**BỘ LỆNH VÀ CÁC CHƯƠNG TRÌNH ỨNG DỤNG CỦA PLC OMRON**

# Mục lục

[Mục lục iii](#_Toc235251570)

[Danh sách các hình x](#_Toc235251571)

[Danh sách các bảng xiii](#_Toc235251572)

[CHUƠNG 1. GIỚI THIỆU PLC CP1L 2](#_Toc235251574)

[1.1. Giới Thiệu 2](#_Toc235251575)

[1.2. Các Tính Năng Của CP1L Các tính năng chính của CP1L bao gồm: 2](#_Toc235251576)

[1.3. Cách Kết Nối Vào Ra 3](#_Toc235251577)

[Bảng 1.1. Các môđun CPU. 3](#_Toc235251578)

[1.4. Các Ứng Dụng Của PLC CP1L 4](#_Toc235251579)

[1.4.1. PLC OMRON tiêu chuẩn cùng với các tính năng thông minh 4](#_Toc235251580)

[1.4.2. Thuận lợi với nhiều ngôn ngữ lập trình 4](#_Toc235251581)

[1.4.3. Dễ dàng kết nối với USB 5](#_Toc235251582)

[1.4.4. Dễ dàng cho việc bảo dưỡng và điều chỉnh ban đầu nhờ vào màn hình LCD cho phép hiển thị và cài đặt 5](#_Toc235251583)

[1.4.5. Điều khiển chuyển động với độ chính xác cao 6](#_Toc235251584)

[1.4.6. Khả năng giao tiếp cùng với truyền thông nối tiếp 8](#_Toc235251585)

[1.4.7. Có khả năng lựa chọn cổng giao tiếp nối tiếp 8](#_Toc235251586)

[CHƯƠNG 2. GIỚI THIỆU PHẦN MỀM CX- PROGRAMMER 10](#_Toc235251587)

[2.1. Giới Thiệu 10](#_Toc235251588)

[2.2. Cách Sử Dụng Phần Mềm 10](#_Toc235251589)

[2.2.1.Tạo một project mới 10](#_Toc235251590)

[2.2.2. Các thành phần trong cửa sổ project 11](#_Toc235251591)

[2.2.3. Các thao tac trên cửa sổ project 12](#_Toc235251592)

[2.2.4. Mô phỏng với Cx- Simulator 17](#_Toc235251593)

[CHƯƠNG 3. BỘ LỆNH CỦA HÃNG OMRON 23](#_Toc235251594)

[3.1. Ladder Instructions 23](#_Toc235251595)

[3.1.1. AND - And 23](#_Toc235251596)

[3.1.2. AND LD - And Load 23](#_Toc235251597)

[3.1.3. AND NOT - And Not 24](#_Toc235251598)

[3.1.4. LD - Load 24](#_Toc235251599)

[3.1.5. LD NOT - Load Not 24](#_Toc235251600)

[3.1.6. OR - Or 24](#_Toc235251601)

[3.1.7. OR LD - OR Load 25](#_Toc235251602)

[3.1.8. OR NOT - Or Not 26](#_Toc235251603)

[3.1.9. OUT – Output 26](#_Toc235251604)

[3.2. Program Control Instructions 27](#_Toc235251605)

[3.2.1. END(01) - End 27](#_Toc235251606)

[3.2.2. IL(02) - Interlock 27](#_Toc235251607)

[ILC(03) - Interlock Clear 27](#_Toc235251608)

[3.2.3. JMP(04) - Jump 27](#_Toc235251609)

[JME(05) - Jump End 27](#_Toc235251610)

[3.2.4. STEP(08) - Step Define 29](#_Toc235251611)

[SNXT(09) - Step Start 29](#_Toc235251612)

[3.2.5. STOP(99) - Stop 30](#_Toc235251613)

[3.3. Bit Control Instructions 30](#_Toc235251614)

[3.3.1. DIFU(13) - Differentiate Up 30](#_Toc235251615)

[3.3.2. KEEP(11) - Keep 31](#_Toc235251616)

[3.3.3-RSET - Reset 32](#_Toc235251617)

[3.3.4. SET - Set 32](#_Toc235251618)

[3.3.5. TST(350) & TSTN(351) - Bit Test 32](#_Toc235251619)

[3.4. Timer/Counter Instructions 33](#_Toc235251620)

[3.4.1. CNT - Counter 33](#_Toc235251621)

[3.4.2. CNTR(12) - Reversible Counter 33](#_Toc235251622)

[3.4.3. CTBL(--) - Register Comparison Table 34](#_Toc235251623)

[3.4.4. INI(--) - Mode Control 35](#_Toc235251624)

[3.4.5. PRV(--) - High-Speed Counter PV Read 35](#_Toc235251625)

[3.4.6. STIM(--) - Interval Timer 36](#_Toc235251626)

[3.4.7. TIM - Timer 37](#_Toc235251627)

[3.4.8. TIML(--) - Long Timer 38](#_Toc235251628)

[3.4.9. TIMH(15) - High-Speed Timer 38](#_Toc235251629)

[3.4.10. TMHH(--) - Very High Speed Timer 39](#_Toc235251630)

[3.5. Subroutine Instructions 39](#_Toc235251631)

[3.5.1. CMCR(--) - PCMCIA Card Macro 39](#_Toc235251632)

[3.5.2. MCRO(99) - Macro 39](#_Toc235251633)

[Bảng 3.1. I/O words khác nhau cho mỗi PLC. 40](#_Toc235251634)

[3.5.3. PMCR(--) - Protocol Macro 40](#_Toc235251635)

[3.5.4. RET(93) - Return 41](#_Toc235251636)

[SBN(92) - Subroutine Define 41](#_Toc235251637)

[3.5.5. SBS(91) - Subroutine Enter 41](#_Toc235251638)

[3.6. Data Shifting Instructions 42](#_Toc235251639)

[3. 6.1. ASFT(--) - Asynchronous Shift Register 42](#_Toc235251640)

[3.6.2. ASL(25) - Arithmetic Shift Left 44](#_Toc235251641)

[3.6.3. ASR(26) - Arithmetic Shift Right 44](#_Toc235251642)

[3.6.4. ROL(27) - Rotate Left 45](#_Toc235251643)

[3.6.5. ROR(28) - Rotate Right 45](#_Toc235251644)

[3.6.6. SFT(10) - Shift Register 46](#_Toc235251645)

[3.6.7. SFTR(84) - Reversible Shift Register 47](#_Toc235251646)

[3.6.8. SLD(74) - One Digit Shift Left 50](#_Toc235251647)

[3.6.9. SRD(75) - One Digit Shift Right 50](#_Toc235251648)

[3.6.10. WSFT(16) - Word Shift 51](#_Toc235251649)

[3.7. Data Movement Instructions 52](#_Toc235251650)

[3.7.1. BSET(71) - Block Set 52](#_Toc235251651)

[3.7.2. BXF2(--) - EM Bank Transfer 53](#_Toc235251652)

[3.7.3. BXFR(125) - Interbank Block Transfer 53](#_Toc235251653)

[3.7.4. COLL(81) - Data Collect 54](#_Toc235251654)

[3.7.5. MOV(21) - Move 55](#_Toc235251655)

[3.7.6. MOVB(82) - Move Bit 56](#_Toc235251656)

[3.7.7. MOVD(83) - Move Digit 57](#_Toc235251657)

[3.7.8. MVN(22) - Move NOT 58](#_Toc235251658)

[3.7.9. XCHG(73) - Data Exchange 59](#_Toc235251659)

[3.7.10. XDMR(--) - Expansion DM Read 59](#_Toc235251660)

[3.7.11-XFER(70) - Block Transfer 60](#_Toc235251661)

[3.7.12. XFRB(--) - Transfer Bits 61](#_Toc235251662)

[3.7.13. XFR2(--) – EM Block Transfer 62](#_Toc235251663)

[3.8. Data Comparison Instructions 62](#_Toc235251664)

[3.8.1. BCMP(68) - Block Compare 62](#_Toc235251665)

[Bảng 3.2. Mô tả cách thực hiện lệnh so sánh khối. 63](#_Toc235251666)

[3.8.2. CMP(20) - Compare 64](#_Toc235251667)

[3.8.3. CMPL(60) - Double Compare 64](#_Toc235251668)

[3.8.4. CPS(--) - Signed Binary Compare 65](#_Toc235251669)

[3.8.5. CPSL(--) - Double Signed Binary Compare 65](#_Toc235251670)

[3.8.6. MCMP(19) - Multi-Word Compare 65](#_Toc235251671)

[3.8.7. SRCH(--) - Data Search 66](#_Toc235251672)

[3.8.8. TCMP(85) - Table Compare 67](#_Toc235251673)

[3.8.9. ZCP(--) - Area Range Control 68](#_Toc235251674)

[Bảng 3.3. Cờ kết quả 68](#_Toc235251675)

[3.8.10. ZCPL(--) - Double Area Range Compare 68](#_Toc235251676)

[Bảng 3.4. Cờ kết quả. 69](#_Toc235251677)

[3.9. Data conversion instruction 69](#_Toc235251678)

[3.9.1. ASC(86) - ASCII Convert 69](#_Toc235251679)

[3.9.2. BCD(24) - Binary-To-BCD 70](#_Toc235251680)

[3.9.3. BCDL(59) - Double Binary-To-Double BCD 70](#_Toc235251681)

[3.9.4. BCNT(67) - Bit Counter 71](#_Toc235251682)

[3.9.5. BIN(23) - BCD-To-Binary 71](#_Toc235251683)

[3.9.6. BINL(58) - Double BCD to Double Binary 71](#_Toc235251684)

[3.9.7. COLM(64) - Line-To-Column 72](#_Toc235251685)

[3.9.8. CTW(--) - Column-to-Word 73](#_Toc235251686)

[3.9.9. DMPX(77) - 16-To-4 Encoder 73](#_Toc235251687)

[3.9.10. HEX(--) - ASCII-To-Hexadecimal 74](#_Toc235251688)

[3.9.11. LINE(--) - Column-To-Line 75](#_Toc235251689)

[3.9.12. SDEC(78) - 7-Segment Decoder 75](#_Toc235251690)

[3.9.13. WTC(64) - Word-to-Column 76](#_Toc235251691)

[3.9.14. MLPX(76) - 4-To-16 Decoder 76](#_Toc235251692)

[3.10. BCD Calculation instructions 77](#_Toc235251693)

[3.10.1. ADD(30) - BCD Add 77](#_Toc235251694)

[3.10.2. ADDL(54) - Double BCD Add 79](#_Toc235251695)

[3.10.3. DEC(39) - Decrement 79](#_Toc235251696)

[3.10.4. DIV(33) - BCD Divide 80](#_Toc235251697)

[3.10.5. DIVL(57) - Double BCD Divide 81](#_Toc235251698)

[3.10.6. INC(38) - Increment 81](#_Toc235251699)

[3.10.7. MUL(32) - BCD Multiply 82](#_Toc235251700)

[3.10.8. MULL(56) - Double BCD Multiply 83](#_Toc235251701)

[3.10.9. SUB(31) - BCD Subtract 83](#_Toc235251702)

[3.10.10. SUBL(55) - Double BCD Subtract 84](#_Toc235251703)

[3.11. Binary Calculation Instructions 85](#_Toc235251704)

[3.11.1. ADB(50) - Binary Add 85](#_Toc235251705)

[3.11.2. ADBL(--) - Double Binary Add 86](#_Toc235251706)

[3.11.3. DBS(--) - Signed Binary Divide 87](#_Toc235251707)

[3.11.4-DBSL(--) - Double Signed Binary Divide 87](#_Toc235251708)

[3.11.5. DVB(53) - Binary Divide 88](#_Toc235251709)

[3.11.6. MBS(--) - Signed Binary Multiply 89](#_Toc235251710)

[3.11.7. MBSL(--) - Double Signed Binary Multiply 89](#_Toc235251711)

[3.11.8. MLB(52) - Binary Multiply 90](#_Toc235251712)

[3.11.9. SBB(51) - Binary Subtract 91](#_Toc235251713)

[3.11.10. SBBL(--) - Double Binary Subtract 91](#_Toc235251714)

[3.12. Special Maths Instructions 92](#_Toc235251715)

[3.12.1. APR(--) - Arithmetic Process 92](#_Toc235251716)

[3.12.2. AVG(--) - Average Value 92](#_Toc235251717)

[3.12.3. MAX(--) - Find Maximum 93](#_Toc235251718)

[3.12.4. MIN(--) - Find Minimum 94](#_Toc235251719)

[3.12.5. ROOT(72) - Square Root 95](#_Toc235251720)

[3.12.6. SUM(--) - Sum 96](#_Toc235251721)

[3.12.7. VCAL(69) - Value Calculate 96](#_Toc235251722)

[3.13. Input Comparison Instructions 97](#_Toc235251723)

[3.13.1. Input Comparison Instructions 97](#_Toc235251724)

[Bảng 3.5. Biểu diễn chức năng code, thuật nhớ (Mnemonic), tên, và chức năng của ngõ vào so sánh. 97](#_Toc235251725)

[3.14. Time Instructions 98](#_Toc235251726)

[3.14.1. HMS(--) - Seconds-To-Hours 98](#_Toc235251727)

[3.14.2. SEC(--) - Hours-To-Seconds 99](#_Toc235251728)

[3.14.3. HTS(65) - Hours-To-Seconds 99](#_Toc235251729)

[3.14.4. STH(66) - Seconds-To-Hours 100](#_Toc235251730)

[3.15. Logic Instructions 100](#_Toc235251731)

[3.15.1. ANDW(34) - Logical AND 100](#_Toc235251732)

[3.15.2. COM(29) - Complement 101](#_Toc235251733)

[3.15.3. ORW(35) - Logical OR 101](#_Toc235251734)

[3.15.4. XNRW(37) - Exclusive NOR 102](#_Toc235251735)

[3.15.5. XORW(36) - Exclusive OR 102](#_Toc235251736)

[3.16. Flag/Register Instructions 103](#_Toc235251737)

[3.16.1. CLC(41) - Clear Carry 103](#_Toc235251738)

[CHƯƠNG 4. CÁC CHƯƠNG TRÌNH ỨNG DỤNG CỦA PLC OMROM 104](#_Toc235251739)

[4.1. Chương Trình Điều Khiển Trò Chơi Dạng “ ĐƯỜNG LÊN ĐỈNH OLYMPIA” 104](#_Toc235251740)

[Bảng 4.1. Các ngõ vào ra. 104](#_Toc235251741)

[4.2. Mạch Tự Động Điều Khiển Của Kho 105](#_Toc235251742)

[Bảng 4.2. Phân bố các ngõ vào ra. 106](#_Toc235251743)

[4.3. Mạch Điều Khiển Bãi Đậu Xe 107](#_Toc235251744)

[4.4. Hoạt Động Điều Dẩn Nước Ra Hay Đổ Nước Vào 109](#_Toc235251745)

[Bảng 4.4. Phân bố thiết bị vào ra. 110](#_Toc235251746)

[4.5. Hệ Thống Tự Động Bôi Trơn Dầu Cho Bánh Xe 111](#_Toc235251747)

[Bảng 4.5.Các thiết bị vào ra. 112](#_Toc235251748)

[4.6. Mạch Điều Khiển Động Cơ Băng Tải 113](#_Toc235251749)

[Bảng 4.6. Các thiết bị vào ra. 113](#_Toc235251750)

[4.7. Điều Khiển Cơ Cấu Máy Lựa Sản Phẩm 114](#_Toc235251751)

[Bảng 4.7. Phân bố thiết bị vào ra. 115](#_Toc235251752)

[KẾT LUẬN 117](#_Toc235251753)

[Tài Liệu Tham Khảo 118](#_Toc235251754)

[[1] Phần mềm CX-Programmer. 118](#_Toc235251755)

[[2] Giáo trình PLC. 118](#_Toc235251756)

[[3] Website: 118](#_Toc235251757)

# Danh sách các hình

[Hình 1.1. PLC CP1L. 2](#_Toc235251758)

[Hình 1.2. Kết nối với các thiết bị khác. 3](#_Toc235251759)

[Hình 1.3. Chuyển đổi để thích ứng với các loại PLC khác. 4](#_Toc235251760)

[Hình 1.4. Khối hàm, biểu đồ hình thang, cấu trúc chữ. 5](#_Toc235251761)

[Hình 1.5. Cổng kết nối USB 5](#_Toc235251762)

[Hình 1.6. Kết nối với LCD và CPU. 5](#_Toc235251763)

[Hình 1.7. Đầu ra xung. 6](#_Toc235251764)

[Hình 1.8. Modbus RTU. 6](#_Toc235251765)

[Hình 1.9. CP1L-L. CP1L-M :      + 30,40,60 đầu I/O. + dung lượng 10 k. + Tốc độ thực hiện 0,55 µs. + số modul mở rộng là 3. 7](#_Toc235251766)

[Hình 1.10. CP1L-M. 7](#_Toc235251767)

[Hình 1.11. CP1L- L. 8](#_Toc235251768)

[Hình 1.12. Giao tiếp cùng các biến tần, đồng hồ hiển thị, màn hình cảm ứng. 8](#_Toc235251769)

[Hình 1.13. Cổng truyền thông. 9](#_Toc235251770)

[Hình 2.1. Chọn kênh truyền tin. 11](#_Toc235251771)

[Hình 2.2. Lựa chọn các thông số cho kênh truyền tin. 11](#_Toc235251772)

[Hình 2.3. Các thành phần trong cửa sổ project. 12](#_Toc235251773)

[Hình 2.4. Các cửa sổ phụ trên màn hình giao diện của CX-Programmer. 12](#_Toc235251774)

[Hình 2.5. Cách thêm tiếp điểm. 13](#_Toc235251775)

[Hình 2.6. Cách thêm cuộn dây. 13](#_Toc235251776)

[Hình 2.7. Cách thêm funtion. 14](#_Toc235251777)

[Hình 2.8. Cách thêm hàng vào rung. 14](#_Toc235251778)

[Hình 2.9. Cách thêm cột vào rung. 15](#_Toc235251779)

[Hình 2.10. Cách chèn thêm một rung. 15](#_Toc235251780)

[Hình 2.12. Các thao tác copy và past. 16](#_Toc235251781)

[Hình 2.13. Thao tác xoá xung. 16](#_Toc235251782)

[Hình 2.14. Cách thêm các tên cục bộ vào trong danh sach. 17](#_Toc235251783)

[Hình 2.15. Chọn loại CPU trong CX-Simulator. 17](#_Toc235251784)

[Hình 2.16. Chọn loại CPU trong CX-Simulator. 18](#_Toc235251785)

[Hình 2.17. Cách chạy mô phỏng. 18](#_Toc235251786)

[Hình 2.18. Biên dịch chương trình. 19](#_Toc235251787)

[Hình 2.19. Kết quả biên dịch. 19](#_Toc235251788)

[Hình 2.20. Kiểm tra lỗi. 20](#_Toc235251789)

[Hình 2.21. Nạp chương trình. 20](#_Toc235251790)

[Hình 2.22. Chế độ Monitor. 21](#_Toc235251791)

[Hình 2.23. Thay đổi chương trình trực tiếp online. 21](#_Toc235251792)

[Hình 2.24. Theo dõi sự thay đổi. 22](#_Toc235251793)

[Hinh 3.1. Điều kiện thực hiện ngõ vào và cờ hoàn thành. 37](#_Toc235251794)

[Hình 3.2. Dịch bit sang phải. 44](#_Toc235251795)

[Hình 3.3. Dịch bit sang phải. 44](#_Toc235251796)

[Hình 3.4. Rotate left. 45](#_Toc235251797)

[Hình 3.6. Định vị word điều khiển. 48](#_Toc235251798)

[Hình 3.7. Dịch dữ liệu giữa St và E sang trái. 50](#_Toc235251799)

[Hình 3.8. Dịch dữ liệu giaữ St và E một số sang phải. 51](#_Toc235251800)

[Hình 3.9**.** Dịch dữ liệu trong từng word giữa ST và trong khối word. 51](#_Toc235251801)

[Hình 3.10. Chép nội dung của S cho tất cả các word từ St và E. 52](#_Toc235251802)

[Hình 3.11. Chép nội dung của S sang D. 55](#_Toc235251803)

[Hình 3.12. Chép nội dung đã định của bit trong S cho bit đã định trong D. 57](#_Toc235251804)

[Hình 3.13. Sao chép nội dung của kí số xác định trong S đến kí số xác định trong D 58](#_Toc235251805)

[Hình 3.14. Truyền nội dung đảo của S cho D. 58](#_Toc235251806)

[Hình 3.15. Trao đổi nội dung của E1 và E2. 59](#_Toc235251807)

[Hình 3.16. Sao chép của S, S+1, …, S+N-1 đến D, D+1 ,…, D+N-1. 60](#_Toc235251808)

[Hình 3.17. Chép những bit nguồn đã định của bit S thành những bit đích đã định trong D. 61](#_Toc235251809)

[Hình 3.18. **B**iến đổi nội dung nhị phân 32 bit của S và S+1 thành 8 kí số của BCD. 70](#_Toc235251810)

[Hình 3.19. Biến đổi 8 kí số trong S và S+1 thành dữ liệu nhị phân 32 bit. 72](#_Toc235251811)

[Hình 3.20. Sao chép cột bit C từ bộ 16 word (S tới S+15) đến 16 bit của word D( 00 tới 15). 73](#_Toc235251812)

[Hình 3.21. Xác định giá trị hexa. 74](#_Toc235251813)

[Hình 3.22. Sao chép bit từ word tới cột. 76](#_Toc235251814)

[Hình 3.23. Biến đổi số hexa sang giá trị Decimal. 77](#_Toc235251815)

[Hình 3.24. Cộng nội dung của CY với giá trị trong Au và Au+1. 79](#_Toc235251816)

[Hình 3.25. Thực hiện Dd chia cho Dr. 80](#_Toc235251817)

[Hình 3.26. Dd và Dd+1 được chia bởi nội dung của Dr và Dr+1. 81](#_Toc235251818)

[Hình 3.27. Nhân Md với nội dung của Mr. 82](#_Toc235251819)

[Hình 3.28. Nhân nội dung của Md và Md+1với nội dung của Mr và Mr+1. 83](#_Toc235251820)

[Hình 3.29. Trừ CY và nội dung của Su và Su+1 và giá trị trong Mi và Mi+1. 84](#_Toc235251821)

[Hình 3.30. Cộng nội dung của Au+1 và Au, nội dung của Ad+1 và Ad, CY. 86](#_Toc235251822)

[Hình 3.31. Chia nội dung số nhị phân của Dd với nội dung của số nhị phân của Dr. 87](#_Toc235251823)

[Hình 3.32. Chia nội dung dữ liệu nhị phân 32 bit trong Dd+1và Dd với nội dung của dữ liệu nhị phân 32 bit trong Dr+1và Dr. 88](#_Toc235251824)

[Hình 3.33. Chia nội dung của Dd bởi nội dung của Dr. 88](#_Toc235251825)

[Hình 3.34. Nhân nội dung số nhị phân của hai word. 89](#_Toc235251826)

[Hình 3.35. Nhân nội dung của dữ liệu nhị phân 32 bit trong Md+1 và Md cùng với dữ liệu nhị phân 32 bit trong Mr+1 và Mr. 90](#_Toc235251827)

[Hình 3.36. Nhân nội dung của Md với nôi dung của Mr. 90](#_Toc235251828)

[Hình 3.37. Trừ CY và giá trị trong Su tới Su+1 và giá trị trong Mi tới Mi+1. 92](#_Toc235251829)

[Hình 3.38. Khai căn bậc hai của Sq và Sq+1. 95](#_Toc235251830)

[Hình 4.1. Chương trình điều khiển trò chơi dạng “ ĐƯỜNG LÊN ĐỈNH OLYMPIA”. 104](#_Toc235251831)

[Hình 4.2. Mạch tự động điều khiển của kho. 106](#_Toc235251832)

[Bảng 4.3. Phân bố thiết bị vào ra. 108](#_Toc235251833)

[Hình 4.4. Hoạt động điều dẩn nước ra hay đổ nước vào. 110](#_Toc235251834)

[Hình 4.5. Hệ thống tự động bôi trơn dầu cho bánh xe. 112](#_Toc235251835)

[Hình 4.6. Hệ thống điều khiển động cơ băng tải. 113](#_Toc235251836)

[Hình 4.7. Hệ thống điều khiển cơ cấu máy lựa sản phẩm. 115](#_Toc235251837)

# Danh sách các bảng

[Bảng 1.1. Các môđun CPU. 3](#_Toc235251838)

[Bảng 3.1. I/O words khác nhau cho mỗi PLC. 40](#_Toc235251839)

[Bảng 3.2. Mô tả cách thực hiện lệnh so sánh khối. 63](#_Toc235251840)

[Bảng 3.3. Cờ kết quả 68](#_Toc235251841)

[Bảng 3.4. Cờ kết quả. 69](#_Toc235251842)

[Bảng 3.5. Biểu diễn chức năng code, thuật nhớ (Mnemonic), tên, và chức năng của ngõ vào so sánh. 97](#_Toc235251843)

[Bảng 4.1. Các ngõ vào ra. 104](#_Toc235251844)

[Bảng 4.2. Phân bố các ngõ vào ra. 106](#_Toc235251845)

[Bảng 4.4. Phân bố thiết bị vào ra. 110](#_Toc235251846)

[Bảng 4.5.Các thiết bị vào ra. 112](#_Toc235251847)

[Bảng 4.6. Các thiết bị vào ra. 113](#_Toc235251848)

[Bảng 4.7. Phân bố thiết bị vào ra. 115](#_Toc235251849)

# Lời mở đầu

Đề tài này sẽ khái quát rõ nhất về các tập lệnh trong OMRON và ứng dụng các tập lệnh này để lập trình cho PLC.Mà cụ thể, chúng ta con có thể sử dụng tập lệnh này (thông qua phần mềm CX-Programmer) tạo và quản lý các dự án tự động hóa, đặt thông số hoạt động cho PLC v.v… Trong đề tài này đã tổng kết được 16 nhóm lệnh với những ý nghĩa và ứng dụng khác nhau và cụ thể được trình bày trong nội dung của bài.

