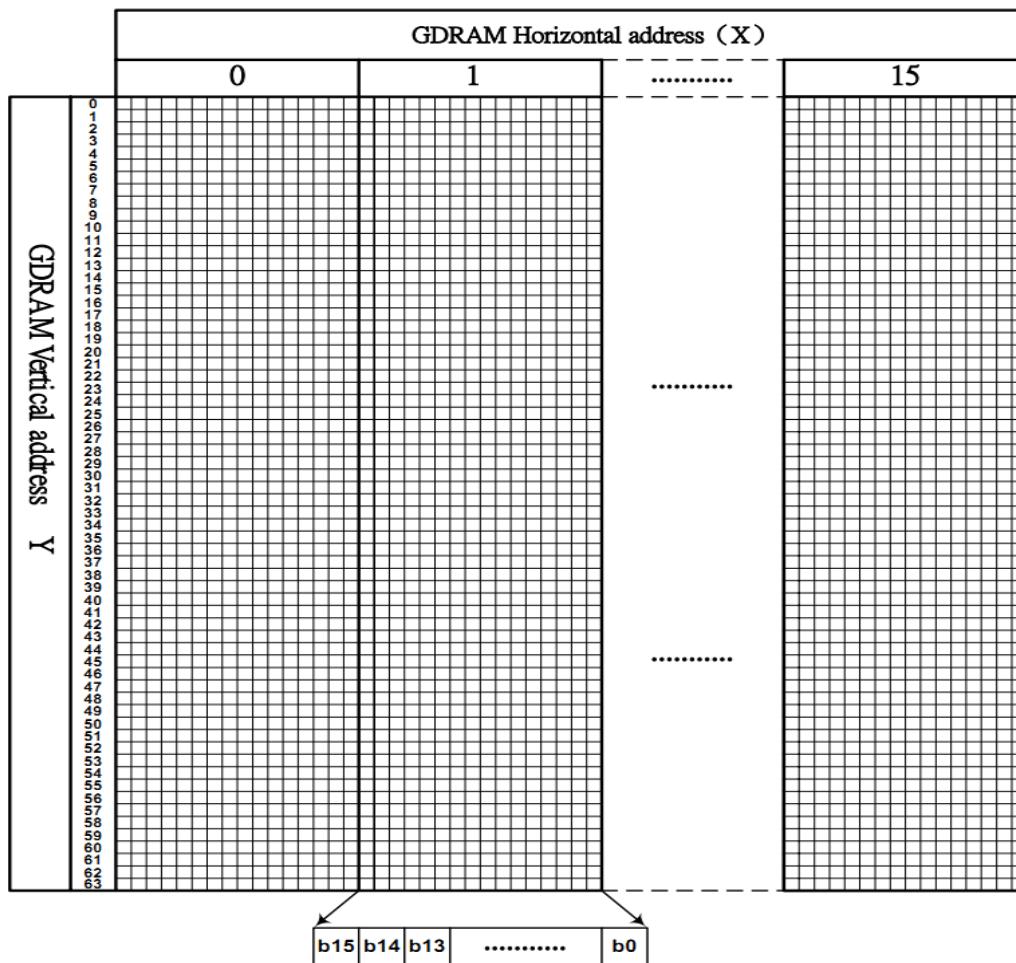
Vùng nhớ GDRAM

Vùng nhớ RAM hiển thị hình ảnh hỗ trợ không gian ánh xạ có kích thước 64×256 bit như hình 6-20.

Địa chỉ vùng nhớ GDRAM được thiết lập bằng cách ghi 2 byte liên tiếp cho địa chỉ chiều ngang và chiều dọc. Hai byte dữ liệu ghi vào GDRAM cho một địa chỉ. Bộ đếm địa chỉ tự động tăng lên 1 để lưu 2 byte tiếp theo.

Trình tự thực hiện như sau:

- Thiết lập địa chỉ dọc (Y) cho GDRAM
- Thiết lập địa chỉ ngang (X) cho GDRAM
- Ghi 8 bit D15 đến D8 vào GDRAM – byte thứ nhất
- Ghi 8 bit D7 đến D0 vào GDRAM – byte thứ hai

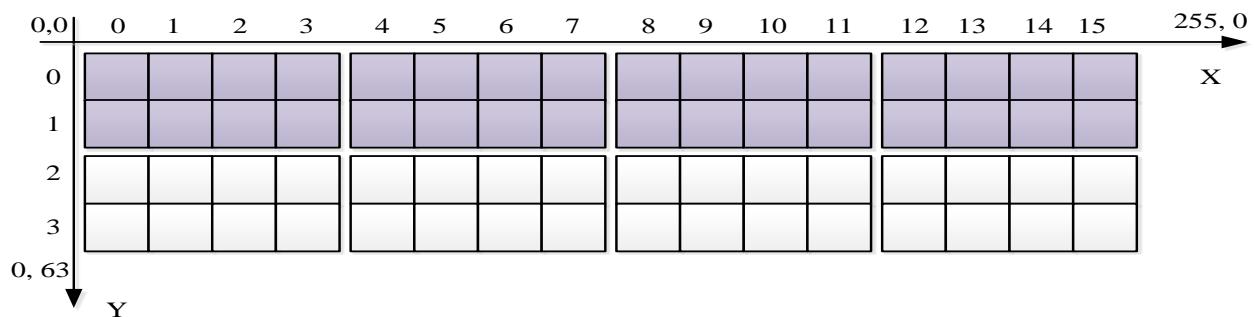


Hình 6-20. Tung độ hiển thị GDRAM và địa chỉ tương ứng của GLCD.

Hình 6-18 cho thấy địa chỉ theo chiều ngang x (horizontal address) là 16 (từ 0 đến 15) tính theo word 16 bit, tính theo bit thì bằng $16 \times 16 = 256$ bit.

Địa chỉ theo chiều dọc y (vertical address) là 64 bit (từ 0 đến 63).

Để giúp các bạn dễ hiểu thì hình 6-18 được vẽ lại như hình 6-21.

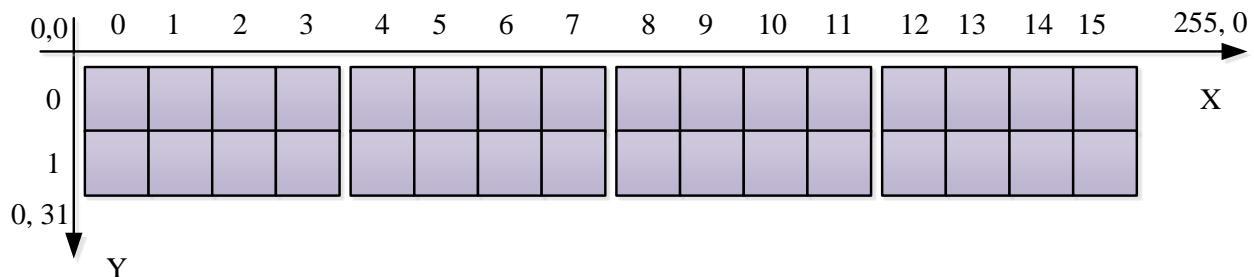
**Hình 6-21. Vẽ lại hình GDRAM theo khái bô nhớ.**

Theo cột thì vẫn là 16 cột, mỗi cột có dữ liệu là 16 bit tương ứng 16 pixel.

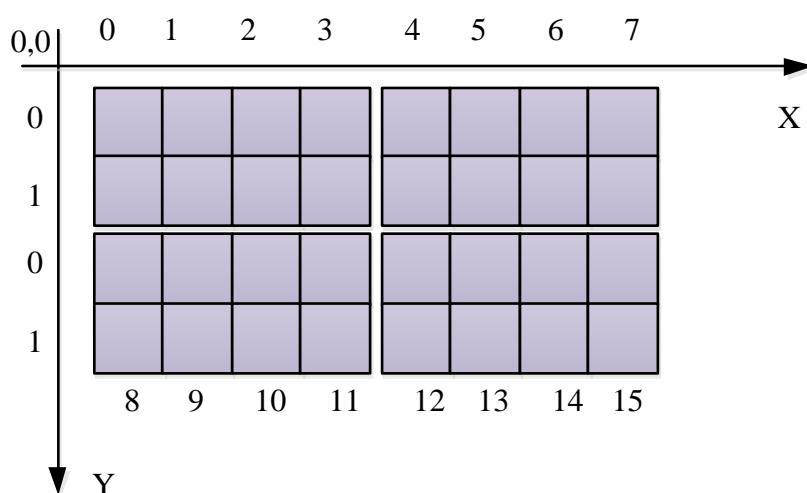
Theo hàng thì cũng chia ra thành 4 hàng, mỗi ô của 1 hàng có 16 bit tương ứng 16 pixel.

Số lượng pixel mà IC ST7920 có thể điều khiển là 64×256 .

Màn hình GLCD gắn ở kit thực hành thì có kích thước là 64×128 . Thực tế thì kích thước này đúng là 32×256 , điều đó có nghĩa là người ta cắt bỏ phần bên dưới và vùng nhớ hiển thị còn lại như hình 6-22.

**Hình 6-22. Bộ nhớ GDRAM của GLCD có kích thước 32×256 .**

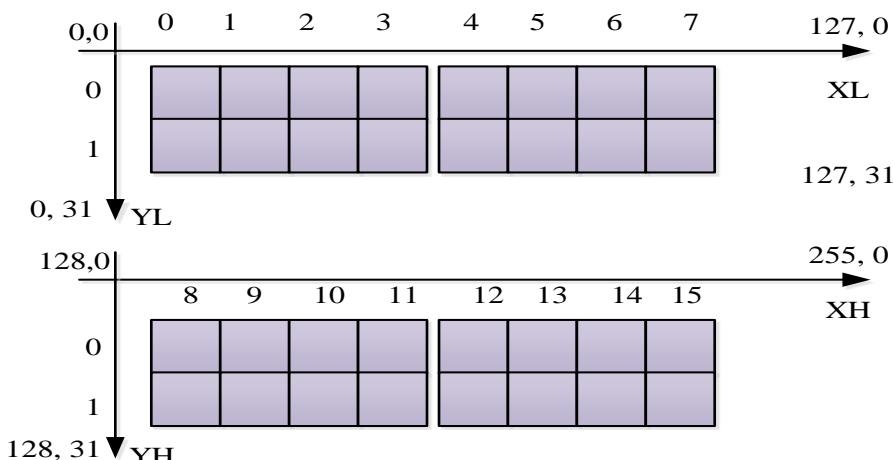
Khi chế tạo, GLCD có kích thước dài và mỏng nên thì người ta bố trí lại như hình 6-23.

**Hình 6-23. Bộ nhớ GDRAM của GLCD 32×256 bố trí theo kích thước 64×128 .**

Người ta cắt 8 vùng nhớ ở bên phải đem xuống đặt bên dưới, khi đó tính theo cách bố trí này thì GLCD có kích thước 64×128 .

Khi lập trình điều khiển 1 pixel hiển thị thì ta không thể tính theo kích thước 64×128 của hình 6-21 được mà phải tính theo kích thước 32×256 của hình 6-22.

Muốn lập trình cho tương thích với kích thước 64×128 , ta vẽ lại hình 6-21 như hình 6-24.



Hình 6-24. Bô nhớ GDRAM của GLCD 32×256 chi tiết.

Giới hạn các tọa độ pixel: Khi lập trình điều khiển 1 pixel hiển thị theo kích thước ($y \times x = 64 \times 128$) thì y phải nằm trong giới hạn từ 0 đến 63 và x từ 0 đến 127.

Khi muốn điều khiển hiển thị 1 pixel, ta tiến hành kiểm tra y: nếu y nằm trong phạm vi nhỏ hơn 32 thì điểm pixel đó nằm ở 2 hàng trên (XL, YL), nếu lớn 32 thì ta phải trừ đi cho 32 và tăng kích thước của x lên 128 và khi đó chúng sẽ nằm ở 2 hàng dưới (XH, YH).

Công việc ta vừa trình bày là quy đổi kích thước 64×128 về kích thước thực là 32×256 .

Các tính toán trên theo trực ngang là tính toán theo pixel, còn tính toán theo từng khói thì có 16 khói từ 0 đến 15. Trong hình thì các con số 0, 1, 2 ... là theo khói.

Chỉ có trực ngang là theo khói còn dọc thì không.

Tại sao lại quan tâm đến khói vì trong lệnh khói tạo địa chỉ cho trực ngang theo khói chỉ có 4 bit địa chỉ nên các tọa độ pixel theo kích thước sau khi chuyển đổi về 64×128 thì ta phải tiếp tục chuyển giá trị 128 về địa chỉ theo khói bằng cách lấy địa chỉ theo trực ngang chia cho 16.

6.5.5 Các lệnh điều khiển GLCD

IC ST7920 có 2 tập lệnh cơ bản và tập lệnh mở rộng như sau:

Tập lệnh cơ bản khi bit RE bằng 0.

Bảng 6-19. Tập lệnh điều khiển cơ bản của GLCD.

LỆNH	R S	R W	D 7	D6	D5	D4	D3	D2	D1	D0	Mô tả	cloc k
------	--------	--------	--------	----	----	----	----	----	----	----	-------	-----------

(1) Clear display	0	0	0	0	0	0	0	0	0	1		Write "20H" to DDRAM and set DDRAM address counter to "00H".	1.6m s
(2) Cursor home	0	0	0	0	0	0	0	0	1	0		Sets DDRAM address counter to zero, returns shifted display to original position. DDRAM contents remain unchanged.	72μs
(3) Entry mode set	0	0	0	0	0	0	0	1	I/D	S		Sets cursor move direction, and specifies automatic shift.	72μs
(4) Display control	0	0	0	0	0	0	1	D	C	B		Turns display (D), cursor on/off (C) or cursor blinking (B).	72μs
(5) Cursor display shift	0	0	0	0	0	1	S/C	R/L	0	0		Set cursor moving and display shift control bit, and the direction, without changing DDRAM data.	72μs
(6) Function set	0	0	0	0	1	DL	x	0 (RE)	0	0		Set interface data length (DL: 4-bit/8-bit). RE = 1 extended instruction. RE = 0 basic instruction.	72μs
(7) Set CGRA M addr	0	0	0	1	AC 5	AC 4	AC 3	AC 2	AC 1	AC 0		Set CGRAM address in address counter. Make sure that in extended instruction SR=0 (Scroll or RAM address select)	72μs
(8) Set DDRA M addr	0	0	1	0 AC 6	AC 5	AC 4	AC 3	AC 2	AC 1	AC 0		Set DDRAM address in address counter. AC6 is fixed to 0.	72μs
(9) Read Busy Flag and Address	0	1	B F	AC 6	AC 5	AC 4	AC 3	AC 2	AC 1	AC 0		Read busy flag (BF) for completion of internal operation, also read out the value of address counter (AC).	0μs
(10) Write	1	0	D 7	D6	D5	D4	D3	D2	D1	D0		Write data to CGRAM/ DDRAM/GDRAM	72μs

data to RAM													
(11) Read data from RAM	1	1	D7	D6	D5	D4	D3	D2	D1	D0	Read data from internal CGRAM/DDRAM/GDRAM	72μs	

Tập lệnh cơ bản khi bit RE bằng 1.

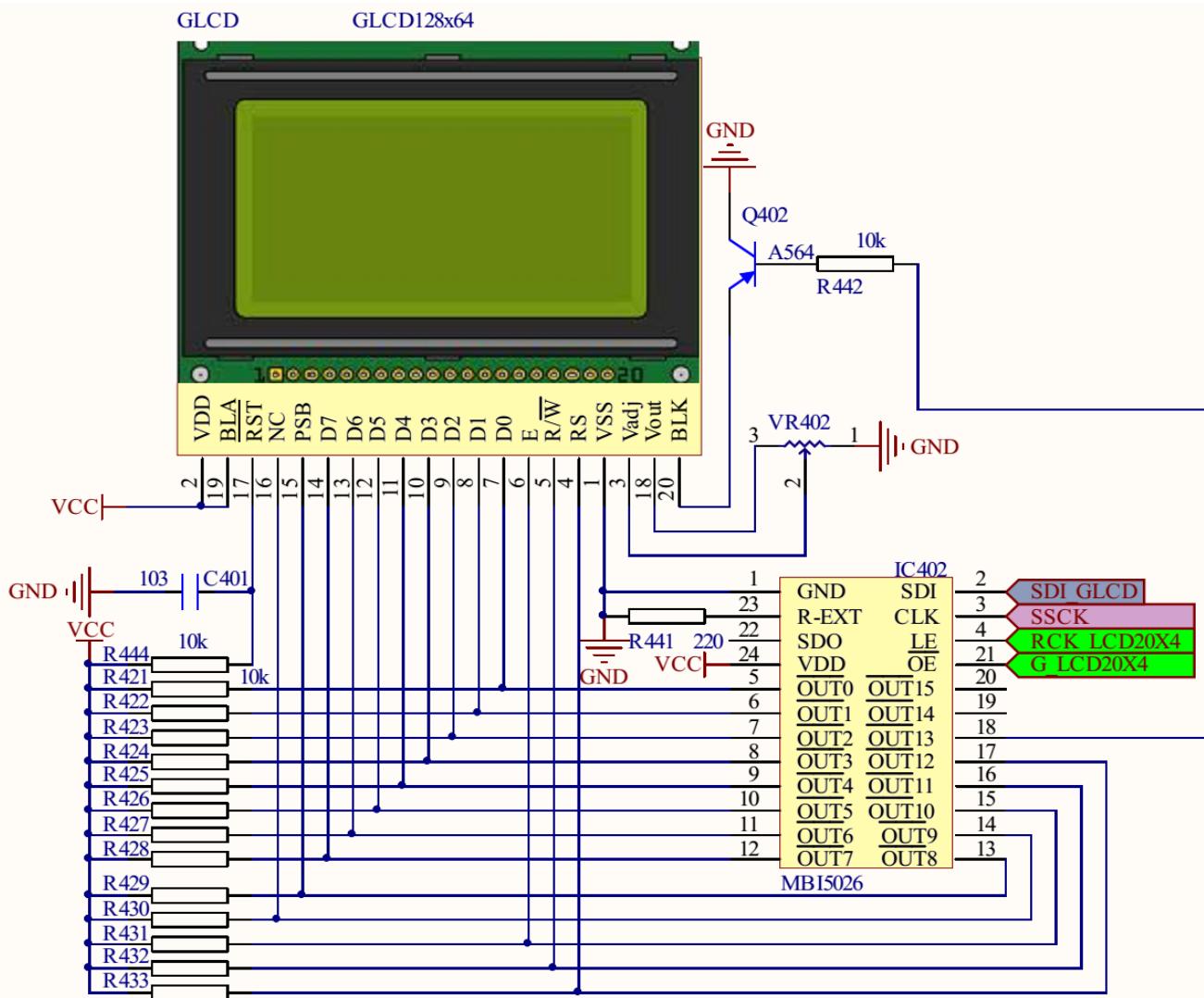
Bảng 6-20. Tập lệnh điều khiển mở rộng của GLCD.

LỆNH	RS	RW	D7	D6	D5	D4	D3	D2	D1	D0	Mô tả	clock
(1) Stand by	0	0	0	0	0	0	0	0	0	1	Enter standby mode, any others instruction can terminate (Com1 .. 32 halted)	1.6ms
(2) Scroll or RAM addr select	0	0	0	0	0	0	0	0	1	SR	SR = 1: enable vertical scroll position SR = 0: enable CGRAM address (basic instruction)	72μs
(3) reverse	0	0	0	0	0	0	0	1	R1	R0	Select 1 out of 4 line (in DDRAM) and decide whether to reverse the display by toggling this instruction R1, R0 initial value is 00	72μs
(4) Extended Function Set	0	0	0	0	1	DL	x	1 (RE)	G	0	Set interface data length (DL: 4-bit/8-bit). RE = 1 extended instruction. RE = 0 basic instruction. G = 1 graphic display on. G = 0 graphic display off.	72μs
(5) Set IRAM or scroll addr	0	0	0	1	AC5	AC4	AC3	AC2	AC1	AC0	SR = 1: AC5 – AC0 the address of vertical scroll.	72μs
(6) Set GRAPHIC RAM addr	0	0	1	0	AC5	AC4	AC3 AC3	AC2 AC2	AC1 AC1	AC0 AC0	Set GDRAM address to address counter (AC). First set vertical address and horizontal address by consecutive writing. Vertical address range AC5 ... AC0.	72μs

6.6 CÁC BÀI THỰC HÀNH DÙNG GLCD

6.6.1 Sơ đồ mạch của GLCD

Kit thực hành có kết nối 1 GLCD 128×64 qua IC mở rộng port là MBI 5026 có sơ đồ như hình sau:



Hình 6-25. Sơ đồ nguyên lý module GLCD 128×64.

Phần điều khiển GLCD đã trình bày ở phần LCD.

6.6.2 Các bài thực hành hiển thị ký tự trên GLCD

GLCD có khả năng hiển thị ký tự và hình ảnh, phần này thực hành các bài hiển thị ký tự trên GLCD giống như trên LCD.

Bài mẫu 651. Chương trình thư viện dùng cho GLCD đã có trong thư mục của bạn, bạn có thể sử dụng.

Tên file là “TV_PICKIT2_SHIFT_GLCD128X64.C”

- Mục đích: biết viết cách khai báo các biến, các tín hiệu điều khiển, các hàm xuất dữ liệu, xuất lệnh, hàm khởi tạo ... của GLCD.
- Chương trình:

```

unsigned int8 glcd_control;
#bit glcd_p      = glcd_control.5
#bit glcd_rs     = glcd_control.4
#bit glcd_rw     = glcd_control.3
#bit glcd_e      = glcd_control.2
#bit glcd_cs2    = glcd_control.1
#bit glcd_cs1    = glcd_control.0

//co 2 che do nen co 2 ham function set
#define glcd_text_mode          0x30
#define glcd_graphic_mode        0x36

#define glcd_clear_display       0x01
#define glcd_cursor_display      0x0c
#define glcd_entry_mode          0x06

#define glcd_addr_line1          0x80
#define glcd_addr_line2          0x90
#define glcd_addr_line3          0x88
#define glcd_addr_line4          0x98

void glcd_xuat_8bit(int8 glcd_data1)
{
    glcd_e=1; xuat_glcd128x64(glcd_control,glcd_data1);
    glcd_e=0; xuat_glcd128x64(glcd_control,glcd_data1);
}

void glcd_command(int8 glcd_data2)
{
    glcd_rs = 0;
    glcd_xuat_8bit(glcd_data2);
}

void glcd_data(int8 glcd_data3)
{
    glcd_rs = 1;
    glcd_xuat_8bit(glcd_data3);
}

void glcd_data_x(int8 glcd_data3)
{
    glcd_rs = 1;
    glcd_xuat_8bit(0xa1);
    glcd_xuat_8bit(0x40);
}

```

```

}

void setup_glcd (int8 glcd_function_set_mode)
{
    glcd_cs1 = 1;
    glcd_cs2 = 1;
    glcd_e = 0;
    glcd_rw = 0;

    glcd_p =0;

    glcd_command(glcd_function_set_mode);
    delay_us(10);
    glcd_command(glcd_cursor_display);
    delay_us(10);
    glcd_command(glcd_clear_display);
    delay_ms(2);
    glcd_command(glcd_entry_mode);
    delay_us(10);
}

#define     glcd_ngang      16 //256 pixel
#define     glcd_doc        32 //32 hang
int8 doc, ngang;

typedef union
{
    unsigned int16 word;
    unsigned int8 nbyte[2];
} dots;

typedef struct
{
    int1 refresh;
    dots pixel[glcd_doc][glcd_ngang];
} gd_ram;

gd_ram gdram_vdk;

void glcd_mau_nen(int1 glcd_color)
{
    int16 d;
    if(glcd_color) d=0xffff; else d=0;
    for (doc=0; doc < glcd_doc; doc++)
    {
        for (ngang=0; ngang < glcd_ngang; ngang++)
            gdram_vdk.pixel[doc][ngang].word = d;
    }
    gdram_vdk.refresh = true;
}

void glcd_pixel(int8 x, int8 y, int1 color)

```

```

{

    int8 b;
    if(y>31){x += 128; y-= 32;};
    doc = y;
    ngang = x/16;
    b = 15 - (x % 16);

    if (color == 1) bit_set (gdrum_vdk.pixel[doc][ngang].word, b);
    else           bit_clear (gdrum_vdk.pixel[doc][ngang].word, b);
    gdrum_vdk.refresh = true;
}

void gram_vdk_to_gram_glcd_all()
{
    if (gdrum_vdk.refresh)
    {
        for (doc = 0; doc <glcd_doc; doc++)
        {
            glcd_command( 0x80 | doc); // set vertical address.
            glcd_command( 0x80 | 0); // set horizontal address.

            for (ngang=0; ngang <glcd_ngang; ngang++)
            {
                glcd_data( gdrum_vdk.pixel[doc][ngang].nbyte[1]);
                glcd_data( gdrum_vdk.pixel[doc][ngang].nbyte[0]);
            }
        }
        gdrum_vdk.refresh = false;
    }
}

void gram_vdk_to_gram_glcd_area(int8 x,int8 y,int8 rong, int8 cao)
{
    int8 hesox,hesoy;
    if (gdrum_vdk.refresh)
    {
        if((x/16)!=(x + rong)/16) rong=((rong+15)/16)+1;
        else                         rong=((rong+15)/16);

        x = x/16;
        for (doc = y; doc <(y+cao); doc++)
        {
            if(doc>31)
            {
                hesox = 8+x;
                hesoy = doc-32;
            }
            else {hesox = x; hesoy = doc;}
            glcd_command( 0x80 |hesoy);
            glcd_command( 0x80 |hesox);
        }
    }
}

```



```

//!0x03, 0xF0, 0x00, 0x01, 0xC0, 0x00, 0xF, 0xC0, 0x03, 0xE0, 0x00, 0x01, 0xC0,
0x00, 0x07, 0xE0,
//!0x07, 0xC0, 0x00, 0x03, 0xC0, 0x00, 0x03, 0xF0, 0x0F, 0xC0, 0x00, 0x17, 0xF4,
0x00, 0x01, 0xF0,
//!0x0F, 0x80, 0x00, 0x1D, 0x98, 0x00, 0x00, 0xF8, 0x1F, 0x00, 0x00, 0x11, 0x8C,
0x00, 0x00, 0xF8,
//!0x1F, 0x01, 0x00, 0x31, 0x84, 0x00, 0x80, 0x7C, 0x3E, 0x06, 0x00, 0x31, 0x86,
0x00, 0x20, 0x7C,
//!0x3E, 0x18, 0x00, 0x71, 0x86, 0x00, 0x1C, 0x3E, 0x3C, 0x43, 0x00, 0x31, 0x84,
0x00, 0xC3, 0x3E,
//!0x7C, 0xC, 0x00, 0x11, 0x8C, 0x00, 0x38, 0x1E, 0x7C, 0x31, 0x00, 0x1D, 0x98,
0x00, 0x86, 0x1F,
//!0x78, 0xC6, 0x00, 0x1F, 0xFC, 0x00, 0x63, 0x1F, 0x78, 0x18, 0x00, 0x03, 0xC0,
0x00, 0x0C, 0x1F,
//!0xF8, 0x63, 0x80, 0x01, 0x80, 0x00, 0xC7, 0x0F, 0xF9, 0x8E, 0x00, 0x00, 0x00,
0x00, 0x38, 0x8F,
//!0xF8, 0x38, 0x80, 0x00, 0x00, 0x01, 0x0C, 0x0F, 0xF8, 0xC3, 0x80, 0x00, 0x00,
0x00, 0xE3, 0x8F,
//!0xF9, 0x8E, 0x00, 0x00, 0x00, 0x30, 0xCF, 0xF8, 0x38, 0xF8, 0x00, 0x00,
0x1F, 0x9E, 0x0F,
//!0xF8, 0xE3, 0xFE, 0x00, 0x00, 0x7F, 0xE3, 0x8F, 0xF9, 0x9F, 0xFF, 0xC0, 0x03,
0xFF, 0xF9, 0xCF,
//!0x78, 0xFF, 0xFF, 0xE0, 0x07, 0xFF, 0x0F, 0x78, 0x1F, 0xFF, 0xE0, 0x07,
0xFF, 0xFC, 0x1F,
//!0x7C, 0x07, 0xFF, 0xE0, 0x07, 0xFF, 0xE0, 0x1F, 0x7C, 0x00, 0xFF, 0xE0, 0x07,
0xFF, 0x80, 0x1E,
//!0x7C, 0x00, 0x1F, 0xE0, 0x07, 0xFC, 0x00, 0x3E, 0x3E, 0x01, 0xCF, 0xE0, 0x07,
0xF3, 0x80, 0x3E,
//!0x3E, 0x01, 0xFF, 0xE0, 0x03, 0xFF, 0x80, 0x7C, 0x1F, 0x01, 0xFF, 0xE0, 0x03,
0xFF, 0xC0, 0x7C,
//!0x1F, 0x01, 0xFF, 0xC0, 0x03, 0xFF, 0xC0, 0x0F, 0x83, 0xFF, 0xC0, 0x01,
0xFF, 0xC0, 0x8F,
//!0x0F, 0x83, 0xFF, 0xC0, 0x01, 0xFF, 0xC1, 0xF0, 0x07, 0xC3, 0xFF, 0x80, 0x01,
0xFF, 0xE3, 0xF0,
//!0x03, 0xE3, 0xFF, 0x80, 0x01, 0xFF, 0xE7, 0xE0, 0x03, 0xF7, 0xFF, 0x80, 0x00,
0xFF, 0xEF, 0xC0,
//!0x01, 0xFF, 0xFF, 0x80, 0x00, 0xFF, 0xFF, 0x80, 0x00, 0xFF, 0xFF, 0x00, 0x00,
0xFF, 0xFF, 0x80,
//!0x00, 0x7F, 0xFF, 0x00, 0x00, 0xFF, 0x00, 0x00, 0x3F, 0xFF, 0x00, 0x00,
0xFF, 0xFE, 0x00,
//!0x00, 0x1F, 0xFF, 0x00, 0x00, 0x7F, 0xF8, 0x00, 0x00, 0x0F, 0xFE, 0x00, 0x00,
0x7F, 0xF0, 0x00,
//!0x00, 0x03, 0xFF, 0x00, 0x00, 0x7F, 0xE0, 0x00, 0x00, 0x00, 0xFF, 0xFC, 0x1F,
0xFF, 0x80, 0x00,
//!0x00, 0x00, 0x7F, 0xFF, 0xFE, 0x00, 0x00, 0x00, 0x00, 0x00, 0x0F, 0xFF, 0x00,
0x8F, 0x00, 0x00,
//!0x00, 0x00, 0x01, 0xFF, 0xFF, 0xC0, 0x00, 0x00, 0x00, 0x00, 0x00, 0x0F, 0x00,
0x00, 0x00, 0x00
//!};

void glcd_xuat_anh(int width,int height,int x,int y)
{
    unsigned int i=0, j=0, k=0;
    unsigned int16 count=0;

```

```

for(j=0;j<height;j++)
{
    for(;i<width;)
    {
        for(k=8;k>0;k--)
        {
            glcd_pixel(i+x,j+y,bit_test(Logo_DHSPKTTP[count],
                (k-1)));
            i++;
        }
        count++;
    }
    i=0;
}

```

c. Giải thích chương trình:

Phần thứ nhất: là định nghĩa dữ liệu điều khiển GLCD:

Phần này giống LCD. Hàm Function Set (FS) có 2 lệnh vì GLCD có 2 chế độ hiển thị giống LCD và chế độ hiển thị hình ảnh được lựa chọn bởi 1 bit nằm trong lệnh FS.

Ở chế độ LCD thì địa chỉ của 4 hàng khác với LCD.

Phần thứ hai: là hàm khởi tạo GLCD.

Có các hàm giống LCD.

Phần thứ ba: là định nghĩa dữ liệu điều khiển GLCD ở chế độ hình ảnh:

```

#define     glcd_ngang    16 //256 pixel
#define     glcd_doc      32 //32 hang
int8 doc, ngang;

typedef union
{
    unsigned int16 word;
    unsigned int8 nbyte[2];
} dots;

typedef struct
{
    int1 refresh;
    dots pixel[glcd_doc][glcd_ngang];
} gd_ram;

gd_ram  gdram_vdk;

```

Hai hàng đầu là định nghĩa giới hạn kích thước thực của GLCD là 32×256

Hàng thứ 3 định nghĩa các biến ngang và đọc 8 bit.

Định nghĩa tập hợp dots 2 thành phần có tên là word kiểu dữ liệu 16 bit và mảng nbyte 2 phần tử.

Định nghĩa kiểu dữ liệu GD_RAM theo cấu trúc gồm thuộc tính refresh kiểu 1 bit và PIXEL là mảng 2 chiều với kiểu dữ liệu là dots, mảng có kích thước với các phần tử đã định nghĩa ở trên.

Hàng cuối cùng là định nghĩa biến gdram_vdk thuộc kiểu dữ liệu ta vừa định nghĩa là GD_RAM. Biến gdram_vdk có kích thước $32 \times 256 = 8192$ bit hay byte 1024 để chứa dữ liệu hiển thị hình ảnh cho GLCD.

Việc định nghĩa vùng nhớ gdram_vdk này là có 2 lý do:

Lý do thứ nhất là dữ liệu của vùng nhớ GDRAM của vi điều khiển mỗi lần gởi là 16 bit hay 2 byte tương ứng với 16 pixel cùng 1 lúc. Ví dụ, ta làm 1 pixel sáng và giữ nguyên trạng thái các pixel lân cận nhưng lệnh gởi dữ liệu ra lại gởi một lần cả 16 pixel cho cả 1 cột như hình 6-8. Việc này làm ảnh hưởng đến các pixel khác.

Lý do thứ hai là do vi điều khiển giao tiếp với GLCD chỉ có một chiều là vi điều khiển chỉ gởi dữ liệu ra GLCD chứ không cho phép đọc về. Nếu có chế độ cho phép đọc về thì ta có thể đọc dữ liệu của hàng đó về xong rồi cập nhật 1 bit tương ứng với pixel ta muốn rồi sau đó ghi trở lại.

Vậy giải pháp cuối cùng là ta phải dùng một vùng nhớ của vi điều khiển PIC 18F4550 gdram_vdk để chứa các dữ liệu tương ứng với dữ liệu hiển thị trên màn hình GLCD.

Các dữ liệu cần điều khiển pixel hay các hình vẽ hay hình ảnh cần hiển thị trên GLCD thì trước tiên ta phải ghi vào vùng nhớ gdram_vdk rồi sau đó mới gởi ra GLCD.

Phản thứ tự: là hàm gởi dữ liệu từ vùng nhớ gdram_vdk sang vùng nhớ gdram_lcd:

```
void gram_vdk_to_gram_lcd_all()
{
    if (gdram_vdk.refresh)
    {
        for (doc = 0; doc < glcd_doc; doc++)
        {
            glcd_command( 0x80 | doc); // set vertical address.
            glcd_command( 0x80 | 0); // set horizontal address.

            for (ngang=0; ngang < glcd_ngang; ngang++)
            {
                glcd_data( gdram_vdk.pixel[doc][ngang].nbyte[1]);
                glcd_data( gdram_vdk.pixel[doc][ngang].nbyte[0]);
            }
        }
        gdram_vdk.refresh = false;
    }
}
```

Đầu tiên là kiểm tra trạng thái, nếu bằng true thì tiến hành gởi.

Để gởi tất cả 1024 byte dữ liệu thì thực hiện bằng các lệnh như sau:

Vòng lặp for thứ nhất theo biến doc từ 0 đến 31:

Khi biến doc = 0 thì:

- Có 2 lệnh gửi địa chỉ ra GLCD để thiết lập địa chỉ, lệnh thứ nhất thiết lập địa chỉ theo trục dọc (vertical) và lệnh thứ 2 thiết lập địa chỉ theo chiều ngang (horizontal) ở dạng khối và luôn bằng 0.
- Vòng lặp for thứ 2 theo biến ngang từ 0 đến 15:
 - Khi biến ngang = 0 thì tiến hành lấy lần lượt 2 byte gửi ra cho khối thứ nhất
 - Địa chỉ khối tự động tăng lên 1
 - Khi biến ngang = 1 thì gửi 2 byte cho khối thứ 2
 - Và cứ thế cho đến khi hết 16 khối. Nếu chưa rõ thì xem hình 6-11.

Tương tự khi biến doc = 1 cho đến khi bằng 31 thì toàn bộ dữ liệu đã được chuyển từ vi điều khiển sang vùng nhớ GDRAM của GLCD.

Phản thứ năm: là hàm xử lý 1 pixel chỉ có 2 trạng thái là on và off ở vùng nhớ gdram_vdk:

```
void lcd_pixel(int8 x, int8 y, int1 color)
{
    int8 b;
    if(y>31){x += 128; y-= 32;};
    doc = y;
    ngang = x/16;
    b = 15 - (x % 16);

    if (color == 1) bit_set (gdram_vdk.pixel[doc][ngang].word, b);
    else           bit_clear (gdram_vdk.pixel[doc][ngang].word, b);
    gdram_vdk.refresh = true;
}
```

Hàm có chức năng làm đổi màu xanh hay trắng của 1 pixel ở tọa độ x,y.

Đầu tiên là công việc chuyển kích thước từ 64×128 về kích thước 32×256 như đã trình bày ở trên bằng cách kiểm tra giá trị doc của y: nếu lớn hơn 32 thì tăng x lên thêm 128 và y trừ đi 32 để chuyển sang XH và YH, nếu nhỏ hơn thì trong hệ XL và YL.

Gán giá trị y cho biến DOC và lại chuyển giá trị x theo pixel về khối bằng cách chia cho 16 như đã trình bày ở trên. Gán giá trị x chia cho 16 cho biến NGANG.

Đến đây ta có 2 thông tin là NGANG và DOC để xác định được từ dữ liệu 16 bit chứa pixel cần xử lý nhưng có tới 16 pixel trong khi ta chỉ xử lý có 1 pixel.

Vậy pixel ta cần xử lý xác định như thế nào? Đó chính là giá trị dư của lần biến đổi pixel theo trục ngang gán cho biến NGANG và đó chính là lệnh “**b = 15 - (x % 16);**”

Để dễ hiểu ta cho ví dụ cần làm pixel có tọa độ (x, y) = (100, 15) sáng.

Với y bằng 15 thì chúng nằm trong vùng XL và YL.

NGANG bằng $100/16 = 6$, b = $15 - (100 \% 16) = 11$.

Vậy ta làm bit thứ 11 của từ dữ liệu 16 bit nằm ở khối thứ 5 (do tính từ 0) và theo chiều dọc là thứ 14 (do tính từ 0).

Lệnh if còn lại kiểm tra màu để thiết lập giá trị cho bit của pixel đó.

Sau khi có 1 pixel được ghi vào thì trạng thái sẽ chuyển sang true để cho phép cập nhật sang GLCD.

Phản thứ sáu: là hàm chuyển dữ liệu mảng chứa file ảnh sang vùng nhớ GRAM_VDK cũng thực hiện tương tự bằng cách gọi hàm lcd_pixel.

Các mảng dữ liệu hình ảnh logo Sư Phạm Kỹ Thuật và bóng đèn.

Bài mẫu 652. Chương trình hiển thị chuỗi trên GLCD font 16×16.

Hiển thị 4 hàng thông tin xem trong chương trình.

Lưu tên file là “BAI_651_GLCD_HIEN_THI_CHUOI”

- Mục đích: biết lập trình hiển thị chuỗi trên GLCD.
- Lưu đồ: sinh viên hãy tự viết.
- Chương trình:

```
#include <tv_pickit2_shift_1.c>
#include <tv_pickit2_shift_glcd128x64.c>

void main()
{
    set_up_port_ic_chot();
    setup_glcd(glcd_text_mode);

    glcd_command(glcd_addr_line1);
    glcd_data("dh-supham kt hcm");

    glcd_command(glcd_addr_line2);
    glcd_data("bo mon dien tu");

    glcd_command(glcd_addr_line3);
    glcd_data("cong nghiep ");

    glcd_command(glcd_addr_line4);
    glcd_data("thuc hanh vdk");

    while(true);
}
```

- Tiến hành biên dịch và nạp.
- Quan sát kết quả: bạn có thể thay đổi nội dung tùy ý.
- Giải thích chương trình: chương trình gần giống như LCD.

Bài mẫu 653. Chương trình đếm sản phẩm dùng counter T0 và hiển thị trên GLCD font 16×16.

Các thông tin phụ thì xem trong chương trình.

Lưu tên file là “BAI_653_GLCD DEM SP_T0”

- Mục đích: biết lập trình đếm sản phẩm và giải mã hiển thị trên màn hình GLCD.
- Lưu đồ: sinh viên hãy tự viết.
- Chương trình:

```
#include <tv_pickit2_shift_1.c>
#include <tv_pickit2_shift_glcd128x64.c>
unsigned int8 t0, t0_tam;
void glcd_hienthi_t0(unsigned t0)
{
    glcd_command(glcd_addr_line2);
    glcd_data(t0/100%10 + 0x30);
    glcd_data(t0/10%10 + 0x30);
    glcd_data(t0%10 + 0x30);
}

void main()
{
    set_up_port_ic_chot();
    setup_glcd(glcd_text_mode);

    glcd_command(glcd_addr_line1);
    glcd_data("dem san pham:");

    setup_timer_0 (t0_ext_l_to_h | t0_div_1|t0_8_bit);
    set_timer0(0);
    t0 = t0_tam = 0;
    glcd_hienthi_t0(t0);

    while(true)
    {
        t0=get_timer0();
        if (t0!= t0_tam)
        {
            glcd_hienthi_t0(t0);
            t0_tam = t0;
        }
        xuat_4led_7doan_giaima_xoa_so0(t0);
        if (t0>=101)    set_timer0(1);
    }
}
```

- Tiến hành biên dịch và nạp.
- Quan sát kết quả: bạn hãy tạo xung và quan sát giá trị đếm trên GLCD.
- Giải thích chương trình: chương trình gần giống như LCD.

Bài tập 654. Hãy viết chương trình đếm sản phẩm dùng counter T0, hiển thị kết quả đếm trên 4 led 7 đoạn, LCD và GLCD.

Mục đích: biết lập trình đếm sản phẩm và giải mã hiển thị trên nhiều module.

Lưu tên file là “BAI_654_GLCD_LCD_4LED_DEM_SP_T0”

6.6.3 Các bài thực hành hiển thị hình ảnh trên GLCD

Phần này thực hành các bài hiển thị hình ảnh trên GLCD.

Để hiển thị hình ảnh số thì ta phải biết hình ảnh có được từ nhiều nguồn thiết bị khác nhau như máy chụp ảnh hay ảnh chụp trên màn hình máy tính hay do phần mềm vẽ tạo ra lưu trên máy tính với nhiều tên mở rộng khác nhau, ... và muốn hiển thị ảnh trên GLCD thì ta phải dùng phần mềm chuyển đổi ảnh cần hiển thị sang file nhị phân (viết gọn ở dạng số hex) – tạm gọi là file ảnh, sau đó ta viết một chương trình gõi lần lượt các mã nhị phân của file ảnh này ra GLCD.

GLCD của kit thực hành là loại đơn sắc chỉ có 2 màu xanh và trắng và ta có thể lập trình chọn 1 trong 2 chế độ màu này.

Để hiển thị 1 ảnh thì về cơ bản là ta vẽ lại từng điểm ảnh (pixel) trên màn hình GLCD và diện tích ảnh trên GLCD đúng với ảnh ta cần hiển thị.

Bài mẫu 661. Chương trình hiển thị logo đại học SPKT trên GLCD.

Lưu tên file là “BAI_661_GLCD_BITMAP_SPKT”

- a. Mục đích: biết lập trình hiển thị hình ảnh 2 màu trên màn hình GLCD.
- b. Lưu đồ: sinh viên hãy tự viết.
- c. Chương trình:

```
#include<tv_pickit2_shift_1.c>
#include <tv_pickit2_shift_lcd128x64.c>
void main()
{
    set_up_port_ic_chot();
    setup_lcd(lcd_graphic_mode);
    lcd_mau_nen(0);
    lcd_xuat_anh(64,64,32,0);
    gdram_vdk_to_gdram_lcd_all();
    while(true);
}
```

- d. Tiến hành biên dịch và nạp.
- e. Quan sát kết quả: bạn sẽ nhìn thấy biểu tượng trường màu trắng trên nền xanh của GLCD.
- f. Giải thích chương trình: trong chương trình chính thực hiện các chức năng là khởi tạo hệ thống port, IC chốt sau đó khởi tạo GLCD chế độ hiển thị hình ảnh (làm thay đổi bit RE để sử dụng tập lệnh mở rộng).

Gọi hàm thiết lập màu nền là màu xanh với thông số là 0. Nền màu trắng thông số là 1.

Gọi hàm xuất dữ liệu của ảnh ra tọa độ (0,0) với chiều cao và chiều dài ảnh là 64. Cuối cùng gọi hàm cập nhật ảnh để hiển thị.

File ảnh logo đã chuyển đổi thành file nhị phân và lưu trong file thư viện.

Bạn hãy xem các hàm trong thư viện để hiểu thêm chi tiết.

Bài tập 662. Hãy viết chương trình hiển thị 2 biểu tượng SPKT trên GLCD ở 2 bên cho cân xứng.

Lưu tên file là “BAI_662_GLCD_2BITMAP_SPKT”

Bài mẫu 663. Chương trình vẽ đường thẳng, đường tròn trên GLCD.

Lưu tên file là “BAI_663_GLCD_DT_DT”

- Mục đích: biết lập trình vẽ đường thẳng hiển thị trên màn hình GLCD.
- Lưu đồ: sinh viên hãy tự viết.
- Chương trình:

```
#include<tv_pickit2_shift_1.c>
#include <tv_pickit2_shift_glcd128x64.c>
#include <graphics.c>
void main()
{
    set_up_port_ic_chot();
    setup_glcd(glcd_graphic_mode);
    glcd_mau_nen(0);
    glcd_line(0, 0, 50, 50, 1);
    glcd_circle(40, 40, 20, 0, 1);

    gdram_vdk_to_gdram_glcd_all();
    while(true);
}
```

hiển thị hình (graphic)
setup_glcd..
gdram_vdk...

- Tiến hành biên dịch và nạp.
- Quan sát kết quả: bạn sẽ nhìn thấy biểu tượng trường màu trắng trên nền xanh của GLCD.
- Giải thích chương trình:

Giống các bài trước, gọi hàm vẽ đường thẳng và hàm vẽ đường tròn.
Các hàm này đã vẽ sẵn trong thư viện graphics.c do CCS hỗ trợ.
Hàm `glcd_line (x1,y1,x2,y2,color)`: gồm 5 thông số tọa độ bắt đầu và kết thúc của đường thẳng nằm trong phạm vi 128, 64 của màn hình, color thì chọn màu trắng trên nền xanh, nếu bạn chọn là số 0 thì màu xanh trên nền xanh ta không thể nhìn thấy được.
Hàm `glcd_circle (x1,y1,r,fill,color)`: gồm 5 thông số tọa độ tâm, bán kính, có tô màu cả hình tròn hay không và màu.
Bạn hãy xem thêm các hàm trong thư viện để hiểu thêm chi tiết và khai thác chúng cho hiệu quả.

Bài tập 664. Hãy viết chương trình đếm sản phẩm dùng counter T0, hiển thị kết quả đếm trên GLCD dạng hình ảnh theo cột, mỗi cột 10 đoạn tương ứng 10 số, 3 cột hàng

trăm, hàng chục, hàng đơn vị. Khi cột hàng đơn vị đếm đến 10 rồi quay về 0 thì cột chục tăng lên 1, tương tự cho các giá trị còn lại. Sử dụng **hàm vẽ hình chữ nhật** để lập trình. Nếu chiều cao của cột ngắn thì bạn có thể hiển thị theo chiều dài.

Lưu tên file là “BAI_664_GLCD_DEM_SP_T0_HT_COT”

mở file thư viện
<graphics.c>

xem bài giới thiệu

Bài tập 665. Hãy viết chương trình đếm giây hiển thị trên màn hình giống đồng hồ kim. Vẽ vòng tròn và vẽ kim giây, sau 1 giây thì xóa kim giây hiện tại, thay đổi tọa độ cuối vẽ kim giây mới, sử dụng timer T1 định thời để đếm chính xác.

Lưu tên file là “BAI_665_GLCD_DEM_GIAY”

6.7 KẾT HỢP NHIỀU MODULE

Ở trên ta đã kết hợp được 3 module gồm GLCD, LCD và 4 led 7 đoạn để thực hiện bài đếm sản phẩm, phần này kết hợp thêm nhiều module.

Bài mẫu 671. Kết hợp nhiều module.

Module 32 led đơn thì chạy đèn quảng cáo.

Module 8 led quét thì đếm giờ phút giây.

Module LCD, GLCD, 4 led thì hiển thị kết quả đếm sản phẩm từ counter T0

Lưu tên file là “BAI_671_TONGHOP_5_MODULE”.

- Mục đích: Biết cách lập trình kết hợp nhiều module.
- Lưu đồ: sinh viên hãy tự viết.
- Chương trình:

```
#include <tv_picket2_shift_1.c>
signed int8 gio, phut, giay, bdn, gia_tri_mod, dem_tg_exit=0;
unsigned int8 t0, t0_tam;
void delay_quet_8led(unsigned int8 d1);
#include <tv_picket2_shift_32led_don.c>
#include <tv_picket2_shift_glcd128x64.c>
#include <tv_picket2_shift_lcd.c>

int_timer1
void interrupt_timer1()
{
    bdn++;
    set_timer1(3036);
}

void glcd_hienthi_t0(unsigned t0)
{
    glcd_command(glcd_addr_line2);
    glcd_data(t0/100%10 + 0X30);
    // ...
}
```

20 /// glcd_rect(x1, y1, x2, y2, fill, color)
 21 /// * Draws a rectangle with one corner at point (x1,y1) and
 22 /// the other corner at point (x2,y2)
 23 /// - fill can be YES or NO
 24 /// - color can be ON or OFF
 25 /// (x1,y1) (x2,y2)
 26 /// 128 x 64 - GLCD
 27 /// cột x hàng

37 /// glcd_text57(x, y, textptr, size, color)
 38 /// * Write the character
 39 /// Char - size - color
 40 /// + Note
 41 /// thay đổi kích thước chữ, số
 42 /// textptr: text pointer - con trỏ
 43 /// mảng ký tự (8 bit) -
 44 /// vd: unsigned int8 kt_ht[10]
 45 /// glcd_text57(x,y, kt_ht, size, color);
 46 /// (chế độ graphic mode, qui trình tham
 47 /// khảo bài 663)

VD: unsigned int8 SP[3]; //CH,DV,null
 SP[0] = T0/10%10 + 0X30; //CHỤC
 SP[1] = T0%10 + 0X30;
 glcd_text57(x,y,SP,size,color);

```

glcd_data(t0/10%10 + 0x30);
glcd_data(t0%10 + 0x30);

lcd_command(lcd_addr_line2);
lcd_data(t0/100%10 + 0x30);
lcd_data(t0/10%10 + 0x30);
lcd_data(t0%10 + 0x30);
}

void giao_ma_gan_cho_8led_quet()
{
    led_7dq[0]= ma7doan [giay %10];
    led_7dq[1]= ma7doan [giay /10];
    led_7dq[3]= ma7doan [phut %10];
    led_7dq[4]= ma7doan [phut /10];
    led_7dq[6]= ma7doan [gio %10];
    led_7dq[7]= ma7doan [gio /10];
}
void tat_2led_chinh()
{
    if      (gia_tri_mod==1) {led_7dq[7]=0xff; led_7dq[6]=0xff;}
    else if (gia_tri_mod==2) {led_7dq[4]=0xff; led_7dq[3]=0xff;}
    else if (gia_tri_mod==3) {led_7dq[1]=0xff; led_7dq[0]=0xff;}
}
//xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
void phim_mod()
{
    if (phim_bt1(50))
    {
        gia_tri_mod++;
        if (gia_tri_mod>=4) gia_tri_mod=0;
        dem_tg_exit=0;
    }
}
//xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
void phim_up()
{
    if (phim_bt0(20))
    {
        dem_tg_exit=0;
        switch (gia_tri_mod)
        {
            case 1:   if (gio==23)     gio=0;
                       else          gio++;
                       break;
            case 2:   if (phut==59)   phut=0;
                       else          phut++;
                       break;
            case 3:   if (giay==59)   giay=0;
                       else          giay++;
                       break;
        }
    }
}

```

```

        default:    break;
    }
}
}

void phim_dw()
{
    if (phim_bt2(20))
    {
        dem_tg_exit=0;
        switch (gia_tri_mod)
        {
            case 1:    if (gio==0)      gio=23;
                        else          gio--;
                        break;
            case 2:    if (phut==0)    phut=59;
                        else          phut--;
                        break;
            case 3:    if (giay==0)    giay=59;
                        else          giay--;
                        break;
            default: break;
        }
    }
}
//xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
void main()
{
    set_up_port_ic_chot();
    setup_timer_1(t1_internal | t1_div_by_8);
    set_timer1(3036);
    enable_interrupts(global);
    enable_interrupts(int_timer1);
    bdn=0;
    giay =0;
    phut=0;
    gio=0;
    gia_tri_mod=0;

    setup_glcd(glcd_text_mode);

    glcd_command(glcd_addr_line1);
    glcd_data("dem san pham:");

    setup_lcd();
    lcd_command(lcd_addr_line1);
    lcd_data("lcd: dem san pham:");

    setup_timer_0 (t0_ext_l_to_h | t0_div_1|t0_8_bit);
    set_timer0(0);
}

```

```

t0_tam = t0 = 0;
xuat_4led_7doan_giaima_xoa_so0(t0);
glcd_hienthi_t0(t0);
while(true)
{
    giao_ma_gan_cho_8led_quet();
    if(bdn<10)
    {
        if ((bdn<5)&&(input(bt0)) && (input(bt2)))
            tat_2led_chinh();
        hien_thi_8led_7doan_quet_all();
        phim_mod();
        phim_up();
        phim_dw();
        t0=get_timer0();
        if (t0!=t0_tam)
        {
            t0_tam = t0;
            glcd_hienthi_t0(t0);
            xuat_4led_7doan_giaima_xoa_so0(t0);
            if (t0>=101) set_timer0(1);
        }
        if (ttct_td==1) sang_tat_32led(2,1);
        if (ttct_td==2) sang_tat_dan_pst_32led(5,1);
        if (ttct_td==3) sang_tat_dan_tsp_32led(5,1);
        if (ttct_td==4) sang_tat_dan_ngoai_vao_32led(5,1);
        if (ttct_td==5) sang_tat_dan_trong_ra_32led(5,1);
        if (ttct_td==6) sang_don_pst_32led(5,1);
        if (ttct_td==7) diem_sang_dich_trai_mat_dan_32led(5,1);
        if (ttct_td==8) sang_don_tsp_32led(5,1);
        if (ttct_td==9) diem_sang_dich_phai_mat_dan_32led(5,1);

        if (ttct_td==10)
            sang_tat_dan_trai_sang_phai_2x16led(5,1);
        if (ttct_td==11)
            sang_tat_dan_phai_sang_trai_2x16led(5,1);
        if (ttct_td==12) diem_sang_di_chuyen_pst_32led(5,1);
        if (ttct_td==13) diem_sang_di_chuyen_tsp_32led(5,1);
        if (ttct_td==14) sang_don_tnt_32led(5,1);
        if (ttct_td==15) sang_don_ttr_32led(5,1);
        if (ttct_td>15) ttct_td=1;
    }
    else
    {
        bdn = bdn-10;
        dem_tg_exit++;
        if (dem_tg_exit==20) gia_tri_mod=0;
        giay++;
        if (giay==60)
        {

```

```

        giay =0;
        phut++;
        if (phut==60)
        {
            phut=0;
            gio++;
            if (gio==24) gio =0;
        }
    }
}

```

d. Tiến hành biên dịch và nạp.

e. Quan sát kết quả: vừa đếm sản phẩm vừa điều khiển 32 led chạy nhiều chương trình.

Bài tập 672. Giống bài 671 nhưng GLCD hiển thị logo spkt và đếm sản phẩm.

Lưu tên file là “BAI_672_TONGHOP_5_MODULE”.

graphic_mode

text_mode

- Xóa màn hình GLCD.

