บทที่ 6

ซอฟท์แวร์ควบคุมระบบ

จากบทที่ 4 ได้กล่าวถึงแนวทางการออกแบบทางซอฟท์แวร์ควบคุมระบบ ซึ่งได้กล่าวถึงข้อดีข้อเสียของแต่ละวิธี และได้เลือกวิธีที่เหมาะสมที่สุด คือ วิธีเขียนซอฟท์แวร์ด้วยแผนภาพขั้นบันได เพราะแผนภาพขั้นบันไดมีความคล้ายคลึงกับวงจรรีเลย์ ซึ่งช่างลิฟท์มีความชำนาญอยู่แล้ว ทำให้ช่างลิฟท์สามารถช่วยผู้พัฒนาระบบได้ นอกจากนี้ยังง่ายต่อผู้พัฒนารุ่นต่อๆ ไป ที่จะทำความเข้าใจกับซอฟท์แวร์ได้ง่าย การเขียนซอฟท์แวร์ด้วยแผนภาพขั้นบันไดนี้ คล้ายคลึงกับเครื่องควบคุมแบบพีแอลซี แต่พีแอลซีที่มีจำนวนอินพุทเอาท์พุทมากเพียงพอจะมีราคาแพงมากไม่เหมาะที่จะนำมาใช้ควบคุมลิฟท์ จึงนำไมโครคอมพิวเตอร์มาใช้ในเครื่องควบคุม

6.1 พีแอลซี

Programmable Logic Controller หรือเรียกย่อๆ ว่า พีแอลซี (PLC) นี้เป็นอุปกรณ์ควบคุมซีเควันซ์ชนิดอีเล็กทรอนิกส์ที่สามารถโปรแกรมได้ ภายในมีไมโครคอมพิวเตอร์เป็นมันสมองสั่งการที่สำคัญ

พีแอลซีจะมีส่วนที่เป็นอินพุทและเอาท์พุทจะใช้ต่อออกไปควบคุมการทำงานของอุปกรณ์หรือเครื่องจักรที่เป็นเป้าหมาย เราสามารถสร้างวงจรหรือแบบของการควบคุมได้โดยการป้อนเป็นโปรแกรมสั่งงานเข้าไปในพีแอลซี โปรแกรมนี้จะทำหน้าที่เหมือนวงจรของรีเลย์ ตัวตั้งเวลาและตัวนับ ที่เคยใช้ตามปกติ เมื่อกดปุ่มบังคับให้พีแอลซีเริ่มทำงานมันจะทำงานได้เหมือนกับวงจรรีเลย์ที่เราป้อนโปรแกรมเข้าไปได้อย่างน่าอัศจรรย์ พีแอลซีจะสร้างอุปกรณ์ควบคุมต่างๆ ภายในเอง เช่น รีเลย์ ตัวตั้งเวลา ตัวนับ ได้โดยซอฟท์แวร์ สิ่งเหล่านี้ไม่มีตัวตนในรูปของวัตถุ แต่จะปรากฏในรูปของฟังก์ชั่นการทำงานที่ตรงกับของจริง นอกจากนั้นการต่อสายเชื่อมโยงอุปกรณ์เหล่านี้เข้าหากันเป็นวงจรก็ทำโดยซอฟท์แวร์ทั้งสิ้น เราสามารถแก้วงจรหรือเพิ่มเติมวงจรได้เพียงแต่แก้ไขโปรแกรมวงจรเท่านั้น การป้อนโปรแกรมสามารถทำได้โดยใช้แป้นป้อนโปรแกรมเล็กๆ หรือ

เครื่องพิเศษที่มีจอภาพและแป้นพิมพ์ การป้อนโปรแกรมสามารถทำได้ง่ายและรวดเร็ว แน่นอนรวดเร็วและสะดวกกว่าการเดินสายไฟใหม่ตามแบบเดิมมาก

6.2 แผนภาพขั้นบันได (Ladder Diagram)

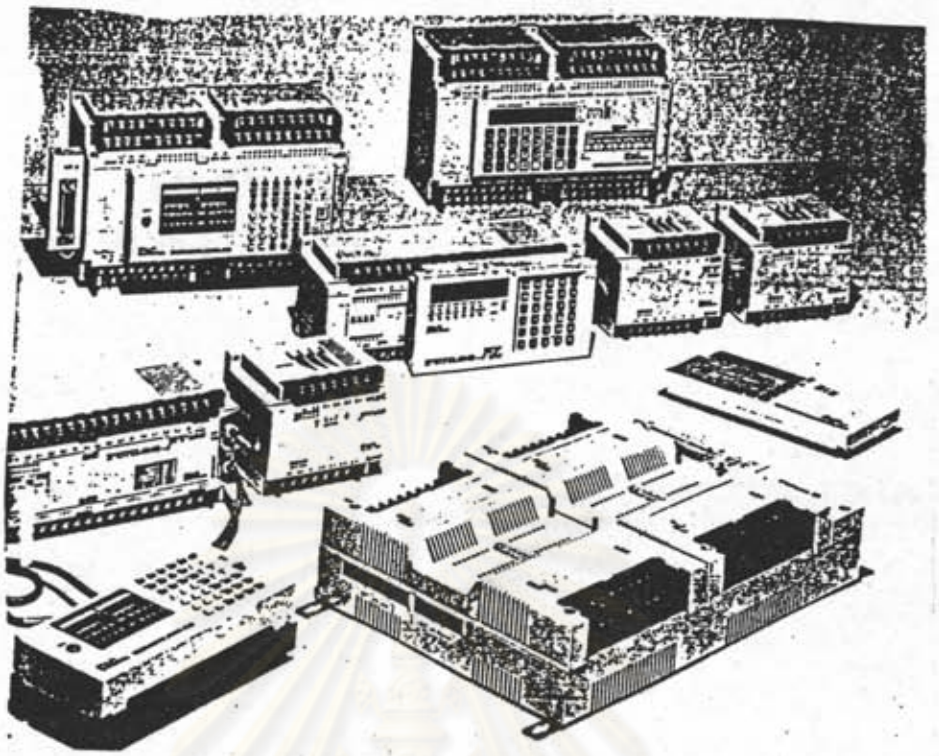
แผนภาพขั้นบันไดเป็นแผนภาพที่ยังยึดหลักของแผนภาพรีเลย์ (Relay Diagram) เพื่อให้ช่างที่ปฏิบัติงานด้านควบคุมแบบลำดับ (Sequence Control) ไม่ต้องเปลี่ยนแนวความคิดความเข้าใจกับแผนภาพของระบบ เพียงแต่เรียนรู้วิธีการโปรแกรมเพิ่มขึ้นเท่านั้น

การเขียนโปรแกรมแบบแผนภาพขั้นบันไดนี้ มีลักษณะคล้ายภาษาโครงสร้าง คือ มีการทดสอบเงื่อนไขในการทำงาน ถ้าเงื่อนไขเป็นจริงก็จะให้ผลการทำงานออกมา เงื่อนไขที่ทดสอบสามารถแทนได้ด้วยการต่อวงจรของหน้าสัมผัสและผลการทำงานสามารถแทนได้ด้วยคอยล์ของรีเลย์ การทดสอบเงื่อนไขจะทำเป็นลำดับจากซ้ายไปขวาจนครบเงื่อนไขที่ทดสอบแล้วนำผลทดสอบไปแสดงออกที่รีเลย์ จากนั้นจึงจะเคลื่อนลงมาทดสอบเงื่อนไขลำดับต่อมาของแผนภาพขั้นบันได การทำงานจึงเป็นลำดับจากซ้ายไปขวาและจากบนลงล่าง เมื่อการทดสอบเงื่อนไขเสร็จสิ้นลงก็จะนำผลการทดสอบไปควบคุมการทำงานของลิฟท์

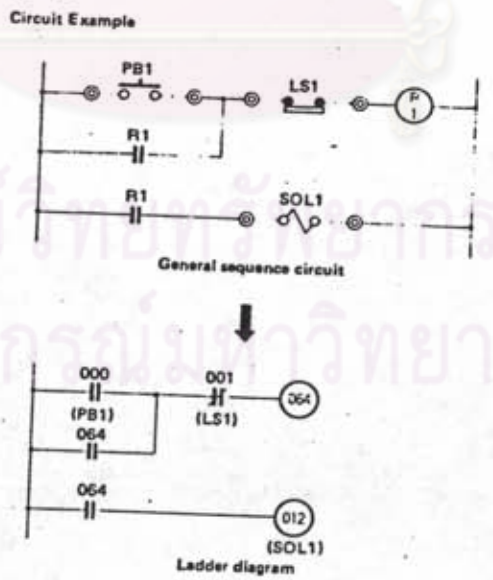
การสร้างโปรแกรมแบบแผนภาพขั้นบันไดนั้น เมื่อเราวาดแผนภาพที่จะใช้ในการควบคุมแล้ว เราจะแปลงแผนภาพนั้นเป็นคำสั่งนีโมนิค (MNEMONIC) จากนั้นจึงป้อนเข้าไมโครคอมพิวเตอร์ ซึ่งไมโครคอมพิวเตอร์จะทำการแปลงคำสั่งนีโมนิคเป็นรหัสคำสั่งต่อไป

6.3 โปรแกรมควบคุมลิฟท์

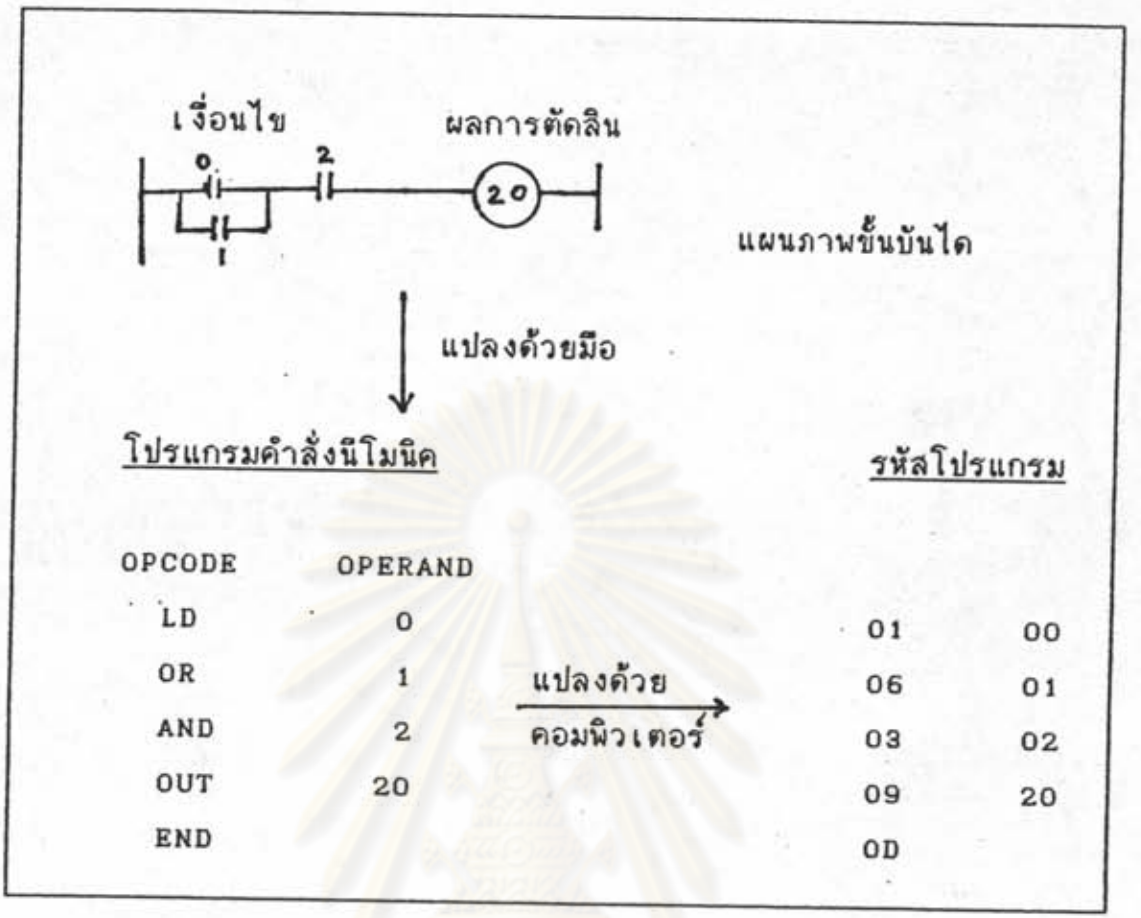
การเขียนโปรแกรมควบคุมลิฟท์จะแบ่งโปรแกรมออกเป็นโมดูล (Module) ตามวิธีการ Modular Programing ทั้งนี้เพื่อให้ง่ายต่อการแก้ไขโปรแกรม เพราะสามารถเพิ่มหรือตัดโปรแกรมได้เป็นโมดูล



รูปที่ 6.1 รูปตัวอย่างพีแอลซี



รูปที่ 6.2 วงจรรีเลย์กับแผนภาพขั้นบันได



รูปที่ 6.3 การแปลงแผนภาพขั้นบันไดเป็นรหัสโปรแกรม

6.3.1 การทำงานของโปรแกรมควบคุม

การทำงานของโปรแกรมควบคุมสามารถแบ่งได้เป็น 2 ส่วน คือ การทำงานในโปรแกรมหลัก และการทำงานเมื่อเกิดการอินเตอร์รัพท์