Nội dung của seminar được trình bày như sau:

***Phần mở đầu***

***Chương 1: Giới Thiệu PLC CP1L***

***Chương 2: Giới Thiệu Phần Mềm Cx- programmer***

***Chương 3: Bộ lệnh của hãng OMRON***

***Chương 4: Các chương trình ứng dụng của PLC OMRON***

***Kết Luận***

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| CHUƠNG 1. GIỚI THIỆU PLC CP1L1.1. Giới Thiệu  Năm 2005, Omron đã đưa ra thị trường dòng Micro PLC với tính năng đột phá CP1H. Tháng 7 năm nay, Omron lại tiếp tục cho ra dòng Micro PLC CP1L đa năng với giá thành thấp hơn CP1H.  omron080907  Hình 1.1. PLC CP1L.  Về hình dáng bên ngoài, CP1L với màu đen khỏe khoắn khá giống CP1H do đều cùng được thiết kế trên nền tảng của dòng CJ1 tiên tiến. Về lâu dài, CP1L sẽ thay thế cho dòng CPM1/2A đã có mặt trên thị trường hơn 15 năm nay. 1.2. Các Tính Năng Của CP1L   Các tính năng chính của CP1L bao gồm: • CPU có sẵn 14, 20, 30, hoặc 40 I/O, mở rộng tối đa 160 I/O, có khả năng truyền thông RS-232 / 485 / 422.  • Kết nối với môđun mở rộng CPM1 hiện có (môđun I/O, đầu vào nhiệt độ, vào / ra tương tự, mạng CompoBus / S, DeviceNet).  • Đầu vào analog 0-10V (độ phân giải 256 bit).  • Đầu vào / ra xung 100kHz, cao hơn nhiều so với tốc độ tối đa 20kHz của CPM2A.  • Bộ nhớ 5/10Kstep (CPM1A có bộ nhớ 2 Kstep, còn CPM2A là 4 Kstep), có bộ nhớ ngoài.  • Lập trình thuận tiện thông qua cổng USB bằng CX-P V7.1, hỗ trợ lập trình bằng FB (Function Block).  • Hỗ trợ chạy mô phỏng bằng CX-Simulator (CPM1/2A không cho phép mô phỏng).   1.3. Cách Kết Nối Vào Ra Cũng như CP1H, dòng CP1L hỗ trợ khả năng kết nối dễ dàng trực tiếp tới các thiết bị khác như biến tần (qua Modbus), điều khiển nhiệt độ (qua định dạng CompoWay / F) như hình minh họa dưới đây.  a783b0a0116ed34b4361dfb48986f7da  Hình 1.2. Kết nối với các thiết bị khác.  Bảng 1.1. Các môđun CPU.   |  |  |  |  |  | | --- | --- | --- | --- | --- | | Môđun CPU | 14 đầu  (8 vào, 6 ra), | 20 đầu  (12 vào, 8 ra) | 30 đầu  (18 vào, 12 ra) | 40 đầu  (24 vào, 16 ra) | | Nối được 1 môđun mở rộng | | Nối được 3 môđun mở rộng | | | Nguồn AC, vào DC, ra relay | CP1L-L14DR-A | CP1L-L20DR-A | CP1L-M30DR-A | CP1L-M40DR-A | | Nguồn DC, vào DC, ra relay | CP1L-L14DR-D | CP1L-L20DR-D | CP1L-M30DR-D | CP1L-M40DR-D | | Nguồn DC, vào DC, ra transistor (NPN) | CP1L-L14DT-D | CP1L-L20DT-D | CP1L-M30DT-D | CP1L-M40DT-D | | Nguồn DC, vào DC, ra transistor (PNP) | CP1L-L14DT1-D | CP1L-L20DT1-D | CP1L-M30DT1-D | CP1L-M40DT1-D |  1.4. Các Ứng Dụng Của PLC CP1L  1.4.1. PLC OMRON tiêu chuẩn cùng với các tính năng thông minh CP1L cung cấp các hàm toán học, hàm PID có thể sử dụng cho những ứng dụng từ đơn giản đến phức tạp  Các chương trình có thể chuyển đổi để thích ứng với các loại PLC khác của Omron mà không cần chỉnh sửa.  cp1l_01 Hình 1.3. Chuyển đổi để thích ứng với các loại PLC khác. 1.4.2. Thuận lợi với nhiều ngôn ngữ lập trình CP1L cung cấp 3 sự lựa trọn về ngôn ngữ lập trình: Khối hàm, biểu đồ hình thang, cấu trúc chữ.  cp1l_02 Hình 1.4. Khối hàm, biểu đồ hình thang, cấu trúc chữ. 1.4.3. Dễ dàng kết nối với USB Đặc biệt thích hợp cho người sử dụng không có cổng giao tiếp nối tiếp trên máy tính xách tay.  cp1l_03 Hình 1.5. Cổng kết nối USB 1.4.4. Dễ dàng cho việc bảo dưỡng và điều chỉnh ban đầu nhờ vào màn hình LCD cho phép hiển thị và cài đặt Có thể gắn thêm 1 màn hình LCD vào CPU để hiển thị, thay đổi dữ liệu, và kiểm tra các trạng thái lỗi một cách trực quan  cp1l_10 Hình 1.6. Kết nối với LCD và CPU. 1.4.5. Điều khiển chuyển động với độ chính xác cao **Đầu ra xung:**CP1L cung cấp 2 đầu ra xung có tần số lớn nhất là 100 KHz. Nó cũng cung cấp tất cả các hàm cho điều khiển chuyển động như: Tăng hình thang, giảm hình thang…  cp1l_04 Hình 1.7. Đầu ra xung.  **Bộ đếm tốc độ cao:**CP1L cung cấp nhiều nhất là 4 đầu vào đếm xung 1 pha, 2 đầu vào nhiều pha  khác, và các đầu vào ngắt.  **Modbus RTU:**CP1L có nhiều lựa chọn cho giao tiếp nối tiếp như giao tiếp Modbus-RTU không cần phải lập trình.  cp1l_05 Hình 1.8. Modbus RTU.  **Tăng số đầu vào ra và dung lượng bộ nhớ:**Cùng với sự lựa chọn CP1L với nhiều dải rộng. Người dùng còn có thể lựa chọn PLC phù hợp với nhiều ứng dụng lớn hơn bằng cách thêm các modul mở rộng.   |  |  | | --- | --- | | CP1L-L :  + 10 đầu I/O.  + dung lượng 5 k. + Tốc độ thực hiện 0,55 µs.  CP1L-L (I/O Size: 10) |  |  |  |  | | --- | --- | | Hình 1.9. CP1L-L. CP1L-M :      + 30,40,60 đầu I/O. + dung lượng 10 k. + Tốc độ thực hiện 0,55 µs. + số modul mở rộng là 3.  cp1l_07  Hình 1.10. CP1L-M.  CP1L- L:  + 14,20 đầu I/O. - dung lượng 5 k. - Tốc độ thực hiện 0,55 µs. - modul mở rộng là 1.  cp1l_06  Hình 1.11. CP1L- L. |  |  1.4.6. Khả năng giao tiếp cùng với truyền thông nối tiếp Nối tiếp RS 485/RS 422. CP1L có khả năng giao tiếp cùng các biến tần, đồng hồ hiển thị, màn hình cảm ứng..  cp1l_08 Hình 1.12. Giao tiếp cùng các biến tần, đồng hồ hiển thị, màn hình cảm ứng. 1.4.7. Có khả năng lựa chọn cổng giao tiếp nối tiếp Giao tiếp 1:1 hoặc 1:N có thể lựa chọn cho cổng truyền thông.  cp1l_09 Hình 1.13. Cổng truyền thông.   |  |  | | --- | --- | | CHƯƠNG 2. GIỚI THIỆU PHẦN MỀM CX- PROGRAMMER 2.1. Giới Thiệu CX-Programmer là phần mềm trung tâm. Không chỉ dùng để lập trình cho PLC, CX-Programmer còn là công cụ để các kỹ sư quản lý 1dự án tự động hóa với PLC làm bộ não hệ thống. Các chức năng chính của CX-Programmer bao gồm:   * Tạo và quản lý các dự án (project) tự động hóa. * Kết nối với PLC qua nhiều đường giao tiếp. * Cho phép thực hiện các thao tác chỉnh sửa & theo dõi khi đang online (như force set/reset, online edit, monitoring,..). * Đặt thông số hoạt động cho PLC. * Cấu hình đường truyền mạng. * Hỗ trợ nhiều chương trình, nhiều PLC trong 1 cùng project và nhiều section trong 1 chương trình.  2.2. Cách Sử Dụng Phần Mềm2.2.1.Tạo một project mới **⬥** Chọn kênh truyền tin    Hình 2.1. Chọn kênh truyền tin.  ⬥ Lựa chọn các thông số cho kênh truyền tin    Hình 2.2. Lựa chọn các thông số cho kênh truyền tin. |  |  2.2.2. Các thành phần trong cửa sổ project   Hình 2.3. Các thành phần trong cửa sổ project. 2.2.3. Các thao tac trên cửa sổ project ⬥ Các cửa sổ phụ trên màn hình giao diện của CX-Programmer.  Trong quá trình làm việcvới CX-Programmer, người sử dụng có thể bật hoặc tắt các cửa sổ phụ.Các cửa sổ này hiển thị các thông tin có liên quan đến các đối tượng và công việc đang được thực thi.    Hình 2.4. Các cửa sổ phụ trên màn hình giao diện của CX-Programmer.  ⬥ Thêm tiếp điểm    Hình 2.5. Cách thêm tiếp điểm.  ⬥ Thêm cuộn dây  Hình 2.6. Cách thêm cuộn dây.  ⬥ Thêm function  Mỗi chương trình đều cầncó ít nhất1 lệnh End để đánh dấu điểm kết thúc của chương trình. Lệnh End và nhiều khối chức năng khác (function) có thể nhập vào dùng công cụ Instruction.    Hình 2.7. Cách thêm funtion.  ⬥ Thêm hàng vào Rung    Hình 2.8. Cách thêm hàng vào rung.  ⬥ Thêm cột vào Rung    Hình 2.9. Cách thêm cột vào rung.  ⬥ Chèn thêm một Rung    Hình 2.10. Cách chèn thêm một rung.  ⬥ Các thao tác copy và past  Ta có thể áp dụng thao tác Copy và Past như với 1chương trình Windows thông thường khác. Đồng thời có thể áp dụngUndo và Redo với các thao tác vừa làm.    Hình 2.12. Các thao tác copy và past.  ⬥ Xóa Rung    Hình 2.13. Thao tác xoá xung.  ⬥ Thêm các tên cục bộ vào trong danh sach    Hình 2.14. Cách thêm các tên cục bộ vào trong danh sach. 2.2.4. Mô phỏng với Cx- Simulator ⬥ Cấu hình CX-Simulator  Chọn loại CPU trong CX-Simulator phải trùng với loại CPU trong CX-Programmer.    Hình 2.15. Chọn loại CPU trong CX-Simulator.    Hình 2.16. Chọn loại CPU trong CX-Simulator.  ⬥ Chạy mô phỏng  Việc chạy mô phỏng với CX-Simulator đển hằm kiểm tra phát hiện các lỗi do sai cú pháp, thiếu/thừa các phần tử, … trong chương trình và tiến hành xử lý khắc phục các lỗi này trước khi tiến hành nạp vào PLC.    Hình 2.17. Cách chạy mô phỏng.  ⬥ Biên dịch chương trình  Bấm vào nútWork Online để kết nối với PLC sau khi đãn ối cáp giữa máy tính với PLC. Sau khi kết nối được thiết lập, CX-Programmer sẽ ở chế độ làm việc Online. Bấm lại vào nút Work Online sẽ chuyển sang chế độ Offline để có thể sửa chương trình.    Hình 2.18. Biên dịch chương trình.  Kết quả biên dịch được hiển thị trong tabcompile của cửa sổ Ouput    Hình 2.19. Kết quả biên dịch.  ⬥ Kiểm tra lỗi  Khi đang online có thể kiểm tra và xóa các lỗi tron PLC bằng cách nhấn đúp vào Error Log.    Hình 2.20. Kiểm tra lỗi.  ⬥ Nạp chương trình    Hình 2.21. Nạp chương trình.  Để chạy chương trình vừa nạp vào PLC, cần chuyển sang chế độ Monitor hoặc Run Mode.    Hình 2.22. Chế độ Monitor.  ⬥ Thay đổi chương trình trực tiếp online  CX-Programmer cho phép sửa chương trình ngay cả khi PLC đang ở chế độ chạy bằng cách dùng tính năng Online Edit.Sau khi thực hiện xong các thay đổi nhớ lưu vào bộ nhớ PLC.    Hình 2.23. Thay đổi chương trình trực tiếp online.  ⬥ Theo dõi sự thay đổi  Với các bit thay đổi nhanh, ta có thể sử dụng chức năng này để phát hiện sự thay đổi một cách trực quan.    Hình 2.24. Theo dõi sự thay đổi. CHƯƠNG 3. BỘ LỆNH CỦA HÃNG OMRON |

## 3.1. Ladder Instructions

### 3.1.1. AND - And



B :bit

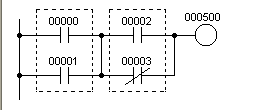
**Mục đích:**

Lệnh And dùng để nối công tắc thường hở với một công tắc đứng trước nó.

**Vùng dữ liệu toán tử:**

B: IO, HR, AR, LR, TC.

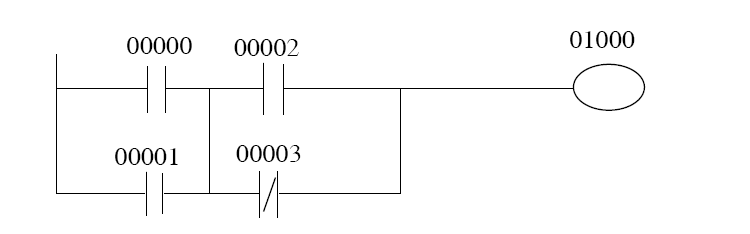
### 3.1.2. AND LD - And Load



**Mục đích:**

Lệnh ANDLD:là lệnh dùng để liên kết hai khối công tắc liên tiếp với nhau.

Ví dụ:



Viết chương trình cho sơ đồ hình thang trên:

Địa chỉ lệnh dữ liệu

00000 LD 00000

00001 OR 00001

00002 LD 00002

00003 ORNOT 00003

00004 ANDLD -------

00005 OUT 01000

**Vùng dữ liệu toán tử:** N/A.

### 3.1.3. AND NOT - And Not



B:bit

**Mục đích:**

Lệnh AND NOT :dùng để nối tiếp một công tắc thường đóng với một công tắc đứng trước nó.

**Vùng toán tử dữ liệu:**

B: IO, HR, AR, LR, TC.

### 3.1.4. LD - Load



B:bit

**Mục đích:**

Lệnh này nạp một công tắc thường hở nối với phía bên trái đường dây điện.

**Vùng dữ liệu toán tử:**

B: IO, HR, AR, LR, TC, TR

### 3.1.5. LD NOT - Load Not



B:bit

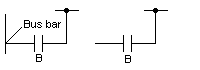
**Mục đích:**

Lệnh này nạp một công tắc thường đóng nối với phía bên trái đường dây điện.

**Vùng dữ liệu toán tử:**

B: IO, HR, AR, LR, TC

### 3.1.6. OR - Or



B:bit

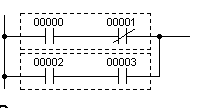
**Mục đích:**

Lệnh OR dùng để nối một công tắc thường hở với một đường dây điện bên trái song songvới một hoặc nhiều công tắc đứng trong cùng một nhánh.

**Vùng dữ liệu toán tử:**

B: IO, HR, AR, LR, TC.

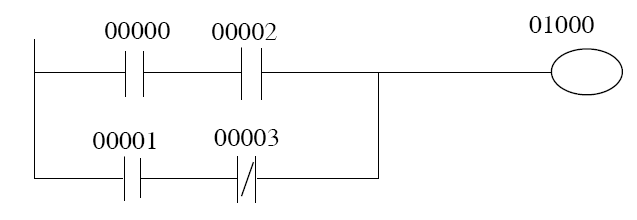
### 3.1.7. OR LD - OR Load



**Mục đích:**

Lệnh ORLD dùng để liên kết hai khối song song nhau.

Ví dụ:



Viết chương trình cho sơ đồ hình thang trên:

Địa chỉ lệnh dữ liệu

00000 LD 00000

00001 AND 00002

00002 LD 00001

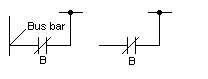
00003 ANDNOT 00003

00004 ORLD -------

00005 OUT 01000

**Vùng dữ liệu toán tử:** N/A

### 3.1.8. OR NOT - Or Not



B: bit

**Mục đích:**

Lệnh OR NOT dùng để nối một công tắc thường đóng với một đường dây điện bên trái song song với một hoặc nhiều công tắc đứng trong cùng một nhánh.

**Vùng dữ liệu toán tử:**

B: IO, HR, AR, LR, TC

### 3.1.9. OUT – Output



**OUT NOT - Output NOT**



B;bit.

**Mục đích:**

Lệnh OUT và OUT NOT dùng để điều khiển trạng thái của bit đã chỉ định theo điều kiện thực hiện ngõ vào là ON thì bit OUT sẽ ON, còn bit OUT NOT sẽ OFF và ngược lại.

**Vùng dữ liệu toán tử:**

B: IO, AR, HR, TC, LR, TR.

## 3.2. Program Control Instructions

### 3.2.1. END(01) - End



**Mục đích:**

Là lệnh đặt ở cuối chương trình, nếu không có lệnh này trong chương trình thì toàn bộ chương trình sẽ không thực hiện.

**Vùng dữ liệu toán tử:** N/A.

### 3.2.2. IL(02) - Interlock

### ILC(03) - Interlock Clear

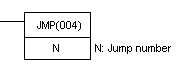


**Mục đích:**

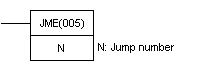
Lệnh IL luôn được dùng để kết hợp với ILC để tạo thành một khối khóa trong. Nếu điều kiện thực hiện ngõ vào cho IL là OFF thì tất cả các ngõ ra và tất cả các giá trị hiện tại của TIMER ở trong IL và ILC được OFF hay reset. Một số lệnh khác thì không hoạt động. Giá trị hiện tại của COUNTER được duy trì. Nếu điều kiện thực hiện ngõ vào là ON thì sự thực hiện của lệnh IL và ILC không ảnh hưởng, chương trình tiếp tục bình thường.

Vùng dữ liệu toán tử: N/A.

### 3.2.3. JMP(04) - Jump



### JME(05) - Jump End



**Mục đích:**

N: là số nhảy.

Số nhảy N trong lệnh là từ 00 đến 99.

JMP luôn luôn được dùng để kết hợp với JME để tạo thành lệnh nhảy để nhảy từ một điểm trong sơ đồ hình thang đến một điểm khác. JMP được định nghĩa là điểm mà tại đó lệnh nhảy được tạo. JME được định nghĩa là điểm đích của lệnh nhảy. Khi điều kiện thực hiên ngõ vào JMP là ON thì bước nhảy không được tạo và chương trình được thực hiện liên tục như đã lập trình. Khi điều kiện thực hiện ngõ vào cho JMP là OFF thì một bước nhảy được thi hành, khi đó chương trình tiếp tục thực hiện tại JME.

Khi số nhảy N của JMP từ 01 đến 99 thì con trỏ lập tức chuyển đến JME với cùng số nhảy N tương ứng, tất cả các lệnh của JMP và JME không thực hiện, trạng thái của những Timer, Counter, Bit Out, Out Not và tất cả trạng thái của bít điếu khiển khác sẽ không thay đổi. Số nhảy này chỉ định nghĩa cho một lần nhảy.