SETUP_GLCD(GLCD_GRAPHIC_MODE); //KHỞI TẠO VÀ XÓA MÀN HÌNH
// LIÊN QUAN ĐẾN GRAPHIC, TEXT VẪN CÒN
SETUP_GLCD(GLCD_TEXT_MODE);

- KHI HIỂN THỊ: TRONG TH NÀY: HIỂN THỊ LOGO TRƯỚC, HT SỐ ĐẾM SP

GLCD_COMMAND(GLCD_GRAPHIC_MODE);
//HIỂN THỊ LOGO - VIẾT NGOÀI WHILE(TRUE)

GLCD_COMMAND(GLCD_TEXT_MODE);
//HIỂN THỊ SỐ ĐẾM SẢN PHẨM - COUNTER - TRONG WHILE(TRUE)

* TRƯỜNG HỢP: HIỂN THỊ SỐ ĐẾM NĂM TRONG VÒNG TRÒN
- CÓ 1 NÚT NHẤN: THAY ĐỔI HÌNH TRÒN - HÌNH VUÔNG

```

VOID KT_CD()
{
    ...
    //XỬ LÝ CHỨC NĂNG
    TT_HT = ~TT_HT; // INT1 TT_HT;
    IF(TT_HT == 0)
    { //HÌNH TRÒN
        setup_glcd(glcd_graphic_mode);
        glcd_mau_nen(0);
        glcd_circle(40, 40, 20, 0, 1);
        gdram_vdk_to_gdram_glcd_all();
    }
    ELSE
    { //HÌNH VUÔNG
        setup_glcd(glcd_graphic_mode);
        glcd_mau_nen(0);
        glcd_RECT(...);
        gdram_vdk_to_gdram_glcd_all();
    }
    GLCD_COMMAND(GLCD_TEXT_MODE);
    ...
}

```

```

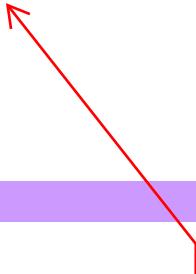
VOID MAIN()
{
    ...
    //HÌNH TRÒN
    setup_glcd(glcd_graphic_mode);
    glcd_mau_nen(0);
    //glcd_circle(40, 40, 20, 0, 1);
    //glcd_rect
    glcd_line
    glcd_line
    gdram_vdk_to_gdram_glcd_all();

    GLCD_COMMAND(GLCD_TEXT_MODE);

    WHILE(TRUE)
}

```


Chương 7: THỰC HÀNH CHUYỂN ĐỔI TƯƠNG TỤ SANG SỐ - ADC, CÁC CẢM BIẾN

- 
- 1. PHẦN CỨNG**
SƠ ĐỒ NGUYÊN LÝ - 191
 - 2. PHẦN MỀM**
CẤU TRÚC ADC - 188
CÁC LỆNH KHỞI TẠO, ĐIỀU KHIỂN - 189
 - 3. BT MẪU**

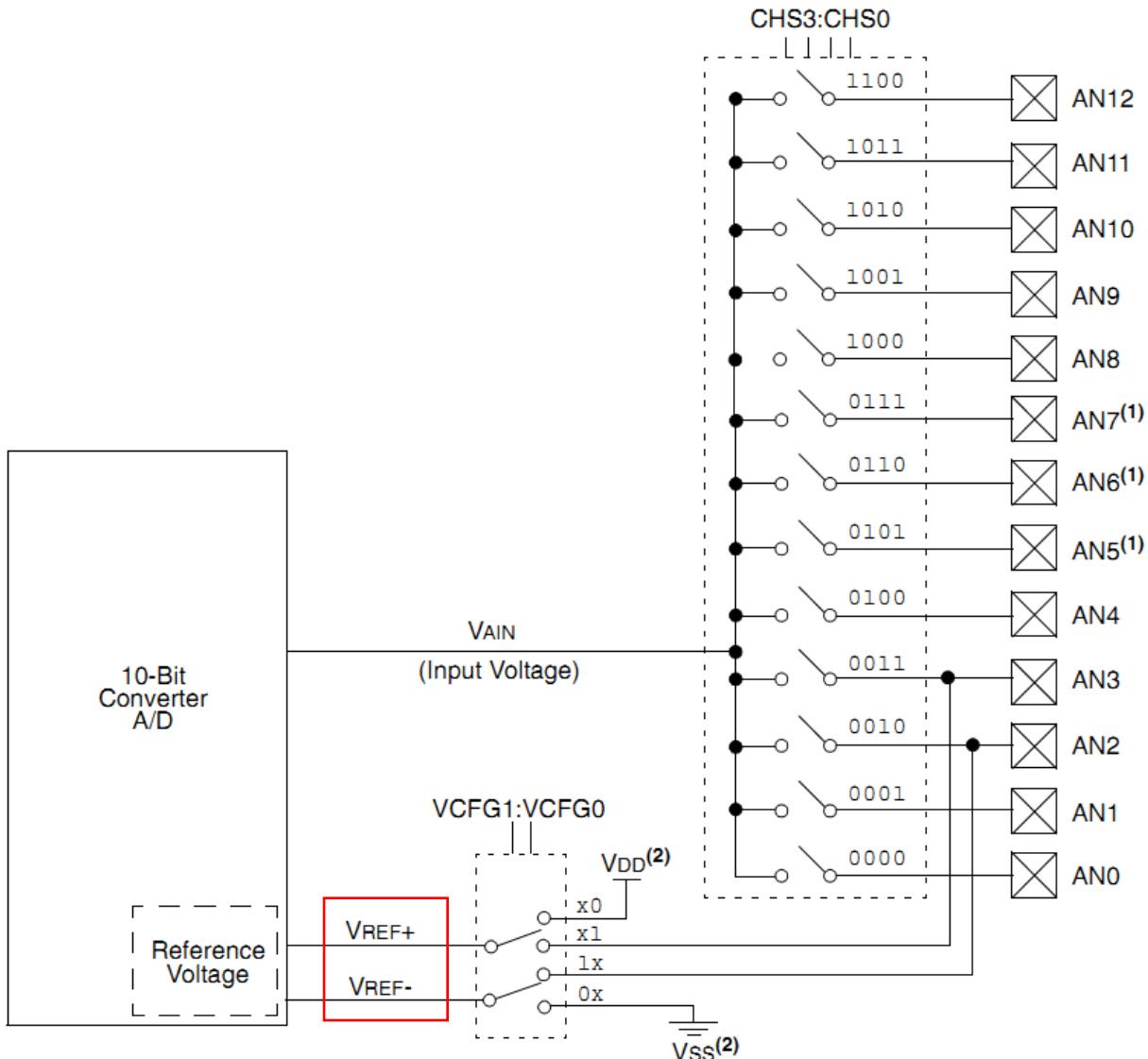
7.1 CHUYỂN ĐỔI TƯƠNG TỰ SANG SỐ CỦA PIC

Vi điều khiển PIC có bộ chuyển đổi tín hiệu tương tự sang tín hiệu số ADC 10 bit đa hợp nhiều kênh tùy thuộc vào từng loại vi điều khiển cụ thể, PIC16F877A có 8 kênh, PIC16F887 có 14 kênh và PIC18F4550 có 13 kênh. Mạch ADC dùng cho các ứng dụng giao tiếp với tín hiệu tương tự có thể nhận từ các cảm biến như cảm biến nhiệt độ LM35, cảm biến áp suất, cảm biến độ ẩm, cảm biến khoảng cách, ...

Phần này sẽ khảo sát chi tiết khói ADC của PIC, các thanh ghi của khói ADC, trình tự thực hiện chuyển đổi, tập lệnh lập trình C cho ADC và ứng dụng ADC để đo nhiệt độ.

7.1.1 Khảo sát ADC của PIC 18F4550

ADC của IC18F4550 có sơ đồ khói như hình 7-1:



Hình 7-1. Sơ đồ khói của ADC PIC18F4550.

Chức năng các thành phần:

- AN0 đến AN12 (analog) là 13 ngõ vào của 13 kênh tương tự được đưa đến mạch đa hợp.
- CHS<3:0> là các ngõ vào chọn kênh của bộ đa hợp tương tự.
- Tín hiệu kênh tương tự đã chọn sẽ được đưa đến bộ chuyển đổi ADC.
- Điện áp tham chiếu dương Vref+ có thể lập trình nối với nguồn cung cấp dương V_{DD} hoặc điện áp tham chiếu bên ngoài nối với ngõ vào Vref+ của chân AN3. Điện áp tham chiếu âm Vref- có thể lập trình nối với nguồn cung cấp V_{SS} hoặc điện áp tham chiếu bên ngoài nối với ngõ vào Vref- của chân AN22 bit lựa chọn là VCFG1:VCFG0.
- **Hai ngõ vào Vref+ và Vref- có chức năng thiết lập độ phân giải cho ADC.**
- Bit ADON có chức năng cho phép ADC hoạt động hoặc tắt bộ ADC khi không hoạt động để giảm công suất tiêu tán, ADON bằng 1 thì cho phép, bằng 0 thì tắt.
- Kết quả chuyển đổi là số nhị phân 10 bit sẽ lưu vào cùp thanh ghi 16 bit có tên là ADRESH và ADRESL, 10 bit kết quả lưu vào thanh ghi 16 bit nên có dạng lưu là canh lè trái và canh lè phải tùy thuộc vào bit lựa chọn có tên ADFM.

← Tr. 193

ADC có 13 kênh nhưng mỗi thời điểm chỉ chuyển đổi 1 kênh và chuyển đổi kênh nào thì phụ thuộc vào 4 bit chọn kênh CHS4:CHS0. Hai ngõ vào điện áp tham chiếu dương và âm có thể lập trình nối với nguồn V_{DD} và V_{SS} hoặc nhận điện áp tham chiếu từ bên ngoài qua 2 chân RA3 và RA2.

Khối ADC độc lập với CPU nên có thể hoạt động khi CPU đang ở chế độ ngủ do xung cung cấp cho ADC lấy từ dao động RC bên trong của khối ADC.

7.1.2 Tập lệnh c cho khối ADC của PIC

Các lệnh của ngôn ngữ lập trình C liên quan đến ADC bao gồm:

Lệnh **SETUP_ADC(MODE)**

Lệnh **SETUP_ADC_PORT(VALUE)**

Lệnh **SET_ADC_CHANNEL(CHAN)**

Lệnh **VALUE=READ_ADC(MODE)**

a. LỆNH SETUP_ADC(MODE) - LỆNH ĐỊNH CÁU HÌNH CHO ADC

Cú pháp: **setup_adc (mode);**

Thông số: **mode**- Analog to digital mode. Xem trong file device.

Chức năng: định cấu hình cho ADC

b. LỆNH SETUP_ADC_PORT(VALUE) - LỆNH ĐỊNH CÁU HÌNH CHO PORT

Cú pháp: **setup_adc_ports (value)**

Hàng số: **value** - là hàng số định nghĩa trong file **devices .h**

18F4550.H
Keyword:
ADC

Chức năng: Thiết lập chân của ADC là tương tự, số hoặc tổ hợp cả 2.

c. LỆNH SET_ADC_CHANNEL(chan) - LỆNH CHỌN KÊNH ADC

Cú pháp: Set_adc_channel (*chan*) - chọn kênh cần chuyển đổi khi đo nhiều hơn 1 kênh, nếu chỉ đo 1 kênh thì không cần.

Thông số: *Chan* là thứ tự kênh cần chuyển đổi. Kênh bắt đầu từ số 0.

Chức năng: Chọn kênh cần chuyển đổi ADC

năng: Phải chờ một khoảng thời gian ngắn khi chuyển kênh, thường thì thời gian chờ khoảng 10μs.

d. LỆNH value=READ_ADC(*mode*) - LỆNH ĐỌC KẾT QUẢ CHUYỂN ĐỔI ADC

Cú pháp: value = read_adc ([*mode*]) - đọc kết quả sau khi chuyển đổi xong

Hằng số: *mode* là 1 trong các hằng số sau:

- ADC_START_AND_READ (thực hiện đọc liên tục, mặc nhiên là mode này)
- ADC_START_ONLY (bắt đầu chuyển đổi)
- ADC_READ_ONLY (đọc kết quả của lần chuyển đổi sau cùng)

Trả về: Kết quả 8 bit hay 16 bit tùy thuộc và khai báo #DEVICE ADC= directive.

Chức năng: Lệnh này sẽ đọc giá trị số sau khi chuyển đổi xong

Khai báo #DEVICE ADC= directive như sau:

#DEVICE	8 bit	10 bit	11 bit	16 bit
---------	-------	--------	--------	--------

7.2 ĐO NHIỆT ĐỘ DÙNG CẢM BIẾN LM35

7.2.1 Cảm biến LM35

Cảm biến LM35 hoạt động với điện áp từ +35V đến 4V.

Điện áp ra thay đổi 10mV cho 1 độ C.

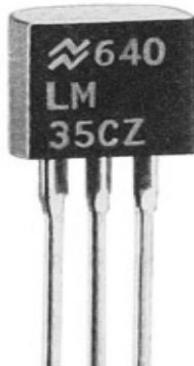
Điện áp ngõ ra từ +6V đến – 1V.

Dòng điện ngõ ra 10mA.

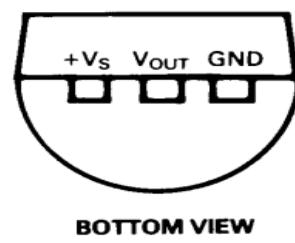
LM35CZ có tầm đo từ -40°C đến $+110^{\circ}\text{C}$, LM35DZ có tầm đo từ 0°C đến $+100^{\circ}\text{C}$.

Hình dạng bên ngoài và tên các tín hiệu

- Hình dạng bên ngoài và tên các tín hiệu như hình



**TO-92
Plastic package**



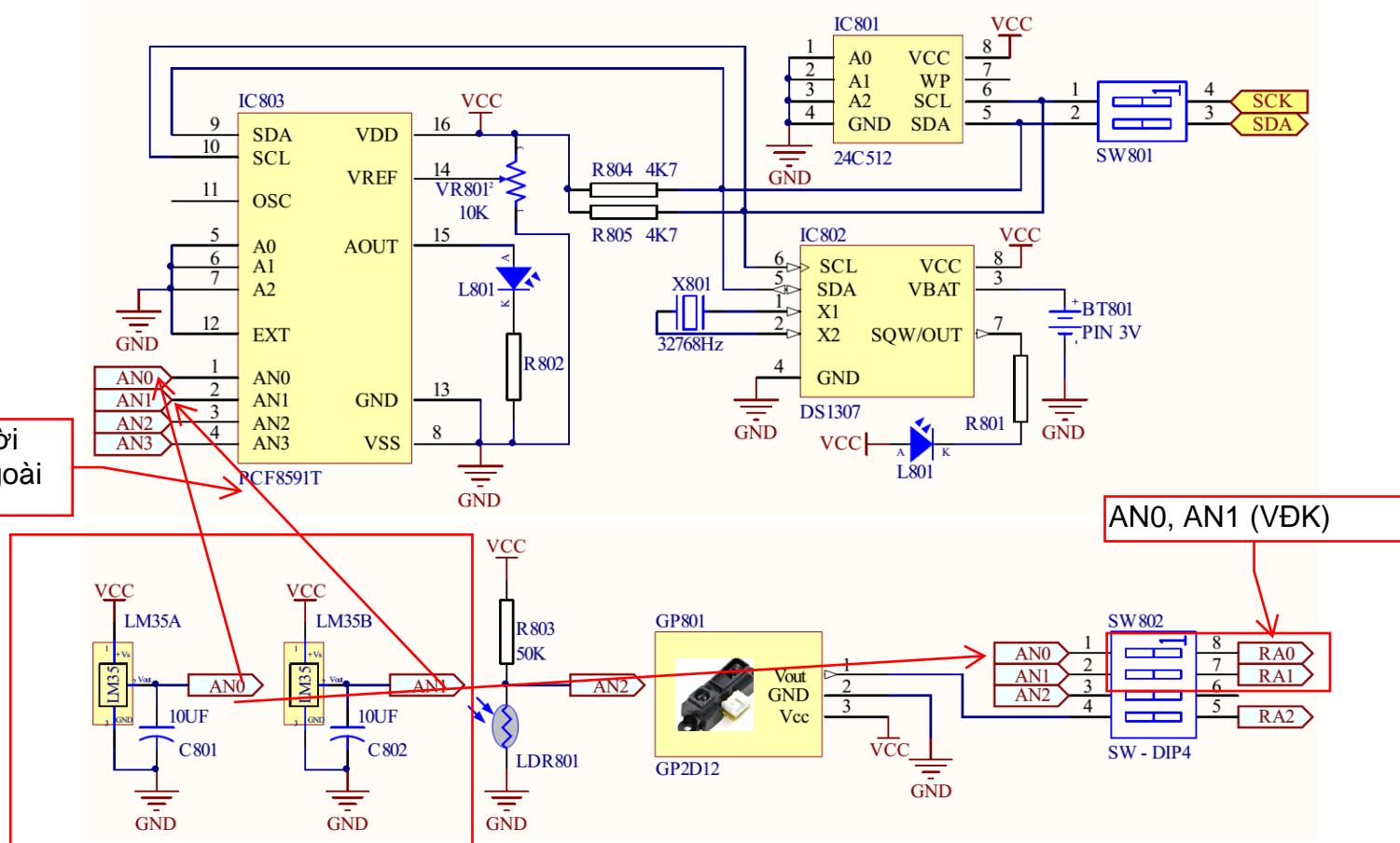
BOTTOM VIEW

Hình 7-2. Hình ảnh cảm biến LM35CZ.

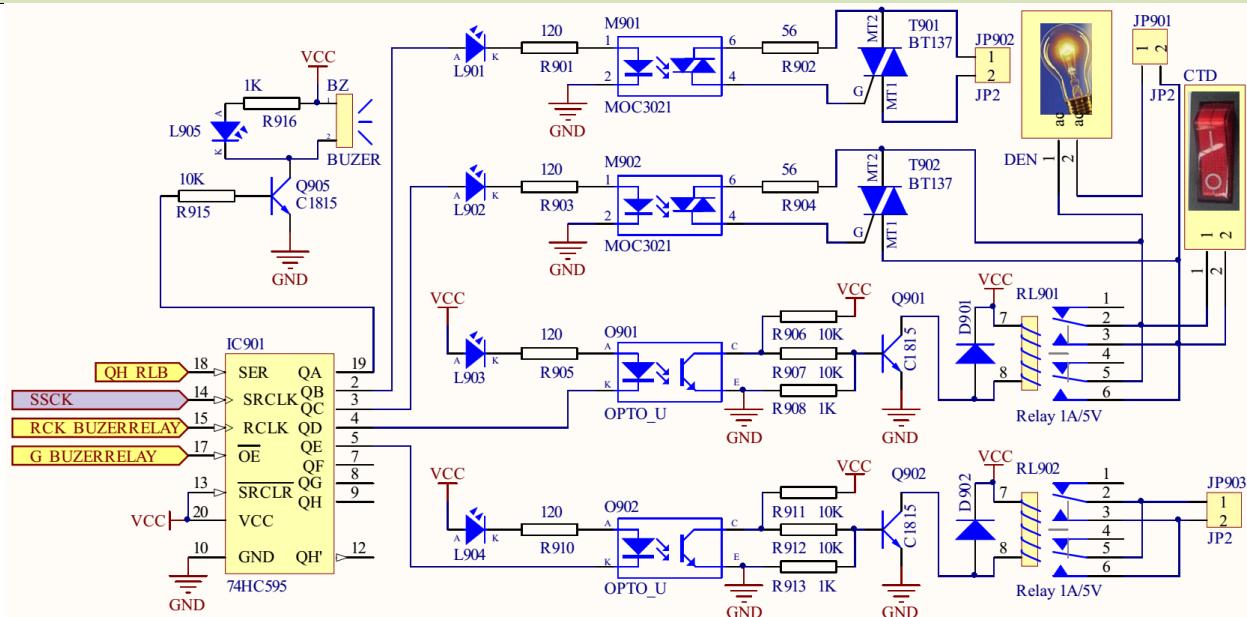
7.2.2 Mạch điện giao tiếp vi điều khiển PIC với cảm biến LM35

Phần này thực hiện các bài chuyển đổi ADC với các tín hiệu vào là các cảm biến.

Mạch điện giao tiếp vi điều khiển với 2 cảm biến LM35A, LM35B, 1 cảm biến quang và 1 cảm biến khoảng cách có sơ đồ mạch như hình 7-3.



Hình 7-3. Sơ đồ nguyên lý module ADC và các cảm biến tương tự.



Hình 7-4. Sơ đồ nguyên lý module điều khiển Relay, Triac, tải 220V AC.

Các cảm biến LM35A và LM35B nối với 2 ngõ vào AN0 và AN1 của vi mạch ADC bên ngoài là PCF8591, đồng thời nối với 2 ngõ vào tương tự RA0/AN0, RA1/AN1 thông qua Switch SW802.

Cảm biến quang nối với ngõ vào AN2 của vi mạch ADC bên ngoài là PCF8591.

Cảm biến khoảng cách nối với ngõ vào tương tự AN2/RA2 thông qua Switch SW802.

Các cảm biến LM35A và LM35B chuyển đổi nhiệt độ thành tín hiệu điện. Trong bộ thí nghiệm thì các cảm biến nhiệt được đặt bên trong đế gần bóng đèn tròn để có thể điều khiển thay đổi nhiệt độ.

Khi đèn sáng thì làm gia tăng nhiệt độ giúp bạn quan sát kết quả đo, khi đèn tắt thì nhiệt độ giảm dần cho đến khi bằng nhiệt độ môi trường.

Để mở đèn thì có 3 contact mắc song song như hình 7-4.

- Điều khiển đèn bằng tay: ta dùng contact thứ 2 nằm trên contact nguồn của bộ thí nghiệm.
- Điều khiển tự động: có 2 contact còn lại là dùng Triac hoặc dùng relay. Ở chế độ tự động thì ta phải chuyển contact bằng tay về chế độ OFF, nếu bạn để ON thì chế độ tự động tắt mở đèn trở nên vô nghĩa.

7.2.3 Đo nhiệt độ dùng cảm biến LM35 hiển thị trên 4 led 7 đoạn

Phần này thực hành ADC và cảm biến nhiệt LM35.

ADC được dùng để chuyển đổi điện áp tương tự từ cảm biến LM35 nên ta phải biết hệ số chuyển đổi của cảm biến.

Cảm biến LM35 có nhiều loại: LM35C và LM35D: loại LM35C thì tầm đo từ -40°C đến 110 °C và LM35D thì tầm đo từ 0°C đến 100 °C. Dòng tiêu thụ của cảm biến với nguồn cung cấp 5VDC là 50 μ A.

Hệ số chuyển đổi tuyến tính khi nhiệt độ thay đổi 1°C thì điện áp ra thay đổi 10mV.

- Tính toán độ phân giải của ADC:

Công thức tính độ phân giải của ADC (step size) là $SS_{ADC} = \frac{V_{REF+} - V_{REF-}}{2^{10} - 1} = \frac{V_{REF+} - V_{REF-}}{1023}$

Thông số 2^{10} là do ADC 10 bit.

Độ phân giải của ADC phải bằng độ phân giải của cảm biến và bằng 10mV:

$$SS_{LM35} = 10mV.$$

Nên ta có: $SS_{ADC} = \frac{V_{REF+} - V_{REF-}}{1023} = SS_{LM35} = 10mV$

Bây giờ ta tính điện áp tham chiếu cho ADC:

$$V_{REF+} - V_{REF-} = SS_{LM35} \times 1023 = 10mV \times 1023 = 10230mV$$

Một phương trình có 2 tham số nên ta phải chọn 1 tham số và tính tham số còn lại:

Chọn điện áp tham chiếu là $V_{REF-} = 0V$

điện áp tham chiếu --> xđ gt vào ra bên trong VĐK (5V, 0V)

Khi đó $V_{REF+} = 10,023V \cong 10V$

Điện áp này lớn hơn lớn hơn điện áp cho phép của ADC là 5V nên không phù hợp, trong khi đó ta có nguồn điện áp $V_{DD} = 5V$ nên ta chọn $V_{REF+} = V_{DD} = 5V$

Khi đó độ phân giải ADC sẽ là: $SS = \frac{V_{REF+} - V_{REF-}}{2^{10} - 1} = \frac{5000mV}{1023} = 4,887mV \cong 5mV$

Độ phân giải ADC là 5mV không tương thích với độ phân giải của cảm biến LM35 bằng 10mV, tỉ lệ chênh lệch là:

$5mV \rightarrow 1$ đơn vị
 $10mV \rightarrow 2$ đơn vị
 $(1$ độ C $)$
 $\rightarrow n.\text{độ} = \text{gt đo}/2;$

$$\frac{10mV}{5mV} = 2$$

giá trị đ. áp ngõ vào để ngõ ra t.đổi 1 đơn vị
 (độ phân giải - step size)

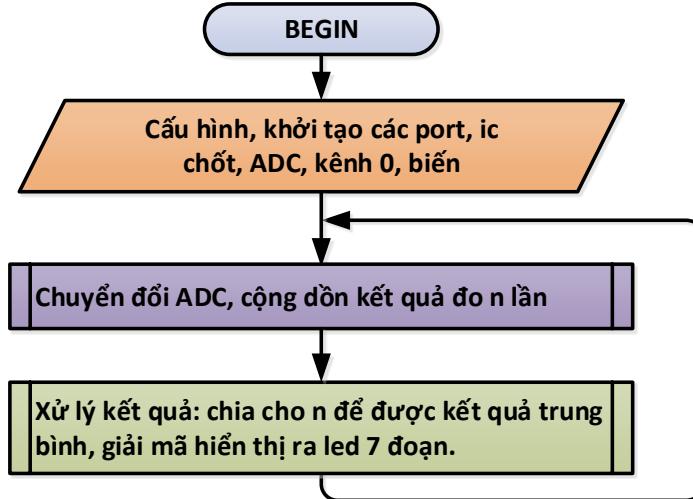
Hiển thị đúng thì lấy kết quả chuyển đổi chia cho tỉ lệ chênh lệch.

Nhưng xác hơn theo 4,887mV thì hệ số là 2,046.

Bài mẫu 701. Chương trình đo nhiệt độ hiển thị trên 4 led 7 đoạn dùng ADC kênh thứ 0. Đo trung bình 100 lần.

Lưu tên file là “BAI_701_LM35A”.

- Mục đích: biết sử dụng cảm biến nhiệt LM35, lập trình ADC để đo và hiển thị nhiệt độ trên module 4 led 7 đoạn, biết cách đo trung bình.
- Lưu đồ:

**Hình 7-5. Lưu đồ chuyển đổi ADC kênh thứ 0 để đọc nhiệt độ từ cảm biến LM35.**

c. Chương trình:

```

#include <tv_pickit2_shift_1.c>
unsigned int8 j, solan=100;
unsigned int16 lm35a;
void main()
{
    set_up_port_ic_chot();
    setup_adc(adc_clock_div_32);
    setup_adc_ports(an0|vss_vdd );
    set_adc_channel(0);
    while(true)
    {
        lm35a = 0;
        for (j=0; j<solan; j++)
        {
            lm35a = lm35a + read_adc();
            delay_ms(1);
        }
        lm35a = lm35a /2.046;
        lm35a = lm35a /solan;
        xuat_4led_7doan_giaima_xoa_so0(lm35a);
    }
}

```

khởi tạo ADC
- khai báo tần số lấy mẫu

sử dụng ngõ vào AN0
và VREF+ = VDD,
VREF- = VSS

chọn kênh thứ 0 để
đo

đọc ADC và cộng dồn

chia tỷ lệ chênh lệch
trang 193

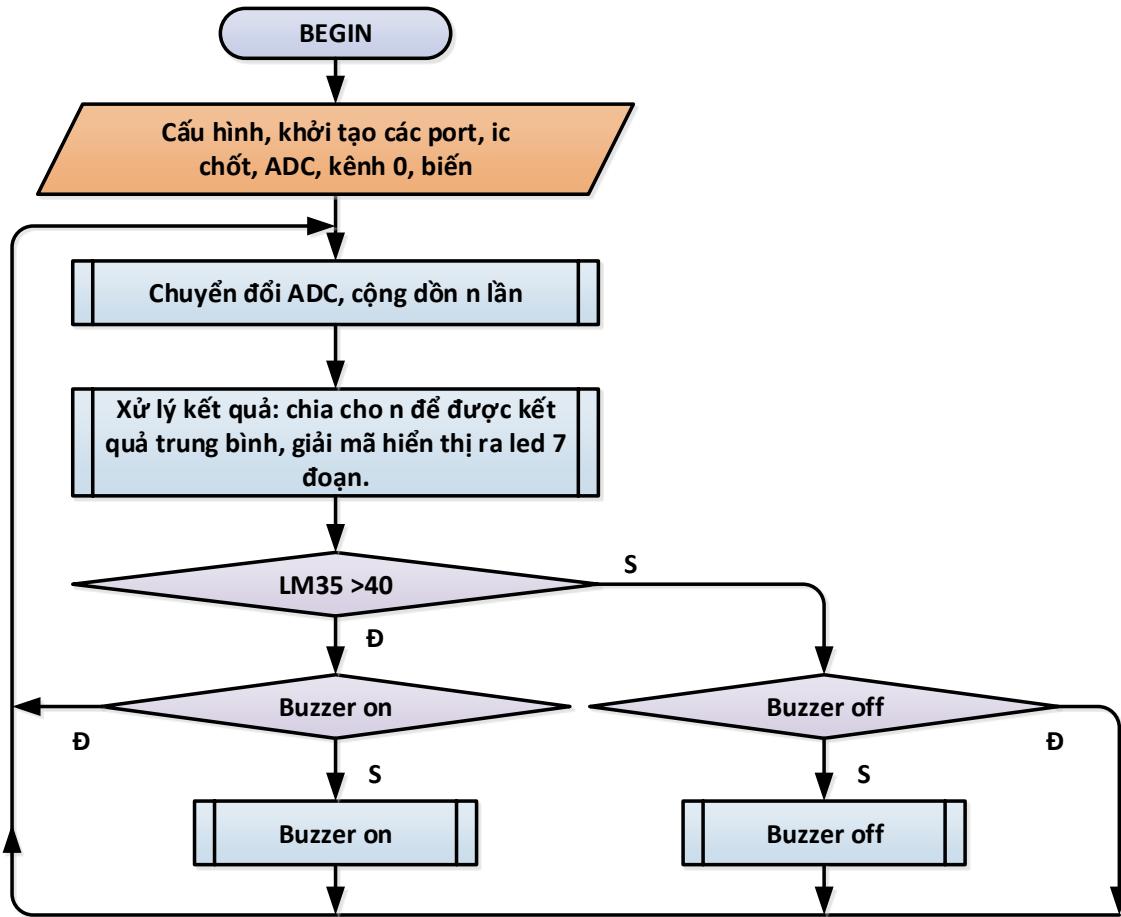
chia lấy trung bình

- d. Tiến hành biên dịch và nạp.
- e. Quan sát kết quả: Mở contact cho đèn sáng để gia tăng nhiệt độ cho thấy sự thay đổi của nhiệt độ, sau đó tắt đèn để xem nhiệt độ giảm.
- f. Giải thích chương trình: kết quả sau khi đo chia cho hệ số chênh lệch độ phân giải, chia cho 100 để được kết quả đúng.
- g. Hãy trả lời câu hỏi: tại sao đo trung bình 100 lần?

Bài mẫu 702. Chương trình đo nhiệt độ hiển thị trên 4 led 7 đoạn dùng ADC kênh thứ 0. Đo trung bình 100 lần, khi nhiệt độ lớn hơn 40 độ thì báo động bằng **buzzer**, khi nhỏ hơn thì tắt báo động.

Lưu tên file là “BAI_702_LM35A_BUZZ”.

- Mục đích: biết so sánh nhiệt độ để báo động, biết tắt mở buzzer.
- Lưu đồ:



Hình 7-6. Lưu đồ chuyển đổi ADC kênh thứ 0 để đọc nhiệt độ và so sánh điều khiển buzzer.

- Chương trình:

```

#include <tv_pickit2_shift_1.c>
#define nd_tren 40 // Line 35
unsigned int8 j, solan=100;
unsigned int16 lm35a;
int1 ttqn=0;
void so_sanh_dk_buzzer()
{
    if ((lm35a>nd_tren)&&(ttqn==0)) // buzzer_on();
    {
        ttqn = 1;
        buzzer_on(); // Line 35
    }
    else if ((lm35a<nd_tren)&&(ttqn==1))
    {
        ttqn = 0;
        buzzer_off(); // Line 35
    }
}
  
```

```

}
void main()
{
    set_up_port_ic_chot();
    setup_adc(adc_clock_div_32);
    setup_adc_ports(an0|vss_vdd );
    set_adc_channel(0);
    ttqn = 0;
    while(true)
    {
        HIEN....QUET();
        delay_lâu hơn -->
        quét led nhiều lần
        for
        {
            lm35a = 0;
            for (j=0; j<solan; j++)
            {
                lm35a = lm35a + read_adc();
                delay_ms(1);
            }
            lm35a = lm35a /2.046;
            lm35a = lm35a /solan;
            xuat_4led_7doan_giaima_xoa_so0(lm35a);
            so_sanh_dk_buzzer();
        }
    }
}

```

đọc kết quả đo nhiều lần --> lấy trung bình

$$(35+36)/2 = 35.5$$

(có phần thập phân)

- khai báo biến dạng số thập phân:

FLOAT LM35A;
INT16 KQ_HT;
KQ_HT = (int16) (LM35A*10); //35.53
355

(BỎ PHẦN THẬP PHÂN CÒN LẠI
- CHỈ LẤY PHẦN NGUYÊN)

- GIẢI MÃ:

TP = MA7DOAN[KQ_HT%10];
DV = MA7DOAN[KQ_HT/10%10] & 0X7F;
CH = MA7DOAN[KQ_HT/100];

* LCD + GLCD:

LCD_COMMAND(0XC0); // HÀNG 2
printf(LCD_DATA, "ND = %3.2f", LM35A);

...
GLCD_COMMAND(0X80);
printf(GLCD_DATA, "ND = %3.2f", LM35A);

- d. Tiến hành biên dịch và nạp.
- e. Quan sát kết quả: Mở contact cho đèn sáng để
- f. Giải thích chương trình: lệnh so sánh nếu nhiệt độ mới mẻ buzz và cho TTQN bằng 1, nếu nhiệt độ giảm dưới 40 và nếu TTQN ở mức 1 thì tắt buzz và cho TTQN bằng 0, sau đó nếu nhiệt độ nhỏ hơn 40 nhưng TTQN bằng 0 thì không cần tắt buzz. Điều này làm giảm bớt quá trình đóng ngắt buzz không cần thiết.
- g. Hãy giải thích hàm làm buzz on và off. Hãy vẽ mạch điều khiển buzz và giải thích mạch.
- h. Hãy giải thích chức năng của biến TTQN.

Bài mẫu 703. Chương trình đo nhiệt độ hiển thị trên 4 led 7 đoạn dùng ADC kênh thứ 0, đo trung bình 100 lần.

Tự động điều khiển mở bóng đèn sáng để tăng nhiệt cho đèn khi lớn hơn 40 độ thì tắt đèn và báo động, khi nhiệt độ giảm dưới 40 độ thì tắt báo động, nhiệt độ nhỏ hơn 35 độ thì mở lại bóng đèn.

Chú ý: chương trình tự động mở đèn và tắt đèn nên bạn không được đóng contact mở đèn.

Lưu tên file là “BAI_703_LM35A_BUZZ_AUTO”.

- a. Mục đích: biết điều khiển đèn tự động tắt mở đèn để giảm hoặc gia tăng nhiệt độ.
- b. Lưu đồ: sinh viên hãy tự viết.
- c. Chương trình:

```
#include <tv_pickit2_shift_1.c>
```

```

#define nd_tren 40
#define nd_duoi 40
unsigned int8 j, solan=100;
unsigned int16 lm35a;
int1 ttqn=0;
void so_sanh_dk_buzzer()
{
    if ((lm35a>nd_tren)&&(ttqn==0))
    {
        ttqn = 1;
        buzzer_on();
        triac_2_off();
    }
    else if ((lm35a<nd_tren)&&(ttqn==1))
    {
        ttqn = 0;
        buzzer_off();
    }
    else if ((lm35a<nd_duoi)&&(ttqn==0)) triac_2_on();
}
void main()
{
    set_up_port_ic_chot();
    setup_adc(adc_clock_div_32);
    setup_adc_ports(an0|vss_vdd );
    set_adc_channel(0);
    ttqn = 0;
    while(true)
    {
        lm35a = 0;
        for (j=0; j<solan; j++)
        {
            lm35a = lm35a + read_adc();
            delay_ms(1);
        }
        lm35a = lm35a /2.046;
        lm35a = lm35a /solan;
        xuat_4led_7doan_giaima_xoa_so0(lm35a);
        so_sanh_dk_buzzer();
    }
}

```

38 > 40: sai
38 < 40: đúng --> t.hiện if
bỏ else if
bỏ ttqn --> bỏ else cuối cùng

- d. Tiến hành biên dịch và nạp.
- e. Quan sát kết quả: chuyên contact đèn sang vị trí off, đèn được mở tắt tự động.
- f. Giải thích chương trình:

Bài tập 704. Làm lại bài 703 vừa hiển thị trên 2 led 7 đoạn vừa hiển thị trên LCD font chữ to.

Lưu tên file là “BAI_704_LM35A_BUZZ_AUTO”.

hàng cuối: nhiệt độ
cuối hàng, kết hợp
các bài ở chương 6

Bài mẫu 705. Chương trình **đo nhiệt độ 2 kênh** hiển thị trên 4 led 7 đoạn dùng ADC kênh thứ 0 và thứ 1. Đo trung bình 100 lần, mỗi kênh hiển thị trên 2 led.

Lưu tên file là “BAI_705_ADC_2LM35”.

- a. Mục đích: biết đo nhiệt độ nhiều kênh.
- b. Lưu đồ: sinh viên hãy tự viết.
- c. Chương trình:

```
#include <tv_pickit2_shift_1.c>
unsigned int8      j, solan=100;
unsigned int16     lm35a, lm35b;

void doc_nd_lm35a()
{
    set_adc_channel(0);
    lm35a = 0;
    for (j=0; j<solan; j++)
    {
        lm35a = lm35a + read_adc();
        delay_us(100);
    }
    lm35a = lm35a /2.046;
    lm35a = lm35a /solan;
}

void doc_nd_lm35b()
{
    set_adc_channel(1);
    lm35b = 0;
    for (j=0; j<solan; j++)
    {
        lm35b = lm35b + read_adc();
        delay_us(100);
    }
    lm35b = lm35b /2.046;
    lm35b = lm35b /solan;
}

void main()
{
    set_up_port_ic_chot();
    setup_adcadc_clock_div_32);
    setup_adc_ports(an0_to_an1|vss_vdd );

    while(true)
    {
        doc_nd_lm35a();
        doc_nd_lm35b();
        xuat_4led_7doan_2so_4led(lm35b, lm35a);
    }
}
```

- d. Tiến hành biên dịch và nạp.
- e. Quan sát kết quả: nếu kết quả không đúng yêu cầu thì kiểm tra lại chương trình.

f. Giải thích chương trình:

Bài tập 706. Thêm vào yêu cầu bài 705 phần tự động điều khiển mở bóng đèn sáng để tăng nhiệt cho đến khi nhiệt độ của 1 trong 2 kênh lớn hơn 40 độ thì tắt đèn và báo động, khi nhiệt độ cả 2 kênh giảm dưới 40 độ thì tắt báo động, nhiệt độ cả 2 kênh nhỏ hơn 35 độ thì mở lại bóng đèn.

Lưu tên file là “BAI_706_2LM35_BUZZ”.

Bài tập 707. Làm lại bài 706 nhưng hiển thị thêm trên **LCD font chữ to**. Hàng 1 và 2 hiển thị kênh 1, hàng 3 và 4 hiển thị kênh 2. Có chú thích thông tin: ‘LM35A : XX’ và ‘LM35B : XX’. XX là nhiệt độ đo **font chữ lớn**, còn lại là kích thước nhỏ.

Lưu tên file là “BAI_707_2LM35_LCD”.

Bài tập 708. Hệ thống vừa đo nhiệt độ vừa đếm sản phẩm trên led 7 đoạn với yêu cầu như sau:

Đo nhiệt độ kênh AN0 hiển thị trên 2 led 7 đoạn bên phải.

Đếm sản phẩm dùng counter T0 hiển thị trên 2 led 7 đoạn bên trái, giới hạn đếm từ 00 đến 99.

Lưu tên file là “BAI_708_LM35A_COUNTER_T0”.

Bài mẫu 709. Chương trình đo nhiệt độ 1 kênh hiển thị trên 2 led 7 đoạn dùng ADC kênh thứ 0 đo trung bình 100 lần.

2 led 7 đoạn còn lại hiển thị giá trị cài đặt trước (GTCDT) để báo động, giá trị mặc nhiên thấp nhất là 35 độ, lớn nhất là 60 độ, có thể điều chỉnh bằng 2 nút nhấn UP và DW.

Khi giá trị đo lớn hơn GTCDT thì tắt đèn và báo động, khi nhiệt độ giảm nhỏ hơn GTCDT thì tắt báo động và khi nhỏ hơn GTCDT-5 thì mở lại đèn gia nhiệt.

Lưu tên file là “BAI_709_LM35_UP_DW”.

- Mục đích: biết lập trình đo nhiệt độ và cài đặt giới hạn đo.
- Lưu đồ: sinh viên hãy tự viết.
- Chương trình:

```
#include <tv_pickit2_shift_1.c>
int1 ttqn;
unsigned int8 j, solan=100;
unsigned int16 gtcdt, lm35a;

void doc_nd_lm35a()
{
    set_adc_channel(0);
}
```

```

lm35a = 0;
for (j=0; j<solan; j++)
{
    lm35a = lm35a + read_adc();
    delay_us(100);
}
lm35a = lm35a /2.046;
lm35a = lm35a /solan;
}

void so_sanh_dk_buzzer()
{
    if ((lm35a>gtcdt)&&(ttqn==0))
    {
        ttqn = 1;
        buzzer_on();
        triac_2_off();
    }
    else if ((lm35a<gtcdt)&&(ttqn==1))
    {
        ttqn = 0;
        buzzer_off();
    }
    else if ((lm35a<gtcdt-3)&&(ttqn==0)) triac_2_on();
}

void phim_up()
{
    if (!input(up))&&( gtcdt<60)
    {
        delay_ms(20);
        {
            if (!input(up))
            {
                gtcdt++;
                xuat_4led_7doan_2so_4led(gtcdt, lm35a);
                while(!input(up));
            }
        }
    }
}

void phim_dw()
{
    if (!input(dw))&&( gtcdt>35)
    {
        delay_ms(20);
        {
            if (!input(dw))
            {
                gtcdt--;
                xuat_4led_7doan_2so_4led(gtcdt, lm35a);
                while(!input(dw));
            }
        }
    }
}

```

```

        }
    }

void main()
{
    set_up_port_ic_chot();
    setup_adc(adc_clock_div_32);
    setup_adc_ports(an0|vss_vdd );
    set_adc_channel(0);
    gtcdt=35;
    uat_4led_7doan_2so_4led(gtcdt, lm35a);
    while(true)
    {
        lm35a=0;
        for (j=0; j<100; j++)
        {
            lm35a=lm35a+read_adc();
            delay_us(200);
        }
        lm35a = lm35a /2.046;
        lm35a = lm35a /100;
        xuat_4led_7doan_2so_4led(gtcdt, lm35a);
        phim_up();
        phim_dw();
        so_sanh_dk_buzzer();
    }
}

```

- d. Tiến hành biên dịch và nạp.
e. Quan sát kết quả: nếu kết quả không đúng yêu cầu thì kiểm tra lại chương trình.
f. Giải thích chương trình:

Bài tập 710. Hệ thống đo nhiệt độ 2 kênh dùng LM35A và LM35B hiển thị trên LCD.

Hàng 1: “NHIET DO LM35A: XX”

Hàng 2 : “NHIET DO LM35B: XX”

Hàng 3 : “DEM SAN PHAM: XX”

thêm hiển thị lcd, led quét
thêm điều khiển led đơn
LCD h.thị số lớn

Có tự động điều khiển đèn tắt khi nhiệt độ 2 kênh lớn hơn 40 và mở đèn khi cả hai nhỏ hơn 35.

Lưu tên file là “BAI_710_LM35_COUNTER_LCD”.

7.3 ĐO KHOẢNG CÁCH DÙNG CẢM BIẾN GP2D12

7.3.1 Cảm biến GP2D12

Phần này thực hành ADC và cảm biến khoảng cách GP2D12 của Sharp.

Tính chất của cảm biến GP2D12

- Tín hiệu ra là điện áp tương tự.
- Chu kỳ mỗi xung cho Led là 32ms.

- Thời gian đáp ứng trung bình là 39ms.
- Thời gian khởi động là 44ms
- Dòng tiêu thụ là 33mA.
- Phát hiện khoảng cách khu vực trong phạm vi 70cm là 6cm.

Hình dạng bên ngoài và tên các tín hiệu

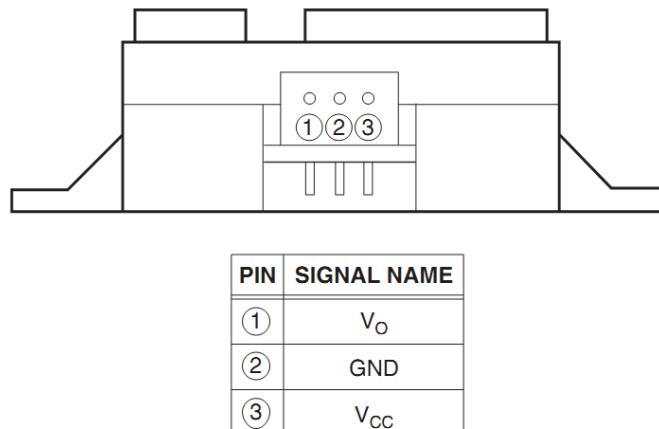
- Hình dạng bên ngoài và tên các tín hiệu như hình 7-7.

Điện áp hoạt động của cảm biến GP2D12:

- Điện áp nguồn cung cấp Vcc từ 4,5V đến 5,5V
- Điện áp tín hiệu ra Vo từ -0,3V đến (Vcc + 0,3V) phụ thuộc vào khoảng cách.
- Nhiệt độ làm việc từ -10 đến 60°C

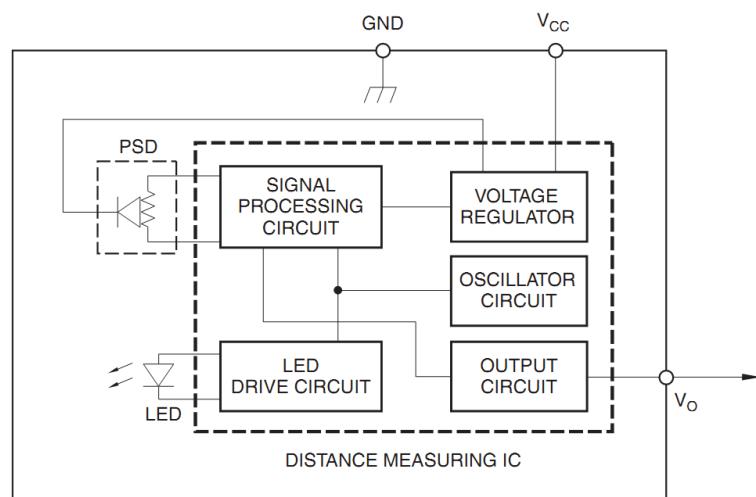
Dạng sóng hoạt động của cảm biến GP2D12:

- Hình 7-9 trình bày dạng sóng của các lần đo của cảm biến:

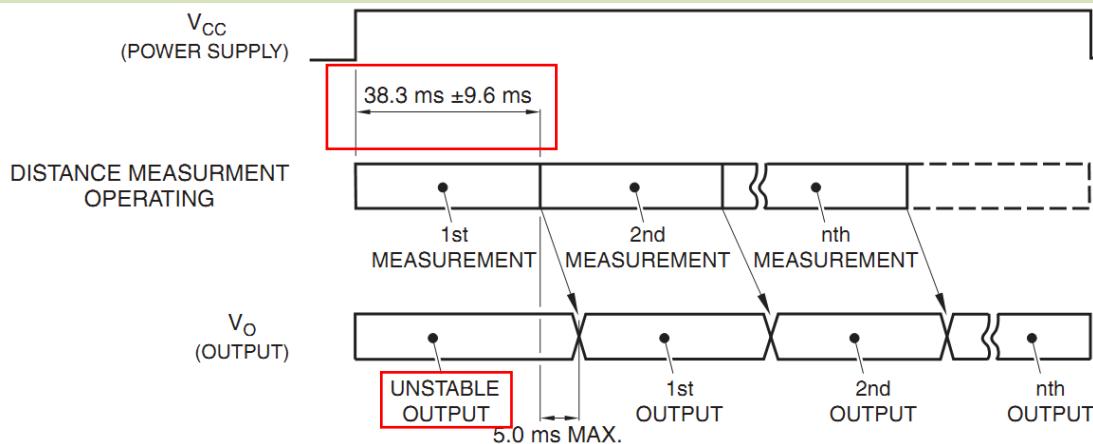


Hình 7-7. Hình cảm biến khoảng cách GP2D12.

Sơ đồ khối bên trong như hình 7-8:



Hình 7-8. Sơ đồ khối bên trong cảm biến khoảng cách GP2D12.



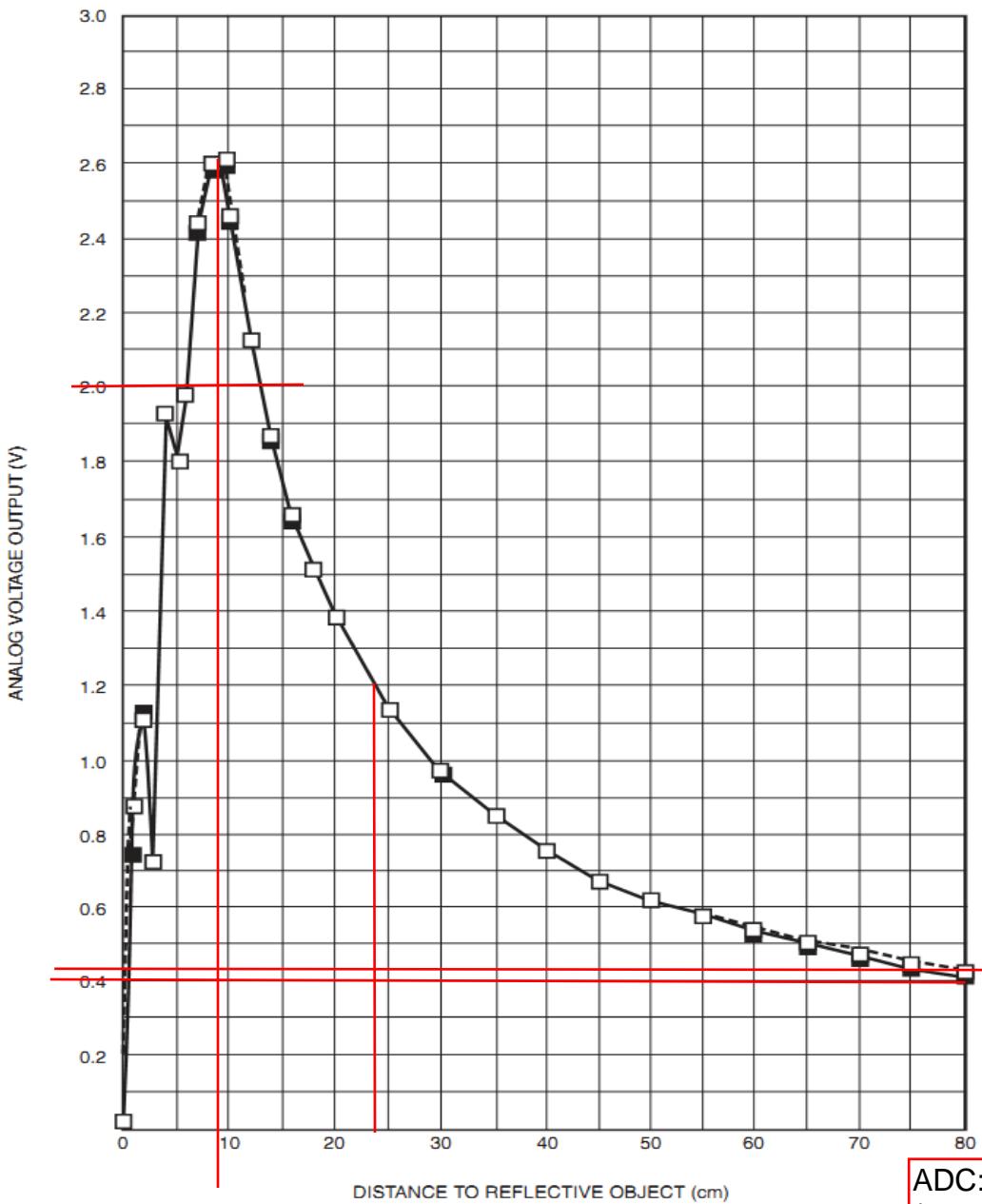
Hình 7-9. Dạng sóng của cảm biến khoảng cách GP2D12.

Quan sát dạng sóng ta thấy mỗi chu kỳ chuyên đổi tối đa là 38,3ms.

Chu kỳ đầu tiên khi cấp điện thì ngõ ra chưa ổn định nên bỏ.

Đồ thị chuyển đổi của cảm biến GP2D12:

- Hình 7-10 trình bày đồ thị chuyển đổi của cảm biến:



Hình 7-10. Đồ thị chuyển đổi của cảm biến khoảng cách GP2D12

Quan sát đồ thị ta thấy trực hoành là khoảng cách với đơn vị là cm, trực tuyế

ADC: step size
(Tr.193)
4,887 mV - 1 đơn vị
5mv - 1 đơn vị
400mV --> 80

Bảng 7-1. Giá trị tại các tọa độ chính.

Toạ độ	Khoảng cách (cm)	Điện áp ra (mV)	DL số	Chênh lệch số	Chia cho 9 cấp
1	80	400	70	80	
2	70	435	87	7	0.77
3	60	530	106	19	2.11

4	50	620	124	18	2	
5	40	750	150	26	2.88	
6	30	970	196	46	5.11	
7	20	1370	276	70	8.88	
8	10	2450	490	214	23.77	

7.3.2 Đo khoảng cách hiển thị trên 4 led 7 đoạn

Bài mẫu 711. Chương trình đo khoảng cách dùng led thu phát GP2D12 hiển thị trên 2 led 7 đoạn. Hiển thị với độ phân giải 10cm.

Lưu tên file là “BAI_711_GP2D12_SS10CM”.

- a. Mục đích: biết lập trình đo khoảng cách từ cảm biến GP2D12 và xử lý kết quả đo với độ phân giải 10cm.
- b. Lưu đồ: sinh viên hãy tự viết.
- c. Chương trình:

```
#include <tv_pickit2_shift_1.c>
unsigned int16 kqadc,k_cach;
void main()
{
    set_up_port_ic_chot();
    setup_adc(adc_clock_div_32);
    setup_adc_ports(an0_to_an2|vss_vdd );
    set_adc_channel(2);
    while(true)
    {
        kqadc=read_adc();
        delay_ms(30);
        if (kqadc>=490) k_cach=10;
        else if (kqadc>276) k_cach=20;
        else if (kqadc>194) k_cach=30;
        else if (kqadc>150) k_cach=40;
        else if (kqadc>122) k_cach=50;
        else if (kqadc>106) k_cach=60;
        else if (kqadc>96) k_cach=70;
        else if (kqadc>80) k_cach=80;
        xuat_4led_7doan_giaima_xoa_so0(k_cach);
    }
}
```

set_adc_channel(0);
 set_adc_channel(2);
 --> chỉ chọn kênh 2
 --> tham khảo bài đo
 2 kênh nhiệt độ
 --> set_adc_channel
 trong hàm con

- d. Tiến hành biên dịch và nạp.
- e. Quan sát kết quả: Bạn để tay che cảm biến GP2D12 cách khoảng 12cm và dịch chuyển tay ra xa và đồng thời quan sát kết quả trên led đơn tương ứng với khoảng cách đo được.
- f. Giải thích chương trình:

Bài mẫu 712. Chương trình đo khoảng cách dùng led thu phát GP2D12 hiển thị trên 2 led 7 đoạn. Hiển thị với độ phân giải 1cm.

Lưu tên file là “BAI_712_GP2D12_SS1CM”.

- Mục đích: biết lập trình đo khoảng cách và xử lý kết quả đo theo độ phân giải 1cm.
- Lưu đồ: sinh viên hãy tự viết.
- Chương trình:

```
#include <tv_pickit2_shift_1.c>
unsigned int8 j;
unsigned int16 kqadc;
float k_cach, clk_cach;
void giao_ma_hienthi_4led (float tam)
{
    unsigned int16 x;
    x =(unsigned int16 ) tam;
    xuat_4led_7doan_giaima_xoa_so0(x);
}
void main()
{
    set_up_port_ic_chot();
    setup_adc(adc_clock_div_32);
    setup_adc_ports(an0 to an2|vss_vdd );
    set_adc_channel(2);
    while(true)
    {
        kqadc=0;
        for (j=0; j<20; j++)
        {
            kqadc=kqadc+read_adc();
            delay_ms(10);
        }
        kqadc=kqadc /20;
        if (kqadc <= 80) k_cach=80;
        else if ((kqadc>80) && (kqadc<87))
        {
            clk_cach = kqadc-80;
            k_cach=80 - clk_cach/0.77;
        }
        else if (kqadc==87) k_cach=70;
        else if ((kqadc>87) && (kqadc<106))
        {
            clk_cach = kqadc-87;
            k_cach=70 - clk_cach/2.11;
        }
        else if (kqadc==106) k_cach=60;
        else if ((kqadc>106) && (kqadc<124))
        {
            clk_cach = kqadc-106;
            k_cach=60 - clk_cach/2;
        }
    }
}
```

viết thành hàm con

VOID DO_KC()

```
{
    SET_ADC_CHANNEL(2);
    delay_ms(1);
    ...
}
```

khÔng cÁch

dÙi liÊu adc chuyÊn
ĐÔi

$$\begin{aligned} 83-80 &= 3 \\ 80\text{cm} - 3/0.77 &\\ (3.89\text{cm}) & \\ 77.11 \text{ cm} & \end{aligned}$$

bÀi mÃu b!
dÒi dÂu }
xuÔng phÃa
dUÓi

```

        else if (kqadc==124) k_cach=50;
        else if ((kqadc>124) &&(kqadc<150))
        {
            clk_cach = kqadc-124;
            k_cach=50 - clk_cach/2.88;
        }
        else if (kqadc==150) k_cach=40;
        else if ((kqadc>150) &&(kqadc<196))
        {
            clk_cach = kqadc-150;
            k_cach=40 - clk_cach/5.11;
        }
        else if (kqadc==196) k_cach=30;
        else if ((kqadc>196) &&(kqadc<276))
        {
            clk_cach = kqadc-196;
            k_cach=30 - clk_cach/8.88;
        }
        else if (kqadc==276) k_cach=20;
        else if ((kqadc>276) &&(kqadc<490))
        {
            clk_cach = kqadc-276;
            k_cach=20 - clk_cach/23.77;
        }
        else if (kqadc==490) k_cach=10;
        giai_ma_hienthi_4led(k_cach);
    }
}

```

- d. Tiến hành biên dịch và nạp.
- e. Quan sát kết quả: giống bài 711 nhưng giá trị thay đổi theo 1cm.
- f. Giải thích chương trình:

Bài mẫu 713. Chương trình đo khoảng cách dùng led thu phát GP2D12 hiển thị trên 2 led 7 đoạn. Hiển thị với độ phân giải 1cm. **Dùng phương trình để tính khoảng cách.**

Lưu tên file là “BAI_713_GP2D12_SS1CM”.

- a. Mục đích: biết lập trình đo khoảng cách và xử lý kết quả đo theo hàm mũ.
- b. Lưu đồ: sinh viên hãy tự viết.
- c. Chương trình:

```

#include <tv_pickit2_shift_1.c>
#include <math.h>
unsigned int16 kqadc;
float k_cach;
void giao_ma_hienthi_4led (float tam)
{
    unsigned int16 x;
    x =(unsigned int16 ) tam;
    xuat_4led_7doan_giaima_xoa_so0(x);
}

```

```

void main()
{
    set_up_port_ic_chot();
    setup_adc(adc_clock_div_32);
    setup_adc_ports(an0_to_an2|vss_vdd );
    set_adc_channel(2);
    while(true)
    {
        kqadc=read_adc();
        delay_ms(40) ←
        k_cach=pow(4277/kqadc,1.106);

        if (k_cach>80) k_cach=80;
        giao_ma_hienthi_4led(k_cach);
    }
}

```

CCS: F1 --> POW

```

void doc_nd_lm35a()
{
    set_adc_channel(0);
    lm35a = 0;
    for (j=0; j<solan; j++)
    {
        lm35a = lm35a + read_adc();
        delay_us(100);
    }
    lm35a = lm35a /2.046;
    lm35a = lm35a /solan;
}

```

705

712

412 --> 616

Tiến hành biên dịch và nạp.

