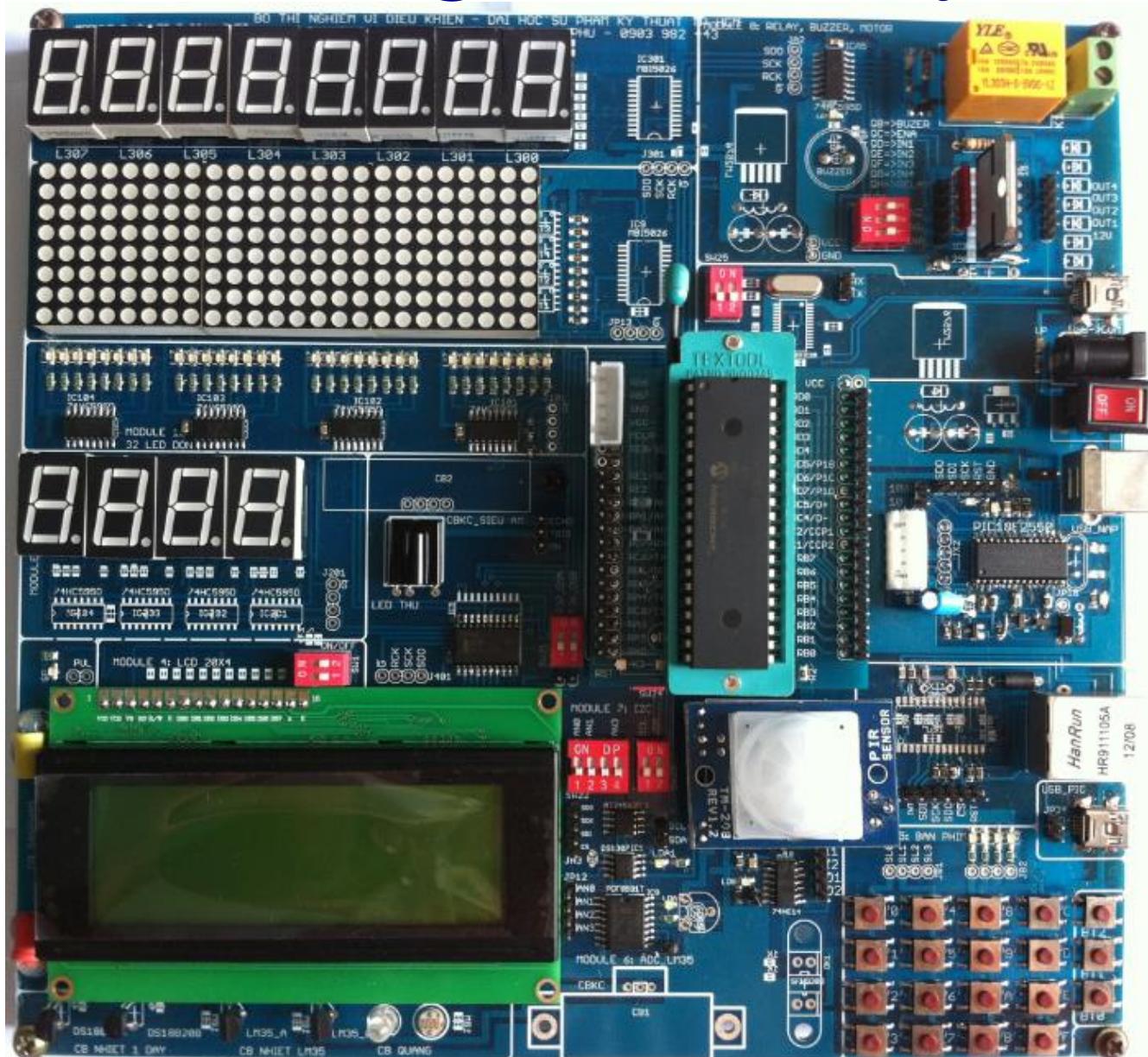


ĐẠI HỌC SƯ PHẠM KỸ THUẬT
KHOA ĐIỆN – ĐIỆN TỬ – BỘ MÔN ĐIỆN TỬ CÔNG NGHIỆP

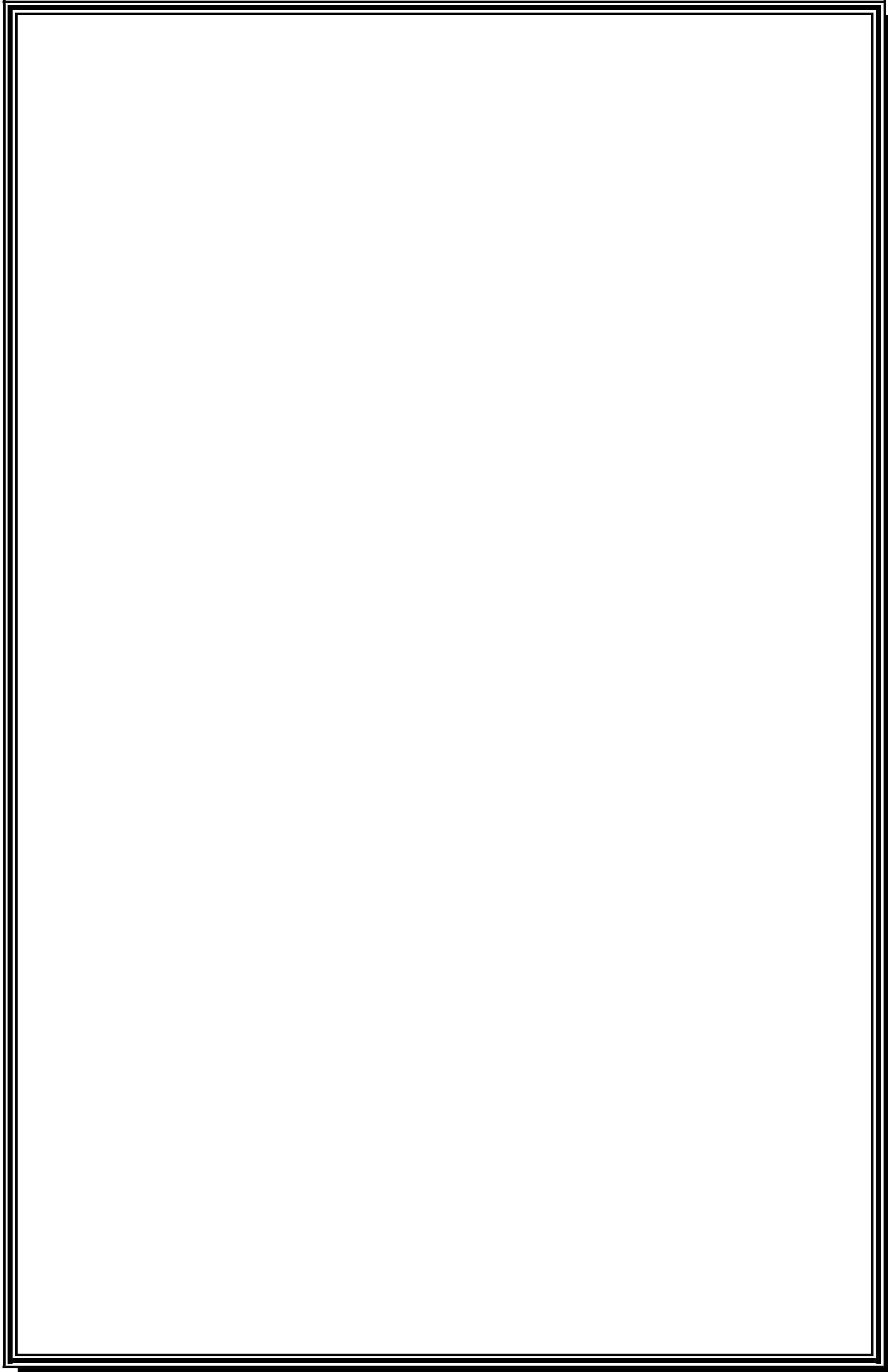
GIÁO TRÌNH VI XỬ LÝ

VI ĐIỀU KHIỂN PIC



PIC16F887

NGUYỄN ĐÌNH PHÚ
TP HCM 08/2016



LỜI NÓI ĐẦU

Bộ vi xử lý ngày càng phát triển hoàn thiện và được sử dụng hầu hết trong các hệ thống điều khiển trong công nghiệp cũng như trong các thiết bị điện tử dân dụng. Chính nhờ vai trò, chức năng của vi xử lý đã đem lại nhiều ưu điểm, nhiều tính năng đặc biệt cho các hệ thống điều khiển.

Các nhà nghiên cứu không ngừng nghiên cứu các hệ thống điều khiển và sử dụng vi xử lý để thay thế nhằm nâng cao khả năng tự động thay thế cho con người, và cũng chính vì thế đã thúc đẩy lĩnh vực vi xử lý ngày càng phát triển không ngừng, thích nghi với yêu cầu điều khiển. Để đơn giản hóa sự phức tạp của phần cứng khi dùng vi xử lý nên các nhà nghiên cứu đã tích hợp hệ vi xử lý, bộ nhớ, các ngoại vi thành một vi mạch duy nhất gọi là vi điều khiển.

Nội dung giáo trình này nghiên cứu các kiến thức cơ bản của vi điều khiển. Do có nhiều họ vi điều khiển khác nhau, từ hệ 8 bit cho đến hệ 32 bit, mức độ tích hợp từ đơn giản đến phức tạp, nhiều hãng chế tạo khác nhau, nhiều chủng loại khác nhau sẽ làm cho người bắt đầu học hay nghiên cứu sẽ gặp nhiều bỡ ngỡ không biết bắt đầu từ hệ nào cho phù hợp, chính vì thế giáo trình này chỉ trình bày họ vi điều khiển 8 bit của hãng Microchip nhằm giúp các bạn sinh viên ngành điện tử chung có một giáo trình để học tập và nghiên cứu một cách dễ dàng.

Các ứng dụng dùng vi điều khiển ở nhiều cấp độ khác nhau từ đơn giản đến phức tạp, giáo trình này chỉ trình bày các ứng dụng đơn giản để các bạn có thể đọc hiểu, từ các kiến thức cơ bản đó bạn có thể thực hiện các ứng dụng điều khiển phức tạp hơn, phần bài tập kèm theo giúp bạn giải quyết các yêu cầu phức tạp. Từ các kiến thức cơ bản của vi điều khiển 8 bit có thể giúp các bạn tự nghiên cứu các vi điều khiển nhiều bit hơn như 16 bit, 32 bit.

Giáo trình biên soạn chia thành 10 chương, chủ yếu trình bày vi điều khiển PIC 16F887:

Chương 1. Đặc tính, cấu trúc, chức năng các port.

Chương 2. Tổ chức bộ nhớ, thanh ghi.

Chương 3. Lệnh hợp ngữ.

Chương 4. Ngôn ngữ lập trình C.

Chương 5. Giao tiếp LED, LCD, phím đơn, ma trận phím.

Chương 6. Timer - Counter.

Chương 7. Chuyển đổi tín hiệu tương tự sang số.

Chương 8. Ngắt.

Chương 9. Điều chế độ rộng xung - PWM.

Chương 10. Truyền dữ liệu UART.

Nội dung chương 1 chủ yếu giới thiệu đặc tính, cấu trúc và chức năng các port của vi điều khiển, người đọc cần phải biết đặc tính của vi điều khiển đang nghiên cứu. Để so sánh khả năng của các vi điều khiển khác nhau ta phải dựa vào đặc tính. Phần cấu trúc bên trong cho bạn biết được tổ chức, mối quan hệ giữa các khối với nhau, chức năng của từng khối. Bạn phải biết tên, ký hiệu đặt tên cho từng port, chức năng của từng port để giúp bạn sử dụng port kết nối đúng với các đối tượng điều khiển.

Nội dung chương 2 giới thiệu cấu trúc tổ chức các loại bộ nhớ tích hợp bên trong vi điều khiển bao gồm bộ nhớ chương trình, bộ nhớ dữ liệu RAM, bộ nhớ ngăn xếp, bộ nhớ Eeprom, các cách truy xuất bộ nhớ.

Nội dung chương 3 giới thiệu về tập lệnh hợp ngữ của vi điều khiển để viết các chương trình bằng hợp ngữ nhưng do lập trình bằng hợp ngữ rất khó và dài khi giải quyết các yêu cầu tính toán phức tạp nên phần này chỉ giới thiệu chứ không nghiên cứu sâu.

Nội dung chương 4 giới thiệu về ngôn ngữ lập trình C cho vi điều khiển PIC, có nhiều trình biên dịch ngôn ngữ lập trình C cho vi điều khiển nhưng tài liệu này trình bày trình biên dịch CCS. Lập trình bằng ngôn ngữ C giúp các bạn viết chương trình dễ hơn so với hợp ngữ, toàn bộ các chương trình trong tài liệu này đều viết bằng ngôn ngữ lập trình C. Để hiểu các chương trình trong giáo trình và viết các chương trình theo yêu cầu thì bạn cần phải nắm rõ tổ chức của moat chương trình C, các kiểu dữ liệu, các toán tử, các thư viện viết sẵn và các lệnh C cơ bản.

Nội dung chương 5 khảo sát chi tiết chức năng các port, sơ đồ mạch của các port, sử dụng các port để xuất nhập tín hiệu điều khiển như led đơn, led 7 đoạn trực tiếp, led 7 đoạn quét, LCD, nút nhấn, bàn phím ma trận. Trong từng yêu cầu sẽ cho bạn biết cách kết nối phần cứng, nguyên lý hoạt động, viết lưu đồ hay trình tự điều khiển và chương trình mẫu, có giải thích từng lệnh hoặc cả chương trình.

Nội dung chương 6 khảo sát chi tiết chức năng của timer-counter tích hợp trong vi điều khiển, cách sử dụng timer – counter để định thời và đếm sự kiện.

Nội dung chương 7 khảo sát chi tiết chức năng của bộ chuyển đổi tín hiệu tương tự thành tín hiệu số (ADC) tích hợp trong vi điều khiển, cách sử dụng ADC để chuyển đổi các tín hiệu tương tự như cảm biến nhiệt để thực hiện các ứng dụng đo nhiệt độ, cảnh báo quá nhiệt độ trong điều khiển và nhiều ứng dụng khác.

Nội dung chương 8 khảo sát chi tiết chức năng ngắn của vi điều khiển, biết được tính năng ưu việt của ngắn, cách sử dụng ngắn để đáp ứng tối ưu các yêu cầu điều khiển nhằm đáp ứng nhanh các sự kiện xảy ra.

Nội dung chương 9 khảo sát chi tiết chức năng điều chế độ rộng xung PWM của vi điều khiển, biết được nguyên lý hoạt động, tính toán các thông số của xung điều chế, biết lập trình sử dụng chức năng PWM để điều khiển thay đổi độ sáng của đèn, thay đổi tốc độ của động cơ DC và nhiều ứng dụng khác.

Nội dung chương 10 khảo sát chi tiết chức năng truyền dữ liệu nối tiếp UART của vi điều khiển, biết được trình tự thực hiện gửi dữ liệu và nhận dữ liệu, thực hiện các yêu cầu truyền dữ liệu giữa vi điều khiển với máy tính và giữa các vi điều khiển với nhau.

Ngoài các kiến thức cơ bản mà tác giả đã trình bày thì còn nhiều chức năng khác của vi điều khiển mà tác giả chưa trình bày thì các bạn có thể tham khảo thêm ở các tài liệu nhà chế tạo cung cấp.

Trong quá trình biên soạn không thể tránh được các sai sót nên rất mong các bạn đọc đóng góp xây dựng và xin hãy gửi về tác giả theo địa chỉ phu_nd@yahoo.com.

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 tác giả có nhiều thời gian thực hiện biên soạn giáo trình này.

Nguyễn Đình Phú.

HÌNH VÀ BẢNG

<i>Hình 1-1. Các thiết bị sử dụng vi xử lý.</i>	2
<i>Hình 1-2. Hệ thống vi xử lý.</i>	2
<i>Hình 1-3. Vì điều khiển được tích hợp từ vi xử lý, bộ nhớ và các ngoại vi.</i>	3
<i>Hình 1-4. Các thiết bị vào, ra và vi điều khiển.</i>	4
<i>Hình 1-5. Cấu hình của vi điều khiển.</i>	7
<i>Hình 1-6. Cấu trúc bên trong của vi điều khiển.</i>	8
<i>Hình 1-7. Sơ đồ chân của PIC 16F887.</i>	10
<i>Hình 2-1. Kiến trúc Von Neumann và Harvard.</i>	20
<i>Hình 2-2. Sơ đồ bộ nhớ chương trình và ngăn xếp.</i>	21
<i>Hình 2-3. Nội dung thanh ghi PC khi thực hiện lệnh Call hay Goto.</i>	22
<i>Hình 2-4. Nhảy trong cùng 1 trang và khác trang bộ nhớ.</i>	22
<i>Hình 2-5. Bộ nhớ ngăn xếp khi thực hiện ngắn và kết thúc ngắn.</i>	24
<i>Hình 2-6. Tổ chức bộ nhớ theo byte.</i>	24
<i>Hình 2-7. Tổ chức bộ nhớ chứa cả mã lệnh và dữ liệu.</i>	24
<i>Hình 2-8. Quá trình thực hiện lệnh 3 bước.</i>	25
<i>Hình 2-9. Quá trình thực hiện lệnh 4 bước.</i>	25
<i>Hình 2-10. Quá trình thực hiện lệnh 5 bước.</i>	26
<i>Hình 2-11. Ký hiệu các bước của quá trình thực hiện lệnh 5 bước.</i>	26
<i>Hình 2-12. Các bước thực hiện chi tiết của quá trình thực hiện lệnh 5 bước.</i>	26
<i>Hình 2-13. Làm xong việc này mới đến việc khác.</i>	27
<i>Hình 2-14. Làm theo cấu trúc pipeline hay dây chuyền.</i>	28
<i>Hình 2-15. Minh họa 2 cấu trúc để thấy hiệu quả về thời gian.</i>	29
<i>Hình 2-16. Cấu trúc pipeline của vi điều khiển PIC 16F887.</i>	29
<i>Hình 2-17. Tổ chức File thanh ghi.</i>	30
<i>Hình 2-18. Truy xuất trực tiếp và gián tiếp bộ nhớ RAM.</i>	31
<i>Hình 2-19. Thanh ghi trạng thái.</i>	32
<i>Hình 3-1. Hệ thống điều khiển đèn giao thông – ảnh minh họa.</i>	36
<i>Hình 3-2. Trình tự biên soạn chương trình Assembly cho đèn khi nạp code.</i>	37
<i>Hình 3-3. Logo phần mềm MPLAB.</i>	38
<i>Hình 3-4. Chương trình dùng hợp ngữ.</i>	39
<i>Hình 3-5. Biên dịch chương trình hợp ngữ.</i>	40
<i>Hình 3-6. Biên dịch kết nối nhiều chương trình hợp ngữ.</i>	40
<i>Hình 3-7. Biên dịch chương trình hợp ngữ tạo file hex và file bin.</i>	41
<i>Hình 3-8. Các dạng mã lệnh.</i>	41
<i>Hình 4-1. Biểu tượng phần mềm CCS.</i>	64
<i>Hình 4-2. Giao diện phần mềm CCS.</i>	64

<i>Hình 5-1. Sơ đồ kết nối port với đối tượng điều khiển.</i>	54
<i>Hình 5-2. Sơ đồ kết nối port xuất nhập tín hiệu điều khiển.</i>	54
<i>Hình 5-3. PortA và thanh ghi định hướng port A.</i>	55
<i>Hình 5-4. Cấu hình chân RA0.</i>	56
<i>Hình 5-5. Cấu hình chân RA1.</i>	56
<i>Hình 5-6. Cấu hình chân RA2.</i>	56
<i>Hình 5-7. Cấu hình chân RA3.</i>	56
<i>Hình 5-8. Cấu hình chân RA4.</i>	57
<i>Hình 5-9. Cấu hình chân RA5.</i>	57
<i>Hình 5-10. Cấu hình chân RA6.</i>	57
<i>Hình 5-11. Cấu hình chân RA7.</i>	57
<i>Hình 5-12. PortB và thanh ghi định hướng port B.</i>	58
<i>Hình 5-13. Thanh ghi ANSELH định cấu hình số tương tự cho portB.</i>	58
<i>Hình 5-14. Thanh ghi WPUB thiết lập cho phép/cấm điện trở treo.</i>	58
<i>Hình 5-15. Thanh ghi IOCB cho phép/cấm ngắt portB thay đổi.</i>	59
<i>Hình 5-16. Cấu hình chân RB<3:0>.</i>	60
<i>Hình 5-17. Cấu hình chân RB<7:4>.</i>	61
<i>Hình 5-18. Các chân PortB giao tiếp với mạch nạp, gỡ rời.</i>	62
<i>Hình 5-19. Port C và thanh ghi TRISC.</i>	62
<i>Hình 5-20. Cấu hình chân RC0.</i>	63
<i>Hình 5-21. Cấu hình chân RC1.</i>	63
<i>Hình 5-22. Cấu hình chân RC2.</i>	63
<i>Hình 5-23. Cấu hình chân RC3.</i>	63
<i>Hình 5-24. Cấu hình chân RC4.</i>	64
<i>Hình 5-25. Cấu hình chân RC5.</i>	64
<i>Hình 5-26. Cấu hình chân RC6.</i>	64
<i>Hình 5-27. Cấu hình chân RC7.</i>	64
<i>Hình 5-28. Port D và thanh ghi TRISD.</i>	65
<i>Hình 5-29. Cấu hình chân RD<4:0>.</i>	65
<i>Hình 5-30. Cấu hình chân RD<7:5>.</i>	65
<i>Hình 5-31. Port E và thanh ghi TRISE.</i>	66
<i>Hình 5-32. Cấu hình chân RE<2:0>.</i>	66
<i>Hình 5-33. Cấu hình chân RE<3>.</i>	66
<i>Hình 5-34. Các dạng tín hiệu của nguồn, MCLR, PWRT, OST.</i>	71
<i>Hình 5-35. Các dạng tín hiệu khi bị sụt giảm nguồn BOR.</i>	71
<i>Hình 5-36. Sơ đồ khởi mạch dao động của PIC.</i>	72
<i>Hình 5-37. Dao động RC bên ngoài.</i>	73
<i>Hình 5-38. Dao động RC bên trong.</i>	73
<i>Hình 5-39. Dao động lấy từ bên ngoài.</i>	73

<i>Hình 5-40. Các dạng dao động LP, XT, HS.</i>	74
<i>Hình 5-41. Chức năng của bộ định thời giám sát - WDT.</i>	75
<i>Hình 5-42. Sơ đồ khôi của bộ định thời giám sát - WDT.</i>	76
<i>Hình 5-43. Các tín hiệu của PIC giao tiếp với mạch nạp dạng nối tiếp ICSP.</i>	75
<i>Hình 5-44. Sơ đồ điều khiển led đơn.</i>	77
<i>Hình 5-45. Lưu đồ điều khiển led đơn chớp tắt.</i>	78
<i>Hình 5-46. Lưu đồ điều khiển led đơn chớp tắt 10 lần.</i>	79
<i>Hình 5-47. Lưu đồ điều khiển led đơn sáng dần tắt dần từ phải sang trái.</i>	80
<i>Hình 5-48. Sơ đồ kết nối portB với 1 led 7 đoạn.</i>	81
<i>Hình 5-49. Hình led 7 đoạn.</i>	81
<i>Hình 5-50. Lưu đồ đếm từ 0 đến 9.</i>	82
<i>Hình 5-51. Sơ đồ kết nối portB, C điều khiển 2 led 7 đoạn.</i>	84
<i>Hình 5-52. Lưu đồ đếm từ 00 đến 99.</i>	84
<i>Hình 5-53. Sơ đồ kết nối 2 port B và D điều khiển 8 led 7 đoạn quét.</i>	86
<i>Hình 5-54. Lưu đồ điều khiển 8 led quét sáng 8 số.</i>	87
<i>Hình 5-55. Sơ đồ kết nối 2 port B và D điều khiển 2 led 7 đoạn quét.</i>	88
<i>Hình 5-56. Lưu đồ đếm từ 00 đến 99 hiển thị trên 2 led quét.</i>	88
<i>Hình 5-57. Sơ đồ điều khiển led và nút nhấn.</i>	91
<i>Hình 5-58. Lưu đồ điều khiển led đơn bằng nút nhấn ON-OFF.</i>	91
<i>Hình 5-59. Sơ đồ điều khiển led và 3 nút nhấn.</i>	93
<i>Hình 5-60. Lưu đồ điều khiển led bằng 3 nút ON-OFF-INV.</i>	93
<i>Hình 5-61. Lưu đồ điều khiển led có chống dội phím INV.</i>	94
<i>Hình 5-62. Sơ đồ kết nối 2 port điều khiển 2 led 7 đoạn, 2 nút nhấn.</i>	95
<i>Hình 5-63. Lưu đồ đếm có điều khiển bằng nút nhấn Start-Stop.</i>	96
<i>Hình 5-64. Bàn phím ma trận 4×4.</i>	97
<i>Hình 5-65. Bàn phím ma trận 4×4 với cột C1 bằng 0.</i>	98
<i>Hình 5-66. Bàn phím ma trận 4×4 với cột C2 bằng 0.</i>	98
<i>Hình 5-67. Bàn phím ma trận 4×4 với cột C3 bằng 0.</i>	99
<i>Hình 5-68. Lưu đồ quét bàn phím ma trận 4×4.</i>	100
<i>Hình 5-69. Lưu đồ quét bàn phím ma trận 4×4 có chống dội.</i>	101
<i>Hình 5-70. Vi điều khiển giao tiếp bàn phím ma trận.</i>	101
<i>Hình 5-71. Lưu đồ quét hiển thị ma trận phím.</i>	102
<i>Hình 5-72. Vi điều khiển giao tiếp bàn phím ma trận và 2 led 7 đoạn.</i>	104
<i>Hình 5-73. Lưu đồ quét ma trận phím và hiển thị mã phím.</i>	104
<i>Hình 5-74. Hình ảnh của LCD.</i>	105
<i>Hình 5-75. Giao tiếp vi điều khiển PIC16F887 với LCD.</i>	107
<i>Hình 5-76. Dạng sóng điều khiển của LCD.</i>	108
<i>Hình 5-77. Lưu đồ hiển thị thông tin trên 2 hàng.</i>	109
<i>Hình 5-78. Lưu đồ hiển thị thông tin và đếm giây.</i>	112

<i>Hình 6-1. Sơ đồ khối của timer0 của PIC16F887.</i>	118
<i>Hình 6-2. Thanh ghi OPTION_REG.</i>	119
<i>Hình 6-3. Thanh ghi INTCON.</i>	120
<i>Hình 6-4. Bộ chia trùớc được gán cho timer T0.</i>	120
<i>Hình 6-5. Bộ chia trùớc được gán cho WDT.</i>	121
<i>Hình 6-6. Thanh ghi lưu kết quả của T1.</i>	121
<i>Hình 6-7. Cấu trúc timer T1.</i>	122
<i>Hình 6-8. Thanh ghi T1CON.</i>	122
<i>Hình 6-9. Timer1 hoạt động định thời.</i>	123
<i>Hình 6-10. T1 hoạt động đếm xung ngoại từ mạch dao động T1.</i>	124
<i>Hình 6-11. T1 hoạt động đếm xung ngoại đưa đến ngõ vào T1CKI.</i>	124
<i>Hình 6-12. Giản đồ thời gian xung đếm của Counter1.</i>	124
<i>Hình 6-13. Kết nối thạch anh tạo dao động.</i>	125
<i>Hình 6-14. Sơ đồ khối của Timer2.</i>	126
<i>Hình 6-15. Thanh ghi T2CON.</i>	127
<i>Hình 6-16. PIC điều khiển 8 led sáng tắt.</i>	130
<i>Hình 6-17. Lưu đồ điều khiển 8 led sáng tắt – định thời 210ms.</i>	130
<i>Hình 6-18. Lưu đồ điều khiển 8 led sáng tắt dùng ngắn – định thời 200ms.</i>	131
<i>Hình 6-19. Lưu đồ điều khiển 8 led sáng tắt – định thời 1s.</i>	132
<i>Hình 6-20. Lưu đồ điều khiển 8 led sáng tắt, định thời 13,107ms dùng T0.</i>	134
<i>Hình 6-21. Lưu đồ điều khiển 8 led sáng tắt, định thời 1s dùng T0.</i>	135
<i>Hình 6-22. Lưu đồ điều khiển 8 led sáng tắt, định thời 13,107ms dùng T2.</i>	136
<i>Hình 6-23. Đếm xung ngoại dùng counter T0.</i>	137
<i>Hình 6-24. Lưu đồ đếm xung ngoại dùng counter T0.</i>	138
<i>Hình 6-25. Đếm xung ngoại dùng counter T0.</i>	139
<i>Hình 6-26. Lưu đồ đếm xung ngoại dùng counter T1.</i>	149
<i>Hình 6-27. Đếm xung ngoại dùng counter T1 hiển thị trên 3 led quét.</i>	141
<i>Hình 6-28. Lưu đồ chương trình đếm dùng counter T1 của PIC 16F887.</i>	141
<i>Hình 7-1. Sơ đồ khối của ADC PIC 16F887.</i>	146
<i>Hình 7-2. Thanh ghi ADCON0.</i>	147
<i>Hình 7-3. Thanh ghi ADCON1.</i>	148
<i>Hình 7-4. Định dạng cặp thanh ghi lưu kết quả.</i>	149
<i>Hình 7-5. Thời gian chuyển đổi 10 bit.</i>	150
<i>Hình 7-6. Sơ đồ mạch đo nhiệt độ dùng PIC16F887 hiển thị trên 3 led trực tiếp.</i>	152
<i>Hình 7-7. Lưu đồ chuyển đổi ADC đo nhiệt độ kênh thứ 0.</i>	152
<i>Hình 7-8. Sơ đồ mạch giao tiếp điều khiển relay, triac, buzzer.</i>	154
<i>Hình 7-9. Lưu đồ chuyển đổi ADC đo nhiệt độ kênh thứ 0.</i>	154
<i>Hình 7-10. Sơ đồ mạch đo nhiệt độ dùng PIC16F887, hiển thị 3 led quét.</i>	155
<i>Hình 7-11. Lưu đồ chuyển đổi ADC đo nhiệt độ hiển thị led quét.</i>	156

<i>Hình 8-1. CPU thực hiện chương trình chính trong 2 trường hợp không và có ngắt.</i>	160
<i>Hình 8-2. Mạch điện ngắt của PIC16F887.</i>	162
<i>Hình 8-3. Bộ nhớ ngăn xếp khi thực hiện ngắt và kết thúc ngắt.</i>	162
<i>Hình 8-4. Thực hiện chương trình chính, bị ngắt và kết thúc ngắt.</i>	162
<i>Hình 8-5. Thanh ghi INTCON.</i>	163
<i>Hình 8-6. Thanh ghi PIE1 và PIR1.</i>	164
<i>Hình 8-7. Thanh ghi PIE2 và PIR2.</i>	165
<i>Hình 8-8. Điều khiển 8 led sáng tắt.</i>	167
<i>Hình 8-9. Lưu đồ điều khiển 8 led sáng tắt– định thời 210ms dùng ngắt.</i>	167
<i>Hình 8-10. Mạch giao tiếp 2 led 7 đoạn quét hiển thị đêm giây.</i>	168
<i>Hình 8-11. Lưu đồ đêm giây dùng Timer định thời báo ngắt.</i>	169
<i>Hình 8-12. Mạch đo nhiệt độ dùng cảm biến LM35.</i>	170
<i>Hình 8-13. Lưu đồ chuyển đổi ADC có báo ngắt.</i>	170
<i>Hình 8-14. Đo nhiệt độ dùng cảm biến LM35 và đêm giây.</i>	172
<i>Hình 8-15. Lưu đồ chuyển đổi ADC có báo ngắt và đêm giây dùng timer báo ngắt.</i>	173
<i>Hình 9-1. Dạng sóng điều chế độ rộng xung.</i>	178
<i>Hình 9-2. Dạng sóng điều chế độ rộng xung và điện áp trung bình 3 cấp độ.</i>	179
<i>Hình 9-3. Sơ đồ khối PWM của khối CCP1 của PIC16F887.</i>	180
<i>Hình 9-4. Số bit của các thanh ghi dùng trong khối PWM của PWM PIC16F887.</i>	180
<i>Hình 9-5. Dạng sóng PWM.</i>	181
<i>Hình 9-6. Sơ đồ khối PWM nâng cao.</i>	182
<i>Hình 9-7. Chế độ hoạt động nǔra cầu.</i>	183
<i>Hình 9-8. Chế độ hoạt động nǔra cầu, điều khiển mạch cầu H đầy đủ.</i>	183
<i>Hình 9-9. Chế độ hoạt động cầu đầy đủ điều khiển mạch cầu H.</i>	184
<i>Hình 9-10. Chế độ hoạt động cầu đầy đủ điều khiển motor mạch cầu H, quay thuận.</i>	184
<i>Hình 9-11. Dạng sóng tín hiệu điều khiển motor quay thuận.</i>	185
<i>Hình 9-12. Chế độ hoạt động cầu đầy đủ điều khiển motor mạch cầu H, quay nghịch.</i>	185
<i>Hình 9-13. Dạng sóng tín hiệu điều khiển motor quay nghịch.</i>	186
<i>Hình 9-14. Mạch điều khiển thay đổi cường độ sáng của đèn dùng PWM.</i>	188
<i>Hình 9-15. Các dạng tín hiệu thay đổi cường độ sáng của đèn dùng PWM.</i>	188
<i>Hình 9-16. Lưu đồ điều khiển đèn sáng dùng PWM.</i>	189
<i>Hình 9-17. Mạch thay đổi cường độ sáng của đèn dùng PWM và 2 nút nhấn.</i>	190
<i>Hình 9-18. Lưu đồ điều khiển đèn sáng dùng PWM và 2 nút nhấn.</i>	190
<i>Hình 9-19. Mạch điều khiển thay đổi tốc độ động cơ dùng PWM.</i>	192
<i>Hình 9-20. Lưu đồ điều khiển thay đổi tốc độ động cơ dùng PWM.</i>	193
<i>Hình 10-1. Hệ thống truyền đồng bộ.</i>	198
<i>Hình 10-2. Hệ thống truyền bắt đồng bộ.</i>	198
<i>Hình 10-3. Sơ đồ khối của khối phát dữ liệu của PIC16F887.</i>	200

<i>Hình 10-4. Dạng sóng truyền dữ liệu.</i>	200
<i>Hình 10-5. Dạng sóng truyền 2 byte dữ liệu.</i>	201
<i>Hình 10-6. Thanh ghi TXSTA.</i>	201
<i>Hình 10-7. Thanh ghi RCSTA.</i>	201
<i>Hình 10-8. Sơ đồ khói của khói nhận dữ liệu của PIC16F887.</i>	203
<i>Hình 10-9. Dạng sóng nhận dữ liệu.</i>	205
<i>Hình 10-10. Thanh ghi BAUDCTL.</i>	205
<i>Hình 10-11. Hệ thống truyền dữ liệu giữa máy tính và vi điều khiển.</i>	208
<i>Hình 10-12. Dạng sóng các mức 1 và 0 của chuẩn RS232.</i>	208
<i>Hình 10-13. Lưu đồ điều khiển truyền dữ liệu.</i>	209
<i>Hình 10-14. Giao diện phần mềm Terminal để gửi dữ liệu.</i>	210
<i>Hình 10-15. Hệ thống truyền dữ liệu giữa máy tính và vi điều khiển, hiển thị LCD.</i>	211
<i>Hình 10-16. Lưu đồ điều khiển truyền dữ liệu, hiển thị trên LCD.</i>	211
<i>Hình 10-17. Hệ thống truyền dữ liệu giữa máy tính và vi điều khiển, có LCD, bàn phím.</i>	212
<i>Hình 10-18. Lưu đồ điều khiển truyền dữ liệu, có thêm LCD và bàn phím.</i>	213
<i>Hình 10-19. Hệ thống truyền dữ liệu giữa máy tính và vi điều khiển, có LCD, bàn phím.</i>	214
<i>Hình 10-20. Lưu đồ điều khiển truyền dữ liệu, có thêm LCD và bàn phím.</i>	215
 <i>Bảng 1-1. Trình bày tóm tắt cấu trúc của 5 loại PIC16F88X.</i>	6
<i>Bảng 3-1. Các tác tố.</i>	42
<i>Bảng 3-2. Tóm tắt tập lệnh của PIC.</i>	42
<i>Bảng 4-1. Các kiểu dữ liệu của phần mềm PIC-C.</i>	54
<i>Bảng 4-2. Các toán tử phổ biến trong ngôn ngữ C.</i>	55
<i>Bảng 5-1. Tên các tín hiệu và chức năng của mạch nạp ICSP.</i>	77
<i>Bảng 5-2. Mã 7 đoạn cho các số thập phân.</i>	81
<i>Bảng 5-3. Các chân của LCD.</i>	106
<i>Bảng 5-4. Các lệnh điều khiển LCD.</i>	107
<i>Bảng 5-5. Địa chỉ của từng kí tự.</i>	109
<i>Bảng 5-6. Thông tin hiển thị 2 hàng kí tự.</i>	112
<i>Bảng 6-1. Lựa chọn hệ số chia của Timer0.</i>	119
<i>Bảng 6-2. Lựa chọn tần số và tụ tương ứng của Timer1.</i>	125
<i>Bảng 7-1. Chọn kênh tương tự của 4 bit CHS.</i>	148
<i>Bảng 7-2. Tần số xung clock tùy chọn phụ thuộc vào tần số bộ dao động.</i>	150
<i>Bảng 8-1. Tóm tắt chức năng các bit trong thanh ghi INTCON có địa chỉ 0x0B.</i>	163
<i>Bảng 8-2. Tóm tắt chức năng các bit trong thanh ghi PIE1.</i>	164
<i>Bảng 8-3. Tóm tắt chức năng các bit trong thanh ghi cho phép ngắt PIR1.</i>	164
<i>Bảng 8-4. Tóm tắt chức năng các bit trong thanh ghi cho phép ngắt PIE2.</i>	165
<i>Bảng 8-5. Tóm tắt chức năng các bit trong thanh ghi cho phép ngắt PIR2.</i>	165
<i>Bảng 9-1. Các chế độ hoạt động của PWM nâng cao.</i>	183

Bảng 9-2. Các trạng thái điều khiển động cơ DC.	192
Bảng 10-1. Tóm tắt chức năng các bit trong thanh ghi TXSTA.	202
Bảng 10-2. Tóm tắt chức năng các bit trong thanh ghi RCSTA.	201
Bảng 10-3. Tóm tắt chức năng các bit trong thanh ghi BAUDCTL.	205
Bảng 10-4. Tóm tắt các công thức tính tốc độ baud.	206

MỤC LỤC

Hình và bảng Chương 1

Trang

VI ĐIỀU KHIỂN PIC 16F887: ĐẶT TÍNH, CẤU TRÚC, CHỨC NĂNG CÁC PORT

1.1	GIỚI THIỆU	2
1.2	KHẢO SÁT VI ĐIỀU KHIỂN MICROCHIP	4
1.2.1	Cấu hình của vi điều khiển pic16f887	4
1.2.2	Sơ đồ cấu trúc của vi điều khiển pic 16f887	7
1.2.3	Khảo sát sơ đồ chân vi điều khiển pic16f887	9
1.3	CÂU HỎI ÔN TẬP – TRẮC NGHIỆM – BÀI TẬP	15
1.3.1	Câu hỏi ôn tập	15
1.3.2	Câu hỏi mở rộng	15
1.3.3	Câu hỏi trắc nghiệm	16
1.3.4	Bài tập	17

Chương 2

VI ĐIỀU KHIỂN PIC 16F887: TỔ CHỨC BỘ NHỚ, THANH GHI

2.1	GIỚI THIỆU	20
2.2	KIẾN TRÚC BỘ NHỚ	20
2.3	TỔ CHỨC BỘ NHỚ CỦA VI ĐIỀU KHIỂN PIC 16F887	20
2.3.1	Tổ chức bộ nhớ chương trình và ngăn xếp	20
2.3.2	Mã lệnh 14 bit	24
2.3.3	Cấu trúc Pipeline	25
2.3.4	Khảo sát bộ nhớ dữ liệu và thanh ghi trạng thái	29
2.3.5	Bộ nhớ dữ liệu eeprom	32
2.3.6	Tóm tắt	32
2.4	CÂU HỎI ÔN TẬP – TRẮC NGHIỆM - BÀI TẬP	33
2.4.1	Câu hỏi ôn tập	33
2.4.2	Câu hỏi mở rộng	33
2.4.3	Câu hỏi trắc nghiệm	33
2.4.4	Bài tập	34

Chương 3

VI ĐIỀU KHIỂN PIC 16F887: LỆNH HỢP NGỮ

3.1	GIỚI THIỆU	36
3.2	NGÔN NGỮ LẬP TRÌNH HỢP NGỮ	36
3.2.1	Phần mềm lập trình hợp ngữ	37
3.2.2	Chương trình hợp ngữ cơ bản	38
3.2.3	Trình biên dịch cho hợp ngữ	39
3.3	LỆNH HỢP NGỮ CỦA VI ĐIỀU KHIỂN PIC 16F887	41
3.3.1	Giới thiệu	41
3.3.2	Khảo sát tập lệnh tóm tắt vi điều khiển pic 16f887	42
3.3.3	Tập lệnh chi tiết	44
3.4	CÂU HỎI ÔN TẬP – TRẮC NGHIỆM – BÀI TẬP	49
3.4.1	CÂU HỎI ÔN TẬP	49
3.4.2	CÂU HỎI MỞ RỘNG	49
3.4.3	CÂU HỎI TRẮC NGHIỆM	49
3.4.4	BÀI TẬP	51

Chương 4

VI ĐIỀU KHIỂN PIC 16F887: NGÔN NGỮ LỆNH TRÌNH C

4.1	GIỚI THIỆU	54
4.2	CÁC THÀNH PHẦN CƠ BẢN CỦA NGÔN NGỮ C	54
4.2.1	Các kiểu dữ liệu của biến	54
4.2.2	Các toán tử	53
4.2.3	Các lệnh c cơ bản	59
4.2.4	Cấu trúc của chương trình c	62
4.2.5	Các thành phần của chương trình c	63
4.2.6	Con trỏ dữ liệu	63
4.2.7	Khai báo mảng	64
4.3	TRÌNH BIÊN DỊCH C, THƯ VIỆN	64
4.3.1	Trình biên dịch C	64
4.3.2	Thư viện PIC16F887 của trình biên dịch C	65
4.4	CÂU HỎI ÔN TẬP – TRẮC NGHIỆM – BÀI TẬP	49
4.4.1	CÂU HỎI ÔN TẬP	49
4.4.2	CÂU HỎI MỞ RỘNG	49
4.4.3	CÂU HỎI TRẮC NGHIỆM	49
4.4.4	BÀI TẬP	51

Chương 5

VI ĐIỀU KHIỂN PIC 16F887:

GIAO TIẾP LED, LCD, PHÍM ĐƠN, MA TRẬN PHÍM

5.1	GIỚI THIỆU	54
5.2	CHỨC NĂNG CÁC PORT CỦA VI ĐIỀU KHIỂN	54
5.3	CÁC PORT CỦA PIC 16F887	54
5.3.1	Porta và thanh ghi trisa	55
5.3.2	Portb và thanh ghi trisb	57
5.3.3	Portc và thanh ghi trisc	62
5.3.4	Portd và thanh ghi trisd	64
5.3.5	Porte và thanh ghi trise	66
5.4	LỆNH TRUY XUẤT PORT DÙNG NGÔN NGỮ CCS-C	67
5.4.1	Lệnh set_tris_x()	67
5.4.2	Lệnh output_x(value)	67
5.4.3	Lệnh output_high(pin)	68
5.4.4	Lệnh output_low(pin)	68
5.4.5	Lệnh output_toggle(pin)	68
5.4.6	Lệnh output_bit(pin,value)	68
5.4.7	Lệnh value = get_tris_x()	68
5.4.8	Lệnh value = input(pin)	68
5.4.9	Lệnh input_state()	69
5.4.10	Lệnh output_drive(pin)	69
5.4.11	Lệnh output_float(pin)	69
5.4.12	Lệnh port_b_pullup()	69
5.5	CẤU HÌNH ĐẶC BIỆT CỦA CPU	70
5.5.1	Cấu hình reset cpu	70
5.5.2	Cấu hình các ngắt đánh thức cpu	71
5.5.3	Cấu hình các dạng dao động của cpu	72
5.5.4	Cấu hình bảo vệ code	74
5.5.5	Bộ định thời giám sát (Watch Dog Timer)	74
5.5.6	Mạch nạp nối tiếp bên trong	76
5.6	CÁC ỨNG DỤNG ĐIỀU KHIỂN LED ĐƠN	77
5.7	CÁC ỨNG DỤNG ĐIỀU KHIỂN LED 7 ĐOẠN TRỰC TIẾP	81
5.8	CÁC ỨNG DỤNG ĐIỀU KHIỂN LED 7 ĐOẠN QUÉT	85
5.9	CÁC ỨNG DỤNG GIAO TIẾP VỚI NÚT NHẤN, BÀN PHÍM	90
5.9.1	Hệ thống ít phím	90
5.9.2	Hệ thống nhiều phím	97
5.10	CÁC ỨNG DỤNG ĐIỀU KHIỂN LCD	105
5.10.1	Giới thiệu LCD	105
5.10.2	Sơ đồ chân của LCD	106

5.10.3	Sơ đồ mạch giao tiếp vi điều khiển với LCD	106
5.10.4	Các lệnh điều khiển LCD	107
5.10.5	Địa chỉ của từng kí tự trên LCD	109
5.10.6	Các chương trình hiển thị trên LCD	109
5.11	CÂU HỎI ÔN TẬP – TRẮC NGHIỆM - BÀI TẬP	113
5.11.1	Câu hỏi ôn tập	113
5.11.2	Câu hỏi mở rộng	113
5.11.3	Câu hỏi trắc nghiệm	113
5.11.4	Bài tập	115

Chương 6

VI ĐIỀU KHIỂN PIC 16F887: TIMER - COUNTER

6.1	GIỚI THIỆU	118
6.2	KHẢO SÁT TIMER0	118
6.2.1	Ngắt của timer0	120
6.2.2	Timer0 đếm xung ngoại	120
6.2.3	Bộ chia trước	120
6.3	KHẢO SÁT TIMER1 CỦA PIC 16F887	121
6.3.1	Timer1 ở chế độ định thời	123
6.3.2	Timer1 ở chế độ đếm xung ngoại	123
6.3.3	Hoạt động của timer1 ở chế độ counter đồng bộ	124
6.3.4	Hoạt động của timer1 ở chế độ counter bắt đồng bộ	125
6.3.5	Đọc và ghi timer1 trong chế độ đếm không đồng bộ	125
6.3.6	Bộ dao động của timer1	125
6.3.7	Reset timer1 sử dụng ngõ ra ccp trigger	125
6.3.8	Reset cắp thanh ghi tmr1h, tmr1l của timer1	126
6.4	KHẢO SÁT TIMER2 CỦA PIC16F887	126
6.4.1	Bộ chia trước và chia sau của timer2	127
6.4.2	Ngõ ra của timer2	127
6.5	CÁC LỆNH CỦA TIMER – COUNTER TRONG NGÔN NGỮ PIC-C	127
6.5.1	Lệnh setup_timer_0(mode)	127
6.5.2	Lệnh setup_timer_1(mode)	128
6.5.3	Lệnh setup_timer_2(mode)	128
6.5.4	Lệnh set_timerx(value)	128
6.5.5	Lệnh get_timerx()	128
6.5.6	Lệnh setup_wdt()	129
6.5.7	Lệnh restart_wdt()	129
6.6	CÁC ỨNG DỤNG ĐỊNH THỜI DÙNG TIMER	129
6.6.1	Định thời dùng timer T1	129
6.6.2	Định thời dùng timer T0	133
6.6.3	Định thời dùng timer T2	136

6.7	CÁC ÚNG DỤNG ĐÉM XUNG NGOẠI DÙNG COUNTER	137
6.7.1	Đếm xung ngoại dùng counter T0	137
6.7.2	Đếm xung ngoại dùng counter T1	137
6.8	CÂU HỎI ÔN TẬP – TRẮC NGHIỆM - BÀI TẬP	143
6.8.1	Câu hỏi ôn tập	143
6.8.2	Câu hỏi mở rộng	143
6.8.3	Câu hỏi trắc nghiệm	143
6.8.4	Bài tập	144

Chương 7

VI ĐIỀU KHIỂN PIC 16F887: CHUYỂN ĐỔI TƯƠNG TỰ SANG SỐ

7.1	GIỚI THIỆU	146
7.2	ADC CỦA VI ĐIỀU KHIỂN PIC 16F887	146
7.2.1	Khảo sát adc của pic 16f887	146
7.2.2	Khảo sát các thanh ghi của pic 16f887	147
7.2.3	Trình tự thực hiện chuyển đổi adc	149
7.2.4	Lựa chọn nguồn xung cho chuyển đổi adc	150
7.3	CÁC LỆNH CỦA ADC TRONG NGÔN NGỮ CCS-C	150
7.3.1	Lệnh setup_adc(mode)	151
7.3.2	Lệnh setup_adc_port(value)	151
7.3.3	Lệnh set_adc_channel(chan)	151
7.3.4	Lệnh value=read_adc(mode)	151
7.4	ÚNG DỤNG ADC CỦA PIC 16F887	151
7.4.1	Đo nhiệt độ dùng cảm biến LM35	151
7.4.2	Đo nhiệt độ có điều khiển tái bóng đèn bằng relay	153
7.4.3	Đo nhiệt độ hiển thị trên led 7 đoạn quét	155
7.5	CÂU HỎI ÔN TẬP – TRẮC NGHIỆM - BÀI TẬP	157
7.5.1	Câu hỏi ôn tập	157
7.5.2	Câu hỏi mở rộng	157
7.5.3	Câu hỏi trắc nghiệm	157
7.5.4	Bài tập	158

Chương 8

VI ĐIỀU KHIỂN PIC 16F887: NGẮT

8.1	GIỚI THIỆU	160
8.2	TỔNG QUAN VỀ NGẮT	160
8.3	NGẮT CỦA VI ĐIỀU KHIỂN PIC16F887	160

8.3.1	Các nguồn ngắt của PIC 16F887	160
8.3.2	Cấu trúc mạch điện ngắt của PIC 16F887	162
8.3.3	Các thanh ghi ngắt của PIC16F887	163
8.4	CÁC LỆNH NGẮT CỦA PIC16F887 TRONG NGÔN NGỮ PIC-C	165
8.4.1	Lệnh Enable_interrupts(level)	165
8.4.2	Lệnh Disable_interrupts(level)	166
8.4.3	Viết chương trình con phục vụ ngắt	166
8.5	CÁC ỨNG DỤNG NGẮT CỦA PIC 16F887	166
8.5.1	Ứng dụng ngắt của timer T1	166
8.5.2	Ứng dụng ngắt của ADC	170
8.5.3	Ứng dụng 2 ngắt của timer và ADC	172
8.6	CÂU HỎI ÔN TẬP – TRẮC NGHIỆM - BÀI TẬP	175
8.6.1	Câu hỏi ôn tập	175
8.6.2	Câu hỏi mở rộng	175
8.6.3	Câu hỏi trắc nghiệm	175
8.6.4	Bài tập	176

Chương 9

VI ĐIỀU KHIỂN PIC 16F887: ĐIỀU CHẾ ĐỘ RỘNG XUNG PWM

9.1	GIỚI THIỆU	178
9.2	KHẢO SÁT PWM	178
9.2.1	Nguyên lý điều chế độ rộng xung PWM	178
9.2.2	Cấu trúc khối điều chế độ rộng xung PWM	179
9.2.3	Tính chu kỳ xung PWM	181
9.2.4	Tính hệ số chu kỳ xung PWM	182
9.3	KHẢO SÁT PWM NÂNG CAO	182
9.3.1	Cấu trúc khối PWM nâng cao	182
9.3.2	Các chế độ hoạt động khối PWM nâng cao	183
9.4	CÁC LỆNH ĐIỀU KHIỂN	186
9.4.1	Lệnh định cấu hình khối CCP	186
9.4.2	Lệnh thiết lập hệ số chu kỳ	186
9.4.3	Lệnh setup_timer_2 - lệnh định cấu hình cho timer_2	186
9.4.4	Lệnh set_timerx(value) - lệnh thiết lập giá trị cho timer	187
9.5	CÁC CHƯƠNG TRÌNH ỨNG DỤNG PWM – BÀI TẬP	187
9.5.1	Điều khiển độ sáng của đèn cấp 1/10 dùng PWM	187
9.5.2	Điều khiển độ sáng của đèn 10 cấp dùng PWM	189
9.5.3	Điều khiển thay đổi tốc độ động cơ dc dùng PWM	191
9.6	CÂU HỎI ÔN TẬP – TRẮC NGHIỆM - BÀI TẬP	175

9.6.1	Câu hỏi ôn tập	196
9.6.2	Câu hỏi mở rộng	196
9.6.3	Câu hỏi trắc nghiệm	196
9.6.4	Bài tập	196

Chương 10

VI ĐIỀU KHIỂN PIC 16F887: TRUYỀN DỮ LIỆU

10.1	GIỚI THIỆU	198
10.2	TỔNG QUAN VỀ CÁC KIỀU TRUYỀN DỮ LIỆU	198
10.3	TRUYỀN DỮ LIỆU NÓI TIẾP ĐỒNG BỘ VÀ KHÔNG ĐỒNG BỘ	198
10.4	TRUYỀN DỮ LIỆU CỦA VI ĐIỀU KHIỂN PIC 16F887	199
10.4.1	Giới thiệu truyền dữ liệu eusart	199
10.4.2	Khối phát dữ liệu esuart của PIC16F887	199
10.4.3	Các thanh ghi txsta và rcsta của PIC16F887	201
10.4.4	Khối nhận dữ liệu esuart của PIC16F887	203
10.4.5	Thanh ghi tạo tốc độ baud của PIC16F887	205
10.5	CÁC LỆNH TRUYỀN DỮ LIỆU EUSART CỦA PIC16F887	207
10.5.1	Lệnh setup_uart(baud, stream)	207
10.5.2	Lệnh puts(string)	207
10.5.3	Lệnh value = getc(), value = fgetc(stream), value = getch(), value = getchar()	
10.5.4	Lệnh value = kbhit()	207
10.6	ỨNG DỤNG TRUYỀN DỮ LIỆU EUSART CỦA PIC16F887	208
10.6.1	Truyền dữ liệu giữa PIC16F887 và máy tính điều khiển led	208
10.6.2	Phần mềm truyền dữ liệu terminal trên máy tính	210
10.6.3	Truyền dữ liệu giữa PIC16F887 và máy tính hiển thị lcd	210
10.6.4	Truyền và nhận dữ liệu giữa PIC16F887 và máy tính	212
10.6.5	Truyền và nhận dữ liệu giữa PIC16F887 và máy tính dùng ngắn để nhận dữ liệu	214
10.7	CÂU HỎI ÔN TẬP – TRẮC NGHIỆM - BÀI TẬP	216
10.7.1	Câu hỏi ôn tập	216
10.7.2	Câu hỏi mở rộng	216
10.7.3	Câu hỏi trắc nghiệm	216
10.7.4	Bài tập	217
	TÀI LIỆU THAM KHẢO	219

VỊ ĐIỀU KHIỂN PIC16F887: ĐẶC TÍNH, CẤU TRÚC, CHỨC NĂNG CÁC PORT

1.1 GIỚI THIỆU

Vi xử lý có rất nhiều loại bắt đầu từ 4 bit cho đến 32 bit, vi xử lý 4 bit hiện nay không còn nhưng vi xử lý 8 bit vẫn còn mặc dù đã có vi xử lý 64 bit.

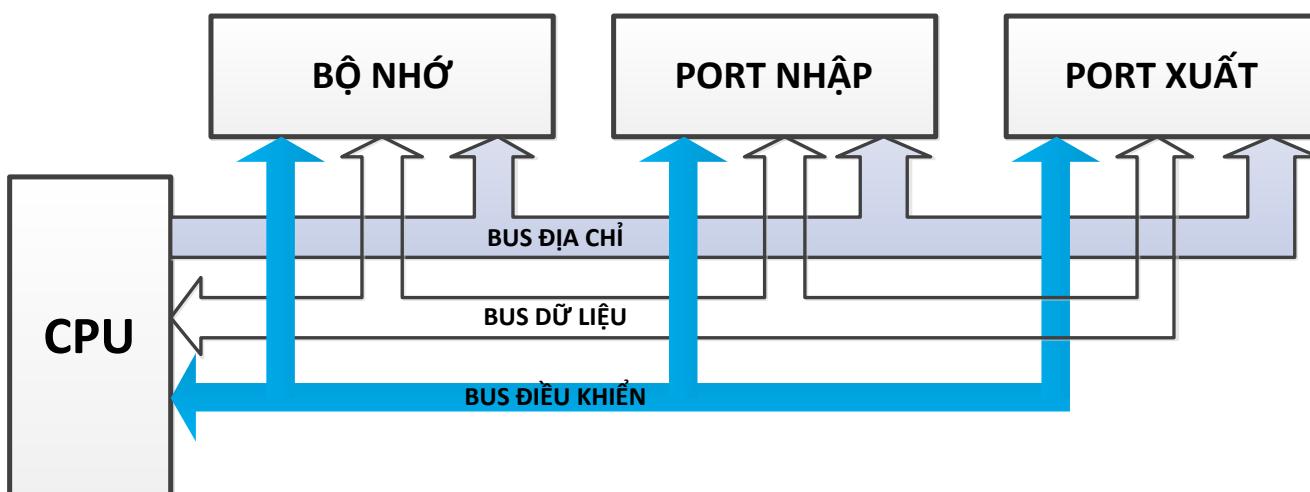
Lý do sự tồn tại của vi xử lý 8 bit là phù hợp với một số yêu cầu điều khiển trong công nghiệp. Các vi xử lý 32 bit, 64 bit thường sử dụng cho các máy tính vì khối lượng dữ liệu của máy tính rất lớn nên cần các vi xử lý càng mạnh càng tốt.

Các hệ thống điều khiển trong công nghiệp sử dụng các vi xử lý 8 bit hay 16 bit như hệ thống điện của xe hơi, hệ thống điều hòa, hệ thống điều khiển các dây chuyền sản xuất, ...



Hình 1-1. Các thiết bị sử dụng vi xử lý.

Khi sử dụng vi xử lý thì phải thiết kế một hệ thống gồm có: Vi xử lý, bộ nhớ, các ngoại vi.



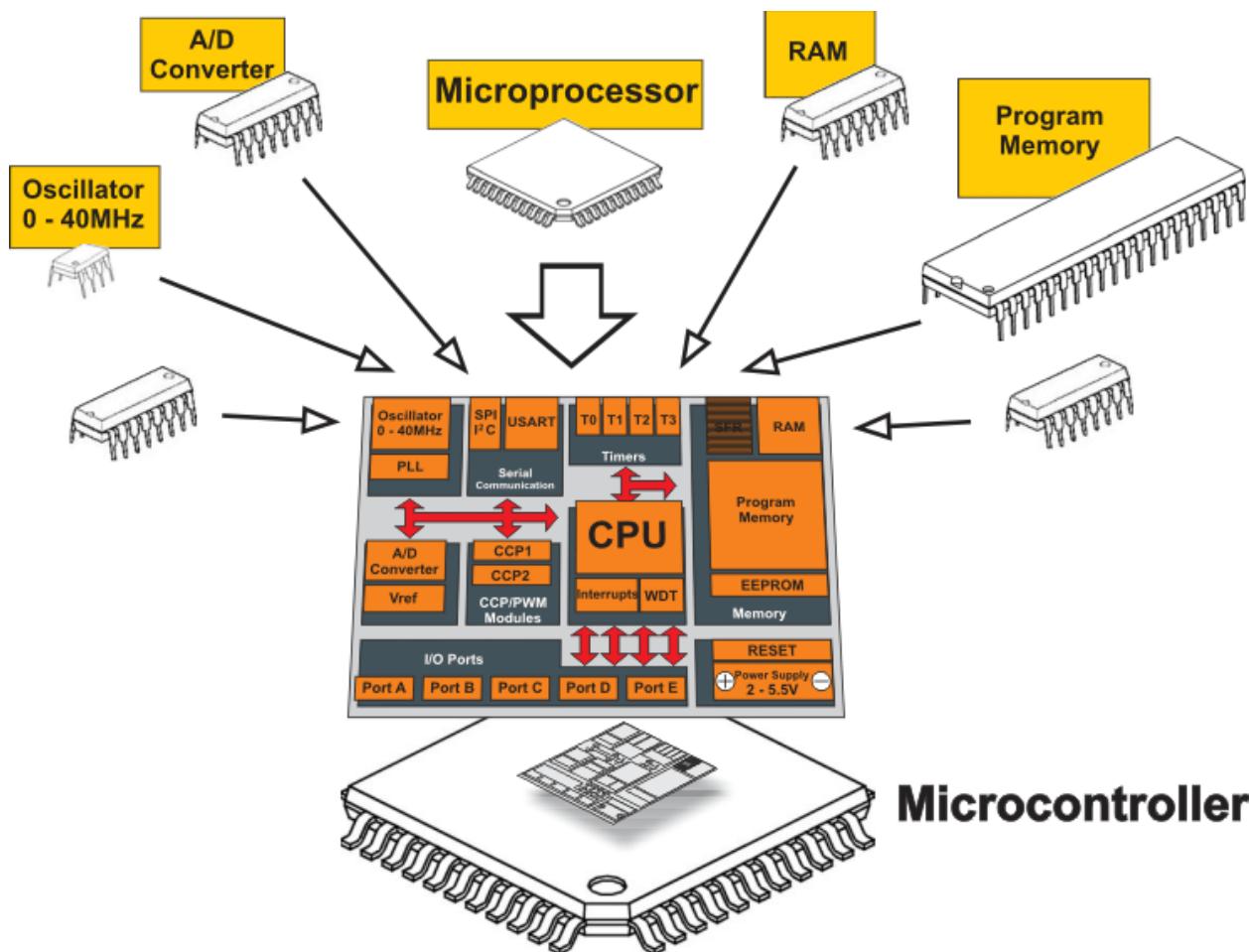
Hình 1-2. Hệ thống vi xử lý.

Bộ nhớ dùng để lưu chương trình cho vi xử lý thực hiện và lưu dữ liệu cần xử lý, các ngoại vi dùng để xuất nhập dữ liệu từ bên ngoài vào xử lý và điều khiển trở lại. Các khối này liên kết với nhau tạo thành một hệ thống vi xử lý.

Yêu cầu điều khiển càng cao thì hệ thống càng phức tạp và nếu yêu cầu điều khiển đơn giản thì hệ thống vi xử lý cũng phải có đầy đủ các khối trên.

Để kết nối các khối trên tạo thành một hệ thống vi xử lý đòi hỏi người thiết kế phải rất hiểu biết về tất cả các thành phần vi xử lý, bộ nhớ, các thiết bị ngoại vi. Hệ thống tạo ra khá phức tạp, chiếm nhiều không gian, mạch in, và vấn đề chính là đòi hỏi người thiết kế hiểu thật rõ về hệ thống. Một lý do nữa là vi xử lý thường xử lý dữ liệu theo byte hoặc word trong khi đó các đối tượng điều khiển trong công nghiệp thường điều khiển theo bit.

Chính vì sự phức tạp nên các nhà chế tạo đã tích hợp bộ nhớ và một số các thiết bị ngoại vi cùng với vi xử lý tạo thành một IC gọi là vi điều khiển – Microcontroller như hình 1-3.

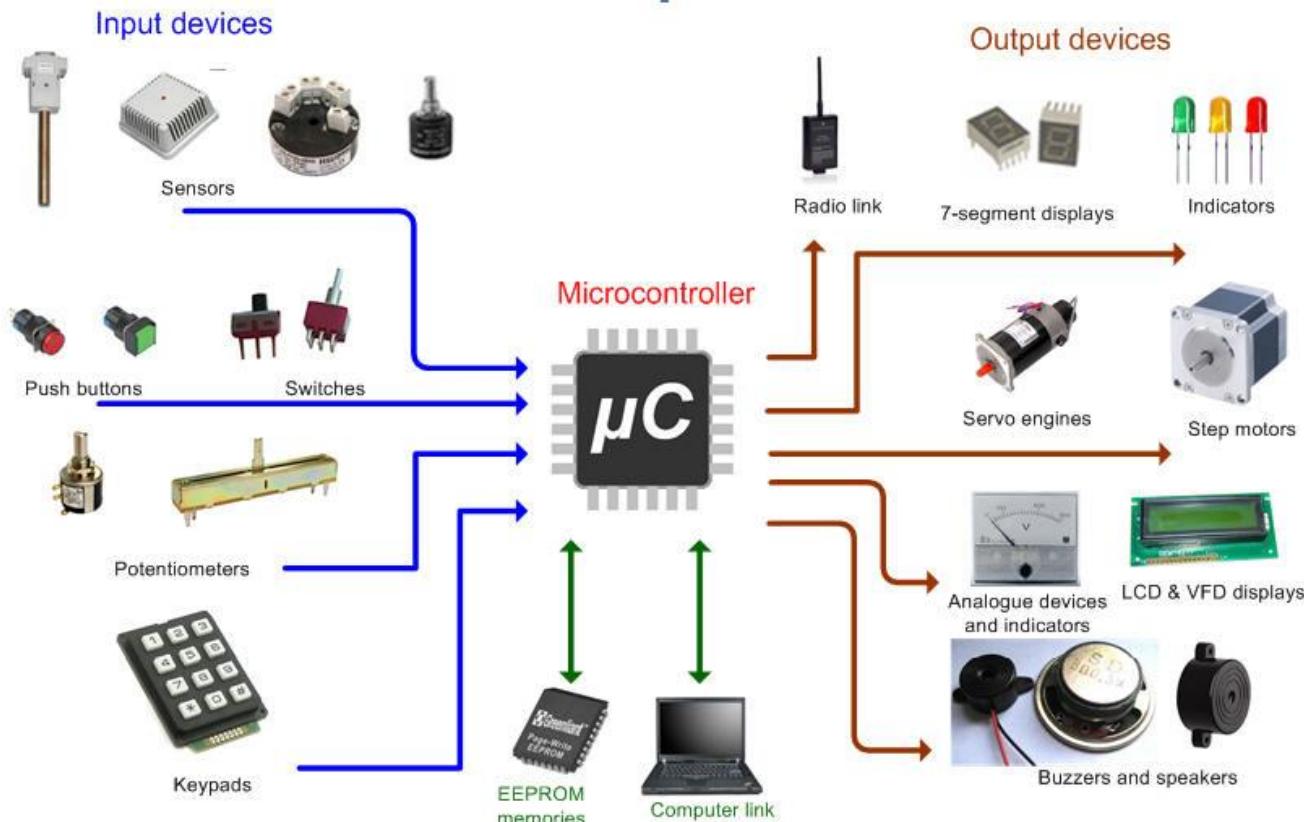


Hình 1-3. Vi điều khiển được tích hợp từ vi xử lý, bộ nhớ và các ngoại vi.

Khi vi điều khiển ra đời đã mang lại sự tiện lợi là dễ dàng sử dụng trong điều khiển công nghiệp, việc sử dụng vi điều khiển không đòi hỏi người sử dụng phải hiểu biết một lượng kiến thức quá nhiều như người sử dụng vi xử lý.

Với một ứng dụng cụ thể thì người thiết kế chỉ cần kết nối vi điều khiển với các thiết bị nhập dữ liệu và các thiết bị xuất dữ liệu để điều khiển. Ví dụ một hệ thống báo chuông giờ học thì cần có vi điều khiển, các thiết bị vào gồm có IC thời gian thực, các nút nhấn để chỉnh thời gian, cài đặt thời gian, các thiết bị ra gồm có màn hình hiển thị như Led hoặc LCD, có transistor, relay hoặc Triac để điều khiển chuông.

Các thiết bị vào và ra cùng với vi điều khiển được thể hiện tiêu biểu như hình 1-4.



Hình 1-4. Các thiết bị vào, ra và vi điều khiển.

Phản tiếp theo chúng ta sẽ khảo sát vi điều khiển để thấy rõ sự tiện lợi trong điều khiển.

Có rất nhiều hãng chế tạo được vi điều khiển, hãng sản xuất nổi tiếng là TI, Microchip, ATMEL, ... tài liệu này sẽ trình bày vi điều khiển tiêu biểu là PIC16F887 của MICROCHIP.

1.2 KHẢO SÁT VI ĐIỀU KHIỂN MICROCHIP

Vi điều khiển hãng Microchip có rất nhiều chủng loại, tích hợp nhiều chức năng, người dùng có thể chọn một vi điều khiển phù hợp với yêu cầu điều khiển. Tài liệu này khảo sát vi điều khiển PIC16F887.

1.2.1 CẤU HÌNH CỦA VI ĐIỀU KHIỂN PIC16F887

Trong tài liệu này trình bày vi điều khiển PIC16F887, các thông số của vi điều khiển như sau:

Đặc điểm thực thi tốc độ cao CPU RISC là:

- Có 35 lệnh đơn.
- Thời gian thực hiện tất cả các lệnh là 1 chu kì máy, ngoại trừ lệnh rẽ nhánh là 2.
- Tốc độ hoạt động:
 - Ngõ vào xung clock có tần số 20MHz.
 - Chu kì lệnh thực hiện lệnh 200ns.
- Có nhiều nguồn ngắt.
- Có 3 kiểu định địa chỉ trực tiếp, gián tiếp và túc thời.

Cấu trúc đặc biệt của vi điều khiển

- Bộ dao động nội chính xác:
 - Sai số $\pm 1\%$
 - Có thể lựa chọn tần số từ 31 kHz đến 8 MHz bằng phần mềm.
 - Cộng hưởng bằng phần mềm.
 - Chế độ bắt đầu 2 cấp tốc độ.
 - Mạch phát hiện hỏng dao động thạch anh cho các ứng dụng quan trọng.
 - Có chuyển mạch nguồn xung clock trong quá trình hoạt động để tiết kiệm công suất.
- Có chế độ ngủ để tiết kiệm công suất.
- Dãy điện áp hoạt động rộng từ 2V đến 5,5V.
- Tầm nhiệt độ làm việc theo chuẩn công nghiệp.
- Có mạch reset khi có điện (Power On Reset – POR).
- Có bộ định thời chờ ổn định điện áp khi mới có điện (Power up Timer – PWRT) và bộ định thời chờ dao động hoạt động ổn định khi mới cấp điện (Oscillator Start-up Timer – OST).
- Có mạch tự động reset khi phát hiện nguồn điện cấp bị sụt giảm, cho phép lựa chọn bằng phần mềm (Brown out Reset – BOR).
- Có bộ định thời giám sát (Watchdog Timer – WDT) dùng dao động trong chip cho phép bằng phần mềm (có thể định thời lên đến 268 giây).
- Đa hợp ngõ vào reset với ngõ vào có điện trở kéo lên.
- Có bảo vệ code đã lập trình.
- Bộ nhớ Flash cho phép xóa và lập trình 100,000 lần.
- Bộ nhớ Eeprom cho phép xóa và lập trình 1,000,000 lần và có thể tồn tại trên 40 năm.
- Cho phép đọc/ghi bộ nhớ chương trình khi mạch hoạt động.
- Có tích hợp mạch gỡ rối.

Cấu trúc nguồn công suất thấp

- Chế độ chờ: dòng tiêu tán khoảng 50nA, sử dụng nguồn 2V.
- Dòng hoạt động:
 - 11 μ A ở tần số hoạt động 32kHz, sử dụng nguồn 2V.
 - 220 μ A ở tần số hoạt động 4MHz, sử dụng nguồn 2V.
- Bộ định thời Watchdog Timer khi hoạt động tiêu thụ 1,4 μ A, điện áp 2V.

Cấu trúc ngoại vi

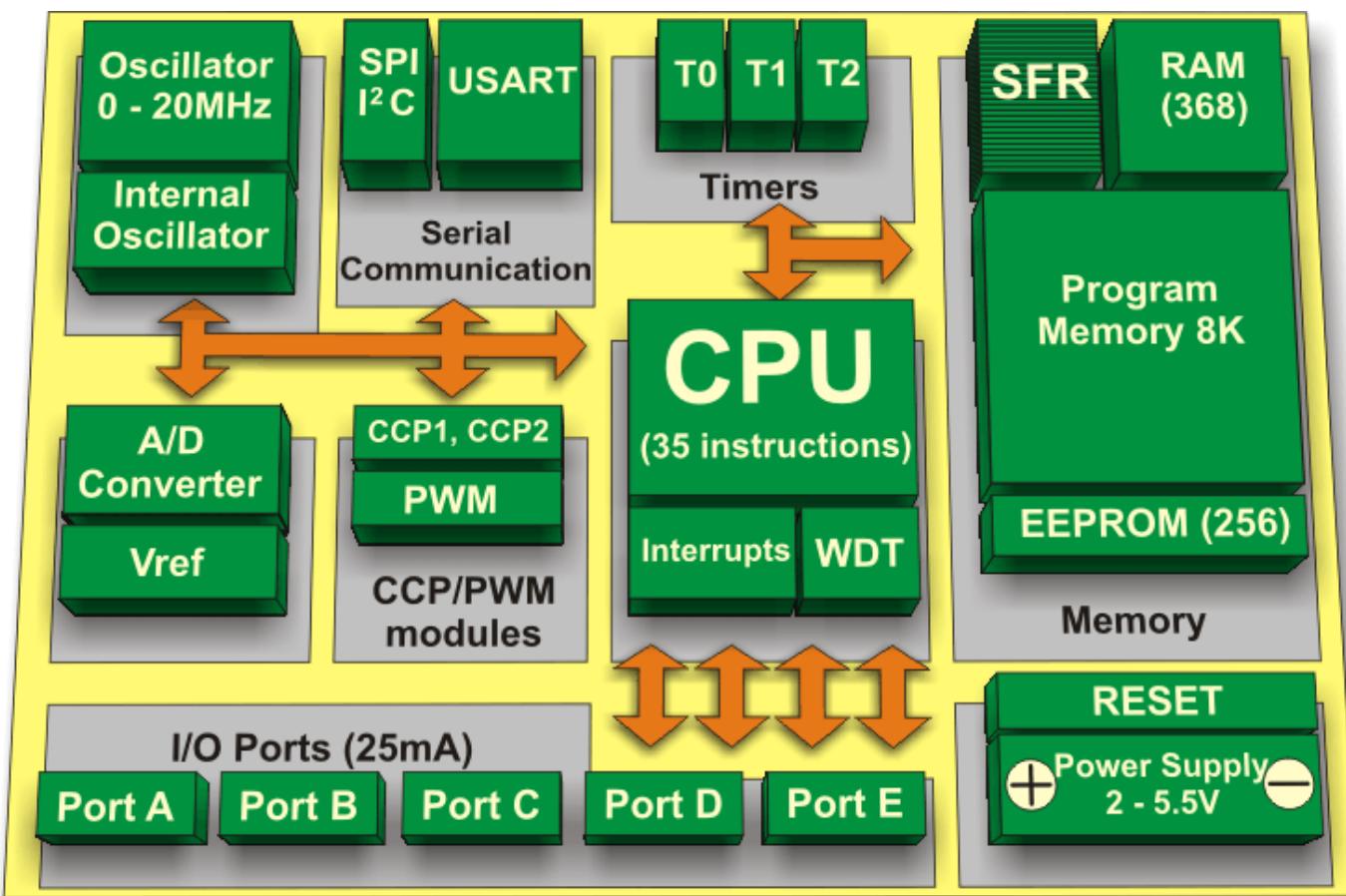
- Có 35 chân I/O cho phép lựa chọn hướng độc lập:
 - Mỗi ngõ ra có thể nhận/cấp dòng lớn khoảng 25mA nên có thể trực tiếp điều khiển led.

- Có các port báo ngắt khi có thay đổi mức logic.
- Có các port có điện trở kéo lên bên trong có thể lập trình.
- Có ngõ vào báo thức khởi động công suất cực thấp.
- Có module so sánh tương tự:
 - Có 2 bộ so sánh điện áp tương tự
 - Có module nguồn điện áp tham chiếu có thể lập trình.
 - Có nguồn điện áp tham chiếu cố định có giá trị bằng 0,6V.
 - Có các ngõ vào và các ngõ ra của bộ so sánh điện áp.
 - Có chế độ chốt SR.
- Có bộ chuyển đổi tương tự sang số:
 - Có 14 bộ chuyển đổi tương tự với độ phân giải 10 bit.
- Có timer0: 8 bit hoạt động định thời/đếm xung ngoại có bộ chia trước có thể lập trình.
- Có timer1:
 - 16 bit hoạt động định thời/đếm xung ngoại có bộ chia trước có thể lập trình.
 - Có ngõ vào công của timer1 để có thể điều khiển timer1 đếm từ tín hiệu bên ngoài.
 - Có bộ dao động công suất thấp có tần số 32kHz.
- Có timer2: 8 bit hoạt động định thời với thanh ghi chu kỳ, có bộ chia trước và chia sau.
- Có module capture, compare và điều chế xung PWM+ nâng cao
 - Có bộ capture 16 bit có thể đếm được xung với độ phân giải cao nhất là 12,5ns.
 - Có bộ điều chế xung PWM với số kênh ngõ ra là 1, 2 hoặc 4, có thể lập trình với tần số lớn nhất là 20kHz.
 - Có ngõ ra PWM điều khiển lái.
- Có module capture, compare và điều chế xung PWM
 - Có bộ capture 16 bit có thể đếm được xung với chu kỳ cao nhất là 12,5ns.
 - Có bộ so sánh 16 bit có thể so sánh xung đếm với chu kỳ lớn nhất là 200ns
 - Có bộ điều chế xung PWM có thể lập trình với tần số lớn nhất là 20kHz.
- Có thể lập trình trên bo ISP thông qua 2 chân.
- Có module truyền dữ liệu nối tiếp đồng bộ MSSP hỗ trợ chuẩn truyền 3 dây SPI, chuẩn I2C ở 2 chế độ chủ và tớ.

Bảng 1-1. Trình bày tóm tắt cấu trúc của 5 loại PIC16F88X.

Device	Program Memory	Data Memory		I/O	10-bit A/D (ch)	ECCP/ CCP	EUSART	MSSP	Comparators	Timers 8/16-bit
	Flash (words)	SRAM (bytes)	EEPROM (bytes)							
PIC16F882	2048	128	128	28	11	1/1	1	1	2	2/1
PIC16F883	4096	256	256	24	11	1/1	1	1	2	2/1
PIC16F884	4096	256	256	35	14	1/1	1	1	2	2/1
PIC16F886	8192	368	256	24	11	1/1	1	1	2	2/1
PIC16F887	8192	368	256	35	14	1/1	1	1	2	2/1

Cấu hình của vi điều khiển được minh họa bằng hình ảnh như hình 1-5:



Hình 1-5. Cấu hình của vi điều khiển.

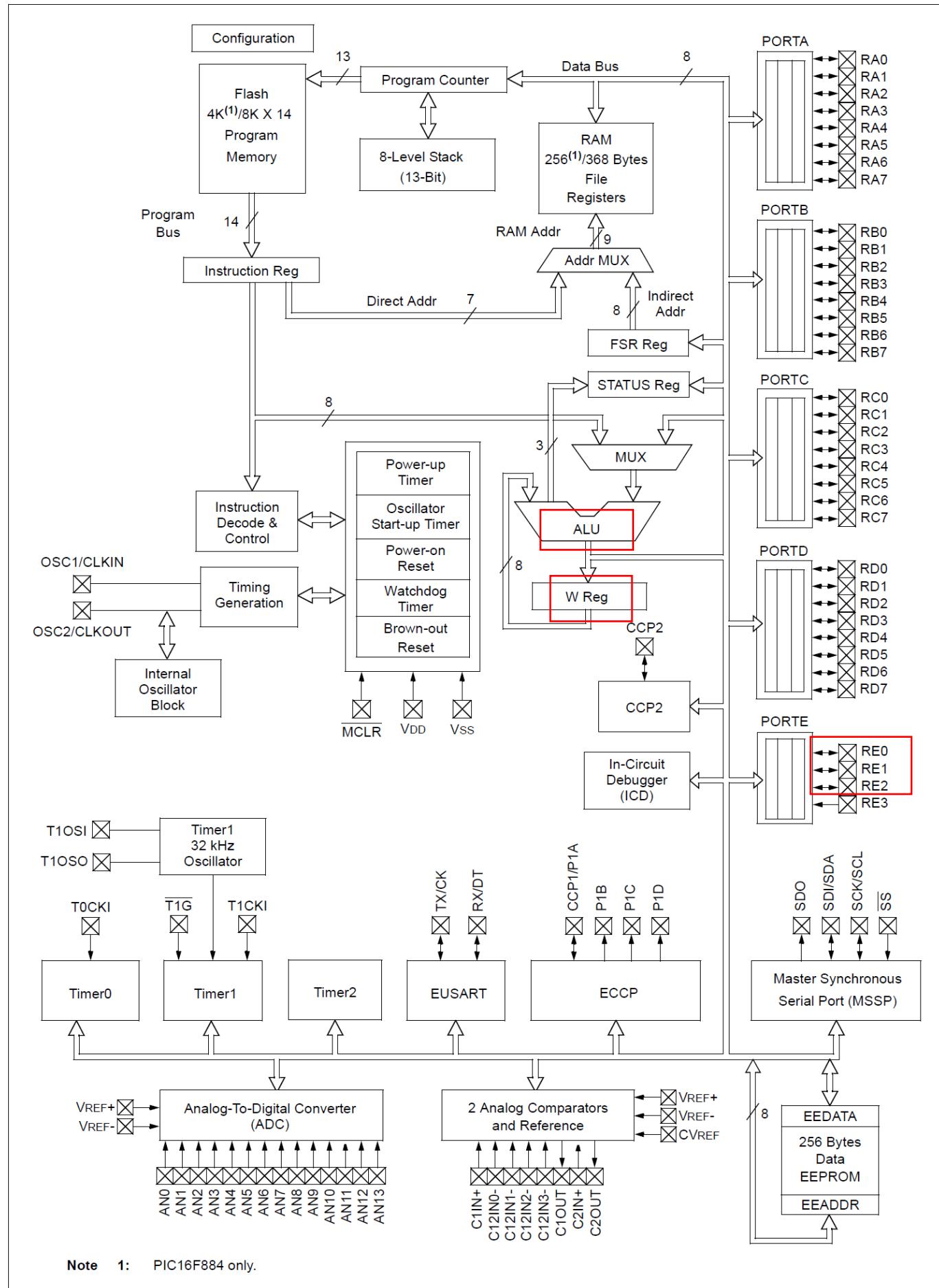
1.2.2 SƠ ĐỒ CẤU TRÚC CỦA VI ĐIỀU KHIỂN PIC 16F887

Sơ đồ cấu trúc vi điều khiển được trình bày ở hình 1-6.

Các khối bên trong vi điều khiển bao gồm:

- Có khối thanh ghi định cấu hình cho vi điều khiển.
- Có khối bộ nhớ chương trình có nhiều dung lượng cho 5 loại khác nhau.
- Có khối bộ nhớ ngăn xếp 8 cấp (8 level stack).
- Có khối bộ nhớ Ram cùng với thanh ghi FSR để tính toán tạo địa chỉ cho 2 cách truy xuất gián tiếp và trực tiếp.
- Có thanh ghi lệnh (Instruction register) dùng để lưu mã lệnh nhận về từ bộ nhớ chương trình.

- Có thanh ghi bộ đếm chương trình (PC) dùng để quản lý địa chỉ của bộ nhớ chương trình.



Hình 1-6. Cấu trúc bên trong của vi điều khiển.

- Có thanh ghi trạng thái (status register) cho biết trạng thái sau khi tính toán của khối ALU.
- Có thanh ghi FSR.
- Có khối ALU cùng với thanh ghi working hay thanh ghi A để xử lý dữ liệu.
- Có khối giải mã lệnh và điều khiển (Instruction Decode and Control).
- Có khối dao động nội (Internal Oscillator Block).
- Có khối dao động kết nối với 2 ngõ vào OSC1 và OSC2 để tạo dao động.
- Có khối các bộ định thời khi cấp điện PUT, có bộ định thời chờ dao động ổn định, có mạch reset khi có điện, có bộ định thời giám sát watchdog, có mạch reset khi phát hiện sụt giảm nguồn.
- Có khối bộ dao động cho timer1 có tần số 32kHz kết nối với 2 ngõ vào T1OSI và T1OSO.
- Có khối CCP2 và ECCP.
- Có khối mạch gỡ rối (In-Circuit Debugger IDC).
- Có khối timer0 với ngõ vào xung đếm từ bên ngoài là T0CKI.
- Có khối truyền dữ liệu đồng bộ/bát đồng bộ nâng cao.
- Có khối truyền dữ liệu đồng bộ MSSP cho SPI và I2C.
- Có khối bộ nhớ Eeprom 256 byte và thanh ghi quản lý địa chỉ EEADDR và thanh ghi dữ liệu EEDATA.
- Có khối chuyển đổi tín hiệu tương tự sang số ADC.
- Có khối 2 bộ so sánh với nhiều ngõ vào ra và điện áp tham chiếu.
- Có khối các port A, B, C, E và D

1.2.3 KHẢO SÁT SƠ ĐỒ CHÂN VI ĐIỀU KHIỂN PIC16F887

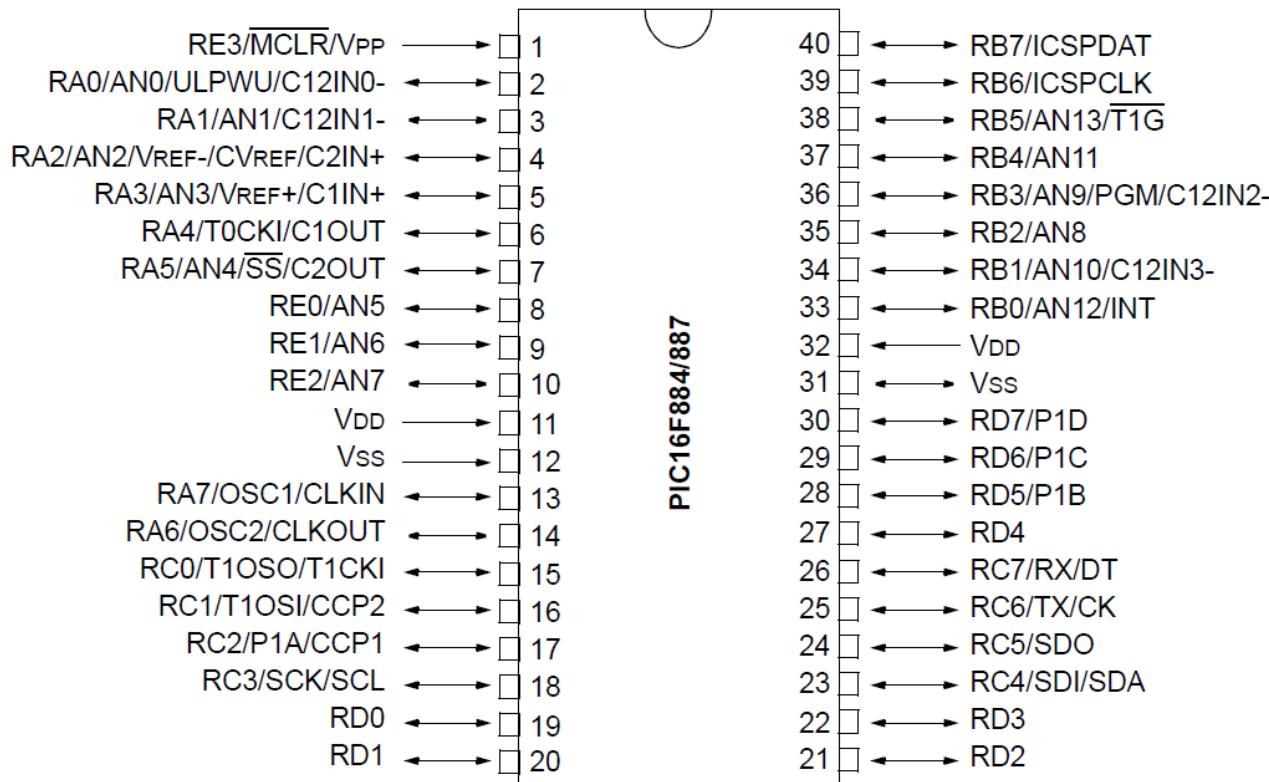
Sơ đồ chân của vi điều khiển PIC16F887 loại 40 chân được trình bày ở hình 1-7.

Vi điều khiển PIC16F887 loại 40 chân, trong đó các chân đều tích hợp nhiều chức năng, chức năng của từng chân được khảo sát theo port.

a. *Chức năng các chân của portA*

- Chân RA0/AN0/ULPWU/C12IN0- (2): có 4 chức năng:
 - RA0: xuất/ nhập số - bit thứ 0 của port A.
 - AN0: ngõ vào tương tự của kênh thứ 0.
 - ULPWU (Ultra Low-power Wake up input): ngõ vào đánh thức CPU công suất cực thấp.
 - C12IN0- (Comparator C1 or C2 negative input): ngõ vào âm thứ 0 của bộ so sánh C1 hoặc C2.
- Chân RA1/AN1/C12IN1- (3): có 3 chức năng:
 - RA1: xuất/nhập số - bit thứ 1 của port A.
 - AN1: ngõ vào tương tự của kênh thứ 1.

- C12IN1- (Comparator C1 or C2 negative input): ngõ vào âm thứ 1 của bộ so sánh C1 hoặc C2.



Hình 1-7. Sơ đồ chân của PIC 16F887.

- Chân RA2/AN2/VREF-/CVREF/C2IN+ (4): có 5 chức năng:
 - RA2: xuất/nhập số - bit thứ 2 của port A.
 - AN2: ngõ vào tương tự của kênh thứ 2.
 - VREF-: ngõ vào điện áp chuẩn (thấp) của bộ ADC.
 - CVREF: điện áp tham chiếu VREF ngõ vào bộ so sánh.
 - C2IN+: ngõ vào dương của bộ so sánh C2.
- Chân RA3/AN3/VREF+/C1IN+ (5): có 4 chức năng:
 - RA3: xuất/nhập số - bit thứ 3 của port A.
 - AN3: ngõ vào tương tự kênh thứ 3.
 - VREF+: ngõ vào điện áp chuẩn (cao) của bộ A/D.
 - C1IN+: ngõ vào dương của bộ so sánh C1.
- Chân RA4/TOCKI/C1OUT (6): có 3 chức năng:
 - RA4: xuất/nhập số – bit thứ 4 của port A.
 - TOCKI: ngõ vào xung clock từ bên ngoài cho Timer0.
 - C1OUT: ngõ ra bộ so sánh 1.
- Chân RA5/AN4/SS / C2OUT (7): có 4 chức năng:
 - RA5: xuất/nhập số – bit thứ 5 của port A.
 - AN4: ngõ vào tương tự kênh thứ 4.

- \overline{SS} : ngõ vào chọn lựa SPI tớ (Slave SPI device).
- C2OUT: ngõ ra bộ so sánh 2.
- Chân RA6/OSC2/CLKOUT (14): có 3 chức năng:
 - RA6: xuất/nhập số – bit thứ 6 của port A.
 - OSC2: ngõ ra dao động thạch anh. Kết nối đến thạch anh hoặc bộ cộng hưởng.
 - CLKOUT: ở chế độ RC, ngõ ra của OSC2, bằng $\frac{1}{4}$ tần số của OSC1.
- Chân RA7/OSC1/CLKIN (13): có 3 chức năng:
 - RA7: xuất/nhập số – bit thứ 7 của port A.
 - OSC1: ngõ vào dao động thạch anh hoặc ngõ vào nguồn xung ở bên ngoài.
 - CLKI: ngõ vào nguồn xung bên ngoài.

b. Chức năng các chân của portB

- Chân RB0/AN12/INT (33): có 3 chức năng:
 - RB0: xuất/nhập số – bit thứ 0 của port B.
 - AN12: ngõ vào tương tự kênh thứ 12.
 - INT: ngõ vào nhận tín hiệu ngắn ngoài.
- Chân RB1/AN10/C12IN3- (34): có 3 chức năng:
 - RB1: xuất/nhập số – bit thứ 1 của port B.
 - AN10: ngõ vào tương tự kênh thứ 10.
 - C12IN3-: ngõ vào âm thứ 3 của bộ so sánh C1 hoặc C2.
- Chân RB2/AN8 (35): có 2 chức năng:
 - RB2: xuất/nhập số – bit thứ 2 của port B.
 - AN8: ngõ vào tương tự kênh thứ 8.
- Chân RB3/AN9/PGM/C12IN2 (36): có 4 chức năng:
 - RB3: xuất/nhập số – bit thứ 3 của port B.
 - AN9: ngõ vào tương tự kênh thứ 9.
 - PGM: Chân cho phép lập trình điện áp thấp ICSP.
 - C12IN1-: ngõ vào âm thứ 2 của bộ so sánh C1 hoặc C2
- Chân RB4/AN11 (37): có 2 chức năng:
 - RB4: xuất/nhập số – bit thứ 4 của port B.
 - AN11: ngõ vào tương tự kênh thứ 11.
- Chân RB5/ AN13/ $\overline{T1G}$ (38): có 3 chức năng:
 - RB5: xuất/nhập số – bit thứ 5 của port B.
 - AN13: ngõ vào tương tự kênh thứ 13.
 - $\overline{T1G}$ (Timer1 gate input): ngõ vào Gate cho phép time1 đếm dùng để đếm độ rộng xung.
- Chân RB6/ICSPCLK (39): có 2 chức năng:

- RB6: xuất/nhập số.
- ICSPCLK: xung clock lập trình nối tiếp.
- Chân RB7/ICSPDAT (40): có 2 chức năng:
 - RB7: xuất/nhập số.
 - ICSPDAT: ngõ xuất nhập dữ liệu lập trình nối tiếp.
- c. ***Chức năng các chân của portC***
- Chân RC0/T1OSO/T1CKI (15): có 3 chức năng:
 - RC0: xuất/nhập số – bit thứ 0 của port C.
 - T1OSO: ngõ ra của bộ dao động Timer1.
 - T1CKI: ngõ vào xung clock từ bên ngoài Timer1.
- Chân RC1/T1OSI/CCP2 (16): có 3 chức năng:
 - RC1: xuất/nhập số – bit thứ 1 của port C.
 - T1OSI: ngõ vào của bộ dao động Timer1.
 - CCP2: ngõ vào Capture2, ngõ ra compare2, ngõ ra PWM2.
- Chân RC2 /P1A/CCP1 (17): có 3 chức năng:
 - RC2: xuất/nhập số – bit thứ 2 của port C.
 - P1A: ngõ ra PWM.
 - CCP1: ngõ vào Capture1, ngõ ra compare1, ngõ ra PWM1.
- Chân RC3/SCK/SCL (18): có 3 chức năng:
 - RC3: xuất/nhập số – bit thứ 3 của port C.
 - SCK: ngõ vào xung clock nối tiếp đồng bộ/ngõ ra của chế độ SPI.
 - SCL: ngõ vào xung clock nối tiếp đồng bộ/ngõ ra của chế độ I²C.
- Chân RC4/SDI/SDA (23): có 3 chức năng:
 - RC4: xuất/nhập số – bit thứ 4 của port C.
 - SDI: ngõ vào dữ liệu trong truyền dữ liệu kiểu SPI.
 - SDA: xuất/nhập dữ liệu I²C.
- Chân RC5/SDO (24): có 2 chức năng:
 - RC5: xuất/nhập số – bit thứ 5 của port C.
 - SDO: ngõ xuất dữ liệu trong truyền dữ liệu kiểu SPI.
- Chân RC6/TX/CK (25): có 3 chức năng:
 - RC6: xuất/nhập số – bit thứ 6 của port C.
 - TX: ngõ ra phát dữ liệu trong chế độ truyền bất đồng bộ USART.
 - CK: ngõ ra cấp xung clock trong chế độ truyền đồng bộ USART.
- Chân RC7/RX/DT (26): có 3 chức năng:
 - RC7: xuất/nhập số – bit thứ 7 của port C.
 - RX: ngõ vào nhận dữ liệu trong chế độ truyền bất đồng bộ EUSART.

- DT: ngõ phát và nhận dữ liệu ở chế độ truyền đồng bộ EUSART.

d. Chức năng các chân của portD

- Chân RD0 (19): có 1 chức năng:
 - RD0: xuất/nhập số – bit thứ 0 của port D.
- Chân RD1 (20): có 1 chức năng:
 - RD1: xuất/nhập số – bit thứ 1 của port D.
- Chân RD2 (21): có 1 chức năng:
 - RD2: xuất/nhập số – bit thứ 2 của port D.
- Chân RD3 (22): có 1 chức năng:
 - RD3: xuất/nhập số – bit thứ 3 của port D.
- Chân RD4 (27): có 1 chức năng:
 - RD4: xuất/nhập số – bit thứ 4 của port D.
- Chân RD5/ P1B (28): có 2 chức năng:
 - RD5: xuất/nhập số – bit thứ 5 của port D.
 - P1B: ngõ ra PWM.
- Chân RD6/ P1C (29): có 2 chức năng:
 - RD6: xuất/nhập số – bit thứ 6 của port D.
 - P1C: ngõ ra PWM.
- Chân RD7/P1D (30): có 2 chức năng:
 - RD7: xuất/nhập số – bit thứ 7 của port D.
 - P1D: ngõ ra tăng cường CPP1
- Chân RE0/AN5 (8): có 2 chức năng:
 - RE0: xuất/nhập số.
 - AN5: ngõ vào tương tự 5.
- Chân RE1/AN6 (9): có 2 chức năng:
 - RE1: xuất/nhập số.
 - AN6: ngõ vào tương tự kênh thứ 6.
- Chân RE2/AN7 (10): có 2 chức năng:
 - RE2: xuất/nhập số.
 - AN7: ngõ vào tương tự kênh thứ 7.
- Chân RE3/MCLR /V_{PP} (1): có 3 chức năng:
 - RE3: xuất/nhập số - bit thứ 3 của port E.
 - MCLR : là ngõ vào reset tích cực mức thấp.
 - V_{PP}: ngõ vào nhận điện áp khi ghi dữ liệu vào bộ nhớ nội flash.
- Chân VDD (11), (32):
 - Nguồn cung cấp dương từ 2V đến 5V.

- Chân VSS (12), (31):
 - Nguồn cung cấp 0V.

e. Chức năng các chân phân chia theo nhóm chức năng

- Chức năng là port I/O:
 - PortA gồm các tín hiệu từ RA0 đến RA7.
 - PortB gồm các tín hiệu từ RB0 đến RB7.
 - PortC gồm các tín hiệu từ RC0 đến RC7.
 - PortD gồm các tín hiệu từ RD0 đến RD7.
 - PortE gồm các tín hiệu từ RE0 đến RE3.
- Chức năng tương tự là các ngõ vào bộ chuyển đổi ADC: có 14 kênh
 - 14 kênh ngõ vào tương tự từ AN0 đến AN13.
 - Hai ngõ vào nhận điện áp tham chiếu bên ngoài là Vref+ và Vref-.
- Chức năng tương tự là các ngõ vào bộ so sánh C1 và C2: có 2 bộ so sánh
 - Có 4 ngõ vào nhận điện áp ngõ vào âm của 2 bộ so sánh là: C12IN0-, C12IN1-, C12IN2-, C12IN3-.
 - Có 2 ngõ vào nhận điện áp tương tự dương cho 2 bộ so sánh là: C1IN+ và C2IN+.
 - Có 2 ngõ ra của 2 bộ so sánh là: C1OUT và C2OUT.
 - Có 1 ngõ vào nhận điện áp tham chiếu chuẩn cấp cho 2 bộ so sánh là: CVREF.
- Chức năng dao động cấp xung cho CPU hoạt động:
 - Có 2 ngõ vào nối với tụ thạch anh để tạo dao động là OSC1 và OSC2.
 - Có 1 ngõ vào nhận tín hiệu dao động từ nguồn khác là CLKIN nếu không dùng tụ thạch anh, có 1 ngõ ra cấp xung clock cho thiết bị khác là CLKOUT.
- Chức năng nhận xung ngoại của T0 và T1:
 - Có 1 ngõ vào nhận xung ngoại cho timer/counter T0 có tên là T0CKI.
 - Có 1 ngõ vào nhận xung ngoại cho timer/counter T1 có tên là T1CKI.
 - Có 2 ngõ vào tạo dao động riêng cho Timer1 hoạt động độc lập có tên là T1OSO và T1OSI.
- Chức năng truyền dữ liệu SPI:
 - Có 1 ngõ vào nhận dữ liệu là SDI.
 - Có 1 ngõ ra phát dữ liệu là SDO.
 - Có 1 ngõ ra phát xung clock là SCK.
 - Có 1 ngõ vào chọn chip khi hoạt động ở chế độ tơ là SS.
- Chức năng truyền dữ liệu I2C:
 - Có 1 ngõ truyền/nhận dữ liệu là SDA.
 - Có 1 ngõ ra phát xung clock là SCL.
- Chức năng truyền dữ liệu đồng bộ ESUART:

- Có 1 ngõ truyền/nhận dữ liệu là DT.
- Có 1 ngõ ra phát xung clock là CK.
- Chức năng truyền dữ liệu không đồng bộ ESUART:
 - Có 1 ngõ nhận dữ liệu là RX.
 - Có 1 ngõ phát dữ liệu là TX.
- Chức năng ngắn:
 - Có 1 ngõ nhận tín hiệu ngắn cứng là INT.
- Chức năng CCP (capture, compare, pulse width modulation):
 - Có 2 tín hiệu cho khối CCP là CCP1 và CCP2.
 - Có 4 tín hiệu cho khối PWM là P1A, P1B, P1C, P1D.
- Chức năng nạp chương trình vào bộ nhớ flash:
 - Có 1 tín hiệu để truyền dữ liệu là ICSPDAT.
 - Có 1 tín hiệu để nhận xung clock là ICSPCLK.
 - Có 1 tín hiệu để điều khiển nạp là PGM.
 - Có 1 tín hiệu để nhận điện áp lập trình là V_{PP}.
- Có 1 ngõ vào reset có tên là MCLR (master clear).
- Có 4 chân cấp nguồn: VDD cấp nguồn dương, VSS nối với 0V.

1.3 CÂU HỎI ÔN TẬP – TRẮC NGHIỆM – BÀI TẬP

1.3.1 CÂU HỎI ÔN TẬP

Câu số 1-1: Hãy nêu cấu hình của vi điều khiển PIC16F887.

Câu số 1-2: Hãy cho biết các loại bộ nhớ mà vi điều khiển PIC16F887 tích hợp.

Câu số 1-3: Hãy trình bày tên và chức năng portA của vi điều khiển PIC16F887.

Câu số 1-4: Hãy trình bày tên và chức năng portB của vi điều khiển PIC16F887.

Câu số 1-5: Hãy trình bày tên và chức năng portC của vi điều khiển PIC16F887.

Câu số 1-6: Hãy trình bày tên và chức năng portD của vi điều khiển PIC16F887.

Câu số 1-7: Hãy trình bày tên và chức năng portE của vi điều khiển PIC16F887.

1.3.2 CÂU HỎI MỞ RỘNG

Câu số 1-8: Hãy nêu cấu hình của vi điều khiển AT89S52.

Câu số 1-9: Hãy cho biết các loại bộ nhớ mà vi điều khiển AT89S52 tích hợp và mở rộng.

Câu số 1-10: Hãy trình bày tên và chức năng các port của vi điều khiển AT89S52.

Câu số 1-11: Hãy tìm hiểu quá trình phát triển của họ vi điều khiển MCS51 và MCS52.

Câu số 1-12: Hãy tìm hiểu các port vi điều khiển AT89C52 và so sánh với vi điều khiển AT89S52.

Câu số 1-13: Hãy tìm hiểu cấu hình vi điều khiển AT89S8252 và so sánh với vi điều khiển AT89S52.

Câu số 1-14: Hãy tìm hiểu cấu hình vi điều khiển AT89C51RD2 và so sánh với vi điều khiển AT89S52.

Câu số 1-15: Hãy tìm hiểu cấu hình vi điều khiển PIC16F877A và so sánh với vi điều khiển PIC16F887.

Câu số 1-16: Hãy tìm hiểu cấu hình vi điều khiển **PIC18F4550** và so sánh với vi điều khiển **PIC16F887**.

1.3.3 CÂU HỎI TRẮC NGHIỆM

Câu 1-1: PIC 16F887 có bao nhiêu port:

- | | |
|-------|-------|
| (a) 3 | (b) 4 |
| (c) 5 | (d) 6 |

Câu 1-2: Port nào của PIC 16F887 có 4 đường:

- | | |
|-------|-------|
| (a) A | (b) B |
| (c) C | (d) E |

Câu 1-3: PIC 16F887 có tích hợp ADC bao nhiêu bit:

- | | |
|------------|------------|
| (a) 8 bit | (b) 9 bit |
| (c) 10 bit | (d) 12 bit |

Câu 1-4: PIC 16F887 có tích hợp bao nhiêu kênh ADC:

- | | |
|-------------|-------------|
| (a) 8 kênh | (b) 14 kênh |
| (c) 10 kênh | (d) 12 kênh |

Câu 1-5: Các tín hiệu truyền dữ liệu I2C của PIC 16F887 có tên là:

- | | |
|--------------|--------------|
| (a) SDI, SCL | (b) SDI, SDO |
| (c) SCL, SDA | (d) SDA, SDI |

Câu 1-6: Các tín hiệu truyền dữ liệu SPI của PIC 16F887 có tên là:

- | | |
|-----------------------|-------------------|
| (a) SDI, SCL, SDO, SS | (b) SDI, SDO, SS |
| (c) SDI, SCK, SDO, SS | (d) SDA, SDI, SCK |

Câu 1-7: Các tín hiệu truyền dữ liệu UART của PIC 16F887 có tên là:

- | | |
|----------------|----------------|
| (a) TX, RX, CK | (b) TX, DT, CK |
| (c) TX, RX | (d) DT, CK |

Câu 1-8: Các tín hiệu truyền dữ liệu SART của PIC 16F887 có tên là:

(a) TX, RX, CK (b) TX, DT, CK

(c) TX, RX (d) DT, CK

Câu 1-9: Các tín hiệu nào nhận xung CK cho timer0:

(a) T0SCK (b) T0SCL

(c) T0CKI (d) T0CK

Câu 1-10: Các tín hiệu nào thiết lập điện áp tham chiếu cho ADC:

(a) CV_{REF} và V_{REF+} (b) CV_{REF} và V_{REF-}(c) V_{REF+} và V_{REF-} (d) CV_{REF-} và V_{REF-}

1.3.4 BÀI TẬP

VỊ ĐIỀU KHIỂN PIC16F887:

TỔ CHỨC BỘ NHỚ, THANH GHI

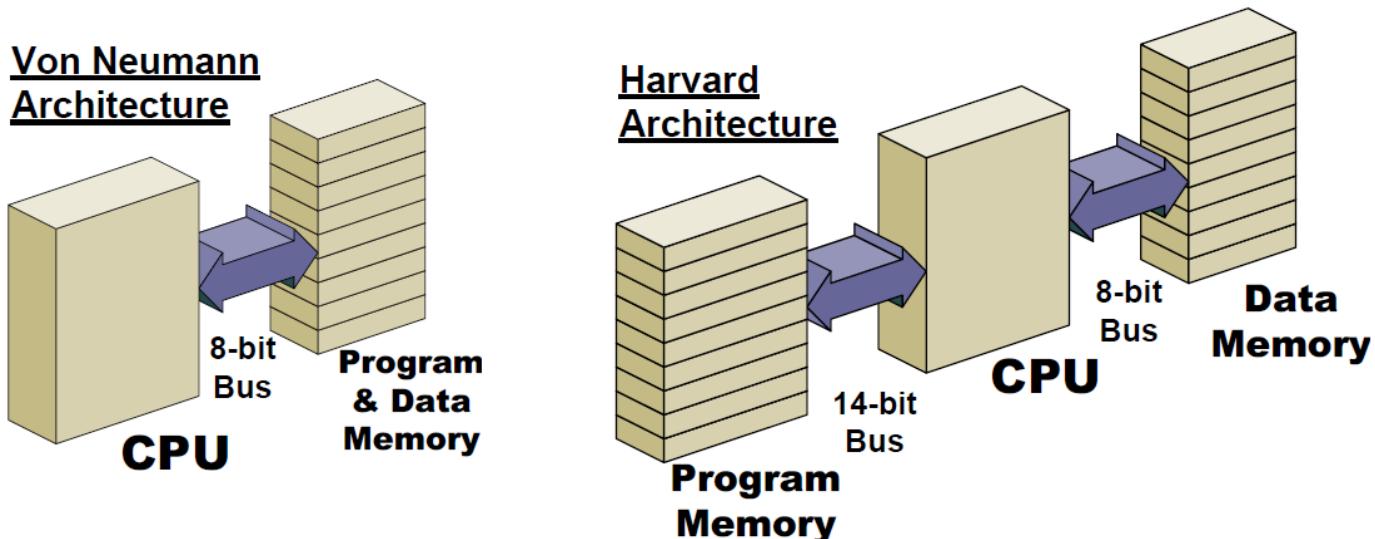
2.1 GIỚI THIỆU

Ở chương này khảo sát tổ chức bộ nhớ bên trong, các thanh ghi của vi điều khiển 8 bit. Sau khi kết thúc chương này thì người đọc có thể biết tổ chức bộ nhớ bên trong, chức năng của từng loại bộ nhớ, tên và chức năng của các thanh ghi đặc biệt.

