คีย์บอร์ดคอมพิวเตอร์ไร้สาย
WIRELESS COMPUTER KEYBOARD

โดย

น.ส. สุวิทา อังคนาสายัณห์ รหัส 36014526
น.ส. อุไร ศรีเหรา รหัส 36014575

อาจารย์ที่ปรึกษา

รศ.ดร. วิวัฒน์ กิรานนท์

อ. วิภา แสงพิสิทธิ์

ปริญญานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิศวกรรมศาสตรบัณฑิต
สาขาวิชาวิศวกรรมโทรคมนาคม
คณะวิศวกรรมศาสตร์
สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง
ปีการศึกษา 2539

ปริญญานิพนธ์ปีการศึกษา 2539

ภาควิชา วิศวกรรมโทรคมนาคม

คณะ วิศวกรรมศาสตร์ สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

เรื่อง คีย์บอร์ดคอมพิวเตอร์ไร้สาย

Wireless Keyboardcomputer

ผู้จัดทำ

1. นางสาว สุวิทา อังคนาสายัณห์ รหัส 36014526
2. นางสาว อุไร ศรีเหรา รหัส 36014575

_____อาจารย์ที่ปรึกษา

(รศ. ดร. วิวัฒน์ กิรานนท์)

_____อาจารย์ที่ปรึกษาร่วม

(อาจารย์ วิภา แสงพิสิทธิ์)

คีย์บอร์ดคอมพิวเตอร์ไร้สาย

(Wireless Computer Keyboard)

บทคัดย่อ

ปริญญานิพนธ์ฉบับนี้เป็นการสร้างคีย์บอร์ดคอมพิวเตอร์แบบไร้สาย ซึ่งสามารถใช้ประโยชน์ได้หลายด้าน เช่น การประชุมแสดงผลผ่านทางจอภาพ โดยผู้ป้อนข้อมูลจะป้อนข้อมูลผ่านทางคีย์บอร์ด ซึ่งจะอยู่ที่ตำแหน่งใดของห้องก็ได้ และแสดงผลผ่านจอภาพขนาดใหญ่ หรือจะแสดงผลผ่านออกจอแสดงผลทุกจอภาพ โดยใช้คีย์บอร์ดเพียงตัวเดียว โดยการต่อโมเด็มเครื่องรับไว้กับเครื่องคอมพิวเตอร์ทุกเครื่องที่จะแสดงผลก็สามารถทำได้ โครงงานนี้เป็นการส่งข้อมูลด้วยความถี่วิทยุย่าน ยูเฮชเอฟ (UHF) ที่ความถี่ประมาณ 300-400 MHz ซึ่งมีการรบกวนจากภายนอกค่อนข้างน้อย และเป็นการโมดูเลท แบบเอเอสเค (amplitude shift keying ; ASK) ส่วนในภาครับก็จะเป็นภาครับแบบเอเอสเคเช่นเดียวกัน และผ่านวงจรปรับแต่งสัญญาณ เพื่อให้ได้รูปสัญญาณที่ต้องการ

ABSTRACT

This thesis concerns about construction of a wireless computer keyboard, it is available in many ways as monitor display conference. The users press the keyboard in any where. It can display in large monitor or represent for all terminal where it is needed . The wireless computer keyboard will be operate by connected all of computer with modem . It will be transmitted the data by using the frequency range 300-400 MHz UHF radio frequency that nearly no noise effect. The ASK technique is use as modulation.

สารบัญ

สารบัญตาราง

สารบัญรูป

สารบัญรูป

บทที่ 1
บทนำ

คีย์บอร์ดไร้สาย ใช้หลักการของการสื่อสารข้อมูลซึ่งเป็นการส่งข้อมูลที่อยู่ในรูปแบบอนุกรมโดยแปลงสัญญาณจากคีย์บอร์ด ซึ่งคีย์บอร์ดสร้างขึ้นมาโดยใช้หลักการ สแกนคีย์ โดยมีซีพียู 8048 ซึ่งอยู่ในตัวคีย์บอร์ดเป็นตัวจัดการเกี่ยวกับส่วนนี้ โดยมันจะสร้างสัญญาณนาฬิกาและข้อมูล ซึ่งสัญญาณทั้งสองจะส่งออกมาซิงโครไนซ์กัน เราก็เอาสัญญาณนาฬิกาและข้อมูล ไปแปลงเป็นสัญญาณแบบอะซิงโครนัส (asynchronous) ช่องเดียว โดยมีซีพียู 8031 เป็นหัวใจหลักในการจัดการในส่วนนี้ เมื่อได้รับสัญญาณพร้อมที่จะส่งแล้ว ก็ส่งผ่านวงจรโมดูเลท ส่งออกอากาศในย่านยูเฮชเอฟ ช่วง 300-400 MHz โดยใช้เทคนิคการส่งแบบ เอเอสเค (Amplitude shift keying) ผ่านออกอากาศไปยังภาครับ แล้วแปลงสัญญาณจากที่รับมาให้เป็นสัญญาณนาฬิกาและข้อมูล จะซิงโครไนซ์กันออกมาตามเดิม โดยใช้ซีพียู 8031 เป็นตัวจัดการในส่วนนี้ จากนั้นก็ต่อเข้าระบบของเมนบอร์ด (mainboard) ใช้งานต่อไป และในระบบของคีย์บอร์ดรุ่นใหม่ คีย์บอร์ดไม่ใช่อินพุทของคอมพิวเตอร์อย่างเดียว แต่คีย์บอร์ดจะเป็นเอาท์พุทด้วยคือ จะมีสัญญาณตอบรับกลับมายังคีย์บอร์ดด้วย โดยสัญญาณที่ตอบรับกลับมานี้จะทำหน้าที่บอกคีย์บอร์ดอยู่ในสถานะใด เช่น NUM LOCK , CAPS LOCK SCROLL LOCK เป็นต้น โดยจะแสดงเป็นแอลอีดีบนคีย์บอร์ด แต่สัญญาณที่ตอบรับมานี้เราไม่จำเป็นต้องส่งก็ได้ เพราะคอมพิวเตอร์จะทราบเองว่าอยู่ในสถานะใด เพียงตอบกลับมาแสดงผลที่แอลอีดีบนคีย์บอร์ดเท่านั้น เพราะฉะนั้นเราจึงไม่สนใจไฟที่แสดงผลบนแอลอีดี และส่งข้อมูลเพียงทิศทางเดียวก็เพียงพอ และเนื้อหาในบทต่างๆ จะพูดถึงทฤษฎีที่เกี่ยวข้องคือ เรื่องของคีย์บอร์ดโดยละเอียด และจากนั้นเป็นเรื่องของดิจิตอลโมดูเลชั่น (digital mofulation) ซึ่งเป็นพื้นฐานสำคัญในการส่งสัญญาณแบบดิจิตอล ต่อมาจะเป็นเรื่องของวงจรออสซิลเลเตอร์ (oscillator) หลักการคำนวณออกแบบวงจร และบทต่อมาจะพูดถึงเรื่องการสื่อสารข้อมูลในรูปแบบต่างๆ และเรื่องสุดท้ายเป็นเรื่องของวงจรการทดลอง ผลการทดลอง และบทสรุป

บทที่2

ทฤษฎีที่เกี่ยวข้อง

2.1 คีย์บอร์ด

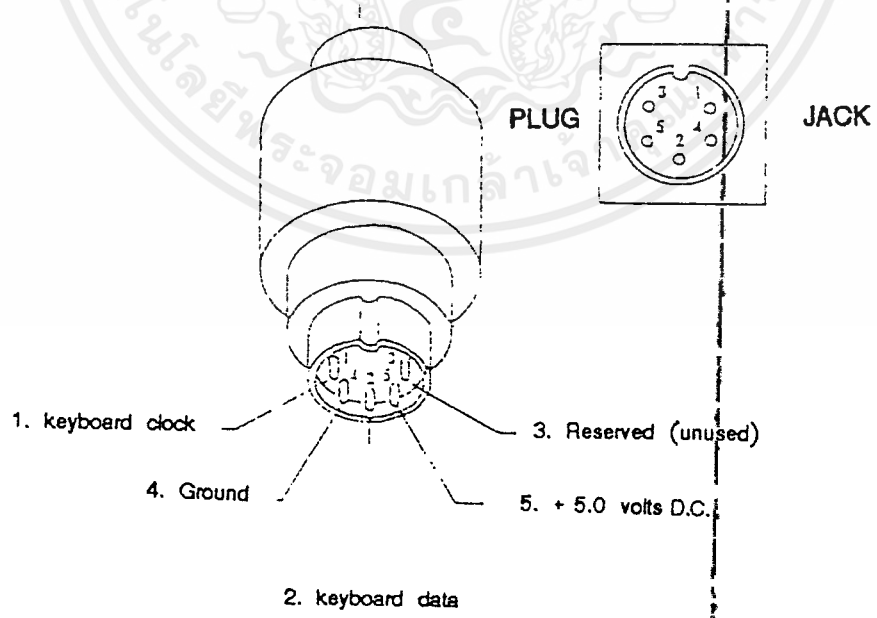
คีย์บอร์ด เป็นอุปกรณ์ที่เชื่อมต่อกับระบบคอมพิวเตอร์ซึ่งทำหน้าที่เป็นอินพุทให้กับระบบเพื่อให้คอมพิวเตอร์สามารถรับคำสั่งและข้อมูลเข้าไปประมวลผลแล้วทำตามขั้นตอนของระบบต่อไป

2.1.1 โครงสร้างของคีย์บอร์ด

โครงสร้างของคีย์บอร์ดโดยทั่วไปจะประกอบไปด้วยแป้นพิมพ์ซึ่งมีประมาณ 101 คีย์ และวงจรภายในประกอบด้วยไมโครโปรเซสเซอร์ โดยมากจะใช้เบอร์ 8048 ซึ่งทำหน้าที่สแกนคีย์บอร์ด และส่งค่าสแกนคีย์บอร์ดมายังเมนบอร์ดของคอมพิวเตอร์ โดยมีขั้วต่อมายังเมนบอร์ดจำนวน 5 เส้น ซึ่งประกอบไปด้วย

1) สายของข้อมูล (data)
2) สัญญาณนาฬิกา (clock)
3) ไฟเลี้ยงขั้วบวก (+5 Volt)
4) กราวด์ (ground)
5) ขาว่าง (nc)

ลักษณะของหัวต่อจะเป็นหัวต่อแบบดีน (din) โดยมีไดอะแกรมของหัวต่อดังรูปที่ 2.1

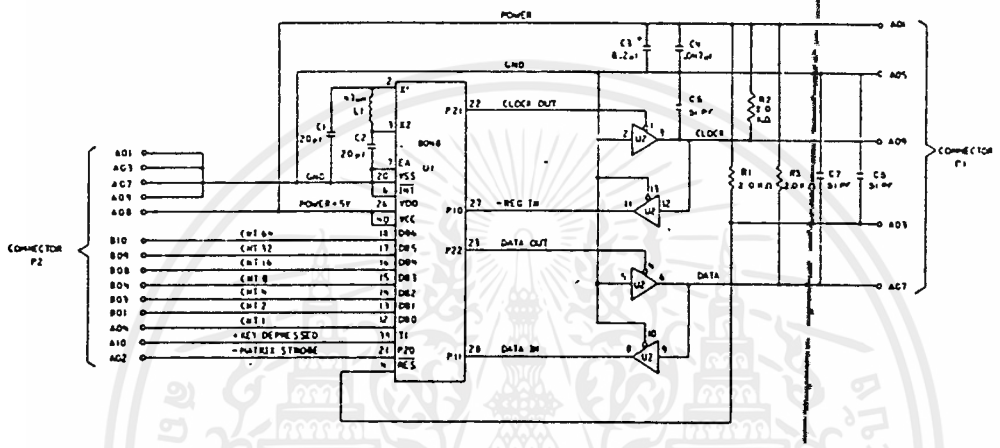


รูปที่ 2.1 หัวต่อแบบดีน (din)

2.1.2 การทำงานของคีย์บอร์ด

บนคีย์บอร์ดจะมีชีพียู 8048 1 ตัว ทำหน้าที่ในการสแกนตรวจสอบการกดคีย์ ชีพียูที่อยู่บนคีย์บอร์ดนี้ จะตรวจสอบการกดคีย์ เมื่อมีการกดคีย์ก็จะรับรู้และส่งข้อมูลออกตามรหัสของคีย์นั้นๆ ซึ่งเราเรียกว่า สแกนคีย์

โครงสร้างการต่อคีย์บอร์ดจะวางเป็นเมตริกซ์มี 23 แถว 4 คอลัมน์ โครงสร้างของชีพียู 8048 แสดงได้ดังรูปที่ 2.2



รูปที่ 2.2 วงจร 8048 ที่อยู่บนคีย์บอร์ด

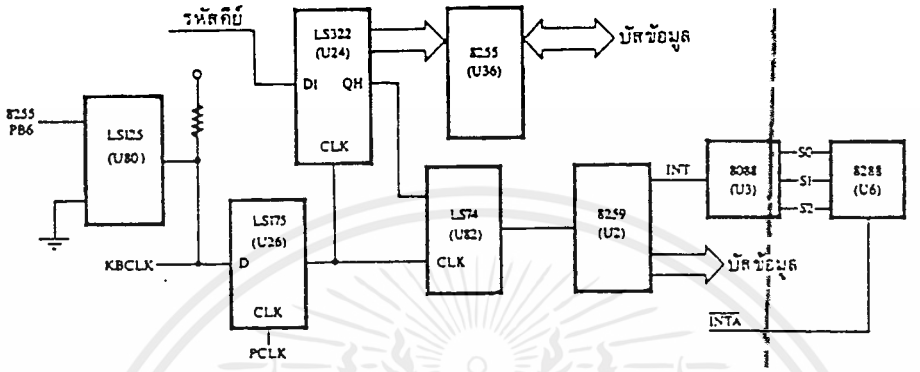
สังเกตว่าชีพียู 8048 จะอ่านการกดคีย์ การตรวจสอบคีย์ หรือแม้แต่การตรวจสอบข้อผิดพลาด แล้วส่งต่อให้เมนบอร์ดต่อไป จะเห็นว่าค่ารหัสที่ได้จากการกดคีย์ยังไม่ใช่รหัสแอสกี (ASCII) เช่น ถ้ากดคีย์ ESC ซึ่งมีค่าคีย์เป็น 1 จะได้รหัสสแกนเป็น 0000001 ถ้ากดคีย์ BREAK (numlock) จะได้รหัสสแกน 1000101 หรือรหัส 45H

อนึ่งค่าที่ได้จากการสแกนคีย์เป็นค่า 7 บิท แต่ 8048 จะเพิ่มบิท 8 ให้เป็น 1 เช่น ถ้ามีการกดคีย์ "P" 8048 จะอ่านค่าได้ 19H เป็นเลขไบนารี่ 0011001 แต่ 8048 จะเพิ่ม 1 ให้อีกตัว คือ 10011001 หรือกลายเป็นรหัส 99H การส่งข้อมูลเป็นการส่งแบบอนุกรม ดังนั้นถ้าไม่มีข้อมูลสายข้อมูลที่ส่งมายังเมนบอร์ดจะเป็น "0" และหากเมื่อ 8048 ต้องการส่งข้อมูลมันจะส่งรหัส "1" นำไปก่อนประมาณ 2 มิลลิวินาที แล้วจึงตามด้วยข้อมูล บิท 1 ที่นำหน้าจึงถือว่าเป็นเริ่มต้น

2.1.3 โครงสร้างการทำงานบนเมนบอร์ด

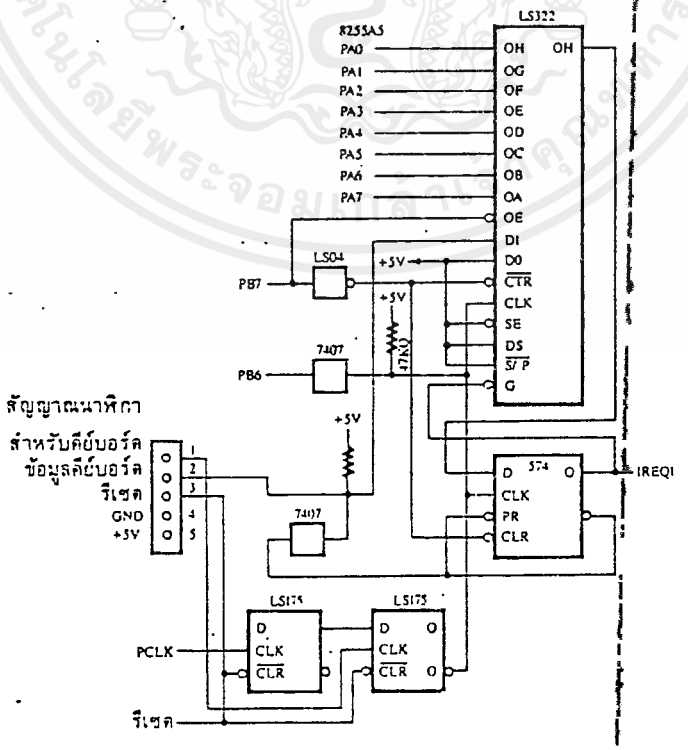
ปกติเมื่อมีการส่งค่ารหัสสแกนคีย์มา รหัสสแกนคีย์นี้มาแบบอนุกรมเข้ามายังไอซี 74LS322 เพื่อเปลี่ยนข้อมูลให้เป็นแบบขนาน เมื่อข้อมูลเข้ามาครบชุดทั้ง 8 บิทแล้ว สัญญาณบิทสูงสุดซึ่งเป็น "1" จะเลยไปเซ็ตฟลิปฟลอปเพื่อส่ง IRQ1 ไปยัง 8259 เมื่อ 8259 ได้รับ IRQ1 ก็จะส่ง INT มาให้กับ 8088 ชีพียูรับ INT ก็จะทำงานตามโปรแกรมไบออสที่วางไว้ โดยการอ่านข้อมูลจากพอร์ท

8255 ในพอร์ท A เมื่อได้ข้อมูลแล้วก็จะ นำมาเก็บไว้บัฟเฟอร์และส่งเอาท์พุทผ่าน 8255 มาเคลียร์ข้อมูลในรีจิสเตอร์ของ LS322 เพื่อรอรับตัวอักษรตัวใหม่ โครงสร้างของการเชื่อมโยงระหว่าง 8259 กับ 8255 แสดงได้ดังไดอะแกรมรูปที่ 2.3



รูปที่ 2.3 ไดอะแกรมการทำงานของ 8259,8255 และรีจิสเตอร์รับข้อมูลคีย์บอร์ด 74LS322

หากจะขยายวงจรส่วนเชื่อมต่อระหว่างหัวต่อ DIN ที่เชื่อมต่อคีย์บอร์ดกับเมนบอร์ดดู เราจะรับสัญญาณโดยมีฟลิปฟลอป 74LS175 เป็นตัวรับสัญญาณนาฬิกา และ 74LS74 เป็นตัวส่งสัญญาณ IRQ1 ดังรูปที่ 2.4



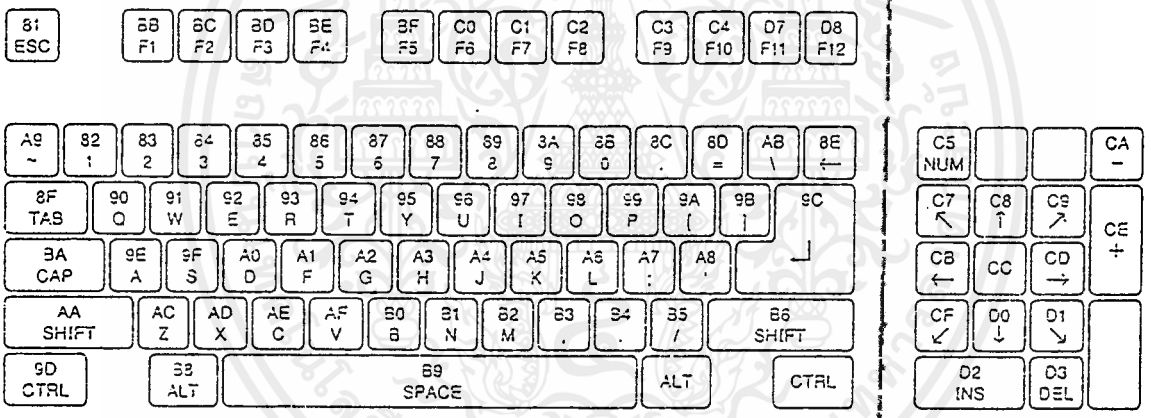
รูปที่ 2.4 วงจรเปลี่ยนข้อมูลจากอนุกรมเป็นขนาน

ปกติสัญญาณ KBDCLK เป็นสัญญาณชิงโครนัสกับการส่งข้อมูลมาจาก 8048 สัญญาณนี้จะได้รับการหน่วงไว้ และทำการทริก 74LS175 เพื่อสร้างสัญญาณให้กับ 74LS322 เพื่อเปลี่ยนข้อมูลอนุกรมเป็นขนาน เมื่อบิทสุดท้ายของรหัสการสแกนส่งครบทั้ง 8 บิทแล้ว ก็จะส่งสัญญาณออกไปยัง 74LS74 เพื่อสร้างสัญญาณ IRQ1 โปรแกรมไบออสจะวางไว้ตรงอินเตอรัพต์คือ รับตามอินเตอรัพต์ INT 9H ตามฮาร์ดแวร์ เพื่ออ่านรหัสสแกนแล้วส่งต่อให้ ซอฟต์แวร์ในโมดูล INT 16H ต่อไป

2.1.4 ค่ารหัสสแกน

ค่ารหัสสแกนคีย์ คือ ค่าจากตำแหน่งที่ถูกกดแล้ว ซีพียู 8048 แปลความหมายเป็นข้อมูลชุดหนึ่ง ซึ่งเรียกข้อมูลของแต่ละชุดนี้ว่า ค่ารหัสสแกน

ตัวคีย์บอร์ดที่ใช้ในโครงงานนี้ เป็นคีย์บอร์ดแบบ XT โดยมีตำแหน่ง ของคีย์และค่ารหัสสแกนคีย์แสดงผลเป็นเลขฐาน 16 ดังรูปที่ 2.5 และตารางที่ 2.1



รูปที่ 2.5 แสดงตำแหน่งคีย์ต่างๆ บนคีย์บอร์ดและรหัสสแกนคีย์

ตำแหน่งคีย์	ค่ารหัสสแกน (เลขฐานสิบหก)	ตำแหน่งคีย์	ค่ารหัสสแกน (เลขฐานสิบหก)	ตำแหน่งคีย์	ค่ารหัสสแกน (เลขฐานสิบหก)
1	01	29	1D	57	39
2	02	30	1E	58	3A
3	03	31	1F	59	3B
4	04	32	20	60	3C
5	05	33	21	61	3D
6	06	34	22	62	3E
7	07	35	23	63	3F
8	08	36	24	64	40
9	09	37	25	65	41
10	0A	38	26	66	42
11	0B	39	27	67	43
12	0C	40	28	68	44
13	0D	41	29	69	45
14	0E	42	2A	71	46
15	0F	43	2B	71	47
16	10	44	2C	72	48
17	11	45	2D	73	49
18	12	46	2E	74	4A
19	13	47	2F	75	4B
20	14	48	30	76	4C
21	15	49	31	77	4D
22	16	50	32	78	4E
23	17	51	33	79	4F
24	18	52	34	80	50
25	19	53	35	81	51
26	1A	54	36	82	52
27	1B	55	37	83	53
28	1C	56	38		

ตารางที่ 2.1 รหัสสแกนของแต่ละคีย์

2.2 การทำงานพื้นฐานของแป้นพิมพ์

ก่อนที่เราจะศึกษาถึงการทำงานของแป้นพิมพ์เราต้องทราบก่อนว่า การทำงานของมันมิได้เป็นดังเช่นที่ดูเหมือนจะเป็นเช่นเมื่อเรากดปุ่มใดๆ บนแป้นพิมพ์แล้วมิได้หมายถึงการส่งตัวอักษรบนปุ่มนั้นเข้าสู่เครื่องคอมพิวเตอร์โดยทันที เช่น เมื่อเรากดปุ่ม A ก็จะมิได้หมายถึงตัวอักษร "A"ในทันทีแต่การทำงานของมันจะผ่านกลไกหลายขั้นตอน ในหัวข้อนี้เราจะได้ศึกษาถึงขั้นตอนการทำงานดังกล่าวและเหตุผลเบื้องหลังการออกแบบการทำงานในลักษณะนี้ เหตุผลที่แป้นพิมพ์ไม่ส่งอักขระของปุ่มที่กดเข้าสู่เครื่องโดยตรง ก็เพราะ

1. ถึงแม้ว่าฮาร์ดแวร์ของคอมพิวเตอร์จะมีประสิทธิภาพสูงมาก แต่มันไม่อยู่ในสถานะที่จะทำงานต่างๆ เองได้ มันไม่มีหน้าที่ในการให้ความหมายสิ่งต่างๆ แก่มัน ดังนั้นในกรณีที่กล่าวว่าแป้นพิมพ์เป็นตัวให้ความหมายว่า เมื่อกดปุ่ม A จะหมายถึงตัวอักษร "A" ย่อมไม่เป็นความจริงเพราะแป้นพิมพ์เป็นฮาร์ดแวร์ย่อมไม่สามารถทำหน้าที่ของซอฟแวร์ได้

2. จะทำให้การทำงานไม่ยืดหยุ่น เนื่องจากมันไม่ใช่ประเด็นที่สำคัญว่าการกดปุ่ม A แล้วจะต้องหมายถึงตัวอักษร A เท่านั้น แต่สิ่งที่สำคัญสำหรับคอมพิวเตอร์คือ ควรจะมีความยืดหยุ่นสามารถเปลี่ยนแปลงได้ดังนั้นจึงเป็นการดี ถ้าฮาร์ดแวร์ของคอมพิวเตอร์จะไม่บังคับเจาะจงความหมายในการกดปุ่มใดๆ

เมื่อมีการกดปุ่มใดปุ่มหนึ่งบนแป้นพิมพ์จะเกิดขั้นตอนการทำงานดังนี้คือ แป้นพิมพ์จะรับรู้ว่าการกดปุ่มเกิดขึ้นและทำการบันทึกรหัสสแกนไว้ ปุ่มแต่ละปุ่มจะมีรหัสเฉพาะเรียกว่า รหัสสแกน (scan code) ดังรูปที่ 2.5 ได้แสดงรูปร่างของคีย์บอร์ดมาตรฐานของพีซีและรหัสสแกนเอาไว้

หลังจากนั้นแป้นพิมพ์จะส่งอินเทอรัพต์หมายเลข 9 ให้แก่โปรเซสเซอร์เพื่อบอกกับคอมพิวเตอร์ว่าขณะนั้นได้มีการกดปุ่มบนแป้นพิมพ์เกิดขึ้น ซึ่งก็จะทำให้โปรเซสเซอร์หยุดค้างการทำงานใดๆที่กำลังทำอยู่ไว้ชั่วคราว เพื่อกระโดดไปยังโปรแกรมที่ดูแลจัดการอินเทอรัพต์เสียก่อน อันได้แก่ส่วนของรอมไบออสนั่นเอง

ในจุดนี้รอมไบออสในส่วนที่ดูแลการอินเทอรัพต์จากแป้นพิมพ์ จะเข้ามาทำงานด้วยการดูว่าเกิดอะไรขึ้นที่แป้นพิมพ์ โดยส่งคำสั่งไปยังแป้นพิมพ์เพื่อถามว่าเกิดอะไรขึ้น แป้นพิมพ์จะตอบกลับโดยบอกกับไบออสว่าปุ่มใดถูกกด ซึ่งการโต้ตอบเหล่านี้จะดำเนินผ่านพอร์ทที่แป้นพิมพ์ใช้อยู่ จากนั้นแป้นพิมพ์จะโต้ตอบส่งรหัสสแกนของปุ่มที่ถูกกดไปยังตำแหน่งพอร์ทอีกตำแหน่งหนึ่ง ซึ่งเป็นตำแหน่งที่รอมไบออสใช้ในการอ่าน

ก่อนที่จะศึกษาต่อไปว่ารอมไบออสจะทำอย่างไรกับข้อมูลที่ได้มา เราจะต้องเรียนรู้ ถึงปฏิบัติการขั้นแรกที่เกิดขึ้นที่แป้นพิมพ์ก่อน

ดังที่ทราบแล้วว่าแป้นพิมพ์จะเก็บค่ารหัสสแกนของปุ่มที่ถูกกด และรอจนกระทั่งไบออสเรียกใช้ (ซึ่งเวลาที่รอนี้จะสั้นมากประมาณ 1/1000 วินาที) ดังนั้นแป้นพิมพ์จึงต้องมีหน่วยความจำขนาดเล็กอยู่ด้วย เพื่อใช้บันทึกค่ารหัสสแกนที่ยังไม่ถูกส่งไปให้กับไบออสเอาไว้ ในบางครั้งเราเรียกเนื้อที่ส่วนนี้ว่าเป็น บัฟเฟอร์ของแป้นพิมพ์ (keyboard buffer) ซึ่งโดยทั่วๆ ไปหน่วยความจำที่ว่านี้จะมีขนาดพอที่จะเก็บบันทึกรหัสสแกนไว้ได้จำนวนหนึ่ง อย่างไรก็ดี มีสิ่งสำคัญสองประการที่คุณควรจะเรียนรู้เกี่ยวกับแป้นพิมพ์ คือ

1. แป้นพิมพ์จะเก็บรหัสสแกนทั้งตอนที่เรากดปุ่มและตอนที่ปล่อยปุ่ม โดยค่ารหัสสแกนของการกดปุ่มนี้จะแตกต่างกัน (กล่าวคือ รหัสของการปล่อยปุ่มพิมพ์จะมีค่าเท่ากับรหัสการกดปุ่มบวกด้วย 128 หรือ 80 ในฐาน 16) นั่นคือ เมื่อมีการกดปุ่ม 1 ครั้งที่แป้นพิมพ์ (ซึ่งหมายถึง ต้องมีการกดปุ่มและปล่อยปุ่มนั่นเอง) จะทำให้ไบออสถูกอินเทอรัพต์ 2 ครั้ง ดังนั้นไบออสจึงสามารถรู้ได้ว่าขณะนี้ ปุ่มกำลังถูกกดอยู่หรือถูกปล่อยแล้ว ตัวอย่างเช่น ไบออสจะสามารถรับได้ว่าคุณกำลังกดปุ่มอักษรตัวใหญ่เพราะปุ่ม ชิฟ (shift) จะถูกกดค้างไว้โดยไม่ปล่อย

2. ในกรณีที่เรากดปุ่มค้างไว้โดยไม่ปล่อยปุ่ม จะเป็นหน้าที่ของส่วนฮาร์ดแวร์ของแป้นพิมพ์ที่จะรับรู้เพราะการกดค้างนั้นนานเกินกว่ากำหนดไว้ (เช่นประมาณครึ่งวินาที) และแป้นพิมพ์ก็จะทำการส่งรหัสสแกนพิเศษอันเป็นรหัสที่หมายถึงการกดปุ่มค้าง (repeat key)

ย้อนกลับมาดูการทำงานของไบออส เมื่อตัวจัดการอินเทอรัพต์ที่ดูแลแป้นพิมพ์ของไบออสเริ่มทำงานมันจะรับค่ารหัสสแกนจากแป้นพิมพ์ (ในคีย์บอร์ดต่างชนิดจะมีรหัสสแกนไม่เหมือนกัน) และจัดการให้ความหมายแก่มัน โดยผ่านการวิเคราะห์อย่างรวดเร็วหลายขั้นตอน เริ่มแรกมันจะตรวจสอบว่า มีการกดปุ่มประเภทพิเศษหรือไม่ บรรดาปุ่มพิเศษประเภทชิฟ ได้แก่ ปุ่มชิฟ (ทั้งทางด้านซ้ายและขวา) ปุ่ม ALT และปุ่ม CTRL ซึ่งถ้ามีการกดมันจะบันทึกว่าอยู่ในสถานะ ชิฟ (shift state) ต่อจากนั้นไบออสจะตรวจสอบว่า มีการกดปุ่มประเภท TOGGLE หรือไม่ ปุ่มประเภทนี้ได้แก่ ปุ่ม CAPS LOCK , NUM LOCK , SCROLL LOCK และ INS ขอให้สังเกตความแตกต่างระหว่างปุ่มประเภท ชิฟ และประเภท TOGGLE สักนิด ลักษณะการใช้งานของปุ่มประเภท ชิฟ นั้น จะต้องกดค้างไว้และกดปุ่มอื่นตาม ซึ่งจะให้ความหมายที่แตกต่างจากการกดปุ่มอักขระดังกล่าวเฉยๆ (เช่น การกดปุ่ม ชิฟ ตามด้วยปุ่ม a ก็จะทำให้หมายถึง A เป็นต้น) ในขณะที่การกดปุ่มประเภท TOGGLE จะกดเพื่อเปลี่ยนสถานะ เช่น การกด CAPS LOCK จะเป็นการเปลี่ยนสถานะให้การกดปุ่มอักขระในเวลาต่อมากลายเป็นตัวใหญ่หมดและถ้าเรากด CAPS LOCK อีกที ก็จะเปลี่ยนสถานะกลับมาสู่ปกติดังเดิม โดยสรุปแล้วปุ่มทั้ง 2 ประเภทนี้ สามารถส่งผลต่อการแปลงรหัสสแกนได้โดยปุ่มประเภท ชิฟ จะส่งผลขณะที่มันถูกกดค้างไว้เท่านั้น ในขณะที่ปุ่มประเภท TOGGLE ส่งผลเมื่อมันอยู่ในสถานะ "ON" สถานะทั้ง 2 ประเภทนี้ (ทั้งปุ่ม SHIFT และ TOGGLE) จะถูกเก็บไว้ในเนื้อที่หน่วยความจำส่วนล่างอันได้แก่ ตำแหน่งแอดเดรส 417 และ 418 (ฐานสิบหก) โดยแต่ละบิทในแต่ละไบท์จะเก็บสถานภาพของการกดปุ่มเอาไว้กล่าวคือ ในไบท์แรกจะเก็บสถานะของการที่กดปุ่มค้างและไบท์ที่สองจะเก็บสถานะของปุ่ม TOGGLE ต่างๆ ว่าจะเป็น "ON" หรือ "OFF"

นอกเหนือจากข้อมูลที่เก็บในตำแหน่งที่ 417 และ 418 แล้วยังมีข้อมูลที่เก็บในตำแหน่ง 496 และ 497 ซึ่งจะไม่ใช้ในคีย์บอร์ดรุ่นเก่าๆ ในเนื้อที่ส่วนนี้จะบรรจุสถานะเอาไว้ว่าปุ่ม ALT หรือ CTRL ด้านขวามือถูกกดค้างไว้หรือไม่ (โดยให้ดูที่บิท 3 และ 2 ของไบท์ที่ 496)

หลังจากที่รอมไบออสได้ตรวจสอบว่า ได้มีการกดปุ่มประเภท ชิฟ และ TOGGLE หรือไม่และอยู่ในสถานะอะไรเป็นที่เรียบร้อยแล้ว มันก็จะตรวจสอบต่อไปว่า ได้มีการกดปุ่มประเภทที่จะทำให้เกิดความหมายพิเศษขึ้นหรือไม่ เช่น กดปุ่ม CTRL - ALT - DEL พร้อมๆ กัน อันทำให้เกิดการบูตดอสขึ้นมาใหม่หรือการกดปุ่ม PAUSE ก็จะทำให้เกิดการหยุดการทำงานชั่วคราวขึ้น ซึ่งเรากำลังจะได้ศึกษารายละเอียดของเนื้อหาส่วนนี้กันต่อไป

ในที่สุดหลังจากตรวจสอบดูแล้วพบว่ามีการกดปุ่ม 2 ประเภทนี้ คือ ปุ่มประเภทชิฟ หรือประเภท TOGGLE พร้อมๆ กันกับกดปุ่มอื่นๆ อีกปุ่มหนึ่ง ไบออสจะจัดการให้ความหมายแก่มัน เช่น ตรวจสอบดูแล้วพบว่ามีการกดปุ่ม ชิฟ และปุ่ม a พร้อมกันก็จะหมายถึง "A" แทนที่จะเป็น "a" ความหมายที่ได้นี้แบ่งได้เป็นสองประเภทคือ ประเภทแรกเป็นรหัสแอสกีที่เกิดจากการกดปุ่มแอสกีใดๆ เช่น A, CONTROL-A ส่วนประเภทที่ 2 ได้แก่รหัสพิเศษที่เกิดจากการกดปุ่มพิเศษ เช่น ปุ่มฟังก์ชั่นต่างๆ หรือปุ่มที่ใช้เลื่อนเคอร์เซอร์เป็นต้น

ในการเก็บบันทึกรหัสแอสกีและรหัสพิเศษของพีซีนั้น ไบออสจะใช้เนื้อที่ 2 ไบท์ ในการเก็บ โดยถ้าเป็นตัวอักษรแอสกีก็จะเก็บรหัสแอสกีไว้ที่ไบท์แรก (และเก็บรหัสสแกนของปุ่มนั้น ไว้ในไบท์ที่สอง) แต่ถ้าเป็นตัวอักขระพิเศษซึ่งไม่มีรหัสแอสกีก็จะเก็บ 0 ไว้ที่ไบท์แรกและเก็บรหัสของปุ่มคีย์พิเศษนี้ไว้ในไบท์ที่ 2

หลังจากที่เรากดปุ่มคีย์ (ซึ่งจะเก็บรหัสสแกนไว้ในบัฟเฟอร์ของคีย์บอร์ด) และไบออสเข้ามาแปลงปุ่มดังกล่าวเป็นข้อมูลสองไบท์แล้วนั้น ข้อมูลทั้งสองไบท์นี้ก็จะถูกเก็บไว้ในบัฟเฟอร์ของไบออสในส่วนที่กันไว้สำหรับคีย์บอร์ด โดยที่ไบออสได้กันเนื้อที่ไว้สำหรับเก็บข้อมูลได้ 32 อักขระและถ้าเกิดการล้นของข้อมูล (over flow) มันจะส่งเสียง beep และตัดข้อมูลที่เข้ามาทีหลัง

หลังจากที่ไบออสแปลความหมายได้แล้ว โปรแกรมของเราก็สามารถไปดึงความหมายนี้มาได้โดยตรงจากรอมไบออสหรืออาจจะผ่านดอส ซึ่งดอสก็จะไปดึงมาจากรอมไบออสอีกทีและ เนื้อหาทั้งหมดที่กล่าวมาก็เป็นข้อมูลและเหตุการณ์ทางเทคนิคที่เกิดขึ้น เมื่อเราใช้คีย์บอร์ดตามปกติและการที่เราต้องทำความเข้าใจกับความรู้พื้นฐานเหล่านี้ ก็เพื่อเป็นหนทางที่สามารถทำให้เราเล่นอะไรแปลกๆ กับคีย์บอร์ดได้

2.2.1 อินเตอรัพต์ INT 9H

สำหรับงานหลักของโปรแกรมไบออสอินเตอรัพต์ INT 9H นั้นก็คือ การนำข้อมูลการกดคีย์ของคีย์บอร์ดผ่านทางไอโอพอร์ท A ของชิป PPI 8255 และจะตีความจากข้อมูลนั้น จากนั้นก็จะนำข้อมูลไปเก็บไว้ในคีย์บอร์ดบัฟเฟอร์ ตำแหน่งของคีย์บอร์ดบัฟเฟอร์จะอยู่ในหน่วยความจำสำรองตอนล่างที่มักเรียกว่า ส่วนเก็บข้อมูลของไบออส (bios data area)