Quan sát kết quả: giống bài 711 nhưng giá trị thay đổi theo 1cm.

Giải thích chương trình:

Bài tập 714. Hãy viết chương trình đo hiển thị trên LCD:

Đo nhiệt độ dùng LM35A và LM35B hiển thị ở hàng thứ 1. ←

Đếm sản phẩm dùng counter T0 hiển thị ở hàng thứ 2.

Đo khoảng cách dùng cảm biến GP2D12 và hiển thị hàng thứ 3.

Lưu tên file là “BAI_714_LCD_2LM35_T0_GP2D12”

tham khảo Bài 705/198

```

void DO_KC()
{
    set_adc_channel(2);
    ....
}

```

7.4 ĐO KHOẢNG CÁCH DÙNG CẢM BIẾN SIÊU ÂM HC-SR 04

7.4.1 Cảm biến HC-SR 04

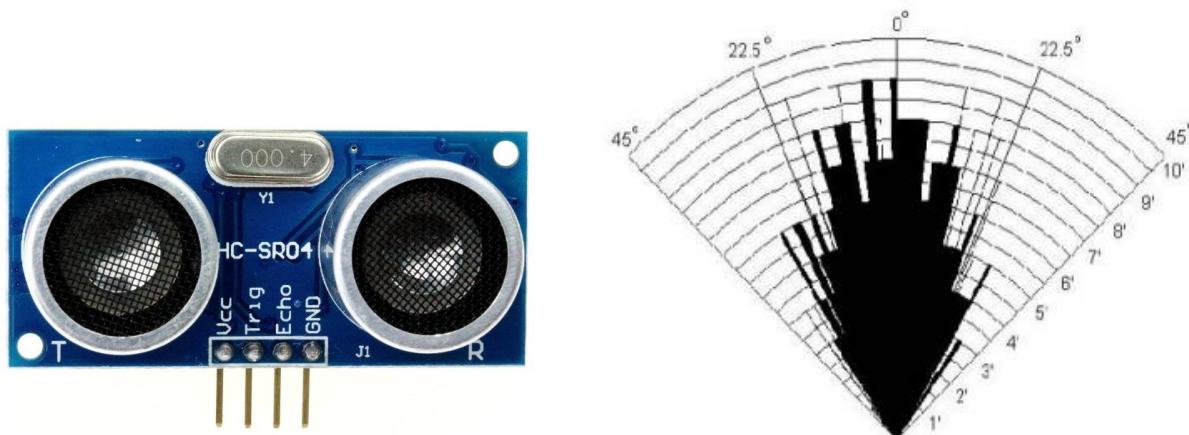
Phần này thực hành cảm biến siêu âm HC-SR 04.

Tính chất của cảm biến HC-SR 04

- Điện áp nguồn cung cấp 5V DC
- Dòng tính < 2mA
- Góc có hiệu lực < 15°
- Khoảng cách đo từ 2cm đến 500cm
- Độ phân giải 0.3 cm

Hình dạng bên ngoài và tên các tín hiệu

- Hình dạng bên ngoài và góc phát sóng như hình 7-11:

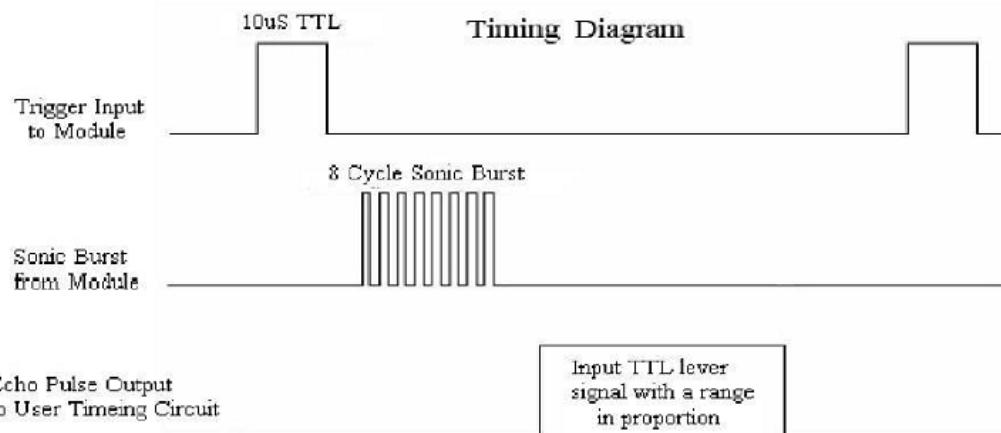


Hình 7-11. Hình cảm biến siêu âm và góc phát thu sóng.

Cảm biến HC-SR 04 có 4 chân: chân cấp nguồn Vcc, chân kích (trigger), chân nhận tín hiệu dội (echo) và chân GND.

Sau này có loại HC-SR 05 có thêm 1 tín hiệu out.

Giản đồ thời gian các tín hiệu:



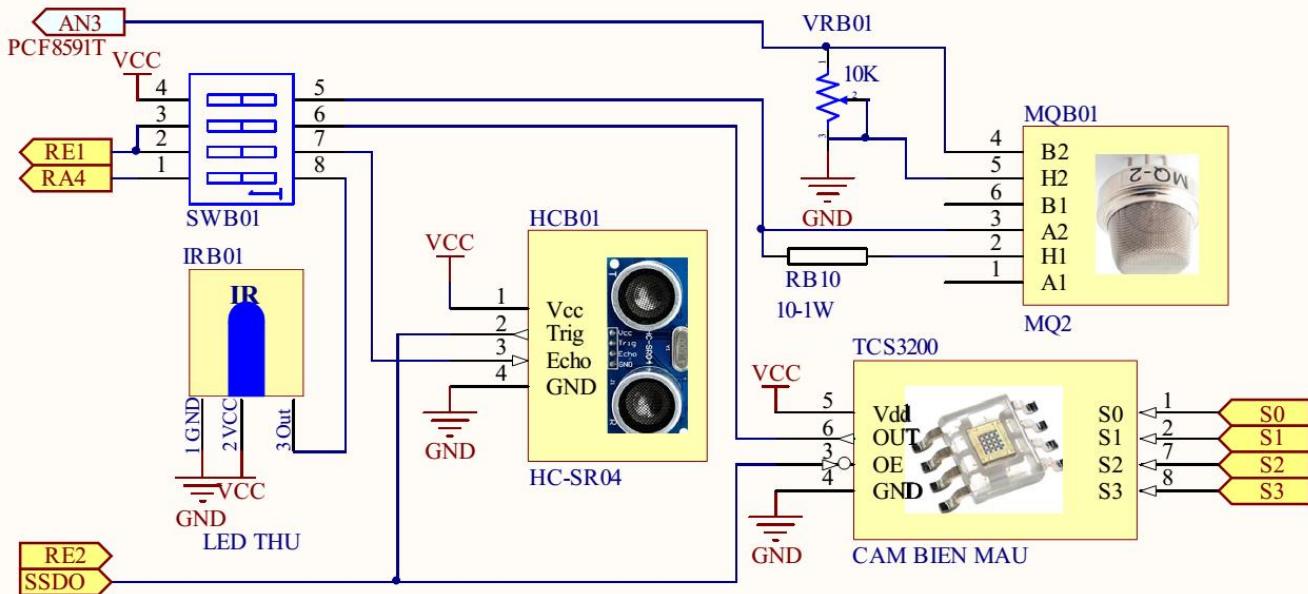
Hình 7-12. Giản đồ thời gian của cảm biến HC-SR04.

Để đo khoảng cách dùng cảm biến siêu âm thì hệ thống điều khiển phải kích tín hiệu chân trigger tạo ra 1 xung có mức logic 1 tối thiểu là $10\mu s$. Tín hiệu tương thích chuẩn TTL. Sau đó thì mạch cảm biến sẽ tạo ra một chuỗi 8 xung phát ra ở biên tử phát, sau khi gặp vật cản nằm trong giới hạn đo thì sóng sẽ phản xạ về và tín hiệu echo lên mức 1 với thời gian ở mức 1 tỷ lệ thuận với khoảng cách đo.

Hệ thống phải đo thời gian tín hiệu echo ở mức logic 1 theo đơn vị thời gian là μs , lấy thời gian đo được chia cho 58 sẽ được khoảng cách cần đo theo đơn vị cm, chia cho 148 sẽ được khoảng cách theo đơn vị inch – các hệ số này do nhà cung cấp cho chúng ta – dựa vào tốc độ sóng phát đi và phản xạ về.

7.4.2 Cảm biến HC-SR 04 trong kit thực hành

Trong kit thực hành có gắn 1 cảm biến HC-SR 04 có sơ đồ kết nối như hình 7-13:

**Hình 7-13. Sơ đồ nguyên lý cảm biến khoảng cách siêu âm.**

Cảm biến siêu âm đã cấp nguồn, chân tín hiệu Trigger nối trực tiếp đến chân RE2/SSDO và chân Echo nối đến SWB01 và nối với chân RE1 cùng chân với tín hiệu OUT của cảm biến màu.

Khi sử dụng thì switch của cảm biến màu chuyển sang OFF, switch của cảm biến siêu âm chuyển sang ON, thường thì switch đã chuyển sẵn, các bạn không cần chuyển.

7.4.3 Đo khoảng cách dùng HCSR-04 hiển thị trên 4 led 7 đoạn

Bài mẫu 721. Chương trình đo khoảng cách dùng cảm biến khoảng cách HC-SR 04 hiển thị trên 4 led 7 đoạn.

Lưu tên file là “BAI_721_HCSR04_DO_KCSA”.

- Mục đích: biết đo khoảng cách dùng cảm biến siêu âm HCSR04 và xử lý kết quả đo hiển thị trên led.
- Lưu đồ: sinh viên hãy tự viết.
- Chương trình:

```
#include <tv_pickit2_shift_1.c>
#define trigger      pin_e2
#define echo        pin_e1
float    kcs;

void  giao_ma_hienthi_4led (float tam)
{
    unsigned int16 x;
    x =(unsigned int16 ) tam;
    xuat_4led_7doan_giaima_xoa_so0(x);
}

void  tao_xung_trigger()
{
    delay_ms(10);
}
```

```

        output_high(trigger);
        delay_ms(10);
        output_low(trigger);

        set_timer1(0);
        while(!input(echo));
        setup_timer_1(t1_internal|t1_div_by_4);

        while(input(echo));
        kcs =get_timer1();
        setup_timer_1(t1_disabled );
    }

void main()
{
    set_up_port_ic_chot();
    setup_timer_1(t1_disabled );
    giao_ma_hienthi_4led(kcs);
    while(true)
    {
        tao_xung_trigger();
        kcs=kcs*0.8;
        kcs = (kcs/58);
        giao_ma_hienthi_4led(kcs);
        delay_ms(500);
    }
}

```

- d. Tiến hành biên dịch và nạp.
- e. Quan sát kết quả: Bạn để tay hoặc sách học che cảm biến siêu âm và dịch chuyển tay ra xa, đồng thời quan sát kết quả trên led đơn tương ứng với khoảng cách đo được, phạm vi đo khoảng 400 cm. Chú ý các nhóm đối diện nhau có thể ảnh hưởng sóng cho nhau.
- f. Giải thích chương trình:

Bài tập 722. Hãy viết chương trình đo khoảng cách dùng cảm biến HCSR04 và hiển thị kết quả đo bằng số lớn trên LCD.

Lưu tên file là “BAI_722_LCD_HCSR04”.

Bài tập 722. Hãy viết chương trình đo khoảng cách dùng cảm biến HCSR04 hiển thị trên LCD.

Lần đo thứ 1 hiển thị ở hàng 1 cho đến khi nhấn BT0 thì giữ nguyên kết quả ở hàng 1.

Lần đo thứ 2 hiển thị ở hàng 2 cho đến khi nhấn BT0 thì giữ nguyên kết quả ở hàng 2.

Sau đó hàng 3 hiển thị tích của 2 lần đo ở hàng 1 và hàng 2.

Nhấn BT1 thì xóa hết và tiến hành đo lại.

Lưu tên file là “BAI_722_LCD_HCSR04_DO_DIENTICH”.

7.5 CẢM BIẾN NHIỆT 1 DÂY

7.5.1 Cảm biến nhiệt DS18B20

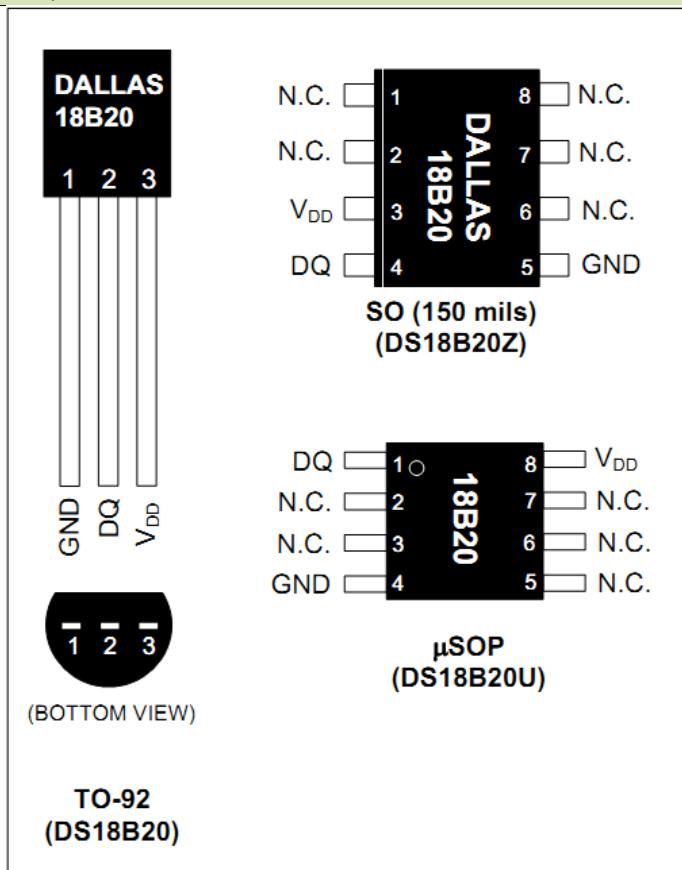
Ở phần trước chúng ta đã sử dụng cảm biến nhiệt là LM35 và bộ chuyển đổi ADC để chuyển đổi điện áp tương tự thành tín hiệu số, ở phần này chúng ta sẽ nghiên cứu và sử dụng cảm biến nhiệt có luôn bộ chuyển đổi ADC bên trong và chỉ có 1 tín hiệu giao tiếp vào ra còn gọi là 1 dây hay one wire.

Các thông tin của cảm biến như sau:

- Chỉ sử dụng một dây giao tiếp.
- Mỗi cảm biến đều có mã 64bit lưu trong bộ nhớ ROM.
- Là cảm biến đa năng thay cho hệ thống cảm biến nhiệt phức tạp.
- Phần cứng đơn giản vì chỉ dùng 1 dây không cần dùng ADC nên rất thích hợp cho vi điều khiển không tích hợp ADC.
- Có thể cấp nguồn từ đường tín hiệu, điện áp từ 3V-5.5V.
- Tầm đo từ -55°C đến +125°C (-67°F đến +257°F).
- Sai số 0.5°C cho tầm đo từ -10°C đến +85°C.
- Có thể chọn độ phân giải bằng phần mềm từ 9 đến 12 bit.
- Thời gian chuyển đổi lớn nhất cho 12bit là 750ms.
- Có thể cài đặt nhiệt độ cảnh báo lưu vào bộ nhớ ROM.
- Bộ cảnh báo luôn thăm dò để gửi lệnh đến thiết bị khi nhiệt độ vượt qua giới hạn cho phép.
- Thiết bị này có nhiều dạng chân: loại 3 chân, 8 chân, hay loại dán.
- Thiết bị được ứng dụng rộng rãi trong kiểm soát nhiệt độ: hệ thống công nghiệp, sản phẩm hàng tiêu dùng, nhiệt kế hay bất kỳ một hệ thống cảm biến nhiệt nào.

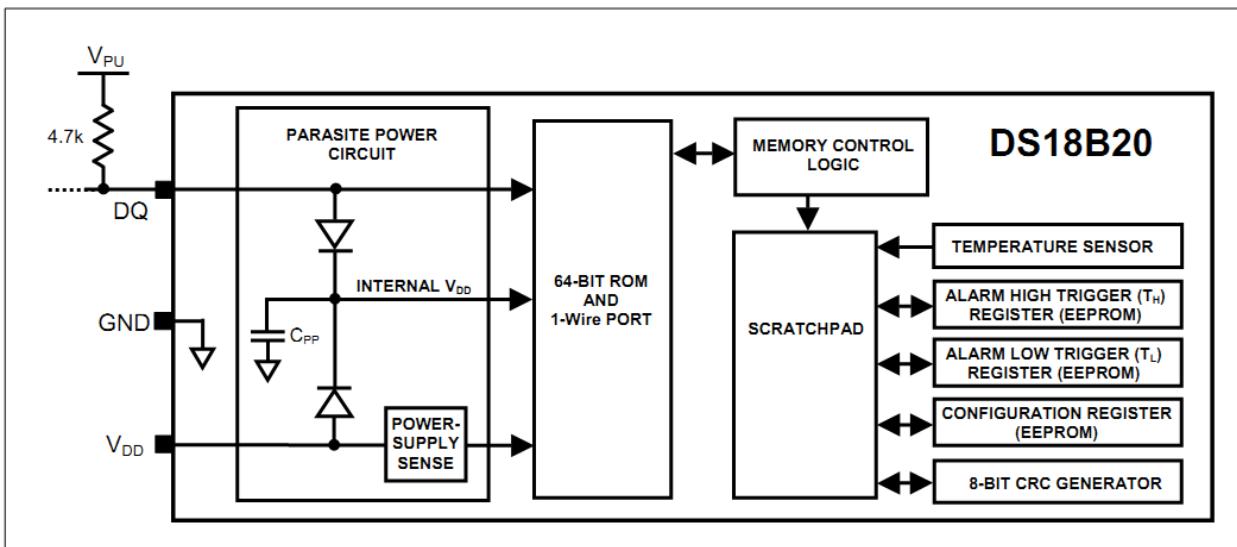
7.5.2 Hình ảnh và tên các chân của cảm biến

Các hình dạng khác nhau của cảm biến như hình 7-14.

*Hình 7-14. Các dạng cảm biến DS18B20.*

7.5.3 Sơ đồ khối của cảm biến DS18B20

Cảm biến DS18B20 có sơ đồ khối như hình 7-15.

*Hình 7-15. Sơ đồ khối của cảm biến DS18B20.*

Các khối bao gồm:

- Khối mạch nguồn cấp cho các khối bên trong hoạt động lấy từ nguồn bên ngoài.

- Khối bộ nhớ ROM 64 bit và port 1 dây.
- Khối điều khiển bộ nhớ.
- Khối scratchpad liên kết với các khối cảm biến nhiệt độ, khối lưu nhiệt độ byte cao T_H và byte thấp T_L để báo động quá nhiệt, khối thanh ghi định cấu hình và bộ kiểm tra mã dùng chuẩn CRC.

Bộ nhớ ROM lưu trữ mã 64 bit duy nhất của cảm biến.

Vùng nhớ scratchpad có 2 byte thanh ghi dùng để lưu trữ dữ liệu số là nhiệt độ từ cảm biến. Ngoài ra còn có 2 thanh ghi lưu nhiệt độ ngưỡng dưới và ngưỡng trên để so sánh với nhiệt độ đo và cảnh báo khi vượt quá giới hạn cài đặt, tên 2 thanh ghi là T_L và T_H.

Thanh ghi định cấu hình cho phép người dùng thiết lập độ phân giải chuyển đổi từ nhiệt độ sang dữ liệu số là 9, 10, 11 hoặc 12 bit. Giá trị thanh ghi định cấu hình và T_H, T_L còn được lưu vào bộ nhớ không bay hơi (EEPROM) nên dữ liệu vẫn còn khi mất điện.

7.5.4 Hoạt động của cảm biến DS18B20

Chức năng chính của DS18B20 là cảm biến nhiệt độ chuyển đổi trực tiếp sang dữ liệu số. Độ phân giải của cảm biến có thể định cấu hình là 9, 10, 11 hoặc 12 bit tương ứng với hệ số chuyển đổi theo thứ tự là 0,5°C, 0,25°C, 0,125°C và 0,0625°C. Độ phân giải mặc nhiên khi cấp điện là 12bit.

Khi cấp điện thì cảm biến DS18B20 ở trạng thái nghỉ để giảm công suất tiêu thụ. Để bắt đầu quá trình chuyển đổi tương tự sang số của nhiệt độ đo thì thiết bị chủ phải cấp phát lệnh chuyển đổi (Convert_T có mã là 44H). Sau khi chuyển đổi xong thì dữ liệu nhiệt độ lưu vào 2 thanh ghi nhiệt độ trong vùng nhớ scratchpad và cảm biến trở lại trạng thái nghỉ.

Nếu cảm biến cấp nguồn bên ngoài thì thiết bị chủ có thể phát các khe thời gian đọc sau khi cấp phát lệnh chuyển đổi nhiệt độ (Convert_T) để kiểm tra cảm biến chuyển đổi xong chưa: nếu cảm biến đáp ứng bằng cách phát bit có mức logic 0 thì quá trình chuyển đổi đang xảy ra, nếu phát bit có mức logic 1 thì quá trình chuyển đổi đã kết thúc.

Nếu cảm biến cấp nguồn thông qua chân tín hiệu thì thực hiện kiểm tra theo cách khác.

Dữ liệu nhiệt độ thuộc hệ số Celsius, nếu $1/2 = 0.5$ enheit thì phải thực hiện chuyển đổi. Dữ liệu nhiệt độ lưu trữ ở dạng số bù 2 số có dấu 16 bit trong thanh ghi như hình 7-16.

LS BYTE	BIT 7	BIT 6	BIT 5	BIT 4	BIT 3	BIT 2	BIT 1	BIT 0
	2^3	2^2	2^1	2^0	2^{-1}	2^{-2}	2^{-3}	2^{-4}
MS BYTE	BIT 15	BIT 14	BIT 13	BIT 12	BIT 11	BIT 10	BIT 9	BIT 8
S = SIGN								

Hình 7-16. Cấu trúc 2 thanh ghi lưu nhiệt độ của cảm biến

Các bit dấu (S) có giá trị là 0 nếu là số dương và bằng 1 nếu là số

..... 0100

>>2

..... 01 00 (bỏ)

& 0...011 (0x0003)

Nếu cảm biến cấu hình ở độ phân giải 12 bit thì tất cả các bit trong thanh ghi đều lưu dữ liệu.

0.5 0.25

Nếu cảm biến cấu hình ở độ phân giải 11 bit thì bỏ bit thứ 0 (BIT 0).

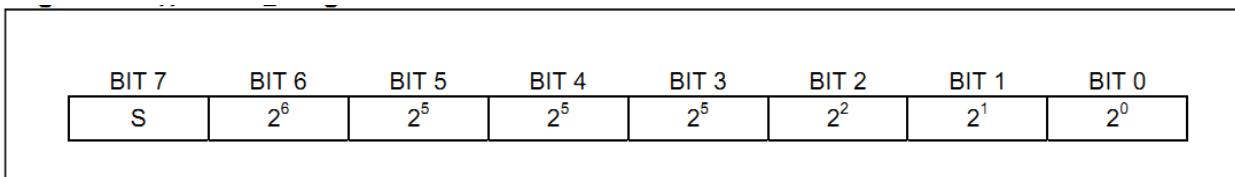
Nếu cảm biến cấu hình ở độ phân giải 10 bit thì bỏ 2 bit thứ 1 và 0 (BIT 1, 0).

Nếu cảm biến cấu hình ở độ phân giải 9 bit thì bỏ 3 bit thứ 2, 1 và 0 (BIT 2, 1, 0).

7.5.5 Hoạt động cảnh báo quá nhiệt của cảm biến DS18B20

$$2^{11} = 0.5$$

Sau khi cảm biến DS18B20 thực hiện xong chuyển đổi nhiệt độ thì giá trị nhiệt độ được so sánh với các giá trị nhiệt độ ngưỡng dưới và ngưỡng trên do người dùng thiết lập trong 2 thanh ghi T_H và T_L – có cấu trúc như hình 7-17.



Hình 7-17. Cấu trúc 2 thanh ghi lưu nhiệt độ báo động của cảm biến DS18B20.

Bit S là bit dấu để thiết lập giá trị dương và âm. Giá trị lưu trong 2 thanh ghi T_H và T_L không bị mất khi mất điện do có lưu trong bộ nhớ EEPROM. Có thể đọc giá trị của 2 thanh ghi T_H và T_L chính là byte thứ 2 và thứ 3 trong vùng nhớ scratchpad.

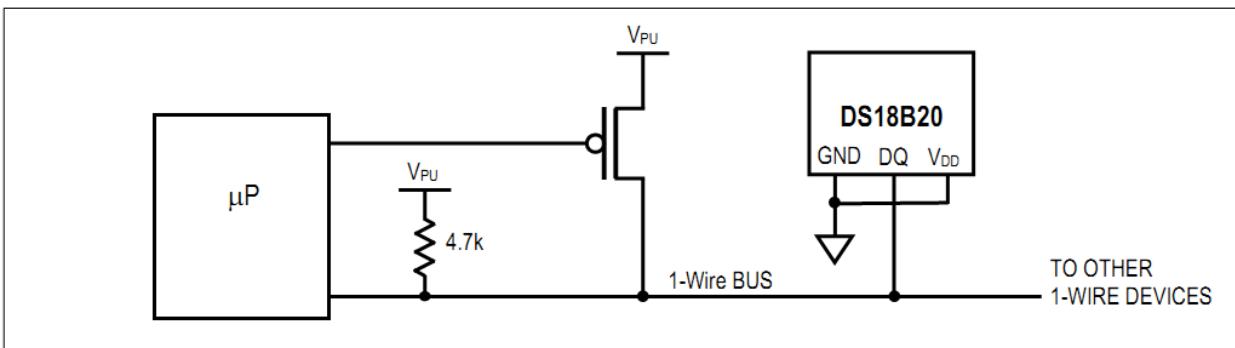
Chỉ có các bit từ bit thứ 11 đến bit thứ 4 (phần nguyên) của 2 thanh ghi nhiệt độ được dùng để so sánh với giá trị lưu trong 2 thanh ghi T_H và T_L .

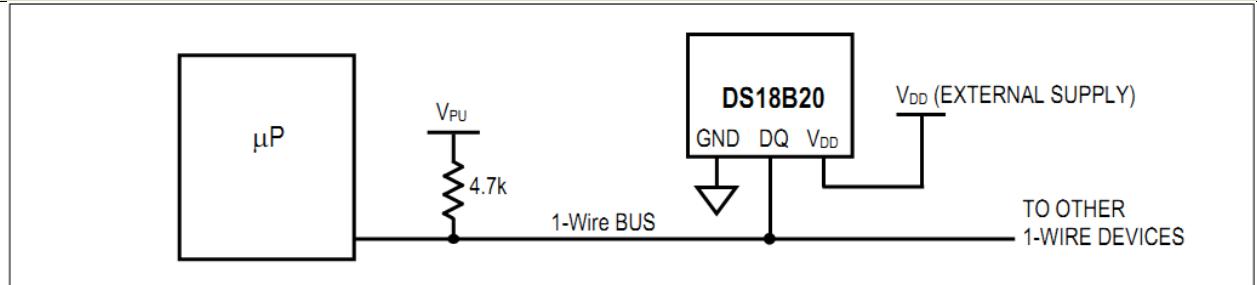
Nếu nhiệt độ đo thấp hơn hoặc bằng giá trị lưu trong thanh ghi T_L hoặc cao hơn hoặc bằng giá trị lưu trong thanh ghi T_H thì cờ báo động quá nhiệt trong cảm biến DS18B20 sẽ được set lên 1. Cờ báo động được cập nhật ở mỗi lần đo nên nếu có báo động ở lần đo này thì cờ sẽ bị xóa sau khi thực hiện lần chuyển đổi tiếp theo.

Thiết bị chủ có thể kiểm tra trạng thái cờ của cảm biến DS18B20 nối với bus bằng cách phát lệnh tìm báo động (Alarm search có mã là ECH). Bất kỳ cảm biến nào có cờ báo động tích cực sẽ đáp ứng lệnh báo động nên thiết bị chủ có thể xác định chính xác cảm biến DS18B20 nào đã xảy ra báo động.

7.5.6 Cấp nguồn cho cảm biến DS18B20

Cảm biến DS18B20 có 2 kiểu cấp nguồn như hình 7-18.



**Hình 7-18. Sơ đồ cấp nguồn cho cảm biến DS18B20.**

Hình trên thì chân cấp nguồn là chân tín hiệu dùng thêm 1 transistor FET, hình bên dưới sử dụng nguồn cấp đúng chân nguồn V_{DD}.

7.5.7 Mã 64 bit của cảm biến DS18B20

Mỗi cảm biến DS18B20 có mã duy nhất là 64 bit như hình 7-19.

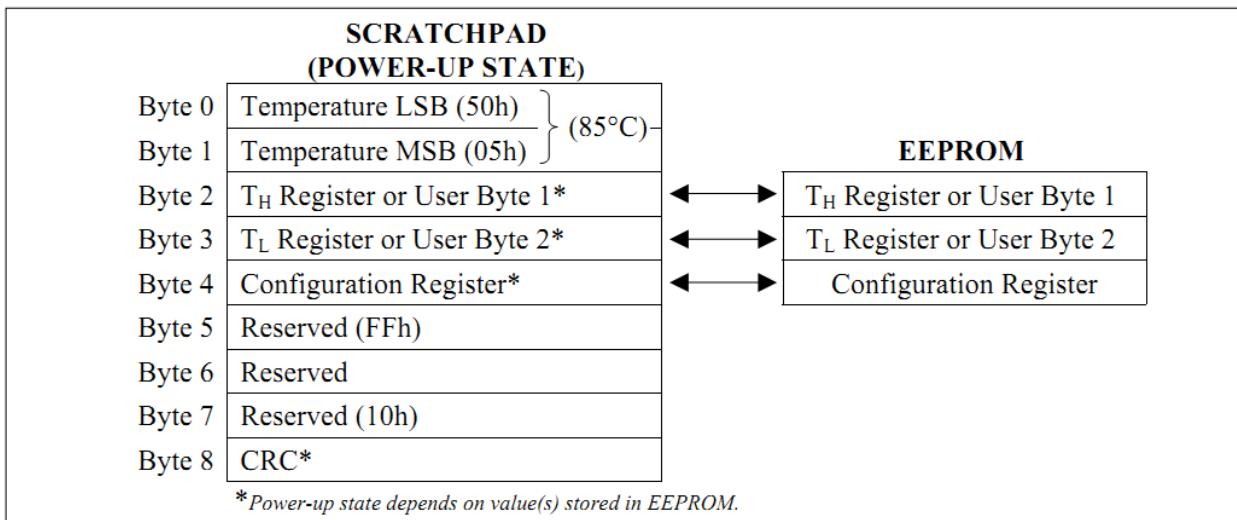
8-BIT CRC	48-BIT SERIAL NUMBER	8-BIT FAMILY CODE (28h)
MSB	LSB MSB	LSB MSB LSB

Hình 7-19. Cấu trúc mã 64 bit của cảm biến DS18B20.

Trong cấu trúc 64 bit thì 8 bit thấp luôn bằng 28H. 48 bit tiếp theo là mã duy nhất. 8 bit cuối cùng là kiểm tra mã thừa tuần hoàn (CRC cyclic redundancy check) được tính toán từ 56 bit. Bộ mã 64 bit lưu trong ROM và khối điều khiển ROM cho phép cảm biến DS18B20 hoạt động như thiết bị 1 dây.

7.5.8 Bộ nhớ rom của cảm biến DS18B20

Bộ nhớ ROM của cảm biến DS18B20 có tổ chức như hình 7-20.

**Hình 7-20. Tổ chức bộ nhớ ROM của cảm biến DS18B20.**

Bộ nhớ của cảm biến DS18B20 bao gồm bộ nhớ SRAM scratchpad và bộ nhớ EEPROM có thể xem là các thanh ghi dùng để lưu trữ giá trị nhiệt độ để so sánh báo động quá nhiệt – hai

thanh ghi có tên là T_H và T_L , một thanh ghi định cấu hình. Nếu chức năng lưu nhiệt độ báo động không được sử dụng thì 2 thanh ghi này có thể dùng để lưu dữ liệu như các ô nhớ thông dụng.

Byte thứ 0 và thứ 1 dùng để lưu nhiệt độ chuyển đổi và theo thứ tự là byte thấp và byte cao, hai byte này chỉ cho phép đọc.

Byte thứ 2 và thứ 3 dùng để lưu nhiệt độ báo động và theo thứ tự là byte thấp và byte cao.

Byte thứ 4 là thanh ghi định cấu hình.

Byte thứ 5, 6, 7 phục vụ cho hoạt động bên trong của cảm biến.

Byte thứ 8 là byte kiểm tra cho tất cả các byte từ thứ 0 đến 7 của vùng nhớ này.

Dữ liệu ghi vào byte 2, 3 và byte 4 của vùng nhớ bằng lệnh Write Scratchpad (4EH), dữ liệu gửi đến DS18B20 theo trình tự bắt đầu với bit có trọng số nhỏ nhất LSB của byte thứ 2 rồi đến byte 3 và byte 4.

Để kiểm tra dữ liệu lưu vào có đúng không thì tiến hành đọc lại bằng lệnh Read Scratchpad (BEH). Khi đọc vùng nhớ Scratchpad thì bit dữ liệu sẽ bắt đầu với bit có trọng số nhỏ nhất của byte thứ 0.

Để truyền giá trị cho thanh ghi T_H , T_L và thanh ghi định cấu hình từ vùng nhớ Scratchpad vào bộ nhớ EEPROM thì phải dùng lệnh Copy Scratchpad (48H).

Dữ liệu trong thanh ghi của vùng nhớ EEPROM không bị mất khi mất điện, khi có điện thì dữ liệu trong bộ nhớ EEPROM sẽ nạp vào vùng nhớ Scratchpad đúng với các giá trị và thanh ghi đã thiết lập.

7.5.9 Thanh ghi định cấu hình của cảm biến DS18B20

Byte thứ 4 của vùng nhớ Scratchpad của cảm biến DS18B20 là thanh ghi định cấu hình có cấu trúc như hình 7-21. Thanh ghi này cho phép thiết lập độ phân giải chuyển đổi của cảm biến bằng 2 bit có tên là R1 và R0. Hai bit sẽ cho phép thiết lập 4 hệ số chuyển đổi 9 bit đến 12 bit và thời gian chuyển đổi tương ứng như bảng 7-2.

0	0	1	1				
BIT 7	BIT 6	BIT 5	BIT 4	BIT 3	BIT 2	BIT 1	BIT 0
0	R1	R0		1	1	1	1

Hình 7-21. Byte thanh ghi điều khiển của cảm biến DS18B20.

Bảng 7-2. Độ phân giải và thời gian chuyển đổi.

R1	R0	RESOLUTION (BITS)	MAX CONVERSION TIME
0	0	9	$t_{CONV}/8$
0	1	10	$t_{CONV}/4$
1	0	11	$t_{CONV}/2$
1	1	12	t_{CONV}

7.5.10 Trình tự hoạt động của cảm biến DS18B20

Trình tự để truy xuất cảm biến DS18B20 như sau:

- Bước 1: khởi động.
- Bước 2: lệnh truy xuất ROM.
- Bước 3: lệnh chức năng DS18B20.

Khởi động: tất cả các giao tiếp với bus 1 dây đều phải bắt đầu với trình tự động. Trình tự khởi động bao gồm 1 xung reset được phát bởi thiết bị chủ và tiếp theo là xung của thiết bị tớ hay của cảm biến DS18B20. Chức năng của xung reset có chức năng kiểm tra thiết bị tớ là cảm biến có nối với bus hay không, nếu có thì thiết bị tớ là cảm biến sẽ trả lời bằng cách tạo ra 1 xung tương ứng – quá trình này còn gọi là kiểm tra sự hiện diện của cảm biến.

Thời gian xung kiểm tra của thiết bị chủ và tớ đều phải tuân thủ quy định về thời gian rất chính xác, nếu sai thì mạch sẽ không hoạt động đúng hay dữ liệu nhận về và gửi đi sẽ sai.

7.5.11 Các lệnh hoạt động của cảm biến DS18B20

a. Các lệnh liên quan đến bộ nhớ rom

Sau khi thiết bị chủ kiểm tra sự hiện diện của cảm biến thì tiếp theo là phát các lệnh đọc ROM. Các lệnh này hoạt động dựa trên bộ mã 64 bit duy nhất lưu trong ROM của mỗi thiết bị và cho phép thiết bị chủ truy xuất tới 1 thiết bị tớ chỉ định duy nhất nếu có nhiều thiết bị tớ là cảm biến nối chung với nhau.

Lệnh ROM cũng cho biết có bao nhiêu thiết bị tớ và loại thiết bị hiện diện nối với bus hoặc thiết bị nào đã có báo động quá nhiệt.

Có 5 lệnh ROM và mỗi lệnh dài 8 bit, lưu đồ hoạt động ROM được trình bày như hình 7-22.

1. Lệnh tìm ROM – Search ROM (mã lệnh là F0H)

Khi hệ thống được cấp điện thì thiết bị chủ tiến hành nhận dạng mã ROM của tất cả các thiết bị tớ nối với bus, kết quả của nhận dạng này cho phép thiết bị chủ xác định số lượng thiết bị tớ và loại thiết bị. Thiết bị chủ đọc mã ROM thông qua quá trình thử và loại – quá trình này yêu cầu thiết bị chủ thực hiện chu kỳ lệnh Search ROM nhiều lần để nhận dạng tất cả các thiết bị tớ.

Sau khi thực hiện xong chu kỳ lệnh tìm ROM thì thiết bị chủ phải trở lại bước 1 để khởi tạo lại khi thực hiện truy xuất thiết bị tớ.

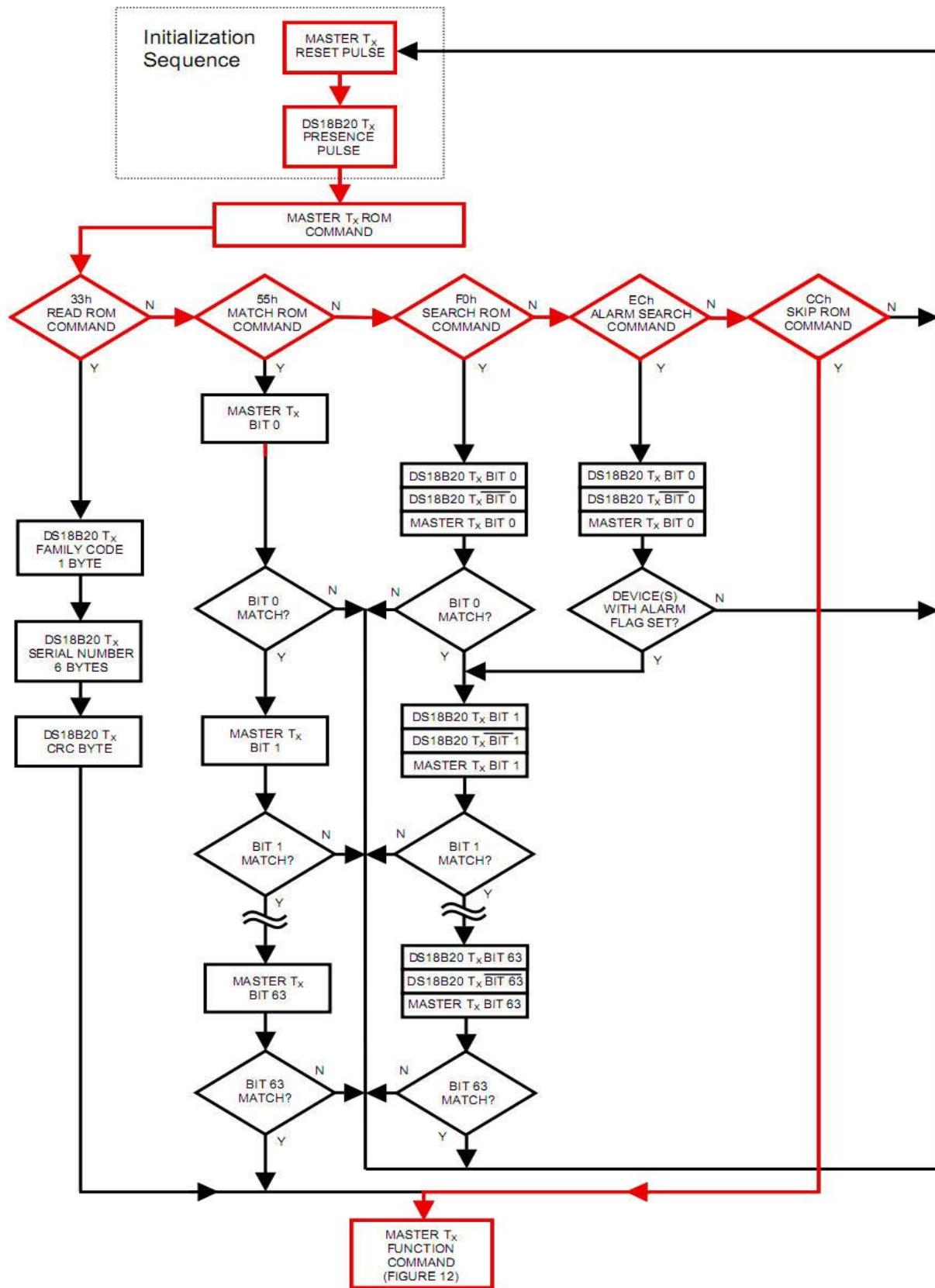
2. Lệnh đọc ROM – Read ROM (mã lệnh là 33H)

Lệnh đọc ROM chỉ có thể được khi chỉ có 1 thiết bị tớ nối với bus. Lệnh này cho phép thiết bị chủ đọc mã ROM 64 bit của thiết bị tớ mà không cần thực hiện lệnh tìm ROM. Nếu lệnh này được sử dụng khi có nhiều thiết bị tớ nối với bus thì dữ liệu sẽ bị sai vì tất cả các thiết bị tớ đều đáp ứng cùng một lúc.

3. Lệnh tương thích ROM – Match ROM (mã lệnh là 55H)

Lệnh tương thích ROM theo sau là mã 64 bit của ROM sẽ cho phép thiết bị chủ định địa chỉ của thiết bị tớ đúng với địa chỉ đó có nối với bus. Chỉ có thiết bị tớ tương thích với địa chỉ 64 bit của ROM sẽ đáp ứng với các lệnh yêu cầu từ thiết bị chủ, tất cả các thiết bị tớ còn lại trên bus sẽ đợi cho đến khi có xung reset tiếp theo.

Figure 11. ROM Commands Flowchart



Hình 7-22. Lưu đồ các lệnh liên quan đến ROM của cảm biến DS18B20.

4. Lệnh bỏ qua đọc ROM – Skip ROM (mã lệnh là CCH)

Thiết bị chủ dùng lệnh skip ROM để định địa chỉ cho tất cả các thiết bị trên bus cùng một lúc mà không cần gởi bất kỳ thông tin nào về code của ROM. Ví dụ thiết bị chủ có thể ra lệnh cho tất cả các thiết bị từ có trên bus thực hiện chuyển đổi nhiệt độ cùng một lúc bằng cách thực hiện lệnh Skip ROM và theo sau là lệnh chuyển đổi Convert T (44H).

5. Lệnh tìm báo động – Alarm Search (mã lệnh là ECH)

Hoạt động của lệnh này giống như hoạt động của lệnh tìm ROM chỉ ngoại trừ là thiết bị từ có cờ báo động tích cực sẽ đáp ứng.

Lệnh này cho phép thiết bị chủ xác định nếu bất kỳ thiết bị từ DS18B20 nào đã xảy ra điều kiện báo động trong quá trình chuyển đổi nhiệt độ gần nhất.

Sau khi thực hiện xong lệnh tìm báo động thì thiết bị chủ phải trở lại bước 1 để khởi động.

Lệnh tương thích ROM theo sau là mã 64 bit của ROM sẽ cho phép thiết bị chủ định địa chỉ của thiết bị từ đúng với địa chỉ có nối với bus. Chỉ có thiết bị từ tương thích với địa chỉ 64 bit của ROM sẽ đáp ứng với các lệnh yêu cầu từ thiết bị chủ, tất cả các thiết bị còn lại trên bus sẽ đợi cho đến khi có xung reset tiếp theo.

b. Lệnh chức năng của DS18B20

Sau khi thiết bị chủ thực hiện lệnh về ROM để địa chỉ hóa cảm biến DS18B20 thì thiết bị chủ có thể thực hiện các lệnh chức năng của DS18B20. Các lệnh này cho phép thiết bị chủ ghi và đọc bộ nhớ Scratchpad của DS18B20, bắt đầu thực hiện quá trình chuyển đổi và xác định chế độ nguồn cung cấp.

Các lệnh chức năng của DS18B20 được tóm tắt ở bảng theo sau và minh họa bằng lưu đồ như hình 7-23.

1. Lệnh chuyển đổi – Convert T (mã lệnh là 44H)

Lệnh này khởi động quá trình chuyển đổi duy nhất. Theo sau lệnh này thì kết quả nhiệt độ sau khi chuyển đổi được lưu vào 2 thanh ghi 2 byte nằm trong bộ nhớ Scratchpad và DS18B20 trả lại trạng thái chờ công suất thấp.

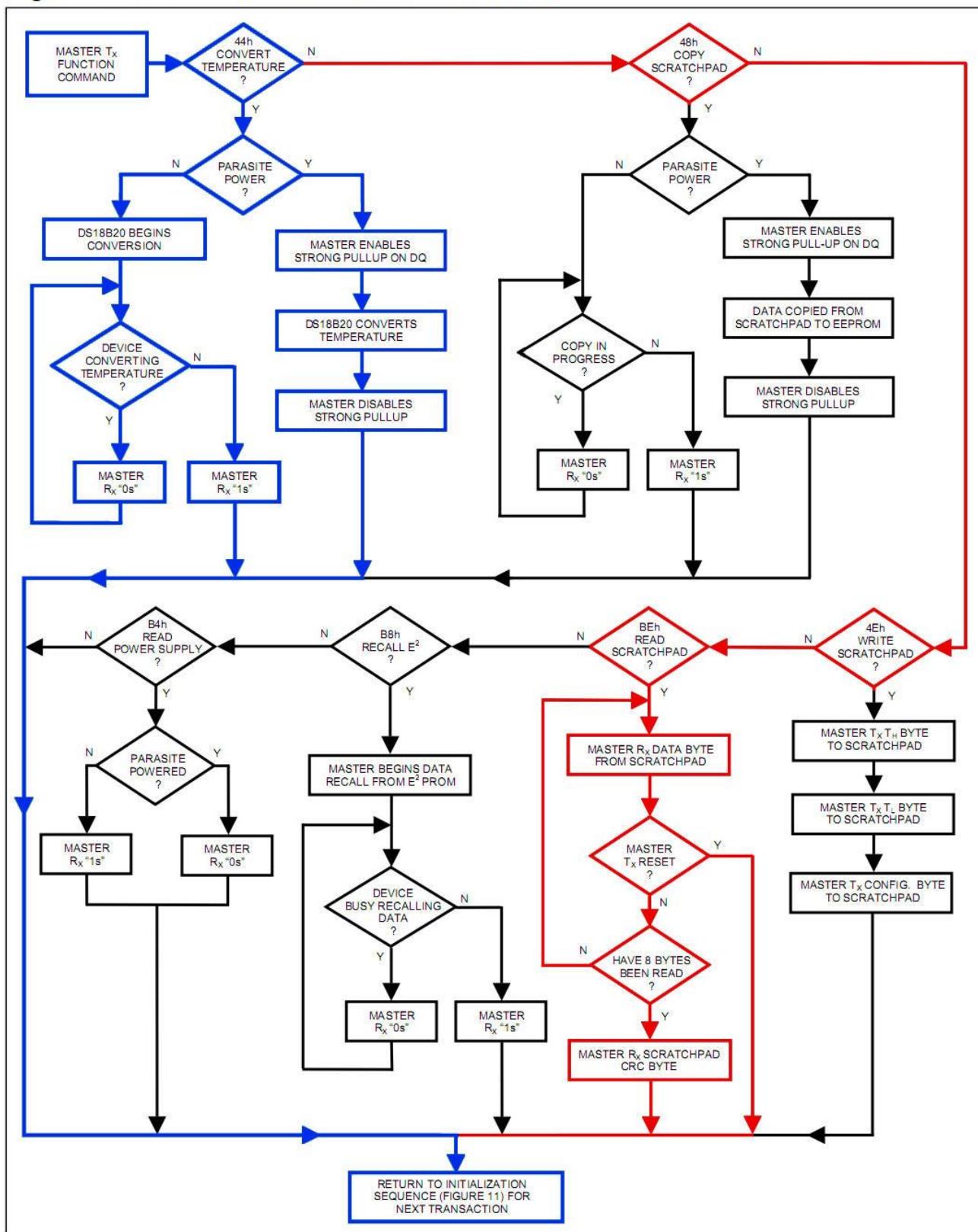
Nếu thiết bị đang sử dụng nguồn cung cấp thông qua chân tín hiệu thì trong khoảng 10 μ s (lớn nhất) sau khi thực hiện lệnh thì thiết bị chủ cho phép kéo lên cho bus nhanh trong khoảng thời gian chuyển đổi trình bày trong phần nguồn cung cấp.

Nếu thiết bị đang sử dụng nguồn cung cấp bên ngoài thì thiết bị chủ có thể thực hiện lệnh đọc theo sau khi thực hiện chuyển đổi Convert T và DS18B20 sẽ đáp ứng bằng cách phát mức logic 0 trong quá trình chuyển đổi và phát mức logic 1 khi quá trình chuyển đổi kết thúc.

2. Lệnh copy vùng nhớ Scratchpad – Copy Scratchpad (mã lệnh là 48H)

Lệnh này copy nội dung các thanh ghi T_H , T_L và thanh ghi định cấu hình của vùng nhớ SRAM Scratchpad (byte thứ 2, 3 và 4) vào bộ nhớ EEPROM.

Figure 12. DS18B20 Function Commands Flowchart



Hình 7- 23. Lưu đồ của các lệnh chức năng của cảm biến DS18B20.

3. Lệnh gọi lại – RECALL E²(mã lệnh là B8H)

Lệnh này nạp lại các giá trị báo động quá nhiệt (T_e và T_L) và dữ liệu định cấu hình từ bộ nhớ EEPROM sang bộ nhớ Scratchpad tương ứng.

Thiết bị chủ có thể cấp phát lệnh đọc sau lệnh Recall E². Cảm biến DS18B20 sẽ xác định trạng thái của lệnh Recall bằng cách phát mức logic 0 trong quá trình thực hiện lệnh Recall và phát mức logic 1 khi quá trình thực hiện lệnh recall hoàn tất.

Lệnh recall được thực hiện tự động khi cảm biến được cấp điện nên dữ liệu có hiệu lực trong bộ nhớ Scratchpad ngay khi thiết bị được cấp điện.

4. Lệnh đọc nguồn cung cấp – READ POWER SUPPLY (mã lệnh là B4H)

Thiết bị chủ cấp phát lệnh này sau khoảng thời gian đọc để xác định xem có bất kỳ cảm biến DS18B20 nào sử dụng nguồn cung cấp bằng đường tín hiệu.

Trong khoảng thời gian đọc nếu thiết bị dùng nguồn cung cấp là dây tín hiệu thì sẽ kéo bus xuống mức logic 0, nếu sử dụng nguồn bên ngoài thì bus vẫn ở mức logic 1.

5. Lệnh ghi dữ liệu vào bộ nhớ Scratchpad – WRITE SCRATCHPAD (mã lệnh là 4EH)

Lệnh này cho phép thiết bị chủ ghi 3 byte dữ liệu vào vùng nhớ scratchpad. Byte dữ liệu thứ 1 sẽ ghi vào thanh ghi T_H (là byte thứ 2 của vùng nhớ Scratchpad), byte thứ 2 ghi vào thanh ghi T_L (byte 3) và byte thứ 3 ghi vào thanh ghi định cấu hình (byte 4). Khi truyền thì byte có trọng số nhỏ nhất LSB được truyền đi đầu tiên. Tất cả các byte phải được ghi trước khi thiết bị chủ phát lệnh reset hoặc dữ liệu có thể bị ngắt.

6. Lệnh đọc dữ liệu bộ nhớ Scratchpad – READ SCRATCHPAD (mã lệnh là BEH)

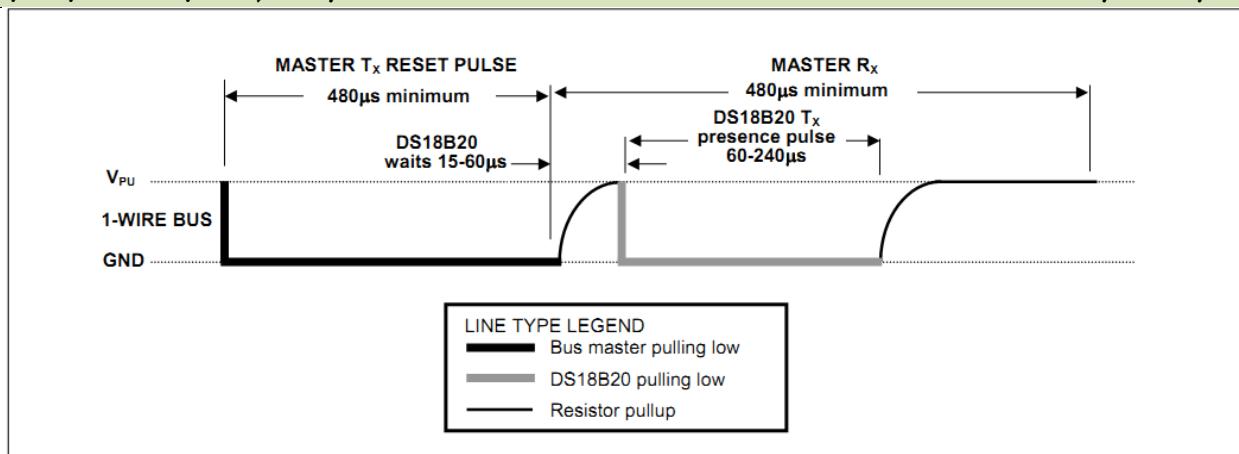
Lệnh này cho phép thiết bị chủ đọc nội dung của vùng nhớ Scratchpad. Khi truyền dữ liệu thì bắt đầu với bit LSB của byte thứ 0 và tiếp tục cho đến byte thứ 9. Thiết bị chủ có thể phát lệnh reset để chấm dứt quá trình đọc bất kỳ lúc nào nếu chỉ đọc một phần dữ liệu của bộ nhớ Scratchpad.

7.5.12 Các dạng tín hiệu của chuẩn 1 dây

Cảm biến DS18B20 dùng chuẩn truyền dữ liệu 1 dây. Có nhiều loại tín hiệu được định nghĩa bởi chuẩn này như sau: xung reset, xung hiện diện, ghi mức logic 0, ghi mức logic 1, đọc mức logic 0, đọc mức logic 1. Thiết bị chủ khởi động tất cả các tín hiệu này sau khi chấp nhận xung hiện diện.

7.5.13 Trình tự khởi động – xung reset và xung hiện diện

Tất cả các giao tiếp với cảm biến DS18B20 bắt đầu với trình tự khởi động bao gồm xung reset từ thiết bị chủ sau là xung hiện diện của cảm biến DS18B20. Trình tự này được minh họa như hình 7-24.



Hình 7-24. Dạng sóng của xung reset và xung hiện diện.

Khi cảm biến DS18B20 gởi xung hiện diện để đáp ứng xung reset thì đồng thời xác định với thiết bị chủ là cảm biến sẵn sàng hoạt động.

Trong quá trình khởi động thì thiết bị chủ (hoạt động ở chế độ phát Tx) tạo xung reset bằng cách kéo tín hiệu bus xuống mức thấp trong khoảng thời gian tối thiểu là 470μs. Sau đó thiết bị chủ thả nỗi tín hiệu bus và hoạt động ở chế độ nhận (hoạt động ở chế độ nhận Rx) để trao quyền điều khiển cho cảm biến DS18B20.

Khi thả nỗi thì điện trở kéo lên sẽ kéo bus về mức logic 1. Khi cảm biến DS18B20 phát hiện cạnh lên của xung thì nó sẽ đợi từ 15μs đến 60μs và sau đó phát đi 1 xung hiện diện bằng cách kéo bus xuống mức logic 0 trong khoảng thời gian từ 60μs đến 240μs.

7.5.14 Các khe thời gian đọc/ghi

Thiết bị chủ ghi dữ liệu vào DS18B20 trong các khe thời gian ghi và đọc dữ liệu từ DS18B20 trong các khe thời gian đọc. Một bit dữ liệu được phát lên bus trong 1 khe thời gian.

Khe thời gian ghi

Có 2 loại khe thời gian ghi là: khe thời gian ghi mức 1 (ghi dữ liệu mức 1 vào cảm biến DS18B20) và khe thời gian ghi mức 0 (ghi dữ liệu mức 0 vào cảm biến DS18B20). Tất cả các khe thời gian có lượng thời gian tối thiểu là 60μs và thời gian tín hiệu khôi phục lại từ mức 0 về mức 1 do điện trở kéo lên tối thiểu là 1 μs.

