ชุดจำลองเครื่องควบคุมจานรับสัญญาณดาวเทียมค้างฟ้า

DIRECTION CONTROL TO RECEIVE GEOSTATIONARY SATELLITE

โดย

นายบัณฑิต จิตประเสริฐ

นายพิพัฒน์ อรรถเกตุถาวร

นายยุทธนา วิเศษวงษา

ปริญญานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาอุตสาหกรรมศาสตร์บัณฑิต

สาขาเทคโนโลยีโทรคมนาคม

คณะวิศวกรรมศาสตร์

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

ปีการศึกษา 2540

หัวข้อปริญญานิพนธ์	**ชุดจำลองเครื่องควบคุมจานรับสัญญาณดาวเทียมค้างฟ้า**
	DIRECTION CONTROL TO RECEIVE
	GEOSTATIONARY SATELLITE
ชื่อนักศึกษา	นายบัณฑิต จิตประเสริฐ
	นายพิพัฒน์ อรรถเกตุถาวร
	นายยุทธนา วิเศษวงษา
อาจารย์ที่ปรึกษา	อ. ประดิษฐ์ วัชรพิบูลย์
ภาควิชา	เทคนิคอุตสาหกรรม
ปีการศึกษา	2540

คณะวิศวกรรมศาสตร์ สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบังอนุมัติให้นับปริญญานิพนธ์ฉบับนี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรอุตสาหกรรมศาสตร์บัณฑิต

คณะกรรมการสอบปริญญานิพนธ์

---------- ประธานกรรมการ
( )
---------- กรรมการ
( )
---------- กรรมการ
( )
---------- กรรมการ
( )
---------- กรรมการ
( )
---------- กรรมการ
( )

ลิขสิทธิ์ของคณะวิศวกรรมศาสตร์ สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

ชุดจำลองเครื่องควบคุมจานรับสัญญาณดาวเทียมค้างฟ้า

โดย	นายบัณฑิต จิตประเสริฐ	รหัส	38013363
	นายพิพัฒน์ อรรถเกตุถาวร	รหัส	38013372
	นายยุทธนา วิเศษวงษา	รหัส	38013376

อาจารย์ที่ปรึกษา อ. ประดิษฐ์ วัชรพิบูลย์

ปีการศึกษา 2540

บทคัดย่อ

ในโครงงานนี้ได้นำเอาสเต็ปปิ้งมอเตอร์มาใช้งานในด้านการเปลี่ยนทิศทางในแนวแกนนอนและแนวแกนตั้ง หรือการเปลี่ยนแปลงในแนวมุมกวาดและในแนวมุมเงย เพื่อใช้สำหรับควบคุมทิศทางของจานรับสัญญาณจากดาวเทียมค้างฟ้า ให้อยู่ในตำแหน่งที่ต้องการ โดยในที่นี้ขนาดของมุมกวาดจะเป็นไปในลักษณะของการหมุนได้โดยรอบตัวซึ่งจะมีมุมขนาด 360 องศา ส่วนมุมที่อยู่ในลักษณะของมุมเงยนั้นจะมีขนาดเพียง 90 องศา โครงงานนี้จะสามารถแบ่งออกตามหน้าที่การ ทำงานได้เป็นสองส่วน คือ ส่วนที่เป็น ฮาร์ดแวร์ (HARDWARE) ซึ่งประกอบไปด้วยส่วนที่เป็นเมคคานิค ซึ่งได้แก่ระบบการหมุนของจานดาวเทียมโดยในส่วนนี้จะใช้สเต็ปปิ้งมอเตอร์เป็นตัวควบคุมในการเคลื่อนที่, วงจรขับสเต็ปปิ้งมอเตอร์(STEPPING - DRIVER) , บอร์ดไมโครคอนโทรลเลอร์ (CONTROLER BOARD) และส่วนที่เป็นซอฟต์แวร์ (SOFT WARE) ซึ่งเป็นส่วนของโปรแกรมที่ใช้สำหรับสั่งงานหรือควบคุมการหมุนของมอเตอร์ให้ไปยังตำแหน่งที่ต้องการ โดยในที่นี้เราจะใช้ภาษาแอสเซมบลี้ที่ใช้กับไมโครคอนโทรลเลอร์ MCS - 51

DIRECTION CONTROL TO RECEIVE GEOSTATIONARY SATELLITE

BY	**MR.BUNDIT JITPRASIRT**	**CODE 38013363**
	MR.PIPUT AUTHAGETTHAVON	**CODE 38013372**
	MR.YUTTANA VISEDWONGSA	**CODE 38013376**

ADVISER **MR.PRADIT WATTCHARAPIBOOL**

YEAR **1997**

ABSTRACT

WE HAVE APPLIED STEPPING MOTOR FOR CHANGES IN DIRECTION HORIZONTALLY AND VERTICALLY IN THIS PROJECT AS WELL AS AZIMUTH ANGLE AND ELEVATION ANGLE. THIS PROJECT IS DESIGNED FOR CONTROLLING THE DIRECTIONS OF SATELLITE ANTENNA TO DESIRABLE DIRECTION AND AZIMUTH ANGLE CAN BE ROTATED UP TO 360 DEGREE. IN THE OTHER HAND ELEVATION ANGLE CAN BE MERELY ROTATED UP TO 90 DEGREE. THIS PROJECT CAN BE DEVIDED INTO 2 PARTS ACCORDING TO ITS FUNCTIONS :

1. HARDWARE COMPOSES OF THREE DIFFERENT MECHANICS WHICH ARE (A) SATELLITE ANTENNA ROTATING SYSTEM CONTROLLED BY STEPPING MOTOR, (B) STEPPING DRIVER, (C) CONTROLLER BOARD AND

2. SOFTWARE IS DESIGNED THE PROGRAM WHICH IS SERVED FOR ORDERING OR CONTROLLING THE ROTATING OF MOTOR TO DESIRABLE DIRECTION AND THE ASSEMBLY LANGUAGE WHICH IS USED FOR MICROCONTROLLER MCS - 51

กิตติกรรมประกาศ

ปริญญานิพนธ์ฉบับนี้สำเร็จลุล่วงไปได้ด้วยความช่วยเหลือของ อ.ประดิษฐ์ วัชรพิบูลย์ อาจารย์ที่ปรึกษา โดยท่านได้ให้คำแนะนำ ข้อคิดเห็นต่างๆ รวมทั้งปัญหาที่มักจะเกิดขึ้นกับโครงงานในลักษณะนี้จากประสบการณ์ของท่าน รวมทั้งเพื่อนร่วมสถาบันที่ให้การช่วยเหลือในการทำงาน ตลอดจนหน่วยงานต้นสังกัดที่ให้โอกาสและสนับสนุนการศึกษามาอย่างสม่ำเสมอ

ท้ายนี้ ผู้จัดทำขอกราบขอบพระคุณ บิดา - มารดา และคณะอาจารย์ทุกท่านที่คอยให้กำลังใจและสนับสนุนมาโดยตลอดจนสำเร็จการศึกษา

นายบัณฑิต จิตประเสริฐ
นายพิพัฒน์ อรรถเกตุถาวร
นายยุทธนา วิเศษวงษา

สารบัญ

เรื่อง	หน้า
บทคัดย่อภาษาไทย	
บทคัดย่อภาษาอังกฤษ	
สารบัญ	
สารบัญภาพ	
สารบัญตาราง	
บทที่ 1 บทนำ	1
บทที่ 2 ทฤษฎีและหลักการ	4
2.1 สเต็ปปิ้งมอเตอร์	4
2.1.1 ชนิดของสเต็ปปิ้งมอเตอร์	4
2.1.2โหมดการทำงานของสเต็ปปิ้งมอเตอร์	11
2.1.3กราฟแสดงคุณลักษณะของสเต็ปปิ้งมอเตอร์	13
2.1.4วิธีการกระตุ้นเฟส	14
2.2 การเขียนโปรแกรมด้วยภาษาแอสเซมบลี้	18
2.2.1รูปแบบภาษาแอสเซมบลี้	18
2.2.2คำสั่งเทียม	20
2.2.3หน่วยความจำและรีจีสเตอร์อ้างอิง	24
2.2.4โครงสร้างของโปรแกรม	24
2.2.5การสิ้นสุดของเอ็กซีคิวส์โปรแกรม	26
2.2.6คำสั่งอินพุตเอาท์พุต	31
2.2.7โปรแกรมแสดงการทำงานของการบริการ	33
2.3ไมโครคอนโทรลเลอร์	36
2.3.1ไมโครคอนโทรลเลอร์ตระกูล 8051	36
2.3.2สถาปัตยกรรมของ 8051	36
2.3.3หน่วยความจำของ 8051	38
2.3.4รีจีสเตอร์หน้าที่พิเศษ	43
2.3.5หน่วยความจำข้อมูลภายนอก	45

2.3.6พอร์ตอินพุต / เอาท์พุตของ 8051 46
2.3.7ลักษณะสมบัติของพอร์ตอินพุต / เอาท์พุต 48
2.3.8การใช้งาน 8255 50
2.3.9เคาน์เตอร์และไทม์เมอร์ 51
2.3.10อินพุตเอาท์พุตข้อมูลอนุกรม 53
2.3.11ข้อมูลอนุกรม 1 - UART มาตรฐาน 55
2.3.12อินเตอร์รัพท์ 57
2.3.13การควบคุมการอินเตอร์รัพท์ 60
2.4วิธีติดตั้งจานรับสัญญาณดาวเทียม 62
2.4.1การสำรวจพื้นที่ 62
2.4.2มุมกวาดและมุมเงย 62
2.4.3การปรับตั้งเมาท์ 65
2.4.4การติดตั้งอุปกรณ์ LNB และ FEEDHORN 66
2.4.5การปรับแต่งโพลาไรเซชั่น 68
2.4.6การเดินสายเคเบิล 68
2.4.7การต่อและขั้วต่อโคแอกเชียล 69
2.4.8การเดินสายสำหรับฟีดฮอร์น 71
2.4.9การเดินสายสำหรับมอเตอร์ขับจานการเดินสายดิน 73
2.4.10การติดตั้งมอเตอร์ขับจานเข้ากับจาน 74
2.4.11การปรับแต่งจานรับสัญญาณ 74
บทที่ 3 การออกแบบและการสร้าง 78
3.1 การออกแบบและการสร้างส่วนประกอบทางด้านฮาร์ดแวร์ 79
3.1.1 วงจรจ่ายกำลังงาน 79
3.1.2วงจ่รขับสเต็ปปิ้งมอเตอร์ 80
3.1.3บอร์ดไมโครคอนโทรลเลอร์ 82
3.1.4ชุดเมคคานิค 87
3.2การออกแบบส่วนประกอบทางด้านซอฟต์แวร์ 91
บทที่ 4 การทดลองและผลการทดลอง 92
1.วงจรจ่ายกำลังงาน 92
2.วงจรขับสเต็ปปิ้งมอเตอร์ 92

3.บอร์ดไมโครคอนโทรลเลอร์ 92
4.ชุดเมคคานิค 93
บทที่ 5 บทสรุป 94
ภาคผนวก
เอกสารอ้างอิง

สารบัญภาพ

เรื่อง	หน้า
รูปที่ 1 BLOCKDIAGRAM แสดงระบบการทำงาน	2
รูปที่ 2.1 ก) ภาพหน้าตัดของ PM มอเตอร์ แบบ 4 เฟส ข) วงจรกระตุ้นเฟสพื้นฐานสำหรับ PM มอเตอร์ 4 เฟส	5
รูปที่ 2.2 แสดงโครงสร้างพื้นฐานของแรเอิร์ธเพอร์มาเนนต์แมกเน็ต	5
รูปที่ 2.3 แสดงกราฟข้อมูลการสูญเสียกำลังงานไฟฟ้าและความเร็วในการหมุนโดยเปรียบเทียบระหว่างสเต็ปปิ้งมอเตอร์ชนิดไฮบริดและชนิดแรเอิร์ธเพอร์มาเนนต์แมกเน็ต	6
รูปที่ 2.4 แสดงการพันขดลวดของสเตเตอร์	7
รูปที่ 2.5 แสดงการจ่ายไฟฟ้าให้กับสเต็ปปิ้งมอเตอร์	8
รูปที่ 2.6 ภาพหน้าตัดและการพันขดลวดของวาริเอเบิลรีลักแตนซ์สเต็ปปิ้งมอเตอร์ 3 เฟส	8
รูปที่ 2.7 แสดงเส้นแรงแม่เหล็กและกระตุ้นเฟส 1	9
รูปที่ 2.8 แสดงขั้นตอนการหมุนของ VR สเต็ปปิ้งมอเตอร์เมื่อมีการกระตุ้นเฟส 1 ไปยังเฟส 2	10
รูปที่ 2.9 ก) ภาพหน้าตัดของ PM มอเตอร์แบบ 4 เฟส ข) วงจรกระตุ้นเฟสพื้นฐานของ PM มอเตอร์ 4 เฟส	10
รูปที่ 2.10 ลำดับขั้นตอนการหมุน 4 เฟส	11
รูปที่ 2.11 แสดงการหมุนในโหมดการทำงาน	12
รูปที่ 2.12 กราฟแสดงความสัมพันธ์ระหว่างอัตราเร็วของสเต็ปกับแรงบิดของการทำงานทั้ง 2 โหมด	13
รูปที่ 2.13 ตารางการกระตุ้นเฟส	14
รูปที่ 2.14 ก) แสดงเส้นทางเดินของเส้นทางแม่เหล็ก ในการกระตุ้นแบบเฟสคู่ ข) แสดงการเข้าตำแหน่งของโรเตอร์	15
รูปที่ 2.15 กราฟแสดงคุณลักษณะโมเมนต์กับความถี่	16
รูปที่ 2.16 แสดงถึง SKELETON OF AN EXE PROGRAM	25

รูปที่ 2.17 แสดง EXE SOURCE PROGRAM WITH CONVENTIONAL SEGMENT 29
รูปที่ 2.18 แสดง EXE SOURCE PROGRAM WITH SIMPLIFIED SEGMENT DIRECTIVES 31
รูปที่ 2.19 แสดงการกำหนดขาของ 8051 37
รูปที่ 2.20 แสดงภาพสัญญาณเวลาแสดงการติดต่อกับหน่วยความจำภายนอก 40
รูปที่ 2.21 การเชื่อมต่อเพื่อขยายหน่วยความจำโปรแกรมด้วย ไอ.ซี.อีพรอมหลายตัว 41
รูปที่ 2.22 โครงสร้างแต่ละบิตภายในพอร์ตอินพุตเอาท์พุตของ 8051 47
รูปที่ 2.23 พิกัดของมุมกวาดและมุมเงยของจานรับสัญญาณที่ติดตั้งอยู่ในกรุงเทพ 63
รูปที่ 2.24 อุปกรณ์วัดมุมเอียง ANGLE FINDER 64
รูปที่ 2.25 มิเตอร์วัดความเข้มของสัญญาณดาวเทียมซึ่งสามารถแสดงผลด้วยเข็มและเสียง 65
รูปที่ 2.26 สายเคเบิลแบบโคแอคเชียล 69
รูปที่ 2.27 ขั้วต่อแบบ F - TYPE 70
รูปที่ 2.28 การเดินสายเชื่อมต่อระหว่างอุปกรณ์ของเครื่องรับสัญญาณแบบ LNB คู่ 72
รูปที่ 3.1 ภาพถ่ายชุดจำลองเครื่องควบคุมจานรับสัญญาณดาวเทียมค้างฟ้า 78
รูปที่ 3.2แสดงรูปวงจรจ่ายกำลังงาน 80
รูปที่ 3.3 แสดงลายวงจรพิมพ์ของวงจรจ่ายกำลังงาน 80
รูปที่ 3.4 แสดงรูปวง่จรขับสเต็ปปิ้งมอเตอร์ 82
รูปที่ 3.5 แสดงลายวงจรพิมพ์ของชุดขับสเต็ปปิ้งมอเตอร์ 82
รูปที่ 3.6 แสดงภาพถ่ายส่วนต่างๆ ประกอบด้วย บอร์ดคอนโทรลเลอร์, วงจรขับสเต็ปปิ้งมอเตอร์, วงจรจ่ายกำลังงาน และชุด LCD MODULE 85
รูปที่ 3.7 แสดงวงจรของบอร์ดคอนโทรลเลอร์ 86
รูปที่ 3.8 ภาพถ่ายแสดงส่วนประกอบชุดควบคุมทางด้านมุมกวาด 89

รูปที่ 3.9 ภาพถ่ายแสดงส่วนประกอบชุดควบคุมแนวมุมเงย 90
รูปที่ 3.10 โฟลว์ชาร์ตแสดงการทำงานของโปรแกรมควบคุม 91

สารบัญตาราง

เรื่อง		หน้า
ตารางที่ 2.1	ชื่อเรียกและความหมายทางกลและทางไฟฟ้าของสเต็ปปิ้งมอเตอร์	17
ตารางที่ 2.2	สัญญาณของ 8051 ที่ใช้ระหว่างการติดต่อเพื่ออ่านข้อมูลจากหน่วยความจำโปรแกรมภายนอก	39
ตารางที่ 3.1	แสดงแอดเดรสของอินพุตเอาท์พุตของ 8255	84
ตารางที่ 3.2	แสดงการควบคุมโหมดการทำงานของ 8255	85

บทที่ 1
บทนำ

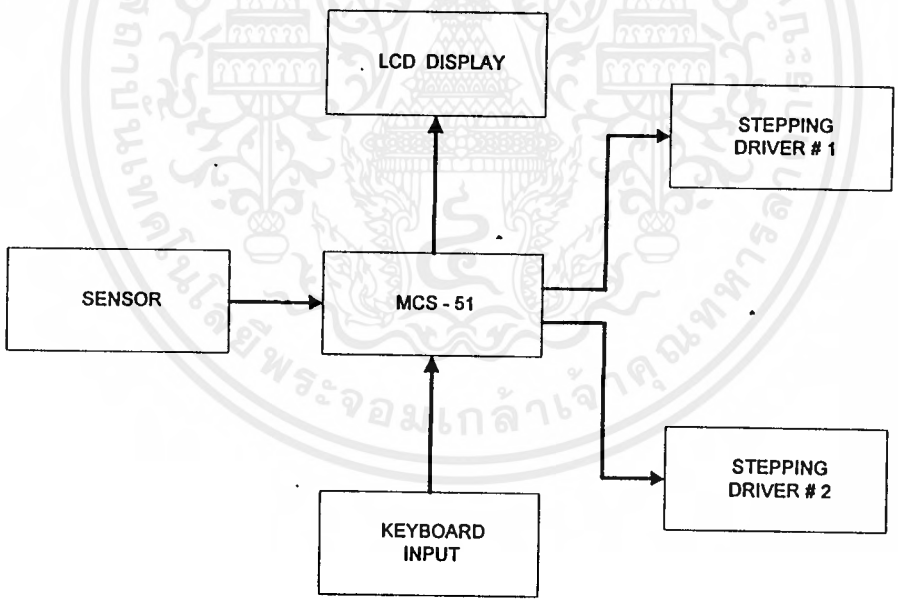
เทคโนโลยีการสื่อสารผ่านดาวเทียมนั้น ในอดีตจะมีการใช้ประโยชน์กันในวงแคบๆ เช่น การสำรวจทรัพยากรธรณี การสำรวจสภาพภูมิอากาศ และเทคโนโลยีทางการทหาร แต่หลังจากนั้นได้มีการพัฒนาเทคโนโลยีดาวเทียมมาใช้ในการสื่อสารมากขึ้น โดยได้นำมาใช้ในระบบโทรศัพท์ ระบบการรับ - ส่งภาพและเสียง รวมทั้งการส่งระบบข้อมูล (DATA) ซึ่งเป็นระบบการสื่อสารที่มีประสิทธิภาพสูง แต่ทั้งนี้ในระบบดังกล่าวจะมีปัญหาอุปสรรคอยู่หลายประการที่จะทำให้ประสิทธิภาพของการรับส่งลดลงจนไม่สามารถที่จะรับ - ส่งได้เลย ซึ่งปัญหาดังกล่าวนั้นจะประกอบไปด้วย กำลังส่งสัญญาณของสถานีภาคพื้นดินหรือของดาวเทียมมีขนาดต่ำเกินไป สถานที่ในการติดตั้งจานรับสัญญาณอยู่ในสภาพที่ไม่เหมาะสม ระยะทางระหว่างดาวเทียมกับสถานที่ที่รับสัญญาณอยู่ห่างกันจนสัญญาณเกิดการสูญเสียจนหมด เป็นต้น นอกจากปัญหาข้างต้นดังกล่าวแล้ว ยังมีองค์ประกอบที่สำคัญอีกอย่างหนึ่งที่มีผลต่อการรับส่งในระบบก็คือความผิดพลาดของมุมของจานรับดาวเทียมซึ่งเกิดจากการติดตั้ง ในโครงงานนี้จะเป็นการนำเสนอการนำสเต็ปปิ้งมอเตอร์ 2 ตัว มาควบคุมทิศทางการหมุนของจานรับสัญญาณดาวเทียม ซึ่งจะต้องหมุนใน 2 ลักษณะ คือ การหมุนในแนวกวาดรอบตัวเองหรือมุมอะซิมุธ (AZIMUTH) และควบคุมด้านมุมเงย หรือ มุมเอลิเวชั่น (ELEVATION) ซึ่งจะทำให้จานรับนี้สามารถหันไปชี้ได้ทุกตำแหน่งเหนือพื้นดินตามที่ต้องการซึ่งจะทำให้สามารถนำชุดควบคุมนี้ไปติดตั้งได้ทุกพื้นที่ในทั่วโลกโดยสามารถควบคุมให้หมุนไปชี้ในตำแหน่งที่ต้องการได้

การที่ในโครงงานนี้ใช้มอเตอร์ 2 ตัว จะมีประโยชน์ที่สามารถควบคุมการหมุนตามต้องการได้เป็นอย่างดี การหมุนแต่ละแนวจะใช้พลังงานที่แตกต่างกัน ดังเช่นในส่วนของมุมเงยซึ่งไม่ต้องการพลังงานในการควบคุมมากนักจึงสามารถใช้มอเตอร์ขนาดเล็กได้ ซึ่งจะเป็นการประหยัดงบประมาณและพลังงานในการขับเคลื่อน นอกจากนี้ การออกแบบ การสร้าง การปรับปรุงหรือเปลี่ยนแปลง สามารถทำได้ง่ายและสะดวกกว่าการใช้มอเตอร์เพียงตัวเดียว

ในส่วนของชุดควบคุมโครงงานนี้ได้ใช้ชุดควบคุมโดยเขียนเป็นภาษาแอสเซมบลี้ของไมโครคอนโทรลเลอร์ MCS - 51 ซึ่งเป็นตระกูลของไมโครคอนโทรลเลอร์ที่มีชุดคำสั่งที่สั้นกระชับ ทำความเข้าใจได้ง่าย รวมทั้งสะดวกในการปรับปรุงแก้ไข และอุปกรณ์ประกอบในส่วนของบอร์ดคอนโทรลหาได้ง่ายเพราะเป็นที่นิยมใช้งานกันอย่างกว้างขวาง โดยในส่วนของไมโครคอนโทรลเลอร์นี้ จะมีส่วนของหน่วยความจำที่สามารถเก็บโปรแกรมที่เป็นข้อมูลแสดงตำแหน่ง

ของดาวเทียมที่กำหนดไว้ โดยในที่นี้จะทำการบรรจุตำแหน่งของดาวเทียมจำนวน 9 ดวง หน่วยความจำนี้จะต้องเป็นหน่วยความจำที่สามารถโปรแกรมข้อมูลเข้าไปใหม่ได้ (EPROM) เพราะเมื่อเราเคลื่อนย้ายชุดควบคุมนี้ไปติดตั้งในที่ใหม่ จะต้องทำการโปรแกรมตำแหน่งใหม่ในการชี้ไปยังดาวเทียม

หลักการทำงานของชุดควบคุมนี้จะเริ่มต้นโดยกำหนดให้มุมทางด้านมุมกวาดของจานรับสัญญาณดาวเทียมไปอยู่ที่ตำแหน่งเริ่มต้น คือ 0 องศา และมุมทางด้านมุมเงยไปอยู่ที่ 90 องศา จากนั้นเมื่อผู้ใช้ต้องการจะรับสัญญาณจากดาวเทียมใด ก็ให้กดตำแหน่งของดาวเทียมนั้น สเต็ปปิ้งมอเตอร์ทั้งสองตัวจะหมุนไปอยู่ในตำแหน่งที่ต้องการ โดยที่ชุดควบคุมจะมี LCD MODULE ทำหน้าที่ในการแสดงผลหมายเลขและตำแหน่งของดาวเทียมที่กำลังรับสัญญาณอยู่ จากนั้นเมื่อต้องการเปลี่ยนตำแหน่งไปรับสัญญาณจากดาวเทียมดวงใหม่ ให้กดตำแหน่งใหม่เข้าไป สเต็ปปิ้งมอเตอร์ทั้งสองตัวจะไปในตำแหน่งที่ต้องการได้ทันที โดยที่ LCD MODULE จะแสดงหมายเลขและตำแหน่งของดาวเทียมดวงนั้นๆ



รูปที่ 1 BLOCKDIAGRAM แสดงระบบการทำงาน

หน้าที่การทำงานของแต่ละส่วนใน BLOCKDIAGRAM มีดังต่อไปนี้

1. MCS - 51 เป็นชุดไมโครคอนโทรลเลอร์ที่ทำหน้าที่ควบคุมการทำงานของระบบ โดยเป็นตัวรับสัญญาณจากคีย์บอร์ด และ เซ็นเซอร์ เพื่อไปควบคุมชุดขับสเต็ปปิ้งมอเตอร์ (STEPPING DRIVER) และควบคุม LCD MODULE
2. คีย์บอร์ด (KEYBOARD) เป็นจุดที่ผู้ใช้งานทำการป้อนตำแหน่งของดาวเทียมเพื่อส่งเข้าไปยังไมโครคอนโทรลเลอร์
3. ชุดขับสเต็ปปิ้งมอเตอร์ (STEPPING DRIVER) เป็นชุดที่ทำหน้าที่ขับสเต็ปปิ้งมอเตอร์ให้หมุนไปตามการสั่งงานของไมโครคอนโทรลเลอร์
4. เซ็นเซอร์ (SENSOR) เป็นตัวตรวจสอบตำแหน่งเริ่มต้นของจานรับ ซึ่งในที่นี้กำหนดไว้ที่ 0 องศา สำหรับมุมกวาด และ 90 องศา สำหรับมุมเงย
5. LCD MODULE ทำหน้าที่แสดงผลหมายเลขและตำแหน่งของดาวเทียมที่กำลังรับสัญญาณ

บทที่ 2
ทฤษฎีและหลักการ

2.1 สเต็ปปิ้งมอเตอร์ (STEPPING MOTER)

สเต็ปปิ้งมอเตอร์เป็นมอเตอร์ชนิดหนึ่งที่ทำหน้าที่เปลี่ยนสัญญาณดิจิตอลทางไฟฟ้าไปเป็นการเคลื่อนที่ทางกล ดังนั้นการติดต่อกับอุปกรณ์ดิจิตอลเป็นไปโดยง่าย และวงจรขยายกำลังจากสัญญาณดิจิตอล (DIGITAL POWER AMPLIFIER) ที่ใช้ก็มีราคาถูกกว่าวงจรขยายกำลังเชิงเส้นอีกด้วย อีกทั้งการออกแบบวงจรควบคุมสเต็ปปิ้งมอเตอร์สามารถทำได้ง่ายกว่าวงจรควบคุมมอเตอร์แบบเซอร์โว และยังสามารถออกแบบวงจรสเต็ปปิ้งมอเตอร์หยุดการทำงานได้อย่างทันทีทันใดอีกด้วย

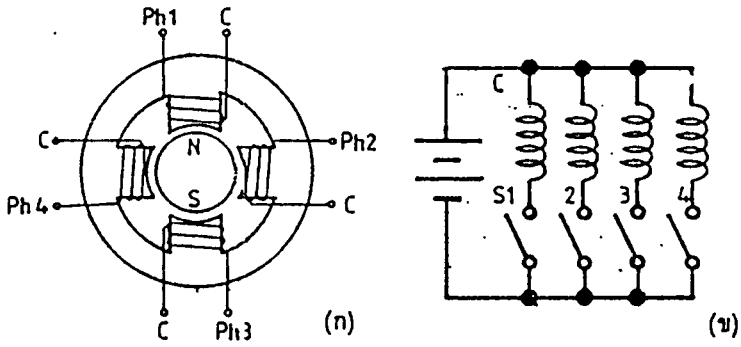
2.1.1 ชนิดของสเต็ปปิ้งมอเตอร์

สเต็ปปิ้งมอเตอร์แบ่งตามมาตรฐานได้ 3 แบบ คือ

1. วาริเอเบิลรีลักแตนซ์ (VARIABLE RELUCTANCE : VR)
2. เพอร์มาเนนต์แมกเน็ต (PERMANENT MAGNET : PM)
3. ไฮบริด (HYBRID)

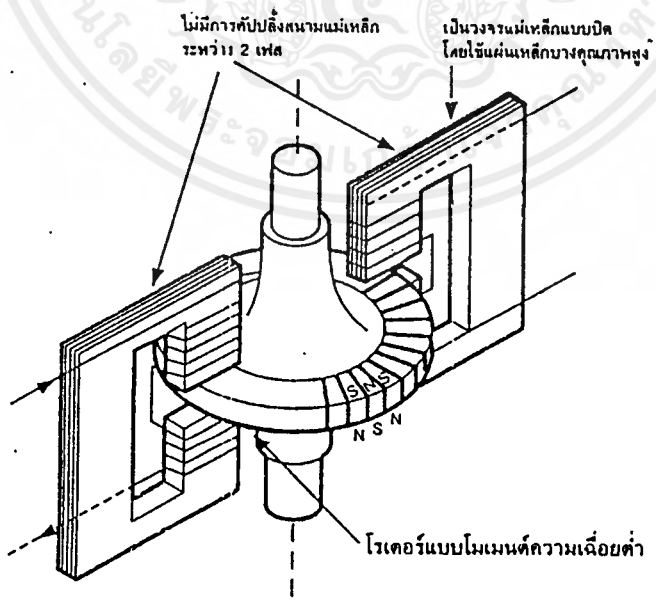
ชนิดวาริเอเบิลรีลักแตนซ์มีโครงสร้างของโรเตอร์แบบมัลติทูธ (MULTI TOOTH) ทำจากเหล็กอ่อนเราทราบได้ว่าเป็นมอเตอร์ชนิดนี้โดยการทดสอบได้ง่ายมากคือ ใช้นิ้วหมุนเพลาของมอเตอร์และสังเกตมอเตอร์ชนิดนี้ที่โรเตอร์ไม่เกิดปรากฏการณ์ทางแม่เหล็ก MAGMATISM มันจึงหมุนได้ตลอดโดยไม่ติดขัดแตกต่างจากชนิด PM และไฮบริดซึ่งมีสนามแม่เหล็กที่โรเตอร์เมื่อหมุนจะรู้สึกขัด ๆ เหมือนเป็นฟันเฟือง สเต็ปปิ้งมอเตอร์ชนิดนี้มีจุดด้อยในเรื่องของความถูกต้องของตำแหน่งและทำงานได้ไม่ดีนักเมื่อมีสเต็ปในการหมุนสูง

ชนิดเพอร์มาเนนต์แมกเน็ตมีโครงสร้างแบบเรียบไม่มีซี่ขั้วแม่เหล็กและบนโรเตอร์จะเป็นแบบแม่เหล็กถาวร การควบคุมทำได้โดยการป้อนกระแสกระตุ้นที่ขดลวดบนสเตเตอร์ แบบ 4 เฟส จะมีขั้วแม่เหล็กอยู่ 4 ขั้ว ซึ่งมีคอยล์พันอยู่แยกจากกัน ขั้วแม่เหล็กถาวรจะถูกแรงดึงดูดจากขั้วแม่เหล็กบนสเตเตอร์เมื่อป้อนกระแสไฟฟ้าเข้าสู่ขดลวด และโรเตอร์จะอยู่คงที่ที่ขั้วแม่เหล็กบนสเตเตอร์นั้นถึงแม้ว่าจะไม่ป้อนกระแสไฟฟ้าอีกต่อไป ทำให้เกิดแรงยึดหน่วงขึ้น สเต็ปปิ้งมอเตอร์ชนิดนี้มีข้อดีในเรื่องของความถูกต้องของตำแหน่ง และความเร็วมากขึ้นเมื่อเปรียบเทียบกับชนิดอื่น



รูปที่ 2.1 ก) ภาพหน้าตัดของ PM มอเตอร์แบบ 4 เฟส
ข) วงจรกระตุ้นเฟสพื้นฐานสำหรับ PM มอเตอร์ 4 เฟส

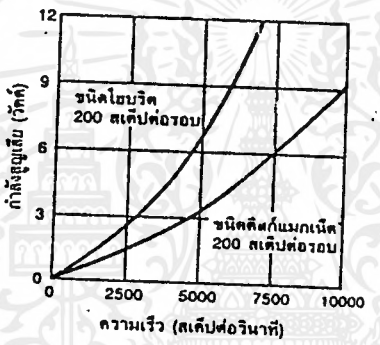
ชนิดไฮบริดเป็นชนิดที่นิยมใช้งานมากที่สุด โดยเฉพาะนำมาใช้งานอย่างมากในอุปกรณ์ที่ใช้ในเครื่องคอมพิวเตอร์ ชนิดมีโครงสร้างภายในซึ่งได้จาก การรวมเอาโครงสร้างของสเตเตอร์ชนิดวาริเอเบิลรีลักแตนซ์ และโครงสร้างของโรเตอร์จากเพอร์มาเนนต์แมกเน็ตมาประกอบเข้าด้วยกันจึงทำให้เป็นมอเตอร์ที่มีแรงยึดหน่วงสูง มีแรงบิดดึงและผลักได้ดีซึ่งมีความคงที่และทำงานได้ดีถึงแม้ว่าจะมีสเต็ปต่อรอบในการหมุนสูง



รูปที่ 2.2 แสดงโครงสร้างพื้นฐานของชนิดแรเอิร์ธเพอร์มาเนนต์แม่กเน็ต

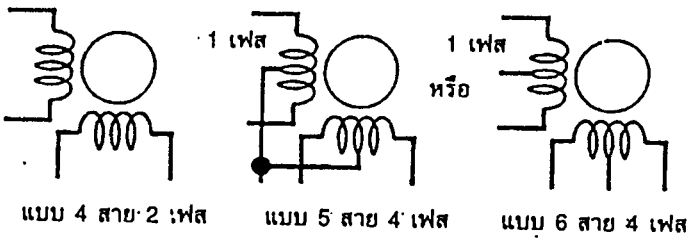
สเต็ปปิ้งมอเตอร์แบบใหม่อีกชนิดหนึ่งที่กล่าวถึงอีกเล็กน้อยคือ ชนิดที่ปรับปรุงมาจากชนิดเพอมาเนนต์แมกเน็ตนั่นคือ ชนิดแรเอิร์ธเพอร์มาเนนต์แม็กเน็ตดังแสดงโครงสร้างในรูปที่ 2.2 หรือที่เรียกกันว่าชนิดดิสค์แมกเน็ตสเต็ปปิ้งมอเตอร์

โครงสร้างของโรเตอร์ของมอเตอร์ชนิดนี้มีลักษณะเป็นแผ่นซึ่งยึดติดกับเพลาของมอเตอร์ การทำงานของมอเตอร์ยังคงเป็นเช่นเดิม แต่ด้วยโครงสร้างแบบใหม่นี้จะทำให้เกิดโมเมนต์ของความเฉื่อยต่ำมาก, มีอัตราเร่งสูง มอเตอร์ชนิดนี้จึงจัดเป็นอีกชนิดหนึ่ง และมันก็มีประสิทธิภาพสูงอีกหลายอย่างเช่น แรงบิดดึง, กำลังทางกลที่ได้ของมอเตอร์, ความถูกต้องของตำแหน่งสูงมาก และความเร็วในการเริ่มหมุนและหยุดสูง อีกทั้งยังมีความสูญเสียของกำลังงานต่ำ ดังแสดงในรูปที่ 2.3



รูปที่ 2.3 แสดงกราฟข้อมูลการสูญเสียกำลังงานไฟฟ้าและความเร็วในการหมุนโดยเปรียบเทียบระหว่างสเต็ปปิ้งมอเตอร์ชนิดไฮบริดและชนิดแรเอิร์ธเพอร์มาเนนต์แมกเน็ต

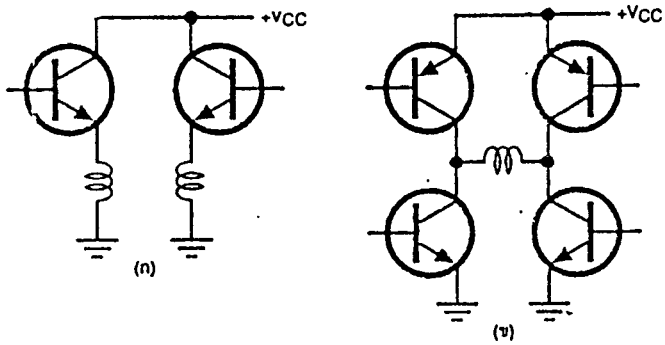
การพันขดลวดหรือคอยล์บนสเต็ปปิ้งมอเตอร์มีอยู่ 2 วิธีคือ แบบไบโพลาร์ (BIPOLAR) และแบบยูนิโพลาร์ (UNIPOLAR) ดังแสดงในรูปที่ 2.4



รูปที่ 2.4 แสดงการพันขดลวดของสเตเตอร์ด้านซ้ายเป็นแบบไบโพลาร์และที่เหลือเป็นแบบยูนิโพลาร์

สเต็ปปิ้งมอเตอร์แบบไบโพลาร์มีการพันขดลวด 1 ขดแต่ละขั้วแม่เหล็กของสเตเตอร์ ขั้วแม่เหล็กที่เกิดขึ้นบนสเตเตอร์ถูกกำหนดโดยทิศทางของกระแสไฟฟ้า และสามารถทำให้เกิดขั้วแม่เหล็กในทิศทางตรงข้ามได้โดยการกลับทิศทางการไหลของกระแสไฟฟ้า ซึ่งการกำหนดทิศทางไหลและกลับทิศทางของกระแสไฟฟ้าทำได้โดยการใช้วงจรสวิตช์ชิ่งกลับขั้วไฟฟ้า

สำหรับยูนิโพลาร์จะมีการพันขดลวด 2 ขดบนแต่ละขั้วแม่เหล็กของสเตเตอร์ ซึ่งแต่ละขดจะทำให้เกิดขั้วแม่เหล็กในทิศทางตรงข้ามกัน การกลับขั้วแม่เหล็กเปลี่ยนไปมาทำได้โดยการสวิตช์กระแสไฟฟ้าจากขดลวดขดหนึ่งไปยังอีกขดลวดหนึ่งแทนเท่านั้น โดยปกติขดลวดทั้งสองจะมีการเชื่อมต่อกันหรือมีจุดร่วมเพื่อลดจำนวนของสายไฟที่ต่อกับมอเตอร์ วงจรกำลังไฟฟ้าของมอเตอร์แบบยูนิโพลาร์ทำได้ง่ายกว่าไบโพลาร์ เพราะมันต้องการเพียงสวิตช์ธรรมดาในการเปิดปิดกำลังไฟฟ้าขดลวดบนสเตเตอร์ในทิศทางที่ต้องการให้หมุนได้ทันที รูปที่ 2.5 แสดงวงจรการจ่ายไฟฟ้าซึ่งทรานซิสเตอร์ทำหน้าที่เป็นสวิตชิ่งให้กับสเต็ปปิ้งมอเตอร์ที่มีการพันขดลวดทั้งสองแบบ จะเห็นว่าแบบยูนิโพลาร์เป็นวงจรที่ง่ายและไม่ซับซ้อน

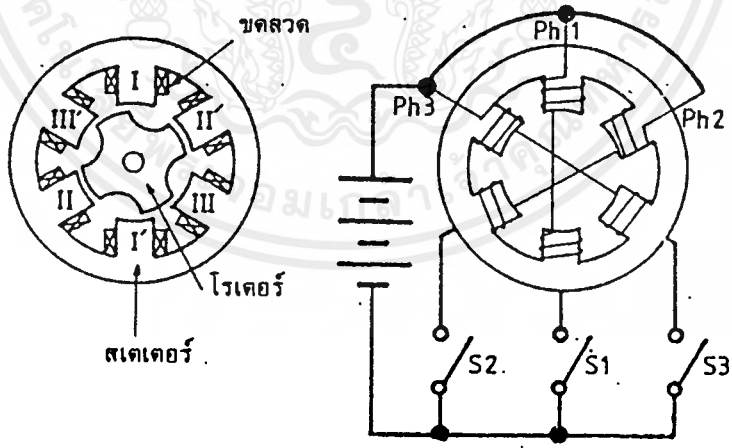


รูปที่ 2.5 แสดงการจ่ายไฟฟ้าให้กับสเต็ปปิ้งมอเตอร์

ก) แบบยูนิโพลาร์ซึ่งใช้ทรานซิสเตอร์เพียงตัวเดียวต่อ 1 คอยล์

ข) แบบไบโพลาร์ ซึ่งต้องใช้ทรานซิสเตอร์ 4 ตัวต่อ 1 คอยล์

การทำงานของสเต็ปปิ้งมอเตอร์ แบบ VR จะเป็นพื้นฐานสำคัญในการทำงานของสเต็ปปิ้งมอเตอร์ ซึ่งจะช่วยให้เข้าใจการทำงานของสเต็ปปิ้งมอเตอร์ชนิดอื่นๆ ได้ง่ายยิ่งขึ้น ดังในรูปจะเป็นภาพแสดงถึงการพันขดลวดแบบ VR มอเตอร์แบบ 3 เฟส มีขั้วเหนือและขั้วใต้อยู่ตรงข้ามกัน 3 คู่ โดยจะพันขดลวดแบบอนุกรมกันในแต่ละชุด ถ้ามีการกระตุ้นเฟสเกิดขึ้นขั้ว I' , II' , III' จะเป็นขั้วใต้ และ I , II , III จะเป็นขั้วเหนือ ทั้งโรเตอร์และสเตเตอร์จะทำจากเหล็กซิลิกอน ซึ่งเป็นวัสดุที่มีความซึมซับสูง สามารถให้เส้นแรงแม่เหล็กไหลผ่านได้มาก



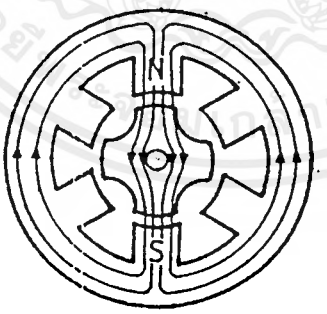
รูปที่ 2.6 ภาพหน้าตัดและการพันขดลวดของวาริเอเบิลรีลักแตนซ์สเต็ปปิ้งมอเตอร์ 3 เฟส

การทำงานจะมีการกระตุ้นที่เฟส I ก่อน (S "ON") ซึ่งจะทำให้เส้นแรงแม่เหล็กเกิดขึ้นดังรูป ตัวโรเตอร์จะพยายามวางตำแหน่งของตัวเองให้อยู่ในทิศทางที่ทำให้เกิดค่าความต้านทาน

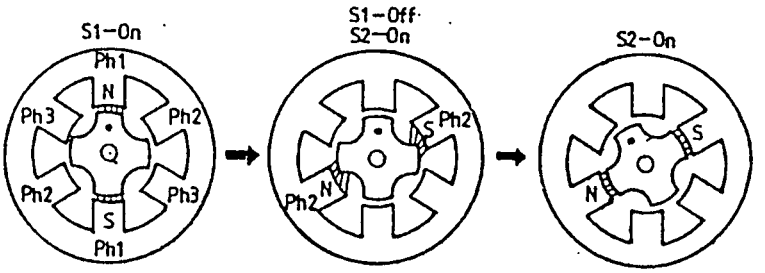
แม่เหล็กน้อยที่สุด ในขณะที่เริ่มต้นที่กระตุ้นที่เฟส II (S1 "OFF" , S2 "ON") เส้นแรงจะไม่อยู่ในแนวทางเดินที่สะดวก จึงทำให้ค่าความต้านทานแม่เหล็กมีค่าสูง ตัวโรเตอร์ก็พยายามปรับตัวเองให้มีค่าความต้านทานแม่เหล็กน้อยที่สุดด้วยการหมุนในทิศทางทวนเข็มนาฬิกาซึ่งแรงบิดที่ใช้หมุนเกิดจากแรงของเส้นแรงแม่เหล็ก แล้วจะไปหยุดในตำแหน่งที่ความต้านทานแม่เหล็กน้อยที่สุด นั่นคือจะหมุนไป 1 สเต็ป หรือ 30 องศานั่นเอง ความสัมพันธ์ระหว่างจำนวนสเต็ปของการหมุนของโรเตอร์ไป 1 รอบ (S) มุมที่เปลี่ยนไป 1 สเต็ป (θ_s) จำนวนเฟสของสเตเตอร์ (m) และจำนวนฟันของโรเตอร์ (Nr) ดังแสดงไว้ในสมการที่ 1

$$S = 360 / \theta_s = mNr \qquad (1)$$

ตัวอย่างเช่น สเต็ปปิ้งมอเตอร์ตัวหนึ่งมี m = 3 , Nr = 4 ก็จะได้ S = 3 * 4 = 12 สเต็ป และมุมในการหมุน θ_s = 360 / 12 = 30 องศา ซึ่งจากสมการที่ 1 ทำให้เราทราบอีกว่าถ้าจะลดค่าของ θ_s ให้น้อยลง อาจทำได้โดยการเพิ่มค่า m หรือค่า Nr ให้สูงขึ้น และลดช่องว่างระหว่างโรเตอร์กับสเตเตอร์ให้มีค่าน้อยๆ เพื่อให้เกิดแรงบิดสูงสุด และยังมีผลต่อความเที่ยงตรงของตำแหน่งมากยิ่งขึ้นด้วย

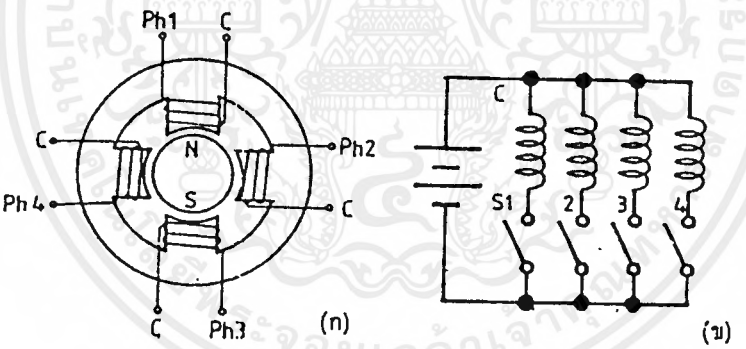


รูปที่ 2.7 แสดงเส้นแรงแม่เหล็กและกระตุ้นเฟส 1

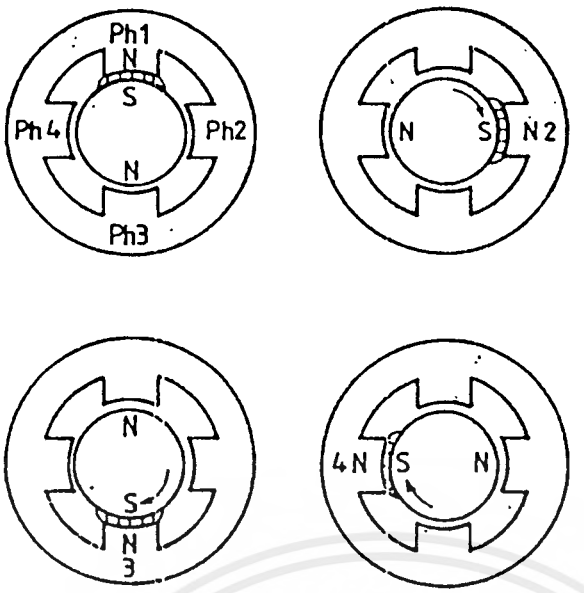


รูปที่ 2.8 แสดงขั้นตอนการหมุนของ VR สเต็ปปิ้งมอเตอร์เมื่อมีการกระตุ้นเฟส 1 ไปยังเฟส 2

สำหรับสเต็ปปิ้งมอเตอร์ชนิดเพอร์มาเนนต์แมกเน็ต หรือ PM จะมีข้อแตกต่างสำคัญจาก VR สเต็ปปิ้งมอเตอร์ คือ โรเตอร์จะเป็นแบบแม่เหล็กถาวรการพันขดลวดจึงจะต้องมีความแตกต่างกันออกไปซึ่งแสดงดังรูป



รูปที่ 2.9 ก) ภาพหน้าตัดของ PM มอเตอร์แบบ 4 เฟส
ข) วงจรกระตุ้นเฟสพื้นฐานของ PM มอเตอร์ 4 เฟส

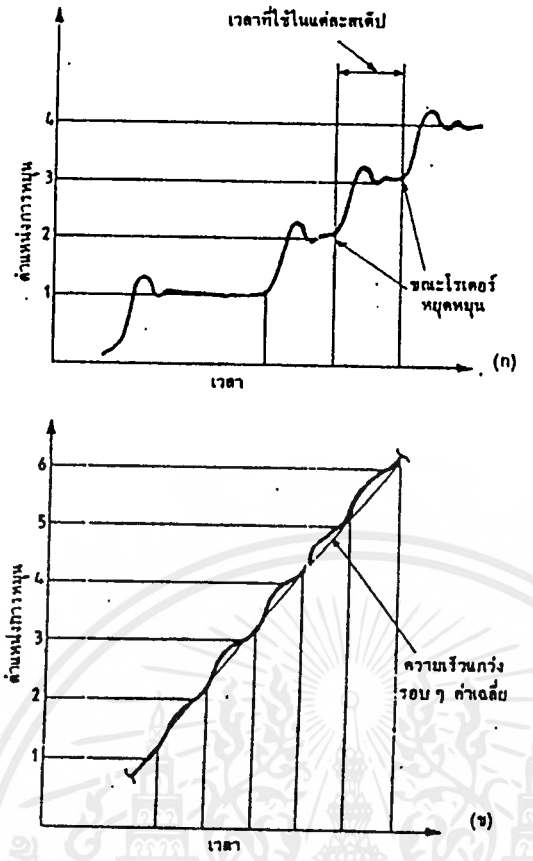


รูปที่ 2.10 ลำดับขั้นตอนการหมุน 4 เฟส

จากรูป ก) จะเห็นว่าสเตเตอร์ในแต่ละขั้วจะมีขดลวดพันอยู่ ซึ่งถือว่าแต่ละขั้วคือหนึ่ง ดังนั้นจากรูปจึงมีทั้งหมด 4 เฟส ด้วยกัน สำหรับการต่อวงจรกระตุ้นเฟสอย่างง่ายแสดงไว้ในรูป ข) จะเห็นว่าปลายขดลวด (c) จะต่อร่วมกันถึงขั้วบวกของแหล่งจ่ายไฟดังนั้นเมื่อเกิดการกระตุ้นที่เฟสใดแล้วขั้วสเตเตอร์ที่เฟสนั้นจะกลายเป็นขั้วเหนือ และจากรูปจะเห็นว่าเป็นการแสดงตำแหน่งของโรเตอร์แต่ละสเต็ป หลังจากถูกกระตุ้นที่เฟส 1 , 2 , 3 และ 4 ตามลำดับ และจะหมุนไปในทิศทางตามเข็มนาฬิกา ทุก 90 องศา ต่อสเต็ป ถ้าต้องการให้หมุนองศาต่อสเต็ปมีค่าลดลงหรือมีความละเอียดในตำแหน่งมากขึ้น จะต้องเพิ่มจำนวนเฟสของสเตเตอร์และจำนวนขั้วแม่เหล็กของโรเตอร์ให้มากขึ้น ข้อเสียของ PM มอเตอร์ คือ ราคาแพง และความหนาแน่นของเส้นแรงแม่เหล็กจะถูกจำกัดที่เส้นแรงแม่เหล็กภายใน ทำให้ไม่สามารถผลิตแรงบิดได้มาก