2.2 KIẾN TRÚC BỘ NHỚ

Có 2 loại kiến trúc bộ nhớ cơ bản là kiến trúc Von Neumann và Harvard.

Hình 2-1 trình bày hai kiến trúc:



Hình 2-1. Kiến trúc Von Neumann và Harvard.

Kiến trúc Von Neumann: với kiến trúc này thì bộ nhớ giao tiếp với CPU thông qua 1 bus dữ liệu 8 bit, bộ nhớ có các ô nhớ chứa dữ liệu 8 bit, bộ nhớ vừa lưu trữ chương trình và dữ liệu.

Ưu điểm: kiến trúc đơn giản.

Khuyết điểm: do chỉ có 1 bus nên tốc độ truy suất chậm, khó thay đổi dung lượng lưu trữ của ô nhớ.

Kiến trúc Harvard: với kiến trúc này thì bộ nhớ được tách ra làm 2 loại bộ nhớ độc lập: bộ nhớ lưu chương trình và bộ nhớ lưu dữ liệu, CPU giao tiếp với 2 bộ nhớ độc lập nên cần 2 bus độc lập. Vì độc lập nên có thể thay đổi số bit lưu trữ của từng bộ nhớ mà không ảnh hưởng lẫn nhau.

Ưu điểm: do chỉ có 2 bus nên tốc độ truy suất nhanh, tùy ý thay đổi số bit của ô nhớ.

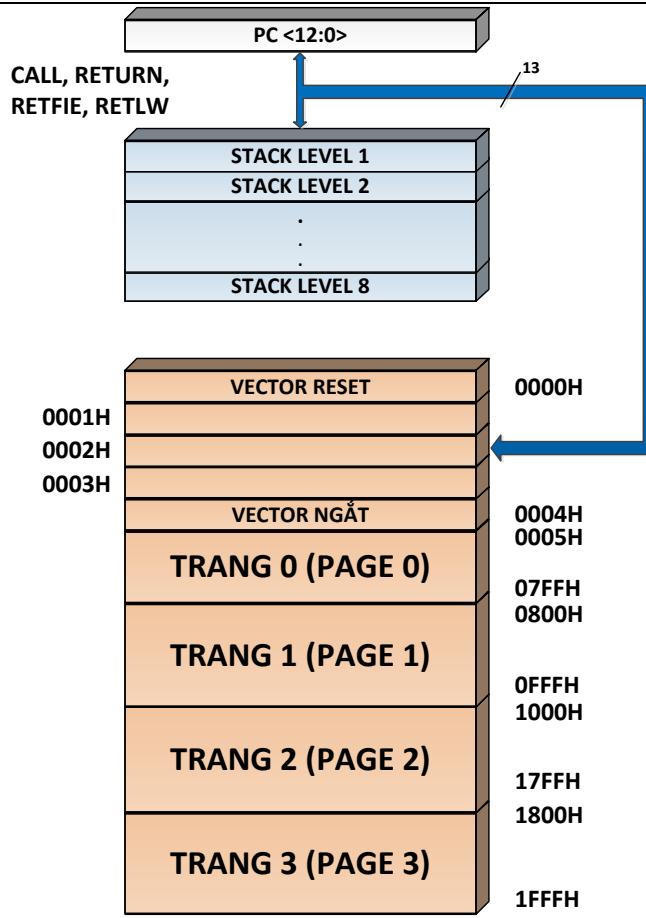
Khuyết điểm: kiến trúc phức tạp.

2.3 TỔ CHỨC BỘ NHỚ CỦA VI ĐIỀU KHIỂN PIC 16F887

2.3.1 TỔ CHỨC BỘ NHỚ CHƯƠNG TRÌNH VÀ NGĂN XÉP

a. Bộ nhớ chương trình

Bộ nhớ chương trình của PIC16F8xx có dung lượng 8K được chia làm 4 trang bộ nhớ, mỗi trang 2K, xem hình 2-2.



Hình 2-2. Sơ đồ bộ nhớ chương trình và ngăn xếp.

Bộ nhớ chương trình có chức năng dùng để lưu chương trình và các dữ liệu cố định, khi vi xử lý truy xuất bộ nhớ chương trình để lấy mã lệnh về xử lý thì chỉ truy xuất một từ dữ liệu hay một ô nhớ duy nhất và thực hiện trong một chu kỳ lệnh duy nhất.

Bộ nhớ có dung lượng 8K ô nhớ sẽ có 13 bit địa chỉ, để quản lý địa chỉ của bộ nhớ chương trình thì do thanh ghi bộ đếm chương trình (Program Counter - PC) đảm nhận.

b. Thanh ghi PC và PCLATH

Thanh ghi bộ đếm chương trình PC sẽ quản lý địa chỉ của bộ nhớ chương trình, thanh ghi PC có độ dài 13 bit sẽ quản lý 8192 ô nhớ tương đương với 8K ô nhớ. Số bit của thanh ghi PC có mối quan hệ với dung lượng bộ nhớ chương trình.

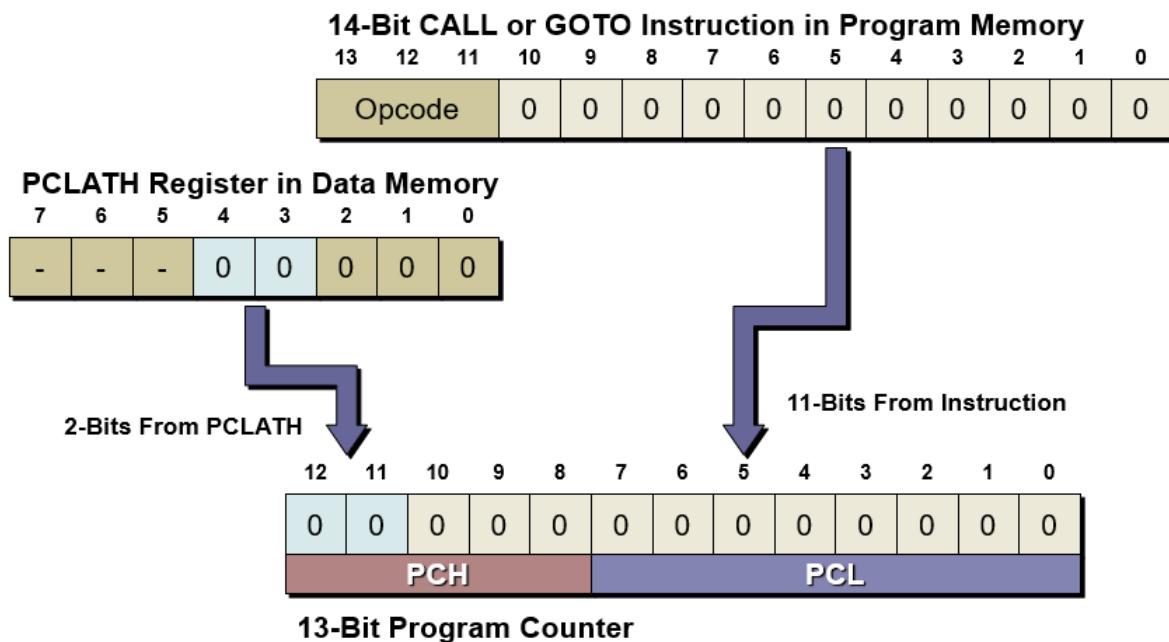
Với vi điều khiển PIC 16F887 thì mỗi ô nhớ chương trình có độ dài 14 bit.

Khi PIC bị reset thì thanh ghi PC có giá trị là 0000H và vi điều khiển PIC sẽ bắt đầu thực hiện chương trình tại địa chỉ 0000H.

Khi có bất kỳ ngắt nào tác động thì vi điều khiển PIC sẽ thực hiện chương trình phục vụ ngắt tại địa chỉ 0004H. Ở một số họ vi điều khiển khác thì có tới 2 địa chỉ ngắt, một cái có mức ưu tiên cao và 1 cái có mức ưu tiên thấp, việc lựa chọn ưu tiên sẽ được cấu hình trong thanh ghi cho phép ngắt.

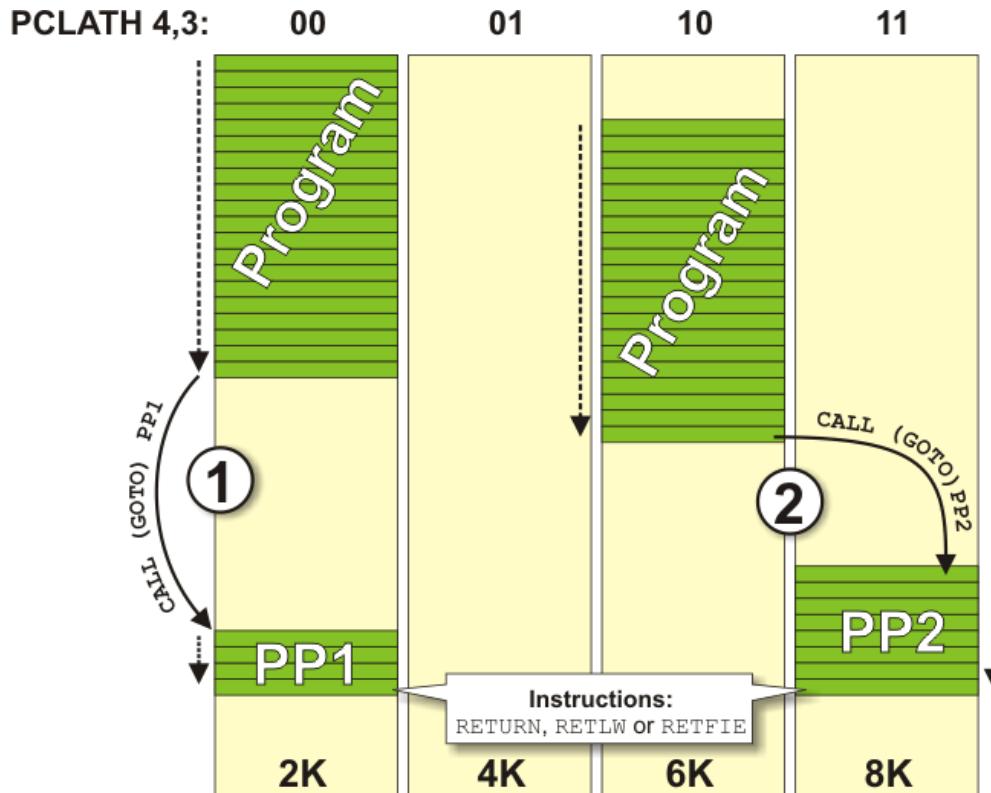
Việc phân chia theo trang bộ nhớ chỉ có tác dụng đối với lệnh nhảy và lệnh gọi chương trình con, khi truy xuất trong nội bộ mỗi trang thì chỉ có 11 bit địa chỉ thấp thay đổi, 2 bit địa chỉ cao giữ nguyên.

Khi thực hiện lệnh gọi hoặc lệnh nhảy thì từ mã lệnh 14 bit trong đó chứa 11 bit địa chỉ thấp và 2 bit địa chỉ cao lấy từ thanh ghi PCLATH, xem hình 2-3.



Hình 2-3. Nội dung thanh ghi PC khi thực hiện lệnh Call hay Goto.

Khi nhảy đến hoặc khi gọi chương trình con nằm trong cùng 1 trang thì lệnh sẽ viết ngắn gọn chỉ có 1 lệnh và mã lệnh chỉ là 1 từ bao gồm cả mã lệnh và địa chỉ 11 bit là địa chỉ của trang hiện tại, xem hình 2-4 trường hợp tương ứng với số 1 minh họa cho chức năng vừa nêu, nhảy trong cùng trang thứ 0.



Hình 2-4. Nhảy trong cùng 1 trang và khác trang bộ nhớ.

Khi nhảy hoặc gọi trong cùng 1 trang thì chỉ thay đổi 11 bit địa chỉ trong thanh ghi PC nên chỉ nhảy đến trong phạm vi 1 trang 2K. Hai bit địa chỉ thứ 12 và 13 lưu trong thanh ghi PCLATH giữ nguyên.

Đoạn chương trình minh họa thực hiện lệnh gọi chương trình con delay trong cùng 1 trang thứ 0:

org 0x000	;page 0
-----------	---------

```

call      delay
...
delay
    org 0x250          ;page 0
    movlw   0xff
    ...
    return

```

Trong đoạn chương trình trên ta không làm thay đổi nội dung của thanh ghi PCLATH nên mặc nhiên là truy xuất cùng trang.

Khi nơi nhảy đến hoặc khi gọi chương trình con nằm ở trang khác thì trước khi thực hiện lệnh nhảy hoặc gọi thì ta phải biết được nơi nhảy đến nằm ở trang nào để tiến hành thay đổi 2 bit địa chỉ cao trong thanh ghi PCLATH cho đúng rồi mới thực hiện lệnh nhảy hoặc gọi. Do có nhiều lệnh hơn nên mã lệnh dài hơn, xem hình xem hình 2-4 trường hợp tương ứng với số 2 minh họa cho chức năng vừa nêu, nhảy từ trang thứ 2 sang trang thứ 3.

Đoạn chương trình minh họa thực hiện lệnh gọi chương trình con delay ở trang khác:

```

org 0x1250          ;page 2
movlw   HIGH,delay
movwf   PCLATH
call    delay
...
delay
    org 0x1850          ;page 3
    movlw   0xff
    ...
    return

```

Trước khi gọi chương trình con ở trang 3 thì ta phải nạp địa chỉ bắt đầu của chương trình con vào thanh ghi W và chỉ lấy byte địa chỉ cao, thực ra chỉ quan tâm đến 2 bit địa chỉ thứ 12 và 13. Sau đó copy địa chỉ đó từ thanh ghi W sang thanh ghi PCLATH để cập nhật 2 bit địa chỉ mới có nghĩa là đã đổi trang bộ nhớ mới, tiến hành gọi chương trình con.

Vậy khi thực hiện khác trang sẽ cần nhiều lệnh hơn nên mã lệnh dài hơn.

Một câu hỏi đặt ra là làm sao ta biết được ta đang ở trang nào và nơi nhảy đến là trang nào? Có nhiều cách để biết nhưng đơn giản nhất như trong chương trình là các địa chỉ bắt đầu rất rõ ràng bằng các chỉ dẫn org 0XXXXX và thường thì phần mềm sẽ hỗ trợ cho bạn các chỉ dẫn để bạn nhảy cho đúng và không cần tính toán gì cả.

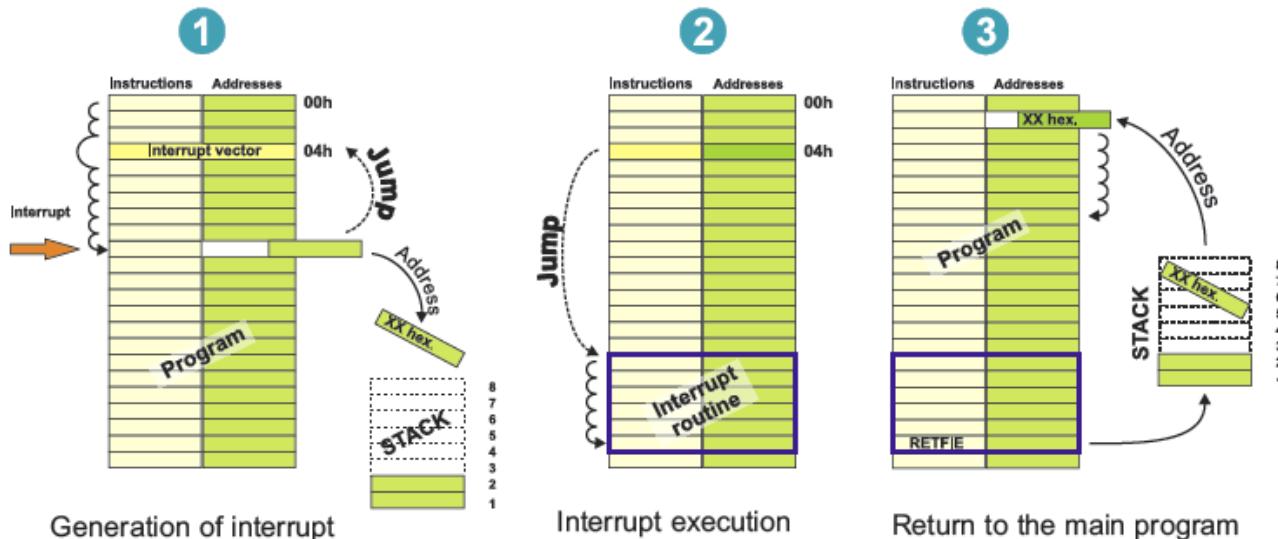
c. Bộ nhớ ngăn xếp

Trong các họ vi điều khiển khác thì bộ nhớ ngăn xếp dùng chung với bộ nhớ dữ liệu, ưu điểm là cấu trúc đơn giản, khuyết điểm là việc dùng chung nếu không biết giới hạn sẽ lấn chiếm lẫn nhau và làm mất dữ liệu lưu trong bộ nhớ ngăn xếp, khi đó dẫn đến vi điều khiển thực hiện sai chương trình.

Ở vi điều khiển PIC thì nhà thiết kế tách bộ nhớ ngăn xếp độc lập với bộ nhớ dữ liệu và chỉ để dùng lưu địa chỉ trả về của thanh ghi PC khi thực hiện lệnh gọi chương trình con và khi thực hiện ngắt, xem hình 2-5.

Trường hợp số 1: khi đang thực hiện chương trình thì có ngắt xảy ra, khi đó vi điều khiển sẽ ngừng chương trình đang thực hiện và thực hiện các công việc sau: **cắt địa chỉ của lệnh tiếp theo** của chương trình đang thực hiện vào bộ nhớ ngăn xếp, ngăn xếp đang có lưu 2

địa chỉ trước đó và bây giờ sẽ lưu thêm 1 địa chỉ mới, tổng cộng là đã sử dụng 3 ô nhớ. Tiếp theo thì vi điều khiển sẽ nhảy về địa chỉ ngắn là 0004H để thực hiện chương trình con của ngắt.



Hình 2-5. Bộ nhớ ngắn xếp khi thực hiện ngắt và kết thúc ngắt.

Trường hợp số 2: tại địa chỉ ngắn 0004H thì nơi đó chứa lệnh nhảy đến chương trình con ngắt nằm ở vùng nhớ bên dưới và bắt đầu thực hiện các lệnh của chương trình con phục vụ ngắt. Việc thực hiện sẽ khi thúc khi gặp lệnh trở về (Return).

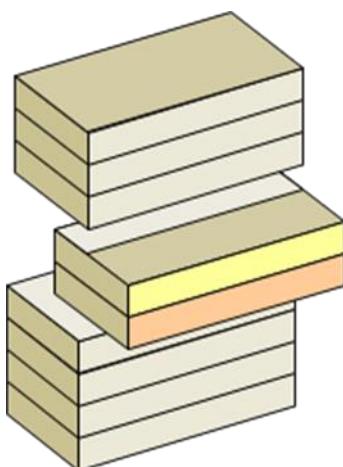
Trường hợp số 3: khi thực hiện lệnh kết thúc thì vi điều khiển sẽ lấy **địa chỉ của lệnh tiếp theo** trong bộ nhớ ngắn xếp để trả lại cho thanh ghi PC và khi đó chương trình bị gián đoạn sẽ tiếp tục được thực hiện tiếp. Khi đó bộ nhớ ngắn xếp trở chỉ còn lại 2 ô như trước.

Do chỉ có 8 ô nhớ nên khi thực hiện các chương trình con lồng vào nhau tối đa là 8 cấp. Do lưu địa chỉ trả về trong thanh ghi PC, mà thanh ghi PC có chiều dài 13 bit nên mỗi ô nhớ ngắn xếp có số bit là 13.

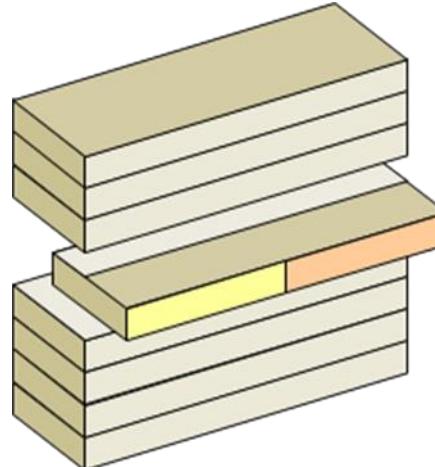
Khi không sử dụng ngắt thì chương trình có thể viết bắt đầu và liên tục tại địa chỉ 0000H, nhưng nếu sử dụng ngắt thì nên dùng lệnh nhảy để tránh vùng nhớ bắt đầu tại địa chỉ 0004H - vì vùng nhớ này dùng để viết chương trình con phục vụ ngắt.

2.3.2 MÃ LỆNH 14 BIT

Với các vi điều khiển 8 bit của các hãng khác thì bộ nhớ chương trình tổ chức theo đơn vị là byte, mỗi ô nhớ lưu trữ dữ liệu 1 byte – xem hình 2-6.



Hình 2-6. Tổ chức bộ nhớ theo byte.



Hình 2-7. Tổ chức bộ nhớ chứa cả mã lệnh và dữ liệu.

Nếu mã lệnh 2 byte gồm 1 byte mã lệnh và 1 byte dữ liệu hay địa chỉ thì dùng 2 ô nhớ liên tiếp để lưu và khi CPU đọc mã lệnh để thực hiện lệnh thì CPU phải thực hiện 2 lần đọc - mỗi lần 1 byte.

Với vi điều khiển PIC thì mỗi ô nhớ của bộ nhớ chương trình có thể lưu trữ dữ liệu nhiều bit bao gồm cả mã lệnh và dữ liệu – xem hình 2-7. Khi CPU đọc mã lệnh để thực hiện thì vi điều khiển CPU chỉ thực hiện 1 lần đọc cả mã lệnh và dữ liệu.

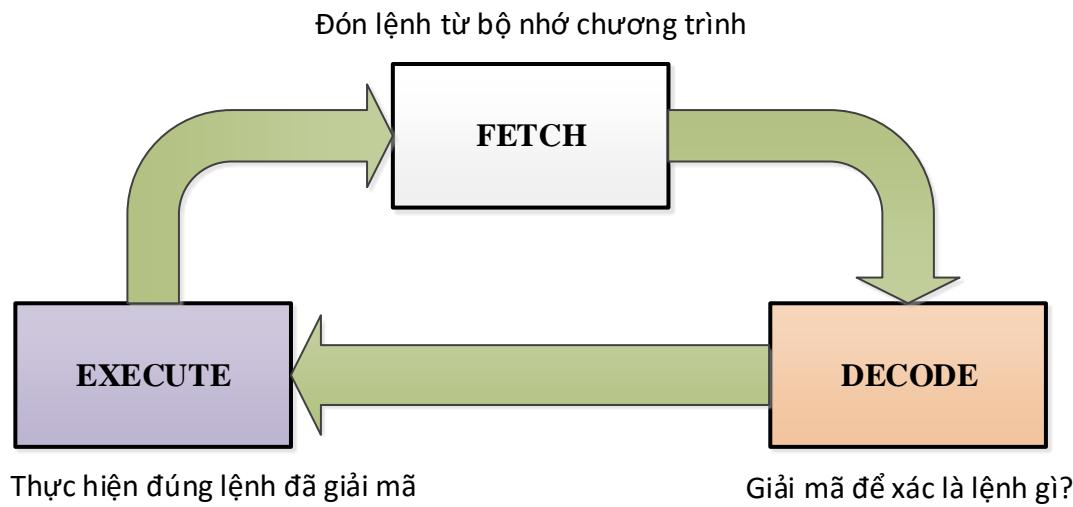
Vậy với tổ chức bộ nhớ của vi điều khiển PIC thì tiết kiệm được 1 chu kỳ đọc dữ liệu vì thế vi điều khiển PIC sẽ có tốc độ thực hiện chương trình nhanh hơn.

Tập lệnh của vi điều khiển PIC là tập lệnh rút gọn nên chỉ dùng có 6 bit nhị phân để mã hóa các lệnh, cùng với dữ liệu xử lý là byte - 8 bit nên tổng cộng là 14 bit.

2.3.3 CÁU TRÚC PIPELINE

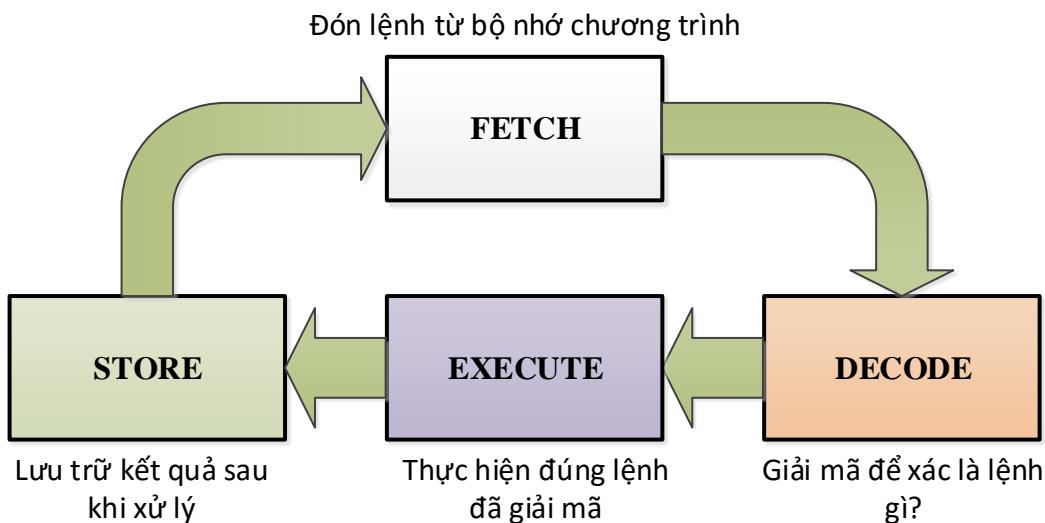
Các vi xử lý 8 bit như 8085, Z80 trước đây thì quá trình thực hiện lệnh theo nhiều nguồn tài liệu được chia thành 3 bước hoặc 4 bước hoặc 5 bước.

- Nếu là 3 bước thì các bước bao gồm: đón lệnh (Fetch), giải mã lệnh (Decode) và thực hiện lệnh (Execute) như hình 2-8.



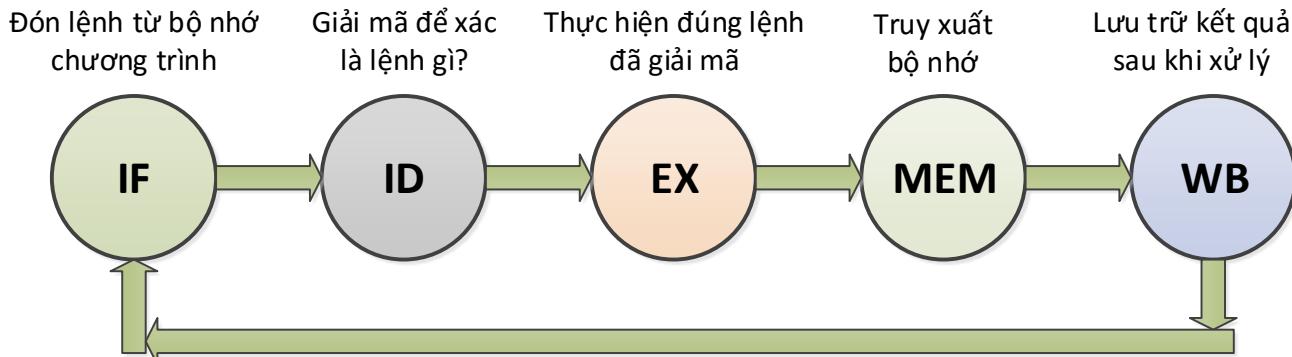
Hình 2-8. Quá trình thực hiện lệnh 3 bước.

- Nếu là 4 bước thì các bước bao gồm: đón lệnh (Fetch), giải mã lệnh (Decode) và thực hiện lệnh (Execute) rồi lưu kết quả sau khi xử lý (Store) như hình 2-9.

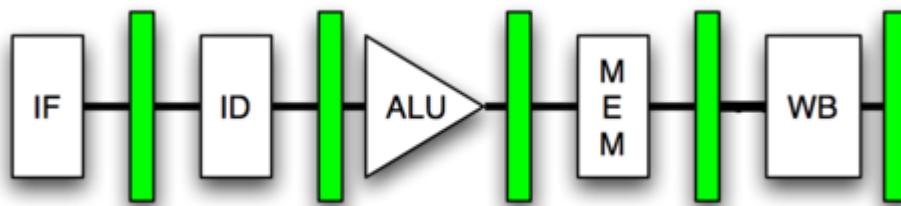


Hình 2-9. Quá trình thực hiện lệnh 4 bước.

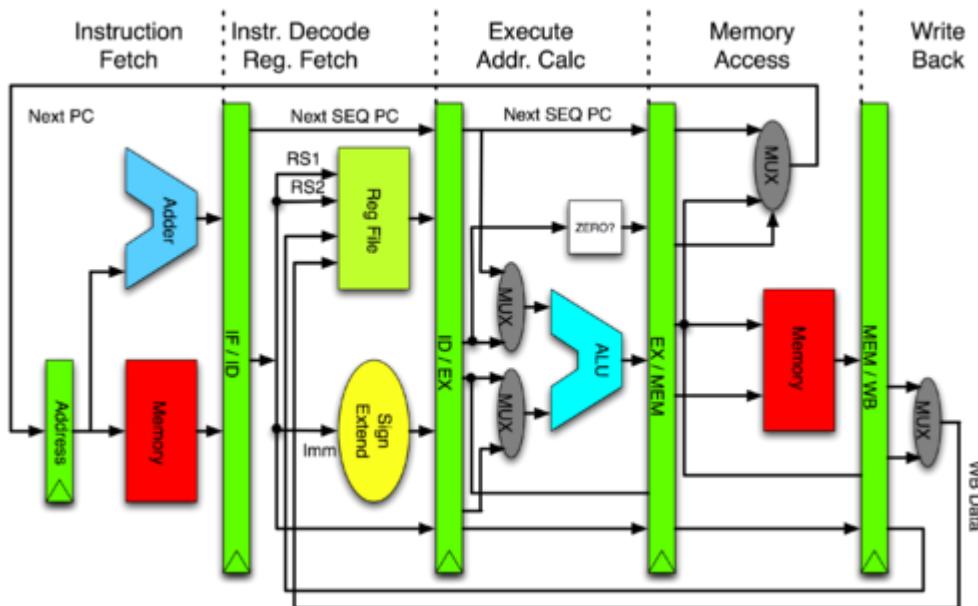
- Nếu là 5 bước thì các bước bao gồm: đón lệnh (Instruction Fetch - IF), giải mã lệnh (Instruction Decode - ID), thực hiện lệnh (Execute - EX), truy xuất bộ nhớ để lấy dữ liệu (Memory Read/Write), lưu kết quả sau khi xử lý (Write Back - WB) như hình 2-10, các bước được ký hiệu như hình 2-11, hình ảnh chi tiết của các bước như hình 2-12.



Hình 2-10. Quá trình thực hiện lệnh 5 bước.



Hình 2-11. Ký hiệu các bước của quá trình thực hiện lệnh 5 bước.



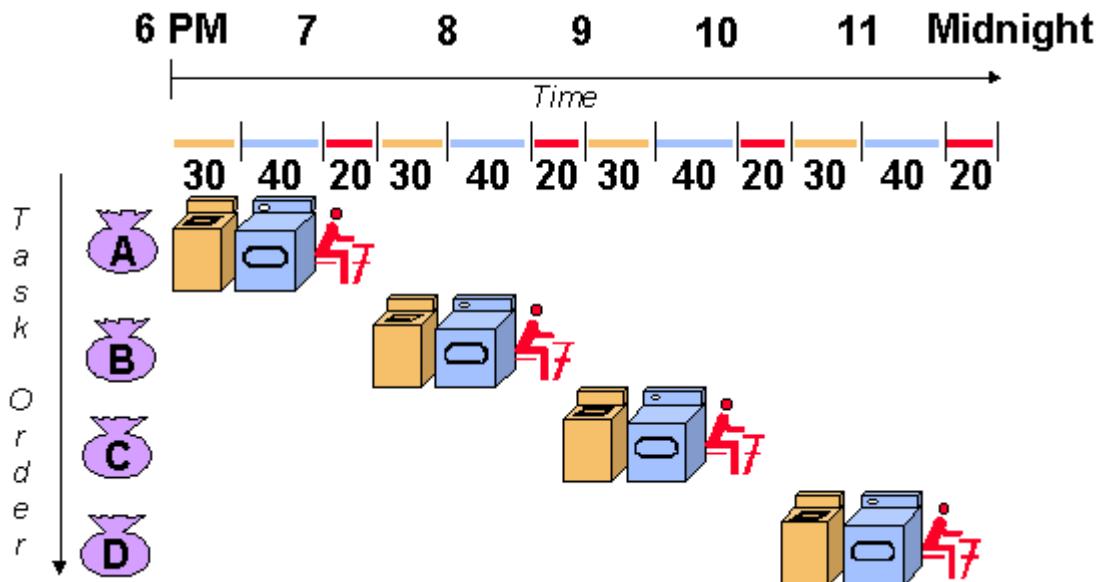
Hình 2-12. Các bước thực hiện chi tiết của quá trình thực hiện lệnh 5 bước.

Về cơ bản thực ra là một, chỉ khác là do lệnh khác nhau, có lệnh không có dữ liệu và địa chỉ, có lệnh có dữ liệu không có địa chỉ, có lệnh có cả địa chỉ 1 byte và dữ liệu 1 byte, có lệnh có 2 byte địa chỉ và 1 byte dữ liệu nên chúng khác nhau là vậy. Quá trình 5 bước thực ra là cách trình bày đầy đủ nhất.

Mỗi một lệnh được thực hiện đầy đủ các bước trên, mỗi bước có thể thực hiện với thời gian tính theo chu kỳ bộ dao động thường là 1 chu kỳ. Lệnh đầy đủ năm bước sẽ thực hiện hết 5 chu kỳ, tương tự cho các lệnh khác.

Vi điều khiển có tập lệnh và cách thức vừa trình bày ở trên được gọi là tập lệnh có cấu trúc phức tạp CISC (Complex Instruction Set Computer).

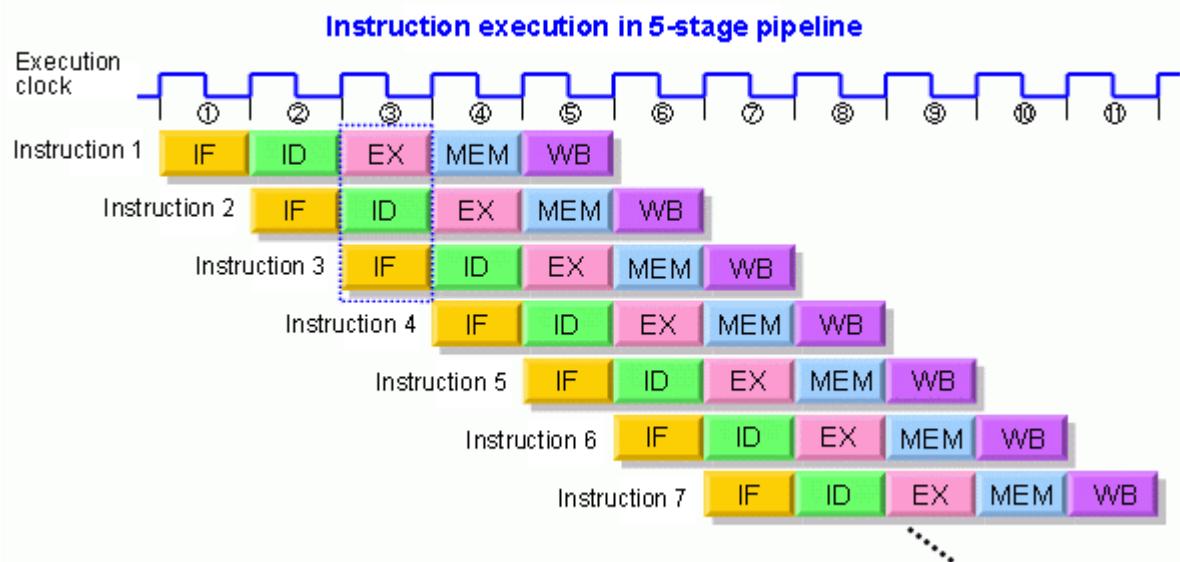
Ở vi xử lý hay vi điều khiển có tập lệnh CISC thì thực hiện từng lệnh: thực hiện lệnh này xong thì mới bắt đầu thực hiện lệnh tiếp theo. Với cách thực hiện này thì dẫn đến hiệu suất không cao hay tốc độ thực hiện chậm, xem hình minh họa 2-13.



Hình 2-13. Làm xong việc này mới đến việc khác.

Với sự nghiên cứu của còn người nhằm làm cho vi xử lý thực hiện càng nhanh càng tốt thì có rất nhiều vấn đề cần cải tiến, cải tiến về cấu trúc bộ nhớ như đã trình bày ở trên cũng là một cách, tăng số bit của từ dữ liệu thành 14 bit, 16 bit, 32 bit, ..., cải tiến tập lệnh, cải tiến quy trình thực hiện lệnh cũng là một cách.

Phần này trình bày cách cải tiến quy trình thực hiện lệnh theo cấu trúc pipeline, tạm dịch là đường ống lệnh. Để thực hiện cấu trúc pipeline thì các nhà nghiên cứu đã cải tiến quy trình đón lệnh, giải mã và thực hiện lệnh sao cho chúng gối đầu lên nhau như hình 2-14.



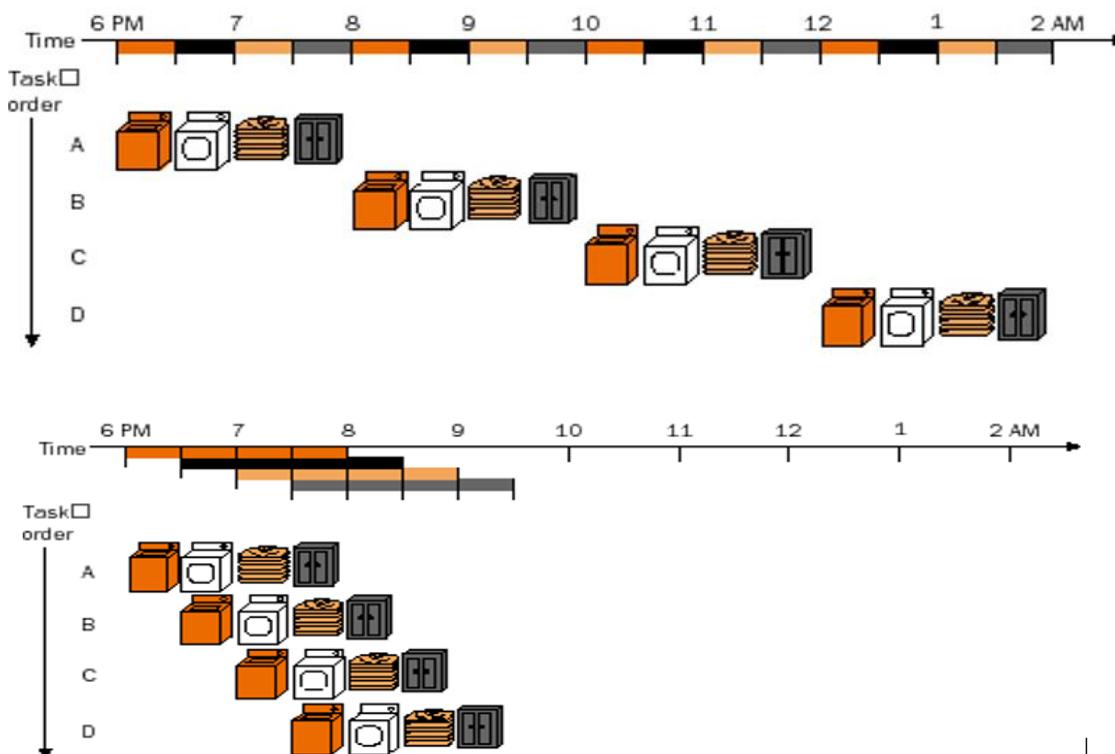
Hình 2-14. Làm theo cấu trúc pipeline hay dây chuyền.

Hoạt động của cấu trúc pipeline thì:

- ◆ Ở chu kỳ xung CK thứ nhất: lệnh thứ nhất được thực hiện bước 1 (IF) đón lệnh.
- ◆ Ở chu kỳ xung CK thứ hai:
 - Lệnh thứ nhất đang chuyển sang bước 2 (ID).
 - Lệnh thứ hai ở được thực hiện ở bước 1 (IF) đón lệnh.
- ◆ Ở chu kỳ xung CK thứ ba:
 - Lệnh thứ nhất đang chuyển sang bước 3 (EX).
 - Lệnh thứ hai ở được thực hiện ở bước 2 (ID) đón lệnh.
 - Lệnh thứ ba ở được thực hiện ở bước 1 (IF) đón lệnh.
- ◆ Ở chu kỳ xung CK thứ tư:
 - Lệnh thứ nhất đang chuyển sang bước 4 (MEM).
 - Lệnh thứ hai đang chuyển sang bước 3 (EX).
 - Lệnh thứ ba ở được thực hiện ở bước 2 (ID) đón lệnh.
 - Lệnh thứ tư ở được thực hiện ở bước 1 (IF) đón lệnh.
- ◆ Ở chu kỳ xung CK thứ năm:
 - Lệnh thứ nhất đang chuyển sang bước 5 (WB).
 - Lệnh thứ hai đang chuyển sang bước 4 (MEM).
 - Lệnh thứ ba đang chuyển sang bước 3 (EX).
 - Lệnh thứ tư ở được thực hiện ở bước 2 (ID) đón lệnh.
 - Lệnh thứ năm ở được thực hiện ở bước 1 (IF) đón lệnh.
- ◆ Ở chu kỳ xung CK thứ sáu:
 - Lệnh thứ nhất đã hoàn tất.
 - Lệnh thứ hai đang chuyển sang bước 5 (WB).
 - Lệnh thứ ba đang chuyển sang bước 4 (MEM).
 - Lệnh thứ tư đang chuyển sang bước 3 (EX).
 - Lệnh thứ năm ở được thực hiện ở bước 2 (ID) đón lệnh.
 - Lệnh thứ sáu ở được thực hiện ở bước 1 (IF) đón lệnh.
- ◆ Ở chu kỳ xung CK thứ bảy:

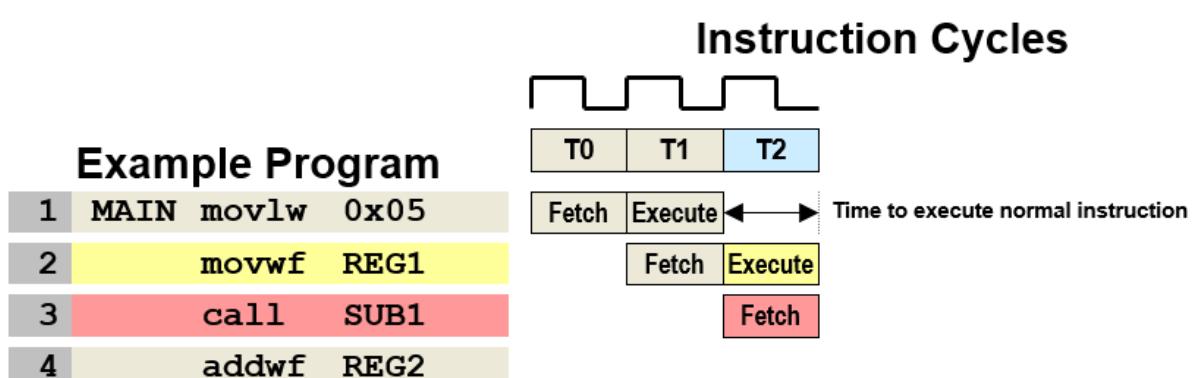
- Lệnh thứ hai đã hoàn tất.
- Lệnh thứ ba đang chuyển sang bước 5 (WB).
- Lệnh thứ tư đang chuyển sang bước 4 (MEM).
- Lệnh thứ năm đang chuyển sang bước 3 (EX).
- Lệnh thứ sáu ở được thực hiện ở bước 2 (ID) đón lệnh.
- Lệnh thứ bảy ở được thực hiện ở bước 1 (IF) đón lệnh.

Vậy từ xung CK thứ 5 trở đi thì mỗi 1 xung thì có 1 lệnh được thực hiện xong, khi đó ta có thể nói rằng mỗi lệnh bây giờ chỉ còn thực hiện với 1 chu kỳ xung CK duy nhất. Đó chính là ưu điểm lớn của cấu trúc pipeline. So sánh hiệu quả của cấu trúc pipeline mang lại về thời gian so với không dùng pipeline như hình 2-15.



Hình 2-15. Minh họa 2 cấu trúc để thấy hiệu quả về thời gian.

Ở vi điều khiển PIC thì quá trình thực hiện lệnh được rút gọn theo cấu trúc RISC (Reduce Instruction Set Computer) chỉ còn có 2 bước là đón lệnh và thực hiện lệnh. Khi áp dụng vào cấu trúc pipeline thì tổ chức có dạng như hình 2-16.



Hình 2-16. Cấu trúc pipeline của vi điều khiển PIC 16F887.

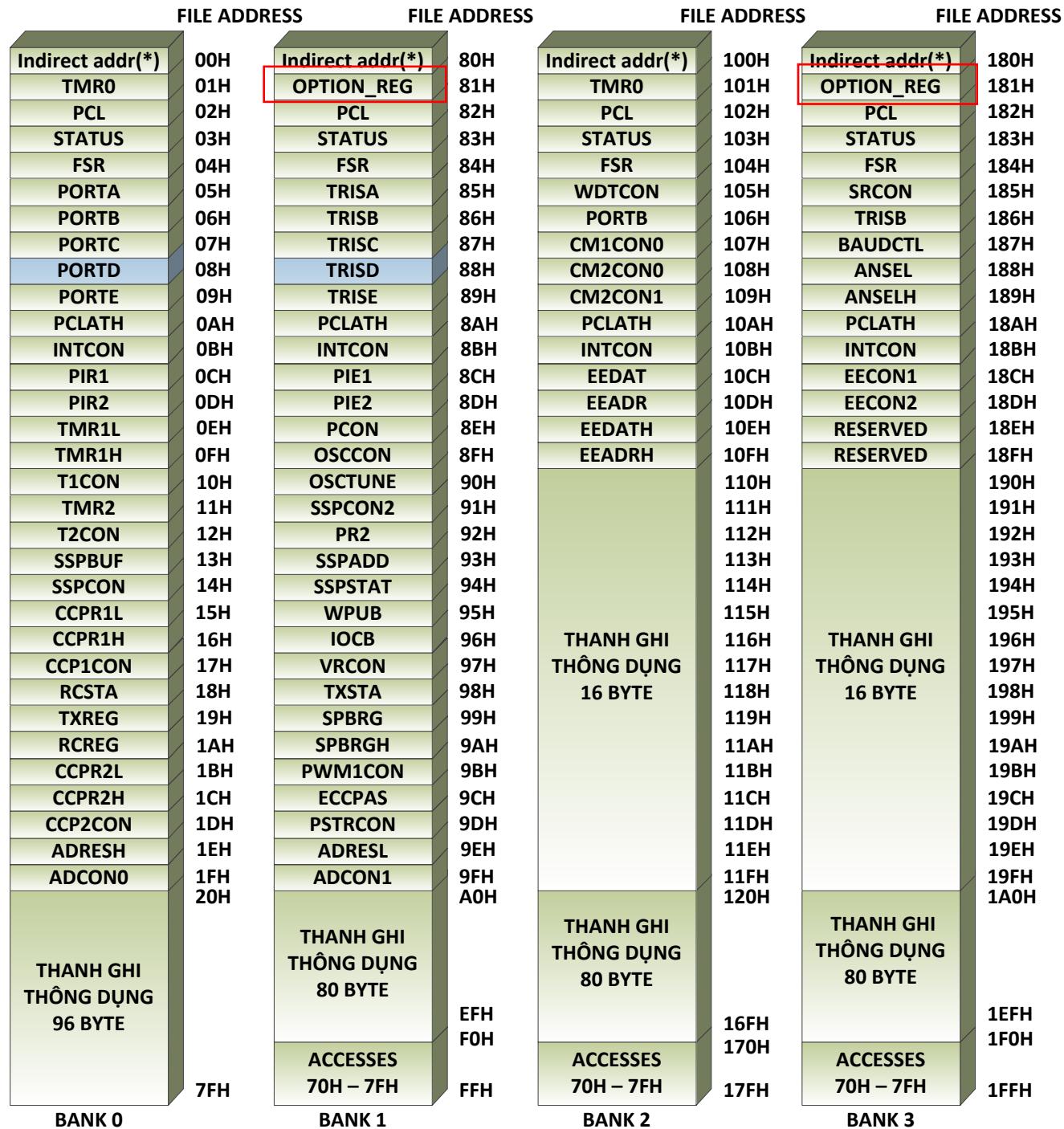
2.3.4 KHẢO SÁT BỘ NHỚ DỮ LIỆU VÀ THANH GHI TRẠNG THÁI

a. Cấu trúc bộ nhớ dữ liệu

Bộ nhớ dữ liệu được phân chia thành 4 Bank, mỗi bank có 128byte bao gồm một số thanh ghi chức năng đặc biệt, còn lại là các ô nhớ thông dụng có chức năng lưu trữ dữ liệu.

Bộ nhớ RAM có 512 ô nhớ và khi đó số bit địa chỉ được dùng là 9 bit ($2^9 = 512$), mỗi 1 ô nhớ có 1 địa chỉ duy nhất.

Toàn bộ các ô nhớ của bộ nhớ dữ liệu được gọi là File thanh ghi.



Hình 2-17. Tổ chức File thanh ghi.

Các thanh ghi có chức năng đặc biệt nằm ở vùng địa chỉ thấp, các ô nhớ còn lại không có gì đặc biệt nằm ở cùng địa chỉ bên trên các thanh ghi chức năng đặc biệt – xem như các ô nhớ

RAM dùng để lưu dữ liệu. Tất cả các bank thanh ghi đều chứa những thanh ghi đặc biệt - xem hình 2-17.

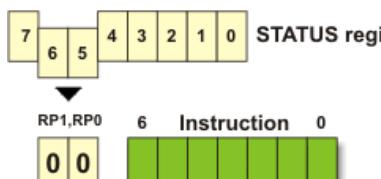
Theo hình 2-17 thì bộ nhớ dữ liệu được chia làm 4 bank thanh ghi, mỗi bank có 128, tổng cộng là 512 ô nhớ, nhưng do có 1 số thanh ghi có chức năng đặc biệt ở bank nào cũng có nên làm giảm số lượng. Ví dụ thanh ghi trạng thái (status) ở 4 bank đều có, thay vì 4 thanh ghi thì chỉ xem là 1, tương tự cho các thanh ghi khác. Số lượng thực nhỏ hơn 512 ô nhớ.

Do sử dụng cấu trúc pipeline nên tập lệnh phải rút gọn dần đến bộ nhớ chương trình được tổ chức theo trang để đơn giản khi truy xuất trong phạm vi 1 trang, tương tự bộ nhớ dữ liệu cũng được phân chia thành 4 bank để đáp ứng cho các yêu cầu trên.

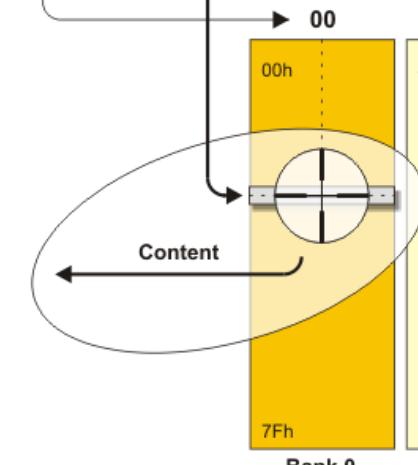
Bộ nhớ RAM 4 bank có 2 cách truy xuất bộ nhớ dữ liệu: truy xuất trực tiếp và truy xuất gián tiếp.

Khi truy xuất trực tiếp: thì các lệnh chỉ được phép truy xuất 1 ô nhớ RAM hay 1 thanh ghi của 1 bank. Địa chỉ 9 bit thì 2 bit địa chỉ cao thứ 7 và thứ 8 cố định, 7 bit chỉ thấp từ 6 đến 0 được phép thay đổi nên chỉ truy xuất trong phạm vi 128 byte tương ứng 1 bank, xem hình 2-18 phần truy xuất trực tiếp.

Direct addressing

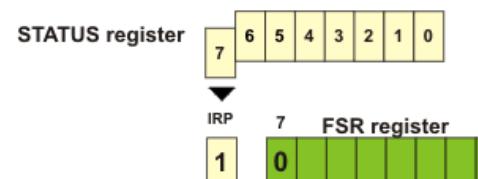


Bank Address

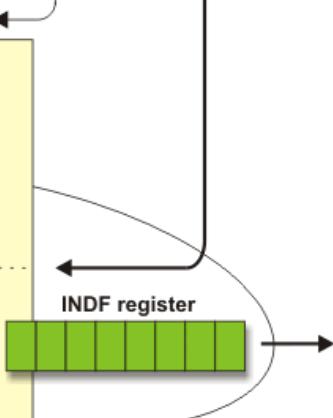


Bank 0 Content

Indirect addressing



Bank Address



Bank

Content

INDF register

Hình 2-18. Truy xuất trực tiếp và gián tiếp bộ nhớ RAM.

Hai bit địa chỉ cao chính lưu ở 2 bit có tên là RP1 và RP0 nằm trong thanh ghi trạng thái. Nếu muốn truy xuất các ô nhớ hay thanh ghi nằm ở bank khác thì phải đổi bank bằng cách thay đổi địa chỉ của 2 bit này. Cách truy xuất này giống như trang bộ nhớ chương trình.

Khi truy xuất gián tiếp: thì các lệnh truy xuất được phép truy xuất 1 ô nhớ RAM hay 1 thanh ghi nằm trong 2 bank: bank 0, 1 hoặc bank 2, 3. Địa chỉ 9 bit thì bit địa chỉ thứ 8 cố định, 8 bit thấp được phép thay đổi nên truy xuất trong phạm vi 256 byte tương ứng 2 bank, xem hình 2-18 phần truy xuất gián tiếp.

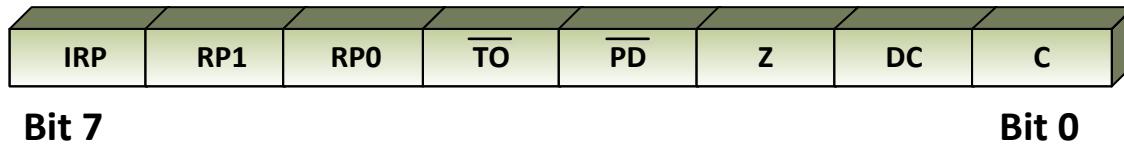
Bit địa chỉ thứ 8 chính lưu ở bit IRP nằm trong thanh ghi trạng thái.

Cách truy xuất trực tiếp thường dùng để truy xuất một vài ô nhớ, cách truy xuất gián tiếp dùng để truy xuất một vùng nhớ liên tục để phục vụ cho cách viết chương trình ở dạng vòng lặp.

Các thanh ghi đã có chức năng không được trình bày trong chương này, khi khảo sát các thành phần ngoại vi thì khi đó sẽ khảo sát các thanh ghi có liên quan một cách chi tiết.

b. *Thanh ghi trạng thái – STATUS REGISTER ĐỊA CHỈ 03H, 83H, 103H, 83H*

Thanh ghi trạng thái chứa trạng thái của khối ALU, trạng thái Reset và các bit chọn bank bộ nhớ dữ liệu. Cấu trúc thanh ghi trạng thái như hình 2-19.



Hình 2-19. Thanh ghi trạng thái.

Chức năng của các bit trong thanh ghi trạng thái:

Bit 7	IRP: bit lựa chọn thanh ghi (dùng địa chỉ gián tiếp). 1 = bank 2, 3 (100h-1FFh) 0 = bank 0, 1 (000h- 0FFh)
Bit 6-5	RP1:RP0: các bit lựa chọn thanh ghi (dùng địa chỉ trực tiếp) 11 = bank 3 (180h-1FFh) 10 = bank 2 (100h- 17Fh) 01 = bank 1 (80h- FFh) 00 = bank 0 (00h- 7Fh)
Bit 4	\overline{TO}: Time-out bit (Bit thời gian chờ) 1 = sau khi mở nguồn, lệnh CLRWDT hoặc SLEEP 0 = thời gian chờ của WDT được thực hiện
Bit 3	\overline{PD}: Power-down bit (bit tắt nguồn) 1= sau khi mở nguồn hoặc bằng lệnh CLRWDT 0= thực thi lệnh SLEEP
Bit 2	Z: Zero bit (bit 0) 1 = khi kết quả bằng 0. 0 = khi kết quả khác 0.
Bit 1	DC: Digit carry/ \overline{borrow} bit (các lệnh ADDWF, ADDLW, SUBLW, SUBWF) (bit tràn / mượn) 1 = khi cộng 4 bit thấp bị tràn. 0 = khi cộng 4 bit thấp không bị tràn.
Bit 0	C: Carry/ \overline{borrow} bit (các lệnh ADDWF, ADDLW, SUBLW, SUBWF) 1 = khi kết quả phép toán có tràn. 0 = khi kết quả phép toán không bị tràn.

Chú ý: Nếu phép toán trừ thì trạng thái của cờ C như sau: nếu phép trừ lớn hơn 0 thì cờ C bằng 0, nếu kết quả trừ nhỏ hơn 0 thì cờ C bằng 1.

Các thanh ghi còn lại sẽ được khảo sát ở các phần có liên quan.

2.3.5 BỘ NHỚ DỮ LIỆU EEPROM

Bộ nhớ dữ liệu Eeprom có dung lượng 256 byte dùng để lưu dữ liệu quan trọng khi mất điện thì dữ liệu này vẫn còn. Cách thức ghi dữ liệu vào bộ nhớ Eeprom sẽ được trình bày ở phần bộ nhớ Eeprom.

2.3.6 TÓM TẮT

Vi điều khiển PIC đang khảo sát có 4 loại bộ nhớ được tích hợp bên trong gồm:

Bộ nhớ chương trình có dung lượng 8k word dùng để lưu chương trình và thanh ghi chưa địa chỉ để quản lý bộ nhớ chương trình là PC. Chương trình điều khiển sau khi viết xong và được biên dịch thành công sẽ được nạp vào vùng nhớ này để vi điều khiển thực thi.

Bộ nhớ ngắn xếp có 8 cấp hay 8 ô nhớ, chỉ dùng để lưu địa chỉ trả về trong thanh ghi PC khi thực hiện lệnh gọi chương trình con và ngắt. Người dùng không thể dùng vùng nhớ này để lưu dữ liệu.

Bộ nhớ RAM xem như có 512 byte được chia làm 4 bank, bao gồm các thanh ghi đã có chức năng và chưa có chức năng là các ô nhớ dùng để lưu dữ liệu phục vụ cho việc viết chương trình điều khiển.

Bộ nhớ dữ liệu Eeprom có dung lượng 256 byte dùng để lưu dữ liệu quan trọng khi mất điện thì dữ liệu này vẫn còn.

2.4 CÂU HỎI ÔN TẬP – TRẮC NGHIỆM - BÀI TẬP

2.4.1 CÂU HỎI ÔN TẬP

Câu số 2-1: Hãy cho biết các loại bộ nhớ mà vi điều khiển PIC16F887 tích hợp.

Câu số 2-2: Hãy trình bày cấu trúc bộ nhớ RAM nội của vi điều khiển PIC16F887.

Câu số 2-3: Hãy cho biết các thanh ghi nào mà các bank đều có của vi điều khiển PIC16F887.

Câu số 2-4: Hãy cho biết tổ chức bộ nhớ chương trình của vi điều khiển PIC16F887.

2.4.2 CÂU HỎI MỞ RỘNG

Câu số 2-5: Hãy tìm hiểu tổ chức bộ nhớ vi điều khiển PIC18F4550 và so sánh với PIC16F887.

Câu số 2-6: Hãy tìm hiểu tổ chức bộ nhớ vi điều khiển PIC18F4620 và so sánh với PIC16F887.

2.4.3 CÂU HỎI TRẮC NGHIỆM

Câu 2-1: Bộ nhớ chương trình của PIC 16F887 có dung lượng là:

- | | |
|--------------|--------------|
| (a) 8K×Byte | (b) 8K×14bit |
| (c) 8K×16bit | (d) 368Byte |

Câu 2-2: Bộ nhớ dữ liệu của PIC 16F887 có dung lượng là:

- | | |
|-------------|--------------|
| (a) 256Byte | (b) 8K×14bit |
|-------------|--------------|

- (c) 8K×16bit (d) 368Byte

Câu 2-3: Bộ nhớ dữ liệu EEPROM của PIC 16F887 có dung lượng là:

Câu 2-4: Bộ nhớ ngăn xếp của PIC 16F887 có dung lượng là:

Câu 2-5: Bộ nhớ chương trình của PIC 16F887 chia làm:

- (a) 2 trang (b) 4 trang
(c) 2 bank (d) 4 bank

Câu 2-6: Bộ nhớ dữ liệu của PIC 16F887 chia làm:

- (a) 2 trang (b) 4 trang
(c) 2 bank (d) 4 bank

Câu 2-7: Mỗi trang bộ nhớ chương trình của PIC 16F887 có dung lượng:

Câu 2-8: Mỗi bank bộ nhớ dữ liệu của PIC 16F887 có dung lượng:

Câu 2-9: Địa chỉ của ô nhớ 1234H thuộc trang bộ nhớ nào?

Câu 2-10: Thanh ghi PC của PIC 16F887 có chiều dài:

Câu 2-11: Phân chia bộ nhớ theo trang có ưu điểm:

- (a) Làm vi điều khiển chạy nhanh
(c) Làm giảm địa chỉ bộ nhớ

(b) Làm tăng số lượng mã code
(d) Làm giảm số lượng mã code

Câu 2-12: Các chương trình con lồng vào nhau của PIC phụ thuộc vào dung lượng:

- (a) Bộ nhớ chương trình (b) Bộ nhớ dữ liệu (c) Bộ nhớ ngắn xέp (d) Bộ nhớ EEPROM

Câu 2-13: Truy xuất trực tiếp bộ nhớ dữ liệu của PIC 16F887 thì:

Câu 2-14: Truy xuất gián tiếp bộ nhớ dữ liệu của PIC 16F887 thì:

- (a) Cho phép tùy ý cả 4 bank
(c) Chỉ cho phép 1 bank

(b) Cho phép 2 bank
(d) Cho phép 3 bank

Câu 2-15: Bit cho phép thay đổi các bank trong truy xuất trực tiếp bộ nhớ dữ liệu của PIC 16F887 là:

Câu 2-16: Bit cho phép thay đổi các bank trong truy xuất gián tiếp bộ nhớ dữ liệu của PIC 16F887 là:

- | | |
|----------------|-------------------|
| (a) IRP1, IRP2 | (b) RP1, RP0 |
| (c) IRP | (d) IRP, RP1, RP0 |

Câu 2-17: Thanh ghi nào đều có trong 4 bank bộ nhớ dữ liệu của PIC 16F887:

- | | |
|-----------|------------|
| (a) PORTB | (b) TRISB |
| (c) PORTA | (d) STATUS |

Câu 2-18: Thanh ghi nào đều có trong 4 bank bộ nhớ dữ liệu của PIC 16F887:

- | | |
|-----------|----------|
| (a) TRISB | (b) PCL |
| (c) TRISA | (d) TMR0 |

Câu 2-19: Khi ngắt xảy ra thì PIC 16F887 sẽ thực hiện chương trình con phục vụ ngắt tại địa chỉ:

- | | |
|-----------|-----------|
| (a) 0004H | (b) 0014H |
| (c) 0000H | (d) 0024H |

Câu 2-20: Địa chỉ bộ nhớ chương trình của PIC 16F887:

- | | |
|------------------------|------------------------|
| (a) Từ 0000H đến 1FFFH | (b) Từ 0800H đến 0FFFH |
| (c) Từ 0000H đến 07FFH | (d) Từ 0000H đến 2FFFH |

2.4.4 BÀI TẬP

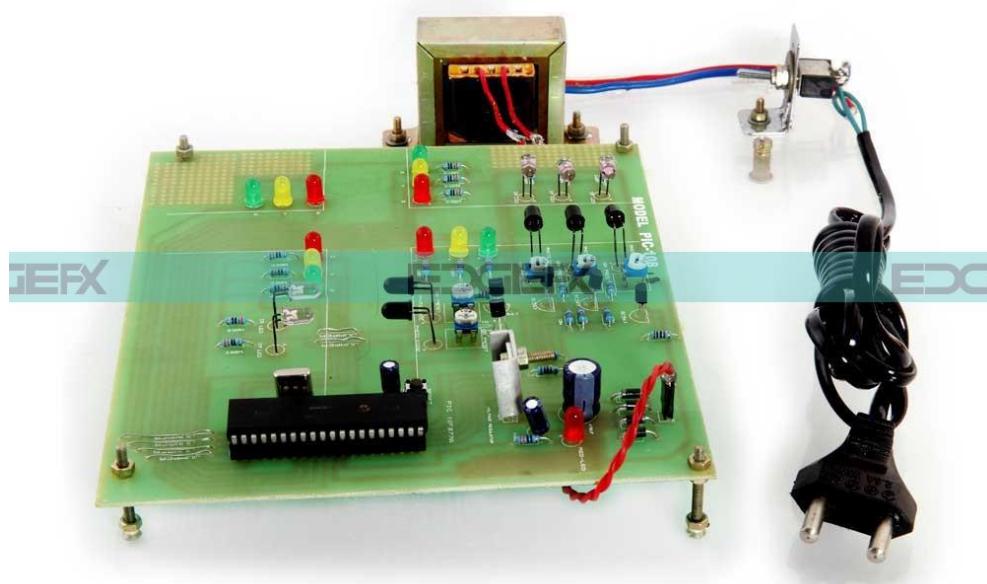
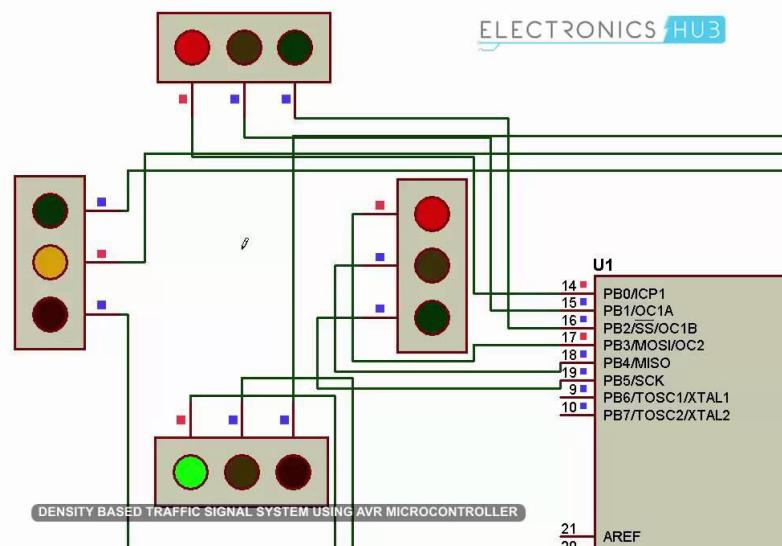
VỊ ĐIỀU KHIỂN PIC16F887: LỆNH HỢP NGỮ

3.1 GIỚI THIỆU

Ở chương này khảo sát tập lệnh hợp ngữ của các vi điều khiển. Sau khi kết thúc chương này bạn sẽ biết mã lệnh nhị phân, lệnh gọi nhớ, các kiểu định địa chỉ bộ nhớ của vi điều khiển, biết tập lệnh hợp ngữ của vi điều khiển

3.2 NGÔN NGỮ LẬP TRÌNH HỢP NGỮ

Vi điều khiển hay vi xử lý là các IC lập trình, khi bạn đã thiết kế hệ thống điều khiển có sử dụng vi xử lý hay vi điều khiển ví dụ như hệ thống điều khiển đèn giao thông cho một ngã tư gồm có các đèn Xanh, Vàng, Đỏ như hình 3-1 và có thể có thêm các led 7 đoạn để hiển thị thời gian thì đó mới chỉ là phần cứng, muốn hệ thống vận hành thì bạn phải viết một chương trình điều khiển nạp vào bộ nhớ nội bộ trong vi điều khiển hoặc bộ nhớ bên ngoài và gắn vào trong hệ thống để hệ thống vận hành và dĩ nhiên bạn phải viết đúng thì hệ thống mới vận hành đúng. Chương trình gọi là phần mềm.



Hình 3-1. Hệ thống điều khiển đèn giao thông – ảnh minh họa.

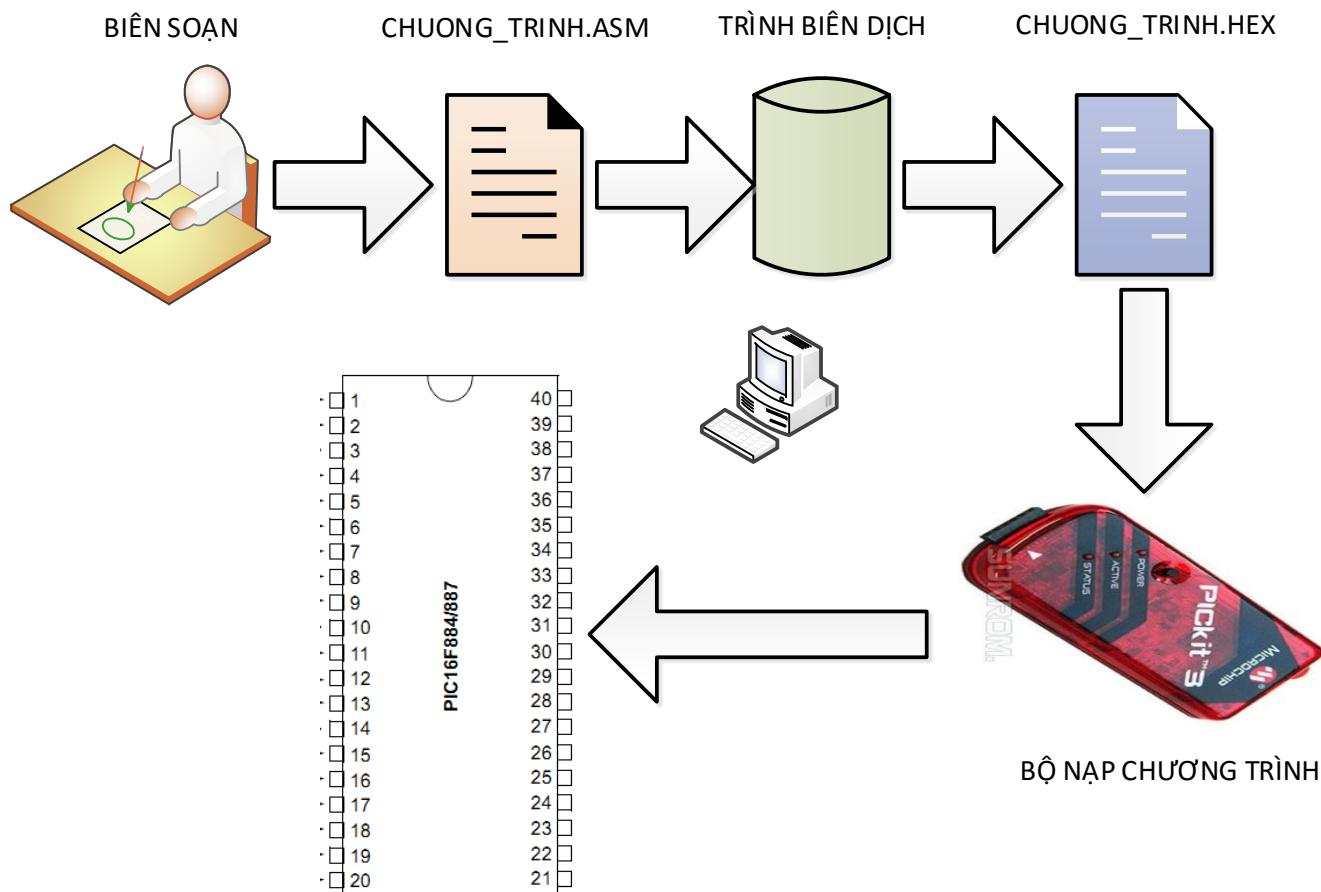
Phần mềm và phần cứng có quan hệ với nhau, người lập trình phải hiểu rõ hoạt động của phần cứng để viết chương trình.

Bất kỳ vi xử lý hay vi điều khiển nào cũng đều có tập lệnh là hợp ngữ còn gọi là ngôn ngữ Assembly, và đó là ngôn ngữ cơ bản, đơn giản. Các vi điều khiển khác họ với nhau thì các lệnh hợp ngữ thường khác nhau.

Khi nghiên cứu bất kỳ một vi xử lý hay một vi điều khiển nào đó thì người ta thường dùng hợp ngữ để lập trình cho các ứng dụng để điều khiển. Tuy nhiên chương trình viết bằng hợp ngữ mất nhiều thời gian, chỉ phù hợp cho các ứng dụng điều khiển đơn giản, nếu muốn điều khiển các ứng dụng có nhiều phép toán phức tạp thì không nên dùng hợp ngữ, khi đó bạn nên dùng ngôn ngữ cấp cao hơn như C.

Giáo trình này không trình bày nhiều và sâu về hợp ngữ nhưng có giới thiệu cơ bản về hợp ngữ cho các bạn biết để giúp bạn hiểu rõ hơn về tổ chức hoạt động bên trong các khối của vi điều khiển. Trong chương này trình bày về hợp ngữ, chương tiếp theo sẽ trình bày về ngôn ngữ lập trình cấp cao như C.

Người lập trình dùng máy tính để tiến hành biên soạn chương trình dạng assembly với tên mở rộng thường phổ biến là “asm”, sau khi biên soạn xong sẽ nhờ trình biên dịch sang file hex với phần mở rộng là “hex” và cuối cùng là dùng bộ nạp để nạp code vào vi điều khiển. Trình tự được thể hiện như hình 3-2.



Hình 3-2. Trình tự biên soạn chương trình Assembly cho đến khi nạp code.

3.2.1 PHẦN MỀM LẬP TRÌNH HỢP NGỮ

Có nhiều cách để biên soạn chương trình bằng hợp ngữ cho vi điều khiển nói chung và vi điều khiển PIC nói riêng, khi bạn dùng vi điều khiển nào thì đều có phần mềm hỗ trợ để biên soạn và kèm theo là trình biên dịch luôn.

Phần mềm khá phổ biến viết hợp ngữ cho PIC là MPLAB có logo như hình 3-3.



Hình 3-3. Logo phần mềm MPLAB.

MPLAB là phần mềm dùng để lập trình cho vi điều khiển PIC bằng hợp ngữ và sau này có hỗ trợ lập trình bằng HI-TECH C. Bạn có thể tải phần mềm này về cài đặt nghiên cứu và sử dụng.

3.2.2 CHƯƠNG TRÌNH HỢP NGỮ CƠ BẢN

Sau khi đã cài đặt phần mềm thì bạn có thể sử dụng để viết chương trình dùng các lệnh hợp ngữ.

Trước khi tiến hành khảo sát chi tiết cấu trúc của một chương trình hợp ngữ thì ta cần biết một số khai niệm sau:

Chương trình là một tập hợp các lệnh được tổ chức theo một trình tự hợp lý để giải quyết đúng các yêu cầu của người lập trình.

Người lập trình là người biết giải thuật để viết chương trình và sắp xếp đúng các lệnh theo giải thuật. Người lập trình phải biết chức năng của tất cả các lệnh của vi điều khiển để viết chương trình.

Tất cả các lệnh có thể có của một ngôn ngữ lập trình còn gọi là **tập lệnh**.

Lệnh của vi điều khiển là một số nhị phân 8 bit, còn gọi là mã máy hay mã đối tượng.

Nếu một vi điều khiển 8 bit ví dụ như AT80C51 có mã lệnh 8 bit thì có 256 tổ hợp khác nhau từ 0000 0000b đến 11111111b tương ứng và khi đó có thể xây dựng được 256 lệnh khác nhau.

Vì xử lý 16 bit như 8086 của Intel có mã lệnh 16 bit thì có 65536 tổ hợp khác nhau từ 0000H đến FFFFH tương ứng và khi đó có thể xây dựng được 16^2 lệnh khác nhau nhưng thường thì sử dụng ít hơn rất nhiều.

Do mã lệnh dạng số nhị phân quá dài và khó nhớ nên các nhà lập trình đã xây dựng một ngôn ngữ lập trình hợp ngữ còn gọi là Assembly cho dễ nhớ, điều này giúp cho việc lập trình được thực hiện một cách dễ dàng và nhanh chóng cũng như đọc hiểu và gõ rối chương trình.

Cấu trúc của một chương trình viết bằng hợp ngữ như hình 3-4.

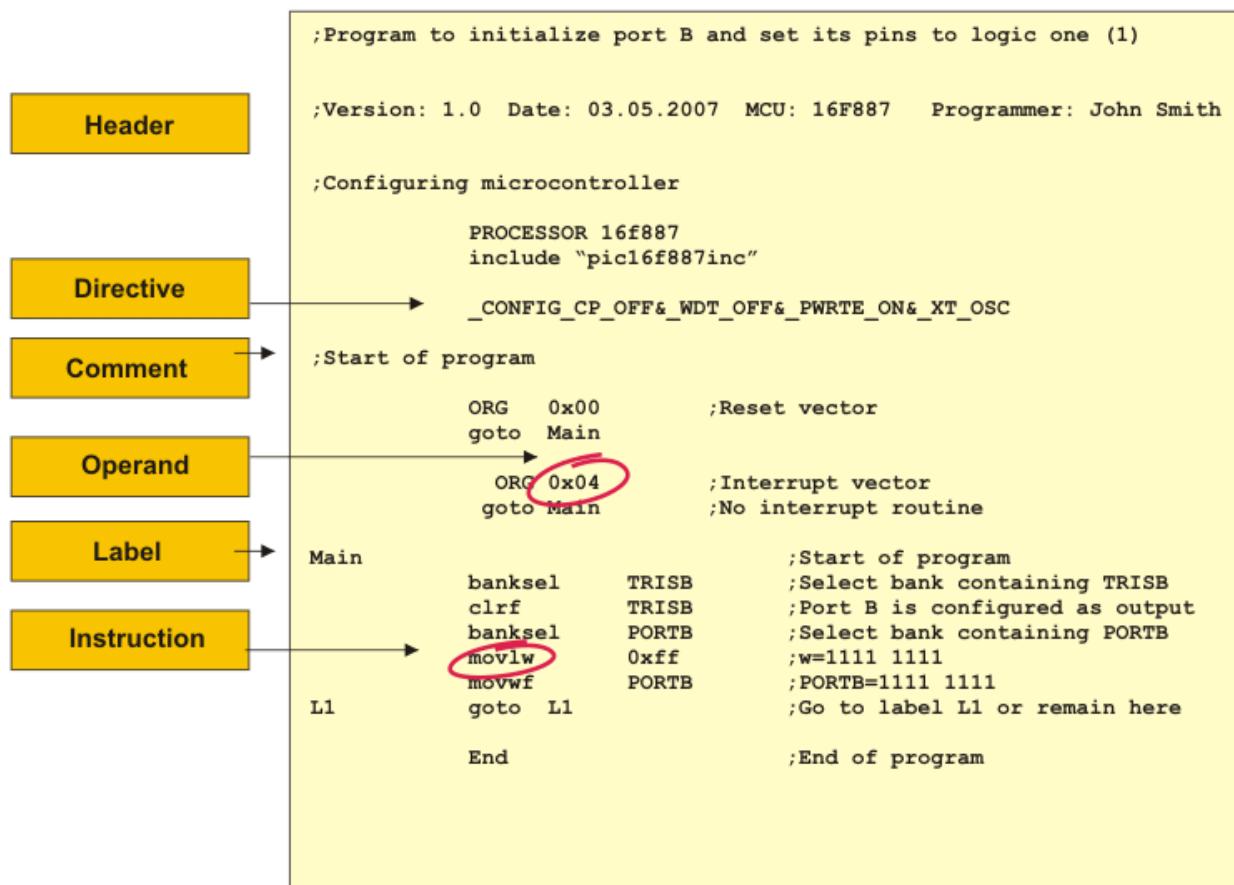
Một chương trình viết bằng hợp ngữ gồm có các thành phần như sau:

Tiêu đề (Header) của chương trình để mô tả chương trình viết điều khiển cái gì để nhanh chóng đọc hiểu trở lại cho sau này.

Chỉ dẫn (Directive) của chương trình để báo cho phần mềm kiểm tra lỗi và biên dịch biết để lấy thông tin khi biên dịch, chỉ dẫn có rất nhiều dạng, khai báo thư viện, định nghĩa biến, ...

Chú thích (Comment) của chương trình dùng để giải thích rõ hơn cho một lệnh hoặc một địa chỉ hoặc một biến nào đó.

Toán hạng hay tác tố (Operand) là các đối tượng chứa dữ liệu để lệnh thực thi, ví dụ lệnh cộng thì phải có 2 toán hạng để thực hiện phép toán.



Hình 3-4. Chương trình dùng hợp ngữ.

Nhãn (Label) thay cho các địa chỉ nơi nhảy đến trong chương trình khi thực hiện các lệnh vòng lặp hoặc khi gọi chương trình con. Thay vì phải đi tìm địa chỉ để nhảy đến thì ta có thể dùng nhãn để nhảy đến và trình biên dịch sẽ đi tìm địa chỉ và thay thế nhãn bằng địa chỉ cho chúng ta. Công việc này phức tạp ta giao cho trình biên dịch làm.

Cuối cùng là các lệnh hợp ngữ (Instruction) dùng để viết chương trình điều khiển.

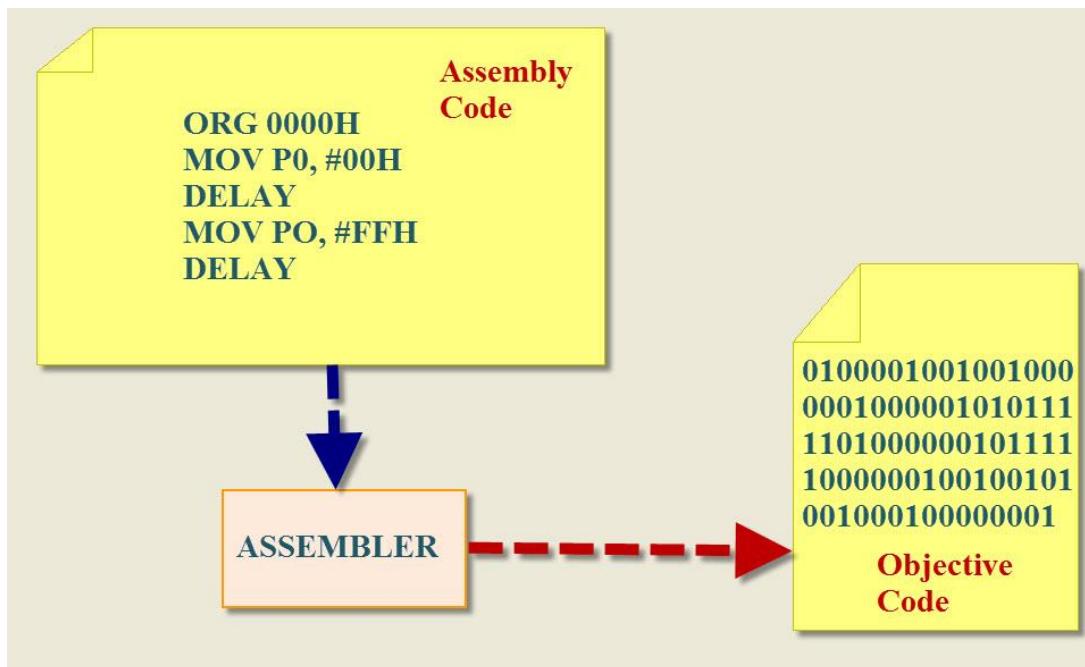
3.2.3 TRÌNH BIÊN DỊCH CHO HỢP NGỮ

Sau khi viết xong một chương trình dùng các lệnh hợp ngữ thì phải tiến hành biên dịch các lệnh hợp ngữ thành mã máy là các số nhị phân để nạp vào bộ nhớ của vi điều khiển. Trình biên dịch tiếng Anh gọi là Assembler.

Từng chỉ dẫn, từng lệnh, từng nhãn, ... được chuyển đổi thành mã nhị phân được thể hiện như hình 3-5.

Chương trình hợp ngữ với tên mở rộng là “asm”, sau khi biên dịch thì được nhiều file cùng tên với file gốc như phần mở rộng khác nhau bao gồm 3 file cơ bản với phần mở rộng là “lst”, “hex” và “map”.

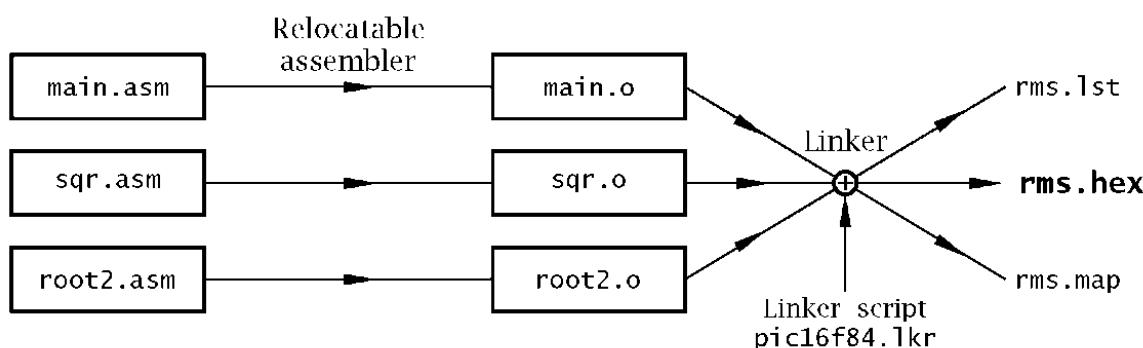
Trong hình 3-5 ta thấy file hex chỉ còn chứa các số hex, nói chính xác là thập lục phân, nói ngắn gọn là hex. Khi mở file này để xem thì thường các mã hex tổ chức ở dạng số hex 8 bit với vi điều khiển có từ mã lệnh 8 bit, 16 bit với vi điều khiển có từ mã lệnh 16 bit.



Hình 3-5. Biên dịch chương trình hợp ngữ.

Cách tổ chức của file hex là giống nhau thường theo một chuẩn quy định chung, bạn có thể tìm hiểu thêm cấu trúc của file này để biết rõ hơn.

Khi bạn viết một ứng dụng điều khiển lớn và phức tạp thì bạn nên chia chương trình thành nhiều file khác nhau và khi biên dịch thì trình biên dịch sẽ biên dịch và liên kết các file đó lại với nhau để tạo ra một file hex duy nhất được thể hiện qua hình 3-6.



Hình 3-6. Biên dịch liên kết nhiều chương trình hợp ngữ.

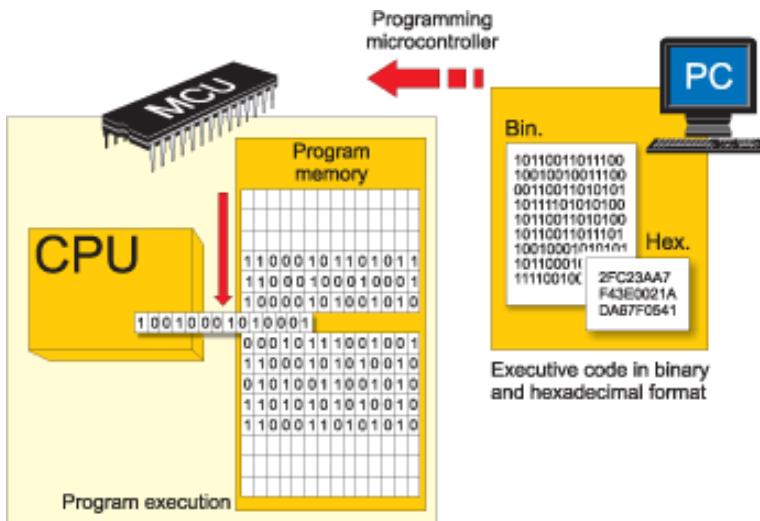
Ngoài file hex dùng để nạp vào bộ nhớ của vi điều khiển để thực thi thì còn một dạng file thứ hai là file “bin” ở định dạng mã nhị phân như hình 3-7.

Sự khác nhau giữa file “hex” và file “bin” bởi nhiều thành phần nhưng khác nhau cơ bản là file hex chỉ biên dịch đúng các lệnh và vùng nhớ được sử dụng trong chương trình nguồn asm, không chứa mã hex của các vùng nhớ không dùng, khi nạp vào bộ nhớ của vi điều khiển sẽ nhanh.

Còn file “bin” thì biên dịch toàn bộ dung lượng bộ nhớ của vi điều khiển, ví dụ vi điều khiển có 8kword thì file bin sẽ có đầy đủ 8kword cho dù chương trình chỉ sử dụng vài trăm

word và phần lớn còn lại chưa dùng đến. Điều này dẫn đến khi nạp code vào bộ nhớ mất nhiều thời gian hơn so với file hex.

Chính vì thế file hex thường được sử dụng rất nhiều đến nỗi người dùng không biết có sự tồn tại của file bin.



Hình 3-7. Biên dịch chương trình hợp ngữ tạo file hex và file bin.

Trước đây các vi xử lý của Intel thường dùng tên file chứa số nhị phân là “com”.

3.3 LỆNH HỢP NGỮ CỦA VI ĐIỀU KHIỂN PIC 16F887

3.3.1 GIỚI THIỆU

Phần trên đã trình bày về ngôn ngữ lập trình Assembly do con người tạo ra, khi sử dụng ngôn ngữ Assembly để viết thì người lập trình vi điều khiển phải học hết tất cả các lệnh và viết đúng theo qui ước về cú pháp, trình tự sắp xếp dữ liệu để chương trình biên dịch có thể biên dịch đúng.

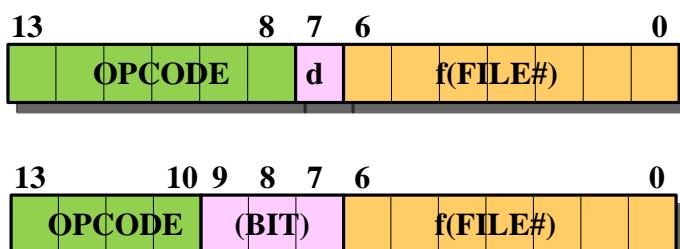
Ở phần này sẽ trình bày chi tiết về tập lệnh của vi điều khiển giúp bạn hiểu rõ từng lệnh để bạn có thể lập trình được.

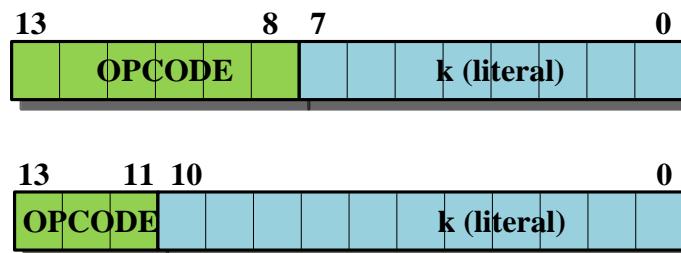
Tập lệnh của PIC 16 được chia ra làm 3 nhóm lệnh:

- Lệnh xử lý bit
- Lệnh xử lý byte
- Lệnh xử lý hàng số và điều khiển

Mỗi lệnh của PIC là một từ dữ liệu 14 bit được chia ra làm 2 nhóm gồm mã lệnh hoạt động (opcode: operation code) và tác tố (operand).

Mã lệnh Opcode cho biết loại lệnh mà CPU phải thực hiện. Các dạng mã lệnh như hình 3-8:





Hình 3-8. Các dạng mã lệnh.

Tác tố operand là dữ liệu mà lệnh sẽ xử lý.

Bảng 3-1. Các tác tố.

TT	Tác tố	Mô tả
1	f	Register file address (0x00 to 0x7F): là địa chỉ 7 bit của file thanh ghi 8 bit, của 1 bank.
2	W	Working register: Thanh ghi làm việc hay thanh ghi A.
3	b	Bit address within an 8-bit file register: là địa chỉ của 1 bit nằm trong thanh ghi file 8 bit
4	k	Literal field, constant data or label: Là hằng số hoặc địa chỉ của nhãn.
5	d	Destination select: lựa chọn nơi lưu dữ liệu: d=0 thì lưu vào W, d=1 thì lưu vào f, mặc nhiên không ghi d trong lệnh thì tương ứng d=1.
6	PC	Program counter: bộ đếm chương trình.
7	TO	Time-out bit: bit báo thời gian đã hết.
8	PD	Power -down bit: bit báo CPU đang làm việc ở chế độ ngủ.

Với lệnh xử lý byte: thì 'f' đại diện cho file thanh ghi và 'd' đại diện cho hướng lưu dữ liệu: nếu 'd' bằng 0 thì dữ liệu sau khi xử lý lưu vào thanh ghi 'W', nếu 'd' bằng 1 thì dữ liệu sau khi xử lý lưu vào thanh ghi 'f'.

Với lệnh xử lý bit: thì 'b' đại diện cho bit nằm trong file thanh ghi 'f'.

Với lệnh xử lý hằng số hoặc điều khiển: thì 'k' đại diện cho hằng số 8 bit hoặc địa chỉ 11 bit.

Một chu kỳ lệnh gồm 4 chu kỳ dao động, nếu sử dụng thạch anh có tần số 4MHz thì thời gian thực hiện mỗi lệnh là 1μs. Hầu hết các lệnh thực hiện mất 1 chu kỳ lệnh, ngoại trừ lệnh kiểm tra điều kiện đúng sai hoặc lệnh làm thay đổi giá trị của thanh ghi PC thì thực hiện mất 2 chu kỳ máy.

3.3.2 KHẢO SÁT TẬP LỆNH TÓM TẮT VI ĐIỀU KHIỂN PIC 16F887

Bảng 3-2. Tóm tắt tập lệnh của PIC.

Nhóm lệnh xử lý byte giữa thanh ghi W với f					
TT	Cú pháp	Chức năng	Chu kỳ	Mã lệnh	Cờ bị ảnh hưởng
1	ADDWF f,d	(W) cộng (f)	1	00 0111 dfff ffff	C, DC, Z
2	ANDWF f,d	(W) and (f)	1	00 0101 dfff ffff	Z
3	CLRF f	Xóa (f)	1	00 0001 1fff ffff	Z
4	CLRW -	Xóa (W)	1	00 0001 0xxx xxxx	Z
5	COMF f,d	Bù (f)	1	00 1001 dfff ffff	Z

6	DECF f,d	Giảm f	1	00 0011 dfff ffff	Z
7	DECEFSZF f,d	Giảm f, bỏ lệnh kế nếu f = 0	1(2)	00 1011 dfff ffff	
8	INC f,d	Tăng f	1	00 1010 dfff ffff	Z
9	INCEFSZF f,d	Tăng f, bỏ lệnh kế nếu f = 0	1(2)	00 1111 dfff ffff	
10	IORWF f,d	(W) or (f)	1	00 0100 dfff ffff	Z
11	MOVF f,d	copy (f)	1	00 1000 dfff ffff	Z
12	MOVWF f	(W) → (f)	1	00 0000 1fff ffff	
13	NOP	Không làm gì	1	00 0000 0xx0 0000	
14	RLF f,d	Xoay trái f xuyên qua cờ C	1	00 1101 dfff ffff	C
15	RRF f,d	Xoay phải f xuyên qua cờ C	1	00 1100 dfff ffff	C
16	SUB f,d	(F) trừ (f)	1	00 0010 dfff ffff	C, DC, Z
17	SWAPF f,d	Hoán chuyển 4 bit của f	1	00 1110 dfff ffff	
18	IORWF f,d	(W) xor (f)	1	00 0110 dfff ffff	Z

Nhóm lệnh xử lý bit

TT	Cú pháp	Chức năng	Chu kỳ	Mã lệnh	Cờ bị ảnh hưởng
1	BCF f,b	Làm bit b trong f xuống 0	1	01 00bb bfff ffff	
2	BSF f,b	Làm bit b trong f lên 1	1	01 01bb bfff ffff	
3	BTFSC f,b	Nếu bit b bằng 0 thì bỏ lệnh kế	1	01 10bb bfff ffff	
4	BTFSS f,b	Nếu bit b bằng 1 thì bỏ lệnh kế	1	01 11bb bfff ffff	

Nhóm lệnh hằng số và điều khiển

TT	Cú pháp	Chức năng	Chu kỳ	Mã lệnh	Cờ bị ảnh hưởng
1	ADDLW k	(W) cộng k → (W)	1	11 111x kkkk kkkk	C, DC, Z
2	ANDLW k	(W) and k → (W)	1	11 1001 kkkk kkkk	Z
3	CALL k	Gọi chương trình con	2	10 0kkk kkkk kkkk	
4	CLRWDT	Xóa bộ định thời	1	00 0000 0110 0100	\overline{TO} , \overline{PD}
5	GOTO k	Nhảy đến địa chỉ k	2	10 1kkk kkkk kkkk	
6	IORLW k	(W) or k	1	11 1000 kkkk kkkk	Z
7	MOVLW k	K → (W)	1	11 00xx kkkk kkkk	
8	RETFIE	Trở về từ ngắt	2	00 0000 0000 1000	
9	RETLW	Trở về từ chương trình con và nạp hằng số vào W	2	11 01xx kkkk kkkk	
10	RETURN	Trở về từ chương trình con	2	00 0000 0000 1000	

11	SLEEP	Cpu vào chế độ chờ	1	00 0000 0110 0011	$\overline{TO}, \overline{PD}$
12	SUBLW k	$K - (W) \rightarrow (W)$	1	11 110x kkkk kkkk	C, DC, Z
13	XORLW k	$(W) \text{ xor } k$	1	11 1010 kkkk kkkk	Z

3.3.3 TẬP LỆNH CHI TIẾT

1. Lệnh: **ADDLW** Cộng hằng số k vào W

- Cú pháp: ADDLW k
- Tác động: $0 \leq k \leq 255$
- Thực thi: $(W) + k \rightarrow (W)$. Cờ ảnh hưởng: C, DC, Z. Chu kỳ thực hiện: 1.
- Chức năng: cộng nội dung thanh ghi W với hằng số k 8 bit và kết quả lưu vào W.

2. Lệnh: **ADDFWF** Cộng W với f

- Cú pháp: ADDWF f,d
- Tác động: $0 \leq f \leq 127, d \in [0,1]$
- Thực thi: $(W) + (f) \rightarrow (\text{dest})$. Cờ ảnh hưởng: C, DC, Z. Chu kỳ thực hiện: 1.
- Chức năng: cộng nội dung thanh ghi W với thanh ghi f. Nếu d=0 thì lưu kết quả vào thanh ghi W, còn d=1 thì lưu vào thanh ghi f.

3. Lệnh: **ANDLW** And hằng số với W

- Cú pháp: ANDLW k
- Tác động: $0 \leq k \leq 255$
- Thực thi: $(W) \text{ AND } (k) \rightarrow (W)$. Cờ ảnh hưởng: Z. Chu kỳ thực hiện: 1.
- Chức năng: And nội dung thanh ghi W với hằng số k 8 bit, kết quả lưu vào thanh ghi W.

4. Lệnh: **ANDWF** And W với F

- Cú pháp: ANDWF f,d
- Tác động: $0 \leq f \leq 127, d \in [0,1]$
- Thực thi: $(W) \text{ AND } (f) \rightarrow (\text{dest})$. Cờ ảnh hưởng: Z. Chu kỳ thực hiện: 1.
- Chức năng: And thanh ghi W với thanh ghi f. Nếu d = 0 thì kết quả lưu vào thanh ghi W, nếu d=1 thì kết quả lưu vào thanh ghi f.

5. Lệnh: **BCF** xoá bit trong thanh ghi F - BIT CLEAR FILE

- Cú pháp: BCF f,b
- Tác động: $0 \leq f \leq 127, 0 \leq b < 7$
- Thực thi: $0 \rightarrow (f)$. Cờ ảnh hưởng: không. Chu kỳ thực hiện: 1.
- Chức năng: xóa bit b trong thanh ghi f.

6. Lệnh: **BSF** set bit trong thanh ghi F - BIT SET FILE

- Cú pháp: BSF f,b
- Tác động: $0 \leq f \leq 127, 0 \leq b < 7$
- Thực thi: $1 \rightarrow (f)$. Cờ ảnh hưởng: không. Chu kỳ thực hiện: 1.
- Chức năng: set bit b trong thanh ghi f lên 1.

7. Lệnh: **BTFS** kiểm tra 1 bit trong thanh ghi F và nhảy nếu bằng 1

- Cú pháp: BTFSS f,b - BIT TEST FILE SKIP IF SET
- Tác tố: $0 \leq f \leq 127, 0 \leq b < 7$
- Thực thi: nhảy nếu $f=1$. Cờ ảnh hưởng: không. Chu kỳ thực hiện: 1(2).
- Chức năng: kiểm tra bit b: nếu bit b bằng 1 thì lệnh kế bị bỏ qua và thay bằng lệnh NOP, nếu bit b bằng 0 thì thực thi lệnh kế.

8. Lệnh: **BTFS** kiểm tra 1 bit trong thanh ghi F và nhảy nếu bằng 0

- Cú pháp: BTFSC f,b - BIT TEST FILE SKIP IF CLEAR
- Tác tố: $0 \leq f \leq 127, 0 \leq b < 7$
- Thực thi: nhảy nếu $f=0$. Cờ ảnh hưởng: không. Chu kỳ thực hiện: 1(2).
- Chức năng: kiểm tra bit b: nếu bit b bằng 0 thì lệnh kế bị bỏ qua và thay bằng lệnh NOP, nếu bit b bằng 1 thì thực thi lệnh kế.

9. Lệnh: **CALL** gọi chương trình con

- Cú pháp: CALL k
- Tác tố: $0 \leq k \leq 2047$
- Thực thi: $(PC + 1) \rightarrow TOS; k \rightarrow PC$ - TOP OF STACK
- Cờ ảnh hưởng: không. Chu kỳ thực hiện: 2.
- Chức năng: gọi chương trình con. Địa chỉ trả về $(PC+1)$ được cất vào ngăn xếp. Thanh ghi PC được nạp địa chỉ của chương trình con và chương trình con được thực hiện.

10. Lệnh: **CLRF** xoá thanh ghi f

- Cú pháp: CLRF f - CLEAR FILE
- Tác tố: $0 \leq f \leq 127$
- Thực thi: $00h \rightarrow (f); 1 \rightarrow Z$. Trạng thái ảnh hưởng: Z. Chu kỳ thực hiện: 1.
- Chức năng: xoá thanh ghi f và cờ Z bằng 1.

11. Lệnh: **CLRW** xoá thanh ghi W

- Cú pháp: CLRW - CLEAR WORKING
- Tác tố: không
- Thực thi: $00h \rightarrow (W), 1 \rightarrow Z$. Cờ ảnh hưởng: Z. Chu kỳ thực hiện: 1.
- Chức năng: xoá thanh ghi W và cờ Z bằng 1.

12. Lệnh: **CLRWDT** xoá WDT

- Cú pháp: CLRWDT - CLEAR WATCHDOG TIMER
- Tác tố: không
- Thực thi: $00 \rightarrow WDT; 0 \rightarrow$ Bộ đếm chia trước của WDT; $1 \rightarrow \overline{TO}; 1 \rightarrow \overline{PD}$
- Cờ ảnh hưởng: $\overline{TO}, \overline{PD}$. Chu kỳ thực hiện: 1.
- Chức năng: lệnh CLRWDT sẽ xoá bộ định thời WDT và bộ đếm chia trước của WDT. Các bit $\overline{PD}, \overline{TO}$ về mức 1.

13. Lệnh: **COMF** bù thanh ghi f

- Cú pháp: COMF f,d
- Tác tố: $0 \leq f \leq 127, d \in [0,1]$
- Thực thi: $(\overline{f}) \rightarrow (dest)$. Cờ ảnh hưởng: Z. Chu kỳ thực hiện: 1.
- Chức năng: bù 1 nội dung thanh ghi f. Nếu d=0 thì kết quả lưu vào thanh ghi W. Nếu d=1 thì kết quả lưu vào thanh ghi f.

14. Lệnh: **DECF** giảm nội dung thanh ghi f

- Cú pháp: DECF f,d
- Tác tố: $0 \leq f \leq 127$, $d \in [0,1]$
- Thực thi: $(f) - 1 \rightarrow (\text{dest})$. Cờ ảnh hưởng: Z. Chu kỳ thực hiện: 1.
- Chức năng: giảm nội dung thanh ghi f đi 1. Nếu $d = 0$ thì kết quả lưu vào thanh ghi W. Nếu $d = 1$ thì kết quả lưu vào thanh ghi f.

15. Lệnh: **DECFSZ** giảm nội dung thanh ghi f và nhảy nếu bằng 0

- Cú pháp: DECFSZ f,d
- Tác tố: $0 \leq f \leq 127$, $d \in [0,1]$
- Thực thi: $(f) - 1 \rightarrow (\text{dest})$; Cờ ảnh hưởng: không. Chu kỳ thực hiện: 1(2).
- Chức năng: giảm nội dung thanh ghi f đi 1. Nếu $d = 0$ thì kết quả lưu vào thanh ghi W. Nếu $d = 1$ thì kết quả lưu vào thanh ghi f. Nếu kết quả bằng 0 thì bỏ qua lệnh kế và thay bằng lệnh NOP.

16. Lệnh: **GOTO** lệnh rẽ nhánh không điều kiện

- Cú pháp: GOTO k
- Tác tố: $0 \leq k \leq 2047$
- Thực thi: $k \rightarrow \text{PC}$. Cờ ảnh hưởng: không. Chu kỳ thực hiện: 2.
- Chức năng: nhảy không điều kiện đến địa chỉ k. Địa chỉ nơi nhảy đến nạp vào PC và CPU tiếp tục thực hiện lệnh tại nơi nhảy đến.

17. Lệnh: **INCF** lệnh tăng nội dung thanh ghi f

- Cú pháp: INCF f,d
- Tác tố: $0 \leq f \leq 127$, $d \in [0,1]$
- Thực thi: $(f) + 1 \rightarrow (\text{dest})$. Cờ ảnh hưởng: Z. Chu kỳ thực hiện: 1.
- Chức năng: tăng nội dung của thanh ghi f lên 1. Nếu $d = 0$ thì kết quả lưu vào thanh ghi W. Nếu $d = 1$ thì kết quả lưu trở lại vào thanh ghi f.

18. Lệnh: **INCFSZ** lệnh tăng nội dung thanh ghi f và nhảy nếu bằng 0

- Cú pháp: INCFSZ f,d
- Tác tố: $0 \leq f \leq 127$, $d \in [0,1]$
- Thực thi: $(f) + 1 \rightarrow (\text{dest})$. Cờ ảnh hưởng: không. Chu kỳ thực hiện: 1(2).
- Chức năng: tăng nội dung của thanh ghi f lên 1. Nếu $d = 0$ thì kết quả lưu vào thanh ghi W. Nếu $d = 1$ thì kết quả lưu vào thanh ghi f. Nếu kết quả là bằng 0 thì bỏ qua lệnh kế và được thay bằng lệnh NOP.

19. Lệnh: **IORLW** lệnh OR hằng số với W

- Cú pháp: IORLW k
- Tác tố: $0 \leq k \leq 255$
- Thực thi: $(W) \text{ OR } k \rightarrow W$. Cờ ảnh hưởng: Z. Chu kỳ thực hiện: 1.
- Chức năng: Or hằng số k 8 bit với W.

20. Lệnh: **IORWF** lệnh OR W với f

- Cú pháp: IORWF f,d
- Tác tố: $0 \leq f \leq 127$
- Thực thi: $(W) \text{ OR } f \rightarrow (\text{dest})$. Cờ ảnh hưởng: Z. Chu kỳ thực hiện: 1.

- Chức năng: Or nội dung thanh ghi W với thanh ghi f. Nếu d=0 thì kết quả lưu vào thanh ghi W. Nếu d=1 thì kết quả lưu vào thanh ghi f.

21. Lệnh: **MOVLW** lệnh nạp dữ liệu vào thanh ghi W

- Cú pháp: MOVLW k
- Tác tố: $0 \leq k \leq 255$
- Thực thi: $k \rightarrow W$. Cờ ảnh hưởng: không. Chu kỳ thực hiện: 1.
- Chức năng: nạp dữ liệu 8 bit k vào thanh ghi W.

22. Lệnh: **MOVF** lệnh copy dữ liệu từ thanh ghi W sang f

- Cú pháp: MOVF f,d
- Tác tố: $0 \leq f \leq 127, d \in [0,1]$
- Thực thi: $(f) \rightarrow (nơi đến)$. Cờ ảnh hưởng: Z. Chu kỳ thực hiện: 1.
- Chức năng: copy nội dung thanh ghi 'f' sang nơi đến tùy thuộc vào giá trị của 'd'. Nếu 'd' = 0 thì nơi đến là thanh ghi W. Nếu 'd'=1 thì nơi đến chính là thanh ghi 'f'. Trường hợp 'd'=1 rất tiện lợi để kiểm tra thanh ghi file vì cờ Z bị ảnh hưởng: nếu nội dung thanh ghi f bằng 0 thì cờ Z bằng 1, nếu khác 0 thì cờ Z bằng 0.

