ระบบการแสดงผลข้อมูลแบบโต้ตอบได้

นาย โรจฤทธิ์ สว่างแจ้ง

นาย เสริมชัย เกียรติยศนากุล

นางสาว อัมพร โชคชัยตระกูลโชติ

โครงงานนี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรวิทยาศาสตร์บัณฑิต

ภาควิชาฟิสิกส์ประยุกต์

คณะวิทยาศาสตร์

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหาร ลาดกระบัง

ปีการศึกษา 2536

Interactive Viewdata

Mr. Roterit Swangchang

Mr. Sermchai Kiattiyosnakul

Miss Amporn Chokchaitragoonpho

A Special Project Submitted in Partial Fulfillment of the

Requirement for the Degree of Bachelor of Science

Department of Applied Physics

Faculty of Science

King Mongkut's Institute of Technology Ladkrabang

1993

หัวข้อโครงงานพิเศษ	ระบบการแสดงผลข้อมูลแบบโต้ตอบได้
โดย	นาย โรจฤทธิ์ สว่างแจ้ง
	นาย เสริมชัย เกียรติยศนากุล
	นางสาว อัมพร โชคชัยตระกูลโพธิ์
ภาควิชา	ฟิสิกส์ประยุกต์
อาจารย์ที่ปรึกษา	ผศ.ดร. วราวุฒิ เถาลัดดา

ภาควิชาฟิสิกส์ประยุกต์ คณะวิทยาศาสตร์ สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหาร ลาดกระบัง อนุมัติให้นับโครงงานพิเศษฉบับนี้ เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรวิทยาศาสตร์บัณฑิต

..... หัวหน้าภาควิชาฟิสิกส์ประยุกต์

(ผศ. ปรีชา เทียนสมประสงค์)

คณะกรรมการสอบโครงงานพิเศษ

..... ประธานกรรมการ

(ผศ. ปรีชา เทียนสมประสงค์)

..... กรรมการ

(อ. อนุชิต จารุวนาวัฒน์)

..... กรรมการ

(อ. วิชาญ ~~ฉกนนทา~~)

หัวข้อโครงงานพิเศษ **Interactive Viewdata**

ผู้จัดทำโครงงาน 1) นายโรจฤทธิ์ สว่างแจ้ง รหัส 33504031

2) นายเสริมชัย เกียรติยศนากุล รหัส 33504044

3) นางสาวอัมพร โชคชัยตระกูลโพธิ์ รหัส 33504049

ภาควิชา ฟิสิกส์ประยุกต์

อาจารย์ที่ปรึกษา ผศ.ดร.วราวุฒิ เถาลัดดา

บทคัดย่อ

Interactive Viewdata เป็นโครงงานที่ศึกษาการรับส่งข้อมูลจากศูนย์ข้อมูลผ่านทางสายโทรศัพท์มายังผู้ใช้ โดยมีโทรทัศน์เป็นตัวแสดงผล (Monitor) และใช้คีย์บอร์ดเป็นตัวเลือกข้อมูล ในส่วนของฮาร์ดแวร์นั้นใช้ Z80180 CPU เป็นตัวควบคุมส่วนกลาง ,6845 CRTC เป็นตัวควบคุมการแสดงผล และมีการสื่อสารข้อมูลผ่านทางโมเดม (MODEM) ทั้งนี้จะมีซอฟแวร์เป็นตัวควบคุมการทำงานทั้งหมด ในการประยุกต์ใช้งาน เช่น การสั่งจองตั๋วเครื่องบิน การสั่งซื้อของจากซุปเปอร์มาเก็ต การแสดงราคาหุ้น เป็นต้น

Special Project Title	Interactive Viewdata
Name	Mr. Roterit Swangchang
	Mr. Sermchai Kiattiyosnakul
	Miss Amporn Chokchaitragoonpho
Special Project Advisor	Dr. Warawoot Thowladda
Department	Applied Physics
Acadamic Year	1993

Abstract

Interactive Viewdata was developed in this special project. It has an objective to study about the data communication between the computer center and the receiving data terminal. Data from data base is transfered through the telephone lines by using a modem. T.V. monitor is used as a data terminal which controlled by 6845 CRT Controller. The operations of this system are controlled by Z80180 CPU.

กิตติกรรมประกาศ

ในนามของผู้จัดทำโครงงานพิเศษ Interactive Viewdata ขอแสดงความขอบคุณทุกท่านมา ณ ที่นี้

1. บิดาและมารดา ที่ให้การอุปการะเลี้ยงดู ตลอดจนให้เงินสนับสนุนในการศึกษา
2. ผศ.ดร.วราวุฒิ เถาลัดดา ให้คำปรึกษาแนะนำ ช่วยเหลือ และให้ความสะดวกในการทำโครงงานพิเศษ
3. อ.วิชิต ศิริโชติ ให้คำปรึกษาแนะนำ และให้แนวทางในการแก้ปัญหา
4. ผศ. อนุพงศ์ สรงประภา กรุณาให้ยืมหนังสือ
5. อ.วิชาญ กกกนทา กรุณาให้ยืมหนังสือ
6. อ.ชวัชชัย ให้ความสะดวกอย่างมากมาย และช่วยเหลือในด้านอุปกรณ์ต่าง ๆ
7. พี่สมภพ พิสิกส์ประยุกต์ รุ่น 4 ให้ยืมเครื่องอัด EPROM
8. คุณเลอศักดิ์ คณิตศาสตร์ประยุกต์ รุ่น 9 ให้ยืมโทรทัศน์ และช่วยถ่ายรูป
9. กิตติชัย อภินทนาพงศ์,ทรงวิทย์ เจริญรวยวัฒนา แนะนำการแก้ปัญหา
10. ภูเบศร์ อุดมทรัพย์ ให้ยืมหนังสือ และเครื่องพิมพ์
11. นิพนธ์ จันทร์ดรี ให้ยืมหนังสือ
12. นรสีห์ สิงหลกะ และ ภัทรียา ฐาปนะพาหะ ให้ยืมเครื่องพิมพ์
13. ชิปไดย ตัณฑ์ประพันธ์ ช่วยเหลือกระดาษทำเอกสาร
14. ปัญญ์ชนัตถ์ แก้วทิพยเนตร เอื้อเฟื้อพาหนะ
15. นาถ ดนตรี และ สุวรรณ ลิขิตวัฒนานุรักษ์ ช่วยเหลือในการทำเอกสาร
16. ประวิทย์ หัตถโชติ,รุ่งเรือง แซ่โค้ว และ สุภัทชดา ผาสุข ช่วยเหลืออุปกรณ์การทำเอกสาร
17. พี่ ๆ เพื่อน ๆ และน้อง ๆ ทุกคน ที่เป็นกำลังใจ และคอยถามไถ่ตลอดเวลา
18. ประเทศชาติ และประชาชนไทยทุกท่าน ที่เสียภาษีให้พวกเราได้มีโอกาสเรียนหนังสือ

โรจฤทธิ์ สว่างแจ้ง

เสริมชัย เกียรติยศนากุล

อัมพร โชคชัยตระกูลโพธิ์

ผู้จัดทำ

สารบัญ

	หน้า
บทที่ 1 บทนำ	1
- หลักการของตัวแปลงสัญญาณ	3
- ประโยชน์ของโครงงาน	4
- วิธีดำเนินงาน	4
บทที่ 2 ระบบไมโครโปรเซสเซอร์ Z80180	6
- ลักษณะทั่วไป	6
- รายละเอียดและหน้าที่ที่สำคัญของสายสัญญาณต่าง ๆ	7
- การอินเทอร์รัพท์	12
- รีจิสเตอร์อินพุทเอาท์พุทภายใน	17
- Operation Mode	22
- Timing	23
- Wait State Generator	24
- Dynamic RAM Refresh Control	26
- DMA Controller	27
- Asybchronous Serial Communication Interface	28
- Programmable Reload Timer	34
- Memory Management Unit	36
บทที่ 3 ปีพีพีพอร์ท	44
- ลักษณะทั่วไปของ ปีพีไอซี 8255	44
- ฟังก์ชันการทำงานของขาสัญญาณต่าง ๆ	44
- การโปรแกรมค่า 8255	45
- โหมดการทำงาน	47

- ลักษณะทั่วไปของ ชิพไอซี 8250 52
- ฟังก์ชันการทำงานของขาสัญญาณต่าง ๆ 52
- การใช้งานรีจิสเตอร์ต่าง ๆ 56
- ลักษณะทั่วไปของ ชิพไอซี 6845 CRT Controller 66
- หน้าที่ของขาสัญญาณต่าง ๆ 67
- รีจิสเตอร์ภายใน 69
- การโปรแกรมค่าลงในรีจิสเตอร์ภายใน 72
- ลักษณะทั่วไปของชิพไอซี LM 1889 TV Modulator 76

บทที่ 4 หลักการเบื้องต้นของการสื่อสาร 77
- การสื่อสารข้อมูล 77
- รูปแบบของการติดต่อสื่อสารแบบอนุกรม 79
- การสื่อสารข้อมูลแบบอะซิงโครนัส 80
- การสื่อสารข้อมูลแบบซิงโครนัส 81
- การตรวจจับข้อมูลที่ผิดพลาดในการสื่อสารแบบอะซิงโครนัส 82
- ลักษณะทั่วไปของ RS-232-C 84
- คุณสมบัติเชิงกลของการเชื่อมต่อ 85
- คุณสมบัติของสัญญาณไฟฟ้า 87
- ลักษณะการทำงานของวงจรต่าง ๆ 89
- ลักษณะการต่อ RS-232-C 99
- โมเด็ม 100
- เทคนิคการมอดดูเลท 100
- มาตรฐานค่าสั่งของโมเด็ม 103

บทที่ 5 หน่วยแสดงผล 110
- หลักการพื้นฐานของโทรทัศน์ 110
- รายละเอียดของสัญญาณต่าง ๆ 111

บทที่ 6 หลักการทำงานของระบบ 113
- หลักการทำงานของส่วนแสดงผล 113
- ลักษณะของวงจรต่าง ๆ 115
- การจัดแอดเดรสของ Character Generator 120
- หลักการทำงานของวงจรคีย์บอร์ด 124

- ลักษณะของวงจร 127
- ฟังก์ชันการทำงานของแต่ละคีย์ 128
- หลักการทำงานของโมเด็ม 130
- หลักการทำงานของเครื่องพิมพ์ 131
- แผนผังการจัดหน่วยความจำ 134

บทที่ 7 ผลการทดลอง 137
- สัญญาณต่าง ๆ ที่ใช้แสดงจอภาพ 137
- โมเด็มและพอร์ทอนุกรม 140
- คีย์บอร์ดและเครื่องพิมพ์ 141
- การทำงานของทั้งระบบ 141

บทที่ 8 สรุปผลวิจารณ์และข้อเสนอแนะ 142
- การพัฒนาในอนาคต 142

ภาคผนวก ก Data Sheet 144
- LM 1889 TV Video Modulator 144
- MC6845 152

ภาคผนวก ข โปรแกรมสำหรับ Z80180 MPU Home Center 176
- ส่วน Flow Chart 176
- ส่วนโปรแกรม 196

ภาคผนวก ค โปรแกรมสำหรับ PC Data-Base Center 238
- ส่วน Flow Chart 238
- ส่วนโปรแกรม 247

ประวัติผู้เขียน 272

เอกสารอ้างอิง 274

สารบัญรูป

หน้า

1. รูปที่ 1.1 แสดงแผนผังการทำงานของระบบ Interactive Viewdata 1
2. รูปที่ 1.2 แสดงแผนผังการทำงานของตัวแปลงสัญญาณ 2
3. รูปที่ 2.1a แสดงแผนภาพโครงสร้างภายในของชิพ Z80180 CPU 6
4. รูปที่ 2.1b แสดงแผนภาพโครงสร้างภายนอกของชิพ Z80180 CPU 7
5. รูปที่ 2.2 แสดงบิทต่าง ๆ ใน INT/TRAP Control Register 14
6. รูปที่ 2.3 แสดงบิทต่าง ๆ ใน I/O Control Register 21
7. รูปที่ 2.4 แสดงบิทต่าง ๆ ใน Operation Mode Control Register 22
8. รูปที่ 2.5 แสดงการเกิดสัญญาณ M1 ที่ไซเคิลถัดไปของการ Fetch Opcode 22
9. รูปที่ 2.6 แสดง I/O Read Write Cycle 23
10. รูปที่ 2.7 แสดงการเปรียบเทียบ T State ระหว่าง Z80 CPU กับ Z80180 CPU 24
11. รูปที่ 2.8 WAIT Timing 24
12. รูปที่ 2.9 แสดง Memory and I/O Wait State Insertion 25
13. รูปที่ 2.10 แสดงบิทต่าง ๆ ใน Refresh Control Register 26
14. รูปที่ 2.11 แสดงบล็อคไดอะแกรมของ ASCI 28
15. รูปที่ 2.12 แสดงบิทต่าง ๆ ใน Status Register 0,1 29
16. รูปที่ 2.13 แสดงบิทต่าง ๆ ใน Control Register A 0,1 31
17. รูปที่ 2.14 แสดงบิทต่าง ๆ ใน ASCI Control Register B 0,1 33
18. รูปที่ 2.15 แสดงบิทต่าง ๆ ใน Timing Control Register 35
19. รูปที่ 2.16 แสดงรูปแบบการจัดหน่วยความจำแบบต่าง ๆ 36
20. รูปที่ 2.17 Map Logical แบบปกติ 37
21. รูปที่ 2.18 แสดงบิทต่าง ๆ ใน Common/Bank Area Register 39
22. รูปที่ 2.19 แสดงการจัดหน่วยความจำตามตัวอย่าง 39
23. รูปที่ 2.20 แสดง Logical และ Physical memory ตามตัวอย่าง 41
24. รูปที่ 2.21 แสดงการจัดหน่วยความจำโดยทั่วไปในส่วน Logical 41
25. รูปที่ 2.22 แสดงการแบ่ง Map เป็นลักษณะกว้าง ๆ 42
26. รูปที่ 2.23 แสดงการจัดหน่วยความจำหลังจากรีเซ็ต 43

27. รูปที่ 3.1 แสดงแผนผังวงจรภายในและภารจัดขาของไอซี 8255 44
28. รูปที่ 3.2 แสดงความหมายของบิทต่าง ๆ ในรหัสควบคุม 46
29. รูปที่ 3.3 แสดงการเซ็ตค่าต่าง ๆ ของ 8255 47
30. รูปที่ 3.4a แสดงโครงสร้างตัวตรวจสอบสัญญาณของพอร์ทอินพุท 47
31. รูปที่ 3.4b แสดงโครงสร้างตัวตรวจสอบสัญญาณของพอร์ทเอาท์พุท 48
32. รูปที่ 3.5 แสดงวงจรการต่อ 8255 ในโหมด 1 48
33. รูปที่ 3.6 แสดงแผนผังเวลาการรับและส่งข้อมูลโดยใช้ตัวตรวจสอบสัญญาณ 50
34. รูปที่ 3.7 แสดงโครงสร้างของพอร์ท A ที่ทำงานแบบพอร์ท 2 ทิศทาง 51
35. รูปที่ 3.8 แสดงการจัดขาของชิพไอซี 8250 52
36. รูปที่ 3.9 แสดงค่าของบิทในรีจิสเตอร์ควบคุมสายสื่อสาร 57
37. รูปที่ 3.10 แสดงค่าของบิทต่าง ๆ ในรีจิสเตอร์แสดงสถานะสายสื่อสาร 59
38. รูปที่ 3.11 แสดงค่าของบิทในรีจิสเตอร์กำหนดอินเทอร์รัพท์ 61
39. รูปที่ 3.12 แสดงค่าของบิทในรีจิสเตอร์อีนาเบิลอินเทอร์รัพท์ 63
40. รูปที่ 3.13 แสดงค่าของบิทต่าง ๆ ในรีจิสเตอร์ควบคุมโมเด็ม 64
41. รูปที่ 3.14 แสดงค่าของบิทต่าง ๆ ในรีจิสเตอร์แสดงสถานะโมเด็ม 65
42. รูปที่ 3.15a แสดงผลในโหมดนอนอินเทอร์เลซ 71
43. รูปที่ 3.15b แสดงผลในโหมดอินเทอร์เลซ 71
44. รูปที่ 3.15c แสดงผลในโหมดอินเทอร์เลซวีดีโอ 71
45. รูปที่ 3.16 แสดงการกำหนดขนาดของเคอร์เซอร์ 72
46. รูปที่ 3.17 แสดงแผนผังการจัดขาสัญญาณต่าง ๆ ของไอซี LM1889 76
47. รูปที่ 4.1 แสดงการส่งข้อมูลแบบขนานระหว่าง CPU กับหน่วยความจำ 77
48. รูปที่ 4.2 แสดงการส่งข้อมูลแบบอนุกรมระหว่างตัวส่งและตัวรับ 78
49. รูปที่ 4.3 แสดงการส่งข้อมูลแบบ Simplex 79
50. รูปที่ 4.4 แสดงการส่งข้อมูลแบบ Half Duplex 80
51. รูปที่ 4.5 แสดงการส่งข้อมูลแบบ Full Duplex 80
52. รูปที่ 4.6 แสดงรูปแบบของการสื่อสารแบบอะซิงโครนัส 81
53. รูปที่ 4.7 แสดงรูปแบบข้อมูลแบบ Character Oriented Frame 81
54. รูปที่ 4.8 แสดงรูปแบบข้อมูลแบบ Bit Oriented Frame 82
55. รูปที่ 4.9 แสดงลักษณะของ DTE และ DCE ที่ใช้ในการสื่อสารข้อมูล 85
56. รูปที่ 4.10 แสดงลักษณะของคอนเน็กเตอร์ DB-25P และ DB-25S 86

57. รูปที่ 4.11 แสดง RS-232-C Voltage Range 87
58. รูปที่ 4.12 แสดงวงจรการเชื่อมต่อ RS-232-C 88
59. รูปที่ 4.13 แสดงผลของ RI ที่มีต่อ Ringing Signal 91
60. รูปที่ 4.14 แสดงสถานะการ "ON" และ "OFF" ของ DCD ในกรณีของ ฟลูดูเพล็กซ์ 92
61. รูปที่ 4.15 แสดง Timing ของสัญญาณ RTS และ CTS 92
62. รูปที่ 4.16 แสดง Timing ของวงจร DA 93
63. รูปที่ 4.17 แสดง Timing ของวงจร DB 94
64. รูปที่ 4.18 แสดง Timing ของวงจร DD 94
65. รูปที่ 4.19 แสดง Secondary Channel โดยใช้เป็น Backward Channel 95
66. รูปที่ 4.20 แสดงขั้นตอนการทำงานของฝ่ายเรียก 97
67. รูปที่ 4.21 แสดงขั้นตอนการทำงานของฝ่ายรับ 98
68. รูปที่ 4.22 แสดงการต่อแบบฟลูดูเพล็กซ์ตามมาตรฐานที่กำหนด 99
69. รูปที่ 4.23 แสดงการต่อแบบ Three-wire Economic Model 99
70. รูปที่ 4.24 แสดงการต่อแบบ Three-wire with Luxury Loop-Back 100
71. รูปที่ 4.25 แสดงการมอดดูเลทสัญญาณแบบ FSK 101
72. รูปที่ 4.26a แสดงการมอดดูเลทสัญญาณแบบ PSK เปลี่ยน 180 องศา 102
73. รูปที่ 4.26b แสดงการมอดดูเลทสัญญาณแบบ PSK เปลี่ยน 90 องศา 102
74. รูปที่ 4.27 แสดงการมอดดูเลทสัญญาณแบบ PAM 102
75. รูปที่ 4.28 ตารางแสดง S-Register ของ Hayes 104
76. รูปที่ 5.1 แสดงการสแกนแบบก้าวหน้า 110
77. รูปที่ 5.2 แสดงการสแกนแบบสลับเส้น 111
78. รูปที่ 5.3 แสดงระดับเกร็ยเสกล 111
79. รูปที่ 6.1 แสดงแผนผังการทำงานของการแปลงสัญญาณในส่วนแสดงผล 114
80. รูปที่ 6.2 แสดงวงจรของบัพเฟอร์ทิศทางเดียว และการต่อบัสแอดเดรส เข้ากับบัพเฟอร์ 115
81. รูปที่ 6.3 แสดงวงจรของบัพเฟอร์สองทิศทาง และการต่อบัสข้อมูลเข้ากับ บัพเฟอร์ 115
82. รูปที่ 6.4 แสดงการสร้างสัญญาณในการเข้าถึงพอร์ทอินพุทเอาท์พุทและ หน่วยความจำ 116

83. รูปที่ 6.5a แสดงการถอดรหัสพอร์ท 116
84. รูปที่ 6.5b แสดงวงจรการถอดรหัสหน่วยความจำ 117
85. รูปที่ 6.6 แสดงวงจรสร้าง Clock 118
86. รูปที่ 6.7 แสดงวงจรการทำงานในส่วนแสดงผล 119
87. รูปที่ 6.8 แสดง Dot Matrix ขนาด 8x11 120
88. รูปที่ 6.9 แสดงข้อมูลในแต่ละเส้นสแกนของฟรอนท์ตัวอักษร "A" 122
89. รูปที่ 6.10 แสดงส่วนสร้างสัญญาณ Composite Video 123
90. รูปที่ 6.11 แสดงส่วนของ RF Modulator 123
91. รูปที่ 6.12 แสดงการตอบสนองต่อการอินเทอร์รัพท์ 125
92. รูปที่ 6.13 แสดงแผนผังการทำงานของส่วนคีย์บอร์ด 126
93. รูปที่ 6.14 แสดงวงจรส่วนคีย์บอร์ด 129
94. รูปที่ 6.15 แสดงการเชื่อมต่อของคอนเน็กเตอร์ระหว่างซีพียูกับโมเด็ม 130
95. รูปที่ 6.16 แสดงรูปแบบของข้อมูลแบบอนุกรม 131
96. รูปที่ 6.17 แสดงสัญญาณต่าง ๆ ของการส่งข้อมูลไปยังเครื่องพิมพ์ 132
97. รูปที่ 6.18 แสดงการต่อพอร์ทเครื่องพิมพ์เข้ากับพอร์ทของ 8255 133
98. รูปที่ 6.19 แสดงการจัดแอดเดรสของหน่วยความจำที่ใช้ในโครงงาน 134
99. รูปที่ 7.1 แสดงรูปสัญญาณ Horizontal Synchronous (HSYNC) 137
100. รูปที่ 7.2 แสดงรูปสัญญาณ Vertical Synchronous (VSYNC) 137
101. รูปที่ 7.3 แสดงรูปสัญญาณ Display Enable (DISPEN) 138
102. รูปที่ 7.4 แสดงรูปสัญญาณวีดีโอ 138
103. รูปที่ 7.5 แสดงรูปสัญญาณ Composite Video 139
104. รูปที่ 7.6 การแสดงผลออกทางหน้าจอโทรทัศน์ 139

สารบัญตาราง

หน้า

1. ตารางที่ 2.1 แสดงความสัมพันธ์ระหว่างสัญญาณ ST,HALT,M1 10
2. ตารางที่ 2.2 แสดงการโปรแกรมรีจิสเตอร์ IL สำหรับ Interrupt Source 13
3. ตารางที่ 2.3 แสดงค่าสภาวะของแฟลก IEF1 และ IEF2 15
4. ตารางที่ 2.4 แสดง I/O Address Map 17
5. ตารางที่ 2.5 แสดงการโปรแกรมค่าในบิท IOA7 และ IOA6 21
6. ตารางที่ 2.6 แสดง Memory Wait States 25
7. ตารางที่ 2.7 แสดง Wait State Insertion 26
8. ตารางที่ 2.8 แสดงการโปรแกรมค่าในบิท CYC1 และ CYC0 27
9. ตารางที่ 2.9 แสดงรูปแบบของข้อมูลเมื่อมีการเซ็ตค่าในบิท MOD2,1,0 32
10. ตารางที่ 2.10 แสดงการกำหนดค่า Baud Rate โดยการเซ็ตค่าต่าง ๆ 34
11. ตารางที่ 2.11 แสดงการกำหนดเอาท์พุทโดยการเซ็ตค่าใน TOC1,0 35
12. ตารางที่ 3.1 แสดงความสัมพันธ์ของแอดเดรส A0 และ A1 กับพอร์ทต่าง ๆ 45
13. ตารางที่ 3.2 แสดงหน้าที่ของสัญญาณต่าง ๆ ของพอร์ท C ในการทำงานเป็นตัวตรวจสอบสัญญาณเมื่อ 8255 ทำงานในโหมด 1 49
14. ตารางที่ 3.3 แสดงการอีนาเบิลอินเทอร์รัพท์โดยการ Set/Reset พอร์ท C 49
15. ตารางที่ 3.4 แสดงการใช้งานพอร์ท C เป็นสัญญาณ Hand Shake 51
16. ตารางที่ 3.5 แสดงการกำหนดค่ารีจิสเตอร์ 53
17. ตารางที่ 3.6 แสดงผลของการรีเซ็ต 54
18. ตารางที่ 3.7 แสดงค่าตัวหารสำหรับการกำหนด Baud Rate 58
19. ตารางที่ 3.8 ฟังก์ชันควบคุมการอินเทอร์รัพท์ 62
20. ตารางที่ 3.9 แสดงหน้าที่ของขาสัญญาณต่าง ๆ ของ 6845 CRTC 67
21. ตารางที่ 3.10 แสดงการกำหนดโหมดการทำงานต่าง ๆ 68
22. ตารางที่ 3.11 ความหมายของรีจิสเตอร์ต่าง ๆ 69
23. ตารางที่ 3.12 แสดงการเลือกชนิดโหมดการอินเทอร์เลชในรีจิสเตอร์ R8 70
24. ตารางที่ 4.1 ข้อเปรียบเทียบระหว่างการส่งข้อมูลแบบขนานและแบบอนุกรม 79
25. ตารางที่ 4.2 แสดงรายละเอียดของขาต่าง ๆ ของคอนเน็คเตอร์ตามมาตรฐานของ RS-232-C 85
26. ตารางที่ 4.3 แสดงความสัมพันธ์ระหว่างสถานะต่าง ๆ กับแรงดัน 87

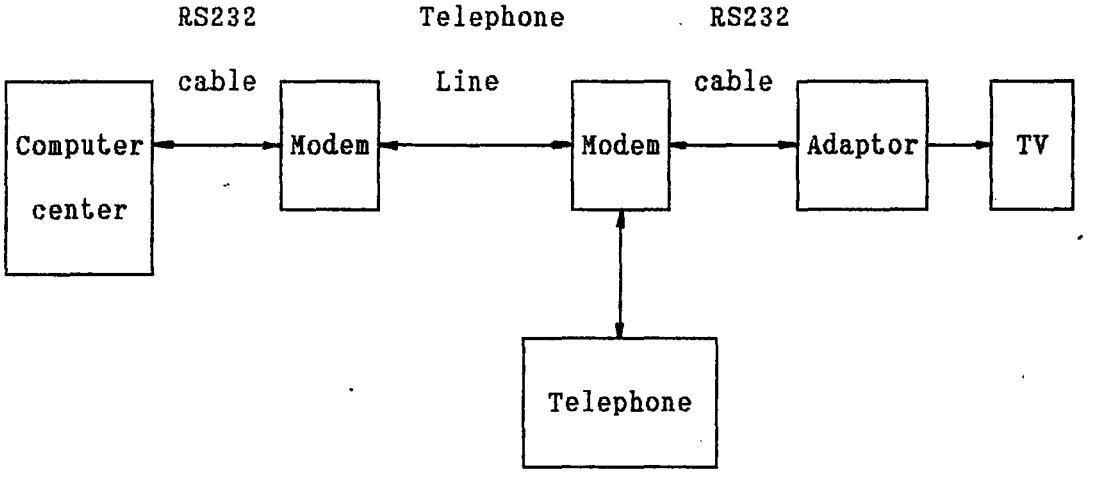
27. ตารางที่ 4.4 แสดง RS-232-C Interchange Circuit 89
28. ตารางที่ 6.1 แสดงการถอดรหัสแอดเดรสของพอร์ทและหน่วยความจำ 116
29. ตารางที่ 6.2 แสดงแอดเดรสที่ใช้ในการเข้าถึง Character Generator 121
30. ตารางที่ 6.3 แสดงค่าการเข้ารหัสของคีย์ต่าง ๆ 127
31. ตารางที่ 6.4 แสดงแผนผังการจัดแอดเดรสของพอร์ทอินพุท/เอาท์พุท 135

บทที่ 1

บทนำ

ปัจจุบันนี้ ได้มีการนำอุปกรณ์ทางด้านอิเล็กทรอนิกส์เข้ามาผสมผสานกับเทคโนโลยีทางด้านการสื่อสาร ทำให้สามารถส่งข่าวสารข้อมูลต่าง ๆ ในรูปของตัวอักษรไปยังเครื่องรับโทรทัศน์ได้ เรียกระบบดังกล่าวว่า "Interactive Viewdata"

ระบบ Interactive Viewdata คือ ระบบที่มีการสื่อสารคมนาคมระหว่างสองฝ่าย (two way link) โดยอาศัยคอมพิวเตอร์มาตอบคำถาม และใช้ช่วยในการถ่ายโอนข้อมูล เมื่อผู้ใช้ต้องการทราบข้อมูล ก็โทรศัพท์เข้ามายังศูนย์คอมพิวเตอร์ จากนั้นศูนย์จะส่งข้อมูลที่เป็นสัญญาณดิจิตอลผ่านหน่วยโมเด็มเพื่อเปลี่ยนให้เป็นสัญญาณเสียงไปยังสายโทรศัพท์ เมื่อข้อมูลถึงปลายทางก็ผ่านหน่วยโมเด็มอีกครั้งหนึ่งเพื่อทำการถอดรหัสข้อมูล แล้วข้อมูลก็ส่งผ่านเข้าไปยังเครื่องแปลงสัญญาณ (Adaptor) เพื่อแสดงผลออกทางหน้าจอโทรทัศน์ ซึ่งสามารถแสดงแผนผังการทำงานได้ดังรูปที่ 1.1



รูปที่ 1.1 แสดงแผนผังการทำงานของระบบ Interactive Viewdata

เมื่อพิจารณาถึงระบบ Interactive Viewdata กับระบบ Teletext นั้นจะพบว่ามีลักษณะที่คล้ายคลึงกันในส่วนของการใช้โทรทัศน์เป็นตัวแสดงผลข้อมูล แต่จริงๆแล้วระบบทั้งสองมีข้อแตกต่างกัน ดังนี้

1. ระบบ Interactive Viewdata จะส่งข้อมูลมาจากศูนย์คอมพิวเตอร์ ผ่านมาตามสายโทรศัพท์ ส่วนระบบ Teletext นั้นส่งข้อมูลมาจากสถานีโทรทัศน์ ซึ่งทำการออกอากาศถ่ายทอดมา

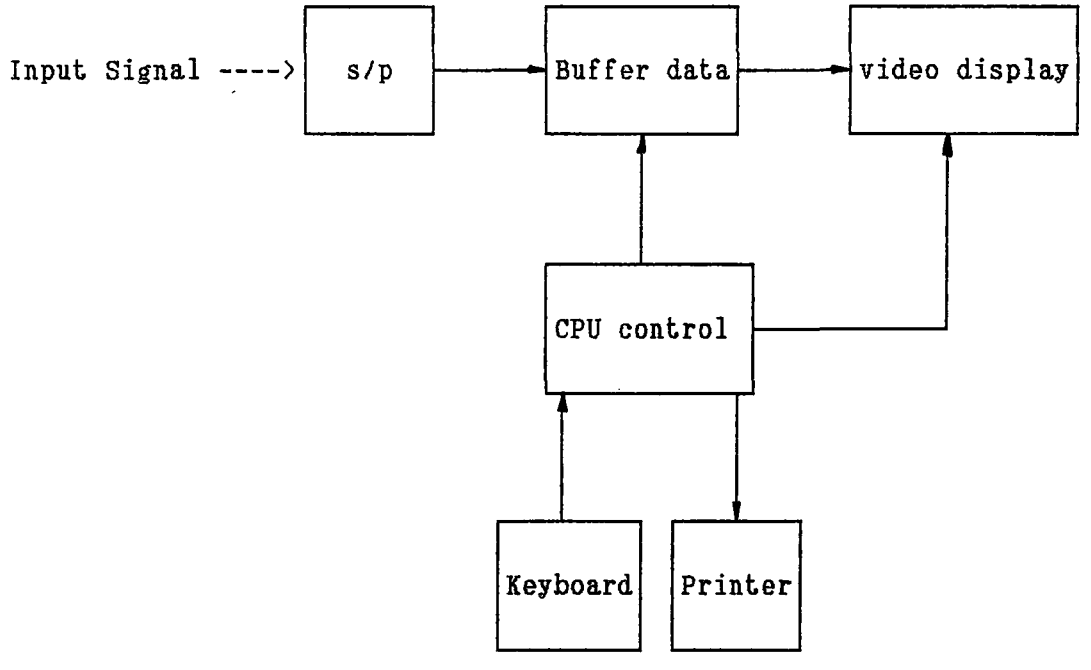
2. ระบบ Interactive Viewdata เป็นระบบการสื่อสารแบบสองทาง ซึ่งผู้ใช้สามารถโต้ตอบกับศูนย์ข้อมูลได้ในทันทีทันใด ส่วนระบบ Teletext นั้นเป็นระบบการสื่อสารแบบทางเดียว ทำให้ผู้ใช้รับทราบข้อมูลได้เพียงอย่างเดียวเท่านั้น

3. ระบบ Interactive Viewdata สามารถส่งข้อมูลเฉพาะส่วนที่ผู้ใช้ต้องการได้ ส่วนระบบ Teletext นั้นจะส่งข้อมูลเป็นจำนวนมากต่อการส่งหนึ่งครั้ง ซึ่งผู้ใช้จะเป็นผู้ระบุหมายเลขหน้าของข้อมูลที่ต้องการโดยทำการล็อคข้อมูลนั้นไว้ให้แสดงออกบนจอโทรทัศน์

4. ระบบ Interactive Viewdata เป็นการบริการข้อมูลให้กับบุคคลใดบุคคลหนึ่งในช่วงเวลาหนึ่ง ๆ ดังนั้นผู้ใช้ต้องจ่ายค่าบริการตามอัตราเวลาของข้อมูล ส่วนระบบ Teletext นั้นเป็นการบริการข้อมูลแก่สาธารณะชน จึงไม่ต้องจ่ายค่าตอบแทน

จากความแตกต่างของระบบทั้งสองที่กล่าวมาข้างต้น สามารถสรุปได้ว่า ถึงแม้ระบบ Interactive Viewdata จะมีราคาที่แพงกว่า แต่ระบบ Interactive Viewdata ก็เป็นระบบที่มีประสิทธิภาพสูง อำนวยความสะดวกให้กับผู้ใช้ได้อย่างมาก และสามารถนำไปประยุกต์ใช้งานได้หลายด้าน เช่น การสั่งจองตั๋วเครื่องบิน รถไฟ การสั่งซื้อสินค้าตามซุปเปอร์มาร์เก็ต การแสดงราคาหุ้น และการเรียนหนังสืออยู่กับบ้าน เป็นต้น

ในส่วนของโครงงานพิเศษนี้ได้เล็งเห็นถึงประโยชน์ที่จะได้รับ จึงได้มีการจัดทำระบบ Interactive Viewdata ขึ้น โดยได้ทำการศึกษาโครงสร้างของตัวแปลงสัญญาณ (Adaptor) ซึ่งมีแผนผังการทำงาน ดังรูปที่ 1.2



รูปที่ 1.2 แสดงแผนผังการทำงานของตัวแปลงสัญญาณ (Adaptor)

หลักการของตัวแปลงสัญญาณ (Adaptor) ประกอบด้วยส่วนต่าง ๆ ดังนี้

1. ส่วนรับข้อมูลจากโมเด็ม

เมื่อมีการติดต่อสื่อสารข้อมูลระหว่างผู้ใช้และศูนย์คอมพิวเตอร์ ผู้ใช้จะโทรศัพท์ติดต่อเข้าไปที่ศูนย์ จากนั้นทางศูนย์ฯ จะทำการส่งข้อมูลมาให้ผู้ใช้โดยการมอดดูเลทสัญญาณข้อมูลให้เป็นสัญญาณเสียง แล้วส่งมาทางสายโทรศัพท์ เมื่อถึงปลายทางจะต้องทำการดีมอดดูเลทสัญญาณข้อมูลก่อนโดยผ่านทางโมเด็มของผู้ใช้ สัญญาณข้อมูลที่ได้จะเป็นแบบอนุกรม จากนั้นสัญญาณข้อมูลจะถูกส่งผ่านเข้าไปยังส่วนรับข้อมูลจากโมเด็ม เพื่อทำการแปลงสัญญาณข้อมูลเป็นแบบขนาน

2. ส่วนพักข้อมูลชั่วคราว

สัญญาณข้อมูลที่แปลงเป็นแบบขนานแล้ว ถูกนำไปเก็บไว้ในส่วนของที่พักข้อมูลชั่วคราว (Buffer Memory) เพื่อรอให้ส่วนสร้างสัญญาณวีดีโอนำข้อมูลดังกล่าวไปใช้งานต่อไป

3. ส่วนสร้างสัญญาณวีดีโอ

ในส่วนนี้มีไอซี 6845 CRTC เป็นตัวควบคุม ซึ่งรับคำสั่งจากซีพียู โดยที่ 6845 CRTC จะสร้างสัญญาณต่าง ๆ เช่น VSYNC,HSYNC และสัญญาณที่จำเป็นในการสร้างภาพ เป็นต้น เมื่อต้องการแสดงผล ส่วนสร้างสัญญาณวีดีโอจะค่อย ๆ นำข้อมูลจากหน่วยพักข้อมูลชั่วคราวไปสร้างเป็นตัวอักษร เพื่อแสดงผลบนจอโทรทัศน์

4. ส่วนควบคุมและประมวลผลข้อมูล (CPU)

ในการทำงานทั้งหมดของระบบ จะมีซีพียูเป็นตัวควบคุม และประมวลผลข้อมูลซึ่งซีพียูที่ใช้คือ Z80180 โดยที่ซีพียูดังกล่าวมีคุณสมบัติที่ดีกว่าซีพียู Z80 หลายประการซึ่งจะกล่าวถึงในรายละเอียดต่อไป

5. ส่วนคีย์บอร์ด

ในส่วนนี้เป็นส่วนของอุปกรณ์อินพุท เพื่อใช้ในการเลือกรายการข้อมูล การป้อนหมายเลขโทรศัพท์ และการใช้ฟังก์ชันคีย์ต่าง ๆ เช่น การพิมพ์ข้อมูล การเบรค การลบ การ ENTER เป็นต้น ซึ่งในการทำงานของคีย์แต่ละคีย์นั้นจะใช้หลักการของการอินเทอร์รัพท์ซีพียู

6. ส่วนพิมพ์ข้อมูล

ส่วนนี้เป็นส่วนที่ขยายขึ้นมา โดยการใช้พอร์ท Printer ของซีพียู Z80180 เมื่อผู้ใช้ต้องการพิมพ์ข้อมูลออกทางเครื่องพิมพ์ ก็ทำได้โดยการเลือกกดคีย์ในส่วนของคีย์บอร์ด จากนั้นข้อมูลจะถูกส่งออกไปยังเครื่องพิมพ์ตามที่ผู้ใช้ต้องการ

ประโยชน์ของโครงงาน

1. สามารถติดต่อทราบข้อมูลข่าวสารต่าง ๆ ได้ เพียงแต่มี โทรทัศน์ โทรศัพท์ และตัวแปลงสัญญาณ (Adaptor) โดยที่ไม่จำเป็นต้องมีคอมพิวเตอร์อยู่ที่บ้าน

2. สามารถเลือกข้อมูลได้ว่าต้องการข้อมูลชนิดใด

3. สามารถใช้บริการได้อย่างรวดเร็วเพียงแต่ติดต่อเข้าไปที่ศูนย์ข้อมูล

4. ข้อมูลสามารถแสดงผลได้ทั้งตัวอักษรและรูปภาพ

5. สามารถแสดงผลออกทางเครื่องพิมพ์ได้

6. สามารถนำเอาไปประยุกต์ใช้ในงานต่าง ๆ เช่น

- การสั่งซื้อสินค้าจากซุปเปอร์มาร์เกต
- การใช้บริการกับธนาคาร
- การสั่งซื้อหนังสือ อาหาร
- การแสดงราคาหุ้น ข่าวสาร บันเทิง พยากรณ์อากาศ
- การสั่งจองตั๋วเครื่องบิน รถไฟ ภาพยนตร์
- การเรียนหนังสือที่บ้าน
- สมุดหน้าเหลืองอิเล็กทรอนิกส์

ฯลฯ

วิธีดำเนินงาน

1. ศึกษาการทำงานของส่วนแสดงผล เช่น การสแกนจอภาพ, ลักษณะความถี่ต่าง ๆ ที่เกี่ยวข้อง, การทำให้เกิดภาพบนจอจากจุด (dot) ต่าง ๆ และตัวควบคุมการแสดงผล 6845

2. ศึกษาโครงสร้างและสถาปัตยกรรมของไมโครโปรเซสเซอร์ Z80180 ซึ่งเป็น CPU ที่ใช้ควบคุมระบบ ซึ่งภายในประกอบไปด้วย ตัวตั้งเวลา (Timer) , การสื่อสารข้อมูลแบบอนุกรม (Serial-Communication),การเข้าถึงหน่วยความจำโดยตรง (DMA),สัญญาณควบคุมโมเด็มต่าง ๆ , การอ้างหน่วยความจำได้ 1 Mbyte , การอินเทอร์รัพท์

3. ทำฮาร์ดแวร์ส่วนแสดงผล (Monitor)

4. เขียนโปรแกรมเพื่อเซตค่า 6845 CRTC ซึ่งประกอบไปด้วยค่าจำนวนตัวอักษรที่ต้องการแสดงต่อบรรทัด,จำนวนบรรทัดต่อหน้า, สัญญาณ HSYNC , VSYNC

5. ทำการทดสอบฮาร์ดแวร์และซอฟแวร์ของระบบแสดงผลทั้งหมด

6. ศึกษาระบบอินเทอร์รัพท์ของ Z80180 และทำฮาร์ดแวร์ส่วนคีย์บอร์ดโดยใช้ระบบอินเทอร์รัพท์เข้าช่วย เพื่อให้ระบบมีประสิทธิภาพมากขึ้น

7. เขียนลายวงจรพิมพ์ของวงจรส่วนแสดงผลและส่วนคีย์บอร์ด

8. ศึกษาระบบการติดต่อสื่อสารแบบอนุกรมของเครื่องคอมพิวเตอร์ และซอฟแวร์ที่ใช้ควบคุมโมเด็ม (AT-Command) และเขียนโปรแกรมเพื่อควบคุมโมเด็ม และโปรแกรมควบคุม Master ทั้งหมด

9. เขียนโปรแกรมควบคุมการทำงานของคีย์บอร์ดในแต่ละคีย์

10. เขียนซอฟแวร์ควบคุมโมเด็ม โดยใช้ Z80180 ควบคุมการทำงาน

11. ประกอบฮาร์ดแวร์ทุกส่วนลงบนแผ่นวงจรพิมพ์

12. รวมซอฟแวร์แต่ละส่วนเข้าด้วยกัน โดยแบ่งเป็น 2 ส่วนใหญ่ ๆ คือ

    12.1 โปรแกรมควบคุมส่วนตัวแม่ (Master)

    12.2 โปรแกรมควบคุมส่วนตัวลูก (Slave)

13. ทดสอบความสมบูรณ์ของการทำงานของระบบทั้งหมด

14. ประกอบระบบทั้งหมด

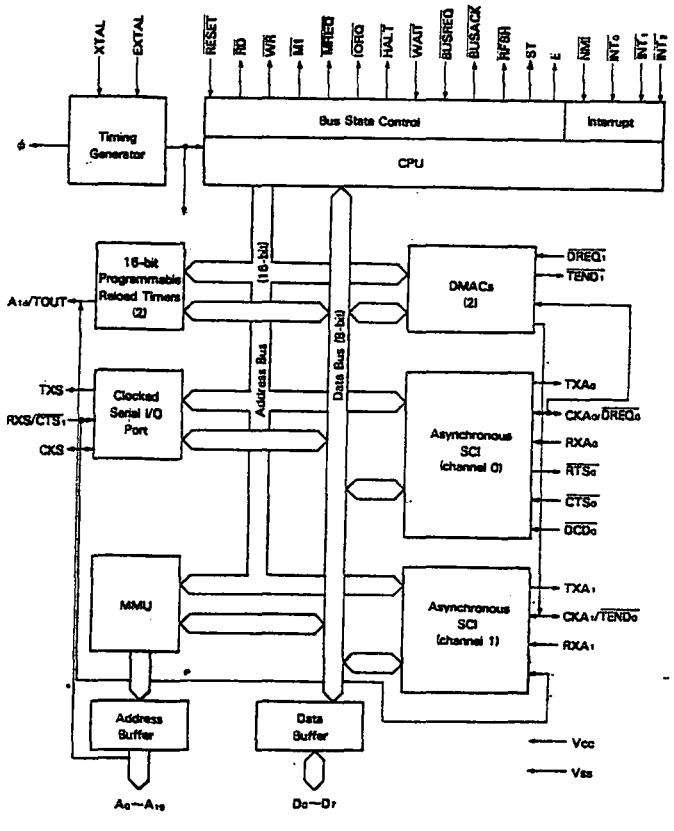
บทที่ 2

ระบบไมโครโปรเซสเซอร์ Z80180

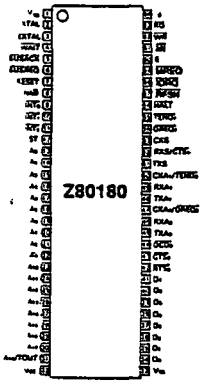
ลักษณะทั่วไป

Z80180 CPU เป็นซีพียูที่มีความสามารถสูง ได้รวบรวมชิพสำคัญอื่น ๆ ไว้ในซีพียูชิพเดียว ทำให้มีลักษณะคล้ายกับซีพียูที่ใช้ในงานควบคุมในจำพวกชิพเดี่ยว (Single Chip) แต่เนื่องจากชิพเดี่ยว (Single Chip) มีข้อดี คือ ระบบเล็ก ราคาถูก ข้อเสีย คือ การโปรแกรมควบคุมค่อนข้างยาก แต่สำหรับ Z80180 CPU นั้น ทางด้านโปรแกรมจะง่ายกว่าเพราะ Z80180 CPU นี้เป็น Supper Compat Z80 CPU กล่าวคือ ชุดคำสั่งทั้งหมดยังเป็น Z80 CPU และได้เพิ่มชุดคำสั่งขึ้นมาเพื่อเพิ่มความสะดวกในการใช้งานอีกด้วย

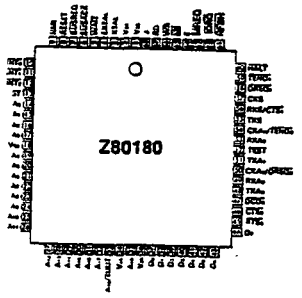
Z80180 CPU ประกอบด้วยไอซีที่เป็น CMOS,มีออสซิเลเตอร์ในตัว ทำงานถึงความถี่ 10 MHz ,ชิพ MMU สามารถอ้างหน่วยความจำได้ 1 Mbyte ,DMA 2 Channel ,พอร์ทสื่อสาร UART 2 Channel ,Clock Serial I/O , 16 Bit Timer Counter และเกี่ยวกับพอร์ทสื่อสาร สามารถทำ Multiprocessor Communication ได้ ซึ่งโครงสร้างของชิพนี้จะเป็นดังรูปที่ 2.1a และ 2.1b



รูปที่ 2.1a แสดงแผนภาพโครงสร้างภายในของชิพ Z80180 CPU



ก. 64 Pin DIP



ข. 68 Pin PLCC

รูปที่ 2.1b แสดงแผนภาพโครงสร้างภายนอกของชิพ Z80180 CPU

รายละเอียดและหน้าที่ที่สำคัญของสายสัญญาณต่าง ๆ มีดังนี้

- A0-A19 Address Bus 0-19:เป็นสายของบัสแอดเดรส จำนวน 20 สาย ซึ่งส่วนภายในของเอาท์พุทเป็นลอจิก 3 สถานะ (Tristate Output) และจะถูก Enable เพื่อเลือกว่าเวลาใดเป็นสัญญาณแอดเดรสของหน่วยความจำหรือของอุปกรณ์อินพุทเอาท์พุทใด ทั้งนี้เพราะสายของบัสแอดเดรส ยังทำหน้าที่เป็นตัวอ้างแอดเดรสสำหรับอุปกรณ์อินพุทเอาท์พุทอีกด้วย ซึ่งในระหว่างที่มีการรีเซ็ต สายของบัสแอดเดรสทั้งหมดจะอยู่ในสภาวะ High Impedance

- $\overline{\text{BUSAK}}$ Bus Acknowledge:เป็นสายสัญญาณเอาท์พุท Active Low จะมีการทำงานก็ต่อเมื่อ Z80180 CPU ตอบสนองต่อการขอบัสของ $\overline{\text{BUSRQ}}$ และจากผลดังกล่าวทำให้บัสข้อมูล บัสแอดเดรส และสัญญาณควบคุมบางเส้นเป็น High Impedance

- $\overline{\text{BUSRQ}}$ Bus Request :เป็นสายสัญญาณอินพุท Active Low เพื่อบอกซีพียูว่าขณะนี้ต้องการใช้บัส ซึ่งทำให้ซีพียูควบคุมบัสโดยใช้หลักการของลอจิก 3 สถานะ ในการทำให้บัสแอดเดรสและบัสข้อมูลแยกออกจากระบบในซีพียู เพื่อให้หน่วยความจำและอุปกรณ์อินพุทเอาท์พุทใช้บัสในการเคลื่อนย้ายข้อมูลระหว่างกัน ซึ่งสัญญาณ $\overline{\text{BUSRQ}}$ นี้มีความสำคัญสูงกว่า $\overline{\text{NMI}}$ โดยจะมีการตรวจสอบสัญญาณทุก ๆ การสิ้นสุดของ Machine Cycle

- CKA0,CKA1 Asynchronous Clock 0 และ 1 :เป็นสายสัญญาณ Clock แบบ 2 ทิศทาง คือ ใช้เป็นสายสัญญาณอินพุท หรือเอาท์พุทก็ได้

- CKS — Serial Clock: เป็นสายสัญญาณ Clock แบบ 2 ทิศทางของ CSI/O
- CLOCK — System Clock: เป็นสายสัญญาณเอาท์พุท Active High ซึ่งใช้เป็น Clock อ้างอิงสำหรับ MPU และ ระบบภายนอก โดยความถี่ของสายสัญญาณเอาท์พุทนี้จะเป็นครึ่งหนึ่งของความถี่คริสตอล หรือ ความถี่ของ Clock อินพุท เช่น คริสตอล 12 MHz ดังนั้น Z80180 CPU ทำงานที่ความถี่ 6 MHz
- $\overline{\text{CTS0}}$-$\overline{\text{CTS1}}$ — Clear to Send 0 และ 1 : เป็นสายสัญญาณอินพุท Active Low ใช้ในการควบคุมโมเด็ม
- D0-D7 — Data Bus: เป็นสายของบัสข้อมูลจำนวน 8 สาย ลักษณะของสายนี้เป็นลอจิก 3 สถานะ สองทิศทาง (Tristate input/output) เพื่อเลือกทิศทางการไหลของข้อมูลระหว่างซีพียูกับหน่วยความจำหรืออุปกรณ์อินพุทเอาท์พุท
- $\overline{\text{DCD0}}$ — Data Carrier Detect 0 : เป็นสายสัญญาณอินพุท Active Low ใช้ควบคุมในการติดต่อกับโมเด็มของ ASCI Channel 0
- $\overline{\text{DREQ0}}$-$\overline{\text{DREQ1}}$ — DMA Request 0 และ 1: เป็นสายสัญญาณอินพุท Active Low ใช้ในการขอ DMA และสายสัญญาณนี้จะโปรแกรมได้ว่าให้ตรวจสัญญาณที่ขอบหรือระดับได้
- E — Enable Clock: เป็นสายสัญญาณเอาท์พุท Active High ใช้ซิงค์การทำงานกับอุปกรณ์ภายนอกระหว่างการทำงานเกี่ยวกับบัส และใช้เชื่อมต่อกับอุปกรณ์ในตระกูล 68XX และ 80XX
- $\overline{\text{HALT}}$ — เป็นสายสัญญาณเอาท์พุท Active Low จะทำงานเมื่อซีพียูกระทำคำสั่ง HALT หรือ SLP
- $\overline{\text{INT0}}$ — Maskable Interrupt 0: เป็นสายสัญญาณอินพุท Active Low โดยสัญญาณดังกล่าวมาจากอุปกรณ์อินพุทเอาท์พุทที่จะอินเทอร์รัพท์ซีพียู ซึ่งมีการตรวจสอบสัญญาณนี้ทุก ๆ การสิ้นสุดของคำสั่ง
- $\overline{\text{INT1}}$, $\overline{\text{INT2}}$ — Maskable Interrupt 1 และ 2 : มีลักษณะเช่นเดียวกับ $\overline{\text{INT0}}$ แต่มีระดับความสำคัญรองลงมาตามลำดับ

- $\overline{\text{IORQ}}$ I/O Request :เป็นสายสัญญาณเอาท์พุท Active Low ซึ่งจะบอกว่าขณะนี้สัญญาณในบัสแอดเดรสจาก A0 - A7 มีค่าของอินพุทเอาท์พุทอยู่ ประโยชน์ของสัญญาณนี้ เพื่อตรวจจับ (Detect) รหัสแอดเดรสในการเขียนหรือการอ่านข้อมูลจากเพอริเฟอรัล
- $\overline{\text{M1}}$ Machine Cycle 1 :ลักษณะเป็นสัญญาณเอาท์พุท Active Low โดยส่งสัญญาณออกมาเพื่อบอกให้ทราบว่ากำลังอยู่ในสภาวะเฟตช์ (Fetch)
- $\overline{\text{NMI}}$ Non Maskable Interrupt :เป็นสายสัญญาณอินพุท Active Low สายสัญญาณนี้จะตอบรับการอินเทอร์รัพท์เสมอ โดยไม่สามารถหยุดด้วยซอฟแวร์ได้
- $\overline{\text{RD}}$ Read:เป็นสายสัญญาณเอาท์พุท Active Low เพื่อบอกให้ทราบว่าขณะนี้ซีพียูต้องการจะอ่านข้อมูลจากหน่วยความจำ หรืออุปกรณ์อินพุทเอาท์พุท
- $\overline{\text{RFSH}}$ Refresh :เป็นสายสัญญาณเอาท์พุท Active Low โดยจะส่งสัญญาณเพื่อบอกว่า ขณะนี้สายของแอดเดรสจะบรรจุข้อมูลแอดเดรสสำหรับการรีเฟรชหน่วยความจำชนิดไดนามิก
- $\overline{\text{RTS0}}$ Request to Send 0 :เป็นสายสัญญาณเอาท์พุท Active Low ใช้โปรแกรมสัญญาณควบคุมโมเด็มของ ASCI Channel 0
- RXA0,RXA1 Receive Data 0 และ 1 :เป็นสายสัญญาณอินพุท Active High ใช้ในการรับข้อมูลจากพอร์ทอนุกรมของ ASCI Channel
- RXS Clocked Serial Receive Data:เป็นสายสัญญาณอินพุท Active-High ใช้ในการรับข้อมูลของ CSIO Channel
- ST Status:เป็นสายสัญญาณเอาท์พุท Active High ใช้แสดงสถานะการทำงานของซีพียู โดยสัญญาณนี้จะถูกใช้ร่วมกับสัญญาณเอาท์พุท M1 และ HALT ดังตารางที่ 2.1

ตารางที่ 2.1 แสดงความสัมพันธ์ระหว่างสัญญาณ ST,$\overline{\text{HALT}}$,$\overline{\text{MI}}$

ST	$\overline{\text{HALT}}$	$\overline{\text{M1}}$	Operation
0	1	0	CPU operation (1st op-code fetch)
1	1	0	CPU operation (2nd op-code and 3rd op-code fetch)
1	1	1	CPU operation (MC except for op-code fetch)
0	x	1	DMA operation
0	0	0	HALT mode
1	0	1	SLEEP mode (including SYSTEM STOP mode)

Note: x - Don't care

MC - Machine cycle

- $\overline{\text{TEND0}}$-$\overline{\text{TEND1}}$ Transfer End 0 และ 1 เป็นสายสัญญาณเอาท์พุท Active Low ใช้แสดงถึงการสิ้นสุดของการย้ายบล็อคของการกระทำ DMA
- TOUT Timer Out เป็นสายสัญญาณเอาท์พุท Active High คือสัญญาณพัลส์จาก PRT Channel 1
- TXA0,TXA1 Transmit Data 0 และ 1 เป็นสายสัญญาณเอาท์พุท Active High ใช้ในการส่งข้อมูลอนุกรมจาก ASCI Channels
- TXS Clocked Serial Transmit Data เป็นสายสัญญาณเอาท์พุท ใช้ในการส่งข้อมูลอนุกรมจาก CSIO Channel โดยสัญญาณ Active High
- $\overline{\text{WAIT}}$ Wait เป็นสายสัญญาณอินพุท Active Low บอกให้ทราบว่า ขณะนี้หน่วยความจำ หรืออุปกรณ์อินพุทเอาท์พุทยังไม่พร้อมที่จะรับ หรือส่งผ่านข้อมูลคือเมื่อส่งสัญญาณนี้เข้าไป ซีพียูจะหยุดรอจนกว่าเลิกสัญญาณ $\overline{\text{WAIT}}$

ซึ่งสัญญาณนี้จะถูกตรวจสอบที่ขอบขาลงของ Clock ลูกที่ 2 ของทุก ๆ Machine Cycle

- $\overline{WR}$ Write เป็นสายสัญญาณเอาท์พุท Active Low บอกให้ทราบว่าขณะนี้ ซีพียูต้องการจะเขียนข้อมูลในหน่วยความจำหรือในอุปกรณ์อินพุทเอาท์พุท
- XTAL Crystal เป็นสายสัญญาณอินพุท Active High ใช้ต่อกับคริสตอล

รายละเอียดของสายสัญญาณที่ถูกมัลติเพล็กซ์

- A18/TOUT ระหว่างการรีเซ็ต สายสัญญาณนี้จะเป็น A18 แต่ถ้ามีการเลือกเซ็ตบิท TOC1 หรือ TOC0 ใน Timer Control Register(TCR) สายสัญญาณนี้จะเป็น TOUT
- CKA0/$\overline{DREQ0}$ ระหว่างการรีเซ็ต สายสัญญาณนี้จะเป็น CKA0 แต่ถ้า DM1 หรือ SM1 ใน DMA Mode Register (DMODE) ถูกเซ็ต สายสัญญาณนี้จะเป็น $\overline{DREQ0}$
- CKA1/$\overline{TEND0}$ ระหว่างการรีเซ็ต สายสัญญาณนี้จะเป็น CKA1 แต่ถ้าบิท CKA1D ใน ASCI Control Register Ch 1 (CNTLA1) ถูกเซ็ต สายสัญญาณนี้จะเป็น $\overline{TEND0}$
- RXS/$\overline{CTS1}$ ระหว่างการรีเซ็ต สายสัญญาณนี้จะเป็น RXS ถ้าบิท CTS1E ใน ASCI Status Register Ch 1 (STAT1) ถูกเซ็ต สายสัญญาณนี้จะเป็น $\overline{CTS1}$

การอินเทอร์รัพท์ (Interrupt)

มีด้วยกัน 12 อินเทอร์รัพท์ แบ่งเป็น

1. อินเทอร์รัพท์ภายนอก 4 อินเทอร์รัพท์ ได้แก่

- $\overline{\text{NMI}}$ (Non Maskable Interrupt)
- $\overline{\text{INT0}}$ (Maskable Interrupt Level 0)
- $\overline{\text{INT1}}$ (Maskable Interrupt Level 1)
- $\overline{\text{INT2}}$ (Maskable Interrupt Level 2)

2. อินเทอร์รัพท์ภายใน 8 อินเทอร์รัพท์ ได้แก่

- TRAP (Undefined Op-code Trap)
- Timer 0
- Timer 1
- DMA Channel 0
- DMA Channel 1
- Clocked Serial I/O Port
- Asynchronous SCI Channel 0
- Asynchronous SCI Channel 1

รีจิสเตอร์ (Register) และแฟลก (Flag) ที่ใช้ควบคุมการอินเทอร์รัพท์

1. Interrupt Vector High Register (I)

ใช้เป็น Vector Table Byte สูง ของการอินเทอร์รัพท์ภายนอก $\overline{\text{INT1}}$, $\overline{\text{INT2}}$, $\overline{\text{INT0}}$ (Mode 2) และอินเทอร์รัพท์ภายในทั้งหมดยกเว้น "TRAP" ซึ่งในการกำหนดค่าตารางเวกเตอร์ของการอินเทอร์รัพท์นี้ จะเป็นตัวชี้ตำแหน่งเริ่มต้นของโปรแกรมย่อยที่ต้องไปกระทำ ซึ่งสามารถเขียนหรืออ่านรีจิสเตอร์ I ได้โดยใช้คำสั่ง LD A,I และ LD I,A และในระหว่างการรีเซ็ต ค่ารีจิสเตอร์ I จะเป็น 00H

2. Interrupt Vector Low Register (IL : I/O Address 33H)

ใช้เป็น Vector Table Byte ต่ำ ของการอินเทอร์รัพท์ภายนอก $\overline{\text{INT1}}$, $\overline{\text{INT2}}$ และอินเทอร์รัพท์ภายในทั้งหมดยกเว้น "TRAP" โดย 3 บิทสูงของ IL สามารถโปรแกรมได้ แต่ 5 บิทหลังจะถูกระบุไว้ดังตารางที่ 2.2 และในระหว่างการรีเซ็ต ค่ารีจิสเตอร์ IL จะเป็น 00H

ตารางที่ 2.2 แสดงการโปรแกรมรีจิสเตอร์ IL สำหรับ Interrupt Source ต่าง ๆ

Interrupt Source	Priority	IL			Fixed Code				
		b7	b6	b5	b4	b3	b2	b1	b0
$\overline{INT1}$	Highest	.	.	.	0	0	0	0	0
$\overline{INT2}$		.	.	.	0	0	0	1	0
PRT Channel 0		.	.	.	0	0	1	0	0
PRT Channel 1		.	.	.	0	0	1	1	0
DMA Channel 0		.	.	.	0	1	0	0	0
DMA Channel 1		.	.	.	0	1	0	1	0
CSI/O		.	.	.	0	1	1	0	0
ASCI Channel 0		.	.	.	0	1	1	1	0
ASCI Channel 1	Lowest	.	.	.	1	0	0	0	0

"." หมายถึง สามารถโปรแกรมได้

ดังนั้นการอินเทอร์รัพท์ $\overline{INT1}$ และ $\overline{INT2}$ จะเป็นการนำค่าในรีจิสเตอร์ I และ IL หรือนำค่าในรีจิสเตอร์ I และ ข้อมูลจากอุปกรณ์ I/O ที่ขออินเทอร์รัพท์ส่งมา (สำหรับ $\overline{INT0}$ Mode 2) มาประกอบกันเป็นค่าแอดเดรสที่เก็บข้อมูลที่จะกระโดดไปทำโปรแกรมย่อยนั้น ๆ เช่น I = 10H และ IL = 40H และในแอดเดรส 1040H มีข้อมูล 00H, 60H ตามลำดับ เมื่อเกิดการอินเทอร์รัพท์ขึ้น ก็จะกระโดดไปทำโปรแกรมที่ตำแหน่ง 6000H นั่นเอง

3. INT/TRAP Control Register (ITC : I/O Address 34H)

BIT	7	6	5	4	3	2	1	0
	TRAP	UFO	-	-	-	ITE2	ITE1	ITE0

รูปที่ 2.2 แสดงบิทต่างๆใน INT/TRAP Control Register

ITE2,1,0 (บิท 2,1,0) Interrupt Enable 2,1,0 ใช้สำหรับ Enable และ Disable การอินเทอร์รัพท์ภายนอก $\overline{INT2}$,$\overline{INT1}$,$\overline{INT0}$ ตามลำดับ ถ้ารีเซ็ตบิทให้เป็น 0 ก็ Disable ถ้าเซ็ตบิทนี้ให้เป็น 1 ก็ Enable แต่ไม่ทำให้เกิดการอินเทอร์รัพท์ขึ้นทันทีจนกว่าจะทำคำสั่ง EI ดังนั้น $\overline{INT0}$ จึงต่างกับ Z80 ตรงที่มีส่วนนี้ และในขณะที่มีการรีเซ็ตระบบเกิดขึ้น บิท ITE0 จะถูกเซ็ตเป็น 1 โดยอัตโนมัติเพื่อให้ขึ้นกับคำสั่ง EI หรือ DI อย่างเดียว แต่ ITE1 และ ITE2 จะเป็น 0

UFO (บิท 6) Undefined Fetch Object เมื่อ TRAP เกิดขึ้น UFO จะให้ค่าของตำแหน่งที่ผิดในคำสั่งนั้นไว้ใน Stack เนื่องจาก TRAP อาจเกิดขึ้นจาก Op-code 2 หรือ 3 ไบท์ UFO จะปรับค่า PC (Program Counter) ให้ คือ ถ้าเป็นคำสั่ง Op-code 2 ไบท์ จะทำให้ PC (Program Counter) ของคำสั่งถัดไปจากคำสั่งที่ไม่ใช่ของ Z80180 CPU ถูกลดลง 1 แต่ถ้า UFO = 1 คำสั่งที่ผิดจะมี Op-code 3 ไบท์ และ PC (Program Counter) จะถูกลดลง 2 ตำแหน่ง และค่า PC (Program Counter) นี้จะถูกเก็บไว้ใน Stack เช่น

Address		Op-code
2000		ED 99
2002	<-	PC ชี้คำสั่งถัดไป

เมื่อซีพียูทำงานมาพบข้อมูลที่ตำแหน่ง 2000H ก็จะเกิดอินเทอร์รัพท์ TRAP ขึ้น และรู้ด้วยว่าเป็นคำสั่ง 2 ไบท์ และ PC ก็ชี้คำสั่งถัดไปคือ แอดเดรส 2002H แต่แฟลก UFO จะถูกทำให้เป็น 0 เพื่อปรับค่า PC นั้นด้วยการลดลง 1 จะได้ค่าแอดเดรสเป็น 2001H ซึ่งก็คือตำแหน่งข้อมูลที่ผิดนั่นเอง สำหรับบิท UFO นี้สามารถอ่านได้อย่างเดียวเท่านั้น

TRAP (บิท 7) ถูกเซ็ตเป็น 1 เมื่อมีการทำคำสั่งที่ไม่มีใน Z80180 CPU ซึ่ง TRAP สามารถรีเซ็ตภายใต้โปรแกรมควบคุมได้ แต่ไม่สามารถเขียน 1 เข้าไปได้และระหว่างการรีเซ็ตระบบ บิทนี้จะเป็น 0

4.Interrupt Enable Flag 1,2 (IEF1,IEF2)

แฟลก IEF1 ใช้ควบคุมการ Enable และ Disable ของการอินเทอร์รัพท์ทั้งหมด ยกเว้นการอินเทอร์รัพท์แบบ $\overline{\text{NMI}}$ และ TRAP

ถ้าแฟลก์ IEF1 = 0 การอินเทอร์รัพท์แบบมาสเคเบิลทั้งหมดจะถูก Disable ซึ่งสามารถเซ็ตให้ IEF1 = 1 ได้โดยใช้คำสั่ง EI (Enable Interrupt) และ IEF1= 0 ได้โดยใช้คำสั่ง DI (Disable Interrupt)

สำหรับแฟลก IEF2 นั้นจะใช้เมื่อมีการอินเทอร์รัพท์แบบ $\overline{\text{NMI}}$ กล่าวคือ ระหว่างที่มีการอินเทอร์รัพท์แบบ $\overline{\text{NMI}}$ นั้น ค่าสภาวะของการรับอินเทอร์รัพท์ครั้งก่อนที่เก็บอยู่ใน IEF1 จะนำไปเก็บไว้ใน IEF2 จากนั้นค่าใน IEF1 จะถูกรีเซ็ต ทำให้ไม่มีการรับอินเทอร์รัพท์อื่น ๆ เกิดซ้อนขึ้นมาได้ และหลังจากการกระทำโปรแกรมย่อยเสร็จสิ้นลง ซึ่งผ่านการทำคำสั่ง RETN ซึ่งมีผลให้เกิดการคืนค่าสภาวะของการรับอินเทอร์รัพท์เดิม (ก่อนการเกิด $\overline{\text{NMI}}$) ที่เก็บใน IEF2 กลับสู่ IEF1

ตารางที่ 2.3 แสดงค่าสภาวะของแฟลก IEF1 และ IEF2

CPU Operation	IEF1	IEF2	คำอธิบาย
RESET	0	0	ไม่มีการรับอินเทอร์รัพท์ยกเว้น $\overline{\text{NMI}}$ และ TRAP
$\overline{\text{NMI}}$	0	IEF1	เก็บค่าใน IEF1 ไว้ใน IEF2
RETN	IEF2	-	คืนค่าใน IEF2 กับสู่ IEF1
Interrupt ยกเว้น	0	0	ไม่มีการรับอินเทอร์รัพท์อื่น ๆ ซ้อน ยกเว้น $\overline{\text{NMI}}$
$\overline{\text{NMI}}$ และ TRAP			และ TRAP
RETI	-	-	-
TRAP	-	-	-
EI	1	1	-
DI	0	0	-
LD A,I	-	-	ย้ายค่าใน IEF2 ไปที่แฟลก P/V
LD A,R	-	-	ย้ายค่าใน IEF2 ไปที่แฟลก P/V

คำสั่งเพิ่มเติม 12 คำสั่ง

SLP - Sleep เมื่อใช้คำสั่งนี้ซีพียูจะหยุดทำงานบางอย่างทำให้ใช้กำลังงานต่ำ

MLT - Multiply ใช้สำหรับคูณเลข 8 บิท(ไม่มีเครื่องหมาย) 2 จำนวน ผลลัพธ์เป็นเลข 16 บิท โดยรีจิสเตอร์ที่ใช้คูณอาจจะเป็น BC, DE, HL, หรือ SP ซึ่งในทุก ๆ กรณีนั้นผลลัพธ์ที่ได้จะเก็บไว้ในรีจิสเตอร์คู่นั้น

OTIM,OTIMR,OTDM,OTDMR - Block I/O เป็นคำสั่งย้ายข้อมูลเป็นบล็อคออกไปที่พอร์ทเอาท์พุท (แอดเดรสพอร์ทต่ำ A0-A7 เท่านั้น) คือจะทำการย้ายข้อมูลออกไปเป็นบล็อคโดยที่หมายเลขพอร์ทเพิ่มขึ้นหรือลดลงตามจำนวนข้อมูล ซึ่งใช้รีจิสเตอร์ HL เป็นตัวชี้ข้อมูลที่จะย้ายออกไป และรีจิสเตอร์ C เป็นหมายเลขพอร์ท ในคำสั่ง OTIM และ OTDM ก็คือการเพิ่มค่า HL ที่ชี้ขึ้นไป 1 หรือลดไป 1 และเพิ่มหรือลดหมายเลขพอร์ทตามด้วยเช่นกัน และเมื่อมีการย้ายข้อมูลออกไปที่พอร์ทเอาท์พุท 1 ครั้ง จะมีการลดค่าตัวนับลง 1 ด้วยซึ่งใช้รีจิสเตอร์ B เป็นตัวนับในการส่งข้อมูล ส่วนคำสั่ง OTIMR และ OTDMR นั้นจะมีลักษณะเช่นเดียวกับคำสั่ง OTIM และ OTDM เพียงแต่จะทำการส่งข้อมูลเพิ่มขึ้นหรือลดลง และหมายเลขพอร์ทเพิ่มขึ้นหรือลดลงตามค่ารีจิสเตอร์ B จนกระทั่ง B = 0

TSTIO m - Test I/O Port ใช้สำหรับทดสอบพอร์ทอินพุทเอาท์พุท คือ จะทำการอ่านค่าพอร์ทที่กำหนดโดยรีจิสเตอร์ C เข้ามาแล้วทำการ AND กับข้อมูล 8 บิทที่ต้องการ โดยที่ค่าข้อมูลที่อินพุทเข้ามานั้นไม่เปลี่ยนแปลงแต่จะให้ผลที่แฟลก และพอร์ทที่อินพุทเข้ามาเป็นค่าเฉพาะแอดเดรสต่ำ A0-A7 เท่านั้น ซึ่งสามารถเปรียบเทียบเป็นโปรแกรมได้ดังนี้

Z80 CPU			Z80180 CPU	
XOR	A	---------->	LD	C,Number Port
IN	A,(Port)		TSTIO	70H
LD	B,A		JP	Z,OK
LD	A,70H			
AND	B			
JP	Z,OK			

TST g - Test Register ใช้สำหรับทดสอบค่าในรีจิสเตอร์ โดยค่าที่กำหนดในรีจิสเตอร์จะ AND กับค่าในแอคคูมูเลเตอร์ (A) ซึ่งจะทำให้มีผลต่อแฟลกตามคำสั่ง AND แต่ค่าในแอคคูมูเลเตอร์ (A) และรีจิสเตอร์ไม่เปลี่ยนแปลง เช่น ตัวอย่าง :

Z80 CPU		Z80180 CPU
LD A,7	---------->	LD A,7
LD C,A		TST B
AND B		JR Z,OK
LD A,C		
JR Z,OK		

TST m - Test Immediate เช่นเดียวกับ TST g เพียงแต่เปลี่ยนจากรีจิสเตอร์เป็นข้อมูลโดยตรงที่นำไป AND กับค่าในแอคคูมูเลเตอร์ (A)

TST (HL) - Test Memory คือจะนำค่าในหน่วยความจำที่ถูกชี้โดย HL มา AND กับค่าในแอคคูมูเลเตอร์ (A) โดยค่าทั้งสองไม่เปลี่ยนแปลงแต่ให้ผลการกระทำที่แฟลก

INO g,(m) - Input,Immediate I/O address คือการอินพุทค่าจากพอร์ท 8 บิท (A0-A7) มายังรีจิสเตอร์ใด ๆ ก็ได้ เช่น A, BC, DE, HL

OUTO (m),g - Output,Immediate I/O address คือ การเอาท์พุทค่าจากรีจิสเตอร์ใด ๆ ไปยังพอร์ท 8 บิท (A0-A7) ซึ่งรีจิสเตอร์ที่ใช้ได้แก่ A, BC, DE, HL

รีจิสเตอร์อินพุทเอาท์พุทภายใน (Internal I/O Registers)

มีด้วยกัน 64 I/O แอดเดรส โดยที่รีจิสเตอร์ดังกล่าวนี้จะเข้าถึงโมดูลอินพุทเอาท์พุทภายใน (ASCI, CSI/O, PRT) และฟังก์ชันควบคุม (DMAC, DRAM Refresh, Interrupt, WAIT State generator,MMU และ I/O relocation)

ตารางที่ 2.4 แสดง I/O Address Map

	Register	Mnemonic	Address	
ASCI	ASCI Control Register A Ch 0	CNTLA0	XX000000B	00H
	ASCI Control Register A Ch 1	CNTLA1	XX000001B	01H
	ASCI Control Register B Ch 0	CNTLB0	XX000010B	02H
	ASCI Control Register B Ch 1	CNTLB1	XX000011B	03H
	ASCI Status Register Ch 0	STAT0	XX000100B	04H

ตารางที่ 2.4 แสดง I/O Address Map (ต่อ)

	Register	Mnemonic	Address	
ASCI	ASCI Status Register Ch 1	STAT1	XX000101B	05H
	ASCI Transmit Data Register Ch 0	TDR0	XX000110B	06H
	ASCI Transmit Data Register Ch 1	TDR1	XX000111B	07H
	ASCI Receive Data Register Ch 0	RDR0	XX001000B	08H
	ASCI Receive Data Register Ch 1	RDR1	XX001001B	09H
CSI/O	CSI/O Control Register	CNTR	XX001010B	0AH
	CSI/O Transmit/Receive Data Register	TRDR	XX001011B	0BH
Timer	Timer Data Register Ch 0L	TMDR0L	XX001100B	0CH
	Timer Data Register Ch 0H	TMDR0H	XX001101B	0DH
	Reload Register Ch 0L	RLDR0L	XX001110B	0EH
	Reload Register Ch 0H	RLDR0H	XX001111B	0FH
	Timer Control Register	TCR	XX010000B	10H
	Reserved		XX010001B	11H
			⟆	⟆
			XX010011B	13H
	Timer Data Register Ch 1L	TMDR1L	XX010100B	14H
	Timer Data Register Ch 1H	TMDR1H	XX010101B	15H
	Reload Register Ch 1L	RLDR1L	XX010110B	16H
	Reload Register Ch 1H	RLDR1H	XX010111B	17H

ตารางที่ 2.4 แสดง I/O Address Map (ต่อ)

	Register	Mnemonic	Address	
Others	Free Running Counter	FRC	XX011000B	18H
	Reserved		XX011001B	19H
			⟨	⟨
			XX011111B	1FH
DMA	DMA Source Address Register Ch 0L	SAR0L	XX100000B	20H
	DMA Source Address Register Ch 0H	SAR0H	XX100001B	21H
	DMA Source Address Register Ch 0B	SAR0B	XX100010B	22H
	DMA Destination Add. Register Ch 0L	DAR0L	XX100011B	23H
	DMA Destination Add. Register Ch 0H	DAR0H	XX100100B	24H
	DMA Destination Add. Register Ch 0BH	DAR0B	XX100101B	25H
	DMA Byte Count Register Ch 0L	BCR0L	XX100110B	26H
	DMA Byte Count Register Ch 0H	BCR0H	XX100111B	27H
	DMA Memory Address Register Ch 1L	MAR1L	XX101000B	28H
	DMA Memory Address Register Ch 1H	MAR1H	XX101001B	29H
	DMA Memory Address Register Ch 1B	MAR1B	XX101010B	2AH
	DMA I/O Address Register Ch 1L	IAR1L	XX101011B	2BH
	DMA I/O Address Register Ch 1H	IAR1H	XX101100B	2CH
	Reserved		XX101101B	2DH
	DMA Byte Count Register Ch 1L	BCR1L	XX101110B	2EH
	DMA Byte Count Register Ch 1H	BCR1H	XX101111B	2FH
	DMA Status Register	DSTAT	XX110000B	30H
	DMA Mode Register	DMODE	XX110001B	31H
	DMA/WAIT Control Register	DCNTL	XX110010B	32H

ตารางที่ 2.4 แสดง I/O Address Map (ต่อ)

	Register	Mnemonic	Address	
INT	IL Register (Interrupt Vector Low)	IL	XX110011B	33H
	INT/TRAP Control Register	ITC	XX110100B	34H
	Reserved		XX110101B	35H
Refresh	Refresh Control Register	RCR	XX110110B	36H
	Reserved		XX110111B	37H
MMU	MMU Common Base Register	CBR	XX111000B	38H
	MMU Bank Base Register	BBR	XX111001B	39H
	MMU Common/Bank Area Register	CBAR	XX111010B	3AH
I/O	Reserved		XX111011B ʃ XX111101B	3BH ʃ 3DH
	Operation Mode Control Register	OMCR	XX111110B	3EH
	I/O Control Register	ICR	XX111111B	3FH

จากแผนผัง (Map) ของอินพุทเอาท์พุทภายใน จะเห็นว่าโปรแกรม Z80 เดิมที่เรามีอยู่อาจจะมีการส่งพอร์ทซ้ำกับอินพุทเอาท์พุทภายในได้ ทำให้โปรแกรมเดิมทำงานไม่ได้ สามารถแก้ไขได้ โดยการโปรแกรมย้ายแผนผัง (Map) อินพุทเอาท์พุทภายใน โดยการควบคุมบิทในรีจิสเตอร์อินพุทเอาท์พุท ICR แอดเดรส 3FH ซึ่งสามารถย้ายไปที่ใดก็ได้ภายใน 256 ตำแหน่ง

I/O Control Register (ICR : I/O Address 3FH)

บิท	7	6	5	4	3	2	1	0
	IOA7	IOA6	IOSTP	–	–	–	–	–
	R/W	R/W	R/W					

รูปที่ 2.3 แสดงบิทต่างๆใน I/O Control Register

IOA7,6 : I/O Address Relocation (บิทที่ 7 และ 6)

บิท IOA7 และ IOA6 จะเป็นตัวกำหนดตำแหน่งอินพุทเอาท์พุทภายในตำแหน่งใหม่ที่จะย้ายไป ซึ่งสามารถโปรแกรมได้ดังนี้

ตารางที่ 2.5 แสดงการโปรแกรมค่าในบิท IOA7 และ IOA6

IOA7	IOA6	ช่วงแอดเดรส I/O
0	0	0000H-003FH
0	1	0040H-007FH
1	0	0080H-00BFH
1	1	00C0H-00FFH

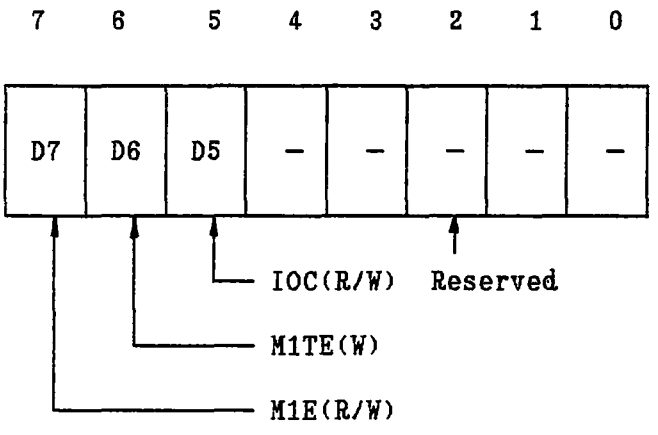
<u>ข้อสังเกต</u> สำหรับ 8 บิทสูงของแอดเดรสอินพุทเอาท์พุทภายใน 16 บิท จะมีค่าเป็น "0" เสมอ และระหว่างการรีเซ็ต บิท IOA7 และ IOA6 จะถูกเคลียร์เป็น "0"

IOSTP : IOSTOP Mode (บิท 5)

เมื่อบิทนี้เป็น "1" จะทำให้อินพุทเอาท์พุทภายในหยุดทำงาน และระหว่างการรีเซ็ตบิทนี้จะถูกเคลียร์เป็น "0"

Operation Mode

Z80180 CPU สามารถกำหนดการทำงานให้เหมือนกับ Z64180 ได้ โดยการเซ็ตค่าบิทใน **Operation Mode Control Register (OMCR:I/O แอดเดรส 3EH)**

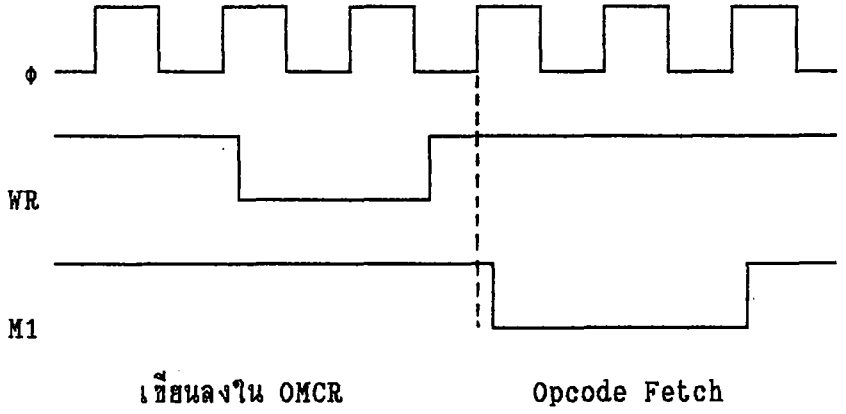


รูปที่ 2.4 แสดงบิทต่างๆใน Operation Mode Control Register

M1E ($\overline{M1}$ Enable) ในระหว่างการรีเซ็ต บิทนี้จะเป็น "1" เมื่อ M1E="1" จะให้สัญญาณ $\overline{M1}$ ออกมาเป็น Low เมื่อมีไซเคิลของการ Fetch Opcode และเนื่องจากการทำคำสั่ง RETI ของ Z80180 CPU จะถูกกระทำ 2 ครั้งใน 1 คำสั่ง จึงทำให้เกิด $\overline{M1}$ ขึ้น 2 ครั้งด้วย ซึ่งทำให้อาจเกิดมีการอินเทอร์รัพท์เข้ามาได้ เมื่อยังไม่หมดคำสั่ง ด้วยเหตุนี้บิท M1E จะถูกเซ็ตเป็น "0" สำหรับ Z80180 CPU เพื่อให้ $\overline{M1}$ ถูกทำงานปกติ คือ เมื่อมีคำสั่ง RETI จะมี $\overline{M1}$ เกิดขึ้นเพียงครั้งเดียว

$\overline{M1TE}$ (M1 Temporary Enable) บิทนี้จะใช้เมื่อมีการต่ออินเทอร์เฟสกับ Z80PIO ในระหว่างการรีเซ็ต บิทนี้จะเป็น "1"

เมื่อ $\overline{M1TE}$ = "0" จะมีสัญญาณ $\overline{M1}$ เกิดขึ้นที่ไซเคิลถัดไปของการ Fetch Opcode โดยไม่ต้องคำนึงถึงการโปรแกรมในบิท M1E ดังรูป 2.5

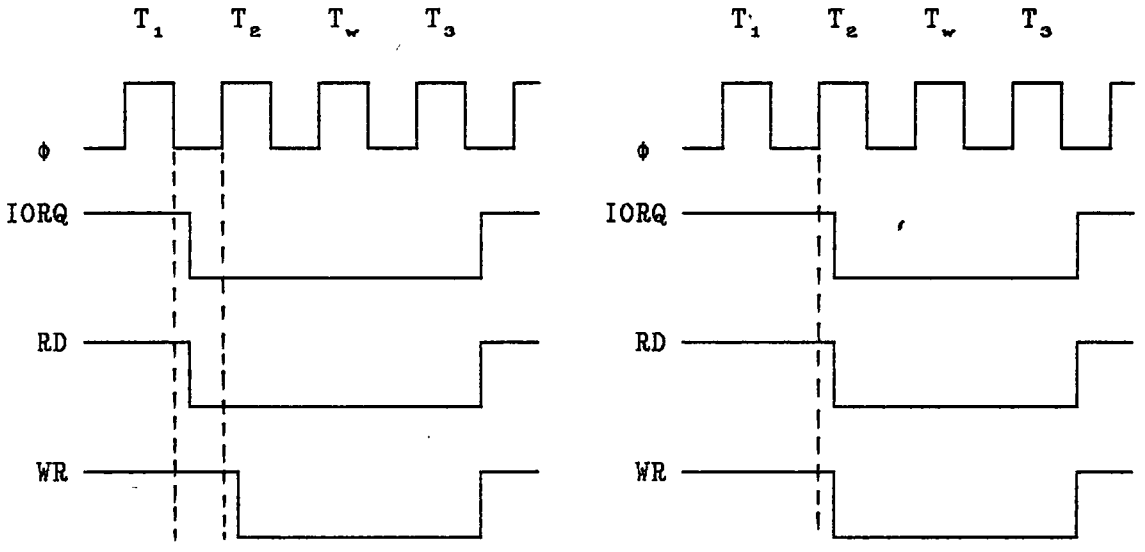


รูปที่ 2.5 แสดงการเกิดสัญญาณ $\overline{M1}$ ที่ไซเคิลถัดไปของการ Fetch Opcode

เมื่อ $\overline{M1TE}$ = "1" การกระทำของสัญญาณ $\overline{M1}$ จะไม่มีการเปลี่ยนแปลง และ M1E จะเป็นตัวควบคุมฟังก์ชันของมัน

$\overline{IOC}$ บิทนี้ใช้ควบคุม Timing ของสัญญาณ $\overline{IORQ}$ และ $\overline{RD}$ เมื่อรีเซ็ตบิทนี้จะเป็น"1"

เมื่อ $\overline{IOC}$ = "1" Timing ของสัญญาณ $\overline{IORQ}$ และ $\overline{RD}$ จะเหมือนกับ Z64180 คือ $\overline{IORQ}$ และ $\overline{RD}$ จะแอคทีฟที่ขอบขาลงของ T1 แต่ถ้า $\overline{IOC}$ = "0" Timing ของสัญญาณ $\overline{IORQ}$ และ $\overline{RD}$ จะตรงกับ Z80 กล่าวคือ $\overline{IORQ}$ และ $\overline{RD}$ แอคทีฟที่ขอบขาขึ้นของ T2 เพื่อให้ใช้อุปกรณ์สนับสนุนของ Z80 ได้ ซึ่งแสดงการเปรียบเทียบได้ดังรูป 2.6



ก) เมื่อ $\overline{IOC}$=0 ข) เมื่อ$\overline{IOC}$=1

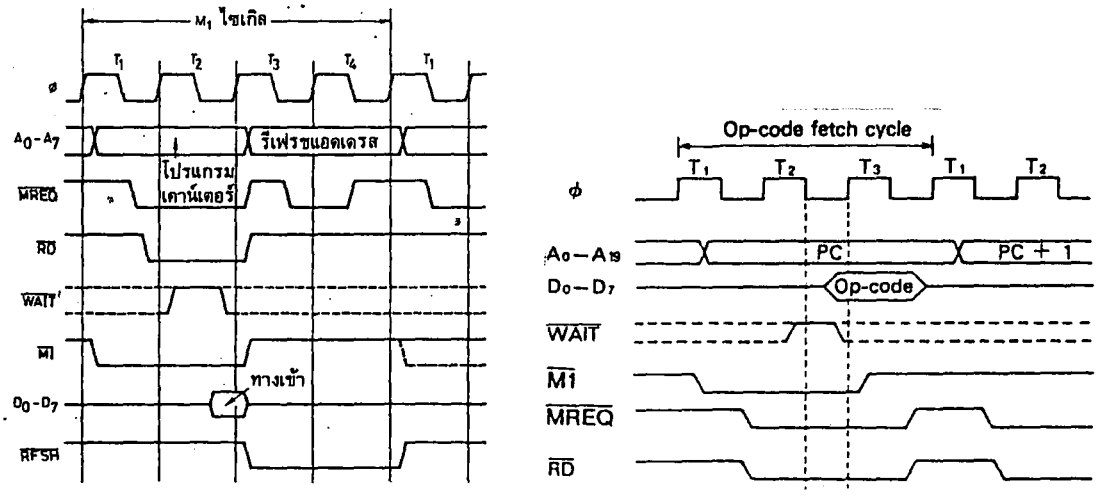
รูปที่ 2.6 แสดง I/O Read Write Cycle

<u>หมายเหตุ</u> ผู้ใช้จะต้องโปรแกรมค่าใน OMCR ก่อนที่จะมีการกระทำคำสั่ง I/O คำสั่งแรก ซึ่งสำหรับโปรแกรมที่ใช้ในโครงงานพิเศษนี้จะโปรแกรมค่าใน OMCR เป็น 00H

<u>Timing</u>

ในการทำงานของ Z80180 CPU นั้นมีความคล้ายคลึงกับ Z80 CPU กล่าวคือ มีการทำงานเป็นไซเคิล โดยสัญญาณที่ปรากฏออกมาเป็นจังหวะ และขึ้นกับสัญญาณนาฬิกา ปกติแล้วจะเรียกจังหวะวนรอบของการทำงานในหนึ่งรอบว่า ไซเคิล (Cycle) โดยไซเคิลของการทำงานหนึ่งคำสั่งประกอบด้วยหลาย ๆ แมชชีนไซเคิล ซึ่งเรียกแต่ละแมชชีนไซเคิลว่า $M_1, M_2, \ldots$ คำสั่งของ Z80180 CPU แต่ละคำสั่งจะใช้จำนวนแมชชีนไซเคิลไม่เท่ากัน บางคำสั่งใช้เพียงแมชชีนไซเคิลเดียว บางคำสั่งใช้หลายแมชชีนไซเคิล ซึ่งพบว่าในการทำคำสั่ง 1 แมชชีนไซเคิลของ Z80180 CPU นั้นจะใช้เวลาน้อยกว่า Z80 CPU อยู่ 1T state คือใช้เวลาใน 1 แมชชีนไซเคิลเพียง 3T state ในขณะที่ Z80 CPU ใช้เวลา 4T state ดังนั้นในขณะที่ให้ Z80180

CPU ทำงานที่ความถี่เดียวกันกับ Z80 CPU พบว่า Z80180 CPU ให้ความเร็วกว่า Z80 CPU ถึง 25% แต่ในขณะเดียวกัน Z80180 CPU ยังสามารถต่อกับสัญญาณนาฬิกาที่ความถี่สูงกว่า Z80 CPU จึงทำให้ความเร็วในการทำงานของ Z80180 CPU ดีกว่ามาก ซึ่งสามารถเปรียบเทียบ T state ของ Z80 CPU กับ Z80180 CPU ได้ดังรูป 2.7