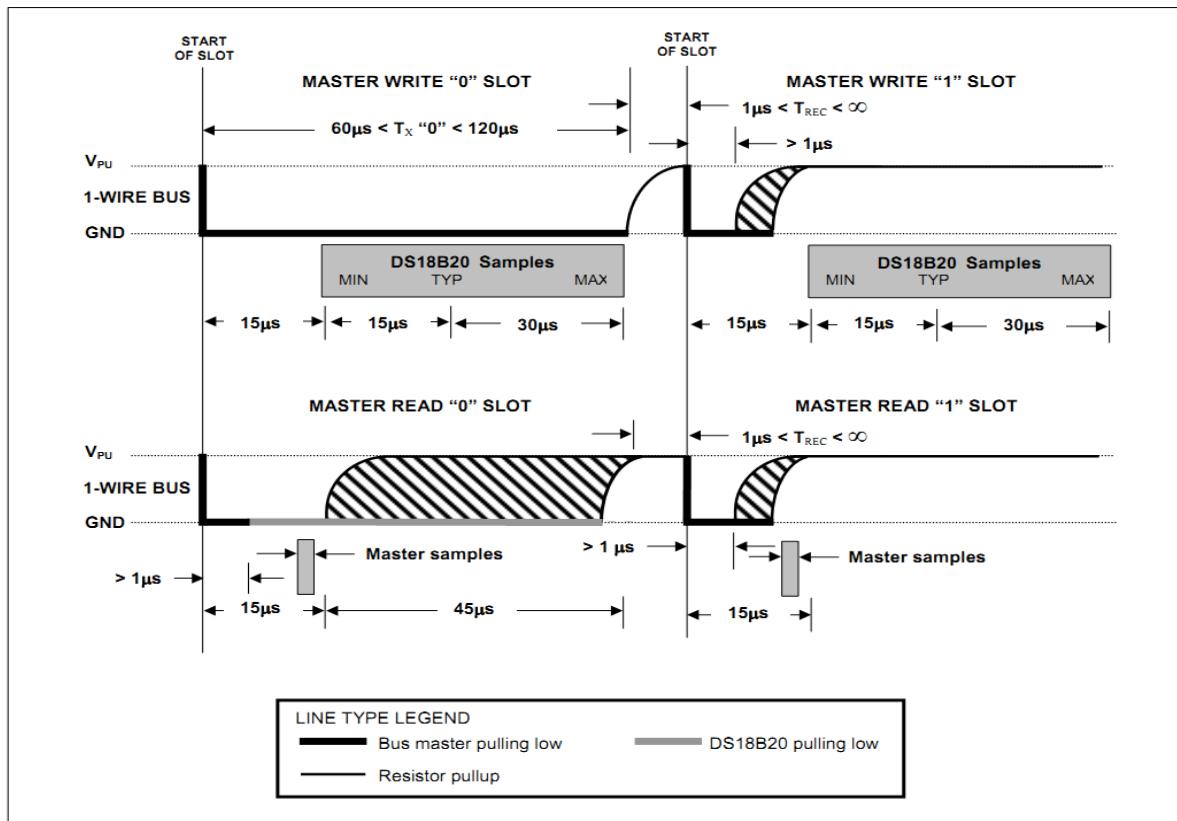
Cả 2 khe thời gian ghi dữ liệu mức logic 1 và 0 đều được khởi động bởi thiết bị chủ kéo bus xuống mức 0 như hình 7-25.

Để tạo ra khe thời gian ghi dữ liệu mức logic 1 thì sau khi phát xung kéo bus xuống mức 0 thì thiết bị chủ phải thả nỗi bus trong vòng 15μs. Khi thả nỗi bus thì điện trở kéo lên sẽ kéo bus về mức 1.

Để tạo ra khe thời gian ghi dữ liệu mức logic 0 thì sau khi phát xung kéo bus xuống mức 0 thì thiết bị chủ tiếp tục giữ bus ở mức 0 trong suốt khe thời gian (ít nhất là 60μs).

Cảm biến DS18B20 sẽ lấy mẫu từ bus với các khoảng thời gian kéo dài từ 15μs đến 60μs sau khi khởi động khe thời gian ghi.

Nếu bus ở mức cao trong suốt thời gian lấy mẫu thì mức 1 sẽ được ghi vào DS18B20, nếu ở mức 0 thì mức 0 được ghi vào.



Hình 7-25. Dạng sóng khe thời gian đọc/ghi.

Khe thời gian đọc

DS18B20 chỉ có thể truyền dữ liệu đến thiết bị chủ khi thiết bị chủ phát lệnh đọc các khe thời gian. Do đó, thiết bị chủ phải tạo các khe thời gian đọc ngay sau khi phát lệnh đọc bộ nhớ Scratchpad hoặc lệnh đọc nguồn cung cấp [B4H], khi đó DS18B20 có thể cung cấp dữ liệu yêu cầu.

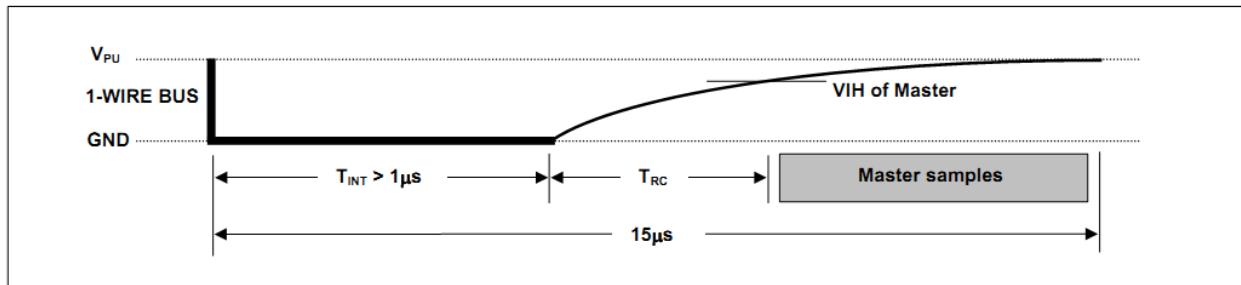
Tất cả các khe thời gian đọc có lượng thời gian tối thiểu là 60μs và thời gian khôi phục tối thiểu là 1μs giữa 2 khe thời gian.

Khe thời gian đọc dữ liệu bắt đầu bởi thiết bị chủ bằng cách kéo bus xuống mức 0 trong khoảng thời gian tối thiểu là 1μs và sau đó là thả nỗi bus – xem hình 7-24.

Sau khi thiết bị chủ khởi tạo xong khe thời gian đọc thì cảm biến DS18B20 sẽ bắt đầu truyền mức 1 hoặc mức 0 lên bus.

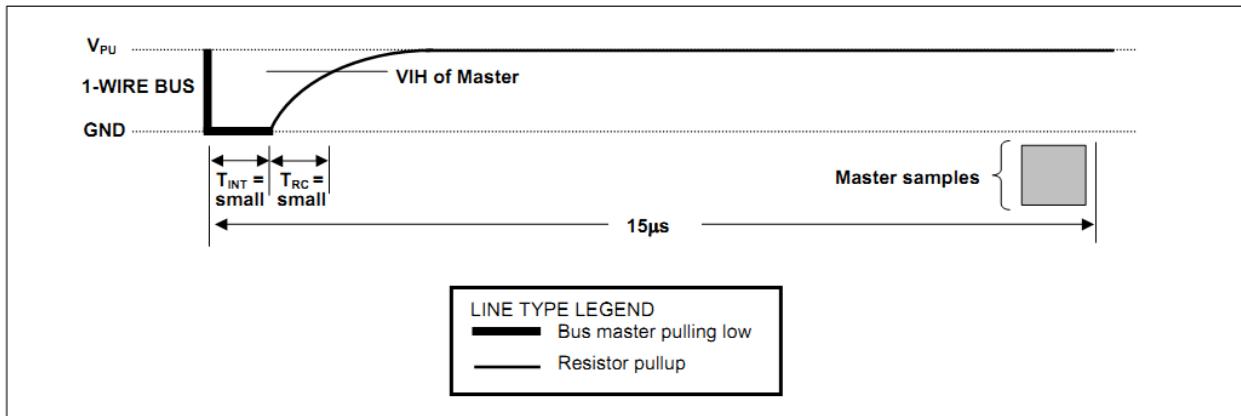
DS18B20 truyền mức 1 bằng cách để bus trở về mức 1 và truyền mức 0 thì kéo bus ở luôn mức 0. Khi đang truyền mức 0, DS18B20 sẽ thả nỗi bus khi gần kết thúc khe thời gian và bus sẽ được kéo lên mức 1 bởi điện trở kéo lên. Ngõ ra dữ liệu từ DS18B20 có hiệu lực trong vòng 15 μs tính từ khi có cạnh xuống của xung khởi tạo khe thời gian đọc. Do đó thiết bị chủ phải thả nỗi bus và sau đó tiến hành lấy mẫu trạng thái bus trong vòng 15 μs tính từ lúc bắt đầu khe thời gian.

Hình 7-26 minh họa tổng thời gian T_{INIT} , T_{RC} và T_{SAMPLE} phải nhỏ hơn $15\mu s$ đối với khe thời gian đọc.



Hình 7-26. Chi tiết dạng sóng khe thời gian đọc mức 1 của thiết bị chủ.

Hình 7-27 minh họa các giới hạn thời gian lớn nhất bằng cách giữ T_{INIT} và T_{RC} nhỏ nhất có thể và định vị thời gian lấy mẫu của thiết bị chủ trong khe thời gian đọc hướng về điểm gần kết thúc chu kỳ khe thời gian.



Hình 7-27. Thu ngắn các khoảng thời gian.

7.6 CÁC CHƯƠNG TRÌNH ĐO NHIỆT ĐỘ DÙNG CẢM BIẾN DS18B20

7.6.1 Mạch điện giao tiếp vi điều khiển với cảm biến DS18B20

Mạch điện giao tiếp vi điều khiển với cảm biến DS18B20 như hình 7-28.

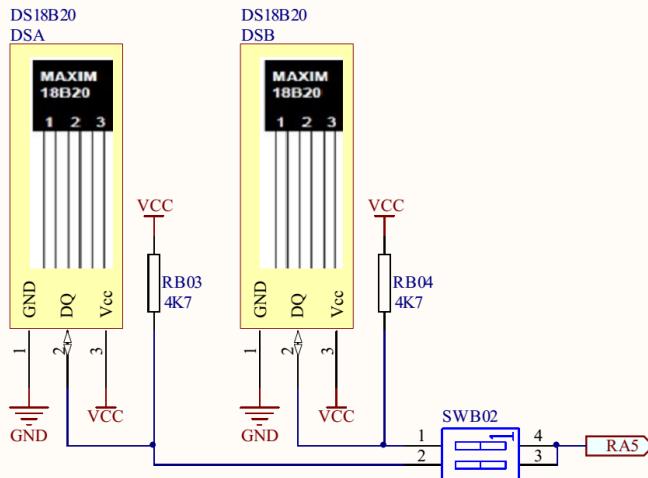
Trong sơ đồ mạch thì có 2 cảm biến: DS18B20A và DS18B20B nối với port RA5 thông qua switch SWB02.

Khi sử dụng thì cảm biến nào thì phải chuyển switch tương ứng sang ON.

Có các bài thực hành cho 2 cảm biến như sau:

Bài thực hành chỉ dùng 1 trong 2 cảm biến: Với bài dạng này thì không cần biết và không cần đọc code của bộ nhớ của cảm biến vì chỉ có 1 cảm biến. Khi dùng thì 1 switch ON, 1 switch còn lại OFF.

Bài thực hành dùng cả 2 cảm biến: Với bài dạng này thì cần phải biết code của bộ nhớ của 2 cảm biến, khi truy xuất cảm biến nào thì phải gửi địa chỉ của cảm biến đó mới truy xuất được. Khi dùng thì 2 switch đều ON. Dạng bài này phức tạp hơn dạng trên.



Hình 7-28. Mạch đo nhiệt độ dùng 2 cảm biến DS18B20.

7.6.2 Đo nhiệt độ dùng cảm biến DS18B20 hiển thị trên led 7 đoạn

Thư viện DS18B20. Chương trình thư viện của cảm biến DS18B20 đã cho trong thư mục của bạn có tên là “TV_PICKIT2_SHIFT_DS18B20.c”.

Bạn có thể sử dụng cho các bài thực hành và nên đọc hiểu.

```
#define touch_pin pin_a5
#include <touch.c>
#define skip_rom 0xcc
#define convert_t 0x44
#define read_scratchpad 0xbe
#define write_scratchpad 0x4e
#define copy_scratchpad 0x48
unsigned int8 ds18al, ds18ah, ds18a_tam;
unsigned int16 ds18a;
void khai_tao_ds18b20()
{
    touch_present();
    touch_write_byte(skip_rom);
    touch_write_byte(write_scratchpad);
    touch_write_byte(0x0); // ghi 2 byte rong
    touch_write_byte(0x0); // 
    touch_write_byte(0x1f); //cau hinh do phan giai 9 bit
    touch_present();
    touch_write_byte(skip_rom);
    touch_write_byte(copy_scratchpad);
}
void doc_giatri_ds18b20()
{
    touch_write_byte(skip_rom);
    touch_write_byte(convert_t);
    touch_present();
    touch_write_byte(skip_rom);
    touch_write_byte(read_scratchpad);
    ds18al = touch_read_byte();
}
```

0x7F - 12bit

muốn lấy phần thập phân - 9 bit (Tr. 215)
 $S \cdot 2^6 - 2^0 \cdot 2^{-1}$

10 bit - 0x3F / 217

```

ds18ah = touch_read_byte();
ds18a = make16(ds18ah, ds18al);
}

```

Bài mẫu 731. Chương trình đọc nhiệt độ từ cảm biến DS18B20A hiển thị trên 2 led 7 đoạn.

Lưu tên file là “BAI_731_DS18B20A”.

- Mục đích: biết lập trình đọc nhiệt độ từ cảm biến 1 dây và hiển thị trên led 7 đoạn.
- Lưu đồ: sinh viên hãy tự viết.
- Chương trình:

```
#include <tv_pickit2_shift_1.c>
#include <tv_pickit2_shift_ds18b20.c>
```

```

void main()
{
    set_up_port_ic_chot();
    khai_tao_ds18b20();
    ds18a_tam=0;
    xuat_4led_7doan_giaima_xoa_so0(0);
    while(true)
    {
        if(touch_present()) {doc_giatri_ds18b20();}

        if(ds18al!=ds18a_tam)
        {
            ds18a_tam = ds18al;
            ds18a = ds18a>>4;
            ds18a = ds18a & 0x0ff;
            xuat_4led_7doan_giaima_xoa_so0(ds18a);
            delay_ms(2000);
            do_trung_binh();
        }
    }
}
```

Cách 2: Bỏ 2 dòng lệnh và thêm delay như bên dưới

Cách 1:
Đưa 2 lệnh xử lý ngay khi đọc xong

Cách 3:
nd_1820 = ds18a;
và hiển thị giá trị theo biến nd_1820

xuat...so0(nd_1820);

- Tiến hành biên dịch và nạp.
- Quan sát kết quả: sẽ hiển thị nhiệt độ môi trường
- Giải thích chương trình:

hiển thị sai khi đưa dòng này ra ngoài if
do giá trị ds18a trước đó không phải là nhiệt độ.
khi hiển thị phải được xử lý

Bài mẫu 732. Chương trình đọc nhiệt độ từ cảm biến DS18B20A hiển thị trên 2 led 7 đoạn, khi nhiệt độ lớn hơn 33 thì 16 led phải sáng, 16 led trái tắt, khi nhiệt độ giảm dưới 31 thì 16 led phải tắt, 16 led trái sáng.

Lưu tên file là “BAI_732_DS18B20A_ON_OFF_32LED”.

- Mục đích: biết lập trình đọc nhiệt độ từ cảm biến 1 dây và hiển thị trên led 7 đoạn và biết so sánh điều khiển led khi quá nhiệt.
- Lưu đồ: sinh viên hãy tự viết.
- Chương trình:

```
#include <tv_pickit2_shift_1.c>
```

Thực hành vi điều khiển

```

#include <tv_pickit2_shift_ds18b20.c>
int1 ttqn;
#define     gh_ndtren      0x33
#define     gh_ndadoi      0x31

void main()
{
    set_up_port_ic_chot();
    khoi_tao_ds18b20();
    ds18a_tam=0;
    xuat_4led_7doan_giaima_xoa_so0(0);
    xuat_32led_don_1dw(0);
    ttqn=0;
    while(true)
    {
        if(touch_present()) {doc_giatri_ds18b20();}
        if(ds18a!=ds18a_tam)
        {
            ds18a_tam = ds18a;
            ds18a = ds18a>>4;
            ds18a = ds18a & 0x0ff;
            xuat_4led_7doan_giaima_xoa_so0(ds18a); //GM_LED_QUET(); HIEN...ALL();

            if ((ds18a>gh_ndtren)&&(ttqn==0))
            {
                ttqn = 1;
                xuat_32led_don_1dw(0x0000ffff);
            }
            else if ((ds18a<gh_ndadoi)&&(ttqn==1))
            {
                ttqn = 0;
                xuat_32led_don_1dw(0xffff0000);
            }
        }
    }
}
//GM_LED_QUET(); --> KHÔNG CẦN GM NỮA
HIEN...ALL();

```

GM_LED_QUET();
HIEN...ALL();

VOID GM_LED_QUET()
{
 LED_7DQ[0] = MA7DOAN[DS18A%10];
 ...
}

- d. Tiến hành biên dịch và nạp.
- e. Quan sát kết quả: Mở đèn gia tăng nhiệt khi quá thì sẽ mở 16 đèn led bên phải, khi giảm xuống thì mở 16 led bên trái.
- f. Giải thích chương trình:

Bài tập 733. Chương trình tự động đo và không chế nhiệt độ trong tầm từ 33 đến 35 độ. Khi nhiệt độ nhỏ hơn 33 thì mở đèn triac để gia tăng nhiệt độ. Khi nhiệt độ quá 35 thì tắt đèn triac và báo động bằng buzzer. Khi nhiệt độ nhỏ hơn 35 thì tắt buzzer.

Hiển thị nhiệt độ đo trên led 7 đoạn bên phải, khi tăng nhiệt thì hiển thị số 35 ở 2 led bên phải, khi giảm thì hiển thị số 33.

Lưu tên file là “BAI_733_DS18B20A_LCD_BUZZER”.

Bài tập 734. Chương trình đọc nhiệt độ từ cảm biến DS18B20A hiển thị trên 2 led 7 đoạn bên trái và đo nhiệt độ từ cảm biến LM35_A hiển thị trên 2 led bên phải để so sánh độ chính xác.

Lưu tên file là “BAI_734_DS18B20A_LM35A”.

Bài tập 735. Hệ thống vừa đọc nhiệt độ từ cảm biến DS18B20A hiển thị trên 2 led 7 đoạn bên phải vừa đếm sản phẩm dùng T0 hiển thị trên led 7 đoạn bên trái đếm từ 00 đến 99.

Lưu tên file là “BAI_735_DS18B20A_DSP_T1”.

Bài mẫu 736. Chương trình đọc nhiệt độ từ cảm biến hiển thị trên 2 led 7 đoạn, 2 led 7 đoạn còn lại hiển thị giá trị cài đặt trước để báo động, giá trị mặc nhiên thấp nhất là 35 độ, lớn nhất là 60 độ, có thể điều chỉnh bằng 2 nút nhấn UP và DW. Khi giá trị đo lớn hơn giá trị cài thì báo động ngược lại thì tắt báo động. Tự động tắt mở đèn.

Lưu tên file là “BAI_736_DS18B20A_UP_DW_BUZZ”.

thay bĂng ma trÂn phím
BÀi 625/147

GhÉp bÀi 714/208

Chương 8: THỰC HÀNH

MODULE 5 – REAL TIME DS13B07, ADC–DAC PCF8591, EEPROM NỐI TIẾP AT24C256 THEO CHUẨN I²C VÀ CÁC CẢM BIẾN

8.1 LÝ THUYẾT I2C

8.1.1 Giới thiệu

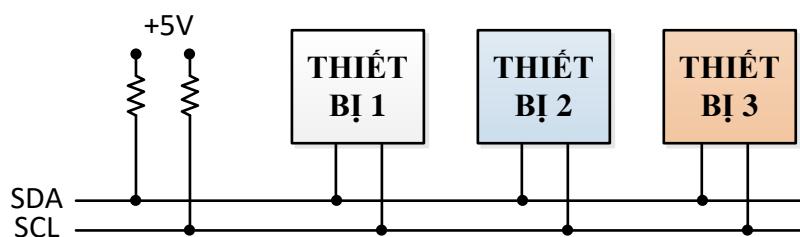
Vi điều khiển PIC18F4550 tích hợp chuẩn truyền dữ liệu I2C để giao tiếp với các thiết bị ngoại vi theo chuẩn I2C.

Có rất nhiều ngoại vi tích chuẩn giao tiếp I2C như bộ nhớ Eeprom nối tiếp, ADC-DAC, Real-time, IC mở rộng ngoại vi, ...

Trong phần này trình bày chuẩn giao tiếp I2C và lập trình cho vi điều khiển PIC18F4550 thực hiện giao tiếp với đồng hồ thời gian thực theo chuẩn I2C.

8.1.2 Tổng quan về truyền dữ liệu chuẩn I2C

I2C, viết tắt của từ Inter-Integrated Circuit, là một chuẩn truyền thông do hãng điện tử Philips Semiconductor sáng lập, cho phép giao tiếp một thiết bị chủ với nhiều thiết bị tớ với nhau như hình 8-1.



Hình 8-1. Hệ thống các thiết bị giao tiếp theo chuẩn I2C.

Chuẩn giao tiếp I2C có 2 đường tín hiệu tên là SDA (serial data) có chức năng truyền tải dữ liệu và tín hiệu SCL (serial clock) truyền tải xung clock để dịch chuyển dữ liệu.

Trong hệ thống truyền dữ liệu I2C, thiết bị nào cung cấp xung clock thì được gọi là chủ (master), thiết bị nhận xung clock được gọi là tớ (slave).

Thiết bị chủ chỉ có 1, thiết bị tớ thì có nhiều, mỗi thiết bị tớ sẽ có 1 địa chỉ độc lập, chuẩn truyền ban đầu dùng địa chỉ 7 bit nên có thể 1 chủ giao tiếp với 128 thiết bị tớ. Các thiết bị sau này tăng thêm số bit địa chỉ nên có thể giao tiếp nhiều hơn.

Địa chỉ của thiết bị tớ thường do nhà chế tạo thiết bị thiết lập sẵn.

8.1.3 Quy trình truyền dữ liệu chuẩn I2C

Quy trình thiết bị chủ giao tiếp với thiết bị tớ để thực hiện việc ghi đọc dữ liệu như sau:

Quá trình thiết bị chủ ghi dữ liệu vào thiết bị tớ

Bước 1: Thiết bị chủ tạo trạng thái START để bắt đầu quá trình truyền dữ liệu – các thiết bị tớ sẽ ở trạng thái sẵn sàng nhận địa chỉ từ thiết bị chủ.

Bước 2: Thiết bị chủ gửi địa chỉ của thiết bị tớ cần giao tiếp – khi đó tất cả các thiết bị tớ đều nhận địa chỉ và so sánh với địa chỉ của mình, các thiết bị tớ sau khi phát hiện không phải địa chỉ của mình thì chờ cho đến khi nào nhận trạng thái START mới.

Trong dữ liệu 8 bit thì có 7 bit địa chỉ và 1 bit điều khiển đọc/ghi (R/W): bit này bằng 0 để báo cho thiết bị tớ sẽ nhận byte tiếp theo.

Bước 3: Thiết bị chủ chờ nhận tín hiệu bắt tay từ thiết bị tớ. Thiết bị tớ nào đúng địa chỉ thì phát 1 tín hiệu trả lời cho chủ biết.

Bước 4: Thiết bị chủ tiến hành gởi địa chỉ của ô nhớ bắt đầu cần ghi dữ liệu, bit R/W ở trạng thái ghi.

Bước 5: Thiết bị chủ chờ nhận tín hiệu trả lời từ thiết bị tớ.

Bước 6: Thiết bị chủ gởi tiến hành gởi dữ liệu để ghi vào thiết bị tớ, mỗi lần ghi 1 byte, sau khi gởi xong thì tiến hành chờ nhận tín hiệu trả lời từ thiết bị tớ, quá trình thực hiện cho đến byte cuối cùng xong rồi thì thiết bị chủ chuyển sang trạng thái STOP để chấm dứt quá trình giao tiếp với thiết bị tớ.

Quá trình thiết bị chủ đọc dữ liệu vào thiết bị tớ

Bước 1: Thiết bị chủ tạo trạng thái START để bắt đầu quá trình truyền dữ liệu, các thiết bị tớ sẽ ở trạng thái sẵn sàng nhận địa chỉ từ thiết bị chủ.

Bước 2: Thiết bị chủ gởi địa chỉ của thiết bị tớ cần giao tiếp, khi đó tất cả các thiết bị tớ đều nhận địa chỉ và so sánh với địa chỉ của mình. Các thiết bị tớ sau khi phát hiện không phải địa chỉ của mình thì chờ cho đến khi nào nhận trạng thái START mới.

Trong dữ liệu 8 bit thì có 7 bit địa chỉ và 1 bit điều khiển đọc/ghi (R/W): bit này bằng 0 để báo cho thiết bị tớ sẽ nhận byte tiếp theo.

Bước 3: Thiết bị chủ chờ nhận tín hiệu bắt tay từ thiết bị tớ. Thiết bị tớ nào đúng địa chỉ thì phát 1 tín hiệu trả lời cho chủ biết.

Bước 4: Thiết bị chủ tiến hành gởi địa chỉ của ô nhớ bắt đầu cần đọc dữ liệu, bit R/W ở trạng thái đọc.

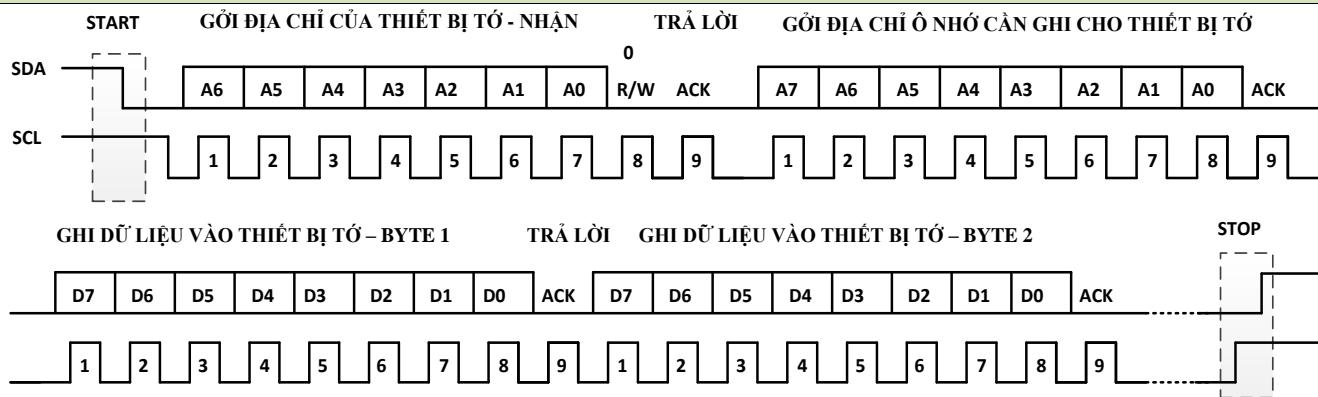
Bước 5: Thiết bị chủ chờ nhận tín hiệu trả lời từ thiết bị tớ.

Bước 6: Thiết bị chủ chuyển sang trạng thái STOP, bắt đầu lại trạng thái START, tiến hành gởi địa chỉ của thiết bị và bit R/W bằng 1 để yêu cầu tớ gởi dữ liệu nội dung ô nhớ của địa chỉ đã nhận.

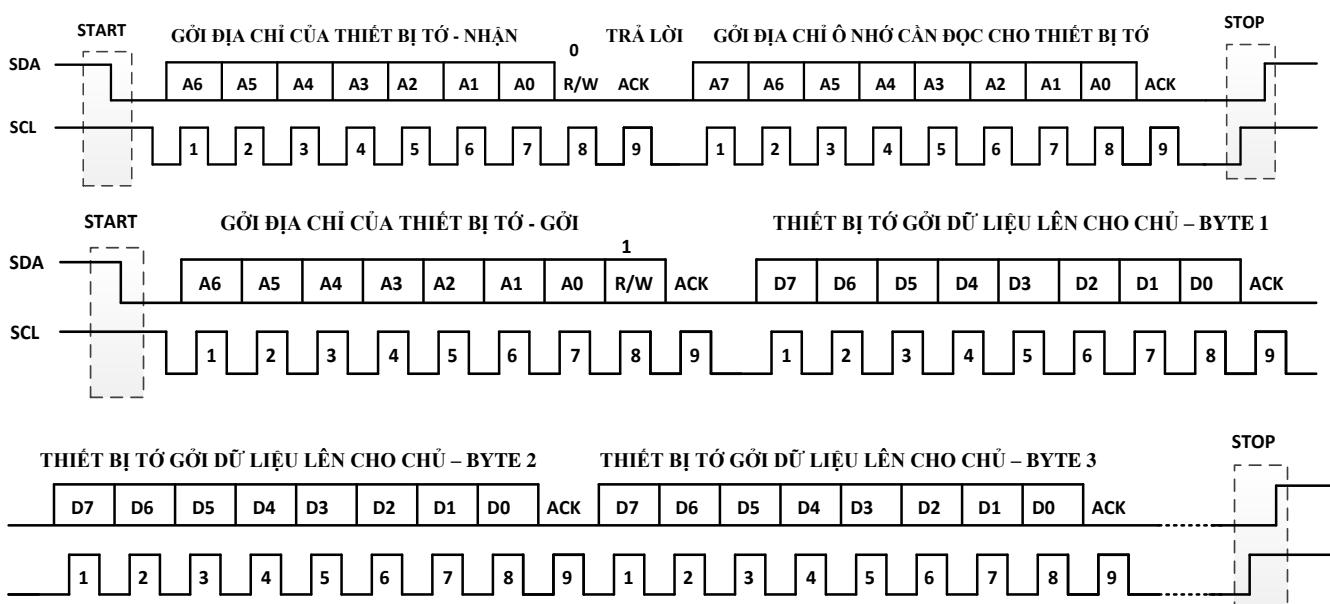
Bước 7: Thiết bị chủ sau khi nhận sẽ báo tín hiệu trả lời, quá trình này thực hiện cho đến khi nhận hết dữ liệu mong muốn thì thiết bị chủ tạo tín hiệu STOP để chấm dứt.

8.1.4 Dạng sóng truyền dữ liệu chuẩn I2C

Dạng sóng truyền dữ liệu I2C của quá trình ghi dữ liệu từ chủ lên tớ như hình 8-2.

**Hình 8-2. Quá trình chủ ghi dữ liệu vào tóm tắt.**

Dạng sóng truyền dữ liệu I2C của quá trình đọc dữ liệu từ tóm tắt như hình 8-3.

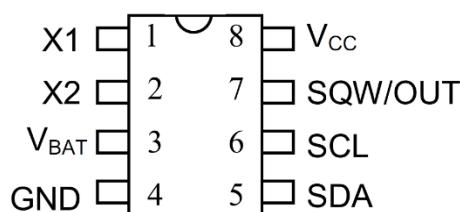
**Hình 8-3. Quá trình chủ đọc dữ liệu từ tóm tắt.**

8.2 KHẢO SÁT IC REALTIME DS13B07

8.2.1 Khảo sát IC DS1307

DS1307 là chip đồng hồ thời gian thực của Dallas Semiconductor. Chip có 64 ô nhớ, trong đó có 8 ô nhớ 8-bit lưu thời gian: giây, phút, giờ, thứ (trong tuần), ngày, tháng, năm và thanh ghi điều khiển ngõ ra, vùng nhớ còn lại là 56 ô dùng có thể dùng để lưu dữ liệu.

DS1307 giao tiếp theo chuẩn nối tiếp I2C nên sơ đồ chân bên ngoài chỉ có 8 chân.

**Hình 8-4. Sơ đồ chân DS1307.**

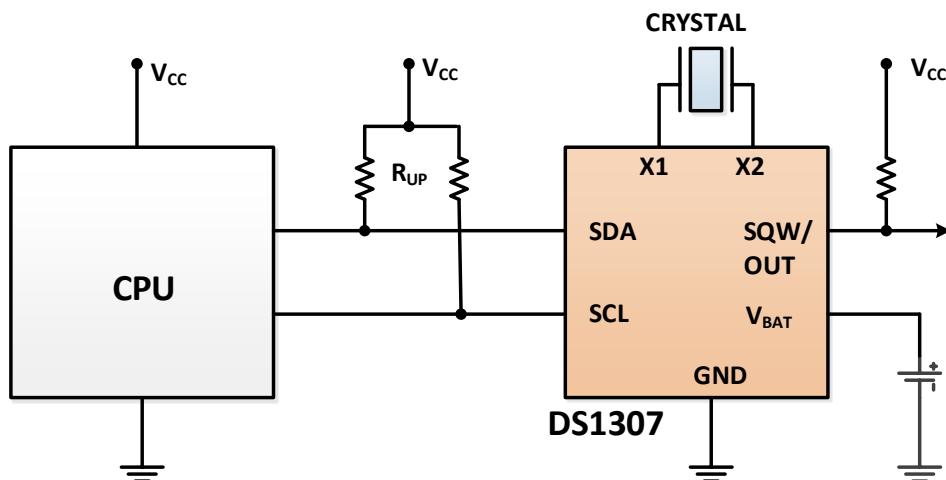
Các chân của DS1307:

- Hai chân **X1** và **X2**: nối với tụ thạch anh có tần số 32768Hz để tạo dao động.
- Chân **V_{BAT}** nối với nguồn dương của pin 3V3 để chip vẫn hoạt động đếm thời gian khi mất nguồn.
- Chân **GND** là chân nối 0V.
- Chân **V_{CC}** nối với nguồn Vcc có thể là 5V hoặc 3V3 tùy thuộc vào từng loại chíp.
- Chân **SQW/OUT** là ngõ ra tạo xung vuông (Square Wave / Output Driver), tần số có thể lập trình, tích cực mức 0, chế độ nhận dòng.
- Hai chân **SCL, SDA** là 2 đường tín hiệu giao tiếp chuẩn I2C.

Tổ chức bộ nhớ bên trong của RT DS1307 như hình 8-6.

Vì 7 thanh ghi đầu tiên là quan trọng nhất trong hoạt động của DS1307 nên ta quan sát chi tiết tổ chức theo từng bit của các thanh ghi này như trong hình 8-7.

Ô nhớ lưu giây (SECONDS): có địa chỉ là 0x00, có chức năng lưu hàng chục giây và hàng đơn vị giây. Bit thứ 7 có tên CH (Clock halt – ngừng đồng hồ), nếu bit này bằng 1 thì bộ dao động trong chip ngừng dao động làm đồng hồ ngừng hoạt động. Nếu muốn đồng hồ hoạt động thì bit này phải bằng 0.



Hình 8-5. Sơ đồ kết nối vi điều khiển với DS1307

ADDRESS



Hình 8-6. Tổ chức bộ nhớ của DS1307

							BIT0
00H	CH	10 SECONDS		SECONDS			00 - 59
01H	0	10 MINUTES		MINUTES			00 - 59
02H	0	12 24	10HR A/P	10 HR	HOURS		01 - 12 00 - 23
03H	0	0	0	0	0	DAY	1 - 7
04H	0	0	10 DATE		DATE		1 - 28, 29, 30, 31
05H	0	0	10 MONTH		MONTH		01 - 12
06H	10 YEAR			YEAR			00 - 99
07H	OUT	0	0	SQWE	0	0	RS1 RS0

Hình 8-7. Tổ chức các thanh ghi thời gian

Ô nhớ lưu phút (MINUTES): có địa chỉ 0x01, có chức năng lưu phút hàng đơn vị và hàng chục, bit 7 luôn bằng 0.

Ô nhớ lưu giờ (HOURS): có địa chỉ 0x02 có chức năng lưu hàng chục giờ và hàng đơn vị giờ ở 2 chế độ 12 giờ và 24 giờ được lựa chọn bởi bit thứ 6 có tên là 12/24.

Nếu bit 12/24 chọn chế độ 24 giờ thì phần hàng chục giờ sử dụng 2 bit thứ 4 và thứ 5 có kí hiệu là 10HR.

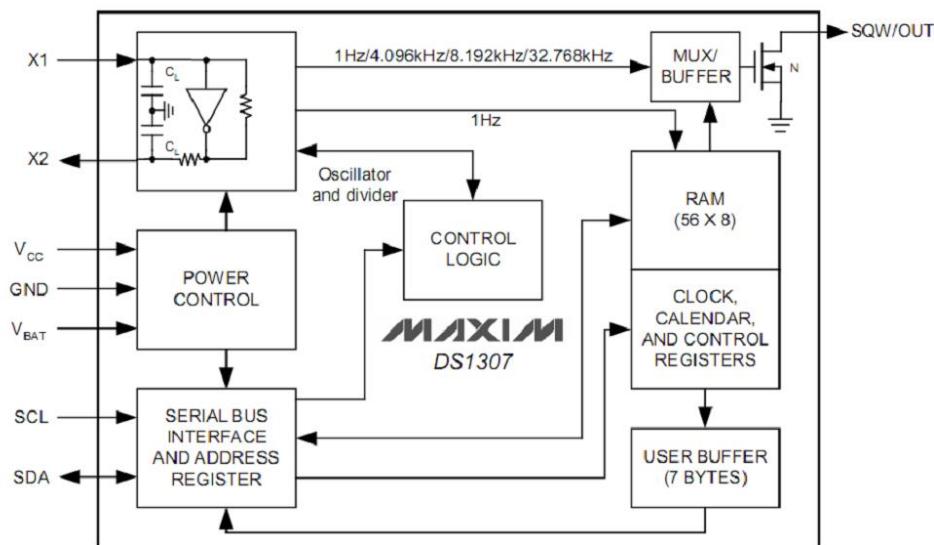
Nếu bit 12/24 chọn chế độ 12 giờ thì phần hàng chục giờ sử dụng bit thứ 4, còn bit thứ 5 sẽ có kí hiệu là A/P tương ứng với 2 chế độ giờ AM và PM.

Bit 7 luôn bằng 0.

Ô nhớ lưu thứ (DAY – ngày trong tuần): có địa chỉ 0x03, có chức năng lưu thứ trong tuần có giá trị từ 1 đến 7 tương ứng từ Chủ nhật đến thứ bảy trong tuần, chỉ sử dụng 3 bit thấp.

Các ô còn lại là ngày tháng năm.

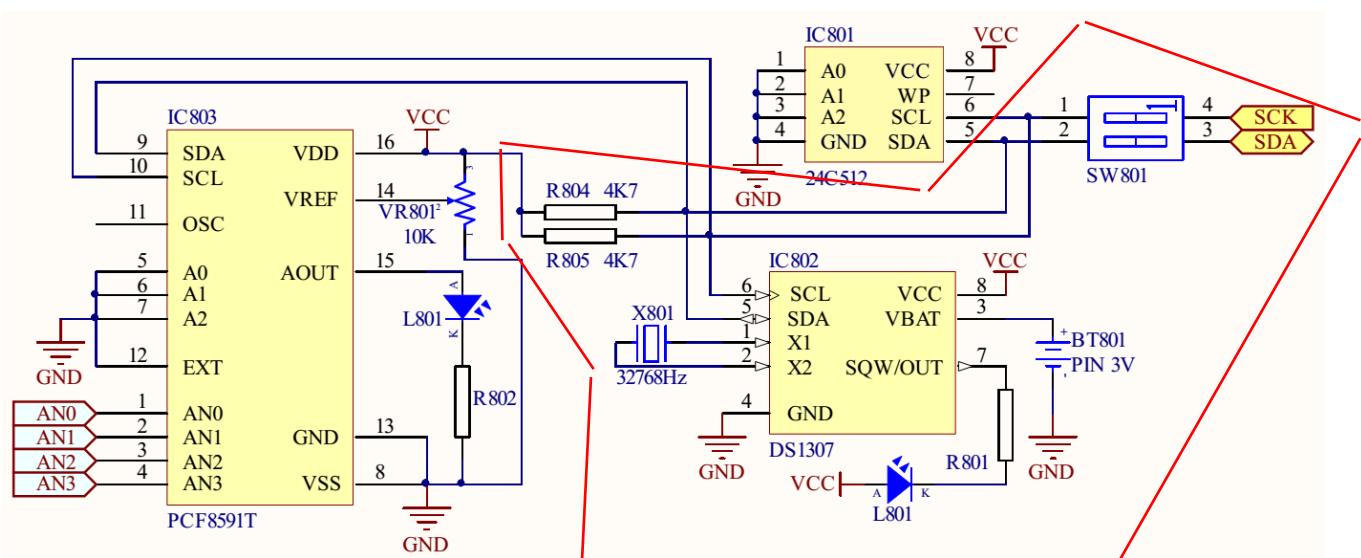
Cấu trúc bên trong của DS1307 như hình 8-8.

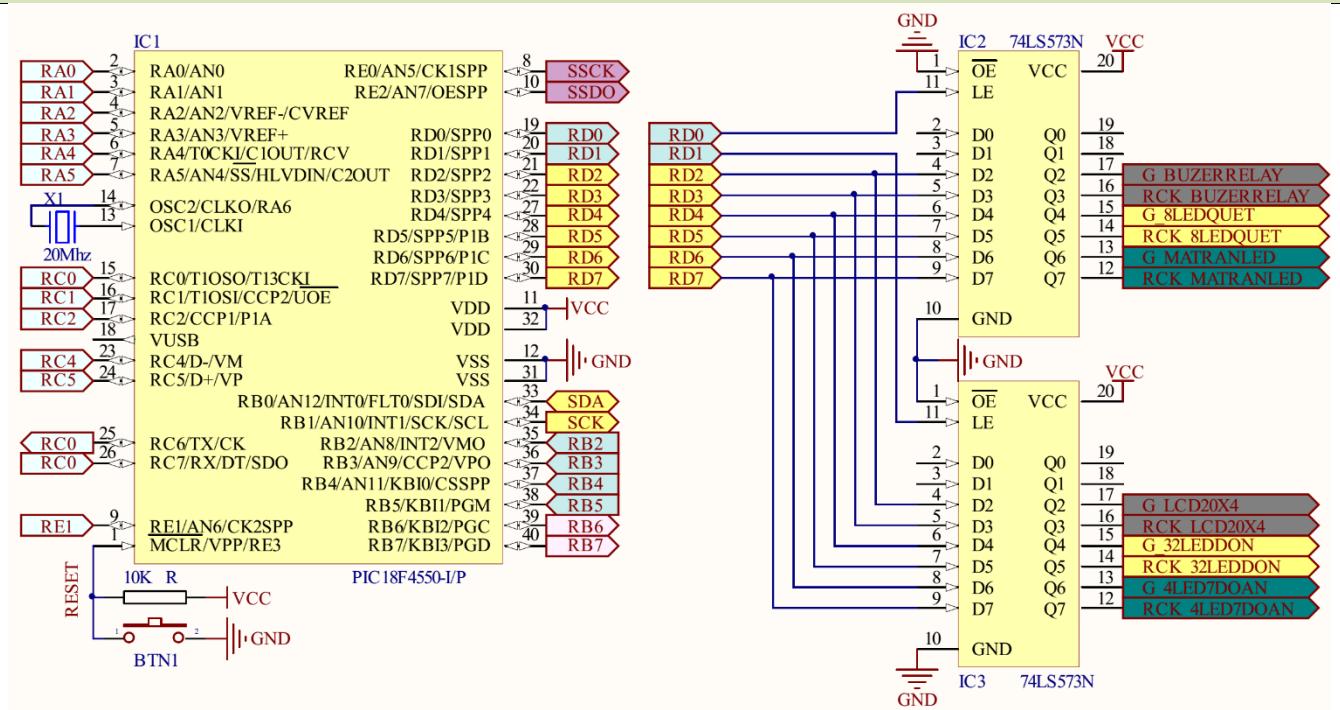


Hình 8-8. Cấu trúc bên trong DS1307

8.2.2 Sơ đồ giao tiếp vi điều khiển với DS1307

Trong phần này sẽ trình bày phần giao tiếp vi điều khiển PIC18F4550 với I2C như hình 8-9:





Hình 8-9. Giao tiếp vi điều khiển các thiết bị theo chuẩn I2C

Khỏi dùng chuẩn I2C giao tiếp với vi điều khiển thông qua 2 chân: RB0/SDA và RB1/SCL, khi sử dụng thì chuyển 2 switch con của SW23 sang vị trí ON.

8.2.3 Các chương trình giao tiếp DS13B07

Thư viện I2C của DS1307. Chương trình thư viện của IC thời gian thực đã cho trong thư mục của bạn có tên là “TV_PICKIT2_SHIFT_DS1307_I2C.c”.

Bạn có thể sử dụng cho các bài thực hành và nên đọc hiểu.

```
#include <tv_pickit2_shift_lcd.c>
#define giay_htai 0x55
#define phut_htai 0x33
#define gio_htai 0x08
#define thu_htai 8
#define ngay_htai 0x16
#define thang_htai 0x06
#define nam_htai 0x13
#define ma_ds 0x98
#define addr_wr_1307 0xd0
#define addr_rd_1307 0xd1
#define addr_mem 0x00

unsigned char nam_ds13, thang_ds13, ngay_ds13, thu_ds13, gio_ds13,
            phut_ds13, giay_ds13, ma_ds13, control_ds13, giaytam;

void thiet_lap_thoi_gian_hien_tai()
{
```

thay đổi mỗi khi thiết
lập thời gian
giá trị khác lần trước
0x90, 0x91, 0x92

```

giay_ds13      =  giay_htai;
phut_ds13      =  phut_htai;
gio_ds13       =  gio_htai;
thu_ds13       =  thu_htai;
ngay_ds13      =  ngay_htai;
thang_ds13     =  thang_htai;
nam_ds13       =  nam_htai;
control_ds13   =  0x90;
ma_ds13        =  ma_ds;

}

void nap_thoi_gian_htai_vao_ds1307()
{
    i2c_start();
    i2c_write(addr_wr_1307);
    i2c_write(0x00);

    i2c_write(giay_ds13);
    i2c_write(phut_ds13);
    i2c_write(gio_ds13);
    i2c_write(thu_ds13);
    i2c_write(ngay_ds13);
    i2c_write(thang_ds13);
    i2c_write(nam_ds13);
    i2c_write(control_ds13);
    i2c_write(ma_ds13);
    i2c_stop();
}

void doc_thoi_gian_tu_realtim()
{
    i2c_start();
    i2c_write(addr_wr_1307);
    i2c_write(addr_mem);
    i2c_start();

    i2c_write(addr_rd_1307);
    giay_ds13      =  i2c_read();
    phut_ds13      =  i2c_read();
    gio_ds13       =  i2c_read();
    thu_ds13       =  i2c_read();
    ngay_ds13      =  i2c_read();
    thang_ds13     =  i2c_read();
    nam_ds13       =  i2c_read();
    control_ds13   =  i2c_read();
    ma_ds13        =  i2c_read(0);      //not ack
    i2c_stop();
}

void doc_giay_tu_realtim()
{
    i2c_start();
    i2c_write(addr_wr_1307);
    i2c_write(addr_mem);
}

```

```

i2c_start();

i2c_write(addr_rd_1307);
giay_ds13      = i2c_read(0);
i2c_stop();
}

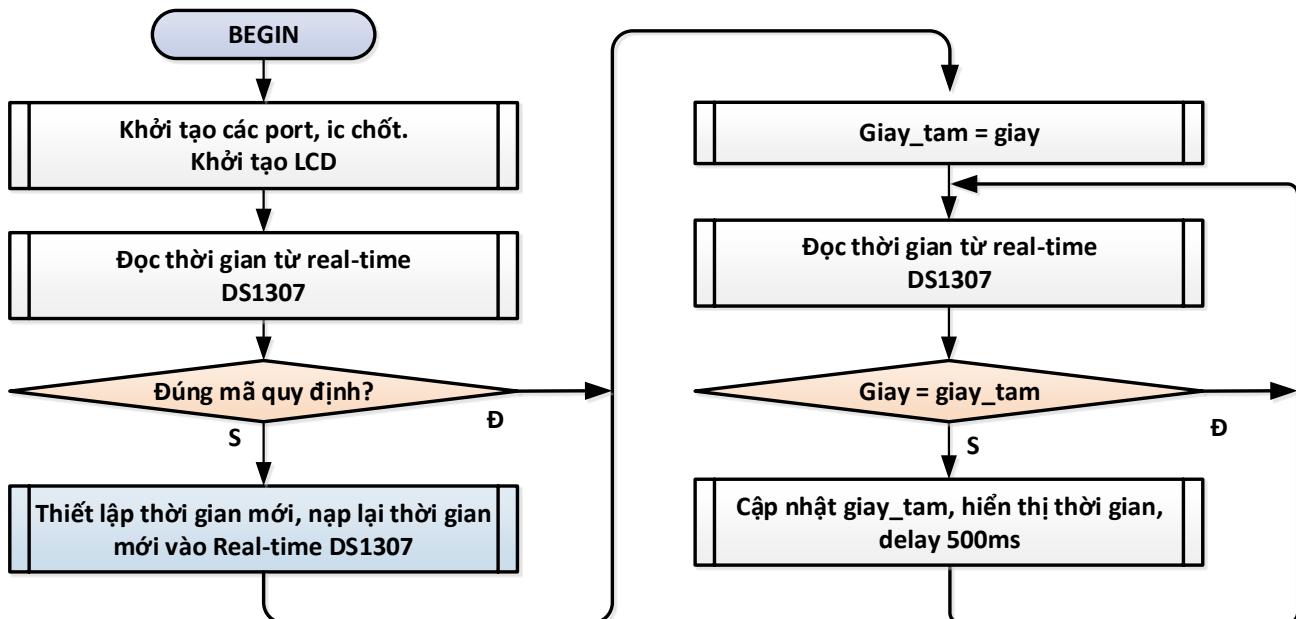
void hien_thi_thoi_gian_ds1307 ( )
{
    lcd_command (0x8c);
    lcd_data(gio_ds13/16 +0x30);
    lcd_data(gio_ds13%16 +0x30);
    lcd_data(' ');
    lcd_data(phut_ds13/16 +0x30);
    lcd_data(phut_ds13%16 +0x30);
    lcd_data(' ');
    lcd_data(giay_ds13/16 +0x30);
    lcd_data(giay_ds13%16 +0x30);
    lcd_command (0xcc);
    lcd_data/ngay_ds13/16 +0x30);
    lcd_data/ngay_ds13%16 +0x30);
    lcd_data(' ');
    lcd_data(thang_ds13/16 +0x30);
    lcd_data(thang_ds13%16 +0x30);
    lcd_data(' ');
    lcd_data/nam_ds13/16 +0x30);
    lcd_data/nam_ds13%16 +0x30);
}

```

Bài mẫu 801. Chương trình đọc thời gian thực hiển thị trên LCD 20x4.

Lưu tên file là “BAI_801_I2C_RT_LCD ”

- Mục đích: biết viết chương trình đọc thời gian thực hiển thị trên LCD.
- Lưu đồ:



Hình 8-10. Lưu đồ đồng hồ số dùng DS1307.

c. Chương trình:

```

#include <tv_pickit2_shift_1.c>
#include <tv_pickit2_shift_ds1307_i2c.c>
void main()
{
    set_up_port_ic_chot();
    setup_lcd();
    lcd_command(lcd_addr_line1);
    lcd_data("clock:");
    doc_thoi_gian_tu_realtim();
    if (ma_ds13!=ma_ds)
    {
        thiet_lap_thoi_gian_hien_tai();
        nap_thoi_gian_htai_vao_ds1307();
    }
    //thiet_lap_thoi_gian_hien_tai();
    //nap_thoi_gian_htai_vao_ds1307();
    doc_thoi_gian_tu_realtim();
    giaytam=giay_ds13;
    while(true)
    {
        doc_thoi_gian_tu_realtim();
        if (giaytam!=giay_ds13)
        {
            giaytam=giay_ds13;
            hien_thi_thoi_gian_ds1307();
        }
        delay_ms(500);
    }
}

```

phần mềm CCS
5.015 trở lên thì bị lỗi
sau khi nạp chương
trình thì phải tắt
nguồn bộ thí nghiệm
rồi mở lại --> ct chạy
(bỏ pin ra)

có thể dùng kết hợp
led quét tránh trường
hợp hiển thị LCD
nhiều lần --> ảnh
hưởng đến led quét

giảm thời gian cập
nhật thông tin
--> led quét
--> bỏ đi

d. Tiến hành biên dịch và nạp.

- e. Quan sát kết quả: nếu kết quả không đúng yêu cầu thì kiểm tra lại chương trình.
- f. **Giải thích chương trình:** Chương trình chính có chức năng đọc thời gian thực từ IC DS1307 rồi so sánh xem giây có thay đổi hay không bằng cách so sánh với giây tạm, nếu khác thì tiến hành giải mã hiển thị trên LCD, cập nhật giây tạm mới. Sau đó delay 500ms mới lập lại chu kỳ tiếp theo.

Trong quá trình khởi tạo các thông số, có thực hiện chức năng: kiểm tra IC DS1307 xem đã khởi tạo lần nào chưa bằng cách kiểm tra mã ta tự thiết lập trong ô nhớ của IC. Nếu chưa thì chương trình sẽ khởi tạo các thông số thời gian hiện tại, sau khi khởi tạo xong thì IC real-time sẽ đếm từ thời gian hiện tại, bạn có thể hiệu chỉnh lại thời gian hiện tại cho phù hợp.

Bài mẫu 802. Chương trình đọc thời gian thực hiển thị trên LCD 20x4, có 3 nút chỉnh thời gian giờ, phút, giây là UP, DW và MOD.

Lưu tên file là “BAI_802_I2C_RT_LCD_3KEY”

- Mục đích: biết viết thêm phần chỉnh thời gian thực gồm giờ phút giây.
- Lưu đồ:
- Chương trình:

```
#include <tv_pickit2_shift_1.c>
#include <tv_pickit2_shift_lcd.c>
#include <tv_pickit2_shift_i2c.c> #include <tv_pickit2_shift_ds1307_i2c.c>
unsigned int16 k;
unsigned char giatri_mod;

void hien_thi_gia_tri_mod()
{
    lcd_command(0x8a);
    lcd_data(giatri_mod+0x30);
}

void phim_mod()
{
    if (!input(mod))
    {
        delay_ms(20);
        if (!input(mod))
        {
            if(giatri_mod<3) giatri_mod++;
            else giatri_mod=0;
            hien_thi_gia_tri_mod();
            while(!input(mod));
        }
    }
}

unsigned int8 hieu_chinh_so_bcd_tang(unsigned int8 x )
```

```

{
    unsigned int8 y;
    y = x;
    x = x & 0x0f;
    if (x==0xa)    y = y + 6;
    return(y);
}

unsigned int8 hieu_chinh_so_bcd_giam(unsigned int8 x )
{
    unsigned int8 y;
    y = x;
    x = x & 0x0f;
    if (x==0xf)    y = y - 6;
    return(y);
}

void luu_giai_ma_hien_thi_sau_khi_chinh()
{
    nap_thoi_gian_htai_vao_ds1307();
    hien_thi_thoi_gian_ds1307( );
}

void phim_up()
{
    if (!input(up))
    {
        delay_ms(20);
        if (!input(up))
        {
            switch (giatri_mod)
            {
                case 1: if (giay_ds13==0x59) giay_ds13=0;
                          else
                          {
                              giay_ds13++;
                              giay_ds13=hieu_chinh_so_bcd_tang(giay_ds13);
                          }
                luu_giai_ma_hien_thi_sau_khi_chinh();
                break;

                case 2: if (phut_ds13==0x59) phut_ds13=0;
                          else
                          {
                              phut_ds13++;
                              phut_ds13=hieu_chinh_so_bcd_tang(phut_ds13);
                          }
                luu_giai_ma_hien_thi_sau_khi_chinh();
                break;

                case 3: if (gio_ds13==0x23) gio_ds13=0;
                          else
                          {
                              gio_ds13++;
                          }
            }
        }
    }
}

```

```

        gio_ds13=hieu_chinh_so_bcd_tang(gio_ds13);
    }
    luu_giai_ma_hien_thi_sau_khi_chinh();
    break;

    default: break;
}
while(!input(up));
}

void phim_dw()
{
    if (!input(dw))
    {
        delay_ms(20);
        if (!input(dw))
        {
            switch (giatri_mod)
            {
                case 1: if (giay_ds13==0) giay_ds13=0x59;
                else
                {
                    giay_ds13--;
                    giay_ds13=hieu_chinh_so_bcd_giam(giay_ds13);
                }
                luu_giai_ma_hien_thi_sau_khi_chinh();
                break;

                case 2: if (phut_ds13==0) phut_ds13=0x59;
                else
                {
                    phut_ds13--;
                    phut_ds13=hieu_chinh_so_bcd_giam(phut_ds13);
                }
                luu_giai_ma_hien_thi_sau_khi_chinh();
                break;

                case 3: if (gio_ds13==0) gio_ds13=0x23;
                else
                {
                    gio_ds13--;
                    gio_ds13=hieu_chinh_so_bcd_giam(gio_ds13);
                }
                luu_giai_ma_hien_thi_sau_khi_chinh();
                break;

                default: break;
            }
            while(!input(dw));
        }
    }
}

```

```

void main()
{
    set_up_port_ic_chot();
    setup_lcd();
    lcd_command(lcd_addr_line1);
    lcd_data("clock:");
    doc_thoi_gian_tu_realtime();
    if (ma_ds13!=ma_ds)
    {
        thiet_lap_thoi_gian_hien_tai();
        nap_thoi_gian_htai_vao_ds1307();
    }
    giatri_mod=0;
    hien_thi_gia_tri_mod();

    doc_thoi_gian_tu_realtime();
    giaytam=giay_ds13;

    while(true)
    {
        doc_thoi_gian_tu_realtime();
        if (giaytam!=giay_ds13)
        {
            giaytam=giay_ds13;
            hien_thi_gia_tri_mod();
            hien_thi_thoi_gian_ds1307();
        }
        for(k=0;k<10000;k++)
        {
            phim_mod();
            phim_up();
            phim_dw();
        }
    }
}

```

nên bỏ luôn for khi có thêm yêu cầu đọc counter, đo nhiệt độ để đáp ứng nhanh, không cần phải chờ kt phím mất thời gian

quét phím nhiều lần
--> tạo delay
--> delay có k.tرا phím
+ quét led 7 đoạn

thêm quét led 7 đoạn --> đảm bảo led không tắt

- d. Tiến hành biên dịch và nạp.
- e. Quan sát kết quả: nếu kết quả không đúng yêu cầu thì kiểm tra lại chương trình.
- f. Giải thích chương trình: Trong chương trình có biến GIATRI_MOD ban đầu gán bằng 0 thì không cho phép chỉnh thời gian. Khi nhấn phím MOD thì thay đổi giá trị của biến GIATRI_MOD lên 1 giá trị, khi vượt quá 3 thì cho về 0. Khi giá trị GIATRI_MOD bằng 1 thì cho phép chỉnh giây, khi bằng 2 thì cho phép chỉnh phút, khi bằng 3 thì cho phép chỉnh giờ.

Khi nhấn nút UP thì dựa vào giá trị của biến GIATRI_MOD mà tăng đổi tương ứng.

Khi nhấn nút DW thì dựa vào giá trị của biến GIATRI_MOD mà giảm đổi tương ứng.

Do các giá trị thời gian là số BCD, khi thực hiện lệnh tăng thì giá trị tăng theo số hex, ví dụ khi giờ bằng 0x09 nếu tăng sẽ thành 0x0A – giá trị này không phù hợp với chế

độ hoạt động mà ta đã chọn cho real-time nên phải hiệu chỉnh số hex thành số BCD tương ứng là 0x10. Các hàm có chức năng kiểm tra và gán trị tương ứng rồi trả kết quả đúng trả về.