Khi số nhảy N cho JMP là ON, CPU sẽ tìm đến JME kế có số nhảy N=00. Để thực hiện nó kiểm tra toàn bộ chương trình tất cả những lệnh và bít điều khiển nằm ở giữa JMP 00 và JME 00 được giữ nguyên. Số nhảy 00 có thể được sử dụng nhiều lần đối với JMP mà chỉ cần một đích nhảy đến JME.

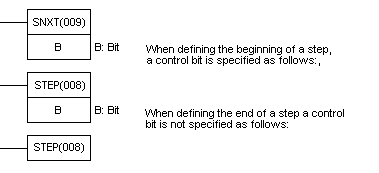
**Vùng dữ liệu toán tử:**

N 0 to 8 (CxxK).  
 0 to 49 (CPM1).  
 0 to 99 (all other PLCs) .

### 3.2.4. STEP(08) - Step Define

# 

### SNXT(09) - Step Start



**Mục đích:**

Lệnh bước Step và SNXT được dùng kết hợp với nhau để đặt điểm dừng giữa những phần trong một chương trình lớn vì vậy những phần có thể thực hiện như những khối và được đặt lại lúc hoàn thành. Step dùng một bít điều khiển trong vùng IR hay HR… để định nghĩa cho phần bắt đầu của đầu chương trình gọi là bước. Step không cần điều kiện thực hiện, tức là sự thực hiện của nó được điều khiển thông qua bit điều khiển B đặt trong SNXT. SNXT là lệnh bắt đầu bước cho STEP thực hiện. Nếu điều kiện thực hiện ngõ vào cùa SNXT có dùng bit điều khiển B trong Step la ON thì bước sẽ đươc thực hiện. Nếu điều kiễn thực hiện ngõ vào của SNXT là OFF thì bước định nghĩa sẽ không được thực hiện, lệnh SNXT phải được con trỏ đọc tới trước khi bắt đầu bước. Bất kì một bước nào trong chương trình mà không được bắt đầu với SNXT thì bước đó sẽ không thực hiện.

Một SNXT được dùng trong chương trình thì sẽ thực hiện bước tiếp diễn cho đến khi Step được thực hiện mà không có bit điều khiển Step. Step không có bít điều khiển được đứng trước bởi SNXT với một bit điều khiển giả, bit điều khiển giả có thể là một bit bất kì không sử dụng trong IR hay HR. Vì vậy nó không phải là bit điều khiển dùng cho Step.

Sự thực hiện của bước được hoàn thành khi có sự xuất hiện của SNXT kế hay bit điều khiển cho bước được reset. Khi bước được hoàn thành, tất cả những bit của IR và HR trong bước được bật trở về OFF và tất cả Timer trong bước được reset về giá trị đặt của nó. Các Counter, thanh ghi dịch những bít dùng KEEP giữ nguyên trạng thái.

**Vùng dữ liệu toán tử:**

B: IO, AR, HR, LR.

### 3.2.5. STOP(99) - Stop



**Mục đích:**

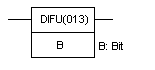
Lệnh này được dùng để dừng việc chạy của chương trình PLC khi ngõ vào relay được bật OFF. Khi relay được bật ON thì sự hoạt động được bắt đầu như trạng thái ban đầu.

Khi trạng thái thực hiện là ON, thì lệnh này sẽ được xử lý như NOP.

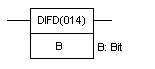
**Vùng dữ liệu toán tử:** N/A.

## 3.3. Bit Control Instructions

### 3.3.1. DIFU(13) - Differentiate Up



**DIFD(14) - Differentiate Down**



**Mục đích:**

Lệnh DIFU và DIFD được dùng để bật ON bit đã định trong một chu ki.

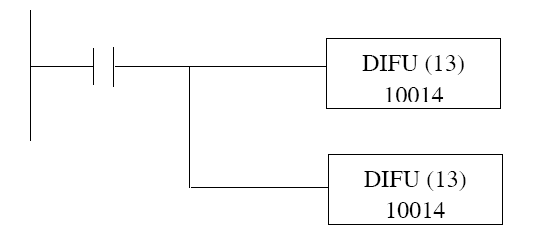
Mỗi khi thực hiện DIFD so sánh điều kiện thực hiện tại ngõ vào với điều kiện trước đó của nó. Nếu điều kiện thực hiện trước đó là OFF và hiện tại là ON, DIFU sẽ bật ON bit đã định. Nếu điều kiện thực hiện trước đó là ON và điều kiện thực hiện hiện tại là ON hay OFF lệnh DIFU sẽ OFF bit đã định.

Còn đối với lệnh DIFD khi thực hiện sẽ so sánh điều kiện thực diện ngõ vào hiện tại với điều kiện trước đó. Nếu điều kiện trước đó là ON và hiện tại là OFF thì lệnh DIFD sẽ bật On bit đã định. Nếu điều kiện thực hiện tại ngõ vào là ON bất chấp điều kiện trước đó là ON hay OFF, lệnh DIFD sẽ OFF bit đã định.

Hai lệnh này không ảnh hưởng đến cờ trạng thái

Ví dụ:

Sơ đồ hình thang:



Viết chương trình cho sơ đồ hình thang trên:

Địa chỉ lệnh dữ liệu

00000 LD 00000

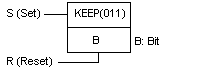
00001 DIFU(13) 10014

00002 DIFD(14) 10015

**Vùng dữ liệu toán tử:**

B: O, AR, HR, LR.

### 3.3.2. KEEP(11) - Keep



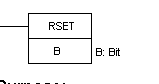
**Mục đích:**

Lệnh Keep dùng để duy trì trạng thái bit đã định theo hai điều kiện thực ngõ vào là S và R. S là ngõ vào Set, R là ngõ vào Reset. Lệnh KEEP hoạt động giống như một relay chốt mà được set bởi S và Reset bởi R.

**Vùng dữ liệu toán tử:**

B: IO, AR, HR, LR.

### 3.3.3-RSET - Reset



**Mục đích:**

Lệnh RESET dùng để OFF bit toán tử khi điều kiện thực hiện là ON và không ảnh hưởng trạng thái của toán tử khi điều kiện thực hiện là OFF.

**Vùng dữ liệu toán tử:**

B: IO, AR, HR, LR.

### 3.3.4. SET - Set



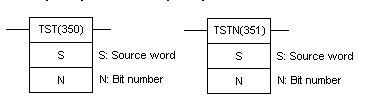
Mục đích:

Lệnh SET dùng để ON bit toán tử khi điều kiện thực hiện là ON và không ảnh hưởng trạng thái bit của toán tử khi điều kiện thực hiện ngõ vào là OFF.

**Vùng dữ liệu toán tử:**

B: IO, AR, HR, LR.

### 3.3.5. TST(350) & TSTN(351) - Bit Test



S: word nguồn.

N: số bit.

M**ục đích:**

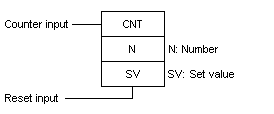
TST bật ON trạng thái thực hiện khi bit trong word là ON và bật OFF trạng thái thực hiện khi bit la OFF.  
 TSTN bật OFF trạng thái thực hiện khi bit trong word là ON và bật ON trạng thái thực hiện khi bit la OFF.

**Vùng dữ liệu toán tử:**

|  |
| --- |
| IO, AR, DM, HR, LR |
| IO, AR, DM, HR, LR, # |

## 3.4. Timer/Counter Instructions

### 3.4.1. CNT - Counter



**Mục đích:**

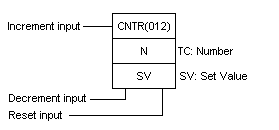
CNT dùng để đếm xuống từ giá trị đặt SV khi điều kiện thực hiện xung đếm chuyển từ trạng thái OFF sang ON, giá trị hiện tại (PV) sẽ được giảm xuống bằng một lần CNT thực hiện một xung đếm CV từ OFF sang ON. Nếu điều kiện xung đếm thay đổi hay chuyển từ trạng thái ON sang OFF thì giá trị PV của CNT không thay đổi, cờ hoàn thành cho một COUNTER được bật ON khi giá trị hiện tại PV bằng không và ở trạng thái ON cho đến khi counter được Reset.

Counter được Reset với ngõ vào Reset R khi R chuyển từ FF sang ON, và PV được reset về SV. Giá trị hiện tại PV của CNT sẽ không reset trong phần chương trình khác hay bởi sự ngắt nguồn.

**Vùng dữ liệu toán tử:**  N là chỉ số TC của CNT chạy từ 000 tới 511.

SV:giá trị đặt ở trong (word, BCD): IO, AR, DM, HR, #.

### 3.4.2. CNTR(12) - Reversible Counter



N: chỉ số TC.

SV: giá trị đặt ở trong.

**Mục đích:**

Là bộ đếm hai chiều. Nó được dùng để đếm giữa giá trị đặt SV và 0 theo sự chuyển đổi một trong hai điều kiện đó là ngõ vào tăng II và ngõ vào giảm DI.

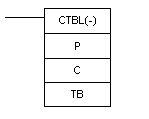
Giá trị hiện tại (PV) sẽ tăng nếu điều kiện đếm được đưa vào ngõ II và sẽ giảm nếu điều kiện đếm được đưa vào ngõ DI. Nếu điều kiện đếm được đưa vào cả hai ngõ II và DI thì giá trị hiện tại PV của CNTR sẽ giữ nguyên không thay đổi.

Khi giá trị hiện tại PV giảm tới 00 thì PV được đặt về giá trị đặt SV và cờ hoàn thành được bật ON cho đến khi giá trị PV giảm trở lại. Khi giá trị hiện tại PV tăng lên tới SV thì giá trị PV được đặt về 0 và cờ hoàn thành được bật sang ON cho đến khi giá tri PV tăng trở lại.

R là ngõ vào Reset của CNTR khi R chuyển từ OFF sang ON, giá trị PV được reset về 0.Giá trị PV sẽ không tăng không giảm khi R đang ON. Counter sẽ đếm trở lại khi R là OFF.

**Vùng toán tử dữ liệu:** SV: IO, AR, DM, HR, LR, #.

### 3.4.3. CTBL(--) - Register Comparison Table



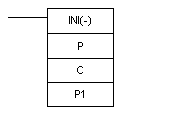
P: port specifier   
C: control data   
TB: 1st comparison table word  
**Mục đích:**

Lập bảng so sánh thanh ghi cho việc sử dụng counter PV ở tốc độ cao. Sự phụ thuộc vào giá trị của C, sự so sánh cùng với PV tốc độ cao có thể bắt đầu một cách liên tục hay nó có thể bắt đầu một cách riêng rẽ cùng với INI.

**Vùng dữ liệu toán tử:**

TB: IO, AR, DM, HR, LR.

### 3.4.4. INI(--) - Mode Control



P: port specifier

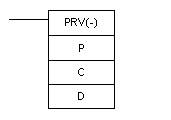
C: control data

P1: 1st PV word

Mục đích:

Lệnh này điều khiển hoạt động counter tốc độ cao và dừng xung ở ngõ ra.  
Port riêng (P) định rõ counter tốc độ cao hay xung ngõ ra mà nó sẽ điều khiển.  
Chức năng của INI được quyết định bởi nguồn dữ liệu điều khiển (C).  
**Vùng dữ liệu toán tử:**  
P1: IO, AR, DM, HR, LR.

### 3.4.5. PRV(--) - High-Speed Counter PV Read



**Mục đích:**

Khi điều kiện thực hiện ngõ vào là ON, lệnh PRV đọc dữ liệu đã định ở P và C rồi ghi nó vào D hay D+1.

Port riêng(P) xác định counter tốc độ cao hay xuất xung.

|  |  |
| --- | --- |
| P | Chức năng |
| 000 | Chỉ định counter tốc độ cao 0 hay xuất xung từ một bit |
| 001 | Chỉ định counter tốc độ cao 1 hay xuất xung từ port 1 |
| 002 | Chỉ định counter tốc độ cao 2 hay xuất xung từ port 2 |

C:dữ liệu điều khiển xác định loại dữ liệu xử lý truy xuất.

|  |  |  |
| --- | --- | --- |
| C | Dữ liệu | Word đích |
| 000 | Giá trị hiện tại PV của Counter tốc độ cao | D và D+1 |
| 001 | Trạng thái của counter tốc độ cao hay xuất xung | D |
| 002 | Dãy kết quả so sánh | D |

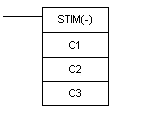
**Vùng dữ liệu toán tử:**

P: port riêng 000, 001, 002

C: dữ liệu điều khiển :000,001,002.

D: word đích đầu tiên: IO, AR, DM, HR, LR.

### 3.4.6. STIM(--) - Interval Timer



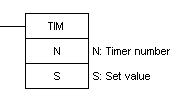
**Mục đích:**

STIM được dùng để điều khiển khoảng timer bằng việc thi hành bốn chức năng cơ bản: bắt đầu timer cho ngắt non-shot,bắt đầu timer cho ngắt scheduled, dừng timer và việc đọc PV của timer.

**Vùng dữ liệu toán tử:**

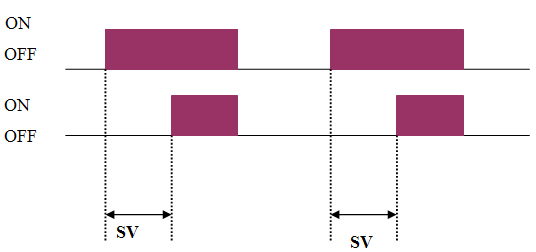
C1:control data #1 000 – 008,010 to 012.  
 C2:control data #2 IO, AR, DM, HR, TC, LR, #.  
 C3:control data #3 IO, AR, DM, HR, TC, LR, #.

### 3.4.7. TIM - Timer



**Mục đích:**N là số timer chạy từ 000 đến 511SV: là giá trị đặt cho timer được đặt từ 000.0 đến 999.9 với đơn vị là 0.1 giây.

Một timer được kích là điều kiện thưc hiện ngõ vào của nó được chuyển sang ON và nó được reset về giá trị đặt khi điều kiện thực hiện chuyển sang OFF. Nếu điều kiện cho timer duy trì trong một khoảng thời gian dài thì giá trị đặt của timer sẽ giảm về , cờ hoàn thành cho số TC dùng được bật ON và duy trì trạng thái cho đến khi timer được reset (đến khi điều kiện thực hiện ngõ vào chuyển sang OFF).

*Ví dụ:*

Điều kiện thực hiện ngõ vào

Cờ hoàn thành

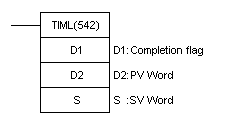
Hinh 3.1. Điều kiện thực hiện ngõ vào và cờ hoàn thành.

**Vùng dữ liệu toán tử:**

N: là số timer TC# 000-511.

SV: giá trị đặc (word, BCD) IO, AR, DM, HR, # sv.

### 3.4.8. TIML(--) - Long Timer



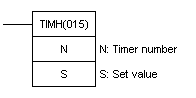
**Mục đích:**

TIML là sự giảm ON-delay timer mà thời gian trong vòng 1 giây hay 10 giây. Giá trị đặt timer có thể là 0-99990s. Long Timer được hoạt hóa khi trạng thái thực hiện của nó tiến đến ON và được reset khi trang thái hoạt động của nó tiến đến OFF.  
 Chú ý: TIML có thể không chính xác khi chu ki thời gian vượt quá 1s hay 10s.

**Vùng dữ liệu toán tử:**

|  |  |
| --- | --- |
| N: | Timer Timer Counter number #0 - #255 |
| SV: | Set Value IR, SR, AR, DM, HR, LR, # |
| C: | Control data 000 for 1 second timer, 001 for 10 second timer |

### 3.4.9. TIMH(15) - High-Speed Timer

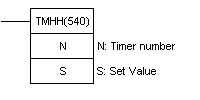


**Mục đích:**

Giá trị SV đặt trong TIMH chạy từ 00.00 đến 99.99 giây. Hoạt động của TIMH giống như TIM.

**Vùng dữ liệu toán tử:** N: chỉ số timer.SV: giá trị dặt(word,BCD) ở trong IO, AR, DM, HR, #.

### 3.4.10. TMHH(--) - Very High Speed Timer



**Mục đích:**

Là sự giảm ON-delay timer mà thời gian trong vòng 1s.Giá trị đặt timer có thể là 0-9.999s.

Long timer được hoạt hóa khi trạng thái thực hiện tiến đến ON và được reset khi trạng thái thực hiện tiến đến OFF.Timer có thể được bắt đầu lại bằng cách bật trạng thái hoạt động của nó lên OFF và ON một lần nữa.

**Vùng dử liệu toán tử:**

N: timer Timer Counter number #0 - #255.

S: set value IR, SR, AR, DM, HR, LR, #.

## 3.5. Subroutine Instructions

### 3.5.1. CMCR(--) - PCMCIA Card Macro

**Mục đích:** CMCR(--) - PCMCIA Card Macro.  
**Vùng dữ liệu toán tử:**   
 OPR1: control word IO, AR, DM, HR, TC, LR, #.  
 OPR2: Command word IO, AR, DM, HR, TC, LR, #.  
 OPR3:destination word IO, AR, DM, HR, TC, LR, #.

### 3.5.2. MCRO(99) - Macro



**Mục đích:**

Khi trạng thái thực hiện là OFF, MCRO không được thực hiện. Khi trạng thái thực hiện là ON, MCRO sao chép nội dung của I1-I1+3 tới IR 096-IR 099, sao chép nội dung của O1-O1+3 tới IR 196 –IR 199 và sau đó gọi và thực hiện chương trình con trong N. Khi chương trình con được hoàn thành, nội dung cùa IR 196 tới IR 199 được chuyển trở lại tới O1 –O1+3 trước khi MCRO được hoàn thành.

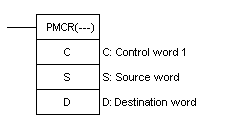
Bảng 3.1. I/O words khác nhau cho mỗi PLC.

|  |  |  |
| --- | --- | --- |
| **PLC** | **IN** | **OUT** |
| CPM1 | 232 | 236 |
| CQM1/IDSC | 96 | 196 |
| C200HS,E,G,X | 290 | 294 |

**Vùng dữ liệu toán tử:**

|  |  |  |
| --- | --- | --- |
| N: | Subroutine number | 0 to 49 (CPM1) 0 to 99 (C200HS) 0 to 255 (CQM1, IDSC and C200HX Family) |
| I1: | 1st input word | IO, AR, DM, HR, TC, LR |
| O1: | 1st output word | IO, AR, DM, HR, LR |

### 3.5.3. PMCR(--) - Protocol Macro



**Mục đích:**

PMCR gọi và thực hiện sự truyền đạt liên tiếp mà nó đã được đăng kí trong Communications Board đã đặt trong PC.

Việc gởi hoặc nhận cho sự truyền đạt liên tục đã đăng kí trong Communications Board phải được đọc hay viết dữ liệu word khi DM không được định rõ cho S và D. Sử dụng một hệ số khi nó không cần thiết để đặt một word cho word ngõ ra đầu tiên.

Khi sự truyền đạt thông tin liên tục không yêu cầu 1 word ngõ vào, địa chỉ word định rõ bất kì cách nào.

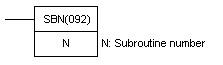
**Vùng dữ liệu toán tử:**

|  |  |  |
| --- | --- | --- |
| C: | Control word | IO, AR, DM, HR, TC, LR, # |
| S: | Source word | IO, AR, DM, HR, TC, LR,# |
| D: | Destination word | IO, AR, DM, HR, LR |

### 3.5.4. RET(93) - Return



### SBN(92) - Subroutine Define



**Mục đích:**

Lệnh SBN dùng để đánh dấu sự bắt đầu của chương trình con, lệnh RET dùng để đánh dấu sự kết thúc của chương trình con. Mỗi chương trình con được xác định bởi số N. Nó do người lập trình định nghĩa cho SBN và trùng với số N của lệnh tạo chương trình con SBS. Tất cả mọi chương trình con phải được lập ở cuối chương trình chính.

**Vùng dữ liệu toán tử:**

Đối với lệnh RET: N/A.

Đối với lệnh SBN:

|  |  |  |
| --- | --- | --- |
| N: | Subroutine number | 0 to 49 (CPM1) 0 to 99 (C100H, C200H, C200HS, C1000H, C2000, C2000H) 0 to 127 (CQM1 (CPU11/21)) 0 to 255 (CQM1 (CPU 4x and IDSC) |

### 3.5.5. SBS(91) - Subroutine Enter



**Mục đích:**

Một chương trình con có thể được gọi bằng cách đặt lệnh SBS trong chương trình chính tại nơi mà ta muốn gọi chương trình con. Số chương trình con N dùng trong SBS cho biết số chương trình con yêu cầu. Khi lệnh SBS thực hiện (tức là điều kiện thực hiện ngõ vào là ON ) những lệnh ở giữa SBN với cùng số chương trình con N và lệnh RET đầu tiên. Sau khi thực hiện xong chương trình con đã gọi con trỏ quay về lệnh tiếp theo.

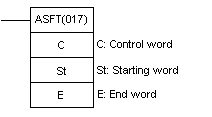
SBS có thể được gọi trong một chương trình con để dịch chuyển chương trình đã thực hiện từ một chương trình con này đến một chương trình con khác. Khi chương trình con thứ hai đã kết thúc chương trình thực hiện quay về thủ tục gốc sau đó hoàn thành chương trình con gốc rồi quay về chương trình chính. Lệnh tạo chương trình con có thể được tạo và gọi lên đến cấp 16 ( một chương trình con không thể gọi chính nó).

**Vùng dữ liệu toán tử:**

|  |  |  |
| --- | --- | --- |
| N: | Subroutine number | 0 to 49 (CPM1) 0 to 99 (C200H, C200HS, C1000H, C2000, C2000H) 0 to 127 (CQM1 (CPU11/21)) 0 to 255 (CQM1 (CPU 4x and IDSC)) |

## 3.6. Data Shifting Instructions

### 3. 6.1. ASFT(--) - Asynchronous Shift Register



**Mục đích:**

St và E phải cùng một vùng dữ liệu.

E phải lớn hay bằng St.

Khi điều kiện thực hiện ngõ vào là OFF lệnh ASFT không thực hiện và chương trình di chuyển đến lệnh kế đó. Khi điều kiện thực hiện ngõ vào là ON thì lệnh ASFT được dùng để tạo thanh ghi dịch word không đồng bộ hai chiều giữa St và E thanh ghi này chỉ dịch word khi word kế đó trong thanh ghi là zero. Nếu không có word nào trong thanh ghi chứa zero thì không có word nào được dịch. Cũng như một word chứa zero, khi nội dung của word dịch đến word kế thì nội dung của word nguồn được đặt về 0. Khi thanh ghi dịch, từng word zero trong thanh ghi được đánh dấu thay thế bằng một word kế.