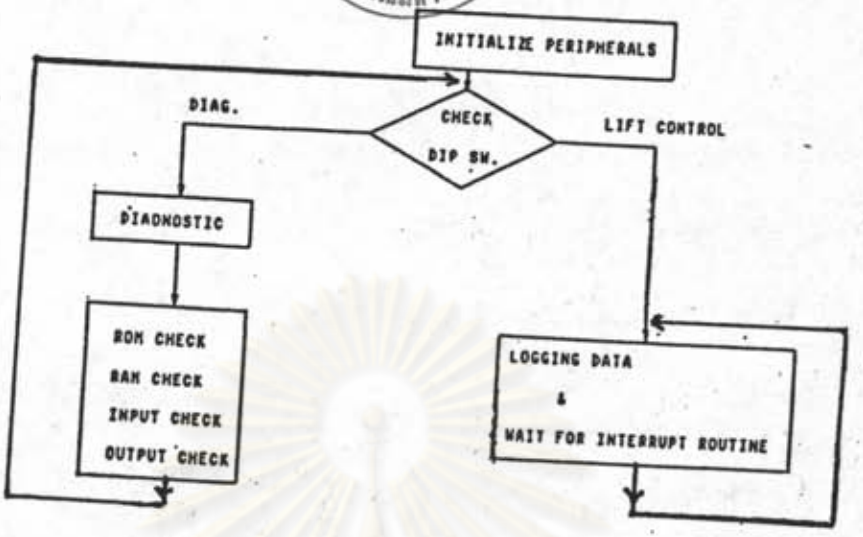
1) การทำงานในโปรแกรมหลัก เป็นโปรแกรมควบคุมการเริ่มต้นการทำงานของลิฟท์ การตรวจสอบความพร้อมของเครื่องควบคุม และการพิมพ์ผลสถานะการทำงานของลิฟท์ การทำงานในโปรแกรมหลักแสดงในรูปที่ 6.4 เมื่อทำการ INITIALIZE PERIPHERAL แล้ว ซีพียูจะมาตรวจที่ดิพสวิทช์ว่าต่อไปเป็นการตรวจสอบระบบ (DIAGNOSTIC) หรือไม่ ถ้าไม่ก็จะพิมพ์ผลออกทางเครื่องพิมพ์แล้วรอการอินเตอร์รัพท์จาก CTC

2) การทำงานในโปรแกรมอินเตอร์รัพท์ การทำงานของโปรแกรมในส่วนนี้จะควบคุมการทำงานของลิฟท์เป็นส่วนใหญ่ โดยจะทำการทดสอบเงื่อนไขการทำงานตามรหัสแผนภาพขั้นบันไดที่เราได้โปรแกรมไว้ แล้วนำผลที่ได้ไปควบคุมการทำงานของลิฟท์ การทำงานในโปรแกรมอินเตอร์รัพท์แสดงในรูปที่ 6.5 เมื่ออ่านอินพุทจากตัวตรวจวัด (SENSOR) ในปล่องลิฟท์และปุ่มกดต่างๆ เข้ามาทำการจัดเก็บไว้แล้ว LIFT SUPPORT FUNCTION จะทำการวิเคราะห์แล้วส่งผลไปให้ LADDER INTERPRETER ทำการประมวลผลต่อไป LADDER INTERPRETER จะอ่านรหัสคำสั่งแผนภาพขั้นบันไดจาก LADDER PROGRAM มาทีละบรรทัด แล้วทำการแปลความหมายและทำงานตามคำสั่งนั้นทันที การทำงานตามแต่ละคำสั่งจะต้องมีการเรียกใช้ FUNCTION SERVICE ROUTINE ซึ่งประกอบด้วยโปรแกรมเล็กๆ แยกตามคำสั่งหลายๆ โปรแกรม การทำงานของ LADDER INTERPRETER จะสิ้นสุดลงเมื่อพบรหัส END ซึ่งอยู่ตอนท้ายของ LADDER PROGRAM จากนั้นจะนำผลการตัดสินเงื่อนไขต่างๆ ซึ่งเก็บไว้ในหน่วยความจำ ส่งเป็นเอาท์พุทออกไปควบคุมลิฟท์ทันที จบจากการทำงานทั้งหมดนี้แล้วจะออกจากโปรแกรมอินเตอร์รัพท์กลับไปสู่โปรแกรมหลักต่อไป โปรแกรมอินเตอร์รัพท์นี้จะทำงานทุกๆ 10 mSec จึงทำให้ดูเหมือนว่าวงจรแผนภาพขั้นบันไดทำงานอยู่ตลอดเวลา ลิฟท์จึงถูกควบคุมด้วยวิธีเช่นนี้

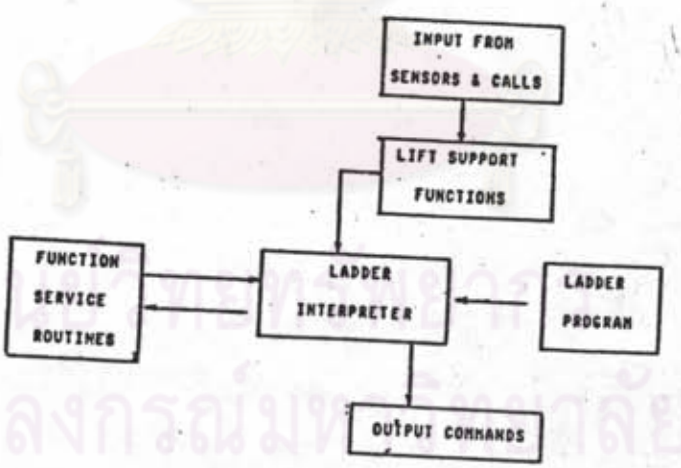
6.3.2 องค์ประกอบของโปรแกรม

ในโปรแกรมหลักมีโปรแกรมที่สำคัญๆ ดังนี้

1) INITIALIZE PERIPHERALS เป็นโปรแกรมสำหรับ INITIALIZE เครื่องควบคุมให้สามารถทำงานได้



รูปที่ 6.4 ผังงานของโปรแกรมหลัก



รูปที่ 6.5 ผังงานของโปรแกรมอินเตอร์รัพท์

2) DIAGNOSTIC PROGRAM — เป็นโปรแกรมสำหรับตรวจสอบความผิดปกติของเครื่องควบคุม แบ่งการตรวจสอบเป็น ROMCHECK, RAMCHECK, INPUTCHECK, OUTPUTCHECK

3) DATA LOGGING — เป็นโปรแกรมสำหรับพิมพ์ข้อมูลที่เกี่ยวกับการทำงานของลิฟท์ ข้อมูลนี้สามารถนำมาใช้วิเคราะห์การใช้ลิฟท์ สมรรถนะของลิฟท์และการซ่อมบำรุงได้

ในโปรแกรมอินเตอร์รัพท์มีโปรแกรมที่สำคัญๆ ดังนี้

1) INPUT FROM SENSORS & CALLS — เป็นโปรแกรมจัดการเกี่ยวกับการอ่านสัญญาณอินพุท แล้วทำการจัดรูปแบบไว้ในหน่วยความจำ เพื่อให้ง่ายต่อการนำไปใช้งาน

2) LIFT SUPPORT FUNCTIONS — เป็นฟังก์ชั่นพิเศษที่เขียนขึ้นใช้กับลิฟท์โดยเฉพาะ ฟังก์ชั่นเหล่านี้ถ้าทำด้วยแผนภาพขั้นบันไดจะมีความยุ่งยากและซับซ้อนมาก จึงต้องใช้ภาษาแอสแซมบลี้เขียนฟังก์ชั่นเหล่านี้ในรูปของ SUB-ROUTINE ซึ่งสามารถเรียกใช้ หรือตัดออกจากระบบได้ง่าย

3) LADDER PROGRAM — เป็นโปรแกรมแผนภาพขั้นบันไดที่เก็บในรูปของรหัส

4) LADDER INTERPRETER — เป็นโปรแกรมสำหรับแปลความหมายของรหัสแผนภาพขั้นบันได และทำงานตามนั้น

5) FUNCTION SERVICE ROUTINE — เป็นกลุ่มของโปรแกรมย่อย ซึ่ง LADDER INTERPRETER นำไปใช้ในการแปลความหมายของรหัสแผนภาพขั้นบันได

6) OUTPUT COMMAND เป็นโปรแกรมที่นำผลที่ประมวลได้ ซึ่งเก็บไว้ในหน่วยความจำออกไปควบคุมอุปกรณ์ขับเคลื่อน

6.3.3 SYSTEM MEMORY MAPPING

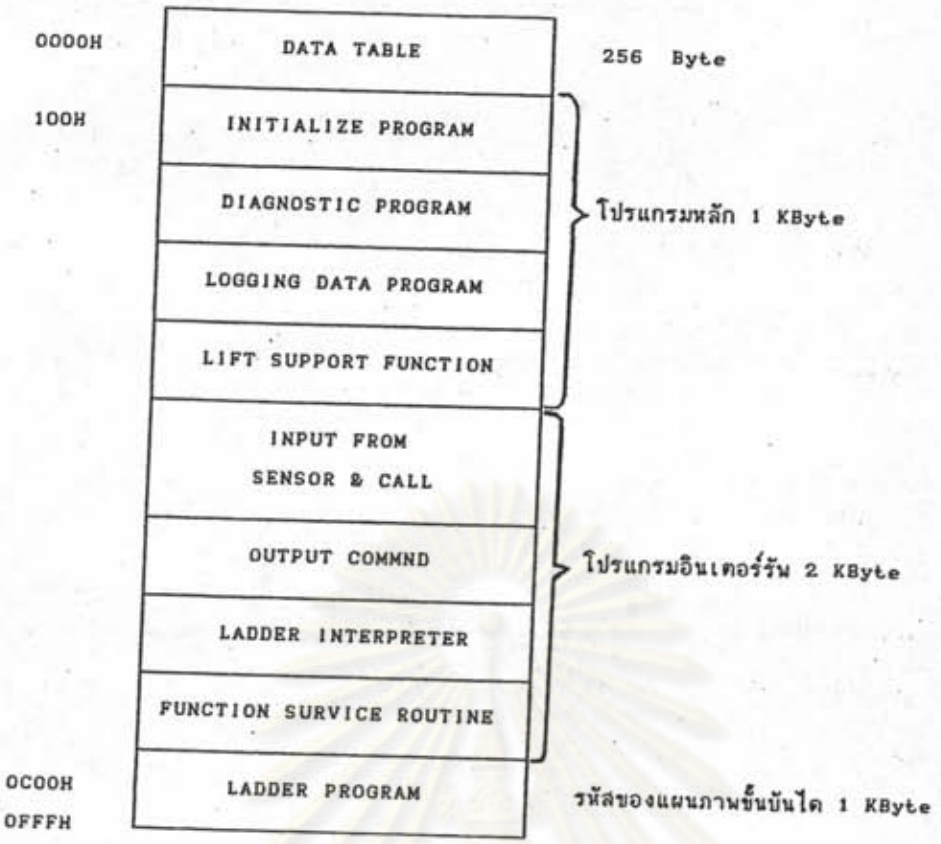
1) PROGRAM MEMORY MAPPING คือ การจัดหน่วยความจำที่ใช้เก็บโปรแกรมทั้งหมดที่ใช้ในการควบคุมการทำงานของลิฟท์ ในส่วนนี้จะเก็บในหน่วยความจำประเภท ROM การจัดพื้นที่สำหรับโปรแกรมแสดงในรูปที่ 6.6

2) DATA MEMORY MAPPING คือ การจัดหน่วยความจำที่ใช้เก็บสถานะการทำงานของระบบ เช่น ปุ่มกด ตำแหน่งลิฟท์ อินพุท เอาท์พุท เนื่องจากสถานะการทำงานของระบบมีการเปลี่ยนแปลงตลอดเวลา ดังนั้นจึงใช้หน่วยความจำประเภท RAM การจัดพื้นที่สำหรับข้อมูลสถานะแสดงในรูปที่ 6.7 เนื่องจากการอ่านหรือเขียนสัญญาณของอินพุทและเอาท์พุทนั้นกระทำพร้อมกัน 8 สัญญาณ ดังนั้นรูปแบบของ INPUT/OUTPUT BUFFER จึงเป็นดังรูปที่ 6.8 โดยใน 1 ไบท์ ซึ่งมีขนาด 8 บิท จะบรรจุสัญญาณได้ 8 สัญญาณ การบรรจุสัญญาณในลักษณะนี้เราจะเรียกว่า PACK DATA เนื่องจากเป็นการไม่สะดวกที่จะนำไปใช้งานในโปรแกรม ดังนั้นเราจึงทำการกระจายในแต่ละบิทให้ไปอยู่ในบิทสุดท้ายในพื้นที่ของ STATUS BUFFER ดังรูปที่ 6.9 การกระจายสัญญาณในลักษณะนี้จะเรียกว่า UNPACK DATA ซึ่งสามารถนำไปใช้ในโปรแกรมได้สะดวก สำหรับการส่งผลการประมวลเป็นเอาท์พุทในการควบคุมก็จะใช้วิธีการเดียวกัน คือ ข้อมูลใน STATUS BUFFER จะเป็นรูปของ UNPACK DATA ก่อนส่งออกไปควบคุมต้องทำการ PACK DATA ก่อน