2.1.2 โหมดการทำงานของสเต็ปปิ้งมอเตอร์

ถ้าจะแบ่งโหมดการทำงานของสเต็ปปิ้งมอเตอร์ตามอัตราเร็วของสเต็ปแต่ละสเต็ป จะแบ่งออกได้เป็น 2 โหมด คือหมุนเป็นสเต็ป และหมุนแบบต่อเนื่อง โดยถ้าหมุนแบบเป็นสเต็ปและมีเวลาหยุดนิ่งก่อนที่จะเปลี่ยนเป็นสเต็ปถัดไป ก็เรียกการทำงานในโหมดนี้ว่า การหมุนเป็นสเต็ป ดังแสดงที่รูป ก) ซึ่งเป็นตัวอย่างการนำไปใช้เป็นเครื่องตอกบัตร



รูปที่ 2.11 แสดงการหมุนในโหมดการทำงาน

ก) หมุนเป็นสเต็ป

ข) หมุนต่อเนื่อง

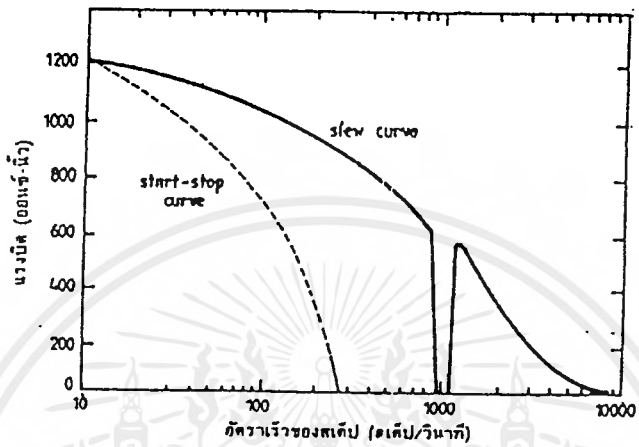
การทำงานคร่าวๆ ก็คือสเต็ปปิ้งมอเตอร์จะเป็นตัวส่งแถบกระดาษเข้าไปในเครื่องปรุกระดาษซึ่งการหมุนของสเต็ปปิ้งมอเตอร์จะเป็นการมุนไปแล้วหยุดชั่วขณะ เพื่อปรุกระดาษให้เรียบร้อยก่อนแล้วค่อยหมุนไปยังตำแหน่งที่ต้องการจะเจาะใหม่

ถ้าเพิ่มความเร็ว ของการหมุนของสเต็ปปิ้งมอเตอร์ ให้เร็วขึ้นและเป็นไปอย่างต่อเนื่องไม่มีการหยุดนิ่งจะเรียกการทำงานนี้ว่า การหมุนแบบต่อเนื่อง ดังแสดงในรูป. ข) ซึ่งจะสามารถหาความสัมพันธ์ ระหว่างความเร็วรอบของมอเตอร์ (n) กับอัตราเร็วของสเต็ป (f) และจำนวนสเต็ปทั้งหมด (S) ได้ดังสมการที่ 2

$$n = 60\ f / s \qquad (2)$$

2.1.3 กราฟแสดงคุณลักษณะของสเต็ปปิ้งมอเตอร์

กราฟแสดงคุณลักษณะของสเต็ปปิ้งมอเตอร์จะเป็นกราฟแสดงความสัมพันธ์ระหว่างอัตราเร็วของสเต็ปกับแรงบิดดังแสดงในรูป



รูปที่ 2.12 กราฟแสดงความสัมพันธ์ระหว่างอัตราเร็วของสเต็ปกับแรงบิดของการทำงาน ทั้ง 2 โหมด

สำหรับกราฟเส้นประเรียกว่า Start Stop Curve หรือ Single Step Load Curve เป็นกราฟที่อยู่ในโหมดการหมุนแบบสเต็ป และเป็นกราฟที่แสดงถึงย่านของแรงบิดที่มอเตอร์สามารถเริ่มและหยุดหมุนได้ โดยปราศจากความผิดพลาดแม้ที่อัตราเร็วของสเต็ปต่างๆ กัน และกราฟอีกเส้นหนึ่งคือ Slew Curve ซึ่งทำงานอยู่ในโหมดการหมุนต่อเนื่องจะเป็นกราฟที่แสดงถึงแรงบิดสูงสุดที่สเต็ปปิ้งมอเตอร์สามารถทำได้ที่อัตราเร็วของสเต็ปต่างๆ กัน ถ้ามีการใช้งานสเต็ปปิ้งมอเตอร์เหนือกราฟอันนี้ก็อาจทำให้เกิดความผิดพลาดได้ ในทางตรงกันข้ามถ้าใช้งานอยู่ภายใต้กราฟนี้แม้จะควบคุมแบบระบบเปิดก็มั่นใจได้ว่าทั้งตำแหน่งและความเร็วมีความเที่ยงตรงแน่นอน โดยตำแหน่งของมอเตอร์สามารถคำนวนได้จากสมการ

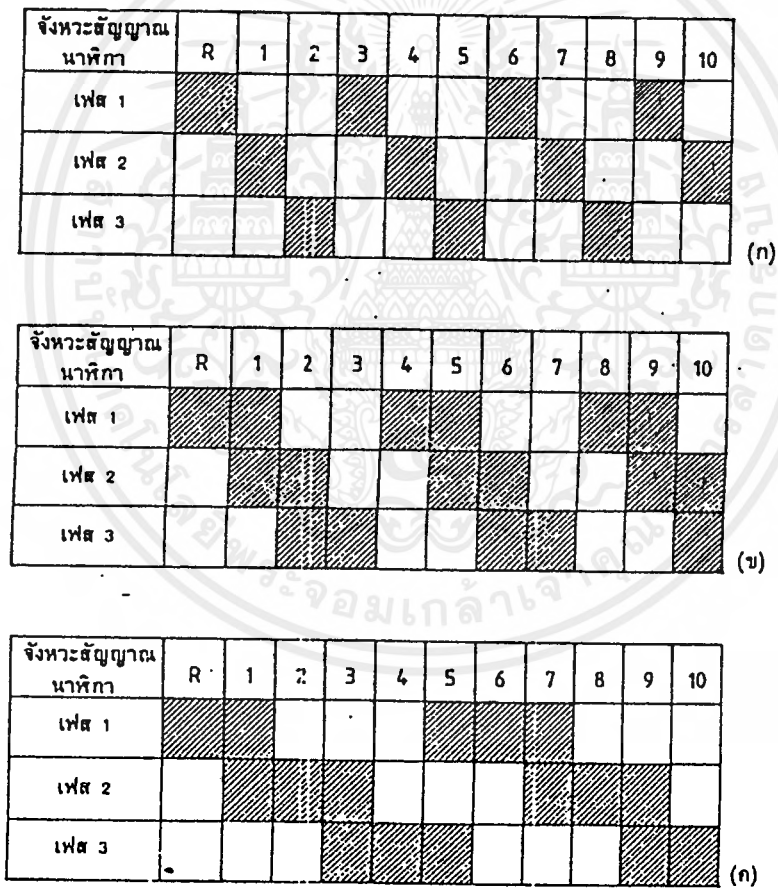
มุมที่เปลี่ยนไป = มุมใน 1 สเต็ป * จำนวนของพัลซ์ที่ป้อนให้

ส่วนความเร็วสามารถหาได้จากสมการที่ 2

สำหรับช่วงที่เส้นกราฟหายไปของ Slew Curve ซึ่งเป็นจุดอ่อนสำคัญที่ต้องคำนึงถึงในการพิจารณาเพื่อใช้งานเพราะช่วงนี้เป็นช่วงที่ไม่เสถียรและควบคุมไม่ได้

2.1.4 วิธีการกระตุ้นเฟส

การที่จะทำให้สเต็ปปิ้งมอเตอร์หมุนได้อย่างต่อเนื่องเหมือนการหมุนของมอเตอร์ไฟฟ้ากระแสตรงนั้นต้องมีการจ่ายกระแสพัลซ์เป็นลำดับอย่างต่อเนื่อง วิธีการที่จะกระตุ้นแบบเฟสมีด้วยกันหลายวิธีแต่จะอธิบายวิธีที่ใช้กันมากเริ่มจากแบบแรก คือ การกระตุ้นแบบเฟสเดียว Single Phase Excitation เป็นการกระตุ้นเฟสเพียงเฟสเดียวเท่านั้นที่สัญญาณนาฬิกาหนึ่งๆ แบบที่ 2 เป็นการกระตุ้น แบบเฟสคู่ Two Phase Excitation ก็จะมีการกระตุ้นเฟส 2 เฟส พร้อมกันในจังหวะสัญญาณนาฬิกาหนึ่งๆ สำหรับแบบสุดท้ายเป็นการกระตุ้นแบบกึ่งสเต็ป Half Step Excitation จะเป็นการรวมเอาทั้งสองแบบเข้าด้วยกันโดยจะทำการกระตุ้นเฟสทั้งสองแบบสลับกันไป ซึ่งลักษณะของการกระตุ้นเฟสทั้งสามแบบสามารถดูได้จากรูป



รูปที่ 2.13 ตารางการกระตุ้นเฟส

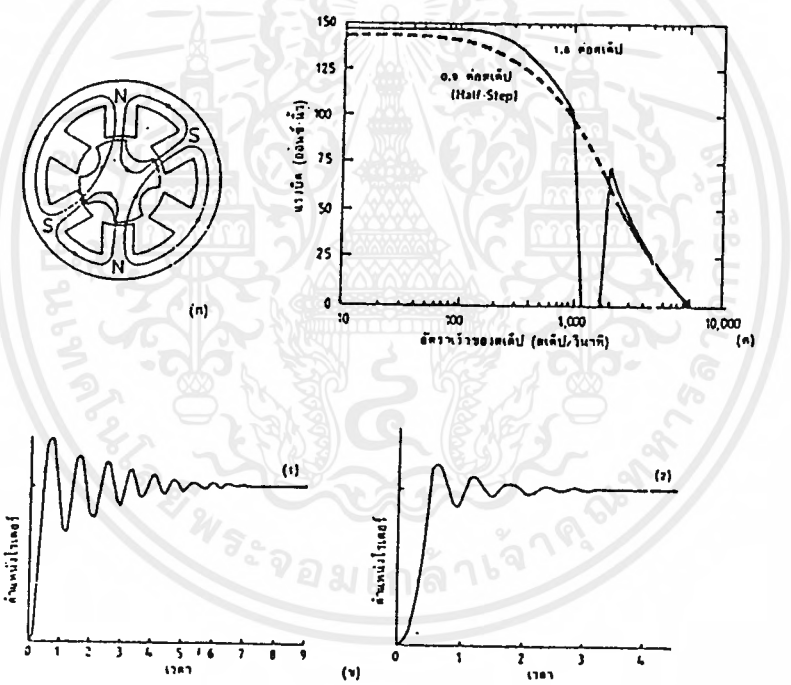
ก) แบบเฟสเดียว Single Phase

ข) แบบเฟสคู่ Two Phase

ค) แบบกึ่งสเต็ป Half Phase

ในการกระตุ้นเฟสแบบเฟสคู่จะมีทิศทางของเส้นและแรงแม่เหล็กไม่เป็นเส้นตรงเหมือนกับการกระตุ้นแบบเฟสเดียว ดังแสดงในรูป ก) แต่ถึงกระนั้นค่าของมุมที่เปลี่ยนไปในหนึ่งสเต็ปก็ยังคงมีค่าเท่าเดิมเหมือนกับการกระตุ้นแบบเฟสเดียว สำหรับการกระตุ้นแบบกึ่งสเต็ปค่ามุมที่เปลี่ยนไปในหนึ่งสเต็ปจะมีค่าลดลงครึ่งหนึ่ง การกระตุ้นแบบเฟสคู่จะมีแรงบิดที่มากกว่าการกระตุ้นแบบเฟสเดียว ขณะเดียวกันยังสามารถเข้าถึงตำแหน่งได้รวดเร็วกว่าดังแสดงไว้ในรูป ข)

ข้อดีอีกอย่างหนึ่งของการกระตุ้นเฟสแบบกึ่งสเต็ปก็คือสามารถลดผลกระทบจากความถี่ในย่านรีโซแนนซ์ได้ แต่ที่ความถี่ต่ำๆ จะมีแรงบิดลดลง ดังแสดงไว้ในรูป ก) และสำหรับทิศทางการหมุนของมอเตอร์ขึ้นอยู่กับทิศทางของลำดับเฟสที่ถูกกระตุ้นด้วย



รูปที่ 2.14 ก) แสดงเส้นทางเดินของเส้นแรงแม่เหล็กในการกระตุ้นแบบเฟสคู่

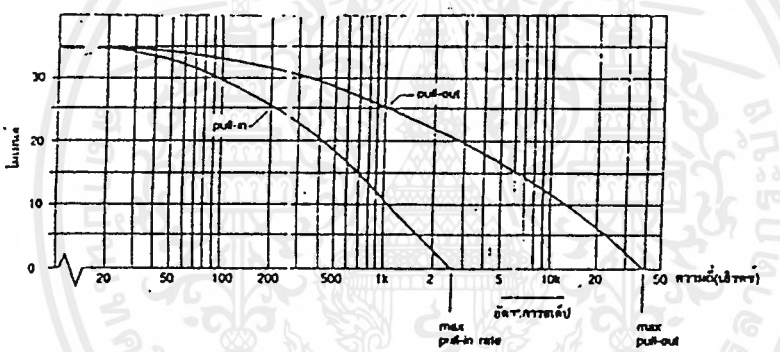
ข) แสดงการเข้าตำแหน่งของโรเตอร์ที่สเต็ปหนึ่งๆ ของแบบ

(1) กระตุ้นแบบเฟสเดียว

(2) กระตุ้นแบบเฟสคู่

คำจำกัดความ

ก่อนที่จะทราบถึงหลักทางคุณสมบัติของสเต็ปปิ้งมอเตอร์ควรจะทราบถึงลักษณะคุณสมบัติของตัวมอเตอร์ก่อน ดังแสดงให้เห็นในตารางจะบอกถึงความหมายของข้อมูลแต่ละอย่างของมอเตอร์ แบ่งประเภทของความหมายได้ 2 ประเภทคือ ทางไฟฟ้าและทางกล ข้อมูลทางกลเป็นสิ่งที่เราต้องการใช้ข้อมูลทางไฟฟ้าใช้สำหรับออกแบบวงจรควบคุมทางอิเล็กทรอนิกส์ ตัวแปรที่มีความหมายที่จะต้องทราบคือ Pull - In Rate (เป็นค่ามากสุดที่ยอมให้เกิดอัตราเร่งสเต็ป) ซึ่งจะมีความสัมพันธ์กับค่าโมเมนต์ความเฉื่อยของโรเตอร์ ซึ่งในทางปฏิบัติแล้วค่าโมเมนต์ความเฉื่อยจะเพิ่มขึ้นได้ด้วยการถูกหมุนโดยตัวมอเตอร์แล้วผลที่ตามมาก็จะทำให้ Pull - In Rate ลดลง ดังแสดงในรูป ซึ่งเป็นกราฟลักษณะคุณสมบัติระหว่างโมเมนต์กับความถี่



รูปที่ 2.15 กราฟแสดงคุณลักษณะโมเมนต์กับความถี่

จะเห็นได้ว่าเมื่อความถี่เพิ่มขึ้นค่าโมเมนต์จะลดลง ที่เป็นเช่นนี้เพราะว่าความถี่ที่เข้ามาสูงขึ้น จะทำให้ค่าอินดัคแตนซ์ที่ขดลวดบนสเตเตอร์สูงขึ้นกระแสจะไหลได้น้อยลง และเป็นผลให้สนามแม่เหล็กน้อยลงด้วย นอกจากนี้กระแสที่ไหลในขดลวดสเตเตอร์ก็ไม่สามารถเปลี่ยนแปลงได้อย่างรวดเร็วด้วย ซึ่งจากกราฟได้แสดงถึงค่าโมเมนต์ 2 โมเมนต์ คือกราฟ Pull - In และกราฟ Pull - Off กราฟ Pull - In ควรจะใช้เมื่อขับมอเตอร์ด้วยความถี่คงที่ ค่าโมเมนต์ก็อยู่ที่ค่าหนึ่ง ส่วนกราฟ Pull - Out ใช้กับการเร่งและการหน่วงความเร็วที่ราบเรียบลื่นไม่กระตุก ซึ่งโมเมนต์จะสูงกว่ากราฟ Pull - In แต่วงจรควบคุมจะซับซ้อนกว่า

ชื่อทางกล	ความหมาย	ชื่อทางไฟฟ้า	ความหมาย
o สเต็ปเปอร์แองเกิล	มุมที่หมุนไปใน 1 สเต็ป มีค่า 360/ จำนวนสเต็ปในการหมุนไป 1 รอบ	o ยูนิโพลาร์กับไบโพลาร์	เป็นชนิดของการพันขดลวดบนตัวสเตเตอร์
o เบรกกิ้งโมเมนท์	เป็นค่าโมเมนต์มากสุดในการบล็อกโรเตอร์ไม่ให้หมุน	o ค่าความเหนี่ยวนำ (L)	เป็นตัวกำหนดขนาดของกระแสที่ในอัตราความเร็วสเต็ปสูงซึ่งสัมพันธ์กับฟลักซ์แม่เหล็ก
o โมเมนต์ (ทอร์ค)	เป็นผลคูณระหว่างระยะทางที่ตั้งฉากกับแรงที่มากระทำ	ค่าความต้านทาน(R)	เป็นตัวจำกัดกระแสที่ขดลวดบนสเตเตอร์กับที่โรเตอร์
o Pull - in rate	ความถี่ที่เริ่มสตาร์ทโดยที่ยังไม่มีการสูญเสีย	o กระแสสเตเตอร์มากสุด	ขึ้นอยู่กับขนาดของขดลวดที่พัน
o Pull - out rate	อัตราของสเต็ปปิ้งเมื่อมีความเร่งคงที่แล้ว		
o โมเมนต์เฉื่อย	เป็นการวัดแรงต้านของวัตถุต่อความเร่งเชิงมุม		

ตารางที่ 2.1 ชื่อเรียกและความหมายทางกลและทางไฟฟ้าของสเต็ปปิ้งมอเตอร์

2.2 การเขียนโปรแกรมด้วยภาษาแอสเซมบลี้

ในที่นี้เราจะกล่าวถึงรูปแบบการเขียนโปรแกรมภาษาแอสเซมบลี้ รวมทั้งการใช้ Comment รูปแบบโดยทั่วๆ ไปของการเขียนโปรแกรมภาษาแอสเซมบลี้ คำสั่งเทียม (Directives) เป็นองค์ประกอบอันหนึ่งที่ใช้ในการเขียนโปรแกรม คำสั่งเทียมที่ใช้สำหรับการกำหนดเซกเมนต์ และ Procedure ดูจากตัวอย่างจะมีการอธิบายที่อยู่ในรูปแบบของโครงสร้างของภาษาแอสเซมบลี้ และจนกระทั่งถึงจุดสุดท้ายของโปรแกรมที่จะทำการเอ็กซิคิวส์

2.2.1 รูปแบบภาษาแอสเซมบลี้ (Assembly Language Comment)

การเขียนคำอธิบายการทำงานของคำสั่ง (Comment) ในการเขียนโปรแกรมภาษาแอสเซมบลี้จะต้องมีการอธิบายให้ชัดเจนตามวัตถุประสงค์ของชุดคำสั่ง การเขียน Comment จะต้องเริ่มต้นด้วย Semicolon (;) ถ้าแอสเซมเบลอร์ตรวจพบเซมิโคลอนมันจะถือว่าข้อความที่ตามมาข้างหลังเป็นคำอธิบายการทำงานของโปรแกรม (Comment) อาจจะเขียนเต็มบรรทัดหรือตามหลังรหัสคำสั่งก็ได้ ดูได้จากตัวอย่างต่อไปนี้

1. ; THIS ENTER LINE IS A COMMENT
2. ADD AX,BX ; COMMENT O SAME LINE AS INSTRUCTION

Comment ที่ตามหลังเซมิโคลอนทั้งหมด แอสเซมเบลอร์จะไม่เปลี่ยนข้อความเหล่านี้เป็นภาษาเครื่องไม่มีผลต่อการทำงานของโปรแกรมที่จะกล่าวในต่อไปนี้ คำสั่งแอสเซมบลี้ใช้ตัวอักษรตัวใหญ่ ตัวเลข ส่วน Comment ใช้ตัวอักษรตัวเล็ก เพื่อสะดวกในการศึกษาโปรแกรม

รูปแบบรหัส (Coding Format)

โดยทั่วๆ ไป Name จะอ้างถึงแอดเดรสของข้อมูล ขณะที่ Label จะอ้างถึงแอดเดรสของคำสั่ง แต่กฎเกณฑ์ในการประยุกต์ใช้งานเหมือนกัน ทั้ง Name และ Label บทนี้จะใช้คำว่า Name ใช้แทนได้ทั้ง Name และ Label รูปแบบทั่วๆ ไป ของคำสั่งมีดังนี คำว่า Name , Operand และ Operation จะถูกการใช้ช่องว่าง 1 ตัวอักษร หรือ 1 Tap สามารถกำหนดตัวอักขระได้สูงสุด 132 ตัว ต่อ บรรทัด แต่ส่วนมากจะใช้ 80 ตัวอักษร เพราะจอภาพมีเพียง 80 ตัวอักษรเท่านั้น

```
NAME        OPERATION       OPERAND         COMMENT
COUNT       DB              1               ; NAME OPERATION , ONE
OPERAND
```

```
MOV         AX,0        ; OPERATION , TWO OPERAND
```

NAME , OPERATION และ OPERAND อาจจะเริ่มต้นได้ทุกคอลัมน์ อย่างไรก็ตามเราจะเริ่มต้นที่คอลัมน์เดียวกัน สำหรับการเขียนโปรแกรม โปรแกรมส่วนมากจะใช้ Editor ช่วยในการเขียนโปรแกรม

NAME หรือ LABEL สามารถใช้อักษรได้ดังต่อไปนี้

ตัวอักษร	A - Z และ a - z
ตัวเลข	0 - 9
เครื่องหมายพิเศษ	Question mark (?)
	Period (.) เฉพาะตัวแรก
	At @
	Underline (_)
	Dollar ($)

ตัวอักขระตัวแรกของ NAME จะต้องนำหน้าด้วยตัวอักษร หรือเครื่องหมายพิเศษ แอสเซมเบลอร์สามารถเข้าใจได้ทั้งตัวใหญ่และตัวเล็ก ความยาวไม่เกิน 31 ตัวอักษร ตัวอย่าง NAME คือ COUNT , PAGE25 , $E10 ส่วนชื่อของรีจีสเตอร์ AX , DI และ AL เป็นคำสงวนที่อ้างถึงรีจีสเตอร์ ใช้เป็น NAME ไม่ได้

```
ADD         AX , BX
```

จากคำสั่งแอสเซมเบลอร์จะรู้ว่า AX และ BX เป็นชื่อที่อ้างถึงรีจีสเตอร์ถ้าคำสั่งเป็น

```
MOV         REGSAVE , AX
```

แอสเซมเบลอร์ก็สามารถจดจำชื่อ REGSAVE ถ้ากำหนดไว้ในโปรแกรม

OPERATION

การกำหนดรายการข้อมูลในโปรแกรมภาษาแอสเซมบลี้ เช่น DB หรือ DW เป็นการกำหนดฟิลด์ของข้อมูล หรือค่าคงที่ เช่นคำสั่งแสดงการทำงานคือ MOV หรือ ADD เป็นส่วนของ OPERATION

OPERAND

รายการข้อมูล หรือตัวประกอบการทำงานของคำสั่ง ที่กำหนดค่าเริ่มแรกตามข้อกำหนดของรายการข้อมูลภายใต้ชื่อ COUNTER รายการข้อมูลหรือเรียกอีกอย่างหนึ่งว่า โอเปอร์แรนด์ จะเป็นตัวกำหนดค่าดังนี้

```
NAME        OPERATION   OPERAND     COMMENT
COUNTER     DB          0           ; DEFINE BYTE ( DB ) WITH 0
```

สำหรับการทำงานของคำสั่ง โอเปอร์แรนด์จะเป็นการแสดงการทำงาน โอเปอร์แรนด์อาจจะมี 1 ตัว หรือ 2 ตัว หรือไม่มีก็ได้ ดังตัวอย่าง

```
                OPERATION   OPERAND     COMMENT
NO OPERAND      RET                     ; RETURN
ONE OPERAND     INC         CX          ; INCREMENT CX
TWO OPERAND     ADD         AX , 12     ; ADD 12 TO AX
```

2.2.2 คำสั่งเทียม (DIRECTIVES)

คำสั่งเทียมจะช่วยสนับสนุนการเขียนโปรแกรมภาษาแอสเซมบลี้ให้สะดวกยิ่งขึ้น คำสั่งเทียมนี้เมื่อเขียนรวมกับคำสั่งภาษาแอสเซมบลี้จะไม่ถูกแปรให้เป็นภาษาเครื่อง หรือบางครั้งคำสั่งเทียมนี้เรียกว่า PSEUDO - CODE

LISTING DIRECTIVE : PAGE AND TITLE

คำสั่ง PAGE และ TITLE จะช่วยควบคุมรูปแบบของรายการพิมพ์ภาษาแอสเซมบลี้

PAGE เป็นคำสั่งเทียมที่บอกถึงจุดเริ่มต้นของการเขียนโปรแกรม ที่บอกจำนวนบรรทัดสูงสุดในแต่ละหน้าและจำนวนขยั่งตัวอักขระสูงสุดในแต่ละบรรทัด มีรูปแบบการใช้ดังนี้

PAGE [LENGTH] , [WIDTH]

จากตัวอย่างการใช้คำสั่ง PAGE ซึ่งกำหนด 60 บรร ทัดต่อหน้า และ 132 อักขระต่อบรรทัด

```
PAGE        60 , 132
```

จำนวนบรรทัดต่อหน้าสามารถกำหนดได้ 10 ถึง 255 บรรทัด และตัวอักษรต่อบรรทัดสามารถกำหนดได้ 60 ถึง 132 ตัวอักขระ ถ้าเป็น PAGE อย่างเดียว แอสเซมบลี้จะกำหนด 50 บรรทัด บรรทัดละ 80 ตัว

TITLE ท่านสามารถใช้คำสั่งเทียม TITLE บอกหัวเรื่องของโปรแกรมที่พิมพ์ 2 บรรทัด ในแต่ละหน้ามีรูปแบบการเขียนดังนี้

TITLE text

ถ้าโปรแกรมมีชื่อ ASMSORT จะเป็นชื่อย่อจะต้องมีคำอธิบายสามารถมีความยาวได้ถึง 60 ตัว ดังนี้

TITLE asmsort Assembly program to sort customer name

SEGMENT DIRECTIVES

โปรแกรมภาษาแอสเซมบลี้ประกอบด้วยหนึ่งเซกเมนต์หรือมากกว่าหนึ่งเซกเมนต์ ในส่วนของ CODE SEGMENT จะใช้สำหรับการเอ็กซิคิวส์รหัสคำสั่ง ส่วน DATA SEGMENT จะใช้ในการกำหนดรายการข้อมูลและ STACK SEGMENT กำหนดพื้นที่ของสแตกในหน่วยความจำ คำสั่งเทียมที่ใช้ในการกำหนดเซกเมนต์คือคำสั่ง SEGMENT มีรูปแบบการเขียนดังนี้

NAME	OPERATION	OPERAND	COMMENT
NAME	SEGMENT	[OPTION]	; BEGIN SEGMENT
-	-		
NAME	ENDS		; END SEGMENT

ชื่อของเซกเมนต์จะต้องอยู่บนสุดและการตั้งชื่อควรจะใช้ชื่อง่ายๆ ส่วนคำสั่ง ENDS เป็นตัวกำหนดจุดสุดท้ายของเซกเมนต์ และต้องมีชื่อกำหนดเหมือนกับชื่อเริ่มต้นของเซกเมนต์ ขนาดของเซกเมนต์ ขนาดของเซกเมนต์จะมีหน่วยความจำ 64 K คำสั่งเทียมเซกเมนต์

(SEGMENT) จะมี OPTION อยู่ 3 ชนิดคือ ALIGMENT , COMBINE และ CLASS มีรูปแบบการเขียนดังนี้

NAME	SEGMENT	'ALIGN COMBINE ' CLASS '

ALIGMENT TYPE วิธีการกำหนดแบบนี้จะกำหนดขอบเขตในเซ็กเมนต์เริ่มต้น สำหรับชนิดที่ต้องการกำหนดขอบเขตใช้คำสั่ง PARA ซึ่งกำหนดขอบเขตของพารากราฟ (PARAGRAPH) ดังนั้นแอดเดรสเริ่มแรกจะถูกหารด้วย 16 หรือ 01 H ยกเว้นโอเปอร์แรนด์ที่แอสเซมเบลอร์สมมติเป็น PARA

COMBINE TYPE วิธีการกำหนดแบบนี้จะเป็นการรวมเข้ากับเซกเมนต์อื่นๆ เมื่อมีการ LINKED เข้าด้วยกัน ชนิดของ COMBINE TYPE คือ STACK , COMMON PUBLIC และ AT expression ตัวอย่างการกำหนดสแตกเซ็กเมนต์ดังนี้

NAME SEGMENT PARA STACK

ท่านจะใช้คำสั่ง PUBLIC และ COMMON เมื่อยกโปรแกรมแอสเซมเบลอร์ แต่ถ้ารวมกันใช้ LINKED ส่วนในกรณีอื่นๆ โปรแกรมที่เขียนจะไม่รวมกับโปรแกรมอื่นๆ

CLASS TYPE จุดเข้านี้จะมีตัว APOSTROPHE S (') กำกับในการใช้กับกลุ่มของเซกเมนต์ซึ่งสัมพันธ์กัน เมื่อมีการ LINK เช่น 'CODE' สำหรับ CODE SEGMENT

NAME ' SEGMENT PARA STACK 'STACK'

คำสั่งเทียม PROC

คำสั่งเทียม PROC ใช้ในส่วนของการกำหนด CODE SEGMENT ที่เป็นส่วนของการเอ็กซีคิวส์โปรแกรมส่วนของ CODE SEGMENT จะมี PROCEDURE หนึ่ง PROCEDURE หรือมากกว่าหนึ่งก็ได้ การกำหนดแต่ละ PROCEDURE จะต้องใช้คำสั่งเทียม PROC

```
SEGNAME     SEGMENT     PARA     COMMENT
PROCNAME    PROC        FAR      ; ONE
                                 ; PROCEDURE
                                 ; WITH IN THE CODE
PROCNAME    ENDP                 ; SEGMENT
SEGNAME     ENDS
```

การกำหนด PROCEDURE จะต้องมีการกำหนดชื่อ OPERAND FAR จะเป็นตัวชี้โปรแกรม DOS สำหรับโหลด PROCEDURE โดยมีคำสั่งเทียม PROC เป็นจุดเข้าที่จะเอ็กซีคิวส์ ส่วนคำสั่งเทียม ENDP จะเป็นตัวกำหนดจุดสิ้นสุดของ PROCEDURE ในส่วนของ CODE SEGMENT จะมี SUB PROCEDURE ตามปกติจะใช้ OPERAND NEAR

คำสั่ง ASSUME

โพรเซสเซอร์จะใช้รีจีสเตอร์ SS กำหนดแอดเดรสของสแต็ก รีจีสเตอร์ DS ใช้กำหนดแอดเดรสของ DATA SEGMENT รีจีสเตอร์ CS กำหนดแอดเดรสของ CODE SEGMENT ถ้าจะต้องมีการบอกชื่อของรีจีสเตอร์เหล่านี้ในแอสเซมเบลอร์ถึงจุดประสงค์ของแต่ละเซกเมนต์ดังต่อไปนี้

```
OPERATION     OPERAND
ASSUME        SS: STACKNAME, DS: DATASEGMENT, CS: CODESEGMENT
```

แต่ในการใช้งานถ้าใช้ ES เก็บข้อมูลเราก็สามารถใช้ ES : DATA SEGMENT แต่ถ้าไม่ใช้ ES เรากำหนด ES : NOTHING

คำสั่ง END

คำสั่งเทียม END จะเป็นการใช้เมื่อสิ้นสุดของการเขียนโปรแกรม หรือสิ้นสุดเซ็กเมนต์ใช้ ENDS หรือเป็นการสิ้นสุด PROCEDURE เราก็ใช้ ENDP

```
OPERATION     OPERAND
END           [ PROCNAME]
```

2.2.3 MEMORY AND REGISTER REFFERENCE

เราจะต้องทำความเข้าใจให้ชัดเจนเกี่ยวกับโอเปอร์แรนด์ที่เป็นชื่อ ชื่ออยู่ในวงเล็บและจำนวน ดังต่อไปนี้สมมติ WORDA กำหนดค่าเวิร์ดในหน่วยความจำดังนี้

```
MOV     AX , BX      ; Move contents of BX to AX
MOV     AX , WORDA   ; Move contents WORDA to AX
MOV     AX , 25      ; Move valuer 25 to AX
MOV     AX , [BX]    ; Move contents of memory location
                     ; specified by BX
```

2.2.4 โครงสร้างโปรแกรม PROGRAM ORGANIZATION

โปรแกรมที่จะทำการเอ็กซิคิวส์มี 2 ชนิดคือ EXE และ COM การพัฒนาโปรแกรมจะต้องพัฒนาเป็น EXE ก่อน ในรูปเป็นการแสดงโครงสร้างของโปรแกรมแบบ EXE มีการเรียงลำดับการกำหนดเซ็กเมนต์ดังนี้

```
STACKSG   SEGMENT   PARA   STACK ' STACK '
DATASG    SEGMENT   PARA   ' DATA
CODESG    SEGMENT   PARA   'CODE'
```

การกำหนดค่าเริ่มแรกของโปรแกรม INITILLIZING A PROGRAM

รูปต่อไปนี้แสดงถึงโครงสร้างของโปรแกรมที่เป็น EXE จะมีการกำหนดค่าเริ่มแรก 2 ชนิด คือ

1. สังเกตุจากกลุ่มรีจีสเตอร์เซกเมนต์ที่จะบอกแอสเซมเบลอร์
2. โหลดค่าแอดเดรสส่งใน DS

```
                PAGE      60 , 132
                TITLE     EXSAM1  Skeleton  of  an  assembly
program
;-----------------------------------------------------------
STACKSG   SEGMENT PARA STACK 'Stack'
          ----
```

```
STACKSG     ENDS
;-------------------------------------------------------
DATASG      SEGMENT PARA 'Data'
            ----
DATASG      ENDS
;-------------------------------------------------------
CODESG      SEGMENT PARA 'Code'
BEGIN       PROC    FAR
            ASSUM   CS : CODESG, DS : DATASG,SS : STACKSG
            MOV     AX , DATASG ; GET ADDRESS OF DATA
SEGMENT.
            MOV     DS , AX     ; STORE IT IN DS
            ----
            MOV     AH , 4CH    ; REQUEST TERMINATE
            INT     21H         ; EXIT TO DOS
BEGIN       ENDP
CODESG      ENDP
            END     BEGIN
```

รูปที่ 2.16 แสดงถึง Skeleton of an EXE Program

คำสั่ง ASSUME ใน CODE SEGMENT จะเ ป็นตัวบอกแอสเซมเบลอร์ถึงกลุ่มรีจีสเตอร์ในเซ็กเมนต์ว่าชื่ออะไร เช่น CS ชื่อ CODESEG , DS ชื่อ DATASEG , และ SS ชื่อ STACKSEG

ASSUME CS : CODSEG , DS : DATASEG , SS : STACKSEG

คำสั่ง ASSUME เป็นการจัดกลุ่มเซ็กเมนต์ของกลุ่มรีจีสเตอร์เซกเมนต์ แอสเซมเบลอร์สามารถกำหนดค่าออฟเซ็ตแอดเดรสในแต่ละเซ็กเมนต์ ดังตัวอย่าง แต่ละคำสั่งที่อยู่ใน CODE SEGMENT ที่กำหนดความยาวคำสั่งแรกจะมีค่าออฟเซ็ตเป็น 0 และถ้าคำสั่งมีความยาว 2 ไบต์ คำสั่งที่ 2 จะมีค่าออฟเซ็ตเป็น 2 ต่อไปเรื่อยๆ ตามความยาวของคำสั่ง

คำสั่งที่กำหนดค่าเริ่มแรกของ DATA SEGMENT ใน DS คือ

```
MOV   AX , DATASEG        ; GET ADDRESS OF DATA SEGMENT
MOVE DS , AX              ; STARE ADDRESS IN DS
```

คำสั่ง MOV คำสั่งแรกจะทำการโหลดแอดเดรสของ DATA SEGMENT ในรีจีสเตอร์ AX และคำสั่ง MOV คำสั่งที่ 2 จะเคลื่อนย้ายข้อมูส AX ไปยัง DS จะเห็นว่ามีการเคลื่อนย้าย ถึง 2 ครั้ง เพราะไม่มีคำสั่งใดที่เคลื่อนย้ายข้อมูลได้โดยตรงจากหน่วยความจำไปยังรีจีสเตอร์เซ็กเมนต์ ถ้าคำสั่ง MOV DS , DATASEG เป็นการทำคำสั่งที่ผิด ไม่สามารถที่จะใช้งานได้

เมื่อ DOS จะทำการโหลดโปรแกรมเอ็กซีคิวส์ไปยังหน่วยความจำ การเอ็กซีคิวส์มีการทำงานดังนี้

1. โครงสร้าง 256 ไบต์ (100H) ของ PROGRAM SEGMENT PREFIX (PSP) จะเป็นตัวกำหนดขอบเขตของหน่วยความจำ
2. โหลดโปรแกรม EXE ทันทีตาม PSP
3. เก็บแอดเดรสของ PSP ใน DS และ ES
4. เก็บแอดเดรสของ COD SEGMENT ใน CS และเซ็ตค่าของ IP เป็นค่าออฟเซ็ตแอดเดรสของคำสั่งปกติน่าจะเป็น 0
5. เก็บค่าแอดเดรสของสแตกใน SS และเซ็ตค่า SP ที่ยอดของสแตก
6. เคลื่อนย้ายการควบคุมไปที่โปรแกรม EXE เพื่อการเอ็กซีคิวส์

โปรแกรม DOS จะโหลดค่าเริ่มแรก CS : IP และ SS : SP แต่การโหลดโปรแกรมจะเซ็ตแอดเดรสของ PSP ใน DS และ ES โดยจะต้องทำการเซ็ตที่สภาวะเริ่มแรกตามในรูป

2.2.5 การสิ้นสุดการเอ็กซีคิวส์โปรแกรม (TERMINATING PROGRAM EXECUTION)

ในส่วนนี้จะอธิบายการสิ้นสุดของโปรแกรมในการเขียนโปรแกรมภาษาแอสเซมบลี้ ดังนี้

DOS SERVICE CALL 4CH

โปรแกรมภาษาแอสเซมบลี้ ต้องทำงานภายใต้ระบบปฏิบัติการที่เรียกว่า DOS ตั้งแต่รุ่น 2.0 เป็นต้นไป โดยใช้คำสั่ง INT 21H ตามรูปที่ 1 โดยเก็บค่า AH = 4CH สำหรับการสิ้นสุดโปรแกรม

```
MOV   H , 4CH             ; DOS SERVICE CALL
MOV   AL , RETCODE                ; RETURN CODE
```

```
INT    21H                    ; TERMINATE AND RETURN TO DOS
```

การ RETURN CODE สำหรับการสิ้นสุดโปรแกรมจะใช้ 0 อาจเขียนได้ดังนี้

```
MOV    AX , 4C00H   ; DOS SERVICE CALL AND RETURN CODE
DOS INERUPT 20H
```

การทำงานชนิดนี้สามารถใช้กับ DOS ได้ทุกรุ่นดังนี้

```
INT    20H          ; TERMINATE AND RETURN CODE
DOS SERVICE CALL 0
```

การทำงานชนิดนี้สามารถใช้กับ DOS ได้ทุกรุ่นดังนี้

```
MOV    AH , 0       ; DOS SERVICE CALL
INT    21H          ; FOR PROGRAM TERMINATION
```

คำสั่ง RET

จะทำหน้าที่ออกจากโปรแกรมเพื่อกลับสู่ DOS หมายความว่าแอดเดรสที่ PUSH ลงในสแตกที่เริ่มต้น (PUSH DS) เป็นผลทำให้คำสั่ง RET จะทำการ POP ที่ยอดแอดเดรสของสแตก (ZEROS) ลงในค่า IP และเพิ่มค่าของ SP เป็น 2 มันจะทำการ POP ที่เวิร์ดต่อไปในค่าของสแตก (แอดเดรสของ PSP) ลงใน CS และเพิ่มค่าของ SP อีก 2 จะได้ค่า CS : IP ใหม่คือเซ็กเมนต์และค่าออฟเซ็ต ที่จุดนี้เป็นจุดเริ่มต้นของ PSP คำสั่ง INT 20 H จะเป็นการควบคุมจะส่งกลับไปที่ DOS

ตัวอย่างของโปรแกรม

รูปที่ 2.17 เป็น SOURCE PROGRAM ของภาษาแอสเซมบลี้ในการบวกค่า 2 ค่า ในรายการที่ข้อมูลในรีจีสเตอร์ AX STACKSEG จะมีข้อมูล 1 ค่า กำหนดโดย DW เป็นการกำหนดค่าขนาด 32 บิต ที่มีค่าเริ่มต้นเป็น 0

DATASEG มีการกำหนดข้อมูลขนาด 3 ตัว มีชื่อ FLDA , FLRB , FLDC

CODESEG เป็นส่วนที่เก็บข้อมูลของโปรแกรมเอ็กซีคิวส์ เพราะว่าคำสั่ง ASSUME เป็นคำสั่งเทียมที่กำหนดค่า CODESEG ในรีจีสเตอร์ CS DATASEG กำหนดค่ารีจีสเตอร์ DS

และ STACKSEG กำหนดค่ารีจีสเตอร์ของ SS ผลการทำงานของ ASSUME จะบอกแอสเซมเบลอร์ในการใช้แอดเดรสของรีจีสเตอร์ CS

สำหรับแอดเดรส CODESEG แอดเดรสในรีจีสเตอร์ DS สำหรับ DATASEG และแอดเดรสของรีจีสเตอร์ SS สำหรับแอดเดรสของ STACKSEG เมื่อมีการโหลดโปรแกรมจากดีสก์ลงในหน่วยความจำ สำหรับการเอ็กซิคิวส์ระบบจะกำหนดการโหลด ACTUAL ADDRESS ใน SS และ CS แต่ไม่มีการเซ็ตค่าเริ่มแรกของ DS และ ES

```
        PAGE  60,132
TITLE       EXASM 1 A (EXE) MOVE AND ADD OPERATIONS
;-------------------------------------------------------------
STACKSG     SEGMENT     PARA STACK ' Stack '
            DW          32 DUP ( 0 )
STACKSG     ENDS
;-------------------------------------------------------------
DATASG      SEGMENT     PARA ' Data '
    FLDA    DW      250
    FLDB    DW      125
    FLDC    DW      ?
DATASG      ENDS
;-------------------------------------------------------------
CODESG      SEGMENT     PARA 'Code'
BEGIN       PROC        FAR
            ASSUME CS:CODESG, DS:DATASG, SS:STACKSG
        MOV AX , DATASG     ; SET ADDRESS OF DATASG
        MOV DS , AX         ; IN DS REGISTER
        MOV AX , FLDA       ; MOVE 250 TO AX
        ADD AX , FLDB       ; ADD 125 TO AX
        MOV FLDC , AX       ; STORE SUM IN FLDC
        MOV AX , 4C00H      ; RETURN TO DOS
        INT 21H
BEGIN   ENDP                ; END OF PROCEDURE
```

```
CODESG      ENDS
            END         BEGIN       ; END OF PROGRAM
```

รูปที่ 2.17 EXE Source Programe with Conventional Segment

สำหรับการทำงานของ PROTECTED MODE ภายใต้การทำงานของโพรเซสเซอร์ตระกูล MCS - 51 จะใช้ DWORD ในการกำหนด ALIGN SEGMENTS ในแอดเดรส DOUBLEWORD เพราะความเร็วในการเข้าถึงหน่วยความจำขนาด 32 บิต ในบัสข้อมูล ในตัวอย่างต่อไปนี้จะใช้คำสั่งเทียม .386 บอกแอสเซมเบลอร์ให้รับรู้ว่าโปรแกรมทำงานภายใต้ MCS - 51

คำสั่ง USE32 จะบอกแอสเซมเบลอร์ในการทำงาน 32 บิต ในส่วนของ PROTECEDS MODE

```
            .386
SEGNAME     SEGMENT     DWORD       USE32
```

การกำหนดค่าเริ่มแรกของรีจีสเตอร์ DS (DATA SEGMENT) มีลักษณะเหมือนกันแต่สามารถใช้รีจีสเตอร์ขนาด 16 บิต

```
MOV EAX , DATASEG           ; GET ADDRESS OF DATA SEGMENT
MOV DS , AX                 ; LOAD 16 BIT PORTION
```

SIMPLIFIES SEGMENT DIRECTIVE

แอสเซมเบลอร์ 0.5 ของไมโครซอฟต์และ BORLAND TURBO ASSEMBLER จะมีการกำหนดเซ็กเมนต์ใน MEMORY MODELS ก่อนการใช้งานทุกๆ ครั้ง ทุกๆ เซ็กเมนต์ มีรูปแบบดังนี้

```
            .MODEL      MEMORY_MODEL
```

MODEL	NUMBER OF CODE SEGMENT	NUMBEROF DATA SEGMENT
SMALL	1	1
MEDUIM	MORE THAN 1	1
COMPACT	1	MORE THAN 1
LARGE	MORE THAN 1	MORE THAN 1

MEMORY MODEL อาจจะเป็น SMALL , MEDUIM , COMPACT หรือ LARGE ในตาราง เราสมารถใช้โมเดลในโปรแกรมเหล่านี้ได้ เช่น SMALL เป็นโมเดลของรหัสคำสั่งที่ใช้เพียง 64 K และในส่วนของข้อมูล 64 K แต่ในตัวอย่างนี้จะใช้เพียง SMALL คำสั่งเทียมโมเดล จะทำงานร่วมกับ ASSUME โดยอัตโนมัติ รูปแบบทั่วๆ ไปของคำสั่งเทียมในการกำหนดค่าของ CODESEG , DATASEG และ STACKSEG มีดังนี้

```
.CODE (NAME)
.DATA
.STACK (SIZE)
```

สำหรับคำสั่งเทียมแต่ละอย่างที่กำหนดค่าแอดเดรสแต่ละเซกเมนต์โดยใช้คำสั่ง SEGMENT และทำร่วมกับคำสั่ง ENDE การกำหนดชื่อเซกเมนต์คือ _TEXT สำหรับ CODE SEGMENT _DATA และ _STACK การใช้ขีดเส้นใต้ (UNDERLINE หรือ BREAK) ที่นำหน้า _TEXT , _DATA เป็นชื่อของเซ็กเมนต์แต่ละชนิดท่านสามารถกำหนด 3 เซกเมนต์ให้แอสเซมเบลอร์ คำสั่งที่กำหนดค่าเริ่มแรกใน DATA SEGMENT คือ

```
MOV   AX , @DATA
MOV   DS , AX
```

จากรูป เราสามารถเปลี่ยนการเขียนโปรแกรมให้สะดวกในการกำหนดเซกเมนต์ดังในตัวอย่างรูป โดยใช้การกำหนดเซกเมนต์ .STACK .DATA และ .CODE โดยไม่ต้องมีคำสั่ง SEGMENT ENDS และ ASSUME

```
                PAGE    60 , 132
                TITLE EXASM1A (EXE) MOVE AND ADD OPERATION
;---------------------------------------------------------------
                .MODEL      SMALL
                .STACK      64          ; DEFINE STACK
                .DATA                   ; DEFINE DATA
FLDA            DW          250
FLDB            DW          125
```

```
FLDC    DW      ?
        .DODE                   ; DEFINE CODE SEGMENT
BEGIN   PROC    FAR
        MOV     AX , @DATA      ; SET ADDRESS OF DATASG
        MOV     DS , AX         ; IN DS REGISTER
        MOV     AX , FLDA       ; MOVE 250 TO AX
        ADD     AX , FLDB       ; ADD 125 TO AX
        MOV     FLDC , AX       ; STORE SUM IN FLDC
        MOV     AX , 4C00H      ; RETURN TO DOS
        INT     21H
BEGIN   ENDP                    ; END OF PTOCEDURE
        END     BEGIN           ; END OF PROGRAM
```

รูปที่ 2.18 EXE Source Program With Simplified Segment Directives

2.2.6 คำสั่ง อินพุต เอาต์พุต

เราได้ศึกษาโครงสร้างโปรแกรมทั้งสองแบบที่ใช้ในการเขียนโปรแกรมภาษาแอสเซมบลี้ และเราได้ศึกษาว่าหน่วยประมวลผลกลาง (CPU) ทำหน้าที่ในการติดต่อกับอุปกรณ์รอบข้าง ส่งผ่านทางรีจีสเตอร์อินพุต เอาท์พุต ที่เรียกว่า I/O PORTS คำสั่งการทำงานของ I/O PORT มี 2 ชนิดคือ IN และ OUT เพราะว่า

1) แอดเดรสพอร์ตขึ้นอยู่กับโมเดลของคอมพิวเตอร์
2) ถ้าใช้โปรแกรมอินพุต เอาท์พุต ของดการบริการจะง่ายกว่าที่จะเขียนโดยโรงงานผู้ผลิต

โปรแกรมอินพุต เอาท์พุตของการบริการมีอยู่ 2 แบบ คือ

1) BASIC INPUT OUTPUT SYSTEM (BIOS)
2) DISK OPERATING SYSTEM (DOS)

โปรแกรมไบออสจะเป็นโปรแกรมที่อยู่ในรอมและมีหน้าที่โดยตรงกับ I/O PORT เราสามารถใช้คำสั่งพื้นฐานในการทำงานของจอภาพ เช่น การเคลื่อนย้ายเคอร์เซอร์ และการเลื่อนจอภาพ ส่วนการบริการของดอส ใช้ในส่วนที่ยุ่งยากกว่า เช่น การพิมพ์ตัวอักขระ

คำสั่ง INT

การใช้คำสั่งของดอสและไบออสเราจะใช้คำสั่ง INT มีรูปแบบดังต่อไปนี้

```
INT    Interupt _ number
```

ขณะที่ Interupt _ number คือจำนวนที่กำหนดการทำงาน ตัวอย่าง คำสั่ง INT 16H ซึ่งเป็นคำสั่งที่ใช้ในการบริการของ ไบออส ทำหน้าที่รับข้อมูลจากคีย์บอร์ด เป็นต้น