23. Lệnh: **MOVWF** lệnh copy dữ liệu

- Cú pháp: MOVWF f
- Tác tố: $0 \leq f \leq 127, d \in [0,1]$
- Thực thi: $(W) \rightarrow f$. Trạng thái ảnh hưởng: không. Chu kỳ thực hiện: 1.
- Chức năng: copy nội dung thanh ghi W sang thanh ghi 'f'.

24. Lệnh: **NOP** lệnh không hoạt động

- Cú pháp: NOP
- Tác tố: không có
- Thực thi: không làm gì cả. Trạng thái ảnh hưởng: không. Chu kỳ thực hiện: 1.
- Chức năng: không làm gì cả

25. Lệnh: **RETFIE** lệnh trở về từ chương trình con phục vụ ngắt.

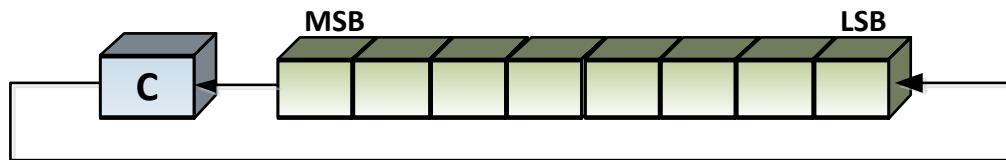
- Cú pháp: RETFIE
- Tác tố: không có.
- Thực thi: TOS \rightarrow PC, 1 \rightarrow GIE. Cờ ảnh hưởng: không. Chu kỳ thực hiện: 2.
- Chức năng: kết thúc chương trình con phục vụ ngắt, trở lại chương trình chính. Địa chỉ trả về ở đỉnh ngăn xếp trả lại cho PC, bit cho phép ngắt toàn cục GIE = 1 để cho phép ngắt.

26. Lệnh: **RETLW** lệnh trả về từ chương trình con

- Cú pháp: RETLW k
- Tác tố: $0 \leq k \leq 255$
- Thực thi: $k \rightarrow W, TOS \rightarrow PC$. Trạng thái ảnh hưởng: không. Chu kỳ thực hiện: 2.
- Chức năng: kết thúc chương trình con để trả lại chương trình chính, đồng thời nạp hàng số k vào thanh ghi W, địa chỉ trả về trong đỉnh ngăn xếp trả lại cho PC.

27. Lệnh: **RLF** lệnh xoay trái qua cờ C

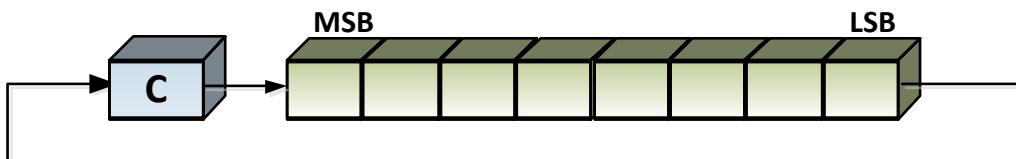
- Cú pháp: RLF f,d
- Tác động: $0 \leq f \leq 127$, $d \in [0,1]$
- Thực thi:



- Trạng thái ảnh hưởng: C. Chu kỳ thực hiện: 1.
- Chức năng: xoay nội dung của thanh ghi f và cờ C sang trái một bit. Nếu d= 0 thì kết quả được lưu vào thanh ghi W. Nếu d= 1 thì kết quả lưu vào thanh ghi f.

28. Lệnh: **RRF** lệnh xoay phải qua cờ C

- Cú pháp: RRF f,d
- Tác động: $0 \leq f \leq 127$, $d \in [0,1]$
- Thực thi:



- Trạng thái ảnh hưởng: C. Chu kỳ thực hiện: 1.
- Chức năng: xoay nội dung của thanh ghi f và cờ C sang phải 1 bit. Nếu d= 0 thì kết quả được lưu vào thanh ghi W. Nếu d= 1 thì kết quả lưu vào thanh ghi f.

29. Lệnh: **RETURN** lệnh kết thúc chương trình con

- Cú pháp: RETURN
- Tác động: không
- Thực thi: TOS \rightarrow PC. Trạng thái ảnh hưởng: không. Chu kỳ thực hiện: 2.
- Chức năng: lệnh kết thúc chương trình con trả lại chương trình chính, địa chỉ trả về trong đinh ngăn xếp trả lại cho PC.

30. Lệnh: **SLEEP** lệnh ngủ

- Cú pháp: SLEEP
- Tác động: không
- Thực thi: 00h \rightarrow WDT; 0 \rightarrow bộ đếm chia trước của WDT; 1 \rightarrow \overline{TO} ; 0 \rightarrow \overline{PD}
- Cờ ảnh hưởng: \overline{TO} , \overline{PD} . Chu kỳ thực hiện: 1.
- Chức năng: $\overline{PD} = 0$ cho biết CPU đang ở chế độ ngủ, bộ dao động ngừng hoạt động. Bit $\overline{TO} = 1$. Bộ định thời WDT và bộ chia trước bị xóa.

31. Lệnh: **SUBLW** lệnh trừ hàng số cho thanh ghi W

- Cú pháp: SUBLW k
- Tác động: $0 \leq k \leq 255$
- Thực thi: $k - (W) \rightarrow (W)$. Cờ ảnh hưởng: C, DC, Z. Chu kỳ thực hiện: 1.
- Chức năng: hàng số k 8 bit trừ cho nội dung thanh ghi W và kết quả lưu vào thanh ghi W.

32. Lệnh: **SUBWF** lệnh trừ thanh ghi f cho thanh ghi W

- Cú pháp: **SUBLW** f,d
- Tác tố: $0 \leq f \leq 127$, $d \in [0,1]$
- Thực thi: $(f) - (W) \rightarrow (\text{dest})$. Cờ ảnh hưởng: C, DC, Z. Chu kỳ thực hiện: 1.
- Chức năng: nội dung thanh ghi f trừ cho nội dung thanh ghi W. Nếu $d=0$ thì kết quả lưu vào thanh ghi W. Nếu $d=1$ thì kết quả lưu vào thanh ghi f.

33. Lệnh: **SWAPF** lệnh hoán chuyển 4 bit của thanh ghi f

- Cú pháp: **SWAPF** f,d
- Tác tố: $0 \leq f \leq 127$, $d \in [0,1]$
- Thực thi: $(f<3:0>) \rightarrow (\text{dest}<7:4>); (f<7:4>) \rightarrow (\text{dest}<3:0>)$
- Cờ ảnh hưởng: không. Chu kỳ thực hiện: 1.
- Chức năng: 4 bit cao và 4 bit thấp của thanh ghi f được đổi với nhau. Nếu $d=0$ thì kết quả lưu vào thanh ghi W. Nếu $d=1$ thì kết quả lưu vào thanh ghi f.

34. Lệnh: **XORLW** lệnh XOR hằng số với W

- Cú pháp: **XORLW** k
- Tác tố: $0 \leq k \leq 255$
- Thực thi: $(W) \text{ XOR } k \rightarrow (W)$
- Cờ ảnh hưởng: Z. Chu kỳ thực hiện: 1.
- Chức năng: XOR nội dung thanh ghi W với hằng số k 8 bit, kết quả lưu vào thanh ghi W.

35. Lệnh: **XORWF** lệnh XOR W với f

- Cú pháp: **XORLW** f,d
- Tác tố: $0 \leq f \leq 127$, $d \in [0,1]$
- Thực thi: $(W) \text{ XOR } (f) \rightarrow (\text{dest})$. Cờ ảnh hưởng: Z
- Chức năng: XOR nội dung thanh ghi W với nội dung thanh ghi f. Nếu $d=0$ thì kết quả được lưu vào thanh ghi W. Nếu $d=1$ thì kết quả lưu vào thanh ghi f.

3.4 CÂU HỎI ÔN TẬP – TRẮC NGHIỆM – BÀI TẬP

3.4.1 CÂU HỎI ÔN TẬP

Câu số 3-1: Hãy cho biết các kiểu định địa chỉ của vi điều khiển PIC16F887.

Câu số 3-2: Hãy cho biết các nhóm lệnh của vi điều khiển PIC16F887.

3.4.2 CÂU HỎI MỞ RỘNG

Câu số 3-3: Hãy tìm hiểu lệnh so sánh cấu trúc CISC và RISC.

Câu số 3-4: Hãy tìm cấu trúc đường ống lệnh.

3.4.3 CÂU HỎI TRẮC NGHIỆM

Câu 3-1: Tập lệnh của PIC 16F877A gồm:

- | | |
|---------------------|--------------------------------------|
| (a) Lệnh xử lý bit | (b) Lệnh xử lý hằng số và điều khiển |
| (c) Lệnh xử lý byte | (d) Tất cả đều đúng |

Câu 3-2: Mã lệnh của PIC 16F877A có chiều dài:

- | | |
|------------|------------|
| (a) 12 bit | (b) 13 bit |
| (c) 14 bit | (d) 15 bit |

Câu 3-3: Kí hiệu ‘f’ trong tập lệnh của PIC 16F877A:

- | | |
|--------------------------|--------------------------------|
| (a) Là tên của thanh ghi | (b) Là địa chỉ của 1 thanh ghi |
| (c) Là hằng số | (d) Là địa chỉ của 1 bit |

Câu 3-4: Kí hiệu ‘b’ trong tập lệnh của PIC 16F877A:

- | | |
|--------------------------|--------------------------------|
| (a) Là tên của thanh ghi | (b) Là địa chỉ của 1 thanh ghi |
| (c) Là hằng số | (d) Là địa chỉ của 1 bit |

Câu 3-5: Kí hiệu ‘d’ trong tập lệnh của PIC 16F877A:

- | | |
|--------------------------|-------------------------------|
| (a) Là tên của thanh ghi | (b) Cho biết nơi nhận dữ liệu |
| (c) Là hằng số | (d) Là địa chỉ của 1 bit |

Câu 3-6: Kí hiệu ‘b’ trong tập lệnh của PIC 16F877A có chiều dài:

- | | |
|-----------|------------|
| (a) 3 bit | (b) 1 bit |
| (c) 7 bit | (d) 14 bit |

Câu 3-7: Kí hiệu ‘f’ trong tập lệnh của PIC 16F877A có chiều dài:

- | | |
|-----------|------------|
| (a) 3 bit | (b) 1 bit |
| (c) 7 bit | (d) 14 bit |

Câu 3-8: Lệnh “ADDLW k” có chức năng:

- | | |
|--------------------------|-------------------------|
| (a) Cộng W với hằng số k | (b) And W với hằng số k |
| (c) Cộng W với hằng số 1 | (d) And W với hằng số 1 |

Câu 3-9: Lệnh “ADDFWF f, d” có chức năng:

- | | |
|--------------------------|--------------------------|
| (a) Cộng W với hằng số f | (b) Cộng W với hằng số d |
| (c) Cộng W với ô nhớ f | (d) Cộng f với hằng số d |

Câu 3-10: Lệnh “BCF f, b” có chức năng:

- | | |
|---------------------------------------|---------------------------------------|
| (a) Xóa bit f trong thanh ghi b | (b) Xóa bit b trong thanh ghi f |
| (c) Làm bit f trong thanh ghi b lên 1 | (d) Làm bit b trong thanh ghi f lên 1 |

Câu 3-11: Lệnh “BSF f, b” có chức năng:

- | | |
|---------------------------------|---------------------------------|
| (a) Xóa bit f trong thanh ghi b | (b) Xóa bit b trong thanh ghi f |
|---------------------------------|---------------------------------|

- (c) Làm bit f trong thanh ghi b lên 1 (d) Làm bit b trong thanh ghi f lên 1

Câu 3-12: Lệnh “BTFSS f, b” có chức năng kiểm tra:

- (a) Bit b trong thanh ghi f và nhảy nếu bằng 1
- (b) Bit f trong thanh ghi b và nhảy nếu = 1
- (c) Bit b trong thanh ghi f và nhảy nếu bằng 0
- (d) Bit f trong thanh ghi b và nhảy nếu = 0

Câu 3-13: Lệnh “BTFSC f, b” có chức năng kiểm tra:

- (a) Bit b trong thanh ghi f và nhảy nếu bằng 1
- (b) Bit f trong thanh ghi b và nhảy nếu = 1
- (c) Bit b trong thanh ghi f và nhảy nếu bằng 0
- (d) Bit f trong thanh ghi b và nhảy nếu = 0

Câu 3-14: Lệnh “DECF f, d” có chức năng:

- | | |
|-------------------------------|-------------------------------|
| (a) Giảm nội dung thanh ghi f | (b) Tăng nội dung thanh ghi f |
| (c) Giảm nội dung thanh ghi d | (d) Tăng nội dung thanh ghi d |

Câu 3-15: Lệnh “DECFSZ f, d” có chức năng:

- | | | |
|---|--------------------------------------|--------------------------------------|
| decrise thanh ghi f
s: skip
z: zero
d: nơi lưu | (a) Giảm thanh ghi f và nhảy nếu Z=1 | (b) Giảm thanh ghi f và nhảy nếu Z=0 |
| | (c) Giảm thanh ghi f và nhảy nếu d=0 | (d) Giảm thanh ghi f và nhảy nếu d=1 |

Câu 3-16: Lệnh “INCFSZ f, d” có chức năng:

- | | |
|--------------------------------------|--------------------------------------|
| (a) Tăng thanh ghi f và nhảy nếu Z=1 | (b) Tăng thanh ghi f và nhảy nếu Z=0 |
| (c) Tăng thanh ghi f và nhảy nếu d=0 | (d) Tăng thanh ghi f và nhảy nếu d=1 |

3.4.4 BÀI TẬP

VỊ ĐIỀU KHIỂN PIC16F887: NGÔN NGỮ LẬP TRÌNH C

4.1 GIỚI THIỆU

Ngôn ngữ lập trình C là một ngôn ngữ lập trình được sử dụng phổ biến, là ngôn ngữ tạo mã hiệu quả, các phần tử lập trình có cấu trúc và một tập hợp phong phú các toán tử.

Ngôn ngữ C một ngôn ngữ lập trình thuận tiện và hiệu quả, nhiều ứng dụng có thể được giải quyết dễ dàng hơn và hiệu quả hơn bằng ngôn ngữ C so với các ngôn ngữ chuyên biệt khác.

Ở chương 3 có thảo luận về ngôn ngữ lập trình hợp ngữ có nhiều hạn chế khi viết chương trình cho một ứng dụng lớn và phức tạp, tuy nhiên vẫn có các ưu điểm là mã lệnh tối ưu, tính toán được thời gian thực hiện các lệnh chính xác, khi viết bằng hợp ngữ thì người lập trình hiểu biết nhiều về cấu trúc phần cứng.

Khi lập trình bằng ngôn ngữ C thì các vấn đề phức tạp được giải quyết một cách nhanh chóng và gọn gàng, tuy nhiên không thể biết chính xác thời gian thực hiện các lệnh, mã sau khi biên dịch thường chưa tối ưu vì trình biên dịch thường phải tạo 1 chương trình khung tổng quát để đáp ứng cho tất cả các dạng lập trình nên code biên dịch dài hơn, người lập trình có thể không cần hiểu biết nhiều về cấu trúc của vi điều khiển hơn.

Vậy có thể nói ưu của ngôn ngữ này là khuyết của ngôn ngữ kia.

Ở chương này giới thiệu các ngôn ngữ lập trình C cho các loại vi điều khiển và các lệnh C cơ bản để phục vụ lập trình cho các ứng dụng.

Do có nhiều họ vi điều khiển của nhiều hãng khác nhau nên các phần mềm lập trình C cho vi điều khiển cũng khác nhau, phần này chỉ trình bày những kiến thức lập trình C chung và cơ bản nhất và tuy thuộc vào từng phần mềm biên dịch mà các bạn tìm hiểu thêm.

Sau khi kết thúc phần này sẽ giúp các bạn biết cấu trúc một chương trình, biết các lệnh C cơ bản để lập trình, biết khai báo các kiểu dữ liệu cho các biến, biết viết chương trình.

4.2 CÁC THÀNH PHẦN CƠ BẢN CỦA NGÔN NGỮ C

4.2.1 CÁC KIỂU DỮ LIỆU CỦA BIẾN

Trong chương trình thường khai báo biến để lưu trữ dữ liệu và xử lý dữ liệu, tùy thuộc vào loại dữ liệu mà ta phải chọn loại dữ liệu cho phù hợp. Các biến của vi xử lý bao gồm bit, byte, word và long word tương ứng với dữ liệu 1 bit, 8 bit, 16 bit và 32 bit.

Các kiểu dữ liệu cơ bản tùy thuộc vào phần mềm sử dụng, vi điều khiển PIC có rất nhiều phần mềm biên dịch như PIC-C, MIKRO-C, MPLAB, trong phần này trình bày phần mềm PIC-C.

Bảng 4-1. Các kiểu dữ liệu của phần mềm PIC-C.

TT	Kiểu dữ liệu	Số bit	Giới hạn
1	Int1	1	0÷1
2	signed char	8	-128÷127
3	unsigned char	8	0÷255
4	signed int hay signed int8	8	-128÷127
5	unsigned int hay unsigned int8	8	0÷255
6	signed int16	16	-32768÷32767

7	unsigned int16	16	0÷65535
8	signed long long	32	-2147483648 to 2147483647
9	unsigned long long	32	0 to 4294967295
10	float	32	±1.175494E-38 to ±3.402823E+38

Ví dụ 4-1: Khai báo các biến

Int1 TT; //khai báo biến trạng thái thuộc kiểu dữ liệu bit.

Unsigned char dem; //khai báo biến đếm (dem) thuộc kiểu kí tự không dấu 8 bit.

Chú ý : phần mềm lập trình PIC-C không phân biệt chữ hoa hay chữ thường nhưng các phần mềm khác thì có phân biệt.

4.2.2 CÁC TOÁN TỬ

Các toán tử là thành phần quan trọng trong lập trình, để lập trình thì chúng ta cần phải hiểu rõ ràng chức năng của các loại toán tử.

Bảng 4-2. Các toán tử phổ biến trong ngôn ngữ C.

TT	Toán tử	Chức năng	Ví dụ
1	+	Toán tử cộng	
2	+=	Toán tử cộng và gán.	$x+=y$ tương đương với $x=x+y$
3	&=	Toán tử and và gán.	$x&=y$ tương đương với $x=x\&y$
4	@	Toán tử địa chỉ	
5	&	Toán tử and	
6	^=	Toán tử ex-or và gán.	$x^=y$ tương đương với $x=x^y$
7	^	Toán tử ex-or	
8	=	Toán tử or và gán.	$x =y$ tương đương với $x=x y$.
9		Toán tử or nhiều đại lượng với nhau thành 1.	Ví dụ or nhiều bit trong 1 byte với nhau
10	--	Giảm	
11	/=	Toán tử chia và gán.	$x/=y$ tương đương với $x=x/y$
12	/	Toán tử chia	
13	==	Toán tử bằng dùng để so sánh	
14	>	Toán tử lớn hơn	
15	>=	Toán tử lớn hơn hay bằng	
16	++	Tăng	
17	*	Toán tử truy xuất gián tiếp, di trước con trỏ	
18	!=	Toán tử không bằng	
19	<=	Toán tử dịch trái và gán	$x <= y$ tương đương với $x=x<y$
20	<	Toán tử nhỏ hơn	
21	<<	Toán tử dịch trái	
22	<=	Toán tử nhỏ hơn hay bằng	

23	&&	Toán tử and	
24	!	Toán tử phủ định (not)	
25		Toán tử or	
26	%=	Toán tử chia lấy số dư và gán	$x \% = y$ tương đương với $x=x\%y$
27	%	Toán tử module	
28	*=	Toán tử nhân và gán	$x * = y$ tương đương với $x=x*y$
29	*	Toán tử nhân	
30	~	Toán tử bù 1	
31	>>=	Toán tử dịch phải và gán	$x >> = y$ tương đương với $x=x>>y$
32	>>	Toán tử dịch phải	
33	->	Toán tử con trả cấu trúc	
34	-=	Toán tử trừ và gán	$x - = y$ tương đương với $x=x-y$
35	-	Toán tử trừ	
36	sizeof	Xác định kích thước theo byte của toán tử	

a. Toán tử gán (=):

Dùng để gán một giá trị nào đó cho một biến

Ví dụ 4-2: $a = 5$; Gán biến a bằng 5

Ví dụ 4-3: $a = 2 + (b = 5)$;

Có chức năng gán biến b bằng 5 rồi cộng với 2 và gán cho biến a, kết quả $b = 5$ và $a = 7$.

b. Toán tử số học (+, -, *, /, %):

Có 5 toán tử để thực hiện các phép toán cộng, trừ, nhân, chia và chia lấy phần dư.

Ví dụ 4-4: $a = 24$; $b = a \% 5$;

Gán a bằng 24, b gán số dư của a chia cho 5, kết quả b bằng 4

Ví dụ 4-5: $a = 123$; $x = a \% 10$; $//x = 3$

$a = a / 10$; $//a = 12$

$y = a \% 10$; $//y = 2$

$z = a / 10$; $//z = 1$

Ví dụ này có chức năng tách từng con số đơn vị, chục, trăm gán cho 3 biến x, y, z.

Các lệnh trên có thể viết gọn lại như sau:

$x = a \% 10$; $//x = 3$

$y = a / 10 \% 10$; $//y = 2$

$z = a / 100$; $//z = 1$

Ví dụ 4-6: $a = 1234;$ $x = a \% 10;$ $//x=4$
 $A = a / 10;$ $//a = 12$
 $y = a \% 10;$ $//y=3$
 $a = a / 10;$ $//a = 12$
 $z = a \% 10;$ $//z=2$
 $v = a / 10;$ $//v=1$

Ví dụ này có chức năng tách từng số đơn vị, chục, trăm, ngàn gán cho 4 biến x, y, z, v.

Các lệnh trên có thể viết gọn lại như sau:

$x = a \% 10;$ $//x=3$
 $y = a / 10 \% 10;$ $//y=2$
 $z = a / 100 \% 10;$ $//z=1$
 $v = a / 1000;$ $//v=1$

c. Toán tử gán phức hợp ($+=$, $-=$, $*=$, $/=$, $\%=$, $>>=$, $<<=$, $\&=$, $\^=$, $/=$):

Tổng quát cho toán tử gán phức hợp: biến $+=$ giá_trị tương đương với biến $=$ biến + giá_trị.

Ví dụ 4-7: $a += 5;$ //tương đương với $a = a + 5;$
 $a /= 5;$ //tương đương $a = a / 5;$
 $b *= x + 1;$ //tương đương $b = b * (x+1);$

d. Toán tử tăng và giảm (++, --)

Ví dụ 4-8: $a++;$ tương đương với $a = a + 1$ hay $a += 1;$

Ví dụ 4-9: $b = 3;$ gán b bằng 3
 $a = ++b;$ kết quả a bằng 4, b bằng 4

Ví dụ 4-10: $b = 3;$ gán b bằng 3
 $A = B ++;$ kết quả a bằng b và bằng 3, b tăng lên 1 bằng 4

Sự khác nhau là “ $++$ ” đặt trước thì tính trước rồi mới gán, đặt sau thì gán trước rồi mới tính.

e. Toán tử quan hệ ($==$, $!=$, $>$, $<$, \geq , \leq):

so sánh $==$
so sánh khác: $!=$

Các toán tử quan hệ dùng để so sánh các biểu thức, kết quả so sánh là đúng hoặc sai.

Các toán tử trên tương ứng là bằng, khác, lớn hơn, nhỏ hơn, lớn hơn hay bằng, nhỏ hơn hay bằng.

Ví dụ 4-11: if (x<100) x+=1;

Lệnh trên kiểm tra nếu x còn nhỏ hơn 100 thì tăng x lên 1.

f. Toán tử logic (!, &&, ||):

Các toán tử trên tương đương là NOT, AND và OR.

Ví dụ 4-12: if ((nhiet_do_1>40) // (nhiet_do_2>40)) {bao_dong();}

Nếu nhiệt độ 1 lớn hơn 40 hoặc nhiệt độ 2 lớn hơn 40 thì báo động. Toán tử ‘||’ trong trường hợp này là or 2 điều kiện, chỉ cần 1 điều kiện đúng thì báo động.

g. Toán tử xử lý bit (&, |, ^, ~, <<, >>):

Các toán tử xử lý bit với bit, các toán tử trên tương đương là AND, OR, XOR, NOT, SHL (dịch trái), SHR (dịch phải).

Ví dụ 4-13:

A=0x77;	//gán A = 0111 0111B
B=0xC9;	//gán B = 1100 1001B
X = A & B;	// X bằng A and với B, kết quả X = 0100 0001B = 0x41
Y = A B;	// Y bằng A or với B, kết quả Y = 1111 1111B = 0xFF
Z = A ^ B;	// Z bằng A xor với B, kết quả Z = 1011 1110B = 0xBE
W = ~A;	// W bằng not A, kết quả W = 1000 1000B = 0x88

Ví dụ 4-14:

A=0x01; //gán A = 0000 0001B

A= (A <<1); //dịch A sang trái 1 bit, kết quả A = 0000 0010B.

A= (A <<1); //dịch A sang trái 1 bit, kết quả A = 0000 0100B.

Khi dịch trái thì bit bên trái mất, bit 0 thêm vào bên phải.

Ví dụ 4-15:

A=0x81; //gán A = 1000 0001B

A= (A >>1); //dịch A sang phải 1 bit, kết quả A = 0100 0000B.

Khi dịch phải thì bit bên phải mất, bit 0 thêm vào bên trái.

Ví dụ 4-16:

A=0x00; //gán A = 0000 0000B

A= (A <<1) +0x01; //dịch A sang trái 1 bit rồi cộng với 1, kết quả A = 0000 0001B.

```
A= (A <<1) +0x01; //dịch A sang trái 1 bit rồi cộng với 1, kết quả A = 0000  
0011B.
```

Khi dịch trái và cộng với 1 thì có chức năng đẩy số 1 thêm vào bên phải, với dữ liệu 8 bit thì sau 8 lần sẽ lấp đầy 8 bit 1.

h. Toán tử lấy kích thước chuỗi theo bye ():

Ví dụ 4-17: A = sizeof (charac); kết quả là A sẽ chứa số byte của chuỗi charac.

4.2.3 CÁC LỆNH C CƠ BẢN

Thành phần quan trọng thứ 3 trong lập trình C là các lệnh của ngôn ngữ C, phần tiếp theo sẽ khảo sát các lệnh cơ bản.

a. Lệnh if và else:

Chức năng: kiểm tra điều kiện nếu thỏa thì làm.

Cú pháp: if (điều_kiện)

{

Lệnh a1;

Lệnh a2;

...

}

else

{

Lệnh b1;

Lệnh b2;

...

}

Ví dụ 4-18:

```
if (x==50)
```

```
    x=1;
```

```
else
```

```
    x=x+1;
```

b. Lệnh lặp while:

Chức năng: lặp lại một thao tác với một số lần nhất định hoặc khi còn thỏa 1 điều kiện nào đó.

Cú pháp: while (điều_kiện)

```
{  
    Lệnh 1;  
    Lệnh 2;  
    ...  
}
```

Ví dụ 4-19:

```
while (x > 0)  
{ x = x - 1 ; }
```

c. Lệnh lặp do while:

Chức năng: làm các lệnh trong dấu ngoặc và thoát nếu điều kiện theo sau lệnh while không đúng.

Cú pháp: do

```
{  
    Lệnh 1;  
    Lệnh 2;  
    ...  
}  
while (điều_kiện);
```

Ví dụ 4-20:

```
Do  
{ x=x+10; }  
while (x < 100)
```

Thực hiện lệnh x bằng x cộng với 10, làm cho đến khi x nhỏ hơn 100.

d. Lệnh lặp for:

Chức năng: làm các lệnh trong dấu ngoặc một số lần nhất định.

Cú pháp: for (giá_trị_bắt_đầu; điều_kiện_kết_thúc; tăng_giá_trị)

```
{
```

Lệnh 1;

Lệnh 2;

...

}

Ví dụ 4-21:

For (int n = 0; n < 100; n++)

{ x=x+10; }

Vòng lặp thực hiện với biến n bằng 0 cho đến khi n bằng 100 thì ngừng.

e. Lệnh rẽ nhánh break:

Chức năng: Lệnh này dùng để thoát khỏi vòng lặp dù cho điều kiện để kết thúc chưa thỏa mãn.

for, while, do/while

f. Lệnh continue:

Chức năng: Lệnh này dùng để kết thúc phần còn lại của vòng lặp để làm lần lặp tiếp theo.

g. Lệnh rẽ nhánh goto:

nhảy không điều kiện

hiện tại

Chức năng: Lệnh này dùng để nhảy đến nhảy trong chương trình.

h. Lệnh switch case:

IF/ELSE

Chức năng: thực hiện 1 công việc tùy thuộc vào hàm.

Cú pháp: switch (cmd)

{

TD:

GOTO TD

Case constant1: Lệnh a1;

Lệnh a2;

Break;

Case constant2: Lệnh b1;

Lệnh b2;

Break;

Default: Lệnh c1;

Lệnh c2;

Break;

}

Ví dụ 4-22:

```

switch (cmd)

{
    Case 0: printf ("cmd 0");

    Break;

    Case 1: printf ("cmd 1");

    Break;

    Default: printf ("bad cmd ");

    Break;

}

```

4.2.4 CẤU TRÚC CỦA CHƯƠNG TRÌNH C

Chương trình C đơn giản như sau

```

/* my first C program*/

#include <16F887.H>

#FUSES NOWDT, PUT, HS, NOPROTECT, NOLVP

VOID MAIN()
{
    SET_TRIS_B(0x00);           //PORTB LA OUTPUT
    OUTPUT_B(0x0F);            //4 LED TAT 4 LED SANG
    WHILE(1)
    {
    }
}

```

“/* my first C program*/” là chú thích cho chương trình nằm trong cặp dấu “/* */” hoặc nằm sau “//”.

Lệnh “#include <16F887.H >” có chức năng báo cho trình biên dịch biết chương trình dùng các thành phần của vi điều khiển có trong file “16F887.H”.

Tiếp theo là khai báo cấu hình của vi điều khiển.

“VOID MAIN ()” cho biết bắt đầu chương trình chính và các lệnh tiếp theo của chương trình chính nằm trong cặp dấu {}.

4.2.5 CÁC THÀNH PHẦN CỦA CHƯƠNG TRÌNH C

Tất cả các chương trình viết bằng ngôn ngữ C bao gồm các chỉ dẫn, các khai báo biến, các định nghĩa biến, các biểu thức, các lệnh và hàm.

a. Chỉ dẫn tiền xử lý

Có 2 chỉ dẫn là #define và #include.

Chỉ dẫn #define có chức năng thay thế 1 đoạn chuỗi bằng một chỉ định đặc biệt nào đó.

Chỉ dẫn # include có chức năng chèn vào một đoạn lệnh từ 1 file khác vào trong chương trình.

Ví dụ 4-23:

```
#define RW PIN_B1
```

```
#include <16F887.H>
```

Chỉ dẫn thứ nhất có chức năng định nghĩa biến RW là bit thứ 1 của portB.

Chỉ dẫn thứ hai có chức năng khai báo chương trình thư viện của vi điều khiển 16F887.h

b. Khai báo

Khai báo biến bao gồm tên và thuộc tính, khai báo hàm, khai báo hằng. Khai báo biến toàn cục nằm bên ngoài hàm, khai báo biến cục bộ thì nằm bên trong hàm.

c. Định nghĩa giá trị cho biến

Dùng để thiết lập giá trị cho 1 biến hoặc 1 hàm.

d. Biểu thức

Là sự kết hợp của toán tử và tác tố để cho ra 1 kết quả duy nhất.

e. Hàm

Một hàm bao gồm các khai báo biến, định nghĩa giá trị, các biểu thức và các lệnh để thực hiện 1 chức năng đặc biệt nào đó.

4.2.6 CON TRỎ DỮ LIỆU

Con trỏ dữ liệu trong ngôn ngữ C chính là kiểu truy xuất gián tiếp dùng thanh ghi, địa chỉ của ô nhớ cần truy xuất lưu trong thanh ghi.

Ví dụ 4-24: khai báo con trỏ, gán địa chỉ con trỏ và lưu dữ liệu:

```
unsigned char *pointer0; //khai báo con trỏ
```

```
pointer0 = mem_address; //gán địa chỉ bắt đầu của vùng nhớ cho con trỏ
```

```
*pointer0 = 0xFF; //lưu giá trị 0xFF vào ô nhớ có địa chỉ lưu trong con trỏ.
```

4.2.7 KHAI BÁO MẢNG

Ví dụ 4-25: khai báo mảng:

```
unsigned char ma7doan[] = {0xC0,0xF9,0xA4,0xB0,0x99,0x92,0x82,0xF8,0x80,0x90} ;
```

```
char chuoi_hien_thi = {"HELLO!"};
```

Khi khai báo mảng thì phần tử đầu tiên có số thứ tự là 0, tiếp theo là 1 và cứ thế tiếp tục.

Ví dụ 4-26: Truy xuất mảng:

```
X = ma7doan[0]; //kết quả X = 0xC0) và đó là mã 7 đoạn của số 0
```

```
Y = ma7doan[5]; //kết quả Y = 0x92) và đó là mã 7 đoạn của số 5
```

4.3 TRÌNH BIÊN DỊCH C, THU' VIỆN

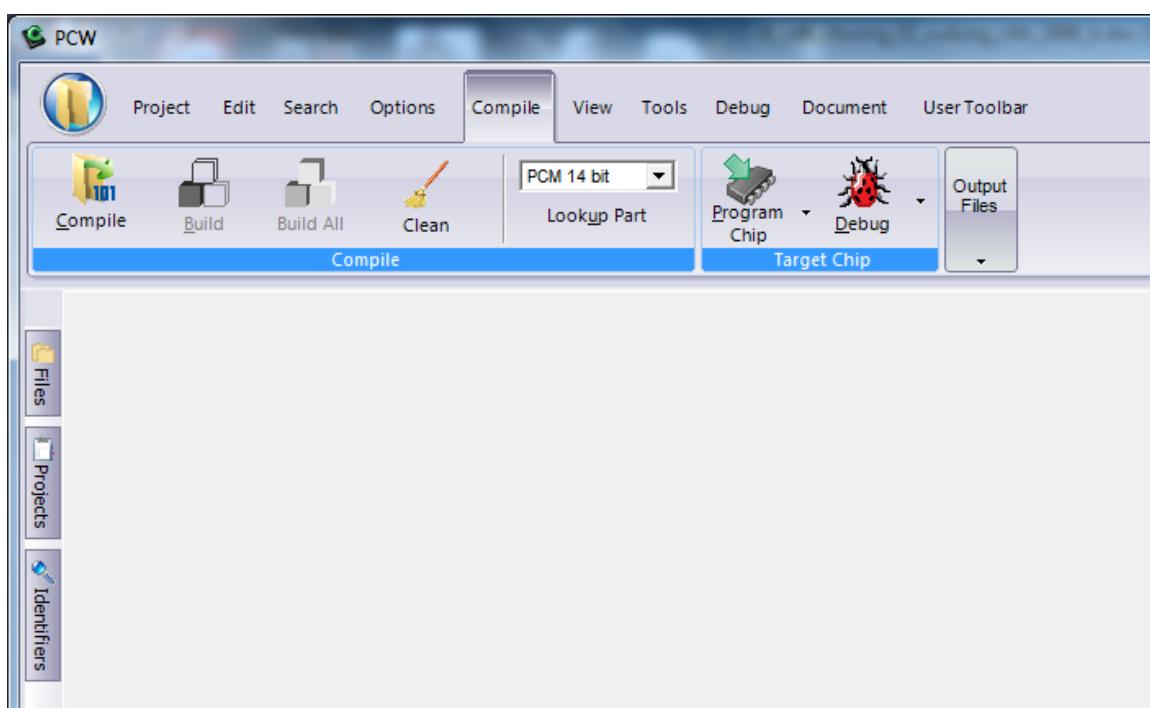
4.3.1 TRÌNH BIÊN DỊCH C

Có nhiều trình soạn thảo và biên dịch cho ngôn ngữ lập trình C của họ vi điều khiển PIC như MPLAB, Mikro C, PIC C. Trong giáo trình này giới thiệu trình biên dịch PIC-C dùng để biên dịch tập tin mã nguồn C cho họ vi điều khiển PIC với nhiều chủng loại khác nhau.

Để lập trình C chúng ta tải phần mềm biên dịch PIC-C, cài đặt, lập trình, biên dịch và sử dụng. Biểu tượng phần mềm CCS sau khi cài xong xuất hiện trên desktop như hình 4-1 và giao diện phần mềm CCS như hình 4-2.



Hình 4-1. Biểu tượng phần mềm CCS.



Hình 4-2. Giao diện phần mềm CCS.

Bạn có thể tham khảo cách sử dụng PIC C một cách chi tiết ở tài liệu thực hành vi điều khiển.

Khi lập trình cho vi điều khiển PIC dùng PIC-C thì các thành phần cơ bản cũng giống như các loại vi điều khiển khác, sự khác biệt chủ yếu ở phần cứng của từng vi điều khiển. Các lệnh liên quan đến phần cứng được trình bày theo phần cứng đó, khi khảo sát port thì có các lệnh PIC-C liên quan đến port, khi khảo sát timer/counter thì có các lệnh PIC-C liên quan đến timer/counter.

Khi dùng phần mềm PIC-C lập trình cho vi điều khiển PIC16F887 thì phải khai báo thư viện <PIC16F887.h>.

Trong file này đã định nghĩa tên các thành phần của vi điều khiển, nếu các tên trong file này không có thì bạn có thể định nghĩa thêm, trong phần mềm PIC-C có rất nhiều thư viện hỗ trợ cho các ứng dụng giúp bạn viết chương trình gọn hơn.

Nội dung của file thư viện:

```
//////// Standard Header file for the PIC16F887 device ///////////
#device PIC16F887
#nolist
//////// Program memory: 8192x14 Data RAM: 367 Stack: 8
//////// I/O: 36 Analog Pins: 14
//////// Data EEPROM: 256
//////// C Scratch area: 77 ID Location: 2000
//////// Fuses: LP,XT,HS,EC_IO,INTRC_IO,INTRC,RC_IO,RC,NOWDT,WDT,NOPUT,PUT
//////// Fuses: NOMCLR,MCLR,NOPROTECT,PROTECT,NOCPD,CPD,NOBROWNOUT,BROWNOUT
//////// Fuses: BROWNOUT_NOSL,BROWNOUT_SW,NOIESO,IESO,NOFCMEN,FCMEN,NOLVP
//////// Fuses: LVP,NODEBUG,DEBUG,WRT,NOWRT,BORV40,BORV21
////////
////////// I/O
// Discrete I/O Functions: SET_TRIS_x(), OUTPUT_x(), INPUT_x(),
//                         PORT_x_PULLUPS(), INPUT(),
//                         OUTPUT_LOW(), OUTPUT_HIGH(),
//                         OUTPUT_FLOAT(), OUTPUT_BIT()
// Constants used to identify pins in the above are:
```

#define	PIN_A0	40
#define	PIN_A1	41
#define	PIN_A2	42
#define	PIN_A3	43
#define	PIN_A4	44
#define	PIN_A5	45
#define	PIN_A6	46
#define	PIN_A7	47
#define	PIN_B0	48
#define	PIN_B1	49
#define	PIN_B2	50
#define	PIN_B3	51
#define	PIN_B4	52
#define	PIN_B5	53

```
#define PIN_B6      54
#define PIN_B7      55

#define PIN_C0      56
#define PIN_C1      57
#define PIN_C2      58
#define PIN_C3      59
#define PIN_C4      60
#define PIN_C5      61
#define PIN_C6      62
#define PIN_C7      63

#define PIN_D0      64
#define PIN_D1      65
#define PIN_D2      66
#define PIN_D3      67
#define PIN_D4      68
#define PIN_D5      69
#define PIN_D6      70
#define PIN_D7      71

#define PIN_E0      72
#define PIN_E1      73
#define PIN_E2      74
#define PIN_E3      75
```

/////////// Useful defines

```
#define FALSE     0
#define TRUE      1

#define BYTE       int8
#define BOOLEAN    int1

#define getc       getch
#define fgetc      getch
#define getchar    getch
#define putc       putchar
#define fputc      putchar
#define fgets      gets
#define fputs      puts
```

/////////// Control

```
// Control Functions: RESET_CPU(), SLEEP(), RESTART_CAUSE()
// Constants returned from RESTART_CAUSE() are:
#define WDT_FROM_SLEEP 3
```

```
#define WDT_TIMEOUT 11
#define MCLR_FROM_SLEEP 19
#define MCLR_FROM_RUN 27
#define NORMAL_POWER_UP 25
#define BROWNOUT_RESTART 26
```

////////////////////////////// Timer 0

// Timer 0 (AKA RTCC) Functions: SETUP_COUNTERS() or SETUP_TIMER_0(),

```
//           SET_TIMER0() or SET_RTCC(),
//           GET_TIMER0() or GET_RTCC()
```

// Constants used for SETUP_TIMER_0() are:

```
#define T0_INTERNAL 0
#define T0_EXT_L_TO_H 32
#define T0_EXT_H_TO_L 48
```

```
#define T0_DIV_1 8
#define T0_DIV_2 0
#define T0_DIV_4 1
#define T0_DIV_8 2
#define T0_DIV_16 3
#define T0_DIV_32 4
#define T0_DIV_64 5
#define T0_DIV_128 6
#define T0_DIV_256 7
```

```
#define T0_8_BIT 0
```

```
#define RTCC_INTERNAL 0 // The following are provided for compatibility
#define RTCC_EXT_L_TO_H 32 // with older compiler versions
#define RTCC_EXT_H_TO_L 48
#define RTCC_DIV_1 8
#define RTCC_DIV_2 0
#define RTCC_DIV_4 1
#define RTCC_DIV_8 2
#define RTCC_DIV_16 3
#define RTCC_DIV_32 4
#define RTCC_DIV_64 5
#define RTCC_DIV_128 6
#define RTCC_DIV_256 7
#define RTCC_8_BIT 0
```

// Constants used for SETUP_COUNTERS() are the above
// constants for the 1st param and the following for
// the 2nd param:

```
/////////////////////////////// WDT
// Watch Dog Timer Functions: SETUP_WDT() or SETUP_COUNTERS() (see above)
//           RESTART_WDT()
// WDT base is 18ms
//
```

#define	WDT_18MS	8
#define	WDT_36MS	9
#define	WDT_72MS	10
#define	WDT_144MS	11
#define	WDT_288MS	12
#define	WDT_576MS	13
#define	WDT_1152MS	14
#define	WDT_2304MS	15

// One of the following may be OR'ed in with the above:

#define	WDT_ON	0x4100
#define	WDT_OFF	0
#define	WDT_DIV_16	0x100
#define	WDT_DIV_8	0x300
#define	WDT_DIV_4	0x500
#define	WDT_DIV_2	0x700
#define	WDT_TIMES_1	0x900 // Default
#define	WDT_TIMES_2	0xB00
#define	WDT_TIMES_4	0xD00
#define	WDT_TIMES_8	0xF00
#define	WDT_TIMES_16	0x1100
#define	WDT_TIMES_32	0x1300
#define	WDT_TIMES_64	0x1500
#define	WDT_TIMES_128	0x1700

```
///////////////////////////// Timer 1
// Timer 1 Functions: SETUP_TIMER_1, GET_TIMER1, SET_TIMER1
// Constants used for SETUP_TIMER_1() are:
//   (or (via l) together constants from each group)
#define T1_DISABLED 0
#define T1_INTERNAL 5
#define T1_EXTERNAL 7
#define T1_EXTERNAL_SYNC 3

#define T1_CLK_OUT 8

#define T1_DIV_BY_1 0
#define T1_DIV_BY_2 0x10
```

```

#define      T1_DIV_BY_4      0x20
#define      T1_DIV_BY_8      0x30

#define      T1_GATE          0x40
#define      T1_GATE_INVERTED 0xC0
/////////////////////////////// Timer 2
// Timer 2 Functions: SETUP_TIMER_2, GET_TIMER2, SET_TIMER2
// Constants used for SETUP_TIMER_2() are:
#define      T2_DISABLED      0
#define      T2_DIV_BY_1       4
#define      T2_DIV_BY_4       5
#define      T2_DIV_BY_16      6

/////////////////////////////// CCP
// CCP Functions: SETUP_CCPx, SET_PWMx_DUTY
// CCP Variables: CCP_x, CCP_x_LOW, CCP_x_HIGH
// Constants used for SETUP_CCPx() are:
#define      CCP_OFF          0
#define      CCP_CAPTURE_FE    4
#define      CCP_CAPTURE_RE    5
#define      CCP_CAPTURE_DIV_4 6
#define      CCP_CAPTURE_DIV_16 7
#define      CCP_COMPARE_SET_ON_MATCH 8
#define      CCP_COMPARE_CLR_ON_MATCH 9
#define      CCP_COMPARE_INT    0xA
#define      CCP_COMPARE_RESET_TIMER 0xB
#define      CCP_PWM           0xC
#define      CCP_PWM_PLUS_1     0x1c
#define      CCP_PWM_PLUS_2     0x2c
#define      CCP_PWM_PLUS_3     0x3c
#define      CCP_1              getenv("SFR:CCPR1L")
#define      CCP_1_LOW=         getenv("SFR:CCPR1L")
#define      CCP_1_HIGH=        getenv("SFR:CCPR1H")
// The following should be used with the ECCP unit only (or these in)
#define      CCP_PWM_H_H       0x0c
#define      CCP_PWM_H_L       0x0d
#define      CCP_PWM_L_H       0x0e
#define      CCP_PWM_L_L       0x0f

#define      CCP_PWM_FULL_BRIDGE 0x40
#define      CCP_PWM_FULL_BRIDGE_REV 0xC0
#define      CCP_PWM_HALF_BRIDGE 0x80

#define      CCP_SHUTDOWN_ON_COMP1 0x100000
#define      CCP_SHUTDOWN_ON_COMP2 0x200000

```

```

#define CCP_SHUTDOWN_ON_COMP 0x300000
#define CCP_SHUTDOWN_ON_INT0 0x400000
#define CCP_SHUTDOWN_ON_COMP1_INT0 0x500000
#define CCP_SHUTDOWN_ON_COMP2_INT0 0x600000
#define CCP_SHUTDOWN_ON_COMP_INT0 0x700000

#define CCP_SHUTDOWN_AC_L 0x000000
#define CCP_SHUTDOWN_AC_H 0x040000
#define CCP_SHUTDOWN_AC_F 0x080000

#define CCP_SHUTDOWN_BD_L 0x000000
#define CCP_SHUTDOWN_BD_H 0x010000
#define CCP_SHUTDOWN_BD_F 0x020000

#define CCP_SHUTDOWN_RESTART 0x80000000

#define CCP_PULSE_STEERING_A 0x01000000
#define CCP_PULSE_STEERING_B 0x02000000
#define CCP_PULSE_STEERING_C 0x04000000
#define CCP_PULSE_STEERING_D 0x08000000
#define CCP_PULSE_STEERING_SYNC 0x10000000

#word CCP_2 = getenv("SFR:CCPR2L")
#byte CCP_2_LOW= getenv("SFR:CCPR2L")
#byte CCP_2_HIGH= getenv("SFR:CCPR2H")
////////////////////////////// SPI
// SPI Functions: SETUP_SPI, SPI_WRITE, SPI_READ, SPI_DATA_IN
// Constants used in SETUP_SPI() are:
#define SPI_MASTER 0x20
#define SPI_SLAVE 0x24
#define SPI_L_TO_H 0
#define SPI_H_TO_L 0x10
#define SPI_CLK_DIV_4 0
#define SPI_CLK_DIV_16 1
#define SPI_CLK_DIV_64 2
#define SPI_CLK_T2 3
#define SPI_SS_DISABLED 1

#define SPI_SAMPLE_AT_END 0x8000
#define SPI_XMIT_L_TO_H 0x4000
////////////////////////////// UART
// Constants used in setup_uart() are:
// FALSE - Turn UART off
// TRUE - Turn UART on

```

```

#define      UART_ADDRESS          2
#define      UART_DATA             4
#define      UART_AUTODETECT       8
#define      UART_AUTODETECT_NOWAIT 9
#define      UART_WAKEUP_ON_RDA    10
#define      UART_SEND_BREAK       13
/////////////////////////////// COMP
// Comparator Variables: C1OUT, C2OUT
// Constants used in setup_comparator() are:
//

#define      NC_NC_NC_NC          0x00
#define      NC_NC                 0x00

//Pick one constant for COMP1
#define      CP1_A0_A3              0x00090080
#define      CP1_A1_A3              0x000A0081
#define      CP1_B3_A3              0x00880082
#define      CP1_B1_A3              0x00280083
#define      CP1_A0_VREF            0x00010084
#define      CP1_A1_VREF            0x00020085
#define      CP1_B3_VREF            0x00800086
#define      CP1_B1_VREF            0x00200087

//Optionally OR with one or both of the following
#define      CP1_OUT_ON_A4          0x00000020
#define      CP1_INVERT              0x00000010
#define      CP1_ABSOLUTE_VREF       0x20000000

//OR with one constant for COMP2
#define      CP2_A0_A2              0x00058000
#define      CP2_A1_A2              0x00068100
#define      CP2_B3_A2              0x00848200
#define      CP2_B1_A2              0x00248300
#define      CP2_A0_VREF            0x00018400
#define      CP2_A1_VREF            0x00028500
#define      CP2_B3_VREF            0x00808600
#define      CP2_B1_VREF            0x00208700

//Optionally OR with one or both of the following
#define      CP2_OUT_ON_A5          0x00002000
#define      CP2_INVERT              0x00001000
#define      CP2_ABSOLUTE_VREF       0x10000000

//Optionally OR with one or both of the following
#define      CP2_T1_SYNC             0x01000000
#define      CP2_T1_GATE              0x02000000

```

```

#define C1OUT 0x107.6
#define C2OUT 0x108.6
////////////////////////////// VREF
// Constants used in setup_vref() are:
//
#define VREF_LOW 0xa0
#define VREF_HIGH 0x80
// Or (with |) the above with a number 0-15
////////////////////////////// INTERNAL RC
// Constants used in setup_oscillator() are:
#define OSC_31KHZ 1
#define OSC_125KHZ 0x11
#define OSC_250KHZ 0x21
#define OSC_500KHZ 0x31
#define OSC_1MHZ 0x41
#define OSC_2MHZ 0x51
#define OSC_4MHZ 0x61
#define OSC_8MHZ 0x71
#define OSC_INTRC 1
#define OSC_NORMAL 0
// Result may be (ignore all other bits)
#define OSC_STATE_STABLE 4
#define OSC_31KHZ_STABLE 2

////////////////////////////// ADC
// ADC Functions: SETUP_ADC(), SETUP_ADC_PORTS() (aka SETUP_PORT_A),
// SET_ADC_CHANNEL(), READ_ADC()
// Constants used for SETUP_ADC() are:
#define ADC_OFF 0 // ADC Off
#define ADC_CLOCK_DIV_2 0x100
#define ADC_CLOCK_DIV_8 0x40
#define ADC_CLOCK_DIV_32 0x80
#define ADC_CLOCK_INTERNAL 0xc0 // Internal 2-6us

// Constants used in SETUP_ADC_PORTS() are:
#define sAN0 1 // A0
#define sAN1 2 // A1
#define sAN2 4 // A2
#define sAN3 8 // A3
#define sAN4 16 // A5
#define sAN5 32 // E0
#define sAN6 64 // E1
#define sAN7 128 // E2
#define sAN8 0x10000 // B2
#define sAN9 0x20000 // B3

```

```
#define sAN10          0x40000 //| B1
#define sAN11          0x80000 //| B4
#define sAN12          0x100000 //| B0
#define sAN13          0x200000 //| B5
#define NO_ANALOGS      0 // None
#define ALL_ANALOG     0x1F00FF // A0 A1 A2 A3 A5 E0 E1 E2 B0 B1 B2 B3 B4 B5
```

// One of the following may be OR'ed in with the above using |

```
#define VSS_VDD        0x0000 // Range 0-Vdd
#define VSS_VREF       0x1000 // Range 0-Vref
#define VREF_VREF      0x3000 // Range Vref-Vref
#define VREF_VDD        0x2000 // Range Vref-Vdd
```

// Constants used in READ_ADC() are:

```
#define ADC_START_AND_READ 7 // This is the default if nothing is
specified
#define DC_START_ONLY      1
#define ADC_READ_ONLY      6
////////////////////////////// INT
// Interrupt Functions: ENABLE_INTERRUPTS(), DISABLE_INTERRUPTS(),
// CLEAR_INTERRUPT(), INTERRUPT_ACTIVE(),
// EXT_INT_EDGE()
```

// Constants used in EXT_INT_EDGE() are:

```
#define L_TO_H          0x40
#define H_TO_L          0
```

// Constants used in ENABLE/DISABLE_INTERRUPTS() are:

```
#define GLOBAL         0x0BC0
#define INT_RTCC        0x000B20
#define INT_RB          0x01FF0B08
#define INT_EXT_L2H     0x50000B10
#define INT_EXT_H2L     0x60000B10
#define INT_EXT         0x000B10
#define INT_AD          0x008C40
#define INT_TBE         0x008C10
#define INT_RDA         0x008C20
#define INT_TIMER1      0x008C01
#define INT_TIMER2      0x008C02
#define INT_CCP1         0x008C04
#define INT_CCP2         0x008D01
#define INT_SSP          0x008C08
#define INT_BUSCOL      0x008D08
#define INT_EEPROM       0x008D10
#define INT_TIMER0      0x000B20
#define INT_OSC_FAIL    0x008D80
#define INT_COMP         0x008D20
```

```
#define INT_COMP2 0x008D40
#define INT_ULPWU 0x008D04
#define INT_RB0 0x0010B08
#define INT_RB1 0x0020B08
#define INT_RB2 0x0040B08
#define INT_RB3 0x0080B08
#define INT_RB4 0x0100B08
#define INT_RB5 0x0200B08
#define INT_RB6 0x0400B08
#define INT_RB7 0x0800B08
#endif
```

4.4 CÂU HỎI ÔN TẬP – TRẮC NGHIỆM - BÀI TẬP

4.4.1 CÂU HỎI ÔN TẬP

Câu số 4-1: Hãy cho biết loại dữ liệu trong ngôn ngữ lập trình C cho vi điều khiển.

Câu số 4-2: Hãy cho biết các toán tử trong ngôn ngữ lập trình C.

4.4.2 CÂU HỎI MỞ RỘNG

Câu số 4-3: Hãy tìm hiểu các kiểu bộ nhớ trong ngôn ngữ lập trình C của vi điều ATMEGA128.

Câu số 4-4: Hãy tìm trình biên dịch Mikro-C.

4.4.3 CÂU HỎI TRẮC NGHIỆM

Câu 4-1: Kiểu dữ liệu “signed char” là kiểu:

- | | |
|------------|------------|
| (a) 8 bit | (b) 16 bit |
| (c) 24 bit | (d) 32 bit |

Câu 4-2: Giới hạn của kiểu dữ liệu “signed char” là:

- | | |
|---------------------|---------------------|
| (a) Từ 0 đến 255 | (b) Từ 0 đến 65535 |
| (c) Từ -256 đến 255 | (d) Từ -128 đến 127 |

Câu 4-3: Kiểu dữ liệu “unsigned char” là kiểu:

- | | |
|------------|------------|
| (a) 8 bit | (b) 16 bit |
| (c) 24 bit | (d) 32 bit |

Câu 4-4: Giới hạn của kiểu dữ liệu “unsigned char” là:

- | | |
|---------------------|---------------------|
| (a) Từ 0 đến 255 | (b) Từ 0 đến 65535 |
| (c) Từ -256 đến 255 | (d) Từ -128 đến 127 |

Câu 4-5: Kiểu dữ liệu “ unsigned int” là kiểu:

- | | |
|------------|------------|
| (a) 8 bit | (b) 16 bit |
| (c) 24 bit | (d) 32 bit |

Câu 4-6: Giới hạn của kiểu dữ liệu “unsigned int” là:

- | | |
|---------------------|---------------------|
| (a) Từ 0 đến 255 | (b) Từ 0 đến 65535 |
| (c) Từ -256 đến 255 | (d) Từ -128 đến 127 |

Câu 4-7: Toán tử nào là toán tử gán:

- | | |
|---------|---------|
| (a) “—” | (b) “%” |
| (c) “/” | (d) “&” |

Câu 4-8: Toán tử nào là chia lấy số dư:

- | | |
|---------|---------|
| (a) “=” | (b) “%” |
| (c) “/” | (d) “&” |

Câu 4-9: Toán tử nào là chia lấy kết quả nguyên:

- | | |
|---------|---------|
| (a) “=” | (b) “%” |
| (c) “/” | (d) “&” |

Câu 4-10: Toán tử nào là so sánh xem có bằng hay không:

- | | |
|----------|----------|
| (a) “=” | (b) “!=” |
| (c) “==” | (d) “<=” |

Câu 4-11: Toán tử nào là so sánh không bằng:

- | | |
|----------|----------|
| (a) “=” | (b) “!=” |
| (c) “==” | (d) “<=” |

Câu 4-12: Toán tử nào là so sánh nhỏ hơn hoặc bằng:

- | | |
|----------|----------|
| (a) “>=” | (b) “!=” |
| (c) “==” | (d) “<=” |

Câu 4-13: Toán tử nào là AND 2 điều kiện với nhau:

- | | |
|---------|----------|
| (a) “!” | (b) “&&” |
| (c) “&” | (d) “ ” |

Câu 4-14: Kiểu dữ liệu nào là 32 bit:

- | | |
|------------------|-------------------|
| (a) int | (b) unsigned long |
| (c) unsigned int | (d) signed int |

Câu 4-15: Toán tử nào là OR 2 điều kiện với nhau:

- | | |
|---------|----------|
| (a) “!” | (b) “&&” |
| (c) “&” | (d) “ ” |

Câu 4-16: Toán tử nào là phủ định dữ liệu:

- | | |
|---------|---------|
| (a) “!” | (b) “~” |
| (c) “&” | (d) “ ” |

Câu 4-17: Toán tử nào là phủ định điều kiện:

- | | |
|---------|----------|
| (a) “!” | (b) “&&” |
| (c) “&” | (d) “ ” |

Câu 4-18: Toán tử nào là AND 2 dữ liệu:

- | | |
|----------|---------|
| (a) “>>” | (b) “~” |
| (c) “&” | (d) “ ” |

Câu 4-19: Toán tử nào là dịch trái dữ liệu:

- | | |
|----------|----------|
| (a) “<<” | (b) “~” |
| (c) “&” | (d) “>>” |

Câu 4-20: Toán tử lấy kích thước theo byte:

- | | |
|---------------|--------------|
| (a) “size” | (b) “sizeof” |
| (c) “sizeoff” | (d) “sizeon” |

Câu 4-21: Vòng lặp while(điều_kiện) sẽ làm khi điều kiện:

- | | |
|---------------|---------------|
| (a) Lớn hơn 1 | (b) Nhỏ hơn 1 |
| (c) Sai | (d) Đúng |

Câu 4-22: Vòng lặp for($i=1; i < 10; i++$) sẽ thực hiện các lệnh trong vòng lặp:

- | | |
|------------|-------------|
| (a) 10 lần | (b) 100 lần |
| (c) 9 lần | (d) 99 lần |

Câu 4-23: Chỉ dẫn “#define” có chức năng:

- | | |
|---|---|
| (a) Chèn 1 đoạn lệnh từ 1 file khác vào | (b) Khai báo thư viện của vi điều khiển |
| (c) Định nghĩa 1 đoạn chuỗi | (d) Khai báo thư viện của C |

Câu 4-24: Chỉ dẫn “#include” có chức năng:

- | | |
|---|---|
| (a) Chèn 1 đoạn lệnh từ 1 file khác vào | (b) Khai báo thư viện của vi điều khiển |
| (c) Khai báo thư viện | (d) Tất cả đều đúng |

4.4.4 BÀI TẬP

VỊ ĐIỀU KHIỂN PIC16F8873: GIAO TIẾP LED, LCD, PHÍM ĐƠN, MÀ TRẬN PHÍM

5.1 GIỚI THIỆU

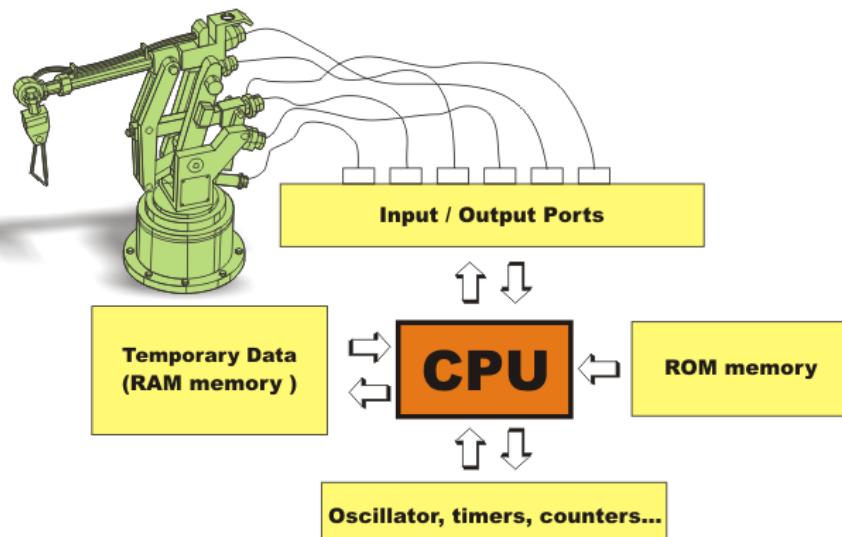
Ở chương này khảo sát các port xuất nhập của các vi điều khiển, các lệnh để định cấu hình cho port và các lệnh xuất nhập dữ liệu cho port.

Sau khi kết thúc chương này thì người đọc có thể sử dụng được các port để điều khiển các đối tượng như led đơn, led 7 đoạn, LCD, nhận tín hiệu điều khiển từ nút nhấn, bàn phím ma trận, các cảm biến.

5.2 CHỨC NĂNG CÁC PORT CỦA VI ĐIỀU KHIỂN

Vi điều khiển có các port để xuất nhập dữ liệu giao tiếp với các đối tượng điều khiển.

Tín hiệu điều khiển từ CPU gửi ra port để điều khiển, đồng thời có các port nhận dữ liệu về để xử lý. Trong một hệ thống luôn có các tín hiệu vào ra ví như hệ thống điều khiển robo như hình 5-1.

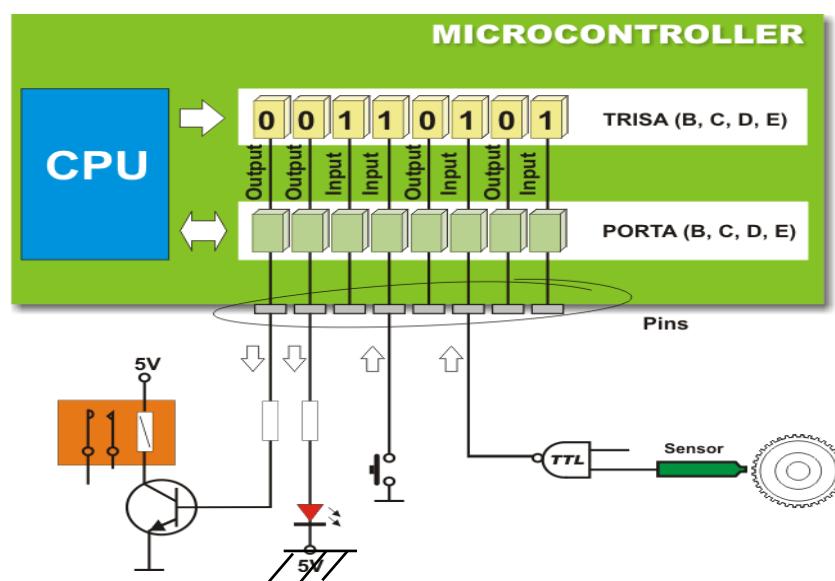


Hình 5-1. Sơ đồ kết nối port với đối tượng điều khiển.

Các port chỉ tạo ra các mức logic tương thích với chuẩn TTL, nếu điều khiển các đối tượng công suất lớn thì phải thêm mạch giao tiếp.

5.3 CÁC PORT CỦA PIC 16F887

Vi điều khiển PIC 16F887 có 5 port là portA, B, C, D và E, khả năng cấp và nhận dòng là 25mA.



Hình 5-2. Sơ đồ kết nối port xuất nhập tín hiệu điều khiển.

Mỗi port của vi điều khiển PIC gồm có thanh ghi port và thanh ghi định hướng cho port.

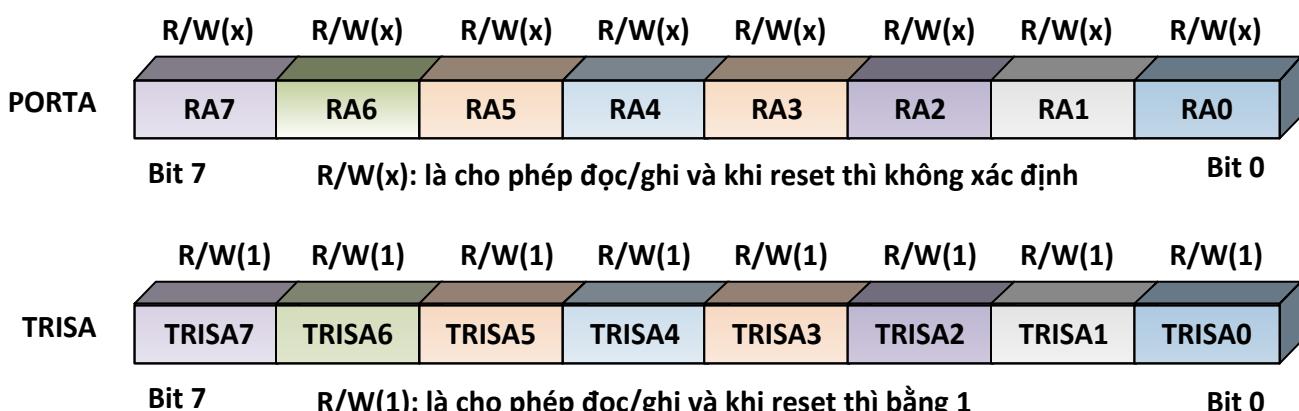
Hình 5-2 trình bày PORTA và thanh ghi định hướng TRISA. Bit của thanh ghi định hướng TRIS bằng 0 thì port có chức năng xuất dữ liệu, nếu bằng 1 thì có chức năng nhập dữ liệu.

Chú ý: '0' tương ứng với 'OUT', '1' tương ứng với 'IN'.

Phần tiếp sẽ khảo sát chi tiết từng port.

5.3.1 PORTA VÀ THANH GHI TRISA

PortA là port hai chiều 8 bit, thanh ghi định hướng là TRISA có chức năng định cấu hình vào ra cho các bit xuất nhập của portA, bit định hướng bằng 0 thì port tương ứng xuất, bit bằng 1 thì bit tương ứng là nhập, portA và thanh ghi định hướng TRISA như hình 5-3.



Hình 5-3. PortA và thanh ghi định hướng port A.

Khảo sát cấu hình mạch cho từng tín hiệu.

Chân RA0/AN0/ULPWU/C12IN0-:	Chân RA1/AN1/C12IN1-:
Có sơ đồ mạch như hình 5-4. Có thể định cấu hình thực hiện 1 trong các chức năng sau: <ul style="list-style-type: none"> • Là tín hiệu xuất nhập số I/O. • Là ngõ vào kênh thứ 0 của khối ADC. • Là ngõ vào âm thứ 0 của bộ so sánh C1, C2 • Là ngõ vào tương tự để đánh thức CPU khỏi chế độ ngủ 	Có sơ đồ mạch như hình 5-5. Có thể định cấu hình thực hiện 1 trong các chức năng sau: <ul style="list-style-type: none"> • Là tín hiệu xuất nhập số I/O. • Là ngõ vào kênh thứ 1 của khối ADC. • Là ngõ vào âm thứ 1 của bộ so sánh C1, C2

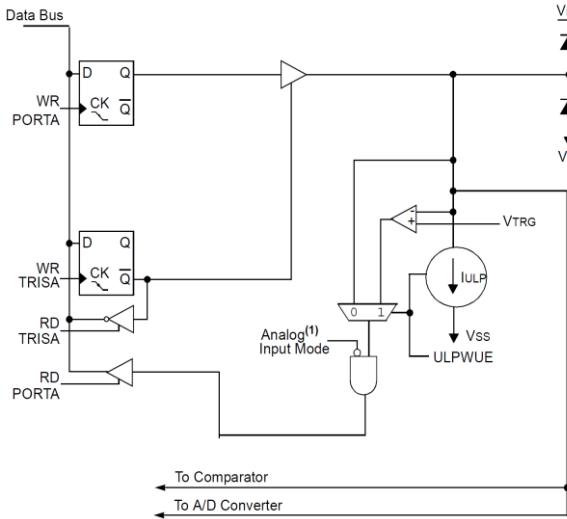
Chân RA2/AN2/V _{REF} -/CVREF/C2IN+:	Chân RA3/AN3/V _{REF} +/C1IN+
Có sơ đồ mạch như hình 5-6. Có thể định cấu hình thực hiện 1 trong các chức năng sau: <ul style="list-style-type: none"> • Là tín hiệu xuất nhập số I/O. • Là ngõ vào kênh thứ 2 của khối ADC. • Là ngõ vào điện áp tham chiếu âm cho 	Có sơ đồ mạch như hình 5-7. Có thể định cấu hình thực hiện 1 trong các chức năng sau: <ul style="list-style-type: none"> • Là tín hiệu xuất nhập số I/O. • Là ngõ vào kênh thứ 3 của khối ADC. • Là ngõ vào điện áp tham chiếu dương

ADC

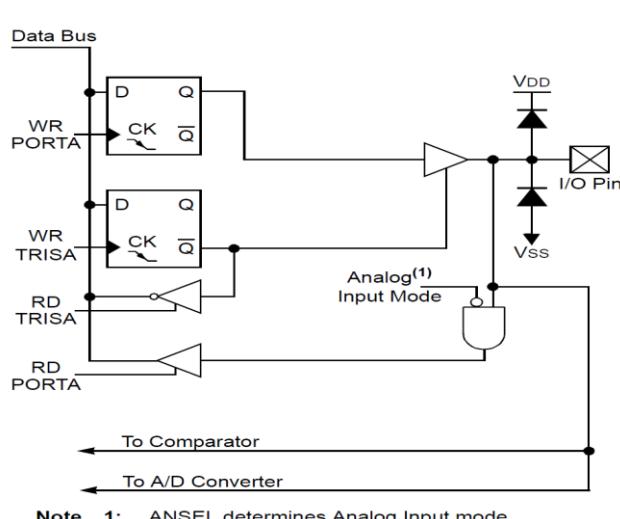
- Là ngõ vào điện áp tham chiếu bộ so sánh.
- Là ngõ vào dương của bộ so sánh C2

cho ADC

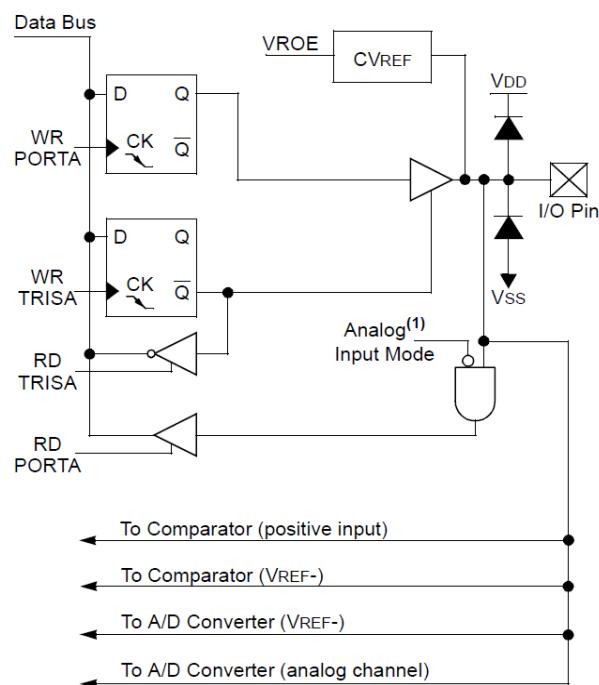
- Là ngõ vào dương của bộ so sánh C1



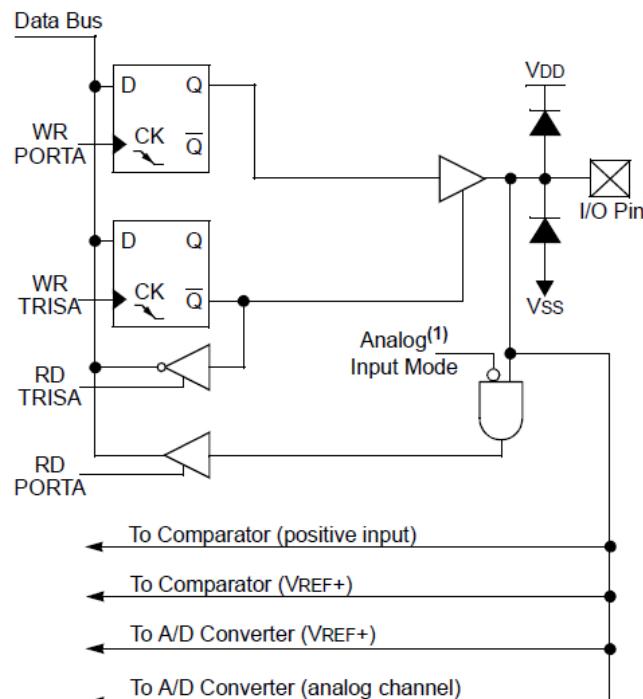
Hình 5-4. Cấu hình chân RA0.



Hình 5-5. Cấu hình chân RA1.



Hình 5-6. Cấu hình chân RA2.



Hình 5-7. Cấu hình chân RA3.

Chân RA4/T0CLI/C1OUT:

Có sơ đồ mạch như hình 5-8.
Có thể định cấu hình thực hiện 1 trong các chức năng sau:

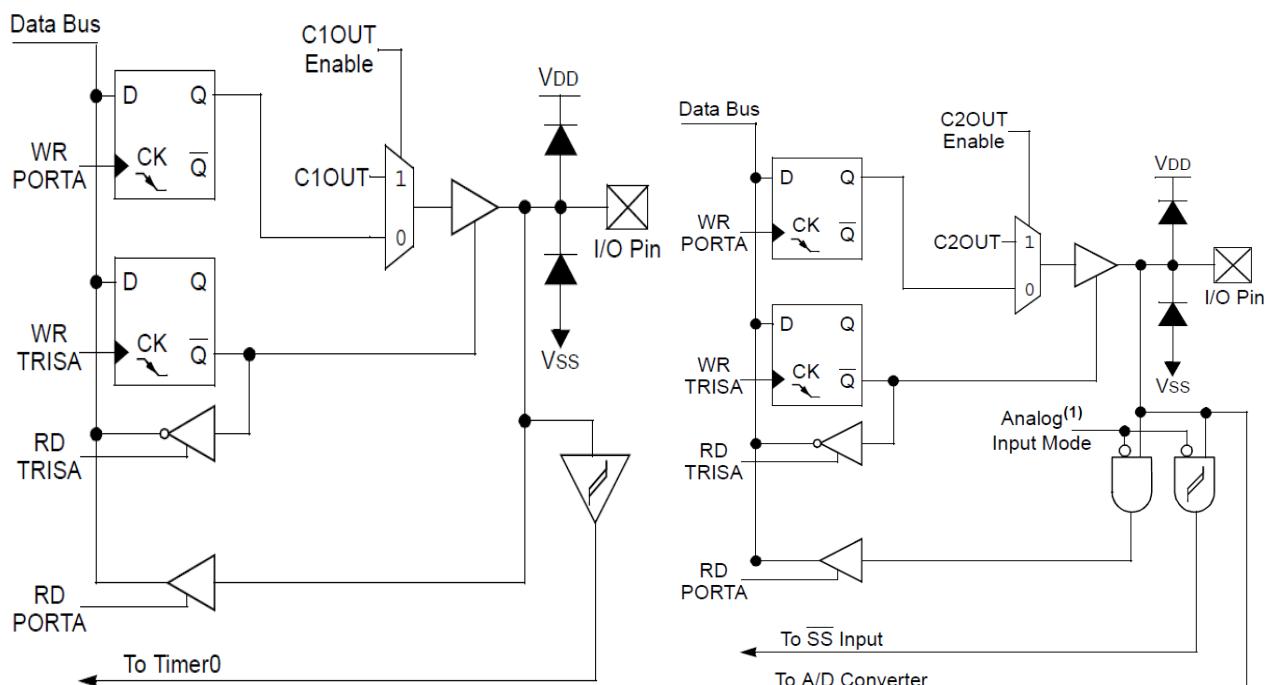
- Là tín hiệu xuất nhập số I/O.
- Là ngõ vào nhận xung ngoại của timer T0.
- Là ngõ ra của bộ so sánh C1.

Chân RA5/AN4/SS / C2OUT:

Có sơ đồ mạch như hình 5-9.
Có thể định cấu hình thực hiện 1 trong các chức năng sau:

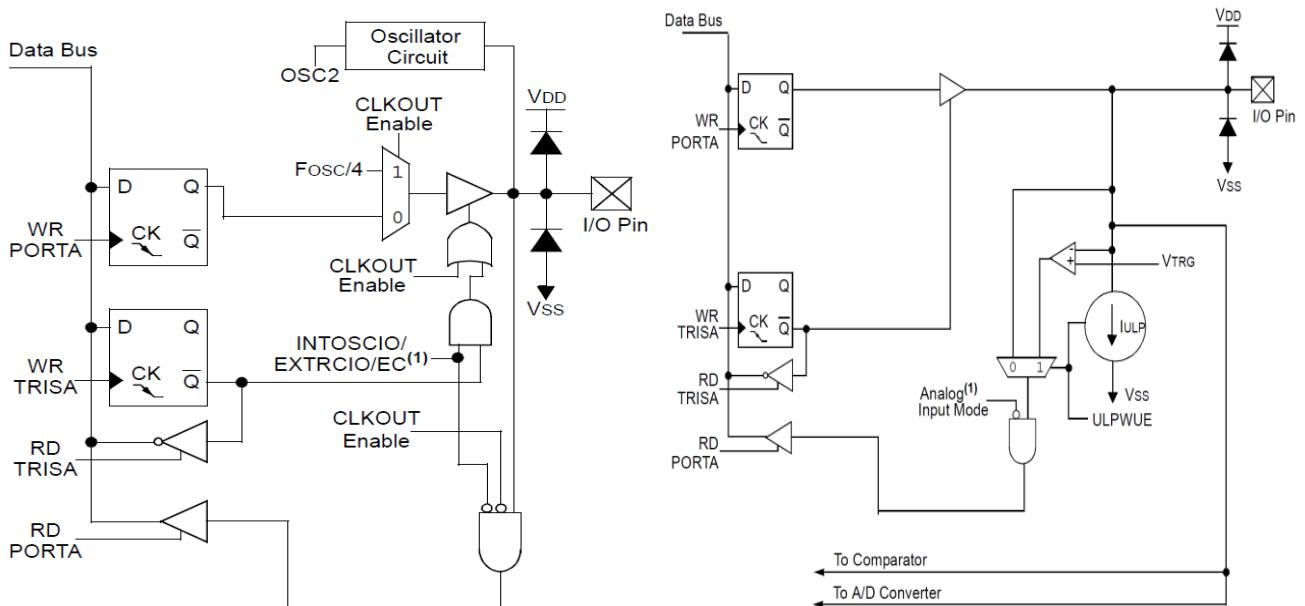
- Là tín hiệu xuất nhập số I/O.
- Là ngõ vào kênh thứ 4 của khối ADC.
- Là ngõ vào chọn tơ của truyền dữ liệu SPI.
- Là ngõ ra của bộ so sánh C2.

Chân RA6/OSC2/CLKOUT:	Chân RA7/OSC1/CLKIN:
<p>Có sơ đồ mạch như hình 5-10.</p> <p>Có thể định cấu hình thực hiện 1 trong các chức năng sau:</p> <ul style="list-style-type: none"> • Là tín hiệu xuất nhập số I/O. • Là nối với dao động thạch anh. • Là ngõ ra cấp xung clock. 	<p>Có sơ đồ mạch như hình 5-11.</p> <p>Có thể định cấu hình thực hiện 1 trong các chức năng sau:</p> <ul style="list-style-type: none"> • Là tín hiệu xuất nhập số I/O. • Là nối với dao động thạch anh. • Là ngõ vào nhận xung clock.



Hình 5-8. Cấu hình chân RA4.

Hình 5-9. Cấu hình chân R45.



Hình 5-10. Cấu hình chân RA6.

Hình 5-11. Cấu hình chân RA7.

5.3.2 PORTB VÀ THANH GHI TRISB

PortB là port hai chiều 8 bit, thanh ghi định hướng là TRISB có chức năng định cấu hình vào ra cho các bit xuất nhập của portB, bit định hướng bằng 0 thì port tương ứng xuất, bit bằng 1 thì bit tương ứng là nhập, portB và thanh ghi định hướng TRISB như hình 5-12:

	R/W(x)	R/W(x)	R/W(x)	R/W(x)	R/W(x)	R/W(x)	R/W(x)	R/W(x)
PORPB	RB7	RB6	RB5	RB4	RB3	RB2	RB1	RB0
TRISB	R/W(1)	R/W(1)	R/W(1)	R/W(1)	R/W(1)	R/W(1)	R/W(1)	R/W(1)
Bit 7	R/W(x): là cho phép đọc/ghi và khi reset thì không xác định						Bit 0	
Bit 7	R/W(1): là cho phép đọc/ghi và khi reset thì bằng 1						Bit 0	

Hình 5-12. PortB và thanh ghi định hướng port B.

Các chức năng tích hợp của portB

PortB có thêm chức năng báo ngắt khi portB có sự thay đổi tín hiệu và có điện trở kéo lên có thể lập trình.

Chức năng tương tự: PortB có đa hợp với các ngõ vào tương tự của khối ADC và tùy chọn tín hiệu số hay tương tự phụ thuộc vào thanh ghi ANSELH, nếu cho các bit trong thanh ghi ANSELH bằng 1 thì khi đó các bit đa hợp các kênh tương tự sẽ đóng vai trò là ngõ vào tương tự.

Thanh ghi ANSELH (analog select high register) có cấu trúc như hình 5-13. Các bit AN<13:8> trong thanh ghi ANSELH nếu bằng 1 thì chọn định cấu hình tương tự, bằng 0 thì định cấu hình số I/O.

U(0)	U(0)	R/W(1)	R/W(1)	R/W(1)	R/W(1)	R/W(1)	R/W(1)
-	-	ANS13	ANS12	ANS11	ANS10	ANS9	ANS8
Bit 7							Bit 0

Hình 5-13. Thanh ghi ANSELH định cấu hình số tương tự cho portB.

Chức năng điện trở kéo lên: mỗi bit của PortB đều có 1 điện trở kéo lên, 8 bit trong thanh ghi WPUB sẽ cho phép /cấm điện trở kéo lên. Khi port đóng vai trò là ngõ ra thì mạch tự động tắt chế độ điện trở kéo lên. Sau khi reset thì cấm điện trở kéo lên do xóa bit RBPU trong thanh ghi OPTION.

Thanh ghi WPUB (Weak Pull Up-Port B register) có cấu trúc như hình 5-14. Các bit WPUB<7:0> trong thanh ghi WPUB nếu bằng 1 thì cho phép điện trở kéo lên, bằng 0 thì không cho phép kéo lên.

| R/W(1) |
|--------|--------|--------|--------|--------|--------|--------|--------|
| WPUB7 | WPUB6 | WPUB5 | WPUB4 | WPUB3 | WPUB2 | WPUB1 | WPUB0 |
| Bit 7 | | | | | | | Bit 0 |

Hình 5-14. Thanh ghi WPUB thiết lập cho phép/cấm điện trở treo.

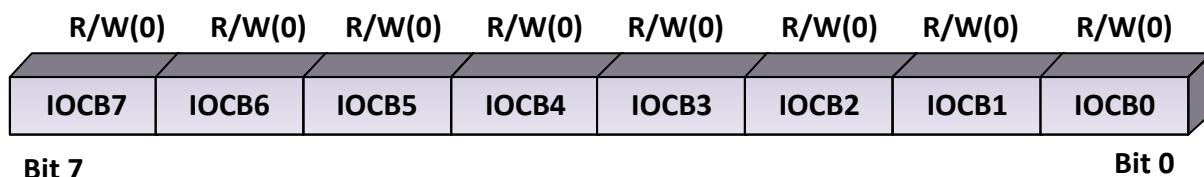
Chức năng ngắt khi có thay đổi: tất cả các bit của PortB đều có thể định cấu hình là ngõ vào ngắt khi có thay đổi trạng thái logic. 8 bit trong thanh ghi IOCB sẽ cho phép hay cấm chức năng này.

Nguyên lý thực hiện chức năng này như sau: giá trị hiện tại của portB được so sánh với giá trị trước đó đã chốt ở lần đọc sau cùng của portB để xác định bit nào đã thay đổi và báo ngắt nếu có sự thay đổi.

Ngắt này có thể đánh thức CPU khỏi chế độ ngủ. Người lập trình phải xóa ngắt này bằng cách thực hiện 2 yêu cầu sau:

- Đọc giá trị của portB thì điều kiện tương thích không còn xảy ra.
- Xóa cờ báo ngắt RBIF. Do giá trị chốt trước đó không bị ảnh hưởng bởi reset CPU nên sau khi reset thì cờ báo ngắt RBIF tiếp tục bằng 1 nếu vẫn còn sự không tương thích nếu không đọc portB.

Thanh ghi IOCB (Interrupt On-Change Port B register) có cấu trúc như hình 5-15. Các bit IOCB<7:0> trong thanh ghi IOCB nếu bằng 1 thì cho phép ngắt, bằng 0 thì cấm.

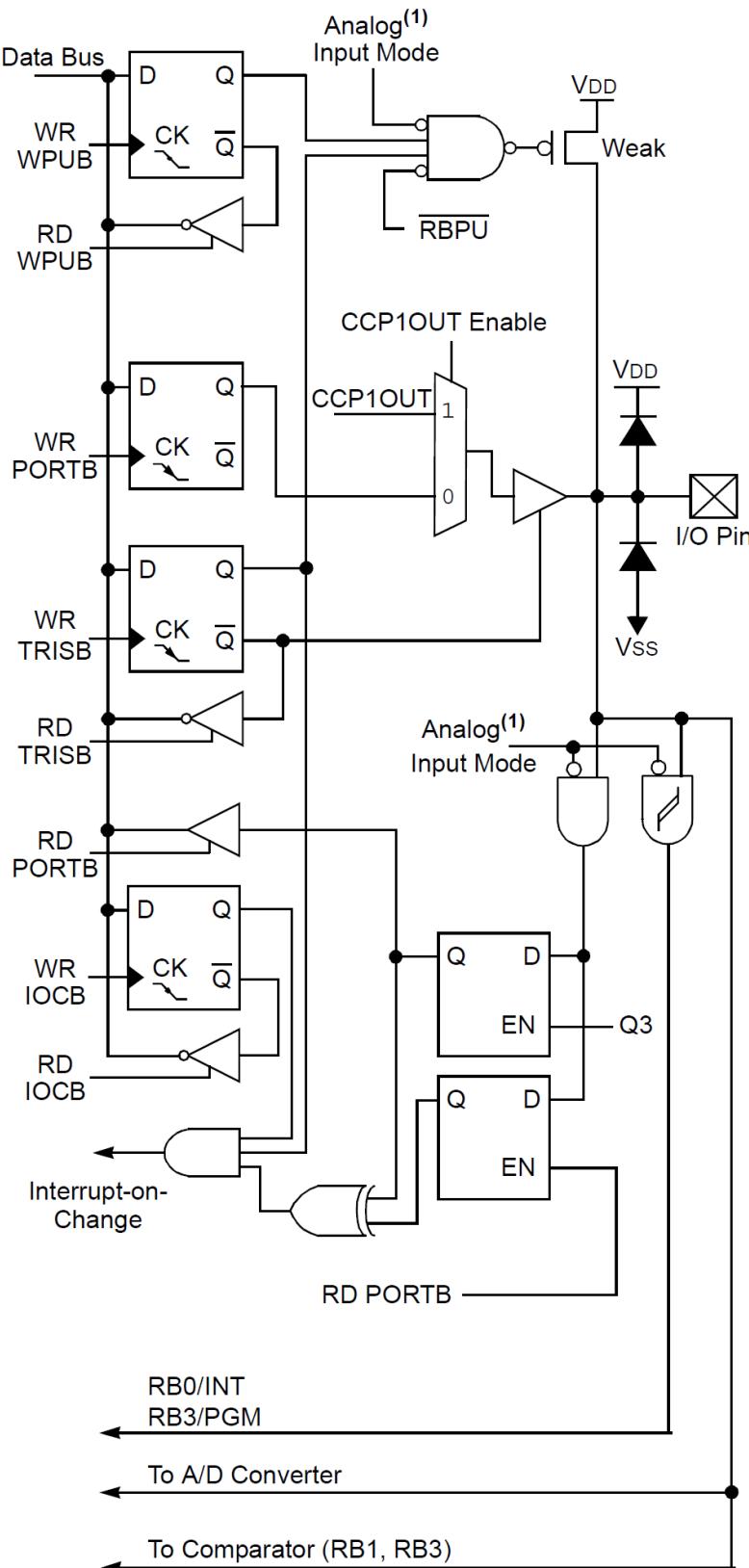


Hình 5-15. Thanh ghi IOCB cho phép/cấm ngắt portB thay đổi.

Khảo sát cấu hình mạch cho từng tín hiệu.

Chân RB0/AN12/INT:	Chân RB1/AN10/P1C/C12IN3-:
<p>Có sơ đồ mạch như hình 5-16.</p> <p>Có thể định cấu hình thực hiện 1 trong các chức năng sau:</p> <ul style="list-style-type: none"> Là tín hiệu xuất nhập số I/O. Là ngõ vào kênh thứ 12 của khói ADC. Là ngõ vào ngắt ngoài INT tác động bằng cạnh. 	<p>Có sơ đồ mạch như hình 5-16.</p> <p>Có thể định cấu hình thực hiện 1 trong các chức năng sau:</p> <ul style="list-style-type: none"> Là tín hiệu xuất nhập số I/O. Là ngõ vào kênh thứ 10 của khói ADC. Là ngõ ra PWM (chỉ có ở PIC16F882/883/886). Là ngõ vào âm thứ 3 của bộ so sánh C1, C2.

Chân RB2/AN8/P1B:	Chân RB3/AN9/PGM/C12IN2-:
<p>Có sơ đồ mạch như hình 5-16.</p> <p>Có thể định cấu hình thực hiện 1 trong các chức năng sau:</p> <ul style="list-style-type: none"> Là tín hiệu xuất nhập số I/O. Là ngõ vào kênh thứ 10 của khói ADC. Là ngõ ra PWM (chỉ có ở PIC16F882/883/886). 	<p>Có sơ đồ mạch như hình 5-16.</p> <p>Có thể định cấu hình thực hiện 1 trong các chức năng sau:</p> <ul style="list-style-type: none"> Là tín hiệu xuất nhập số I/O. Là ngõ vào kênh thứ 9 của khói ADC. Là ngõ vào cho phép lập trình nối tiếp điện áp thấp. Là ngõ vào âm thứ 2 của bộ so sánh C1, C2.

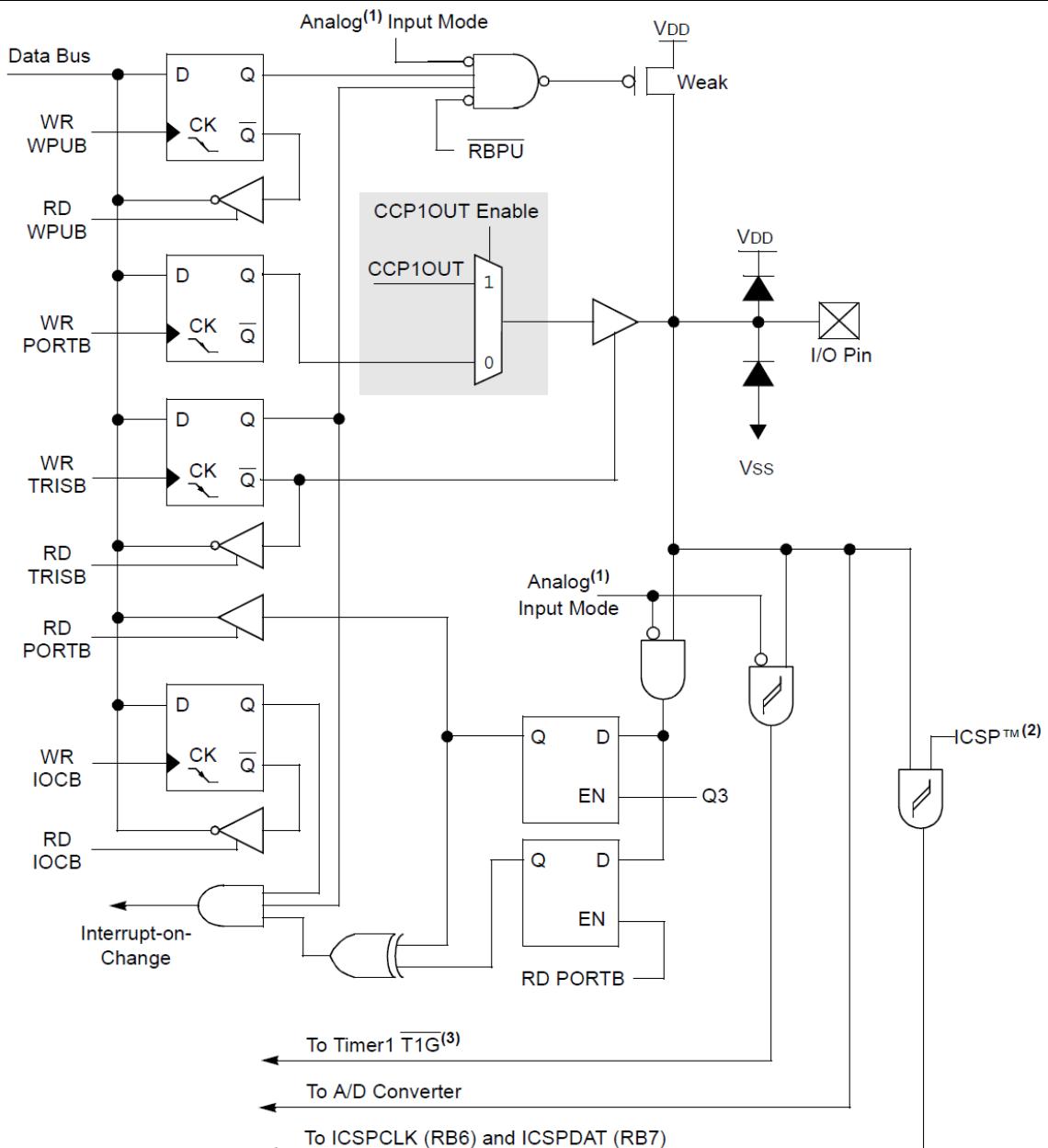


Hình 5-16. Cấu hình chân RB<3:0>.

Chân RB4/AN11/P1D:	Chân RB5/AN13/T1̄G:
<p>Có sơ đồ mạch như hình 5-17.</p> <p>Có thể định cấu hình thực hiện 1 trong các chức năng sau:</p> <ul style="list-style-type: none"> Là tín hiệu xuất nhập số I/O. Là ngõ vào kênh thứ 11 của khối ADC. 	<p>Có sơ đồ mạch như hình 5-17.</p> <p>Có thể định cấu hình thực hiện 1 trong các chức năng sau:</p> <ul style="list-style-type: none"> Là tín hiệu xuất nhập số I/O. Là ngõ vào kênh thứ 13 của khối ADC.

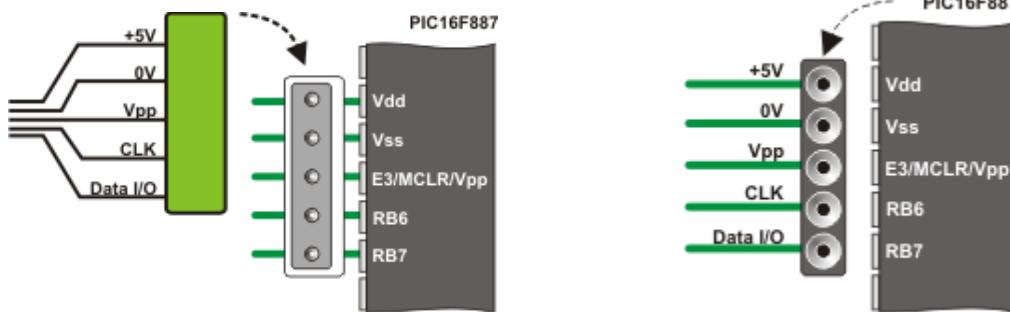
- | | |
|---|-------------------------------|
| • Là ngõ ra PWM (chỉ có ở PIC16F882/883/886). | • Là ngõ vào công của Timer1. |
|---|-------------------------------|

Chân RB6/ICSPCLK:	Chân RB7/ICSPDAT:
<p>Có sơ đồ mạch như hình 5-17.</p> <p>Có thể định cấu hình thực hiện 1 trong các chức năng sau:</p> <ul style="list-style-type: none"> • Là tín hiệu xuất nhập số I/O. • Là ngõ vào nhận xung clock lập trình nối tiếp cho bộ nhớ chương trình. 	<p>Có sơ đồ mạch như hình 5-17.</p> <p>Có thể định cấu hình thực hiện 1 trong các chức năng sau:</p> <ul style="list-style-type: none"> • Là tín hiệu xuất nhập số I/O. • Là ngõ dữ liệu lập trình nối tiếp cho bộ nhớ chương trình.



Hình 5-17. Cấu hình chân RB<7:4>.

Ba chân RB3/PGM, RB6/ **ICSPCLK** và RB7/ **ICSPDAT** được đa hợp với mạch điện gỡ rói bên trong và mạnh lập trình để nạp chương trình vào bộ nhớ nội. Sơ đồ kết nối mạch nạp và mạch gỡ rói:



Hình 5-18. Các chân PortB giao tiếp với mạch nạp, gỡ rói.

5.3.3 PORTC VÀ THANH GHI TRISC

PortC là port hai chiều 8 bit, thanh ghi định hướng là TRISC có chức năng định cấu hình vào ra cho các bit xuất nhập của portC, bit định hướng bằng 0 thì port tương ứng xuất, bit bằng 1 thì bit tương ứng là nhập, portC và thanh ghi định hướng TRISC như hình 5-19:

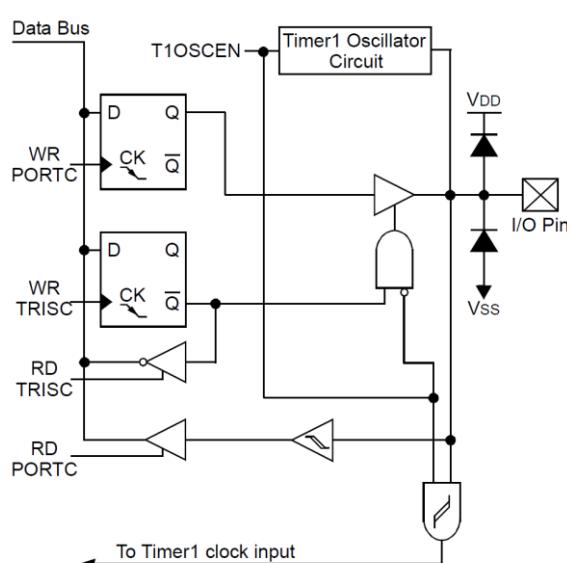
	R/W(x)	R/W(x)	R/W(x)	R/W(x)	R/W(x)	R/W(x)	R/W(x)	R/W(x)
PORTC	RC7	RC6	RC5	RC4	RC3	RC2	RC1	RC0
Bit 7	R/W(x): là cho phép đọc/ghi và khi reset thì không xác định						Bit 0	
TRISC	TRISC7	TRISC6	TRISC5	TRISC4	TRISC3	TRISC2	TRISC1	TRISCO
Bit 7	R/W(1): là cho phép đọc/ghi và khi reset thì bằng 1						Bit 0	

Hình 5-19. Port C và thanh ghi TRISC.

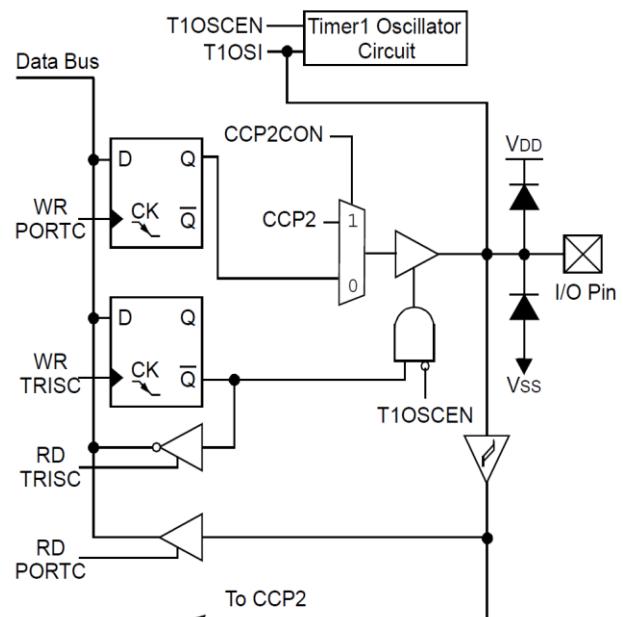
Khảo sát cấu hình mạch cho từng tín hiệu.

Chân RC0/T1OSO/T1CKI:	Chân RC1/T1OSI/CCP2:
Có sơ đồ mạch như hình 5-20. Có thể định cấu hình thực hiện 1 trong các chức năng sau: <ul style="list-style-type: none"> Là tín hiệu xuất nhập số I/O. Là ngõ ra của bộ dao động Timer1. Là ngõ vào nhận xung ngoại của Timer1. 	Có sơ đồ mạch như hình 5-21. Có thể định cấu hình thực hiện 1 trong các chức năng sau: <ul style="list-style-type: none"> Là tín hiệu xuất nhập số I/O. Là ngõ vào của bộ dao động Timer1. Là ngõ vào của Capture và ngõ ra của Compare/PWM của bộ so sánh C2.

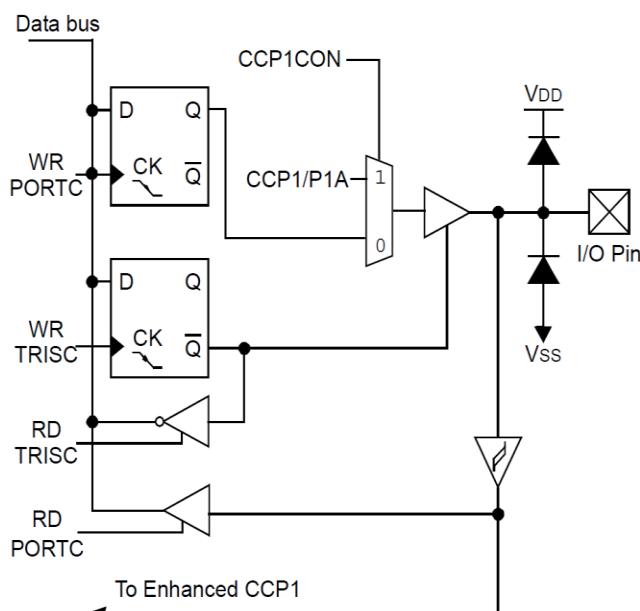
Chân RC2/PIA/CCP1:	Chân RC3/SCK/SCL:
Có sơ đồ mạch như hình 5-22. Có thể định cấu hình thực hiện 1 trong các chức năng sau: <ul style="list-style-type: none"> Là tín hiệu xuất nhập số I/O. Là ngõ ra của PWM. Là ngõ vào của Capture và ngõ ra của Compare của bộ so sánh C1. 	Có sơ đồ mạch như hình 5-23. Có thể định cấu hình thực hiện 1 trong các chức năng sau: <ul style="list-style-type: none"> Là tín hiệu xuất nhập số I/O. Là ngõ cấp xung cho truyền dữ liệu SPI. Là ngõ cấp xung cho truyền dữ liệu I2C.



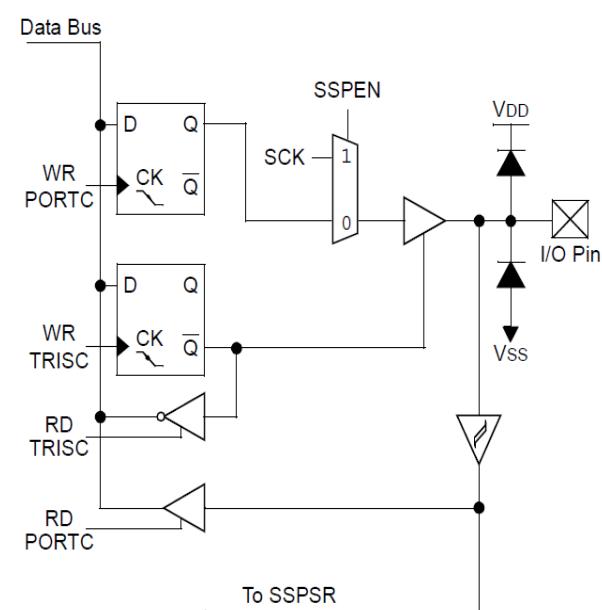
Hình 5-20. Cấu hình chân RC0.



Hình 5-21. Cấu hình chân RC1.



Hình 5-22. Cấu hình chân RC2.



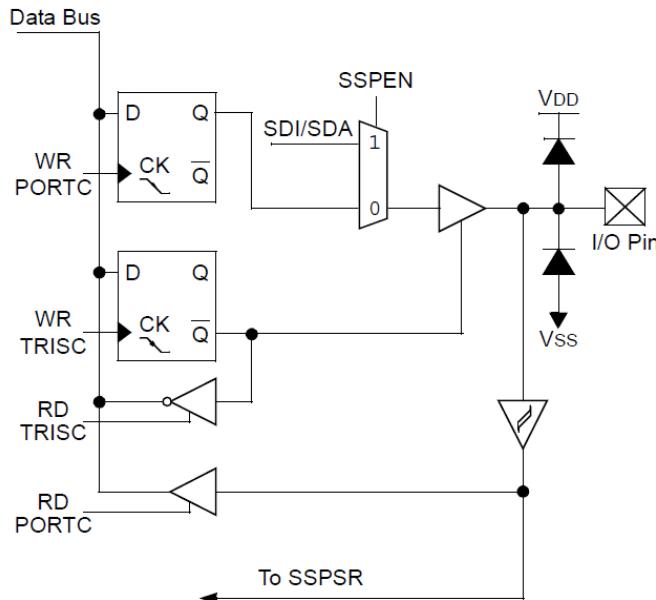
Hình 5-23. Cấu hình chân RC3.

Chân RC4/SDI/SDA:	Chân RC5/SDO:
<p>Có sơ đồ mạch như hình 5-24.</p> <p>Có thể định cấu hình thực hiện 1 trong các chức năng sau:</p> <ul style="list-style-type: none"> Là tín hiệu xuất nhập số I/O. Là ngõ vào dữ liệu cho truyền dữ liệu SPI. Là ngõ dữ liệu cho truyền dữ liệu I2C. 	<p>Có sơ đồ mạch như hình 5-25.</p> <p>Có thể định cấu hình thực hiện 1 trong các chức năng sau:</p> <ul style="list-style-type: none"> Là tín hiệu xuất nhập số I/O. Là ngõ ra dữ liệu cho truyền dữ liệu SPI.

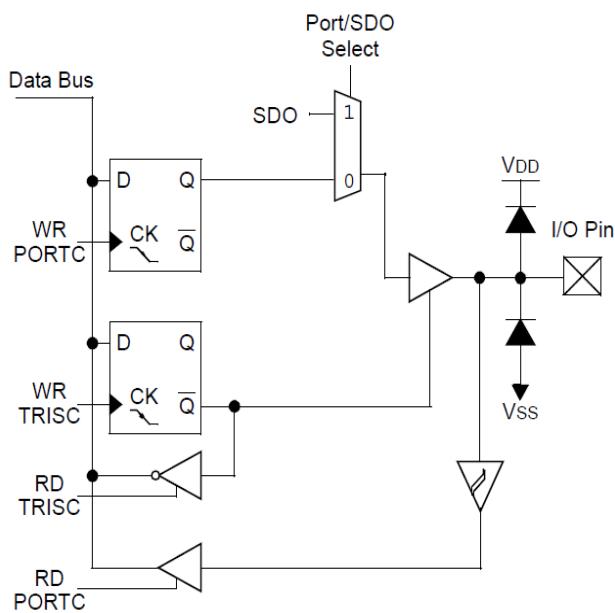
Chân RC6/TX/CK:	Chân RC7/RX/DT:
<p>Có sơ đồ mạch như hình 5-26.</p> <p>Có thể định cấu hình thực hiện 1 trong các chức năng sau:</p>	<p>Có sơ đồ mạch như hình 5-27.</p> <p>Có thể định cấu hình thực hiện 1 trong các chức năng sau:</p>

- Là tín hiệu xuất nhập số I/O.
- Là ngõ ra phát dữ liệu cho truyền dữ liệu nối tiếp bất đồng bộ.
- Là ngõ vào ra cho truyền dữ liệu nối tiếp đồng bộ.