สาเหตุที่ต้องมีการตีความหมายของข้อมูลที่รับมาก็เพราะ ข้อมูลที่ออกมาจากคีย์บอร์ดจะอยู่ในรูปตัวเลขที่เรียกว่า "สแกนโค้ด" สแกนโค้ดนี้ไม่ใช่รหัสของตัวอักษรใดๆ ดังที่ปรากฏบนคีย์บอร์ด แต่เป็นเพียงรหัสที่บอกตำแหน่งของปุ่มที่ถูกกด

นอกจากนี้ อินเตอรัพต์ INT 9H ยังมีหน้าที่ ในการดูสถานะของการกดคีย์ด้วย อันได้แก่ CAP LOCK และ ชิฟ ทั้งสองข้าง เพื่อที่จะสามารถบอกได้ว่า อักษรที่กดนั้นเป็น "A" หรือ "B" เพื่อที่จะกำหนดรหัสแอสกีได้ถูกต้อง ทั้งสแกนโค้ดและรหัสแอสกี ที่ได้จากการตีความหมายจถูกนำไปเก็บไว้ในคีย์บอร์ดบัฟเฟอร์ ซึ่งเมื่อถึงจุดนี้ก็แสดงว่า INT 9H ได้ทำหน้าที่ของมันครบสมบูรณ์แล้ว ซีพียูก็จะหวนกลับไปทำงานที่หยุดค้างไว้ ณ จุดก่อนที่มันจะถูกสั่งให้ทำการอินเตอรัพต์

2.2.2 อินเตอรัพต์ INT 16H

โปรแกรมนี้มีหน้าที่ตรวจสอบข้อมูลมาคอยอยู่ในบัฟเฟอร์แล้วหรือยัง INT 16H นี้ มีฟังก์ชั่น 0, 1 และ 2 ฟังก์ชั่น

ฟังก์ชั่น 0 มีหน้าที่ ในการนำข้อมูลจากคีย์บอร์ดบัฟเฟอร์ส่งให้กับโปรแกรม การใช้งานฟังก์ชั่นนี้ มีจุดที่จะต้องระมัดระวังอยู่อย่างหนึ่งก็คือ ถ้าเรียกใช้ฟังก์ชั่น 0 นี้ กรณีไม่มีข้อมูลในคีย์บอร์ดบัฟเฟอร์แทนที่ฟังก์ชั่น 0 จะส่งค่า 0 กลับมายังโปรแกรมใช้งาน กลับเป็นการสร้างขั้นตอนการรอคอยขึ้น ซึ่งจะรอจนกว่าจะมีข้อมูลส่งมาเก็บในบัฟเฟอร์ จึงทำให้โปรแกรมใช้งานต้องหยุดรอไปด้วย การที่ INT 16H ต้องสร้างขั้นตอนการรอคอยขึ้น ก็เพื่อความรวดเร็วในกรณีที่มีการกดคีย์บอร์ดบ่อยและเร็ว ซึ่งทันทีที่ INT 9H นำข้อมูลมาไว้ในคีย์บอร์ดบัฟเฟอร์ INT 16H ฟังก์ชั่น 0 ก็จะนำข้อมูลนั้นๆ ไปประมวลผลใช้งานได้ทันทีเช่นกัน

ฟังก์ชั่น 1 นั้นจะมีประโยชน์มาก สำหรับโปรแกรมการใช้งานทั่วไปที่ไม่ต้องการรอคอย ในการใช้งานจริงมักใช้ฟังก์ชั่น 1 นี้ก่อน แล้วจึงตามด้วยฟังก์ชั่น 0 เพื่อตรวจดูว่ามีข้อมูลในบัฟเฟอร์หรือไม่ ถ้าไม่มีจะได้กลับไปทำงานอย่างอื่นได้ แต่ถ้ามีข้อมูล จึงค่อยเรียกฟังก์ชั่น 0 เพื่อจะรับข้อมูลทั้งหมดเข้ามาทีเดียว

ฟังก์ชั่น 2 จะทำหน้าที่ส่งข้อมูลในการกดคีย์พิเศษเพิ่มเติมต่างๆ อันได้แก่คีย์ CAPSLOCK, INSERT, NUM LOCK, ALT, CTRL และคีย์ชิฟ ทั้งสองข้าง

2.2.3 รายละเอียดเพิ่มเติมเกี่ยวกับอินเตอรัพต์

เนื่องจากหลักการอินเตอรัพต์เป็นสิ่งที่สำคัญ ที่จะทำให้สามารถเข้าใจถึงขั้นตอนการประมวลผลของฟังก์ชั่นการรับคีย์บอร์ด ดังนั้นจะขอกล่าวรายละเอียดเพิ่มเติมสักเล็กน้อย ที่เกี่ยวกับขั้นตอนการทำงานของอินเตอรัพต์ว่ามีขั้นตอนอะไรเกิดขึ้นบ้างสำหรับซีพียูในตระกูล 8086 จะมีวิธีการตอบสนองต่อคำสั่ง INT ซึ่งเป็นซอฟต์แวร์อินเตอรัพต์เช่นเดียวกับฮาร์ดแวร์อินเตอรัพต์คือ เมื่อเกิดการอินเตอรัพต์ขึ้นซีพียูจะถูกควบคุมโดยโปรแกรมที่มีชื่อว่า รูทีน จัดการอินเตอรัพต์คำสั่ง INT จะใช้คู่กับคำสั่ง IRET ซึ่งเป็นการคืนการควบคุมซีพียูไปให้กับโปรแกรมที่เรียกอินเตอรัพต์สำหรับขั้นตอนการใช้คำสั่ง INT โดยละเอียด ดังนี้

เมื่อมีการใช้ INT เกิดขึ้นโดยมีสาเหตุจากฮาร์ดแวร์ภายนอก หรือสั่งภายในโปรแกรมเองก็ตาม ซีพียูจะเก็บสถานะปัจจุบันของแฟล็กรีจิสเตอร์ไว้ โดยการเก็บไว้ในสแตก (stack) จากนั้นจะเคลียร์อินเตอรัพต์แฟล็ก (IF) และแทร็ปแฟล็ก (TF) ให้มีค่าเป็น 0 ในการเคลียร์ IF นี้จะมีเช่นเดียวกันกับคำสั่งเคลียร์อินเตอรัพต์ (CLI) คือในระหว่างนั้นซีพียูจะไม่อนุญาตให้มีการเรียกอินเตอรัพต์ซ้อนเข้าไปอีก จนกว่าขบวนการอินเตอรัพต์จะทำงานเสร็จ ตัวอย่างเช่น การรับและเก็บข้อมูลจากพอร์ทที่ใช้ในการสื่อสารในกรณีที่โปรแกรมมีขนาดยาว ซีพียูจะอนุญาตให้สามารถเรียกใช้อินเตอรัพต์ได้อีกครั้งถ้าข้อมูลที่รับมาจากพอร์ทสื่อสารอยู่ในความปลอดภัยแล้ว ซึ่งการอนุญาตในลักษณะนี้จะกระทำได้โดยใช้คำสั่งเซ็ตอินเตอรัพต์ (STI) ซึ่งถ้าโปรแกรมจัดการอินเตอรัพต์ยาวไม่มากนักก็จะไม่นิยมใช้คำสั่ง STI นั่นก็คือ จะไม่อนุญาตให้ใช้อินเตอรัพต์จนกว่างานอินเตอรัพต์ที่อยู่นั้นจะเสร็จ

ต่อมาไมโครโปรเซสเซอร์จะเก็บค่าของเซ็กเมนต์รีจิสเตอร์ CS ไว้ในสแตกเพื่อที่จะนำเซ็กเมนต์รีจิสเตอร์ CS ไปโหลดค่าแอดเดรสเซ็กเมนต์ของโปรแกรมอินเตอรัพต์ ที่เก็บไว้ในตารางอินเตอรัพต์เวกเตอร์มาใส่ไว้ ซึ่งในส่วนตารางอินเตอรัพต์นี้จะอยู่ในตำแหน่งเริ่มต้นของหน่วยความจำ โดยสามารถคำนวณตำแหน่งหน่วยความจำที่เก็บแอดเดรสเริ่มต้นของโปรแกรมอินเตอรัพต์ได้ จากการเอา 4 คูณหมายเลขอินเตอรัพต์ ค่าที่ได้จะเป็นตำแหน่งที่เห็นค่าออฟเซ็ตและเอาตำแหน่งดังกล่าว บวกด้วย 2 ก็จะได้ ค่าตำแหน่งตามตารางอินเตอรัพต์เวกเตอร์ จะเก็บแอดเดรสของออฟเซ็ตก่อน แล้วจึงเก็บแอดเดรสของเซ็กเมนต์ ยกตัวอย่างเช่น INT 9H แอดเดรส 36,37 จะเก็บตำแหน่งของออฟเซ็ตและแอดเดรส 38,39 จะเก็บตำแหน่งของเซ็กเมนต์ ส่วนรีจิสเตอร์ PC หรือ IP ก็จะถูกเก็บไว้ในสแตกเช่นเดียวกัน เพื่อที่ว่าจะได้นำตำแหน่งของออฟเซ็ตของโปรแกรมอินเตอรัพต์ที่พบในตารางมาใส่ ซึ่งจะมีค่าเท่ากับหมายเลขอินเตอรัพต์ คูณ 4 นั่นเอง

มาถึงขั้นตอนนี้ การควบคุมการทำงานก็จะอยู่ที่คำสั่งในตำแหน่ง ซึ่งชี้โดยค่าที่บรรจุลงไปใหม่ในรีจิสเตอร์ CS และ IP คือ CS : IP ซึ่งตำแหน่งนี้ก็คือ ตำแหน่งเริ่มต้นของโปรแกรมอินเตอรัพต์ โดยปกติแล้วสิ่งแรกที่โปรแกรมอินเตอรัพต์จะทำก็คือ การอนุญาตให้มีการเรียกอินเตอรัพต์ซ้อนขึ้นมาได้ โดยการใช้คำสั่ง STI ในการทำงานของโปรแกรมอินเตอรัพต์นั้น อาจจะมีการเปลี่ยนแปลงของรีจิสเตอร์ต่างๆได้ แต่จะต้องกระทำหลังจากได้เก็บค่าเหล่านั้นไว้ในสแตกแล้ว หลังจากเสร็จงานของโปรแกรมอินเตอรัพต์แล้ว ตัวจัดการอินเตอรัพต์จะนำค่าของรีจิสเตอร์ต่างๆ กลับออกมาจากสแตก แล้วใส่ไว้ในรีจิสเตอร์เหมือนกับตอนก่อนทำอินเตอรัพต์ และถ้าเป็นการจัดการสำหรับฮาร์ดแวร์อินเตอรัพต์แล้ว จะมีการส่งคำสั่งแสดงการสิ้นสุดของตัวจัดการไปยังตัวควบคุมอินเตอรัพต์ (pic:grogrammable interrupt control) เพื่อเป็นการบอกว่าสิ้นสุดกระบวนการอินเตอรัพต์แล้ว สิ่งสุดท้ายที่ตัวอินเตอรัพต์จะทำก็คือ คำสั่ง IRET ซึ่งจะเป็นการคืนการควบคุมไปยังโปรแกรมที่ทำการประมวลผลก่อนการเกิดอินเตอรัพต์ โดยในขั้นตอนนี้ จะทำในทำนองตรงกันข้ามกับขั้นตอนในตอนแรก คือข้อมูล 2 ไบต์ที่อยู่ในตำแหน่งสูงสุดบนสแตกจะถูกนำมาใส่ไว้ในรีจิสเตอร์ PC หรือ IP ค่านี้คือ ออฟเซ็ตของคำสั่งต่อไปที่จะถูกประมวลผล จากนั้นก็จะนำข้อมูลในสแตก 2 ไบต์ถัดมาใส่ไว้ในเซ็กเมนต์รีจิสเตอร์ CS ซึ่งค่านี้คือค่าเซ็กเมนต์ของคำสั่งต่อไปที่จะประมวลผล ส่วนข้อมูล 2 ไบต์สูงสุดท้าย จะเป็นค่าของแฟลกรีจิสเตอร์ก็เป็นอันว่า ตอนนี้การควบคุมได้ย้ายมาอยู่ที่โปรแกรมที่ถูกขัดจังหวะราบถ้วนเหมือนเดิม เนื่องจากว่าสถานะการทำงานของโปรแกรม โดยเฉพาะอย่างยิ่งค่าในรีจิสเตอร์จะถูกเก็บในที่เดิม ดังนั้นโปรแกรมที่ถูกขัดจังหวะจะไม่รู้เลยว่าเกิดอะไรขึ้นและจะดำเนินการไปตามปกติโดยไม่มีการสูญหายของข้อมูล

จะเห็นว่าการใช้วิธีการค้นหาตำแหน่ง โดยอาศัยข้อมูลในตารางอินเตอรัพต์เวกเตอร์นี้ จะเป็นการง่ายที่เราจะเปลี่ยนโปรแกรมอินเตอรัพต์ โดยไม่ต้องเปลี่ยนแปลงฟังก์ชั่นการทำงานของไมโครโปรเซสเซอร์ ซึ่งทำได้ โดยการเข้าไปแก้ไขข้อมูลที่เก็บไว้ในตารางอินเตอรัพต์เวกเตอร์ โปรแกรมนั้น อาจจะแตกต่างกันบ้างเป็นเรื่องธรรมดามากสำหรับ DOS รุ่นใหม่ อย่างเช่น DOS 3.X จะมีโปรแกรมอินเตอรัพต์ที่แตกต่างจาก DOS ซึ่งก็ทำนองเดียวกับไบออสอินเตอรัพต์ IBM PC/AT และเครื่องที่เลียนแบบก็จะมีความแตกต่างด้วย

2.3 การสื่อสารข้อมูล

2.3.1 การส่งข้อมูลแบบอนุกรม และ แบบขนาน (Serial vs. Parallel)

ข้อมูลอาจจะส่งผ่านโดยใช้สายคู่เดียวคือ เป็นการส่งผ่านแบบอนุกรม (serial) หรือใช้หลายคู่สายคือ เป็นการส่งผ่านแบบขนาน (parallel) ในการส่งข้อมูลแบบอนุกรม ข้อมูลไบนารี่จะถูกส่งออกไปครั้งละ 1 บิทที่เวลาหนึ่ง ส่วนในการส่งข้อมูลแบบขนาน ข้อมูลแต่ละบิทจะมีสายส่งเฉพาะและทุกบิทของข้อมูลที่แต่ละสายส่ง จะถูกส่งออกในเวลาเดียวกัน รูปที่ 2.6 (ก) แสดงตัวอย่างของการส่งข้อมูลแบบอนุกรมจากเทอร์มินอลไปยังคอมพิวเตอร์สำหรับรูปแบบไบนารี่ 01000100 บัสข้อมูลภายในระบบคอมพิวเตอร์ เป็นตัวอย่างหนึ่งของการส่งข้อมูลแบบขนาน ดังแสดงในรูปที่ 2.6 (ข)

การส่งข้อมูลแบบขนานย่อมเร็วกว่าส่งข้อมูลแบบอนุกรม เพราะว่าทุกบิทถูกส่งออกไปพร้อมกัน ดังนั้น การส่งข้อมูลแบบขนานจะถูกนำมาใช้สำหรับการทำงานภายในคอมพิวเตอร์ เช่น การส่งผ่านข้อมูลระหว่างซีพียูกับหน่วยความจำ (memory) หรือระหว่างซีพียูและไอโอชิฟ (I/O chip) และระหว่างหน่วยความจำกับไอโอชิฟ เป็นต้น

อย่างไรก็ตามการใช้สายหลายๆ เส้น จะรุงรัง ราคาสูงและรับสัญญาณรบกวนได้ง่าย ถ้าเอาการส่งข้อมูลแบบขนานมาใช้กับ การส่งข้อมูลระหว่างคอมพิวเตอร์กับคอมพิวเตอร์ หรือระหว่างคอมพิวเตอร์กับอุปกรณ์รอบนอก (peripheral equipments) เช่น แป้นพิมพ์กับจอภาพเครื่องพิมพ์ หรือ พล็อตเตอร์ เป็นต้น แม้ว่าบางระบบจะใช้การส่งข้อมูลแบบขนานกับการส่งผ่านข้อมูลไปยังอุปกรณ์รอบนอก

ส่วนใหญ่จะใช้แบบอนุกรม มีไอโอชิฟซึ่งสามารถแปลงข้อมูลแบบขนานจากซีพียูเพื่อเป็นแบบอนุกรมก่อนที่จะส่งข้อมูลออกไป และไอโอชิฟเหล่านี้ก็สามารถรับข้อมูลแบบอนุกรม และแปลงกลับมาเป็นแบบขนาน เพื่อว่ามันจะสามารถอ่านได้โดยซีพียูหรือหน่วยความจำ เพราะว่าการสื่อสารข้อมูลเกิดขึ้นระหว่างคอมพิวเตอร์ หรือคอมพิวเตอร์กับอุปกรณ์รอบนอก (peripheral) ดังนั้นในโครงงานนี้จะกล่าวถึง หรือเกี่ยวข้องกับการส่งข้อมูลแบบอนุกรมมากกว่าแบบขนาน โดยในหัวข้อต่อไปจะพูดถึงชนิดต่างๆ ของการส่งข้อมูลแบบอนุกรม

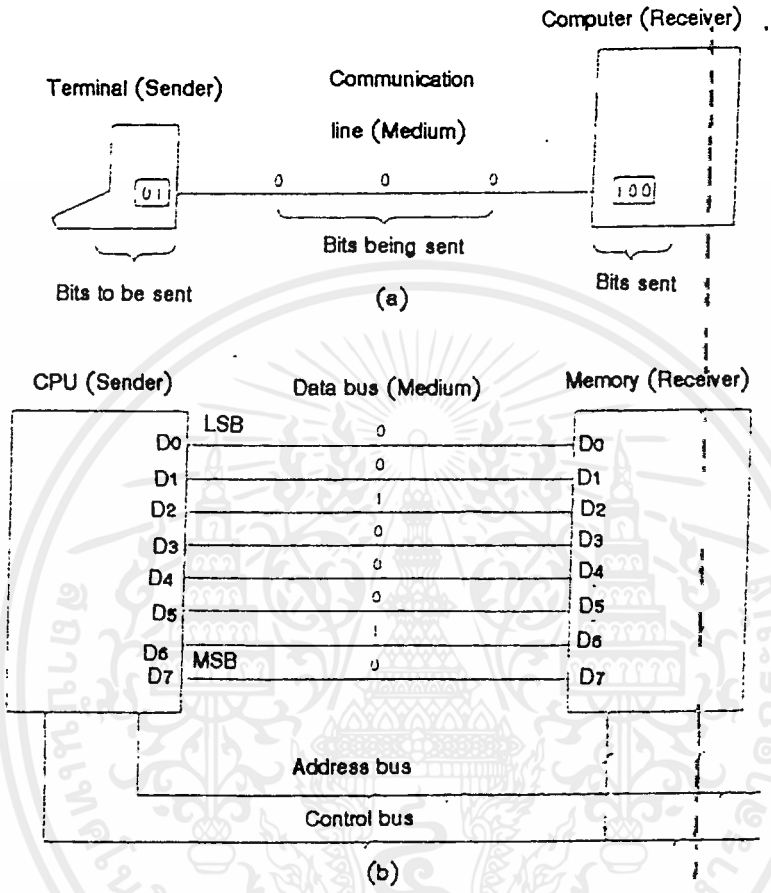
2.3.2 การส่งข้อมูลอนุกรมแบบอะซิงโครนัส (Asynchronous) กับแบบซิงโครนัส (Synchronous)

การสัญญาณแบบอนุกรมนั้น แบ่งออกเป็นสองวิธีคือ การส่งสัญญาณแบบอะซิงโครนัสและการส่งแบบซิงโครนัส เพื่อให้ตัวส่งและตัวรับสามารถทำงานสอดคล้องกันได้ทั้งคู่ ต้องใช้วิธีการส่งสัญญาณแบบเดียวกัน ตัวรับต้องสามารถตรวจจับการเริ่มต้นและการสิ้นสุดของอักขระ (character) 1 ตัวให้ได้ สำหรับการส่งข้อมูลแบบอะซิงโครนัส และเริ่มต้นการสิ้นสุดของบล็อคของอักขระสำหรับการส่งข้อมูลแบบซิงโครนัส

2.3.2.1 การส่งสัญญาณแบบอะซิงโครนัส (Asynchronous Transmission)

แบบอะซิงโครนัส หมายความว่า ที่เวลาหนึ่งอักขระ 1 ตัวสามารถถูกส่งออกไปได้ การส่งข้อมูลแบบอะซิงโครนัสจะใช้กับการส่งข้อมูลที่มีความเร็วต่ำ (น้อยกว่า 19,200 bit per second [bps])

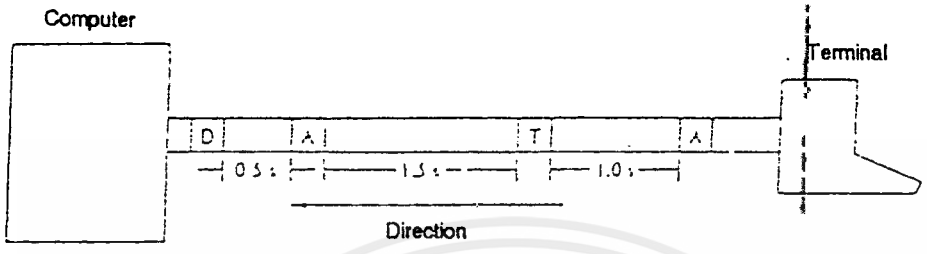
และใช้อุปกรณ์ราคาไม่แพงนัก เช่น CRT เทอร์มินอล , เครื่องพิมพ์และพล็อตเตอร์ เป็นวิธีที่ได้รับความนิยมในการส่งข้อมูลแบบนี้ เพราะว่าวงจรจะถูกออกแบบง่าย



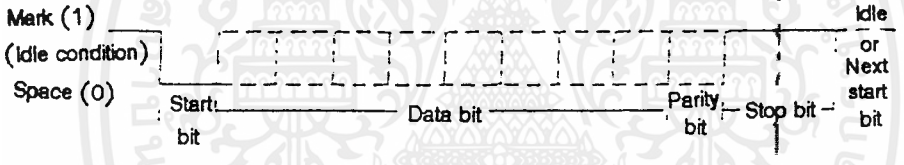
รูปที่ 2.6 ตัวอย่างของ ก. การส่งสัญญาณแบบอนุกรม
ข. การส่งสัญญาณแบบขนานหลายตัว

ซึ่งสามารถลดต้นทุนในการผลิตได้ วิธีนี้ยังอนุญาตให้มีการต่อเนื่องข้อมูล (bursts of data) กล่าวคือ เวลาระหว่างอักขระไม่จำเป็นต้องเท่ากัน เปรียบเทียบได้กับการพิมพ์หนังสือ คือ เวลาระหว่างการกดแป้นพิมพ์ของอักขระแต่ละตัวไม่จำเป็นต้องเท่ากัน อันอาจเนื่องมาจากคำที่กำลังพิมพ์ ตำแหน่งของอักขระบนแป้นพิมพ์ หรือการกดปุ่มยกแคร่พิมพ์ หรือการสัมผัสแป้นพิมพ์หนักเบาของผู้พิมพ์ เป็นต้น รูปที่ 2.7 แสดงตัวอย่างการส่งคำว่า "DATA" จากเทอร์มินอล ไปยังคอมพิวเตอร์ด้วยช่วงเวลา (time intervals) ที่แตกต่างกันระหว่างแต่ละอักขระ อะซิงโครนัสเป็นเทอมที่ทำให้เข้าใจผิดได้ง่าย เพราะว่าความหมายของมันแสดงให้เห็นว่า ไม่สัมพันธ์กันระหว่างตัวส่งและตัวรับ จริงๆ แล้ว ตัวรับจะสัมพันธ์กันสำหรับแต่ละอักขระที่รับเข้ามา โดยการใช้บิทเริ่มต้น (start bit) ของแต่ละอักขระ รูปที่ 2.8 แสดงรูปแบบของการส่งข้อมูลแบบอะซิงโครนัส จะเห็นว่า ในการส่งอักขระ 1 ตัว แบบอะซิงโครนัสประกอบด้วย 4 ส่วนด้วยกันคือ บิทเริ่มต้น (start bit), บิทข้อมูล

(data bit) , บิทพาริตี้ (parity bit) และบิทหยุด (stop bit [อาจมีได้ 1,1.5, หรือ 2 บิท]) แม้ว่าบิทพาริตี้จะเป็นตัวเลือกคือ มีหรือไม่มีก็ได้ แต่ระบบส่วนใหญ่จะใช้มัน ดังเราจะดูจากตัวอย่าง โดยมีทั้งการใช้และไม่ใช้บิทพาริตี้



รูปที่ 2.7 อักขระในการสื่อสารแบบอะซิงโครนัสสามารถส่งที่เวลาใดๆ



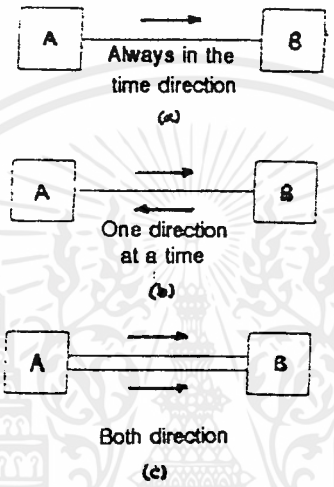
รูปที่ 2.8 รูปแบบอักขระสำหรับการส่งสัญญาณแบบอะซิงโครนัส

แม้ว่าระบบการสื่อสารแบบอะซิงโครนัสง่ายต่อการออกแบบสร้างและใช้ แต่เป็นแบบวิธีการส่งข้อมูลที่ไม่ค่อยมีประสิทธิภาพ เนื่องจากในการส่งข้อมูลแต่ละอักขระอย่างน้อยที่สุด จะต้องประกอบด้วยบิทเริ่มต้น 1 บิท และบิทหยุด 1 บิท ซึ่งผลอันนี้เราเรียกว่า เป็นค่า overhead factor โดยสามารถแสดงเป็นเปอร์เซนต์ของค่า overhead หรือจำนวนบิทที่ต้องสิ้นเปลืองเสมอไปในการส่งข้อมูลทุกอักขระ

2.3.3 โหมดการสื่อสาร (Communication Modes)

การสื่อสารแบ่งออกได้เป็น 3 แบบคือ การส่งข้อมูลแบบทางเดียว (simplex) แบบกึ่งสองทาง (half-duplex) และแบบสองทาง (full-duplex) รูปที่ 2.9 แสดงตัวอย่างของแต่ละระบบที่กล่าวมา ในการส่งข้อมูลแบบทางเดียว ข้อมูลจะถูกส่งไปในทิศทางเดียวเสมอ รูปที่ 2.9 (ก) แสดงการส่งข้อมูลจากจุด A ไปยังจุด B เนื่องจากไม่มีทางย้อนกลับ (nonreturn path) ไม่มีทางที่อุปกรณ์ที่จุด B จะส่งสัญญาณมาที่จุด A ตัวอย่างของการส่งแบบทางเดียว (simplex) ที่เห็นได้ง่ายๆ คือ การส่งโทรทัศน์และวิทยุกระจายเสียงข่าวสารจะถูกส่งไปในทิศทางเดียว ไม่มีทางที่จะส่งข่าวสารจากผู้รับกลับไปยังสถานีส่ง

การสื่อสารแบบกึ่งสองทาง (half -duplex) ยอมให้การส่งจากจุด A ไปยังจุด B และจากจุด B ไปยังจุด A ได้ แต่คนละเวลากัน ตัวอย่างการใช้งานการส่งข้อมูลแบบกึ่งสองทางนั้น ช่องสัญญาณที่ใช้ต้องเป็นแบบ "TURNED AROUND" นี้คือข้อเสียของการนำไปใช้งาน เพราะว่าอุปกรณ์ที่จุด A ต้องหยุดการส่งเสียก่อน จุด B จึงจะสามารถส่งได้ การส่งวิทยุแบบสองทาง (two-way radio) เป็นตัวอย่างของช่องสัญญาณแบบกึ่งสองทาง (half-duplex) เพราะว่าคนหนึ่งต้องบอก "ยกเลิก" ช่องสัญญาณเสียก่อน คนที่สองถึงจะสามารถส่งได้

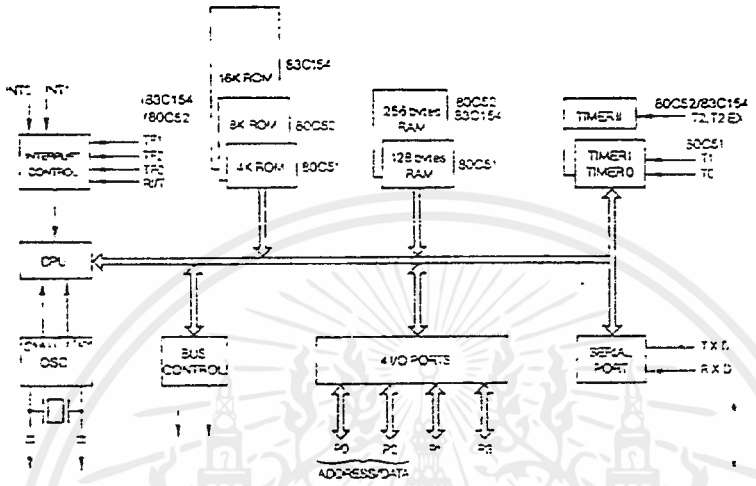


รูปที่ 2.9 ชนิดของช่องสัญญาณ แบบ
(ก) ทางเดียว (simplex)
(ข) กึ่งสองทาง (half-duplex)
(ค) สองทาง (full-duplex)

การสื่อสารแบบสองทาง (full-duplex) สามารถส่งข้อมูลจาก A ไปยัง B และจาก B ไปยัง A ได้ในขณะเดียวกันเครือข่ายโทรศัพท์เป็นแบบสองทาง (full-duplex) แม้ว่าจะเป็นการไม่สุภาพแต่คู่สนทนาสามารถพูดสวนกันได้ในเวลาเดียวกัน

2.4 โครงสร้างและการทำงานของไมโครคอนโทรลเลอร์ตระกูล 8051

บล็อคไดอะแกรมของโครงสร้างภายใน แสดงได้ดังรูปที่ 2.10



รูปที่ 2.10 บล็อคไดอะแกรมโครงสร้างของ 8051

โครงสร้างหลักของ ไมโครคอนโทรลเลอร์ 8051 ประกอบด้วย

- CPU ขนาด 8 บิท
- มีวงจรออสซิลเลเตอร์(OSCILLATOR) ในตัว
- มีหน่วยความจำแบบรอม ขนาด 4 กิโลไบท์ (ใน 8031 จะไม่มี)
- มีหน่วยความจำแบบแรม ขนาด 128 ไบท์
- มีรีจิสเตอร์สำหรับหน้าที่พิเศษ 21 รีจิสเตอร์
- มีสายนำสัญญาณสำหรับเชื่อมต่อกับอุปกรณ์ภายนอก 32 เส้น
- สามารถต่อหน่วยความจำภายนอกได้ 64 กิโลไบท์
- มีไทเมอร์/เคาน์เตอร์(TIMER/COUNTER) ขนาด 16 บิท จำนวน 2 ชุด
- มีโครงสร้างของการอินเทอรัพท์(INTERRUPT)ได้ 5 แห่ง ประกอบด้วยุความความสำคัญ 2 ระดับ
- มีพอร์ททำงานแบบ ฟูลดูเพล็ก(FULL DUPLEX) 1 พอร์ท
- สามารถอ้างอิงแอดเดรส(ADDRESS) แบบบิทได้ในกรณีที่มีการประมวลผลแบบบูลีน (BOOLEAN)

ลักษณะการจัดการของหน่วยความจำจะเป็นดังรูปที่ 2.11

- SCON SERIAL COUNTER
- SBUF SERAIL DATA BUFFER
- PCON POWER CONTROL

หน่วยความจำภายในที่ใช้เก็บข้อมูลจะถูกแบ่งออกเป็นสองส่วนคือ แอดเดรส 0 ถึง 127 ใช้เก็บข้อมูลภายใน และแอดเดรสที่ 128 ถึง 255 จะทำหน้าที่ เป็นรีจิสเตอร์ที่ทำหน้าที่พิเศษตามชื่อที่กำหนด และมีตำแหน่งที่แน่นอน เฉพาะที่ส่วนแรกตั้งแต่แอดเดรส 0 ถึง 31 จะถูกแบ่งออกเป็น 4 กลุ่มๆ ละ 8 ไบท์ เรียกแต่ละกลุ่มว่า รีจิสเตอร์แบงค์ (BANK REGISTER) 0 ถึง 3 เรียงกันไปตามลำดับ แต่จะแบงค์จะประกอบด้วย รีจิสเตอร์สำหรับใช้งานทั่วไปอีก 8 ตัว เริ่มจาก 0 ถึง 7 เรียงไปตามลำดับเช่นกัน หรือจะเรียกใช้งานตามค่าแอดเดรสโดยตรงเลยก็ได้

สำหรับรายละเอียดของรีจิสเตอร์ที่ทำหน้าที่พิเศษต่างๆ เป็นดังนี้

- แอกคิวมูเลเตอร์ (ACCUMULATER) เป็นรีจิสเตอร์ที่ใช้เก็บข้อมูล หรือ ผลลัพธ์ของคำสั่ง
- รีจิสเตอร์ B เป็นรีจิสเตอร์ ที่ใช้งานเฉพาะคำสั่ง การคูณและ การหาร เท่านั้น
- สแต็กพ้อยเตอร์ (STACK POINTER) เป็นรีจิสเตอร์ขนาด 8 บิท ค่าของสแต็กจะอยู่ที่ใดๆ ของหน่วยความจำแรมจำนวน 128 ไบท์ภายในชิพ เมื่อ 8051 ถูกรีเซ็ท (RESET) ค่าสแต็กจะถูกกำหนดเริ่มต้นไว้ที่ 07H
- ดาต้าพ้อยเตอร์ (DATA POINTER) เป็นรีจิสเตอร์ขนาด 16 บิท แยกออกเป็นไบท์สูง (DPH) และ ไบท์ต่ำ(DPL) หน้าที่หลักคือ เอาไว้เก็บค่าแอดเดรสจำนวน 16 บิท
- พอร์ท 0 ถึง 3 จะเป็นพอร์ทขนาด 8 บิท ที่สั่งให้เป็นอินพุท (INPUT) หรือ เอาท์พุท (OUTPUT) ก็ได้ และแบบที่ค้างสภาวะทางลอจิกไว้ได้ โดยสั่งผ่านทางรีจิสเตอร์ที่ทำหน้าที่พิเศษ คือ รีจิสเตอร์ P0 ถึง P3 พอร์ท 0 และ พอร์ท 2 ยังใช้เชื่อมต่อหน่วยความจำภายนอกด้วย นอกนี้ พอร์ท 3 ยังถูกใช้เป็นพอร์ทอนุกรมรับสัญญาณอินเทอร์รัพท์ภายนอก เป็นไทเมอร์ และ สัญญาณอ่าน หรือ เขียนข้อมูล ไปยังหน่วยความจำภายนอกด้วยดังนี้

ขาของพอร์ท 3	หน้าที่อื่นนอกจากเป็น I/O
P3.0	RXD (รับสัญญาณอนุกรม)
P3.1	TXD (ส่งสัญญาณอนุกรม)
P3.2	$\overline{\text{INT0}}$ (อินเทอร์รัพท์จากภายนอก)
P3.3	$\overline{\text{INT1}}$ (อินเทอร์รัพท์จากภายนอก)
P3.4	T0 (ไทเมอร์ 0 รับสัญญาณจากภายนอก)
P3.5	T1 (ไทเมอร์ 1 รับสัญญาณจากภายนอก)
P3.6	$\overline{\text{WR}}$ (สัญญาณเขียนหน่วยความจำภายนอก)
P3.7	$\overline{\text{RD}}$ (สัญญาณอ่านหน่วยความจำภายนอก)

- บัฟเฟอร์ (BUFFER) ข้อมูลอนุกรมถูกแยกเป็นสองรีจิสเตอร์ ตัวหนึ่งสำหรับส่งข้อมูลออก และอีกตัวหนึ่งสำหรับรับข้อมูลเข้า

- รีจิสเตอร์ควบคุมและแสดงผลสถานะ ได้แก่ รีจิสเตอร์ IP, IE, TMOD, TCON, SCON และ PCON ซึ่งแต่ละตัวจะประกอบด้วยบิทที่ใช้ควบคุมและ แสดงสถานะสำหรับการอินเทอร์รัพท์ ของระบบไทม์เมอร์ และ พอร์ทอนุกรม

หลังจากทำความเข้าใจกับโครงสร้างภายในแล้ว ลองพิจารณาถึงการใช้งานของสายสัญญาณที่ปรากฏภายนอกเพื่อให้เข้าใจได้ดีขึ้น

จากโครงสร้างภายใน 8051 จะมีบัส(BUS)ทาง 4 เส้น และพอร์ทขนาด 8 บิท เมื่อใช้หน่วยความจำภายในตัว ในกรณีที่ไม่ใช้หน่วยความจำภายใน พอร์ท 0 และ พอร์ท 2 จะถูกใช้เป็นบัสของข้อมูลและแอดเดรส ดังนั้นพอร์ททั้งสองยังคงใช้งานเป็นอินพุทและเอาท์พุท โดยพอร์ท 0 ทำหน้าที่เป็นสายสัญญาณ แอดเดรส A0.....A7 และยังเป็นบิทข้อมูล D0.....D7 ส่วนพอร์ท 2 ใช้งานเป็นสายสัญญาณแอดเดรส A8.....A15

เอาท์พุทของขา $\overline{RD}$ และ $\overline{WR}$ มาจากสายเอาท์พุท ของพอร์ท 3 เพื่อใช้ในการอ่านและเขียนข้อมูลของหน่วยความจำภายนอก

ขา $\overline{PSEN}$ เป็นขารับสัญญาณสำหรับเปิดให้มีการอ่านหน่วยความจำภายนอก โดยสัญญาณ $\overline{PSEN}$ จะทำงานสองครั้งในหนึ่งรอบคำสั่ง เหมือนสัญญาณ ALE เนื่องจากมีการอ่านข้อมูลจำนวนสองไบท์ ในแต่ละรอบคำสั่ง และขา $\overline{PSEN}$ นี้ จะทำงานต่อเมื่อ มีภาษาเครื่องเก็บอยู่ในหน่วยความจำภายนอกเท่านั้น

ขา $\overline{EA}$ เป็นอินพุทที่ใช้ร่วมกับแอดเดรสภายนอกโดยมีค่าลอจิก 0 เมื่อโปรเซสเซอร์ (PROCESSOR) อ่านคำสั่งจากหน่วยความจำภายนอก

2.5 การใช้งาน ไทม์เมอร์ / เคาน์เตอร์

การใช้งานไทม์เมอร์ / เคาน์เตอร์ ภายในของ MCS-8051 จะต้องมีการกำหนดรูปแบบการใช้งานต่าง ๆ เสียก่อน จึงจะทำงานได้อย่างถูกต้อง โดยการกำหนดค่าเริ่มต้นและการอ่านค่าจากไทม์เมอร์รีจิสเตอร์ เพราะฉะนั้นการจะเริ่มต้นใช้งานไทม์เมอร์ได้ จะต้องมีการกำหนดค่าในรีจิสเตอร์ต่าง ๆ ซึ่งรีจิสเตอร์ที่จะกำหนดจะมีดังนี้