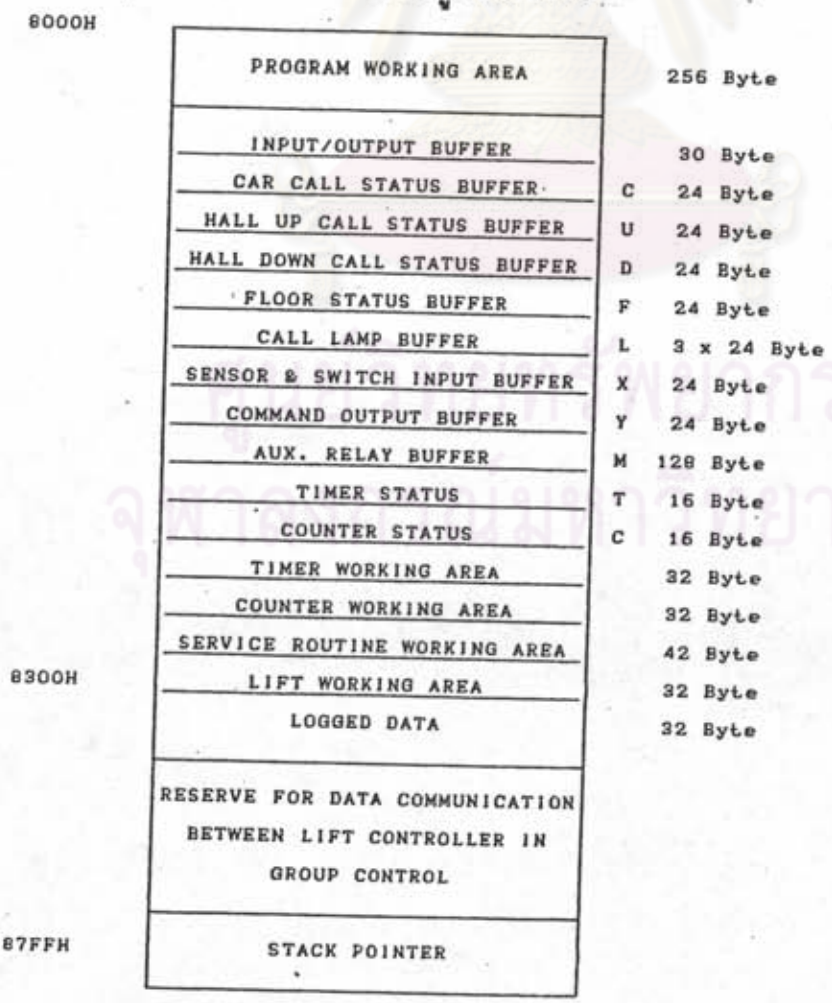
6.4 รายละเอียดของโปรแกรม

จากที่ได้กล่าวแล้วว่าโปรแกรมควบคุมสามารถแบ่งส่วนใหญ่ๆ ได้ 2 ส่วนคือ โปรแกรมหลัก และโปรแกรมอินเตอร์รัพท์ ซึ่งมีรายละเอียดดังนี้

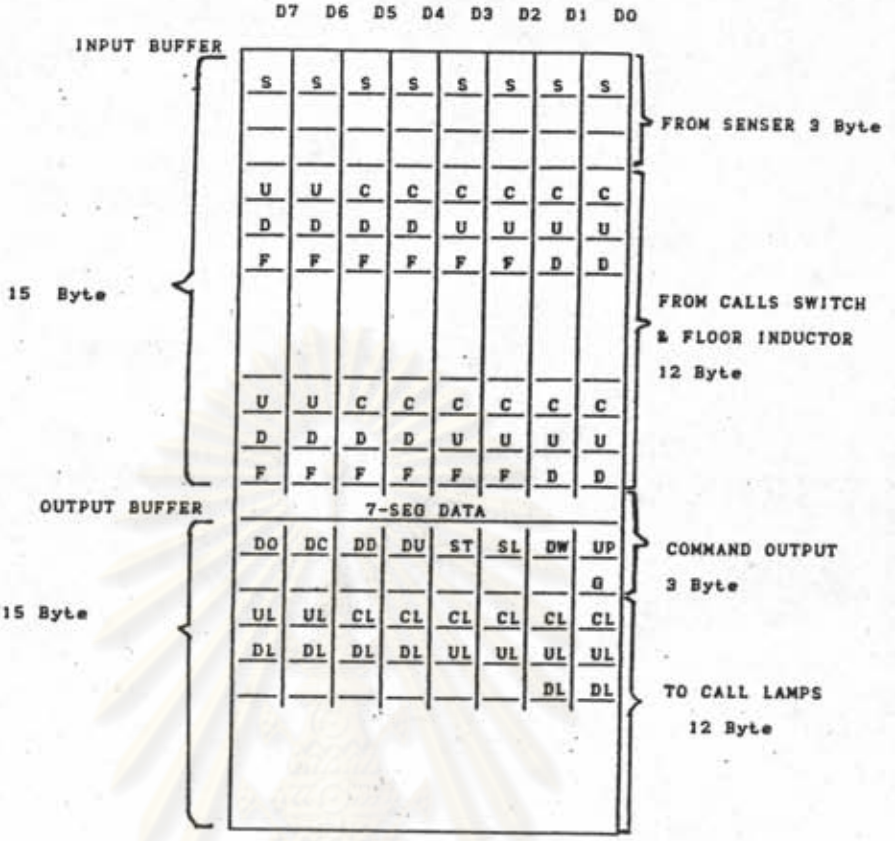
รูปที่ 6.6 การจัดโปรแกรมในหน่วยความจำ



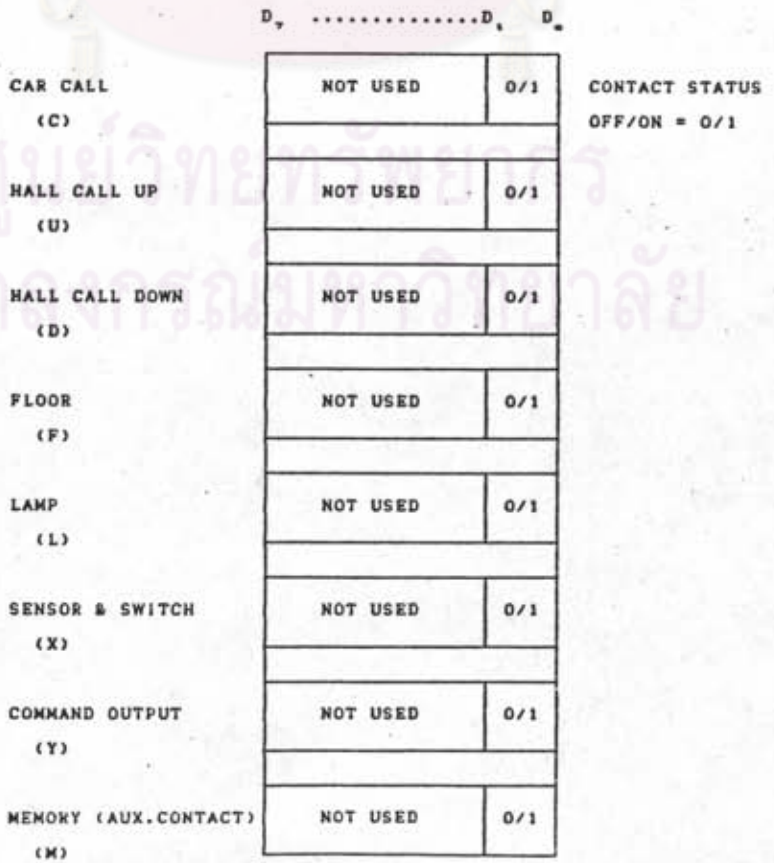
รูปที่ 6.7 การจัดข้อมูลใน RAM



6.8 การจัดข้อมูลใน I/O BUFFER



6.9 การจัดข้อมูลใน STATUS BUFFER



รายละเอียดในโปรแกรมหลักมีดังนี้

1) INITIALIZE PROGRAM เป็นโปรแกรมที่จัดการให้ระบบสามารถเริ่มต้นทำงานได้ โดยจะทำการเซ็ทค่า STACK POINTER ค่าสถานะเริ่มต้นของลิฟท์ ค่าเฉพาะของลิฟท์ ซึ่งได้แก่ จำนวนชั้น เวลาในการเปิดปิดประตู จำนวนครั้งสูงสุดของการเปิดประตูกลับเนื่องจากมีของขัดประตู จำนวนชุดของ FLOOR MODULE ค่าคงที่ของเวลาที่ใช้ในการอินเตอร์รัพท์ ซึ่งค่าเหล่านี้เป็นค่าเฉพาะของลิฟท์ในแต่ละอาคารที่ติดตั้ง ซึ่งผู้ออกแบบจะเป็นผู้กำหนดตามความเหมาะสม โดยกำหนดในโปรแกรมเป็นตารางค่าคงที่ นอกจากนี้ INITIALIZE PROGRAM ยังเซ็ท MODE การทำงานของ CTC และ 8255 บนซีพียูบอร์ด MODE การทำงานของ 8255 บนแผงวงจรที่ใช้กับเครื่องพิมพ์ MODE การทำงานของ 8255 บนอินพุทบอร์ด และยังตรวจสอบว่าเกิดรีเซ็ทเนื่องจาก POWER ON หรือ WATCHDOG

2) DIAGNOSTIC PROGRAM เป็นโปรแกรมที่ใช้ทดสอบฮาร์ดแวร์ของไมโครคอมพิวเตอร์ เพื่อช่วยให้การตรวจซ่อมง่ายขึ้น การทดสอบจะกระทำก่อนการใช้งานลิฟท์ เพื่อให้แน่ใจว่าฮาร์ดแวร์ของไมโครคอมพิวเตอร์อยู่ในสภาพพร้อมที่จะใช้งาน และทดสอบในกรณีเครื่องควบคุมลิฟท์เสียเพื่อตรวจสอบว่าเสียเนื่องจากฮาร์ดแวร์ของไมโครคอมพิวเตอร์หรือไม่ และเสียส่วนใด

ในส่วนของ DIAGNOSTIC PROGRAM สามารถแบ่งได้เป็น

ก) ROM CHECK ใช้ในการตรวจสอบ ROM ว่าโปรแกรมใน ROM ถูกต้องหรือไม่ โดยทำการบวกค่าเลขฐานสิบหกทั้งหมดใน ROM แล้วนำค่าผลบวก 2 ไบท์ หลังไปเทียบกับค่า 1630 H ซึ่งเป็นค่าผลบวกที่ถูกต้อง

ข) RAM CHECK ใช้ในการตรวจสอบ RAM โดยจะเขียนค่า OAAH (1010 1010) แล้วอ่านออกมาว่าเป็น OAAH หรือเปล่า จากนั้นจะเขียนค่า 55H (0101 0101) แล้วอ่านออกมาว่าเป็น 55H หรือเปล่า ทำเช่นนี้ทุกๆ แอดเดรสของ RAM ถ้าเขียนและอ่านได้ตรงกันและถูกต้องทุกแอดเดรส แสดงว่า RAM ใช้งานได้ปกติ

ค) OUT PUT CHECK ใช้ในการตรวจสอบเอาท์พุทบอร์ด โดยจะให้เอาท์พุทแต่ละช่อง ON แล้วแสดงผลออกที่ LED บนอินเตอร์เฟสบอร์ดจนครบทุกช่อง

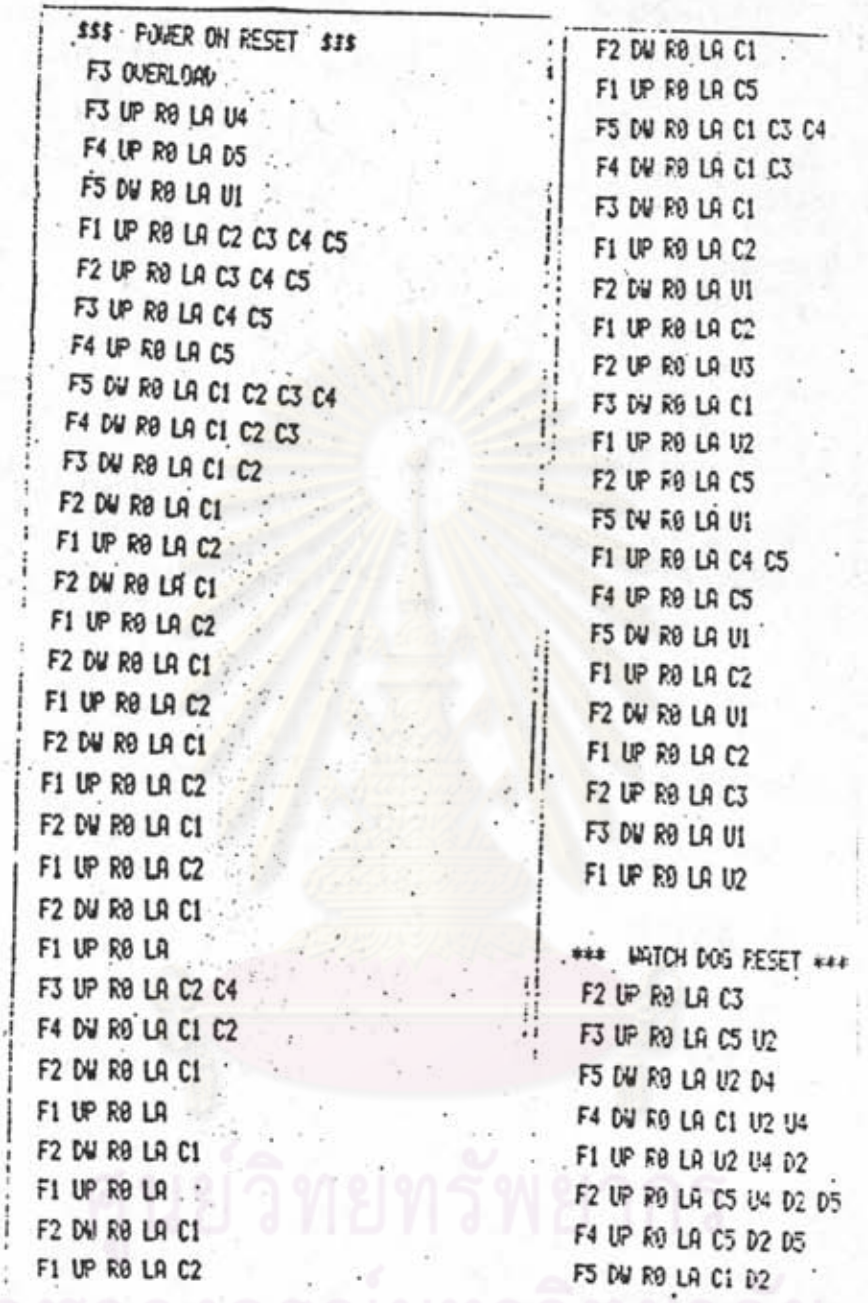
4) INPUT CHECK ใช้ในการตรวจสอบอินพุทบอร์ด โดยเมื่อได้รับสัญญาณอินพุทแล้ว จะทำการรับรู้ด้วยการให้เอาท์พุทแสดงผลออกที่ LED บนอินเตอร์เฟสบอร์ด