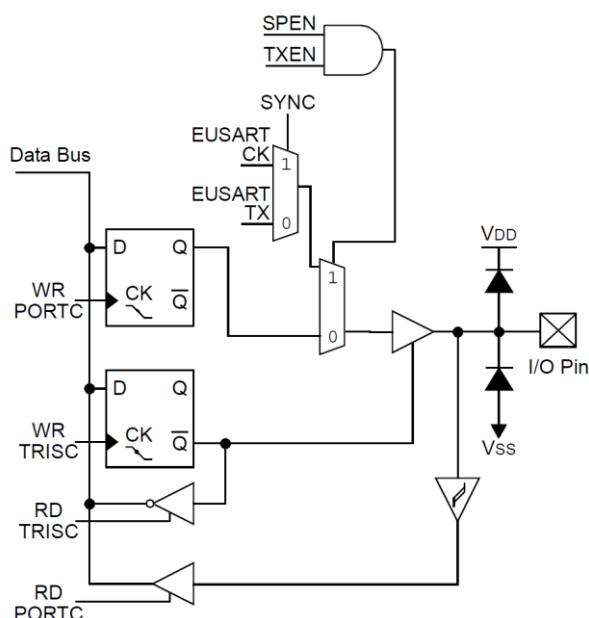
- Là tín hiệu xuất nhập số I/O.
- Là ngõ vào nhận dữ liệu cho truyền dữ liệu nối tiếp bất đồng bộ.
- Là ngõ ra cấp xung clock cho truyền dữ liệu nối tiếp đồng bộ.



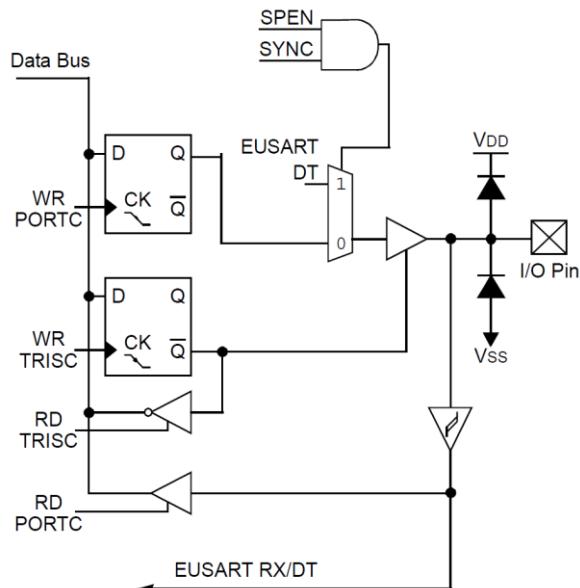
Hình 5-24. Cấu hình chân RC4.



Hình 5-25. Cấu hình chân RC5.



Hình 5-26. Cấu hình chân RC6.



Hình 5-27. Cấu hình chân RC7.

5.3.4 PORTD VÀ THANH GHI ĐỊNH HƯỚNG TRISD

PortD là port hai chiều 8 bit, thanh ghi định hướng là TRISD có chức năng định cấu hình vào ra cho các bit xuất nhập của portD, bit định hướng bằng 0 thì port tương ứng xuất, bit bằng 1 thì bit tương ứng là nhập, portD và thanh ghi định hướng TRISD như hình 5-28.

Khảo sát cấu hình mạch cho từng tín hiệu.

Chân RD<4 :0>

Sơ đồ mạch điện của các chân từ RC0 như hình 5-29, chỉ có chức năng là xuất nhập số.

	R/W(x)	R/W(x)	R/W(x)	R/W(x)	R/W(x)	R/W(x)	R/W(x)	R/W(x)
PORTE	RD7	RD6	RD5	RD4	RD3	RD2	RD1	RD0
TRISD	R/W(1)	R/W(1)	R/W(1)	R/W(1)	R/W(1)	R/W(1)	R/W(1)	R/W(1)
Bit 7	R/W(x): là cho phép đọc/ghi và khi reset thì không xác định						Bit 0	
Bit 7	R/W(1): là cho phép đọc/ghi và khi reset thì bằng 1						Bit 0	

Hình 5-28. Port D và thanh ghi TRISD.

Chân RD5/P1B

Sơ đồ mạch điện của các chân từ RD5 như hình 5-30. Chân này được cấu hình thực hiện 1 trong các chức năng sau:

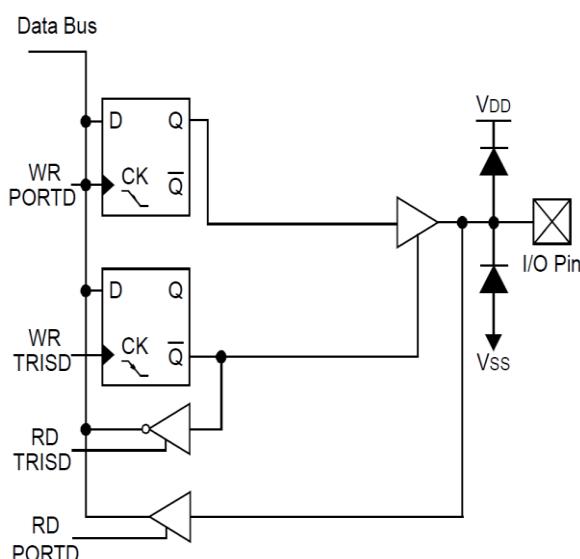
- Là tín hiệu xuất nhập số I/O.
- Là ngõ ra của PWM.

Chân RD6/P1C

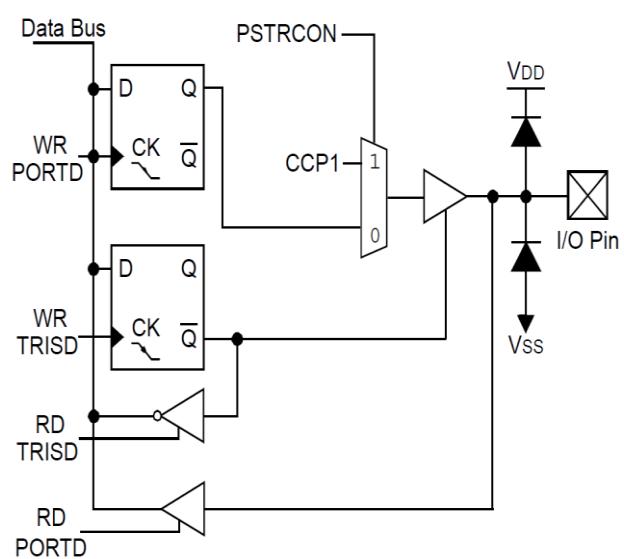
Sơ đồ mạch điện của các chân từ RD6 như hình 5-30.

Chân này được cấu hình thực hiện 1 trong các chức năng sau:

- Là tín hiệu xuất nhập số I/O.
- Là ngõ ra của PWM (chỉ có ở PIC16F884/887).



Hình 5-29. Cấu hình chân RD<4:0>.



Hình 5-30. Cấu hình chân RD<7:5>.

Chân RD7/P1D

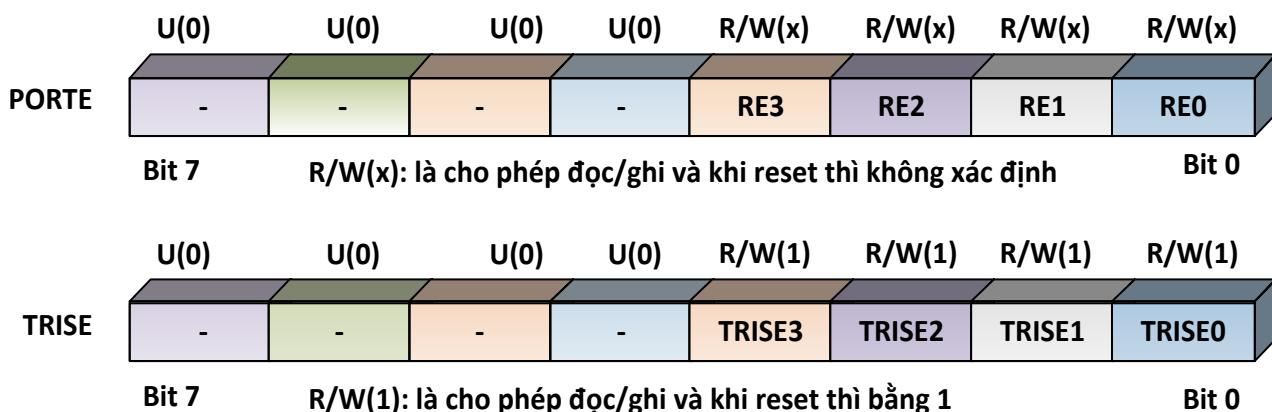
Sơ đồ mạch điện của các chân từ RD7 như hình 5-30.

Chân này được cấu hình thực hiện 1 trong các chức năng sau:

- Là tín hiệu xuất nhập số I/O.
 - Là ngõ ra của PWM (chỉ có ở PIC16F884/887).

5.3.5 PORTE VÀ THANH GHI TRISE

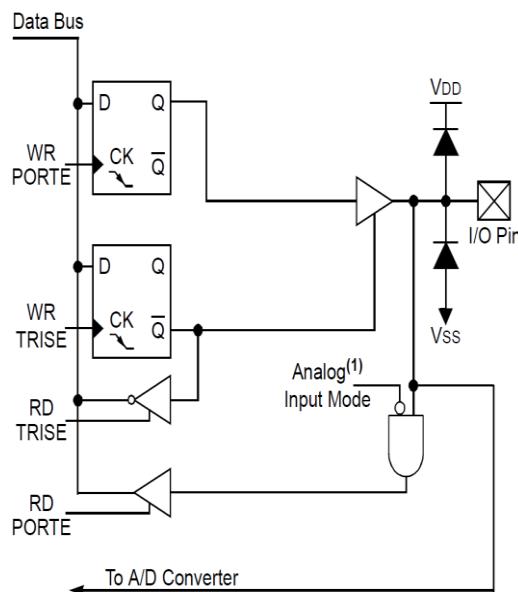
PortE là port hai chiều 4 bit, thanh ghi định hướng là TRISE có chức năng định cấu hình vào ra cho các bit xuất nhập của portE, bit định hướng bằng 0 thì port tương ứng xuất, bit bằng 1 thì bit tương ứng là nhập, portE và thanh ghi định hướng TRISE như hình 5-31.



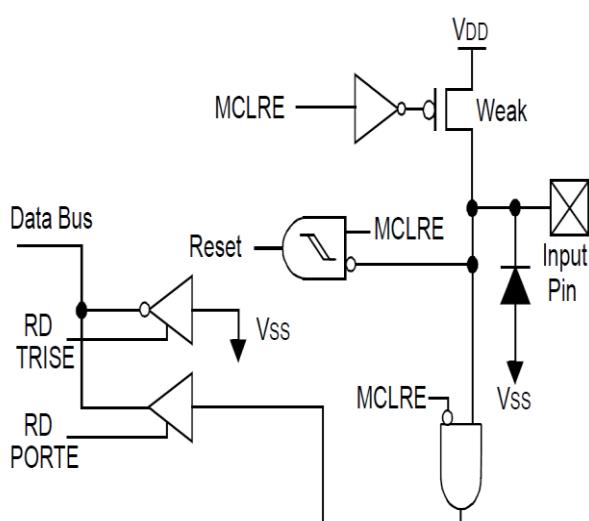
Hình 5-31. Port E và thanh ghi TRISE.

Khảo sát cấu hình mạch cho từng tín hiệu.

Chân RE0/AN5:	Chân RE1/AN6:
<p>Có sơ đồ mạch như hình 5-32.</p> <p>Có thể định cấu hình thực hiện 1 trong các chức năng sau:</p> <ul style="list-style-type: none"> • Là tín hiệu xuất nhập số I/O. • Là ngõ vào kênh tương tự thứ 5 của ADC. 	<p>Có sơ đồ mạch như hình 5-32.</p> <p>Có thể định cấu hình thực hiện 1 trong các chức năng sau:</p> <ul style="list-style-type: none"> • Là tín hiệu xuất nhập số I/O. • Là ngõ vào kênh tương tự thứ 6 của ADC.



Hình 5-32. Cấu hình chân RE<2:0>.



Hình 5-33. Cấu hình chân RE<3>.

Chân RE2/AN7: *Chân RE3/ \overline{MCLR} /V_{PP}:*

Có sơ đồ mạch như hình 5-32. Có thể định cấu hình thực hiện 1 trong các chức năng sau: <ul style="list-style-type: none"> • Là tín hiệu xuất nhập số I/O. • Là ngõ vào kênh tương tự thứ 7 của ADC. 	Có sơ đồ mạch như hình 5-33. Có thể định cấu hình thực hiện 1 trong các chức năng sau: <ul style="list-style-type: none"> • Là tín hiệu xuất nhập số I/O tùy thuộc vào bit chọn reset trong thanh ghi định cấu hình. • Là ngõ vào reset CPU. • Là ngõ nhận điện áp lập trình Vpp.
--	---

5.4 LỆNH TRUY XUẤT PORT DÙNG NGÔN NGỮ CCS-C

Các lệnh của ngôn ngữ lập trình C liên quan đến các port bao gồm:

- Lệnh OUTPUT_FLOAT(pin): có chức năng thả nỗi chân của port.
- **Lệnh OUTPUT_LOW(pin):** có chức năng cho 1 chân của port xuống mức 0.
- **Lệnh OUTPUT_HIGH(pin):** có chức năng cho 1 chân của port lên mức 1.
- Lệnh OUTPUT_TOGGLE(pin): có chức năng đảo trạng thái 1 chân của port.
- **Lệnh OUTPUT_BIT (pin, value):** có chức năng xuất giá trị value ra 1 chân của port.
- **Lệnh OUTPUT_X(value):** có chức năng xuất dữ liệu 8 bit ra portX.
- **Lệnh SET_TRIS_X(value):** có chức năng định cấu hình cho portX.
- Lệnh GET_TRIS_X(): có chức năng đọc giá trị đã định cấu hình cho port gán cho biến.
- Lệnh INPUT_X(): có chức năng đọc giá trị của port gán cho biến.
- **Lệnh INPUT(pin):** có chức năng đọc giá trị 1 chân của port gán cho biến.
- Lệnh PORT_A_PULLUPS(value): có chức năng treo điện trở của portA.
- Lệnh PORT_B_PULLUPS(value): có chức năng treo điện trở của portB.
- Lệnh INPUT_STATE(pin): có chức năng đọc trạng thái 1 chân của port.
- Lệnh OUTPUT_DRIVE(pin): có chức năng định cấu hình cho 1 chân của port.

5.4.1 LỆNH SET_TRIS_X()

Cú pháp: set_tris_x(value); x là port A, B, C, D, E

Thông số: value là 1 số nguyên 8 bit tương ứng với các bit của port I/O.

Chức năng: định hướng cho các port I/O (TRI-State) là vào hay ra. Mỗi bit tương ứng 1 chân. Mức 1 thì chân tương ứng là ngõ vào, mức 0 là ngõ ra.

Có hiệu lực: cho tất cả các vi điều khiển PIC.

Ví dụ 5-1: SET_TRIS_B (0x0F); // 0F=00001111: B7- B4 là ngõ ra, B3-B0 là ngõ vào.

5.4.2 LỆNH OUTPUT_X(VALUE)

Cú pháp: output_x (value)

- Thông số: value là hằng số 8 bit kiểu int.
 Chức năng: Xuất dữ liệu 1 byte ra portx.
 Có hiệu lực: Lệnh này áp dụng cho tất cả các port.
Ví dụ 5-1: OUTPUT_B(0xF0); xuất dữ liệu F0 ra portB.

5.4.3 LỆNH OUTPUT_HIGH(PIN)

- Cú pháp: output_high(pin); tương đương lệnh BSF PORTX, B
 Thông số: pin là chân xuất dữ liệu - hãy xem file định nghĩa của thiết bị "device.h" để biết tên chính xác.
 Chức năng: làm 1 chân của port lên mức cao.
 Có hiệu lực: lệnh này áp dụng cho tất cả các thiết bị.
Ví dụ 5-3: OUTPUT_HIGH(PIN_A0); // làm cho chân RA0 của port A lên 1

5.4.4 LỆNH OUTPUT_LOW(PIN)

- Cú pháp: output_low(pin); tương đương lệnh BCF PORTX, B
 Thông số: pin là chân xuất dữ liệu - hãy xem file định nghĩa của thiết bị "device.h" để biết tên chính xác.
 Chức năng: làm 1 chân của port xuống mức thấp.
 Có hiệu lực: lệnh này áp dụng cho tất cả các thiết bị.
Ví dụ 5-4: output_low(pin_a0); // làm cho chân RA0 của PortA xuống mức 0

5.4.5 LỆNH OUTPUT_TOGGLE(PIN)

- Cú pháp: output_toggle(pin);
 Thông số: pin là chân xuất dữ liệu - hãy xem file định nghĩa của thiết bị "device.h" để biết tên chính xác.
 Chức năng: làm đảo trạng thái 1 chân của port.
 Có hiệu lực: lệnh này áp dụng cho tất cả các thiết bị.
Ví dụ 5-5: OUTPUT_TOGGLE(PIN_B0); // đảo trạng thái chân RB0 của port B

5.4.6 LỆNH OUTPUT_BIT(PIN,VALUE)

- Cú pháp: output_bit (pin, value);
 Thông số: pin là chân xuất dữ liệu - hãy xem file định nghĩa của thiết bị "device.h" để biết tên chính xác.
 Chức năng: xuất dữ liệu 0 hoặc 1 ra 1 chân của port.
 Có hiệu lực: lệnh này áp dụng cho tất cả các thiết bị.
Ví dụ 5-6: output_bit(pin_b0,0); // xuất dữ liệu 0 ra RB0

5.4.7 LỆNH value = GET_TRIS_X()

- Cú pháp: value = get_tris_x();
 Thông số: không có thông số
 Kết quả trả về: là byte dữ liệu đã định cấu hình từ thanh ghi TRIS
 Chức năng: kết quả trả về là giá trị của thanh ghi TRIS của các port A, B, C or D.
 Có hiệu lực: lệnh này áp dụng cho tất cả các thiết bị.

5.4.8 LỆNH value = INPUT(pin)

- Cú pháp: value = input(pin);
 Thông số: pin là chân để đọc - hãy xem file định nghĩa của thiết bị "device.h" để biết tên chính xác

- Kết quả trả về: 0 (or FALSE) nếu chân ở mức thấp, 1 (or TRUE) nếu chân ở mức cao.
- Chức năng: đọc dữ liệu 1 bit từ 1 chân của port, chân này phải ở cấu hình là chân vào.
- Có hiệu lực: lệnh này áp dụng cho tất cả các thiết bị.
- Ví dụ 5-7:** while (! Input (PIN_B1)); //đợi cho đến khi chân RB1 lên mức cao

5.4.9 LỆNH INPUT_STATE()

- Cú pháp: value = input_state(**pin**);
- Thông số: pin là chân để đọc - hãy xem file định nghĩa của thiết bị "device.h" để biết tên chính xác
- Trả về: kết quả đọc bằng 1 nếu chân đọc ở mức cao, kết quả đọc bằng 0 nếu chân đọc ở mức thấp.
- Chức năng: lệnh đọc mức logic của 1 chân nhưng không làm thay đổi hướng của chân.
- Có hiệu lực: lệnh này áp dụng cho tất cả các thiết bị.
- Ví dụ 5-8:** level = input_state(pin_A3);

5.4.10 Value = INPUT_X()

- Cú pháp: value = input_x();
- Thông số: không có.
- Kết quả trả về: là dữ liệu 8 bit của portx.
- Chức năng: lệnh đọc mức logic của 1 chân nhưng không làm thay đổi hướng của chân.
- Có hiệu lực: lệnh này áp dụng cho tất cả các thiết bị.

5.4.11 LỆNH OUTPUT_DRIVE(PIN)

- Cú pháp: output_drive(**pin**);
- Thông số: pin là chân được định nghĩa trong file "device.h".
- Trả về: không có.
- Chức năng: thiết lập chân (pin) là chế độ xuất.
- Có hiệu lực: lệnh này áp dụng cho tất cả các thiết bị.
- Ví dụ 5-10:** output (pin_A0);

5.4.12 LỆNH OUTPUT_FLOAT(PIN)

- Cú pháp: output_float(**pin**);
- Thông số: pin là chân được định nghĩa trong file "device.h".
- Trả về: không có.
- Chức năng: thiết lập chân (pin) là chế độ nhập và thả nỗi chân tín hiệu này để thiết bị khác ở bên ngoài toàn quyền điều khiển chân này để đưa dữ liệu vào vi điều khiển.
- Có hiệu lực: lệnh này áp dụng cho tất cả các thiết bị.

Ví dụ 5-11:

```
if ((data & 0x80) == 0)
    Output_low (pin_A0);
else
    output_float (pin_A0);
```

5.4.13 LỆNH PORT_B_PULLUP()

- Cú pháp: port_b_pullup(**value**);

- Thông số: value có 2 giá trị là true và false.
- Trả về: không có.
- Chức năng: thiết lập port B treo lên nguồn qua điện trở kéo lên bên trong. Nếu value là true thì treo lên nguồn, nếu là false thì không treo.
- Có hiệu lực: lệnh này áp dụng cho tất cả các thiết bị.
- Ví dụ 5-12:** port_b_pullup(false);

5.5 CẤU HÌNH ĐẶC BIỆT CỦA CPU

Vi điều khiển PIC 16F887 có một loạt các tính năng nhằm tối đa hóa độ tin cậy của hệ thống, giảm thiểu chi phí thông qua việc loại bỏ các thành phần bên ngoài, cung cấp các tính năng tiết kiệm công suất và bảo vệ code đã lập trình trong bộ nhớ.

Các cấu trúc bao gồm:

- ◆ Reset
 - Reset khi có điện – POR (Power On Reset)
 - Bộ định thời khi có điện – PWRT (Power Up Timer)
 - Bộ định thời cho bộ dao động lúc bắt đầu – OST (Oscillator Start Up Timer)
 - Reset khi sụt giảm nguồn cung cấp – BOR (Brown Out Reset)
- ◆ Các ngắt
 - ◆ Bộ định thời giám sát – WDT (Watch Dog Timer)
 - ◆ Lựa chọn bộ dao động
 - ◆ Chế độ ngủ
 - ◆ Bảo vệ code
 - ◆ Xác định mã nhận dạng chip
 - ◆ Chọn chế độ lập trình nối tiếp

5.5.1 CẤU HÌNH RESET CPU

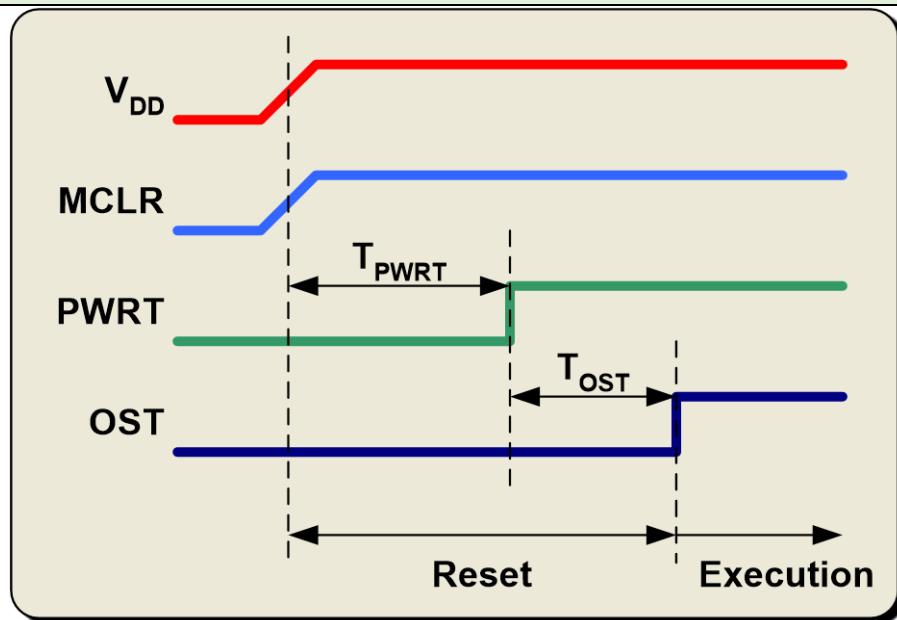
Reset cứng của CPU là chân MCLR, khi có điện thì tín hiệu reset này cũng chuyển từ thấp lên cao để cho CPU bắt đầu làm việc. Xem hình 5-34.

Vi điều khiển PIC 16F887 có 2 bộ định thời để kéo dài thời gian cần thiết khi mới có nguồn điện cung cấp cho CPU.

Nếu sử dụng thêm bộ định thời khi có điện PWRT thì sẽ kéo dài thời gian reset CPU thêm 64ms tính từ khi bắt đầu có điện cung cấp cho CPU với mục đích nhằm chờ nguồn điện cung cấp cho CPU ổn định. Xem hình 5-34.

Nếu sử dụng thêm bộ định thời cho bộ dao động lúc bắt đầu OST thì sẽ kéo dài thêm thời gian reset CPU với mục đích nhằm chờ bộ dao động hoạt động ổn định. Xem hình 5-34.

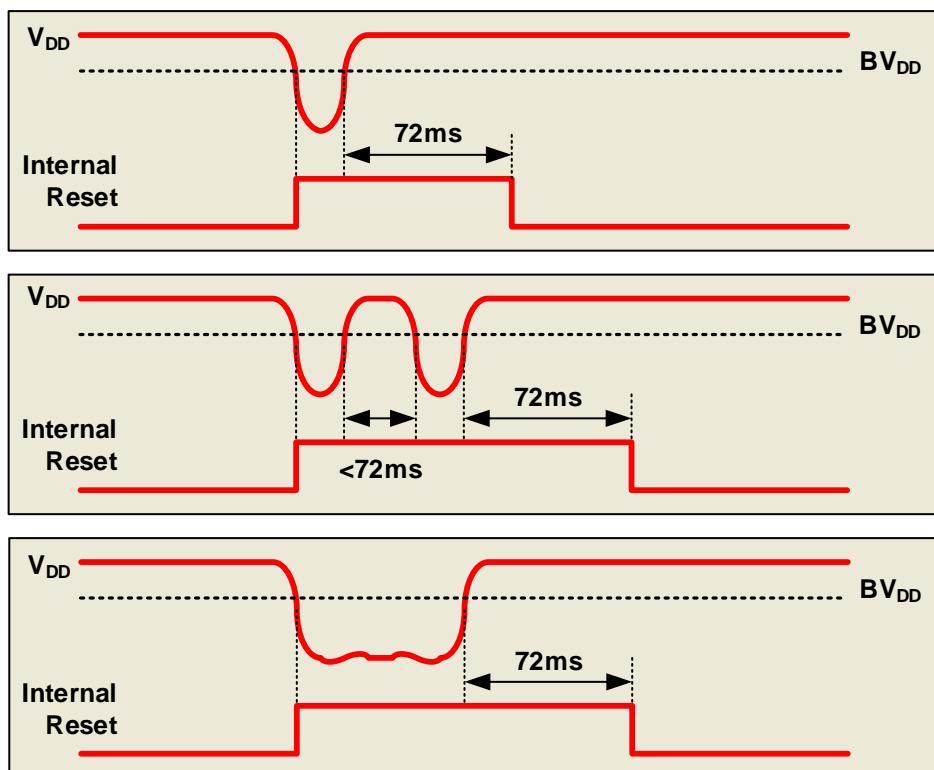
BOR xảy ra khi CPU đang hoạt động thì nguồn bị sụt giảm thấp hơn ngưỡng điện áp bằng BV_{DD} thì sẽ xảy ra hiện tượng reset, tín hiệu reset bên trong sẽ lên mức cao khi nguồn sụt giảm thấp hơn BV_{DD} và tiếp tục ở mức cao và nếu nguồn trở lại đúng điện áp V_{DD} thì tín hiệu reset ở bên trong kéo dài thêm 64ms. Hình 5-35 trình bày các dạng BOR.



Hình 5-34. Các dạng tín hiệu của nguồn, MCLR, PWRT, OST.

Điện áp BV_{DD} là 1 trong 4 giá trị: 2.5V, 2.7V, 4.2V và 4.5V.

Các dạng reset được lựa chọn bởi các bit trong thanh ghi cấu hình.



Hình 5-35. Các dạng tín hiệu khi bị sụt giảm nguồn BOR.

5.5.2 CÁU HÌNH CÁC NGẮT ĐÁNH THỨC CPU

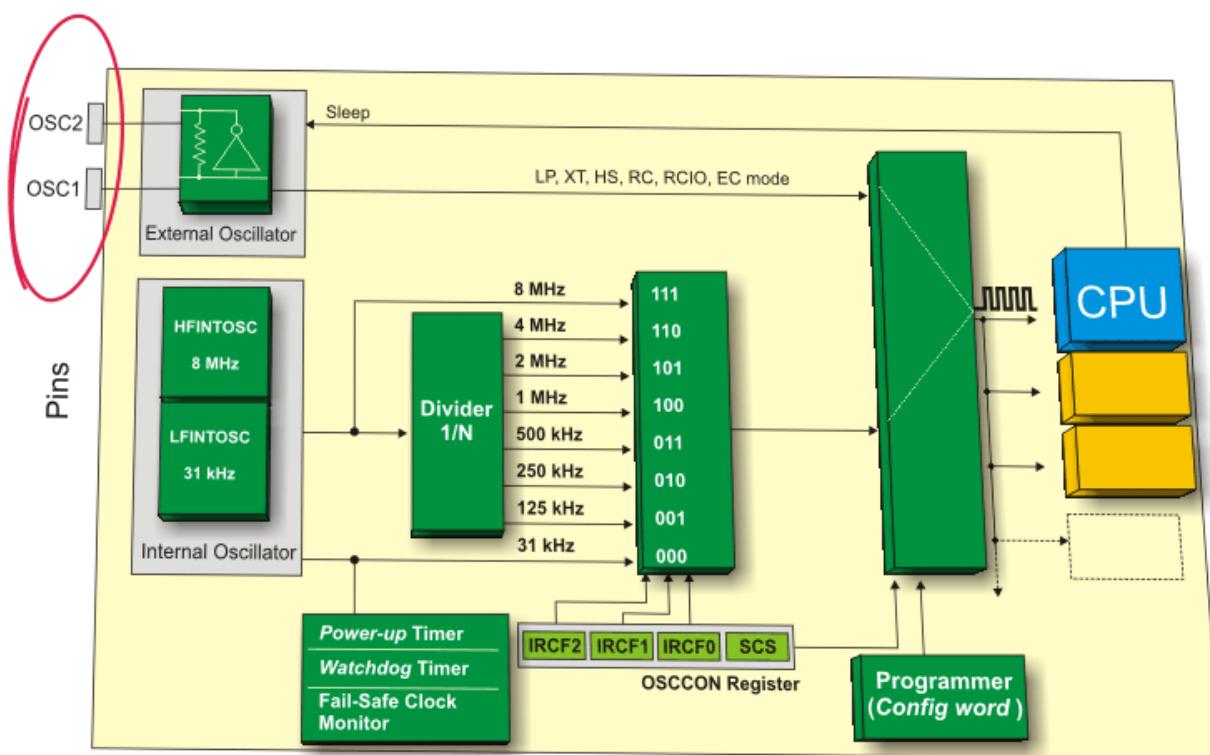
Các ngắt được lập trình trong thanh ghi cấu hình dùng để đánh thức CPU khỏi chế độ ngủ bao gồm các nguồn reset và nguồn ngắt như sau:

- Chân MCLR bị kéo xuống mức thấp
- Bộ định thời giảm sát WDT hết thời gian
- Ngắt xảy ra ở chân INT

- Ngắt của timer T1 và T3 của họ PIC 18
- Ngắt xảy ra khi ADC chuyển đổi xong
- Ngắt khi bộ so sánh thay đổi trạng thái
- Ngắt khi có sự kiện capture ở ngõ vào
- Ngắt khi port B thay đổi
- Ngắt khi xảy ra truyền dữ liệu I2C

5.5.3 CẤU HÌNH CÁC DẠNG DAO ĐỘNG CỦA CPU

Các lựa chọn bộ dao động hệ thống được lựa chọn ở các bit dao động trong thanh ghi cấu hình dùng để chọn 1 trong các dạng dao động. Sơ đồ khái mạc dao động của PIC như hình 5-36.



Hình 5-36. Sơ đồ khái mạc dao động của PIC.

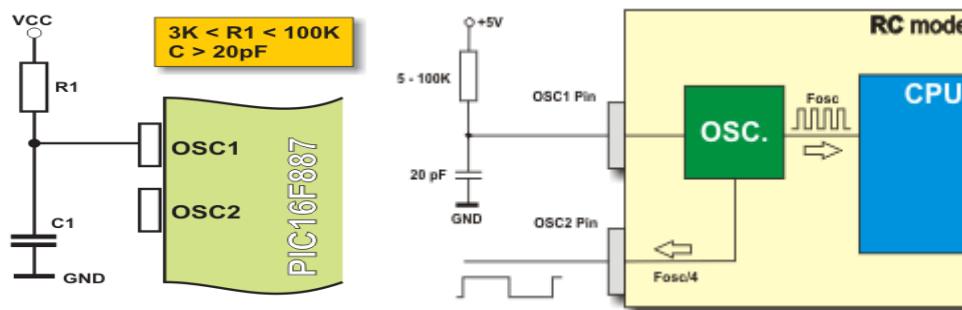
Các dạng dao động như sau:

- RC là dao động RC gắn ở bên ngoài, tần số thấp nằm trong giới hạn từ DC đến 4MHz. Chân RA6/OSC2/CLKOUT là chân ngõ ra của bộ dao động, điện trở R và tụ C được nối với chân RA7/OSC1/CLKIN. Xem hình 5-37.
- RCIO là dao động RC gắn ở bên ngoài, tần số thấp nằm trong giới hạn từ DC đến 4MHz. Chân RA6/OSC2/CLKOUT dùng làm chân IO, điện trở R và tụ C được nối với chân RA7/OSC1/CLKIN. Xem hình 5-37.
- INTOSC hay INTRC là dao động RC đã tích hợp ở bên trong hay còn gọi là dao động nội, tần số là 4MHz hoặc 8Mhz với sai số $\pm 2\%$. Chân RA6/OSC2/CLKOUT là chân ngõ ra của bộ dao động, chân RA7/OSC1/CLKIN là chân ngõ vào của bộ dao động. Xem hình 5-38.
- INTOSCO hay INTRC là dao động RC đã tích hợp ở bên trong hay còn gọi là dao động nội, tần số là 4MHz hoặc 8Mhz với sai số $\pm 2\%$. Chân

RA6/OSC2/CLKOUT dùng làm chân IO, chân RA7/OSC1/CLKIN dùng làm chân IO. Xem hình 5-38.

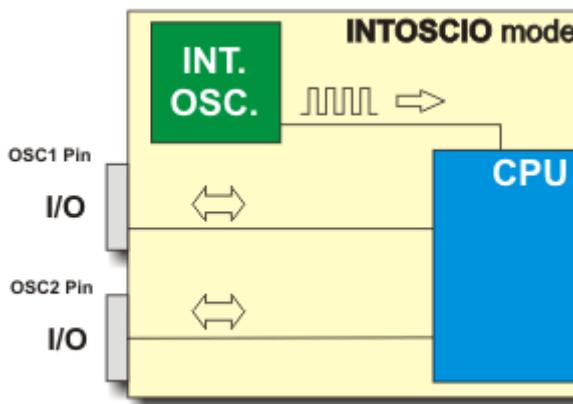
- EC (External Clock) là dao động tạo xung ở bên ngoài và cấp cho vi điều khiển, tần số phụ thuộc vào dao động ở ngoài nằm trong giới hạn của CPU. Chân RA6/OSC2/CLKOUT dùng làm chân IO, chân RA7/OSC1/CLKIN là chân nhận xung dao động từ bên ngoài. Xem hình 5-39.
- HS là dao động thạch anh, tần số cao nằm trong giới hạn từ 4MHz đến 20MHz.
- XT là dao động thạch anh, tần số chuẩn nằm trong giới hạn từ 100kHz đến 4MHz.
- LP là dao động thạch anh, tần số thấp nằm trong giới hạn từ 5kHz đến 200kHz.

Sơ đồ mạch dao động RC bên ngoài và các thông số của điện trở và tụ điện cho ở hình 5-37.



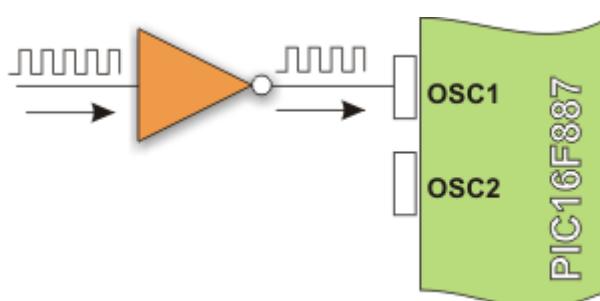
Hình 5-37. Dao động RC bên ngoài.

Sơ đồ mạch dao động RC bên trong cho ở hình 5-38.



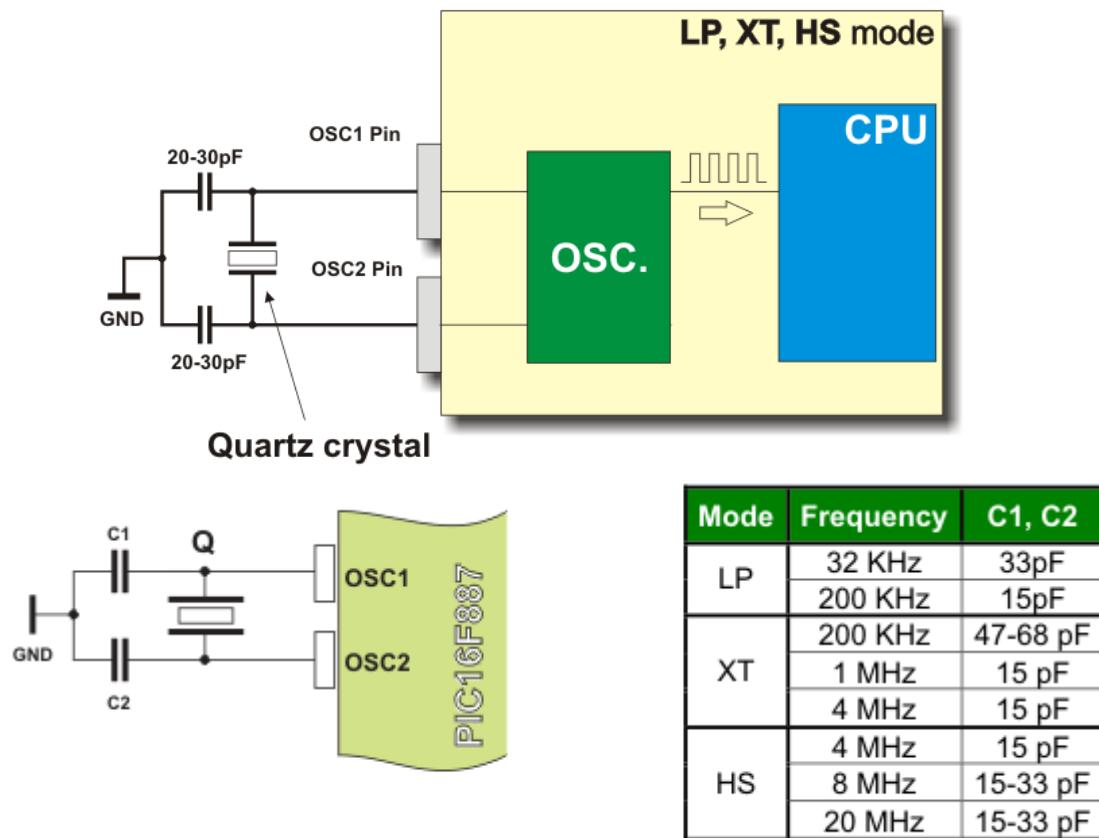
Hình 5-38. Dao động RC bên trong.

Sơ đồ mạch dao động EC lấy tín hiệu dao động từ bên ngoài cho ở hình 5-39.



Hình 5-39. Dao động lấy từ bên ngoài.

Sơ đồ mạch dao động HS, XT, LP và các thông số của các tụ điện cho ở hình 5-40.



Hình 5-40. Các dạng dao động LP, XT, HS.

5.5.4 CÁU HÌNH BẢO VỆ CODE

Chương trình sau khi viết xong dịch ra file hex nạp vào bộ nhớ của vi điều khiển, nếu không khóa lại thì người khác có thể đọc ra và copy sang vi điều khiển khác một cách dễ dàng.

Nếu muốn bảo vệ không cho người khác copy thì ta phải khóa lại và khi đọc ra thì chỉ toàn là dữ liệu FFH.

Với vi điều khiển khác thì việc nạp code và khóa độc lập nhau: nạp xong rồi khóa, với vi điều khiển PIC thì bạn phải cấu hình các bit trong thanh ghi cấu hình để khóa luôn, khi biên dịch xong và nạp thì bộ nạp sẽ nạp từ cấu hình vào thanh ghi cấu hình và nạp code vào bộ nhớ chương trình.

Trong phần mềm lập trình thì có các từ khóa để lập trình rất đơn giản: PROTECT là khóa, NOPROTECT là không khóa. Nếu không chọn thì mặc nhiên là không khóa.

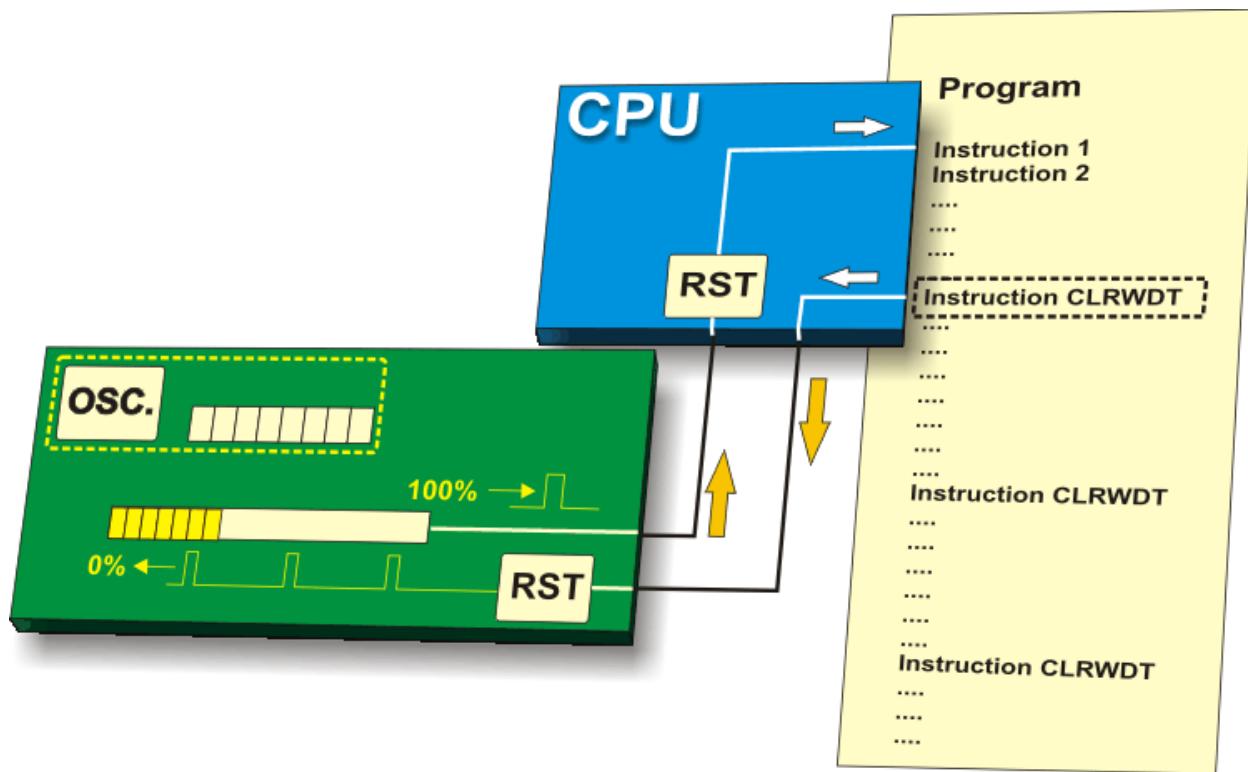
5.5.5 BỘ ĐỊNH THỜI GIÁM SÁT (WATCH DOG TIMER)

Bộ định thời giám sát có chức năng đếm thời gian theo yêu cầu và khi đếm đủ thời gian yêu cầu thì nó sẽ tạo tín hiệu reset CPU, xem hình 5-41.

Chức năng này có nhiều ứng dụng tùy thuộc vào yêu cầu cụ thể và tùy thuộc vào người lập trình. Ứng dụng chính của bộ định thời giám sát này dùng để giám sát CPU thực hiện chương trình tránh rơi vào những vòng lặp vô tận vì sự hư hỏng của phần cứng.

Ví dụ hệ thống điều khiển thang máy trong các tòa nhà cao tầng, khi bạn muốn di chuyển từ tầng này sang tầng khác thì khi đó CPU sẽ ra lệnh cho motor hoạt động kéo thang máy di

chuyển và kiểm tra công tắc hành trình của tầng mà bạn muốn đến và khi thang máy đến đúng tầng bạn muốn thì thang máy sẽ làm tiếp điểm hành trình đóng lại và đúng vị trí, thang dừng.



Hình 5-41. Chức năng của bộ định thời giám sát - WDT.

Mọi thứ hoạt động tốt, ổn định thì không có gì để bàn luận nhưng vì theo thời gian thì tiếp điểm của công tắc hành trình bị hỏng và như vậy không tìm được điều kiện để dừng thang máy, nếu thang tiếp tục chạy lên tầng cao hơn thì không đúng lại chạy xuống gấp tầng thấp hơn cũng không đúng, vậy thì làm sao để dừng thang máy?

Có nhiều giải pháp như thường xuyên bảo trì và sử dụng đúng tuổi thọ dù chưa hư hỏng vẫn phải thay mới, một giải pháp đơn giản là sử dụng bộ định thời giám sát WDT.

Để làm điều này thì ta phải biết thời gian di chuyển của thang máy đến tầng ta muốn là bao nhiêu thời gian, ta cho thang di chuyển vừa kiểm tra công tắc hành trình vừa tính thời gian, nếu công tắc hành trình còn tốt vì bỏ qua kiểm tra thời gian, nếu hết thời gian mà công tắc hành trình chưa đóng thì như vậy là công tắc hành trình có vấn đề cần phải dừng chương trình và báo động.

Làm sao để dừng chương trình thì ta phải dùng bộ định thời giám sát, cài đặt thời gian cho bộ định thời giám sát lớn hơn thời gian của thang máy di chuyển ta muốn.

Nếu thang máy di chuyển đúng thì thời gian kết thúc sớm hơn thời gian reset của bộ định thời đang đếm và khi đó ta phải xóa bộ định thời giám sát và xem như không có vấn đề gì.

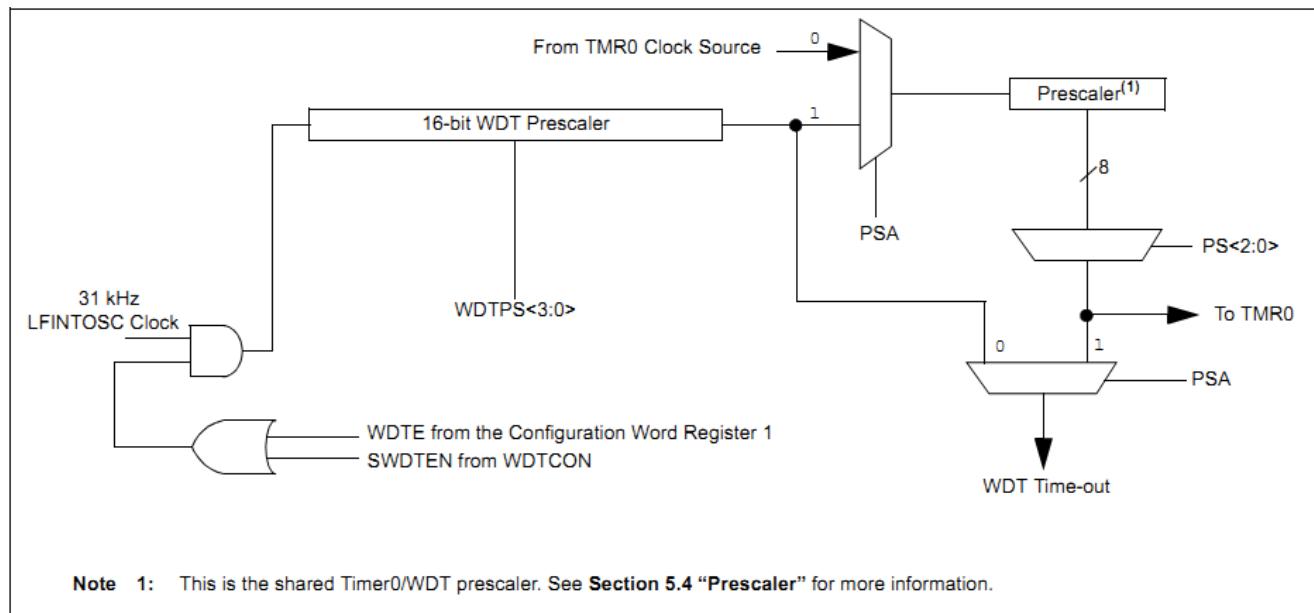
Nếu công tắc hành trình hỏng thì thang máy di chuyển không đúng và thời gian kéo dài hơn thời gian của bộ định thời giám sát và khi bộ định thời giám sát đếm xong thì nó sẽ reset CPU khỏi vòng lặp đang kiểm tra công tắc hành trình và tiến hành báo động.

Trong hình 5-41 chúng ta thấy CPU thực hiện chương trình và nếu thực hiện bình thường, không có vấn đề hư hỏng thì trong chương trình phải tiến hành xóa bộ định thời giám sát để CPU không bị reset. Nếu không xóa thì khi thời gian định thời đã hết thì bộ định thời giám sát sẽ reset CPU.

Trong thanh ghi cấu hình có 1 bit tên WDTE dùng để cho phép bộ định thời giám sát hoạt động khi bằng 0 và không cho phép khi bằng 1.

Khi lập trình bằng PIC C thì lựa chọn rất đơn giản là NOWDT là không dùng bộ định thời giám sát, WDT tức là cho phép sử dụng và tiến hành cấu hình các thanh ghi của WDT để thiết lập thời gian reset mong muốn.

Nguồn dao động cấp cho bộ định thời giám sát lấy từ dao động nội LFINOSC có tần số khoảng 31kHz. Qua bộ chia 16 bit và bộ chia 8 bit dùng chung với timer T0. Sơ đồ khói bộ định thời giám sát như hình 5-42.

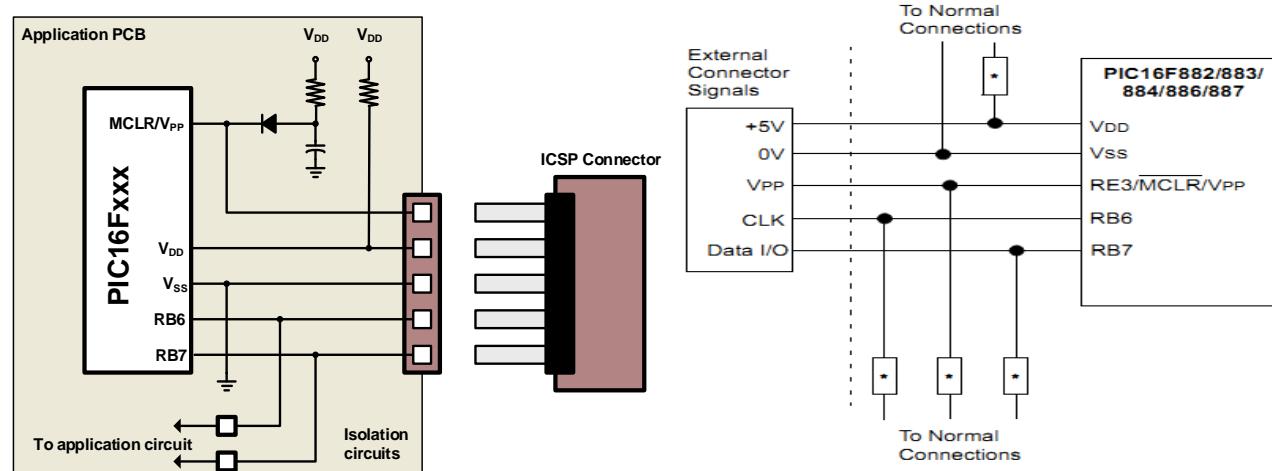


Hình 5-42. Sơ đồ khói của bộ định thời giám sát - WDT.

5.5.6 MẠCH NẠP NỐI TIẾP BÊN TRONG

Vì điều khiển PIC 16F887 có thể lập trình để nạp code vào bộ nhớ chương trình ở nối tiếp khi chip được gắn trong các mạch điện ứng dụng. Lập trình nạp code nối tiếp được thực hiện đơn giản bằng 2 đường tín hiệu: xung clock, dữ liệu và 3 tín hiệu gồm có nguồn V_{DD}, GND và điện áp lập trình.

Để thực hiện nạp code nối tiếp vào bộ nhớ chương trình thì phải có mạch nạp nối tiếp (In-Circuit Serial Programming - ICSP) như đã giới thiệu ở chương trước và ở phần trình bày port B. Các tín hiệu của vi điều khiển PIC giao tiếp với mạch nạp như hình 5-43.



Hình 5-43. Các tín hiệu của PIC giao tiếp với mạch nạp dạng nối tiếp ICSP.

Các tín hiệu nạp và chức năng cách tín hiệu cho ở bảng

Bảng 5-1. Tên các tín hiệu và chức năng của mạch nạp ICSP.

Chân	Chức năng
V _{PP}	Chân cấp điện áp lập trình 13V
V _{DD}	Nguồn cung cấp
GND hay V _{SS}	Chân nối đất
RB6	Chân nhận xung clock
RB7	Chân dữ liệu vào ra và nhận lệnh lập trình

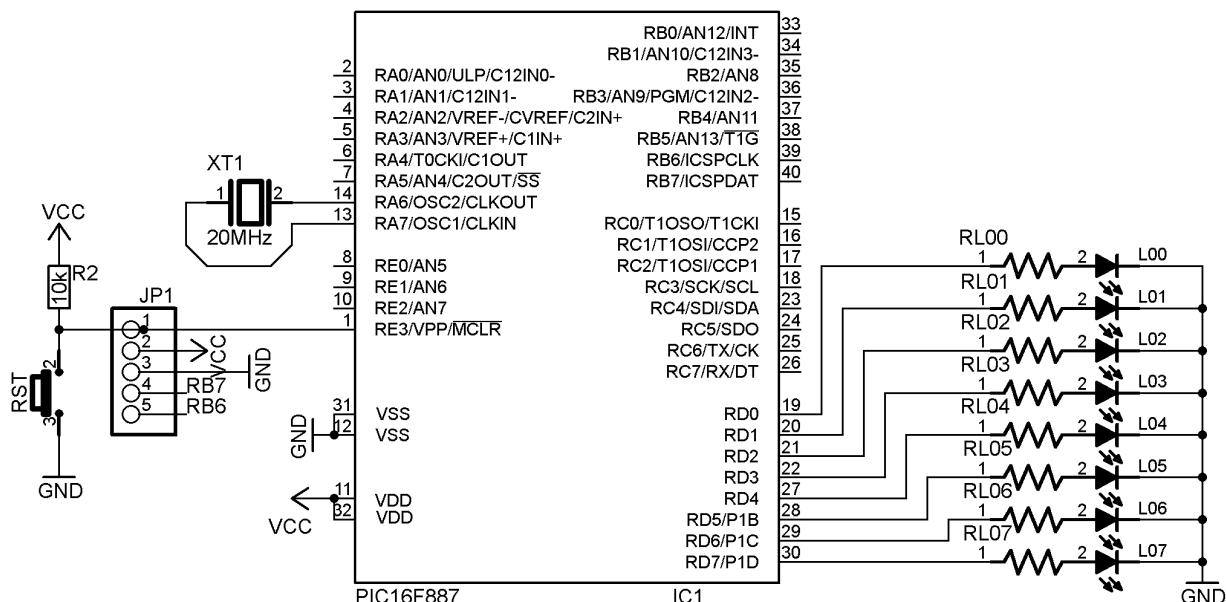
Trong thanh ghi cấu hình có bit cho phép lựa chọn chế độ lập nạp code ở mức điện áp thấp 5V hoặc điện áp cao 13V. Các bộ nạp PIC Kit2 và 3 thường lập trình bằng điện áp cao.

Thanh ghi cấu hình còn nhiều sự lựa chọn khác, phần vừa rồi trình bày các cấu hình quan trọng cần biết, các cấu hình còn lại bạn có thể tham khảo thêm trong datasheet. Phần tiếp theo trình bày các ứng dụng cơ bản của vi điều khiển PIC.

5.6 CÁC ÚNG DỤNG ĐIỀU KHIỂN LED ĐƠN

Bài 5-1: Dùng vi điều khiển 16F887 điều khiển 8 led đơn sáng tắt.

- Sơ đồ mạch: như hình 5-44.



Hình 5-44. Sơ đồ điều khiển led đơn.

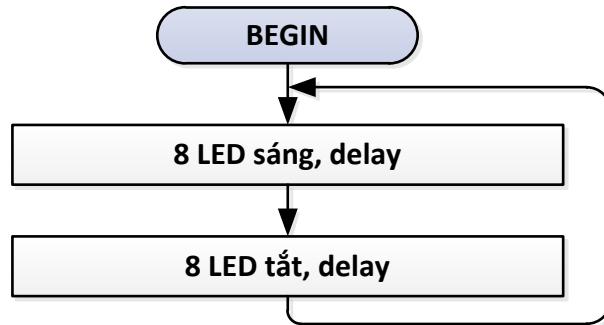
Mạch dùng portD kết nối với 8 led đơn. Mức logic 0 làm led tắt, mức logic 1 làm led sáng, điện áp của led là 2V, dòng qua led chọn 10mA, điện trở hạn dòng cho led được tính như sau:

$$R = \frac{V_{CC} - V_{LED}}{I_{LED}} = \frac{5V - 2V}{10mA} = 300\Omega$$

Có thể dùng điện trở 300Ω hoặc 330Ω.

Mạch sử dụng thạch anh có tần số 20MHz, có nút nhấn reset, điện trở reset $10k\Omega$ và có pinheader 5 chân dùng để kết nối với mạch nạp nối tiếp.

- Lưu đồ:



Hình 5-45. Lưu đồ điều khiển led đơn chớp tắt.

- Chương trình:

```

#include <16F887.H>
#FUSES NOWDT, PUT, HS, NOPROTECT, NOLVP
#USE DELAY(CLOCK=20M)

VOID MAIN()
{
    SET_TRIS_D(0x00);
    WHILE(TRUE)
    {
        OUTPUT_D(0xFF);      DELAY_MS(1000);
        OUTPUT_D(0x00);      DELAY_MS(1000);
    }
}
  
```

- Giải thích chương trình:

Hàng thứ nhất “#INCLUDE <16F887.H>” là khai báo thư viện đang sử dụng là PIC 16F887.

Hàng thứ hai “#FUSES NOWDT, PUT, HS, NOPROTECT, NOLVP” là khai báo cấu hình cho PIC.

Hàng thứ ba “#USE DELAY(CLOCK=20M)” khai báo tần số tự thạch anh mà vi điều khiển sử dụng, với khai báo trên thì tần số sử dụng là 20MHz.

Các hàm định thời là:

- **DELAY_MS(VALUE)** - Thời gian định thời là mili giây.
- **DELAY_US(VALUE)** - Thời gian định thời là μ s.

Chương trình chính gồm: Lệnh “SET_TRIS_D(0x00);” có chức năng khởi tạo portD là port xuất.

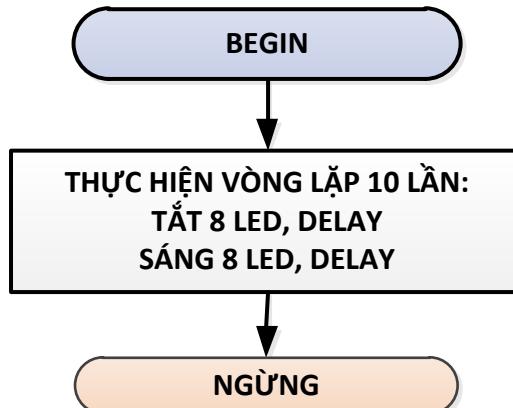
Lệnh “OUTPUT_D(0xFF);” có chức năng xuất dữ liệu 0xFF ra portD làm 8 led sáng, tiến hành gọi hàm delay, sau đó xuất dữ liệu 0x00 ra portD làm 8 led tắt, gọi hàm delay và lặp lại.

Các thông số cấu hình: “**NOWDT**” là không sử dụng bộ định thời giám sát (No watchdog timer), “**PUT**” là sử dụng bộ định thời khi có nguồn để kéo dài thêm thời gian reset vi điều

khiến để chờ nguồn điện ổn định, thời gian kéo dài thêm 72ms (Power up timer), “HS” là sử dụng bộ dao động tần số cao từ 4MHz đến 20MHz (High Speed), “NOPROTECT” là không sử dụng bảo vệ mã code nạp vào bộ nhớ flash bên trong, “NOLVP” là không sử dụng chế độ nạp code dùng nguồn điện áp thấp 5V mà dùng nguồn 12,5V.

Bài 5-2: Dùng vi điều khiển 16F887 điều khiển 8 led đơn sáng tắt 10 lần rồi tắt luôn.

- Sơ đồ mạch: giống hình 5-44.
- Lưu đồ:



Hình 5-46. Lưu đồ điều khiển led đơn chớp tắt 10 lần.

- Chương trình:

```

#include <16F887.H>
#FUSES NOWDT, PUT, HS, NOPROTECT, NOLVP
#USE DELAY(CLOCK=20M)
UNSIGNED INT8 I;
VOID MAIN()
{
    SET TRIS D(0x00);
    FOR(I=0; I<10; I++)
    {
        OUTPUT_D(0xFF);
        OUTPUT_D(0x00);
        DELAY_MS(1000);
        DELAY_MS(1000);
    }
    WHILE(TRUE) {} // Dừng tại chỗ
}
    
```

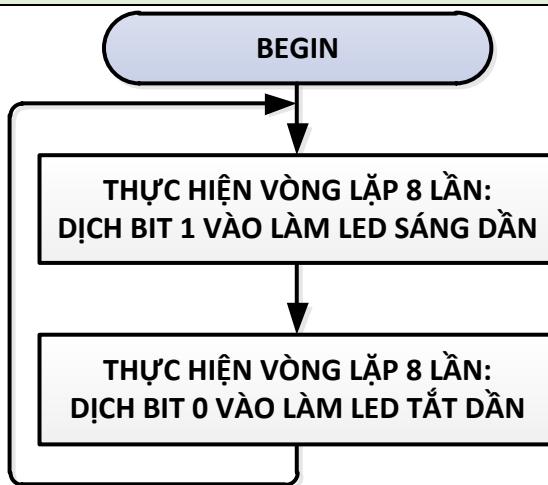
Das Programm ist ein C-Code für den 16F887-Mikrocontroller. Es verwendet die Headerdatei #include <16F887.H>. Die Fuses sind wie folgt konfiguriert: #FUSES NOWDT, PUT, HS, NOPROTECT, NOLVP. Der Zeitabstand zwischen den Schritten wird über #USE DELAY(CLOCK=20M) festgelegt. Der Main-Block beginnt mit der Initialisierung von TRIS D auf 0x00. Danach wird ein For-Schleife ausgeführt, die 10 Mal durchlaufen wird. In jeder Iteration wird zuerst die LED auf HIGH gesetzt (OUTPUT_D(0xFF)), dann auf LOW (OUTPUT_D(0x00)). Zwischen den Schritten gibt es eine 1000ms-Delay (DELAY_MS(1000)). Nach Abschluss der Schleife wird ein unendlicher While-Zyklus (WHILE(TRUE) {}) eingeschlossen, der als 'Halt an Stelle' (Dừng tại chỗ) markiert ist.

- Giải thích chương trình:

Chương trình này cho vòng lặp for thực hiện 10 lần điều khiển 8 led chớp tắt, sau đó thực hiện lệnh while(1) và không có lệnh nào trong vòng lặp nên xem như nhảy tại chỗ.

Bài 5-3: Dùng vi điều khiển 16F887 điều khiển 8 led đơn sáng dần dần tắt dần từ phải sang trái.

- Sơ đồ mạch: giống hình 5-44.
- Lưu đồ:



Hình 5-47. Lưu đồ điều khiển led đơn sáng dần tắt dần từ phải sang trái.

- Chương trình:

```

#include <16F887.H>
#FUSES NOWDT, PUT, HS, NOPROTECT, NOLVP
#USE DELAY(CLOCK=20M)
UNSIGNED INT8 I, X;
VOID MAIN()
{
    SET_TRIS_D(0x00);
    X=0x00; OUTPUT_D(X);
    DELAY_MS(500);
    WHILE(TRUE)
    {
        FOR (I=0; I<8; I++)
        {
            X = (X<<1)+0x01; OUTPUT_D(X); DELAY_MS(500);
        }
        FOR (I=0; I<8; I++)
        {
            X = (X<<1); OUTPUT_D(X); DELAY_MS(500);
        }
    }
}
  
```

TẮT HẾT

SÁNG HẾT TRƯỚC KHI TẮT DÀN LÀ LÂY TRẠNG THÁI CUỐI CỦA SÁNG DÀN PHÍA TRÊN

- Giải thích chương trình:

Khởi tạo portD là xuất dữ liệu, gán biến X bằng 0x00, xuất giá trị của X ra portD làm 8 led tắt, delay.

Vòng lặp for thứ nhất thực hiện 8 lần: tiến hành xoay trái dữ liệu của X và cộng với 0x01. Khi xoay trái dữ liệu thì số 0 được đẩy vào, cộng với 0x01 là để đẩy số 1 vào X. Dữ liệu biến X được xuất ra portD điều khiển led sáng dần, sau 8 lần thì X sẽ bằng 1111_1111 – 8 led sáng hết.

Vòng lặp for thứ hai thực hiện 8 lần: tiến hành xoay trái dữ liệu của X. Khi xoay trái dữ liệu thì số 0 được đẩy vào. Dữ liệu biến X được xuất ra portD điều khiển led tắt dần, sau 8 lần thì X sẽ bằng 0000_0000 – 8 led tắt hết.

Sau đó lặp lại cú thê 8 led sáng dần xong rồi tắt dần.

Bài tập 5-1: Hãy viết chương trình điều khiển 1 led đơn L00 của mạch điện hình 5-34 sáng 1 giây tắt 1 giây, các led còn lại không ảnh hưởng.

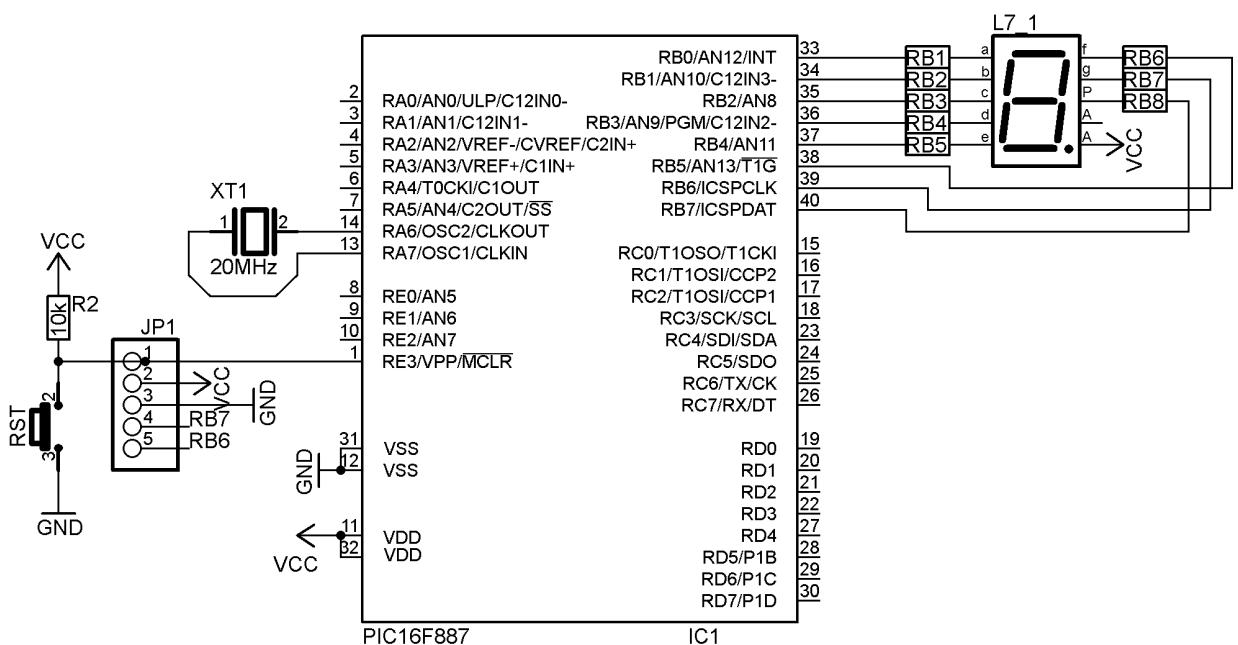
Bài tập 5-2: Hãy viết chương trình điều khiển 8 led đơn của mạch điện hình 5-34 sáng dần lên và tắt dần từ phải sang trái, sau đó sáng dần và tắt dần từ trái sang. Các bước thực hiện gồm viết lưu đồ và chương trình.

Bài tập 5-3: Hãy viết chương trình điều khiển 16 led đơn sáng dần lên và tắt dần từ phải sang trái dùng 2 port B và D, thời gian trễ 1 giây. Các bước thực hiện gồm vẽ mạch, viết lưu đồ và chương trình.

5.7 CÁC ỨNG DỤNG ĐIỀU KHIỂN LED 7 ĐOẠN TRỰC TIẾP

Bài 5-4: Dùng vi điều khiển 16F887 kết nối với 1 led 7 đoạn anode chung và viết chương trình đếm từ 0 đến 9 với thời gian trễ tùy chọn.

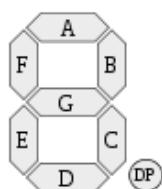
- Sơ đồ mạch: dùng portB kết nối với 1 led 7 đoạn anode chung. Mức logic 1 làm led sáng, mức logic 0 làm led tắt, điện trở hạn dòng cho led là 330Ω .



Hình 5-48. Sơ đồ kết nối portB với 1 led 7 đoạn.

- Mã 7 đoạn:

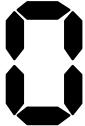
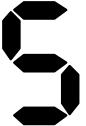
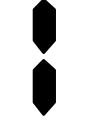
Led 7 đoạn và tên các đoạn như hình 5-39:



Hình 5-49. Hình led 7 đoạn.

Bảng 5-2. Mã 7 đoạn cho các số thập phân:

TP	SỐ NHỊ PHÂN								HEX	TP	SỐ NHỊ PHÂN								HEX		
	7	6	5	4	3	2	1	0			7	6	5	4	3	2	1	0			
DP	G	F	E	D	C	B	A	DP	G	F	E	D	C	B	A	DP	G	F	E	D	HEX

	1	1	0	0	0	0	0	C0		1	0	0	1	0	0	1	0	92	
	1	1	1	1	1	0	0	1	F9		1	0	0	0	0	0	1	0	82
	1	0	1	0	0	1	0	A4		1	1	1	1	1	0	0	0	F8	
	1	0	1	1	0	0	0	B0		1	0	0	0	0	0	0	0	80	
	1	0	0	1	1	0	0	1	99		1	0	0	1	0	0	0	0	90

- Lưu đồ:



Hình 5-50. Lưu đồ đếm từ 0 đến 9.

- Chương trình:

```

#include <16F887.H>
#FUSES NOWDT, PUT, HS, NOPROTECT, NOLVP
#USE DELAY(CLOCK=20M)
CONST UNSIGNED CHAR MA7DOAN[10]={0xC0,0xF9,0xA4,0xB0,0x99,
0x92,0x82,0xF8,0x80,0x90};
    
```

```

SIGNED INT DEM;
UNSIGNED INT MA_DEM;
VOID MAIN()
{
    SET_TRIS_B(0x00);
    WHILE (TRUE)
    {
        FOR (DEM=0;DEM<10;DEM++)
        {
            MA_DEM = MA7DOAN[DEM];
            OUTPUT_B(MA_DEM);
            DELAY_MS(300);
        }
    }
}
    
```

```

    }
}

```

- Giải thích chương trình:

Sau các lệnh khai báo thư viện, cấu hình và khai báo tần số sử dụng là khai báo mảng chứa mã 7 đoạn và các biến đếm, biến chứa mã.

Chương trình chính khởi tạo các portB là xuất, cho vòng lặp for với biến “DEM” chạy từ 0 đến 9, tiến hành giải mã và gởi mã led 7 đoạn tương ứng với biến DEM ra port để hiển thị, delay và lặp lại.

Chương trình trên có thể viết gọn hơn như sau:

```

#include <16F887.H>
#FUSES NOWDT, PUT, HS, NOPROTECT, NOLVP
#USE DELAY(CLOCK=16M)
CONST UNSIGNED CHAR MA7DOAN [10]={0xC0,0xF9,0xA4,0xB0,0x99,0x92,0x82,0xF8,
0x80,0x90};
SIGNED INT      DEM;

VOID MAIN()
{
    SET_TRIS_B(0x00);
    WHILE(TRUE)
    {
        FOR (DEM=0;DEM<10;DEM++)
        {
            OUTPUT_B(MA7DOAN[DEM]);
            DELAY_MS(300);
        }
    }
}

```

Trong chương trình xuất mã 7 đoạn của biến DEM ra portB để hiển thị luôn nên không cần sử dụng biến trung gian MA_DEM.

Trong chương trình thì các hàng khai báo loại vi điều khiển, cấu hình, tần số clock và mã 7 đoạn ít thay đổi nên tạo thành 1 file thư viện <TV_16F887.C> lưu các khai báo như sau:

```

#include <16F887.H>
#FUSES NOWDT, PUT, HS, NOPROTECT, NOLVP
#USE DELAY(CLOCK=16M) ← 20M --> tùy theo ph.cứng
CONST UNSIGNED CHAR MA7DOAN [10]={0xC0,0xF9,0xA4,0xB0,0x99,0x92,0x82,0xF8,
0x80,0x90,0x88,0x83,0XC6,0xA1,0x86,0x8E}; ← 10 phần tử từ 0-9

```

Khi đó chương trình chính được viết lại như sau [16] từ 0-F

```

#include <TV_16F887.C>
SIGNED INT      DEM;

VOID MAIN()
{
    SET_TRIS_B(0x00);
    WHILE(TRUE)
    {
        FOR (DEM=0; DEM<10; DEM++)
        {
            OUTPUT_B(MA7DOAN[DEM]);
            DELAY_MS(300);
        }
    }
}

```

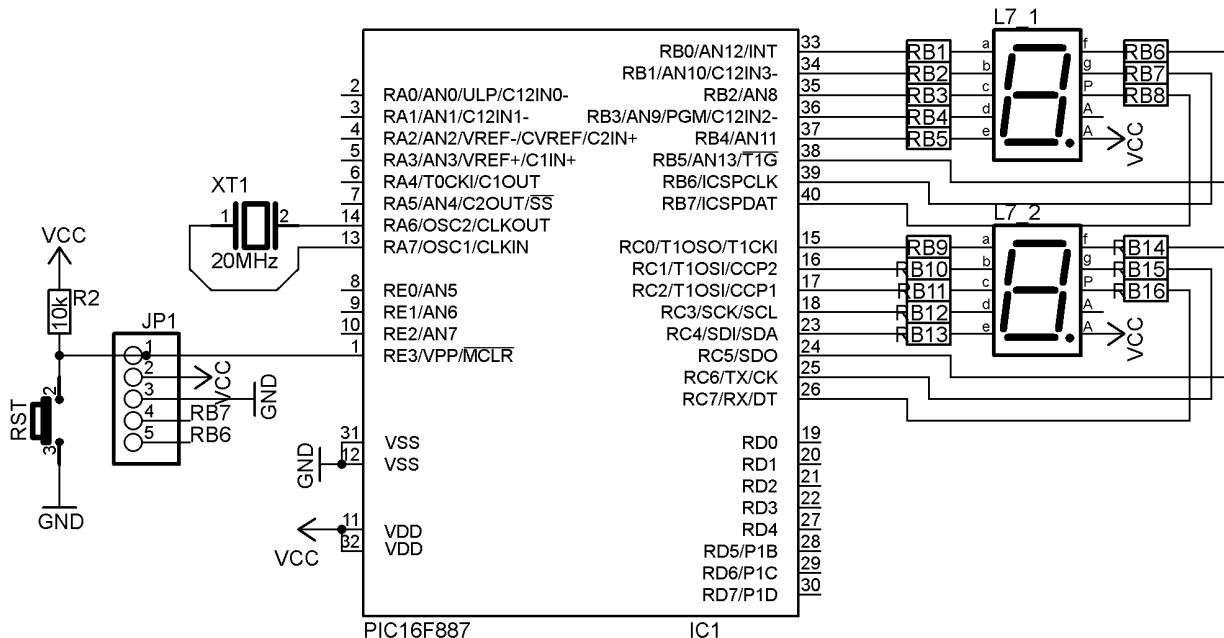
Từ đây về sau ta sẽ sử dụng thư viện này và có thể bổ sung thêm các nội dung mới.

Bài tập 5-4: Hãy viết chương trình thực hiện đếm từ 9 xuống 0.

Hãy vẽ mạch bằng Proteus và mô phỏng.

Bài 5-5: Dùng vi điều khiển 16F887 kết nối với 2 led 7 đoạn anode chung và viết chương trình đếm từ 00 đến 99 với thời gian trễ tùy chọn.

- Sơ đồ mạch: dùng portB và C kết nối với 2 led 7 đoạn anode chung.



Hình 5-51. Sơ đồ kết nối portB, C điều khiển 2 led 7 đoạn.

- Lưu đồ:



Hình 5-52. Lưu đồ đếm từ 00 đến 99.

- Chương trình:

```

#include<tv_16f887.c>
SIGNED INT DEM;
VOID MAIN()
{
    SET_TRIS_B(0x00);    SET_TRIS_C(0x00);
}
    
```

```

WHILE (TRUE)
{
    FOR (DEM=0 ; DEM<100 ; DEM++)
    {
        OUTPUT_C (MA7DOAN [DEM/10]); //CHUC
        OUTPUT_B (MA7DOAN [DEM%10]); //DONVI
        DELAY_MS (300);
    }
}

```

- Giải thích chương trình:

Chương trình chính khởi tạo các portB và C là xuất, cho vòng lặp for với biến “DEM” chạy từ 0 đến 99, tiến hành chia tách hàng đơn vị, hàng chục rồi giải mã và gởi ra port tương ứng để hiển thị, delay.

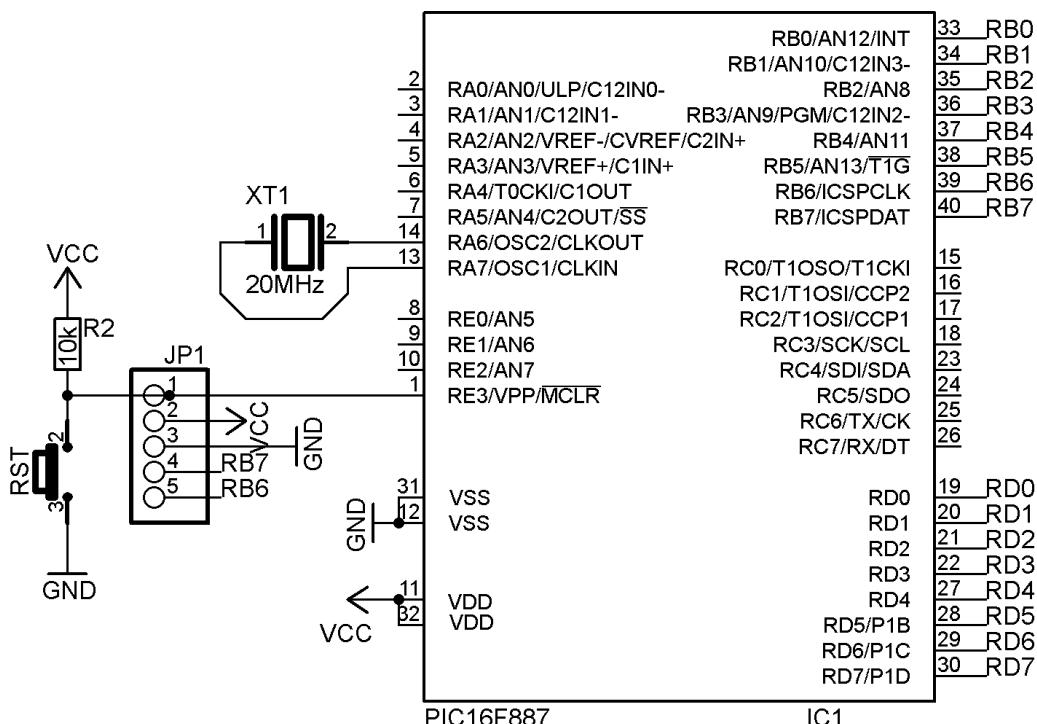
Bài tập 5-5: Hãy viết chương trình thực hiện đếm từ 000 đến 999 hiển thị trên 3 led 7 đoạn, thời gian trẻ tuỳ chọn. Các bước thực hiện bao gồm: vẽ mạch, viết lưu đồ, viết chương trình và mô phỏng.

5.8 CÁC ÚNG DỤNG ĐIỀU KHIỂN LED 7 ĐOẠN QUÉT

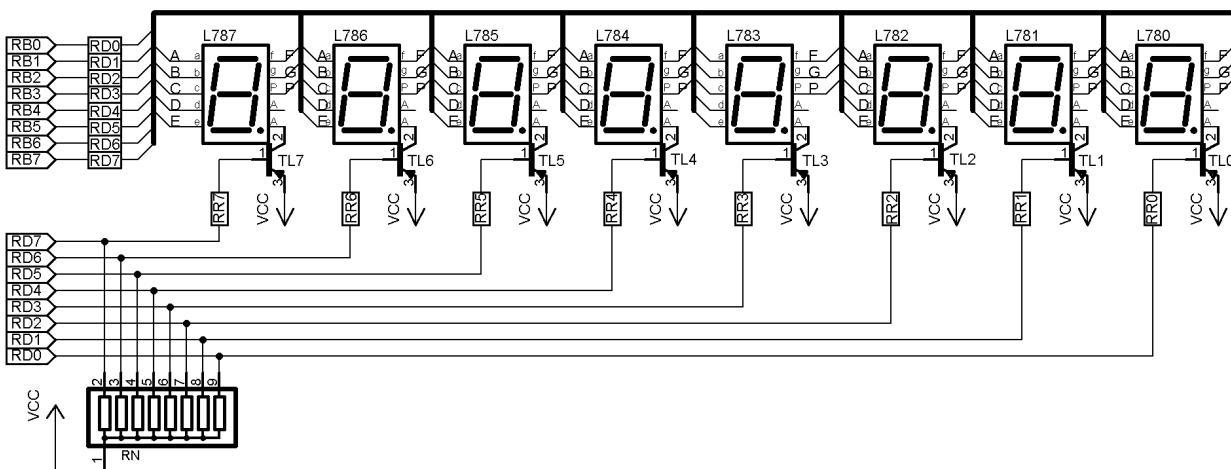
Khi số led 7 đoạn cần nhiều hơn ví dụ là 8 led thì ta không thể kết nối trực tiếp 1 port điều khiển 1 led vì không đủ port. Có nhiều phương pháp mở rộng để điều khiển được nhiều led nhưng ở đây trình bày phương pháp quét 8 led dùng 2 port.

Bài 5-6: Dùng vi điều khiển 16F887 kết nối với 8 led 7 đoạn anode chung theo phương pháp quét và viết chương trình hiển thị 8 số từ số 0 đến số 7 trên 8 led.

- Sơ đồ mạch:



A,B,C,D,E,F,G,P



Hình 5-53. Sơ đồ kết nối 2 port B và D điều khiển 8 led 7 đoạn quét.

Mạch dùng portB kết nối với các đoạn “a, b, c, d, e, f, g, dp” và portD kết nối điều khiển 8 transistor đóng ngắt.

Nguyên lý quét 8 led: do 8 led nối song – song các đoạn nên mỗi một thời điểm bạn chỉ xuất 1 mã 7 đoạn của led cần hiển thị ra portB đồng thời bạn xuất mã điều khiển transistor ra portD để cho phép transistor của led cần sáng dẫn, thực hiện delay một khoảng thời gian ngắn rồi tắt transistor đã mở và thực hiện tương tự cho led thứ 2, cứ thế cho đến led cuối cùng rồi lặp lại.

Thời gian cho led sáng tùy thuộc vào số lượng led sao cho chu kỳ quét tắt cả các led nhanh hơn đáp ứng tần số của mắt ta quan sát thì ta sẽ nhìn thấy các led sáng hết.

Với 8 led 7 đoạn theo sơ đồ ở trên thì thời gian sáng là 1/8 và thời gian tắt là 7/8 thì bạn chọn chu kỳ quét 8ms tương ứng với tần số 125Hz là phù hợp, bạn có thể chọn tần số cao hơn.

Ưu điểm phương pháp quét led: hiển thị được nhiều thông tin, mạch đơn giản.

Khuyết điểm: chương trình phải quét liên tục thì led mới sáng, do thời gian sáng ít, thời gian tắt nhiều nên led sẽ sáng mờ hơn so với phương pháp kết nối trực tiếp.

Giải pháp để tăng độ sáng của led: nếu quét khoảng 4 led thì led sáng rõ, nếu nhiều led hơn thì nên giảm điện trở hạn dòng từ 330Ω xuống còn khoảng 220Ω .

Tính toán lựa chọn transistor

Ở phương pháp quét thì transistor đóng vai trò như một công tắc khi tắt thì không cho dòng qua led nên led tắt, khi mở thì cho phép dòng qua led để led sáng đúng con số mong muốn. Để lựa chọn transistor thì ta giả sử cho tắt cả 7 đoạn và đầu chấm thập phân sáng và dòng qua mỗi đoạn là 15mA, tổng dòng là $15mA \times 8$ bằng 90mA, nên chọn transistor có dòng làm việc I_C lớn hơn bằng 90mA là được. Các transistor có dòng từ 100mA đến 150mA khá phổ biến là A564, A1015.

Câu hỏi: nếu quét 16 led thì bạn sẽ chọn giải pháp nào cho led sáng?

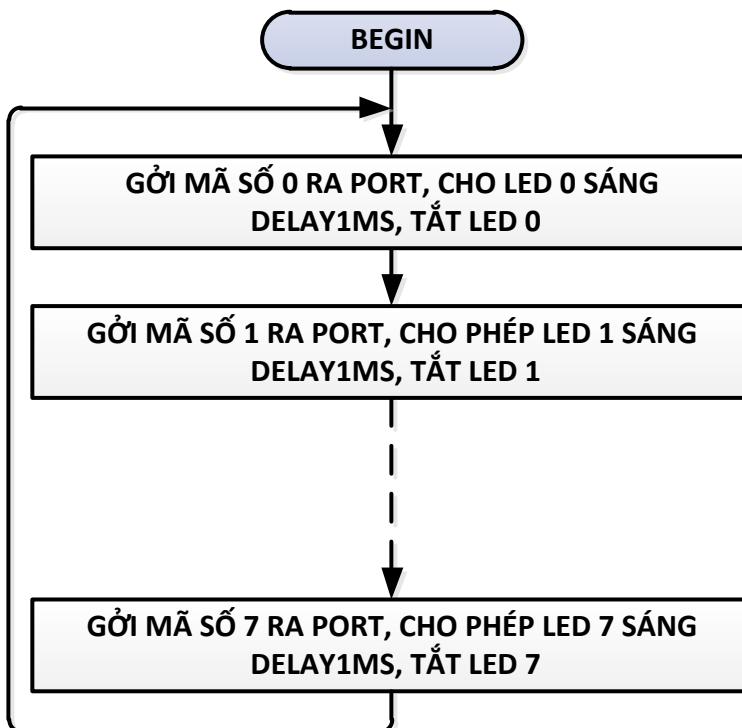
Trả lời: có 2 giải pháp: giải pháp thứ nhất là kết nối giống như sơ đồ ở trên dùng portB nối song song các đoạn của 16 led, dùng 2 port điều khiển 16 transistor để cho phép led sáng.

Với giải pháp này thì thời gian tắt của led là 15/16 – thời gian quá lớn led sẽ mờ và có thể nháy.

Giải pháp thứ hai là tách làm 2 nhóm 8 led: dùng port thứ nhất điều khiển các đoạn của 8 led nhóm thứ nhất, dùng port thứ hai điều khiển các đoạn của 8 led nhóm thứ hai, dùng port thứ ba điều khiển 8 transistor điều khiển chung 2 nhóm, mỗi lần mở 1 transistor dùng chung cho cả 2 led.

Với giải pháp này thì thời gian tắt của led là 7/8 – ít hơn giải pháp 1 nên led sáng rõ hơn.

- Lưu đồ:



Hình 5-54. Lưu đồ điều khiển 8 led quét sáng 8 số.

- Chương trình:

```

#include<TV_16F887.C>
VOID MAIN()
{
    SET_TRIS_B(0x00);           SET_TRIS_D(0x00);
    WHILE (TRUE)
    {
        OUTPUT_B(0xC0);        OUTPUT_LOW(PIN_D0);
        OUTPUT_HIGH(PIN_D0);
        OUTPUT_B(0XF9);        OUTPUT_LOW(PIN_D1);
        OUTPUT_HIGH(PIN_D1);
        OUTPUT_B(0XA4);        OUTPUT_LOW(PIN_D2);
        OUTPUT_HIGH(PIN_D2);
        OUTPUT_B(0XB0);        OUTPUT_LOW(PIN_D3);
        OUTPUT_HIGH(PIN_D3);
        OUTPUT_B(0x99);        OUTPUT_LOW(PIN_D4);
        OUTPUT_HIGH(PIN_D4);
        OUTPUT_B(0X92);        OUTPUT_LOW(PIN_D5);
        OUTPUT_HIGH(PIN_D5);
        OUTPUT_B(0X82);        OUTPUT_LOW(PIN_D6);
        OUTPUT_HIGH(PIN_D6);
        DELAY_MS(1);
    }
}
  
```

```

        OUTPUT_B(0XF8) ;
        OUTPUT_LOW(PIN_D7) ;
        DELAY_MS(1) ;
        OUTPUT_HIGH(PIN_D7) ;
    }
}

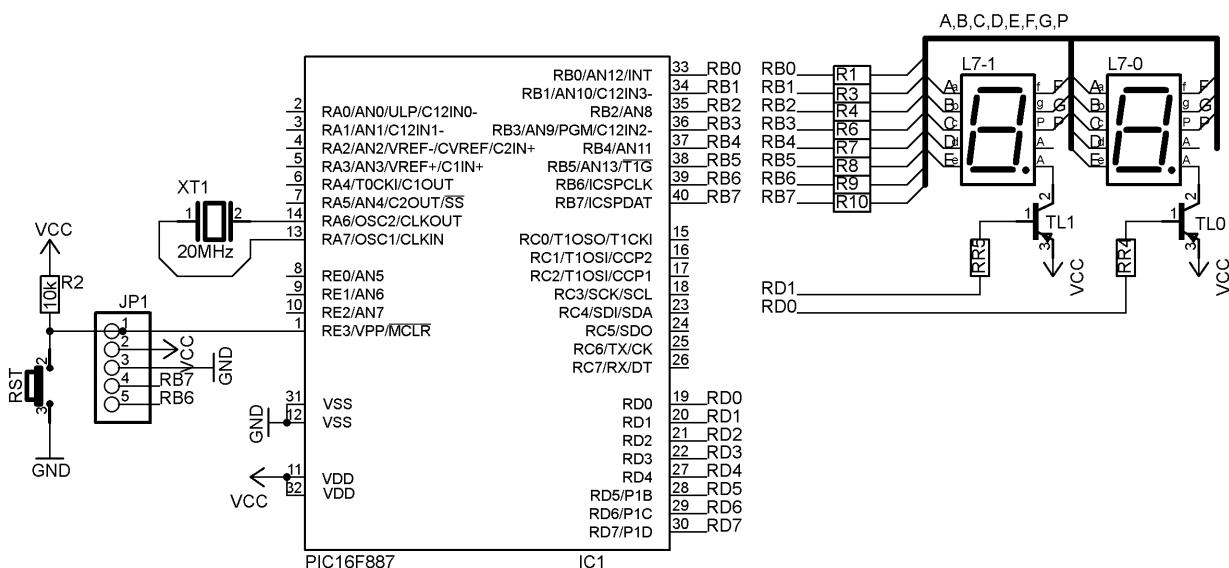
```

- Giải thích chương trình:

Chương trình chính khởi tạo 2 port B và D xuất dữ liệu. Lệnh xuất mã 7 đoạn ra portB và lệnh làm 1 bit của portD xuống mức 0 để cho transistor dẫn, gọi hàm delay 1ms sau đó tắt transistor và tiến hành cho led tiếp theo.

Bài 5-7: Dùng vi điều khiển 16F887 kết nối với 2 led 7 đoạn anode chung theo phương pháp quét và viết chương trình đếm từ 00 đến 99 hiển thị trên 2 led 7 đoạn quét, thời gian trễ tùy chọn.

- Sơ đồ mạch: do đếm 2 số nên kết nối 2 led.



Hình 5-55. Sơ đồ kết nối 2 port B và D điều khiển 2 led 7 đoạn quét.

- Lưu đồ:



Hình 5-56. Lưu đồ đếm từ 00 đến 99 hiển thị trên 2 led quét.

- Chương trình:

```

#include<TV_16F887.C>
UNSIGNED INT8 DEM, I;

VOID HIENTHI_DELAY()
{
    FOR (I=0; I<200; I++)
    {
        OUTPUT_B(MA7DOAN[DEM %10]);
        OUTPUT_LOW(PIN_D0);
        OUTPUT_HIGH(PIN_D0);
        OUTPUT_B(MA7DOAN[DEM /10]);
        OUTPUT_LOW(PIN_D1);
        OUTPUT_HIGH(PIN_D1);
    }
}

VOID MAIN()
{
    SET_TRIS_B(0x00);    SET_TRIS_D(0x00);
    OUTPUT_D(0xFF);
    WHILE (TRUE)
    {
        FOR (DEM=0; DEM<100; DEM++)
        {
            HIENTHI_DELAY();
        }
    }
}

```

Chương trình đếm đúng nhưng hiển thị không được, chớp. Một chu kỳ chớp sáng rồi tắt.

Lý do: mỗi led chỉ sáng 1/300 ms, tắt 299/300ms.

```

#include<TV_16F887.C>
UNSIGNED INT8 DEM, I;

VOID HIENTHI_DELAY()
{
    OUTPUT_B(MA7DOAN[DEM %10]);
    OUTPUT_LOW(PIN_D0);
    OUTPUT_HIGH(PIN_D0);
    OUTPUT_B(MA7DOAN[DEM /10]);
    OUTPUT_LOW(PIN_D1);
    OUTPUT_HIGH(PIN_D1);
}

VOID MAIN()
{
    SET_TRIS_B(0x00);    SET_TRIS_D(0x00);
    OUTPUT_D(0xFF);
    WHILE (TRUE)
    {
        FOR (DEM=0; DEM<100; DEM++)
        {
            HIENTHI_DELAY();
            DELAY_MS(298);
        }
    }
}

```

Bài tập 5-6: Hãy viết chương trình thực hiện đếm từ 000 đến 999 hiển thị trên 3 led 7 đoạn kết nối theo phương pháp quét, thời gian chờ tùy chọn. Các bước thực hiện bao gồm: vẽ mạch, viết lưu đồ, viết chương trình và mô phỏng.

```
#INCLUDE<TV_16F887.C>
UNSIGNED INT16 DEM, I;

VOID HIENTHI_DELAY()
{
    FOR (I=0; I<200; I++)
    {
        OUTPUT_B(MA7DOAN[DEM % 10]);
        OUTPUT_LOW(PIN_D0);
        OUTPUT_HIGH(PIN_D0);
        OUTPUT_LOW(PIN_D1);
        OUTPUT_HIGH(PIN_D1);
        OUTPUT_LOW(PIN_D2);
        OUTPUT_HIGH(PIN_D2);
    }
}

VOID MAIN()
{
    SET_TRIS_B(0x00);    SET_TRIS_D(0x00);
    OUTPUT_D(0xFF);
    WHILE (TRUE)
    {
        FOR(DEM=0; DEM<1000; DEM++)
        {
            HIENTHI_DELAY();
        }
    }
}
```

5.9 CÁC ỨNG DỤNG GIAO TIẾP VỚI NÚT NHẤN, BÀN PHÍM

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:

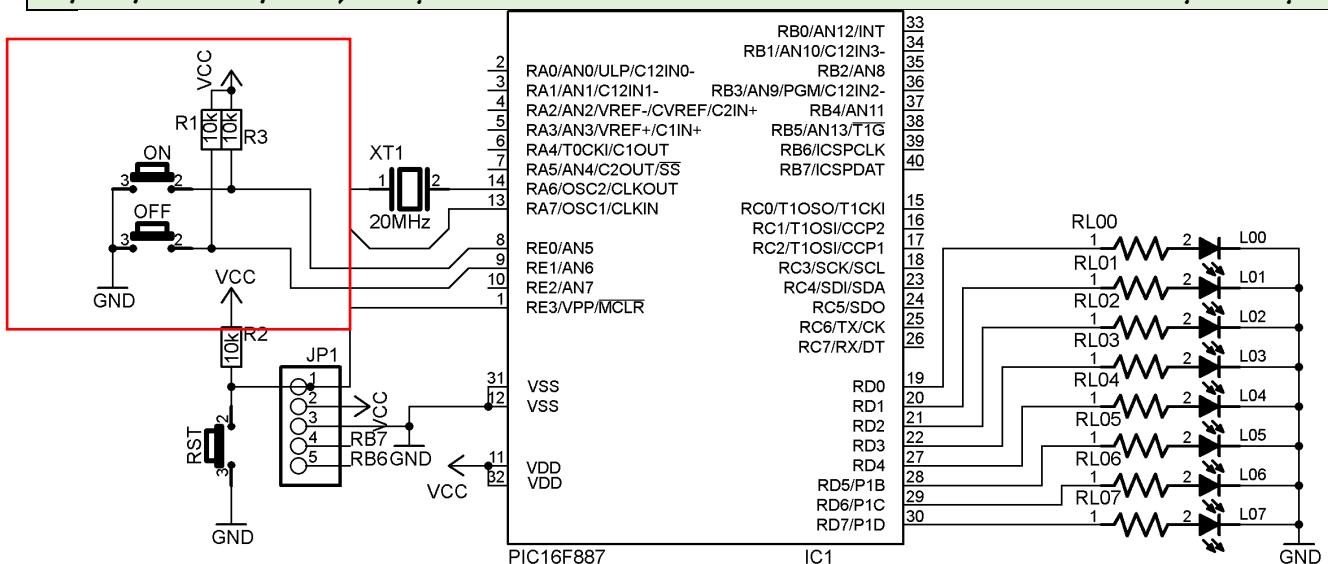
- Hệ thống ít phím: ví dụ điều khiển động cơ bằng 3 phím: start, stop, inv, đồng hồ có 3 đến 4 phím để chỉnh thời gian.
- Hệ thống nhiều phím: bàn phím máy tính, bàn phím điện thoại, ...

9.1.1 HỆ THỐNG ÍT PHÍM

Để hiểu giao tiếp vi điều khiển với bàn phím ta khảo sát các bài ứng dụng theo sau.

Bài 5-8: Dùng vi điều khiển 16F887 giao tiếp với 8 led đơn và 2 nút nhấn ON, OFF. Khi cấp điện thì 8 led tắt, khi nhấn ON thì 8 led sáng, khi nhấn OFF thì 8 led tắt.

- Sơ đồ mạch:



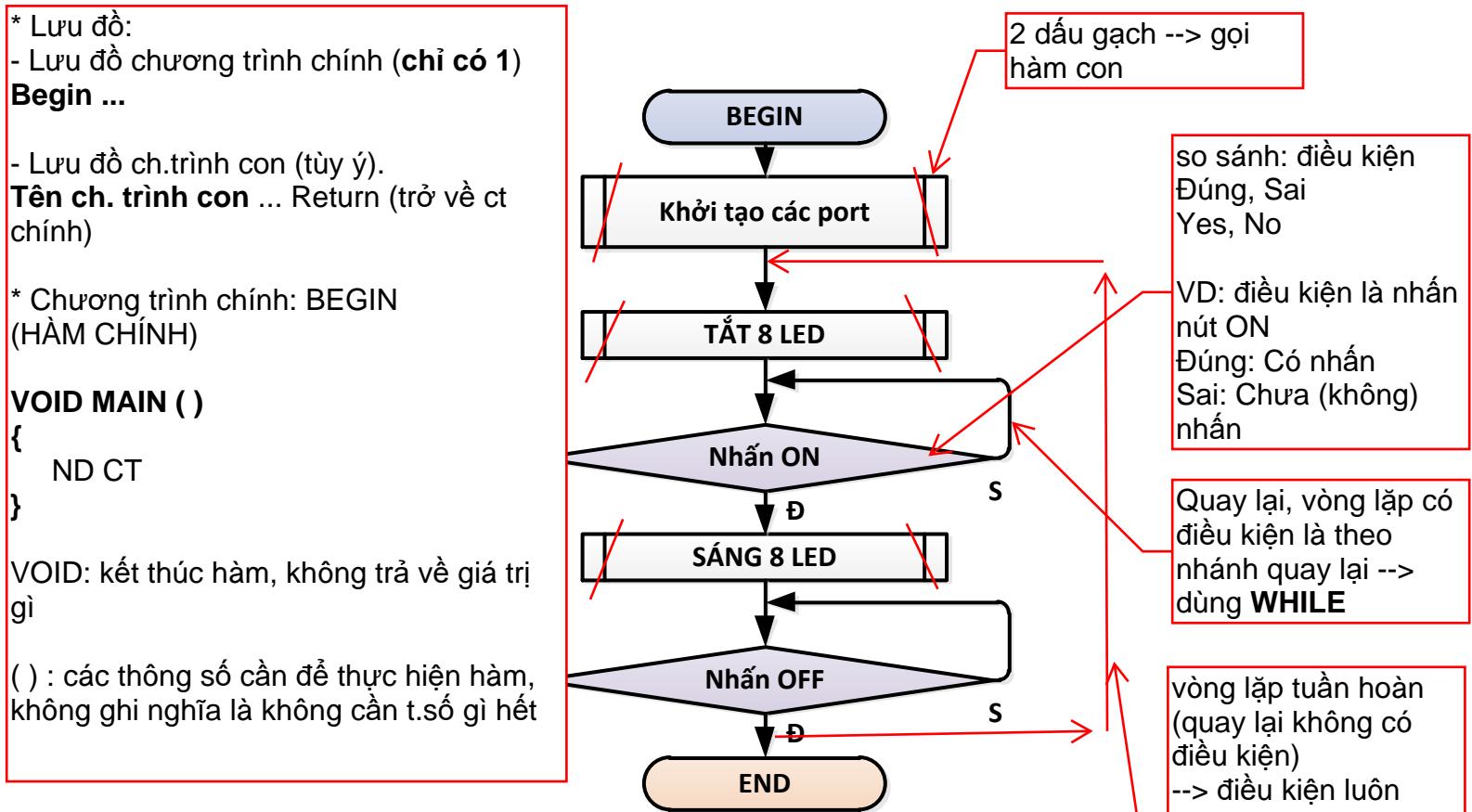
Hình 5-57. Sơ đồ điều khiển led và nút nhấn.

- Nguyên lý hoạt động của phím nhấn:

Hoạt động của 2 nút nhấn: theo sơ đồ kết nối trên thì bình thường 2 nút nhấn mở, 2 ngõ vào có 2 điện trở kéo lên nguồn nên mức logic là 1.

Khi ta nhấn phím thì ngắn mạch ngõ vào xuống mass làm ngõ vào ở mức logic 0, khi nhả phím thì ngõ vào trở về lại mức logic 1.

Khi lập trình ta kiểm tra xem có nhấn phím hay không bằng cách kiểm tra mức logic nếu bằng 1 thì không nhấn, bằng 0 thì có nhấn.



Hình 5-58. Lưu đồ điều khiển led đơn bằng nút nhấn ON-OFF

- Chương trình:

```
#INCLUDE<TV_16F887.C>
#define ON PIN_E
#define OFF PIN_E
VOID MAIN()
{
    SET_TRIS_E(0xFF);
    SET_TRIS_D(0x00);
    WHILE(TRUE)
    {
        OUTPUT_D(0X00);
        WHILE (INPUT(ON));
        OUTPUT_D(0xFF);
        WHILE (INPUT(OFF));
    }
}
```

```
while(input(ON) == 1);
//không thấy giá trị so sánh

    if( ! input(ON)
hoặc if(input(ON) == 0)
// dấu ! so sánh với giá trị 0
```

- Giải thích chương trình:

Lệnh “SET_TRIS_E(0FF)” có chức năng khởi tạo portE là port nhập vì nối với 2 nút nhấn.

Lệnh “SET_TRIS_D(0x00);” có chức năng khởi tạo portD là port xuất.

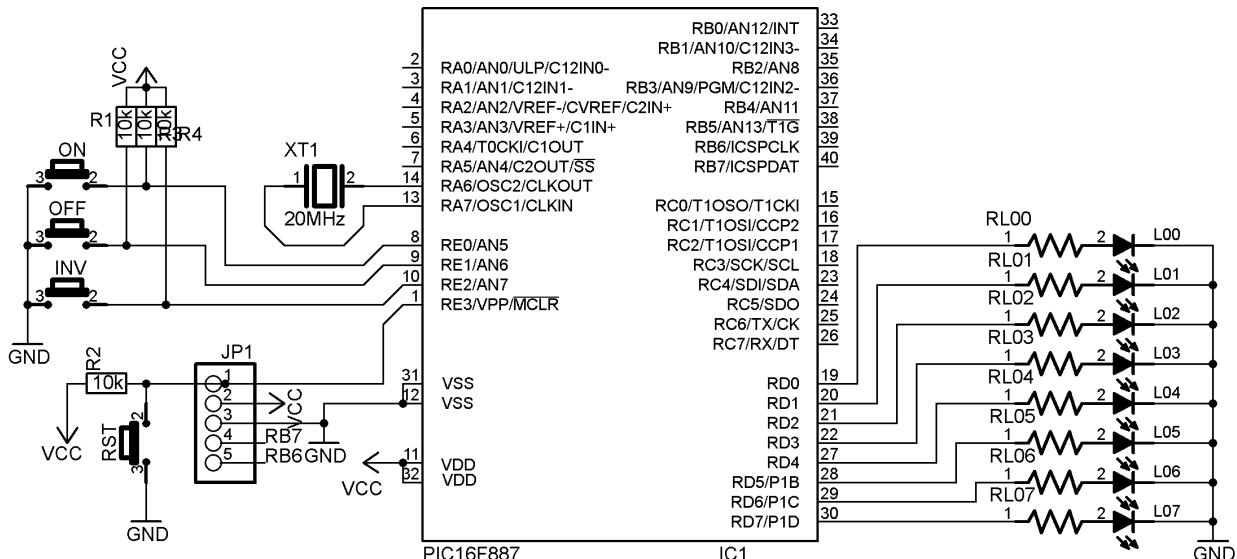
Lệnh “OUTPUT_D(0x00);” có chức năng tắt 8 led

Lệnh “WHILE (INPUT(ON));” có chức năng kiểm tra nút nhấn ON: nếu bằng 1 là không nhấn phím ON thì tiếp tục kiểm tra cho đến khi nhấn phím ON thì vòng lặp while kết thúc và thực hiện lệnh xuất dữ liệu 0xFF ra portD làm 8 led sáng.

Lệnh “WHILE (INPUT(OFF));” có chức năng kiểm tra nút nhấn OFF: nếu bằng 1 là không nhấn phím OFF thì tiếp tục kiểm tra cho đến khi nhấn phím OFF thì vòng lặp while kết thúc và quay về thực hiện lệnh xuất dữ liệu 0x00 ra portD làm 8 led tắt rồi lặp lại từ đầu.