INT 21H

คำสั่ง INT 21H ซึ่งเป็นคำสั่งการบริการของดอส จำนวนที่กำหนดการทำงานต่างๆ ที่เรียกว่า Function Call จะเก็บไว้ใน AH และตามด้วยคำสั่ง INT 21H ดังตัวอย่างต่อไปนี้

FUNCTION NUMBER	ROUTINE
1	SINGLE - KEY INPUT
2	SINGLE - CHAR OUTPUT
9	CHARACTER STRING OUTPUT

การทำงานของการบริการ 1

INPUT AH = 1

OUTPUT AL = ASCII CODE (ถ้ามีการพิมพ์ตัวอักขระ)
= 0 (ไม่มีการพิมพ์)

เราสามารถเขียนคำสั่งได้ดังนี้

```
MOV     AH , 1      ; INPUT KEY FUNCTION
INT     21H         ; ASCII CODE IN AL
```

การทำงานของคำสั่งนี้จะเป็นการรอคอยผู้ใช้ให้กดคีย์บอร์ด ถ้าผู้ใช้กดคีร์บอร์ด ข้อมูลรหัส ASCII จะรับค่าเก็บไว้ใน AL จะเป็น 0 คำสั่ง INT 21H จะทำงานตามข้อมูลที่รับมาใน AL

การทำงานของคำสั่ง INT 21H การบริการที่ 1 จะไม่มี PROMPT ให้ผู้ใช้ในการป้อนข้อมูล ทำให้ผู้ใช้อาจไม่รู้ว่าคอมพิวเตอร์กำลังรอรับข้อมูลอยู่ คำสั่งต่อไปนี้จะเป็นตัวสร้าง PROMPT

การทำงานของการบริการ 2

INPUT AH = 2

DL = ASCII CODE (การพิมพ์อักขระทีละตัวหรือตัวควบคุม)

OUTPUT AL = ASCII CODE (แสดงตัวอักขระหรือตัวควบคุมทีละตัว)

เราสามารถเขียนคำสั่งได้ดังนี้

```
MOV     AH , 2      ; DISPLAY CHARACTER
MOV     DL , ' ? '  ; CHARACTER IS ' ? '
INT     21H         ; DISPLAY CHARACTER
```

หลังจากการแสดงตัวอักขระแล้ว เคอร์เซอร์จะเลื่อนไปอีก 1 ตำแหน่ง ถ้าเป็นตัวอักขระควบคุม ข้อมูลที่อยู่ใน DL จะอยู่ในรูปรหัส ASCII เมื่อเอ็กซีคิวส์คำสั่ง INT 21H ก็จะทำงานตามการควบคุมรหัสที่กำหนด จากคำสั่งที่เขียนขึ้น 3 คำสั่ง ข้างบนเราจะเห็นว่า เครื่องหมาย '?' จะแสดงบนจอภาพ

ASCII	SYMBOL	FUNCTION
7	BEL	BEEP (SOUNDS A TONE)
8	BS	BACKSPACE
9	HT	TAB
A	LF	LINE FEED
D	CR	CARRIAGE RETURN

2.2.7 โปรแกรมแสดงการทำงานของการบริการ

การเขียนโปรแกรมจากการอ่านข้อมูลของคีย์บอร์ดและแสดงผลในบรรทัดต่อไป เราเริ่มต้นจากการแสดงเครื่องหมาย '?'

```
MOV     AH , 1      ; DISPLAY CHARACTER FUNCTION
MOV     DL , '?'    ; CHARACTER IS '?'
INT     21H         ; DISPLAY CHARACTER
```

หลังจากจอภาพแสดงเครื่องหมาย Prompt '?' แล้ว ให้อ่านข้อมูลโดยใช้คำสั่งดังนี้

```
MOV     AH , 1      ; READ CHAR FUNCTION
INT     21H         ; CHAR IN AL
```

หลังจากรับข้อมูลและให้แสดงผลที่บรรทัดต่อไป ก่อนที่จะแสดงผลตัวอักขระใน AL จะต้องเก็บไว้ก่อนดังนี้

```
MOV     BL , AL     ; SAVE IT IN BL
```

การเคลื่อนย้ายเคอร์เซอร์ไปเริ่มต้นที่บรรทัดใหม่จะต้องเอ็กซีคิวส์คำสั่ง CR และ LF ซึ่งมีรูปแบบการทำงานดังต่อไปนี้

```
MOV     AH , 2
MOV     DL , 2D         ; CRRIAGE RETURN
INT     21H
MOV     DL , 0AH        ; LINE FFED
INT     21H
```

หลังจากนั้นก็ให้นำค่าของ AL ที่เก็บอยู่ใน BL ออกมาใช้การบริการที่ 2 และตามด้วยคำสั่ง INT 21 H ดังนี้

```
MOV     DL , BL     ; GET CHARACTER
INT     21H         ; AND DISPLAY IT
```

เป็นการสิ้นสุดการทำงานของโปรแกรมที่สมบูรณ์ ดังการเขียนโปรแกรมต่อไปนี้

```
        TITLE       SAMPLE PROGRAM
        .MODEL SMALL
        .STACK 100H
        .CODE
MAIN    PROC
```

```
; DISPLAY PROMPT
  MOV   AH , 2          ; DISPLAY CHARACTER FUNCTION
  MOV   DL , '?'        ; CHARACTER IS '?'
  INT   21H             ; DISPLAY CHARACTER
; INPUT A CHARACTER
  MOV   AH , 1          ; READ CHAR FUNCTION
  INT   21H             ; CHAR IN AL
  MOV   BL , AL         ; SAVE IT IN VL
; GO TO A NEW LINE
  MOV   AH , 2
  MOV   DL , 0DH        ; LINE FFED
  INT   21H
  MOV   DL , 0AH        ; LINE FFED
  INT   21H
; DISPLAY CHARACTER
  MOV   DL , BL         ; GET CHARACTER
  INT   21H             ; AND DISPLAY IT
; RETURN TO DOS
  MOV   AH , 4CH
  INT   21H
MAIN ENDP
  END   MAIN
```

2.3 ไมโครคอนโทรลเลอร์ (MCS-51 Microcontroller)

ไมโครคอนโทรลเลอร์เดิมมีแนวความคิดพื้นฐานมาจากไมโครโพรเซสเซอร์ เพียงแต่จะแตกต่างกันที่ว่าไมโครคอนโทรลเลอร์จะนำเอาอุปกรณ์ที่ทำให้ระบบสมบูรณ์เช่น หน่วยความจำ อินพุตและเอาท์พุตพอร์ตเข้าไปอยู่ในชิปตัวเดียวกัน

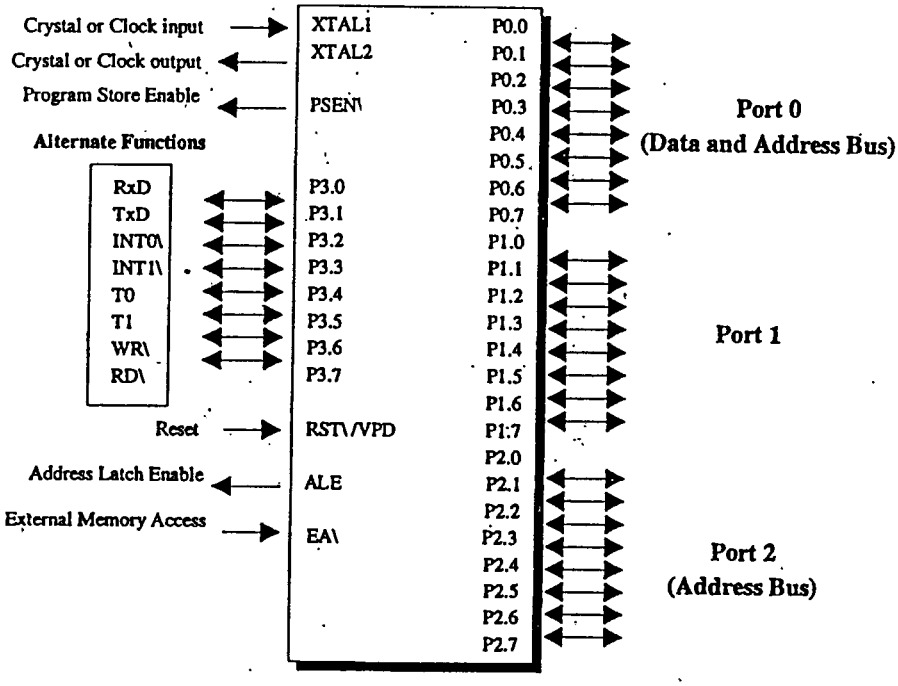
2.3.1 ไมโครคอนโทรลเลอร์ตระกูล 8051

บรรดาไมโครคอนโทรลเลอร์ที่มีการผลิตจากบริษัทต่างๆ จำนวนมากนั้น ไมโครคอนโทรลเลอร์ของบริษัทอินเทล ในตระกูล MCS-51 ได้มีการนำไปใช้กันแพร่หลายมาก ในระยะที่ผ่านมาได้มีหลายบริษัทได้รับลิขสิทธิ์ในการนำไปผลิตจำหนายและได้มีการเพิ่มประสิทธิภาพ และหน่วยทำงานต่างๆ ให้มากขึ้นไปอีกทำให้ในปัจจุบันมีไมโครคอนโทลเลอร์จากบริษัทผู้ผลิตต่างๆ ที่มีพื้นฐานมาจาก MCS-51 ของบริษัทอินเทลเป็นจำนวนมาก

2.3.2 สถาปัตยกรรมของ 8051

8051 ไมโครคอนโทรลเลอร์จริงๆ แล้วจะรวมเอาตระกูล 8031 ถึง 8751 ทั้งหมด ซึ่งเป็นพวก NMOS และจะมีส่วนประกอบของ Cmos ใน Package หลายชนิด

รุ่นที่ปรับปรุงใหม่ 8052 ซึ่งตระกูลนี้มีหลายชนิด และมีชนิดหนึ่งที่สามารถโปรแกรมด้วยภาษาเบสิก (Basic) รุ่นนี้เกิดจากความต้องการของบริษัทผู้ผลิต ที่ไม่ต้องการให้เกิดช่องว่างทางการตลาดจึงต้องมีหลายๆ แบบ โดยที่ในรุ่น 8051 จะมีขาอยู่ 40 ขา ดังรูป



รูปที่ 2.19 แสดงการกำหนดขาของ 8051

หน้าที่การทำงานของแต่ละขามีดังนี้

- ขาที่ 1-8 ทำหน้าที่เป็นอินพุตเอาท์พุตของพอร์ต 1

- ขาที่ 9 ทำหน้าที่รับสัญญาณรีเซ็ท (Reset) จากภายนอก เพื่อให้ไมโครคอนโทรลเลอร์เริ่มการทำงานใหม่ทั้งหมด

- ขาที่ 10 - 17 ทำหน้าที่เป็นอินพุตเอาท์พุตของพอร์ต 3 และนอกจากนี้แต่ละขายังสามารถทำหน้าที่อื่นๆ ได้ดังต่อไปนี้อีกคือ

- ขาที่ 10 ทำหน้าที่รับข้อมูลแบบอนุกรมจากอุปกรณ์ภายนอก

- ขาที่ 11 ทำหน้าที่ส่งข้อมูลแบบอนุกรมให้กับอุปกรณ์ภายนอก

- ขาที่ 12 ทำหน้าที่รับสัญญาณอินเทอรัพท์ 0 (Intrerupt 0) จากภายนอกโดยสัญญาณอินเทอรัพท์นี้จะทำงานเมื่อเป็นลอจิกต่ำ

- ขาที่ 13 ทำหน้าที่รับสัญญาณอินเทอรัพท์1 จากภายนอกโดยสัญญาณอินเทอรัพท์นี้จะทำงานเมื่อเป็น ลอจิกต่ำเหมือนกับขาที่ 12

- ขาที่ 14 ทำหน้าที่รับสัญญาณจากภายนอกให้กับไทม์เมอร์ 0 (Timer 0)

- ขาที 15 ทำหน้าที่รับสัญญาณจากภายนอกให้กับ ไทม์เมอร์ 1

- ขาที่ 16 ทำหน้าที่ส่งสัญญาณไปยังอุปกรณ์ตัวอื่นๆ เมื่อต้องการเขียนข้อมูลไปยังอุปกรณ์ตัวนั้น

- ขาที่ 17 ทำหน้าที่ส่งสัญญาณไปยังอุปกรณ์ตัวอื่นๆ เมื่อต้องการอ่านข้อมูลจากอุปกรณ์ตัวนั้นๆ

- ขาที่ 18 - 19 เป็นขาสำหรับรับสัญญาณจากคริสทัล (Crystal) เพื่อกำเนิดสัญญาณนาฬิกาให้กับตัวไมโครคอนโทรลเลอร์ โดยภายในตัวไมโครคอนโทรลเลอร์เองก็จะมีวงจรออสซิลเลเตอร์อยู่ในตัวเองอยู่แล้ว

- ขาที่ 20 ใช้ต่อกราวด์ของระบบ

- ขาที่ 21 - 28 ทำหน้าที่เป็นอินพุตเอาท์พุตของพอร์ต 2 หรือทำหน้าที่ส่งคำสั่งเป็น High byte address bus เพื่อต่อหน่วยความจำภายนอกสำหรับเก็บข้อมูล หรือเพื่อเก็บชุดคำสั่ง หรือต่อกับอุปกรณ์และไอซีอื่นๆ ภายนอกไมโครคอนโทรลเลอร์

- ขาที่ 29 ทำหน้าที่ส่งสัญญาณเพื่อสั่งให้หน่วยความจำภายนอกชิปซึ่งเก็บชุดคำสั่งไว้ทำงาน

- ขาที่ 30 ทำหน้าที่ส่งสัญญาณควบคุมการแลตซ์ (Latch) ค่า Low byte address bus จากพอร์ต 0 ในการติดต่อกับส่วนที่ใช้ในการเก็บชุดคำสั่งภายนอก

- ขาที่ 31 เป็นขาที่ผู้ใช้สามารถเลือกให้ไมโครคอนโทรลเลอร์นี้ทำงานจากชุดคำสั่งภายในหรือทำงานจากชุดคำสั่งภายนอกไมโครคอนโทรลเลอร์ก็ได้ โดยเมื่อต่อขานี้เข้ากับ Logic High ไมโครคอนโทรลเลอร์ก็จะทำงานจากโปรแกรมภายในไมโครคอนโทรลเลอร์เอง และเมื่อต่อเข้ากับ Logic Low ไมโครคอนโทรลเลอร์ก็จะทำงานจากโปรแกรมภายนอกตัวไมโครคอนโทรลเลอร์

- ขาที่ 32-39 ทำหน้าที่เป็นอินพุตเอาท์พุตของพอร์ต 0 หรือทำหน้าที่เป็น Low byte address buss สลับกับการเป็น Data bus

- ขาที่ 40 ต่อกับแหล่งจ่ายเพื่อจ่ายพลังงานให้กับไมโครคอนโทรลเลอร์

2.3.3 หน่วยความจำของ 8051

การแบ่งหน่วยความจำของ 8051 จะสามารถแบ่งออกได้เป็น 2 ลักษณะคือ

1. หน่วยความจำโปรแกรม
2. หน่วยความจำข้อมูล

หน่วยความจำโปรแกรม

หน่วยความจำโปรแกรมเป็นหน่วยความจำที่ใช้สำหรับเก็บคำสั่งหรือโปรแกรมที่ผู้ใช้พัฒนาขึ้นมา โดยอาจประกอบอยู่ภายในตัวไอซีหรือเป็นไอซีหน่วยความจำประเภทอีพรอม ซึ่งในกรณีหลังนั้นจำเป็นจะต้องใช้พอร์ตอินพุต/เอาท์พุตทำหน้าที่แอสเดรสบัสและด้าต้าบัส เพื่อให้สามารถต่อเขากับหน่วยความจำ ไอซีมาตราฐานได้

การเชื่อมต่อหน่วยความจำโปรแกรมภายนอกเข้ากับ 8051

เนื่องจากระบบบัสแอสเดรสและบัสข้อมูลของไมโครคอนโทรลเลอร์8051 เป็นลักษณะการใช้มัลติเพล็กซ์จากพอร์ตเดียวกัน กล่าวคือ ในระยะเวลาเริ่มต้น เส้นสัญญาณเหล่านี้ของพอร์ตจะใช้ในการส่งค่าแอสเดรสของตำแหน่งที่ต้องการติดต่อด้วย ในช่วงเวลาต่อมาจึงจะเปลี่ยนไปเป็นสภาวะ High Impedance เพื่อใช้งานในสถานะของบัสข้อมูล แต่เนื่องจากว่าอีพรอม (EPROM)ที่ใช้กันโดยทั่วไปนั้นไม่ใช้การมัลติเพล็กซ์ และมีขาสัญญาณบัสแอสเดรสและบัสข้อมูลแยกจากกันโดยชัดเจนดังนั้นการเชื่อมต่ออีพรอมเพื่อเป็นหน่วยความจำโปรแกรม จึงจำเป็นต้องมีวงจรประเภทแลตซ์ประกอบขึ้นเพิ่มเติมเพื่อทำการค้างค่าของแอสเดรสที่ส่งออกมาจาก 8051 ในช่วงเวลาแรกให้กับขาสัญญาณแอสเดรสของอีพรอมต่อไป

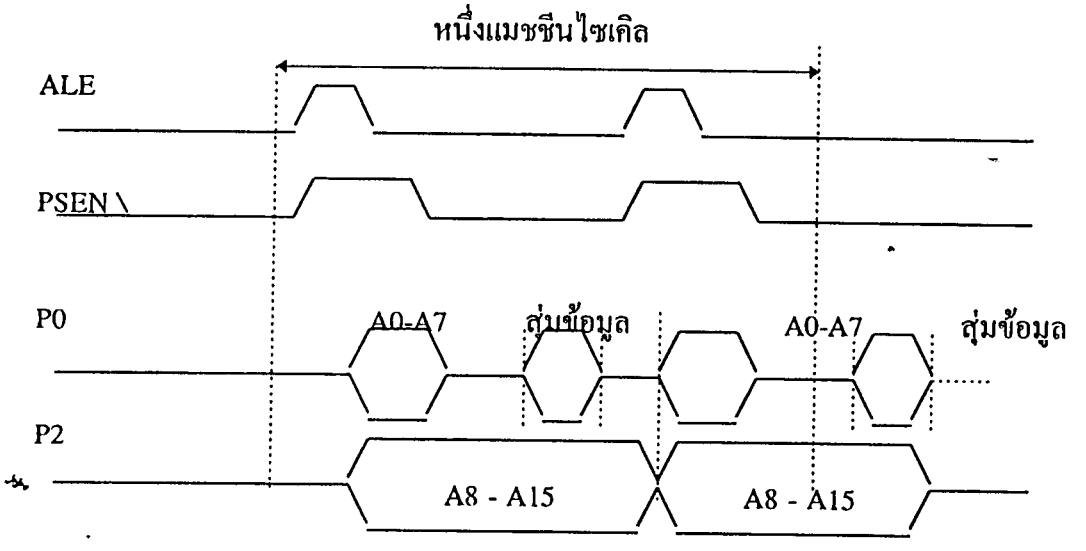
จากตารางในรูปที่ 2.2 แสดงให้เห็นถึงสัญญาณต่างๆ ของ 8051 ซึ่งนำมาใช้ในการติดต่อกับหน่วยความจำภายนอก

สัญญาณ	คำจำกัดความ	ขาสัญญาณ	หน้าที่
EA	External Acess	31	เลือกประเภทหน่วยความจำภายในหรือนอก
ALE	Address Enable	30	สัญญาณเอาท์พุตสำหรับแลตซ์ข้อมูล
P2.0-P2.3	Port2	21-28	เป็นข้อมูลแอสเดรสไบต์สูงของหน่วยความจำ
P0.0-P0.7	Port0	39-32	มัลติเพล็กซ์สัญญาณบัสแอสเดรสและบัสข้อมูล
Psen	Program Store Enable	29	สัญญาณระบุการอ่านให้กับหน่วยความจำ

ตารางที่ 2.2 สัญญาณของ 8051 ที่ใช้ระหว่างการติดต่อเพื่ออ่านข้อมูลจากหน่วยความจำโปรแกรมภายนอก

สัญญาณ EA (External Acess) ใช้ในการกำหนดเลือกว่า จะอ่านข้อมูลมาจากหน่วยความจำโปรแกรมภายนอกหรือภายในตัวไอซีไมโครคอนโทรลเลอร์เอง ซึ่งหากเป็นระดับลอจิกต่ำจะอ่านข้อมูลมาจากหน่วยความจำข้อมูลภายนอก และกรณีตรงข้ามก็จะอ่านข้อมูลมาจากหน่วยความจำภายในตัวไอซีเอง สิ่งที่ควรสังเกตุ คือเมื่อมีการอ่านข้อมูลจากหน่วยความจำข้อมูลภายใน และมีการใช้งานแอสเดรสที่อยู่ในช่วงที่สูงกว่าค่าสูงสุดของหน่วยความจำข้อมูลภายในกรณีเช่นนี้ 8051 จะทำการอ่านค่าแอสเดรสที่สูงกว่ามาจากหน่วยความจำโปรแกรมภายนอกโดยอัตโนมัติ

จากแผนภาพแสดงเวลาการติดต่อกับหน่วยความจำโปรแกรมภายนอกจะเห็นว่าภายในช่วงเวลาของแมชชีนไซเคิลหนึ่ง 8051 จะมีการไปนำข้อมูลมา 2 ครั้ง ในช่วงเวลาเริ่มต้นของการติดต่อหน่วยความจำภายนอก P0 จะเป็นค่าของแอสเดรสไบต์ต่ำ และเวลาต่อมาจึงเป็นบัสข้อมูล การส่งค่าแอสเดรสไบต์ต่ำนี้จะอยู่ในราวช่วงเวลาขอบขาลงของสัญญาณ ALE และจะยังคงอยู่จนเมื่อสัญญาณ PSEN เปลี่ยนเป็นสัญญาณระดับลอจิกต่ำ ดังนั้นการออกแบบจึงมักใช้สัญญาณ ALE นี้ทำการให้ไอซีแลตช์ภายนอกค้างระดับสัญญาณแอสเดรสเหล่านี้ไว้ ส่วนสัญญาณ PSEN จะใช้ในการเลือกไอซีอีพรอมให้ทำงานและอ่านค่าข้อมูลกลับมา



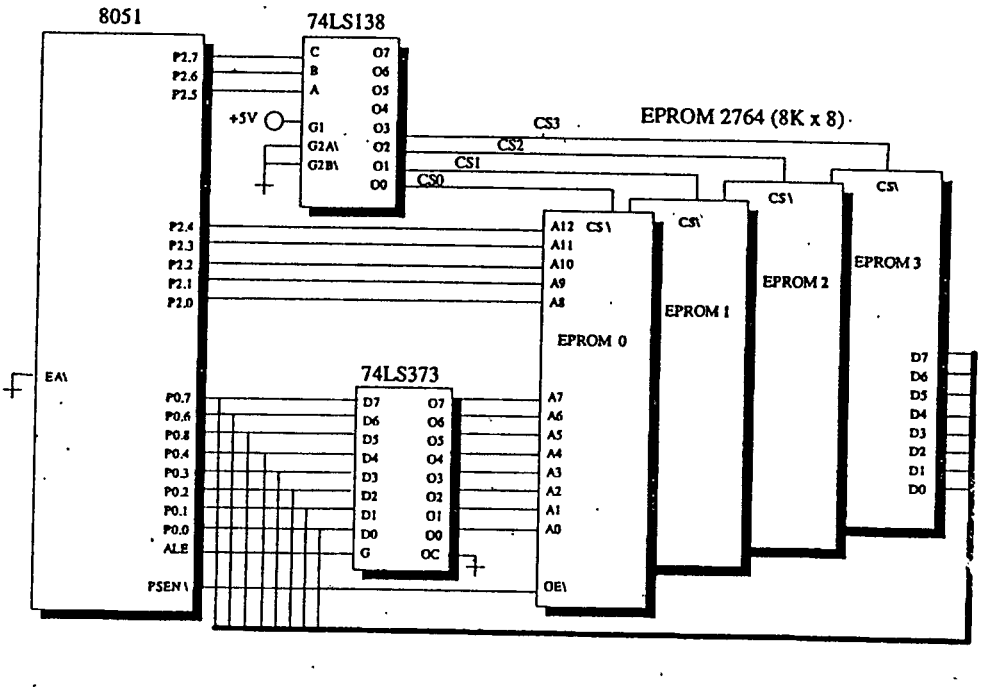
รูปที่ 2.20 แสดงภาพสัญญาณเวลาแสดงการติดต่อกับหน่วยความจำภายนอก

ขณะเมื่อสัญญาณ PSEN เป็นระดับลอจิกต่ำอีพรอมก็จะทำการถอดรหัสค่าแอสเดรส และส่งข้อมูลที่อยู่ในตำแหน่งนี้ออกมา โดยสัญญาณ PSEN นี้จะค้างสภาวะการเป็นลอจิกต่ำไว้ช่วงเวลาหนึ่งเพื่อให้ข้อมูลถูกส่งออกมาจากอีพรอมเรียบร้อยแล้วจึงค่อยกลับไปเป็นระดับลอจิกสูงเช่นเดิม

และในช่วงจังหวะขาขึ้นของสัญญาณ PSEN 8051 ทำการสุ่มอ่านข้อมูลเข้ามา สำหรับข้อมูลทางพอร์ต 2 ซึ่งเป็นค่าแอสเดรสไบต์สูง จะถูกส่งออกมาในราวช่วงกึ่งกลางระหว่างที่สัญญาณ ALE เป็นลอจิกสูง ซึ่งก็เป็นเวลาใกล้เคียงกับการส่งแอสเดรสไบต์ต่ำออกมาทางพอร์ต 0 สำหรับค่าแอสเดรสพอร์ต 2 นั้น จะค้างอยู่ตลอดช่วงรอบเวลาของการติดต่อกับหน่วยความจำโปรแกรม

การใช้งาน 8051 แบบไม่มีหน่วยความจำโปรแกรมภายในนั้น จำเป็นจะต้องเชื่อมต่อเข้ากับหน่วยความจำโปรแกรมซึ่งเป็นไอซีอีพรอมและจะต้องกำหนดให้เริ่มต้นที่แอสเดรส 0000Hเสมอ ทั้งนี้เพราะเมื่อมีการรีเซ็ทหรือเริ่มต้นการจ่ายไฟให้กับระบบ 8051 จะได้เริ่มต้นการทำงานตามคำสั่งนี้ทันที

โดยปกติแล้ว 8051 สามารถต่อหน่วยความจำโปรแกรมภายนอกได้สูงสุดไม่เกิน 64 กิโลไบต์ ดังนั้นในกรณีที่ใช้ไอซีอีพรอมที่มีหน่วยความจำไม่ถึง 64 กิโลไบต์ก็สามารถนำมาต่อกันให้เพิ่มจนถึง 64 กิโลไบต์ได้ โดยการต่อสัญญาณเพื่อเลือกตัวไอซี การเลือกตัวไอซีสามารถทำได้โดยการใช้วงจรถอดรหัสแอสเดรส เพื่อกำหนดพื้นที่ ภายใน หน่วยความจำ แต่ละตัวดังรูปที่ 2.21



รูปที่ 2.21 การเชื่อมต่อเพื่อขยายหน่วยความจำโปรแกรม ด้วยไอซีอีพรอมหลายตัว

หน่วยความจำข้อมูล

หน่วยความจำข้อมูลของ 8051 มีไว้ใช้สำหรับเก็บข้อมูลหรือตัวแปรที่เกิดขึ้นในขณะที่กำลังประมวลผลโปรแกรมไว้เป็นการชั่วคราว โดยพื้นฐานแล้วหน่วยความจำข้อมูลจัดเป็นหน่วยความจำแบบสแตติก ดังนั้นเมื่อไม่มีการจ่ายไฟฟ้าให้กับระบบก็จะมีผลทำให้ข้อมูลที่เก็บไว้ภายในสูญหาย สำหรับพื้นที่ของหน่วยความจำข้อมูลภายในของ 8051 สามารถมีได้สูงสุดไม่เกิน 64 กิโลไบต์ และแยกประเภทออกเป็น 2 ลักษณะตามตำแหน่งที่ตั้งของหน่วยความจำนั้น คือหน่วยความจำภายใน ซึ่งเป็นแรมที่อยู่ภายในตัวไอซีเอง และหน่วยความจำข้อมูลภายนอกซึ่งเป็นการใช้ไอซีหน่วยความจำมาเพิ่มเติมเข้าไปในวงจร ลักษณะเดียวกับการนำเอาไอซีอีพรอมมาต่อเป็นหน่วยความจำโปรแกรม

หน่วยความจำข้อมูลภายใน

8051 มีจำนวนทั้งหมด 256 ไบต์ โดยสามารถจำแนกออกได้เป็น 2 ลักษณะ คือ พื้นที่เฉพาะสำหรับตัวประมวลผลกลาง ซึ่งจะเรียกว่า รีจิสเตอร์ และพื้นที่ใช้งานทั่วไปสำหรับโปรแกรมที่ผู้ใช้สร้างขึ้นมาหน่วยความจำขนาด 128 ไบต์แรก บริเวณนี้จะมีตำแหน่งแอสเดรสอยู่ในช่วง 00H - 7FH ซึ่งมีการจำแนกออกเป็น3 ส่วนตามการใช้งานคือ

บริเวณแอสเดรส 00H - 1FH แบ่งออกเป็นกลุ่มข้อมูลจำนวน 8 ไบต์ รวมทั้งหมด 4 กลุ่ม โดยที่พื้นที่ในแต่ละกลุ่มจะถูกใช้งานในฐานะของรีจิสเตอร์ใช้งานทั่วไป ซึ่งมีชื่อเรียกว่า รีจิสเตอร์ R0 - R7

บริเวณแอสเดรส 20H - 2FH จำนวน 16 ไบต์ พื้นที่บริเวณนี้เป็นพื้นที่ส่วนสำหรับผู้ใช้ซึ่งจะมีความพิเศษแตกต่างไปจากหน่วยความจำส่วนอื่นๆ เนื่องจากผู้ใช้สามารถอ้างถึงหน่วยความจำในบริเวณนี้ได้ในลักษณะไบต์ข้อมูลหรือในลักษณะบิตข้อมูลได้โดยตรง ดังนั้นหากมองในลักษณะบิตข้อมูลแล้วจะมีพื้นที่ตัวแปรบิตให้ใช้ได้มากถึง 128 บิต

บริเวณแอสเดรส 30H - 7FH เป็ นบริเวณที่ผู้ใช้สามารถนำไปใช้งานได้อย่างอิสระ โดยสามารถอ้างถึงได้ในลักษณะของไบต์ข้อมูลตามปกติเท่านั้น

หน่วยความจำขนาด 128 ไบต์ถัดไป เป็นพื้นที่ตั้งแต่บริเวณแอสเดรส 80H - FFH เป็นบริเวณของหน่วยความจำที่มีการใช้งานเฉพาะจาก 8051 เท่านั้น โดยจะนำมาใช้เป็นตำแหน่งของรีจิสเตอร์หน้าที่พิเศษ จำนวน 20 ตำแหน่ง

2.3.4 รีจิสเตอร์หน้าที่พิเศษ

เป็นรีจิสเตอร์สำหรับการควบคุมหน้าที่และการทำงานของพอร์ตทั้งหมด โดยมีตำแหน่งอยู่ในบริเวณแอสเดรส 80H - FFH การใงานรีจิสเตอร์พิเศษสามารถระบุชื่อหรืออ้างถึงตำแหน่งของรีจิสเตอร์เหล่านั้นก็ได้

แอกคิวมูเลเตอร์ รีจีสเตอร์ (Accumulator) หรือ ACC

เป็นรีจีสเตอร์ขนาด 8 บิต ทำหน้าที่ในการเก็บข้อมูลที่จะส่งให้กับหน่วยทำงานภายใน CPU และเก็บผลลัพท์ที่ได้จากการทำงานนั้นการทำงานของ รีจิสเตอร์แอคคิวมูเลเตอร์นี้มีลักษณะการทำงานเช่นเดียวกับแอคคิวมูเลเตอร์ทั่วๆ ไป

รีจิสเตอร์ B (Registor B)

เป็นรีจิสเตอร์ที่ใช้สำหรับการทำคำสั่งคูณและหารตัวเลข ในกรณีที่ไม่ใช้ในการคำนวนทางด้านคณิตศาสตร์ ก็สามารถนำไปใช้งานเช่นเดียวกับริจิสเตอร์ทั่วๆไปได้

โปรแกรมเคาน์เตอร์ (Program counter)

เป็นรีจิสเตอร์ที่ใช้ชี้ตำแหน่งแอสเดรสของหน่วยความจำโปรแกรม ซึ่งจะต้องไปทำงานในลำดับถัดไป การใช้งานภายในโปรแกรมจะเรียกว่า รีจิสเตอร์ PC

สแต็กพอยน์เตอร์ (Stack pointer)

เป็นรีจิสเตอร์ขนาด 8 บิท ทำหน้าที่เก็บตำแหน่งของตัวชี้หรือพอยน์เตอร์ของบริเวณสแต็กสำหรับเก็บข้อมูลของแอคคิวมูเลเตอร์ รีจิสเตอร์ต่างๆ รวมทั้งข้อมูลจากโปรแกรมโดยปกติแล้วเมื่อทำการเริ่มต้นระบบใหม่ภายหลังจากการจ่ายไฟฟ้า หรือมีการรีเซ็ทเกิดขึ้น ค่าภายในสแต็กพอยน์เตอร์จะมีค่า 07H ซึ่งเป็นตำแหน่งแอสเดรหัสภายในเนื้อที่บริเวณ 128 ไบต์แรกของหน่วยความจำข้อมูลภายใน การใช้งานภายในจะเรียกว่า รีจิสเตอร์ SP

ตัวชี้ข้อมูลหรือดาต้าพอยน์เตอร์ (Data pointer)

เป็นรีจิสเตอร์ขนาด 16 บิทซึ่งเรียกว่ารีจิสเตอร์ DPTR และสามารถใช้งานแยกเป็นรีจิสเตอร์ขนาด 8 บิท 2 ตัวคือ รีจิสเตอร์ DPH และรีจิสเตอร์ DPL เพื่อเก็บค่าของแอสเดรสของหน่วยความจำที่ต้องใช้ภายในโปรแกรมหรืออาจจะเป็นแอสเดรสของอุปกรณ์ภายนอก ซึ่งกำหนดให้ติดต่อกันโดยใช้ตำแหน่งของหน่วยความจำนั้นภายในโปรแกรม

โปรแกรมสเตตัสเวิร์ด (PSW)

รีจิสเตอร์นี้ทำหน้าที่บอกถึงแฟล็กสภาวะการทำงานต่างๆ รวมทั้งบิทสำหรับการกำหนดเลือกแบงค์ของรีจิสเตอร์ที่ใช้งานด้วย

รีจิสเตอร์ที่เกี่ยวข้องกับพอร์ต (Port register)

รีจิสเตอร์เหล่านี้จะมีความเกี่ยวข้องกับการทำงานของพอร์ตอินพุต/เอาท์พุตโดยตรง ซึ่งแต่ละตัวจะเป็นรีจิสเตอร์ขนาด 8 บิท สามารถใช้งานได้ในลักษณะของการอินพุต หรือการเอาท์พุตข้อมูลได้ การดำเนินการใดๆที่เกี่ยวกับพอร์ตทั้ง 4 นี้ข้อมูลที่ตำแหน่งของพอร์ตเหล่านี้เปลี่ยนแปลงไปเช่นกัน นอกจากนี้ พอร์ต P0 และ พอร์ต P2 ยังสามารถที่จะใช้ในการติดต่อกับหน่วยความจำโปรแกรมหรือหน่วยความจำข้อมูลภายนอกได้ โดยพอร์ต P2 จะเป็นค่าแอสเดรสของ 8 บิทบนของหน่วยความจำ ส่วนพอร์ต P0 นั้นในช่วงเริ่มแรกจะเป็นค่าของแอสเดรสในช่วง 8 บิทล่างของหน่วยความจำ ช่วงเวลาต่อมาจึงนำพอร์ต P0 ไปใช้เป็นบัสในการรับหรือส่งข้อมูลกับหน่วยอุปกรณ์ภายนอก สำหรับพอร์ต P3 นอกเหนือจะนำไปใช้เป็นพอร์ตอินพุตเอาท์พุตตามปกติแล้วยังนำไปใช้ในฐานะบัสควบคุมเกี่ยวกับการอินเตอร์รัปต์ได้อีกด้วย

รีจิสเตอร์ SBUF

เป็นบัฟเฟอร์ขนาด 8 บิท สำหรับการติดต่อสื่อสารข้อมูลแบบอนุกรมทั้งการรับและการส่งข้อมูล ซึ่งตามความเป็นจริงแล้วบัฟเฟอร์นี้มีอยู่ด้วยกัน 2 ชุด และแยกจากกันอย่างชัดเจน สำหรับการส่งและการรับ โดยซีพียูจะทำการเลือกบัฟเฟอร์ที่เหมาะสมให้โดยอัตโนมัติ

รีจิสเตอร์ PCON

เป็นรีจิสเตอร์ที่ใช้ในการทำหน้าที่การทำงานในสามลักษณะ ซึ่งได้แก่ การควบคุมการทำงานของโพรเซสเซอร์ (บิท IDL และ PD) การกำหนดอัตราการทวีคูณของอัตราเร็วในการสื่อสารข้อมูลอนุกรม (บิท SMOD) และแฟล็กสภาวะสำหรับการใช้งานทั่วไป (บิท GR0) และบิท GR1)

บิท PD (Power down)

เป็นการกำหนดให้ลดกำลังไฟฟ้าที่จ่ายให้กับส่วนของโพรเซสเซอร์ภายในลง โดยยังคงมีกำลังไฟฟ้าจ่ายให้กับส่วนหน่วยความจำข้อมูลภายในผ่านทางขาสัญญาณ RST วิธีการนี้มักนำมาใช้ในกรณีที่มีการตรวจสอบการไม่มีกำลังไฟฟ้า (Power failure) โดยวงจรตรวจสอบภายนอกจะต้อง

มีการอินเตอร์รัพท์เข้ามา เพื่อทำการเก็บข้อมูลที่กำลังประมวลผลอยู่ก่อน และเมื่อมีกระแสไฟฟ้าจ่ายให้เป็นปกติจึงค่อยนำข้อมูลนั้นมาประมวลผลต่อไป

บิท IDL (Idle mode)

เป็นการกำหนดโพรเซสเซอร์หยุดทำงานชั่วขณะ (Sleep) และจะกลับมาอยู่ในสภาพปกติอีกครั้งเมื่อทำการรีเซ็ททางฮาร์ดแวร์ หรือมีการอินเตอร์รัพท์อย่างใดอย่างหนึ่งเกิดขึ้น การทำงานในลักษณะนี้สามารถเกิดขึ้นได้เนื่องจากว่าสภาวะการหยุดทำงานชั่วขณะนั้น เป็นเพียงเพื่อห้ามไม่ให้มีสัญญาณนาฬิกาจ่ายให้ส่วนของโพรเซสเซอร์เท่านั้น ส่วนของวงจรการอินเทอร์รัพท์พอร์ตอนุกรมและวงจรจับเวลา ยังคงมีสัญญาณนาฬิกาอยู่เป็นปกติ

รีจิสเตอร์ IP,IE,TMOD,SCON

เป็นกลุ่มของรีจิสเตอร์ที่ทำหน้าที่กำหนดการควบคุม และการทำงานของการอินเตอร์รัพท์ต่างๆ ของ 8051

2.3.5 หน่วยความจำข้อมูลภายนอก

การใช้หน่วยความจำข้อมูลภายนอกเป็นวิธีการแก้ปัญหาอย่างหนึ่ง ในกรณีที่ทีความต้องการหน่วยความจำสำหรับการเก็บข้อมูลชั่วคราว หรือตัวแปรของโปรแกรมมากเกินกว่าขนาดของหน่วยความจำข้อมูลภายใน ซึ่งมีขนาดเพียง 128 หรือ 256 ไบต์ เท่านั้นบางครั้งการใช้หน่วยความจำข้อมูลภายนอกยังเหมาะกับงานประยุกต์บางอย่างที่จำเป็น ต้องมีการเก็บสำรองข้อมูลบางอย่างไว้ไม่ให้สูญหายแม้ว่าจะไม่มีการจ่ายไฟฟ้าให้แก่ระบบก็สามารถทำได้โดยการใช้ไอซีหน่วยความจำแรม พร้อมแบตเตอรี่สำรองประเภทลิเที่ยมหรือนิเกิลแคดเมี่ยมเป็นตัวเก็บข้อมูลเหล่านี้ไว้แทน อย่างไรก็ตามไม่ว่าสาเหตุของการนำไอซีหน่วยความจำภายนอกมาใช้งานจะเป็นอย่างไร จะมีผลทำให้พอร์ตอินพุต/เอาท์พุตของ 8051 ถูกนำไปใช้เพื่อติดต่อกับหน่วยความจำเหล่านี้แทนดังนั้นจึงจำเป็นต้องมีการใช้วงจรประกอบอื่นๆ เพื่อชดเชยความสามารถเหล่านั้นของ 8051 แทน

การเชื่อมต่อหน่วยความจำข้อมูลภายนอก

การเชื่อมต่อหน่วยความจำข้อมูลภายนอกเข้ากับระบบของไมโครคอนโทรลเลอร์ 8051 จะมีวิธีการเหมือนกับการเชื่อมต่อหน่วยความจำ EPROM โดยมีหลักการทำงานตามที่กล่าวมาคือ จะมีไอซีแลตช์ทำหน้าที่ค้างค่าแอสเดรส ให้กับอินพุตของหน่วยความจำแรมส่วนขา RD ,WR ก็จะทำ

การต่อโดยตรงกับ 8051 และถ้าต้องการต่อแรมหลายๆ ตัวก็ใช้การถอดรหัสแอสเดรสแบบเดียวกับการต่ออีพรอม

2.3.6 พอร์ตอินพุต / เอาท์พุตของ 8051

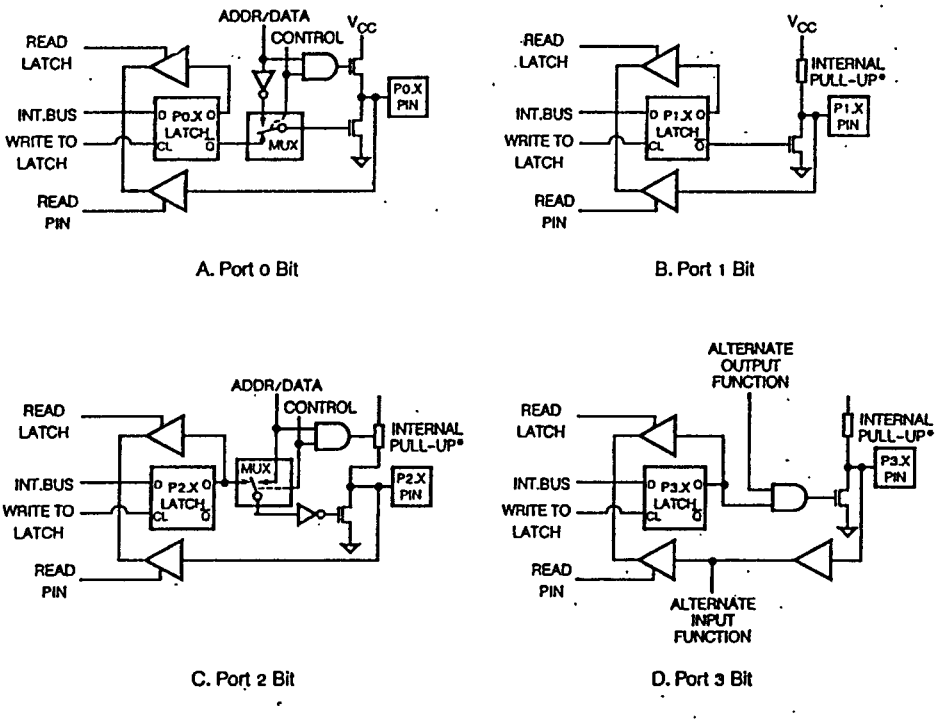
พอร์ตของ 8051 จะสามารถแบ่งออกได้เป็น 2 ลักษณะ คือ พอร์ตแบบขนานและพอร์ตแบบอนุกรม

พอร์ตแบบขนานของ 8051

8051 มีโครงสร้างของพอร์ตที่สามารถใช้งานแบบขนานได้จำนวนทั้งหมด 4 พอร์ต เรียกชื่อเรียงตามลำดับว่า P0 , P1 , P2 และ P3 และเป็นพอร์ตขนาด 8 บิททั้งหมด การใช้งานพอร์ตสามารถทำได้ทั้งในลักษณะของเส้นสัญญาณเดี่ยวๆ หรือของกลุ่มสัญญาณได้ นอกจากนี้พอร์ต 0 , 2 และ 3 ยังสามารถนำไปใช้งานอื่นๆ ที่ไม่ใช่เป็นอินพุตเอาท์พุตพอร์ตได้โดยพอร์ต 0 จะทำหน้าที่มัลติเพล็กซ์ ระหว่างบัสแอสเดรสไบต์ต่ำและบัสข้อมูลสำหรับการติดต่อวงจรประกอบร่วมกับข้อมูลบัสแอสเดรสไบต์สูงซึ่งจะส่งออกมาทางพอร์ต 2 สำหรับพอร์ต 3 นอกเหนือจากการนำไปใช้เป็นพอร์ตปกติสามารถนำไปใช้เป็นสัญญาณขอการอินเตอร์รัพท์ต่างๆ ซึ่งรวมถึงการสร้างสัญญาณควบคุมการอ่านและการเขียนเพื่อทำหน้าที่ควบคุมการอ่านและการเขียนหน่วยความจำข้อมูลภายนอกด้วย

โครงสร้างการทำงานของ พอร์ต 8051

จากลักษณะโครงสร้างของแต่ละบิทภายในพอร์ตทั้งหมดของ 8051 ซึ่งได้แสดงไว้ จะเห็นว่ามีความคล้ายคลึงกับลักษณะโครงสร้างที่เรียกว่า QUASI-BIDIRECTIONAL PORT ยกเว้นพอร์ต 0 ซึ่งเพียงแต่ไม่มีตัวต้านทานที่ทำหน้าที่ Pull-up สัญญาณไว้ภายในเท่านั้น วงจรประกอบอื่นภายในยังมีฟลิปฟลอปแบบ D ซึ่งมีผลทำให้สามารถแลทซ์หรือคางสภาวะของสัญญาณได้ นอกจากนี้ในส่วนเอาท์พุตของฟลิปฟลอปเฉพาะของพอร์ต 0 และ พอร์ต 2 จะมีโครงสร้างที่ทำหน้าที่คล้ายกับสวิทซ์เพิ่มเติมขึ้นเพื่อควบคุมให้เอาท์พุตนี้ต่อกับส่วนของทรานซิสเตอร์ในระหว่างที่ไม่ได้มีการทำงานในลักษณะของบัสแอสเดรสหรือบัสข้อมูล สำหรับบัฟเฟอร์จำนวน 2 ตัวของทุกบิทในพอร์ตนั้นจะมีการทำงานโดยแยกอิสระ โดยตัวที่อยู่ทางด้านบนจะยอมให้สัญญาณผ่านได้ก็ต่อเมื่อมีการอ่านข้อมูลที่ค้างไว้เท่านั้น ส่วนตัวที่อยู่ทางด้านล่างจะถูกใช้งานเฉพาะเมื่อมีการอ่านสถานะของขาสัญญาณเท่านั้น



รูปที่ 2.22 โครงสร้างแต่ละบิทภายในพอร์ตอินพุตเอาท์พุตของ8051

การใช้งานเป็นพอร์ตอินพุต

การใช้งานเป็นพอร์ตอินพุตจะต้องเริ่มด้วยการส่งข้อมูลที่มีค่าเป็น 1 ออกมาทางบิทของพอร์ตนั้นก่อนเป็นลำดับแรก เพื่อหยุดการทำงานของทรานซิสเตอร์ที่ทำหน้าที่ขับสัญญาณเอาท์พุตของบิทนั้น ทำให้ขาสัญญาณของบิทถูกต่อเข้ากับตัวต้านทานที่ทำหน้าที่ Pull-up ภายในซึ่งมีผลทำให้บิตนั้นๆ ของพอร์ต 1,2 และ 3 เป็นสภาวะลอจิกสูง ตัวต้านทานนี้มีค่าประมาณ 50 กิโลโอห์ม ซึ่งเป็นค่าที่สูงมาก และทำให้อุปกรณ์ภายนอกสามารถขับสัญญาณของพอร์ตเหล่านี้เป็นลอจิกต่ำได้ง่าย สำหรับบิทของพอร์ต 0 นั้น แม้ว่ามีหลักการทำงานที่คล้ายคลึงกับบิตอื่น แต่เนื่องจากการที่ไม่มีตัวต้านทานที่ทำหน้าที่ Pull-up ภายในไว้ ทำให้เมื่อทรานซิสเตอร์ที่ทำหน้าที่ขับสัญญาณเอาท์พุตเหล่านั้นหยุดการทำงาน ก็จะเป็นผลทำให้ขาสัญญาณนี้อยู่ในสภาวะอิมพิแดนซ์สูงแทน

การใช้งานเป็นพอร์ตเอาท์พุต

เมื่อมีการส่งข้อมูลที่มีค่าเป็น 0 ให้กับแต่ละบิตของพอร์ตทุกพอร์ต ข้อมูลเหล่านี้จะถูกส่งให้กับฟลิปฟลอปซึ่งจะทำหน่าที่ค้างสภาวะเหล่านี้ไว้ และมีผลทำให้ทรานซิสเตอร์ที่ทำหน้าที่ขับสัญญาณเอาท์พุตเหล่านั้นทำงาน ดังนั้นขาสัญญาณก็จะมีสภาวะเป็นลอจิกต่ำด้วย

ส่วนการส่งข้อมูลที่มีค่าเป็นหนึ่งออกมานั้น ในกรณีที่เป็นการทำงานของแต่ละบิต ของพอร์ต 1,2 หรือ 3 จะทำให้ทรานซิสเตอร์ที่ทำหน้าที่ขับสัญญาณเอาท์พุตนั้นหยุดการทำงาน มีผลทำให้ขาของสัญญาณเป็นลอจิกด้วยตัวต้านทานที่ Pull-up อยู่ภายในนั้น แต่สำหรับการทำงานของแต่ละบิตภายในพอร์ต 0 นั้น จะมีผลที่แตกต่างกันออกไป โดยขาสัญญาณจะมีสภาวะเป็นอิมพิแดนซ์สูงแทน เนื่องจากไม่มีตัวต้านทานอยู่ภายในเชื่อมอยู่นั่นเอง ดังนั้นการใช้งานพอร์ต 0 เป็นการเอาท์พุตข้อมูล จำเป็นต้องใช้ตัวต้านทานภานนอก Pull-up สัญญาณไว้กับลอจิกสูงแทน

ความสามารถอีกประการหนึ่งที่เกี่ยวกับพอร์ตอินพุตเอาท์พุตของ 8051 เป็นวิธีการอ่านค่าลอจิกจากพอร์ตซึ่งมีได้สองวิธี คือ การอ่านค่าลอจิกที่ขาสัญญาณ และการอ่านค่าลอจิกของการแลตซ์ที่ พอร์ต วิธีการอ่านค่าสัญญาณทั้ง 2 แบบจะช่วยให้ระบบสามารถทำงานได้ด้วยความถูกต้องมากยิ่งขึ้น