ในการทดสอบนี้จะต้องปลดคอนเนตเตอร์ของเอาท์พุทซึ่งต่อไปยังส่วนขับเคลื่อนลิฟท์ออกก่อนเสมอ ถ้าไม่ปลดคอนเนตเตอร์นี้ออก โปรแกรมส่วนนี้จะไม่สามารถทำงานได้

3) DATA LOGGING PROGRAM เป็นโปรแกรมที่ใช้พิมพ์สถานะและผลการทำงานของลิฟท์ ตัวอย่างการพิมพ์อยู่ในรูปที่ 6.10 ซึ่งอธิบายได้ดังนี้ ถ้าลิฟท์เริ่มทำงานจะพิมพ์ POWER ON RESET แสดงว่าการเริ่มทำงานนั้นเกิดจากการเริ่มป้อนไฟเข้าสู่เครื่องควบคุม แต่ถ้าพิมพ์ WATCHDOG RESET แสดงว่าเครื่องควบคุมได้ทำงานผิดพลาด และ WATCHDOG ได้ทำการรีเซ็ทให้เริ่มทำงานใหม่ เครื่องพิมพ์จะพิมพ์สภาวะของลิฟท์ทุกครั้งที่บลิฟท์มีการเคลื่อนที่ ข้อมูลที่พิมพ์ออกมาได้แก่ ตำแหน่งชั้นที่ลิฟท์อยู่ ทิศทางการเคลื่อนที่ จำนวนครั้งที่เกิด REOPEN เนื่องจากประตูปิดไม่สนิท ประตูปิดสุด สัญญาณปุ่มกดที่ผู้ใช้เรียก OVERLOAD, 80 % OL, การเกิดรีเซ็ทเนื่องจากการป้อนไฟเข้าใหม่, การเกิดรีเซ็ทเนื่องจาก WATCHDOG ข้อมูลที่พิมพ์มีประโยชน์ คือ ทำให้ทราบถึงการใช้งานลิฟท์ว่ามากน้อยเพียงใด และชั้นไหนที่ผู้คนใช้มาก ซึ่งอาจนำข้อมูลนี้ไปใช้ในการคำนวณหาการจราจรของลิฟท์ได้ นอกจากนี้ทำให้ทราบถึงการทำงานของประตู ซึ่งกลอุปกรณ์ประตูมักสร้างปัญหาให้แก่ช่างลิฟท์เสมอ จากข้อมูลนี้ทำให้ช่างลิฟท์สามารถทราบได้ว่ากลอุปกรณ์ทางประตูที่ชั้นใดเกิดปัญหาขึ้น จากผลการพิมพ์สามารถรู้ว่าแหล่งจ่ายไฟมีการดับบ่อยเพียงใด และไมโครคอมพิวเตอร์มีการทำงานออกนอกโปรแกรมหรือทำงานผิดปกติบ่อยเพียงใด

รายละเอียดโปรแกรมอินเตอร์รัพท์

1) INPUT FROM SENSOR & CALL เป็นโปรแกรมที่ทำหน้าที่อ่านอินพุทจากอินพุทบอร์ด การอ่านอินพุทบอร์ดของ I/O MODULE เป็นบอร์ดแรกซึ่งจะได้สัญญาณจากปล่องลิฟท์ จากนั้นจะอ่านสัญญาณจากอินพุทบอร์ดของ FLOOR MODULE ซึ่งจะได้สัญญาณของปุ่มกดและตำแหน่งชั้นที่ลิฟท์อยู่ การอ่านสัญญาณจากอินพุทบอร์ดสามารถกำหนดจำนวนบอร์ดที่จะอ่านได้ในโปรแกรม จำนวนอินพุทบอร์ดอย่างน้อยต้องมี 2 บอร์ด เป็นของ I/O MODULE 1 บอร์ด และเป็นของ FLOOR MODULE 1 บอร์ด ถ้าจำนวนชั้นสูงกว่า 6 ชั้น จะต้องเพิ่ม



รูปที่ 6.10 ตัวอย่างการพิมพ์ผล

จำนวน FLOOR MODULE ขึ้น โดย FLOOR MODULE 1 ชุด สามารถควบคุมลิฟท์ได้ 6 ชั้น และเพิ่มได้สูงสุด 3 ชุด คือ สามารถควบคุมลิฟท์ได้ 24 ชั้น การอ่านอินพุทจะอ่านผ่านทางพอร์ทของ 8255 บนอินพุทบอร์ด โดยผ่านทางพอร์ท A, B, C จำนวน 3 พอร์ทๆ ละ 8 ช่องสัญญาณ ดังนั้นการอ่านสัญญาณจะอ่านครั้งละ 8 ช่องสัญญาณ แล้วนำมาเก็บไว้ในหน่วยความจำของ INPUT BUFFER จำนวน 1 ไบท์ต่อการอ่าน 1 ครั้ง การอ่านสัญญาณจะอ่าน 3 ครั้งต่ออินพุทบอร์ด 1 บอร์ด ทำการอ่านเช่นนี้จนครบทุกบอร์ด จากนั้นจะทำการกระจายสัญญาณใน INPUT BUFFER ซึ่งแต่ละไบท์จะเก็บสัญญาณอินพุท 8 ช่องสัญญาณ ไปไว้ในบิทสุดท้ายของ STATUS BUFFER ที่ทำเช่นนี้เพื่อความสะดวกในการนำไปใช้ในโปรแกรม ดังนั้นการกระจายสัญญาณใน INPUT BUFFER 1 ไบท์ สามารถกระจายไปอยู่ใน STATUS BUFFER ได้ 8 ไบท์

2) LIFT SUPPORT FUNCTION เป็นโปรแกรมที่เขียนขึ้นเพื่อใช้งานกับลิฟท์โดยเฉพาะ โปรแกรมเหล่านี้ได้แก่

ก) Floor Position เป็นโปรแกรมย่อยที่ใช้ในการหาตำแหน่งชั้นของลิฟท์ให้ได้ผลออกมาเป็นเลขฐานสิบหก

ข) 7-Seg Floor Display เป็นโปรแกรมที่แปลงตำแหน่งชั้นของลิฟท์ซึ่งเป็นเลขฐานสิบหกให้เป็นรหัส 7-SEG เพื่อส่งไปแสดงบอกเลขชั้น

ค) Call Register มีหน้าที่รับการกดปุ่มซึ่งเกิดจากการเรียกของ Hall Call และ Car Call จะแสดงผลการ Register ด้วยการให้หลอดไฟของปุ่มนั้นติด

ง) Call Analysis ใช้ในการวิเคราะห์สัญญาณปุ่มกดว่าสัญญาณเหล่านี้ ต่ำกว่า เท่ากับ สูงกว่าชั้นที่ตัวลิฟท์อยู่ ค่าที่ได้จากการวิเคราะห์จะถูกนำไปใช้ในการทำงานของแผนภาพขั้นบันได

จ) Set Lamp เป็นการรับรู้สัญญาณของปุ่มกด แล้วทำให้หลอดไฟติด โปรแกรมนี้ถูกเรียกใช้โดย Call Register

ฉ) Clear Lamp เป็นโปรแกรมที่ใช้ดับหลอดไฟของปุ่มกดหลังจากที่ลิฟท์ได้ไปรับใช้แล้ว

3) LADDER PROGRAM เป็นรหัสที่ได้แปลงมาจากแผนภาพขั้นบันได โดยเราจะแปลงแผนภาพขั้นบันไดให้เป็นคำสั่งนิโมนิคก่อน จากนั้นก็จะป้อนคำสั่งนิโมนิคเข้าสู่คอมพิวเตอร์ ซึ่งเราจะกำหนดให้คอมพิวเตอร์ว่าคำสั่งนี้ให้แปลเป็นรหัสอะไรไว้ในตอนต้น จากนั้นคอมพิวเตอร์ก็จะแปลงภาษาโปรแกรมให้เป็นรหัส

ของโปรแกรม ซึ่งรหัสเหล่านี้ LADDER INTERPRETER จะนำไปใช้งานต่อไป

4) LADDER INTERPRETER การทำงานแสดงในรูปที่ 6.11 อธิบายการทำงานได้ดังนี้ LADDER INTERPRETER จะอ่านรหัสจาก LADDER PROGRAM แล้วนำไปตรวจในตารางเพื่อหาค่าแอดเดรสของ SERVICE ROUTINE ที่จะทำ จากนั้นจะทำตามคำสั่งใน SERVICE ROUTINE การทำงานจะวนเวียนในลักษณะนี้ จนท้ายสุดของโปรแกรมจะพบคำสั่ง END จึงจะทำการออกคำสั่งไปควบคุมการทำงานของลิฟท์

5) SERVICE ROUTINE โปรแกรมย่อยเหล่านี้เป็นโปรแกรมย่อยที่ INTERPRETER ใช้ในการแปลแผนภาพขั้นบันได SERVICE ROUTINE สามารถแบ่งได้เป็น

ก) คำสั่งพื้นฐาน คำสั่งเหล่านี้มีการทำงานที่ง่ายไม่ซับซ้อน การทำงานแสดงในตารางที่ 6.1 ซึ่งอธิบายดังนี้

ให้ A = ACCUMULATOR
X = INPUT
S = STACK
Y = OUTPUT

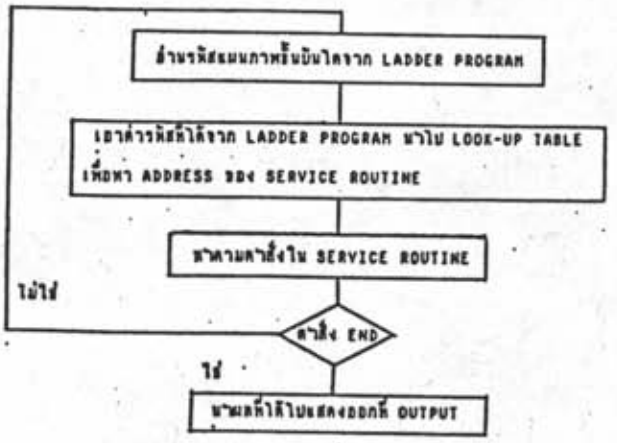
คำสั่ง LD, LDI นำค่า X หรือ $\overline{X}$ มาเก็บใน A แต่จะมีการเก็บค่าของ A ไว้ใน S ก่อน เพื่อเก็บไว้ใช้ในคำสั่ง ANB, ORB คำสั่งประเภทนี้มีขนาด 2 ไบท์ ไบท์แรกเป็น OPCODE เพื่อให้รู้ว่าการกระทำเป็นอะไร ไบท์สองเป็น OPERAND เพื่อให้รู้ตำแหน่ง CONTACT STATUS ที่กระทำ

คำสั่ง AND, ANI เป็นการนำค่า X หรือ $\overline{X}$ มาทำการ AND กับค่าใน A แล้วเก็บค่าไว้ใน A คำสั่งประเภทนี้มีขนาด 2 ไบท์เหมือนกัน

คำสั่ง OR, ORI เป็นการนำค่า X หรือ $\overline{X}$ มาทำการ OR กับค่าใน A แล้วเก็บค่าไว้ใน A คำสั่งประเภทนี้มีขนาด 2 ไบท์

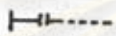
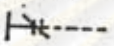
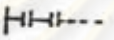
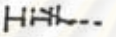
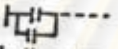
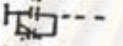
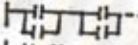
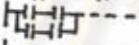
คำสั่ง ANB, ORB เป็นการนำค่าใน S มาทำการ AND หรือ OR กับค่าใน A ซึ่งเป็นสถานะในปัจจุบัน แล้วเก็บค่าไว้ใน A คำสั่งประเภทนี้มีขนาด 1 ไบท์ เพราะเป็นการกระทำอย่างเดียว ไม่เกี่ยวกับตำแหน่ง

คำสั่ง OUT เป็นการนำค่าประมวลผลใน A ไปไว้ที่ Y

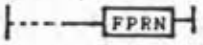
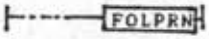
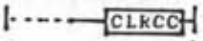
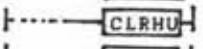
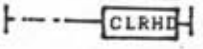


รูปที่ 6.11 การทำงานของ LADDER INTERPRETER

ตารางที่ 6.1 แสดงการทำงานของ FUNCTION SERVICE ROUTINE

ฟังก์ชั่น	สัญญลักษณ์	การทำงาน
LD		S←A; A←X
LDI		S←A; A←$\overline{X}$
AND		A←A.X
ANI		A←A.$\overline{X}$
OR		A←A+X
ORI		A←A+$\overline{X}$
ANB		A←A.S
ORB		A←A+S
OUT		Y←A
TIM		ON TIMER RELAY
CNT		COUNTER
RST		RESET COUNTER
END	END	PROGRAM END

ตารางที่ 6.2 การทำงานของ FUNCTION CALL

ฟังก์ชั่น	สัญญลักษณ์	การทำงาน
FPRN		พิมพ์การทำงานของลิฟท์
FOLPRN		พิมพ์เมื่อลิฟท์บรรทุกน้ำหนักเกิน
CLRCC		ดับหลอดไฟปุ่มกดในตัวลิฟท์
CLRHU		ดับหลอดไฟปุ่มกดหน้าชั้นขาขึ้น
CLRHD		ดับหลอดไฟปุ่มกดหน้าชั้นขาลง

ข) คำสั่ง TIM การทำงานของ TIMER เป็นแบบ ON DELAY TIMER คือ เมื่อคอยล์ของรีเลย์ ON แล้วจะมีการหน่วงเวลาไปสักครู่หนึ่งตามเวลาที่กำหนดไว้ หน้าสัมผัสรีเลย์จึงจะ ON และทันทีที่คอยล์ของรีเลย์ OFF หน้าสัมผัสก็จะ OFF ทันที คำสั่งของ TIMER จะมีขนาด 4 ไบท์ โดย 2 ไบท์แรกเป็น OPCODE และ OPERAND ส่วน 2 ไบท์หลัง เป็นการกำหนดค่าเวลาหน่วง โดยสามารถกำหนดเวลาหน่วงได้ 999.9 วินาที มีความละเอียด 0.1 วินาที