1. **รีจิสเตอร์ TMOD** เป็นรีจิสเตอร์ขนาด 8 บิท อยู่ตำแหน่งที่ 89H ไม่สามารถอ้างตำแหน่งแบบบิทได้ รายละเอียดของการกำหนด ค่าบิทต่าง ๆ จะเป็นดังนี้

บิท	7	6	5	4	3	2	1	0
	Gate	$C/\overline{T}$	M1	M0	Gate	$C/\overline{T}$	M1	M0
	← Timer 1 →				← Timer 0 →			

ตารางที่ 2.2 แสดงหน้าที่ของบิทต่าง ๆ ของรีจิสเตอร์ TMOD

บิท	สัญลักษณ์	หน้าที่
7/3	GATE	การอีเนเบิล OR เกท เพื่อควบคุมการทำงานของไทม์เมอร์ GATE= 1 ไทม์เมอร์ จะทำงานถ้าสัญญาณขา IRn ในรีจิสเตอร์ TCON เป็น1 ด้วย
6/2	C/$\overline{T}$	ถ้าบิท C/$\overline{T}$ = 1 เป็นการเลือกการทำงานแบบนับพัลส์จากภายนอกที่ป้อนเข้าทางขา P3.5 (T1) หรือ P3.4 (TO) (COUNTER)
5/1	M1	ใช้งานในการเลือกโหมดการทำงานดังตารางที่ 2.3
4/0	M0	ใช้งานในการเลือกโหมดการทำงานดังตารางที่ 2.3

ตารางที่ 2.3 ตารางแสดงโหมดการใช้งานของ TMOD

M 1	M 0	โหมดที่ใช้งาน	ลักษณะการทำงาน
0	0	0	ไทม์เมอร์ขนาด 13 บิท
0	1	1	ไทม์เมอร์ขนาด 16 บิท
1	0	2	ไทม์เมอร์ขนาด 8 บิท แบบโหลดค่ากลับอัตโนมัติ
1	1	3	ไทม์เมอร์แบบใช้งานอิสระ

ตัวอย่างการให้ไทเมอร์ 1 ทำงานในโหมด 1 (16 บิท ไทเมอร์) รับสัญญาณนาฬิกา จากภายใน MCS -51 คำสั่งจะเป็นดังนี้

MOV TMOD, # 00010000B

จากคำสั่งนี้เซ็ทให้ M1= 0 เป็นการเลือกโหมด 1 ให้ C/$\overline{T}$ และ GATE = 0 เพื่อเลือกใช้สัญญาณนาฬิกาจาก วงจรออสซิลเลเตอร์ภายในชิพ และอีนาเบิลไทเมอร์ ให้ทำงานเมื่อบิท TR1 ในรีจีสเตอร์ TCON เซ็ทเป็น 1 (ในขณะนี้ ถือว่า TR1=0)

2. **รีจิสเตอร์ TH0, TL0 และ TH1, TL1** รีจิสเตอร์ขนาด 8 บิท ใช้กำหนดค่าการนับของไทม์เมอร์ ตำแหน่งของ TH0, TL0 จะอยู่ที่ 8CH, 8AH และตำแหน่งของ TH1, TL1 จะอยู่ 8DH, 8BH และเนื่องจากว่าไทม์เมอร์ภายในนี้ เป็นไทม์เมอร์แบบนับขึ้นจนถึงค่าสูงสุดคือ FFH (ในกรณีที่เลือกการทำงานแบบ 8 บิท), FFFFH (ในกรณีที่เลือกการทำงานแบบ 16 บิท) แล้วจะเกิดโอเวอร์โฟลว์ (เปลี่ยนจาก FFH หรือ FFFFH เป็นค่า 0) ฉะนั้นการกำหนดค่าการนับ จะต้องนำค่าสูงสุดของโหมดการทำงานนั้น ๆ มาลบออกด้วยค่าที่ต้องการ ตัวอย่าง ถ้าต้องการให้ไทม์เมอร์ 1 นับ 100 พัลส์ในการทำงานโหมด 1 (ไทม์เมอร์ 16 บิท)

- ค่าสูงสุดของการนับคือ FFFFH
- ค่าที่จะนำมาลบคือ 100D = 64H
- ผลลัพธ์ที่ได้จะเป็น FFFFH - 64H = FF9CH
- นำข้อมูลไบท์บนของผลลัพธ์ (FFH) ใส่ใน TH1

- นำข้อมูลไบท์ล่างของผลลัพธ์ (9CH) ใส่ใน TL1

```
MOV   TL1,#9CH
MOV   TH1,#0FFH
```

3. **รีจิสเตอร์ TCON** เป็นรีจิสเตอร์ขนาด 8 บิท อยู่ตำแหน่งที่ 88H สามารถอ้างตำแหน่งแบบบิทได้ ใช้งานในการควบคุมการทำงานของไทม์เมอร์และ ควบคุมการทำงานอินเทอร์รัพท์ด้วย รายละเอียดเกี่ยวกับบิทต่าง ๆ เป็นดังนี้

ตารางที่ 2.4 ตารางแสดงสัญลักษณ์แต่ละบิทของ TCON

บิท	7	6	5	4	3	2	1	0
	TF1	TR1	TF0	TR0	IE1	IT1	IE0	IT0

บิท	สัญลักษณ์	หน้าที่
7	TF1	บิทโอเวอร์โฟลว์ของไทม์เมอร์ 1 เมื่อ ไทม์เมอร์ 1 นับค่าจนเปลี่ยนจากค่าสูงสุด(FFH,FFFFH) เป็น 0 บิทนี้ จะถูกเซ็ทค่า เป็น 1 และบิทนี้จะเป็น 0 เมื่อกลับจากการทำงานในโปรแกรมบริการอินเทอร์รัพท์(Interrupt Service-Routine) หรือจะใช้การเคลียร์ค่าจากซอร์ฟแวร์
6	TR1	บิทควบคุมการเริ่มทำงานหรือ หยุดการทำงานไทม์เมอร์ 1 กล่าวคือ ถ้าบิท TR1 = 1 หมายถึงเริ่มต้นทำงาน ไทม์เมอร์ 1 TR1 = 0 หมายถึงหยุดการทำงานไทม์เมอร์ 1 ไม่ได้หมายความว่า ค่าภายในจะถูกเคลียร์
5	TF0	ความหมายและการใช้งาน เหมือนกับ TF1 แต่เป็นของ ไทม์เมอร์ 0
4	TF0	ความหมายและการใช้งาน เหมือนกับ TR1 แต่เป็นของ ไทม์เมอร์ 0
3	IE1	บิทแสดง การอินเทอรัพท์ภายนอกจากขา P3.3 (INT) ซึ่งเกิดจากขอบขา
2	IT1	บิทนี้ใช้สำหรับเลือกวิธีการอินเทอรัพท์ภายนอกจากขา P3.3(INT1) ถ้าบิท IT1 = 0 เป็นการเลือกการเกิดอินเทอรัพท์ จากระดับ ลอจิก 0 ถ้าบิท IT1 = 1 เป็นการเลือกการเกิดอินเทอรัพท์ จากขอบขาลง
1	IE0	การใช้งานเหมือนกับ IE1 แต่เป็นการอินเทอรัพท์ จากขา P 3.2 (INT0)
0	IT0	การใช้งานเหมือนกับ IT1 แต่เป็นการอินเทอรัพท์ จากขา P 3.2 (INT0)

ตัวอย่าง เมื่อทำการกำหนดค่าของไทม์เมอร์ เรียบร้อยแล้วจะเริ่มต้นทำงานไทม์เมอร์ 1 โดยการเซ็ทให้บิท TR1 มีค่าเป็น 1

- คำสั่งที่ให้ไทม์เมอร์เริ่มนับคือ

```
SETB  TR1
```

- เมื่อไทม์เมอร์ทำงานแล้ว ช่วงเวลาที่รอให้ไทม์เมอร์เกิดโอเวอร์โฟล์ว อาจจะทำการวนรอบคอย ดังนี้

```
WAIT :    JNB  TF1, WAIT
```

- เมื่อไทเมอร์เกิดโอเวอร์โฟล์ว โปรแกรมจะออกจากการวนรอบคำสั่งที่ใช้หยุดการนับของไทม์เมอร์ และเคลียร์แฟกโอเวอร์โฟล์ว คือ CLR TR1, CLR TR0

ในบางกรณีอาจต้องการอ่านค่าในไทม์เมอร์เข้ามาตรวจสอบ ในขณะที่ไทเมอร์กำลังทำงานอยู่ โอกาสที่จะผิดพลาดอยู่ที่การอ่านค่าเข้ามาถึงสองครั้ง เพราะไทเมอร์ขนาด 16 บิท เกิดจากการใช้รีจิสเตอร์ขนาด 8 บิทมาทำงานร่วมกัน การผิดพลาดจะเกิดขึ้นเมื่อเกิดโอเวอร์โฟล์ว (หรือตัวทด) จากไบท์ต่ำมายังไบท์สูงเพื่อแก้ปัญหานี้ ให้อ่านค่าไบท์สูงก่อน แล้วจึงอ่านค่าไบท์ต่ำ จากนั้นกลับมาอ่านค่าไบท์สูงอีกครั้ง ถ้าค่าไบท์สูงที่อ่านครั้งแรกและครั้งหลังไม่เท่ากัน ให้เริ่มขบวนการทั้งหมดอีกครั้งหนึ่ง

ตัวอย่างเช่น ต้องการอ่านค่า TL1/TH1 มาไว้ยังรีจิสเตอร์ R6/R7 โปรแกรมจะเขียนได้ดังนี้

```
AGAIN: MOV   A,TH1
       MOV   R6, TL1
       CJNE  A,TH1, AGAIN   ;ตรวจสอบค่าระหว่าง A กับ TH1
       MOV   R7 , A
```

2.6 การอินเทอร์รัพท์

การโปรแกรมใช้งานอินเทอรัพท์

ในตัวไมโครคอนโทรลเลอร์ MCS-51 จะมีรีจิสเตอร์ สำหรับโปรแกรมการทำงานอินเทอรัพท์อยู่ 3 ตัว คือ IE (Interrupt Enable) , IP (Interrupt Priority), TCON (Timer Control) ซึ่งรายละเอียดจะเป็นดังนี้

1. รีจิสเตอร์ IE (Interrupt Enable) อยู่ตำแหน่งแอดเดรส 0A8H สามารถอ้างตำแหน่งแบบบิทได้

ตารางที่ 2.5 ตารางแสดงสัญลักษณ์หน้าที่ของรีจิสเตอร์ IE

บิท	7	6	5	4	3	2	1	0
	EA	-	ET2	ES	ET1	EX1	ET0	EX0

บิท	สัญลักษณ์	หน้าที่
7	EA	บิท EA =1 การอินเทอรัพท์จะเกิดขึ้นได้ โดยขึ้นกับบิทควบคุมของอินเทอรัพท์นั้น ๆ EA = 0 ไม่ยอมให้มีการอินเทอรัพท์เกิดขึ้นได้

6	-	ไม่ใช้
5	ET2	สงวนไว้ใช้ภายหน้า
4	ES	บิท ES=1 ยอมให้มีการอินเทอรัพท์จากพอร์ทอนุกรม
		บิท ES=0 ไม่ยอมให้มีการอินเทอรัพท์จากพอร์ทอนุกรม
3	ET1	บิท ET = 1 ยอมให้มีการอินเทอรัพท์จาก TIMER 1
		บิท ET = 0 ไม่ยอมให้มีการอินเทอรัพท์จาก TIMER 1
2	EX1	บิท ES = 1 ยอมให้มีการอินเทอรัพท์จากภายนอกผ่านขา INT1
		บิท ES = 0 ไม่ยอมให้มีการอินเทอรัพท์จากภายนอกผ่านขา INT1
1	ET0	บิท ET = 1 ยอมให้มีการอินเทอรัพท์จาก TIMER 0
		บิท ET = 0 ไม่ยอมให้มีการอินเทอรัพท์จาก TIMER 0
0	EX0	บิท EX0 = 1 ยอมให้มีการอินเทอรัพท์จากภายนอกผ่านขา INT1
		บิท EX0 = 0 ไม่ยอมให้มีการอินเทอรัพท์จากภายนอกผ่านขา INT1

สามารถเขียนเป็นโปรแกรม ได้ดังนี้

```
MOV IE, # 86H
```

2. รีจีสเตอร์ IP (Interrupt Priority)

ตารางที่ 2.6 ตารางแสดงสัญลักษณ์หน้าที่ของรีจิสเตอร์ IP

บิท	7	6	5	4	3	2	1	0
	-	-	PT2	PS	PT1	PX1	PT0	PX0

บิท	สัญลักษณ์	หน้าที่
7	-	ไม่ใช้งาน
6	-	ไม่ใช้งาน
5	PT2	สงวนไว้ใช้งานภายหน้า
4	PS	ลำดับความสำคัญของการอินเทอรัพท์จากพอร์ทอนุกรม
3	PT1	ลำดับความสำคัญของการอินเทอรัพท์จาก TIMER1
2	PX1	ลำดับความสำคัญของการอินเทอรัพท์จาก INT1
1	PT0	ลำดับความสำคัญของการอินเทอรัพท์จาก TIMER0
0	PX0	ลำดับความสำคัญของการอินเทอรัพท์จาก INT0

ถ้าบิทต่าง ๆ ถูกเซ็ตเป็น 1 หมายถึง ลำดับความสำคัญสูงสุด

เคลียร์เป็น 0 หมายถึง ลำดับความสำคัญต่ำสุด

การกำหนดลำดับความสำคัญของแหล่งการเกิดอินเทอรัพท์ เท่ากัน CPU จะเรียงลำดับความสำคัญดังนี้

IE0	-	ลำดับความสำคัญสูงสุด
TF0	-	"
IE1	-	"
TF1	-	"
SERIAL	-	ลำดับความสำคัญต่ำสุด

3. รีจิสเตอร์ TCON (Timer Control) เป็นรีจิสเตอร์ขนาด 8 บิท อยู่ตำแหน่งที่ 88H สามารถอ้างตำแหน่งแบบบิทได้ใช้งานในการควบคุมการทำงานของไทม์เมอร์และควบคุมการทำงานอินเทอรัพท์ด้วย รายละเอียดเกี่ยวกับบิทต่าง ๆ เป็นดังนี้

ตารางที่ 2.7 ตารางแสดงสัญญลักษณ์หน้าที่ของรีจิสเตอร์ TCON

บิท	7	6	5	4	3	2	1	0
	TF1	TR1	TF0	TR0	IE1	IT1	IE0	IT0

บิท	สัญญลักษณ์	หน้าที่การใช้งาน
4-7	TF1	บิทควบคุมของไทม์เมอร์
3	IE1	บิทแสดงการอินเทอรัพท์ภายนอกจากขาสัญญาณ P3.3 (INT) ซึ่งเกิดจากขอบขาบิทนี้จะถูกเซ็ตและ เคลียร์ค่าจากฮาร์ดแวร์เท่านั้นบิท IE จะถูกเซ็ตเป็น 1 เมื่อมีการอินเทอรัพท์ จากขอบขาลงที่P3.3 (INT 1) บิท IE จะถูกเคลียร์เป็น 0 เมื่อทำงานโปรแกรมบริการอินเทอรัพท์เสร็จเรียบร้อย
2	IT1	บิทนี้ใช้สำหรับเลือกวิธีการ อินเทอรัพท์ภายนอกจากขา P3.3 (INT1) ถ้าบิท IT1 = 0 เป็นการเลือกการเกิดอินเทอรัพท์จากระดับลอจ การเลือกการเกิดอินเทอรัพท์จากขอบขาลง
1	IE0	ความหมายและการใช้งาน เหมือนกับ IE1 แต่เป็นการอินเทอรัพท์จากขา P3.3 (INT0)
0	IT0	ความหมายและการใช้งาน เหมือนกับ IT1 แต่เป็นการอินเทอรัพท์จากขา P3.3 (INT0)

* **ข้อควรระวัง** ถ้ามีการโปรแกรมให้ตรวจสอบที่ระดับสัญญาณ LOW จะต้องทำให้ระดับสัญญาณที่ขา INT 0, INT1 เป็น 1 ก่อนที่จะใช้คำสั่ง RETI เพื่อป้องกันการเกิดอินเทอรัพท์ซ้อน

ตำแหน่งของการอินเทอรัพท์

เมื่อเกิดการอินเทอรัพท์ขึ้น แอดเดรสที่โปรแกรมจะกระโดดไปทำงานถูกกำหนดค่าไว้แน่นอนอยู่แล้วดังนี้

ตารางที่ 2.8 ตารางแสดงตำแหน่งแอดเดรสของแหล่งการเกิดอินเทอรัพท์

แหล่งการเกิดอินเทอรัพท์	ตำแหน่งแอดเดรส	ตำแหน่งบน EVB 8051
IE0	0003H	FFE0H
TF0	000BH	FFE3H
IE1	0013H	FFE6H
TF1	001BH	FFE9H
SERIAL	0023H	FFECH

2.7 การใช้งานพอร์ทอนุกรม (โหมด 1)

รีจิสเตอร์ต่าง ๆ ที่เกี่ยวข้องในการใช้งานพอร์ทอนุกรม

1. รีจิสเตอร์ควบคุมไทม์เมอร์ เนื่องจากว่าการใช้งานพอร์ทอนุกรมนั้นจะมีสิ่งที่จะต้องคำนึงถึงก็คือ อัตราการรับส่งของข้อมูลหรือเรียกว่า อัตราบอด (Buad Rate) จริง ๆ แล้วก็คือ จังหวะการเลื่อนข้อมูลเข้าหรือออกจาก MCS 8051 นั่นเอง โดยอัตราบอดนี้จะสามารถสร้างภายในของ MCS 8051 ได้จากไทม์เมอร์แชนแนล 1 โดยทำงานในโหมด 2 คือ โหลดค่ากลับอัตโนมัติ ดังนั้น รจิสเตอร์ที่จะต้องทำงานโปรแกรม จะมีดังนี้

- TMOD ตำแหน่ง 89H ทำหน้าที่ เลือกโหมดของไทม์เมอร์
- TCON ตำแหน่ง 88H ทำหน้าที่ เริ่มต้นการสร้างอัตราบอด
- TH1 ตำแหน่ง 8CH ทำหน้าที่ ใส่ข้อมูลการนับของไทม์เมอร์ 1 เพื่อสร้างอัตราบอดตามต้องการ

2. รีจิสเตอร์ควบคุมการลดกำลัง เนื่องจากว่าการสร้างอัตราบอดนั้น จะต้องนำบิทในรีจิสเตอร์ PCON มาใช้ในการคำนวณข้อมูลของ TH1 ดังนั้น รีจิสเตอร์ที่ใช้ก็คือ

- PCON ตำแหน่ง 87H ทำหน้าที่ ในการคำนวณข้อมูลที่จะใส่ในรีจิสเตอร์ TH1 ซึ่งมีรายละเอียดดังนี้

ตารางที่ 2.9 ตารางแสดงหน้าที่แต่ละบิทของ PCON

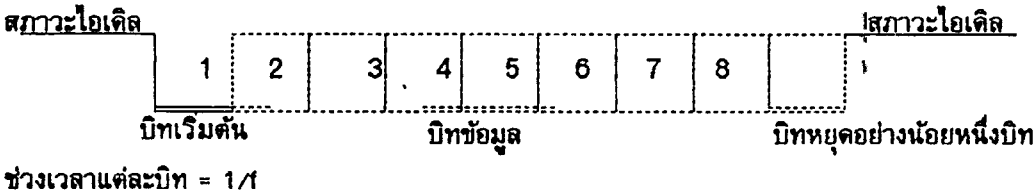
บิท	7	6	5	4	3	2	1	0
	SMOD	-	-	-	GF 1	GF 0	PD	IDL

บิต	สัญลักษณ์	รายละเอียด
7	SMOD	บิทที่ใช้สำหรับเปลี่ยนแปลงแก้ไขอัตราการส่งข้อมูลแบบอนุกรมโดยใช้ไทม์เมอร์ 1 เข้าช่วยถ้าเลือกใช้การรับส่งข้อมูลในโหมด 1,2 และ 3 แต่ถ้าเคลียร์เป็น 0 จะใช้ไทม์เมอร์ 1 ตามปกติที่เป็นอยู่ตามการกำหนดในช่วงเริ่มต้นของโปรแกรม

5	SM2	ใช้เป็นบิทแสดงการติดต่อระหว่าง ไมโครโปรเซสเซอร์หลายตัวด้วยกัน (Multiprocessor Communication)เซ็ตและเคลียร์โดยการใช้โปรแกรมควบคุม ในกรณีนี้จะใช้เฉพาะโหมด 2 และ 3 เมื่อบิทนี้ถูกเซ็ตเป็น1 การอินเตอร์รัพท์ก็จะเกิดขึ้นถ้าข้อมูลที่รับเข้ามา มีบิท 9 เป็น 1 ในทำนองกลับกัน ถ้าข้อมูลในบิทที่รับเข้ามาเป็น 0 การอินเตอร์รัพท์จะไม่เกิดขึ้น เมื่อใช้งานในโหมด 1 สัญญาณอินเตอร์รัพท์เกิดขึ้น เมื่อได้รับบิทหยุดที่ถูกต้องเท่านั้น ถ้าใช้ในโหมด 0 บิทนี้จะถูกเคลียร์ให้เป็น 0
4	REN	บิทอีนาเบิล การรับบิทนี้จะถูกเซ็ตเป็น 1 เมื่อต้องการรับสัญญาณอนุกรมและจะถูกเคลียร์ให้เป็น 0 เมื่อไม่ต้องการรับสัญญาณอนุกรม
3	TB8	ใช้เลือกว่าจะให้ส่งบิท 8 หรือไม่ และจะถูกรีเซ็ตหรือเคลียร์โดยโปรแกรมในโหมด 2 หรือ 3
2	RB8	ใช้เลือกว่าจะให้รับบิท 8 หรือไม่ ใช้สำหรับกรณีรับข้อมูลในโหมด 2 หรือ 3 บิทหยุดในโหมด 1 ส่วนในโหมด 0 จะไม่ใช้งานโหมดนี้
1	TI	แฟล็กอินเทอร์รัพท์เมื่อรับข้อมูลในโหมด 0 จะถูกเซ็ตให้เป็น 1 หลังจากที่ส่งบิท 7 ออกไปแล้ว ในโหมดอื่นๆ จะเซ็ตเป็น 1 เมื่อบิทหยุดเริ่มส่งออกไปการเคลียร์บิทนี้ จะทำได้โดยใช้โปรแกรม
0	RI	แฟล็กอินเทอร์รัพท์เมื่อรับข้อมูลในโหมด 0 จะถูกเซ็ตเป็น 1 หลังจากรับบิท 7 เรียบร้อยแล้ว ในโหมดอื่นๆ จะถูกเซ็ตเป็น 1 เมื่อรับบิทหยุดได้ครึ่งหนึ่ง การเคลียร์บิทนี้จะทำโดยใช้โปรแกรมการอ้างอิงแบบบิทแอดเดรสของรีจิสเตอร์นี้คือ SCON.0 ถึง SCON.7

การส่งข้อมูลแบบอนุกรมในโหมด 1 (Standard UART)

การส่งข้อมูลในโหมดนี้จะเป็นการส่งข้อมูล แบบอะซิงโครนัส 8 บิทข้อมูล 1 บิทเริ่มต้น และ 1 บิทหยุด ดังรูปที่ 2.8 การทำงานโหมดนี้ จะทำโดยการกำหนดข้อมูลในรีจิสเตอร์ SCON บิท SM0 และบิท SM1 ให้มีค่าเป็น "01" ซึ่งเป็นการกำหนดให้รีจิสเตอร์ SBUF ทำหน้าที่เป็นตัวรับส่งข้อมูลขนาด 10 บิท แบบฟูลดูเพล็กซ์ (Full Duplex) ซึ่งสามารถรับและส่งข้อมูลได้ในเวลาเดียวกัน โดยใช้ขา RXD ทำหน้าที่ รับสัญญาณอนุกรมที่เข้ามา และขา TXD ทำการส่งข้อมูลแบบอนุกรมออกไปภายนอก



รูปที่ 2.12 รูปแบบการรับส่งข้อมูลในโหมด 1

การส่งข้อมูลจะเริ่มต้นด้วยการส่งบิทเริ่มต้น (Start Bit) ออกไปก่อน แล้วตามด้วยบิทข้อมูล (โดยส่งบิท 0 ออกไปก่อน) จากนั้นจึงเป็นการส่งบิทหยุด(Stop Bit) แฟล็ก TI จะเซ็ตเมื่อส่งข้อมูลครบทั้ง 10 บิท

การรับข้อมูลจะเริ่มจาก ลอจิกในสายสัญญาณเปลี่ยนสภาวะจาก 1 เป็น 0 (ขอบบวกของบิทเริ่มต้น) บิทเริ่มต้นจะถูกข้ามไปไม่สนใจจะส่งข้อมูลอีก 8 บิทที่เหลือ เข้ารีจิสเตอร์เลื่อนบิท ภายในพอร์ทอนุกรมเมื่อครบทั้ง 8 บิทแล้ว สิ่งต่อไปนี้จะเกิดขึ้น

1. บิทที่ 9 (บิทหยุด) จะถูกเก็บเข้าไปในบิท RB8 ของ SCON
2. SBUF จะทำการโหลดข้อมูลทั้ง 8 บิทเข้าตัว
3. แฟล็ก RI จะเซ็ต

สิ่งที่กล่าวมาทั้งหมดจะเป็นจริงก็ต่อเมื่อ

1. RI = 0 และ
2. SM2 = 1 และบิทหยุดที่รับเข้ามาเป็น 1 หรืออีกกรณีหนึ่งคือ SM2 = 0

การคำนวณอัตราบอดของโหมด 1

ไทเมอร์ 1 จะถูกใช้เป็นตัวสร้างอัตราบอด เมื่อกำหนดการรับส่งข้อมูลแบบอนุกรมในโหลด 1 โดยไทเมอร์ 1 จะถูกใช้ให้ทำงาน ในโหมด 2 ซึ่งจะโหลดค่าเข้าไปอัตโนมัติ การคำนวณจะเป็นดังนี้

$$f_{buad} = \frac{2^{SMOD} \times \text{Oscillator}}{32d \quad 12d \times [256d - (TH1)]} = \text{ความถี่ของออสซิลเลเตอร์}$$

เมื่อ SMOD จะเป็นบิทควบคุมอยู่ในรีจิสเตอร์ PCON ซึ่งจะเป็น 0 หรือ 1 ก็ได้ถ้าเป็น 0 จะเป็นความถี่ปกติ ถ้าเป็น 1 ความถี่จะเพิ่มขึ้นเป็นสองเท่าดังได้กล่าวมาแล้วใน รีจิสเตอร์ PCON

ถ้าไทเมอร์ 1 ไม่ถูกใช้งานในโหมด 2 ค่าอัตราบอดจะเป็น

$$f_{buad} = \frac{2^{SMOD} \times (\text{timer 1 overflow frequency})}{32d}$$

ถ้าเลือกใช้อัตราบอดมาตรฐาน ควรจะเลือกความถี่ของคริสตอลให้ถูกต้องคือ 11.0592 เมกกะเฮิร์ซ สมมติว่าถ้าต้องการอัตราบอดเป็น 9600 เฮิร์ซ ค่าที่จะต้องกำหนดลงไปใน TH1 คือ

$$TH1 = \frac{256d - 2^{0} \times 11.0592 \times 10^{6}}{32d \quad 12 \times 9600d} = 253.000d$$

$$= 0FDH$$

เมื่อกำหนดให้ SMOD ในสมการนี้เป็น 0 ตารางที่ 2.11 แสดงการกำหนดค่าข้อมูล และความผิดพลาดของอัตราบอดที่สร้างขึ้นจากคริสตอล

ตารางที่ 2.11 ตารางการเลือกใช้อัตราบอดและค่ากำหนดเริ่มต้น

BAUD RATE	CRYTAL FREQUENCY	SMOD	TH1 RELOAD VALUE	ACTUAL BAUD RATE	ERROR
9600	12.000 MHz	1	-7(F9H)	8923	7%
2400	12.000 MHz	0	-13(F3H)	1202	0.16%
1200	12.000 Mhz	0	-26(E6H)	1202	0.16%
19200	11.059 Mhz	1	-3(FDH)	19200	0
9600	11.059 Mhz	0	-3(FDH)	9600	0
2400	11.059 Mhz	0	-12(F4H)	2400	0
1200	11.059 MHz	0	-24(E8H)	1200	0

การกำหนดค่าเริ่มต้นและการใช้งานรีจิสเตอร์ของพอร์ทอนุกรมบิทต่าง ๆ ของ SCON

- บิท SM0 (บิทที่ 7) และ บิท SM1 (บิทที่ 6)

เป็นการเลือกโหมดการทำงานพอร์ทอนุกรมสามารถเลือกได้ 4 โหมด ในการทดลองนี้จะเลือกใช้โหมด 1 ซึ่งสามารถเขียนโปรแกรมได้ดังนี้

```
MOV    SCON,#01XXXXXXB    ; ให้ค่าบิทที่ 7 และ บิทที่ 6 เป็น '01', X - ค่า '0' หรือ
                            '1' ก็ได้
```

- บิท SM2 (บิทที่ 5)

บิทนี้ถ้าเซ็ตเป็น '1' จะใช้งานในโหมดโปรเซสเซอร์หลายตัว (Multiprocessor) ดังนั้น การใช้งานโหมด 1 มิใช่เป็นการติดต่อโปรเซสเซอร์หลายตัว จึงให้บิทนี้เป็น '0' ซึ่งสามารถเขียนโปรแกรมได้ดังนี้

```
MOV    SCON,#XX0XXXXB    ; ให้ค่าบิทที่ 5 เป็น '0', X - ค่า '0' หรือ '1' ก็ได้
```

- บิท REN (บิทที่ 4)

ถ้าเซ็ตเป็น '1' หมายถึง MCS51 สามารถรับข้อมูล จากพอร์ทอนุกรมได้

ถ้าเคลียร์เป็น '0' หมายถึง MCS51 ไม่สามารถรับข้อมูล จากพอร์ทอนุกรมได้

สามารถเขียนเป็นโปรแกรมได้ 2 แบบ คือ

```
SETB   REN                ; บิทแอดเดรส (ควรมีการ equate ตำแหน่งของ REN ก่อน)
MOV    SCON,#XXX1XXXXB    ; ไบท์แอดเดรส (ให้ค่าบิทที่ 4 เป็น '1'), X มีค่า '0' หรือ
                            '1' ก็ได้
```


- บิท TB8 (บิทที่ 3) จะใช้งานในการติดต่อข้อมูลโหมด 2 และ โหมด 3

- บิท RB8 (บิทที่ 2) จะใช้งานในการติดต่อข้อมูลโหมด 2 และ โหมด 3

- บิท TI (บิทที่ 1) จะเซ็ตเมื่อทุกบิทของข้อมูล ส่งออกไปหมดแล้ว เพื่อเป็นการแสดงว่า บัฟเฟอร์ของการส่งข้อมูลว่าง ถ้าซอฟท์แวร์ต้องการส่งข้อมูลออกไปยังอุปกรณ์ที่ต่ออยู่กับพอร์ทอนุกรม โปรแกรมจะต้องตรวจสอบว่า พอร์ทอนุกรมว่างหรือไม่ (ซึ่งก็คือตรวจสอบว่าตัวอักษรที่แล้วส่งออกไปหรือยัง) ถ้าว่างก็ส่งข้อมูลต่อไป สมมติว่าต้องการส่งข้อมูล ออกจากแอกคิวมูเลเตอร์ โปรแกรมจะเป็นดังนี้

```
WAIT:   JNB TI,WAIT     ; ตรวจสอบ TI จนกว่าจะเซ็ต
        CLR TI          ; เคลียร์ TI
        MOV A,SBUF      ; อ่านตัวอักษรจาก SBUF
```

- บิท RI (บิทที่ 0) จะเซ็ตเมื่อเสร็จสิ้นการรับข้อมูลแล้ว เพื่อเป็นการแสดงว่าบัฟเฟอร์ ของการรับข้อมูลเต็มข้อกำหนดนี้จะตรวจสอบโดยซอฟท์แวร์หรือโปรแกรมให้เกิดการอินเท่อรัพท์ก็ได้ ถ้าซอฟท์แวร์ต้องการรับข้อมูลจากอุปกรณ์ที่ต่ออยู่กับพอร์ทอนุกรม เช่น เทอร์มินอล ตัวโปรแกรมจะรอจนกระทั่งบิท RI เซ็ต แล้วเคลียร์ด้วยโปรแกรมพร้อมทั้งอ่านข้อมูลมาจาก SBUF ดังนี้

```
WAIT:   JNB RI,WAIT     ; ตรวจสอบ RI จนกว่าจะเซ็ต
        CLR RI          ; เคลียร์บิท RI
        MOV A,SBUF      ; อ่านตัวอักษรจาก SBUF
```

ดังนั้น ถ้าสมมติว่าต้องการให้พอร์ทอนุกรม สามารถรับและส่งข้อมูลได้ และทำงานในโหมด 1 เราจะกำหนดค่าของ SCON เป็นดังนี้

SM0	SM1	SM2	REN	TB8	RB8	TI	RI
0	1	0	1	0	0	1	0

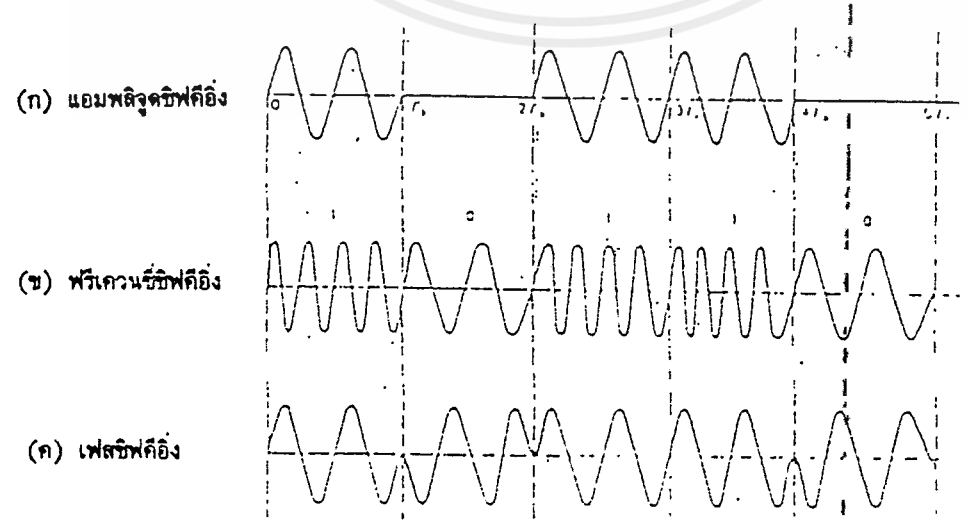
สามารถเขียนเป็นโปรแกรมได้ดังนี้

```
MOV    SCON,#01010010B
```

2.8 ดิจิตอลโมดูเลชั่น

ปัจจุบันการสื่อสารนิยมใช้ดิจิตอลโมดูเลชั่นมากขึ้น เพราะระบบดิจิตอลโมดูเลชั่นมีความเชื่อถือสูงและมีราคาถูกลง เนื่องจากได้มีการพัฒนาทางดิจิตอลอิเลคทรอนิกส์และไอซีอย่างรวดเร็ว ทำให้ต้นทุนการผลิตถูกลง นอกจากนี้ดิจิตอลโมดูเลชั่นยังสามารถใช้วิธีลงรหัสพัลส์ก่อนโมดูเลชั่น เพื่อลดพรอบบะบิวลิตี้ ความผิดพลาดให้ต่ำลง การโมดูเลทตัวพาด้วยข่าวสารดิจิตอลนี้ ทำได้หลายวิธีด้วยกัน ในที่นี้ เราจะศึกษาเทคนิคการโมดูเลทที่ข่าวสารดิจิตอลเปลี่ยนแปลงขนาด ความถี่ และเฟสของตัวพา รูปที่ 2.13 แสดงรูปคลื่นของดิจิตอลโมดูเลชั่น วิธีต่างๆที่ใช้ในการส่งข่าวสารไบนารี่ ผ่านช่องสื่อสาร (communication channel) ซึ่งโดยมากจะมีการตอบสนองไม่ดี บริเวณความถี่ใกล้ศูนย์ จึงคำนึงได้ว่าเป็นช่องความถี่ผ่านแคบ (bandpass channel) ในรูปที่ 2.13 (ก) สัญญาณดิจิตอล 1 และ 0 จะสวิตช์ขนาดของตัวพา ให้มีค่าสองค่า คือ เปิดและปิด รูปคลื่นที่ประกอบด้วยพัลส์เปิด หรือมาร์ค (mark) จะแทนเลขไบนารี่ "1" และพัลส์ปิดหรือสเปส (space) จะแทนเลขไบนารี่ "0" ดิจิตอลโมดูเลชั่นนี้เรียกว่า แอมปลิจูดชิฟคีอิ่ง (amplitude shift keying หรือ ASK) รูปที่ 2.13 (ข) แสดงรูปคลื่นของฟรีเควนซีชิฟคีอิ่ง (frequency shift keying หรือ FSK) ที่มีการเปลี่ยนแปลงความถี่ของตัวพาระหว่าง สองความถี่ตามสัญญาณดิจิตอล 1 และ 0 สัญญาณมาร์คที่มีความถี่สูงจะแทนพัลส์ 1 และสัญญาณสเปสที่มีความถี่ต่ำ จะแทนด้วยพัลส์ 0 ส่วนรูปที่ 2.13 (ค) แสดงรูปคลื่นของเฟสชิฟคีอิ่ง (phase shift keying หรือ PSK) ที่มีการเปลี่ยนเฟสของตัวพาระหว่างสองเฟส คือ 0 องศา และ 180 องศา ตามสัญญาณดิจิตอล 1 และ 0 เฟสชิฟคีอิ่งชนิดนี้เรียกว่า สองเฟส พีเอสเค(2 PSK) มีข้อสังเกตว่าในกรณีของเอฟเอสเค และพีเอสเค ขนาดของตัวพามีค่าคงที่เสมอ และรูปคลื่นที่ถูกโมดูเลทแล้วของดิจิตอลโมดูเลชั่น ทั้ง 3 วิธีมีค่าต่อเนื่องทุกเวลา

ในบทนี้ เราจะศึกษาวิสัยความสามารถของระบบดิจิตอลโมดูเลชั่น ทั้ง 3 ระบบในขณะที่มีเสียงรบกวน วิสัยความสามารถของดิจิตอลโมดูเลชั่นมักจะวัดในรูปของ พรอบบะบิวลิตี้ ความผิดพลาด เพื่อให้เกิดความเข้าใจง่ายขึ้น เราจะศึกษาเครื่องกรองความถี่แบบอ๊อฟติมัม (optimum filter) ซึ่งเป็นเครื่องกรองความถี่แบบแม็ช (matched filter) หรือเครื่องรับแบบคอร์รีเลชั่น (correlation receiver) ที่ใช้ในการดีโมดูเลท เอเอสเค เอฟเอสเค และพีเอสเค



รูปที่ 2.13 รูปคลื่นของดิจิตอลโมดูเลชั่นที่ใช้ในการส่งข้อมูลไบนารี่

2.8.1 เอเอสเค (ASK)

2.8.1.1 การจัดสัญญาณเอเอสเค (ASK signalling)

