ปีการศึกษา 2534

รหัสแถบ

(BARCODE SYSTEM)

โดย

นางสาว นฤมล เลาหชัยบูณย์ เลขประจำตัว 31.1124

นางสาว ลลิตา ยาแก้ว เลขประจำตัว 31.1226

อาจารย์ที่ปรึกษา อาจารย์สุรพันธ์ เอื้อไพบูลย์

อาจารย์ที่ปรึกษาร่วม รศ.ดร.มนัส สังวรศิลป์

ปริญญานิพนธ์ ปีการศึกษา 2534

ภาควิชา ภาคอิเลคทรอนิคส์

คณะ วิศวกรรมศาสตร์

เรื่อง ระบบการใช้งานรหัสแถบ (Barcode System)

ผู้จัดทำ นางสาว นฤมล เลาหชัยบุญย์ เลขประจำตัว 31.1124

นางสาว ฉลิตา ยาแก้ว เลขประจำตัว 31.1226

---------- (อาจารย์ที่ปรึกษา)

ระบบใช้งานรหัสแถบ

นางสาว นฤมล เลาหชัยบุญย์ 31.1124

นางสาว ลลิตา ยาแก้ว 31.1226

อาจารย์ที่ปรึกษา

อาจารย์สุรพันธ์ เอื้อไพบูลย์

อาจารย์ที่ปรึกษาร่วม

รศ.ดร.มนัส สังวรศิลป์

ภาคการศึกษาที่ 1และ2 ปีการศึกษา 2534

บทคัดย่อ

การจัดการข้อมูลเกี่ยวกับงานทะเบียน งานสถิติ รายการสินค้า และอื่นๆ เข้ามามีบทบาทในชีวิตประจำวันมากยิ่งขึ้น ไม่ว่าจะในระบบธุรกิจ ระบบราชการ ดังนั้นจึงได้มีความพยายามหาวิธีการที่ดีที่สุด มีประสิทธิภาพ ใช้งานได้คล่องตัวมาประยุกต์ใช้ระบบการใช้งานรหัสแถบ (BARCODE SYSTEM) ก็เป็นงานอีกอย่างหนึ่งที่ได้พัฒนาขึ้นมา

ปริญญานิพนธ์ฉบับนี้ จะนำเสนอรายละเอียด หลักการทำงาน และวิธีการใช้เครื่องมือ จัดการทำงานของรหัสแถบ

ชนิดของรหัสแถบที่นิจารณาในระบบนี้ คือ รหัส3ใน9 เป็นรหัสแถบที่ครอบคลุมทุกๆ อักขระ ทั้งอักษร ตัวเลข และเครื่องหมายต่างๆ ทำให้ประยุกต์ใช้กับงานได้หลากหลาย สำหรับการประมวลผลจะถูกแบ่งแยกเป็น 2 ส่วน คือ ส่วนถอดรหัสข้อมูล และขบวนการจัดการข้อมูลบนคอมพิวเตอร์ ซึ่งข้อมูลถ่ายเทเข้าโดยผ่านระบบสื่อสารมาตรฐาน RS 232โดยรวมทั้งระบบแล้ว จะเน้นการทำงานผ่านทางซอฟท์แวร์ ภาษาแอสเซมบลี (ASSEMBLY 8031) ภาษาปาสคาล (TURBO PASCAL) ดังนี้แล้ว จะทำให้ระบบมีขนาดเล็ก ใช้งานได้สะดวกยิ่งขึ้น

BARCODE SYSTEM

ABSTRACT

Nowadays data managing such as registry , statistics and others are more necessary for daily life. Thus we must find the best method that has more efficient and more flexible for works to use. Barcode system is an example of developement.

This thesis shows the details and the use of the barcode instrument.

The type of used code is 3of9 code. It's the code that contains whole of character , integer and sign so that it can apply for many applications. The process is classified in 2 parts. One is the data decoding and the other is the data process on computer.

The data are conveyed to computer by passing through RS 232 communication system. We focus on using software ,the 8031 assembly and turbo pascal. Therefore the system has a small size and more convenient to use .

สารบัญ

บทที่ 1

บทนำ

การใช้งานรหัสแถบที่เห็นโดยทั่วไป คือ ตามหีบห่อสินค้าจะมีป้ายบรรจุแถบขาว-ดำ(สีอื่นก็มี) ตามซุปเปอร์มาร์เกตก็จะใช้หัวอ่านอ่านข้อมูลมาเก็บไว้ผ่านเครื่องคิดเงิน (Cash Register) นี่ก็คือการใช้งาน ณ.จุดขาย เมื่อมีการขายสินค้าเกิดขึ้น ก็จะทำการป้อนข้อมูลข้อรหัสสินค้า หรือใช้ในการจัดการเรื่องของรายการสินค้าในสต๊อค หรือใช้แสดงยอดเงิน ซึ่งเป็นการพัฒนาธุรกิจให้ทำงานคล่องตัวยิ่งขึ้น นอกจากนี้ยังมีการใช้งานทะเบียนอื่นๆอีกด้วย

เป็นที่ยอมรับโดยทั่วไปแล้วว่า คอมพิวเตอร์มีความสามารถในการประมวลผลข้อมูลได้อย่างรวดเร็ว ข้อผิดพลาดและเวลาในการทำงานของคอมพิวเตอร์กับมนุษย์นั้น เมื่อเทียบกันแล้วแตกต่างกันมาก แต่เพื่อมิให้ข้อผิดลาดเกิดขึ้น เราจะต้องป้อนข้อมูลที่ถูกต้องแก่เครื่อง การทำงานของระบบรหัสแถบก็เพื่องานลักษณะนี้ด้วยความสามารถ ดังนี้

- รวดเร็วกว่าเมื่อเทียบกับการป้อนข้อมูลผ่านแป้นพิมพ์ (key board)
- ถูกกว่า ไม่มีข้อผิดพลาด
- ทนทาน เพราะรหัสแถบไม่ใช้ชิ้นส่วนอิเลคโทรนิคส์

เครื่องคอมพิวเตอร์ถูกสร้างขึ้นมาเพื่ออ่านแต่ข้อมูลที่ถูกต้อง ลักษณะของรหัสแถบที่ดีจะต้องมีความผิดพลาดน้อย รหัสแสดงการเริ่มต้น การหยุด การตรวจการเรียงข้อความก่อนหลัง

บทที่2

ทฤษฎีของรหัสแถบ

รหัสแถบสามารถสร้างขึ้นมาโดยใช้ความแตกต่างระหว่างความกว้างและความแคบของแถบสี สิ่งที่ต้องระวังก็ คือ หมึกที่จะดูดกลืนแสงของหัวอ่าน โดยเฉพาะอย่างยิ่ง ถ้าเป็นหลอดแอลอีดี (LED) สีแดงจะใช้รหัสแถบที่มีแถบย่อยเล็กมากไม่ได้ และถ้าเป็นวิธีการพิมพ์ด้วยวิธียิงกระทบ (inject) จะต้องระวังความเข้มของหมึก

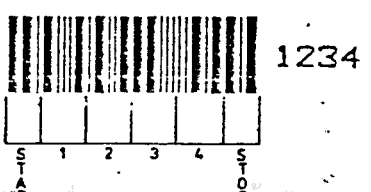
หลักการของรหัสแถบ

รหัสแถบเป็นการแทนข้อมูลที่เป็นรหัสเลขฐานสอง (Binary Code) ในรูปแบบของแถบขาว-ดำที่มีความกว้างของแถบต่างกัน ดังรายละเอียดต่อไปนี้

ชนิดของรหัสแถบ

1.ชนิดรหัส 2 ใน 5

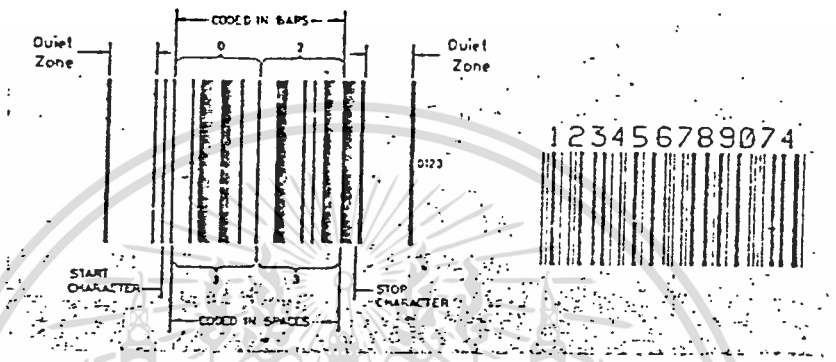
ใน 1 รหัสจะประกอบไปด้วย 5 แถบ (5 บิต) แต่จะมีแถบกว้างที่มีค่าเป็น 1 เพียง 2 แถบ (2 บิต) เท่านั้น ส่วนบิตที่เหลือเป็น 0 ทั้งหมด คือ การแทนด้วยแถบแคบ 3 แถบ โดยไม่นำส่วนที่เป็นช่องว่างมาใช้เลย รหัส 2 ใน 5 นี้เป็นรหัสที่ใช้แทนข้อมูลได้เฉพาะตัวเลข 0-9 เท่านั้น เริ่มต้นจาก Start Code 3 bit คือ 110 กับปิดท้ายด้วย Stop Code 3 bit คือ 101 ดังรูป 2.1



รป 2.1 รหัสแถบแบบ2ใน5

2.ชนิดรหัส 2 ใน 5 แบบสอดแทรก

รหัสแบบนี้คล้ายคลึงกับแบบแรกมาก ได้ดัดแปลงนำส่วนที่เป็นช่องว่างทั้ง 2 ชนิด มาใช้งานด้วย โดยการแทรกรหัสลงไปอีก 1 รหัส ทุก ๆ ช่วง 5 แถบของรหัสปกติที่เป็นแถบสีดำ แต่ก็ยังแทนรหัสได้เพียง 0-9 ดังรูป 2.2



รูป2.2 แสดงรหัสแถบ 2ใน5 แบบสอดแทรก

จากรูปใช้แทนรหัสตัวเลข 1 2 3 4 ... ตามลำดับ การใช้งานจะเริ่มต้นส่วนที่เป็น Start Code ทางด้านซ้ายประกอบด้วยแถบแคบ 2 แถบ และช่องว่างแคบ 2 แถบสลับกัน ส่วนทางด้านขวาเป็น Stop Code ประกอบด้วยแถบกว้าง 1 แถบ และแถบแคบ 1 แถบตามลำดับ ภายในระหว่าง Start และ Stop code แบ่งเป็น 2 ส่วน ส่วนแรก คือ ส่วนที่เป็นแถบดำกว้างและแคบ จะใช้แทนรหัสเหมือน 2 ใน 5 ปกติ จากตัวอย่าง 5 แถบแรกที่เป็นสีดำแทนค่าได้เท่ากับ 1 แถบขาวในช่วงเดียวกันเท่ากับ 2 สีดำช่วงต่อมาแทนได้เท่ากับ 3 สีขาวเท่ากับ 4 เช่นนี้ตลอดไปจนหมด รวมเป็นค่าที่อ่านได้เท่ากับ 1234...

3.ชนิดรหัส 3 ใน 9

เป็นรหัสที่ใช้แทนตัวอักขระทั้งหมด 44 ท้อักขระ เป็นอักษรตัวใหญ่ 26 รหัส เลข 0-9 10 รหัส และอักขระพิเศษอีก 8 รหัส เป็นการประยุกต์ใช้รหัส 2 ใน 5 โดยการนำเอาส่วนที่เป็นแถบดำ 5 แถบ และแถบว่าง 4 แถบ รวมเป็น 9 แถบแทน 1 รหัส ในแถบดำ 5 แถบนั้น ประกอบด้วยแถบกว้างที่เป็นบิต 1 อยู่ 2 แถบ และแถบแคบที่เป็นบิต 0 อีก 3 แถบ ส่วนแถบว่าง 4 แถบ ประกอบด้วยแถบกว้างที่เป็นบิต 1 อยู่ 1 แถบและแถบแคบเป็นบิต 0 อีก 3 แถบ ดังนั้นเมื่อรวมทั้งหมด 9 แถบ จะเป็นบิต 1 อยู่ 3 แถบและบิต 0 อยู่ 6 แถบ รหัส 3 ใน 9 มีส่วน Start Code และ Stop Code ด้วยรหัสเดียวกัน คือ * (Asterisk) ซึ่งมีรหัสฐาน 2 เป็น

แถบ 00110 และช่องว่าง 1000 ข้อดีของรหัสชนิดนี้ คือใช้งานได้กงวางขวางมากขึ้น รหัสทั้ง
หมดแทนได้ตามรูป 2.3

รูป 2.3แสดงรหัสแบบ 3ใน9

อักขระ	แพตเทิร์น	แถบ	ช่องว่าง	อักขระ	แพตเทิร์น	แถบ	ช่องว่าง
1		10001	0100	M		11000	0001
2		01001	0100	N		00101	0001
3		11000	0100	O		10100	0001
4		00101	0100	P		01100	0001
5		10100	0100	Q		00011	0001
6		01100	0100	R		10010	0001
7		00011	0100	S		01010	0001
8		10010	0100	T		00110	0001
9		01010	0100	U		10001	1000
0		00110	0100	V		01001	1000
A		10001	0010	W		11000	1000
B		01001	0010	X		00101	1000
C		11000	0010	Y		10100	1000
D		00101	0010	Z		01100	1000
E		10100	0010	-		00011	1000
F		01100	0010	.		10010	1000
G		00011	0010	SPACE		01010	1000
H		10010	0010	*		00110	1000
I		01010	0010	$		00000	1110
J		00110	0010	/		00000	1101
K		10001	0001	+		00000	1011
L		01001	0001	%		00000	0111

ตารางที่1 แสดงเลขรหัสฐานสองของรหัส 3ใน9

4.ชนิดรหัส Codabar

รหัส Codabar ประกอบด้วย 7 บิต โดย 4 บิตเป็นแถบดำ และ 3 บิตเป็นช่องว่างใช้แทนตัวเลข 0-9 เครื่องหมาย - $: /. + A B C และ D รหัส codabar ที่สมบูรณ์จะต้องมีรหัสที่ใช้แทนตัวอักษร A B C หรือ D เป็นส่วนเริ่มต้นหรือสิ้นสุด ภายในประกอบด้วยรหัสของ Codabar ที่เป็นตัวเลขและเครื่องหมาย ซึ่งทำให้มีความยาวไม่แน่นอนเพราะ 12 รหัสแรกมีบิทที่เป็น 1อยู่ 2บิท 4รหัสต่อมามีบิท 1อยู่ 3บิท (codabar ใช้ทั้งแถบดำและขาวแทนข้อมูลในหนึ่งรหัส) และ 4รหัสสุดท้ายเป็นรหัสของ A B-C D ที่กำหนดขึ้นมาเพื่อเป็นรหัสเริ่มต้นและสิ้นสุด

5. ชนิดรหัส UPC (Universal Product Code)

เป็นที่นิยมกับสินค้าหลายประเภท ดังรูป 2.4

รูป 2.4แสดงรหัสแบบ 3ใน9

จากรูปจะเห็นได้ว่า รหัสชนิดนี้แบ่งเป็น 2ส่วน ซึ่งถูกแบ่งด้วยแถบสีดำเล็กๆที่ยาวกว่าแถบอื่น 2 เส้นคั่นอยู่ตรงกลาง (เลขรหัสฐานสองของแถบคั่นกลางนี้เป็น 01010) และยังมีแถบลักษณะเดียวกัน2ชุดอยู่ทางซ้าย-ขวาสุด (เลขรหัสฐานสองของแถบนี้คือ 101) แถบทั้ง 3ชุดนี้ เรียกว่า Guide Bar ซึ่งปกติจะยาวกว่าแถบอื่น ทำให้แบ่งรหัสแถบเป็น 2ส่วน คือ ส่วนทางซ้าย และส่วนทางขวาเป็นตัวตรวจสอบความถูกต้อง (Check Digit) ซึ่งคำนวณมาจากหลักที่เหลือ โดยตรวจสอบทางซ้ายสุดมาจากเลข 5หลักทางซ้าย และหลักทางขวามาจากเลข 5หลักที่อยู่ทางด้านขวา ซึ่งแถบสำหรับตรวจสอบนี้บางครั้งก็นิมพ์าวเท่ากับส่วนที่เป็นไกด์บาร์ (guide bar) จากรหัสยูพีซี รหัสทางซ้ายจะใช้กับรหัสแถบแบบยูพีซีในส่วนทางซ้ายเท่านั้น จะใช้สลับกับทางขวาไม่ได้ ในส่วนของรหสทางซ้ายจะขึ้นต้นด้วยบิท 0 และลงท้ายด้วยบิท 1 เสมอ จะมีการตรวจสอบเป็นแบบบิทคี่ (odd parity) ส่วนรหัสทางขวาจะกลับกับรหัสทางซ้าย คือมีบิท 1 เป็นบิทเริ่มต้นและ 0 เป็นบิทสิ้นสุด การตรวจสอบบิทเป็นแบบบิทคู่ รหัสทางซ้ายและทางขวาธ้ง