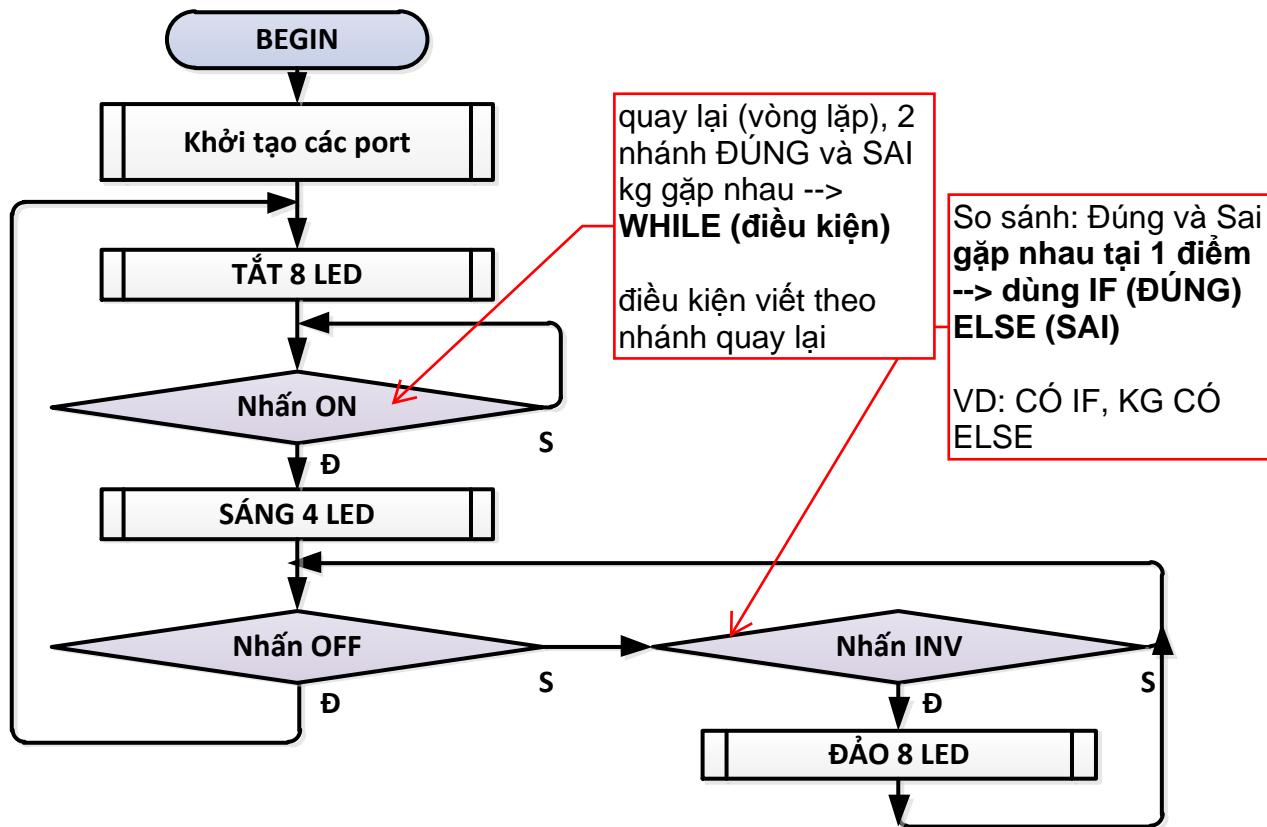
Bài 5-9: Dùng vi điều khiển 16F887 giao tiếp với 8 led đơn và 3 nút nhấn ON, OFF, INV. Khi cấp điện thì 8 led tắt, khi nhấn ON thì 4 led thấp ($D<3:0>$) sáng, khi nhấn INV thì đảo trạng thái của 8 led: 4 sáng thành tắt, 4 tắt thành sáng, khi nhấn OFF thì 8 led tắt.

- Sơ đồ mạch:



Hình 5-59. Sơ đồ điều khiển led và 3 nút nhấn.

- Lưu đồ:



Hình 5-60. Lưu đồ điều khiển led bằng 3 nút ON-OFF-INV.

- Chương trình:

```

#INCLUDE<TV_16F887.C>
#define ON      PIN_E0
#define OFF     PIN_E1
#define INV    PIN_E2
UNSIGNED INT8 X=0;
VOID MAIN()
{
    SET_TRIS_E(0xFF);
    SET_TRIS_D(0x00);
    WHILE (TRUE)
    {
        X=0;    OUTPUT_D(X);
        WHILE (INPUT(ON));
        X= 0XF; OUTPUT_D(X);
        DO
        {
            IF (!INPUT(INV))
            {
                X= ~ X;
                OUTPUT_D(X);
            }
        } WHILE (INPUT(OFF));
    }
}

```

Annotations in the code highlight specific logic points:

- 'X=0; OUTPUT_D(X);' is annotated with 'không ghi giá trị so sánh --> ngầm hiểu so sánh với 1' (not writing comparison value --> implicitly understand comparison with 1).
- 'IF (!INPUT(INV))' is annotated with 'có dấu ! --> so sánh với 0' (has ! --> compare with 0).

- Giải thích chương trình:

Trong vòng lặp while(true) thì lệnh while tiếp theo kiểm tra có nhấn nút ON hay không? Nếu có nhấn thì cho 4 led sáng, nếu không thì chờ nhấn.

Lệnh do while tiến hành nếu có nhấn nút INV thì đảo giá trị 8 led, vòng lặp chỉ thoát khi nhấn phím OFF.

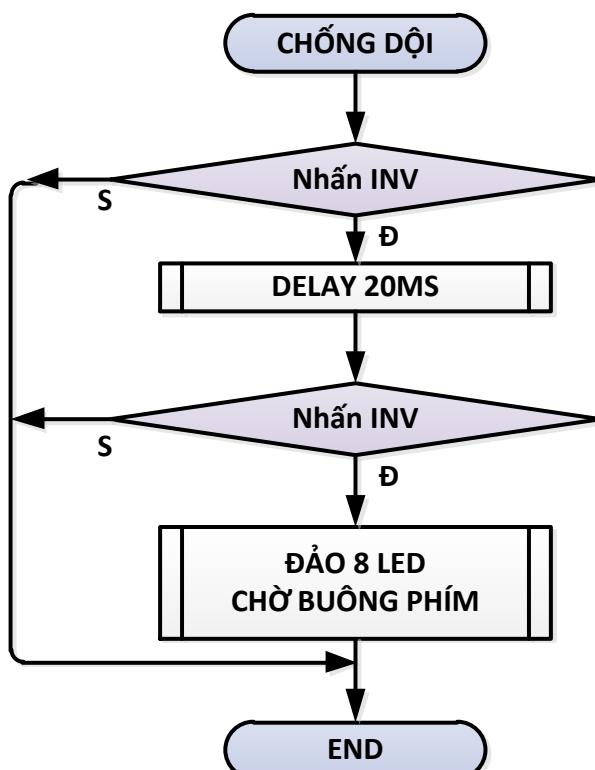
Bài 5-10: Dùng vi điều khiển 16F887 giao tiếp với 8 led đơn và 3 nút nhấn ON, OFF, INV. Khi cấp điện thì 8 led tắt, khi nhấn ON thì 4 led thấp ($D<3:0>$) sáng, khi nhấn INV thì đảo trạng thái của 8 led: 4 sáng thành tắt, 4 tắt thành sáng, khi nhấn OFF thì 8 led tắt. Có chống dội nút nhấn INV.

Trong chương trình này khi chạy thực tế thì do tốc độ của vi điều khiển quá nhanh, 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 xác định rõ ràng thì phải chống dội bằng cách kiểm tra phím nhấn, nếu có thì delay, rồi xử lý 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 như sau:

- Lưu đồ: như hình 5-50.



Hình 5-61. Lưu đồ điều khiển led có chống dội phím INV.

- Chương trình:

```

#define<tv_16f887.c>
#define ON      PIN_E0
#define OFF     PIN_E1
#define INV    PIN_E2
UNSIGNED INT8 X=0;
VOID CHONGDOI_INV ()
{

```

```

IF (!INPUT(INV))
{
    DELAY_MS (20);
    IF (!INPUT(INV))
    {
        X = ~ X;      OUTPUT_D(X);
        WHILE (!INPUT(INV));
    }
}

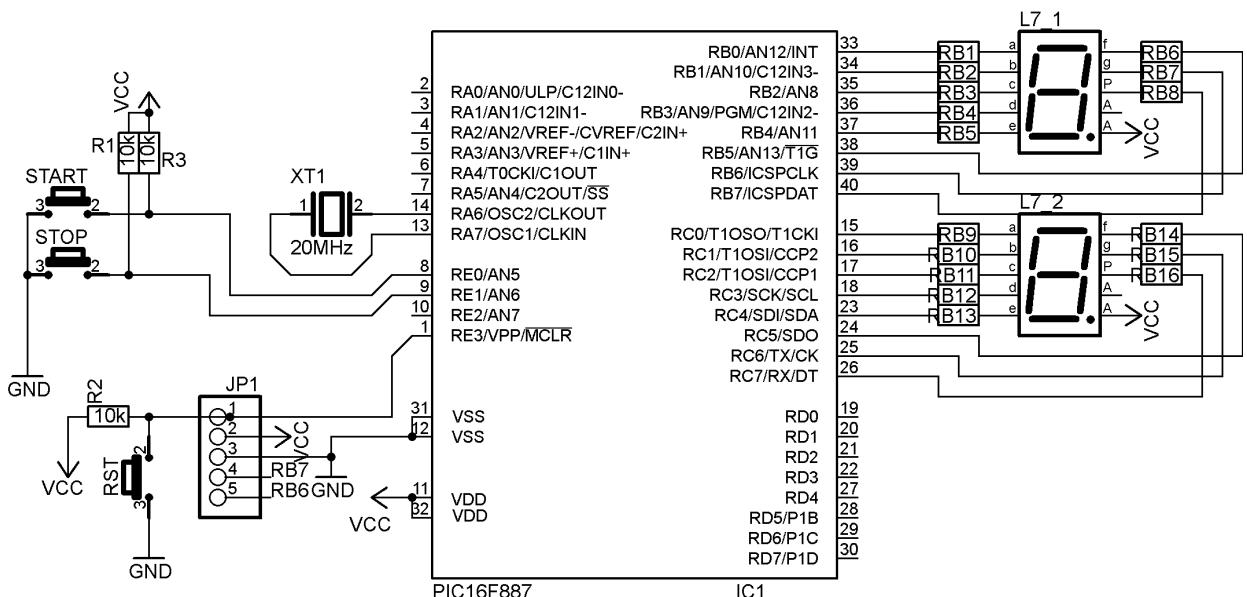
VOID MAIN()
{
    SET_TRIS_E(0xFF);
    SET_TRIS_D(0x00);
    WHILE (TRUE)
    {
        X=0;      OUTPUT_D(X);
        WHILE (INPUT(ON));
        X= 0XF;  OUTPUT_D(X);
        DO
        {
            CHONGDOI_INV();
        }WHILE (INPUT(OFF));
    }
}

```

Bài tập 5-7: Hãy viết chương trình điều khiển 8 led đơn bằng 2 nút nhấn SANG, TAT. Khi nhấn nút SANG thì 8 led sáng dần từ phải sang trái, mỗi lần nhấn thì 1 led sáng, khi nhấn nút TAT thì 8 led tắt dần từ trái sang phải, mỗi lần 1 led. Các bước thực hiện bao gồm: vẽ mạch, viết lưu đồ, viết chương trình và mô phỏng.

Bài 5-11: Mạch đếm thời gian từ 00 đến 99, dùng vi điều khiển PIC 16F887 kết nối với 2 led 7 đoạn anode chung và 2 nút nhấn Start, Stop. Viết chương trình thực hiện chức năng: khi cấp điện thì led hiển thị 00, khi nhấn Start thì mạch đếm từ 00 đến 99, nếu nhấn Stop thì ngừng tại giá trị đang đếm, nhấn Start thì đếm tiếp.

- #### • Sơ đồ mạch:



Hình 5-62. Sơ đồ kết nối 2 port điều khiển 2 led 7 đoạn, 2 nút nhấn.

- Lưu đồ:



Hình 5-63. Lưu đồ đếm có điều khiển bằng nút nhấn Start-Stop.

- Giải thích lưu đồ:

Bài này giống bài đếm đã khảo sát, chỉ thêm phần kiểm tra nút nhấn, biến cho phép ban đầu gán bằng 0 không cho phép đếm, chương trình kiểm tra có nhấn nút Start làm biến cho phép bằng 1 để cho phép mạch đếm. Nếu nhấn nút Stop sẽ làm biến cho phép bằng 0, mạch ngừng tại giá trị đang đếm.

- Chương trình:

```

#include <TV_16F887.C>
#define START PIN_E0
#define STOP PIN_E1
signed int8 DEM;
int1 CHOPHEP = 0;
void main()
{
    set_tris_e(0xFF);
    set_tris_c(0x00);
    set_tris_d(0x00);
    CHOPHEP=0;
    while (true)
    {
        for (DEM=0; DEM<100; DEM++)
        {
            output_c(MA7DOAN[DEM %10]);
            output_d(MA7DOAN[DEM /10]);
            delay_ms(500);
            do
            {
                if (!input(start)) {CHOPHEP=1;}
                else if (!input(stop)) {CHOPHEP=0;}
            }while (CHOPHEP==0);
        }
    }
}
  
```

}
}
}

Bài tập 5-8: Hãy thêm vào bài 5-11 một nút nhấn có tên là INV có chức năng đảo chiều đếm lên, đếm xuống. Đang đếm lên nếu nhấn INV thì sẽ đếm xuống từ giá trị đang đếm, tương tự khi đang đếm xuống mà nhấn INV thì sẽ đếm lên. Các bước thực hiện bao gồm: vẽ mạch, viết lưu đồ, viết chương trình và mô phỏng.

9.1.2 HỆ THỐNG NHIỀU PHÍM

Với cách 1 thì mỗi phím sử dụng 1 ngõ vào để kết nối, 16 phím sẽ dùng 16 ngõ vào như vậy sẽ dùng rất nhiều đường tín hiệu để giao tiếp, cách kết nối dùng ít tín hiệu hơn là kiểu ma trận phím.

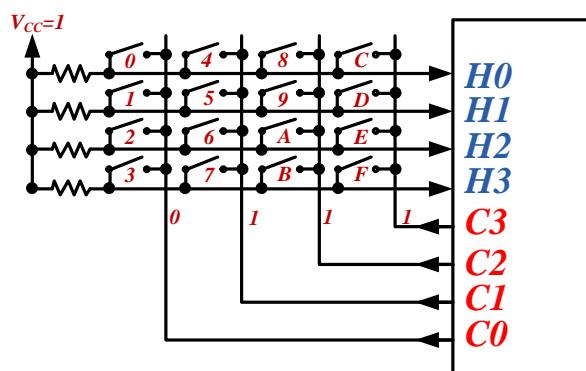
Cách kết nối dạng ma trận thì 16 phím chỉ dùng 8 tín hiệu: 4 cho hàng và 4 cho cột – gọi là ma trận 4×4 sẽ được 16 phím, ma trận 8×8 sẽ được 64 phím.

Tổng quát ma trận bàn phím $m \times n$ thì số đường tín hiệu bằng " $m+n$ ", số phím bằng " $m \times n$ "

- Ưu điểm của cách kết nối theo dạng ma trận là tiết kiệm đường điều khiển.
- Khuyết điểm: chương trình phức tạp.

Khảo sát bàn phím ma trận $4 \times 4 = 16$ phím:

Sơ đồ mạch bàn phím như hình 5-64:



Hình 5-64. Bàn phím ma trận 4×4 .

Trong ma trận 4×4 thì có 4 hàng và 4 cột, hàng được chọn là tín hiệu vào – cột được chọn là tín hiệu ra, hàng thì treo lên nguồn Vcc qua điện trở nên mức logic của hàng luôn là mức 1.

Các phím nhấn thường hở nên 4 hàng luôn ở mức 1 hay $H_3H_2H_1H_0 = 1111$.

Cột là tín hiệu ra nên chúng ta điều khiển xuất dữ liệu ra cột tùy ý.

Để phân biệt các phím thì mỗi phím có 1 tên được đánh theo số thập lục phân từ '0' đến 'F'.

Để xem có phím nào nhấn hay không ta tiến hành quét từng cột bằng cách cho 1 cột ở mức 0, 3 cột còn lại ở mức 1 và kiểm tra tất cả các hàng, nếu tất cả các hàng vẫn ở mức logic 1 tức là không có nhấn phím, nếu có 1 hàng xuống mức 0 thì đã có nhấn phím. Cụ thể như sau:

Quét cột thứ 0

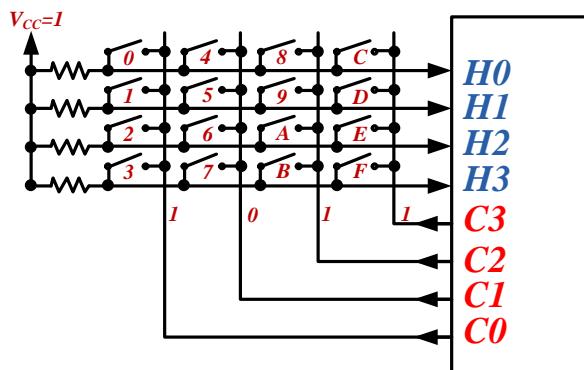
Xuất dữ liệu ra cột là: $C_3C_2C_1C_0 = 1110$ như hình 5-54, kiểm tra hàng nào bằng không?

- Nếu bằng $H0 = 0$ thì đã nhấn phím số ‘0’.
- Nếu bằng $H1 = 0$ thì đã nhấn phím số ‘1’.
- Nếu bằng $H2 = 0$ thì đã nhấn phím số ‘2’.
- Nếu bằng $H3 = 0$ thì đã nhấn phím số ‘3’.
- Nếu không có phím nhấn nào ở cột C_0 được nhấn thì phải quét cột tiếp theo.

Quét cột thứ 1

Xuất dữ liệu ra cột là: $C_3C_2C_1C_0 = 1101$ như hình 5-55, kiểm tra hàng nào bằng không?

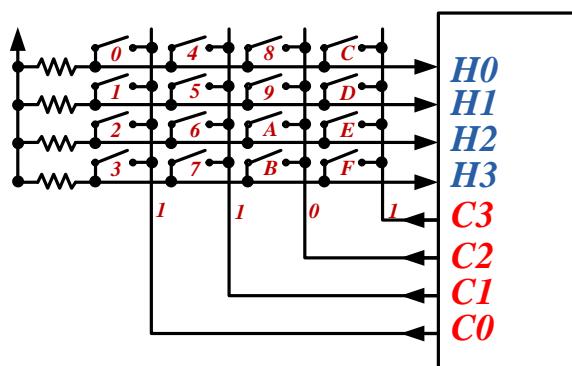
- Nếu bằng $H0 = 0$ thì đã nhấn phím số ‘4’.
- Nếu bằng $H1 = 0$ thì đã nhấn phím số ‘5’.
- Nếu bằng $H2 = 0$ thì đã nhấn phím số ‘6’.
- Nếu bằng $H3 = 0$ thì đã nhấn phím số ‘7’.
- Nếu không có phím nhấn nào ở cột C_1 được nhấn thì phải quét cột tiếp theo.



Hình 5-65. Bàn phím ma trận 4×4 với cột C_1 bằng 0.

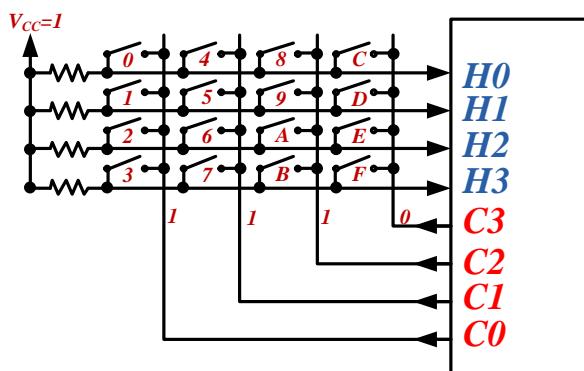
Quét cột thứ 2

- Xuất dữ liệu ra cột là: $C_3C_2C_1C_0 = 1011$ như hình 5-56, kiểm tra hàng nào bằng không?
- Nếu bằng $H0 = 0$ thì đã nhấn phím số ‘8’.
- Nếu bằng $H1 = 0$ thì đã nhấn phím số ‘9’.
- Nếu bằng $H2 = 0$ thì đã nhấn phím số ‘A’.
- Nếu bằng $H3 = 0$ thì đã nhấn phím số ‘B’.
- Nếu không có phím nhấn nào ở cột C_2 được nhấn thì phải quét cột tiếp theo.



Hình 5-66. Bàn phím ma trận 4×4 với cột C_2 bằng 0.**Quét cột thứ 3**

- Xuất dữ liệu ra cột là: $C_3C_2C_1C_0 = 0111$ như hình 5-57, kiểm tra hàng nào bằng không?
- Nếu bằng $H0 = 0$ thì đã nhấn phím số ‘C’.
- Nếu bằng $H1 = 0$ thì đã nhấn phím số ‘D’.
- Nếu bằng $H2 = 0$ thì đã nhấn phím số ‘E’.
- Nếu bằng $H3 = 0$ thì đã nhấn phím số ‘F’.
- Nếu không có phím nhấn nào ở cột C_3 đến đây xem như kết thúc.

Hình 5-67. Bàn phím ma trận 4×4 với cột C_3 bằng 0.

Tên các phím để chúng ta phân biệt, còn chương trình phân biệt các phím bằng mã, mỗi phím có 1 mã phím do phần mềm lập trình xây dựng, để đơn giản mỗi phím được đặt mã như sau:

Phím số ‘0’ có mã là 00H, phím số ‘1’ có mã là 01H, ... phím ‘F’ có mã là 0FH.

Hoạt động quét bàn phím được thực hiện theo lưu đồ như hình 5-68.

Giải thích lưu đồ:

Mã quét cột là 4 bit thấp bao gồm các trạng thái: “1110B”, “1101B”, “1011B”, “0111B” kết hợp với 4 bit cao của hàng luôn là “1111B” nên dữ liệu kết hợp theo byte viết dạng số hex là “FEH”, “FDH”, “FBH”, “F7H”.

Gán cột bắt đầu từ 0, hàng gán bằng FFH, mã phím ban đầu gán bằng FFH.

Xuất lần lượt từng mã quét cột ra port quét phím, tiến hành kiểm tra từng hàng xem hàng nào bằng 0 thì phím tương ứng được nhấn. Khi đó ta gán giá trị tương ứng cho từng hàng.

Cuối cùng kiểm tra hàng xem có bằng FFH hay không, nếu bằng thì không phím nào được nhấn trong lúc thực hiện chương trình quét, thoát với mã phím bằng FFH, nếu khác FFH thì đã có nhấn phím, tiến hành tính toán mã phím và thoát với mã phím là của phím đã nhấn.

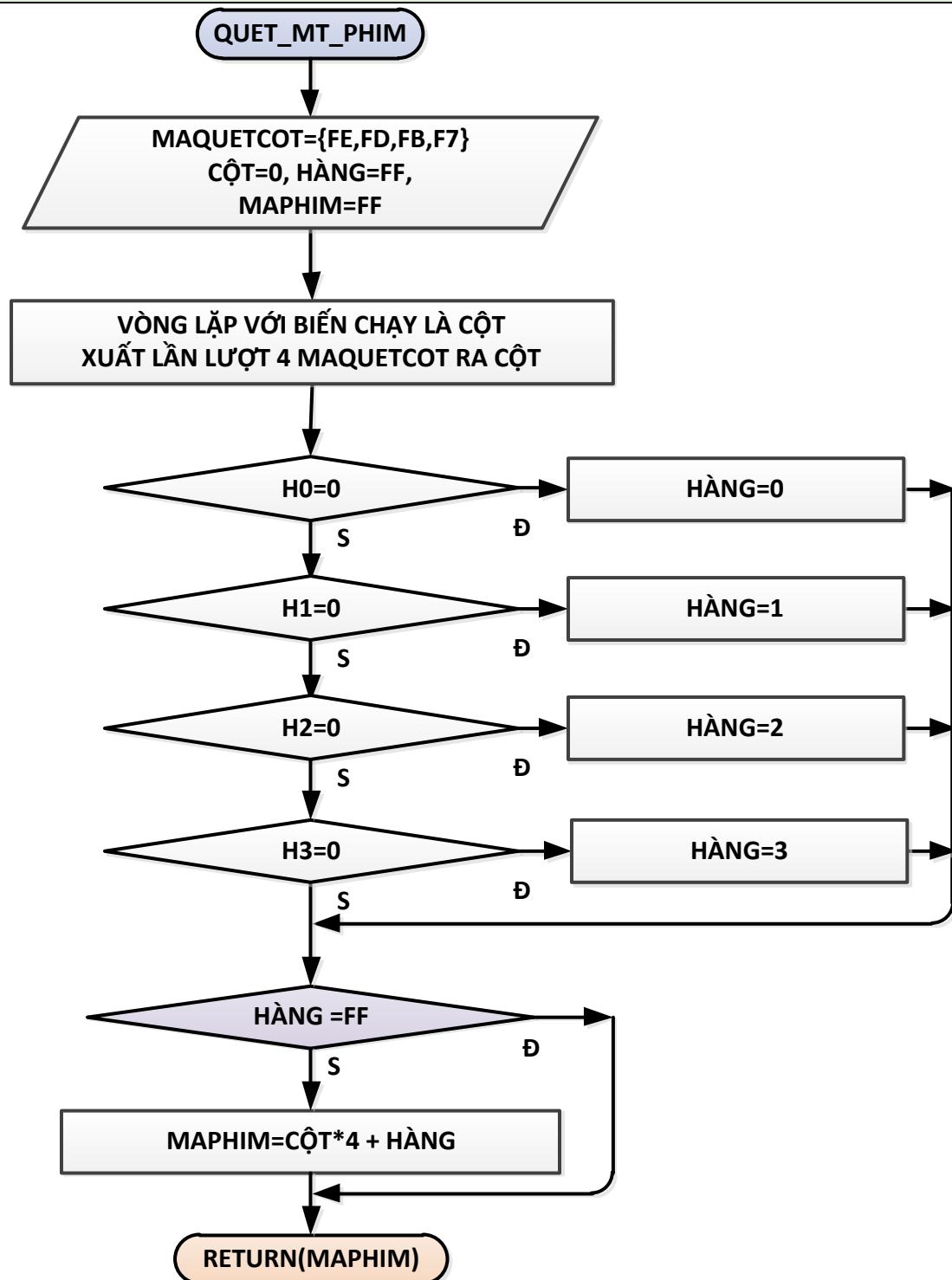
Nếu quét cột thứ 0 thì cột bằng 0, theo công thức mã phím sẽ có giá trị từ 0H đến 3H.

Nếu quét cột thứ 1 thì cột bằng 1, theo công thức mã phím sẽ có giá trị từ 4H đến 7H.

Nếu quét cột thứ 2 thì cột bằng 2, theo công thức mã phím sẽ có giá trị từ 8H đến BH.

Nếu quét cột thứ 3 thì cột bằng 3, theo công thức mã phím sẽ có giá trị từ CH đến FH.

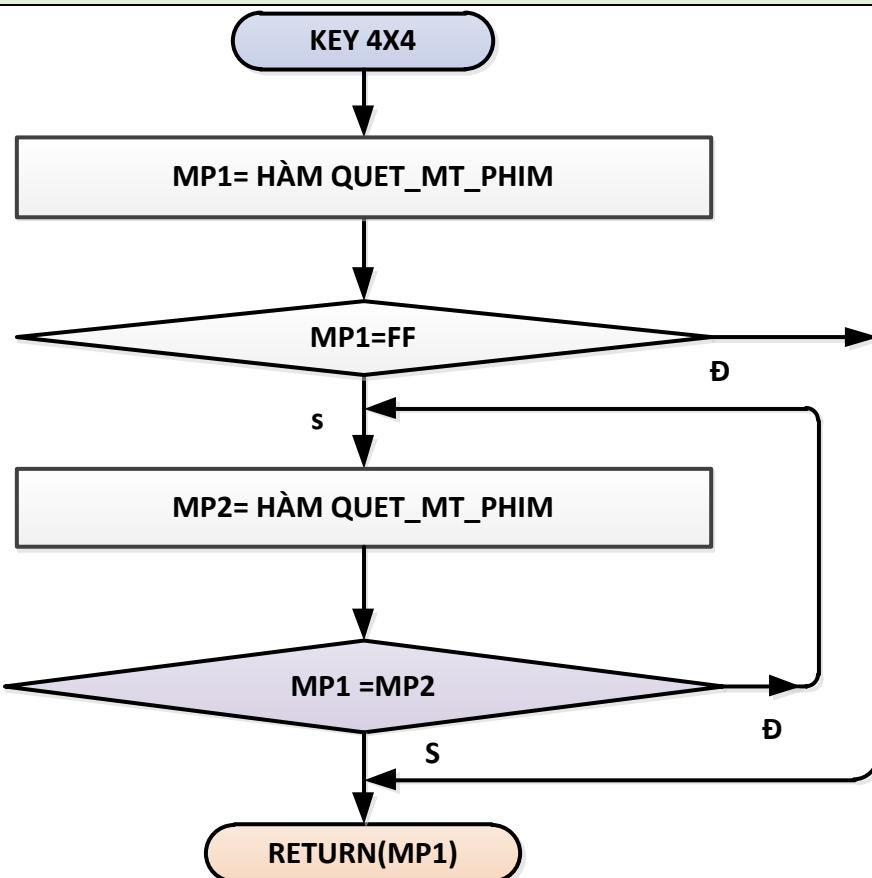
Khi quét phím thì sinh ra hiện tượng dội phím nên ta phải thực hiện thêm phần chống dội.

Hình 5-68. Lưu đồ quét bàn phím ma trận 4×4 .

Trong lưu đồ này gọi hàm quét phím ma trận 4×4 , sau đó gán mã phím cho biến MP1, kiểm tra nếu MP1 bằng FFH thì không có nhấn phím nên không cần chống dội và thoát.

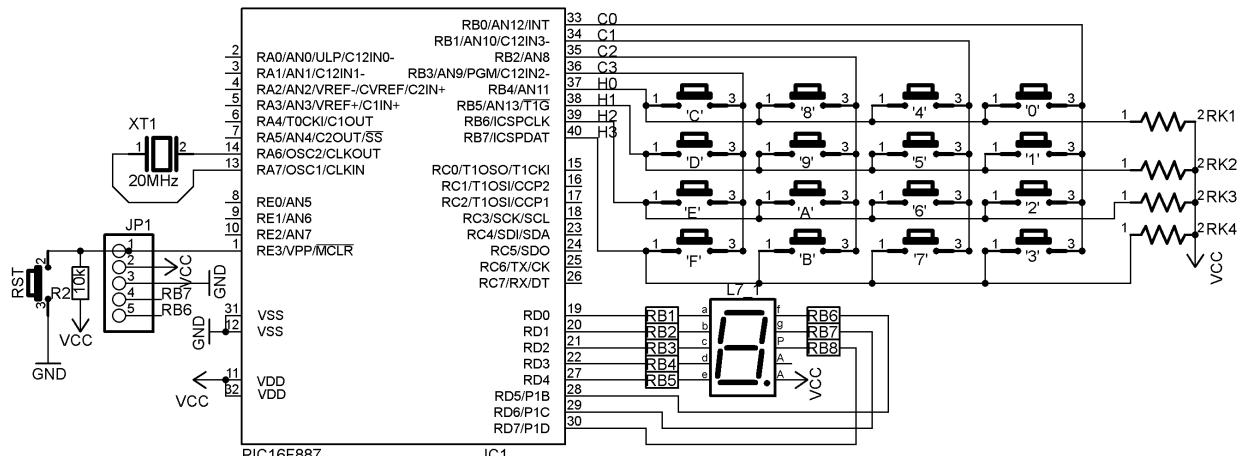
Nếu có nhấn thì gọi hàm quét phím và gán mã phím cho biến MP2, so sánh 2 mã phím: nếu 2 mã phím bằng nhau thì có nghĩa là ta chưa buông phím đã nhấn nên quay lại kiểm tra chờ cho đến khi ta buông phím. Khi ta buông phím thì 2 mã phím sẽ khác nhau và chương trình sẽ thoát và trả về mã phím nhấn lần đầu là MT1.

Để hiểu chương trình quét bàn phím ma trận ta khảo sát bài ứng dụng theo sau.

Hình 5-69. Lưu đồ quét bàn phím ma trận 4×4 có chổng dội.

Bài 5-12: Dùng vi điều khiển PIC 16F887 giao tiếp với 1 led 7 đoạn và ma trận phím 4×4 . Viết chương trình quét bàn phím ma trận, hiển thị tên của phím trên led 7 đoạn.

- Sơ đồ mạch:



Hình 5-70. Vi điều khiển giao tiếp bàn phím ma trận.

Trong sơ đồ dùng portB giao tiếp với bàn phím ma trận 4×4 và portD điều khiển 1 led 7 đoạn.

- Lưu đồ:



Hình 5-71. Lưu đồ quét hiển thị ma trận phím.

- Chương trình chính:

```

#include<TV_16F887.C>
#include<TV_KEY4X4.C>
SIGNED INT8 MP;
VOID MAIN()
{
    SET_TRIS_D(0x00);      OUTPUT_D(0x7F);
    SET_TRIS_B(0xF0);      PORT_B_PULLUPS(0XF0);
    WHILE(TRUE)
    {
        MP= KEY_4X4();
        IF (MP!=0xFF)
        {
            OUTPUT_D(MA7DOAN[MP]);
        }
    }
}
  
```

- Chương trình con quét phím: "TV_KEY4X4.C"

```

CONST UNSIGNED CHAR MAQUETCOT[4]={0xFE,0xFD,0xFB,0xF7};
UNSIGNED INT QUET_MT_PHIM()
{
    SIGNED     INT8     MAPHIM, HANG, COT;
    MAPHIM=HANG=0XF;
    FOR(COT=0;COT<4;COT++)
    {
        OUTPUT_B(MAQUETCOT[COT]);
        IF          (!INPUT(PIN_B4)) {HANG=0; BREAK;}
        ELSE        IF          (!INPUT(PIN_B5)) {HANG=1; BREAK;}
        ELSE        IF          (!INPUT(PIN_B6)) {HANG=2; BREAK;}
        ELSE        IF          (!INPUT(PIN_B7)) {HANG=3; BREAK;}
    }
}
  
```

```

        }
        IF (HANG!=0xFF)      MAPHIM    = COT*4 + HANG;
        RETURN(MAPHIM);
    }

UNSIGNED INT KEY_4X4()
{
    UNSIGNED INT8 MPT1,MPT2;
    MPT1=QUET_MT_PHIM();
    IF (MPT1!=0xFF)
    {
        DELAY_MS(10);
        MPT1=QUET_MT_PHIM();
        DO{MPT2=QUET_MT_PHIM();}WHILE (MPT2==MPT1);
    }
    RETURN(MPT1);
}

```

- Giải thích chương trình chính:

Các lệnh khởi tạo portB, D và điện trở kéo lên, theo sơ đồ thì dùng 4 điện trở bên ngoài thì không cần nhưng nếu ta dùng bàn phím 4x4 có sẵn chưa có điện trở thì ta phải dùng điện trở kéo lên bên trong.

Lệnh “OUTPUT_D(0x7F);” có chức năng làm dấu chấm của led 7 đoạn sáng để báo hiệu chương trình đã hoạt động.

Lệnh “MP=KEY_4X4();” có chức năng gọi chương trình con quét phím ma trận có chống dội. Nếu có nhấn thì tiến hành giải mã 7 đoạn và gửi ra portD để nhìn thấy mã phím dạng số hex. Các mã 7 đoạn của 16 số hex lưu trong file thư viện.

Nếu không nhấn phím nào thì quay trở lại chờ nhấn phím.

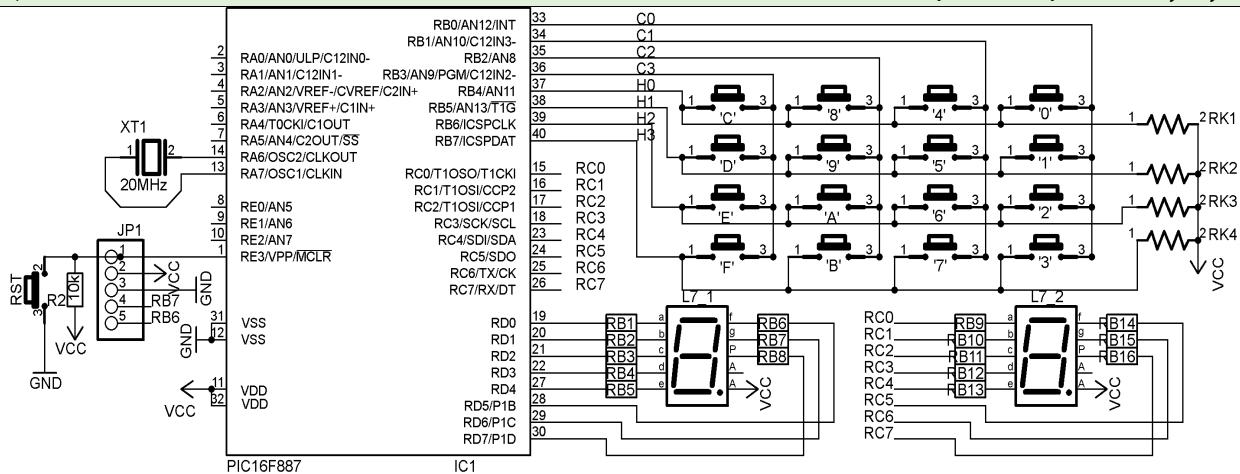
- Giải thích chương trình con:

Chương trình con KEY_4x4 có chức năng gọi chương trình con quét phím và chống dội.

Hai chương trình con này giống như lưu đồ đã viết và đã giải thích.

Bài 5-13: Dùng vi điều khiển PIC 16F887 giao tiếp với 2 led 7 đoạn và ma trận phím 4x4. Viết chương trình quét bàn phím ma trận, hiển thị mã phím nhấn và dịch. Đầu tiên thì hiển thị dấu chấm ở led 7 đoạn thứ 0, khi nhấn phím bất kỳ ví dụ là 3 thì dấu chấm sẽ dịch sang led 7 đoạn thứ 1, số 3 sẽ hiển thị ở led thứ 0, nếu nhấn tiếp phím số 5 thì số 3 của led thứ 0 sẽ chuyển sang led thứ 1, số 5 sẽ hiển thị ở led thứ 0.

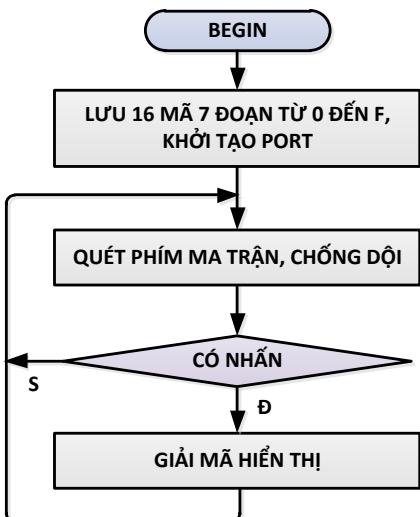
- Sơ đồ mạch:



Hình 5-72. Vị trí điều khiển giao tiếp bàn phím ma trận và 2 led 7 đoạn.

Trong sơ đồ dùng portB giao tiếp với bàn phím ma trận 4×4 , portC và D điều khiển 2 led 7 đoạn.

- Lưu đồ:



Hình 5-73. Lưu đồ quét ma trận phím và hiển thị mã phím.

- Chương trình chính:

```

#include<TV_16F887.C>
#include<TV_KEY4X4.C>
SIGNED INT8 MP, SOTHU1, SOTHU0;
VOID MAIN()
{
    SET_TRIS_D(0x00);
    SET_TRIS_B(0xF0);
    SOTHU0=0XF;
    OUTPUT_D(SOTHU0);
    WHILE(TRUE)
    {
        MP= KEY_4X4();
        IF (MP!=0xFF)
        {
            SOTHU1=SOTHU0;
            SOTHU0=MA7DOAN[MP];
            OUTPUT_D(SOTHU0);
            OUTPUT_C(SOTHU1);
        }
    }
}
    
```

}
}

- Giải thích chương trình chính:

Các lệnh khởi tạo portB, C, D và điện trở kéo lên. Gán 2 biến số thứ 1 bằng FFH, số thứ 0 bằng 7FH khi gởi ra portC và D thì led thứ 1 tắt, led thứ 0 sáng dấu chấm thập phân để cho biết chương trình đã chạy.

Gọi chương trình quét phím có chống dội và kiểm tra mã phím nếu có nhấn thì tiến hành chuyển mã hiển thị của led thứ 0 sang cho led thứ 1, giải mã 7 đoạn của phím mới nhấn, tiến hành gởi 2 mã ra 2 led.

Quay lại làm tương tự khi ta nhấn các phím khác.

Bài tập 5-9: Hãy hiệu chỉnh bài 5-13 sau cho chương trình chỉ cho phép nhấn các phím số thập phân từ 0 đến 9. Các bước thực hiện bao gồm: viết lưu đồ, viết chương trình và mô phỏng.

Bài tập 5-10: Từ bài tập 5-9 thêm yêu cầu khi nhấn phím C thì có chức năng xoá các số đang hiển thị và hiển thị lại dấu chấm thập phân. Các bước thực hiện bao gồm: viết lưu đồ, viết chương trình và mô phỏng.

Bài tập 5-11: Từ bài tập 5-10 thêm yêu cầu khi nhấn phím D thì có chức năng xoá bỏ phím nhấn sau cùng và hiển thị trở lại 2 giá trị của 2 led đã hiển thị trước đó. Các bước thực hiện bao gồm: viết lưu đồ, viết chương trình và mô phỏng.

Chú ý: lần đầu khi mới cấp điện thì không có hiệu lực cho phím D.

Bài tập 5-12: Hãy hiệu chỉnh bài 5-13 sau cho chương trình khi nhấn các phím số thập phân từ 0 đến 9 thì hiển thị trên ở led thứ 0, khi nhấn các phím từ A đến F thì hiển thị giá trị thập phân ở 2 led, ví dụ ta nhấn phím F thì không hiển thị F mà hiển thị 15. Các bước thực hiện bao gồm: viết lưu đồ, viết chương trình và mô phỏng.

5.10 CÁC ỨNG DỤNG ĐIỀU KHIỂN LCD

5.10.1 GIỚI THIỆU LCD

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. 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 16×2 có hình dáng như hình 5-74.



Hình 5-74. Hình ảnh của LCD.

5.10.2 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, 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.

Bảng 5-3. Các chân của LCD.

Thứ tự	Tên tín hiệu	I/O	Mô tả
1	V _{SS}	Nguồn	GND
2	V _{DD}	Nguồn	+5V
3	Vo	Điện áp	Điều khiển ánh sáng nền
4	RS	INPUT	Register Select
5	R/W	INPUT	Read/Write
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	A	I	Nguồn dương +5V
16	K	I	GND

Trong 16 chân của LCD được chia ra làm 3 dạng tín hiệu như sau:

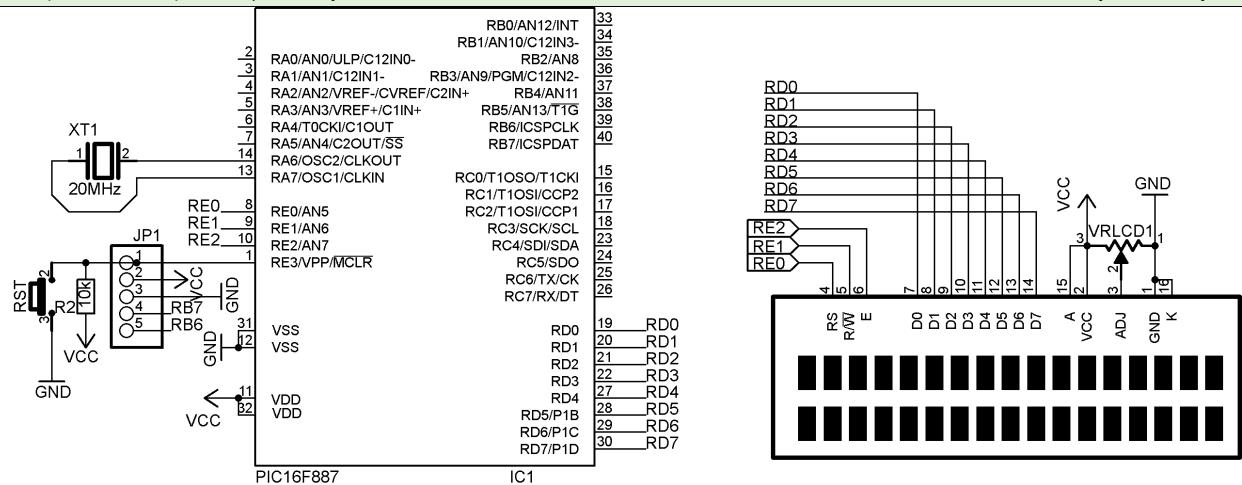
Các chân cấp nguồn: Chân số 1 là chân nối mass (0V), chân thứ 2 là Vdd nối với nguồn +5V. Chân thứ 3 dùng để chỉnh contrast thường nối với biến trở.

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.

5.10.3 SƠ ĐỒ MẠCH GIAO TIẾP VI ĐIỀU KHIỂN VỚI LCD

Trong phần này sẽ trình bày phần giao tiếp vi điều khiển PIC 16F887 với LCD như hình 5-75.



11 Read data	1	1	Read data	Reads data from CCGRAM or DDRAM	3
12 Write data	1	0	Write data	Write data to CCGRAM or DDRAM	3

Lệnh xoá màn hình “Clear Display”: khi thực hiện lệnh này thì LCD sẽ bị xoá và bộ đếm địa chỉ được xoá về 0.

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 trỏ về vị trí gốc đã bị dịch trước đó. Nội dung bộ nhớ RAM hiển thị DDRAM không bị thay đổi.

Lệnh thiết lập lỗi vào “Entry mode set”: lệnh này dùng để thiết lập lỗi vào cho các kí tự hiển thị, bit ID = 1 thì con trỏ tự động tăng lên 1 mỗi khi có 1 byte dữ liệu ghi vào bộ hiển thị, khi ID = 0 thì con trỏ sẽ không tăng; dữ liệu mới sẽ ghi đè lên dữ liệu cũ. Bit S = 1 thì cho phép dịch chuyển dữ liệu mỗi khi nhận 1 byte hiển thị.

Lệnh điều khiển con trỏ hiển thị “Display Control”: lệnh này dùng để điều khiển con trỏ (cho hiển thị thì bit D = 1, tắt hiển thị thì bit D = 0), tắt mở con trỏ (mở con trỏ thì bit C = 1, tắt con trỏ thì bit C = 0), và nhấp nháy con trỏ (cho nhấp nháy thì bit B = 1, tắt thì bit B = 0).

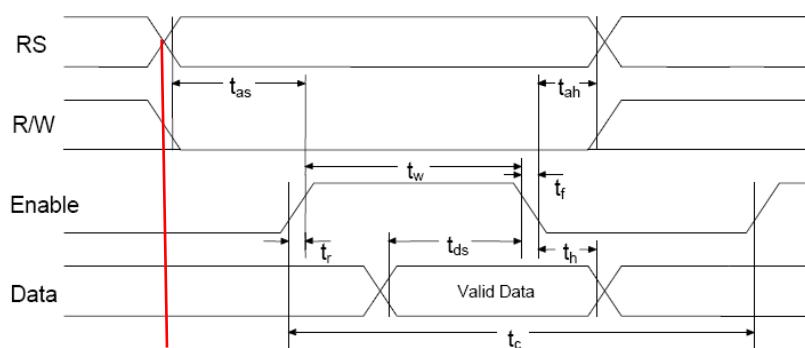
Lệnh di chuyển con trỏ “Cursor /Display Shift”: lệnh này dùng để điều khiển di chuyển con trỏ hiển thị dịch chuyển (SC = 1 cho phép dịch chuyển, SC = 0 thì không cho phép), hướng dịch chuyển (RL = 1 thì dịch phải, RL = 0 thì dịch trái). Nội dung bộ nhớ DDRAM vẫn không đổi.

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.

Dang sóng các tín hiệu khi thực hiện ghi dữ liệu vào LCD như hình 5-76:



Hình 5-76. Dạng sóng điều khiển của LCD.

Nhìn vào dang sóng ta có thể thấy được trình tự điều khiển như sau:

- Điều khiển tín hiệu RS.
 - Điều khiển tín hiệu R/W xuống mức thấp.

- Điều khiển tín hiệu E lên mức cao để cho phép.
- Xuất dữ liệu D7÷D0.
- Điều khiển tín hiệu E về mức thấp.
- Điều khiển tín hiệu R/W lên mức cao trở lại.

5.10.5 ĐỊA CHỈ CỦA TỪNG KÍ TỰ TRÊN LCD

LCD16x2 có 2 hàng mỗi hàng 16 kí tự.

Hàng 1: kí tự tận cùng bên trái có địa chỉ là 0x80, kế là 0x81, cuối cùng là 0x8F.

Hàng 2: kí tự tận cùng bên trái có địa chỉ là 0xC0, kế là 0xC1, cuối cùng là 0xCF.

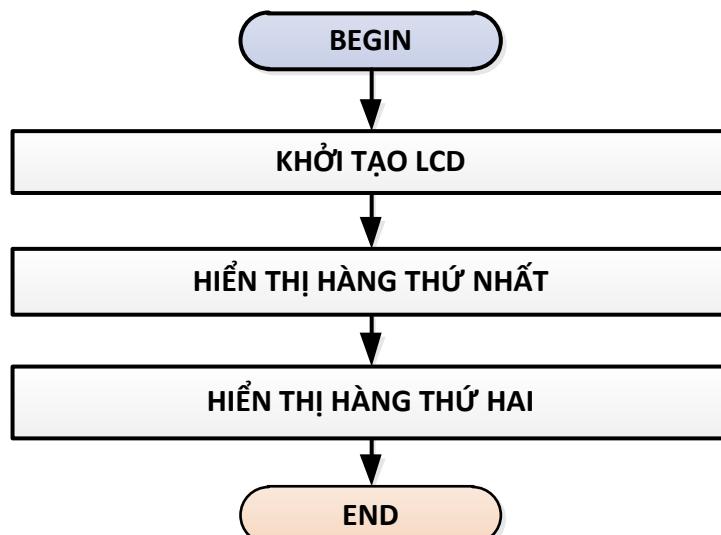
Bảng 5-5. Địa chỉ của từng kí tự.

Địa chỉ	80	81	82	83	84	85	86	87	88	89	8A	8B	8C	8D	8E	8F
Kí tự hàng 1		D	H		S	P		K	Y		T	H	U	A	T	
Kí tự hàng 2		T	P		H	O		C	H	I		M	I	N	H	
Địa chỉ	C0	C1	C2	C3	C4	C5	C6	C7	C8	C9	CA	CB	CC	CD	CE	CF

5.10.6 CÁC CHƯƠNG TRÌNH HIỂN THỊ TRÊN LCD

Bài 5-14: Sử dụng mạch giao tiếp vi điều khiển PIC16F887 với LCD 16×2 ở hình 5-65, hãy viết chương trình hiển thị 2 hàng thông tin như trong bảng 5-4.

- Sơ đồ mạch: giống như hình 5-75.
- Lưu đồ:



Hình 5-77. Lưu đồ hiển thị thông tin trên 2 hàng.

- Chương trình chính:

```

#include <TV_16F887.C>
#include <TV_LCD.C>
UNSIGNED INT I;
const unsigned char HANG1[16]={" DH SP KY THUAT "};
const unsigned char HANG2[16]={" TP HO CHI MINH "};
  
```

```

VOID MAIN()
{
    SET_TRIS_E(0x00);
    SET_TRIS_D(0x00);
    LCD_SETUP();

    LCD_COMMAND(ADDR_LINE1);
    DELAY_US(10);
    FOR (I=0; I<16; I++) { LCD_DATA(HANG1[I]); }

    LCD_COMMAND(ADDR_LINE2);
    DELAY_US(10);
    FOR (I=0; I<16; I++) { LCD_DATA(HANG2[I]); }
    while(TRUE) {}
}

```

- Giải thích chương trình chính:

Khai báo 2 thư viện: “TV_16F887.C” đã viết, “TV_LCD.C” viết bên dưới.

Khởi tạo các port, khởi tạo LCD.

Khởi tạo địa chỉ hàng 1, thực hiện vòng lặp for cho hiển thị lần lượt 16 ký tự của hàng 1.

Khởi tạo địa chỉ hàng 2, thực hiện vòng lặp for cho hiển thị lần lượt 16 ký tự của hàng 2.

Thực hiện vòng lặp while: nhảy tại chỗ.

- Chương trình thư viện “TV_LCD.C”:

```

#ifndef LCD_RS
#define LCD_RS PIN_E0
#endif

#ifndef LCD_RW
#define LCD_RW PIN_E1
#endif

#ifndef LCD_E
#define LCD_E PIN_E2
#endif

#ifndef OUTPUT_LCD
#define OUTPUT_LCD OUTPUT_D
#endif

#define FUNCTION_SET 0X38 //GIAO TIEP 8 BIT, 2 HANG
#define DISPLAY_CONTROL 0XF
#define CLEAR_DISPLAY 0X01
#define ENTRY_MODE 0X06
#define SHIFT_LEFT 0X18
#define SHIFT_RIGHT 0X1C
#define ADDR_LINE1 0X80
#define ADDR_LINE2 0XC0

void LCD_COMMAND(UNSIGNED_CHAR MDK) //RS = 0
{
    OUTPUT_LOW(LCD_RS);           OUTPUT_LCD(MDK);
    OUTPUT_HIGH(LCD_E);          DELAY_US(20);
}

```

IF NOT DEFINE
NẾU CHƯA ĐỊNH
NGHĨA

```

        OUTPUT_LOW(LCD_E) ;           DELAY_US(20);
}

void LCD_DATA(UNSIGNED CHAR MHT)           //RS = 1
{
    RS = 1; //thanh ghi hiển thị
    OUTPUT_HIGH(LCD_RS);
    OUTPUT_HIGH(LCD_E);
    OUTPUT_LOW(LCD_E);
}

VOID LCD_SETUP()
{
    OUTPUT_LOW(LCD_E);
    OUTPUT_LOW(LCD_RS);
    OUTPUT_LOW(LCD_RW);
    LCD_COMMAND(FUNCTION_SET);
    LCD_COMMAND(DISPLAY_CONTROL);
    LCD_COMMAND(CLEAR_DISPLAY);
    LCD_COMMAND(ENTRY_MODE);
}

```

lcd_command('clear_display')
delay_ms(2);
lcd_command('function set')
delay_ms(1);

DELAY_MS(1);

DELAY_MS(2);

- Giải thích chương trình thư viện LCD:

Ba lệnh đầu tiên có chức năng kiểm tra xem biến LCD_RS chưa định nghĩa (IFNDEF được viết tắt của các từ if not define: nếu chưa định nghĩa) thì tiến hành định nghĩa. Nếu đã định nghĩa rồi thì không định nghĩa nữa.

Với cách thức này có chức năng: với các bài LCD này thì ta đã dùng portD và các bit của portE nhưng sang các ứng dụng khác, ta dùng port khác với portD và E, ví dụ không dùng portD mà dùng portC thì ta chỉ cần định nghĩa portC trong chương trình chính, không cần thay đổi các định nghĩa trong chương trình con LCD đã viết.

Tương tự cho các định nghĩa khác.

Có thể viết định nghĩa các port đơn giản như sau:

```

#define LCD_RS      PIN_E0
#define LCD_RW      PIN_E1
#define LCD_E       PIN_E2

```

#define OUTPUT_LCD OUTPUT_C

trong chương trình của mình mà không cần thay đổi thư viện LCD

#ifndef --> if not define

Nhưng nếu viết như thế mà ta định nghĩa lại các tín hiệu thì sẽ báo lỗi.

Chương trình con “LCD_COMMAND” thực hiện chức năng gửi lệnh ra LCD.

Chương trình con “LCD_DATA” thực hiện chức năng gửi dữ liệu ra LCD để hiển thị.

Chương trình con “LCD_SETUP” thực hiện chức năng khởi tạo LCD.

Chương trình hiển thị cách thứ 2:

```

#include <TV_16F887.C>
#include <TV_LCD.C>

VOID MAIN()
{
    SET_TRIS_E(0x00);
    SET_TRIS_D(0x00);
    LCD_SETUP();

    LCD_COMMAND(ADDR_LINE1);   0x80 - vị trí đầu hàng 1 DELAY_US(10);
    LCD_DATA(" DH-SP-KT-TPPHCM");
    LCD_COMMAND(ADDR_LINE2);   DELAY_US(10);
    LCD_DATA(" WELCOME TO VDK ");
}

while(TRUE) {}

```

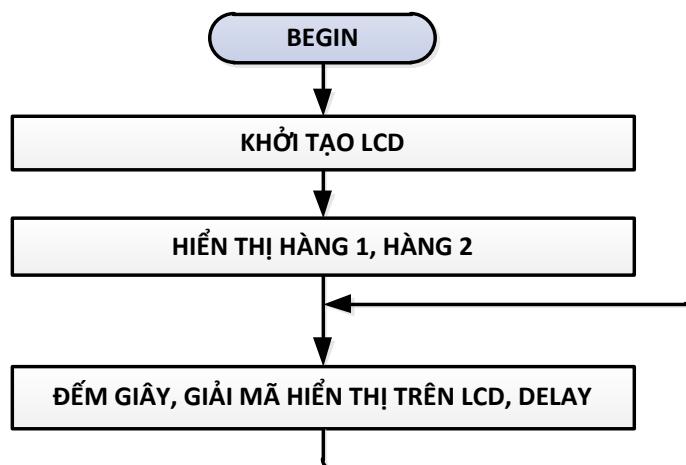
BT MẪU LCD1. KHAI
BÁO THƯ VIỆN<TV_LCD.C>2.
KHỞI TẠO LCD
SET_TRIS...
LCD_SETUP();3. 2
HÀM GM_LCD,
HT_LCD

Bài 5-15: Sử dụng mạch giao tiếp vi điều khiển PIC16F887 với LCD 16x2 ở hình 5-65, hãy viết chương trình hiển thị 2 hàng thông tin như trong bảng 5-7. Trong đó 2 ô cuối của hàng 1 sẽ hiển thị thời gian đếm giây chưa cần chính xác.

Bảng 5-6. Thông tin hiển thị 2 hàng kí tự.

Địa chỉ	80	81	82	83	84	85	86	87	88	89	8A	8B	8C	8D	8E	8F
Kí tự hàng 1	D	O	N	G		H	O	:						S	S	
Kí tự hàng 2	*	T	P		H	O		C	H	I		M	I	N	H	*
Địa chỉ	C0	C1	C2	C3	C4	C5	C6	C7	C8	C9	CA	CB	CC	CD	CE	CF

- Sơ đồ mạch: giống như hình 5-65.
- Lưu đồ:



Hình 5-78. Lưu đồ hiển thị thông tin và đếm giây.

- Chương trình:

```

#include<TV_16F887.C>
#include<TV_LCD.C>
UNSIGNED CHAR I, GIAY;
const unsigned char HANG1[16]={"DONG HO: "};
const unsigned char HANG2[16]={"*TP HO CHI MINH*"};
void LCD_HIENTHI()
{
    LCD_COMMAND(0x8E);      DELAY_US(10);
    LCD_DATA((GIAY/10)+0X30);
    LCD_DATA((GIAY%10)+0X30);
}

void MAIN()
{
    SET_TRIS_E(0x00);      SET_TRIS_D(0x00);
    LCD_SETUP();
    LCD_COMMAND(ADDR_LINE1);      DELAY_US(10);
    FOR(I=0; I<16; I++) { LCD_DATA(HANG1[I]); }
    LCD_COMMAND(ADDR_LINE2);      DELAY_US(10);
}
  
```

```

FOR (I=0; I<16; I++)
{
    LCD_DATA(HANG2[I]);
}

while(TRUE)
{
    FOR (GIAY=0; GIAY<60; GIAY++)
    {
        LCD_HIENTHI();
        DELAY_MS(1000);
    }
}
}

```

Các thông tin hiển thị trên LCD là mã ASCII, giá trị của giây được tách ra thành số BCD và cộng thêm với 0x30 sẽ thành mã ASCII, khi đó mới hiển thị trên LCD.

Bài tập 5-13: Hãy viết chương trình đếm phút giây hiển thị trên LCD. Phút hiển thị cùng hàng với giay, cách 1 ô.

Bài tập 5-14: Hãy viết chương trình đếm giờ phút giây hiển thị trên LCD.

5.11 CÂU HỎI ÔN TẬP – TRẮC NGHIỆM - BÀI TẬP

5.11.1 CÂU HỎI ÔN TẬP

Câu số 5-1: Hãy nêu các chức năng portA, B, C, D và E của vi điều khiển PIC16F887.

Câu số 5-2: Hãy so sánh port0 của vi điều khiển AT89S52 và portA của PIC16F887.

Câu số 5-3: Hãy so sánh port1 của vi điều khiển AT89S52 và portB của PIC16F887.

5.11.2 CÂU HỎI MỞ RỘNG

Câu số 5-4: Hãy tìm hiểu các port của vi điều khiển PIC18F4550 và so sánh với PIC16F887.

5.11.3 CÂU HỎI TRẮC NGHIỆM

Câu 5-1: Vi điều khiển PIC 16F887 có mấy port:

- | | |
|-------|-------|
| (a) 3 | (b) 4 |
| (c) 5 | (d) 6 |

Câu 5-2: Port nào của PIC 16F887 có điện trở kéo lên bên trong:

- | | |
|-----------|-----------|
| (a) PortA | (b) PortB |
| (c) PortC | (d) PortD |

Câu 5-3: Port nào của PIC 16F887 có ngõ vào ngắn ngoài INT:

- | | |
|-----------|-----------|
| (a) PortA | (b) PortB |
| (c) PortC | (d) PortD |

Câu 5-4: Thanh ghi TRISX của PIC 16F887 có chức năng:

- | | |
|-----------------------------|------------------------------|
| (a) Định hướng cho ADC | (b) Lưu dữ liệu cho các port |
| (c) Định hướng cho các port | (d) Lưu địa chỉ của các port |

Câu 5-5: Khai báo biến “INT x” thì biến x là:

- | | |
|------------|------------|
| (a) 1 bit | (b) 8 bit |
| (c) 16 bit | (d) 32 bit |

Câu 5-6: Khai báo biến “INT16 y” thì biến y là:

- | | |
|------------|------------|
| (a) 1 bit | (b) 8 bit |
| (c) 16 bit | (d) 32 bit |

Câu 5-7: Khai báo biến “unsigned int y” thì biến y là số nguyên dương:

- | | |
|--------------------------------|--------------------------------|
| (a) Có giá trị từ 0 đến 255 | (b) Có giá trị từ 0 đến 25 |
| (c) Có giá trị từ -128 đến 127 | (d) Có giá trị từ -128 đến 255 |

Câu 5-8: Khai báo biến “signed int y” thì biến y là số nguyên dương:

- | | |
|--------------------------------|--------------------------------|
| (a) Có giá trị từ 0 đến 255 | (b) Có giá trị từ 0 đến 256 |
| (c) Có giá trị từ -128 đến 127 | (d) Có giá trị từ -128 đến 255 |

Câu 5-9: Kí hiệu “x = y/z” là lệnh có chức năng:

- | | |
|-------------------------------|---------------------------------|
| (a) X bằng y chia z lấy số dư | (b) X bằng y chia z lấy kết quả |
| (c) X bằng z chia y lấy số dư | (d) X bằng z chia y lấy kết quả |

Câu 5-10: Kí hiệu “x = y%z” là lệnh có chức năng:

- | | |
|-------------------------------|---------------------------------|
| (a) X bằng y chia z lấy số dư | (b) X bằng y chia z lấy kết quả |
| (c) X bằng z chia y lấy số dư | (d) X bằng z chia y lấy kết quả |

Câu 5-11: Khai báo nào là khai báo kí tự:

- | | |
|------------|----------|
| (a) INT | (b) Char |
| (c) FFloat | (d) Long |

Câu 5-12: Lệnh nào định cấu hình cho portB:

- | | |
|-----------------------|---------------------|
| (a) Output_b(value) | (b) Output_X(value) |
| (c) Set_tris_b(value) | (d) Input_X(value) |

Câu 5-13: Lệnh nào xuất giá trị 8 bit ra portB:

- | | |
|-----------------------|------------------------|
| (a) Output_b(value) | (b) Output_high(value) |
| (c) Set_tris_b(value) | (d) Input_b(value) |

Câu 5-14: Lệnh nào làm 1 tín hiệu của port xuống mức thấp:

- | | |
|------------------------|----------------------|
| (a) Output_low(pin) | (b) Output_high(pin) |
| (c) Output_toggle(pin) | (d) Output_pull(pin) |

Câu 5-15: Vì điều khiển PIC thì các port:

- | | |
|-----------------------|---------------------------|
| (a) Chỉ truy xuất bit | (b) Truy xuất bit và byte |
| (c) Truy xuất byte | (d) Truy xuất 16 bit |

Câu 5-16: Trong mạch quét 8 led thì transistor có chức năng:

- | | |
|-------------------------|--------------------|
| (a) Khuếch đại dòng | (b) Khuếch đại áp |
| (c) Khuếch đại dòng, áp | (d) Điều khiển led |

Câu 5-17: Trong mạch quét 8 led thì mỗi thời điểm có:

- | | |
|----------------|----------------|
| (a) 2 led sáng | (b) 1 led sáng |
| (c) 3 led sáng | (d) 8 led sáng |

Câu 5-18: Trong mạch quét 8 led nếu chương trình không quét thì:

- | | |
|--------------------|-------------------|
| (a) 8 led vẫn sáng | (b) 8 led sáng mờ |
|--------------------|-------------------|

- (c) 8 led tăt (d) 1 led ság

Câu 5-19: Nếu mở rộng mạch quét 8 led thành 16 led thì dùng tổng cộng:

Câu 5-20: Trong mạch quét 8 led thì thời gian led sáng mỗi led là:

Câu 5-21: Trong mạch quét 8 led thì thời gian led tắt mỗi led là:

Câu 5-22: Mã 7 đoạn số 9 của led anode chung là:

Câu 5-23: Mã 7 đoạn số hex F của led anode chung là:

5.11.4 BÀI TẬP

VỊ ĐIỀU KHIỂN PIC16F887: TIMER - COUNTER

6.1 GIỚI THIỆU

Ở chương này khảo sát các timer/counter của các vi điều khiển, các timer/counter có chức năng đếm xung nội có chu kỳ đã biết để định thời điều khiển thiết bị theo thời gian, có thể đếm xung ngoại để đếm sự kiện như đếm số vòng dây quấn, đếm sản phẩm, đếm tiền, ...

Timer có nhiều tiện ích trong điều khiển nên hầu hết các vi điều khiển đều tích hợp, ở chương này chúng ta sẽ khảo sát timer/counter và các ứng dụng.

Vi điều khiển PIC họ 16F887 có 3 timer T0, T1 và T2. T0 là timer/counter 8 bit, T1 là timer/counter 16 bit, cả 2 đều có bộ chia trước. T2 là timer 8 bit có bộ chia trước và chia sau phục vụ cho các ứng dụng đặc biệt.

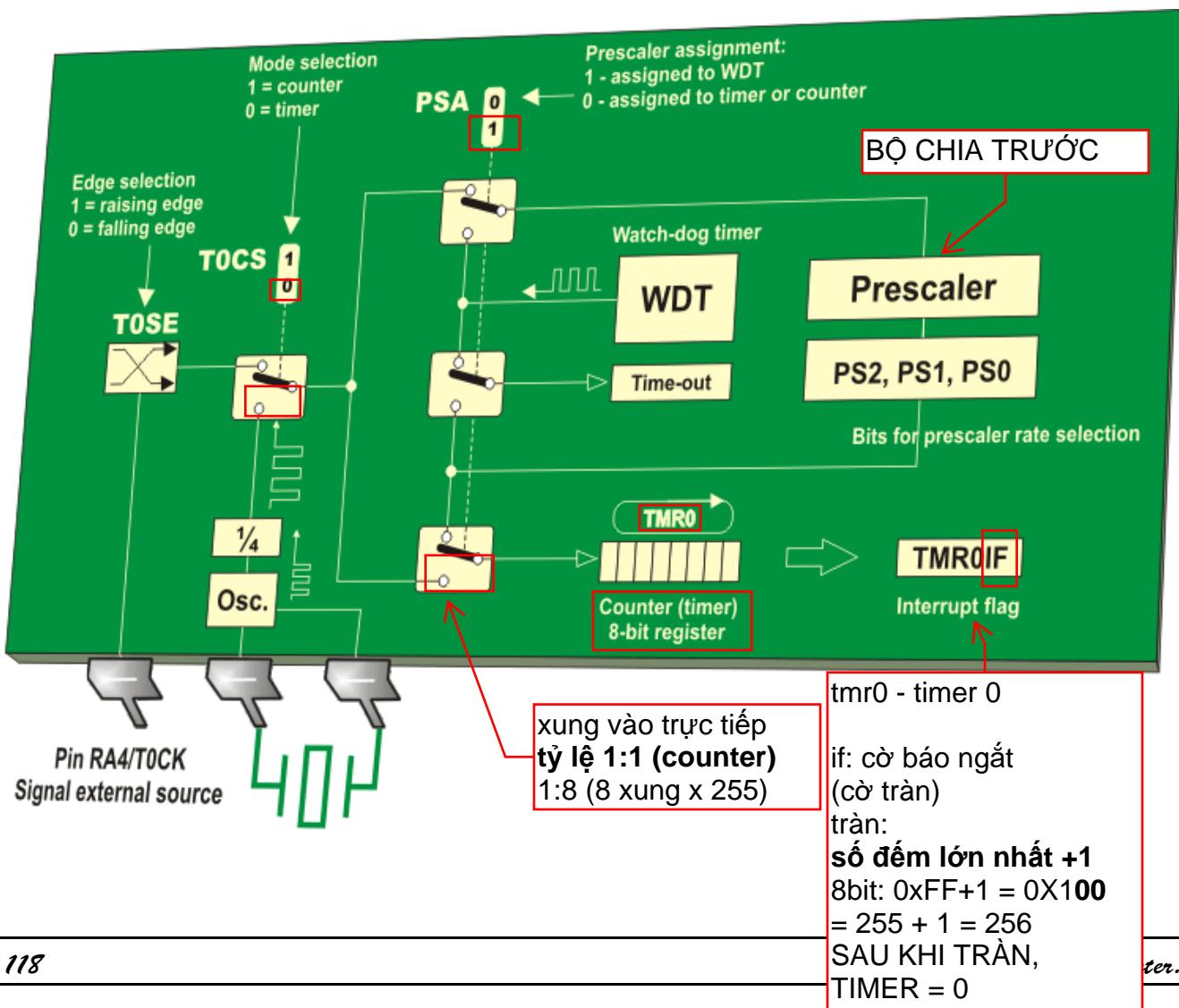
Sau khi kết thúc chương này bạn có thể sử dụng được timer/counter của các vi điều khiển.

6.2 KHẢO SÁT TIMER0

Bộ timer0/counter0 có những đặc điểm sau:

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

Sơ đồ khái niệm của timer0 và bộ chia trước với WDT như hình 6-1:



Hình 6-1. Sơ đồ khói của timer0 của PIC16F887.

Để sử dụng timer0 thì phải khảo sát chức năng của thanh ghi điều khiển timer là OPTION_REG.

Cấu hình thanh ghi và chức năng các bit như sau:

	R/W(1)							
OPTION_REG	RBPU	INTEDG	T0CS	T0SE	PSA	PS2	PS1	PS0

Bit 7

R/W(1): là cho phép đọc/ghi và khi reset thì bằng 1

Bit 0

Bit 7

RBPU Counter --> OPTION_REG = 0x38 --> xem cu trúc b nh (TR.30)

1= c #BYTE OPTION_REG 0X81 //trang 30

0= k

Bit 6

INTEDG: bit lựa chọn cạnh ngắt – Interrupt edge select bit

1= cho phép ngắt cạnh xuống chân INT.

0= cho phép ngắt cạnh lên chân INT.

Bit 5

T0CS: bit lựa chọn nguồn xung cho TMR0 - TMR0 Clock Source Select bit.

1= sê đếm xung ngoại đưa đến chân T0CKI.

0= sê đếm xung clock nội bên trong.

Bit 4

T0SE: bit lựa chọn cạnh tích cực T0SE - TMR0 Source Edge Select bit.

0= tích cực **cạnh lên** ở chân T0CKI.1= tích cực **cạnh xuống** ở chân T0CKI.

Bit 3

PSA: bit gán bộ chia trước - prescaler assignment.

1= gán bộ chia cho WDT.

0= gán bộ chia Timer0.

Bit 2-0

PS2:PS0: các bit lựa chọn tỉ lệ bộ chia trước - prescaler rate select bits:

Bảng 6-1. Lựa chọn hệ số chia của Timer0.

Bit lựa chọn	Tỉ lệ TMR0	Tỉ lệ WDT
000	1:2	1:1
001	1:4	1:2
010	1:8	1:4
011	1:16	1:8
100	1:32	1:16
101	1:64	1:32
110	1:128	1:64
111	1:256	1:128

Nếu bit T0CS bằng 1 thì chọn chế độ đếm xung ngoài Counter. Trong chế độ đếm xung ngoài thì xung đếm đưa đến chân RA4/T0CKI. Bit T0SE = 0 thì chọn cạnh lên, ngược lại thì chọn cạnh xuống.

Bộ chia trước không thể đọc/ghi có mối quan hệ với Timer0 và Watchdog Timer.

6.2.1 NGẮT CỦA TIMER0

Khi giá trị đếm trong thanh ghi TMR0 tràn từ FFh về 00h thì phát sinh ngắt, cờ báo ngắt TMR0IF lên 1. Ngắt có thể ngăn bằng bit cho phép ngắt TMR0IE.

Trong chương trình con phục vụ ngắt Timer0 phải xóa cờ báo ngắt TMR0IF. Ngắt của TMR0 không thể kích CPU thoát khỏi chế độ ngủ vì bộ định thời sẽ ngừng khi CPU ở chế độ ngủ.

Trong datasheet thì bit cho phép ngắt có tên là T0IE và cờ báo ngắt là T0IF. Hai bit này nằm trong thanh ghi INTCON ở vị trí thứ 5 và thứ 2.

Thanh ghi INTCON nằm trong vùng nhớ RAM có địa chỉ là 0xB0.

R/W(0)	R/W(X)						
GIE	PEIE	TOIE	INTE	RBIE	TOIF	INTF	RBIF

Hình 6-3. Thanh ghi INTCON.

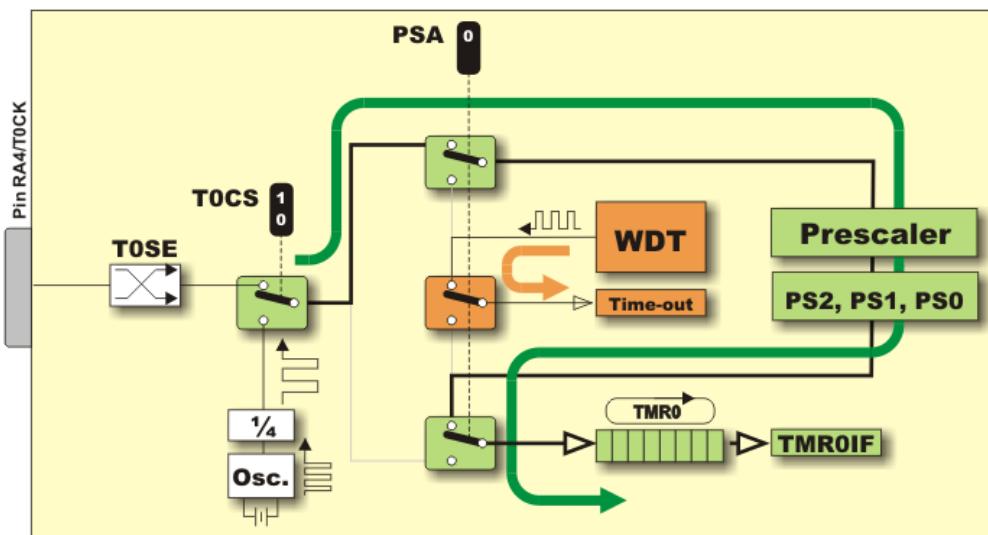
6.2.2 TIMER0 ĐẾM XUNG NGOẠI

Muốn đếm xung ngoại thì xung được đưa đến ngõ vào T0CKI, việc đồng bộ tín hiệu xung ngõ vào T0CKI với xung clock bên trong được thực hiện bằng cách lấy mẫu ngõ ra bộ chia ở những chu kì Q2 và Q4 của xung clock bên trong. Điều này rất cần thiết cho T0CKI ở trạng thái mức cao ít nhất 2 Tosc và ở trạng thái mức thấp ít nhất 2 Tosc.

6.2.3 BỘ CHIA TRƯỚC

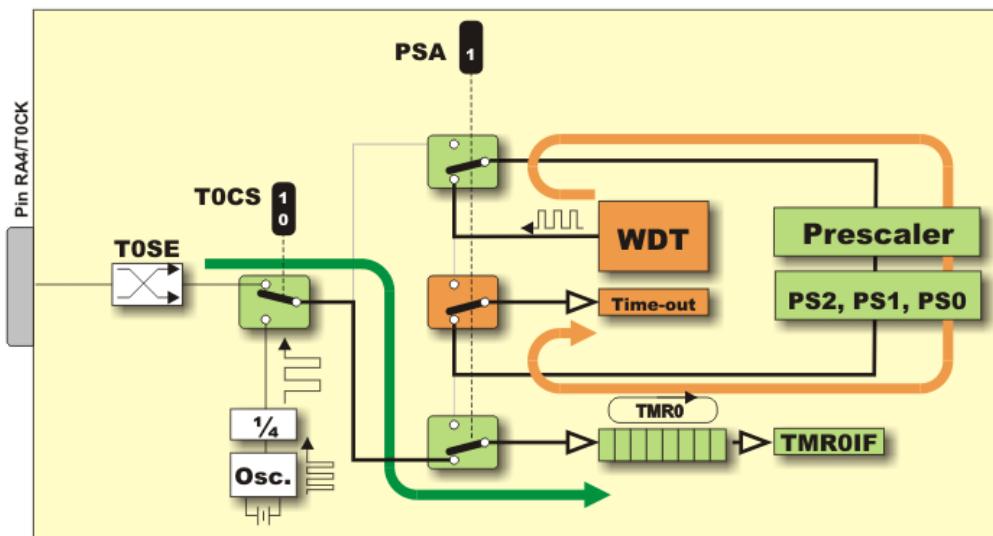
Bộ chia trước có thể gán cho Timer0 hoặc gán cho Watchdog Timer. Các bit PSA và PS2:PS0 chọn đối tượng gán và tỉ lệ chia.

Khi được gán cho Timer0 thì tất cả các lệnh ghi giá trị cho thanh ghi TMR0 sẽ xoá bộ chia trước. Hình 6-4 thì bộ chia trước được gán cho timer T0, xung qua bộ chia rồi mới lưu vào tgi TMR0, khi đó bộ định thời giám sát hoạt động độc lập, khi bộ định thời đếm tràn thì sẽ reset CPU ngay.



Hình 6-4. Bộ chia trước được gán cho timer T0.

Khi được gán cho WDT thì lệnh CLRWDT sẽ xoá bộ chia trước cùng với Watchdog Timer. Hình 6-5 thì bộ chia trước được gán cho bộ định thời giám sát, khi bộ định thời WDT đếm tràn thì qua bộ chia, sau khi đủ hệ số chia thì mới reset CPU. Xung đến bộ định thời T0 thì đi trực tiếp vào thanh ghi TMR0.



Hình 6-5. Bộ chia trước được gán cho WDT.

6.3 KHẢO SÁT TIMER1 CỦA PIC 16F887

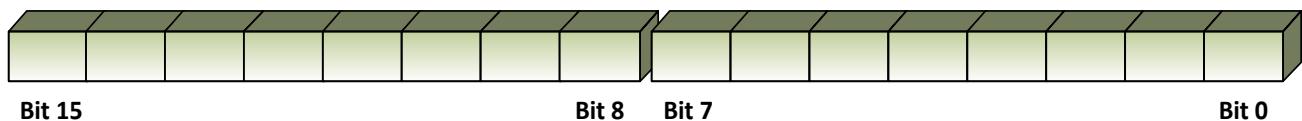
Timer1 có những đặc tính sau:

- Là timer/counter 16 bit.
- Có 2 nguồn xung đếm bên trong và bên ngoài, có thể lập trình được.
- Có bộ chia trước 3 bit.
- Có bộ dao động LP tùy chọn.
- Hoạt động đồng bộ hoặc không đồng bộ.
- Điều khiển timer hoạt động thông qua bộ so sánh hoặc chân T1G.
- Phát sinh ngắt khi tràn.
- Đánh thức CPU khi tràn và sử dụng xung bên ngoài.
- Có chức năng capture/compare.

Timer1 là bộ định thời/đếm 16 bit gồm 2 thanh ghi 8 bit (TMR1H và TMR1L) – có thể đọc và ghi. Hai thanh ghi này tăng từ 0000h đến FFFFh và quay trở lại 0000h.

Khi bị tràn thì Timer1 sẽ phát sinh ngắt, cò báo ngắt TMR1IF (PIR1<0>) lên mức 1. Timer1 có bit cho phép/cấm ngắt là TMR1IE (PIE1<0>).

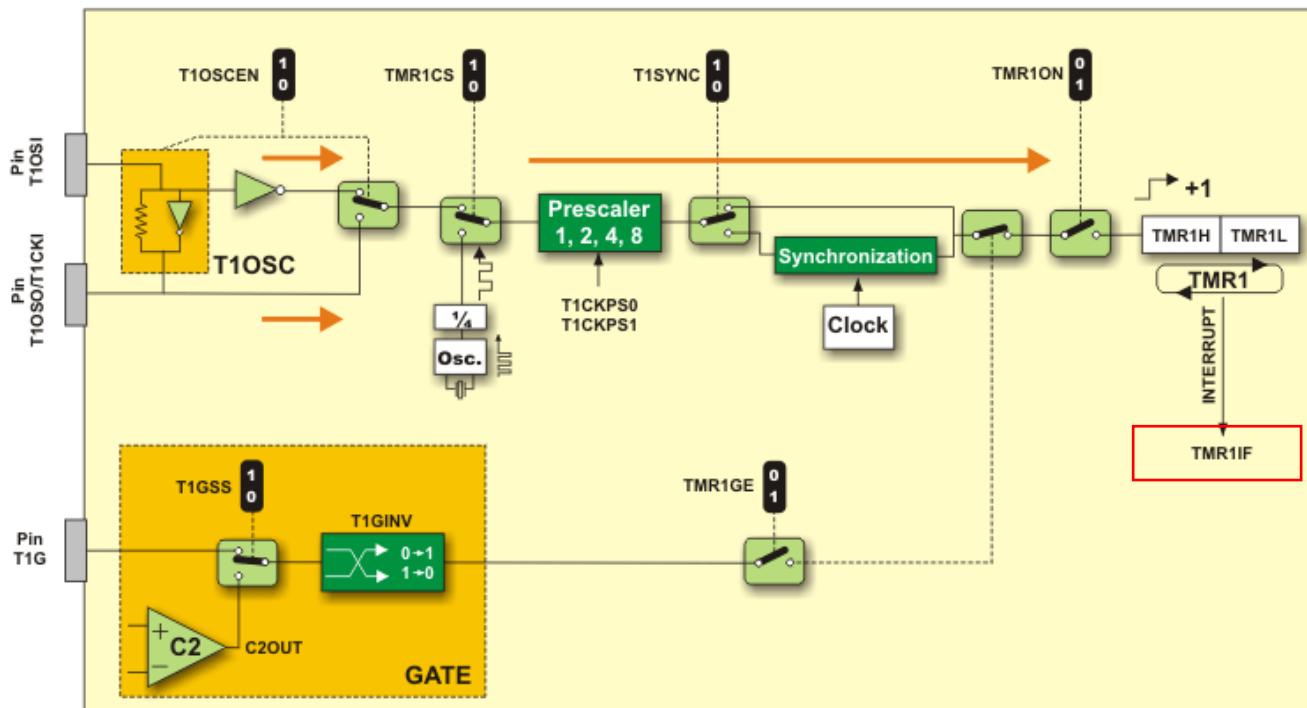
Thanh ghi TMR1H



Thanh ghi TMR1L

Hình 6-6. Thanh ghi lưu kết quả của T1.

Cấu trúc của Timer1:



Hình 6-7. Cấu trúc timer T1.

Khảo sát thanh ghi điều khiển Timer1

	R/W(0)	R/W(0)	R/W(0)	R/W(0)	R/W(0)	R/W(0)	R/W(0)	R/W(0)
T1CON	T1GINV	TMR1GE	T1CKPS1	T1CKPS0	T1OSCEN	T1SYNC	TMR1CS	TMR1ON
Bit 7	R/W(0): là cho phép đọc/ghi và khi reset thì bằng 0					Bit 0		

Hình 6-8. Thanh ghi T1CON.

- Bit 7 **T1GINV:** bit đảo cổng của Timer1 - Timer1 Gate Invert bit
 1= cổng Timer1 tích cực mức 1 (Timer1 đếm khi cổng ở mức 1).
 0= cổng Timer1 tích cực mức 0 (Timer1 đếm khi cổng ở mức 0).
- Bit 6 **TMR1GE:** bit cho phép cổng của Timer1 - Timer1 Gate Enable bit
 Nếu TMR1ON = 0: bit này sẽ không có tác dụng.
 Nếu TMR1ON = 1:
 1= Timer1 mở nếu cổng của Timer1 không tích cực.
 0= Timer1 mở.
- Bit 5-4 **T1CKPS1:T1CKPS0:** các bit lựa chọn bộ chia - Timer1 input Clock Prescale Select bits
 11=1:8 giá trị chia.
 10=1:4 giá trị chia.
 01=1:2 giá trị chia.
 00=1:1 giá trị chia.
- Bit 3 **T1OSCEN:** bit ĐK cho phép bộ dao động Timer1 - Timer1 Oscillator Enable Control bit
 1= bộ dao động được phép.

0= Tắt bộ dao động.

Bit 2 T1SYNC: bit ĐK đồng bộ ngõ vào xung clock bên ngoài của timer1

Khi **TMR1CS = 1:**

1= không đồng bộ ngõ vào với clock.

0= đồng bộ ngõ vào với clock.

Khi **TMR1CS = 0:**

Bit này bị bỏ qua. Timer1 dùng xung clock bên trong khi **TMR1CS = 0.**

Bit 1 TMR1CS: bit lựa chọn nguồn xung clock của timer1

1= Chọn nguồn xung clock từ bên ngoài ở chân RC0/T1OSO/T1CKI (cạnh lên).

0= Chọn xung nội bên trong (Fosc/4).

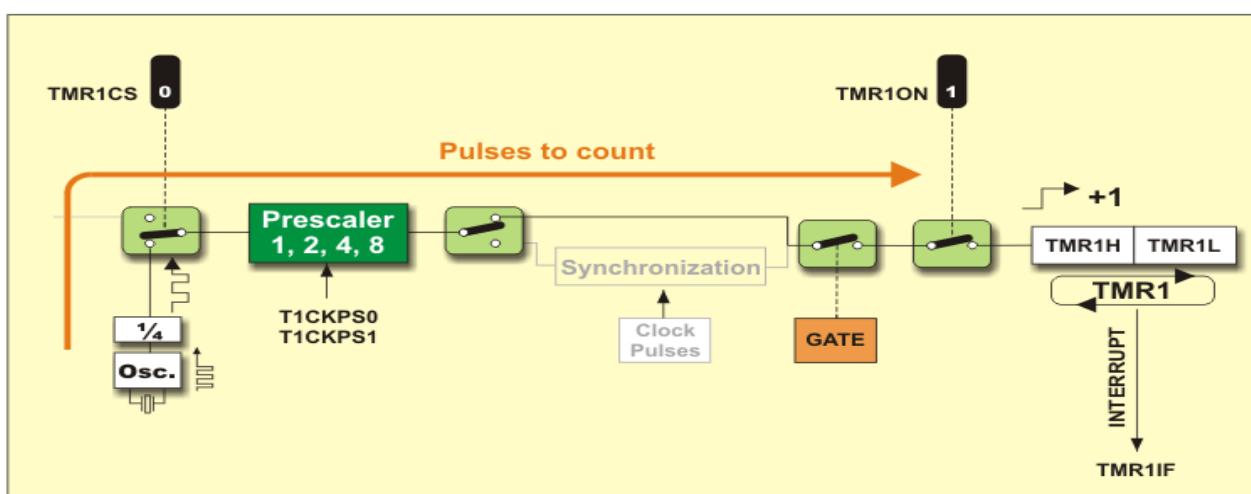
Bit 0 TMR1ON: bit điều khiển Timer1

1= Cho phép Timer1 đếm.

0= Timer1 ngừng đếm.

6.3.1 TIMER1 Ở CHẾ ĐỘ ĐỊNH THỜI

Nếu bit TMR1CS bằng 0 thì T1 hoạt động định thời đếm xung nội có tần số bằng Fosc/4. Bit điều khiển đồng bộ **T1SYNC** không bị ảnh hưởng do xung clock bên trong luôn đồng bộ.

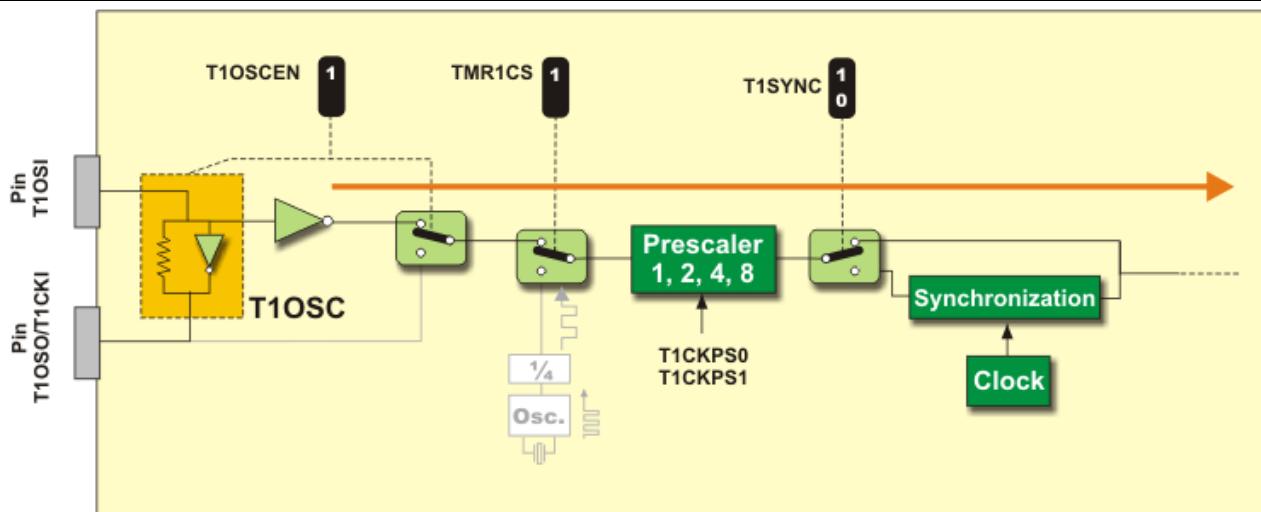


Hình 6-9. Timer1 hoạt động định thời.

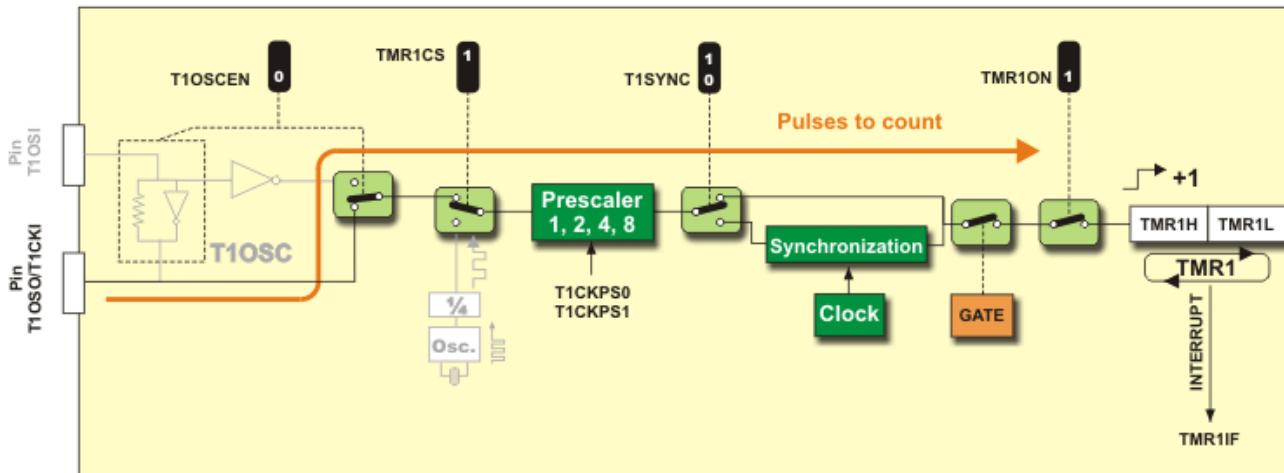
6.3.2 TIMER1 Ở CHẾ ĐỘ ĐÉM XUNG NGOẠI

Nếu bit TMR1CS bằng 1 thì T1 hoạt động đếm xung ngoại. Xung ngoại có 2 nguồn xung phụ thuộc vào bit T1OSCEN:

- Nếu bit T1OSCEN bằng 1 thì T1 đếm xung ngoại từ mạch dao động của T1 - xem hình 6-10.
- Nếu bit T1OSCEN bằng 0 thì T1 đếm xung ngoại đưa đến ngõ vào T1CKI - xem hình 6-11.

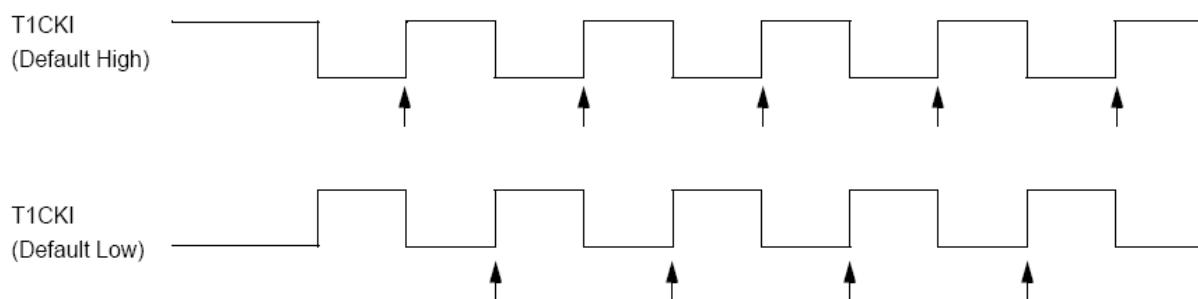


Hình 6-10. T1 hoạt động đếm xung ngoại từ mạch dao động T1.



Hình 6-11. T1 hoạt động đếm xung ngoại đưa đến ngõ vào T1CKI.

Timer1 tăng giá trị khi có xung cạnh lên. Counter chỉ tăng giá trị đếm sau khi nhận 1 xung cạnh xuống được minh họa hình 6-12.



Hình 6-12. Giản đồ thời gian xung đếm của Counter1.

Hình trên thì ngõ vào đang ở mức 1, counter sẽ đếm khi có xung cạnh lên.

Hình dưới thì ngõ vào đang ở mức thấp, xung cạnh lên thứ nhất thì mạch vẫn chưa đếm, xung xuống mức thấp và khi có cạnh lên thứ 2 thì counter bắt đầu đếm.

6.3.3 HOẠT ĐỘNG CỦA TIMER1 Ở CHẾ ĐỘ COUNTER ĐÖNG BỘ

Khi bit TMR1CS bằng 1 thì T1 hoạt động ở chế độ Counter:

- Nếu bit T1OSCEN bằng 1 thì đếm xung từ mạch dao động của T1.

- Nếu bit T1OSCEN bằng 1 thì đếm xung cạnh lên đưa đến ngõ vào RC0/T1OSO/T1CKI.

Nếu bit $\overline{T1SYNC}$ bằng 0 thì ngõ vào xung ngoại được đồng bộ với xung bên trong. Ở chế độ đồng bộ, nếu CPU ở chế độ ngủ thì Timer1 sẽ không đếm vì mạch đồng bộ ngừng hoạt động.

6.3.4 HOẠT ĐỘNG CỦA TIMER1 Ở CHẾ ĐỘ COUNTER BẤT ĐỒNG BỘ

Nếu bit $\overline{T1SYNC}$ bằng 1 thì xung ngõ vào từ bên ngoài không được đồng bộ. Bộ đếm tiếp tục tăng bất đồng bộ với xung bên trong. Bộ đếm vẫn đếm khi CPU ở trong chế độ ngủ và khi tràn sẽ phát sinh ngắt và đánh thức CPU. Ngắt T1 có thể ngăn được.

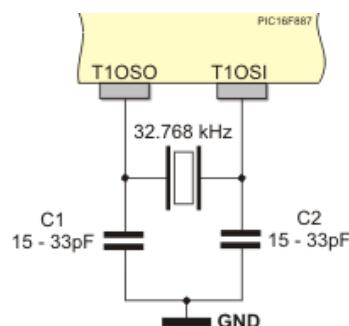
6.3.5 ĐỌC VÀ GHI TIMER1 TRONG CHẾ ĐỘ ĐÉM KHÔNG ĐỒNG BỘ

Timer T1 cho phép đọc giá trị các thanh ghi TMR1H hoặc TMR1L khi timer đang đếm xung bất đồng bộ bên ngoài.

Khi ghi thì nên ngừng timer lại rồi mới ghi giá trị mong muốn vào các thanh ghi. Nếu ghi mà timer đang đếm vẫn được nhưng có thể tạo ra một giá trị đếm không dự đoán được hay không chính xác.

6.3.6 BỘ DAO ĐỘNG CỦA TIMER1

Mạch dao động được tích hợp bên trong và tụ thạch anh nối giữa 2 chân T1OSI và T1OSO để tạo dao động - xem hình 6-13. Bộ dao động được phép hoạt động khi bit T1OSCEN bằng 1.



Hình 6-13. Kết nối thạch anh tạo dao động.

Bộ dao động là dao động công suất thấp, tốc độ tối đa 200 kHz. Bộ dao động vẫn tiếp tục chạy khi CPU ở chế độ ngủ. Bộ dao động chỉ dùng với tụ thạch anh 32768Hz. Tần số này bằng với 2^{15} và khi chia cho bộ đếm 15 bit thì được tần số đúng bằng 1Hz. Bảng 6-2 trình bày cách lựa chọn tụ cho bộ dao động Timer1.

Bảng 6-2. Lựa chọn tần số và tụ tương ứng của Timer1.

Loại dao động	Tần số	Tụ C1	Tụ C2
LP	32kHz	33pF	33pF
	100kHz	15pF	15pF
	200kHz	15pF	15pF

6.3.7 RESET TIMER1 SỬ DỤNG NGÕ RA CCP TRIGGER

Nếu khởi CCP1 và CCP2 được định cấu hình ở chế độ so sánh để tạo ra “xung kích” (CCP1M3:CCP1M0 = 1011), tín hiệu này sẽ reset Timer1.

Chú ý: “xung kích” từ khói CCP1 và CCP2 sẽ không làm cờ ngắt TMR1IF ($\text{PIR1}<0>$) bằng 1.

Timer1 phải định cấu hình ở chế độ định thời hoặc bộ đếm đồng bộ để tạo tiện ích cho cấu trúc này. Nếu Timer1 đang hoạt động ở chế độ đếm bát đồng bộ thì hoạt động Reset không thể thực hiện được.

6.3.8 RESET CẤP THANH GHI TMR1H, TMR1L CỦA TIMER1

Hoạt động reset lúc cấp nguồn POR (Power On Reset) hoặc bất kì reset nào khác không ảnh hưởng đến hai thanh ghi TMR1H và TMR1L, ngoại trừ khi xảy ra “xung kích” của CCP1 và CCP2 thì xóa hai thanh ghi về 00H.

Thanh ghi T1CON được reset về 00h

Reset lúc cấp nguồn POR hoặc khi Brown-out Reset sẽ xóa thanh ghi T1CON và timer T1 ở trạng thái ngừng (OFF) và hệ số chia trước là 1:1. Các reset khác thì thanh ghi không bị ảnh hưởng.

6.4 KHẢO SÁT TIMER2 CỦA PIC16F887

Timer2 là timer 8 bit có bộ chia trước (prescaler) và có bộ chia sau (postscaler).

Timer2 được sử dụng điều khiển xung khi khói CCP hoạt động ở chế độ PWM (pulse width modulation) hoặc chế độ truyền dữ liệu đồng bộ.

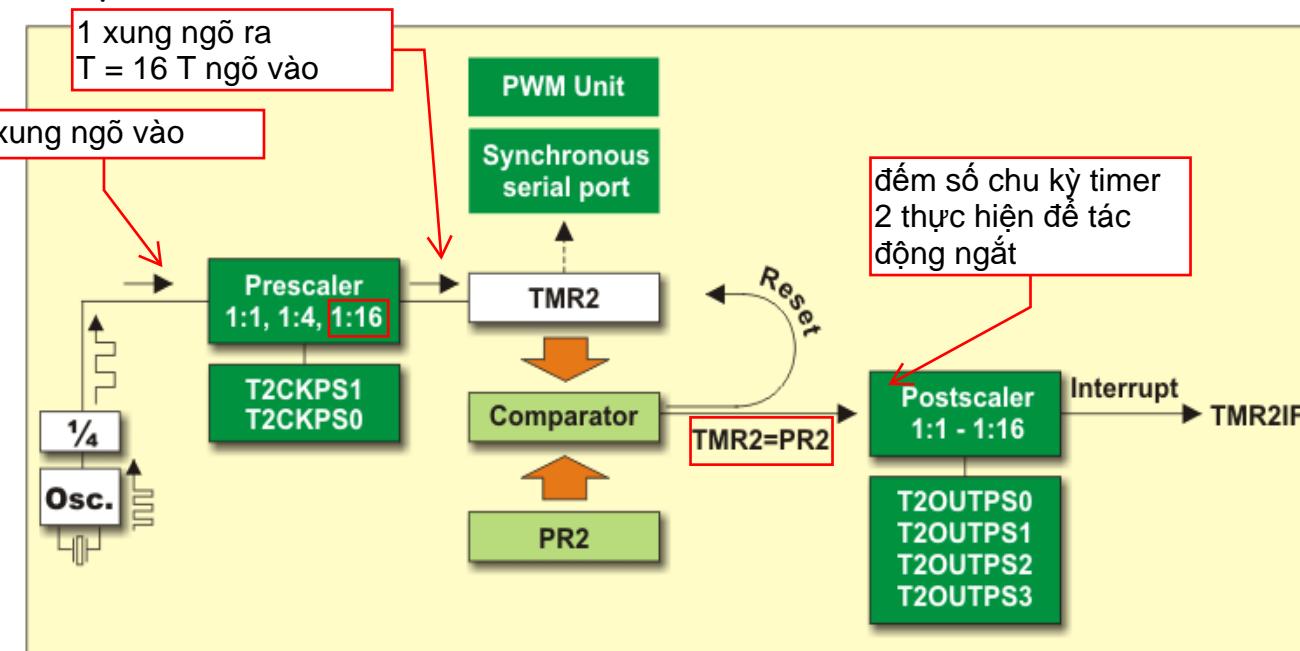
Thanh ghi TMR2 có thể đọc/ghi và xoá khi bị reset.

Xung nội (FOSC/4) qua bộ chia trước có hệ số chia: “1:1”, “1:4” hoặc “1:16” được lựa chọn bằng các bit điều khiển T2CKPS1:T2CKPS0 ($\text{T2CON}<1:0>$).

PR2 là thanh ghi chu kỳ 8 bit dùng để so sánh. Timer2 tăng giá trị từ 00h cho đến khi bằng giá trị lưu trong thanh ghi PR2 thì reset về 00h rồi lặp lại.

PR2 là thanh ghi có thể đọc/ghi, khi hệ thống bị reset thì thanh ghi PR được khởi tạo giá trị FFH.

Ngõ ra của TMR2 đi qua bộ chia sau (postscaler) 4 bit trước khi phát sinh yêu cầu ngắt làm cờ TMR2IF ($\text{PIR1}<1>$) lên 1. Khi bit TMR2ON bằng 0 thì tắt Timer2 để giảm công suất tiêu thụ.



Hình 6-14. Sơ đồ khối của Timer2.

Thanh ghi điều khiển timer2:

	U(0)	R/W(0)	R/W(0)	R/W(0)	R/W(0)	R/W(0)	R/W(0)	R/W(0)
T2CON	-	TOUTPS3	TOUTPS2	TOUTPS1	TOUTPS0	TMR2ON	T2CKPS1	T2CKPS0

Bit 7 U(0): chưa dùng

Bit 0

Hình 6-15. Thanh ghi T2CON.

Bit 7 Chưa sử dụng nếu đọc sẽ có giá trị ‘0’.

Bit 6-3 **TOUTPS3:TOUTPS0:** các bit lựa chọn ngõ ra bộ chia sau (Postscaler) của Timer2

0000=1:1	0001=1:2	0010=1:3	0011=1:4
0100=1:5	0101=1:6	0110=1:7	0111=1:8
1000=1:9	1001=1:10	1010=1:11	1011=1:12
1100=1:13	1101=1:14	1110=1:15	1111=1:16

Bit 2 **TMR2ON:** Bit điều khiển cho phép/cấm Timer2

1= cho phép timer2 đếm.

0= Timer2 ngừng đếm.

Bit 1-0 **T2CKPS1:T2CKPS0:** bit lựa chọn hệ số chia trước cho nguồn xung clock của timer2

00= hệ số chia là 1.

01= hệ số chia là 4.

1x= hệ số chia là 16.

6.4.1 BỘ CHIA TRƯỚC VÀ CHIA SAU CỦA TIMER2

Bộ đếm chia trước và bộ chia sau sẽ bị xóa khi xảy ra một trong các sự kiện sau:

- Thực hiện ghi dữ liệu vào thanh ghi TMR2.
- Thực hiện ghi dữ liệu vào thanh ghi T2CON.
- Bất kỳ Reset nào tác động.

TMR2 không bị xóa khi ghi dữ liệu vào thanh ghi T2CON.

6.4.2 NGÕ RA CỦA TIMER2

Ngõ ra của TIMER2 được nối tới khối SSP – khối này có thể tùy chọn để tạo ra xung nhịp.

6.5 CÁC LỆNH CỦA TIMER – COUNTER TRONG NGÔN NGỮ PIC-C

Các lệnh của ngôn ngữ lập trình C liên quan đến timer/counter bao gồm:

- Lệnh `SETUP_TIMER_X()`
- Lệnh `SET_TIMER_X()`
- Lệnh `SETUP_COUNTERS()`
- Lệnh `SETUP_WDT()`
- Lệnh `RESTART_WDT()`
- Lệnh `GET_TIMER_X()`

6.5.1 LỆNH `SETUP_TIMER_0(MODE)`

Cú pháp: `setup_timer_0(mode)`

Thông số: `mode` có thể là 1 hoặc 2 hằng số định nghĩa trong file `device.h`. Các thông số gồm `RTCC_INTERNAL`, `RTCC_EXT_L_TO_H` hoặc `RTCC_EXT_H_TO_L`, `RTCC_DIV_2`, `RTCC_DIV_4`, `RTCC_DIV_8`, `RTCC_DIV_16`, `RTCC_DIV_32`, `RTCC_DIV_64`, `RTCC_DIV_128`, `RTCC_DIV_256`.

Các hằng số từ nhiều nhóm khác nhau thì có thể or với nhau.

Chức năng: Định cấu hình cho TIMER0.

Có hiệu lực: Cho tất cả các vi điều khiển PIC.

6.5.2 LỆNH `SETUP_TIMER_1(MODE)`

Cú pháp: `setup_timer_1(mode)`

Thông số: `mode` có thể là 1 hoặc 2 hằng số định nghĩa trong file `device.h`. Các thông số gồm `T1_DISABLED`, `T1_INTERNAL`, `T1_EXTERNAL`, `T1_EXTERNAL_SYNC`, `TC_CLK_OUT`, `T1_DIV_BY_1`, `T1_DIV_BY_2`, `T1_DIV_BY_4`, `T1_DIV_BY_8`.

Các hằng số từ nhiều nhóm khác nhau thì có thể or với nhau.

Chức năng: Khởi tạo cho TIMER1.

Với tần số thạch anh là 20MHz và khởi tạo `T1_DIV_BY_8` thì timer sẽ tăng giá trị sau mỗi khoảng thời gian $1.6\mu s$. Timer sẽ tràn sau 104.8576 ms.

Có hiệu lực: Cho tất cả các vi điều khiển PIC có timer 1.

6.5.3 LỆNH `SETUP_TIMER_2(MODE)`

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 timer2 bị reset hay giá này trị dùng để nạp cho thanh ghi PR2.

`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 ngắn – chọn tỉ lệ chia của bộ chia sau.

Chức năng: khởi tạo cho TIMER2.

Mode chỉ định kiểu bộ chia của timer từ tần số của mạch dao động.

Giá trị của timer có thể đọc hoặc ghi dùng lệnh `GET_TIMER2()` và `SET_TIMER2()`.

TIMER2 là timer 8 bit.

Có hiệu lực: cho tất cả các vi điều khiển PIC có timer 2.

6.5.4 LỆNH `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.

Có hiệu lực: cho tất cả các vi điều khiển PIC có timer.

6.5.5 LỆNH `GET_TIMERx()`

Cú pháp: value = get_timerX() ; x là 0, 1, 2

Thông số: **không có.**

Chức năng: đọc giá trị của TIMER/COUNTER.

Có hiệu lực: cho tất cả các vi điều khiển PIC có timer.

6.5.6 LỆNH SETUP_WDT()

Cú pháp: setup_wdt (**mode**)

Thông số: For PCB/PCM parts: WDT_18MS, WDT_36MS, WDT_72MS, WDT_144MS, WDT_288MS, WDT_576MS, WDT_1152MS, WDT_2304MS

For PIC®18 parts: WDT_ON, WDT_OFF

Chức năng: Thiết lập cho bộ định thời watchdog.