ค) คำสั่ง CNT เป็นรีเลย์ที่ใช้นับเหตุการณ์ที่เกิดขึ้นเมื่อครบจำนวนครั้งที่กำหนดไว้ หน้าสัมผัสของ COUNTER ก็จะ ON ถ้ามีการรีเซ็ท COUNTER ก็จะเป็นการตั้งค่านับขึ้นอีกครั้ง และทำให้หน้าสัมผัสของ COUNTER จะ OFF ทันที คำสั่ง COUNTER มีขนาด 4 ไบท์ เหมือน TIMER โดย 2 ไบท์หลังสามารถกำหนดค่าสูงสุดของการนับได้ 9999 ครั้ง

ง) คำสั่ง RST คำสั่งนี้ใช้ในการให้ COUNTER ทำการตั้งค่านับใหม่ และทำให้หน้าสัมผัสของ COUNTER จะ OFF ทันที คำสั่งนี้มีขนาด 2 ไบท์

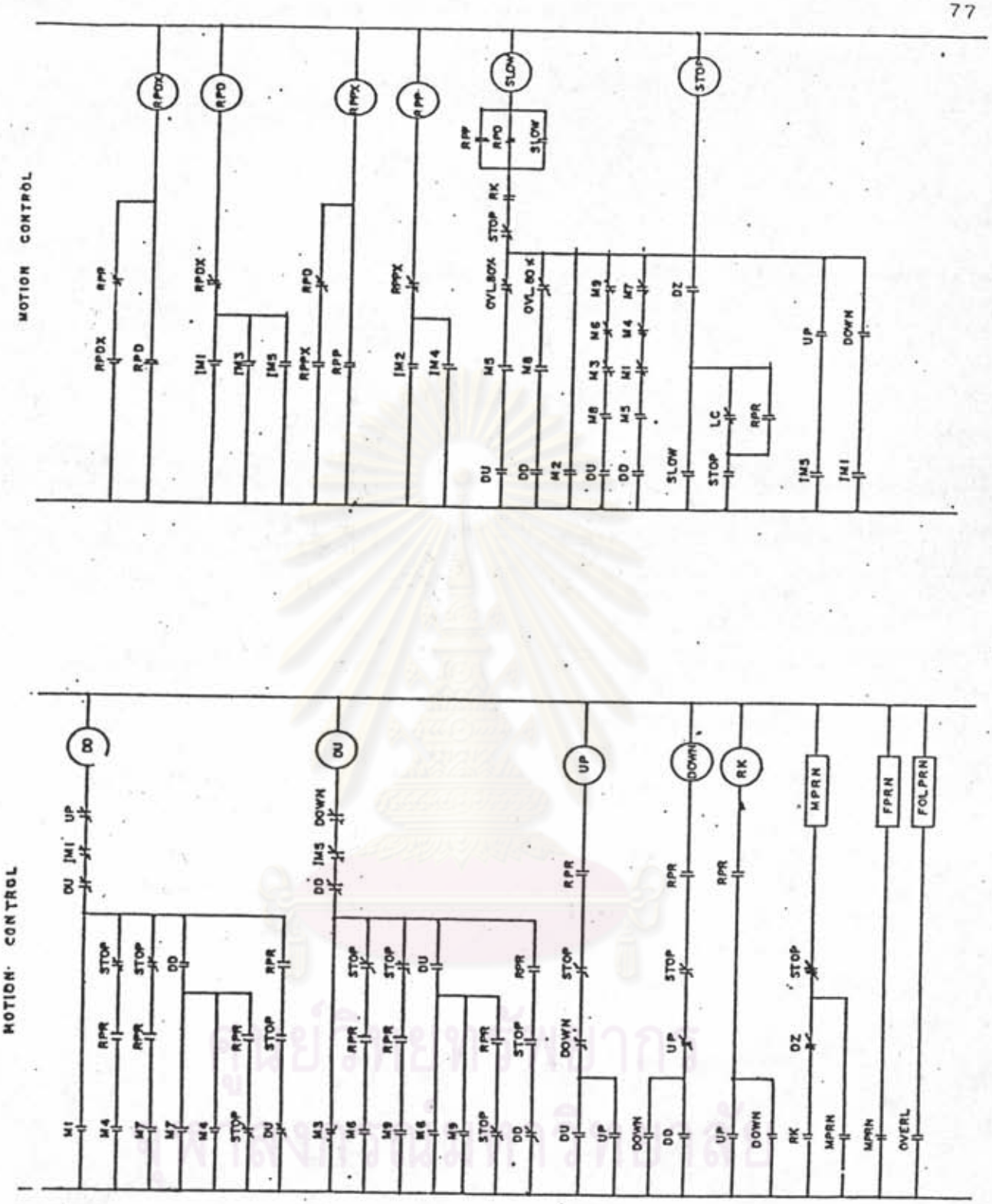
จ) คำสั่ง END เป็นคำสั่งให้ LADDER INTERPRETTER รู้ว่ารหัสใน PROGRAM LADDER ได้สิ้นสุดลงแล้ว

6) FUNCTION CALL เป็นกลุ่มของโปรแกรมย่อยที่สร้างขึ้นเพื่อให้ทำงานเฉพาะอย่าง โดย FUNCTION CALL จะทำงานหรือไม่ขึ้นกับเงื่อนไขก่อนหน้าที่ได้ทดสอบ การทำงานของ FUNCTION CALL แสดงในตารางที่ 6.2

7) OUTPUT COMMAND เป็นโปรแกรมที่ใช้จัดเตรียมคำสั่งที่จะไปควบคุมการทำงานของลิฟท์ ทั้งนี้เพราะหลังจากที่ได้ผลของคำสั่งมาแล้ว คำสั่งเหล่านี้จะกระจายอยู่ในบิทสุดท้ายของแต่ละไบท์ ดังนั้นโปรแกรมนี้จึงทำหน้าที่รวบรวมคำสั่งควบคุม โดยจะอัดคำสั่งควบคุม 8 คำสั่งไว้ในแต่ละบิทของหน่วยความจำขนาด 1 ไบท์

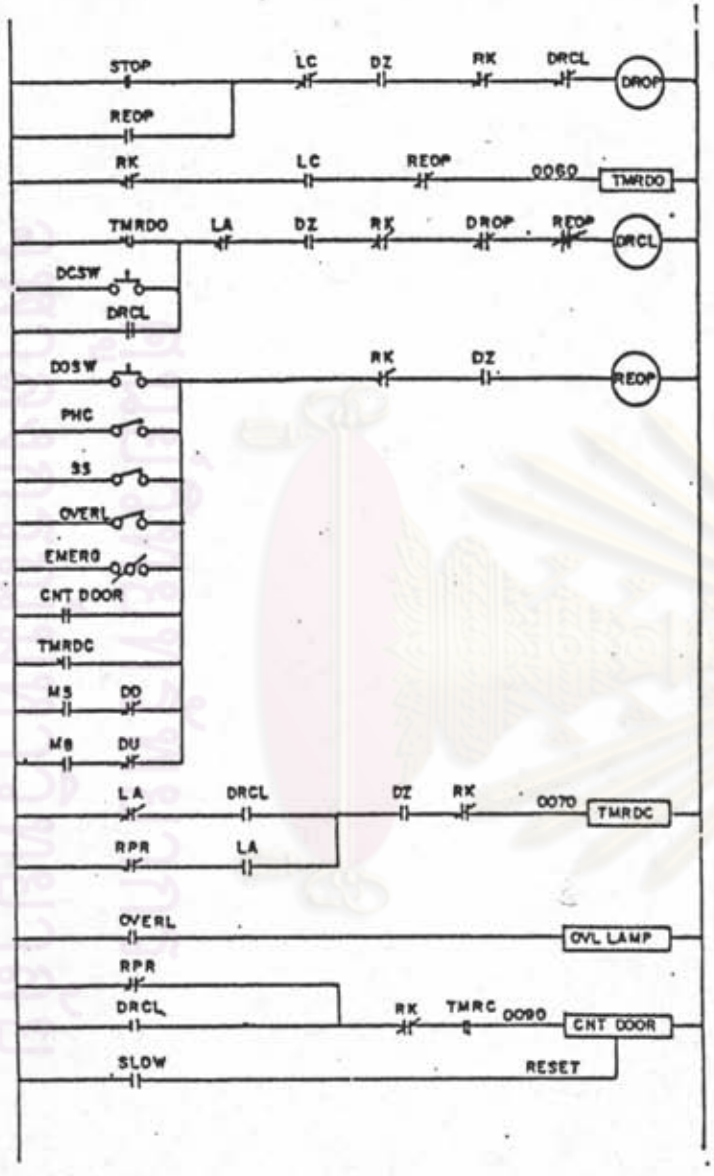
6.5 การทำงานของแผนภาพขั้นบันได

การควบคุมการทำงานของลิฟท์จะเป็นไปตามแผนภาพขั้นบันไดดังแสดงในรูปที่ 6.12 ก) และรูปที่ 6.12 ข) ซึ่งการทำงานจะกล่าวในหัวข้อต่อไปนี้

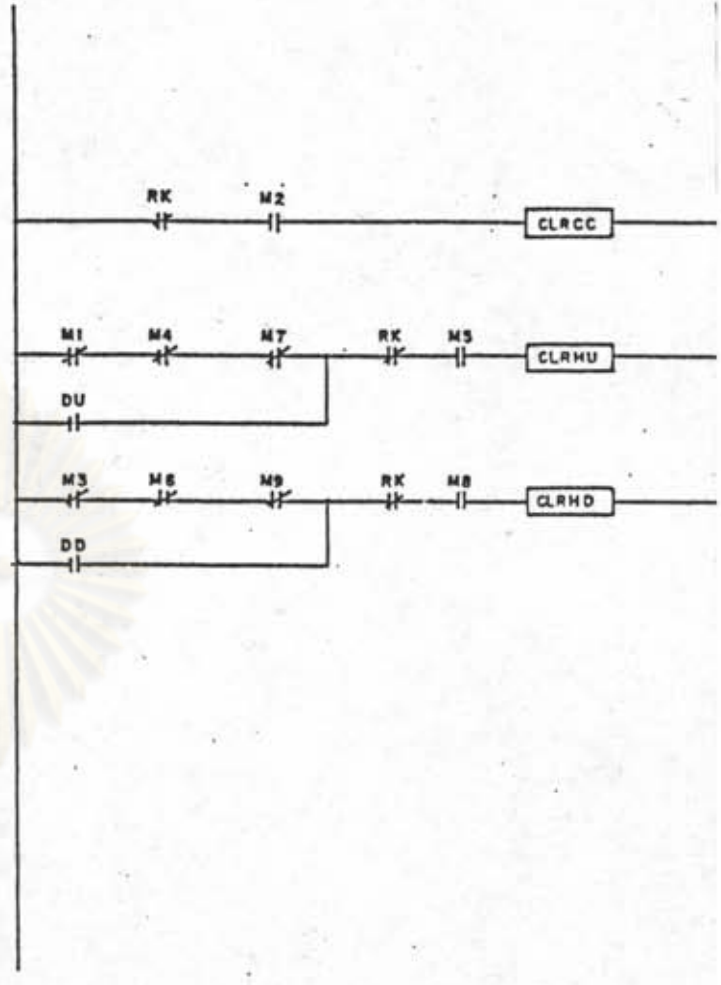

MOTION CONTROL
RPDX
RPD
RPPX
RPP
SLOW
STOP
RK
OVL_BOX
DZ
LC
UP
DOWN
MOTION CONTROL
DU
DD
UP
DOWN
RK
MPRN
FPRN
FOLPRN
OVERL

รูปที่ 6.12 ก)

DOOR CONTROL



LAMP CONTROL



รูปที่ 6.12 ข)

6.5.1 ความหมายของหน้าสัมผัสและรีเลย์

ในแผนภาพขั้นบันไดจะพบว่ามีหน้าสัมผัสที่ไม่ใช่หน้าสัมผัสของรีเลย์ในแผนภาพขั้นบันได หน้าสัมผัสเหล่านี้ได้มาจากผลการทำงานของโปรแกรม Call Analysis ใน Lift Support Function หน้าสัมผัสเหล่านี้มีความหมายดังนี้

M1 = CAR CALL ที่ต่ำกว่าชั้นที่ลิฟท์อยู่
- M1 = 1 เมื่อมีการกดเรียกต่ำกว่าชั้นที่ลิฟท์อยู่
- M1 = 0 เมื่อไม่มีการกดเรียกต่ำกว่าชั้นที่ลิฟท์อยู่

M2 = CAR CALL เท่ากับชั้นที่ลิฟท์อยู่
- M2 = 1 เมื่อลิฟท์วิ่งมาถึงชั้นที่มีการกดเรียกอยู่
- M2 = 0 เมื่อไม่มีการกดเรียกเมื่อลิฟท์วิ่งถึงชั้นที่ลิฟท์วิ่งมาถึง

M3 = CAR CALL ที่สูงกว่าชั้นที่ลิฟท์อยู่
- M3 = 1 เมื่อมีการกดเรียกชั้นที่สูงกว่าที่ลิฟท์อยู่
- M3 = 0 เมื่อไม่มีการกดเรียกสูงกว่าชั้นที่ลิฟท์อยู่

M4 = HALL UP ที่ต่ำกว่าชั้นที่ลิฟท์อยู่ (ปุ่มกดหน้าชั้นในทิศทางขึ้น △)
- M4 = 1 เมื่อมีการกดเรียกปุ่มหน้าชั้นในทิศทางขึ้น
- M4 = 0 เมื่อไม่มีการกดเรียกปุ่มหน้าชั้นในทิศทางขึ้น

M5 = HALL UP เท่ากับชั้นอยู่
- M5 = 1 เมื่อลิฟท์วิ่งขึ้นมาถึงชั้นที่มีการกดเรียกหน้าชั้นในทิศทางขึ้น
- M5 = 0 เมื่อไม่มีการกดเรียกหน้าชั้นในทิศทางที่ลิฟท์วิ่งขึ้นมาถึง