เป็นเลขแบบ 1 คอมพลิเมนต์ (1's complement) ซึ่งกันและกัน ในแถบดำและแถบขาวยังแบ่งอย่างละ 4ขนาด คือแถบดำแคบสุด มีค่า 1 ขนาดที่ 2 กว้างกว่าขนาดแคบสุดเล็กน้อย มีค่า 11 และขนาดที่ 3 มีค่า 111 ส่วนแถบกว้างที่สุด มีค่า 1111ทำนองเดียวกับแถบขาว แต่ละรหสตัวเลขจะประกอบไปด้วย แถบขาว-ดำอย่างละ 2แถบ

ลักษณะของรหัสแถบที่ดี

- ความกว้างและจำนวนของแถบต่อรหสควรจะคงที่
- สามารถใช้แทนตัวเลขหรือตัวเลขปนอักษรได้ครบ
- มีโครงสร้างแบบง่ายๆ
- การอ่านด้วยความเร็วที่ต่างกันควรได้ค่าที่ถูกต้องเสมอ
- มีความหนาแน่นของข้อมูลต่อความกว้างของแถบสูง

บทที่3

ทฤษฎีเครื่องมือที่ใช้ในการอ่านรหัสแถบ (BARCODE READING DEVICE)

ส่วนแรกที่จะกล่าวถึงคือ หัวอ่าน เป็นส่วนที่ใช้ในการแยกข้อมูลที่ถูกจัดเข้าเป็นรหัสแถบและแปลงข้อมูลนั้นให้อยู่รูปแบบของดิจิตอล ขณะนี้ข้อมูลก็พร้อมสำหรับการประมวลผลด้วยซีพียู ข้อมูลที่ถูกถอดรหัสแล้วจะถูกส่งตรงไปยังคอมพิวเตอร์ หรือเก็บไว้ก่อนเพื่อรอการใช้ภายหลัง หรือใช้กับโปรแกรมประยุกต์ที่อยู่ในตัวหัวอ่าน

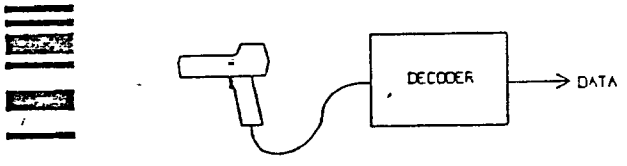
ข้อกำหนดพื้นฐาน 5 ประการสำหรับส่วนถอดรหัส

1. สามารถพิจารณาความกว้างแคบของแถบดำ และแถบขาว(space)
2. สามารถจัดแบ่งระดับของความกว้าง ทั้งนี้ขึ้นอยู่กับชนิดของรหัสใช้งาน เช่น

   แบ่งเป็น 2ระดับ สำหรับรหัส 3ใน9 , 2ใน5

   " 4ระดับ " ยูพีซี ,เอียน (EAN)
3. ให้ความมั่นใจได้ว่าความกว้างที่ถูกจัดแบ่งเหมาะสมกับการถอดรหัสสำหรับรหัสแต่ละชนิด สามารถเปรียบเทียบโครงสร้างความกว้างที่อ่านได้กับตารางข้อมูลหลักของรหัสนั้นๆ เพื่อแปลงเป็นรหัสแอสกี้ (ASCII CODE)
4. ถ้าการกลับลำดับมีผลต่อการแปลงรหัส ทิศทางการอ่านจะถูกกำหนดโดยการตรวจสอบอักขระสตาร์ท และสต็อป
5. ยืนยันได้ว่า มีบริเวณนำหน้ารหัส (quite zone)ที่ปลายทั้งสองข้างของรหัสแถบ

โดยรวมแล้ว หัวอ่านประกอบด้วย อิเล็คโตร-ออปโต สกรีน (electro-opto screen) รวมเข้ากับซอร์ฟแวร์ของส่วนประมวลผล ดังแสดงในรูป 3.1 ซึ่งประกอบด้วย 2ส่วน ดังนี้



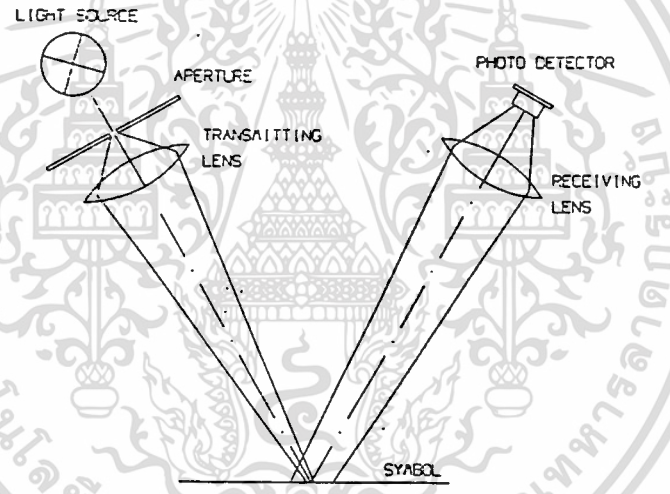
รูปที่ 3.1 แสดงระบบหัวอ่าน

1.ส่วนนำข้อมูลเข้า (INPUT DEVICES)

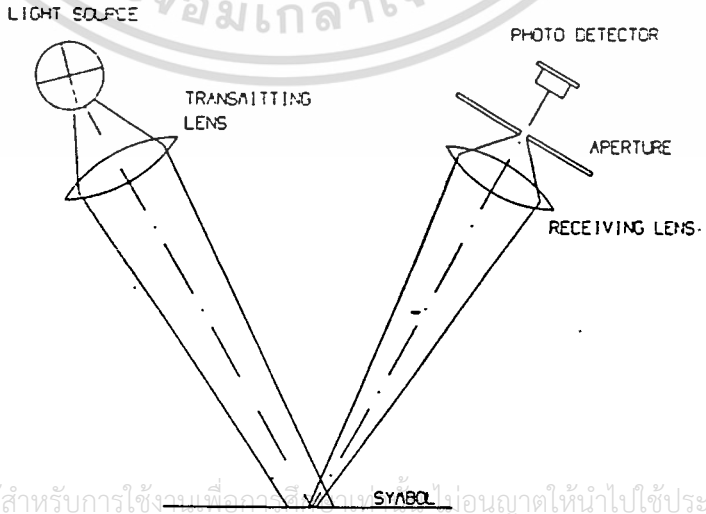
เป็นหน่วยที่ใช้เทคนิคการแปลงแสงไปเป็นไฟฟ้าในการสแกนผ่านรหัสแถบ ลักษณะการลากขึ้นอยู่กับการเคลื่อนไหวมือของผู้ลาก หรือการสแกนภายใน หรือ โดยการเคลื่อนรหัสแถบผ่าน สัญญาณเอาท์พุทที่เกิดขึ้นขณะนั้น คือตัวแทนของการสะท้อนแสงในขณะนั้น ณ.จุดที่ถูกสแกน

เครื่องมือส่วนนี้ มักจะเป็น แอคทีฟ ซิสเต็ม (active system) มันทำงานด้วยการส่งพลังงานแสงไปยังรหัสแถบ แล้วตรวจสอบจำนวนแสงที่สะท้อนกลับมา ส่วนสว่างจะสะท้อนแสงได้มากกว่าส่วนมืด

พื้นที่ของรหัสซึ่งถูกตรวจสอบนี้เรียกว่า สปอต(spot) สปอตควรจะประกอบด้วย ความกว้างที่แคบที่สุดของรหัสแถบที่ถูกสแกน สปอตสามารถอยู่ในแบบที่มีการเก็บแสงแบบกว้าง จากแสงที่โฟกัสแม่นยำ ดังรูป 3.2 หรือโดยการปล่อยแสงแบบกว้างแล้วให้แสงถูกโฟกัสผ่านช่องเก็บแสง ดังรูป 3.3

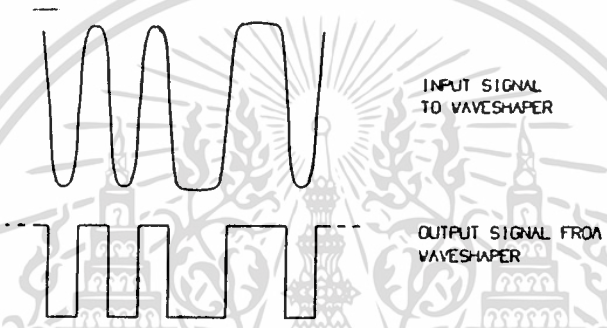


รูปที่ 3.2 แสดงการโฟกัสแสงที่แม่นยำ



รูปที่ 3.3 แสดงการบีบแสงให้โฟกัสผ่านช่องเก็บแสง

แสงที่สะท้อนจากสปอต จะถูกบังคับทิศทางไปยังโฟโตไดโอด (photo diode detector) ซึ่งจะสร้างกระแสค่าหนึ่งไม่มากนัก แต่เป็นสัดส่วนกับแสงที่สะท้อนมา แอปลิฟายในหัวอ่านจะขยายสัญญาณจากโฟโตไดโอดให้อยู่ในระดับใช้งาน สังเกตุว่า อนาลอก โวลต์เตจเป็นสัดส่วนกับการสะท้อนแสง เพื่อจะแยกแถบขาวและดำ อนาลอก โวลต์เตจจะถูกแปลงเป็นรูปคลื่นดิจิตอล (digital waveform) ด้วยวงจรแปลงรูปคลื่น (wave shaper) ดังรูป



รูปที่ 3.4 แสดงสัญญาณจากโฟโตไดโอด

ในการสแกนด้วยความเร็วคงที่ เป็นการง่ายที่จะตัดสินความกว้าง แคบของแถบขาวและดำจริงด้วยการวัดค่าเวลา ในอีกแง่หนึ่งก็คือการลากด้วยความเร็วคงที่ เป็นการง่ายที่จะเปลี่ยนจากไทม์โดเมนไปเป็น สเปซโดเมน (space domain) แต่ถ้าการลากมีความเร่งเป็นการยากที่จะวัดความกว้าง แคบของแถบขาว ดำจากการตรวจสอบสัญญาณไฟฟ้าจริงจากหัวอ่าน ความง่ายและความแม่นยำขึ้นอยู่กับซอร์ฟแวร์ในส่วนการถอดรหัส (decoder part)

ความยาวคลื่น หลายค่าที่ได้จากการสแกน แต่มักจะอยู่ในหน่วย นาโนเมตร ในระยะแรกมักจะใช้ย่านอินฟราเรด 900 nm. ปัจจุบันจะใช้สเปคตรัมในย่านแสงที่ตามองเห็น 750-450nm.

เทคนิคการพิมพ์บางอย่างไม่ได้แยกความตัดกันได้เพียงพอเมื่อใช้ช่วงอินฟราเรด ในขณะที่การพิมพ์อื่นๆโดยเฉพาะที่เป็นการพิมพ์ด้วยหมึกคาร์บอน (carbon based ink) ให้ความตัดกันเพียงพอสำหรับช่วงแสงที่ตามองเห็น

ความยาวคลื่นทั่วๆไปที่ใช้ในการสแกน มักจะเป็น 633 และ 900 nm. เนื่องจากค่า 900nm.

จะมองไม่เห็นด้วยตาเปล่า จึงต้องมีการพิมพ์แบบพิเศษ (security badge application) ในบางสภาพถ้ารหัสแถบ ถูกทำให้เปื้อนได้ง่ายด้วยจารบี น้ำมัน หรือว่าเลือด ค่า 900nm.จะใช้งานได้ดีกว่า

ชนิดของหัวอ่าน

	FIXED BEAM	MOVING BEAM
HAND HELD	CONTACT NON-CONTACT	NON-CONTACT
FIX MOUNT	NON-CONTACT	NON-CONTACT

1.HANDHELD FIX BEAM CONTACT

การใช้งานจะต้องสัมผัสกับรหัสแถบ ส่วนประกอบภายในจะไม่มีส่วนกลไกใดๆที่จะช่วยการสแกนให้เป็นแบบอัตโนมัติ ลำแสงที่ใช้จะคงที่ การสแกนขึ้นกับผู้ลาก มีด้วยกัน 2แบบ คือ 1. WAND 2. CONTACT GUN สำหรับ WAND เรียกได้อีกอย่างว่า LIGHT PEN รอยแตะซึ่งเกิดจากการสัมผัสควรจะมีขนาดเล็กที่สุด เพื่อลดความเสียหายที่จะเกิดจากรอยขีดข่วน

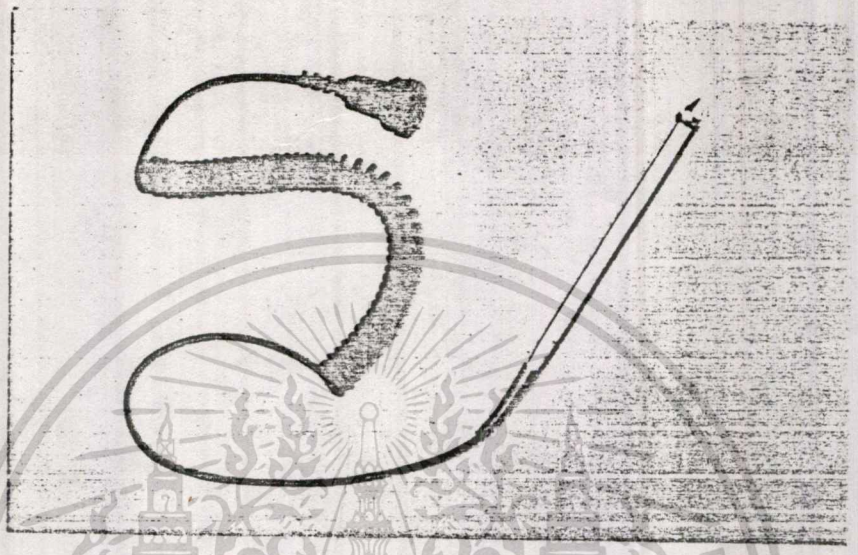
ผู้ใช้ควรจะมีทักษะในการใช้งาน และควรจะเรียนรู้ข้อผิดพลาดที่มักเกิดขึ้น ซึ่งเป็นเรื่องไม่ยาก ดังนี้ 1.การสแกนช้าไปทำให้เกิดการกระตุก

2.การหยุดลากก่อนจะสิ้นสุดรหัสจริงๆ (quiet zone)

3.ไม่เริ่มการลากที่จุดแรกจริงๆ (leading quiet zone)

4.ลากผ่านเลยแถบดำแถบสุดท้าย เช่นหัวอ่านลอยขึ้น

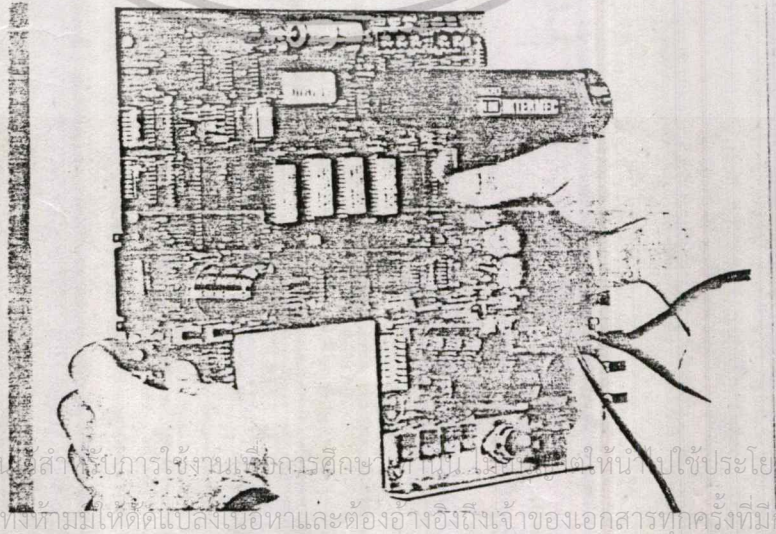
แม้ว่าจะมีข้อจำกัดในการใช้งาน แต่หัวอ่านชนิดนี้ก็เป็นที่นิยมใช้ทั่วๆไป เพราะราคาถูก สะดวก ดังรูปที่ 3.5