Bài 803 với yêu cầu đơn giản giúp bạn dễ dàng tiếp cận và hiểu chương trình. Để hiệu chỉnh đầy đủ thì tiếp tục thực hiện bài 803.

Bài mẫu 803. Thêm vào chương trình của bài trên phần hiệu chỉnh **ngày, tháng và năm**.

Lưu tên file là “BAI_803_I2C_RT_LCD_3KEY”

- Mục đích: biết viết thêm phần chỉnh thời gian thực gồm ngày tháng năm.
- Lưu đồ: sinh viên hãy tự viết.

```
#include <tv_pickit2_shift_ds1307_i2c.c>
#include <tv_pickit2_shift_1.c>
#include <tv_pickit2_shift_lcd.c>
#include <tv_pickit2_shift_i2c.c>

unsigned int16 k;
unsigned char giatri_mod;

void hien_thi_gia_tri_mod ()
{
    lcd_command(0x8a);
    lcd_data(giatri_mod+0x30);
}

void phim_mod()
{
    if (!input(mod))
    {
        delay_ms(20);
        if (!input(mod))
        {
            if(giatri_mod<6) giatri_mod++;
            else giatri_mod=0;
            hien_thi_gia_tri_mod ();
            while(!input(mod));
        }
    }
}

unsigned int8 hieu_chinh_so_bcd_tang(unsigned int8 x )
{
    unsigned int8 y;
    y = x;
    x = x & 0x0f;
    if (x==0xa) y = y + 6;
    return(y);
}
```

THÚ

2-7-1 (CN) - Hình 8.7/236

': KHOẢNG TRẮNG
HT THÚ'

BÀI GIÁNG, COPY HÀM HIỂN THỊ
TRONG THƯ VIỆN, ĐƯA VÀO
CHƯƠNG TRÌNH CỦA MÌNH VÀ ĐỔI
TÊN MỚI HT_DS1307

NHẤP NHÁY KHI CHỈNH

614/Tr.143

BT: CHỈ HIỂN THỊ

PHUT - GIÂY (SỐ LỚN)
THÚ - NGÀY (SỐ LỚN)

(HÀM ĐỌC KHÔNG THAY ĐỔI, chỉ
thay đổi hàm hiển thị)

NHẤP NHÁY SỐ LỚN - TẠO KÝ TỰ
TỪ CÁC ĐOẠN ĐÃ CÓ,
THÊM KÝ TỰ KHOẢNG TRẮNG LỚN

32,32,32, 32,32,32
0x20,0x20...

Bài xóa số 0 vô nghĩa số lớn: 644/157

```

unsigned int8 hieu_chinh_so_bcd_giam(unsigned int8 x )
{
    unsigned int8 y;
    y = x;
    x = x & 0x0f;
    if (x==0x0f)    y = y - 6;
    return(y);
}

void luu_giai_ma_hien_thi_sau_khi_chinh()
{
    nap_thoi_gian_htai_vao_ds1307();
    hien_thi_thoi_gian_ds1307( );
}

void phim_up()
{
    if (!input(up))
    {
        delay_ms(20);
        if (!input(up))
        {
            switch (giatri_mod)
            {
                case 1: if (giay_ds13==0x59) giay_ds13=0;
                           else
                           {
                               giay_ds13++;
                               phut_ds13=hieu_chinh_so_bcd_tang(giay_ds13);
                           }
                           luu_giai_ma_hien_thi_sau_khi_chinh();
                           break;

                case 2: if (phut_ds13==0x59) phut_ds13=0;
                           else
                           {
                               phut_ds13++;
                               phut_ds13=hieu_chinh_so_bcd_tang(phut_ds13);
                           }
                           luu_giai_ma_hien_thi_sau_khi_chinh();
                           break;

                case 3: if (gio_ds13==0x23) gio_ds13=0;
                           else
                           {
                               gio_ds13++;
                               gio_ds13=hieu_chinh_so_bcd_tang(gio_ds13);
                           }
                           luu_giai_ma_hien_thi_sau_khi_chinh();
                           break;

                case 4: if (ngay_ds13==0x31) ngay_ds13=1;
                           else
                           {
                               ngay_ds13++;
                               ngay_ds13=hieu_chinh_so_bcd_tang(ngay_ds13);
                           }
                           luu_giai_ma_hien_thi_sau_khi_chinh();
                           break;
            }
        }
    }
}

```

```

        {
            ngay_ds13++;
            ngay_ds13=hieu_chinh_so_bcd_tang(ngay_ds13);
        }
        luu_giai_ma_hien_thi_sau_khi_chinh();
        break;

    case 5: if (thang_ds13==0x12) thang_ds13=1;
    else
    {
        thang_ds13++;
        thang_ds13=hieu_chinh_so_bcd_tang(thang_ds13);
    }
    luu_giai_ma_hien_thi_sau_khi_chinh();
    break;

    case 6: if (nam_ds13==0x99) nam_ds13=0;
    else
    {
        nam_ds13++;
        nam_ds13=hieu_chinh_so_bcd_tang(nam_ds13);
    }
    luu_giai_ma_hien_thi_sau_khi_chinh();

    break;
    default: break;
}
while(!input(up));
}
}

void phim_dw()
{
    if (!input(dw))
    {
        delay_ms(20);
        if (!input(dw))
        {
            switch (giatri_mod)
            {
                case 1: if (giay_ds13==0) giay_ds13=0x59;
                else
                {
                    giay_ds13--;
                    phut_ds13=hieu_chinh_so_bcd_giam(giay_ds13);
                }
                luu_giai_ma_hien_thi_sau_khi_chinh();
                break;

                case 2: if (phut_ds13==0) phut_ds13=0x59;
                else
                {
                    phut_ds13--;
                }
            }
        }
    }
}

```

```

        phut_ds13=hieu_chinh_so_bcd_giam(phut_ds13);
    }
    luu_giai_ma_hien_thi_sau_khi_chinh();
    break;

case 3: if (gio_ds13==0) gio_ds13=0x23;
else
{
    gio_ds13--;
    gio_ds13=hieu_chinh_so_bcd_giam(gio_ds13);
}
luu_giai_ma_hien_thi_sau_khi_chinh();
break;

case 4: if (ngay_ds13==0) ngay_ds13=0x31;
else
{
    ngay_ds13--;
    ngay_ds13=hieu_chinh_so_bcd_giam(ngay_ds13);
}
luu_giai_ma_hien_thi_sau_khi_chinh();
break;

case 5: if (thang_ds13==1) thang_ds13=0x12;
else
{
    thang_ds13--;
    thang_ds13=hieu_chinh_so_bcd_giam(thang_ds13);
}
luu_giai_ma_hien_thi_sau_khi_chinh();
break;

case 6: if (nam_ds13==0) nam_ds13=0x99;
else
{
    nam_ds13--;
    nam_ds13=hieu_chinh_so_bcd_giam(nam_ds13);
}
luu_giai_ma_hien_thi_sau_khi_chinh();
break;

default: break;
}
while(!input(dw));
}
}

void nhap_nhay_con_tro()
{
switch (giatri_mod)
{
case 1: lcd_command(0xd3);
break;
}
}

```

```

        case 2: lcd_command(0xd0);
        break;
        case 3: lcd_command(0xcd);
        break;
        case 4: lcd_command(0x8d);
        break;
        case 5: lcd_command(0x90);
        break;
        case 6: lcd_command(0x93);
        break;
    default: break;
}
}

void main()
{
    set_up_port_ic_chot();
    setup_lcd();
    lcd_command(lcd_addr_line1);
    lcd_data("clock:");
    doc_thoi_gian_tu_realtime();
    if (ma_ds13!=ma_ds)
    {
        thiet_lap_thoi_gian_hien_tai();
        nap_thoi_gian_htai_vao_ds1307();
    }
    giatri_mod=0;
    hien_thi_gia_tri_mod();

    doc_thoi_gian_tu_realtime();
    giaytam=giay_ds13;

    while(true)
    {
        doc_thoi_gian_tu_realtime();
        if (giaytam!=giay_ds13)
        {
            giaytam=giay_ds13;
            hien_thi_gia_tri_mod();
            hien_thi_thoi_gian_ds1307();
        }
        for(k=0;k<10000;k++)
        {
            phim_mod(); phim_up(); phim_dw();
        }
        if (giatri_mod!=0) nhap_nhay_con_tro();
    }
}

```

- d. Tiến hành biên dịch và nạp.
- e. Quan sát kết quả:

- f. Giải thích chương trình: Trong chương trình có chỉnh được ngày tháng năm và có cài tiến chương trình con hiệu chỉnh số BCD tăng và giảm để có thể hiệu chỉnh rộng hơn, cho phép nhập nháy con trỏ tại vị trí của thông số chỉnh.

614/ Tr. 143

Bài tập 804. Hãy hiệu chỉnh chương trình trên sao cho khi hiệu chỉnh đối tượng nào thì đối tượng đó nháy nháy (không phải con trỏ), ví dụ chỉnh giờ thì giờ nháy nháy, giải thuật là cho đối tượng đó sáng $\frac{1}{2}$ giây rồi tắt $\frac{1}{2}$ giây còn lại.

Lưu tên file là “BAI_804_I2C_RT_LCD_3KEY_NN”

Bài tập 805. Hiệu chỉnh bài 804 sao cho khi phút tăng lên 1 thì buzzer kêu bíp, thời gian mở buzzer là 1 giây hoặc 2 giây.

Lưu tên file là “BAI_805_I2C_RT_LCD_3KEY_BUZZER”

Bài tập 806. Hiệu chỉnh bài 804 có thêm chức năng hẹn giờ tắt mở thiết bị với yêu cầu như sau: thêm hàng thứ 3 hiển thị thời gian hẹn chỉ có giờ phút giây, sử dụng các phím MOD, UP và DW để cài đặt thời gian tắt mở thiết bị, khi đúng giờ thì mở buzzer, sau 5 giây thì tắt buzzer.

Lưu tên file là “BAI_806_I2C_RT_LCD_3KEY_ALARM”

SỐ BCD, SS giay = 30
IF(GIAY == 0X30)

Bài tập 807. Hiệu chỉnh bài 805 có thêm chức năng reng chuông giờ học, với yêu cầu như sau:

Các thông số reng chuông (dùng buzzer thay cho chuông):

07-00-00 reng 1 hồi chuông, 00 giây mở chuông, 03 giây tắt chuông.

07-50-00 reng 1 hồi chuông, 00 giây mở chuông, 03 giây tắt chuông.

08-40-00 reng 1 hồi chuông, 00 giây mở chuông, 03 giây tắt chuông.

Hàng thứ 3 hiển thị thời gian sắp đến chuông chỉ có giờ phút giây.

Lưu tên file là “BAI_807_I2C_RT_LCD_3KEY_CHUONG”

Chú ý: các thông số thời gian bạn có thể thay đổi thành ngắn để không phải chờ đợi quá lâu như sau:

07-00-00 reng 1 hồi chuông, 00 giây mở chuông, 03 giây tắt chuông.

07-02-00 reng 1 hồi chuông, 00 giây mở chuông, 03 giây tắt chuông.

07-04-00 reng 1 hồi chuông, 00 giây mở chuông, 03 giây tắt chuông.

Bạn có thể thêm nhiều mốc thời gian để reng chuông, có thể thay đổi nhiều hồi reng chuông khác nhau.

Bài tập 808. Hãy viết chương trình hiển thị giờ phút giây bằng số có kích thước lớn đã làm ở chương LCD.

Lưu tên file là “BAI_808_I2C_RT_LCD_TO”

Bài tập 809. Hãy viết chương trình hiển thị giờ phút giây bằng số có kích thước bình thường và kích thước lớn đã làm ở chương LCD. Có 1 nút để chuyển chế độ hiển thị bình thường và hiển thị kích thước lớn.

Lưu tên file là “BAI_809_I2C_RT_LCD_TO_NHO”

8.3 KHẢO SÁT IC PCF8591

8.3.1 Giới thiệu IC PCF8591

Ở chương 7 chúng ta đã thực hành các bài chuyển đổi tương tự sang số sử dụng các bộ chuyển đổi tích hợp bên trong vi điều khiển. Ở chương này chúng ta thực tập các bài chuyển đổi tương tự sang số và số sang tương tự do IC PCF8591 thực hiện, IC này giao tiếp với vi điều khiển theo chuẩn I2C.

IC PCF8591 là một chip CMOS thu thập dữ liệu đơn dùng nguồn cung cấp đơn công suất thấp 8-bit với 4 ngõ vào tương tự và một ngõ ra tương tự giao tiếp nối tiếp I2C.

Ba chân địa chỉ A0, A1 và A2 được sử dụng cho lập trình địa chỉ phần cứng, cho phép mở rộng lên đến 8 thiết bị kết nối với theo chuẩn bus I2C mà không cần thêm phần cứng. Địa chỉ, điều khiển và dữ liệu đến và đi từ thiết bị sẽ được chuyển nối tiếp thông qua 2 dây của bus I2C.

Các chức năng của thiết bị bao gồm ghép kênh các ngõ vào tương tự, chuyển đổi tương tự sang số 8 bit, chức năng theo dõi và giữ. Tốc độ chuyển đổi tối đa được cho bởi tốc độ tối đa của bus I2C.

8.3.2 Cấu hình IC PCF8591

- Nguồn cung cấp là nguồn đơn từ 2,5V đến 6V.
- Dòng điện không sử dụng rất thấp.
- Dữ liệu vào và ra nối tiếp theo chuẩn I2C.
- Lựa chọn địa chỉ I2C bởi 3 chân địa chỉ phần cứng.
- Tốc độ lấy mẫu cao nhất bằng tốc độ I2C.
- Có 4 kênh tương tự ngõ vào có thể cấu hình như ngõ vào đơn cực hoặc vi sai.
- Có thể lựa chọn kênh tự động tăng.
- Điện áp tương tự từ V_{SS} đến V_{DD}.
- Có mạch điện theo dõi và giữ trên chip.

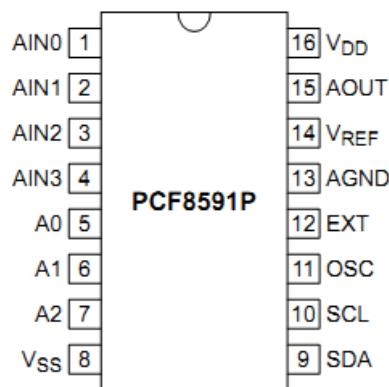
- Chuyển đổi A/D sáp xỉ liên tiếp 8 bit.
- Có 1 bộ DAC với 1 ngõ ra tương tự.

8.3.3 Ứng dụng IC PCF8591

- Dùng để giám sát nguồn.
- Thiết lập các tham chiếu.
- Vòng điều khiển tương tự.

8.3.4 Sơ đồ chân IC PCF8591

Sơ đồ chân IC như hình 8-11:



Hình 8-11. Sơ đồ khối IC PCF8591.

Chức năng từng chân như bảng 8-1.

Bảng 8-1. Tên và chức năng từng chân của IC PCF8591.

TT	Tên chân	Chân	Chức năng
1	AIN0	1	Ngõ vào tương tự thứ 0 (analog inputs A/D converter)
2	AIN1	2	Ngõ vào tương tự thứ 1 (analog inputs A/D converter)
3	AIN2	3	Ngõ vào tương tự thứ 2 (analog inputs A/D converter)
4	AIN3	4	Ngõ vào tương tự thứ 3 (analog inputs A/D converter)
5	A0	5	Ngõ vào thiết lập địa chỉ thiết bị (hardware slave address)
6	A1	6	Ngõ vào thiết lập địa chỉ thiết bị (hardware slave address)
7	A2	7	Ngõ vào thiết lập địa chỉ thiết bị (hardware slave address)
8	VSS	8	Chân nối mass (ground supply voltage)
9	SDA	9	I2C- bus serial data input and output

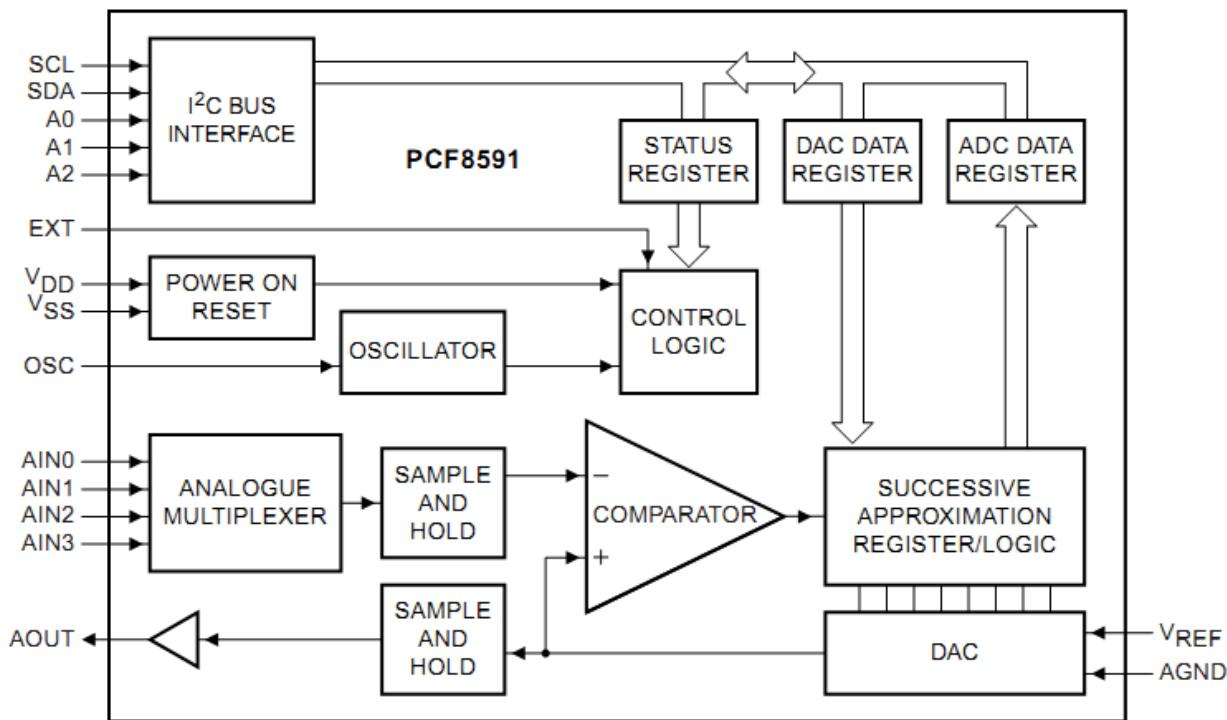
10	SCL	10	I2C-bus serial clock input
11	OSC	11	Ngõ vào ra của dao động (oscillator input/output)
12	EXT	12	Ngõ vào dao động lựa chọn (external/internal switch for oscillator input)
13	AGND	13	Chân nối mass của tín hiệu tương tự (analog ground supply)
14	VREF	14	Ngõ vào nối điện áp tham chiếu (voltage reference input)
15	AOUT	15	Ngõ ra tín hiệu tương tự của DA (analog output (D/A converter))
16	VDD	16	Nguồn cung cấp (supply voltage)

8.3.5 Sơ đồ khối IC PCF8591

Sơ đồ khối IC như hình 8-12:

Cấu trúc của IC bao gồm các khối:

- Khối giao tiếp chuẩn I2C.
- Khối nguồn cung cấp và reset
- Khối dao động nội
- Khối điều khiển logic
- Thanh ghi trạng thái
- Thanh ghi dữ liệu DAC
- Thanh ghi dữ liệu ADC
- Khối đa hợp 4 kênh tương tự
- Khối ADC-DAC bao gồm hai khối lấy mẫu và giữ cho hai ngõ vào, bộ so sánh, bộ chuyển đổi DAC và thanh ghi xấp xỉ liên tiếp, nguồn điện áp tham chiếu.
- Khối khuếch đại đệm tín hiệu tương tự.



Hình 8-12. Sơ đồ khối IC PCF8591.

8.3.6 Địa chỉ ngoài IC PCF8591

Mỗi thiết bị PCF8591 trong hệ thống bus I²C được kích hoạt bằng cách gửi đúng địa chỉ cho thiết bị. Địa chỉ bao gồm một phần cố định và một phần lập trình. Phần lập trình được thiết lập bởi các chân địa chỉ A0, A1 và A2.

Địa chỉ luôn là byte đầu tiên được gửi sau khi thực hiện xong điều kiện *start* trong giao thức bus I²C. Trong byte địa chỉ thì bit cuối cùng là bit xác nhận đọc / ghi.

8.3.7 Thanh ghi điều khiển của IC PCF8591

Byte thứ hai gửi đến IC PCF8591 được lưu trữ trong thanh ghi điều khiển (control register) để điều khiển các chức năng của thiết bị như hình 8-13.

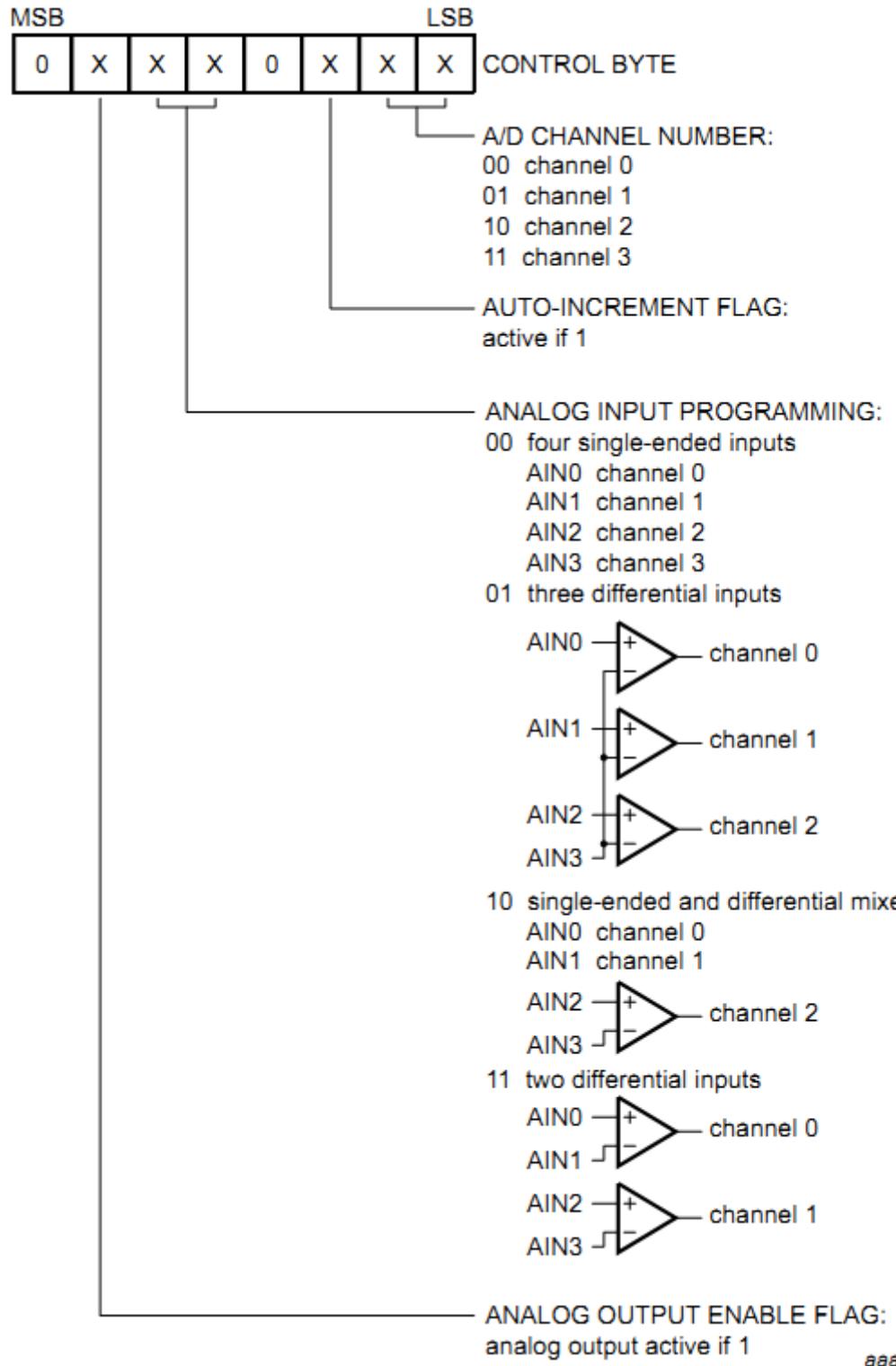
Bốn bit cao của thanh ghi điều khiển được sử dụng để thiết lập:

- Bit thứ 7 luôn bằng 0
- Bit thứ 6 dùng để cho phép xuất tín hiệu tương tự ở ngõ ra tương tự - hoạt động DAC
- Bit thứ 5 và 4 dùng để thiết lập cho các chế độ ngõ vào tương tự như chế độ ngõ vào duy nhất hoặc chế độ vi sai ngõ vào.

Bốn bit thấp:

- Bit thứ 3 luôn bằng 0
- Bit thứ 2 là bit cho phép tự động tăng kênh nếu bằng 1, kênh chọn tự động tăng lên sau mỗi lần chuyên đổi xong.

- Bit thứ 1 và 0 dùng để chọn 1 trong 4 kênh khi thực hiện đọc dữ liệu.



Hình 8-13. Cấu trúc thanh ghi điều khiển của IC PCF8591.

Nếu chế độ tự động tăng được yêu cầu trong các ứng dụng thì phải sử dụng bộ dao động nội bộ và cờ cho phép ngõ ra tương tự phải bằng 1 (bit thứ 6 của thanh ghi điều khiển). Điều này cho phép bộ dao động nội bộ chạy liên tục. Cách này sẽ ngăn chặn lỗi chuyển đổi do dao động

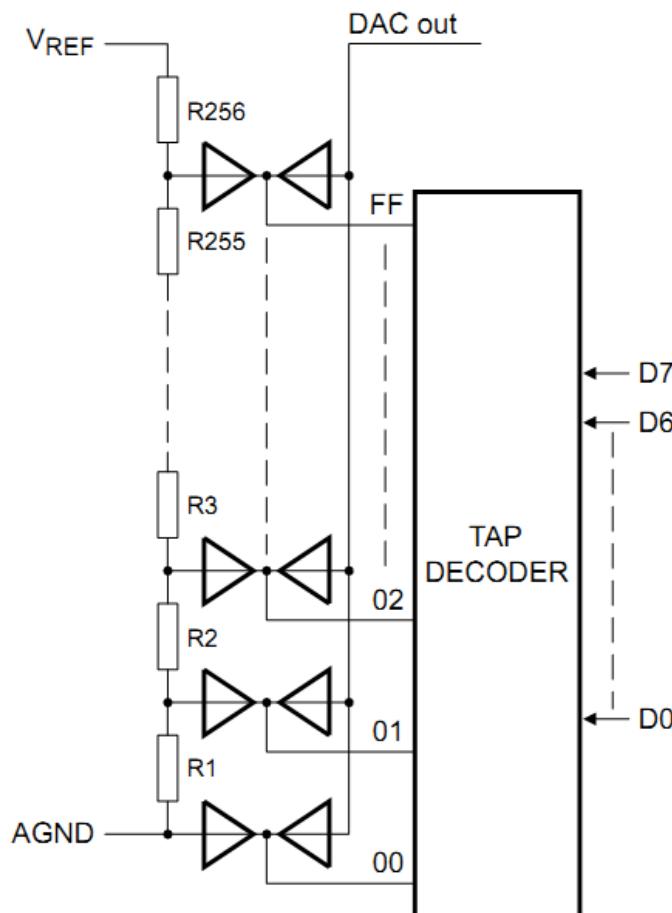
khởi động chậm trễ gây ra. Cờ cho phép ngõ ra tương tự có thể reset ở thời điểm khác để giảm công suất tiêu thụ điện khi không còn chuyển đổi.

Bit có trọng số cao nhất được dành riêng cho chức năng trong tương lai có thể và được thiết lập ở mức logic 0. Sau khi cấp điện thì mạch reset khi có điện POR sẽ reset tất cả các bit của thanh ghi điều khiển và tắt dao động để tiết kiệm năng lượng và ngõ ra tín hiệu tương tự ở trạng thái tổng trở cao.

8.3.8 Chuyển đổi D/A của IC PCF8591

Byte thứ ba được gửi đến IC PCF8591 sẽ lưu trữ trong thanh ghi dữ liệu DAC và được chuyển đổi thành điện áp tương tự bởi bộ chuyển đổi DAC.

Chuyển đổi DA bao gồm một chuỗi cầu điện trở mắc nối tiếp và nối với điện áp tham chiếu bên ngoài tạo thành 256 cấp điện áp khác nhau và switch lựa chọn. Bộ giải mã sẽ chuyển mạch cho một trong 256 cấp điện áp này đến ngõ ra DAC – xem hình 8-14.

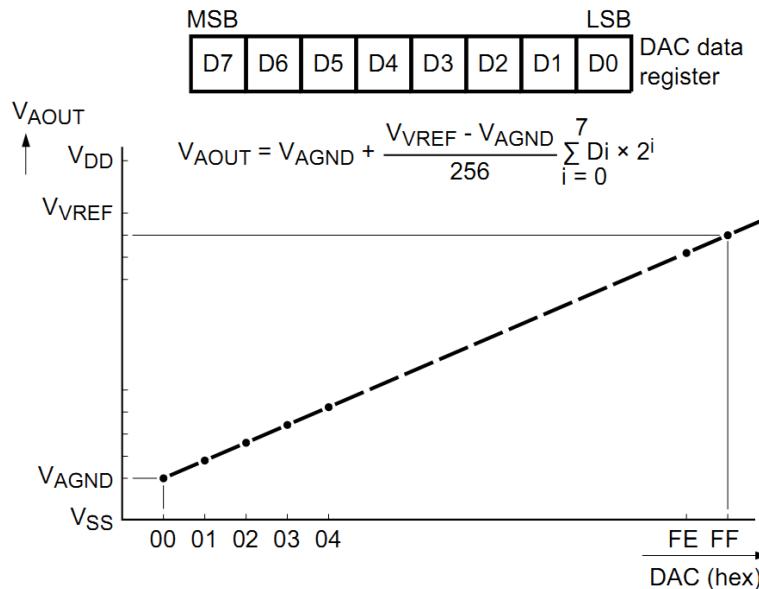


Hình 8-14. Cấu trúc khối DAC của IC PCF8591.

Điện áp đầu ra tương tự được đệm bởi một bộ khuếch đại tự động hệ số KĐ bằng 1. Thiết lập bit cho phép xuất ngõ ra tương tự trong thanh ghi điều khiển để tắt mở bộ đệm này. Trong trạng thái hoạt động, điện áp đầu ra được giữ cho đến khi một byte dữ liệu tiếp tục được gửi đi để chuyển đổi tiếp.

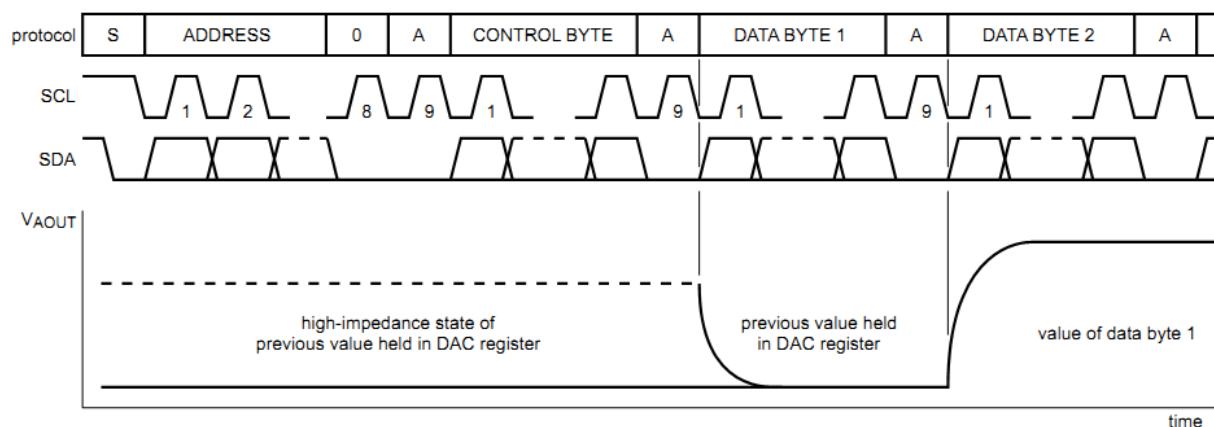
Bộ chuyển đổi DA trên chip cũng được sử dụng để phục vụ cho chuyển đổi AD liên tiếp xấp xỉ. Để khỏi giải thoát khỏi DA phục vụ cho chuyển đổi AD thì mạch khuếch đại có độ lợi bằng 1 cần phải có thêm mạch theo dõi và giữ để giữ điện áp ngõ ra tương tự trong lúc thực hiện chuyển đổi AD.

Công thức cho điện áp ngõ ra được thể hiện trong hình 8-15:



Hình 8-15. Công thức điện áp ra và đặc tính chuyển đổi.

Các dạng sóng của chuyển đổi DA theo trình tự được thể hiện trong hình 8-16.



Hình 8-16. Dạng sóng chuyển đổi D/A.

8.3.9 Chuyển đổi A/D của IC PCF8591

Chuyển đổi A/D sử dụng kỹ thuật chuyển đổi xấp xỉ liên tiếp. Bộ chuyển đổi D/A chuyển đổi và bộ so sánh độ lợi cao được sử dụng tạm thời trong chu kỳ chuyển đổi A/D.

Một chu kỳ chuyển đổi A/D được bắt đầu sau khi nhận địa chỉ hợp lệ ở chế độ đọc gởi đến IC PCF8591. Chu kỳ chuyển đổi A/D được kích hoạt khi xuất hiện cạnh của xung đồng hồ trả

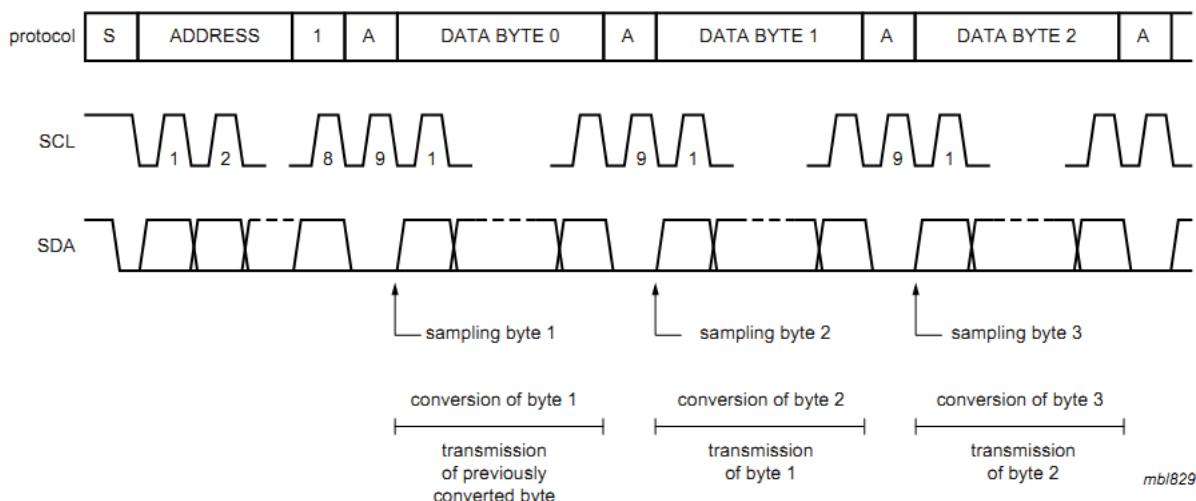
lời (acknowledge) và thực hiện chuyển đổi trong khi đó vẫn thực hiện truyền kết quả của chuyển đổi trước đó, xem hình 8-17.

Kết quả chuyển đổi được lưu trong thanh ghi dữ liệu ADC và chờ truyền đi. Nếu cờ tự động tăng cờ được thiết lập, các kênh tiếp theo được chọn. Byte đầu tiên truyền đi trong chu kỳ đọc có chứa mã kết quả chuyển đổi của chu kỳ đọc trước.

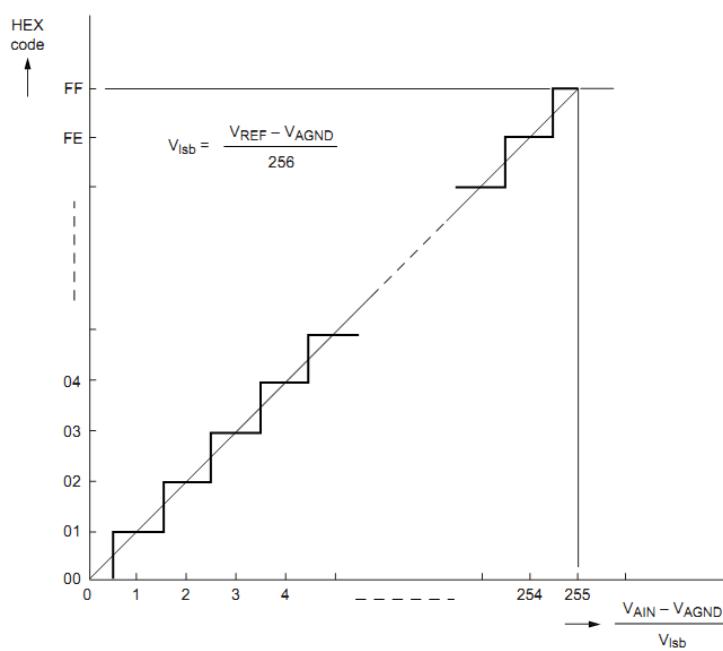
Sau điều kiện POR, các byte đầu tiên đọc có giá trị là 80h.

Tốc độ tối đa của chuyển đổi A/D bằng với tốc độ thực tế của bus I2C.

Dạng sóng kết quả chuyển đổi theo chế độ ngõ vào đơn cực như hình 8-18.

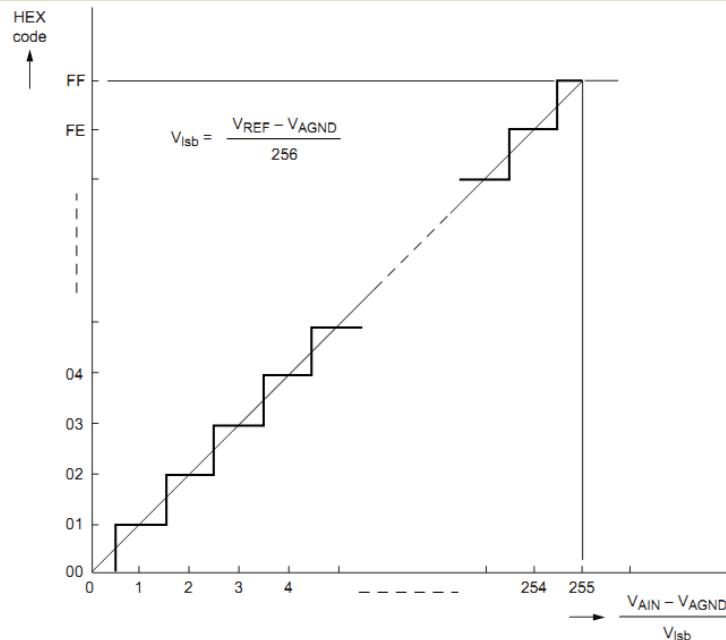


Hình 8-17. Dạng sóng truyền dữ liệu của IC PCF8591.



Hình 8-18. Dạng sóng chuyển đổi A/D ở chế độ đơn cực.

Dạng sóng kết quả chuyển đổi theo chế độ vi sai ngõ vào như hình 8-19:

**Hình 8-19. Dạng sóng chuyển đổi A/D ở chế độ ngõ vào vi sai.**

8.3.10 Điện áp tham chiếu của IC PCF8591

Cả hai chuyển đổi D/A và A/D đều sử dụng điện áp tham chiếu bên ngoài cấp cho các chân VREF và AGND. Chân AGND được nối với mass tương tự hoặc có thể nối với nguồn điện áp DC để chỉnh điện áp nếu bị lệch.

8.3.11 Bộ dao động của IC PCF8591

Bộ dao động trên chip tạo ra các tín hiệu đồng hồ cần thiết cho chuyển đổi A/D và làm tươi bộ khuếch đại đệm. Khi sử dụng dao động này chân EXT phải nối với VSS (GND). Tần số dao động xuất hiện tại chân OSC.

Nếu chân EXT kết nối với VDD thì ngõ ra dao động OSC sẽ chuyển sang trạng thái trở kháng cao cho phép nhận tín hiệu đồng hồ từ bên ngoài đưa đến chân OSC.

8.3.12 Địa chỉ thanh ghi của IC PCF8591

Byte địa chỉ của IC PCF8591 như hình 8-20:

Bit	Slave address								0 LSB
	7 MSB	6	5	4	3	2	1		
slave address	1	0	0	1	A2	A1	A0	R/W	

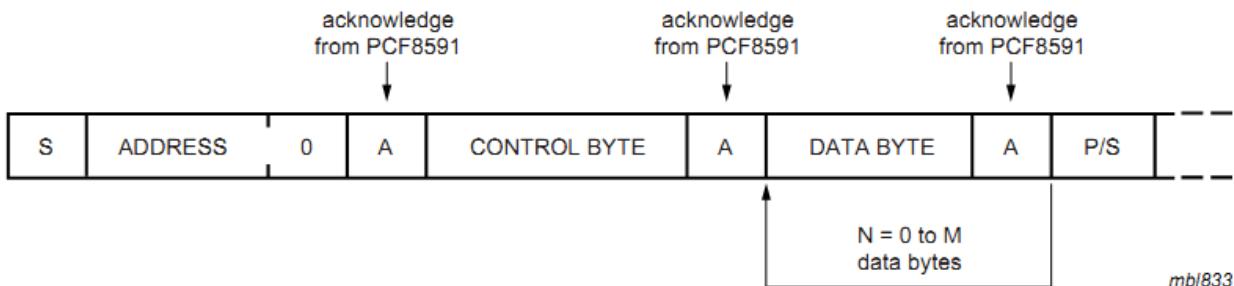
Hình 8-20. Địa chỉ của IC PCF8591.

Tín hiệu đọc/ghi như hình 8-21:

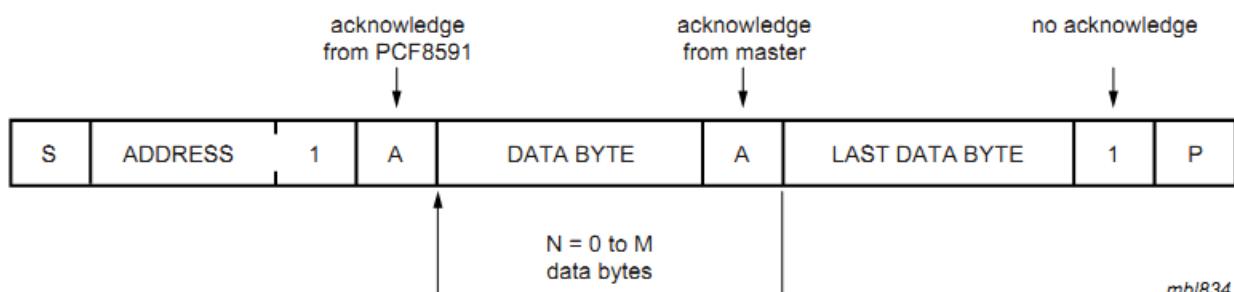
R/W	Description
0	write data
1	read data

Hình 8-21. Tín hiệu đọc ghi.

Chế độ ghi dữ liệu vào IC PCF 8591 như hình 8-22:

**Hình 8-22. Chế độ ghi dữ liệu vào IC PCF8591.**

Chế độ đọc dữ liệu từ IC PCF 8591 như hình 8-23:

**Hình 8-23. Chế độ đọc dữ liệu từ IC PCF8591.**

Về giao tiếp chuẩn truyền dữ liệu I2C thì đã trình bày chi tiết ở phần giao tiếp với IC thời gian thực DS13B07 nên ở đây không trình bày lại.

8.3.13 Giao tiếp vi điều khiển với IC PCF8591

Hình 8-9 ở trên, trình bày kết nối với vi điều khiển theo chuẩn I2C giống các thiết bị khác.

Địa chỉ của IC PCF 8591 trong bộ thực hành là 0x90 đối với lệnh ghi và 0x91 đối với lệnh đọc vì 3 chân thiết lập địa chỉ cứng nối mass.

8.3.14 Các chương trình giao tiếp IC PCF 8591

Độ phân giải của IC PCF8591 được điều chỉnh bằng biến trở bên ngoài và đã hiệu chỉnh với độ phân giải là 10mV để tương thích với cảm biến LM35.

Thư viện PCF8591. Chương trình thư viện điều khiển IC PCF8591 theo chuẩn I2C đã cho trong thư mục của bạn có tên là.

“TV_PICKIT2_SHIFT_PCF8591_I2C.c”

Bạn có thể sử dụng cho các bài thực hành và nên đọc hiểu.

```

#define pcf8591_address_wr    0x90
#define pcf8591_address_rd    0x91

#define pcf8591_channel_0      0x40
#define pcf8591_channel_1      0x41
#define pcf8591_channel_2      0x42
#define pcf8591_channel_3      0x43

void pcf8591_chonkenh(unsigned char kenh)
{
    i2c_start();
    i2c_write(pcf8591_address_wr);
    i2c_write(kenh);
    i2c_stop();
}

void pcf8591_xuat_dac(unsigned char dac_out)
{
    i2c_start();
    i2c_write(pcf8591_address_wr);
    i2c_write(pcf8591_channel_0);
    i2c_write(dac_out);
    i2c_stop();
}

unsigned char pcf8591_doc_adc()
{
    unsigned char tam;
    i2c_start();
    i2c_write(pcf8591_address_rd);
    tam= i2c_read(0);
    i2c_stop();
    return tam;
}

```

Bài mẫu 811. Chương trình đo nhiệt độ từ cảm biến LM35A nối với kênh AN0 của IC ADC PCF 8591, hiển thị kết quả đo trên module 4 led 7 đoạn.

Lưu tên file là “BAI_811_PCF8591_4LED_LM35A.C”

- Mục đích: biết lập trình dùng ADC giao tiếp chuẩn I2C để đo nhiệt độ từ cảm biến LM35 hiển thị trên led 7 đoạn.
- Lưu đồ: sinh viên hãy tự viết.
- Chương trình:

```

#include <tv_pickit2_shift_1.c>
#include <tv_pickit2_shift_pcf8591_i2c.c>
unsigned int8 i,dac;
unsigned int16 adc_lm35a;//,adc_lm35b,adc_cds;
void main()
{

```

```

set_up_port_ic_chot();
dac=0;
pcf8591_xuat_dac(250);
while(true)
{
    delay_us(100);
    adc_lm35a=0;
    for(i=0;i<100;i++)
    {
        pcf8591_chonkenh(pcf8591_channel_0); //lm35a
        delay_us(20);
        adc_lm35a=adc_lm35a + pcf8591_doc_adc();
    }
    adc_lm35a= adc_lm35a/100;
    xuat_4led_7doan_giaima_xoa_so0(adc_lm35a);
}
}

```

- d. Tiến hành biên dịch và nạp.
- e. Quan sát kết quả: hiển thị nhiệt độ của cảm biến LM35A trên 2 led 7 đoạn. Có thể mở đèn gia tăng nhiệt độ để thấy sự thay đổi.
- f. Giải thích chương trình:

Bài tập 812. Hãy thực hiện giống bài 811 nhưng thêm phần báo chuông khi nhiệt độ quá 40 độ C.

Lưu tên file là “BAI_812_PCF8591_4LED_LM35A_BUZ.C”

Bài tập 813. Hãy thực hiện giống bài 812 nhưng đo kênh AN1 nối với cảm biến LM35B và hiển thị trên 2 led 7 đoạn bên trái.

Lưu tên file là “BAI_813_PCF8591_4LED_LM35B_BUZ.C”

Bài tập 814. Hãy viết chương trình tự động đo và không chế nhiệt độ trong tầm từ 33 đến 35 độ. Khi nhiệt độ nhỏ hơn 33 thì mở đèn triac để gia tăng nhiệt độ. Khi nhiệt độ quá 35 thì tắt đèn triac và báo động bằng buzzer. Khi nhiệt độ nhỏ hơn 35 thì tắt buzzer.

Lưu tên file là “BAI_814_PCF8591_4LED_LM35A_BUZ_DEN.C”

Bài mẫu 815. Chương trình đo nhiệt độ từ 2 cảm biến LM35A nối với 2 kênh AN0 và AN1 của IC ADC PCF 8591, hiển thị kết quả đo trên module 4 led 7 đoạn, mỗi kênh 2 led.

Lưu tên file là “BAI_815_PCF8591_4LED_LM35AB.C”

- a. Mục đích: biết lập trình dùng ADC giao tiếp chuẩn I2C để đo nhiệt độ từ 2 kênh nối với 2 cảm biến LM35 hiển thị trên led 7 đoạn.
- b. Lưu ý: sinh viên hãy tự viết.
- c. Chương trình:

```
#include <tv_pickit2_shift_1.c>
#include <tv_pickit2_shift_pcf8591_i2c.c>
#define solando 100
unsigned int8 i;
unsigned int16 adc_lm35a,adc_lm35b;//adc_cds;
void main()
{
    set_up_port_ic_chot();
    while(true)
    {
        adc_lm35a=0;
        for(i=0;i<solando;i++)
        {
            pcf8591_chonkenh(pcf8591_channel_0); //lm35a
            delay_us(20);
            adc_lm35a=adc_lm35a + pcf8591_doc_adc();
        }
        adc_lm35b=0;
        for(i=0;i<solando;i++)
        {
            pcf8591_chonkenh(pcf8591_channel_1); //lm35b
            delay_us(20);
            adc_lm35b=adc_lm35b + pcf8591_doc_adc();
        }
        adc_lm35a= adc_lm35a/solando ;
        adc_lm35b= adc_lm35b/solando ;
        xuat_4led_7doan_2so_4led(adc_lm35b, adc_lm35a);
    }
}
```

- d. Tiến hành biên dịch và nạp.
- e. Quan sát kết quả: hiển thị nhiệt độ của 2 cảm biến LM35A và LM35B trên 4 led 7 đoạn. Có thể mở đèn gia tăng nhiệt độ để thấy sự thay đổi.
- f. Giải thích chương trình:

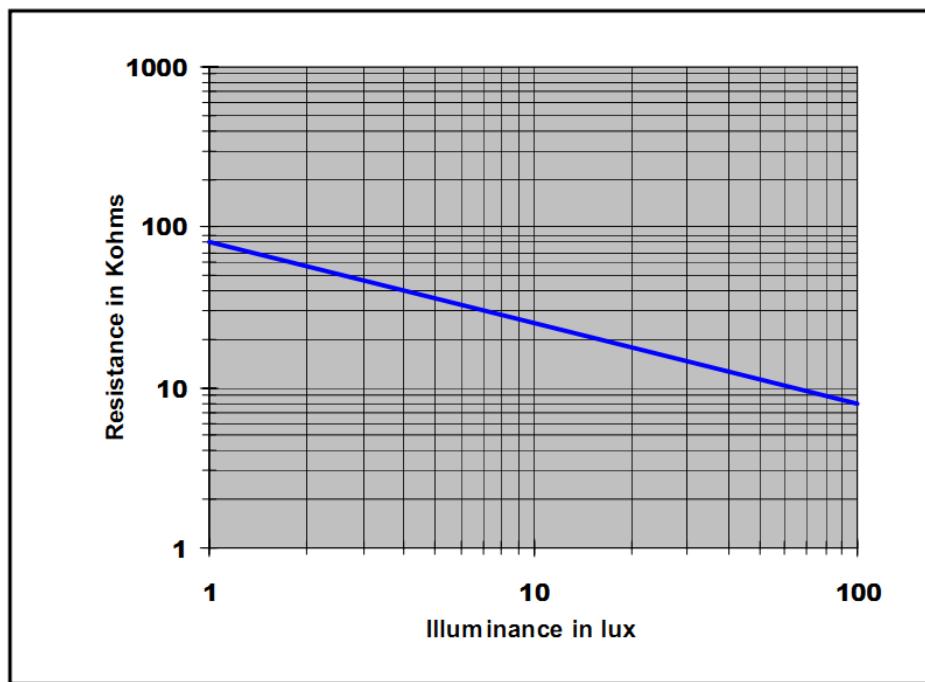
8.4 CẢM BIẾN QUANG TRỞ

8.4.1 Khảo sát quang trở

Trong bộ thực hành có dùng cảm biến quang để chuyển sự thay đổi của ánh sáng thành điện trở và qua cầu điện trở thành sự thay đổi của điện áp.

Mối quan hệ giữa giá trị điện trở và cường độ ánh sáng như hình 8-24.

CELL RESISTANCE VS. ILLUMINANCE



Hình 8-24. Mối quan hệ giữa sự thay đổi ánh sáng và điện trở.

Trong đồ thị ta thấy khi cường độ ánh sáng tăng thì giá trị điện trở giảm. Mối quan hệ theo tỷ lệ nghịch.

Do chúng ta không có thiết bị đo sánh sáng chuẩn nên ta không thể hiển thị giá trị đo ánh sáng được mà chỉ hiển thị giá trị số sau chuyên đổi trên led.

Các ứng dụng thường dùng để điều khiển đèn theo ánh sáng như tự động mở đèn khi trời tối hay tắt đèn khi trời sáng.

8.4.2 Vi điều khiển giao tiếp với quang trở

Trong bộ thực hành thì cảm biến quang trở nối với ngõ vào AN2 của IC ADC PCF8591, xem hình 8-9.

8.4.3 Các chương trình điều khiển dùng quang trở

Bài mẫu 821. Chương trình đo ánh sáng (illuminance in lux) dùng cảm biến photocell CDS PDV-P8103 nối với kênh AN3 của IC ADC PCF 8591, hiển thị kết quả đo trên module 4 led 7 đoạn. Chỉ thấy sự thay đổi.

Lưu tên file là “BAI_821_PCF8591_4LED_CDS.C”

- a. Mục đích: biết lập trình dùng ADC giao tiếp chuẩn I2C để đọc giá trị thay đổi của quang trở hiển thị trên led 7 đoạn.
- b. Lưu ý: sinh viên hãy tự viết.
- c. Chương trình:

```
#include <tv_pickit2_shift_1.c>
#include <tv_pickit2_shift_pcf8591_i2c.c>
```

```

#define      solando    100
unsigned   char    i;
unsigned   int16   adc_cds;
void main()
{
    set_up_port_ic_chot();
    while(true)
    {
        adc_cds=0;
        for(i=0;i<solando;i++)
        {
            pcf8591_chonkenh(pcf8591_channel_2); //cds
            delay_us(20);
            adc_cds=adc_cds + pcf8591_doc_adc();
        }
        adc_cds=adc_cds/solando;
        xuat_4led_7doan_giaima_xoa_so0(adc_cds);
    }
}

```

- d. Tiến hành biên dịch và nạp.
- e. Quan sát kết quả: sau khi nạp xong thì ta nhìn thấy giá trị số hiển thị trên module led 7 đoạn. Bạn có thể dùng tay che ánh sáng bằng cách để gần quang trở chứ không cần phải chạm vào, khi đó giá trị số hiển thị trên led sẽ thay đổi theo chiều tăng vì chúng tỷ lệ nghịch như trong đồ thị.
- f. Giải thích chương trình:

Bài mẫu 822. Chương trình điều khiển đèn tự động sáng hoặc tắt dùng cảm biến photocell CDS PDV-P8103 nối với kênh AN3 của IC ADC PCF 8591, hiển thị kết quả đo trên module 4 led 7 đoạn.

Khi trời tối thì mở đèn sáng, khi trời sáng thì đèn tắt.

Lưu tên file là “BAI_822_PCF8591_4LED_CDS_ON_OFF_DEN.C”

- a. Mục đích: biết lập trình điều khiển đèn tự động dùng quang trở.
- b. Lưu đồ: sinh viên hãy tự viết.
- c. Chương trình:

```

#include <tv_pickit2_shift_1.c>
#include <tv_pickit2_shift_pcf8591_i2c.c>
#define solando    100
#define gioihan_dk    180
unsigned   char    i;
unsigned   int16   adc_cds;
int1 tt_triac=0;

void sosanh_cds_dk_den()
{
    if ((adc_cds>gioihan_dk)&&(tt_triac==0))
    {
        tt_triac = 1;
        triac_2_on();
    }
}

```

```

else if ((adc_cds<gioihan_dk-20)&&(tt_triac==1))
{
    tt_triac = 0;
    triac_2_off();
}

void main()
{
    set_up_port_ic_chot();
    tt_triac=0;
    while(true)
    {
        adc_cds=0;
        pcf8591_chonkenh(pcf8591_channel_2);      //cds
        for(i=0;i<solando;i++)
        {
            adc_cds=adc_cds + pcf8591_doc_adc();
            delay_us(20);
        }
        adc_cds=adc_cds/solando;
        xuat_4led_7doan_giaima_xoa_so0(adc_cds);
        sosanh_cds_dk_den();
    }
}

```

- d. Tiến hành biên dịch và nạp.
- e. Quan sát kết quả: sau khi nạp xong thì ta nhìn thấy giá trị số hiển thị trên module led 7 đoạn. Bạn có thể dùng tay che ánh sáng cho đến khi vượt ngưỡng cho trong chương trình là biến “**GIOIHAN_DK**” cho là 180 thì điều khiển led sáng và ngược lại thì đèn tắt. Bạn có thể thay đổi ngưỡng cho phù hợp với thực tế.
- f. Giải thích chương trình:

8.5 CẢM BIẾN CHUYÊN ĐỘNG PIR

8.5.1 Khảo sát cảm biến chuyển động PIR

Cảm biến có hình ảnh như hình 8-25:



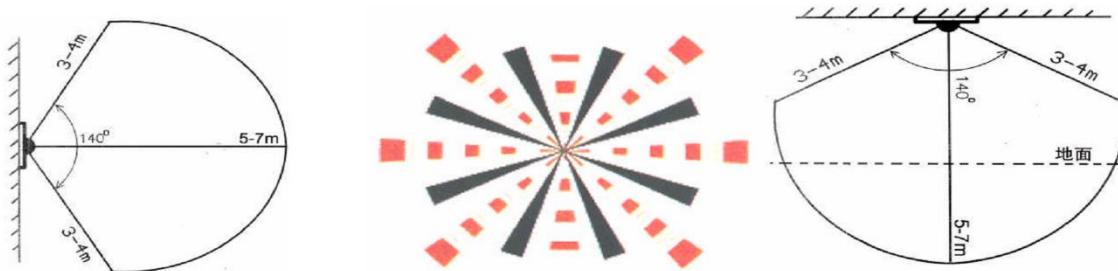
Hình 8-25. Hình ảnh cảm biến chuyển động.