สัญญาณเอเอสเค (ASK) เป็นสัญญาณดิจิตอลโมดูเลชั่นที่ค้นพบก่อน ระบบดิจิตอลโมดูเลชั่นอื่น โดยอาศัยหลักการของแอมปลิจูดโมดูเลชั่น สัญญาณเอเอสเคนี้ เวลามีสัญญาณมาร์คหรือสัญญาณเปิด จะส่งสัญญาณพัลส์ เวลามีสัญญาณสเปสหรือสัญญาณปิดจะไม่ส่งสัญญาณ ซึ่งมีรูปคลื่นดังนี้

$$S1(t) = A \cos wct \quad \text{สำหรับสัญญาณมาร์ค}$$
$$S2(t) = 0 \quad \text{สำหรับสัญญาณสเปส} \qquad \text{.....(2.1)}$$

สัญญาณมาร์คส่งไม่จำเป็นต้องเป็นพัลส์สี่เหลี่ยมเสมอไป อาจเป็นพัลส์ที่ผ่านเครื่องกรองความถี่ต่ำผ่าน แล้วเป็นพัลส์ที่จัดรูปร่าง (shaped pulse) ก็ได้สัญญาณเอเอสเค อาจใช้เป็นสัญญาณโทรเลข หรือใช้เป็นสัญญาณดิจิตอลโมดูเลชั่นแบบต่างๆ ที่ไม่ต้องการความเร็วในการส่ง (bit rate) สูง สัญญาณเอเอสเค เมื่อส่งผ่านช่องสื่อสารจะถูกรบกวนโดยเสียงรบกวนเก๊าเซียน จะได้สัญญาณรวม

$$Vi(t) = \begin{cases} A \cos wct + n(t) & \text{สำหรับสัญญาณมาร์ค} \\ n(t) & \text{สำหรับสัญญาณสเปส} \end{cases} \qquad \text{.....(2.2)}$$

2.8.2 การเปรียบเทียบระบบดิจิตอลโมดูเลชั่น

ในการเปรียบเทียบวิสัยความสามารถของระบบดิจิตอลโมดูเลชั่น เราต้องพิจารณาในแง่ทฤษฎีหัวข้อต่อไปนี้

1. แถบความถี่ (bandwidth)
2. พรอบบะบิวลิตี้ ความผิดพลาด (probability of error)

แต่ในแง่ปฏิบัติ เวลาเราเลือกระบบดิจิตอลโมดูเลชั่น เพื่อใช้งานนั้น เราควรคำนึงถึงหัวข้ออื่นนอกเหนือจากหัวข้อก่อนด้วย เช่น อัตราความเร็วในการส่งข้อมูล ราคา ความยากง่าย ของอุปกรณ์ในการใช้ และการซ่อมบำรุงรักษา ความเข้ากันได้ของอุปกรณ์ที่จะจัดซื้อกับระบบเก่าที่มีอยู่ ภูมิต้านทาน (immunity) ของระบบดิจิตอลโมดูเลชั่นที่มีต่อเสียงรบกวนและความเสื่อมโทรมของช่องสัญญาณ (channel impairmint) เช่น ความไม่เป็นเชิงเส้น การสั่นของเฟส (phase jitter) เฟดดิ้ง (fading) และความเสถียรภาพของความถี่ (frequency stability)

หัวข้อเหล่านี้มีความสำคัญต่อการเลือกซื้อระบบดิจิตอลโมดูเลชั่นมาก เช่น ถ้าช่องสัญญาณ (channel) เกิดเฟดดิ้งอยู่เสมอ ถ้าเราเลือกส่งด้วยสัญญาณพีเอสเค จะได้ค่าพรอบบะบิวลิตี้ความผิดพลาดมากกว่าการเลือกส่งข้อมูลด้วยสัญญาณนอนโคฮีเร็นท์เอฟเอสเค เพราะในขณะที่เกิดเฟดดิ้งสัญญาณอ้างอิงโคฮีเร็นท์สำหรับพีเอสเค จะเพี้ยนไป ทำให้เกิดความผิดพลาดมาก เป็นต้น ในกรณีที่

มีข้อจำกัดทางด้านกำลังของสัญญาณ (power limitation) เช่น การรับส่งสัญญาณดาวเทียม ควรเลือกใช้แบบโคฮีเร็นท์ เพราะแบบโคฮีเร็นท์ใช้กำลังของสัญญาณน้อยกว่าแบบนอนโคฮีเร็นท์ที่อัตราการส่งและค่าพรอบบะบิวลิตี้ความผิดพลาดเดียวกัน

ความต้องการแถบความถี่ (bandwidth requirement) สัญญาณเอเอสเค ดีพีเอสเค และพีเอสเค จะมีความต้องการความถี่ประมาณ 2 เท่าของอัตราความเร็วในการส่งข้อมูล(data rated) สัญญาณเอฟเอสเคต้องการแถบความถี่มากกว่า 2 เท่าของอัตราความเร็วในการส่งข้อมูล ส่วนสัญญาณดิจิตอลเอฟเอ็มที่นิยมใช้กันส่วนมากเป็นแบบแถบความถี่แคบ (narrowband)นั้น ต้องการแถบความถี่อยู่ระหว่าง 2-3 เท่าของอัตราความเร็วในการส่งข้อมูล ฉะนั้นถ้าต้องการเลือกใช้ระบบดิจิตอลโมดูเลชั่นในการที่มีข้อจำกัดทางด้านแถบความถี่ (bandwidth limitation) ก็ไม่ควรเลือกระบบเอฟเอสเค

2.9 วงจรออสซิลเลเตอร์

ระบบการสื่อสารโดยทั่วๆไปมีความจำเป็นต้องใช้คลื่นรูปไซน์ในการทำงานเป็นอย่างมาก หรืออาจพูดอีกนัยหนึ่งได้ว่าระบบการสื่อสารจะทำงานไม่ได้ ถ้าขาดแหล่งรูปไซน์มีวงจรหลายชนิดที่ใช้ผลิตคลื่นรูปไซน์เหล่านี้ เช่น วงจรออสซิลเลเตอร์แบบป้อนกลับ (feedback oscillator) วงจร RC และวงจร LC ความถี่ที่ผลิตนี้เริ่มตั้งแต่ความถี่เสียงถึงความถี่วีเฮฟเอฟ (VHF)

2.9.1 วงจรออสซิลเลเตอร์ป้อนกลับแบบบวก

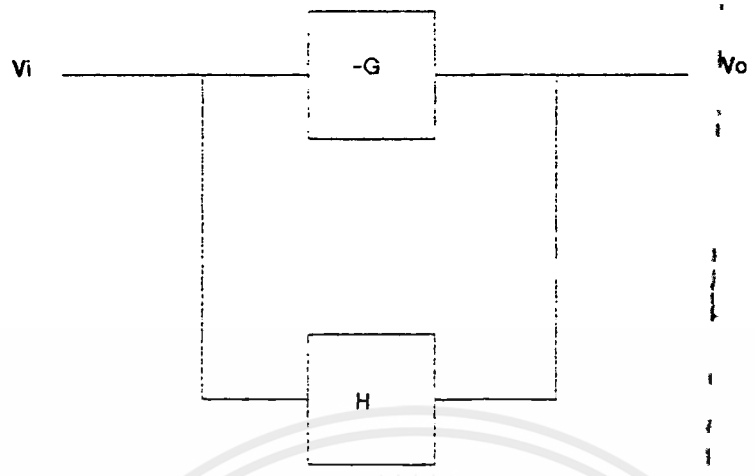
การป้อนกลับแบบบวก (positive feedback) เป็นพื้นฐานของวงจรออสซิลเลเตอร์ โดยทั่วไปที่ใช้กันอยู่ รูปที่ 2.14 แสดงวงจรขยายแบบป้อนกลับ (feedback amplifier) ภายใต้สถานะอันหนึ่งสามารถทำให้วงจรนี้มีการป้อนกลับแบบบวกและกลายเป็นวงจรผลิตความถี่ (oscillator circuit) ขึ้นมา ข้อแม้ของวงจรที่ทำให้เกิดการออสซิลเลทได้นั้น จะต้องมีอัตราการขยายในลูปป้อนกลับ จะมากกว่า 1 เฟสที่เปลี่ยน 90 องศา (phase shift) ในลูปนี้จะต้องเป็นเฟสบวกคูณด้วย 2 เรเดียน หรือ 360 องศา จะต้องเกิดขึ้นในเวลาเดียวกัน จึงจะทำให้เกิดการออสซิลเลทได้

$$\text{อัตราขยายในลูป (loop gain)} = |GH|\theta \quad \text{.....(2.3)}$$

กำหนด $n = 0,1,2,3,.....\theta = n\,360$ องศา และ $|GH| \geq 1$

วงจรขยายโดยทั่วๆไป จะมีเฟสของสัญญาณเอาท์พุทตรงข้ามกลับสัญญาณอินพุท 180 องศา และมีอัตราขยายมากกว่า 1 สมมติให้ G ตามรูปที่ 2.14 คือการขยายวงจร และ H คือการป้อนกลับของลูป ดังนั้น อัตราการขยายของวงจรภายในลูปนี้ คือผลคูณของ G และ H และการขยายทั้งหมดของวงจร (overrall gain) ที่มีการป้อนกลับคือ

$$Av = G/(1-GH) \quad \text{.....(2.4)}$$



รูปที่ 2.14 วงจรออสซิลเลเตอร์ที่ใช้การป้อนกลับแบบบวก

กำหนดให้

Av = การขยายทั้งหมดของวงจร

G = ส่วนการขยายวงจร

H = ส่วนการขยายของการป้อนกลับ

GH = การขยายของลูป

ค่าที่กำหนดทั้งหมดนี้ เป็นค่าคอมเพล็กซ์ (complex) ซึ่งหมายถึง ขนาด และ เฟส (magnitude and phase angle) รวมอยู่ด้วย

ความถี่ของออสซิลเลเตอร์นี้ ถูกกำหนดโดยส่วนประกอบของ H ซึ่งทำให้เฟสเปลี่ยนไป 180 องศา การเลือกค่าขององค์ประกอบเหล่านี้ต้องเลือกด้วยความระมัดระวัง เพื่อให้เกิดเฟสชิฟ 180 องศา ที่ความถี่เดียวกันนี้ และความถี่นี้คือ ความถี่ของออสซิลเลเตอร์ที่นำไปใช้วงจรออสซิลเลเตอร์แบ่งออกเป็น 3 ชนิดคือ

1. วงจรออสซิลเลเตอร์ที่ใช้ LC
2. วงจรออสซิลเลเตอร์ที่ใช้ RC
3. วงจรออสซิลเลเตอร์ที่ใช้ ก้อนผลึก (crystal)

2.9.2 วงจรออสซิลเลเตอร์แบบฮาร์ทเล่ย์

จุดสังเกตุของวงจรแบบนี้ อยู่ที่วงจรจูน LC ที่มีการแท็ปคอยล์สำหรับเป็นวงจร คอยล์ป้อนกลับ (inductive feedback) แทนที่จะเป็นคอยล์ทริกเกอร์แบบแยก จากรูป C1 และ L1 ประกอบกันเป็นวงจรจูน การแท็ปสัญญาณจากคอยล์ L1 ที่จุด G ก็เพื่อเป็นการจ่ายแรงดันคอลเล็คเตอร์ L2 ในวงจร คือ RF โช้ค (choke) จุดแท็ปสัญญาณ G จะต่ออยู่กับกราวด์โดยมี C4 เป็นบายพาสคาปาซิเตอร์อยู่

สัญญาณเอาท์พุทของออสซิลเลเตอร์ จะจ่ายออกที่ขาคอลเลคเตอร์ซึ่งมีระดับแรงดันไฟฟ้าเท่ากับ V_{AG} ซึ่งเป็นความต่างศักย์ระหว่างจุด A บนคอยล์ L1 เทียบกับจุด G ส่วนในด้านตรงข้ามกับจุดแท็ปแรงดันไฟสลับป้อนกลับเท่ากับ V_{BG} ซึ่งถูกคัปปลิ้งโดย C2 ไปยังขาเบสของ Q2 การป้อนกลับของสัญญาณในลักษณะนี้เป็นแบบบวก เพราะมีความต่างเฟส 180 องศา เมื่อเทียบกับ V_{AG} ซึ่งผลลัพธ์ที่เกิดขึ้น จะก่อให้เกิดการออสซิลเลทผลิตสัญญาณไฟสลับจ่ายออกมาที่เอาท์พุท ด้วยความถี่รโซแนนท์ของวงจร LC

พิจารณาที่ระดับแรงดันไฟตรง Vc มีค่าเท่ากับ 28 V เพราะความต้านทานไฟตรงของคอยล์ RF , L1 และ L2 มีค่าน้อยมาก ไม่นำมาคำนวณก็ได้ ขาอิมิตเตอร์มีแรงดันไฟไบอัสตนเองเท่ากับ 1 V จาก R_E โดยมี C_E เป็นตัวรักษาเสถียรภาพของการไบอัสแรงดันไฟฟ้าฟอร์เวิร์ดที่ขาเบสจ่ายผ่าน R1,R2 ซึ่งแบ่งมาจากแหล่งจ่ายไฟ +28 V ดังนั้นค่า $V_{BE} = 1.4 - 1.0 = 0.4$ V ซึ่งน้อยกว่าค่าแรงดันไฟฟ้าคัทออฟ 0.5 V แต่ค่าแรงดันยอดทางด้านบวกของแรงดันไฟฟ้าป้อนกลับจะขับให้ขาเบสมีระดับแรงดันไฟฟ้าเป็นบวก ซึ่งสามารถทำให้ Q1 นำกระแสไฟฟ้าและเกิดการออสซิเลทได้ หน้าที่ของอุปกรณ์แต่ละตัวในวงจร รูปที่ 2.15 สามารถสรุปได้ดังนี้

L1 : อินดักแตนซ์สำหรับวงจรจูน , มีการแท็ปเพื่อป้อนสัญญาณกลับ

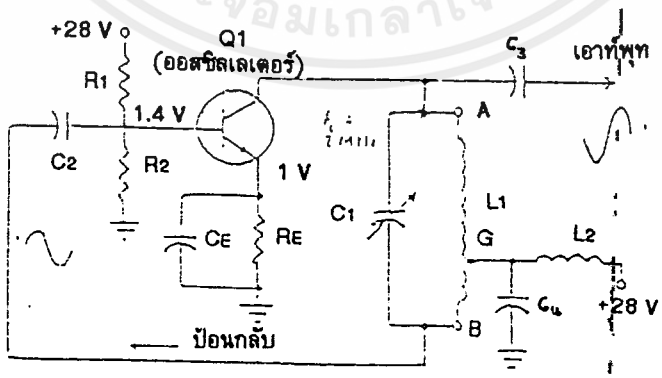
C1 : คาปาซิเตอร์สำหรับวงจรจูน, เปลี่ยนแปลงค่าได้

L2 : RF โช้คทำหน้าที่แยกสัญญาณออสซิลเลเตอร์ออกจากแหล่งจ่ายไฟ

C4 : บายพาสสำหรับต่อจุดแท็ปจาก L1 เพื่อคืนสัญญาณไปสลับลงกราวด์

R1 : ร่วมกับ R2 ป้อนแรงดันไบอัสตรงเข้ากับขาเบส

C2 : คัปปลิ้งสัญญาณป้อนกลับเข้าไปยังขาเบสร่วมกับ C2 เป็นวงจร RC คัปปลิ้งสัญญาณร่วมกับ R1 เป็นวงจรแบ่งแรงดันสำหรับไบอัสขาเบสร่วมกับ C2 เป็นวงจรกรองแบบ RC สำหรับแปลงไฟของสัญญาณไบอัส

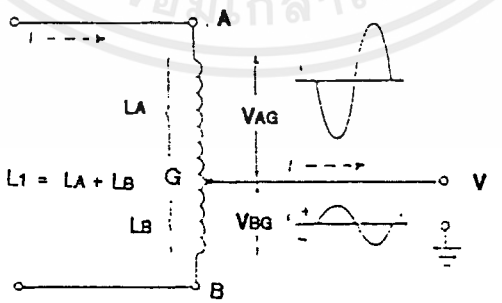


รูปที่ 2.15 วงจรออสซิลเลเตอร์แบบฮาร์ทเล่ย์

2.9.3 การกลับเฟสของสัญญาณแท็ปคอยล์

เหตุผลที่ใช้อธิบายว่าเพราะอะไรการแท็ปสัญญาณของ L1 จึงช่วยให้เกิดการป้อนกลับแบบบวก ก่อนอื่นพิจารณาส่วนของคอยล์ L1 ซึ่งแบ่งออกได้เป็น สองส่วนคือ L_A และ L_B วิเคราะห์การไหลของกระแสอิเล็คตรอนเข้าไปยังจุด A จะเห็นว่าทิศทางการไหลผ่านคอยล์ L_A ระหว่างจุด A กับจุด G แล้วไหลไปสู่แหล่งจ่ายไฟ V+ ซึ่งในกรณีนี้ คอยล์ L_B ไม่มีส่วนเกี่ยวข้องใดๆ กับทิศทางการไหลของกระแสแต่อย่างไรก็ตาม คอยล์ทั้งสองส่วนก็ต่อเนื่องกันอยู่ ดังนั้น L_B จึงเป็นตัวหม้อแปลงคัปปลิ้งสัญญาณไปสู่ L_A ได้ ในการแปรผันของแรงดันไฟสลับ สมมติให้ I มีค่าเพิ่มขึ้นของเลนซ์ (lenz law) จะได้ว่าเกิดการเหนี่ยวนำด้วยตัวเองขึ้น (self induced) เกิดแรงดันไฟฟ้า V_{AG} ซึ่งมีขั้วเป็นลบที่จุด A เพื่อต่อต้านการเพิ่มขึ้นของ I ยิ่งกว่านั้น แรงดันที่เหนี่ยวนำขึ้นมานี้ ส่งผลให้คอยล์ทั้งหมดมีแรงดันไฟฟ้าเป็นลบ และเนื่องจากลักษณะของการพันคอยล์เป็นแบบในทิศทางเดียวกัน ดังนั้นจึงมีสนามแม่เหล็กเหมือนกันตลอดทั้งคอยล์ จุด A ถือว่าเป็นจุดปลายสุดของแรงดันไฟลบที่เหนี่ยวนำขึ้นมาเมื่อเปรียบเทียบกับจุดอื่นๆ หรือขดลวดถัดมาด้านล่าง (ตามรูปที่ 2.16) ส่วนจุด B เมื่อพิจารณาตามแรงดันไฟฟ้าที่เหนี่ยวนำขึ้นมา จุด B จะมีแรงดันเป็นบวกเมื่อเปรียบเทียบกับขดถัดไป ที่อยู่เหนือขึ้นไป (ตามรูปที่2.16) ดังนั้นทั้งจุด A และจุด B จึงมีขั้วตรงกันข้ามเสมอ เมื่อเทียบกับแท็ปนั้น คือ V_{AG} และ V_{BG} จะมีเฟสของสัญญาณต่างกัน 180 องศาเสมอ ในขณะที่จุดหนึ่งเป็นลบมากที่สุด อีกจุดหนึ่งก็จะมีเฟสเป็นบวกมากที่สุด เนื่องจากจุดแท็ป G ต่ออยู่กับกราวด์ เพราะฉะนั้น V_{AG} และ V_{BG} จึงเป็นสัญญาณสลับที่มีขั้วตรงกันข้ามกันเสมอ เมื่อเทียบกับจุดกราวด์

อ้างถึงวงจรรูปที่ 2.16 V_{AG} จะถูกป้อนกลับแบบบวกไปเข้ายังอินพุทของขาเบสโดยทั่วๆไป จุดแท็ปจะป้อนกลับแรงดันไฟฟ้าประมาณ 1 ใน 3 ของแรงดันไฟฟ้าที่ตกคร่อมคอยล์ทั้งหมด



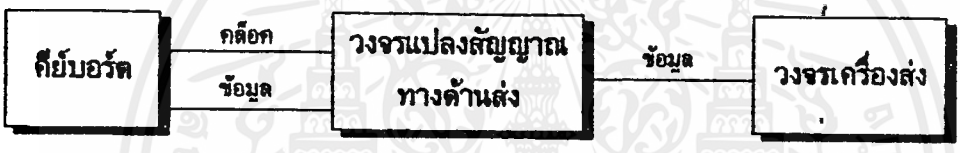
รูปที่ 2.16 เฟสของแรงดันไฟฟ้าในการแท็ปคอยล์ Lb จะต่างเฟส 180 องศา เทียบกับแรงดันไฟฟ้าในคอยล์

บทที่ 3
การออกแบบวงจรและการทำงานของคีย์บอร์ดไร้สาย

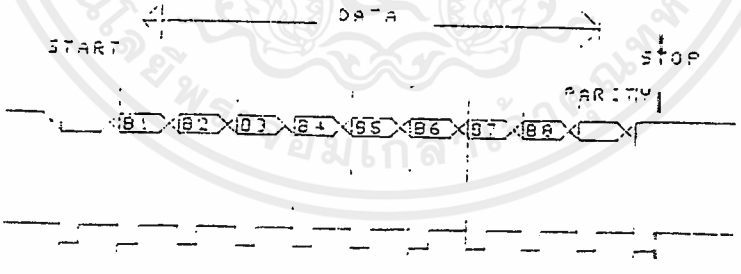
3.1 ภาคส่ง

3.1.1 รายละเอียดของบล็อคไดอะแกรมทางภาคส่ง

จากรูปที่ 3.1 จะเป็นบล็อคไดอะแกรมของทางภาคส่ง ซึ่งส่วนแรกเป็นบล็อคไดอะแกรมของสัญญาณนาฬิกาและข้อมูลซึ่งส่งออกมาจากตัวคีย์บอร์ด ซึ่งจะส่งสัญญาณนี้ออกมาเมื่อมีการกดคีย์และปล่อยคีย์ตามค่ารหัสสแกนตามหลักการที่กล่าวมาในบทที่ 2 และมีรูปสัญญาณดังแสดงในรูป 3.2 ส่วนรูปที่ 3.1 ในส่วนที่สองจะเป็นส่วนแปลงสัญญาณจากสัญญาณนาฬิกาและข้อมูลที่ซิงโครไนซ์กันออกมาให้เป็นแบบอะซิงโครนัสเพียงช่องเดียว โดยใช้ซีพียู 8031 จัดการในส่วนนี้

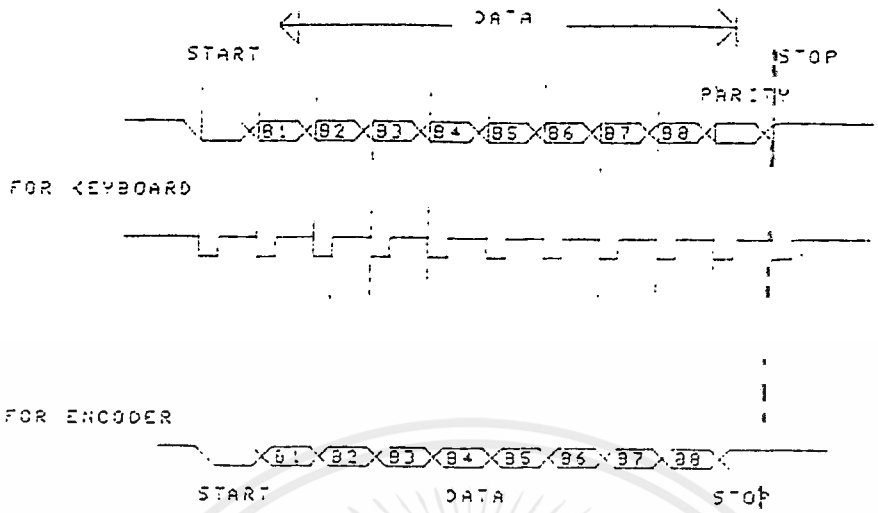


รูปที่ 3.1 บล็อคไดอะแกรมของภาคส่ง



รูปที่ 3.2 สัญญาณที่ส่งออกมาจากคีย์บอร์ด

จากบล็อคไดอะแกรมส่วนที่สองของรูปที่ 3.1 จะเป็นส่วนของวงจรแปลงสัญญาณนาฬิกาและข้อมูลที่ซิงโครไนซ์กันออกมาให้เป็นแบบอะซิงโครนัส เพื่อจะได้ลดจำนวนช่องในการส่งข้อมูลทั้งสัญญาณนาฬิกาและข้อมูลที่ส่งมาจากคีย์บอร์ด ให้เหลือสัญญาณเพียงช่องเดียว ดูจากรูปที่ 3.3



รูปที่ 3.3 รูปสัญญาณ

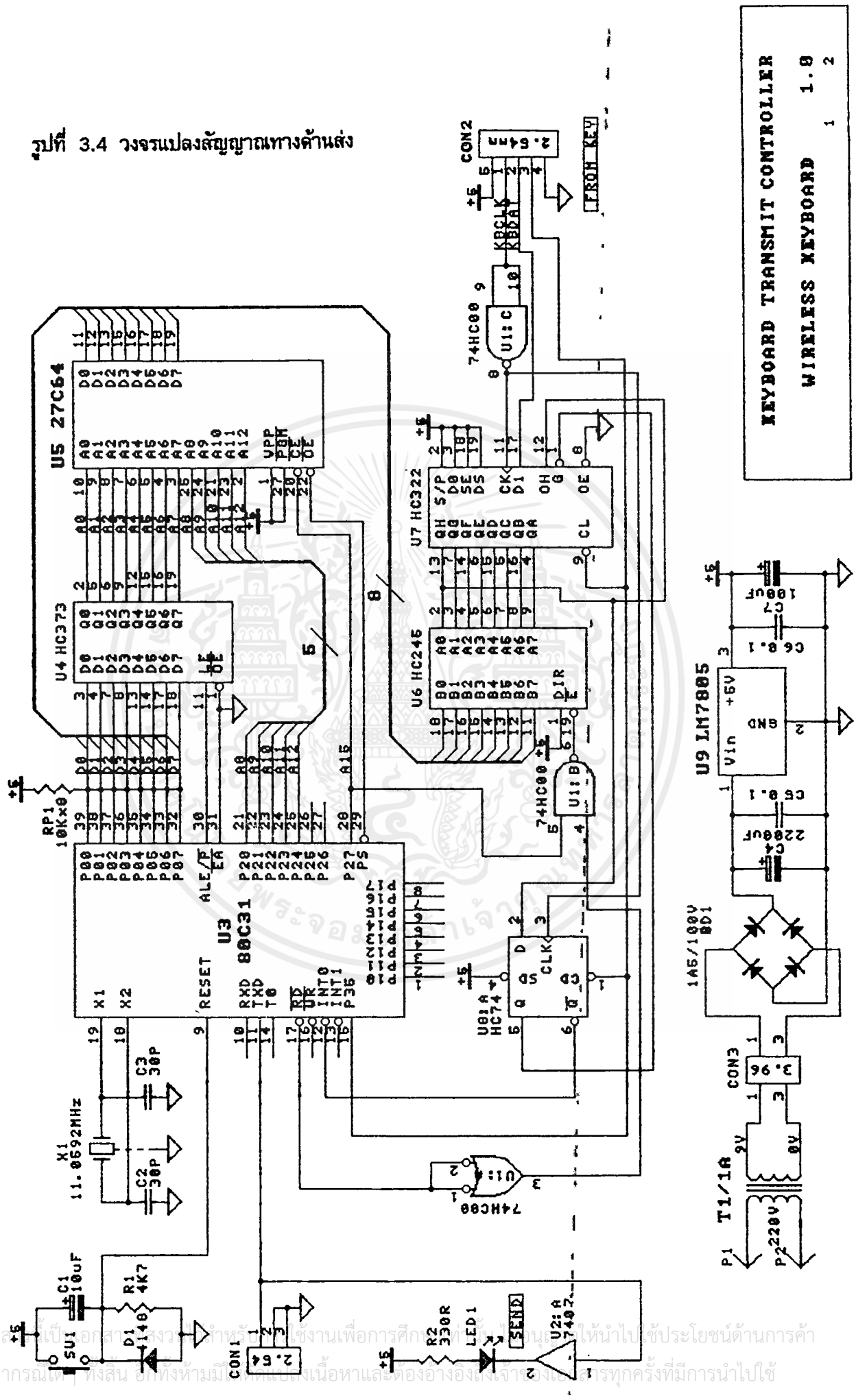
3.1.2 การทำงานของวงจรแปลงสัญญาณทางด้านส่ง

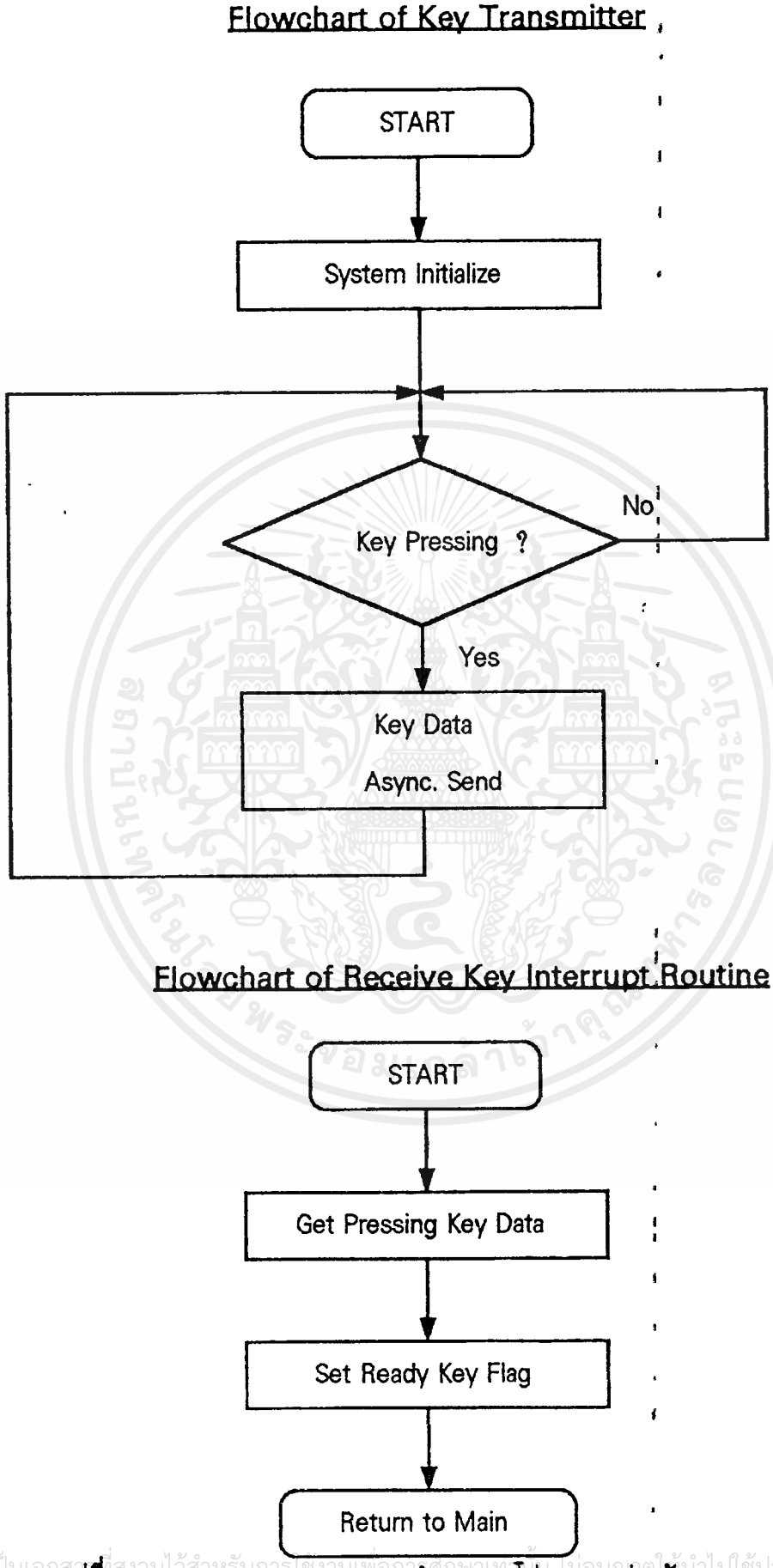
จากวงจรรูปที่ 3.4 เป็นวงจรแปลงทางด้านส่ง โดยมี IC เบอร์ 74HC 322 ซึ่งเป็น 8-bit shift register เป็นตัวแปลงสัญญาณอนุกรม ให้เป็นแบบขนาน เมื่อชิฟข้อมูลครบ 8 บิท OH จะส่งบิท 1 ไปเป็นอินพุทของ D ฟลิบฟลอบ คือ IC HC 74 ทำให้ขา $\overline{Q}$ เป็น 0 ป้อนให้ 8031 ที่ INT0 เพื่อให้ 8031 รู้ว่ามีข้อมูลของคีย์มารออยู่ 8031 ก็จะส่งค่า 0 มาออกทางขา $\overline{RD}$ เข้าขา $\overline{E}$ ของ 245 เพื่อให้ HC 245 ส่งข้อมูลไปยัง 2764 จากนั้น 8031 จะโหลดข้อมูลจาก 2764 เมื่อ 8031 รับข้อมูลมาแล้ว จะส่งพัลส์ที่เป็น low เล็กๆ มาเคลียร์ D ฟลิบฟลอบ และ HC 322 เพื่อให้พร้อมที่จะรับข้อมูลเข้ามาใหม่ ซึ่งข้อมูลที่ 8031 รับเข้ามานั้น จะถูกแปลงเป็นข้อมูลแบบอะซิงโครนัส โดยกำหนดให้ส่งที่บอดเรท 2400 โดยส่งออกมาที่พอร์ทอนุกรม TX ส่งไปให้วงจรเครื่องส่ง เพื่อทำการโมดูเลทแบบแอมพลิจูดชิฟคีอิ่ง (ASK) แล้วส่งออกอากาศไป

3.1.3 FLOW CHART ของวงจรแปลงสัญญาณทางภาคส่ง

ส่วนแรกของ FLOW CHART เป็นส่วนของการกำหนดค่าเริ่มต้นของโปรแกรม (Initial) โดยกำหนดให้ส่งข้อมูลในโหมด 1 เป็นการส่งแบบอะซิงโครนัส โดยมีบิทเริ่มต้น และบิทข้อมูลจำนวน 8 บิท ไม่มีพาริตี้บิท และเลือกโหมดของไทเมอร์ เพื่อใช้กำหนดอัตราการส่งข้อมูล (Baud rate) จากนั้นจะทำการตรวจสอบการกดคีย์ เมื่อมีการกดคีย์บอร์ดจะมีการอินเตอรัพต์เกิดขึ้น ซึ่งใน Interrupt Routine นั้น จะมีการเซ็ตบิท F0 (Key Flag) ไว้เพื่อให้ Main Program ทำการตรวจสอบการกดคีย์ โดยถ้า F0 = 0 แสดงว่า ไม่มีการกดคีย์ แต่ถ้า F0 = 1 แสดงว่า มีการกดคีย์เกิดขึ้น เมื่อ Main Program ตรวจสอบพบว่าบิท F0 = 1 ก็จะทำงานตามโปรแกรมโดยส่งข้อมูลแบบอะซิงโครนัสออกไป

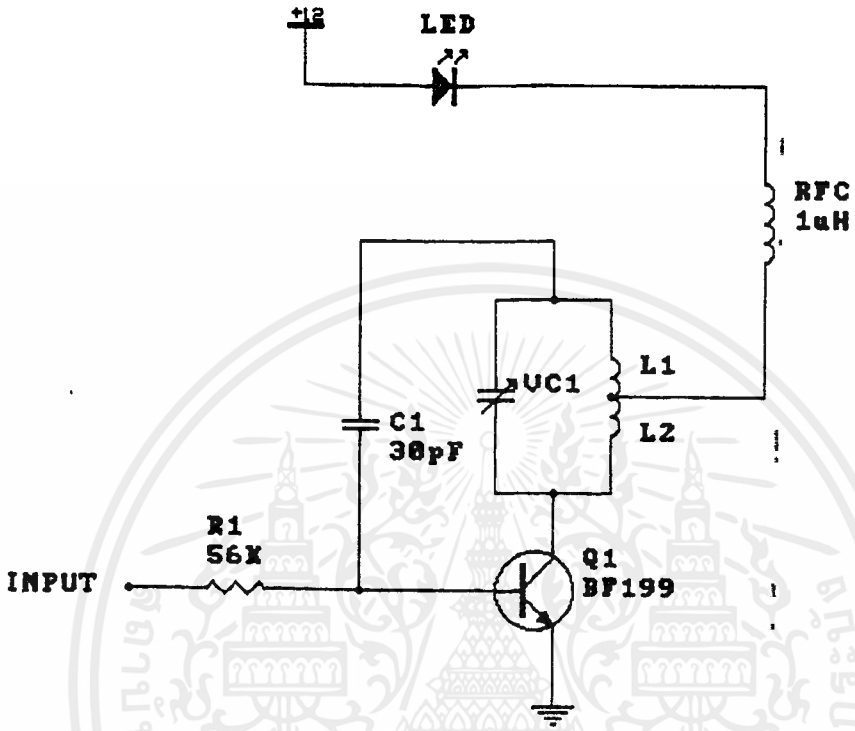
รูปที่ 3.4 วงจรแปลงสัญญาณทางด้านส่ง





รูปที่ 3.5 FLOW CHART **การทำงานของโปรแกรมแปลงสัญญาณทางภาคส่ง**

3.1.4 การทำงานของวงจรเครื่องส่ง



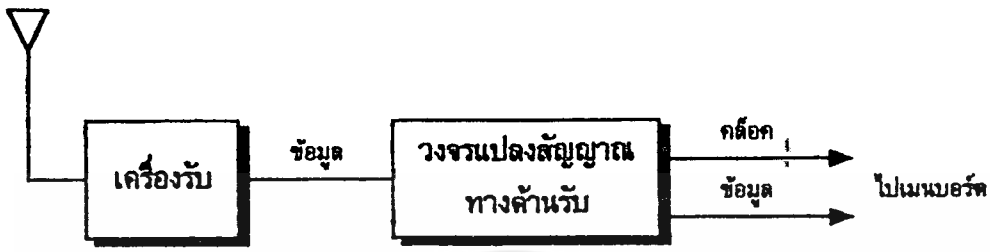
รูปที่ 3.6 วงจรเครื่องส่ง

สำหรับเครื่องส่งประกอบด้วย Q1 , VC1 , L1, L2 , C3 และ RFC โดย C1 ทำหน้าที่เป็นสวิตช์เพื่อให้วงจรกำเนิดความถี่ VC1 , L1, L2 ทำการกำเนิดความถี่ แล้วส่งออกอากาศ RFC จะทำหน้าที่แยกการทำงานของวงจรออสซิลเลเตอร์ออกจากภาคจ่ายไฟ สำหรับ L1 และ L2 เป็นตัวเหนี่ยวนำที่สร้างขึ้นจากลายทองแดงของแผ่นวงจรพิมพ์

เอาท์พุทจากวงจรแปลงสัญญาณดีย์บอร์ดทางภาคส่ง จะถูกส่งเข้าวงจรออสซิลเลเตอร์ เพื่อส่งออกอากาศไปยังเครื่องรับ ดังนั้นคลื่นที่ส่งออกไป จะประกอบด้วยข้อมูลจากวงจรแปลงสัญญาณทางภาคส่ง และคลื่นพาหะที่มีความถี่สูงในย่าน UHF ประมาณ 300 MHz