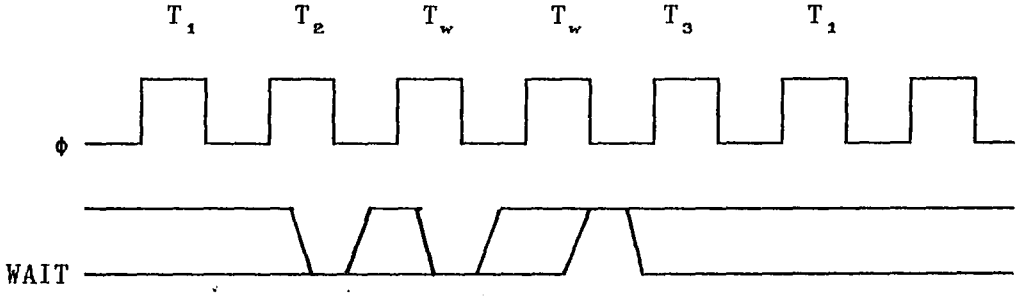


ก. Z80 CPU ข. Z80180 CPU

รูปที่ 2.7 แสดงการเปรียบเทียบ T State ระหว่าง Z80 CPU กับ Z80180 CPU

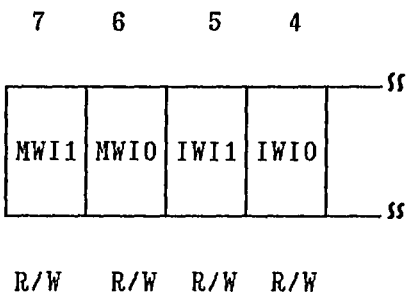
WAIT State Generator

ซีพียู Z80180 CPU นั้นจะทำงานด้วยความถี่ที่สูงขึ้น จึงอาจทำให้หน่วยความจำหรืออุปกรณ์อินพุทเอาท์พุททำงานไม่ทัน จึงต้องมีสัญญาณมาเป็นตัวช่วยกำหนดความพร้อมระหว่างซีพียูกับอุปกรณ์ภายนอก นั่นคือสัญญาณ $\overline{WAIT}$ ซึ่ง Z80 CPU นั้นจะต้องให้อุปกรณ์ภายนอกส่งสัญญาณนี้มาให้ แต่สำหรับ Z80180 CPU สามารถโปรแกรมจำนวน WAIT State เพื่อเพิ่มเข้าไปในขณะที่ซีพียูปฏิบัติคำสั่งหรือทำ DMA ได้



รูปที่ 2.8 $\overline{WAIT}$ Timing

การโปรแกรมจะใช้ 4 บิทของ DMA/WAIT Control Register (DCNTL : I/O Address 32H)



รูปที่ 2.9 แสดง Memory and I/O Wait State Insertion

MWI1,MWI0 - Memory Wait Insertion บิท 7 และ 6

บิททั้งสองนี้ใช้ในการเพิ่มจำนวน WAIT State จาก 0-3 Tw ของการเข้าถึงหน่วยความจำ (รวม Memory mapped I/O) โดยการโปรแกรมบิท MWI1,MWI0 ดังตารางที่ 2.6

ตารางที่ 2.6 แสดง Memory Wait States

MWI1	MWI0	จำนวนของ Wait State
0	0	0
0	1	1
1	0	2
1	1	3

IWI1,IWI0 - I/O Wait Insertion บิท 5 และ 4

สำหรับไซเคิลซีพียูและดีเอ็มเอ ซึ่งมีการเข้าถึงอินพุทเอาท์พุทภายนอก (และไซเคิล Interrupt Acknowledge) จะมีการเพิ่ม WAIT State(Tw) 1-6 Tw ได้โดยการโปรแกรมค่าในบิท IWI1,IWI0 ดังแสดงในตารางที่ 2.7

ตารางที่ 2.7 แสดง Wait State Insertion

IWI1	IWI0	I/O ภายนอก	$\overline{INT0}$
0	0	1	2
0	1	2	4
1	0	3	5
1	1	4	6

ในระหว่างที่รีเซ็ต บิทควบคุม WAIT State ทั้ง 4 บิทจะเป็น 1 ทั้งหมด คืออยู่ในโหมดของ WAIT State สูงสุด

ตัวอย่าง ต้องการเพิ่ม WAIT State ในการเข้าถึงหน่วยความจำ 2 Tw จะโปรแกรมได้ดังนี้

```
IN0  A,(32H)      ;IN ค่าในรีจิสเตอร์ DMA/WAIT
AND  10111111B    ;ใส่ค่าเฉพาะบิท 7 และ 6 เท่านั้น
OUT0 (32H),A      ;ที่ใช้ IN และ AND เพราะว่ารีจิสเตอร์มีการกำหนด
                  ;เกี่ยวกับ DMA ดังนั้นเราจึงใส่เฉพาะบิทที่ต้องการ
                  ;โปรแกรม
```

Dynamic RAM Refresh Control

ในการใช้หน่วยความจำแบบ DRAM นี้ จำเป็นจะต้องมีการรีเฟรชข้อมูลอยู่ตลอดเวลา การรีเฟรชหน่วยความจำนี้เสมือนเป็นการแอคเซส (Access) หน่วยความจำ โดยที่ DRAM จะต้องถูกแอคเซสที่ทุก ๆ แอดเดรสในช่วงเวลาไม่เกิน 2 มิลลิวินาที

สำหรับซีพียู Z80180 CPU ให้แอดเดรส A_0-A_7 สำหรับ Dynamic RAM และยังสามารถโปรแกรมเวลารีเฟรช โดยการโปรแกรมที่ **Refresh Control Register (RCR :Address I/O 36H)**

บิท	7	6	5	4	3	2	1	0
	REFE	REFW	-	-	-	-	CYC1	CYC0

รูปที่ 2.10 แสดงบิทต่างๆใน Refresh Control Register

RERE - Refresh Enable เมื่อบิทนี้มีค่าเป็น 0 จะดิสเอเบิล แต่ถ้าเป็น 1 จะให้สัญญาณรีเฟรช ซึ่งในระหว่างการรีเซ็ต บิทนี้จะเป็น 1

REFW - Refresh Wait เมื่อบิทนี้เป็น 0 จะให้สัญญาณรีเฟรชทุก ๆ 2 clock ถ้าบิทนี้เป็น 1 จะเพิ่ม Refresh Wait เข้าในอีก 1 ซึ่งในระหว่างการรีเซ็ต บิทนี้จะเป็น 1

CYC1,CYC0 - Cycle Interval ใช้กำหนดช่วงเวลาในการรีเฟรช เช่นกรณีของไดนามิกแรมจะต้องรีเฟรช 128 ครั้ง ทุก ๆ 2 มิลลิวินาที (หรือ 256 ครั้งทุก ๆ 4 มิลลิวินาที) เพราะฉะนั้นสัญญาณรีเฟรชแต่ละครั้งจะต้องไม่น้อยกว่าหรือเท่ากับ 15.625 ไมโครวินาที จากตารางค่าที่ขีดเส้นใต้ เป็นค่าโปรแกรมที่เหมาะสมกับสัญญาณนาฬิกาที่ใช้ในระบบ

ตารางที่ 2.8 แสดงการโปรแกรมค่าในบิท CYC1 และ CYC0

CYC1	CYC0	Insertion Interval	Time Interval			
			φ:8 MHz	6 MHz	4 MHz	2.5 MHz
0	0	10 states	1.25μs	1.66μs	2.5μs	4.0μs
0	1	20 states	2.50μs	3.30μs	5.0μs	8.0μs
1	0	40 states	5.00μs	6.60μs	10.0μs	16.0μs
1	1	80 states	10.00μs	13.30μs	20.0μs	32.0μs

DMA Controller (DMAC)

มีด้วยกัน 2 Channel เพื่อเป็นการเพิ่มความเร็วในการย้ายข้อมูล โดยการกระทำที่ไม่ต้องผ่านซีพียู โดยมีความสามารถดังนี้

-Memory Address Space โดยสามารถกำหนดแอดเดรสต้นทาง (Source) และแอดเดรสปลายทาง (Destination) ที่ใดก็ได้ใน 1024 Kbyte

-I/O Address Space สามารถกำหนดที่ใดก็ได้ใน 64 KByte ทั้งแอดเดรสต้นทาง (Source) และ แอดเดรสปลายทาง (Destination)

-Transfer Length ใช้เป็นตัวนับในการย้ายข้อมูลได้เป็นบล็อค ๆ ละ 64 KByte

-$\overline{\text{DREQ}}$ เป็นขาอินพุท จะตรวจสจับที่ระดับหรือขอบของสัญญาณ

-$\overline{\text{TEND}}$ เป็นขาเอาท์พุท เพื่อบอกกับอุปกรณ์ภายนอกว่าทำ DMA หมดบล็อคแล้ว

-Transfer Rate ในการย้ายข้อมูลแต่ละครั้งจะเกิดทุก ๆ 6 Clock และ สำหรับ

หน่วยความจำ หรืออินพุทเอาท์พุทที่ทำงานช้า ก็สามารถเพิ่ม WAIT State เข้าไปใน DMA ได้ ซึ่งในระบบ System Clock (ϕ)=6 MHz อัตราการย้ายข้อมูลจะสูงถึง 1 MByte ใน 1 วินาที (ไม่มี WAIT State)

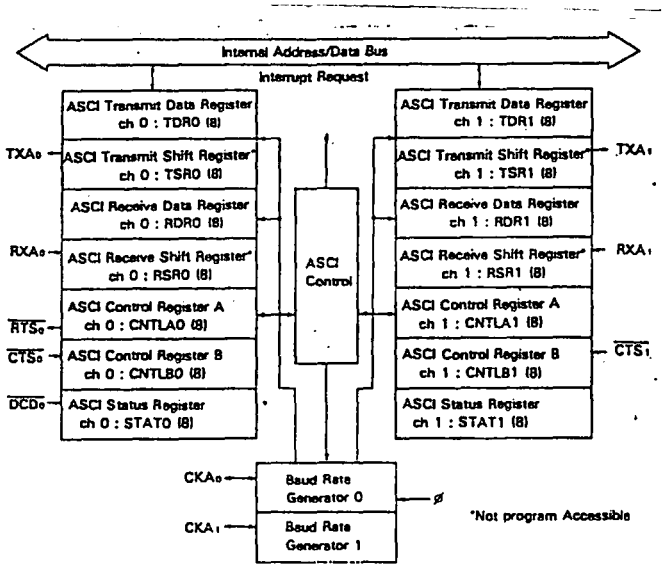
ความสามารถของแต่ละ Channel

1.Channel 0 สามารถย้ายข้อมูล Memory <-> Memory , Memory <-> I/O, Memory <-> Memory I/O Map และสามารถให้แอดเดรสในการย้ายข้อมูลเพิ่ม, ลด หรือให้คงที่ได้ การย้ายข้อมูลจะเป็นแบบ Cycle Steal (ชโมยเวลาเป็นช่วง) คือเมื่อการย้ายข้อมูลครบ 1 หรือ 2 ไบท์ ก็จะคืนบัสให้กับซีพียู จนอุปกรณ์พร้อมก็จะย้ายข้อมูลต่อ ทำอย่างนี้สลับกันไปจนหมดบล็อค ซึ่งการย้ายข้อมูลแบบนี้จะใช้สำหรับอุปกรณ์ที่ทำงานช้า ส่วนอีกแบบหนึ่งคือ Burst เป็นการย้ายข้อมูลแบบต่อเนื่อง กล่าวคือ จะมีการย้ายข้อมูลจนจบบล็อค จึงจะคืนบัสให้ซีพียู

2.Channel 1 ใช้ในการย้ายข้อมูล Memory <-> I/O โดยที่แอดเดรสของหน่วยความจำสามารถกำหนดให้เพิ่มหรือลดได้

Asynchronous Serial Communication Interface (ASCI)

มีด้วยกัน 2 Channel โดยมีบล็อคไดอะแกรม ดังรูป 2.11



รูปที่ 2.11 แสดงบล็อคไดอะแกรมของ ASCI

รีจิสเตอร์ที่สำคัญ

1. Transmit Shift Register 0,1 (TSR0,1) เป็นชิฟท์รีจิสเตอร์ที่รับข้อมูลจาก Transmit Data Registor (TDR) แล้วนำข้อมูลนั้นชิฟท์ออกที่ขา TXA

2. Transmit Data Registor 0,1 (TDR0,1 : I/O Adress 06H,07H) เป็นรีจิสเตอร์ที่ใช้ส่งข้อมูลออกไปที่ขา TXD โดยการนำข้อมูลใน TDR ส่งไปที่ TSR เมื่อ TSR ว่างลงและสามารถที่จะเขียนข้อมูลเข้าไปที่ TDR ได้อีก ในขณะที่ TSR กำลังชิฟท์ข้อมูลออกไปที่ขา TXA

3. Receive Shift Register 0,1 (RSR0,1) เป็นรีจิสเตอร์ที่รับข้อมูลจากขา RXAเมื่อรับจนเต็มบัฟเฟอร์แล้ว ก็จะชิฟท์ไปที่ RDR แต่ถ้า RSR ไม่ว่าง ในขณะที่มีการรับข้อมูลไบท์ต่อไปเข้ามาอีก จะทำให้เกิดข้อมูลทับซ้อนกันขึ้น และเกิดความผิดพลาดได้ ซึ่งผลของความผิดพลาดจะแสดงที่รีจิสเตอร์สถานะ โดยที่รีจิสเตอร์นี้ไม่สามารถโปรแกรมได้

4. Receive Data Register 0,1 (RDR0,1 : I/O Address 08H,09H) เป็นรีจิสเตอร์ที่ใช้เก็บข้อมูลที่รับเข้ามาจากขา RXA และในขณะที่ RDR กำลังบรรจุข้อมูลที่รับเข้ามาจากRSR นั้น ข้อมูลไบท์ถัดไปสามารถรับเข้ามาต่อได้

5. Status Register 0,1 (STAT0,1: I/O Address 04H,05H) ในแต่ละ Channel มีรีจิสเตอร์ใช้สำหรับตรวจสอบการสื่อสารเกี่ยวกับการผิดพลาด และสถานะสัญญาณควบคุมโมเด็ม การอีนาเบิลและการดิสเอเบิล ASCI ดังรูป 2.12

บิท	7	6	5	4	3	2	1	0
STAT0	RDRF	OVRN	PE	FE	RIE	DCDO	TDRE	TIE

บิท	7	6	5	4	3	2	1	0
STAT1	RDRF	OVRN	PE	FE	RIE	CTSIE	TDRE	TIE

รูปที่ 2.12 แสดงบิทต่าง ๆ ใน Status Register 0,1

RDRF - Receive Data Register Full (บิท 7) บิทนี้จะถูกเซ็ตเป็น 1 เมื่อข้อมูลที่รับเข้ามาถูกส่งไปที่ RDR เรียบร้อยแล้ว (ครบไบท์) แต่ถ้าการรับข้อมูลเกิดความผิดพลาด บิท RDRF ก็ยังถูกเซ็ตเป็น 1 ค้างอยู่ ทำให้ข้อมูลที่ผิดพลาดนั้นถูกส่งไปที่ RDR และคงอยู่เช่นกัน ดังนั้นต้องทำการเคลียร์ Flag Error พบว่าบิท RDRF นี้ถูกเคลียร์เป็น 0 เมื่อมีการอ่าน RDR, เมื่ออินพุท DCD_0 เป็น High, กำลังอยู่ในโหมด IOSTOP และในระหว่างการรีเซ็ต

OVRN - Overrun Error (บิท 6) บิทนี้จะเป็น 1 ก็ต่อเมื่อ RDR เต็ม และ RSR เต็มแล้วยังมีการรับข้อมูลเข้ามาอีก โดยที่บิทนี้จะถูกเคลียร์เมื่อบิท EFR ใน CNTLA มีค่าเป็น 0 DCD0 เป็น High IOSTOP และระหว่างการรีเซ็ต

PE - Parity Error (บิท 5) บิทนี้เป็น 1 เมื่อข้อมูลที่รับเข้ามามีพาริตี้ผิดพลาด และบิทนี้จะถูกเคลียร์เช่นเดียวกับ OVERUN

FE - Framing Error (บิท 4) เมื่อข้อมูลที่รับเข้ามามีรูปแบบผิดไปจากที่กำหนด จะทำให้บิทนี้ถูกเซ็ตเป็น 1 และการเคลียร์ก็เช่นเดียวกับ OVERUN

RIE - Receive Interrupt Enable (บิท 3) เมื่อบิทนี้เป็น 1 จะอนุญาติให้ ASCI ทำการขออินเทอร์รัพท์ได้ โดยที่ RDRF,OVRN,PE หรือ FE ถูกเซ็ตเป็น 1 ด้วย เมื่อนั้น ASCI ก็จะให้สัญญาณอินเทอร์รัพท์ สำหรับ ASCI Channel 0 Interrupt สามารถเกิดขึ้นโดย การเปลี่ยนแปลงที่ขาสัญญาณอินพุทภายนอกที่ขา DCD0 จาก Low เป็น High และในระหว่างการ รีเซ็ต บิท RIE จะถูกเคลียร์เป็น 0

$\overline{\text{DCD0}}$ - Data Carrier Detect Bit (บิท 2) บิทนี้จะถูกเซ็ตเป็น 1 เมื่อขา อินพุท $\overline{\text{DCD0}}$ เป็น High และจะถูกเคลียร์เป็น 0 จากการอ่าน STAT0 ในครั้งแรก จากนั้นขา อินพุท $\overline{\text{DCD0}}$ จะถูกเปลี่ยนจาก High เป็น Low และระหว่างการรีเซ็ต เมื่อ $\overline{\text{DCD0}}$ เป็น 1 ส่วนของภาครับจะไม่ทำงาน

CTSIE - Channel 1 $\overline{\text{CTS}}$ Enable (บิท 1) Channel 1 จะมีขาอินพุท $\overline{\text{CTS1}}$ ภายนอก ซึ่งมัลติเพล็กซ์กับ RXS เมื่อเซ็ตบิทนี้ให้เป็น 1 จะถูกเลือกเป็นขา $\overline{\text{CTS1}}$

TDRE - Transmit Data Register Empty (บิท 0) เป็นตัวบอกว่าข้อมูลพร้อม ที่จะส่งได้หรือไม่ ถ้าเป็น 1 คือพร้อมที่จะส่งข้อมูลแล้ว ให้เขียนข้อมูลเข้าไปที่ TDR ได้ และเมื่อ มีการเขียนข้อมูลเข้าไปที่ TDR ก็จะทำให้ TDRE เป็น 0 และข้อมูลใน TDR ก็จะถูกส่งให้ TSR จน TDR ว่างลง TDRE ก็จะกลับเป็น 1 อีกครั้ง

TIE - Transmit Interrupt Enable (บิท 0) เมื่อบิทนี้เป็น 1 จะอนุญาติให้ ASCI ใช้การส่งแบบอินเทอร์รัพท์ได้ โดยที่ TDRE ต้องเป็น 1 ด้วย และในระหว่างการรีเซ็ตบิท นี้จะถูกเคลียร์เป็น 0

6. Control Register A (CNTLA0,1 : I/O Address 00H-01H) เป็นรีจิส เตอร์กำหนดการทำงาน ประกอบด้วยบิทต่าง ๆ ดังรูปที่ 2.13

บิท	7	6	5	4	3	2	1	0
CNTLA0	MPE	RE	TE	RTS0	MPBR/EFR	MOD2	MOD1	MOD0

บิท	7	6	5	4	3	2	1	0
CNTLA1	MPE	RE	TE	CKA1D	MPBR/EFR	MOD2	MOD1	MOD0

รูปที่ 2.13 แสดงบิทต่าง ๆ ใน Control Register A 0,1

MPE - Multiprocessor Mode Enable (บิท 7) ใช้ในการอีนาเบิลการสื่อสารแบบไมโครโปรเซสเซอร์ร่วม เมื่อมีการเลือกโหมดการสื่อสารแล้ว (MP=1 ใน CNTLB) ในการสื่อสารแบบนี้รูปแบบของการสื่อสารจะมีบิทพิเศษเพิ่มเข้ามาเรียกว่าบิท MPB ซึ่งบิทนี้จะใช้ในการตรวจสอบหรือใช้งาน เมื่ออีนาเบิลบิท MPE ให้เป็น 1 และถ้า MPB=1 เมื่อนั้นภาครับของมัลติ-โปรเซสเซอร์จะทำงานคือ RDRF และ Error Flag จะทำงาน แต่ถ้า MPB=0 ASCI จะไม่สนใจข้อมูลไบท์นั้น ถ้า MPE=0 จะไม่สามารถทำการสื่อสารแบบไมโครโปรเซสเซอร์ร่วมได้ แม้จะเซ็ต MP เป็น 1 แล้วก็ตาม

RE - Receiver Enable (บิท 6) ถ้าบิทนี้เป็น 1 จะอีนาเบิลการรับของ ASCI แต่ถ้าบิทนี้เป็น 0 จะดิสเอเบิลการรับ แต่ RDRF และ Error Flag จะไม่ถูกรีเซ็ตตาม

TE - Transmit Enable (บิท 5) ถ้าบิทนี้เป็น 1 จะอีนาเบิลการส่ง แต่ถ้าบิทนี้เป็น 0 จะดิสเอเบิล แต่แฟลก TDRE จะไม่ถูกรีเซ็ตตาม

$\overline{\text{RTS0}}$ - Request to Send Channel 0 (บิท 4 ในรีจิสเตอร์ CNTLA0) บิทนี้จะให้ผลเช่นเดียวกับขาเอาท์พุท RTS0 คือ ถ้าบิทนี้เป็น 1 ขาเอาท์พุท $\overline{\text{RTS0}}$ ก็จะเป็น 1 ถ้าบิทนี้เป็น 0 ขาเอาท์พุทก็จะเป็น 0 เช่นกัน และในระหว่างการรีเซ็ตบิท RTS0 จะถูกเซ็ตเป็น 1

CKA1D - CKA1 Clock Disable (บิท 4 ในรีจิสเตอร์ CNTLA1) ขาของ CKA1 จะมัลติเพล็กซ์กับ $\overline{\text{TEND0}}$ เมื่อบิทนี้เป็น 1 จะเลือกเป็นขา $\overline{\text{TEND0}}$ แต่ถ้าบิทนี้เป็น 0 ก็จะเป็นขา Clock ของ ASCI Channel 1 และในระหว่างรีเซ็ตบิทนี้จะเป็น 0

MPBR/EFR - Multiprocessor Bit Receive/Error Flag (บิท 3) เมื่อบิทนี้ถูกอ่านจะใช้ดูบิท MPB ในกรณีที่พบไมโครโปรเซสเซอร์ที่จะทำการติดต่อกันแล้ว และดิสเอเบิล

ไมโครโปรเซสเซอร์ตัวอื่น ๆ โดยการส่งบิท MPB ให้เป็น 0 เมื่ออ่านจะได้รู้ว่า MPB เป็น 0 จริง แต่ถ้าเขียนให้บิทนี้เป็น 0 จะเป็นการรีเซ็ต Error Flag ในการรับ

MOD2,1,0 - ASCI Data Format Mode 2,1,0 (บิท 2,1,0) โดยที่

MOD2 = 0 ข้อมูลเป็นแบบ 7 บิท

= 1 ข้อมูลเป็นแบบ 8 บิท

MOD1 = 0 ไม่มีพาริตี้

= 1 มีพาริตี้

MOD0 = 0 มี 1 Stop Bit

= 1 มี 2 Stop Bit

ซึ่งสามารถสรุปได้ดังตารางที่ 2.9

ตารางที่ 2.9 แสดงรูปแบบของข้อมูลเมื่อมีการเซ็ตค่าในบิท MOD2,1,0

MOD2	MOD1	MOD0	รูปแบบของข้อมูล
0	0	0	Start + 7 Bit Data + 1 Stop Bit
0	0	1	Start + 7 Bit Data + 2 Stop Bit
0	1	0	Start + 7 Bit Data + Parity + 1 Stop Bit
0	1	1	Start + 7 Bit Data + Parity + 2 Stop Bit
1	0	0	Start + 8 Bit Data + 1 Stop Bit
1	0	1	Start + 8 Bit Data + 2 Stop Bit
1	1	0	Start + 8 Bit Data + Parity + 1 Stop Bit
1	1	1	Start + 8 Bit Data + Parity + 2 Stop Bit

7. ASCI Control Register B 0,1 (CNTLB0,1:I/O Address 02H,03H)

บิท	7	6	5	4	3	2	1	0
CNTLB0,1	MPBT	MP	CTS/PS	PEO	DR	SS2	SS1	SS0

รูปที่ 2.14 แสดงบิทต่าง ๆ ใน ASCI Control Register B 0,1

MPBT - Multiprocessor Bit Transmit (บิท 7) ใช้ในการส่งบิท MPB โดย ถ้าบิท MPBT=1 เมื่อนั้นบิท MPB=1 และถ้าบิท MPBT=0 ทำให้บิท MPB=0 ด้วย และในระหว่าง การรีเซ็ตไม่สามารถกำหนดได้

MP - Multiprocessor Mode (บิท 6) ถ้าบิทนี้เป็น 1 จะเป็นการเซ็ตการติดต่อ แบบไมโครโปรเซสเซอร์ร่วม โดยใช้รูปแบบของ MOD2 กับ MOD0 โดยยกเว้น MOD1 ดังนี้

Start Bit + 7 หรือ 8 Data Bit + MPB Bit + 1 หรือ 2 Stop Bit

ในระหว่างการรีเซ็ตบิท MP จะเป็น 0

$\overline{\text{CTS}}$/PS - Clear to Send/Prescale (บิท 5) เมื่ออ่านบิทนี้จะใช้แสดงสถานะ ของขาอินพุท $\overline{\text{CTS}}$ ภายนอก ถ้าขา $\overline{\text{CTS}}$ เป็น High ภาคส่งของ ASCI จะไม่ทำงานแต่ถ้าเขียน เข้าไปที่บิทนี้จะเป็นการกำหนด Baud Rate และในระหว่างการรีเซ็ตบิทนี้จะเป็น 0

PEO - Parity Even Odd (บิท 4) บิทนี้จะไม่มีผลต่อการอีนาเบิล หรือดิสเอเบิล ของพาริตี้ (MOD1 ในรีจิสเตอร์ CNTLA) แต่ถ้าใช้เลือกว่าเมื่อมีการอีนาเบิลพาริตี้ใน MOD1 จะ ให้พาริตี้คู่หรือคี่ ถ้า PEO = 0 คือพาริตี้คู่ แต่ถ้า PEO = 1 คือพาริตี้คี่

DR - Divide Ratio (บิท 3) ใช้กำหนด Baud Rate โดยในระหว่างการรีเซ็ต บิทนี้จะเป็น 0

SS2,1,0 - Source/Speed Select 2,1,0 (บิท 2,1,0) ใช้กำหนด Clock ว่าจะให้เป็นภายใน หรือภายนอก (โดยภายนอกคือขา Clock CKA) และเป็นตัวกำหนด Baud-Rate ด้วย ในระหว่างการรีเซ็ตบิททั้ง 3 บิทนี้จะเป็น 1 คือ เป็นการใช้ Clock จากภายนอก นั่นเอง และจากที่กล่าวมา การกำหนด Baud Rate จึงมีด้วยกันหลายตัวสามารถสรุปเป็นตาราง ดังนี้

ตารางที่ 2.10 แสดงการกำหนดค่า Baud Rate โดยการเซ็ตค่าต่าง ๆ

Prescaler		Sampling Rate		Baud Rate				General Divide Ratio	Baud Rate (Example) (BPS)			CKA	
PS	Divide Ratio	DR	Rate	SS2	SS1	SS0	Divide Ratio		φ=6.144 MHz	φ=4.608 MHz	φ=3.072 MHz	I/O	Clock Frequency
0	φ÷10	0	16	0	0	0	÷1	φ÷160	38400		19200		φ÷10
				0	0	1	2	320	19200		9600		20
				0	1	0	4	640	9600		4800		40
				0	1	1	8	1280	4800		2400	0	80
				1	0	0	16	2560	2400		1200		160
				1	0	1	32	5120	1200		600		320
				1	1	0	64	10240	600		300		640
				1	1	1	–	fc÷16	–	–	–	1	fc
		1	64	0	0	0	÷1	φ÷640	9600		4800		φ÷16
				0	0	1	2	1280	4800		2400		20
				0	1	0	4	2560	2400		1200		40
				0	1	1	8	5120	1200		600	0	80
				1	0	0	16	10240	600		300		160
				1	0	1	32	20480	300		150		320
				1	1	0	64	40960	150		75		640
				1	1	1	–	fc÷64	–	–	–	1	fc
1	φ÷30	0	16	0	0	0	÷1	φ÷480		9600			φ÷30
				0	0	1	2	960		4800			60
				0	1	0	4	1920		2400			120
				0	1	1	8	3840		1200		0	240
				1	0	0	16	7680		600			480
				1	0	1	32	15360		300			960
				1	1	0	64	30720		150			1920
				1	1	1	–	fc÷16	–	–	–	1	fc
		1	64	0	0	0	÷1	φ÷1920		2400			φ÷30
				0	0	1	2	3840		1200			60
				0	1	0	4	7680		600			120
				0	1	1	8	15360		300		0	240
				1	0	0	16	30720		150			480
				1	0	1	32	61440		75			960
				1	1	0	64	122880		37.5			1920
				1	1	1	–	fc÷64	–	–	–	1	fc

Programmable Reload Timer (PRT)

ประกอบด้วย 2 Channel เป็น 16 Bit Programmable Reload Timer และที่ Channel 1 จะมีขาเอาท์พุทซึ่งสามารถที่จะเซ็ตให้เป็น High ,Low หรือ Toggle ก็ได้ ดังนั้นจึงสามารถนำเอาท์พุทดังกล่าวนี้ไปใช้ในการกำเนิดรูปคลื่นได้ตามต้องการ ซึ่งแต่ละ Channel มีรีจิสเตอร์ที่สำคัญดังนี้

1. Timer Data Register (TMDR0,1 : I/O Address Ch0L;0CH,Ch0H; 0Dh,Ch1L;14H,Ch1H;15H) เป็นรีจิสเตอร์ 16 บิท ใช้กำหนด Timer โดยแอดเดรส I/O สูงเก็บค่า Timer ค่าสูง โดย TMDR จะนับลง 1 ครั้งทุก ๆ 20 Clock ของระบบ เมื่อ TMDR นับลงเป็น 0 ค่าใน Reload จะถูกโหลดมาให้กับ TMDR โดยอัตโนมัติ และการอ่านค่าใน TMDR สามารถอ่านได้เลยโดยไม่ต้องหยุด PRT แต่ถ้าเป็นการเขียนต้องหยุด PRT ก่อน ซึ่งในระหว่างการรีเซ็ต TMDR0 และ TMDR1 จะเป็น OFFFFH

2: Timer Reload Register (RLDR:I/O Address Ch0L;0EH,Ch0H;0EH, Ch1L;16H,Ch1H-17H) เป็นรีจิสเตอร์ 16 บิทที่ใช้ในการเก็บค่าเริ่มต้นของ Timer โดยที่จะมีการโหลดค่าที่อยู่ใน RLDR ไปให้กับ TMDR โดยอัตโนมัติเมื่อ TMDR ลดลงเป็น 0 ซึ่งในระหว่างการรีเซ็ต ค่าใน RLDR0 และ RLDR1 จะเป็น OFFFFH

3. Timer Control Register (TCR : I/O Address 10H) เป็นรีจิสเตอร์ที่ใช้แสดงสถานะและควบคุมดังนี้

บิท	7	6	5	4	3	2	1	0
	TIF1	TIF0	TIE1	TIE0	TOC1	TOC0	TDE1	TDE0

รูปที่ 2.15 แสดงบิทต่าง ๆ ใน Timer Control Register

TIF1 - Timer Interrupt Flag 1 (บิท 7) เมื่อ TMDR1 ลดลงเป็น 0 TIF1 จะถูกเซ็ตเป็น 1 และถ้า TIE1 = 1 ก็จะทำให้เกิดอินเทอร์รัพท์ขึ้นได้ ทั้งนี้ TIF1 จะถูกเคลียร์เป็น 0 ก็ต่อเมื่อทำการอ่านค่าใน TCR และอ่านค่าในไบท์สูงหรือไบท์ต่ำของ TMDR1 ในระหว่างการรีเซ็ตค่าของ TIF จะเป็น 0

TIF0 - Timer Interrupt Flag 0 (บิท 6) มีหลักการเช่นเดียวกับ TIF1

TIE1,0 - Timer Interrupt Enable 1,0 (บิท 5,4) เมื่อเซ็ตให้เป็น 1 จะอนุญาติให้อินเทอร์รัพท์ได้ และในระหว่างการรีเซ็ตบิททั้งสองนี้จะเป็น 0

TOC1,0 - Timer Output Control 1,0 (บิท 3,2) บิททั้งสองนี้ใช้ควบคุมขาเอาท์พุทของ PRT1 โดยถ้าทั้งสองบิทนี้เป็น 0 จะเป็นการใช้งาน A18 นอกนั้นจะเป็นการกำหนดให้ TOUT เป็น High,Low หรือ Toggle ดังตารางที่ 2.11 และในระหว่างการรีเซ็ต 2 บิทนี้จะเป็น 0

ตารางที่ 2.11 แสดงการกำหนดเอาท์พุทโดยการเซ็ตค่าใน TOC1,0

TOC1	TOC0	เอาท์พุท
0	0	แอดเดรส A18
0	1	Toggle
1	0	Low
1	1	High

TDE1,0 - Timer Down Count Enable (บิท 1,0) เมื่อเซ็ตค่าให้เป็น 1 ก็คือให้เริ่มทำการนับ TMDR ได้ แต่ถ้าเป็น 0 จะทำให้การนับหยุดทำงาน และในระหว่างการรีเซ็ตบิททั้งสองนี้จะเป็น 0

ในการคำนวณเวลา เราทราบแล้วว่า Timer จะนับลงทุก ๆ 20 Clock ของระบบ

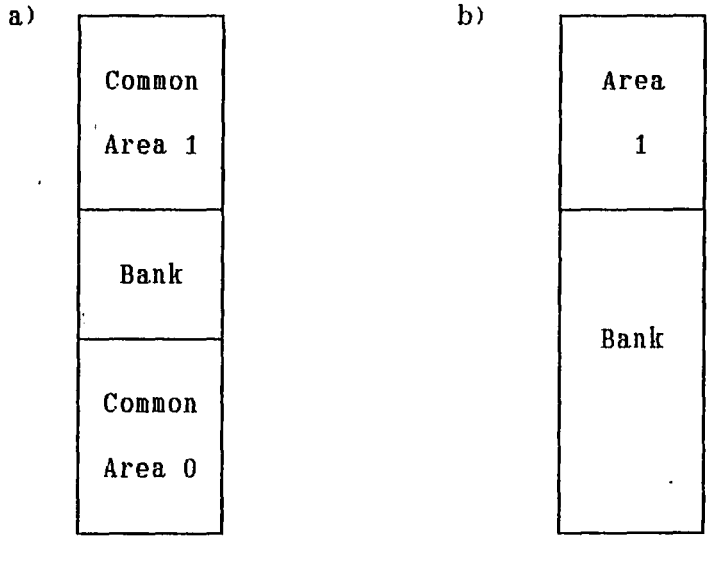
ถ้าคริสตอลที่ใช้งานคือ 12 MHz ความถี่ที่ทำงานบนบอร์ดจะเป็น 6 MHz ดังนั้นการหาค่า Timer คือ $T = 1/F = 1/(6x10^6) = 0.1666\ \mu s$ ต่อ 1 Clock

ดังนั้น ถ้า 20 Clock จะเป็น $0.1666\ \mu s \times 20 = 3.333\ \mu s$

นั่นคือ Timer จะนับลง 1 ครั้งทุกๆ $3.333\ \mu s$ ที่คริสตอล 12 MHz เช่นให้ TMDR มีค่าเท่ากับ 1 และเซ็ตแฟลกอินเทอร์รัพท์ไว้ จะทำให้ PRT เกิดการอินเทอร์รัพท์ทุกๆ $6.666\mu s$ เพราะการนับลงจะนับจากตัวเองลงก่อนคือ 1 แล้วก็เป็น 0 ดังนั้นจึงเท่ากับ 2 ครั้งนั่นเอง และถ้าให้กำเนิดสัญญาณสี่เหลี่ยมที่ TOUT ก็จะทำให้เกิดการ Toggle กันทุก ๆ จำนวนที่ให้นับ เช่น ตัวอย่างข้างบนจะเป็น High $6.666\mu s$ และ Low $6.666\mu s$ ดังนั้น 1 ลูกสัญญาณจะประมาณ $13\ \mu s$ หรือความถี่จะลดลงหนึ่งเท่าของค่าเวลาที่คิดจากจำนวนครั้งในการนับค่าก็ได้

Memory Management Unit (MMU)

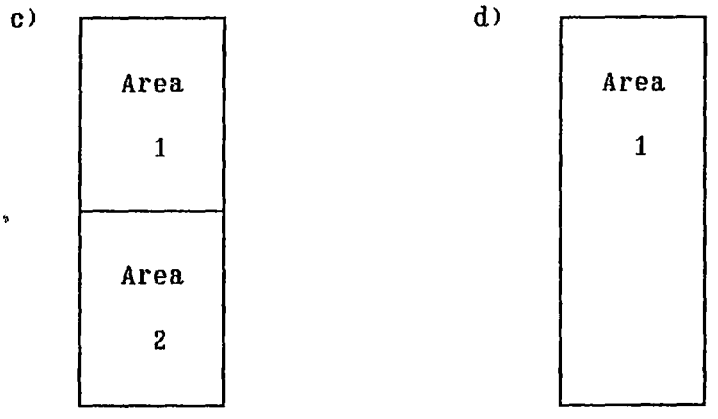
ใช้เป็นตัวขยายหน่วยความจำ (Memory) จาก 64K (Logical) เป็น 1 Mbyte (Physical) โดยการแบ่ง 64 Kbyte Logical (คือ แอดเดรสปกติที่ใช้เช่นเดียวกับ Z80 CPU) ในการใช้งานด้วยกันเป็น 3 ส่วน คือ Common Area 0 ,Bank Area และ Common Area 1 โดยการกำหนดโปรแกรมจัดแผนผังทางลอจิก (Map Logical) ในรีจิสเตอร์ I/O CBAR (แอดเดรส 3AH) ซึ่งในรีจิสเตอร์นี้จะถูกแบ่งเป็น 2 นิบเบิ้ล คือ 4 บิทสูงและ 4 บิทต่ำ โดย 4 บิทสูงใช้โปรแกรมพื้นที่ของ Common Area 1 และ 4 บิทต่ำใช้โปรแกรมในส่วน Bank Area ดังนั้นการโปรแกรมรีจิสเตอร์ CBAR นี้ก็จัดแผนผัง (Map) ได้เป็น 2^2 คือ 4 รูปแบบ ดังรูปที่ 2.16



A1>Bank>A0 A1>Bank=A0(0H)

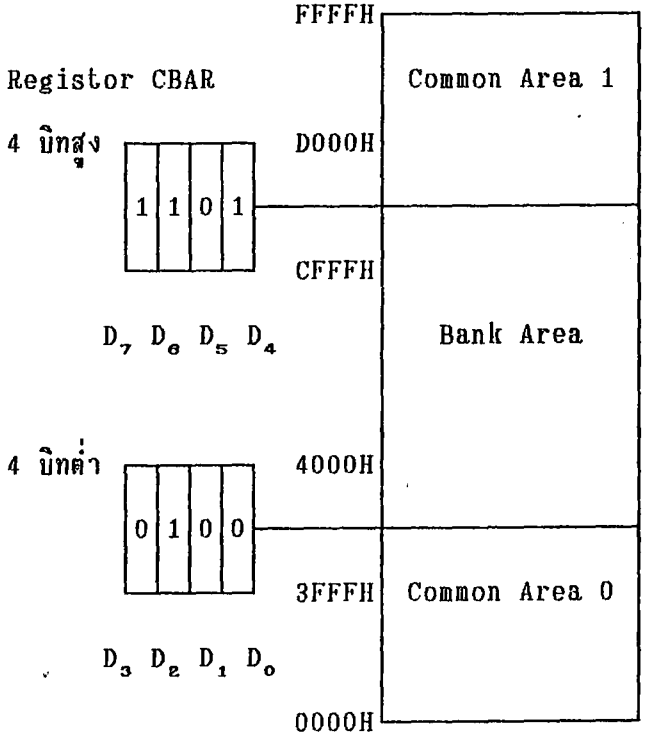
CBAR = D4H CBAR = F0H (ขณะรีเซ็ตจะเป็นเช่นนี้)

รูปที่ 2.16 แสดงรูปแบบการจัดหน่วยความจำแบบต่าง ๆ



รูปที่ 2.16 แสดงรูปแบบการจัดหน่วยความจำแบบต่าง ๆ (ต่อ)

Map Logical ปกติ



รูปที่ 2.17 Map Logical แบบปกติ

จากรูปเป็นการโปรแกรมรีจิสเตอร์ CBAR ให้ Map Logical เป็น Common Area 0 ตั้งแต่แอดเดรส 0000H-3FFFH , Bank Area ตั้งแต่แอดเดรส 4000H-CFFFH และ Common Area 1 ตั้งแต่แอดเดรส D000H-FFFFH ทั้งนี้เป็นไปตามค่าในรีจิสเตอร์ CBAR ทั้ง 2 นิบเบิล

เพราะนิบเบิ้ลสูงเป็นของ Area 1 ซึ่งคือ ODH ก็คือ Area 1 เริ่มตั้งแต่ D000H-FFFFH และนิบเบิ้ลต่ำจะเป็นจุดสิ้นสุดของ Bank Area ซึ่งก็คือ 04H ก็คือ ถัดจาก Area 1 เป็นต้นไปจนถึง 4000H เป็น Bank Area และที่เหลือจึงเป็น Area 0 นั่นเอง

จากค่าที่โปรแกรมใน CBAR จึงทำให้โปรแกรม Common Area ทั้ง 2 และ Bank Area ได้ตั้งแต่ 4 Kbyte ขึ้นไป เช่น ให้นิบเบิ้ลสูงของ CBAR = OFH ก็คือ Common Area 1 มีค่าตั้งแต่แอดเดรส F000H-FFFFH (คือ 4 K อย่างต่ำนั่นเอง) และจุดที่น่าสังเกตจากการจัด Map ทั้ง 4 รูปแบบ นั่นก็คือ Common Area 0 และ Bank Area สามารถมีตำแหน่งที่ทับซ้อนกันได้ (ตำแหน่งเดียวกัน) และ Common Area 1 กับ Bank Area ก็สามารถโปรแกรมให้อยู่ที่ใดก็ได้อย่างอิสระตั้งแต่ 4 Kbyte ขึ้นไปของส่วน Physical Address (1 Mbyte โดยใช้ร่วมกับรีจิสเตอร์อีก 2 ตัว) ส่วน Common Area 0 นั้นจะเป็น Based หรือมอนิเตอร์ของระบบนั่นเอง

จากที่กล่าวมายังไม่ได้พูดถึงการขยายหน่วยความจำ ออกไปมากกว่า 64 K เพราะว่าการที่จะย้ายหน่วยความจำเกินกว่า 64 K นั้น จะต้องอ้างอิงส่วนของ Logical ด้วย เนื่องด้วยคำสั่งของ Z80 ไม่สามารถอ้างหน่วยความจำเกินกว่านี้ได้ ดังนั้นการอ้างถึงหน่วยความจำทั้งหมดจึงยังเป็นส่วนของ Logical แต่ข้อมูลที่ถูกกระทำจริงเป็นส่วนของ Physical เช่น ในคำสั่งอาจเป็นดังนี้

```
LD  A,(8000H)
```

ซึ่งดูจากคำสั่งนี้จะเป็นการทำกับตำแหน่ง 8000H (สมมติในส่วน Bank Area) แต่เราเซ็ต Physical Area ไว้ที่ 10000H นั่นหมายความว่า การทำคำสั่งข้างบนนี้ ข้อมูลจะถูกกระทำที่แอดเดรส 10000H นั่นเอง

<u>การคิด Physical Address</u>

1) จะกระทำในส่วนของ Bank และ Common Area 1 โดยผ่านทางรีจิสเตอร์ I/O CBR และ BBR คูณด้วย 1000H แล้วนำค่าที่ได้บวกกับ Logical Address ของส่วนนั้น ๆ (Bank หรือ Common Area 1)

2) การกระทำทั้งหมดเกิดขึ้นภายในซีพียูเอง ดังนั้นการอ้างแอดเดรสในโปรแกรมก็ยังเป็น 64 K คือตาม Logical ที่กำหนดใน CBAR นั่นเอง

<u>รีจิสเตอร์ที่ใช้ควบคุม</u> (Register Control)

1. **Common/Bank Area Register (CBAR : I/O Address 3AH)** ใช้กำหนดพื้นที่ของ Logical ที่เป็น Common Area 0 , Bank Area และ Common Area 1

บิท	7	6	5	4	3	2	1	0
	CA3	CA2	CA1	CA0	BA3	BA2	BA1	BA0

รูปที่ 2.18 แสดงบิทต่าง ๆ ใน Common/Bank Area Register

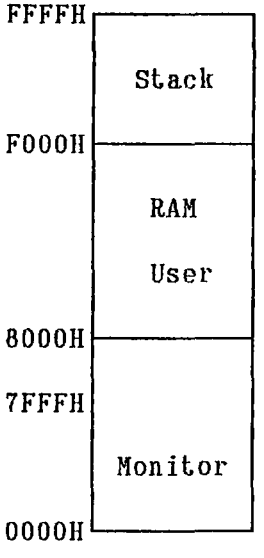
CA3 - CA0 เป็นตัวกำหนดแอดเดรสเริ่มต้นของ Common Area 1

BA3 - BA0 เป็นตัวกำหนดจุดแอดเดรสสุดท้ายของพื้นที่ Bank Area ที่ต่อจากจุดเริ่มต้นของ Common Area 1

2. Common Base Register (CBR : I/O Address 38H) เป็นรีจิสเตอร์ I/O 8 บิท เพื่อใช้กำเนิด Physical Common Area 1

3. Bank Base Register (BBR : I/O Address 39H) ใช้กำเนิด Physical Bank Area

ตัวอย่าง กำหนดให้มอนิเตอร์ (Monitor) ที่ 0000H-7FFFH และ RAM ใช้งานที่ 10000H โดยมีพื้นที่ Stack ที่ 18000H จากนั้นก็ต้องกำหนด Map ใน Logical โดยสมมติให้ Stack มีเนื้อที่ 4 K นอกนั้นเป็นส่วนของ Bank Area จากนั้นก็หาค่าให้กับ Bank Area และ Common Area 1 เช่น Map Logical กำหนดได้เป็นดังนี้



รูปที่ 2.19 แสดงการจัดหน่วยความจำตามตัวอย่าง

คำนวณหาค่าใส่ให้กับ BBR และ CBR โดยมี Stack ที่ 18000H (Physical) ที่ Logical เป็น F000H ดังนั้นค่าที่ให้กับ CBR เป็น

18000 _

F000

09000

จากที่ทราบแล้วว่า ค่าใน CBR จะคูณด้วย 1000H ดังนั้นในทางกลับกัน เมื่อจะนำค่ามาให้กับ CBR ก็ต้องทำการหารค่าผลต่างนั้นด้วย 1000H ก็จะได้ค่าใน CBR = 09H

ส่วน RAM User (Bank Area) ก็เช่นเดียวกัน จะได้เป็น

10000 _

8000

08000

นำค่าที่ได้หารด้วย 1000H จะได้ค่าใน BBR = 08H

ดังนั้นการโปรแกรมจากโจทย์ตัวอย่างจะเป็น

CBAR = 0F8H , BBR = 08H , CBR = 09H

เมื่อคำนวณกลับจะได้

CBAR นิบเบิ้ลต่ำแอดเดรสสุดท้ายของ Bank Area เป็น

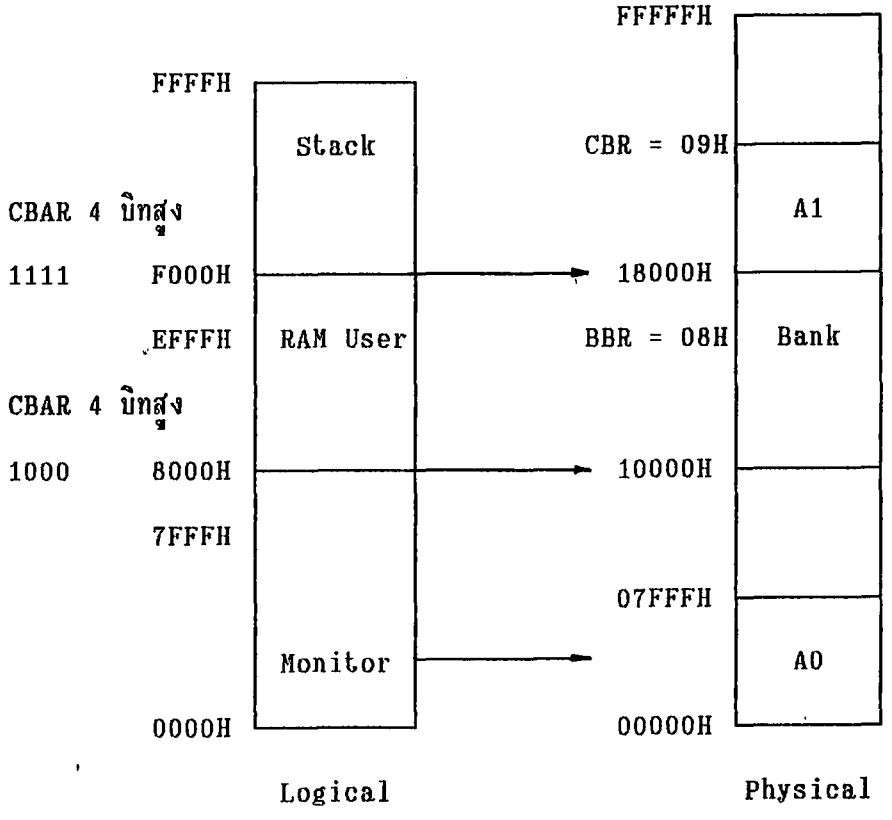
8000H + (BBR = (8)x1000H) = 10000H

CBAR นิบเบิ้ลสูงแอดเดรสเริ่มต้นของ Common Area 1 เป็น

F000H + (CBR = (9)x1000H) = 18000H

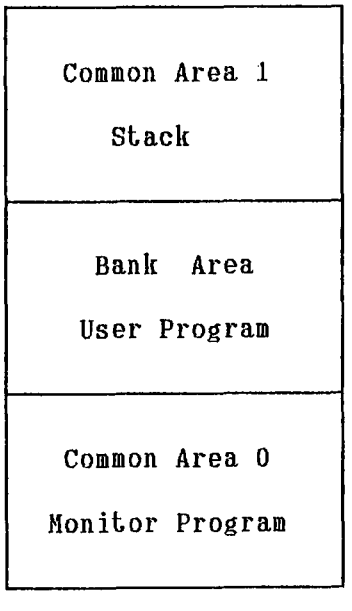
การโปรแกรม

```
LD   A,0F8H
OUT0 (CBAR),A
LD   A,08H
OUT0 (BBR),A
LD   A,09H
OUT0 (CBR),A
```



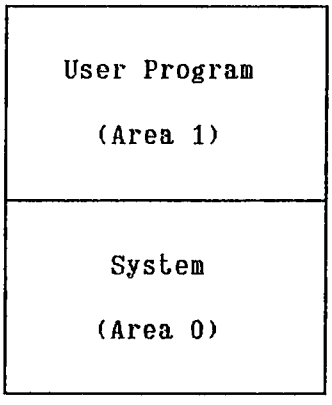
รูปที่ 2.20 แสดง Logical และ Physical memory ตามตัวอย่าง

ดังนั้นเราอาจจะกล่าวได้ว่าใน Logical ส่วนมากจะถูกจัดเป็นดังรูป 2.21



รูปที่ 2.21 แสดงการจัดหน่วยความจำโดยทั่วไปในส่วน Logical

โดยให้ส่วนของ Common Area 1 และ มอนิเตอร์คงที่ ส่วนของ Bank Area ให้ย้ายไปที่ใด ๆ ก็ได้ใน 1 Mbyte จะเป็นการขยายพื้นที่ของการใช้งานโดยใช้เนื้อที่ของ Stack เป็นการกระทำกับตัวแปรหรือข้อมูลอื่นใน 64 K อื่น ๆ เมื่อเรามอง Logical 64 K ออกเป็นหน้า ๆ (Page) ใน 1 Mbyte แต่ข้อเสียในการจัดแบบนี้จะทำให้ใช้ Bank Area ได้ไม่เต็มที่ กล่าวคือเมื่อให้ ROM Monitor อยู่ที่ 0000H-7FFFH และ RAM เริ่มตั้งแต่ 8000H-FFFFH ซึ่งจะเห็นว่า RAM ในส่วนนี้จะต้องเป็น Stack ด้วย เมื่อเรายึด Bank ออกไปที่ Physical อื่นก็จะไม่สามารถใช้ได้ถึง 32K เช่นมี RAM ที่ตำแหน่ง 18000H-1FFFFH อีกเราจะใช้ได้แค่ 24K เพราะพอเราอ้างที่ 0F000H แทนที่ข้อมูลจะถูกกระทำที่ 1F000H ก็จะมากระทำที่ 0F000H แทนตามที่กำหนด Common Area 1 ไว้ใน Logical จึงอาจแบ่ง Map เป็นลักษณะกว้าง ๆ ดังนี้



รูปที่ 2.22 แสดงการแบ่ง Map เป็นลักษณะกว้าง ๆ

โดยกำหนดให้ Common Area 1 เป็นส่วนของ User Program ส่วน System เป็นของ Common Area 0 ดังนั้นเมื่อเราให้ Common Area 1 เริ่มที่แอดเดรส 8000H ก็จะใช้ RAM ได้ถึง 32 K เต็ม ส่วน System ก็คือส่วนของ Area 0 ซึ่งเป็นส่วนมอนิเตอร์ แต่ในส่วนนี้เราได้กำหนดไว้ถึง 32 K คือจากแอดเดรส 7FFFH ลงไปถึง 0000H ซึ่งใน System นี้เราอาจจะใส่ RAM ไว้ในแอดเดรสช่วงนี้เพื่อเป็นเนื้อที่ของ Stack ก็จะทำให้เราย้ายเนื้อที่ของการใช้งานได้เต็ม

<u>สรุปได้ว่า</u>

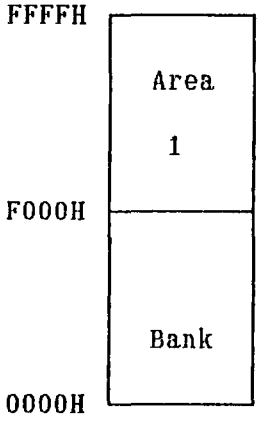
1) ในระหว่างการรีเซ็ต Logical ใน CBAR จะถูกกำหนดด้วยค่า 0F0H

2) ให้กำหนด Map Address ของ Logical ก่อนที่รีจิสเตอร์ CBAR (3AH)

3) BBR และ CBR จะเป็นตัวกำหนดตำแหน่งของข้อมูลในการใช้งานจริงในพื้นที่ 1 Mbyte (Physical Address) ซึ่งระหว่างการรีเซ็ต ค่าในรีจิสเตอร์ BBR และ CBR จะถูกรีเซ็ตเป็น 0

4) ในการคิดค่า Physical Address คือ นำค่าใน BBR หรือ CBR คูณด้วย 1000H แล้วบวกด้วย Logical ของพื้นที่นั้น ๆ

ข้อสังเกต ในการจัดรูปแบบการเซ็ต Map ทั้ง 4 อย่างที่กล่าวในตอนต้น เช่น ในขณะที่มีการรีเซ็ต CBAR = F0H จะเป็นดังรูป 2.23



รูปที่ 2.23 แสดงการจัดหน่วยความจำหลังจากรีเซต

ถ้าระบบของเรามีมอนิเตอร์ ที่แอดเดรส 0000H-7FFFH ที่ 8000H-FFFFH โดยมี Stack ที่แอดเดรส FE00H ถ้าไม่มีการเข้าไป Control Register ของ MMU ทั้ง 3 ตัวนี้ ระบบของเราก็ยังทำงานอยู่ได้ แต่ถ้าเราโปรแกรมให้ BBR = 10H ตอนนี้ระบบจะทำงานไม่ได้ สาเหตุที่เป็นเช่นนั้น เพราะเราไม่ได้โปรแกรม Logical ให้มี Common Area 0 แต่ที่ตอนแรกใช้งานได้เพราะ Bank Area ทับซ้อนกับ Common Area 0 อยู่ แต่เมื่อโปรแกรม BBR = 10H แล้ว Bank Area จะกลายเป็น 10000H จึงทำให้ Common Area 0 ไม่มีจึงทำให้ระบบทำงานไม่ได้

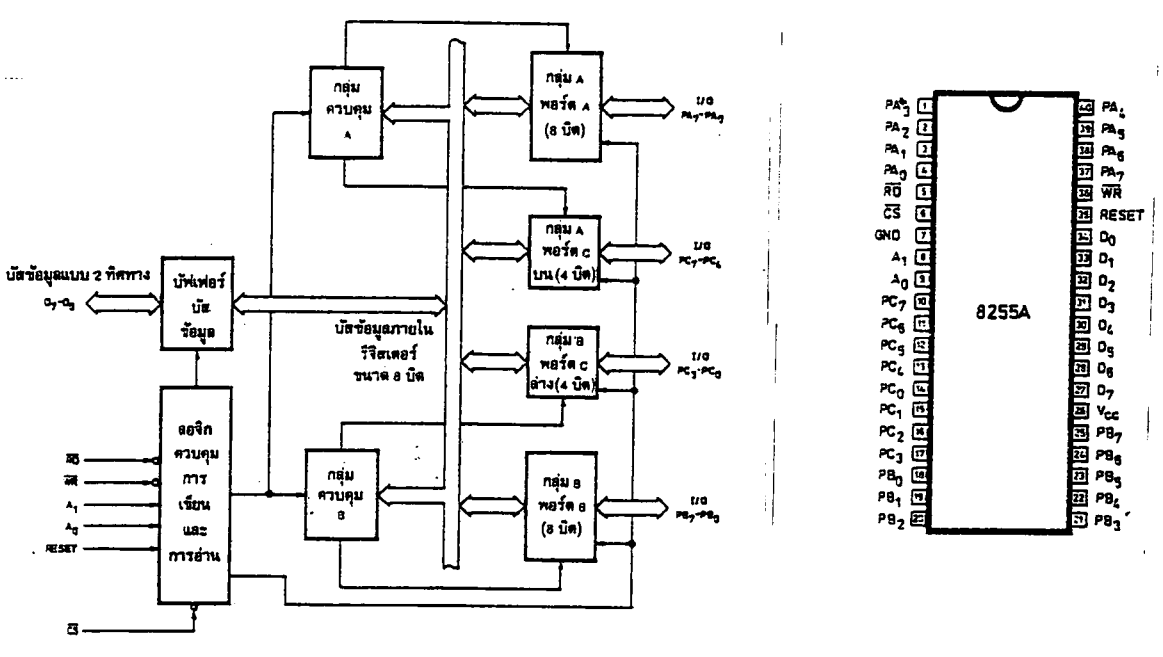
บทที่ 3

ชิพอินพุตเอาท์พุต

ชิพไอซี 8255

ลักษณะทั่วไป

8255 เป็นชิพไอซีที่มี 40 ขา ทำหน้าที่เป็นพอร์ท ซึ่งภายในชิพจะมีทั้งหมด 3 พอร์ท คือ พอร์ท A พอร์ท B และพอร์ท C โดยที่พอร์ท A และพอร์ท B จะเป็นพอร์ทขนาด 8 บิท ส่วนพอร์ท C จะแบ่งเป็นพอร์ท C บนและพอร์ท C ล่าง พอร์ทละ 4 บิท การทำงานของทุกพอร์ทเป็นได้ทั้งพอร์ทอินพุท และพอร์ทเอาท์พุท ขึ้นอยู่กับการโปรแกรมค่าจากไมโครโปรเซสเซอร์ การจัดขาของไอซีและแผงผังวงจรภายในแสดงดังรูปที่ 3.1



รูปที่ 3.1 แสดงแผนผังวงจรภายในและการจัดขาของไอซี 8255

ฟังก์ชันการทำงานของขาสัญญาณต่าง ๆ

ขาทั้ง 40 ขาของ 8255 ประกอบด้วย

D_0-D_7 เป็นขาที่ต่อเข้ากับบัสข้อมูลของซีพียูเพื่อใช้สำหรับส่งข้อมูลระหว่าง 8255 กับซีพียู

A_0-A_1 ขาแอดเดรส ใช้ในการกำหนดพอร์ทว่า พอร์ท A พอร์ท B พอร์ท C และพอร์ทควบคุมมีหมายเลขพอร์ทเป็นอะไร โดยค่า A_0 และ A_1 มีความสัมพันธ์กับพอร์ทต่าง ๆ ดังนี้

ตารางที่ 3.1 แสดงความสัมพันธ์ของแอดเดรส A0 และ A1 กับพอร์ทต่าง ๆ

A_1	A_0	พอร์ท
0	0	A
0	1	B
1	0	C
1	1	ควบคุม

$\overline{CS}$ เป็นขาเลือกชิพ ซึ่งเป็นขาอินพุทรับสัญญาณจากภายนอก ซึ่งถ้าขานี้เป็นลอจิก "0" จะทำให้ 8255 ต่อเข้ากับระบบบัสของไมโครโปรเซสเซอร์ ทำให้ไมโครโปรเซสเซอร์อ่านหรือเขียนข้อมูลกับพอร์ทต่าง ๆ ได้

$\overline{RD}$ เป็นสัญญาณอินพุทที่ส่งมาจากซีพียู ถ้าสัญญาณนี้เป็นลอจิก "0" ในขณะที่สัญญาณ $\overline{CS}$ เป็นลอจิก "0" ด้วย ซีพียูก็สามารถอ่านข้อมูลเข้าไปจากพอร์ทอินพุทของ 8255 ได้

$\overline{WR}$ เป็นสัญญาณการเขียนที่ส่งมาจากซีพียู ถ้าสัญญาณนี้เป็นลอจิก "0" ในขณะที่สัญญาณ CS เป็นลอจิก "0" ด้วย ซีพียูก็สามารเขียนข้อมูลผ่านออกไปที่พอร์ทเอาท์พุทของ 8255 ได้

$\overline{RESET}$ เป็นสัญญาณจากภายนอก เพื่อทำการเคลียร์สถานะต่าง ๆ ของ 8255

เมื่อ 8255 ถูกรีเซ็ต พอร์ททุกพอร์ทจะมีสถานะเริ่มต้นเป็นพอร์ทอินพุท

PA_0-PA_7 เป็นสายส่งสัญญาณที่เป็นพอร์ทของ 8255 มีชื่อว่า พอร์ท A

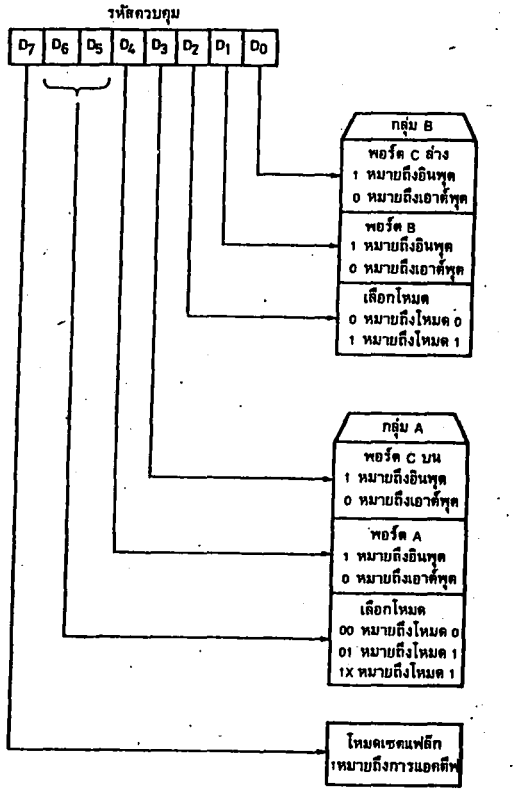
PB_0-PB_7 เป็นสายส่งสัญญาณที่เป็นพอร์ทของ 8255 มีชื่อว่า พอร์ท B

PC_0-PC_7 เป็นสายส่งสัญญาณที่เป็นพอร์ทของ 8255 มีชื่อว่า พอร์ท C

การโปรแกรมค่า 8255

ในการเริ่มต้นใช้งาน 8255 ต้องมีการโปรแกรมค่า หรือส่งรหัสควบคุม (Control Word) ไปยังพอร์ทควบคุมของ 8255 เสียก่อน โดยขณะที่เราใช้งานพอร์ทควบคุม ค่า A_1 และ A_0 จะเป็น 1 ทั้งคู่ ซึ่งรหัสควบคุมที่ส่งไปจะแสดงถึงโหมดการทำงานและหน้าที่ของพอร์ทต่าง ๆ ว่าเป็นพอร์ทอินพุทหรือพอร์ทเอาท์พุท

ความหมายของบิทต่าง ๆ ในรหัสควบคุม (Control Word) แสดงดังรูปที่ 3.2



รูปที่ 3.2 แสดงความหมายของบิทต่าง ๆ ในรหัสควบคุม

บิท D_7 ถ้าเป็น 1 จะแสดงถึงการกำหนดให้ 8255 ทราบค่าที่โปรแกรมในบิทต่าง ๆ โดยทำงานเป็นพอร์ทอินพุทและเอาท์พุทตามค่าที่โปรแกรม

บิท D_6 และ D_5 เป็นการเลือกโหมดในการทำงานของพอร์ท A ซึ่งมี 3 โหมด

บิท D_4 ถ้าเป็น 1 หมายถึงกำหนดพอร์ท A เป็นพอร์ทอินพุท ถ้ามีค่าเป็น 0 หมายถึง กำหนดพอร์ท A เป็นพอร์ทเอาท์พุท

บิท D_3 ถ้าเป็น 1 หมายถึง กำหนดให้พอร์ท C บน เป็นพอร์ทอินพุท ถ้ามีค่าเป็น 0 หมายถึง กำหนดให้พอร์ท C บน เป็นพอร์ทเอาท์พุท

บิท D_2 เป็นบิทที่แสดงการเลือกโหมดการทำงานของพอร์ท B ซึ่งมี 2 โหมด

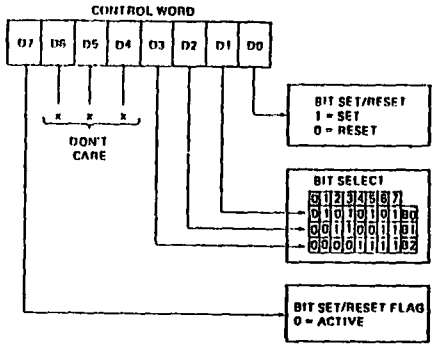
บิท D_1 ถ้าเป็น 1 หมายถึงกำหนดพอร์ท B เป็นพอร์ทอินพุท ถ้ามีค่าเป็น 0 หมายถึง กำหนดพอร์ท B เป็นพอร์ทเอาท์พุท

บิท D_0 ถ้าเป็น 1 หมายถึงกำหนดพอร์ท C ล่าง เป็นพอร์ทอินพุท ถ้ามีค่าเป็น 0 ก็ หมายถึงกำหนดพอร์ท C ล่างเป็นพอร์ทเอาท์พุท

<u>ตัวอย่าง</u> ต้องการให้พอร์ท A และพอร์ท C ล่าง ทำงานเป็นอินพุทพอร์ทในโหมด 0

จะได้ รหัสควบคุม (Control Word) คือ 10010001B = 91H

ในกรณีที่รหัสควบคุมมีค่าในบิทที่ 7 เป็น 0 จะได้รูปแบบการทำงานของ 8255 เป็นดังรูปที่ 3.3



รูปที่ 3.3 แสดงการเซ็ตค่าต่าง ๆ ของ 8255

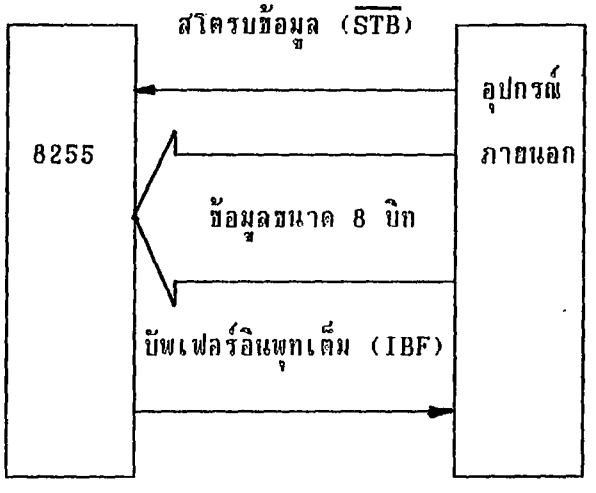
ซึ่งการใช้งานแบบนี้ใช้ได้เฉพาะพอร์ท C เท่านั้น และต้องทำงานในโหมด 0 ใช้ในการเซ็ตหรือรีเซ็ตบิทที่ต้องการ เพื่อปิดเปิดอุปกรณ์ได้อย่างอิสระ

โหมดการทำงาน

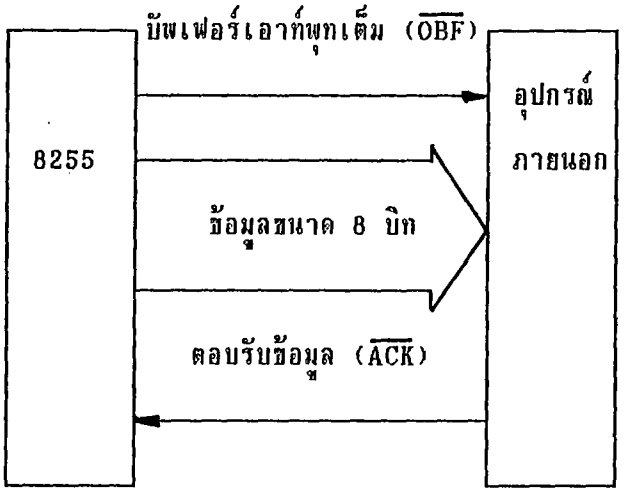
โหมดการทำงานของ 8255 แบ่งได้เป็น 3 โหมด คือ

1. โหมด 0 เป็นโหมดที่กำหนดให้พอร์ททุกพอร์ทของ 8255 เป็นพอร์ทอินพุทเอาท์พุททั่วไป ดังที่กล่าวมาข้างต้น

2. โหมด 1 เป็นโหมดที่ทำให้พอร์ทอินพุทเอาท์พุทของ 8255 มีการตรวจสอบสัญญาณ (Hand Shaking) โดยใช้พอร์ท A และพอร์ท B เป็นพอร์ทอินพุทเอาท์พุท และใช้พอร์ท C บนเป็นสัญญาณ Hand Shake ของพอร์ท A ส่วนพอร์ท C ล่างเป็นสัญญาณ Hand Shake ของพอร์ท B โดยการใช้งานโหมด 1 แสดงดังรูปที่ 3.4a และรูปที่ 3.4b

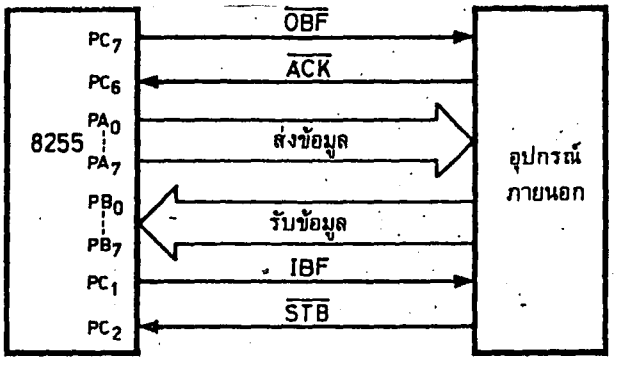


รูปที่ 3.4a แสดงโครงสร้างตัวตรวจสอบสัญญาณของพอร์ทอินพุท



รูปที่ 3.4b แสดงโครงสร้างตัวตรวจสอบสัญญาณของพอร์ทเอาท์พุท

ในการต่อวงจรเพื่อใช้งาน 8255 ในโหมด 1 เป็นดังรูปที่ 3.5



รูปที่ 3.5 แสดงวงจรการต่อ 8255 ในโหมด 1

เมื่อโปรแกรม 8255 ให้ทำงานในโหมด 1 แล้ว 8255 จะให้พอร์ท C เป็นสัญญาณควบคุม โดยแต่ละบิทของพอร์ท C แทนสัญญาณต่าง ๆ ดังตารางที่ 3.2

ตารางที่ 3.2 หน้าที่ของสัญญาณต่าง ๆ ของพอร์ท C ในการทำงานเป็นตัวตรวจสอบสัญญาณเมื่อ 8255 ทำงานในโหมด 1

ขา	กรณีอินพุท	กรณีเอาท์พุท
PC_0	$INTR_B$	$INTR_B$
PC_1	IBF_B	$\overline{OBF}_B$
PC_2	$\overline{STB}_B$	$\overline{ACK}_B$
PC_3	$INTR_A$	$INTR_A$
PC_4	$\overline{STB}_A$	I/O
PC_5	I/O	$\overline{ACK}_A$
PC_6	I/O	$\overline{OBF}_A$

จากตารางพบว่า ในกรณีอินพุท PC_6 และ PC_7 สามารถโปรแกรมให้เป็นอินพุทหรือเอาท์พุทอิสระได้ และในการใช้โหมด 1 นี้สามารถส่งสัญญาณไปอินเทอร์รัพท์ซีพียูได้อีกด้วย โดยต้องทำการอีนาเบิลอินเทอร์รัพท์เสียก่อน โดยการส่ง "1" ไปที่บิทอินเทอร์รัพท์ ดังตารางที่ 3.3

ตารางที่ 3.3 แสดงการอีนาเบิลอินเทอร์รัพท์โดยการ Set/Reset พอร์ท C

พอร์ท	กำหนดเป็น	สัญญาณ INT	ทำการ Set/Reset ที่
A	อินพุท	$INTE_A$	PC_4
A	เอาท์พุท	$INTE_A$	PC_6
B	อินพุท	$INTE_B$	PC_2
B	เอาท์พุท	$INTE_B$	PC_2

โดยในการ Set/Reset นี้ให้ทำการ Set/Reset ไปที่รหัสควบคุมเหมือนกับการใช้งาน Set/Reset พอร์ท C กรณีที่บิทที่ 7 ของรหัสควบคุม (Control Word) เป็น "0"

สัญญาณต่าง ๆ ในการทำ Hand Shake ของโหมด 1

กรณีอินพุท

IBF เป็นสัญญาณเอาท์พุท แอคทีฟที่ลอจิก "1" เป็นสัญญาณที่บอกให้อุปกรณ์ภายนอกที่ต้องการส่งข้อมูลเข้ามาทราบว่า บัฟเฟอร์เต็ม ดังนั้นอย่าเพิ่งส่งข้อมูลเข้ามาอีก

$\overline{STB}$ เป็นสัญญาณอินพุท แอคทีฟที่ลอจิก "0" เป็นสัญญาณที่อุปกรณ์ภายนอกส่งมาเพื่อบอกให้ 8255 ทราบว่าได้ส่งข้อมูลมาให้แล้ว

INTR เป็นสัญญาณเอาท์พุท แอคทีฟที่ลอจิก "1" ส่งไปเพื่อบอกกับซีพียูว่าให้ซีพียูมาอ่านข้อมูลจากบัฟเฟอร์ของ 8255

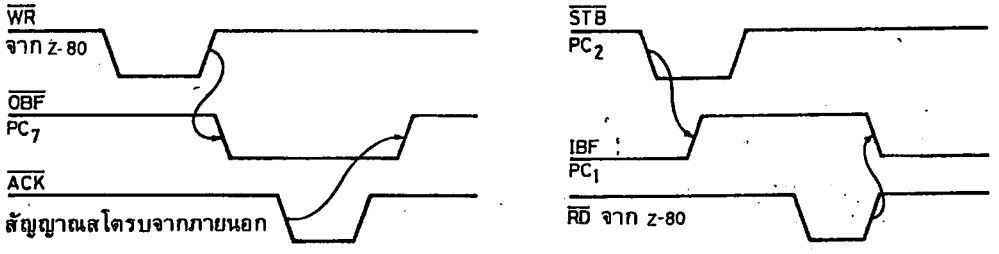
กรณีเอาท์พุท

$\overline{OBF}$ เป็นสัญญาณเอาท์พุท แอคทีฟที่ลอจิก "0" เป็นสัญญาณเพื่อส่งไปบอกกับอุปกรณ์ภายนอกว่าให้มาอ่านข้อมูลในบัฟเฟอร์ 8255 ซึ่งซีพียูส่งมาให้

$\overline{ACK}$ เป็นสัญญาณอินพุท แอคทีฟที่ลอจิก "0" เป็นสัญญาณตอบรับจากอุปกรณ์ภายนอกว่าได้อ่านข้อมูลจากบัฟเฟอร์ไปแล้ว

INTR เป็นสัญญาณเอาท์พุท แอคทีฟที่ลอจิก "1" เป็นสัญญาณเพื่อส่งไปบอกกับซีพียูให้ซีพียูส่งข้อมูลถัดไปออกมาได้

โดยแผนผังการทำงานของสัญญาณต่าง ๆ แสดงดังรูปที่ 3.6aและ 3.6b



ก. เมื่อเป็นพอร์ทเอาท์พุท ข. เมื่อเป็นพอร์ทอินพุท

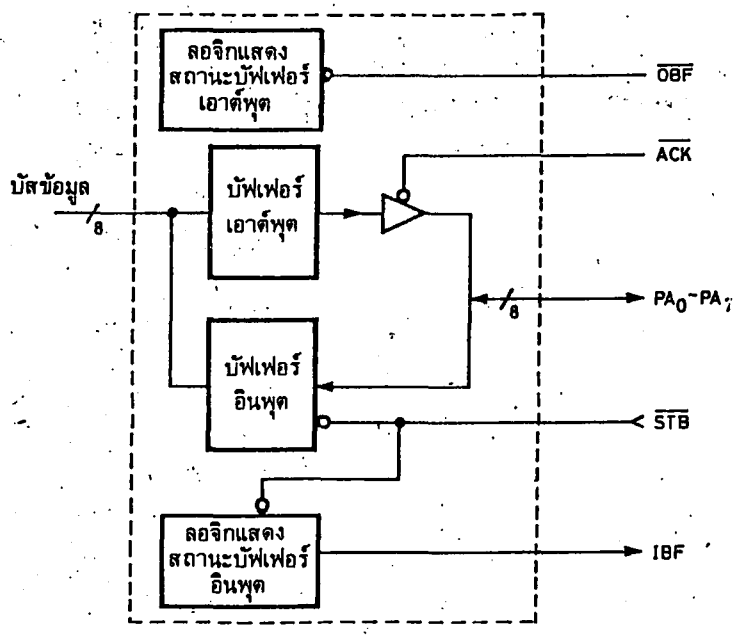
รูปที่ 3.6 แสดงแผนผังเวลาการรับและส่งข้อมูลโดยใช้ตัวตรวจสอบสัญญาณ

3. โหมด 2 ทำได้กับพอร์ท A เท่านั้น ในโหมดนี้จะใช้พอร์ท A เป็นพอร์ทแบบสองทิศทาง คือ สามารถเป็นได้ทั้งพอร์ทอินพุท และพอร์ทเอาท์พุท โครงสร้างของพอร์ท A ทั้งอินพุทและเอาท์พุท จะมีสัญญาณ Hand Shake ทั้งคู่ โดยพอร์ท C ทำหน้าที่เป็นสัญญาณ Hand Shake ซึ่งสัญญาณแต่ละขาของพอร์ท C เป็นดังตารางที่ 3.4

ตารางที่ 3.4 แสดงการใช้งานพอร์ท C เป็นสัญญาณ Hand Shake

พอร์ท C	ความหมาย
PC_0	I/O
PC_1	I/O
PC_2	I/O
PC_3	$INTR_A$
PC_4	$\overline{STB}_A$
PC_5	IBF_A
PC_6	$\overline{ACK}_A$
PC_7	$\overline{OBF}_A$

ลักษณะโครงสร้างของพอร์ท A ที่ทำงานแบบ 2 ทิศทาง แสดงดังรูปที่ 3.7



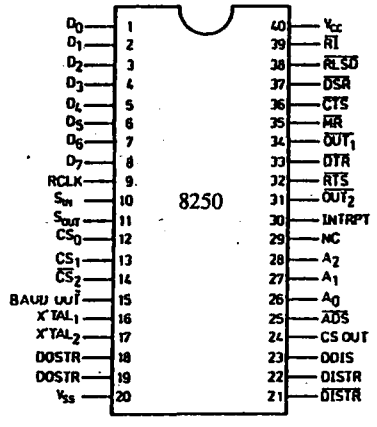
รูปที่ 3.7 แสดงโครงสร้างของพอร์ท A ที่ทำงานแบบพอร์ท 2 ทิศทาง

ลักษณะการ Hand Shake ทางอินพุทและเอาท์พุทของโหมด 2 มีหลักการเหมือนกับโหมด 1 และในการทำงานแบบอินเทอร์รัพท์ สามารถส่งอีนาเบิลอินเทอร์รัพท์ INTE ให้เป็น 1 โดยกรณีเอาท์พุทให้ Set/Reset ที่ PC_6 กรณีอินพุทให้ Set/Reset ที่ PC_4

ชิพไอซี 8250

ลักษณะทั่วไป

8250 เป็นชิพไอซีที่ใช้ในอะแดปเตอร์การสื่อสารแบบอะซิงโครนัส (Asynchronous Communication Adaptor) ในไมโครคอมพิวเตอร์ทั่วไป ซึ่ง 8250 เป็นชิพที่มีขนาด 40 ขา ทำหน้าที่ในการรับส่งข้อมูลแบบอนุกรม โดยสามารถโปรแกรมความเร็วในการรับส่งได้ตั้งแต่ 50 ถึง 9,600 Baud ลักษณะข้อมูลที่ส่งอาจมี 5 6 7 หรือ 8 บิท และบิทสิ้นสุดเป็น 1 3/2 หรือ 2 บิทก็ได้ ภายในตัวไอซีจะมีการจัดลำดับความสำคัญของอินเทอร์รัพท์ต่างๆ ไว้ นอกจากนี้ภายในชิพยังมีขาสัญญาณต่าง ๆ เพื่อใช้ในการติดต่อกับโมเด็มได้โดยตรง สำหรับการกำหนดการทำงานในรูปแบบต่าง ๆ ทำได้โดยโปรแกรมค่าเริ่มต้นให้กับรีจิสเตอร์ภายใน ลักษณะการจัดขาสัญญาณต่างๆ ของชิพแสดงดังรูปที่ 3.8



รูปที่ 3.8 แสดงการจัดขาของชิพไอซี 8250

ฟังก์ชันการทำงานของขาสัญญาณต่าง ๆ

1. สัญญาณอินพุท

CS0,CS1,$\overline{CS2}$ - Chip Select ขา 12,13 และ 14 ตามลำดับ เป็นขาสัญญาณในการเลือกชิพให้ทำงาน โดยชิพจะถูกเลือกให้ทำงานเมื่อขา CS0,CS1 เป็นลอจิก 1 และ $\overline{CS2}$ เป็นลอจิก 0 สัญญาณเลือกชิพจะได้รับการแลทช์ไว้ ในขณะที่สัญญาณ $\overline{ADS}$ มีสถานะเป็น 0 โดยการเลือกชิพมีไว้เพื่อให้ซีพียูติดต่อกับ 8250

DISTR,$\overline{DISTR}$ - Data Input Strobe ขา 22 และ 21 ตามลำดับ เมื่อสัญญาณที่ DISTR เป็น 1 และ $\overline{DISTR}$ เป็น 0 ในขณะที่สัญญาณเลือกชิพแอคทีฟ ซีพียูสามารถอ่านข้อมูลจากรีจิสเตอร์ภายใน 8250 ได้

DOSTR,$\overline{\text{DOSTR}}$ - Data Output Strobe ขา 19 และ 18 ตามลำดับเมื่อสัญญาณที่ DOSTR เป็น 1 และ $\overline{\text{DOSTR}}$ เป็น 0 ขณะที่สัญญาณเลือกชิพแอคทีฟ ซีพียูสามารถเขียนข้อมูลมายังรีจิสเตอร์ภายใน 8250 ได้

$\overline{\text{ADS}}$ - Address Strobe ขา 25 เมื่อสัญญาณมีค่าเป็น 0 จะทำให้มีการแลทช์ค่า A_0-A_2 เพื่อเลือกรีจิสเตอร์ภายใน โดยการเลือกรีจิสเตอร์จะทำขณะสัญญาณเลือกชิพแอคทีฟ

A_0,A_1,A_2 - Register Select ขา 26,27 และ 28 ตามลำดับ เป็นสัญญาณที่กำหนดแอดเดรสของรีจิสเตอร์ภายใน เพื่อให้ซีพียูสามารถอ่านหรือเขียนข้อมูลกับรีจิสเตอร์ภายใน 8250 ได้ ซึ่งการทำงานของ A_0-A_2 ในการติดต่อกับรีจิสเตอร์ภายในต่าง ๆ นี้ขึ้นกับบิท DLAB (Divisor latch access bit) ด้วยดังตารางที่ 3.5

ตารางที่ 3.5 แสดงการกำหนดค่ารีจิสเตอร์

DLAB	A_2	A_1	A_0	รีจิสเตอร์
0	0	0	0	บัฟเฟอร์สำหรับตัวรับข้อมูล (อ่าน) โฮลดิงรีจิสเตอร์สำหรับส่งข้อมูล (เขียน)
0	0	0	1	อีนาเบิลอินเทอร์รัพท์
x	0	1	0	กำหนดอินเทอร์รัพท์
x	0	1	1	ควบคุมสายสื่อสาร
x	1	0	0	ควบคุมโมเด็ม
x	1	0	1	แสดงสถานะสายสื่อสาร
x	1	1	0	แสดงสถานะโมเด็ม
x	1	1	1	ไม่ใช้
1	0	0	0	แลทช์ตัวหาร (LSB)
1	0	0	1	แลทช์ตัวหาร (MSB)

MR - Master Reset ขา 35 เมื่อสัญญาณมีค่าเป็น 1 จะรีเซ็ตการทำงานของชิพ 8250 โดยทำให้ค่าต่าง ๆ ในรีจิสเตอร์ถูกเคลียร์หมด ยกเว้นบัฟเฟอร์ตัวรับ ตัวส่ง และแลทช์ตัวหาร ขณะทำการรีเซ็ตจะมีผลต่อสัญญาณเอาท์พุทด้วย ดังตารางที่ 3.6

ตารางที่ 3.6 แสดงผลของการรีเซ็ต

รีจิสเตอร์/สัญญาณ	การควบคุมการรีเซ็ต	สถานะเมื่อรีเซ็ต
รีจิสเตอร์อีนาเบิลอินเทอร์รัพท์	มาสเตอร์รีเซ็ต	ทุกบิทเป็น 0
รีจิสเตอร์กำหนดอินเทอร์รัพท์	มาสเตอร์รีเซ็ต	บิท 0 เป็น 1 บิท 1-2 เป็น 0 บิท 3-7 เป็น 0 ตลอด
รีจิสเตอร์ควบคุมสายสื่อสาร	มาสเตอร์รีเซ็ต	ทุกบิทเป็น 0
รีจิสเตอร์ควบคุมโมเด็ม	มาสเตอร์รีเซ็ต	ทุกบิทเป็น 0
รีจิสเตอร์แสดงสถานะสายสื่อสาร	มาสเตอร์รีเซ็ต	ทุกบิทเป็น 0 ยกเว้นบิท 5 และบิท 6 เป็น 1
รีจิสเตอร์แสดงสถานะโมเด็ม	มาสเตอร์รีเซ็ต	บิท 0-3 เป็น 0 บิท 4-7 เป็นสัญญาณอินพุท
SOUT	มาสเตอร์รีเซ็ต	เป็น 1
INTRPT (RCVR Error)	อ่าน LSR/มาสเตอร์รีเซ็ต	เป็น 0
INTRPT (RCVR Data Ready)	อ่าน RBR/มาสเตอร์รีเซ็ต	เป็น 0
INTRPT (สถานะโมเด็มเปลี่ยน)	อ่าน MSR/มาสเตอร์รีเซ็ต	เป็น 0
$\overline{\text{OUT2}}$	มาสเตอร์รีเซ็ต	เป็น 1
$\overline{\text{RTS}}$	มาสเตอร์รีเซ็ต	เป็น 1
$\overline{\text{DTR}}$	มาสเตอร์รีเซ็ต	เป็น 1
$\overline{\text{OUT1}}$	มาสเตอร์รีเซ็ต	เป็น 1

RCLK - Receiver clock ขา 9 เป็นขาที่รับสัญญาณนาฬิกาเพื่อกำหนด Baud-Rate โดยสัญญาณที่รับเข้ามาจะมีค่าเป็น 16 เท่าของ Baud Rate

SIN - Serial Input ขา 10 เป็นขารับข้อมูลอนุกรมจากสายส่งขณะทำการติดต่อสื่อสาร

$\overline{\text{CTS}}$ - Clear to Send ขา 36 เป็นสัญญาณที่ใช้ในการติดต่อกับโมเด็ม เงื่อนไขของสัญญาณจะเก็บไว้ในบิท 4 (บิท CTS) ของรีจิสเตอร์แสดงสถานะโมเด็ม ซึ่งซีพียูสามารถอ่านไปตรวจสอบได้

หมายเหตุ เมื่อไรก็ตามที่บิท CTS ของรีจิสเตอร์สถานะโมเด็มมีการเปลี่ยนสถานะไป

จะมีการอินเทอร์รัพท์เกิดขึ้น ถ้าบิทอีนาเบิลอินเทอร์รัพท์สถานะโมเด็มของรีจิสเตอร์อีนาเบิลอิน-เตอร์รัพท์ถูกเซ็ตเป็น 1

$\overline{DSR}$ - Data Set Ready ขา 37 เมื่อสัญญาณนี้เป็น 0 แสดงว่าโมเด็มพร้อมแล้วที่จะทำการส่งข้อมูล โดยที่ $\overline{DSR}$ เป็นสัญญาณอินพุทจากโมเด็ม ซึ่งเงื่อนไขของสัญญาณจะเก็บไว้ในบิทที่ 5 (บิท DSR) ของรีจิสเตอร์แสดงสถานะโมเด็ม ซึ่งซีพียูสามารถอ่านไปตรวจสอบได้

หมายเหตุ เมื่อไรก็ตามที่บิท DSR ของรีจิสเตอร์สถานะโมเด็มมีการเปลี่ยนสถานะไปจะมีการอินเทอร์รัพท์เกิดขึ้น ถ้าบิทอีนาเบิลอินเทอร์รัพท์สถานะโมเด็มของรีจิสเตอร์อีนาเบิลอิน-เตอร์รัพท์ถูกเซ็ตเป็น 1

$\overline{RLSD}$ - Received Line Signal Detect เมื่อสัญญาณนี้เป็น 0 หมายถึงโมเด็มสามารถตรวจสอบสัญญาณพาหะได้แล้ว โดยที่ $\overline{RLSD}$ เป็นสัญญาณอินพุทจากโมเด็ม ซึ่งเงื่อนไขของสัญญาณจะเก็บไว้ในบิทที่ 7 (บิท RLSD) ของรีจิสเตอร์แสดงสถานะโมเด็ม ซึ่งซีพียูสามารถอ่านเข้าไปตรวจสอบได้

หมายเหตุ เมื่อไรก็ตามที่บิท RLSD ของรีจิสเตอร์สถานะโมเด็มมีการเปลี่ยนสถานะไป จะมีการอินเทอร์รัพท์เกิดขึ้น ถ้าบิทอีนาเบิลอินเทอร์รัพท์สถานะโมเด็มของรีจิสเตอร์อีนาเบิลอินเทอร์รัพท์ถูกเซ็ตเป็น 1

$\overline{RI}$ - Ring Indicator ขา 39 เมื่อสัญญาณนี้เป็น 0 แสดงว่ามีเสียงกริ่งโทรศัพท์เข้ามา โดยที่ $\overline{RI}$ เป็นสัญญาณอินพุทจากโมเด็ม ซึ่งเงื่อนไขของสัญญาณจะเก็บไว้ในบิทที่ 6 (บิท RI) ของรีจิสเตอร์แสดงสถานะโมเด็ม ซึ่งซีพียูสามารถอ่านเข้าไปตรวจสอบได้

หมายเหตุ เมื่อไรก็ตามที่บิท RI ของรีจิสเตอร์แสดงสถานะโมเด็มมีการเปลี่ยนจาก 1 ไปเป็น 0 จะมีการอินเทอร์รัพท์เกิดขึ้น ถ้าบิทอีนาเบิลอินเทอร์รัพท์สถานะโมเด็มของรีจิสเตอร์อีนาเบิลอินเทอร์รัพท์ถูกเซ็ตเป็น 1

V_{cc} ขา 40 เป็นขาไฟเลี้ยง ต้องการไฟเลี้ยง +5Vdc

V_{ss} ขา 20 เป็นขากราวนด์

2. สัญญาณเอาท์พุท

$\overline{DTR}$ - Data Terminal Ready ขา 33 เมื่อสัญญาณนี้เป็น 0 แสดงว่าชิพ 8250 พร้อมที่จะรับส่งข้อมูล สัญญาณที่ขานี้ทำให้แอคทีฟได้โดยการโปรแกรมค่าลงในบิทที่ 0 (บิท DTR) ของรีจิสเตอร์ควบคุมโมเด็ม

$\overline{RTS}$ - Request to Send ขา 32 เมื่อสัญญาณนี้เป็น 0 แสดงว่าชิพ 8250 พร้อมที่จะข้อมูล สัญญาณที่ขานี้ทำให้แอคทีฟได้โดยการโปรแกรมค่าลงในบิท 1(บิท RST) ของรีจิสเตอร์ควบคุมโมเด็ม