Bộ định thời watchdog được sử dụng để reset phần cứng nếu phần mềm lập trình bị sa lầy ở những vòng lặp vô tận hoặc một sự kiện nào đó chờ chúng xảy ra nhưng chúng lại không bao giờ xảy ra, để chấm dứt việc chờ đợi này thì ta phải sử dụng bộ định thời watchdog.

Trước khi tiến hành vào vòng lặp thì ta phải cho phép bộ định thời watchdog đếm với khoảng thời gian lập trình trước và nếu sự kiện đó không xảy ra thì bộ định thời watchdog sẽ hết thời gian lập trình sẽ reset phần cứng thoát khỏi vòng lặp.

	PCB/PCM	PCH
Enable/Disable	#fuses	setup_wdt()
Timeout time	setup_wdt()	#fuses
Restart	restart_wdt()	restart_wdt()

Có hiệu lực: Cho tất cả các thiết bị

Yêu cầu: #fuses, hằng số được định nghĩa trong file devices.h

6.5.7 LỆNH RESTART_WDT()

Cú pháp: restart_wdt()

Thông số: Không có

Chức năng: Khởi tạo lại bộ định thời watchdog. Nếu bộ định thời watchdog được cho phép thì lệnh này sẽ khởi tạo lại giá trị định thời để ngăn chặn không cho bộ định thời reset CPU vì sự kiện chờ đợi đã xảy ra.

	PCB/PCM	PCH
Enable/Disable	#fuses	setup_wdt()
Timeout time	setup_wdt()	fuses
Restart	restart_wdt()	restart_wdt()

Có hiệu lực: Cho tất cả các thiết bị

Yêu cầu: #fuses, hằng số được định nghĩa trong file devices.h

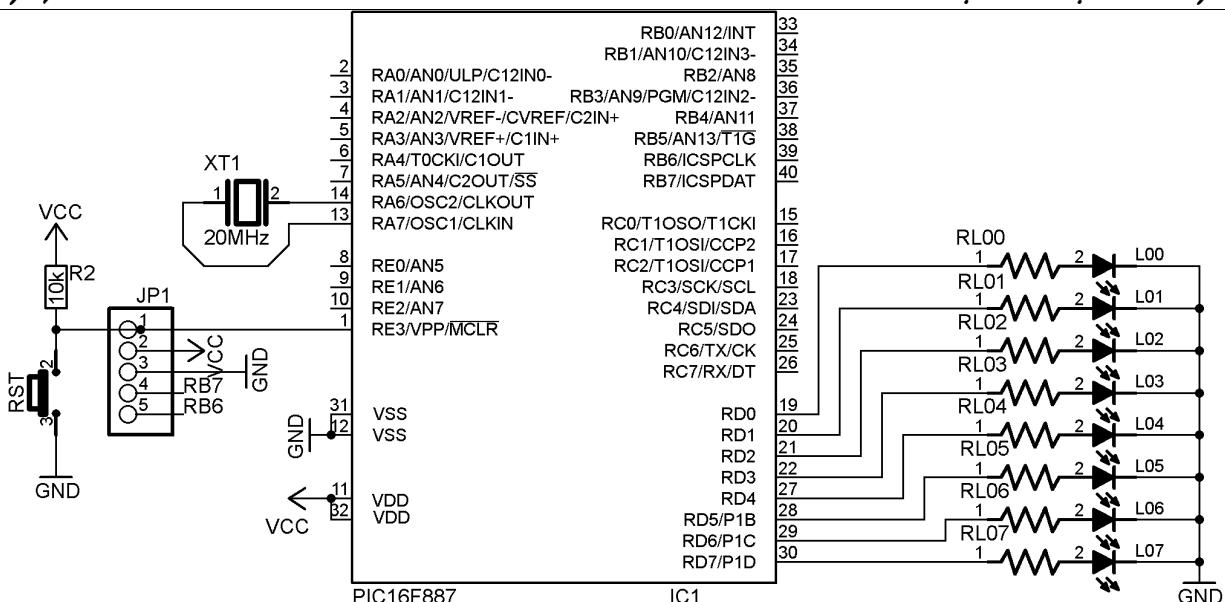
6.6 CÁC ỨNG DỤNG ĐỊNH THỜI DÙNG TIMER

6.6.1 ĐỊNH THỜI DÙNG TIMER T1

Bài 6-1: Dùng vi điều khiển PIC 16F887 điều khiển 8 led đơn sáng tắt dùng timer T1 với chu kỳ delay là 210ms.

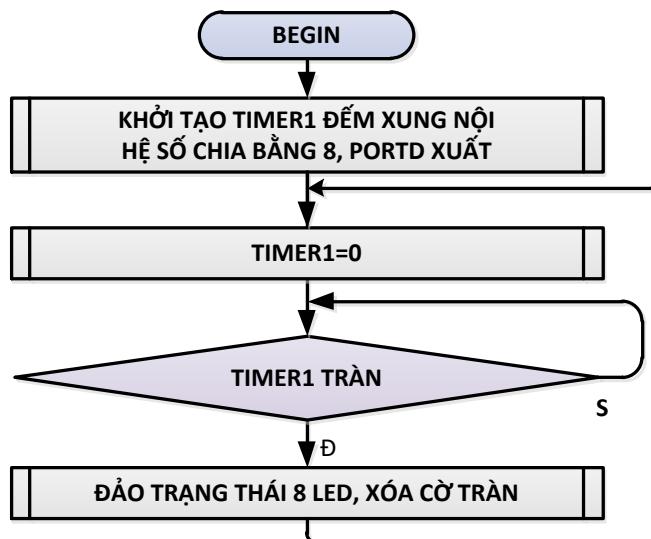
- Sơ đồ mạch: như hình 6-16.

Sử dụng portD để điều khiển 8 led đơn, thạch anh sử dụng là 20Mhz.



Hình 6-16. PIC điều khiển 8 led sáng tắt.

- Lưu đồ:



Hình 6-17. Lưu đồ điều khiển 8 led sáng tắt– định thời 210ms.

- Chương trình:

```
#INCLUDE <TV_16F887.C>
#BIT TMR1IF = 0x0C.0 //bit thu 0 (TMR1IF) cua PIR1
UNSIGNED INT8 x;
VOID MAIN()
{
    SET_TRIS_D(0x00);
    x=0X00;
    OUTPUT_D(x);
    SETUP_TIMER_1(T1_INTERNAL | T1_DIV_BY_8);
    SET_TIMER1(0);
    WHILE (TRUE)
    {
        IF (TMR1IF==1)
        {
            x=~x;      OUTPUT_D(x);      TMR1IF=0;
        }
    }
}
```

}
}

- Giải thích chương trình và tính toán thời gian delay:

Khởi tạo timer1 đếm xung nội có tần số 20MHz qua bộ chia 4 còn 5MHz. Sử dụng bộ chia trước với tỉ lệ chia là 8 nên xung vào bộ đếm với tần số còn lại là $5\text{MHz}/8 = 0,625\text{MHz}$ hay bằng 625kHz, chu kỳ là 1,6μs.

Khởi tạo biến X bằng 00 và gởi biến đếm ra portD sẽ làm các led tắt.

Lệnh if kiểm tra cờ tràn TMR1IF của timer1: nếu chưa tràn thì chờ cho đến khi tràn thì TMR1IF bằng 1, lệnh đảo biến X, giá trị này gởi ra port sẽ làm các led sáng, xóa cờ tràn để báo hiệu tràn cho lần sau.

Nếu tràn lần tiếp theo thì lệnh $X = \sim X$ sẽ nghịch đảo biến X bằng 0x00, giá trị này gởi ra port sẽ làm các led tắt, ...

Timer T1 đếm 65536, thời gian đếm bằng số xung đếm nhân với chu kỳ mỗi xung: $\text{xung} \times \text{Txung} = 65536 \times 1,6\mu\text{s} = 104857,6 \mu\text{s} \approx 105\text{ms}$ nên thời gian sáng và tắt của 8 led đều bằng 105ms hay chu kỳ bằng 210ms.

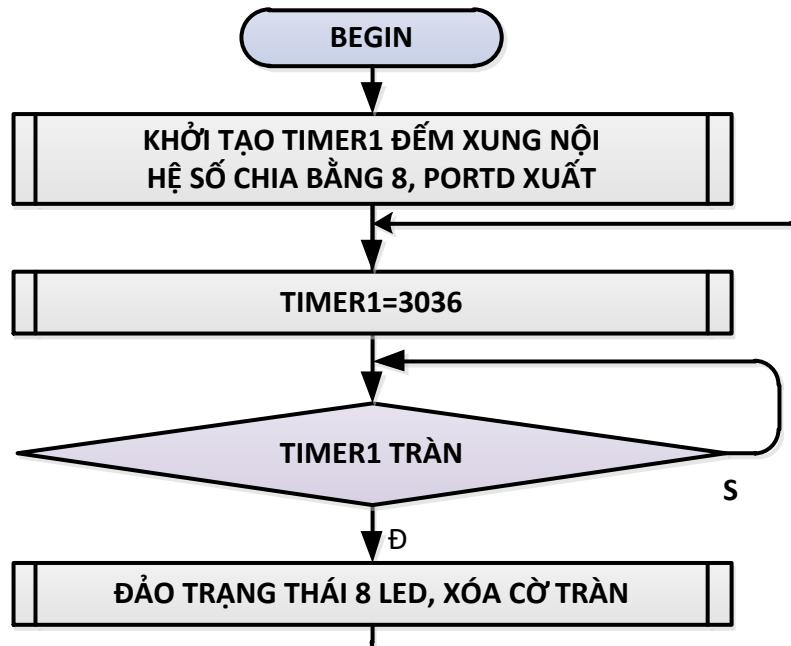
Trong chương trình này sử dụng bit TMR1IF để kiểm tra nhưng thư viện <16F887.H> không định nghĩa bit này, nên có thêm hàng lệnh "#bit tmr1if = 0x0C.0" để định nghĩa bit cờ tràn.

Cờ tràn nằm ở bit thứ 0, trong thanh ghi PIR1 có địa chỉ 0x0C.

Thời gian định thời lớn nhất có thể có của Timer1 là 105ms với tần số thạch anh là 20MHz, bạn chỉ có thể định thời nhỏ hơn nếu chỉ dùng timer, muốn định thời lớn hơn thì phải dùng thêm biến.

Bài 6-2: Dùng vi điều khiển PIC 16F887 điều khiển 8 led đơn sáng tắt dùng timer T1 với chu kỳ delay là 200ms.

- Sơ đồ mạch: giống bài 6-1.
- Lưu đồ:



Hình 6-18. Lưu đồ điều khiển 8 led sáng tắt dùng ngắn – định thời 200ms.

- Chương trình:

```
#INCLUDE<TV_16F887.C>
#BIT      TMR1IF      = 0x0C.0
```

```

UNSIGNED INT8      X;
VOID MAIN()
{
    SET_TRIS_D(0x00);      X=0X00;      OUTPUT_D(X);
    SETUP_TIMER_1(T1_INTERNAL | T1_DIV_BY_8);
    SET_TIMER1(3036);
    WHILE (TRUE)
    {
        IF (TMR1IF==1)
        {
            X=~X;      OUTPUT_D(X);      SET_TIMER1(3036);
            TMR1IF=0;
        }
    }
}

```

- Giải thích chương trình:

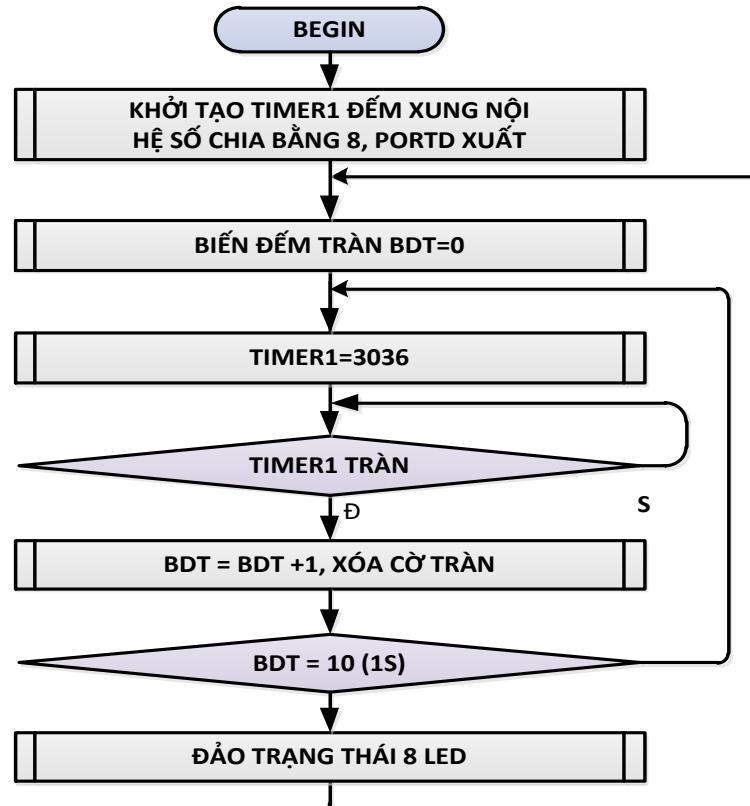
Chương trình này khởi tạo giá trị bắt đầu đếm cho timer T1 là 3036, timer đếm thêm 62500 xung nữa thì tràn, số lượng xung đếm là 62500 xung, mỗi xung là 1,6 μ s nên thời gian sáng bằng thời gian tắt bằng $62500 \times 1,6\mu\text{s} = 1.000.000\mu\text{s} = 100\text{ms}$, chu kỳ bằng 200ms.

Bạn có thể tính như sau: thời gian delay là 100ms hay 100,000 μ s, mỗi chu kỳ xung là 1,6 μ s. Khi đó số xung đếm bằng thời gian chia cho chu kỳ xung sẽ được 62500 xung phù hợp với giá trị 16 bit của Timer1.

Nếu số lượng xung tính lớn hơn số bit của timer thì giảm thời gian đếm.

Bài 6-3: Dùng vi điều khiển PIC 16F887 điều khiển 8 led đơn sáng tắt dùng timer T1 với chu kỳ delay là 2s, 1 giây sáng, 1 giây tắt.

- Sơ đồ mạch: giống bài 6-1.
- Lưu đồ:



Hình 6-19. Lưu đồ điều khiển 8 led sáng tắt – định thời 1s.

- Chương trình:

```

#include <TV_16F887.C>
#BIT TMR1IF = 0x0C.0
UNSIGNED INT8 X, BDT;
VOID MAIN()
{
    SET_TRIS_D(0x00); X=0; BDT=0; OUTPUT_D(X);
    SETUP_TIMER_1(T1_INTERNAL | T1_DIV_BY_8);
    SET_TIMER1(3036);
    WHILE (TRUE)
    {
        IF (TMR1IF==1)
        {
            TMR1IF=0; SET_TIMER1(3036); BDT++;
            IF (BDT ==10)
            {
                OUTPUT_D(X); X=~X; BDT = 0;
            }
        }
    }
}

```

Diagram annotations:

- A red box highlights the code block: `SET_TIMER1(3036); //khởi tạo Timer`
- A red box highlights the code block: `IF (TMR1IF==1)` to `SET_TIMER1(3036); BDT++;`

- Giải thích chương trình:

Chương trình chính kiểm tra cờ tràn, nếu tràn thì khởi tạo lại và tăng biến đếm tràn BDT lên 1, kiểm tra biến đếm tràn BDT bằng 10 hay chưa, nếu chưa bằng thì thoát chờ báo tràn tiếp theo, nếu bằng thì đảo trạng thái của led và xóa biến tràn.

Thời gian sáng bằng thời gian tắt và bằng $100\text{ms} \times 10 = 1000\text{ms} = 1\text{s}$.

Bài tập 6-1: Hãy viết chương trình điều khiển 8 led đơn nối với portD sáng 2 giây, tắt 1 giây, dùng Timer1 định thời.

Bài tập 6-2: Hãy viết chương trình đếm giây chính xác hiển thị trên 2 led 7 đoạn kết nối trực tiếp dùng portB và C, dùng Timer1 định thời.

6.6.2 ĐỊNH THỜI DÙNG TIMER T0

Bài 6-4: Dùng vi điều khiển 16F887 điều khiển 8 led đơn sáng tắt dùng timer T0 với chu kỳ delay là 26ms, 13ms sáng, 13ms tắt.

- Sơ đồ mạch: giống bài 6-1.
- Lưu đồ: như hình 6-20.
- Chương trình:

```

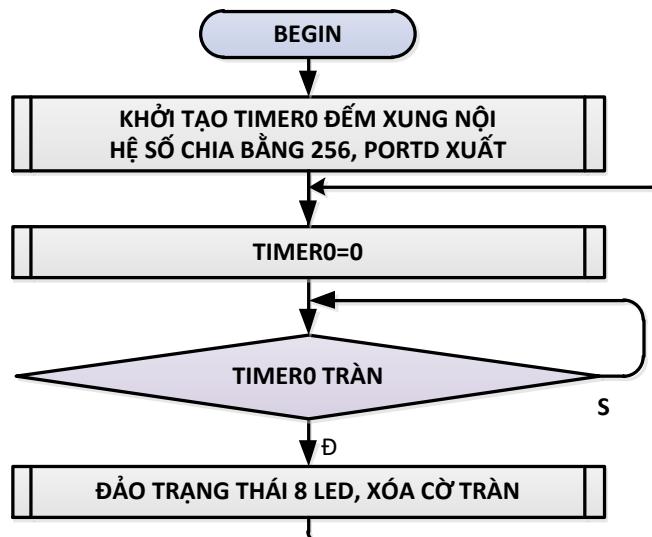
#include <TV_16F887.C>
#BIT TMR0IF = 0x0B.2
UNSIGNED INT8 X;
VOID MAIN()
{
    SET_TRIS_D(0x00);
    X=0; OUTPUT_D(X);
    SETUP_TIMER_0(T0_INTERNAL | T0_DIV_256);
    SET_TIMER0(0);
}

```

```

WHILE (TRUE)
{
    IF (TMR0IF==1)
    {
        TMR0IF=0;      SET_TIMER0 (0);
        X=~X;          OUTPUT_D (X);
    }
}

```



Hình 6-20. Lưu đồ điều khiển 8 led sáng tắt, định thời 13,107ms dùng T0.

- Giải thích chương trình:

Khởi tạo timer0 đếm xung nội có tần số 20MHz qua bộ chia 4 còn 5MHz. Sử dụng bộ chia trước với tỉ lệ chia là 256 nên xung vào bộ đếm với tần số còn lại là $5\text{MHz}/256 = 0,01953125\text{MHz}$ hay bằng $19,53125\text{kHz}$, chu kỳ là $51,2\mu\text{s}$.

Timer T0 đếm 256 xung, thời gian đếm bằng số xung đếm nhân với chu kỳ mỗi xung: $\text{xung} \times \text{Txung} = 256 \times 51,2\mu\text{s} = 13107,2\mu\text{s}$, thời gian sáng và tắt của 8 led đều bằng 13,107 ms hay chu kỳ bằng 26,214ms.

Trong chương trình này sử dụng bit TMR0IF để kiểm tra nhưng thư viện <16F887.H> không định nghĩa bit này, nên có thêm hàng lệnh "#bit tmr0if = 0x0B.2" để định nghĩa bit cờ tràn.

Cờ tràn nằm ở bit thứ 2, trong thanh ghi INTCON có địa chỉ 0x0B.

Thời gian định thời lớn nhất có thể có của Timer0 là 13,107ms với tần số thạch anh là 20MHz, bạn chỉ có thể định thời nhỏ hơn nếu chỉ dùng timer, muốn định thời lớn hơn thì phải dùng thêm biến.

Bài 6-5: Dùng vi điều khiển 16F887 điều khiển 8 led đơn sáng tắt dùng timer T0 với chu kỳ delay là 2s, 1s sáng, 1s tắt.

- Sơ đồ mạch: giống bài 6-1.
- Lưu đồ: như hình 6-21.
- Chương trình:

```

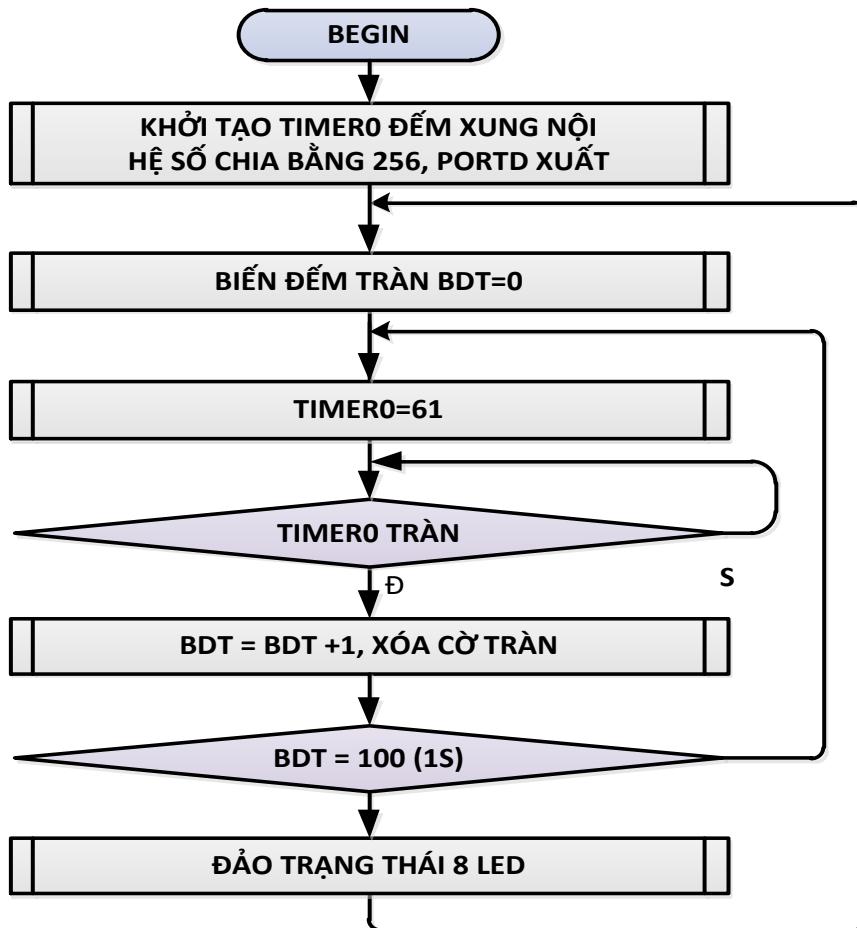
#define<TV_16F887.C>
#BIT TMR0IF      = 0x0B.2
UNSIGNED INT8   X, BDT;

```

```

VOID MAIN()
{
    SET_TRIS_D(0x00);
    X=0;      BDT=0;      OUTPUT_D(X);
    SETUP_TIMER_0(T0_INTERNAL | T0_DIV_256);
    SET_TIMER0(61);
    WHILE (TRUE)
    {
        IF (TMR0IF==1)
        {
            TMR0IF=0;      SET_TIMER0(61);      BDT++;
            if (BDT==100)
            {
                BDT=0;      X=~X;      OUTPUT_D(X);
            }
        }
    }
}

```



Hình 6-21. Lưu đồ điều khiển 8 led sáng tắt, định thời 1s dùng T0.

- Giải thích chương trình:

Timer T0 đếm 195 xung, thời gian đếm bằng số xung đếm nhân với chu kỳ mỗi xung: xung×Txung = 195×51,2μs = 9984μs ≈ 10ms.

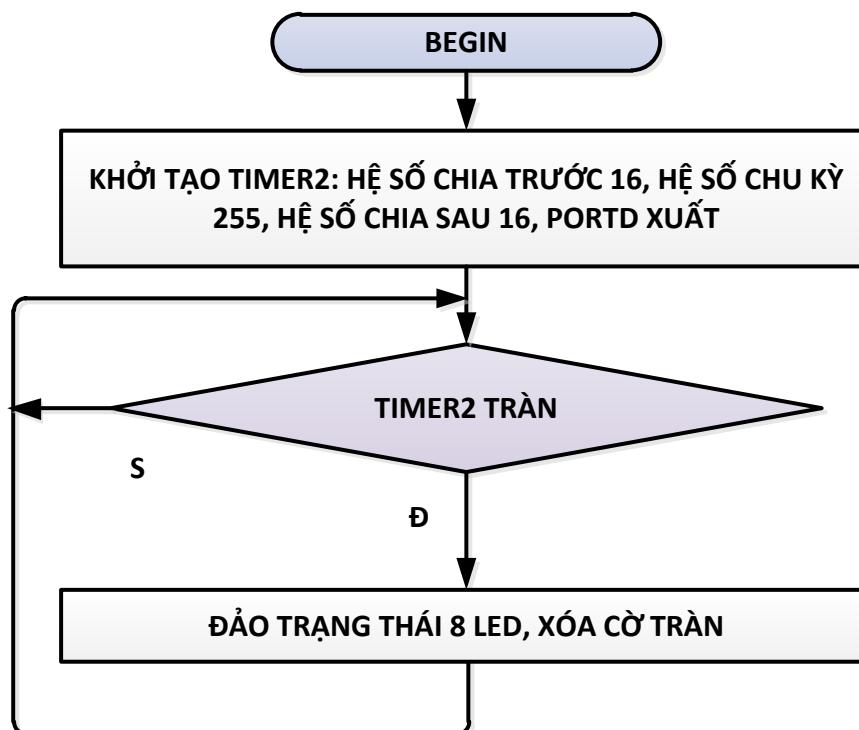
Sau 100 lần ngắt thì tiến hành đảo biến X và gởi ra port nên thời gian sáng và tắt của 8 led đều bằng 1s.

Trong chương trình này có sai số: thiếu 16 μs cho khoảng thời gian 10ms, thiếu 1600 μs cho khoảng thời gian 1s.

6.6.3 ĐỊNH THỜI DÙNG TIMER T2

Bài 6-6: Dùng vi điều khiển 16F887 điều khiển 8 led đơn sáng tắt dùng timer T2 với chu kỳ delay là 26ms, 13ms sáng, 13ms tắt.

- Sơ đồ mạch: giống bài 6-1.
- Lưu đồ:



Hình 6-22. Lưu đồ điều khiển 8 led sáng tắt, định thời 13,107ms dùng T2.

- Chương trình:

```

#include<tv_16f887.c>
#bit TMR2IF = 0x0C.1
unsigned int8 x;
void main()
{
    set_tris_d(0x00);      x=0;      output_d(x);
    setup_timer_2(T2_DIV_BY_16, 255, 16);
    while(true)
    {
        if (TMR2IF==1)
        {
            TMR2IF=0;      x=~x;      output_d(x);
        }
    }
}
  
```

- Giải thích chương trình:

Khởi tạo Timer2 đếm xung nội có tần số 20MHz qua bộ chia 4 còn 5MHz. Sử dụng bộ chia trước với tỉ lệ chia là 16 nên xung vào bộ đếm với tần số còn lại là $5\text{MHz}/16 = 0,3125\text{MHz}$ hay bằng $312,5\text{kHz}$, chu kỳ là $3,2\mu\text{s}$.

Timer T2 đếm 256 xung vì so sánh với giá trị của thanh ghi PR2 bằng 255, thời gian đếm bằng số xung đếm nhân với chu kỳ mỗi xung: $\text{xung} \times \text{Txung} = 256 \times 3,2\mu\text{s} = 819,2\mu\text{s}$.

Bộ chia sau là 16 nên sau 16 lần thì báo ngắn, thời gian sáng và tắt của 8 led đều bằng $819,2\mu s \times 16 ms = 13107,2 \mu s$ hay chu kỳ bằng 26,214ms.

Trong chương trình này sử dụng bit TMR2IF để kiểm tra nhưng thư viện <16F887.H> không định nghĩa bit này, nên có thêm hàng lệnh “#bit tmr2if = 0x0C.1” để định nghĩa bit cờ tràn.

Cờ tràn nằm ở bit thứ 1, trong thanh ghi PIR1 có địa chỉ 0x0C.

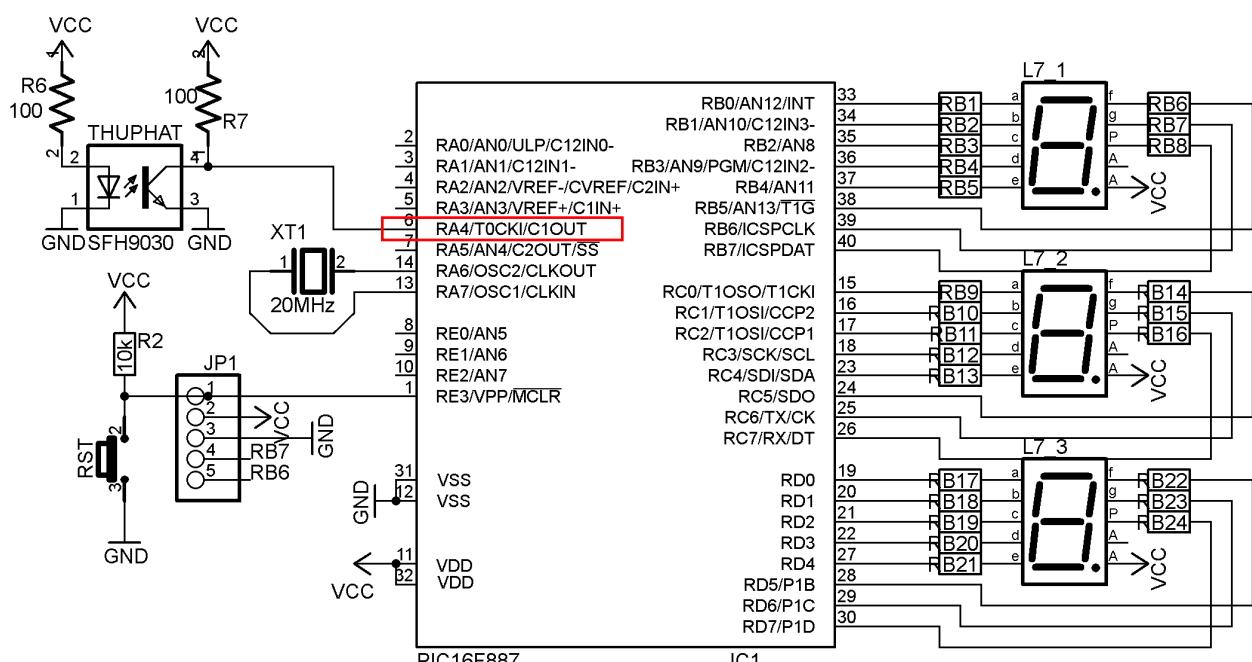
Bài tập 6-3: Hãy viết chương trình đếm giây hiển thị trên 2 led 7 đoạn kết nối trực tiếp dùng portB và C, dùng Timer2 định thời.

6.7 CÁC ÚNG DỤNG ĐÉM XUNG NGOẠI DÙNG COUNTER

6.7.1 ĐÉM XUNG NGOẠI DÙNG COUNTER T0

Bài 6-7: Dùng vi điều khiển 16F887 **đếm xung ngoại dùng T0**, kết quả đếm hiển thị trên 3 led kết nối trực tiếp với 3 port, giới hạn đếm từ 000 đến 255, khi bằng 255 thì quay về 0.

- Sơ đồ mạch:



Hình 6-23. Đếm xung ngoại dùng counter T0.

- Giải thích nguyên lý hoạt động của mạch:

Mạch tạo xung dùng led thu và led phát, khi không có vật che giữa led thu và led phát thì transistor được phân cực dẫn bảo hoà, ngõ ra cực C ở mức 0. Khi có sản phẩm che giữa led phát và transistor thì transistor không được phân cực, tắt nên ngõ ra cực C ở mức 1. Khi sản phẩm đi qua thì tín hiệu về lại mức 0, vậy mỗi sản phẩm đi qua sẽ tạo ra 1 xung.

Xung được đưa đến ngõ vào RA4/T0CKI, 3 port B, C và D để điều khiển 3 led 7 đoạn.

- Lưu đồ: hình 6-23.

Lưu đồ khởi tạo 10 mã 7 đoạn, timer hoạt động đếm xung ngoại, xung đếm tích cực cạnh lên, tỉ lệ chia là 1:1 có nghĩa là không chia, cho giá trị đếm bắt đầu từ 0. Vòng lặp thực hiện công việc đọc giá trị của counter đếm được đếm chuyển thành số BCD, giải mã và gửi ra port để hiển thị ra led.

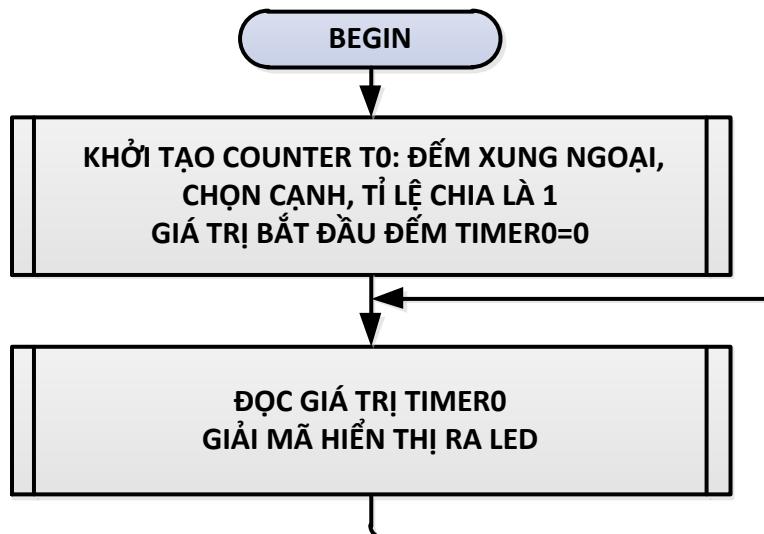
- Chương trình:

```
#INCLUDE<TV_16F887.C>
UNSIGNED INT8 T0;
```

```

VOID MAIN()
{
    SET_TRIS_B(0x00);    SET_TRIS_C(0x00);    SET_TRIS_D(0x00);
    SETUP_TIMER_0(T0_EXT_L_TO_H | T0_DIV_1);
    SET_TIMERO(0);
    WHILE (TRUE)
    {
        T0=GET_TIMER0();
        OUTPUT_B(MA7DOAN[T0%10]);
        OUTPUT_C(MA7DOAN[T0/10%10]);
        OUTPUT_D(MA7DOAN[T0/100]);
    }
}

```



Hình 6-24. Lưu đồ đếm xung ngoại dùng counter T0.

- Giải thích chương trình:

Chương trình khởi tạo 3 port xuất dữ liệu, timer hoạt động chế độ đếm xung ngoại, hệ số chia bằng 1, cho giá trị đếm ban đầu bằng 0.

Vòng lặp while tiến hành đọc giá trị của timer0 gán cho biến “T0”, tiến hành chia để tách từng số đơn vị, chục, trăm, giải mã 7 đoạn và gửi ra port điều khiển led sáng.

Do counter T0 chỉ đếm 8 bit nên giới hạn đếm từ 0 đến 255, muốn đếm giá trị lớn hơn thì phải xử lý hoặc dùng counter T1 đếm 16 bit.

Trong chương trình trên thì mới bắt đầu hiển thị cả 3 số 0, chương trình sau sẽ có chức năng tắt 2 số 0 vô nghĩa của hàng trăm và hàng chục.

```

#define<TV_16F887.C>
UNSIGNED INT8    T0, MATRAM, MACHUC;
VOID MAIN()
{
    SET_TRIS_B(0x00);    SET_TRIS_C(0x00);    SET_TRIS_D(0x00);
    SETUP_TIMER_0(T0_EXT_L_TO_H | T0_DIV_1);
    SET_TIMERO(0);
    WHILE (TRUE)
    {
        T0=GET_TIMER0();
        MATRAM=MA7DOAN[T0/100];
    }
}

```

```

MACHUC=MA7DOAN[T0%10];
IF (MATRAM==0XC0)
{
    MATRAM=0xFF;
    IF (MACHUC==0XC0) MACHUC=0xFF;
}
OUTPUT_B(MA7DOAN[T0%10]);
OUTPUT_C(MACHUC);
OUTPUT_D(MATRAM);
}

```

T0 = GET_TIMER0();
GM();
HT();

UNSIGNED INT16 KQ_T0=0; TAM =
GET_TIMER0(); IF (TAM == 1)
{ SET_TIMER0(0); KQ_T0 ++;
IF(KQ_T0 == 1000) KQ_T0 = 0;}//
COUNTER DEM XUONGUNSIGNED
INT16 KQ_T0=999; TAM = GET_TIMER0()
(); IF (TAM == 1) { SET_TIMER0(0);
KQ_T0 --; IF(KQ_T0 == 0) KQ_T0 =
999;}

- Giải thích chương trình:

Tách hàng trăm, hàng chục rồi tiến hành giải mã bằng 0 hay không, nếu bằng 0 thì tiến hành gán mã FF. Khiến led 7 đoạn thì làm led 7 đoạn tắt. Tiếp theo kiểm tra hàng chục rồi đến hàng trăm?

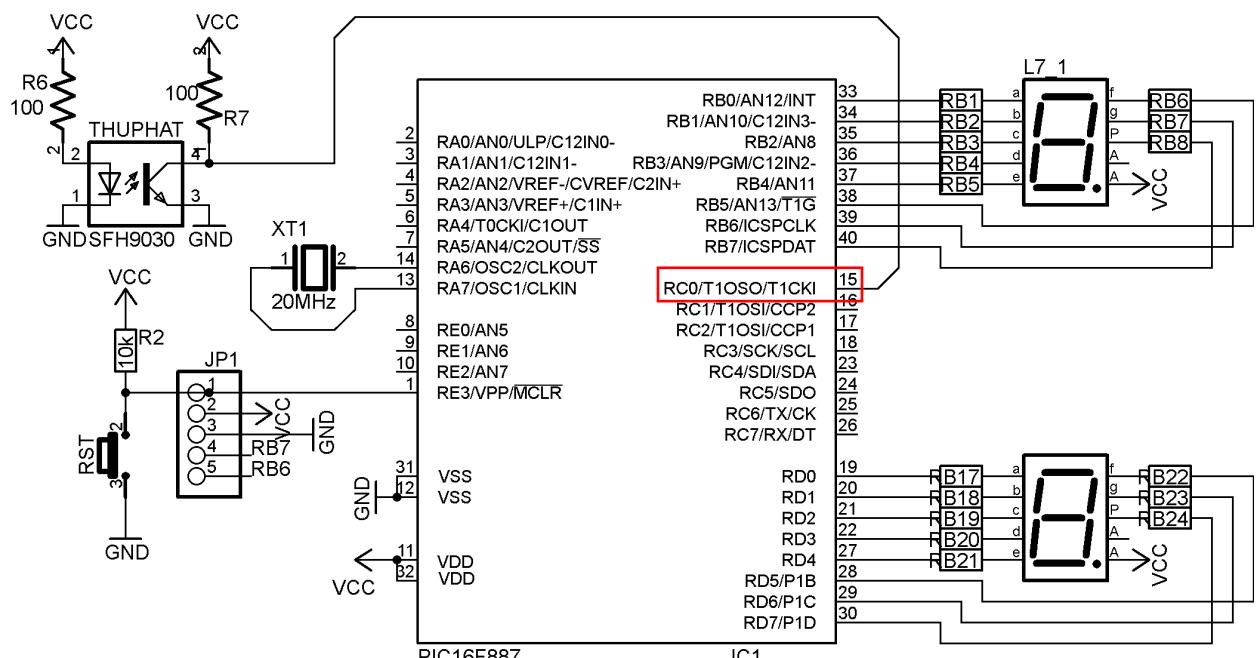
Bài tập 6-4: Hãy viết chương trình đếm xung ngoại dùng T0 hiển thị kết quả đếm từ 0 đến 999 trên 3 led 7 đoạn kết nối trực tiếp.

Bài tập 6-5: Hãy viết chương trình đếm xung ngoại dùng T0 hiển thị kết quả đếm từ 0 đến 999 trên LCD.

6.7.2 ĐẾM XUNG NGOẠI DÙNG COUNTER T1

Bài 6-8: Dùng vi điều khiển 16F887 đếm xung ngoại dùng T1, kết quả đếm hiển thị trên 2 led 7 đoạn kết nối trực tiếp với 2 port, giới hạn đếm từ 00 đến 99, khi bằng 100 thì quay về 0.

- Sơ đồ mạch:



Mạch tạo xung giống như trên. Xung bây giờ đưa đến chân T1CKI/RC0 nên portC không thể dùng để điều khiển led 7 đoạn và chỉ còn 2 port đầy đủ là B và D nên chỉ kết nối 2 led 7 đoạn trực tiếp.

- Lưu đồ: hình 6-26.



Hình 6-26. Lưu đồ đếm xung ngoại dùng counter T1.

Lưu đồ khởi tạo 10 mã 7 đoạn, timer hoạt động đếm xung ngoại, xung đếm tích cực cạnh lên, tỉ lệ chia là 1:1 có nghĩa là không chia, cho giá trị đếm bắt đầu từ 0. Vòng lặp thực hiện công việc đọc giá trị của counter đếm được đếm chuyển thành số BCD, giải mã và gởi ra port để hiển thị ra led.

- Chương trình:

```

#INCLUDE<TV_16F887.C>
UNSIGNED INT16 T1;
VOID MAIN()
{
    SET_TRIS_B(0x00);    SET_TRIS_D(0x00);
    SETUP_TIMER_1(T1_EXTERNAL | T1_DIV_BY_1 );
    SET_TIMER1(0);
    WHILE (TRUE)
    {
        T1=GET_TIMER1();
        OUTPUT_B(MA7DOAN[T1%10]);
        IF (T1/10==0) OUTPUT_D(0xFF);
        ELSE           OUTPUT_D(MA7DOAN[T1/10]);
        IF (T1==100)   SET_TIMER1(0);
    }
}
  
```

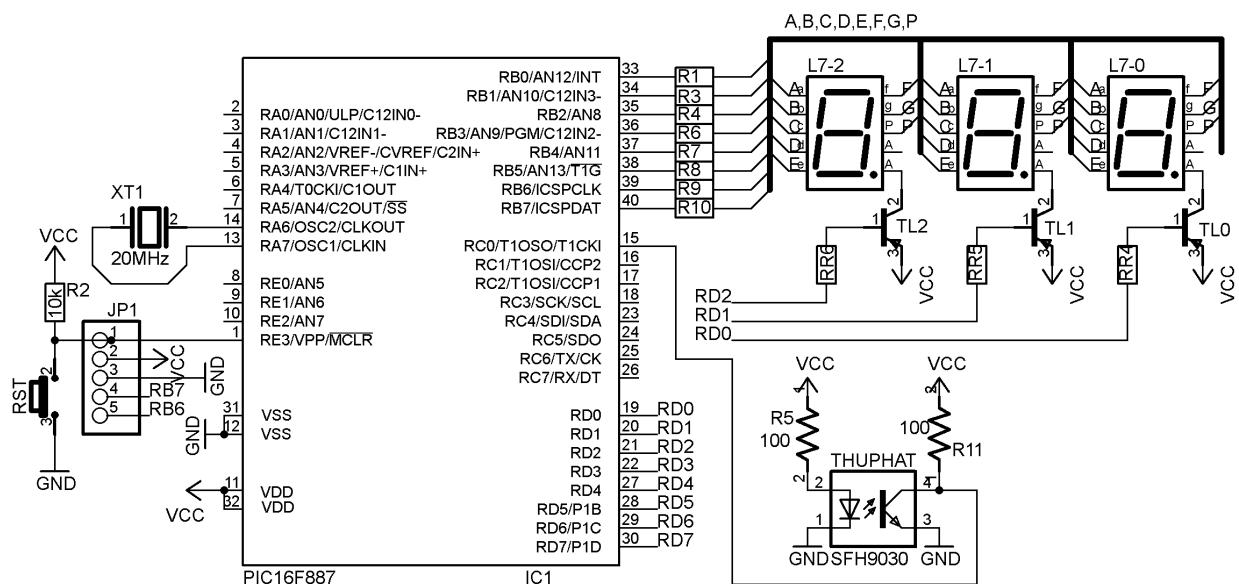
- Giải thích chương trình:

Chương trình khởi tạo 3 port xuất dữ liệu, timer hoạt động chế độ đếm xung ngoại, hệ số chia bằng 1, cho giá trị đếm ban đầu bằng 0.

Vòng lặp while tiến hành đọc giá trị của timer1 gán cho biến “T1”, tiến hành chia để tách từng số đơn vị, chục, giải mã 7 đoạn và gởi ra port điều khiển led sáng.

Bài 6-9: Dùng vi điều khiển 16F887 đếm xung ngoại dùng T1, kết quả đếm hiển thị trên 3 led quét nối với 2 port, giới hạn đếm từ 0 đến 999, khi bằng 1000 thì quay về 0.

- Sơ đồ mạch:



Hình 6-27. Đếm xung ngoại dùng counter T1 hiển thị trên 3 led quét.

- Giải thích nguyên lý hoạt động của mạch:

Mạch tạo xung giống như bài đã trình bày, xung ngõ ra của mạch đưa đến ngõ vào T1CKI ở port RC0, dùng port B và 3 bit của portD để điều khiển 3 led 7 đoạn.

- Lưu đồ: hình 6-28.



Hình 6-28. Lưu đồ chương trình đếm dùng counter T1 của PIC 16F887.

Lưu đồ khởi tạo 10 mã 7 đoạn, timer 1 hoạt động đếm xung ngoại, tỉ lệ chia là 1 có nghĩa là không chia, cho giá trị đếm bắt đầu từ 0. Vòng lặp thực hiện công việc đọc giá trị của counter đếm được đếm chuyển thành số BCD, giải mã và quét led hiển thị.

- Chương trình:

```
#INCLUDE<TV_16F887.C>
UNSIGNED INT16    T1;
UNSIGNED CHAR      MADONVI, MACHUC, MATRAM;

VOID  GIAIMA ()
{
    MATRAM = MA7DOAN[T1 /100];
    MACHUC = MA7DOAN[T1/10 % 10];
```

```

MADONVI= MA7DOAN[T1 % 10];
IF (MATRAM == 0xC0)
{
    MATRAM=0xFF;
    IF (MACHUC == 0xC0 )
    {
        MACHUC=0xFF;
    }
}
VOID HIENTHI()
{
    OUTPUT_B(MADONVI);           OUTPUT_LOW(PIN_D0);
    DELAY_MS(1);                 OUTPUT_HIGH(PIN_D0);

    OUTPUT_B(MACHUC);           OUTPUT_LOW(PIN_D1);
    DELAY_MS(1);                 OUTPUT_HIGH(PIN_D1);

    OUTPUT_B(MATRAM);           OUTPUT_LOW(PIN_D2);
    DELAY_MS(1);                 OUTPUT_HIGH(PIN_D2);
}

VOID MAIN()
{
    SET_TRIS_B(0x00);           SET_TRIS_D(0x00);
    SETUP_TIMER_1(T1_EXTERNAL | T1_DIV_BY_1);
    SET_TIMER1(0);
    WHILE (TRUE)
    {
        T1=GET_TIMER1();
        GIAIMA();
        HIENTHI();
        IF (T1==1000) SET_TIMER1(0);
    }
}

```

- Giải thích chương trình:

Chương trình khởi tạo 2 port xuất dữ liệu, timer 1 hoạt động chế độ đếm xung ngoại, hệ số chia bằng 1, cho giá trị đếm ban đầu bằng 0.

Vòng lặp while tiến hành đọc giá trị của timer1 gán cho biến “T1”, gọi chương trình chia để tách từng số đơn vị, chục, trăm, giải mã 7 đoạn và quét led hiển thị kết quả đếm.

Chương trình này khi thực hiện chương trình con quét hiển thị 3 led thì thời gian delay cho mỗi led là 1ms, 3 led mất 3ms, nếu xung đếm có chu kỳ nhỏ hơn 3ms thì không hiển thị theo đúng giá trị đếm.

Bài tập 6-6: Hãy viết chương trình đếm xung ngoại dùng T1 hiển thị kết quả đếm từ 0 đến 9999 trên LCD.

Bài tập 6-7: Hãy viết chương trình đếm xung ngoại 2 kênh dùng T0 và T1 hiển thị kết quả đếm từ 0 đến 9999 trên LCD. T0 hiển thị ở hàng 1, T1 hiển thị ở hàng 2.

6.8 CÂU HỎI ÔN TẬP – TRẮC NGHIỆM

6.8.1 CÂU HỎI ÔN TẬP

Câu số 6-1: Hãy cho biết tên các thanh ghi liên quan đến timer T0 của vi điều khiển PIC16F887.

Câu số 6-2: Hãy cho biết chức năng các bit trong thanh ghi OPTION REG của PIC16F887.

Câu số 6-3: Hãy cho biết tên các thanh ghi liên quan đến timer T1 của vi điều khiển PIC16F887.

Câu số 6-4: Hãy cho biết chức năng các bit trong thanh ghi T1CON của vi điều khiển PIC16F887.

Câu số 6-5: Hãy cho biết tên các cờ báo ngắt và các bit cho phép ngắt của T0 và T1 của PIC16F887.

Câu số 6-6: Hãy cho biết chức năng các bit trong thanh ghi T2CON của vi điều khiển PIC16F887.

Câu số 6-7: Hãy cho biết tên cờ báo ngắt và bit cho phép ngắt timer T2 của PIC16F887.

6.8.2 CÂU HỎI MỞ RỘNG

Câu số 6-8: Hãy tìm hiểu các timer của vi điều khiển PIC18F4550 và so sánh với PIC16F887.

6.8.3 CÂU HỎI TRẮC NGHIỆM

Câu 6-1: Vì điều khiển PIC 16F887 có mấy timer:

Câu 6-2: Vì điều khiển PIC 16F887 có mấy Counter:

Câu 6-3: Timer T0 của PIC 16F887 là:

Câu 6-4: Timer T1 của PIC 16F887 là:

Câu 6-5: Timer nào của PIC 16F887 có bộ chia trước vào bộ chia sau:

- (a) T0, T1, T2 (b) T2 (c) T0, T1 (d) T0, T2

Câu 6-6: Timer nào của PIC 16F887 chỉ có bộ chia trước:

- (a) T0, T1, T2 (b) T2 (c) T0, T1 (d) T0, T2

Câu 6-7: Timer0 của PIC 16F887 phát sinh yêu cầu ngắt khi:

Câu 6-8: Timer1 của PIC 16F887 phát sinh yêu cầu ngắt khi:

Câu 6-9: Timer2 của PIC 16F887 phát sinh yêu cầu ngắt khi:

Câu 6-10: Timer nào của PIC 16F887 cho phép đếm xung ngoại:

- (a) T0, T1, T2 (b) T2 (c) T0, T1 (d) T0, T2

Câu 6-11: Timer nào của PIC 16F887 có chế độ hoạt động ON /OFF:

- (a) T0, T1, T2 (b) T2, T1 (c) T0, T1 (d) T1

Câu 6-12: Timer0 của PIC 16F887 thì bit nào gán bộ chia trước cho timer0 hay watchdog timer:

Câu 6-13: Timer0 của PIC 16F887 thì bit nào lựa chọn xung nội hay xung ngoại:

6.8.4 BÀI TẬP

VỊ ĐIỀU KHIỂN PIC16F887:

CHUYỂN ĐỔI TƯƠNG TỰ SANG SỐ

7.1 GIỚI THIỆU

Ở chương này khảo sát vi điều khiển giao tiếp với vi mạch chuyển đổi tương tự sang số (ADC) và vi điều khiển có tích hợp ADC để thực hiện các ứng dụng trong đo lường và điều khiển.

Sau khi kết thúc chương này bạn có thể kết nối vi điều khiển không có tích hợp bộ chuyển đổi ADC với các vi mạch ADC, sử dụng được vi điều khiển có tích hợp ADC, biết trình tự thực hiện quá trình chuyển đổi ADC, biết tính toán độ phân giải, chuyển đổi và tính trung bình kết quả.

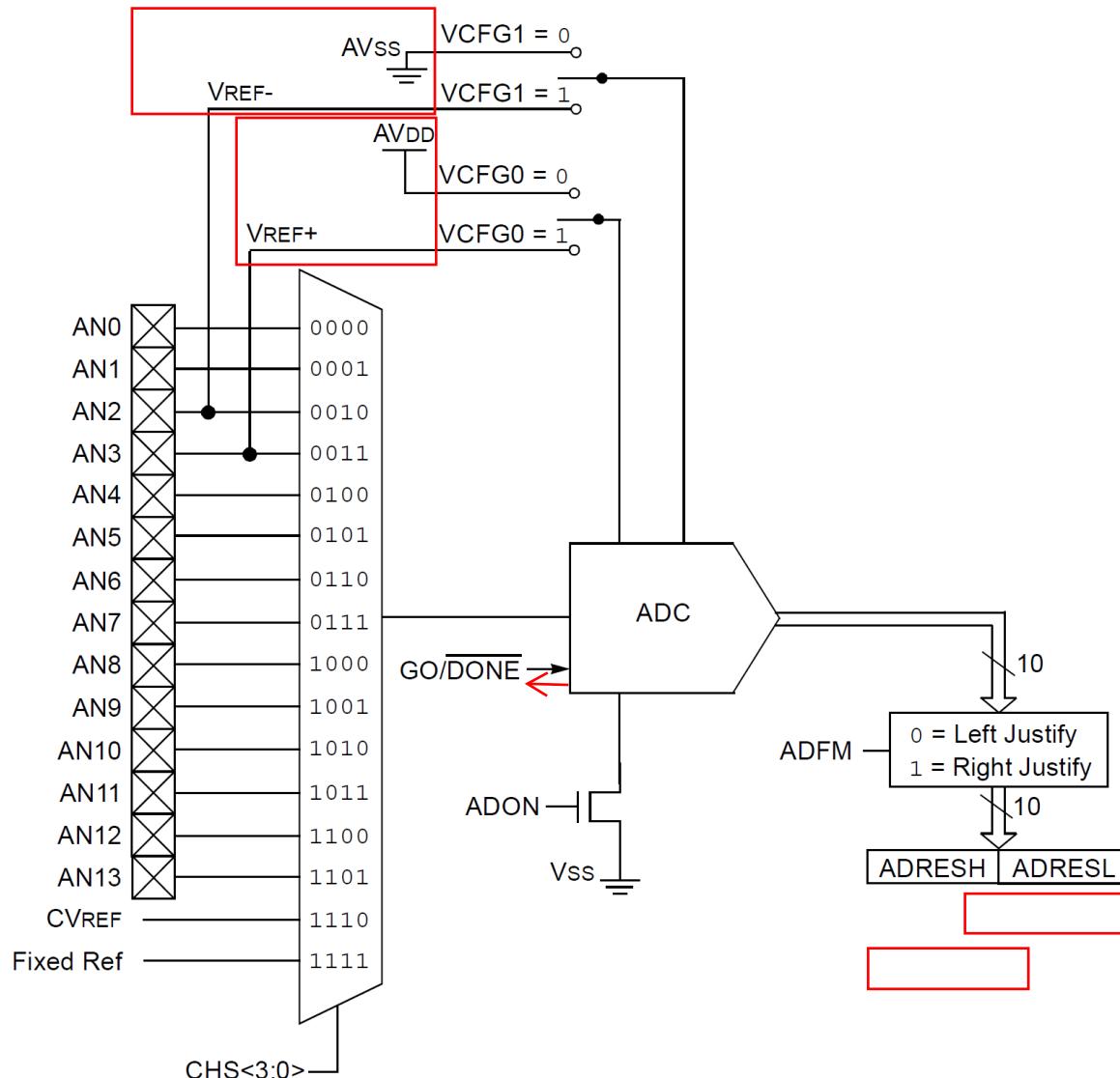
7.2 ADC CỦA VI ĐIỀU KHIỂN PIC 16F887

Vi điều khiển PIC 16F877A có bộ chuyển đổi tín hiệu tương tự sang tín hiệu số ADC 10 bit đa hợp 8 kênh và PIC 16F887 có 14 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.

7.2.1 KHẢO SÁT ADC CỦA PIC 16F887

ADC của PIC16F887 có sơ đồ khói như hình 7-1:



Hình 7-1. Sơ đồ khói của ADC PIC 16F887.

Chức năng các thành phần:

- AN0 đến AN13 (analog) là 14 ngõ vào của 14 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 AV_{DD} hoặc điện áp tham chiếu bên ngoài nối với ngõ vào Vref+ của chân AN3, bit lựa chọn là VCFG0.
- Điện áp tham chiếu âm Vref- có thể lập trình nối với nguồn cung cấp AV_{SS} hoặc điện áp tham chiếu bên ngoài nối với ngõ vào Vref- của chân AN2, bit lựa chọn là VCFG1.
- 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 tắt.
- Kết quả chuyển đổi là số nhị phân 10 bit sẽ lưu vào cặp 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.
- Ngoài các tín hiệu vừa trình bày thì còn có nguồn xung clock cấp cho ADC để thực hiện chuyển đổi, trong hình không có trình bày nhưng nó phải có và nguồn xung được lấy từ dao động RC bên trong của khối ADC.

ADC có 14 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 VDD và VSS hoặc nhận điện áp tham chiếu từ bên ngoài qua 2 chân RA3 và RA2.

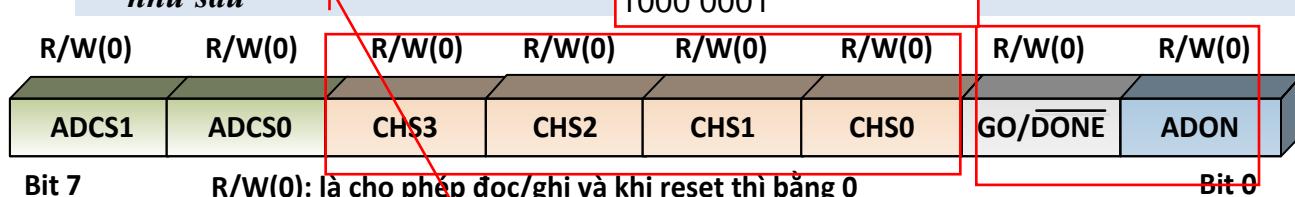
Khối ADC độc lập với CPU nên có thể hoạt động khi CPU đang ở chế độ ngủ do xung cung cấp cho ADC lấy từ dao động RC bên trong của khối ADC.

7.2.2 KHẢO SÁT CÁC THANH GHI CỦA PIC 16F887

Khối ADC có 4 thanh ghi:

- Thanh ghi lưu kết quả byte cao: ADRESH (A/D Result High Register)
- Thanh ghi lưu kết quả byte thấp: ADRESL (A/D Result Low Register)
- Thanh ghi điều khiển ADC thứ 0: ADCON0 (A/D Control Register 0)
- Thanh ghi điều khiển ADC thứ 1: ADCON1 (A/D Control Register 1)

a. *Thanh ghi ADCON0 (ADC Control register) chứa các bit điều khiển khối ADC như sau*



Hình 7-2. Thanh ghi ADCON0.

Bit 7-6 ADCS<1:0>: Các bit lựa chọn xung chuyển đổi AD (AD Conversion Clock Select bits)

00 = Fosc/2

01 = Fosc/8

$$0x1F = 0x81;$$

#BYTE ADCON0=0X1F
ADCON0 = 0X81;
(tr. 30)

10 = Fosc/32

11 = F_{RC} (xung clock lấy từ bộ dao động nội bên trong có tần số lớn nhất là 500 kHz)

Bit 5-3 **CHS<3:0>**: Các bit lựa chọn kênh tương tự (Analog Channel Select bits) như bảng 7-1.

Bảng 7-1. Chọn kênh tương tự của 4 bit CHS.

CHS<3:0>	Kênh	CHS<3:0>	Kênh	CHS<3:0>	Kênh	CHS<3:0>	Kênh
0000	0	0100	4	1000	8	1100	12
0001	1	0101	5	1001	9	1101	13
0010	2	0110	6	1010	10	1110	C _{VREF}
0011	3	0111	7	1011	11	1111	Điện áp tham chiếu cố định bằng 0,6V

Bit 2 **GO / \overline{DONE}** : bit báo trạng thái chuyển đổi ADC (A/D Conversion status bit)

Khi cho $GO / \overline{DONE} = 1$ để bắt đầu thực hiện quá trình chuyển đổi.

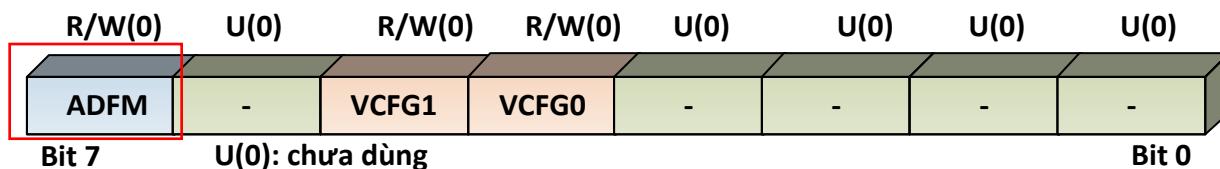
Sau khi chuyển đổi xong thì mạch chuyển đổi làm bit GO / \overline{DONE} xuống mức 0.

Bit 0 **ADON**: bit mở nguồn cho ADC hoạt động (AD ON bit):

ADON = 1 có chức năng mở nguồn cho khôi chuyển đổi ADC hoạt động.

ADON = 0 sẽ tắt nguồn khôi chuyển đổi ADC để giảm công suất tiêu thụ.

b. Thanh ghi ADCON1 thiết lập các chân của port là tương tự hoặc xuất nhập số IO.



Hình 7-3. Thanh ghi ADCON1.

Bit 7 **ADFM**: bit lựa chọn định dạng kết quả ADC (AD Result Format Select bit):

ADFM = 1: có chức năng canh lè phải, 6 bit MSB của ADRESH có giá trị là ‘0’.

ADFM = 0: có chức năng canh lè trái, 6 bit LSB của ADRESL có giá trị là ‘0’.

Bit 5 **VCFG1**: bit lựa chọn điện áp tham chiếu (Voltage reference bit)

VCFG1 = 1: có chức năng nối điện áp tham chiếu Vref- với ngõ vào AN3.

VCFG1 = 0: có chức năng nối điện áp tham chiếu Vref- với V_{SS}.

Bit 4 **VCFG0**: bit lựa chọn điện áp tham chiếu (Voltage reference bit)

VCFG0 = 1: có chức năng nối điện áp tham chiếu Vref+ với ngõ vào AN3.

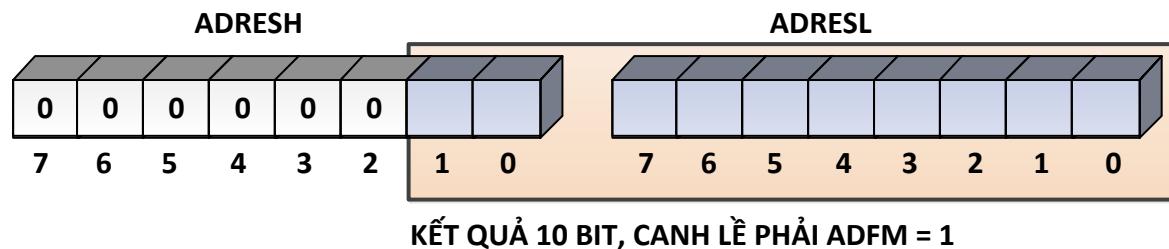
VCFG0 = 0: có chức năng nối điện áp tham chiếu Vref+ với V_{DD}.

Bit 6, 3-0: chưa dùng nếu đọc sẽ có giá trị là ‘0’

c. Thanh ghi ADRESH và ADRESL

Cặp thanh ghi 16 bit ADRESH: ADRESL dùng để lưu kết quả chuyển đổi 10 bit của ADC sau khi chuyển đổi xong. Do kết quả chỉ có 10 bit nhưng lưu trong cặp thanh ghi 16 bit nên có 2 kiểu định dạng tùy thuộc vào bit ADFM (ADC Format).

Hình sau trình bày 2 kiểu định dạng của cặp thanh ghi kết quả:



KẾT QUẢ 10 BIT, CANH LỀ TRÁI ADFM = 0

Hình 7-4. Định dạng cặp thanh ghi lưu kết quả.

Chức năng của canh lè phục vụ cho 2 khả năng xử lý kết quả: nếu lấy kết quả 10 bit để xử lý thì nên chọn chế độ canh lè phải, còn nếu lấy kết quả 8 bit thì chọn chế độ canh lè trái và chỉ lấy kết quả của thanh ghi byte cao ADRESH, bỏ đi 2 bit có trọng số thấp nhất của thanh ghi ADRESL và chú ý đến tính toán giá trị, trong chương trình C nếu không khai báo thuộc tính ADC 10 bit thì phần mềm tự động lấy giá trị 8 bit cao.

7.2.3 TRÌNH TỰ THỰC HIỆN CHUYỂN ĐỔI ADC

Để thực hiện chuyển đổi ADC thì phải thực hiện các bước sau:

Bước 1: Cấu hình cho port:

- Cấu hình cho các port ở chế độ ngõ vào tương tự.
- Không được định cấu hình cho các port ở chế độ xuất dữ liệu.

Bước 2: Cấu hình cho module ADC:

- Chọn xung clock cho chuyển đổi ADC.
- Định cấu hình cho điện áp chuẩn.
- Chọn kênh ngõ vào tương tự cần chuyển đổi.
- Chọn định dạng cho 2 thanh ghi lưu kết quả.
- Mở nguồn cho ADC.

Bước 3: Thiết lập cấu hình ngắt ADC nếu sử dụng:

- Xóa cờ báo ngắt ADIF của ADC.
- Cho bit ADIE bằng 1 để cho phép ADC ngắt.
- Cho bit PEIE bằng 1 để cho phép ngắt ngoại vi.
- Cho bit GIE bằng 1 để cho phép ngắt toàn cục.

Bước 4: Chờ hết thời gian ổn định theo yêu cầu.

Bước 5: Bắt đầu chuyển đổi bằng cách cho bit GO/DONE lên 1

Bước 6: Kiểm tra chuyển đổi ADC kết thúc bằng cách:

- Kiểm tra liên tục bit GO/DONE nếu về 0 thì quá trình chuyển đổi kết thúc.
- Nếu dùng ngắt thì chờ ngắt ADC xảy ra.

Bước 7: Đọc cặp thanh ghi kết quả (ADRESH: ADRESL), xóa bit ADIF nếu dùng ngắt.

Bước 8: Thực hiện chuyển đổi kế tiếp.

7.2.4 LỰA CHỌN NGUỒN XUNG CHO CHUYỂN ĐỔI ADC

Tần số xung clock cho bộ chuyển đổi ADC được lựa chọn bằng phần mềm bởi các bit ADCS nằm trong thanh ghi ADCON0. Có 4 lựa chọn cho nguồn xung clock như sau:

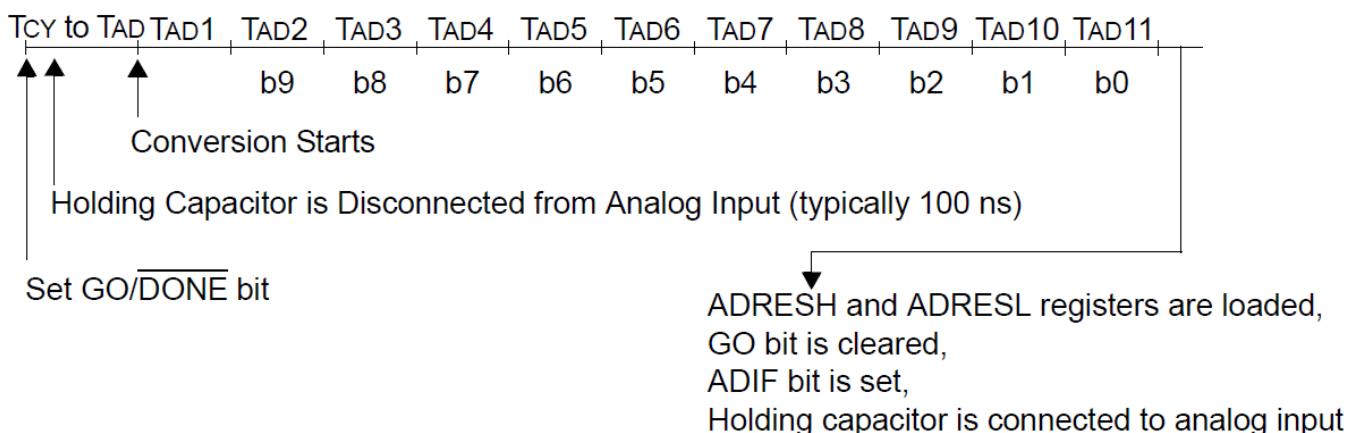
- Fosc/2
- Fosc/8
- Fosc/32
- F_{RC} lấy từ bộ dao động bên trong.

Hệ số chia được lựa chọn tùy thuộc vào tần số của hệ thống theo bảng 7-2.

Bảng 7-2. Tần số xung clock tùy chọn phụ thuộc vào tần số bộ dao động.

Chu kỳ xung clock của ADC (T _{AD})	Tần số thiết bị (Fosc)				
Xung clock cho ADC	ADCS<1:0>	20MHz	8MHz	4MHz	1MHz
Fosc/2	00	100ns	250ns	500ns	2μs
Fosc/8	01	400ns	1μs	2μs	8μs
Fosc/32	10	1.6μs	4μs	8μs	32μs
khi mô phỏng Proteus F _{RC}	11	2-6μs	2-6μs	2-6μs	2-6μs

Thời gian chuyển đổi ADC cho mỗi bit được xác định bởi chu kỳ T_{AD}. Để chuyển đổi hoàn tất 10 bit sẽ dùng tối thiểu 11 chu kỳ T_{AD} như hình 7-5.



Hình 7-5. Thời gian chuyển đổi 10 bit.

7.3 CÁC LỆNH CỦA ADC TRONG NGÔN NGỮ CCS-C

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)

7.3.1 LỆNH SETUP_ADC(MODE)Cú pháp: setup_adc (*mode*);Thông số: *mode*- Analog to digital mode. Xem trong file device.

Chức năng: định cấu hình cho ADC

Có hiệu lực: Cho các PIC có tích hợp ADC.

Ví dụ 7-1: setup_adc(ADC_CLOCK_INTERNAL);**7.3.2 LỆNH SETUP_ADC_PORT(VALUE)**Cú pháp: setup_adc_ports (*value*)Hàng số: *value* - là hàng số định nghĩa trong file devices .h

Chức năng: Thiết lập chân của ADC là tương tự, số hoặc tổ hợp cả 2.

Hiệu lực: Cho các PIC có ADC.

Ví dụ 7-2: setup_adc_ports(ALL_ANALOG); // sử dụng tất cả các kênh tương tự**7.3.3 LỆNH SET_ADC_CHANNEL(chan)**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 1 khoảng thời gian ngắn khi chuyển kênh, thường thì thời gian chờ khoảng 10μs.

Ví dụ 7- 3: SET_ADC_CHANNEL(2);

DELAY_US(10);

VALUE = READ_ADC();

7.3.4 LỆNH value=READ_ADC(mode)Cú pháp: value = read_adc ([*mode*]) - đọc kết quả sau khi chuyển đổi xong

Hàng số: mode là 1 trong các hàng 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

năng: Khai báo #DEVICE ADC= directive như sau:

#DEVICE 8 bit 10 bit 11 bit 16 bit

Ví dụ 7-4: setup_adc(ADC_CLOCK_INTERNAL);

setup_adc_ports(ALL_ANALOG);

set_adc_channel(1);

read_adc(ADC_START_ONLY);

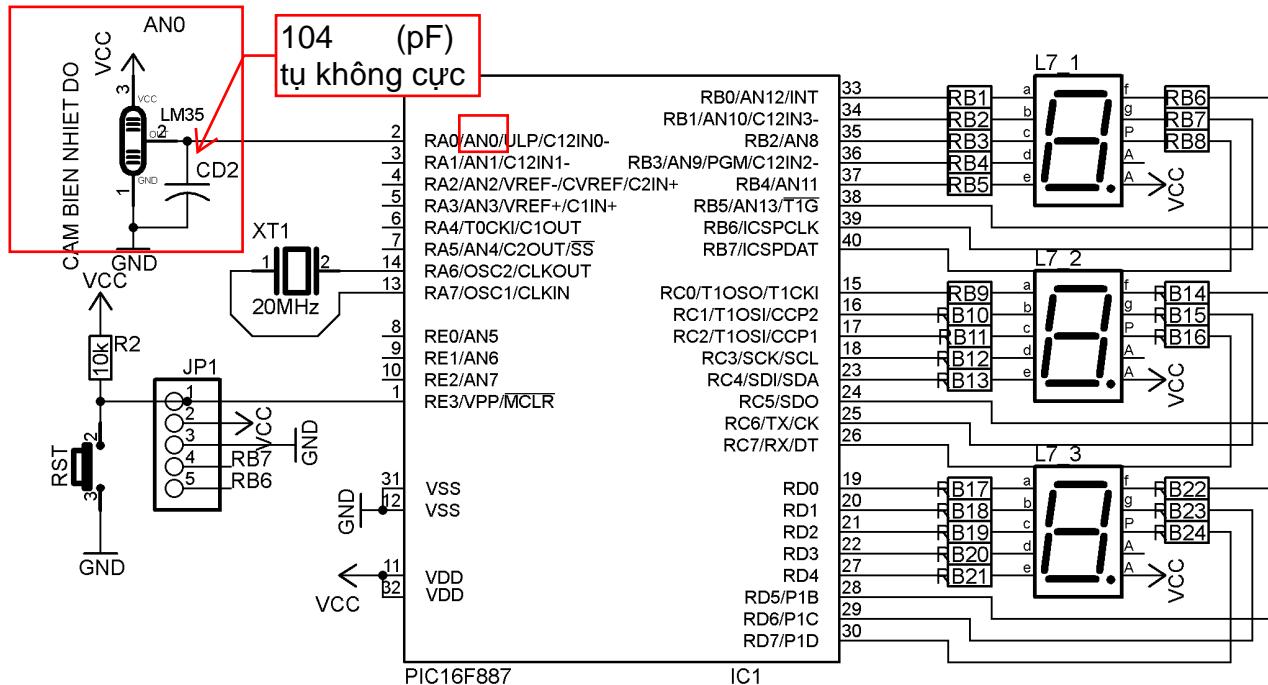
sleep();

value=read_adc(ADC_READ_ONLY);

7.4 ÚNG DỤNG ADC CỦA PIC 16F887**7.4.1 ĐO NHIỆT ĐỘ DÙNG CẢM BIẾN LM35**

Bài 7-1: Dùng ADC của vi điều khiển PIC 16F887 và cảm biến LM35 để đo nhiệt độ hiển thị trên 3 led 7 đoạn anode chung dùng 3 port, chỉ hiển thị đơn vị, chục và trăm. Sử dụng nguồn điện áp tham chiếu V_{REF+} nối với V_{DD} bằng 5V và V_{REF-} nối với nguồn V_{SS} bằng 0V.

- Sơ đồ mạch:



Hình 7-6. Sơ đồ mạch đo nhiệt độ dùng PIC16F887 hiển thị trên 3 led trực tiếp.

- Tính toán độ phân giải

Theo yêu cầu thì các ngõ vào điện áp tham chiếu là $V_{REF-} = 0V, V_{REF+} = V_{DD} = 5V$

Nên độ phân giải (step size) là $SS = \frac{V_{REF+} - V_{REF-}}{2^{10} - 1} = \frac{5000mV}{1023} = 4,887mV$

Thông số 2^{10} là do ADC 10 bit.

Điện áp toàn giao full scale = FS: $FS = SS \times 1023 = 4,887mV \times 1023 = 5,000mV = 5V$

Độ phân giải ADC là 4,887mV không tương thích với độ phân giải của cảm biến LM35 bằng 10mV, tỉ lệ chênh lệch là:

$$\frac{10mV}{4,887mV} = 2,046$$

Để có kết quả hiển thị đúng thì lấy kết quả chuyển đổi chia cho tỉ lệ chênh lệch.

- Lưu đồ:



Hình 7-7. Lưu đồ chuyển đổi ADC đo nhiệt độ kênh thứ 0.

- Chương trình:

```

#include<16F887.C>
#DEVICE    ADC=10          khai báo ngay sau dòng
#include<16F887.h>

UNSIGNED INT16  KQADC;
UNSIGNED INT     J, MATRAM;

VOID  HIEN THI()
{
    MATRAM = MA7DOAN[KQADC/100];
    IF (MATRAM == 0xC0)           MATRAM=0xFF;
    OUTPUT_B(MA7DOAN[KQADC%10]);
    OUTPUT_C(MA7DOAN[KQADC/10%10]);
    OUTPUT_D(MATRAM);
}

VOID MAIN()
{
    SET_TRIS_B(0x00);   SET_TRIS_D(0x00);
    SET_TRIS_C(0x00);   SET_TRIS_A(0xFF);
    SETUP_ADC(ADC_CLOCK_DIV_8); //khởi tạo ADC
    SETUP_ADC_PORTS(SAN0);
    SET_ADC_CHANNEL(0);
    WHILE(TRUE)
    {
        KQADC=0;
        FOR (J=0; J<200; J++)
        {
            KQADC=KQADC+READ_ADC(); //đọc adc
            DELAY_MS(1);
        }
        KQADC= KQADC /2.046;
        KQADC=KQADC/200;
        HIEN THI();
    }
}

```

- Giải thích chương trình:

Trong vòng lặp while ta cho biến KQADC bằng 0, vòng lặp for thực hiện 200 lần đo, kết quả cộng dồn vào biến KQADC.

Kết quả đúng của 200 lần đo sẽ chia cho 200 lần để được kết quả đo trung bình và chia cho hệ số 2.046 để có kết quả đúng hệ số độ phân giải, sau đó tiến hành giải mã hiển thị và xoá số 0 vô nghĩa.

7.4.2 ĐO NHIỆT ĐỘ CÓ ĐIỀU KHIỂN TẢI BÓNG ĐÈN BẰNG RELAY

Bài 7-2: Thêm vào bài 7-1 yêu cầu: khi nhiệt độ tăng nhưng nhỏ hơn 40 độ thì relay đóng, khi lớn hơn 40 độ thì relay ngắt, khi nhiệt độ giảm thấp hơn 35 độ thì relay đóng trở lại.

- Sơ đồ mạch: như hình 7-8.
- Giải thích sơ đồ mạch:

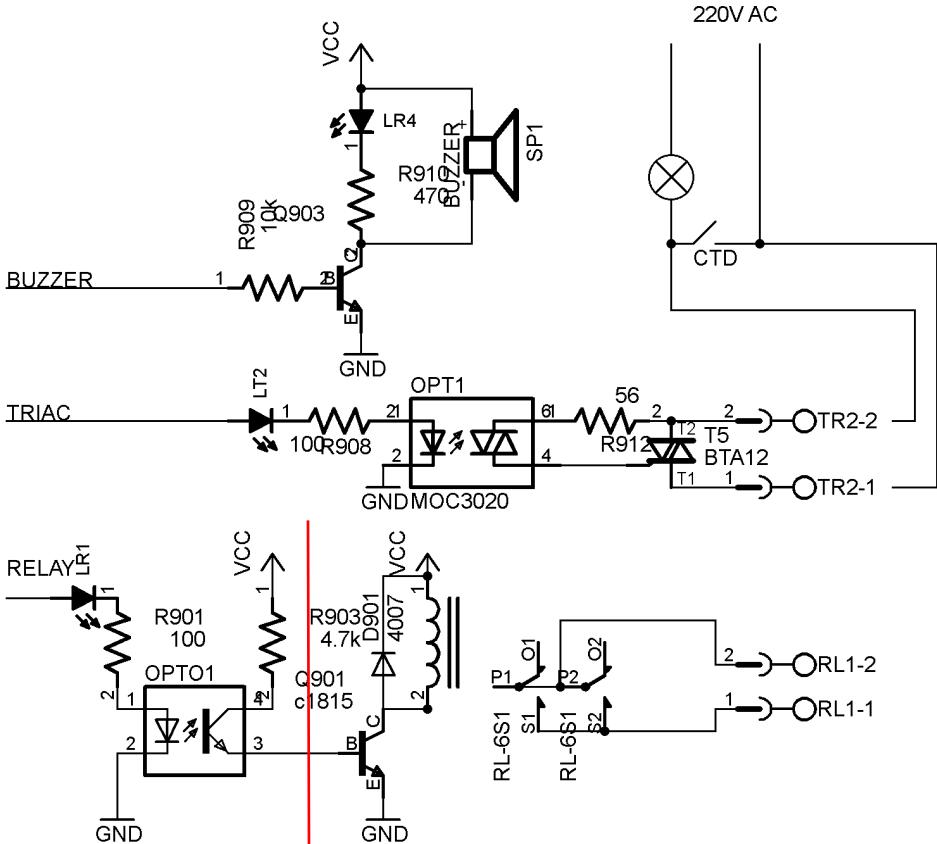
Sơ đồ mạch gồm 3 đối tượng điều khiển:

Buzzer để tạo âm thanh bip dùng 1 buzzer và mạch khuếch đại dùng transistor, transistor với dòng làm việc lớn hơn dòng của buzzer khoảng 120mA. Mức 1 làm buzzer kêu, mức 0 tắt buzzer.

Mạch điều khiển tải ac dùng triac và opto cách ly. Mức 1 làm triac dẫn, mức 0 làm triac tắt.

Mạch điều khiển tải công suất dùng relay và opto cách ly. Mức 1 làm opto dẫn, transistor dẫn, relay dẫn, mức 0 làm opto tắt, transistor tắt, relay tắt.

Khi sử dụng đối tượng nào thì ta chỉ cần kết nối 1 port của vi điều khiển với 1 trong các mạch trên.



Hình 7-8. Sơ đồ mạch giao tiếp điều khiển relay, triac, buzzer.

- Lưu đồ:



Hình 7-9. Lưu đồ chuyển đổi ADC đo nhiệt độ kênh thứ 0.

- Chương trình:

```
#DEFINE RELAY PIN_A3
IF (KQADC>40) OUTPUT_LOW(RELAY);
ELSE IF (KQADC<35) OUTPUT_HIGH(RELAY);
```

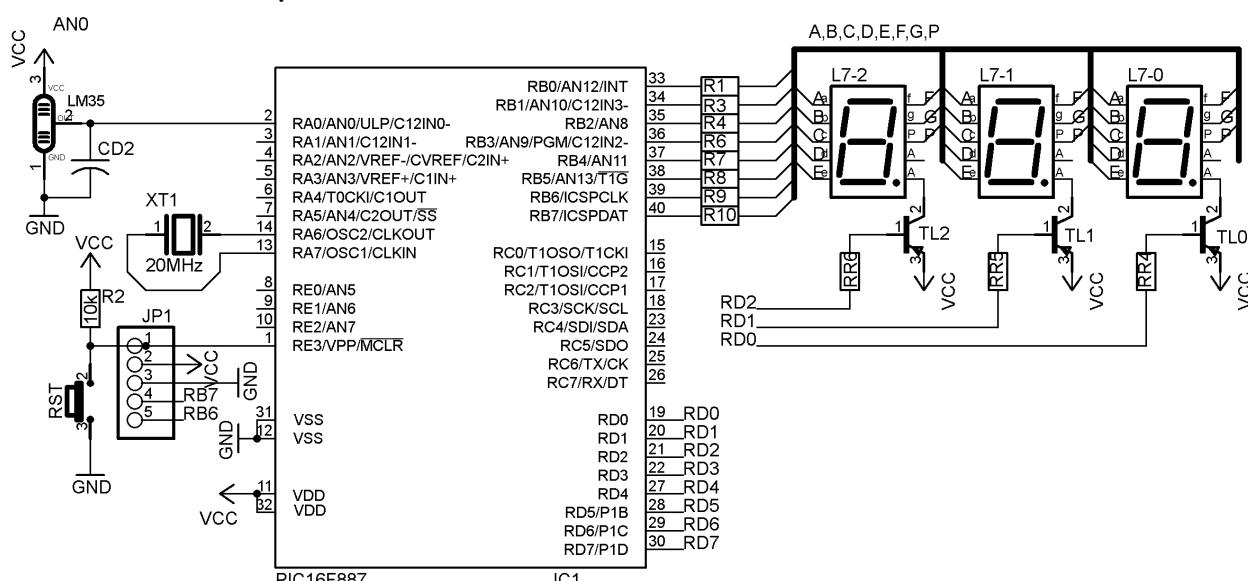
- Giải thích chương trình:

Chương trình này giống như bài 7-1, chỉ thêm phần khai báo định nghĩa bit điều khiển RELAY là RA3 (chọn port nào chưa dùng là được). Trong chương trình chính thì thêm hàng lệnh trên có chức năng so sánh nếu nhiệt độ lớn hơn 40 thì relay tắt, nhỏ hơn 35 thì mở.

7.4.3 ĐO NHIỆT ĐỘ HIỂN THỊ TRÊN LED 7 ĐOẠN QUÉT

Bài 7-3: Dùng ADC của vi điều khiển PIC 16F887 và cảm biến LM35 để đo nhiệt độ hiển thị trên 3 led 7 đoạn anode chung kết nối thep phương pháp quét, sử dụng nguồn điện áp tham chiếu V_{REF+} nối với V_{DD} bằng 5V và V_{REF-} nối với nguồn V_{SS} bằng 0V.

- Sơ đồ mạch:



Hình 7-10. Sơ đồ mạch đo nhiệt độ dùng PIC16F887, hiển thị 3 led quét.

- Giải thích sơ đồ mạch:

Mạch dùng 3 led 7 đoạn để hiển thị kết quả nhiệt độ kết nối theo phương pháp quét sử dụng port B và 3 bit của port D để điều khiển 3 transistor.

Cảm biến LM35 nối với ngõ vào kênh tương tự thứ 0 (SAN0).

Mạch reset và các chân để kết nối mạch nạp, tụ thạch anh 20MHz.

- Lưu đồ: như hình 7-11.

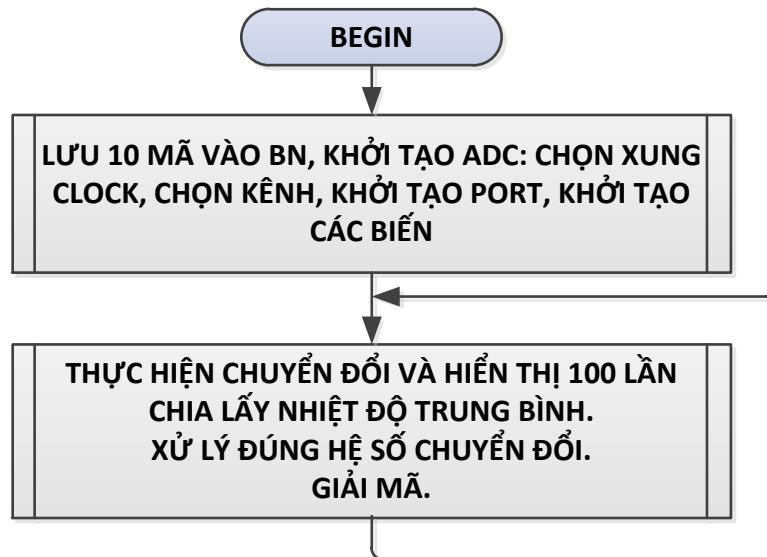
- Chương trình:

```
#INCLUDE<TV_16F887.C>
UNSIGNED INT16 KQADC;
UNSIGNED INT J, MADONVI, MACHUC, MATRAM;
VOID GIAIMA ()
{
    MATRAM = MA7DOAN[KQADC /100];
    MACHUC = MA7DOAN[KQADC /10 % 10];
    MADONVI= MA7DOAN[KQADC % 10];
    IF (MATRAM == 0XC0) MATRAM=0xFF;
}
VOID HIENTHI ()
```

```

OUTPUT_B(MADONVI);
DELAY_MS(1);
OUTPUT_B(MACHUC);
DELAY_MS(1);
OUTPUT_B(MATRAM);
DELAY_MS(1);
}
VOID MAIN()
{
    SET_TRIS_B(0x00);    SET_TRIS_D(0x00);        SET_TRIS_A(0x01);
SETUP_ADC(ADC_CLOCK_DIV_2);
SETUP_ADC_PORTS(SAN0);
SET_ADC_CHANNEL(0);
WHILE (TRUE)
{
    KQADC=0;
    FOR (J=0; J<100; J++)
    {
        KQADC=KQADC+READ_ADC();
        HIENTHI();
    }
    KQADC= KQADC /2.046;
    KQADC=KQADC/100;
    GIAIMA();
}
}

```



Hình 7-11. Lưu đồ chuyển đổi ADC đo nhiệt độ hiển thị led quét.

- Giải thích chương trình:

Trong vòng lặp while ta cho biến KQADC bằng 0, vòng lặp for thực hiện 100 lần đo, kết quả cộng dồn vào biến KQADC.

Chương trình này giống như các bài trước, sự khác nhau là trong quá trình chuyển đổi thì thực hiện hiển thị đê 3 led 7 đoạn sáng liên tục và vì thời gian quét 3 led dài nên ta giảm bớt số lần đo trung bình xuống còn 100 để tăng nhanh đáp ứng thời gian.

Ở các bài này thì công việc chính là chuyển đổi ADC và hiển thị, nếu các chương trình ứng dụng lớn thực hiện nhiều công việc thì ta phải giảm số lần đo trung bình xuống cho phù hợp với đáp ứng thời gian.

(a) 2

(b) 4

(c) 3

(d) 8

Câu 7-13: Các bit lựa chọn xung clock cho ADC của PIC 16F887 có tên là:

(a) ADCS2: ADCS0

(b) PCFG3: PCFG0

(c) PCFG2: PCFG0

(d) CHS3: CHS0

Câu 7-14: Lệnh “SETUP_ADC(MODE)” của PIC 16F887 có chức năng là:

(a) Chọn kênh

(b) Chọn độ phân giải

(c) Đọc giá trị

(d) Định cấu hình

Câu 7-15: Bit ra lệnh chuyển đổi ADC của PIC 16F887 có tên là:

(a) ADCS2: ADCS0

(b) GO/DONE

(c) ADON

(d) ADFM

Câu 7-16: Bit chọn định dạng kết quả ADC cạnh trái hoặc phải của PIC16F887 có tên là:

(a) ADCS2: ADCS0

(b) GO/DONE

(c) ADON

(d) ADFM

Câu 7-17: Lệnh “SETUP_ADC_PORT(VALUE)” của PIC 16F887 có chức năng là:

(a) Thiết lập các kênh

(b) Chọn độ phân giải

(c) Đọc giá trị

(d) Chọn tốc độ

Câu 7-18: Lệnh “SET_ADC_CHANNEL(CHAN)” của PIC 16F887 có chức năng là:

(a) Thiết lập các kênh

(b) Chọn độ phân giải

(c) Đọc giá trị

(d) Chọn kênh

Câu 7-19: Lệnh “VALUE=READ_ADC(MODE)” của PIC 16F887 có chức năng là:

(a) Thiết lập các kênh

(b) Chọn độ phân giải

(c) Đọc giá trị

(d) Chọn tốc độ

Câu 7-20: Công thức tính giá trị số nhị phân sau khi chuyển đổi ADC của PIC16F887 là:

$$(a) N = \frac{V_{REF+} - V_{REF-}}{2^{10} - 1}$$

$$(b) N = \frac{V_I - V_{REF+}}{SS}$$

$$(c) N = \frac{V_I - V_{REF-}}{SS}$$

$$(d) N = \frac{V_I - V_{REF-}}{SS} (2^{10})$$

7.5.4 BÀI TẬP

Bài tập 7-1: Mạch đo nhiệt độ 2 kênh dùng cảm biến LM35, vi điều khiển PIC16F887 và 3 led 7 đoạn.

Hãy vẽ mạch, viết lưu đồ và chương trình đo nhiệt độ 2 kênh lần lượt hiển thị trên 3 led, mỗi kênh đo trong khoảng thời gian 1s, định thời 1 giây chuyển kênh dùng Timer T1.

Bài tập 7-2: Mạch đo nhiệt độ 8 kênh dùng cảm biến LM35, vi điều khiển PIC16F887 và 3 led 7 đoạn.

Hãy vẽ mạch, viết lưu đồ và chương trình đo nhiệt độ 8 kênh lần lượt hiển thị trên 3 led, mỗi kênh đo trong khoảng thời gian 1s, định thời 1 giây chuyển kênh dùng Timer T1.

Bài tập 7-3: Mạch đo nhiệt độ dùng cảm biến LM35, vi điều khiển PIC16F887 và LCD 16x2.

Hãy vẽ mạch, viết lưu đồ và chương trình đo nhiệt độ nối với kênh AN0 và hiển thị kết quả trên LCD.

Bài tập 7-4: Mạch đo nhiệt độ 2 kênh dùng 2 cảm biến LM35 và vi điều khiển PIC16F887 hiển thị kết quả đo kênh 1 ở hàng 1, kênh 2 ở hàng 2 của LCD 16x2.

Hãy vẽ mạch, viết lưu đồ và chương trình.

Bài tập 7-5: Mạch đo nhiệt độ 1 kênh dùng cảm biến LM35 và đếm sản phẩm dùng vi điều khiển PIC16F887 hiển thị kết quả đo hàng 1 và kết quả đếm ở hàng 2 của LCD 16x2.

Hãy vẽ mạch, viết lưu đồ và chương trình.

Chương 8

VỊ ĐIỀU KHIỂN PIC16F8873

NGẤT

8.1 GIỚI THIỆU

Ngắt có nhiều tiện ích trong điều khiển nên hầu hết các vi điều khiển đều tích hợp, ở chương này chúng ta sẽ khảo sát nguyên lý hoạt động của ngắt, các nguồn ngắt, vector địa chỉ ngắt, viết chương trình con phục vụ ngắt cho các ứng dụng.

Sau khi kết thúc chương này bạn có thể sử dụng được ngắt của các vi điều khiển.

8.2 TỔNG QUAN VỀ NGẮT

Ngắt sử dụng trong vi xử lý hay vi điều khiển hoạt động như sau: vi xử lý hay vi điều khiển luôn thực hiện một chương trình thường gọi là chương trình chính, khi có tác động từ bên ngoài bằng phần cứng hay tác động bên trong làm cho vi xử lý ngừng thực hiện chương trình chính để thực hiện một chương trình khác (còn gọi là **chương trình phục vụ ngắt ISR**) và sau khi thực hiện xong vi xử lý trở lại thực hiện tiếp chương trình chính. Quá trình làm gián đoạn vi xử lý thực hiện chương trình chính xem như là ngắt.

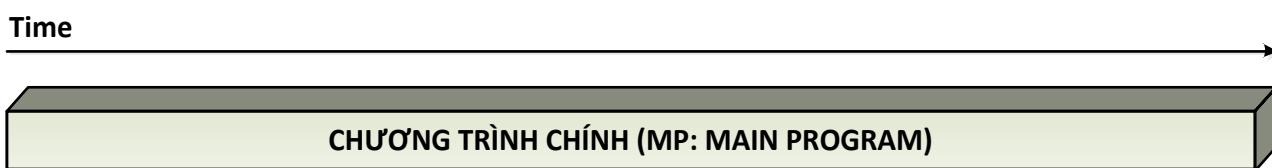
Có nhiều tác động làm ngừng chương trình chính gọi là các nguồn ngắt, ví dụ khi timer/counter đếm tràn sẽ phát sinh yêu cầu ngắt.

Ngắt đóng một vai trò quan trọng trong lập trình điều khiển, vi xử lý hay vi điều khiển sử dụng ngắt để đáp ứng nhiều sự kiện quan trọng khác trong khi vẫn đảm bảo thực hiện được chương trình chính.

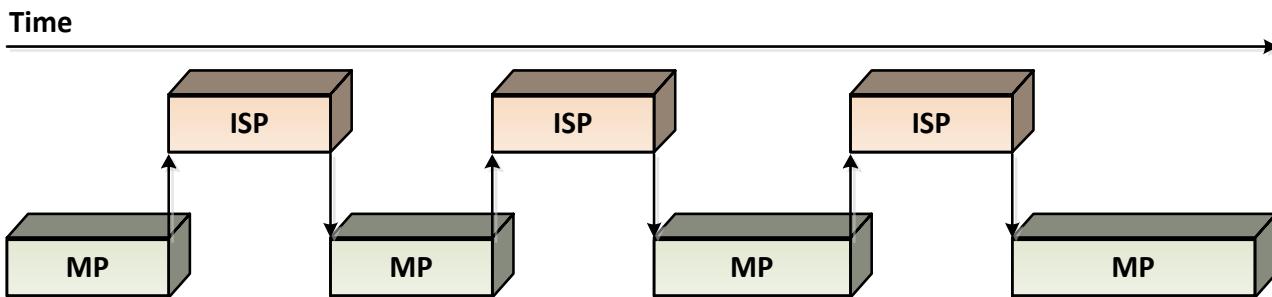
Ví dụ trong khi vi điều khiển đang thực hiện chương trình chính thì có dữ liệu từ hệ thống khác gởi đến, khi đó vi điều khiển ngừng chương trình chính để thực hiện chương trình phục vụ ngắt nhận dữ liệu xong rồi trở lại tiếp tục thực hiện chương trình chính, hoặc có một tín hiệu báo ngắt từ bên ngoài thì vi điều khiển sẽ ngừng thực hiện chương trình chính để thực hiện chương trình ngắt rồi tiếp tục thực hiện chương trình chính.

Có thể sử dụng ngắt để yêu cầu vi điều khiển thực hiện nhiều chương trình cùng một lúc có nghĩa là các chương trình được thực hiện xoay vòng.

CPU thực hiện chương trình trong trường hợp có ngắt và không có ngắt như hình 8-1.



(a) Chương trình chính được thực hiện liên tục, không bị gián đoạn.



(b) Chương trình chính bị gián đoạn để thực hiện chương trình con phục vụ ngắt.

Hình 8-1. CPU thực hiện chương trình chính trong 2 trường hợp không và có ngắt.

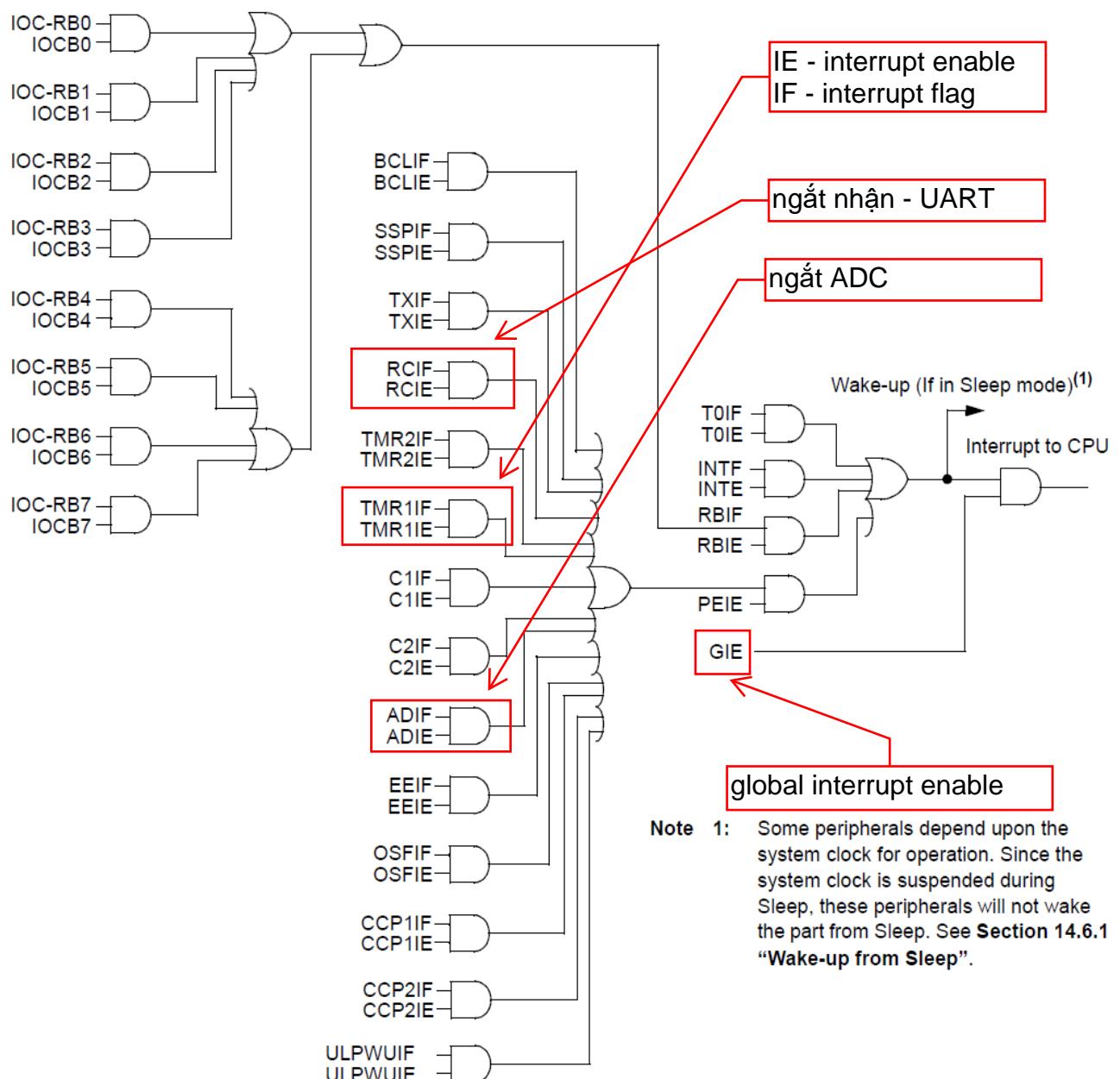
8.3 NGẮT CỦA VI ĐIỀU KHIỂN PIC16F887

8.3.1 CÁC NGUỒN NGẮT CỦA PIC16F887

Vi điều khiển PIC 16F887 có nhiều nguồn ngắt:

- Ngắt ngoài RB0/INT.

- Ngắt của timer T0 khi đếm tràn.
- Ngắt của timer T1 khi đếm tràn.
- Ngắt của timer T2 khi giá trị đếm bằng giá trị của thanh ghi PR2.
- Ngắt portB thay đổi.
- Ngắt của 2 bộ so sánh điện áp tương tự.
- Ngắt của bộ chuyển đổi ADC.
- Ngắt khi ghi dữ liệu vào Eeprom.
- Ngắt khi bộ giám sát phát hiện nguồn xung clock bị hỏng.
- Ngắt của khối CCP tăng cường.



Hình 8-2. Mạch điện ngắt của PIC16F887.

- Ngắt truyền và nhận dữ liệu EUSART.
- Ngắt đánh thức CPU với nguồn công suất cực thấp.

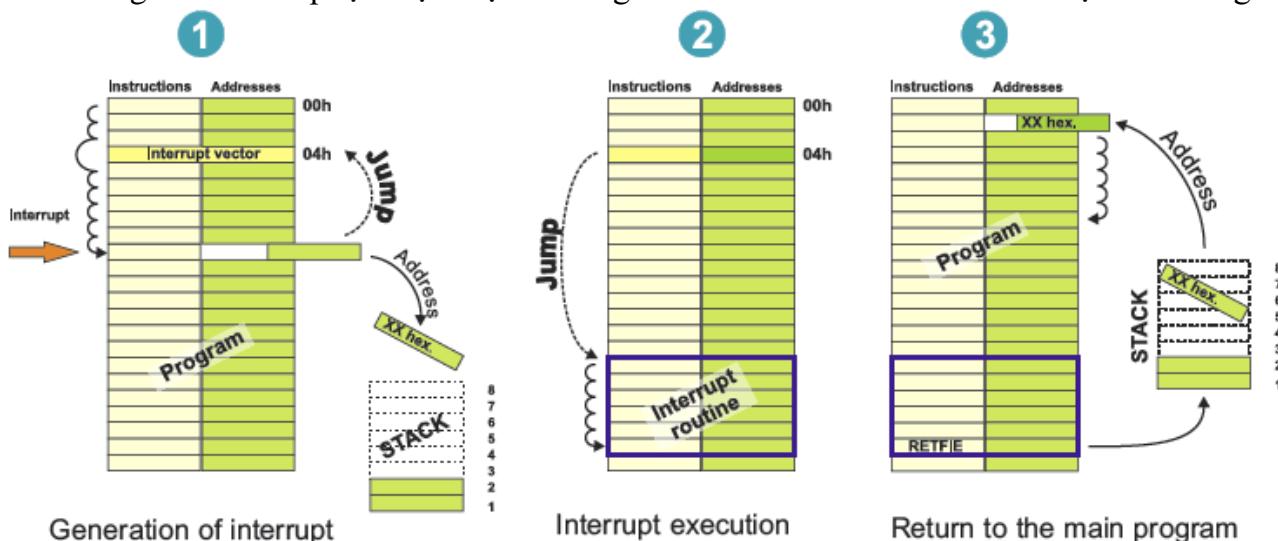
- Ngắt của khối truyền dữ liệu đồng bộ MSSP.

8.3.2 CẤU TRÚC MẠCH ĐIỆN NGẮT CỦA PIC16F887

Cấu trúc mạch điện ngắt của PIC16F877 như hình 8-2.

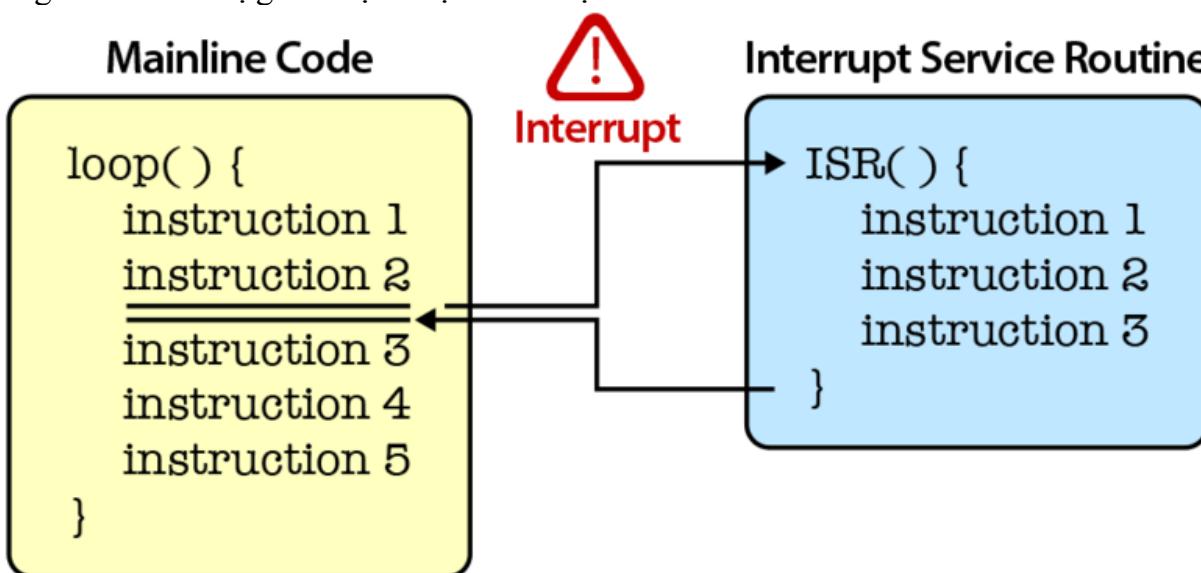
Trong mạch điện chúng ta có thể nhìn thấy để CPU thực hiện ngắt của timer T0 thì phải hội đủ các điều kiện sau: bit cho phép ngắt T0IE = 1 và bit cho phép ngắt toàn cục GIE = 1. Khi timer T0 đếm và tràn sẽ làm cờ báo ngắt T0IF = 1 và khi đó ngắt sẽ xảy ra.

Khi một ngắt được đáp ứng thì bit GIE bị xóa để không cho phép bắt kí ngắt nào khác xảy ra nữa, địa chỉ trở về được cất vào trong ngăn xếp và thanh ghi PC được nạp địa chỉ 0004H và chương trình tiếp tục được thực hiện tại địa chỉ 0004H cho đến khi thực hiện xong chương trình con phục vụ ngắt kết thúc bởi lệnh return thì lấy lại địa chỉ đã lưu trong ngăn xếp trả lại cho thanh ghi PC để tiếp tục thực hiện chương trình như hình 8-3 và đã thảo luận ở chương 2.



Hình 8-3. Bộ nhớ ngăn xếp khi thực hiện ngắt và kết thúc ngắt.

Chương trình đang được thực hiện thường là chương trình chính và khi ngắt xảy ra thì chương trình chính bị gián đoạn được minh họa như hình 8-4.



Hình 8-4. Thực hiện chương trình chính, bị ngắt và kết thúc ngắt.

Đang thực hiện đến lệnh thứ 2 thì ngắt xảy ra, CPU ngừng thực hiện chương trình chính, nhảy sang thực hiện chương trình con phục vụ ngắt và sau khi thực hiện xong quay trở lại thực hiện tiếp chương trình chính.

Một câu hỏi đặt ra là do chỉ có 1 vector địa chỉ ngắt là 0004H thì làm sao để biết ngắt nào đang thực hiện?

Câu trả lời là: Nếu trong chương trình chỉ sử dụng có 1 ngắt thì khi ngắt xảy ra thì chương trình con phục ngắt đó sẽ được thực hiện.

Nếu trong chương trình sử dụng nhiều nguồn ngắt thì chương trình ngắt viết tại địa chỉ 0004H sẽ kiểm tra xem cờ báo ngắt nào lên 1, nếu bằng 0 thì kiểm tra tiếp cờ khác, nếu bằng 1 thì thực hiện chương trình ngắt tương ứng.

Quá trình kiểm tra cờ báo ngắt nào bằng 1 và thực hiện chương trình phục vụ ngắt tương ứng trước cũng xác định luôn thứ tự ưu tiên ngắt.