รูปที่ 3.5 แสดงหัวอ่านแบบ WAND

2.HAND HELD FIX BEAM NON-CONTACT

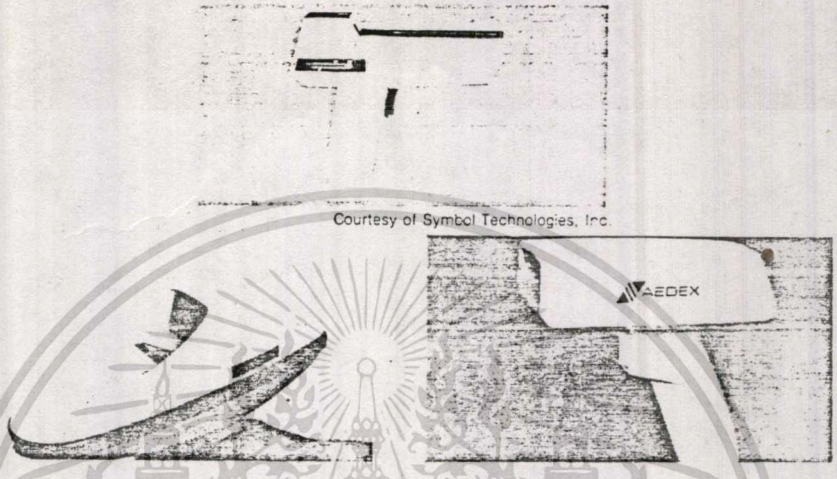
ตัวสแกนชนิดนี้ก็ยังขึ้นอยู่กับการเคลื่อนมือของผู้ใช้ แต่จุดโฟกัสจะอยู่ห่างจากปลายหัวอ่าน ดังนั้น การสัมผัสนรหัสแถบจึงไม่จำเป็น ดังรูปที่ 3.6



รูปที่ 3.6 แสดงหัวอ่านแบบมือถือ ลำแสงคงที่ ไม่สัมผัส

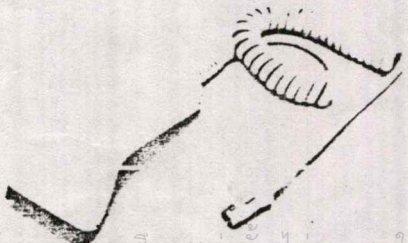
3.HAND HELD MOVING BEAM

มักจะใช้เลเซอร์ (solid state lazer) หรือ อีเลียม-นีออน เป็นแหล่งแสง การสแกนถูกสร้างโดยการหมุนกระจก ภายในตัวสแกน ดังรูป 3.7

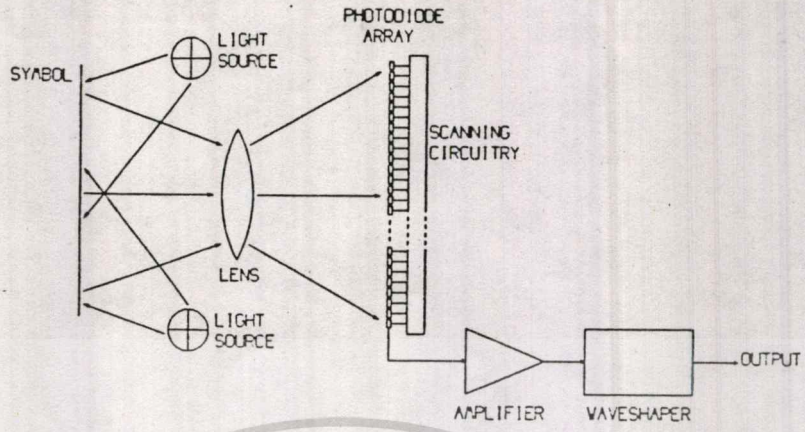


รูปที่ 3.7 แสดงหัวอ่านแบบมือถือ ลำแสงกวาด

ซีซีดี (CCD charged-couple device) ดังรูป 3.8 ก็จัดอยู่ในประเภทนี้ แต่ไม่ใช้หลักกลไก หลักการของมันก็คือ ปล่อยแสงคลอบคลุมทั้งรหัสแถบ ทำการดิจิไตส์(digitize) โดยอาศัยแผงลิเนียร์ โฟโตไดโอด (linear photo diode array) ส่วนการสแกนจริงๆทำโดยวงจรเฉพาะ (scaning circuitry) จากแต่ละไดโอดในแบบลำดับ(sequential)และอาศัย charged-couple device semiconductor technology ดังรูป3.9



รูปที่ 3.8 แสดงหัวอ่านแบบซีซีดี



รูปที่ 3.9แสดงการทำงานแบบซีซีดี

4.FIX MOUNT FIX BEAM

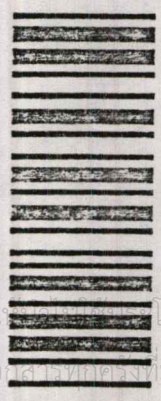
การสแกนอาศัยการเคลื่อนตัวของรหัสแถบ การประยุกต์ใช้งานทั่วไป คือติดรหัสแถบไว้บนวัตถุที่เคลื่อนไปบนสายพาน ดังนั้นจึงมีโอกาสสแกนเพียงครั้งเดียว จึงต้องใช้การพิมพ์รหัสแถบที่มีคุณภาพสูง หัวอ่านแสงดังรูปที่ 3.10

รูปที่ 3.10 แสดงหัวอ่านแบบยึด ลำแสงคงที่

5.FIX MOUNT MOVING BEAM

ก็ใช้หลักการเดิมประกอบกัน สามารถใช้ได้กับทั้งรหัสแถบแนวนอน(picket fence) แนวตั้ง (ladder) ดังรูปที่ 3.11

PICKET FENCE ORIENTATION



LADDER ORIENTATION

รูปที่ 3.11 แสดงรหัสแถบแนวนอน และแนวตั้ง

2.ส่วนการถอดรหัส (DECODER)

ตามที่ได้บรรยายไว้ข้างต้น ส่วนการถอดรหัสของระบบการอ่าน จะวิเคราะห์สัญญาณที่ได้จากการอ่าน ถอดรหัสข้อมูลที่ถูกบรรจุไว้ในรหัสแถบนั้น

ขบวนการถอดรหัส

ระบบการทำงานจะต้องประกอบโมดูลถอดรหัส หน้าที่นี้มักจะใช้ซอร์ฟแวร์ ทำงานบนไมโครโปรเซสเซอร์แต่อาจจะใช้ฮาร์ดแวร์แทนก็ได้ วิธีการใดก็ตามที่จะใช้จะต้องมีขั้นตอนเหล่านี้เป็นสำคัญ

1.แบ่งแยกให้ได้ว่า สัญญาณที่อ่านอยู่เป็นแถบขาว หรือ ดำ

2.วัดความกว้างของแต่ละสัญญาณที่อ่านได้จากการสแกน

3.การแบ่งแยกระดับความกว้างโดยอาศัยอัลกอริทึมต่างๆ (algorithm)

4.ถอดรหัสจากแถบ โดยการเปรียบเทียบค่าความกว้างที่ถูกแบ่งแยกกับตารางที่แสดงค่าจำกัดสำหรับแต่ละอักขระ

5.มีอัลกอริทึม (algorithm) บังคับทิศทางการสแกน ถ้ารหัสชนิดนั้นสแกนได้ทิศทางเดียว

6.การตรวจสอบเพิ่มเติมเพื่อที่จะยืนยันความถูกต้องของการสแกน

-บริเวณที่ว่างที่ถูกต้อง (quiet zone)

-อักขระตรวจสอบ

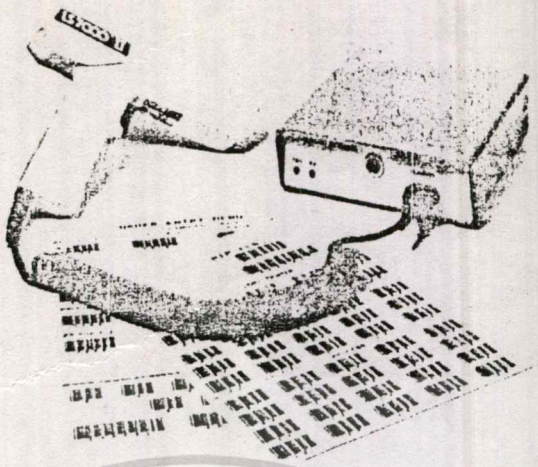
-การสังเกตความเร็วการสแกนจะต้องอยู่ในค่าจำกัดที่กำหนดไว้แล้ว

7.การส่งข้อมูลที่ถอดรหัสได้ที่มีผลต่อการถอดรหัสครั้งต่อไป

ชนิดของตัวถอดรหัส

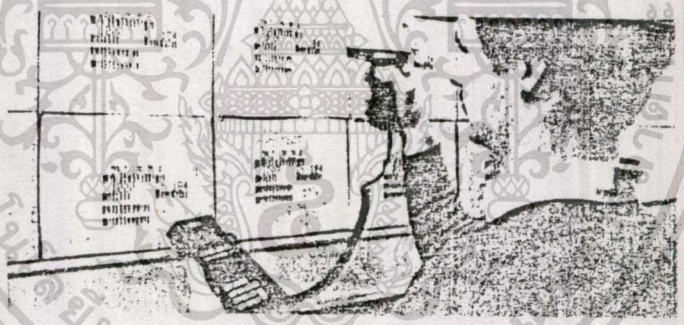
ขึ้นอยู่กับแหล่งจ่ายไฟ และการสื่อสารข้อมูล

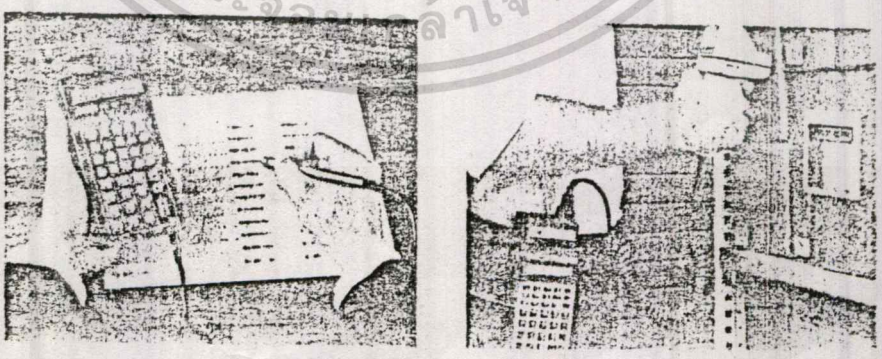
1.ออนไลน์ (On line decoder) ใช้ไฟเช่นเดียวกับไฟบ้าน (AC Line) มีการถ่ายเทข้อมูลระหว่างตัวถอดรหัสส่วนประมวลผลที่อยู่กับที่ เช่น คอมพิวเตอร์ ดังรูป 3.12



รูปที่ 3.12 แสดงส่วนตัวถอดรหัสแบบ ออนไลน์

2.พอร์ทเทเบิล (Portable) ประกอบด้วยส่วนเก็บข้อมูลบนบอร์ดและใช้แบตเตอรี่แห้ง ข้อมูลจะถูกเก็บไว้ให้คงที่ในขณะที่สะสมข้อมูล ดังรูป 3.13 จากนั้นเมื่อต้องการใช้งานจึงดึง(dumped) ออกมาต่อเข้ากับโฮสท์ คอมพิวเตอร์ (host computer)





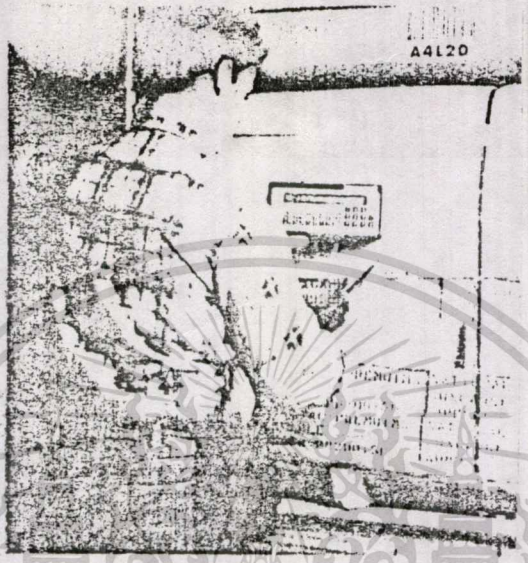
รูปที่ 3.13 แสดงตัวถอดรหัสแบบพอร์เทเบิล

การใช้งาน มักจะนำไปตรวจรหัสแถบที่อยู่กับที่ เช่น การตรวจสอบสต็อคสินค้า

ส่วนประกอบหลัก คือ ซีพียูประมวลผล รอม(ROM) แรม(RAM) และส่วนดิสเพล (Display)

3.พอร์ทเทเบิล ไร้สาย (Wireless Portable) การทำงานก็คล้ายพอร์ทเทเบิล แต่การนำ

กลับไปเชื่อมต่อจะใช้คลื่นวิทยุ ข้อมูลสามารถส่งโดยตรงไปยังคอมพิวเตอร์ ข้อมูลที่แสดงการผิดพลาด หรือเป็นสัญญาณกระตุ้นก็สามารถส่งจากคอมพิวเตอร์ไปยังตัวถอดรหัส

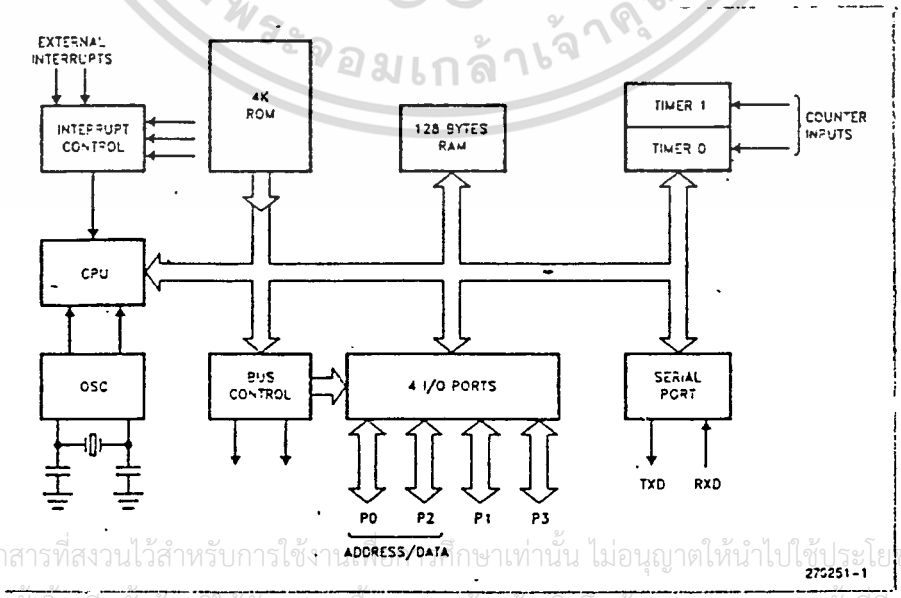


บทที่4

ไมโครคอนโทรลเลอร์ 8031 (Introduction to 8031)

ชิพเบอร์ 8031นี้ เป็นต้นแบบสำหรับตระกูลเอ็มซีเอส 51(MCS-51) ลักษณะโดยทั่วไปของ8031