$\overline{OUT1}$ - Output1 ขา 34 สัญญาณขานี้ทำให้แอคทีฟได้โดยการโปรแกรมค่าลงในบิทที่ 2 (บิท OUT1) ของรีจิสเตอร์ควบคุมโมเด็ม

$\overline{OUT2}$ - Output2 ขา 31 สัญญาณขานี้ทำให้แอคทีฟได้โดยการโปรแกรมค่าลงในบิทที่ 3 (บิท OUT2) ของรีจิสเตอร์ควบคุมโมเด็ม

CSOUT - Chip Select Out ขา 24 เมื่อสัญญาณนี้มีค่าเป็น 1 แสดงว่าชิพนี้ถูกเลือกโดยซีพียูทางขา CS0-CS2

DDIS - Driver Disable ขา 23 เมื่อสัญญาณนี้มีค่าเป็น 0 แสดงว่าซีพียูกำลังอ่านข้อมูลจากชิพ 8250 และเมื่อสัญญาณนี้มีค่าเป็น 1 จะดิสเอเบิลการรับส่งภายนอก

$\overline{BAUDOUT}$ - Baud Out ขา 15 เป็นขาสัญญาณที่มีความถี่เป็น 16 เท่าของ Baud-Rate

INTRPT - Interrupt ขา 30 เมื่อสัญญาณนี้มีค่าเป็น 1 จะมีการส่งสัญญาณอินเทอร์รัพท์ออกไปจาก 8250

SOUT - Serial Out ขา 11 เป็นขาที่ใช้ส่งข้อมูลอนุกรมออกไปยังสายสื่อสาร

3. สัญญาณอินพุทหรือเอาท์พุท

D_0-D_7 - Data Bus ขา 1-8 เป็นขาสัญญาณ 2 ทิศทาง ใช้ในการติดต่อกับซีพียู

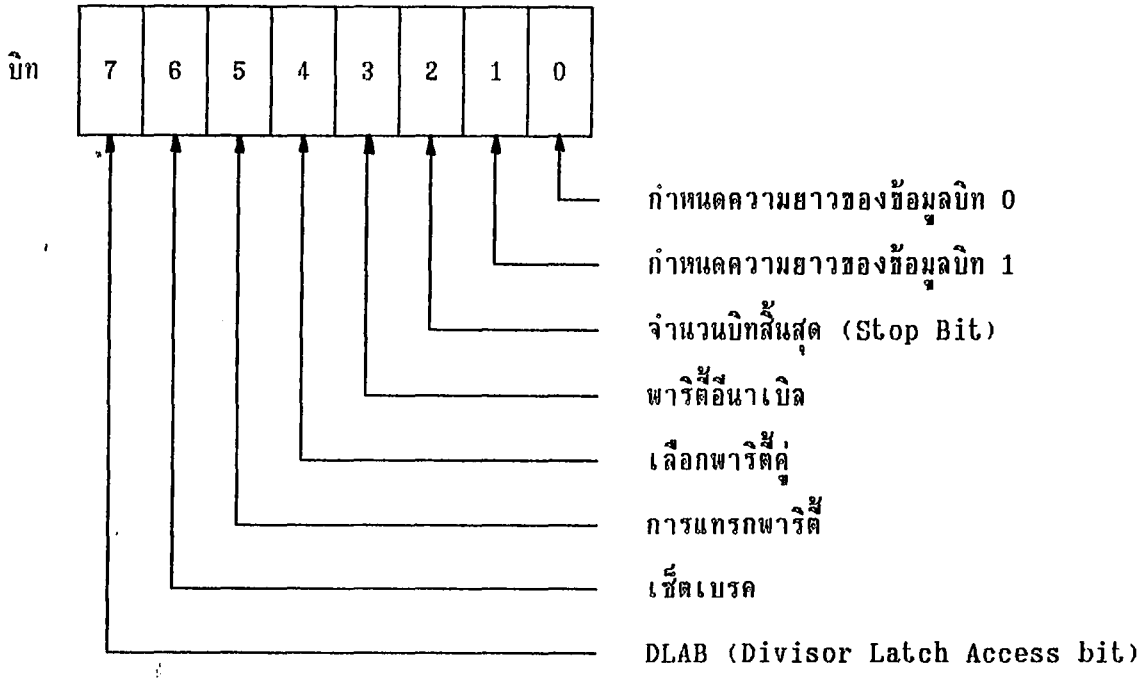
XTAL1,XTAL2 -External clock Input/Output ขา 16 และ 17 ตามลำดับ เป็นขาที่ใช้ต่อคริสตอลเพื่อสร้างสัญญาณนาฬิกา

การใช้งานรีจิสเตอร์ต่าง ๆ

1. รีจิสเตอร์ควบคุมสายสื่อสาร (Line Control Register) ใช้ควบคุมรูปแบบการสื่อสารแบบอะซิงโครนัส เป็นรีจิสเตอร์ขนาด 8 บิท ดังรูปที่ 3.9 โดยแต่ละบิทมีความหมายดังนี้

บิท 0 และ 1 ใช้ในการกำหนดความยาวของข้อมูลในการรับส่งว่าเป็นขนาดกี่บิทดังนี้

บิท 1	บิท 0	ความยาวข้อมูล
0	0	5 บิท
0	1	6 บิท
1	0	7 บิท
1	1	8 บิท



รูปที่ 3.9 แสดงค่าของบิทในรีจิสเตอร์ควบคุมสายสื่อสาร

บิท 2 เป็นบิทที่ใช้ในการกำหนดจำนวนบิทสิ้นสุด (Stop Bit) ในการรับส่งข้อมูลว่ามีความยาวกี่บิท

- ถ้าบิทนี้เป็น 0 แสดงว่าจำนวนบิทสิ้นสุด (Stop Bit) เท่ากับ 1 บิท
- ถ้าบิทนี้เป็น 1 ในกรณีส่งข้อมูลแบบ 5 บิท ความยาวของบิทสิ้นสุด (Stop Bit) จะเท่ากับ 1.5 บิท กรณีส่งข้อมูลแบบ 6,7,8 บิท ความยาวของบิทสิ้นสุดจะเท่ากับ 2 บิท

บิท 3 เป็นบิทที่ใช้อีนาเบิลให้มีการตรวจสอบพาริตี้ โดยถ้าบิทนี้เป็น 1 ก็จะทำให้มีการตรวจสอบพาริตี้ขณะทำการสื่อสาร

บิท 4 เป็นบิทที่ใช้ในการเลือกว่าต้องการพาริตี้คู่หรือพาริตี้คี่ในการสื่อสาร โดยถ้าบิท 3 ถูกอีนาเบิล เมื่อบิท 4 เป็น 0 จะหมายถึงพาริตี้คี่ แต่ถ้าบิท 4 เป็น 1 จะหมายถึงพาริตี้คู่

บิท 5 เป็น Stick Parity Bit โดยเมื่อบิท 3 เป็น 1 และบิท 5 เป็น 1 บิทพาริตี้ที่ถูกส่งไปโดยตัวส่งหรือที่รับโดยตัวรับจะเป็น 0 เมื่อบิท 4 เป็น 1 ในทำนองเดียวกันพาริตี้ที่ถูกส่งไปโดยตัวส่งหรือที่รับโดยตัวรับจะเป็น 1 เมื่อบิท 4 เป็น 0

บิท 6 เป็นบิทที่ควบคุมการเบรค (Break) เมื่อบิทนี้มีค่าเป็น 1 ส่วนของ Serial Output (SOUT) จะได้รับการกำหนดให้เป็น 0 ตลอด

บิท 7 เป็นบิท Divisor Latch Access Bit (DLAB) จะต้องเซ็ตเป็น 1 เมื่อต้องการเข้าถึงรีจิสเตอร์แลทช์ตัวหารและต้องเซ็ตเป็น 0 เมื่อต้องการเข้าถึงบัฟเฟอร์รับส่งข้อมูลและรีจิสเตอร์อีนาเบิลอินเทอร์รัพท์

2. <u>รีจิสเตอร์แลทช์ตัวหาร</u> (Divisor Latch Register)

ใช้ในการกำหนดตัวหารเพื่อกำหนดค่า Baud Rate ในการรับส่งข้อมูล ซึ่งโดยปกติ Baud Rate ได้รับการกำหนดเทียบกับสัญญาณนาฬิกา 1.8432 MHz โดยตัวหารมีค่าตั้งแต่ 1 ถึง 2^{16}-1 ค่าความถี่เอาท์พุทของตัวกำหนด Baud Rate มีค่าเท่ากับ 16 x Baud Rate ดังนั้น

ตัวหาร = ความถี่สัญญาณอินพุท/(Baud Rate x 16)

เช่น ความถี่สัญญาณอินพุท = 1.8432 MHz

ต้องการ Baud Rate = 2,400

จะได้ ตัวหาร = 1.8432/(2,400x16) = 48

เนื่องจากตัวหารในการกำหนด Baud Rate มีค่าสูงสุดถึง 2^{16}-1 ดังนั้นจึงต้องใช้รีจิสเตอร์ 8 บิท 2 ตัวเพื่อเก็บค่าตัวหาร โดยในการกำหนดหรือโปรแกรมค่าในรีจิสเตอร์แลทช์ตัวหารทั้ง 2 ตัวคือ LSB และ MSB ต้องให้ DLAB บิทในรีจิสเตอร์ควบคุมสายสื่อสารมีค่าเป็น 1 ค่าของตัวหาร เมื่อเทียบกับสัญญาณนาฬิกา 1.8432 MHz เป็นดังตารางที่ 3.7

ตารางที่ 3.7 ค่าตัวหารสำหรับการกำหนด Baud Rate

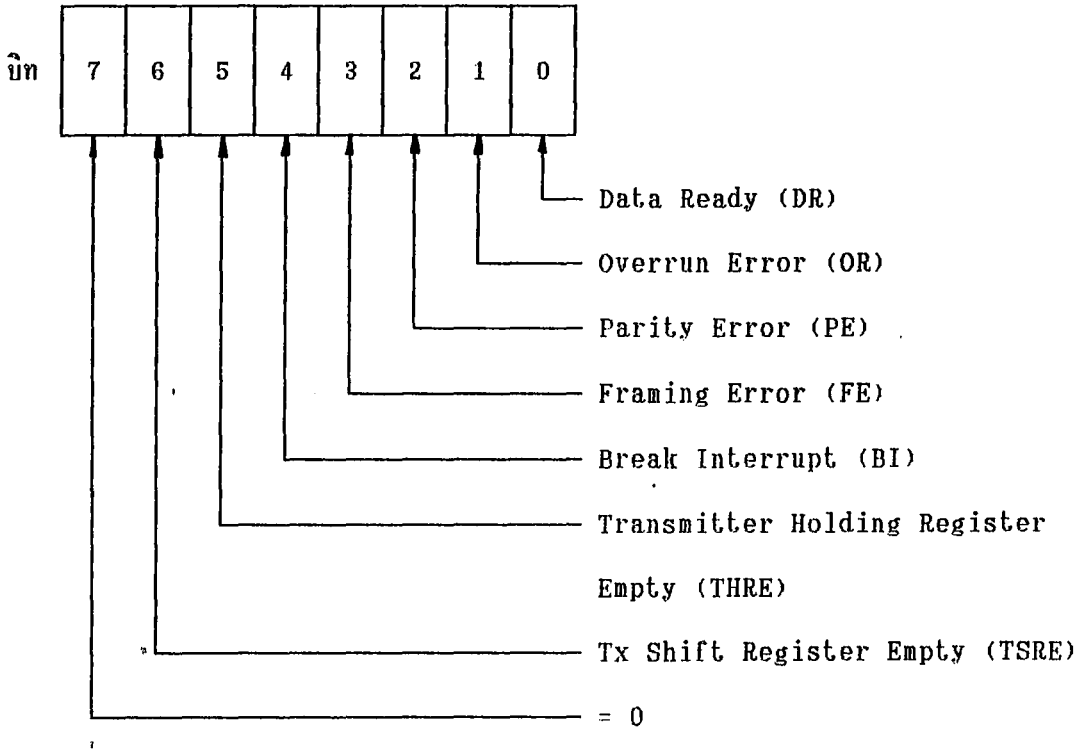
Baud Rate	ตัวหาร		ค่าผิดพลาด
	ฐานสิบ	ฐานสิบหก	
50	2304	900	-
75	1536	600	-
110	1047	417	0.026
134.5	857	359	0.058
150	768	300	-
300	384	180	-
600	192	0C0	-
1200	96	060	-
1800	64	040	-
2000	58	03A	0.69
2400	48	030	-

ตารางที่ 3.7 ค่าตัวหารสำหรับการกำหนด Baud Rate (ต่อ)

Baud Rate	ตัวหาร		ค่าผิดพลาด
	ฐานสิบ	ฐานสิบหก	
3600	32	020	-
4800	24	018	-
7200	16	010	-
9600	12	00C	-

3. <u>รีจิสเตอร์แสดงสถานะสายสื่อสาร</u> (Line Status Register)

เป็นรีจิสเตอร์ 8 บิท ให้ข้อมูลแก่ซีพียูเกี่ยวกับการสื่อสารข้อมูลในสายสื่อสาร โดยค่าของบิทแต่ละบิทเป็นดังนี้



รูปที่ 3.10 แสดงค่าของบิทต่าง ๆ ในรีจิสเตอร์แสดงสถานะสายสื่อสาร

บิท 0 บิทนี้เป็นบิทที่บอกสถานะการรับข้อมูล โดยถูกเซ็ตเป็น 1 เมื่อมีข้อมูลเข้ามาในบัฟเฟอร์รับข้อมูลครบทุกบิทแล้ว บิทนี้จะได้รับการรีเซ็ตให้เป็น 0 หลังจากที่ซีพียูได้อ่านข้อมูลใน

บัพเฟอร์ไปแล้ว หรืออาจให้ซีพียูเขียนลอจิก 0 ไปที่บิทนี้เลยก็ได้

บิท 1 บิทนี้แสดงถึง Overrun Error โดยถูกเซ็ตเป็น 1 เมื่อมี Overrun Error เกิดขึ้น คือขณะที่มีข้อมูลในบัพเฟอร์รับข้อมูล แต่ซีพียูยังไม่ได้อ่านไป แล้วมีข้อมูลชุดใหม่มาทับข้อมูลชุดเก่าในบัพเฟอร์นี้ ทำให้ข้อมูลเก่าสูญเสียไป บิทนี้จะได้รับการรีเซ็ตให้เป็น 0 โดยซีพียูเมื่อซีพียูอ่านค่าในรีจิสเตอร์แสดงสถานะสายสื่อสาร

บิท 2 บิทนี้จะแสดงถึง Parity Error โดยถูกเซ็ตเป็น 1 เมื่อมี Parity Error เกิดขึ้น คือถ้ามีการตรวจสอบบิทพาริตี้แล้วไม่เป็นไปตามที่กำหนด บิทนี้จะได้รับการรีเซ็ตให้เป็น 0 โดยซีพียูเมื่อซีพียูอ่านค่าในรีจิสเตอร์แสดงสถานะสายสื่อสาร

บิท 3 บิทนี้แสดงถึง Framing Error โดยถูกเซ็ตเป็น 1 เมื่อมี Framing Error เกิดขึ้น คือ เมื่อข้อมูลที่รับเข้ามาไม่มีบิทสิ้นสุด (Stop Bit) บิทนี้จะได้รับการรีเซ็ตให้เป็น 0 โดยซีพียูเมื่อซีพียูอ่านค่าในรีจิสเตอร์แสดงสถานะสายสื่อสาร

บิท 4 บิทนี้จะแสดงถึง Break Interrupt โดยถูกเซ็ตเป็น 1 เมื่อมีการรับข้อมูลอินพุทเป็น 0 เป็นเวลายาวนานกว่าเวิร์ดของการสื่อสาร

บิท 5 เป็นบิทที่บอกว่า ชิพ 8250 พร้อมที่จะรับข้อมูลตัวใหม่เพื่อที่จะส่งออกไปยังสายสื่อสาร ดังนั้นเมื่อ 8250 พร้อมที่จะรับข้อมูลแล้ว บิทนี้จะถูกเซ็ตเป็น 1 โดยที่บิทนี้สามารถสร้างอินเทอร์รัพท์เพื่อไปบอกกับซีพียูได้ เมื่อบิทอีนาเบิลอินเทอร์รัพท์โฮลดิงรีจิสเตอร์ว่างของรีจิสเตอร์อีนาเบิลอินเทอร์รัพท์ถูกเซ็ตเป็น 1 และบิท 5 จะถูกรีเซ็ตเป็น 0 เมื่อซีพียูเขียนข้อมูลชุดใหม่มายังโฮลดิงรีจิสเตอร์สำหรับส่งข้อมูล (Transmitter Holding Register)

บิท 6 เป็นบิทที่บอกว่าชิฟท์รีจิสเตอร์ว่างเปล่า บิทนี้จะได้รับการเซ็ตเป็น 1 เมื่อไม่มีข้อมูลในชิฟท์รีจิสเตอร์ และบิทนี้จะถูกรีเซ็ตให้เป็น 0 เมื่อมีการย้ายข้อมูลจากโฮลดิงรีจิสเตอร์สำหรับส่งข้อมูล (Transmitter Holding Register) มายังชิฟท์รีจิสเตอร์สำหรับส่งข้อมูล (Transmitter Shift Register)

บิท 7 บิทนี้จะเป็น 0 ตลอด

4. <u>รีจิสเตอร์กำหนดอินเทอร์รัพท์</u> (Interrupt Identification Register) ไอซี 8250 มีความสามารถในการส่งอินเทอร์รัพท์ภายในชิพ เพื่อให้การทำงานระหว่างตัวมันกับซีพียูเป็นไปอย่างมีประสิทธิภาพ โดยที่ชิพ 8250 กำหนดความสำคัญของอินเทอร์รัพท์ไว้ 4 ระดับ

-ระดับแรก อินเทอร์รัพท์จากสถานะการรับข้อมูลจากสายสื่อสาร (Receiver Line Status)

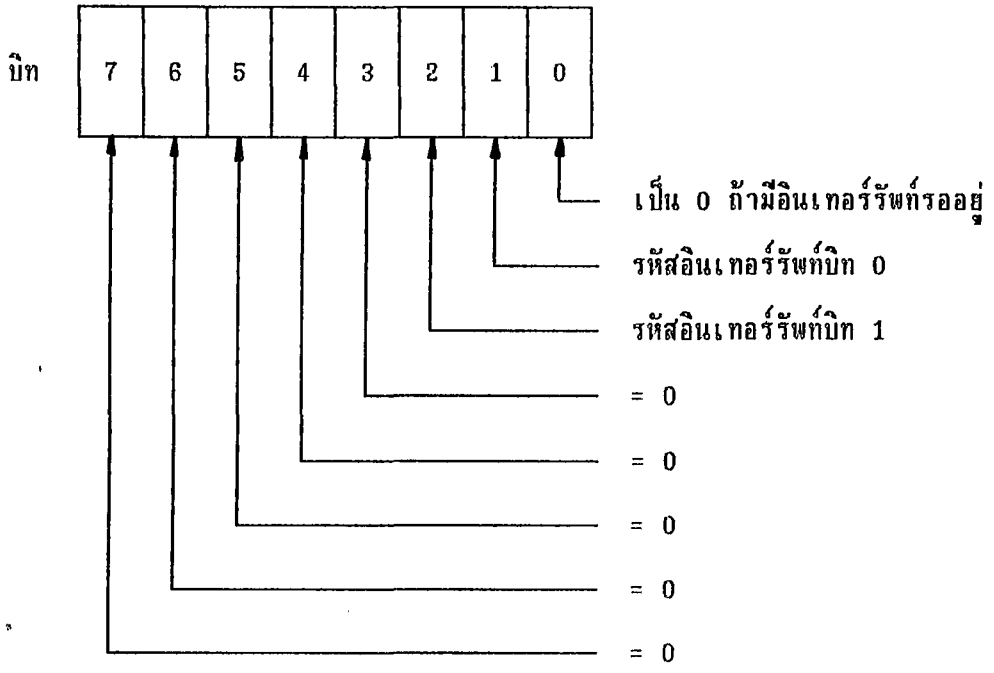
-ระดับที่สอง อินเทอร์รัพท์จากการพร้อมรับข้อมูล (Received Data Ready)

-ระดับที่สาม อินเทอร์รัพท์จากโฮลดิงรีจิสเตอร์ว่าง (Transmitter Holding

Register Empty)

-ระดับที่สี่ อินเทอร์รัพท์จากสถานะของโมเด็ม (Modem Status)

ในกรณีที่มีความต้องการอินเทอร์รัพท์หลายระดับพร้อม ๆ กัน ชิพ 8250 จะให้บริการอินเทอร์รัพท์ที่มีระดับความสำคัญมากก่อน แล้วจึงค่อยบริการอินเทอร์รัพท์ที่มีความสำคัญรองลงไป โดยจะเก็บสถานะการอินเทอร์รัพท์ไว้ในรีจิสเตอร์กำหนดอินเทอร์รัพท์ ซึ่งเป็นรีจิสเตอร์ขนาด 8 บิท ดังรูปที่ 3.11 ซึ่งแต่ละบิทมีความหมายดังนี้



รูปที่ 3.11 แสดงค่าของบิทในรีจิสเตอร์กำหนดอินเทอร์รัพท์

บิท 0 เป็นบิทที่แสดงว่ามีการอินเทอร์รัพท์เกิดขึ้นหรือไม่ ถ้าบิทนี้เป็น 0 แสดงว่ามีอินเทอร์รัพท์ แต่ถ้าบิทนี้เป็น 1 แสดงว่าไม่มีอินเทอร์รัพท์เกิดขึ้น ซีพียูสามารถตรวจสอบได้โดยวิธีการโพลลิ่ง (Polling)

บิท 1 และ 2 เป็นบิทที่ใช้บอกว่าการอินเทอร์รัพท์ที่เกิดขึ้นนั้นมาจากอินเทอร์รัพท์ตามฟังก์ชันใด ดังนี้

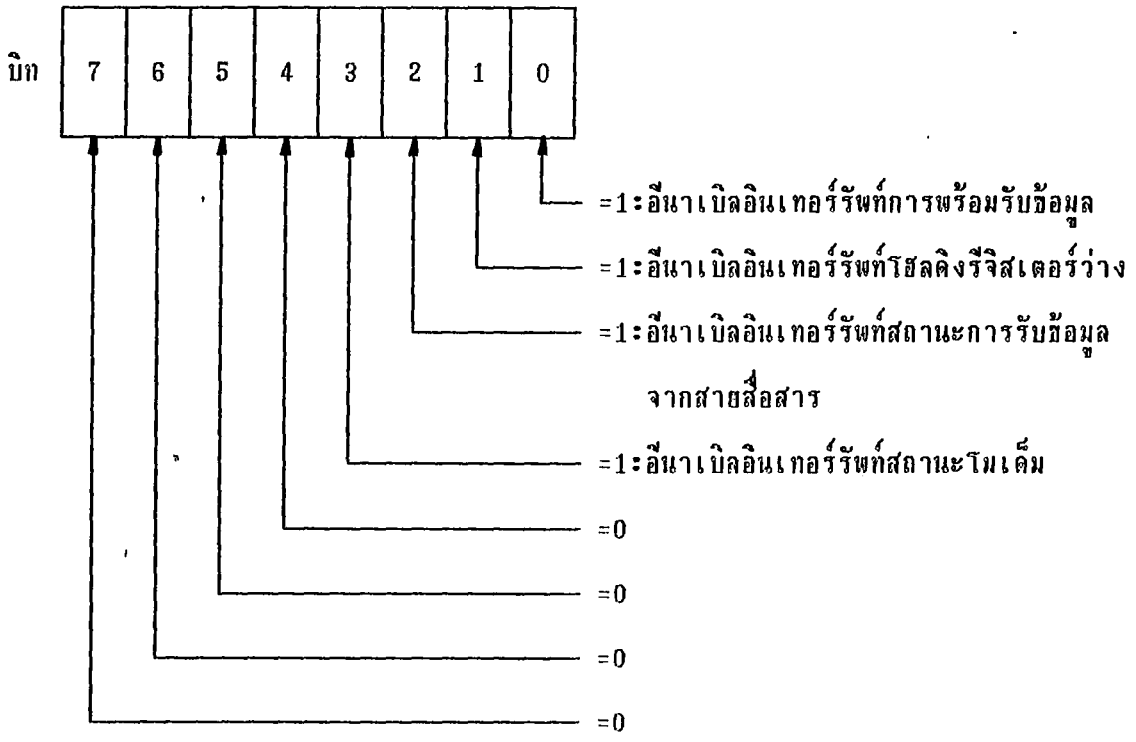
ตารางที่ 3.8 ฟังก์ชันควบคุมการอินเทอร์รัพท์ (Interrupt Function Control)

บิท2	บิท1	บิท0	ระดับความสำคัญ	ชนิดของอินเทอร์รัพท์	แหล่งกำเนิดอินเทอร์รัพท์	การรีเซ็ต
0	0	1	-	ไม่เกิด	ไม่มี	-
1	1	0	สูงสุด	สถานะการรับข้อมูลจากสายสื่อสาร	Overrun Error Parity Error Framing Error Break Interrupt	อ่านข้อมูลจากรีจิสเตอร์สถานะสายสื่อสาร
1	0	0	ที่สอง	การพร้อมรับข้อมูล	มีข้อมูลที่บัพเฟอร์ตัวรับ	อ่านข้อมูลจากบัพเฟอร์ตัวรับ
0	1	0	ที่สาม	โฮลดิงรีจิสเตอร์ว่าง	โฮลดิงรีจิสเตอร์สำหรับส่งข้อมูลว่าง	อ่านรีจิสเตอร์กำหนดอินเทอร์รัพท์หรือเขียนข้อมูลไปยังโฮลดิงรีจิสเตอร์สำหรับส่งข้อมูล
0	0	0	ที่สี่	สถานะโมเด็ม	CTS,DSR,RI,RLSD	อ่านรีจิสเตอร์แสดงสถานะโมเด็ม

บิท 3-7 มีค่าเป็น 0 ตลอดเวลา

5. <u>รีจิสเตอร์อีนาเบิลอินเทอร์รัพท์</u> (Interrupt Enable Register)

เป็นรีจิสเตอร์ขนาด 8 บิท ใช้ในการอีนาเบิลอินเทอร์รัพท์ทั้ง 4 แบบ โดยเมื่อมีการอินเทอร์รัพท์เกิดขึ้นไม่ว่าจะเป็นอินเทอร์รัพท์จากแหล่งกำเนิดตัวไหน ก็จะมีสัญญาณอินเทอร์รัพท์ไปบอกซีพียูโดยผ่านทางขา INTRPT ดังรูปที่ 13.12 ความหมายของแต่ละบิทของรีจิสเตอร์อีนาเบิลอินเทอร์รัพท์เป็นดังนี้



รูปที่ 3.12 แสดงค่าของบิทในรีจิสเตอร์อีนาเบิลอินเทอร์รัพท์

บิท 0 ใช้ในการอีนาเบิลอินเทอร์รัพท์การพร้อมรับข้อมูล โดยเซ็ตเป็น 1

บิท 1 ใช้ในการอีนาเบิลอินเทอร์รัพท์โฮลดิงรีจิสเตอร์ว่าง โดยเซ็ตเป็น 1

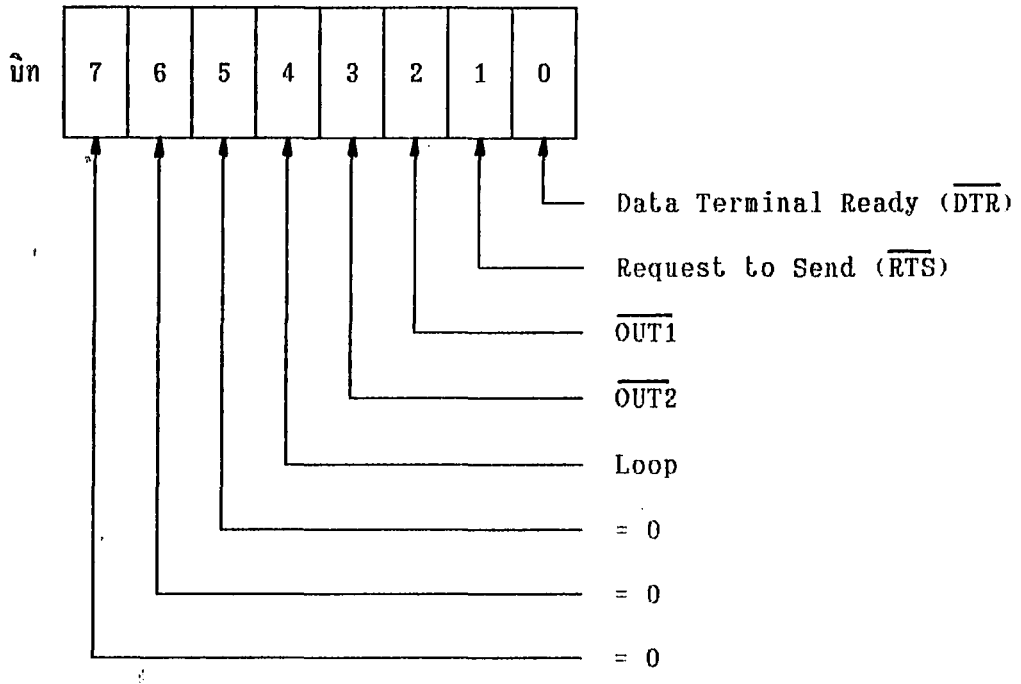
บิท 2 ใช้ในการอีนาเบิลอินเทอร์รัพท์สถานะการรับข้อมูลจากสายสื่อสาร โดยเซ็ตให้เป็น 1

บิท 3 ใช้ในการอีนาเบิลอินเทอร์รัพท์สถานะโมเด็ม โดยเซ็ตเป็น 1

บิท 4-7 ถูกกำหนดให้เป็น 0 เสมอ

6. <u>รีจิสเตอร์ควบคุมโมเด็ม</u> (Modem Control Register)

รีจิสเตอร์นี้มีไว้เพื่อให้ซีพียูส่งผ่านข้อมูลมาเก็บ เพื่อเป็นรหัสสำหรับควบคุมการทำงานของโมเด็ม ดังรูปที่ 3.13 โดยความหมายของแต่ละบิทเป็นดังนี้



รูปที่ 3.13 แสดงค่าของบิทต่าง ๆ ในรีจิสเตอร์ควบคุมโมเด็ม

บิท 0 บิทนี้ใช้ในการควบคุมสัญญาณ Data Terminal Ready (DTR) เมื่อบิทนี้มีค่าเป็น 1 เอาท์พุทของ DTR จะเป็น 0 และถ้าบิทนี้เป็น 0 เอาท์พุทของ DTR จะเป็น 1

บิท 1 บิทนี้ใช้ในการควบคุมสัญญาณ Request to Send(RTS) เมื่อบิทนี้มีค่าเป็น 1 เอาท์พุทของ RTS จะเป็น 0 และถ้าบิทนี้เป็น 0 เอาท์พุทของ RTS จะเป็น 1

บิท 2 บิทนี้ใช้ในการควบคุมสัญญาณ Output1(OUT1) เมื่อบิทนี้มีค่าเป็น 1 เอาท์พุทของ OUT1 จะเป็น 0 และถ้าบิทนี้เป็น 0 เอาท์พุทของ OUT1 จะเป็น 1

บิท 3 บิทนี้ใช้ในการควบคุมสัญญาณ Output2(OUT2) เมื่อบิทนี้มีค่าเป็น 1 เอาท์พุทของ OUT2 จะเป็น 0 และถ้าบิทนี้เป็น 0 เอาท์พุทของ OUT2 จะเป็น 1

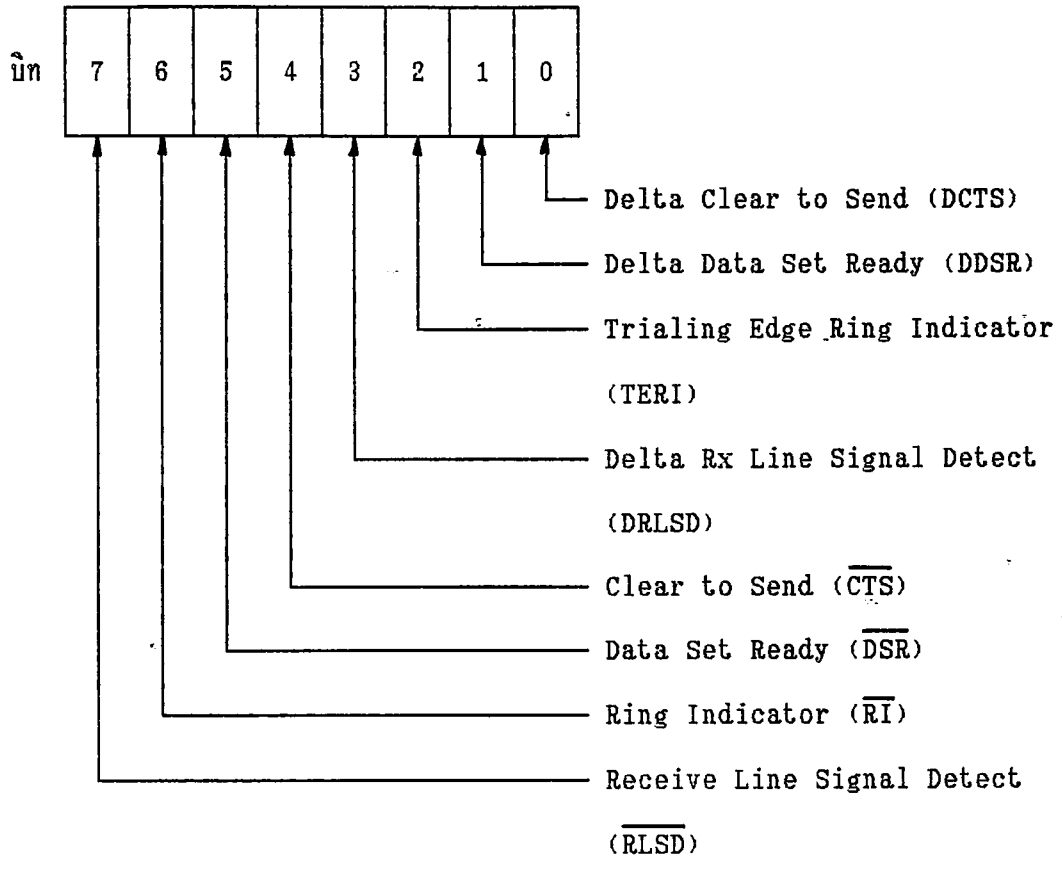
หมายเหตุ การใช้งานของขาสัญญาณ OUT1 และ OUT2 ขึ้นกับการออกแบบของผู้ใช้ว่าต้องการใช้งานลักษณะใด

บิท 4 ใช้สำหรับการกำหนดวงรอบสำหรับตรวจสอบการทำงานของ 8250

บิท 5-7 บิทเหล่านี้จะถูกกำหนดเป็น 0 เสมอ

7. รีจิสเตอร์แสดงสถานะโมเด็ม (Modem Status Register)

รีจิสเตอร์แสดงสถานะโมเด็มจะเป็นตัวที่รับสถานะจากโมเด็มมาเก็บไว้ เพื่อให้ซีพียูอ่านไปตรวจสอบดูได้ ดังรูปที่ 3.14 โดยความหมายแต่ละบิทเป็นดังนี้



รูปที่ 3.14 แสดงค่าของบิทต่างๆในรีจิสเตอร์แสดงสถานะโมเด็ม

บิท 0 บิทนี้ใช้สำหรับแสดงการเปลี่ยนแปลงของสัญญาณ $\overline{CTS}$ หลังจากถูกอ่านจากซีพียูครั้งสุดท้าย โดยเมื่อซีพียูอ่านรีจิสเตอร์สถานะโมเด็มแล้ว บิทนี้จะเป็น 0 และเมื่อไรก็ตามที่ขาสัญญาณ $\overline{CTS}$ มีการเปลี่ยนแปลง บิทนี้จะได้รับการเซ็ตให้เป็น 1

บิท 1 เช่นเดียวกับบิท 0 แต่ใช้แสดงการเปลี่ยนแปลงของสัญญาณ $\overline{DSR}$

บิท 2 บิทนี้ใช้สำหรับแสดงการเปลี่ยนแปลงของสัญญาณ $\overline{RI}$ โดยจะเปลี่ยนจาก 1 ไปเป็น 0 เมื่อไรก็ตามที่ขาสัญญาณ $\overline{RI}$ มีการเปลี่ยนแปลงจาก 1 ไปเป็น 0 บิทนี้จะได้รับการเซ็ตเป็น 1

บิท 3 เช่นเดียวกับบิท 0 แต่ใช้แสดงการเปลี่ยนแปลงของสัญญาณ $\overline{RLSD}$

บิท 4 บิทนี้จะเก็บค่าสัญญาณคอมพลีเมนต์ของขาสัญญาณ $\overline{CTS}$

บิท 5 บิทนี้จะเก็บค่าสัญญาณคอมพลีเมนต์ของขาสัญญาณ $\overline{DSR}$

บิท 6 บิทนี้จะเก็บค่าสัญญาณคอมพลีเมนต์ของขาสัญญาณ $\overline{RI}$

บิท 7 บิทนี้จะเก็บค่าสัญญาณคอมพลีเมนต์ของขาสัญญาณ $\overline{RLSD}$

8. รีจิสเตอร์บัฟเฟอร์สำหรับตัวส่งข้อมูล (Receiver Buffer Register)

เป็นรีจิสเตอร์ขนาด 8 บิท ใช้สำหรับรับข้อมูลที่มาจากสายสื่อสาร ในการที่จะอ่านข้อมูลจากรีจิสเตอร์ต้องทำให้บิท DLAB ในรีจิสเตอร์ควบคุมสายสื่อสารเป็น 0

9.รีจิสเตอร์โฮลดิงสำหรับส่งข้อมูล (Transmitter Holding Register)

เป็นรีจิสเตอร์ขนาด 8 บิท ใช้เป็นบัฟเฟอร์สำหรับส่งข้อมูล โดยจะทำหน้าที่รับข้อมูลจากซีพียู และส่งข้อมูลออกไปยังสายสื่อสาร ในการเขียนข้อมูลมายังรีจิสเตอร์ตัวนี้ ต้องทำให้บิท DLAB ในรีจิสเตอร์ควบคุมสายสื่อสารเป็น 0

6845 CRT Controller

การแสดงภาพให้เกิดขึ้นบนจอโทรทัศน์นั้น ต้องมีส่วนแสดงผลที่จะกำเนิดสัญญาณวีดีโอ HSYNC,VSYNC ซึ่งความถี่ที่ใช้ในประเทศไทยเป็นระบบ PAL มีความถี่ HSYNC =15.625 kHz ความถี่ VSYNC = 50 Hz และมีสัญญาณวีดีโอที่อ่านข้อมูลมาจาก Character Generator โดยในส่วนนี้จะมีไอซีเบอร์ MC6845 CRTC ของบริษัทโมโตโรล่าเป็นตัวจัดการทั้งหมด สัญญาณที่ได้ทั้ง 3 สัญญาณนี้ต้องนำมามอดดูเลทกันเป็นสัญญาณคอมโพสิทวีดีโอ (Composite Video) แต่การที่จะแสดงผลออกทางโทรทัศน์โดยผ่านทางช่องสายอากาศ จะต้องมีการสร้างสัญญาณพาหะเข้ามามอดดูเลทด้วย ดังนั้นในส่วนนี้จะใช้ไอซี LM1889 ของบริษัทเนชันแนล เซมิคอนดัคเตอร์

ลักษณะทั่วไป

6845 CRTC มีรีจิสเตอร์ภายใน 19 ตัว สำหรับการโปรแกรมค่าที่จะนำไปคำนวณเพื่อกำเนิดสัญญาณ มีขาสัญญาณทั้งหมด 40 ขา สามารถจัดหน้าที่ของแต่ละขาได้เป็น 3 หมวด คือ

1. สัญญาณที่ใช้ในการเชื่อมต่อ 6845 CRTC กับไมโครโปรเซสเซอร์
2. สัญญาณที่ใช้ในการเชื่อมต่อ 6845 CRTC กับหน่วยความจำแสดงผล และ Character Generator
3. สัญญาณที่ใช้ในการเชื่อมต่อ 6845 CRTC กับส่วนแสดงผล

รายละเอียดของขาต่าง ๆ แสดงดังตารางที่ 3.9

ตารางที่ 3.9 แสดงหน้าที่ของขาสัญญาณต่าง ๆ ของ 6845 CRTC

ชนิด	ขา	หน้าที่
เชื่อมต่อกับไมโครโปรเซสเซอร์	D_0-D_7	บัสข้อมูล
	CS	สัญญาณเลือกชิพ
	RS	สัญญาณเลือกรีจิสเตอร์
	R/$\overline{W}$	สัญญาณเลือกการเขียนและการอ่าน
	E	อีนาเบิล
	CLK/$\overline{RESET}$	สัญญาณนาฬิกาและสัญญาณรีเซ็ต
	V_{cc}-V_{ss}	แหล่งไฟเลี้ยง
เชื่อมต่อกับหน่วยความจำแสดงผล	MA_0-MA_{13}	แอดเดรสของหน่วยความจำแสดงผล
และ Character Generator	RA_0-RA_4	สัญญาณกำหนดแอดเดรสราสเตอร์
สัญญาณที่ติดต่อกับส่วนแสดงผล	HSYNC	สัญญาณซิงโครนัสแนวราบ
(Monitor)	VSYNC	สัญญาณซิงโครนัสแนวตั้ง
	DISPEN	อีนาเบิลการแสดงผล
	CURSOR	การแสดงเคอร์เซอร์
	LPSTB	ปากกาแสง

หน้าที่ของขาสัญญาณต่าง ๆ

1. ขาสัญญาณที่ติดต่อกับไมโครโปรเซสเซอร์

- Data Bus : D0-D7 เป็นบัสสองทิศทาง ทำการส่งข้อมูลระหว่าง 6845 CRTC กับไมโครโปรเซสเซอร์ ปกติสายข้อมูลนี้จะเป็น High Impedance

- Chip Select : CS เป็นสัญญาณเลือกชิพให้ตัว 6845 CRTC ทำงาน โดยให้มีการอ่านและการเขียนรีจิสเตอร์ภายใน ขาสัญญาณนี้แอคทีฟ Low

- Register Select : RS เป็นสัญญาณที่ใช้เลือกรีจิสเตอร์ ถ้า RS = 0 จะเป็นรีจิสเตอร์แอดเดรส ถ้า RS = 1 จะเป็นรีจิสเตอร์ข้อมูล

- Read/Write : R/$\overline{W}$ ขาสัญญาณนี้กำหนดให้รีจิสเตอร์ภายในอ่านหรือเขียน ถ้า R/$\overline{W}$ = 0 จะเป็นการเขียนข้อมูล ปกติสายข้อมูลนี้จะเป็น High Impedance

- Enable : E ขาสัญญาณนี้ใช้อนุญาติให้มีการส่งข้อมูลจากรีจิสเตอร์ภายในเข้า

ออกผ่านบัสข้อมูล โดยทำงานที่ขอบขาลง ปกติสายข้อมูลนี้จะเป็น High Impedance

- Clock : CLK สัญญาณนาฬิกาบอกเวลาใน 6845 CRTC

- Reset: $\overline{\text{RESET}}$ เป็นการเริ่มต้นการทำงานใหม่

ขาสัญญาณที่เชื่อมต่อกับหน่วยความจำแสดงผล

- Refresh Memory Address : MA0-MA13 เป็นสัญญาณกำหนดแอดเดรสของ 6845 CRTC เพื่อบอกว่าการแสดงผลใช้ข้อมูลในหน่วยความจำของแอดเดรสใด สามารถควบคุมหน่วยความจำได้ 16 กิโลไบท์

- Row Address : RA0-RA4 เป็นสัญญาณที่บอกว่า การแสดงผลนั้นอยู่ในแถวใดของตัวอักษร ใช้ส่งไปเพื่อควบคุม และเรียกข้อมูลจาก Character Generator ส่งไปยังส่วนแสดงผล (Monitor) สามารถกำหนดรายละเอียดได้ถึง $2^5 = 32$ เส้น

2. สัญญาณเชื่อมต่อกับหน่วยแสดงผล (Monitor)

- Horizontal and Vertical Synchronous : VSYNC กับ HSYNC เป็นขากำเนิดสัญญาณที่ต่อกับจอภาพโดยตรง หรือมีการร่วมกันกับสัญญาณวีดีโอ เป็นสัญญาณคอมโพสิทวีดีโอ (Composite Video) หรือผสมกับสัญญาณพาหะต่อเข้ากับช่องสายอากาศของโทรทัศน์ ทั้งนี้จะขึ้นอยู่กับชนิดของจอภาพด้วย

- Display Enable : DISPEN เป็นสัญญาณที่ต่อเพื่ออนุญาติให้มีการแสดงภาพในแอดเดรสที่ระบุไว้ ขาสัญญาณนี้แอคทีฟที่สถานะ High

- Cursor : CURSOR เป็นสัญญาณบอกตำแหน่งที่ซิงโครนัสกับแอดเดรสของ MA0-MA13 เพื่อกำหนดตำแหน่งของเคอร์เซอร์บนจอภาพ

- Light Pen Strobe : LPSTB เป็นสัญญาณบอกตำแหน่งของปากกาแสงที่อยู่บนจอภาพและเมื่อ LPSTB ทำงานพร้อมกับ RESET จะกำหนดโหมดการทำงานได้ดังตารางที่ 3.10

ตารางที่ 3.10 แสดงการกำหนดโหมดการทำงานต่าง ๆ

RESET	LPSTB	โหมดการทำงาน
0	0	Reset
0	1	Test
1	0	Normal
1	1	Normal

ในการทำงานของ 6845 CRTC นั้น สัญญาณ HSYNC กับ VSYNC จะมีความถี่และลักษณะของภาพตัวอักษรได้ตามต้องการ จะได้รับการกำหนดค่าในรีจิสเตอร์ โดยการโปรแกรมในรีจิสเตอร์ทั้ง 19 ตัว

รีจิสเตอร์ภายใน

การกำหนดค่าในรีจิสเตอร์ สามารถกำหนดได้โดยผ่านทางบัสข้อมูล โดยมีรีจิสเตอร์ AR (Address Register) เป็นตัวชี้แอดเดรส และอีก 18 ตัวเป็นส่วนที่รับการโปรแกรม

1. Address Registe : AR เป็นรีจิสเตอร์ขนาด 5 บิท สามารถเขียนได้อย่างเดียว โดยมีหน้าที่เก็บค่าแอดเดรสของรีจิสเตอร์อีก 18 ตัว เมื่อสัญญาณ RS และ CS มีสถานะเป็น Low ทั้งคู่ แอดเดรสรีจิสเตอร์จะถูกเลือก และเมื่อ CS มีสถานะเป็น Low ส่วน RS มีสถานะเป็น High รีจิสเตอร์ที่ถูกชี้ด้วยแอดเดรสรีจิสเตอร์ จะถูกเลือก

2. Timing Register : R0-R17 เป็นรีจิสเตอร์ที่กำหนดค่าโดยผู้ใช้ให้กับ 6845 CRTC นำไปคำนวณ เพื่อกำเนิดสัญญาณที่ใช้เชื่อมต่อกับจอภาพ สามารถแสดงได้ดังตารางที่ 3.11

ตารางที่ 3.11 ความหมายของรีจิสเตอร์ต่าง ๆ

	รีจิสเตอร์		เขียน (W) อ่าน (R)	จำนวนบิท
	หมายเลข	ชื่อ		
การกำหนดช่วงเวลาในแนวราบ	R0	ช่วงเวลาแนวราบ	W	8
	R1	จำนวนตัวอักษรต่อแถว	W	8
	R2	ตำแหน่ง HSYNC	W	8
	R3	ความกว้าง HSYNC	W	4
	R4	ช่วงเวลาแนวดิ่งทั้งหมด	W	7
การกำหนดช่วงเวลาในแนวตั้ง	R5	การปรับ VSYNC	W	5
	R6	จำนวนแถวต่อเฟรม	W	7
	R7	ตำแหน่ง VSYNC	W	7
	R8	โหมดการอินเทอร์เลซ	W	2
	R9	จำนวนสแกนต่อแถว	W	5

ตารางที่ 3.11 ความหมายของรีจิสเตอร์ต่าง ๆ (ต่อ)

	รีจิสเตอร์		เขียน (W) อ่าน (R)	จำนวนบิท
	หมายเลข	ชื่อ		
การกำหนดการทำงานพื้นฐาน	R10	ตำแหน่งขอบบนเคอร์เซอร์	W	7
	R11	ตำแหน่งขอบล่างเคอร์เซอร์	W	5
	R12	-	—	—
	R13	-	—	—
	R14	ตำแหน่งเคอร์เซอร์สูง	R/W	6
	R15	ตำแหน่งเคอร์เซอร์ต่ำ	R/W	8
	R16	ตำแหน่งปากกาแสงสูง	R	6
	R17	ตำแหน่งปากกาแสงต่ำ	R	8

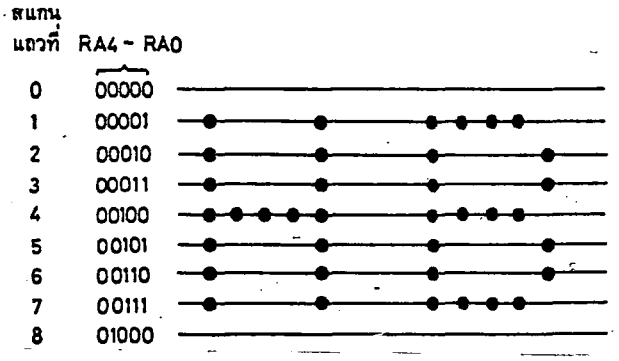
<u>รีจิสเตอร์ที่น่าสนใจมีดังนี้</u>

- R8 โหมดการอินเทอร์เลซ (Interlace) จากที่ทราบแล้ว จะได้ว่ามีการสแกน 3 แบบ คือ อินเทอร์เลซ นอนอินเทอร์เลซ และอินเทอร์เลซแบบวีดีโอ สามารถเลือกชนิดโหมดการอินเทอร์เลซในรีจิสเตอร์ R8 ได้จากบิท 0 และบิท 1 ดังตารางที่ 3.12

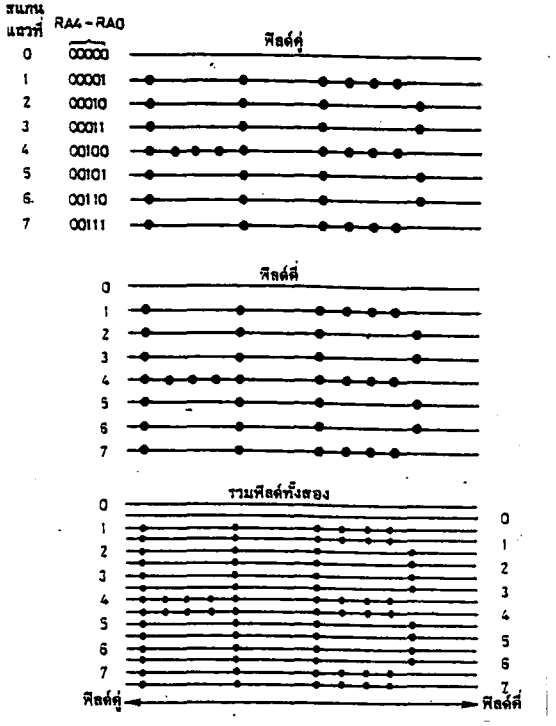
ตารางที่ 3.12 แสดงการเลือกชนิดโหมดการอินเทอร์เลซในรีจิสเตอร์ R8

บิท 1	บิท 0	โหมดการทำงาน
0	0	Noninterlace
1	0	Noninterlace
0	1	Interlace
1	1	Interlace And Video

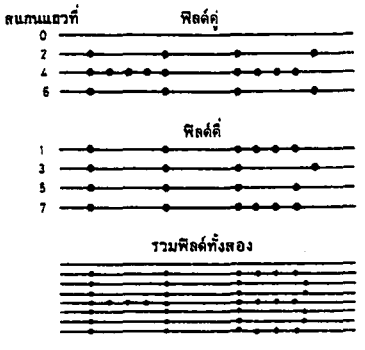
โดยมีการแสดงผลการอินเทอร์เลซในโหมดต่าง ๆ ดังรูปที่ 3.15



รูปที่ 3.15a แสดงผลในโหมดนอนอินเทอร์เลซ (Noninterlace Mode)

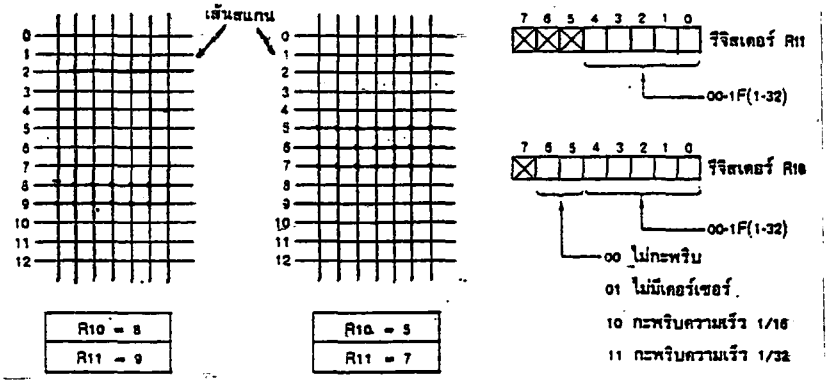


รูปที่ 3.15b แสดงผลในโหมดอินเทอร์เลซ (Interlace Mode)



รูปที่ 3.15c แสดงผลในโหมดอินเทอร์เลซวีดีโอ (Interlace Video Mode)

- R10 และ R11 เป็นจุดเริ่มต้นของเคอร์เซอร์และจุดจบของเคอร์เซอร์ ใช้กำหนดความสูงของเคอร์เซอร์ ได้โดยการกำหนดให้ 5 บิทแรกของ R11 เป็นขอบล่าง และ 5 บิทแรกของ R10 เป็นขอบบน บิทที่ 5 และ 6 ของ R10 จะกำหนดการกระพริบของเคอร์เซอร์ โดยแสดงดังรูปที่ 3.16



รูปที่ 3.16 แสดงการกำหนดขนาดของเคอร์เซอร์

การโปรแกรมค่าลงในรีจิสเตอร์ภายในของ 6845 CRTC จะต้องมีการถอดรหัสพอร์ท 2 พอร์ท พอร์ทแรก คือ พอร์ทรีจิสเตอร์แอดเดรส และพอร์ทรีจิสเตอร์ข้อมูล เมื่อต้องการกำหนดค่าลงในรีจิสเตอร์ตัวใด ก็กำหนดหมายเลขของรีจิสเตอร์นั้นลงในพอร์ทรีจิสเตอร์แอดเดรส และกำหนดค่าที่จะโปรแกรมลงในรีจิสเตอร์นั้นที่รีจิสเตอร์ข้อมูล

ตัวอย่าง ในโครงงานนี้ได้กำหนดพอร์ทรีจิสเตอร์แอดเดรสเป็น D0H และ พอร์ทรีจิสเตอร์ข้อมูลเป็น D1H การโปรแกรมรีจิสเตอร์ R8 ด้วยค่า 00H

```
LD    B,08D
OUT0  0D0H,B
LD    B,00H
OUT0  0D1H,B
```

การกำหนดค่าในรีจิสเตอร์นั้น ได้มาจากการคำนวณค่าคุณสมบัติพื้นฐานต่าง ๆ ดังนี้

1. ความถี่ HSYNC (B1)
2. ความถี่ VSYNC (B2)
3. เวลาการรีเทรซ (Retrace) ทางแนวราบน้อยสุด (B3)
4. เวลาการรีเทรซ (Retrace) ทางน้อยตั้งน้อยสุด (B4)
5. จำนวนตัวอักษรต่อบรรทัด (B5)
6. จำนวนแถวใน 1 หน้า (B6)
7. จำนวนจุดตามแนวราบของตัวอักษร 1 ตัว (B7)

8. จำนวนจุดตามแนวตั้งของตัวอักษร 1 ตัว (B8)

9. จำนวนจุดระหว่างตัวอักษร (B9)

10. จำนวนช่องว่างระหว่างบรรทัด (B10)

ค่าคุณสมบัติพื้นฐานเหล่านี้จะสามารถนำไปคำนวณหาค่าต่าง ๆ ได้ดังนี้

1. ค่าความถี่จุดโดยประมาณ (Dot Frequency) ; f'

$$f' = B5x(B7+B9)/[(1/B1)-B3)]$$

2. เวลาของตัวอักษร ; tc

$$tc = 1/[(R0+1)xB1]$$

3. ความถี่จุด (Dot Frequency) ;f

$$f = (B7+B9)/tc$$

4. เวลาการสแกน ; t_{s1}

$$t_{s1} = (R0+1)xtc$$

5. ผลรวมของเส้นสแกน ; n

$$n = 1/(B2xt_{s1})$$

6. ค่าจำนวนเต็ม และตัวเหลือ ; N , R

$$n/(B8+B10) = N + [R/(B8+B10)]$$

7. เวลาตามแนวขวาง ;tcr

$$tcr = (B8+B10)xtsl$$

8. เวลาย้อนกลับทางแนวราบ ; thr

$$thr = (R0+1-B5)x(B7+B9)/f$$

9. เวลาย้อนกลับทางแนวตั้ง ; tvr

$$tvr = (B1/B2)-(B6x(B8+B10)xtsl)$$

ค่าที่ได้มา สามารถนำไปหาค่าโปรแกรมในรีจิสเตอร์ได้ดังนี้

$$R0 = \{f'/[B1x(B7+B9)]\}-1$$

$$R1 = B5$$

$$R2 = R1 + (R3/2)$$

$$R3 = (R0-R1)/3$$

$$R4 = N-1$$

$$R5 = R$$

R6 = B6

R7 = (R4+1)-[(16-R5)/(B8+B10)] โดยที่ R7>=R6

R9 = (B8+B10)-1

ตัวอย่างการคำนวณในโครงงานนี้ เป็นดังนี้

B1 = 15625 Hz

B2 = 50 Hz

B3 = 10% ของ HSYNC = 6.4×10^{-6} sec

B4 = 3 % ของ VSYNC = 500×10^{-6} sec

B5 = 40

B6 = 25

B7 = 7

B8 = 10

B9 = 1

B10 = 1

นำค่าต่างที่กำหนดให้แทนในตัวแปรต่าง ๆ ได้ดังนี้

1. Dot frequency f' = B5x(B7+B9)/[(1/B1)-B3]

   = 40x(7+2)/[(1/15625)-6.4μs] = 6.25×10^{6} Hz

2. R0 = {f'/[B1x(B7+B9)]}-1 = {6.25×10^{6}/[15625x(7+2)]}-1 = 44.4-1 = 44D

3. R1 = B5 = 40D

4. R3 = (R0-R1)/3 = (44-40)/3 = 1.33 $\simeq$ 2

5. R2 = R1 + (R3)/2 = 40 + (2/2) = 41

6. tc = 1/[(R0+1)xB1] = 1/[(44+1)x15625] = 1.42×10^{-6} sec

7. Exact Dot Frequency f = (B7+B9)/tc = (7+2)/(1.42×10^{-6})

   = 6.34×10^{6} Hz

8. Scan Line Time t_{s1} = (R0+1)xtc = (44+1)x1.42×10^{-6}

   = 63.9×10^{-6} sec

9. Total Scan Line n = 1/($B2 \times t_{s1}$) = 1/($50 \times 63.9 \times 10^{-6}$)

   = 312.99 $\simeq$ 313 เส้น

10. $n/(B8+B10) = N+[R/(B8+B10)]$

$313/(10+1) = N+[R/(10+1)]$

จะได้ $N = 28$

$R = 5$

11. $R4 = N-1 = 28-1 = 27$

12. $R5 = R = 5$

13. R6 = B6 = 25 แถว

14. $R7 = (R4+1)-[(16-R5)/(B8+B10)]$ โดยที่ $R7 \geq R6$

$= (27+1)-[(16-5)/(10+1)] = 28-1 = 27$

ซึ่งพบว่า R7 มีค่ามากกว่า R6 แสดงว่าค่าใช้ได้

15. $R9 = (B8+B10)-1 = (10+1)-1 = 10$

16. $tcr = (B8+B10)xtsl = (10+1)x63.9x10^{-6} = 702.9x10^{-6}$ sec

17. $thr = (R0+1-B5)x(B7+B9)/f = (44+1-40)x(7+2)/6.34x10^{6}$

$= 7.098x10^{-6}$ sec

ซึ่งมีค่ามากกว่า B3 แสดงว่าค่าถูกต้อง

18. $tvr = [(B1/B2)-B6x(B8+B10)]xtsl = [(15625/50)-25x(10+1)]x63.9x10^{-6}$

$= 2396.25x10^{-6}$ sec

ซึ่งมีค่ามากกว่า B4 แสดงว่าค่าถูกต้อง

สามารถตรวจสอบความถูกต้องจาก การคำนวณค่า VSYNC

$VSYNC = 1/[(tcrx(R4+1))+(tslxR5)]$

$= 1/[(702.9x10^{-6}x28)+(63.9x10^{-6}x5)]$

$= 49.99 \simeq 50$ Hz

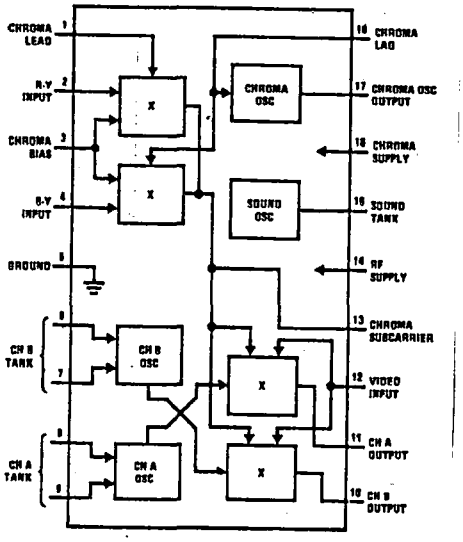
ซึ่งที่ได้ค่าที่ได้ตรงกับ B2 แสดงว่าค่าถูกต้อง

LM 1889 TV Modulator

ลักษณะทั่วไป

LM 1889 เป็นไอซีทีวีมอดดูเลเตอร์ (TV Modulator) ออกแบบมาเพื่อใช้ในภาคมอดดูเลเตอร์ของอุปกรณ์ต่าง ๆ เช่น เครื่องเล่นวีดีโอ คอมพิวเตอร์ ทีวีเกม เป็นต้น

ข้อดีของไอซีตัวนี้คือ มีฟังก์ชันในส่วนต่าง ๆ หลายฟังก์ชัน เช่น ภาคกำเนิดคลื่น Sub Carrier ของเสียง ภาคกำเนิดคลื่น Sub Carrier ของแสงและ RF Modulator ให้ความถี่เอาท์พุทไม่เกิน 100 MHz โดยมีการจัดขาสัญญาณ ดังรูปที่ 3.17



รูปที่ 3.17 แสดงแผนผังการจัดขาสัญญาณต่าง ๆ ของไอซี LM1889

ในการสร้างความถี่พาหะ ใช้วงจร LC หรือ ใช้คริสตอลเป็นตัวสร้างสัญญาณออสซิล-เลเตอร์เข้าที่ขา 6,7 สำหรับ Channel B และขา 8,9 ที่ Channel A มีขา 12 เป็นขาสัญญาณอินพุท มีขา 11 เป็นขาสัญญาณเอาท์พุทสำหรับ Channel A และมีขา 10 เป็นขาสัญญาณเอาท์พุทสำหรับ Channel B

บทที่ 4

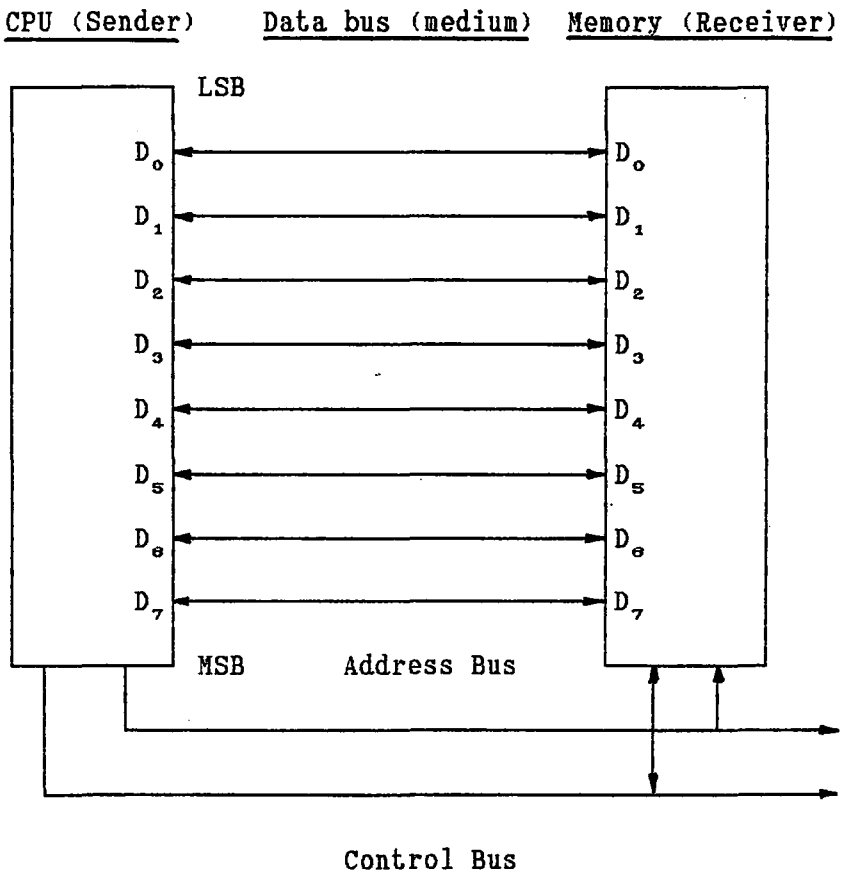
หลักเบื้องต้นของการสื่อสาร

การสื่อสารข้อมูล (Data Communication)

รูปแบบการสื่อสารข้อมูล

การสื่อสารข้อมูลแบ่งออกได้เป็น 2 แบบใหญ่ ๆ ด้วยกันคือ

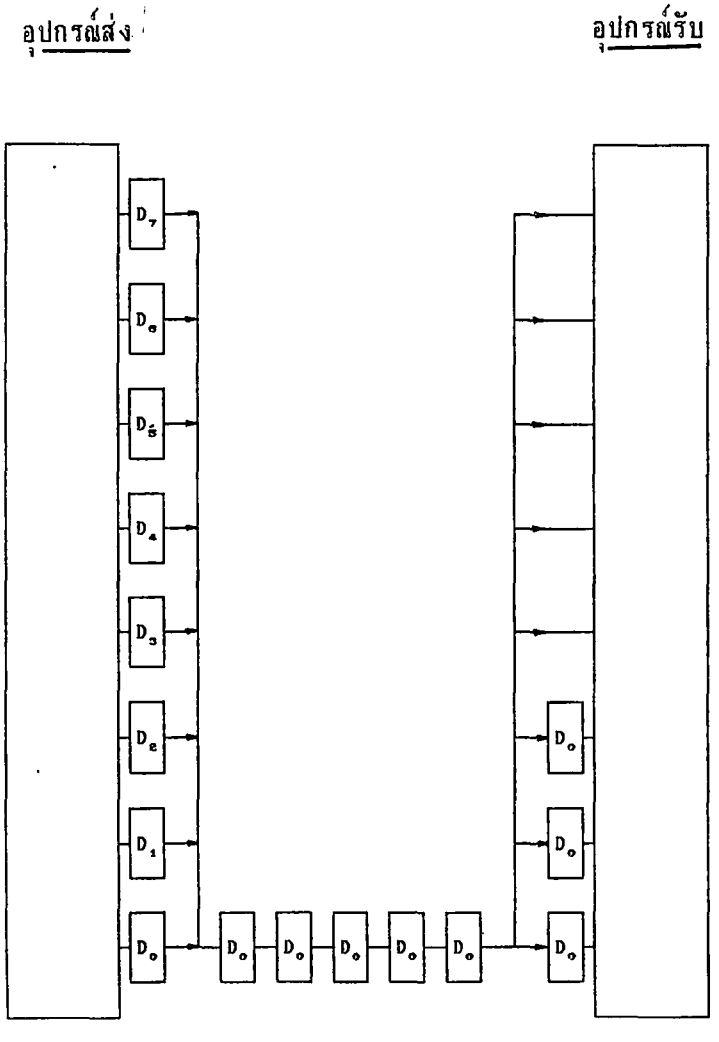
1. การสื่อสารแบบขนาน ข้อมูลทุกๆ บิทจะถูกส่งออกไปพร้อมกันในครั้งเดียว เช่น ถ้าข้อมูลที่ส่งเป็น 1010 ข้อมูลทั้งหมดนี้จะถูกส่งออกไปพร้อมกัน โดยผ่านสายส่งข้อมูล 4 เส้นโดยแต่ละบิทจะส่งในสายส่งคนละเส้น ในการสื่อสารหรือส่งผ่านข้อมูลแบบขนานนี้นิยมใช้กับการเชื่อมต่อที่เป็นอินพุทเอาท์พุท การต่อระหว่างระบบไมโครโปรเซสเซอร์และหน่วยความจำต่าง ๆ การต่อระหว่างอุปกรณ์ที่อยู่บนแผ่นวงจรพิมพ์เดียวกัน ซึ่งสามารถพิจารณาการสื่อสารข้อมูลแบบขนานได้ดังรูป 4.1



รูปที่ 4.1 แสดงการส่งข้อมูลแบบขนานระหว่าง CPU กับหน่วยความจำ

2. การสื่อสารข้อมูลแบบอนุกรม ข้อมูลแต่ละบิทจะถูกส่งเรียงกันออกไปเป็นลำดับต่อเนื่องกันไป โดยที่บิทที่มีนัยสำคัญน้อยสุด(Least Significant Bit)ถูกส่งออกไปก่อน แล้วตามด้วยบิทที่มีนัยสำคัญสูงขึ้นเรื่อย ๆ และบิทที่มีนัยสำคัญสูงสุด (Most Significant Bit)จะถูกส่งออกมาเป็นบิทสุดท้าย เช่น ถ้าข้อมูลที่ส่งคือ 1010 : เลข 0 ทางขวามือสุดจะถูกส่งออกมาก่อน เพราะเป็นบิทนัยสำคัญต่ำสุด แล้วตามด้วยบิทที่ 2 คือเลข 1 บิทที่ 3 คือเลข 0 และบิทสุดท้ายคือเลข 1 ตามลำดับโดยที่ข้อมูลแต่ละบิทจะส่งในสายส่งเส้นเดียวกัน การสื่อสารข้อมูลแบบอนุกรมนี้นิยมใช้กับการสื่อสารในระยะทางไกล ๆ เช่น การสื่อสารผ่านสายโทรศัพท์ เป็นต้น

เมื่อพิจารณาการสื่อสารข้อมูลแบบอนุกรมในรูปที่ 4.2 ข้อมูลจากจุดส่งจะถูกเปลี่ยนให้เป็นแบบอนุกรมเสียก่อน จึงค่อยมีการทยอยส่งออกไปครั้งละ 1 บิท ไปยังจุดรับ และที่จุดรับจะมีการเปลี่ยนข้อมูลแบบอนุกรมที่รับเข้ามาเป็นแบบขนานอีกครั้งหนึ่ง เพื่อนำข้อมูลเหล่านั้นไปใช้งานต่อไป



รูปที่ 4.2 แสดงการส่งข้อมูลแบบอนุกรมระหว่างตัวส่งและตัวรับ

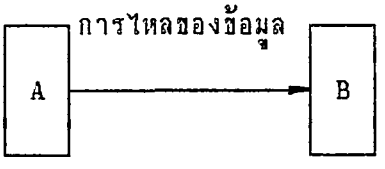
ตาราง 4.1 ข้อเปรียบเทียบระหว่างการส่งข้อมูลแบบขนานและแบบอนุกรม

พารามิเตอร์	การส่งข้อมูลแบบขนาน	การส่งข้อมูลแบบอนุกรม
1.ระยะทาง	ปกติจะน้อยกว่า 100 ฟุต	ส่งได้ตั้งแต่ระยะสั้น ๆ จนถึงระยะเป็นไมล์
2.ความเร็ว	อัตราความเร็วสูงมาก	อัตราความเร็วช้ากว่า โดยจะอยู่ในช่วง 0-2 ล้าน บิทต่อวินาที
3.ระดับของสัญญาณ	จะใช้ระดับสัญญาณ TTL คือสัญญาณลอจิก 1 แทนด้วยแรงดัน 5 V สัญญาณลอจิก 0 แทนด้วยแรงดัน 0 V	ใช้มาตรฐาน EIA-RS232C คือ มีระดับสัญญาณไฟขนาด $\pm$12 V แทนลอจิก 0 และลอจิก 1 หรืออาจใช้ระดับสัญญาณ TTL ก็ได้ แต่ใช้กันน้อยมาก
4.ความผิดพลาดของสัญญาณ	ถ้าส่งสัญญาณในระยะทางไกล ๆ ความผิดพลาดจะเกิดขึ้นได้ง่าย	ความผิดพลาดของสัญญาณจะมีน้อยลง
5.ค่าใช้จ่าย	ถ้าส่งในระยะทางไกล ๆ จะสิ้นเปลืองค่าใช้จ่ายมาก เพราะต้องใช้สายส่งสัญญาณหลายเส้น	สิ้นเปลืองน้อยกว่าหลายเท่า เพราะจำนวนสายส่งมีจำนวนน้อย

<u>รูปแบบของการติดต่อสื่อสารแบบอนุกรม</u>

การติดต่อแบบอนุกรมอาจจะแบ่งตามรูปลักษณะการส่งข้อมูลได้ 3 แบบ คือ

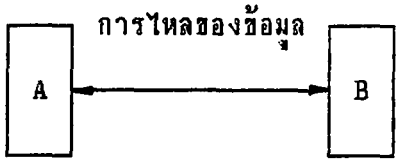
1.<u>การส่งข้อมูลแบบ Simplex</u> เป็นการส่งข้อมูลได้ทางเดียวเท่านั้น เช่น การส่งกระจายเสียงวิทยุหรือโทรทัศน์ โดยที่ตัวส่งจะทำหน้าที่ส่งข้อมูลได้อย่างเดียว และไม่สามารถรับข้อมูลได้ ส่วนตัวรับจะรับข้อมูลได้อย่างเดียว และไม่สามารถส่งข้อมูลได้ ดังแสดงในรูปที่ 4.3



รูปที่ 4.3 แสดงการส่งข้อมูลแบบ Simplex

โดยที่ A ทำหน้าที่เป็นตัวส่งข้อมูล ส่วน B ทำหน้าที่เป็นตัวรับข้อมูล

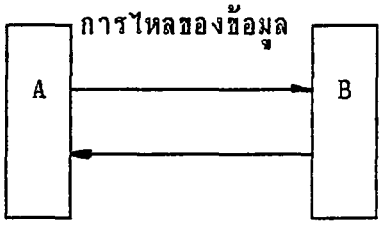
2.<u>การส่งข้อมูลแบบ Half Duplex</u> เป็นการส่งข้อมูลได้ทั้งสองทิศทาง แต่ต้องมีการผลัดกันรับและผลัดกันส่งข้อมูล จะส่งและรับพร้อมกันไม่ได้ เช่น ระบบ ATM ระบบวิทยุติดต่อ ซึ่งแสดงได้ในรูปที่ 4.4



รูปที่ 4.4 แสดงการส่งข้อมูลแบบ Half Duplex

โดยที่ A และ B ทำหน้าที่เป็นตัวส่งและตัวรับข้อมูล แต่ทำหน้าที่คนละเวลากัน คือถ้า A เป็นตัวส่ง B จะเป็นตัวรับ และในทางกลับกัน ถ้า A เป็นตัวรับ B จะเป็นตัวส่ง

3.<u>การส่งข้อมูลแบบ Full Duplex</u> เป็นการส่งข้อมูลได้ทั้งสองทิศทาง โดยสามารถส่งและรับข้อมูลได้ในเวลาเดียวกัน เช่น การสื่อสารทางโทรศัพท์ เป็นต้น ซึ่งสามารถแสดงได้ดังรูปที่ 4.5

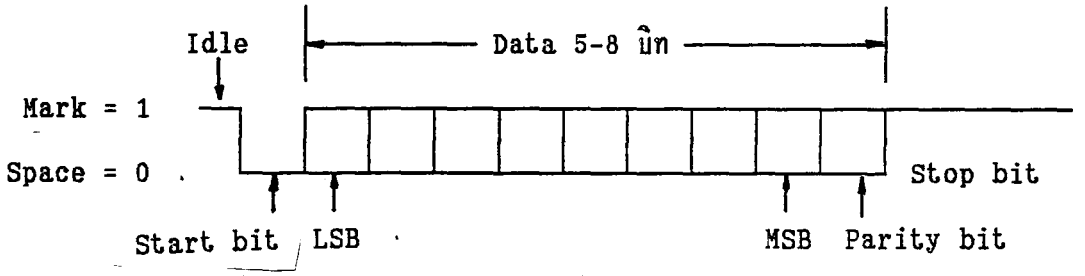


รูปที่ 4.5 แสดงการส่งข้อมูลแบบ Full Duplex

โดยที่ A และ B สามารถเป็นได้ทั้งตัวส่งและตัวรับในเวลาเดียวกัน

<u>การสื่อสารข้อมูลแบบอะซิงโครนัส</u> (Asynchronous)

การสื่อสารข้อมูลแบบนี้ คือ ระบบการรับส่งข้อมูลที่ระยะเวลาระหว่างข้อมูลแต่ละคำที่ถูกส่งออกไปมีค่าไม่แน่นอน เช่น การส่งข้อมูลจากคอมพิวเตอร์ A ไปยังคอมพิวเตอร์ B ในการป้อนข้อมูลจากคีย์บอร์ดนั้นผู้พิมพ์ข้อมูลไม่สามารถพิมพ์ด้วยอัตราเร็วที่คงที่ได้ จึงทำให้ระยะเวลาที่กดแต่ละครั้งห่างไม่เท่ากัน ซึ่งในการสื่อสารข้อมูลแบบอะซิงโครนัสนั้น ข้อมูลที่ส่งประกอบด้วยบิทเริ่มต้น (Start bit) ส่วนของข้อมูล (Data) และบิทสิ้นสุด (Stop bit) ดังแสดงได้ในรูปที่ 4.6



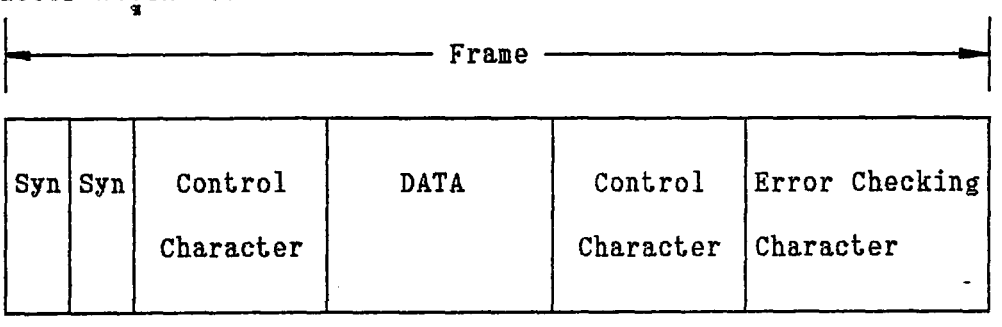
รูปที่ 4.6 แสดงรูปแบบของการสื่อสารแบบอะซิงโครนัส

ขณะที่สถานะของการส่งเป็นแบบว่างหรือไอเดิล (Idle) คือยังไม่มีสัญญาณส่งออกมา สายส่งจะมีสัญญาณและมีแรงดันตลอดเวลาเพื่อความแน่ใจว่าฝ่ายรับและฝ่ายส่งยังติดต่อกันอยู่ เมื่อมีการเริ่มส่ง ฝ่ายส่งจะส่งข้อมูลบอกจุดเริ่มต้น สัญญาณของอะซิงโครนัสจะเป็นลอจิก "0" หนึ่งช่วงสัญญาณนาฬิกา บิทนี้เรียกว่าบิทสตาร์ท (Start bit) ข้อมูล 1 ตัวอักษรที่ตามหลังบิทสตาร์ทจะมีขนาดตั้งแต่ 5 บิท จนถึง 8 บิท โดยบิทที่มีนัยสำคัญน้อยที่สุด (LSB) จะถูกส่งออกมาก่อนและไล่ไปจนถึงบิทที่มีนัยสำคัญมากที่สุด (MSB) หลังจากนั้นจะเป็นบิทพาริตี้ (Parity bit) ซึ่งอาจจะมีหรือไม่มีก็ได้ บิทนี้จะทำหน้าที่เป็นบิทตรวจสอบความถูกต้องของข้อมูลที่มีการรับส่ง ต่อจากนั้นจะเป็นบิทสิ้นสุด (Stop bit) ซึ่งเป็นลอจิก "1" โดยที่ความกว้างของบิทสิ้นสุดอาจเป็น 1,1.5 หรือ 2 พัลส์ของสัญญาณนาฬิกา ซึ่งแล้วแต่ผู้รับและผู้ส่งจะตกลงใช้กันเอง

การสื่อสารข้อมูลแบบซิงโครนัส (Synchronous)

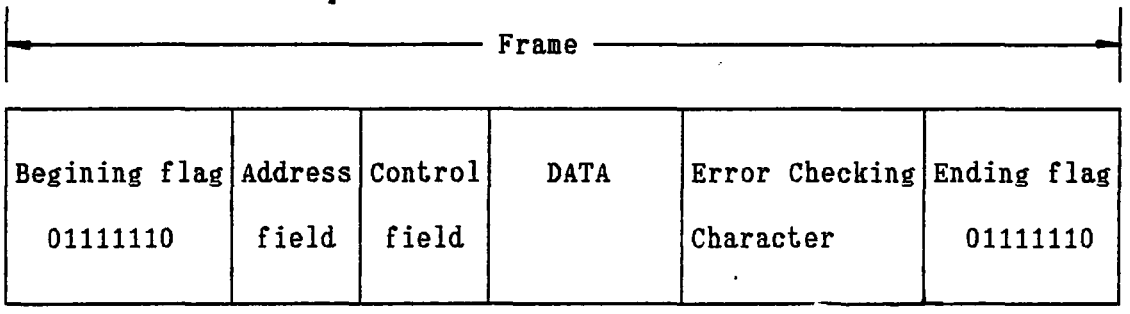
คือ ระบบการรับส่งข้อมูลที่ข้อมูลแต่ละคำถูกส่งออกไปตามเวลาที่แน่นอน โดยจะมีการส่งสัญญาณนาฬิกาไปพร้อม ๆ กับสัญญาณข้อมูล ข้อมูลที่ถูกส่งออกไปจะเป็นแบบต่อเนื่อง ไม่มีบิทเริ่มต้น บิทสิ้นสุด และบิทพาริตี้ โดยการส่งข้อมูลแบบซิงโครนัสนี้จะมี 2 รูปแบบที่แตกต่างกันดังนี้

1.Character Oriented Frame โดยจะเริ่มด้วยอักษรนำ คือ Syn Character ซึ่ง Syn Character จะเป็นรูปแบบของเลขฐานสอง "0" และ "1" สลับกัน หลังจากนั้นจะตามด้วย Control Information ,Data ,Control Character และ Error Checking Character ดังรูปที่ 4.7



รูปที่ 4.7 แสดงรูปแบบข้อมูลแบบ Character Oriented Frame

2.Bit Oriented Frame จะมีการใช้ Bit pattern เป็นตัวบอกจุดเริ่มต้นและจุดสิ้นสุดของแต่ละเฟรม ซึ่ง Bit pattern นี้จะมีขนาด 8 บิท และเรียกว่า แฟลก (Flag) โดยแฟลกที่ใช้มีรูปแบบเป็น 01111110 หลังจากส่ง begining flag ไปแล้ว จะตามด้วย Address field , control field , Data , Error Checking Character และ Ending flag ดังแสดงในรูปที่ 4.8



รูปที่ 4.8 แสดงรูปแบบข้อมูลแบบ Bit Oriented Frame

การตรวจจับข้อมูลที่ผิดพลาดในการสื่อสารแบบอะซิงโครนัส

ข้อผิดพลาดที่เกิดขึ้นเนื่องจากการสื่อสารมีหลายประการด้วยกัน ได้แก่

- Framing Error
- Parity Error
- Receiver Overrun Error

1. Framing Error

การตรวจจับ Framing Error เกิดขึ้นดังนี้ เมื่ออุปกรณ์ทางด้านตัวรับ ทำการรับสัญญาณของบิทสิ้นสุด (Stop bit) ซึ่งปกติจะมีลอจิก "1" แต่ปรากฏว่าสัญญาณที่รับได้เป็นสัญญาณลอจิก "0" ฝ่ายรับจะทราบทันทีว่ามีข้อผิดพลาดของข้อมูลเกิดขึ้น

2. Parity Error

ในการสื่อสารแบบอะซิงโครนัสนี้ อาจจะมีบิทพาริตี้ (Parity bit) เพิ่มเข้ามาก็ได้ โดยบิทนี้จะอยู่ตามหลังข้อมูลที่ส่งไป บิทพาริตี้จะทำหน้าที่ในการบอกให้ด้านรับข้อมูลทราบว่าข้อมูลที่ส่งมาแต่ละไบท์นั้น มีจำนวนบิทที่เป็น "1" อยู่เป็นจำนวนคี่หรือจำนวนคู่ ซึ่งบิทพาริตี้จะเป็น "1" หรือ "0" ขึ้นอยู่กับข้อมูลที่ส่งออกมาว่ามีจำนวนบิทที่เป็น "1"เป็นจำนวนคู่หรือคี่ และขึ้นกับอุปกรณ์รับส่งข้อมูลด้วยว่าถูกโปรแกรมให้รับส่งบิทพาริตี้ในลักษณะของพาริตี้คู่หรือพาริตี้คี่

ในกรณีอุปกรณ์รับส่งถูกออกแบบให้เป็นพาริตี้คู่ อุปกรณ์ส่งข้อมูลจะทำการส่งบิทพาริตี้เป็นลอจิก "1" ออกไปเมื่อจำนวนบิทที่เป็น 1 ของข้อมูลเป็นจำนวนคี่ และจะทำการส่งบิทพาริตี้เป็นลอจิก "0" เมื่อจำนวนบิทที่เป็น 1 ของข้อมูลเป็นจำนวนคู่ สำหรับกรณีอุปกรณ์รับส่งถูกออก

แบบให้เป็นพาริตี้คี่ อุปกรณ์ส่งข้อมูลจะทำการส่งบิทพาริตี้เป็นลอจิก "1" ออกไป เมื่อจำนวนบิทที่เป็น 1 ของข้อมูลเป็นจำนวนคู่ และจะส่งบิทพาริตี้เป็นลอจิก "0" ในกรณีที่บิทที่เป็น 1 ของข้อมูลเป็นจำนวนคี่

ตัวอย่าง ข้อมูลที่ส่งคือ 54H และ 55H

ข้อมูล 54H = 01010100B มีจำนวนบิทที่เป็น 1 เป็นจำนวนคี่

กรณีการส่งแบบพาริตี้คู่ อุปกรณ์ส่งจะส่งบิทพาริตี้เป็นลอจิก "1"

กรณีการส่งแบบพาริตี้คี่ อุปกรณ์ส่งจะส่งบิทพาริตี้เป็นลอจิก "0"

ข้อมูล 55H = 01010101B มีจำนวนบิทที่เป็น 1 เป็นจำนวนคู่

กรณีการส่งแบบพาริตี้คู่ อุปกรณ์ส่งจะส่งบิทพาริตี้เป็นลอจิก "0"

กรณีการส่งแบบพาริตี้คี่ อุปกรณ์ส่งจะส่งบิทพาริตี้เป็นลอจิก "1"

ในการรับส่งข้อมูลนั้น ฝ่ายส่งและฝ่ายรับ ต้องมีการโปรแกรมให้ตรวจสอบบิทพาริตี้ที่สอดคล้องกัน คือ พาริตี้คู่/พาริตี้คี่/ไม่มีพาริตี้ เมื่อฝ่ายส่งทำการส่งข้อมูล ฝ่ายรับก็จะรับข้อมูลและตรวจสอบบิทพาริตี้ว่าตรงกับฝ่ายส่งหรือไม่ โดยจะทำการเปรียบเทียบค่าของบิทพาริตี้ที่รับได้กับค่าบิทพาริตี้ของข้อมูลจริง ถ้าผลของการเปรียบเทียบไม่ตรงกัน ฝ่ายรับก็จะทราบได้ทันทีว่าข้อมูลมีความผิดพลาดของข้อมูลเกิดขึ้น

3. Receiver Overrun Error

ในการรับส่งข้อมูล อุปกรณ์ที่ทำหน้าที่รับส่งข้อมูลจะมีบัฟเฟอร์เก็บข้อมูลที่จะรับส่งอยู่ชุดหนึ่ง ในกรณีของอุปกรณ์ฝ่ายรับข้อมูลนั้น บัฟเฟอร์จะรับข้อมูลที่เป็นแบบขนานมาจากชิฟรีจิสเตอร์ ซึ่งในการควบคุมการทำงานนั้นจะต้องมาอ่านข้อมูลในบัฟเฟอร์เมื่อบัฟเฟอร์ที่รับข้อมูลเต็ม แต่ถ้าไม่มีการอ่านข้อมูลในบัฟเฟอร์เข้าไป เมื่อมีข้อมูลไบท์ต่อไปเข้ามาข้อมูลชุดใหม่ก็จะไปทับข้อมูลชุดเก่า ทำให้เกิด Overrun Error เกิดขึ้น

ในการใช้งานทั่วไปแล้ว เราจะต้องทำการแฮนด์เช็ค (Hand-shake) กันระหว่างด้านส่งและด้านรับ เพื่อป้องกันการรับส่งข้อมูลไม่ทันและการเกิด Overrun Error ขึ้นได้

ความเร็วในการสื่อสารข้อมูลแบบอนุกรม

1. Bit rate หมายถึง จำนวนของเลขฐานสองที่ส่งใน 1 วินาที เช่น Bit rate อาจจะเป็น 300,1200,2400,4800 และ 9600 bps

2. Baud rate หมายถึง การเปลี่ยนแปลงของสัญญาณใน 1 วินาที

พิจารณาความสัมพันธ์ระหว่าง Bit rate และ Baud rate

1. ถ้ามีการส่งสัญญาณ 1 สัญญาณ แทน 1 บิท จะทำให้ Bit rate เท่ากับ Baud rate

2. ถ้ามีการส่งสัญญาณ 1 สัญญาณแทน 2 บิท จะทำให้ Bit rate เป็น 2 เท่าของ Baud rate

จากความสัมพันธ์ดังกล่าว จะได้ว่า

Bit rate = (จำนวนบิทใน 1 ไบท์) x Baud rate

RS-232-C

ลักษณะทั่วไป

มาตรฐาน RS-232-C ถูกสร้างขึ้นเพื่อเป็นมาตรฐานแก่อุปกรณ์ต่าง ๆ ที่จะเชื่อมต่อกันผ่านทางสายโทรศัพท์ มาตรฐานนี้ถูกตั้งโดย EIA ย่อมาจาก Electronic Industries Association ในปี ค.ศ.1969 สำหรับมาตรฐาน RS-232-C นั้น ตัวอักษร RS แทนคำว่า "Recommened Standard" ตัวเลข 232 แทนหมายเลขของมาตรฐาน ส่วนอักษร C แสดงให้รู้ว่ามาตรฐานได้รับการแก้ไขปรับปรุง

มาตรฐาน RS-232-C กล่าวถึงการสื่อสารข้อมูลระหว่าง DTE (Data Terminal Equipment) และ DCE (Data Communication Equipment) โดยลักษณะของ DTE และ DCE เป็นดังนี้

Data Terminal Equipment (DTE)

1. เป็นอุปกรณ์ทางดิจิตอลที่ประกอบไปด้วยตัวส่งข้อมูล (data source) หรือตัวรับข้อมูล (data sink) หรือเป็นทั้งตัวส่งและตัวรับข้อมูลก็ได้

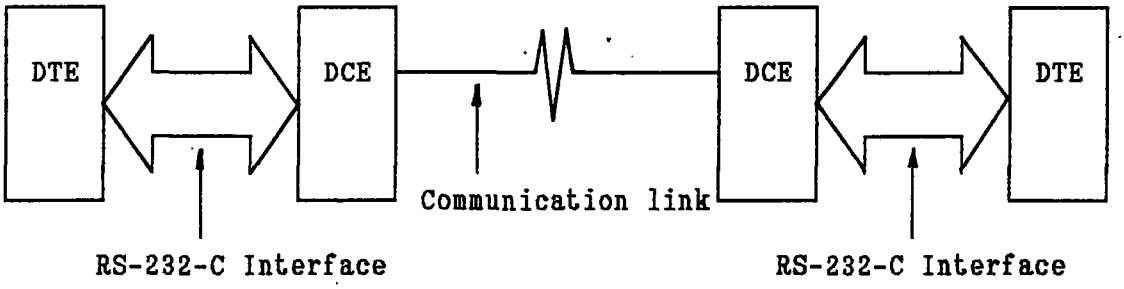
2. เป็นอุปกรณ์ที่ประกอบด้วยหน่วยฟังก์ชันต่อไปนี้ เช่น Control logic, Buffer Store และอุปกรณ์อินพุทหรือเอาท์พุทจำนวนหนึ่ง