Hướng dịch được xác định trong C. C cũng được dùng để reset thanh ghi, tất cả mọi word trong thanh ghi có thể reset được. Bằng cách xác định word mong muốn trong St và E.

Bít 00 tới 12 của C không dùng. Bít 13 là bit chỉ hướng dịch. Bít 13 ON là dịch xuống ( về hướng word có địa chỉ thấp ) và OFF là dịch lên ( về hướng word có địa chỉ cao ). Bít thứ 14 là cho phép dịch bít: ON cho phép thanh ghi dịch hoạt động theo bit 13 và OFF không cho phép thanh ghi dịch. Bít thứ 15 là bit reset ( set về 0). Thanh ghi sẽ được reset trong khoảng St tới E khi ASFT được thực hiện cùng với bit 15 ON. Bit 15 OFF thì hoạt động bình thường.

*Ví dụ:*  
Ví dụ mô tả lệnh ASFT được dùng để dịch những word trong thanh ghi có 11 word được tạo từ DM0100 và DM0110 với C= #6000= 040 0000 0000 0000 St (DM0100) và bit thứ 13 của C là ON.

Sơ đồ hình thang:



Chương trình cho sơ đồ hình thang:

Địa chỉ lệnh dữ liệu

00000 LD 00000

00001 ASFT (17) #6000

DM 0100

DM0110

**Vùng dữ liệu toán tử:**

|  |  |  |
| --- | --- | --- |
| C: | Control word | IO, AR, DM, HR, LR, # |
| St: | Starting word | IO, AR, DM, HR, LR |
| E: | End word | IO, AR, DM, HR, LR |

### 3.6.2. ASL(25) - Arithmetic Shift Left



**Mục đích:**

Khi điều kiện thực hiện ngõ vào là OFF ASL không thực hiện. Khi điều kiện thực hiện ngõ vào là ON, ASL dịch số không vào cho bit 15 của Wd và từng bit một trong Wd được dịch sang phải.



Hình 3.2. Dịch bit sang phải.

**Vùng dữ liệu toán tử:**

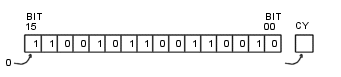
|  |  |  |
| --- | --- | --- |
| Wd: | Shift word | IO, AR, DM, HR, LR |

### 3.6.3. ASR(26) - Arithmetic Shift Right



**Mục đích:**

Dịch 0 vào trong bit 15 của word, dịch các bit của word về bên phải 1 bit và dịch bit 00 vào trong CY.

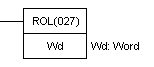


Hình 3.3. Dịch bit sang phải.

**Vùng dữ liệu toán tử:**

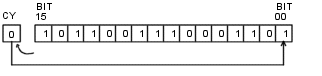
|  |  |  |
| --- | --- | --- |
| Wd: | Shift word | IO, AR, DM, HR, LR |
|  |  |  |
|  |  |  |

### 3.6.4. ROL(27) - Rotate Left



**Mục đích:**

Khi điều kiện thực hiện ngõ vào là OFF, ROL không thực hiện. Khi điều kiện thực hiện ngõ vào là ON, ROL dịch tất cả từng bít 1 của Wd sang trái. Bít được dịch vào 00 cho Wd và bít thứ 15 của Wd được dịch cho CY.



Hình 3.4. Rotate left.

**Vùng dữ liệu toán tử:**

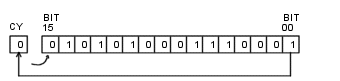
|  |  |  |
| --- | --- | --- |
| Wd: | Rotate word | IO, AR, DM, HR, LR |

### 3.6.5. ROR(28) - Rotate Right



**Mục đích:**

Khi điều kiện thực hiện ngõ vào là OFF, ROR không thực hiện. Khi điều kiện thực hiện ngõ vào là ON, ROR dịch tất cả từng bít 1 của Wd sang phải. Bit CY được dịch vào bit 15 cuaWd, bit 00 được dịch vào CY.

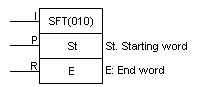


Hình 3.5. Rotate right.

**Vùng dữ liệu toán tử:**

|  |  |  |
| --- | --- | --- |
| Wd: | Rotate word | IO, AR, DM, HR, LR |

### 3.6.6. SFT(10) - Shift Register



**Mục đích:**

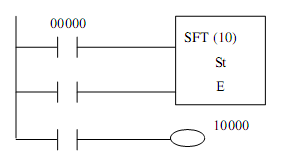
Lệnh SFT được điểu khiển bởi ba điều kiện thực hiện I, P và R. SFT thực hiện khi điều kiện cho P là ON trước đó là OFF và R là OFF. Khi đó điều kiện thực hiện I được dịch vào bít bên phải của một thanh ghi dịch đã định nằm trong St là E. Nếu I là ON thì [1] được dịch vào trong thanh ghi, nếu I là OFF thi [0] được dịch vào. Khi I được dịch vào trong thanh ghi, tất cả các bit trước trong thanh ghi sẽ được dịch sang trái và bit cuối cùng bên trái trong thanh ghi sẽ mất đi.

Nếu điều kiện thực hiện P không thay đổi ( hoặc ON hoặc OFF ) hay chuyển từ On sang OFF thì thanh ghi dịch sẽ không ảnh hưởng.

Nếu điều kiện thực hiện R là ON thì tất cả các bit trong thanh ghi dịch sẽ resetvà thanh ghi dịch sẽ không thực hiện cho đến khi R được OFF trở lại.

*Ví dụ:*  
 Ví dụ sau dùng xung clock 1 giây ( bit SR25502) cho ngõ vào Pp để dịch dữ liệu từ ngõ vào I cho IR010 để bật ON khi bit OUT 10000 khi bit IR01007 là một thông qua bit điều khiển IR00000.

Sơ đồ hình thang:



Chương trình được thực hiện như sau:

Địa chỉ lệnh toán tử

00000 LD 00000

00001 LD 25502

00002 LD 00001

00003 SFT (10)

010

010

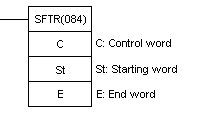
00004 LD 01007

00005 OUT 10000

**Vùng dữ liệu toán tử:**

|  |  |  |
| --- | --- | --- |
| St: | Starting Word | IO, AR, HR, LR |
| E: | End Word | IO, AR, HR, LR |

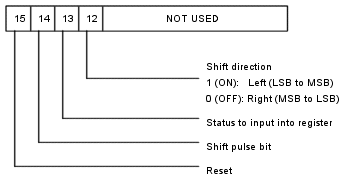
### 3.6.7. SFTR(84) - Reversible Shift Register



**Mục đích:**

St phải nhỏ hơn hay bằng E.

SFTR dùng để tạo thanh ghi dịch một hay nhiều word mà có thể dịch dữ liệu theo hai chiều là dịch trái hay dịch phải để tạo một thanh ghi một word tức là xác định St và E cùng word. Word điều khiển C cho biết hướng dịch, trạng thái nhập vào thanh ghi, xung dịch và ngõ vào reset, word điều khiển được định vị như sau:



Hình 3.6. Định vị word điều khiển.

Ghi chú: shift direction ( hướng dịch )

Status to input into register( trạng thái nhập vào thanh ghi)

Shift pulse bit (bit xung dịch )

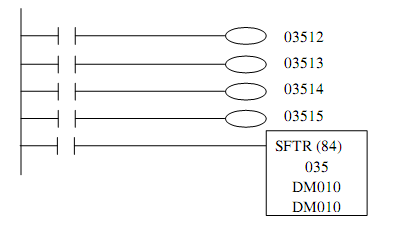
Dữ liệu trong thanh ghi dịch sẽ được dịch một bit theo hướng đã định bởi bit thứ 12 một bit dịch xuất cho CY và trạng thái của bit thứ 13 được chuyển cho bit cuối cùng mỗi khi SFTR được thực hiện với điều kiện thực hiện là ON, bit reset thì OFF và bit 14 thì ON.

Nếu SFTR được thực hiện với điều kiện thực hiện là OFF hay nếu SFTR thực hiện với bit 14 là OFF thanh ghi dịch sẽ duy trì trạng thái không thay đổi. Nếu thanh ghi dịch được thực hiện với điều kiện thực hiện là ON và bit reset (bit 15) là OFF thì toàn bộ thanh ghi dịch và cờ CY sẽ đặt về 0.

*Ví dụ:*

Trong ví dụ sau: IR00000, IR 00001, IR00002, IR00003 được dùng để điều khiển trạng thái của bit C trong SFTR thanh ghi dịch DM0010 được điều khiển thông qua IR00004.

Sơ đồ hình thang:



Viết chương trình cho sơ đồ hình thang:

Địa chỉ lệnh dữ liệu

00000 LD 00000

00001 OUT 03512

00002 LD 00001

00003 OUT 03513

00004 LD 00002

00005 OUT 03514

00006 LD 00003

00007 OUT 03515

00008 LD 00004

00009 SETR (10) 035

DM010

DM010

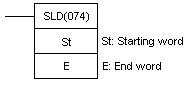
**Vùng dữ liệu toán tử:**

C: word điều khiển (control word) IO, AR, DM, HR, LR..

St: word bắt đầu (starting word) IO, AR, DM, HR, LR..

E: word cuối (end word) IO, AR, DM, HR, LR..

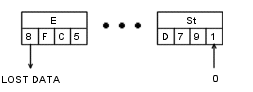
### 3.6.8. SLD(74) - One Digit Shift Left



**Mục đích:**

St và E phài cùng một vùng dữ liệu, E phải lớn hơn St.

Khi điều kiện thực hiện ngõ vào là OFF, SLD không thực hiện. Khi điều kiện thực hiện ngõ vào là ON, SLD dịch dữ liệu giữa St và E một số (4 bit) sang trái.Số 0 được ghi vào số bên phải của St và nội dung của số bên trái bị mất.



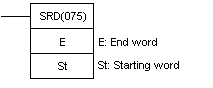
Hình 3.7. Dịch dữ liệu giữa St và E sang trái.

**Vùng dữ liệu toán tử:**

St: word bắt đầu IO, AR, DM, HR, LR.

E: word cuối IO, AR, DM, HR, LR.

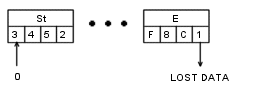
### 3.6.9. SRD(75) - One Digit Shift Right



**Mục đích:**

St và E phải cùng một vùng dữ liệu, E phải lớn hơn St.

Khi điều kiện thực hiện ngõ vào là OFF, SRD không thực hiện. Khi điều kiện thực hiện ngõ vào là ON, SRD dịch dữ liệu giữa St và E một số (4 bit) sang phải.Số 0 được ghi vào số bên trái của St và nội dung của số bên phải bị mất.



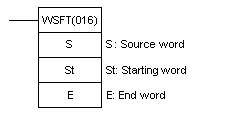
Hình 3.8. Dịch dữ liệu giaữ St và E một số sang phải.

**Vùng dữ liệu toán tử:**

St: word bắt đầu IO, AR, DM, HR, LR.

E: word cuối IO, AR, DM, HR, LR.

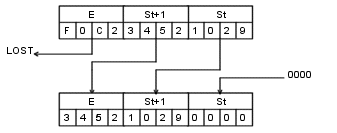
### 3.6.10. WSFT(16) - Word Shift



**Mục đích:**

St và E phải cùng một vùng dữ liệu, E phải lớn hơn St.

Khi điều kiện thực hiện ngõ vào là OFF, WSFT không thực hiện. Khi điều kiện thực hiện ngõ vào là ON, WSFT dịch dữ liệu trong từng word giữa ST và trong khối word. Số 0 được ghi vào St và nội dung của E bị mất.



Hình 3.9**.** Dịch dữ liệu trong từng word giữa ST và trong khối word.

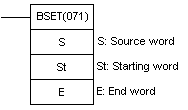
**Vùng dữ liệu toán tử:**

St : word bắt đầu IO, AR, DM, HR, LR.

E: word cuối IO, AR, DM, HR, LR.

## 3.7. Data Movement Instructions

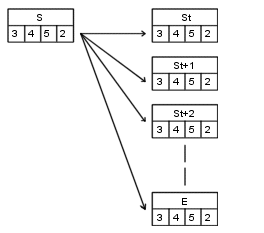
### 3.7.1. BSET(71) - Block Set



**Mục đích:**

St phải nhỏ hơn hay bằng E, St và E phải cùng một vùng dữ liệu.

Khi điều kiện thực hiện ngõ vào là OFF, BSET không thực hiện. Khi điều kiện thực hiện ngõ vào là ON, BSET chép nội dung của S cho tất cả các word từ St và E.



Hình 3.10. Chép nội dung của S cho tất cả các word từ St và E.

BSET có thể được dùng để chuyển giá trị hiện tai PV của timer và counter (điều này không thể thực hiện với lệnh MOV hay MVN).

**Vùng dữ liệu toán tử:**

S: dữ liệu nguồn IO, AR, DM, HR, TC, LR, #.

St: word bắt đầu IO, AR, DM, HR, TC, LR.

E: word cuối IO, AR, DM, HR, TC, LR.

### 3.7.2. BXF2(--) - EM Bank Transfer



**Mục đích:**

BXF2 sao chép nội dung của S, S+1, …, S+N đến D, D+1, .., D+N. Nếu một số được sử dụng cho S và D, số đó định rõ một địa chỉ trong nguồn hoặc nơi đưa tới nhóm EM đã định rõ trong C.

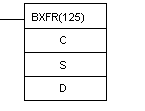
C chứa nguồn và nhóm các số đưa tới nếu dữ liệu được dời chổ tới hoặc từ EM. Nhóm các số được bác bỏ trừ phi hằng số được sử dụng cho S và D.

S+1 chứa số của word để chuyển và phải là BCD (1 tới 6144).

**Nguồn dữ liệu toán tử:**

|  |  |  |
| --- | --- | --- |
| C: | First control word | IO, AR, DM, EM, HR, TC, LR, # |
| S: | Starting source word | IO, AR, DM, EM, HR, TC, LR, # |
| D: | Starting destination word | IO, AR, DM, EM, HR, TC, LR, # |

### 3.7.3. BXFR(125) - Interbank Block Transfer



**Mục đích:**

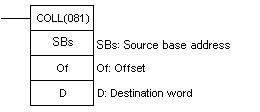
Khi trạng thái thực hiện là OFF, BXFR không thực hiện. Khi trạng thái thực hiện là ON, BXFR di chuyển các word định rõ một cách liên tục từ nhóm nguồn đến phần đầu cùng với word định rõ trong nhóm được định rõ.

Số các word để di chuyển và nhóm các số được đặt như dữ liệu BCD trong hai word điều khiển( C và C+1).

**Vùng dữ liệu toán tử:**

|  |  |  |
| --- | --- | --- |
| C: | First control word | IO, AR, DM, EM. HR, TC, LR |
| S: | First source word | IO, AR, DM, EM, HR, TC, LR |
| D: | First destination word | IO, AR, DM, HR, TC, LR |

### 3.7.4. COLL(81) - Data Collect



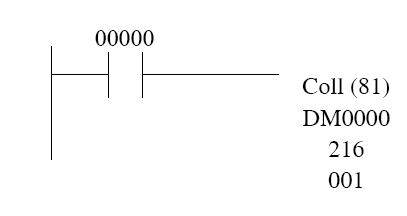
**Mục đích:**

Of (C) phải là BCD.  
 Lệnh COLL được dùng để thu thập dữ liệu hoạt động theo kiểu xếp chồng vào trước ra trước FIFO (First In First Out) hay kiểu vào sau ra trước FILO (First In Last Out) tùy theo nội dung của word C.

Khi bit 12 tới bit 15 của word C từ 0 tới 7, lệnh COLL được dùng để chọn dữ liệu. Toàn bộ nội dung của C xác định phần bù , kí hiệu là Of. Khi điều kiện thực hiện ngõ vào là OFF lệnh COLL không thực hiện, khi điều kiện thực hiện ngõ vào là ON thì lệnh COLL chép nội dung word nguồn cho D. Word nguồn có địa chỉ là địa chỉ của SBs cộng với nội dung của Of. SBs và SBs+Of phải cùng vùng dữ liệu.

*Ví dụ:*  
 Trình bày cách sử dụng lệnh COLL để tạo một ngăn xếp giữa DM0001 và DM 0005. DM0000 hoạt động như một ngăn xếp.  
 Khi IR00000 chuyển từ OFF sang ON, lệnh COLL chép nội dung của DM0005 ( DM0000+5) cho IR 001, nội dung của con trỏ ngăn xếp sau đó giảm đi một.

Sơ đồ hình thang:



Chương trình của sơ đồ hình thang trên:

Địa chỉ lệnh toán tử

0000 LD 00000

0001 Coll (81) DM0000

216

001

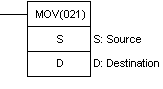
**Vùng dữ liệu toán tử:**

SBs: word nguồn gốc IO, AR, DM, HR, TC, LR.

Of: offset IO, AR, DM, HR, TC, LR, #.

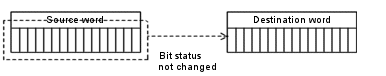
D: word đích IO, AR, DM, HR, TC, LR.

### 3.7.5. MOV(21) - Move



**Mục đích:**

Khi điều kiện thực hiện ngõ vào là OFF thì lệnh MOV không thực hiện. Khi điều kiện ngõ vào là ON thì lệnh MOV sẽ chép nội dung của S sang D.



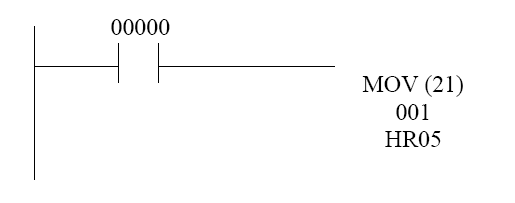
Hình 3.11. Chép nội dung của S sang D.

Bit status not changed: trạng thái bit không thay đổi.

*Ví dụ:*

Sau khi trình bày sự thực hiện của lệnh MOV chép nội dung của IR 001 cho HR 05 khi IR00000 chuyển từ OFF sang ON.

Sơ đồ hình thang:



Chương trình được viết như sau:

Địa chỉ lệnh toán tử

00000 LD 00000

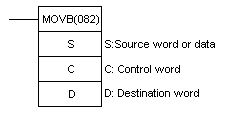
00001 MOV (21) 01

HR05

**Vùng dữ liệu toán tử:**

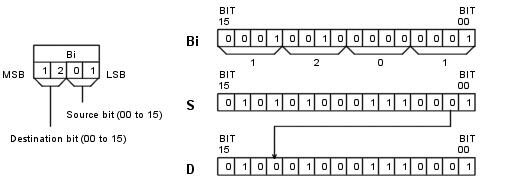
|  |  |  |
| --- | --- | --- |
| S: | Source word | IO, AR, DM, HR, TC, LR, #. |
| D: | Destination word | IO, AR, DM, HR, LR. |

### 3.7.6. MOVB(82) - Move Bit



**Mục đích:**

Hai số bên phải và hai số bên trái của B1 phải ở trong khoảng từ 00 tới 15.  
Khi điều kiện thực hiện ngõ vào là OFF lệnh MOVB không được thực hiện. Khi điều kiện thực hiện ngõ vào là ON lệnh MOVB chép nội dung đã định của bit trong S cho bit đã định trong D. Bit trong S và D được xác định bởi Bi hai số bên phải bit chỉ định bit nguồn và hai số bên trái chỉ định bit đích.

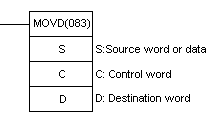


Hình 3.12. Chép nội dung đã định của bit trong S cho bit đã định trong D.

**Vùng dữ liệu toán tử:**

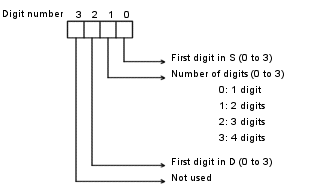
|  |  |  |
| --- | --- | --- |
| S: | Source word | IO, AR, DM, HR, LR, #. |
| C: | Control Word | IO, AR, DM, HR, TC, LR, #. |
| D: | Destination word | IO, AR, DM, HR, LR. |

### 3.7.7. MOVD(83) - Move Digit



**Mục đích:**

Sao chép nội dung của kí số xác định trong S đến kí số xác định trong D. Kí số đầu tiên để sao chép, số của các kí số để sao chép và kí số để nhận việc sao chép được định rõ trong Di. Các kí số từ S sẽ được sao chép đến các kí số liên tiếp trong D từ kí số đầu tiên và liên tiếp cho đến số các kí số đã định rõ.

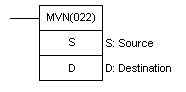


Hình 3.13. Sao chép nội dung của kí số xác định trong S đến kí số xác định trong D

**Vùng dữ liệu toán tử:**

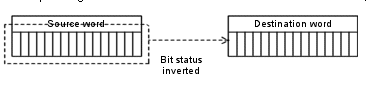
|  |  |  |
| --- | --- | --- |
| S: | Source word | IO, AR, DM, HR, TC, LR, #. |
| C: | Control Word | IO, AR, DM, HR, TC, LR, #. |
| D: | Destination word | IO, AR, DM, HR, TC, LR. |

### 3.7.8. MVN(22) - Move NOT



**Mục đích:**

Khi điều kiện thực hiện ngõ vào là ON thì lệnh MVN sẽ truyền nội dung đảo của S cho D. Mỗi bit ON của S thì tương ứng là với bit OFF trong D và bit OFF trong S tương ứng với bít ON trong D.



Hình 3.14. Truyền nội dung đảo của S cho D.

**Vùng dữ liệu toán tử:**

|  |  |  |
| --- | --- | --- |
| S: | Source word(word nguồn) | IO, AR, DM, HR, TC, LR, #. |
| D: | Destination word(word đích) | IO, AR, DM, HR, LR. |

### 3.7.9. XCHG(73) - Data Exchange



**Mục đích:**

Trao đổi nội dung của E1 và E2



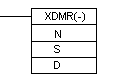
Hình 3.15. Trao đổi nội dung của E1 và E2.

Nếu muốn trao đổi nội dung của các khối mà kích cỡ của nó lớn hơn 1 word, thì sử dụng các word làm việc như bộ đệm trung gian để giữ một trong số các khối.

**Vùng dữ liệu toán tử:**

|  |  |  |
| --- | --- | --- |
| E1: | Exchange word 1 | IO, AR, DM, HR, TC, LR. |
| E2: | Exchange word 1 | IO, AR, DM, HR, TC, LR. |

### 3.7.10. XDMR(--) - Expansion DM Read



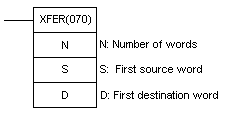
**Mục đích:**

Khi trạng thái thực hiện là OFF, XDMR không thực hiện. Khi trạng thái thực hiện là ON, XDMR sao chép nội dung của các word mở rộng DM S tới S+N-1 đến word đích D tới D+N-1.