M6 = HALL UP ที่สูงกว่าชั้นที่ลิฟท์จอดอยู่
- M6 = 1 เมื่อมีการกดเรียกหน้าชั้นในทิศทางขึ้นที่สูงกว่าชั้นที่ลิฟท์อยู่
- M6 = 0 เมื่อไม่มีการกดเรียกหน้าชั้นในทิศทางขึ้นที่สูงกว่าชั้นที่ลิฟท์อยู่

M7 = HALL DOWN ที่ต่ำกว่าชั้นที่ลิฟท์อยู่ (ปุ่มกดหน้าชั้นในทิศทางลง ▽)
- M7 = 1 เมื่อมีการกดเรียกปุ่มหน้าชั้นในทิศทางลงที่ต่ำกว่าชั้นที่ลิฟท์อยู่
- M7 = 0 เมื่อไม่มีการกดเรียกปุ่มหน้าชั้นในทิศทางลงที่

ต่ำกว่าชั้นที่ลิฟท์อยู่

M8 = HALL DOWN เท่ากับชั้น

M8 = 1 เมื่อลิฟท์วิ่งลงมาถึงชั้นที่มีการกดเรียกหน้าชั้นในทิศทางลง

M8 = 0 เมื่อไม่มีการกดเรียกหน้าชั้นในทิศทางลงขณะที่ลิฟท์วิ่งลงมาถึง

M9 = HALL DOWN ที่สูงกว่าชั้นที่ลิฟท์อยู่

M9 = 1 เมื่อมีการกดเรียกปุ่มหน้าชั้นในทิศทางลงที่สูงกว่าชั้นที่ลิฟท์อยู่

M9 = 0 เมื่อไม่มีการกดเรียกปุ่มหน้าชั้นในทิศทางลงที่สูงกว่าชั้นที่ลิฟท์อยู่

IM1, IM2, IM3, IM4, IM5 = ตัวบอกตำแหน่งขณะที่ลิฟท์อยู่ เมื่อลิฟท์อยู่ชั้นไหน IM... ชั้นนั้นจะเท่ากับ 1 นอกนั้นเป็น 0 หมด เช่น เมื่อลิฟท์จอดอยู่ชั้น 1 IM1 จะเท่ากับ 1, IM2, IM3, IM4, IM5 = 0 หมด

ความหมายของรีเลย์ในแผนภาพขั้นบันไดมีดังนี้

รีเลย์ DD = ตัวกำหนดทิศทางลง เป็น 1 เมื่อถูกกำหนดให้เป็นทิศทางลง (DIRECTION DOWN)

รีเลย์ DU = ตัวกำหนดทิศทางขึ้น เป็น 1 เมื่อถูกกำหนดให้เป็นทิศทางขึ้น (DIRECTION UP)

รีเลย์ UP = ตัวกำหนดให้ Power Contactor Up ทำงาน เป็น 1 เมื่อ DU ถูกกำหนดให้เป็น 1

รีเลย์ DOWN = ตัวกำหนดให้ Power Contactor Down ทำงาน เป็น 1 เมื่อ DD ถูกกำหนดให้เป็น 1

รีเลย์ RK = RUNNING RELAY

รีเลย์ MPRN = คำสั่งให้ PRINTER ทำงาน

รีเลย์ FPRN = รีเลย์สำหรับพิมพ์สถานะการทำงานของลิฟท์

รีเลย์ FOLPRN = รีเลย์สำหรับพิมพ์ผลเมื่อลิฟท์รับน้ำหนักบรรทุกเกินอัตรา (Overload)

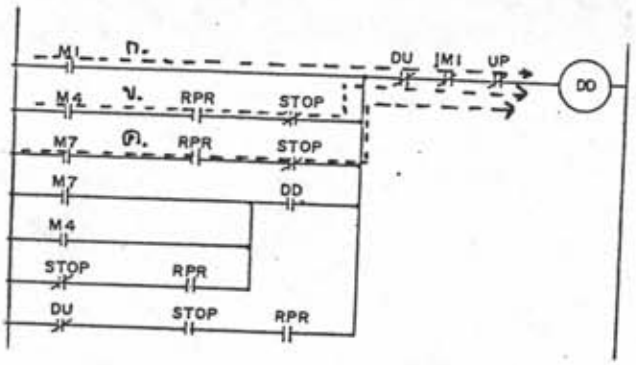
รีเลย์ RPD = ลิฟท์อยู่ชั้นคี่ รีเลย์ RPD = 1
รีเลย์ RPDX = รีเลย์ช่วยของ RPD
รีเลย์ RPP = ลิฟท์อยู่ชั้นคู่ รีเลย์ RPP = 1
รีเลย์ RPPX = รีเลย์ช่วยของ RPP
รีเลย์ SLOW = ตัวกำหนดให้ลิฟท์วิ่งช้าลง รีเลย์ SLOW = 1
รีเลย์ STOP = คำสั่งให้ลิฟท์หยุด
รีเลย์ DROP = คำสั่งให้ประตูลิฟท์เปิด เมื่อประตูเปิด DROP = 1
รีเลย์ TMRDO = TIMER กำหนดเวลาให้ประตูเปิดค้างนานเท่าไหร่
รีเลย์ DRCL = คำสั่งให้ประตูลิฟท์ปิด เมื่อประตูปิด DRCL = 1
รีเลย์ REOP = คำสั่งให้ประตู RE-OPEN
รีเลย์ TMRDC = TIMER กำหนดเวลาให้ประตูปิดกลับ เมื่อการทำงานของระบบประตูผิดปกติ
รีเลย์ OVL LAMP = OVER LOAD LAMP
รีเลย์ CNT DOOR = COUNTER DOOR ตัวกำหนดเมื่อระบบประตูผิดปกติจะนับจำนวนครั้งที่ปิดกลับ
รีเลย์ CLRCC = CLEAR CAR CALL
รีเลย์ CLRHU = CLEAR HALL CALL UP
รีเลย์ CLRHD = CLEAR HALL CALL DOWN

6.5.2 การทำงานของ Motion Control

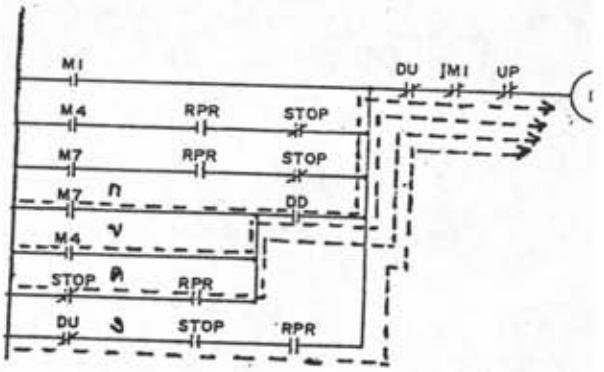
ในแผนภาพขั้นบันไดที่แสดงนี้ ใช้สำหรับควบคุมลิฟท์ 5 ชั้น ถ้าในการควบคุมลิฟท์ที่มีจำนวนชั้นต่างไปจากนี้ จะต้องแก้ไขโปรแกรมในบางส่วน สมมุติให้ลิฟท์อยู่ชั้นที่ 3 ในรูปที่ 6.13 รีเลย์ DD จะเป็น 1 เมื่อ

1) มีคนกด CAR CALL ต่ำกว่าชั้นที่ลิฟท์อยู่ เช่น กดเรียกชั้น 1 หรือ 2 ก็ได้ จะทำให้ M1 = 1 ไม่มีทิศทางขึ้น DU = 0 ลิฟท์ไม่ได้จอดชั้น 1 IM1 = 0 ลิฟท์ไม่ได้วิ่งขึ้น UP = 0 ฉะนั้นในกรณีมีคนเรียก CAR CALL ต่ำกว่าชั้น LADDER จะทำงานตามรูปตร ก. ในรูปที่ 6.13

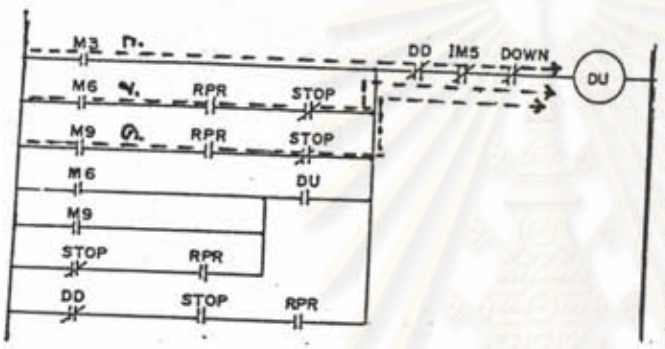
2) มีคนกดเรียก HALL UP ที่ต่ำกว่าชั้นที่ลิฟท์อยู่ โดยจะผ่าน M4 = 1 (RPR = 1 เมื่อประตูปิดสนิทแล้ว เพื่อเปิดโอกาสให้ผู้โดยสารที่เข้าอยู่ในลิฟท์มีสิทธิ์กดเรียก ใช้ลิฟท์ก่อน ในกรณีที่ผู้ใช้ลิฟท์ต้องการใช้ลิฟท์ที่ทิศทางตรงกันข้ามกัน)



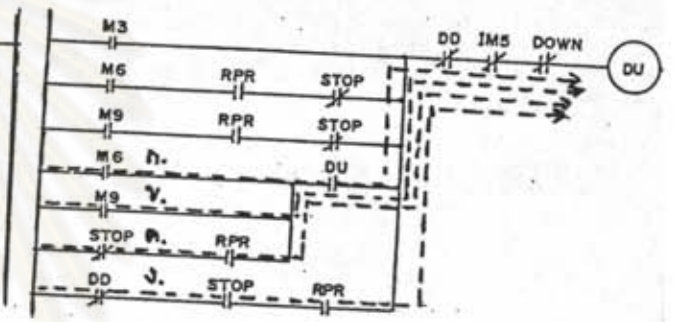
รูปที่ 6.13



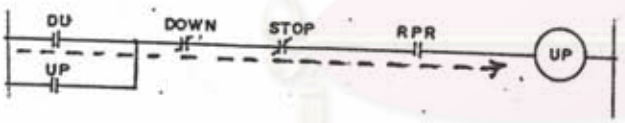
รูปที่ 6.14



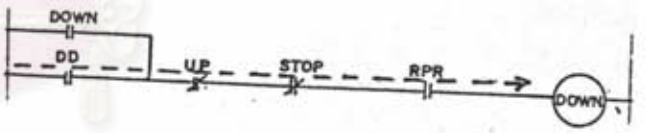
รูปที่ 6.15



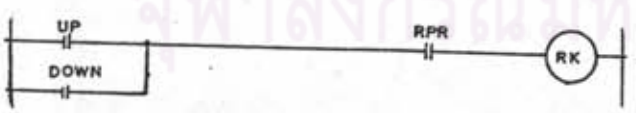
รูปที่ 6.16



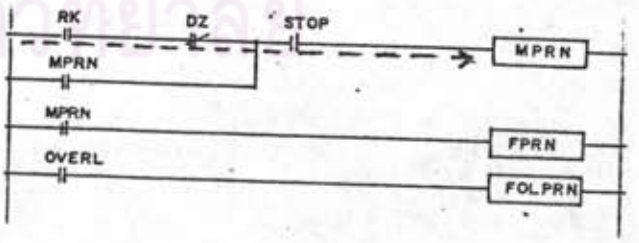
รูปที่ 6.17



รูปที่ 6.18



รูปที่ 6.19



รูปที่ 6.20

ถ้ากรณีที่ไม่มีผู้โดยสารใช้ลิฟท์ภายในตัวลิฟท์ คนที่กดเรียกหน้าชั้น HALL UP มีสิทธิ์ก่อนเมื่อประตูปิดสนิทแล้ว และผ่าน STOP = 0 (เพราะลิฟท์จอด) หน้าสัมผัส STOP = 0 ใช้ HOLD ให้รีเลย์ DD = 1 จนกว่าลิฟท์จะมีคำสั่งหยุด รีเลย์ STOP ถึงจะไป RESET ให้รีเลย์ DD เป็น 0 (รูปที่ 6.13) ฉะนั้นในกรณีคนเรียก HALL UP ต่ำกว่าชั้นจะทำงานตาม LADDER ลูกศร ข. ในรูปที่ 6.13

3) มีคนกดเรียก HALL DOWN ต่ำกว่าชั้นที่ลิฟท์อยู่จะทำให้ M7 = 1 การทำงานของ LADDER จะตามลูกศร ค. ตามรูปที่ 6.13 (การทำงานของ LADDER RPR และ STOP เหมือนข้อที่ 2)

4) ในกรณีลิฟท์วิ่งลงมี HALL UP ต่ำกว่า จะผ่านตาม LADDER ลูกศร ก. ในรูปที่ 6.14 และ HALL DOWN ตามลูกศร ข. สำหรับตามลูกศร ค. ใช้สำหรับ HOLD รีเลย์ DD ให้เป็น 1 ก่อน เพราะตอนมีคำสั่งให้ SLOW M7 และ M4 จะเป็น 0 ทันที ดังนั้นตามลูกศร ค. จะช่วย HOLD จนกระทั่งมีคำสั่ง STOP และในขณะเดียวกันตามลูกศร ง. จะช่วย HOLD รีเลย์ DD ต่อจนกระทั่งลิฟท์จอดสนิท และประตูเริ่มเปิดออก รีเลย์ DD = 0 ตามรูปที่ 6.14

การทำงานของ LADDER ตามรูปที่ 6.15 นี้ เหมือนกับ LADDER ในรูปที่ 6.13 แต่ทิศทางกลับกัน รีเลย์ DD จะเป็น 1 เมื่อ

1) เมื่อมีการกดเรียก CAR CALL ที่สูงกว่าชั้นที่ลิฟท์อยู่ จะทำให้ M3 = 1 และรีเลย์ DD = 1 ตามลูกศร ก. ในรูปที่ 6.15