Các thông số của cảm biến:

- Điện áp hoạt động từ 5V đến 20V DC.

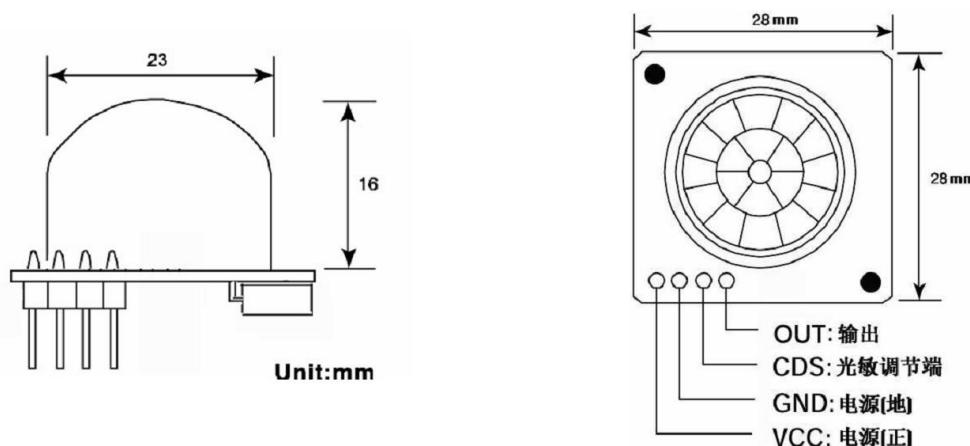
- Tín hiệu điện áp ra 2 mức cao và thấp: mức cao lớn nhất là 3,3V.
- Thời gian delay từ 5s đến 18 phút.

Phạm vi nhận biết di chuyển khi có gắn ống kính như hình 8-26:



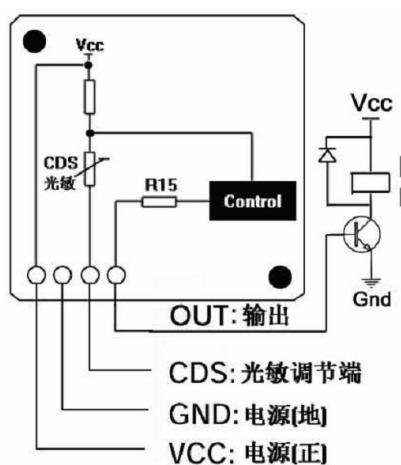
Hình 8-26. Phạm vi nhận biết của cảm biến chuyển động.

Sơ đồ chân của cảm biến PIR như hình 8-27:



Hình 8-27. Sơ đồ chân và tên các chân của cảm biến chuyển động.

Kết nối ngõ ra với tải như hình 8-28:

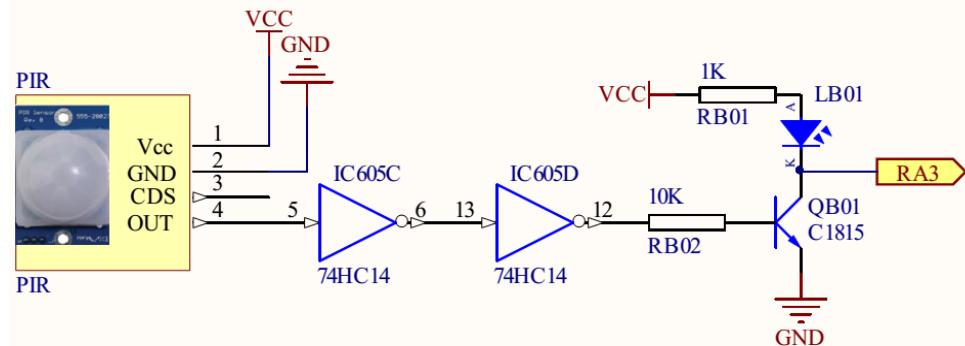


Hình 8-28. Sơ đồ kết nối tải của cảm biến chuyển động.

Trong bộ thí nghiệm thì cảm biến PIR được nối đến chân RA3 của vi điều khiển PIC.

8.5.2 Vi điều khiển giao tiếp với cảm biến PIR

Trong bộ thực hành thì cảm biến PIR nối với ngõ vào RA3/AN3 của vi điều khiển như hình 8-29:



Hình 8-29. Sơ đồ của cảm biến chuyển động PIR với vi điều khiển.

8.5.3 Các chương trình điều khiển dùng cảm biến PIR

Bài mẫu 831. Chương trình điều khiển đèn tự động sáng khi phát hiện có người chuyển động hoặc tắt đèn khi không có người chuyển động dùng cảm biến PIR.

Lưu tên file là “BAI_831_PIR_DK_DEN.C”

- Mục đích: biết lập trình dùng cảm biến PIR để điều khiển tắt mở đèn.
- Lưu đồ: sinh viên hãy tự viết.
- Chương trình:

```
#include <tv_pickit2_shift_1.c>
#define pir      pin_a3

int1 tt_triac=0;
void kiemtra_pir_mo_den()
{
    if(input(pir)&&(tt_triac==0))
    {
        triac_2_off();
        tt_triac=1;
    }
    else if (!input(pir)&&(tt_triac==1))
    {
        triac_2_on();
        tt_triac=0;
    }
}
void main()
{
    set_up_port_ic_chot();
    tt_triac=0;
    while(true)
    {
        kiemtra_pir_mo_den();
    }
}
```

- d. Tiến hành biên dịch và nạp.
- e. Quan sát kết quả: Sau khi nạp xong, nếu cảm biến PIR phát hiện không có chuyển động (đèn màu trắng gần cảm biến PIR tắt) thì đèn triac tắt, nếu có chuyển động (đèn màu trắng sáng) thì đèn triac sáng. Nếu phát hiện bạn chuyển động khiến đèn bật sáng thì các bạn hãy ngồi im không nhúc nhích cho đến khi đèn tắt, sau đó trọn mắt to hết cỡ xem nó có phát hiện không, nếu không thì hãy liết mắt qua lại mà nó cũng không mở thì hãy chuyển động để nó phát hiện và mở đèn. Có thể dùng báo động bằng âm thanh nhưng ôn nên dùng đèn để báo hiệu.
- f. Giải thích chương trình:

Bài tập 832. Chương trình điều khiển theo 2 yêu cầu:

Yêu cầu 1: dùng cảm biến photocell CDS PDV-P8103 để điều khiển đèn: khi trời sáng thì tắt đèn, khi trời tối thì mở đèn. Nguồn so sánh là 180.

Yêu cầu 2: dùng cảm biến PIR để điều khiển đèn: khi không có người chuyển động thì 32 led tắt, khi có người chuyển động thì 32 led sáng.

Lưu tên file là “BAI_832_PIR_DK_32LED_CDS_DK_DEN.C”

Bài tập 833. Chương trình điều khiển đèn tự động sáng hoặc tắt theo 2 yêu cầu:

Yêu cầu 1: dùng cảm biến photocell CDS PDV-P8103 để điều khiển đèn: khi trời sáng thì tắt đèn, khi trời tối thì mở đèn. Nguồn so sánh là 180.

Yêu cầu 2: dùng cảm biến PIR để điều khiển đèn: khi không có người chuyển động thì tắt đèn, khi có người chuyển động thì mở đèn. Chỉ có hiệu lực khi trời sáng.

Lưu tên file là “BAI_833_PIR_CDS_DK_DEN.C”

8.6 DAC IC PCF8591 ĐIỀU KHIỂN ĐÈN

IC PCF8591 là ADC và DAC, ở phần trước thì đã thực hành chức năng chuyển đổi tín hiệu tương tự thành tín hiệu số, phần này thực hành chuyển đổi tín hiệu số sang tín hiệu tương tự để điều khiển đèn led ở ngõ ra.

Bài mẫu 841. Chương trình chuyển đổi DAC của IC ADC PCF 8591, hiển số nhị phân trên module 4 led 7 đoạn, dữ liệu số gửi đến IC PCF xuất ra chân Aout (xem hình) có nối với led đơn.

Điện áp ra bằng dữ liệu số nhận với độ phân giải.

Trong bài thực hành này ta sẽ xuất dữ liệu từ 210 đến 255 để nhìn thấy led tắt khi giá trị thấp, cường độ sáng tăng dần khi giá trị số tăng, khi bằng 255 thì cường độ sáng cực đại.

Do led gắn điện trở hạn dòng và độ phân giải thấp nên giá trị số phải lớn thì ta mới nhìn thấy led sáng.

Lưu tên file là “BAI_841_DAC_PCF8591_UP.C”

- a. Mục đích: biết sử dụng chức năng DAC của IC PCF8591 điều khiển đèn led.

- b. Lưu đồ: sinh viên hãy tự viết.
c. Chương trình:

```
#include <tv_pickit2_shift_1.c>
#include <tv_pickit2_shift_pcf8591_i2c.c>
unsigned int8    dac;
void main()
{
    set_up_port_ic_chot();
    dac = 210;
    while(true)
    {
        xuat_4led_7doan_giaima_xoa_so0(dac);
        xuat_32led_don_4byte(0,0,0,dac);
        pcf8591_xuat_dac(dac);
        delay_ms(500);
        if (dac==0) dac=210;
        else         dac++;
    }
}
```

- d. Tiến hành biên dịch và nạp.
e. Quan sát kết quả: sau khi nạp xong thì ta nhìn thấy giá trị số hiển thị trên module led 7 đoạn tăng dần và trên led đơn cũng vậy. Hãy quan sát led màu xanh lá nằm bên phải IC PCF 8591 gần biến trở nút áo sẽ thấy cường độ sáng tăng dần. Khi đạt 255 thì led sáng nhất sau đó giá trị số về lại 210 thì led tắt. Điện áp tương tự ra bắt đầu từ giá trị 1 tương ứng 10mV nhưng phải dùng đồng hồ thì mới thấy.
f. Giải thích chương trình:

Bài tập 842. Hãy thêm vào chương trình 8-11 sao cho khi tăng lên đến 255 thì giá trị số giảm dần về 210 rồi lặp lại.

Lưu tên file là “BAI_842_DAC_PCF8591_UP_DW.C”

8.7 IC EEPROM AT24C256

8.7.1 Khảo sát IC nhớ EEPROM AT24C256

IC nhớ EEPROM AT24C256 có dung lượng 128Kbit hay 32Kbyte hay 32768 byte giao tiếp chuẩn nối tiếp 2 dây I2C.

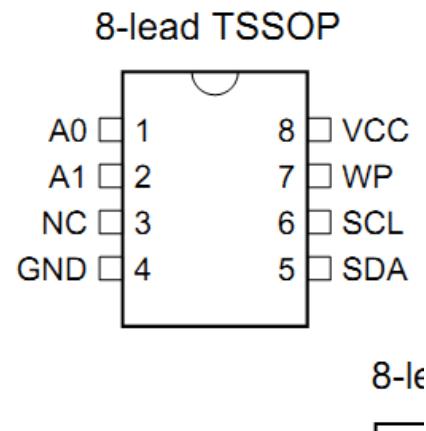
Có thể hoạt động truyền dữ liệu hai chiều với tốc độ 1MHz khi dùng nguồn 5V, 400kHz khi dùng nguồn 2,7V và 100kHz khi dùng nguồn 1,8V.

Cho phép ghi/đọc dữ liệu theo trang bộ nhớ 64 byte.

Cho phép ghi 100,000 lần, dữ liệu có thể lưu trữ lên đến 40 năm.

Sơ đồ chân và tên các chân như hình 8-30:

Pin Name	Function
A0–A1	Address Inputs
SDA	Serial Data
SCL	Serial Clock Input
WP	Write Protect
NC	No Connect



Hình 8-30. Sơ đồ chân và tên các chân của IC Eeprom AT24C256.

Sơ đồ khối bên trong của IC như hình 8-31.

Trong sơ đồ khối gồm các khối “START STOP LOGIC” có chức năng phát hiện các trạng thái Start và Stop của chuẩn truyền dữ liệu I2C.

Khối “DEVICE ADDRESS COMPARATOR” có chức năng so sánh địa chỉ từ thiết bị chủ gửi đến.

Khối “SERIAL CONTROL LOGIC” có chức năng điều khiển truyền dữ liệu nối tiếp.

Các khối xử lý địa chỉ và giải mã địa chỉ cùng với bộ nhớ eeprom.

Có 1 transistor có chức năng tắt mở trong quá trình không được truy xuất.

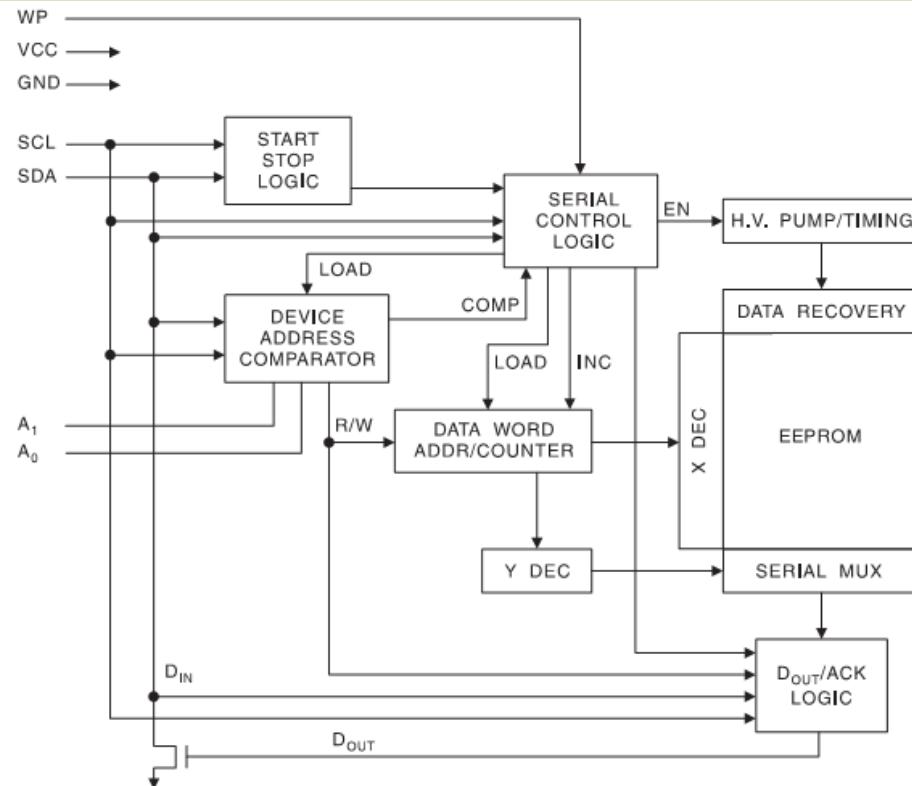
Chức năng các chân

Chức năng chân SCL và SDA theo chuẩn I2C.

Chân WP khi nối GND thì cho phép ghi, nếu nối Vcc thì không cho phép ghi. Nếu để hở thì bên trong có điện trở kéo xuống mức 0.

Hai chân A₀ và A₁ dùng để thiết lập địa chỉ cho IC nhớ khi giao tiếp nhiều IC nhớ.

- Nếu chỉ có 1 IC nhớ này có dung lượng 256Kbit thì 2 bit địa chỉ này sẽ là A₁A₀ = 00.
- Nếu ghép 2 IC nhớ để có dung lượng 512Kbit thì IC thứ nhất có 2 bit địa chỉ là A₁A₀ = 00, IC thứ hai có 2 bit địa chỉ là A₁A₀ = 01.
- Nếu ghép 4 IC nhớ để có dung lượng 1Mbit thì IC thứ nhất có 2 bit địa chỉ là A₁A₀ = 00, IC thứ hai có 2 bit địa chỉ là A₁A₀ = 01, IC thứ ba có 2 bit địa chỉ là A₁A₀ = 10, IC thứ tư có 2 bit địa chỉ là A₁A₀ = 11.

**Hình 8-31. Sơ đồ khái niệm của IC EEPROM AT24C256.****Tổ chức bộ nhớ**

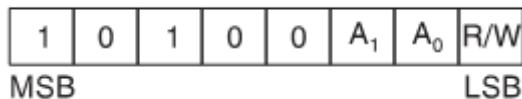
IC nhớ EEPROM AT24C256 có dung lượng 256Kbit được tổ chức thành 512 trang bộ nhớ, mỗi trang có dung lượng 64byte. Địa chỉ truy xuất ngẫu nhiên là 15 bit.

Hoạt động của 2 tín hiệu SCL và SDA

Theo chuẩn giao tiếp I2C như đã trình bày.

Địa chỉ của thiết bị

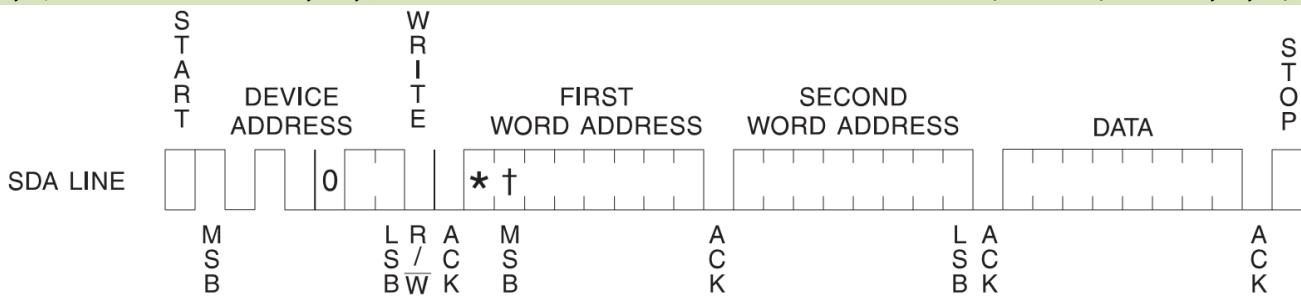
Địa chỉ như hình 8-32:

**Hình 8-32. Tổ chức địa chỉ của IC EEPROM AT24C256.**

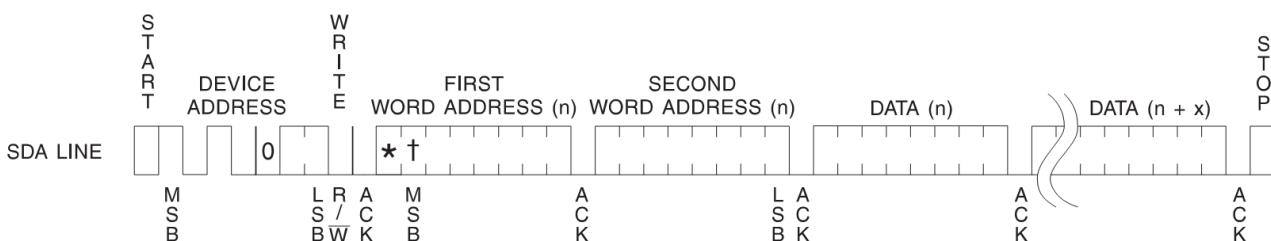
Có 7 bit địa chỉ như các thiết bị khác, bit LSB là bit lựa chọn chế độ đọc hay ghi, 2 bit địa chỉ A₁A₀ như đã trình bày ở trên. Kit thiết kế thì A₁A₀ = 00 nên địa chỉ ghi là 0xA0 và đọc là 0xA1.

Hoạt động ghi dữ liệu của IC EEPROM AT24C256

Có 2 chế độ ghi: ghi theo byte và ghi theo trang như các hình 8-33 và 8-34:

**Hình 8-33. Ghi theo byte của IC Eeprom AT24C256.**

Hoạt động ghi 1 byte giống như đã trình bày ở đầu chương, gồm các bước: thực hiện start, gởi địa chỉ thiết bị, gởi địa chỉ byte cao của ô nhớ cần ghi, gởi địa chỉ byte thấp của ô nhớ cần ghi và tiến hành gởi byte dữ liệu và thực hiện stop để kết thúc.



Notes: (* = DON'T CARE bit)

(† = DON'T CARE bit for the 128K)

Hình 8-34. Ghi theo trang của IC Eeprom AT24C256.

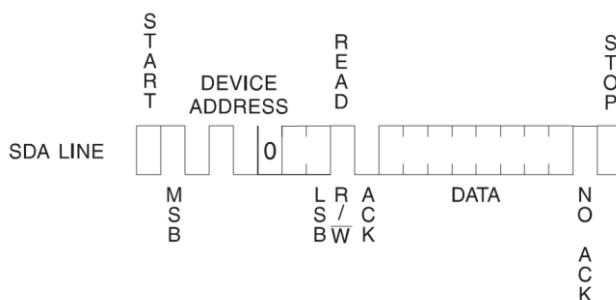
Hoạt động ghi 1 trang bộ nhớ gồm 64 byte gồm các bước: thực hiện start, gởi địa chỉ thiết bị, gởi địa chỉ byte cao của trang cần ghi, gởi địa chỉ byte thấp của trang cần ghi và tiến hành gởi byte dữ liệu thứ 0, byte thứ 1, ... cho đến byte thứ 63 và thực hiện stop để kết thúc.

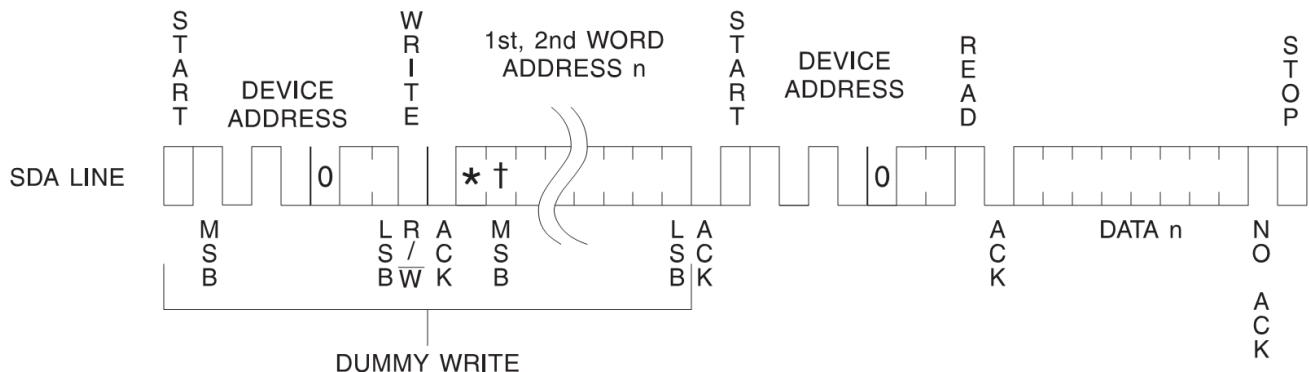
Ta chỉ có thể ghi ít hơn 63 chứ nhiều hơn thì nó sẽ cuộn tròn.

Bên trong IC nhớ sẽ tự động tăng 6 bit địa chỉ thấp để nhận đúng 64 byte.

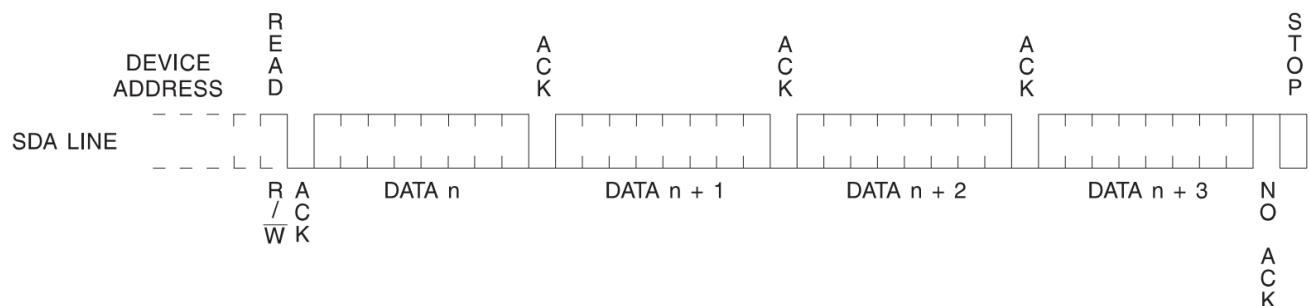
Khi dung lượng lớn và liên tục thì ghi theo trang sẽ nhanh hơn ghi theo byte.

Có 3 chế độ ghi: đọc byte hiện tại, đọc byte ngẫu nhiên và đọc theo trang như các hình sau:

**Hình 8-35. Đọc byte hiện tại của IC Eeprom AT24C256.**



Hình 8-36. Đọc byte ngẫu nhiên của IC Eeprom AT24C256.



Hình 8-37. Đọc nhiều byte liên tục của IC Eeprom AT24C256.

8.7.2 Mạch điện giao tiếp với IC nhớ EEPROM AT24C256

Vì điều khiển PIC18F4550 với bộ nhớ Eeprom AT24C256 như hình 8-9.

8.7.3 Các chương trình ghi đọc IC nhớ EEPROM AT24C256

Thực hành các bài ghi đọc bộ nhớ nối tiếp Eeprom AT24C256

Thư viện AT24C256. Chương trình thư viện điều khiển IC AT24C256 theo chuẩn I²C đã cho trong thư mục của bạn có tên là.

“TV_PICKIT2_SHIFT_AT24C256_I2C.c”

Bạn có thể sử dụng cho các bài thực hành và nên đọc hiểu.

```
#define addr_wr_24xxx 0xa0
#define addr_rd_24xxx 0xa1
void reset_24xxx()
{
    output_high(pin_b1);
    i2c_start();
    i2c_stop();
}
void khai_tao_ghi_24xxx(unsigned int16 addr)
{
    i2c_start();
    i2c_write(addr_wr_24xxx);
    i2c_write(addr>>8);
```

```

    i2c_write(addr);
}

void khai_tao_doc_24xxx(unsigned int16 addr)
{
    i2c_start();
    i2c_write(addr_wr_24xxx);
    i2c_write(addr>>8);
    i2c_write(addr);
    i2c_start();
    i2c_write(addr_rd_24xxx);
}
void ghi_1byte_24xxx(unsigned int16 addr,unsigned int8 data_1byte)
{
    i2c_start();
    i2c_write(addr_wr_24xxx);
    i2c_write(addr>>8);
    i2c_write(addr);
    i2c_write(data_1byte);
    i2c_stop();
}

unsigned int8 doc_1byte_24xxx(unsigned int16 addr)
{
    unsigned int8 data_1byte;
    i2c_start();
    i2c_write(addr_wr_24xxx);
    i2c_write(addr>>8);
    i2c_write(addr);
    i2c_start();
    i2c_write(addr_rd_24xxx);
    data_1byte = i2c_read(0);
    i2c_stop();
    return(data_1byte);
}

```

Bài mẫu 851. Chương trình ghi dữ liệu 64 byte bắt đầu từ con số 100 vào trang bộ nhớ thứ 0. Địa chỉ tăng từ 0 đến 63 và con số ghi vào cũng tăng từ 100 đến 163.

Sau khi ghi xong thì tiến hành đọc lại dữ liệu vừa hiển thị trên 8 led đơn và led 7 đoạn, địa chỉ cũng hiển thị trên 16 led đơn.

Lưu tên file là “BAI_851_AT24C256_WR_RD_PAGE.C”

- Mục đích: biết lập trình ghi dữ liệu vào và đọc dữ liệu ra bộ nhớ nối tiếp AT24C256.
- Lưu đồ: sinh viên hãy tự viết.
- Chương trình:

```

#include <tv_pickit2_shift_1.c>
#include <tv_pickit2_shift_at24c256_i2c.c>
unsigned int8 i, dl_doc,page;

void ghi_dulieu_pageX_vao_at24c256(unsigned int16 pageX)

```

```

{
    unsigned int16 addr_mem;
    unsigned int8 addr_l, addr_h, data_ghi;

    if (pagex>511) pagex=0;
    addr_mem = (pagex)*64;
    addr_l = addr_mem;
    addr_h = addr_mem>>8;

    khoi_tao_ghi_24xxx(addr_mem);
    for(i=0;i<64;i++)
    {
        data_ghi= i + 100;
        i2c_write(data_ghi);
        xuat_32led_don_4byte(addr_h,addr_l|i,0,data_ghi);
    }
    i2c_stop();
}

void doc_dulieu_pagex_tu_at24c256(unsigned int8 x, unsigned int16 pagex)
{
    unsigned int16 addr_mem;
    unsigned int8 addr_l, addr_h;

    if (pagex>511) pagex=0;
    addr_mem = (pagex)*64;
    addr_l = addr_mem;
    addr_h = addr_mem>>8;

    khoi_tao_doc_24xxx(addr_mem);
    for(i=0;i<63;i++)
    {
        dl_doc = i2c_read();
        xuat_4led_7doan_giaima_xoa_so0(dl_doc);
        xuat_32led_don_4byte(addr_h,addr_l|i,0,dl_doc);
        delay_ms(x);
    }
    dl_doc = i2c_read(0);
    i2c_stop();
    xuat_4led_7doan_giaima_xoa_so0(dl_doc);
    xuat_32led_don_4byte(addr_h,addr_l,0,dl_doc);
    delay_ms(x);
}
void main()
{
    set_up_port_ic_chot();
    reset_24xxx();
    page=0;
    ghi_dulieu_pagex_vao_at24c256(page);
    doc_dulieu_pagex_tu_at24c256(10,page);
    while(true)
    {
        doc_dulieu_pagex_tu_at24c256(200,page);
        delay_ms(500);
    }
}

```

}
}

- d. Tiến hành biên dịch và nạp.
- e. Quan sát kết quả: Sau khi nạp xong thì quá trình ghi sẽ thực hiện rất nhanh, sau đó đến quá trình đọc có thể điều chỉnh thời gian delay để dễ quan sát. Dữ liệu đọc hiển thị đúng bắt đầu từ 100 trên led 7 đoạn và dạng số nhị phân ở 8 led màu đỏ bên phải. Địa chỉ tương ứng với từng ô nhớ hiển thị ở 16 led đơn bên trái. Bạn có thể thay đổi trang bộ nhớ và dữ liệu ghi vào.
- f. Giải thích chương trình:

Bài tập 852. Hãy viết chương trình có chức năng sau:

Đếm sản phẩm hiển thị trên led 7 đoạn ở 2 led phải và có cài giới hạn đếm hiển thị ở 2 led trái bằng 2 nút nhấn UP và DW.

Giá trị cài lưu vào bộ nhớ AT24C256 để khi mất điện thì nó vẫn nhớ giá trị đã cài.

Giới hạn từ 00 đến 99.

Lưu tên file là “BAI_852_AT24C256_T0_4LED.C”

Bài tập 853. Giống bài 852 nhưng hiển thị trên LCD.

Hàng 1 hiển thị giá trị đếm. Hàng 2 hiển thị giá trị cài.

Lưu tên file là “BAI_853_AT24C256_T0_LCD.C”

Bài tập 854. Hãy viết thêm vào bài 853 chức năng sau:

Dùng thêm 1 ô nhớ để lưu số chu kỳ đã đếm được và hiển thị ở hàng 3.

Có thêm nút CLR để xóa chu kỳ đếm về 0 nếu cần xóa.

Lưu tên file là “BAI_854_AT24C256_T0_LCD_CHUKY_CLR.C”

Bài tập 855. Hãy tìm hiểu bộ nhớ Eeprom của PIC và viết lại bài 854 dùng bộ nhớ eeprom nội của PIC.

Lưu tên file là “BAI_855_EPROM_PIC_T0_LCD_CHUKY_CLR.C”

Chương 9: THỰC HÀNH

MODULE 6 – ĐỘNG CƠ BUỚC, ĐỘNG CƠ DC VÀ ENCODER, PWM, ĐIỀU KHIỂN PID

9.1 ĐIỀU KHIỂN ĐỘNG CƠ BƯỚC

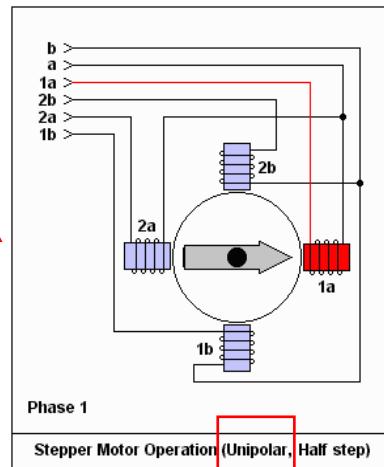
9.1.1 Giới thiệu động cơ bước

Động cơ bước có hình ảnh minh họa như hình 9-1 và sơ đồ nguyên lý như hình 9-2.



Hình 9-1. Động cơ bước.

12V, điện trở cuộn dây $\rightarrow I = U/R$
 \rightarrow linh kiện cấp dòng cho đ.cơ
 góc quay: 1.8 độ/1 bước
 $\rightarrow 1 \text{ vòng} = 360/1.8 = 200 \text{ bước}$



bipolar - lưỡng cực
 4 dây

unipolar - đơn cực
 5 dây, 6 dây (a,b nối sǎn bên trong)

Hình 9-2. Sơ đồ nguyên lý động cơ bước.

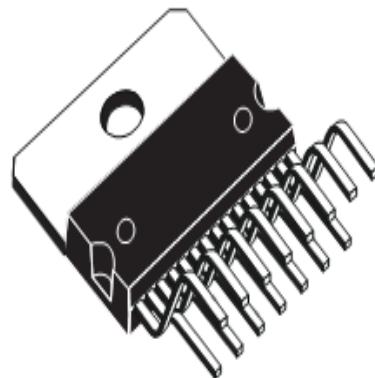
Động cơ bước có nhiều loại và nhiều cách điều khiển tùy thuộc vào các ứng dụng cụ thể, trong kit thực hành có trang bị 1 motor bước để các bạn thực hành.

Trong hình 9-2, động cơ bước có 6 đầu dây: 1a, 2a và 1b, 2b và 2 đầu dây chung là a và b.

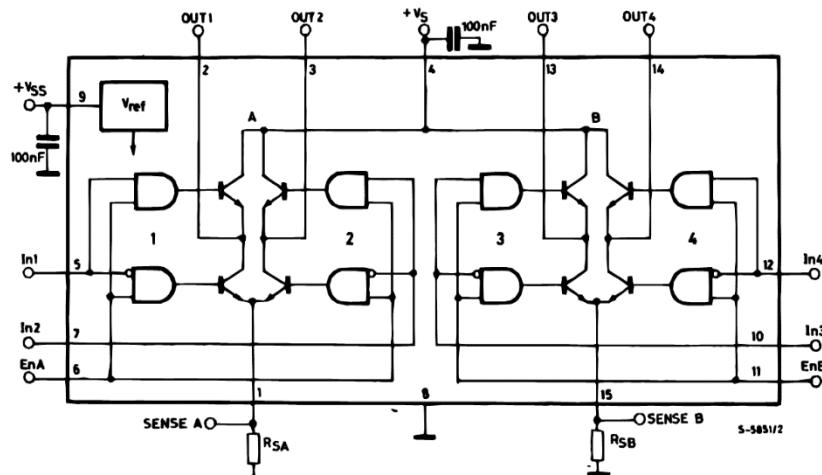
Động cơ bước có nhiều ứng dụng trong thực tế, ví dụ như ô đĩa máy tính sử dụng động cơ bước để di chuyển đầu đọc từ rãnh này sang rãnh kia, phạm vi di chuyển ngắn, còn động cơ DC quay với tốc độ cao thì rất khó điều khiển. Trong ô đĩa, động cơ DC quay với tốc độ 5400 vòng trên 1 phút để quay đĩa.

9.1.2 IC giao tiếp công suất điều khiển L298

Các động cơ bước có nhiều chủng loại tùy thuộc vào ứng dụng, trong bộ thí nghiệm sử dụng động cơ bước có dòng là 1A. Vì điều khiển không thể trực tiếp điều khiển được mà phải dùng IC chuyên dùng để điều khiển là L298. Hình ảnh IC và sơ đồ nguyên lý bên trong như các hình sau:



Hình 9-3. Hình dạng IC L298.

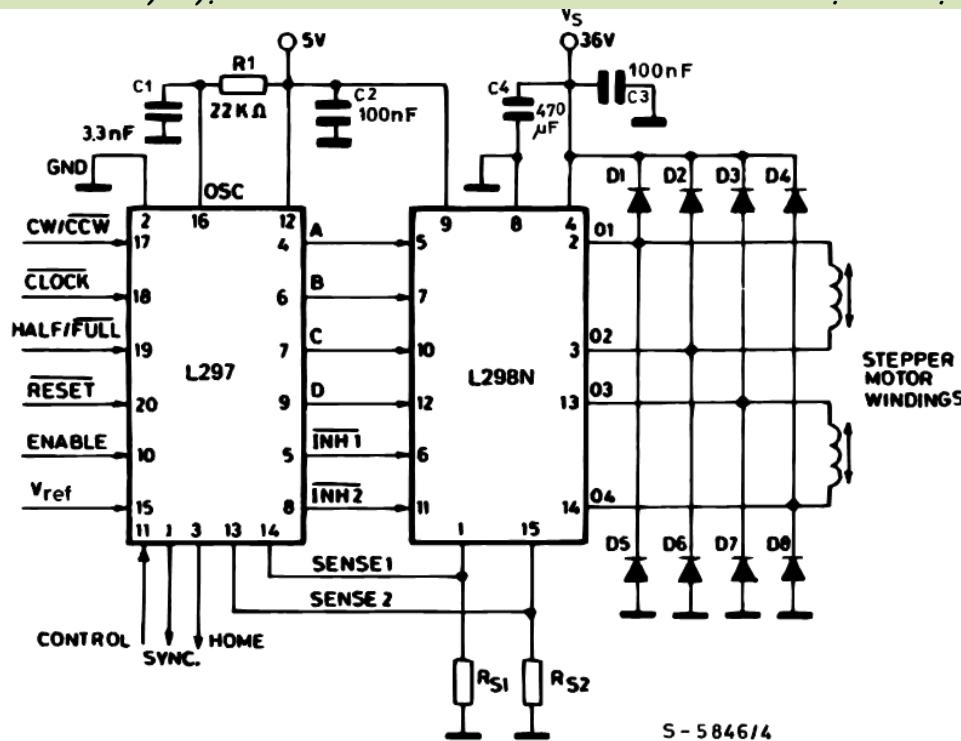


Hình 9-4. Sơ đồ nguyên lý IC L298.

Bên trong IC có các công nghệ logic và 2 cặp cầu H chuyên dùng để điều khiển động cơ DC và có thể điều khiển được 2 động cơ DC hoặc 1 động cơ bước.

IC L298 có 2 điện áp: điện áp cấp cho các transistor để điều khiển động cơ có thể hoạt động tối đa lên đến 46V. Khi sử dụng động cơ có nguồn cung cấp bao nhiêu thì ta cần đúng bấy nhiêu, ví dụ động cơ dùng nguồn 24V thì ta cần nguồn 24V. Điện áp thứ hai là cấp cho các công nghệ TTL hoạt động bình thường là 5V DC.

Sơ đồ mạch điều khiển 1 động cơ bước như hình sau:

**Hình 9-5. IC L298 điều khiển 1 động cơ bước.**

Trong hình có 2 IC L298 và L297 dùng để điều khiển động cơ bước. Bạn chỉ quan sát phần IC L298 có 4 ngõ ra điều khiển 4 đầu dây của động cơ bước. 8 diode dùng để bảo vệ các transistor giải phóng năng lượng của cuộn dây về nguồn.

IC L297 chuyên để điều khiển động cơ bước nhưng trong bộ thí nghiệm không sử dụng nên không đề cập.

9.1.3 Mạch giao tiếp điều khiển động cơ bước

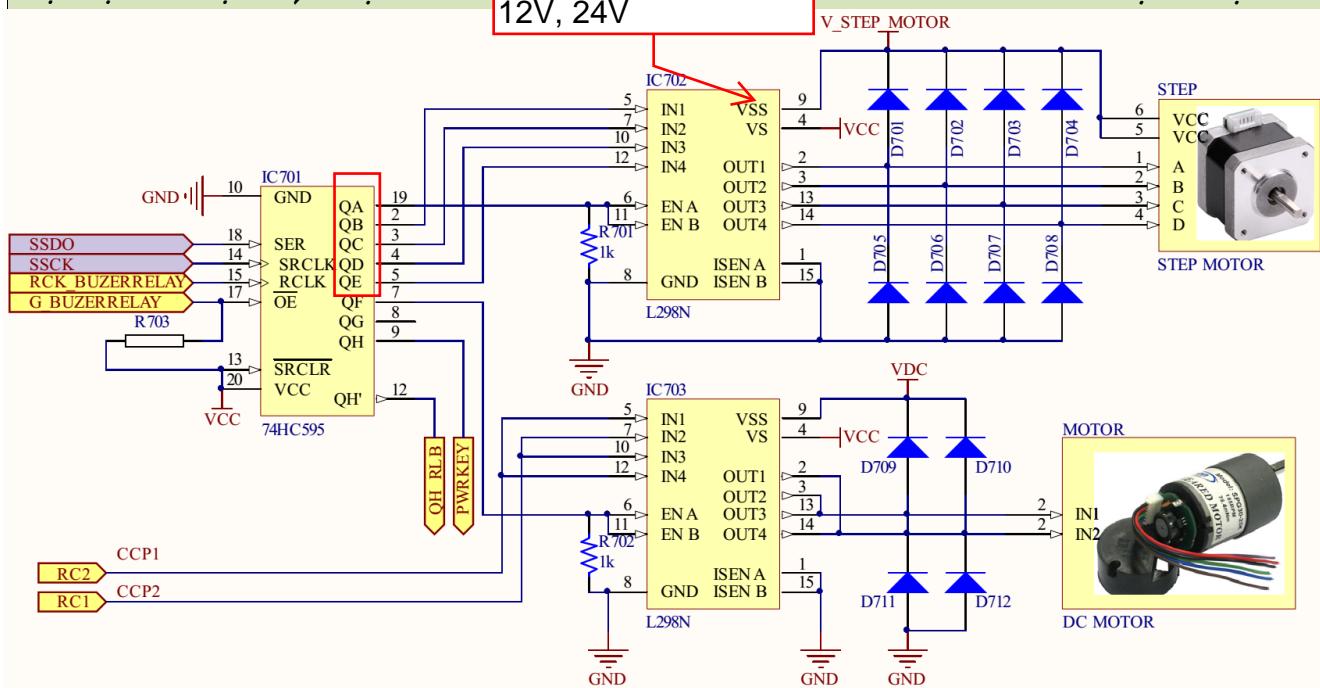
Sơ đồ mạch giao tiếp với động cơ bước và động cơ DC như hình 9-6.

Trong mạch sử dụng 1 IC thanh ghi 74HC595 để mở rộng port điều khiển động cơ bước và DC, 1 thanh ghi dịch điều khiển 2 relay, 2 triac và 1 buzzer.

Phản vi điều khiển PIC giao tiếp với IC thanh ghi dịch

Có 4 tín hiệu điều khiển là:

- SDO dùng để dịch dữ liệu nối tiếp.
- SCK dùng để cấp xung clock.
- RCK_RLB dùng điều khiển nạp dữ liệu song song từ bên trong IC thanh ghi dịch ra bên ngoài để điều khiển.
- G_RLB dùng để cho phép xuất dữ liệu.



Hình 9-6. Mạch giao tiếp động cơ bước và động cơ DC.

Phản IC thanh ghi dịch giao tiếp với động cơ

Năm tín hiệu từ QA đến QE nối với các tín hiệu tương ứng là ENAX, IN1X, IN2X, IN3X, IN4X của IC702 – L298. Bốn ngõ ra OUT1, OUT2, OUT3, OUT4 sẽ nối với 4 đầu dây CA, CB, CC, CD của động cơ bước, hai đầu dây chung nối lên nguồn cấp cho động cơ. Có 8 diode dùng để bảo vệ các transistor bên trong IC L298.

9.1.4 Mã điều khiển động cơ bước

Dữ liệu điều khiển động cơ bước từ vi điều khiển sẽ gửi ra thanh ghi dịch, dữ liệu từ thanh ghi dịch sẽ đưa đến IC công suất L298 để điều khiển động cơ bước.

Có 2 loại dữ liệu điều khiển động cơ bước: **điều khiển bước đú (full step)** và **nửa bước (half step)** được trình bày trong 2 bảng sau.

Bảng 9-1. Các trạng thái điều khiển động cơ bước theo bước đú.

	SỐ NHỊ PHÂN 8 BIT								SỐ HEX	TRẠNG THÁI ĐỘNG CƠ
TT	QH	QG	QF	QE	QD	QC	QB	QA		
1				X	X	X	X	0		Không cho phép
2	0	0	0	1	1	1	0	1	0x1D	Động cơ quay 1 bước
3	0	0	0	1	0	1	1	1	0x17	4 cuộn dây, có máy đầu cuộn thay đổi ?
4	0	0	0	1	1	0	1	1	0x1B	2 c.dây
5	0	0	0	0	1	1	1	1	0x0F	

Khi muốn điều khiển động cơ bước quay thì bit ENA phải bằng 1, 4 bit còn lại sẽ có giá trị như trong bảng cột số hex.

Nếu gởi 4 mã theo thứ tự từ trên xuống thì động cơ bước sẽ quay theo chiều kim đồng hồ, nếu gởi ngược lại thứ tự từ dưới lên thì động cơ sẽ quay ngược chiều kim đồng hồ.

Bảng 9-2. Các trạng thái điều khiển động cơ bước dạng nửa bước.

	SỐ NHỊ PHÂN 8 BIT								SỐ HEX	TRẠNG THÁI ĐỘNG CƠ
	QH	QG	QF	QE	QD	QC	QB	QA		
TT				IN4	IN3	IN2	IN1	ENA		
1				X	X	X	X	0		Không cho phép
2	0	0	0	1	1	1	0	1	0x1D	Động cơ quay ½ bước
3	0	0	0	1	0	1	0	1	0x15	1 c.dây thay đổi
4	0	0	0	1	0	1	1	1	0x17	
5	0	0	0	1	0	0	1	1	0x13	
6	0	0	0	0	0	0	1	1	0x03	
7	0	0	0	0	1	0	1	1	0x0B	
8	0	0	0	0	1	0	0	1	0x09	
9	0	0	0	0	1	1	0	1	0x0D	

9.1.5 Các biến và hàm điều khiển Relay, Triac, Buzzer, động cơ

Như đã trình bày module này có 2 byte dữ liệu để xuất ra điều khiển nhiều đối tượng khác nhau.

Trong thư viện đã định nghĩa sẵn 1 biến dữ liệu 16 bit và sau đó định nghĩa các bit điều khiển cho từng đối tượng như sau:

```
unsigned int16 rbdc;
#bit step_motor_enable = rbdc.0 //1=ena, 0=dis
#bit step_motor_in1 = rbdc.1
#bit step_motor_in2 = rbdc.2
#bit step_motor_in3 = rbdc.3
#bit step_motor_in4 = rbdc.4
#bit dc_enable = rbdc.5 //1=ena, 0=dis
#bit pwrkey = rbdc.6 //khoi tao sim900

#bit buzzer = rbdc.8 //1=on, 0=off
#bit triac_1 = rbdc.9 //1=on, 0=off
#bit triac_2 = rbdc.10 //1=on, 0=off
#bit relay_1 = rbdc.11 //0=on, 1=off
#bit relay_2 = rbdc.12 //0=on, 1=off
```

phản cứng

Hàm thứ 901: xuất 2 byte ra module relay, triac, buzzer, động cơ:

```

void xuat_buzzer_relay()
{
    unsigned int8 rbdc1,rbdc2;
    rbdc1 = rbdc;
    rbdc2 = rbdc>>8;
    xuat_1byte(rbdc2);
    xuat_1byte(rbdc1);

    mo_ic_74573_b_thong_dl();
    output_high(rck_buzerelay);
    output_low(rck_buzerelay);
    mo_relay_buzzer_dc;
    chot_ic_74573_b_goi_du_lieu;
}

```

Hàm này tách dữ liệu 16 bit thành 2 byte rồi xuất đi.

Hàm thứ 902: buzzer on: buzzer sẽ kêu

```

void buzzer_on()
{
    buzzer=1;
    xuat_buzzer_relay();
}

```

Hàm này làm bit buzzer bằng 1 rồi gọi hàm xuất dữ liệu ra thanh ghi dịch để điều khiển.

Hàm thứ 903: buzzer off: tắt buzzer

```

void buzzer_off()
{
    buzzer=0;
    xuat_buzzer_relay();
}

```

Hàm này làm bit buzzer bằng 0 rồi gọi hàm xuất dữ liệu ra thanh ghi dịch để điều khiển.

Hàm thứ 910: triac 1 on: đèn tròn AC sáng

```

void triac_1_on()
{
    triac_1=1;
    xuat_buzzer_relay();
}

```

Hàm thứ 911: triac 1 off: đèn tròn AC tắt

```

void triac_1_off()
{
    triac_1=0;
    xuat_buzzer_relay();
}

```

9.1.6 Các bài thực hành điều khiển động cơ bước

Thư viện động cơ bước. Chương trình thư viện của động cơ bước đã cho trong thư mục của bạn có tên là “TV_PICKIT2_SHIFT_STEP_MOTOR.C”.

Bạn có thể sử dụng cho các bài thực hành và nên đọc hiểu.

a. Chương trình:

```

unsigned int8 stepmotor_fullstep[] = {0x1d,0x17,0x1b,0x0f};
unsigned int8 stepmotor_halfstep[] = {0x1d,0x15,0x17,0x13,0x03,0x0b,0x09,0x0d};
unsigned int8 stepmotor_i=0, stepmotor_delay;
int1 stepmotor_tn=0,stepmotor_onoff=0;
void step_motor_quay_thuan_fs()
{
    rbdc=rbdc & 0xffe0;
    rbdc=rbdc | stepmotor_fullstep[stepmotor_i];
    xuat_buzzer_relay();
    stepmotor_i++;
    stepmotor_i = stepmotor_i & 0x03;
}
void step_motor_quay_nghich_fs()
{
    rbdc=rbdc & 0xffe0;
    rbdc=rbdc | stepmotor_fullstep[stepmotor_i];
    xuat_buzzer_relay();
    stepmotor_i--;
    stepmotor_i = stepmotor_i & 0x03;
}
void motor_step_run_fs()
{
    if (stepmotor_tn) step_motor_quay_thuan_fs();
    else step_motor_quay_nghich_fs();
}

void step_motor_quay_thuan_hs()
{
    rbdc=rbdc & 0xffe0;
    rbdc=rbdc | stepmotor_halfstep[stepmotor_i];
    xuat_buzzer_relay();
    stepmotor_i++;
    stepmotor_i = stepmotor_i & 0x07;
}
void step_motor_quay_nghich_hs()
{
    rbdc=rbdc & 0xffe0;
    rbdc=rbdc | stepmotor_halfstep[stepmotor_i];
    xuat_buzzer_relay();
    stepmotor_i--;
    stepmotor_i = stepmotor_i & 0x07;
}
void motor_step_run_hs()
{
    if (stepmotor_tn) step_motor_quay_thuan_hs();
    else step_motor_quay_nghich_hs();
}

```

b. Giải thích thông qua các câu hỏi:

Câu 1: Hãy cho biết lệnh “**RBDC=RBDC & 0xFFE0;**” có chức năng gì? Cho ví dụ minh họa?

Câu 2: Hãy cho biết lệnh “**RBDC=RBDC | STEPMOTOR_HALFSTEP[STEPMOTOR_I];**” có chức năng gì? Cho ví dụ minh họa?

Câu 3: Hãy cho biết lệnh “**STEPMOTOR_I = STEPMOTOR_I & 0X03;**” có chức năng gì? Cho ví dụ minh họa?

Câu 4: Hãy cho biết lệnh sự khác nhau của 2 hàm “**STEP_MOTOR_QUAY_THUAN_FS();**” và “**STEP_MOTOR_QUAY_NGHICH_FS();**”.

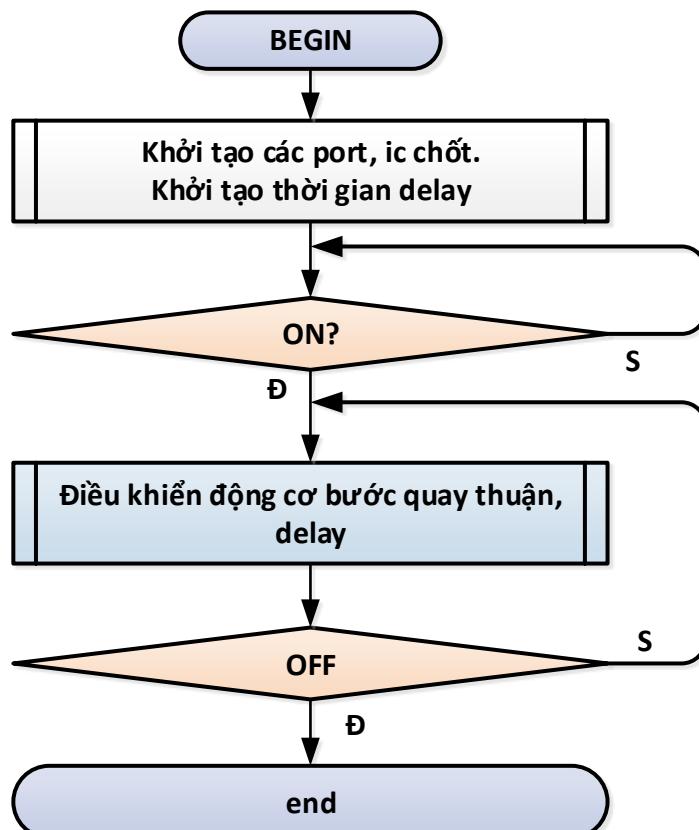
Câu 5: Hãy cho biết lệnh sự khác nhau của 2 hàm “**STEP_MOTOR_QUAY_THUAN_HS();**” và “**STEP_MOTOR_QUAY_THUAN_HS();**”.

Bài mẫu 901. Chương trình điều khiển động cơ bước có 2 nút điều khiển: ON, OFF.

Khi nhấn ON thì động cơ quay theo chiều kim đồng hồ, khi nhấn OFF thì ngừng.

Lưu tên file là “BAI_901_STEP_ON_OFF”

- a. Mục đích: Biết lập trình điều khiển động cơ bước bằng 2 nút nhấn ON và OFF.
- b. Lưu đồ:



Hình 9-7. Lưu đồ điều khiển động cơ bước quay thuận bằng 2 nút ON, OFF.

- c. Chương trình:

```
#include <tv_pickit2_shift_1.c>
#include <tv_pickit2_shift_step_motor.c>
```

```

void main()
{
    set_up_port_ic_chot();
    steppmotor_delay=20;
    steppmotor_onoff=1; //de khai canh bao
    while(true)
    {
        while(input(on));
        do
        {
            step_motor_quay_thuan_fs();
            delay_ms(steppmotor_delay);
        }
        while (input(off));
    }
}

```

dùng biến trạng thái, hàm chống dội phím

904 SO_BUOC++;

tốc độ động cơ bước phụ thuộc tốc độ xuất dữ liệu điều khiển

- d. Tiến hành biên dịch và nạp.
- e. Quan sát kết quả: sau khi nạp xong thì nhấn nút ON động cơ sẽ quay thuận, nhấn OFF sẽ ngừng động cơ. Có thể điều chỉnh thời gian delay giữa 2 lần gởi để thay đổi tốc độ động cơ, nếu thời gian ngắn quá thì động cơ có thể đứng yên vì không đáp ứng được.
- f. Hãy giải thích chương trình và đặt các câu hỏi nếu chưa hiểu.

Bài tập 902. Hãy hiệu chỉnh bài 601 sau cho khi nhấn ON thì động cơ **quay ngược** chiều kim đồng hồ, nhấn OFF thì ngừng. Thứ tự dữ liệu điều khiển động cơ bước giữ nguyên.

Lưu tên file là “BAI_902_STEP_ON_OFF_QN”

Bài tập 903. Hãy viết chương trình điều khiển động cơ bước có 3 nút điều khiển: ON, OFF, INV.

Khi nhấn ON thì động cơ quay theo chiều kim đồng hồ.

Khi nhấn INV thì động cơ đảo chiều đang quay, nếu động cơ ngừng thì phím này không có tác dụng.

Khi nhấn OFF thì động cơ ngừng.

Lưu tên file là “BAI_903_STEP_ON_OFF_INV”

- a. Mục đích: Biết lập trình điều khiển động cơ bước bằng 3 nút nhấn ON, OFF và INV.
- b. Lưu đồ: sinh viên hãy tự viết.
- c. Chương trình:

```

#include <tv_pickit2_shift_1.c>
#include <tv_pickit2_shift_step_motor.c>
void phim_inv()
{
}

```

```

if (!input(inv))
{
    delay_ms(20);
    {
        if (!input(inv))
        {
            stepmotor_tn=~stepmotor_tn;
            do
            {
                motor_step_run_fs();
                delay_ms(stepmotor_delay);
            }
            while(!input(inv));
        }
    }
}
void main()
{
    set_up_port_ic_chot();
    stepmotor_delay=5;
    stepmotor_onoff=1; //de khai canh bao
    while(true)
    {
        while(input(on)); //cho nhan on
        do
        {
            motor_step_run_fs();
            delay_ms(stepmotor_delay);
            phim_inv();
        }
        while (input(off));
    }
}

```

so sánh biến trạng thái và điều khiển
--> nên đặt phần ct dk ở ngoài while(true)
if(stepmotor_tn==1)
{ }
else
{ }

viết lại thành hàm con
chống dội phím
--> dùng biến trạng thái

- d. Tiến hành biên dịch và nạp.
- e. Quan sát kết quả: sau khi nạp xong thì nhấn nút ON động cơ sẽ quay thuận, nhấn INV thì động cơ đảo chiều, nhấn OFF sẽ ngừng động cơ.
- f. Hãy giải thích chương trình và đặt các câu hỏi nếu chưa hiểu.
- g. Câu hỏi: tại sao trong chương trình kiểm tra phím INV chờ buông phím lại gọi hai hàm trong vòng do while? Nếu viết giống như bài 323 thì hiện tượng gì xảy ra?

Bài tập 904. Hãy viết chương trình điều khiển động cơ bước giống bài 903 nhưng thêm phần **hiển thị số bước** trên 4 led 7 đoạn. Khi nhấn nút đảo chiều động cơ thì xóa luôn số bước đang hiển thị để bắt đầu lại từ đầu.

Lưu tên file là “BAI_904_STEP_ON_OFF_INV_HT_BUOC”

Làm sao để biết động cơ bước quay 1 vòng thì có bao nhiêu bước?

Bài tập 905. Hãy viết chương trình điều khiển động cơ bước giống bài 904 nhưng 3 led 7 đoạn bên phải hiển thị số bước của 1 vòng, led thứ 4 còn lại hiển thị số vòng.