2.3.7 ลักษณะสมบัติของพอร์ตอินพุต/เอาท์พุต

จากที่กล่าวมาแล้วว่าพอร์ต 1,2 และ 3 ของ 8051 มีตัวต้านทานทำหน้าที่ Pull-up ขาสัญญาณไว้และมีค่าประมาณ 50 กิโลโอห์ม ซึ่งถือว่าเป็นค่าที่สูงมาก เป็นผลทำให้การเปลี่ยนแปลงระดับสัญญาณลอจิกจากสูงไปต่ำทำได้อย่างรวดเร็ว แต่ในกรณีตรงกันข้ามจะใช้เวลาในการเปลี่ยนแปลงระดับสัญญาณยาวนานกว่ามาก ทั้งนี้เนื่องจากว่ากระแสจะไหลผ่านตัวต้านทานเหล่านี้ได้น้อยมาก ดังนั้นในการแก้ปัญหาจึงได้มีการออกแบบตัวต้านทานเพิ่มขึ้นอีกหนึ่งตัวขนานไว้โดยมีค่าประมาณ 1 กิโลโอห์ม เรียกว่า Speed-up Resistor ซึ่งยอมให้กระแสไหลผ่านได้มากขึ้น ประมาณ 50-100 เท่า และจะมีการเชื่อมต่อตัวต้านทานที่เพิ่มขึ้นเฉพาะเมื่อมีการเปลี่ยนแปลงระดับสัญญาณจากลอจิกต่ำไปเป็นลอจิกสูงเท่านั้น

การเพิ่มจำนวนพอร์ตอินพุตเอาท์พุต

จากที่กล่าวมาแล้วว่า เมื่อมีการเพิ่มเติมหน่วยความจำภายนอกไม่ว่าจะเป็นหน่วยความจำโปรแกรมหรือหน่วยความจำข้อมูลให้กับระบบ ก็จำเป็นจะต้องนำพอร์ต 0 และพอร์ต 2 ทั้งหมดไปใช้งานในฐานะของบัสแอสเดรสและบัสข้อมูล ดังนั้นหากมีความต้องการที่จะใช้ พอร์ตมากกว่าที่เหลืออยู่ จะต้องทำการเพิ่มจำนวนพอร์ตขึ้นโดยการเพิ่มจำนวนพอร์ตสามารถทำได้ โดยการใช้ไอซีพื้นฐานอื่นๆ เช่นไอซี TTL เพิ่มเติมเข้าไปในการออกแบบ เพื่อทำงานเป็นพอร์ตให้กับระบบแทนเช่นไอซีบัฟเฟอร์สามสถานะ หรือไอซีแลตซ์เป็นต้นแต่การใช้ไอซีเหล่านี้มีข้อจำกัดหลายประการ เช่นในกรณีที่มีความจำเป็นต้องใช้พอร์ตหลายๆพอร์ต จะต้องใช้ไอซีจำนวนหลายตัว และไอซีแต่ละตัวก็จะถูกกำหนดหน้าที่ของมันว่าจะต้องเป็นพอร์ตอินพุตหรือพอร์ตเอาท์พุตได้เพียง

อย่างเดียว ถ้าหากต้องการเปลี่ยนเปลี่ยนหน้าที่ของพอร์ตเหล่านั้นก็จะต้องออกแบบวงจรใหม่ ดังนั้นจึงมีการใช้ไอซีที่ออกแบบมาเพื่อทำหน้าที่นี้โดยเฉพาะ โดยไอซีที่นิยมใช้กับพวก ไมโครคอนโทรลเลอร์หรือไมโครโพรเซสเซอร์ ได้แก่ไอซีของบริษัทอินเทลคือไอซี 8255

ลักษณะเบื้องต้น

8255 นั้นเป็นไอซี LSI ขนาด 40 ขา ซึ่ง 8255 นี้มีพอร์ตสำหรับทำหน้าที่รับส่งข้อมูลอยู่ด้วยกัน 3 พอร์ตคือ พอร์ต A,B และพอร์ต C โดยพอร์ต C นี้จะแบ่งออกเป็น 2 ส่วนคือ พอร์ต C บนกับพอร์ต C ล่าง นอกจากนี้ยังมี พอร์ตอีกพอร์ตหนึ่งที่ทำหน้าที่ควบคุมการทำงานของพอร์ต A,B,C โดยการรับคำสั่งมาจากซีพียู พอร์ตนี้เราเรียกว่าพอร์ตควบคุม พอร์ตนี้จะใช้งานก็ต่อเมื่อ ซีพียูต้องการกำหนดลักษณะการทำงานของพอร์ตต่างๆ หรือต้องการเปลี่ยนแปลงหลังจากที่กำหนดไว้เดิม

หน้าที่ของขาต่างๆ

CS (CHIP SELECT) ขานี้ใช้สำหรับรับสัญญาณจากภายนอกเข้ามาเพื่อเลือกว่าจะให้ 8255 ตัวนั้นทำงานหรือไม่ โดยถ้าขานี้ได้รับลอจิก 0 จะทำให้ตัว 8255 ต่อเข้ากับระบบบัสต่างๆ และพร้อมที่จะติดต่อกับ ซีพียูได้ แต่ถ้าได้รับลอจิก 1 ก็จะปลดตัวเองออกจากระบบ โดยการทำเป็นอิมพิแดนซ์สูง

RD (READ ENABLE) เป็นขาอินพุตที่จะรับสัญญาณจากซัพียูโดยที่ถ้าขานี้ได้รับลอจิก 0 และในขณะนั้นขา CS ก็ได้รับลอจิก 0 8255 จะทำการส่งข้อมูลจากพอร์ตที่ซีพียูต้องการติดต่อด้วยให้แก่ซีพียูทางบัสข้อมูล

WR (WRITE ENABLE) เป็นขาที่ทำหน้าที่ตรงกันข้ามกับขา RD คือเมื่อได้รับสัญญาณลอจิก 0 8255 จะรับข้อมูลจากบัสข้อมูลของซีพียู เพื่อส่งออกไปยังพอร์ตที่ซีพียูกำหนด

RESET เป็นขาที่ทำหน้าที่ รีเซ็ท 8255 เมื่อใดที่ 8255ได้รับสัญญาณรีเซ็ท มันจะกลับเข้าสู่โหมดอินพุตทุกๆ พอร์ต ขารีเซ็ทนี้จะใช้เมื่อต้องการเคลียร์สถานะต่างๆ ของ 8255

D0-D7 คือขาข้อมูลที่ใช้ในการติดต่อรับส่งข้อมูลกับซีพียู โดยจะต่อเข้ากับบัสข้อมูลของซีพียู เพื่อให้ซีพียูส่งข้อมูลออกไปยังพอร์ต หรือรับข้อมูลเข้าไปยังตัวซีพียู

A0-A1 คือขาแอสเดรสที่ใช้ในการเลือกพอร์ตที่ต้องการจะติดต่อซึ่งมีความเป็นไปได้ทั้งหมด 4 ค่าคือ

0 0 = PORT A

0 1 = PORT B

1 0 = PORT C

1 1 = CONTROL PORT

PA0-PA7 เป็นขาสัญญาณของพอร์ต A

PB0-PB7 เป็นขาสัญญาณของพอร์ต B

PC0-PC7 เป็นขาสัญญาณของพอร์ต C

2.3.8 การใช้งาน 8255

8255 นั้นแบ่งลักษณะการทำงานออกเป็น 3 โหมด

- โหมด 0 เป็นโหมดอินพุตหรือโหมดเอาท์พุตอย่างใดอย่างหนึ่ง ซึ่งทั้งสามพอร์ตสามารถทำงานได้ในโหมดนี้

- โหมด 1 เป็นโหมดอินพุตหรือเอาท์พุตอย่างใดอย่างหนึ่งเช่นกันแต่จะมีลักษณะการทำงานในลักษณะของ Hand Shaking โดยโหมดนี้จะทำงานได้เฉพาะพอร์ต A และพอร์ต B

- โหมด 2 เป็นโหมด Bidirectional คือเป็นไปได้ทั้งอินพุตและเอาท์พุตพอร์ตในเวลาเดียวกัน และทำงานแบบ Hand Shaking เช่นเดียวกับโหมด 1 แต่โหมดนี้จะใช้ได้เฉพาะพอร์ต A เท่านั้น

การกำหนดโหมดการทำงานของ 8255 นั้นทำได้โดย ซีพียูทำการส่งรหัสควบคุมผ่านทางบัสข้อมูลมายังพอร์ตควบคุม ของ 8255 รหัสควบคุมจะมีขนาด 1 ไบต์เรียกว่า Control byte และในแต่ละบิทของ Control byte จะมีความหมายเฉพาะของตัวเองซึ่งจะอธิบายได้ดังนี้

บิท D7 เป็นบิทที่แสดงว่า ไบต์นี้เป็นรหัสควบคุมที่จะมีผลต่อการกำหนดโหมดการทำงานของ 8255

บิท D6-D5 มีความหมายในการเลือกโหมดของพอร์ต A ซึ่งสามารจะทำงานได้ทั้ง 3 โหมด โดยแต่ละลอจิกจะมีความหมายดังนี้

0 0 = MODE 0

0 1 = MODE 1

1 X = MODE 2

บิท D4 ถ้าเป็นลอจิก 0 หมายถึงสั่งให้พอร์ต A เป็นเอาท์พุตพอร์ต และถ้าเป็น 1 พอร์ต A จะเป็นอินพุตพอร์ต

บิท D3 เป็นบิทที่กำหนดการทำงานของพอร์ต C บน ถ้าเป็น 0 จะเป็นเอาท์พุต และในทางกลับกันถ้าเป็น 1 จะเป็นอินพุตพอร์ต

บิท D2 เป็นบิทที่ใช้กำหนดโหมดการทำงานของพอร์ต B ถ้าเป็น 0 หมายถึงทำงานในโหมด 0 และถ้าเป็น 1 หมายถึงทำงานในโหมด 1

บิท D1 เป็นบิทที่ใช้กำหนดหน้าที่ของพอร์ต B ว่าจะใช้เป็นพอร์ตอินพุตหรือเอาท์พุต โดยถ้าบิทนี้เป็น 0 จะเป็นอินพุตพอร์ต และถ้าเป็น 1 ก็จะเป็นเอาท์พุตพอร์ต

บิท D0 เป็นบิทที่ใช้กำหนดการเป็นอินพุตหรือเอาท์พุตของพอร์ต C ล่าง ถ้าเป็น 0 จะเป็นเอาท์พุต และถ้าเป็น 1 จะเป็นอินพุต

2.3.9 เคาน์เตอร์และไทม์เมอร์ (COUNTER AND TIMER)

งานประยุกต์ที่ใช้ไมโครคอนโทลเลอร์เป็นจำนวนมากจำเป็นต้องใช้การนับเหตุการณ์จากภายนอกเช่นความถี่พัลซ์หรือการสร้างความแน่นอนของการหน่วงเวลาภายใน ระหว่างการทำงานของคอมพิวเตอร์ทั้ง 2 อย่างนี้สามารถทำให้สำเร็จได้โดยใช้เทคนิคซอฟแวร์ แต่รอบของโปรแกรมสำหรับการนับหรือเวลาเพื่อให้โพรเซสเซอร์ทำงานโดยฟังค์ชั่นอื่นไม่ถูกใช้งาน การปลอดปล่อยโดยใช้เคาน์เตอร์ (T0 , t1) ขนาด 16 บิทนับ และเป็นตัวใช้งานทั่วๆไป เคาน์เตอร์แต่ละตัวอาจถูกโปรแกรมให้นับพัลซ์ภายใน หรือเป็นไทม์เมอร์ หรือโปรแกรมให้นับพัลซ์ภายนอก

เคาน์เตอร์ถูกแบ่งเป็นรีจีสเตอร์ 8 บิท 2 ตัว เรียกว่าไทม์เมอร์ไบท์ต่ำและไทม์เมอร์ไบต์สูง การทำงานของไทม์เมอร์ทุกตัวถูกควบคุมโดยสถานะบิทใน TMOD , TCON และโปรแกรมคำสั่งบางตัว

TMOD จะใช้ตรวจสอบไทม์เมอร์ 2 ตัวเท่านั้น และสามารถพิจราณาเป็นรีจีสเตอร์ขนาด 4 บิท 2 ตัวต่อกันอยู่ โดยที่แต่ละตัวขะควบคุมไทม์เมอร์ 1 ตัว Tcon จะควบคุมบิทและแฟลกของไทม์เมอร์ในนิบเบิลบน และควบคุมบือและแฟลกของการอินเทอร์รัพท์จากภายนอกด้วยนิบเบิลล่าง

การอินเทอร์รัพท์ไทม์เมอร์และเคาน์เตอร์ (Interupt timer and counter)

เคาน์เตอร์จะอยู่รวมบนชิพเพื่อทำให้โพรเซสเซอร์ปลอดปล่อยจากงาน Timing และ Counting เมื่อโปรแกรมต้องการนับพัลซ์ภายในหรือเหตุการณ์ภายนอก จะต้องเก็บค่าๆ หนึ่งในเคาน์เตอร์ 1 ตัว จำนวนนี้จะแสดงการนับที่มากที่สุดซึ่งน้อยกว่าค่าที่ต้องการอยู่ 1 เคาน์เตอร์ เพิ่มจากจำนวนเริ่มต้นไปยังค่าที่มากที่สุด แต่ต่อจากนั้นจะนับจนเป็น 0 ซึ่งเป็นพัลซ์สุดท้ายและจะเซตไทม์เมอร์แฟลก สภาวะของแฟลกทดสอบได้ด้วยคำสั่งที่บอกให้โปรแกรมนับจนเสร็จหรือแฟล็กอาจใช้อินเทอร์รัพท์โปรแกรม

ไทม์มิ่ง(TIMING)

ถ้าเคาน์เตอร์ถูกตั้งเป็นไทม์เมอร์ก็จะนับความถี่นาฬิกาภายใน 8051 ออสซิลเลเตอร์หารด้วย 12 เช่น ความถี่คริสทัล 6 MHz จะได้ไทม์เมอร์ 500 KHz

สัญญาณนาฬิกาของออสซิลเลเตอร์จะเป็นพัลซ์มาสู่ไทม์เมอร์ บิท C/T ใน TMOD ต้องเป็น 0 บิท TRX ใน TCON ต้องเป็น 1 (ไทม์เมอร์ทำงาน) และบิทเกทใน TMOD ต้องเป็น 0 และขา INTRX ต้องเป็น 1 ในทางตรงกันข้ามเคาน์เตอร์ทำหน้าที่เคาน์เตอร์ได้โดยเกทไปที่เคาน์เตอร์โดยบิททำงาน (Runbit) และเกทบิท (Gate bit) หรือบิท INTX

การทำงานในโหมดไทม์เมอร์

การทำงานของไทม์เมอร์มีทั้งหมด 4 โหมด ซึ่งกำหนดโดยบิท M1 , M0 ใน TMOD

ไทม์เมอร์โหมด 0

การตั้งโหมดไทม์เมอร์ X(0,1) เป็น 00B ใน TMOD เป็นเหตุผลให้ใช้ THX เป็นเคาน์เตอร์ 8 บิท และ TLX เป็นเคาน์เตอร์ 5 บิท พัลซ์ที่เป็นอินพุตจะถูกหารด้วย 32 ใน TL ดังนั้น TH จะนับความถี่ที่ถูกหารด้วย 384 เช่น ความถี่ออสซิลเลเตอร์ 6MHz ให้ความถี่สุดท้ายใน TH เป็น 15625 Hz แฟลกไทม์เมอร์เมื่อถูกเซ็ทเมื่อ THX นับจาก FFh เป็น 00h หรือ 0.0164secสำหรับคริสทัล 6 MHz ถ้า THX เริ่มต้นที่ 00h

ไทม์เมอร์โหมด 1

ไทม์เมอร์ในโหมด 1 คล้ายกับไทม์เมอร์ในโหมด 0 ยกเว้น TLX จะใช้เต็ม 8 บิท เมื่อบิทโหมดเซ็ทเป็น 01b ใน TMOD แฟลกไทม์เมอร์จะเซ็ทในเวลา 0.1311sec เมื่อใช้คริสทัล 6 MHz

ไทม์เมอร์โหมด 2

การเซ็ทบิทโหมดเป็น 10b ใน TMOD จะใช้เฉพาะไทม์เมอร์ TLX เป็นตัวนับ 8 บิท TLX จะเก็บค่าที่โหลดไปใน TLX ทุกครั้งที่ TLX เกิดโอเวอร์โฟล์จาก FFh เป็น 000h แฟลกไทม์เมอร์จะเซ็ทเมื่อ TLX เกิดโอเวอร์โฟล์

ไทม์เมอร์โหมด 3

ไทม์เมอร์โหมด 0 และ 1 อาจจะโปรแกรมเป็นโหมด 0 , 1 , 2 อิสระจากกันในลักษณะคล้ายๆกันแต่ไม่สำหรับโหมด 3 ไทม์เมอร์จะไม่ทำงานเป็นอิสระจากกัน ไทม์เมอร์ 1 จะเป็นตัวทำให้หยุดนับบิทควบคุม TR1 และไทม์เมอร์แฟลก 1:TF1 จะถูกใช้โดยไทม์เมอร์ 0

ไทม์เมอร์ 0 จะแยกเป็นตัวนับขนาด 8 บิทออกจากกัน TL0 ถูกควบคุมโดยการจัดเรียงเกทในรูป 2.6.6 และเซ็ทไทม์เมอร์แฟลก :TF0 เมื่อเกิดโอเวอร์โฟล์จาก FFh ถึง 00h TH0 สัญญาณ

นาฬิกาของไทม์เมอร์ (หารจากออสซิลเลเตอร์ด้วย 12) ภายใต้การควบคุมของ TR1 เท่านั้นและเซ็ทแฟลก TF1 เมื่อเกิดการโอเวอร์โฟล์

ไทม์เมอร์ 1 อาจจะใช้ในโหมด 0 , 1 , 2 , ขณะที่ไทม์เมอร์ 0 มีข้อยกเว้นที่สำคัญคือการอินเทอร์รัพท์ไม่ได้เกิดจากไทม์เมอร์ 1 ขณะที่ไทม์เมอร์ 10 ใช้โอเวอร์โฟล์แฟลก TF1 การสวิทซ์ไทม์เมอร์ 1 ในโหมด 3 จะหยุดการนับและเก็บค่าในไทม์เมอร์ 1 ไทม์เมอร์สามารถใช้ Buad rate สำหรับพอร์ตอนุกรม หรือโหมด 0 , 1 , 2 ซึ่งไม่เกิดขึ้นกับการอินเตอร์รัพท์

เคาท์ติ้ง (Counting)

Timing และ Counting มีข้อแตกต่างกันสิ่งเดียวคือ แหล่งพัลซ์นาฬิกาของเคาท์เตอร์เมื่อใช้เป็นไทม์เมอร์ พัลซ์นาฬิกามาจากวงจรออสซิลเลเตอร์ผ่านวงจรหาร 12 เมื่อใช้เป็นเคาน์เตอร์ ขา T0 จะเป็นพัลซ์ให้กับเคาน์เตอร์ 0 และขา T1 แก่เคาน์เตอร์ 1 บิท C/T ใน TMOD ต้องเซ็ทเป็น 1 เพื่อให้พัลซ์จากขา TX แก่วงจรควบคุมดังรูป

พัลซ์อินพุตบน TX จะถูกสุ่มระหว่าง P2 ของสถานะที่ 5 ของทุก Machine cycle การเปลี่ยนอินพุตจาก 1 เป็น 0 ระหว่างการสุ่มจะเพิ่มค่าเคาน์เตอร์ทุกสถานะ 1 และ 0 ของพัลซ์อินพุตต้องคงที่อย่างน้อยที่สุด 1 Machine cycle เพื่อมั่นใจว่าการนับเชื่อใจได้ เนื่องจากมี 25 พัลซ์ ความถี่อินพุตสูงสุดยังสามารถนับได้อย่างแน่นอนคือความถี่ของออสซิลเลเตอร์หารด้วย 24 สำหรับคริสทัล 6 MHz ซึ่งให้ความถี่สูงสุดออกมา 250 KHz

2.3.10 อินพุตเอาท์พุตข้อมูลอนุกรม

คอมพิวเตอร์ต้องสามารถติดต่อกับคอมพิวเตอร์อื่นๆ ในระบบมัลติโพรเซสเซอร์สมัยใหม่ การติดต่อที่มีประสิทธิภาพทางหนึ่งคือการส่ง และการรับบิทข้อมูลและอนุกรม 8051 มีวงจรติดต่อข้อมูลแบบอนุกรมโดยใช้รีจิสเตอร์ SBUF เก็บข้อมูล SCON ควบคุมการสื่อสาร PCON ควบคุมอัตราข้อมูล และขา RXD และ TXD ต่อกับเครือข่ายข้อมูลอนุกรม

SBUF มีอยู่ 2 รีจิสเตอร์อันหนึ่งใช้เขียนและเก็บข้อมูลที่จะส่งออกของ 8051 ผ่าน TXD อีกตัวใช้อ่าน และเก็บข้อมูลที่ได้รับจจากภายนอกผ่าน RXD รีจิสเตอร์ทั้ง 2 นี้ มีแอสเดรส 99h

มีโหมดที่โปรแกรมได้ทั้งหมด 4 โหมด สำหรับการสื่อสารข้อมูลแบบอนุกรมซึ่งเลือกโดยบิท SMX ใน SCON ส่วน Baud rate กำหนดโดยโหมดที่ใช้

การอินเทอร์รัพท์ข้อมูลอนุกรม

การสื่อสารข้อมูลอนุกรมเป็นขบวนการที่ล่าช้า ใช้เวลาหลาย msec ต่อไบท์ข้อมูล แฟลกข้อมูลอนุกรมอยู่ใน SCON เพื่อช่วยให้การส่งและการรับมีประสิทธิภาพ สังเกตุด้วยว่าการส่งข้อมูลอยู่ภายใต้การควบคุมของโปรแกรมแต่การรับข้อมูลไม่สามารถเดาได้ และรับที่เวลาใดก็ได้ซึ่งอยู่นอกเหนือการควบคุมด้วยโปรแกรม

แฟลกข้อมูลอนุกรมใน SCON TI และ RI จะเซ็ทเมื่อไบท์ข้อมูลถูกส่ง (TI) หรือรับ (RI) แฟลกจะ OR กันเพื่อสร้างอินเทอร์รัพท์โปรแกรม โปรแกรมต้องอ่านแฟลกเหล่านี้เพื่อหาสาเหตุการอินเทอร์รัพท์จะทำการเคลียร์แฟลก ซึ่งไม่เหมือนแฟลกไทม์เมอร์ที่ต้องเคลียร์ตัวเอง ผลลัพท์อันนี้ขึ้นกับโปรแกรมเมอร์ที่จะเขียนรูทีน ให้จัดการกับแฟลกข้อมูลอนุกรม

การส่งข้อมูล

การส่งบิทข้อมูลอนุกรม เริ่มต้นโดยข้อมูลถูกเขียนลง SBUF TI เว็ทเป็น 1 เมื่อข้อมูลส่งเรียบร้อยแล้วและ SBUF ว่างและข้อมูลอีกไบต์สามารถส่งได้ถ้าโปรแกรมผิดพลาดในการแฟลก TI และเขียนเกิน SBUF ขณะที่ข้อมูลไบต์ก่อนกำลังส่งอยู่ ผลคือไม่สามารถคาดเดาสิ่งที่เกิดขึ้นได้

การรับข้อมูล

เริ่มต้นด้วย ถ้าบิทรับข้อมูล (REN) ใน SCON เซ็ทเป็น 1 ทุกโหมดยกเว้นโหมด 0 RI ต้องเคลียร์เป็น 0 แฟลกอินเทอร์รัพท์การรับข้อมูล RI จะเซ็ทหลังจากรับข้อมูลแล้วทุกโหมด การเซ็ท REN จะเป็นการควบคุมโปรแกรมโดยตรงเท่านั้นซึ่งจะไม่รับข้อมูลที่ไม่ต้องการ RI ต้องเป็น 0 ในโหมด 0 เพื่อป้องกันการรับข้อมูลใหม่จนกว่าโปรแกรมจะรับข้อมูลเก่าและรีเซ็ท RI

การรับสามารถเริ่มต้นในโหมด 1 , 2 , 3 ถ้า RI เซ็ทเมื่อบิทข้อมูลอนุกรมเริ่มปรากฏ RI ต้องรีเซ็ทโดยโปรแกรมก่อนที่บิทสุดท้ายจะรับ มิฉะนั้นข้อมูลจะสูญหาย ข้อมูลภายในไม่สามารถส่งไปSBUF จนกว่าข้อมูลสุดท้ายจะได้รับเพื่อให้การส่งข้อมูลนั้นอ่านจาก SBUF ขณะที่มีการรับข้อมูลใหม่

โหมดการส่งข้อมูลอนุกรม

ผู้ออกแบบ 8051 ได้รวบรวมโหมดในการส่งข้อมูลอนุกรมไว้ 4 โหมด ทำให้การสื่อสารข้อมูลสามารถทำได้หลายทางและมี Baud rate หลายขนาด โหมดจะถูกเลือกโดยโปรแกรมเมอร์โดยการเซ็ทบิทโหมด SM0 และ SM1 ใน SCON Baud rate จะคงที่ในโหมด 0 และสามารถเปลี่ยนแปลงได้เมื่อใช้ไทม์เมอร์ 1 และบิทที่เปลี่ยน Baud rate อนุกรม (SMOD) ซึ่งอยู่ใน PCON สำหรับโหมด 1 , 2 , 3

ข้อมูลอนุกรมในโหมด 0 โหมดการเลื่อนรีจิสเตอร์

การเซ็ทบิท SM0 และ SM1 ใน SCON เป็น 00b ทำให้ SBUF ทำการรับหรือส่งข้อมูล 8 บิทโดยใช้ขา RXD ทั้ง 2 หน้าที่ขา TXD จะต่อกับแหล่งกำเนิดพัลซ์ภายใน ซึ่งจะให้พัลซ์ที่เลื่อนแก่วงจรภายนอก ความถี่ที่เลื่อนหรือ Buad rate จะคงที่ (เท่ากับความี่ออสซิลเลเตอร์ / 12) อัตราขนาดนี้จะใช้เป็นไทม์เมอร์เมื่อโครงสร้างเป็นไทม์เมอร์ สัญญาณนาฬิกาที่เลื่อน TXD เป็น Square wave ซึ่งเป็น 0 ใน สภาวะ S3 , S4 , S5 ของ Machine cycle และเป็น 1 สำหรับสภาวะ S6 , S1 , S2 จากรูปแสดง Timing ของการส่งข้อมูลรีจิสเตอร์ที่เลื่อนข้อมูลโหมด 0 เมื่อข้อมูลส่งออกมาทาง RXD ข้อมูลเปลี่ยนที่ขอบขาลงของ S6P2 หรือ 1 พัลซ์นาฬิกา หลังจากขอบขาขึ้นของสัญญาณนาฬิการที่เลื่อน TRX ออก ผู้ออกกแบบระบบต้องออกแบบวงจรภายนอก ซึ่งรับข้อมูลที่ส่งออกมาเพื่อให้ข้อมูลที่รับเชื่อถือได้

ข้อมูลที่รับเข้ามาทางขา RXD ควรจะพร้อมกับสัญญาณนาฬิกาที่เลื่อนที่เกิดที่ TXD ข้อมูลถูกสุ่มที่ขอบขาลงของ S5P2 และเลื่อนไปยัง SBUF ที่ขอบขาขึ้นของสัญญาณนาฬิการที่เลื่อน

โหมด 0 ไม่ได้ตั้งใจให้ใช้สำหรับการสื่อสารข้อมูลระหว่างคอมพิวเตอร์ แต่เป็นวิธีที่จะได้ข้อมูลอนุกรมที่ความเร็วสูง โดยใช้ Discrete logic เพื่อให้ได้อัตราข้อมูลสูง Buad rate ที่ใช้ในโหมด 0 จะสูงกว่ามาตราฐานมาก เช่น คริสทัล 6 MHz ได้อัตราเลื่อน 500KHz

2.3.11 ข้อมูลอนุกรม 1-UART มาตราฐาน

เมื่อ SM0 และ SM1 เซ็ทเป็น 01b SBUF จะเป็นตัวรับและตัวส่ง 10 บิท โดยจะรับและส่งข้อมูลในเวลาเดียวกัน ขา RXD จะรับข้อมูลทั้งหมด และขา TXD จะส่งข้อมูลทั้งหมด

Buad rate กำหนดได้ด้วย

f_{buad} = (2^{SMOD} * ความถี่ออสซิลเลเตอร์)/64ข้อมูลที่ส่งเป็น Start bit 1 บิท ข้อมูล 8 บิท สตอปบิท 1 บิท แฟลกอินเทอร์รัพท์ TI จะเซ็ททันทีที่ 10 บิทถูกส่งแล้ว ระยะของแต่ละบิทคือส่วนกลับของความถี่ของ Buad rate แต่ละบิทจะเป็น 1 หรือ 0 ตลอดช่วง

ข้อมูลที่รับจะมีลำดับเหมือนเดิมการรับจะถูกทริกที่ขอบขาลงของ Start bit และต่อไปเรื่อยๆ ถ้าสต็อปบิท เป็น 0 ครึ่งหนึ่งของ Start bit นี่เป็นการวัดที่มีการรบกวนน้อยถ้าวงจรรับถูกทริกโดยสัญญาณรบกวนบนสายส่ง การตรวจสอบสภาวะ 0 หลังจากครึ่งบิทจะเป็นการจำกัดการรับข้อมูลที่ผิดพลาด

บิทข้อมูลที่เลื่อนเข้าตัวรับที่โปรแกรม Buad rate ไว้และ Word ข้อมูลจะถูกส่งไปยัง SBUF ถ้าเงื่อนไขตามนี้เป็นจริง Ri = 0, SM = 0 หรือ Stop bit = 0 ,RI = 1 เป็นการบอกว่าโปรแกรมได้อ่านไบต์ข้อมูลมาก่อนและพร้อมรับข้อมูลต่อไป โดยปกติ Stop bit จะทำให้ส่งข้อมูล

ไปยัง SBUF ได้สมบูรณ์ที่สภาวะ SM2 โดยที่ SM2 = 0 ทำให้สามารถรับไบต์และสตอปบิท ซึ่งเป็นข้อจำกัดในโหมดนี้ แต่มีประโยชน์มากในโหมด 2 และ 3 ถ้า SM2 = 1 ทำให้รับเฉพาะ สตอปบิท ที่ถูกต้องเท่านั้น และป้องกันการรบกวน

ใน 10 บิทนี้ ที่ตำแหน่งสุดท้ายของการรับ เป็นการชี้ว่าข้อมูลที่รับมาก่อนยังไม่ถูกโปรแกรมอ่าน หรือถ้าเงื่อนไขอื่นไม่จริง ข้อมูลใหม่จะไม่ถูกโหลด และจะสูญหายไป

Buad rate โหมด 1

ไทม์เมอร์ 1 ถูกใช้สร้าง Buad ในโหมด 1 โดยใช้โอเวอร์โฟล์แฟลกของไทม์เมอร์ เพื่อกำหนดความถี่ Buad rate ถ้าไทม์เมอร์ 1 ใช้ในโหมดเป็นไทม์เมอร์ 8 บิทออโตโหลด (Autoload) จะสร้าง Baud rate ได้

$f_{baud} = (2^{SMOD} * \text{ความถี่ออสซิลเลเตอร์})/((32*12*[256-TH1])$

SMOD เป็นบิทควบคุมใน PCON และอาจเป็น 0 หรือ 1

ถ้าไทม์เมอร์ 1 ไม่ได้ทำงานในโหมด 2 Buaad rate จะเป็น

$f_{buad} = (2^{SMOD} * \text{Timer 1 overflow flag})/32$

และไทม์เมอร์ 1 สามารถใช้สัญญาณนาฬิกาภายในหรือเป็นเคาน์เตอร์ซึ่งรับพัลซ์นาฬิกาจากภายนอกผ่านขา T1

ความถี่ออสซิลเลเตอร์ที่เลือกจะช่วยสร้าง Buad rate ทั้งแบบมาตราฐานและไม่มาตราฐาน ถ้าต้องการ Buad rate มาตราฐาน คริสทัล 11.0592 MHz ควรใช้ ซึ่งจะใช้อัตรามาตราฐาน 9600Hz และ TH1 จะมีค่าดังนี้

$TH1 = 256-[(2^{6}*11.0592*10^{6})/(32*12*9600)] = 0FDh = 253.0000d$

ถ้า SMOD ถูกเคลียร์เป็น 0

ข้อมูลอนุกรมโหมด 2 โหมดมัลติโพรเซสเซอร์

โหมด 2 คล้ายกับโหมด 1 เว้นแต่ว่ามีการส่ง 11 บิท คือ Start bit 1 บิท , ข้อมูล 9 บิท , สตอปบิท 1 บิทดังรูป บิทข้อมูลที่ 9 ได้จาก TB8 ใน SCON ระหว่างการส่งและเก็บในบิท RB8 ของ SCON เมื่อรับข้อมูลทั้ง Start bit และ สตอปบิทจะทิ้งไป

ในกรณีโหมด 0 Buad rate จะมากกว่ามาตรฐานมาก อัตราข้อมูลที่สูงนี้เป็นที่ต้องการใน Application หลายๆ มัลติโพรเซสเซอร์ข้อมูลสามารถรวบรวมได้อย่างรวดเร็วจากเครือข่ายของไมโครคอนโทรลเลอร์ที่ใช้สื่อสาร ถ้าใช้ Buad rtae สูง

เงื่อนไขการเซ็ท RI ในโหมด 1RI ต้องเป็น 0 ก่อนที่จะรับบิทสุดท้าย และ SM2 ต้องเป็น 0 หรือข้อมูลบิทที่ 9 ต้องเป็น 1 การเซ็ท RI ขึ้นกับสภาวะของ SM2 ในการรับ 8051 และสภาวะ 9 ซึ่งทำให้ Multiprocessing เป็นไปได้โดยให้ตัวรับถูกอินเทอร์รัพท์โดยข้อมูลบางตัว ในขณะที่ตัวรับอื่นๆ ไม่สนใจข้อมูลนี้ เฉพาะ 8051 เท่านั้นที่ SM2 เซ็ทเป็น 0 จะถูกอินเทอร์รัพท์โดยข้อมูลที่รับ ซึ่งข้อมูลบิทที่ 9 เซ็ทเป็น 1 จะไม่ถูกอินเทอร์รัพท์โดยข้อมูล พร้อมกับข้อมูลบิท 9 เป็น 0 ตัวรับทั้งหมดจะถูกอินเทอร์รัพท์โดยข้อมูล และข้อมูลบิทที่ 9 เซ็ทเป็น 1 ซึ่งสภาวะของ SM2 จะไม่ขัดขวางการรับข้อมูล

รายละเอียดเหล่านี้จะให้คอมพิวเตอร์ที่ทำการสั่ง ติดต่อกับคอมพิวเตอร์ตัวรับที่ถูกเลือกโดยไม่มีการอินเตอร์รัพท์คอมพิวเตอร์ตัวรับอื่นๆ คอมพิวเตอร์ตัวรับอาจถูกสั่งเป็นตัวรับ ตัวส่งหรือๆไม่สนใจคำสั่ง โดยโค้ดไบต์ที่ส่งพร้อมกับข้อมูลบิท 9 เป็น 1 1 ในข้อมูลบิท 9 จะอินเตอร์รัพท์ตัวรับทุกตัว คำสั่งนี้จะถูกโปรแกรมให้มีผลต่อโค้ดไบต์เพื่อโปรแกรมสภาวะของ SM2 ใน SCON ตัวรับที่ถูกเลือกจะทำให้บิท 9 เซ็ทเป็น 0 ขณะที่ตัวรับอื่นๆ จะไม่สนใจตัวส่งสามารถเปลี่ยนตัวรับโดยส่งบิทที่เซ็ทเป็น 1 ซึ่งจะสั่งตัวรับใหม่ให้เซ็ท SM2 เป็น 0 ขณะที่ตัวอื่นๆ จะเซ็ท SM2 เป็น 1

ข้อมูลในอนุกรมโหมด 2

โหมด 3 เหมือนกับโหมด 2 ยกเว้น Buad rate ซึ่งเหมือนโหมด 1

2.3.12 อินเทอร์รัพท์

โปรแกรมคอมพิวเตอร์มีเพียง 2 ทางเท่านั้น ที่จะหาเงื่อนไขที่เป็นจริงของวงจรภายในและภายนอก วิธีแรก คือใช้คำสั่งซอฟต์แวร์ กระโดดบนสภาวะแฟลกและพอร์ตพิน วิธี 2 คือ สัญญาณทางฮาร์ดแวร์ที่เรียกว่าอินเตอร์รัพท์ ซึ่งทำให้โปรแกรมไปเรียกซับรูทีน (Sub routine) เทคนิคของซอฟต์แวร์ใช้เวลาของโพรเซสเซอร์ ซึ่งสามารถไปใช้งานอื่นๆ อินเตอร์รัพท์จะใช้เวลาของโพรเซสเซอร์เมื่อถูกทำเมื่อต้องการใช้โปรแกรม การประยุกต์ใช้งานเกือบทั้งหมดของไมโครคอนโทลเลอร์เกี่ยวข้องกับเหตุการณ์ ที่เร็วพอที่จะควบคุมภาวะแวดล้อมที่สร้างเหตุการณ์

อินเทอร์รัพท์อาจสร้างโดยการทำงานของชิพภายในหรือโดยแหล่งภายนอก อินเทอร์รัพท์บางตัวทำให้ 8051 ทำฮาร์ดแวร์ให้เรียกซับรูทีน มีตำแหน่งที่กำหนดมาก่อน(โดยผู้ออกแบบ 8051) ในโปรแกรมเมมโมรี่

มีอินเทอร์รัพท์ 5 ชนิดใน 8051 3 ชนิดในนี้จะถูกสร้างโดยอัตโนมัติ โดยการทำงานภายในของไทม์เมอร์แฟลก 0 ไทม์เมอร์แฟลก 1 และอินเทอร์รัพท์พอร์ตอนุกรม (RI หรือ TI) อิน

เทอร์รัพท์อีก 2 ตัวถูกทริกโดยสัญญาณภายนอกโดยวงจรที่ต่อกับขา INT0 และ INT1 (ขาของพอร์ต P3.2 และ P3.3) หน้าที่ของอินเทอร์รัพท์ จะอยู่ภายใต้การควบคุมของโปรแกรม โปรแกรมเมอร์สามารถเปลี่ยนบิทควบคุมในรีจิสเตอร์อินเทอร์รัพท์ (IE) ,Interupt priority register (IO) และ Timer control register (TCON) โปรแกรมสามารถสกัดกั้นอินเทอร์รัพท์ทั้งหมดหรือบางตัวจากโปรแกรมโดยการเซ็ทหรือเคลียร์บิทในรีจิสเตอร์เหล่านี้

หลังจากอินเทอร์รับถูกจัดการโดยซับรูทีนซึ่งทำได้โดยโปรแกรมเมอร์ ในโปรแกรมเมมโมรี่ โปรแกรมอินเทอร์รัพท์ต้องทำงานที่คำสั่งที่เกิดอินเทอร์รัพท์ โปรแกรมถูกทำโดยเก็บค่า PC ไปบนสแตกในแรมก่อนเปลี่ยน PC เป็นแอสเดรสของอินเทอร์รัพท์ในรอม ค่า PC จะได้จากสแตกหลังจากคำสั่ง RETI ได้ทำแล้วที่ส่วนท้ายของซับรูทีน

อินเทอร์รัพท์แฟลกไทม์เมอร์

เมื่อไทม์เมอร์/เคาน์เตอร์ เกิดโอเวอร์โฟล์ ผลคือไทม์เมอร์แฟลก (TF0,TF1) จะเซ็ทเป็น 1 แฟลกถูกเคลียร์เป็น 0 เมื่ออินเทอร์รัพท์ทำให้โปรแกรมเรียกซับรูทีนของไทม์เมอร์ในเมมโมรี่

อินเทอร์รัพท์พอร์ตอนุกรม

เมื่อรับข้อมูลแล้ว บิทอินเทอร์รัพท์ (RI) ต้องเซ็ทเป็น 1 ใน SCON เมื่อข้อมูลส่งเรียบร้อยแล้วบิทอินเทอร์รัพท์ (TI) ต้องเซ็ทใน SCON และจะนำมา OR กันเพื่อหาตัวอินเทอร์รัพท์แก่โปรเซสเซอรี ซึ่งเป็นการอินเทอร์รัพท์พอร์ตอนุกรมบิทเหล่านี้จะไม่ถูกเคลียร์เมื่อการเรียกโปรแกรมอินเทอร์รัพท์ถูกทำโดยโพรเซสเซอร์โปรแกรมที่จัดการสื่อสารข้อมูลอนุกรมต้องรีเซ็ท RI หรือ TI เพื่อจะทำข้อมูลถัดไป

การอินเทอร์รัพท์ภายนอก

ขา INT0 และ INT1 จะใช้โดยวงจรภายนอก การอินพุตบนขาเหล่านี้ต้องเซ็ทอินเทอร์รัพท์แฟลก IE0 และ IE1 ใน SCON เป็น 1 โดยวิธี 2 วิธี แฟลก IEX อาจเซ็ทเมื่อขา INTX เป็น 0 หรือแฟลกอาจเซ็ทเมื่อมีการเปลี่ยนแปลงจาก 1 เป็น 0 บนขา INTX บิท IT0 , IT1 จะโปรแกรมขา INTX สำหรับการอินเทอร์รัพท์ที่ 0 เมื่อเซ็ทเป็น 0 และโปรแกรมขา INTX สำหรับอินเทอร์รัพท์ที่การเปลี่ยนลอจิกเมื่อเซ็ทเป็น 1

แฟลก IEX จะรีเซ็ทเมื่อการอินเทอร์รัพท์ที่การเปลี่ยนลอจิก พบโดยโปรแกรมและเข้าถึงซับรูทีน นี่เป็นผลของนักออกแบบระบบและโปรแกรมเมอร์ที่ต้องรีเซ็ทการอินเทอร์รัพท์ภายนอก

ว่าจะทำที่ลอจิกไหนเมื่อใช้โปรแกรม วงจรภายนอกต้องเป็น 1 ก่อนที่ RETI จะถูกจัดความผิดพลาดในเรื่องนี้จะทำให้เกิดการอินเทอร์รัพท์ทันทีหลังจาก RETI ด้วยแหล่งอินเทอร์รัพท์เดียวกัน

การรีเซ็ท (Reset)

การรีเซต สามารถมองเป็นการอินเทอร์รัพท์สูงสุด เพราะโปรแกรมไม่สามารถสกัดกั้นได้ การอินเทอร์รัพท์ชนิดนี้บ่อยครั้งเราเรียกว่า Nonmaskable เนื่องจากไม่มีบิทใดที่สามารถหยุดมันได้ ซึ่งไม่เหมือนกับการอินเทอร์รัพท์แบบอื่น PC จะไม่เก็บค่าโปรแกรมครั้งหลังสุด รีเซตเป็นคำสั่งที่สมบูรณ์ในการกระโดดไปที่ 0000h และเริ่มทำงานที่จุดนี้

เมื่อลอจิก 1 ใช้ที่ขา RST 8051 จะอยู่ในภาวะรีเซต หลังจากขา RST เป็น 0 รีจิสเตอร์ภายในจะมีค่าดังต่อไปนี้

REGISTER	VALUE(HEX)
PC	0000
DPTR	0000
A	00
B	00
SP	07
SPW	00
P0-3	FF
IP	XXX00000b
IE	0XX00000b
TCON	00
TMOD	00
TH0	00
TL0	00
TH1	00
TL1	00
SCON	00
SBUF	XX
PCON	0XXXXXXXb

แรมภายในจะไม่เปลี่ยนบนโดยการรีเซ็ตแต่สถานะของแรมภายในเมื่อเปิดเครื่องครั้งแรก จะสุ่ม และแบงค์ 0 จะถูกเลือกและทุกบิทใน PSW เป็น 0

2.3.13 การควบคุมการอินเทอร์รัพท์

ความสามารถ / ไม่สามารถอินเทอร์รัพท์

บิทในรีจิสเตอร์ EI ต้องเซ็ตเป็น 1 ถ้าแหล่งอินเท่อร์รัพท์ถูกใช้ และเซ็ตเป็น 00 เมื่อไม่ใช้ บิท EA ซึ่งเป็นบิทที่สำคัญซึ่งสามารถ / ไม่สามารถอินเทอร์รัพท์ทุกแหล่งโปรแกรมต้องสามารถยับยั้งอินเทอร์รัพท์ทั้งหมดหรือบางตัว เพื่อให้งานที่สำคัญได้ทำเสร็จรีจิสเตอร์ EI จะเก็บค่าบิทที่โปรแกรมได้ เพื่อทำให้อินเทอร์รัพท์ได้ตามต้องการ และเมื่อเลือกอินเทอร์รัพท์แล้วแหล่งอินเทอร์รัพท์แต่ละแหล่งอาจถูกเลือกหรือไม่ก็ได้

บ่อยครั้งที่ต้องการอินเทอร์รัพท์ที่สำคัญกว่าโดยทันที บิทของรีจิสเตอร์ IP อาจเซ็ตโดยโปรแกรมเพื่อกำหนดความสำคัญของแต่ละแหล่งอินเทอร์รัพท์ เพื่อให้อินเทอรรัพท์ที่สำคัญกว่าเกิดก่อน เมื่อมีการอินเทอร์รัพท์เกิดขึ้นพร้อมกันตั้งแต่ 2 แหล่งขึ้นไป

ความสำคัญของการอินเทอร์รัพท์

บิทในIP ถูกกำหนดถ้าอินเทอร์รัพท์ต้องการลำดับความสำคัญ บิทจะเซ็ตเป็น 1 เมื่อเป็นการอินเทอร์รัพท์ที่มีความสำคัญสูง และเป็น 0 เมื่ออินเทอร์รัพท์มีความสำคัญต่ำ อินเทอร์รัพท์ที่มีความสำคัญสูงไม่อาจถูกอินเทอร์รัพท์โดยตัวที่มีความสำคัญต่ำกว่า จนกว่าตัวที่สำคัญสูงกว่าจะทำเสร็จ

ถ้าอินเทอร์รัพท์ 2 ตัวที่มีความสำคัญเท่ากันเกิดพร้อมกันจะมีการทำตามลำดับดังนี้

1. IE0
2. TF0
3. IE1
4. TF1
5. Serial = or TI

การอินเทอร์รัพท์อนุกรมสามารถให้ลำดับที่สำคัญกว่าทำก่อนโดยเซ็ตบิท PS ใน IP เป็น 1 และบิทอื่นๆเป็น 0

จุดหมายการอินเทอร์รัพท์

แต่ละอินเทอร์รัพท์ทำให้โปรแกรมทำฮาร์ดแวร์ ให้เรียกแอสเดรสที่ตรวจสอบแล้วในโปรแกรมเมมโมรี่นี่เป็นสิ่งที่โปรแกรมเมอร์ต้องวางรูทีนในแอสเดรสที่เกิดการอินเทอร์รัพท์

อินเทอร์รัพท์จะเก็บค่าใน PC เมื่อทำการอินเทอร์รัพท์ จะเก็บค่าในสแต็ก และเรียกตำแหน่งที่ต้องการดังตารางนี้

INTERUPT	ADDRESS
IE0	0003
TF0	000B
IE1	0013
TF1	001B
SERIAL	0023

คำสั่ง RETI จะอยู่ที่ท้ายของรูทีน และรีเซ็ตลอจิกของการอินเทอร์รัพท์ เพื่อให้อินเทอร์รัพท์อีกตัวสามารถรับการอินเทอร์รัพท์ได้ อินเทอร์รัพท์อาจเกิดขึ้นแต่ไม่ถูกทำ เนื่องจากสภาวะสกัดกั้นมีอยู่ตลอดการอินเทอร์รัพท์ นี่เป็นความต้องการเบื้องต้นในการใช้อินเทอร์รัพท์ INTX ที่ทำที่ลอจิก

อินเทอร์รัพท์ด้วยซอฟต์แวร์

เมื่อแฟลกอินเทอร์รัพท์เกิดเซ็ตเป็น 1 โดยวิธีใดๆ อินเทอร์รัพท์จะเกิดขึ้นไม่มีการขัดขวาง วิธีนี้โปรแกรมสามารถอินเทอร์รัพท์แหล่งใดๆ ด้วยตัวเอง โดยเซ็ตแฟลกอินเทอร์รัพท์ที่ต้องการเป็น 1 โดยใช้โปรแกรมคำสั่ง

2.4 วิธีติดตั้งจานรับสัญญาณดาวเทียม (Satellite dish installation)

ในการติดตั้งจานรับสัญญาณดาวเทียมควรคำนึงถึงหัวข้อใหญ่ๆดังต่อไปนี้คือ

2.4.1 การสำรวจพื้นที่

เนื่องจากสัญญาณโทรทัศน์ที่ส่งมาจากดาวเทียมค้างฟ้าเป็นสัญญาณไมโครเวฟซึ่งเดินทางจากจุดหนึ่งไปสู่อีกจุดหนึ่งเป็นลักษณะเส้นตรง (Line of sight) และเนื่องจากดาวเทียมที่ใช้งานประเภทนี้ จะถูกส่งให้ไปลอยอยู่ในวงโคจรจีโอสเตชันนารี (Geostationary orbit) ซึ่งตำแหน่งของมันจะอยู่เหนือตลอดเส้นศูนย์สูตรของโลก (Earth's equator) ดังนั้นหากเราอยู่บนพื้นที่ที่เหนือเส้นศูนย์สูตร อย่างเช่นประเทศไทย เราก็ต้องหันหน้าของจานลงไปในทางทิศใต้ และถ้าเป็นประเทศที่ตั้งอยู่ใต้เส้นศูนย์สูตรลงไป อย่างเช่นประเทศออสเตรเลียก็จะต้องหันหน้าของจานขึ้นไปทางทิศเหนือ การสำรวจพื้นที่ในขั้นต้น ควรพิจารณาว่าพื้นที่ตรงส่วนที่เราจะติดตั้งมีตึกสูงๆ หรือต้นไม้บดบังทิศทางที่เราจะรับสัญญาณหรือไม่ หรือพื้นที่บริเวณนั้นมีสายไฟแรงสูง หรือวัตถุอื่นๆ กั้นขวางทิศทางของสัญญาณที่จะลงมาสู่จานรับสัญญาณของเราหรือไม่ และควรตรวจสอบดูด้วยว่าบริเวณใกล้เคียงพื้นที่ที่จะติดตั้งจะมีโครงการก่อสร้างอาคารสูงที่มีผลกระทบต่อการรับสัญญาณดาวเทียมหรือไม่