Thứ tự ngắt và ưu tiên ngắt của PIC16F887 khác với vi điều khiển AT89S52. Ở vi điều khiển AT89S52 thì mỗi ngắt có 1 địa chỉ cố định và có 2 cấp ưu tiên cao và thấp được phân chia cố định, trong cùng cấp ưu tiên thì kiểm tra quét vòng, gấp ngắt nào trước thì phục vụ trước.

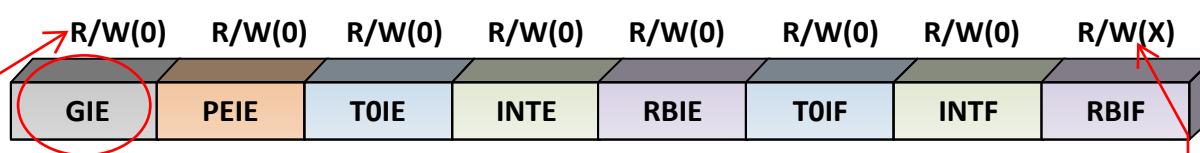
8.3.3 CÁC THANH GHI NGẮT CỦA PIC16F887

Trong vi điều khiển PIC16F887 có 5 thanh ghi phục vụ cho ngắt là INTCON, PIE1, PIE2, PIR1, PIR2. Tóm tắt chức năng của từng thanh ghi như sau:

a. **Thanh ghi INTCON (Interrupt CONTROL)**

trang 30, cấu trúc bộ nhớ của VĐK

Thanh ghi INTCON có địa chỉ 0x0B và tóm tắt chức năng của thanh ghi như hình sau:



Có thể đọc, ghi (0): reset --> = 0

Hình 8-5. Thanh ghi INTCON.

don't care, không quan tâm, bằng 0 hay 1 cũng được

Bảng 8-1. Tóm tắt chức năng các bit trong thanh ghi INTCON có địa chỉ 0x0B

Bit	Kí hiệu	Chức năng (cho phép = 1; cấm = 0)
INTCON.7	GIE	Bit cho phép/cấm toàn bộ các nguồn ngắt.
INTCON.6	PEIE	Bit cho phép ngắt ngoại vi
INTCON.5	TOIE	Bit cho phép ngắt timer T0.
INTCON.4	INTE	Bit cho phép ngắt ngoài.
INTCON.3	RBIE	Bit cho phép ngắt portB thay đổi.
INTCON.2	TOIF	Cờ báo ngắt của timer T0.
INTCON.1	INTF	Cờ báo ngắt của ngắt ngoài.
INTCON.0	RBIF	Cờ báo ngắt portB thay đổi.

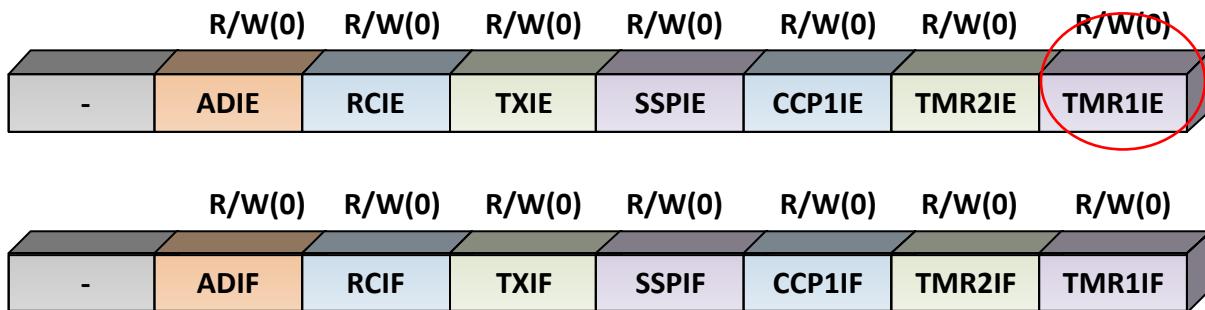
Trong thanh ghi trên thì bit GIE là bit cho phép/cấm ngắt toàn bộ các nguồn ngắt, bit PEIE cho phép/cấm các nguồn ngắt ngoại vi.

Các bit còn lại là bit cho phép/cấm ngắt và cờ báo ngắt của ngắt INT, ngắt portB thay đổi và ngắt của timer0. Bằng 1 thì cho phép, bằng 0 thì không cho phép.

Ví dụ 8-1: Muốn cho phép timer T0 ngắt thì lập trình cho các bit như sau: GIE = 1, TOIE = 1.

b. **Thanh ghi PIE1 (Peripheral Interrupt Enable) và PIR1 (Peripheral Interrupt Request)**

Thanh ghi PIE1 có địa chỉ 0x8C và thanh ghi PIR1 có địa chỉ 0x0C và tổ chức của 2 thanh ghi như hình sau:



Hình 8-6. Thanh ghi PIE1 và PIR1.

Bảng 8-2. Tóm tắt chức năng các bit trong thanh ghi PIE1.

Bit	Kí hiệu	Chức năng (cho phép = 1; cấm = 0)
PIE1.7	-	Bit chưa có chức năng.
PIE1.6	ADIE	Bit cho phép ADC ngắt.
PIE1.5	RCIE	Bit cho phép ngắt nhận dữ liệu.
PIE1.4	TXIE	Bit cho phép ngắt phát dữ liệu.
PIE1.3	SSPIE	Bit cho phép ngắt truyền dữ liệu đồng bộ.
PIE1.2	CCP1IE	Bit cho phép ngắt khối CCP1.
PIE1.1	TMR2IE	Bit cho phép ngắt của timer T2.
PIE1.0	TMR1IE	Bit cho phép ngắt của timer T1.

Bảng 8-3. Tóm tắt chức năng các bit trong thanh ghi cho phép ngắt PIR1.

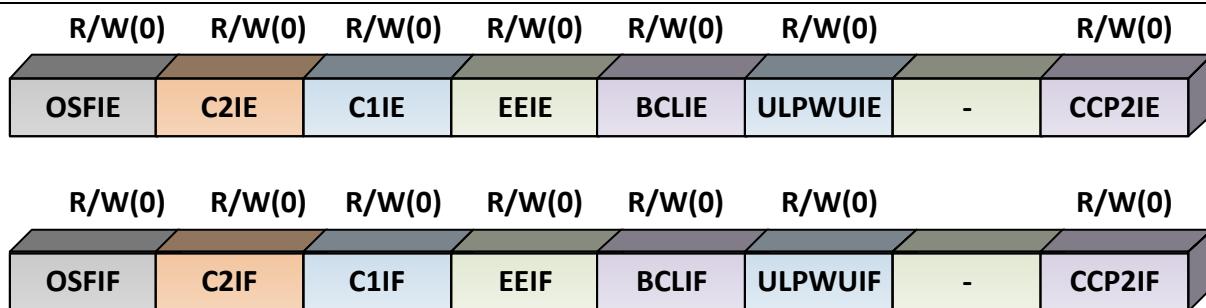
Bit	Kí hiệu	Chức năng (ngắt xảy ra = 1; chưa xảy ra = 0)
PIR1.7	-	Bit chưa có chức năng.
PIR1.6	ADIF	Cờ báo ngắt của ADC.
PIR1.5	RCIF	Cờ báo ngắt nhận dữ liệu.
PIR1.4	TXIF	Cờ báo ngắt phát dữ liệu.
PIR1.3	SSPIF	Cờ báo ngắt truyền dữ liệu đồng bộ.
PIR1.2	CCP1IF	Cờ báo ngắt khối CCP1.
PIR1.1	TMR2IF	Cờ báo ngắt của timer T2.
PIR1.0	TMR1IF	Cờ báo ngắt của timer T1.

Hai thanh ghi chứa các bit cho phép ngắt và cờ báo ngắt tương ứng của 7 ngắt ngoại vi.

Ví dụ 8-2: Muốn cho phép timer2 ngắt thì lập trình cho các bit như sau: GIE = 1, TMR2IE = 1 và bit cho phép ngắt ngoại vi PEIE = 1.

c. **Thanh ghi PIE2 (Peripheral Interrupt Enable) và PIR2 (Peripheral Interrupt Request)**

Thanh ghi PIE2 có địa chỉ 0x8D và thanh ghi PIR2 có địa chỉ 0x0D và tổ chức của 2 thanh ghi như hình sau:



Hình 8-7. Thanh ghi PIE2 và PIR2.

Bảng 8-4. Tóm tắt chức năng các bit trong thanh ghi cho phép ngắt PIE2.

Bit	Kí hiệu	Chức năng (cho phép = 1; cấm = 0)
PIE2.7	OSFIE	Bit cho phép ngắt của bộ dao động hỏng.
PIE2.6	C2IE	Bit cho phép ngắt của khối so sánh 2.
PIE2.5	C1IE	Bit cho phép ngắt của khối so sánh 1.
PIE2.4	EEIE	Bit cho phép ngắt của bộ nhớ eeprom.
PIE2.3	BCLIE	Bit cho phép ngắt xung đột truyền dữ liệu I2C.
PIE2.2	ULWUIE	Bit cho phép ngắt của bộ đánh thức CPU với công suất cực thấp.
PIE2.1	-	Bit chưa có chức năng.
PIE2.0	CCP2IE	Bit cho phép ngắt của CCP2.

Bảng 8-5. Tóm tắt chức năng các bit trong thanh ghi cho phép ngắt PIR2.

Bit	Kí hiệu	Chức năng (cho phép = 1; cấm = 0)
PIR2.7	OSFIF	Cờ báo ngắt của bộ dao động hỏng.
PIR2.6	C2IF	Cờ báo ngắt của khối so sánh 2.
PIR2.5	C1IF	Cờ báo ngắt của khối so sánh 1.
PIR2.4	EEIF	Cờ báo ngắt của bộ nhớ eeprom.
PIR2.3	BCLIF	Cờ báo ngắt xung đột truyền dữ liệu I2C.
PIR2.2	ULWUIF	Cờ báo ngắt của bộ đánh thức CPU với công suất cực thấp.
PIR2.1	-	Bit chưa có chức năng.
PIR2.0	CCP2IF	Cờ báo ngắt của CCP2.

8.4 CÁC LỆNH NGẮT CỦA PIC16F887 TRONG NGÔN NGỮ PIC-C

Các lệnh của ngôn ngữ lập trình C liên quan đến ngắt bao gồm:

8.4.1 LỆNH ENABLE_INTERRUPTS(LEVEL)

Lệnh có chức năng cho phép ngắt.

Khi muốn sử dụng ngắt thì phải tiến hành thực hiện lệnh cho phép ngắt và tên của nguồn ngắt nào (**level**) thì đã được định nghĩa trong **file thư viện** của vi điều khiển tương ứng. Ở vi điều khiển PIC 16F887 thì bạn có thể xem các thông tin level ở file thư viện đã trình bày ở chương 4.

Trong thư viện có trình bày các lệnh liên quan đến ngắt, tên các nguồn ngắt nằm gần cuối file để biết và sử dụng cho đúng, phần ứng dụng ngắt sẽ trình bày chi tiết cho các bạn biết rõ hơn và biết cách sử dụng.

Ví dụ 8-3. Lệnh cho phép ngắt timer T1:

```
ENABLE_INTERRUPTS (GLOBAL) ;
ENABLE_INTERRUPTS (INT_TIMER1) ;
```

Bạn phải thực hiện 2 lệnh, một lệnh cho timer T1 và 1 lệnh cho ngắt toàn cục.

8.4.2 LỆNH DISABLE_INTERRUPTS(LEVEL)

Lệnh có chức năng không cho phép ngắt.

Cách sử dụng lệnh này cũng áp dụng các thông số như lệnh cho phép.

8.4.3 VIẾT CHƯƠNG TRÌNH CON PHỤC VỤ NGẮT

Khi sử dụng ngắt nào thì phải viết chương trình con phục vụ ngắt đó và thủ tục viết thông qua ví dụ 8-4 ngắt của timer T1 như sau:

Ví dụ 8-4. Chương trình con phục vụ ngắt của timer T1:

```
#int_timer1
void interrupt_timer1()
{
    X=~X;
    OUTPUT_D (X);
}
```

Thông tin quan trọng nhất là “**#int_timer1**”: bắt đầu bằng dấu “#”, theo sau là tên quy định như trong thư viện đã định nghĩa, mỗi ngắt có 1 cái tên riêng.

Tiếp theo là khai báo “**void interrupt_timer1 ()**” thì tên “**interrupt_timer1**” có thể đặt tùy ý nhưng đặt gần sát với ngắt cho rõ ràng.

Phần còn lại là các lệnh tùy thuộc vào yêu cầu xử lý.

8.5 CÁC ỨNG DỤNG NGẮT CỦA PIC 16F887

Phần này trình bày các ứng dụng ngắt đơn giản của PIC16F887, qua các ứng dụng này giúp bạn biết viết lưu đồ cho những ứng dụng có dùng ngắt, biết viết chương trình có sử dụng ngắt và đặc biệt là biết thêm cách tính toán thời gian delay cho các timer. Từ các kiến thức cơ bản này sẽ giúp bạn hiểu và viết được các ứng dụng khác.

8.5.1 ỨNG DỤNG NGẮT CỦA TIMER T1

Bài 8-1: Dùng vi điều khiển PIC16F887 điều khiển 8 led đơn sáng tắt sử dụng ngắt timer T1 với chu kỳ delay là 210ms.

- Sơ đồ mạch: như hình 8-8.
- Lưu đồ: như hình 8-9.
- Chương trình:

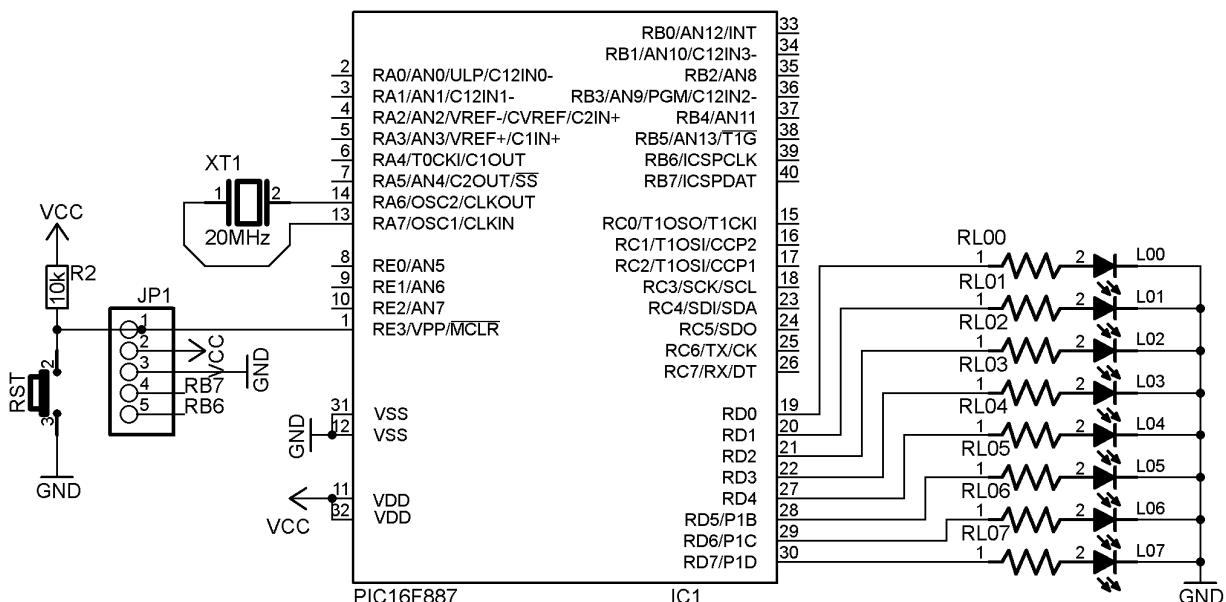
```
#INCLUDE<TV_16F887.C>
UNSIGNED INT8    X;
#int_timer1
void interrupt_timer1()
{
    X=~X;
```

```

    OUTPUT_D(X);
}

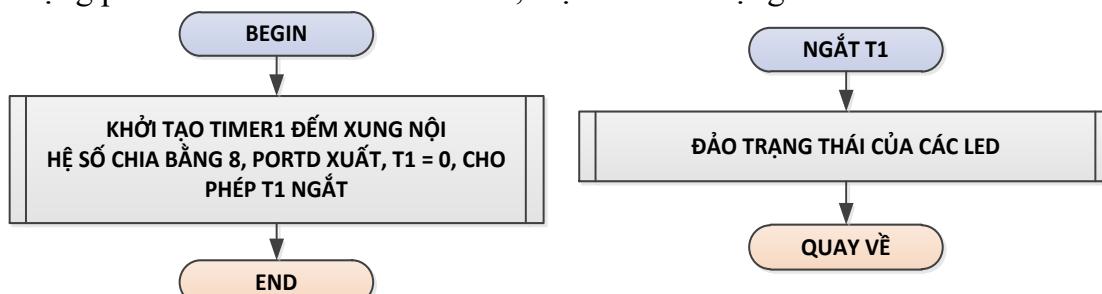
VOID MAIN()
{
    SET_TRIS_D(0x00);
    X=0x00;      OUTPUT_D(X);
    SETUP_TIMER_1(T1_INTERNAL | T1_DIV_BY_8);
    SET_TIMER1(0);
    ENABLE_INTERRUPTS(GLOBAL);
    ENABLE_INTERRUPTS(INT_TIMER1);
    WHILE(TRUE)
    {
    }
}

```



Hình 8-8. Điều khiển 8 led sáng tắt.

Sử dụng portB để điều khiển 8 led đơn, thạch anh sử dụng là 20Mhz.



Hình 8-9. Lưu đồ điều khiển 8 led sáng tắt- định thời 210ms dùng ngắt.

- Giải thích chương trình và tính toán thời gian delay:

Chương trình này về cơ bản giống chương trình đã viết ở chương 6 phần timer. Bài này sử dụng ngắt khi timer đếm bị tràn, chương trình chính thực hiện cho phép ngắt toàn cục và cho phép Timer1 ngắt. Sau khi Timer1 đếm và tràn thì chương trình chính sẽ ngừng và thực hiện chương trình phục vụ ngắt của Timer1 bao gồm 2 lệnh: đảo giá trị của biến X và xuất dữ liệu ra portD và kết thúc để trở lại chương trình chính cho đến khi tràn lần tiếp theo.

Thời gian tính toán giống như bài 6-1, thời gian sáng bằng thời gian tắt là 105ms, chu kỳ 210ms.

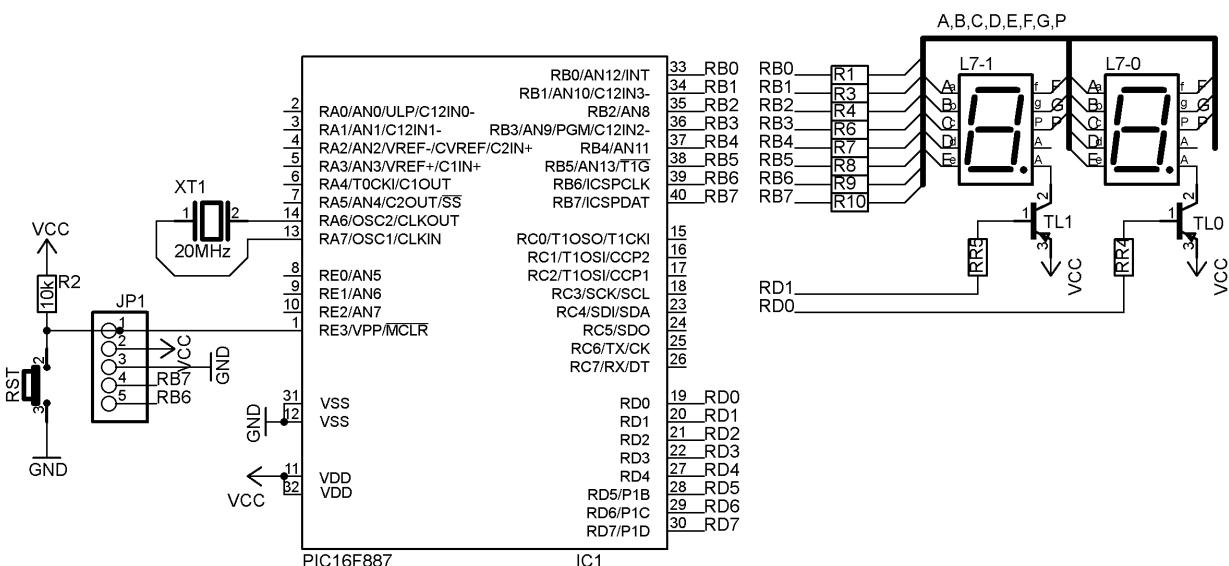
Chương trình chính sau khi thực hiện khởi tạo xong thì không làm gì cả nên ta chưa thấy rõ ưu điểm của ngắt.

Khai báo nhãn của chương trình ngắn cũng như các tên cho trong file thư viện của vi điều khiển, không được đặt tên khác trừ khi bạn định nghĩa lại.

Bài 8-1: Dùng vi điều khiển PIC16F887 điều khiển 16 led đơn: 8 LED sáng tắt dần và 8 led chớp tắt sử dụng ngắn timer T1 với chu kỳ delay là 210ms.

Bài 8-2: Chương trình đếm giây **chính xác** hiển thị trên 2 led 7 đoạn quét dùng vi điều khiển PIC 16F887 dùng Timer và ngắn.

- Sơ đồ mạch:



Hình 8-10. Mạch giao tiếp 2 led 7 đoạn quét hiển thị đếm giây.

- Lưu đồ: như hình 8-11.
- Chương trình:

```
#INCLUDE<TV_16F887.C>
UNSIGNED INT8 GIAY, BDT;
int _timer1
void interrupt_timer1()
{
    SET_TIMER1(3036);
    BDT++;
}

VOID HIENTHI()
{
    OUTPUT_B(MA7DOAN[GIAY %10]);
    DELAY_MS(1);
    OUTPUT_B(MA7DOAN[GIAY/10]);
    DELAY_MS(1);
}

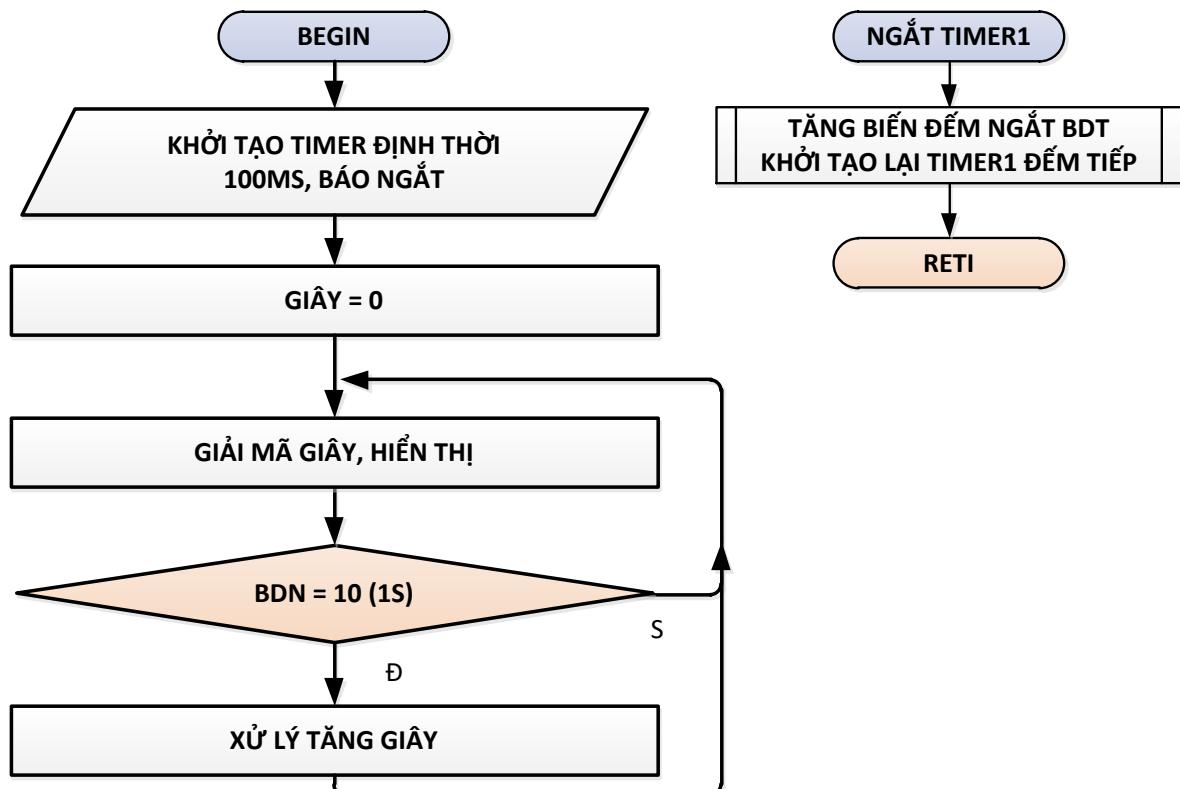
VOID MAIN()
{
    SET_TRIS_B(0x00);    SET_TRIS_D(0x00);
    SETUP_TIMER_1(T1_INTERNAL | T1_DIV_BY_8);
    SET_TIMER1(3036);
    ENABLE_INTERRUPTS(GLOBAL);    ENABLE_INTERRUPTS(INT_TIMER1);
    GIAY = 0;    BDT=0;
    WHILE (TRUE)
    {
}
```

khi timer 1 tràn (đếm xong 1 chu kỳ) -->
xảy ra ngắn
cờ tràn sẽ tự động
xóa
-->
- thiết lập lại giá trị
ban đầu cho timer
- tăng số lần đếm
timer thực hiện

```

IF (BDT >=10)
{
    BDT = 0; ←
    IF (GIAY == 59) GIAY = 0;
    ELSE             GIAY++;
}
ELSE HIENTHI();
}

```



Hình 8-11. Lưu đồ đếm giây dùng Timer định thời báo ngắt.

- Giải thích chương trình:

Chương trình chính thực hiện khởi tạo Timer1 định thời đếm 100ms (xem lại bài 6-2 và 6-3). Gán giá trị GIAY bằng 0, biến đếm tràn BDT bằng 0, cho phép Timer1 ngắt khi tràn còn được gọi là biến đếm ngắt vì mỗi lần tràn thì phát sinh ngắt.

Vòng lặp while thực hiện kiểm tra xem nếu biến đếm tràn nhỏ hơn 10 thì tiến hành gọi hàm hiển thị, thời gian thực hiện chương trình con hiển thị lớn hơn 2ms và nhỏ hơn nhiều so với thời gian định thời 100ms của Timer.

Khi biến đếm tràn BDT bằng 10 hoặc lớn hơn 10 thì tương đương 1 giây nên reset lại biến đếm tràn, kiểm tra và tăng giá trị của giây.

Ở bài này thì chương trình luôn thực hiện công việc thường xuyên là quét 2 led hiển thị, trong khi đó Timer1 vẫn đếm: cả 2 hoạt động song – song cho đến khi Timer tràn thì ngừng chương trình chính để đi thực hiện chương trình con phục vụ ngắt là tăng giá trị biến đếm tràn và khởi tạo lại giá trị bắt đầu chu kỳ mới cho Timer, sau khi làm xong thì trở lại chương trình chính thực hiện tiếp công việc bị gián đoạn lúc ngắt.

Trong chương trình này mạch đếm thời gian chính xác vì timer đếm chính xác và khi ngắt xảy ra thì chỉ thực hiện 2 lệnh cố định nên thời gian luôn là hằng số.

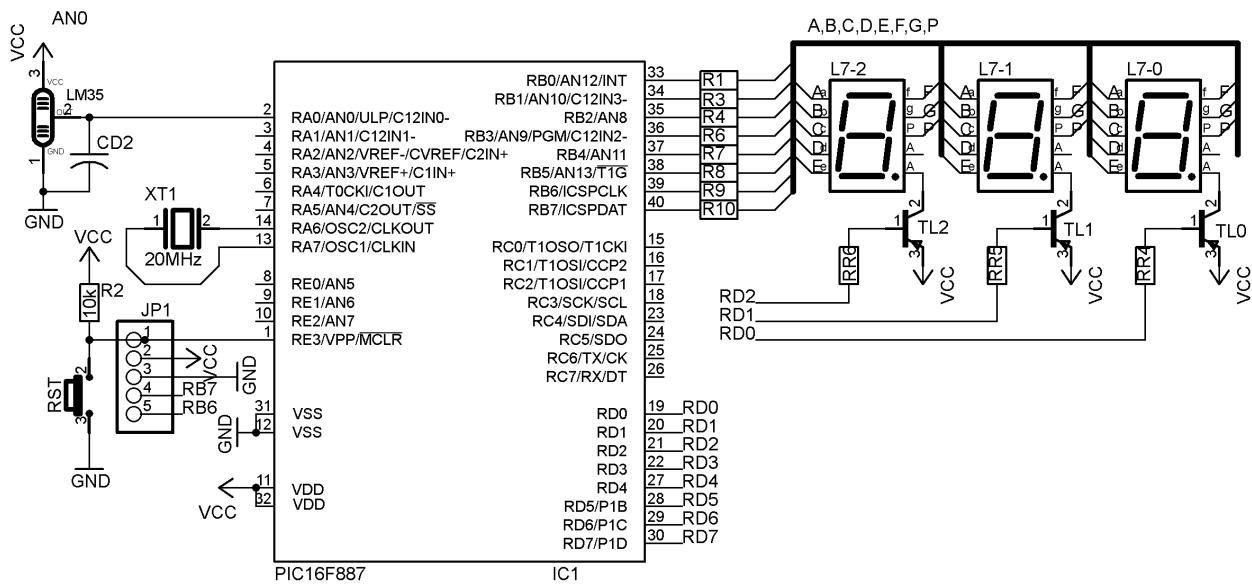
Bài tập 8-1: Hãy vẽ mạch, viết lưu đồ và chương trình mạch đếm phút giây hiển thị trên 4 led 7 đoạn kết nối theo phương pháp quét.

Bài tập 8-2: Hãy vẽ mạch, viết lưu đồ và chương trình mạch để hiển thị giờ phút giây hiển thị trên 6 led 7 đoạn kết nối theo phương pháp quét.

8.5.2 ÚNG DỤNG NGẮT CỦA ADC

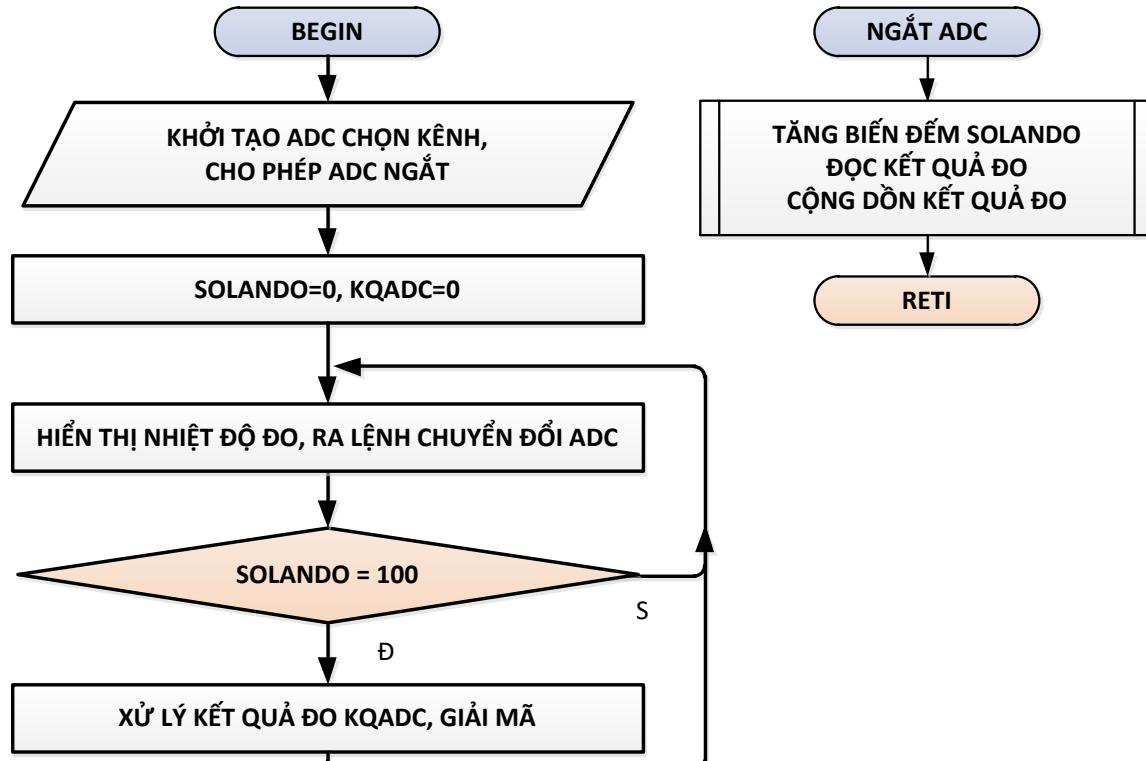
Bài 8-3: Chương trình đo nhiệt độ dùng cảm biến LM35, dùng ADC của vi điều khiển PIC 16F887 báo ngắt sau khi chuyển đổi xong, hiển thị kết quả đo trên 3 led đoạn kết nối theo phương pháp quét, sử dụng điện áp tham chiếu V_{DD} và V_{SS} .

- Sơ đồ mạch: giống như mạch đã trình bày ở chương 7.



Hình 8-12. Mạch đo nhiệt độ dùng cảm biến LM35.

- Lưu đồ: như hình 8-13.



Hình 8-13. Lưu đồ chuyển đổi ADC có báo ngắt.

- Chương trình:

```

#include<TV_16F887.C>
UNSIGNED INT16 KQADC=0;
UNSIGNED INT SOLANDO, MADONVI, MACHUC, MATRAM;
#INT_AD
void interrupt_ADC()
{
    SOLANDO++;
    KQADC=KQADC+READ_ADC(ADC_READ_ONLY);
}

VOID GIAIMA()
{
    MATRAM = MA7DOAN[KQADC/100];
    MACHUC = MA7DOAN[KQADC/10 % 10];
    MADONVI= MA7DOAN[KQADC % 10];
    IF (MATRAM == 0xC0) MATRAM=0xFF;
}

VOID HIENTHI ()
{
    OUTPUT_B(MADONVI);           OUTPUT_LOW(PIN_D0);
    DELAY_MS(1);                 OUTPUT_HIGH(PIN_D0);
    OUTPUT_B(MACHUC);            OUTPUT_LOW(PIN_D1);
    DELAY_MS(1);                 OUTPUT_HIGH(PIN_D1);
    OUTPUT_B(MATRAM);            OUTPUT_LOW(PIN_D2);
    DELAY_MS(1);                 OUTPUT_HIGH(PIN_D2);
}

VOID MAIN()
{
    SET_TRIS_B(0x00);   SET_TRIS_D(0x00);      SET_TRIS_A(0x01);
    SETUP_ADC(ADC_CLOCK_DIV_2);
    SETUP_ADC_PORTS(SAN0);
    SET_ADC_CHANNEL(0);
    ENABLE_INTERRUPTS(GLOBAL);
    SOLANDO=0;
    ENABLE_INTERRUPTS(INT_AD);
    WHILE (TRUE)
    {
        IF (SOLANDO<100)
        {
            HIENTHI();
            READ_ADC(ADC_START_ONLY);
        }
        ELSE
        {
            SOLANDO=0;
            KQADC= KQADC /2.046;
            KQADC=KQADC/100;
            GIAIMA();
            KQADC=0;
        }
    }
}

```

đọc kết quả, khi xảy ra ngắt (đã chuyển đổi xong)

chỉ cho phép chuyển đổi, không có chờ đọc kết quả

- Giải thích chương trình:

Chương trình chính thực hiện khởi tạo ADC (xem lại các của chương 7). Khởi tạo cho phép ADC ngắt.

Vòng lặp while thực hiện kiểm tra xem nếu biến đếm số lần đo nếu nhỏ hơn 100 thì chỉ thực hiện chương trình con hiển thị nhiệt độ đo lần trước và ra lệnh chuyển đổi ADC. Khi số

lần đo bằng 100 thì tiến hành xử lý kết quả đo, giải mã và xoá kết quả đếm về 0 để thực hiện chu kỳ đo trung bình lần tiếp theo.

Lệnh “READ_ADC(ADC_START_ONLY);” có chức năng là ra lệnh chuyển đổi.

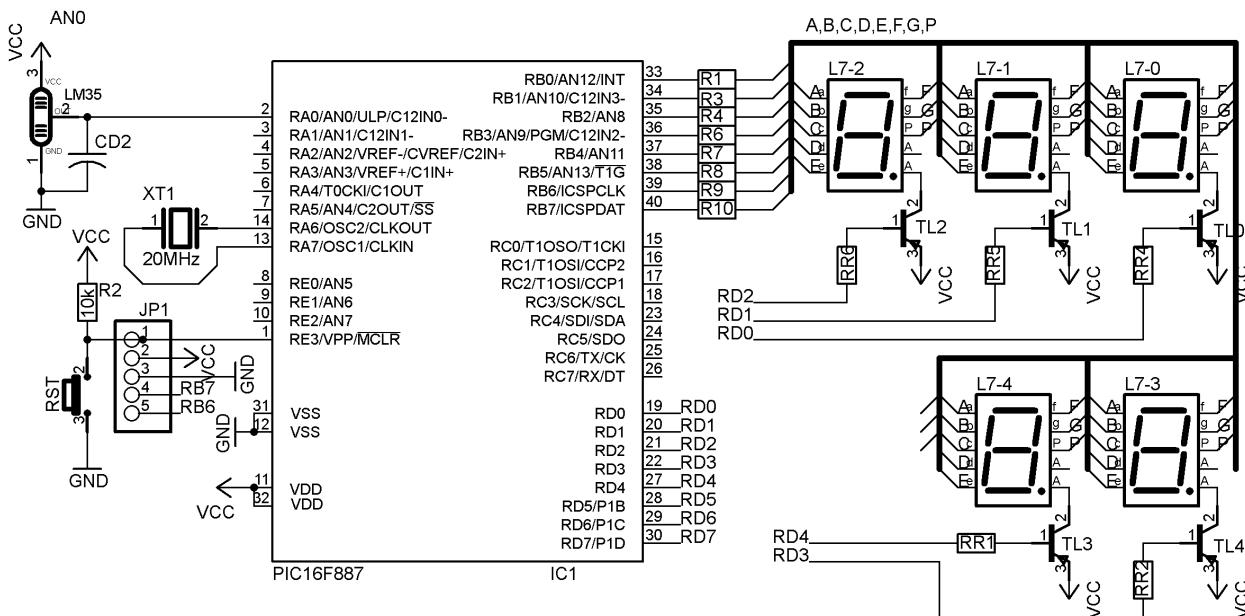
Khi ra lệnh chuyển đổi thì chờ ADC chuyển đổi xong sẽ báo ngắt, chương trình con phục vụ ngắt của ADC sẽ tiến hành tăng biến số lần đo và cộng dồn kết quả đo.

Lệnh “READ_ADC(ADC_READ_ONLY);” có chức năng là đọc kết quả chuyển đổi.

8.5.3 ỨNG DỤNG 2 NGẮT CỦA TIMER VÀ ADC

Bài 8-4: Chương trình đo nhiệt độ và đếm giây. Đếm giây dùng timer định thời báo ngắt hiển thị giây trên led 7 đoạn quét, đo nhiệt độ dùng cảm biến LM35, dùng ADC của vi điều khiển PIC 16F887 báo ngắt sau khi chuyển đổi xong, hiển thị kết quả đo trên 3 led đoạn kết nối theo phương pháp quét, sử dụng điện áp tham chiếu V_{DD} và V_{SS} .

- Sơ đồ mạch: 2 led hiển thị giây và 3 led hiển thị nhiệt độ nên tổng số led là 5.



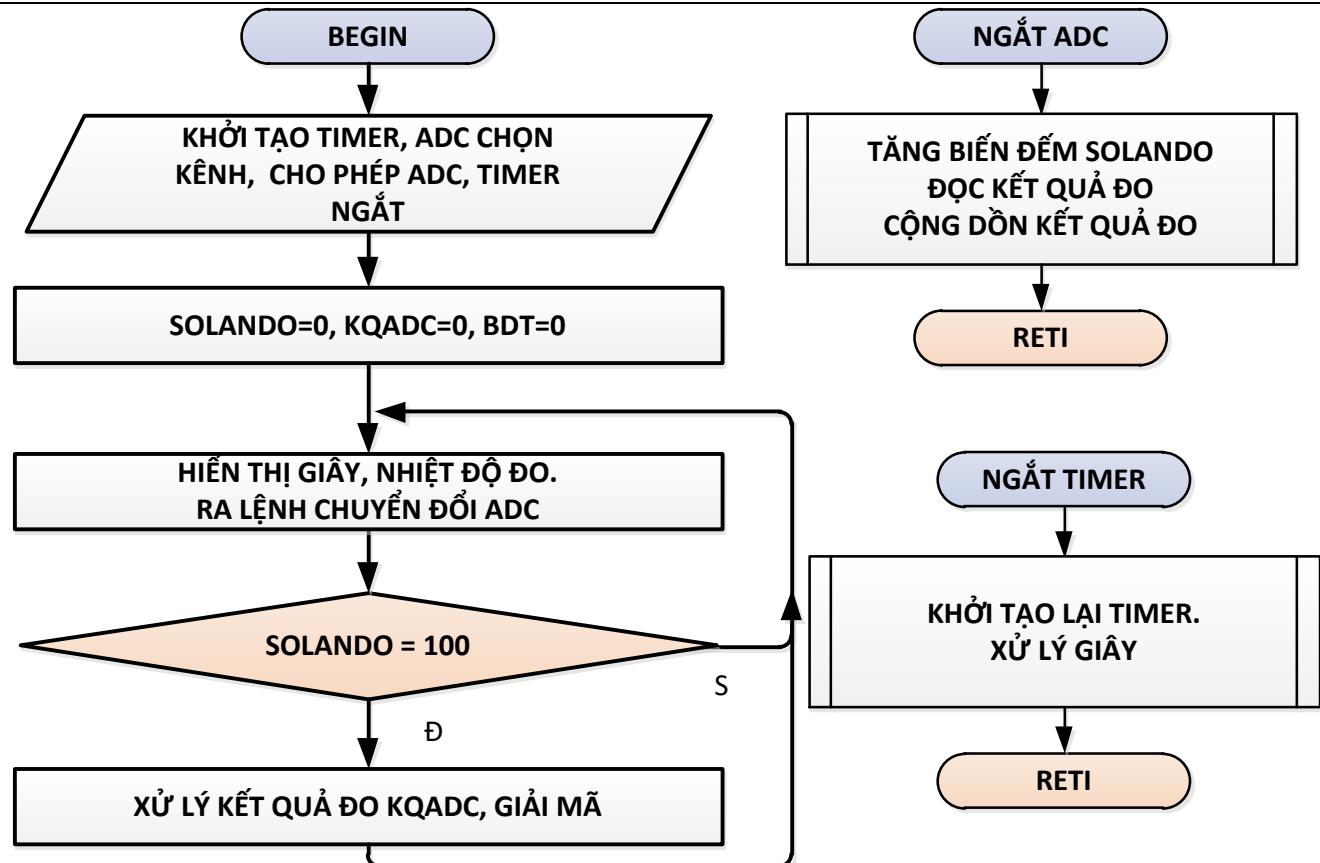
Hình 8-14. Đo nhiệt độ dùng cảm biến LM35 và đếm giây.

- Lưu đồ: như hình 8-15.
 - Giải thích lưu đồ:

Hệ thống thực hiện 2 chức năng đếm giây và đo nhiệt độ. Cả 2 đều sử dụng ngắn, bài này tổng hợp của bài 8-2 và 8-3 để cho thấy có thể một ứng dụng có thể sử dụng nhiều ngắn.

Khi timer đếm tràn thì tiến hành xử lý ngắt của timer thực hiện việc xử lý giây.

Khi ADC chuyển đổi xong thì tiến hành công dồn kết quả ADC.



Hình 8-15. Lưu đồ chuyển đổi ADC có báo ngắt và đếm giây dùng timer báo ngắt.

- Chương trình:

```

#include<TV_16F887.C>
UNSIGNED INT16      KQADC=0;
UNSIGNED INT          GIAY,BDT,SOLANDO, MADONVI, MACHUC, MATRAM;
#int_timer1
void interrupt_timer1()
{
    SET_TIMER1(3036);
    BDT++;
    IF(BDT>=10)
    {
        BDT = 0;
        IF(GIAY == 59) GIAY = 0;
        ELSE           GIAY++;
    }
}
#INT_AD
void interrupt_ADC()
{
    SOLANDO++;
    KQADC=KQADC+READ_ADC(ADC_READ_ONLY);
}
VOID GIAIMA()
{
    MATRAM = MA7DOAN[KQADC/100];
    MACHUC = MA7DOAN[KQADC/10 % 10];
    MADONVI= MA7DOAN[KQADC % 10];
    IF (MATRAM == 0xC0) MATRAM=0xFF;
}
    
```

```

VOID HIENTHI ()
{
    OUTPUT_B(MADONVI);
    DELAY_MS(1);
    OUTPUT_B(MACHUC);
    DELAY_MS(1);
    OUTPUT_B(MATRAM);
    DELAY_MS(1);

    OUTPUT_B(MA7DOAN[GIAY %10]);
    DELAY_MS(1);
    OUTPUT_B(MA7DOAN[GIAY/10]);
    DELAY_MS(1);
}

VOID MAIN()
{
    SET_TRIS_B(0x00);      SET_TRIS_D(0x00);          SET_TRIS_A(0x01);
    SETUP_ADC(ADC_CLOCK_DIV_2);
    SETUP_ADC_PORTS(SAN0);
    SET_ADC_CHANNEL(0);

    SETUP_TIMER_1(T1_INTERNAL | T1_DIV_BY_8);
    SET_TIMER1(0);

    ENABLE_INTERRUPTS(GLOBAL);
    ENABLE_INTERRUPTS(INT_AD);
    ENABLE_INTERRUPTS(INT_TIMER1);

    SOLANDO=0;      GIAY = 0;      BDT=0;
    WHILE (TRUE)
    {
        IF (SOLANDO<100)
        {
            HIENTHI();
            READ_ADC(ADC_START_ONLY);
        }
        ELSE
        {
            SOLANDO=0;
            KQADC= KQADC /2.046;
            KQADC=KQADC/100;
            GIAIMA();
            KQADC=0;
        }
    }
}

```

- Giải thích chương trình:

Chương trình chính thực hiện khởi tạo ADC giống như bài 8-3 và khởi tạo Timer1 giống như bài 8-2.

Vòng lặp while thực hiện các lệnh giống bài 8-3.

Chương trình con hiển thị có chức năng hiển thị nhiệt độ và giây.

Khi timer đếm tràn thì phát sinh ngắt và chương trình phục vụ ngắt thực hiện tăng biến đếm tràn, kiểm tra nếu đúng 1 giây thì tiến hành xử lý giây.

Phản ADC giống như đã giải thích ở bài 8-3.

Trong chương trình phục vụ ngắt không phải là hằng số cố định vì các lệnh kiểm tra biến đếm tràn sẽ có 2 trường hợp: không thoả thì thoát – thời gian thực hiện ngắn hơn, khi thoả điều kiện thì thực hiện nhiều lệnh hơn và thời gian dài hơn, hiện tượng này trong 1 số chương trình sẽ gây ra sai sót nhưng ở bài này do timer đếm độc lập và báo ngắt nên không gây ảnh hưởng vì ta khởi tạo lại timer ngay từ lúc bắt đầu chương trình phục vụ ngắt nên timer sẽ tự động đếm chu kỳ tiếp theo và không phụ thuộc vào các lệnh còn lại của chương trình con phục vụ ngắt.

Bài tập 8-3: Hãy vẽ mạch, viết lưu đồ và chương trình mạch để giờ phút giây hiển thị trên 6 led 7 đoạn kết nối theo phương pháp quét và đo nhiệt độ hiển thị trên 3 led 7 đoạn. Cá 2 đều dùng ngắt.

Bài tập 8-4: Giống bài 8-3 nhưng ADC không dùng ngắt.

Bài tập 8-5: Một vi điều khiển PIC16F887 giao tiếp với 3 led 7 đoạn loại anode chung, 2 cảm biến nhiệt độ LM35 nối với kênh AN0, AN1.

Hãy viết lưu đồ và chương trình chuyển đổi nhiệt độ dùng ngắt của ADC để biết quá trình chuyển đổi xong, mỗi kênh chuyển đổi trong thời gian 5 giây, dùng timer1 đếm thời gian và báo ngắt để chuyển kênh.

Bài tập 8-6: Một vi điều khiển PIC16F887 giao tiếp với 4 led 7 đoạn loại anode chung, 1 led hiển thị số thứ tự kênh, 3 led còn lại hiển thị nhiệt độ, 4 cảm biến nhiệt độ LM35 nối với 4 kênh từ AN0 đến AN3, có 3 nút nhấn: 1 nút chọn chế độ tự động hay bằng tay, 2 nút còn lại là UP và DW, nhấn UP là tăng lên để chọn kênh ADC cao hơn, DW thì giảm.

Hãy viết lưu đồ và chương trình chuyển đổi nhiệt độ ở chế độ tự động dùng ngắt của ADC để biết quá trình chuyển đổi xong, mỗi kênh chuyển đổi trong thời gian 5 giây, dùng timer1 đếm thời gian và báo ngắt để chuyển kênh.

Ở chế độ bằng tay thì chuyển kênh khi nhấn UP hoặc DW.

8.6 CÂU HỎI ÔN TẬP – TRẮC NGHIỆM

8.6.1 CÂU HỎI ÔN TẬP

Câu 8-1: Hãy cho biết tên các thanh ghi liên quan đến ngắt của vi điều khiển PIC16F887.

Câu 8-2: Hãy cho biết chức năng các bit trong các thanh ghi ngắn của vi điều khiển PIC16F887.

Câu 8-3: Hãy cho biết vector địa chỉ ngắt của vi điều khiển PIC16F887.

Câu 8-4: Hãy cho biết cách xử lý ưu tiên ngắn của vi điều khiển PIC16F887.

8.6.2 CÂU HỎI MỞ RỘNG

Câu 8-5: Hãy so sánh ngắt của vi điều khiển AT89S52 với PIC16F887.

Câu 8-6: Hãy so sánh ngắt của vi điều khiển PIC18F4550 với PIC16F887.

8.6.3 CÂU HỎI TRẮC NGHIỆM

Câu 8-1: Vì điều khiển PIC 16F887 có mấy nguồn ngắn:

Câu 8-2: Vì điều khiển PIC 16F887 có mảng thanh ghi liên quan đến ngắn:

- (a) 3 (b) 4 (c) 5 (d) 6

Câu 8-3: Thanh ghi OPTION_REG chưa mấy nguồn ngắn:

- (a) 3 (b) 4 (c) 5 (d) 6

Câu 8-4: Ngắt nào của PIC 16F887 được xem là ngắn trong:

- (a) Timer0 (b) Timer1 (c) Timer2 (d) ADC

Câu 8-5: Ngắt nào của PIC 16F887 được xem là ngắn trong:

- (a) RCIE (b) Timer1 (c) Timer2 (d) INT

Câu 8-16: Bit nào của PIC 16F887 là bit cho phép ngắn ngoại vi:

- (a) PEIE (b) GIE (c) T0IE (d) ADIE

Câu 8-7: Ngắt nào của PIC 16F887 có mức ưu tiên cao nhất:

- (a) ADC (b) Timer0 (c) INT (d) Không có

Câu 8-8: Bit nào của PIC 16F887 là bit cho phép/cấm ngắn toàn bộ:

- (a) PEIE (b) GIE (c) T0IE (d) ADIE

Câu 8-9: Mạch điện xử lý ngắn của PIC16F887 gồm các cổng:

- | | |
|-----------------|------------------|
| (a) AND, NOR | (b) AND |
| (b) (c) AND, OR | (d) AND, OR, NOT |

Câu 8-10: Vector địa ngắn của PIC16F887 là:

- (a) 0000H (b) 0003H (c) 000BH (d) 0004H

Câu 8-11: Ngắt ngoài thứ 1 của AT89S52 có số thứ tự trong lập trình C là:

- (a) Thứ 0 (b) Thứ 1 (c) Thứ 2 (d) Thứ 3

Câu 8-12: Ngắt truyền dữ liệu của AT89S52 có số thứ tự trong lập trình C là:

- (a) Thứ 5 (b) Thứ 6 (c) Thứ 4 (d) Thứ 3

8.6.4 BÀI TẬP

Bài tập 8-1: Mạch đo nhiệt độ 2 kênh dùng cảm biến LM35, vi điều khiển PIC16F887 và 3 led 7 đoạn.

Hãy vẽ mạch, viết lưu đồ và chương trình đo nhiệt độ 2 kênh lần lượt hiển thị trên 3 led, mỗi kênh đo trong khoảng thời gian 1s, dùng Timer T1 để định thời 1 giây chính xác sử dụng ngắn để chuyển kênh.

Bài tập 8-2: Mạch đồng số dùng vi điều khiển PIC16F887 và LCD 16x2.

Hãy vẽ mạch, viết lưu đồ và chương trình đếm giờ phút giây hiển thị trên LCD hàng 1, dùng Timer T1 để định thời 1 giây chính xác sử dụng ngắn.

Bài tập 8-3: Mạch đồng số, đếm sản phẩm dùng vi điều khiển PIC16F887 và LCD 16x2.

Hãy vẽ mạch, viết lưu đồ và chương trình đếm giờ phút giây hiển thị trên LCD ở hàng 1, dùng Timer T1 để định thời 1 giây chính xác sử dụng ngắn, đếm sản phẩm dùng counter T0 hiển thị trên LCD ở hàng 2.

VỊ ĐIỀU KHIỂN PIC16F887:

ĐIỀU CHẾ ĐỘ RỘNG XUNG - PWM

9.1 GIỚI THIỆU

Vì điều khiển PIC họ 16F887 có 2 bộ PWM (Pulse Width Modulation) cơ bản dùng để điều khiển tốc độ động cơ DC.

Phần này sẽ khảo sát chi tiết khói PWM của PIC và tập lệnh lập trình C cho PWM, các ứng dụng dùng PWM.

9.2 KHẢO SÁT PWM

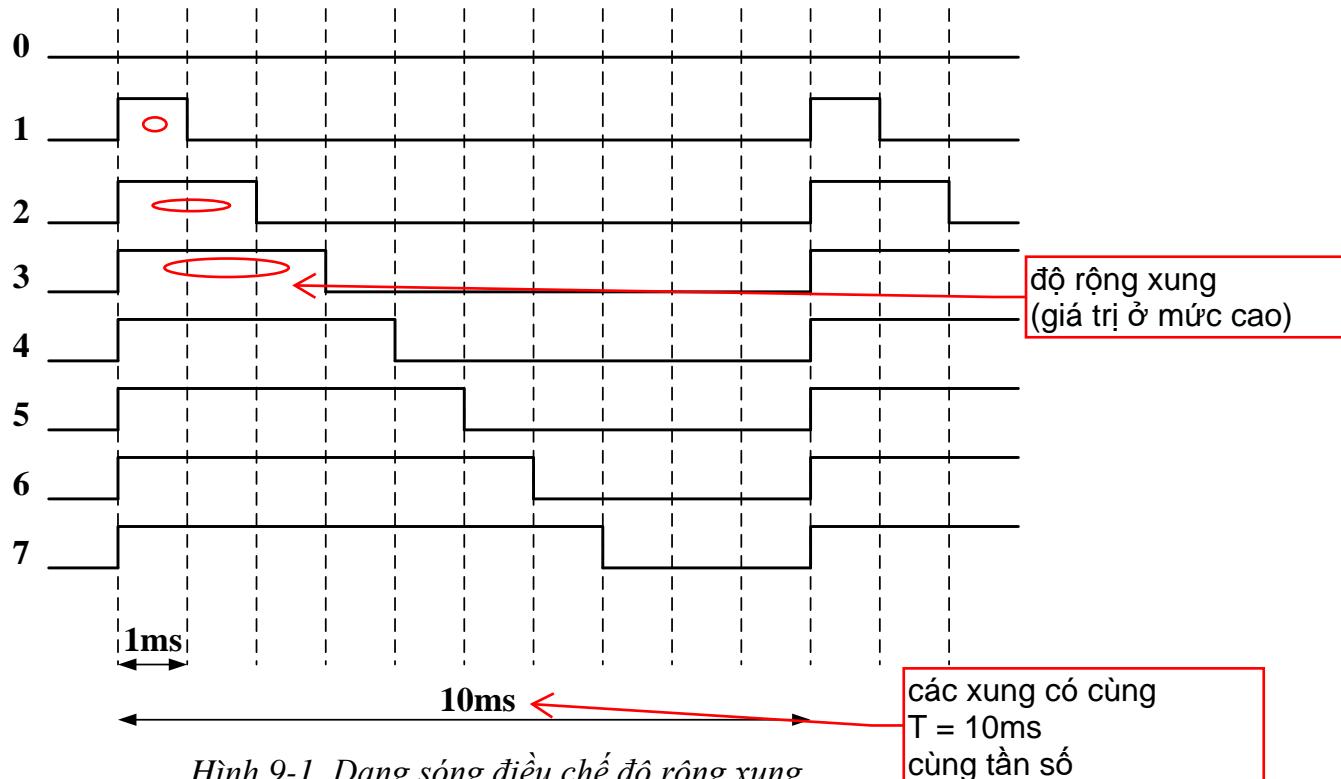
9.2.1 NGUYÊN LÝ ĐIỀU CHẾ ĐỘ RỘNG XUNG PWM

Nguyên lý điều chế độ rộng xung là mạch tạo ra xung vuông có chu kỳ là hằng số nhưng hệ số công tác (còn gọi là hệ số chu kỳ - duty cycle) có thể thay đổi được. Sự thay đổi của hệ số chu kỳ làm thay đổi điện áp hoặc dòng trung bình hoặc dòng điện trung bình.

Sự thay đổi điện áp hoặc dòng trung bình dùng để điều khiển các tải như động cơ DC thì làm thay đổi tốc độ động cơ, điều khiển bóng đèn thì làm thay đổi cường độ sáng của bóng đèn,

...

Các dạng sóng điều chế độ rộng xung với các **hệ số chu kỳ khác nhau** như hình 9-1.



Hình 9-1. Dạng sóng điều chế độ rộng xung.

Cho chu kỳ 10ms, ở cấp tốc độ 0 thì tín hiệu bằng 0. Điện áp hay dòng trung bình sẽ bằng 0, nếu tín hiệu này điều khiển đèn led thì đèn led sẽ tắt.

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 $(10*1)/10 = 1\text{mA}$. Led sáng mờ với dòng là 1mA.

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 $(10*2)/10 = 2\text{mA}$. Led sáng hơn với dòng là 2mA.

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 $(10*5)/10 = 5\text{mA}$.

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. Led sáng cực đại với dòng là 10mA.

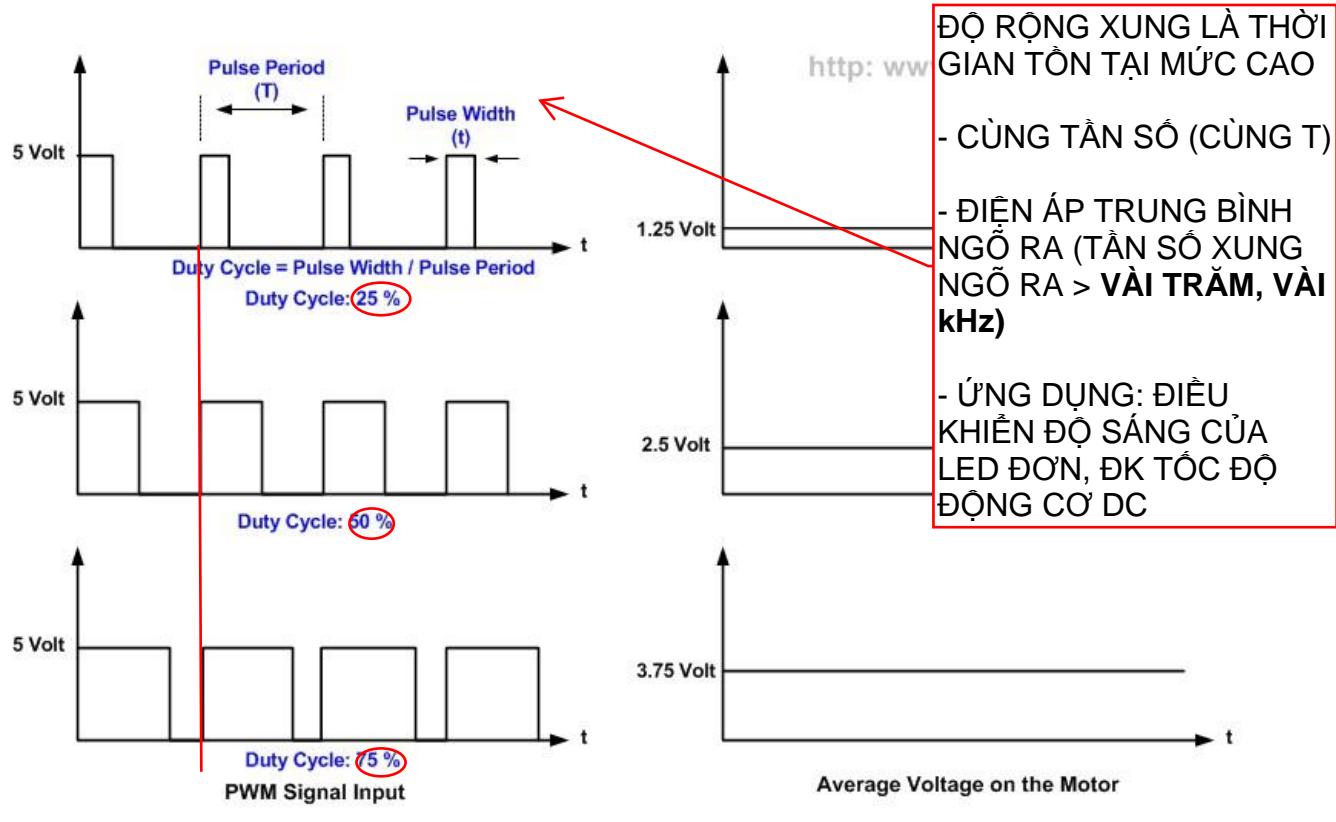
Khi thay đổi hệ số chu kỳ thì chỉ thay đổi thời gian xung ở mức 1, còn chu kỳ thì không đổi.

Có thể tính được điện áp trung bình thay cho dòng trung bình như hình 9-2 trong 3 trường hợp cấp độ 25%, 50% và 75%.

Ở cấp độ 25% thì điện áp trung bình của 5V thì được 1.25V.

Ở cấp độ 50% thì điện áp trung bình của 5V thì được 2.5V.

Ở cấp độ 75% thì điện áp trung bình của 5V thì được 3.75V.

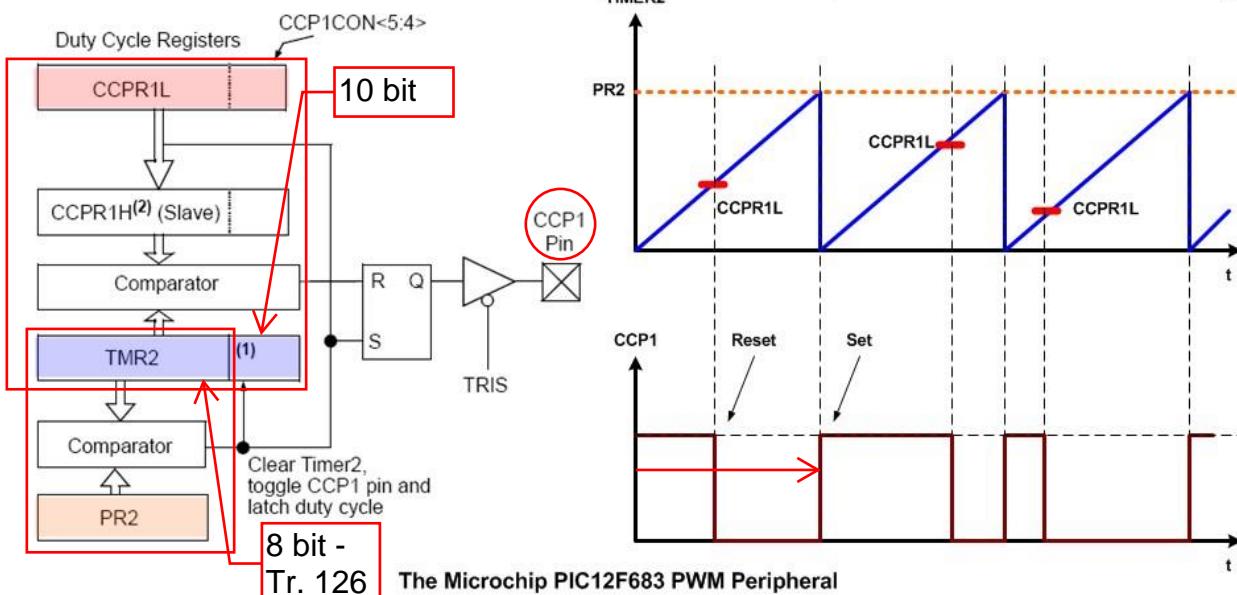


Hình 9-2. Dạng sóng điều chế độ rộng xung và điện áp trung bình 3 cấp độ.

Qua phần giới thiệu này đã cho chúng ta biết rõ khái niệm điều chế độ rộng xung, phần tiếp theo chúng ta sẽ khảo sát mạch điện có chức năng tạo ra dạng sóng PWM và các phương trình tính toán để tạo ra chu kỳ tín hiệu và hệ số chu kỳ.

9.2.2 CẤU TRÚC KHÓI ĐIỀU CHẾ ĐỘ RỘNG XUNG PWM

Vì điều khiển PIC16F887 có 2 khói PWM cơ bản là CCP1 và CCP2 và PWM nâng cao, sơ đồ khói PWM của khói CCP1 có cấu trúc như hình 9-3.

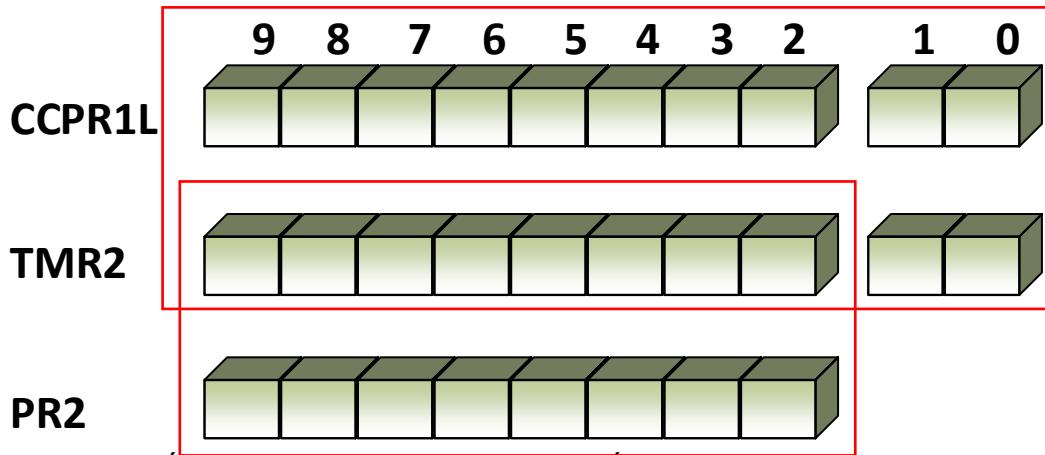


Hình 9-3. Sơ đồ khối PWM của khối CCP1 của PIC16F887.

Giải thích nguyên lý PWM

Khối PWM gồm có 2 mạch so sánh: mạch so sánh 2 dữ liệu 8 bit nằm bên dưới và mạch so sánh 2 dữ liệu 10 bit nằm bên trên.

Trước khi tiến hành so sánh các thanh ghi chúng ta cần biết số bit của các thanh ghi dùng để so sánh được minh họa như hình 9-4.



Hình 9-4. Số bit của các thanh ghi trong khối PWM của PWM PIC16F887.

- Thanh ghi PR2 là thanh ghi 8 bit.
- Thanh ghi TMR2 là thanh ghi 8 bit kết hợp với hằng số chia 4 của bộ dao động tương đương 2 bit nên thành 10 bit.
- Thanh ghi CCPR1L 8 bit cùng với 2 bit thấp CCP1CON<5:4> hoặc với tên khác là <DC1B1:DC1B0> thành 10 bit.

Chức năng của timer T2 dùng để đếm xung từ bộ dao động nội chuyên phục vụ cho khối CCPx. Giới hạn đếm của timer T2 bắt đầu từ 10 bit bằng 0000_0000_00B cho đến khi 8 bit cao lưu trong thanh ghi TMR2 bằng với 8 bit của thanh ghi PR2 thì timer T2 bị reset về 0 và bắt đầu chu kỳ mới và cứ thế lặp đi lặp lại.

Phần này chưa tính toán chi tiết nhưng luôn thỏa 1 điều kiện là giá trị nạp cho thanh ghi PR2 luôn lớn hơn giá trị của thanh ghi CCPR1L.

Cho ngõ ra Q của FF RS ở mức 1, time T2 bắt đầu đếm từ 0.

Khi giá trị đếm của Timer2 10 bit bằng với giá trị 10 bit của CCP1L:CCP1CON<5:4> thì bộ so sánh 10 bit sẽ kích ngõ vào reset flip flop RS làm ngõ ra Q về 0, nếu bộ đếm 3 trạng thái được phép thì ngõ ra CCP1 về 0. Bộ so sánh 10 bit này xảy ra trước vì giới hạn so sánh của hai thanh ghi nhỏ hơn.

Timer T2 tiếp tục đếm tăng giá trị và bộ so sánh 8 bit sẽ tiến hành so sánh 8 bit cao của TMR2 với 8 bit PR2, do nó so sánh 8 bit cao nên điều kiện so sánh bằng của nó xảy ra sau bộ so sánh 10 bit.

Khi giá trị TMR2 bằng giá trị của thanh ghi PR2 xảy ra thì mạch so sánh sẽ kích ngõ vào S của flip flop RS làm ngõ ra Q lên mức 1 và nếu bộ đếm 3 trạng thái được phép thì ngõ ra CCP1 sẽ lên mức 1.

Đồng thời kích mạch nạp giá trị 10 bit từ thanh ghi CCP1L sang thanh ghi CCP1H. Nếu hệ số chu kỳ thay đổi thì giá trị mới này sẽ được cập nhật vào chu kỳ tiếp theo, nếu không thay đổi thì giữ nguyên hệ số chu kỳ.

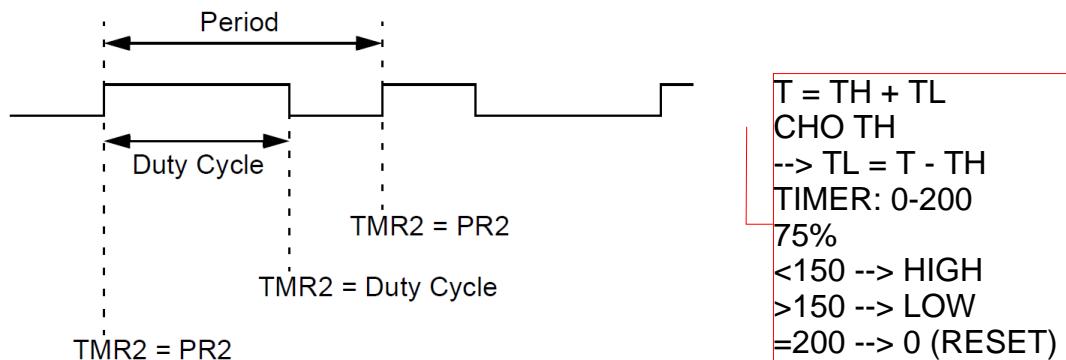
Timer T2 reset về 0 để bắt đầu lại 1 chu kỳ mới. Quá trình này lặp lại liên tục để tạo ra dạng sóng PWM liên tục, dạng sóng như trong hình 9-3.

Chú ý 1: Để xuất tín hiệu Q của flip flop ra chân RC2/CCP1 thì phải khởi tạo RC2 là port xuất.

Chú ý 2: PIC 16F887 có 2 khối PWM nên các tên như CCPx, thanh ghi CCPRxL, CCPRxH với x có thể là 1 hoặc 2.

9.2.3 TÍNH CHU KỲ XUNG PWM

Dạng sóng điều chế PWM như hình 9-5.



Hình 9-5. Dạng sóng PWM.

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.

Chu kỳ PWM của PIC16F887 được tính theo công thức:

$$PERIOD_{PWM} = [(PR2) + 1] * 4 * T_{OSC} * PV_{TMR2} \quad (9-1)$$

Trong đó: T_{OSC} là chu kỳ của tụ thạch anh tạo dao động.

PV_{TMR2} (Prescale Value) giá trị chia trước của timer2.

Khi giá trị của timer 2 (TMR2) bằng giá trị của thanh ghi PR2 thì 3 sự kiện sau sẽ xảy ra:

- Thanh ghi TMR2 bị xóa.
- Tín hiệu ngõ ra CCPx lên mức 1, ngoại trừ hệ số chu kỳ bằng 0% thì CCPx vẫn ở mức 0.
- Hệ số chu kỳ PWM được chuyển từ thanh ghi CCPRxL sang thanh ghi CCPRxH.

9.2.4 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 PIC16F887 được tính theo công thức:

$$\text{DUTY_CYCLE}_{\text{PWM}} = (\text{CCPRxL} : \text{CCPxCON } < 5 : 4 >) * T_{\text{OSC}} * PV_{\text{TMR2}} \quad (9-2)$$

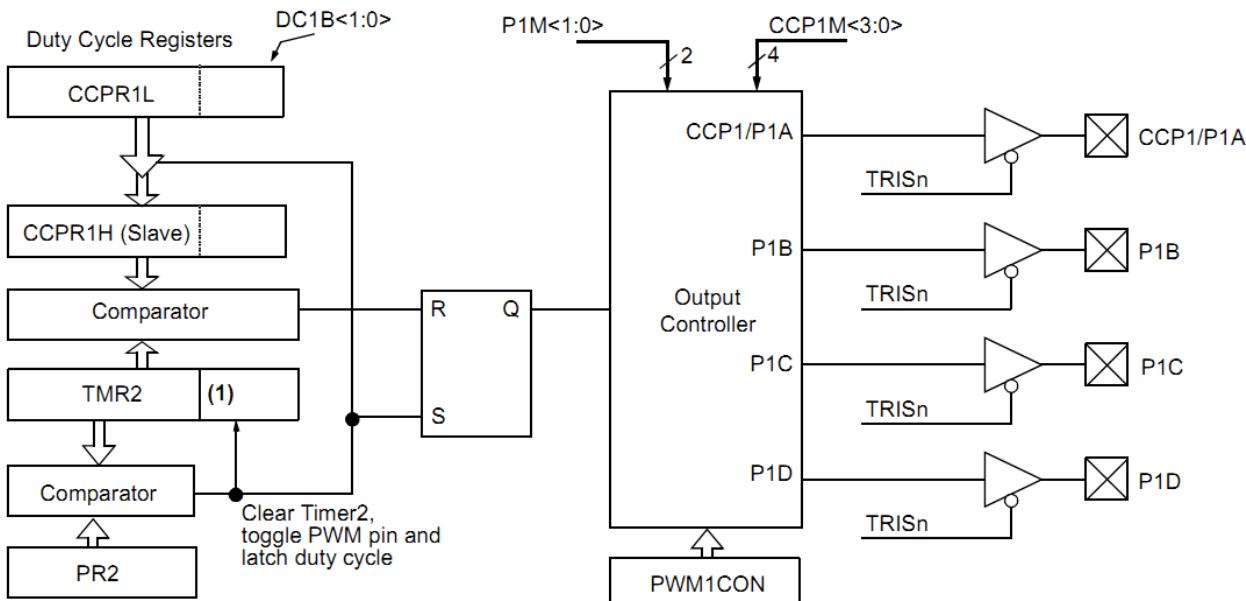
Chú ý: kí hiệu <5:4> cho biết vị trí bit thứ 5 và thứ 4.

9.3 KHẢO SÁT PWM NÂNG CAO

9.3.1 CẤU TRÚC KHỐI PWM NÂNG CAO

Khối PWM còn có chế độ hoạt động nâng cao có thể có đến 4 ngõ ra PWM với độ phân giải 10 bit và hoạt động ở nhiều chế độ khác nhau gồm:

PWM đơn, PWM nuga cầu H, PWM cầu H đầy đủ, chạy thuận, PWM cầu H đầy đủ, chạy nghịch



Note 1: The 8-bit timer TMR2 register is concatenated with the 2-bit internal Q clock, or 2 bits of the prescaler to create the 10-bit time base.

Hình 9-6. Sơ đồ khối PWM nâng cao.

Để lựa chọn 1 trong 4 chế độ này phụ thuộc vào 2 bit có tên P1M nằm trong thanh ghi CCP1CON. Khối PWM nâng cao có 4 ngõ ra có tên là P1A, P1B, P1C và P1D. Cực tính các chân ngõ ra PWM được cấu hình bởi 4 bit CCP1M nằm trong thanh ghi CCP1CON.

Cấu hình PWM nâng cao như hình 9-6.

9.3.2 CÁC CHẾ ĐỘ HOẠT ĐỘNG KHỐI PWM NÂNG CAO

Khối PWM nâng cao có 4 chế độ hoạt động như đã trình bày, phần này trình bày các chế độ hoạt động của khối PMW nâng cao. Các bit để chọn chế độ hoạt động như bảng sau:

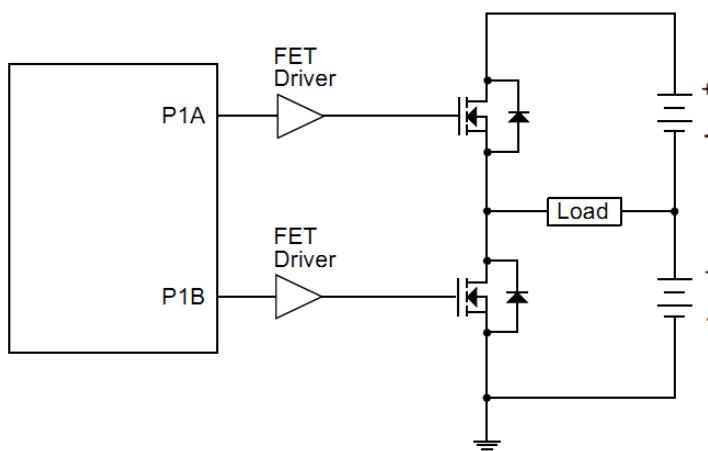
Bảng 9-1. Các chế độ hoạt động của PWM nâng cao.

ECCP Mode	P1M<1:0>	CCP1/P1A	P1B	P1C	P1D
Single	00	Yes ⁽¹⁾	Yes ⁽¹⁾	Yes ⁽¹⁾	Yes ⁽¹⁾
Half-Bridge	10	Yes	Yes	No	No
Full-Bridge, Forward	01	Yes	Yes	Yes	Yes
Full-Bridge, Reverse	11	Yes	Yes	Yes	Yes

Note 1: Pulse Steering enables outputs in Single mode.

Chế độ đơn thì giống PWM như đã khảo sát, chỉ điều khiển tải 1 chiều.

Chế độ hoạt động nǔa cầu có cấu trúc như hình 9-7.

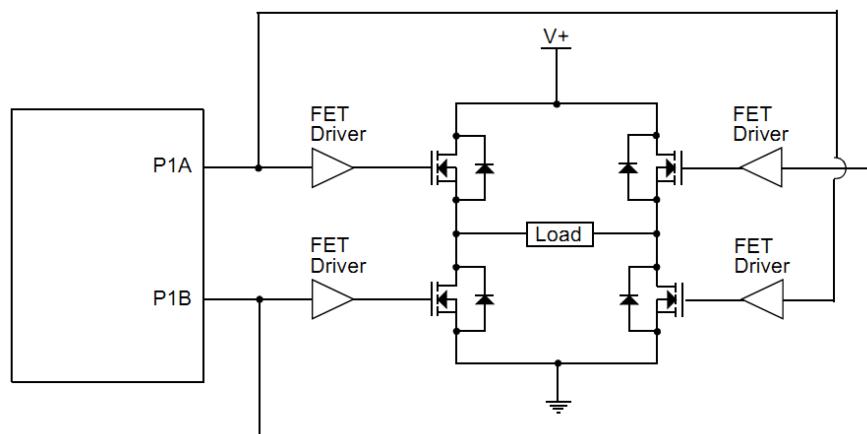


Hình 9-7. Chế độ hoạt động nǔa cầu.

Trong chế độ này thì nếu xung PWM xuất hiện ở ngõ ra P1A thì ngõ ra P1B phải ở mức 0 làm transistor tắt. Khi đó tải được điều khiển với dòng điện chạy từ cực dương của nguồn bên trên, qua transistor trên, qua tải và về cực âm của nguồn.

Ngược lại nếu xung PWM xuất hiện ở ngõ ra P1B thì ngõ ra P1A phải ở mức 0 làm transistor tắt. Khi đó tải được điều khiển với dòng điện chạy từ cực dương của nguồn bên dưới, qua tải, qua transistor dưới, về cực âm của nguồn. Dòng điện ngược chiều ở trên, nếu tải là động cơ thì có thể điều khiển được động cơ quay 2 chiều.

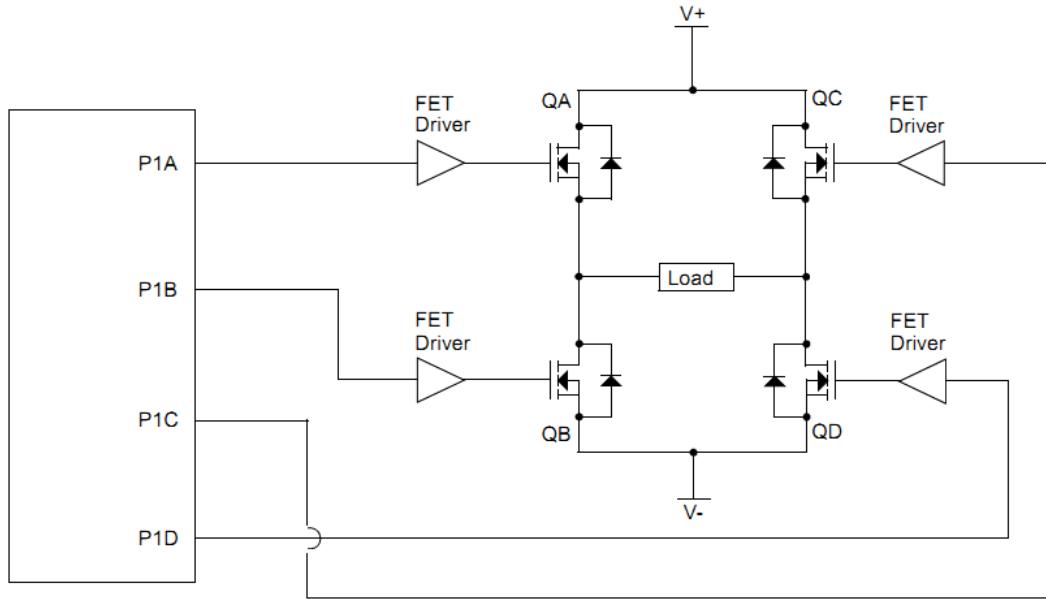
Chế độ hoạt động ngõ ra nǔa cầu điều khiển mạch cầu H đầy đủ như hình 9-8.



Hình 9-8. Chế độ hoạt động nǔa cầu, điều khiển mạch cầu H đầy đủ.

Nguyên lý hoạt động của mạch này giống như trên nhưng chỉ cần dùng 1 nguồn cung cấp nhưng cần 4 transistor.

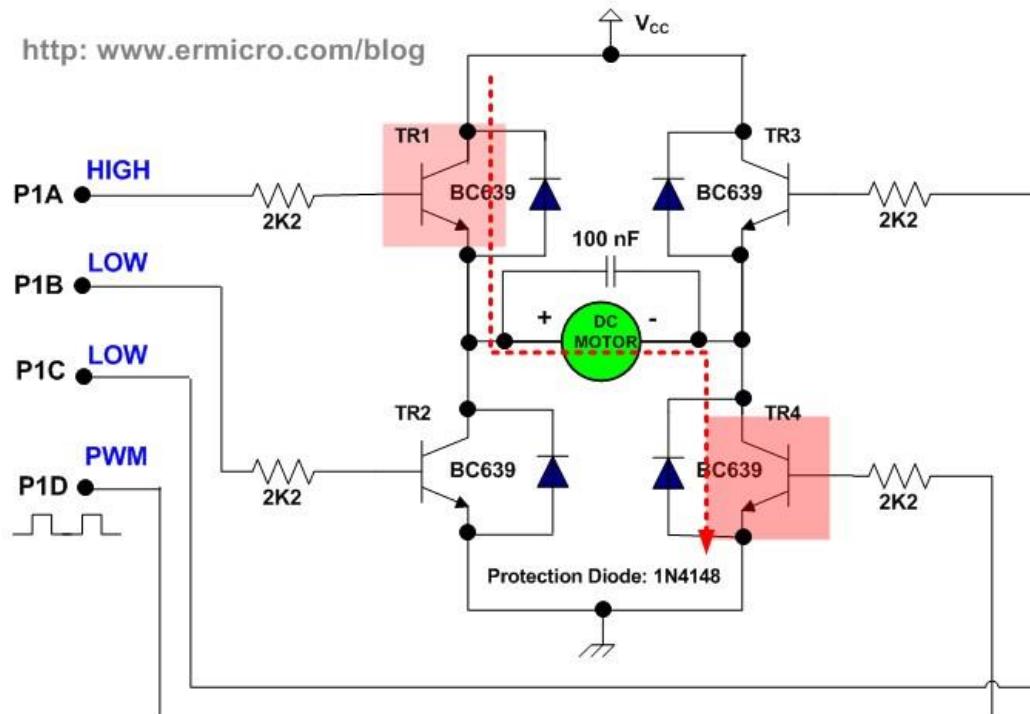
Chế độ hoạt động ngõ ra cầu đầy đủ điều khiển mạch cầu H như hình 9-9.



Hình 9-9. Chế độ hoạt động cầu đầy đủ điều khiển mạch cầu H.

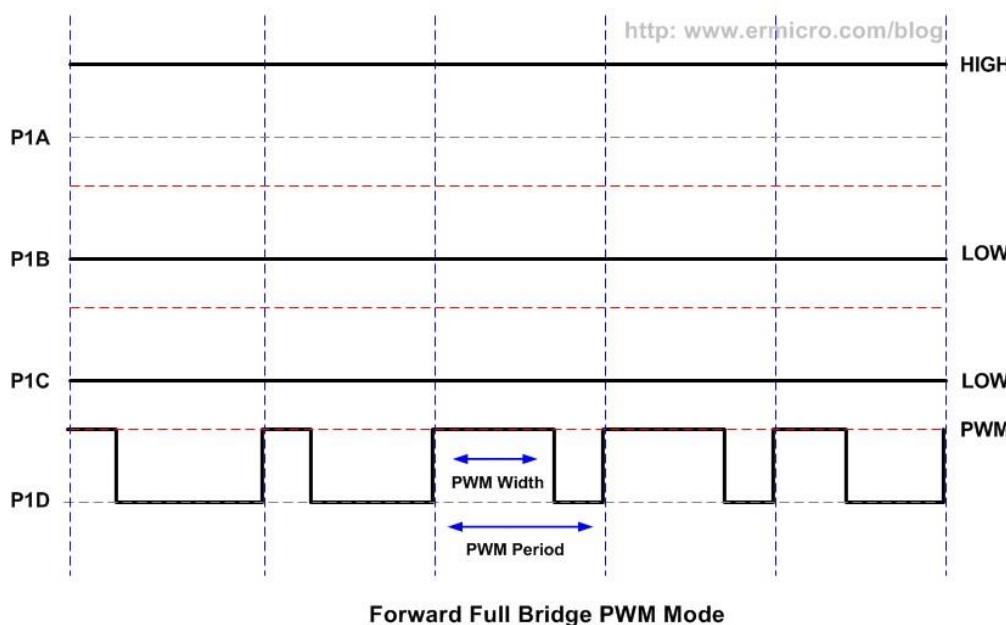
Nguyên lý hoạt động của mạch này cho phép điều khiển PWM cho từng ngõ ra.

Ví dụ muốn điều khiển tải là motor quay thuận thì sơ đồ mạch và dạng sóng như hình 9-10, 9-11.



Forward Full Bridge PWM Mode H-Bridge Circuit

Hình 9-10. Chế độ hoạt động cầu đầy đủ điều khiển motor mạch cầu H, quay thuận.

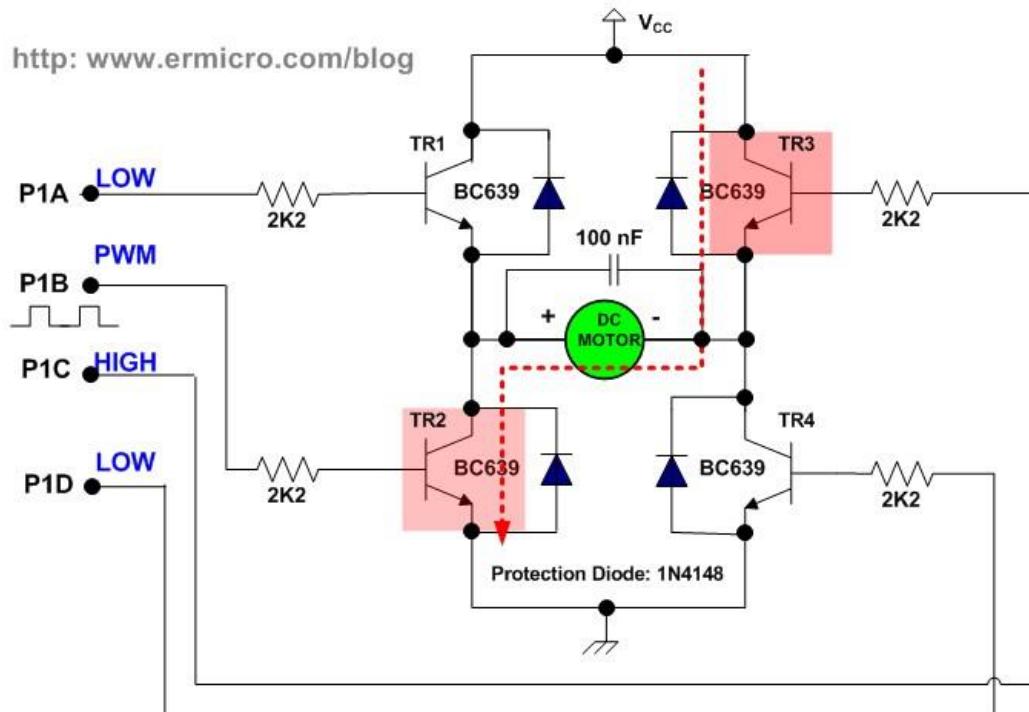


Forward Full Bridge PWM Mode

Hình 9-11. Dạng sóng tín hiệu điều khiển motor quay thuận.

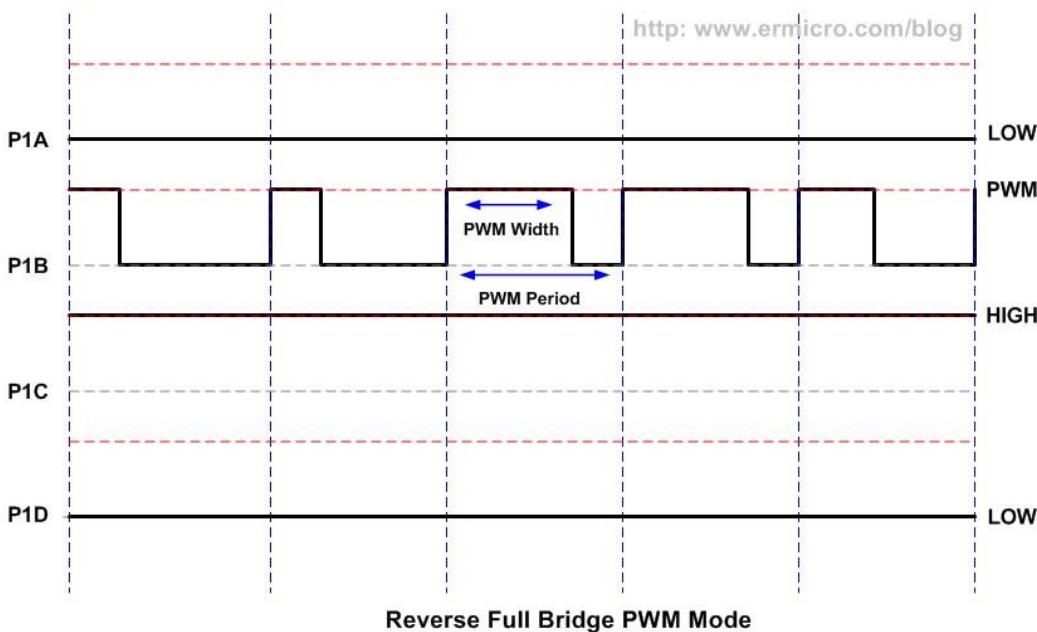
Trong chế độ này thì ngõ ra P1A ở mức HIGH để cho phép transistor TR1 dẫn, xung PWM xuất hiện ở ngõ ra P1D để điều khiển transistor TR4 và dòng chạy qua tải như đường đứt đoạn, 2 ngõ ra còn lại ở mức 0 để ngắt 2 transistor còn lại. Động cơ quay thuận.

Muốn đảo chiều động cơ thì điều khiển như hình 9-12 và 9-13.



Reverse Full Bridge PWM Mode H-Bridge Circuit

Hình 9-12. Chế độ hoạt động cầu đầy đủ điều khiển motor mạch cầu H, quay nghịch.



Hình 9-13. Dạng sóng tín hiệu điều khiển motor quay nghịch.

Để điều khiển các ngõ ra P1A, P1B, P1C và P1D thì điều khiển bởi 4 bit CCP1M nằm trong thanh ghi CCP1CON.

9.4 CÁC LỆNH ĐIỀU KHIỂN

9.4.1 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 khói CCP.

Hiệu lực: Cho tất cả các vi điều khiển PIC tích hợp phần cứng CCP

Ví dụ 9-1: `setup_ccp1(CCP_PWM);` khởi tạo khói CCP1 có chức năng PWM

9.4.2 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 như sau

9.4.3 LỆNH SETUP_TIMER_2 - LỆNH ĐỊNH CẤU HÌNH CHO 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 ngắn.

Chức năng: Khởi tạo cho TIMER2.

Mode chỉ định kiểu bộ chia của timer từ tần số của mạch dao động.

Giá trị của timer có thể đọc hoặc ghi dùng lệnh GET_TIMER2() và SET_TIMER2().

TIMER2 là timer 8 bit.

Có hiệu lực: Cho tất cả các vi điều khiển PIC có timer 2.

9.4.4 LỆNH SET_TIMERx(value) - LỆNH THIẾT LẬP GIÁ TRỊ CHO TIMER

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.

Có hiệu lực: cho tất cả các vi điều khiển PIC có timer.

Ví dụ 9-2: SET_TIMER2 (0); //reset timer2

9.5 CÁC CHƯƠNG TRÌNH ỦNG DỤNG PWM – BÀI TẬP

9.5.1 ĐIỀU KHIỂN ĐỘ SÁNG CỦA ĐÈN CẤP 1/10 DÙNG PWM

Bài 9-1: Sử dụng **PWM** của PIC 16F887 để điều khiển 1 đèn led. Cho tần số tụ thạch anh là **20MHz**. Cho chu kỳ PWM là **0,8ms**.
Hãy tính toán các thông số và viết chương trình để điều khiển led sáng với cấp độ 1 bằng 1 phần 10 độ sáng cực đại.

- Tính toán

Tần số thạch anh: $f_{OSC} = 20MHz$ nên chu kỳ là:

$$T_{OSC} = \frac{1}{f_{OSC}} = \frac{1}{20MHz} = 0,05\mu S = 50ns$$

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 $PR2$:

postscale
không dùng - chia 1:1

$$[(PR2) + 1] = \frac{PERIOD_{PWM}}{4 * T_{OSC} * PV_{TMR2}} = \frac{800,000ns}{4 * 50ns * 16} = 250$$

Vậy $PR2 = 249$.

Lệnh khởi tạo cho timer2 là: `setup_timer_2(T2_DIV_BY_16, 249, 1)`

Tính hệ số chu kỳ:

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:

$$DUTY_CYCLE_{PWM} = PERIOD_{PWM} \boxed{MAX}$$

Ta có công thức: $DUTY_CYCLE_{PWM} = (CCPRxL:CCPxCON < 5:4>) * T_{OSC} * PV_{TMR2}$

Ta tìm giá trị lớn nhất của $CCPRxL:CCPxCON < 5:4>$.

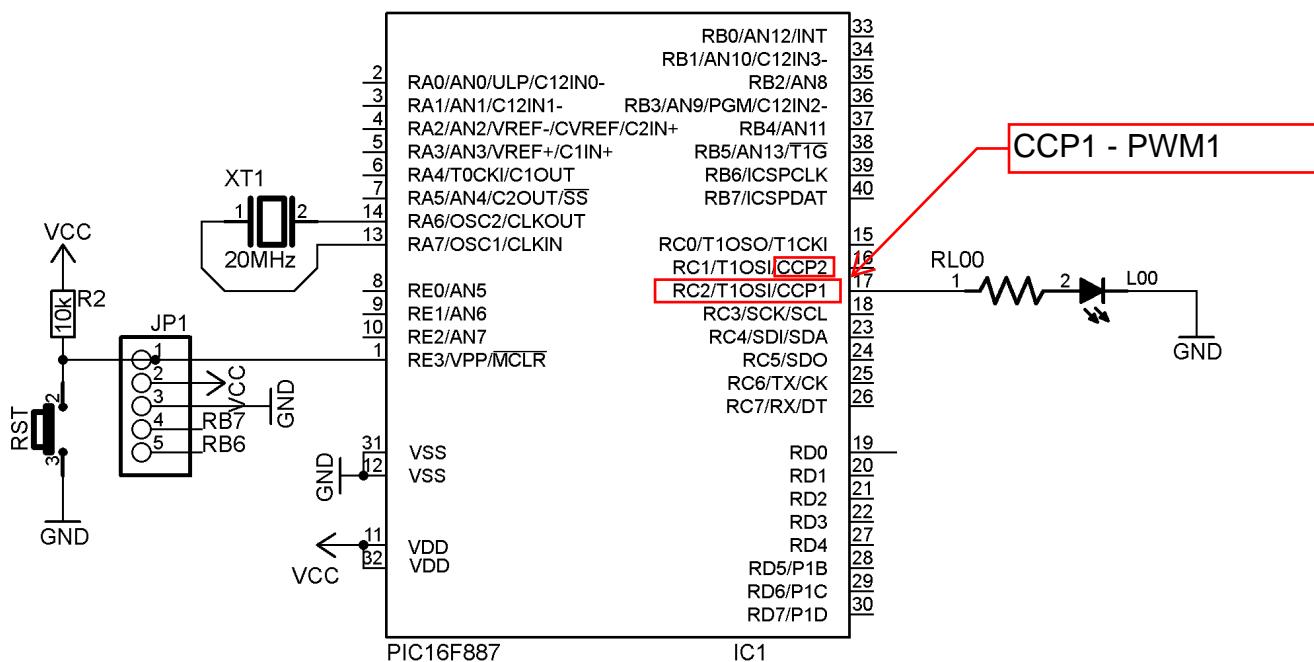
Suy ra: $(CCPRxL:CCPxCON < 5:4>) = \frac{DUTY_CYCLE_{PWM_MAX}}{T_{OSC} * PV_{TMR2}} = \frac{PERIOD_{PWM}}{T_{OSC} * PV_{TMR2}}$

$$(CCPRxL:CCPxCON < 5:4>) = \frac{PERIOD_{PWM}}{T_{OSC} * PV_{TMR2}} = \frac{800,000ns}{50ns * 16} = 1000 \quad \boxed{100\% \text{ độ rộng}}$$

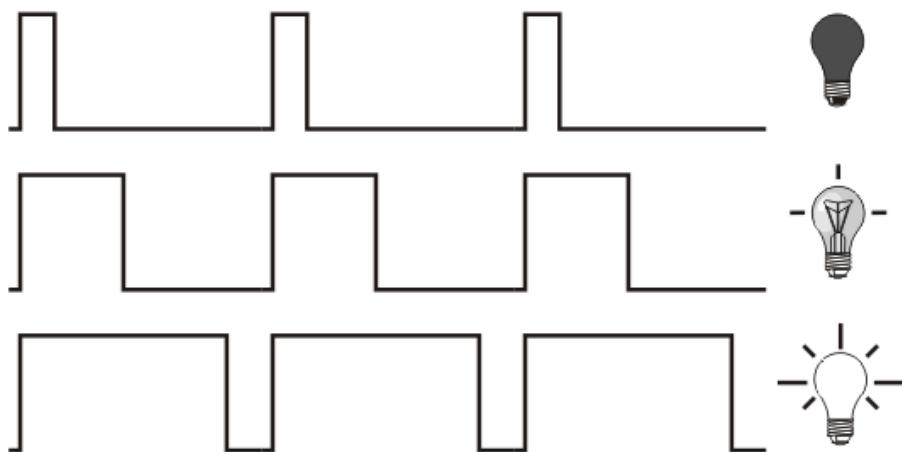
Giá trị điều khiển theo yêu cầu là 100.

Cấp độ 0 là 0, cấp độ 1 là 100, cấp độ 2 là 200, ... cấp độ cực đại là 1000.

- Mạch điều khiển:

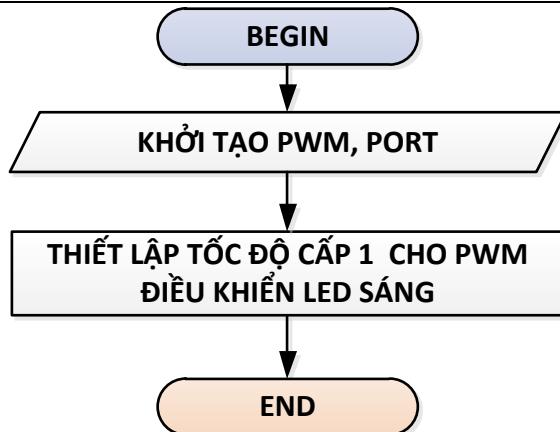


Hình 9-14. Mạch điều khiển thay đổi cường độ sáng của đèn dùng PWM.



Hình 9-15. Các dạng tín hiệu thay đổi cường độ sáng của đèn dùng PWM.

- Lưu đồ:



Hình 9-16. Lưu đồ điều khiển đèn sáng dùng PWM.

- Chương trình:

```

#include <TV_16F887.C>
UNSIGNED INT16 BIEN_TOC_DO;
VOID MAIN()
{
    SET_TRIS_C(0x00);
    SETUP_CCP1(CCP_PWM);
    SETUP_TIMER_2(T2_DIV_BY_16, 249, 1);
    BIEN_TOC_DO=100;
    SET_PWM1_DUTY(BIEN_TOC_DO);
    WHILE (TRUE) {}
}
  
```

Annotations on the code:

- //khởi tạo PWM
- thiết lập độ rộng xung % - dưới dạng số

- Giải thích chương trình:

Chương trình chính thực hiện các yêu cầu:

Khởi tạo portC là xuất, chọn chế độ PWM, khởi tạo Timer2 hoạt động đếm xung nội với các thông số đã tính bao gồm: chọn bộ chia trước là 16, hằng số nạp cho thanh ghi PR2 là 249, chọn bộ chia sau là 1.

Gán biến tốc độ bằng 100, tương ứng cấp tốc độ là 1. Khởi tạo hệ số công tác để tạo xung cấp tốc độ 1, thực hiện while.

9.5.2 ĐIỀU KHIỂN ĐỘ SÁNG CỦA ĐÈN 10 CẤP DÙNG PWM

Bài 9-2: Sử dụng PWM của PIC 16F887 để điều khiển 1 đèn led. Cho tần số tụ thạch anh là 20MHz. Cho chu kỳ PWM là 0,8ms.

Hãy tính toán các thông số và viết chương trình điều khiển led **thay đổi độ sáng 10 cấp** bằng 2 nút nhấn **UP** và **DW**. **Không tính cấp độ 0**.

Hiển thị cấp độ dạng số nhị phân ở các **led đơn** nối với portD.

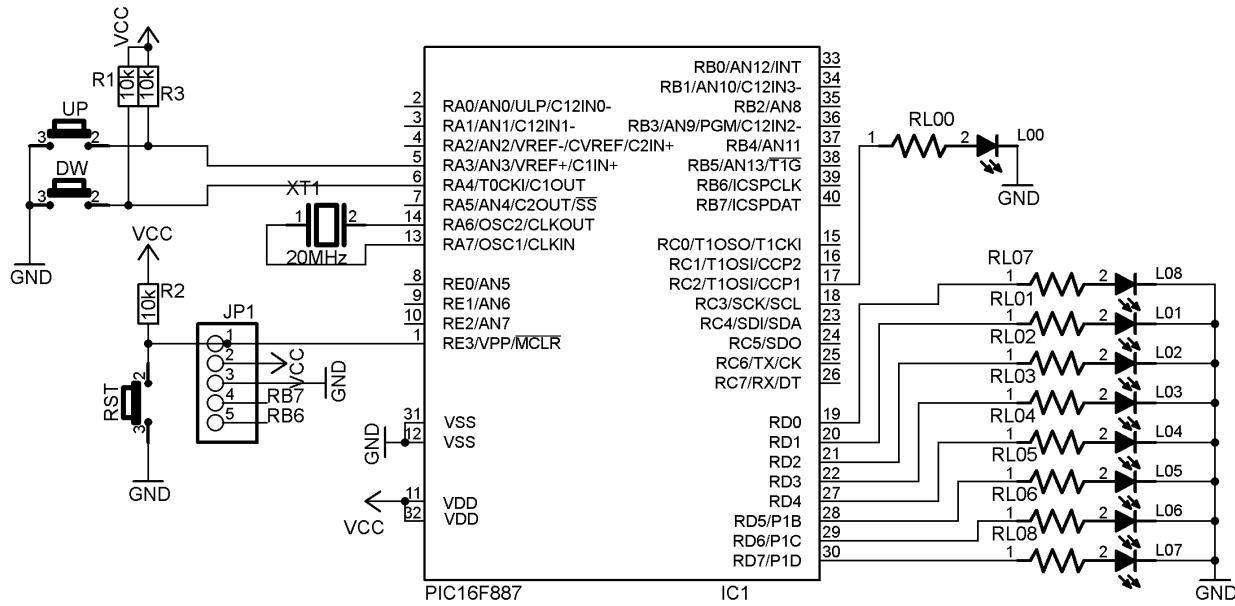
Giá trị thay đổi cho **1 cấp** là **10**.

- Tính toán: giống bài 9-1

Theo tính toán thì thông số thay đổi hệ số chu kỳ từ 0 đến 1000.

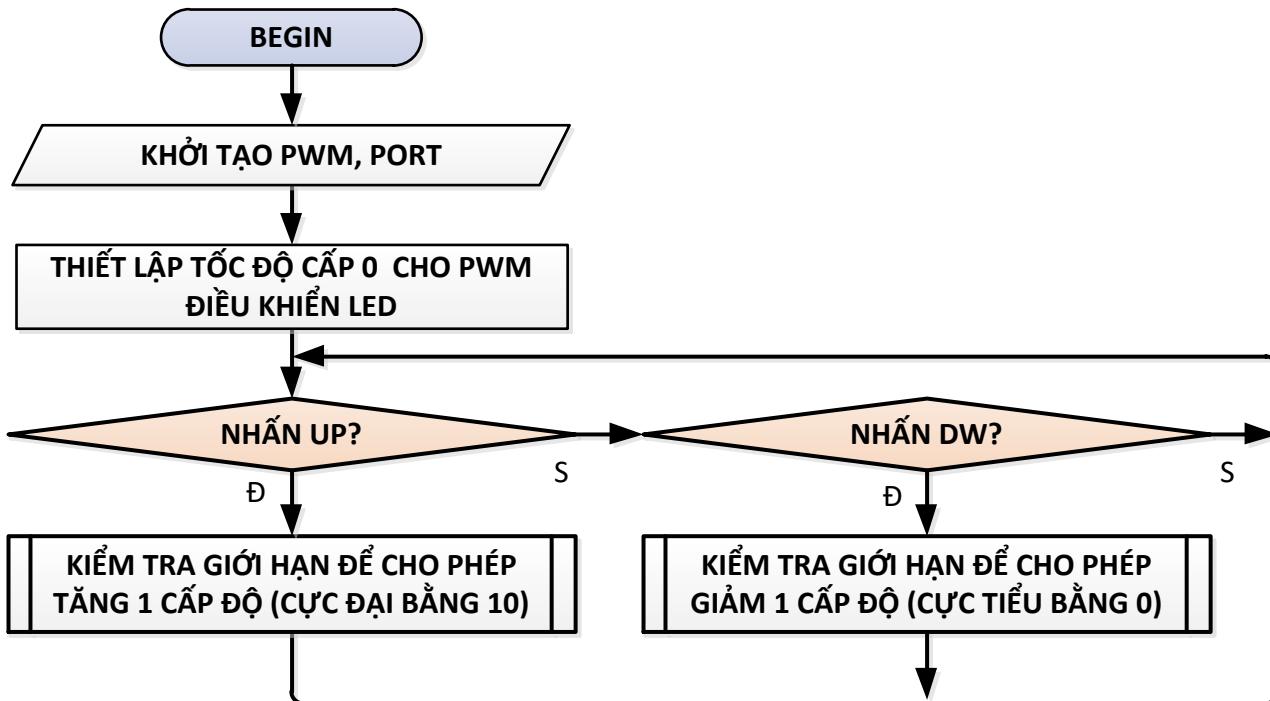
Theo yêu cầu của bài thì giá trị thay đổi mỗi cấp là 10, khi đó: cấp độ 0 là 0, cấp độ 1 là 10, cấp độ 2 là 20, ... cấp độ 10 là 100. Các giá trị từ 100 đến 1000 thì không sử dụng. Lý do khi chạy thực nghiệm thì từ 0 đến 100 ta nhìn thấy cường độ sáng của led thay đổi, còn từ 110 đến 1000 thì không còn quan sát được bằng mắt thường nữa.