2) เมื่อมีการกดเรียก HALL UP ที่สูงกว่าลิฟท์อยู่ (ขณะที่ประตูกำลังเปิดอยู่ ลิฟท์กำลังวิ่งอยู่) รีเลย์ DD จะ = 1 ก็ต่อเมื่อรอเวลาให้ประตูปิดสนิท (RPR = 1) เสียก่อน เพื่อให้สิทธิ์ผู้ที่อยู่ที่ลิฟท์สามารถเป็นผู้ที่เลือกทิศทางก่อน ตามลูกศร ข. ในรูปที่ 6.15

3) ในกรณีที่มีการเรียก HALL DOWN ที่สูงกว่าลิฟท์อยู่ (ในขณะที่ประตูกำลังเปิดหรือลิฟท์กำลังวิ่งลง) รีเลย์ DD จะ = 1 ตามกรณีเหมือนข้อ 2 ตามลูกศร ค. ในรูปที่ 6.15

4) ในกรณีลิฟท์วิ่งขึ้นมี HALL UP สูงกว่าจะผ่านทางลูกศร ก. และ HALL DOWN ตามลูกศร ข. ในรูปที่ 6.16 สำหรับลูกศร ค. ใช้สำหรับ HOLD รีเลย์ DU ให้เป็น 1 ก่อน เพราะตอนมีคำสั่ง SLOW M6 และ M9 จะเป็น 0 ทันที ดังนั้น ลูกศร ค. จะช่วย HOLD จนกระทั่งมีคำสั่ง STOP ใน

ขณะเดียวกันลูกศร ง. จะช่วย HOLD รีเลย์ DU ต่อจนกระทั่งลิฟท์จอดสนิทและประตูเริ่มเปิดออก และ RPR = 0 จะทำให้ DU = 0 ด้วย ตามรูปที่ 6.16

เมื่อมีการกดเรียกจนทำให้เกิดการทำงานของรีเลย์แสดงทิศทางแล้ว ต่อไปจะเป็นการทำงานของรีเลย์ที่เกี่ยวกับการเคลื่อนที่

<u>คำสั่งให้ลิฟท์วิ่งขึ้น</u>

รีเลย์ UP = 1 เมื่อทิศทางขึ้นถูกกำหนดโดย DU ตามลูกศรในรูปที่ 6.17 DU เป็น = 1 เพราะถูกกำหนดทิศทางให้ขึ้น DOWN = 0 เพราะไม่มีการวิ่งลง STOP = 0 เพราะลิฟท์ไม่วิ่ง RPR = 1 เมื่อประตูปิดสนิท และรีเลย์ UP จะ HOLD ตัวเองจนกระทั่งมีการเกิดคำสั่ง STOP ขึ้น (STOP = 1)

<u>คำสั่งให้ลิฟท์วิ่งลง</u>

รีเลย์ DOWN = 1 เมื่อทิศทางลงถูกกำหนดโดย DD (รูปที่ 6.13) ตามลูกศรในรูปที่ 6.18 DD = 1 เพราะถูกกำหนดทิศทางให้ลง UP = 0 เพราะไม่มีการวิ่งขึ้น STOP = 0 เพราะลิฟท์ไม่วิ่ง RPR = 1 เมื่อประตูปิดสนิท และรีเลย์ DOWN จะ HOLD ตัวเองจนกระทั่งมีคำสั่ง STOP ขึ้น (STOP = 1) ในรูปที่ 6.19 RK = 1 เมื่อมีคำสั่ง UP หรือ DOWN เกิดขึ้น และ RPR มีไว้เพื่อป้องกันมิให้เกิดลิฟท์วิ่งได้ เมื่อประตูยังเปิดอยู่หรือประตูปิดสนิทแต่วงจรทางไฟฟ้ายังไม่ต่อ ฉะนั้น RPR จะเท่ากับ 1 ก็ต่อเมื่อวงจรทางไฟฟ้าจะต้องถูกต้องทุกชั้น

<u>คำสั่งให้ Printer ทำงาน</u>

เมื่อเกิดคำสั่งทำให้ลิฟท์วิ่งขึ้น, ลง ตามรูปที่ 6.19 ทำให้ RK = 1 และเมื่อลิฟท์วิ่งพ้น Door Zone จะทำให้ DZ = 0 และ STOP = 0 ทำให้ MPRN = 1 ตามลูกศรรูปที่ 6.20 เมื่อ MPRN = 1 จะทำให้เครื่องพิมพ์ทำงานพิมพ์ผลสถานะของลิฟท์ออกมา และในกรณีเกิด

Overload ขึ้น จะทำให้เครื่องพิมพ์ทำงานพิมพ์ผล Overload ออกมา

การทำงานของ RPP และ RPD

รีเลย์ RPP และ RPD เป็นรีเลย์สำหรับบอกชั้นคู่ชั้นคี่ ซึ่งจะใช้ประโยชน์ในการสั่งลิฟท์ให้ SLOW

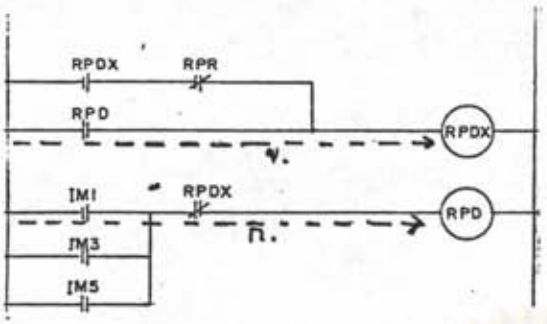
รีเลย์ RPD = 1 เมื่อลิฟท์อยู่ในตำแหน่งของชั้นคี่ (ของชั้นที่ลิฟท์อยู่) เมื่อลิฟท์อยู่ชั้นคี่ IM1, IM3, IM5 จะถูกกำหนดให้เป็น 1 โดย INPUT เป็นคำสั่งจาก INDUCTOR ในปล่องลิฟท์เป็นตัวกำหนด

สมมุติเมื่อลิฟท์อยู่ในตำแหน่งชั้น 1 INDUCTOR ชั้น 1 จะทำงานและทำให้ CONTACT บิดลง GROUND และทำให้ IM1 = 1 ในกรณีชั้นอื่นก็เหมือนกัน เมื่อ IM1 = 1 จะทำให้ RPD = 1 ตามลูกศร ก. รูปที่ 6.21 RPDX = 0 (เพราะ RPDX จะ = 1 เมื่อ LADDER ทำงานจนครบแล้วกลับมาเริ่มใหม่) RPDX เป็นตัวช่วย DELAY ให้ RPD = 1 ชั่วขณะหนึ่ง
รีเลย์ RPDX = 1 ตามลูกศร ข. และจะ HOLD ตัวเองจนกระทั่งลิฟท์วิ่งไปถึงชั้นคู่ และ RPP = 1 ในกรณีเดียวกัน RPDX จะทำให้ RPD = 0 ด้วย การทำงานของ RPP และ RPDX จะเป็นในทำนองเดียวกัน

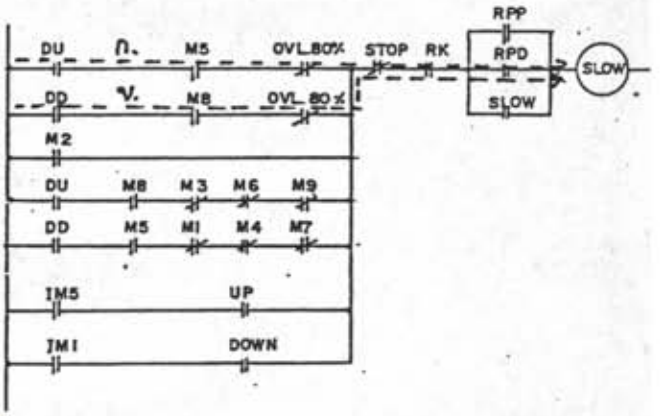
คำสั่ง SLOW

SLOW จะเท่ากับ 1 ได้ก็ต่อเมื่อ

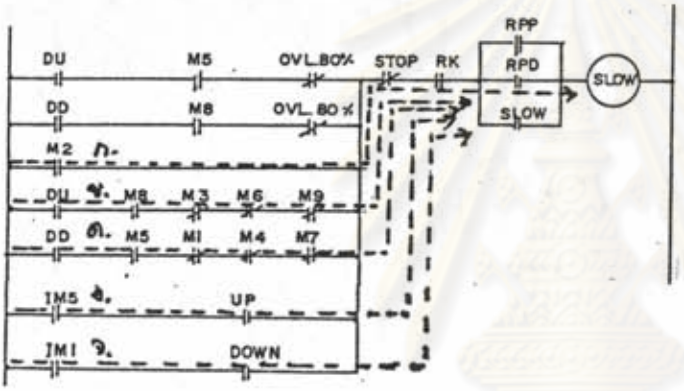
1) เมื่อลิฟท์วิ่งขึ้น (DU = 1) ถึงชั้นที่ HALL UP เท่ากับชั้นที่มีการกดเรียกไว้อยู่ (M5 = 1) ไม่มีการบรรทุกน้ำหนักเกิน 80 % (OVERL 80 % = 0) ลิฟท์กำลังวิ่งอยู่ (STOP = 0) (RK = 1) และอยู่ระหว่างที่ลิฟท์กำลังเปลี่ยนชั้นอยู่ (ระหว่างชั้นคู่เป็นคี่ หรือชั้นคี่เป็นชั้นคู่ RPP หรือ RPD ตัวใดตัวหนึ่ง = 1) และ SLOW = 1 ช่วงระหว่างนั้น SLOW จะ HOLD ตัวเอง เพราะ RPD และ RPP จะเป็น 1 อยู่ชั่วขณะเท่านั้น ตามลูกศร ก. ในรูปที่ 6.22 ลิฟท์จะ SLOW ไปจนกว่าจะเกิดคำสั่ง STOP ในรูปที่ 6.24



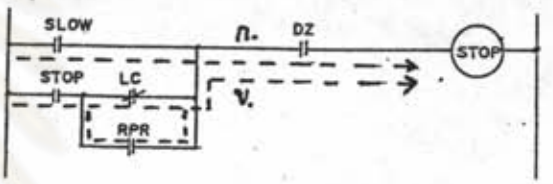
รูปที่ 6.21



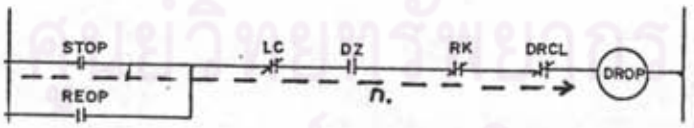
รูปที่ 6.22



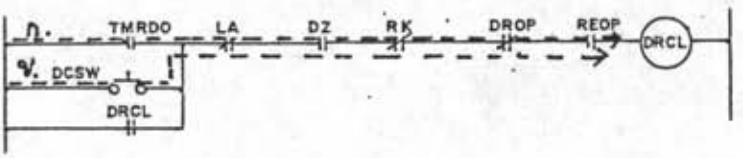
รูปที่ 6.23



รูปที่ 6.24



รูปที่ 6.25



รูปที่ 6.26

2) เมื่อลิฟท์วิ่งลง (DD = 1) ถึงชั้นที่มี HALL DOWN เท่ากับชั้นที่เรียกไว้อยู่ (M8 = 1) ไม่มีการบรรทุกน้ำหนักเกิน 80 % (OVERL 80 % = 0) ตามลูกศร ข. ในรูปที่ 6.22 ลิฟท์จะ SLOW ไปจนกระทั่งมีคำสั่ง STOP ในรูปที่ 6.24

<u>หมายเหตุ</u> ถ้ามีการบรรทุกน้ำหนักเกินเมื่อใด OVERL 80 % จะเท่ากับ 1 และจะไม่มีการจอด SLOW ชั้นนั้น (จึงไม่เกิดมีการจอดรับสัญญาณหน้าชั้น ลิฟท์จะผ่านไปจอดเฉพาะชั้นที่มีการกดเรียกในตัวลิฟท์เท่านั้น) เป็นสัญญาณ INPUT มาจากตัวลิฟท์

3) เมื่อลิฟท์วิ่งไปถึงชั้นที่มีการกดเรียก CAR CALL เท่ากับชั้นที่เรียกไว้อยู่ (ทั้งขึ้นหรือลง) ตามลูกศร ก. ในรูปที่ 6.23

4) เมื่อลิฟท์วิ่งขึ้น (DU = 1) ถึงชั้นที่มีการกดเรียก HALL DOWN เท่ากับชั้น (M8 = 1) และไม่มีการกดเรียก CAR CALL ที่สูงกว่าชั้น (M3 = 0) ไม่มีการกดเรียก HALL UP ที่สูงกว่าชั้น (M6 = 0) ไม่มีการกดเรียก HALL DOWN ที่สูงกว่าชั้น (M9 = 0) จะเกิดคำสั่ง SLOW ตามลูกศร ข. ในรูปที่ 6.23

* แต่ถ้าเกิดกรณีที่มีการกดเรียก M3, M6 หรือ M9 = 1 ขึ้นเมื่อใดลิฟท์จะไม่จอด จะไปจอดตามที่มี CALL สูงสุด ในลูกศร ก. ในรูปที่ 6.22

5) เมื่อลิฟท์วิ่งลง (DD = 1) ถึงชั้นที่มีการกดเรียก HALL UP เท่ากับชั้น (M5 = 1) และไม่มีการเรียก CAR CALL ที่ต่ำกว่า (M1 = 0) ไม่มีการเรียก HALL UP ที่ต่ำกว่า (M4 = 0) ไม่มี HALL DOWN ที่ต่ำกว่า (M7 = 0) จะเกิดคำสั่ง SLOW ตามรูปที่ 6.23 ลูกศร ค.