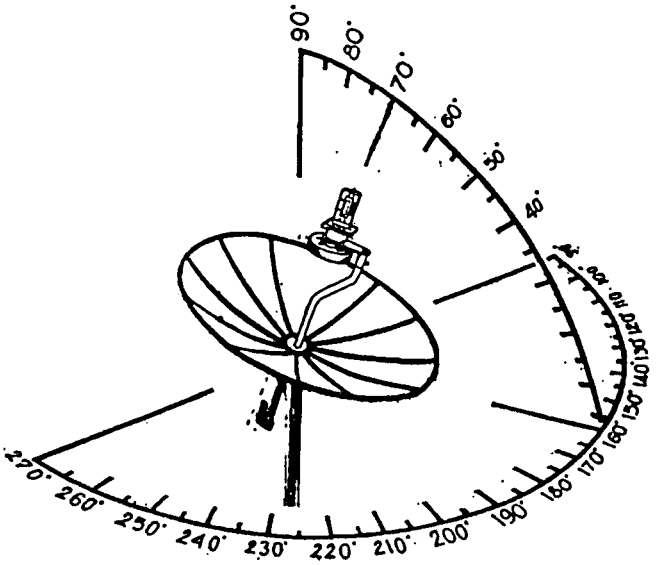
วิธีง่ายๆ ที่ควรเริ่มทำการสำรวจก็คือ ไปยืนอยู่ในพื้นที่ที่เราจะติดตั้งจานรับสัญญาณ แล้วมองขึ้นไปบนฟ้าทางทิศใต้ (ตัวอย่างเช่น ถ้าอยู่ในกรุงเทพ และต้องการรับสัญญาณจากดาวเทียมเอเชียแซทก็ให้หันหน้าขึ้นไปบนฟ้าทางทิศใต้ แล้วเงยหน้าขึ้นทำมุมกับแนวระนาบของพื้นดินขึ้นไปประมาณ 70 องศา) ว่าปราศจากสิ่งกีดขวางหรือไม่ จากนั้นลองมองเอียงไปทางทิศตะวันออกเฉียงใต้และทิศตะวันตกเฉียงใต้ดูด้วยว่ามีอะไรมาบดบังหรือไม่ ถ้าไม่มีก็แสดงว่าเราสามารถติดตั้งจานรับสัญญาณดาวเทียมในตำแหน่งนั้นได้

2.4.2 มุมกวาดและมุมเงย

มุมกวาดหรือมุมอาซิมุธ (Azimuth) และมุมเงยหรือมุมเอลีเวชั่น (Elevation) คือองค์ประกอบพื้นฐานร่วมที่ใช้ในการพิจารณาตำแหน่งของดาวเทียมแต่ละดวงที่อยู่บนท้องฟ้าโดยมีมุมกวาดเป็นตัวบอกทิศทางของดาวเทียวจากตำแหน่งที่เรายืนอยู่ ส่วนมุมเงยนั้นจะเป็นมุมซึ่งจานรับสัญญาณดาวเทียมแหงนหน้าขึ้นไปหาดาวเทียม โดยที่ดาวเทียมทุกดวงจะมีค่าของมุมกวาดและมุมเงยในแต่ละพื้นที่ที่จะติดตั้งจานรับสัญญาณของมันโดยเฉพาะ

เมื่อเราทราบมุมเงยของดาวเทียมที่เราต้องการแล้ว เราก็สามารถใช้เข็มทิศในการหาทิศของดาวเทียม เพื่อปรับหน้าของจานให้ชี้ไปยังดาวเทียมได้อย่างถูกต้อง การใช้เข็มทิศนั้นควรจะกระทำ

ในที่โล่ง ทางที่ดีควรให้ห่างจากสิ่งที่เป็นโลหะขนาดใหญ่ หรือบริเวณที่มีสายไฟฟ้าแรงสูงพาดผ่าน หรือมีหม้อแปลงไฟฟ้าอยู่เหนือบริเวณนั้น

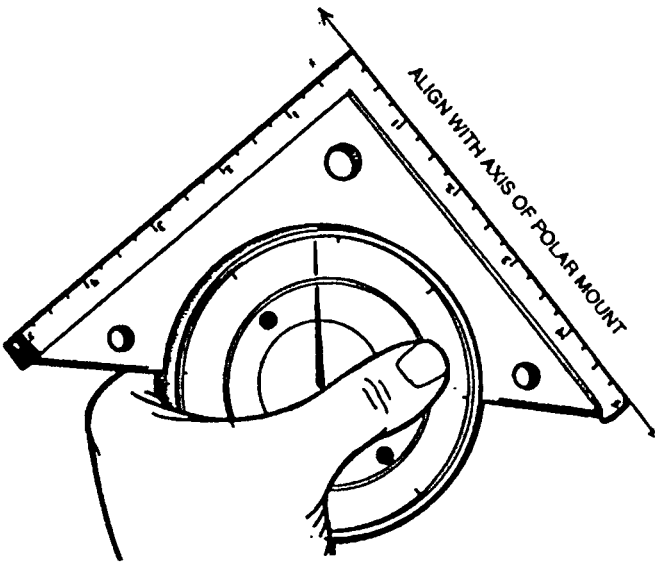


รูปที่ 2.23 พิกัดของมุมกวาดและมุมเงยของจานรับสัญญาณที่ติดตั้งอยู่ในกรุงเทพ

พื้นที่ที่จะติดตั้งจานรับสัญญาณบางพื้นที่นั้นหากอยู่บนพื้นที่ที่อยู่เหนือเส้นศูนย์สูตรก็จะทำให้มีมุมเงยที่แหงนหน้าขึ้นไปหาดาวเทียมมาก อย่างเช่นประเทศอินโดนีเซียหรือบรูไน ซึ่งตรงกันข้ามกับพื้นที่ที่อยู่ค่อนไปทางทิศเหนือก็จะมีมุมเงยที่ต่ำกว่า และในบางจุดนั้นมุมเงยของดาวเทียมค่อนข้างจะอยู่ที่ปลายสุดของทิศตะวันออก หรือทิศตะวันตกของพื้นที่ที่เราจะติดตั้งจานรับสัญญาณ ทำให้ทิศทางของดาวเทียมเมื่อมองจากพื้นที่ดังกล่าวตำลงไปหาพื้นโลก ทำให้ไม่สามารถรับสัญญาณจากดาวเทียมดวงนั้นได้ ซึ่งเป็นเหตุผลของคำตอบที่ว่าทำไมสถานีรับสัญญาณโทรทัศน์จากดาวเทียมในประเทศไทยไม่สามารถรับสัญญาณภาพที่ส่งภายในประเทศที่อยู่ทวีปอเมริกาเหนือได้ ส่วนสถานีรับสัญญาณในทวีปอเมริกาเหนือก็ไม่สามารถรับสัญญาณภาพจากดาวเทียมในทวีปเอเชียได้เช่นกัน

เครื่องมือที่ใช้วัดมุมเอียง (Inclinometer) ซึ่งเรามักจะเห็นใช้งานในงานของช่างไม้นั้นสามารถที่จะนำมาใช้วัดมุมเงยของจานรับสัญญาณได้ ซึ่งบางครั้งเรามักจะเรียกเครื่องมือชนิดนี้ว่า Angle finder เครื่องมือชนิดนี้สามารถนำไปใช้ในช่วงที่เรากำลังสำรวจพื้นที่ที่จะติดตั้งจานรับสัญญาณ เพื่อพิจารณาว่ามีสิ่งใดกีดขวางทิศทางของดาวเทียมที่ต้องการรับสัญญาณที่เราติดตั้งหรือไม่ วิธีใช้ก็คือยืนถือเครื่องมือนี้ที่บริเวณศูนย์กลางของพื้นที่ที่จะติดตั้งจานรับสัญญาณ จากนั้นหันเครื่องมือไปยังทิศทางหรือมุมกวาดของดาวเทียมที่ต้องการจะรับแล้วเอียงเครื่องมือขึ้นหรือลงจน

กระทั่งตรงกับค่ามุมเงยที่เราหามาได้ ใช้สายตามองไปตามแนวของเครื่องมือ ตามรูปที่ --- (ด้านที่มีลูกศรแสดงเอาไว้) ก็จะทำให้สามารถเห็นสิ่งกีดขวางซึ่งอาจจะมาบดบังสัญญาณระหว่างดาวเทียมและจานรับสัญญาณของเราได้ทันที



รูปที่ 2.24 อุปกรณ์วัดมุมเอียง Angle finder

เมื่อเห็นว่าสิ่งใดสิ่งหนึ่งมากีดขวางการรับสัญญาณจากดาวเทียมดวงใดดวงหนึ่งหรือหลายดวงก็ตาม เราสามารถจะใช้วิธีเคลื่อนย้ายสิ่งกีดขวางเหล่านี้ ซึ่งอาจเป็นกิ่งไม้หรือต้นไม้ออกไปก็จะแก้ปัญหาได้ แต่ถ้าสิ่งกีดขวางนั้นเป็นอาคารหรือภูเขาก็จะต้องทำการย้ายตำแหน่งที่ตั้งของจานรับสัญญาณใหม่โดยอาจจะนำไปไว้บนหลังคาหรือทำเสายึดจานรับให้สูงจากพื้นดิน โดยให้มีระดับความสูงพอที่จะทำให้วิวระหว่างดาวเทียมบนท้องฟ้ากับจานรับปราศจากสิ่งกีดขวาง

ยังมีสาเหตุอีกประการหนึ่งที่มีผลทำให้การรับสัญญาณจากดาวเทียมอาจจะไปอยู่ระหว่างทิศทางของสัญญาณไมโครเวฟ ที่ส่งจากจุดหนึ่งไปอีกจุดหนึ่ง เช่นสถานีเชื่อมโยงสัญญาณโทรศัพท์ด้วยสัญญาณไมโครเวฟ ขององค์การโทรศัพท์แห่งประเทศไทย ที่เรียกว่า Microwave link เป็นต้น และยังมีบางพื้นที่ที่จัดอยู่ในประเภทที่สามารถจะรบกวนการรับสัญญาณจากดาวเทียมได้ เช่น บริเวณรอบสนามบิน อู่เรือ หน่วยที่ตั้งทหาร ซึ่งใช้สัญญาณไมโครเวฟในการติดต่อสื่อสารกันมาก

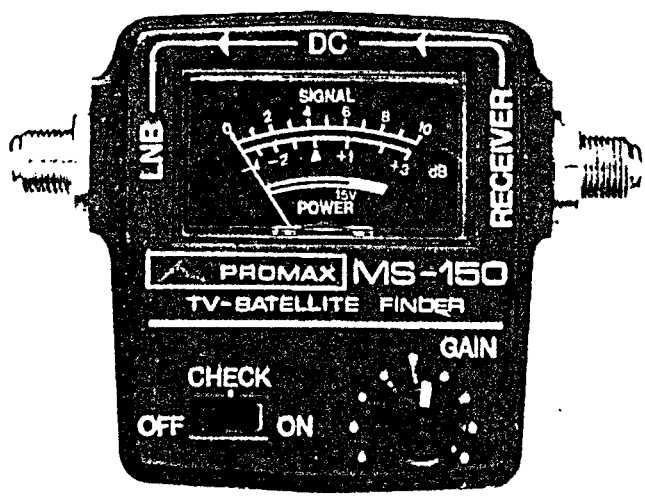
วิธีที่ดีที่สุดที่จะตรวจสอบว่า บริเวณนั้นมีการรบกวนจากสัญญาณไมโครเวฟหรือไม่ ก็คือใช้จานรับสัญญาณและเครื่องรับแบบเคลื่อนย้ายได้ไปทดสอบในสถานที่จริง เพื่อดูว่ามีสัญญาณอะไรมารบกวนบ้าง ซึ่งเป็นวิธีที่ดีและได้ผลถูกต้องที่สุด โดยการทดสอบนั้นได้กระทำที่จุดที่จะทำการติดตั้งจานรับสัญญาณจริงๆ

สถานที่บางจุดที่ทำการทดสอบนั้น จะต้องไปกระทำกันในวันที่มีการทำงาน เนื่องจากวันนั้นจะมีการใช้โทรศัพท์และใช้สัญญาณไมโครเวฟของเครื่องมืออื่นๆ กันอย่างเต็มที่ หากอยู่ในพื้นที่ธุรกิจก็จะต้องทดสอบกันในเวลาที่มีการทำงานเช่นกันด้วย

2.4.3 การปรับตั้งเมาท์

การตั้งเสาซึ่งใช้สำหรับเป็นแกนหมุนของเมาท์แบบโพลาร์นั้น จะต้องจัดให้แนวของตัวแกนวางอยู่ในทางทิศเหนือให้มากที่สุดเท่าที่จะมากได้ ถ้าหากมีการคาดเคลื่อนไปเพียงเล็กน้อย ก็จะมีผลต่อการกวาดเพื่อหาตำแหน่งของดาวเทียมที่โคจรอยู่ในวงโคจรทำให้การรับสัญญาณได้ไม่ดีนัก การตั้งตำแหน่งของแกนดังกล่าว สามารถใช้เข็มทิศที่นำมาใช้ในการสำรวจพื้นที่ที่จะติดตั้งจานรับสัญญาณ

เมื่อติดตั้งเสาหลักจนได้ตำแหน่งที่ถูกต้องแล้ว ให้นำจานรับสัญญาณติดตั้งลงบนเมาท์ที่อยู่บนยอดเสา ทำการยึดน็อตไม่ต้องแน่นมากเพียงเพื่อยึดจานเอาไว้ให้อยู่ในขณะที่ทำการทดลองกวาดหาตำแหน่งของดาวเทียมแต่ละดวงเท่านั้นก็พอ โดยในการทดลองหาตำแหน่งดาวเทียมจะใช้มิเตอร์ในการวัดความแรงของสัญญาณ เมื่อทดลองจนได้คุณภาพของการรับสัญญาณที่ดีที่สุดในแต่ละดวง แล้วให้ขันน็อตทุกตัวให้แน่นก็ถือว่าเสร็จสิ้นในกระบวนการนี้



รูปที่ 2.25 มิเตอร์วัดความเข้มของสัญญาณดาวเทียม ซึ่งสามารถแสดงผลด้วยเข็มและเสียง

2.4.4 การติดตั้งอุปกรณ์ LNB และ Feedhorn

อุปกรณ์คู่นี้จะถูกติดตั้งอยู่ที่ด้านหน้าของจานตรงจุดโฟกัส โดยจะต้องนำ LNB ประกอบเข้ากับตัว Feedhorn ในขณะที่ทำการประกอบอุปกรณ์ห้ามใช้มือหรืออื่นๆ ไปสัมผัสกับโพรบซึ่งเป็นแกนโลหะเล็กๆ ที่อยู่ในเมาท์ของ LNB เพราะจะทำให้มีคราบของไขมันหรือสิ่งสกปรกเกิดขึ้นบนโพรบนี้ จะมีผลทำให้ประสิทธิภาพของการรับสัญญาณลดน้อยลงไปก็ได้ อีกสิ่งหนึ่งที่ต้องสนใจเป็นพิเศษก็คือบริเวณช่วงรอยต่อของ Feedhorn กับ LNB นั้นจะมีร่องสำหรับใส่ขอบยางเพื่อป้องกันน้ำ ฝุ่นละอองและความชื้นเพื่อไม่ให้เข้าไปใน LNB ดังนั้นในการประกอบอุปกรณ์ทั้ง 2 เข้าด้วยกันจะต้องดูให้แน่ใจว่าใส่ขอบยางดังกล่าวไว้ถูกต้องหรือไม่

การติดตั้งฟีดฮอร์นเข้ากับจุดศูนย์กลางที่อยู่ด้านหน้าของจานนั้น จะต้องอยู่ที่ตำแหน่งของจุดโฟกัสอย่างแท้จริงซึ่งระยะห่างของจุดศูนย์กลางของจานรับสัญญาณกับเมาท์ของฟีดฮอร์นนั้นจะมีระยะแตกต่างกันออกไปถ้าจานรับสัญญาณมีเส้นผ่านศูนย์กลางหรือความลึกของจานไม่เท่ากัน ระยะของจุดโฟกัสนี้ทางโรงงานผู้จะกำหนดมาไว้ให้กับจานรับสัญญาณที่ผลิตมาแต่ละแบบ

สำหรับขาที่ใช้ยึดฟีดฮอร์นเข้ากับจานรับสัญญาณ จะมีอยู่ด้วยกัย 2 แบบ แบบแรกเรียกว่า Botton hook หรือบางครั้งจะเรียกว่า LNB Tube มีลักษณะเป็นท่อเหล็กเพียงท่อเดียวยื่นเข้ามาจากจุดศูนย์กลางของจานรับสัญญาณ โดยมีความยาวใกล้เคียงกับระยะโฟกัสของจานรับสัญญาณ (สามารถปรับได้) ที่ปลายของท่อเหล็กที่อยู่เหนือจานรับสัญญาณจะถูกดัดให้งอเป็นรูปตะขอเพื่อเอาไว้ยึดตัวฟีดฮอร์นให้ด้านหน้าของฟีดฮอร์นหันเข้าไปหาด้านหน้าของจาน บางครั้งอาจจะต้องใช้สายยึดหรือ Guy-wire มาช่วยยึดตัว Booton hook นี้ด้วย เพื่อเพิ่มความแข็งแรงหรือเพื่อป้องกันไม่ให้ตำแหน่งของฟีดฮอร์นเกิดการเปลี่ยนแปลง ประโยชน์จากจุดนี้ก็คือในกรณีที่เราติดตั้ง LNB ทั้งแบบ C-Band และ Ku-Band เข้ากับฟีดฮอร์น เพื่อให้สามารถรับสัญญาณทั้ง 2 แบบได้นั้น ตำแหน่งของฟีดฮอร์นจะต้องอยู่กับที่ โดยไม่มีการเคลื่อนไหวหรือเปลี่ยนแปลง อันเนื่องมาจากน้ำหนักของLNB เอง หรือจากลมพายุ ซึ่งขาแบบ Botton hook นี้ ค่อนข้างจะอ่อนไหวง่าย ต่อสิ่งแวดล้อมเหล่านี้ และผลที่สำคัญอีกประการหนึ่งก็คือ ลำคลื่นของสัญญาณ Ku-Band นี้มีขนาดเล็กกว่าลำคลื่นของ C-Band ดังนั้นส่วนโค้งของจานรับสัญญาณกับตำแหน่งของฟีดฮอร์นจะต้องรักษาให้อยู่ในตำแหน่งที่ปรับเอาไว้ในครั้งแรกโดยไม่มีการเปลี่ยนแปลง

ในการที่จะตรวจสอบว่าตัวฟีดฮอร์นได้ถูกติดตั้งอยู่ตรงจุดศูนย์กลางของจานรับสัญญาณหรือไม่นั้น สามารถจะทำได้โดยการใช้วิธีวัดจากขอบของปากจานรับสัญญาณ ไปยังขอบของฟีดฮอร์นที่อยู่บริเวณเปลายเปิดของมัน ให้มีระยะเท่ากันทั้ง 4 ด้าน ซึ่งตรงจุดนี้หากเราใช้ Guy-wire มาช่วยยึด Botton hook แล้วก็จะทำให้การติดตั้งฟีดฮอร์นให้อยู่ตรงกับตำแหน่งจุดศูนย์กลางของจานรับสัญญาณทำได้ง่ายและถูกต้องยิ่งขึ้นโดยการปรับความตึงของGuy-wire

แกนยึดฟีดฮอร์นแบบที่ 2 จะเป็นแบบที่ใช้แกนยึดแบบหลายแกน (Multi-legged support) มีทั้งแบบใช้สามแกนและสี่แกน โดยแต่ละแกนจะมีความยาวเท่ากัน แกนยึดฟีดฮอร์นแบบนี้จะมีคุณภาพดีกว่าแบบ Botton hook ไม่ว่าจะเป็นเรื่องของความแข็งแรงที่มีมากกว่า ตำแหน่งของฟีดฮอร์นไม่คาดเคลื่อนง่าย และระยะของจุดโฟกัสแน่นอน จึงเหมาะที่จะใช้กับระบบรับสัญญาณย่าน Ku-Band เพราะเชื่อถือได้มากกว่า

ในการหาจุดโฟกัสของจานรับสัญญาณนั้น สามารถทำได้โดยการปรับฟีดฮอร์นขึ้นและลงจนกระทั่งความแรงของสัญญาณที่อ่านได้ จากมิเตอร์ของเครื่องรับมีค่าสูงสุด การปรับตำแหน่งของจุดโฟกัสลักษณะนี้จะทำได้ง่ายก็ต่อเมื่อเราใช้แกนยึดฟีดฮอร์นแบบ Botton hook เพราะการปรับสามารถทำได้โดยยึดท่อฟีดฮอร์นเพียงเดียว ซึ่งถ้าเป็นแกนยึดแบบ Multi-legged แล้วจะต้องปรับทุกๆแกนทำให้ไม่ค่อยสะดวกเท่าใดนัก แต่ถ้าเราสามารถคำนวณหาระยะโฟกัสของจานรับสัญญาณนั้นได้ เราก็จะสามารถปรับแต่งระยะของจุดดังกล่าวได้ง่ายขึ้น ไม่ว่าจะเป็นแกนยึดฟีดฮอร์นแบบ Botton hook หรือแบบ Multi-legged

วิธีการคำนวณหาระยะของจุดโฟกัสสามารถกระทำได้โดยวิธีนำขนาดของเส้นผ่านศูนย์กลางของจานรับสัญญาณที่มีหน่วยเป็นฟุตไปคูณกับค่าอัตรา f/D ของจานรับสัญญาณซึ่งกำหนดมาจากโรงงาน ตัวอย่างเช่น จานรับสัญญาณมีเส้นผ่านศูนย์กลาง 10 ฟุต มีค่าอัตราf/D เท่ากับ 0.45 ดังนั้นจะมีระยะของจุดโฟกัสเท่ากับ 10 X 0.45 = 4.5 ฟุต หรือ 54 นิ้ว

การวัดเส้นผ่านศูนย์กลาง จะต้องวัดที่ขอบของจานจากด้านหนึ่งผ่านจุดศูนย์กลางไปอีกด้านหนึ่ง และรัศมีได้จากครึ่งหนึ่งของเส้นผ่านศูนย์กลางส่วนความลึกของจานนั้นจะวัดจากจุดศูนย์กลางของจาน ที่บริเวณท้องจานไปตั้งฉากกับแนวระนาบของขอบจานซึ่งอาจจะใช้วิธีวัดง่ายๆ คือ ใช้เชือกขึงให้ตึงจากขอบด้านหนึ่งถึงขอบอีกด้านหนึ่ง แล้ววัดจากจุดกึ่งกลางของเชือกไปที่จุดศูนย์กลางของจานก็จะทำให้ทราบความลึกของจาน

ฟีดฮอร์นที่มีจำหน่ายในท้องตลาดทุกวันนี้ มักจะมี Scalar ring ที่สามารถปรับได้มาให้ ซึ่งสามารถจะถอดแยกออกมาได้เป็น 2 ส่วน คือส่วนที่เป็น Scalar plate ซึ่งมีลักษณะเป็นแผ่นวงกลมมีรูดรงกลางสำหรับใส่ท่อนำคลื่นบริเวณรอบรูดังกล่าวจะเป็นลักษณะวงแหวนหลายวงซ้อนกันอยู่ โดยเชื่อมติดอยู่กับ Scalar plate ดังกล่าว อีกส่วนหนึ่งก็คือท่อนำคลื่นแบบวงกลมซึ่งจะต้องนำไปติดตั้งเข้ากับ LNB และท่อนำคลื่นสามารถจะเลื่อนขึ้นลงภายในScalar plate ได้

การติดตั้งฟีดฮอร์นแบบนี้ จะต้องให้ระยะของปากท่อนำคลื่นสัมพันธ์กับค่าอัตราของ f/D ของจานรับสัญญาณนั้น

สิ่งที่ต้องตรวจเช็คเสมอในการติดตั้งฟีดฮอร์นข้อสุดท้ายคือ จะต้องใช้เครื่องมือวัดความเอียงมาวัดระนาบของปากฟีดฮอร์นกับระนาบของหน้าจานรับสัญญาณ และปรับให้อยู่ในตำแหน่งขนานให้มากที่สุด

2.4.5 การปรับแต่งโพลาไรเซชั่น

ขายึดฟีดฮอร์นที่ให้มากับจานรับสัญญาณแบบคงที่มีไว้สำหรับยึดตัวฟีดฮอร์นกับ LNB ให้อยู่คงที่ที่บริเวณส่วนหน้าของจานโดยจะรักษาระยะห่างระหว่างฟีดฮอร์นกับพื้นผิวของจานให้คงที่และถูกต้องอยู่เสมอด้วย แต่ก็ยังคงสามารถปรับได้อีกเล็กน้อยเพื่อให้ได้สัญญาณที่ดีขึ้น

ดาวเทียมที่มีการส่งสัญญาณโทรทัศน์แบบ DBS จะใช้โพลาไรเซชั่นแบบ Circular เพียงอย่างเดียวในการส่งสัญญาณโทรทัศน์ลงมา จึงไม่จำเป็นต้องมีการปรับจูนโพลาไรเซชั่นของฟีดฮอร์นอีก เนื่องจากถูกปรับแต่งมาจากโรงงานเรียบร้อยแล้ว ส่วนดาวเทียมที่ส่งสัญญาณโทรทัศน์ระบบอื่นจะใช้โพลาไรเซชั่นแบบลิเนียร์ ซึ่งจะมีทั้งแบบแนวนอนและแนวตั้งการปรับแต่งโพลาไรเซชั่นของฟีดฮอร์นแบบนี้จะทำการปรับได้จากปุ่มปรับโพลาไรเซชั่นที่อยู่ที่เครื่องรับ

2.4.6 การเดินสายเคเบิล

จากอุปกรณ์ LNB ที่อยู่ที่จานรับสัญญาณมายังเครื่องรับสัญญาณดาวเทียมที่อยู่ภายในบ้านและจากเครื่องรับสัญญาณนี้ก็จะใช้สายโคแอกเชียลเส้นสั้นๆ ต่อจากเครื่องรับไปเข้ายังโทรทัศน์ สายเคเบิลที่ใช้เชื่อมต่อระหว่างจานรับสัญญาณกับเครื่องรับสัญญาณ จะแบ่งได้เป็น 3 กลุ่มได้แก่ สายโคแอกเชียลที่เดินจากตัว LNB มายังเครื่องรับ สายเคเบิลที่เดินมาจากเซอร์โวมอเตอร์ของฟีดฮอร์น มายังตำแหน่งของเซอร์โวมอเตอร์ที่อยู่ด้านหลังของเครื่องรับซึ่งมีสายอยู่ภายในเคเบิลนี้อยู่ 3 เส้น และสายเคเบิลกลุ่มสุดท้ายนี้จะมีสายภายในอยู่ 4-5 เส้น ซึ่งจะเดินจากมอเตอร์ของ Actuator ที่อยู่ด้านล่างของจานรับสัญญาณมายังขั้วสำหรับควบคุมที่อยู่ด้านหลังเครื่องรับ

ทางที่ดีเราควรจะใช้สายเคเบิลที่ใช้กับระบบการรับสัญญาณโทรทัศน์จากดาวเทียมโดยเฉพาะ ซึ่งจะนำเอาสายเคเบิลทั้ง 3 กลุ่มที่กล่าวไปแล้วรวมติดเอาไว้ด้วยกัน (ในบ้านเรามักจะเรียกว่าสาย TVRO) เป็นแถบเรียงกันทั้ง 3 เส้น

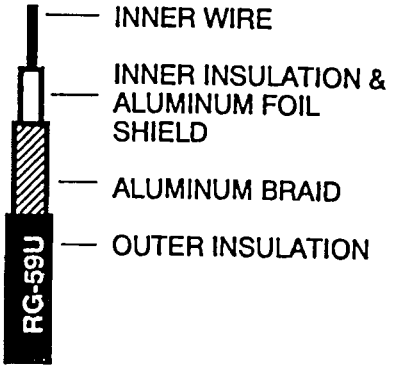
ก่อนที่จะต่อสายเคเบิลเข้ากับระบบเราควรจะใช้สายเคเบิลอีกชุดหนึ่งที่มีความยาวไม่มากนัก มาต่อเข้ากับเครื่องรับสัญญาณและเครื่องรับโทรทัศน์ไว้ชั่วคราวก่อน โดยนำเครื่องรับมาตั้งไว้ในบริเวณที่ใกล้กับจานรับสัญญาณ เพื่อที่จะทำการปรับแต่งส่วนต่างๆของจานรับสัญญาณ ตลอดจนการทดลองกวาดหาตำแหน่งดาวเทียม เพื่อที่จะทำให้สะดวกต่อการสังเกตสัญญาณภาพบนจอเครื่องรับโทรทัศน์ และความแรงของสัญญาณบนมิเตอร์ให้สัมพันธ์กับการปรับแต่งตำแหน่งของ

จานรับสัญญาณ จากนั้นจึงทำการต่อสายเคเบิลที่ใช้งานจริงเข้ากับเครื่องรับสัญญาณและเครื่องรับโทรทัศน์ที่อยู่ภายในบ้าน ซึ่งในขั้นตอนสุดท้ายนี้ อาจใช้มิเตอร์วัดความแรงสัญญาณไปต่อเอาไว้ที่จานรับสัญญาณแล้วให้อีกคนหนึ่งไปยืนสังเกตเอาไว้ จากนั้นให้คนที่อยู่ภายในบ้านทำการปรับจากตัวเครื่องรับสัญญาณ

ข้อพิจารณาอีกประการหนึ่งในการพิจารณาซื้อสาย TVRO มาใช้งานก็คือฟีดฮอร์นที่เราใช้นั้นเป็นแบบ Hybrid feed ซึ่งใช้ LNB แบบ C-Band และ Ku-Band ได้พร้อมกัน 2 ตัว หรือเป็นแบบ Dual C-Band feed ซึ่งใช้ LNB แบบ C-Band ได้พร้อมกันทั้ง 2 ตัว ถ้าเกิดเป็นไปตามนี้สาย TVRO ที่ใช้จะต้องเป็นแบบที่มีสายโคแอกเชียล 2 เส้น หรือในกรณีขั้นต้นยังใช้ฟีดฮอร์นที่มี LNB เพียงตัวเดียวแต่ต่อไปในอนาคตอาจจะเปลี่ยนมาใช้แบบที่มี LNB 2 ตัว จะต้องใช้สายเคเบิลที่เป็นสายโคแอกเชียลที่แยกกันต่างหาก

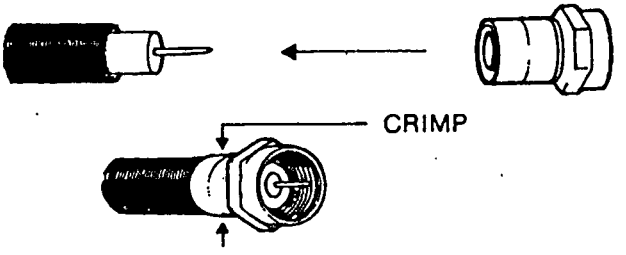
2.4.7 สายต่อและขั้วต่อโคแอกเชียล

สายที่ใช้ในการเดินสัญญาณจากจานรับสัญญาณมายังเครื่องรับโทรทัศน์เรียกว่าสายโคแอกเชียล ซึ่งสายนี้จะทำหน้าที่นำสัญญาณไฟฟ้าจากอุปกรณ์ LNB มายังเครื่องรับสัญญาณ



รูปที่ 2.26 สายเคเบิลแบบโคแอคเชียล

ขั้วต่อที่ใช้สำหรับสายโคแอกเชียลสำหรับโทรทัศน์ผ่านดาวเทียมนั้น เราจะเรียกว่าขั้วต่อแบบเอฟ (F- Connector) โดยจะต่อเข้ากับปลายสายทั้ง 2 ของโคแอกเชียล เพื่อนำไปต่อเข้ากับ LNB และเครื่องรับสัญญาณที่อยู่ภายในบ้าน การยึดขั้วต่อแบบเอฟเข้ากับสายโคแอกเชียลจะต้องใช้เครื่องมือพิเศษเฉพาะ



รูปที่ 2.27 ขั้วต่อแบบ F-TYPE

เมื่อเรานำสายโคแอกเชียลที่มีขั้วต่อแบบเอฟยึดติดกันแล้วไปต่อเข้ากับขั้วต่อตัวเมียซึ่งอยู่ด้านหลังของเครื่องรับสัญญาณจะต้องระวังไม่ให้สายทองแดงที่เป็นแกนของสายโคแอกเชียลหักหรืองอเพราะอาจไปแตะเข้ากับส่วนที่อยู่ด้านนอกของขั้วต่อซึ่งมีสภาพเป็นกราวด์จะทำให้เกิดการลัดวงจรได้ วิธีการใส่ที่ถูกต้องคือ จะต้องให้ส่วนที่เป็นสายทองแดงเสียบลงไปในช่องเล็กๆ ตรงกลางขั้วต่อตัวเมีย จากนั้นหมุนเกลียวไปทางขวาก็จะช่วยให้สาเหตุที่กล่าวมาหมดไป ข้อแนะนำอีกวิธีหนึ่งก็คือ ระหว่างการต่อสายโคแอกเชียลนี้ให้ปิดสวิตซ์เครื่องหรือดึงปลั๊กไฟออกจากตัวเครื่องรับดังได้กล่าวไว้ก่อนหน้านี้แล้วครั้งหนึ่ง

สายโคแอกเชียลที่ใช้ในการนำสัญญาณไมโครเวฟจาก LNB ลงมายังเครื่องรับที่อยู่ภายในบ้านนั้น จะต้องใช้สายโคแอกเชียลที่มีค่าอิมพิแดนซ์ 75 โอห์ม เท่ากับที่ใช้ในเครื่องรับโทรทัศน์ ดังนั้นในการเลือกซื้อจะต้องพิจารณาให้ดี เพราะจะมีสายโคแอกเชียลที่มีค่าอิมพีแดนซ์ 50 โอห์มขายอยู่

สายโคแอกเชียลที่มีจำหน่ายตามท้องตลาดจะมีหลายชนิดเช่น สาย RG-59U ควรใช้ความยาวที่ไม่เกิน 100 ฟุต เนื่องจากมีอัตราของการสูญเสียของสัญญาณสูงซึ่งหากต้องการเดินสายยาวกว่านี้ควรใช้สายแบบ RG-6 หรือ RG-11 เพราะมีขนาดของเส้นผ่านศูนย์กลางภายในที่ใหญ่กว่าอัตราการสูญเสียภายในจึงต่ำกว่า และจำเป็นต้องใช้ขั้วต่อแบบเอฟที่มีขั้วต่อขนาดใหญ่กว่าด้วย ปัญหาที่เกิดจากการสูญเสียเมื่อใช้สายโคแอกเชียลที่มีความยาวหลายร้อยฟุตสามารถแก้ได้โดยการใช้อุปกรณ์ขยายสัญญาณย่านความถี่ UHF มาต่ออนุกรมเข้ากับสายเคเบิลก็จะทำให้ชดเชยสัญญาณที่สูญเสียไปได้

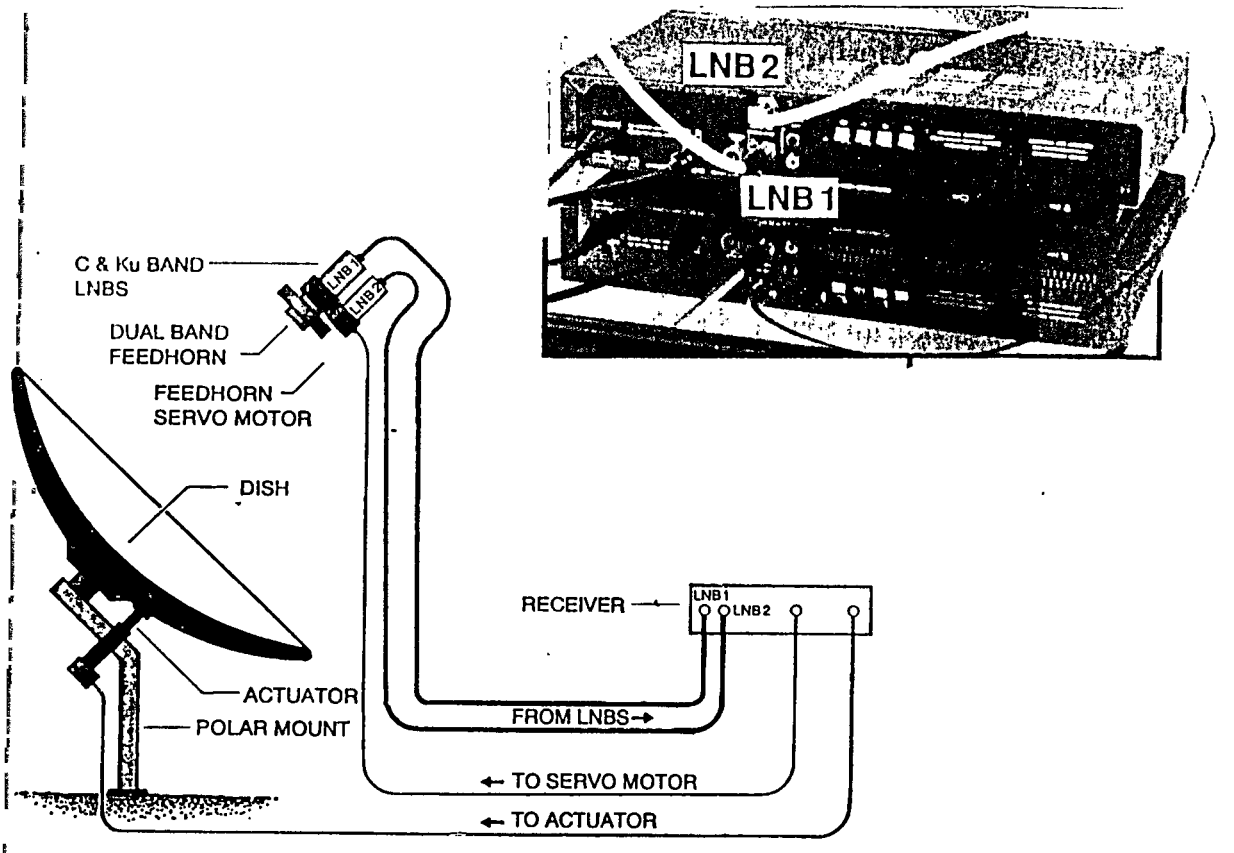
สายโคแอกเชียลที่ใช้นั้น นอกจากจะทำหน้าที่นำสัญญาณไมโครเวฟจาก LNB ลงมายังเครื่องรับสัญญาณแล้ว ยังทำหน้าที่ส่งแรงดันไฟเลี้ยงขึ้นไปเลี้ยงให้ LNB และ Line amplifier ที่ต่อ

อนุกรมอยู่ระหว่างสายด้วย โดยส่งเป็นแรงดันไฟตรงผ่านไปทางสายทองแดงที่เป็นแกนกลางของสายโคแอกเชียล LNB และ Line amplifier จะมีไฟเลี้ยงอยู่ตลอดเวลาตราบเท่าที่เครื่องรับสัญญาณถูกเปิดหรือทำงานอยู่ ดังนั้นในการติดตั้งอุปกรณ์ครั้งแรกควรจะถอดปลั๊กไฟของเครื่องรับสัญญาณออกเสียก่อนและตรวจดูสายโคแอกเชียลว่ามีการลัดวงจรหรือไม่ จากนั้นจึงต่อปลายด้านหนึ่งเข้า LNB และปลายอีกด้านหนึ่งเข้ากับขั้วต่อที่อยู่ด้านหลังเครื่องรับสัญญาณ

ข้อควรปฏิบัติอีกข้อหนึ่งก็คือ ที่ขั้วต่อ LNB นั้น หลังจากที่ต่อสายโคแอกเชียลเข้าไปแล้วควรใช้แถบเทปสำหรับพันกันน้ำป้องกันเอาไว้ด้วย หรืออาจใช้วิธีใช้ซิลิโคนซีลเลอร์ฉีดหุ้มเอาไว้ก็สามารถป้องกันน้ำได้เช่นกัน แต่ก็มีบริษัทผู้ผลิตหลายบริษัทได้ผลิตพลาสติดสำหรับครอบ LNB มาให้ด้วยซึ่งสามารถป้องกันน้ำได้อีกชั้นหนึ่ง

2.4.8 การเดินสายสำหรับฟีดฮอร์น

สายที่ใช้ควบคุมฟีดฮอร์นจะเป็นส่วนหนึ่งของสาย TVRO ซึ่งได้กล่าวไปแล้วในหัวข้อที่ผ่านมา ภายในจะมีสายทองแดงมีฉนวนหุ้มอยู่ 3 เส้น (สายทองแดงประมาณเบอร์ 22 หรืออาจใหญ่กว่าสำหรับบางยี่ห้อ) สายทั้ง 3 เส้นจะถูกกำหนดให้มีสีแตกต่างกัน และแต่ละสีนั้นก็จะถูกกำหนดให้นำไปต่อเข้ากับขั้วต่อที่อยู่ด้านหลังของเครื่องรับโดยสีขาวจะต่อเข้ากับขั้วของสัญญาณพัลซ์ (Pulse) สีแดงจะถูกต่อเข้ากับขั้วแรงดันไฟ (+V หรือ Power) และสีดำจะต่อเข้ากับขั้วกราวด์เสมอ ซึ่งทั้งสามสายนี้จะถูกหุ้มด้วยอลูมิเนียมฟรอย์เพื่อป้องกันไม่ให้สัญญาณรบกวนเข้าไปรบกวนสัญญาณพัลซ์ที่ใช้ควบคุมฟีดฮอร์น จากนั้นจึงหุ้มด้วยพลาสติดที่เป็นฉนวนอีกชั้นหนึ่ง สัญญาณพัลซ์ที่ใช้ควบคุมฟีดฮอร์นนั้นจะถูกส่งมามาจากเครื่องรับสัญญาณ เพื่อใช้ควบคุมตำแหน่งของ
โพรบที่อยู่ภายในฟีดฮอร์นให้มีโพลาไรซ์ตรงกับโพลาไรซ์ของทรานสปอนเดอร์ของดาวเทียมมากที่สุด และสามารถที่จะโปรแกรมได้



รูปที่ 2.28 การเดินสายเชื่อมต่อระหว่างอุปกรณ์ของเครื่องรับสัญญาณแบบ LNB คู่

การควบคุมโพรบเพื่อปรับโพลาไรซ์นี้ ส่วนใหญ่จะใช้เซอร์โวมอเตอร์ในการหมุนตัวโพรบโดยโพรบนี้ จะหมุนได้ก็ต่อเมื่อมีสัญญาณพัลซ์ไปควบคุม สิ่งที่เราควรจะรับทราบไว้ก็คือการเคลื่อนที่หรือการหมุนของโพรบนั้นจะมีการจำกัดเอาไว้ทั้งที่ทวนเข็มและตามเข็มนาฬิกา การติดตั้งฟีดฮอร์นบนจานรับสัญญาณจึงต้องปรับตำแหน่งของฟีดฮอร์นให้เหมาะสม เพื่อที่จะทำให้โพรบเคลื่อนที่จากแนวนอนไปสู่แนวตั้งในช่วง 90 องศาได้โดยไม่เคลื่อนไปถึงจุดที่ถูกจำกัดเอาไว้ มีบริษัทผู้ผลิตหลายแห่งได้ให้คู่มือในการติดตั้ง และปรับตำแหน่งของฟีดฮอร์นที่อยู่บนจานรับสัญญาณมาไว้ให้ด้วย ซึ่งจะเป็นแนวทางในการปรับแต่งได้ง่ายยิ่งขึ้นหากไม่มีมาก็อาจใช้วิธีคลายน็อตยึดฟีดฮอร์นกับขายึดให้หลวมแล้วใช้มือหมุนฟีดฮอร์นจากนั้นทดลองปรับโพรบดูว่าสามารถ

หมุนหรือเคลื่อนที่จากแนวนอนไปยังแนวตั้งในช่วง 90 องศาได้ครบหรือไม่ถ้าได้ก็แสดงว่าพร้อมที่จะใช้งานแล้ว

2.4.9 การเดินสายสำหรับมอเตอร์ขับจาน

สายควบคุมสุดท้ายที่อยู่ในสาย TVRO จะเป็นสายที่ใช้ควบคุมมอเตอร์ขับจานหรือที่เรียกกันว่าแอคทูเอเตอร์ (Actuator) ภายในจะประกอบไปด้วยสาย 2 ชุดทั้งหมด 5 เส้น ชุดแรกจะเป็นสายอ่อนขนาดเบอร์ 14 หรือเบอร์ 16 จะใช้เป็นสายสำหรับจ่ายแรงดันไฟตรงให้กับมอเตอร์ ส่วนอีก 3 เส้นจะเป็นสายอ่อนขนาดเบอร์ 22 และมีซีลด์พันไว้อีกชั้นหนึ่ง สายชุดนี้จะถูกต่อระหว่างเซ็นเซอร์ของแอคทูเอเตอร์ กับขั้วต่อที่อยู่ด้านหลังเครื่องรับสัญญาณดาวเทียม

การต่อสายเซ็นเซอร์ทั้งสามเส้นเข้ากับแอคทู เอเตอร์นั้น จะเหมือนกับการต่อสายเข้ากับเซอร์โวมอเตอร์ของฟีดฮอร์น นั่นก็คือจะมีสายสำหรับจ่ายแรงดัน สัญญาณพัลซ์ และกราวด์ แต่แอคทูเอเตอร์ที่ผลิตออกมาจำหน่ายทุกวันนี้ ส่วนใหญ่จะไม่มีขั้วสำหรับต่อแรงดันในส่วนของตัวเซ็นเซอร์เลย จะใช้เฉพาะขั้วสำหรับสัญญาณพัลซ์และกราวด์เท่านั้น แต่ในกรณีที่แอคทูเอเตอร์มีขั้วของเซ็นเซอร์มาไว้ให้ทั้ง 3 ขั้ว ให้นำไฟบวก 5 โวลท์ จากขั้วต่อด้านหลังของเครื่องรับสัญญาณต่อเข้ากับขั้วของเซ็นเซอร์ที่มีสายสีแดงต่ออยู่ ส่วนสายสัญญาณพัลซ์และกราวด์สามารถต่อสลับกันได้

ส่วนสายไฟที่จ่ายแรงดันให้กั บแอคทูเอเตอร์นั้นขนาดของสายทั้ง 2 เส้นจะใหญ่กว่าสายไฟที่ต่อเข้ากับเซ็นเซอร์โดยจะจ่ายไฟให้กับมอเตอร์เพื่อทำหน้าที่ขับจานให้เอียงไปทางทิศตะวันตกและทิศตะวันออกได้ การต่อสายไฟคู่นี้เข้ากับขั้วต่อทางด้านหลังของเครื่องรับในขั้นแรก ไม่จำเป็นต้องดูขั้วว่าเป็นไฟบวกหรือไฟลบก็ได้ เนื่องจากเครื่องบางรุ่นจะไม่มีเครื่องหมายบวกลบเอาไว้ให้ ดังนั้นจึงต้องใช้วิธีการสังเกตดูว่า ถ้าเรากดปุ่มควบคุมหน้าเครื่องรับสัญญาณหรือบนรีโมทคอนโทรลให้จานรับสัญญาณเอียงไปทางด้านทิศตะวันออกแล้วจานรับสัญญาณจะมีการตอบสนองในการเอียงไปในทิศตะวันออกหรือไม่ถ้าใช่ก็แสดงว่าเราต่อขั้วไฟได้ถูกต้องแล้ว แต่ถ้าเกิดมีการเอียงไปในทิศตรงกันข้าม ให้ใช้วิธีกลับขั้วของสายไฟคู่นี้ที่อยู่ทางด้านหลังของเครื่องรับสัญญาณก็จะใช้งานได้อย่างถูกต้อง

เครื่องรับสัญญาณดวงเทียมบางยี่ห้อต้องการที่จะลดขนาดของเครื่องรับให้เล็กลง และลดอุณหภูมิการทำงานภายในตัวเครื่องให้ลดลงด้วย จึงใช้วิธีแยกแหล่งจ่ายไฟสำหรับมอเตอร์ซึ่งจะมีหม้อแปลงเป็นส่วนประกอบออกมาไว้ภายนอกตัวเครื่องโดยหม้อแปลงชุดนี้ต้องมีแรงดันไฟสลับขาเข้า 220 โวลท์ แล้วเปลี่ยนให้เป็นแรงดันไฟตรง 24-36 โวลท์เพื่อจ่ายให้กับมอเตอร์กระแสตรงของแอคทูเอเตอร์ต่อไป

แหล่งจ่ายไฟภายนอกของเครื่องรับสัญญาณประเภทนี้สามารถจะเอาวางไว้ที่ด้านหลังของเครื่องรับหรือที่อื่นๆ ที่ห่างจากเครื่องรับสัญญาณได้ ยิ่งกว่านั้นหากเราใช้เครื่องรับสัญญาณ 2 เครื่อง เราก็ไม่จำเป็นต้องเสียเงินเพื่อซื้อแหล่งจ่ายไฟชุดที่ 2 มาเพิ่มให้กับระบบอีก เพราะว่าเราใช้เครื่องรับสัญญาณเพียงเครื่องเดียวที่ใช้ในการควบคุมตำแหน่งของจานรับสัญญาณ

ปัญหาของเครื่องแอคทูเอเตอร์ที่สำคัญมากก็คือ ปัญหาของเรื่องน้ำที่จะเข้าไปในมอเตอร์ก็จะทำให้เกิดการลัดวงจรได้ ดังนั้นหลังจากที่ต่อสายเข้ากับแอคทูเอเตอร์แล้วจะต้องตรวจดูว่าแผ่นยางกันน้ำที่ให้มาได้ใส่เอาไว้ดีแล้วหรือไม่ แล้วจึงค่อยยึดด้วยน็อตให้แน่น และควรปรับส่วนที่มอเตอร์ติดตั้งอยู่ ให้ตำแหน่งของรูระบายน้ำที่ผู้ผลิตได้เจาะเอาไว้ให้อยู่ทางด้านล่าง เพื่อให้น้ำสามารถระบายได้อย่างสะดวกทำให้ไม่เกิดความชื้นตกค้างอยู่ภายในมอเตอร์

2.4.10 การเดินสายดิน

เนื่องจากประเทศไทยเป็นประเทศที่อยู่ในเขตร้อนและมีฝนตกชุกทำให้เกิดปรากฏการณ์ทางธรรมชาติบางอย่างซึ่งอาจทำให้เกิดอันตรายกับการรับสัญญาณดาวเทียมของเราได้ ปรากฏการณ์ดังกล่าวเช่น ฟ้าฝ่า ฟ้าแลบ เป็นต้น ดังนั้นจึงควรจะมีระบบป้องภัยเอาไว้ด้วย

ถ้าตำแหน่งที่ติดตั้งจานรับสัญญาณอยู่ใกล้กับระบบกราวด์ของไฟฟ้ากระแสสลับของบ้านให้ใช้สายทองแดงเบอร์ 10AWG หรือใหญ่กว่าต่อเข้ากับระบบกราวด์ของบ้านเข้ากับโครงของท่อเหล็กซึ่งใช้เป็นเสาสำหรับตั้งจานรับสัญญาณได้โดยตรง แต่ถ้าตำแหน่งของจานรับสัญญาณห่างจากตัวบ้านเราจะต้องทำระบบกราวด์ขึ้นมาใหม่โดยแยกออกมาจากระบบกราวด์ของบ้าน วิธีก็คือใช้กราวด์รอด (Ground rod) ฝังลงไปในดินบริเวณที่อยู่ใกล้กับจานรับสัญญาณแล้วใช้สายทองแดงเบอร์ 10 AWG หรือใหญ่กว่าต่อเข้ากับแท่งทองแดงที่โคนเสาสำหรับตั้งจานรับสัญญาณ