3.2 ภาครับ

3.2.1 รายละเอียดบล็อคไดอะแกรมทางด้านรับ



รูปที่ 3.7 บล็อคไดอะแกรมทางภาครับ

จากรูปที่ 3.7 ส่วนที่สองจะเป็นวงจรแปลงสัญญาณทางด้านรับ ทำหน้าที่แปลงสัญญาณแบบอะซิงโครนัสช่องเดียว ให้เป็นสัญญาณแบบซิงโครนัส โดยสัญญาณนาฬิกาและข้อมูลจะซิงโครไนซ์กันออกมาตามเดิม โดยมีซีพียู 8031 เป็นตัวจัดการในส่วนนี้

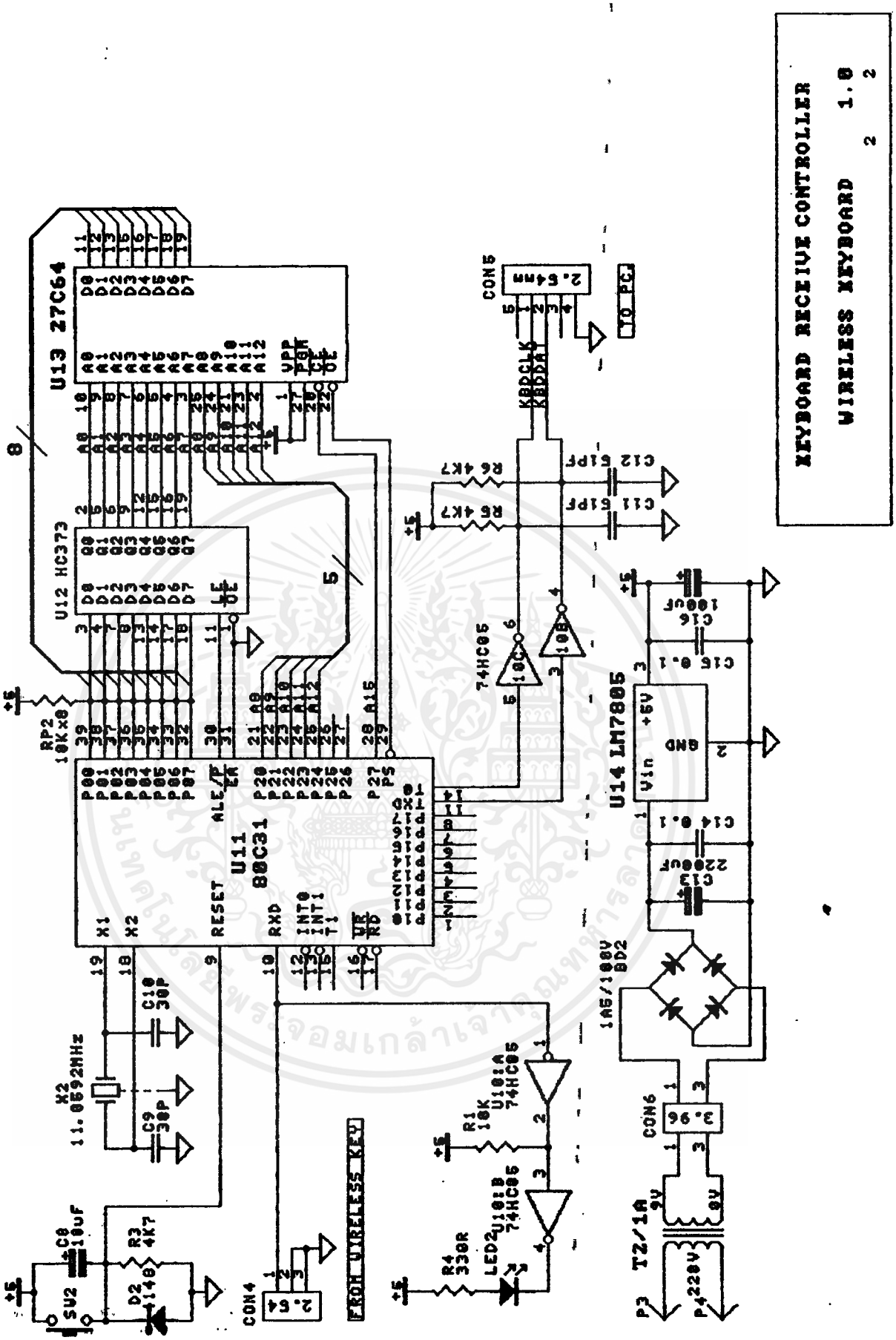
3.2.2 การทำงานของวงจรแปลงสัญญาณทางด้านรับ

จากรูปที่ 3.8 เป็นวงจรแปลงสัญญาณทางภาครับ ซึ่งทำหน้าที่แปลงสัญญาณอะซิงโครนัสที่ได้รับมาจากวงจรด้านส่ง ให้เป็นสัญญาณแบบซิงโครนัส กล่าวคือ มีทั้งสัญญาณนาฬิกา และข้อมูล จะซิงโครไนซ์กันออกมา เหมือนที่ส่งออกมาจากคีย์บอร์ด แล้วส่งไปให้เครื่องคอมพิวเตอร์ โดยมี 8031 เป็นตัวจัดการในส่วนนี้

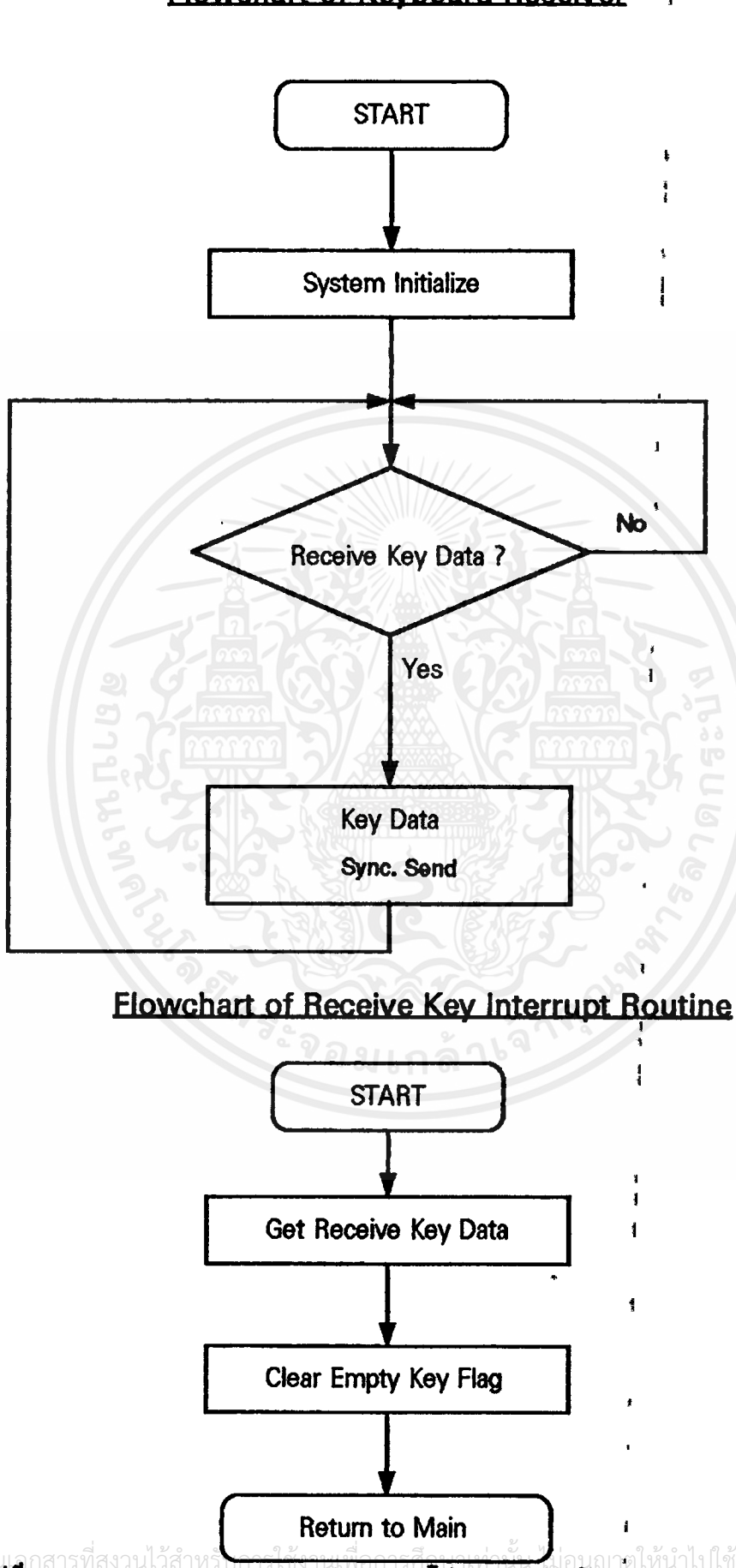
3.2.3 FLOW CHART ของวงจรแปลงสัญญาณทางด้านรับ

ส่วนแรกเป็นส่วนการกำหนดค่าเริ่มต้น คือเลือกโหมดของการรับข้อมูลในโหมด 1 ซึ่งประกอบไปด้วย บิทเริ่มต้น และตามด้วยบิทข้อมูลจำนวน 8 บิท ไม่มีการตรวจสอบพาริตี้บิท และเลือกโหมดของไทเมอร์ เพื่อกำหนดอัตราการส่งข้อมูล (Baud rate) ให้ตรงกับทางด้านส่ง

จากนั้น Main Program จะทำการตรวจสอบการรับข้อมูล โดยเมื่อมีข้อมูลเข้ามาจะเกิดการอินเตอรัพต์เกิดขึ้น ซึ่งใน Interrupt Routine จะมีเซ็ต Flag RI ไว้เพื่อให้ Main Program ตรวจสอบการรับข้อมูลได้ ถ้า RI = 0 แสดงว่ามีข้อมูลยังไม่เต็ม Buffer ยังนำข้อมูลออกไม่ได้ แต่ถ้า RI = 1 แสดงว่าข้อมูลเข้ามาเต็ม Buffer แล้ว สามารถนำข้อมูลออกไปได้ เมื่อ Main Program ตรวจสอบแล้วพบว่า RI = 1 ก็จะทำงานตามโปรแกรมต่อไปโดยส่งข้อมูลแบบซิงโครนัสออกไป



รูปที่ 3.8 วงจรแปลงสัญญาณทางด้านรับ

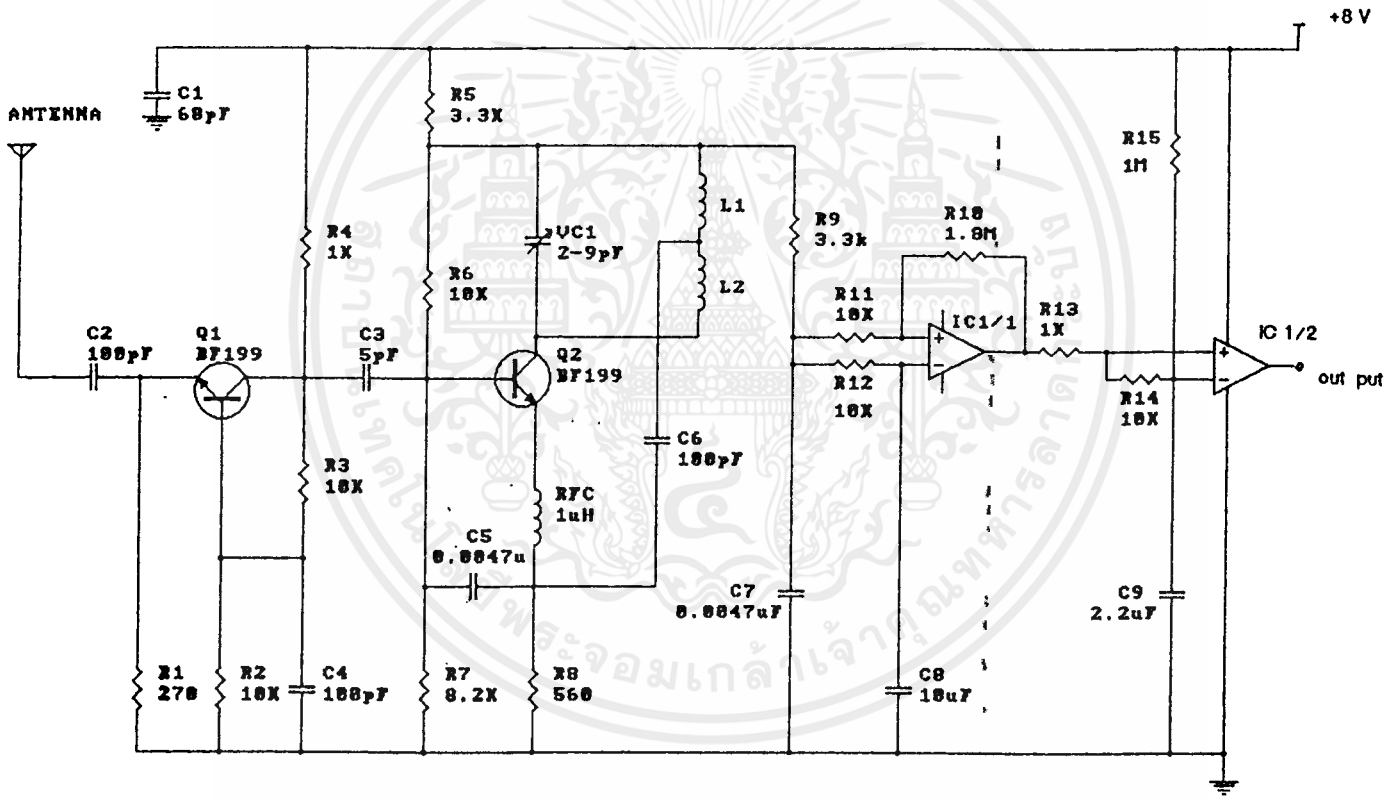


รูปที่ 3.9 FLOW CHART การทำงานของโปรแกรมแปลงสัญญาณทางภาครับ

3.2.4 การทำงานของวงจรเครื่องรับ

คลื่นที่ส่งมาจากทางภาคส่ง จะเข้ามาทางสายอากาศ ผ่าน Q1 ที่ต่อเป็นวงจรขยายเบสร่วมที่มีอัตราการขยายแรงดันสูงมาก ทำหน้าที่ขยายสัญญาณที่รับได้ ป้อนเข้าวงจรจูนที่ประกอบด้วย Q2 , VC1 , L1 , และ L2 สำหรับ L1 และ L2 นี้ เป็นตัวเหนี่ยวนำที่สร้างขึ้นจากลายทองแดงของวงจรพิมพ์เช่นเดียวกับคลื่นส่ง

เมื่อสัญญาณผ่านวงจรจูนออกมาแล้ว จะเข้าสู่วงจรจัดแต่งรูปคลื่น ที่ประกอบด้วย IC1/1 และ IC1/2 ต่อร่วมกับตัวต้านทาน และตัวเก็บประจุ ทำหน้าที่เป็นตัวจัดแต่งรูปคลื่น โดยรับสัญญาณมาจาก Q2 สัญญาณเอาท์พุทของ IC1/2 จะมีลักษณะเป็นพัลส์ที่มีข้อมูลรวมอยู่ด้วย จากนั้นสัญญาณนี้จะถูกส่งต่อให้กับวงจรแปลงสัญญาณทางภาครับ



รูปที่ 3.10 วงจรเครื่องรับ

บทที่ 4
ผลการทดลอง

เนื่องจากโครงงานนี้ แบ่งการทำงานออกเป็น 2 ส่วนใหญ่ๆ คือส่วนของวงจรแปลงสัญญาณทางภาคส่งและทางภาครับ กับส่วนของวงจรเครื่องส่งและเครื่องรับ ดังนั้นในการตรวจสอบวงจรจึงได้ทำการตรวจสอบแยกส่วนกัน โดยในส่วนของวงจรแปลงสัญญาณทางด้านส่งและด้านรับ จะใช้เครื่อง logic analyzer ในการตรวจสอบสัญญาณ I/p และ o/p ของวงจร เมื่อได้ทำการส่งข้อมูลมาจากคีย์บอร์ด ส่วนในวงจรเครื่องส่งและเครื่องรับจะใช้เครื่อง spectrum analyzer ช่วยในการวัดความถี่ของเครื่องส่งและเครื่องรับและแบนวิดท์ของสัญญาณ

นอกจากนี้ยังได้ทำการทดสอบการทำงานของเครื่องส่งและเครื่องรับ โดยใช้ oscilloscope ในการเปรียบเทียบสัญญาณระหว่างอินพุทที่เข้าเครื่องส่ง กับเอาท์พุทที่ออกจากเครื่องรับว่าสอดคล้องกันหรือไม่

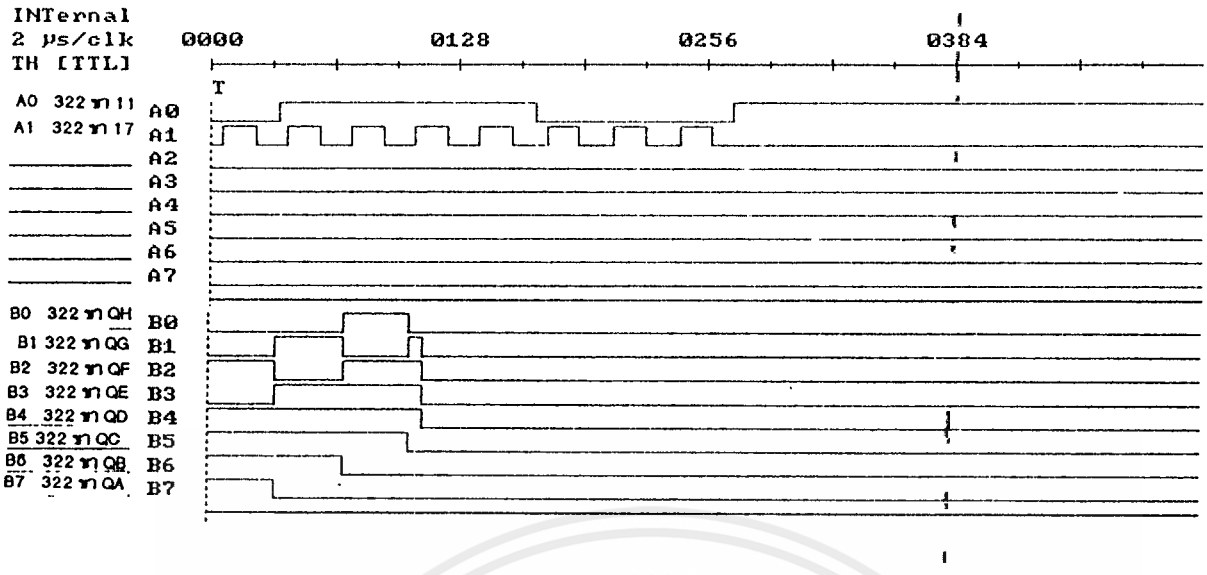
4.1 ทดสอบวงจรแปลงสัญญาณทางด้านส่ง

4.1.1 วัดสัญญาณ keyboard data และ keyboard clock เทียบกับ สัญญาณเอาท์พุทของ 8 - bit shift registers

ทำการต่อคีย์บอร์ดเข้ากับวงจรแปลงสัญญาณทางด้านส่ง แล้วใช้ logic analyzer จับสัญญาณโดยใช้

1. POD A0 จับสัญญาณ keyboard data ที่ขา 11 ของ HC 322
2. POD A1 จับสัญญาณ keyboard clock ที่ขา 17 ของ HC 322
3. POD B0-B7 จับสัญญาณที่ออกจาก HC 322 คือที่ขา QH - QA ตามลำดับ

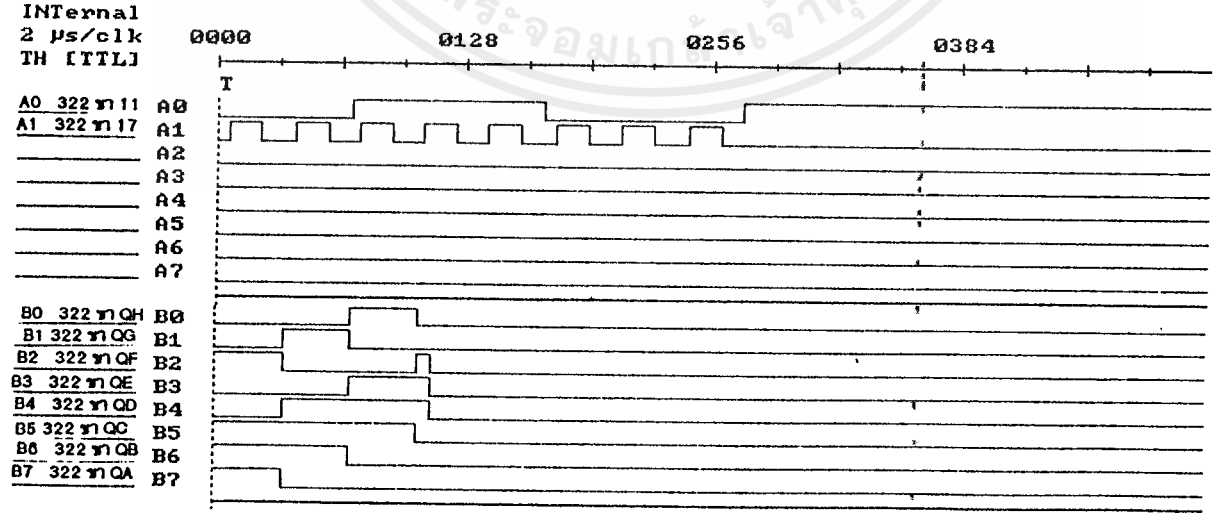
จากการทดสอบพบว่าถ้าไม่ได้กดคีย์ จะไม่มีสัญญาณพัลส์ใดๆ ออกมา แต่เมื่อลองกดคีย์บอร์ดจะสามารถจับสัญญาณพัลส์ของข้อมูลได้ เนื่องจากเราทราบค่ารหัสสแกนของคีย์แต่ละตัวบนคีย์บอร์ด เราจึงตรวจสอบได้ว่าข้อมูลที่ส่งจากคีย์บอร์ดมายัง HC 322 นั้นจะถูกต้องตามรหัสสแกนหรือไม่ อีกทั้งยังตรวจสอบการทำงานของ HC 322 ซึ่งเป็น IC ที่ทำหน้าที่เป็น 8- bit shift registers ว่าทำงานถูกต้องหรือไม่ด้วย ดังนั้นเราจึงทดลองกดคีย์ตัวอักษร "a" ซึ่งมีค่ารหัสสแกน คือ "1E" หรือ "00011110" ซึ่งได้ผลการทดลองดังรูปที่ 4.1



รูปที่ 4.1 แสดงสัญญาณ keyboard data, keyboard clock, o/p ของ 8-bit shift registers เมื่อกดคีย์ตัวอักษร "a"

จากรูปที่ 4.1 จะพบว่า ผลการทดลองที่ได้ถูกต้องตามทฤษฎี คือ ข้อมูลพัลส์ที่ส่งออกมาจาก keyboard ตรงกับค่ารหัสสแกนของตัวอักษร "a" โดยจะเริ่มต้นจาก start bit และตามด้วยข้อมูล 8 bit เริ่มจากบิท 0 ถึง บิท 7 ตามด้วย stop bit โดยจะซิงโครไนซ์มากับสัญญาณ clock ซึ่งถูกกลับเฟสมาจาก 74HC00 แล้วผ่านเข้า HC 322 ซึ่งจะชิฟข้อมูล เริ่มจากบิทแรกคือ D0 ซึ่งจะถูก shift ไปสิ้นสุดที่ QH ทำให้ IC HC 322 มี เอาท์พุท ที่ QH - QA เป็นบิท D0 -D7 ตามลำดับ ซึ่งมีค่าไบนารี่ตรงกับค่ารหัสสแกนของตัว "a" คือ "01111000"

จากการทดลอง เราพบว่า สามารถส่งข้อมูลคีย์บอร์ดออกมาได้ทุกคีย์ แต่ในที่นี้ จะยกมาเพียงบางคีย์เท่านั้น เช่น คีย์ ENTER ซึ่งมีค่ารหัสสแกนคือ "1C" หรือ "00011100" ซึ่งจะได้รูปสัญญาณดังรูปที่ 4.2



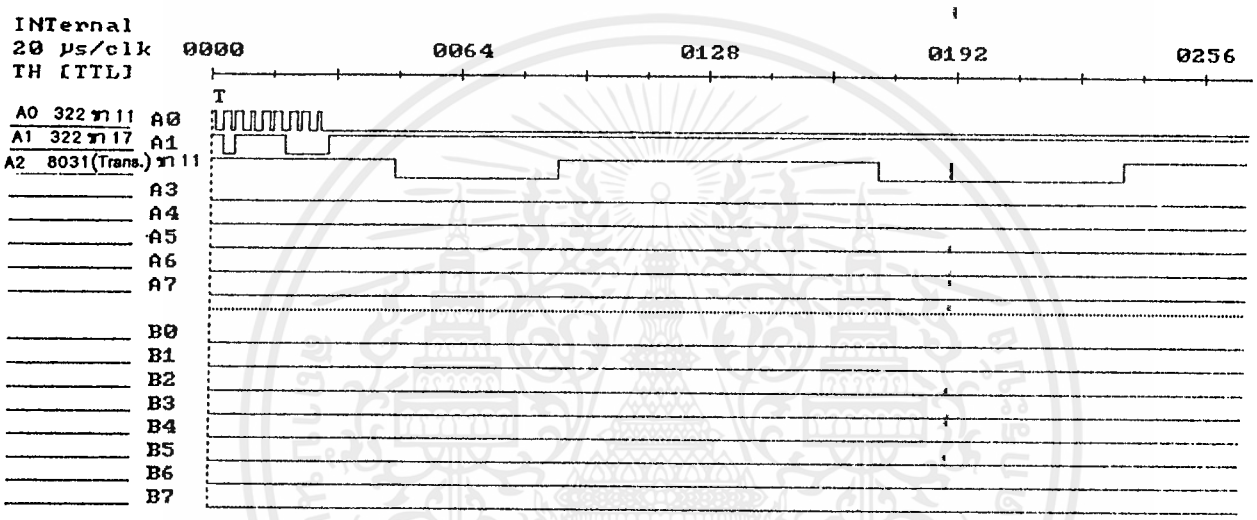
รูปที่ 4.2 แสดงสัญญาณ keyboard data, keyboard clock, o/p ของ 8-bit shift registers เมื่อกดคีย์ "Enter"

4.1.2 จับสัญญาณที่ส่งมาจากคีย์บอร์ดเทียบกับเอาท์พุทที่ออกมาวงจรแปลงสัญญาณ

สัญญาณเอาท์พุทที่ส่งมาจากคีย์บอร์ดคือ สัญญาณข้อมูลกับสัญญาณคล็อค ที่ซิงโครไนซ์กันออกมา กล่าวคือส่งข้อมูลแบบซิงโครนัส ส่วนเอาท์พุทที่ออกจากวงจรเครื่องแปลงสัญญาณนั้น จะเป็นข้อมูลแบบอะซิงโครนัส

ทำการทดลองใช้ POD A0 จับสัญญาณ keyboard clock ที่ขา 11 ของ HC 322 ใช้ POD A1 จับสัญญาณ keyboard data ที่ขา 17 ของ HC 322 ใช้ POD A2 จับสัญญาณเอาท์พุทของวงจรแปลงสัญญาณที่ขา 11 ของ 8031

จากนั้นได้ทดลองกดคีย์ตัวอักษร "a" ซึ่งมีค่ารหัสสแกนเป็น "1E" ได้รูปสัญญาณดังในรูปที่ 4.3



รูปที่ 4.3 แสดงสัญญาณเมื่อกดคีย์ตัว"a"ส่งออกจากคีย์บอร์ดซึ่งเป็นการส่งแบบซิงโครนัส เทียบกับเอาท์พุทที่ออกจากวงจรแปลงสัญญาณซึ่งส่งแบบอะซิงโครนัส

จากรูปที่ 4.3 แสดงให้เห็นว่า พัลส์ของสัญญาณที่ได้นั้น มีความสอดคล้องกัน และมีค่าตรงกับค่ารหัสสแกนของตัวอักษร "a" จากการทดลองพบว่า สามารถส่งข้อมูลออกมาได้ทุกคีย์ และได้ผลในลักษณะเดียวกัน กับตัวอักษร "a"

4.1.3 ทดสอบวงจรแปลงสัญญาณทางด้านส่งโดยใช้ display

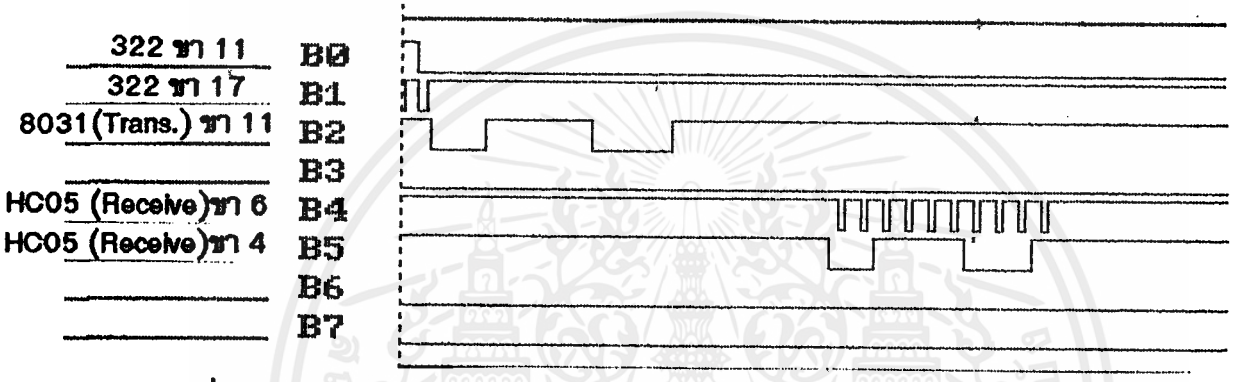
ต่อ display เข้ากับ port 1 ของ 8031 แล้วทดลองกดคีย์ พบว่า display จะแสดงค่ารหัสสแกนของแต่ละคีย์ออกมาได้อย่างถูกต้อง

จากการทดลองทั้งหมดนี้ ทำให้พอจะสรุปได้ว่า วงจรแปลงสัญญาณทางด้านส่งนี้ สามารถทำงานได้ตามวัตถุประสงค์ จึงได้นำวงจรแปลงสัญญาณทางด้านส่งนี้ไปต่อเข้ากับวงจรแปลงสัญญาณทางด้านรับ เพื่อตรวจการทำงานของวงจรแปลงสัญญาณทางด้านรับต่อไป

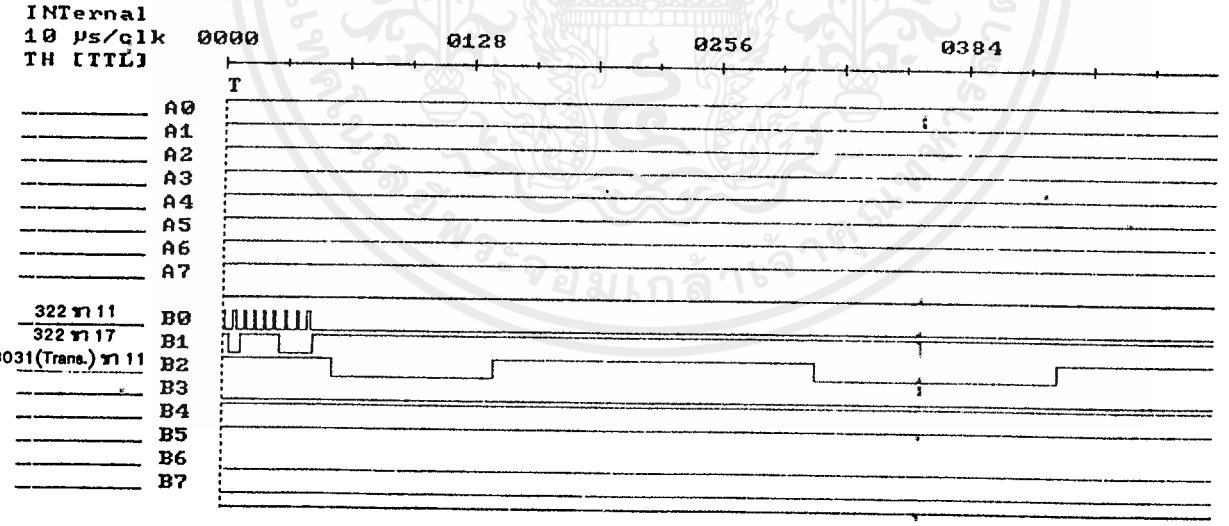
4.2 ทดสอบวงจรแปลงสัญญาณทางด้านรับ

นำวงจรแปลงสัญญาณทางด้านรับต่อกับวงจรแปลงสัญญาณทางด้านส่ง แล้วลองส่งข้อมูลโดยการกดคีย์บนคีย์บอร์ด แล้วใช้ logic analyzer ตรวจจับสัญญาณที่เป็น i/p และ o/p ของ วงจรแปลงสัญญาณทางด้านส่ง ซึ่ง o/p ของวงจรแปลงสัญญาณทางด้านส่งก็คือ i/p ที่จะเข้าวงจรแปลงสัญญาณทางด้านรับ นำสัญญาณเหล่านี้เปรียบเทียบกับสัญญาณ o/p ที่ออกจากวงจรแปลงสัญญาณทางด้านรับ

จากการทดลองกดคีย์บนคีย์บอร์ดทุกคีย์ พบว่าสัญญาณที่ตรวจจับได้แต่ละจุดมีความสอดคล้องกัน และมีค่าไบนารี่ถูกต้องตามค่ารหัสสแกนคีย์ของแต่ละคีย์ แต่ในที่นี้ได้นำมาแสดงให้ดูเพียง ตัวอักษร "a" ดังรูปที่ 4.4 ส่วนในรูปที่ 4.5 เป็นรูปที่ขยายของ POD 0 , POD 1, POD 2



รูปที่ 4.4 แสดงสัญญาณ i/p และ o/p ของวงจรแปลงสัญญาณทางด้านส่งเปรียบเทียบกับ o/p ที่ออกจากวงจรแปลงสัญญาณทางด้านรับ เมื่อกดคีย์ตัวอักษร "a"

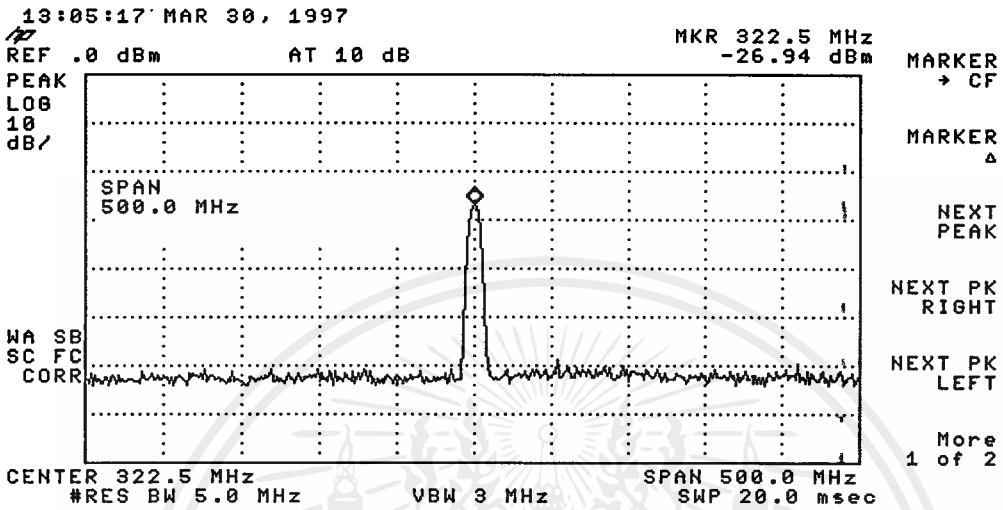


รูปที่ 4.5 รูปขยายของสัญญาณที่ POD0, POD1 และ POD2 ซึ่งเป็น i/p และ o/p ของวงจรแปลงสัญญาณทางด้านส่ง เมื่อกดคีย์ตัวอักษร "a"

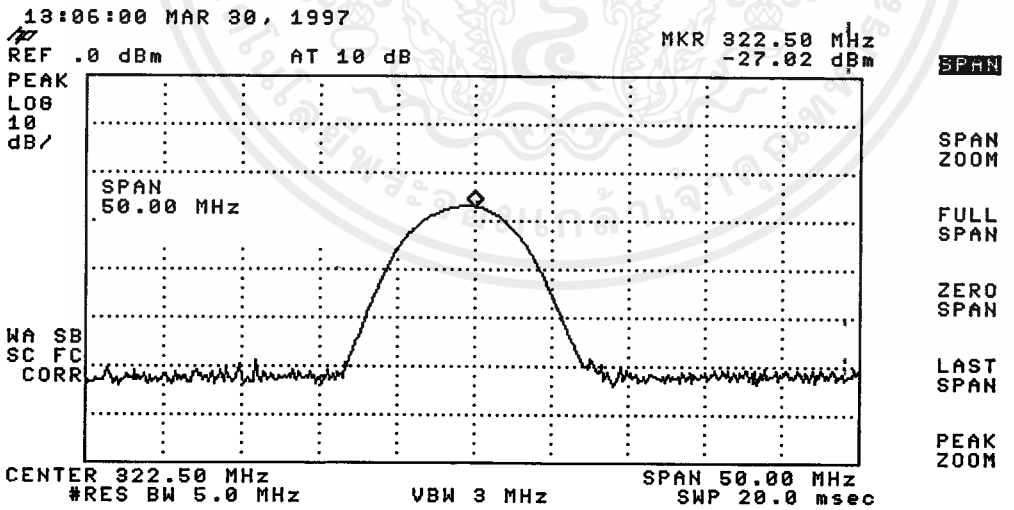
นอกจากนี้ยังได้ทดลองต่อ display ตัวหนึ่งไว้ที่ port 1 ของ 8031 ทางด้านส่ง และต่อ display อีกตัวหนึ่งไว้ที่ port 1 ของ 8031 ทางด้านรับ เพื่อตรวจสอบว่า 8031 ของทางด้านรับและทางด้านส่งส่งค่ารหัสสแกนของแต่ละคีย์ที่กดถูกต้องหรือไม่ ซึ่งผลที่ได้จากการทดลองพบว่าถูกต้องทุกคีย์

4.3 ทดสอบวงจรเครื่องส่งและเครื่องรับ

เมื่อวงจรทุกส่วนทำงานได้แล้ว ก็นำเครื่องส่งไปวัดความถี่และแบนวิดท์ โดยใช้ spectrum analyzer วัดสัญญาณ ซึ่งวัดความถี่ได้ 322.5 MHz วัดแบนวิดท์ได้ 5 MHz ส่วนรูปที่ได้นั้นแสดงได้ดังรูปที่ 4.6 และ ส่วนรูปที่ 4.7 เป็นรูปความถี่ที่ SPAN ออกมา