ตัวอย่างของ DTE เช่น คอมพิวเตอร์, CRT display, printer เป็นต้น

Data Communication Equipment (DCE)

เป็นอุปกรณ์ที่มีฟังก์ชันการทำงานต่าง ๆ ที่ทำให้เกิดการเชื่อมต่อทางการสื่อสาร ซึ่งปกติมักใช้ร่วมกับระบบโทรศัพท์ ทำให้มีการย้ายข้อมูลจากจุดหนึ่งไปอีกจุดหนึ่ง โดยที่ DCE จะทำหน้าที่สร้างการเชื่อมต่อ ทำให้การเชื่อมต่อยังดำรงต่อไป และยุติการเชื่อมต่อ นอกจากนี้ยังใช้เปลี่ยนลักษณะสัญญาณและสร้างรหัสสัญญาณต่าง ๆ ที่จำเป็นต้องใช้ในการสื่อสารข้อมูลกับ DTE

ตัวอย่างของ DCE เช่น MODEM ซึ่งลักษณะของ DTE และ DCE ที่ใช้ในการสื่อสารข้อมูล แสดงดังรูป 4.9



รูปที่ 4.9 แสดงลักษณะของ DTE และ DCE ที่ใช้ในการสื่อสารข้อมูล

คุณสมบัติเชิงกลของการเชื่อมต่อ

รายละเอียดของขาต่าง ๆ ของคอนเน็คเตอร์ตามมาตรฐาน RS-232-C ได้แสดงไว้ในตารางที่ 4.2

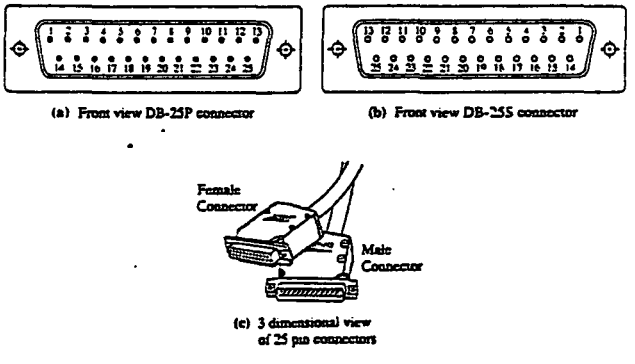
ตาราง 4.2 แสดงรายละเอียดของขาต่าง ๆ ของคอนเน็กเตอร์ตามมาตรฐานของ RS-232-C

ขา	วงจร	ความหมายของวงจร
1	AA	Protective Ground
2	BA	Transmitted Data
3	BB	Received Data
4	CA	Request to Send
5	CB	Clear to Send
6	CC	Data Set Ready
7	AB	Signal Ground
8	CF	Received Line Signal Detector
9/10	-	(Reserved for Data Set Testing)
11	-	Unassigned
12	SCF	Secondary Received Line Signal Detector
13	SCB	Secondary Clear to Send
14	SBA	Secondary Transmitted Data
15	DB	Transmit Signal Element Timing (DCE Source)

ตาราง 4.2 แสดงรายละเอียดของขาต่าง ๆ ของคอนเน็คเตอร์ตามมาตรฐานของ RS-232-C (ต่อ)

ขา	วงจร	ความหมายของวงจร
16	SBB	Secondary Received Data
17	DD	Receive Signal Element Timing
18	-	Unassigned
19	SCA	Secondary Request to Send
20	CD	Data Terminal Ready
21	CG	Signal Quality Detector
22	CE	Ring Indicator
23	CH/CI	Data Signal Rate Select (DTE/DCE Source)
24	DA	Transmit Signal Element Timing (DTE Source)
25	-	Unassigned

ในมาตรฐานนี้ไม่มีการกล่าวถึงปลั๊กตัวผู้ หรือปลั๊กตัวเมียของคอนเน็กเตอร์ ว่าต้องมีรูปร่างลักษณะอย่างไร ซึ่งในปัจจุบันมักนิยมใช้คอนเน็กเตอร์แบบ DB-25 ซึ่งบางทีเรียกว่าแบบ D-type 25 Pin Connector ในการเชื่อมต่อ โดยคอนเน็กเตอร์ตัวผู้จะเป็นชนิดปลั๊ก (Plug) คือ DB-25P ซึ่งจะใช้กับอุปกรณ์ DTE และคอนเน็กเตอร์ตัวเมียจะเป็นชนิดซอกเกต (Socket) คือ DB-25S ซึ่งจะใช้กับอุปกรณ์ DCE โดยลักษณะคอนเน็กเตอร์แสดงดังรูป 4.10



รูปที่ 4.10 แสดงลักษณะของคอนเน็กเตอร์ DB-25P และ DB-25S

ในมาตรฐาน RS-232-C นั้น ได้กำหนดความยาวของสายเคเบิลไว้ไม่เกิน 50 ฟุต ระหว่าง DTE และ DCE แต่ในการใช้งานจริงสายเคเบิลอาจยาวกว่า 50 ฟุตก็ได้ ถ้ารู้สภาพแวดล้อมของสายเคเบิล โดยอัตราการส่งข้อมูลอยู่ในช่วง 0 ถึง 20,000 บิทต่อวินาที (Bps)

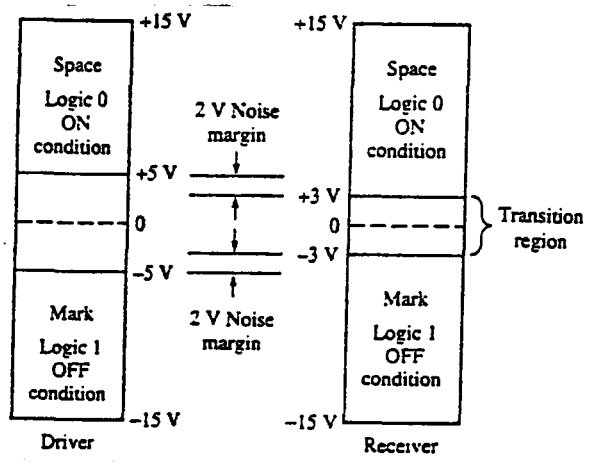
คุณสมบัติของสัญญาณไฟฟ้า

ในส่วนของมาตรฐานทางไฟฟ้า ความสัมพันธ์ระหว่างสถานะของสัญญาณต่าง ๆ กับแรงดัน แสดงดังนี้

ตาราง 4.3 แสดงความสัมพันธ์ระหว่างสถานะต่าง ๆ กับแรงดัน

	$-15V<V_1<-3V$	$3V<V_1<15V$
Logic State	1	0
Signal	mark	space
Control	off	on

โดยจากตารางพบว่า ในระบบ RS-232-C จะใช้ลอจิกลบ (Negative logic) แทนระดับแรงดันต่าง ๆ คือ ลอจิก "1" จะมีระดับแรงดันเป็นลบมากกว่าลอจิก "0" โดยแรงดันของระดับสัญญาณต่าง ๆ จะถูกวัดเทียบกับวงจร Signal ground และช่วงของระดับแรงดันระหว่าง -3 V ถึง +3 V เป็นช่วงของการเปลี่ยนแปลงลอจิก จึงไม่มีการระบุสถานะของสัญญาณในช่วงนี้ ในรูปที่ 4.11 แสดงช่วงของระดับแรงดันสำหรับตัวส่งและตัวรับ

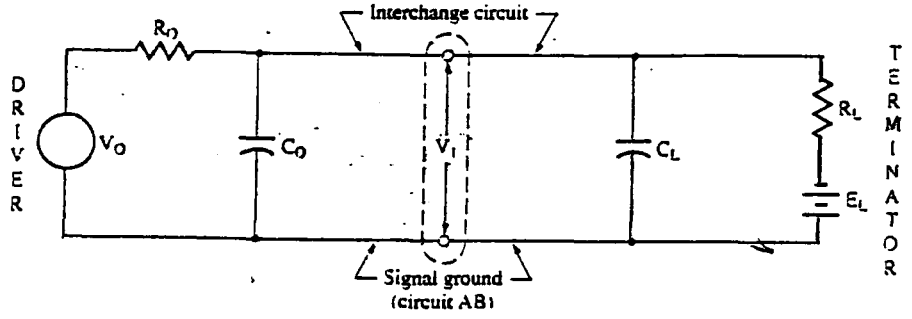


รูปที่ 4.11 แสดง RS-232-C Voltage Range

จากรูปพบว่า ตัวส่ง (Driver) เมื่อส่งสัญญาณลอจิก "0" ต้องจ่ายแรงดันระหว่าง +5 V ถึง +15 V ส่วนตัวรับ (Receiver) จะถือว่าแรงดันที่อยู่ภายในช่วง +3 V ถึง +15 V แทนลอจิก "0" ส่วนกรณีที่ตัวส่ง (Driver) ส่งสัญญาณลอจิก "1" นั้น ต้องจ่ายแรงดันระหว่าง -5 V ถึง -15 V ซึ่งตัวรับ (Receiver) จะถือว่าแรงดันที่อยู่ภายในช่วง -3 V ถึง -15 V แทนลอจิก 1 จากการเปรียบเทียบระดับสัญญาณของตัวส่งและตัวรับ พบว่า RS-232-C ยอมให้มีการดรอบ (drop) ของสัญญาณในช่วง 2 V เกิดขึ้นได้ ซึ่งก็คือค่า Noise Margin นั่นเอง

<u>วงจรสมมูล</u> (Equivalent Circuit)

จากรูปที่ 4.12 แสดงวงจรสมมูล (Equivalent Circuit) สำหรับการเชื่อมต่อ RS-232-C ซึ่งตัวส่งสัญญาณ (Drive) คือ DTE และตัวรับสัญญาณ (Terminator) คือ DCE



รูปที่ 4.12 แสดงวงจรการเชื่อมต่อ RS-232-C

จากรูปสามารถอธิบายได้ดังนี้

V_o เป็นแรงดันขณะเปิดวงจรของตัวส่ง (Driver) เทียบกับกราวด์ จะต้องไม่เกิน 25 โวลต์

R_o เป็นความต้านทานภายในของตัวส่ง (Driver) โดยขณะปิดวงจร กระแสที่ไหลผ่านตัวต้านทานต้องไม่เกิน 0.5 แอมป์

C_o เป็นค่าผลรวมความจุของตัวส่ง (Driver) ซึ่งเกิดจากผลรวมของค่าความจุของตัวส่ง และค่าความจุของสายส่งทางด้านตัวส่ง

C_L เป็นค่าผลรวมความจุของตัวรับ (Terminator) ซึ่งเกิดจากผลรวมของความจุของโหลดกับค่าความจุของสายส่งทางด้านโหลด โดย C_L จะต้องไม่เกิน 2,500 pF

R_L เป็นความต้านทานของตัวรับ (Terminator) ซึ่งเป็นโหลดในการเชื่อมต่อ ซึ่งมีค่าระหว่าง 3,000-7,000 โอห์ม โดยที่ค่า R_L ต้องไม่น้อยกว่า 3,000 โอห์ม เมื่อป้อนแรงดันไม่เกิน 25 โวลท์ และ R_L จะต้องไม่เกิน 7,000 โอห์ม เมื่อป้อนแรงดันจาก 3-25 โวลท์

E_L เป็นแรงดันขณะที่เปิดวงจรของโหลดหรือตัวรับ (Terminator) จะต้องมีค่าไม่เกิน 2 โวลท์

V_1 เป็นแรงดันที่จุดเชื่อมต่อ

ลักษณะการทำงานของวงจรต่าง ๆ

วงจรต่าง ๆ ตามตาราง 4.2 สามารถแยกออกได้เป็น 5 ประเภทคือ

1. กราวด์ หรือ Common Return (A)
2. วงจรข้อมูล (Data Circuit) (B)
3. วงจรควบคุม (Control Circuit) (C)
4. วงจรของสัญญาณฐานเวลา (Timing Circuit) (D)
5. วงจรของ Channel ที่สอง (Secondary Circuit) (S)

ทิศทางการทำงานของแต่ละวงจร แสดงดังตาราง 4.4

ตาราง 4.4 RS-232-C Interchange Circuit

Interchange circuit		Direction	
		DTE	DCE
AA	Protective ground (Does not exist in Version D)		
AB	Signal ground		
BA	Transmitted Data		
BB	Received data		
CA	RTS		
CB	CTS		
CC	DCE Ready (also known as Data Set Ready)		
CD	DTR		
CE	Ring-calling Indicator		
CF	Received Line Signal Detector (Data Carrier Detect)		
RL/CG	Remote Loop Back/Signal Quality Dectector		
CH	Data Signal Rate Selector (DTE source)		
CI	Data Signal Rate Selector (DCE source)		
DA	Transmitter Signal Element Timing (DTE source)		
DB	Transmitter Signal Element Timing (DCE source)		
DD	Receiver Signal Element Timing (DCE source)		
SBA	Secondary Transmitted Data		
SBB	Secondary Received Data		
SCA	Secondary RTS		
SCB	Secondary CTS		
SCF	Secondary Received Line Signal Detector		

การทำงานของแต่ละวงจร สรุปได้ดังนี้

กราวด์ (Ground)

วงจร AA : Protective Ground ขา 1

ในการใช้งานวงจรนี้อาจนำไปต่อกับตัวถังอุปกรณ์ เพื่อใช้เป็นสายดิน หรืออาจปล่อยลอยไว้โดยไม่มีการใช้งานก็ได้

วงจร AB : Signal Ground ขา 7

วงจรนี้นำไปใช้เป็นจุดอ้างอิงของสัญญาณที่ใช้ในวงจรต่าง ๆ จะต้องต่อไว้เสมอไม่ว่าจะมีการประยุกต์ใช้งานแบบใด

วงจรข้อมูล (Data Circuit)

วงจร BA : Transmitted Data (TxD) ขา 2

วงจรนี้ใช้ในการส่งผ่านข้อมูลจาก DTE ไปยัง DCE โดยในขณะที่ยังไม่มีการส่งข้อมูล DTE จะทำให้วงจร BA มีสถานะลอจิกเป็น 1 (Mark) ตลอดเวลา ข้อมูลจะถูกส่งไปยัง DCE ก็ต่อเมื่อวงจรควบคุมอยู่ในสถานะ ON ซึ่งวงจรควบคุมเหล่านี้ได้แก่ Request-to-send (RTS) Clear-to-send (CTS),Data Set Ready (DSR) และ Data Terminal Ready (DTR)

วงจร BB : Received Data (RxD) ขา 3

สัญญาณของวงจรนี้จะส่งออกมาจาก DTE ไปยัง DCE โดยมีการใช้งานดังนี้

สำหรับการส่งข้อมูลแบบซิมเพล็กซ์ (Simplex) และ ฟูลดูเพล็กซ์ (Full Duplex) เมื่อวงจรนี้อยู่ในสถานะ "ON" DCE จะอยู่ในโหมดการส่งข้อมูล (Transmit Mode) ซึ่ง DCE จะรับข้อมูลจาก DTE และส่งออกไปยังตัวกลางในการสื่อสารข้อมูล (Communication Link) เช่น DCE ที่เราใช้เป็นโมเด็ม โมเด็มจะส่งข้อมูลที่รับมาจาก DTE ไปยังเครือข่ายโทรศัพท์ แต่เมื่อวงจรนี้อยู่ในสถานะ "OFF" DCE จะไม่อยู่ในโหมดการส่งข้อมูล

สำหรับการส่งข้อมูลแบบฮาล์ฟดูเพล็กซ์ เมื่อวงจรอยู่ในสถานะ "ON" DCE จะอยู่ในโหมดการส่งข้อมูล แต่ถ้าวงจรนี้อยู่ในสถานะ "OFF" DCE จะอยู่ในโหมดการรับข้อมูล คือจะรับข้อมูลจากเครือข่ายสื่อสาร และส่งข้อมูลไปที่ DTE

วงจร CB : Clear to Send (CTS) ขา 5

สัญญาณของวงจรนี้จะส่งออกจาก DCE ไปยัง DTE เมื่อวงจรนี้อยู่ในสถานะ "ON" จะเป็นการบอกให้ DTE ทราบว่า ขณะนี้ DCE พร้อมที่จะรับข้อมูลที่จะส่งมาจาก DTE แล้ว เมื่อวงจรนี้อยู่ในสถานะ "OFF" จะเป็นการบอกให้ DTE ทราบว่า ขณะนี้ DCE ไม่พร้อมที่จะรับข้อมูล

สัญญาณ CTS จะอยู่ในสถานะ "ON" ก็ต่อเมื่อ สัญญาณ RTS และ DSR จะต้องอยู่ในสถานะ "ON" ทั้งคู่ โดยที่ CTS จะมีสถานะ "ON" หลังจาก RTS มีสถานะ "ON" ไปแล้วเป็นเวลา 8-50 มิลลิวินาที หรืออาจใช้เวลานานถึง 200 มิลลิวินาทีก็ได้

วงจร CC : Data Set Ready (DSR) ขา 6

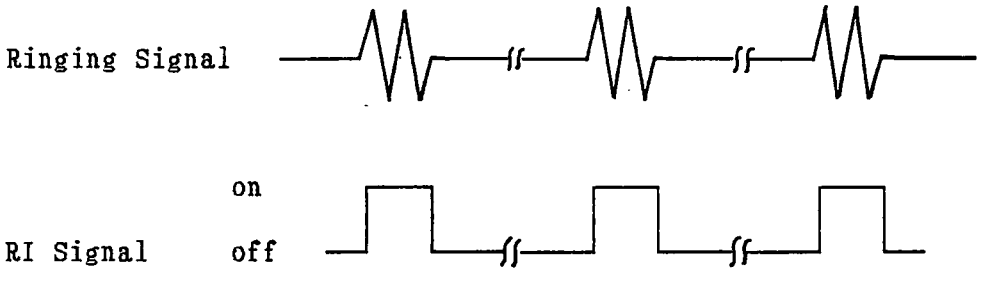
สัญญาณของวงจรนี้จะส่งออกมาจาก DCE ไปยัง DTE เมื่อวงจรนี้มีสถานะเป็น "ON" แสดงว่า DCE ได้ถูกต่อกับช่องทางสื่อสารแล้ว ในกรณีที่ DCE สามารถต่อโทรศัพท์ได้โดยอัตโนมัติ เมื่อวงจรนี้มีสถานะ "ON" ก็หมายถึง DCE ของเรา (Local) ได้ต่อโทรศัพท์เรียก DCE ที่เราต้องการติดต่อด้วย (Remote) และ DCE (Remote) ได้ตอบรับต่อการเรียกแล้ว ทำให้สามารถสื่อสารข้อมูลระหว่างกันได้

วงจร CD : Data Terminal Ready (DTR) ขา 20

สัญญาณของวงจรนี้จะส่งออกจาก DTE ไปยัง DCE โดยสัญญาณ DTR นี้จะต้องอยู่ในสถานะ "ON" ก่อนที่ DSR จะมีสถานะ "ON" เมื่อใดก็ตามที่ DCE ต่อกับช่องทางการสื่อสารแล้ว เมื่อ DTR เปลี่ยนสถานะจาก "ON" เป็น "OFF" DCE จะตัดการเชื่อมต่อกับ DCE (Remote) ทันที

วงจร CE : Ring Indicator (RI) ขา 22

สัญญาณของวงจรนี้จะส่งออกจาก DTE ไปยัง DCE โดยที่สัญญาณ DTR นี้ต้องมีสถานะเป็น "ON" แสดงว่า DCE กำลังได้รับสัญญาณเสียงกริ่ง (Ringing Signal) ที่มีเข้ามา เมื่อสัญญาณ OFF แสดงว่า DCE ไม่ได้รับสัญญาณเสียงกริ่ง เราใช้สัญญาณควบคุมนี้ในกรณีที่ใช้โมเด็มที่สามารถตอบรับต่อการเรียกได้โดยอัตโนมัติ (Auto Answer)

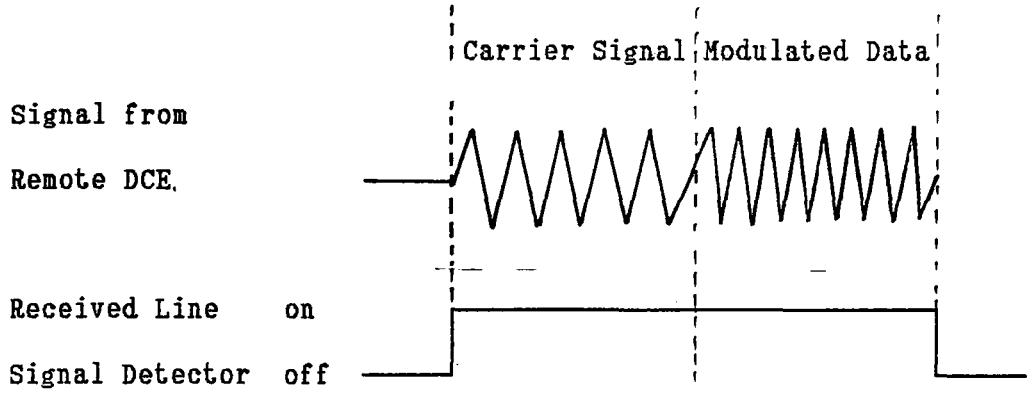


รูปที่ 4.13 แสดงผลของ RI ที่มีต่อ Ringing Signal

วงจร CF : Received Line Signal Detector

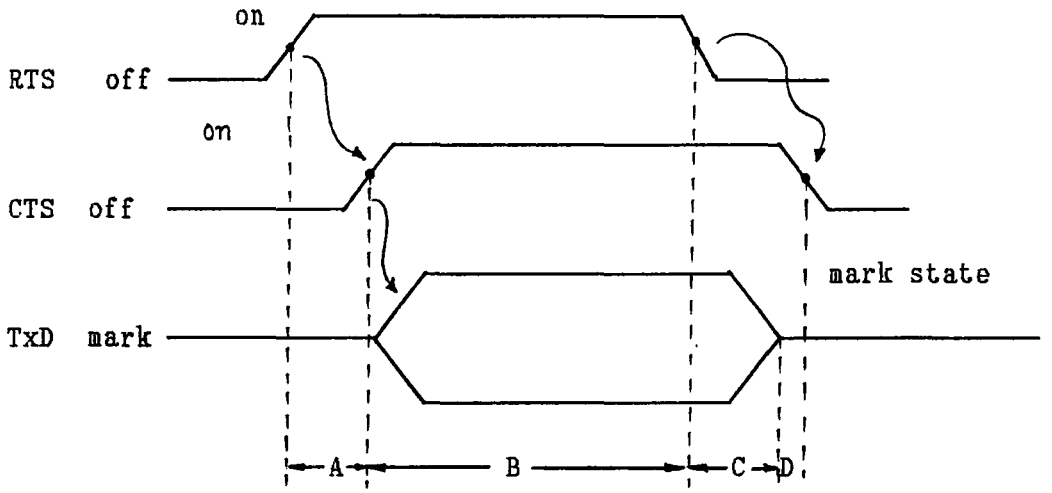
หรือ Data Carrier Detect (DCD) ขา 8

สัญญาณของวงจรนี้จะส่งออกจาก DCE ไปยัง DTE เมื่อ DCD มีสถานะ "ON" แสดงว่า DCE ได้รับสัญญาณ Carrier ที่ส่งมาจาก DCE อีกด้านหนึ่ง (Remote) ซึ่งจะนำไปใช้ในการดีมอดดูเลทได้แล้ว เมื่อ DCD มีสถานะ "OFF" แสดงว่า DCE ไม่สามารถรับสัญญาณที่จะมาดีมอดดูเลทได้ ในกรณีของฮาล์ฟดูเพล็กซ์ DCD จะมีสถานะ "OFF" เมื่อ RTS มีสถานะ "ON"



รูปที่ 4.14 แสดงสถานะการ "ON" และ "OFF" ของ DCD ในกรณีของฟูลดูเพล็กซ์ (Full Duplex)

การทำ Handshake ของสัญญาณ RTS และ CTS โดยคิดว่า DSR มีสถานะ "ON" อยู่ตลอดเวลา แสดงได้ดังรูปที่ 4.15



รูปที่ 4.15 แสดง Timing ของสัญญาณ RTS และ CTS

จากรูปที่ 4.15 ในช่วง A นี้ DTE จะทำให้ RTS มีสถานะเป็น "ON" เพื่อบอกให้ DCE ทราบว่า DTE ต้องการส่งข้อมูล ซึ่งเมื่อ DCE รับทราบ ก็จะจัดตั้งช่องทางการสื่อสาร และทำให้ CTS มีสถานะ "ON" ซึ่ง CTS จะมีสถานะ "ON" หลังจาก RTS มีสถานะ "ON" ไปแล้วเป็นเวลาประมาณ 8-50 มิลลิวินาที ซึ่งเป็นการแจ้งให้ DTE ทราบว่า DTE สามารถเริ่มต้นส่งข้อมูลได้แล้ว ในช่วง B ข้อมูลจะถูกส่งผ่านทางวงจร Transmitted Data (TxD) เมื่อข้อมูลถูกส่งออกไปหมดแล้ว DTE จะ OFF สัญญาณ RTS เพื่อบอกให้ DCE ทราบว่า DTE ไม่ต้องการที่จะส่งข้อมูลอีกต่อไป ในช่วง C เมื่อ DCE ส่งข้อมูลทั้งหมดออกไปยัง Communication Link วงจร TxD จะเข้าสู่สถานะ Mark อีกครั้ง และในช่วง D DCE จะ OFF สัญญาณ CTS เพื่อแจ้งให้ DTE ทราบว่า DCE พร้อมที่จะรับข้อมูลชุดใหม่เพื่อส่งออกไป เมื่อใดก็ตามที่ RTS เปลี่ยน

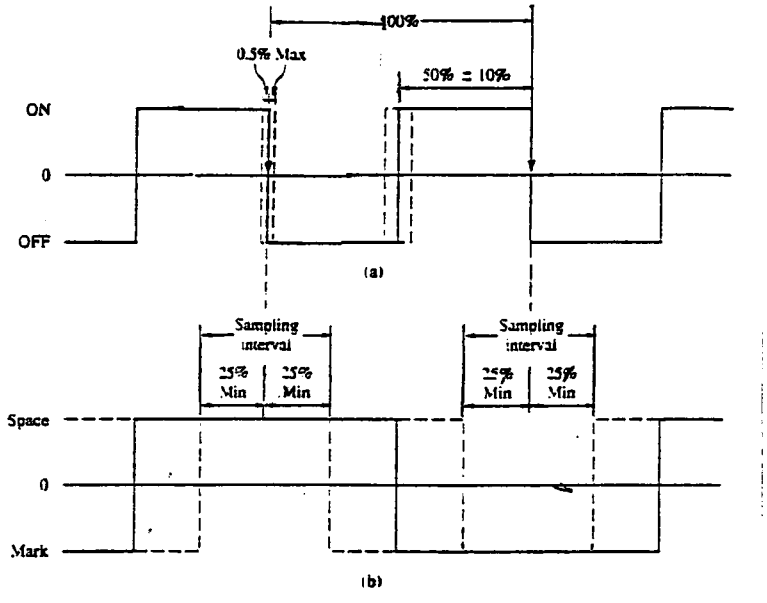
สถานะจาก "ON" เป็น "OFF" แล้ว RTS จะมีสถานะ "ON" อีกครั้ง เมื่อ DCE ทำให้ CTS เปลี่ยนสถานะจาก "ON" ไปเป็น "OFF" ซึ่งการทำเช่นนี้เป็นการป้องกันไม่ให้เกิด Overrun Error ขึ้นได้

วงจรของสัญญาณฐานเวลา (Timing Circuit)

เป็นวงจรที่ใช้ในการสื่อสารแบบซิงโครนัส ซึ่งมี 3 วงจร คือ

วงจร DA : Transmitting Signal Element Timing ขา 24

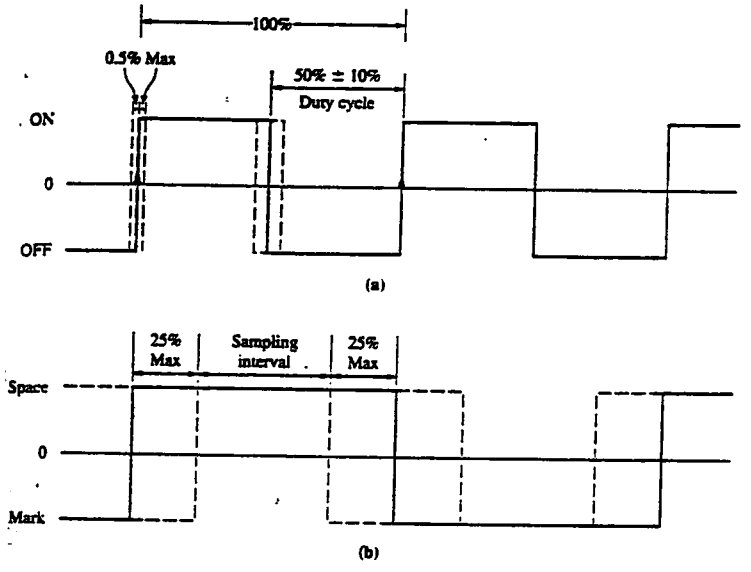
สัญญาณของวงจรนี้ส่งออกจาก DTE ไปยัง DCE เพื่อซิงโครนัสกับข้อมูลที่ส่งผ่านไปบนวงจร TxD สัญญาณนี้จะเปลี่ยนสถานะจาก "ON" ไปเป็น "OFF" ที่จุดกึ่งกลางของบิทของข้อมูลแต่ละข้อมูลที่ส่งออกไป ตัวอย่าง Timing แสดงดังรูปที่ 4.16



รูปที่ 4.16 แสดง Timing ของวงจร DA

วงจร DB : Transmitting Signal Element Timing ขา 15

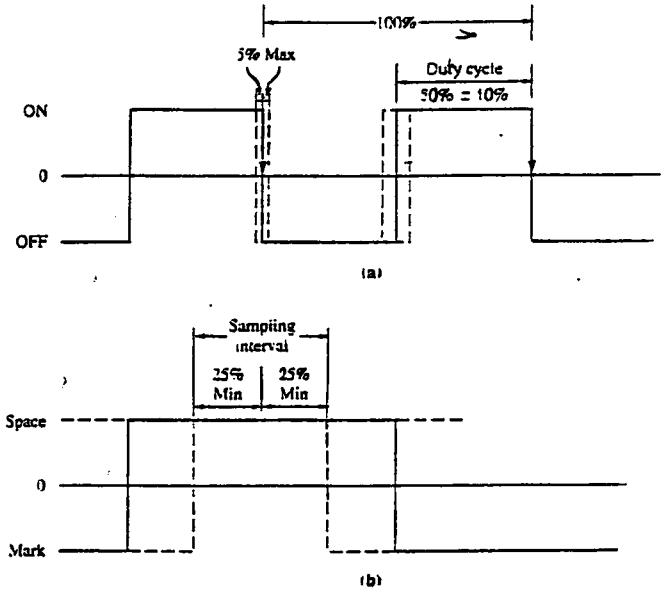
สัญญาณของวงจรส่งออกจาก DCE ไป DTE โดยที่ DTE จะส่งข้อมูลออกมาบนวงจร TxD เมื่อสัญญาณนี้เปลี่ยนสถานะจาก "OFF" ไปเป็น "ON" ดังรูปที่ 4.17



รูปที่ 4.17 Timing Circuit DB

วงจร DD : Receiver Signal Element Timing ขา 17

สัญญาณของวงจรนี้จะส่งออกจาก DCE ไปยัง DTE โดยสัญญาณนี้จะเปลี่ยนสถานะจาก "ON" ไปเป็น "OFF" และที่จุดกึ่งกลางบิทของข้อมูลที่รับเข้ามาทางวงจร RxD โดยสัญญาณของวงจรนี้ถูกใช้เมื่อ DCD มีสถานะเป็น "ON" ตลอดเวลาเท่านั้น ซึ่งการเปลี่ยนแปลงของสัญญาณนี้แสดงได้ดังรูปที่ 4.18

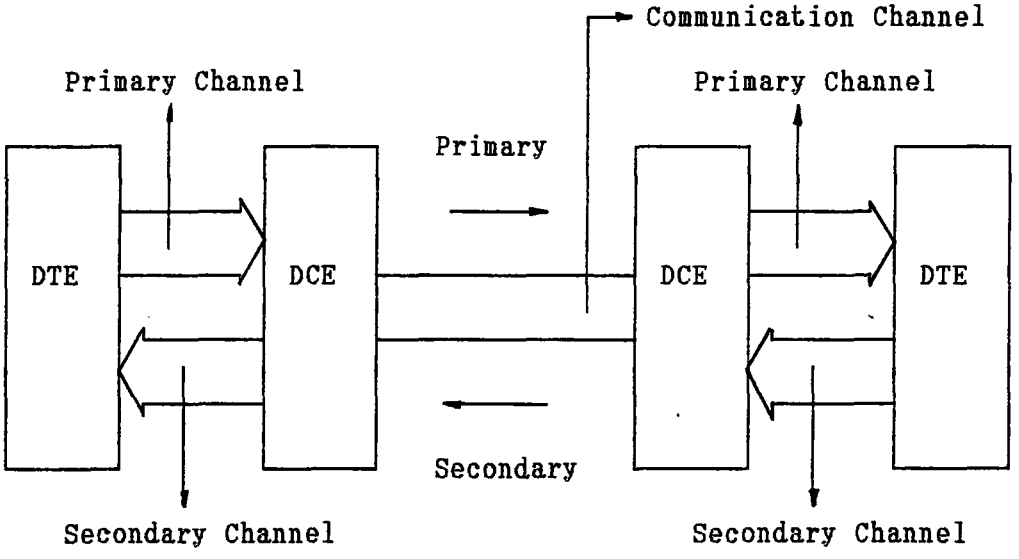


รูปที่ 4.18 Timing Circuit DD

วงจรของ Channel ที่สอง (Secondary Circuit)

ในวงจรต่าง ๆ ที่กล่าวมาก่อนหน้านี้ ใช้ในการสื่อสารของ Channel ที่หนึ่ง แต่ใน RS-232-C ยังมี Channel ที่สองที่ใช้ในการสื่อสารด้วย โดยอาจใช้เป็นซิมเพล็กซ์ (Simplex) ฮาล์ฟดูเพล็กซ์ (Half-duplex) หรือฟูลดูเพล็กซ์ (Full-duplex) ก็ได้ แต่อัตราการส่งข้อมูลของ Channel ที่สองจะน้อยกว่าอัตราการส่งข้อมูลของ Channel ที่หนึ่ง โดย Channel ที่สองแบ่งแยกได้เป็น 2 ประเภท คือ Backward Channel และ Auxiliary Channel

Backward Channel มีทิศทางการสื่อสารตรงข้ามกับ Channel ที่หนึ่ง (Primary Channel) ดังรูปที่ 4.19 โดย Backward Channel จะไม่ใช้ในการส่งข้อมูลแต่ใช้สำหรับการอินเทอร์รัพท์ โดยจะถูกส่งจากฝ่ายรับข้อมูลไปยังฝ่ายส่งข้อมูล เพื่อให้หยุดการส่งข้อมูล



รูปที่ 4.19 แสดง Secondary Channel โดยใช้เป็น Backward Channel

Auxiliary Channel มีทิศทางการสื่อสารไม่ขึ้นกับ Primary Channel คือมีทิศทางเดียวกัน หรือทิศทางตรงกันข้ามกับ Primary Channel ก็ได้

สำหรับวงจรของ Channel ที่สอง (Secondary Channel) มี 5 วงจรดังนี้

วงจร SBA : Secondary Transmitted Data ขา 14

วงจรนี้ใช้ในการส่งข้อมูลเหมือนกับวงจร BA แต่ใช้สำหรับ Channel ที่สอง ขณะที่ไม่มีการส่งผ่านข้อมูล วงจรนี้จะมีสถานะเป็น Mark ตลอดเวลา

วงจร SBB : Secondary Received Data ขา 16

วงจรนี้มีลักษณะคล้ายกับวงจร BB (Received Data) แต่ใช้กับ Channel ที่สอง

วงจร SCA : Secondary RTS ขา 19

วงจรนี้มีลักษณะคล้ายกับวงจร CA (RTS) แต่ใช้กับ Channel ที่สอง ซึ่งในกรณีที่มีการใช้งาน Channel ที่สองเป็น Backward Channel ถ้า RTS มีสถานะ "OFF" สามารถอีนาเบิลวงจร SCA ได้ เพื่อความแน่ใจว่า Channel ที่หนึ่งและ Channel ที่สอง จะไม่มีการส่งข้อมูลในทิศทางเดียวกัน

วงจร SCB : Secondary CTS ขา 13

วงจรนี้จะมีลักษณะคล้ายวงจร CB (CTS)

วงจร SCF : Secondary Received Line Signal Detector ขา 12

วงจรนี้จะแสดงถึงการรับสัญญาณพาหะ (Carrier) บน Channel ที่สอง ซึ่งจะมีลักษณะคล้ายกับวงจร CF (DCD)

<u>การส่งข้อมูลแบบฟูลดูเพล็กซ์</u> (Full-duplex)

ในการส่งข้อมูลแบบฟูลดูเพล็กซ์ (Full-duplex) ผ่านทางเครือข่ายโทรศัพท์ ต้องใช้สัญญาณต่าง ๆ ต่อไปนี้

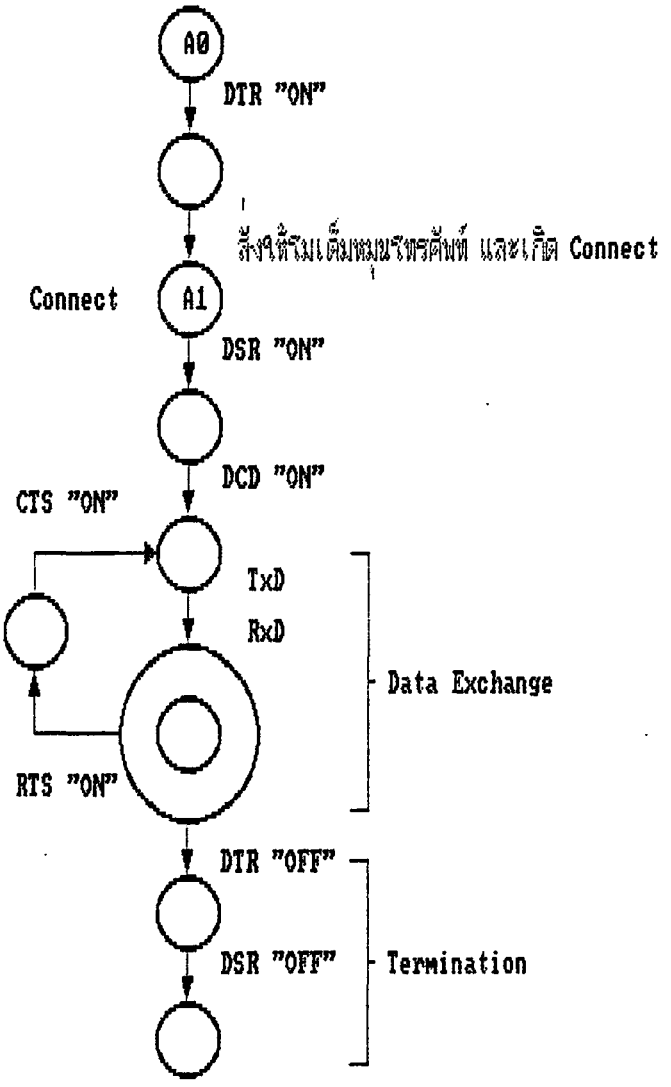
- Signal Ground
- Transmitted Data
- Received Data
- Request to Send
- Clear to Send
- Data Set Ready
- Data Terminal Ready
- Received Line Signal Detector
- Ring Indicator

โดยในการสื่อสารผ่านโทรศัพท์นี้ เราจะมีฝ่ายหนึ่งเป็นฝ่ายเรียก (Call Origination) ฝ่ายหนึ่งเป็นฝ่ายตอบรับต่อการเรียก (Answer) ซึ่งทั้งสองฝ่ายจะมีการเปลี่ยนแปลงของสัญญาณต่าง ๆ ดังนี้

<u>กรณีฝ่ายเรียก</u> (Call Origination)

เริ่มจากสภาวะ A_0 คือ สภาวะเริ่มต้น DTE จะทำให้ DTR อยู่ในสถานะ "ON" และมีการสั่งโมเด็มให้หมุนโทรศัพท์ เมื่อฝ่ายรับตอบรับต่อการเรียก ก็เข้าสู่สภาวะ A_1 คือสภาวะที่มีการติดต่อ หลังจากนั้น DCE จะทำให้ DSR มีสถานะ "ON" เมื่อ DSR และ DTR มีสถานะเป็น "ON" DCE จะส่งพาหะ (Carrier) ไปตามสายโทรศัพท์ ซึ่ง DCE ปลายทางก็จะทำการ

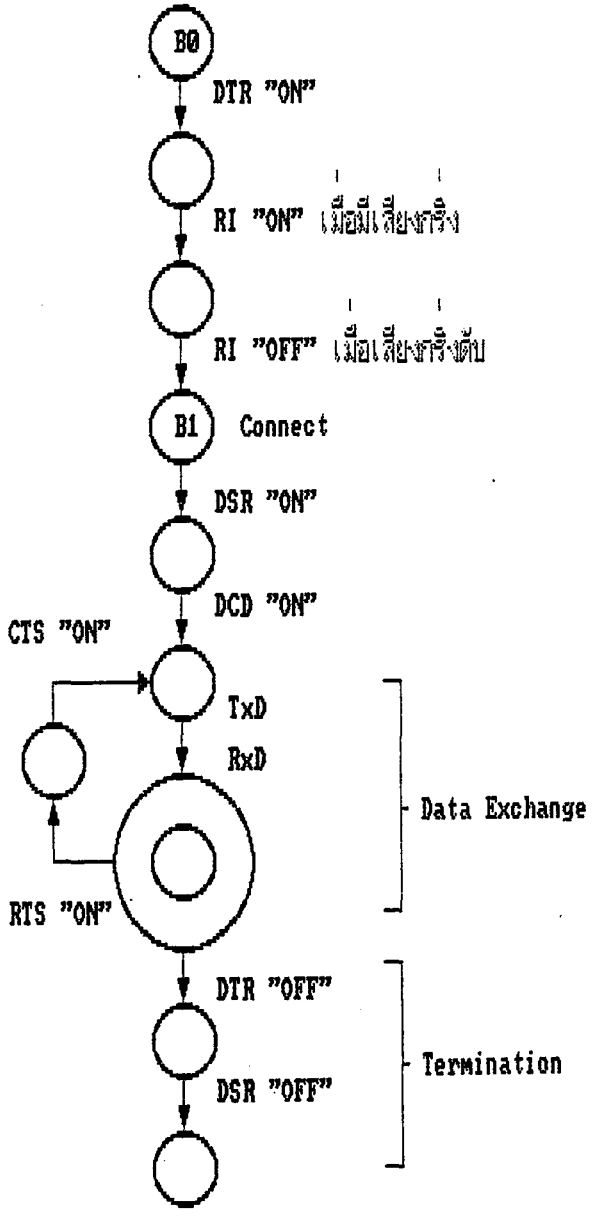
ส่งพาหะ (Carrier) กลับมาเช่นเดียวกัน เมื่อ DCE ของเราจับสัญญาณพาหะ (Carrier) ที่ส่งมาจากปลายทางได้ ก็จะทำให้สัญญาณ DCD มีสถานะ "ON" ซึ่งหลังจาก DCD มีสถานะ "ON" แล้วนี้ ก็จะเข้าสู่สภาวะที่สามารถแลกเปลี่ยนข้อมูลกันได้ (Data Exchange) ซึ่งเมื่อจะเริ่มส่งข้อมูล DTE จะทำให้สัญญาณ RTS มีสถานะ "ON" และ DCE ก็จะทำให้สัญญาณ CTS มีสถานะ "ON" เมื่อส่งข้อมูลหมดแล้ว DTE จะทำให้สัญญาณ RTS มีสถานะ "OFF" และ DCE ก็จะส่งข้อมูลที่เหลือไปให้หมด แล้วทำให้สัญญาณ CTS มีสถานะ "OFF" และเมื่อต้องการส่งข้อมูลชุดใหม่ก็จะเริ่มสภาวะการแลกเปลี่ยนข้อมูล (Data Exchange) อีกครั้งหนึ่ง จนกระทั่งเมื่อใดที่ DTE ทำให้สัญญาณ DTR มีสถานะ "OFF" DCE ก็จะทำให้สัญญาณ DSR มีสถานะ "OFF" ด้วย และเลิกการติดต่อในทันที ดังแผนผังในรูปที่ 4.20



รูปที่ 4.20 แสดงขั้นตอนการทำงานของฝ่ายเรียก

<u>กรณีฝ่ายรับตอบรับต่อการเรียก</u> (Answer)

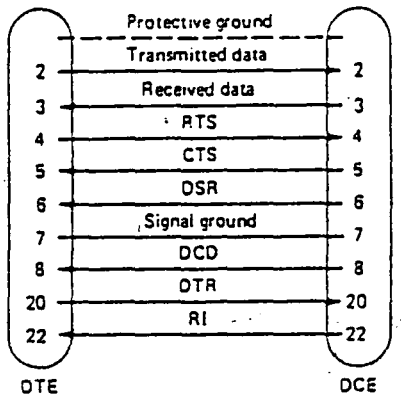
เริ่มจากสภาวะ B_0 คือ สภาวะเริ่มต้น DTE จะทำให้ DTR มีสถานะ "ON" เมื่อมีสัญญาณเสียงกริ่งโทรศัพท์เข้ามา DCE จะทำให้ RI มีสถานะ "ON" และ "OFF" ตามจังหวะการดังและดับของเสียงกริ่ง เมื่อ DCE ตอบรับต่อการเรียก ก็จะเข้าสู่สภาวะ B_1 คือ สภาวะที่มีการติดต่อ หลังจากนั้น DCE จะทำให้ DSR มีสถานะ "ON" เมื่อ DTR และ DSR มีสถานะเป็น "ON" DCE ก็จะส่งพาหะ (Carrier) ไปยังฝ่ายเรียก และรอรับพาหะ (Carrier) ที่ส่งมาจากฝ่ายเรียก เมื่อ DCE จับพาหะ (Carrier) ได้ก็จะทำให้สัญญาณ DCD มีสถานะ "ON" และเข้าสู่สภาวะ Data Exchange และ Termination เหมือนกับฝ่ายเรียก ซึ่งแสดงไดอะแกรมได้ดังรูปที่ 4.21



รูปที่ 4.21 แสดงขั้นตอนการทำงานของฝ่ายรับ

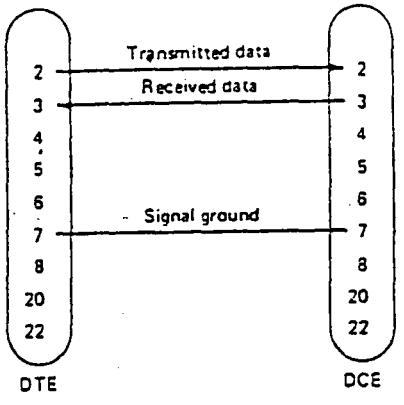
ลักษณะการต่อ RS-232-C

1. การต่อ RS-232-C แบบฟูลดูเพล็กซ์(Full-duplex) ซึ่งถูกต้องตามมาตรฐาน ซึ่งแสดงดังรูปที่ 4.22



รูป 4.22 แสดงการต่อแบบฟูลดูเพล็กซ์ตามมาตรฐานที่กำหนด

2. การต่อแบบ Three-wire Economic Model เป็นการต่อแบบฟูลดูเพล็กซ์ (Full-duplex) ที่ใช้วงจรน้อยที่สุดคือใช้เพียงวงจร Transmitted Data,Received Data และ Signal Ground ซึ่งมีการต่อแสดงได้ดังรูปที่ 4.23 แต่การต่อแบบนี้มีข้อผิดพลาดเกิดขึ้นคือ อุปกรณ์บางตัวในระบบไมโครคอมพิวเตอร์ต้องใช้ RTS และ CTS ด้วย ดังนั้นอุปกรณ์เหล่านี้จะส่งข้อมูลได้เมื่อได้รับสัญญาณ CTS ก่อน ตัวอย่างอุปกรณ์เช่น IC ประเภท USART



รูป 4.23 แสดงการต่อแบบ Three-wire Economic Model

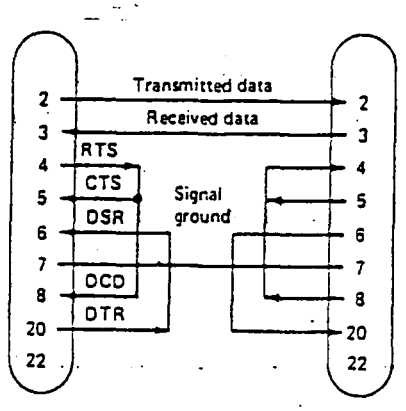
3. การต่อแบบ Three-wire with Luxury Loop-Back การต่อแบบนี้เป็นการแก้ปัญหาที่เกิดขึ้นจากการต่อตามข้อ 6.2 โดยมีการทำ Loop-Back ของสายสัญญาณดังนี้

Request to Send ต่อเข้ากับ Clear to Send

Request to Send ต่อเข้ากับ Received Line Signal Detector

Data Terminal Ready ต่อเข้ากับ Data Set Ready

ซึ่งแสดงการต่อได้ดังรูปที่ 4.24



รูป 4.24 แสดงการต่อแบบ Three-wire with Luxury Loop-Back

โมเด็ม

โมเด็ม (Modem) ใช้ในการเชื่อมโยงการส่งข้อมูลผ่านทางสายโทรศัพท์ ย่อมาจาก Modulate and Demodulate) คือ การผสมสัญญาณพาหะเข้าไปกับสัญญาณข้อมูล และการนำสัญญาณพาหะออกจากสัญญาณข้อมูล

จุดประสงค์ที่จำเป็นต้องมีการมอดดูเลทสัญญาณความถี่พาหะ เพื่อทำให้สามารถส่งไปได้ในระยะทางไกล โดยเกิดการผิดเพี้ยนของสัญญาณน้อยที่สุด

โมเด็ม เป็นอุปกรณ์ที่ทำหน้าที่ร่วมระหว่างการมอดดูเลท และดีมอดดูเลท แล้วทำการแปลงสัญญาณดิจิตอลที่ออกมาจากเครื่องคอมพิวเตอร์เพื่อส่งผ่านสายโทรศัพท์ ทางด้านรับก็จะแปลงสัญญานเปลี่ยนเป็นดิจิตอล เพื่อให้ส่งไปที่คอมพิวเตอร์โดยการสื่อสารผ่าน RS-232C

วิธีการนับความเร็วของโมเด็มแบ่งได้เป็น 2 แบบ

1. Baud Rate หมายถึง หน่วยของอัตราการส่งสัญญาณดิจิตอลที่แทนข่าวสารข้อมูลคิดต่อ 1 วินาที

2. **บิทต่อวินาที** (bit/sec) หมายถึง จำนวนของเลขฐานสองที่แทนข้อมูล ซึ่งถูกส่งออกไปใน 1 วินาที

เทคนิคการมอดดูเลท ที่ใช้กันในปัจจุบันมี 3 วิธี

1. Frequency Shift Keying (FSK)

เป็นโมเด็มความเร็วต่ำประมาณ 300-1200 บิทต่อวินาที ซึ่งแทนค่าสถานะทางดิจิตอลแทนด้วยความถี่ที่ต่างกันโดย

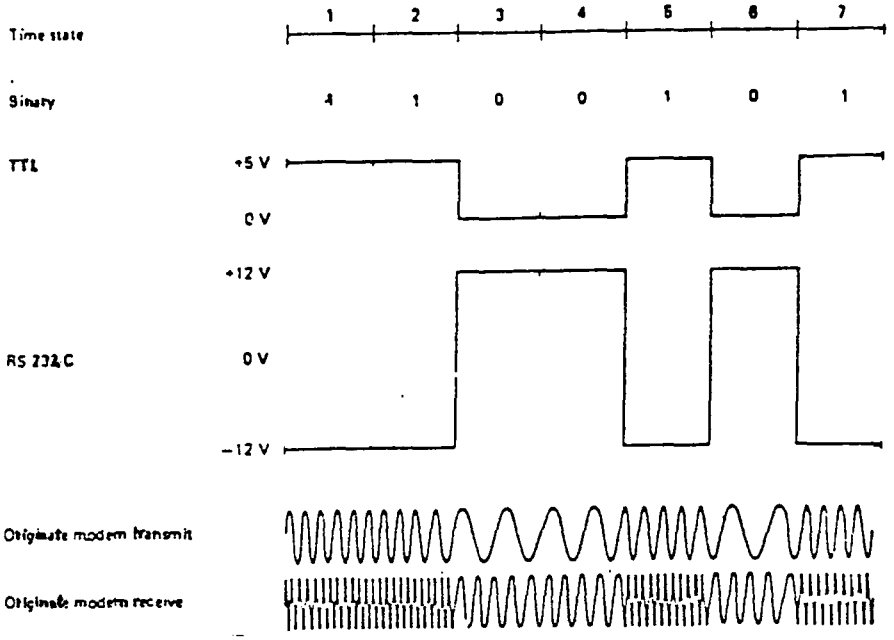
ทางด้านส่ง มีการแทนสถานะ "1" ด้วยความถี่ 1270 Hz

มีการแทนสถานะ "0" ด้วยความถี่ 1070 Hz

ทางด้านรับ มีการแทนสถานะ "1" ด้วยความถี่ 2225 Hz

มีการแทนสถานะ "0" ด้วยความถี่ 2025 Hz

สามารถแสดงได้ดังรูปที่ 4.25



รูปที่ 4.25 แสดงการมอดดูเลทสัญญาณแบบ FSK

2. Phase Shift Keying (PSK)

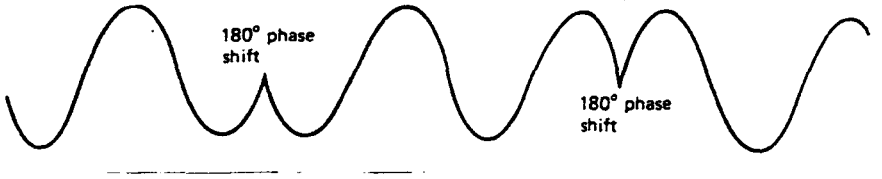
ใช้หลักการแทนข้อมูล "0" หรือ "1" ด้วยการเปลี่ยนแปลงมุมของช่วงสัญญาณ โดย

"0" แทนด้วยมุมของสัญญาณต่อเนื่อง

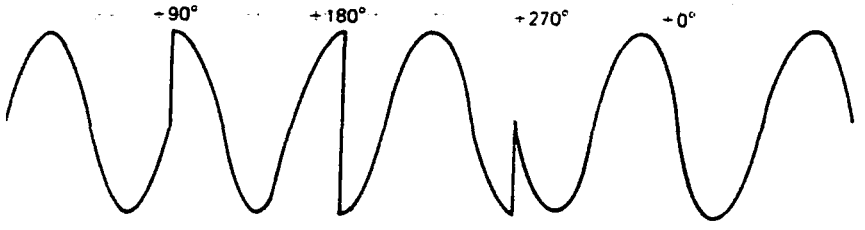
"1" แทนด้วยการเปลี่ยนแปลงมุมสัญญาณ 180 องศา

สามารถทำให้อัตราส่งสูงขึ้นโดยมีการเปลี่ยนมุมให้น้อยลง เช่น ให้ "0" แทนสัญญาณต่อเนื่อง "1" แทนสัญญาณเปลี่ยนแปลง 90 องศา และยังทำให้เร็วมากขึ้นอีกโดยการเปลี่ยนการทำให้มุมถี่ขึ้น เช่น 45 องศา,22.5 องศา โดยจะมีอัตราการส่ง 1200 บิทต่อวินาที ที่มุม

เปลี่ยน 180 องศา ,2400 บิทต่อวินาที ที่ 90 องศา ,4800 บิทต่อวินาทีที่มุมเปลี่ยน 45 องศา 9600 บิทต่อวินาที ที่มุมเปลี่ยน 22.5 องศา มีตัวอย่างแสดงดังรูป 4.26a และ 4.26b



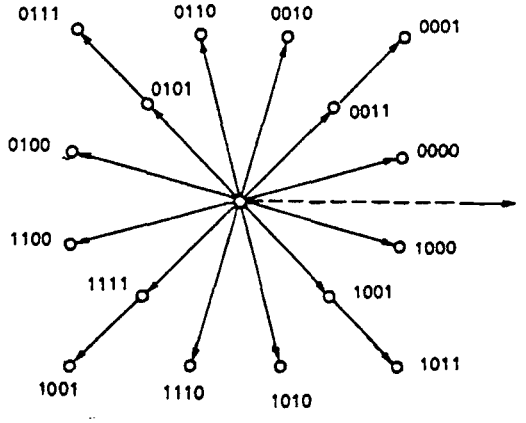
รูปที่ 4.26a แสดงการมอดดูเลทสัญญาณแบบ PSK เปลี่ยน 180 องศา



รูปที่ 4.26b แสดงการมอดดูเลทสัญญาณแบบ PSK เปลี่ยน 90 องศา

3. Phase Amplitude Modulation (PAM)

เป็นเทคนิคของ PSK กับการผสมสัญญาณแบบ AM (Amplitude Modulation) เข้าด้วยกัน โดยเมื่อมีการเปลี่ยนมุมไปแล้ว ยังมีแอมปลิจูดที่เพิ่มมากขึ้น เพื่อเพิ่มความแตกต่างมากขึ้น โดยจะมีอัตราการส่งประมาณ 9600 บิทต่อวินาที สามารถแสดงได้ดังรูปที่ 4.27 โดยแทนการเปลี่ยนแปลงเลขฐาน 2 จำนวน 4 บิท จาก 0000B - 1111B



รูปที่ 4.27 แสดงการมอดดูเลทสัญญาณแบบ PAM

Hayes Command Set

มาตรฐานคำสั่งของโมเด็ม

ในยุคแรกของการใช้งานโมเด็มต้องอาศัยคนเข้าช่วยเหลือเกือบทุกอย่าง เนื่องจากโมเด็มทำหน้าที่รับส่งสัญญาณ และเปลี่ยนแปลงสัญญาณที่ได้รับมาให้เครื่องคอมพิวเตอร์เท่านั้น การหมุนโทรศัพท์ และการติดต่อกับอีกด้านหนึ่งเพื่อส่งข้อมูลไปจนถึงการรับโทรศัพท์สำหรับโมเด็มด้านรับนั้น ต้องใช้คนทำในสิ่งเหล่านี้ทั้งหมด โมเด็มแบบนี้จึงใช้งานลำบาก เพราะผู้ใช้ต้องมีความเข้าใจระบบการทำงานของโมเด็มพอสมควร การปรับฟังก์ชันการทำงานต่าง ๆ ของโมเด็มมักจะใช้การผลักสวิทช์เล็ก ๆ (DIP Switch) บนตัวโมเด็ม ซึ่งในการปรับสวิทช์จะไม่สะดวกในการใช้งานแล้วยังจำยากด้วยว่าสวิทช์ตัวไหนใช้ทำอะไร และต้องผลักสวิทช์ไปทางไหน การสั่งงานของโมเด็มโดยใช้คำสั่งจึงเกิดขึ้น

ปกติโมเด็มจะติดต่อกับเครื่องคอมพิวเตอร์ เพื่อทำหน้าที่รับส่งข้อมูลอยู่แล้ว ดังนั้นในขณะที่โมเด็มยังไม่ได้ติดต่อกับปลายทางเพื่อส่งข้อมูล คอมพิวเตอร์ก็สามารถส่งคำสั่งต่าง ๆ ให้โมเด็มได้โดยไม่รบกวนการส่งข้อมูลแต่อย่างใด ในขั้นแรกเมื่อเปิดสวิทช์ให้โมเด็มทำงานสัญญาณที่โมเด็มได้รับจากคอมพิวเตอร์ จะถือว่าเป็นคำสั่งทั้งหมดจนกว่า จะติดต่อกับโมเด็มปลายทางได้ และเมื่อโมเด็มทำการส่งข้อมูลผ่านสายส่ง สัญญาณต่าง ๆ ที่คอมพิวเตอร์ส่งให้โมเด็มจะถือว่าเป็นข้อมูลทั้งหมด จนกระทั่งหยุดส่งข้อมูล โดยการเลิกการติดต่อกับปลายทาง หรือการวางสายโทรศัพท์นั่นเอง โมเด็มก็จะกลับมาอยู่ในช่วงรับคำสั่งจากคอมพิวเตอร์อีกครั้งหนึ่ง

บริษัท Hayes Microcomputer Products Inc. เป็นผู้คิดชุดคำสั่งชุดหนึ่งขึ้นมาเพื่อสั่งงานโมเด็มสำหรับคอมพิวเตอร์ส่วนบุคคล และได้รับความนิยมอย่างมากจนถือเป็นมาตรฐานอันหนึ่ง มาตรฐานคำสั่งนี้เรียกว่า Hayes Command Set เป็นคำสั่งที่ช่วยให้ผู้ใช้สามารถกำหนดการทำงานต่างๆ ของโมเด็มได้โดยใช้ซอฟท์แวร์สั่งจากคอมพิวเตอร์ไปยังโมเด็มโดยตรง ทำให้ไม่ต้องปรับสวิทช์เพื่อเลือกการทำงานแบบต่าง ๆ ของโมเด็มอีกต่อไป ซึ่งโมเด็มที่ใช้กับเครื่องคอมพิวเตอร์ส่วนบุคคลเกือบทั้งหมดจะรับคำสั่งตามมาตรฐานของ Hayes นี้ ซึ่งคำสั่งต่าง ๆ จะกล่าวถึงต่อไป

คำสั่งโมเด็มจะควบคุมการทำงานที่จำเป็นทั้งหมดของโมเด็ม เช่น การตอบรับสัญญาณโทรศัพท์ที่เรียกเข้ามา เลือกให้ทำงานในแบบ Echo on หรือ Echo off ต่อเข้าสายโทรศัพท์หรือวางสายโทรศัพท์รีเซ็ตโมเด็ม สั่งให้โมเด็มหมุนโทรศัพท์ตามเบอร์ที่กำหนด ปรับพารามิเตอร์ต่าง ๆ ของโมเด็ม ฯลฯ ซึ่งถ้าหากไม่ใช้คำสั่งโมเด็มแล้ว ผู้ใช้จะต้องกำหนดตัวแปรเหล่านั้นด้วยวิธีการผลักสวิทช์บนโมเด็มตามที่กล่าวข้างต้น จะเห็นได้ว่าการใช้คำสั่งมีความสะดวกและง่ายต่อการใช้งานมาก ข้อดีอันหนึ่งของการใช้คอมพิวเตอร์ส่งคำสั่งให้โมเด็มก็คือ ซอฟท์แวร์ติดต่อ

สื่อสาร สามารถปรับตัวแปรต่าง ๆ ของโมเด็มให้เป็นไปอย่างที่ต้องการ โดยที่ผู้ใช้ไม่ต้องรู้รายละเอียดใด ๆ เลย โปรแกรมจะจัดการให้เสร็จ และติดต่อส่งข้อมูลได้ทันที ซึ่งโปรแกรมคนละโปรแกรมอาจใช้งานโมเด็มไม่เหมือนกัน แต่ละโปรแกรมก็จะปรับโมเด็มให้ทำงานต่างกันได้โดยไม่ต้องแก้ไขส่วนที่เป็นฮาร์ดแวร์ของโมเด็มเลย การใช้งานโมเด็มจึงมีความคล่องตัวมากกว่าการใช้สวิทช์เลือกแบบเก่า ซึ่งถ้ามีการเปลี่ยนแปลงอะไรก็ต้องปรับสวิทช์กันทีหนึ่งทุกครั้งไป และอาจเกิดความผิดพลาดได้ง่ายกว่าการใช้คำสั่งงานโมเด็ม

ภายในตัวโมเด็มจะมีหน่วยความจำพิเศษ สำหรับเก็บตัวแปรในการทำงานแทนที่สวิทช์แบบเก่า หน่วยความจำนี้จะยังคงเก็บค่าต่าง ๆ เอาไว้ได้ แม้ว่าจะปิดโมเด็ม หรือดึงปลั๊กโมเด็มออกก็ตาม โมเด็มที่ใช้คำสั่งของ Hayes เรียกหน่วยความจำส่วนนี้ว่า S-Register เอาไว้ใช้เก็บพารามิเตอร์ในการทำงานของโมเด็ม เช่น จำนวนครั้งที่จะตอบรับสัญญาณเรียกเข้า หรือช่วงเวลาสำหรับรอสัญญาณก่อนหมุนโทรศัพท์ ฯลฯ ดังรูปที่ 4.28

Smartmodem S-Register Set

Register	Description	Units	Range	Default	
S0	Select ring to answer on	Rings	0-255	0	
S1	Ring count (incremented with each ring)	Rings	0-255	0	
S2	Escape-sequence character		ASCII	0-127	43(+)
S3	Carriage-return character	ASCII	0-127	13	
S4	Line-feed character	ASCII	0-127	10	
S5	Backspace character	ASCII	0-32,127	8	
S6	Wait-time before blind *dialing*	Seconds	2-255	2	
S7	Wait-time for *carrier*/dial tone	Seconds	1-255	30	
S8	Length of pause for comma in number	Seconds	0-255	2	
S9	Carrier-detect response time	.1 second	1-255	6	
S10	Delay between carrier loss/hangup	.1 second	1-255	7/14	
S11	Duration/spacing of DTMF tones	.001 second	50-255	70/195	
S12	Escape-sequence guard time	.02 second	20-255	50	
S16	Test mode				
S18	Select test timer	Seconds	0-255	0	
S25	Data terminal ready change detect time	.01 second	0-255	5	
S26	Select RTS to CTS delay	.01 second	0-255	1	

รูปที่ 4.28 ตารางแสดง S-Register ของ Hayes

การใช้งานของ S-Register

Reading or Changing a Register

ในการอ่านค่าปัจจุบันของรีจิสเตอร์ จะใช้คำสั่ง Sr? และการเปลี่ยนค่าในรีจิสเตอร์ใช้คำสั่ง Sr=n ค่าของ r แสดงหมายเลขของรีจิสเตอร์ ส่วนค่า n แสดงค่าใหม่ที่จะเซ็ตให้กับรีจิสเตอร์ r

ตัวอย่าง ATS77 อ่านค่าในรีจิสเตอร์ S7

ATS7=150 เซ็ตค่าในรีจิสเตอร์ S7 เป็น 150

S0 - Auto-Answer Ring Number

รีจิสเตอร์นี้เป็นตัวกำหนดจำนานของการส่งสัญญาณกริ่ง ก่อนที่โมเด็มจะตอบรับต่อการเรียก ซึ่งค่าใน S0 อยู่ในช่วง 0-255 มีค่าปกติเป็น 0 กล่าวคือ โมเด็มไม่ตอบรับแบบอัตโนมัติ ถ้าค่าใน S0 ถูกเซ็ตเป็น 2 โมเด็มจะตอบรับต่อการเรียกหลังจากเสียงกริ่งครั้งที่ 2

S1 - Ring Count

รีจิสเตอร์นี้จะเพิ่มค่าทีละ 1 ก็ต่อเมื่อโมเด็มมีการตรวจจับสัญญาณกริ่งบนสายโทรศัพท์ได้ในแต่ละครั้ง และรีจิสเตอร์นี้จะถูกเคลียร์เป็น 0 หลังจากที่เสียงกริ่งครั้งสุดท้ายหยุดลงเป็นเวลา 8 วินาที

รีจิสเตอร์ S1 จะถูกอีนาเบิลเมื่อ S0 ถูกเซ็ตเป็นค่าที่ไม่เท่ากับศูนย์ คือมีการตอบรับแบบอัตโนมัติ

S2 - Escape Character

รีจิสเตอร์นี้ใช้เก็บรหัสแอสกี (ASCII) ที่เป็นเลขฐานสิบของ Escape Character ซึ่งค่าปกติเป็น 43 คือรหัส ASCII ของ "+"

S3 - Carriage Return Character

รีจิสเตอร์นี้ใช้เก็บรหัสแอสกี (ASCII) ที่เป็นเลขฐานสิบของ Caraige Return Character ซึ่งอักษร (Character) ดังกล่าวจะอยู่ท้ายของชุดคำสั่งและรหัสการตอบรับ

S4 - Line Feed Character

รีจิสเตอร์นี้ใช้ในการเก็บรหัสแอสกี (ASCII) ที่เป็นเลขฐานสิบของ Line Feed Character ซึ่งอักษรตัวนี้จะใช้ต่อกับอักษร Carriage Return หลังจากที่มีรหัสคำสั่งตอบรับส่งกลับมา

S5 - Backspace Character

รีจิสเตอร์นี้ใช้ในการเก็บรหัสแอสกี (ASCII) ที่เป็นเลขฐานสิบของ Backspace Character ซึ่งอักษรตัวนี้ใช้เป็นคีย์ Backspace สำหรับการเขียนและเป็นอักษรที่ Echo เพื่อย้ายเคอร์เซอร์บนจอกลับถอยหลัง เมื่ออักษรตัวนี้ถูกกด ก็จะลบอักษรตัวก่อนหน้านี้ที่อยู่ในแถวของคำสั่ง อักษร Backspace นี้อยู่ในช่วง 0-32 และ 127 เท่านั้น ซึ่งค่าปกติคือ 08 (ASCII)

S6 - Wait Time Before Dialing

รีจิสเตอร์นี้เป็นตัวกำหนดเวลาที่โมเด็มใช้ในการรอคอยก่อนที่จะเริ่มการหมุนโทรศัพท์ ซึ่งเวลาล่าช้านี้เป็นเวลาที่ให้ศูนย์ข้อมูลตรวจสอบเงื่อนไขต่าง ๆ ของการติดต่อ ถ้าเวลาที่กำหนด

ใน S6 มีค่าน้อยมาก ๆ โมเด็มจะเริ่มต้นหมุนเร็วมากเช่นกัน ทำให้การติดต่อไม่ได้ผล ซึ่งเวลาที่ล่าช้าต่ำสุดก่อนการหมุนโทรศัพท์ของโมเด็มคือ 2 วินาที แม้ว่า S6 จะถูกเซ็ตด้วยค่าที่น้อยกว่า 2 วินาทีก็ตาม ค่าของ S6 นี้อยู่ในช่วง 2-255 โดยมีค่าปกติเป็น 2 (วินาที)

S7 - Wait for Carrier

รีจิสเตอร์นี้เป็นตัวกำหนดเวลาที่โมเด็มใช้รอคอยสัญญาณพาหะ หลังจากการหมุนหรือการตอบรับต่อการเรียก ถ้ามีการตรวจพบสัญญาณพาหะได้ในช่วงเวลามีกำหนด ก็จะมีการติดต่อกันได้ ถ้าตรวจไม่พบในช่วงเวลาดังกล่าว แสดงว่าการติดต่อไม่ได้ผล และมีรหัสตอบรับกลับมาว่า NO CARRIER ซึ่งค่าใน S8 อยู่ในช่วง 2-255 โดยมีค่าปกติเป็น 45 (วินาที)

S8 - Pause Time for a Comma (,) Dial Modifier

รีจิสเตอร์นี้เป็นตัวกำหนดเวลาที่โมเด็มหยุดเมื่อมีการใช้คำสั่งคอมมา มีค่าอยู่ในช่วง 0-255 ซึ่งค่าปกติเป็น 2 (วินาที)

S9 - Carrier Detect Time

รีจิสเตอร์นี้เป็นตัวกำหนดเวลาที่มีสัญญาณพาหะคงอยู่อย่างต่อเนื่อง เพื่อให้โมเด็มรับรู้ว่าขณะนี้มีการติดต่อแล้ว ถ้าเวลาใน S9 มีค่ามาก จะทำให้โมเด็มมีเวลามากพอในการแยกความแตกต่างระหว่างสัญญาณพาหะออกจากสัญญาณรบกวนได้ โดยมีค่าอยู่ในช่วง 1-255 และค่าปกติเป็น 6 (0.1 วินาที)

S10 - Carrier Lost Time

รีจิสเตอร์นี้เป็นตัวกำหนดเวลาที่สัญญาณพาหะสูญหายไป เพื่อให้โมเด็มรับรู้ว่าในขณะนี้ไม่มีการติดต่อแล้ว S10 มีค่าอยู่ในช่วง 1-255 และค่าปกติเป็น 14 (0.1 วินาที)

S11 - Touch-Tone Dialing Speed

รีจิสเตอร์นี้เป็นตัวกำหนดช่วงเวลาของสัญญาณโทน และช่วงการเว้นว่าง เมื่อมีการหมุนแบบโทน ซึ่งค่าปกติของอัตราการหมุนประมาณ 5.26 ตัวเลขต่อวินาที S11 มีค่าอยู่ในช่วง 70-255 และค่าปกติเป็น 95 (0.01 วินาที)

S12 - Escape Guard Time

รีจิสเตอร์นี้เป็นตัวกำหนด Escape Guard Time ของการไม่มีสัญญาณที่ต่อเนื่องก่อนและหลัง Escape Character เพื่อทำลำดับการ Escape ให้สมบูรณ์

ซอฟท์แวร์ที่ใช้สื่อสารสามารถเปลี่ยนค่าตัวแปรเหล่านี้ชั่วคราว หรือเปลี่ยนค่าถาวรไปเลยก็ได้ โดยใช้คำสั่งเก็บค่าตัวแปรเอาไว้ หน่วยความจำพิเศษนี้ บางชนิดใช้แบทเตอรีเล็ก ๆ คอยจ่ายไฟให้เวลาที่เราปิดโมเด็ม เพื่อป้องกันค่าต่าง ๆ หายไปจากหน่วยความจำ ดังนั้น เมื่อใช้โมเด็มไปนาน ๆ แบทเตอรีดังกล่าวจะหมดลง เราจำเป็นต้องเปลี่ยนอันใหม่ให้ ไม่เช่นนั้น

การทำงานของโมเด็มอาจผิดพลาดได้ เนื่องจากค่าของตัวแปรในหน่วยความจำหายไป โมเด็มบางชนิดเก็บค่าตัวแปรในหน่วยความจำแบบที่ไม่ต้องใช้แบทเตอรีจ่ายไฟสำรองให้โมเด็ม ทำให้ไม่ต้องเปลี่ยนแบทเตอรีภายในให้มัน และโมเด็มบางแบบก็ไม่ยอมให้มีการเปลี่ยนค่าตัวแปรเหล่านี้อย่างถาวรโดยเก็บค่าต่าง ๆ เอาไว้ใน ROM (Read Only Memory) การเปลี่ยนค่าตัวแปรจะเป็นไปชั่วคราวเท่านั้น เมื่อเปิดปิดโมเด็มใหม่ค่าต่าง ๆ จะกลับเหมือนเดิมตามที่ผู้ผลิตกำหนดเอาไว้ใน ROM ของโมเด็มนั่นเอง

มาตรฐานคำสั่งของ Hayes

Hayes Command เป็นคำสั่งที่ใช้สั่งงานโมเด็ม มีอีกชื่อหนึ่งเรียกว่า AT Command เพราะคำสั่งทุกคำสั่งขึ้นต้นด้วยตัวอักษร AT เสมอ เมื่อจบคำสั่งให้ปิดท้ายด้วย CR (Carriage Return) หรือกดปุ่ม Enter โมเด็มจะรับคำสั่งนั้นไปทำงานทันที และตอบรับคำว่า OK กลับมา คำสั่งเรียงตามอักษร A ถึง Z มีดังนี้คือ

AT - (Attention Code) เป็นคำสั่งที่ใช้เคลียร์บัฟเฟอร์คำสั่ง และบอกโมเด็มให้ทราบถึงความเร็วในการส่งของคอมพิวเตอร์ของผู้ใช้ รูปแบบข้อมูล และพาริตี้ โดยคำสั่งใด ๆ ยกเว้น A/ ต้องตามหลังคำสั่ง AT และจบด้วยอักษร CR (Carriage Return)

D - (Dial in Originate Mode) คำสั่งนี้ใช้ส่งให้โมเด็มทำการหมุนโทรศัพท์อัตโนมัติ แทนการที่ใช้คนหมุนเอง เรียกว่า Auto Dialing คำสั่งนี้มักตามด้วยชนิดของโทรศัพท์ที่ต่ออยู่ด้วย เช่น ATDT จะเป็นคำสั่งให้โมเด็มหมุนโทรศัพท์แบบกดปุ่ม และ ATDP จะเป็นคำสั่งสำหรับหมุนโทรศัพท์แบบมือหมุน ตัวอย่างการใช้งาน เช่น ATDT 2345678 ตามด้วย Return หรือ Enter เป็นตัวจบคำสั่ง

E - Command Characters Echo คำสั่งนี้ใช้สั่งให้โมเด็ม Echo ข้อความกลับไปให้เครื่องคอมพิวเตอร์โดยใช้คำสั่ง ATE0 สำหรับไม่ให้โมเด็ม Echo และ ATE1 เพื่อให้โมเด็ม Echo ข้อความกลับไปให้เครื่องคอมพิวเตอร์ ในขณะที่โมเด็มรับคำสั่งอยู่ใน Command Mode เมื่อเราใช้โมเด็มรับส่งข้อมูลแบบฟูลดูเพล็กซ์ (Full Duplex) จะต้องเลือกใช้ ATE1 เพื่อให้ข้อความที่เราพิมพ์ส่งไปยังโมเด็ม Echo กลับมาให้เราเห็นบนจอภาพ และถ้าเป็นฮาล์ฟดูเพล็กซ์ (Half Duplex) ก็เลือกใช้ ATE0 เพราะเครื่องคอมพิวเตอร์จะพิมพ์ข้อความนั้นบนจอภาพอยู่แล้ว คำสั่ง ATE0 และ ATE1 นี้มีผลเฉพาะก่อนการเชื่อมต่อกับปลายทางเท่านั้น เมื่อเราติดต่อกับปลายทางได้แล้ว โมเด็มจะทำการรับส่งข้อมูลผ่านสายโทรศัพท์โดยไม่มีการ Echo ข้อความใดทั้งสิ้น เครื่องคอมพิวเตอร์ต้นทาง และปลายทางจะเป็นผู้รับผิดชอบการ Echo ข้อความตามแบบฟูลดูเพล็กซ์ (Full Duplex) และฮาล์ฟดูเพล็กซ์ (Half Duplex) กันเองจนสิ้นสุดการติดต่อ

H - Hook Control เป็นคำสั่งให้โมเด็มวางสายโทรศัพท์ หรือปลดตัวเองออกจากสายโทรศัพท์และวางหูนั่นเอง ส่วนมากคำสั่งนี้จะใช้เมื่อเลิกติดต่อกับปลายทางโดยสั่ง ATH เพื่อวางหูหลังจากส่งข้อมูลจบแล้ว ก่อนใช้คำสั่งนี้เราต้องสั่งให้โมเด็มกลับมาอยู่ในสภาวะรับคำสั่งเสียก่อน

H0 โมเด็มจะตัดสายโทรศัพท์

H1 โมเด็มจะใช้สายโทรศัพท์

I - Identifications คำสั่งนี้ใช้ในการตรวจสอบ Product code,Firmware Version ของโมเด็ม และการทดสอบ Firmware โดยมีรูปแบบของคำสั่งดังนี้

I0 ใช้ในการแสดง Product Code ของโมเด็ม

I1 ใช้ในการแสดง Firmware Version ของโมเด็ม

I2 ใช้ในการทดสอบ Checksum ของโมเด็มและส่งรหัสตอบรับ OK หรือ ERROR กลับมา

N? - Read Redial Times Setting คำสั่งนี้ใช้ในการเรียกดูจำนวนครั้งของการหมุนซ้ำที่ถูกเซ็ตไว้ครั้งก่อน

N=n - Set Redial Times คำสั่งนี้เป็นตัวกำหนดว่า ในกรณีที่สายไม่ว่าง จะให้โมเด็มทำการหมุนทวนเลขหมายเดิมกี่ครั้ง ซึ่ง n อยู่ในช่วง 0-15 มีหน่วยเป็นครั้ง ซึ่งค่าปกติคือ n = 0 แสดงว่า โมเด็มจะไม่มีการหมุนทวนเลขหมาย หลังจากที่สามารถติดต่อได้แล้ว ตัวนับจำนวนครั้งของการหมุนทวนเลขหมายจะถูกเคลียร์เป็น 0 นั่นคือ โมเด็มจะรู้ว่าไม่ต้องหมุนทวนเลขหมายอีกแล้ว

N5? - Read Redial Interval Setting คำสั่งนี้ใช้แสดงช่วงเวลาระหว่างการหมุนทวนเลขหมาย 2 ครั้งที่ติดกันที่ถูกเซ็ตไว้ครั้งก่อน

N5=n - Set Redialing Interval คำสั่งนี้ใช้ในการกำหนดช่วงเวลาระหว่างการหมุนทวนเลขหมาย 2 ครั้งที่ติดกัน โดยการให้ n มีค่าในช่วง 0-255 มีหน่วยเป็น 2 วินาที ซึ่งค่าปกติคือ n=2 ตัวอย่าง N5=45 คือ โมเด็มจะทำการหมุนทวนเลขหมายทุก ๆ 90 วินาที

P - Pulse Dialing คำสั่งนี้จะใส่ไว้ก่อนหน้าของหมายเลขโทรศัพท์ เพื่อสั่งให้โมเด็มหมุนหมายเลขโทรศัพท์ที่ตามหลังแบบพัลส์(โทรศัพท์แบบหมุน) โดยกำหนดในเวลา 1 วินาที จะมีพัลส์ได้ 10 พัลส์

ตัวอย่าง ATDT9,P78222-0002

คำสั่งนี้คือสั่งให้โมเด็มหมุนเลข 9 แบบโทน (Tone) และหยุดคอย 2 วินาที (",") หลังจากนั้นจะทำการหมุนหมายเลข 782-0002 แบบพัลส์

Sr - Read Register Value คำสั่งนี้ใช้ในการดูค่าของรีจิสเตอร์ r ที่เซ็ตไว้

ตัวอย่าง ATS5?S7? คำสั่งนี้คือ ให้โมเด็มแสดงค่าของรีจิสเตอร์ S5 และ S7

Sr=n - Set Register Value คำสั่งนี้ใช้เซ็ตค่าในรีจิสเตอร์ r ด้วยค่า n

ตัวอย่าง ATS0 = 3 คำสั่งนี้คือ รีจิสเตอร์ 0 มีค่าเป็น 3

หมายเหตุ เมื่อมีการปิดโมเด็ม หรือมีการรีเซ็ตโดยคำสั่ง ATZ9 ค่าที่เซ็ตไว้ในรีจิสเตอร์ทั้งหมดจะกลับเป็นค่าเดิมเหมือนกับในตอนแรกที่เก็บไว้ใน NVRAM หรือ Firmware ROM ของโมเด็ม

T - Touch Tone Dialing คำสั่งนี้จะใส่ไว้ก่อนหมายเลขโทรศัพท์ เพื่อสั่งให้โมเด็มทำการหมุนหมายเลขโทรศัพท์ที่ตามหลังแบบโทน (Tone) ความเร็วของการหมุนแบบโทนนี้จะมีรีจิสเตอร์ S11 เป็นตัวกำหนด ซึ่งปกติค่าที่ตั้งไว้ประมาณ 5.26 ตัวเลขต่อวินาที

ตัวอย่าง ATDP9,T782-0002

คำสั่งนี้คือสั่งให้โมเด็มทำการหมุนเลข 9 แบบพัลส์ จากนั้นหยุดคอย 2 วินาที (",") แล้วทำการหมุนหมายเลข 782-0002 แบบโทน

W - Wait for a Second Dial Tone คำสั่งนี้จะใส่ไว้ในชุดคำสั่งของการหมุน ซึ่งทำให้โมเด็มหยุดการหมุนแบบโทนอย่างต่อเนื่องเป็นเวลา 3 วินาที ก่อนที่จะมีการหมุนตัวเลขตัวถัดไป ซึ่งโมเด็มจะแสดง NO DIALTONE ถ้าตรวจจับโทนไม่ได้ หรือจะเป็น BUSY เมื่อสายไม่ว่าง

"," - Pause (เครื่องหมายคอมมา) คำสั่งนี้จะสั่งโมเด็มหยุดคอยด้วยช่วงเวลาหนึ่ง ๆ ที่กำหนดไว้ในรีจิสเตอร์ S8 จากนั้นจึงค่อยทำคำสั่งต่อไปตามชุดของคำสั่ง โดยช่วงเวลาของการหยุดปกติคือ 2 วินาที และสามารถเปลี่ยนค่านี้ได้โดยการเซ็ตค่าในรีจิสเตอร์ S8 ถ้าต้องการหยุดคอยเป็นเวลานาน ๆ ก็ใช้คำสั่งนี้ต่อ ๆ กันได้

บทที่ 5

หน่วยแสดงผล

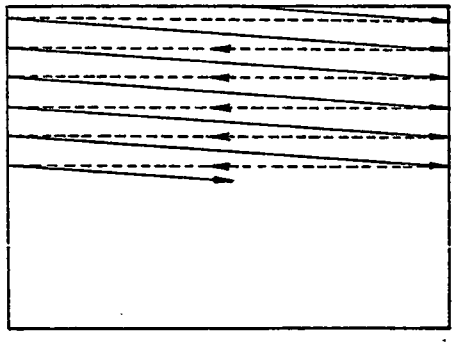
หลักการพื้นฐานของโทรทัศน์

การเกิดภาพของโทรทัศน์ ใช้หลักการของการนำภาพมาแสดงต่อกัน โดยให้แต่ละภาพมีลักษณะต่างกันเล็กน้อย ในภาพที่เกิดขึ้นมานั้นจะประกอบด้วยจุดเล็ก ๆ หลาย ๆ จุด โดยที่จุดเหล่านี้เรียกว่าพิกเซล (PIXEL)

ภาพที่เกิดขึ้นในโทรทัศน์ จะเกิดจากเส้นขวางเล็ก ๆ ในแนวนอนจำนวนมาก โดยที่ใน 1 ภาพจะมีจำนวนเส้นสแกนแตกต่างกันไป เช่น ระบบที่ใช้ในประเทศไทย ใช้ระบบการสแกน 625 เส้น

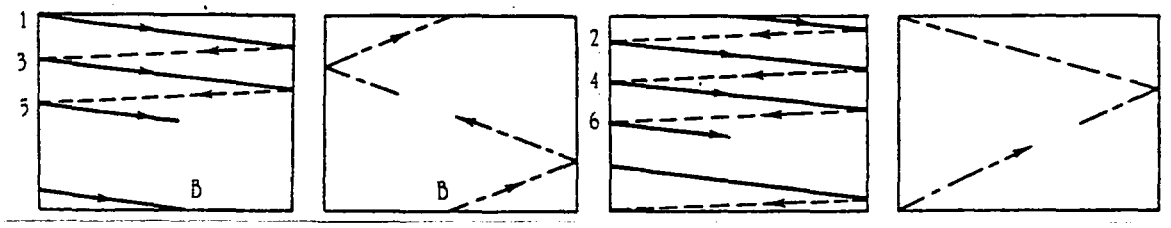
การสแกนเกิดขึ้นจากหลอดภาพ (ซึ่งมีโครงสร้างคล้ายหลอดสุญญากาศ) ปล่อยอิเล็กตรอนมาจากคาโทด แล้วดึงลำอิเล็กตรอนไปกระทบจอของหลอดภาพ ซึ่งฉาบด้วยสารเรืองแสง โดยการทำให้ลำอิเล็กตรอนเบี้ยไปตามทางที่ต้องการ โดยใช้การเหนี่ยวนำทางแม่เหล็กไฟฟ้า ซึ่งสามารถแบ่งแยกการสแกนตามลักษณะวิธีเป็น 2 แบบ

1. <u>การสแกนแบบก้าวหน้า</u> (Progressive scanning หรือเรียกว่า Non-Interlace scanning) จะเป็นการสแกนตั้งแต่จุดเริ่มต้นโดยเริ่มที่ขอบบนซ้ายไล่ไปเรื่อย ๆ ทีละเส้นจนจบที่เส้นสุดท้ายแล้วจะย้อนกลับ (retrace) มาเริ่มต้นใหม่ ดังรูป 5.1



รูปที่ 5.1 แสดงการสแกนแบบก้าวหน้า (Progressive scanning)

2. <u>การสแกนแบบสลับเส้น</u> (Interlace scanning) เริ่มตั้งแต่ขอบบนซ้ายไล่ลงไปจนจบเหมือนแบบที่ 1 แต่จะเกิดการสแกนเฉพาะเส้นคี่เท่านั้น แล้วจะเกิดการย้อนกลับ (retrace) มาที่จุดกึ่งกลางของเส้นแรก แล้วสแกนต่อมาที่เส้นคู่เหมือนแบบที่ 1 จนจบก็จะ retrace ใหม่กลับไปที่จุดเริ่มต้นอีกครั้ง ดังรูป 5.2

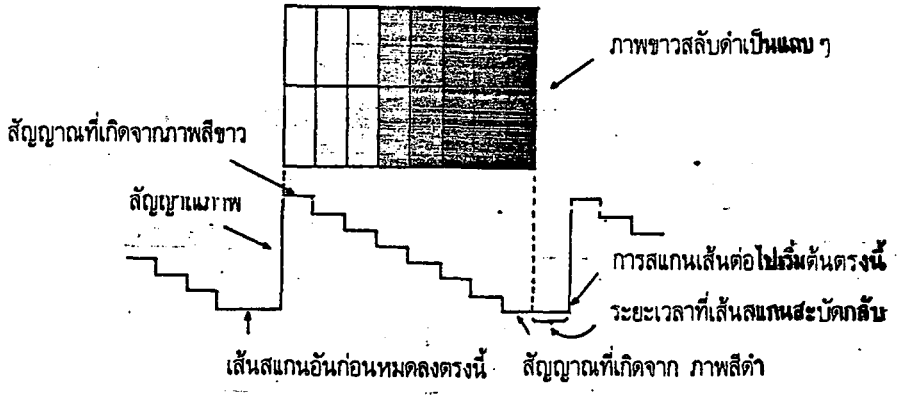


รูปที่ 5.2 แสดงการสแกนแบบสลับเส้น (Interlace scanning)

เวลาของการย้อนกลับทั้งแนวตั้งและแนวนอน จะเกิดเป็นช่วงเวลาสั้น ๆ แต่จะสร้างสัญญาณรบกวนเกิดขึ้น จึงต้องมีสัญญาณแบลงค์กิ้งเพื่อจะมาลบออกอีกครั้ง วิธีการสแกนแบบนี้เรียกว่าการสแกนแบบราสเตอร์ (Raster Scanning)

รายละเอียดของสัญญาณต่างๆ

1. **สัญญาณจอภาพ** เป็นสัญญาณที่ส่งไปเพื่อให้เกิดภาพ โดยที่สีขาวจะเป็นภาพที่สว่างมากที่สุด และสีดำคือภาพที่ไม่มีความสว่างเลย สามารถอธิบายได้จากเกร์ยสเกล ซึ่งจะเห็นได้ว่าสัญญาณที่แรงที่สุด จะมีความสว่างมากที่สุด โดยจะมีการลดหลั่นเป็นแบบขั้นบันได ดังรูป 5.3



รูปที่ 5.3 แสดงระดับเกร์ยเสกล

2. **สัญญาณแบลงกิ้ง** เป็นสัญญาณที่ส่งไปเพื่อลบเส้นสะบัดกลับ มี 2 แบบ คือ สัญญาณแบลงกิ้งในแนวตั้ง (Vertical Blanking) และสัญญาณแบลงกิ้งในแนวนอน (Horizontal Blanking) โดยที่เวลา 1/15625 วินาที จะมีการลบเส้นทางแนวนอน 1 ครั้งและใน 1/50 วินาที จะมีการลบเส้นทางแนวตั้งหนึ่งครั้ง โดยที่จะส่งไปพร้อมกับสัญญาณซิงโครนัส

3. **สัญญาณซิงโครนัส** เป็นสัญญาณที่ส่งมา เพื่อช่วยให้สัญญาณต่าง ๆ ของเครื่องส่งและเครื่องรับทั้งแนวตั้งและแนวนอนทำงานได้สอดคล้องกันมี 2 ชนิด คือ การซิงโครนัสแนวนอน (Horizontal Synchronous) และการซิงโครนัสแนวตั้ง (Vertical Synchronous) จะมีค่าความถี่เท่ากับความถี่ของกระแสรูปฟันเลื่อย ที่ใช้ในวงจรหักเหตามแนวนั้น ๆ

4) **สัญญาณอีควอไลซิ่ง** เป็นสัญญาณที่ช่วยให้สัญญาณซิงโครนัสในแนวตั้ง และแนวนอนทำงานได้พร้อมกัน

บทที่ 6

หลักการทำงานของระบบ

หลักการทำงานของส่วนแสดงผล

ส่วนการแสดงผลมีการจัดแผนผังได้ดังรูปที่ 6.1 ซึ่งมีการทำงานของส่วนแสดงผลเป็นดังนี้

1. ซีพียูต้องส่งข้อมูลไปเซ็ตค่ารีจิสเตอร์ของ 6845 CRTC เพื่อกำเนิดสัญญาณต่าง ๆ ตามค่าที่ต้องการ เช่น HSYNC , VSYNC โดยส่งข้อมูลผ่านทางเกตข้อมูล