2.4.11 การติดตั้งมอเตอร์ขับจานเข้ากับจาน

การบังคับให้จานรับสัญญาณกวาดไปยังตำแหน่งของดาวเทียมที่อยู่ในวงโคจรนั้น สามารถใช้เครื่องควบคุมซึ่งใช้มอเตอร์เป็นตัวขับได้ 2 แบบ แบบแรกเป็นแบบ Horizon-to-Horizon drive ซึ่งจะใช้ชุดมอเตอร์สำหรับขับจานติดตั้งที่เมาท์ของจานรับสัญญาณได้โดยตรง ข้อดีของเครื่องควบคุมแบบนี้ก็คือ สามารถเคลื่อนตำแหน่งของจานให้กวาดในแนวของคลาคออบิทได้ตลอดแนวที่สามารถมองเป็น ณ ตำแหน่งที่จานรับสัญญาณตั้งอยู่ส่วนแบบที่ 2 เป็นแบบ Single-stroke actuator drive จะใช้วิธีให้มอเตอร์หมุนและไปขับแขนที่ติดตั้งอยู่ในกระบอกคล้ายโช๊คอัพเลื่อนเข้าหรือออกเพื่อบังคับให้จานเคลื่อนไปยังตำแหน่งที่ต้องการได้เครื่องควบคุมแบบนี้มีข้อเสียก็คือ สามารถจะเคลื่อนจานรับสัญญาณให้เคลื่อนที่ไปตามแนวคลาคออบิทได้น้อยกว่าแบบแรก

ก่อนที่จะทำการติดตั้งแอคทูเอเตอร์เข้ากับตัวจานรับสัญญาณนั้นควรจะยกตัวจานให้เอียงขึ้นเล็กน้อย โดยอาจใช้ท่อนไม้หรืออะไรก็ได้ที่ไม่ทำให้ตัวจานเสียหายมารองรับเอาไว้จากนั้นให้ใส่แขนของแอคทูเอเตอร์เข้ากับตัวจานเสียก่อน โดยพยายามทำตามข้อแนะนำในคู่มือที่โรงงานให้มาเนื่องจากจุดที่ติดตั้งอยู่ใต้จานรับสัญญาณมีอยู่ 2 จุดคือ ด้านทิศตะวันออกและทิศตะวันตกของเสาที่ตั้งเอาไว้เพื่อยึดตัวจานรับสัญญาณ (กรณีนี้หมายความว่า เราหันด้านหกน้ำของจานรับสัญญาณไปทางทิศใต้เรียบร้อยแล้ว) แล้วเราจะนำแขนดังกล่าวไปติดไว้ตรงจุดไหน วิธีการพิจารณาก็คือต้องพิจารณาว่าพื้นที่ที่ติดตั้งจานรับสัญญาณนั้นอยู่ทางทิศตะวันออกของตะวันตกของทวีปเอเชีย ถ้าอยู่ทางทิศตะวันออกก็ให้นำแขนของแอคทูเอเตอร์ไปติดเอาไว้ที่จุดที่อยู่ทางทิศตะวันตกของจาน และถ้าอยู่ทางทิศตะวันตกก็นำไปติดเอาไว้ที่จุดที่อยู่ทางทิศตะวันออก แล้วถ้าจานรับสัญญาณแบบนี้นำมาติดตั้งในพื้นที่ของในประเทศไทย จะต้องนำแขนของแอคทูเอเตอร์ไปติดอยู่ตรงไหนของจานรับสัญญาณ คำตอบก็คือต้องเป็นจุดที่อยู่ทางทิศตะวันตก

2.4.12 การปรับแต่งจานรับสัญญาณ

หลังจากที่มีการติดตั้งจานรับสัญญาณจะต้องมีการปรับแต่งตำแหน่งของจานเพื่อให้สามารถรับสัญญาณจากดาวเทียมได้ดีที่สุด ส่วนที่ต้องทำการปรับจะแบ่งเป็นส่วนต่างๆดังนี้

การปรับจุดจำกัดมุมกวาด

มีจุดประสงค์เพื่อให้การเลื่อนเข้าออกในกระบอกของแขนของแอคทูเอเตอร์สามารถหยุดการเคลื่อนที่ได้ก่อนที่จะถึงจุดที่เป็นลิมิตสวิตซ์จะตัววงจรจ่างกระแสไฟให้กับมอเตอร์หากปล่อยให้แขนของแอคทูเอเตอร์เคลื่อนไปถึงจุดที่ยืดออกไปหรือหดเข้ามาจนสุดแล้วตัวสลิปคลัทช์ของมอเตอร์จะทำงานโดยมีเสียงดังคลิก จากนั้นมอเตอร์ก็จะหยุดทำงานทันทีและถ้าหากแขนของแอคทูเอเตอร์เกิดมีอาการฝืดจนไม่สามารถเคลื่อนอย่างเป็นปกติได้ดังเดิมเราสามารถจะแก้ไขได้โดยถอดตัวมอเตอร์ออกจากกระบอกของแอคทูเอเตอร์แล้วใช้ปลายไขควงสอดเข้าไปในช่องด้านล่างของตัวกระบอกที่ต่อเข้ากับมอเตอร์(ซึ่งมีแกนกลางที่หมุนได้และมีร่องสำหรับสวมเข้ากับแกนรูปกากบาทที่ต่อมาจากมอเตอร์) ให้ปลายข่องไขควงอยู่ในร่องของแกนกลางแล้วหมุนไขควงจนกระทั่งแขนของแอคทูเอเตอร์เริ่มเคลื่อนที่ได้จากนั้นให้ใส่มอเตอร์กลับเข้าไปที่เดิมก็จะใช้งานได้ต่อไป

ข้อแนะนำในการติดตั้งลิมิตของแอคทูเอเตอร์นั้นควรจะตั้งเอาไว้ตรงจุดที่ผ่านตำแหน่งของดาวเทียมดวงสุดท้ายที่เราสามารถรับได้ชัดเจน โดยไม่จำเป็นต้องไปตั้งไว้ที่จุดซึ่งอยู่ก่อนลิมิตสวิตซ์ ของแอคทูเอเตอร์จะทำงานเสมอไป

การปรับตั้งมุมซีนิธ

การปรับตำแหน่งของ Polar axis สามารถกระทำได้โดยยกจานรับสัญญาณให้แหงนหน้าไปสู่จุดที่สูงสุดในท้องฟ้าหรือเรียกตำแหน่งนี้ว่ามุมซีนิธ (Zenith Arc) วิธีการก็คือหันด้านหน้าของจานลงไปทางทิศใต้ (เพราะประเทศไทยตั้งอยู่เหนือเส้นศูนย์สูตร) ให้ใช้เครื่องมุมวัดมุมวางบนแกนโพลาร์ ของจานรับสัญญาณจากนั้นให้ปรับแกนโพลาร์ของเมาท์ขึ้นไปจนกระทั่งได้มุมเงยที่ถูกต้องของพื้นที่บริเวณนั้น

การปรับแต่งมุมลาดเอียง

การปรับแต่มุมลาดเอียงนี้จะใช้ในการปรับมุมระหว่าง Polar axis กับเมาท์ของจานรับสัญญาณ ทำให้จานรับสัญญาณสามารถที่จะทำการกวาดในคลาคออบิทได้อย่างถูกต้องมากยิ่งขึ้น มุม Declination ณ ที่ตำแหน่งใดๆของพื้นที่ที่จะพิจารณา จากตำแหน่งเส้นรุ้งที่พาดผ่านพื้นที่นั้น ในการปรับ Declination ทางผู้ผลิตจะทำการปรับมาให้ตรงกับพื้นที่ที่จะนำจานรับสัญญาณไปติดตั้งหรือหากเราต้องการจะคำนวณหามุม Declination ด้วยตนเองก็สามารถทำได้โดยใช้สูตร

Declination angle = Arc Zenith - Polar axis angle

การที่เราสามารถปรับมุม Declination ที่ Polar axis ของเมาท์ได้นั้น จะทำให้การกวาดหาตำแหน่งของดาวเทียมบนคลาคออบิทกระทำได้มากและถูกต้องยิ่งขึ้นเพียงแต่เราปรับค่าของมุม Declination ให้เหมาะสมถูกต้องก็พอ การปรับทำได้โดยนำเครื่องวัดมุมวางบนแผ่นเพลทที่ติดอยู่ด้านหลังของจานรับสัญญาณ (เพลทนี้จะขนานกับขอบของจาน) จากนั้นปรับมุมเงยของจานให้มีค่าเท่ากับมุมของ Polar axis บวกกับค่าของมุม Delcination เมื่อสังเกตุให้ดีจะเห็นว่าจานรับสัญญาณจะเอียงลงมาอีกเล็กน้อย

วิธีการกวาดหาสัญญาณดาวเทียม

ในส่วนของเครื่องรับสัญญาณนั้น ปัจจุบันถูกผลิตออกมาหลายรูปแบบและหลายลักษณะการใช้งาน บางครั้งเครื่องได้ถูกออกแบบให้หาตำแหน่งของดาวเทียมและโปรแกรมโดยตัวผู้ใช้ บางเครื่องก็มีความสามารถมาก เพียงแต่ให้เราหาตำแหน่งและกำหนดให้ดวงหนึ่งเป็น Upper และอีกดวงหนึ่งเป็น Lower ให้กับเครื่องเท่านั้น จากนั้นก็จะมีระบบอัตโนมัติในการหาตำแหน่งที่ตรงที่สุดของจานพร้อมทั้งปรับโพลาไรซ์ให้ตรงกับดาวเทียมให้มากที่สุดอีกด้วย เมื่อแน่ใจว่าสัญญาณ

ภาพที่รับได้ดีที่สุดแล้ว ก็จะทำการโปรแกรมค่าต่างๆ ลงในเครื่องรับได้ทันทีและเมื่อเราเปิดเครื่องรับอีกครั้ง ระบบอัตโนมัติต่างๆ ก็จะดำเนินงานให้เราทั้งหมด

ความเที่ยงตรงในการทำงานของระบบอัตโนมัติเหล่านี้ บางครั้งก็เชื่อถือไม่ได้เช่นกับซึ่งบ่อยครั้งมักจะทำงานผิดพลาดจนต้องไปทำการปรับแต่งด้วยตนเองอีกทางออกที่ดีคือในการติดตั้งครั้งแรก ให้ทำการติดตั้งและปรับละเอียดด้วยมือเมื่อได้ผลเป็นที่หน้าพอใจแล้วจึงทำการบันทึกค่าต่างๆลง

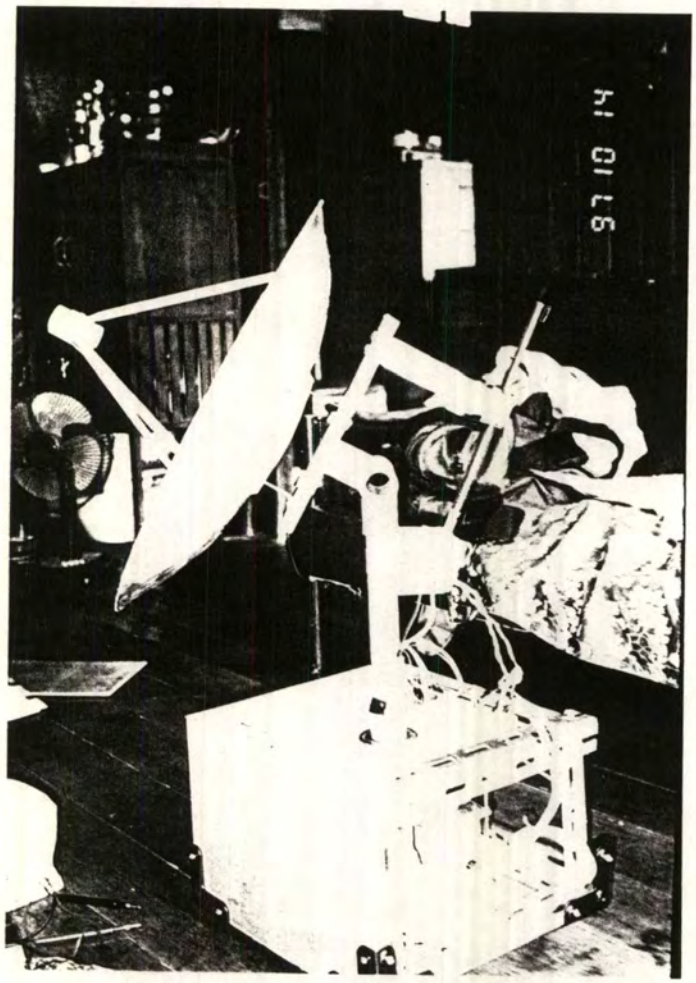
ขั้นตอนต่อไปนี้ จะเป็นสรุปวิธีการที่จะต้องปฏิบัติเพื่อจะทำการติดตั้งจานรับสัญญาณดาวเทียมให้ตรงกับตำแหน่งของดาวเทียมค้างฟ้าโดยมีขั้นตอนดังต่อไปนี้

1. สำรวจพื้นที่ที่จะทำการติดตั้งจานรับสัญญาณดาวเทียม
2. ติดตั้งเมาท์และจานรับสัญญาณเข้ากันพื้นที่
3. ปรับมุมกวาดและมุมเงยของจานโดยอาศัยข้อมูลจากตารางตำแหน่งดาวเทียม
4. โปรแกรมตำแหน่งที่รับสัญญาณได้ดีที่สุดลงในหน่วยความจำแล้วทำการปรับมุมกวาดและมุมเงยของดาวเทียมดวงต่อไป

บทที่ 3

การออกแบบและการสร้าง

สำหรับการออกแบบและการสร้างโครงงานชุดนี้ จะแบ่งออกได้เป็น 2 ส่วน คือ ส่วนที่เป็นฮาร์ดแวร์ซึ่งประกอบด้วย วงจรจ่ายกำลังงาน (POWER SUPPLY) , วงจรขับสเต็ปปิ้งมอเตอร์ (STEPPING DRIVER) , บอร์ดไมโครคอนโทรลเลอร์ (MICRO CONTROLLER BOARD) และชุดเมคคานิค (MECHANIC) ส่วนที่สองจะเป็นส่วนของซอฟต์แวร์ซึ่งจะเป็นภาษาแอสเซมบลี้ของ MCS - 51



รูปที่ 3.1 ภาพถ่ายชุดจำลองเครื่องควบคุมจานรับสัญญาณดาวเทียมค้างฟ้า

3.1 การออกแบบและการสร้างส่วนประกอบทางด้านฮาร์ดแวร์

3.3.1 วงจรจ่ายกำลังงาน

หลักการทำงาน

วงจรจ่ายกำลังงานนี้จะใช้รูปแบบของวงจรจ่ายกำลังงานโดยทั่วๆ ไป ซึ่งจะใช้ไอซีเร็กกูเลต (REGULATE IC) เป็นตัวควบคุมระดับแรงดันให้คงที่ ซึ่งต้องการระดับแรงดัน 5 โวลต์ ดังนั้น ไอซีเร็กกูเรตนี้จะใช้เบอร์ 7805 แต่เนื่องจากในการใช้งานจำเป็นต้องใช้กระแสที่จ่ายให้กับชุดขับมอเตอร์ด้วย ซึ่งต้องการปริมาณกระแสที่ค่อนข้างสูง จึงต้องใช้เพาเวอร์ทรานซีสเตอร์เป็นตัวเพิ่มกระแส โดยใช้ทรานซีสเตอร์เบอร์ MJ2955 นอกจากนั้นยังมีส่วนของตัวเก็บประจุที่ช่วยรักษาระดับแรงดันให้คงที่ด้วย

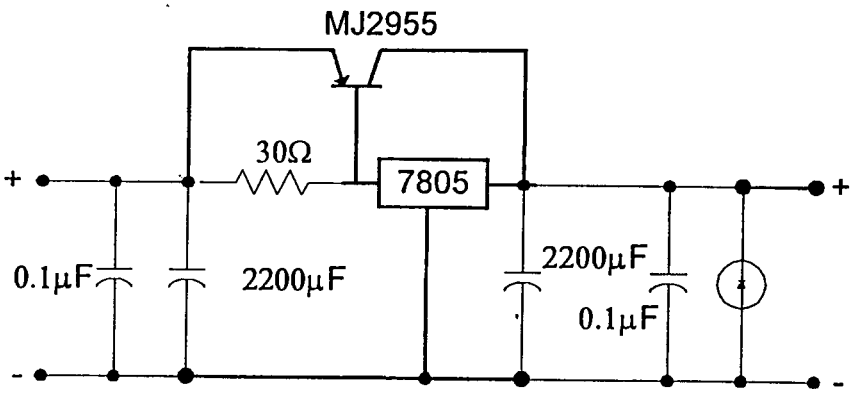
การสร้างวงจร

ในการสร้างหรือประกอบวงจรนั้น เมื่อเราทำการออกแบบวงจรเสร็จแล้ว นำไปออกแบบลายวงจรโดยใช้คอมพิวเตอร์ในที่นี้จะใช้โปรแกรม AUTOCAD เป็นตัวออกแบบ เมื่อได้ลายวงจรให้นำมาคัดลอกลงในแผ่นวงจรพิมพ์และทำการสร้าง เป็นแผ่นวงจรพิมพ์ต่อไป ก่อนที่จะทำการเจาะแผ่นวงจรพิมพ์เพื่อเตรียมประกอบอุปกรณ์

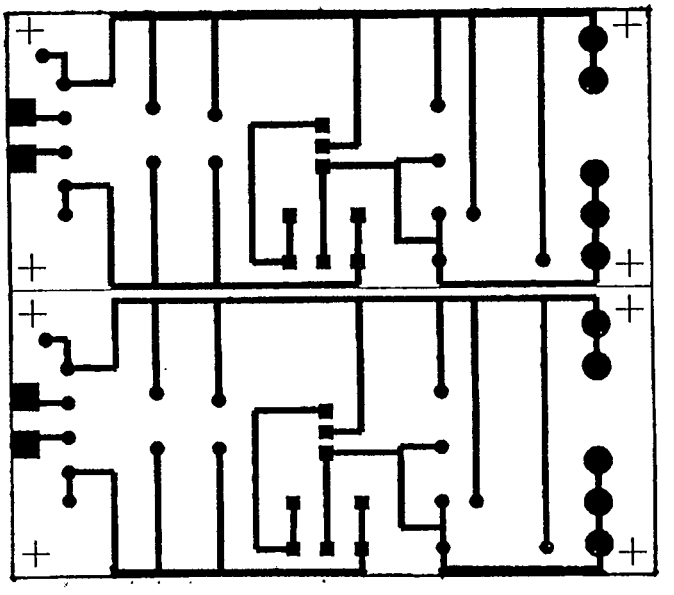
หลังจากเตรียมแผ่นวงจรพิมพ์แล้ว ให้เริ่มประกอบวงจร โดยเริ่มจากอุปกรณ์ที่มีความสูงน้อย เช่น ตัวต้านทาน , ไดโอด ไปหาอุปกรณ์ ที่มีความสูงมากขึ้นตามลำดับ เช่น ทรานซีสเตอร์ , ตัวเก็บประจุ และไอซีเร็กกูเลต

รายการอุปกรณ์

อุปกรณ์		จำนวน	
1. ไอซีเร็กกูเลต เบอร์ 7805		1	ตัว
2. ทรานซีสเตอร์ ชนิด PNP เบอร์ MJ 2955		1	ตัว
3. ตัวต้านทาน ขนาด 30 โอห์ม		1	ตัว
4. ตัวเก็บประจุ ขนาด 0.1 μF และ 2200μF	อย่างละ	2	ตัว
5. ไดโอด		1	ตัว



รูปที่ 3.2 แสดงรูปวงจรจ่ายกำลังงาน



รูปที่ 3.3 แสดงลายวงจรพิมพ์ของวงจรจ่ายกำลังงาน

3.1.2 วงจรขับสเต็ปปิ้งมอเตอร์

หลักการทำงาน

วงจรขับสเต็ปปิ้งมอเตอร์จะประกอบไปด้วยอุปกรณ์หลักๆ คือ ออปโตไอโซเลเตอร์ - ทรานซีสเตอร์ ซึ่งจะทำหน้าที่คล้ายกับสวิตช์ ปิด - เปิด ให้มีกระแสไหลผ่านไปยังขดลวดของมอเตอร์ สาเหตุที่ใช้ออปโตไอโซเลเตอร์เนื่องจากจะทำงานโดยใช้แสงเป็นตัวควบคุมซึ่งจะไม่มีการต่อถึงกันในทางไฟฟ้าจึงทำให้ความต้านทานระหว่างวงจรมีค่าสูงถึงประมาณ 10^{12} โอห์ม จึงเป็นการป้องกันความเสียหายของวงจรที่อาจเกิดขึ้นได้จากการลัดวงจร ทั้งนี้ได้ใช้ออปโตไอโซเลเตอร์เบอร์ 4N25 ซึ่งในการทำงานของวงจรจะรับสัญญาณอินพุตมาจากพอร์ต 8255 ของชุดคอนโทรลเลอร์ แล้วนำมาสั่งการให้สวิตช์ทำการปิด - เปิดวงจร และส่งสัญญาณไปยังทรานซีสเตอร์

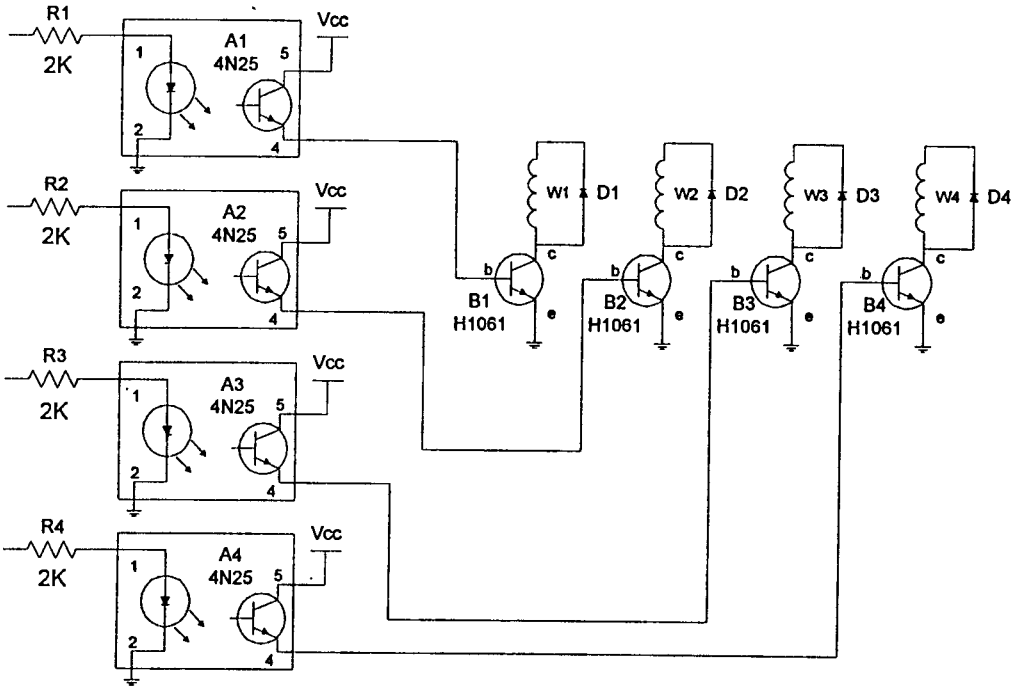
ซึ่งทำหน้าที่ในการป้อนกระแสไปขับสเต็ปปิ้งมอเตอร์ให้หมุนไปตามต้องการ ส่วนของไดโอดที่ต่อคร่อมอยู่กับขดลวดของมอเตอร์นั้นจะทำหน้าที่ในการป้องกันกระแสไหลย้อนกลับซึ่งเกิดจากขดลวดของมอเตอร์เมื่อทรานซีสเตอร์หยุดทำงาน เป็นการป้องกันไม่ให้ทรานซีสเตอร์เกิดความเสียหายได้

การสร้างวงจร

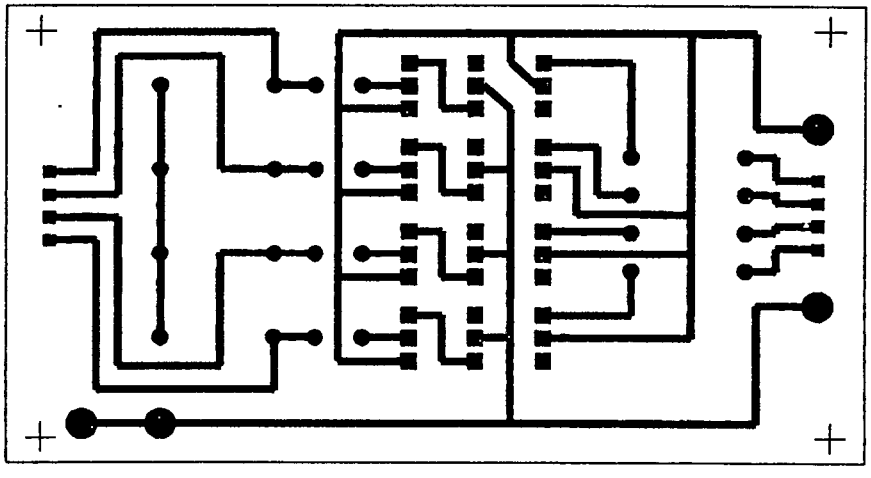
การสร้างจะเริ่มจากการออกแบบลายวงจรจากวงจรที่ได้ออกแบบไว้แล้วนำมาสร้างวงจรพิมพ์ในลักษณะเดียวกันกับวงจรจ่ายกำลังงาน จากนั้นเริ่มประกอบอุปกรณ์โดยเริ่มจากอุปกรณ์ที่มีความสูงน้อย เช่น ตัวต้านทาน ไดโอด จากนั้นจะเป็นอุปกรณ์ที่มีความสูงมากขึ้น จำพวก ออปโตไอโซเลเตอร์ , ทรานซีสเตอร์ โดยในการติดตั้ง ออปโตไอโซเลเตอร์ ควรจะใช้ซ็อกเก็ตรองรับ เนื่องจากจะเป็นการป้องกันไม่ให้ออปโตไอโซเลเตอร์เกิดความเสียหายเนื่องมาจากการบัดกรีและเป็นการสะดวกในการเปลี่ยนแปลงและตรวจซ่อม

รายการอุปกรณ์

อุปกรณ์	จำนวน	
1. ตัวต้านทานขนาด 2K	4	ตัว
2. ออปโตไอโซเลเตอร์ เบอร์ 4N25	4	ตัว
3. ทรานซีสเตอร์ เบอร์ H1061	4	ตัว
4. ไดโอด	4	ตัว



รูปที่ 3.4 แสดงรูปวงจรขับสเต็ปปิ้งมอเตอร์



รูปที่ 3.5 แสดงลายวงจรของชุดขับสเต็ปปิ้งมอเตอร์

3.1.3 บอร์ดไมโครคอนโทรลเลอร์

หลักการทำงาน

ในส่วนของบอร์ดไมโครคอนโทรลเลอร์นั้นหากจะทำการออกแบบและทำการสร้างเองนั้นจะทำให้ไม่คุ้มค่าในเชิงธุรกิจเนื่องจากมีชุดที่ผลิตออกมาเพื่อจำหน่ายแล้วสามารถใช้งานได้ดี โดยจะใช้ชุดไมโครคอนโทรลเลอร์ที่ผลิตโดยบริษัทศิลารีเสิร์ดจำกัด รุ่น V - 3155 ซึ่งเป็นชุดไมโคร

คอนโทรลเลอร์ที่ได้พัฒนามาจนมีประสิทธิภาพสูงและเป็นที่นิยมใช้กันอย่างกว้างขวางโดยในบอร์ดดังกล่าวมีรายละเอียดดังต่อไปนี้

คุณสมบัติของบอร์ด V-3155

CPU	80C31 (40 PIN-DIP OF MCS-51)
CLOCK	11.0592MHz
MEMORY	0/32K SOCKET (PROGRAM)
PORT	8 BIT (PORT1 OF MCS-51)
	4 BIT (/INT0 , /INT1 , /T0 , T1 OF MCS-51)
	24 BIT (PORT OF 8255)
LED	1 POWER LED
SWITCH	1 RESET SWITCH
CONNECTER	40 PIN MCS-51 SYSTEM BUS
	26 PIN 8255
	3 PIN RS-232
	2 PIN 5V DC
JUMPER	2 WAY JUMPER FOR /EA SELECT (INT,EXT)
	2 WAY JUMPER FOR MEMORY SOCKET (8- 16K,32K)
POWER SUPPLY	5V DC CURRENT 210mA (WITH 27C256 EPROM)
PCB SIZE	8.8 X 7.1 cm

รายละเอียดต่างๆบนบอร์ด

ไมโครคอนโทรลเลอร์บนบอร์ด V-3155 จะใช้ไมโครคอนโทรลเลอร์เบอร์ 80C31 เป็นหลัก แต่อย่างไรก็ตามผู้ใช้สามารถใช้กับไมโครคอนโทรลเลอร์เบอร์ต่างๆ ในตระกูล MCS-51 ที่เป็นแบบ 40 PIN DIP ได้ทั้งหมดเช่น 8032 8751 8752 ซึ่งจะทำให้ได้คุณสมบัติเป็นไปตามโครงสร้างของเบอร์นั้นๆ การเลือก JUMPER / EAจะใช้เพื่อการเลือกให้ทำงานจาก ROM หรือ EPROM ภายในตัวไมโครคอนโทรลเลอร์ หรือเลือกจากอีพรอมภายนอก ทั้งนี้การเลือก /EA ในตำแหน่ง INT จะใช้กับไมโครคอนโทรลเลอร์ที่มีโปรแกรมอยู่ภายในเท่านั้น ซึ่งปกติจะเป็น 8751 หรือ 8752 (กรณี 8051 หรือ 8052 มีรอมอยู่ภายในก็จริงแต่ในทางปฏิบัติ การโปรแกรมรอมภายในของ 8051 หรือ 8052 จะทำได้จากโรงงานผู้ผลิตเท่านั้นและรันที่จำนวนมากๆ เพราะฉะนั้นในทางปฏิบัติไม่ควรจะมีเบอร์ 8051 หรือ 8052 ขายอยู่ในท้องตลาดเลย

หน่วยความจำและการเลือก Jumper

หน่วยความจำที่ใช้ได้เป็นแบบ ROM(EPROM) สามารถเลือกเบอร์ต่างๆ ได้ตามความต้องการ โดยใช้ JUMPER ที่อยู่ใต้ 74LS373 ซึ่งสามารถเลือกหน่วยความจำได้ตั้งแต่ 8K-32K จะกระทำโดยปรับตัว JUMPER ให้ตรงกับช่องที่ระบายที่ขาวทึบตรงตามเบอร์ที่ต้องการ

พอร์ตอนุกรม

เป็นพอร์ตที่ใช้ในการสื่อสารอนุกรมของบอร์ด V-3155 ซึ่งสามารถเลือกใช้หรือไม่ก็ได้ ถ้าต้องการก็ใช้ใช้ร่วมกับชิพเบอร์ MAX232 และใช้งานที่ขั้วต่อแบบ 3 ขา

SYSTEM BUS และ PORT 8255 BUS

SYSTEM BUS ของบอร์ด V-3155 จะมใช้ขาสัญญาณจากกตัวไมโครคอนโทรลเลอร์ เป็นขนาด 40 ขา ซึ่งใช้สำหรับขยายระบบตามต้องการทั้งนี้ยังสามารถดึงพอร์ต 1 และขาสัญญาณ / INT0 , /INT1 , /T0 , /T1ออกมาเพื่อใช้งานประยุกต์ใดๆ ก็ได้ ส่วนพอร์ต 8255 บัสซึ่งใช้ชิพเบอร์ 8255 ซึ่งทำหน้าที่เป็นพอร์ตอินพุตเอาท์พุตมีตำแหน่งแอสเดรสดังนี้

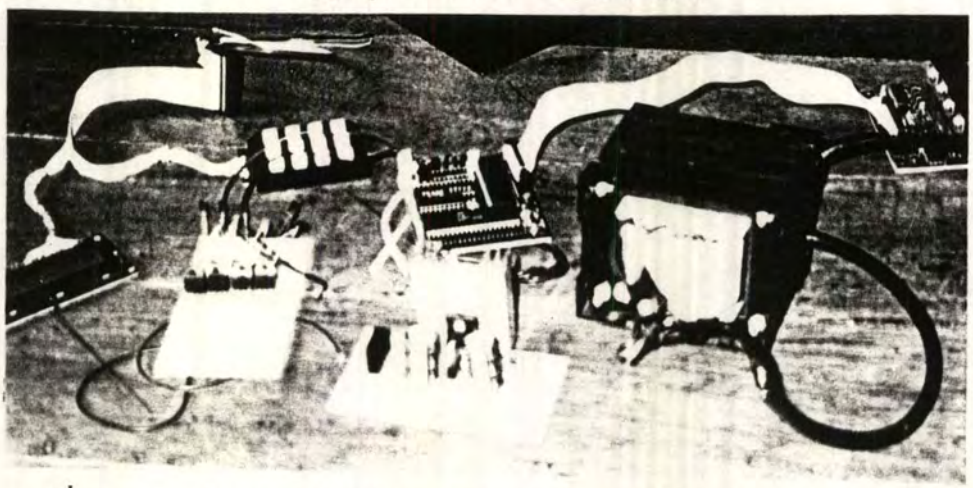
USER PORT1	แอสเดรส 0000H+8255 Offset addr = actual
PORT A	ตำแหน่งแอสเดรส 0000H +00H = 0000H
PORT B	ตำแหน่งแอสเดรส 0000H + 01H = 0001H
PORT C	ตำแหน่งแอสเดรส 0000H + 02H = 0002H
MODE PORT	ตำแหน่งแอสเดรส 0000H + 03H = 0003H

ตารางที่ 3.1 แสดงแอดเดรสของอินพุตเอาท์พุตของ 8255

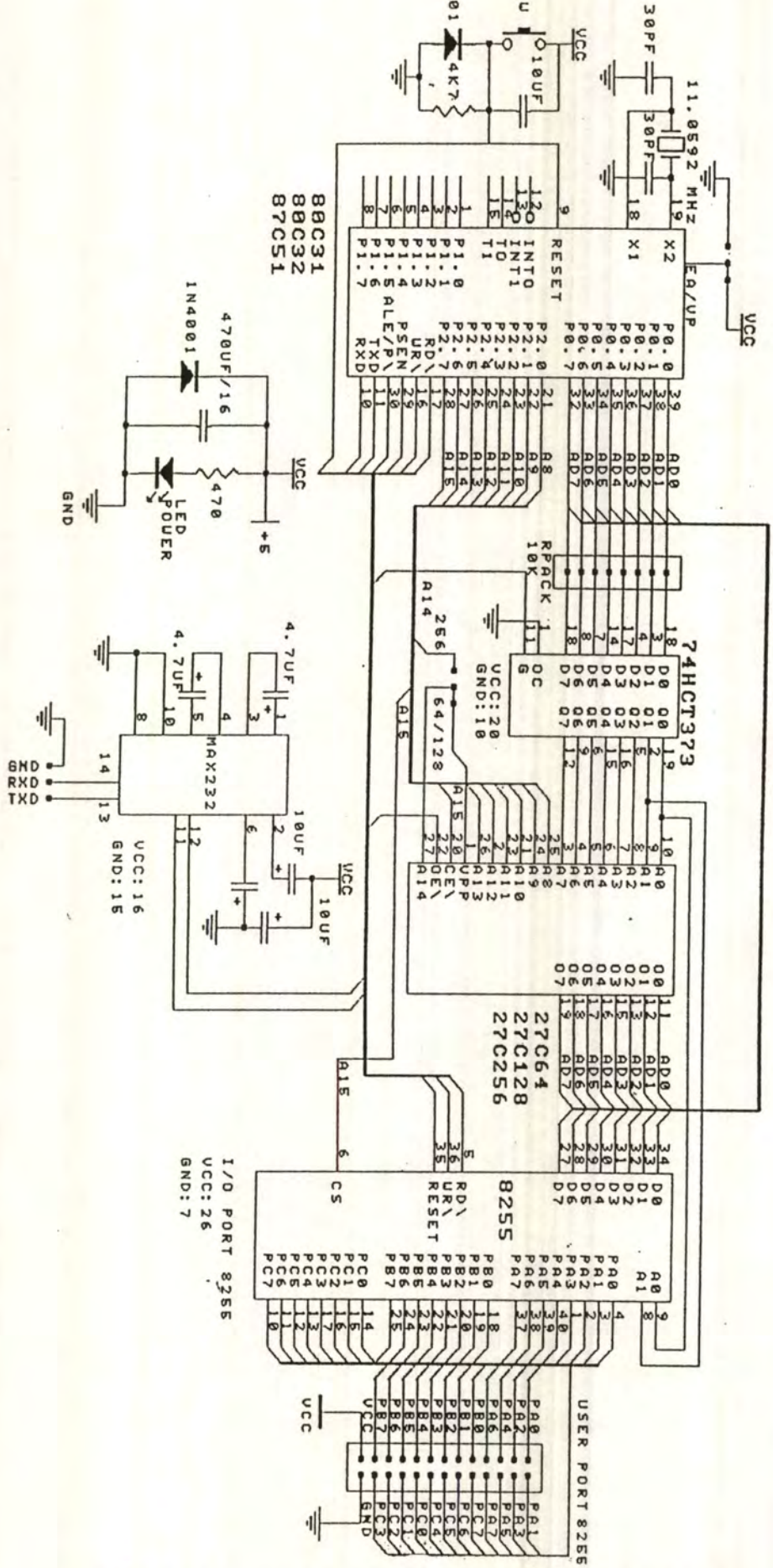
ก่อนที่จะใช้งานพอร์ต 8255 ผู้ใช้ต้องทำการกำหนดโหมดการทำงาน (Configulation) ของพอร์ต A,B และ C ให้เป็นพอร์ อินพุตหรือเอาท์พุต โดยทำการเขียนค่า Control code ไปที่โหมดพอร์ต ซึ่งโหมดพอร์ต นี้สามารถเขียนได้เท่านั้นไม่สามารถอ่านได้

PORT A	PORT C UP	PORT B	PORT C LOW	C O N T R O L CODE
OUTPUT	OUTPUT	OUTPUT	OUTPUT	80H
OUTPUT	OUTPUT	OUTPUT	INPUT	81H
OUTPUT	OUTPUT	INPUT	OUTPUT	82H
OUTPUT	OUTPUT	INPUT	INPUT	83H
OUTPUT	INPUT	OUTPUT	OUTPUT	88H
OUTPUT	INPUT	OUTPUT	INPUT	89H
OUTPUT	INPUT	INPUT	OUTPUT	8AH
OUTPUT	INPUT	INPUT	INPUT	8BH
INPUT	OUTPUT	OUTPUT	OUTPUT	90H
INPUT	OUTPUT	OUTPUT	INPUT	991H
INPUT	OUTPUT	INPUT	OUTPUT	92H
INPUT	OUTPUT	INPUT	INPUT	93H
INPUT	INPUT	OUTPUT	OUTPUT	98H
INPUT	INPUT	OUTPUT	INPUT	99H
INPUT	INPUT	INPUT	OUTPUT	9AH
INPUT	INPUT	INPUT	INPUT	9BH

ตารางที่ 3.2 แสดงการควบคุมโหมดการทำงานของ 8255



รูปที่ 3.6 ภาพถ่ายแสดงส่วนต่างๆ ประกอบด้วย บอร์ดคอนโทรลเลอร์, วงจรขับสเต็ปปิ้งมอเตอร์, วงจรจ่ายกำลังงาน และชุด LCD MODULE



รูปที่ 3.7 แสดงวงจรของบอร์ดไมโครคอนโทรลเลอร์ MCS - 51

3.1.4 ชุดเมคคานิค (MECHANIC)

ชุดเมคคานิคหรือส่วนที่เป็นกลไกจะทำงานในลักษณะการเคลื่อนที่ไปตามคำสั่งของชุดควบคุม ในการออกแบบและการสร้างสามารถแยกออกได้ 3 ส่วน ดังนี้

ส่วนฐานรองรับ

ส่วนนี้จะเป็นส่วนที่ต้องรองรับการติดตั้งจากอุปกรณ์ที่เป็นชุดเมคคานิคทั้งหมด ซึ่งจำเป็นที่จะต้องมีความมั่นคงแข็งแรง แต่ทั้งนี้จะต้องสะดวกต่อการเคลื่อนย้ายเพื่อไปติดตั้งในที่ใหม่ ตามวัตถุประสงค์ของโครงงาน

วัสดุและอุปกรณ์

- โครงเหล็ก
- ไม้อัดหนาประมาณ 1 ซม. ใช้รองรับเสาแกนกลาง
- ไม้อัดบางใช้สำหรับครอบโครงเหล็ก
- ตลับลูกปืน (BEARING)
- นัตและสกรูขนาดต่างๆ

ขั้นตอนในการประกอบ

- นำโครงเหล็กมาประกอบเป็นรูปทรงสี่เหลี่ยมผืนผ้าให้มั่นคง
- นำแผ่นไม้อัดหนาประมาณ 1 ซม. ซึ่งได้เจาะรูขนาดเท่าตลับลูกปืนจำนวน 2 แผ่น มาติดตั้งทั้งส่วนล่างและส่วนบนของโครงเหล็ก
- ติดตลับลูกปืนเข้ากับรูของไม้อัดที่เตรียมไว้
- ส่วนของไม้อัดบางให้ใช้ประกอบเป็นลักษณะกล่องครอบด้านนอกของโครงเหล็ก ซึ่งขั้นตอนนี้จะประกอบหลังจากติดตั้งชุดควบคุมมุมกวาดเสร็จสิ้นแล้ว

ชุดควบคุมทางด้านมุมกวาด

หลักการทำงาน

ในส่วนของชุดควบคุมในทางมุมกวาดนั้นจำเป็นจะต้องใช้มอเตอร์ที่มีขนาดใหญ่ เพราะเป็นชุดที่ติดอยู่กับฐานรองรับทางด้านล่าง ในการหมุนจะต้องออกแรงหมุนส่วนประกอบทั้งหมดที่

เป็นเมคคานิคที่อยู่ถัดขึ้นมา ตั้งแต่ เสาแกนกลาง ข้อต่อต่างๆ ชุดควบคุมการหมุนในแนวมุมเงย รวมทั้งจานรับสัญญาณ โดยจะสามารถหมุนได้รอบตัวเองหรือ 360 องศา โดยในการใช้มอเตอร์ขับเคลื่อนจะใช้เฟืองซึ่งมีอัตราทด 2 : 1 เป็นการเชื่อมต่อระหว่างมอเตอร์กับเสาแกนกลาง ทั้งนี้เพื่อเป็นการทดอัตรากำลังงาน

ขั้นตอนการประกอบ

- ติดตั้งเฟืองไว้ที่ส่วนของเสาแกนกลางในบริเวณที่อยู่ในโครงเหล็ก
- ติดตั้งสเต็ปปิ้งมอเตอร์ไว้กับไม้อัดหนาแผ่นล่างโดยให้เฟืองของมอเตอร์ขบกับเฟืองของเสาแกนกลางจนสนิท
- ประกอบท่อ PVC เข้ากับเสาไม้แกนกลางให้แน่น

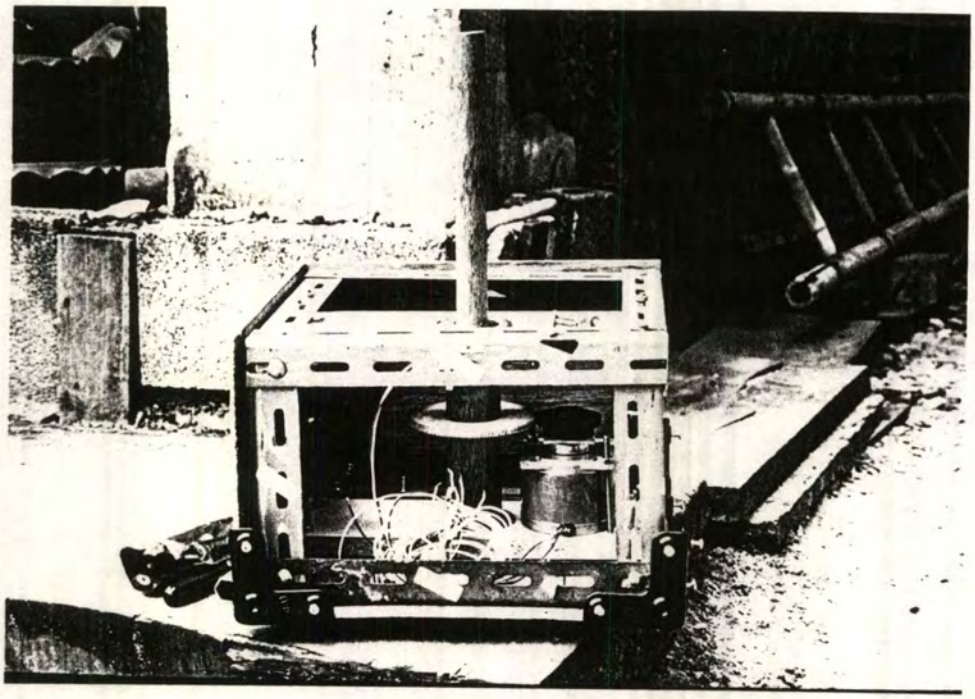
ชุดควบคุมด้านมุมเงย

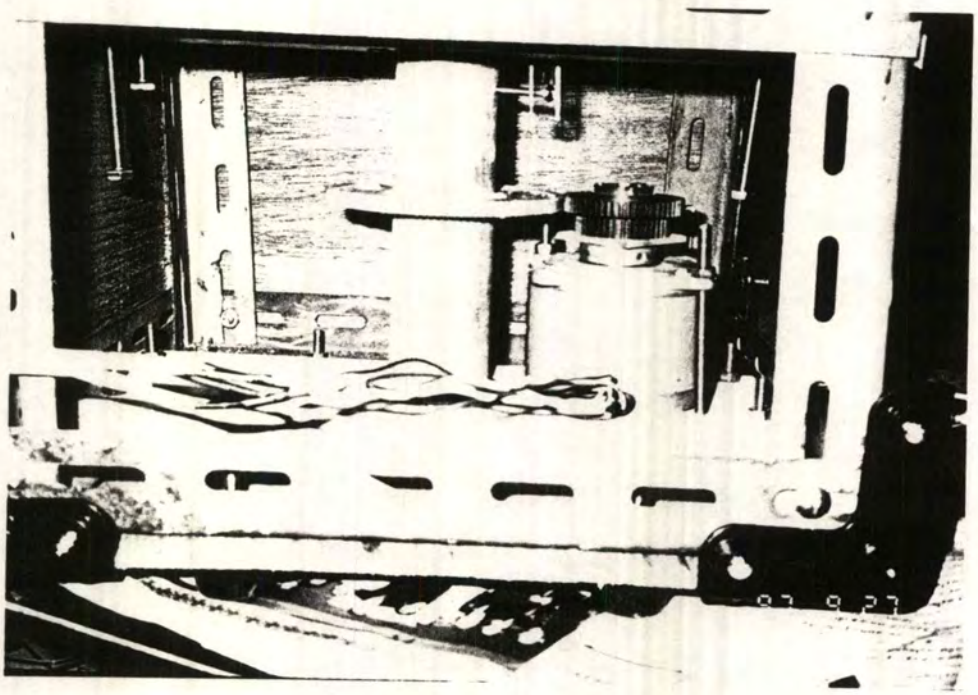
หลักการทำงาน

ในส่วนของมุมเงยจะไม่ใช้การขับเคลื่อนจากมอเตอร์โดยตรงแต่จะออกแบบโดยการควบคุมการเคลื่อนที่จากทางด้านข้างซึ่งจะใช้แท่งเกลียวซึ่งมีขนาดร่องเกลียวที่คงที่ในการขับยันให้ฐานรองรับจาน เกิดการเอียงตั้งแต่มุม 0 - 90 องศา ได้

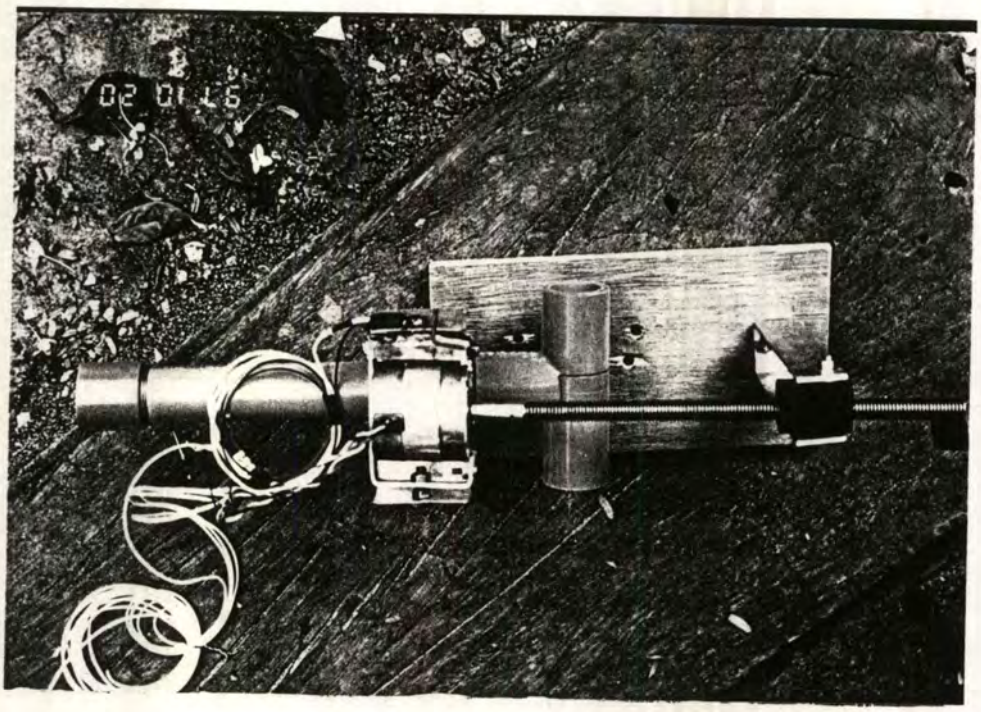
ขั้นตอนการประกอบ

- ประกอบข้อต่อ PVC ให้หนาแน่น และสามารถปรับระดับได้ตั้งแต่ 0 - 90 องศา
- ประกอบแผ่นเหล็กดัดยึดมอเตอร์ติดกับเสาแกนกลาง
- ประกอบสเต็ปปิ้งมอเตอร์เข้ากับแผ่นเหล็กดัด
- ประกอบชุดร่องเกลียวและบล็อกเข้ากับมอเตอร์
- ประกอบแผ่นไม้อัดรองรับจานกับบล็อกส่วนปลายของร่องเกลียว
- ติดตั้งจานรับจำลองลงบนแผ่นไม้อัดรองรับ