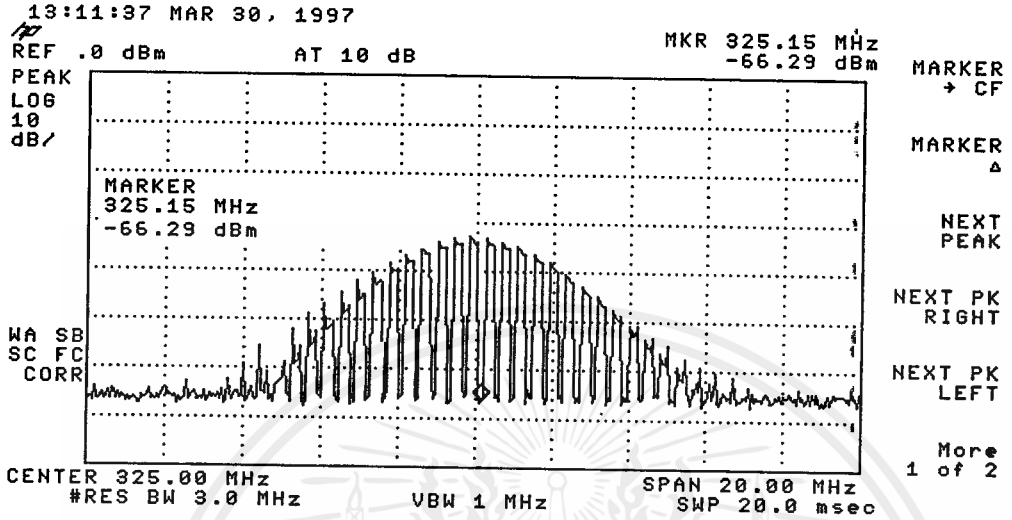


รูปที่ 4.6 รูปแสดงความถี่ที่ส่งโดยเครื่องส่ง



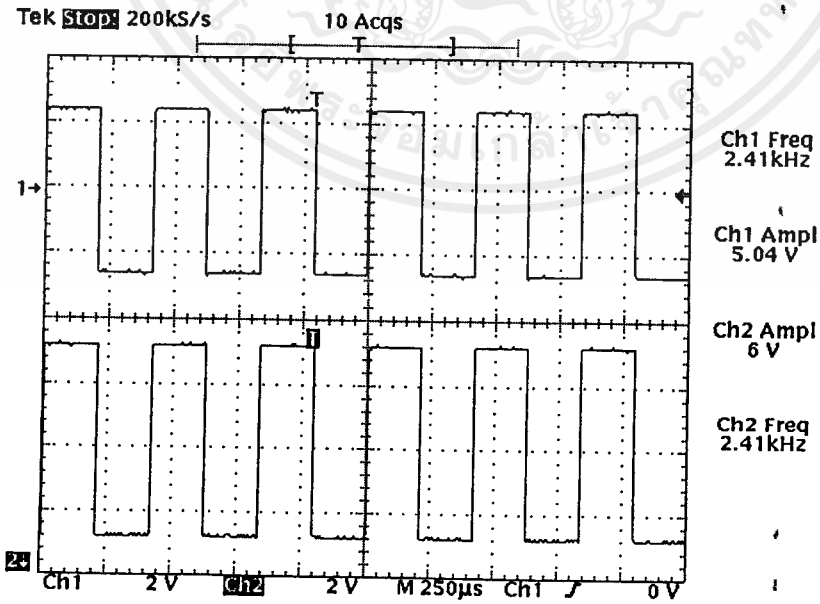
รูปที่ 4.7 รูปความถี่ที่ span ออกมา

นอกจากนี้ได้ทำการทดสอบการทำงานของเครื่องส่ง โดยการป้อนอินพุท 2.5 Vpp ความถี่ 2.4 KHz เพื่อดูรูปสัญญาณที่ถูกโมดูเลทออกมา ดังแสดงในรูปที่ 4.8 ซึ่งพบว่าได้รูปสัญญาณค่อนข้างสมบูรณ์ และมีสัญญาณรบกวนน้อย



รูปที่ 4.8 รูปสัญญาณที่ถูกมอดูเลทและส่งโดยเครื่องส่ง

ในการปรับแต่งเครื่องรับนั้น เราก็ป้อนสัญญาณพัลส์ที่ความถี่ประมาณ 2.4 KHz เข้าที่เครื่องส่ง แล้วปรับค่าคาปาซิเตอร์ปรับค่าได้ ให้ได้รูปสัญญาณที่ตรงกันระหว่างอินพุทเข้าเครื่องส่งกับเอาท์พุทที่ออกจากเครื่องรับ ดังรูปที่ 4.9 เป็นรูปสัญญาณที่ปรับแต่งเรียบร้อยแล้ว



รูปที่ 4.9 รูปสัญญาณ i/p ของเครื่องส่งและ o/p ของเครื่องรับซึ่งได้รับการปรับแต่งแล้ว

จากการทดลองทั้งหมด พบว่าวงจรแต่ละส่วนสามารถทำงานได้เป็นอย่างดี เราจึงนำมาทดลองต่อวงจรแปลงสัญญาณทางด้านส่งและด้านรับ เข้ากับวงจรเครื่องส่งกับเครื่องรับ เพื่อทดสอบว่าสามารถทำงานร่วมกันได้หรือไม่ แต่เนื่องจากคีย์บอร์ดที่เรานำมาใช้ในการทดลองทั้งหมดนี้ เป็นคีย์บอร์ดแบบ XT ซึ่งเป็นคีย์บอร์ดรุ่นเก่า และใช้งานได้กับเฉพาะเครื่อง PC รุ่นเก่า ซึ่งหาค่อนข้างยาก จึงไม่สะดวกที่จะนำมาทดลอง เราจึงต้องใช้ display ในการตรวจสอบการทำงาน และแสดงผล แทนการแสดงผลบนหน้าจอคอมพิวเตอร์ โดยเราจะนำ display 2 ตัว ต่อไว้ที่ port ของ 8031 ทั้งทางด้านรับและทางด้านส่ง เพื่อเปรียบเทียบค่ารหัสสแกนของแต่ละคีย์ที่ส่ง

ถ้า display ทั้งทางด้านรับและทางด้านส่ง ยังแสดงค่ารหัสสแกนตรงกัน ก็แสดงว่าวงจรทำงานได้เป็นปกติดี

รูปที่ 4.10 แสดงเครื่องคีย์บอร์ดไร้สาย

บทที่ 5
บทวิจารณ์และสรุปผล

จากการทำโครงงานวิศวกรรมที่ผ่านมา ทำให้ได้ความรู้เพิ่มมากขึ้น ในด้านการส่งข้อมูลแบบดิจิตอล และการนำสัญญาณมาทำการแปลงข้อมูลในรูปแบบต่างๆ ทำให้ได้รู้จักการนำเอาไมโครคอนโทรลเลอร์มาใช้ในการสื่อสารข้อมูล ตลอดจนการปรับแต่งสัญญาณเครื่องรับและเครื่องส่ง แม้จะเกิดปัญหามากมายในตอนแรก เช่น ปัญหาเกี่ยวกับการปรับแต่งเครื่องรับ ให้รับส่งกันได้ และปัญหาเกี่ยวกับการตรวจจับสัญญาณจากคีย์บอร์ด ซึ่งเป็นสัญญาณที่ไม่ต่อเนื่อง แต่ปัญหาทั้งหมดก็สามารถแก้ไขได้

ส่วนปัญหาใหญ่ที่พบในการทำโครงงานนี้คือ คีย์บอร์ดที่เรานำมาใช้ในโครงงานนี้ เป็นคีย์บอร์ดแบบ XT ซึ่งเป็นคีย์บอร์ดรุ่นเก่า ที่จะต้องทำงานร่วมกับเครื่องคอมพิวเตอร์ชนิด XT ด้วย แต่เนื่องจากผู้ทำโครงงาน ไม่มีเครื่องคอมพิวเตอร์รุ่นนี้ และหาค่อนข้างยาก จึงไม่ได้แสดงผลของการส่งข้อมูลจากคีย์บอร์ดให้ปรากฏบนคอมพิวเตอร์ แต่ใช้ display แทนจอคอมพิวเตอร์ ซึ่งจะแสดงได้เพียงค่ารหัสสแกนของแต่ละคีย์เท่านั้น นอกจากนี้ยังมีปัญหา ในการเชื่อมต่อวงจรแปลงสัญญาณทางด้านส่งเข้ากับเครื่องส่ง และการเชื่อมต่อวงจรแปลงสัญญาณทางด้านรับ เข้ากับเครื่องรับ ซึ่งมีไฟเลี้ยงของแต่ละวงจรไม่เท่ากัน จึงจำเป็นต้องมีการเพิ่มอุปกรณ์ ที่เป็นบัฟเฟอร์และขยายหรือลดแรงดันลงด้วย จึงจะทำให้ทุกวงจรสามารถทำงานร่วมกันได้อย่างมีประสิทธิภาพ

ภาคผนวก

```
                        1   ;Description:      PC Keyboard Transmitter
                        2   ;Hardware:         MCS-51 uController
                        3   ;                  and Interface Board
                        4   ;Assembler:        SXA51 on MS-DOS 6.20
                        5   ;File Name:        KeySend.Asm
                        6
                        7   ;---------- Variable Setting -----
00F4=                   8   Baud        EQU    0F4H            ;2400 Baud Rate
                        9
                       10   ;---------- Port Setting ---------
0090=                  11   Sin         BIT    P1.0            ;Serial In
0091=                  12   Sck         BIT    P1.1            ;Shift  Clock
0092=                  13   Lck         BIT    P1.2            ;Latch  Data
00B5=                  14   KeyClr      BIT    P3.5            ;KeyBoard Clear
                       15
                       16   ;---------- Internal RAM ---------
0030                   17               ORG    30H
0030                   18   DataBuf:    DS     1               ;Key Data Buffer
0031                   19   DispBuf:    DS     2               ;Display Buffer
                       20
                       21   ;---------- Reset System ---------
0000                   22               ORG    0000H
0000 020014            23               LJMP   SFRInit
                       24
                       25   ;---------- Key Interrupt --------
0003 C0E0              26   KeyInt:     PUSH   ACC             ;Save Register
0005 908000            27               MOV    DPTR,#8000H     ;Key Port
0008 E0                28               MOVX   A,@DPTR         ;Get Key Data
0009 F530              29               MOV    DataBuf,A
000B D2D5              30               SETB   F0              ;Set Key Flag
000D C2B5              31               CLR    KeyClr          ;Clear Key Flag
000F D2B5              32               SETB   KeyClr
0011 D0E0              33               POP    ACC
0013 32                34               RETI
                       35
                       36   ;---------- SFR Initial ----------
0014 C2B5              37   SFRInit:    CLR    KeyClr          ;Keyboard Clear
0016 7E0A              38               MOV    R6,#10
0018 120081            39               CALL   DelayxM         ;Reset Delay
001B D2B5              40               SETB   KeyClr
001D C2D5              41               CLR    F0              ;Clear Key Flag
001F C291              42               CLR    Sck             ;Idle  S-Display
0021 C292              43               CLR    Lck
0023 75A881            44               MOV    IE,#10000001B
0026 758920            45               MOV    TMOD,#00100000B
0029 759852            46               MOV    SCON,#01010010B
002C 758DF4            47               MOV    TH1,#Baud       ;Baud  Setting
002F C289              48               CLR    TCON.1          ;Falling Edge Int#0
0031 D28E              49               SETB   TCON.6          ;Start Timer1
0033 753100            50               MOV    DispBuf+0,#0
0036 753200            51               MOV    DispBuf+1,#0
0039 120055            52               CALL   SDisp           ;Clear S-Display
                       53
003C 30D5FD            54   KeyWait:    JNB    F0,KeyWait
003F E530              55               MOV    A,DataBuf       ;Get Key Data
0041 F531              56               MOV    DispBuf,A
0043 12004D            57               CALL   ASend           ;Send Key Data
0046 120055            58               CALL   SDisp           ;Display Key Data
0049 C2D5              59               CLR    F0
004B 80EF              60               SJMP   KeyWait
```

```
                99        ;------------ Serial Display -------
009A C0E0       100       SDisp:      PUSH    ACC
009C C000       101                   PUSH    00H
009E C001       102                   PUSH    01H
00A0 C002       103                   PUSH    02H
00A2 C291       104                   CLR     Sck             ;Idle State
00A4 C292       105                   CLR     Lck
00A6 7832       106                   MOV     R0,#DispBuf+1
00A8 7902       107                   MOV     R1,#2           ;No.Byte
00AA 7A08       108       SDisp1:     MOV     R2,#8           ;No.Bit
00AC E6         109                   MOV     A,@R0           ;Display Data
00AD 33         110       SDisp2:     RLC     A
00AE 9290       111                   MOV     Sin,C
00B0 D291       112                   SETB    Sck             ;Clock Active
00B2 C291       113                   CLR     Sck
00B4 DAF7       114                   DJNZ    R2,SDisp2
00B6 18         115                   DEC     R0
00B7 D9F1       116                   DJNZ    R1,SDisp1
00B9 D292       117                   SETB    Lck             ;Latch Active
00BB C292       118                   CLR     Lck
00BD D002       119                   POP     02H
00BF D001       120                   POP     01H
00C1 D000       121                   POP     00H
00C3 D0E0       122                   POP     ACC
00C5 22         123                   RET
                124
                125       ;---------- 1uSec.Delay ----------
00C6 7F32       126       Delay1U:    MOV     R7,#50          ;Delay 1uSec.
00C8 DFFE       127                   DJNZ    R7,$
00CA 22         128                   RET
                129
                130       ;---------- XmSec.Delay ----------
00CB 1200D1     131       DelayxM:    CALL    Delay1M         ;Delay XmSec.
00CE DEFB       132                   DJNZ    R6,DelayxM
00D0 22         133                   RET
                134
                135       ;---------- 1mSec.Delay ----------
00D1 7FE5       136       Delay1M:    MOV     R7,#229         ;Delay 1mSec.
00D3 00         137       Delay1M1:   NOP
00D4 00         138                   NOP
00D5 DFFC       139                   DJNZ    R7,Delay1M1
00D7 22         140                   RET
0000=           141                   END
```

```
                1       ;Description:       PC Keyboard Receiver
                2       ;Hardware:          MCS-51 uController
                3       ;                   and Interface Board
                4       ;Assembler:         SXA51 on MS-DOS 6.20
                5       ;File Name:         KeyRecv.Asm
                6
                7       ;---------- Variable Setting -----
00F4=           8       Baud        EQU     0F4H            ;2400 Baud Rate
                9
               10       ;---------- Port Setting ---------
0090=          11       Sin         BIT     P1.0            ;Serial In
0091=          12       Sck         BIT     P1.1            ;Shift  Clock
0092=          13       Lck         BIT     P1.2            ;Latch  Data
00B4=          14       KbClk       BIT     P3.4            ;Keyboard Clock
00B1=          15       KbDat       BIT     P3.1            ;Keyboard Data
               16
               17       ;---------- Internal RAM ---------
0030           18                   ORG     30H
0030           19       DataBuf:    DS      1               ;Key Data Buffer
0031           20       DispBuf:    DS      2               ;Display Buffer
               21
               22       ;---------- Reset System ---------
0000           23                   ORG     0000H
0000 7E0A      24                   MOV     R6,#10
0002 1200CB    25                   CALL    DelayxM         ;Reset Delay
0005 C2B4      26                   CLR     KbClk           ;Idle  Keyboard
0007 C2B1      27                   CLR     KbDat
0009 C291      28                   CLR     Sck             ;Idle  S-Display
000B C292      29                   CLR     Lck
000D 75A890    30                   MOV     IE,#10010000B
0010 75B810    31                   MOV     IP,#00010000B
0013 758920    32                   MOV     TMOD,#00100000B
0016 759852    33                   MOV     SCON,#01010010B
0019 758DF4    34                   MOV     TH1,#Baud       ;Baud  Setting
001C D28E      35                   SETB    TCON.6          ;Start Timer1
001E 8010      36                   SJMP    Main
               37
               38       ;---------- Key Interrupt --------
0023           39                   ORG     0023H           ;Receive Interrupt
0023 C298      40       KeyInt:     CLR     RI              ;Clear RxD Flag
0025 C0E0      41                   PUSH    ACC             ;Save Register
0027 E599      42                   MOV     A,SBUF          ;Get  Key Data
0029 F530      43                   MOV     DataBuf,A
002B D2D5      44                   SETB    F0              ;Set  Key Flag
002D D0E0      45                   POP     ACC
002F 32        46                   RETI
               47
0030 753100    48       Main:       MOV     DispBuf+0,#0
0033 753200    49                   MOV     DispBuf+1,#0
0036 12009A    50                   CALL    SDisp           ;Clear S-Display
0039 C2D5      51       Main1:      CLR     F0              ;Clear Key Flag
003B 30D5FD    52       KeyWait:    JNB     F0,KeyWait
003E E530      53                   MOV     A,DataBuf       ;Get Key Data
0040 F531      54                   MOV     DispBuf,A
0042 4400      55                   ORL     A,#00H
0044 85D0D5    56                   MOV     F0,PSW.0        ;Save Parity Flag
0047 12009A    57                   CALL    SDisp
004A 12004F    58                   CALL    ASend           ;Send  Key Data
004D 80EA      59                   SJMP    Main1
```

```
                62      ;---------- ASync.Send ----------
004D 3099FD     63      ASend:      JNB     TI,$            ;Tx Buffer Empty?
0050 C299       64                  CLR     TI              ;Clear Tx Flag
0052 F599       65                  MOV     SBUF,A          ;Data to Tx Buffer
0054 22         66                  RET
                67
                68      ;---------- Serial Display -------
0055 C0E0       69      SDisp:      PUSH    ACC
0057 C000       70                  PUSH    00H
0059 C001       71                  PUSH    01H
005B C002       72                  PUSH    02H
005D C291       73                  CLR     Sck             ;Idle State
005F C292       74                  CLR     Lck
0061 7832       75                  MOV     R0,#DispBuf+1
0063 7902       76                  MOV     R1,#2           ;No.Byte
0065 7A08       77      SDisp1:     MOV     R2,#8           ;No.Bit
0067 E6         78                  MOV     A,@R0           ;Display Data
0068 33         79      SDisp2:     RLC     A
0069 9290       80                  MOV     Sin,C
006B D291       81                  SETB    Sck             ;Clock Active
006D C291       82                  CLR     Sck
006F DAF7       83                  DJNZ    R2,SDisp2
0071 18         84                  DEC     R0
0072 D9F1       85                  DJNZ    R1,SDisp1
0074 D292       86                  SETB    Lck             ;Latch Active
0076 C292       87                  CLR     Lck
0078 D002       88                  POP     02H
007A D001       89                  POP     01H
007C D000       90                  POP     00H
007E D0E0       91                  POP     ACC
0080 22         92                  RET
                93
                94      ;---------- XmSec.Delay ----------
0081 120087     95      DelayxM:    CALL    Delay1M         ;Delay XmSec.
0084 DEFB       96                  DJNZ    R6,DelayxM
0086 22         97                  RET
                98
                99      ;---------- 1mSec.Delay ----------
0087 7FE5      100      Delay1M:    MOV     R7,#229         ;Delay 1mSec.
0089 00        101      Delay1M1:   NOP
008A 00        102                  NOP
008B DFFC      103                  DJNZ    R7,Delay1M1
008D 22        104                  RET
0000=          105                  END
```

กิตติกรรมประกาศ

รายงานฉบับนี้สำเร็จลุล่วงลงได้ โดยได้รับคำแนะนำทางด้านความรู้ การให้คำปรึกษาจากอาจารย์ที่ปรึกษา ทางผู้จัดทำจึงกราบขอบพระคุณอาจารย์ รศ. ดร. วิวัฒน์ กิรานนท์ ไว้ ณ ที่นี้ พร้อมทั้ง อาจารย์ วิภา แสงพิสิทธิ์ ที่ให้คำแนะนำในการทำโครงงานในครั้งนี้ ขอบคุณ คุณพ่อ คุณแม่ ที่คอยเป็นกำลังใจให้เสมอ และขอขอบคุณ คุณ พงศ์พันธ์ ศุภกาญจนกันติ ที่ให้ความช่วยเหลือในการใช้เครื่องมืออุปกรณ์ต่าง ๆ รวมทั้งรุ่นพี่ที่มหานครทุกคน ที่ช่วยเหลือให้คำแนะนำในการใช้เครื่องมือ และอุปกรณ์ต่าง ๆ ในการทำโครงงานครั้งนี้ รวมทั้งเพื่อน ๆ ทุกคน ที่ให้ความช่วยเหลือในเรื่องข้อมูล

เอกสารอ้างอิง

[1] บัญทิต จามรภูติ , " ฮาร์ดแวร์ไมโครคอมพิวเตอร์ 8088,80286,80386" ซีเอ็ดยูเคชั่น, พ.ศ. 2521

[2] ประสิทธ์ ประพิฒมงคลการ ดร," หลักการระบบสื่อสาร" ซีเอ็ดยูเคชั่น ,พ.ศ. 2521

[3] ปีเตอร์ นอร์ตัน ," เข้าใจการทำงาน IBM PC INSIDE THE IBM PC AND PS/2" ซีเอ็ดยูเคชั่น , พ.ศ. 2521

[4] ยืน ภู่วรวรรณ รศ, "เทคโนโลยี ฮาร์ดแวร์ IBM PC" ซีเอ็ดยูเคชั่น , พ.ศ. 2521

[5] FREDRICK F. DRISCOLL , "DATA COMMUNICATIONS" ,SAUNDERS COLLEGE PUBLISHING, 1992

[6] IBM TECHNICAL REFERENCE

[7] สุเจตน์ จันทรังษ์ , " ไมโครคอนโทรลเลอร์ชิพเดี่ยว 8051" วิทยาลัยมหานคร, พ.ศ. 2535

PRELIMINARY

AUGUST 1988

DATA SHEET 80C51-L / 80C31-L

CMOS SINGLE-CHIP 8 BIT 3V-MICROCONTROLLER

- **80C51-L – CMOS SINGLE-SHIP 8-BIT MICROCONTROLLER with factory mask-programmable ROM**
- **80C31-L - CMOS SINGLE-CHIP 8-BIT CONTROL-ORIENTED CPU with RAM and I/O**
- **80C51-L/C31-L: 0 TO 6 MHz, VCC = 2.7V TO 6V**

FEATURES

- POWER CONTROL MODES
- 128 x 8 BIT RAM
- 32 PROGRAMMABLE I/O LINES
- TWO 16-BIT TIMER/COUNTERS
- 64K PROGRAM MEMORY SPACE
- FULLY STATIC DESIGN
- HIGH PERFORMANCE SAJI VI CMOS PROCESS
- BOOLEAN PROCESSOR
- 5 INTERRUPT SOURCES
- PROGRAMMABLE SERIAL PORT
- 64K DATA MEMORY SPACE
- TEMPERATURE RANGE: 0 TO 70°C

DESCRIPTION

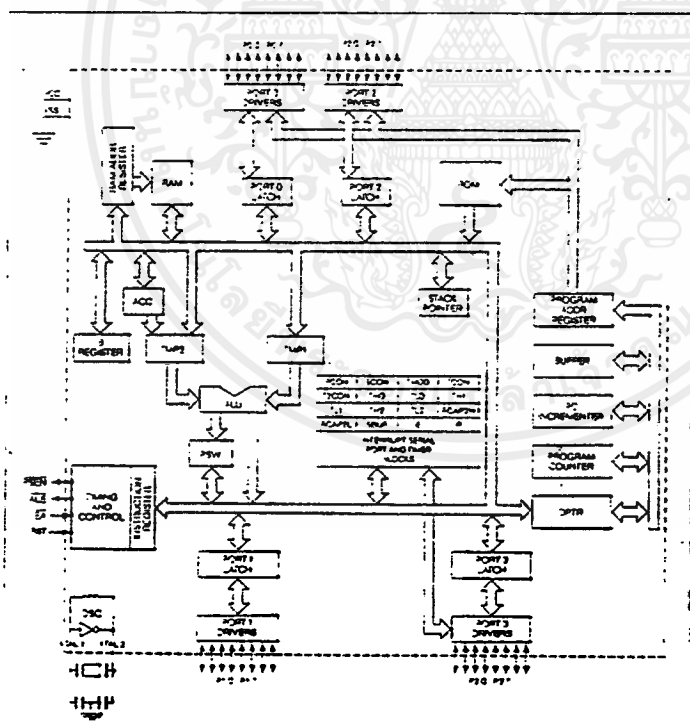


Figure 1. Block Diagram

MHS's 80C51 and 80C31 are high performance CMOS versions of the 8051/8031 NMOS single chip 8 bit μC and is manufactured using a self-aligned silicon gate CMOS process (SAJI VI).

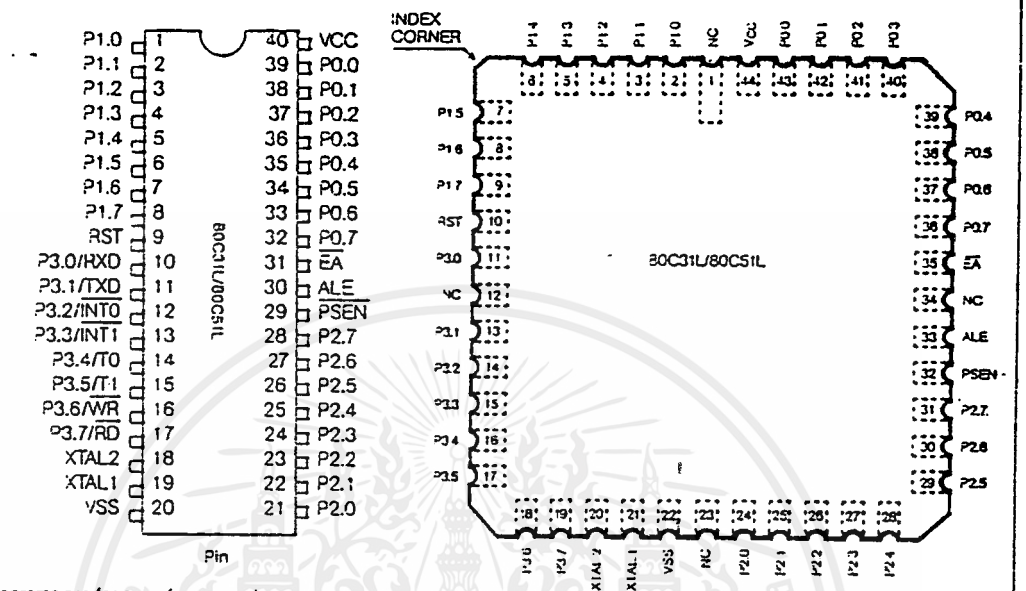
The fully static design of the MHS 80C51/80C31 allows to reduce system power consumption by bringing the clock frequency down to any value, even DC, without loss of data.

The 80C51 retains all the features of the 8051: 4K bytes of ROM; 128 bytes of RAM; 32 I/O lines; two 16 bit timers; a 5-source 2-level interrupt structure; a full duplex serial port; and on-chip oscillator and clock circuits.

In addition, the 80C51 has two software-selectable modes of reduced activity for further reduction in power consumption. In the Idle Mode the CPU is frozen while the RAM, the timers, the serial port, and the interrupt system continue to function. In the Power Down Mode the RAM is saved and all other functions are inoperative.

The 80C31 is identical to the 80C51 except that it has no on-chip ROM.

Figure 2. Configurations



Diagrams are for pin reference only.
Package sizes are not to scale.

IDLE AND POWER DOWN OPERATION

Figure 3 shows the internal Idle and Power Down clock configuration. As illustrated, Power Down operation stops the oscillator. Idle mode operation allows the interrupt, serial port, and timer blocks to continue to function while the clock to the CPU is gated off. These special modes are activated by software via the Special Function Register, its hardware address is 87H. PCON is not bit addressable.

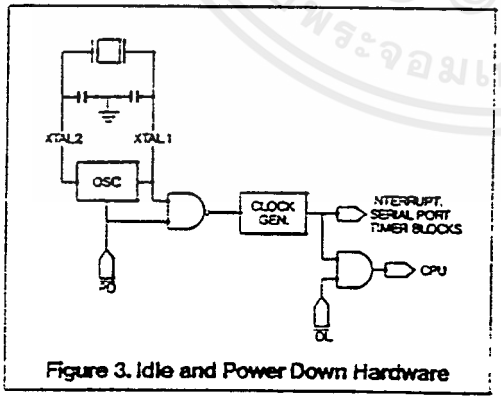


Figure 3. Idle and Power Down Hardware

PCON: Power Control Register

(MSB) (LSB)

SMOD	-	-	-	GF1	GF0	PD	IDL

Symbol	Position	Name and Function
SMOD	PCON.7	Double Baud rate bit. When set to a 1, the baud rate is doubled when the serial port is being used in either modes 1, 2 or 3.
-	PCON.6	(Reserved)
-	PCON.5	(Reserved)
-	PCON.4	(Reserved)
GF1	PCON.3	General-purpose flag bit.
GF0	PCON.2	General-purpose flag bit.
PD	PCON.1	Power Down bit. Setting this bit activates power down operation.
IDL	PCON.0	Idle mode bit. Setting this bit activates idle mode operation.

If 1's are written to PD and IDL at the same time, PD takes precedence. The reset value of PCON is (0XXX0000).

Table 1. Status of the external pins during Idle and Power Down modes

Mode	Program Memory	ALE	PSEN	PORT0	PORT1	PORT2	PORT3
Idle	Internal	1	1	Port Data	Port Data	Port Data	Port Data
Idle	External	1	1	Floating	Port Data	Address	Port Data
Power Down	Internal	0	0	Port Data	Port Data	Port Data	Port Data
Power Down	External	0	0	Floating	Port Data	Port Data	Port Data

IDLE MODE

The instruction that sets PCON.0 is the last instruction executed before the Idle mode is activated. Once in the Idle mode the CPU status is preserved in its entirety: the Stack Pointer, Program Counter, Program Status Word, Accumulator, RAM, and all other registers maintain their data during Idle. *Table 1* describes the status of the external pins during Idle mode.

There are two ways to terminate the Idle mode. Activation of any enabled interrupt will cause PCON.0 to be cleared by hardware, terminating Idle mode. The interrupt is serviced, and following RETI, the next instruction to be executed will be the one following theinstruction that wrote a 1 to PCON.0.

The flag bits GF0 and GF1 may be used to determine whether the interruct was received during normal execution or during the Idle mode. For example, the instruction that writes to PCON.0 can also set or clear one or both flag bits. When Idle mode is terminated by an enabled interrupt, the service routine can examine the status of the flag bits.

The second way of terminating the Idle mode is with a hardware reset. Since the oscillator is still running, the hardware reset needs to be active for only 2 machine cycles (24 oscillator periods) to complete the reset operation.

POWER DOWN MODE

The instruction that sets PCON.1 is the last executed prior to entering power down. Once in power down, the oscillator is stopped. The contents of the onchip RAM and the Special Function Register is saved during power down mode. A hardware reset is the only way of exiting the power down mode. The hardware reset initiates the Special Function Register *(see Table 1)*.

In the Power Down mode. V_{CC} may be lowered to minimize circuit power consumption. Care must be taken to ensure the voltage is not reduced until the power down mode is entered, and that the voltage is restored before the hardware reset is applied which frees the oscillator. Reset should not be released until the oscillator has restarted and stabilized.

Table 1 describes the status of the external pins while in the power down mode. It should be noted that if the power down mode is activated while in external program memory, the port data that is held in the Special Function Register P2 is restored to Port 2. If the data is a 1, the port pin is held high during the power down mode by the strong pullup, T1, shown in *Figure 4*.

STOP CLOCK MODE

Due to static design, the MHS 80C31/C51 clock speed can be reduced until 0 MHz without any data loss in memory or registers. This mode allows step by step utilization, and permits to reduce system power consumption by bringing the clock frequency down to any value. At 0 MHz, the power consumption is the same as in the Power Down Mode.

80C51 I/O PORTS

The I/O port drive of the 80C51 is similar to the 8051. The I/O buffers for Ports 1, 2, and 3 are implemented as shown in *figure 4*.

When the port latch contains a 0, all pFETS in *figure 4* are off while the nFET is turned on. When the port latch makes a 0-to-1 transition, the nFET turns off. The strong pullup pFET, T1, turns on for two oscillator periods, pulling the output high very rapidly. As the output line is drawn high, pFET T3 turns on through the inverter to supply the IOH source current. This inverter and T3 form a latch which holds the 1 and is supported by T2. When Port 2 is used as an address port, for access to external program of data memory, any address bit that contains a 1 will have his strong pullup turned on for the entire duration of the external memory access.

When an I/O pin on Ports 1, 2, or 3 is used as an input, the user should be aware that the external circuit must sink current during the logical 1-to-0 transition. The maximum sink current is specified as ITL under the D.C.Specifications. When the input goes below approximately 2V, T3 turns off to save ICC current. Note, when returning to a logical 1, T2 is the only internal pullup that is on. This will result in a slow rise time if the user's circuit does not force the input line high.

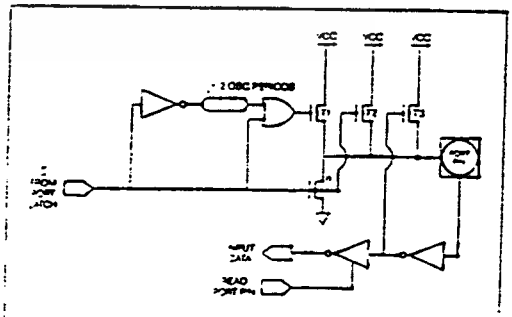


Fig. 4. I/O Buffers in the 80C51 (Ports 1, 2, 3)

80C51 PIN DESCRIPTIONS

V_{SS}
Circuit ground potential

V_{CC}
Supply voltage during normal, Idle, and Power Down operation.

Port 0
Port 0 is an 8-bit open drain bi-directional I/O port. Port 0 pins that have 1's written to them float, and in that state can be used as high-impedance inputs.

Port 0 is also the multiplexed low-order address and data bus during accesses to external Program and Data Memory. In this application it uses strong internal pullups when emitting 1's. Port 0 also outputs the code bytes during program verification in the 80C51. External pullups are required during program verification. Port 0 can sink eight LS TTL inputs.

Port 1
Port 1 is an 8-bit bi-directional I/O port with internal pullups. Port 1 pins that have 1's written to them are pulled high by the internal pullups, and in that state can be used as inputs. As inputs, Port 1 pins that are externally being pulled low will source current (IIL, on the data sheet) because of the internal pullups.

Port 1 also receives the low-order address bytes during program verification. In the 80C51 Port 1 can sink/source three LS TTL inputs. It can drive CMOS inputs without external pullups.

Port 2
Port 2 is an 8-bit bi-directional I/O port with internal pullups. Port 2 pins that have 1's written to them are pulled high by the internal pullups, and in that state can be used as inputs. As inputs, Port 2 pins that are externally being pulled low will source current (IIL, on the data sheet) because of the internal pullups. Port 2 emits the high-order address byte during fetches from external Program Memory and during accesses to external Data Memory that use 16-bit addresses (MOVX @ DPTR). In this application, it uses strong internal pullups when emitting 1's. During accesses to external Data Memory that uses 8-bit addresses (MOVX @ Ri), Port 2 emits the contents of the P2 Special Function Register.

It also receives the high-order address bits and control signals during progam verification in the 80C51. Port 2 can sink/source three LS TTL inputs. It can drive CMOS inputs without external pullups.

Port 3
Port 3 is an 8-bit bi-directional I/O port with internal pullups. Port 3 pins that have 1's written to them are pulled high by the internal pullups, and in that state can be used as inputs. As inputs, Port 3 pins that are externally being pulled low will source current (IIL, on the data sheet) because of the pullups. It also serves the functions of various special features of the MCS-51 Family, as listed below.

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

Port 3 can sink/source three LS TTL inputs. It can drive CMOS inputs without external pullups.

RST
A high level on this for two machine cycles while the oscillator is running resets the device. An internal pull-down resistor permits Power-On reset using only a capacitor connected to V_{CC}.

ALE
Address Latch Enable output for latching the low byte of the address during accesses to external memory. ALE is activated as though for this purpose at a constant rate of 1/6 the oscillator frequency except during an external data memory access at which time one ALE pulse is skipped. ALE can sink/source 8 LS TTL inputs. It can drive CMOS inputs without an external pullup.

$\overline{PSEN}$
Program Store Enable output is the read strobe to external Program Memory. PSEN is activated twice each machine cycle during fetches from external Program Memory. (However, when executing out of external Program Memory, two activations of PSEN are skipped during each access to external Data Memory). PSEN is not activated during fetches from Internal Program Memory. PSEN can sink/source 8 LS TTL inputs. It can drive CMOS inputs without an external pullup.

$\overline{EA}$
When EA is held high, the CPU executes out of internal Program Memory (unless the Program Counter exceeds 0FFFH). When EA is held low, the CPU executes only out of external Program Memory. EA must not be floated.

XTAL1
Input to the inverting amplifier that forms the oscillator. Receives the external oscillator signal when an external oscillator is used.

XTAL2
Output of the inverting amplifier that forms the oscillator, and input to the internal clock generator. This pin should be floated when an external oscillator is used.

OSCILLATOR CHARACTERISTICS

XTAL1 and XTAL2 are the input and output respectively, of an inverting amplifier which is configured for use as an on-chip oscillator, as shown in figure 5. Either a quartz crystal or ceramic resonator may be used. To drive the device from an external clock source, XTAL1 should be driven while XTAL2 is left unconnected as shown in *figure 6*. There are no requirements on the duty cycle of the external clock signal, since the input to the internal clocking circuitry is through a divide-by-two flip-flop, but minimum and maximum high and low times specified on the Data Sheet must be observed.

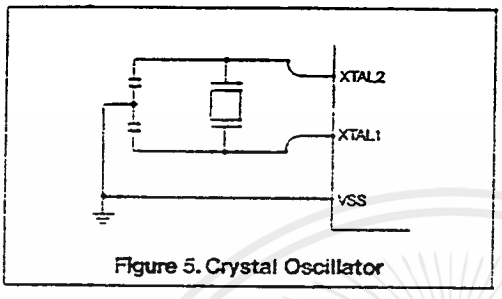


Figure 5. Crystal Oscillator

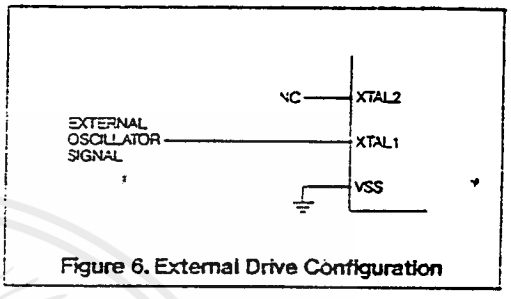


Figure 6. External Drive Configuration

ABSOLUTE MAXIMUM RATINGS*

Ambient Temperature Under Bias:
Commercial ..... 0°C to 70°C
Industrial ..... −40°C to 85°C
Storage Temperature ..... −65°C to +150°C
Voltage on V_{CC} to V_{SS} ..... −0.5V to +7V
Voltage on Any Pin to V_{SS} ..... −0.5V to V_{CC} +0.5V
Power Dissipation 1W*

*This value is based on the maximum allowable die temperature and the thermal resistance of the package.