2. เมื่อต้องการแสดงข้อมูล ซีพียูจะต้องเขียนข้อมูลนั้นมาเก็บไว้ที่ส่วนของวีดีโอ-แรม (Video RAM) โดยเก็บในรูปของรหัสแอสกี (ASCII Code) สำหรับการเข้าถึงแอดเดรสของวีดีโอแรม (Video RAM) นี้จะผ่านตัวมัลติเพล็กซ์ (Multiplexer) ซึ่งทำหน้าที่มัลติเพล็กซ์แอดเดรสระหว่างซีพียู กับ 6845 CRTC และข้อมูลที่ต้องการแสดงจะถูกส่งผ่านทางเกตข้อมูล

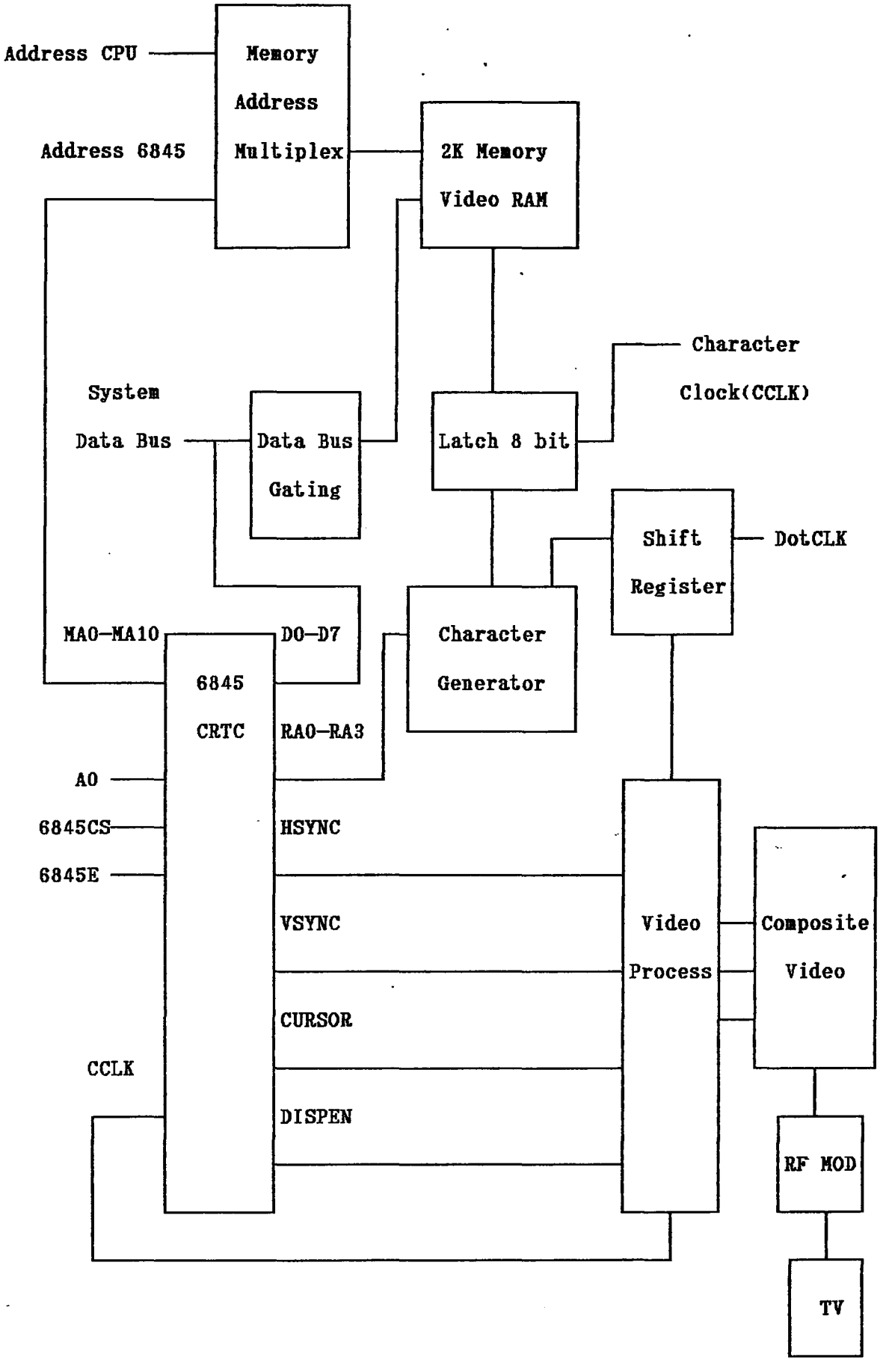
3. หลังจากที่ซีพียูเขียนข้อมูลที่วีดีโอแรมแล้ว 6845 CRT Controller จะเป็นตัวควบคุมการแสดงผลแทน โดยที่ 6845 CRTC จะส่งแอดเดรสของข้อมูลที่ต้องการแสดงไปผ่านที่ตัวมัลติเพล็กซ์ ข้อมูลที่ต้องการแสดงซึ่งเป็นรหัสแอสกี (ASCII Code) จะถูก6845 CRTC อ่านออกมาและถูกส่งผ่านไปในที่แลทช์ 8 บิท

4. 6845 CRTC จะส่งสัญญาณที่แสดงแถว คือ RA0-RA3 ของเมตริกซ์แบบจุด (Dot Matrix)ของตัวกำเนิดตัวอักษร (Character Generator) ซึ่งเป็นที่เก็บฟอนท์ (Font) ของตัวอักษรต่าง ๆ

5. เมตริกซ์แบบจุด (Dot Matrix) แต่ละแถวจะถูกอ่านออกมาทีละแถวตามค่า RA0-RA3 และถูกส่งไปที่ชิฟท์รีจิสเตอร์ (Shift Register) ซึ่งจะทำการเลื่อนข้อมูลออกไปผ่านส่วนประมวลผลสัญญาณวีดีโอ

6. สัญญาณจากส่วนประมวลผลวีดีโอ ซึ่งเป็นสัญญาณคอมโพสิทวีดีโอ (Composite Video) จะถูกมอดดูเลทด้วยสัญญาณคลื่นความถี่วิทยุ (RF) และส่งผ่านเข้าทางช่องสัญญาณคลื่นความถี่วิทยุ (RF) ของโทรทัศน์เพื่อแสดงผล

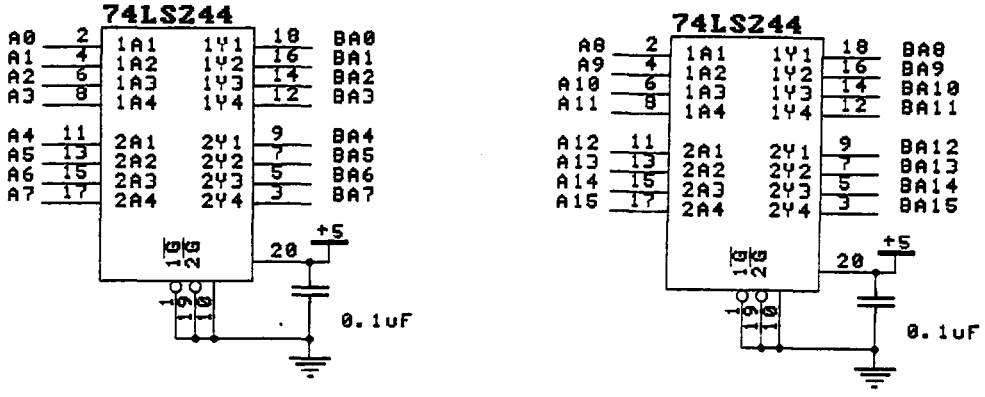
7. เมื่อเมตริกซ์แบบจุด (Dot Matrix) ถูกเลื่อน (Shift) ออกมาจนครบทุกแถว ก็จะปรากฏเป็นตัวอักษรตามที่ต้องการ



รูปที่ 6.1 แสดงแผนผังการทำงานของการแปลงสัญญาณในส่วนแสดงผล

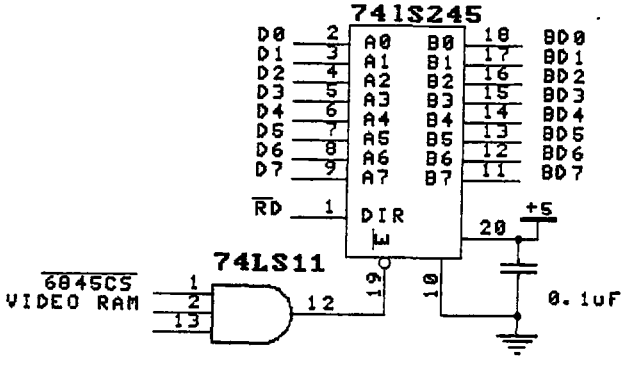
ลักษณะของวงจรต่าง ๆ

1. บัฟเฟอร์แอดเดรส (Address Buffer) และบัฟเฟอร์ข้อมูล (Data Buffer) เป็นส่วนเชื่อมต่อกับระบบบัสของไมโครโปรเซสเซอร์ โดยที่บัสแอดเดรสจะถูกต่อเข้ากับบัฟเฟอร์ทิศทางเดียวขนาด 8 บิท 2 ตัว ซึ่งในโครงงานนี้ได้ใช้ไอซีเบอร์ 74LS244 ดังรูป 6.2



รูปที่ 6.2 แสดงวงจรของบัฟเฟอร์ทิศทางเดียว และการต่อบัสแอดเดรสเข้ากับบัฟเฟอร์

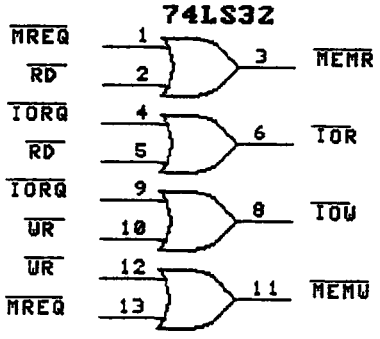
บัสข้อมูลจะถูกต่อกับบัฟเฟอร์สองทิศทาง ขนาด 8 บิท ไอซีเบอร์ 74LS245 ดังแสดงในรูปที่ 6.3



รูปที่ 6.3 แสดงวงจรของบัฟเฟอร์สองทิศทาง และการต่อบัสข้อมูลเข้ากับบัฟเฟอร์

จากรูปจะเห็นว่ามีสัญญาณ 6845CS และ VIDEO RAM ผ่านเกทแอนด์ (AND Gate) ซึ่งใช้ไอซีเบอร์ 74LS11 เพื่ออีนาเบิลการทำงานของบัฟเฟอร์สองทิศทาง (ไอซี 74LS245)

2. ส่วนสร้างสัญญาณในการเข้าถึงพอร์ทอินพุทเอาท์พุท และหน่วยความจำ จะถูกต่อผ่านเกทออร์ (OR Gate) ไอซีเบอร์ 74LS32 เพื่อให้มีการซิงโครไนซ์ของสัญญาณต่าง ๆ ดังแสดงดังรูป 6.4



รูปที่ 6.4 แสดงการสร้างสัญญาณในการเข้าถึงพอร์ทอินพุทเอาท์พุทและหน่วยความจำ

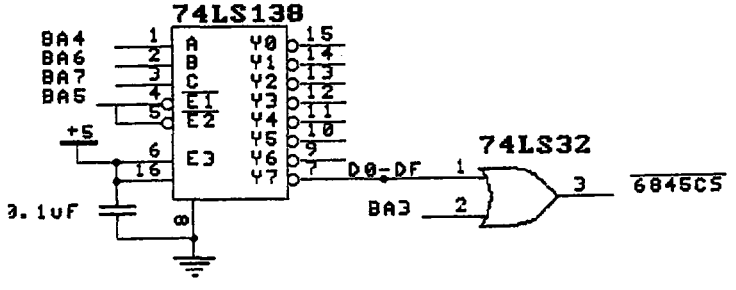
3. ส่วนถอดรหัสพอร์ทและหน่วยความจำ เพื่อใช้ในการเข้าถึงแอดเดรสต่าง ๆ ของพอร์ทและหน่วยความจำ ซึ่งจะมีการถอดรหัสโดยผ่านไอซีเบอร์ 74LS138 ดังตารางที่ 6.1

ตารางที่ 6.1 แสดงการถอดรหัสแอดเดรสของพอร์ทและหน่วยความจำ

A_7	A_6	A_5	A_4	A_3	A_2	A_1	A_0
1	1	0	1	0	X	X	0
1	1	0	1	0	X	X	1

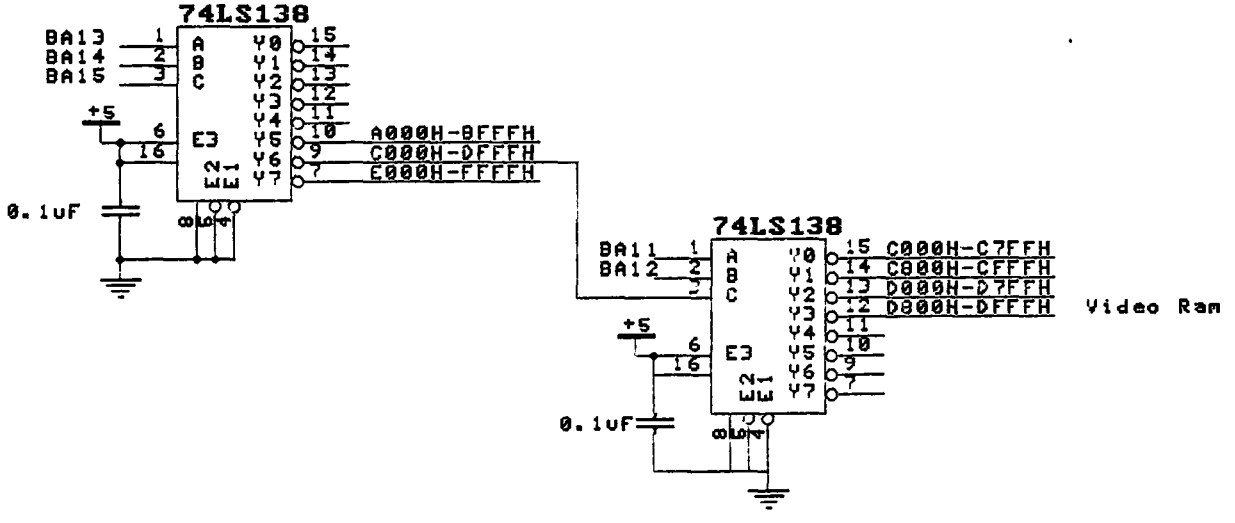
X : Don't care

โดยในกรณีที่ติดต่อด้วยรีจิสเตอร์แอดเดรสของ 6845CRT Controller ใช้แอดเดรส 0D0H และการติดต่อกับรีจิสเตอร์ข้อมูลของ 6845CRT Controller ใช้แอดเดรส 0D1H



รูปที่ 6.5a แสดงการถอดรหัสพอร์ท

การถอดรหัสหน่วยความจำ (Decode Memory) จะใช้สัญญาณ BA11-BA15 ในการอ้างแอดเดรสของหน่วยความจำ ซึ่งหน่วยความจำที่ต้องการติดต่อ คือวีดีโอแรม (VIDEO RAM) มีขนาด 2 กิโลไบท์ การถอดรหัสจะถูกต่อผ่าน 74LS138 2 ตัว เพื่อให้ได้แอดเดรสขนาด 2 กิโลไบท์โดยแอดเดรสของที่วีดีโอแรมที่ได้คือ 0D800H-0DFFFH ดังรูปที่ 6.5b

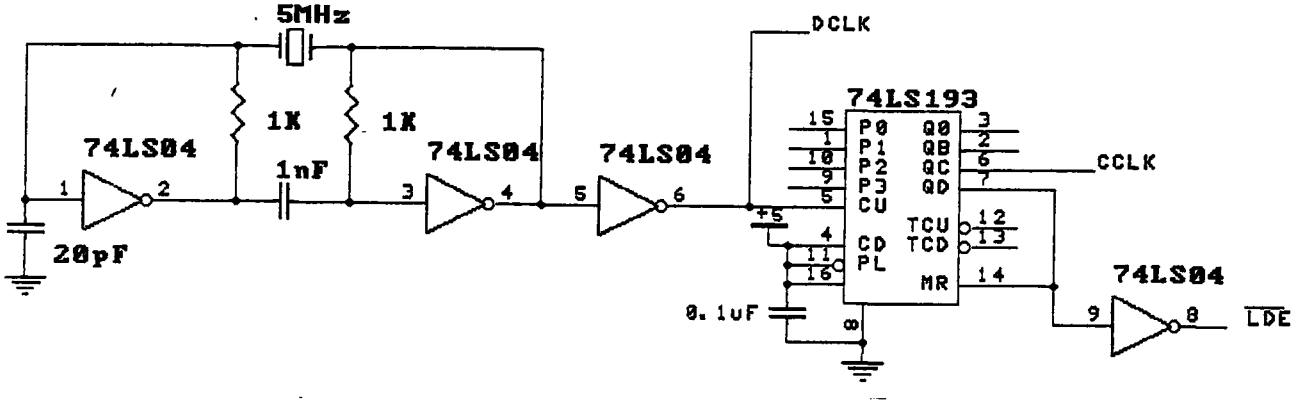


รูปที่ 6.5b แสดงวงจรการถอดรหัสหน่วยความจำ

4. ส่วนสร้าง Dot Clock และ Character clock โดยที่ Dot Clock จะถูกต่อกับชิฟท์รีจิสเตอร์ เพื่อใช้ในการเลื่อนจุด (Dot) ของตัวอักษร ส่วน Character Clockจะถูกต่อกับ 6845 CRTC เพื่อนำไปสร้างสัญญาณต่างๆ (เช่น HSYNC,VSYNC) และต่อเข้ากับแลทช์ ในการสร้าง Dot Clock ให้ดูการคำนวณในเรื่อง 6845 CRTC

ส่วน Character Clock เป็น Clock ที่ใช้ในการดึงตัวอักษรแต่ละตัวออกมาแสดง ซึ่งจะมีขนาดเป็น 1/8 ของ Dot Clock

ในการสร้าง Dot Clock จึงใช้คริสตอลขนาด 6 MHz ต่อกับอินเวอร์เตอร์ (ไอซีเบอร์ 74LS04) เพื่อกำเนิดความถี่ 6 MHz ออกมา โดย 74LS193 ที่ใช้ทำหน้าที่เป็นวงจรหารแปด เพื่อให้ Character Clock ดังรูปที่ 6.6



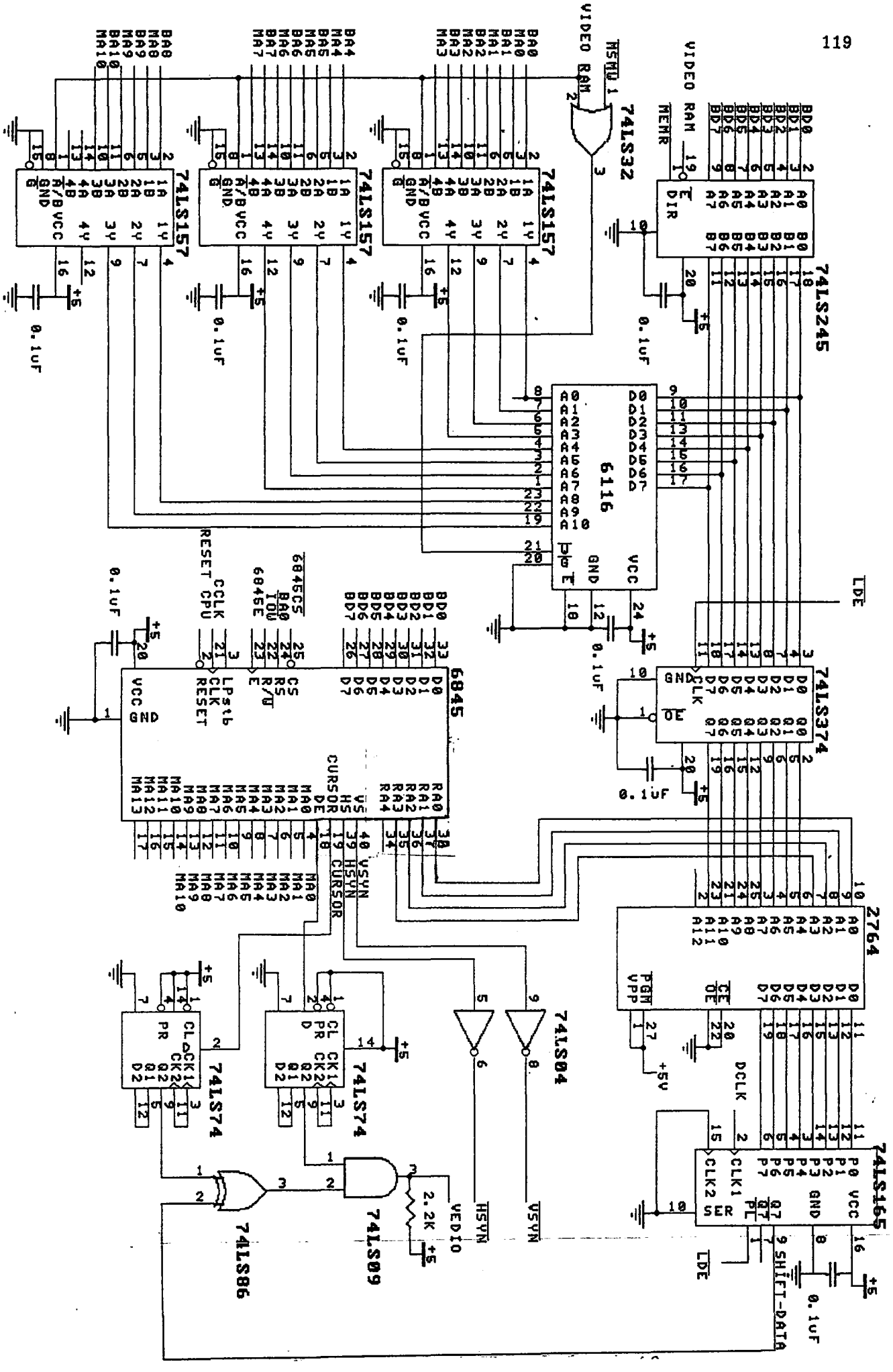
รูปที่ 6.6 แสดงวงจรสร้าง Clock

5. ส่วนของการมัลติเพล็กซ์แอดเดรส ใช้ไอซีเบอร์ 74LS157 3 ตัว ในการมัลติเพล็กซ์แอดเดรสระหว่างซีพียูกับ 6845 CRTC โดยมีขาควบคุมการมัลติเพล็กซ์คือขา A/B โดยที่ขานี้จะถูกต่อกับขาสัญญาณ VIDEO RAM ซึ่งมาจากการถอดรหัสหน่วยความจำ ในกรณีขา A/B เป็น LOW ซีพียูจะเป็นตัวเข้าถึง VIDEO RAM ดังรูปที่ 6.7

6. VIDEO RAM เป็น RAM ขนาด 2 กิโลไบท์ (ไอซีเบอร์ 6116) ใช้ในการเก็บข้อมูลที่ต้องการแสดงผล

7 Gate Buffer เป็นตัวแยกสัญญาณระหว่าง Primary Data Bus กับSecondary Data Bus เพื่อไม่ให้มีการรบกวนกันของสัญญาณข้อมูลระหว่างที่ซีพียูทำงาน หรือ 6845 CRT Controller ทำงาน โดยในที่นี้ใช้ไอซีเบอร์ 74LS245 เป็น Gate Buffer ดังรูป 6.7

8. แลทช์ (ไอซีเบอร์ 74LS374) เป็นตัวที่ใช้ในการแลทช์ข้อมูลที่เป็นค่ารหัสแอสกีของข้อมูลที่เก็บไว้ใน VIDEO RAM ซึ่งสัญญาณที่ผ่านแลทช์ข้อมูลจะรวมกับสัญญาณ RA0-RA3ของ 6845 CRT Controller เพื่อชี้แอดเดรสใน Character Generator แล้วดึงข้อมูลที่เป็น Dot Matrix ออกมา ดังรูปที่ 6.7



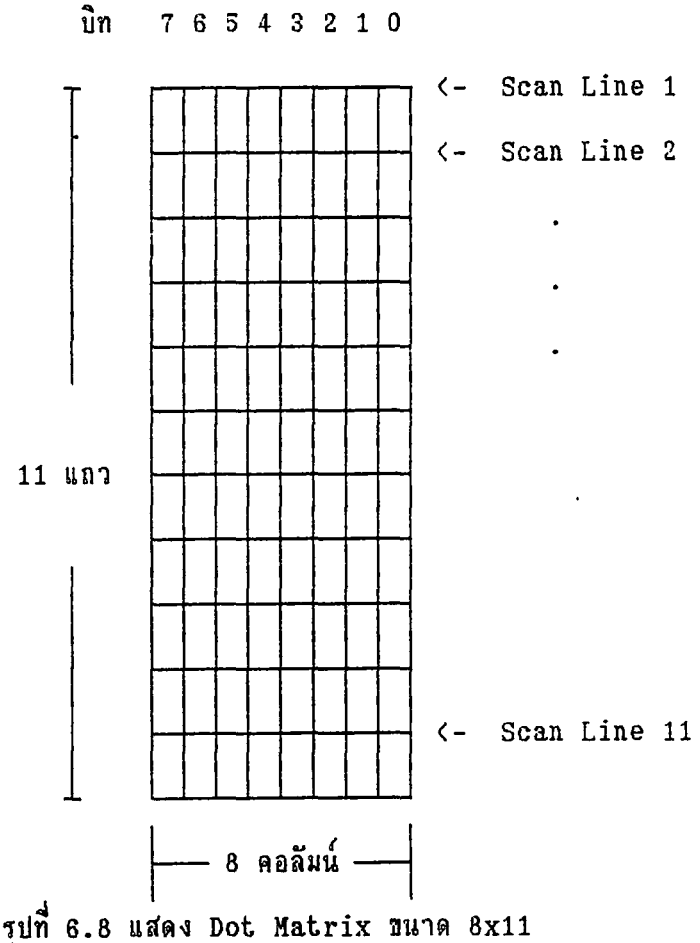
รูปที่ 6.7 แสดงวงจรการทำงานในส่วนแสดงผล

9. Character Generator เป็น ROM ที่ใช้เก็บฟอนท์ (Font) ของตัวอักษรต่างๆ ซึ่งมีลักษณะดังนี้

การจัดแอดเดรสของ Character Generator

ในการเข้าถึงแอดเดรสของ Character Generator จะใช้ข้อมูลที่เป็นรหัสแอสกี (ASCII Code) รวมกับสัญญาณ Scan Line ของ 6845 CRTC คือ RA0-RA3 ดังตาราง 6.2

ในการส่งสัญญาณ RA0-RA3 นั้น 6845 CRTC จะส่งสัญญาณ Scan Line RA0-RA3 ออกมาค้างอยู่ค่าหนึ่ง จนกว่าจะแสดงฟอนท์ครบจำนวนตัวอักษรต่อ 1 บรรทัด 6845 CRTC จึงส่ง RA0-RA3 ค่าต่อไปออกมา เมื่อ 6845 CRTC ส่ง Scan Line RA0-RA3 ครบทั้ง 11 แถว ก็จะปรากฏตัวอักษรของแถวนั้นออกมา และเริ่มแสดงตัวอักษรบรรทัดต่อไป



รูปที่ 6.8 แสดง Dot Matrix ขนาด 8x11

สำหรับการจัดเก็บข้อมูล Dot Matrix ใน Character Generator นั้นจึงใช้หลักการเอาค่ารหัสแอสกี 8 บิท รวมกับค่า RA0-RA3 จำนวน 4 บิท มาเป็นแอดเดรสในการจัดเก็บ โดย RA0-RA3 เป็น 4 บิทต่ำ และค่ารหัสแอสกีอีก 8 บิทเป็น 8 บิทสูง จึงทำให้แอดเดรสของ Character Generator เท่ากับ 8 + 4 = 12 เส้น ดังนั้น Character Generator ต้องมีความจุอย่างน้อย 2^{12} = 4 กิโลไบท์

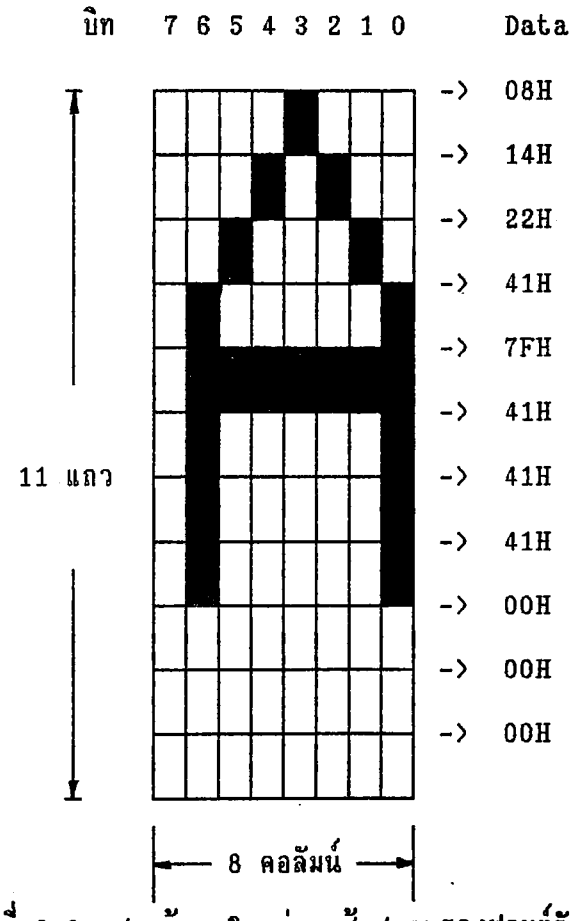
<u>ตัวอย่าง</u> สมมติ อักษร A ซึ่งมีรหัสแอสกีเป็น 41H ค่ารหัสแอสกีนี้จะไปรวมกับ RA0-RA3 ซึ่งมี 4 บิท โดยฟรอนท์ที่ใช้มี 11 Scan Line ดังนั้น แอดเดรสที่ใช้เก็บ Dot Matrix ของอักษร 'A' คือ X410H-X41AH ดังตารางที่ 6.2

ตารางที่ 6.2 แสดงแอดเดรสที่ใช้ในการเข้าถึง Character Generator

แอดเดรส												ข้อมูล	ความหมาย
รหัส ASCII								RA					
B7	B6	B5	B4	B3	B2	B1	B0	R3	R2	R1	R0		
0	1	0	0	0	0	0	1	0	0	0	0	08H	ASCII 41H เส้นสแกนที่ 1
0	1	0	0	0	0	0	1	0	0	0	1	14H	ASCII 41H เส้นสแกนที่ 2
0	1	0	0	0	0	0	1	0	0	1	0	22H	ASCII 41H เส้นสแกนที่ 3
0	1	0	0	0	0	0	1	0	0	1	1	41H	ASCII 41H เส้นที่สแกน 4
0	1	0	0	0	0	0	1	0	1	0	0	7FH	ASCII 41H เส้นที่สแกน 5
0	1	0	0	0	0	0	1	0	1	0	1	41H	ASCII 41H เส้นที่สแกน 6
0	1	0	0	0	0	0	1	0	1	1	0	41H	ASCII 41H เส้นที่สแกน 7
0	1	0	0	0	0	0	1	0	1	1	1	41H	ASCII 41H เส้นที่สแกน 8
0	1	0	0	0	0	0	1	1	0	0	0	00H	ASCII 41H เส้นที่สแกน 9

ตารางที่ 6.2 แสดงแอดเดรสที่ใช้ในการเข้าถึง Character Generator (ต่อ)

แอดเดรส												ข้อมูล	ความหมาย
รหัส ASCII								RA					
B7	B6	B5	B4	B3	B2	B1	B0	R3	R2	R1	R0		
0	1	0	0	0	0	0	1	1	0	0	1	00H	ASCII 41H เส้นที่สแกน 10
0	1	0	0	0	0	0	1	1	0	1	0	00H	ASCII 41H เส้นที่สแกน 11
0	1	0	0	0	0	0	1	ไม่ใช้งาน				–	–

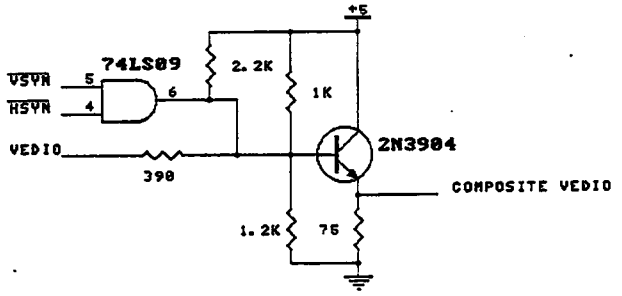


รูปที่ 6.9 แสดงข้อมูลในแต่ละเส้นสแกนของฟอนท์ตัวอักษร "A"

10. ชิฟท์รีจิสเตอร์ (Shift Register) เป็นส่วนที่ทำหน้าที่เลื่อนข้อมูลเมตริกซ์แบบจุด (Dot Matrix) ที่ผ่านมาจาก Character Generator เพื่อส่งไปแสดงผล โดยที่ในโครงงานนี้ใช้ไอซีเบอร์ 74LS165 เป็นชิฟท์รีจิสเตอร์ ดังรูปที่ 6.7

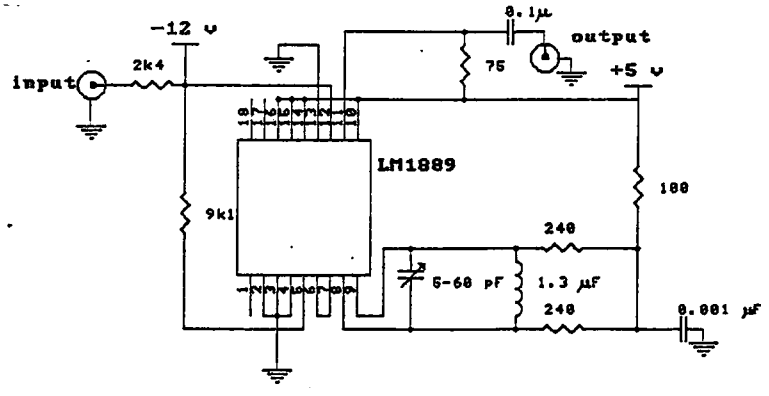
11. 6845 CRTC เป็นตัวควบคุมการแสดงผล โดยจะมีการสร้างสัญญาณต่าง ๆ ที่จำเป็นต่อการแสดงผลออกมา เช่น HSYNC,VSYNC,CURSOR และ DISPEN เป็นต้น โดยสัญญาณ CURSOR และ DISPEN จะใช้ร่วมกับสัญญาณ Dot ที่มาจากชิฟท์รีจิสเตอร์ เพื่อสร้างสัญญาณวีดีโอ ดังรูปที่ 6.7

12. ส่วนสร้างสัญญาณ Composite Video ใช้ทรานซิสเตอร์เพื่อรวมสัญญาณ วีดีโอ และ HSYNC กับ VSYNC ที่ผ่านจากเกทแอนด์ (AND Gate) ใช้ไอซีเบอร์ 74LS09 เพื่อสร้างเป็นสัญญาณคอมโพสิทวีดีโอ (Composite Video)ดังรูปที่ 6.10



รูปที่ 6.10 แสดงส่วนสร้างสัญญาณ Composite Video

13. RF Modulator เป็นส่วนที่ใช้สร้างสัญญาณพาหะเพื่อทำการมอดดูเลทกับสัญญาณ Composite Video แล้วส่งเข้าช่องสายอากาศของโทรทัศน์ในการแสดงผล โดยในที่นี้ใช้ไอซีเบอร์ LM1889 TV Modulator เป็นตัวสร้างสัญญาณพาหะ และทำการมอดดูเลทกับสัญญาณคอมโพสิทวีดีโอ (Composite Video) ซึ่งมีวงจรดังรูปที่ 6.11



รูปที่ 6.11 แสดงส่วนของ RF Modulator

หลักการทำงานของวงจรคีย์บอร์ด

คีย์บอร์ด (Keyboard) เป็นอุปกรณ์อินพุทที่สำคัญที่ใช้ในการติดต่อระหว่างซีพียูกับผู้ใช้ ซึ่งวิธีการติดต่อมีด้วยกัน 2 วิธีคือ

1. การโพลลิ่ง (Polling)

ในกรณีที่ข้อมูล (ในลักษณะไบท์) ที่ส่งจากคีย์บอร์ดเข้าไปยังซีพียู การไหลของข้อมูลจะไม่สม่ำเสมอ และซีพียูก็ไม่สามารถทำนายว่าข้อมูลตัวใหม่จะมาถึงเมื่อใด แต่จุดบกพร่องดังกล่าวนี้สามารถใช้การโพลแก้ไขได้ โดยขบวนการนี้จะทำการตรวจสอบคีย์บอร์ดว่ามีข้อมูลส่งเข้ามาหรือไม่ โดยจะทำการตรวจสอบตลอดเวลา การตรวจสอบการรับข้อมูลจะจัดการกับข้อมูลที่รับเข้ามาด้วยความเร็วที่สูงกว่าอัตราเร็วของข้อมูลที่ส่งเข้ามาทางคีย์บอร์ด ดังนั้นข้อมูลจึงไม่ขาดหายไป พบว่าซีพียูจะคอยรับข้อมูลที่ป้อนเข้ามาจากคีย์บอร์ดเสมอ นั่นคือ ซีพียูจะสูญเสียเวลาไปประมาณ 90 % ของคาบเวลา (T) ไปกับ "Dry Poll" ซึ่งก็คือช่วงเวลาที่ซีพียูทำการส่งสัญญาณการโพลออกไปตรวจสอบที่พอร์ทต่าง ๆ แต่ปรากฏว่าไม่มีข้อมูลถูกป้อนเข้ามา สำหรับในกรณีที่ซีพียูส่งสัญญาณการโพลออกไปตรวจสอบ แล้วพบว่ามีข้อมูลที่ต้องการส่งเข้ามา จะเรียกว่า "Wet Poll"

2. การอินเทอร์รัพท์ (Interrupt)

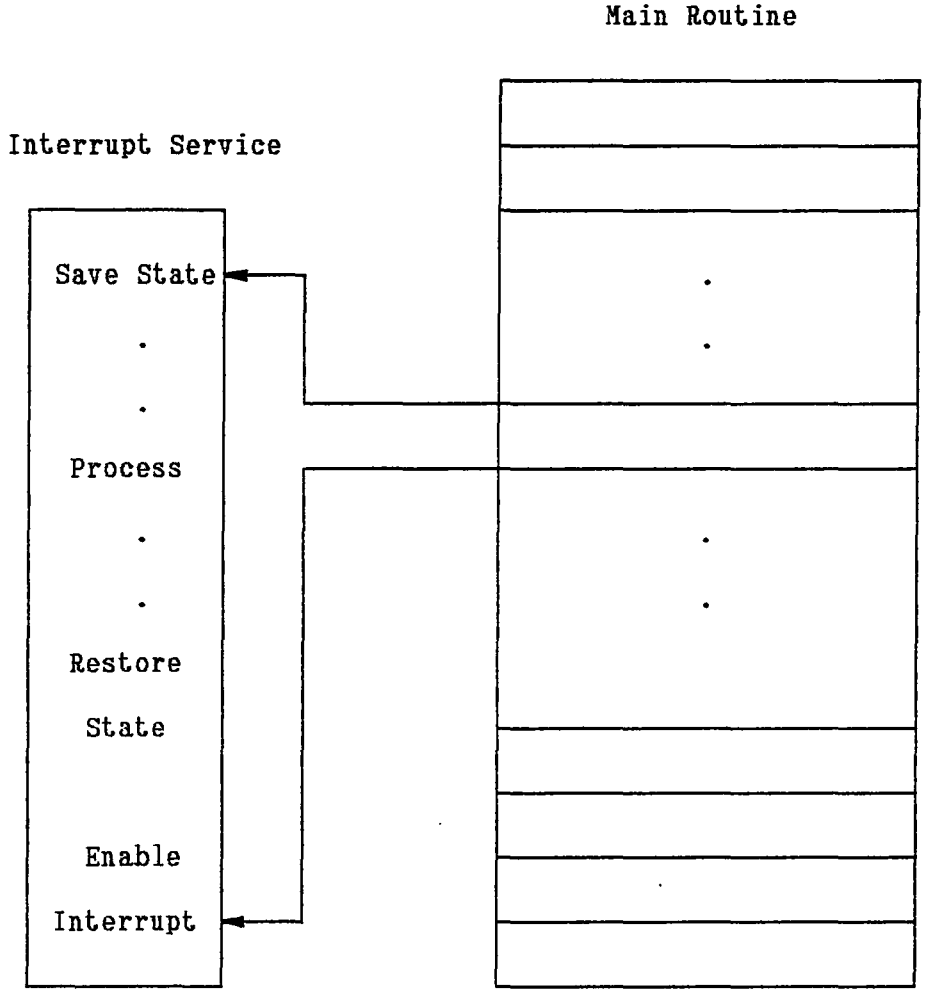
ในขบวนการอินเทอร์รัพท์ อุปกรณ์รอบข้างทุกชิ้นจะต้องปฏิบัติงานอยู่เสมอ ถ้าตัวอุปกรณ์พร้อมที่จะส่ง/รับข้อมูลได้แล้ว ก็จะส่งสัญญาณไปให้แก่ซีพียู ซึ่งถ้าเปรียบเทียบระหว่างการโพลกับการอินเทอร์รัพท์จะได้ว่า สมมติให้เหตุการณ์นี้เป็นการรอรับโทรศัพท์อยู่ ถ้าเป็นเทคนิคการโพล เราจะต้องคอยยกหูโทรศัพท์ถามว่ามีใครกำลังโทรศัพท์มาหาหรือไม่บ่อย ๆ จึงไม่จำเป็นต้องติดกริ่งโทรศัพท์ไว้ แต่ถ้าหากเราใช้เทคนิคการอินเทอร์รัพท์ โทรศัพท์จะต้องมีกริ่งและคอยจนกว่าจะมีใครโทรศัพท์มาหา กริ่งก็จะดังขึ้น เราก็รู้ได้ทันทีว่ามีคนโทรศัพท์มา จึงค่อยยกหูโทรศัพท์ขึ้นมา

ดังนั้นในลักษณะเดียวกัน ถ้าเราเปรียบโทรศัพท์เหมือนกับอุปกรณ์รอบข้างชิ้นหนึ่งนั่นคือระบบโทรศัพท์จะต้องทำงานอยู่ตลอดเวลา เพื่อคอยรับสัญญาณเรียก แล้วสร้างเสียงกริ่งให้ดังออกมา (คือเปรียบกับการอินเทอร์รัพท์) ในระบบ Z80180 และเบอร์อื่น ๆ จะมีการอินเทอร์รัพท์อยู่ 2 ลักษณะคือ มาสเคเบิลอินเทอร์รัพท์ (Maskable Interrupt) และนอนมาสเคเบิลอินเทอร์-รัพท์ (Nonmaskable Interrupt) ซึ่งมีข้อแตกต่างกันตรงที่

- Maskable Interrupt นั้นถ้าส่งสัญญาณมาขออินเทอร์รัพท์ต่อซีพียู สามารถปฏิเสธการขออินเทอร์รัพท์ได้ ซึ่งเรียกว่าสามารถดิสเอเบิลการอินเทอร์รัพท์ได้

- Nonmaskble Interrupt เป็นการอินเทอร์รัพท์ที่ซีพียูไม่สามารถปฏิเสธการขออินเทอร์รัพท์ได้

อุปกรณ์รอบข้างสามารถส่งสัญญาณขออินเทอร์รัพท์ (แบบมาสเคเบิล) ได้คือส่งสัญญาณอินเทอร์รัพท์รีเควส (Interrupt Request) นอกจากนี้ก็จะมีสัญญาณขออินเทอร์รัพท์แบบนอน-มาสเคเบิลอินเทอร์รัพท์



รูปที่ 6.12 แสดงการตอบสนองต่อการอินเทอร์รัพท์

ขั้นตอนการตอบสนองต่อการอินเทอร์รัพท์

จุดประสงค์ของการอินเทอร์รัพท์ คือการทำให้ซีพียูพักจากการทำงานในโปรแกรมหลักไว้ชั่วคราวก่อน จากนั้นจะกระโดดไปทำงานในส่วนของโปรแกรมตอบสนองต่อการอินเทอร์รัพท์ Interrupt Service Routines) และหลังจากที่เสร็จจากการทำงานในส่วนของโปรแกรมตอบสนองต่อการอินเทอร์รัพท์แล้ว ก็จะกระโดดกลับไปทำงานตามโปรแกรมหลักต่อไป การที่ซีพียูจดจำตำแหน่งของคำสั่งในโปรแกรมหลักที่ทำค้างไว้ชั่วคราวได้นั้น ซีพียูจะต้องมีการเก็บตำแหน่งเดิมนั้นไว้ที่สแตกก่อน ซึ่งสามารถสรุปขั้นตอนได้ดังต่อไปนี้

1. อุปกรณ์ภายนอกส่งสัญญาณ INT เข้ามายังซีพียู

2. ถ้าซีพียูยอมให้มีการอินเทอร์รัพท์ได้ หรือทำคำสั่ง EI (Enable Interrupt) ซีพียูก็จะตอบรับต่อการขออินเทอร์รัพท์นั้นด้วย Interrupt Acknowledge Cycle

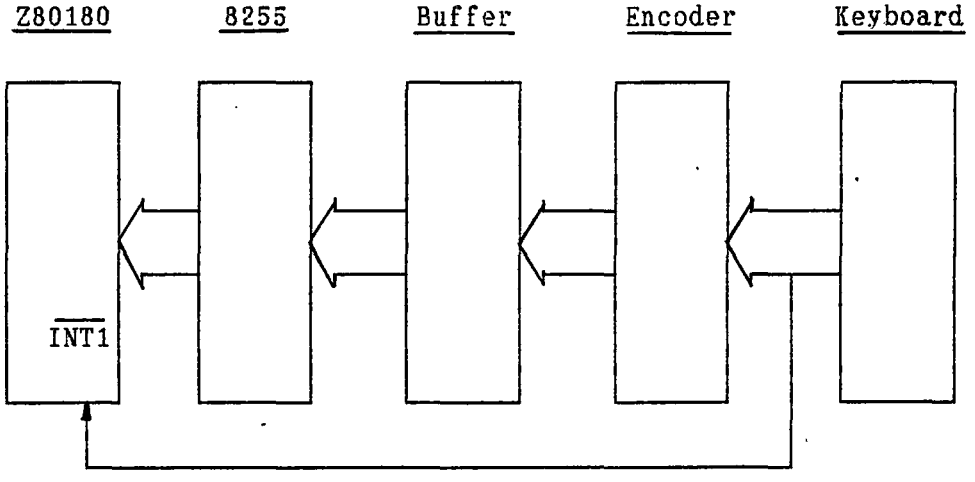
3. อุปกรณ์ภายนอกเมื่อรับรู้ว่า ซีพียูยอมให้มีการอินเทอร์รัพท์ได้ก็จะส่งอินเทอร์รัพท์เวกเตอร์เข้ามายังซีพียู

4. ซีพียูจะนำอินเทอร์รัพท์เวกเตอร์นั้นมาเป็นตัวชี้ตำแหน่งของคำสั่งที่จะต้องกระโดดไปทำ (โดยซีพียูจะต้องเก็บสถานะของตำแหน่งในโปรแกรมหลักขณะนั้นไว้ในแสตก)

5. ซีพียูจะกระโดดไปปฏิบัติตามคำสั่ง ณ ตำแหน่งที่ชี้โดยอินเทอร์รัพท์เวกเตอร์(เป็นอินเทอร์รัพท์เซอร์วิสรูทีน)

6. ก่อนจบการทำงานตามโปรแกรมอินเทอร์รัพท์เซอร์วิสรูทีน ซีพียูก็จะดึงตำแหน่งที่เก็บไว้ในสแตกกลับมา แล้วกระโดดไปยังตำแหน่งนั้นแล้วทำตามคำสั่งในโปรแกรมหลักต่อไป

การทำงานทั่วไป



รูปที่ 6.13 แสดงแผนผังการทำงานของส่วนคีย์บอร์ด

ในการทำงานของคีย์บอร์ดนั้นอาศัยหลักการของการอินเทอร์รัพท์ โดยใช้ $\overline{\text{INT1}}$ ของ Z80180 ซึ่งเป็นอินเทอร์รัพท์ที่แอคทีฟ LOW โดยเมื่อมีการกดคีย์บอร์ดจะมีการส่งสัญญาณไปทำการอินเทอร์รัพท์ซีพียู ขณะเดียวกันข้อมูลของคีย์บอร์ดที่ถูกกดจะถูกเข้ารหัสโดยตัว Encoder และส่งผ่านบัฟเฟอร์ไปให้ 8255 เมื่อซีพียูรับทราบการอินเทอร์รัพท์แล้ว ก็จะมาอ่านค่าคีย์ที่พอร์ทข้อมูลของ 8255 แล้วซีพียูก็นำข้อมูลคีย์นี้ไปเปิดตารางดูว่าเป็นคีย์อะไร เพื่อทำงานตามฟังก์ชันของคีย์นั้น ๆ ต่อไป

ตารางที่ 6.3 แสดงค่าการเข้ารหัส (Encode) ของคีย์ต่าง ๆ

คีย์ต่าง ๆ	ค่าที่เข้ารหัส
NUMBER 0	0FEH
NUMBER 1	0FDH
NUMBER 2	0FCH
NUMBER 3	0FBH
NUMBER 4	0FAH
NUMBER 5	0F9H
NUMBER 6	0F8H
NUMBER 7	0F7H
NUMBER 8	0EFH
NUMBER 9	0DFH
INFORMATION	0CFH
SEND DATA	0BFH
DIAL PHONE	0AFH
ENTER	09FH
BREAK	08FH
PRINT	07FH
PAGE DOWN	0F6H
PAGE UP	06FH

ลักษณะของวงจร

1. เอนโคดเดอร์ (Encoder) ทำหน้าที่เข้ารหัสคีย์ต่าง ๆ ใช้ไอซี 74LS147 ซึ่งเป็นตัวเข้ารหัสจากอินพุท 10 อินพุท แล้วเปลี่ยนเป็นเอาท์พุท 4 เอาท์พุท ในโครงงานนี้ใช้คีย์บอร์ดทั้งหมด 18 คีย์ จึงใช้ 74LS147 2 ตัว โดยมีการต่อดังรูปที่ 2.2.1 ข้อมูลที่ได้จากตัวเอนโคดเดอร์แต่ละตัวจะมีขนาด 4 บิท จะถูกผ่านเข้าบัฟเฟอร์ (ไอซีเบอร์ 74LS244) เป็นบัฟเฟอร์ขนาด 8 บิท และถูกส่งผ่านไปที่พอร์ทข้อมูลของ 8255 ดังรูป 6.13

2. ส่วนสร้างสัญญาณการอินเทอร์รัพท์ ใช้เกทแนนด์ (NAND Gate) ที่มี 8 อินพุท (ไอซีเบอร์ 74LS30) 2 ตัว ร่วมกับเกทแอนด์ (AND Gate) ใช้ไอซีเบอร์ 74LS11 1 ตัว โดยสัญญาณที่ผ่านจากไอซีเบอร์ 74LS30 จะต้องผ่านอินเวอร์เตอร์ (ไอซีเบอร์ 74LS04) ก่อน ซึ่งสัญญาณที่ได้จากอินเวอร์เตอร์ และไอซีเบอร์ 74LS11 ถูกต่อกับเกทแอนด์ (AND Gate) ซึ่งมี อินพุท 3 อินพุท (ไอซีเบอร์ 74LS11 ตัว 2) เพื่อสร้างสัญญาณแอคทีฟ LOW ไปอินเทอร์รัพท์ซีพียู ดังรูป 6.14

ฟังก์ชันการทำงานของแต่ละคีย์มีดังนี้

1. 0-9 เป็นคีย์แสดงตัวเลข 0-9

2. Enter เป็นคีย์ที่ User ใช้เมื่อสิ้นสุดการป้อนข้อมูลแล้ว

3. Page Up เป็นคีย์ที่ใช้ในการเลื่อนข้อมูลที่อยู่ก่อนข้อมูลปัจจุบัน 1 หน้ามาแสดงที่หน้าจอ

4. Page Down เป็นคีย์ที่ใช้ในการเลื่อนข้อมูลที่อยู่ถัดไปจากข้อมูลปัจจุบัน 1 หน้ามาแสดงที่หน้าจอ

5. Dial Phone เป็นคีย์ที่ใช้เมื่อต้องการใส่หมายเลขโทรศัพท์ที่ต้องการติดต่อ และจะทำการหมุนโทรศัพท์อัตโนมัติ

6. Send Data เป็นคีย์ที่รับข้อมูลจากคีย์บอร์ด และส่งข้อมูลดังกล่าวผ่านทางพอร์ทอนุกรม

7. Information เป็นคีย์ที่ใช้แสดงข้อมูลทั่ว ๆ ไป เช่น
   - เวลาทั้งหมดที่ใช้ในการติดต่อสื่อสาร
   - คิดค่าบริการที่เสีย
   - อื่น ๆ

8. Print เป็นคีย์ที่ใช้ในการสั่งให้พิมพ์ข้อมูลที่รับเข้ามาจากพอร์ทอนุกรม

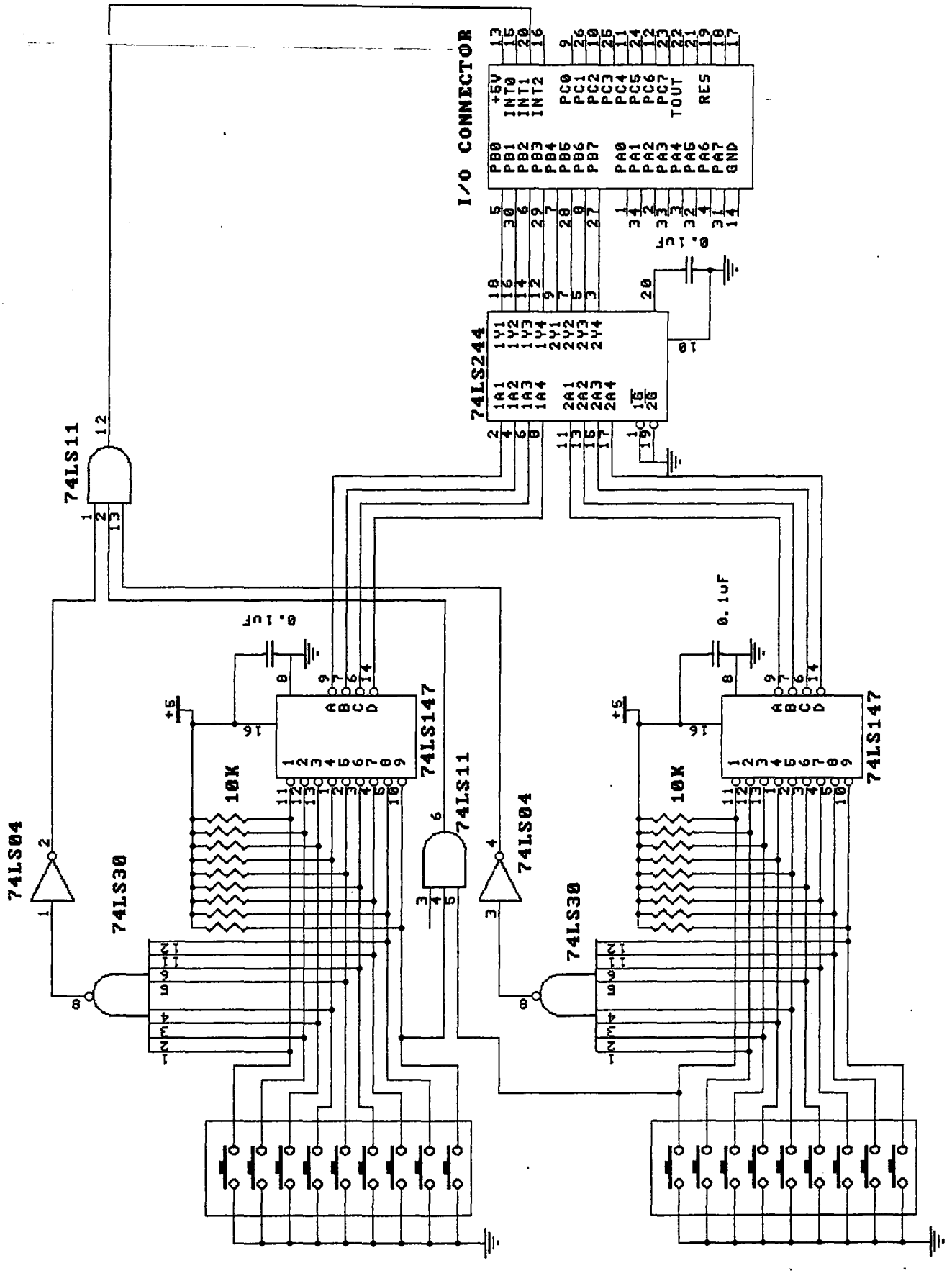
9. Break เป็นคีย์ที่ทำหน้าที่ 2 อย่างคือ Break และ Delete Data

   กรณีการ Break แบ่งย่อยเป็น

   1. Break การพิมพ์ข้อมูล เมื่อกดคีย์นี้จะทำให้ไม่มีการพิมพ์ข้อมูลส่วนที่เหลือ

   2. Break การติดต่อหรือเป็นการตัดการสื่อสาร (วางหูโทรศัพท์)

   กรณี Delete Data จะทำหน้าที่ลบข้อมูลที่เราใส่เข้าไป เช่น กรณีใส่หมายเลขโทรศัพท์ผิด หรือ กรณีพิมพ์ข้อมูลที่จะส่งไปผิด



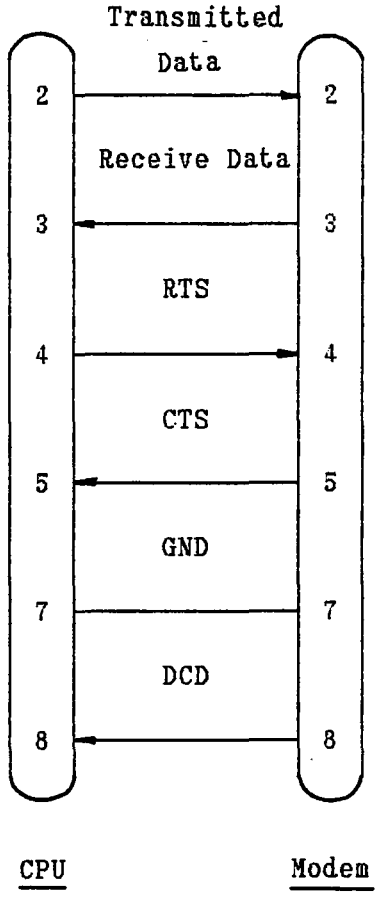
รูปที่ 6.14 แสดงวงจรส่วนคีย์บอร์ด

หลักการทำงานของโมเด็ม

ในโครงงานนี้ใช้สมาร์ทโมเด็ม เชื่อมต่อการสื่อสารผ่านทางโทรศัพท์ ซึ่งใช้การเขียนโปรแกรมควบคุมการทำงาน โดยคำสั่งที่ใช้ควบคุมการทำงานคือชุดคำสั่ง AT Command โดยมีมาตรฐานในการเชื่อมต่อระหว่างซีพียูกับสมาร์ทโมเด็ม คือ มาตรฐาน RS-232C โดยในส่วนของโมเด็มแบ่งเป็น 2 ส่วนคือ

1. ส่วนฐานข้อมูล ซึ่งจัดทำขึ้นเพื่อทดสอบระบบของโครงงาน โดยใช้ไมโครคอมพิวเตอร์เป็นที่เก็บฐานข้อมูล

2. ส่วนตัวรับ เป็นตัวรับข้อมูล ซึ่งถือได้ว่าเป็นส่วนหลักของโครงงานนี้ และในการเชื่อมต่อระหว่างซีพียูกับโมเด็มเป็นดังรูป 6.15



รูปที่ 6.15 แสดงการเชื่อมต่อของคอนเน็กเตอร์ระหว่างซีพียูกับโมเด็ม

ในการเขียนโปรแกรมควบคุมตัวโมเด็มจะอาศัย AT Command ดังที่กล่าวมา โดย AT Command ที่ส่งไปจะส่งในรูปรหัสแอสกี เมื่อโมเด็มรับคำสั่งเข้าไป ก็จะทำงานต่าง ๆ ดังที่ต้องการ

หลักการทำงานของเครื่องพิมพ์ (Printer)

การเชื่อมต่อ Printer

Printer นับว่าเป็นสิ่งสำคัญที่ใช้เป็นตัวพิมพ์ข่าวสาร หรือข้อมูลที่เราต้องการเก็บไว้เป็นหลักฐาน

ในการเชื่อมต่อ Printer สามารถทำได้ 2 รูปแบบคือ

1.แบบอนุกรม (Serial)

รูปแบบของข้อมูลจะมีลักษณะดังรูปที่ 6.16

Stop bit	bit 7	bit 6	bit 5	bit 4	bit 3	bit 2	bit 1	bit 0	Start bit

รูปที่ 6.16 แสดงรูปแบบของข้อมูลแบบอนุกรม

ข้อมูลที่จะพิมพ์จะถูกส่งออกไปทีละบิตโดยจะมีบิต start นำหน้าและมีบิต Stop เป็นบิทสิ้นสุด โดยการสื่อสารจะกระทำผ่านพอร์ทอนุกรม

2.แบบขนาน (Parallel)

ข้อมูลที่จะพิมพ์จะถูกส่งออกไปพร้อมกันทีละไบท์ โดยจะมีสัญญาณสโตรป (STROB) เป็นตัวบอกเครื่องพิมพ์ให้พิมพ์ข้อมูลตัวนั้น

โครงสร้างภายใน Printer

ภายใน Printer ประกอบไปด้วยซีพียู รอม แรม และพอร์ท เครื่องจะทำงานโดยโปรแกรมที่อยู่ภายในรอม โดยรอมจะทำหน้าที่เก็บชุดคำสั่งควบคุม และ Character Pattern ตามมาตรฐาน ส่วนแรมจะทำหน้าที่ 2 อย่างคือ ใช้เป็นบัฟเฟอร์เก็บข้อมูลที่จะพิมพ์ ซึ่งส่งมาจากไมโครคอมพิวเตอร์ และใช้เก็บรูปแบบของตัวอักษรชั่วคราว ซึ่งรูปแบบนี้อาจได้มาจากรอมหรือการเขียนขึ้นมาใหม่

สัญญาณต่าง ๆ ใน Printer

$\overline{\text{STROBE}}$ เป็นสัญญาณอินพุทส่งมาจากอุปกรณ์ภายนอก เช่น ไมโครคอมพิวเตอร์ เพื่อบอกให้เครื่องพิมพ์รับข้อมูลไป โดยทั่วไปจะมีความกว้างของพัลส์ประมาณ 0.5 ถึง 1 μs

$\overline{\text{ACKNLG}}$ เป็นสัญญาณเอาท์พุท เพื่อบอกอุปกรณ์ภายนอกให้ทราบว่าขณะนี้ Printer รับข้อมูลไว้เรียบร้อยแล้ว โดยทั่วไปจะมีความกว้างพัลส์ประมาณ 5-12 μs

BUSY เป็นสัญญาณเอาท์พุท เพื่อบอกให้อุปกรณ์ภายนอกว่า Printer พร้อมที่จะรับข้อมูลหรือไม่ โดยถ้าเป็น "1" จะหมายถึง Printer ไม่พร้อมที่จะรับข้อมูล โดยทั่วไปจะมีความกว้างพัลส์ประมาณ 0.5 μs

PE เป็นสัญญาณบอกว่ากระดาษพิมพ์หมด

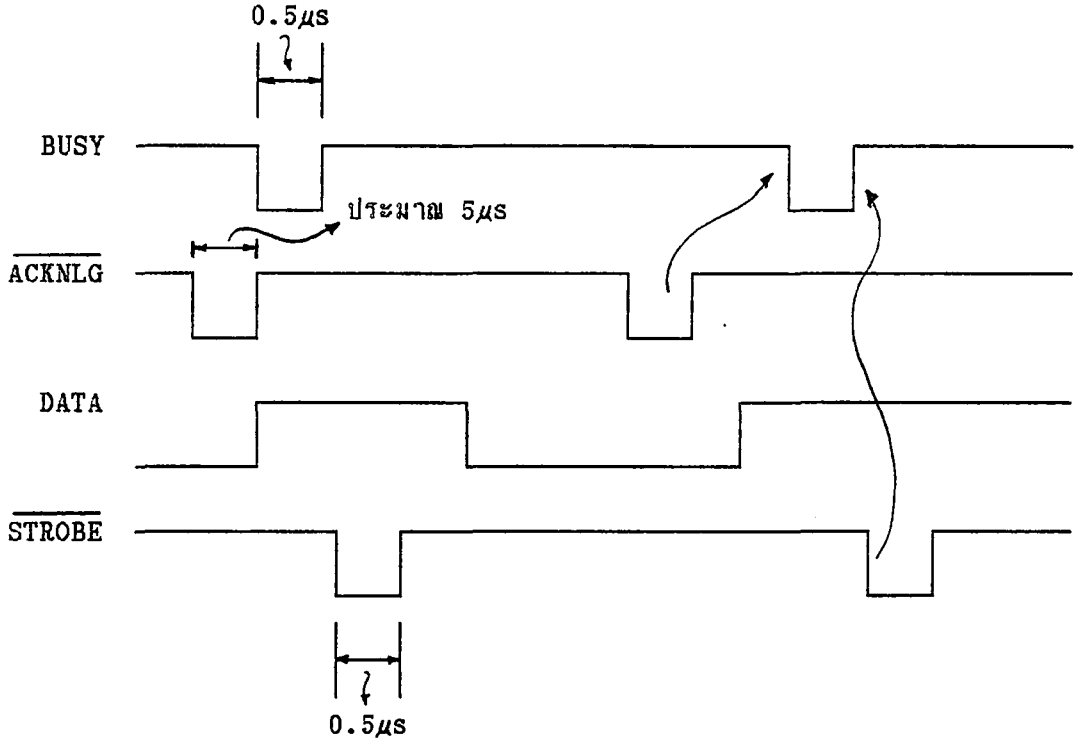
Select เป็นสัญญาณเอาท์พุท บอกสถานะของ Printer ว่าใช้งานอยู่หรือไม่ได้ใช้งาน ถ้าเป็น "1" จะหมายถึง Printer อยู่ในสถานะการใช้งาน

$\overline{\text{AUTO FEEDXT}}$ เป็นสัญญาณอินพุท เมื่อ Printer ได้รับสัญญาณนี้จะเลื่อนบรรทัดใหม่ 1 บรรทัด หลังจากพิมพ์เสร็จ

$\overline{\text{INIT}}$ เป็นสัญญาณอินพุท เมื่อ Printer ได้รับสัญญาณนี้ Printer จะเลื่อนหัวพิมพ์สู่จุดเริ่มต้นพิมพ์ ข้อมูลต่าง ๆ จะถูกละทิ้งหมด โดยทั่วไปจะมีความกว้างพัลส์ประมาณ 50 μs

$\overline{\text{ERROR}}$ เป็นสัญญาณเอาท์พุท เพื่อบอกอุปกรณ์ภายนอกว่ามีข้อผิดพลาดเกิดขึ้น

สัญญาณของการส่งข้อมูลไปยัง Printer ดังรูปที่ 6.17

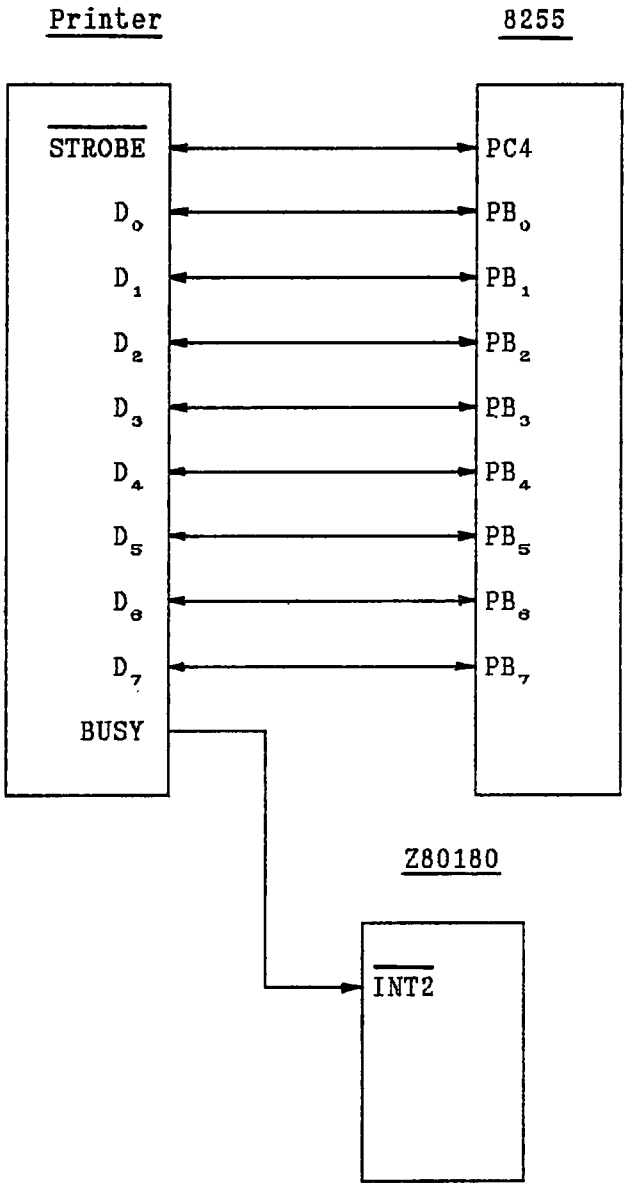


รูปที่ 6.17 แสดงสัญญาณต่าง ๆ ของการส่งข้อมูลไปยัง Printer

เมื่อ Printer พร้อมที่จะรับข้อมูล ก็จะทำการส่งสัญญาณ $\overline{\text{ACKNLG}}$ ออกไปเป็น "0" เมื่อพัลส์ของ $\overline{\text{ACKNLG}}$ ตกลงเป็น "0" นานประมาณ 5μs ที่ขอบขาขึ้นของ $\overline{\text{ACKNLG}}$ ขาสัญญาณ BUSY จาก Printer ก็จะเป็น "0" นานประมาณ 0.5 μs เพื่อบอกกับอุปกรณ์ภายนอกที่จะส่งข้อมูลให้ Printer ว่าตอนนี้ให้ส่งข้อมูลมาได้แล้ว พอหมดพัลส์ของ BUSY อุปกรณ์ที่ส่งข้อมูลมายัง

Printer ต้องส่งสัญญาณ $\overline{\text{STROBE}}$ มาให้ Printer เพื่อเป็นการบอกให้ Printer อ่านข้อมูลที่ส่งมาให้นั้น โดยสัญญาณ $\overline{\text{STROBE}}$ ต้องมีความกว้างพัลส์อย่างน้อย 0.5μs จากรูปจะพบว่า ขณะที่สัญญาณ $\overline{\text{STROBE}}$ เป็น "0" สัญญาณ BUSY จะเป็น "1" ซึ่งหมายถึง ตอนนี้ข้อมูลตัวต่อไปไม่สามารถส่งมาได้เพราะ Printer กำลังกระทำเกี่ยวกับข้อมูลอยู่

การต่อเครื่องพิมพ์จะต่อผ่านพอร์ท B ของ 8255 ซึ่งจะมีการต่อดังรูปที่ 6.18

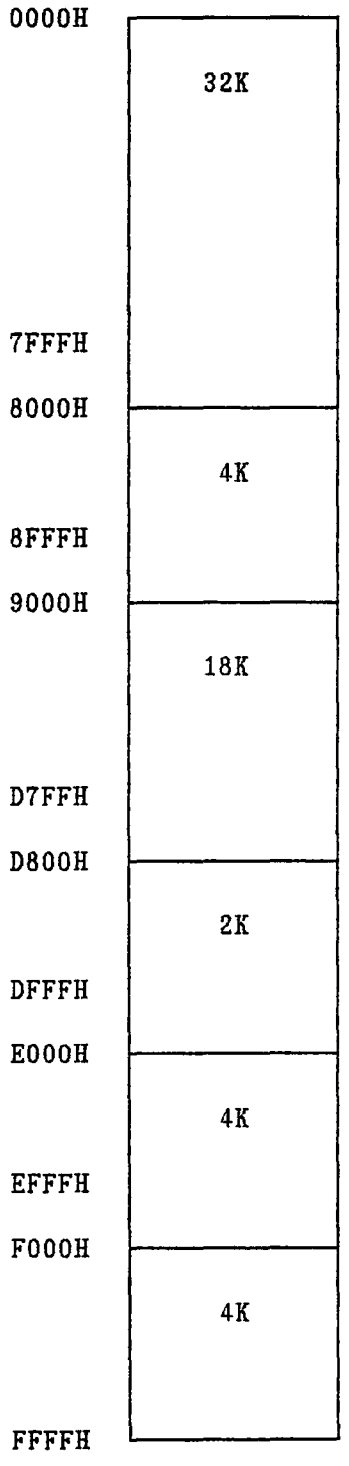


รูปที่ 6.18 แสดงการต่อพอร์ทเครื่องพิมพ์เข้ากับพอร์ทของ 8255

การทำงานเมื่อซีพียูต้องการพิมพ์ข้อมูล ซีพียูจะส่งข้อมูลผ่านพอร์ท B ของ 8255 ออกไปยังเครื่องพิมพ์ พร้อมกับส่งสัญญาณ $\overline{\text{STROBE}}$ ออกไปทางพอร์ท C ขาที่ 4 เพื่อบอกให้เครื่องพิมพ์รับข้อมูลเข้าไป เมื่อเครื่องพิมพ์รับข้อมูลเรียบร้อยแล้ว เครื่องพิมพ์จะส่งสัญญาณอินเทอร์รัพท์มาบอกซีพียู โดยส่งผ่าน $\overline{\text{INT2}}$ เพื่อให้ซีพียูส่งข้อมูลชุดต่อไปออกมา

แผนผังการจัดหน่วยความจำ

หน่วยความจำที่ใช้ในโครงงานมีลักษณะดังรูปที่ 6.19



รูปที่ 6.19 แสดงการจัดแอดเดรสของหน่วยความจำที่ใช้ในโครงงาน

รายละเอียดการจัดแอดเดรสของหน่วยความจำเป็นดังนี้

แอดเดรส 0000H-7FFFH ขนาด 32 กิโลไบท์ เป็นส่วนโปรแกรมมอนิเตอร์ ที่ควบคุมการทำงานของระบบทั้งหมด ซึ่งประกอบไปด้วยตัวโปรแกรม และค่าคงที่ต่าง ๆ

แอดเดรส 8000H-8FFFH ขนาด 4 กิโลไบท์ เป็นพื้นที่ของ RAM ที่ใช้เก็บข้อมูลขณะที่โปรแกรมมอนิเตอร์มีการเขียน หรือการอ่านข้อมูล

แอดเดรส 9000H-D7FFH ขนาด 18 กิโลไบท์ เป็นพื้นที่บัฟเฟอร์ใช้ในการเก็บข้อมูลที่รับมาจากการสื่อสาร ซึ่งสามารถเก็บข้อมูลได้ทั้งหมดประมาณ 18 หน้าของแฟ้มข้อมูล

แอดเดรส D800H-DFFFH ขนาด 2 กิโลไบท์ เป็นส่วน VIDEO RAM ใช้เก็บข้อมูลที่ต้องการแสดงผล

แอดเดรส E000H-EFFFH ขนาด 4 กิโลไบท์ เป็นพื้นที่ว่าง

แอดเดรส F000H-FFFFH ขนาด 4 กิโลไบท์ เป็นพื้นที่ของสแต็ก (Stack Area) ใช้เก็บข้อมูลขณะที่ซีพียูทำคำสั่ง PUSH CALL หรือการตอบสนองการอินเทอร์รัพท์

ตารางที่ 6.4 แสดงแผนผังการจัดแอดเดรสของพอร์ทอินพุท/เอาท์พุท

พอร์ทอินพุท/เอาท์พุท	แอดเดรส
ASCI Control Register A Cho	00H
ASCI Control Register B Cho	02H
ASCI Status Register Ch0	04H
ASCI Transmit Data Register Ch0	06H
ASCI Receive Data Register Ch0	08H
Timer Data Register Ch 0L	0CH
Timer Data Register Ch 0H	0DH
Reload Register Ch 0L	0EH
Reload Register Ch 0H	0FH
Timer Control Register	10H
Interrupt Vector Low Register	33H
INT/TRAP Control Register	34H

ตารางที่ 6.4 แสดงแผนผังการจัดแอดเดรสของพอร์ทอินพุท/เอาท์พุท (ต่อ)

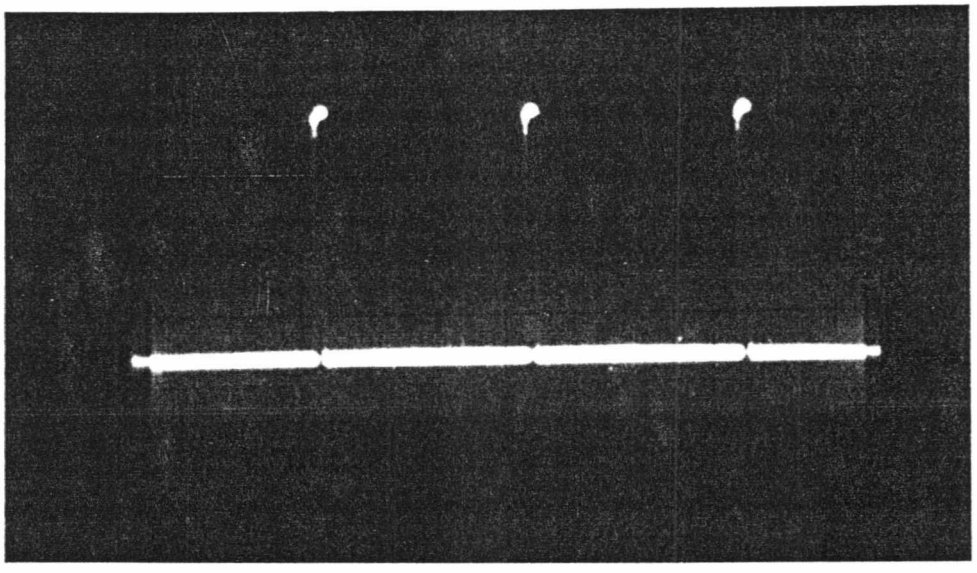
พอร์ทอินพุท/เอาท์พุท	แอดเดรส
MMU Common Base Register	38H
MMU Bank Base Register	39H
MMU Common/Bank Area Register	3AH
Operation Mode Control Register	3EH
PortA 8255_1	8CH
PortB 8255_1	8DH
PortC 8255_1	8EH
Port Control 8255_1	8FH
PortA 8255_2	9CH
PortB 8255_2	9DH
PortC 8255_2	9EH
Port Control 8255_2	9FH
Port Address 6845CRT	0D0H
Port Data 6845CRT	0D1H

บทที่ 7

ผลการทดลอง

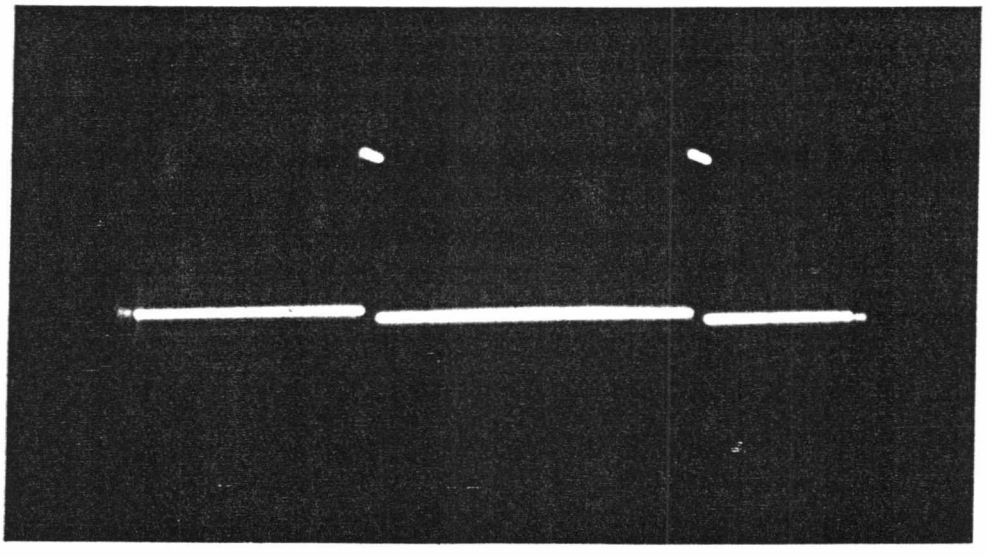
สัญญาณต่าง ๆ ที่ใช้ในการแสดงจอภาพ

1. สัญญาณ Horizontal Synchronous (HSYNC) เป็นสัญญาณรูปคลื่นสี่เหลี่ยม มีความถี่เท่ากับ 18.3 kHz มี V_{p-p} = 4 Volt ลักษณะของคลื่นที่ได้ดังรูปที่ 7.1



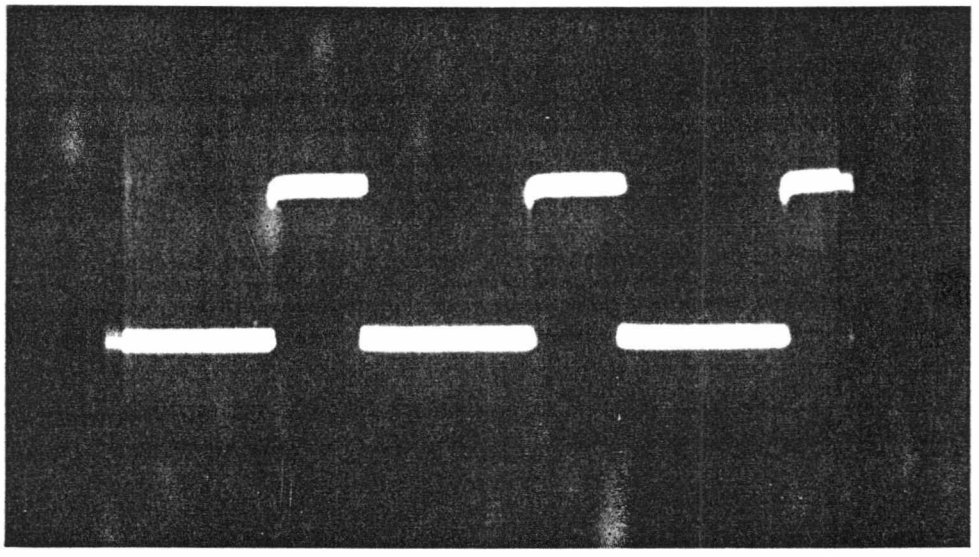
รูปที่ 7.1 แสดงรูปสัญญาณ Horizontal Synchronous (HSYNC)

2. สัญญาณ Vertical Synchronous (VSYNC) เป็นสัญญาณรูปคลื่นสี่เหลี่ยมมีความถี่เท่ากับ 53 Hz มี V_{p-p} = 4 Volt ลักษณะของคลื่นที่ได้เป็นดังรูปที่ 7.2



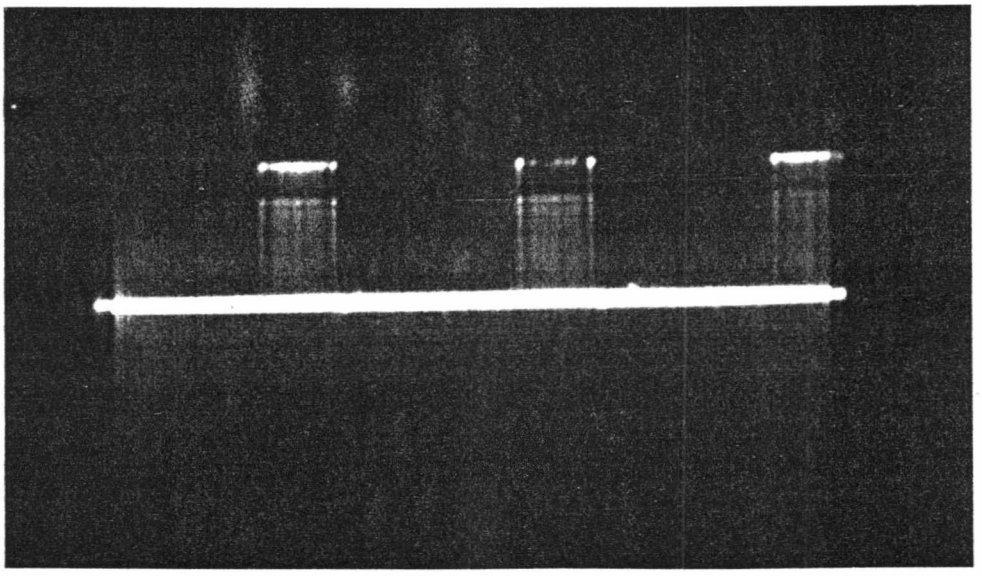
รูปที่ 7.2 แสดงรูปสัญญาณ Vertical Synchronous (VSYNC)

3. สัญญาณ Display Enable (DISPEN) เป็นสัญญาณรูปคลื่นสี่เหลี่ยม มีความถี่เท่ากับ 14.6 Hz มี V_{P-P} = 4 Volt ลักษณะของคลื่นที่ได้ แสดงดังรูปที่ 7.3



รูปที่ 7.3 แสดงรูปสัญญาณ Display Enable (DISPEN)

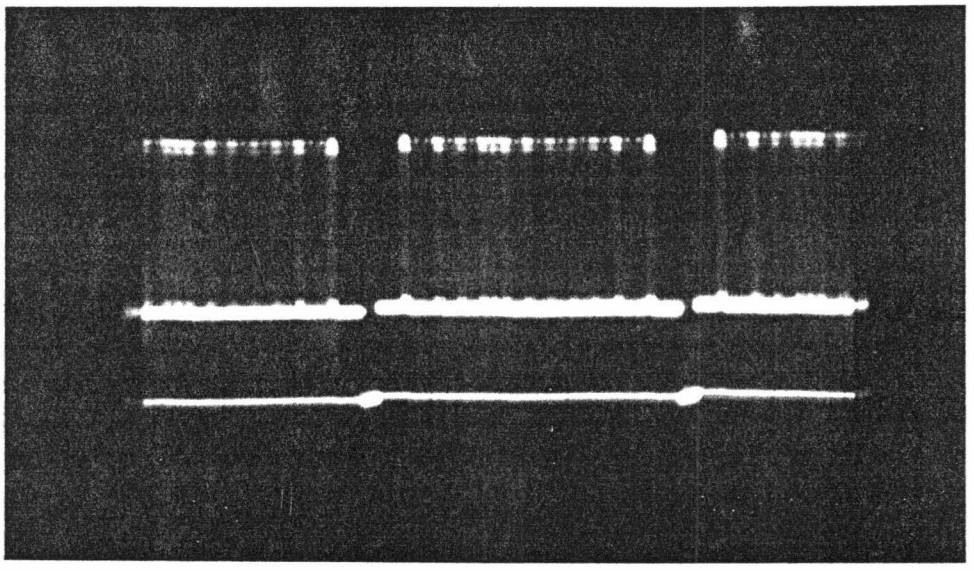
4.สัญญาณวีดีโอ เป็นสัญญาณรูปคลื่นสี่เหลี่ยม ซึ่งแทนค่าใน Dot Matrix ของรูปแบบตัวอักษร จุดใดที่มีสถานะเป็น 1 จะทำให้สัญญาณวีดีโอเป็น High จุดใดมีสถานะเป็น 0 จะทำให้สัญญาณวีดีโอเป็น Low ซึ่งสัญญาณวีดีโอมี V_{P-P} = 3 Volt ลักษณะของคลื่นที่ได้ แสดงได้ดังรูปที่ 7.4



รูปที่ 7.4 แสดงรูปสัญญาณวีดีโอ

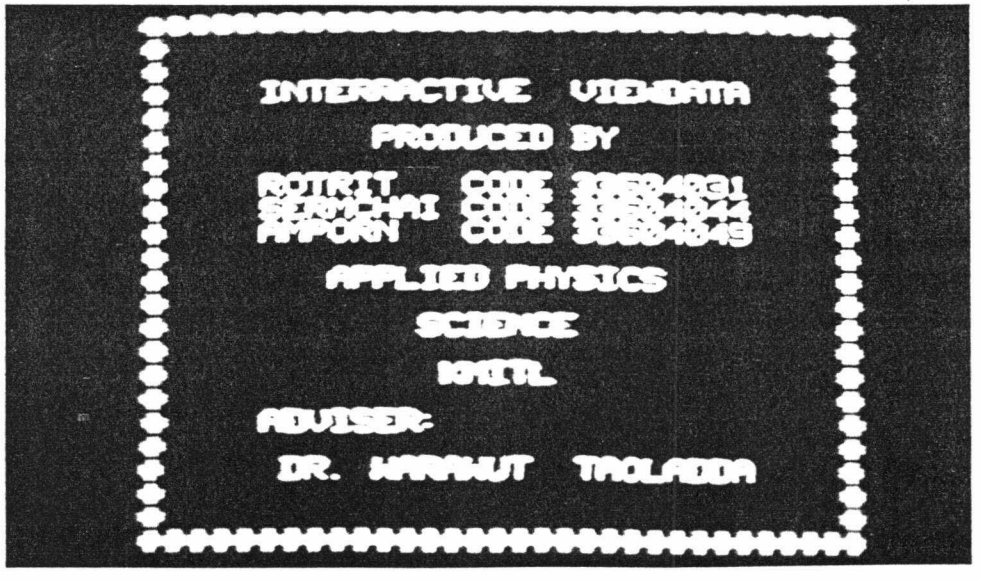
5. สัญญาณคอมโพสิตวีดีโอ (Composite Video) เป็นสัญญาณซึ่งเกิดจากการรวม

สัญญาณ Horizontal Synchronous สัญญาณ Vertical Synchronous และ สัญญาณวีดีโอ ซึ่งสัญญาณนี้มี V_{p-p} = 2.6 Volt ลักษณะของคลื่นที่ได้ แสดงดังรูปที่ 7.5



รูปที่ 7.5 แสดงรูปสัญญาณ Composite Video

6. สัญญาณพาหะ (Carrier Signal) เป็นสัญญาณมีความถี่ประมาณ 80.0 MHz เพื่อนำไปมอดดูเลทกับสัญญาณคอมโพสิตวีดีโอ (Composite Video) แล้วส่งเข้าช่องสายอากาศของโทรทัศน์ดังรูปที่ 1.6.1 โดยภาพที่ปรากฏบนจอโทรทัศน์จะมีการคลาดเคลื่อนเกิดขึ้นในบรรทัดที่ 1-4 ของจอภาพ แต่ที่ส่วนอื่น ๆ เป็นปกติ



รูปที่ 7.6 การแสดงผลออกทางหน้าจอโทรทัศน์

โมเด็ม และ พอร์ทอนุกรม

1. เมื่อเราเขียนโปรแกรม AT Command ส่งไปให้โมเด็ม โมเด็มจะรับคำสั่งและทำตามคำสั่งที่เราส่งไปให้ เช่น

AT DT 2763511 โมเด็มจะทำการหมุนเบอร์โทรศัพท์ 276-3511 แบบโทน (TONE) ให้โดยอัตโนมัติ

2. สัญญาณต่าง ๆ ในระบบ RS 232C ซึ่งใช้ติดต่อระหว่างไมโครโปรเซสเซอร์กับโมเด็มดังนี้

- Request To Send

ขณะ On คือเมื่อมีการส่งข้อมูล จะเป็นสถานะ 0 มีแรงดันเท่ากับ 7.36 V

ขณะ Off คือเมื่อมีการส่งข้อมูล จะเป็นสถานะ 1 มีแรงดันเท่ากับ -8.36 V

- CLear To Send

ขณะ On คือเมื่อมีการส่งข้อมูล จะเป็นสถานะ 0 มีแรงดันเท่ากับ 7.36 V

ขณะ Off คือเมื่อมีการส่งข้อมูล จะเป็นสถานะ 1 มีแรงดันเท่ากับ -7.9 V

- Data Carrier Detect

ขณะ On จะเป็นสถานะ 0 มีแรงดันเท่ากับ 7.29 V

ขณะ Off จะเป็นสถานะ 1 มีแรงดันเท่ากับ -7.18 V

- Transmit Data

ขณะข้อมูลที่ส่งมีสถานะเป็น 0 มีแรงดันเท่ากับ 8.75 V

ขณะข้อมูลที่ส่งมีสถานะเป็น 1 มีแรงดันเท่ากับ -9.25 V

- Receive Data

ขณะรับข้อมูล ถ้าข้อมูลที่รับเป็นสถานะ 1 มีแรงดันเท่ากับ 8.7 V

ขณะรับข้อมูล ถ้าข้อมูลที่รับเป็นสถานะ 0 มีแรงดันเท่ากับ -7.99 V

3. ข้อมูลที่ทำการรับส่ง ในบางครั้งจะมีข้อผิดพลาดเกิดขึ้น คือ มีข้อมูลหรือตัวอักษรผิดเพี้ยนปนเข้ามา แต่ในบางครั้งก็ไม่มี

4. ในการรับส่งข้อมูลที่ความเร็วสูงปรากฏว่า จะมีการเลื่อน (Shift) ของข้อมูลบนจอแสดงผล แต่ในกรณีความเร็วต่ำจะไม่มีการเลื่อนของข้อมูล หรือเลื่อนเพียงเล็กน้อยเท่านั้น

คีย์บอร์ด และ เครื่องพิมพ์

การทำงานของคีย์บอร์ด เมื่อมีการกดคีย์ค้างนานเกินไป จะทำให้มีการอินเทอร์รัพท์ซ้อนเกิดขึ้น ซึ่งจะทำให้ซีพียูทำงานซ้ำอย่างเดิม แต่ถ้าหากกดคีย์โดยไม่มีการกดค้าง การทำงานของซีพียู ก็จะเป็นไปโดยปกติ ไม่มีข้อผิดพลาดเกิดขึ้น

การทำงานของเครื่องพิมพ์ เมื่อมีการสั่งพิมพ์ เครื่องพิมพ์จะเริ่มทำการพิมพ์ข้อมูลที่รับมาโดยอัตโนมัติ และจะหยุดพิมพ์ทันทีที่จำนวนข้อมูลที่ต้องการพิมพ์ครบ หรือจะหยุดพิมพ์ทันทีที่มีการสั่งหยุดการพิมพ์ โดยใช้คีย์ Break ผ่านทางคีย์บอร์ด

การทำงานของทั้งระบบ

เมื่อนำระบบต่าง ๆ มาเชื่อมต่อกัน และเริ่มทำงาน ระบบสามารถทำงานสัมพันธ์กันได้ แต่จะมีข้อผิดพลาดในส่วนการแสดงผลบรรทัดที่ 1-4 และ การชนของข้อมูลเมื่อความเร็วในการสื่อสารเพิ่มขึ้น

บทที่ 8

สรุปผลวิจารณ์และข้อเสนอแนะ

เมื่อนำส่วนต่าง ๆ ซึ่งประกอบด้วย ไมโครโปรเซสเซอร์ ส่วนแสดงผล โมเด็ม และคีย์บอร์ดมาต่อร่วมกัน ระบบสามารถทำงานสัมพันธ์กันได้ แต่ก็ยังมีปัญหาเกิดขึ้น 4 จุดระหว่างการทำงานคือ

1. ขณะที่แสดงผลบนจอจะมีการเลื่อน (Shift) เกิดขึ้นที่บรรทัด 1-4 ของจอแสดงผล โดยอาจมีสาเหตุมาจาก

1.1 สัญญาณ Horizontal Synchronous ของ 6845 CRTC คลาดเคลื่อนไปจากความถี่ Horizontal Synchronous ของจอโทรทัศน์

1.2 อาจมีสัญญาณรบกวนวงจร LC

1.3 ค่า L ที่ได้จากการพันคลาดเคลื่อนจากค่าจริงที่ต้องการใช้ในวงจร

ข้อเสนอแนะ

- ควรมีส่วนป้องกันสัญญาณรบกวนโดยการทำกราวด์เพลน (Ground Plane) ให้มีขนาดใหญ่ขึ้น และสร้างส่วนกำบัง (Shield) ด้วยอลูมิเนียม

- สร้างวงจรปรับความถี่ Horizontal Synchronous และ ความถี่ Vertical Synchronous ของวงจรเพื่อให้ได้ค่าเท่ากับความถี่ Horizontal Synchronous และความถี่ Vertical Synchronous ของโทรทัศน์

2. ขณะที่รับส่งข้อมูลด้วยความเร็วสูงจะมีการเลื่อนของข้อมูลที่เก็บในวีดีโอแรม แต่กรณีที่ส่งด้วยความเร็วต่ำการเลื่อนของข้อมูลจะน้อยลง โดยที่มีสาเหตุมาจากระบบแสดงผล ระบบสื่อสาร และตัวไมโครโปรเซสเซอร์มีความเร็วไม่สัมพันธ์กัน

ข้อเสนอแนะ

- ทำฮาร์ดแวร์ส่วนที่ใช้ในการสื่อสาร (โมเด็ม) ขึ้นมาแล้วใช้ไมโครโปรเซสเซอร์ควบคุมโดยตรงโดยไม่ใช้สมาร์ทโมเด็ม (Smart Modem) เพราะในตัวสมาร์ทโมเด็มจะมีไมโครโปรเซสเซอร์ควบคุมภายในอยู่แล้ว

- ใช้วิธี Handshaking เข้ามาแก้ไขโดยให้มีการตรวจสอบความพร้อมของอุปกรณ์อินพุทและเอาท์พุท และไมโครโปรเซสเซอร์

3. ในส่วนของโปรแกรมที่เป็นฐานข้อมูล ซึ่งจัดทำขึ้นเพื่อใช้ในการทดสอบระบบ ในตัวโปรแกรมจะมีปัญหาในเรื่องการรับข้อมูล คือ ในบางครั้งข้อมูลที่รับเข้าไปมีข้อผิดพลาดเกิดขึ้น ทำให้ผู้ส่งต้องมีการส่งข้อมูลชุดเดิมซ้ำ ซึ่งปัญหาที่เกิดขึ้นอาจมีสาเหตุมาจากสายส่ง หรือ สัญญาณรบกวน

ข้อเสนอแนะ

- เขียนโปรแกรมตรวจสอบข้อผิดพลาด เช่น ตรวจสอบพาริตี้ (Parity Check), ตรวจสอบผลรวม (Checksum) ของข้อมูลที่รับเข้ามา ว่าถูกต้องหรือไม่ ถ้าข้อมูลไม่ถูกต้องให้ส่งสัญญาณกลับไปบอกตัวส่งให้ส่งข้อมูลชุดเดิมซ้ำ

- ปรับความเร็วในการสื่อสารให้น้อยลง

4. ในส่วนคีย์บอร์ด จะมีการอินเทอร์รัพท์ซ้อนเกิดขึ้นในบางครั้ง ซึ่งมีสาเหตุมาจากการกดปุ่มค้างนานเกินไป

ข้อเสนอแนะ

- ใช้โปรแกรมหน่วงเวลา เมื่อมีการอินเทอร์รัพท์เกิดขึ้น

การพัฒนาในอนาคต

1. ทำฮาร์ดแวร์ในส่วนของการสื่อสาร (โมเด็ม) เข้าไปรวมไว้กับส่วนแสดงผลและคีย์บอร์ด

2. ปรับปรุงฮาร์ดแวร์ส่วนแสดงผล เพื่อให้แสดงกราฟฟิค (Graphic) ได้

3. ปรับปรุงฮาร์ดแวร์ส่วนแสดงผล เพื่อให้แสดงผลภาษาไทยได้

4. ทำการขยายหน่วยความจำ เพื่อรับข้อมูลได้มากขึ้น

5. ทำการขยายส่วนที่จะเก็บข้อมูลอย่างถาวรได้

6. สร้างวงจรเพื่อรับส่งข้อมูลได้ทั้งภาพและเสียง

ภาคผนวก ก

National Semiconductor

PRELIMINARY

LM1889 TV Video Modulator

General Description

The LM1889 is designed to interface audio, color difference, and luminance signals to the antenna terminals of a TV receiver. It consists of a sound subcarrier oscillator, chroma subcarrier oscillator, quadrature chroma modulators, and RF oscillators and modulators for two low-VHF channels.