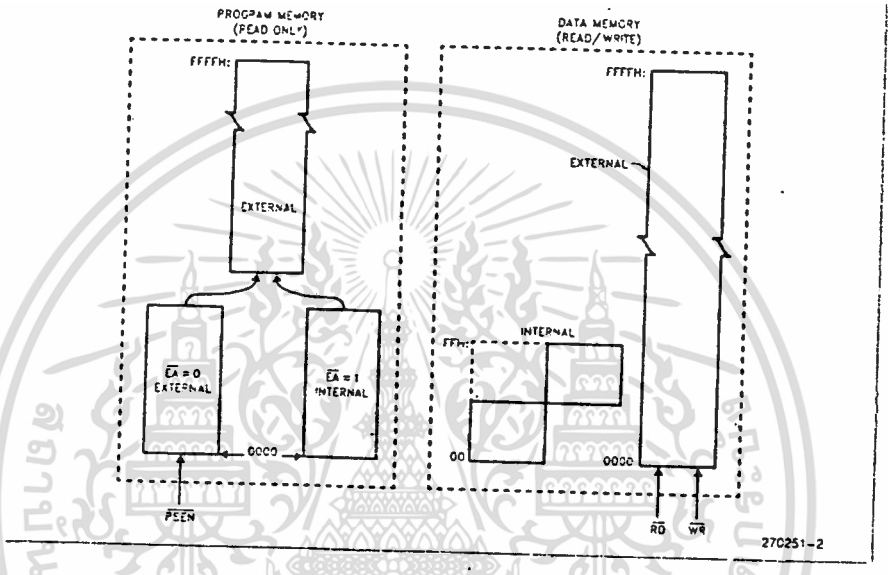
- 8 บิทซีพียู สำหรับควบคุมการใช้งาน
- ความสามารถเพิ่มเติมเกี่ยวกับ การจัดการบูลลีน (Boolean Single-bit logic)
- 64K หน่วยความส่วนโปรมแกรม (Program Memory) และ64K หน่วยความจำข้อมูล (Data Memory)
- 4K ไบท์ หน่วยความจำโปรมแกรมบนชิพ (8051และอื่นๆ ยกเว้น 8031)
- 128ไบท์ ดาต้าแรมบนชิพ (Data Ram)
- 32 สาย 2ทิศทาง อินพุท เอาท์พุทพอร์ท
- 2ชุด 16บิท ตัวจับเวลา และตัวนับ (Timer/counter)
- การสื่อสาร 2ทิศทาง (Full Duplex UART)
- 6 แหล่งการเกิดอินเทอร์รัพท์ พร้อม 2ระดับความสำคัญ
- สัญญาณนาฬิกาบนชิพ



รูปที่ 4.1 แสดงบล็อคไดอแกรมของ 8031

การจัดการหน่วยความจำ

จะใช้ค่าทางลอจิกในการแยกหน่วยความจำส่วนโปรแกรม ออกจากหน่วยความจำข้อมูล ดังรูป 4.2

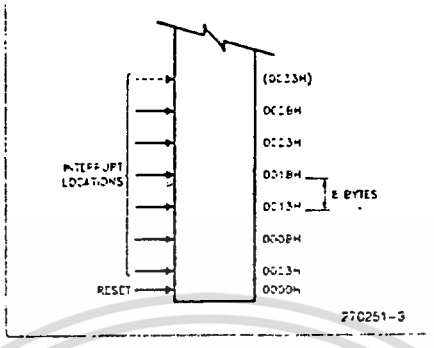


รูปที่ 4.2 แสดงการแยกหน่วยความจำ

ไมโครคอนโทรลเลอร์ตระกูลนี้ จะมีการแยกสองส่วนนี้ออกจากกัน หน่วยความจำข้อมูลจะถูกใช้งานด้วยสัญญาณ 8เส้นแอสเดรส ทำให้มีความรวดเร็วในการเก็บค่า การติดต่อกับซีพียู อย่างไรก็ตาม 16 เส้นแอสเดรส ก็สามารถทำได้ผ่าน ดีพีทีอาร์ (DPTR Reg.) สำหรับส่วนโปรแกรมสามารถอ่านได้เท่านั้น เขียนไม่ได้ สัญญาณสโตรปการอ่านสำหรับหน่วยความจำโปรมแกรมภายนอก คือ PSEN มีขนาดใหญ่ได้ถึง 64K แต่ถ้าเป็นส่วนหน่วยความจำข้อมูล จะใช้ RD,WR

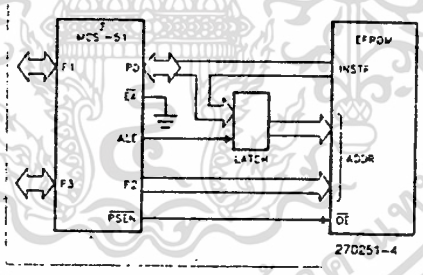
หน่วยตวามจำโปรแกรม

รูป 4.3แสดงหน่วยความจำส่วนล่าง หลังจากการรีเซ็ท ซีพียูจะเริ่มทำงานที่ 0000H จากรูปจะเห็นว่า อินเทอร์แต่ละชนิดถูกกำหนดตำแหน่งไว้แล้ว แต่ละตำแหน่งห่างกัน 8 ไบท์ การอินเทอร์รัพท์ทำให้ซีพียูกระโดดไปที่แอดเดรสนั้นเพื่อทำการบริการ ถ้าไม่ใช้งานเป็นอินเทอร์รัพท์ที่ว่างนั้นก็ใช้เป็นหน่วยความจำ สัญญาณ EA (External Access) จะกำหนดให้หน่วยความจำเป็นแบบภายในหรือภายนอก สัญญาณ PSENใช้สำหรับการเฟทช์หน่วยความจำภายนอกเท่านั้น



รูปที่ 4.3 แสดงหน่วยความจำส่วนล่าง

ลักษณะทางฮาร์ดแวร์สำหรับการจัดการหน่วยจำภายนอกแสดงดังรูป 4.4



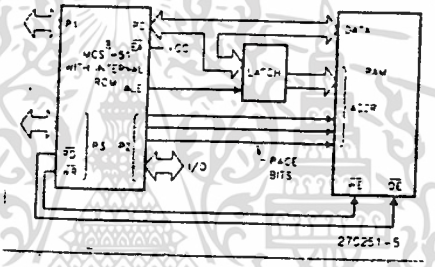
รูปที่ 4.4 แสดงการต่ออาร์ดแวร์หน่วยความจำภายนอก

สังเกตุพอร์ท 0,2 จะถูกใช้ในลักษณะบัส พอร์ท0 ใช้เป็นมัลติเพลกซ์ข้อมูลกับแอดเดรส แสดงไบท์ต่ำของพีซี(PC Program counter) หลังจากนั้นจะอยู่ในสถานะลอย (Float State) รอการรับคำสั่งจากหน่วยความจำโปรแกรม ระหว่างที่พอร์ท0เก็บค่าไบท์ต่ำสัญญาณ ALE จะแลทช์ค่านี้ไว้ ขณะเดียวกันพอร์ท2 จะแสดงไบท์สูงของพีซี แล้ว PSENจะสโตรปเอาคำสั่งมาจากอีพรอม หน่วยความจำโปรมแกรมจะใช้ 16เส้นแอดเดรส แม้ว่าขนาดความจำจะเล็กกว่า 64 K

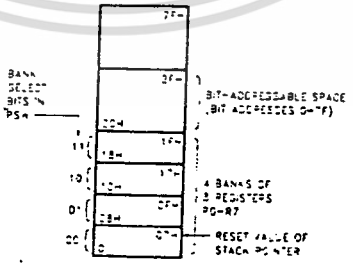
หน่วยความจำข้อมูล

รูปที่ 4.5 แสดงการใช้งานหน่วยความจำภายนอกขนาด 2K กรณีนี้ซีพียูจะทำงานกับรอมภายใน

พอร์ท0ใช้งานเป็นแอดเดรส/ข้อมูล และ 3เส้นของพอร์ท2เป็นบิทเพจ (Page Bit) สามารถขยายหน่วย
ความจำเป็น 64K อาจจะอ้างแอดเดรสเป็น 1หรือ2ไบท์ 1ไบท์จะใช้ในกรณีที่มีบิทเพจ (Page Bit)
2ไบท์ใช้เมื่อพอร์ท2 ทำหน้าที่แสดงแอดเดรสไบท์บน การจัดหน่วยความจำของส่วนภายในแสดงดังรูป 4.6
แบ่งเป็น 3ส่วน ส่วนล่าง128ไบท์ ส่วนบน128ไบท์ รีจิสเตอร์พิเศษ



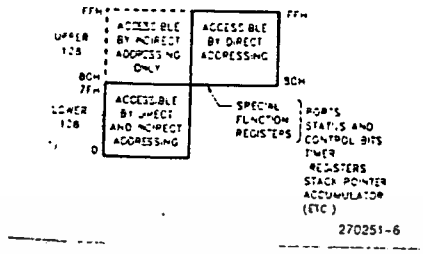
รูปที่ 4.5 แสดงการใช้งานหน่วยความจำภายนอก

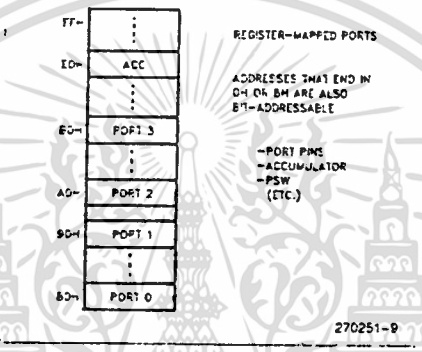


รูปที่ 4.6 แสดงการจัดหน่วยความจำภายใน

ปกติแล้ว หน่วยความจำภายในจะใช้แอดเดรสขนาด 1ไบท์แสดง 256ตำแหน่ง อย่างไรก็ตาม
สามารถขยายเป็น 384 ไบท์ โดยใช้แยกการอ้างแอดเดรสช่วง 80-FFH เป็นทางตรง และทางอ้อม

ส่วนล่าง 128ไบท์ แสดงดังรูป 4.7 32ไบท์ล่างสุดถูกแบ่งเป็น 4แบงก์ๆละ 8รีจิสเตอร์ (R0-R7) กำหนดการเลือกจาก 2บิท ของรีจิสเตอร์พีเอสดับบลิว (PSW Reg.) ถัดขึ้นมาอีก 16ไบท์เป็นส่วนที่ใช้กับการทำงานแบบบิท (bit address) บิทที่ 0-7FH 128บิท





รูปที่ 4.7 แสดงการขยายหน่วยความจำภายใน

SFR MEMORY MAP

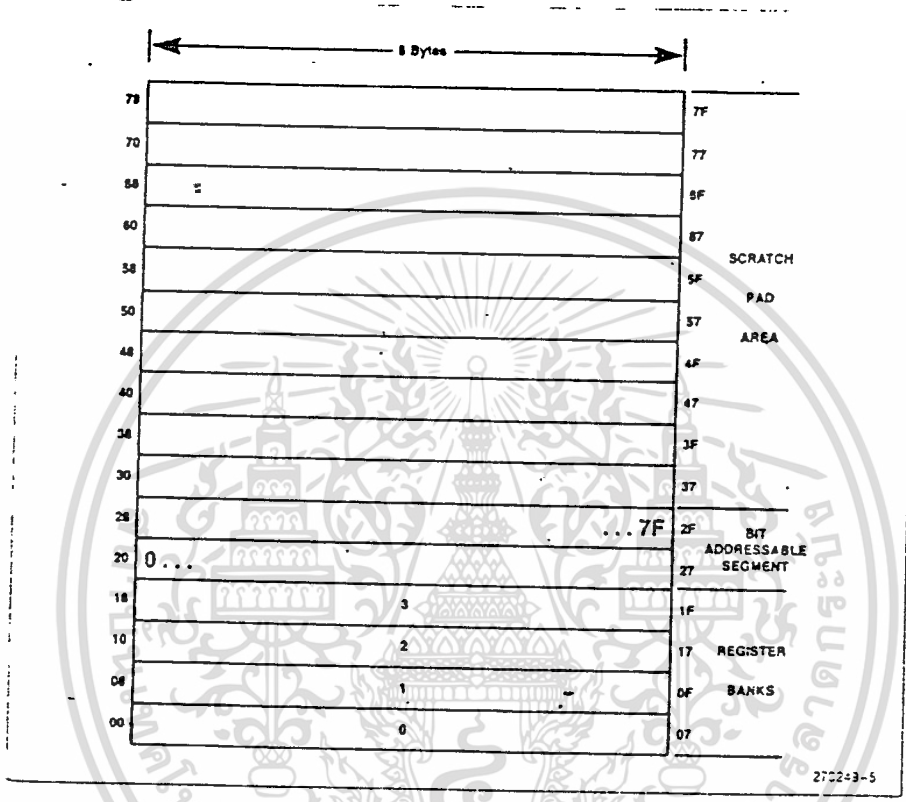
	8 Bytes								
F8									FF
F0	B								F7
E8									EF
E0	ACC								E7
D8									DF
D0	PSW								D7
C8	T2CON		RCAP2L	RCAP2H	TL2	TH2			CF
C0									C7
B8	IP								BF
B0	P3								B7
A8	IE								AF
A0	P2								A7
98	SCON	SBUF							9F
90	P1								97
88	TCON	TMOD	TL0	TL1	TH0	TH1			8F
80	P0	SP	DPL	DPH				PCON	87

↑
Bit Addressable

รูปที่ 4.8แสดงรายละเอียดของรีจิสเตอร์พิเศษ (SFR Speacial Function Register)

ทำหน้าที่แลทช์ค่าของพอร์ทต่างๆ ค่าตัวจับเวลา และการควบคุมอุปกรณ์รอบนอก รีจิสเตอร์เหล่านี้จะต้องอ้างแอดเดรสแบบโดยตรงเท่านั้น 16ตำแหน่งในเอสเอฟอาร์ (SFR)ใช้งานทั้งแบบบิทและแบบไบท์ แบบบิทใช้กับแอดเดรสที่ลงท้ายด้วย 000B

รายละเอียดเพิ่มเติมแสดงดังรูป 4.9 และตารางที่4.1



Symbol	Name	Address
*ACC	Accumulator	0E0H
*B	B Register	0F0H
*PSW	Program Status Word	0D0H
SP	Stack Pointer	81H
DPTR	Data Pointer 2 Bytes	
DPL	Low Byte	82H
DPH	High Byte	83H
*P0	Port 0	80H
*P1	Port 1	90H
*P2	Port 2	0A0H
*P3	Port 3	0B0H
*IP	Interrupt Priority Control	0B8H
*IE	Interrupt Enable Control	0A8H
TMOD	Timer/Counter Mode Control	89H
*TCON	Timer/Counter Control	88H
*+ T2CON	Timer/Counter 2 Control	0C8H
TH0	Timer/Counter 0 High Byte	8CH
TL0	Timer/Counter 0 Low Byte	8AH
TH1	Timer/Counter 1 High Byte	8DH
TL1	Timer/Counter 1 Low Byte	8BH
+ TH2	Timer/Counter 2 High Byte	0CDH
+ TL2	Timer/Counter 2 Low Byte	0CCH
+ RCAP2H	T/C 2 Capture Reg. High Byte	0CBH
+ RCAP2L	T/C 2 Capture Reg. Low Byte	0CAH
*SCON	Serial Control	98H
SBUF	Serial Data Buffer	99H
PCON	Power Control	87H

* = Bit addressable
+ = 8052 only

การอ้างแอดเดรส

1.ทางตรง(Direct)ใช้ได้กับ หน่วยความจำข้อมูลภายใน และ รีจิสเตอร์พิเศษเท่านั้นโอเปอร์แรนด์ขนาด 8บิท

2.ทางอ้อม(Indirect)ใช้กับรีจิสเตอร์ R0,R1 หรือ Stack Pointer เป็นค่าแอดเดรสของโอเปอร์แลนด์ ใช้ได้กับทั้ง แรมภายใน ภายนอก ถ้าเป็นรีจิสเตอร์ 16 บิทต้องเป็นดีพีทีอาร์

3.ทางรีจิสเตอร์ R0-R7ของทุกแบงก์(Register Instructions) ใช้ได้กับคำสั่งที่ทำงานกับรีจิสเตอร์โดยตรง แสดงโดย3บิทที่บรรจุอยู่ในออปโคด คำสั่งเหล่านี้จะมีจำนวนไบท์น้อย

4.เจาะจงรีจิสเตอร์ (Register-specfic Instruction) คำสั่งที่ถูกเจาะจงให้ใช้กับบางรีจิสเตอร์ เช่น ACC

5.ทันที (Immediate) เช่น MOV A,#100 ค่าคงที่นี้จะเป็นฐาน 16ก็ได้

6.การชี้ (Index Addressing) สำหรับหน่วยความจำโปรแกรมเท่านั้น อ่านได้อย่างเดียว ใช้ในการค้นตารางโดย PC หรือ DPTRเป็นค่าฐาน ใช้ในคำสั่งกระโดด (JUMP)

ชุดคำสั่ง

คำสั่งของ MCS-51 ใช้ได้อย่างมีประสิทธิภาพกับการควบคุมแบบ8บิทด้วยการอ้างแอดเดรสหลายๆแบบที่ทำงานรวดเร็ว ในการเข้าถึงแรมภายในทำให้การโอเปอเรทแบบไบท์ทำงานด้วยคำสั่งที่สั้น อีกทั้งขยายการใช้งานไปในแบบบิทด้วย

MCS®-51 INSTRUCTION SET

Table 10. 8051 Instruction Set Summary

Interrupt Response Time: Refer to Hardware Description Chapter.