***NOTICE:**

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

DC CHARACTERISTICS

T_A = −40°C to 85°C; V_{CC} = 2.7V to 6V; V_{SS} = 0V; F = 0 to 6 MHz

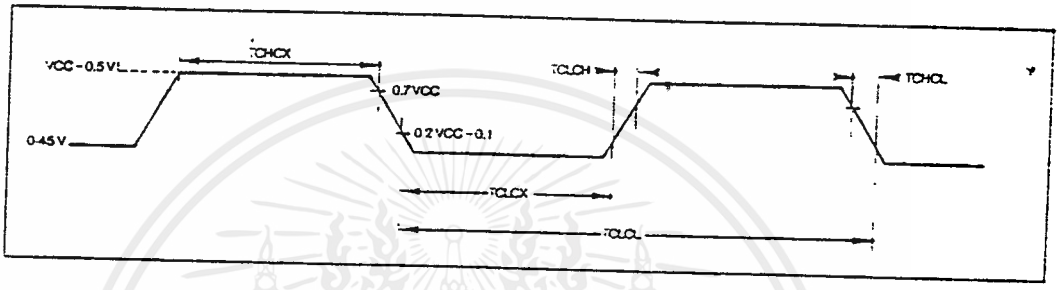
Symbol	Parameter	Min	Max	Unit	Test Conditions
V_{IL}	Input Low Voltage	−0.5	$0.2V_{CC}$ −0.1	V	
V_{IH}	Input High Voltage (Except XTALs and RST)	$0.2V_{CC}$ −0.9	V_{CC} −0.5	V	
V_{IH1}	Input High Voltage to RST for Reset	$0.7V_{CC}$	V_{CC} −0.5	V	
V_{IH2}	Input High Voltage To XTAL 1	$0.7V_{CC}$	V_{CC} −0.5	V	
V_{PD}	Power Down Voltage To V_{CC} In PD Mode	2.0	5.0	V	
V_{OL}	Output Low Voltage (Ports 1, 2, 3)		0.45	V	I_{OL} = 1.6mA (note 1)
V_{OL1}	Output Low Voltage Port 0, ALE, $\overline{PSEN}$		0.45	V	I_{OL} = 3.2mA (note 1)
V_{OH}	Output High Voltage Ports 1, 2, 3	$0.9V_{CC}$		V	I_{OH} = −10µa
		2.4		V	I_{OH} = −60µA, V_{CC} = 5V ± 10%
V_{OH1}	Output High Voltage (Port 0 in External In External Bus Mode), ALE, $\overline{PSEN}$	$0.9V_{CC}$		V	I_{OH} = −40µA
		2.4		V	I_{OH} = −400µA, V_{CC} = 5V ± 10%
I_{IL}	Logical 0 Input Current Ports 1,2,3		−50	µA	Vin = 0.45V
I_{LI}	Input Leakage Current		± 10	µA	0.45 < Vin < V_{CC}
I_{TL}	Logical 1 to 0 Transition Current (Ports 1, 2, 3)		−500	µA	Vin = 2.0V
I_{CCPD}	Power Supply Current (Power Down Mode)	50	10	µA	V_{CC} = 2.0V to 5.5V (note 2)
R_{RST}	RST Pulldown Resistor	50	150	kΩ	
C_{IO}	Capacitance of I/O Buffer		10	pF	f_C = 1MHz, T_A = 25°C

Note 1:

Capacitive loading on Ports 0 and 2 may cause spurious noise pulses to be superimposed on the V_{OL}s of ALE and Ports 1 and 3. The noise is due to external bus capacitance discharging into the Port 0 and Port 2 pins when these pins make 1-to-0 transitions during bus operations. In the worst cases (capacitive loading 100 pF), the noise pulse on the ALE line may exceed 0.45V with maxi V_{OL} peak 0.6V. A Schmitt Trigger use is not necessary.

EXTERNAL CLOCK DRIVE CHARACTERISTICS (XTAL1)

Symbol	Parameter	Variable Clock freq = 0 to 6 MHz		Unit
		Min	Max	
TCLCL	Oscillator Period	166		ns
TCHCX	High Time	20		ns
TCLCX	Low Time	20		ns
TCLCH	Rise Time		20	ns
TCHCL	Fall Time		20	ns



AC CHARACTERISTICS

(T_A = −40°C to 85°C, V_{CC} = 2.7 V to 6 V, V_{SS} = 0 V)
(Load Capacitance for Port 0, ALE, and PSEN = 100 pF; Load Capacitance for All Other Outputs = 80 pF).

EXTERNAL PROGRAM MEMORY CHARACTERISTICS

Symbol	Parameter	Min	Max	Units
TLHLL	ALE Pulse Width	2TCLCL − 40		ns
TAVLL	Address Valid to ALE	TCLCL − 55		ns
TLLAX	Address Hold After ALE	TCLCL − 35		ns
TLLIV	ALE to Valid Instr In		4TCLCL − 170	ns
TLLPL	ALE to PSEN	TCLCL − 25		ns
TPLPH	PSEN Pulse Width	3TCLCL − 35		ns
TPLIV	PSEN to Valid Instr In		3TCLCL − 220	ns
TPXIX	Input Instr Hold After PSEN	0		ns
TPXIZ	Input Instr Float After PSEN		TCLCL − 20	ns
TPXAV	PSEN to Address Valid	TCLCL − 8		ns
TAVIV	Address to Valid Instr In		5TCLCL − 220	ns
TPLAZ	PSEN Low to Address Float		0	ns

See next page for External Data Memory Characteristics.

EXTERNAL DATA MEMORY CHARACTERISTICS

Symbol	Parameter	Min	Max	Units
TRLRH	RD Pulse Width	6TCLCL – 100		ns
TWLWH	WR Pulse Width	6TCLCL – 100		ns
TLLAX	Data Address Hold After ALE	TCLCL – 35		ns
TRLDV	RD to Valid Data In		5TCLCL – 165	ns
TRHDX	Data Hold After RD	0		ns
TRHDZ	Data Float After RD		2TCLCL – 70	ns
TLLDV	ALE to Valid Data In		8TCLCL – 150	ns
TAVDV	Address to Valid Data In		9TCLCL – 165	ns
TLLWL	ALE to WR or RD	3TCLCL – 50	3TCLCL – 50	ns
TAVWL	Address to WR or RD	4TCLCL – 130		ns
TQVWX	Data Valid to WR Transition	TCLCL – 60		ns
TQVWH	Data Setup to WR High	7TCLCL – 150		ns
TWHQX	Data Hold After WR	TCLCL – 50		ns
TRLAZ	RD Low to Address Float		0	ns
TWHLH	RD or WR High to ALE High	TCLCL – 40	TCLCL – 40	ns

MAXIMUM ICC (mA)

	Operating (Note 3)			Idle (Note 4)		
Freq. VCC	2.7V	5V	6V	2.7V	5V	6V
1 MHz	0.8 mA	1.6 mA	1.8 mA	400 µA	900 µA	1 mA
8 MHz	2 mA	8 mA	10 mA	1.2 mA	3.5 mA	3.8 mA

Note 2:

Power Down ICC is measured with all output pins disconnected; EA = Port 0 = VCC; XTAL2 N.C.; RST = VSS

Note 3:

ICC is measured with all output pins disconnected; XTAL1 driven with TCLCH, TCHCL = 5 ns, VIL = VSS – 0.5V; VIH VCC – 0.5V; XTAL2 N.C.; EA = RST = Port 0 = VCC. ICC would be slightly higher if a crystal oscillator used.

Note 4:

Idle ICC is measured with all output pins disconnected; XTAL1 driven TCLCH, TCHCL = 5 ns, VIL = VSS – 0.5V; VIH = VCC – 0.5V; XTAL2 N.C.; Port 0 = VCC; EA = RST = VSS.

EXPLANATION OF THE AC SYMBOLS

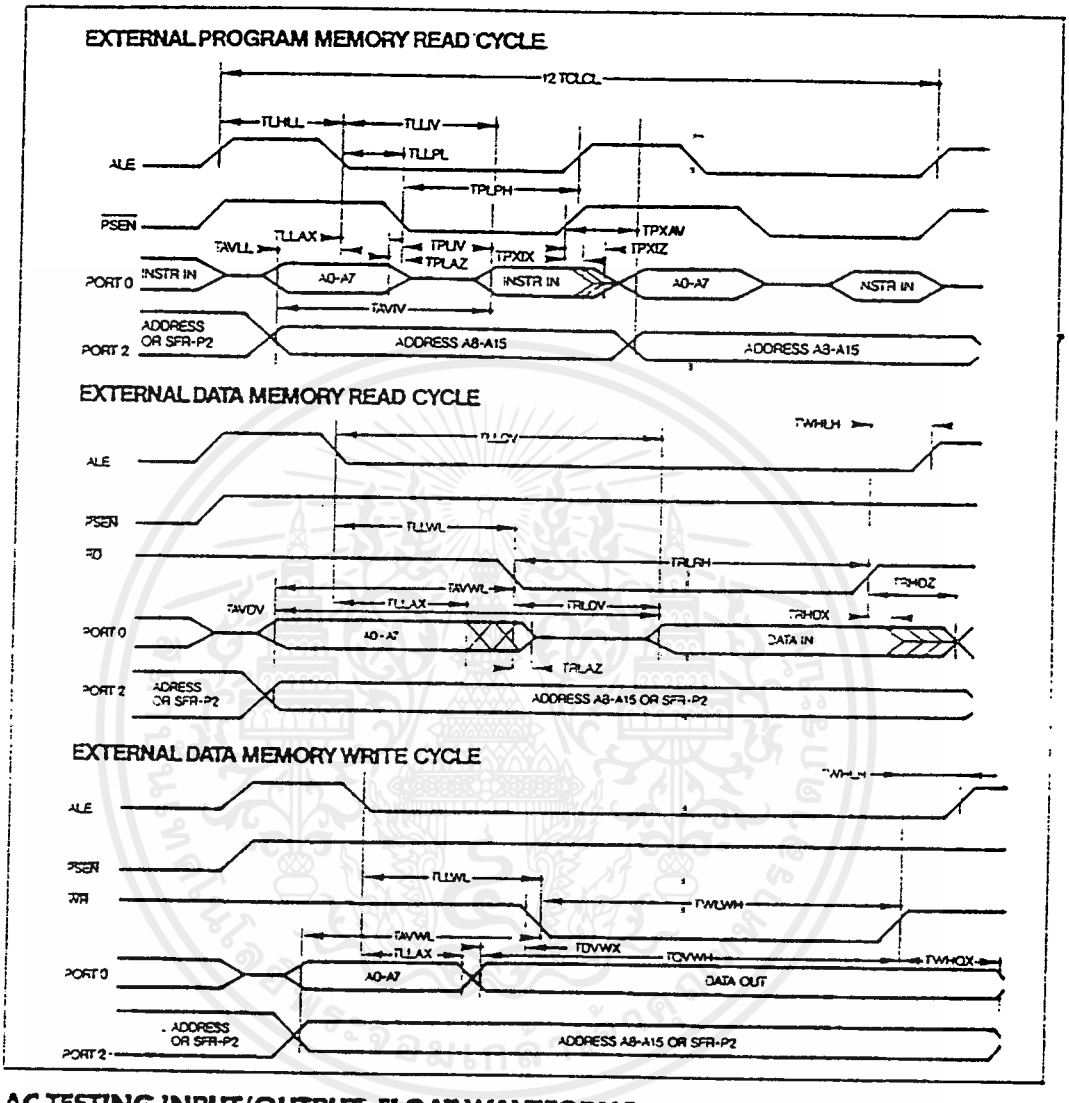
Each timing symbol has 5 characters. The first character is always a 'T' (stands for time). The other characters, depending on their positions, stand for the name of a signal or the logical status of that signal. The following is a list all the characters and what they stand for.

EXAMPLE:

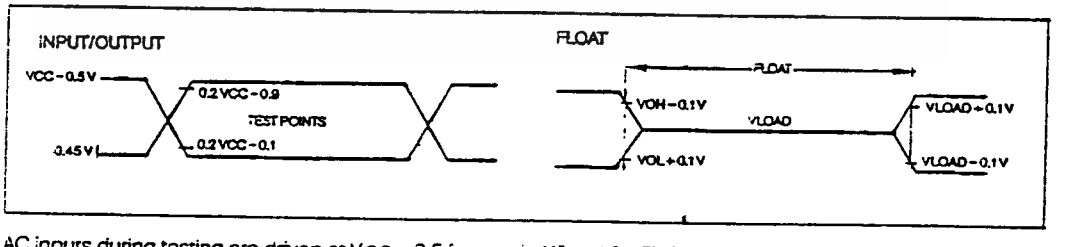
TAVLL = Time for Address Valid to ALE low.
TLLPL = Time for ALE low to PSEN low.

A: Address.
C: Clock.
D: Input data.
H: Logic level HIGH.
I: Instruction (program memory contents).
L: Logic level LOW, or ALE.
P: PSEN
Q: Output data.
R: READ signal.
T: Time.
V: Valid.
W: WRITE signal.
X: No longer a valid logic level.
Z: Float.

AC TIMING DIAGRAMS



AC TESTING INPUT/OUTPUT, FLOAT WAVEFORMS



AC inputs during testing are driven at VCC −0.5 for a logic "1" and 0.45V for a logic "0". Timing measurements are made at VIH min for a logic "1" and VIL max for a logic "0". For timing purposes a port pin is no longer floating when a 100 mV change from load voltage occurs and begins to float when a 100 mV change from the loaded VOH/VOL level occurs. IoI/IoH ≥ = 20 Ma.

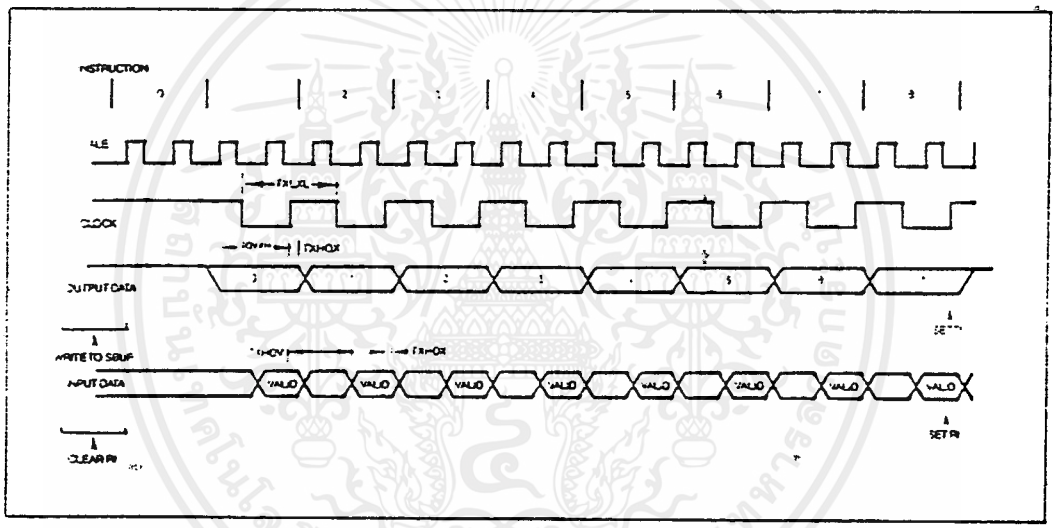
SERIAL PORT TIMING - SHIFT REGISTER MODE

A.C. CHARACTERISTICS:

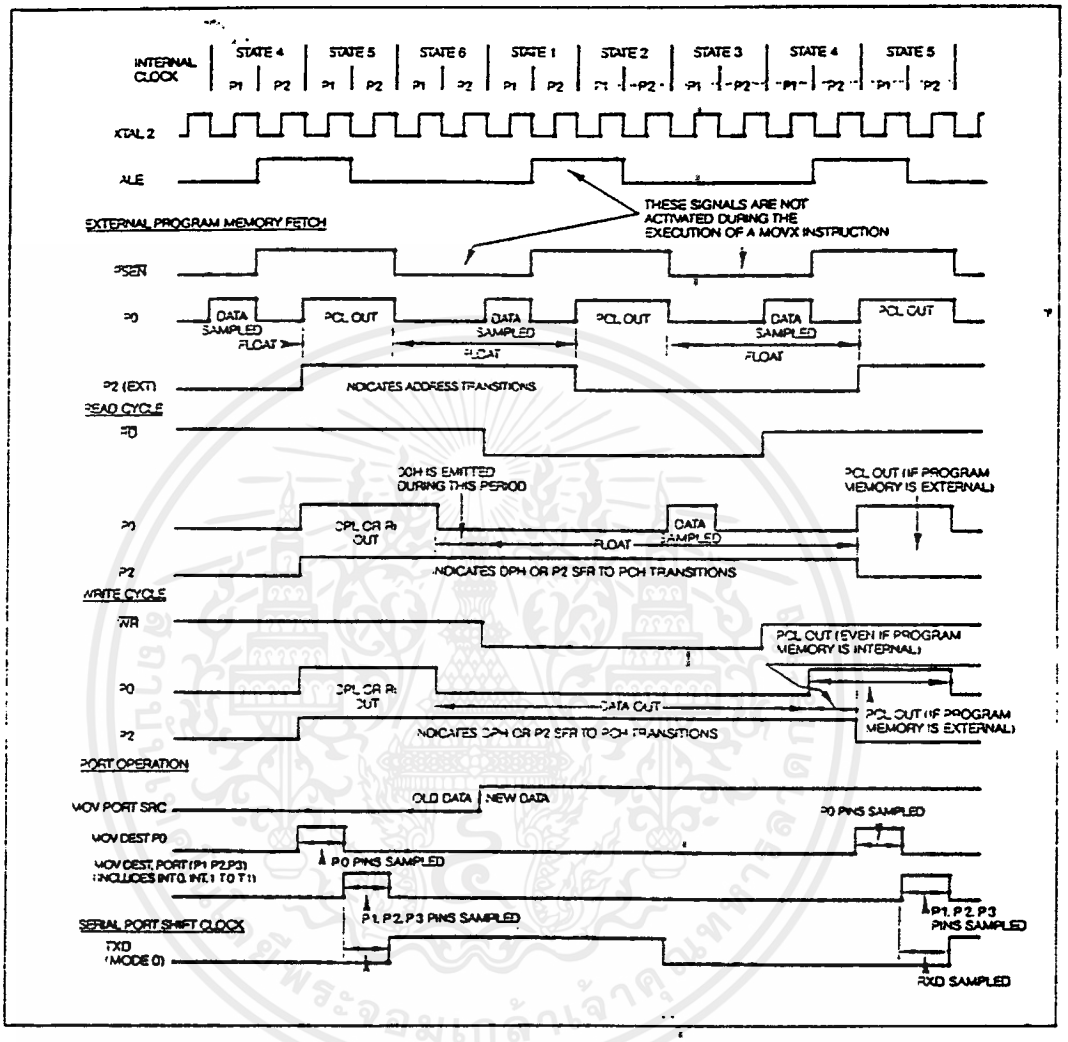
(T_A = 0°C to 70°C; V_{SS} = 0V; V_{CC} = 2.7V to 6V; Load Capacitance = 80 pF)

Symbol	Parameter	Min	Max	Units
TXLXL	Serial Port Clock Cycle Time	12TCLCL		µs
TQVXH	Output Data Setup to Clock Rising Edge	10TCLCL-133		ns
TXHQX	Output Data Hold After Clock Rising Edge	2TCLCL-117		ns
TXHDX	Input Data Hold After Clock Rising Edge	0		ns
TXHDV	Clock Rising Edge to Input Data Valid		10TCLCL-133	ns

SHIFT REGISTER TIMING WAVEFORMS



CLOCK WAVEFORMS



This diagram indicates when signals are clocked internally. The time it takes the signals to propagate to the pins, however, ranges from 25 to 125 ns. This propagation delay is dependent on variables such as temperature and pin loading. Propagation also varies from output to output and component. Typically though ($T_A = 25°C$ fully loaded) RD and WR propagation delays are approximately 50 ns. The other signals are typically 85 ns. Propagation delays are incorporated in the AC specifications.

Table 1. MCS®-51 Instruction Set Description

ARITHMETIC OPERATIONS

Mnemonic		Description	Byte	Cyc
ADD	A,Rn	Add register to Accumulator	1	1
ADD	A,direct	Add direct byte to Accumulator	2	1
ADD	A,@Ri	Add indirect RAM to Accumulator	1	1
ADD	A,#data	Add immediate data to Accumulator	2	1
ADDC	A,Rn	Add register to Accumulator with Carry	1	1
ADDC	A,direct	Add direct byte to A with Carry flag	2	1
ADDC	A,@Ri	Add indirect RAM to A with Carry flag	1	1
ADDC	A,#data	Add immediate data to A with Carry flag	2	1
SUBB	A,Rn	Subtract register from A with Borrow	1	1
SUBB	A,direct	Subtract direct byte from A with Borrow	2	1
SUBB	A,@Ri	Subtract indirect RAM from A with Borrow	1	1
SUBB	A,#data	Subtract immed. data from A with Borrow	2	1
INC	A	Increment Accumulator	1	1
INC	Rn	Increment register	1	1
INC	direct	Increment direct byte	2	1
INC	@Ri	Incriment indirect RAM	1	1
INC	DPTR	Incriment Data Pointer	1	2
DEC	A	Decrement Accumulator	1	1
DEC	Rn	Decrement register	1	1
DEC	direct	Decrement direct byte	2	1
DEC	@Ri	Decrement indirect RAM	1	1
MUL	AB	Multiply A & B	1	4
DIV	AB	Divide A by B	1	4
DA	A	Decimal Adjust Accumulator	1	1

LOGICAL OPERATIONS

Mnemonic		Destination	Byte	Cyc
ANL	A,Rn	AND register to Accumulator	1	1
ANL	A,direct	AND direct byte to Accumulator	2	1
ANL	A,@Ri	AND indirect RAM to Accumulator	1	1
ANL	A,#data	AND immediate data to Accumulator	2	1
ANL	direct,A	AND Accumulator to direct byte	2	1
ANL	direct,#data	AND immediate data to direct byte	3	2
ORL	A,Rn	OR register to Accumulator	1	1
ORL	A,direct	OR direct byte to Accumulator	2	1
ORL	A,@Ri	OR indirect RAM to Accumulator	1	1
ORL	A,#data	OR immediate data to Accumulator	2	1
ORL	direct,A	OR Accumulator to direct byte	2	1
ORL	direct,#data	OR immediate data to direct byte	3	2
XRL	A,Rn	Exclusive-OR register to Accumulator	1	1
XRL	A,direct	Exclusive-OR direct byte to Accumulator	2	1
XRL	A,@Ri	Exclusive-OR indirect RAM to A	1	1
XRL	A,#data	Exclusive-OR immediate data to A	2	1
XRL	direct,A	Exclusive-OR Accumulator to direct byte	2	1
XRL	direct,#data	Exclusive-OR immediate data to direct	3	2
CLR	A	Clear Accumulator	1	1
CPL	A	Complement Accumulator	1	1
RL	A	Rotate Accumulator Left	1	1
RLC	A	Rotate A Left through the Carry flag	1	1
RR	A	Rotate Accumulator Right	1	1
RRC	A	Rotate A Right through Carry flag	1	1
SWAP	A	Swap nibbles within the Accumulator	1	1

Table 1. (Cont.)

DATA TRANSFER

Mnemonic		Description	Byte	Cyc
MOV	A,Rn	Move register to Accumulator	1	1
MOV	A,direct	Move direct byte to Accumulator	2	1
MOV	A,@Ri	Move indirect RAM to Accumulator	1	1
MOV	A,#data	Move immediate data to Accumulator	2	1
MOV	Rn,A	Move Accumulator to register	1	1
MOV	Rn,direct	Move direct byte to register	2	2
MOV	Rn,#data	Move immediate data to register	2	1
MOV	direct,A	Move Accumulator to direct byte	2	1
MOV	direct,Rn	Move register to direct byte	2	2
MOV	direct,direct	Move direct byte to direct	3	2
MOV	direct,@Ri	Move indirect RAM to direct byte	2	2
MOV	direct,#data	Move immediate data to direct byte	3	2
MOV	@Ri,A	Move Accumulator to indirect RAM	1	1
MOV	@Ri,direct	Move direct byte to indirect RAM	2	2
MOV	@Ri,#data	Move immediate data to indirect RAM	2	1
MOV	DPTR,#data 16	Load Data Pointer with a 16-bit constant	3	2
MOVC	A,@A+DPTR	Move Code byte relative to DPTR to A	1	2
MOVC	A,@A+PC	Move Code byte relative to PC to A	1	2
MOVX	A,@Ri	Move External RAM (8-bit addr) to A	1	2
MOVX	A,@DPTR	Move External RAM (16-bit addr) to A	1	2
MOVX	@Ri,A	Move A to External RAM (8-bit addr)	1	2
MOVX	@DPTR,A	Move A to External RAM (16-bit addr)	1	2
PUSH	direct	Push direct byte onto stack	2	2
POP	direct	Pop direct byte form stack	2	2
XCH	A,Rn	Exchange register with Accumulator	1	1
XCH	A,direct	Exchange direct byte with Accumulator	2	1
XCH	A,@Ri	Exchange indirect RAM with A	1	1
XCHD	A,@Ri	Exchange low-order nibble ind RAM with A	1	1

BOOLEAN VARIABLE MANIPULATION

Mnemonic		Description	Byte	Cyc
CLR	C	Clear Carry flag	1	1
CLR	bit	Clear direct bit	2	1
SETB	C	Set Carry flag	1	1
SETB	bit	Set direct Bit	2	1
CPL	C	Complement Carry flag	1	1
CPL	bit	Complement direct bit	2	1
ANL	C,bit	AND direct bit to Carry flag	2	2
ANL	C,/ bit	AND complement of direct bit to Carry	2	2
ORL	C,bit	OR direct bit to Carry flag	2	2
ORL	C,/ bit	OR complement of direct bit to Carry	2	2
MOV	C,bit	Move direct bit to Carry flag	2	1
MOV	bit,C	Move Carry flag to direct bit	2	2

PROGRAM AND MACHINE CONTROL

Mnemonic		Description	Byte	Cyc
ACALL	addr 11	Absolute Subroutine Call	2	2
LCALL	addr 16	Long Subroutine Call	3	2
RET		Return from subroutine	1	2
RETI		Return from interrupt	1	2
AJMP	addr 11	Absolute Jump	2	2
LJMP	addr 16	Long Jump	3	2
SJMP	rel	Short Jump (relative addr)	2	2
JMP	@A+DPTR	Jump indirect relative to the DPTR	1	2
JZ	rel	Jump if Accumulator is Zero	2	2
JNZ	rel	Jump if Accumulator is Not Zero	2	2
JC	rel	Jump if Carry flag is set	2	2
JNC	rel	Jump if No Carry flag	2	2

Table 1. (Cont.)

PROGRAM AND MACHINE CONTROL (cont.)				
Mnemonic		**Description**	**Byte**	**Cyc**
JB	bit,rel	Jump if direct Bit set	3	2
JNB	bit,rel	Jump if direct Bit Not set	3	2
JBC	bit,rel	Jump if direct Bit is set & Clear bit	3	2
CJNE	A,direct,rel	Compare direct to A & Jump if Not Equal	3	2
CJNE	A,#data,rel	Comp. immed. to A & Jump if Not Equal	3	2
CJNE	Rn,#data,rel	Comp. immed. to reg & Jump if Not Equal	3	2
CJNE	@Ri,#data, rel	Comp. immed. to ind. & Jump if Not Equal	3	2
DJNZ	Rn,rel	Decrement register & Jump if Not Zero	2	2
DJNZ	direct, rel	Decrement direct & Jump if Not Zero	3	2
NOP		No operation	1	1

Notes on data addressing modes:

Rn	– Working register R0-R7
direct	– 128 internal RAM locations, any I/O port, control or status register
@Ri	– Indirect internal RAM location addressed by register R0 or R1
#data	– 8-bit constant included in instruction
#data 16	– 16-bit constant included as bytes 2 & 3 of instruction
bit	– 128 software flags, any I/O pin, control or status bit

Notes on program addressing modes:

addr 16	– Destination address for LCALL & LJMP may be anywhere within the 64-k program memory address space
Addr 11	– Destination address for ACALL & AJMP will be within the same 2-k page of program memory as the first byte of the following instruction
rel	– SJMP and all conditional jumps include an 8-bit offset byte. Range is −127−128 bytes relative to first byte of the following instruction

Table 2. Instruction Opcodes in Hexadecimal Order

Hex Code	Number of Bytes	Mnemonic	Operands
00	1	NOP	
01	2	AJMP	code addr
02	3	LJMP	code addr
03	1	RR	A
04	1	INC	A
05	2	INC	data addr
06	1	INC	@R0
07	1	INC	@R1
08	1	INC	R0
09	1	INC	R1
0A	1	INC	R2
0B	1	INC	R3
0C	1	INC	R4
0D	1	INC	R5
0E	1	INC	R6
0F	1	INC	R7
10	3	JBC	bit addr,code addr
11	2	ACALL	code addr
12	3	LCALL	code addr
13	1	RRC	A
14	1	DEC	A
15	2	DEC	data addr
16	1	DEC	@R0
17	1	DEC	@R1
18	1	DEC	R0
19	1	DEC	R1
1A	1	DEC	R2
1B	1	DEC	R3
1C	1	DEC	R4
1D	1	DEC	R5
1E	1	DEC	R6
1F	1	DEC	R7
20	3	JB	bit addr,code addr
21	2	AJMP	code addr
22	1	RET	
23	1	RL	A
24	2	ADD	A,data
25	2	ADD	A,data addr
26	1	ADD	A,@R0
27	1	ADD	A,@R1
28	1	ADD	A,R0
29	1	ADD	A,R1
2A	1	ADD	A,R2
2B	1	ADD	A,R3
2C	1	ADD	A,R4
2D	1	ADD	A,R5
2E	1	ADD	A,R6
2F	1	ADD	A,R7
30	3	JNB	bit addr,code addr
31	2	ACALL	code addr
32	1	RETI	

Hex Code	Number of Bytes	Mnemonic	Operands
33	1	RLC	A
34	2	ADDC	A,#data
35	2	ADDC	A,data addr
36	1	ADDC	A,@R0
37	1	ADDC	A,@R1
38	1	ADDC	A,R0
39	1	ADDC	A,R1
3A	1	ADDC	A,R2
3B	1	ADDC	A,R3
3C	1	ADDC	A,R4
3D	1	ADDC	A,R5
3E	1	ADDC	A,R6
3F	1	ADDC	A,R7
40	2	JC	code addr
41	2	AJMP	code addr
42	2	ORL	data addr,A
43	3	ORL	data addr,#data
44	2	ORL	A,#data
45	2	ORL	A,data addr
46	1	ORL	A,@R0
47	1	ORL	A,@R1
48	1	ORL	A,R0
49	1	ORL	A,R1
4A	1	ORL	A,R2
4B	1	ORL	A,R3
4C	1	ORL	A,R4
4D	1	ORL	A,R5
4E	1	ORL	A,R6
4F	1	ORL	A,R7
50	2	JNC	code addr
51	2	ACALL	code addr
52	2	ANL	data addr,A
53	3	ANL	data addr,#data
54	2	ANL	A,#data
55	2	ANL	A,data addr
56	1	ANL	A,@R0
57	1	ANL	A,@R1
58	1	ANL	A,R0
59	1	ANL	A,R1
5A	1	ANL	A,R2
5B	1	ANL	A,R3
5C	1	ANL	A,R4
5D	1	ANL	A,R5
5E	1	ANL	A,R6
5F	1	ANL	A,R7
60	2	JZ	code addr
61	2	AJMP	code addr
62	2	XRL	data addr,A
63	3	XRL	data addr,#data
64	2	XRL	A,#data
65	2	XRL	A,data addr

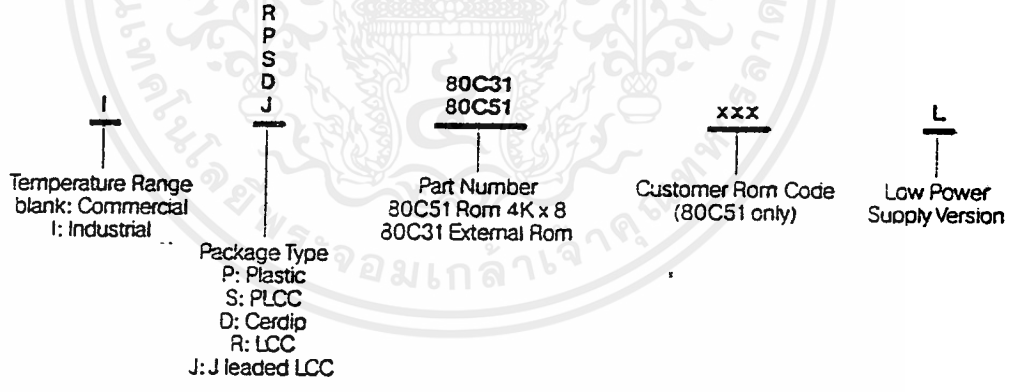
Table 2. (Cont.)

Hex Code	Number of Bytes	Mnemonic	Operands
66	1	XRL	A,@R0
67	1	XRL	A,@R1
68	1	XRL	A,R0
69	1	XRL	A,R1
6A	1	XRL	A,R2
6B	1	XRL	A,R3
6C	1	XRL	A,R4
6D	1	XRL	A,R5
6E	1	XRL	A,R6
6F	1	XRL	A,R7
70	2	JNZ	code addr
71	2	ACALL	code addr
72	2	ORL	C,bit addr
73	1	JMP	@A+DPTR
74	2	MOV	A,#data
75	3	MOV	data addr,#data
76	2	MOV	@R0,#data
77	2	MOV	@R1,#data
78	2	MOV	R0,#data
79	2	MOV	R1,#data
7A	2	MOV	R2,#data
7B	2	MOV	R3,#data
7C	2	MOV	R4,#data
7D	2	MOV	R5,#data
7E	2	MOV	R6,#data
7F	2	MOV	R7,#data
80	2	SJMP	code addr
81	2	AJMP	code addr
82	2	ANL	C,bit addr
83	1	MOVC	A,@A+PC
84	1	DIV	AB
85	3	MOV	data addr,data addr
86	2	MOV	data addr,@R0
87	2	MOV	data addr,@R1
88	2	MOV	data addr,R0
89	2	MOV	data addr,R1
8A	2	MOV	data addr,R2
8B	2	MOV	data addr,R3
8C	2	MOV	data addr,R4
8D	2	MOV	data addr,R5
8E	2	MOV	data addr,R6
8F	2	MOV	data addr,R7
90	3	MOV	DPTR,#data
91	2	ACALL	code addr
92	2	MOV	bit addr,C
93	1	MOVC	A,@A+DPTR
94	2	SUBB	A,#data
95	2	SUBB	A,data addr
96	1	SUBB	A,@R0
97	1	SUBB	A,@R1
98	1	SUBB	A,R0

Hex Code	Number of Bytes	Mnemonic	Operands
99	1	SUBB	A,R1
9A	1	SUBB	A,R2
9B	1	SUBB	A,R3
9C	1	SUBB	A,R4
9D	1	SUBB	A,R5
9E	1	SUBB	A,R6
9F	1	SUBB	A,R7
A0	2	ORL	C,bit addr
A1	2	AJMP	code addr
A2	2	MOV	C,bit addr
A3	1	INC	DPTR
A4	1	MUL	AB
A5		reserved	
A6	2	MOV	@R0,data addr
A7	2	MOV	@R1,data addr
A8	2	MOV	R0,data addr
A9	2	MOV	R1,data addr
AA	2	MOV	R2,data addr
AB	2	MOV	R3,data addr
AC	2	MOV	R4,data addr
AD	2	MOV	R5,data addr
AE	2	MOV	R6,data addr
AF	2	MOV	R7,data addr
B0	2	ANL	C,bit addr
B1	2	ACALL	code addr
B2	2	CPL	bit addr
B3	1	CPL	C
B4	3	CJNE	A,#data,code addr
B5	3	CJNE	A,data addr,code addr
B6	3	CJNE	@R0,#data,code addr
B7	3	CJNE	@R1,#data,code addr
B8	3	CJNE	R0,#data,code addr
B9	3	CJNE	R1,#data,code addr
BA	3	CJNE	R2,#data,code addr
BB	3	CJNE	R3,#data,code addr
BC	3	CJNE	R4,#data,code addr
BD	3	CJNE	R5,#data,code addr
BE	3	CJNE	R6,#data,code addr
BF	3	CJNE	R7,#data,code addr
C0	2	PUSH	data addr
C1	2	AJMP	code addr
C2	2	CLR	bit addr
C3	1	CLR	C
C4	1	SWAP	A
C5	2	XCH	A,data addr
C6	1	XCH	A,@R0
C7	1	XCH	A,@R1
C8	1	XCH	A,R0
C9	1	XCH	A,R1
CA	1	XCH	A,R2
CB	1	XCH	A,R3

Table 2. (Cont.)

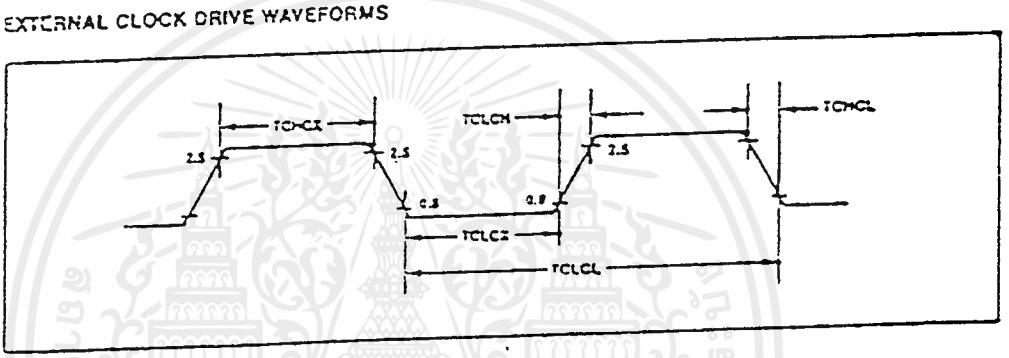
Hex Code	Number of Bytes	Mnemonic	Operands
CC	1	XCH	A,R4
CD	1	XCH	A,R5
CE	1	XCH	A,R6
CF	1	XCH	A,R7
D0	2	POP	data addr
D1	2	ACALL	code addr
D2	2	SETB	bit addr
D3	1	SETB	C
D4	1	DA	A
D5	3	DJNZ	data addr,code addr
D6	1	XCHD	A,@R0
D7	1	XCHD	A,@R1
D8	2	DJNZ	R0,code addr
D9	2	DJNZ	R1,code addr
DA	2	DJNZ	R2,code addr
DB	2	DJNZ	R3,code addr
DC	2	DJNZ	R4,code addr
DD	2	DJNZ	R5,code addr
DE	2	DJNZ	R6,code addr
DF	2	DJNZ	R7,code addr
E0	1	MOVX	A,@DPTR
E1	2	AJMP	code addr
E2	1	MOVX	A,@R0
E3	1	MOVX	A,@R1
E4	1	CLR	A
E5	2	MOV	A,data addr
E6	1	MOV	A,@R0
E7	1	MOV	A,@R1
E8	1	MOV	A,R0
E9	1	MOV	A,R1
EA	1	MOV	A,R2
EB	1	MOV	A,R3
EC	1	MOV	A,R4
ED	1	MOV	A,R5
EE	1	MOV	A,R6
EF	1	MOV	A,R7
F0	1	MOVX	@DPTR,A
F1	2	ACALL	code addr
F2	1	MOVX	@R0,A
F3	1	MOVX	@R1,A
F4	1	CPL	A
F5	2	MOV	data addr,A
F6	1	MOV	@R0,A
F7	1	MOV	@R1,A
F8	1	MOV	R0,A
F9	1	MOV	R1,A
FA	1	MOV	R2,A
FB	1	MOV	R3,A
FC	1	MOV	R4,A
FD	1	MOV	R5,A
FE	1	MOV	R6,A
FF	1	MOV	R7,A



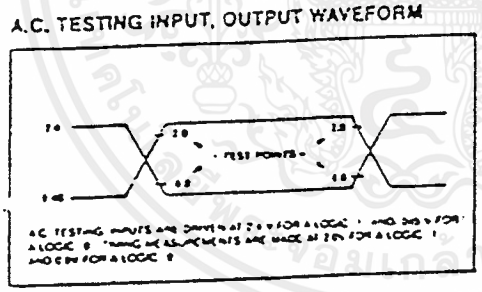
EXTERNAL CLOCK DRIVE

Symbol	Parameter	Min	Max	Units
1/TCLCL	Oscillator Frequency	3.5	12	MHz
TCHCX	High Time	20		ns
TCLCX	Low Time	20		ns
TCLCH	Rise Time		20	ns
TCHCL	Fall Time		20	ns

EXTERNAL CLOCK DRIVE WAVEFORMS



A.C. TESTING INPUT, OUTPUT WAVEFORM



intel

2764A
ADVANCED 64K (8Kx8) UV ERASABLE PROM

- Fast 180 nsec Access Time —HMOS II*-E Technology
- Low Power
  —60 mA Maximum Active
  —20 mA Maximum Standby
- Two Line Control
- Intelligent Programming™ Algorithm —Fastest EPROM Programming
- Intelligent Identifier™ Mode —Automated Programming Operations
- Compatible with 2764, 27128, 27256
- ±10% V_{CC} Tolerance Available

The Intel 2764A is a 5V only, 65,536-bit ultraviolet erasable and electrically-programmable read-only memory (EPROM). The 2764A is an advanced version of the 2764 and is fabricated with Intel's HMOSII-E technology which significantly reduces die size and greatly improves the device's performance, power consumption, reliability and producibility.