**Vùng dữ liệu toán tử:**

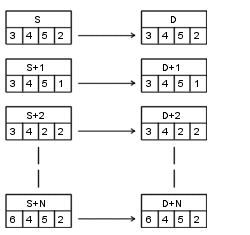
|  |  |  |
| --- | --- | --- |
| N: | Number of words | IO, AR, DM, HR, TC, LR, #. |
| S: | 1st exp. DM word | IO, AR, DM, HR, TC, LR, #. |
| D: | 1st destination word | IO, AR, DM, HR, LR. |

### 3.7.11-XFER(70) - Block Transfer



**Mục đích:**

Sao chép của S, S+1, …, S+N-1 đến D, D+1 ,…, D+N-1.

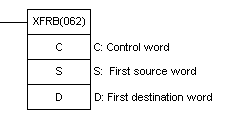


Hình 3.16. Sao chép của S, S+1, …, S+N-1 đến D, D+1 ,…, D+N-1.

**Vùng dữ liệu toán tử:**

|  |  |  |
| --- | --- | --- |
| N: | Number of words | IO, AR, DM, HR, TC, LR, #. |
| S: | 1st source word | IO, AR, DM, HR, TC, LR. |
| D: | 1st destination word | IO, AR, DM, HR, TC, LR. |

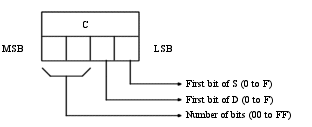
### 3.7.12. XFRB(--) - Transfer Bits



**Mục đích:**

Những bit nguồn phải cùng vùng dữ liệu.

Khi thực hiện ngõ vào là OFF, lệnh XFRB không thực hiện. Khi thực hiện ngõ vào là ON thì lệnh XFRB chép những bit nguồn đã định của bit S thành những bit đích đã định trong D. Hai số bên phải của C xác định bit bắt đầu trong S và D hai số bên trái xác định số bit sẽ được chép.

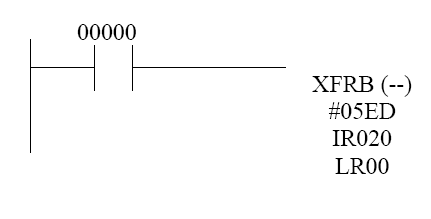


Hình 3.17. Chép những bit nguồn đã định của bit S thành những bit đích đã định trong D.

*Ví dụ:*

Trong ví dụ lệnh XFRB được dùng để truyền 5 bit từ IR020 và IR021 cho LR 00 và LR 01. Bít bắt đầu trong IR020 là D (số 13) và bít bắt đầu trong LR00 là E (số 14).

Lưu đồ hình thang:



Chương trình của lưu đồ:

Địa chỉ lệnh toán tử

0000 LD 00001

0001 XFRB (--) #05ED

020

00

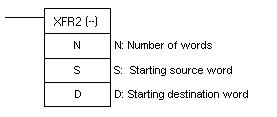
**Vùng dữ liệu toán tử:**

C: word nguồn IO, AR, DM, HR, TC, LR, #.

S:word nguồn đầu tiên IO, AR, DM, HR, TC, LR.

D: word đich đầu tiên IO, AR, DM, HR, LR.

### 3.7.13. XFR2(--) – EM Block Transfer



**Mục đích:**

XFR2 Sao chép của S,S+1, …, S+N đến D, D+1 ,…, D+N.Nếu một số được dùng cho S hay D, số đó sẽ định rõ một địa chỉ trong khối EM hiện thời.

**Vùng dữ liệu toán tử:**

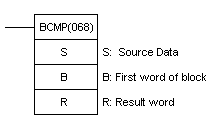
N:số word (BCD) IO, AR, DM, HR, TC, LR, #.

S:word nguồn bắt đầu IO, AR, DM, HR, TC, LR, #.

D:word đích bắt đầu IO, AR, DM, HR, TC, LR, #.

## 3.8. Data Comparison Instructions

### 3.8.1. BCMP(68) - Block Compare



**Mục đích:**

Khi điều kiện thực hiện ngõ vào là OFF, BCMP không thực hiện. Khi điều kiện thực hiện ngõ vào là ON, BCMP so sanh CD với dãy định nghĩa bởi một khối CB, CB+1, CB+2,…, CB+31, mỗi dãy được định nghĩa bởi hai word, word đầu là word giới hạn,word thứ hai là giới hạn trên. Nếu dữ liệu CD nằm trong khoảng này thì bit tương ứng trong R được xếp cho mỗi lần so sánh đúng. Những bit còn lại trong R sẽ được reset về 0 (bật về OFF).

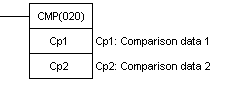
Bảng 3.2. Mô tả cách thực hiện lệnh so sánh khối.

|  |  |
| --- | --- |
| CB =< CD =< CB+1 | Bit 00 of R |
| CB+2 =< CD =< CB+3 | Bit 01 of R |
| CB+4 =< CD =< CB+5 | Bit 02 of R |
| CB+6 =< CD =< CB+7 | Bit 03 of R |
| CB+8 =< CD =< CB+9 | Bit 04 of R |
| CB+10 =< CD =< CB+11 | Bit 05 of R |
| CB+12 =< CD =< CB+13 | Bit 06 of R |
| CB+14 =< CD =< CB+15 | Bit 07 of R |
| CB+16 =< CD =< CB+17 | Bit 08 of R |
| CB+18 =< CD =< CB+19 | Bit 09 of R |
| CB+20 =< CD =< CB+21 | Bit 10 of R |
| CB+22 =< CD =< CB+23 | Bit 11 of R |
| CB+24 =< CD =< CB+25 | Bit 12 of R |
| CB+26 =< CD =< CB+27 | Bit 13 of R |
| CB+28 =< CD =< CB+29 | Bit 14 of R |
| CB+30 =< CD =< CB+31 | Bit 15 of R |

**Vùng dữ liệu toán tử:**

|  |  |  |
| --- | --- | --- |
| CD: | Compare data | IO, AR, DM, HR, TC, LR, #. |
| CB: | 1st comparison block word | IO, AR, DM, HR, TC, LR. |
| R: | Result word | IO, AR, DM, HR, TC, LR |

### 3.8.2. CMP(20) - Compare



**Mục đích:**

Lệnh CMP dùng để so sánh hai dữ liệu CP1 và CP2 và xuất kết quả ra các cờ GR, EQ, và LE trong vùng SR.

EQ:ON nếu CP1=CP2

LE:ON nếu CP1<CP2

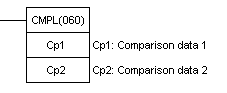
GR:ON nếu CP1>CP2

**Vùng dữ liệu toán tử:**

CP1: word so sánh thứ nhất IO, AR, DM, HR, TC, LR, #.

Cp2: word so sánh thứ hai IO, AR, DM, HR, TC, LR, #.

### 3.8.3. CMPL(60) - Double Compare



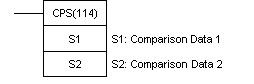
**Mục đích:**

Nhập vào nội dung hệ thập lục phân 4 kí số của CP1+1 cùng với nội dung đó của CP1,tương tự ta cũng làm như vậy với CP2+1 và Cp2 để tạo thành hai số hệ thập lục phân 8 kí số,CP1+1, CP1 và CP2+1 ,CP2.Hai số 8 kí số được so sánh sau đó và kết quả xuất ra đến cờ GR, EQ và LE trong vùng SR.

**Vùng dữ liệu toán tử:**

|  |  |  |
| --- | --- | --- |
| Cp1: | 1st word of first compare word pair | IO, AR, DM, HR, TC, LR. |
| Cp2: | 1st word of second compare word pair | IO, AR, DM, HR, TC, LR. |

### 3.8.4. CPS(--) - Signed Binary Compare



**Mục đích:**

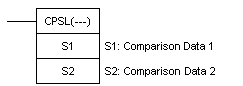
So sánh nội dung nhị phân 16 bit (4 kí số) trong CP1 và CP2 và xuất kết quả đến cờ GR , EQ, và LE trong vùng SR.

**Vùng toán tử dữ liệu:**

CP1: word so sánh thứ nhất IO, AR, DM, HR, TC, LR, #.

CP2: word so sánh thứ hai IO, AR, DM, HR, TC, LR, #.

### 3.8.5. CPSL(--) - Double Signed Binary Compare



**Mục đích:**

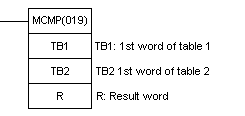
So sánh nội dung nhị phân 32 bit (8 kí số) trong CP1+1,CP1 và CP2+1,CP2 và xuất kết quả đến cờ GR , EQ, và LE trong vùng SR.

**Vùng dữ liệu toán tử:**

CP1: word so sánh thứ nhất IO, AR, DM, HR, TC, LR.

CP2: word so sánh thứ hai IO, AR, DM, HR, TC, LR.

### 3.8.6. MCMP(19) - Multi-Word Compare



**Mục đích:**

TB1 và TB1+15 phải cùng dữ liệu.

TB2 và TB2+15 phải cùng dữ liệu.

Khi điều kiện thực hiện ngõ vào là OFF, lệnh MCPM không thực hiện. Khi điều kiện thực hiện ngõ vào là ON, lệnh MCMP dùng để so sánh giá trị của TB1 với TB2, TB1+1 với TB2+2, …, TB1+15 với TB2+15.

Nếu nội dung của TB1 bằng nội dung của TB2 thì bit 00 của R sẽ được bật về OFF và nếu nội dung của word TB1+1 bằng nội dung của TB2+1 thì bit 01 của R sẽ được bật về OFF và tương tự cho những bit tiếp theo, word nào so sanh không bằng nhau thì bit tương ứng trong R được bật ON.

Cờ EQ On khi toàn bộ nội dung của hai bản là bằng nhau tức là R=0000.

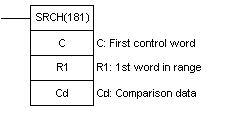
**Vùng dữ liệu toán tử:**

TB1: word đầu tiên của bảng 1

TB2:word so sanh của bảng 2

R: word kết quả

### 3.8.7. SRCH(--) - Data Search



**Mục đích:**

SRCH tìm kiếm dãy của bộ nhớ từ R1 đến R1+N-1 cho địa chỉ mà chứa dữ liệu so sánh trong C.Nếu một hoặc nhiều địa chỉ chứa dữ liệu so sánh, thì cờ EQ được bật ON và địa chỉ thấp nhất chứa dữ liệu so sánh được định rõ trong C+1. Địa chỉ định rõ được sự khác nhau cho vùng DM:

★Cho một địa chỉ trong vùng DM, địa chỉ word được ghi vào C+1. Ví dụ, nếu địa chỉ thấp nhất chứa dữ liệu so sánh là DM 0114, sau đó #0114 được ghi vào trong C+1.

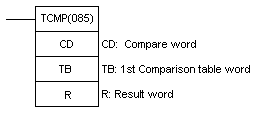
★Cho một địa chỉ trong vùng dữ liệu khác, các địa chỉ từ việc tìm kiếm ban đầu được ghi vào trong C+1. Ví dụ như một địa chỉ thấp nhất chứa dữ liệu so sánh là IR 114 thì word đầu tiên trong dãy tìm kiếm là IR 014 sau đó #0100 được ghi vào trong C+1.

Nếu không có địa chỉ nào trong dãy chứa dữ liệu so sánh thì cờ EQ được bật lên ON.

**Vùng dữ liệu toán tử:**

|  |  |  |
| --- | --- | --- |
| N: | Number of words | IO, AR, DM, HR, TC, LR, # |
| R1: | 1st word in range | IO, AR, DM, HR, TC, LR |
| C: | Comparison data, result word | IO, AR, DM, HR, LR |

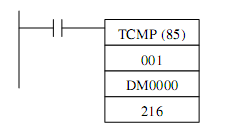
### 3.8.8. TCMP(85) - Table Compare



**Mục đích:**

Khi điều kiện thực hiện ngõ vào là OFF thì lệnh TCMP không thực hiện, khi điều kiện thực hiện ngõ vào là ON, lệnh TCMP so sánh dữ liệu CD với nội dung TB, TB+1, TB+2,…, TB+15. Nếu nội dung của CD bằng nội dung của word này tương ứng với bit trong R được đặt lên 1. Nếu nội dung của CD bằng nội dung của TB thì bit 00 của R được bật lên ON, tương tự nếu CD bằng với TB+1 thì bit 01 của R được bật lên ON. Đồng thời tất cả các bit còn lại trong R sẽ được reset về 0.

*Ví dụ:*



Chương trình của sơ đồ hình thang:

Địa chỉ lệnh toán tử

00000 LD 00000

00001 TCMP(85)

001

DM0000

216

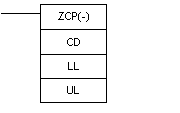
**Vùng dữ liệu toán tử:**

CD: dữ liệu so sánh IO, AR, DM, HR, TC, LR, #.

TB: word đầu tiên trong bản so sánh IO, AR, DM, HR, TC, LR.

R: word kết quả IO, AR, DM, HR, TC, LR.

### 3.8.9. ZCP(--) - Area Range Control

****

**Mục đích:**

So sánh CD với vùng đã định rõ bởi giới hạn LL ở vị trí thấp hơn và giới hạn UL ờ vị trí cao hơn và kết quả xuất ra đến các cờ GR, EQ và LE trong vùng SR.

Bảng 3.3. Cờ kết quả

|  |  |  |  |
| --- | --- | --- | --- |
| Comparison result | Flag Status | | |
|  | GR (SR 25505) | EQ (SR 25506) | LE (SR 25507) |
| CD < LL | 0 | 0 | 1 |
| LL < CD < UL | 0 | 1 | 0 |
| UL < CD | 1 | 0 | 0 |

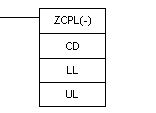
**Vùng dữ liệu toán tử:**

CD: dữ liệu so sánh IO, AR, DM, HR, TC, LR, #.

LL: vùng giới hạn thấp hơn IO, AR, DM, HR, TC, LR, #.

UL: vùng giới hạn cao hơn IO, AR, DM, HR, TC, LR, #.

### 3.8.10. ZCPL(--) - Double Area Range Compare



**Mục đích:**

So sánh giá trị 8 kí số trong CD+1, CD với dãy đã định rõ bởi giới hạn LL+1, LL vị trí thấp hơn và giới hạn UL+1, UL vị trí cao hơn, và xuất kết quả đến các cờ GR, EQ, và LE trong vùng SR.

Bảng 3.4. Cờ kết quả.

|  |  |  |  |
| --- | --- | --- | --- |
| Comparison result | Flag Status | | |
|  | GR (SR 25505) | EQ (SR 25506) | LE (SR 25507) |
| CD, CD+1 < LL+1, LL | 0 | 0 | 1 |
| LL+1, LL < CD+1, CD < UL+1, UL | 0 | 1 | 0 |
| UL+1, UL < CD+1, CD | 1 | 0 | 0 |

**Vùng dữ liệu toán tử:**

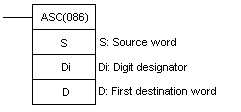
CD: dữ liệu so sánh IO, AR, DM, HR, LR.

LL: vùng giới hạn thấp hơn IO, AR, DM, HR, LR.

UL: vùng giới hạn cao hơn IO, AR, DM, HR, LR.

## 3.9. Data conversion instruction

### 3.9.1. ASC(86) - ASCII Convert



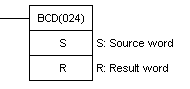
**Mục đích :**

Biến đổi kí số đã chỉ định của S thành mã ASCII 8 bit tương đương và đặt nó vào trong word gởi tới bắt đầu cùng với D. Bất kì hoặc tất cả các kí số trong S có thể được biến đổi theo thứ tự từ kí số đầu tiên đã định rõ.

**Vùng dữ liệu toán tử:**

|  |  |  |
| --- | --- | --- |
| S: | Source word | IO, AR, DM, HR, TC, LR |
| Di: | Digit designator | IO, AR, DM, HR, TC, LR, # |
| D: | 1st destination word | IO, AR, DM, HR, LR |

### 3.9.2. BCD(24) - Binary-To-BCD



**Mục đích:**

Nội dung của S không được vượt quá 270 F. Nếu nội dung của S >270F, khi biến đổi kết quả sẽ lớn hơn 9999.

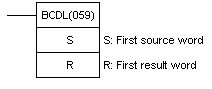
Lệnh BCD (24) dùng để biến nội dung của S dạng nhị phân hay thập phân sang BCD tương ứng và đặt trong R.

**Vùng dữ liệu toán tử:**

S: word nguồn IO, AR, DM, HR, LR.

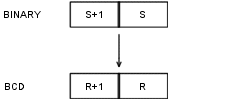
R: word kết quả IO, AR, DM, HR, LR.

### 3.9.3. BCDL(59) - Double Binary-To-Double BCD



**Mục đích:**

BCDL biến đổi nội dung nhị phân 32 bit của S và S+1 thành 8 kí số của BCD tương ứng và xuất dữ liệu biến đổi đến R và R+1.



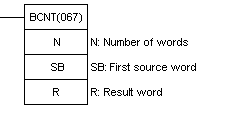
Hình 3.18. **B**iến đổi nội dung nhị phân 32 bit của S và S+1 thành 8 kí số của BCD.

**Vùng dữ liệu toán tử:**

S:word nguồn đầu tiên IO, AR, DM, HR, LR.

R:word kết quả đầu tiên IO, AR, DM, HR, LR.

### 3.9.4. BCNT(67) - Bit Counter



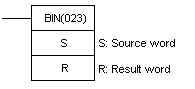
**Mục đích:**

BCNT đếm toàn bộ số của các bit mà là ON trong tất cả các word giữa SB và SB+N-1 và đặt kết quả vào trong R. N phải là BCD ở giữa 0000 và 6656.

**Vùng dữ liệu toán tử:**

|  |  |  |
| --- | --- | --- |
| N: | Number of words | IO, AR, DM, HR, TC, LR, #. |
| SB: | 1st source word | IO, AR, DM, HR, TC, LR. |
| R: | Result word | IO, AR, DM, HR, TC, LR. |

### 3.9.5. BIN(23) - BCD-To-Binary



**Mục đích:**

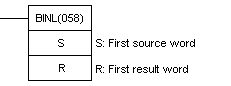
Khi điều kiện thực hiện ngõ vào là OFF, lệnh BIN(23) không thực hiện. Khi điều kiện thực hiện ngõ vào là ON, lệnh BIN(23) biến đổi nội dung của BCD của S sang bit nhị phân tương đương và kết quả xuất ra vào R.

**Vùng dữ liệu toán tử:**

S: word nguồn IO, AR, DM, HR, TC, LR.

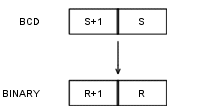
R: word kết quả IO, AR, DM, HR, LR.

### 3.9.6. BINL(58) - Double BCD to Double Binary



**Mục đích:**

Biến đổi 8 kí số trong S và S+1 thành dữ liệu nhị phân 32 bit và xuất dữ liệu nhị phân biến đổi đến R và R+1.

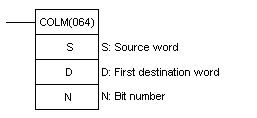


Hình 3.19. Biến đổi 8 kí số trong S và S+1 thành dữ liệu nhị phân 32 bit.

**Vùng dữ liệu toán tử:**

|  |  |  |
| --- | --- | --- |
| S: | 1st source word | IO, AR, DM, HR, TC, LR |
| R: | 1st result word | IO, AR, DM, HR, LR |

### 3.9.7. COLM(64) - Line-To-Column



**Mục đích:**

Sao chép 16 bit của word S ( 00 tới 15) tới cột của các bit, C, của bộ 16 word (D tới D+15).

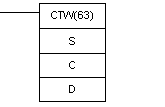
**Vùng dữ liệu toán tử:**

S: word nguồn đầu tiên IO, AR, DM, HR, TC, LR.

D: word đích đầu tiên IO, AR, DM, HR, TC, LR.

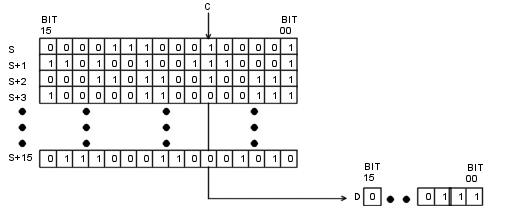
N: cột bit được định rõ (BCD) IO, AR, DM, HR, TC, LR,#.

### 3.9.8. CTW(--) - Column-to-Word



**Mục đích:**

Sao chép cột bit C từ bộ 16 word (S tới S+15) đến 16 bit của word D( 00 tới 15).

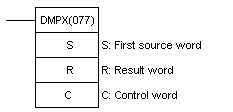


Hình 3.20. Sao chép cột bit C từ bộ 16 word (S tới S+15) đến 16 bit của word D( 00 tới 15).

**Vùng dữ liệu toán tử:**

|  |  |  |
| --- | --- | --- |
| S: | 1st word of 16-word source set | IO, AR, DM, HR, LR |
| C: | Column bit designator (BCD) | IO, AR, DM, HR, TC, LR, # |
| D: | Destination word | IO, AR, DM, HR, LR |

### 3.9.9. DMPX(77) - 16-To-4 Encoder

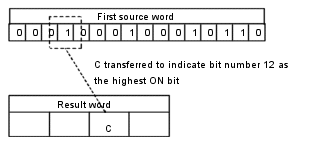


**Mục đích:**

Hai số bên phải của Di mỗi số có gia trị từ 0 đến 3.

Tất cả các word R phải cùng vùng dữ liệu.

Khi điều kiện thực hiện ngõ vào là OFF thì lệnh DMPX không thực hiện. Khi điều kiện thực hiện ngõ vào là ON, lệnh DMPX xác định giá trị hexa tương ứng với vị trí bit ON, sau đó chuyển giá trị hexa để xác định số trong R. Số nhận trong R cũng như số lượng số nhận do Di xác định.

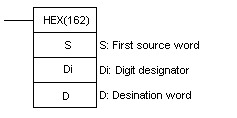


Hình 3.21. Xác định giá trị hexa.

**Vùng dữ liẹu toán tử:**

|  |  |  |
| --- | --- | --- |
| S: | 1st source word | IO, AR, DM, HR, TC, LR |
| R: | Result word | IO, AR, DM, HR, LR |
| C: | Control word | IO, AR, DM, HR, TC, LR, # |

### 3.9.10. HEX(--) - ASCII-To-Hexadecimal



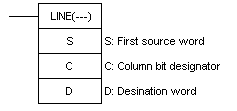
**Mục đích:**

Biến đổi byte định rõ đối với Mã ASCII từ word nguồn thành hexa tương ứng và đặt nó vào trong D. Mã ASCII (up to 4) có thể được biến đổi bắt đầu cùng với byte được định rõ trong S. Giá trị hexa được biến đổi thành được đặt vào trong D từ kí số đã định. Byte đầu tiên (8 bit bên trái hay 8 bit bên phai), số các byte được biến đổi và kí số của D nhận giá trị hexa đầu tiên được định rõ trong Di.