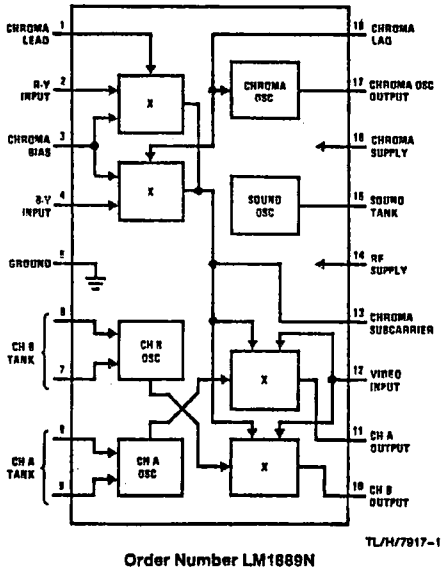
The LM1889 allows video information from VTR's, games, test equipment, or similar sources to be displayed on black and white or color TV receivers. When used with the MM57100 and MM53104, a complete TV game is formed.

Features

- dc channel switching
- 12V to 18V supply operation
- Excellent oscillator stability
- Low intermodulation products
- 5 Vp-p chroma reference signal
- May be used to encode composite video

Block Diagram

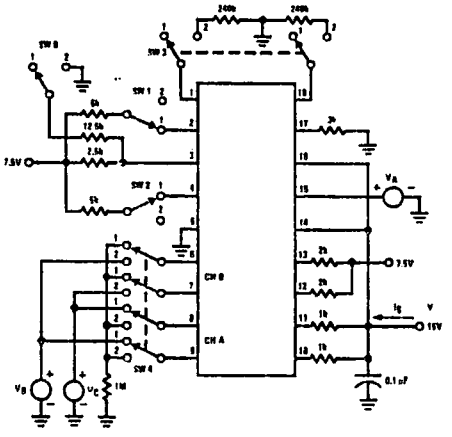
Dual-In-Line Package



TL/H/7917–1

Order Number LM1889N
See NS Package Number N18A

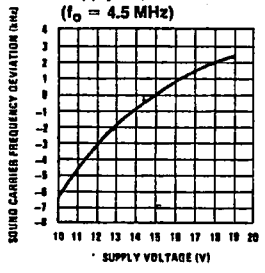
DC Test Circuit



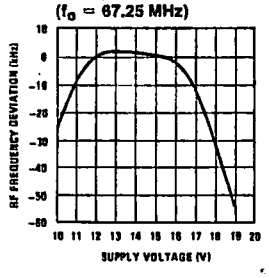
TL/H/7917–2

Typical Performance Characteristics

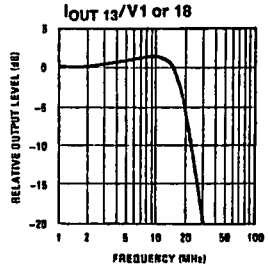
Sound Carrier Oscillator Supply Dependence (f_o = 4.5 MHz)



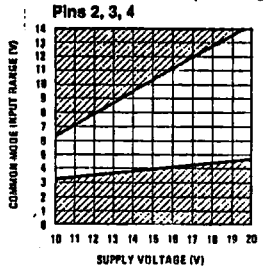
RF Oscillator Frequency Supply Dependence (f_o = 67.25 MHz)



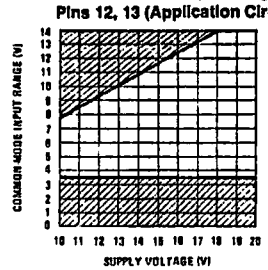
Chroma Modulator Transconductance Bandwidth $I_{OUT\ 13}/V1$ or 18



Chroma Modulator Common-Mode Input Range Pins 2, 3, 4



RF Modulator Common-Mode Input Range Pins 12, 13 (Application Circuit)



TL/H/7917–4

Circuit Description (Refer to Circuit Diagram)

The sound carrier oscillator is formed by differential amplifier Q3, Q4 operated with positive feedback from the pin 15 tank to the base of Q4.

The chroma oscillator consists of the inverting amplifier Q16, Q17 and Darlington emitter follower Q11, Q12. An external RC and crystal network from pin 17 to pin 18 provides an additional 180 degrees phase lag back to the base of Q17 to produce oscillation at the crystal resonance frequency. (See AC test circuit).

The feedback signal from the crystal is split in a lead-lag network to pins 1 and 18, respectively, to generate the subcarrier reference signals for the chroma modulators. The R-Y modulator consists of multiplier devices Q29, Q30 and Q21–Q24, while the B-Y modulator consists of Q31, Q32 and Q25–Q28. The multiplier outputs are coupled through a balanced summing amplifier Q37, Q38 to the input of the RF modulators at pin 13. With 0 offset at the lower pairs of the multipliers, no chroma output is produced. However, when either pin 2 or pin 4 is offset relative to pin 3 a subcarrier output current of the appropriate phase is produced at pin 13.

The channel B oscillator consists of devices Q56 and Q57 cross-coupled through level-shift zener diodes Q54 and Q55. A current regulator consisting of devices Q39–Q43 is used to achieve good RF frequency stability over supply and temperature. The channel B modulator consists of multiplier devices Q58, Q59 and Q50–Q53. The top quad is coupled to the channel B tank through isolating devices Q48 and Q49. A dc offset between pins 12 and 13 offsets the lower pair to produce an output RF carrier at pin 10. That carrier is then modulated by both the chroma signal at pin 13 and the video and sound carrier signals at pin 12. The channel A modulator shares pin 12 and 13 buffers Q45 and Q44 with channel B and operates in an identical manner.

The current flowing through channel B oscillator diodes Q54, Q55 is turned around in Q60, Q61 and Q62 to source current for the channel B RF modulator. In the same manner, the channel A oscillator Q71–Q74 uses turn around Q77, Q78 and Q79 to source the channel A modulator. One oscillator at a time may be activated by connecting its tank to supply (see ac test circuit). The corresponding modulator is then activated by its current turn-around, and the other oscillator/modulator combination remains "OFF".

Absolute Maximum Ratings

If Military/Aerospace specified devices are required, please contact the National Semiconductor Sales Office/Distributors for availability and specifications.

Supply Voltage V14, V16 max	19 V_{DC}
Power Dissipation Package (Note 1)	1800 mW
Operating Temperature Range	0°C to +70°C
Storage Temperature Range	−55°C to +150°C
Chroma Osc Current I_{17} max	10 mA_{DC}
(V16−V15) max	±5 V_{DC}
(V14−V10) max	7V
(V14−V11) max	7V
Lead Temperature (Soldering, 10 sec.)	260°C

DC Electrical Characteristics (dc Test Circuit, All SW Normally Pos. 1, V_A = 15V, V_B = V_C = 12V)

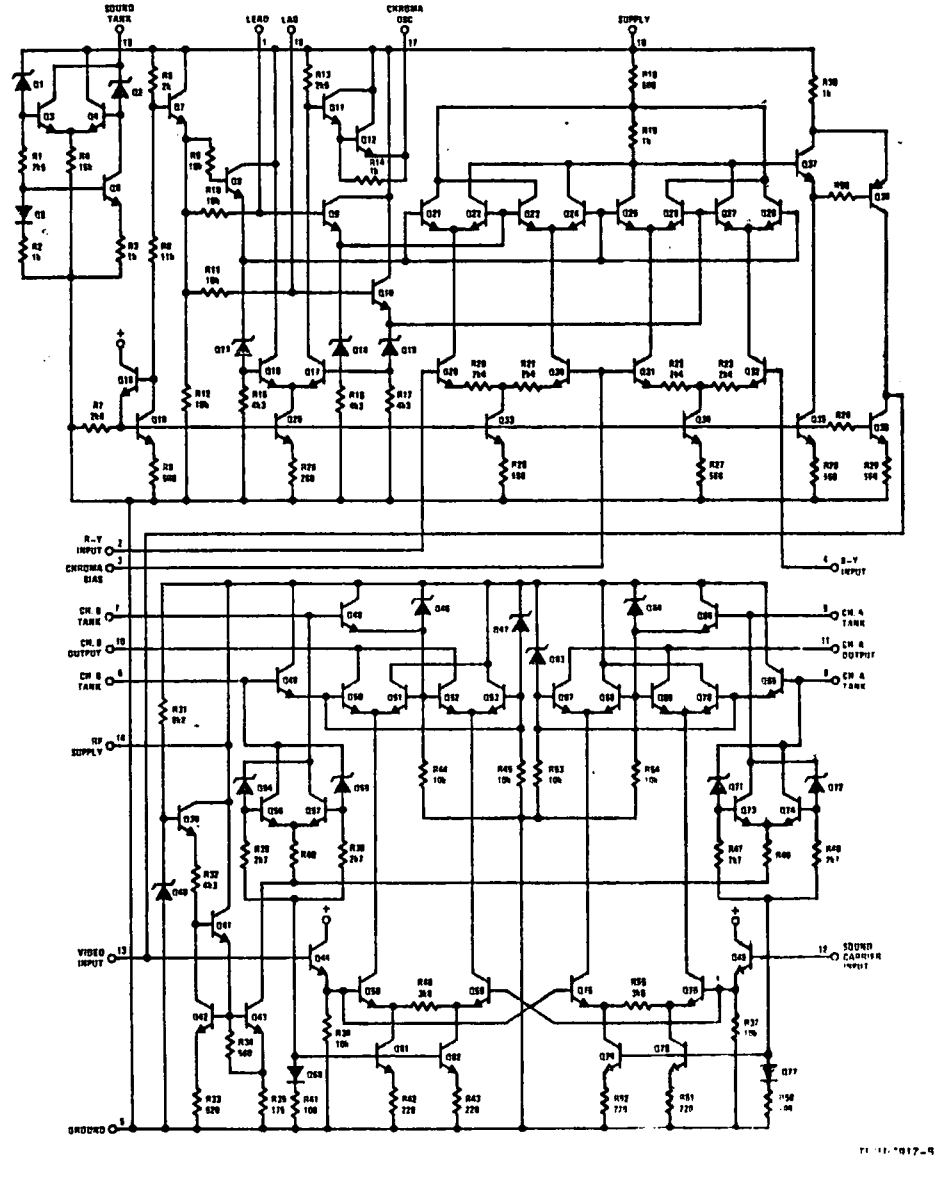
Symbol	Parameter	Conditions	Min	Typ	Max	Units
I_S	Supply Current		20	35	45	mA
ΔI_{15}	Sound Oscillator, Current Change	Change V_A from 12.5 to 17.5V	0.3	0.6	0.9	mA
V17	Chroma Oscillator Balance		9.5	11.0	12.5	V
V13	Chroma Modulator Balance		7.0	7.4	7.8	V
ΔV13	R-Y Modulator Output Level	SW 3, Pos. 2, Change SW 1 from Pos. 1 to Pos. 2	0.6	0.9	1.2	V
ΔV13	B-Y Modulator Output Level	SW 3, Pos. 2, Change SW 2 from Pos. 1 to Pos. 2	0.6	0.9	1.2	V
ΔV13/ΔV3	Chroma Modulator Conversion Ratio	SW 3, Pos. 2, Change SW 0 from Pos. 1 to Pos. 2 Divide ΔV13 by ΔV3	0.45	0.70	0.95	V/V
V8, V9	Ch. A Oscillator "OFF" Voltage	SW 4, Pos. 2		1.0	3.0	V
I_9	Ch. A Oscillator Current Level	V_B = 12V, V_C = 13V	3.0	4.0	5.5	mA
V6, V7	Ch. B Oscillator "OFF" Voltage			1.0	3.0	V
I_6	Ch. B Oscillator Current Level	SW 4, Pos. 2, V_B = 12V, V_C = 13V	3.0	4.0	5.5	mA
ΔV11/(V13−V12)	Ch. A Modulator Conversion Ratio	SW 1, SW2, SW 3, Pos. 2, Measure ΔV11(V10) by Changing from V_B = 12.5V, V_C = 11.5V to V_B = 11.5V, V_C = 12.5V and Divide by V13−V12	0.35	0.55	0.75	V/V
ΔV10/(V13−V12)	Ch. B Modulator Conversion Ratio	Divide as Above	0.35	0.55	0.75	V/V

AC Electrical Characteristics (AC Test Circuit, V = 15V)

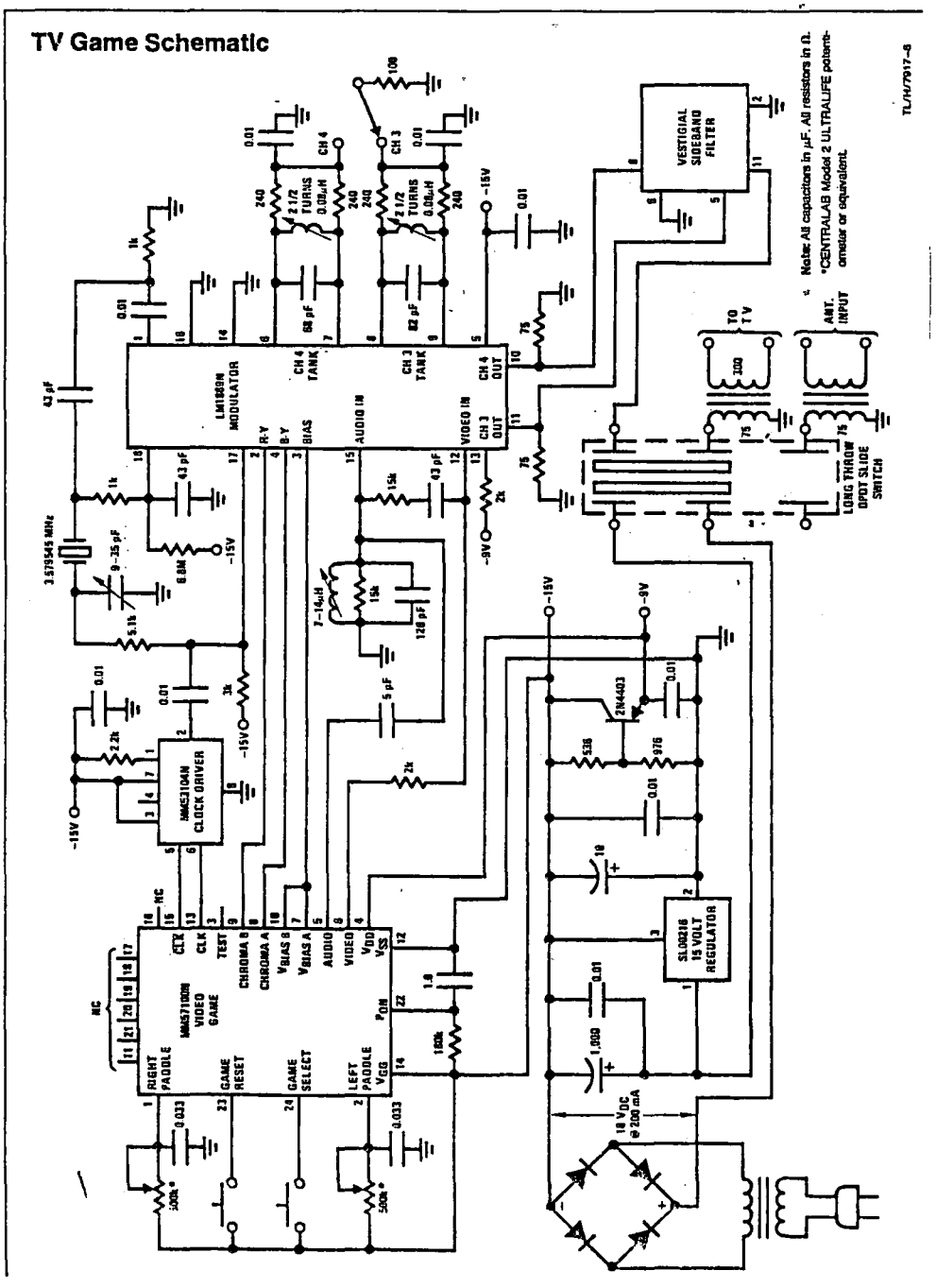
Symbol	Parameter	Conditions	Min	Typ	Max	Units
V17	Chroma Oscillator Output Level	C_{LOAD} ≤ 20 pF	4	5		Vp-p
V15	Sound Carrier Oscillator Level	Loaded by RC Coupling Network	2	3	4	Vp-p
V8, V9	Ch. 3 RF Oscillator Level	Ch. SW. Pos. 3, f = 61.25 MHz, Use FET Probe	200	350		mVp-p
V6, V7	Ch. 4 RF Oscillator Level	Ch. Sw. Pos. 4, f = 67.25 MHz Use FET Probe	200	350		mVp-p

Note 1: For operation in ambient temperatures above 25°C, the device must be derated based on a 150° maximum junction temperature and a thermal resistance of 70°C C/W junction to ambient.

Circuit Diagram



TV Game Schematic



Applications Information

Subcarrier Oscillator

The oscillator is a crystal-controlled design to ensure the accuracy and stability required of the subcarrier frequency for use with television receivers. Lag-lead networks (R2C2 and C1R1) define a quadrature phase relationship between pins 1 and 18 at the subcarrier frequency of 3.579545 MHz. Other frequencies can be used and where high stability is not a requirement, the crystal can be replaced with a parallel resonant L-C tank circuit—to provide a 2 MHz clock, for example. Note that since one of the chrominance modulators is internally connected to the feedback path of the oscillator, operation of the oscillator at other than the correct subcarrier frequency precludes chrominance modulation.

When an external subcarrier source is available or preferred, this can be used instead. For proper modulator operation, a subcarrier amplitude of 500 mVp-p is required at pins 1 and 18. If the quadrature phase shift networks shown in the application circuit are retained, about 1 Vp-p subcarrier injected at the junction of C1 and R2 is sufficient. The crystal, C4 and R3 are eliminated and pin 17 provides a 5 Vp-p signal shifted +125°C from the external reference.

Chrominance Modulation

The simplest method of chroma encoding is to define the quadrature phases provided at pins 1 and 18 as the color difference axes R-Y and B-Y. A signal at pin 2 (R-Y) will give a chrominance subcarrier output from the modulator with a relative phase of 90°C compared to the subcarrier output produced by a signal at pin 4 (B-Y). The zero signal dc level of the R-Y and B-Y inputs will determine the bias level required at pin 3. For example, a pin 2 signal that is 1V positive with respect to pin 3 will give 0.6 Vp-p subcarrier at a relative phase of 90°C. If pin 2 is 1V negative with repsect to pin 3, the output is again 0.6 Vp-p, but with a relative phase of 270°C. When a simultaneous signal exists at pin 4, the subcarrier output level and phase will be the vector sum of the quadrature components produced by pin 2 and 4 inputs. Clearly, with the modulation axes defined as above, a negative pulse on pin 4 during the burst gate period will produce the chrominance synchronizing "burst" with a phase of 180°. Both color difference signals must be dc coupled to the modulators and the zero signal dc level of both must be the same and within the common-mode range of the modulators.

The 0.6 Vp-p/V_{dc} conversion gain of the chrominance modulators is obtained with a 2 kΩ resistor connected at pin 13. Larger resistor values can be used to increase the gain, but capacitance at pin 13 will reduce the bandwidth. Notice that equi-bandwidth encoding of the color difference signals is implied as both modulator outputs are internally connected and summed into the same load resistor.

Sound Oscillator

Frequency modulation is achieved by using a 4.5 MHz tank circuit and deviating the center frequency via a capacitor or a varactor diode. Switching a 5 pF capacitor to ground at an audio frequency rate will cause a 50 kHz deviation from 4.5 MHz. A 1N5447 diode biassed −4V from pin 16 will give ±20 kHz deviation with a 1 Vp-p audio signal. The coupling network to the video modulator input and the varactor diode bias must be included when the tank circuit is tuned to center frequency.

A good level for the RF sound carrier is between 2% and 20% of the picture carrier level. For example, if the peak video signal offset of pin 12 with respect to pin 13 is 3V, this corresponds to a 30 mVrms picture RF carrier. The source impedance at pin 12 is defined by the external 2 kΩ resistor and so a series network of 15 kΩ and 24 pF will give a sound carrier level at −32 dB to the picture carrier.

RF Modulation

Two RF channels are available, with carrier frequencies up to 100 MHz being determined by L-C tank circuits at pins 6, 7, 8 and 9. The signal inputs (pins 12, 13) to both modulators are common, but removing the power supply from an RF oscillator tank circuit will also disable that modulator.

As with the chrominance modulators, it is the offset between the two signal input pins that determines the level of RF carrier output. Since one signal input (pin 13) is also internally connected to the chrominance modulators, the 2 kΩ load resistor at this point should be connected to a bias source within the common-mode input range of the video modulators. However, this bias source is independent of the chrominance modulator bias and where chrominance modulation is not used, the 2 kΩ resistor is eliminated and the bias source connected directly to pin 13.

To preserve the dc content of the video signal, amplitude modulation of the RF carrier is done in one direction only, with increasing video (toward peak white) decreasing the carrier level. This means the active composite video signal at pin 12 must be offset with respect to pin 13 and the sync pulse should produce the largest offset (i.e., the offset voltage of pin 12 with respect to pin 13 should have the same polarity as the sync pulses.

The largest video signal (peak white) should not be able to suppress the carrier completely, particularly if sound transmission is needed. For example, a signal with 1V sync amplitude and 2.5V peak white (3.5 Vp-p, negative polarity sync) and a black level at 5 V_{dc} will require a dc bias of 8V on pin 13 for correct modulation. A simple way of obtaining the required offset is to bias pin 13 at 4 × (sync amplitude) from the sync tip level at pin 12.

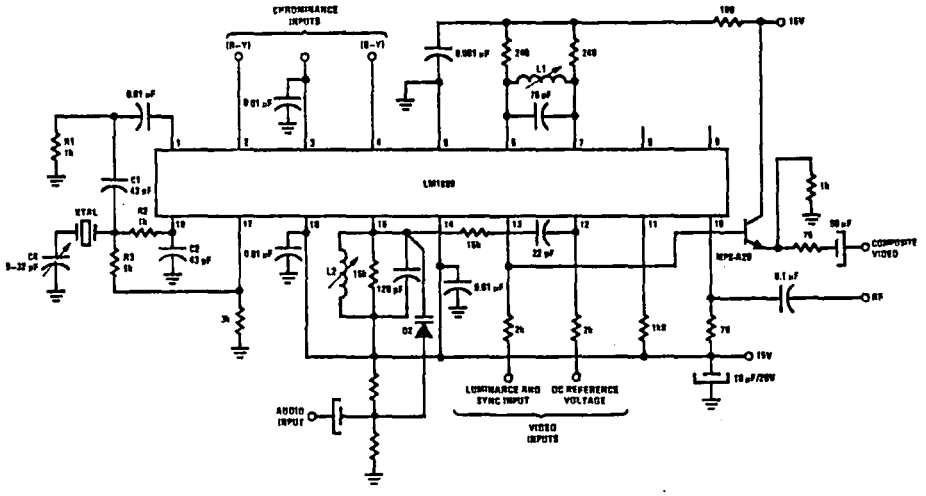
Applications Information (Continued)

Split Power Supplies

The LM1889 is designed to operate over a wide range of supply voltages so that much of the time it can utilize the signal source power supplies. An example of this is shown in *Figure 2* where the composite video signal from a character generator is modulated onto an RF carrier for display on a conventional home TV_receiver. The LM1889 is biased between the −12V and +5V supplies and pin 13 is put at ground. A 9.1 kΩ resistor from pin 12 to −12V dc offsets the video input signal (which has sync tips at ground) to establish the proper modulation depth − R1/R2 = V_{IN}/12 x 0.875. This design is for monochrome transmission and features an extremely low external parts count.

DC Clamped Inputs

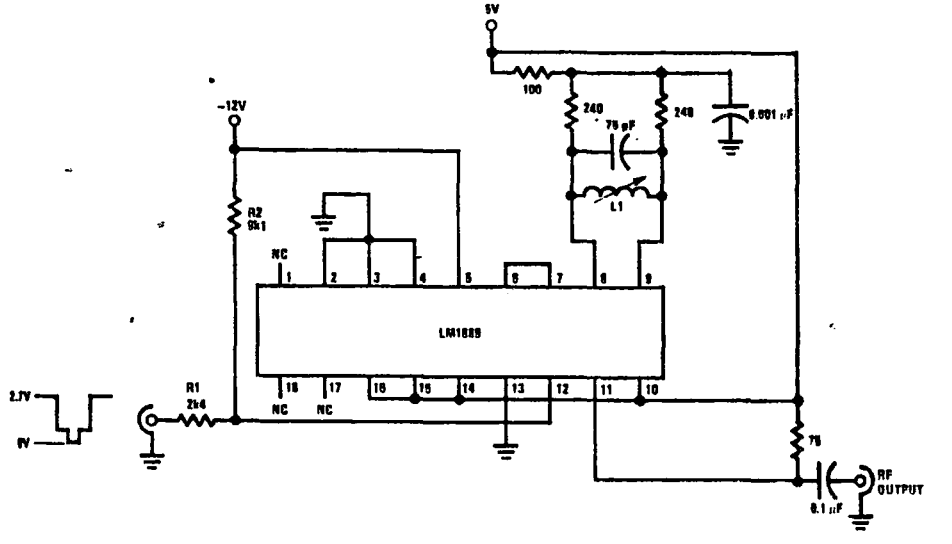
Utilizing a DC clamp will make matching the LM1889 to available signal generator outputs a simple process. *Figure 3* shows the LM1889 configured to accept the composite video patterns available from a Tektronix Type 144 generator that has black level at ground and negative polarity syncs. In this application, the chroma oscillator amplifier is used to provide a gain of two. The 100k pot adjusts the overall DC level of the amplified signal which determines the modulation depth of the RF output. Clamping the input requires a minimum of DC correction to obtain the correct DC output level. This allows the adjustment to be a high impedance that will have minimum effect on the amplifier closed loop gain.



TL/H/7917-7

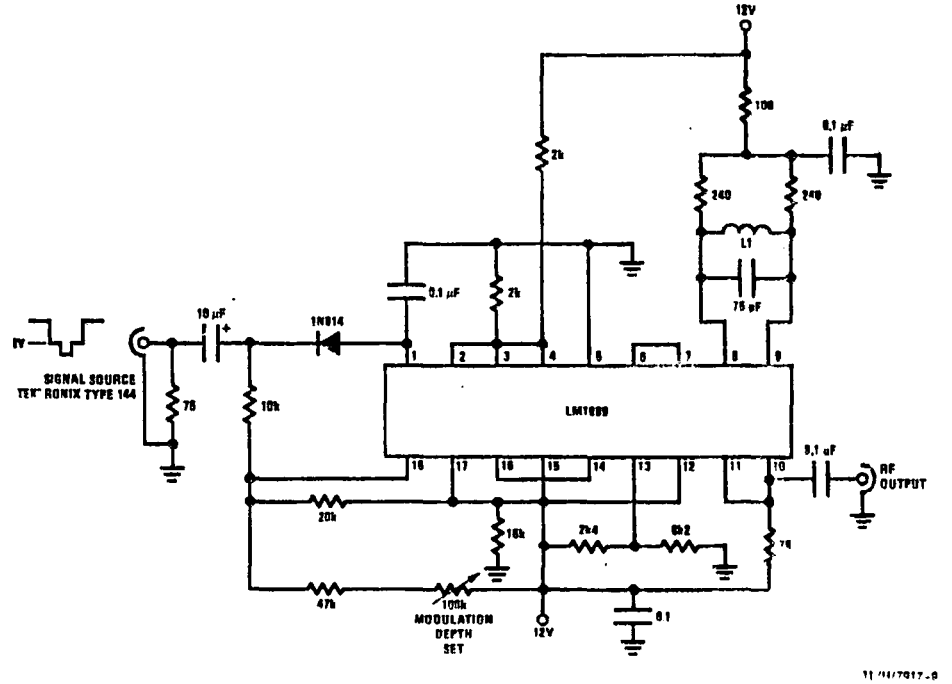
FIGURE 1. Luminance and Chrominance Encoding Composite Video or RF Output

Applications Information (Continued)



TL/H/7917-8

FIGURE 2. Low-Cost Monochrome Modulator for Character Generator Display



TL/H/7917-9

FIGURE 3. DC Clamped Modulator for NTSC Pattern Generators

MOTOROLA
SEMICONDUCTOR
TECHNICAL DATA

MC6845

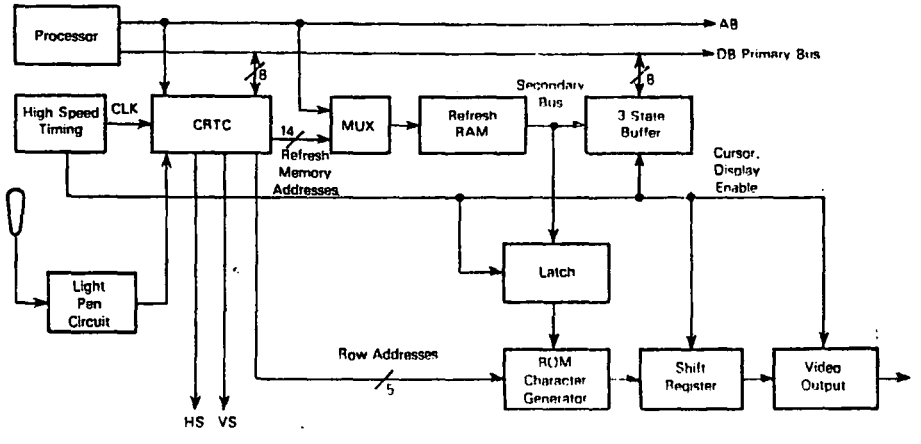
CRT Controller (CRTC)

The MC6845 CRT controller performs the interface between an MPU and a raster-scan CRT display. It is intended for use in MPU-based controllers for CRT terminals in stand-alone or cluster configurations.

The CRTC is optimized for the hardware/software balance required for maximum flexibility. All keyboard functions, reads, writes, cursor movements, and editing are under processor control. The CRTC provides video timing and refresh memory addressing.

- Useful in Monochrome or Color CRT Applications
- Applications Include "Glass-Teletype," Smart, Programmable, Intelligent CRT Terminals; Video Games; Information Displays
- Alphanumeric, Semi-Graphic, and Full-Graphic Capability
- Fully Programmable Via Processor Data Bus. Timing May Be Generated for Almost Any Alphanumeric Screen Format, e.g., 80 × 24, 72 × 64, 132 × 20
- Single +5 V Supply
- M6800 Compatible Bus Interface
- TTL-Compatible Inputs and Outputs
- Start Address Register Provides Hardware Scroll (by Page or Character)
- Programmable Cursor Register Allows control of Cursor Format and Blink Rate
- Light Pen Register
- Refresh (Screen) Memory May be Multiplexed Between the CRTC and the MPU Thus Removing the Requirements for Line Buffers or External DMA Devices
- Programmable Interlace or Non-Interlace Scan Modes
- 14-Bit Refresh Address Allows Up to 16K of Refresh Memory for Use in Character or Semi-Graphic Displays
- 5-Bit Row Address Allows Up to 32 Scan-Line Character Blocks
- By Utilizing Both the Refresh Addresses and the Row Addresses, a 512K Address Space is Available for Use in Graphics Systems
- Refresh Addresses are Provided During Retrace, Allowing the CRTC to Provide Row Addresses to Refresh Dynamics RAMs
- Pin Compatible with the MC6835

FIGURE 1 — TYPICAL CRT CONTROLLER APPLICATION



MAXIMUM RATINGS

Rating	Symbol	Value	Unit
Supply Voltage	V_{CC}	0.3 to +7.0	V
Input Voltage	V_{in}	0.3 to +7.0	V
Operating Temperature Range MC6845, MC68A45, MC68B45 MC6845C, MC68A45C	T_A	T_L to T_H 0 to 70 40 to +85	C
Storage Temperature Range	T_{stg}	55 to +150	C

The device contains circuitry to protect the inputs against damage due to high static voltages or electric fields; however, it is advised that normal precautions be taken to avoid application of any voltage higher than maximum rated voltages to this high-impedance circuit. For proper operation it is recommended that V_{in} and V_{out} be constrained to the range $V_{SS} \leq (V_{in}$ or $V_{out}) \leq V_{CC}$.

THERMAL CHARACTERISTICS

Characteristic	Symbol	Value	Unit
Thermal Resistance Plastic Package Cerdip Package	θ_{JA}	 100 60	C/W

RECOMMENDED OPERATING CONDITIONS

Characteristics	Symbol	Min	Typ	Max	Unit
Supply Voltage	V_{CC}	4.75	5.0	5.25	V
Input Low Voltage	V_{IL}	-0.3	–	0.8	V
Input High Voltage	V_{IH}	2.0	–	V_{CC}	V

POWER CONSIDERATIONS

The average chip-junction temperature, T_J, in °C can be obtained from:

$$T_J = T_A + (P_D \cdot \theta_{JA}) \quad (1)$$

where:

T_A = Ambient Temperature, °C
θ_{JA} = Package Thermal Resistance, Junction-to-Ambient, °C/W
P_D = $P_{INT} + P_{PORT}$
P_{INT} = $I_{CC} \times V_{CC}$, Watts — Chip Internal Power
P_{PORT} = Port Power Dissipation, Watts — User Determined

For most applications $P_{PORT} < P_{INT}$ and can be neglected. P_{PORT} may become significant if the device is configured to drive Darlington bases or sink LED loads.

An approximate relationship between P_D and T_J (if P_{PORT} is neglected) is:

$$P_D = K + (T_J + 273°C) \quad (2)$$

Solving equations (1) and (2) for K gives:

$$K = P_D \cdot (T_A + 273°C) + \theta_{JA} \cdot P_D^2 \quad (3)$$

where K is a constant pertaining to the particular part, K can be determined from equation (3) by measuring P_D (at equilibrium) for a known T_A. Using this value of K, the values of P_D and T_J can be obtained by solving equations (1) and (2) iteratively for any value of T_A

DC ELECTRICAL CHARACTERISTICS (V_{CC} = 5.0 Vdc ± 5%, V_{SS} = 0, T_A = 0 to 70°C unless otherwise noted, see Figures 2-4)

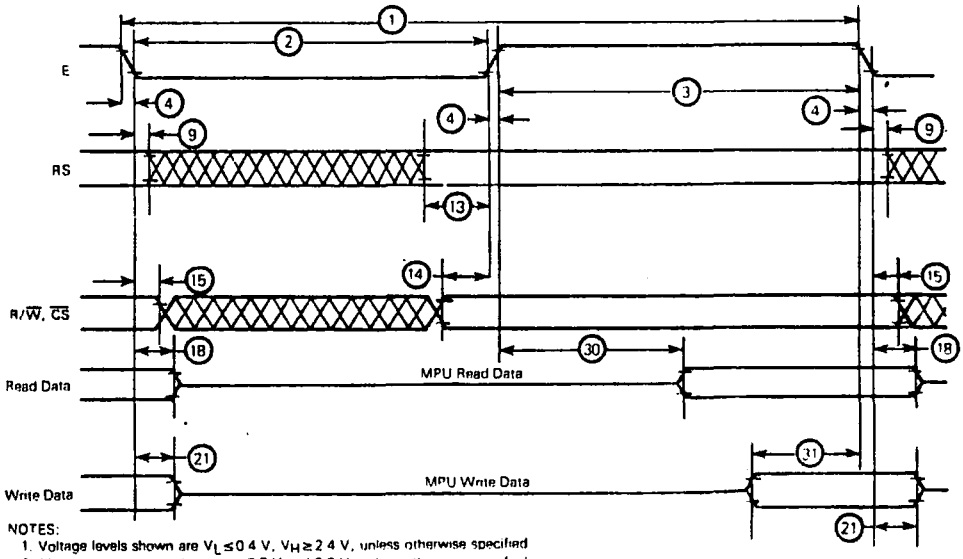
Characteristic		Symbol	Min	Typ	Max	Unit
Input High Voltage		V_{IH}	2.0	–	V_{CC}	V
Input Low Voltage		V_{IL}	−0.3	–	0.8	V
Input Leakage Current		I_{in}	–	0.1	2.5	μA
Hi-Z State Input Current (V_{CC} = 5.25 V) (V_{in} = 0.4 to 2.4 V)		I_{TSI}	−10	–	10	μA
Output High Voltage (I_{Load} = −205 μA) (I_{Load} = −100 μA)	D0-D7 Other Outputs	V_{OH}	2.4 2.4	3.0 3.0	– –	V
Output Low Voltage (I_{Load} = 1.6 mA)		V_{OL}	–	0.3	0.4	V
Internal Power Dissipation (Measured at T_A = 0°C)		P_{INT}	–	600	750	mW
Input Capacitance	D0-D7 All Others	C_{in}	– –	– –	12.5 10	pF
Output Capacitance	All Outputs	C_{out}	–	–	10	pF

BUS TIMING CHARACTERISTICS (See Notes 1 and 2) (Reference Figures 2 and 3)

Ident. Number	Characteristic	Symbol	MC6845		MC68A45		MC68B45		Unit
			Min	Max	Min	Max	Min	Max	
1	Cycle Time	t_{cyc}	1.0	10**	0.67	10	0.5	10**	µs
2	Pulse Width, E Low	PW_{EL}	430	–	280	–	210	–	ns
3	Pulse Width, E High	PW_{EH}	450	–	280	–	220	–	ns
4	Clock Rise and Fall Time	t_r, t_f	–	25	–	25	–	20	ns
9	Address Hold Time (RS)	t_{AH}	10	–	10	–	10	–	ns
13	RS Setup Time Before E	t_{AS}	80	–	60	–	40	–	ns
14	R/$\overline{W}$ and $\overline{CS}$ Setup Time Before E	t_{CS}	80	–	60	–	40	–	ns
15	R/$\overline{W}$ and $\overline{CS}$ Hold Time	t_{CH}	10	–	10	–	10	–	ns
18	Read Data Hold Time	t_{DHR}	20	50*	20	50*	20	50*	ns
21	Write Data Hold Time	t_{DHW}	10	–	10	–	10	–	ns
30	Peripheral Output Data Delay Time	t_{DDR}	–	290	–	180	0	150	ns
31	Peripheral Input Data Setup Time	t_{DSW}	165	–	80	–	60	–	ns

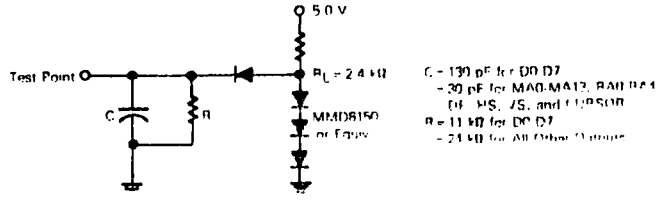
* The data bus output buffers are no longer sourcing or sinking current by t_{DHR} maximum (high-impedance)
** The E clock may be low for extended periods provided the CLK input is active

FIGURE 2 – MC6845 BUS TIMING



NOTES:
1. Voltage levels shown are $V_L \leq 0.4$ V, $V_H \geq 2.4$ V, unless otherwise specified
2. Measurement points shown are 0.8 V and 2.0 V, unless otherwise specified

FIGURE 3 – BUS TIMING TEST LOAD



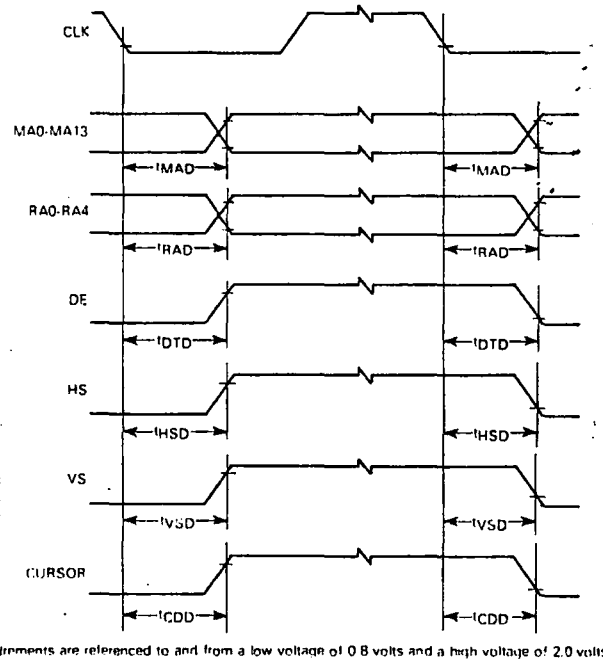
C = 130 pF for D0-D7
= 30 pF for MA0-MA13, RA0-RA4, DE, HS, VS, and CURSOR
R = 11 kΩ for D0-D7
= 24 kΩ for All Other Outputs

CRTC TIMING CHARACTERISTICS (Reference Figures 4 and 5)

Characteristic	Symbol	Min	Max	Unit
Minimum Clock Pulse Width, Low	P_{WCL}	150	–	ns
Minimum Clock Pulse Width, High	P_{WCH}	150	–	ns
Clock Frequency	f_c	–	3.0	MHz
Rise and Fall Time for Clock Input	t_{cr}, t_{cf}	–	20	ns
Memory Address Delay Time	t_{MAD}	–	160	ns
Raster Address Delay Time	t_{RAD}	–	160	ns
Display Timing Delay Time	t_{DTD}	–	250	ns
Horizontal Sync Delay Time	t_{HSD}	–	250	ns
Vertical Sync Delay Time	t_{VSD}	–	250	ns
Cursor Display Timing Delay Time	t_{CDD}	–	250	ns
Light Pen Strobe Minimum Pulse Width	PW_{LPH}	80	–	ns
Light Pen Strobe Disable Time	t_{LPD1}	–	80	ns
	t_{LPD2}	–	10	ns

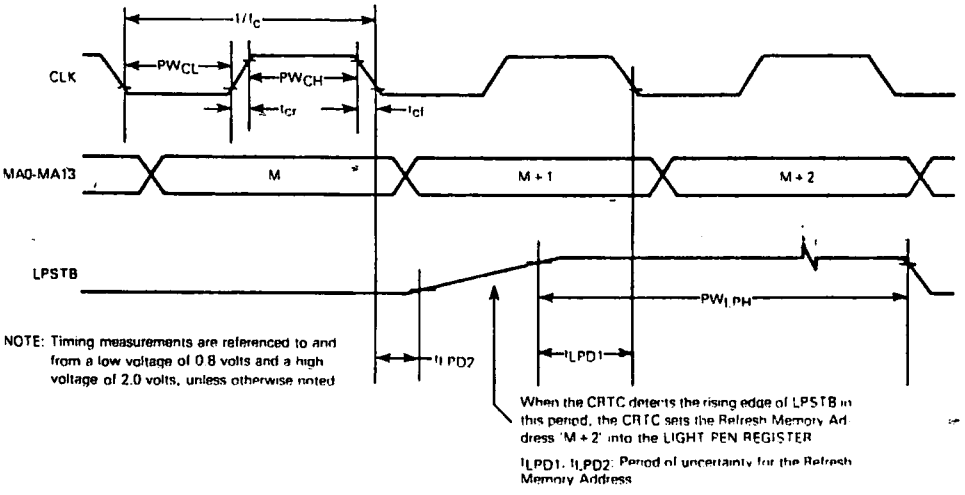
NOTE The light pen strobe must fall to low level before VS pulse rises.

FIGURE 4 – CRTC TIMING CHART



NOTE Timing measurements are referenced to and from a low voltage of 0.8 volts and a high voltage of 2.0 volts unless otherwise noted.

FIGURE 5 — CRTC-CLK, MA0-MA13, AND LPSTB TIMING DIAGRAM



CRTC INTERFACE SYSTEM DESCRIPTION

The CRT controller generates the signals necessary to interface a digital system to a raster scan CRT display. In this type of display, an electron beam starts in the upper left hand corner, moves quickly across the screen and returns. This action is called a horizontal scan. After each horizontal scan the beam is incrementally moved down in the vertical direction until it has reached the bottom. At this point one frame has been displayed, as the beam has made many horizontal scans and one vertical scan.

Two types of raster scanning are used in CRTs, interlace and non-interlace, shown in Figures 6 and 7. Non-interlace scanning consists of one field per frame. The scan lines in Figure 6 are shown as solid lines and the retrace patterns are indicated by the dotted lines. Increasing the number of frames per second will decrease the flicker. Ordinarily, either a 50 or 60 frame per second refresh rate is used to minimize beating between the CRT and the power line frequency. This prevents the displayed data from weaving.

Interlace scanning is used in broadcast TV and on data monitors where high density or high resolution data must be displayed. Two fields, or vertical scans are made down the screen for each single picture or frame. The first field (even field) starts in the upper left hand corner; the second (odd field) in the upper center. Both fields overlap as shown in Figure 7, thus interlacing the two fields into a single frame.

In order to display the characters on the CRT screen the frames must be continually repeated. The data to be displayed is stored in the refresh (screen) memory by the MPU controlling the data processing system. The data is usually written in ASCII code, so it cannot be directly displayed as characters. A character generator ROM is typically used to convert the ASCII codes into the "dot" pattern for every character.

The most common method of generating characters is to create a matrix of dots "x" dots (columns) wide and "y" dots (rows) high. Each character is created by selectively filling in

FIGURE 6 — RASTER SCAN SYSTEM (NON-INTERLACE)

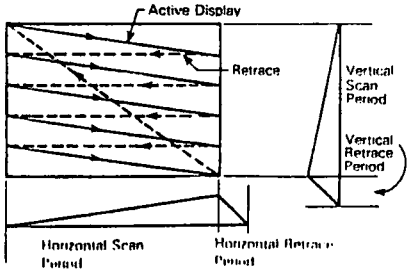
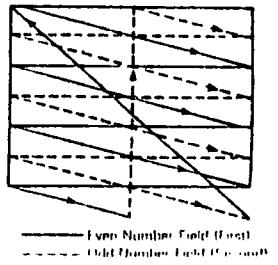


FIGURE 7 — RASTER SCAN SYSTEM (INTERLACE)



the dots. As "x" and "y" get larger a more detailed character may be created. Two common dot matrices are 5 x 7 and 7 x 9. Many variations of these standards will allow Chinese, Japanese, or Arabic letters instead of English. Since characters require some space between them, a character block larger than the character is typically used, as shown in Figure 8. The figure also shows the corresponding timing and levels for a video signal that would generate the characters.

Referring to Figure 1, the CRT controller generates the refresh addresses (MA0-MA13), row addresses (RA0-RA4), and the video timing (vertical sync – VS, horizontal sync – HS, and display enable – DE). Other functions include an internal cursor register which generates a cursor output when its contents compare to the current refresh address. A light pen strobe input signal allows capture of the refresh address in an internal light pen register.

All timing in the CRTC is derived from the CLK input. In alphanumeric terminals, this signal is the character rate. The video rate or "dot" clock is externally divided by high-speed logic (TTL) to generate the CLK input. The high-speed logic must also generate the timing and control signals necessary for the shift register, latch, and MUX control.

The processor communicates with the CRTC through an 8-bit data bus by reading or writing into the 19 registers.

The refresh memory address is multiplexed between the processor and the CRTC. Data appears on a secondary bus separate from the processor's bus. The secondary data bus concept in no way precludes using the refresh RAM for other purposes. It looks like any other RAM to the processor. A number of approaches are possible for solving contentions for the refresh memory:

1. Processor always gets priority. (Generally, "hash" occurs as MPU and CRTC clocks are not synchronized.)
2. Processor gets priority access anytime, but can be synchronized by an interrupt to perform accesses only during horizontal and vertical retrace times.
3. Synchronize the processor with memory wait cycles (states).
4. Synchronize the processor to the character rate as shown in Figure 9. The M6800 processor family works works very well in this configuration as constant cycle lengths are present. This method provides no overhead for the processor as there is never a contention for a memory access. All accesses are transparent.

FIGURE 8 – CHARACTER DISPLAY ON THE SCREEN AND VIDEO SIGNAL

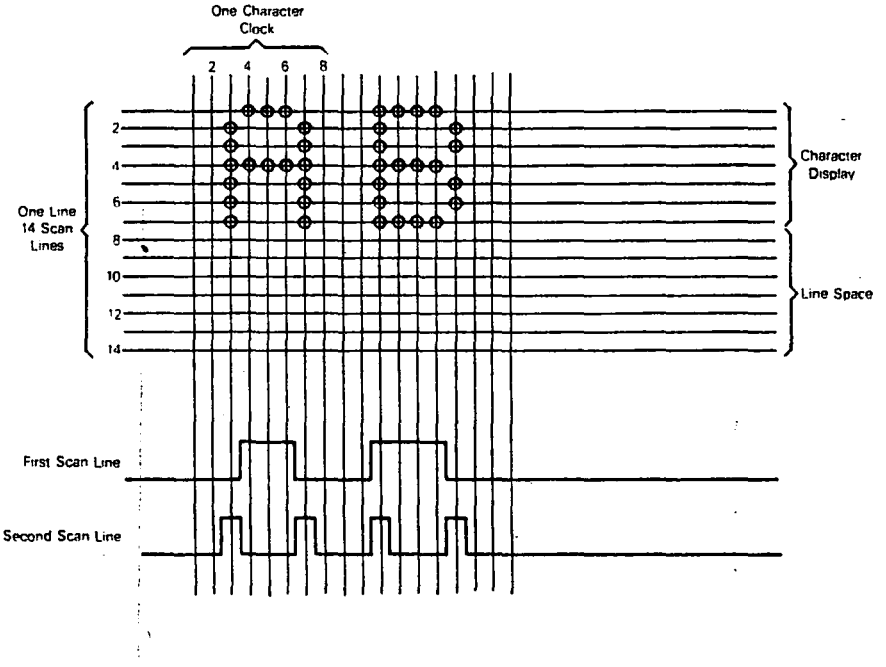
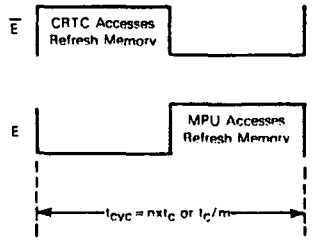


FIGURE 9 — TRANSPARENT REFRESH MEMORY CONFIGURATION TIMING USING M6800 FAMILY MPU



Where: m, n are integers; t_c is character period

PIN DESCRIPTION

PROCESSOR INTERFACE

The CRTC interfaces to a processor bus on the bidirectional data bus (D0-D7) using $\overline{CS}$, RS, E, and R/$\overline{W}$ for control signals.

Data Bus (D0-D7) — The bidirectional data lines (D0-D7) allow data transfers between the internal CRTC register file and the processor. Data bus output drivers are in the high-impedance state until the processor performs a CRTC read operation.

Enable (E) — The enable signal is a high-impedance TTL/MOS compatible input which enables the data bus input/output buffers and clocks data to and from the CRTC. This signal is usually derived from the processor clock. The high-to-low transition is the active edge.

Chip Select ($\overline{CS}$) — The $\overline{CS}$ line is a high-impedance TTL/MOS compatible input which selects the CRTC, when low, to read or write to the internal register file. This signal should only be active when there is a valid stable address being decoded from the processor.

Register Select (RS) — The RS line is a high-impedance TTL/MOS compatible input which selects either the address register (RS = 0) or one of the data register (RS = 1) of the internal register file.

Read/Write (R/$\overline{W}$) — The R/$\overline{W}$ line is a high-impedance TTL/MOS compatible input which determines whether the internal register file gets written or read. A write is defined as a low level.

CRT CONTROL

The CRTC provides horizontal sync (HS), vertical sync (VS), and display enable (DE) signals

NOTE

Care should be exercised when interfacing to CRT monitors, as many monitors claiming to be "TTL compatible" have transistor input circuits which require the CRTC or TTL devices buffering signals from the CRTC/video circuits to exceed the maximum rated drive currents.

Vertical Sync (VS) and Horizontal Sync (HS) — These TTL-compatible outputs are active high signals which drive the monitor directly or are fed to the video processing circuitry to generate a composite video signal. The VS signal determines the vertical position of the displayed text while the HS signal determines the horizontal position of the displayed text.

Display Enable (DE) — This TTL-compatible output is an active high signal which indicates the CRTC is providing addressing in the active display area.

REFRESH MEMORY/CHARACTER GENERATOR ADDRESSING

The CRTC provides memory addresses (MA0-MA13) to scan the refresh RAM. Row addresses (RA0-RA4) are also provided for use with character generator ROMs. In a graphics system, both the memory addresses and the row addresses would be used to scan the refresh RAM. Both the memory addresses and the row addresses continue to run during vertical retrace thus allowing the CRTC to provide the refresh addresses required to refresh dynamic RAMs.

Refresh Memory Addresses (MA0-MA13) — These 14 outputs are used to refresh the CRT screen with pages of data located within a 16K block of refresh memory. These outputs are capable of driving one standard TTL load and 30 pF.

Row Addresses (RA0-RA4) — These five outputs from the internal row address counter are used to address the character generator ROM. These outputs are capable of driving one standard TTL load and 30 pF.

OTHER PINS

Cursor — This TTL-compatible output indicates a valid cursor address to external video processing logic. It is an active high signal.

Clock (CLK) — The CLK is a TTL/MOS-compatible input used to synchronize all CRT functions except for the processor interface. An external dot counter is used to derive this signal which is usually the character rate in an alphanumeric CRT. The actual transition is high to low

Light Pen Strobe (LPSTB) – A low-to-high transition on this high-impedance TTL/MOS-compatible input latches the current Refresh Address in the light pen register. The latching of the refresh address is internally synchronized to the character clock (CLK).

V_{CC} and V_{SS} – These inputs supply +5 Vdc ±5% to the CRTC.

$\overline{RESET}$ – The $\overline{RESET}$ input is used to reset the CRTC. A low level on the $\overline{RESET}$ input forces the CRTC into the following state:

(a) All counters in the CRTC are cleared and the device stops the display operation.

(b) All the outputs are driven low.

NOTE

The horizontal sync output is not defined until after R2 is programmed.

(c) The control registers of the CRTC are not affected and remain unchanged.

Functionality of $\overline{RESET}$ differs from that of other M6800 parts in the following functions:

(a) The $\overline{RESET}$ input and the LPSTB input are encoded as shown in Table 1.

TABLE 1 – CRTC OPERATING MODE

$\overline{RESET}$	LPSTB	Operating Mode
0	0	Reset
0	1	Test Mode
1	0	Normal Mode
1	1	Normal Mode

The test mode configures the memory addresses as two independent 7-bit counters to minimize test time.

(b) After $\overline{RESET}$ has gone low and (LPSTB = 0), MA0-MA13 and RA0-RA4 will be driven low on the falling edge of CLK. $\overline{RESET}$ must remain low for at least one cycle of the character clock (CLK).

(c) The CRTC resumes the display operation immediately after the release of $\overline{RESET}$. DE and the CURSOR are not active until after the first frame has been displayed.

CRTC DESCRIPTION

The CRTC consists of programmable horizontal and vertical timing generators, programmable linear address register, programmable cursor logic, light pen capture register, and control circuitry for interface to a processor bus. A block diagram of the CRTC is shown in Figure 10.

All CRTC timing is derived from the CLK, usually the output of an external dot rate counter. Coincidence (CO) circuits continuously compare counter contents to the contents of the programmable register file, R0-R17. For horizontal timing generation, comparisons result in: 1) horizontal sync pulse (HS) of a frequency, position, and width determined by the registers; 2) horizontal display signal of a frequency, position, and duration determined by the registers.

The horizontal counter produces H clock which drives the scan line counter and vertical control. The contents of the raster counter are continuously compared to the maximum scan line address register. A coincidence resets the raster counter and clocks the vertical counter.

Comparisons of vertical counter contents and vertical registers result in: 1) vertical sync pulse (VS) of a frequency and position determined by the registers; 2) vertical display of a frequency and position determined by the registers.

The vertical control logic has other functions.

1. Generate row selects, RA0-RA4, from the raster count for the corresponding interlace or non-interlace modes.
2. Extend the number of scan lines in the vertical total by the amount programmed in the vertical total adjust register.

The linear address generator is driven by the CLK and locates the relative positions of characters in memory with their positions on the screen. Fourteen lines, MA0-MA13, are available for addressing up to four pages of 4K characters, eight pages of 2K characters, etc. Using the start address register, hardware scrolling through 16K characters is possible. The linear address generator repeats the same sequence of addresses for each scan line of a character row.

The cursor logic determines the cursor location, size, and blink rate on the screen. All are programmable.

The light pen strobe going high causes the current contents of the address counter to be latched in the light pen register. The contents of the light pen register are subsequently read by the processor.

Internal CRTC registers are programmed by the processor through the data bus, D0-D7, and the control signals – R/$\overline{W}$, $\overline{CS}$, RS, and E.

REGISTER FILE DESCRIPTIONS

The nineteen registers of the CRTC may be accessed through the data bus. Only two memory locations are required as one location is used as a pointer to address one of the remaining eighteen registers. These eighteen registers control horizontal timing, vertical timing, interlace operation, row address operation, and define the cursor, cursor address, start address, and light pen register. The register addresses and sizes are shown in Table 2.

ADDRESS REGISTER

The address register is a 5-bit write-only register used as an "indirect" or "pointer" register. It contains the address of one of the other eighteen registers. When both RS and $\overline{CS}$ are low, the address register is selected. When $\overline{CS}$ is low and RS is high, the register pointed to by the address register is selected.

TIMING REGISTERS R0-R9

Figure 11 shows the visible display area of a typical CRT monitor giving the point of reference for horizontal registers as the left-most displayed character position. Horizontal registers are programmed in character clock time units with respect to the reference as shown in Figure 12. The point of reference for the vertical registers is the top character position displayed. Vertical registers are programmed in scan line times with respect to the reference as shown in Figure 13.

Horizontal Total Register (R0) – This 8-bit write-only register determines the horizontal sync (HS) frequency by defining the HS period in character times. It is the total of the displayed characters plus the non-displayed character times (retrace) minus one.

FIGURE 10 – CRTC BLOCK DIAGRAM

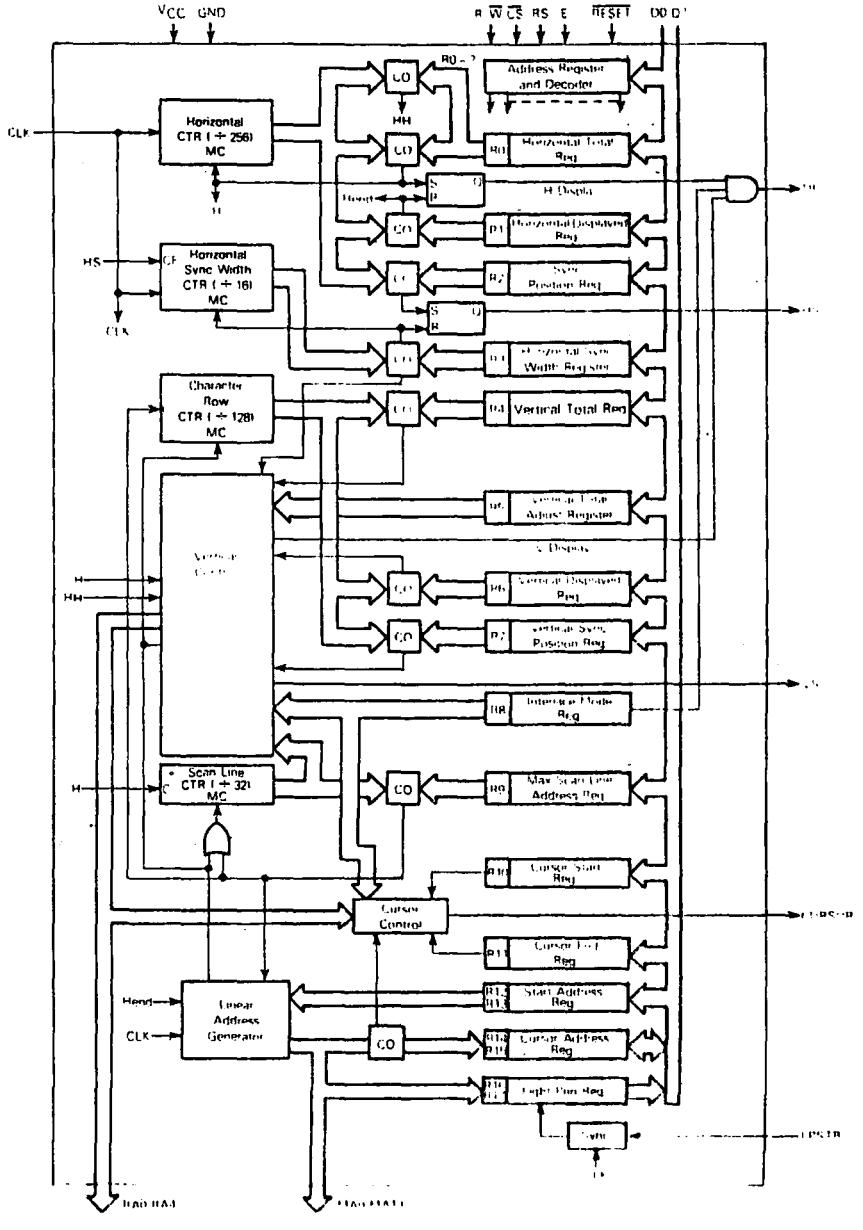
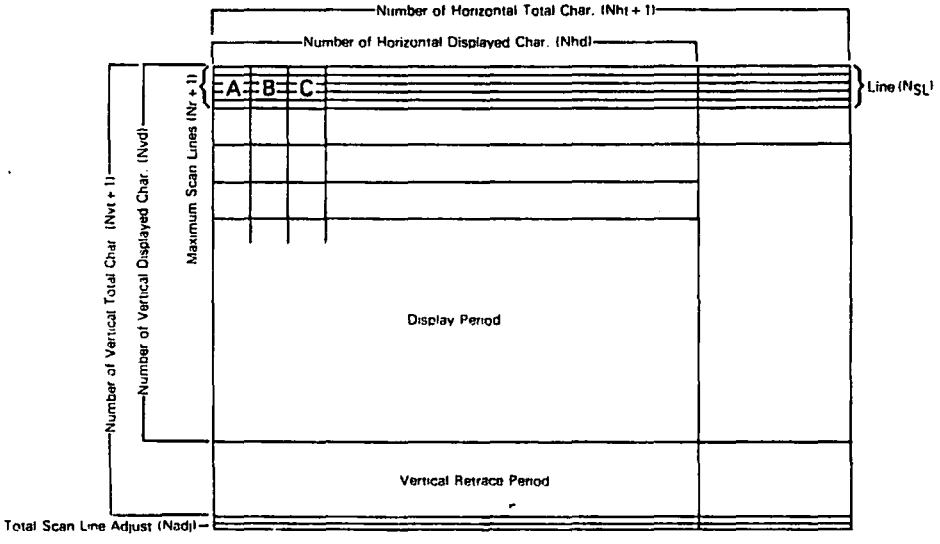


TABLE 2 — CRTC INTERNAL REGISTER ASSIGNMENT

$\overline{CS}$	RS	Address Register 4	3	2	1	0	Register #	Register File	Program Unit	Read	Write	Number of Bits 7	6	5	4	3	2	1	0
1	X	X	X	X	X	X	X	–	–	–	–								
0	0	X	X	X	X	X	AR	Address Register	–	No	Yes								
0	1	0	0	0	0	0	R0	Horizontal Total	Char.	No	Yes								
0	1	0	0	0	0	1	R1	Horizontal Displayed	Char.	No	Yes								
0	1	0	0	0	1	0	R2	H. Sync Position	Char.	No	Yes								
0	1	0	0	0	1	1	R3	Sync Width	–	No	Yes					H	H	H	H
0	1	0	0	1	0	0	R4	Vertical Total	Char. Row	No	Yes								
0	1	0	0	1	0	1	R5	V. Total Adjust	Scan Line	No	Yes								
0	1	0	0	1	1	0	R6	Vertical Displayed	Char. Row	No	Yes								
0	1	0	0	1	1	1	R7	V. Sync Position	Char. Row	No	Yes								
0	1	0	1	0	0	0	R8	Interlace Mode and Skew	Note 1	No	Yes							I	I
0	1	0	1	0	0	1	R9	Max Scan Line Address	Scan Line	No	Yes								
0	1	0	1	0	1	0	R10	Cursor Start	Scan Line	No	Yes		B	P			(Note 2)		
0	1	0	1	0	1	1	R11	Cursor End	Scan Line	No	Yes								
0	1	0	1	1	0	0	R12	Start Address (H)	–	No	Yes	0	0						
0	1	0	1	1	0	1	R13	Start Address (L)	–	No	Yes								
0	1	0	1	1	1	0	R14	Cursor (H)	–	Yes	Yes	0	0						
0	1	0	1	1	1	1	R15	Cursor (L)	–	Yes	Yes								
0	1	1	0	0	0	0	R16	Light Pen (H)	–	Yes	No	0	0						
0	1	1	0	0	0	1	R17	Light Pen (L)	–	Yes	No								

NOTES:
1. The interlace is shown in Table 3.
2. Bit 5 of the cursor start raster register is used for blink period control, and bit 6 is used to select blink or no-blink.

FIGURE 11 — ILLUSTRATION OF THE CRT SCREEN FORMAT



NOTE 1: Timing values are described in Table 5.

Horizontal Displayed Register (R1) – This 8-bit write-only register determines the number of displayed characters per line. Any 8-bit number may be programmed as long as the contents of R0 are greater than the contents of R1.

Horizontal Sync Position Register (R2) – This 8-bit write-only register controls the HS position. The horizontal sync position defines the horizontal sync delay (front porch) and the horizontal scan delay (back porch). When the programmed value of this register is increased, the display on the CRT screen is shifted to the left. When the programmed value is decreased the display is shifted to the right. Any 8-bit number may be programmed as long as the sum of the contents of R2 and R3 are less than the contents of R0. R2 must be greater than R1.

Sync Width Register (R3) – This 8-bit write-only register determines the width of the horizontal sync (HS) pulse. The vertical sync pulse width is fixed at 16 scan-line times.

The HS pulse width may be programmed from 1-to-15 character clock periods thus allowing compatibility with the HS pulse width specifications of many different monitors. If zero is written into this register then no HS is provided.

Horizontal Timing Summary (Figure 12) – The difference between R0 and R1 is the horizontal blanking interval. This interval in the horizontal scan period allows the beam to return (retrace) to the left side of the screen. The retrace time is determined by the monitor's horizontal scan components. Retrace time is less than the horizontal blanking interval. A good rule of thumb is to make the horizontal blanking about 20% of the total horizontal scanning period for a CRT. In inexpensive TV receivers, the beam overscans the display screen so that aging of parts does not result in underscanning. Because of this, the retrace time should be about one third the horizontal scanning period. The horizontal sync delay, HS pulse width, and horizontal scan delay are typically programmed with a 1:2:2 ratio.

Vertical Total Register (R4) and Vertical Total Adjust Register (R5) – The frequency of VS is determined by both R4 and R5. The calculated number of character row times is usually an integer plus a fraction to get exactly a 50 or 60 Hz vertical refresh rate. The integer number of character row times minus one is programmed in the 7-bit write-only vertical total register (R4). The fraction of character line times is programmed in the 5-bit write-only vertical total adjust register (R5) as the number of scan lines required.

Vertical Displayed Register (R6) – This 7-bit write-only register specifies the number of displayed character rows on the CRT screen, and is programmed in character row times. Any number smaller than the contents of R4 may be programmed into R6.

Vertical Sync Position (R7) – This 7-bit write-only register controls the position of vertical sync with respect to the reference. It is programmed in character row times. When the programmed value of this register is increased, the display position of the CRT screen is shifted up. When the programmed value is decreased the display position is shifted down. Any number equal to or less than the vertical total (R4) and greater than or equal to the vertical displayed (R6) may be used.

Interlace Mode and Skew Register (R8) – The MC6845 only allows control of the interlace modes as programmed by the low order two bits of this write-only register. Table 3 shows the interlace modes available to the user. These modes are selected using the two low order bits of this 8 bit write-only register.

TABLE 3 – INTERLACE MODE REGISTER

Bit 1	Bit 0	Mode
0	0	Normal Sync Mode (Non-Interlace)
1	0	
0	1	Interlace Sync Mode
1	1	Interlace Sync and Video Mode

In the normal sync mode (non-interlace) only one field is available as shown in Figures 6 and 14a. Each scan line is refreshed at the VS frequency (e.g., 50 or 60 Hz).

Two interlace modes are available as shown in Figures 7, 14b, and 14c. The frame time is divided between even and odd alternating fields. The horizontal and vertical timing relationship (VS delayed by one half scan line time) results in the displacement of scan lines in the odd field with respect to the even field.

In the interlace sync mode the same information is painted in both fields as shown in Figure 14b. This is a useful mode for filling in a character to enhance readability.

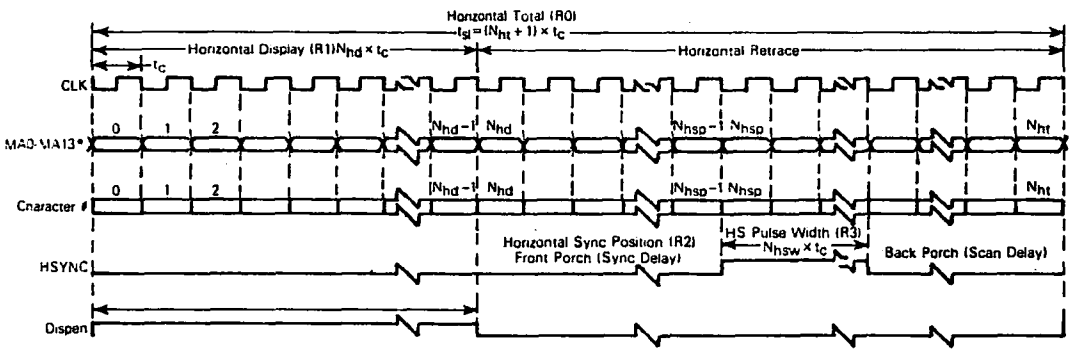
In the interlace sync and video mode, shown in Figure 14c, alternating lines of the character are displayed in the even field and the odd field. This effectively doubles the given bandwidth of the CRT monitor.

Care must be taken when using either interlace mode to avoid an apparent flicker effect. This flicker effect is due to the doubling of the refresh time for all scan lines since each field is displayed alternately and may be minimized with proper monitor design (e.g., longer persistence phosphors).

In addition, there are restrictions on the programming of the CRTC registers for interlace operation:

1. The horizontal total register value, R0, must be odd (i.e., an even number of character times).
2. For interlace sync and video mode only, the maximum scan line address, R9, must be odd (i.e., an even number of scan lines).
3. For interlace sync and video mode only, the number (Nvd) programmed into the vertical display register (R6) must be one half the actual number required. The even numbered scan lines are displayed in the even field and the odd numbered scan lines are displayed in the odd field.
4. For interlace sync and video mode only, the cursor start register (R10) and cursor end register (R11) must both be even or both odd depending on which field the cursor is to be displayed in. A full block cursor will be displayed in both the even and the odd field when the cursor end register (R11) is programmed to a value greater than the value in the maximum scan line address register (R9).

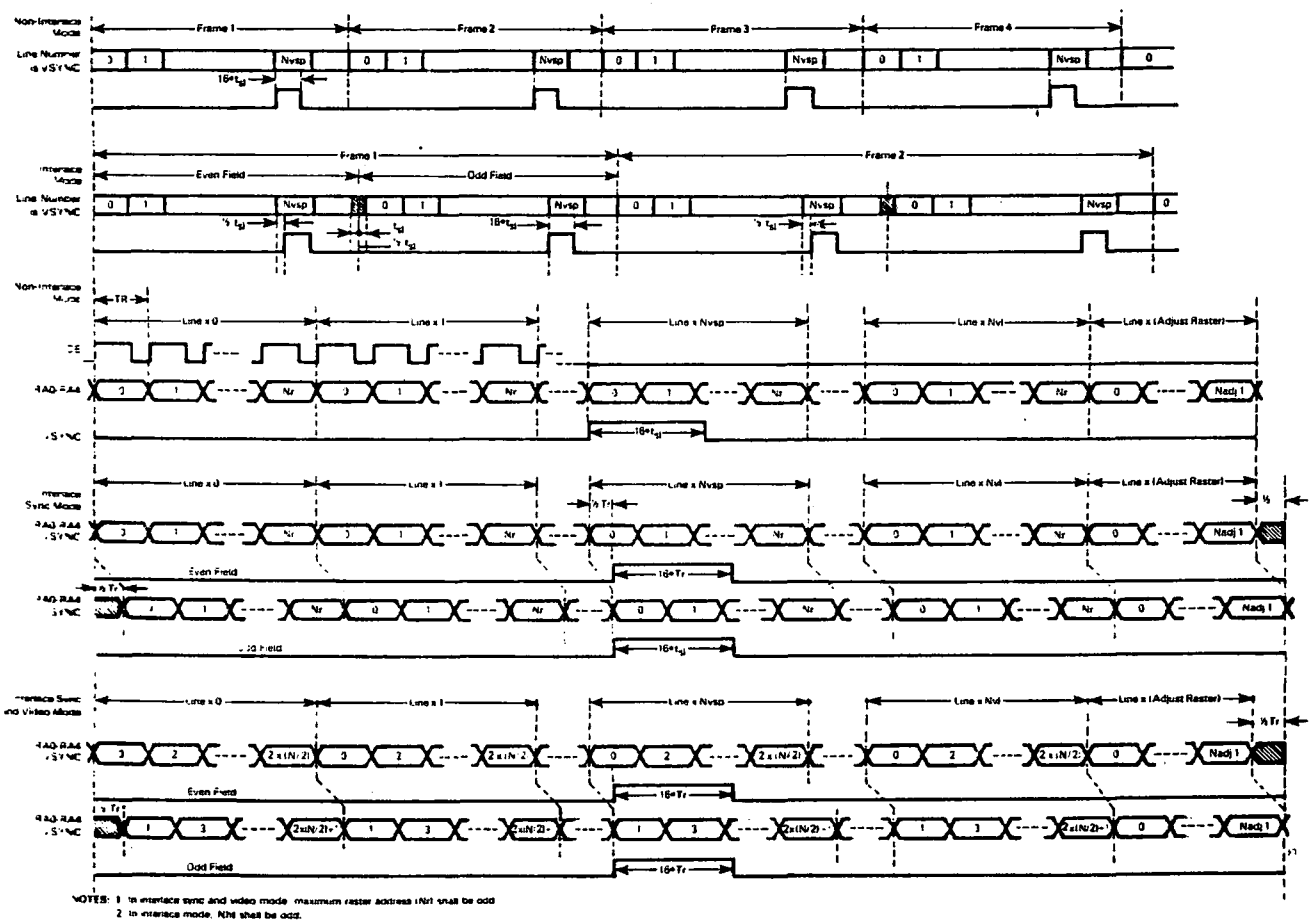
FIGURE 12 – CRTC HORIZONTAL TIMING



* Timing is shown for first displayed scan row only. See chart in Figure 15 for other rows. The initial MA is determined by the contents of start address register, R12/R13. Timing is shown for R12/R13=0.

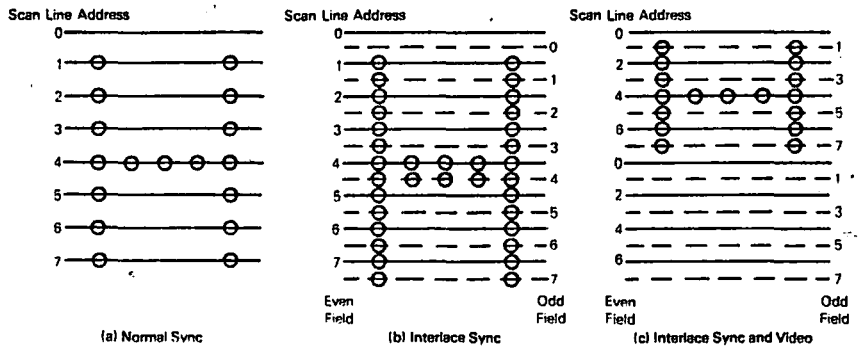
NOTE. Timing values are described in Table 5.

FIGURE 13 – CRTC VERTICAL TIMING



NOTES: 1. In interlace sync and video mode, maximum raster address (Nr) shall be odd.
2. In interlace mode, Nht shall be odd.

FIGURE 14 — INTERLACE CONTROL



Maximum Scan Line Address Register (R9) — This 5-bit write-only register determines the number of scan lines per character row including the spacing; thus, controlling operation of the row address counter. The programmed value is a maximum address and is one less than the number of scan lines.

CURSOR CONTROL

Cursor Start Register (R10) and Cursor End Register (R11) — These registers allow a cursor of up to 32 scan lines in height to be placed on any scan line of the character block as shown in Figure 15. R10 is a 7-bit write-only register used to define the start scan line and the cursor blink rate. Bits 5 and 6 of the cursor start address register control the cursor operation as shown in Table 4. Non-display, display, and two blink modes (16 times or 32 times the field period) are available. R11 is a 5-bit write-only register which defines the last scan line of the cursor.

TABLE 4 — CURSOR START REGISTER

Bit 6	Bit 5	Cursor Display Mode
0	0	Non-Blink
0	1	Cursor Non-Display
1	0	Blink, 1/16 Field Rate
1	1	Blink, 1/32 Field Rate

Example of cursor display mode

When an external blink feature on characters is required, it may be necessary to perform cursor blink externally so that both blink rates are synchronized. Note that an invert/non-invert cursor is easily implemented by programming the CRTC for a blinking cursor and externally inverting the video signal with an exclusive-OR gate.

Cursor Register (R14-H, R15-L) — This 14-bit read/write register pair is programmed to position the cursor anywhere in the refresh RAM area; thus, allowing hardware paging and scrolling through memory without loss of the original cursor position. It consists of an 8-bit low order (MA0-MA7) register and a 6-bit high order (MA8-MA13) register.

OTHER REGISTERS

Start Address Register (R12-H, R13-L) — This 14-bit write-only register pair controls the first address output by the CRTC after vertical blanking. It consists of an 8-bit low order (MA0-MA7) register and a 6-bit high order (MA8-MA13) register. The start address register determines which portion of the refresh RAM is displayed on the CRT screen. Hardware scrolling by character or page may be accomplished by modifying the contents of this register.

Light Pen Register (R16-H, R17-L) — This 14-bit read-only register pair captures the refresh address output by the CRTC on the positive edge of a pulse input to the LPSTB pin. It consists of an 8-bit low order (MA0-MA7) register and a 6-bit high order (MA8-MA13) register. Since the light pen pulse is asynchronous with respect to refresh address timing an internal synchronizer is designed into the CRTC. Due to delays (Figure 5) in this circuit, the value of R16 and R17 will need to be corrected in software. Figure 16 shows an interrupt driven approach although a polling routine could be used.

FIGURE 15 – CURSOR CONTROL

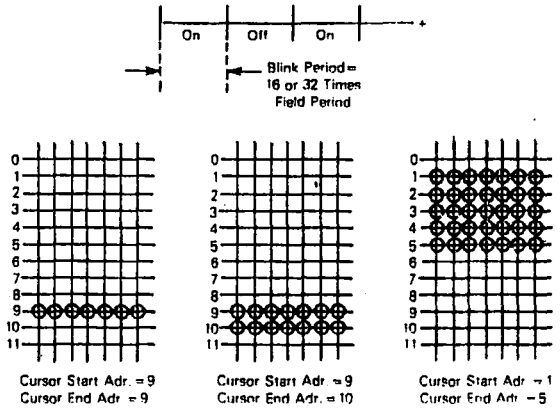
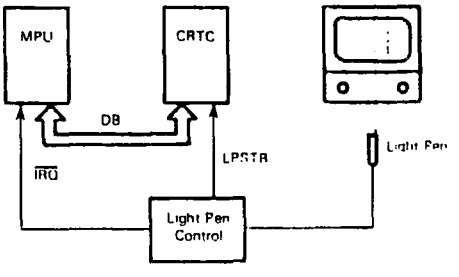


FIGURE 16 – INTERFACING OF LIGHT PEN



OPERATION OF THE CRTC

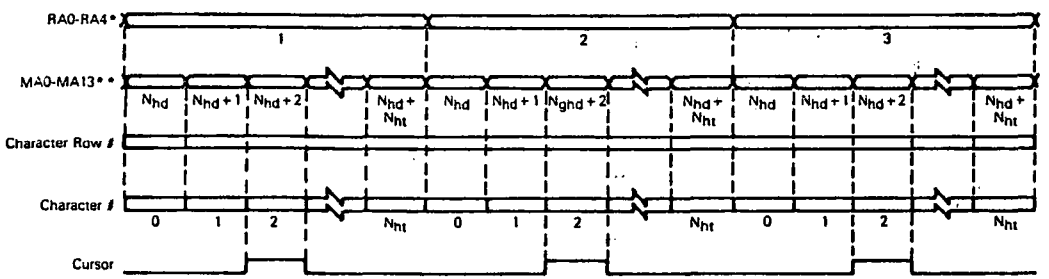
TIMING CHART OF THE CRT INTERFACE SIGNALS

Timing charts of CRT interface signals are illustrated in this section. When values listed in Table 5 are programmed into CRTC control registers, the device provides the outputs as shown in the timing diagrams (Figures 12, 13, 17, and 18). The screen format is shown in Figure 11 which illustrates the relation between refresh memory address (MA0-MA13), raster address (RA0-RA4), and the position on the screen. In this example, the start address is assumed to be zero

TABLE 5 – VALUES PROGRAMMED INTO CRTC REGISTERS

Reg. #	Register Name	Value	Programmed Value
R0	H Total	$N_{ht}+1$	N_{ht}
R1	H Displayed	N_{hd}	N_{hd}
R2	H Sync Position	N_{hsp}	N_{hsp}
R3	H Sync Width	N_{hsw}	N_{hsw}
R4	V Total	$N_{vt}+1$	N_{vt}
R5	V Scan Line Adjust	N_{adj}	N_{adj}
R6	V Displayed	N_{vd}	N_{vd}
R7	V Sync Position	N_{vsp}	N_{vsp}
R8	Interlace Mode		
R9	Max. Scan Line Address	N_{sl}	N_{sl}

FIGURE 17 – CURSOR TIMING

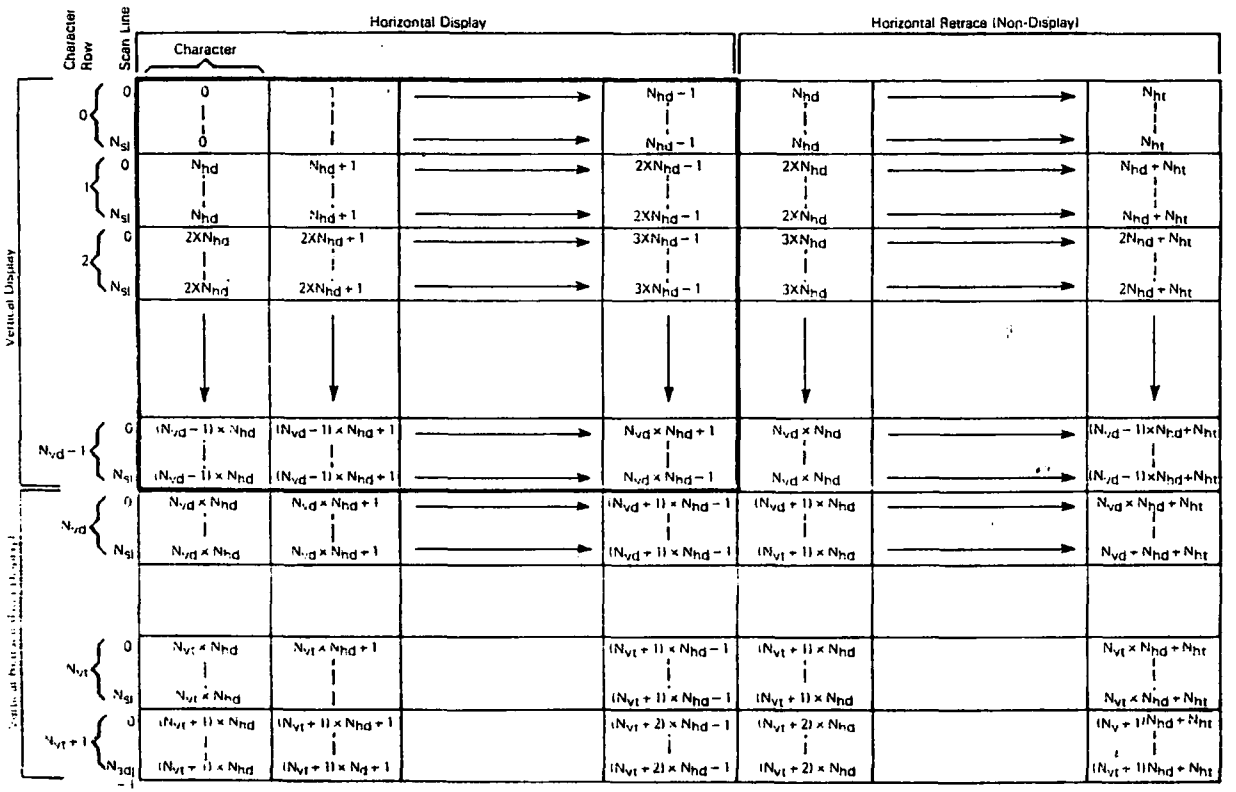


* Timing is shown for non-interlace and interlace sync modes.
Example shown has cursor programmed as:
Cursor Register = $N_{hd} + 2$
Cursor Start = 1
Cursor End = 3

** The initial MA is determined by the contents of start address register, R12/R13. Timing is shown for R12/R13 = 0

NOTE 1: Timing values are described in Table 5

FIGURE 18 — REFRESH MEMORY ADDRESSING (MA0-MA13) STAGE CHART



NOTE 1 The initial MA is determined by the contents of start address register. R12/R13. Timing is shown for R12/R13=0. Only non-interlace and interlace sync modes are shown.

DETERMINING REGISTER CONTENTS

Some of the register contents are determined rather easily. They are:

Register	Name	Contents
R8	Interlace Mode Register	See Table 3
R10	Cursor Start	See Figure 15 and Table 4
R11	Cursor End	See Figure 15
R12	Start Address (H)	User programs first memory location to be displayed
R13	Start Address (L)	
R14	Cursor (H)	User programs desired cursor location
R15	Cursor (L)	
R16	Light Pen (H)	Can be loaded via light-pen strobe only
R17	Light Pen (L)	

The remaining register contents must be determined from some basic data related to the CRT monitor and from the user-desired display format. The CRTC reference sheet (see Figure 19) gives a set of formulas for calculating the register contents as well as other useful characteristics of the display. This type of data is summarized under basic parameters in Figures 20 and 21; most or all of this data must be supplied by the user before he can determine the contents for registers R0-R7 and R9. All variables B_1-B_{10} are equal to basic parameters 1 through 10.