Instructions that Affect Flag Settings(1)

Instruction	Flag			Instruction	Flag		
	C	OV	AC		C	OV	AC
ADD	X	X	X	CLR C	0		
ADDC	X	X	X	CPL C	X		
SUBB	X	X	X	ANL C,bit	X		
MUL	0	X		ANL C,/bit	X		
DIV	0	X		ORL C,bit	X		
DA	X			ORL C,bit	X		
RRC	X			MOV C,bit	X		
RLC	X			CJNE	X		
SETB C	1						

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

Note on instruction set and addressing modes:

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

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

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

#data — 8-bit constant included in instruction.

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

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

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

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

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

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

Table 10. 8051 Instruction Set Summary (Continued)

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

Table 10. 8051 Instruction Set Summary (Continued)

Mnemonic		Description	Byte	Oscillator Period
DATA TRANSFER (Continued)				
MOV	@Ri,direct	Move direct byte to indirect RAM	2	24
MOV	@Ri,#data	Move immediate data to indirect RAM	2	12
MOV	DPTR,#data16	Load Data Pointer with a 16-bit constant	3	24
MOVC	A,@A+DPTR	Move Code byte relative to DPTR to Acc	1	24
MOVC	A,@A+PC	Move Code byte relative to PC to Acc	1	24
MOVX	A,@Ri	Move External RAM (8-bit addr) to Acc	1	24
MOVX	A,@DPTR	Move External RAM (16-bit addr) to Acc	1	24
MOVX	@Ri,A	Move Acc to External RAM (8-bit addr)	1	24
MOVX	@DPTR,A	Move Acc to External RAM (16-bit addr)	1	24
PUSH	direct	Push direct byte onto stack	2	24
POP	direct	Pop direct byte from stack	2	24
XCH	A,Rn	Exchange register with Accumulator	1	12
XCH	A,direct	Exchange direct byte with Accumulator	2	12
XCH	A,@Ri	Exchange indirect RAM with Accumulator	1	12
XCHD	A,@Ri	Exchange low-order Digit indirect RAM with Acc	1	12

Mnemonic		Description	Byte	Oscillator Period
BOOLEAN VARIABLE MANIPULATION				
CLR	C	Clear Carry	1	12
CLR	bit	Clear direct bit	2	12
SETB	C	Set Carry	1	12
SETB	bit	Set direct bit	2	12
CPL	C	Complement Carry	1	12
CPL	bit	Complement direct bit	2	12
ANL	C,bit	AND direct bit to CARRY	2	24
ANL	C,/bit	AND complement of direct bit to Carry	2	24
ORL	C,bit	OR direct bit to Carry	2	24
ORL	C,/bit	OR complement of direct bit to Carry	2	24
MOV	C,bit	Move direct bit to Carry	2	12
MOV	bit,C	Move Carry to direct bit	2	24
JC	rel	Jump if Carry is set	2	24
JNC	rel	Jump if Carry not set	2	24
JB	bit,rel	Jump if direct Bit is set	3	24
JNB	bit,rel	Jump if direct Bit is Not set	3	24
JBC	bit,rel	Jump if direct Bit is set & clear bit	3	24
PROGRAM BRANCHING				
ACALL	addr11	Absolute Subroutine Call	2	24
LCALL	addr16	Long Subroutine Call	3	24
RET		Return from Subroutine	1	24
RETI		Return from interrupt	1	24
AJMP	addr11	Absolute Jump	2	24
LJMP	addr16	Long Jump	3	24
SJMP	rel	Short Jump (relative addr)	2	24

Table 10. 8051 Instruction Set Summary (Continued)

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

Timer/Counter

ไมโครคอนโทรลเลอร์ตระกูล MCS-51 จะมีรีจิสเตอร์ที่ทำหน้าที่เป็น Timer/Counter ขนาด 16 บิท จำนวน 2 ตัว คือ Timer 0 และ Timer 1 แต่ถ้าเป็นเบอร์ 8052 จะมี Timer/Counter เพิ่มมาให้อีก 1 ตัว คือ Timer 2 โดย Timer/Counter ทั้งสามตัวเป็นรีจิสเตอร์ที่อยู่ในหน่วยความจำบริเวณ SFR ซึ่งผู้ใช้สามารถกำหนดการทำงานให้เป็น Timer หรือ Counterได้อย่างใดอย่างหนึ่ง โดยมีรายละเอียดในแต่ละอย่างดังนี้

- <u>ในโหมด Timer</u> ค่าของรีจิสเตอร์จะถูกเพิ่มค่าทุกๆ machine cycle ดังนั้นจึงสามารถคิดว่าการทำงานในโหมดนี้เป็นการนับ machine cycle ก็ได้ และเนื่องจากใน 1 machine cycle ของ MCS-51 ประกอบไปด้วย 12 oscillator period ดังนั้นอัตราเร็วในการนับ (Count Rate) จึงมีค่าเป็น 1/12 ของความถี่ออสซิสเลเตอร์ที่ใช้ในระบบ

- <u>ในโหมด Counter</u> ค่าในรีจิสเตอร์จะถูกเพิ่มค่าครั้งละหนึ่งตามการเปลี่ยนสถานะที่ขา T0,T1 หรือ T2 (ใน 8052) โดยการเปลี่ยนสถานะต้องเปลี่ยนค่าจาก 1 เป็น 0 (1 to 0 Transition)

MCS-51 จะตรวจสอบสถานะของลอจิกที่ขา T0,T1 โดยตรวจสอบ (Sampled) ระหว่าง S5P2 ของแต่ละ machine cycle รายละเอียดในการตรวจสอบการเปลี่ยนสถานะลอจิกที่แต่ละขาจะมีลักษณะที่เหมือนกันดังนี้

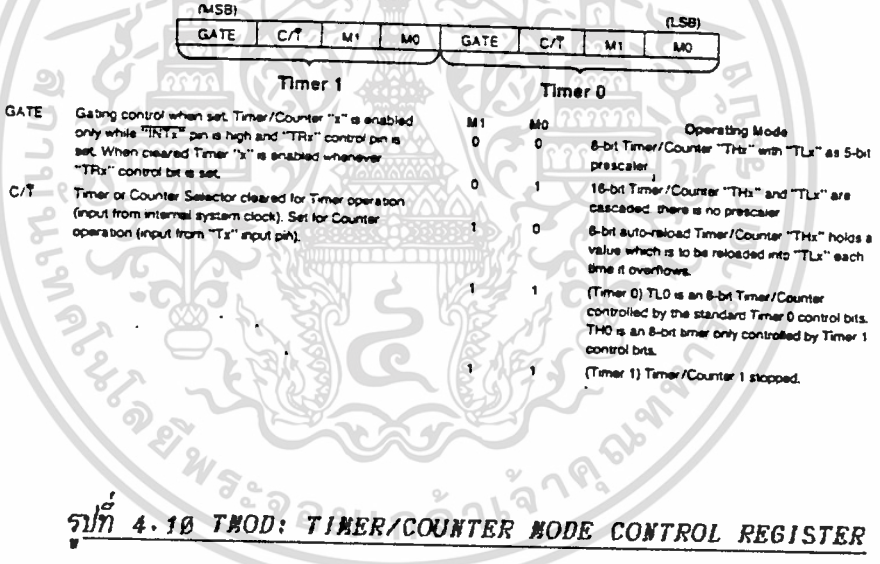
เมื่อสถานะลอจิกที่ขา T0,T1 หรือ T2 มีค่าเป็น 1 (high) ใน S5P2 ของ cycle ใด และใน cycle ถัดไปที่ S5P2 เช่นกัน หากสถานะลอจิกที่ขา T0,T1 หรือ T2 มีค่าเป็น 0 (Low) จะส่งผลให้ค่าที่อยู่ในรีจิสเตอร์ (Timer/Counter Register) ถูกเพิ่มขึ้นอีก 1 ในช่วง S3P1 ของ cycle ที่ถัดจาก cycle ซึ่งตรวจพบการเปลี่ยนสถานะของลอจิก ดังนั้น MCS-51 จำเป็นจะต้องใช้ machine cycle จำนวน 2 cycle (24 Oscillator Period) เพื่อตรวจสอบการเปลี่ยนสถานะลอจิกจาก 0 เป็น 1 ที่ขา T0,T1 หรือ T2 จึงทำให้อัตราการนับสูงสุดของ Counter มีค่าเท่ากับ 1/24 ของความถี่ออสซิลเลเตอร์ที่ใช้ในระบบ โดยไม่มีข้อจำกัดในเรื่อง duty cycle (อัตราส่วนของช่วงเวลาที่สัญญาณมีสถานะลอจิกเป็น 1 ต่อคาบเวลาของสัญญาณ) ของสถานะของสัญญาณแต่เพื่อให้มั่นใจว่าสถานะลอจิกจะถูกตรวจสอบ (sampled) เข้ามาอย่างน้อย 1 ครั้ง ก่อนที่มันจะเปลี่ยนระดับ จึงควรให้สถานะลอจิกของสัญญาณคงค่า 0 หรือ 1 ไว้อย่างน้อย 1 machine cycle เต็ม

นอกจากผู้ใช้จะสามารถเลือกการทำงานของรีจิสเตอร์ให้เป็น Timer หรือ Counter ได้แล้ว MCS-51 ยังอนุญาติให้ผู้ใช้กำหนดให้ Timer หรือ Counter มีการทำงานที่พิเศษแยกย่อยลงไปอีกถึง 4

แบบ (โหมด 0,1,2,3) ตามความเหมาะสมของการใช้งาน (T0,T1 มีให้เลือก 4 แบบ แต่ T2 มีให้เลือก 3 แบบ) ดังจะได้กล่าวต่อไป

TIMER 0 และ TIMER 1

Timer/Counter 2 ตัวนี้มีอยู่ทั้งในไมโครคอนโทรลเลอร์ MCS-51 ทุกเบอร์ ผู้ใช้สามารถที่จะเลือกการทำงานให้เป็น Timer หรือ Counter ได้โดยการเปลี่ยนค่าบิทในรีจิสเตอร์ TMOD โดย TIMER0 จะใช้บิท 2 ส่วน TIMER1 ใช้บิท 6 เป็นตัวเลือก (บิท C/T* ในรีจิสเตอร์ TMOD) ดังแสดงในรูปที่ 4.10 โดยหากบิทนี้มีค่าเป็น 0 หมายถึงเลือกให้เป็น Timer (นับจำนวน machine cycle) ถ้าบิทนี้มีค่าเป็น 1 หมายถึงเลือกให้เป็น Counter (นับจำนวนการเปลี่ยนสถานะจาก 1 เป็น 0 ที่ขา T0 T1)



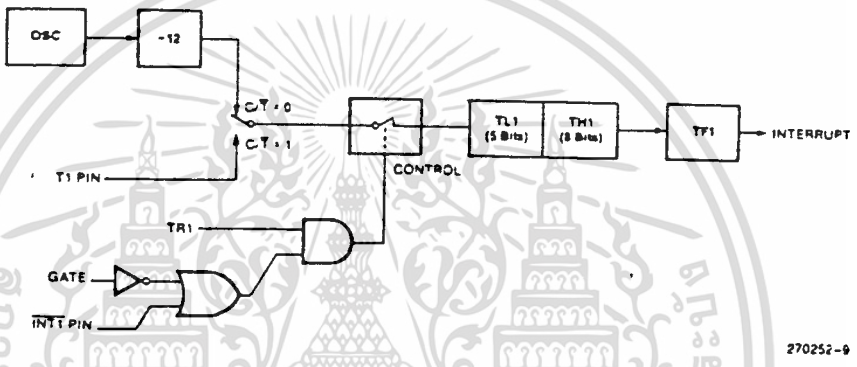
รูปที่ 4.10 TMOD: TIMER/COUNTER MODE CONTROL REGISTER

Timer/Counter ทั้ง 2 ตัวสามารถทำงานแตกต่างกันออกไป 4 แบบ ทั้งนี้โดยการเลือกด้วยค่าของบิท M0,M1 ในรีจิสเตอร์ TMOD เช่นเดียวกับบิท C/T* ดังแสดงในภาพที่ 1 การทำงานในโหมด 0 1,2 จะคล้ายๆ กันสำหรับ Timer/Counter ทั้ง 2 แต่ในโหมด 3 ของ TIMER ทั้งสอง จะมีการทำงานที่ต่างออกไปจาก 3 โหมดแรก รายละเอียดการทำงานทั้ง 4 แบบของ TIMER ทั้งสองมีดังนี้

MODE 0

แต่ละ Timer/Counter ในโหมดนี้จะถูกกำหนดการทำงานให้เป็น Counter ขนาด 8 บิท ซึ่งจะถูกเพิ่มค่าต่อเมื่อนับสัญญาณจาก machine cycle หรือ ตรวจพบการเปลี่ยนสถานะลอจิกที่ขา T0,T1 ครบ 32 ครั้งแล้ว (เป็น Counter ขนาด 8 บิทที่ถูกหารด้วย 32) กล่าวอีกอย่างหนึ่งคือ Timer แต่

ละตัวจะทำงานเป็น counter เพื่อนับ machine cycle หรือการเปลี่ยนสถานะลอจิกที่ขา T0 , T1 ขนาด 13 บิทนั่นเอง โดยการนับสัญญาณ 32 ครั้งแรก จะใช้รีจิสเตอร์ TLx จำนวน 5 บิท ($2^5 = 32$) เมื่อรีจิสเตอร์ TLx นับครบ 32 บิทแล้ว จึงจะมีผลให้มีการเพิ่มค่าในรีจิสเตอร์ THx ซึ่งมีขนาด 8 บิท ทำให้ TIMER ทั้งสองทำงานเป็น Counter (นับจำนวน machine cycle หรือนับจำนวนครั้งการเปลี่ยนสถานะที่ขา T0,T1 ขึ้นกับบิท C/T* ในรีจิสเตอร์ TMOD) ขนาด 13 บิท โดยมีการทำงานดังแสดงในรูป 4.11



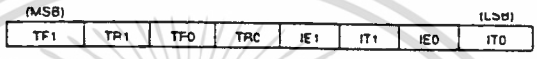
รูปที่ 4.11 Timer/Counter 1 Mode 0 : 13 Bits Counter

จากรูปจะเห็นว่ารีจิสเตอร์ TL1 และ TH1 จะมาประกอบกันเข้าเป็น counter ขนาด 13 บิท เมื่อค่าในรีจิสเตอร์ทั้งสองเปลี่ยนจากเป็น 1 ทั้งหมดไปเป็น 0 ทั้งหมด จะมีผลไปเซ็ทบิท Timer Interupt Flag TF1 ซึ่งก่อให้เกิดอินเทอร์รัพท์ของ TIMER ขึ้น (ควบคุมได้ด้วยรีจิสเตอร์ IE) อินพุทที่ใช้นับถูก enable ให้กับ Timer *เมื่อบิท TR1 มีค่าเป็น 1* (ในรีจิสเตอร์ TCON ดังแสดงในภาพที่ 2) และ *Gate = 0 หรือ สถานะที่ขา INT1* เป็น 1 อย่างใดอย่างหนึ่ง* การควบคุมด้วยบิททั้งสองมีดังแสดงในภาพที่ 4.11

การเซ็ทให้ Gate เป็น 1 จะยอมให้ Timer ถูกควบคุมโดยสถานะที่ขา INT1* ทำให้การนำ MCS-51 ไปใช้วัดความกว้างของพัลส์ทำได้ง่าย ส่วนการเคลียร์ให้ Gate เป็น 0 จะมีผลให้การควบคุมอินพุทที่ใช้นับ TIMER ทั้งสองกระทำได้โดยบิท TR1

บิท TR0,TR1 จะอยู่ในรีจิสเตอร์ TCON ดังแสดงในภาพที่ 2 ส่วนบิท Gate อยู่ในรีจิสเตอร์ TMOD ดังแสดงในภาพที่ 1

รีจิสเตอร์ขนาด 13 บิท ประกอบด้วย 8 บิทของรีจิสเตอร์ TH1 และ 5 บิทของรีจิสเตอร์ TL1 ส่วน 3 บิทบนของรีจิสเตอร์ TL1 ไม่ถูกกำหนดและไม่ถูกใช้ การเซ็ทค่าของบิท Run Flag (TR1) ก็ไม่ได้เคลียร์ค่าในรีจิสเตอร์ทั้งสอง



Symbol	Position	Name and Significance
TF1	TCON.7	Timer 1 overflow Flag. Set by hardware on Timer/Counter overflow. Cleared by hardware when processor vectors to interrupt routine.
TR1	TCON.6	Timer 1 Run control bit. Set/cleared by software to turn Timer/Counter on/off.
TF0	TCON.5	Timer 0 overflow Flag. Set by hardware on Timer/Counter overflow. Cleared by hardware when processor vectors to interrupt routine.
TR0	TCON.4	Timer 0 Run control bit. Set/cleared by software to turn Timer/Counter on/off.