Lưu tên file là “BAI_905_STEP_ON_OFF_INV_HT_BUOC_VONG”

Bài tập 906. Hãy viết chương trình điều khiển động cơ bước có thể thay đổi tốc độ bằng 3 nút điều khiển: STOP, UP, DW.

Khi nhấn UP thì động cơ quay theo chiều kim đồng hồ tăng tốc.

Khi nhấn DW thì động cơ quay theo chiều kim đồng hồ giảm tốc.

Có 3 led 7 đoạn hiển thị số bước, led thứ tư hiển thị số cấp từ 0 đến 9. Cấp 0 thì ngừng. Cấp 1 đến cấp 9: tương ứng với các khoảng thời gian delay: 45, 40, 35, 30, 25, 20, 15, 10, 5 ms.

Khi nhấn STOP thì động cơ ngừng.

Lưu tên file là “BAI_906_STEP_STOP_UP_DW”

Bài tập 907. Hãy viết thêm vào bài 906 phím có chức năng đảo chiều INV

Lưu tên file là “BAI_907_STEP_STOP_UP_DW_INV”

Bài tập 908. Hãy viết chương trình điều khiển động cơ quay theo số vòng được đặt trước dùng bàn **phím ma trận**, dùng các phím số từ **0 đến 9** để nhập số vòng, để đơn giản chỉ nhập 1 con số hiển thị ở 1 led 7 đoạn.

Nhấn phím D thì động cơ quay thuận và số vòng giảm dần đến 0 thì ngừng.

Lưu tên file là “BAI_908_STEP_KEY_44”

Bài tập 909. Hãy hiệu chỉnh lại bài 606 nhưng điều khiển động cơ quay nữa bước.

Lưu tên file là “BAI_909_STEP_STOP_UP_DW_INV_HS”

9.2 ĐIỀU KHIỂN ĐỘNG CƠ DC, ENCODER

9.2.1 Giới thiệu động cơ DC, ENCODER

Động cơ DC và encoder có hình ảnh như hình 9-8 và 9-9:

BT1:

904 --> QUAY THUẬN **50 BƯỚC**
--> QUAY NGHỊCH **25 BƯỚC**
RỒI **DỪNG LUÔN**, CHỜ NHẤN
PHÍM RESET, HOẶC PHÍM
KH_DONG

BT2: (B.GIẢNG - 423,625)

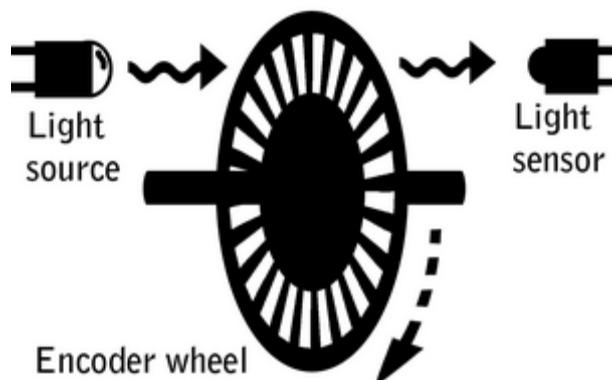
Tương tự BT1, **cài đặt** số bước quay thuận, còn quay ngược là bằng 1/2 số bước quay thuận.

BT3: GHÉP

DS18B20, DS1307, DK DC
BƯỚC, nút DAO_CHIEU



Hình 9-8 Động cơ DC trong BTN.



Hình 9-9. Encoder gắn sau động cơ.

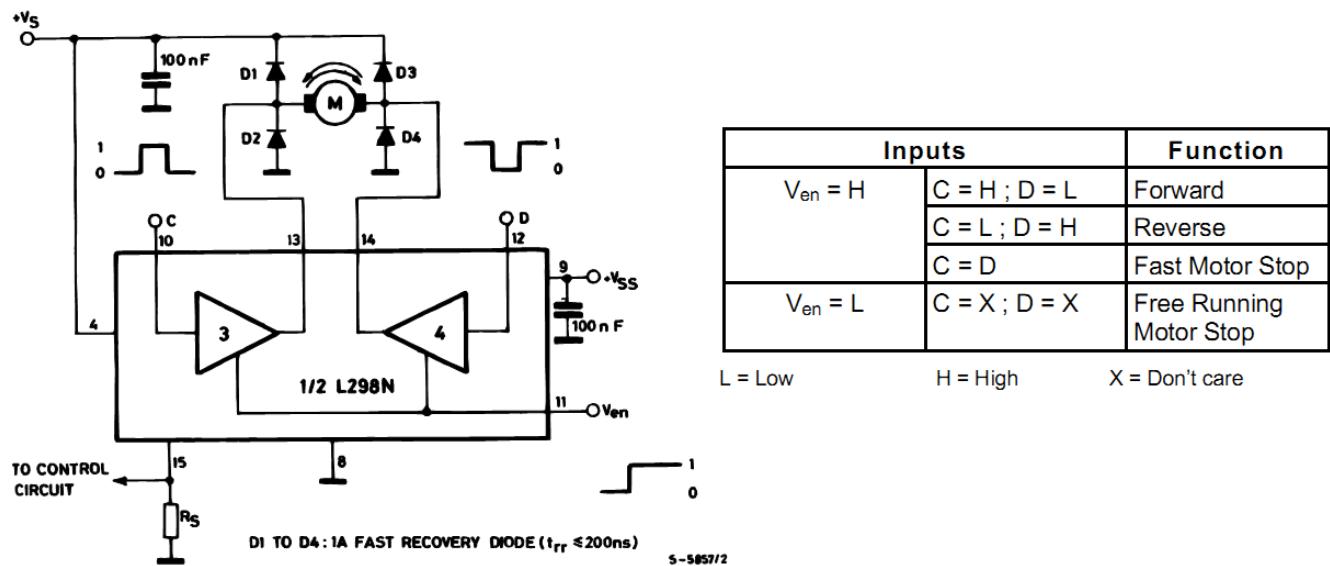
Động cơ DC có nhiều loại. Trong kit thực hành có trang bị 1 động cơ DC dùng nguồn 24V, dòng 1A, có encoder gắn vào trực động cơ tạo ra 45 cho 1 vòng quay để các bạn thực hành.

Loại encoder trong BTN này đơn giản cho các ứng dụng không cần xác định chiều, loại encoder có 2 đầu dây ra A và B dùng để xác định chiều của động cơ, A lên mức logic 1 nhanh hơn B thì động cơ quay thuận, ngược lại khi đảo chiều thì B lên mức 1 nhanh hơn A.

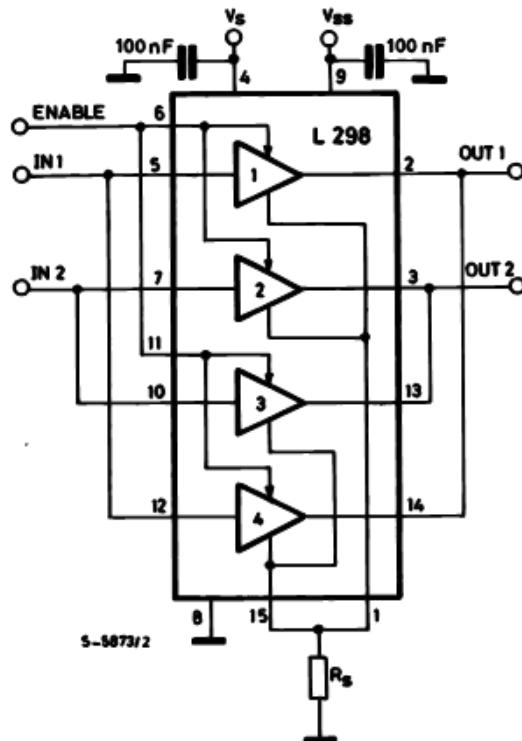
Sơ đồ mạch điều khiển 1 động cơ dùng L298 và các trạng thái điều khiển như hình 9-10.

Hãy quan sát hình và bảng trạng thái, bạn sẽ biết được nguyên lý điều khiển động cơ DC.

Nếu chỉ dùng 1 IC L298 điều khiển 1 động cơ DC thì dư 1 cầu H. Để tránh lãng phí và tăng khả năng cấp dòng ta sử dụng 2 cầu H mắc song song để điều khiển 1 động cơ DC như hình 9-11.



Hình 9-10. IC L298 điều khiển 1 động cơ DC dùng 1 cầu H.



Hình 9-11. IC L298 điều khiển 1 động cơ DC dùng 2 cầu H.

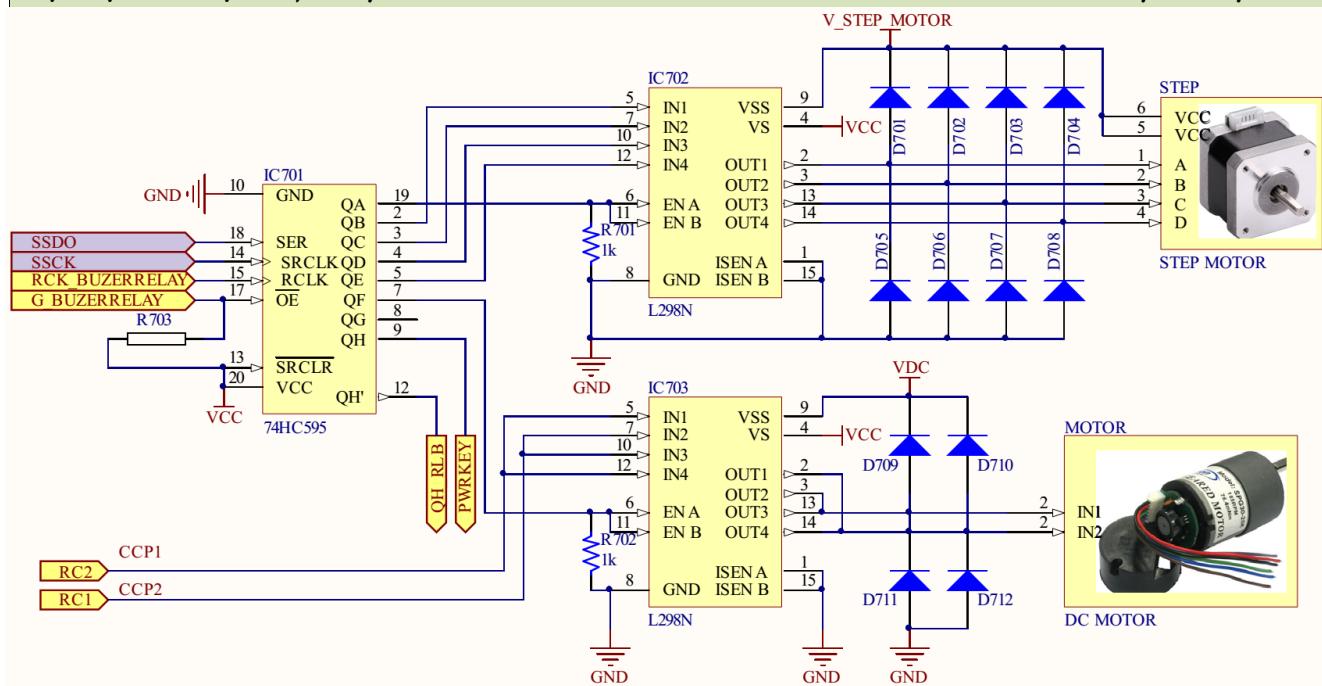
Trong bộ thực hành thì dùng theo cách này.

9.2.2 Mạch giao tiếp điều khiển động cơ DC

Sơ đồ mạch giao tiếp với động cơ DC như hình 9-12.

Trong sơ đồ mạch sử dụng IC703 – L298 để điều khiển động cơ DC qua IC trung gian mở rộng port IC701 – 74HC595D.

Tín hiệu QF nối với tín hiệu ENA của IC703 – L298.

**Hình 9-12. Mạch giao tiếp động cơ bước và động cơ DC.**

IC L298 có thể điều khiển được 2 động cơ DC nhưng kit thực hành chỉ có 1 động cơ nên 2 mạch điều khiển động cơ được nối song song để tăng khả năng chịu dòng. Trong sơ đồ mạch thì 2 cặp ngõ vào input1, input4 và input2, input3 nối với nhau. Hai cặp tín hiệu ngõ vào này sẽ nối trực tiếp với 2 chân RC1/CCP2 và RC2/CCP1 của vi điều khiển PIC.

Tương tự, ngõ ra cũng nối với nhau, chỉ còn 2 ngõ ra nối với động cơ DC.

9.2.3 Dữ liệu điều khiển động cơ DC

Dữ liệu điều khiển động cơ DC từ vi điều khiển sẽ gửi ra thanh ghi dịch, dữ liệu từ thanh ghi dịch sẽ đưa đến IC công suất L298 để điều khiển tín hiệu cho phép/cảm IC L298.

Bảng 9-3. Các trạng thái điều khiển động cơ DC.

TT	ENA	INPUT1	INPUT2	TRẠNG THÁI ĐỘNG CƠ
1	0	X	X	NGỪNG
2	1	0	0	
3	1	1	1	
4	1	1	0	QUAY THUẬN
5	1	0	1	QUAY NGHỊCH

9.2.4 Các bài thực hành điều khiển động cơ DC

Thư viện động cơ DC. Để tránh lặp lại, ta viết chương trình thư viện cho động cơ DC như sau.

Lưu tên file là “TV_PICKIT2_SHIFT_DC_MOTOR.C”

a. Chương trình:

```

int1 dcmotor_tn = 0, dcmotor_onoff=0;
void dc_motor_enable()
{
    dc_enable=1;
    xuat_buzzer_relay();
}
void dc_motor_quay_thuan()
{
    output_high(pin_c1);
    output_low(pin_c2);
}
void dc_motor_quay_nghich()
{
    output_low(pin_c1);
    output_high(pin_c2);
}
void dc_motor_stop()
{
    output_low(pin_c1);
    output_low(pin_c2);
}
void dc_motor_control()
{
    if (dcmotor_onoff==1)
    {
        if (dcmotor_tn)      dc_motor_quay_thuan();
        else                 dc_motor_quay_nghich();
    }
    else                  dc_motor_stop();
}

```

b. Giải thích thông qua các câu hỏi:

Câu 1: Hãy cho biết lệnh “**DC_ENABLE=1**;” có chức năng gì hay điều khiển cái gì.

Câu 2: Các lệnh trong hàm “**DC_MOTOR_QUAY_THUAN()**” có chức năng gì, điều khiển cái gì.

Câu 3: Các lệnh trong hàm “**DC_MOTOR_QUAY_NGHICH()**” có chức năng gì, điều khiển cái gì.

Câu 4: Các lệnh trong hàm “**DC_MOTOR_STOP()**” có chức năng gì, điều khiển cái gì.

Câu 5: Các lệnh trong hàm “**DC_MOTOR_CONTROL**” có chức năng gì, điều khiển cái gì.

Bài mẫu 911. Chương trình điều khiển động cơ DC có 2 nút điều khiển: ON, OFF.

Khi nhấn ON thì động cơ quay theo chiều kim đồng hồ, khi nhấn OFF thì ngừng.

Có 8 led sáng cho biết động cơ quay, tắt thì động cơ ngừng.

Lưu tên file là “**BAI_911_DC_MOTOR_ON_OFF**”

- Mục đích: biết điều khiển động cơ DC bằng 2 nút nhấn ON và OFF.
- Lưu đồ: sinh viên hãy tự viết.
- Chương trình:

```
#include <tv_pickit2_shift_1.c>
#include <tv_pickit2_shift_dc_motor.c>
void main()
{
    set_up_port_ic_chot();
    dc_motor_enable();
    dc_motor_stop();
    while(true)
    {
        while(input(on));
        {
            dc_motor_quay_thuan();
            xuat_32led_don_1byte(0xff);
        }
        while(input(off));
        {
            dc_motor_stop();
            xuat_32led_don_1byte(0);
        }
    }
}
```

- d. Tiến hành biên dịch và nạp.
- e. Quan sát kết quả: Sau khi nạp xong thì nhấn nút ON động cơ sẽ quay thuận, nhấn OFF sẽ ngừng động cơ. Trong chương trình có gởi dữ liệu điều khiển động cơ ra 32 led đơn để bạn có thể nhìn thấy. Động cơ chạy với tốc độ cực đại.
- f. Giải thích chương trình: bạn hãy giải thích.

Bài mẫu 912. Chương trình điều khiển động cơ có 3 nút điều khiển: ON, OFF, INV. Khi nhấn INV thì đảo chiều động cơ đang quay, trình tự thực hiện: tiến hành kiểm tra động cơ nếu đang quay thì ngừng 1 giây sau đó mới đảo chiều, nếu động cơ đang ngừng thì nhấn đảo chiều không có tác dụng.

Có led đơn báo trạng thái ngừng, quay thuận, quay ngược.

Lưu tên file là “BAI_912_DC_ON_OFF_INV”.

- a. Mục đích: biết điều khiển động cơ DC chạy thuận và chạy nghịch.
- b. Lưu đồ: sinh viên hãy tự viết.
- c. Chương trình:

```
#include <tv_pickit2_shift_1.c>
#include <tv_pickit2_shift_dc_motor.c>
void phim_inv()
{
    if (!input(inv))
    {
        delay_ms(20);
        {
            if (!input(inv))
            {
                dc_motor_stop();
                delay_ms(1000);
            }
        }
    }
}
```

```

        dcmotor_tn=~dcmotor_tn;
        dc_motor_control();
        while(!input(inv));
    }
}
void main()
{
    set_up_port_ic_chot();
    dc_motor_enable();
    dc_motor_stop();
    while(true)
    {
        while(input(on));           //cho nhan stop
        dcmotor_onoff=1;
        dc_motor_control();
        do
        {
            phim_inv();
        }while(input(off));
        dc_motor_stop();
    }
}

```

- d. Tiến hành biên dịch và nạp.
- e. Quan sát kết quả: Sau khi nạp xong thì nhấn nút ON động cơ sẽ quay thuận, nhấn OFF sẽ ngừng động cơ, nhấn INV thì động cơ ngừng rồi sau đó đảo chiều.
- f. Giải thích chương trình:

Bài tập 913. Hãy viết lại chương trình bài 912 sao cho các lệnh nằm trong vòng lặp while(true) chỉ dùng lệnh if và thỏa điều kiện là chỉ kiểm tra nút nhấn ON khi động cơ đang ngừng, khi động cơ chạy thì chỉ kiểm tra nút nhấn OFF hoặc INV.

Thêm phần điều khiển 32 led: khi động cơ ngừng thì 32 led tắt, khi động cơ chạy thuận thì 16 led phải sáng, nghịch thì 16 led trái sáng.

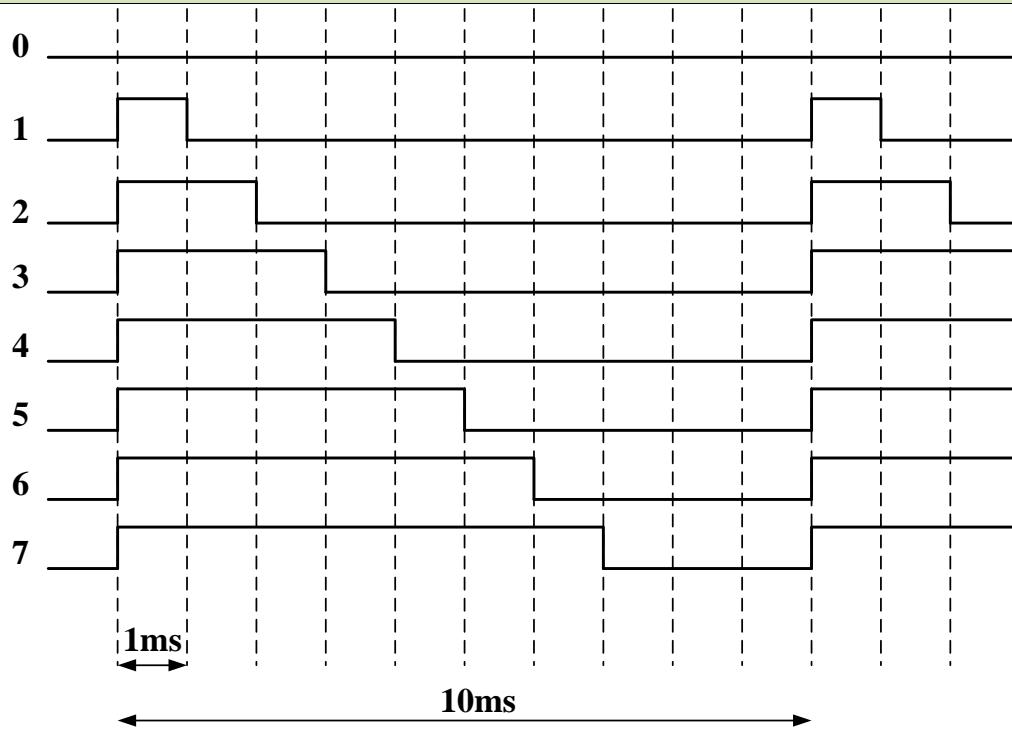
Lưu tên file là “BAI_913_DC_ON_OFF_INV_ONLY_IF”.

9.3 ĐIỀU KHIỂN THAY ĐỔI TỐC ĐỘ ĐỘNG CƠ DC PWM, ENCODER

9.3.1 Giới thiệu PWM (PULSE WIDTH MODULATION)

Nhắc lại lý thuyết:

Để giúp bạn hiểu PWM là gì thì chúng ta khảo sát dạng sóng điều chế độ rộng xung PWM như hình 9-13.



Hình 9-13. Dạng sóng điều chế độ rộng xung.

Quan sát dạng sóng hình 9-13 chúng ta thấy:

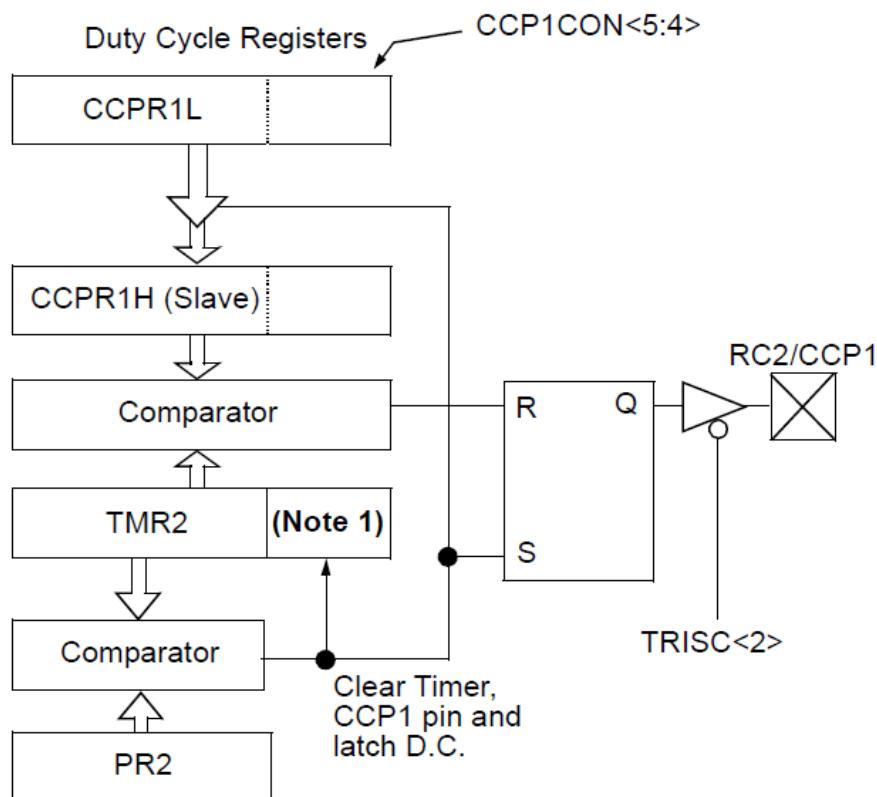
- Chu kỳ tín hiệu là 10ms.
- Ở cấp tốc độ 0 thì tín hiệu bằng 0, động cơ ngừng.
- Ở cấp tốc độ 1 thì tín hiệu điều khiển ở mức 1 chỉ có 1ms, ở mức 0 là 9ms. Giả sử dòng tạo ra là 10 mA khi ở mức 1, khi đó dòng trung bình là $(10*1)/10 = 1$ mA. Nếu điều khiển động cơ thì động cơ chạy 1/10 tốc độ định mức.
- Ở cấp tốc độ 2 thì tín hiệu điều khiển ở mức 1 chỉ có 2ms, ở mức 0 là 8ms. Giả sử dòng tạo ra là 10 mA khi ở mức 1, khi đó dòng trung bình là $(10*2)/10 = 2$ mA. Nếu điều khiển động cơ thì động cơ chạy 2/10 tốc độ định mức.
- Ở cấp tốc độ 5 thì tín hiệu điều khiển ở mức 1 chỉ có 5ms, ở mức 0 là 5ms. Giả sử dòng tạo ra là 10 mA khi ở mức 1, khi đó dòng trung bình là $(10*5)/10 = 5$ mA.
- Ở cấp tốc độ 10 thì tín hiệu điều khiển ở mức 1 là 10ms, ở mức 0 là 0ms. Giả sử dòng tạo ra là 10 mA khi ở mức 1, khi đó dòng trung bình là $(10*10)/10 = 10$ mA. Tốc độ đạt cực đại.

Khi thay đổi tốc độ thì chỉ thay đổi thời gian xung ở mức 1, còn gọi là hệ số chu kỳ (hay hệ số công tác), còn chu kỳ thì không đổi.

Ứng dụng điều chế độ rộng xung PWM dùng để điều khiển tốc độ động cơ DC, thay đổi độ sáng của đèn, ...

9.3.2 Cấu trúc khối điều chế độ rộng xung PWM của PIC

PIC có tích hợp module điều khiển PWM và có sơ đồ khối như hình 9-14:

**Hình 9-14. Sơ đồ khối của PWM PIC.**

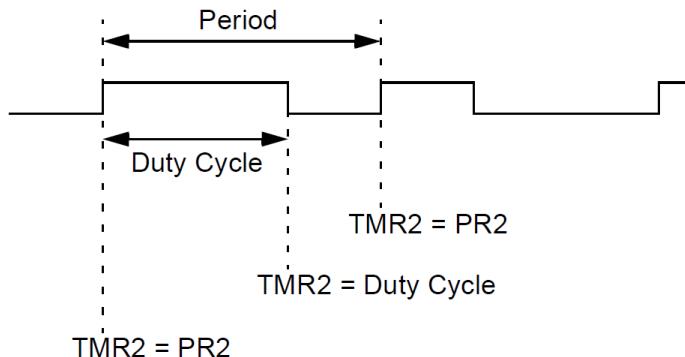
Khối PWM gồm có 2 mạch so sánh: mạch so sánh 8 bit với mạch so sánh 10 bit.

Mạch so sánh 8 bit sẽ so sánh giá trị đếm của timer T2 với giá trị của thanh ghi PR2 (period register), giá trị trong timer T2 tăng từ giá trị đặt trước cho đến khi bằng giá trị của PR2 thì mạch so sánh sẽ set flip flop RS làm ngõ ra CCPx lên mức 1. Đồng thời nạp giá trị 10 bit từ thanh ghi CCPRxL sang thanh ghi CCPRxH, timer T2 bị reset và bắt đầu đếm lại cho đến khi giá trị của timer T2 bằng giá trị của CCPRxH thì mạch so sánh sẽ reset flip flop RS làm ngõ ra CCPx về mức 0. Quá trình này lặp lại.

Dạng sóng điều chế PWM như hình 9-15.

Chu kỳ không thay đổi, muốn thay đổi thời gian xung ở mức 1 thì ta thay đổi hệ số chu kỳ (duty cycle). Khi hệ số chu kỳ thay đổi thì điện áp hay dòng trung bình thay đổi.

Hệ số chu kỳ càng lớn thì dòng trung bình càng lớn, nếu điều khiển động cơ sẽ làm thay đổi tốc độ.

**Hình 9-15. Dạng sóng PWM.**

9.3.3 Tính chu kỳ xung PWM

Chu kỳ PWM của vi điều khiển PIC được tính theo công thức:

$$PERIOD_{PWM} = [(PR2) + 1] * 4 * T_{OSC} * PV_{TMR2}$$

Trong đó: T_{OSC} là chu kỳ của tụ thạch anh tạo dao động.

PV_{TMR2} (Prescale Value) giá trị chia trước của timer2.

$PR2$ (period register) thanh ghi chu kỳ.

9.3.4 Tính hệ số chu kỳ xung PWM

Hệ số chu kỳ được thiết lập bởi giá trị lưu trong thanh ghi 10 bit gồm thanh ghi 8 bit CCPRxL và 2 bit còn lại là bit thứ 4 và thứ 5 lưu ở trong thanh ghi CCPxCON – kí hiệu là CCPxCON<5:4>.

Giá trị của hệ số chu kỳ là 10 bit nên có thể thay đổi từ 0 đến 1023 tạo ra 1024 cấp giá trị điều khiển.

Giá trị 10 bit thì 8 bit có trọng số lớn lưu trong thanh ghi CCPRxL và 2 bit còn lại có trọng số thấp thì ở CCPxCON<5:4>.

Hệ số chu kỳ của vi điều khiển PIC được tính theo công thức:

$$DUTY_CYCLE_{PWM} = (CCPRxL : CCPxCON < 5:4 >) * T_{OSC} * PV_{TMR2}$$

9.3.5 Các lệnh điều khiển PWM

a. LỆNH ĐỊNH CẤU HÌNH KHỐI CCP

Cú pháp: setup_ccp1 (*mode*) or setup_ccp1 (*mode, pwm*)

Thông số: **mode** là hằng số. Giá trị của mode xem trong file thiết bị, một vài thông số:

Disable the CCP: CCP_OFF

Set CCP to PWM mode: CCP_PWM Enable Pulse Width Modulator

Chức năng: Khởi tạo khối CCP.

b. LỆNH THIẾT LẬP HỆ SỐ CHU KỲ

Cú pháp: set_pwm1_duty (*value*)

Thông số: **value** có thể là hằng số 8 bit hoặc 16 bit.

Chức năng: Ghi giá trị 10 bit vào PWM để thiết lập hệ số chu kỳ.

Giá trị 10 bit được dùng để xác định lượng thời gian của tín hiệu PWM ở mức 1 trong một chu kỳ.

c. LỆNH ĐỊNH CÁU HÌNH TIMER2 – SETUP_TIMER_2

Cú pháp: **Setup_timer_2(mode, period, postscale)**

Thông số: **mode** có thể là 1 trong các thông số: T2_DISABLED, T2_DIV_BY_1, T2_DIV_BY_4, T2_DIV_BY_16

Period là số nguyên có giá trị từ 0 đến 255 dùng để xác định khi nào giá trị timer bị reset.

postscale là số nguyên có giá trị từ 1 đến 16 dùng để xác định timer tràn bao nhiêu lần trước khi phát sinh tín hiệu ngắt.

Chức năng: Khởi tạo cho TIMER2.

d. LỆNH THIẾT LẬP GIÁ TRỊ BẮT ĐẦU – SET_TIMERx(value)

Cú pháp: **set_timerX(value)** ; x là 0, 1, 2

Thông số: **value** là hằng số nguyên 8 hoặc 16 bit dùng để thiết lập giá trị mới cho timer.

Chức năng: thiết lập giá trị bắt đầu cho TIMER.

9.3.6 Tính toán điều khiển tốc độ động cơ bằng PWM

Phần này trình bày cách chọn chu kỳ, tính toán các thông số cho phù hợp để điều khiển thay đổi tốc độ động cơ DC dựa vào phần cứng đã thiết kế của BTN.

Cho tần số tụ thạch anh BTN đang dùng là 20MHz.

Cho chu kỳ PWM là 0.8ms.

Hãy tính toán các thông số để điều khiển tốc độ của động cơ từ tốc độ thấp nhất đến tốc độ nhanh nhất.

Tính toán:

Tần số thạch anh vi điều khiển sử dụng: $f_{osc} = 20MHz$

$$\text{nên chu kỳ là: } T_{osc} = \frac{1}{f_{osc}} = \frac{1}{20MHz} = 0,5\mu S = 50ns$$

Tính chu kỳ PWM: $PERIOD_{PWM} = [(PR2) + 1] * 4 * T_{osc} * PV_{TMR2} = 0.8mS = 800,000ns$

Chỉ biết được $T_{osc} = 50ns$ còn các thông số $PR2, PV_{TMR2}$ thì chưa biết.

Phải chọn 1 thông số và tính thông số còn lại: chú ý $PR2$ có giá trị 8 bit từ 0 đến 255, còn PV_{TMR2} có 3 giá trị là chia 1, chia 4 và chia 16.

Chọn hệ số chia lớn nhất là 16 hay $PV_{TMR2} = 16$

Khi đó tìm giá trị còn lại $PR2$:

$$[(PR2)+1] = \frac{PERIOD_{PWM}}{4 * T_{OSC} * PV_{TMR2}} = \frac{800,000ns}{4 * 50ns * 16} = 250$$

Vậy $PR2 = 249$.

Gán giá trị 249 và hệ số chia trước cho timer T2 thì ta phải sử dụng Lệnh khởi tạo cho timer2 là:

setup_timer_2(T2_DIV_BY_16, 249, 1)

Tính hệ số chu kỳ (duty):

Hệ số chu kỳ thay đổi sẽ làm giá trị trung bình của tín hiệu thay đổi, hệ số chu kỳ nhỏ nhất là bằng 0, khi đó tín hiệu ra CCPx ở mức 0, hệ số chu kỳ tăng làm tín hiệu xuất hiện và giá trị trung bình tăng, hệ số chu kỳ lớn nhất bằng chu kỳ của PWM.

Cho hệ số chu kỳ bằng chu kỳ để tính toán giới hạn hệ số chu kỳ:

$$DUTY_CYCLE_{PWM} = PERIOD_{PWM_MAX}$$

Ta có công thức: $DUTY_CYCLE_{PWM} = (CCPRxL : CCPxCON < 5:4>) * T_{OSC} * PV_{TMR2}$

Ta tìm giá trị lớn nhất của $CCPRxL : CCPxCON < 5:4>$.

$$\text{Suy ra: } (CCPRxL : CCPxCON < 5:4>) = \frac{DUTY_CYCLE_{PWM_MAX}}{T_{OSC} * PV_{TMR2}} = \frac{PERIOD_{PWM}}{T_{OSC} * PV_{TMR2}}$$

$$(CCPRxL : CCPxCON < 5:4>) = \frac{PERIOD_{PWM}}{T_{OSC} * PV_{TMR2}} = \frac{800,000ns}{50ns * 16} = 1000$$

Vậy giới hạn của hệ số chu kỳ là từ 0 đến 1000 đối với các thông số đã cho.

Nếu số cấp điều khiển là 10 cấp thì giá trị cho mỗi cấp là 100.

Nếu số cấp điều khiển là 20 cấp thì giá trị cho mỗi cấp là 50.

Tùy theo yêu cầu mà bạn tính toán được giá trị thay đổi cho mỗi cấp.

9.3.7 Các bài thực hành thay đổi tốc độ động cơ dc bằng PWM

Thư viện động cơ DC phần PWM.

Để tránh lặp lại thì ta:

Mở file “TV_PICKIT2_SHIFT_DC_MOTOR.C”

Rồi viết thêm vào nội dung sau:

a. Chương trình:

```
//XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
//CAC HAM CHO DK DONG CO DC BANG PWM
//XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
unsigned int16    pwm_duty;
void dc_motor_ktao_ccp1_thuan()
{
    setup_ccp2(ccp_off);
    setup_ccp1(ccp_pwm);
    output_low(pin_c1);
}
```

```

void dc_motor_ktao_ccp2_nghich()
{
    setup_ccp1(ccp_off);
    setup_ccp2(ccp_pwm);
    output_low(pin_c2);
}
void dc_motor_ktao_ccpx_ktao()
{
    if (dcmotor_tn)    dc_motor_ktao_ccp1_thuan();
    else                dc_motor_ktao_ccp2_nghich();
}
void dc_motor_pww_setup_duty()
{
    if (dcmotor_tn)    set_pwm1_duty(pwm_duty);
    else                set_pwm2_duty(pwm_duty);
}
void dc_motor_pwm_ccp1_ccp2_stop()
{
    pwm_duty=0;
    setup_ccp1(ccp_off);
    setup_ccp2(ccp_off);
    dc_motor_ktao_ccpx_ktao();
    dc_motor_pww_setup_duty();
}

```

b. Giải thích thông qua các câu hỏi:

Câu 1: Hãy cho biết hàm “DC_MOTOR_KTAO_CCP1_THUAN ()” có chức năng gì?

Câu 2: Hãy cho biết hàm “DC_MOTOR_KTAO_CCP2_NGHICH ()” có chức năng gì?

Câu 3: Hãy cho biết hàm “DC_MOTOR_KTAO_CCPIX_KTAO ()” có chức năng gì?

Câu 4: Hãy cho biết hàm “DC_MOTOR_PWW_SETUP_DUTY ()” có chức năng gì?

Câu 5: Hãy cho biết hàm “DC_MOTOR_PWM_CCP1_CCP2_STOP ()” có chức năng gì?

Bài mẫu 921. Chương trình dùng CCP1 ở chế độ PWM để điều khiển động cơ quay thuận thay đổi tốc độ bằng 2 nút điều khiển: UP, DW và nút STOP.

Khi nhấn UP thì động cơ quay thuận và tăng tốc lên, khi nhấn DW thì động cơ giảm tốc, nhấn STOP thì ngừng.

Số cấp điều khiển là 20 hiển thị trên 2 led 7 đoạn.

Sử dụng các thông số tính toán ở trên

Lưu tên file là “BAI_921_DC_PWM_CCP1_UP_DW_STOP”.

- Mục đích: biết lập trình sử dụng CCP1 điều khiển động cơ DC quay thuận thay đổi tốc độ.
- Lưu đồ: sinh viên hãy tự viết.
- Chương trình:

```

#include <tv_pickit2_shift_1.c>
signed int8  pwm_capso;
unsigned int16   duty;

```

```

void phim_up()
{
    if (!input(up) && (duty<1000))
    {
        delay_ms(20);
        if (!input(up))
        {
            duty= duty+50;
            set_pwm1_duty(duty);
            pwm_capso++;
            xuat_4led_7doan_giaima_xoa_so0(pwm_capso);
            delay_ms(200);
        }
    }
}

void phim_dw()
{
    if (!input(dw) && (duty>0))
    {
        delay_ms(20);
        if (!input(dw))
        {
            duty= duty-50;
            pwm_capso--;
            xuat_4led_7doan_giaima_xoa_so0(pwm_capso);
            set_pwm1_duty(duty);
            delay_ms(200);
        }
    }
}

void main()
{
    set_up_port_ic_chot();
    dc_enable=1;
    output_low(pin_c1);
    xuat_buzzer_relay();

    setup_ccp1(ccp_pwm);
    setup_timer_2(t2_div_by_16,249,1);
    duty=0;
    pwm_capso=0;
    xuat_4led_7doan_giaima_xoa_so0(pwm_capso);
    set_pwm1_duty(duty);

    while(true)
    {
        phim_dw();
        phim_up();
        if(!input(stop))
        {
            duty=0;
            pwm_capso=0;
        }
    }
}

```

```

        xuat_4led_7doan_giaima_xoa_so0(pwm_capso);
        set_pwm1_duty(duty);
    }
}

```

- d. Tiến hành biên dịch và nạp.
- e. Quan sát kết quả: Sau khi nạp xong thì nhấn nút UP động cơ sẽ tăng tốc, nhấn DW thì giảm tốc, nhấn STOP thì động cơ ngừng. Chú ý khi ở cấp tốc độ 1 hoặc 2 thì moment chưa đủ làm động cơ quay nên ta tiếp tục tăng cấp cho đến khi đủ moment quay động cơ. Có led hiển thị các cấp tốc độ.
- f. Giải thích chương trình:

Bài tập 922. Hãy lấy kết quả tính toán ở trên và lập trình theo yêu cầu.

Nút nhấn BT0 làm nút stop để ngừng động cơ.

Nút nhấn BT1 thì điều khiển động cơ chạy 35% tốc độ tối đa.

Nút nhấn BT2 thì điều khiển động cơ chạy 70% tốc độ tối đa.

Nút nhấn BT3 thì điều khiển động cơ chạy 100% tốc độ tối đa.

Hiển thị số cấp trên module 4 led 7 đoạn.

Lưu tên file là “BAI_922_DC_PWM_CCP1_3_CAP_TOCDO”.

Bài mẫu 923. Dùng vi điều khiển PIC 18F4550 thay đổi tốc độ động cơ DC bằng 4 nút điều khiển: UP, DW, STOP, INV.

Khi nhấn UP thì động cơ tăng tốc lên, khi nhấn DW thì động cơ giảm tốc, nhấn STOP thì ngừng.

Khi nhấn INV thì động cơ ngừng và đảo chiều, 16 led phải sáng thì DC quay thuận, 16 led trái sáng thì động cơ quay nghịch.

Số cấp điều khiển là 20 hiển thị trên 2 led 7 đoạn.

Lưu tên file là “BAI_923_DC_PWM_2CCP_UP_DW_STOP_INV”.

- a. Mục đích: lập trình điều khiển động cơ thay đổi tốc độ theo PWM, sử dụng CCP1 và CCP2 để điều khiển tốc độ theo cả 2 chiều.
- b. Lưu đồ: sinh viên hãy tự viết.
- c. Chương trình:

```

#include <tv_pickit2_shift_1.c>
#include <tv_pickit2_shift_dc_motor.c>
signed     int8      pwm_capso;
unsigned   int16     y=0;

void xuat_led_thuan_nghich()
{
    if (!dcmotor_tn)  xuat_32led_don_2word(0,y);
    else              xuat_32led_don_2word(y,0);
}

```

```
}

void xoa_so_cap_toc_do()
{
    pwm_capso=0;
    xuat_4led_7doan_giaima_xoa_so0(pwm_capso);
}

void phim_inv_pwm()
{
    if (!input(inv))
    {
        delay_ms(20);
        {
            if (!input(inv))
            {
                delay_ms(10);
                dcmotor_tn=~dcmotor_tn;
                xuat_led_thuan_nghich();
                xoa_so_cap_toc_do();
                dc_motor_pwm_ccp1_ccp2_stop();
                while(!input(inv));
            }
        }
    }
}

void phim_up_pwm()
{
    if (!input(up)&&(pwm_duty<1000))
    {
        delay_ms(20);
        if (!input(up))
        {
            pwm_duty= pwm_duty+50;
            dc_motor_pww_setup_duty();
            pwm_capso++;
            xuat_4led_7doan_giaima_xoa_so0(pwm_capso);
            delay_ms(200);
        }
    }
}

void phim_dw_pwm()
{
    if (!input(dw)&&(pwm_duty>0))
    {
        delay_ms(20);
        if (!input(dw))
        {
            pwm_duty= pwm_duty-50;
            pwm_capso--;
            xuat_4led_7doan_giaima_xoa_so0(pwm_capso);
            dc_motor_pww_setup_duty();
            delay_ms(200);
        }
    }
}
```

```

        }
    }

void phim_stop_pwm()
{
    if(!input(stop))
    {
        dc_motor_pwm_ccp1_ccp2_stop();
        xoa_so_cap_toc_do();
    }
}

void main()
{
    set_up_port_ic_chot();
    dc_motor_enable();
    setup_timer_2(t2_div_by_16,249,1);

    dcmotor_tn = 0;
    dcmotor_onoff=0;
    y = 0xffff;
    xuat_led_thuan_nghich();
    pwm_duty=0;
    dc_motor_ktao_ccpx_ktao();
    dc_motor_pww_setup_duty();

    pwm_capso=0;
    xuat_4led_7doan_giaima_xoa_so0(pwm_capso);
    while(true)
    {
        phim_dw_pwm();
        phim_up_pwm();
        phim_inv_pwm();
        phim_stop_pwm();
    }
}
}

```

- d. Tiến hành biên dịch và nạp.
- e. Quan sát kết quả: Sau khi nạp xong thì nhấn nút UP động cơ sẽ tăng tốc theo chiều thuận, nhấn DW thì giảm tốc, nhấn STOP thì động cơ ngừng. Khi nhấn INV thì động cơ đang chạy sẽ ngừng và chọn chiều ngược lại. Khi đó bạn nhấn UP thì sẽ thấy động cơ quay chiều ngược lại.
Có led hiển thị các cấp tốc độ và 32 led đơn hiển thị trạng thái thuận nghịch.
- f. Giải thích chương trình:

Bài tập 924. Hãy lập trình điều khiển động DC có 3 nút: UP, DW, STOP.

Khi nhấn UP 1 lần (không cầm giữ) thì động cơ tự động tăng tốc lên, khi nhấn DW thì động cơ giảm tốc, nhấn STOP thì ngừng.

Số cấp điều khiển là 20 hiển thị trên 2 led 7 đoạn, thời gian trễ là 1 giây.

Lưu tên file là “BAI_924_DC_UP_DW_STOP_AUTO”.

Bài mẫu 925. Giống bài 921 nhưng có thêm phần đếm xung từ encoder đưa đến ngõ vào của counter của timer T1.

Giá trị xung đếm được hiển thị trên 8 led 7 đoạn. Số cấp tốc độ hiển thị module 4 led.

Lưu tên file là “BAI_925_DC_UP_DW_STOP_ENCODER_T1”.

- Mục đích: đã biết sử dụng T1 để định thời và bây giờ biết đếm xung từ encode dùng counter T1.
- Lưu đồ: sinh viên hãy tự viết.
- Chương trình:

```
#include <tv_pickit2_shift_1.c>
unsigned int16 t1,t1_tam,duty;
signed int8 pwm_capso;

void phim_up()
{
    if (!input(up) && (duty<1000))
    {
        delay_quet_8led(100);
        if (!input(up))
        {
            duty= duty+50;
            set_pwm1_duty(duty);
            pwm_capso++;
            xuat_4led_7doan_giaima_xoa_so0(pwm_capso);
            delay_quet_8led(100);
        }
    }
}

void phim_dw()
{
    if (!input(dw) && (duty>0))
    {
        delay_quet_8led(100);
        if (!input(dw))
        {
            duty= duty-50;
            pwm_capso--;
            xuat_4led_7doan_giaima_xoa_so0(pwm_capso);
            set_pwm1_duty(duty);
            delay_quet_8led(200);
        }
    }
}

void main()
{
    set_up_port_ic_chot();
    dc_enable=1;
    output_low(pin_c1);
```

```

xuat_buzzer_relay();

setup_ccp1(ccp_pwm);
setup_timer_2(t2_div_by_16,249,1);
duty=0; pwm_capso=0;
xuat_4led_7doan_giaima_xoa_so0(pwm_capso);
set_pwm1_duty(duty);

setup_timer_1(t1_external_sync | t1_div_by_1);
set_timer1(0);
t1=t1_tam=0;
giai_ma_gan_cho_8led_quet_16_xoa(t1);

while(true)
{
    phim_dw();
    phim_up();
    t1=get_timer1();
    if(t1!=t1_tam)
    {
        giai_ma_gan_cho_8led_quet_16_xoa(t1);

        t1_tam=t1;
    }
    hien_thi_8led_7doan_quet();

    if(!input(stop))
    {
        duty=0; pwm_capso=0;
        xuat_4led_7doan_giaima_xoa_so0(pwm_capso);
        set_pwm1_duty(duty);
    }
}
}
}

```

- d. Tiến hành biên dịch và nạp.
- e. Quan sát kết quả: bạn tăng tốc độ để nhìn thấy giá trị đếm trên led khi động cơ quay.
- f. Giải thích chương trình:

Bài tập 926. Hiệu chỉnh bài 925 sao cho khi động cơ quay đến khi counter T1 tràn thì điều khiển động cơ ngừng.

Lưu tên file là “BAI_926_DC_UP_DW_STOP_ENCODER_T1_TRAN”.

Bài mẫu 927. Chương trình điều khiển thay đổi tốc độ động cơ, đếm xung từ encoder đưa đến ngõ vào của timer T1, tính được tốc độ động cơ quay vòng/phút.

Số cấp điều khiển là 100.

Lưu tên file là “BAI_927_DC_UP_DW_STOP_TOCDO_8LED”.

- a. Mục đích: tính được tốc độ động cơ vòng/phút.
- b. Lưu đồ: sinh viên hãy tự viết.
- c. Chương trình:

```
#include <tv_pickit2_shift_1.c>
#include <tv_pickit2_shift_lcd.c>
signed int16 t1,duty;
signed int8 bdn,pwm_capso;
#int_timer3
void interrupt_timer3()
{
    set_timer3(3036);
    bdn++;
    if (bdn==10)
    {
        bdn=0;
        t1=get_timer1();
        set_timer1(0); //đọc sau 1 giây
        t1=t1/45; t1=t1*60; //hay t1=(t1*4)/3
        giai_ma_gan_cho_8led_quet_16_xoa(t1);
    }
}
void phim_up()
{
    if (!input(up)&&(duty<1000))
    {
        delay_quet_8led(100);
        if (!input(up))
        {
            duty= duty+10;
            set_pwml1_duty(duty);
            pwm_capso++;
            xuat_4led_7doan_giaima_xoa_so0(pwm_capso);
            delay_quet_8led(300);
        }
    }
}
void phim_dw()
{
    if (!input(dw)&&(duty>0))
    {
        delay_quet_8led(100);
        if (!input(dw))
        {
            duty= duty-10;
            set_pwml1_duty(duty);
            pwm_capso--;
            xuat_4led_7doan_giaima_xoa_so0(pwm_capso);
            delay_quet_8led(300);
        }
    }
}
void main()
```

```

{
    set_up_port_ic_chot();
    dc_enable=1;
    output_low(pin_c1);
    xuat_buzzer_relay();

    setup_ccp1(ccp_pwm);
    setup_timer_2(t2_div_by_16,249,1);
    duty=0;
    pwm_capso=0;
    xuat_4led_7doan_giaima_xoa_so0(pwm_capso);
    set_pwm1_duty(duty);

    enable_interrupts(global);
    enable_interrupts(int_timer3);
    setup_timer_3(t3_internal|t3_div_by_8);
    set_timer3(3036);

    setup_timer_1 (t1_external_sync | t1_div_by_1);
    set_timer1(0);
    bdn = 0;
    while(true)
    {
        phim_dw();
        phim_up();
        hien_thi_8led_7doan_quet();
        if(!input(stop))
        {
            duty=0;
            pwm_capso=0;
            set_timer1(0);
            xuat_4led_7doan_giaima_xoa_so0(pwm_capso);
            set_pwm1_duty(duty);
        }
    }
}

```

- d. Tiến hành biên dịch và nạp.
- e. Quan sát kết quả: Hàng 1 hiển thị cấp tốc độ nằm trong giới hạn điều khiển từ 0 đến 20. Hàng 2 hiển thị tốc độ của động cơ. Nhấn nút UP để cho động cơ chạy ta sẽ biết được tốc độ động cơ tương ứng với từng cấp.
- f. Giải thích chương trình: Trong chương trình sử dụng timer T1 đếm xung từ encoder, sử dụng timer T3 làm bộ định thời đếm 100ms báo ngắt, sau 10 lần ngắt sẽ được 1 giây. Khi được 1 giây thì đọc giá trị của counter T1 rồi xóa giá trị của counter T1. Lấy giá trị vừa đọc chia cho 45 ta sẽ được số vòng trong vòng 1 giây. Lấy giá trị sau khi chia nhân với 60 giây để được tốc độ động cơ trong vòng 1 phút. Encoder tạo ra 45 xung khi động cơ quay được 1 vòng.

Bài tập 928. Giống bài 925 nhưng thay đổi tốc độ động cơ 10 cấp từ 0 đến 9 được chọn bằng 10 nút nhấn từ 0 đến 9 của bàn phím ma trận. Chọn chiều quay thuận hay nghịch bằng phím A, ngừng là phím B.

Lưu tên file là “BAI_928_DC_UP_DW_STOP_ENCODER_T1_8LED”.

9.4 ĐIỀU KHIỂN KẾT HỢP 2 ĐỘNG CƠ BUỚC VÀ DC

Bài tập 931. Hãy viết chương trình điều khiển 2 động cơ: DC và bước bằng 4 nút: BT0(ON_MTDC), BT1(OFF_MTDC), BT2(ON_MTB), BT3(OFF_MTB).

Có led đơn báo trạng thái ngừng, quay của động cơ DC và động cơ bước.

Động cơ DC chỉ ON và OFF.

Lưu tên file là “BAI_931_DC_SMT_ON_OFF”.

Bài tập 932. Hãy viết chương trình điều khiển 2 động cơ: DC và bước bằng 3 nút: UP, DW, OFF.

UP và DW dùng để thay đổi tốc độ cho cả 2 động số, có cấp là 10, hiển thị trên 2 led 7 đoạn. Khi nhấn OFF thì ngừng.

Lưu tên file là “BAI_932_DC_SMT_UP_DW_OFF”.

9.5 ĐIỀU KHIỂN ỔN ĐỊNH TỐC ĐỘ ĐỘNG CƠ DC

9.3.1 Giới thiệu PID

Phần này bạn nên xem lại lý thuyết điều khiển dùng PID.

9.3.2 Điều khiển động cơ dùng PID

Bài mẫu 931. Chương trình điều khiển động cơ DC ổn định tốc độ dùng phương pháp điều khiển PID. Nút UP dùng để tăng tốc độ, nút DW dùng để giảm tốc độ và STOP dùng để ngừng.

Lưu tên file là “BAI_931_DC_UP_DW_STOP_PID”.

- Mục đích: biết điều khiển ổn định tốc độ động cơ dùng PID.
- Lưu đồ: sinh viên hãy tự viết.
- Chương trình:

```
#include <tv_picket2_shift_1.c>
#include <tv_picket2_shift_lcd.c>
signed int16 t1;
signed int8 donvi,chuc,tram/ngan;
signed int16 xung,xungdat,duty,e,ecu=0,tocdo;
float kp=2,kd=0.2,ki=0.002,tp=0,td=0,ti=0;

#int_timer3
void interrupt_timer3() //chu kỳ ngắt là 100ms=0,1s
{
    set_timer3(3036);
    xung = get_timer1();
    set_timer1(0);
    e=xungdat - xung;
    tp = e;
```

```

        td = (e-ecu)*10;
        ti = ti+e*0.1;
        ecu=e;
        duty = duty + kp*tp + kd*td + ki*ti;
        if(duty>1000)    duty =1000;
        else if(duty<0)  duty=0;
        set_pwm1_duty(duty);
    }

void hienthi_lcd_t1 (signed int16 tam)
{
    donvi = (tam %10)+0x30;
    chuc = tam/10%10 +0x30 ;
    tram = tam/100%10 +0x30;
    ngan = tam/1000%10 +0x30;
    if (ngan==0x30)
    {
        ngan=' ';
        if (tram==0x30)
        {
            tram=' ';
            if (chuc==0x30) chuc=' ';
        }
    }
    lcd_command(0xd0);
    lcd_data(ngan);
    lcd_data(tram);
    lcd_data(chuc);
    lcd_data(donvi);
}

void hienthi_lcd_cap_tocdo ()
{
    lcd_command(0x92);
    lcd_data(tocdo/10+0x30);
    lcd_data(tocdo%10+0x30);
}
void hienthi_lcd_xung_dat ()
{
    lcd_command(0xa5);
    lcd_data(xungdat/100+0x30);
    lcd_data(xungdat/10%10+0x30);
    lcd_data(xungdat%10+0x30);
}
void hienthi_lcd_xung_dem ()
{
    lcd_command(0xe5);
    lcd_data(xung/100+0x30);
    lcd_data(xung/10%10+0x30);
    lcd_data(xung%10+0x30);
}

void phim_up ()
{

```

```

if (!input(up) && (xungdat<300))
{
    delay_ms(20);
    if (!input(up))
    {
        xungdat= xungdat+10;
        tocdo++;
        hienthi_lcd_cap_tocdo();
        delay_ms(200);
    }
    enable_interrupts(int_timer3);
}
void phim_dw()
{
    if (!input(dw) && (xungdat>0))
    {
        delay_ms(20);
        if (!input(dw))
        {
            xungdat= xungdat-10;
            tocdo--;
            hienthi_lcd_cap_tocdo();
            delay_ms(200);
        }
    }
}

void main()
{
    set_up_port_ic_chot();
    set_tris_c(0xf9);
    set_tris_b(0x3c);
    enable_interrupts(global);
    setup_timer_3(t3_internal|t3_div_by_8);
    set_timer3(3036);

    setup_lcd();
    lcd_command(lcd_addr_line1);
    lcd_data("cap toc do:");
    lcd_command(lcd_addr_line2);
    lcd_data("toc do dco-rpm:");
    lcd_command(lcd_addr_line3);
    lcd_data("xung dat: ");
    lcd_command(lcd_addr_line4);
    lcd_data("xung dem: ");

    setup_ccp1(ccp_pwm);
    setup_timer_2(t2_div_by_16,249,1);

    tocdo=0;
    dc_enable=1;
    xuat_buzzer_relay();
    output_low(pin_c1);
}

```

```

duty=0;
set_pwm1_duty (duty);

setup_timer_1 (t1_external_sync | t1_div_by_1);
set_timer1(0);

hienthi_lcd_cap_tocdo();
xungdat=0;
while(true)
{
    phim_dw();
    phim_up();
    if(!input(stop))
    {
        disable_interrupts(int_timer3);
        duty=0;
        set_timer1(0);
        tocdo=0;
        xungdat =0;
        xung =0;
        hienthi_lcd_cap_tocdo();
        set_pwm1_duty(0);
    }
    t1=(xung*40)/3; //t1=(xung*40*15)/3*15=(xung*600)/45;
    hienthi_lcd_t1(t1);
    hienthi_lcd_xung_dat();
    hienthi_lcd_xung_dem();
}
}

```

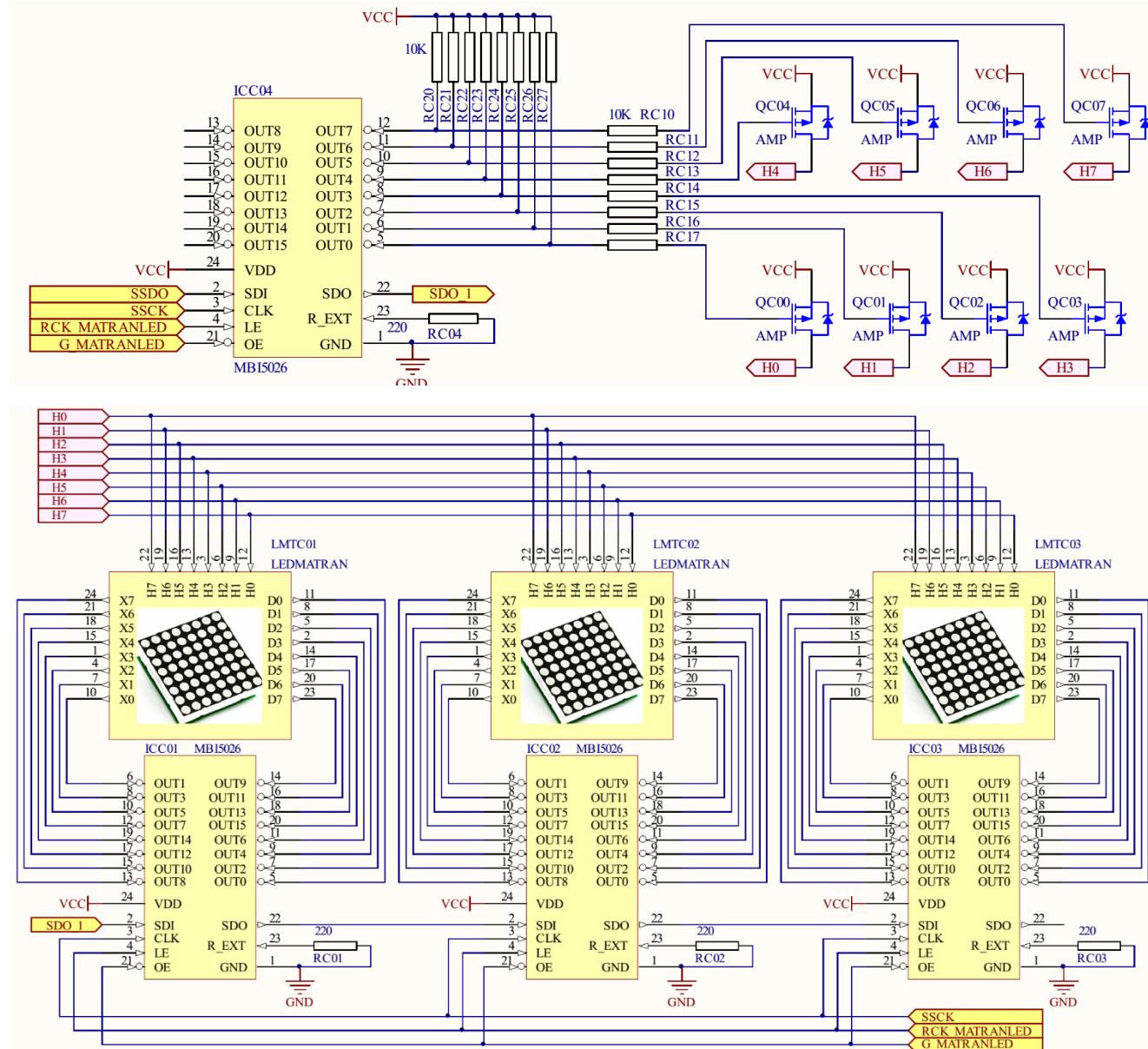
- d. Tiến hành biên dịch và nạp.
- e. Quan sát kết quả: Hàng 1 hiển thị cấp tốc độ, hàng 2 hiển thị tốc độ động cơ, hàng 3 hiển thị số xung cài đặt trước, hàng 4 hiển thị số xung đếm được từ encoder. Sau khi nạp xong chương trình thì nhấn nút UP để tăng giá trị xung cài đặt lên 10 đơn vị, động cơ sẽ chạy ổn định khi giá trị cài đặt là 30, hệ thống sẽ tự động điều chỉnh tốc độ thực gần đúng với tốc độ cài.
- f. Giải thích chương trình: Trong chương trình sử dụng timer T1 đếm xung từ encoder, sử dụng timer T3 làm bộ định thời đếm 100ms báo ngắn. Khi được 100ms thì đọc giá trị của counter T1 rồi xóa giá trị của counter T1. Lấy giá trị vừa đọc tính toán sai lệch giữa xung cài và xung đếm, tín hằng số tỷ lệ, hằng số vi phân, hằng số tích phân, cập nhật lại giá trị và tính toán lại hệ số duty để thay đổi tốc độ động cơ cho phù hợp đúng với giá trị cài đặt. Giới hạn cho hệ số duty là từ 0 đến 1000, giá trị 1000 tương ứng với số xung cài đặt lớn nhất là 300. Tốc độ động cơ được tính toán bằng số xung đếm được trong vòng 100ms nhân với 10 để được 1 giây, nhân tiếp với 60 để được 1 giây rồi chia cho số xung của 1 vòng là 45 sẽ được tốc độ động cơ là số vòng trên phút. Đơn giản 2 hệ số cho 15 nên phương trình còn lại là nhân 40 chia 3.