FIGURE 19 — CRTC REFERENCE SHEET

Register Function		Intermediate Calculations			Register Calculations	
R0	Horizontal Total	Symbol	Description	Calculation	Register	Calculation
R1	Horizontal Displayed	f'	Dot frequency (1st approx.)	$\frac{B_5 \cdot (B_7 + B_9)}{(1/B_1) - B_3}$	R0	$\frac{f'}{B_1 \cdot (B_7 + B_9)} - 1$
R2	Horizontal Sync Position					
R3	Horizontal Sync Width	t_c	Character Time	$\frac{1}{[(R0)+1] \cdot B_1}$	R1	B_5
R4	Vertical Total	f	Dot frequency	$\frac{B_7 + B_9}{t_c}$		
R5	Vertical Total Adjust				R2	$(R1) + \frac{(R3)}{2}$
R6	Vertical Displayed	t_{sl}	Scan line time	$[(R0)+1] \cdot t_c$	R3	$\frac{(R0) - (R1)}{3}$
R7	Vertical Sync Position	n	Total # of scan lines	$\frac{1}{B_2 \cdot t_{sl}}$		
R8	Interlace Mode				R4	N − 1
R9	Maximum Scan Line Address	N	Integer	$\frac{n}{B_8 + B_{10}} = N + \frac{R}{B_8 + B_{10}}$	R5	R
R10	Cursor Start	R	and Integer remainder			
R11	Cursor End				R6	B_6
R12	Start Address (H)	t_{cr}	Character row time	$(B_8 + B_{10}) \cdot t_{sl}$	R7	$[(R4)+1] - \frac{16 - (R5)}{B_8 + B_{10}} \geq (R7) \geq (R6)$
R13	Start Address (L)	t_{hr}	Horizontal retrace time	$\leq \frac{[(R0)+1-B_5] \cdot (B_7 + B_9)}{f}$		
R14	Cursor (H)				R9	$(B_8 + B_{10}) - 1$
R15	Cursor (L)	t_{vr}	Vertical retrace time	$\leq \frac{B_1}{B_2} - B_6 (B_8 + B_{10}) \cdot t_{sl}$		
R16	Light Pen (H)					
R17	Light Pen (L)					

In Figures 20 and 21, worksheet example calculations are shown for 32 x 16 and 80 x 24 display formats respectively. The following items are keyed to the figures. Basic parameters (1) through (10) have been provided; items (1) through (4) are data about the CRT monitor and items (5) through (10) are data about the user's desired display.

(1) Calculate the approximate dot frequency. The user should verify that the bandwidth of his CRT monitor will accomodate this frequency.

(2) Calculate R0. The resultant answer will usually be an integer plus a fraction. Assume the next high integer.

(3) Fill in value for R1.

(4) Calculate R3. Use the next highest integer. In these examples the sync width was chosen to be one third of the horizontal blanking interval.

(5) Calculate R2. Again, use the next highest integer.

(6) Calculate t_c, character tie. This is the time required for one scan line of one character block to be written.

(7) Calculate the exact dot frequency.

(8) Calculate t_{sl}, scan line time. This is the time required for one scan line of one character row to be written including retrace time.

(9) Calculate n. This is the total number of scan lines for each frame. Discard any fraction.

(10) Calculate N and R.

(11) Calculate R4.

(12) Fill in R5.

(13) Fill in R6.

(14) Calculate R7. If there is no possible value for R7, then the display demands for the CRT monitor exceed its capability. A compromise adjustment must be made in basic parameter 6, 8, or 10.

(15) Calculate R9.

(16) Calculate t_{cr}. This the time required for one character row to be written.

(17) Calculate t_{hr}. $t_{hr} > B3$

(18) Calculate t_{vr}. $t_{vr} > B4$

In Figure 20, calculation (B) verifies that the vertical period is 16.667 milliseconds or 60 hertz. The expression used is:

$$t_{cr} \times [(R4) + 1] + [t_{sl} \times (R5)] = V_P$$

Another check is calculation of horizontal sync pulse width R3. $t_c = PW_{HS}$ (typically approximately equals 4 microseconds).

For convenience, a blank worksheet is provided in Figure 22.

FIGURE 20 – CRTC WORKSHEET EXAMPLE CALCULATION (32 x 16)

Basic Parameters (B1-B10)		Symbol	Intermediate Calculations	Value	Register	Register Calculations	Decimal	Hex
1. Horizontal frequency	= 15750 ± 500	(1) f	$\frac{32 \times (5+2)}{1 - 11 \times 10^{-6}}{15750}$	= 4.27×10^6	(2) R0	$\frac{4.27 \times 10^6}{15,750 \times (5+2)}$ - 1 =	38	26
2. Vertical frequency	= 60	(6) t_c	$\frac{1}{39 \times 15750}$	= 1.63×10^{-6}	(3) R1	R5 = 32	32	20
3. Minimum Horizontal retrace time	= 11×10^{-6}	(7) f	$\frac{5+2}{1.63 \times 10^{-6}}$	= 4.29×10^6	(5) R2	$32 + \frac{3}{2}$	33	21
4. Minimum vertical retrace time	= 10^{-3}	(8) t_{sl}	$39 \times 1.63 \times 10^{-6}$	= 63.6×10^{-6}	(4) R3	$38 - \frac{12}{3}$ =	2	2
5. # of displayed characters per row	= 32	(9) n	$\frac{1}{60 \times 63.6 \times 10^{-6}}$	= 262	(11) R4	17 - 1 =	16	10
6. # of displayed character rows	= 16	(10) N		17	(12) R5	R = 7	7	7
7. # of dots in character dot matrix row	= 5	R	$\frac{262}{7+8}$	= 7	(13) R6	R6 = 16	16	10
8. # of displayed character rows					(14) R7	(A)	16	10
9. # of scan lines in character • matrix column	= 7	(16) t_{cr}	$17 \times 8 \times 63.6 \times 10^{-6}$	= 954×10^{-6}	(15) R9	7 + 8 - 1 =	14	0E
					R10			
9. Number of dots between horizontal adjacents	= 2	(17) t_{hr}	$\frac{(38+1-32)\times(5+2)}{4.29 \times 10^6}$	= 11.42×10^{-6}	R11			
10. Number of scan lines between vertical adjacents	= 8	(18) t_{vr}	$\frac{15750}{60} - 16(7+8) \times 63.6 \times 10^{-6}$	= 1.431×10^{-3}	R12			
					R13			
(A) $16 + 1 - \frac{16-7}{7+8} \geq (R7) \geq 16$					R14			
$17 - 6 \geq (R7) \geq 16$			(B) $954 \times 10^{-6} \times 17 = 16.218 \times 10^{-3}$					
			$+ \frac{63.6 \times 10^{-6} \times 7 = 445 \times 10^{-3}}{16.663 \text{ ms}} = \frac{1}{f}$		R15			
			$f = 60$ Hz					

FIGURE 21 — CRTC WORKSHEET EXAMPLE CALCULATION (80 x 24)

Basic Parameters (B1-B10)		Symbol	Intermediate Calculations	Value	Register	Register Calculations	Decimal	Hex
1. Horizontal frequency	= 18,600	① t'	$\frac{80+(7+2)}{\frac{1}{18600}-11\times10^{-6}}$	$16+838\times10^{6}$	② R0	$\frac{16.838\times10^{6}-1}{(18,600)(9)}$	100	64
					③ R1	B5 = 80	80	50
2. Vertical frequency	= 60	⑥ t_c	$\frac{1}{(100+1)+18600}$	$\frac{532.31\times10^{-9}}{}$	⑤ R2	$80+\frac{7}{2}$	84	54
3. Minimum Horizontal retrace time	= 11×10^{-6}	⑦ t	$\frac{7+2}{532.31\times10^{-9}}$	16.907×10^{6}	④ R3	$\frac{R0-R1}{3}$	7	07
4. Minimum vertical retrace time	= 1×10^{-3}	⑧ t_{sl}	$(100+1)(532.31\times10^{-9})$	53.76×10^{-6}	⑪ R4	28 − 1	27	1B
5. # of displayed characters per row	= 80	⑨ n	$\frac{1}{(60)(53.76\times10^{-6})}$	310	⑫ R5	R = 2	02	02
					⑬ R6	B6 = 24	24	18
6. # of displayed charactor rows	= 24	⑩ N		28	⑭ R7	Ⓐ	25	19
7. # of dots in character dot matrix row	= 7	R	$\frac{310}{11}$	2	R8		0	0
					⑮ R9	(9 + 2) − 1	10	0A
8. # of scan lines in charactor + matrix column	= 9	⑯ t_{cr}	$(9+2)(53.76\times10^{-6})$	591.39×10^{-6}	R10		00	00
9. Number of dots between horizontal adjacents	= 2	⑰ t_{hr}	$\leq \frac{(101-80)(7+2)}{16.907\times10^{6}}$	11.17×10^{-6}	R11		11	0B
					R12			00
10. Number of scan lines between vertical adjacents	= 2	⑱ t_{vr}	$\leq \left[\frac{18600}{60}-24(11)\right]53.76\times10^{-6}$	2.47×10^{-3}	R13		128	80
					R14		128	00
					R15			80

Ⓐ $(27+1)-\frac{(16-2)}{11} \geq R7 \geq 24$

$26.72 \geq R7 \geq 24$

Ⓑ B2 = $1/[(t_{cr})(R4+1)+(t_{sl})(R5)]$

= $1/[(591.39\times10^{6})(28)+(53.76\times10^{-6})(2)]$

= $1/16.667\times10^{-3}$

= 60

FIGURE 22 — CRTC WORKSHEET

Basic Parameters		Intermediate Calculations		Register Calculations			
		Symbol	Value	Register	Decimal	Hex
1. Horizontal frequency	= _____	f	_____	R0	_____	_____
2. Vertical frequency	= _____	t_c	_____	R1	_____	_____
3. Minimum Horizontal retrace time	= _____	t	_____	R2	_____	_____
				R3	_____	_____
4 Minimum vertical retrace time	= _____	t_{sl}	_____	R4	_____	_____
				R5	_____	_____
5. # of displayed characters per row	= _____	n	_____	R6	_____	_____
				R7	_____	_____
6. # of displayed charactor rows	= _____	N	_____	R8	_____	_____
				R9	_____	_____
7. # of dots in character dot matrix row	= _____	R	_____	R10	_____	_____
				R11	_____	_____
8. # of scan lines in charactor • matrix column	= _____	t_{cr}	_____	R12	_____	_____
				R13	_____	_____
9. Number of dots between horizontal adjacents	= _____	t_{hr}	_____	R14	_____	_____
				R15	_____	_____
10. Number of scan lines between vertical adjacents	= _____	t_{vr}	_____	R16	_____	_____
				R17	_____	_____
				R18	_____	_____
				R19	_____	_____

CRTC INITIALIZATION

Register R0-R15 must be initialized after the system is powered up. The processor will normally load the CRTC register file from a firmware table. The program required to initialize the CRTC for a 80 x 24 format (example calculation #2) is shown in Figure 23.

The CRTC registers will have an initial value at power up. When using a direct drive monitor (sans horizontal oscillator) these initial values may result in out-of-tolerance operation. CRTC programming should be done immediately after power up especially in this type of system.

ADDITIONAL CRTC APPLICATIONS

The foremost system function which may be performed by the CRTC controller is the refreshing of dynamic RAM. This is quite simple as the refresh addresses continually run.

Note that the LPSTB input may be used to support additional system functions other than a light pen. A digital to analog converter (DAC) and comparator could be configured to use the refresh addresses as a reference to a DAC composed of a resistive adder network connected to a comparator. The output of the comparator would generate the LPSTB input signifying a match between the refresh address analog level and the unknown voltage.

The light-pen strobe input could also be used as a character strobe to allow the CRTC refresh addresses to decode a keyboard matrix. Debouncing would need to be done in software.

Both the VS and HS outputs may be used as a real time clock. Once programmed, the CRTC will provide a stable reference frequency.

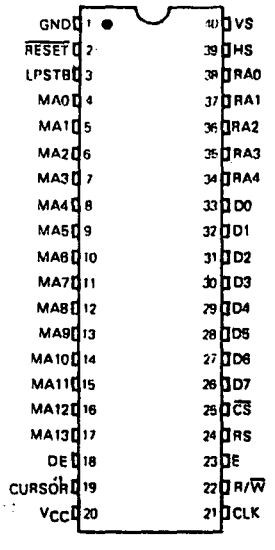
FIGURE 23 — MC6800 PROGRAM FOR CRTC INITIALIZATION

```
PAGE 001  CRTCINIT.SA:0   MC6845 CRTC Initialization Program

00001                          NAM    MC6845
00002                          TTL    / MC6845-1 CRTC initialization program
00003                          OPT    G,S,LLE=85 print FCB's, FDB's & XREF table
00004                    ********************************************************
00005                    * Assign CRTC addresses
00006                    *
00007          9000  A CRTCAD EQU    $9000     Address Register
00008          9001  A CRTCRG EQU    CRTCAD+1  Data Register
00009                    ********************************************************
00010                    * Initialization program
00011                    *
00012A 0000                    ORG    0         a place to start
00013A 0000 5F                 CLRB             clear counter
00014A 0001 CE 1020  A         LDX    #CRTTAB   table pointer
00015A 0004 F7 9000  A CRTC1   STAB   CRTCAD    load address register
00016A 0007 A6 00    A         LDAA   0,X       get register value from table
00017A 0009 B7 9001  A         STAA   CRTCRG    program register
00018A 000C 08                 INX              increment counters
00019A 000D 5C                 INCB
00020A 000E C1 10    A         CMPB   $10       finished?
00021A 0010 26 F2 0004         BNE    CRTC1     no: take branch
00022A 0012 3F                 SWI              yes: call monitor
00023                    ********************************************************
00024                    * CRTC register initialization table
00025                    * 80 x 24 non-interlaced format
00026A 1020                    ORG    $1020     start of table
00027A 1020    65    A CRTTAB FCB    $64,$50   R0, R1 - H total & H displayed
     A 1021    50    A
00028A 1022    56    A         FCB    $54,$07   R2, R3 - HS pos. & HS width
     A 1023    09    A
00029A 1024    18    A         FCB    $1B,$02   R4, R5 - V total & V total adj.
     A 1025    0A    A
00030A 1026    18    A         FCB    $18,$19   R6, R7 - V displayed $ VS pos.
     A 1027    18    A
00031A 1028    00    A         FCB    $00,$0A   R8, R9 - Interlace & Max scan line
     A 1029    0B    A
00032A 102A    00    A         FCB    $00,$0B   R10,R11- Cursor start & end
     A 102B    0B    A
00033A 102C    0080  A         FDB    $0080     R12,R13- Start Address
00034A 102E    0080  A         FDB    $0080     R14,R15- Cursor Address
00035                          END
TOTAL ERRORS 00000--00000

CRTC1   0004   CRTCAD 9000   CRTCRG 9001   CRTTAB 1020
```

ORDERING INFORMATION

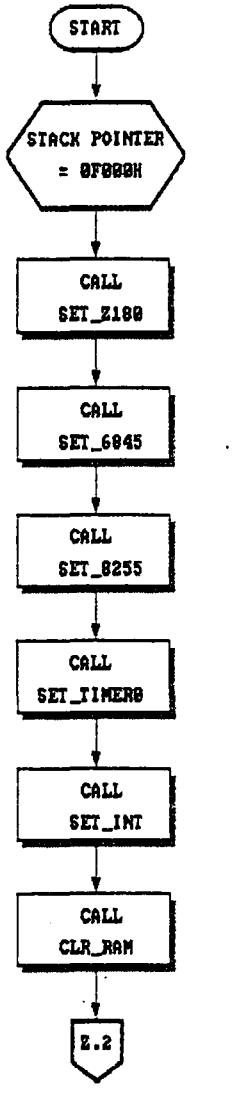
Package Type	Frequency (MHz)	Temperature	Order Number
Cerdip	1.0	0°C to 70°C	MC6845S
S Suffix	1.0	-40°C to +85°C	MC6845CS
	1.5	0°C to 70°C	MC68A45S
	1.5	-40°C to +85°C	MC68A45CS
	2.0	0°C to 70°C	MC68B45S
Plastic	1.0	0°C to 70°C	MC6845P
P Suffix	1.0	-40°C to +85°C	MC6845CP
	1.5	0°C to 70°C	MC68A45P
	1.5	-40°C to +85°C	MC68A45CP
	2.0	0°C to 70°C	MC68B45P

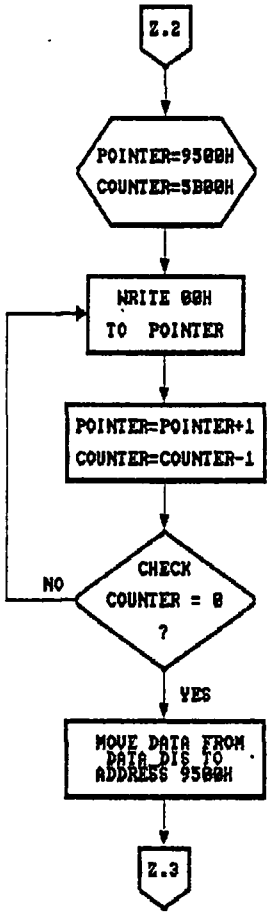
PIN ASSIGNMENT



ภาคผนวก ข

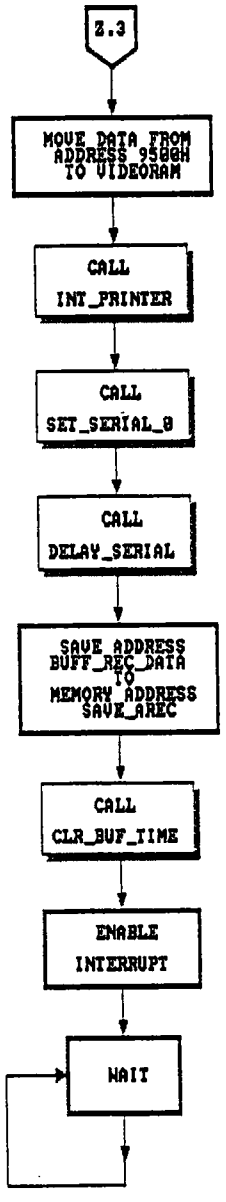
MAIN PROGRAM





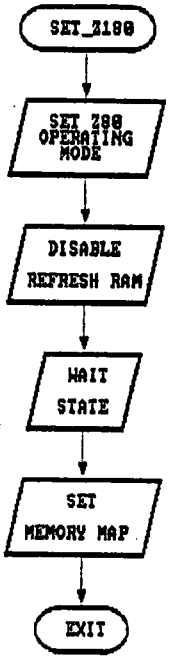
MAIN PROGRAM

(CONTINUE)



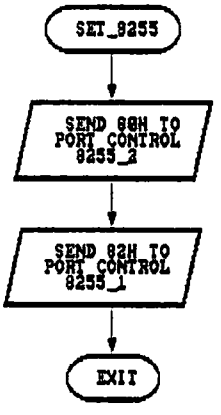
SUB ROUTINE

SET_Z180



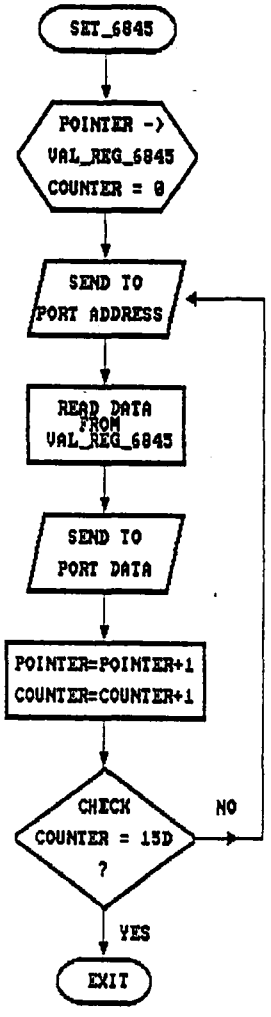
SUB ROUTINE

SET 8255



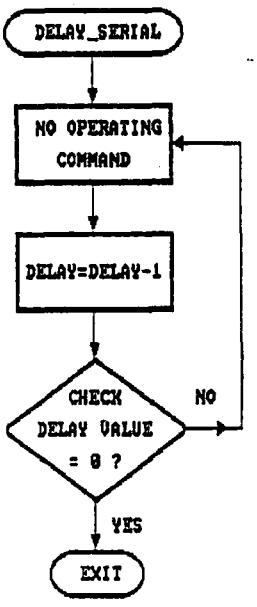
SUB ROUTINE

SET_6845



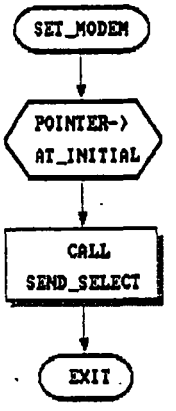
SUB ROUTINE

DELAY SERIAL



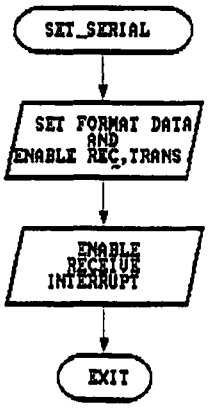
SUB ROUTINE

SET MODEM



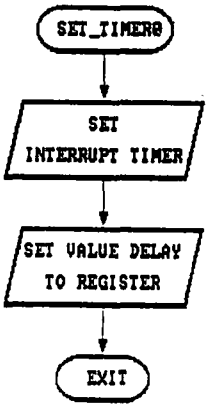
SUB ROUTINE

SET_SERIAL_PORT



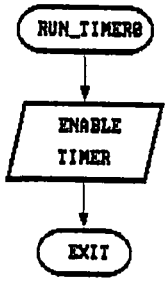
SUB ROUTINE

SET_TIMER0



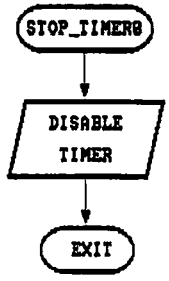
SUB ROUTINE

RUN_TIMER0



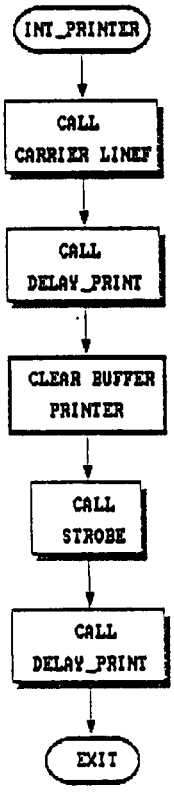
SUB ROUTINE

STOP_TIMER0



SUB ROUTINE

INITIAL PRINTER

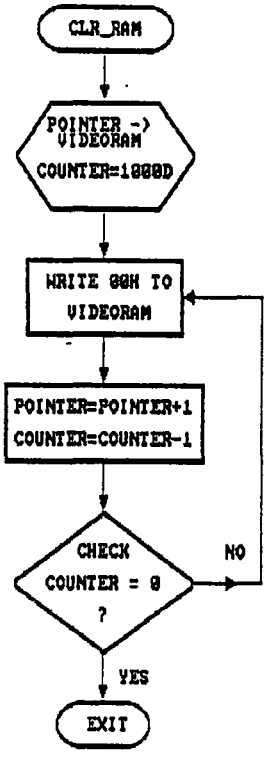


SUB ROUTINE

SET_INTERRUPT

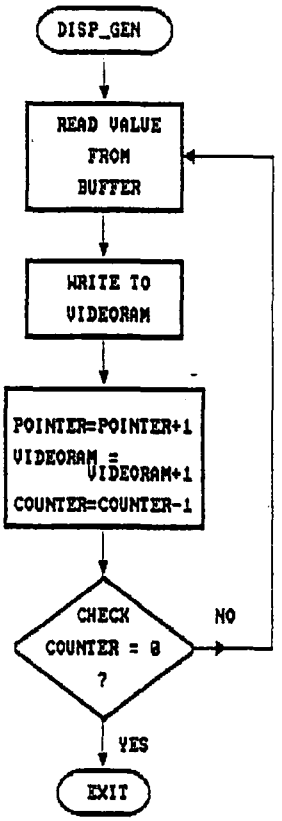
SUB ROUTINE

CLEAR VIDEORAM



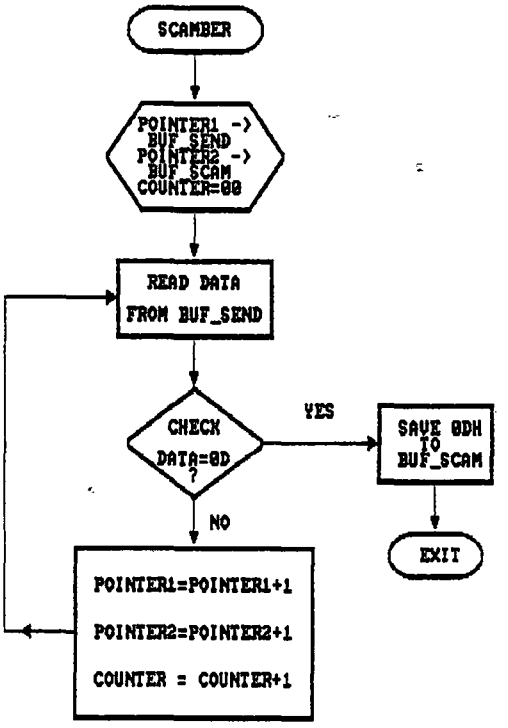
SUB ROUTINE

DISPLAY DATA



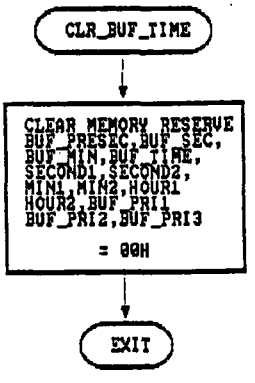
SUB ROUTINE

SCAMBER



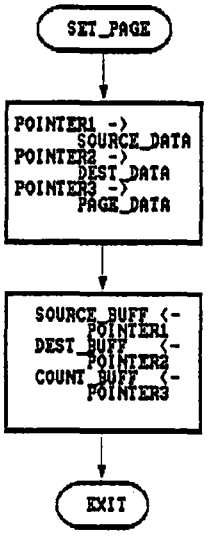
SUB ROUTINE

CLEAR BUFFER TIMER



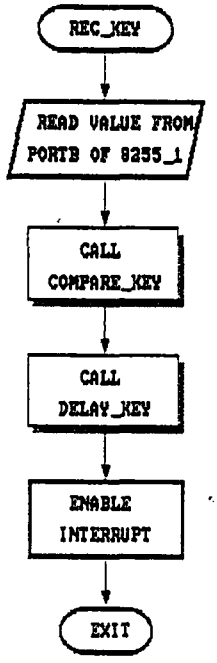
SUB ROUTINE

SET PAGE



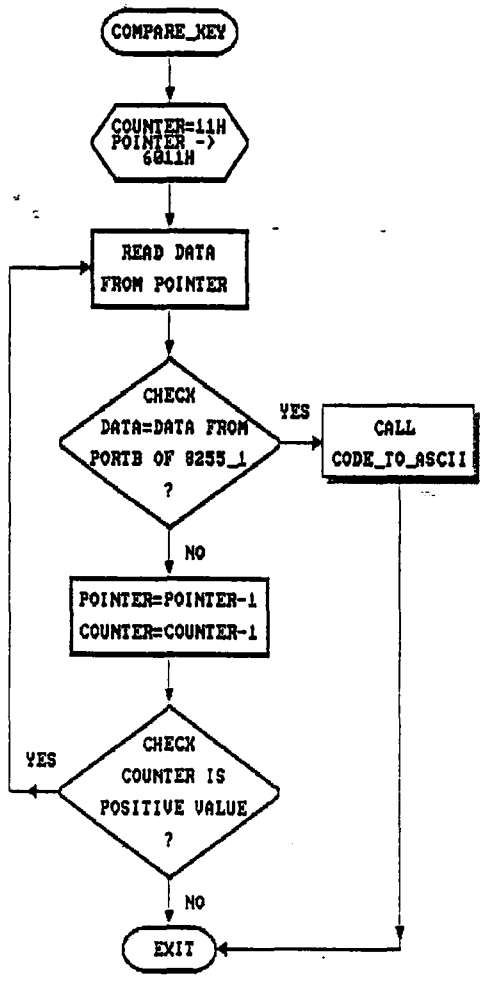
SUB ROUTINE

RECEIVE_KEY



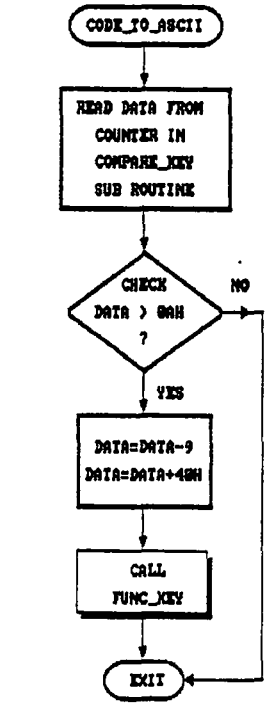
SUB ROUTINE

COMPARE_KEY



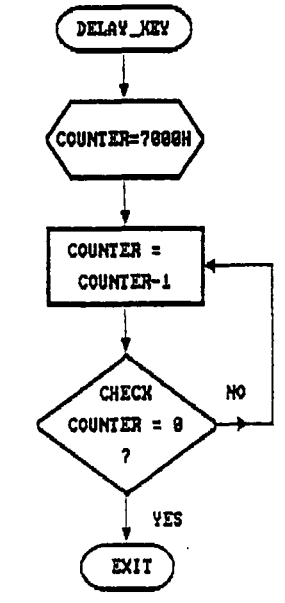
SUB ROUTINE

CODE KEY TO ASCII



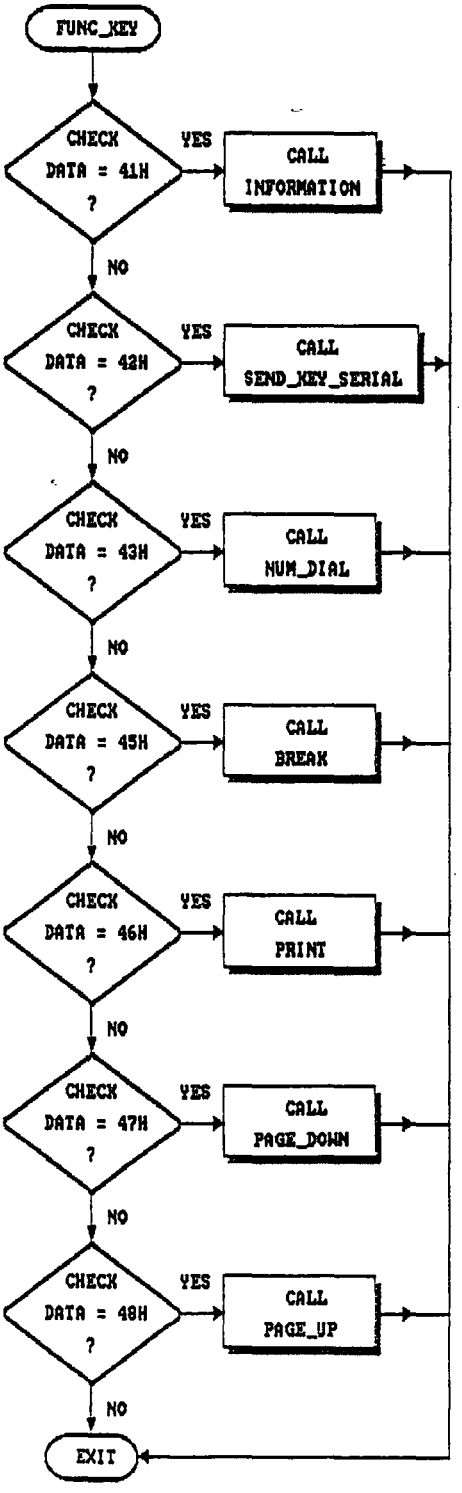
SUB ROUTINE

DELAY_KEY



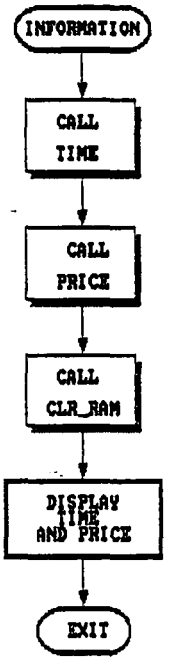
SUB ROUTINE

FUNCTION_KEY



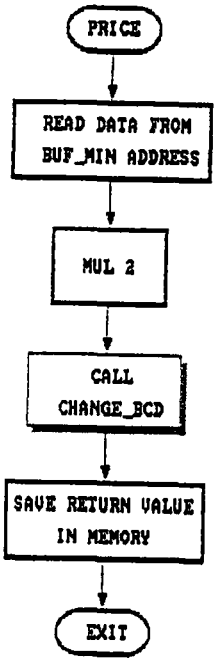
SUB ROUTINE

INFORMATION



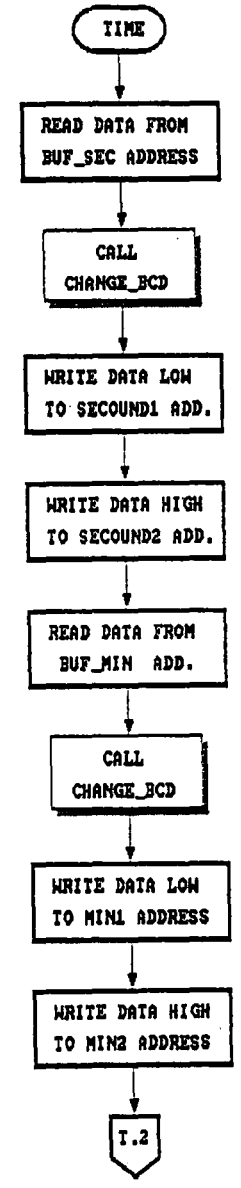
SUB ROUTINE

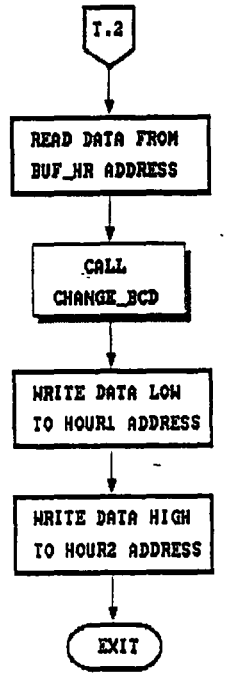
PRICE



SUB ROUTINE

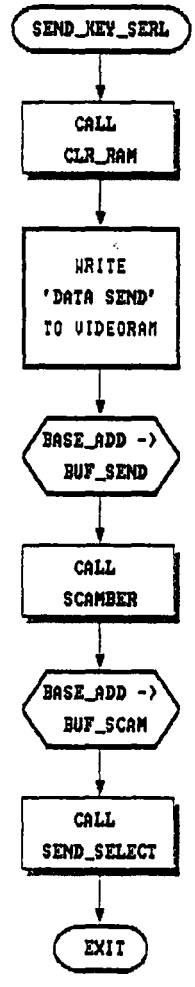
TIMER

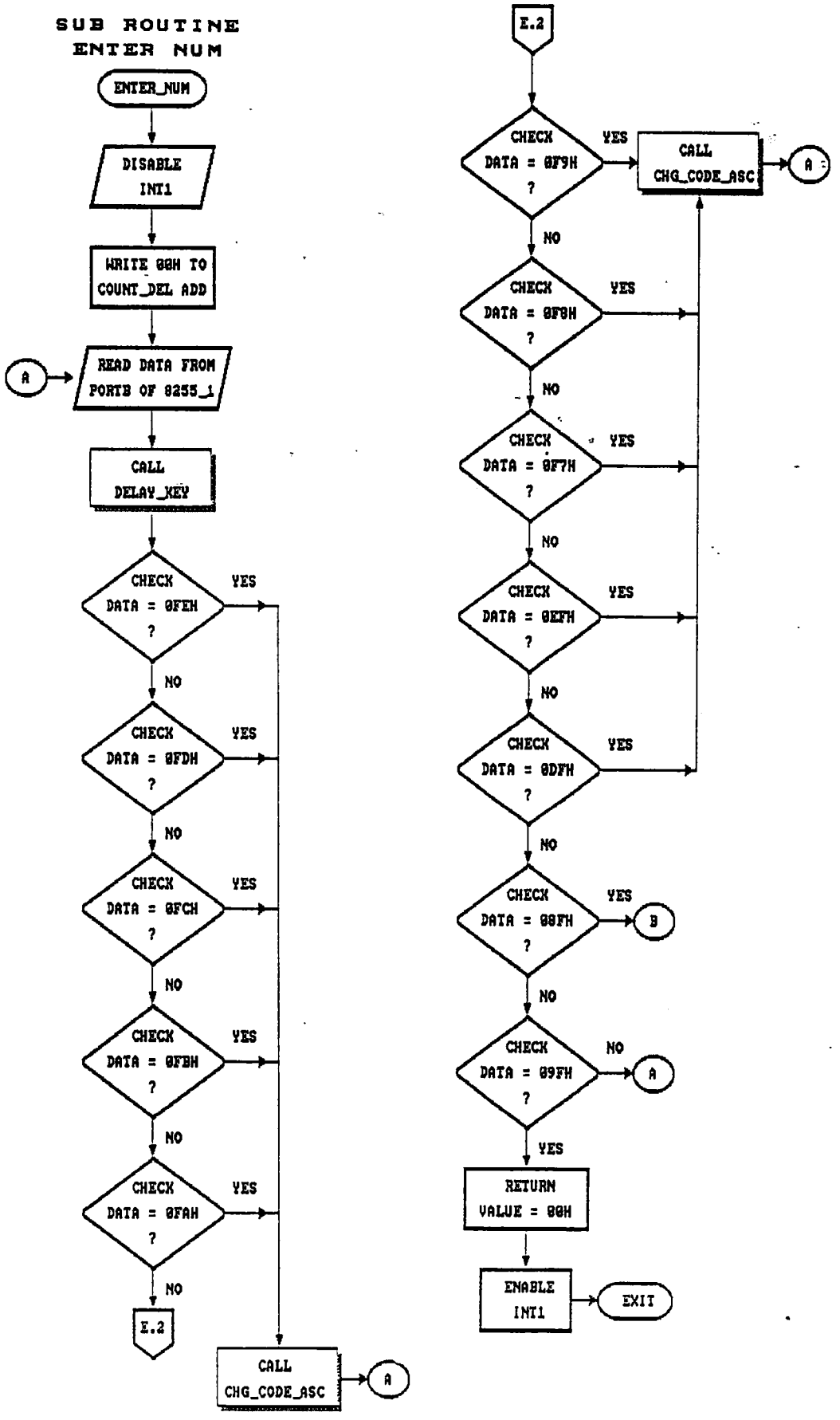


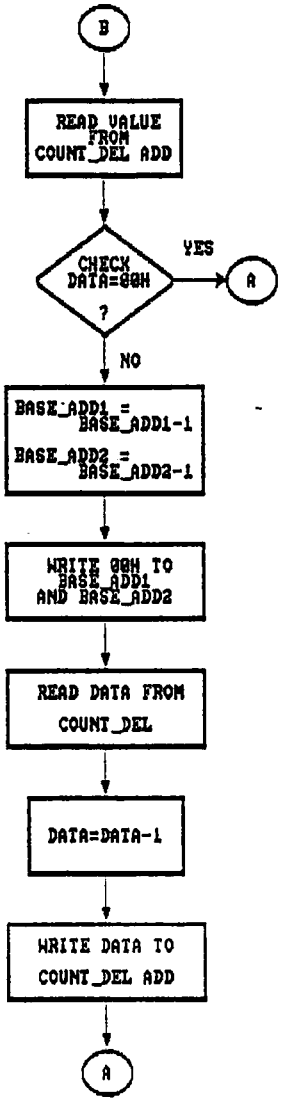


SUB ROUTINE

RECEIVE DATA FROM KEY AND SEND VIA SERIAL

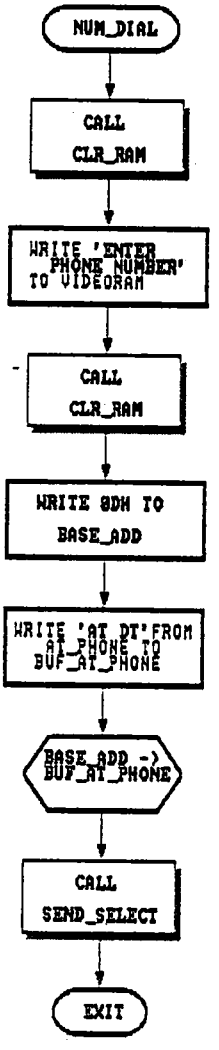



SUB ROUTINE
ENTER NUM
ENTER_NUM
DISABLE
INT1
WRITE 00H TO
COUNT_DEL ADD
A
READ DATA FROM
PORTB OF 8255_1
CALL
DELAY_KEY
CHECK
DATA = 0FEH
?
YES
NO
CHECK
DATA = 0FDH
?
YES
NO
CHECK
DATA = 0FCH
?
YES
NO
CHECK
DATA = 0FBH
?
YES
NO
CHECK
DATA = 0FAH
?
YES
NO
E.2
CALL
CHG_CODE_ASC
A
E.2
CHECK
DATA = 0F9H
?
YES
CALL
CHG_CODE_ASC
A
NO
CHECK
DATA = 0F8H
?
YES
NO
CHECK
DATA = 0F7H
?
YES
NO
CHECK
DATA = 0EFH
?
YES
NO
CHECK
DATA = 0DFH
?
YES
NO
CHECK
DATA = 09FH
?
YES
B
NO
CHECK
DATA = 09FH
?
NO
A
YES
RETURN
VALUE = 00H
ENABLE
INT1
EXIT

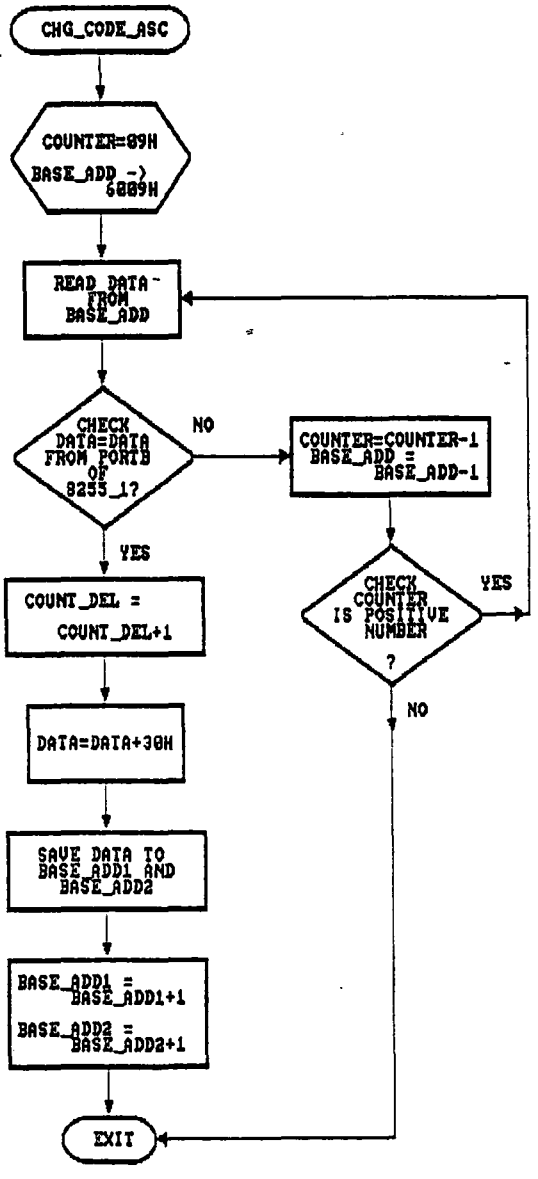


SUB ROUTINE

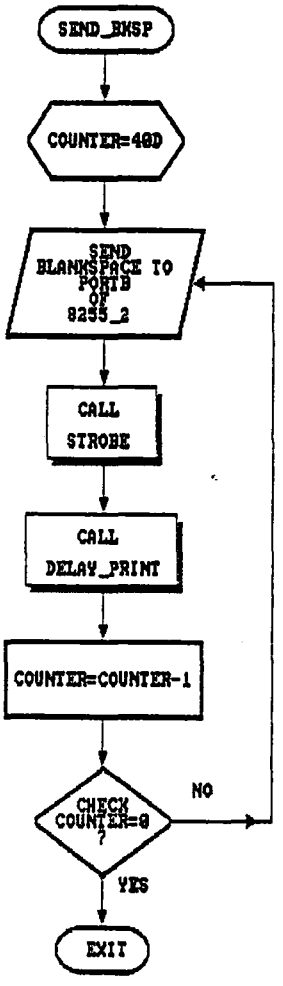
NUMBER DIAL



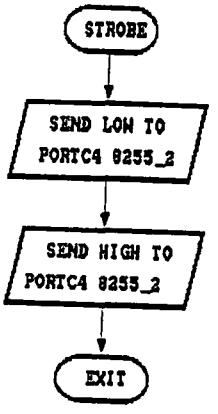
SUB ROUTINE
CHANGE CODE
TO ASCII



SUB ROUTINE
SEND_BLANK

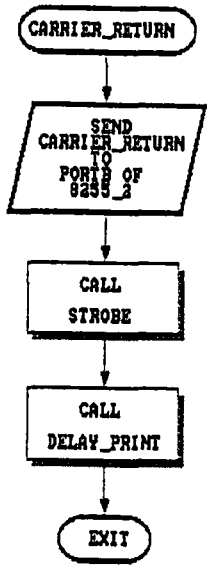


SUB ROUTINE
STROBE



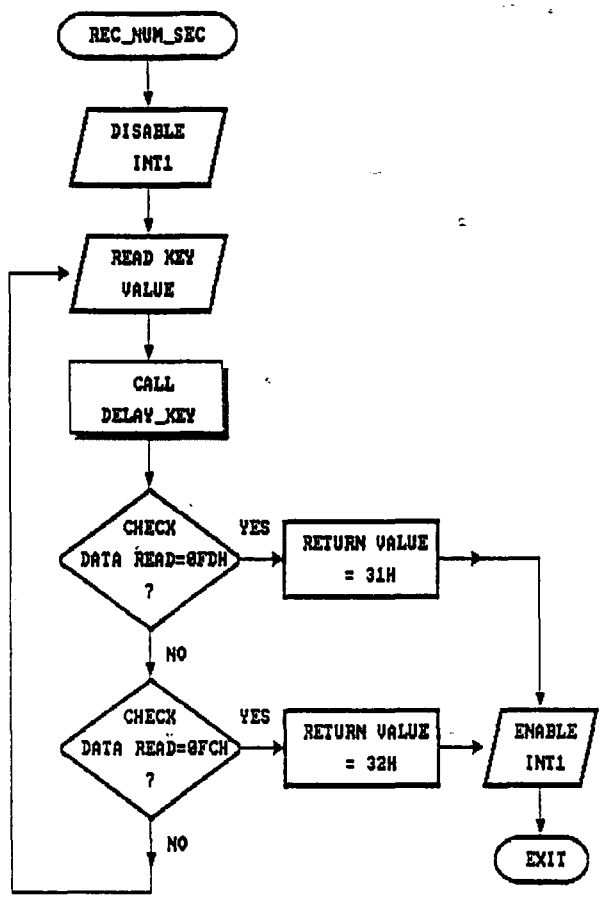
SUB ROUTINE

CARRIER_LINEFEED



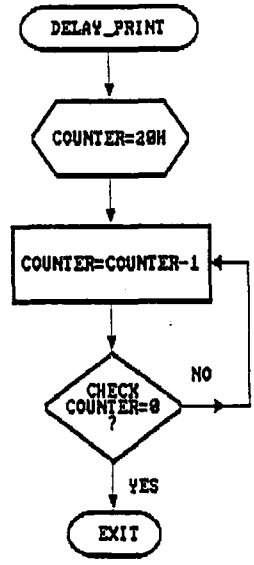
SUB ROUTINE

REC_NUM_SEC



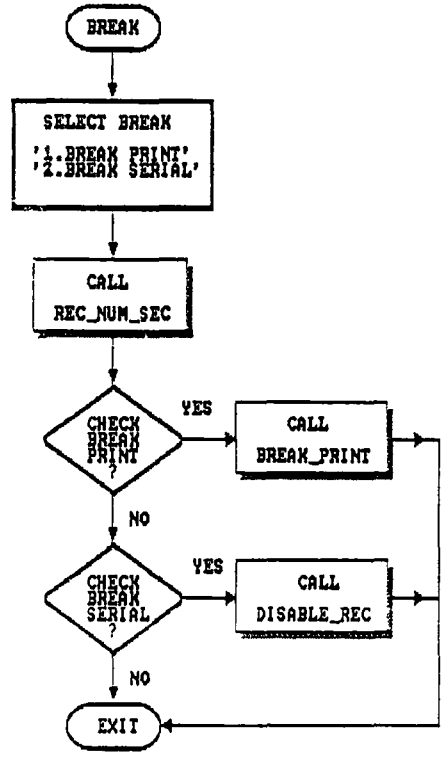
SUB ROUTINE

DELAY_PRINT



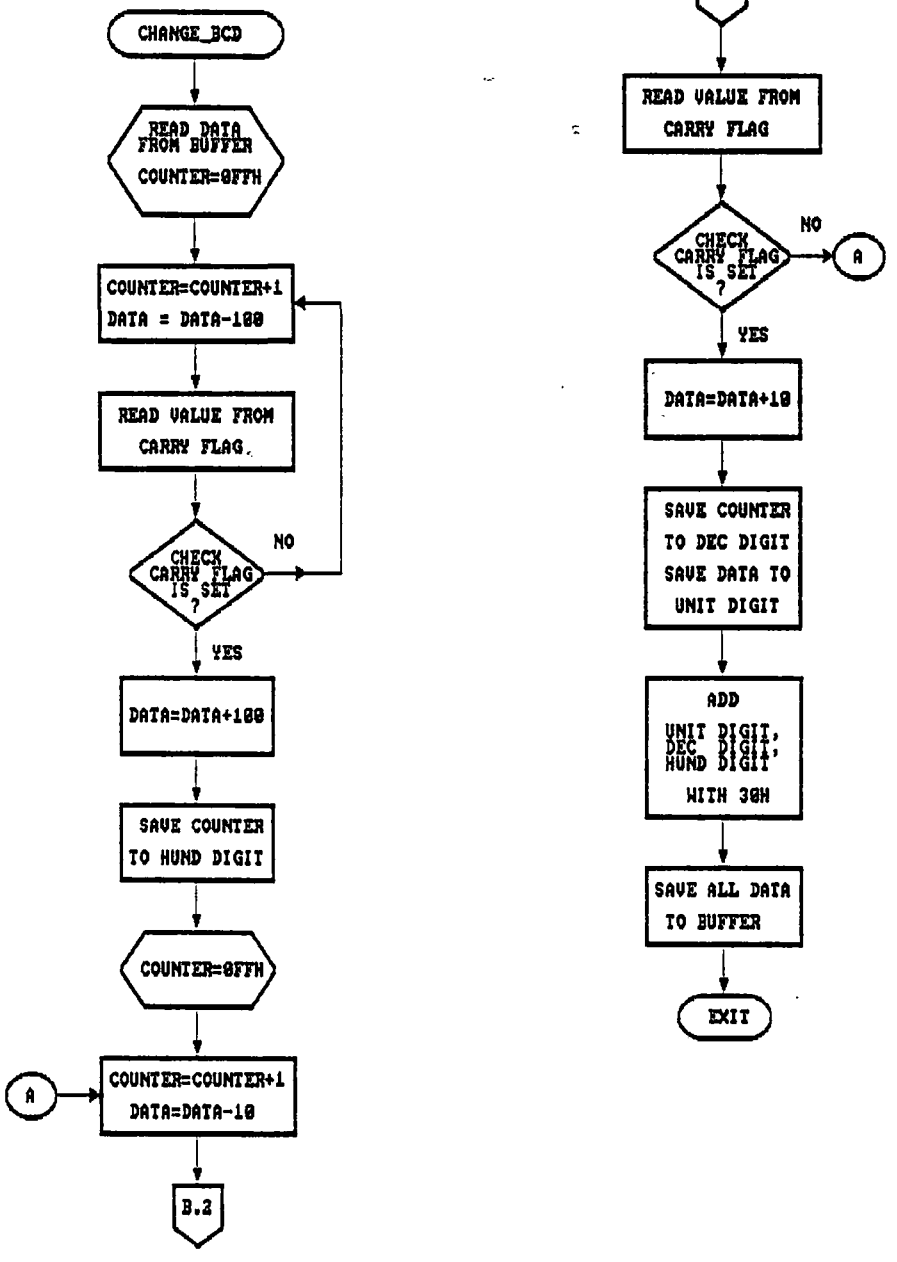
SUB ROUTINE

BREAK



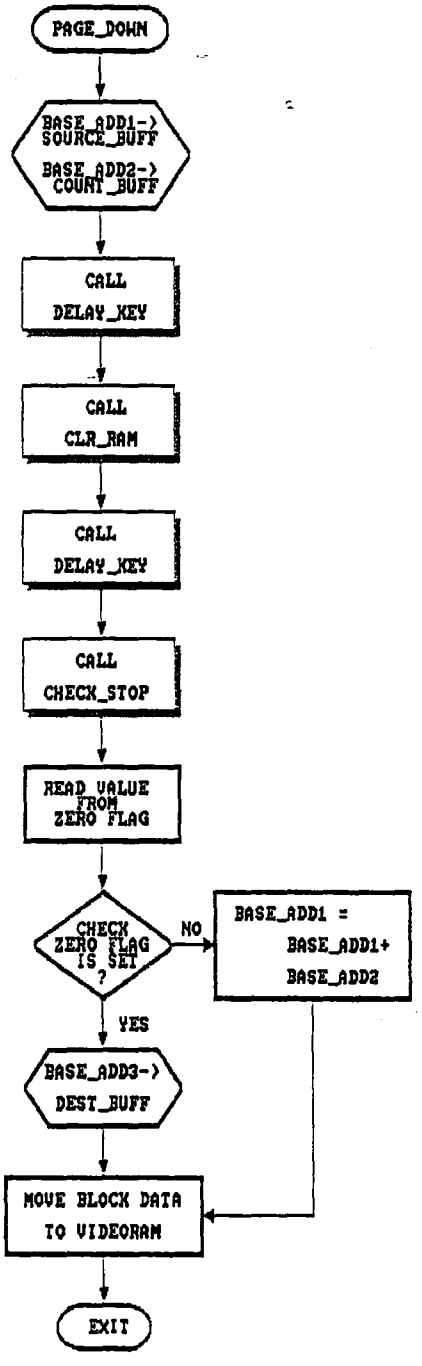
SUB ROUTINE

CHANGE HEX TO BCD



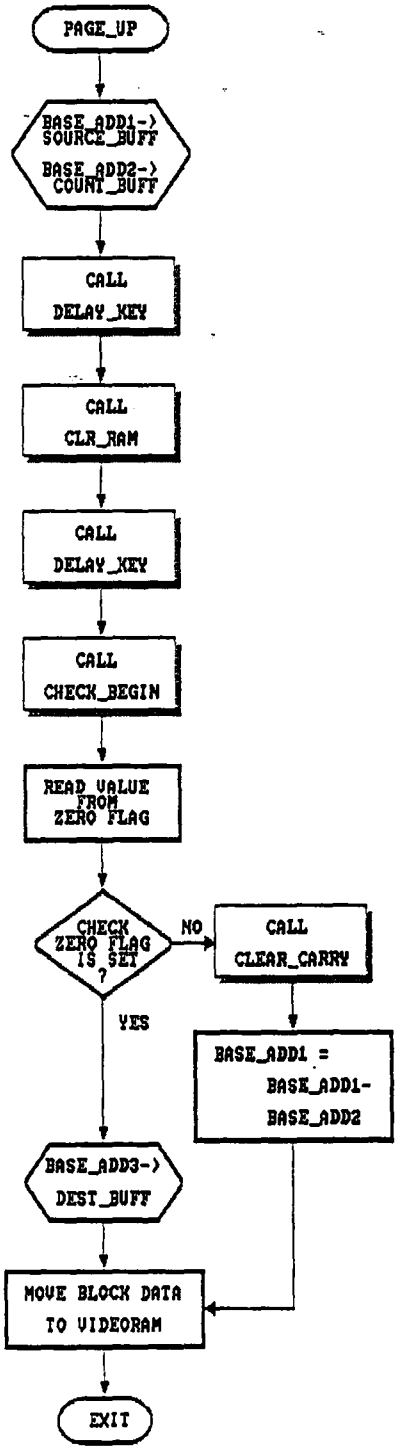
SUB ROUTINE

PAGE_DOWN



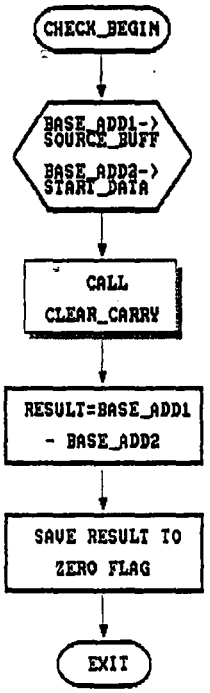
SUB ROUTINE

PAGE_UP



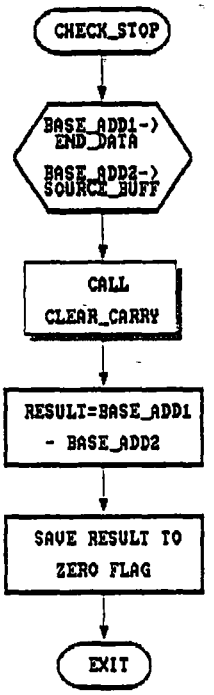
SUB ROUTINE

CHECK_BEGIN



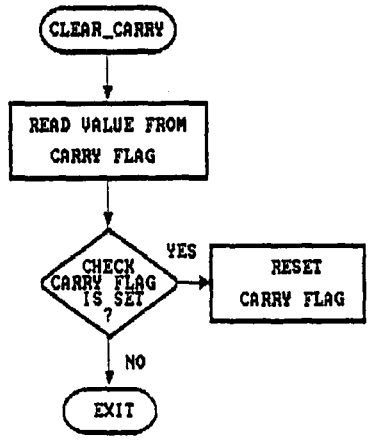
SUB ROUTINE

CHECK_STOP



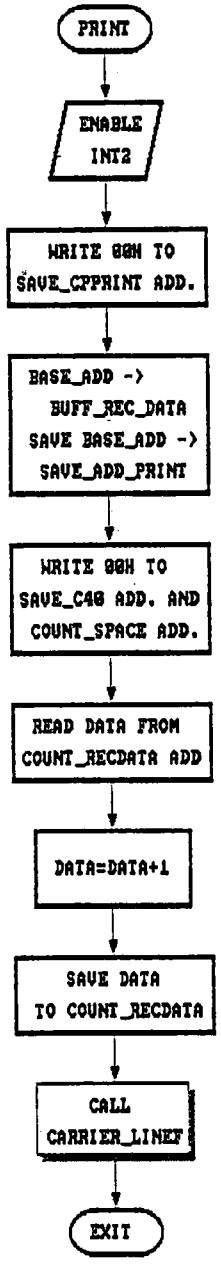
SUB ROUTINE

CLEAR_CARRY



SUB ROUTINE

PRINT



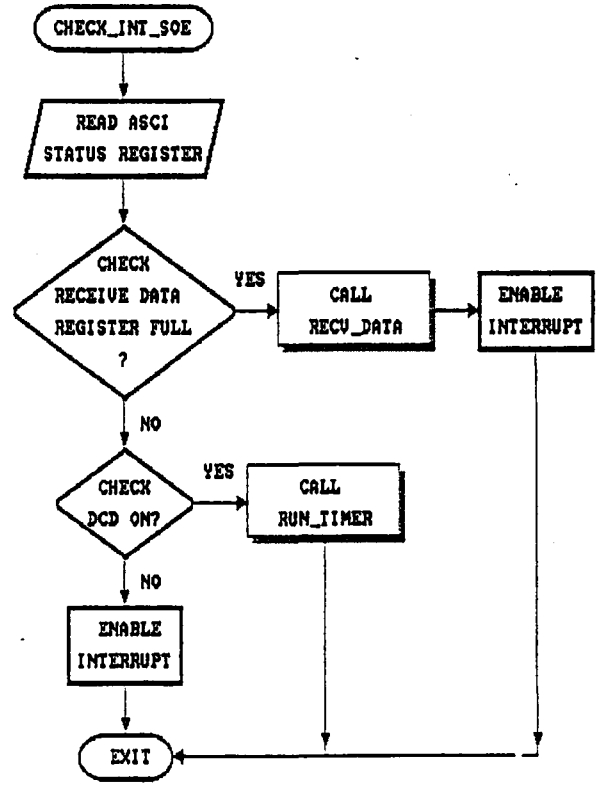
SUB ROUTINE

BREAK PRINT

SUB ROUTINE

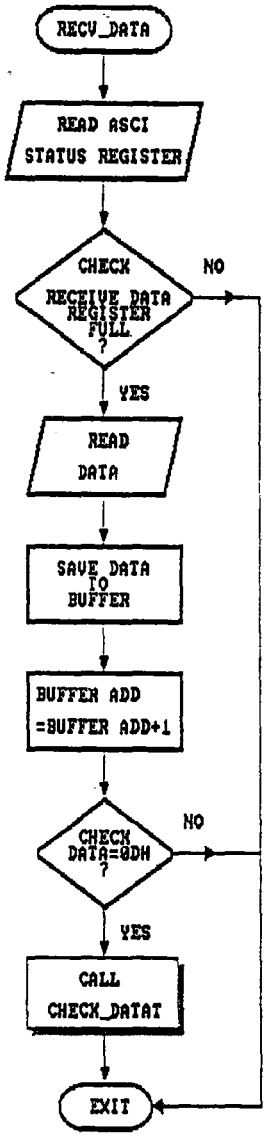
CHECK INTERRUPT

SOURCE



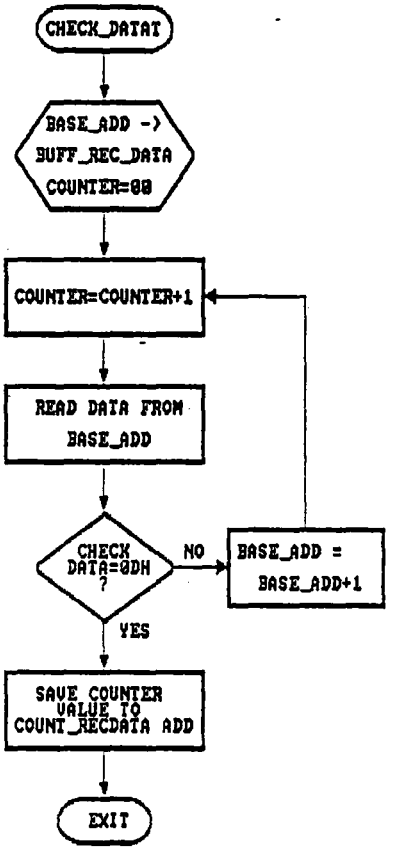
SUB ROUTINE

RECEIVE DATA



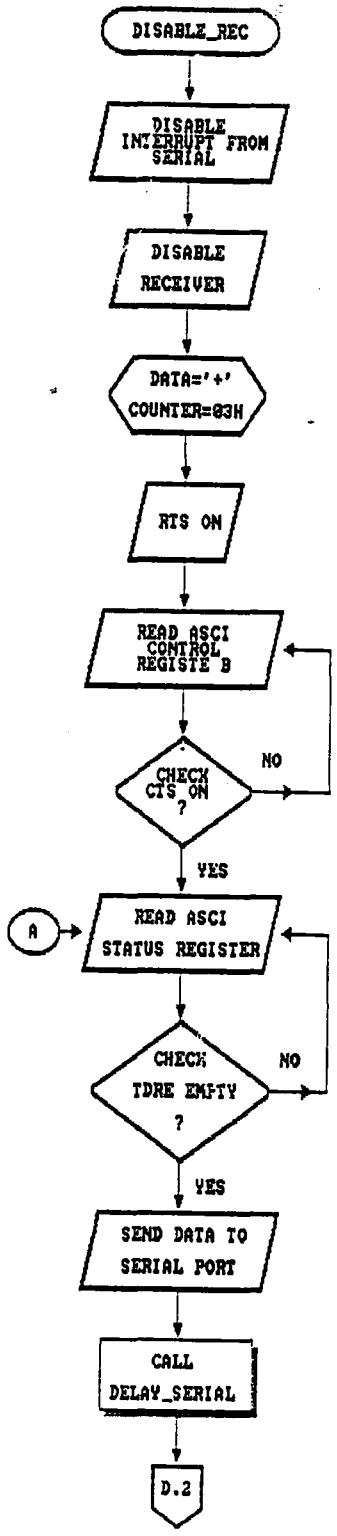
SUB ROUTINE

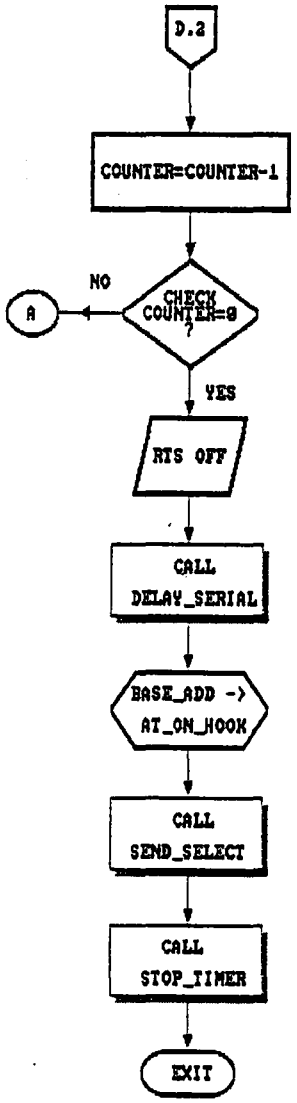
CHECK TOTAL DATA



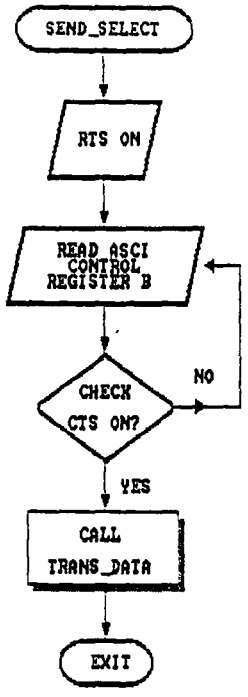
SUB ROUTINE

DISABLE_REC

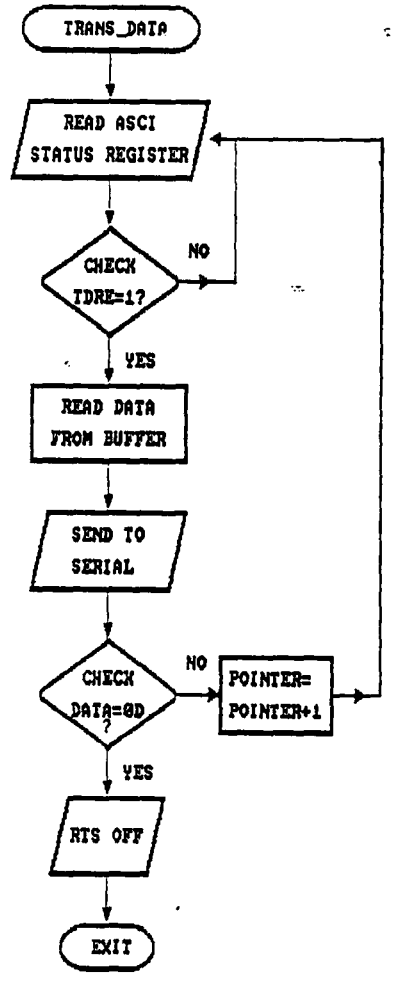




SUB ROUTINE
SEND_SELECT

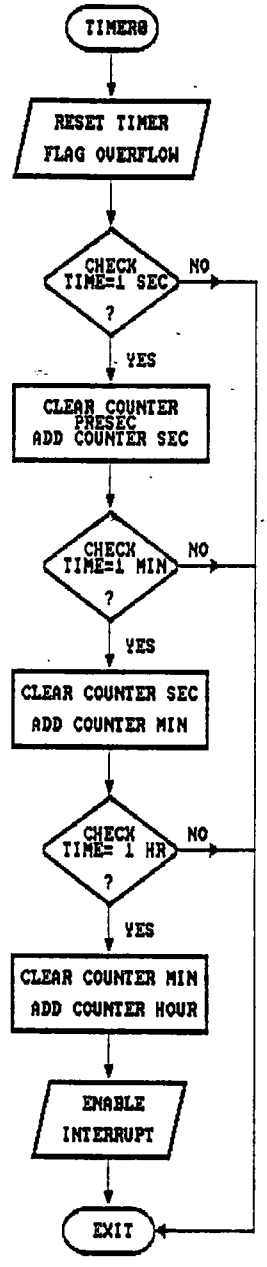


SUB ROUTINE
TRANSMIT DATA



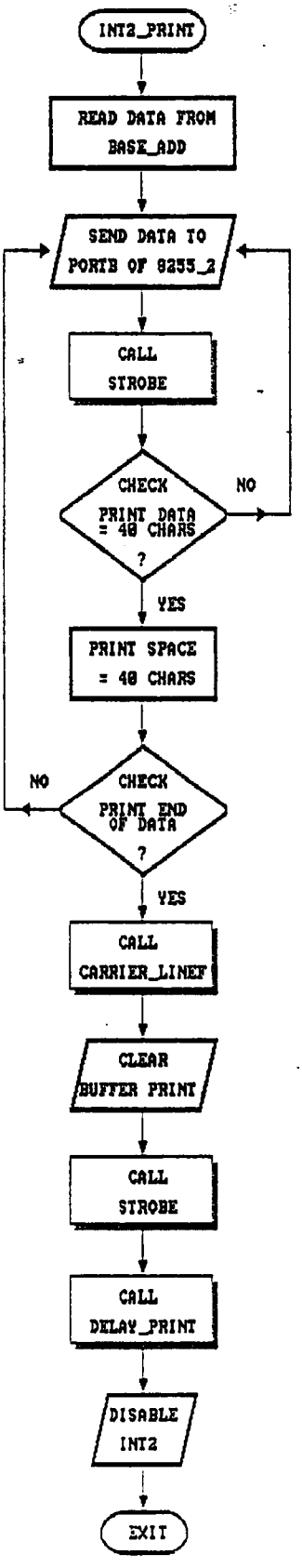
SUB ROUTINE

TIMER



SUB ROUTINE

INT2_PRINT



```
;*******************************************************************************
;*******************************************************************************
;******************   PROGRAM MONITOR INTERRACTIVE VIEWDATA   ******************
;******************       FOR Z80180 MPU HOME CENTER          ******************
;*******************************************************************************
;*******************************************************************************

            .ORG  0000H

;*******************************************************************************
;**********************     CONSTANT  VALUE OF VARIABLE     **********************
;*******************************************************************************

            .EQU  OMCR,3EH          ;OPERATION MODE CONTROL REGISTER
            .EQU  CBAR,3AH          ;MMU COMMON/BANK AREA REGISTER
            .EQU  CBR,38H           ;MMU COMMON BASE REGISTER
            .EQU  BBR,39H           ;MMU BANK BASE REGISTER
            .EQU  IL,33H            ;INTERRUPT VECTOR LOW REGISTER
            .EQU  CNTLA0,00H        ;ASCI CONTROL REGISTER A CH-0
            .EQU  CNTLB0,02H        ;ASCI CONTROL REGISTER B CH-0
            .EQU  STAT0,04H         ;ASCI STATUS REGISTER CH-0
            .EQU  TDR0,06H          ;ASCI TRANSMIT DATA REGISTER CH-0
            .EQU  RDR0,08H          ;ASCI RECEIVE DATA REGISTER CH-0
            .EQU  RCR,36H           ;REFRESH CONTROL REGISTER
            .EQU  ITC,34H           ;INT/TRAP CONTROL REGISTER
            .EQU  DCNTL,32H         ;DMA/WAIT CONTROL REGISTER
            .EQU  TMDR0L,0CH        ;TIMER DATA REGISTER LOW CH0
            .EQU  TMDR0H,0DH        ;TIMER DATA REGISTER HIGH CH0
            .EQU  RLDR0L,0EH        ;RELOAD REGISTER LOW CH0
            .EQU  RLDR0H,0FH        ;RELOAD REGISTER HIGH CH0
            .EQU  TCR,10H           ;TIMER CONTROL REGISTER
```

```
.EQU  LF,0AH              ;LINE FEED
.EQU  CR,0DH              ;CARRIER RETURN
.EQU  PADD_6845,0D0H      ;PORT ADDRESS 6845
.EQU  PDAT_6845,0D1H      ;PORT DATA 6845
.EQU  PORTCN1_8255,8FH
.EQU  PORTCN2_8255,9FH
.EQU  PORTB2_8255,9DH
.EQU  PORTC2_8255,9EH
.EQU  BUFF_REC_DATA,9500h ;BUFFER SAVE RECEIVE DATA FROM ASCI
.EQU  VIDEO_RAM,0D800H    ;VIDEO RAM DISPLAY DATA
.EQU  GENERAL_DISPLAY,0DB74H
.EQU  DISP_TIME,0D8A6H
.EQU  DISP_PRICE,0D8F6H
.EQU  DISP_FRONT,0D856H
.EQU  DISP_TEXT,0D8F6H
.EQU  COUNT_RECDATA,8000H ;SAVE COUNT RECEIVE DATA
.EQU  SAVE_AREC,8002H
.EQU  BUF_PRESEC,8004H    ;SAVE COUNT PRESEC
.EQU  BUF_SEC,8005H       ;SAVE COUNT SEC
.EQU  BUF_MIN,8006H       ;SAVE COUNT MIN
.EQU  BUF_HR,8007H        ;SAVE COUNT HOUR
.EQU  HOUR2,8008H
.EQU  HOUR1,8009H
.EQU  MIN2,800AH
.EQU  MIN1,800BH
.EQU  SECOUND2,800CH
.EQU  SECOUND1,800DH
.EQU  BUF_PRI3,800EH
.EQU  BUF_PRI2,800FH
.EQU  BUF_PRI1,8010H
```

```
.EQU  REGIST_ADD,80012H
.EQU  SOURCE_BUFF,8014H
.EQU  SOURCE_2,8016H
.EQU  DEST_BUFF,8018H
.EQU  COUNT_BUFF,801AH
.EQU  HUND,801CH
.EQU  BCD,801DH
.EQU  FRACTION,801EH
.EQU  COUNT_DEL,8020H
.EQU  SAVE_C40,8022H
.EQU  SAVE_CPRINT,8024H
.EQU  SAVE_ADD_PRINT,8026H
.EQU  COUNT_SPACE,8028H
.EQU  SAVE_VIDEO,8030H
.EQU  SOURCE_DATA,9500H
.EQU  DEST_DATA,0D800H
.EQU  PAGE_DATA,1000D
.EQU  END_DATA,0A0B8H
.EQU  START_DATA,9500H
.EQU  BUF_DIALNUM,8086H
.EQU  BUF_AT_PHONE,8080H
.EQU  BUF_SEND,8060H
.EQU  BUF_BREAK,80B0H
.EQU  BUF_SCAN,80D0H
.EQU  CANCEL,18H            ;COMMAND CLEAR BUFFER OF PRINTER
.EQU  BACKSPACE,20H
```

```
;*******************************************************************************
;*************************     MAIN   PROGRAM    *******************************
;*******************************************************************************
            LD      SP,0F000H
            CALL    SET_Z180            ;SET OPERTING MODE
            CALL    SET_6845
            CALL    SET_8255
            CALL    SET_TIMER0
            CALL    SET_INT
            CALL    CLR_RAM
            LD      HL,9500H
            LD      BC,5B00H
CLR_DATA:   LD      A,00H
            LD      (HL),A
            INC     HL
            DEC     BC
            LD      A,C
            OR      B
            JR      NZ,CLR_DATA
            LD      HL,DATA_DIS
            LD      DE,09500H
            LD      BC,1000D
            LDIR
            LD      BC,1000D
            LD      HL,9500H
            LD      DE,0D800H
            LDIR
            CALL    INI_PRINTER
            CALL    SET_SERIAL_0
            LD      BC,01FFH
```

```
                CALL    DELAY_SERIAL
                CALL    SET_MODEM
                LD      BC,0FFFFH
                CALL    DELAY_SERIAL
                CALL    DELAY_SERIAL
                CALL    DELAY_SERIAL
                LD      IY,BUFF_REC_DATA
                LD      (SAVE_AREC),IY
                LD      IX,VIDEO_RAM
                LD      (SAVE_VIDEO),IX
                CALL    SET_PAGE
                CALL    CLR_BUF_TIME
                EI
HEAR:           JR      HEAR
;*******************************************************************************
;******************     SUB ROUTINE SET OPERATING MODE     **********************
;*******************************************************************************
SET_Z180:       LD      A,00H
                OUT0    (OMCR),A            ;Z80 OPERATING MODE
                OUT0    (RCR),A             ;DISABLE REFRESH RAM
                LD      A,10110000B         ;WAIT STATE MEMORY=2 , I/O=4
                OUT0    (DCNTL),A
                LD      C,0F8H              ;BEGIN BANK AREA AT ADDRESS 8000H
                OUT0    (CBAR),C            ;BEGIN COMMON AREA1 AT ADDRESS 0F000H
                LD      C,00H
                OUT0    (CBR),C
                OUT0    (BBR),C             ;PAGE 0 ACTIVE
                RET
```

```
;*****************************************************************************
;**********************      SUB ROUTINE SET 6845CRTC     **********************
;*****************************************************************************
SET_6845:       LD      IX,VAL_REG_6845      ;TABLE OF VALUE REGISTER
                LD      B,00H
LOOP_SET_6845:  OUT0    (PADD_6845),B
                LD      A,(IX+0H)
                OUT0    (PDAT_6845),A
                INC     IX
                INC     B
                LD      A,B
                CP      15D
                JR      NZ,LOOP_SET_6845
                RET
;*****************************************************************************
;**********************       SUB ROUTINE SET 8255       **********************
;*****************************************************************************
SET_8255:       LD      A,80H                ;PORT B OUTPUT , PORT C UPPER OUTPUT
                OUT0    (PORTCN2_8255),A     ;CONNECTER 2
                LD      A,82H                ;PORT B INPUT PORT
                OUT0    (PORTCN1_8255),A     ;CONNECTER 1
                RET
;*****************************************************************************
;**********************     SUB ROUTINE INITIAL PRINTER    **********************
;*****************************************************************************
INI_PRINTER:    CALL    CARRIER_LINEF
                CALL    DELAY_PRINT
                LD      A,CANCEL             ;CLEAR BUFFER OF PRINTER
                OUT0    (PORTB2_8255),A
                CALL    STROBE
```

```
                CALL    DELAY_PRINT
                RET
;*******************************************************************************
;**********************    SUB ROUTINE CLEAR VIDEO RAM    ***********************
;*******************************************************************************
CLR_RAM:        LD      IX,VIDEO_RAM
                LD      BC,1000D
                LD      D,00H
CLRM:           LD      (IX+0),D
                INC     IX
                DEC     BC
                LD      A,B
                OR      C
                JR      NZ,CLRM
                RET
;*******************************************************************************
;**********************     SUB ROUTINE SET SERIAL PORT0    ********************
;*******************************************************************************
SET_SERIAL_0:   LD      C,01110100B         ;ENABLE RECEIVER,ENABLE TRANSMITTER,
                OUT0    (CNTLA0),C          ;RTS-PIN OFF,RESET ALL ERROR FLAG,
                                            ;START-BIT,8 DATA-BIT,1 STOP-BIT
                LD      C,00001101B         ;PRESCALE ==> /10,DIVIDE RATIO ==>64
                OUT0    (CNTLB0),C          ;SOURCE/SPEED SELECT ==> /32
                IN0     C,(STAT0)
                RES     0,C                 ;TRANSMIT INTERRUPT DISABLE
                SET     3,C                 ;RECEIVE INTERRUPT ENABLE
                OUT0    (STAT0),C
                RET
```

```
;*******************************************************************************
;*******************   SUB ROUTINE DELAY TIME INPUT  BC-REGISTER    *************
;*******************************************************************************
DELAY_SERIAL:  PUSH   BC
               PUSH   AF
DEL_SERIAL:    NOP
               NOP
               NOP
               NOP
               DEC    BC
               LD     A,B
               OR     C
               JR     NZ,DEL_SERIAL
               POP    AF
               POP    BC
               RET
;*******************************************************************************
;*******************   SUB ROUTINE SET MODEM   *********************************
;*******************************************************************************
SET_MODEM:     LD     IX,AT_INITIAL        ;INITIAL MODEM
               CALL   SEND_SELECT
               RET
;*******************************************************************************
```

```
;*****************************************************************************
;******************   SUB ROUTINE SEND CHARACTER  ********************************
;******************   INPUT IX ==> BASE INDEX     ********************************
;*****************************************************************************
SEND_SELECT:   PUSH   DE
               PUSH   IX
               IN0    D,(CNTLA0)          ;READ ASCI CONTROL REGISTER A
               RES    4,D                 ;RTS-PIN ON
               OUT0   (CNTLA0),D
CTS_READY:     IN0    D,(CNTLB0)          ;READ ASCI CONTROL REGISTER B
               BIT    5,D                 ;CHECK CTS-PIN ON
               JR     NZ,CTS_READY        ;YES SEND DATA
               CALL   TRANS_DATA
               POP    IX
               POP    DE
               RET
;*****************************************************************************
;*****************************************************************************
;***********************   SUB ROUTINE  TRANSMITT DATA   ************************
;*****************************************************************************
TRANS_DATA:    PUSH   DE
               PUSH   AF
TDR0_READY:    IN0    A,(STAT0)           ;READ ASCI STATUS REGISTER
               BIT    1,A                 ;CHECK TDRE = 1 ?
               JR     Z,TDR0_READY        ;NO CHECK AGAIN
               LD     D,(IX+0H)           ;YES ==> READ DATA FROM BUFFER
               OUT0   (TDR0),D            ;SEND DATA VIA TRANSMIT DATA REGISTER
               LD     A,D
               CP     0DH                 ;CHECK ENTER ?
               JR     Z,END_TRANS         ;YES ==> END TRANSMITT
```

```
                INC     IX                  ;NO ==> NEXT CHARACTER
                JR      TDRO_READY
END_TRANS:      IN0     D,(CNTLA0)          ;RTS_PIN OFF
                SET     4,D
                OUT0    (CNTLA0),D
                POP     AF
                POP     DE
                RET
;*******************************************************************************
;*******************************************************************************
;************************  SUB ROUTINE CHECK STOLEN  ***************************
;*******************************************************************************
SCAMBER:        LD      IX,BUF_SEND
                LD      IY,BUF_SCAM
                LD      B,00H
CON_SCAM:       LD      A,(IX+0)
                CP      0DH
                JR      Z,EXIT_SCAM
                XOR     5AH             ;SPECIAL VALUE
                LD      (IY+0),A
                INC     IX
                INC     IY
                INC     B
                JP      CON_SCAM
EXIT_SCAM:      LD      A,0DH
                LD      (IY+0),A
                RET
;*******************************************************************************
```

```
;*****************************************************************************
;************************  SUB ROUTINE SET INTERRUPT  ************************
;*****************************************************************************
SET_INT:        LD      A,07FH
                LD      I,A
                LD      B,00H
                OUT0    (IL),B
                IN0     A,(ITC)
                OR      00000110B           ;ENABLE INT1,INT2
                OUT0    (ITC),A
                RET
;*****************************************************************************
;***************************  SUB ROUTINE SET TIMER 0  ***************************
;*****************************************************************************
SET_TIMER0:     LD      B,10H
                OUT0    (TCR),B             ;SET INTERRUPT TIMER
                LD      B,0BAH
                OUT0    (TMDR0L),B          ;SET DATA REGISTER LOW = 0BAH
                OUT0    (RLDR0L),B          ;SET RELOAD REGISTER LOW = 0BAH
                LD      B,0DAH
                OUT0    (TMDR0H),B          ;SET DATA REGISTER HIGH = 0DAH
                OUT0    (RLDR0H),B          ;SET RELOAD REGISTER HIGH = 0DAH
                RET
;*****************************************************************************
;*********************    SUB ROUTINE RUN TIMER 0    *****************************
;*****************************************************************************
RUN_TIMER0:     IN0     A,(TCR)
                OR      00000013            ;ENABLE TIMER
                OUT0    (TCR),A
                RET
```

```
;*******************************************************************************

;************************  SUB ROUTINE STOP TIMER 0  ***************************

;*******************************************************************************

STOP_TIMER0:   IN0    A,(TCR)
               AND    11111110B          ;DISABLE TIMER
               OUT0   (TCR),A
               RET

;*******************************************************************************

;*******************  SUB ROUTINE CLEAR BUFFER DATA OF TIMER  ******************

;*******************************************************************************

CLR_BUF_TIME:  LD     A,00H
               LD     (BUF_PRESEC),A
               LD     (BUF_SEC),A
               LD     (BUF_MIN),A
               LD     (BUF_HR),A
               LD     (SECOUND1),A
               LD     (SECOUND2),A
               LD     (MIN1),A
               LD     (MIN2),A
               LD     (HOUR1),A
               LD     (HOUR2),A
               LD     (BUF_PRI1),A
               LD     (BUF_PRI2),A
               LD     (BUF_PRI3),A
               RET
```

```
;*****************************************************************************
;***********************    SUB ROUTINE SET PAGE UP/DOWN  ************************
;*****************************************************************************
SET_PAGE:       LD      HL,SOURCE_DATA
                LD      (SOURCE_BUFF),HL
                LD      DE,DEST_DATA
                LD      (DEST_BUFF),DE
                LD      IX,PAGE_DATA
                LD      (COUNT_BUFF),IX
                RET
;*****************************************************************************
;************************  SUB ROUTINE DISPLAY DATA  *************************
;*****************************  INPUT REGISTER    ****************************
;*****************************  IY = INDEX DATA   ****************************
;************************  HL = ADDRESS RAM DISPLAY  *************************
;************************  B = COUNT OF STRING DISPLAY  **********************
;*****************************************************************************
DISP_GEN:       PUSH    BC
DIS_CON:        LD      A,(IY+0)
                LD      (HL),A
                INC     HL
                INC     IY
                DJNZ    DIS_CON
                POP     BC
                RET
;*****************************************************************************
```

```
;*******************************************************************************
;**********************   SUB ROUTINE RECEIVE KEYPRESS  ************************
;*******************************************************************************
                .ORG    4000H
REC_KEY:        PUSH    IX
                PUSH    IY
                PUSH    AF
                PUSH    BC
                PUSH    DE
                PUSH    HL
                IN0     C,(8DH)             ;READ KEY
                CALL    COMPARE_KEY
                CALL    DELAY_KEY
                POP     HL
                POP     DE
                POP     BC
                POP     AF
                POP     IY
                POP     IX
                EI
                RETI
;*******************************************************************************
```

```
;*******************************************************************************
;************************  SUB ROUTINE DELAY KEY  ******************************
;*******************************************************************************
DELAY_KEY:      PUSH    BC
                PUSH    AF
                LD      BC,7000H
DEL_KEY:        DEC     BC
                LD      A,B
                OR      C
                JR      NZ,DEL_KEY
                POP     AF
                POP     BC
                RET
;*******************************************************************************
;************************  SUB ROUTINE COMPARE KEYPRESS  ***********************
;************************  INPUT  REGISTER C = KEYPRESS  ***********************
;************************  OUTPUT REGISTER D = CODEKEY   ***********************
;*******************************************************************************
COMPARE_KEY:    LD      D,011H              ;COUNTER OF KEY
                LD      HL,6011h
COMP_KEY:       LD      A,(HL)
                CP      C
                JR      Z,BUFFER_KEY
                DEC     HL
                DEC     D
                JP      P,COMP_KEY
                JR      EXIT_KEY
BUFFER_KEY:     CALL    CODE_TO_ASCII
EXIT_KEY:       RET
```

```
;*****************************************************************************
;**********************   SUB ROUTINE CODEKEY TO ASCII   **********************
;************************   INPUT  REGISTER D = CODEKEY   ************************
;**************************   OUTPUT REGISTER A = ASCII   **************************
;*****************************************************************************
CODE_TO_ASCII: LD     A,d
               CP     0AH
               JR     NC,LARGE_NUM
               JR     E_KEYBOARD
LARGE_NUM:     LD     A,d
               SUB    9H
               ADD    A,40H              ;CHANG TO ASCII
               CALL   FUNC_KEY
E_KEYBOARD:    RET
;*****************************************************************************
;**********************   SUB ROUTINE CHECK FUNCTION KEY   **********************
;*****************************************************************************
FUNC_KEY:      CP     41H
               JR     Z,KEY_INFOR        ;INFORMATION KEY
               CP     42H                ;RECEIVE DATA FROM KEYBOARD
               JR     Z,KEY_SEND         ;AND SEND VIA SERIAL
               CP     43H
               JR     Z,KEY_DIAL         ;ENTER PHONE NUMBER AND DIAL
               CP     45H
               JR     Z,KEY_BREAK        ;ON-HOOK AND OFFLINE
               CP     46H
               JR     Z,KEY_PRINT        ;PRINT DATA  VIA PRINTER
               CP     47H
               JR     Z,KEY_PADN         ;PAGE DOWN
               CP     48H
```

```
                JR      Z,KEY_PAUP          ;PAGE UP
                JP      TURN_FUNC
KEY_INFOR:      CALL    INFORMATION
                JR      TURN_FUNC
KEY_SEND:       CALL    SEND_KEY_SERIAL
                JR      TURN_FUNC
KEY_DIAL:       CALL    NUM_DIAL
                JR      TURN_FUNC
KEY_BREAK:      CALL    BREAK
                JR      TURN_FUNC
KEY_PRINT:      CALL    PRINT
                JR      TURN_FUNC
KEY_PADN:       CALL    PAGE_DOWN
                JR      TURN_FUNC
KEY_PAUP:       CALL    PAGE_UP
TURN_FUNC:      RET
;*******************************************************************************
;******************  SUB ROUTINE DISPLAY INFORMATION  **************************
;*******************************************************************************
INFORMATION:    CALL    TIME
                CALL    PRICE
                CALL    CLR_RAM
                LD      HL,DISP_FRONT
                LD      IY,VIEW_DATA
                LD      B,25D
                CALL    DISP_GEN
                LD      HL,DISP_TIME
                LD      IY,TIME_DATA
                LD      B,0AH
                CALL    DISP_GEN
```

```
                LD      C,02H
                LD      IY,HOUR2
INFOR_1:        LD      B,02H
                CALL    DISP_GEN
                LD      A,':'
                LD      (HL),A
                INC     HL
                DEC     C
                JR      NZ,INFOR_1
                LD      B,02H
                CALL    DISP_GEN
                LD      HL,DISP_PRICE
                LD      IY,PRICE_DATA
                LD      B,0AH
                CALL    DISP_GEN
                LD      IY,BUF_PRI3
                LD      A,(IY+0)
                CP      30H
                JR      Z,INFOR_2
                LD      B,03H
                CALL    DISP_GEN
                JR      EXIT_PRICE
INFOR_2:        INC     IY
                LD      A,(IY+0)
                CP      30H
                JR      Z,INFOR_3
                LD      B,02H
                CALL    DISP_GEN
                JR      EXIT_PRICE
INFOR_3:        INC     IY
```

```
                LD      B,01H
                CALL    DISP_GEN
EXIT_PRICE:     LD      IY,UNIT
                LD      B,06H
                CALL    DISP_GEN
                RET
;*******************************************************************************
;************************* SUB ROUTINE  TIME  ********************************
;*******************************************************************************
TIME:           LD      A,(BUF_SEC)
                CALL    CHANGE_BCD
                LD      A,L             ;FIRST UNIT OF SEC
                LD      (SECOUND1),A
                LD      A,H             ;SECOND UNIT OF SEC
                LD      (SECOUND2),A
                LD      A,(BUF_MIN)
                CALL    CHANGE_BCD
                LD      A,L             ;FIRST UNIT OF MIN
                LD      (MIN1),A
                LD      A,H             ;SECOND UNIT OF MIN
                LD      (MIN2),A
                LD      A,(BUF_HR)
                CALL    CHANGE_BCD
                LD      A,L             ;FIRST UNIT OF HOUR
                LD      (HOUR1),A
                LD      A,H             ;SECOND UNIT OF HOUR
                LD      (HOUR2),A
                RET
;*******************************************************************************
```

```
;*******************************************************************************
;**********************   SUB ROUTINE CHANG HEX TO BCD  ************************
;*******************************************************************************
CHANGE_BCD:     LD      L,0FFH
CH_BCD1:        INC     L
                SUB     100D
                JR      NC,CH_BCD1
                ADD     A,100D
                LD      (FRACTION),A
                LD      A,L
                LD      (HUND),A
                LD      L,0FFH
                LD      A,(FRACTION)
CH_BCD2:        INC     L
                SUB     10
                JR      NC,CH_BCD2
                ADD     A,10
                LD      C,A
                LD      A,L
                RLCA
                RLCA
                RLCA
                RLCA
                OR      C
                LD      (BCD),A
                AND     0FH                 ;ELIMINATE 4 UPPER BIT
                ADD     A,30H
                LD      L,A
                LD      A,(BCD)
                SRL     A
```

```
                SRL     A
                SRL     A
                SRL     A                       ;ELIMINATE 4 LOWER BIT
                ADD     A,30H
                LD      H,A
                LD      A,(HUND)
                ADD     A,30H
                LD      B,A
                RET
;*******************************************************************************

;*******************************************************************************
;************************   SUB ROUTINE  PRICE   ********************************
;*******************************************************************************
PRICE:          LD      A,(BUF_MIN)
                SLA     A
                CALL    CHANGE_BCD
                LD      A,L                     ;FIRST UNIT OF PRICE
                LD      (BUF_PRI1),A
                LD      A,H                     ;SECOND UNIT OF PRICE
                LD      (BUF_PRI2),A
                LD      A,B                     ;THIRD UNIT OF PRICE
                LD      (BUF_PRI3),A
                RET
;*******************************************************************************
```

```
;*******************************************************************************
;***********************  SUB ROUTINE RECEIVE DATA FROM KEY  *******************
;************************  AND SEND DATA VIA SERIAL PORT  ***********************
;*******************************************************************************
SEND_KEY_SERL: CALL   CLR_RAM
               LD     IY,MENU_SEND
               LD     HL,GENERAL_DISPLAY
               LD     B,12D
               CALL   DISP_GEN
               LD     IX,BUF_SEND
               CALL   ENTER_NUM
               LD     A,0DH              ;ENTER KEY
               LD     (IX+0),A
               CALL   SCAMBER
SEND_AGAINT:   LD     IX,BUF_SCAM
               CALL   SEND_SELECT
SEND_OK:       RET
;*******************************************************************************
;***********************  SUB ROUTINE ENTER NUMBER OF DIAL PHONE ****************
;***************************  KEY ENTER NUMBER  OF DIAL  PHONE  ******************
;*******************************************************************************
NUM_DIAL:      CALL   CLR_RAM
               LD     IY,DISP_DIAL
               LD     HL,GENERAL_DISPLAY
               LD     B,22D
               CALL   DISP_GEN
               LD     IX,BUF_DIALNUM
               CALL   ENTER_NUM
               LD     A,0DH              ;ENTER KEY
               LD     (IX+0),A
```

```
                LD      HL,AT_PHONE
                LD      DE,BUF_AT_PHONE
                LD      BC,06H
                LDIR
                LD      IX,BUF_AT_PHONE
                CALL    SEND_SELECT
                RET
;*******************************************************************************
;*********************   SUB ROUTINE RECEIVE DATA FROM KEYBOARD   ***************
;**********************  OUTPUT REGISTER B = 00H IF KEY=ENTER   ***************
;***************************         C = KEYPRESS   ******************************
;*******************************************************************************
ENTER_NUM:      IN0     A,(ITC)
                AND     11111101B           ;DISABLE INT1 FROM KEYBOARD
                OUT0    (ITC),A
                LD      A,00H
                LD      (COUNT_DEL),A       ;CLEAR NUMBER OF DELETE
READ_KEY:       IN0     C,(8DH)
                CALL    DELAY_KEY
                LD      A,C
                CP      0FEH
                JR      Z,CHANG_ASCII
                CP      0FDH
                JR      Z,CHANG_ASCII
                CP      0FCH
                JR      Z,CHANG_ASCII
                CP      0FBH
                JR      Z,CHANG_ASCII
                CP      0FAH
                JR      Z,CHANG_ASCII
```

```
                CP      0F9H
                JR      Z,CHANG_ASCII
                CP      0F8H
                JR      Z,CHANG_ASCII
                CP      0F7H
                JR      Z,CHANG_ASCII
                CP      0EFH
                JR      Z,CHANG_ASCII
                CP      0DFH
                JR      Z,CHANG_ASCII
                CP      08FH
                JR      Z,DELETE            ;DELETE DATA
                CP      9FH
                JR      Z,EXIT_READ
                JP      READ_KEY
CHANG_ASCII:    CALL    CHG_CODE_ASC
                JP      READ_KEY
DELETE:         LD      A,(COUNT_DEL)
                CP      00H
                JR      NZ,DEL_DATA
                JP      READ_KEY
DEL_DATA:       DEC     IX
                DEC     HL
                LD      A,00H
                LD      (IX+0),A
                LD      (HL),A
                LD      A,(COUNT_DEL)
                DEC     A
                LD      (COUNT_DEL),A
                JP      READ_KEY
```

```
EXIT_READ:    LD    B,00H              ;RETURN B=00H IF KEY = ENTER
              IN0   A,(ITC)
              OR    00000010B          ;ENABLE INT1
              OUT0  (ITC),A
              RET
;*******************************************************************************
;************************** SUB ROUTINECHECK DATA AND ****************************
;**************************   CHANG CODE TO ASCII  ******************************
;*******************************************************************************
CHG_CODE_ASC: LD    D,09H
              LD    IY,6009H
CHECK_CODE:   LD    A,(IY+0)
              CP    C
              JR    Z,TO_ASCII
              DEC   IY
              DEC   D
              JP    P,CHECK_CODE
              JR    CHANG_RET
TO_ASCII:     LD    A,(COUNT_DEL)
              INC   A
              LD    (COUNT_DEL),A
              LD    A,D
              ADD   A,30H              ;CHANG TO ASCII
              LD    (IX+0),A
              LD    (HL),A
              INC   IX
              INC   HL
CHANG_RET:    RET
;*******************************************************************************
```

```
;*******************************************************************************
;**********************   SUB ROUTINE BREAK   **********************************
;*******************************************************************************
BREAK:          LD      HL,0D8C8H
                LD      IY,DATA_BREAK
                LD      B,120D
                CALL    DISP_GEN
                CALL    REC_NUM_SEC
                LD      A,B
                CP      31H
                JR      Z,BREAK_1           ;BREAK PRINT
                CP      32H
                JR      Z,BREAK_2           ;BREAK SERIAL
                JP      EXIT_BREAK
BREAK_1:        CALL    BREAK_PRINT
                JP      EXIT_BREAK
BREAK_2:        CALL    DISABLE_REC
EXIT_BREAK:     RET
;*******************************************************************************
;**********************   SUB ROUTINE RECEIVE KEY 1 OR 2   *********************
;*****************************   OUTPUT REGISTER B   ***************************
;*******************************************************************************
REC_NUM_SEC:    IN0     A,(ITC)
                AND     11111101B           ;DISABLE INT1 FROM KEYBOARD
                OUT0    (ITC),A
READK_1_2:      IN0     C,(8DH)
                CALL    DELAY_KEY
                LD      A,C
                CP      0FDH                ;KEYPRESS FOR KEY1
                JR      Z,BREAK_PT
```

```
                CP      0FCH            ;KEYPRESS FOR KEY2
                JR      Z,BREAK_COMM
                JR      READK_1_2
BREAK_PT:       LD      B,31H           ;ASCII FOR KEY1
                JR      EXIT_K1_2
BREAK_COMM:     LD      B,32H           ;ASCII FOR KEY2
EXIT_K1_2:      IN0     A,(ITC)
                OR      00000010B       ;ENABLE INT1
                OUT0    (ITC),A
                RET
;*****************************************************************************
;***********************   SUB ROUTINE BREAK PRINT  ****************************
;*****************************************************************************
BREAK_PRINT:    LD      A,CANCEL        ;CLEAR BUFFER OF PRINT
                OUT0    (PORTB2_8255),A
                CALL    STROBE
                CALL    DELAY_PRINT
                CALL    CARRIER_LINEF
                IN0     A,(ITC)
                AND     11111011B       ;DISABLE INT2
                OUT0    (ITC),A
                RET
```

```
;*******************************************************************************
;************************  SUB ROUTINE PRINT DATA  ****************************
;*******************************************************************************
PRINT:          IN0     A,(ITC)
                OR      00000100B
                OUT0    (ITC),A             ;ENABLE INT2
                LD      HL,0000H
                LD      (SAVE_CPRINT),HL
                LD      IY,BUFF_REC_DATA
                LD      (SAVE_ADD_PRINT),IY
                LD      A,00H
                LD      (SAVE_C40),A
                LD      (COUNT_SPACE),A
                LD      HL,(COUNT_RECDATA)
                INC     HL
                LD      (COUNT_RECDATA),HL
                CALL    CARRIER_LINEF
                RET
;*******************************************************************************
;***************  SUB ROUTINE CARRIER RETURN AND LINE FEED  ********************
;*******************************************************************************
CARRIER_LINEF:  LD      A,CR                ;CARRIER RETURN
                OUT0    (PORTB2_8255),A
                CALL    STROBE
                CALL    DELAY_PRINT
                LD      A,LF                ;LINE FEED
                OUT0    (PORTB2_8255),A
                CALL    STROBE
                CALL    DELAY_PRINT
                RET
```

```
;*******************************************************************************
;************************   SUB ROUTINE DELAY PRINT   ****************************
;*******************************************************************************
DELAY_PRINT:   PUSH   BC
               LD     B,20H
DELAY_P:       DJNZ   DELAY_P
               POP    BC
               RET
;*******************************************************************************
;*********************   SUB ROUTINE ACTIVE STROBE SIGNAL   ********************
;*******************************************************************************
STROBE:        LD     A,00H              ;LOW
               OUT0   (PORTC2_8255),A
               XOR    A
               LD     A,0FFH             ;HIGH
               OUT0   (PORTC2_8255),A
               RET
;*******************************************************************************
;************************   SUB ROUTINE SEND BLANK   ***************************
;*******************************************************************************
SEND_BKSP:     LD     B,40D
PR_BKSP:       LD     A,BACKSPACE
               OUT0   (PORTB2_8255),A
               CALL   STROBE
               NOP
               NOP
               CALL   DELAY_PRINT
               DJNZ   PR_BKSP
               RET
;*******************************************************************************
```

```
;*******************************************************************************
;************************  SUB ROUTINE PAGE DOWN  ************************
;*******************************************************************************
                .ORG    3600H
PAGE_DOWN:      PUSH    AF
                PUSH    HL
                PUSH    DE
                PUSH    BC
                CALL    DELAY_KEY
                CALL    CLR_RAM
                CALL    DELAY_KEY
                LD      HL,(SOURCE_BUFF)
                LD      BC,(COUNT_BUFF)
                CALL    CHECK_STOP
                JR      Z,END_LOOP
                ADD     HL,BC
                LD      (SOURCE_BUFF),HL
END_LOOP:       LD      DE,(DEST_BUFF)
                LDIR
                POP     BC
                POP     DE
                POP     HL
                POP     AF
                RET
```

```
;*******************************************************************************
;**************************  SUB ROUTINE PAGE UP  ******************************
;*******************************************************************************
                .ORG    3700H
PAGE_UP:        PUSH    AF
                PUSH    HL
                PUSH    DE
                PUSH    BC
                CALL    DELAY_KEY
                CALL    CLR_RAM
                CALL    DELAY_KEY
                LD      HL,(SOURCE_BUFF)
                LD      BC,(COUNT_BUFF)
                CALL    CHECK_BEGIN
                JR      Z,OUT_LOOP
                CALL    CLEAR_CARRY
                SBC     HL,BC
                LD      (SOURCE_BUFF),HL
OUT_LOOP:       LD      DE,(DEST_BUFF)
                LDIR
                POP     BC
                POP     DE
                POP     HL
                POP     AF
                RET
```

```
;*******************************************************************************
;************************  SUB ROUTINE CHECK BEGIN  ************************
;*******************************************************************************
                .ORG    3800H
CHECK_BEGIN:    PUSH    HL
                PUSH    DE
                LD      HL,(SOURCE_BUFF)
                LD      DE,START_DATA
                CALL    CLEAR_CARRY
                SBC     HL,DE
                LD      A,L
                OR      H
                POP     DE
                POP     HL
                RET
;*******************************************************************************
;**********************  SUB ROUTINE CHECK STOP  *******************************
;*******************************************************************************
                .ORG    3900H
CHECK_STOP:     PUSH    HL
                PUSH    DE
                LD      HL,END_DATA
                LD      DE,(SOURCE_BUFF)
                CALL    CLEAR_CARRY
                SBC     HL,DE
                LD      A,L
                OR      H
                POP     DE
                POP     HL
                RET
```

```
;*****************************************************************************
;******************   SUB ROUTINE CLEAR CARRY FLAG   ***************************
;*****************************************************************************
CLEAR_CARRY:   JR     NC,NO_CARRY
               CCF
NO_CARRY:      RET
;*****************************************************************************
;**********************   SUB ROUTINE SERVICE INTERRUPT  ***********************
;***************************   CHECK INTERRUPT SOURCE   ************************
;*****************************************************************************
               .ORG   4500H
;*****************************************************************************
;********************   SUB ROUTINE CHECK INTERUPT SOURCE  ***********************
;*****************************************************************************
CHECK_INT_SOE: PUSH   IX
               PUSH   IY
               PUSH   AF
               PUSH   BC
               PUSH   DE
               PUSH   HL
               IN0    A,(STAT0)          ;READ ASCI STATUS REGISTER
               IN0    A,(STAT0)
               BIT    7,A                ;CHECK RECEIVE DATA REGISTER FULL ?
               JR     NZ,RECEIVE_FULL    ;YSE ==> RECEIVE DATA
               BIT    2,A                ;CHECK DCD-PIN ON ?
               JR     Z,DCD_ON           ;YES ==> ENABLE TIMER
               JP     RETN_SERIAL        ;RETURN INTERRUPT
RECEIVE_FULL:  CALL   RECV_DATA
               JP     RETN_SERIAL
DCD_ON:        CALL   RUN_TIMER0
```

```
RETN_SERIAL:  POP    HL
              POP    DE
              POP    BC
              POP    AF
              POP    IY
              POP    IX
              EI
              RETI
;*****************************************************************************
;**********************  SUB ROUTINE  RECEIVE DATA  ****************************
;*****************************************************************************
RECV_DATA:    LD     IY,(SAVE_AREC)
              LD     IX,(SAVE_VIDEO)
              IN0    A,(STAT0)
REC_CON:      IN0    A,(STAT0)
              BIT    7,A                ;CHECK RECEIVE DATA REGISTER FULL ?
              JR     Z,RETURN_REC       ;NO ==> CHECK CON
              IN0    A,(RDR0)           ;YES ==> READ DATA
              LD     (IY+0),A           ;SAVE DATA TO BUFFER
              LD     B,A                ;SAVE
              IN0    A,(CNTLA0)
              AND    11110111B
              OUT0   (CNTLA0),A         ;CLEAR ERROR FLAG
              INC    IY
              LD     (SAVE_AREC),IY
              LD     DE,0DBE8H
              XOR    A
              PUSH   IX
              POP    HL
              SBC    HL,DE
```

```
                JP      NC,CHECK_ETX
                LD      A,B
                LD      (IX+0),A            ;SAVE DATA TO VIDEO RAM
                INC     IX
                LD      (SAVE_VIDEO),IX
CHECK_ETX:      LD      A,B
                CP      0DH                 ;CHECK DATA END ?
                JR      NZ,RETURN_REC       ;NO ==> EXIT FROM INTERRUPT
                LD      IY,BUFF_REC_DATA
                LD      (SAVE_AREC),IY
                LD      IX,VIDEO_RAM
                LD      (SAVE_VIDEO),IX
                CALL    CHECK_DATAT         ;YES ==> CHECK TOTAL RECEIVE DATA
RETURN_REC:     RET
;*******************************************************************************
;*********************   SUB ROUTINE CHECK TOTAL DATA   ************************
;*******************************************************************************
CHECK_DATAT:    LD      IY,BUFF_REC_DATA
                LD      HL,0000H            ;CLEAR COUNTER
CHK_CON:        INC     HL
                LD      A,(IY+0)
                CP      0DH                 ;CHECK ENTER ?
                JR      Z,END_CMP           ;YES ==> EXIT
                INC     IY                  ;NO ==> CHECK CONTINUTE
                JR      CHK_CON
END_CMP:        LD      (COUNT_RECDATA),HL  ;SAVE COUNT VALUE IN MEMORY
                RET
;*******************************************************************************
```

```
;*****************************************************************************
;***********************  SUB ROUTINE ON-HOOK  *******************************
;*****************************************************************************
DISABLE_REC:  IN0   A,(STAT0)          ;DISABLE ALL INTERRUPT FROM SERIAL
              AND   11110110B
              OUT0  (STAT0),A
              IN0   A,(CNTLA0)
              AND   10111111B          ;DISABLE RECEIVER
              OUT0  (CNTLA0),A
              LD    C,'+'
              LD    B,03H
              IN0   D,(CNTLA0)         ;READ ASCI CONTROL REGISTER A
              RES   4,D                ;RTS-PIN ON
              OUT0  (CNTLA0),D
CTS1_READY:   IN0   D,(CNTLB0)         ;READ ASCI CONTROL REGISTER B
              BIT   5,D                ;CHECK CTS-PIN ON
              JR    NZ,CTS1_READY      ;YES SEND DATA
TDR01_READY:  IN0   A,(STAT0)          ;READ ASCI STATUS REGISTER
              BIT   1,A                ;CHECK TDRE = 1 ?
              JR    Z,TDR01_READY      ;NO CHECK AGAIN
              LD    D,C
              OUT0  (TDR0),D           ;SEND DATA VIA TRANSMIT DATA REGISTER
              CALL  DELAY_SERIAL
              CALL  DELAY_SERIAL
              DEC   B
              JR    NZ,TDR01_READY
END1_TRANS:   IN0   D,(CNTLA0)         ;RTS_PIN OFF
              SET   4,D
              OUT0  (CNTLA0),D
              LD    B,20H
```

```
KKK:        CALL    DELAY_SERIAL
            CALL    DELAY_SERIAL
            DJNZ    KKK
            LD      IX,AT_ON_HOOK       ;AT-COMMAND ON-HOOK
            CALL    SEND_SELECT         ;SEND COMMAND ON-HOOK
            CALL    STOP_TIMER0         ;DISABLE TIMER
            RET
;*******************************************************************************
;*****************  SUB ROUTINE SERVICE INTERRUPT TIMER  ***********************
;*******************************************************************************
            .ORG    5000H
TIMER0:     PUSH    AF
            IN0     A,(TCR)
            IN0     A,(TMDR0L)          ;RESET TIF0
            LD      A,(BUF_PRESEC)
            INC     A
            LD      (BUF_PRESEC),A
            CP      05H                 ;CHECK 1 SEC ?
            JR      NZ,EXIT_TIMER0      ;NO
            LD      A,00H
            LD      (BUF_PRESEC),A      ;YES CLEAR
            LD      A,(BUF_SEC)
            INC     A
            LD      (BUF_SEC),A
            CP      60D                 ;CHECK 1 MIN ?
            JR      NZ,EXIT_TIMER0      ;NO
            LD      A,00H
            LD      (BUF_SEC),A         ;YES CLEAR
            LD      A,(BUF_MIN)
            INC     A
```

```
                LD      (BUF_MIN),A
                CP      60D             ;CHECK 1 HOUR ?
                JR      NZ,EXIT_TIMER0  ;NO
                LD      A,00H
                LD      (BUF_MIN),A     ;YES CLEAR
                LD      A,(BUF_HR)
                INC     A
                LD      (BUF_HR),A
EXIT_TIMER0:    POP     AF
                EI
                RETI
;*****************************************************************************
;******************  SUB ROUTINE SERVICE INTERRUPT PRINTER  ******************
;*****************************************************************************
                .ORG    5500H
INT2_PRINT:     PUSH    AF
                PUSH    BC
                PUSH    DE
                PUSH    HL
                PUSH    IY
                LD      HL,(SAVE_CPRINT)
                LD      DE,(COUNT_RECDATA)
                XOR     A
                SBC     HL,DE
                JR      Z,END_PRINT
                LD      A,(SAVE_C40)
                CP      40D             ;CHECK DATA 40 BYTE
                JR      NZ,PRINT_CON
                LD      A,(COUNT_SPACE)
                CP      40D             ;CHECK SPACE 40 BYTE
```

```
                JR      NZ,PR_SPACE
                CALL    CARRIER_LINEF
                LD      A,00H
                LD      (SAVE_C40),A
                LD      (COUNT_SPACE),A
PRINT_CON:      LD      IY,(SAVE_ADD_PRINT)
                LD      A,(IY+0)
                OUT0    (PORTB2_8255),A
                CALL    STROBE
                INC     IY
                LD      (SAVE_ADD_PRINT),IY
                LD      A,(SAVE_C40)
                INC     A
                LD      (SAVE_C40),A
                LD      HL,(SAVE_CPRINT)
                INC     HL
                LD      (SAVE_CPRINT),HL
                JP      EX_PRINT
PR_SPACE:       LD      A,BACKSPACE
                OUT0    (PORTB2_8255),A
                CALL    STROBE
                LD      A,(COUNT_SPACE)
                INC     A
                LD      (COUNT_SPACE),A
                JP      EX_PRINT
END_PRINT:      CALL    CARRIER_LINEF
                LD      A,CANCEL            ;CLEAR BUFFER OF PRINTER
                OUT0    (PORTB2_8255),A
                CALL    STROBE
                CALL    DELAY_PRINT
```

```
                IN0     A,(ITC)
                AND     11111011B
                OUT0    (ITC),A             ;DISABLE INT2
EX_PRINT:       POP     IY
                POP     HL
                POP     DE
                POP     BC
                POP     AF
                EI
                RETI
;*******************************************************************************
;*******************************************************************************
;*****************************  DATA  AREA  ************************************
;*******************************************************************************
;*******************************************************************************
                .ORG    7F00H
                .DB     00H,40H

                .ORG    7F02H
                .DB     00H,55H

                .ORG    7F04H
                .DB     00H,50H

                .ORG    7F0EH
                .DB     00H,45H

                .ORG    6000H
                .DB     0FEH,0FDH,0FCH,0FBH,0FAH,0F9H,0F8H,0F7H
                .DB     0EFH,0DFH,0CFH,0BFH,0AFH,9FH,8FH,7FH,0F6H,6FH
```