The 2764A provides access times to 180 ns(2764A-1). This is an improvement over the fastest 2764 access time of 200 ns. This is compatible with high-performance microprocessors, such as Intel's 8 MHz iAPX 188 allowing full speed operation without the addition of WAIT states. The 2764A is also directly compatible with the 12 MHz 8051 family.

Several advanced features have been designed into the 2764A that allow fast and reliable programming—the Intelligent Programming Algorithm and the Intelligent Identifier Mode. Programming equipment that takes advantage of these innovations will electronically identify the 2764A and then rapidly program it using an efficient programming method.

The 2764A also offers reduced power consumption compared to the 2764. The maximum active current on faster speed parts is 60 mA while the maximum standby current is only 20 mA. The standby mode lowers power consumption without increasing access time.

Two-line control and JEDEC-approved, 28 pin packaging are standard features of all Intel higher density EPROMs. This ensures easy microprocessor interfacing and minimum design efforts when upgrading, adding or choosing between non-volatile memory alternatives.

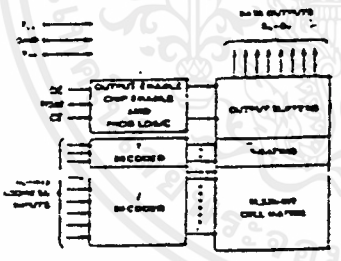


Figure 1. Block Diagram

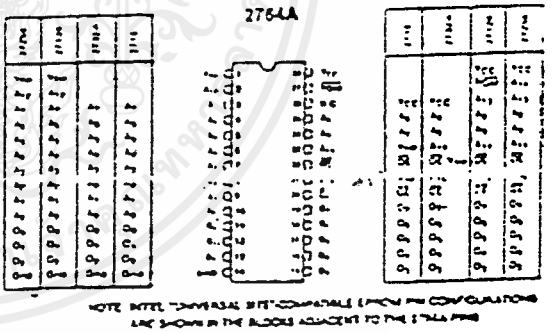


Figure 2. Pin Configurations

MODE SELECTION

MODE	$\overline{CE}$	$\overline{OE}$	$\overline{PGM}$	A9	V_{PP}	V_{CC}	Outputs
Read	V_{IL}	V_{IL}	V_{IH}	X	V_{CC}	V_{CC}	D_{OUT}
Output Disable	V_{IL}	V_{IH}	V_{IH}	X	V_{CC}	V_{CC}	High Z
Standby	V_{IH}	X	X	X	V_{CC}	V_{CC}	High Z
Program	V_{IL}	V_{IH}	V_{IL}	X	V_{PP}	V_{CC}	D_{IN}
Program Verify	V_{IL}	V_{IL}	V_{IH}	X	V_{PP}	V_{CC}	D_{OUT}
Program Inhibit	V_{IH}	X	X	X	V_{PP}	V_{CC}	High Z
Intelligent Identifier (Manufacturer)	V_{IL}	V_{IL}	V_{IH}	V_H	V_{CC}	V_{CC}	89H
Intelligent Identifier (Device)	V_{IL}	V_{IL}	V_{IH}	V_H	V_{CC}	V_{CC}	0CH

PIN NAMES

A_0–A_{12}	ADDRESSES
$\overline{CE}$	CHIP ENABLE
$\overline{OE}$	OUTPUT ENABLE
O_0–O_7	OUTPUTS
$\overline{PGM}$	PROGRAM
N.C.	NO CONNECT

*HMOS is a patented process of Intel Corporation

ABSOLUTE MAXIMUM RATINGS*

Temperature Under Bias ..... −10°C to +80°C
Storage Temperature ..... −65°C to +125°C
All Input or Output Voltages with Respect to Ground ..... +6.5V to −0.6V
Voltage on Pin 24 with Respect to Ground ..... +13.5V to −0.6V
V_{PP} Supply Voltage with Respect to Ground During Programming ..... +14V to −0.6V

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

D.C. AND A.C. OPERATING CONDITIONS DURING READ

	2764A-1, 2764A-2, 2764A 2764A-3, 2764A-4	2764A-20, 2764A-25 2764A-30, 2764A-45
Operating Temperature Range	0°–70°C	0°–70°C
V_{CC} Power Supply[1,2]	5V ± 5%	5V ± 10%
V_{PP} Voltage[2]	$V_{PP} = V_{CC}$	$V_{PP} = V_{CC}$

READ OPERATION

D.C. CHARACTERISTICS

Symbol	Parameter	Limits Min	Typ[3]	Max	Unit	Conditions
I_{LI}	Input Load Current			10	µA	$V_{IN} = 5.5V$
I_{LO}	Output Leakage Current			10	µA	$V_{OUT} = 5.5V$
I_{PP1}[1]	V_{PP} Current Read			5	mA	$V_{PP} = 5.5V$
I_{CC1}[1]	V_{CC} Current Standby			20/25[5]	mA	$\overline{CE} = V_{IH}$
I_{CC2}[1]	V_{CC} Current Active			50/75[5]	mA	$\overline{CE} = \overline{OE} = V_{IL}$
V_{IL}	Input Low Voltage	−.1		+.8	V	
V_{IH}	Input High Voltage	2.0		$V_{CC}+1$	V	
V_{OL}	Output Low Voltage			.45	V	$I_{OL} = 2.1$ mA
V_{OH}	Output High Voltage	2.4			V	$I_{OH} = -400$ µA
V_{PP}[1]	V_{PP} Read Voltage	3.8		V_{CC}	V	$V_{CC} = 5.0V \pm 0.25V$

A.C. CHARACTERISTICS

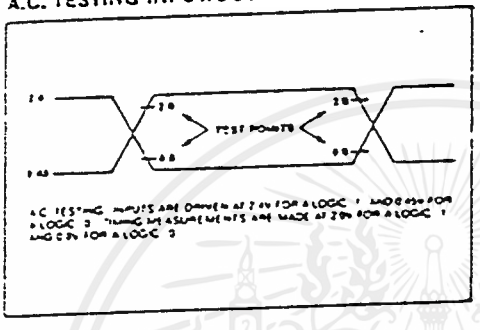
Symbol	Parameter	2764A-1 Limits Min	Max	2764A-20 & 2764A-2 Limits Min	Max	2764A-25 & 2764A Limits Min	Max	2764A-30 & 2764A-3 Limits Min	Max	2764A-45 & 2764A-4 Limits Min	Max	Unit	Test Conditions
t_{ACC}	Address to Output Delay		180		200		250		300		450	ns	$\overline{CE} = \overline{OE} = V_{IL}$
t_{CE}	$\overline{CE}$ to Output Delay		180		200		250		300		450	ns	$\overline{OE} = V_{IL}$
t_{OE}	$\overline{OE}$ to Output Delay		85		75		100		120		150	ns	$\overline{CE} = V_{IL}$
t_{DF}[4]	$\overline{OE}$ or $\overline{CE}$ High to Output Data Float	0	55	0	55	0	60	0	105	0	130	ns	$\overline{CE} = V_{IL}$
t_{OH}	Output Hold from Addresses $\overline{CE}$ or $\overline{OE}$ Whichever Occurred First	0		0		0		0		0		ns	$\overline{CE} = \overline{OE} = V_{IL}$

NOTES: 1. V_{CC} must be applied simultaneously or before V_{PP} and removed simultaneously or after V_{PP}.
2. V_{PP} may be connected directly to V_{CC} except during programming. The supply current would then be the sum of I_{CC} and I_{PP}.
3. Typical values are for $T_A = 25°C$ and nominal supply voltages.
4. This parameter is only sampled and is not 100% tested. Output Float is defined as the point where data is no longer driven — see timing diagram on the following page.
5. Max I_{CC} rating differs with access time. Rating of 60 mA active and 20 mA standby are for 2764As at 200 nsec and 180 nsec access time only.

CAPACITANCE[2] ($T_A = 25°C$, $f = 1MHz$)

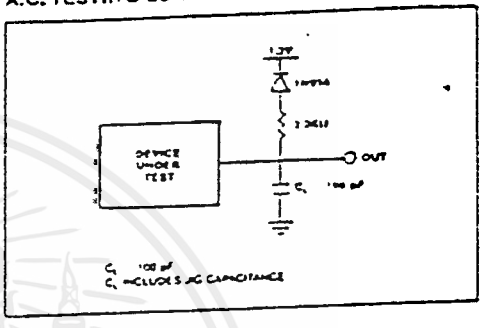
Symbol	Parameter	Typ.[1]	Max.	Unit	Conditions
C_{in}	Input Capacitance	4	6	pF	$V_{in} = 0V$
C_{out}	Output Capacitance	8	12	pF	$V_{out} = 0V$

A.C. TESTING INPUT/OUTPUT WAVEFORM



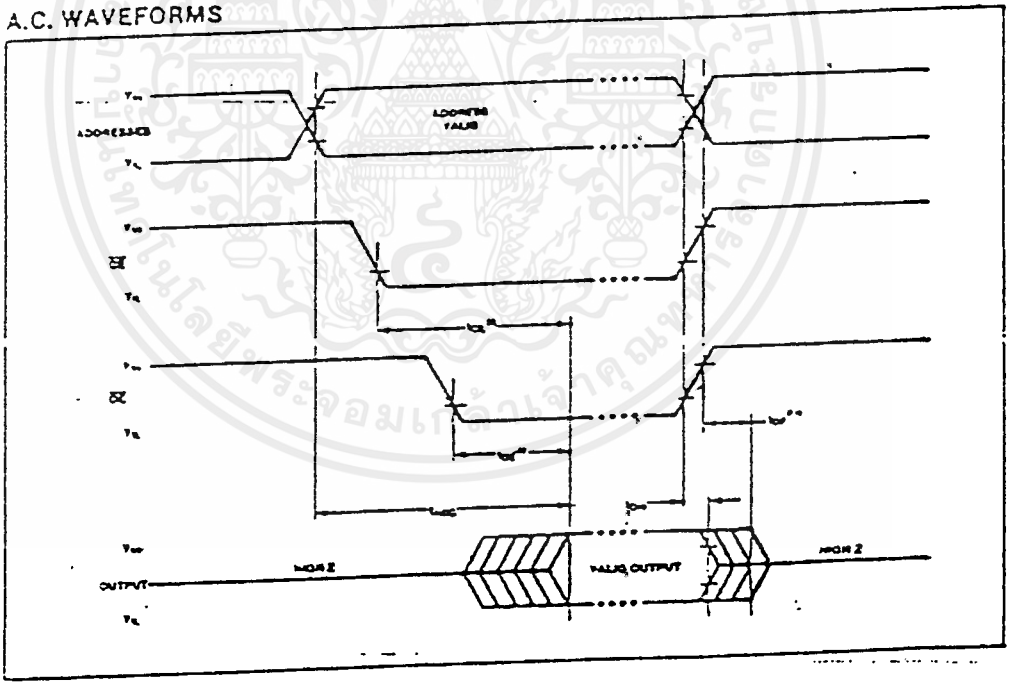
A.C. TESTING INPUTS ARE DRIVEN AT 2.4V FOR A LOGIC "1" AND 0.45V FOR A LOGIC "0". TIMING MEASUREMENTS ARE MADE AT 2.0V FOR A LOGIC "1" AND 0.8V FOR A LOGIC "0".

A.C. TESTING LOAD CIRCUIT



C_L = 100 pF
C_L INCLUDES JIG CAPACITANCE

A.C. WAVEFORMS



NOTES: 1. Typical values are for T_A = 25°C and nominal supply voltages.
2. This parameter is only sampled and is not 100% testing.
3. $\overline{OE}$ may be delayed up to $t_{CE} - t_{OE}$ after the falling edge of $\overline{CE}$ without impact on t_{CE}.
4. t_{DF} is specified from $\overline{OE}$ or $\overline{CE}$, whichever occurs first.

DEVICE OPERATION

The seven modes of operation of the 2764A are listed in Table 1. A single 5V power supply is required in the read mode. All inputs are TTL levels except for V_{PP} and 12V on A9 for Intelligent Identifier mode.

Table 1. MODE SELECTION

MODE \ PINS	$\overline{CE}$ (20)	$\overline{OE}$ (22)	$\overline{PGM}$ (27)	A_9 (24)	V_{PP} (1)	V_{CC} (28)	Outputs (11–13, 15–19)
Read	V_{IL}	V_{IL}	V_{IH}	X	V_{CC}	V_{CC}	D_{OUT}
Output Disable	V_{IL}	V_{IH}	V_{IH}	X	V_{CC}	V_{CC}	High Z
Standby	V_{IH}	X	X	X	V_{CC}	V_{CC}	High Z
Verify	V_{IL}	V_{IL}	V_{IH}	X	V_{PP}	V_{CC}	D_{OUT}
Program Inhibit	V_{IH}	X	X	X	V_{PP}	V_{CC}	High Z
Intelligent Identifier	V_{IL}	V_{IL}	V_{IH}	V_H	V_{CC}	V_{CC}	Code
Intelligent Programming	V_{IL}	V_{IH}	V_{IL}	X	V_{PP}	V_{CC}	D_{IN}

NOTES:
1. X can be V_{IL} or V_{IH}.
2. $V_H = 12.0V \pm 0.5V$.

READ MODE

The 2764A has two control functions, both of which must be logically active in order to obtain data at the outputs. Chip Enable ($\overline{CE}$) is the power control and should be used for device selection. Output Enable ($\overline{OE}$) is the output control and should be used to gate data from the output pins, independent of device selection. Assuming that addresses are stable, the address access time (t_{ACC}) is equal to the delay from $\overline{CE}$ to output (t_{CE}). Data is available at the outputs after a delay of t_{OE} from the falling edge of $\overline{OE}$, assuming that $\overline{CE}$ has been low and addresses have been stable for at least $t_{ACC} - t_{OE}$.

STANDBY MODE

The 2764A has standby mode which reduces the maximum current from 75 mA to 35 mA. The 2764A is placed in the standby mode by applying a TTL-high signal to the $\overline{CE}$ input. When in standby mode, the outputs are in a high impedance state, independent of the $\overline{OE}$ input.

Output OR-Tieing

Because EPROMs are usually used in larger memory arrays, Intel has provided 2 control lines which accommodate this multiple memory connection. The two control lines allow for:

a) the lowest possible memory power dissipation, and
b) complete assurance that output bus contention will not occur.

To use these two control lines most efficiently, $\overline{CE}$ (pin 20) should be decoded and used as the primary device selecting function, while $\overline{OE}$ (pin 22) should be made a common connection to all devices in the array and connected to the $\overline{READ}$ line from the system control bus. This assures that all deselected memory devices are in their low power standby mode and that the output pins are active only when data is desired from a particular memory device.

System Considerations

The power switching characteristics of HMOS II-E EPROMs require careful decoupling of the devices. The supply current, I_{CC}, has three segments that are of interest to the system designer—the standby current level, the active current level, and the transient current peaks that are produced by the falling and rising edges of Chip Enable. The magnitude of these transient current peaks is dependent on the output capacitive loading of the device. The associated transient voltage peaks can be suppressed by complying with Intel's Two-Line Control, as detailed in Intel's Application Note AP-72, Order Number 8556, and by properly selected decoupling capacitors. It is recommended that a 0.1 μF ceramic capacitor be used on every device between V_{CC} and GND. This should be a high frequency capacitor of low inherent inductance and should be placed as close to the device as possible. In addition, a 4.7 μF bulk electrolytic capacitor should be used between V_{CC} and GND for every eight devices. The bulk capacitor should be located near where the power supply is connected to the array. The purpose of the bulk capacitor is to overcome the voltage droop caused by the inductive effect of PC board traces.

PROGRAMMING MODES

Caution: Exceeding 14V on pin 1 (V_{PP}) will permanently damage the 2764A.

Initially, and after each erasure, all bits of the 2764A are in the "1" state. Data is introduced by selectively programming "0s" into the desired bit locations. Although only "0s" will be programmed, both "1s" and "0s" can be present in the data word. The only way to change a "0" to a "1" is by ultraviolet light erasure.

The 2764A is in the programming mode when V_{PP} input is at 12.5V and $\overline{CE}$ and $\overline{PGM}$ are both at TTL low. The data to be programmed is applied 8 bits in parallel to the data output pins. The levels required for the address and data inputs are TTL.

inteligent Programming™ Algorithm

The 2764A inteligent Programming Algorithm rapidly programs Intel 2764A EPROMs using an efficient and reliable method particularly suited to the production programming environment. Typical programming time for individual devices is on the order of one and a half minutes. Programming reliability is also ensured as the incremental program margin of each byte is continually monitored to determine when it has been successfully programmed. A flowchart of the 2764A inteligent Programming Algorithm is shown in Figure 3.

The inteligent Programming Algorithm utilities two different pulse types: initial and overprogram. The duration of the initial $\overline{PGM}$ pulse(s) is one millisecond, which will then be followed by a longer overprogram pulse of length 3X msec. X is an iteration counter and is equal to the number of the initial one millisecond pulses applied to a particular 2764A location, before a correct verify occurs. Up to 25 one-millisecond pulses per byte are provided for before the overprogram pulse is applied.

The entire sequence of program pulses and byte verifications is performed at V_{CC} = 6.0V and V_{PP} = 12.5V. When the inteligent Programming cycle has been completed, all bytes should be compared to the original data with V_{CC} = V_{PP} = 5.0V.

Program Inhibit

Programming of multiple 2764As in parallel with different data is easily accomplished by using the Program Inhibit mode. A high-level $\overline{CE}$ or $\overline{PGM}$ input inhibits the other 2764As from being programmed.

Except for $\overline{CE}$, all like inputs (including $\overline{OE}$) of the parallel 2764As may be common. A TTL low-level pulse applied to the $\overline{CE}$ input with V_{PP} at 12.5V will program the selected 2764A.

Verify

A verify should be performed on the programmed bits to determine that they have been correctly programmed. The verify is performed with $\overline{OE}$ at V_{IL}, $\overline{CE}$ at V_{IL}, $\overline{PGM}$ at V_{IH} and V_{PP} at 12.5V.

inteligent Identifier™ Mode

The inteligent Identifier Mode allows the reading out of a binary code from an EPROM that will identify its manufacturer and type. This mode is intended for use by programming equipment for the purpose of automatically matching the device to be programmed with its corresponding programming algorithm. This mode is functional in the 25°C ±5°C ambient temperature range that is required when programming the 2764A.

To activate this mode, the programming equipment must force 11.5V to 12.5V on address line A9 (pin 24) of the 2764A. Two identifier bytes may then be sequenced from the device outputs by toggling address line A0 (pin 10) from V_{IL} to V_{IH}. All other address lines must be held at V_{IL} during inteligent Identifier Mode.

Byte 0 (A0 = V_{IL}) represents the manufacturer code and byte 1 (A0 = V_{IH}) the device identifier code. For the Intel 2764A, these two identifier bytes are given in Table 2. All identifiers for manufacturer and device codes will possess odd parity, with the MSB (O_7) defined as the parity bit.

Table 2. 2764A inteligent Identifier™ Bytes

Identifier \ Pins	A_0 (10)	O_7 (19)	O_6 (18)	O_5 (17)	O_4 (16)	O_3 (15)	O_2 (13)	O_1 (12)	O_0 (11)	Hex Data
Manufacturer Code	V_{IL}	1	0	0	0	1	0	0	1	89
Device Code	V_{IH}	0	0	0	0	1	0	0	0	08

NOTES:
1. A_9 = 12.0V ±0.5V
2. A_1–A_8, A_{10}–A_{12}, $\overline{CE}$, $\overline{OE}$ = V_{IL}
3. A_{14} = V_{IH} or V_{IL}

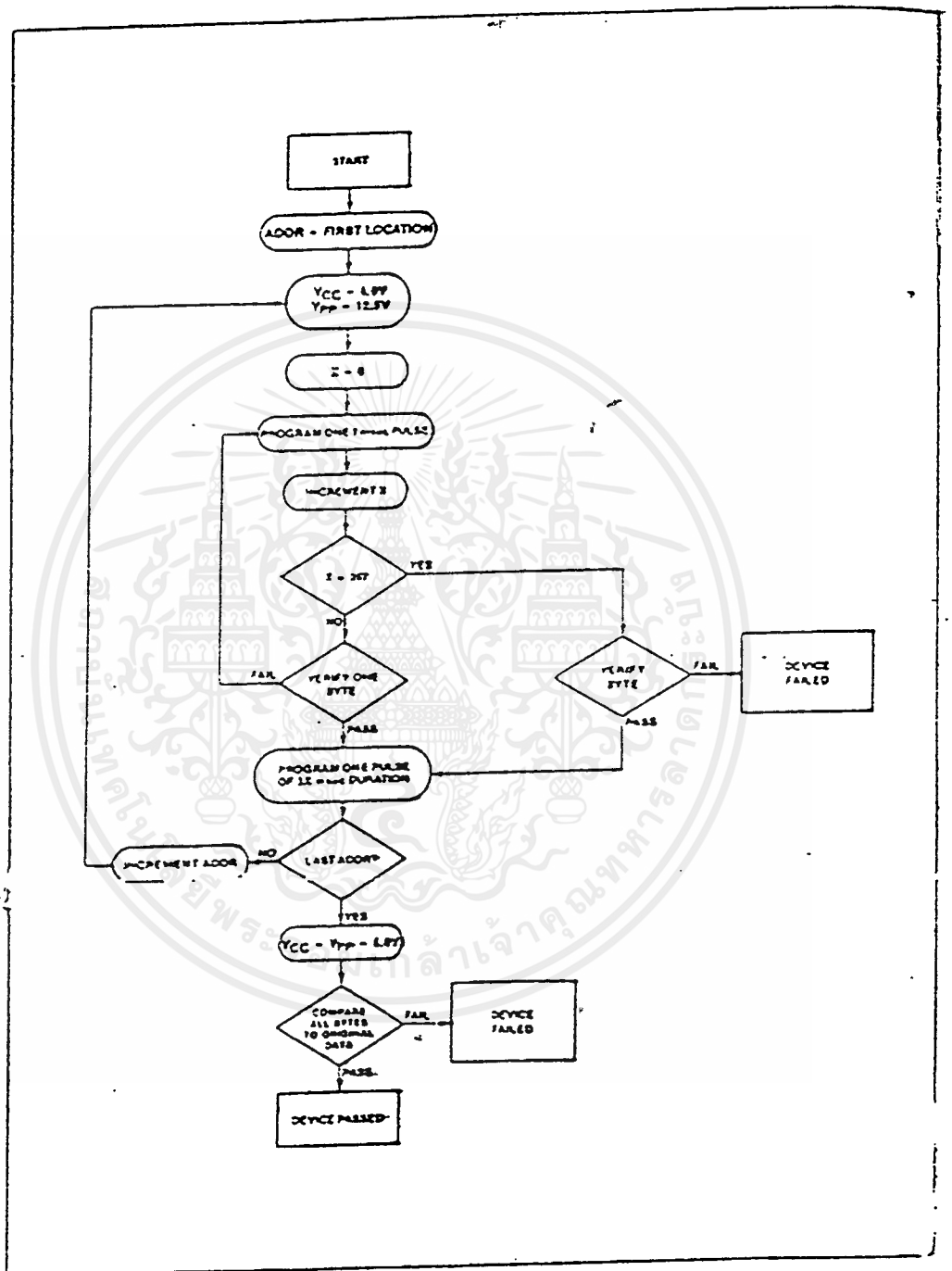


Figure 3. 2764A Intelligent Programming™ Flowchart

ERASURE CHARACTERISTICS

The erasure characteristics of the 2764A are such that erasure begins to occur upon exposure to light with wavelengths shorter than approximately 4000 Angstroms (Å). It should be noted that sunlight and certain types of fluorescent lamps have wavelengths in the 3000—4000 Å range. Data show that constant exposure to room level fluorescent lighting could erase that typical 2764A in approximately 3 years, while it would take approximately 1 week to cause erasure when exposed to direct sunlight. If the 2764A is to be exposed to these types of lighting conditions for extended periods of time, opaque labels should be placed over the 2764A window to prevent unintentional erasure.

The recommended erasure procedure for the 2764A is exposure to shortwave ultraviolet light which has a wavelength of 2537 Angstroms (Å). The integrated dose (i.e., UV intensity x exposure time) for erasure should be a minimum of 15 Wsec/cm^2. The erasure time with this dosage is approximately 15 to 20 minutes using an ultraviolet lamp with a 12000 μW/cm^2 power rating. The 2764A should be placed within 1 inch of the lamp tubes during erasure. The maximum integrated dose a 2764A can be exposed to without damage is 7258 Wsec/cm^2 (1 week @ 12000 μW/cm^2). Exposure of the 2764A to high intensity UV light for long periods may cause permanent damage.

RELEVANT INTEL LITERATURE

AR-265 Versatile Algorithm. Equipment Cut Programming Time
AR-358 EPROM Reliability Data Summary

int$_e$ligent Programming™ Algorithm

D.C. PROGRAMMING CHARACTERISTICS:
T_A = 25 ± 5°C, V_{CC} = 6.0V ± 0.25V, V_{PP} = 12.5V ± 0.5V

Symbol	Parameter	Limits			Test Conditions (see Note 1)
		Min.	Max.	Unit	
I_{LI}	Input Current (All Inputs)		10	µA	$V_{IN} = V_{IL}$ or V_{IH}
V_{IL}	Input Low Level (All Inputs)	-0.1	0.8	V	
V_{IH}	Input High Level	2.0	V_{CC}	V	
V_{OL}	Output Low Voltage During Verify		0.45	V	I_{OL} = 2.1 mA
V_{OH}	Output High Voltage During Verify	2.4		V	I_{OH} = -400 µA
I_{CC2}	V_{CC} Supply Current (Program & Verify)		75	mA	
I_{PP2}	V_{PP} Supply Current (Program)		50	mA	$\overline{CE} = V_{IL}$
V_{ID}	A$_9$ int$_e$ligent Identifier Voltage	11.5	12.5	V	

NOTES:
1. V_{CC} must be applied simultaneously or before V_{PP} and removed simultaneously or after V_{PP}.

A.C. PROGRAMMING CHARACTERISTICS:
T_A = 25 ± 5°C, V_{CC} = 6.0V ± 0.25V, V_{PP} = 12.5V ± 0.5V

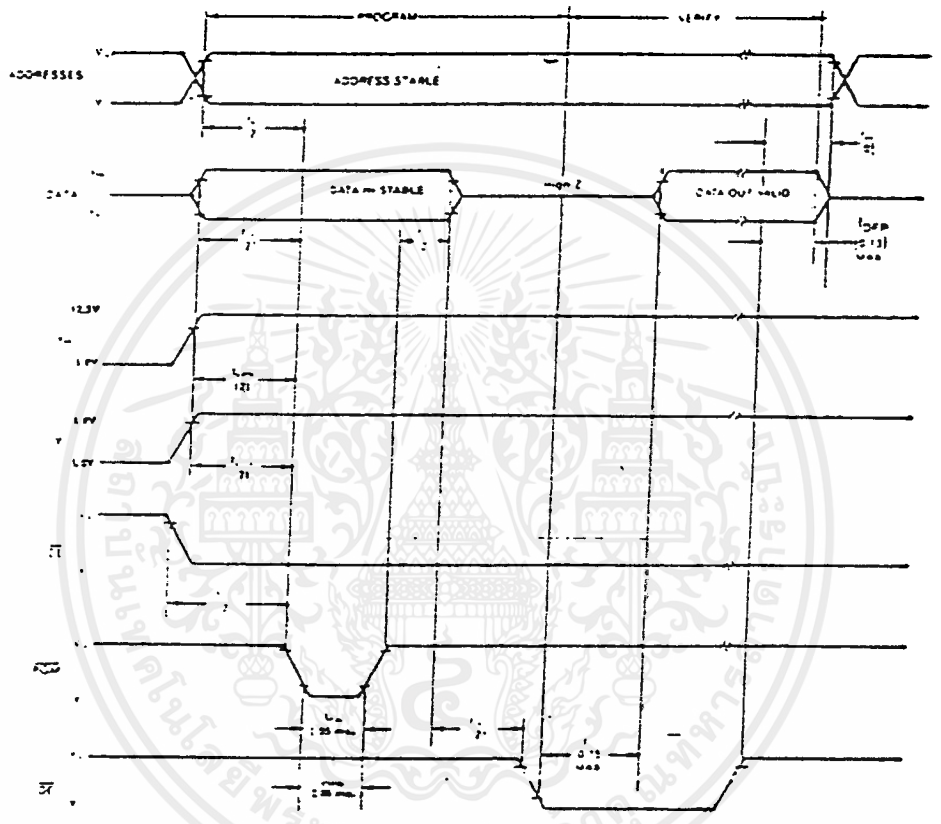
Symbol	Parameter	Limits				Test Conditions* (see Note 1)
		Min.	Typ.	Max.	Unit	
t_{AS}	Address Setup Time	2			µs	
t_{OES}	$\overline{OE}$ Setup Time	2			µs	
t_{DS}	Data Setup Time	2			µs	
t_{AH}	Address Hold Time	0			µs	
t_{DH}	Data Hold Time	2			µs	
t_{DFP}[4]	$\overline{OE}$ High to Output Float Delay	0		130	ns	
t_{VPS}	V_{PP} Setup Time	2			µs	
t_{VCS}	V_{CC} Setup Time	2			µs	
t_{PW}	$\overline{PGM}$ Initial Program Pulse Width	0.95	1.0	1.05	ms	(see Note 3)
t_{OPW}	$\overline{PGM}$ Overprogram Pulse Width	2.85		78.75	ms	(see Note 2)
t_{OE}	Data Valid from $\overline{OE}$			150	ns	

*A.C. CONDITIONS OF TEST

Input Rise and Fall Times (10% to 90%) ... 20 ns
Input Pulse Levels ..... 0.45V to 2.4V
Input Timing Reference Level ..... 0.8V and 2.0V
Output Timing Reference Level ... 0.8V and 2.0V

NOTES:
1. V_{CC} must be applied simultaneously or before V_{PP} and removed simultaneously or after V_{PP}.
2. The length of the overprogram pulse may vary from 2.85 msec to 78.75 msec as a function of the iteration counter value X.
3. Initial Program Pulse width tolerance is 1 msec ±5%.
4. This parameter is only sampled and is not 100% tested. Output Float is defined as the point where data is no longer driven—see timing diagram.

Intelligent Programming™ WAVEFORMS



1. ALL TIMES SHOWN IN ( ) ARE MINIMUM AND IN μSEC UNLESS OTHERWISE SPECIFIED.
2. THE INPUT TIMING REFERENCE LEVEL IS .8V FOR V_{IL} AND 2V FOR A V_{IH}.
3. t_{OE} AND t_{DFP} ARE CHARACTERISTICS OF THE DEVICE BUT MUST BE ACCOMMODATED BY THE PROGRAMMER.
4. WHEN PROGRAMMING THE 2764A, A 0.1 μF CAPACITOR IS REQUIRED ACROSS V_{PP} AND GROUND TO SUPPRESS SPURIOUS VOLTAGE TRANSIENTS WHICH CAN DAMAGE THE DEVICE.

National Semiconductor

microCMOS

MM54HC373/MM74HC373 TRI-STATE® Octal D-Type Latch

General Description

These high speed OCTAL D-TYPE LATCHES utilize microCMOS Technology, 3.5 micron silicon gate P-well CMOS. They possess the high noise immunity and low power consumption of standard CMOS integrated circuits, as well as the ability to drive 15 LS-TTL loads. Due to the large output drive capability and the TRI-STATE feature, these devices are ideally suited for interfacing with bus lines in a bus organized system.

When the LATCH ENABLE input is high, the Q outputs will follow the D inputs. When the LATCH ENABLE goes low, data at the D inputs will be retained at the outputs until LATCH ENABLE returns high again. When a high logic level is applied to the OUTPUT CONTROL input, all outputs go to a high impedance state, regardless of what signals are present at the other inputs and the state of the storage elements.

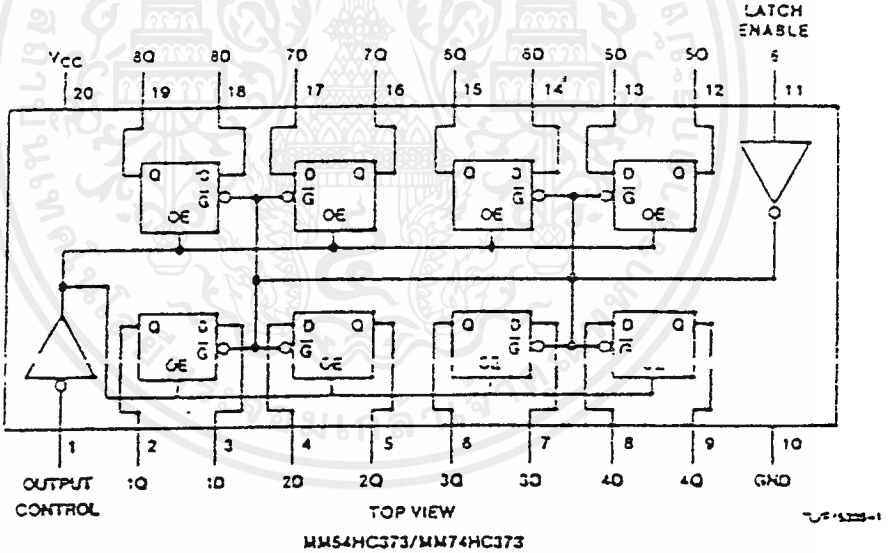
The 54HC/74HC logic family is speed, function, and pinout compatible with the standard 54LS/74LS logic family. All inputs are protected from damage due to static discharge by internal diode clamps to V_{CC} and ground.

Features

- Typical propagation delay: 18 ns
- Wide operating voltage range: 2 to 6 volts
- Low input current: 1 μA maximum
- Low quiescent current: 80 μA maximum (74 series)
- Output drive capability: 15 LS-TTL loads

Connection Diagram

Dual-In-Line Package



TOP VIEW

MM54HC373/MM74HC373

54HC373 (J) 74HC373 (J,N)

Truth Table

Output Control	Latch Enable	Data	373 Output
L	H	H	H
L	H	L	L
L	L	X	Q_0
H	X	X	Z

H = high level, L = low level

Q_0 = level of output before steady-state input conditions were established

Z = high impedance

Absolute Maximum Ratings (Notes 1 & 2)

Supply Voltage (V_{CC})	−0.5 to +7.0V
DC Input Voltage (V_{IN})	−1.5 to V_{CC} +1.5V
DC Output Voltage (V_{OUT})	−0.5 to V_{CC} +0.5V
Clamp Diode Current (I_{IK}, I_{OK})	±20 mA
DC Output Current, per pin (I_{OUT})	±35 mA
DC V_{CC} or GND Current, per pin (I_{CC})	±70 mA
Storage Temperature Range (T_{STG})	−65°C to +150°C
Power Dissipation (P_D)	500 mW
Lead Temperature (T_L) (Soldering 10 seconds)	260°C

Operating Conditions

	Min	Max	Units
Supply Voltage (V_{CC})	2	6	V
DC Input or Output Voltage (V_{IN}, V_{OUT})	0	V_{CC}	V
Operating Temperature Range (T_A)			
MM74HC	−40	+85	°C
MM54HC	−55	+125	°C
Input Rise or Fall Times (t_r, t_f) V_{CC} = 2.0V		1000	ns
V_{CC} = 4.5V		500	ns
V_{CC} = 6.0V		400	ns

DC Electrical Characteristics

Parameter	Conditions	V_{CC}	T_A = 25°C	74HC T_A = −40 to 85°C	54HC T_A = −55 to 125°C	Units
			Typ	Guaranteed Limits		
Minimum High Level Input Voltage		2.0V		1.5	1.5	V
		4.5V		3.15	3.15	V
		6.0V		4.2	4.2	V
Maximum Low Level Input Voltage		2.0V		0.3	0.3	V
		4.5V		0.9	0.9	V
		6.0V		1.2	1.2	V
Minimum High Level Output Voltage	V_{IN} = V_{IH} or V_{IL} |I_{OUT}| ≤ 20 μA	2.0V	2.0	1.9	1.9	V
		4.5V	4.5	4.4	4.4	V
		6.0V	6.0	5.9	5.9	V
	V_{IN} = V_{IH} or V_{IL} |I_{OUT}| ≤ 6.0 mA	4.5V	4.2	3.98	3.84	3.7	V
	|I_{OUT}| ≤ 7.8 mA	6.0V	5.7	5.48	5.34	5.2	V
Maximum Low Level Output Voltage	V_{IN} = V_{IH} or V_{IL} |I_{OUT}| ≤ 20 μA	2.0V	0	0.1	0.1	0.1	V
		4.5V	0	0.1	0.1	0.1	V
		6.0V	0	0.1	0.1	0.1	V
	V_{IN} = V_{IH} or V_{IL} |I_{OUT}| ≤ 6.0 mA	4.5V	0.2	0.26	0.33	0.4	V
	|I_{OUT}| ≤ 7.8 mA	6.0V	0.2	0.26	0.33	0.4	V
Maximum Input Current	V_{IN} = V_{CC} or GND	6.0V		±0.1	±1.0	±1.0	μA
Maximum TRI-STATE Output Leakage Current	V_{IN} = V_{IH} or V_{IL}, OC = V_{IH} V_{OUT} = V_{CC} or GND	6.0V		±0.5	±5	±10	μA
Maximum Quiescent Supply Current	V_{IN} = V_{CC} or GND I_{OUT} = 0 μA	6.0V		8.0	80	160	μA

Note 1: Absolute Maximum Ratings are those values beyond which damage to the device may occur.

Note 2: Unless otherwise specified all voltages are referenced to ground.

Note 3: Power Dissipation temperature derating — plastic "N" package: −12 mW/°C from 65°C to 85°C. Ceramic "J" package: −12 mW/°C from 100°C to 125°C.

Note 4: For a power supply of 5V ±10% the worst case output voltages (V_{OH} and V_{OL}) occur for HC at 4.5V. Thus the 4.5V values should be used when designing with this supply. Worst case V_{IH} and V_{IL} occur at V_{CC} = 5.5V and 4.5V respectively. (The V_{IH} value at 5.5V is 3.85V.) The worst case leakage current (I_{IN}, I_{CC}, and I_{OZ}) occur for CMOS at the higher voltage and so the 6.0V values should be used.