- Mạch điều khiển:



Hình 9-17. Mạch thay đổi cường độ sáng của đèn dùng PWM và 2 nút nhấn.

- Lưu đồ:



Hình 9-18. Lưu đồ điều khiển đèn sáng dùng PWM và 2 nút nhấn.

- Chương trình:

```
#INCLUDE <TV_16F887.C>
#define UP PIN_A3
#define DW PIN_A4
UNSIGNED INT CAPDO=0;

VOID PHIM_UP ()
{
    IF (!INPUT(UP))
    {
        DELAY_MS (20);
        IF (!INPUT(UP))
        {
            CAPDO++;
            IF (CAPDO>10)
                CAPDO=10;
        }
    }
}
```

```

        IF (CAPDO<10)
        {
            CAPDO++;
            SET_PWM1_DUTY(CAPDO*10);
            OUTPUT_D(CAPDO);
            DO {} WHILE (!INPUT(UP));
        }
    }

VOID PHIM_DW ()
{
    IF (!INPUT(DW))
    {
        DELAY_MS (20);
        IF (!INPUT(DW))
        {
            IF (CAPDO>0)
            {
                CAPDO--;
                SET_PWM1_DUTY(CAPDO*10);
                OUTPUT_D(CAPDO);
                DO {} WHILE (!INPUT(DW));
            }
        }
    }
}

VOID MAIN()
{
    SET_TRIS_C(0x00);
    SET_TRIS_A(0xFF);
    SET_TRIS_D(0x00);
    OUTPUT_D(CAPDO);

    SETUP_CCP1(CCP_PWM);
    SETUP_TIMER_2(T2_DIV_BY_16, 249, 1);
    SET_PWM1_DUTY(CAPDO*10);

    WHILE (TRUE)
    {
        PHIM_UP();
        PHIM_DW();
    }
}

```

TRẠNG THÁI BD

- Giải thích chương trình:

Chương trình chính thực hiện các yêu cầu:

Khởi tạo các port, PWM, Timer2 và tốc độ theo số cấp ban đầu là 0.

Vòng lặp while thực hiện kiểm tra nhấn phím UP và DW.

Chương trình con phím UP kiểm tra nếu có nhấn thì tiến hành chống dội và kiểm tra nếu chưa bằng cấp 10 thì tăng cấp độ lên 1, cập nhật tốc độ mới, xuất ra port để biết cấp tốc độ đang thực hiện.

Chương trình con phím DW kiểm tra nếu có nhấn thì tiến hành chống dội và kiểm tra nếu chưa bằng cấp 0 thì giảm bớt 1 cấp độ, cập nhật tốc độ mới, xuất ra port để biết cấp tốc độ đang thực hiện.

Bài tập 9-1: Hãy thiết kế mạch và viết chương trình giống bài 9-2 nhưng không hiển thị cấp tốc độ ở led đơn mà hiển thị ở 2 led 7 đoạn và có thêm 1 nút Stop khi nhấn thì tắt led.

9.5.3 ĐIỀU KHIỂN THAY ĐỔI TỐC ĐỘ ĐỘNG CƠ DC DÙNG PWM

Bài 9-3: Sử dụng PWM của PIC 16F887 để điều khiển thay đổi tốc độ động cơ DC. Cho tần số tụ thạch anh là 20MHz. Cho chu kỳ PWM là 0,8ms. Động cơ dùng nguồn 24V DC, dòng 3A.

Hãy tính toán các thông số và viết chương trình điều khiển động cơ thay đổi tốc độ 10 cấp bằng 2 nút nhấn UP và DW. Nút Stop khi nhấn thì động cơ ngừng, **không tính cấp độ 0**.

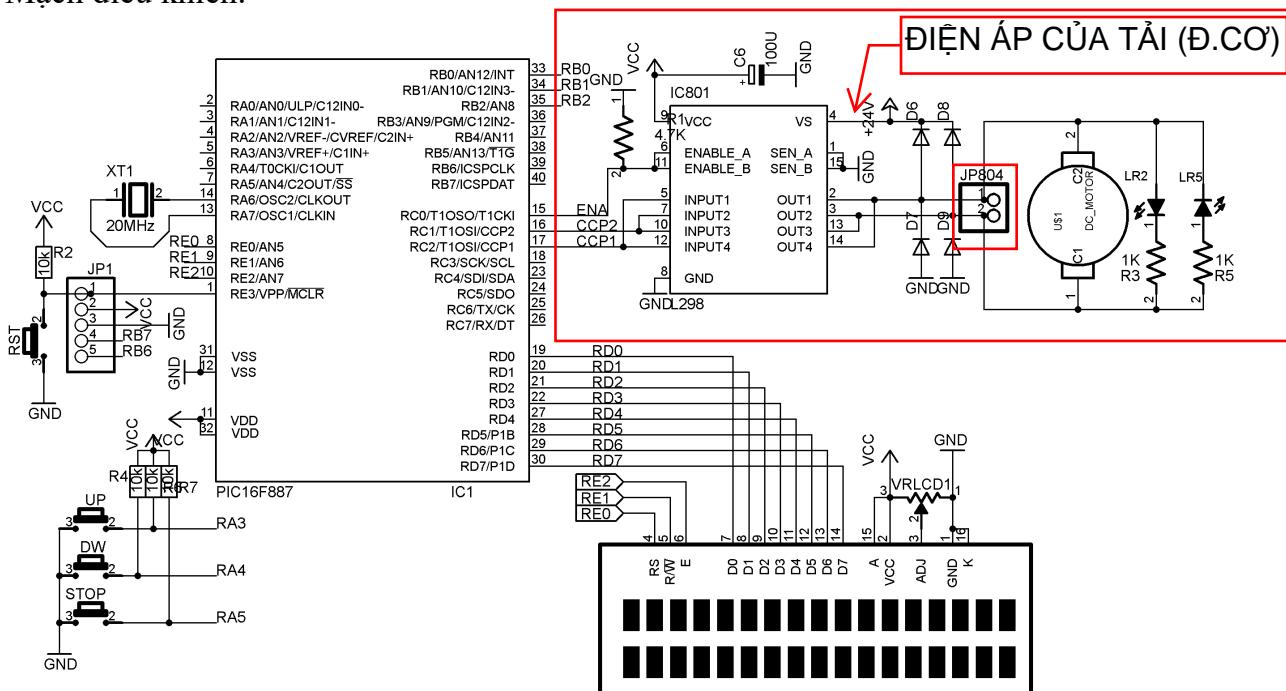
Hiển thị cấp độ trên LCD.

Giá trị thay đổi cho 1 cấp là 100.

VẼ VÀO TỜ A4

- Tính toán: giống bài 9-1

- Mạch điều khiển:



Hình 9-19. Mạch điều khiển thay đổi tốc độ động cơ dùng PWM.

Để bài yêu cầu điều khiển động cơ DC dùng nguồn 24V DC và dòng là 3A thì ta phải dùng thêm mạch giao tiếp công suất mới đủ dòng và áp điều khiển động cơ.

Có rất nhiều IC giao tiếp nhưng trong tài liệu này chọn 1 IC khá phổ biến là L298 có thể điều khiển áp lên đến 48V DC và dòng tổng lên đến 4A – theo datasheet.

Bên trong L298 có 2 mạch điều khiển nên có thể điều khiển được 2 động cơ, mỗi động cơ 2A.

Trong sơ đồ mạch ở trên thì 2 mạch điều khiển mắc song song nên mới có khả năng cấp dòng 3A cho động cơ.

Có 5 tín hiệu điều khiển nhưng do mắc song song 2 mạch công suất nên chỉ còn 3.

Tín hiệu cho phép ENA: khi ở mức 0 thì không cho phép bắt chấp 2 tín hiệu còn lại, mức 1 thì cho phép động cơ quay tùy thuộc vào 2 tín hiệu còn lại.

Bảng 9-2. Các trạng thái điều khiển động cơ DC.

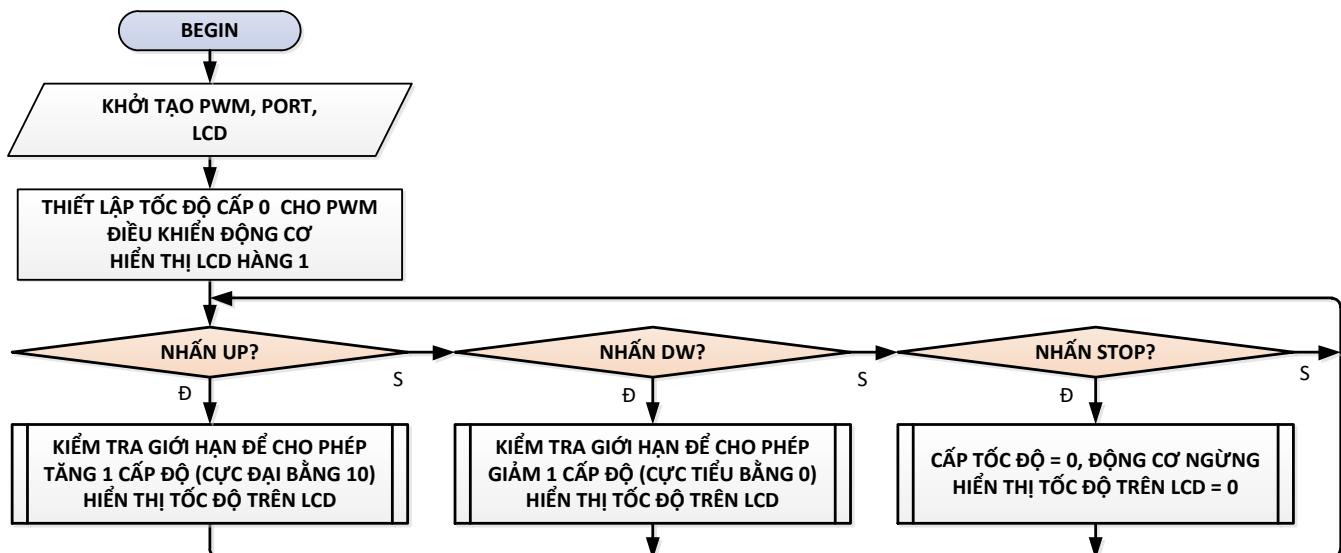
TT	ENA	CCP1	CCP2	TRẠNG THÁI ĐỘNG CƠ
1	0	X	X	NGƯỜNG
2	1	0	0	NGƯỜNG
3	1	1	1	NGƯỜNG
4	1	1	0	QUAY THUẬN
5	1	0	1	QUAY NGHỊCH

GHI CHÚ, ĐỀ
ĐIỀU KHIỂN

Có 2 led mắc song song với động cơ để báo hiệu chiều quay của động cơ.

Hai tín hiệu CCP1 và CCP2 điều khiển 2 tín hiệu IN của IC L298 với mục đích có thể điều khiển động cơ quay thuận, quay ngược và có thể thay đổi được tốc độ ở cả 2 chế độ quay thuận và ngược.

- Lưu đồ:



Hình 9-20. Lưu đồ điều khiển thay đổi tốc độ động cơ dùng PWM.

- Chương trình:

```

#include <TV_16F887.C>
#include <TV_LCD.C>

#define UP PIN_A3
#define DW PIN_A4
#define STOP PIN_A5
#define MOTOR_ENA PIN_C0

const unsigned char HANG1[16] = {"DK TOC DO DONGCO"};
SIGNED INT CAPDO=0, I;

VOID HIENTHI_CAP_TOODO ()
{
    LCD_COMMAND(ADDR_LINE2+14);    DELAY_US(10);
    LCD_DATA(CAPDO/10+0X30);
    LCD_DATA(CAPDO%10+0X30);
}

VOID PHIM_UP ()
{
    IF (!INPUT(UP))
    {
        DELAY_MS (20);
        IF (!INPUT(UP))
        {
            IF (CAPDO<10)
            {
                CAPDO++;           SET_PWM1_DUTY(CAPDO*10);
                HIENTHI_CAP_TOODO ();
                DO {} WHILE (!INPUT(UP));
            }
        }
    }
}

VOID PHIM_DW ()
{
    IF (!INPUT(DW))
    {
        ...
    }
}
    
```

```

DELAY_MS (20);
IF (!INPUT (DW)) >1
{
    IF (CAPDO>0)
    {
        CAPDO--; SET_PWM1_DUTY (CAPDO*10);
        HIENTHI_CAP_TOCDO ();
        DO {} WHILE (!INPUT (DW));
    }
}

VOID PHIM_STOP()
{
    IF (!INPUT (STOP))
    {
        CAPDO=0;
        SET_PWM1_DUTY (CAPDO*10);
        HIENTHI_CAP_TOCDO ();
    }
}

VOID MAIN()
{
    SET_TRIS_A(0xFF); setup_ccp2(ccp_off);
    SET_TRIS_C(0X00); SET_TRIS_D(0X00); SET_TRIS_E(0X00);
    SETUP_CCP1(CCP_PWM);
    SETUP_TIMER_2(T2 DIV BY 16,249,1);
    SET_PWM1_DUTY (CAPDO*10); //KHỞI TẠO LCD,
                           //ADC, PWM, TIMER,
                           //COUNTER, NGẮT...
                           //(TỜ A4)

    LCD_SETUP();
    LCD_COMMAND (ADDR_LINE1); DELAY_US (10);
    LCD_DATA("DK TOC DO DONGCO");

    OUTPUT_HIGH(MOTOR_ENA);
    HIENTHI_CAP_TOCDO ();
    WHILE (TRUE)
    {
        PHIM_UP();
        PHIM_DW();
        PHIM_STOP();
    }
}

```

PHÍM START:
SETUP_CCP1(CCP_PWM);
hoặc
OUTPUT_HIGH(MOTOR_ENA);

CHO PHÉP ĐỘNG CƠ

TRẠNG THÁI BẢN ĐẦU

- Giải thích chương trình:

Chương trình chính thực hiện các yêu cầu:

Khởi tạo các port, PWM, Timer2 và tốc độ theo số cấp ban đầu là 0.

Khởi tạo LCD, hiển thị thông tin cho hàng 1.

Cho phép motor sẵn sàng hoạt động và hiển thị cấp độ ban đầu là 0.

Vòng lặp while thực hiện kiểm tra nhấn phím UP, DW và STOP.

Chương trình con phím UP kiểm tra nếu có nhấn thì tiến hành chống dội và kiểm tra nếu chưa bằng cấp 10 thì tăng cấp độ lên 1, cập nhật tốc độ mới, hiển thị ra LCD cấp tốc độ đang thực hiện.

Chương trình con phím DW kiểm tra nếu có nhấn thì tiến hành chống dội và kiểm tra nếu chưa bằng cấp 0 thì giảm bớt 1 cấp độ, cập nhật tốc độ mới, hiển thị ra LCD cấp tốc độ đang thực hiện.

Chương trình con phím STOP kiểm tra nếu có nhấn thì tiến hành cho cấp tốc độ bằng 0, cho động cơ ngừng, hiển thị LCD cấp tốc độ bằng 0.

Bài tập 9-2: Hãy hiệu chỉnh bài 9-3 sao cho khi nhấn nút STOP thì động cơ ngừng đồng thời **đảo chiều động cơ**

Khi động cơ quay thuận thì hiển thị “FOR” trên LCD ở 3 ký tự đầu hàng 2.

~~setup_ccp1(ccp_off);
setup_ccp2(ccp_pwm);~~

Bài tập 9-3: Hãy hiệu chỉnh bài 9-3 với 2 nút nhấn UP, DW bây giờ có tên là START, INV.
Khi nhấn START thì động cơ quay tốc độ tự động tăng dần từng cấp sau mỗi giây
cho đến khi đạt cấp 10. Chiều quay mặc nhiên ban đầu là quay thuận.
Khi nhấn INV thì động cơ ngừng ngay và đảo chiều quay.
Nút thứ 3 là STOP chỉ khi nhấn thì động cơ giảm tốc cho đến khi ngừng, mỗi lần
giảm 1 cấp cho đến khi bằng 0.
Khi động cơ quay thuận thì hiển thị “FOR” trên LCD ở 3 ký tự đầu hàng 2.
Khi động cơ quay ngược thì hiển thị “REV” trên LCD ở 3 ký tự đầu hàng 2.

Bài tập 9-4: Hãy hiệu chỉnh bài 9-3 thay vì dùng 3 nút nhấn thì dùng bàn phím ma trận 4x4 nối
với portB.
Khi động cơ quay thuận thì hiển thị “FOR” trên LCD ở 3 ký tự đầu hàng 2.
Khi động cơ quay ngược thì hiển thị “REV” trên LCD ở 3 ký tự đầu hàng 2.

Bài tập 9-5: Hãy hiệu chỉnh bài 9-3 thay vì dùng 3 nút nhấn thì dùng bàn phím ma trận 4x4 nối
với portB.
11 phím số có mã từ 0x0 đến 0xA dùng để điều khiển 11 cấp tương ứng, khi nhấn
số 0 thì động cơ ngừng, khi nhấn phím số 1 thì động cơ chạy cấp tốc độ 1, khi nhấn
phím số 2 thì động cơ chạy cấp 5, ... tương tự cho các phím còn lại. Chỉ chạy 1
chiều.

Bài tập 9-6: Hãy thêm vào bài tập 9-5 các yêu cầu sau:
Khi nhấn phím có mã là 0x0B thì đảo chiều động cơ, hiển thị FOR và Rev tương
ứng trên LCD.
Khi nhấn phím có mã là 0x0d mà động cơ đang chạy thì không có tác dụng, nếu
động cơ đang ngừng thì tự động quay theo chiều đã chọn và tăng tốc cho đến khi
đạt cực đại, thời gian tăng mỗi cấp là 1 giây.
Khi nhấn phím có mã là 0x0e mà động cơ đang ngừng thì không có tác dụng, nếu
động cơ đang chạy thì tự động giảm tốc cho đến khi ngừng, thời gian giảm mỗi cấp
là 1 giây.

Bài tập 9-7: Hãy thiết kế hệ thống điều khiển 2 motor để kéo 2 bánh xe của 1 chiếc xe 3 bánh bằng
4 nút nhấn TÓI, LÙI, PHẢI, TRÁI.
Khi nhấn nút TÓI thì xe chạy tới, nếu tiếp tục nhấn và giữ thì xe chạy tăng tốc, khi
không còn nhấn thì xe ngừng. Nếu muốn chạy tiếp thì nhấn lại.
Khi nhấn nút LÙI thì xe chạy lùi, nếu tiếp tục nhấn và giữ thì xe chạy lùi tăng tốc,
khi không còn nhấn thì xe ngừng. Nếu muốn chạy tiếp thì nhấn lại.

Khi nhấn nút PHẢI thì xe vừa chạy tới vừa quẹo phải: động cơ bên phải chạy cấp độ 1, động cơ bên trái chạy cấp độ tăng dần nếu tiếp tục nhấn và giữ, sự chênh lệch này làm xe quẹo phải.

Khi nhấn nút TRÁI thì xe vừa chạy tới vừa quẹo trái: động cơ bên trái chạy cấp độ 1, động cơ bên phải chạy cấp độ tăng dần nếu tiếp tục nhấn và giữ, sự chênh lệch này làm xe quẹo trái.

Thiết kế mạch dùng 2 PWM của vi điều khiển PIC16F887 dùng transistor và relay, mỗi PWM điều khiển 1 động cơ. Relay có chức năng đảo chiều. PWM có chức năng thay đổi tốc độ.

9.6 CÂU HỎI ÔN TẬP – TRẮC NGHIỆM

9.5.4 CÂU HỎI ÔN TẬP

Câu 9-1: Hãy cho biết tên các thanh ghi liên quan đến khôi CCP của vi điều khiển PIC16F887.

Câu 9-2: Hãy cho biết ứng dụng của PWM trong điều khiển động cơ.

9.5.5 CÂU HỎI MỞ RỘNG

Câu 9-3: Hãy so sánh vi điều khiển không tích hợp chức năng PWM và có tích hợp PWM trong điều khiển thay đổi tốc độ động cơ DC.

9.5.6 CÂU HỎI TRẮC NGHIỆM

VỊ ĐIỀU KHIỂN PIC16F8873: TRUYỀN DỮ LIỆU UART

10.1 GIỚI THIỆU

Truyền dữ liệu nối tiếp có nhiều tiện ích trong điều khiển nên hầu hết các vi điều khiển đều tích hợp, ở chương này chúng ta sẽ khảo sát nguyên lý hoạt động truyền dữ liệu đồng bộ và không đồng bộ, cách viết chương trình truyền dữ liệu cho các ứng dụng.

Sau khi kết thúc chương này bạn có thể sử dụng được truyền dữ liệu nối tiếp của các vi điều khiển.

10.2 TỔNG QUAN VỀ CÁC KIỂU TRUYỀN DỮ LIỆU

Có nhiều kiểu truyền dữ liệu phổ biến tích hợp trong các họ vi điều khiển bao gồm:

- Truyền dữ liệu nối tiếp đồng bộ và **bát đồng bộ** (USART Universal Synchronous Asynchronous Receiver and Transmitter).
- Truyền dữ liệu giữa các vi điều khiển với các thiết bị ngoại vi (SPI Serial Peripheral Interface).
- Truyền dữ liệu 2 dây (I2C: Inter-Integrated Circuit).

Ở kiểu truyền nối tiếp đồng bộ thì có 1 đường dữ liệu và 1 đường xung clock, thiết bị nào cấp xung clock thì thiết bị đóng vai trò là chủ, thiết bị nhận xung clock đóng vai trò là tớ, tốc độ truyền dữ liệu phụ thuộc vào tần số xung clock.

Ở kiểu truyền nối tiếp bát đồng bộ thì có 1 đường phát dữ liệu và 1 đường nhận dữ liệu, không còn tín hiệu xung clock nên gọi là bát đồng bộ. Để truyền được dữ liệu thì cả bên phát và bên nhận phải tự tạo xung clock có cùng tần số và thường gọi là tốc độ truyền dữ liệu (baud), ví dụ 2400baud, 4800baud, ..., 2400baud có nghĩa là truyền 2400 bit trên 1 giây.

Hai kiểu truyền SPI và I2C sẽ được trình bày ở chương tiếp theo.

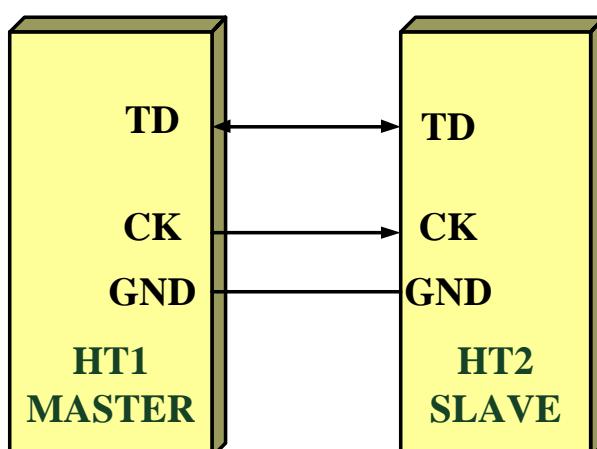
10.3 TRUYỀN DỮ LIỆU NỐI TIẾP ĐỒNG BỘ VÀ KHÔNG ĐỒNG BỘ

Truyền dữ liệu đồng bộ gồm các đường truyền dữ liệu (DT) và tín hiệu xung clock (CK) – chức năng của CK dùng để dịch chuyển dữ liệu, **mỗi 1 xung ck là 1 bit dữ liệu được truyền đi**.

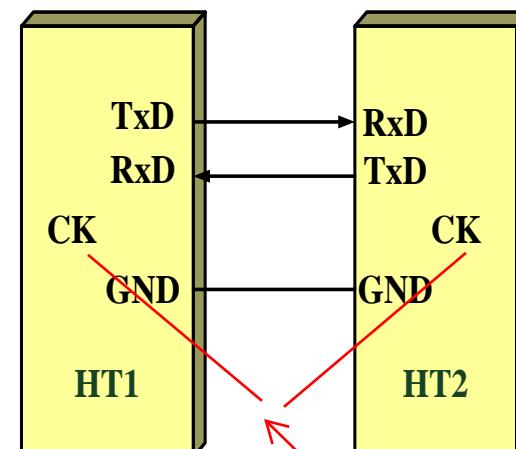
Trong hệ thống truyền dữ liệu đồng bộ, hệ thống nào cung cấp xung CK thì đóng vai trò là master (chủ) – những hệ thống còn lại nhận xung ck đóng vai trò là slave (tớ).

Tốc độ truyền dữ liệu chính là tốc độ của xung ck – chính là tần số xung ck.

Ví dụ tần số xung ck là 1MHz thì tốc độ truyền dữ liệu là 1MBPS – 1M baud.



Hình 10-1. Hệ thống truyền đồng bộ.



Hình 10-2. Hệ thống truyền bát đồng bộ.

CÓ CÙNG TẦN SỐ
TỐC ĐỘ BAUD
(BIT PER SECOND)
BPS

Truyền dữ liệu không đồng bộ giống như hệ thống truyền dữ liệu đồng bộ nhưng không có xung CK. Không còn phân biệt chủ và tớ – các hệ thống là ngang cấp.

Mỗi 1 xung ck là 1 bit dữ liệu được truyền đi – bây giờ không còn xung CK thì làm sao để truyền dữ liệu?

Để truyền dữ liệu thì mỗi hệ thống phải có 1 mạch dao động tạo xung CK – hai hệ thống sẽ có 2 mạch dao động độc lập nhưng phải cùng tần số hay cùng tốc độ.

10.4 TRUYỀN DỮ LIỆU CỦA VI ĐIỀU KHIỂN PIC 16F887

10.4.1 GIỚI THIỆU TRUYỀN DỮ LIỆU EUSART

Vi điều khiển PIC16F887 có khối truyền dữ liệu đồng bộ, bất đồng bộ đa năng cài tiến. Khối truyền dữ liệu nối tiếp đa năng bao gồm bộ phát xung clock tạo tốc độ truyền, các thanh ghi dịch và bộ đếm dữ liệu rất cần thiết để thực hiện truyền hoặc nhận dữ liệu nối tiếp một cách độc lập.

Khối EUSART cũng có thể xem là giao tiếp truyền dữ liệu nối tiếp SCI (Serial Communication Interface), có thể định cấu hình cho truyền dữ liệu bất đồng bộ song công hoặc đồng bộ bán song công.

Truyền dữ liệu song công được sử dụng để truyền dữ liệu giữa các hệ thống ngoại vi như thiết bị đầu cuối CRT và máy tính.

Truyền dữ liệu đồng bộ bán song công được sử dụng để truyền dữ liệu giữa các hệ thống ngoại vi như các bộ ADC, DAC, bộ nhớ nối tiếp EEPROM hoặc các bộ vi điều khiển. Các thiết bị này thường không có nguồn xung clock bên trong để tạo tốc độ baud nên cần phải sử dụng nguồn xung clock từ bên ngoài.

Khối truyền dữ liệu của PIC16F887 có khả năng:

- Hoạt động truyền và dữ liệu song công bất đồng bộ.
- Bộ đếm nhận chứa được 2 kí tự.
- Bộ đếm phát chứa 1 kí tự.
- Có thể lập trình chiều dài dữ liệu 8 bit hoặc 9 bit.
- Có khói phát hiện địa chỉ 9 bit
- Có khói phát hiện bộ đếm nhận bị tràn
- Có khói phát hiện lỗi khung của kí tự nhận về.
- Có thể hoạt động chế độ chủ ở kiểu truyền dữ liệu đồng bộ bán song công.
- Có thể hoạt động chế độ tớ ở kiểu truyền dữ liệu đồng bộ bán song công.
- Có thể lập trình chọn cực cho xung clock ở chế độ truyền đồng bộ.

Khối EUSART được sử dụng cho các cấu trúc mở rộng sau, thích hợp cho hệ thống bus mạng kết nối cục bộ (LIN: Local Interconnect Network):

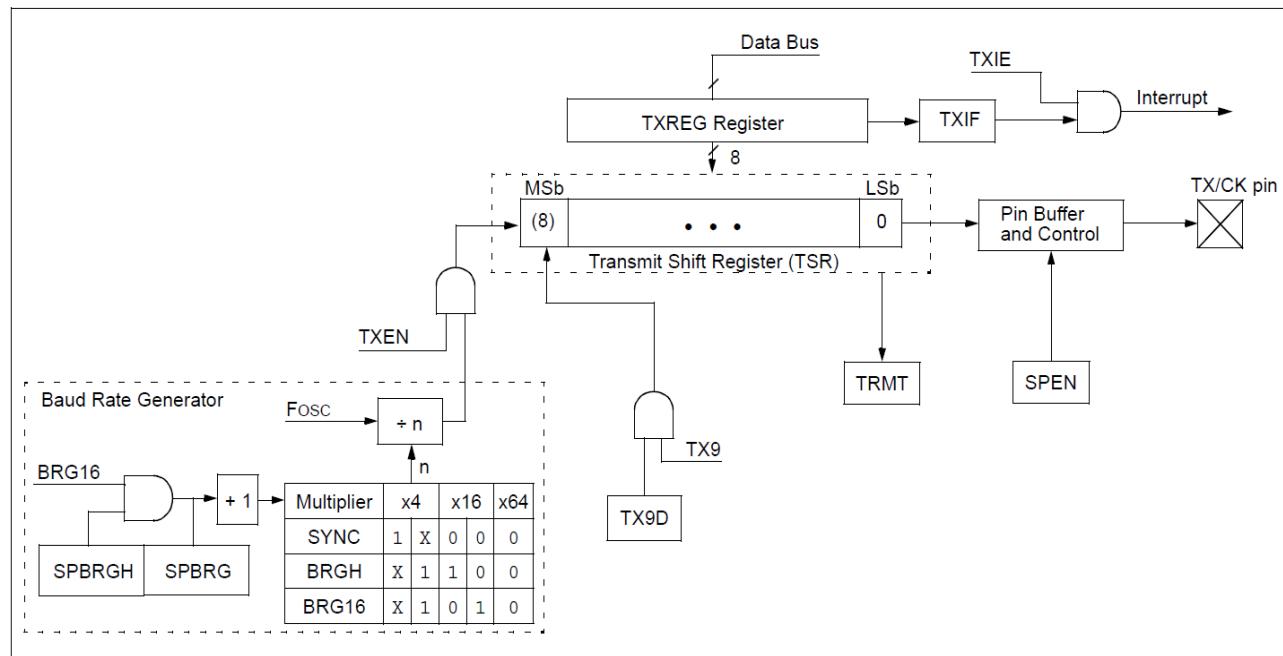
- Tự động phát hiện và thiết lập tốc độ baud.
- Có khói đánh thức PIC khỏi chế độ ngủ.
- Phát kí tự ngừng 13 bit.

10.4.2 KHỐI PHÁT DỮ LIỆU ESUART CỦA PIC16F887

Sơ đồ khối của khối phát như hình 10-3.

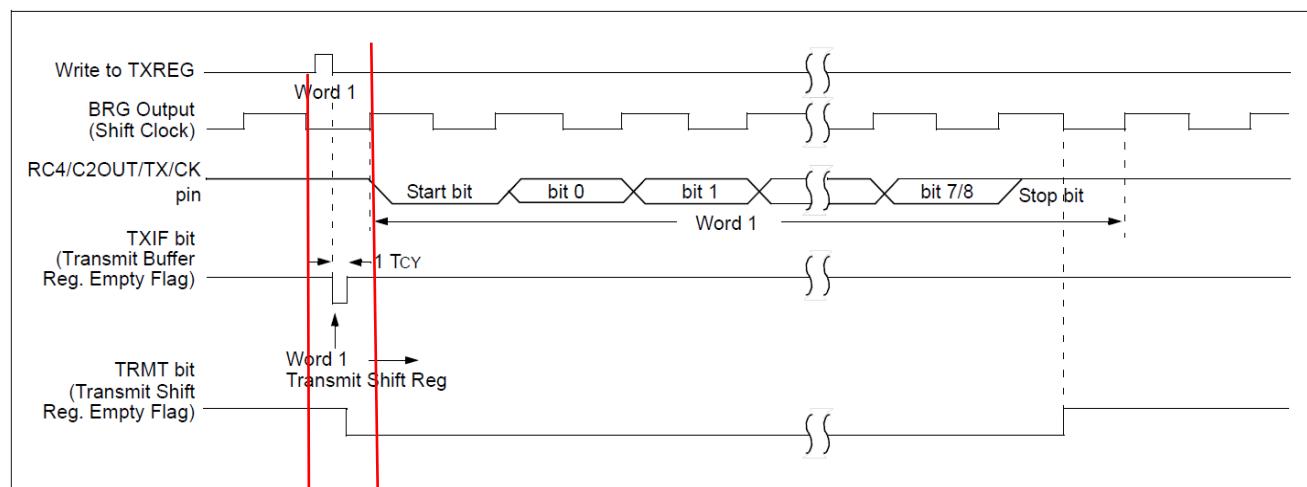
Thiết lập truyền dữ liệu bắt đồng bộ

- Thiết lập giá trị cho cặp thanh ghi SPBRGH và SPBRG và BRGH và bit BRG16 để có tốc độ baud mong muốn.
- Cho phép truyền dữ liệu bắt đồng bộ bằng cách xóa bit SYNC làm bit SPEN lên 1.
- Nếu truyền dữ liệu 9 bit thì thiết lập bit cho phép TX9 lên 1.
- Làm bit TXEN lên 1 để cho phép truyền dữ liệu.
- Nếu muốn sử dụng ngắt thì cho bit TXIE lên 1, cho phép ngắt ngoại vi và cho phép ngắt toàn cục.
- Nếu truyền dữ liệu 9 bit thì phải gán giá trị bit thứ 9 cho bit TX9D
- Tiến hành nạp giá trị cần truyền vào thanh ghi TXREG, khi đó quá trình truyền dữ liệu sẽ bắt đầu.



Hình 10-3. Sơ đồ khái niệm của khối phát dữ liệu của PIC16F887.

Dạng sóng truyền dữ liệu bắt đồng bộ như hình 10-4.



Hình 10-4. Dạng sóng truyền dữ liệu.

Giải thích dạng sóng:

Dạng sóng tín hiệu thứ hai là BRG: là tín hiệu tạo xung clock để dịch dữ liệu được phát liên tục.

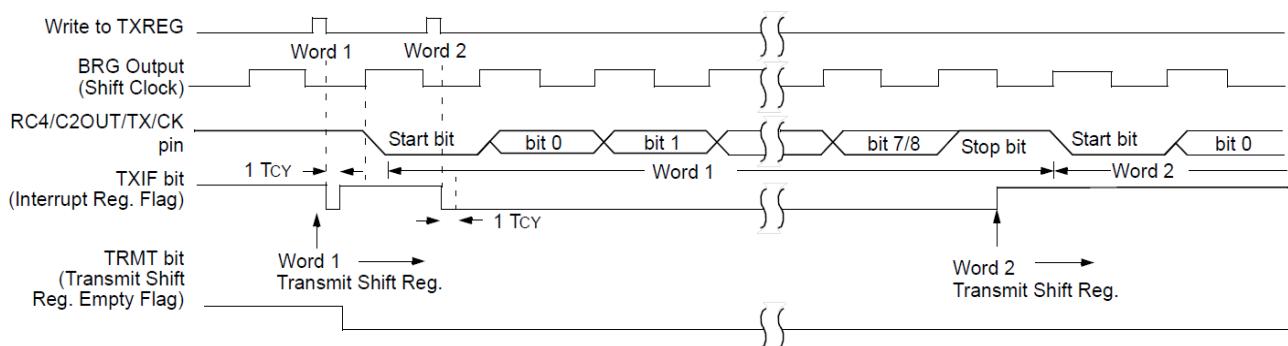
Dạng sóng tín hiệu thứ nhất: là khi thực hiện lệnh ghi dữ liệu cần truyền byte thứ nhất vào thanh ghi TXREG.

Dạng sóng thứ tư cho biết trạng thái của cờ báo ngắt TXIF: tín hiệu từ 1 xuống 0 rồi lên 1 trở lại cho biết trạng thái thanh ghi bộ đếm còn rỗng, thời gian xuống mức 0 chỉ tồn tại 1 chu kỳ.

Dạng sóng thứ năm cho biết trạng thái của bit TRMT: tín hiệu từ 1 xuống 0 để cho biết quá trình dịch dữ liệu bắt đầu và trở lại mức 0 khi đã dịch xong dữ liệu.

Dạng sóng thứ ba là ngõ xuất dữ liệu nối tiếp ra ngoài: bit đầu tiên được phát là bit Start, tiếp theo là bit dữ liệu thứ 0, 1, ..., đến bit dữ liệu cuối cùng là bit thứ 7 nếu truyền 8 bit, là bit thứ 8 nếu truyền 9 bit, bit cuối cùng là bit Stop.

Dạng truyền thứ 2 như hình 10-5, ở kiểu này thì truyền 2 byte:



Hình 10-5. Dạng sóng truyền 2 byte dữ liệu.

Giải thích dạng sóng:

Dạng sóng tín hiệu thứ nhất: là khi thực hiện lệnh ghi dữ liệu cần truyền byte thứ nhất vào thanh ghi TXREG, sau đó thực hiện tiếp byte thứ 2. Bộ đếm có thể chứa 2 byte để truyền đi.

Dạng sóng thứ tư cho biết trạng thái của cờ báo ngắt TXIF: tín hiệu từ 1 xuống 0 rồi lên 1 trở lại cho biết trạng thái thanh ghi bộ đếm còn rỗng 1 byte, sau đó nhận tiếp byte thứ 2 thì tín hiệu này xuống lại mức 0 và ở luôn mức 0 vì bộ đếm đầy. Sau khi truyền xong 1 byte thì bộ đếm rỗng nên tín hiệu này lên 1 trở lại, ta có thể ghi dữ liệu để phát tiếp.

Dạng sóng thứ ba là ngõ xuất dữ liệu nối tiếp ra ngoài: sẽ phát lần lượt từng byte.

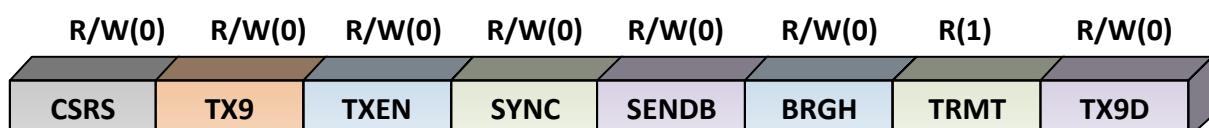
10.4.3 CÁC THANH GHI TXSTA VÀ RCSTA CỦA PIC16F887

Hoạt động của khối USART được điều khiển thông qua 3 thanh ghi như sau:

- Thanh ghi điều khiển và trạng thái của khối phát (TXSTA – transmit Status and Control).
- Thanh ghi điều khiển và trạng thái của khối nhận (RCSTA – transmit Status and Control).

a. Thanh ghi điều khiển và trạng thái của khối phát TXSTA

Tổ chức của thanh ghi như hình sau:



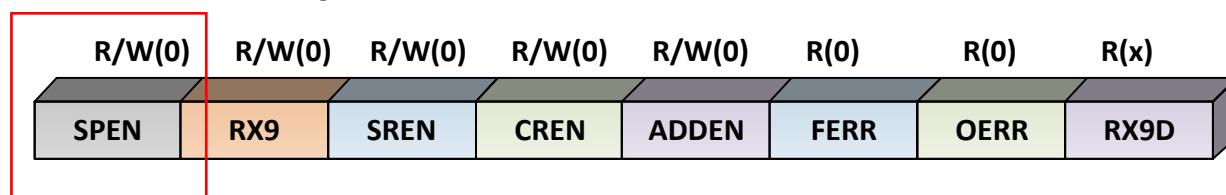
Hình 10-6. Thanh ghi TXSTA.

Bảng 10-1. Tóm tắt chức năng các bit trong thanh ghi TXSTA.

Bit	Kí hiệu	Chức năng (cho phép = 1; cấm = 0)
TXSTA.7	CSRS	Clock Source Select bit: Ở chế độ bất đồng bộ thì không có tác dụng. Ở chế độ bất đồng bộ thì có chức năng: CSRS = 1: Hoạt động ở chế độ chủ - phát xung clock. CSRS = 0: Hoạt động ở chế độ tớ - nhận xung clock.
TXSTA.6	TX9	9 bit Transmit Enable bit. TX9 = 1: thì truyền dữ liệu 9 bit. TX9 = 0: thì truyền dữ liệu 8 bit.
TXSTA.5	TXEN	Transmit Enable bit: TXEN = 1: cho phép phát dữ liệu. TXEN = 0: cấm phát dữ liệu.
TXSTA.4	SYNC	EUSART mode select bit: SYNC = 1: Cho phép truyền đồng bộ. SYNC = 0: Cho phép truyền bất đồng bộ.
TXSTA.3	SENDB	Send Break Character bit Ở chế độ không đồng bộ: SENDB = 1: gửi ngắt đồng bộ cho truyền dữ liệu tiếp theo SENDB = 0: báo hiệu truyền ngắt đồng bộ đã hoàn tất. Ở chế độ đồng bộ: không có tác dụng.
TXSTA.2	BRGH	High Baud Rate Select bit: BRGH=1: chế độ tốc độ cao. BRGH=0: chế độ tốc độ thấp. Ở chế độ đồng bộ: không có tác dụng.
TXSTA.1	TRMT	Bit xác định trạng thái của thanh ghi TSR. TRMT xuống mức 0 khi đang truyền dữ liệu. TRMT lên 1 khi đã truyền xong.
TXSTA.0	TX9D	Bit lưu dữ liệu phát thứ 9.

b. Thanh ghi điều khiển và trạng thái của khối nhận RCSTA

Tổ chức của thanh ghi như hình sau:



Hình 10-7. Thanh ghi RCSTA.

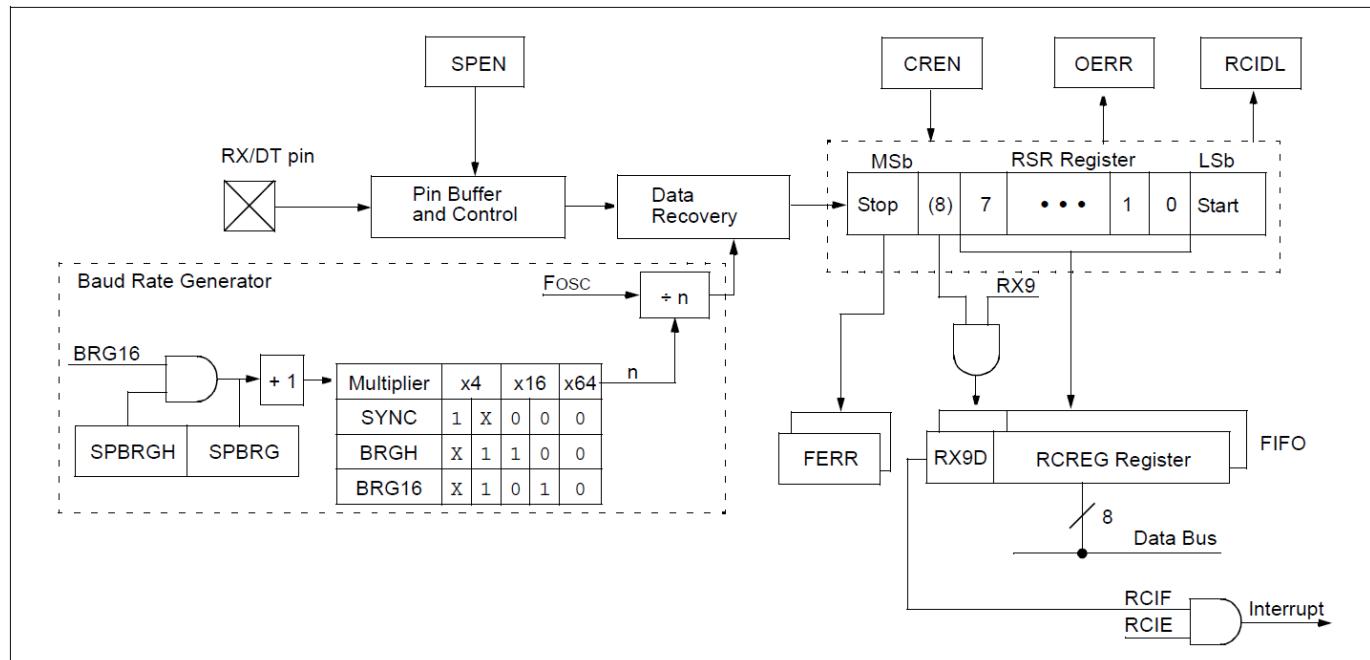
Bảng 10-2. Tóm tắt chức năng các bit trong thanh ghi RCSTA.

Bit	Kí hiệu	Chức năng (cho phép = 1; cấm = 0)
RCSTA.7	SPEN	Serial Port Enable bit: SPEN = 1: cho phép port nối tiếp (cấu hình các chân RX/TD và TX/CK là các chân cho truyền dữ liệu nối tiếp). SPEN = 0: thì cấm.
RCSTA.6	RX9	9 bit Receive Enable bit. RX9 = 1: thì cho phép nhận dữ liệu 9 bit. RX9 = 0: thì cho phép nhận dữ liệu 8 bit.
RCSTA.5	SREN	Single Receive Enable bit: Ở chế độ không đồng bộ: không có tác dụng. Ở chế độ đồng bộ - chủ SREN = 1: cho phép nhận đơn.

		SREN = 0: cảm nhận đơn. Bit SREN bị xóa sau khi quá trình nhận hoàn tất. Ở chế độ đồng bộ - tó: không có tác dụng.
RCSTA.4	CREN	Continuous Receive Enable bit: Ở chế độ không đồng bộ: CREN = 1: Cho phép nhận. CREN = 0: Cảm nhận. Ở chế độ đồng bộ: CREN = 1: Cho phép nhận liên tục cho đến khi bit CREN bị xóa. CREN = 0: Cảm nhận liên tục.
RCSTA.3	ADDEN	Address Detect Enable bit Ở chế độ không đồng bộ 9 bit: ADDEN = 1: Cho phép phát hiện địa chỉ, cho phép ngắt và nạp bộ đệm nhận khi bit RSR<8> lên 1. ADDEN = 0: cảm bộ phát hiện địa chỉ, tắt cả các byte được nhận và bit thứ 9 có thể dùng làm bit kiểm tra chẵn lẻ. Ở chế độ không đồng bộ 8 bit: không có tác dụng.
RCSTA.2	FERR	Framing Error bit: FERR = 1: xảy ra lỗi khung dữ liệu. FERR = 0: không có lỗi khung xảy ra.
RCSTA.1	OERR	Overrun Error bit OERR = 1: xảy ra lỗi tràn. OERR = 0: không xảy ra lỗi tràn.
RCSTA.0	RX9D	Ninth bit of Receive Data. Bit này dùng để lưu dữ liệu nhận về của bit thứ 9 hoặc có thể là bit kiểm tra chẵn lẻ.

10.4.4 KHỐI NHẬN DỮ LIỆU ESUART CỦA PIC16F887

Sơ đồ khối nhận dữ liệu của khối EUASRT như hình 10-8.



Hình 10-8. Sơ đồ khối của khối nhận dữ liệu của PIC16F887.

Dữ liệu nhận vào ở chân RX/DT và điều khiển khôi phục dữ liệu. Khối khôi phục dữ liệu là khối dịch tốc độ cao hoạt động gấp 16 lần tốc độ baud, trong khi đó thanh ghi nhận dữ liệu (RSR: Receive Shift Register) hoạt động dịch dữ liệu cùng với tốc độ baud.

Khi tắt cả 8 bit hoặc 9 bit dữ liệu được dịch vào thì ngay lập tức sẽ được truyền cho bộ đệm chưa được 2 kí tự dạng FIFO, nếu có thêm một kí tự nữa mà phần mềm chưa đọc 2 byte

đã nhận thì sẽ phát sinh lỗi tràn. Phần mềm không thể truy xuất bộ đệm FIFO và thanh ghi RSR mà chỉ được phép truy xuất thanh ghi RCREG.

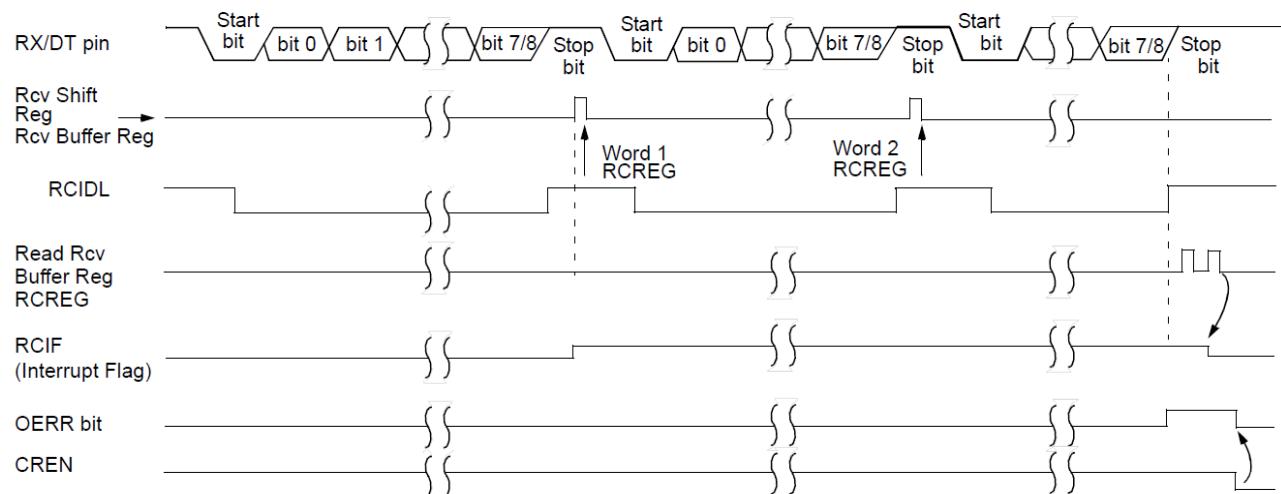
Khởi tạo quá trình nhận dữ liệu:

- Thiết lập giá trị cho cặp thanh ghi SPBRGH và SPBRG và BRGH và bit BRG16 để có tốc độ baud mong muốn.
- Cho phép truyền dữ liệu bắt đồng bộ bằng cách xóa bit SYNC làm bit SPEN lên 1.
- Nếu nhận dữ liệu 9 bit thì thiết lập bit cho phép RX9 lên 1.
- Làm bit CREN lên 1 để cho phép nhận dữ liệu.
- Nếu muốn sử dụng ngắt thì cho bit RCIE lên 1, cho phép ngắt ngoại vi và cho phép ngắt toàn cục.
- Cờ báo ngắt RCIF sẽ lên 1 khi một kí tự chuyển từ RSR sang bộ đệm, ngắt sẽ phát sinh nếu được phép.
- Đọc thanh ghi trạng thái nhận RCSTA để kiểm tra các cờ báo lỗi.
- Nhận dữ liệu 8 bit từ thanh ghi RCREG.
- Nếu có lỗi tràn xảy ra thì xóa bit cho phép nhận CREN

Khởi tạo quá trình nhận dữ liệu ở chế độ phát hiện 9 bit địa chỉ: chế độ này thường được dùng trong hệ thống truyền theo chuẩn RS-485.

- Thiết lập giá trị cho cặp thanh ghi SPBRGH và SPBRG và BRGH và bit BRG16 để có tốc độ baud mong muốn.
- Cho phép truyền dữ liệu bắt đồng bộ bằng cách xóa bit SYNC làm bit SPEN lên 1.
- Nếu muốn sử dụng ngắt thì cho bit RCIE lên 1, cho phép ngắt ngoại vi và cho phép ngắt toàn cục.
- Nếu nhận dữ liệu 9 bit thì thiết lập bit cho phép RX9 lên 1.
- Cho phép phát hiện địa chỉ 9 bit bằng cách cho bit ADDEN lên 1.
- Làm bit CREN lên 1 để cho phép nhận dữ liệu.
- Cờ báo ngắt RCIF sẽ lên 1 khi một kí tự chuyển từ RSR sang bộ đệm, ngắt sẽ phát sinh nếu được phép.
- Đọc thanh ghi trạng thái nhận RCSTA để kiểm tra các cờ báo lỗi. Bit thứ 9 luôn là 1.
- Nhận dữ liệu 8 bit từ thanh ghi RCREG, phần mềm sẽ tiến hành kiểm tra địa chỉ của thiết bị.
- Nếu có lỗi tràn xảy ra thì xóa bit cho phép nhận CREN.
- Nếu đúng địa chỉ của thiết bị thì tiến hành xóa bit ADDEN để cho phép nhận dữ liệu vào bộ đệm và phát sinh ngắt.

Dạng sóng nhận dữ liệu như hình 10-9:



Hình 10-9. Dạng sóng nhận dữ liệu.

Dạng sóng tín hiệu thứ nhất: là dữ liệu nhận về ở chân RX/DT, có 3 kí tự được nhận liên tục.

Dạng sóng tín hiệu thứ hai: cho biết sau khi nhận xong 1 kí tự (nhận xong bit stop) thì nạp lưu vào thanh ghi bộ đếm.

Dạng sóng tín hiệu thứ ba: cho biết tín hiệu RCIDL xuống mức 0 khi phát hiện bit start và lên 1 khi phát hiện bit stop.

Dạng sóng tín hiệu thứ năm: cho biết cờ nhận RCIF lên mức 1 khi nhận được 1 kí tự và xuống mức 0 khi đã tiến hành đọc xong các kí tự đã nhận.

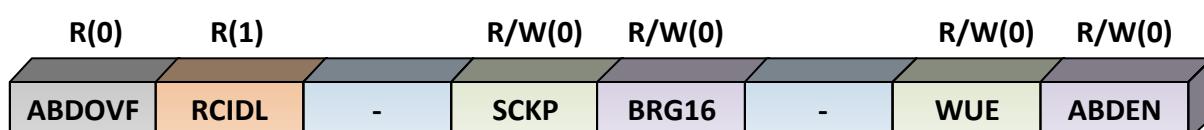
Dạng sóng tín hiệu thứ sáu: cho biết bit báo lỗi tràn OERR lên mức 1 khi nhận tới 3 kí tự.

Dạng sóng tín hiệu thứ bảy: cho biết khi xóa bit CREN thì xóa luôn bit báo lỗi tràn OERR.

Dạng sóng tín hiệu thứ tư: cho biết tiến hành đọc 2 byte dữ liệu (đã mất 1 byte), khi đọc xong thì xóa luôn cờ báo ngắt.

10.4.5 THANH GHI TẠO TỐC ĐỘ BAUD CỦA PIC16F887

Tổ chức của thanh ghi điều khiển tốc độ baud (BAUDCTL – Baud Rate Control) như hình 10-10.



Hình 10-10. Thanh ghi BAUDCTL.

Bảng 10-3. Tóm tắt chức năng các bit trong thanh ghi BAUDCTL.

Bit	Kí hiệu	Chức năng (cho phép = 1; cấm = 0)
BAUDCTL.7	ABDOVF	Auto – Baud detect Overflow bit: bit tự động phát hiện tràn tốc độ baud Ở chế độ không đồng bộ: ABDOVF = 1: cho biết timer tạo tốc độ baud tự động bị tràn. ABDOVF = 0: cho biết timer tạo tốc độ baud tự động không bị tràn. Ở chế độ đồng bộ: không có tác dụng.
BAUDCTL.6	RCIDL	Receive Idle Flag bit: bit cờ ngừng nhận Ở chế độ không đồng bộ: RCIDL = 1: cho biết bộ nhận đang ngừng – lên 1 khi phát hiện bit stop.

		RCIDL = 0: cho biết bắt đầu nhận bit start cho đến bit dữ liệu cuối cùng. Ở chế độ đồng bộ: không có tác dụng.
BAUDCTL.4	SCKP	Synchronous Clock Polarity bit: Ở chế độ không đồng bộ: SCKP = 1: phát dữ liệu đảo đến chân TX/CK. SCKP = 0: phát dữ liệu không đảo đến chân TX/CK. Ở chế độ đồng bộ: SCKP = 1: dữ liệu được dịch khi có cạnh lên của xung clock. SCKP = 0: dữ liệu được dịch khi có cạnh xuống của xung clock.
BAUDCTL.3	BRG16	16 bit Baud Rate generator bit BRG16 = 1: sử dụng bộ tạo tốc độ baud 16 bit. BRG16 = 0: sử dụng bộ tạo tốc độ baud 8 bit.
BAUDCTL.1	WUE	Wake – up Enable bit: bit cho phép đánh thức CPU Ở chế độ không đồng bộ: WUE = 1: cho phép bộ nhận chờ cho đến khi có xung cạnh xuống. Sẽ không có kí tự nào được nhận nếu bit RCIF thiết lập ở mức 1. WUE sẽ tự động xóa sau khi bit RCIF lên 1. WUE = 0: Bộ nhận hoạt động bình thường. Ở chế độ đồng bộ: không có chức năng.
BAUDCTL.0	ABDEN	Auto-Baud Detect Enable bit: bit cho phép phát hiện tốc độ baud tự động. ABDEN = 1: cho phép hoạt động tốc độ baud tự động. ABDEN = 0: không cho phép hoạt động tốc độ baud tự động

Bộ phát tốc độ baud (Baud Rate Generator) là timer 8 bit hoặc 16 bit để tạo tốc độ cho hoạt động truyền dữ liệu ESUART đồng bộ và bất đồng bộ.

Mặc nhiên thì khôi tạo tốc độ baud BRG hoạt động chế độ 8 bit, nếu cho bit BRG16 bằng 1 thì hoạt động chế độ 16 bit.

Cặp thanh ghi SPBRGH và SPBRG xác định chu kỳ tạo tốc độ baud của timer.

Trong chế độ truyền đồng bộ thì bộ nhân chu kỳ tốc độ baud được xác định bởi cả 2 bit BRGH trong thanh ghi TXSTA và bit BRG16 trong thanh ghi BAUDCTL.

Trong chế độ truyền bất đồng bộ thì bộ nhân chu kỳ tốc độ baud chỉ được xác định bởi bit BRGH.

Bảng 10-4. Tóm tắt các công thức tính tốc độ baud.

Configuration Bits			BRG/EUSART Mode	Baud Rate Formula
SYNC	BRG16	BRGH		
0	0	0	8-bit/Asynchronous	FOSC/[64 (n+1)]
0	0	1	8-bit/Asynchronous	FOSC/[16 (n+1)]
0	1	0	16-bit/Asynchronous	
0	1	1	16-bit/Asynchronous	
1	0	x	8-bit/Synchronous	FOSC/[4 (n+1)]
1	1	x	16-bit/Synchronous	

Ứng với từng trạng thái của 3 bit SYNC, BRG16 và BRGH mà có các công thức tính tốc độ baud tương ứng. Trong công thức tính tốc độ baud thì n là giá trị của cặp thanh ghi SPBRGH và SPBRG.

Ví dụ 10-1: Hãy tính giá trị n của cặp thanh ghi để tốc độ truyền là 9600BAUD, sử dụng thạch anh có tần số 16MHz, hoạt động bất đồng bộ, bộ phát tốc độ BRG 8 bit.

$$\text{Công thức tính tốc độ baud: } TDTDL = TDBAUD = \frac{F_{osc}}{64(n+1)} = \frac{F_{osc}}{64([SPBGRH : SPBRG] + 1)}$$

Tính giá trị của cặp thanh ghi:

$$SPBGRH : SPBRG = \frac{F_{osc}}{64 \times TDBAUD} - 1 = \frac{16000000}{64 \times 9600} - 1 = 25,041$$

Giá trị để nạp cho cặp thanh ghi là 25, khi đó tốc độ thực là:

$$TDBAUD_{TINH} = \frac{F_{osc}}{64([SPBGRH : SPBRG] + 1)} = \frac{16000000}{64(25+1)} = 9615$$

$$\text{Sai số so với tốc độ 9600 là: } SAISO = \frac{TDBAUD_{TINH} - TDBAUD}{TDBAUD} = \frac{9615 - 9600}{9600} = 0,16\%$$

10.5 CÁC LỆNH TRUYỀN DỮ LIỆU EUSART CỦA PIC16F887

Các lệnh của ngôn ngữ lập trình C liên quan đến ngắt bao gồm:

10.5.1 LỆNH SETUP_UART(BAUD, STREAM)

Cú pháp: `setup_uart(baud, stream)`, `setup_uart(baud)`, `set_uart_speed(baud,[stream])`

Thông số: **baud, stream** có thể là 1 hoặc 2 hằng số: hằng số thứ nhất là tốc độ, hằng số thứ 2 là tên của cổng giao tiếp nếu hệ thống có nhiều cổng giao tiếp EUART để phân biệt. Nếu chỉ có 1 thì không cần thông số thứ 2.

Chức năng: thiết lập tốc độ baud cho EUART.

Có hiệu lực: cho tất cả các vi điều khiển PIC.

Ví dụ 10-2: thiết lập tốc độ 9600 baud: `setup_uart(9600);`

10.5.2 LỆNH PUTS(STRING)

Cú pháp: `puts(string)`

Thông số: **string** là chuỗi ký tự gửi đi.

Chức năng: có chức năng gửi từng ký tự ra port nối tiếp, sau khi **gửi xong** chuỗi ký tự thì **tự động gửi thêm ký tự** RETURN (có mã là 13) và ký tự xuống hàng (LINE-FEED có mã là 10).

Có hiệu lực: cho tất cả các vi điều khiển PIC.

Ví dụ 10-3: `puts(" | HELLO| ");`

10.5.3 LỆNH Value = Getc(), Value = Fgetc(Stream), Value = Getch(), Value = Getchar()

Cú pháp: `value = getc()`, `value = fgetc(stream)`, `value = getch()`, `value = getchar()`

Thông số: **stream** là tên **của cổng EUART** nếu có.

Chức năng: có chức năng **chờ cho đến khi có ký tự gửi đến thì nhận**.

Có hiệu lực: cho tất cả các vi điều khiển PIC.

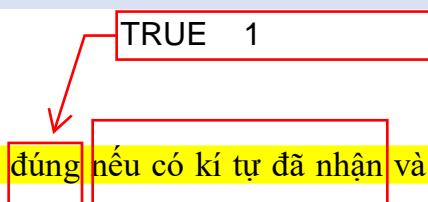
Ví dụ 10-4: `kitu = getch();`

10.5.4 LỆNH value = KBHIT()

Cú pháp: `value = KBHIT()`

Thông số: **không có**

Chức năng: **có chức năng trả về kết quả đúng** nếu có ký tự đã nhận và trả về kết quả sai nếu không có ký tự.



Có hiệu lực: cho tất cả các vi điều khiển PIC.

Ví dụ 10-5: if (KBHIT) kitu = getch(); // nếu có kí tự thì nhận

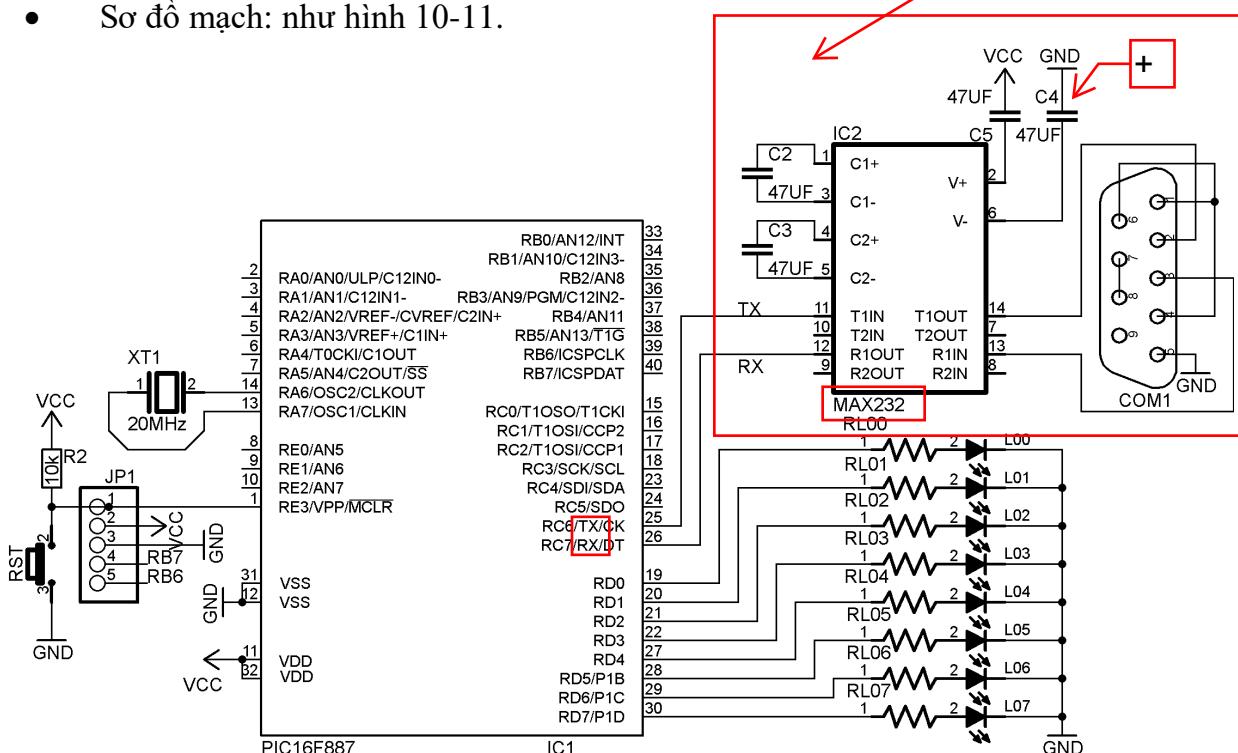
10.6 ỨNG DỤNG TRUYỀN DỮ LIỆU EUSART CỦA PIC16F887

Phần này trình bày các ứng dụng truyền dữ liệu đơn giản của PIC16F887, qua các ứng dụng này giúp bạn biết sử dụng chức năng truyền dữ liệu của PIC16F887.

10.6.1 TRUYỀN DỮ LIỆU GIỮA PIC16F887 VÀ MÁY TÍNH ĐIỀU KHIỂN LED

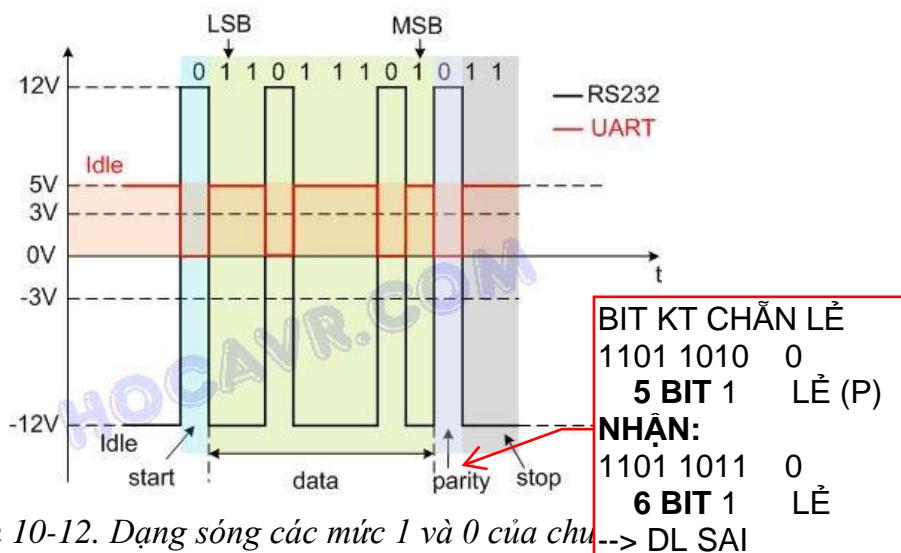
Bài 10-1: Một hệ thống truyền dữ liệu giữa máy tính và vi điều khiển PIC 16F887: máy tính sẽ gửi dữ liệu xuống vi điều khiển, vi điều khiển sẽ nhận dữ liệu và gửi ra portD nối với 8 led đơn. Máy tính sử dụng phần mềm Terminal.

- Sơ đồ mạch: như hình 10-11.



Hình 10-11. Hệ thống truyền dữ liệu giữa máy tính và vi điều khiển.

Trong sơ đồ mạch có dùng IC2 có tên là MAX232 dùng để chuyển đổi tín hiệu RS232 của cổng COM của máy tính thành chuẩn TTL để tương thích với vi điều khiển, xem hình 10-12.



Hình 10-12. Dạng sóng các mức 1 và 0 của chuỗi DI SA

Mức logic 0 của chuẩn RS232 có điện áp 12V, mức logic 1 có điện áp -12V, chuẩn này khác với chuẩn TTL nên phải chuyển đổi.

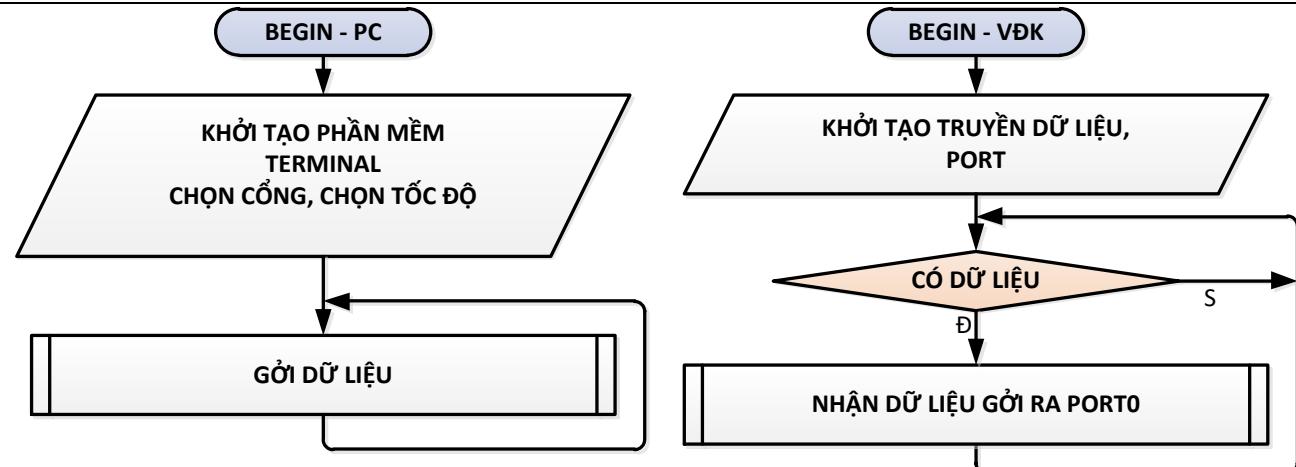
IC MAX232 có tích hợp 2 bộ chuyển đổi, trong mạch này chỉ dùng 1 bộ chuyển đổi. Các tụ điện trong mạch theo sở tay của IC do nhà chế tạo cung cấp.

Truyền dữ liệu UART thường dùng cổng giao tiếp là 9 chân có tên là COM1 (trước dùng cổng 25 chân nhưng đã bỏ). Chú ý khi vẽ mạch in (PCB) thì bạn phải chọn cổng COM 9 chân loại hàn vào mạch in và khi chọn linh kiện bạn phải chọn đúng loại cái. Các dây cáp kết nối sử dụng đầu jack đặc nên sẽ giúp bạn kết nối đúng.

Sử dụng portD để điều khiển 8 led đơn, thạch anh sử dụng là 20Mhz.

- Lưu đồ: như hình 10-13.
- Chương trình: cho vi điều khiển nhận dữ liệu

```
#INCLUDE      <TV_16F887.C>
#USE         rs232 (baud=9600,xmit=pin_c6,rcv=pin_c7)
UNSIGNED INT8 RDATA=0;
VOID MAIN()
{
    SET_TRIS_D (0x00);      OUTPUT_D (RDATA);
    WHILE (TRUE)
    {
        IF (KBHIT ())
        {
            RDATA = GETCH ();
            OUTPUT_D (RDATA);
        }
    }
}
```



Hình 10-13. Lưu đồ điều khiển truyền dữ liệu.

- Giải thích chương trình:

Hàng thứ 1 là khai báo thư viện sử dụng giống các bài đã viết.

Hàng thứ 2 khai báo thiết lập truyền dữ liệu gồm 3 thông số: chọn tốc độ truyền là 9600 baud, chọn port phát dữ liệu là chân thứ 6 của portC, chọn port nhận dữ liệu là chân thứ 7 của portC.

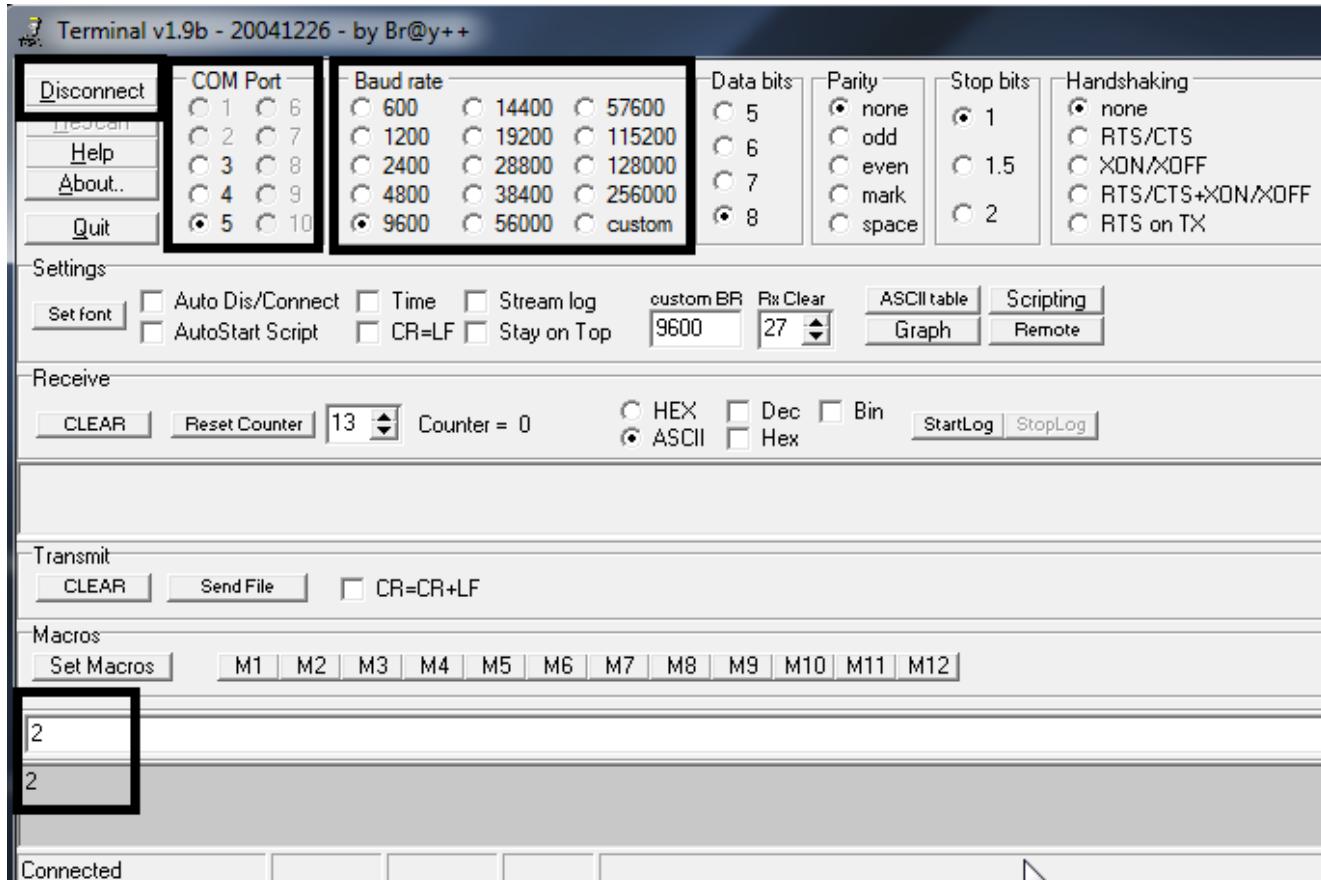
Theo sơ đồ chân thì 2 chân RC6 và RC7 là TD và RD nhưng bạn cũng có thể chọn chân khác để kết nối cũng được, khi đó bạn phải hiệu chỉnh mạch điện và cả chương trình.

Trong vòng lặp while thì tiến hành kiểm tra xem nếu có dữ liệu thì tiến hành nhận và xuất ra portD.

10.6.2 PHẦN MỀM TRUYỀN DỮ LIỆU TERMINAL TRÊN MÁY TÍNH

Để truyền dữ liệu từ máy tính xuống vi điều khiển thì ta phải tiến hành xây dựng 1 chương trình, tuy nhiên để đơn giản thì ta sử dụng phần mềm đã có sẵn là Terminal.

- Giao diện phần mềm Terminal như hình 10-14.



Hình 10-14. Giao diện phần mềm Terminal để gửi dữ liệu.

Bạn phải xem trong “Device Manager” để xem cổng COM đang sử dụng là COM thứ bao nhiêu và tiến hành chọn cho đúng cổng COM port, trong tài liệu này máy tính sử dụng cổng COM5.

Tốc độ mặc nhiên của phần mềm là 9600 baud.

Bạn tiến hành chọn mục “Connect”, sau khi kết nối thành công thì tên của nút nhấn trở thành “Disconnect”.

Hàng trắng bên dưới bạn đánh ký tự cần gởi và nhấn enter thì phần mềm sẽ gởi xuống máy tính. Trong hình thì đã gởi ký tự số 2.

Phần mềm Terminal gởi ký tự ở dạng mã ASCII. Khi gởi số 2 thì máy tính sẽ gởi mã ASCII của số 2 là 0x32, dữ liệu điều khiển portD sẽ là 00110010.

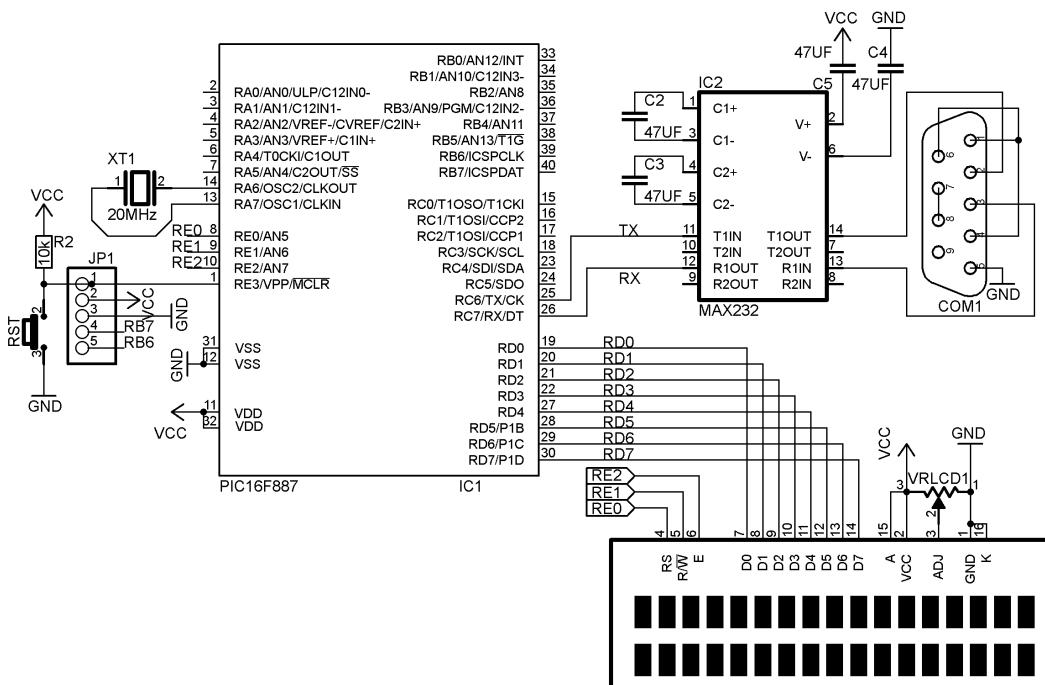
Mục đích của bài này cho các bạn thấy cách viết chương trình truyền dữ liệu khá đơn giản, các bài tiếp theo sẽ hiển thị được nhiều ký tự và rõ ràng hơn ví dụ bạn gởi số 2 trên máy tính thì bạn có thể nhìn thấy được số 2 hiển thị trên LCD bên vi điều khiển, bạn có thể gởi 1 chuỗi nhiều ký tự.

10.6.3 TRUYỀN DỮ LIỆU GIỮA PIC16F887 VÀ MÁY TÍNH HIỂN THỊ LCD

Bài 10-2: Một hệ thống truyền dữ liệu giữa máy tính và vi điều khiển PIC 16F887: máy tính sẽ gởi dữ liệu xuống vi điều khiển, vi điều khiển sẽ nhận dữ liệu và hiển thị trên LCD. Máy tính sử dụng phần mềm Terminal.

Tốc độ truyền là 9600 baud.

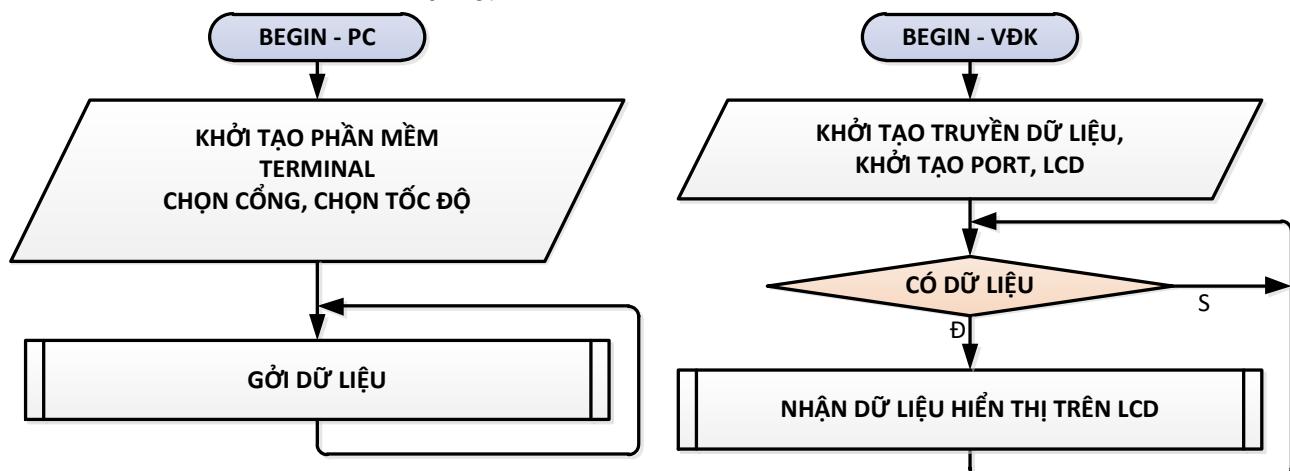
- Sơ đồ mạch: như hình 10-15.



Hình 10-15. Hệ thống truyền dữ liệu giữa máy tính và vi điều khiển, hiển thị LCD.

Sơ đồ mạch dùng LCD để hiển thị được mã ASCII để tương thích với máy tính.

- Lưu đồ: như hình 10-16.



Hình 10-16. Lưu đồ điều khiển truyền dữ liệu, hiển thị trên LCD.

- Chương trình: cho vi điều khiển nhận dữ liệu

```

#include <TV_16F887.C>
#include <TV_LCD.C>
#use rs232(baud=9600,xmit=pin_c6,rcv=pin_c7)
UNSIGNED INT8 RDATA=0,I;
const unsigned char HANG1[16]={"GIAO TIEP MAY TINH"};
VOID MAIN()
{
    SET_TRIS_E(0x00);
    LCD_SETUP();
    LCD_COMMAND(ADDR_LINE1);
    FOR (I=0; I<16; I++)
        LCD_COMMAND(ADDR_LINE2);
    WHILE (TRUE)
    {
        SET_TRIS_D(0x00);
        DELAY_US(10);
        { LCD_DATA(HANG1[I]); }
        DELAY_US(10);
    }
}
    
```

```

IF (KBHIT ()) {
{
    RDATA = GETCH ();
    LCD_DATA (RDATA);
}
}

```

- Giải thích chương trình:

Về cơ bản thì chương trình này chỉ khác là dùng LCD để hiển thị được các ký tự từ máy tính gửi xuống vì LCD hiển thị ký tự mã ASCII.

Do LCD sử dụng là 16x2 có thể hiển thị được 16 ký tự nên bạn có thể gửi được 16 ký tự hiển thị trên LCD cùng 1 lúc, nếu gửi nhiều hơn thì các ký tự thứ 17 sẽ không nhìn thấy.

Ở bài minh họa này ta đã gửi dữ liệu từ máy tính xuống hiển thị trên LCD nhưng chưa thực hiện gửi dữ liệu từ vi điều khiển lên máy tính. Bài tiếp theo ta sẽ minh họa cho cả 2 chiều.

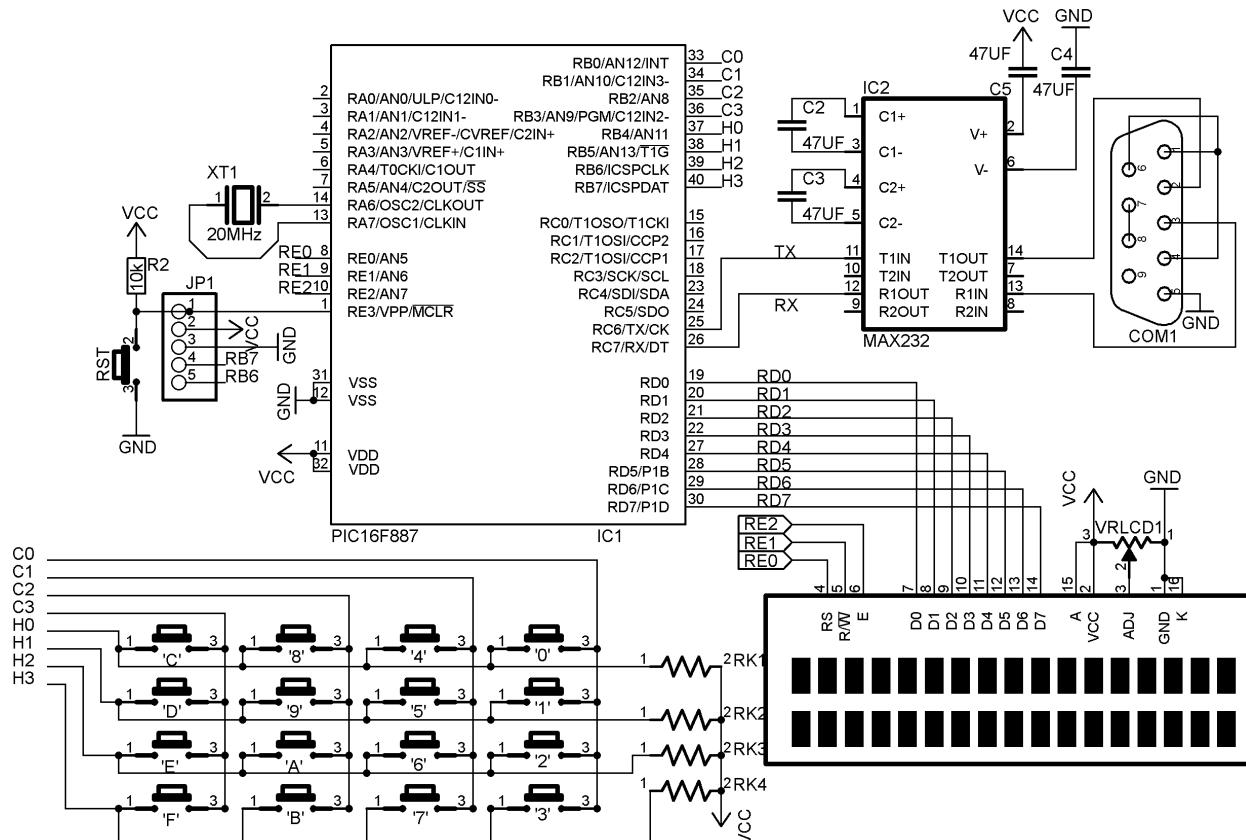
10.6.4 TRUYỀN VÀ NHẬN DỮ LIỆU GIỮA PIC16F887 VÀ MÁY TÍNH

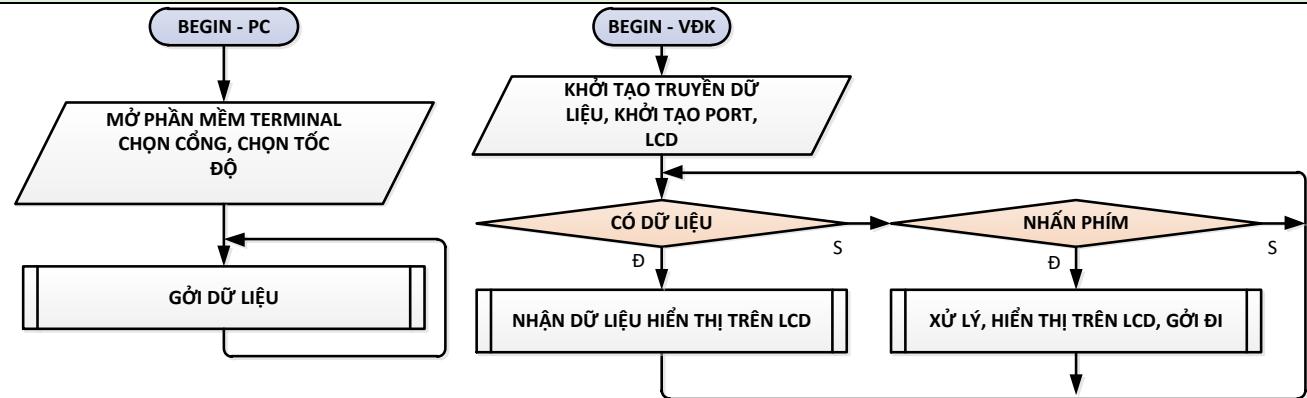
Bài 10-3: Một hệ thống truyền dữ liệu giữa máy tính và vi điều khiển PIC 16F887: máy tính sẽ gửi dữ liệu xuống vi điều khiển, vi điều khiển sẽ nhận dữ liệu và hiển thị trên LCD ở hàng 2. Khi nhấn 1 phím bất kỳ của bàn phím ma trận 4x4 thì mã phím đó hiển thị trên LCD ở hàng 1, đồng thời gửi về máy tính.

Máy tính sử dụng phần mềm Terminal.

Tốc độ truyền là 9600 baud.

- Sơ đồ mạch: như hình 10-17.





Hình 10-18. Lưu đồ điều khiển truyền dữ liệu, có thêm LCD và bàn phím.

- Chương trình:

```

#include <TV_16F887.C>
#include <TV_LCD.C>
#include <TV_KEY4X4.C>
#use rs232(baud=9600,xmit=pin c6,rcv=pin c7)
UNSIGNED INT8 RDATA=0, MP, TDATA, VITRI_H1=0, VITRI_H2=0;
VOID MAIN()
{
    SET_TRIS_E(0x00);           SET_TRIS_D(0x00);
    SET_TRIS_B(0xF0);          PORT_B_PULLUPS(0XF0);
    LCD_SETUP();
    LCD_COMMAND(ADDR_LINE1);   DELAY_US(10);
    LCD_DATA("GIAO TIEP MAY TINH");
    WHILE (TRUE)
    {
        IF (KBHIT())
        {
            LCD_COMMAND(ADDR_LINE2+VITRI_H2);
            DELAY_US(10);
            RDATA = GETCH();
            LCD_DATA(RDATA);
            IF (VITRI_H2==15)    VITRI_H2=0;
            ELSE                  VITRI_H2++;
        }
        MP= KEY_4X4();
        IF (MP!=0xFF)
        {
            LCD_COMMAND(ADDR_LINE1+VITRI_H1);
            DELAY_US(10);
            IF (MP<10)           TDATA = MP+0X30;
            ELSE                  TDATA = MP+0X37;
            PUTC(TDATA);
            LCD_DATA(TDATA);
            IF (VITRI_H1==15)    VITRI_H1=0;
            ELSE                  VITRI_H1++;
        }
    }
}
  
```

- Giải thích chương trình:

Chương trình chính thực hiện 2 yêu cầu:

Yêu cầu thứ 1: Kiểm tra nếu có dữ liệu từ máy tính gửi xuống thì tiến hành hiển thị ở hàng 2.

Yêu cầu thứ 2: Gọi chương trình con quét ma trận phím, nếu có phím nhấn thì tiến hành kiểm tra nếu các phím số có mã từ 0 đến 9 thì chuyển sang mã ASCII bằng cách cộng thêm 0x30 rồi hiển thị và gởi về máy tính, nếu là các phím có mã từ 0x0A đến 0x0F thì cộng thêm 0x37 để chuyển thành các ký tự tương ứng từ A đến F để hiển thị trên LCD và gởi về máy tính.

Do mỗi hàng chỉ hiển thị được 16 ký tự và 2 hàng độc lập về địa chỉ nên phải thiết lập địa chỉ trước khi gửi dữ liệu, khi gửi 1 ký tự ra LCD thì địa chỉ tăng lên 1, khi bằng 16 thì cho địa chỉ về 0 để về lại đầu hàng.

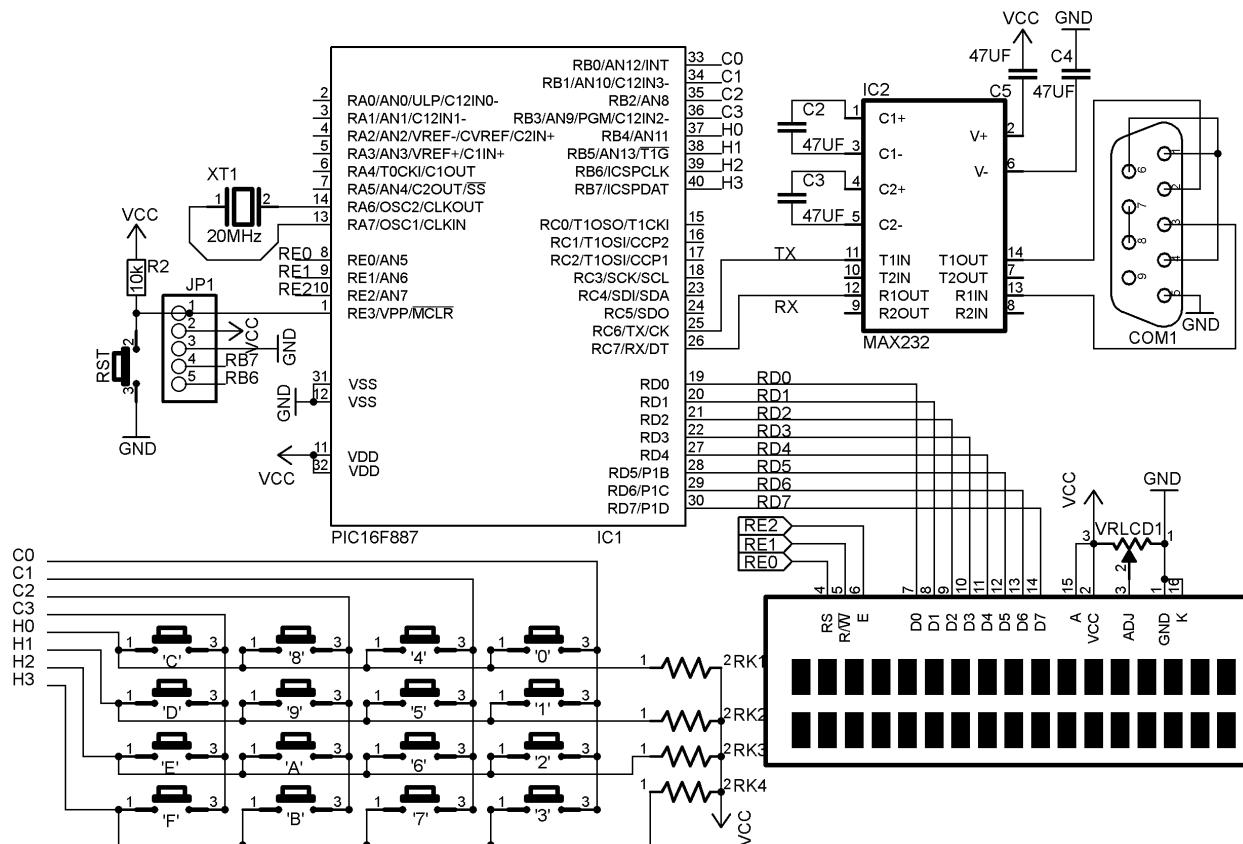
10.6.5 TRUYỀN VÀ NHẬN DỮ LIỆU GIỮA PIC16F887 VÀ MÁY TÍNH DÙNG NGẮT ĐỂ NHẬN DỮ LIỆU

Bài 10-4: Một hệ thống truyền dữ liệu giữa máy tính và vi điều khiển PIC 16F887: máy tính sẽ gửi dữ liệu xuống vi điều khiển, vi điều khiển sẽ nhận dữ liệu và hiển thị trên LCD ở hàng 2. Khi nhấn 1 phím bất kỳ của bàn phím ma trận 4x4 thì mã phím đó hiển thị trên LCD ở hàng 1, đồng thời gửi về máy tính.

Máy tính sử dụng phần mềm Terminal.

Tốc độ truyền là 9600 baud. **Sử dụng ngắt để nhận dữ liệu.**

- Sơ đồ mạch: như hình 10-19.



Hình 10-19. Hệ thống truyền dữ liệu giữa máy tính và vi điều khiển, có LCD, bàn phím.

Sơ đồ mạch thêm phần giao tiếp bàn phím ma trận 4x4 để thực hiện chức năng khi nhấn phím nào thì mã phím đó sẽ hiển thị trên LCD đồng thời gửi về máy tính.

- Lưu đồ: như hình 10-20.
 - Chương trình:

```
#INCLUDE      <TV_16F887.C>
#INCLUDE      <TV_LCD.C>
```

```
#INCLUDE      <TV_KEY4X4.C>
#USE          rs232(baud=9600,xmit=pin_c6,rcv=pin_c7)
UNSIGNED INT8 RDATA=0, MP, I, TDATA, VITRI_H1=0, VITRI_H2=0;
const unsigned char HANG1[16]={"GIAO TIEP MAY TINH"};
```

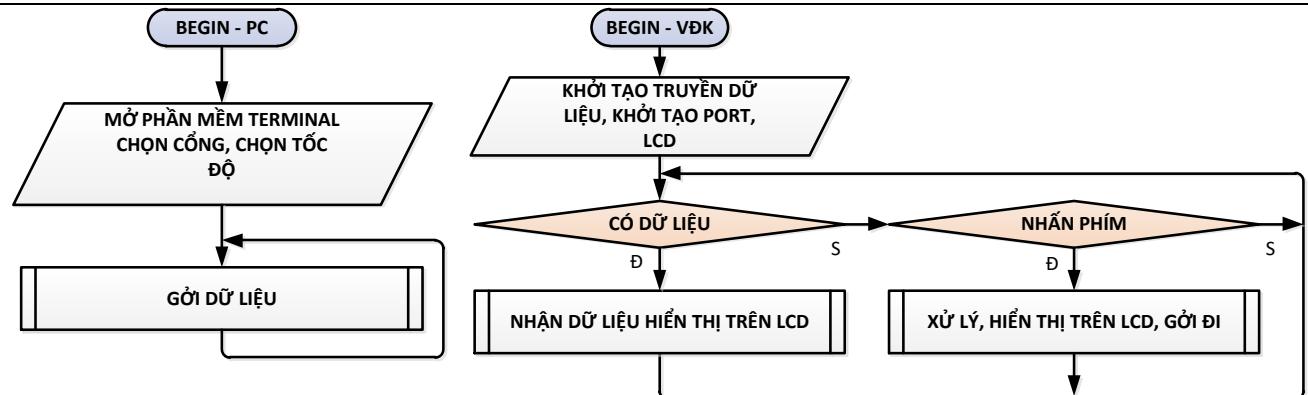
```
#INT_RDA
void interrupt_UART()
{
    IF (KBHIT())
    {
        LCD_COMMAND (ADDR_LINE2+VITRI_H2);
        DELAY_US(10);
        RDATA = GETCH();
        LCD_DATA(RDATA);
        IF (VITRI_H2==16) VITRI_H2=0;
        ELSE VITRI_H2++;
    }
}
```

kiểm tra có dữ liệu hay không
có thể bỏ --> vì khi xảy ra ngắt
thì chắc chắn là có dữ liệu

nhận dữ liệu

```
VOID MAIN()
{
    SET_TRIS_E(0x00);           SET_TRIS_D(0x00);
    SET_TRIS_B(0xF0);           PORT_B_PULLUPS(0XF0);
    ENABLE_INTERRUPTS(GLOBAL);
    ENABLE_INTERRUPTS(INT_RDA);

    LCD_SETUP();
    LCD_COMMAND(ADDR_LINE1);    DELAY_US(10);
    FOR (I=0;I<16;I++)         { LCD_DATA(HANG1[I]);}
    WHILE (TRUE)
    {
        MP= KEY_4X4();
        IF (MP!=0xFF)
        {
            LCD_COMMAND(ADDR_LINE1+VITRI_H1);
            DELAY_US(10);
            IF (MP<10)          TDATA = MP+0X30;
            ELSE                 TDATA = MP+0X37;
            PUTC(TDATA);
            LCD_DATA(TDATA);
            IF (VITRI_H1==16)   VITRI_H1=0;
            ELSE                 VITRI_H1++;
        }
    }
}
```



Hình 10-20. Lưu đồ điều khiển truyền dữ liệu, có thêm LCD và bàn phím.

- Giải thích chương trình:

Chương trình chính thực hiện 2 yêu cầu:

Yêu cầu thứ 1: Kiểm tra nếu có dữ liệu từ máy tính gửi xuống thì tiến hành hiển thị ở hàng 2.

Yêu cầu thứ 2: Gọi chương trình con quét ma trận phím, nếu có phím nhấn thì tiến hành kiểm tra nếu các phím số có mã từ 0 đến 9 thì chuyển sang mã ASCII bằng cách cộng thêm 0x30 rồi hiển thị và gửi về máy tính, nếu là các phím có mã từ 0x0A đến 0x0F thì cộng thêm 0x37 để chuyển thành các ký tự tương ứng từ A đến F để hiển thị trên LCD và gửi về máy tính.

Do mỗi hàng chỉ hiển thị được 16 ký tự và 2 hàng độc lập về địa chỉ nên phải thiết lập địa chỉ trước khi gửi dữ liệu, khi gửi 1 ký tự ra LCD thì địa chỉ tăng lên 1, khi bằng 16 thì cho địa chỉ về 0 để về lại đầu hàng.

Bài tập 10-1: Hãy hiệu chỉnh chương trình bài 10-3 dùng ngắt để nhận dữ liệu.

Bài tập 10-2: Một hệ thống điều khiển dùng 2 vi điều khiển PIC 16F887: vi điều khiển A có portD nối với 8 led đơn, vi điều khiển B cũng có portD nối với 8 led. Dữ liệu 8 bit từ vi điều khiển A điều khiển portD sáng dần rồi tắt dần hai chiều, đồng thời gửi sang vi điều khiển B để hiển thị ra 8 led.

Hai vi điều khiển giao tiếp bất đồng bộ tốc độ 9600 baud.

Hãy vẽ sơ đồ mạch, viết lưu đồ và chương trình.

Bài tập 10-3: Một hệ thống điều khiển dùng 2 vi điều khiển PIC16F887: vi điều khiển A kết nối với 2 nút nhấn ON và OFF, vi điều khiển B nối với 8 led. Khi nhấn ON thì vi điều khiển A gửi dữ liệu sang vi điều khiển B làm 8 led đơn sáng, khi nhấn OFF thì gửi mã làm 8 led tắt.

Hai vi điều khiển giao tiếp bất đồng bộ tốc độ 9600 baud.

Hãy vẽ sơ đồ mạch, viết lưu đồ và chương trình.

10.7 CÂU HỎI ÔN TẬP – TRẮC NGHIỆM

10.6.1 CÂU HỎI ÔN TẬP

Câu 10-1: Hãy cho biết tên các thanh ghi liên quan đến truyền dữ liệu của vi điều khiển PIC16F887.

Câu 10-2: Hãy cho biết chức năng của các bit trong thanh ghi TXSTA của vi điều khiển PIC16F887.

Câu 10-3: Hãy cho biết chức năng của các bit trong thanh ghi RCSTA của vi điều khiển PIC16F887.

Câu 10-4: Hãy cho biết chức năng của các bit trong thanh ghi BAUDCTL của vi điều khiển PIC16F887.

10.6.2 CÂU HỎI MỞ RỘNG

Câu 10-5: Hãy tìm hiểu truyền dữ liệu của vi điều khiển PIC18F4550 và so sánh với PIC16F887.

10.6.3 CÂU HỎI TRẮC NGHIỆM

Câu 10-1: Vi điều khiển PIC 16F887 có mấy chế độ truyền dữ liệu ESUART:

(a) 2

(b) 3

(c) 4

(d) 5

Câu 10-2: Vì điều khiển PIC 16F887 khi truyền đồng bộ thì các tín hiệu có tên là:

- (a) RxD, TxD (b) RxD, CK (c) TD, CK (d) TD, TK

Câu 10-3: Vì điều khiển PIC 16F887 khi truyền bất đồng bộ thì các tín hiệu có tên là:

- (a) RxD, TxD (b) RxD, CK (c) TD, CK (d) TD, TK

Câu 10-4: Vì điều khiển PIC 16F887 thì thanh ghi nào thiết lập tốc độ truyền dữ liệu:

- (a) TXSTA (b) RCSTA (c) BAUDCTL (d) TMOD

Câu 10-5: Trong công thức tính tốc độ của PIC16F887 thì biến “Fosc” là:

- | | |
|----------------------------------|--------------------------------------|
| (a) Tần số dao động của timer T1 | (b) Tần số dao động của tụ thạch anh |
| (c) Tần số dao động của timer T0 | (d) Tần số dao động của ADC |

Câu 10-21: Cặp thanh ghi nào của PIC16F887 lưu giá trị để tạo tốc độ baud:

- | | |
|---------------------|--------------------|
| (a) TXSTA, RCSTA | (b) BAUDCTL, RCSTA |
| (c) TSPBGRH, SPBRGL | (d) TSPBGRH, SPBRG |

Câu 10-22: Vì điều khiển PIC 16F887 sử dụng timer nào để thiết lập tốc độ truyền dữ liệu:

- | | |
|--------|----------------------|
| (a) T0 | (b) T1 |
| (c) T2 | (d) Không dùng timer |

Câu 10-23: Bit cho phép ngắt và cờ ngắt nhận dữ liệu của PIC 16F887:

- | | |
|----------------|----------------|
| (a) RCIF, TXIF | (b) TXIE, TXIF |
| (c) RCIE, RCIF | (d) TXIE, RCIE |

Câu 10-24: Bit cho phép ngắt và cờ ngắt phát dữ liệu của PIC 16F887:

- | | |
|----------------|----------------|
| (a) RCIF, TXIF | (b) TXIE, TXIF |
| (c) RCIE, RCIF | (d) TXIE, RCIE |

10.6.4 BÀI TẬP

TÀI LIỆU THAM KHẢO

- [1]. MICROCHIP – PIC 16F87X DATASHEET, 1997
- [2]. MICROCHIP – PICmicro Mid- Range MCU family reference Manual, 1997
- [3]. AVTAR SINGH – WALTER A TRIEBEL, “The 8088 Microprocessor – Programming, interfacing, software, hardware, and Applications”, Prentice Hall International Editions.
- [4]. DOUGLAS V. HALL, “Microprocessor and Interfacing Programming, and hardware”, McGraw – Hill International Editions.
- [5]. John Uffenbeck, “The 8088/8086 family : Designing, programming and interfacing”, Prentice Hall, 1987
- [6]. James L. Antonakos, “The 68000 Microprocessor: hardware and software principles and applications”, Prentice Hall fifth edition 2004.
- [7]. Jack L. Davies, “The Innovative 80x86 – Volume I: the 80286 Microprocessor, architecture”, Prentice Hall.
- [8]. Jack L. Davies, “Z80 Family CPU user manual”, www.zilog.com.
- [9]. MetaLink Corporation Chandler – Arizona, “8051 Cross Assembler User’s Manual”, 1996
- [10]. “MCS51 Microcontroller Family User’s Manual”, 1994
- [11]. “M68332 User’s Manual”
- [12]. Hồ Trung Mỹ, Vi Xử Lý, Nhà xuất bản Đại Học Quốc Gia Tp HCM
- [13]. Ngô Diên Tập, Kỹ Thuật AVR, Nhà xuất bản KH & KT – 2003.
- [14]. Văn Thế Minh, Kỹ Thuật vi xử lý, Nhà xuất bản giáo dục – 1997.
- [15]. Các tài liệu của các vi điều khiển ATMEL, MICROCHIP.