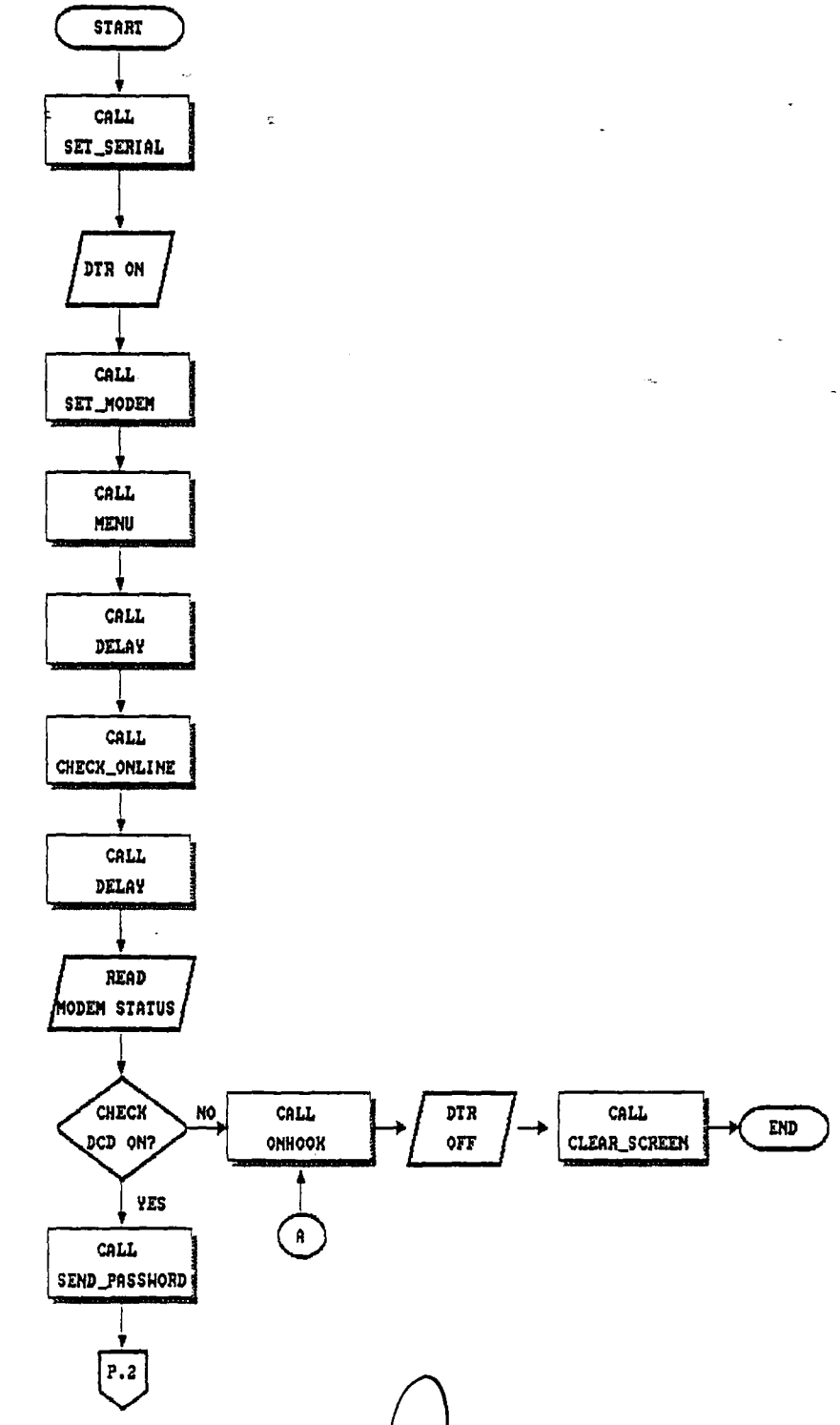
```
VAL_REG_6845:  .DB    44D,40D,41D,02D,27D,05D,25D,27D
               .DB    01H,10D,20H,02H,00H,00H,00H,00H

AT_INITIAL:    .DB    "AT E0 S7=50 S11=70 V0 Q1 S0=2^M",0DH
AT_PHONE:      .DB    "AT DT "
AT_ON_HOOK:    .DB    "AT H0",0DH

DATA1_DIS:     .DB    "VIEWDATA"
DISP_DIAL:     .DB    "ENTER PHONE NUMBER  : "
MENU_SEND:     .DB    "DATA SEND : "
VIEW_DATA:     .DB    "INTERRACTIVE  VIEWDATA... "
TIME_DATA:     .DB    "TIME  ==> "
PRICE_DATA:    .DB    "PRICE ==> "
DATA_BREAK:    .DB    "         ENTER  YOUR  CHOICE          "
               .DB    "       PRESS 1 ==> BREAK PRINT        "
               .DB    "       PRESS 2 ==> BREAK SERIAL       "
UNIT:          .DB    " BAHT "
```

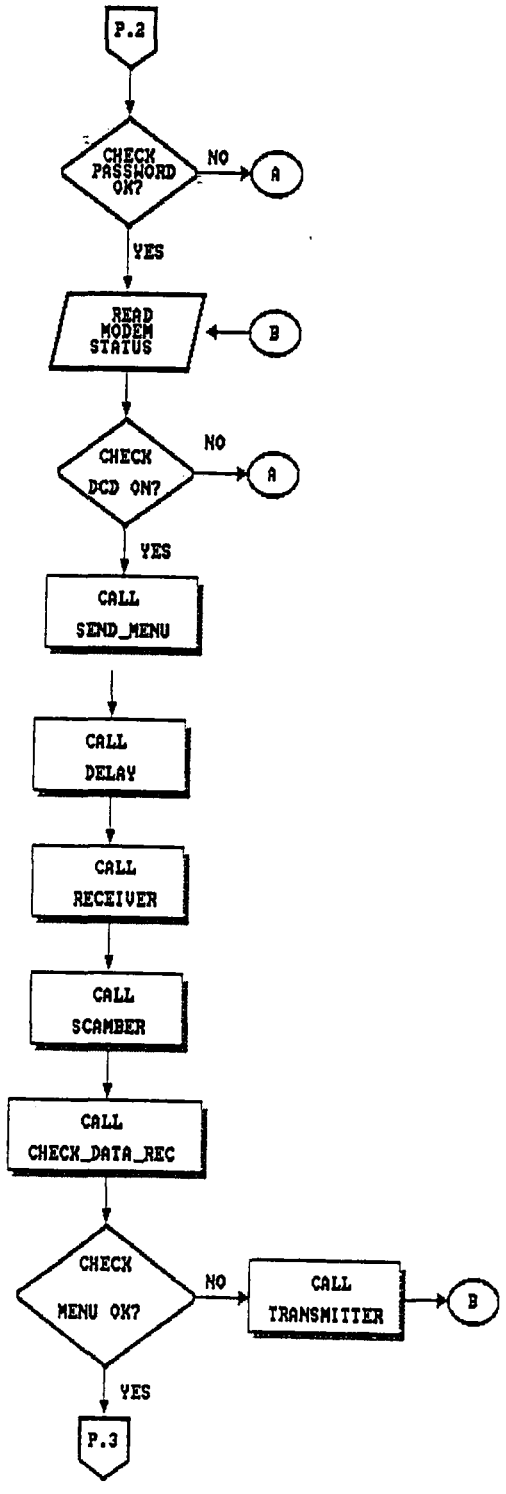
```
DATA_DIS:     .DB   "                                  "
              .DB   "                                  "
              .DB   "  ################################  "
              .DB   "  #                              #  "
              .DB   "  #                              #  "
              .DB   "  #   INTERRACTIVE  VIEWDATA    #  "
              .DB   "  #                              #  "
              .DB   "  #        PRODUCED BY           #  "
              .DB   "  #                              #  "
              .DB   "  #   ROTRIT   CODE 33504031     #  "
              .DB   "  #   SERMCHAI CODE 33504044     #  "
              .DB   "  #   AMPORN   CODE 33504049     #  "
              .DB   "  #                              #  "
              .DB   "  #      APPLIED PHYSICS         #  "
              .DB   "  #                              #  "
              .DB   "  #         SCIENCE              #  "
              .DB   "  #                              #  "
              .DB   "  #          KMITL               #  "
              .DB   "  #                              #  "
              .DB   "  #   ADVISER:                   #  "
              .DB   "  #                              #  "
              .DB   "  #    DR. WARAWUT  TAOLADDA     #  "
              .DB   "  #                              #  "
              .DB   "  ################################  "
              .DB   "                                   "
;*******************************************************************************
```

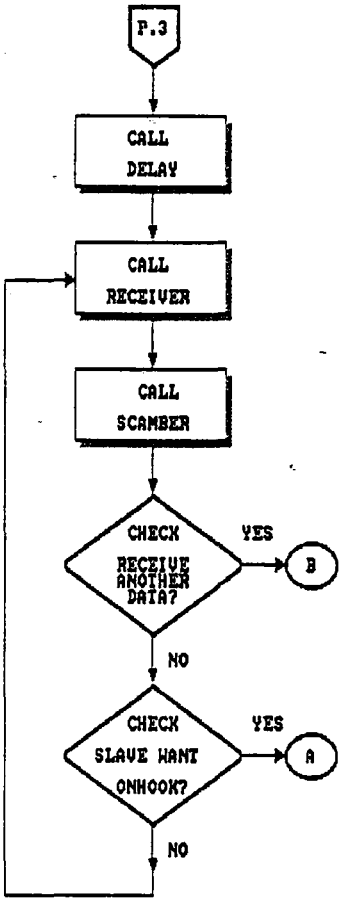
ภาคผนวก ค

MAIN PROGRAM

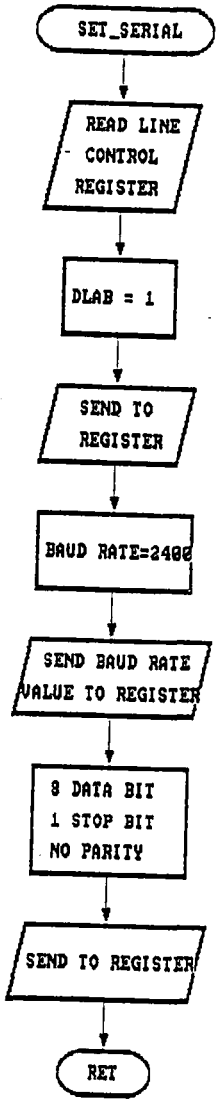
MASTER



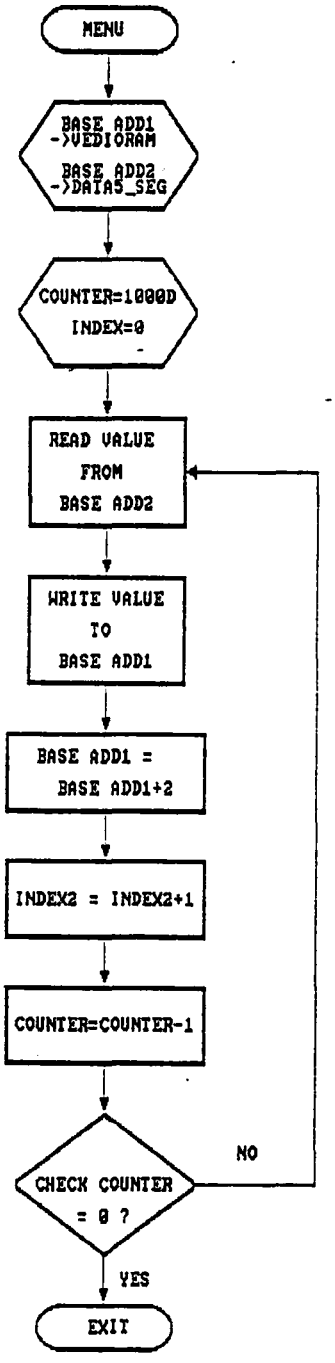

P.2
CHECK PASSWORD OK?
NO
A
YES
READ MODEM STATUS
B
CHECK DCD ON?
NO
A
YES
CALL SEND_MENU
CALL DELAY
CALL RECEIVER
CALL SCAMBER
CALL CHECK_DATA_REC
CHECK MENU OK?
NO
CALL TRANSMITTER
B
YES
P.3


P.3
CALL DELAY
CALL RECEIVER
CALL SCAMBER
CHECK RECEIVE ANOTHER DATA?
YES
B
NO
CHECK SLAVE WANT ONHOOK?
YES
A
NO

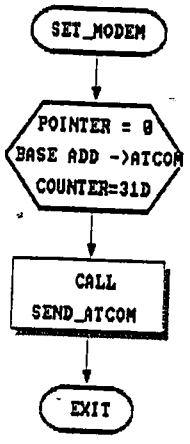
SUB ROUTINE
SET SERIAL



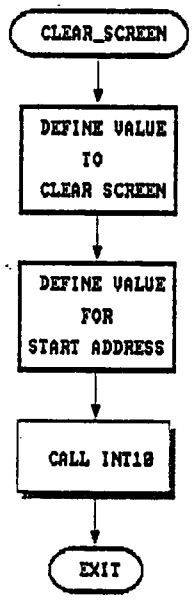
SUB ROUTINE
DISPLAY MENU



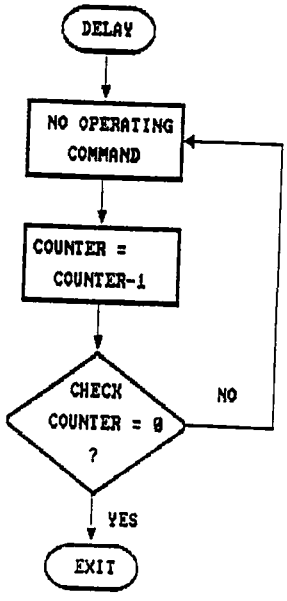
SUB ROUTINE
SET MODEM



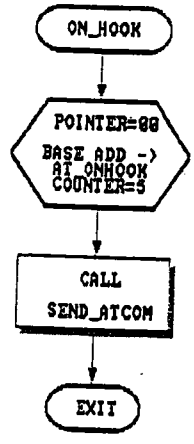
SUB ROUTINE
CLEAR SCREEN



SUB ROUTINE
DELAY

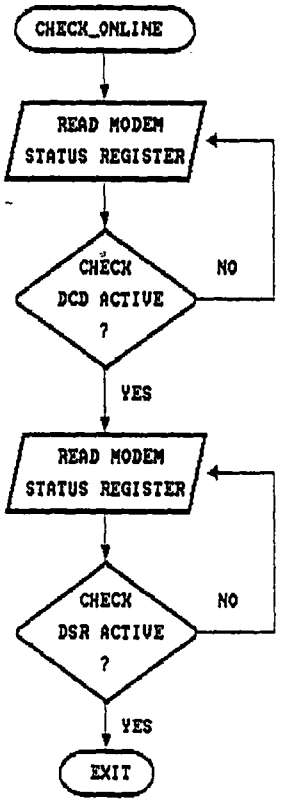


SUB ROUTINE
ON-HOOK



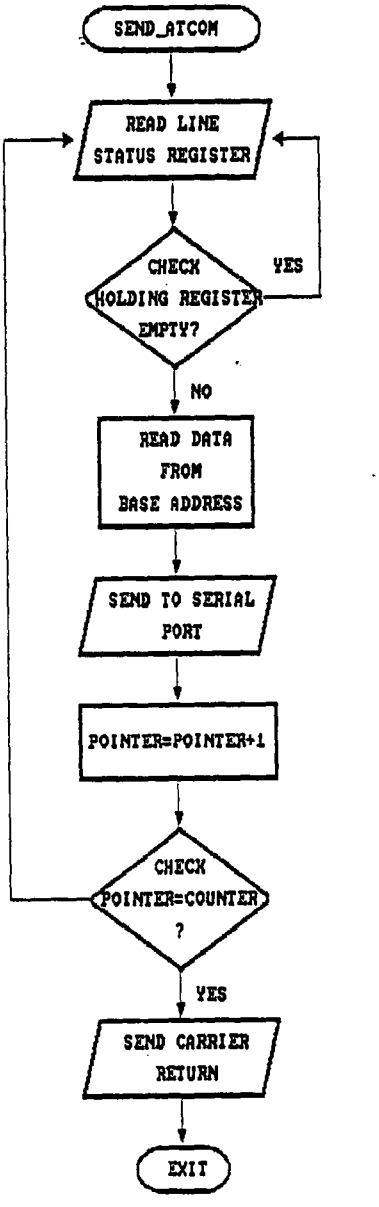
SUB ROUTINE

CHECK ONLINE

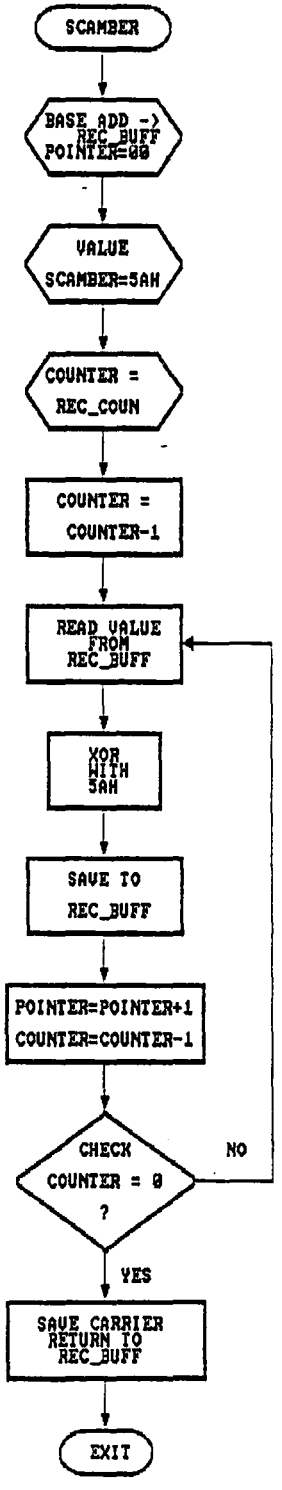


SUB ROUTINE

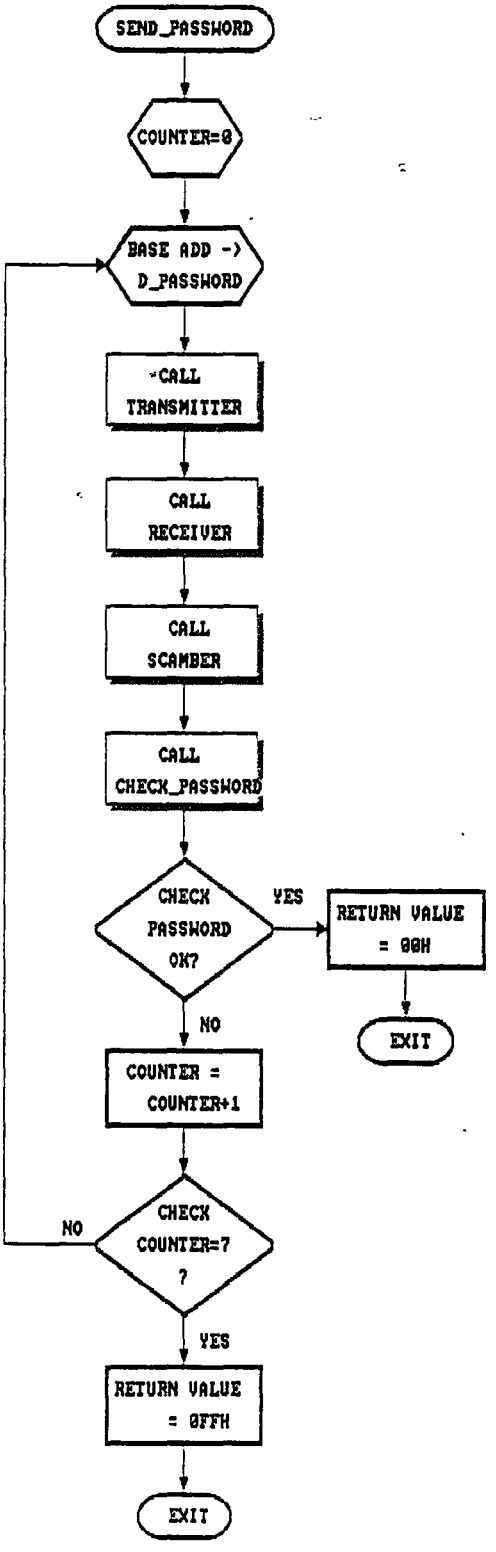
SEND AT-COMMAND



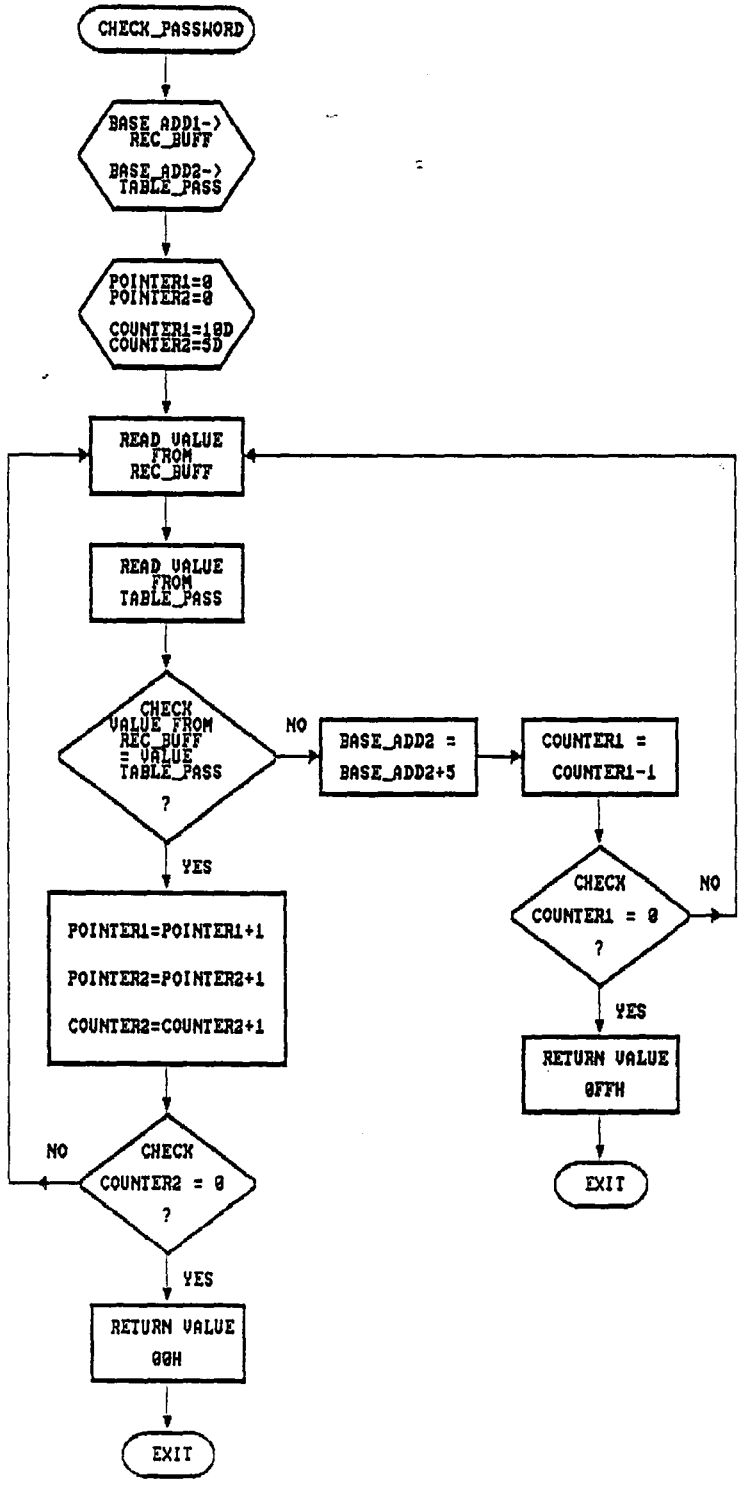
SUB ROUTINE
SCAMBER



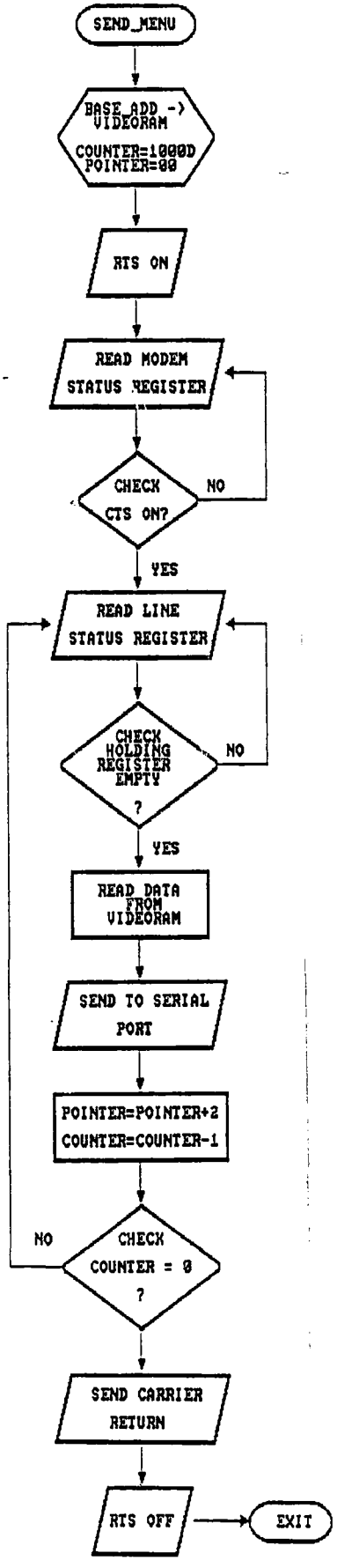
SUB ROUTINE
SEND PASSWORD



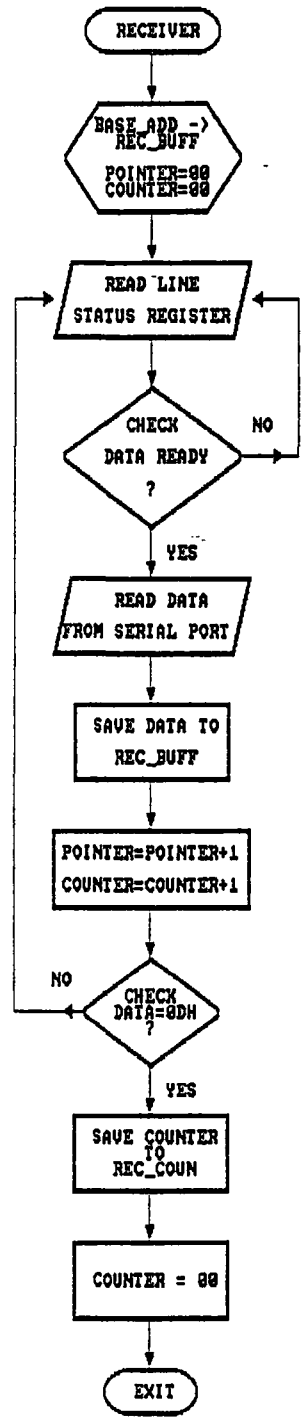
SUB ROUTINE
CHECK PASSWORD



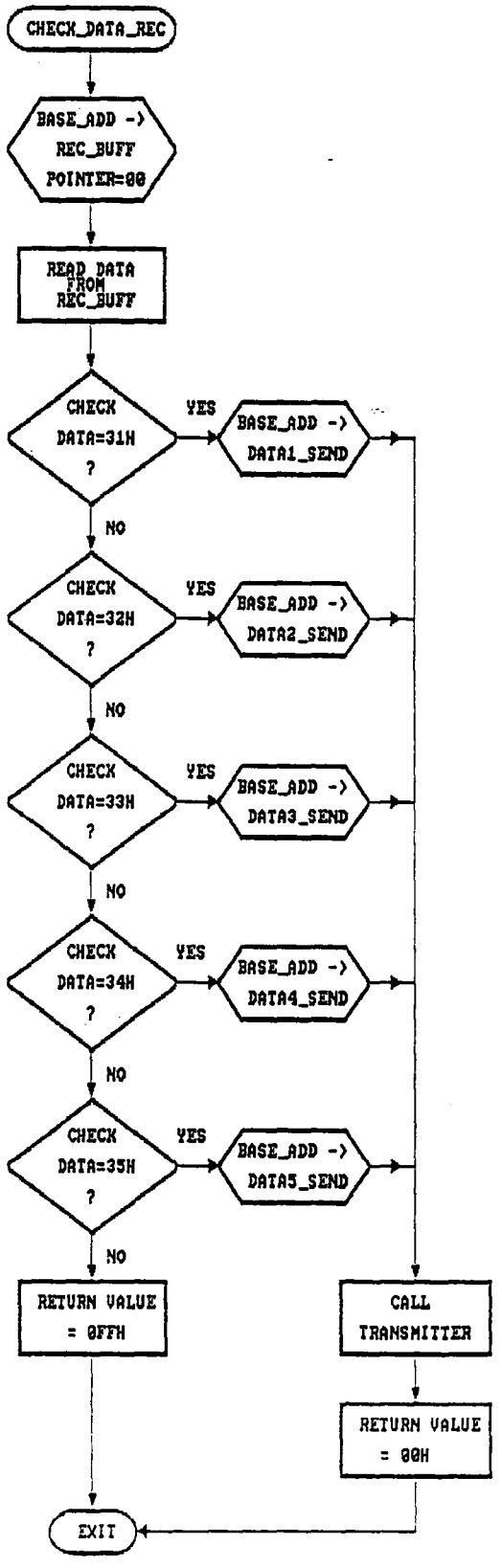
SUB ROUTINE
SEND MENU



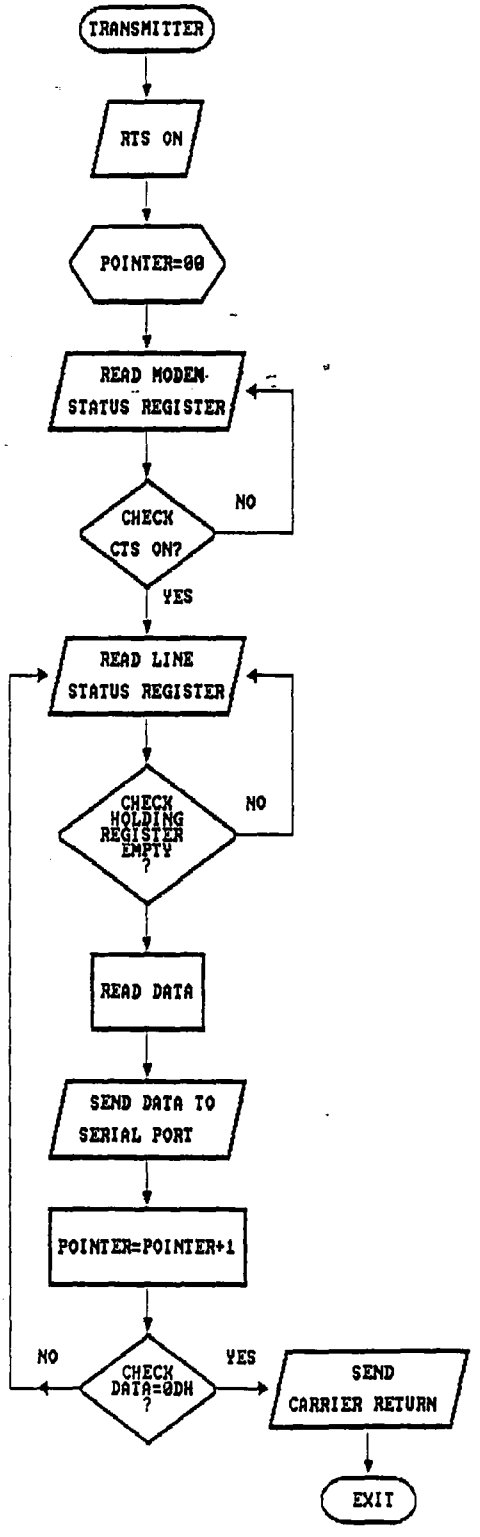
SUB ROUTINE
RECIEVER



SUB ROUTINE
CHECK_DATA_REC



SUB ROUTINE
TRANSMIT DATA



```
;*******************************************************************************
;*******************************************************************************
;******************    PROGRAM MONITOR INTERRACTIVE VIEWDATA   ******************
;******************         FOR   PC  DATA-BASE CENTER         ******************
;*******************************************************************************
;*******************************************************************************
      PAGE            40,150
      TITLE           ASYNCHONOUS SERIAL COMMUNICATION
      SUBTTL          SERIAL FOR COM1
      DATA_GROUP      GROUP DATA1_SEG,DATA2_SEG,DATA3_SEG,DATA4_SEG
;###############################################################################
;##################        BEGIN INSTRUCTION SET              ##################
;###############################################################################
;-------------------------------------------------------------------------------
;CODE AREA
      CODE_SEG        SEGMENT PARA PUBLIC 'CODE'
                      ASSUME  CS:CODE_SEG,SS:STACK_SEG,DS:DATA_GROUP,
                              ES:DATA5_SEG
;*******************************************************************************
;************************       ROUTINE SERIAL COM1          *******************
;*******************************************************************************
      SERIAL_1        PROC    FAR
                      MOV   AX,DATA_GROUP
                      MOV   DS,AX
                      CALL  SET_SERIAL      ;SET SERIAL PORT COM1
                      MOV   DX,SERIAL+4     ;MODEM CONTROL REGISTER
                      MOV   AL,01H          ;DTR-PIN ON
                      OUT   DX,AL
                      CALL  SET_MODEM       ;SET MODEM
                      CALL  MENU
```

```
                MOV   CX,0FFFFH      ;COUNTER OF DELAY
                CALL  DELAY
                CALL  CHECK_ONLINE   ;CHECK ONLINE
                MOV   CX,0FFFFH
                CALL  DELAY
                CALL  DELAY
                MOV   DX,SERIAL+6    ;MODEM STATUS REGISTER
                IN    AL,DX
                TEST  AL,80H         ;CHECK DCD ON
                JZ    OFF_LINE
                CALL  SEND_PASSWORD
                CMP   BL,0FFH        ;PASSWORD OK ?
                JZ    OFF_LINE       ;PASSWORD INCORRECT
DCD_CHANG:      IN    AL,DX
                TEST  AL,80H
                JZ    OFF_LINE
                CALL  SEND_MENU
                MOV   CX,0FFFFH
                CALL  DELAY
                CALL  DELAY
                CALL  RECEIVER
                CALL  SCAMBER
                CALL  CHECK_DATA_REC
                CMP   AH,00H         ;CHECK MENU OK ?
                JZ    NEW_DATA       ;AH=0 OK , AH=0FFH ERROR
                LEA   BX,MENU_ERROR
                CALL  TRANSMITTER
                JMP   DCD_CHANG
NEW_DATA:       MOV   AL,30H
DELAY_AGAIN:    MOV   CX,0FFFFH
```

```
                CALL  DELAY
                DEC   AL
                JNZ   DELAY_AGAIN
CK_REC:         CALL  RECEIVER
                CALL  SCAMBER
                LEA   BX,REC_BUFF
                MOV   SI,00H
                MOV   AL,[BX+SI]
                CMP   AL,31H          ;CHECK RECEIVE ANOTHER DATA
                JZ    DCD_CHANG
OFF_LINE:       CALL  ON_HOOK
                MOV   DX,SERIAL+4     ;MODEM CONTROL REGISTER
                IN    AL,DX
                AND   AL,0FEH         ;DTR_PIN OFF
                OUT   DX,AL
                MOV   CX,0FFFFH
                CALL  DELAY
                CALL  CLEAR_SCREEN
                MOV   AH,02H          ;FUNCTION RETURN KEY STATUS
                INT   16H             ;BIOS INTERRUPT FOR KEYBOARD
END_PROG:       MOV   AH,4CH          ;FUNCTION RETURN TO DOS
                INT   DOSCALL
                RET
SERIAL_1        ENDP
;*******************************************************************************
;**********************     SUB ROUTINE DISPLAY  MENU      **********************
;*******************************************************************************
MENU            PROC  NEAR
                PUSH  AX
                PUSH  BX
```

```
                PUSH  CX
                PUSH  SI
                PUSH  DS
                MOV   AX,0B800H       ;DS REGISTER POINT TO ADDRESS
                MOV   DS,AX           ;VIDEO RAM
                MOV   AX,DATA5_SEG
                MOV   ES,AX
                MOV   BX,0000H        ;BEGIN AT ROW 0,COLUMN 0
                MOV   CX,1000D        ;COUNT DATA IN VIDEO-RAM
                LEA   BP,DATA_MENU
                MOV   SI,00H
DIS_MENU:       MOV   AL,ES:[BP+SI]
                MOV   [BX],AL
                INC   BX              ;CHANGE OVER BYTE ATTRIBUTE
                INC   BX              ;NEXT CHARACTER
                INC   SI
                LOOP  DIS_MENU
                POP   DS
                POP   SI
                POP   CX
                POP   BX
                POP   AX
                RET
MENU            ENDP
```

```
;*****************************************************************************
;*******************   SUB ROUTINE SET SERIAL PORT FOR COM2   ******************
;*******************       8-DATABIT,1-STOPBIT,NO-PARITY      ******************
;*******************           BAUD-RATE 2400 BIT/SEC         ******************
;*****************************************************************************
SET_SERIAL      PROC  NEAR
                MOV   DX,SERIAL+3     ;LINE CONTROL REGISTER
                IN    AL,DX
                OR    AL,80H          ;BIT-DLAB=1
                OUT   DX,AL
                MOV   DX,SERIAL       ;LSB OF LATCH DIVIDER
                MOV   AX,0180H        ;BAUD-RATE 2400
                OUT   DX,AL
                INC   DX              ;MSB OF LATH DIVIDER
                MOV   AL,AH
                OUT   DX,AL
                MOV   DX,SERIAL+3     ;LINE CONTROL REGISTER
                MOV   AL,00000011B    ;8-DATABIT,1-STOPBIT,NO-PARITY
                OUT   DX,AL
                RET
SET_SERIAL      ENDP
;*****************************************************************************
;*****************************************************************************
;*********************       SUB ROUTINE CLEAR SCREEN       ******************
;*****************************************************************************
CLEAR_SCREEN    PROC  NEAR
                MOV   AX,0600H        ;AH=06H(SCROOLL),AL=00H(FULL SCREEN)
                MOV   BH,07H          ;ATTRIBUTE
                MOV   CX,0000H        ;UPPER LEFT ROW:COLUMN
                MOV   DX,184FH        ;LOWER RIGHT ROW:COLUMN
```

```
                INT   10H
                RET
CLEAR_SCREEN    ENDP
;*****************************************************************************
;**********************     SUB ROUTINE INITIAL MODEM     **********************
;**********************      SEND AT-COMMAND TO MODEM     **********************
;*****************************************************************************
SET_MODEM       PROC  NEAR
                MOV   SI,00H          ;POINTER
                LEA   BX,ATCOM1       ;BUFFER AT COMMAND
                MOV   AL,COUN_CHA
                MOV   COUNT_ATCOM,AL ;COUNT CHARACTER
                CALL  SEND_ATCOM
                RET
SET_MODEM       ENDP
;*****************************************************************************
;*******************    SUB ROUTINE SEND AT COMMAND TO MODEM    ******************
;*****************************************************************************
SEND_ATCOM      PROC  NEAR
                PUSH  AX
                PUSH  BX
                PUSH  DX
                PUSH  SI
SEND1:          MOV   DX,SERIAL+5     ;LINE STATUS REGISTER
CHECK_HOLD_EMP: IN    AL,DX
                TEST  AL,20H
                JZ    CHECK_HOLD_EMP ;TEST HOLDING REGISTER EMPTY
                MOV   AL,[BX+SI]      ;SEND AT-COMMAND
                MOV   DX,SERIAL       ;TRANSMITTER HOLDING REGISTER
                OUT   DX,AL
```

```
                INC   SI
                MOV   AX,SI
                CMP   AL,COUNT_ATCOM
                JNZ   SEND1
                MOV   AL,0DH          ;ENTER CHARACTER
                OUT   DX,AL
                POP   SI
                POP   DX
                POP   BX
                POP   AX
                RET
    SEND_ATCOM  ENDP
;*******************************************************************************
;********************      SUB ROUTINE CHECK ONLINE      ***********************
;*******************************************************************************
    CHECK_ONLINE  PROC  NEAR
                MOV   DX,SERIAL+6     ;MODEM STATUS REGISTER
    CKDCD_ON:   IN    AL,DX
                TEST  AL,80H          ;CHECK DCD ACTIVE
                JZ    CKDCD_ON        ;NO ACTIVE
    CKDSR_ON:   IN    AL,DX
                TEST  AL,20H          ;CHECK DSR ACTIVE
                JZ    CKDSR_ON        ;NO ACTIVE
                RET
    CHECK_ONLINE  ENDP
;*******************************************************************************
:*****************   SUB ROUTINE NO CONNECT LINE ===> ONHOOK   *****************
;*******************************************************************************
    ON_HOOK     PROC  NEAR
                PUSH  BX
```

```
                PUSH  CX
                PUSH  SI
                MOV   SI,00H              ;POINTER
                LEA   BX,AT_ONHOOK        ;AT ONHOOK
                MOV   CL,COUNT_ON[SI]     ;MOVE COUNTER
                MOV   COUNT_ATCOM[SI],CL  ;STORE COUNTER
                CALL  SEND_ATCOM          ;SEND AT-COMMAND
                POP   SI
                POP   CX
                POP   BX
                RET
    ON_HOOK     ENDP
;*****************************************************************************
;*********************        SUB ROUTINE DELAY          **********************
;*********************        INPUT     CX               **********************
;*****************************************************************************
    DELAY       PROC  NEAR
                PUSH  CX
    WAIT_REST:  NOP
                NOP
                NOP
                LOOP  WAIT_REST
                POP   CX
                RET
    DELAY       ENDP
```

```
;*******************************************************************************
;*********************     SUB ROUTINE SEND DATA FOR      **********************
;*********************       ENTER CURREN PASSWORD        **********************
;*******************************************************************************
      SEND_PASSWORD   PROC  NEAR
                      MOV   AH,00H        ;COUNT NUMBER OF SEND
                      MOV   COUNT_SEND_P,AH
      TRANS_NEW:      LEA   BX,D_PASSWORD
      SEND_NEW:       CALL  TRANSMITTER
                      CALL  RECEIVER
                      CALL  SCAMBER
                      CALL  CHECK_PASSWORD
                      CMP   AL,00H          ;CHECK PASSWORD OK ?
                      JZ    PASSWORD_OK     ;AL=0 OK ,AL=0FFH ERROR
                      MOV   AH,COUNT_SEND_P
                      INC   AH
                      MOV   COUNT_SEND_P,AH
                      CMP   AH,07H          ;CHECK NUMBER OF SEND
                      JNZ   TRANS_NEW
                      MOV   BL,0FFH         ;RETURN BL=0FFH PASSWORD INCORRECT
                      JMP   END_PASS
      PASSWORD_OK:    MOV   BL,00H          ;RETURN BL=00H PASSWORD CORRECT
      END_PASS:       RET
      SEND_PASSWORD   ENDP
;*******************************************************************************
;*********************      SUB ROUTINE CHECK STOLEN      **********************
;*******************************************************************************
      SCAMBER         PROC  NEAR
                      LEA   BX,REC_BUFF
                      MOV   SI,00H
```

```
                MOV   CH,5AH
                MOV   CL,REC_COUN
                DEC   CL

CON_SCAM:       MOV   AL,[BX+SI]
                XOR   AL,CH
                MOV   [BX+SI],AL
                INC   SI
                DEC   CL
                JNZ   CON_SCAM
EXIT_SCAM:      MOV   AL,0DH
                MOV   [BX+SI],AL
                RET
SCAMBER         ENDP
;*******************************************************************************
;*********************    SUB ROUTINE CHECK PASSWORD      **********************
;*********************    OUTPUT    AL                    **********************
;*******************************************************************************
CHECK_PASSWORD  PROC  NEAR
                LEA   BX,REC_BUFF
                LEA   BP,TABLE_PASS
                MOV   SI,00H          ;COUNT CHARACTER OF SOURCE
                MOV   DI,00H          ;COUNT CHARACTER OF DESTINATION
                MOV   CL,10D          ;COUNT PASSWORD
                MOV   CH,05H
CHAR_NEXT:      MOV   AL,[BX+SI]      ;SOURCE
                MOV   AH,DS:[BP+DI]   ;DESTINATION
                CMP   AL,AH
                JNZ   PASS_NEXT
                INC   SI
```

```
                INC   DI
                DEC   CH
                JNZ   CHAR_NEXT
                MOV   AL,00H          ;PASSWORD OK
                JMP   END_CKPASS
PASS_NEXT:      ADD   BP,05H
                DEC   CL
                JNZ   CHAR_NEXT
                MOV   AL,0FFH         ;PASSWORD ERROR
END_CKPASS:     RET
CHECK_PASSWORD  ENDP
;*******************************************************************************
;************************      SUB ROUTINE SEND MENU      ************************
;*******************************************************************************
SEND_MENU       PROC  NEAR
                PUSH  AX
                PUSH  BX
                PUSH  CX
                PUSH  DX
                PUSH  SI
                PUSH  DS
                MOV   AX,0B800H       ;DS REGISTER POINT TO ADDRESS
                MOV   DS,AX           ;VIDEO RAM
                MOV   BX,0000H        ;BEGIN AT ROW 0,COLUMN 0
                MOV   CX,1000D        ;COUNT DATA IN VIDEO-RAM
                MOV   DX,SERIAL+4     ;MODEM CONTROL REGISTER
                MOV   AL,OUT2_DTR_RTS;RTS-PIN ON
                OUT   DX,AL
                MOV   DX,SERIAL+6     ;MODEM STATUS REGISTER
CHECK_SEND_M1:  IN    AL,DX
```

```
                TEST  AL,10H        ;TEST CTS-PIN ON ?
                JZ    CHECK_SEND_M1
SEND_PAGE:      MOV   DX,SERIAL+5   ;LINE STATUS REGISTER
CHECK_SEND_M2:  IN    AL,DX
                TEST  AL,20H        ;TEST HOLDING REGISTER EMPTY ?
                JZ    CHECK_SEND_M2
                MOV   AL,[BX]       ;GET DATA FROM VIDEO-RAM
                MOV   DX,SERIAL     ;TRANSMITTER HOLDING REGISTER
                OUT   DX,AL         ;SEND DATA
                INC   BX            ;CHANGE OVER BYTE ATTRIBUTE
                INC   BX            ;NEXT CHARACTER
                LOOP  SEND_PAGE
                MOV   AL,0DH        ;SEND ENTER
                OUT   DX,AL
                MOV   DX,SERIAL+4   ;MODEM CONTROL REGISTER
                MOV   AL,OUT2_DTR   ;RTS-PIN OFF
                OUT   DX,AL
                POP   DS
                POP   SI
                POP   DX
                POP   CX
                POP   BX
                POP   AX
                RET
SEND_MENU       ENDP
;*******************************************************************************
;*********************       SUB ROUTINE  RECEIVER       **********************
;*******************************************************************************
RECEIVER        PROC  NEAR
                PUSH  AX
```

```
                PUSH  BX
                PUSH  CX
                PUSH  DX
                PUSH  SI
                LEA   BX,REC_BUFF
                MOV   AL,COUNT_DI
                MOV   AH,00H
                MOV   DI,AX
REC_CON_1:      MOV   DX,SERIAL+5     ;LINE STATUS REGISTER
REC_CON_2:      IN    AL,DX           ;INTERRUPT RESET CONTROL
                TEST  AL,01H          ;CHECK DATA READY ? ==> DR-BIT=1
                JZ    REC_CON_2
                MOV   DX,SERIAL       ;RECEIVER BUFFER REGISTER
                IN    AL,DX           ;READ DATA
                MOV   [BX+DI],AL      ;SAVE DATA IN BUFFER
                INC   DI
                INC   COUNT_DI
                CMP   AL,0DH          ;CHECK END TEXT
                JNZ   REC_CON_1
END_TEXT:       MOV   AL,COUNT_DI
                MOV   REC_COUN,AL     ;SAVE COUNT
                MOV   COUNT_DI,00H    ;CLEAR POINTER RECEIVE DATA
RETURN_REC:     POP   SI
                POP   DX
                POP   CX
                POP   BX
                POP   AX
                RET
RECEIVER        ENDP
;*******************************************************************************
```

```
;*****************************************************************************
;*********************    SUB ROUTINE CHECK RECEIVE DATA    *********************
;*********************  OUTPUT    AH                       *********************
;*****************************************************************************
      CHECK_DATA_REC  PROC  NEAR
                      PUSH  SI
                      LEA   BX,REC_BUFF    ;POINTER TO DATA BUFFER RECEIVE
                      MOV   SI,00H
                      MOV   AL,[BX+SI]     ;READ DATA FROM RECEIVE BUFFER
                      CMP   AL,31H         ;CHECK MENU 1 ?
                      JZ    DATA_1
                      CMP   AL,32H         ;CHECK MENU 2 ?
                      JZ    DATA_2
                      CMP   AL,33H         ;CHECK MENU 3 ?
                      JZ    DATA_3
                      CMP   AL,34H         ;CHECK MENU 4 ?
                      JZ    DATA_4
                      CMP   AL,35H         ;CHECK MENU 5 ?
                      JZ    DATA_5
                      MOV   AH,0FFH        ;RETURN AH = 0FFH ==> ERROR
                      JMP   END_CHECK_DATA
      DATA_1:         PUSH  BX
                      LEA   BX,DATA1_SEND  ;POINT TO DATA-BASE 1
                      CALL  TRANSMITTER    ;SEND DATA-BASE 1
                      MOV   AH,00H         ;RETURN AH = 00H ==>  OK
                      POP   BX
                      JMP   END_CHECK_DATA
      DATA_2:         PUSH  BX
                      LEA   BX,DATA2_SEND  ;POINT TO DATA-BASE 2
                      CALL  TRANSMITTER    ;SEND DATA-BASE 2
```

```
                MOV   AH,00H          ;RETURN AH = 00H ==>  OK
                POP   BX
                JMP   END_CHECK_DATA
DATA_3:         PUSH  BX
                LEA   BX,DATA3_SEND   ;POINT TO DATA-BASE 3
                CALL  TRANSMITTER     ;SEND DATA-BASE 3
                MOV   AH,00H          ;RETURN AH = 00H ==>  OK
                POP   BX
                JMP   END_CHECK_DATA
DATA_4:         PUSH  BX
                LEA   BX,DATA4_SEND   ;POINT TO DATA-BASE 4
                CALL  TRANSMITTER     ;SEND DATA-BASE 4
                MOV   AH,00H          ;RETURN AH = 00H ==>  OK
                POP   BX
                JMP   END_CHECK_DATA
DATA_5:         PUSH  BX
                LEA   BX,DATA5_SEND   ;POINT TO DATA-BASE 5
                CALL  TRANSMITTER     ;SEND DATA-BASE 5
                MOV   AH,00           ;RETURN AH = 00H ==>  OK
                POP   BX
END_CHECK_DATA: POP   SI
                RET
CHECK_DATA_REC  ENDP
```

```
;*****************************************************************************
;*********************      SUB ROUTINE TRANSMIT DATA      *********************
;*********************      INPUT     BX                   *********************
;*****************************************************************************
      TRANSMITTER     PROC  NEAR
                      PUSH  AX
                      PUSH  BX
                      PUSH  CX
                      PUSH  DX
                      PUSH  SI
                      MOV   DX,SERIAL+4     ;MODEM CONTROL REGISTER
                      MOV   AL,OUT2_DTR_RTS;RTS-PIN ON
                      OUT   DX,AL
                      MOV   SI,00H
                      MOV   DX,SERIAL+6     ;MODEM STATUS REGISTER
      CHECK_SEND1:    IN    AL,DX
                      TEST  AL,10H          ;TEST CTS-PIN ON ?
                      JZ    CHECK_SEND1
      SEND_DATA:      MOV   DX,SERIAL+5     ;LINE STATUS REGISTER
      CHECK_SEND2:    IN    AL,DX
                      TEST  AL,20H          ;TEST HOLDING REGISTER EMPTY ?
                      JZ    CHECK_SEND2
                      MOV   AL,[BX+SI]      ;GET DATA
                      MOV   DX,SERIAL       ;TRANSMITTER HOLDING REGISTER
                      OUT   DX,AL           ;SEND DATA
                      INC   SI
                      CMP   AL,ETX
                      JNZ   SEND_DATA
                      MOV   DX,SERIAL+4     ;MODEM CONTROL REGISTER
                      MOV   AL,OUT2_DTR     ;RTS-PIN OFF
```

```
                    OUT   DX,AL
                    POP   SI
                    POP   DX
                    POP   CX
                    POP   BX
                    POP   AX
                    RET
      TRANSMITTER   ENDP
;*******************************************************************************
CODE_SEG            ENDS
;###############################################################################
;!!!!!!!!!!!!!!!!!!!!        END INSTRUCTION SET        !!!!!!!!!!!!!!!!!!!!!!
;###############################################################################

;*******************************************************************************
;DATA AREA-1
   DATA1_SEG        SEGMENT PARA 'DATA'
      SERIAL        EQU   03F8H          ;PORT COM1
      OUT2_DTR      EQU   09H            ;OUT2-PIN,DTR-PIN ON
      OUT2_DTR_RTS  EQU   0BH            ;OUT2-PIN,DTR-PIN,RTS-PIN ON
      ATCOM1        DB    'AT E0 S7=50 S11=70 Q1 X4 S0=2^M'
      COUN_CHA      DB    31             ;COUNT ATCOM1
      AT_ONHOOK     DB    'AT H0'        ;AT-COMMAND ON-HOOK
      COUNT_ON      DB    5              ;COUNT CHARACTE AT_ONHOOK
      COUNT_SEND_P  DB    ?
      COUNT_ATCOM   DB    ?
      DATA1_SEG     ENDS
```

```
;*******************************************************************************
;DATA AREA-2
      DATA2_SEG       SEGMENT PARA 'DATA'
      DOSCALL         EQU   21H           ;DOS INTERRUPT
      STR_MAX         DB    50            ;MAX CHARACTER
      STR_REAL        DB    ?             ;ACTUAL CHARACTER
      STR_INP         DB    50 DUP(?)     ;BUFFER INPUT KEY
      REC_COUN        DB    ?
      REC_BUFF        DB    80 DUP(?)
      COUNT_DI        DB    0
      DATA2_SEG       ENDS
;*******************************************************************************
;DATA AREA-3
      DATA3_SEG       SEGMENT PARA 'DATA'
      D_PASSWORD      DB    "                                "
                      DB    "                                "
                      DB    "       ENTER CURRENT PASSWORD :  "
                      DB    "                                ",0DH
      TABLE_PASS      DB    "12345","72981","11111","22222","21332"
                      DB    "13125","25666","57657","65756","65776"
      MENU_ERROR      DB    "                                "
                      DB    "            MENU ERROR          "
                      DB    "                                ",0DH
      ANOTHER         DB    "                                "
                      DB    "     DO YOU WANT ANOTHER DATA ? "
                      DB    "     PLEASE  PRESS 0 ===>  YES  "
                      DB    "     PLEASE  ON-HOOK ===>  NO   "
                      DB    "                                ",0DH
      DATA3_SEG       ENDS
;*******************************************************************************
```

```
;DATA AREA-5

DATA5_SEG       SEGMENT PARA 'DATA'

DATA_MENU       DB "                                    "
                DB "                                    "
                DB "                                    "
                DB "              SAWADDEE              "
                DB "    INTERRACTIVE  VIEWDATA  SYSTEM   "
                DB "          BY  SOLIDY  #9            "
                DB "                                    "
                DB "    ********************************"
                DB "    ***     SYSTEM  MENU        ***"
                DB "    ********************************"
                DB "                                    "
                DB "      1) TRAIN TIME-TABLE           "
                DB "                                    "
                DB "      2) EDISON ELECTROMART         "
                DB "                                    "
                DB "      3) ..............             "
                DB "                                    "
                DB "      4) ..............             "
                DB "                                    "
                DB "      5) ..............             "
                DB "                                    "
                DB "          PLEASE SELECT MENU        "
                DB "                                    "
                DB "    ********************************"
                DB "                                    "

DATA5_SEG       ENDS

;*******************************************************************************
```

```
;*****************************************************************************
;DATA AREA-4
        DATA4_SEG       SEGMENT PARA 'DATA'
        ETX             EQU   0DH
        DATA1_SEND      DB  "                                    "
                        DB  "       INTERACTIVE VIEWDATA PROJECT "
                        DB  "                SOLIDY#9            "
                        DB  "          KMITL TRAIN TIME-TABLE    "
                        DB  "      ******************************* "
                        DB  "           BANGKOK-PRACHOMKLAO      "
                        DB  "      ******************************* "
                        DB  "      DEPT-ARRV TIME  TYPE  NO. PRICE "
                        DB  "      ******************************* "
                        DB  "      06.00-06.55 AM  DIES 109  6 Bth "
                        DB  "      06.20-07.12 AM  DIES 239  6 Bth "
                        DB  "      07.00-08.00 AM  DIES 151  6 Bth "
                        DB  "      08.05-08.55 AM  DIES 203  6 Bth "
                        DB  "      09.40-10.33 AM  DIES 183  6 Bth "
                        DB  "      10.40-11.27 AM  DIES 153  6 Bth "
                        DB  "      11.25-12.27 AM  NORM 251  6 Bth "
                        DB  "      01.10-01.55 PM  DIES 187  6 Bth "
                        DB  "      03.10-03.58 PM  DIES 155  6 Bth "
                        DB  "      03.20-04.17 PM  NORM 193  6 Bth "
                        DB  "      05.00-05.53 PM  NORM 201  6 Bth "
                        DB  "      05.25-06.20 PM  DIES 181  6 Bth "
                        DB  "      06.05-07.10 PM  NORM 173  6 Bth "
                        DB  "      ******************************* "
                        DB  "                                    "
                        DB  "                          [continue]"
                        DB  "                                    "
```

```
DB  "     ********************************   "
DB  "          PRACHOMKLAO-BANGKOK           "
DB  "     ********************************   "
DB  "     DEPT-ARRV TIME  TYPE  NO. PRICE    "
DB  "     ********************************   "
DB  "     06.17-07.30 AM  DIES  202 6 Bth     "
DB  "     07.01-07.55 AM  DIES  182 6 Bth     "
DB  "     08.08-09.05 AM  DIES  194 6 Bth     "
DB  "     09.01-10.10 AM  NORM  252 6 Bth     "
DB  "     09.56-10.35 AM  DIES  186 6 Bth     "
DB  "     10.59-11.50 AM  DIES  188 6 Bth     "
DB  "     01.13-02.00 PM  DIES  154 6 Bth     "
DB  "     03.11-04.05 PM  DIES  204 6 Bth     "
DB  "     04.04-04.55 PM  DIES  184 6 Bth     "
DB  "     04.25-05.20 PM  NORM  250 6 Bth     "
DB  "     04.40-06.00 PM  NORM  193 6 Bth     "
DB  "     06.20-07.10 PM  NORM  110 6 Bth     "
DB  "     07.20-08.15 PM  NORM  240 6 Bth     "
DB  "     ********************************   "
DB  "                                          "
DB  "     PRODUCE  BY:                         "
DB  "                                          "
DB  "           ,SERM...IWS...TUD              "
DB  "                                          "
DB  "     ********************************   ",ETX
```

```
DATA2_SEND      DB  "  ************************************ "
                DB  "  *         EDISON ELECTROMART       * "
                DB  "  ************************************ "
                DB  "  * TYPE      DESCRIPTION       PRICE* "
                DB  "  ************************************ "
                DB  "  * T.V.C   TANIN 14-932AVR    6,450-* "
                DB  "  *           - 14'                  * "
                DB  "  *           - Remote Control       * "
                DB  "  *           - AV-Auto Tuning       * "
                DB  "  *----------------------------------* "
                DB  "  * T.V.C   SHARP 2171        11,400-* "
                DB  "  *           - 21'                  * "
                DB  "  *           - Remote Control       * "
                DB  "  *           - AV-IN-OUT            * "
                DB  "  *             Multi System         * "
                DB  "  *----------------------------------* "
                DB  "  * T.V.C   SAMPO XT-6698      25,800* "
                DB  "  *             - 25'                * "
                DB  "  *             - Remote Control     * "
                DB  "  *             - Stereo AV-Multi    * "
                DB  "  *               System             * "
                DB  "  ************************************ "
                DB  "                        ------------  "
                DB  "                        ¦ Next Page ¦ "
                DB  "                        ------------  "
```

```
DB "   ************************************ "
DB "   *        EDISON ELECTROMART        * "
DB "   ************************************ "
DB "   * TYPE      DESCRIPTION      PRICE* "
DB "   ************************************ "
DB "   * V.D.O   MITSUBISHI M-16   13,500-* "
DB "   *          - Able to record        * "
DB "   *          - Remote Control        * "
DB "   *----------------------------------* "
DB "   * V.D.O   THOMSON T-3        6,150-* "
DB "   *          - Play Only             * "
DB "   *          - Remote Control        * "
DB "   *----------------------------------* "
DB "   * V.D.O   SHARP 6V3          6,400-* "
DB "   *          - Play Only             * "
DB "   *          - Remote Control        * "
DB "   *----------------------------------* "
DB "   * V.D.O   TOSHIBA C-2        6,850-* "
DB "   *          - Play Only             * "
DB "   *          - Remote Control        * "
DB "   ************************************ "
DB "                          ------------  "
DB "                          ¦ Next Page ¦  "
DB "                          ------------  "
DB "                                        "
```

```
                DB  "   ************************************ "
                DB  "   *         EDISON ELECTROMART       * "
                DB  "   ************************************ "
                DB  "   * TYPE      DESCRIPTION       PRICE* "
                DB  "   ************************************ "
                DB  "   * RADIO   SAMPO AL-9750       3,500* "
                DB  "   *          - 280 W                 * "
                DB  "   *          - KARAOKE               * "
                DB  "   *          - 2 Way Speaker         * "
                DB  "   *            System                * "
                DB  "   *----------------------------------* "
                DB  "   * RADIO   PHILIPS FW-21      9,900-* "
                DB  "   *          - 300 W                 * "
                DB  "   *          - Remote Control        * "
                DB  "   *          - Digital Tuner         * "
                DB  "   *          - Bitstream CD          * "
                DB  "   *            player                * "
                DB  "   *----------------------------------* "
                DB  "   * RADIO   AUDIO LINE AL-739  2,300-* "
                DB  "   *          - 200 W                 * "
                DB  "   *          - Auto Reverse          * "
                DB  "   *          - 2 Way Speaker         * "
                DB  "   *            System                * "
                DB  "   ************************************ "
                DB  "                                        "
DATA3_SEND      DB  "PLE",ETX
DATA4_SEND      DB  "PHYSICS",ETX
DATA5_SEND      DB  "SCIENCE",ETX
DATA4_SEG       ENDS
;*******************************************************************************
```

```
;*******************************************************************************
;STACK AREA
      STACK_SEG       SEGMENT PARA STACK 'STACK'
                      DB    200 DUP (?)
      STACK_SEG       ENDS
;*******************************************************************************

                      END
;-------------------------------------------------------------------------------
```

ประวัติผู้เขียน

นายเสริมชัย เกียรติยศนากุล เกิดวันที่ 17 เมษายน 2514 ใช้ชีวิตวัยเด็กที่มหาสารคาม เข้ารับการศึกษาในโรงเรียนวาปีปทุม จังหวัดมหาสารคาม จนจบการศึกษาระดับมัธยมต้นและย้ายที่อยู่ตามครอบครัวมาใช้ชีวิตวัยรุ่นที่จังหวัดสมุทรปราการ เข้าศึกษาต่อระดับมัธยมศึกษาตอนปลายที่โรงเรียนด่านสำโรง จังหวัดสมุทรปราการ ปัจจุบันอาศัยอยู่รัชดา กรุงเทพฯ

ในปี 2533 ได้ผ่านการสอบคัดเลือกของทบวงมหาวิทยาลัยโดยไม่ยากลำบากนัก ได้รับการศึกษาระดับปริญญาตรี ที่ภาควิชาฟิสิกส์ประยุกต์ คณะวิทยาศาสตร์ สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหาร ลาดกระบัง ขณะที่กำลังศึกษาอยู่ได้รับความเชื่อถือจากเพื่อน ๆ ในด้านความรู้ทางวิชาการอย่างสูง สำเร็จการศึกษาในระดับปริญญาตรีในปีการศึกษา 2536 ปัจจุบันได้ร่วมกับเพื่อนอีกสองคนทำโครงงานพิเศษเรื่อง Interactive Viewdata

ประวัติการฝึกงาน

ได้รับการฝึกงานที่ สถาบันวิจัยวิทยาศาสตร์ และเทคโนโลยีแห่งประเทศไทย ในระหว่างวันที่ 1เมษายน - 31 พฤษภาคม 2536 ในเรื่องดิจิตอลอิเล็กทรอนิกส์และไมโครโปรเซสเซอร์

นายโรจฤทธิ์ สว่างแจ้ง เกิดเมื่อวันที่ 16 กรกฎาคม 2515 ที่โรงพยาบาลมิชชันนครปฐม เข้าศึกษาจนจบประถมศึกษาปีที่ 6 ที่โรงเรียนยอแซฟอุปถัมภ์ จังหวัดนครปฐม เนื่องจากมีการย้ายถิ่นที่อยู่มากรุงเทพฯ ได้เข้าศึกษาในระดับมัธยมศึกษาที่โรงเรียนวัดบวรมงคล บางพลัด กรุงเทพฯ จนจบระดับมัธยมศึกษาตอนปลาย และได้พยายามอ่านหนังสือ จนสามารถผ่านการสอบคัดเลือกเข้าศึกษา ของทบวงมหาวิทยาลัย ได้ศึกษาระดับปริญญาตรีที่ ภาควิชาฟิสิกส์ประยุกต์ คณะวิทยาศาสตร์ สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหาร ลาดกระบัง มีโอกาสได้เข้าร่วมกิจกรรมของทางคณะ ฯ หลายครั้ง และได้พยายามตั้งใจเรียนจนคาดว่าจะสามารถจบการศึกษาในระดับปริญญาตรีในปีการศึกษา 2536 ปัจจุบันได้ร่วมกับเพื่อนอีกสองคนทำโครงพิเศษในเรื่อง Interactive Viewdata

ประวัติการฝึกงาน

ได้ผ่านการฝึกงานที่ สำนักงานพลังงานปรมาณูเพื่อสันติ ในฝ่ายอุปกรณ์อิเล็กทรอนิกส์ แผนกคอมพิวเตอร์ ระหว่างวันที่ 1 เมษายน - 31 พฤษภาคม 2536 ได้รับการอบรม และค้นคว้าในเรื่อง ฮาร์ดแวร์ ของเครื่อง IBM PC

นางสาว อัมพร โชคชัยตระกูลโพธิ์ เกิดเมื่อวันที่ 7 มิถุนายน 2515 ที่โรงพยาบาลจุฬาลงกรณ์ กรุงเทพฯ มีภูมิลำเนาอยู่ที่จังหวัดฉะเชิงเทรา การศึกษาเริ่มแรกได้เรียนกับครูข้างบ้าน หลังจากจบหลักสูตรแล้วจึงสมัครเข้าเรียนที่โรงเรียนวัดแสนภูดาษ ได้รับการเลื่อนขั้นชั้นประถมศึกษาปีที่ 2 ทันที จากนั้นเข้ารับการศึกษาต่อในระดับมัธยมศึกษาตอนต้น และตอนปลายที่โรงเรียนเบญจมราชรังสฤษฎิ์ จังหวัดฉะเชิงเทรา ด้วยผลการเรียนที่อยู่ในเกณฑ์ดี จึงได้รับคัดเลือกจากการสอบโควต้าของคณะวิทยาศาสตร์ สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหาร ลาดกระบัง โดยเลือกเรียนในภาควิชาฟิสิกส์ประยุกต์ ขณะศึกษาได้ทำกิจกรรมของทางคณะ ฯ มากมาย ปัจจุบันได้ร่วมกับเพื่อนอีกสองคนจัดทำโครงการพิเศษเรื่อง Interactive Viewdata

ประวัติการฝึกงาน

ได้รับการฝึกงานที่บริษัท PTR-HITECH ระหว่างวันที่ 1 เมษายน - 15 พฤษภาคม 2536 โดยมีการอบรมเรื่อง PLC (Programmable Logic Controller)

เอกสารอ้างอิง

1. ยุทธนา ลีลาศวัฒนกุล, สมชาย วิวัฒนศานต์ "วีดีโอมอดูเลเตอร์" เซมิคอนดักเตอร์อิเลคทรอนิคส์, 104 (2534) : 226-230.
2. "IC น่าสนใจ LM1889" เซมิคอนดักเตอร์อิเลคทรอนิคส์,104 (2534) : 316-319.
3. ยืน ภู่วรวรรณ เทคนิคการประยุกต์และใช้งานไอซีทีทีแอล, บริษัท ซีเอ็ดยูเคชัน, พิมพ์ครั้งที่ 1, หน้า 157-185, พ.ศ. 2521.
4. ยืน ภู่วรวรรณ, วัฒนา เชี่ยงกูล ไมโครโปรเซสเซอร์ ไมโครคอมพิวเตอร์, บริษัท ซีเอ็ดยูเคชัน, พิมพ์ครั้งที่ 8, พ.ศ. 2532.
5. ยืน ภู่วรวรรณ ทฤษฎีและการประยุกต์ไมโครโปรเซสเซอร์ Z-80, บริษัท ซีเอ็ดยูเคชัน, หน้า 112-136, พ.ศ. 2533.
6. ยืน ภู่วรวรรณ เทคโนโลยีฮาร์ดแวร์ IBM PC, บริษัท ซีเอ็ดยูเคชัน, หน้า 97-111, 126-148, พ.ศ. 2533.
7. ดร. ศิริวรรณ ฉันทาดิศัย หลักการเขียนโปรแกรม ภาษาแอสเซมบลี 8088, พิมพ์ครั้งที่ 2, สำนักพิมพ์ประกายพรึก, พ.ศ. 2534.
8. คู่มือเทียบเบอร์ไอซี TTL, บริษัท ซีเอ็ดยูเคชัน, พ.ศ. 2534.
9. ดร. ธวัช เมตสวรรค์, โยชิคะซึ ชาวามูระ เทคนิคการซ่อมเครื่องรับโทรทัศน์สี , พิมพ์ครั้งที่ 4, สำนักพิมพ์ดวงกมล, พ.ศ. 2528.
10. บัณฑิต จามารภูติ ฮาร์แวร์ไมโครคอมพิวเตอร์ 8088,80286,80386, บริษัท ซีเอ็ดยูเคชัน, หน้า 71-79.
11. เจน สงสมพันธุ์ เทคโนโลยีโทรทัศน์, โรงเรียน อิเลคทรอนิกส์กรุงเทพ.
12. น.ต. ดร.ไพศาล สงวนหมู่,รศ.ยืน ภู่วรวรรณ การสื่อสารข้อมูลและไมโครคอมพิวเตอร์เนตเวอร์ค, บริษัทซีเอ็ดยูเคชัน.
13. ชูชัย ธนสารตั้งเจริญ, ทินกร ดุ๊ก การสื่อสารข้อมูล, พิมพ์ครั้งที่ 2, หน้า 137-152, หจก. สำนักพิมพ์ฟิสิกส์เซ็นเตอร์.
14. สุพจน์ ปุณณชัยยะ MODEM, พิมพ์ครั้งที่ 3, หน้า 69-78, บริษัท อินเฟอร์เมติกบิซิเนส พับลิเคชัน จำกัด, พ.ศ 2535
15. Kane, Gerry CRT Controller Handbook, pp. 4.1-4.41.
16. IBM Personal Computer XT System Technical Reference, pp.75-81, 185-213.
17. Zilog Microprocessor Data Book Technical Manual Z80180, 1988.
18. Paul Bates, P. Practical Digital And Data Communications with LSI Applications, pp. 83-105, Prentice-Hall,INC., Engle wood cliffs, New Jersey, 1987.

19. Discovery 2400 User Guide, Datatronic Technology,INC., Taipei, Taiwan.

20. Uffenbeck, John. The 8086/8088 Family : Design,Programming And Interfacing, Prentice Hall,INC., New Jersey, 1987.

21. Brenner, Robert C. IBM PC Advanced Troubleshooting & Repair, W.Sam & Company, Macmillan,INC., Indiana, 1989.