Symbol	Position	Name and Significance
IE1	TCON.3	Interrupt 1 Edge flag. Set by hardware when external interrupt edge detected. Cleared when interrupt processed.
IT1	TCON.2	Interrupt 1 Type control bit. Set/cleared by software to specify falling edge/low level triggered external interrupts.
IE0	TCON.1	Interrupt 0 Edge flag. Set by hardware when external interrupt edge detected. Cleared when interrupt processed.
IT0	TCON.0	Interrupt 0 Type control bit. Set/cleared by software to specify falling edge/low level triggered external interrupts.

รูปที่ 4.12 TCON : Timer/Counter Control Register

MODE 1

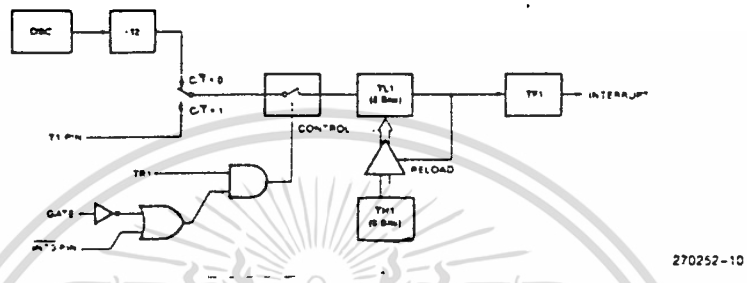
การทำงานในโหมด 1 นี้มีการทำงานเหมือนในโหมด 0 ทุกประการ เว้นแต่ค่าในรีจิสเตอร์ Timer จะถูกใช้งานครบทั้ง 16 บิทเลย นั่นคือในโหมดนี้ Timer/Counter จะเป็น counter หรือ Timer (ขึ้นกับบิท C/T* ในรีจิสเตอร์ TMOD เช่นกัน) ขนาด 16 บิทแทน

MODE 2

การทำงานในโหมด 2 จะกำหนดให้รีจิสเตอร์ Timer/Counter ทำงานเป็น counter ขนาด 8-bits (ใช้รีจิสเตอร์ TLx) ซึ่งจะทำการโหลดค่าเองโดยอัตโนมัติด้วยค่าในรีจิสเตอร์ THx เมื่อเกิด Overflow ในรีจิสเตอร์ TLx ค่าในรีจิสเตอร์ THx นี้สามารถกำหนดได้ล่วงหน้าโดยซอฟท์แวร์ และจะ

ไม่เปลี่ยนแปลงเมื่อถูกโหลดไปไว้ในรีจิสเตอร์ TL1 การทำงานของโหมด 2 มีดังแสดงในภาพที่ 4.13

การทำงานโหมดนี้ มีไว้เพื่อใช้ในการกำเนิดสัญญาณอินเทอร์รัพท์เป็นจังหวะไปเรื่อยๆ หรือใช้เป็นตัวกำหนดเวลาการอินเทอร์รัพท์ที่คงที่ได้



รูปที่ 4.13 Timer/Counter 1 Mode 2 : 8 bits Auto-Reload

MODE 3

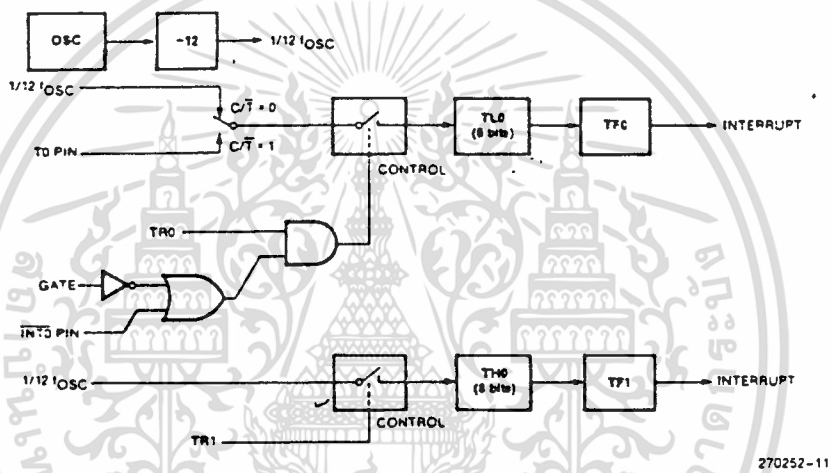
ในการทำงานของ Timer/Counter โหมด 3 นี้ Timer 1 จะไม่มีการนับ ซึ่งมีผลเหมือนกับให้ค่า TR1 = 0 แต่ Timer 0 จะมีการทำงานดังต่อไปนี้

Timer 0 ในโหมด 3 จะทำให้รีจิสเตอร์ TL0 และ TH0 ทำงานเหมือนเป็น Counter ขนาด 8 บิท แยกต่างหากจากกัน 2 ตัว สำหรับการทำงานในโหมด 3 ของ Timer 0 แสดงในรูปที่ 5 ซึ่งจะเห็นว่ารีจิสเตอร์ TL0 ใช้การควบคุมจากบิท C/T*, Gate, TR0, INT0* และ TF0 ส่วนรีจิสเตอร์ TH0 จะถูกบังคับให้เป็น Counter ที่นับจำนวน machine cycle และถูกควบคุมการทำงานโดยบิท TR1 และ TF1 ของ Timer 1 ดังนั้นขณะนี้รีจิสเตอร์ TH0 จะควบคุมการอินเทอร์รัพท์ของ Timer 1 ดังแสดงในภาพ 4.14

MODE 3 นี้มีไว้เพื่อการใช้งานที่ต้องการ Timer หรือ Counter 8 บิทเพิ่มขึ้นดังนี้

เมื่อใช้ Timer 0 ในโหมด 3 8051 สามารถมองเหมือนว่ามันมี Timer/Counter 3 ตัว และ 8052 มี Timer/Counter 4 ตัว โดยเมื่อ Timer 0 กำลังทำงานอยู่ในโหมด 3 Timer 1 ยังสามารถใช้นับได้ โดยการควบคุมสามารถทำได้ด้วยการบังคับให้ Timer 1 สวิทช์ไปมาระหว่างโหมด 3 และโหมดอื่น (หาก Timer 1 อยู่ในโหมด 3 จะหยุดการนับ หากอยู่ในโหมดอื่นจะนับต่อไปเรื่อยๆ) ดัง

นั้นจึงเปรียบเสมือนว่ามี Timer/Counter ใช้มากขึ้นอีก 1 ตัวจาก Timer 1 (เพิ่มจากรีจิสเตอร์ TL0 และ TH0) โดยทั่วไปจะใช้ Timer 1 เป็นตัวกำหนด Baud Rate แต่จริงๆ แล้ว Timer 1 สามารถถูกใช้ในงานใดๆ ก็ได้ที่ไม่ต้องการการอินเทอร์รัพท์ ทั้งนี้เพราะการอินเทอร์รัพท์ของ Timer 1 ถูกใช้โดย Timer 0 ไปแล้วนั่นเอง



รูปที่ 4.14 TIMER/COUNTER 0 MODE3: TWO 8BIT COUNTER

Timer 2

ความจริง Timer 2 ไม่มีใน 8051 แต่มีใน 8052 ซึ่งเป็นเบอร์ที่นิยมใช้งานกันพอสมควร จึงขอกล่าวเกี่ยวกับ Timer 2 ในเบอร์ 8052 ไว้ด้วยดังนี้

Timer 2 เป็น Timer/Counter ขนาด 16 บิท ซึ่งมีเฉพาะใน 8052 โดยมันสามารถทำงานเป็น Timer หรือ Counter อย่างใดอย่างหนึ่งเหมือน Timer 0 และ Timer 1 ผู้ใช้สามารถเลือกการทำงานได้ด้วยบิท C/T*2 ในรีจิสเตอร์ T2CON (รูปที่ 4.15) ใน Timer 2 นี้มีการทำงานอยู่ 3 โ
ด้วยกัน คือ Capture , Auto-Reload และ Baud Rate Generator ซึ่งสามารถเลือกได้โดยกำหนดค่าบิทในรีจิสเตอร์ T2CON ดังแสดงในตารางที่ 3

(MSB)							(LSB)
TF2	EXF2	RCLK	TCLK	EXEN2	TR2	C/T2	CP/RL2

Symbol	Position	Name and Significance
TF2	T2CON.7	Timer 2 overflow flag set by a Timer 2 overflow and must be cleared by software. TF2 will not be set when either RCLK = 1 or TCLK = 1.
EXF2	T2CON.6	Timer 2 external flag set when either a capture or reload is caused by a negative transition on T2EX and EXEN2 = 1. When Timer 2 interrupt is enabled, EXF2 = 1 will cause the CPU to vector to the Timer 2 interrupt routine. EXF2 must be cleared by software.
RCLK	T2CON.5	Receive clock flag. When set, causes the serial port to use Timer 2 overflow pulses for its receive clock in Modes 1 and 3. RCLK = 0 causes Timer 1 overflow to be used for the receive clock.
TCLK	T2CON.4	Transmit clock flag. When set, causes the serial port to use Timer 2 overflow pulses for its transmit clock in modes 1 and 3. TCLK = 0 causes Timer 1 overflows to be used for the transmit clock.
EXEN2	T2CON.3	Timer 2 external enable flag. When set, allows a capture or reload to occur as a result of a negative transition on T2EX if Timer 2 is not being used to clock the serial port. EXEN2 = 0 causes Timer 2 to ignore events at T2EX.
TR2	T2CON.2	Start/stop control for Timer 2. A logic 1 starts the timer.
C/T2	T2CON.1	Timer or counter select. (Timer 2) 0 = Internal timer (OSC/12) 1 = External event counter (falling edge triggered).
CP/RL2	T2CON.0	Capture/Reload flag. When set, captures will occur on negative transitions at T2EX if EXEN2 = 1. When cleared, auto-reloads will occur either with Timer 2 overflows or negative transitions at T2EX when EXEN2 = 1. When either RCLK = 1 or TCLK = 1, this bit is ignored and the timer is forced to auto-reload on Timer 2 overflow.

รูปที่ 4.15 T2CON : TIMER/COUNTER 2 CONTROL REGISTER

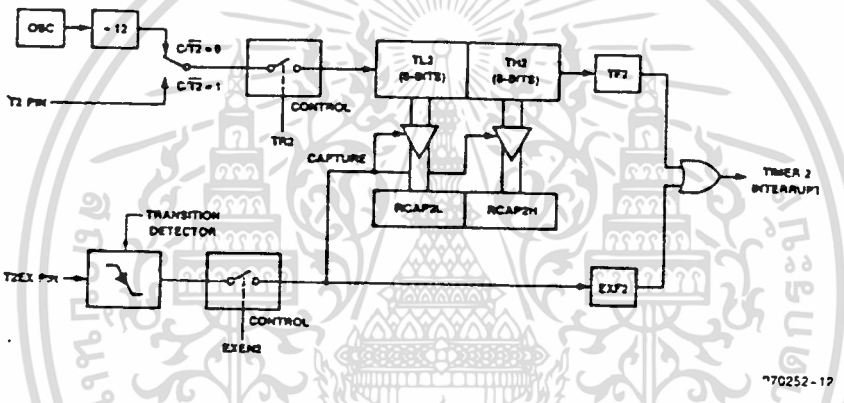
ตารางที่ 3

RCLK + TCLK	CP/RL2	TR2	Mode
0	0	1	16-bit Auto-Reload
0	1	1	16-bit Capture
1	X	1	Baud Rate Generator
X	X	0	(off)

การทำงานแต่ละโหมดของ Timer 2 มีรายละเอียดดังนี้

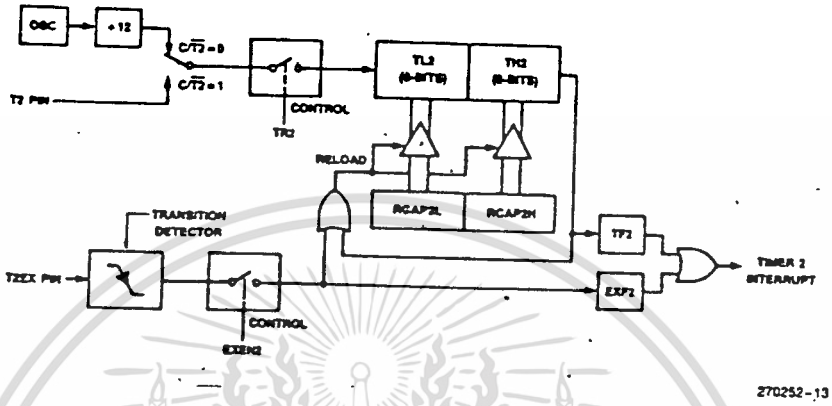
Capture Mode มี 2 Options ให้เลือก สามารถถูกเลือกได้โดยบิท EXEN2 ในรีจิสเตอร์ T2CON ซึ่งถ้าบิท EXEN2 = 0 แล้ว Timer 2 จะเป็น Timer หรือ Counter ขนาด 16 บิท ซึ่งเมื่อเกิด Overflow จะไปเซ็ทบิท TF2 (Timer 2 Overflow Flag) ส่งผลให้เกิดอินเทอร์รัพท์ได้ แต่ถ้าบิท EXEN2 = 1 Timer 2 จะยังคงทำงานเหมือนที่กล่าวมาแล้ว แต่มีคุณสมบัติพิเศษที่เพิ่มเข้ามาคือ สถานะลอจิกที่มีการเปลี่ยนจาก 1 เป็น 0 ที่ขา T2EX จะทำให้ค่าปัจจุบัน (current value)

ใน Timer 2 register (รีจิสเตอร์ TL2,TH2) ถูกโหลด (Captured) ลงไปในรีจิสเตอร์ RCAP2L และ RCAP2H ตามลำดับ (RCAP2L และ RCAP2H เป็นรีจิสเตอร์ในบริเวณ SFR ที่มีเพิ่มขึ้นมาใน 8052) นอกจากนี้ การเปลี่ยนระดับ (Transition) ที่ขา T2EX จะทำให้บิท EXF2 ใน T2CON ถูกเซ็ท และบิท EXF2 สามารถทำให้เกิดอินเทอร์รัพท์ได้เช่นเดียวกับ TF2 (Capture Mode สามารถอธิบายการทำงานได้ดังรูปที่ 4.16)