Để thấy được sự ổn định thì phải có tải thay đổi nhưng trong bộ thí nghiệm không có kéo tải nên khó nhìn thấy chức năng của PID.

Chương 10: CÁC BÀI THỰC HÀNH MODULE 7 – ĐIỀU KHIỂN LED MA TRẬN 2 MÀU

10.1 SƠ ĐỒ MẠCH GIAO TIẾP VI ĐIỀU KHIỂN VỚI LED MA TRẬN

Trong phần này sẽ trình bày phần giao tiếp vi điều khiển PIC18F4550 với 3 led ma trận như hình 10-1:



Hình 10-1. Sơ đồ nguyên lý giao tiếp vi điều khiển với 3 led ma trận.

Kit có tích hợp module giao tiếp 3 led ma trận 2 màu xanh và đỏ, sử dụng 4 IC thanh ghi dịch 16 bit để điều khiển hàng và cột.

10.2 MÃ KÍ TỰ CHO LED MA TRẬN DẠNG QUÉT CỘT

Theo sơ đồ mạch điện có 3 led ma trận 8×8 sẽ có 8 hàng và 24 cột, có 2 phương pháp điều khiển led: quét cột và quét hàng.

Có 24 cột thì khi quét sẽ có 1 cột sáng, 23 cột tắt, thời gian sáng cho 1 cột là 1/24, thời ngắn nên led sẽ sáng mờ - đây là khuyết điểm. Ưu điểm là lập trình tạo mã hiển thị cho các ký tự đơn giản.

Có 8 hàng thì khi quét sẽ có 1 hàng sáng, 7 hàng tắt, thời gian sáng cho 1 hàng là 1/8, thời gian dài hơn so với quét cột nên led sáng rõ, đẹp – đây là ưu điểm. Khuyết điểm là lập trình tạo mã hiển thị cho các ký tự liên kết khá phức tạp.

a. Mã quét cột:

Mã quét cột cho 1 led gồm 16 bit: 8 bit cho 8 cột đỏ, 8 bit cho 8 cột xanh, khi quét cột thì mỗi một thời điểm chỉ cho 1 cột ở mức logic 1, các cột còn lại ở mức 0.

Do kết nối mạch tối ưu theo vị trí các chân IC thanh ghi dịch MBI 5026 với các led nên thứ tự các chân điều khiển không liên tục với nhau.

Bảng 10-1. Mã quét cột cho led màu đỏ sáng, xanh tắt.

T T	bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0		
Cộ t	D 3	X 3	D 2	X 2	D 1	X 1	D 0	X 0	B H	X 4	D 4	X 5	D 5	X 6	D 6	X 7	D 7	B L	2 BYT E
1								1		02								00	0200
2						1				08								00	0800
3				1						20								00	2000
4		1								80								00	8000
5										00		1						40	0040
6										00				1				10	0010
7										00						1		04	0004
8										00							1	01	0001

{0x0200,0x0800,0x2000,0x8000,0x0040,0x0010,0x0004,0x0001}

Chú ý: các ô còn lại đều bằng 0.

Bảng 10-2. Mã quét cột cho led màu xanh sáng, đỏ tắt.

T T	bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0		
Cộ t	D 3	X 3	D 2	X 2	D 1	X 1	D 0	X 0	B H	X 4	D 4	X 5	D 5	X 6	D 6	X 7	D 7	B L	2 BYT E
1									1	01								00	0100
2							1			04								00	0400
3					1					10								00	1000
4			1							40								00	4000

5										00	1										80	0080
6										00				1							20	0020
7										00						1					08	0008
8										00								1		02	0002	

{0x0100,0x0400,0x1000,0x4000,0x0080,0x0020,0x0008,0x0002}

Bảng 10-3. Mã quét cột cho led màu xanh và đỏ đều sáng thành màu cam.

T T	bit	15	14	13	12	11	10	9	8	B H	7	6	5	4	3	2	1	0		
		Cộ t	D 3	X 3	D 2	X 2	D 1	X 1	D 0		X 4	D 4	X 5	D 5	X 6	D 6	X 7	D 7	B L	2 BYT E
1								1	1	03									00	0300
2						1	1			0C									00	0C00
3				1	1					30									00	3000
4		1	1							C0									00	C000
5										00	1	1							C0	00C0
6										00			1	1					30	0030
7										00						1	1		0C	000C
8										00							1	1	03	0003

{0x0300,0x0C00,0x3000,0xC000,0x00C0,0x0030,0x000C,0x0003}

Mã điều khiển cả 2 màu xanh đỏ có thể OR 2 mã của riêng từng led với nhau.

b. Mã kí tự cho kiểu quét cột:

Dữ liệu điều khiển hàng: mức 1 thì led sáng, mức 0 thì led tắt: mỗi kí tự dùng ma trận 5×7 và một cột khoảng cách giữa 2 kí tự nên tổng số kí tự là 6.

Bảng 10-4. Mã các kí tự mẫu A, B, C, D.

	CHỮ A					CHỮ B					CHỮ C					CHỮ D									
Bit	0	1	2	3	4	5	6	7	0	1	2	3	4	5	6	7	0	1	2	3	4	5	6	7	
B7(8)																									
B6(4)			X				X	X	X	X				X	X	X			X	X	X	X			
B5(2)		X		X			X				X		X				X		X				X		
B4(1)	X			X		X				X		X						X					X		
B3(8)	X				X		X	X	X				X					X					X		
B2(4)	X	X	X	X	X		X				X		X					X					X		
B1(2)	X				X		X				X		X					X		X			X		
B0(1)	X				X		X	X	X					X	X	X			X	X	X	X			

MÃ	1F	24	44	24	1F	00	7F	49	49	49	36	00	3E	41	41	41	22	00	7F	41	41	41	3E	00
----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----

Bảng 10-5. Bảng mã ma trận của các kí tự.

TT	Kí tự	Mã ma trận
1	A	0x1F,0x24,0x44,0x24,0x1F,0x00
2	B	0x7F,0x49,0x49,0x49,0x36,0x00
3	C	0x3E,0x41,0x41,0x41,0x22,0x00
4	D	0x7F,0x41,0x41,0x41,0x3E,0x00
5	E	0x3E,0x49,0x49,0x41,0x22,0x00
6	F	0x7F,0x48,0x48,0x48,0x40,0x00
7	G	0x3E,0x41,0x49,0x49,0x2E,0x00
8	H	0x7F,0x08,0x08,0x08,0x7F,0x00
9	I	0x41,0x41,0x7F,0x41,0x41,0x00
10	J	0x42,0x41,0x7E,0x40,0x40,0x00
11	K	0x7F,0x08,0x14,0x22,0x41,0x00
12	L	0x7F,0x01,0x01,0x01,0x02,0x00
13	M	0x7F,0x20,0x10,0x20,0x7F,0x00
14	N	0x7F,0x20,0x10,0x08,0x7F,0x00
15	O	0x3E,0x41,0x41,0x41,0x3E,0x00
16	P	0x3F,0x48,0x48,0x48,0x30,0x00
17	Q	0x3E,0x41,0x45,0x43,0x3E,0x0
18	R	0x3F,0x48,0x4C,0x4A,0x31,0x00
19	S	0x32,0x49,0x49,0x49,0x26,0x00
20	T	0x60,0x40,0x7F,0x40,0x60,0x00
21	U	0x7E,0x01,0x01,0x01,0x7E,0x00
22	V	0x7C,0x02,0x01,0x02,0x7C,0x00
23	W	0x7E,0x01,0x06,0x01,0x7E,0x00
24	X	0x41,0x22,0x1C,0x22,0x41,0x00

25	Y	0x70,0x08,0x07,0x08,0x70,0x00
26	Z	0x63,0x45,0x49,0x51,0x63,0x00
27	0	0x3E,0x41,0x41,0x41,0x3E,0x00
28	1	0x11,0x21,0x7F,0x01,0x01,0x00
29	2	0x27,0x49,0x49,0x49,0x31,0x00
30	3	0x22,0x41,0x49,0x49,0x36,0x00
31	4	0x0C,0x14,0x24,0x7F,0x04,0x00
32	5	0x72,0x51,0x51,0x51,0x4E,0x00
33	6	0x3E,0x49,0x49,0x49,0x26,0x00
34	7	0x41,0x42,0x44,0x48,0x70,0x00
35	8	0x36,0x49,0x49,0x49,0x36,0x00
36	9	0x32,0x49,0x49,0x49,0x3E,0x00

10.3 CÁC CHƯƠNG TRÌNH ĐIỀU KHIỂN LED MA TRẬN QUÉT CỘT

Thư viện điều khiển led ma trận theo kiểu quét cột. Chương trình thư viện có tên là: “TV_PICKIT2_SHIFT_MATRAN_QUET_COT.c”.

Đã có trong thư mục của bạn, bạn có thể sử dụng cho các bài thực hành led ma trận theo kiểu quét cột và nên đọc hiểu.

```
#include <tv_pickit2_shift_font.c>
//đu lieu ma quet cot
const unsigned int16 ma_quet_cot_do[8]={
0x0200,0x0800,0x2000,0x8000,0x0040,0x0010,0x0004,0x0001};
const unsigned int16 ma_quet_cot_xanh[]={
0x0100,0x0400,0x1000,0x4000,0x0080,0x0020,0x0008,0x0002};
const unsigned int16 ma_quet_cot_cam[]={
0x0300,0x0c00,0x3000,0xc000,0x00c0,0x0030,0x000c,0x0003};

unsigned int8    dl_hienthi[6*20];
unsigned int8    slkt,mtb0,mtb1,mtb2;
unsigned int16   ma_hang, mqc_led0, mqc_led1, mqc_led2;
unsigned int32   tt_qc=1;
#define  tg_delay   100

unsigned int8 doi_1_bit_1_sang_thutu(unsigned int8 bx)
{
    unsigned int8 ttcot=0;
    bx= bx>>1;
```

```

        while(bx!=0)
        {
            ttcot++;
            bx= bx>>1;
        }
        return(ttcot);
    }

void lay_ma_ky_tu(unsigned int8 slkt_x)
{
    unsigned int8 kytu,i,n,j;
    n=0;
    for (i=0; i<slkt_x; i++)
    {
        kytu = chuoi_hienthi[i];
        if ((kytu>=0x41)&&(kytu<=0x5a))           kytu = kytu - 0x41;
        else if ((kytu>=0x30)&&(kytu<=0x39))      kytu = (kytu - 0x30) + 26;
        for(j=0;j<6 ;j++,n++)
            dl_hienthi[n]=hieu_chinh_4bit_cao(matran_kytu[kytu][j]);
    }
}

void lay_ma_quet_24_cot_xanh()
{
    mtb0 = tt_qc;
    mtb1 = tt_qc>>8;
    mtb2 = tt_qc>>16;
    if (mtb0!=0)
    {
        mtb0=doi_1_bit_1_sang_thutu(mtb0);
        mqc_led0 = ma_quet_cot_xanh[mtb0];
        mqc_led1 = 0;
        mqc_led2 = 0;
    }
    else if (mtb1!=0)
    {
        mtb1=doi_1_bit_1_sang_thutu(mtb1);
        mqc_led0 = 0;
        mqc_led1 = ma_quet_cot_xanh[mtb1];
        mqc_led2 = 0;
    }
    else if (mtb2!=0)
    {
        mtb2=doi_1_bit_1_sang_thutu(mtb2);
        mqc_led0 = 0;
        mqc_led1 = 0;
        mqc_led2 = ma_quet_cot_xanh[mtb2];
    }
}

void lay_ma_quet_24_cot_do()
{
    mtb0 = tt_qc;
    mtb1 = tt_qc>>8;
}

```

```

mtb2 = tt_qc>>16;
if (mtb0!=0)
{
    mtb0=doi_1_bit_1_sang_thutu(mtb0);
    mqc_led0 = ma_quet_cot_do[mtb0];
    mqc_led1 = 0;
    mqc_led2 = 0;
}
else if (mtb1!=0)
{
    mtb1=doi_1_bit_1_sang_thutu(mtb1);
    mqc_led0 = 0;
    mqc_led1 = ma_quet_cot_do[mtb1];
    mqc_led2 = 0;
}
else if (mtb2!=0)
{
    mtb2=doi_1_bit_1_sang_thutu(mtb2);
    mqc_led0 = 0;
    mqc_led1 = 0;
    mqc_led2 = ma_quet_cot_do[mtb2];
}
}

void lay_ma_quet_24_cot_cam()
{
    mtb0 = tt_qc;
    mtb1 = tt_qc>>8;
    mtb2 = tt_qc>>16;
    if (mtb0!=0)
    {
        mtb0=doi_1_bit_1_sang_thutu(mtb0);
        mqc_led0 = ma_quet_cot_cam[mtb0];
        mqc_led1 = 0;
        mqc_led2 = 0;
    }
    else if (mtb1!=0)
    {
        mtb1=doi_1_bit_1_sang_thutu(mtb1);
        mqc_led0 = 0;
        mqc_led1 = ma_quet_cot_cam[mtb1];
        mqc_led2 = 0;
    }
    else if (mtb2!=0)
    {
        mtb2=doi_1_bit_1_sang_thutu(mtb2);
        mqc_led0 = 0;
        mqc_led1 = 0;
        mqc_led2 = ma_quet_cot_cam[mtb2];
    }
}
void lay_so_luong_ky_tu_chuoi()
{
    slkt=0;
}

```

```

        while(chuoi_hienthi[slkt])
        {
            slkt++;
        }
        xuat_4led_7doan_giaima_xoa_so0(slkt);
    }

void quet_4kytu_mau_xanh(unsigned int8 sck, unsigned int8 vitri)
{
    unsigned int8 ck,j;
    for (ck=0;ck<sck;ck++)
    {
        tt_qc = 1;
        for(j=vitri;j<vitri+24;j++)
        {
            lay_ma_quet_24_cot_xanh();
            ma_hang = dl_hienthi[j];
            xuat_matranled(ma_hang,mqc_led2,mqc_led1,mqc_led0);
            delay_us(tg_delay);
            xuat_matranled(0,0,0,0);
            tt_qc = tt_qc << 1;
        }
    }
}

void quet_4kytu_mau_do(unsigned int8 sck, unsigned int8 vitri)
{
    unsigned int8 ck,j;
    for (ck=0;ck<sck;ck++)
    {
        tt_qc = 1;
        for(j=vitri;j<vitri+24;j++)
        {
            lay_ma_quet_24_cot_do();
            ma_hang = dl_hienthi[j];
            xuat_matranled(ma_hang,mqc_led2,mqc_led1,mqc_led0);
            delay_us(tg_delay);
            xuat_matranled(0,0,0,0);
            tt_qc = tt_qc <<1;
        }
    }
}

void quet_4kytu_mau_cam(unsigned int8 sck, unsigned int8 vitri)
{
    unsigned int8 ck, j;
    for (ck=0;ck<sck;ck++)
    {
        tt_qc = 1;
        for(j=vitri;j<vitri+24;j++)
        {
            lay_ma_quet_24_cot_cam();
            ma_hang = dl_hienthi[j];
            xuat_matranled(ma_hang,mqc_led2,mqc_led1,mqc_led0);
        }
    }
}

```

```

        delay_us(tg_delay);
        xuat_matranled(0,0,0,0);
        tt_qc = tt_qc <<1;
    }
}

```

Bài mẫu 1001. Chương trình điều khiển LED ma trận hiển thị 4 kí tự chữ ABCD màu đỏ.

Lưu tên file là “BAI_1001_MT_ABCD_DO”

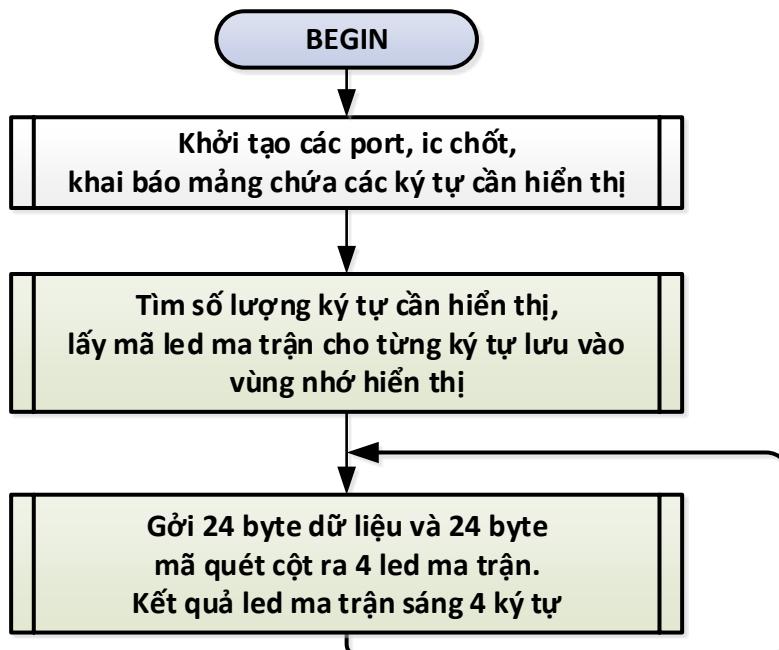
- Mục đích: biết mã ma trận, chuyển đổi mã ASCII sang mã led ma trận, xử lý đúng thứ tự và quét led ma trận theo cột.
- Lưu đồ: như hình 10-2.
- Giải thích lưu đồ :

Phần đầu thì khởi tạo port và IC chốt giống như các chương trình trước, khai báo mảng chứa các kí tự cần hiển thị.

Phần tiếp theo thì tìm số lượng kí tự trong mảng cần hiển thị là bao nhiêu, tối đa không quá 20 kí tự vì vùng nhớ đệm khai báo trong thư viện tối đa là 20.

Sau đó lấy mã led ma trận của từng kí tự có trong vùng nhớ đã tạo sẵn trong thư viện, mỗi kí tự 6 byte, 5 byte và 1 byte khoảng cách giữa 2 kí tự lưu vào vùng nhớ để hiển thị.

Tiến hành gởi dữ liệu của vùng nhớ hiển thị ra ma trận led.



Hình 10-2. Lưu đồ điều khiển hiển thị 4 kí tự trên 3 led ma trận.

- Chương trình:

```
#include <tv_pickit2_shift_1.c>
const unsigned char chuoi_hienthi [] = {"abcd"};
```

```
#include <tv_pickit2_shift_matran_quet_cot.c>
void main()
{
    set_up_port_ic_chot();
    lay_so_luong_ky_tu_chuoi();
    lay_ma_ky_tu(slkt);
    while(true)
    {
        quet_4kytu_mau_do(50,0);
    }
}
```

- e. Tiến hành biên dịch và nạp.
- f. Quan sát kết quả: sau khi nạp xong thì LED sáng chữ ABCD. Bạn có thể thay đổi sang các kí tự khác.
- g. Giải thích chương trình: các chương trình xử lý: lấy số lượng chuỗi cần hiển thị, chuyển mã ASCII sang mã led ma trận lưu vào vùng nhớ hiển thị rồi hiển thị ra led.

Bài tập 1002. Hãy điều chỉnh trình bài 1001 để hiển thị “0123” màu xanh.

Lưu tên file là “BAI_1002_MT_0123_XANH”

Bài tập 1003. Hãy điều chỉnh trình bài 1001 để hiển thị “ILOY” màu cam.

Lưu tên file là “BAI_1003_MT_ILOY_CAM”

Bài tập 1004. Chương trình điều khiển LED ma trận hiển thị “ILOY” màu xanh, màu đỏ rồi sau đó hiển thị màu cam và lặp lại.

Lưu tên file là “BAI_1004_MT_ILOY_XANH_DO_CAM”

Bài tập 1005. Chương trình điều khiển LED ma trận hiển thị chuỗi “DAIHOCSUPHAMKYTHUAT” màu xanh.

Lưu tên file là “BAI_1005_MT_CHUOI_XANH_DICH_TRAI”

- a. Mục đích: biết hiển thị dịch chuỗi.
- b. Lưu đồ: sinh viên hãy tự viết.
- c. Chương trình:

```
#include <tv_pickit2_shift_1.c>
const unsigned char chuoi_hienthi []={"abcdefghijkl"};
#include <tv_pickit2_shift_matran_quet_cot.c>
unsigned int16 vt;
void main()
{
    set_up_port_ic_chot();
    lay_so_luong_ky_tu_chuoi();
    lay_ma_ky_tu(slkt);
    while(true)
```

```

    {
        for (vt=0; vt < (slkt-4)*6; vt++)
            quet_4kytu_mau_xanh(30,vt);
    }
}

```

- d. Tiến hành biên dịch và nạp.
- e. Quan sát kết quả: sau khi nạp xong thì led ma trận hiển thị 4 kí tự màu xanh, sau đó dịch sang trái.
- f. Giải thích chương trình:

Bài tập 1006. Hãy lập trình giống bài 1005 nhưng hiển thị lần lượt 3 màu xanh, đỏ, cam.

Lưu tên file là “BAI_1006_MT_CHUOI_XANH_DO_CAM_DICH_TRAI”

Bài tập 1007. Hãy lập trình đếm từ 0 đến 9 hiển thị trên led ma trận màu xanh.

Thời gian delay 1s gần đúng hoặc chính xác thì dùng timer T1.

Lưu tên file là “BAI_1007_MT_DEM_1SO_XANH”

Bài tập 1008. Hãy lập trình đếm từ 00 đến 99 hiển thị trên led ma trận màu xanh.

Thời gian delay 1s gần đúng hoặc chính xác thì dùng timer T1.

Lưu tên file là “BAI_1008_MT_DEM_2SO_XANH”

Bài tập 1009. Hãy lập trình đếm từ 000 đến 999 hiển thị trên led ma trận màu xanh.

Thời gian delay 1s gần đúng hoặc chính xác thì dùng timer T1.

Lưu tên file là “BAI_1009_MT_DEM_3SO_XANH”

10.4 MÃ KÍ TỰ CHO LED MA TRẬN DẠNG QUÉT HÀNG

Ở các bài thực hành led ma trận theo phương pháp quét cột thì led sáng không rõ với nguyên nhân đã trình bày, phần này sẽ trình bày cách quét hàng. Khi quét hàng thì mỗi một thời điểm chỉ có 1 hàng sáng, muốn led nào sáng thì cột tương ứng sẽ ở mức 0. Mỗi một cột chỉ có 1 led sáng duy nhất nên ngõ vào của IC dễ dàng đáp ứng nên led sẽ sáng rõ, lý do thứ 2 là chỉ có 8 hàng nên thời gian sáng là 1/8 so với quét cột là 1/24 – đây chính là ưu điểm của quét hàng.

Khuyết điểm của quét hàng là phải xử lý mã kí tự cần hiển thị: một kí tự gồm 5 byte:

Ở quét cột thì lấy byte thứ 1 gởi ra hàng – cho cột thứ 1 sáng, tương tự cho đèn byte thứ 5 và cột thứ 5 sáng.

Ở quét hàng thì lấy bit thứ 0 của byte thứ 1 gởi ra cột thứ 1, lấy bit thứ 0 của byte thứ 2 gởi ra cột thứ 2, lấy bit thứ 0 của byte thứ 3 gởi ra cột thứ 3, lấy bit thứ 0 của byte thứ 4 gởi ra cột

thứ 4, lấy bit thứ 0 của byte thứ 5 gởi ra cột thứ 5 – cho hàng thứ 0 sáng. Tại thời điểm này thì hàng thứ 0 của kí tự sáng.

Tiếp tục: lấy bit thứ 1 của byte thứ 1 gởi ra cột thứ 1, lấy bit thứ 1 của byte thứ 2 gởi ra cột thứ 2, lấy bit thứ 1 của byte thứ 3 gởi ra cột thứ 3, lấy bit thứ 1 của byte thứ 4 gởi ra cột thứ 4, lấy bit thứ 1 của byte thứ 5 gởi ra cột thứ 5 – cho hàng thứ 1 sáng. Tại thời điểm này thì hàng thứ 1 của kí tự sáng.

Tương tự cho bit thứ 2 của các byte – hàng thứ 2 sáng, bit thứ 3 của các byte – hàng thứ 3 sáng, bit thứ 4 của các byte – hàng thứ 4 sáng, ..., bit thứ 7 của các byte – hàng thứ 7 sáng.

Do có 3 led 24 cột thì phải thực hiện 24 byte.

Mã quét hàng: mỗi một thời điểm chỉ cho 1 hàng ở mức logic 1, các hàng còn lại bằng 0.

Bảng 10-6. Mã quét hàng.

Bit	0	1	2	3	4	5	6	7	Số hex
B4(1)				1					10
B5(2)			1						20
B6(4)		1							40
B7(8)	1								80
B3(8)					1				08
B2(4)						1			04
B1(2)							1		02
B0(1)								1	01

Mã quét hàng theo sơ đồ mạch: {0x01,0x02,0x04,0x080,0x80,0x40,0x20,0x10}

Do có 3 led 24 cột xanh và 24 cột đỏ nên tổng số cột là 48 cột, số byte cần để lưu dữ liệu cho 48 cột là 48 byte từ byte thứ 0 cho cột thứ 0 đến byte thứ 47 cho cột thứ 47.

Trong các chương trình điều khiển quét hàng ta phải khai báo mảng có 48 byte để chứa dữ liệu dịch ra cột: 24 byte cho cột màu đỏ, 24 byte cho cột màu xanh.

Do kết nối phần cứng thuận cho việc kết nối dây nên các cột không theo thứ tự liên tục mà chúng theo thứ tự như bảng sau:

Bảng 10-7. Bảng thứ tự các cột màu đỏ.

TT 48 BYTE GỎI RA CỘT	00	01	02	03	04	05	06	07
CỘT LED1	CD3		CD2		CD1		CD0	
TT 48 BYTE GỎI RA CỘT	08	09	10	11	12	13	14	15
CỘT LED1		CD4		CD5		CD6		CD7

TT 48 BYTE GỎI RA CỘT	16	17	18	19	20	21	22	23
CỘT LED2	CD11		CD10		CD9		CD8	
TT 48 BYTE GỎI RA CỘT	24	25	26	27	28	29	30	31
CỘT LED2		CD12		CD13		CD14		CD15
TT 48 BYTE GỎI RA CỘT	32	33	34	35	36	37	38	39
CỘT LED3	CD19		CD18		CD17		CD16	
TT 48 BYTE GỎI RA CỘT	40	41	42	43	44	45	46	47
CỘT LED3		CD20		CD21		CD22		CD23

Giải thích cho các thông tin trong bảng:

Dữ liệu của byte thứ 0 (00) khi dịch ra cột thì sẽ làm cột thứ 3 của led đỏ sáng (CD3).

Dữ liệu của byte thứ 2 (02) khi dịch ra cột thì sẽ làm cột thứ 2 của led đỏ sáng (CD2).

Dữ liệu của byte thứ 4 (04) khi dịch ra cột thì sẽ làm cột thứ 1 của led đỏ sáng (CD1).

Dữ liệu của byte thứ 6 (06) khi dịch ra cột thì sẽ làm cột thứ 0 của led đỏ sáng (CD0).

Dữ liệu của byte thứ 9 (09) khi dịch ra cột thì sẽ làm cột thứ 4 của led đỏ sáng (CD4).

Tương tự cho các cột đỏ còn lại, các ô trống là của 24 cột màu xanh – xem bảng tiếp theo.

Bảng 10-8. Bảng thứ tự các cột màu xanh.

TT 48 BYTE GỎI RA CỘT	00	01	02	03	04	05	06	07
LED1		CX3		CX2		CX1		CX0
TT 48 BYTE GỎI RA CỘT	08	09	10	11	12	13	14	15
LED1	CX4		CX5		CX6		CX7	
TT 48 BYTE GỎI RA CỘT	16	17	18	19	20	21	22	23
LED2		CX11		CX10		CX9		CX8
TT 48 BYTE GỎI RA CỘT	24	25	26	27	28	29	30	31
LED2	CX12		CX13		CX14		CX15	
TT 48 BYTE GỎI RA CỘT	32	33	34	35	36	37	38	39
LED3		CX19		CX18		CX17		CX16
TT 48 BYTE GỎI RA CỘT	40	41	42	43	44	45	46	47
LED3	CX20		CX21		CX22		CX23	

Ví dụ 10-1: Yêu cầu muốn hiển thị một chuỗi 4 ký tự “ABCD” trên 3 led ma trận 8×8, sáng màu đỏ. Mỗi ký tự có 5 byte và 1 byte khoảng cách giữa 2 ký tự liên tiếp nên tổng là 6 byte, 4 ký tự thì tổng là 24 byte.

Chuỗi mã của 4 kí tự như sau:

0x1F,0x24,0x44,0x24,0x1F,0x00: sẽ hiển thị từ cột thứ 0 đến cột thứ 5

0x7F,0x49,0x49,0x49,0x36,0x00: sẽ hiển thị từ cột thứ 6 đến cột thứ 11

0x3E,0x41,0x41,0x41,0x22,0x00: sẽ hiển thị từ cột thứ 12 đến cột thứ 17

0x7F,0x41,0x41,0x41,0x3E,0x00: sẽ hiển thị từ cột thứ 18 đến cột thứ 23

24 byte của 4 kí tự trên sẽ theo thứ tự tương ứng với 24 cột màu đỏ nhưng do thứ tự kết nối không liên tục nên cần phải copy chuỗi dữ liệu 24 byte của 4 kí tự ABCD ở trên tương ứng với thứ tự 48 byte gởi ra cột như sau:

Byte thứ 0 của kí tự A là 0x1F sẽ gán cho byte thứ 06 của mảng 48 byte, xem bảng 10-7.

Byte thứ 1 của kí tự A là 0x24 sẽ gán cho byte thứ 04 của mảng 48 byte, xem bảng 10-7.

Tương tự cho các byte còn lại.

Thư viện điều khiển led ma trận theo kiểu quét hàng. Chương trình thư viện có tên là:

“TV_PICKIT2_SHIFT_MATRAN_QUET_HANG.c”.

Đã có trong thư mục của bạn, bạn có thể sử dụng cho các bài thực hành led ma trận theo kiểu quét hàng và nêu đọc hiểu.

```
#include <tv_pickit2_shift_font.c>
const unsigned int8
ma_quet_hang[8]={0x01,0x02,0x04,0x08,0x80,0x40,0x20,0x10};
#define slkt_hienthi    40
unsigned int8   dl_hienthi[6*slkt_hienthi];
unsigned int8   slkt;

unsigned int8   dl_cot_cd[48]      ={} ;
unsigned int8   dl_cot_dich[48]     ={} ;
unsigned int8   dl_cot_cd_do[24]    ={} ;
unsigned int8   dl_cot_cd_xanh[24]  ={} ;
unsigned int8   dl_cot_dich_d[24]   ={} ;
unsigned int8   dl_cot_dich_x[24]   ={} ;
unsigned int16 vtkt1;

unsigned int8 doi_1_bit_1_sang_thutu(unsigned int8 bx)
{
    unsigned int8 ttcot=0;
    bx= bx>>1;
    while(bx!=0)
    {
        ttcot++;
        bx= bx>>1;
    }
    return(ttcot);
}

void xoa_dl_hienthi()
```

```

{
    unsigned int8 j;
    for(j=0;j<6*slkt_hienthi;j++)
        dl_hienthi[j]=0;
}

void lay_ma_1_so_matran(unsigned int8 so, unsigned int8 vitri_luu)
{
    unsigned int8 n,j;
    n=0;
    so = so + 26;
    for(j=0;j<6 ;j++,n++)
        dl_hienthi[n+vitri_luu*6] = matran_kytu[so][j];
}

void lay_so_luong_ky_tu_chuoi()
{
    slkt=0;
    while(chuoi_hienthi[slkt])
    {
        slkt++;
    }
    xuat_4led_7doan_giaima_xoa_so0(slkt);
}

void lay_ma_ky_tu(unsigned int8 slkt_x)
{
    unsigned int8 kytu,i,n,j;
    n=0;
    for (i=0; i<slkt_x; i++)
    {
        kytu = chuoi_hienthi[i];
        if ((kytu>=0x41)&&(kytu<=0x5a)) kytu = kytu - 0x41;
        else if ((kytu>=0x30)&&(kytu<=0x39)) kytu = (kytu - 0x30) + 26;
        for(j=0;j<6 ;j++,n++)
            dl_hienthi[n]=matran_kytu[kytu][j];
    }
}

void tinh_vi_tri_ky_tu(unsigned int16 k)
{
    unsigned int8 y;
    y=k/6;
    vtktl=(y*6)+5;
}
//sap xep cac ky tu mau xanh 1,2,3,4
void ky_tu_xanh_so_1(unsigned int8 k)
{
    dl_cot_cd[1]=dl_hienthi[k+3];    dl_cot_cd[3]=dl_hienthi[k+2];
    dl_cot_cd[5]=dl_hienthi[k+1];    dl_cot_cd[7]=dl_hienthi[k+0];
    dl_cot_cd[8]=dl_hienthi[k+4];    dl_cot_cd[10]=dl_hienthi[k+5];
}
void ky_tu_xanh_so_2(unsigned int8 k)
{
}

```

```

        dl_cot_cd[12]=dl_hienthi[k+6];    dl_cot_cd[14]=dl_hienthi[k+7];
        dl_cot_cd[17]=dl_hienthi[k+11];   dl_cot_cd[19]=dl_hienthi[k+10];
        dl_cot_cd[21]=dl_hienthi[k+9];    dl_cot_cd[23]=dl_hienthi[k+8];
}
void ky_tu_xanh_so_3(unsigned int8 k)
{
    dl_cot_cd[24]=dl_hienthi[k+12];   dl_cot_cd[26]=dl_hienthi[k+13];
    dl_cot_cd[28]=dl_hienthi[k+14];   dl_cot_cd[30]=dl_hienthi[k+15];
    dl_cot_cd[37]=dl_hienthi[k+17];   dl_cot_cd[39]=dl_hienthi[k+16];
}
void ky_tu_xanh_so_4(unsigned int8 k)
{
    dl_cot_cd[33]=dl_hienthi[k+19];   dl_cot_cd[35]=dl_hienthi[k+18];
    dl_cot_cd[40]=dl_hienthi[k+20];   dl_cot_cd[42]=dl_hienthi[k+21];
    dl_cot_cd[44]=dl_hienthi[k+22];   dl_cot_cd[46]=dl_hienthi[k+23];
}

//sap xep cac ky tu mau do 1,2,3,4
void ky_tu_do_so_1(unsigned int8 k)
{
    dl_cot_cd[0]=dl_hienthi[k+3];    dl_cot_cd[2]=dl_hienthi[k+2];
    dl_cot_cd[4]=dl_hienthi[k+1];    dl_cot_cd[6]=dl_hienthi[k+0];
    dl_cot_cd[9]=dl_hienthi[k+4];    dl_cot_cd[11]=dl_hienthi[k+5];
}
void ky_tu_do_so_2(unsigned int8 k)
{
    dl_cot_cd[13]=dl_hienthi[k+6];   dl_cot_cd[15]=dl_hienthi[k+7];
    dl_cot_cd[16]=dl_hienthi[k+11];  dl_cot_cd[18]=dl_hienthi[k+10];
    dl_cot_cd[20]=dl_hienthi[k+9];   dl_cot_cd[22]=dl_hienthi[k+8];
}
void ky_tu_do_so_3(unsigned int8 k)
{
    dl_cot_cd[25]=dl_hienthi[k+12];  dl_cot_cd[27]=dl_hienthi[k+13];
    dl_cot_cd[29]=dl_hienthi[k+14];  dl_cot_cd[31]=dl_hienthi[k+15];
    dl_cot_cd[36]=dl_hienthi[k+17];  dl_cot_cd[38]=dl_hienthi[k+16];
}
void ky_tu_do_so_4(unsigned int8 k)
{
    dl_cot_cd[32]=dl_hienthi[k+19];  dl_cot_cd[34]=dl_hienthi[k+18];
    dl_cot_cd[41]=dl_hienthi[k+20];  dl_cot_cd[43]=dl_hienthi[k+21];
    dl_cot_cd[45]=dl_hienthi[k+22];  dl_cot_cd[47]=dl_hienthi[k+23];
}
//xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
void ky_tu_xanh_hoac_do_so_1(unsigned int8 k)
{
    tinh_vi_tri_ky_tu(k);
    if (dl_hienthi[vktl1]==0)
    {
        dl_cot_cd[1]=dl_hienthi[k+3];    dl_cot_cd[3]=dl_hienthi[k+2];
        dl_cot_cd[5]=dl_hienthi[k+1];    dl_cot_cd[7]=dl_hienthi[k+0];
        dl_cot_cd[8]=dl_hienthi[k+4];    dl_cot_cd[10]=dl_hienthi[k+5];
    }
    else
    {
}

```

```

        dl_cot_cd[0]=dl_hienthi[k+3];    dl_cot_cd[2]=dl_hienthi[k+2];
        dl_cot_cd[4]=dl_hienthi[k+1];    dl_cot_cd[6]=dl_hienthi[k+0];
        dl_cot_cd[9]=dl_hienthi[k+4];    dl_cot_cd[11]=dl_hienthi[k+5];
    }
}

void ky_tu_xanh_hoac_do_so_2(unsigned int8 k)
{
    if (dl_hienthi[vtkt1+6]==0)
    {
        dl_cot_cd[12]=dl_hienthi[k+6];    dl_cot_cd[14]=dl_hienthi[k+7];
        dl_cot_cd[17]=dl_hienthi[k+11];   dl_cot_cd[19]=dl_hienthi[k+10];
        dl_cot_cd[21]=dl_hienthi[k+9];    dl_cot_cd[23]=dl_hienthi[k+8];
    }
    else
    {
        dl_cot_cd[13]=dl_hienthi[k+6];    dl_cot_cd[15]=dl_hienthi[k+7];
        dl_cot_cd[16]=dl_hienthi[k+11];   dl_cot_cd[18]=dl_hienthi[k+10];
        dl_cot_cd[20]=dl_hienthi[k+9];    dl_cot_cd[22]=dl_hienthi[k+8];
    }
}
void ky_tu_xanh_hoac_do_so_3(unsigned int8 k)
{
    if (dl_hienthi[vtkt1+12]==0)
    {
        dl_cot_cd[24]=dl_hienthi[k+12];   dl_cot_cd[26]=dl_hienthi[k+13];
        dl_cot_cd[28]=dl_hienthi[k+14];   dl_cot_cd[30]=dl_hienthi[k+15];
        dl_cot_cd[37]=dl_hienthi[k+17];   dl_cot_cd[39]=dl_hienthi[k+16];
    }
    else
    {
        dl_cot_cd[25]=dl_hienthi[k+12];   dl_cot_cd[27]=dl_hienthi[k+13];
        dl_cot_cd[29]=dl_hienthi[k+14];   dl_cot_cd[31]=dl_hienthi[k+15];
        dl_cot_cd[36]=dl_hienthi[k+17];   dl_cot_cd[38]=dl_hienthi[k+16];
    }
}
void ky_tu_xanh_hoac_do_so_4(unsigned int8 k)
{
    if (dl_hienthi[vtkt1+18]==0)
    {
        dl_cot_cd[33]=dl_hienthi[k+19];   dl_cot_cd[35]=dl_hienthi[k+18];
        dl_cot_cd[40]=dl_hienthi[k+20];   dl_cot_cd[42]=dl_hienthi[k+21];
        dl_cot_cd[44]=dl_hienthi[k+22];   dl_cot_cd[46]=dl_hienthi[k+23];
    }
    else
    {
        dl_cot_cd[32]=dl_hienthi[k+19];   dl_cot_cd[34]=dl_hienthi[k+18];
        dl_cot_cd[41]=dl_hienthi[k+20];   dl_cot_cd[43]=dl_hienthi[k+21];
        dl_cot_cd[45]=dl_hienthi[k+22];   dl_cot_cd[47]=dl_hienthi[k+23];
    }
}

void sapxep_vitri_48byte_mau_theo_kytu(unsigned int8 k1)
{
    ky_tu_xanh_hoac_do_so_1(k1);
}

```

```

ky_tu_xanh_hoac_do_so_2(k1);
ky_tu_xanh_hoac_do_so_3(k1);
ky_tu_xanh_hoac_do_so_4(k1);
}
//xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
void sapxep_vitri_48byte_cot_xanh(unsigned int8 k1)
{
    ky_tu_xanh_so_1(k1);    ky_tu_xanh_so_2(k1);
    ky_tu_xanh_so_3(k1);    ky_tu_xanh_so_4(k1);
}
void sapxep_vitri_48byte_cot_do(unsigned int8 k1)
{
    ky_tu_do_so_1(k1);    ky_tu_do_so_2(k1);
    ky_tu_do_so_3(k1);    ky_tu_do_so_4(k1);
}

void sapxep_vitri_xanh_do_xanh_do(unsigned int8 k1)
{
    ky_tu_xanh_so_1(k1);    ky_tu_do_so_2(k1);
    ky_tu_xanh_so_3(k1);    ky_tu_do_so_4(k1);
}
void sapxep_vitri_do_xanh_do_xanh(unsigned int8 k1)
{
    ky_tu_do_so_1(k1);    ky_tu_xanh_so_2(k1);
    ky_tu_do_so_3(k1);    ky_tu_xanh_so_4(k1);
}

void sapxep_vitri_xanh_xanh_do_do(unsigned int8 k1)
{
    ky_tu_xanh_so_1(k1);    ky_tu_xanh_so_2(k1);
    ky_tu_do_so_3(k1);    ky_tu_do_so_4(k1);
}
void sapxep_vitri_do_do_xanh_xanh(unsigned int8 k1)
{
    ky_tu_do_so_1(k1);    ky_tu_do_so_2(k1);
    ky_tu_xanh_so_3(k1);    ky_tu_xanh_so_4(k1);
}

void xoa_dl_48_cot()
{
    unsigned int8 j1;
    for (j1=0;j1<48;j1++) dl_cot_cd[j1]=0;
}
//xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
void copy_48_cot_hienthi()
{
    unsigned int8 i;
    for (i=0;i<48;i++) dl_cot_dich[i]=dl_cot_cd[i];
}

void xuat_1bit_cua_48_cot_hienthi()
{
    unsigned int8 i;
    for (i=0;i<48;i++) dl_cot_dich[i]=xuat_1bit(dl_cot_dich[i]);
}

```

```

}

//xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
void copy_24_cot_hienthi_xanh(unsigned int8 k1)
{
    unsigned int8 i;
    for (i=0;i<24;i++)
        dl_cot_cd_xanh[i+k1]= dl_hienthi[i+k1];
}
//xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
void copy_24_cot_hienthi_do(unsigned int8 k1)
{
    unsigned int8 i;
    for (i=0;i<24;i++)
        dl_cot_cd_do[i+k1]= dl_hienthi[i+k1];
}
//xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
void copy_24_cot_hienthi_xanh_do_de_dich()
{
    unsigned int8 i;
    for (i=0;i<48;i++)
    {
        dl_cot_dich_d[i]= dl_cot_cd_do[i];
        dl_cot_dich_x[i]= dl_cot_cd_xanh[i];
    }
}
void xoa_dl_24_cot_xanh_do()
{
    unsigned int8 j1;
    for (j1=0;j1<24;j1++)
    {
        dl_cot_cd_do[j1]=0;
        dl_cot_cd_xanh[j1]=0;
    }
}
void xuat_dulieu_hienthi_matran()
{
    mo_ic_74573_b_thong_dl();
    output_high(rck_matranled);
    output_low(rck_matranled);
    mo_led_matran;
    chot_ic_74573_b_goi_du_lieu;
}

void xuat_1bit_cua_48_cot_hienthi_c2()
{
    signed ix;
    for(ix=3;ix>-1;ix--)
    {
        dl_cot_dich_d[ix]=xuat_1bit(dl_cot_dich_d[ix]);
        dl_cot_dich_x[ix]=xuat_1bit(dl_cot_dich_x[ix]);
    }
    for(ix=4;ix<8;ix++)
    {
        dl_cot_dich_d[ix]=xuat_1bit(dl_cot_dich_d[ix]);
    }
}

```

```

        dl_cot_dich_x[ix]=xuat_1bit(dl_cot_dich_x[ix]);
    }
    for(ix=11;ix>7;ix--)
    {
        dl_cot_dich_d[ix]=xuat_1bit(dl_cot_dich_d[ix]);
        dl_cot_dich_x[ix]=xuat_1bit(dl_cot_dich_x[ix]);
    }
    for(ix=12;ix<16;ix++)
    {
        dl_cot_dich_d[ix]=xuat_1bit(dl_cot_dich_d[ix]);
        dl_cot_dich_x[ix]=xuat_1bit(dl_cot_dich_x[ix]);
    }

    for(ix=19;ix>15;ix--)
    {
        dl_cot_dich_d[ix]=xuat_1bit(dl_cot_dich_d[ix]);
        dl_cot_dich_x[ix]=xuat_1bit(dl_cot_dich_x[ix]);
    }
    for(ix=20;ix<24;ix++)
    {
        dl_cot_dich_d[ix]=xuat_1bit(dl_cot_dich_d[ix]);
        dl_cot_dich_x[ix]=xuat_1bit(dl_cot_dich_x[ix]);
    }
}

void hien_thi_ma_tran_quet_hang(unsigned int8 ck)
{
    unsigned int8 j,i;
    for (i=0;i<ck;i++)
    {
        copy_48_cot_hienthi();
        for(j=0;j<8;j++)
        {
            xuat_1bit_cua_48_cot_hienthi();
            xuat_2byte(ma_quet_hang[j]);
            xuat_dulieu_hienthi_matran();
            delay_us(500);
            xuat_matranled(0,0,0,0);
        }
    }
}

```

Với 3 led ma trận 8×8 có 24 cột, mỗi kí tự 6 byte nên có thể hiển thị được 4 kí tự, việc copy dữ liệu sẽ được chia ra cho từng kí tự độc lập để dễ điều khiển.

4 kí tự màu xanh và 4 kí tự màu đỏ, mỗi kí tự là 1 chương trình con màu xanh hoặc màu đỏ, các chương trình con sắp xếp lại cho cả 4 kí có thể thực hiện các màu như sau :

- 4 kí tự màu xanh.
- 4 kí tự màu đỏ.
- 4 kí tự: xanh xanh đỏ đỏ.
- 4 kí tự: đỏ đỏ xanh xanh.
- 4 kí tự xen kẽ: đỏ xanh đỏ xanh.

- 4 kí tự xen kẽ: xanh đỏ xanh đỏ.

Trong các chương trình con copy để sắp xếp lại đúng thứ tự cột thì có thêm tham số K với mục đích để điều khiển dịch chuỗi trên led ma trận:

Khi K = 0 thì sẽ copy dữ liệu từ byte thứ 0 đến byte thứ 47 đem gởi ra led hiển thị.

Khi K = 1 thì sẽ copy dữ liệu từ byte thứ 1 đến byte thứ 48 đem gởi ra led hiển thị - so với K = 0 thì xem như chuỗi hiển thị dịch đi 1 cột.

Khi K = 2 thì sẽ copy dữ liệu từ byte thứ 2 đến byte thứ 49 đem gởi ra led hiển thị - so với K = 1 thì xem như chuỗi hiển thị dịch đi thêm 1 cột nữa.

Tương tự cho đến khi hết chuỗi cần hiển thị.

Các bài hiển thị chuỗi đứng yên thì cho tham số K = 0.

10.5 CÁC CHƯƠNG TRÌNH ĐIỀU KHIỂN LED MA TRẬN QUÉT HÀNG

Bài mẫu 1011. Chương trình điều khiển LED ma trận hiển thị chữ ABCD màu xanh theo phương pháp quét hàng.

Lưu tên file là “BAI_1011_MTQH_ABCD_XANH”

- Mục đích: biết cách điều khiển led quét hàng.
- Lưu đồ: sinh viên hãy tự viết.
- Chương trình:

```
#include <tv_pickit2_shift_1.c>
const unsigned char chuoi_hienthi []={"abcd"};
#include <tv_pickit2_shift_matran_quet_hang.c>

void main()
{
    set_up_port_ic_chot();
    lay_so_luong_ky_tu_chuoi();
    lay_ma_ky_tu(slkt);
    xoa_dl_48_cot();
    sapxep_vitri_48byte_cot_xanh(0);
    while(true)
    {
        hien_thi_ma_tran_quet_hang(50);
    }
}
```

- Tiến hành biên dịch và nạp.
- Quan sát kết quả: sau khi nạp xong thì LED sáng chữ ABCD rõ đẹp hơn so với quét cột.
- Giải thích chương trình:

Bài tập 1012. Hãy hiệu chỉnh chương trình bài 1011 để 4 kí tự sáng màu đỏ.

Lưu tên file là “BAI_1012_MTQH_ABCD_DO”

GỢI Ý :

Thay sapxep_vitri_48byte_cot_xanh(0); bằng ham sapxep_vitri_48byte_cot_do(0);

Bài mẫu 1013. Chương trình điều khiển LED ma trận hiển thị chữ ABCD màu xanh rồi màu đỏ.

Lưu tên file là “BAI_1013_MTQH_ABCD_XANH_DO”

- Mục đích: biết cách điều khiển led quét hàng hiển thị 2 màu.
- Lưu đồ: sinh viên hãy tự viết.
- Chương trình:

```
#include <tv_pickit2_shift_1.c>
const unsigned char chuo_i_hienthi [] = {"abcd"};
#include <tv_pickit2_shift_matran_quet_hang.c>
void main()
{
    set_up_port_ic_chot();
    lay_so_luong_ky_tu_chuo_i();
    lay_ma_ky_tu(slkt);

    while(true)
    {
        xoa_dl_48_cot();
        sapxep_vitri_48byte_cot_xanh(0);
        hien_thi_ma_tran_quet_hang(50);

        xoa_dl_48_cot();
        sapxep_vitri_48byte_cot_do(0);
        hien_thi_ma_tran_quet_hang(50);
    }
}
```

- Tiến hành biên dịch và nạp.
- Quan sát kết quả: sau khi nạp xong thì LED sáng chữ ABCD màu xanh rồi đến màu đỏ.
- Giải thích chương trình:

Bài mẫu 1014. Chương trình điều khiển LED ma trận hiển thị chuỗi màu xanh dịch từ phải sang trái.

Lưu tên file là “BAI_1014_MTQH_CHUOI_XANH_DICH”

- Mục đích: biết cách điều khiển led quét hàng hiển thị chuỗi dịch trái.
- Lưu đồ: sinh viên hãy tự viết.
- Chương trình:

```
#include <tv_pickit2_shift_1.c>
const unsigned char chuo_i_hienthi [] = {"abcdefghijklm"};
#include <tv_pickit2_shift_matran_quet_hang.c>
unsigned int8 vitri;
void matran_qh_dich_mau_xanh(unsigned int8 i)
{
    xoa_dl_48_cot();
    sapxep_vitri_48byte_cot_xanh(i);
```

```

        hien_thi_ma_tran_quet_hang(50);
    }
void main()
{
    set_up_port_ic_chot();
    lay_so_luong_ky_tu_chuoi();
    lay_ma_ky_tu(slkt);

    while(true)
    {
        vitri=0;
        for(vitri=0; vitri<(slkt-4)*6; vitri++)
            matran_qh_dich_mau_xanh(vitri);
    }
}

```

- d. Tiến hành biên dịch và nạp.
- e. Quan sát kết quả: sau khi nạp xong thì LED sáng chuỗi sẽ dịch chuyển.
- f. Giải thích chương trình:

Bài tập 1015. Chương trình điều khiển LED ma trận hiển thị chữ ABCD dịch từ phải sang trái màu xanh rồi đến màu đỏ.

Lưu tên file là “BAI_1015_MTQH_ABCD_XANH_DO_DICH”

Bài tập 1016. Chương trình điều khiển LED ma trận hiển thị chữ ABCD dịch từ phải sang trái màu xanh rồi đến màu đỏ và màu cam.

Lưu tên file là “BAI_1016_MTQH_ABCD_XANH_DO_CAM_DICH”

Bài mẫu 1017. Chương trình điều khiển LED ma trận đếm 1 số từ 0 đến 9.

Lưu tên file là “BAI_1017_MTQH_DEM_1SO_XANH”

- a. Mục đích: biết cách điều khiển led quét hàng hiển thị giá trị đếm 1 số.
- b. Lưu đồ: sinh viên hãy tự viết.
- c. Chương trình:

```

#include <tv_pickit2_shift_1.c>
const unsigned char chuoi_hienthi[]{"1234"};
#include <tv_pickit2_shift_matran_quet_hang.c>
unsigned int8 i;
void main()
{
    set_up_port_ic_chot();
    //lay_so_luong_ky_tu_chuoi();
    xoa_dl_hienthi();
    slkt=1;
    while(true)
    {

```

```

for(i=0; i<10; i++)
{
    lay_ma_1_so_matran(i,3);
    xoa_dl_48_cot();
    sapxep_vitri_48byte_cot_xanh(0);
    hien_thi_ma_tran_quet_hang(100);
}
}
}

```

- d. Tiến hành biên dịch và nạp.
- e. Quan sát kết quả: sau khi nạp xong thì LED hiển thị giá trị đếm từ 0 đến 9.
- f. Giải thích chương trình:

Bài tập 1018. Hãy viết chương trình đếm 2 số từ 00 đến 99.

Lưu tên file là “BAI_1018_MTQH_DEM_2SO_XANH”

Bài tập 1019. Hãy viết chương trình đếm 3 số từ 000 đến 999.

Lưu tên file là “BAI_1019_MTQH_DEM_3SO_XANH”

Bài tập 1020. Hãy viết chương trình đếm sản phẩm dùng timer T0 hiển thị giá trị đếm trên led ma trận, giới hạn đếm từ 00 đến 99.

Lưu tên file là “BAI_1020_MTQH_DEM_SP_T0”

Bài tập 1021. Hãy viết chương trình đo nhiệt độ hiển thị trên 2 led ma trận.

Lưu tên file là “BAI_1021_MTQH_ADC_LM35”

TÀI LIỆU THAM KHẢO

- [1]. MICROCHIP, PIC 16F87X DATASHEET, 1997.
- [2]. MICROCHIP, PIC 18F4550 DATASHEET, 2009.
- [3]. Nebojsa Matic, PIC microcontrollers for beginner, too.
- [4]. ITead – Studio Ultrasonic ranging module: HC-SR04, 2010.
- [5]. ATMEL – AT24C512 DATASHEET, 2007.
- [6]. SHARP – GP2D12 optoelectronic Device DATASHEET, 2007.
- [7]. MICROCHIP – PICmicro Mid- Range MCU family reference Manual, 1997
- [8]. Nguyễn Đình Phú, Vi xử lý - PIC, Đại Học Sư Phạm Kỹ Thuật TP HCM, 2017.