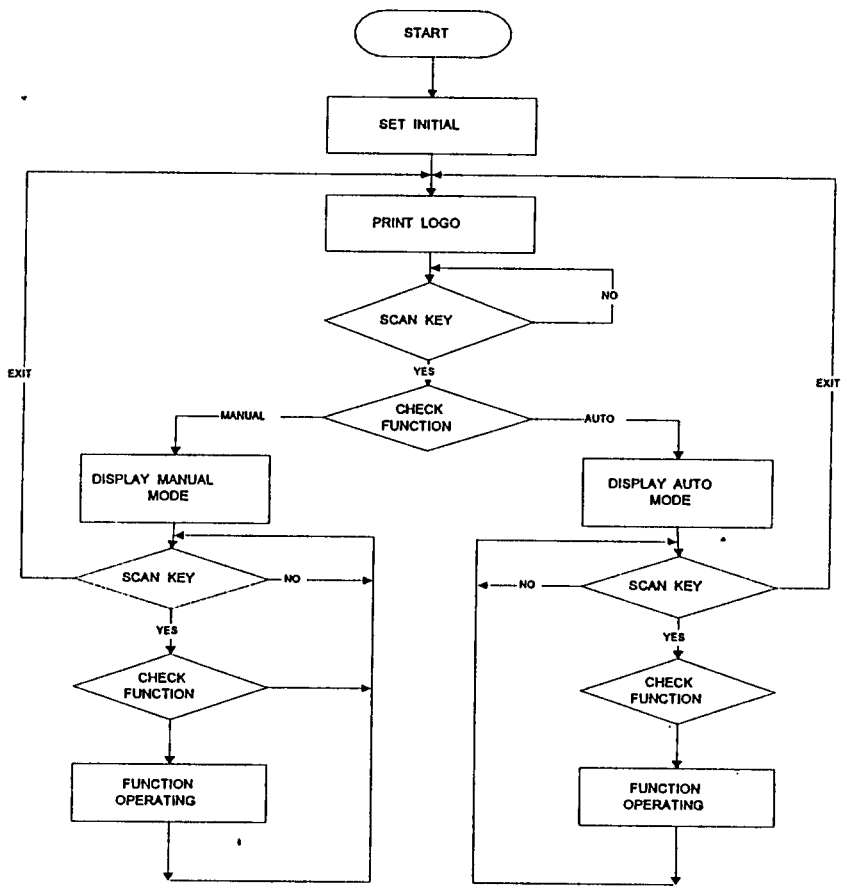
รูปที่ 3.8 ภาพถ่ายแสดงส่วนประกอบชุดควบคุมทางด้านมุมกวาด



รูปที่ 3. 9 ภาพถ่ายแสดงส่วนประกอบชุดควบคุมแนวมุมเงย

3.2 การออกแบบส่วนประกอบทางด้านซอฟท์แวร์

ในการออกแบบ และ สร้างในส่วนของซอฟท์แวร์ ซึ่งเป็นการเขียนโปรแกรมภาษาแอสเซมบลี้ของ MCS - 51 นั้น จะเป็นการสั่งการให้สเต็ปปิ้งมอเตอร์หมุนไปทางด้านหน้าหรือหมุนกลับมาอีกด้านตามความต้องการของผู้ออกแบบได้ซึ่งจะนำคุณสมบัติดังกล่าวมาใช้ประโยชน์ในการควบคุมการเคลื่อนที่ของทั้งสองมุม โดยในการเขียนโปรแกรมดังกล่าวนั้นจะทำการเขียนลงในคอมพิวเตอร์ส่วนบุคคล (PC) และทำการปรับปรุงแก้ไขจนกระทั่งวงจรสามารถที่จะทำงานได้เป็นอย่างดี จึงจะทำการบันทึกโปรแกรมดังกล่าวเข้าไปเก็บไว้ในหน่วยความจำของชุดควบคุมเพื่อใช้งานต่อไป โดยโฟลว์ชาร์ตแสดงการทำงานของโปรแกรมดังรูปที่ 3.6



รูปที่ 3.10 โฟลว์ชาร์ตแสดงการทำงานของโปรแกรมควบคุม

บทที่ 4

การทดลองและผลการทดลอง

หลังจากที่ทำการสร้างส่วนประกอบของชุดโครงงานเป็นที่เรียบร้อยแล้ว ขั้นตอนต่อไปจะเป็นการทดสอบการทำงาน โดยในที่นี้เพื่อเป็นการสะดวกในการตรวจสอบจะทำการแยกการตรวจสอบทีละส่วนตามลำดับดังต่อไปนี้

1. วงจรจ่ายกำลังงาน (POWER SUPPLY)

หลังจากที่ทำการประกอบวงจรเสร็จเรียบร้อยแล้วจะเป็นขั้นตอนของการทดลอง โดยเริ่มจากการป้อนไฟเข้าไปในวงจร แล้วทำการวัดทางด้านเอาท์พุตว่ามีแรงดันขนาด 5 โวลต์ หรือไม่ หากไม่ได้ขนาดตามที่ต้องการให้ทำการตรวจสอบวงจรโดยเริ่มจากส่วนของไอซีเร็กกูเลท ทรานซีสเตอร์ และไดโอด แต่ถ้าหากพบว่ามีแรงดันตามที่กำหนด แสดงว่าการทำงานของวงจรเป็นไปตามปกติ ซึ่งจากการทดสอบในโครงงานนี้จะพบว่าสามารถจ่ายกระแสไฟได้เป็นปกติ

2. วงจรขับสเต็ปปิ้งมอเตอร์ (STEPPING DRIVER)

ในการทดสอบวงจรขับสเต็ปปิ้งมอเตอร์จะเริ่มจากการจ่ายแรงดัน DC ขนาด 5 โวลต์ ให้วงจร แล้วทดลองนำไปขับสเต็ปปิ้งมอเตอร์ หากไม่สามารถขับมอเตอร์ให้หมุนได้แสดงว่าวงจรเกิดความผิดพลาดให้ทำการตรวจสอบที่วงจรใหม่ โดยเริ่มจากตัวออปโตไอโซเลเตอร์ และทรานซีสเตอร์ หากพบว่ามีความผิดปกติให้ทำการเปลี่ยนแปลงหรือแก้ไข แต่ในวงจรที่ใช้งานนี้พบว่าสามารถทำงานได้เป็นปกติ

3. บอร์ดไมโครคอนโทรลเลอร์ (MICROCONTROLLER BOARD)

ในการทดลองบอร์ดไมโครคอนโทรลเลอร์นั้นสามารถทดลองได้โดยใช้คอมพิวเตอร์ส่วนบุคคล (PC) ซึ่งจะใช้ RS232 เป็นตัวเชื่อมต่อ ซึ่งจะสามารถตรวจสอบการทำงานของโปรแกรมได้ว่ามีความผิดพลาดหรือไม่ และสามารถทำการแก้ไขได้เลย ซึ่งจากการทดลองพบว่าสามารถทำการโปรแกรมได้เป็นอย่างดี

4. ชุดเมคคานิค (MECHANIC)

สำหรับการทดลองชุดเมคคานิคจะแยกการทดสอบออกเป็น 2 ส่วน คือส่วนที่ทำการควบคุมทางด้านมุมกวาด และส่วนที่ควบคุมทางด้านมุมเงย โดยในส่วนของมุมกวาดจะทำการทดลองโดยทำการหมุนเสาแกนกลางและสังเกตว่าสามารถที่จะหมุนไปได้รอบตัวเองหรือ 360 องศา หรือไม่ หากพบว่ามีการติดขัดให้รีบทำการแก้ไขทันที ซึ่งในชุดที่ประกอบขึ้นมานี้ยังไม่พบปัญหาแต่อย่างใด และส่วนที่สองคือชุดที่ควบคุมด้านมุมเงยโดยการหมุนที่ร่องเกลียวซึ่งยึดติดอยู่กับแกนของมอเตอร์ว่าสามารถที่จะบังคับให้แผ่นไม้อัดรองรับจานเปลี่ยนมุมจาก 0 - ๗๐ องศา ได้หรือไม่ หากไม่ได้ตามข้อกำหนดให้ทำการปรับปรุงแก้ไข ซึ่งพบว่าชุดที่จะใช้งานสามารถทำงานได้เป็นปกติ

5. ส่วนของโปรแกรม

ซึ่งเป็นฮาร์ดแวร์ที่ใช้ควบคุมการทำงานให้ทำการทดลองโดยใช้เครื่องคอมพิวเตอร์ส่วนบุคคลว่าสามารถที่จะทำงานได้ตามข้อกำหนดหรือไม่ หากพบว่ามีข้อผิดพลาดให้ทำการแก้ไขได้ทันที ซึ่งวงจรที่นำมาใช้งานได้ปรับปรุงจนสามารถใช้งานได้เป็นอย่างดี

หลังจากที่ทำการทดสอบทุกๆ ส่วนของชุดโครงงานแล้ว ให้นำวงจรแต่ละส่วนมาประกอบกัน จากนั้นป้อนไฟให้กับระบบ และทดลองใช้งานโดยการป้อนตำแหน่งของดาวเทียมที่ต้องการจะรับสัญญาณ แล้วสังเกตดูว่าชุดควบคุมสามารถที่จะหมุนไปชี้ยังตำแหน่งที่เราต้องการได้หรือไม่ ทั้งนี้จะมีส่วนของมอนิเตอร์ที่เป็นทั้ง LCD MODULE และส่วนของอุปกรณ์ที่ติดตั้งอยู่ในส่วนของฮาร์ดแวร์เป็นตัวแสดงตำแหน่งให้สามารถตรวจสอบได้

บทที่ 5
บทสรุป

ผู้จัดทำได้สร้างโครงงานนี้ขึ้นมาโดยมีจุดประสงค์ที่จะใช้เป็นเครื่องต้นแบบ หรือเพื่อให้สามารถนำไปใช้กับงานจริงได้ ตามประสบการณ์ทางด้านเทคนิคที่ได้รับการศึกษามา และจากการแนะนำรวมทั้งการสนับสนุนจากอาจารย์ที่เกี่ยวข้อง โดยเฉพาะอย่างยิ่งอาจารย์ที่ปรึกษาที่มีส่วนเป็นอย่างมากในการช่วยเหลือเป็นอย่างดีมาโดยตลอดจนกระทั่งออกมาเป็นผลสำเร็จ ซึ่งจากการปฏิบัติงานในครั้งนี้สามารถที่จะสรุปในส่วนของปัญหาอุปสรรค แนวทางในการแก้ไขรวมทั้งข้อเสนอแนะในการนำโครงงานชิ้นนี้ไปใช้งานหรือพัฒนาต่อไป โดยมีรายละเอียดดังต่อไปนี้

ปัญหาอุปสรรค

จากการสร้างโครงงานชิ้นนี้ขึ้นมาจนสำเร็จ ได้พบปัญหาอุปสรรคต่างๆ ที่เกิดขึ้นประกอบไปด้วย

1. ปัญหาเรื่องความผิดพลาดในการชี้ตำแหน่งของจานรับสัญญาณเดาวเทียม โดยพบว่าเกิดความผิดพลาดขึ้นทั้งทางด้านมุมกวาดและมุมเงย ซึ่งจะมีค่าความผิดพลาดอยู่ประมาณ 4 - 5 องศา ในแต่ละแนว ซึ่งสาเหตุในส่วนนี้เกิดจากการใช้มอเตอร์ที่ผ่านการใช้งานมาแล้วทำให้ประสิทธิภาพลดลง
2. ปัญหาเรื่องความล่าช้าในการสแกนของมอเตอร์ทั้งแนวมุมกวาดและมุมเงย ซึ่งทำให้ต้องเสียเวลาในการรอคอยนาน กว่าที่เครื่องควบคุมจะทำการชี้ตำแหน่งตามต้องการ
3. ในส่วนของชุดเมคคานิคไม่มีความมั่นคงแข็งแรงเพียงพอ โดยสามารถที่จะโยกคลอนได้เล็กน้อยซึ่งอาจจะเป็นอีกสาเหตุหนึ่งของความผิดพลาดในการชี้ตำแหน่ง

แนวทางแก้ไขปัญหา

1. ใช้มอเตอร์ที่ความละเอียดสูงขึ้นจะทำให้มีความละเอียดในการชี้ตำแหน่งเพิ่มขึ้นตามความต้องการ นอกจากนี้ยังสามารถแก้ไขปัญหาดังกล่าวได้โดยใช้ ดี.ซี. มอเตอร์ มาทำงานแทนสเต็ปปิ้งมอเตอร์ ซึ่งจะมีความแม่นยำในการชี้ตำแหน่งมากขึ้น แต่ทั้งนี้จะต้องมีการเปลี่ยนแปลงในส่วนของโปรแกรมควบคุมด้วย
2. เลือกใช้มอเตอร์ที่ใหม่ มีคุณภาพสูง รวมทั้งให้กำลังงานที่สูงขึ้น ทั้งนี้เพื่อเป็นการเพิ่มแรงบิดให้มากขึ้นจะมีผลทำให้ความเร็วในการสแกนเพิ่มขึ้น

3. ออกแบบและประกอบในส่วนของชุดเมคคานิคขึ้นมาใหม่โดยให้มีความแข็งแรงมั่นคง และหนาแน่นเพียงพอเพื่อลดความผิดพลาดในส่วนนี้ได้

การพัฒนา

จากที่ได้กล่าวไว้แล้วว่าในการสร้างโครงงานชุดนี้ขึ้นมามีจุดประสงค์ที่จะให้เป็นเครื่องต้นแบบและให้เกิดแนวความคิดแก่ผู้ที่มีความสนใจในการสร้างโครงงานที่ทำงานในลักษณะนี้ให้สามารถนำไปประยุกต์ใช้ได้อย่างมีประสิทธิภาพ แต่ทั้งนี้ก็สามารถที่จะนำโครงงานชิ้นนี้ไปประยุกต์ใช้งานจริงได้โดยทำการติดตั้งจานรับสัญญาณดาวเทียมเข้าไปแทนที่จานจำลอง แต่จานที่นำมาติดตั้งจะต้องมีขนาดและน้ำหนักที่ใกล้เคียงกับจานจำลอง จากนั้นให้เดินสายสัญญาณให้เรียบร้อย นอกจากนั้นจะต้องทำการเปลี่ยนแปลงและปรับปรุงในส่วนของโปรแกรมให้มีประสิทธิภาพสูงขึ้น

```
#include <reg51.h>

extern void lcdwi(unsigned char i);
extern void lcdwd(unsigned char d);

char putchar(char c);
void lcdwdd(char dd);
void lcdposition(unsigned char posi);
void lcdclr(void);
unsigned char Cursor;

char putchar(char c)           //   Putchar to LCD
  {
    if (c == '\n')             //  next line
      {
         if (Cursor<16)
           lcdposition(16);
         else
           lcdposition(0);
      }
   else
   lcdwdd (c);
   return (c);
  }


void lcdwdd(char dd)
  {
    if(Cursor == 16)
      lcdposition (16);
    else if (Cursor == 32)
      lcdposition (0);
    Cursor++;
    lcdwd(dd);
  }

void lcdposition(unsigned char posi)     /* Position LCD cursor */
  {
    Cursor = posi;
    if (posi < 16)
         lcdwi(posi + 0x80  );
    else
         lcdwi(posi - 16  + 0xc0 );
  }

void lcdclr(void)                    /* Clear LCD and home cursor */
  {
    int i;
    Cursor = 0;
    lcdwi(1);
    for(i=0;i<2000;i++){;;}
```

```
#include <reg51.h>
#include <stdio.h>

extern void Delayms(int del);
extern void lcdini(void);
extern void lcdclr(void);
extern void lcdposition(unsigned char posi);

void el_step_up(void);
void el_step_dn(void);
void az_step_l(void);
void az_step_r(void);
void autmode(void);
void display(unsigned char ind,unsigned char *ptr);
void setorg(void);
void set_sat(char i);
void findsat(char sat);
void sysinit(void);

sbit azifg = P1^0;
sbit elefg = P1^1;
extern bit gkeyol;
bit step1,step2,step3,step4;  // status flag
bit ele_l,ele_r,azi_u,azi_d;
unsigned char xdata *p;

int elecur;                   // current elevation value
int azicur;                   // current azimuth value

unsigned char elebuf[9];
unsigned char azibuf[9];

code char logo [] = "   SATELLITE     CONTROL  V-1.0";

code char manual1 [] = "1. = SET ORIGIN\n2. = FIND SAT";

code char manmenu [] = "1. = SET ORIGIN "
                       "2. = EXIT       ";



code char autmenu [] = "1. = FIND SAT-1 "
                       "2. = FIND SAT-2 "
                       "3. = FIND SAT-3 "
                       "4. = FIND SAT-4 "
                       "5. = FIND SAT-5 "
                       "6. = FIND SAT-6 "
                       "7. = FIND SAT-7 "
                       "8. = FIND SAT-8 "
                       "9. = FIND SAT-9 "
                       "                ";
```

```
//*************************************************
//************ main function **********************

void main(void)
  {
     char i;
     Delayms(500);
     sysinit();                    // system initial
     printf(logo);
     Delayms(1500);
     setorg();
     while(1)                      // loop forever
         {
           lcdposition(0);
           printf(logo);
           if('A' == getkey())
             {
               do{
                  lcdclr();
                  printf(manual1);
                  Delayms(250);
                  while(' ' == (i = getkey()));
                  switch(i)
                     {
                       case'1':setorg();break;
                       case'2':autmode();break;
                     }
                Delayms(100);
               }while(i != 'A');
             }
         }
  }

void setorg(void)
  {
    lcdclr();
    printf(" Origin Setting\n");
    while(elefg)
          {
            el_step_up();
          }
    p = 0;
    *p = ((*p & 0x0f) | 0x00);
    while(azifg)
          {
            az_step_1();
          }
    Delayms(500);
    p = 0;
    *p = ((*p & 0xf0) | 0x0f);
    lcdclr();
    printf("   Origin  OK\n");
```

```
    elecur = 90;
    azicur = 0;
    Delayms(2000);

}
void autmode(void)
{
    unsigned char index,m;
    index = 0;
    do{
       if('A'== (m = getkey()))
          {
            if((index+1) == 9)
               index = 0;
            else index++;
          }
       lcdposition(0);
       display(index,autmenu);
       switch(m)
          {
            case '1':findsat(1);break;
            case '2':findsat(2);break;
            case '3':findsat(3);break;
            case '4':findsat(4);break;
            case '5':findsat(5);break;
            case '6':findsat(6);break;
            case '7':findsat(7);break;
            case '8':findsat(8);break;
            case '9':findsat(9);break;
          }
     }while(m != 'B');
     Delayms(200);
  }


void findsat(char sat)
   {
     unsigned char stp1;
     unsigned char stp2;

     while((elebuf[sat-1] - elecur) > 0)
       {
         for(stp1=0;stp1 <(elebuf[sat-1] - elecur);stp1++)
             {
               for(stp2=0;stp2<200;stp2++) el_step_up();
               elecur++;
               lcdclr();
               printf("ELEVATION = %3d\n",elecur);
               printf("AZIMUTH   = %3d",azicur);
             }
       }
```

```
    while((elebuf[sat-1] - elecur) < 0)
            {
             for(stp1=0;stp1 <(elecur - elebuf[sat-1]);stp1++)
                 {
                   for(stp2=0;stp2 < 200;stp2++) el_step_dn();
                   elecur--;
                   lcdclr();
                   printf("ELEVATION = %3d\n",elecur);
                   printf("AZIMUTH   = %3d",azicur);
                 }
            }

    while((azibuf[sat-1] - azicur) > 0)
            {
               for(stp1=0;stp1 < (azibuf[sat-1] - azicur);stp1++)
                    {
                     az_step_r();
                     azicur++;
                     lcdclr();
                     printf("ELEVATION = %3d\n",elecur);

              printf("AZIMUTH   = %3d",azicur);
                    }
             }

    while((azibuf[sat-1] - azicur) < 0)
            {
             for(stp1=0;stp1 < (azicur - azibuf[sat-1]);stp1++)
                 {
                   az_step_l();
                   azicur--;
                   lcdclr();
                   printf("ELEVATION = %3d\n",elecur);
                   printf("AZIMUTH   = %3d",azicur);
                 }
            }

    lcdclr();
    printf(" Sat [%d] Standby\n",(int)sat);
    // printf(" Press # to Exit");
    while('B'!= getkey());
     p  = 0;
    *p = 0x0f;
    Delayms(300);
  }


void display(unsigned char ind,unsigned char *ptr)  // display menu
  {
   int cal;
   int temp;
```

```
  temp = cal = (ind * 16);
  for(cal;cal<(temp+32);cal++)
      putchar(*(ptr+cal));
}
void el_step_up(void)        // elevation drive 1 step up
   {
    p=0;
    if(!step1)
              {
                *p = ((*p & 0x0f) | 0x10);
                Delayms(4);
                *p = ((*p & 0x0f) | 0x20);
                Delayms(4);
                step1 = 1;
                step2=step3=step4=0;
              }
    else if(!step2)
             {
                *p = ((*p & 0x0f) | 0x20);
                Delayms(4);
                *p = ((*p & 0x0f) | 0x40);
                Delayms(4);
                step2 = 1;
                step3=step4=0;
             }
    else if(!step3)
             {
                *p = ((*p & 0x0f) | 0x40);
                Delayms(4);
                *p = ((*p & 0x0f) | 0x80);
                Delayms(4);
                step3 = 1;
                step4 = 0;
              }

     else if(!step4)
             {
                *p = ((*p & 0x0f) | 0x80);
                Delayms(4);
                *p = ((*p & 0x0f) | 0x10);
                Delayms(4);
                step1=step2=step3=step4=0;
              }

  }
void el_step_dn(void)            // elevation drive 1 step down
   {
    p=0;
    if(!step1)
             {
```

```
                *p = ((*p & 0x0f) | 0x80);
                Delayms(4);
                *p = ((*p & 0x0f) | 0x40);
                Delayms(4);
                step1 = 1;
                step2=step3=step4=0;
            }
    else if(!step2)
            {
               *p = ((*p & 0x0f) | 0x40);
               Delayms(4);
               *p = ((*p & 0x0f) | 0x20);
               Delayms(4);
               step2 = 1;
               step3=step4=0;
             }
    else if(!step3)
            {
               *p = ((*p & 0x0f) | 0x20);
               Delayms(4);
               *p = ((*p & 0x0f) | 0x10);
               Delayms(4);
               step3 = 1;
               step4 = 0;
            }

     else if(!step4)
            {
               *p = ((*p & 0x0f) | 0x10);
               Delayms(4);
               *p = ((*p & 0x0f) | 0x80);
               Delayms(4);
               step1=step2=step3=step4=0;
            }

 }

void az_step_r(void)                    // azimut drive 1 step right
   {
     p=0;
     if(!step1)
               {
                 *p = ((*p & 0xf0) | 0x0e);
                 Delayms(100);
                 *p = ((*p & 0xf0) | 0x0d);
                 Delayms(100);
                 step1 = 1;
                 step2=step3=step4=0;
               }
     else if(!step2)
            {
```

```
                *p = ((*p & 0xf0) | 0xd);
                Delayms(100);
                *p = ((*p & 0xf0) | 0xb);
                Delayms(100);
                step2 = 1;
                step3=step4=0;
              }
    else if(!step3)
              {
                *p = ((*p & 0xf0) | 0x0b);
                Delayms(100);
                *p = ((*p & 0xf0) | 0x07);
                Delayms(100);
                step3 = 1;
                step4 = 0;
              }

     else if(!step4)
              {
                *p = ((*p & 0xf0) | 0x07);
                Delayms(100);
                *p = ((*p & 0xf0) | 0x0e);
                Delayms(100);
                step1=step2=step3=step4=0;
              }

 }
void az_step_l(void)          // azimuth drive 1 step left
  {
    p=0;
    if(!step1)
              {
                *p = ((*p & 0xf0) | 0x07);
                Delayms(100);
                *p = ((*p & 0xf0) | 0x0b);
                Delayms(100);
                step1 = 1;
                step2=step3=step4=0;
              }
    else if(!step2)
              {
                *p = ((*p & 0xf0) | 0x0b);
                Delayms(100);
                *p = ((*p & 0xf0) | 0x0d);
                Delayms(100);
                step2 = 1;
                step3=step4=0;
              }
    else if(!step3)
              {
                *p = ((*p & 0xf0) | 0x0d);
```

```
                Delayms(100);
                *p = ((*p & 0xf0) | 0x0e);
                Delayms(100);
                step3 = 1;
                step4 = 0;
              }

     else if(!step4)
             {
                *p = ((*p & 0xf0) | 0x0e);
                Delayms(100);
                *p = ((*p & 0xf0) | 0x07);
                Delayms(100);
                step1=step2=step3=step4=0;
              }

}
void sysinit(void)                          // system initial
{
    Delayms(500);
    p = 0x03;
    *p = 0x88;
    p = 0;
    *p = 0xff;
    elecur = 90;azicur = 0;
    elebuf[0]=42;azibuf[0]=107;
    elebuf[1]=64;azibuf[1]=130;
    elebuf[2]=68;azibuf[2]=140;
    elebuf[3]=73;azibuf[3]=160;
    elebuf[4]=70;azibuf[4]=218;
    elebuf[5]=68;azibuf[5]=224;
    elebuf[6]=60;azibuf[6]=240;
    elebuf[7]=47;azibuf[7]=251;
    elebuf[8]=44;azibuf[8]=253;
    lcdini();
    gkeyol = 1;
}
```

```
NAME    SATA

PROG    SEGMENT   CODE
BITFG   SEGMENT   BIT
VAR1    SEGMENT   DATA


PORTA   EQU       0000H
PORTB   EQU       0001H
PORTC   EQU       0002H
CPORT   EQU       0003H
AUTOR1  EQU       0Fh             ;AUTO REPEAT DELAY (FIRST)
AUTOR2  EQU       2H              ;AUTO REPEAT DELAY (SECOND)


PUBLIC  LCDINI
PUBLIC  _DELAYMS
PUBLIC  _LCDWI
PUBLIC  _LCDWD
PUBLIC  _GETKEY
PUBLIC  GKEYOL              ; GET KEYBOARD ONE LOOP

        RSEG  VAR1
AUTODL: DS    2
KEYBUF: DS    1


        RSEG      BITFG
LCDFG:  DBIT      1
GKEYOL: DBIT      1
GKEYPF: DBIT      1


        RSEG      PROG

; ********** DMSEC SUB **********
; DELAY 1/1000 SECOND
; IN  = R6,R7  R6 = MSB, R7 = LSB
; REG = A,R5,R6,R7

_DELAYMS:
DMSEC:  MOV    R5,#230      ;1 MSEC LOOP
DMSEC1: NOP
        NOP
        DJNZ   R5,DMSEC1
        DJNZ   R7,DMSEC
        MOV    A,R6
        CJNE   A,#0,DMSEC2
        RET
DMSEC2: DEC    R6
        SJMP   DMSEC
        RET

; ****** LCDWI SUB ***************
; LCD WRITE COMMAND
_LCDWI: CLR    LCDFG
        LCALL  LCDWDI
```

```
        RET

;****** LCDWD SUB ****************
; LCD WRITE DATA
_LCDWD:  SETB   LCDFG
         LCALL  LCDWDI
         RET

;****** LCDWDI SUB **************
; LCD WRITE DATA OR COMMAND
; IN   =  A
; OUT  =  NONE
; REG  =  DPTR,A,B,R7

LCDWDI:  PUSH   DPL
         PUSH   DPH
         PUSH   B
         PUSH   ACC
         MOV    B,R7              ;SAVE DATA
         MOV    DPTR,#PORTB
         MOVX   A,@DPTR
         JB     LCDFG,$+7         ;LCDFG 1 = WRITE DATA
         CLR    ACC.5             ;CLR RS  TO WRITE COMMAND
         SJMP   $+4
         SETB   ACC.5             ;SET RS  TO WRITE DATA
         MOVX   @DPTR,A
         SETB   ACC.4
         MOVX   @DPTR,A

         MOVX   A,@DPTR
         ANL    A,#0F0H
         PUSH   ACC
         MOV    A,B
         ANL    A,#0F0H           ;SEND MSB 4 BIT
         SWAP   A
         MOV    R7,A
         POP    ACC
         ORL    A,R7
         MOVX   @DPTR,A
         LCALL  LCDDEL1
         MOVX   A,@DPTR
         CLR    ACC.4
         MOVX   @DPTR,A
         SETB   ACC.4
         MOVX   @DPTR,A
         LCALL  LCDDEL

         MOVX   A,@DPTR
         ANL    A,#0F0H
         PUSH   ACC
         MOV    A,B
         ANL    A,#0FH            ;SEND LSB 4 BIT
         MOV    R7,A
         POP    ACC
         ORL    A,R7
         MOVX   @DPTR,A
         LCALL  LCDDEL1
```

```
        MOVX    A,@DPTR
        CLR     ACC.4
        MOVX    @DPTR,A
        SETB    ACC.4
        MOVX    @DPTR,A
        LCALL   LCDDEL              ;DELAY FOR WAIT BUSY
        LCALL   LCDDEL1
        POP     ACC
        POP     B
        POP     DPH
        POP     DPL
        RET

;  ****** LCDINI SUB ***********
;; LCDINITIAL FOR LCD 4 BIT INTERFACE

LCDINI:  PUSH    DPL
        PUSH    DPH
        PUSH    ACC
        MOV     DPTR,#PORTB
        MOVX    A,@DPTR
        CLR     ACC.5               ;RS = PB.5  RS = 0 --> WRITE COMMAND
        MOVX    @DPTR,A
        SETB    ACC.4               ;EN = PB.4
        MOVX    @DPTR,A

        LCALL   LCDDEL1

        MOVX    A,@DPTR
        ANL     A,#0F0H
        ORL     A,#03H
        MOVX    @DPTR,A
        MOVX    A,@DPTR
        CLR     ACC.4               ;EN = PB.4
        MOVX    @DPTR,A
        SETB    ACC.4               ;EN = PB.4
        MOVX    @DPTR,A

        LCALL   LCDDEL1

        MOVX    A,@DPTR
        ANL     A,#0F0H
        ORL     A,#03H
        MOVX    @DPTR,A
        MOVX    A,@DPTR
        CLR     ACC.4               ;EN = PB.4
        MOVX    @DPTR,A
        SETB    ACC.4               ;EN = PB.4
        MOVX    @DPTR,A

        LCALL   LCDDEL1

        MOVX    A,@DPTR
        ANL     A,#0F0H
        ORL     A,#03H
        MOVX    @DPTR,A
        MOVX    A,@DPTR
```

```
        CLR     ACC.4                   ;EN = PB.4
        MOVX    @DPTR,A
        SETB    ACC.4                   ;EN = PB.4
        MOVX    @DPTR,A

        LCALL   LCDDEL1

        MOVX    A,@DPTR
        ANL     A,#0F0H
        ORL     A,#02H
        MOVX    @DPTR,A
        MOVX    A,@DPTR
        CLR     ACC.4                   ;EN = PB.4
        MOVX    @DPTR,A
        SETB    ACC.4                   ;EN = PB.4
        MOVX    @DPTR,A

        LCALL   LCDDEL1

        MOV     R7,#28H                 ;4 bit, 1 line, 4X7 matrix
        LCALL   _LCDWI
        MOV     R7,#0CH                 ;Display on, cursor off
        LCALL   _LCDWI
        MOV     R7,#01
        LCALL   _LCDWI
        LCALL   LCDDEL
        POP     ACC
        POP     DPH
        POP     DPL
        RET

LCDDEL: PUSH    02
        MOV     R2,#0
        DJNZ    R2,$
        POP     02
        RET

LCDDEL1: PUSH   02
        MOV     R2,#15
        DJNZ    R2,$
        POP     02
        RET

; ********** GKEY SUB **********
; SCAN AND WAIT FOR KEY AND KEY BEEP
; IN  =  GKEYPF                         (KEY PRESS)
;     =  GKEYOL                         (ONE LOOP SCAN)
; OUT =  A
; REG =  A,R0,R1,R2,R3,R4,R5,R6,R7,DPTR

_GETKEY:
GKEY:   MOV     R4,#4                   ;LOOP
        MOV     R5,#4                   ;RELEASE KEY COUNT
        MOV     R6,#1                   ;KEY-TABLE COUNT (SHIFT-BIT)

GKEY1:  MOV     A,R6
        CPL     A
```

```
        MOV     DPTR,#PORTC         ;OUT COLUMN
        MOVX    @DPTR,A

        MOV     R3,#0               ;DELAY
        DJNZ    R3,$

        MOV     DPTR,#PORTC         ;IN ROW
        MOVX    A,@DPTR
        ANL     A,#11110000B
        CJNE    A,#11110000B,GKEY3
        DJNZ    R5,GKEY2
        CLR     GKEYPF
        MOV     AUTODL,#AUTOR1
        SJMP    GKEY4

GKEY2:  MOV     A,R6                ;NEXT
        RL      A
        MOV     R6,A
        DJNZ    R4,GKEY1
        JNB     GKEYOL,GKEY         ;JUMP IF GKEYOL IS CLEAR

GKEY3:  ORL     A,R6
        JNB     GKEYOL,GKEY5        ;GKEYOL --> 1 == ONELOOP
        JNB     GKEYPF,GKEY5        ;IS NEW PRESS
        DEC     AUTODL              ;CHECK AUTO REPEAT
        MOV     A,AUTODL
        JNZ     GKEY4
        MOV     AUTODL,#AUTOR2      ;AUTO OK
        MOV     R7,KEYBUF
        RET

GKEY4:  MOV     A,#20H              ;NOTHING
        MOV     R7,A
        RET

GKEY5:  SETB    GKEYPF              ;*** NEW PRESS ***
        MOV     R2,A                ;LOOKUP TABLE
        MOV     R3,#11
        MOV     DPTR,#KEYTAB
GKEY51: CLR     A
        MOVC    A,@A+DPTR
        CLR     C
        SUBB    A,R2
        JZ      GKEY7
        INC     DPTR
        DJNZ    R3,GKEY51

GKEY7:  MOV     A,R3
        LCALL   HTOA
        MOV     A,R3
        MOV     KEYBUF,A
        MOV     R7,A
        JNB     GKEYOL,GKEY8
        RET                                 ;EXIT NEW PRESS

GKEY8:  MOV     R2,#10
GKEY9:  MOV     R3,#0
```

```
        DJNZ    R3,$
        DJNZ    R2,GKEY9
        RET

KEYTAB: DB      0E4H,0E1H,0D4H      ; * 0 9
        DB      0D2H,0D1H,0B4H      ; 8 7 6
        DB      0B2H,0B1H,074H      ; 5 4 3
        DB      072H,071H,0E2H      ; 2 1 #

;******* HTOA.SUB ************
;CONVERT HEXADECIMAL TO ASCII CODE
;IN  =  A
;OUT =  R2,R3
HTOA:   PUSH    ACC
        SWAP    A
        ACALL   HTOAS
        MOV     R2,A
        POP     ACC
        ACALL   HTOAS
        MOV     R3,A
        RET
HTOAS:  ANL     A,#0FH
        CJNE    A,#0AH,$+3
        JNC     HTOAS1
        ORL     A,#30H
        RET
HTOAS1: SUBB    A,#9
        ORL     A,#40H
        RET

        END
```

ภาคผนวก ก.

ตระกูลของไมโครคอนโทรลเลอร์

(Families of Microcontroller)

ตารางที่ ก1.1 แสดงไมโครคอนโครลเลอร์ตระกูล COP400 และ COP800 (Family of National Semiconductor)

Function / Model	Internal Memory		Parallel I / O	Timer / Counter	Serial I / O	A / D	PWM	DMA	Inter-rupt
	ROM	RAM							
COP413C	0.5 KB	32 x 4	16						
COP413CH	0.5 KB	32 x 4	16						
COP410C	0.5 KB	32 x 4	19						
COP411C	0.5 KB	32 x 4	16						
COP424C	1 KB	64 x 4	23	Y					1
COP425C	1 KB	64 x 4	20	Y					
COP426C	1 KB	64 x 4	16	Y					
COP444C	2 KB	128 x 4	23	Y					1
COP445C	2 KB	128 x 4	20	Y					
COP820C	1 KB	64 B	3 x 8	1 x 16					3
COP821C	1 KB	64 B	5 x 4	1 x 16					3
COP822C	1 KB	64 B	2 x 8	1 x 16					3
COP8720C	1 KB	64 B	3 x 8	1 x 16					3
COP8721C	1 KB	64 B	5 x 4	1 x 16					3
COP8722C	1 KB	64 B	2 x 8	1 x 16					3

ตารางที่ ก1.1 (ต่อ) แสดงไมโครคอนโครลเลอร์ตระกูล COP800, HPC16040 (Family of National Semiconductor)

Function / Model	Internal Memory		Parallel I / O	Timer / Counter	Serial I / O	A / D	PWM	DMA	Inter-rupt
	ROM	RAM							
COP840C	2 KB	128 B	3 x 8	1 x 16					3
COP841C	2 KB	128 B	5 x 4	1 x 16					3
COP888CF	4 KB	128 B	36 / 40	2 x 16		8	2		10
COP888CG	4 KB	192 B	36 / 40	2 x 16	1		2		12
COP888CK	4 KB	192 B	36 / 40	2 x 16	1	8	2		13
COP8788CKMH	4 KB	192 B	36 / 40	2 x 16	1	8	2		13
COP8073	2.5 KB	64 B	36 / 40		1				
HPC16003		256 B	52	8 x 16			Y		8
HPC16083	8 KB	256 B	52	8 x 16			Y		8
HPC16084	8 KB	253 B	52	8 x 16		8 x 8	Y		8
HPC16400		256 B	52	4 x 16	12 x HDLC		Y	4	8
HPC16900		256 B	52						
HPC16083MH	8KBEPROM	256 B	52	8 x 16			Y		8
HPC16084MH	8KBEPROM	256 B	52	8 x 16		8 x 8	Y		8

ตารางที่ ก1.2 แสดงไมโครคอนโครลเลอร์ตระกูล Zilog (Single Chip Microcontroller)

Serial	Model	Package		Memory		I / O				Inter-rupt
		DIP	PGA	ROM	RAM	I/O (pins)	Serial I/O	Timer / Counter	DMA	
Z8	Z860x	Z8601	Z8602	2 KB	128 B	32	1	2 x 8	/	6
	Z861x	Z8611	Z8612	4 KB	128 B	32	1	2 x 8	/	6
	Z867x	Z8671		Burn in BASIC	128 B	32	1	2 x 8	/	6
Z-UPC	Z809x	Z8090		2 KB	256 B	32	By	2 x 8	/	
	Z859x	Z8590		2 KB	256 B	32	Software	2 x 28	/	
Super 8	Super 8			8 - 16 KB	256 B	32	1	2 x 16	Yes	7

ตารางที่ ก1.3 แสดงไมโครคอนโครลเลอร์ตระกูล NEC (Single Chip Microcontroller)

Family		Function	Internal Memory ROM	Internal Memory RAM	I / O (pins)	Timer / Counter	A / D	Inter-rupt	Fully duplex Serial	Watch dog
8 Bit	Pre-limitary	μPD7800		128 B	32	1 x 12		5	Yes	
		μPD7801	4 KB	128 B	48	1 x 12		5	Yes	
		μPD7802	6 KB	64 B	48	1 x 12		5	Yes	
		μPD78C05		128 B	46	1 x 12		3	Yes	
		μPD78C06	4 KB	128 B	46	1 x 12		3	Yes	
	En-hanced	μPD7807		256	36	2 x 8		11	Yes	Y
		μPD7809	8 KB	256	48	1 x 16		11	Yes	Y
		μPD7810		256	32	1 x 16	8 x 8	11	Yes	
		μPD7811	4 KB	256	44	1 x 16	8 x 8	11	Yes	
16 Bit		783 x x	8 KB	256	48	2 x 16	4 x 8	15	Yes	Y

ตารางที่ ก1.4 แสดงไมโครคอนโครลเลอร์ตระกูล TMS7000

Company \ Function	Model	Internal Memory: ROM	Internal Memory: RAM	Timer / Counter	Parallel I / O	Serial I / O	Interrupt
TI	7000		128 B	1 x 16	4 x 8		4
	7001		128 B	2 x 16	4 x 8	1	2
	7020	2 KB	128 B	1 x 16	4 x 8		4
	7040	4 KB	128 B	1 x 16	4 x 8		4
	7041	4 KB	128 B	2 x 16	4 x 8	1	6
	70P161	16 KB EPROM	128 B	2 x 16	4 x 8	1	6
	70C00			1 x 16	4 x 8		4
	70C20	2 KB	128 B	1 x 16	4 x 8		4
	70C40	4 KB	256 B	2 x 16	4 x 8		4
	70C02		128 B	1 x 16	4 x 8	1	6
	07C42	4 KB	128 B	2 x 16	4 x 8	1	6
	70C82	8 KB EPROM	256 B	2 x 16	4 x 8	1	6
Sceg	72720	2 KB E^2PROM	256 B	2 x 16	4 x 8		4
GI	7060	6 KB	256 B	2 x 16	4 x 8		4
	7080	8 KB	256 B	2 x 16	4 x 8		4
	70100	10 KB	256 B	2 x 16	4 x 8		4
	70122	12 KB	256 B	2 x 16	4 x 8		4

ตารางที่ ก1.5 แสดงไมโครคอนโครลเลอร์ตระกูล 6500 / 1

Company	Model	Internal Memory		I / O				Interrupt
		ROM	RAM	Timer / Counter	Parallel I / O	Serial I / O	A / D	
Rockwell	6500 / 1	2 KB	64 B	1 x 16	4 x 8			4
	6500 / 11	3 KB	192 B	2 x 16	4 x 8	1		8
	6500 / 12	3 KB	192 B	2 x 16	7 x 8	1		8
	6500 / 13	256 B	192 B	2 x 16	4 x 8	1		8
	R65F11	3 KB	192 B	2 x 16	4 x 8	1		9
	R65F12	3 KB	192 B	2 x 16	7 x 8	1		9
	6500 / 41	1.5 KB	64 B	1 x 16	2 x 8 1 x 7			6
	6500 / 42	1.5 KB	64 B	1 x 16	5 x 8 1 x 7			6
	6500 / 43	256 B	64 B	1 x 16	2 x 8 1 x 7			7
Mitsu-bishi	50700	8 KB	512 B	2 x 16	7 x 8	1	8 bit	8
	37740	16 KB	512 B	8 x 16	7 x 8	2	7 x 8 bit	8
WDC	65C124 65C134 65C254					token-passing		

ตารางที่ ก1.6 แสดงไมโครคอนโครลเลอร์ตระกูล 3870 (F8) (Family of Fairchild and Mostek)

Function / Model	Internal Memory			Timer / Counter	Parallel I / O	Serial I / O	Interrupt
	ROM	EPROM	RAM				
3870	1K / 4K		64 B	1 x 8	4 x 8		1
38E70		2 K	64 B	1 x 8	4 x 8		1
38P70			64 B + 64 B	1 x 8	4 x 8		1
3870 / 42	4 K		64 B + 64 B	1 x 8	4 x 8		1
3871				1 x 8	2 x 8		
3873	1K / 2K		64 B	1 x 8	3 x 8 1 x 5	Half Duplex	1
3875 / 42	2K / 4K		64 B + 64 B	1 x 8	4 x 8		1
2870	1 K		64 B + 64 B	1 x 8	2 x 8 1 x 4		1

ตารางที่ ก1.7 แสดงไมโครคอนโครลเลอร์ตระกูล PiC1600 (Family of General Instrument)

Model	Internal Memory		I / O (pins)	Package (pins)
	ROM	RAM		
1650	512 x 12	32 B	8 x 4	40
1655	512 x 12	32 B	20	28
1670	1024 x 12	39 B	8 x 4	40
1645	256 x 12	16 B	3 x 4	18
16C52	256 x 12	32 B	3 x 4	18
16C54	512 x 12	32 B	3 x 4	18
16C53	256 x 12	32 B	20	28
16C55	512 x 12	32 B	20	28

ภาคผนวก ข.

ชุดคำสั่งของ 8051

MCS®-51 INSTRUCTION SET

Table 10. 8051 Instruction Set Summary

Interrupt Response Time: Refer to Hardware Description Chapter.

Instructions that Affect Flag Settings(1)

Instruction	Flag			Instruction	Flag		
	C	OV	AC		C	OV	AC
ADD	X	X	X	CLR C	O		
ADDC	X	X	X	CPL C	X		
SUBB	X	X	X	ANL C,bit	X		
MUL	O	X		ANL C,/bit	X		
DIV	O	X		ORL C,bit	X		
DA	X			ORL C,bit	X		
RRC	X			MOV C,bit	X		
RLC	X			CJNE	X		
SETB C	1						

(1)Note that operations on SFR byte address 208 or bit addresses 209-215 (i.e., the PSW or bits in the PSW) will also affect flag settings.

Note on instruction set and addressing modes:

Rn — Register R7–R0 of the currently selected Register Bank.

direct — 8-bit internal data location's address. This could be an Internal Data RAM location (0–127) or a SFR [i.e., I/O port, control register, status register, etc. (128–255)].

@Ri — 8-bit internal data RAM location (0–255) addressed indirectly through register R1 or R0.

#data — 8-bit constant included in instruction.

#data 16 — 16-bit constant included in instruction.

addr 16 — 16-bit destination address. Used by LCALL & LJMP. A branch can be anywhere within the 64K-byte Program Memory address space.

addr 11 — 11-bit destination address. Used by ACALL & AJMP. The branch will be within the same 2K-byte page of program memory as the first byte of the following instruction.

rel — Signed (two's complement) 8-bit offset byte. Used by SJMP and all conditional jumps. Range is −128 to +127 bytes relative to first byte of the following instruction.

bit — Direct Addressed bit in Internal Data RAM or Special Function Register.

Mnemonic		Description	Byte	Oscillator Period
ARITHMETIC OPERATIONS				
ADD	A,Rn	Add register to Accumulator	1	12
ADD	A,direct	Add direct byte to Accumulator	2	12
ADD	A,@Ri	Add indirect RAM to Accumulator	1	12
ADD	A,#data	Add immediate data to Accumulator	2	12
ADDC	A,Rn	Add register to Accumulator with Carry	1	12
ADDC	A,direct	Add direct byte to Accumulator with Carry	2	12
ADDC	A,@Ri	Add indirect RAM to Accumulator with Carry	1	12
ADDC	A,#data	Add immediate data to Acc with Carry	2	12
SUBB	A,Rn	Subtract Register from Acc with borrow	1	12
SUBB	A,direct	Subtract direct byte from Acc with borrow	2	12
SUBB	A,@Ri	Subtract indirect RAM from ACC with borrow	1	12
SUBB	A,#data	Subtract immediate data from Acc with borrow	2	12
INC	A	Increment Accumulator	1	12
INC	Rn	Increment register	1	12
INC	direct	Increment direct byte	2	12
INC	@Ri	Increment direct RAM	1	12
DEC	A	Decrement Accumulator	1	12
DEC	Rn	Decrement Register	1	12
DEC	direct	Decrement direct byte	2	12
DEC	@Ri	Decrement indirect RAM	1	12

Courtesy of INTEL Corp.

Table 10. 8051 Instruction Set Summary (Continued)

Mnemonic		Description	Byte	Oscillator Period
ARITHMETIC OPERATIONS (Continued)				
INC	DPTR	Increment Data Pointer	1	24
MUL	AB	Multiply A & B	1	48
DIV	AB	Divide A by B	1	48
DA	A	Decimal Adjust Accumulator	1	12
LOGICAL OPERATIONS				
ANL	A,Rn	AND Register to Accumulator	1	12
ANL	A,direct	AND direct byte to Accumulator	2	12
ANL	A,@Ri	AND indirect RAM to Accumulator	1	12
ANL	A,#data	AND immediate data to Accumulator	2	12
ANL	direct,A	AND Accumulator to direct byte	2	12
ANL	direct,#data	AND immediate data to direct byte	3	24
ORL	A,Rn	OR register to Accumulator	1	12
ORL	A,direct	OR direct byte to Accumulator	2	12
ORL	A,@Ri	OR indirect RAM to Accumulator	1	12
ORL	A,#data	OR immediate data to Accumulator	2	12
ORL	direct,A	OR Accumulator to direct byte	2	12
ORL	direct,#data	OR immediate data to direct byte	3	24
XRL	A,Rn	Exclusive-OR register to Accumulator	1	12
XRL	A,direct	Exclusive-OR direct byte to Accumulator	2	12
XRL	A,@Ri	Exclusive-OR indirect RAM to Accumulator	1	12
XRL	A,#data	Exclusive-OR immediate data to Accumulator	2	12
XRL	direct,A	Exclusive-OR Accumulator to direct byte	2	12
XRL	direct,#data	Exclusive-OR immediate data to direct byte	3	24
CLR	A	Clear Accumulator	1	12
CPL	A	Complement Accumulator	1	12
LOGICAL OPERATIONS (Continued)				
RL	A	Rotate Accumulator Left	1	12
RLC	A	Rotate Accumulator Left through the Carry	1	12
RR	A	Rotate Accumulator Right	1	12
RRC	A	Rotate Accumulator Right through the Carry	1	12
SWAP	A	Swap nibbles within the Accumulator	1	12
DATA TRANSFER				
MOV	A,Rn	Move register to Accumulator	1	12
MOV	A,direct	Move direct byte to Accumulator	2	12
MOV	A,@Ri	Move indirect RAM to Accumulator	1	12
MOV	A,#data	Move immediate data to Accumulator	2	12
MOV	Rn,A	Move Accumulator to register	1	12
MOV	Rn,direct	Move direct byte to register	2	24
MOV	Rn,#data	Move immediate data to register	2	12
MOV	direct,A	Move Accumulator to direct byte	2	12
MOV	direct,Rn	Move register to direct byte	2	24
MOV	direct,direct	Move direct byte to direct	3	24
MOV	direct,@Ri	Move indirect RAM to direct byte	2	24
MOV	direct,#data	Move immediate data to direct byte	3	24
MOV	@Ri,A	Move Accumulator to indirect RAM	1	12

Courtesy of INTEL Corp

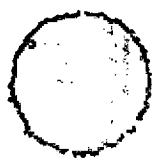
Table 10. 8051 Instruction Set Summary (Continued)

Mnemonic		Description	Byte	Oscillator Period
DATA TRANSFER (Continued)				
MOV	@Ri,direct	Move direct byte to indirect RAM	2	24
MOV	@Ri,#data	Move immediate data to indirect RAM	2	12
MOV	DPTR,#data16	Load Data Pointer with a 16-bit constant	3	24
MOVC	A,@A+DPTR	Move Code byte relative to DPTR to Acc	1	24
MOVC	A,@A+PC	Move Code byte relative to PC to Acc	1	24
MOVX	A,@Ri	Move External RAM (8-bit addr) to Acc	1	24
MOVX	A,@DPTR	Move External RAM (16-bit addr) to Acc	1	24
MOVX	@Ri,A	Move Acc to External RAM (8-bit addr)	1	24
MOVX	@DPTR,A	Move Acc to External RAM (16-bit addr)	1	24
PUSH	direct	Push direct byte onto stack	2	24
POP	direct	Pop direct byte from stack	2	24
XCH	A,Rn	Exchange register with Accumulator	1	12
XCH	A,direct	Exchange direct byte with Accumulator	2	12
XCH	A,@Ri	Exchange indirect RAM with Accumulator	1	12
XCHD	A,@Ri	Exchange low-order Digit indirect RAM with Acc	1	12
BOOLEAN VARIABLE MANIPULATION				
CLR	C	Clear Carry	1	12
CLR	bit	Clear direct bit	2	12
SETB	C	Set Carry	1	12
SETB	bit	Set direct bit	2	12
CPL	C	Complement Carry	1	12
CPL	bit	Complement direct bit	2	12
ANL	C,bit	AND direct bit to CARRY	2	24
ANL	C,/bit	AND complement of direct bit to Carry	2	24
ORL	C,bit	OR direct bit to Carry	2	24
ORL	C,/bit	OR complement of direct bit to Carry	2	24
MOV	C,bit	Move direct bit to Carry	2	12
MOV	bit,C	Move Carry to direct bit	2	24
JC	rel	Jump if Carry is set	2	24
JNC	rel	Jump if Carry not set	2	24
JB	bit,rel	Jump if direct Bit is set	3	24
JNB	bit,rel	Jump if direct Bit is Not set	3	24
JBC	bit,rel	Jump if direct Bit is set & clear bit	3	24
PROGRAM BRANCHING				
ACALL	addr11	Absolute Subroutine Call	2	24
LCALL	addr16	Long Subroutine Call	3	24
RET		Return from Subroutine	1	24
RETI		Return from interrupt	1	24
AJMP	addr11	Absolute Jump	2	24
LJMP	addr16	Long Jump	3	24
SJMP	rel	Short Jump (relative addr)	2	24

Courtesy of INTEL Corp.