**Vùng dữ liệu toán tử:**

|  |  |  |
| --- | --- | --- |
| S: | Source word | IO, AR, DM, HR, TC, LR, # |
| Di: | Digit designator | IO, AR, DM, HR, TC, LR, # |
| D: | 1st destination word | IO, AR, DM, HR, LR |

### 3.9.11. LINE(--) - Column-To-Line



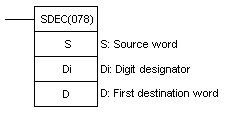
**Mục đích:**

Sao chép cột bit C từ bộ 16 word (S tới S+15 ) tới word 16 bit D (00 tới 15).

**Vùng dữ liệu toán tử:**

|  |  |  |
| --- | --- | --- |
| S: | 1st word of source set | IO, AR, DM, HR, TC, LR. |
| C: | Column bit designator | IO, AR, DM, HR, TC, LR, #.. |
| D: | Destination word | IO, AR, DM, HR, TC, LR. |

### 3.9.12. SDEC(78) - 7-Segment Decoder



**Mục đích:**

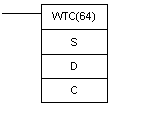
Biến đổi kí số đã định rõ của S thành 8 bit tương ứng, mã 7 đoạn bày ra và đặt vào trong word nhận bắt đầu cùng với D.

Bất kì hay tất cả các kí số trong S có thể được biến đổi trong chuổi từ kí số đầu tiên đã định. Kí số đầu tiên, nhóm cúa các kí số được biến đổi và môt nữa của D nhận mã hiển thị 7 đoạn đầu tiên (8 bit bên trái hay 8 bit bên phải) được định rõ trong Di.

**Vùng dữ liệu toán tử:**

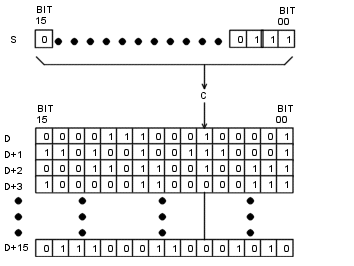
|  |  |  |
| --- | --- | --- |
| S: | Source word | IO, AR, DM, HR, TC, LR. |
| Di: | Digit designator | IO, AR, DM, HR, TC, LR, #. |
| D: | 1st destination word | IO, AR, DM, HR, LR. |

### 3.9.13. WTC(64) - Word-to-Column



**Mục đích:**

Sao chép 16 bit của word S (00 tới 15) đến cột của các bit, C, của bộ 16 word (D tới D+15).

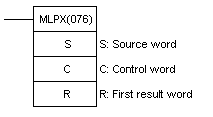


Hình 3.22. Sao chép bit từ word tới cột.

**Vùng dữ liệu toán tử:**

|  |  |  |
| --- | --- | --- |
| S: | Source word | IO, AR, DM, HR, LR. |
| D: | 1st word of the destination set | IO, AR, DM, HR, LR. |
| C: | Column bit designator | IO, AR, DM, HR, TC, LR, #. |

### 3.9.14. MLPX(76) - 4-To-16 Decoder



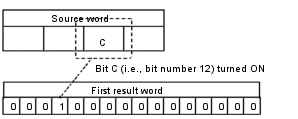
**Mục đích:**

Hai số bên phải của Di mỗi số có giá trị từ 0 đến 3.

Tất cả các word phải cùng dữ liệu.

Khi điều kiện thực hiện ngõ vào là OFF, lệnh MLPX không thực hiện. Khi điều kiện thực hiện ngõ vào là ON, lệnh MLPX biến đổi tối đa 4 số hexa trong S sang giá trị Decimal từ 0 đến 15. Mỗi giá trị sau khi biến đổi xác định vị trí bit trương ứng trong word kết quả và bật ON bit đó.

Thanh ghi Di cho biết số đầu tiên và số lượng số được biến đổi của word S.Cho ví dụ, Di sẽ là 0001



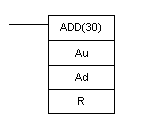
Hình 3.23. Biến đổi số hexa sang giá trị Decimal.

**Vùng dữ liệu toán tử:**

|  |  |  |
| --- | --- | --- |
| S: | Source word | IO, AR, DM, HR, TC, LR. |
| C: | Control word | IO, AR, DM, HR, TC, LR, #. |
| R: | 1st result word | IO, AR, DM, HR, LR. |

## 3.10. BCD Calculation instructions

### 3.10.1. ADD(30) - BCD Add



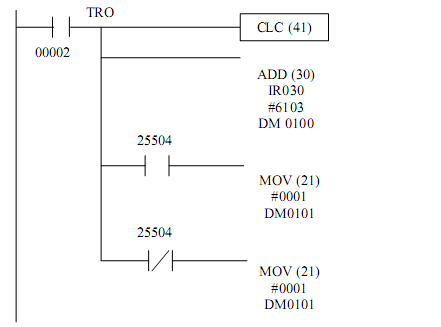
**Mục đích:**

Khi điều kiện thực hiện ngõ vào là OFF, lệnh ADD không thực hiện. Khi điều kiện thực hiện ngõ vào là ON, lệnh ADD công nội dung của Au, Ag, CY và đặt kết quả vào trong R. CY sẽ đặt lên 1 nếu kết quả lớn hơn 9999.

*Ví dụ:*

Trong ví dụ được lập trình (xem hình) nếu 00002 mà ON thì chương trình sẽ xóa CY với CLC(41) và cộng nội dung IR030 với hằng số 6103 đặt kết quả vào trong DM0100 và sau đó di chuyển 0000 hay 0001 vào DM0101 tùy thuộc vào trạng thái CY (SR25504). Điều này để đảm bảo rằng bất kì carry nào từ số sau cùng cũng được giữ trong R+1 vì thế toàn bộ kết quả sau cùng được xem như là một dữ liệu tám số.

Sơ đồ hình thang:



Viết chương trình cho sơ đồ hình thang trên:

Địa chỉ lệnh dữ liệu

00000 LR 00002

00001 OUT TRO 0

00002 CLC (41)

00003 ADD (30)

030

#6103

DM0100

00004 AND 25504

00005 MOV (21)

0001

DM0101

00006 LD TR0

00007 AND NOT 25504

00008 MOV (21)

#0000

DM0101

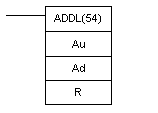
**Vùng dữ liệu toán tử:**

Au: augen word (BCD ) IO, AR, DM, HR, TC, LR, #.

Ad: addend word( BCD) IO, AR, DM, HR, TC, LR, #.

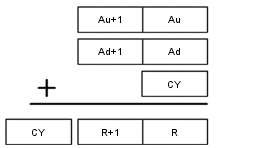
R: word kết quả IO, AR, DM, HR, LR.

### 3.10.2. ADDL(54) - Double BCD Add



**Mục đích:**

ADDL cộng nội dung của CY với giá trị 8 kí số trong Au và Au+1 và kết quả được đặt vào trong R và R+1.Cy sẽ được đặt lên nếu kết quả lớn hơn 99999999.



Hình 3.24. Cộng nội dung của CY với giá trị trong Au và Au+1.

**Vùng dữ liệu toán tử:**

Au: augen word (BCD ) IO, AR, DM, HR, TC, LR.

Ad: addend word( BCD) IO, AR, DM, HR, TC, LR.

R: word kết quả IO, AR, DM, HR, LR.

### 3.10.3. DEC(39) - Decrement



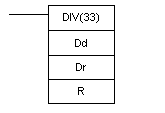
**Mục đích:**

Khi điều kiện thực hiện ngõ vào từ OFF sang ON lệnh này dùng để giảm word Wd đi môt đơn vị, lệnh này không ảnh hưởng đến cờ CY.

**Vùng dữ liệu toán tử:**

Wd: word giảm IO, AR, DM, HR, LR.

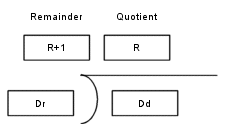
### 3.10.4. DIV(33) - BCD Divide



**Mục đích:**

R và R+1 cùng vùng dữ liệu .

Khi điều kiện thực hiện ngõ vào là OFF, lệnh DIV không thực hiện và chương trình chuyển tới lệnh kế tiếp. Khi điều kiện thực hiện ngõ vào là ON , lệnh DIV được thực hiện Dd chia cho Dr và kết quả cất trong R và R+1. Thương số được cất trong R và số dư cất trong R+1.



Hình 3.25. Thực hiện Dd chia cho Dr.

*Ví dụ:*

Khi IR00000 mà ON theo lập trình sau, nội dung của IR216 được chia cho nội dung của HR09 và kết quả cất trong DM0018, dữ liệu ví dụ và tính toán được trình bày lập trình sau:

Lược đồ hình thang:



Viết chương trình cho lược đồ trên:

Địa chỉ lệnh toán tử

00000 LD 00000

00001 DIV (33) 216

HR 07

**Vùng dữ liệu toán tử:**

Dd: Divident ( BCD) IO, AR, DM, HR, TC, LR, #.

Dr: Divisorword IO, AR, DM, HR, TC, LR, #.

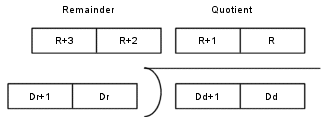
R: word kết quả IO, AR, DM, HR, LR.

### 3.10.5. DIVL(57) - Double BCD Divide



**Mục đích:**

Nội dung của 8 kí số của Dd và Dd+1 được chia bởi nội dung của Dr và Dr+1 và kết quả được đặt vào trong R tới R+3, thương số cất trong R tới R+1, số dư được cất trong R+2 tới R+3.



Hình 3.26. Dd và Dd+1 được chia bởi nội dung của Dr và Dr+1.

**Vùng dữ liệu toán tử:**

|  |  |  |
| --- | --- | --- |
| Dd: | 1st dividend word | IO, AR, DM, HR, TC, LR. |
| Dr: | 1st divisor word | IO, AR, DM, HR, TC, LR. |
| R: | 1st result word | IO, AR, DM, HR, LR. |

### 3.10.6. INC(38) - Increment



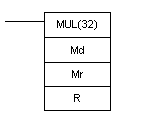
**Nội dung:**

Khi điều kiện thực hiện ngõ vào từ OFF sang ON lệnh này dùng để tăng word Wd lên môt đơn vị, lệnh này không ảnh hưởng đến cờ CY.

**Vùng dữ liệu toán tử:**

Wd: word tăng IO, AR, DM, HR, LR.

### 3.10.7. MUL(32) - BCD Multiply



**Mục đích:**

Khi điều kiện thực hiện ngõ vào là OFF, lệnh MUL không thực hiện.Khi điều kiện thực hiện ngõ vào là ON, lệnh MUL nhân Md với nội dung của Mr và kết quả đặt trong R và R+1.



Hình 3.27. Nhân Md với nội dung của Mr.

Ví dụ:

Xem lược đồ sau: khi IR00000 mà ON thì nội dung của IR 013 và DM0005 được nhân và kết quả được đặt trong HR07 và HR08, dữ liệu ví dụ và tính toán được trình bày dưới chương trình.

Lược đồ hình thang:



Chương trình cho lược đồ hình thang:

Địa chỉ lệnh dữ liệu

0000 LD 00001

0001 MUL (32) 013

DM00005

HR 07

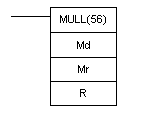
**Vùng dữ liệu toán tử:**

Md: Multiolicand (BCD) IO, AR, DM, HR, TC, LR, #.

Mr: Multiplier (BCD) IO, AR, DM, HR, TC, LR, #.

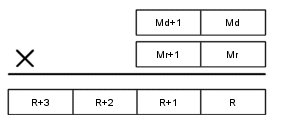
R: word kết quả đầu tiên IO, AR, DM, HR, LR.

### 3.10.8. MULL(56) - Double BCD Multiply



**Mục đích:**

Mull nhân nội dung 8 kí số của Md và Md+1với nội dung của Mr và Mr+1 và kết quả đặt vào trong R đến R+1.

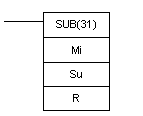


Hình 3.28. Nhân nội dung của Md và Md+1với nội dung của Mr và Mr+1.

**Vùng dữ liệu toán tử:**

|  |  |  |
| --- | --- | --- |
| Md: | 1st multiplicand word | IO, AR, DM, HR, TC, LR |
| Mr: | 1st multiplier word | IO, AR, DM, HR, TC, LR |
| R: | 1st result word | IO, AR, DM, HR, LR |

### 3.10.9. SUB(31) - BCD Subtract



**Mục đích :**

Khi điều kiện thực hiện ngõ vào là OFF, lệnh SUB không thực hiện. Khi điều kiện thực hiện ngõ vào ON, lệnh SUB trừ nội dung của Su, Mi và đặt kết quả vảo trong R. Nếu kết quả là âm thì CY sẽ đặt lên 1và bù 10 kết quả thực đặt trong R. Biến đổi bù 10 để có kết quả đúng, trừ nội dung của R từ zero.

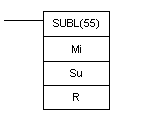
**Vùng dữ liệu toán tử:**

Mi: word bị trừ IO, AR, DM, HR, TC, LR, #.

Su: word trừ IO, AR, DM, HR, TC, LR, #.

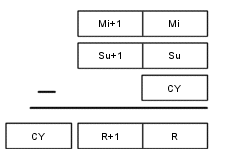
R:word kết quả IO, AR, DM, HR, LR.

### 3.10.10. SUBL(55) - Double BCD Subtract



**Mục đích:**

SUBL trừ CY và nội dung 8 kí số của Su và Su+1 và giá trị 8 kí số trong Mi và Mi+1 và đặt kết quả vào trong R và R+1.Nếu kết quả là âm, CY được đặt lên và bù 10 của kết quả hiện tại được đặt vào trong R.



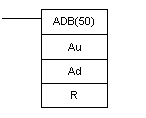
Hình 3.29. Trừ CY và nội dung của Su và Su+1 và giá trị trong Mi và Mi+1.

**Vùng dữ liệu toán tử:**

|  |  |  |
| --- | --- | --- |
| Mi: | 1st minuend word | IO, AR, DM, HR, TC, LR |
| Su: | 1st subtrahend word | IO, AR, DM, HR, TC, LR |
| R: | 1st result word | IO, AR, DM, HR, LR |

## 3.11. Binary Calculation Instructions

### 3.11.1. ADB(50) - Binary Add



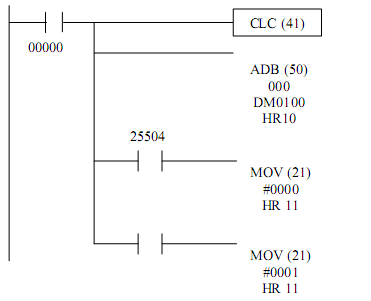
**Mục đích :**

Khi điều kiện thực hiện ngõ vào là OFF thì lệnh ADB không thực hiện. Nếu điều kiện thực hiện ngõ vào là ON thì ADB cộng nội dung của Au, Au và CY và đặt kết quả vào trong R. CY sẽ được đặt lên 1 nếu kết quả lớn hơn FFFF.

*Ví dụ:*

Ví dụ mô tả phép cộng 4 số với CY dùng để thay thế giá trị #0000 hay #0001 trong R+1 để đảm bảo rằng bất kỳ carry được giữ trước .

Sơ đồ hình thang:



Viết chương trình cho sơ đồ hình thang:

Địa chỉ lệnh dữ liệu

00000 LD 00000

00001 OUT TR 0

00002 CLC (41)

00003 ADB (50)

010

DM0100

HR 10

00004 AND NOT

00005 MOV (21)

#0000

HR 11

00006 LD TR 0

00007 AND 25504

00008 MOV (21)

#0001

HR 11

**Vùng dữ liệu toán tử:**

Au: Augend word IO, AR, DM, HR, TC, LR, #.

Ad:Addend word IO, AR, DM, HR, TC, LR, #.

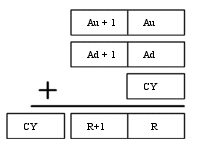
R: word kết quả IO, AR, DM, HR, LR.

### 3.11.2. ADBL(--) - Double Binary Add



**Mục đích :**

ADBL cộng nội dung 8 kí số của Au+1 và Au, nội dung 8 kí số của Ad+1 và Ad, CY và đặt kết quả vào trong R. CY sẽ được đặt lên 1 nếu kết quả lớn hơn FFFF FFFF.



Hình 3.30. Cộng nội dung của Au+1 và Au, nội dung của Ad+1 và Ad, CY.

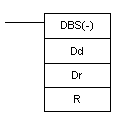
**Vùng dữ liệu toán tử:**

Au: 1st Augend word IO, AR, DM, HR, TC, LR.

Ad: 1stAddend word IO, AR, DM, HR, TC, LR.

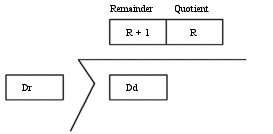
R: word kết quả IO, AR, DM, HR, LR.

### 3.11.3. DBS(--) - Signed Binary Divide



**Mục đích:**

DBS chia nội dung số nhị phân của Dd với nội dung của số nhị phân của Dr và ngõ ra kết quả số nhị phân 8 kí số được đặt vào trong R+1 và R. Thương số được đặt vào trong R và phần dư được đặt vào trong R+1.



Hình 3.31. Chia nội dung số nhị phân của Dd với nội dung của số nhị phân của Dr.

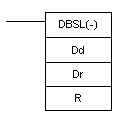
**Vùng dữ liệu toán tử:**

Dd: số bị chia IO, AR, DM, HR, TC, LR, #.

Dr: số chia IO, AR, DM, HR, TC, LR, #.

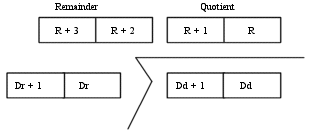
R: word kết quả IO, AR, DM, HR, LR.

### 3.11.4-DBSL(--) - Double Signed Binary Divide



**Mục đích:**

DBSL chia nội dung dữ liệu nhị phân 32 bit (8 kí số) trong Dd+1và Dd với nội dung của dữ liệu nhị phân 32 bit trong Dr+1và Dr và ngõ ra kết quả số nhị phân 16 kí số được đặt vào trong R+3 tới R. Thương số được đặt vào trong R+1 và R , phần dư được đặt vào trong R+3 và R+2.



Hình 3.32. Chia nội dung dữ liệu nhị phân 32 bit trong Dd+1và Dd với nội dung của dữ liệu nhị phân 32 bit trong Dr+1và Dr.

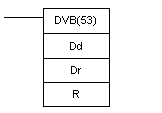
**Vùng dữ liệu toán tử:**

Dd: word bị chia IO, AR, DM, HR, TC, LR.

Dr: word chia IO, AR, DM, HR, TC, LR.

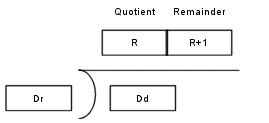
R: word kết quả IO, AR, DM, HR, LR.

### 3.11.5. DVB(53) - Binary Divide



**Mục đích:**

DVB chia nội dung của Dd bởi nội dung của Dr và kết quả được đặt vào trong R và R+1, thương số được đặt trong R, số dư được đặt trong R+1.



Hình 3.33. Chia nội dung của Dd bởi nội dung của Dr.

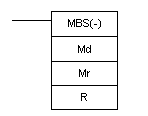
**Vùng dữ liệu toán tử:**

Dd: word bị chia IO, AR, DM, HR, TC, LR, #.

Dr:word chia IO, AR, DM, HR, TC, LR, #.

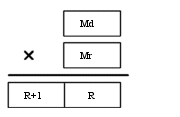
R: word kết quả IO, AR, DM, HR, LR.

### 3.11.6. MBS(--) - Signed Binary Multiply



**Mục đích:**

MBS nhân nội dung số nhị phân của hai word và xuất kết quả số nhị phân 8 kí số vào trong R+1 và R. Bốn kí số bên phải của kết quả được đặt vào trong Rvà bốn kí số bên trái được đặt vào trong R+1.

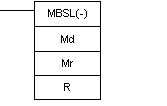


Hình 3.34. Nhân nội dung số nhị phân của hai word.

**Vùng dữ liệu toán tử:**

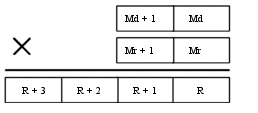
|  |  |  |
| --- | --- | --- |
| Md: | Multiplicand word | IO, AR, DM, HR, TC, LR, #. |
| Mr: | Multiplier word | IO, AR, DM, HR, TC, LR, #. |
| R: | 1st result word | IO, AR, DM, HR, LR. |

### 3.11.7. MBSL(--) - Double Signed Binary Multiply



**Mục đích:**

MBSL nhân nội dung của dữ liệu nhị phân 32 bit (8 kí số) trong Md+1 và Md cùng với dữ liệu nhị phân 32 bit trong Mr+1 và Mr và xuất kết quả nhị phân 16 kí số vào trong R+3 tới R.

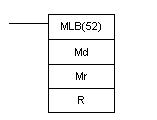


Hình 3.35. Nhân nội dung của dữ liệu nhị phân 32 bit trong Md+1 và Md cùng với dữ liệu nhị phân 32 bit trong Mr+1 và Mr.

**Vùng dữ liệu toán tử:**

|  |  |  |
| --- | --- | --- |
| Md: | 1st multiplicand word | IO, AR, DM, HR, TC, LR. |
| Mr: | 1st multiplier word | IO, AR, DM, HR, TC, LR. |
| R: | 1st result word | IO, AR, DM, HR, LR. |

### 3.11.8. MLB(52) - Binary Multiply



**Mục đích:**

MLB nhân nội dung của Md với nôi dung của Mr, đặt bốn kí số bên phải của kết quả vào trong R và đặt bốn kí số bên trái vào R+1.

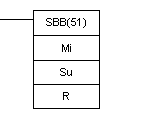


Hình 3.36. Nhân nội dung của Md với nôi dung của Mr.

**Vùng dữ liệu toán tử:**

|  |  |  |
| --- | --- | --- |
| Md: | Multiplicand word | IO, AR, DM, HR, TC, LR, #. |
| Mr: | Multiplier word | IO, AR, DM, HR, TC, LR, #. |
| R: | 1st result word | IO, AR, DM, HR, LR. |

### 3.11.9. SBB(51) - Binary Subtract



**Mục đích :**

Khi điều kiện thực hiện ngõ vào là OFF lệnh SBB không thực hiện. Khi điều kiện thực hiện là ON lệnh SBB trừ nội dung của Su và CY từ Mi và kết quả được đặt vào trong R. Nếu kêt quả là âm thì CY đặt lên và bù 10 kết quả thực đặt trong R. Biến đổi bù 10 để kết quả đúng.