รูปที่ 4.16 TIMER 2ใน CAPTURE MODE

Auto-Reload Mode มี 2 Options ให้เลือกเช่นเดียวกัน ซึ่งถูกเลือกโดยบิท EXEN2 ในรีจิสเตอร์ T2CON โดยถ้าบิท EXEN2 = 0 แล้ว เมื่อค่าใน Timer 2 เปลี่ยนจากเป็น 1 ทั้งหมดไปเป็น 0 ทั้งหมด (นับครบ 16 บิท) ไม่เพียงแต่จะทำให้บิท TF2 ถูกเซ็ท แต่จะทำให้ค่าในรีจิสเตอร์ของ Timer 2 ถูกโหลดอีกครั้งด้วยค่า 16 บิทในรีจิสเตอร์ RCAP2L และ RCAP2H ซึ่งสามารถตั้งไว้ล่วงหน้าได้ด้วยซอฟท์แวร์ ถ้าบิท EXEN2 = 1 Timer 2 จะยังคงทำงานเหมือนดังที่กล่าวมา แต่มีคุณสมบัติพิเศษเพิ่มขึ้นคือการเปลี่ยนสถานะจาก 1 เป็น 0 ที่ขา T2EX จะไปกระตุ้นให้มีการโหลดค่าจากรีจิสเตอร์ RCAP2L และ RCAP2H ไปยังรีจิสเตอร์ของ Timer 2 ทันทีและเซ็ทบิท TF2 เช่นกัน กล่าวคือ การโหลดค่าจากรีจิสเตอร์ RCAP2L และ RCAP2H ไปยังรีจิสเตอร์ของ Timer 2 จะเกิดขึ้นได้โดยไม่ต้องรอให้ Timer 2 เกิด Overflow นั่นเอง (ดูรูปอธิบายในรูปที่ 4.17)



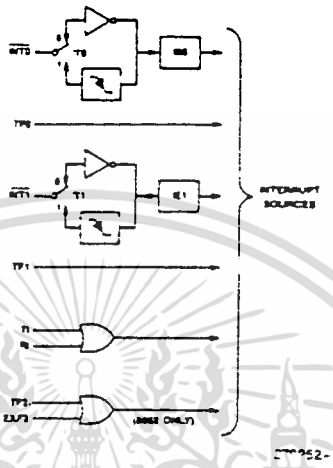
รูปที่ 4.17 *TIMER2 IN AUTO-RELOAD MODE*

Baud Rate Generator Mode

ถูกเลือกโดยการให้บิท RCLK = 1 และ/หรือ TCLK = 1 ซึ่งจะอธิบายร่วมกับเรื่อง Serial Port ต่อไป

INTERRUPTS

8051 มี Interrupt Source ให้ 5 interrupt ด้วยกัน ส่วนใน 8052 มีให้ 6 interrupt source ดังแสดงในรูปที่ 4.18 โดย interrupt แต่ละตัวมีรายละเอียดดังนี้



รูปที่ 4.18 MCS-51 Interrupt Sources

External Interrupts INT0* และ INT1* เป็นอินเทอร์รัพท์ที่เกิดขึ้นภายนอก แต่ละตัวสามารถเลือกให้เป็น Level-Activated หรือ Transition-Activated อย่างใดอย่างหนึ่งขึ้นอยู่กับบิท IT0 และ IT1 ในรีจิสเตอร์ TCON

Flag ที่ทำให้เกิดการอินเทอร์รัพท์ในซีพียูจริงๆ แล้วคือค่าในบิท IE0 และ IE1 ของรีจิสเตอร์ TCON โดยเมื่อเกิดอินเทอร์รัพท์จากภายนอกขึ้น Flag ซึ่งทำให้เกิดอินเทอร์รัพท์จะถูกเคลียร์โดยฮาร์ดแวร์เมื่อซีพียูถูกย้ายไปทำงานที่ interrupt service routine ก็ต่อเมื่ออินเทอร์รัพท์ตัวนั้นเป็นชนิด transition-activated แต่ถ้าอินเทอร์รัพท์ตัวนั้นเป็นชนิด level-activated แล้ว เมื่อซีพียูถูกย้ายไปทำงานที่ interrupt service routine จะไม่เคลียร์บิท IEx ให้ วงจรภายนอกที่ขออินเทอร์รัพท์ซีพียูจะต้องทำหน้าที่ควบคุมสถานะของสัญญาณที่ขา INTx* ให้กลับสู่สภาพเดิมเอง

Timer 0 และ Timer 1 Interrupts ถูกทำให้เกิดอินเทอร์รัพท์ได้โดยบิท TF0 และ TF1 ซึ่งถูกเซ็ทโดยการเปลี่ยนค่าจากเป็น 1 ทั้งหมดเป็น 0 ทั้งหมดใน Timer/Counter Register ของแต่ละตัว (ยกเว้น Timer 0 ในโหมด 3) เมื่อ Timer Interrupt เกิดขึ้น Flag ที่ทำให้เกิดอินเทอร์รัพท์จะถูกเคลียร์โดยฮาร์ดแวร์บนชิพเมื่อซีพียูถูกย้ายไปทำงานในส่วน Service Routine

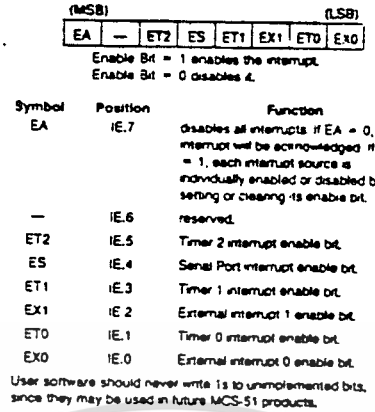
Serial Port Interrupt สามารถสร้างสัญญาณอินเทอร์รัพท์ได้ โดยสัญญาณที่จะทำให้เกิดอินเทอร์รัพท์ได้มาจากบิท TI หรือ RI โดยนำมาผ่านเกทแบบ OR และ Flag ทั้งสองตัวนี้จะไม่ถูกเคลียร์โดยฮาร์ดแวร์เมื่อซีพียูไปทำงานในส่วน Service Routine เพราะจริงๆ แล้วใน Service Routine จะต้องหาว่า RI หรือ TI เป็นตัวทำให้เกิดอินเทอร์รัพท์ขึ้น และบิทนั้นจะถูกเคลียร์ได้โดยซอฟท์แวร์เท่านั้น

ใน 8052 จะมี Timer 2 เพิ่มขึ้นมาอีก 1 ตัว ซึ่ง Timer 2 สามารถทำให้เกิดการอินเทอร์รัฟท์ได้ โดยนำบิท TF2 และ EXF2 มาผ่านเกท OR โดย Flag ทั้งสองตัวนี้จะไม่ถูกเคลียร์โดยฮาร์ดแวร์ เมื่อซีพียูไปทำงานในส่วน Service Routine เหมือนการเกิดอินเทอร์รัพท์ใน Serial Port โดยใน Service Routine จะต้องหาว่า TF2 หรือ EXF2 เป็นตัวทำให้เกิดการอินเทอร์รัพท์ และบิทที่ทำให้เกิดอินเทอร์รัพท์จะต้องได้รับการเคลียร์โดยซอฟท์แวร์เองด้วย

บิททั้งหมดที่ทำให้เกิดสัญญาณอินเทอร์รัพท์สามารถถูกเซ็ทหรือเคลียร์โดยซอฟท์แวร์ โดยมีผลเหมือนกับว่ามันถูกเซ็ทหรือถูกเคลียร์โดยฮาร์ดแวร์ นั่นคือ อินเทอร์รัพท์ทั้งหมดที่กล่าวมาสามารถ enable ให้เกิดสัญญาณอินเทอร์รัพท์ หรืออินเทอร์รัพท์ที่กำลังรอการบริการจากซีพียูอยู่สามารถถูกยกเลิกได้ด้วยซอฟท์แวร์

Interrupt Source แต่ละชนิดเหล่านี้ สามารถถูก enable เป็นตัว ๆ หรือถูก disable เป็นตัวๆ โดยการเปลี่ยนแปลงบิทใน SFR IE ซึ่งแต่ละบิทมีความหมายดังในรูปที่ 2 และจะเห็นได้ว่าในรีจิสเตอร์ IE จะมีบิทที่เป็น Global Disable Bit (EA) อยู่ด้วย ซึ่งสามารถ disable Interrupts ทั้งหมดได้ในครั้งเดียว

สังเกตุว่าในรูปที่ 4.19 ตำแหน่งบิท IE6 , IE5 ไม่ถูกใช้ใน 8051 เพราะถูกสงวนไว้ใช้ใ Microcontroller MCS-51 ตัวอื่น ซึ่งมีคุณสมบัติเพิ่มขึ้น ดังนั้นซอฟท์แวร์ของผู้ใช้ไม่ควรจะมีคำสั่งเขียนค่า 1 ลงไปในบิทตำแหน่งเหล่านี้ เพื่อให้ซอฟท์แวร์นั้นสามารถใช้กับไอซี Microcontroller ตัวใหม่ๆ ในตระกูลนี้ได้นั่นเอง



(MSB) (LSB)

EA	—	ET2	ES	ET1	EX1	ET0	EX0

Enable Bit = 1 enables the interrupt.
Enable Bit = 0 disables it.

Symbol	Position	Function
EA	IE.7	disables all interrupts. If EA = 0, no interrupt will be acknowledged. If EA = 1, each interrupt source is individually enabled or disabled by setting or clearing its enable bit.
—	IE.6	reserved.
ET2	IE.5	Timer 2 interrupt enable bit.
ES	IE.4	Serial Port interrupt enable bit.
ET1	IE.3	Timer 1 interrupt enable bit.
EX1	IE.2	External interrupt 1 enable bit.
ET0	IE.1	Timer 0 interrupt enable bit.
EX0	IE.0	External interrupt 0 enable bit.

User software should never write 1s to unimplemented bits, since they may be used in future MCS-51 products.

รูปที่ 4.19 IE : Interrupt Enable Register

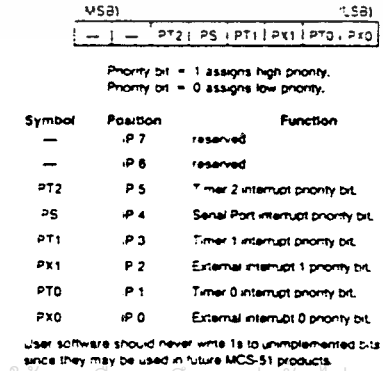
Priority Level Structure

แต่ละ Interrupt Source สามารถถูกเลือกระดับความสำคัญได้ 2 ระดับ โดยการเซ็ทหรือเคลียร์บิทในรีจิสเตอร์ IP ดังในรูปที่ 4.20

ระดับความสำคัญของการขออินเทอร์รัพท์มีได้ 2 ระดับ ได้แก่

Low Priority Interrupt : สามารถถูก Interrupted โดย High Priority Interrupt ได้ แต่ไม่สามารถถูก Interrupted โดย Low Priority Interrupt ตัวอื่นๆ ได้

High Priority Interrupt : ไม่สามารถถูก Interrupted โดย Interrupt Source ตัวอื่นได้เลย



(MSB) (LSB)

—	—	PT2	PS	PT1	PX1	PT0	PX0

Priority bit = 1 assigns high priority.
Priority bit = 0 assigns low priority.

Symbol	Position	Function
—	IP.7	reserved
—	IP.6	reserved
PT2	IP.5	Timer 2 interrupt priority bit.
PS	IP.4	Serial Port interrupt priority bit.
PT1	IP.3	Timer 1 interrupt priority bit.
PX1	IP.2	External interrupt 1 priority bit.
PT0	IP.1	Timer 0 interrupt priority bit.
PX0	IP.0	External interrupt 0 priority bit.

User software should never write 1s to unimplemented bits, since they may be used in future MCS-51 products.

รูปที่ 4.20 IP : Interrupt Priority Register

ถ้ามีการขออินเทอร์รัพท์พร้อมกัน 2 ชนิด ซึ่งมีระดับความสำคัญในการขออินเทอร์รัพท์ไม่เท่ากัน อินเทอร์รัพท์ที่มีระดับความสำคัญสูงกว่า (High Priority Interrupt) จะได้รับการบริการก่อน แต่ถ้ามีการขออินเทอร์รัพท์พร้อมกัน 2 ชนิด ซึ่งมีระดับความสำคัญเท่ากัน ลำดับการรับสัญญาณอินเทอร์รัพท์ภายในจะกำหนดเองว่าอินเทอร์รัพท์ชนิดใดจะถูกบริการก่อน ดังนั้นภายในระดับความสำคัญของการขออินเทอร์รัพท์หนึ่งๆ จะมี Second Priority Structure (ระดับความสำคัญในการขออินเทอร์รัพท์ย่อย) ดังนี้

Source	Priority Within Level
1. IE0	Highest
2. TF0	
3. IE1	
4. TF1	
5. RI+TI	
6. TF2+EXF2	Lowest

สังเกตุว่า Priority Within Level Structure จะถูกใช้เพียงเพื่อแก้ปัญหาการเกิดสัญญาณอินเทอร์รัพท์พร้อมกันขึ้นมาเท่านั้น (Simultaneous Request Of The Same Priority Level)

IP Register มีบิทที่ไม่ถูกใช้งานอยู่จำนวนหนึ่ง คือ IP.7 และ IP.6 โดยว่างทั้งใน 8051 และ 8052 โดยใน 8051 จะมีบิทที่ว่างเพิ่มมาอีก 1 บิทคือ IP.5 และเช่นเดียวกัน ซอฟท์แวร์ของผู้ใช้ไม่ควรมีการเขียนค่า 1 ไปที่ตำแหน่งบิทเหล่านี้ เพราะมันอาจถูกนำไปใช้ใน MCS-51 ตัวใหม่ๆ ในอนาคตต่อไป

MCS-51 จัดการกับสัญญาณอินเทอร์รัพท์อย่างไร

Interrupt Flag ของอินเทอร์รัพทแต่ละชนิดจะถูกตรวจสอบ (Sample) ที่ทุกๆ S5P2 ของทุก machine cycle ค่าที่ตรวจสอบจะถูกรับเข้ามาระหว่าง machine cycle ที่ตามมา โดยใน 8052 Timer 2 Interrupt cycle จะต่างออกไป ดังจะอธิบายในหัวข้อ Response Time

การตอบสนองต่อสัญญาณอินเทอร์รัพท์จะมีแผนผังดังในรูปที่ 4 ซึ่งจะเห็นว่าประกอบไปด้วย cycle

ต่างๆ กัน คือ cycle ตรวจจับอินเทอร์รัพท์ (จะตรวจสถานะของบิทที่อินเทอร์รัพท์ซีพียูทุกๆ S5P2 ของทุก machine cycle) cycle การหาชนิดของอินเทอร์รัพท์ (ตรวจสอบว่าเป็นอินเทอร์รัพท์ชนิดใด) และ cycle การสร้างคำสั่ง LONG CALL ไปยัง interrupt service routine ที่เหมาะสม (ใช้ 2 cycle) และ cycle ของการทำคำสั่งที่อยู่ใน interrupt service routine โดยมีรายละเอียดดังนี้

ถ้า Flag ของอินเทอร์รัพท์ตัวใดตัวหนึ่งอยู่ในสถานะถูกเซ็ทที่ S5P2 ของ cycle ตรวจจับอินเทอร์รัพท์ซึ่งเป็น cycle ก่อนหน้า cycle การหาชนิดของอินเทอร์รัพท์ซึ่งจะทำการค้นหาว่าอินเทอร์รัพท์ชนิดใดเป็นผู้ขอเข้ามา ใน cycle ถัดไปก็จะทำคำสั่ง LONG CALL ไปที่ interrupt service routine ที่เหมาะสม cycle การทำคำสั่ง LONG CALL โดยฮาร์ดแวรจะถูกกระทำสำเร็จก็ต่อเมื่อไม่ถูกป้องกันโดยสภาวะดังต่อไปนี้

1 ซีพียูกำลังทำคำสั่งใน interrupt service routine ของอินเทอร์รัพท์ซึ่งมีความสำคัญเทียบเท่าหรือสูงกว่าอยู่ในขณะนั้น

2 cycle ที่ตรวจหาชนิดของอินเทอร์รัพท์ไม่ใช่ cycle สุดท้ายของคำสั่งที่ซีพียูกำลังปฏิบัติงานอยู่ นั่นคือซีพียูกำลังทำงานของคำสั่งใดๆ ยังไม่เสร็จสิ้นขณะตรวจหาชนิดของอินเทอร์รัพท์พบ