Table 10. 8051 Instruction Set Summary (Continued)

Mnemonic		Description	Byte	Oscillator Period
PROGRAM BRANCHING (Continued)				
JMP	@A+DPTR	Jump indirect relative to the DPTR	1	24
JZ	rel	Jump if Accumulator is Zero	2	24
JNZ	rel	Jump if Accumulator is Not Zero	2	24
CJNE	A,direct,rel	Compare direct byte to Acc and Jump if Not Equal	3	24
CJNE	A,#data,rel	Compare immediate to Acc and Jump if Not Equal	3	24
CJNE	Rn,#data,rel	Compare immediate to register and Jump if Not Equal	3	24
CJNE	@Ri,#data,rel	Compare immediate to indirect and Jump if Not Equal	3	24
DJNZ	Rn,rel	Decrement register and Jump if Not Zero	2	24
DJNZ	direct,rel	Decrement direct byte and Jump if Not Zero	3	24
NOP		No Operation	1	12



Courtesy of INTEL Corp

ภาคผนวก ค.

ข้อมูลเกี่ยวกับ 8051 โดยสังเขป

PRELIMINARY

MCS®-51
8-BIT CONTROL-ORIENTED MICROCOMPUTERS
8031/8051
8031AH/8051AH
8032AH/8052AH
8751H/8751H-8

- High Performance HMOS Process
- Internal Timers/Event Counters
- 2-Level Interrupt Priority Structure
- 32 I/O Lines (Four 8-Bit Ports)
- 64K Program Memory Space
- Security Feature Protects EPROM Parts Against Software Piracy
- Boolean Processor
- Bit-Addressable RAM
- Programmable Full Duplex Serial Channel
- 111 Instructions (64 Single-Cycle)
- 64K Data Memory Space

The MCS®-51 products are optimized for control applications. Byte-processing and numerical operations on small data structures are facilitated by a variety of fast addressing modes for accessing the internal RAM. The instruction set provides a convenient menu of 8-bit arithmetic instructions, including multiply and divide instructions. Extensive on-chip support is provided for one-bit variables as a separate data type, allowing direct bit manipulation and testing in control and logic systems that require Boolean processing.

The 8051 is the original member of the MCS-51 family. The 8051AH is identical to the 8051, but it is fabricated with HMOS II technology.

The 8751H is an EPROM version of the 8051AH; that is, the on-chip Program Memory can be electrically programmed, and can be erased by exposure to ultraviolet light. It is fully compatible with its predecessor, the 8751-8, but incorporates two new features: a Program Memory Security bit that can be used to protect the EPROM against unauthorized read-out, and a programmable baud rate modification bit (SMOD). The 8751H-8 is identical to the 8751H but only operates up to 8 MHz.

The 8052AH is an enhanced version of the 8051AH. It is backwards compatible with the 8051AH and is fabricated with HMOS II technology. The 8052AH enhancements are listed in the table below. Also refer to this table for the ROM, ROMless, and EPROM versions of each product.

Device	Internal Memory		Timers/ Event Counters	Interrupts
	Program	Data		
8052AH	8K x 8 ROM	256 x 8 RAM	3 x 16-Bit	6
8051AH	4K x 8 ROM	128 x 8 RAM	2 x 16-Bit	5
8051	4K x 8 ROM	128 x 8 RAM	2 x 16-Bit	5
8032AH	none	256 x 8 RAM	3 x 16-Bit	6
8031AH	none	128 x 8 RAM	2 x 16-Bit	5
8031	none	128 x 8 RAM	2 x 16-Bit	5
8751H	4K x 8 EPROM	128 x 8 RAM	2 x 16-Bit	5
8751H-8	4K x 8 EPROM	128 x 8 RAM	2 x 16-Bit	5

October 1988
Order Number: 270048-004

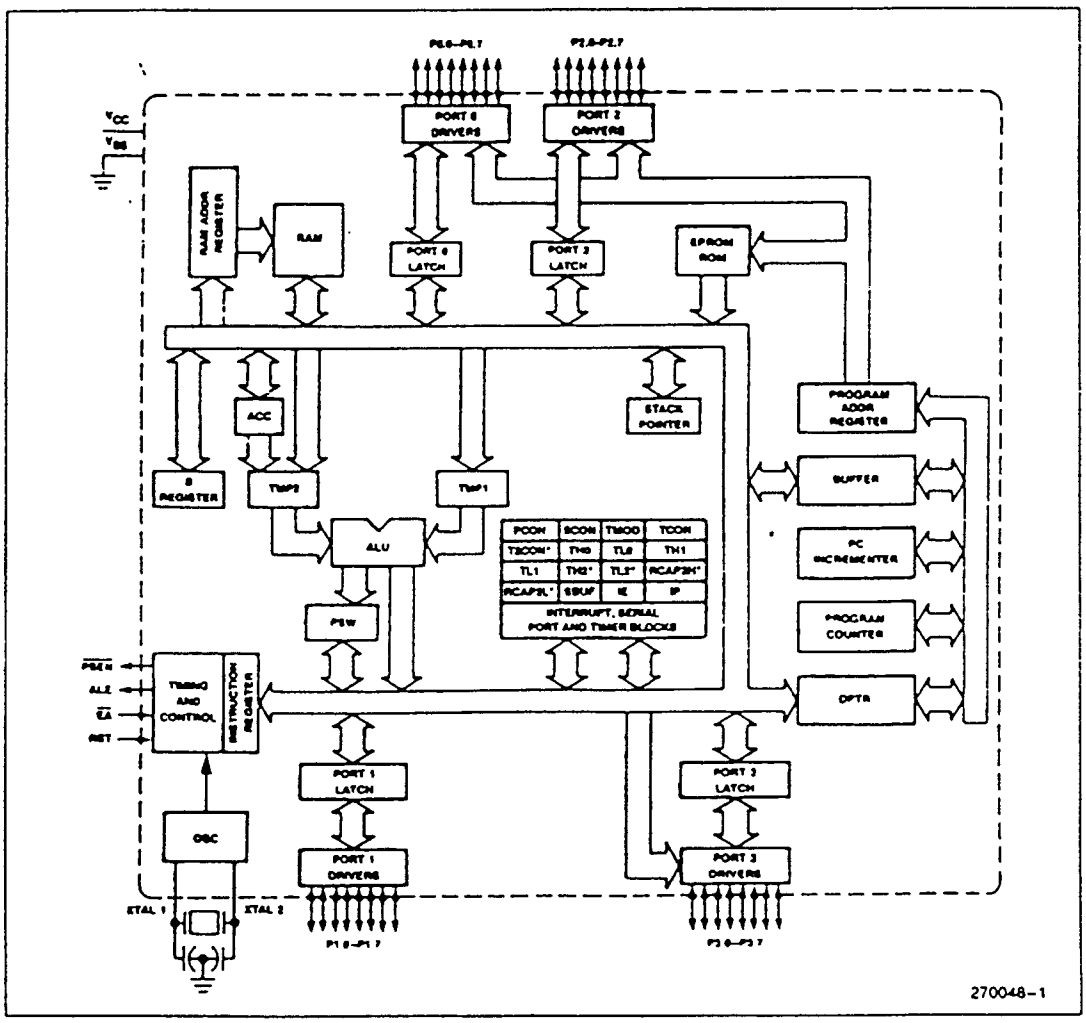


Figure 1. MCS®-51 Block Diagram

PACKAGES

Part	Prefix	Package Type
8051AH/ 8031AH	P	40-Pin Plastic DIP
	D	40-Pin CERDIP
	N	44-Pin PLCC
8052AH/ 8032AH	P	40-Pin Plastic DIP
	D	40-Pin CERDIP
	N	44-Pin PLCC
8751H/ 8751H-8	D	40-Pin CERDIP
	R	44-Pin LCC

PIN DESCRIPTIONS

V_{CC}: Supply voltage.

V_{SS}: Circuit ground.

Port 0: Port 0 is an 8-bit open drain bidirectional I/O port. As an output port each pin can sink 8 LS TTL inputs.

Port 0 pins that have 1s written to them float, and in that state can be used as high-impedance inputs.

Port 0 is also the multiplexed low-order address and data bus during accesses to external Program and Data Memory. In this application it uses strong internal pullups when emitting 1s and can source and sink 8 LS TTL inputs.

Port 0 also receives the code bytes during programming of the EPROM parts, and outputs the code bytes during program verification of the ROM and EPROM parts. External pullups are required during program verification.

Courtesy of INTEL Corp.

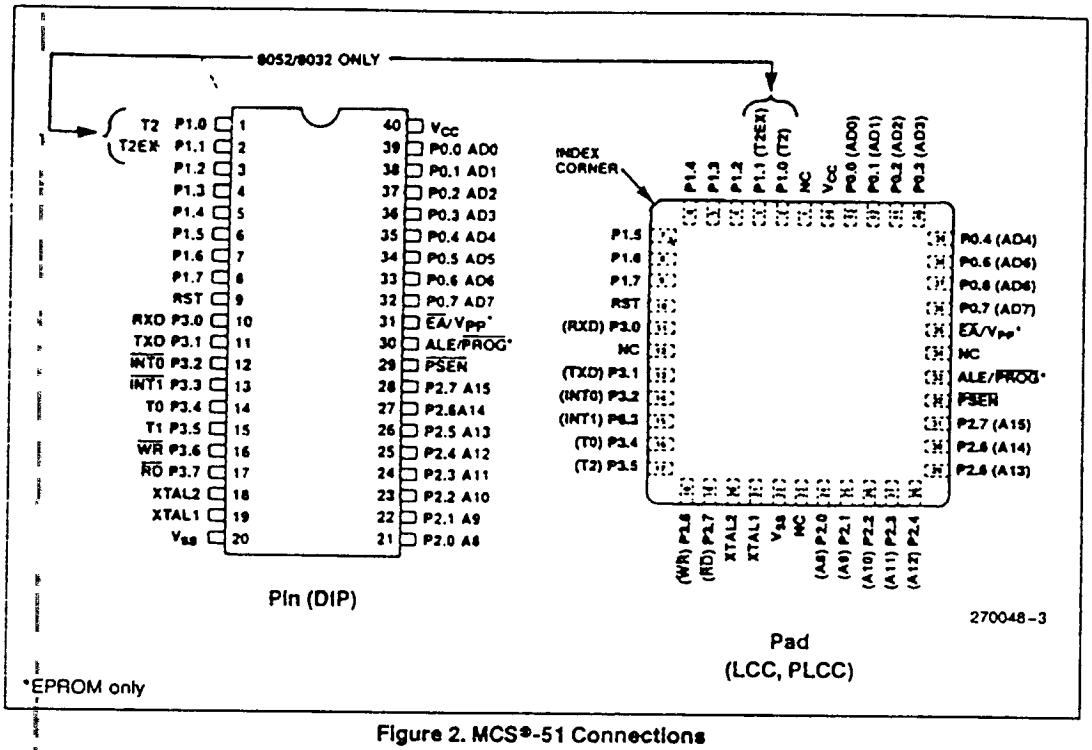


Figure 2. MCS®-51 Connections

Port 1: Port 1 is an 8-bit bidirectional I/O port with internal pullups. The Port 1 output buffers can sink/source 4 LS TTL inputs. Port 1 pins that have 1s written to them are pulled high by the internal pullups, and in that state can be used as inputs. As inputs, Port 1 pins that are externally being pulled low will source current (I_{IL} on the data sheet) because of the internal pullups.

Port 1 also receives the low-order address bytes during programming of the EPROM parts and during program verification of the ROM and EPROM parts.

In the 8032AH and 8052AH, Port 1 pins P1.0 and P1.1 also serve the T2 and T2EX functions, respectively.

Port 2: Port 2 is an 8-bit bidirectional I/O port with internal pullups. The Port 2 output buffers can sink/source 4 LS TTL inputs. Port 2 pins that have 1s written to them are pulled high by the internal pullups, and in that state can be used as inputs. As inputs, Port 2 pins that are externally being pulled low will source current (I_{IL} on the data sheet) because of the internal pullups.

Port 2 emits the high-order address byte during fetches from external Program Memory and during accesses to external Data Memory that use 16-bit addresses (MOVX @DPTR). In this application it uses strong internal pullups when emitting 1s. During accesses to external Data Memory that use 8-bit addresses (MOVX @Ri), Port 2 emits the contents of the P2 Special Function Register.

Port 2 also receives the high-order address bits during programming of the EPROM parts and during program verification of the ROM and EPROM parts.

Port 3: Port 3 is an 8-bit bidirectional I/O port with internal pullups. The Port 3 output buffers can sink/source 4 LS TTL inputs. Port 3 pins that have 1s written to them are pulled high by the internal pullups, and in that state can be used as inputs. As inputs, Port 3 pins that are externally being pulled low will source current (I_{IL} on the data sheet) because of the pullups.

Port 3 also serves the functions of various special features of the MCS-51 Family, as listed below:

Port Pin	Alternative Function
P3.0	RXD (serial input port)
P3.1	TXD (serial output port)
P3.2	$\overline{INT0}$ (external interrupt 0)
P3.3	$\overline{INT1}$ (external interrupt 1)
P3.4	T0 (Timer 0 external input)
P3.5	T1 (Timer 1 external input)
P3.6	$\overline{WR}$ (external data memory write strobe)
P3.7	$\overline{RD}$ (external data memory read strobe)

RST: Reset input. A high on this pin for two machine cycles while the oscillator is running resets the device.

ALE/$\overline{PROG}$: Address Latch Enable output pulse for latching the low byte of the address during accesses to external memory. This pin is also the program pulse input ($\overline{PROG}$) during programming of the EPROM parts.

In normal operation ALE is emitted at a constant rate of 1/6 the oscillator frequency, and may be used for external timing or clocking purposes. Note, however, that one ALE pulse is skipped during each access to external Data Memory.

$\overline{PSEN}$: Program Store Enable is the read strobe to external Program Memory.

When the device is executing code from external Program Memory, $\overline{PSEN}$ is activated twice each machine cycle, except that two $\overline{PSEN}$ activations are skipped during each access to external Data Memory.

$\overline{EA}$/V_{PP}: External Access enable $\overline{EA}$ must be strapped to V_{SS} in order to enable any MCS-51 device to fetch code from external Program memory locations starting at 0000H up to FFFFH. $\overline{EA}$ must be strapped to V_{CC} for internal program execution.

Note, however, that if the Security Bit in the EPROM devices is programmed, the device will not fetch code from any location in external Program Memory.

This pin also receives the 21V programming supply voltage (VPP) during programming of the EPROM parts.

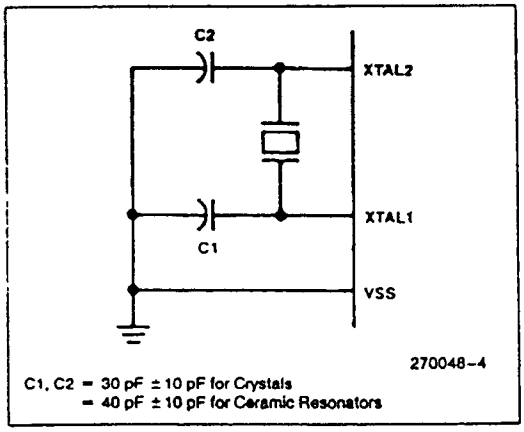


Figure 3. Oscillator Connections

XTAL1: Input to the inverting oscillator amplifier.

XTAL2: Output from the inverting oscillator amplifier.

OSCILLATOR CHARACTERISTICS

XTAL1 and XTAL2 are the input and output, respectively, of an inverting amplifier which can be configured for use as an on-chip oscillator, as shown in Figure 3. Either a quartz crystal or ceramic resonator may be used. More detailed information concerning the use of the on-chip oscillator is available in Application Note AP-155, "Oscillators for Microcontrollers."

To drive the device from an external clock source, XTAL1 should be grounded, while XTAL2 is driven, as shown in Figure 4. There are no requirements on the duty cycle of the external clock signal, since the input to the internal clocking circuitry is through a divide-by-two flip-flop, but minimum and maximum high and low times specified on the Data Sheet must be observed.

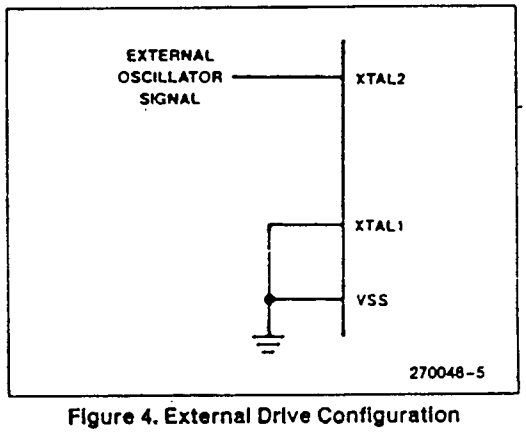


Figure 4. External Drive Configuration

DESIGN CONSIDERATIONS

If an 8751BH or 8752BH may replace an 8751H in a future design, the user should carefully compare both data sheets for DC or AC Characteristic differences. Note that the V_{IH} and I_{IH} specifications for the $\overline{EA}$ pin differ significantly between the devices.

Exposure to light when the EPROM device is in operation may cause logic errors. For this reason, it is suggested that an opaque label be placed over the window when the die is exposed to ambient light.

ABSOLUTE MAXIMUM RATINGS*

Ambient Temperature Under Bias .....0°C to 70°C
Storage Temperature .....−65°C to +150°C
Voltage on $\overline{EA}$/V_{PP} Pin to V_{SS} ....−0.5V to +21.5V
Voltage on Any Other Pin to V_{SS} ....−0.5V to +7V
Power Dissipation 1.5W

Notice: Stresses above those listed under "Absolute Maximum Ratings" may cause permanent damage to the device. This is a stress rating only and functional operation of the device at these or any other conditions above those indicated in the operational sections of this specification is not implied. Exposure to absolute maximum rating conditions for extended periods may affect device reliability.

D.C. CHARACTERISTICS T_A = 0°C to 70°C; V_{CC} = 5V ±10%; V_{SS} = 0V

Symbol	Parameter		Min	Max	Units	Test Conditions
V_{IL}	Input Low Voltage (Except $\overline{EA}$ Pin of 8751H & 8751H-8)		−0.5	0.8	V	
V_{IL1}	Input Low Voltage to $\overline{EA}$ Pin of 8751H & 8751H-8		0	0.7	V	
V_{IH}	Input High Voltage (Except XTAL2, RST)		2.0	V_{CC} + 0.5	V	
V_{IH1}	Input High Voltage to XTAL2, RST		2.5	V_{CC} + 0.5	V	XTAL1 = V_{SS}
V_{OL}	Output Low Voltage (Ports 1, 2, 3)*			0.45	V	I_{OL} = 1.6 mA
V_{OL1}	Output Low Voltage (Port 0, ALE, $\overline{PSEN}$)*					
		8751H, 8751H-8		0.60 0.45	V V	I_{OL} = 3.2 mA I_{OL} = 2.4 mA
		All Others		0.45	V	I_{OL} = 3.2 mA
V_{OH}	Output High Voltage (Ports 1, 2, 3, ALE, $\overline{PSEN}$)		2.4		V	I_{OH} = −80 μA
V_{OH1}	Output High Voltage (Port 0 in External Bus Mode)		2.4		V	I_{OH} = −400 μA
I_{IL}	Logical 0 Input Current (Ports 1, 2, 3, RST) 8032AH, 8052AH All Others			−800 −500	μA μA	V_{IN} = 0.45V V_{IN} = 0.45V
I_{IL1}	Logical 0 Input Current to $\overline{EA}$ Pin of 8751H & 8751H-8 Only			−15	mA	V_{IN} = 0.45V
I_{IL2}	Logical 0 Input Current (XTAL2)			−3.2	mA	V_{IN} = 0.45V
I_{LI}	Input Leakage Current (Port 0) 8751H & 8751H-8 All Others			±100 ±10	μA μA	0.45 ≤ V_{IN} ≤ V_{CC} 0.45 ≤ V_{IN} ≤ V_{CC}
I_{IH}	Logical 1 Input Current to $\overline{EA}$ Pin of 8751H & 8751H-8			500	μA	V_{IN} = 2.4V
I_{IH1}	Input Current to RST to Activate Reset			500	μA	V_{IN} < (V_{CC} − 1.5V)
I_{CC}	Power Supply Current: 8031/8051 8031AH/8051AH 8032AH/8052AH 8751H/8751H-8			 160 125 175 250	 mA mA mA mA	 All Outputs Disconnected; $\overline{EA}$ = V_{CC}
C_{IO}	Pin Capacitance			10	pF	Test freq = 1 MHz

NOTE:
Capacitive loading on Ports 0 and 2 may cause spurious noise pulses to be superimposed on the V_{OL}s of ALE and Ports 1 and 3. The noise is due to external bus capacitance discharging into the Port 0 and Port 2 pins when these pins make 1-to-0 transitions during bus operations. In the worst cases (capacitive loading > 100 pF), the noise pulse on the ALE line may exceed 0.8V. In such cases it may be desirable to qualify ALE with a Schmitt Trigger, or use an address latch with a Schmitt Trigger STROBE input.

Courtesy of INTEL Corp.

A.C. CHARACTERISTICS

T_A = 0°C to +70°C; V_{CC} = 5V ±10%; V_{SS} = 0V;
Load Capacitance for Port 0, ALE, and $\overline{PSEN}$ = 100 pF;
Load Capacitance for All Other Outputs = 80 pF

Symbol	Parameter	12 MHz Oscillator		Variable Oscillator		Units
		Min	Max	Min	Max	
1/TCLCL	Oscillator Frequency			3.5	12.0	MHz
TLHLL	ALE Pulse Width	127		2TCLCL−40		ns
TAVLL	Address Valid to ALE Low	43		TCLCL−40		ns
TLLAX	Address Hold after ALE Low	48		TCLCL−35		ns
TLLIV	ALE Low to Valid Instr In 8751H All Others		 183 233		 4TCLCL−150 4TCLCL−100	 ns ns
TLLPL	ALE Low to $\overline{PSEN}$ Low	58		TCLCL−25		ns
TPLPH	$\overline{PSEN}$ Pulse Width 8751H All Others	 190 215		 3TCLCL−60 3TCLCL−35		 ns ns
TPLIV	$\overline{PSEN}$ Low to Valid Instr In 8751H All Others		 100 125		 3TCLCL−150 3TCLCL−125	 ns ns
TPXIX	Input Instr Hold after $\overline{PSEN}$	0		0		ns
TPXIZ	Input Instr Float after $\overline{PSEN}$		63		TCLCL−20	ns
TPXAV	$\overline{PSEN}$ to Address Valid	75		TCLCL−8		ns
TAVIV	Address to Valid Instr In 8751H All Others		 267 302		 5TCLCL−150 5TCLCL−115	 ns ns
TPLAZ	$\overline{PSEN}$ Low to Address Float		20		20	ns
TRLRH	$\overline{RD}$ Pulse Width	400		6TCLCL−100		ns
TWLWH	$\overline{WR}$ Pulse Width	400		6TCLCL−100		ns
TRLDV	$\overline{RD}$ Low to Valid Data In		252		5TCLCL−165	ns
TRHDX	Data Hold after $\overline{RD}$	0		0		ns
TRHDZ	Data Float after $\overline{RD}$		97		2TCLCL−70	ns
TLLDV	ALE Low to Valid Data In		517		8TCLCL−150	ns
TAVDV	Address to Valid Data In		585		9TCLCL−165	ns
TLLWL	ALE Low to $\overline{RD}$ or $\overline{WR}$ Low	200	300	3TCLCL−50	3TCLCL+50	ns
TAVWL	Address to $\overline{RD}$ or $\overline{WR}$ Low	203		4TCLCL−130		ns
TQVWX	Data Valid to $\overline{WR}$ Transition 8751H All Others	 13 23		 TCLCL−70 TCLCL−60		 ns ns
TQVWH	Data Valid to $\overline{WR}$ High	433		7TCLCL−150		ns
TWHQX	Data Hold after $\overline{WR}$	33		TCLCL−50		ns
TRLAZ	$\overline{RD}$ Low to Address Float		20		20	ns
TWHLH	$\overline{RD}$ or $\overline{WR}$ High to ALE High 8751H All Others	 33 43	 133 123	 TCLCL−50 TCLCL−40	 TCLCL+50 TCLCL+40	 ns ns

NOTE:
*This table does not include the 8751-8 A.C. characteristics (see next page).

Courtesy of INTEL Corp.

This Table is only for the 8751H-8

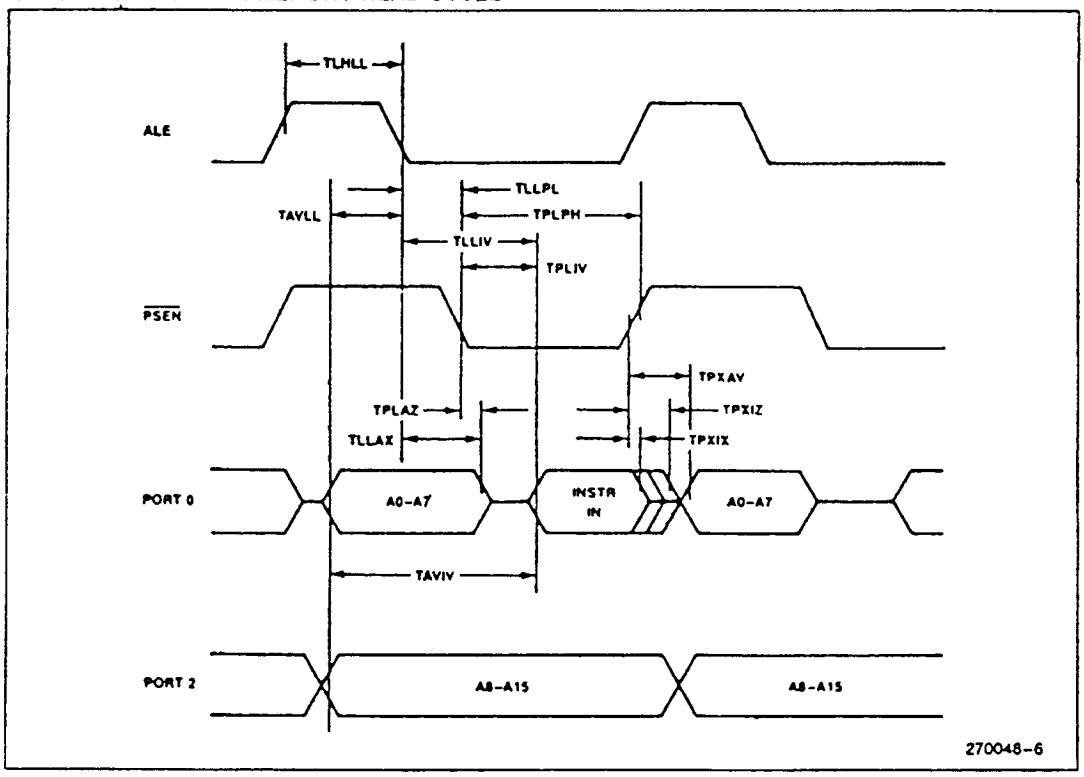
A.C. CHARACTERISTICS

T_A = 0°C to +70°C; V_{CC} = 5V ± 10%; V_{SS} = 0V;
Load Capacitance for Port 0, ALE, and $\overline{PSEN}$ = 100 pF;
Load Capacitance for All Other Outputs = 80 pF

Symbol	Parameter	8 MHz Oscillator		Variable Oscillator		Units
		Min	Max	Min	Max	
1/TCLCL	Oscillator Frequency			3.5	8.0	MHz
TLHLL	ALE Pulse Width	210		2TCLCL−40		ns
TAVLL	Address Valid to ALE Low	85		TCLCL−40		ns
TLLAX	Address Hold after ALE Low	90		TCLCL−35		ns
TLLIV	ALE Low to Valid Instr In		350		4TCLCL−150	ns
TLLPL	ALE Low to $\overline{PSEN}$ Low	100		TCLCL−25		ns
TPLPH	$\overline{PSEN}$ Pulse Width	315		3TCLCL−60		ns
TPLIV	$\overline{PSEN}$ Low to Valid Instr In		225		3TCLCL−150	ns
TPXIX	Input Instr Hold after $\overline{PSEN}$	0		0		ns
TPXIZ	Input Instr Float after $\overline{PSEN}$		105		TCLCL−20	ns
TPXAV	$\overline{PSEN}$ to Address Valid	117		TCLCL−8		ns
TAVIV	Address to Valid Instr In		475		5TCLCL−150	ns
TPLAZ	$\overline{PSEN}$ Low to Address Float		20		20	ns
TRLRH	$\overline{RD}$ Pulse Width	650		6TCLCL−100		ns
TWLWH	$\overline{WR}$ Pulse Width	650		6TCLCL−100		ns
TRLDV	$\overline{RD}$ Low to Valid Data In		460		5TCLCL−165	ns
TRHDX	Data Hold after $\overline{RD}$	0		0		ns
TRHDZ	Data Float after $\overline{RD}$		180		2TCLCL−70	ns
TLLDV	ALE Low to Valid Data In		850		8TCLCL−150	ns
TAVDV	Address to Valid Data In		960		9TCLCL−165	ns
TLLWL	ALE Low to $\overline{RD}$ or $\overline{WR}$ Low	325	425	3TCLCL−50	3TCLCL+50	ns
TAVWL	Address to $\overline{RD}$ or $\overline{WR}$ Low	370		4TCLCL−130		ns
TQVWX	Data Valid to $\overline{WR}$ Transition	55		TCLCL−70		ns
TQVWH	Data Valid to $\overline{WR}$ High	725		7TCLCL−150		ns
TWHQX	Data Hold after $\overline{WR}$	75		TCLCL−50		ns
TRLAZ	$\overline{RD}$ Low to Address Float		20		20	ns
TWHLH	$\overline{RD}$ or $\overline{WR}$ High to ALE High	75	175	TCLCL−50	TCLCL+50	ns

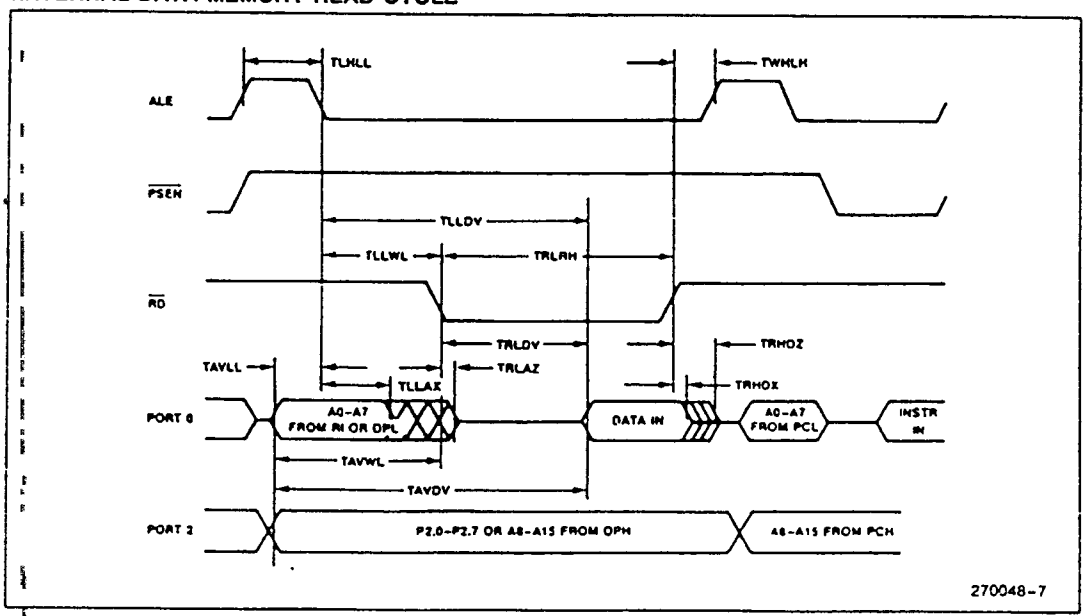
Courtesy of INTEL Corp.

EXTERNAL PROGRAM MEMORY READ CYCLE

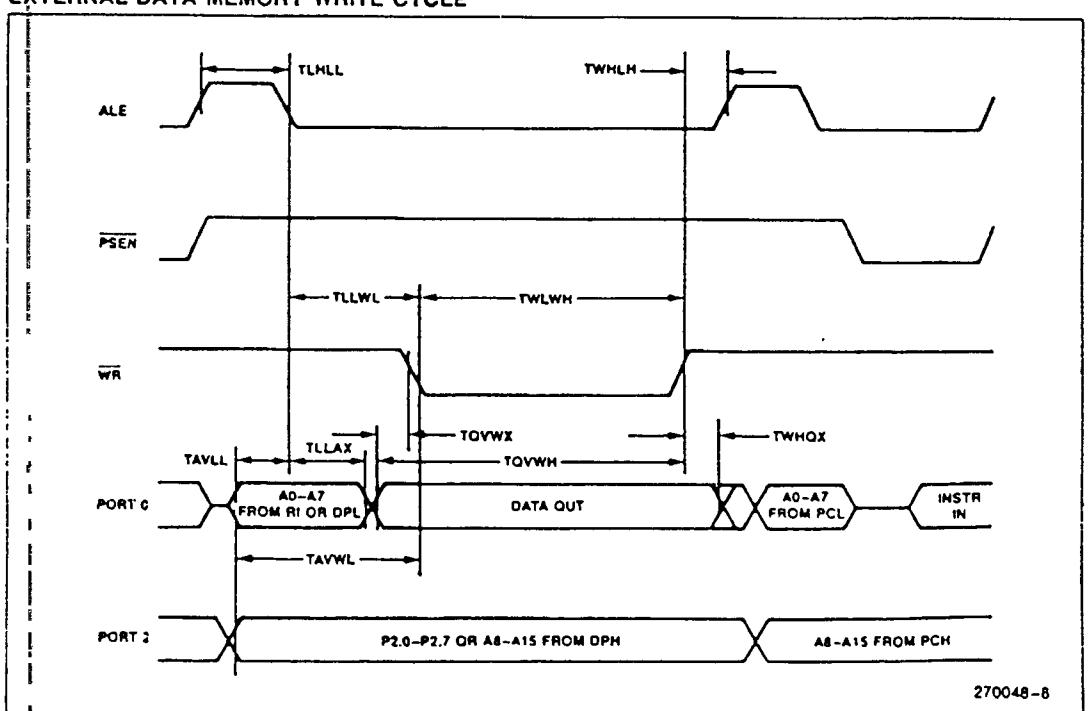


Courtesy of INTEL Corp.

EXTERNAL DATA MEMORY READ CYCLE



EXTERNAL DATA MEMORY WRITE CYCLE



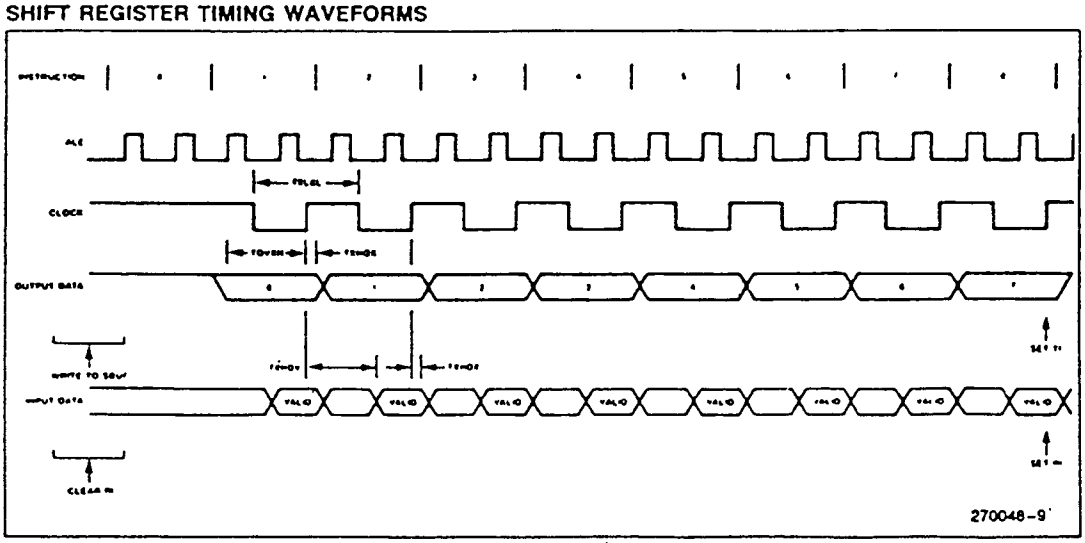
Courtesy of INTEL Corp.

SERIAL PORT TIMING—SHIFT REGISTER MODE

Test Conditions: T_A = 0°C to 70°C; VCC = 5V ±10%; VSS = 0V; Load Capacitance = 80 pF

Symbol	Parameter	12 MHz Oscillator		Variable Oscillator		Units
		Min	Max	Min	Max	
TXLXL	Serial Port Clock Cycle Time	1.0		12TCLCL		μs
TQVXH	Output Data Setup to Clock Rising Edge	700		10TCLCL − 133		ns
TXHQX	Output Data Hold after Clock Rising Edge	50		2TCLCL − 117		ns
TXHDX	Input Data Hold after Clock Rising Edge	0		0		ns
TXHDV	Clock Rising Edge to Input Data Valid		700		10TCLCL − 133	ns

SHIFT REGISTER TIMING WAVEFORMS

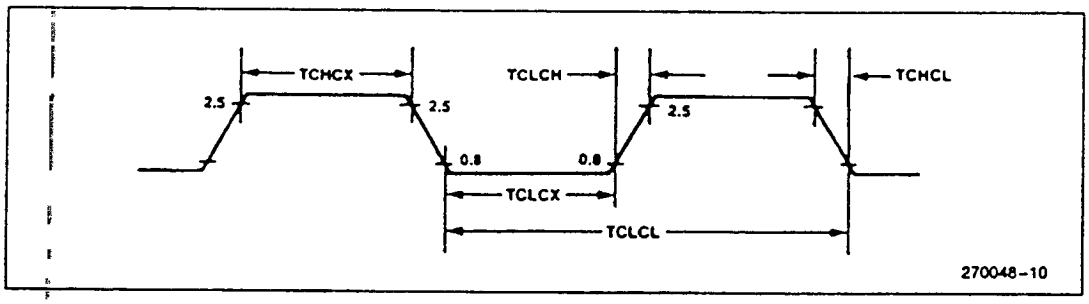


Courtesy of INTEL Corp.

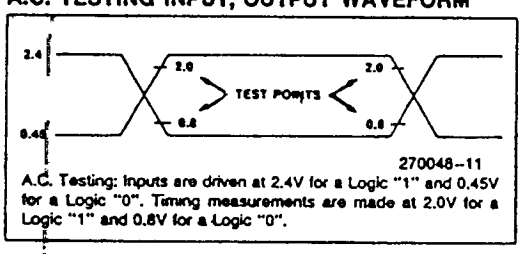
EXTERNAL CLOCK DRIVE

Symbol	Parameter	Min	Max	Units
1/TCLCL	Oscillator Frequency (except 8751H-8) 8751H-8	3.5 3.5	12 8	MHz MHz
TCHCX	High Time	20		ns
TCLCX	Low Time	20		ns
TCLCH	Rise Time		20	ns
TCHCL	Fall Time		20	ns

EXTERNAL CLOCK DRIVE WAVEFORM



A.C. TESTING INPUT, OUTPUT WAVEFORM



A.C. Testing: Inputs are driven at 2.4V for a Logic "1" and 0.45V for a Logic "0". Timing measurements are made at 2.0V for a Logic "1" and 0.8V for a Logic "0".

Courtesy of INTEL Corp.

EPROM CHARACTERISTICS

Table 3. EPROM Programming Modes

Mode	RST	PSEN	ALE	EA	P2.7	P2.6	P2.5	P2.4
Program	1	0	0*	VPP	1	0	X	X
Inhibit	1	0	1	X	1	0	X	X
Verify	1	0	1	1	0	0	X	X
Security Set	1	0	0*	VPP	1	1	X	X

NOTE:
"1" = logic high for that pin
"0" = logic low for that pin
"X" = "don't care"
"VPP" = +21V ±0.5V
*ALE is pulsed low for 50 ms.

Programming the EPROM

To be programmed, the part must be running with a 4 to 6 MHz oscillator. (The reason the oscillator needs to be running is that the internal bus is being used to transfer address and program data to appropriate internal registers.) The address of an EPROM location to be programmed is applied to Port 1 and pins P2.0–P2.3 of Port 2, while the code byte to be programmed into that location is applied to Port 0. The other Port 2 pins, and RST, PSEN, and EA should be held at the "Program" levels indicated in Table 3. ALE is pulsed low for 50 ms to program the code byte into the addressed EPROM location. The setup is shown in Figure 5.

Normally EA is held at a logic high until just before ALE is to be pulsed. Then EA is raised to +21V, ALE is pulsed, and then EA is returned to a logic high. Waveforms and detailed timing specifications are shown in later sections of this data sheet.

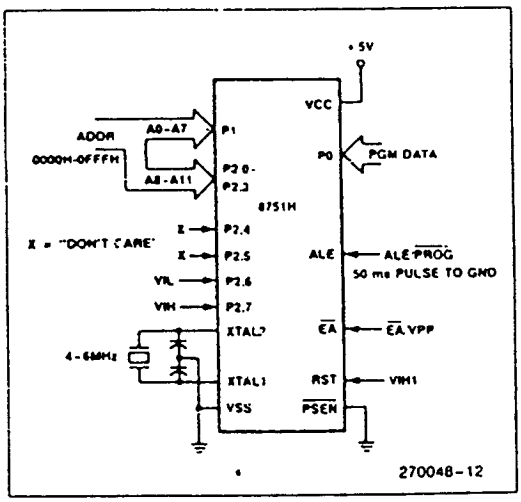


Figure 5. Programming Configuration

Note that the EA/VPP pin must not be allowed to go above the maximum specified VPP level of 21.5V for any amount of time. Even a narrow glitch above that voltage level can cause permanent damage to the device. The VPP source should be well regulated and free of glitches.

Program Verification

If the Security Bit has not been programmed, the on-chip Program Memory can be read out for verification purposes, if desired, either during or after the programming operation. The address of the Program Memory location to be read is applied to Port 1 and pins P2.0–P2.3. The other pins should be held at the "Verify" levels indicated in Table 3. The contents of the addressed location will come out on Port 0. External pullups are required on Port 0 for this operation.

The setup, which is shown in Figure 6, is the same as for programming the EPROM except that pin P2.7 is held at a logic low, or may be used as an active-low read strobe.

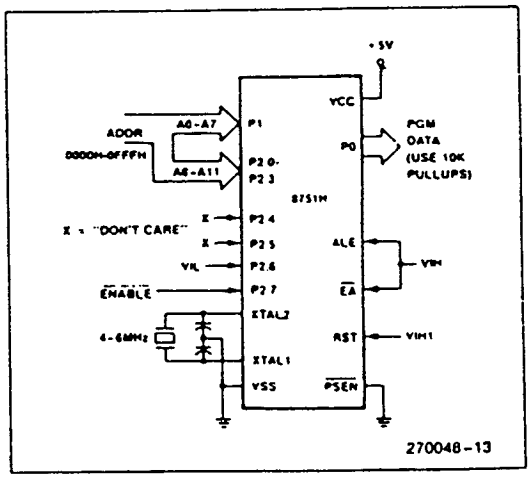


Figure 6. Program Verification

Courtesy of INTEL Corp.

EPROM Security

The security feature consists of a "locking" bit which when programmed denies electrical access by any external means to the on-chip Program Memory. The bit is programmed as shown in Figure 7. The setup and procedure are the same as for normal EPROM programming, except that P2.6 is held at a logic high. Port 0, Port 1, and pins P2.0–P2.3 may be in any state. The other pins should be held at the "Security" levels indicated in Table 3.

Once the Security Bit has been programmed, it can be cleared only by full erasure of the Program Memory. While it is programmed, the internal Program Memory can not be read out, the device can not be further programmed, and it **can not execute out of external program memory**. Erasing the EPROM, thus clearing the Security Bit, restores the device's full functionality. It can then be reprogrammed.

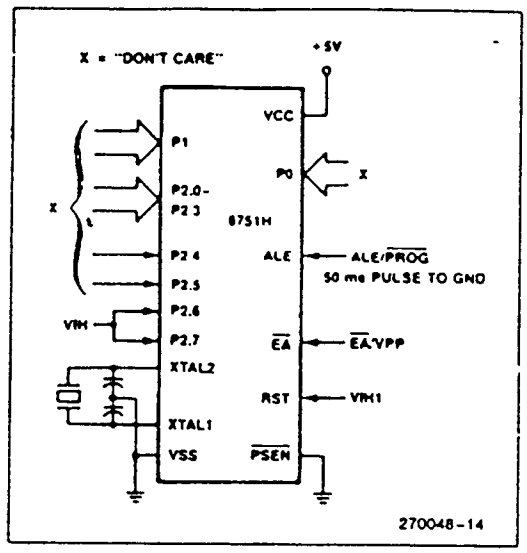


Figure 7. Programming the Security Bit

Erasure Characteristics

Erasure of the EPROM begins to occur when the chip is exposed to light with wavelengths shorter than approximately 4,000 Angstroms. Since sunlight and fluorescent lighting have wavelengths in this range, exposure to these light sources over an extended time (about 1 week in sunlight, or 3 years in room-level fluorescent lighting) could cause inadvertent erasure. If an application subjects the device to this type of exposure, it is suggested that an opaque label be placed over the window.

The recommended erasure procedure is exposure to ultraviolet light (at 2537 Angstroms) to an integrated dose of at least 15 W-sec/cm^2. Exposing the EPROM to an ultraviolet lamp of 12,000 μW/cm^2 rating for 20 to 30 minutes, at a distance of about 1 inch, should be sufficient.

Erasure leaves the array in an all 1s state.

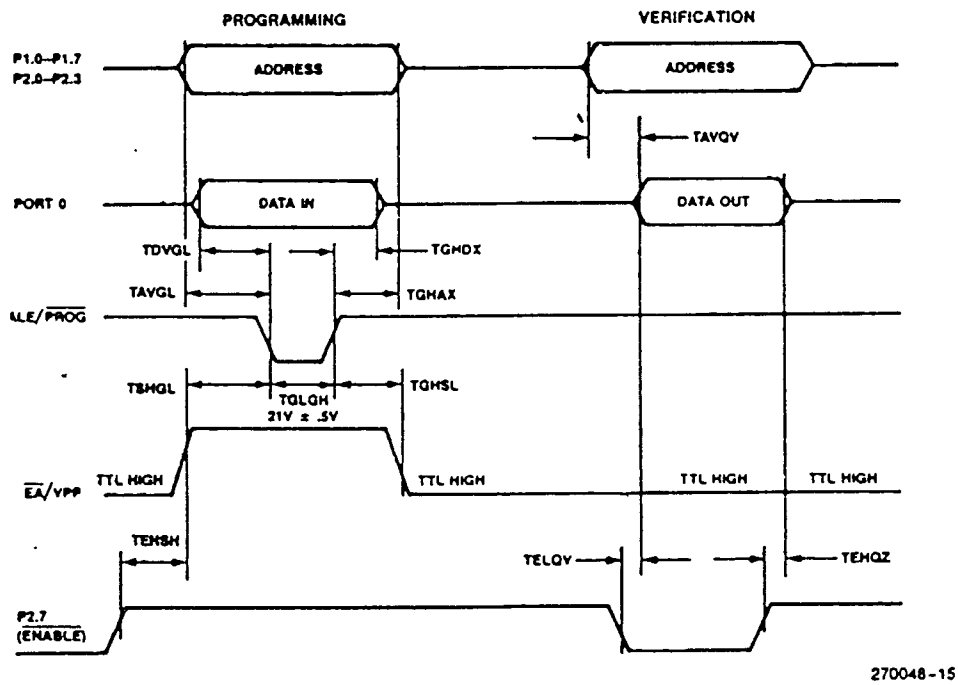
EPROM PROGRAMMING AND VERIFICATION CHARACTERISTICS

T_A = 21°C to 27°C; VCC = 5V ± 10%; VSS = 0V

Symbol	Parameter	Min	Max	Units
VPP	Programming Supply Voltage	20.5	21.5	V
IPP	Programming Supply Current		30	mA
1/TCLCL	Oscillator Frequency	4	6	MHz
TAVGL	Address Setup to $\overline{\text{PROG}}$ Low	48TCLCL		
TGHAX	Address Hold after $\overline{\text{PROG}}$	48TCLCL		
TDVGL	Data Setup to $\overline{\text{PROG}}$ Low	48TCLCL		
TGHDX	Data Hold after $\overline{\text{PROG}}$	48TCLCL		
TEHSH	P2.7 ($\overline{\text{ENABLE}}$) High to VPP	48TCLCL		
TSHGL	VPP Setup to $\overline{\text{PROG}}$ Low	10		µs
TGHSL	VPP Hold after $\overline{\text{PROG}}$	10		µs
TGLGH	$\overline{\text{PROG}}$ Width	45	55	ms
TAVQV	Address to Data Valid		48TCLCL	
TELQV	$\overline{\text{ENABLE}}$ Low to Data Valid		48TCLCL	
TEHQZ	Data Float after $\overline{\text{ENABLE}}$	0	48TCLCL	

Courtesy of INTEL Corp.

EPROM PROGRAMMING AND VERIFICATION WAVEFORMS



For programming conditions see Figure 5. For verification conditions see Figure 6.

DATA SHEET REVISION SUMMARY

The following are the key differences between this and the -003 version of this data sheet:

1. Introduction was expanded to include product descriptions.
2. Package table was added.
3. Design Considerations added.
4. Test Conditions for I_{IL1} and I_{IH} specifications added to the DC Characteristics.
5. Data Sheet Revision Summary added.

Courtesy of INTEL Corp.

1) ยืน ภู่วรวรรณ, วัฒนา เชียงกูล "ไมโครโพรเซสเซอร์ ไมโครคอมพิวเตอร์"
2) สุนทร วิทูสุรพจน์ "การใช้งานไมโครคอนโทรลเลอร์ ตระกูล 8051"
3) สุนทร วิทูสุรพจน์ "การโปรแกรมภาษาแอสเซมบลี้ของไมโครคอนโทรลเลอร์ ตระกูล 8051"
4) ดร. อิน มาน หยาง, ทศพล ปราชญสมพงษ์ "การออกแบบระบบไมโครคอนโทรลเลอร์ ตระกูล 8051"
5) รัชนัย อินทุใส, ไตรภพ อินทุใส "ไมโครคอนโทรลเลอร์ 8051"
6) รังสรรค์ แก้วสรรค์, "การติดตั้งและการใช้งานจานรับสัญญาณดาวเทียม"
7) EIT, "ET Hardware Lab Experiment",Ett Co., Ltd, 1990