* ถ้าเกิดที M1, M4 หรือ M7 ขึ้นเมื่อใด ลิฟท์จะไปจอดตามลูกศรในรูปที่ 6.22

<u>ในกรณีข้อ</u> ที่ 4 และ 5 ใช้สำหรับเมื่อมีการกดเรียก HALL CALL ที่ตรงข้ามกับทิศทางที่ลิฟท์วิ่ง เช่น มี HALL CALL UP ที่อยู่ต่ำกว่าลิฟท์ ดังนั้นลิฟท์จะวิ่งลงมารับ HALL UP และมี HALL CALL DOWN ที่อยู่สูงกว่าลิฟท์ ลิฟท์จะวิ่งขึ้นไปรับ HALL DOWN

6) ลิฟท์จะถูกบงคับให้เข้า SLOW เมื่อลิฟท์วิ่งขึ้นถึงชั้นสุดท้าย ตามรูปที่ 6.23 ลูกศร ง. ถ้าลิฟท์ไม่สามารถเข้า SLOW ได้ตามปกติในข้อที่ 1 และข้อ 4

7) ลิฟท์จะถูกบังคับให้เข้า SLOW เมื่อลิฟท์วิ่งลงถึงชั้นสุดท้าย ตามรูปที่ 6.23 ลูกศร จ. ถ้าลิฟท์ไม่สามารถเข้า SLOW ได้ตามปกติในข้อที่ 2 และข้อที่ 5

คำสั่ง STOP

คำสั่ง STOP จะเกิดขึ้น (STOP = 1) เมื่อเกิด SLOW = 1 แล้ว และเมื่อลิฟท์วิ่งลงมาถึงเขตของ Door Zone (เป็นตัวปรับให้ชั้นจอดเสมอชั้นด้วย) เป็นคำสั่ง INPUT มาจากปล่องลิฟท์ (DZ = 1) ทำให้ STOP = 1 ตามรูปที่ 6.24 ลูกศร ก. และ STOP จะ HOLD ตัวเองตามรูปที่ 6.24 และจะ HOLD จนกว่าประตูเปิดสุด (LC = 1) สำหรับหน้าสัมผัสที่ต่อขนานอยู่กับ LC เพื่อช่วยป้องกันเมื่อลิฟท์วิ่ง ถ้า LC เกิดกระพริบ (PULSE) ขึ้นจะทำให้ลิฟท์หยุด

6.5.3 การทำงานของ Door Control

คำสั่ง DROP

รีเลย์ DROP เป็นคำสั่งให้ประตูเปิด DROP = 1 ก็ต่อเมื่อ

1) คำสั่ง STOP = 1, LC = 0 เพราะประตูยังปิดอยู่ DZ = 1, RK = 0 เพราะลิฟท์จอด, DRCL = 0 เพราะไม่มีคำสั่งให้ประตูปิด เงื่อนไขเหล่านี้จะทำให้ DROP = 1 ตามลูกศร ก. รูปที่ 6.25 DROP จะเป็น 1 จนกระทั่งประตูปิดสุด LC = 1 จะทำให้ DROP = 0

2) มีคำสั่ง REOP (RE-OPEN) ขณะที่ประตูกำลังปิด รีเลย์ TMRDO เป็น Timer Relay ทีจะ Hold ให้ประตูเปิดอยู่จนครบ 6 วินาที

คำสั่ง DRCL

รีเลย์ DRCL เป็นคำสั่งให้ประตูปิด DRCL = 1 ก็ต่อเมื่อ

1) เมื่อไม่มีคนใช้ลิฟท์ (ประตูปิดเองตามปกติ) เมื่อครบเวลาของการเปิดประตูแล้ว TMRDO = 1, ขณะนั้นประตูยังปิดไม่สนิท LA = 0, ลิฟท์อยู่ใน Door Zone DZ = 1, ลิฟท์จอดอยู่ RK = 0, DROP = 0 เพราะประตูปิดสุดแล้ว ไม่มีการกด Re-opne REOP = 0 จากเงื่อนไขเหล่านี้จะทำให้ DRCL = 1 ตามลูกศร ก. รูปที่ 6.26 DRCL จะ Hold

ตัวเอง (เนื่องจาก TMRDO จะ ON ชั่วขณะ) จนกระทั่งประตูปิดสุด LA = 1 ทำให้ DRCL = 0

2) ในกรณีที่คนใช้ลิฟท์ (ที่อยู่ในลิฟท์) ต้องการกดเร่งให้ประตูปิดเร็วๆ ขึ้น โดยการกด Door Close Bulton (ปุ่มกดในลิฟท์) ตามลูกศร ข. ก็จะทำให้ประตูปิดทันทีโดยไม่ต้องรอเวลาการปิดจาก TMRDO

คำสั่ง REOP

รีเลย์ REOP จะเป็นรีเลย์เกี่ยวกับการเปิดประตูกลับออกใหม่ (RE - OPEN) ในขณะที่ประตูกำลังปิด ในรูปที่ 6.27 REOP = 1 เมื่อ

1) มีการกดปุ่ม Door Open (ปุ่มกดในตัวลิฟท์) ลูกศร ก.

2) มีคนหรือสิ่งของบังแสงของ PHOTO CELL (อยู่ที่ขอบประตูตัวลิฟท์) ลูกศร ข.

3) มีคนหรือสิ่งของกระทบ SAFETY SHOSE (อยู่ที่บานประตูตัวลิฟท์) ลูกศร ค.

4) มีการบรรทุกน้ำหนักเกินตามที่ตั้งไว้ (อยู่ที่ตัวลิฟท์) ลูกศร ง.

5) มีการกดสวิทช์ STOP (อยู่ที่แผงปุ่มกดในลิฟท์) ลูกศร จ.

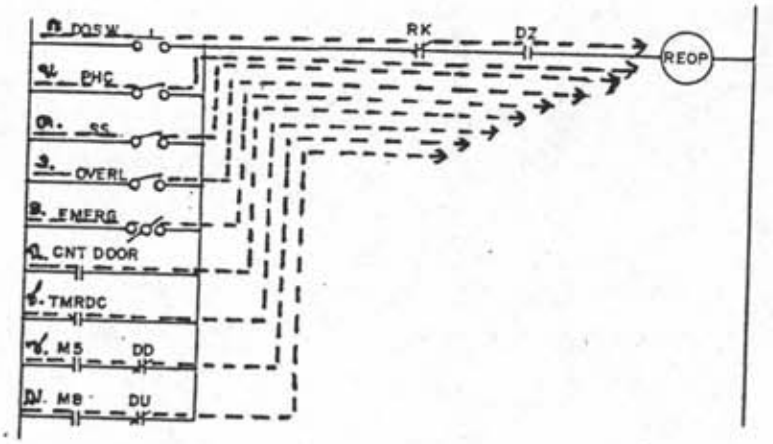
6) มีการผิดปกติของระบบประตู ประตูไม่สามารถปิดสนิทได้จะเกิดการนับเวลาขึ้น ลูกศร ฉ. (ดูการทำงานของ CNT DOOR ในหัวข้อถัดไป)

7) มีการผิดปกติของ CONTACT ประตูทางไฟฟ้า เมื่อประตูปิดสุดแล้ว แต่ CONTACT (วงจรทางไฟฟ้าไม่ครบวงจร) ประตูไม่สมบูรณ์ ลูกศร ช. (ดูการทำงาน TMRDC ในหัวข้อถัดไป)

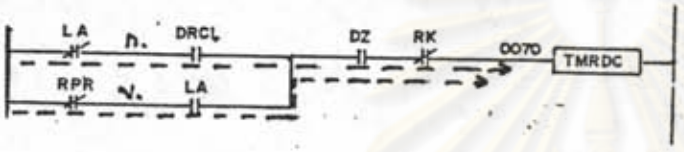
8) มีการกดปุ่ม HALL CALL ขาขึ้น (HALL UP เท่ากับชั้น) หน้าชั้น ลูกศร ซ.

9) มีการกดปุ่ม HALL CALL ขาลง (HALL DOWN เท่ากับชั้น) หน้าชั้น ลูกศร ฌ.

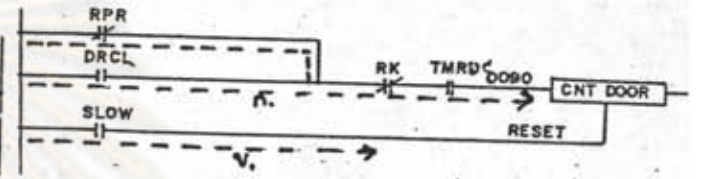
TIMER TMRDC จะเริ่มนับเมื่อประตูเริ่มปิด ตามลูกศร ก. รูปที่ 6.28 ประตูเปิด LA = 0 คำสั่งประตูปิด DRCL = 1 DZ = 1 อยู่ในระยะ Door Zone ลิฟท์จอดอยู่ ถ้าประตูเกิดผิดปกติไม่สามารถปิดได้สนิทในระยะเวลาที่ตั้งไว้ เมื่อ TMRDC ON จะไปทำให้ RE-OPEN REOP = 1 และอีก



รูปที่ 6.27

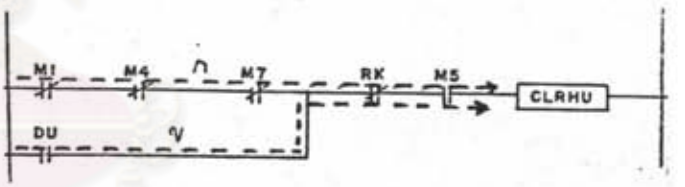


รูปที่ 6.28

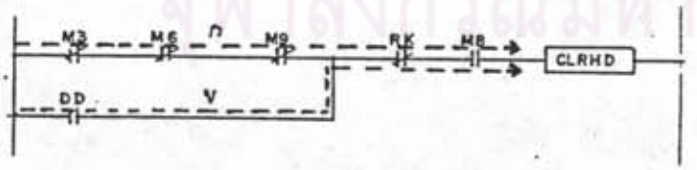


รูปที่ 6.29

รูปที่ 6.30



รูปที่ 6.31



รูปที่ 6.32

กรณีเมื่อประตูปิดสนิทแต่สวิทช์ประตูไม่ต่อ RPP เป็น 0 อยู่ ลูกศร ข. TIMER ก็จะเริ่มนับ เมื่อครบเวลาก็จะ ON ไปทำให้ REOP = 1

CNT DOOR COUNTER DOOR ใช้นับจำนวนครั้งที่ประตูปิด-เปิด ถ้าลิฟท์ไม่สามารถวิ่งได้ในเวลาที่กำหนด รีเลย์ CNT DOOR จะยกเลิกคำสั่งทั้งหมดให้ประตูเปิดค้างไว้ ตามลูกศร ก. รูปที่ 6.29 เมื่อประตูเปิด RPP = 0 RK = 0 เพราะลิฟท์ไม่วิ่ง เมื่อประตูทำงานผิดปกติ TMRDC = 1 จะเกิดการนับ 1 ครั้ง และในเวลาเดียวกัน TMRDC จะสั่งให้ REOP = 1 เมื่อมีคำสั่งประตูปิดก็จะเริ่มนับใหม่อีกครั้ง เป็นเช่นนี้ไปเรื่อยๆ จนกว่าจะนับครบตามจำนวนครั้งที่ถูกกำหนดไว้ (DRCL ที่ต่อขนาน RPR เพื่อช่วยซึ่งกันและกัน) สำหรับลูกศร ข. ใช้เพื่อ RESET การนับเดิมที่นับไว้ทั้งเมื่อลิฟท์สามารถวิ่งได้ และเข้า SLOW จะ RESET CNT.DOOR

LAMP CONTROL

CLEAR CAR CALL

ถ้าลิฟท์หยุด RK = 0 และมี CAR CALL เท่ากับชั้นก็จะ CLEAR สัญญาณ CAR CALL ตามรูปที่ 6.30

CLEAR HALL UP

1) ถ้าลิฟท์วิ่งขึ้นและทิศทางยังขึ้นอยู่ด้วย ถ้ามี CAR CALL ต่ำกว่าชั้น M1 = 1 มี HALL UP ต่ำกว่าชั้น M4 = 1 และมี HALL DOWN ต่ำกว่าชั้น M7 = 1 คำสั่ง CLEAR HALL UP จะผ่านทางลูกศร ข. ทาง DU สัญญาณทิศทางขึ้นยังอยู่ DU = 1 RK = 0 ลิฟท์อยุด และมี HALL UP เท่ากับชั้น จะ CLEAR สัญญาณชั้นนั้น รูปที่ 6.31

2) ถ้าลิฟท์วิ่งลง DU = 0 มีสัญญาณ HALL UP อยู่ แต่ไม่มี CAR CALL ต่ำกว่าชั้น M1 = 0 ไม่มี HALL UP ต่ำกว่าชั้น M4 = 0 ไม่มี HALL DOWN ต่ำกว่าชั้น M7 = 0 คำสั่ง CLEAR สัญญาณจะผ่านทางลูกศร ก. รูปที่ 6.31

CLEAR HALL DOWN

1) ถ้าลิฟท์วิ่งลงมี HALL DOWN เท่ากับชั้น จะผ่านทางลูกศร ข. รูปที่ 6.32 DD = 1 ลิฟท์มสัญญาณลงอยู่ DD = 1 ลิฟท์จอดอยู่ RK = 0 M8 = 1 เพราะมี HALL DOWN เท่ากับชั้น จะมีคำสั่ง CLEAR สัญญาณชั้นนั้น

2) ถ้าลิฟท์วิ่งขึ้นมี HALL DOWN เท่ากับชั้น จะ CLEAR ผ่านทางลูกศร ก. รูปที่ 6.10 ค) ลิฟท์วิ่งขึ้น DD = 0 ไม่มี CAR CALL สูงกว่าชั้น M3 = 0 ไม่มี HALL UP สูงกว่าชั้น M6 = 0 และไม่มี HALL DOWN สูงกว่าชั้น M9 = 0

เมื่อวงจรแผนภาพขั้นบันไดทำงานเสร็จสิ้นลง ไมโครคอมพิวเตอร์ก็จะทำการส่งเอาท์พุทออกไปควบคุมการทำงานของวงจรรีเลย์รับคำสั่ง ซึ่งวงจรรีเลย์รับคำสั่งไปควบคุมคอนแทกเตอร์ขับลิฟท์