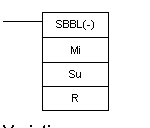
**Vùng dữ liệu toán tử:**

Mi: Minuend word (word bị trừ) IO, AR, DM, HR, TC, LR, #.

Su: Addend word (word trừ) IO, AR, DM, HR, TC, LR, #.

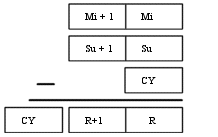
R: result word (word kết quả) IO, AR, DM, HR, LR.

### 3.11.10. SBBL(--) - Double Binary Subtract



**Mục đích:**

SBBL thực hiện trừ CY và giá trị 8 kí số trong Su tới Su+1 và giá trị 8 kí số trong Mi tới Mi+1.Kết quả được đặt vào trong R và R+1.Nếu kết quả là âm, CY được đặt lên và bù hai của kết quả hiện thời được đặt vào trong R và R+1. Sử dụng NEGL để biến đổi bù hai để được kết quả đúng.



Hình 3.37. Trừ CY và giá trị trong Su tới Su+1 và giá trị trong Mi tới Mi+1.

**Vùng dữ liệu toán tử:**

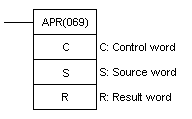
Mi: Minuend word (word bị trừ đầu tiên) IO, AR, DM, HR, TC, LR

Su: Addend word (word trừ đầu tiên) IO, AR, DM, HR, TC, LR

R: result word (word kết quả đầu tiên) IO, AR, DM, HR, LR

## 3.12. Special Maths Instructions

### 3.12.1. APR(--) - Arithmetic Process



**Mục đích:**

Khi trạng thái hoạt động là OFF ,APR không thực hiện. Khi trạng thái hoạt động là ON thì sự hoạt động của APR phụ thuộc vào việc điều khiển word C.

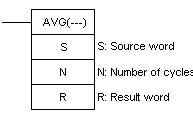
**Vùng dữ liệu toán tử:**

C: dữ liệu điều khiển IO, AR, DM, HR, TC, LR, #.

S: word nguồn IO, AR, DM, HR, TC, LR, #.

D:word đích IO, AR, DM, HR, TC, LR.

### 3.12.2. AVG(--) - Average Value



**Mục đích:**

AVG dùng để tính toán giá trị trung bình của S dến hết chu kì của N.

Mỗi lần AVG được thực hiện, nội dung của S được tích trữ trong các word D+2 tới D+N+1. Trong sự thực hiện đầu tiên, AVG ghi nội dung của S đến D+2; trong lần thực hiện thứ hai đến D+3,V.V….Trong lần thực hiện thứ N, AVG ghi giá trị của S đến D+N+1, AVG tính toán giá trị trung bình của giá trị tích trữ trong D+2 tới D+N+1 và ghi giá trị trung bình đến D.

Chú ý: N phải là BCD từ #0001 đến 0064

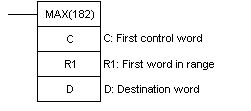
**Vùng dữ liệu toán tử:**

S: word nguồn IO, AR, DM, HR, TC, LR.

N: số chu kỳ IO, AR, DM, HR, TC, LR, #.

D: word đích đầu tiên IO, AR, DM, HR, LR.

### 3.12.3. MAX(--) - Find Maximum



**Mục đích:**

MAX tìm kiếm dãy bộ nhớ từ R đến R+N-1 cho địa chỉ mà nó chứa giá trị lớn nhất và xuất giá trị lớn nhất đến word đích.

Nếu bit 14 của C là ON, MAX nhận ra địa chỉ của word chứa chứa giá trị lớn nhất trong D+1. Địa chỉ được xác định khác nhau cho vùng DM:

⮚ Cho một địa chỉ trong vùng DM, địa chỉ word được ghi tới C+1. Ví dụ như nếu địa chỉ chứa giá trị lớn nhất là DM 0114, sau đó #0114 được ghi vào D+1.

⮚ Cho một địa chỉ trong vùng dữ liệu khác, số địa chỉ từ việc tìm kiếm ban đầu được ghi vào D+1. Ví dụ, nếu địa chỉ chứa giá trị lớn nhất là IR 114 và word đầu tiên trong dãy tìm kiếm là IR 014, sau đó 0100 được ghi vào D+1.

Nếu bit 14 của C là ON và nhiều hơn một địa chỉ chứa cùng một giá trị lớn nhất, vị trí của các địa chỉ thấp nhất sẽ được xuất ra tới D+1.

Số word bên trong dãy được chứa trong 3 kí số bên phải của C, mà phải là BCD ở giữa 001 và 999.

Khi bit 15 của C là OFF, dữ liệu bên trong dãy được xử trí như unsigned binary và khi bit là ON thì dữ liệu được xử trí như signed binary.

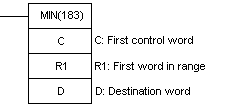
**Vùng dữ liệu toán tử:**

C: dữ liệu điều khiển IO, AR, DM, HR, TC, LR, #.

R1: word đầu tiên trong dãy IO, AR, DM, HR, TC, LR.

D: word đích IO, AR, DM, HR, LR.

### 3.12.4. MIN(--) - Find Minimum

****

**Mục đích:**

MIN tìm kiếm dãy bộ nhớ từ R1 đến R1+N-1 cho địa chỉ mà nó chứa giá trị nhỏ nhất và xuất giá trị nhỏ nhất đến word đích.

Nếu bit 14 của C là ON, MIN nhận ra địa chỉ của word chứa chứa giá trị nhỏ nhất trong D+1.Địa chỉ được xác định khác nhau cho vùng DM:

⮚ Cho một địa chỉ trong vùng DM, địa chỉ word được ghi tới C+1. Ví dụ như nếu địa chỉ chứa giá trị nhỏ nhất là DM 0114, sau đó #0014 được ghi vào D+1.

⮚ Cho một địa chỉ trong vùng dữ liệu khác, số địa chỉ từ việc tìm kiếm ban đầu được ghi vào D+1. Ví dụ, nếu địa chỉ chứa giá trị nhỏ nhất là IR114 và word đầu tiên trong dãy tìm kiếm là IR 014, sau đó #0100 được ghi vào D+1.

Nếu bit 14 của C là ON và nhiều hơn một địa chỉ chứa cùng một giá trị nhỏ nhất, vị trí của các địa chỉ thấp nhất sẽ được xuất ra tới D+1.

Số word bên trong dãy được chứa trong 3 kí số bên phải của C, mà phải là BCD ở giữa 001 và 999.

Khi bit 15 của C là OFF, dữ liệu bên trong dãy được xử trí như unsigned binary và khi bit là ON thì dữ liệu được xử trí như signed binary.

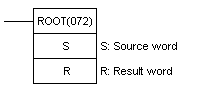
**Vùng dữ liệu toán tử:**

C: dữ liệu điều khiển IO, AR, DM, HR, TC, LR, #.

R1: word đầu tiên trong dãy IO, AR, DM, HR, TC, LR.

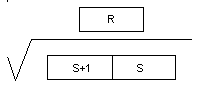
D: word đích IO, AR, DM, HR, LR.

### 3.12.5. ROOT(72) - Square Root



**Mục đích:**

Khi điều kiện thực hiện ngõ vào là OFF, lệnh ROOT không thực hiện. Khi điều kiện thực hiện ngỏ vào là ON lệnh ROOT khai căn bậc hai của Sq và Sq+1 rồi cất kết quả trong R.



Hình 3.38. Khai căn bậc hai của Sq và Sq+1.

*Ví dụ:*

Ví dụ sau mô tả sự khai căn bậc hai của một số 8 số kết quả là một số 4 số. Với số dư sẽ được làm tròn.

Trong ví dụ này:

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Sq+1 | Sq |  | 6 | 3 | 2 | 5 | 0 | 5 | 6 | 1 |

Nên √63250561=7953,0221 làm tròn số kết quả là 7953.

Sơ đồ hình thang:



Chương trình cho sơ đồ hình thang:

Địa chỉ lệnh toán tử

00000 LD 00000

00001 ROOT (72)

MD 0000

001

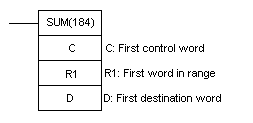
|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **DM0001** | | | | **DM0000** | | | |
| **6** | **3** | **2** | **5** | **0** | **5** | **6** | **1** |

**Vùng dữ liệu toán tử:**

Sq: word nguồn đầu IO, AR, DM, HR, TC, LR.

R: word kết quả IO, AR, DM, HR, LR.

### 3.12.6. SUM(--) - Sum



**Mục đích:**

SUM cộng nội dung của các word R1 đến R+N-1 hoặc các byte trong word R1 đến R1+N/2-1 và xuất giá trị đó đến word nguồn ( D và D+1). Dữ liệu có thể được cộng như số nhị phân hay BCD và sẽ được xuất ra trong dạng tương tự. Dữ liệu nhị phân có thể là signed hay unsigned.

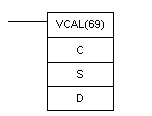
Vùng dữ liệu toán tử:

C: dữ liệu điều khiển IO, AR, DM, HR, LR, #.

R1: word đầu tiên trong dãy IO, AR, DM, HR, TC, LR.

D: word đích đầu tiên IO, AR, DM, HR, LR.

### 3.12.7. VCAL(69) - Value Calculate



**Mục đích:**

Sự hoạt động của VCAL phụ thuộc vào việc điều khiển word C. Nếu C là #0000 hay #0001, VCAL tính toán sin(x) hay cos(x). Nếu C là một địa chỉ VCAL tính toán f(x) của việc đưa ra chức năng trong word C.

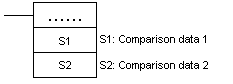
X là nội dung của S.

**Vùng dữ liệu toán tử:**

|  |  |  |
| --- | --- | --- |
| C: | Control word | IO, AR, DM, HR, TC, LR, # |
| S: | Source word | IO, AR, DM, HR, TC, LR |
| D: | Result word | IO, AR, DM, HR, TC, LR |

## 3.13. Input Comparison Instructions

### 3.13.1. Input Comparison Instructions



**Mục đích:**

Khi trạng thái thực hiện là OFF, chỉ thị so sánh ngõ vào không được thực hiện và tiến trình tiếp tục đến phần còn lại của dòng chỉ dẩn. Khi trạng thái thực hiện là ON, chỉ thị so sánh ngõ vào so sánh hằng số and/or nội dung của các word lý thuyết for signed or unsigned data và sẽ tạo ra một trạng thái hoạt động ON khi trạng thái so sánh được tiếp nhận. Nếu trạng thái không được tiếp nhận, phần còn lại của dòng chỉ dẩn sẽ được điều khiển, và sự thực hiện sẽ chuyển đến dòng chỉ dẩn tiếp theo.

Các kiểu so sánh cơ bản :

= bằng

<> không bằng

< bé hơn

<= bé hơn hoặc bằng

> lớn hơn

>= lớn hơn hay bằng

Bảng 3.5. Biểu diễn chức năng code, thuật nhớ (Mnemonic), tên, và chức năng của ngõ vào so sánh.

|  |  |  |  |
| --- | --- | --- | --- |
| Code | Mnemonic | Name | Function |
| 300 | = | Equal | True when S1=S2 |
| 301 | =L | Double equal | True when S1=S2 |
| 302 | =S | Signed equal | True when S1=S2 |
| 303 | =SL | Double signed equal | True when S1=S2 |
| 305 | <> | Not equal | True when S1S2 |
| 306 | <>L | Double not equal | True when S1S2 |
| 307 | <>S | Signed not equal | True when S1S2 |
| 308 | <>SL | Double signed not equal | True when S1S2 |
| 310 | < | Less than | True when S1<S2 |
| 311 | <L | Double less than | True when S1<S2 |
| 312 | <S | Signed less than | True when S1<S2 |
| 313 | <SL | Double signed less than | True when S1<S2 |
| 315 | <= | Less than or equal | True when S1S2 |
| 316 | <=L | Double less than or equal | True when S1S2 |
| 317 | <=S | Signed less than or equal | True when S1S2 |
| 318 | <=SL | Double signed less than or equal | True when S1S2 |
| 320 | > | Greater than | True when S1>S2 |
| 321 | >L | Double greater than | True when S1>S2 |
| 322 | >S | Signed greater than | True when S1>S2 |
| 323 | >SL | Double signed greater than | True when S1>S2 |
| 325 | >= | Greater than or equal | True when S1S2 |
| 326 | >=L | Double greater than or equal | True when S1S2 |
| 327 | >=S | Signed greater than or equal | True when S1S2 |
| 328 | >=SL | Double signed greater than or equal | True when S1S2 |

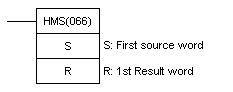
**Vùng dữ liệu toán tử:**

S1: dữ liẹu so sánh 1 IO, AR, DM, HR, TC LR, #\*.

S2: dữ liẹu so sánh 2 IO, AR, DM, HR, TC LR, #\*.

## 3.14. Time Instructions

### 3.14.1. HMS(--) - Seconds-To-Hours



**Mục đích:**

HMS dùng để biến đổi thời gian có kí hiệu là giây thành giờ /phút/ giây tương đương.

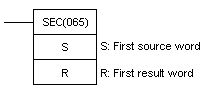
Số giây đã định rõ trong S và S+1 được biến đổi thành giờ/ phút/ giây và đặt vào trong R và R+1.

Đối với kết quả số giây đặt vào trong bit 00 tới bit 07 và số phút đặt vào trong bit 08 tới 15 của R. Giờ sẽ đu7ọc đặt vào trong R+1. Giá trị lớn nhất là 9999 giờ, 59 phút và 59 giây.

**Vùng dữ liệu toán tử:**

|  |  |  |
| --- | --- | --- |
| S: | 1st source word | IO, AR, DM, HR, TC LR. |
| R: | 1st result word | IO, AR, DM, HR, TC LR. |

### 3.14.2. SEC(--) - Hours-To-Seconds



**Mục đích:**

SEC dùng để biến đổi thời gian kí hiêụ với giờ/ phut/ giây thành giá trị tương đương chỉ là giây.

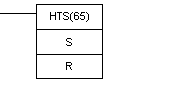
Cho một dữ liệu nguồn, số giây được định rõ trong các bit từ 00 đến 07 và số phút được định rõ trong các bit từ 08 đến 15 của S. Số giờ được định rõ trong S+1. Giá trị lớn nhất là 9999 giờ, 59 phut và 59 giây.

Kết quả được xuất đến R và R+1. Giá trị lờn nhất đạt được là 35 999 999 giây.

**Vùng dữ liệu toán tử:**

|  |  |  |
| --- | --- | --- |
| S: | 1st source word | IO, AR, DM, HR, TC LR. |
| R: | 1st result word | IO, AR, DM, HR, TC LR. |

### 3.14.3. HTS(65) - Hours-To-Seconds



**Mục đích:**

HTS dùng để biến đổi thời gian kí hiêu với giờ/ phut/ giây thành giá trị tương đương chỉ là giây.

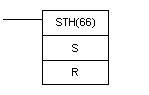
Cho một dữ liệu nguồn, số giây được định rõ trong các bit từ 00 đến 07 và số phút được định rõ trong các bit từ 08 đến 15 của S. Số giờ được định rõ trong S+1. Giá trị lớn nhất là 9999 giờ, 59 phút và 59 giây.

Kết quả được xuất đến R và R+1. Giá trị lờn nhất đạt được là 35 999 999 giây.

**Vùng dữ liệu toán tử:**

|  |  |  |
| --- | --- | --- |
| S: | 1st source word | IO, AR, DM, HR, TC, LR. |
| R: | 1st result word | IO, AR, DM, HR, TC, LR. |

### 3.14.4. STH(66) - Seconds-To-Hours



**Mục đích:**

STH dùng để biến đổi thời gian có kí hiệu là giây thành giờ /phút/ giây tương đương.

Số giây đã định trong S và S+1 đượ biến đỗi thành giờ/phút / giây và đặt vào trong R và R+1.

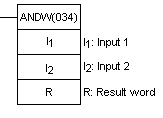
Đối với kết quả , số giây được đặt vào trong các bit từ 00 đến 07 và số phút được đặt vào trong các bit từ 08 tới 15 của R. Số giờ được đặt vào trong R+1. Giá trị lớn nhất sẽ là 9999 giờ, 59 phút và 59 giây.

**Vùng dữ liệu toán tử:**

|  |  |  |
| --- | --- | --- |
| S: | 1st source word | IO, AR, DM, HR, TC, LR. |
| R: | 1st result word | IO, AR, DM, HR, TC, LR. |

## 3.15. Logic Instructions

### 3.15.1. ANDW(34) - Logical AND



**Mục đích:**

Khi điều kiện thực hiện ngõ vào là OFF lệnh ANDW không thực hiện. Khi điều kiện thực hiện ngõ vào là ON, lệnh ANDW thực hiện phép logic AND nội dung của I1 và I2, sau đó lưu kết quả trong R.

*Ví dụ:*

**I1**

**15 00**

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **1** | **0** | **0** | **1** | **1** | **0** | **0** | **1** | **1** | **0** | **0** | **1** | **1** | **0** | **0** | **1** |

**I2**

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **0** | **1** | **0** | **1** | **0** | **1** | **0** | **1** | **0** | **1** | **0** | **1** | **0** | **1** | **0** | **1** |

**R**

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **0** | **0** | **0** | **1** | **0** | **0** | **0** | **1** | **0** | **0** | **0** | **1** | **0** | **0** | **0** | **1** |

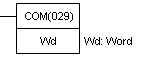
**Vùng dữ liệu toán tử:**

I1: ngõ vào 1 IO, AR, DM, HR, TC, LR, #

I2: ngõ vào 2 IO, AR, DM, HR, TC, LR, #

R: word kết quả IO, AR, DM, HR, LR

### 3.15.2. COM(29) - Complement



**Mục đích:**

Khi điều kiện thực hiện ngõ vào là OFF thì lệnh COM không thực hiện. Khi điều kiện thực hiện ngõ vào ON thì lệnh COM xoá tất cà các bit ON ( mức logic 1) và đặt thành OFF ( mức logic 0) trong Wd, và đặt tất cả các mức logic 0 thành mức logic 1.

*Ví dụ:*

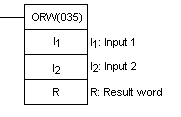
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Trước khi thực hiện lệnh COM(29)** | **1** | **0** | **0** | **1** | **1** | **0** | **0** | **1** | **1** | **0** | **0** | **1** | **1** | **0** | **0** | **1** |

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Sau khi thực hiện lệnh COM(29)** | **0** | **1** | **1** | **0** | **0** | **1** | **1** | **0** | **0** | **1** | **1** | **0** | **0** | **1** | **1** | **0** |

**Vùng dữ liệu toán tử:**

Wd: word bù IO, AR, DM, HR, LR.

### 3.15.3. ORW(35) - Logical OR



**Mục đích:**

Lệnh ORW thực hiện phép logic OR nội dung của I1 và I2, sau đó lưu kết quả trong R.

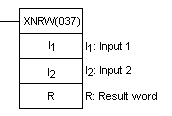
**vùng dữ liệu toán tử:**

I1: ngõ vào 1 IO, AR, DM, HR, TC, LR, #.

I2: ngõ vào 2 IO, AR, DM, HR, TC, LR, #.

R: word kết quả IO, AR, DM, HR, LR.

### 3.15.4. XNRW(37) - Exclusive NOR



**Mục đích:**

Lệnh XNRW sẽ EX- NOR nội dung của I1 và I2, sau đó lưu kết quả trong R.

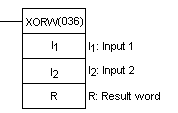
**vùng dữ liệu toán tử:**

I1: ngõ vào 1 IO, AR, DM, HR, TC, LR, #.

I2: ngõ vào 2 IO, AR, DM, HR, TC, LR, #.

R: word kết quả IO, AR, DM, HR, LR.

### 3.15.5. XORW(36) - Exclusive OR



**Mục đích:**

Khi điều kiện thực hiện ngõ vào là OFF thì lệnh XORW không thực hiện. Khi điều kiện thực hiện ngõ vào ON thì lệnh XORW sẽ EX-OR nội dung của I1 và I2, sau đó kết quả lưu vào trong R.

Ví dụ:

**I1**

**15 00**

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **1** | **0** | **0** | **1** | **1** | **0** | **0** | **1** | **1** | **0** | **0** | **1** | **1** | **0** | **0** | **1** |

**I2**

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **0** | **1** | **0** | **1** | **0** | **1** | **0** | **1** | **0** | **1** | **0** | **1** | **0** | **1** | **0** | **1** |

**R**

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **0** | **1** | **0** | **1** | **0** | **1** | **0** | **1** | **0** | **1** | **0** | **1** | **0** | **1** | **0** | **1** |

**Vùng dữ liệu toán tử:**

I1: ngõ vào 1 IO, AR, DM, HR, TC, LR, #.

I2: ngõ vào 2 IO, AR, DM, HR, TC, LR, #.

R: word kết quả IO, AR, DM, HR, LR.

## 3.16. Flag/Register Instructions

### 3.16.1. CLC(41) - Clear Carry



**Mục đích:**

Khi điều kiện thực hiện ngõ vào là OFF lệnh CLC không thực hiện. Khi điều kiện thực hiện ngõ vào là ON CLC bật OFF CY.

**Vùng dữ liệu toán tử:**

N/A

3.16.2. STC(40) - Set Carry



**Mục đích:**

Khi điều kiện thực hiện ngõ vào là OFF lệnh STC không thực hiện. Khi điều kiện thực hiện ngõ vào là ON STC bật ON CY.

**Vùng dữ liệu toán tử:**

N/A.

# CHƯƠNG 4. CÁC CHƯƠNG TRÌNH ỨNG DỤNG CỦA PLC OMROM

## 4.1. Chương Trình Điều Khiển Trò Chơi Dạng “ ĐƯỜNG LÊN ĐỈNH OLYMPIA”

Yêu cầu:

Sau khi người dẩn chương trình đã nêu xong các câu hỏi, các đấu thủ (player) sẽ bấm nút phía trước mặt để trả lời câu hỏi. Ai bấm trước sẽ được trả lời trước. Sau khi bất cứ đấu thủ nào bấm nút, chuông sẽ kêu trong 10 giây. Cùng lúc đó, đèn trước mặt đấu thủ đó sẽ sáng và chỉ được tắt( reset) bởi người dẩn chương trình.

Hình minh họa:

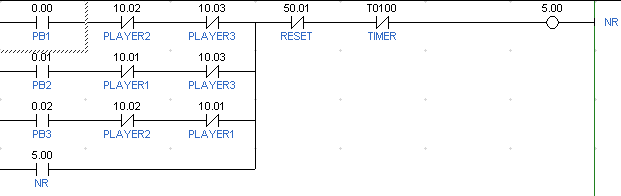


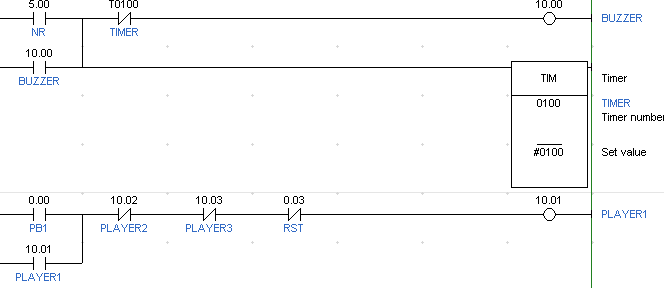
Hình 4.1. Chương trình điều khiển trò chơi dạng “ ĐƯỜNG LÊN ĐỈNH OLYMPIA”.

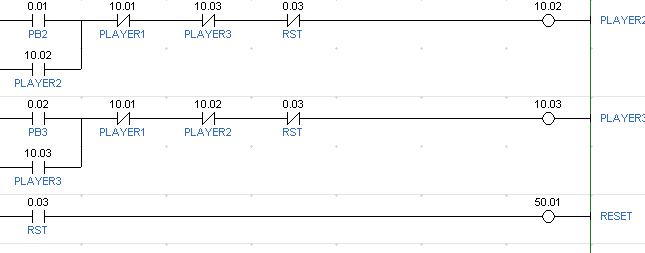
Bảng 4.1. Các ngõ vào ra.

|  |  |
| --- | --- |
| Ngõ vào | Ngõ ra |
| 00000 – nút bấm đấu thủ 1(PB1) | 01000 – còi |
| 00001 – nút bấm đối thủ 2 (PB2) | 01001 – đén của đấu thủ 1 |
| 00002 – nút bấm đấu thủ 3 (PB3) | 01002 – đèn của đấu thủ 3 |
| 00003 – nút tắt (reset) | 01003 – đèn của đấu thủ 3 |

Chương trình



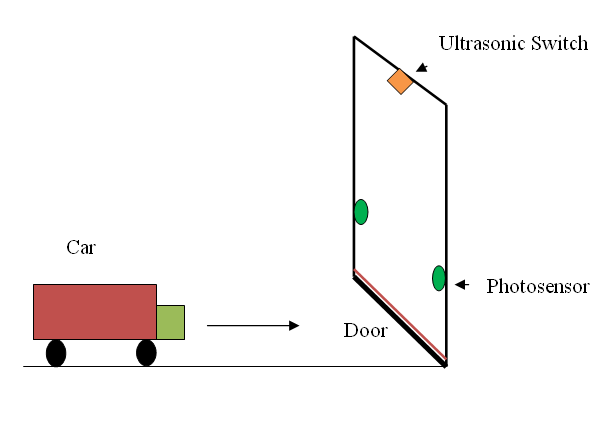




## 4.2. Mạch Tự Động Điều Khiển Của Kho

Đầu vào chuyển mạch siêu thanh (ultrasonic switch) phát hiện sự có mặt của xe. Bộ cảm biến quang điện (photosensor) sẽ phát hiện chiếc xe đang đi qua cửa. Mạch điều khiển đầu ra truyền cho motor của cửa kho mở hay đóng.

Hình minh họa:

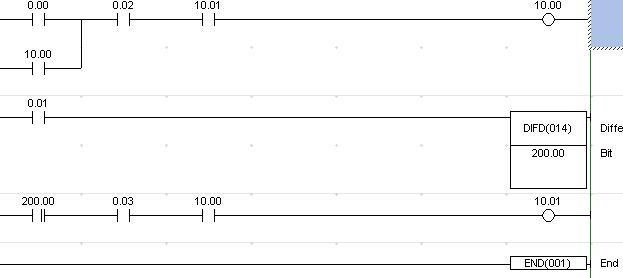


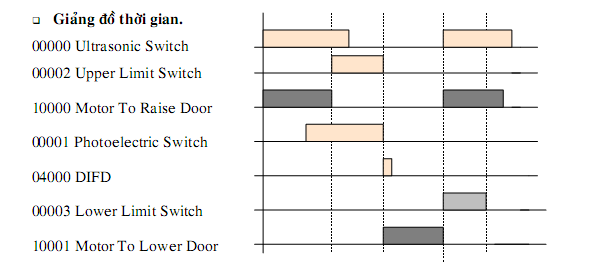
Hình 4.2. Mạch tự động điều khiển của kho.

Bảng 4.2. Phân bố các ngõ vào ra.

|  |  |  |  |
| --- | --- | --- | --- |
| Inputs | Devices | Output | Devices |
| 00000 | Untrosonic Switch | 01000 | Motor To Raise Door |
| 00001 | Photoelectric Switch | 01001 | Motor To Lower Door |
| 00002 | Door upper limit switch |  |  |
| 00003 | Door Lower limit switch |  |  |

Giảng đồ hình thang:

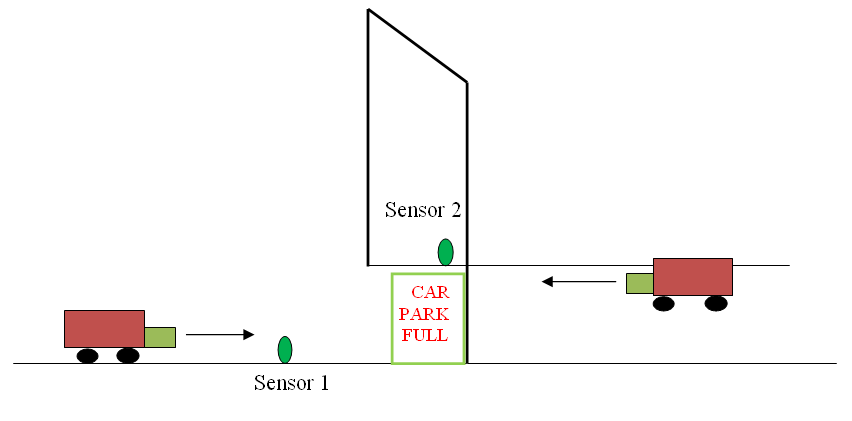




## 4.3. Mạch Điều Khiển Bãi Đậu Xe

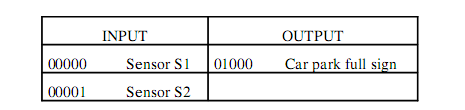
Đây là một chương trình điều khiển bãi đỗ xe đơn giản chỉ cho phép tối đa là 100 chiếc xe được đậu tại một thời điểm. Mỗi khi có xe mới đi vào, Sensor (S1) sẽ phát hiện và PLC sẽ cộng 1 vào tổng số xe hiện đang có trong bãi và sẽ trừ đi 1 khi Sensor (S2) phát hiện có` xe đi ra khỏi bãi đậu xe. Khi có đủ 100 xe trong bãi thì đèn hiệu “ CAR PARK FULL “ sẽ sáng để báo hiệu các xe khác không được vào bãi.

Hình minh họa:

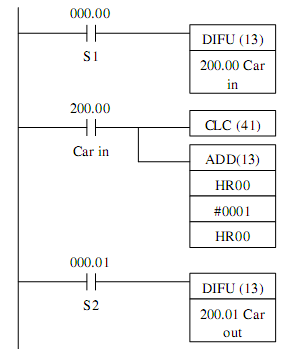


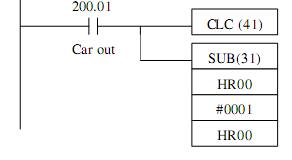
Hình 4.3. Mạch điều khiển bãi đậu xe.

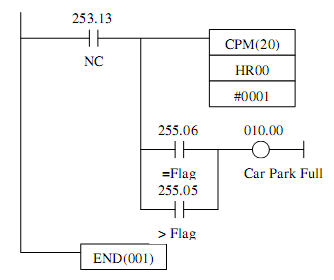
Bảng 4.3. Phân bố thiết bị vào ra.



Sơ đồ hình thang:







## 4.4. Hoạt Động Điều Dẩn Nước Ra Hay Đổ Nước Vào

Quy trình hoạt động :

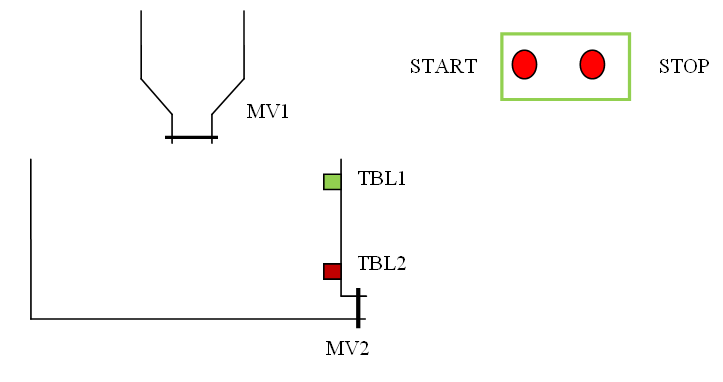
⮚ Khi nút bắt đầu PB1 (start putton) được ấn, van cung cấp nước MV1 (water supply vale) mở và nước bắt đầu được đổ vào thùng. Trong thời gian này motor M bắt đầu họat động.

⮚ Khi mức nước qua TBM2( lower level switch) và tiến đến TBL1 (upper level switch) ,MV1 đóng và motor M dừng lại.

⮚ Sau đó MV2 (Drain vale) mở và bắt đầu dẫn nước ra ngoài, khi mức nước xuống dưới TBL2, MV2 đóng lại.

⮚ Khi chu kì hoạt động được bốn lần, quá trình hoạt động kết thúc.

Hình minh họa:

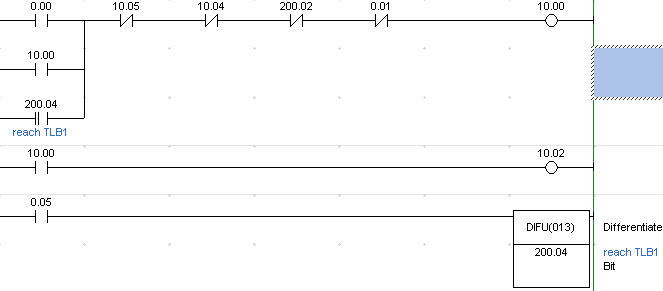


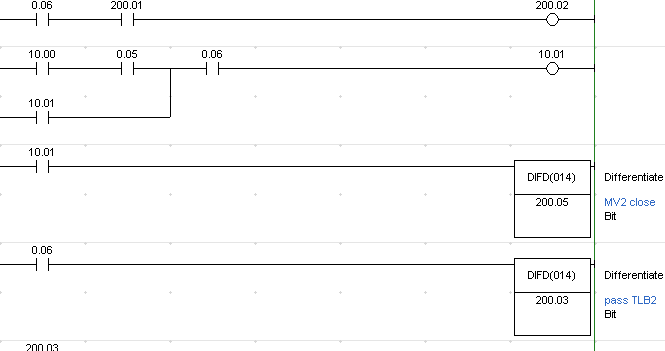
Hình 4.4. Hoạt động điều dẩn nước ra hay đổ nước vào.

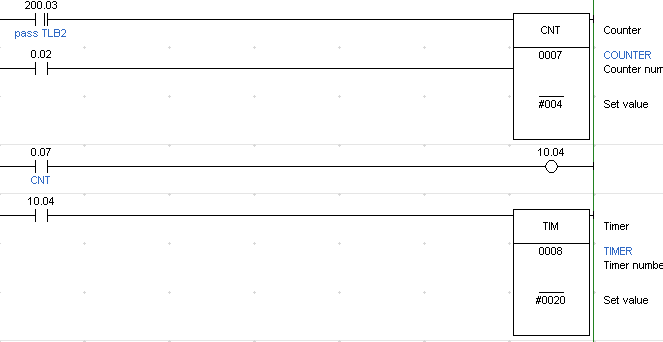
Bảng 4.4. Phân bố thiết bị vào ra.

|  |  |  |  |
| --- | --- | --- | --- |
| Inputs | Devices | Output | Devices |
| 00000 | Start button (PB1) | 01000 | Water supply vale (MV1) |
| 00001 | Stop button (PB2) | 01001 | Drain vale (MV2) |
| 00002 | Reset button (PB3) | 01002 | Stirring motor (M) |
| 00005 | Upper level switch (TLB1) | 01004 | End indicator |
| 00006 | Lower level switch (TLB2) | 01005 | Buzzer |

Sơ đồ hình thang:







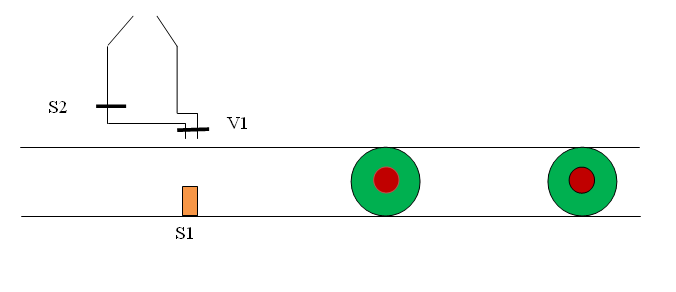


## 4.5. Hệ Thống Tự Động Bôi Trơn Dầu Cho Bánh Xe

Qui trình hoạt động:

Khi bánh xe di chuyển về phía cảm biến S1, S1 sẽ phát hiện bánh xe và sẽ ra tín hiệu cho valve điện từ (V1) để cấp dầu bôi trơn cho bánh xe. Valve V1 sẽ mở trong khoảng thời gian ngắn để cấp một lượng dầu định trước cho bánh xe. Khi cảm biến S2 phát hiện mức dầu trong bồn chứa ( Tank) thấp nó sẽ ra tín hiệu cảnh báo.

Hình minh họa:

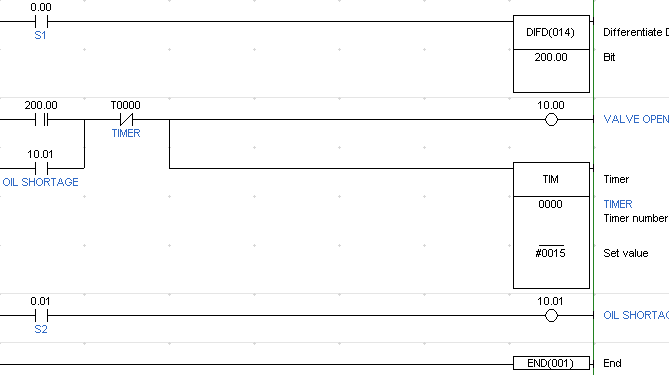


Hình 4.5. Hệ thống tự động bôi trơn dầu cho bánh xe.

Bảng 4.5.Các thiết bị vào ra.

|  |  |
| --- | --- |
| Input | Output |
| 00000 Position detection (S1) | 01000 Electromagnetic valve for oil |
| 00001 lower limit of lever (S2) | 01001 Oil shortage arlam indicator |

Sơ đồ hình thang:

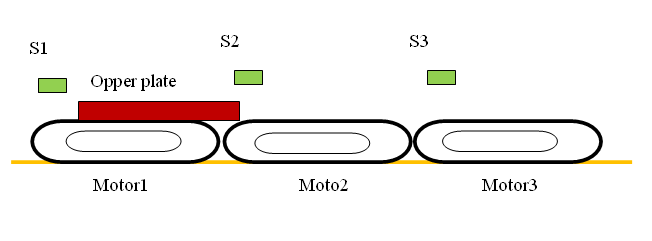


## 4.6. Mạch Điều Khiển Động Cơ Băng Tải

Mô tả quy trình hoạt động :

Băng tải gồm có ba phân đoạn và cần điều khiển sao cho động cơ của mỗi phân đoạn chỉ chạy khi có đối tượng ( tấm đồng – opper plate ) đang nằm trên phân đoạn tương ứng vị trí của tám kim loại được xác định bởi các cảm biến tiệm cận đặt gần nó ( sensor 1,2,3). Khi tấm kim loại nằm trong tầm phát hiện của một sensor , động cơ tương ứng sẽ vẫn làm việc. Khi tấm kim loại nằm ngoài tầm phát hiện của sensor, một Timer trể sẽ được kích hoạt và khi thời gian đặt của timer hết động cơ tương ứng sẽ ngừng.

Hình minh họa:

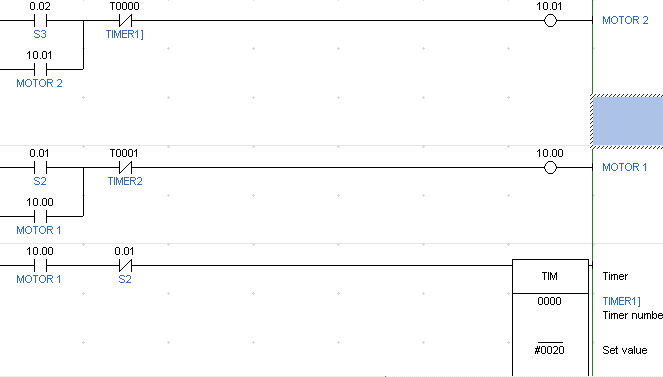


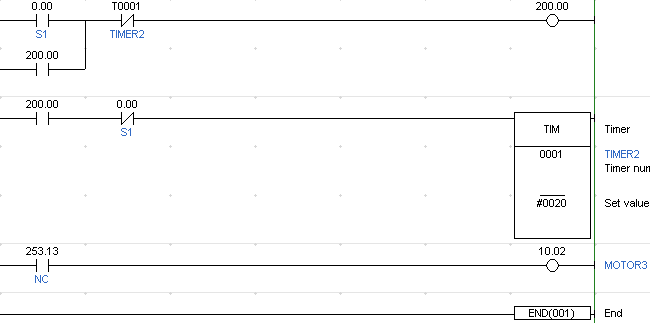
Hình 4.6. Hệ thống điều khiển động cơ băng tải.

Bảng 4.6. Các thiết bị vào ra.

|  |  |  |  |
| --- | --- | --- | --- |
| Input | Thiết bị ngoài | Output | Thiết bị ngoài |
| 00000 | Sensor 1 | 01000 | Motor 1 |
| 00001 | Sensor 2 | 01001 | Motor 2 |
| 00002 | Sensor 3 | 01002 | Motor 3 |

Sơ đồ hình thang:



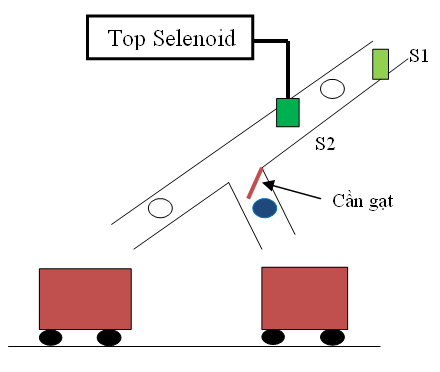


## 4.7. Điều Khiển Cơ Cấu Máy Lựa Sản Phẩm

Mô tả: trong ứng dụng này, hệ thống sẽ lựa sản phẩm thiếu chất lượng trộn lẩn sản phẩm đúng tiêu chuẩn. Ví dụ phân biệt bóng đen và bóng trắng và dựa vào hai contennơ khác nhau.

Khi bắt đầu hoạt động, sensor (S1) sẽ cảm nhận sự hiện diện của bóng. Cuộn selenoit ở trên (top solenoid) thả ra cho sensor (S2) cảm nhận màu trước khi thả vào contennơ. Nếu bóng đen, cần gạt (1002) sẽ được kích hoạt và chuyển hướng bóng đen xuống contennơ khác.

Hình minh họa:

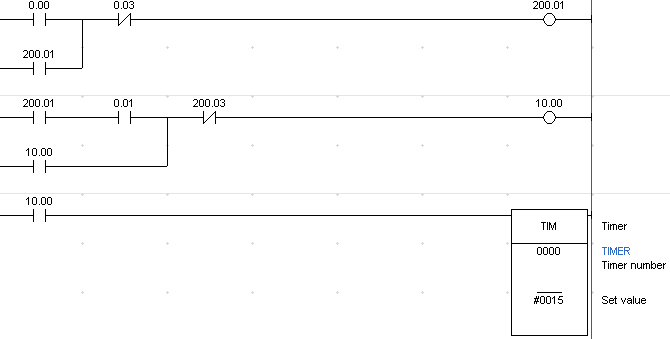


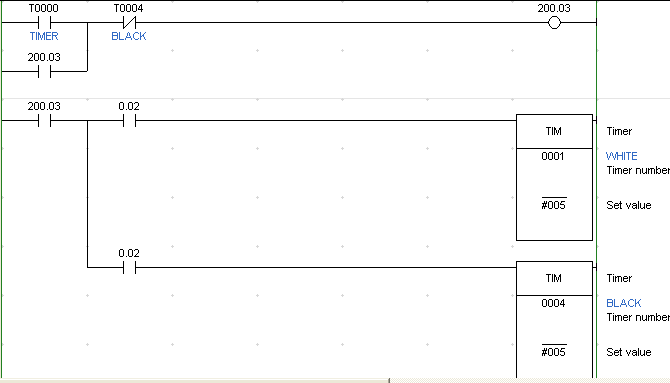
Hình 4.7. Hệ thống điều khiển cơ cấu máy lựa sản phẩm.

Bảng 4.7. Phân bố thiết bị vào ra.

|  |  |  |  |
| --- | --- | --- | --- |
| Inputs | Devices | Output | Devices |
| 00000 | Start PB | 01000 | Top Cylinder |
| 00001 | Ball PB | 01001 | Button Cylinder |
| 00002 | Color sensor (S1) | 01002 | Pusher |
| 00003 | Stop PB | 01002 |  |

Sơ đồ hình thang:









# KẾT LUẬN

Tài liệu này giúp em hiểu sâu hơn về ý nghĩa của các lệnh, thành thạo hơn trong cách sử sụng phần mềm Cx- programmer. Không chỉ vậy, nó còn giúp em biết thêm về cách lập trình điều khiển hệ thống dây chuyền tự động. Tài liệu này còn có thể giúp được các bạn khóa sau cập nhật các lệnh một cách nhanh chống, biết cách sử dụng phần mềm một cách dễ dàng thông qua các hình chỉ dẩn trong bài bao cáo. Từ đó, các bạn có thể phát triển để lập trình để điều khiển các hệ thống dây truyền tự động phức tạp.

# Tài Liệu Tham Khảo

## [1] Phần mềm CX-Programmer.

## [2] Giáo trình PLC.

## [3] Website:

http://cdtvn.net/index.php?option=com\_content&view=article&id=607:cp1l-micro-plc&catid=51:mechatronics-hardware&Itemid=267

http://www.ducvietjsc.com.vn/website/product.aspx?product\_id=181&TypeProduct\_id=47