3 คำสั่งที่กำลังปฏิบัติอยู่ในขณะนั้นเป็น RETI หรือคำสั่งใดๆ ที่มีการเขียนข้อมูลไปยังรีจิสเตอร์ IE หรือ IP

สภาวะทั้งสามนี้จะป้องกันมิให้ซีพียูทำคำสั่ง LONG CALL ไปยัง interrupt service routine ขณะตรวจพบการขออินเทอร์รัพท์

สภาวะที่ 1 มีไว้เพื่อให้อินเทอร์รัพท์ซึ่งมีความสำคัญสูงกว่าได้รับการบริการก่อน โดยอินเทอร์รัพท์ที่มีความสำคัญต่ำกว่าไม่สามารถขัดจังหวะได้

สภาวะที่ 2 ทำให้แน่ใจว่าคำสั่งที่กำลังทำอยู่ในขณะนั้นจะถูกทำให้เสร็จก่อนการย้ายไปทำคำสั่งใดๆ ที่ interrupt service routine ที่เกี่ยวข้อง

สภาวะที่ 3 ทำให้แน่ใจว่า ถ้าคำสั่งที่ทำอยู่เป็น RETI หรือการเข้าถึงข้อมูลในรีจิสเตอร์ IE หรือ IP แล้ว อย่างน้อยจะต้องมีคำสั่งอีก 1 คำสั่งถูกปฏิบัติ ก่อนอินเทอร์รัพท์ใดๆ จะถูกบริการ

cycle การตรวจหาชนิดของอินเทอร์รัพท์จะถูกทำซ้ำแต่ละ machine cycle และค่าที่รับเข้ามา (ชนิดของอินเทอร์รัพท์) จะเป็นค่าที่ปรากฏเมื่อ S5P2 ของ machine cycle ก่อนหน้านี้ สังเกตุว่าถ้าบิทที่ทำหน้าที่อินเทอร์รัพท์ถูกเซ็ท (Active) แต่ไม่ถูกตอบสนองเพราะ 1 ใน 3 ของสภาวะ

ข้างต้น และปรากฏว่าไม่ active เมื่อสภาวะที่ป้องกันการทำคำสั่ง LONG CALL หมดไปแล้ว อินเทอร์รัพท์นี้จะไม่ได้รับการบริการ (ทำคำสั่งใน interrupt service routine) หรือกล่าวได้ว่าสถานะของอินเทอร์รัพท์แต่ละตัวซึ่งครั้งหนึ่งเคย active แต่ไม่ได้รับการบริการเพราะถูกป้องกันด้วยสภาวะดังกล่าวมาข้างต้น จะไม่ถูกจำไว้ เพราะแต่ละ cycle ของการตรวจหาชนิดของอินเทอร์รัพท์จะรับสถานะของอินเทอร์รัพท์ใหม่เข้ามาเสมอ

สังเกตุว่าถ้าอินเทอร์รัพท์ที่มีความสำคัญสูงถูก active ก่อน S5P2 ของ machine cycle ที่มีชื่อย่อว่า C3 ในรูปที่ 4.21 และไม่มีสภาวะที่ป้องกันการทำคำสั่ง LONG CALL มันก็จะได้รับการบริการในหว่าง C5 และ C6 โดยปราศจากการบริการของอินเทอร์รัพท์ที่มีความสำคัญน้อยกว่า

ดังนั้น Processor จะรับรู้การอินเทอร์รัพท์โดยการทำคำสั่ง LONG CALL ไปที่ interrupt service routine ที่เหมาะสม ในบางกรณีมันจะเคลียร์บิทที่ก่อให้เกิดอินเทอร์รัพท์ด้วยเลย แต่ในบางกรณีมันจะไม่เคลียร์ให้ ดังนั้นผู้ใช้ต้องบรรจุคำสั่งเคลียร์บิทเหล่านี้เองในซอฟท์แวร์

- external interrupt flag (IE0 และ IE1) จะถูกเคลียร์ต่อเมื่อเป็น transition-activated คำสั่ง LONG CALL ที่กระทำโดยฮาร์ดแวร์จะ PUSH ข้อมูลของ Program Counter ไปไว้ที่ stack (ไม่มีการ save ค่าของรีจิสเตอร์ PSW) และโหลดค่าของ PC ใหม่ด้วยแอดแดรสที่ตรงกับ interrupt service routine ที่เกี่ยวข้องกับสัญญาณอินเทอร์รัพท์ขณะนั้น ดังแสดงข้างล่าง

SOURCE	VECTOR ADDRESS
IE0	0003H
TF0	000BH
IE1	0013H
TF1	001BH
RI+TI	0023H
TF2+EXF2	002BH

การทำคำสั่งใน interrupt service routine นั้นจะทำไปเรื่อยๆ จนกระทั่งพบคำสั่ง RETI ซึ่งเป็นตัวบอกซีพียูว่า interrupt routine สิ้นสุดลงแล้ว จากนั้นซีพียูจะไป POP เอาค่า 2 ไบท์สูงสุดจาก stack และโหลดให้กับ PC เพื่อกลับไปทำงานเดิมที่ทำอยู่ก่อนได้รับอินเทอร์รัพท์

สังเกตุว่าคำสั่ง RET ธรรมดาจะสามารถ POP เอาค่า 2 ไบท์สูงสุดจาก stack และโหลดให้กับ PC ได้เช่นเดียวกับคำสั่ง RETI แต่หากใช้ RET แล้ว interrupt control system จะยังคิดว่า

อินเทอร์รัพท์ยังคงถูกกระทำอยู่

EXTERNAL INTERRUPT

แหล่งกำเนิดอินเทอร์รัพท์จากภายนอกสามารถถูกโปรแกรมให้เป็น level-activated หรือ transition-activated ได้โดยการเซ็ทหรือเคลียร์บิท IT1,IT0 ในรีจิสเตอร์ TCON โดยถ้า ITx มีค่าเป็น 0 external interrupt จะเป็นชนิด level-activated โดยสถานะ low ที่ขา INTx* จะเป็นการอินเทอร์รัพท์ซีพียู ถ้าบิท ITx=1 , external interrupt x จะเป็นชนิด transition-activated (edge-trigged) ในโหมดนี้ถ้าการตรวจสอบที่ขา INTx* พบว่ามีสถานะเป็น high ใน 1 cycle และเป็น low ใน cycle ถัดไป interrupt request flag IEx ในรีจิสเตอร์ TCON จะถูกเซ็ท แล้วบิท IEx นี้จะไปอินเทอร์รัพท์ซีพียูต่อไป

เพราะว่า external interrupt ถูกตรวจสอบ 1 ครั้งในแต่ละ machine cycle และ input high หรือ low ควรจะคงค่าไว้อย่างน้อย 12 period ของความถี่ออสซิลเลเตอร์ เพื่อให้แน่ใจในการตรวจสอบ ถ้า external interrupt เป็น transition-activated แหล่งกำเนิดอินเทอร์รัพท์ภายนอกจะต้องคงค่าเป็น low อย่างน้อย 1 cycle เพื่อให้มั่นใจว่าการเปลี่ยนแปลงถูกตรวจพบอย่างแน่นอน ดังนั้น interrupt request flag จะถูกเซ็ท IEx จะถูกเคลียร์โดยอัตโนมัติ เมื่อซีพียูทำคำสั่ง LONG CALL ไปที่ interrupt service routine

ถ้าา external interrupt เป็น level-activated วงจรที่กำเนิดอินเทอร์รัพท์ภายนอกต้องรักษาสถานะการขออินเทอร์รัพท์ (Active) ไว้จนกระทั่งอินเทอร์รัพท์เกิดขึ้นจริงๆ และมันต้องเปลี่ยนสถานะของการขออินเทอร์รัพท์ให้กลับมีค่าเหมือนเดิม (Deactive) ก่อนที่ interrupt service routine จะถูกกระทำเสร็จ มิฉะนั้นแล้ว interrupt service routine เดียวกันนี้จะถูกทำงานซ้ำอีก

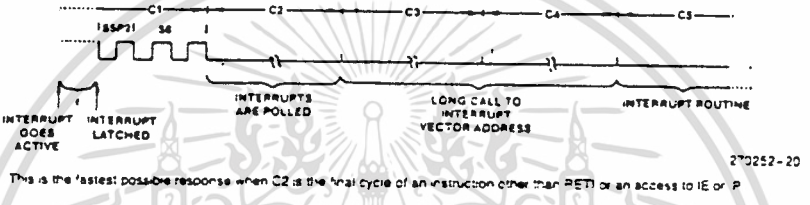
RESPONSE TIME

ระดับของ INT0* และ INT1* จะถูก invert และถูกคงค่าไว้ในบิท IE0 และ IE1 ที่ S5P2 ของแต่ละ machine cycle เช่นเดียวกับ Timer 2 overflow flag และ EXF2 รวมทั้ง Serial Port flag (RI,TI) จริงๆ แล้วสถานะของบิทที่ทำหน้าที่อินเทอร์รัพท์ซีพียูจะยังไม่ถูกรับเข้ามาโดยวงจรภายใน จนกระทั่ง machine cycle ถัดไป

FLAG ของ TIMER 0 และ TIMER 1 (TF0 และ TF1) จะถูกเซ็ทที่ S5P2 ของ cycle ที่ TIMER เกิด overflow แล้วค่า FLAG นี้จะถูกรับโดยวงจรภายในใน cycle ถัดไป แต่ FLAG ของ

TIMER 2 จะถูกเซ็ทที่ S2P2 และถูกรับค่าใน cycle เดียวกันกับ cycle ที่เกิด overflow

ถ้ามีการขออินเทอร์รัพท์และเงื่อนไข 3 ข้อที่กล่าวมาถูกต้องสำหรับ MCS-52 ที่จะรับรู้ว่ามีการขออินเทอร์รัพท์ได้ Hardware subroutine CALL (คำสั่ง CALL ที่ถูกกระทำโดยฮาร์ดแวร์) ซึ่ง CALL ไปที่ interrupt service routine ที่ถูกเรียก จะเป็นคำสั่งถัดไปที่จะถูกกระทำ ตัวคำสั่ง CALL เองจะใช้เวลา 2 machine cycle ดังนั้น การตอบสนองต่ออินเทอร์รัพท์จะต้องใช้เวลาอย่างน้อย 3 machine cycle เต็มๆ ซึ่งเป็นเวลาระหว่างการ activate ของการเรียกอินเทอร์รัพท์กับการเริ่มต้นทำงานคำสั่งแรกใน interrupt service routine ซึ่งอธิบายได้ดังรูปที่ 4.21



รูปที่ 4.21 INTERUPT RESPONSE TIMING DIAGRAM

ช่วงเวลาในการตอบสนองที่ยาวกว่านี้ อาจเกิดขึ้นได้ หากการเรียกอินเทอร์รัพท์ถูกขัดไว้ด้วยเงื่อนไข 1 ใน 3 ข้อที่กล่าวไว้ข้างต้น ถ้าอินเทอร์รัพท์ที่มีลำดับความสำคัญเท่ากันหรือสูงกว่ากำลังถูกกระทำอยู่ เวลาในการรอคอยที่เพิ่มขึ้นนี้ จะขึ้นอยู่กับลักษณะของ interrupt service routine ที่กำลังถูกกระทำอยู่ ถ้าคำสั่งที่กำลังกระทำอยู่ไม่อยู่ใน cycle สุดท้ายของคำสั่งนั้น เวลาในการรอคอยที่เพิ่มขึ้นจะไม่สามารถมากกว่า 3 cycle ได้ เพราะคำสั่งที่ยาวที่สุดคือ MUL , DIV ใช้เวลาเพียง 4 cycle เท่านั้น และหากคำสั่งที่กำลังกระทำเป็น RETI หรือการเขียนข้อมูลไปยังรีจิสเตอร์ IE หรือ IP เวลารอที่เพิ่มขึ้นนั้น ก็จะไม่มากกว่า 5 cycle (อย่างมากที่สุดอีก 1 cycle เพื่อให้กระทำคำสั่งที่ทำอยู่เสร็จกับอีก 4 cycle ที่ใช้ทำคำสั่งถัดไป ในกรณีที่คำสั่งถัดไปนั้นเป็น MUL หรือ DIV)

ดังนั้น ในระบบที่มีการใช้อินเทอร์รัพท์เพียงชนิดเดียว เวลาในการตอบสนองอินเทอร์รัพท์จะมากกว่า 3 cycle และน้อยกว่า 9 cycle เสมอ

SERIAL INTERFACE

ใน MCS-51 มี Serial Port ซึ่งสามารถรับและส่งข้อมูลแบบอนุกรมได้โดยผู้ใช้ไม่จำเป็นต้องต่อไอซีเพิ่มเข้าไป ทำให้มีความสะดวกในการนำไปประยุกต์ใช้งานที่ต้องมีการติดต่อข้อมูลแบบอนุกรมมาก

Serial Port ที่มีใน MCS-51 สามารถทำงานได้ในแบบ Full Duplex หมายความว่า มันสามารถรับและส่งข้อมูลได้พร้อมๆ กัน โดยในการรับข้อมูลจะมีบัฟเฟอร์ให้ด้วย ทำให้ MCS-51 สามารถรับข้อมูลไบต์ที่สองซึ่งถูกส่งตามเข้ามาหลังไบท์แรกได้ บัฟเฟอร์ที่มีใน MCS-51 นี้จะสามารถรับข้อมูลไบท์ที่สองก่อนที่ไบต์แรกซึ่งรับเข้ามาก่อนจะถูกอ่านจาก Recieve Register ไปเก็บไว้ในหน่วยความจำ (แต่ถ้าไบต์แรกยังไม่ถูกอ่านเมื่อเวลาที่การรับของไบต์ที่สองสิ้นสุดลง หนึ่งไบต์ในสองไบต์จะสูญหายไป)

Serial Port จริงๆ แล้วประกอบไปด้วยรีจิสเตอร์ขนาด 8 บิทจำนวนสองตัว แต่ละตัวมีชื่อเรียกตามหน้าที่ดังนี้คือ Recieve Register สำหรับการรับข้อมูล และ Transmit Register สำหรับการส่งข้อมูล ซึ่งรีจิสเตอร์ทั้งสองมีตำแหน่งเดียวกันใน SFR คือตำแหน่งของรีจิสเตอร์ SBUF (99H) โดยการเข้าถึงข้อมูลของรีจิสเตอร์แต่ละตัว MCS-51 จะรู้เองว่าผู้ใช้ต้องการติดต่อกับรีจิสเตอร์ตัวใด เพราะในการเขียนข้อมูลไปที่รีจิสเตอร์ SBUF จะหมายถึงการโหลดค่าไปยัง Transmit Register ส่วนการอ่านข้อมูลในรีจิสเตอร์ SBUF หมายถึงการรับข้อมูลจาก Receive Register

การใช้งาน Serial Port ใน MCS-51 มีความสะดวกและคล่องตัวสูง ทั้งนี้เนื่องจากผู้ใช้สามารถกำหนดการทำงานในโหมดที่แตกต่างกันได้ถึง 4 ประเภท โดยการกำหนดค่าให้กับรีจิสเตอร์ SMOD ที่แตกต่างกัน เพื่อความเหมาะสมกับงานแต่ละงานดังนี้

MODE 0 : การทำงานในโหมด 0 ข้อมูลแบบอนุกรมจะถูกรับเข้ามาและส่งออกไปภายนอกผ่านทางขา RXD ทั้งสองกรณี ส่วนขา TXD มีไว้กำเนิดสัญญาณ Shift Clock เพื่อเป็นตัวกำหนดจังหวะในการรับและส่งข้อมูล ในโหมดนี้จะทำการรับส่งข้อมูลแบบ 8 บิท (8 Data Bits) โดยรับและส่งบิทต่ำสุดก่อน (LSB First) ส่วนอัตราการส่งข้อมูล (BAUD Rate) ถูกกำหนดไว้ที่ 1/12 ของความถี่ออสซิลเลเตอร์ที่ใช้ในระบบ ในการทำงานโหมด 0 นี้จะไม่มี start bit และ stop bit

MODE 1 : ในการทำงานของโหมด 1 นี้ มีการรับและส่งข้อมูลครั้งละ 10 บิท โดยการส่งข้อมูลจะส่งผ่านทางขา TXD ส่วนการรับข้อมูลจะรับทางขา RXD ข้อมูลทั้ง 10 บิทนั้นประกอบไปด้วย 1 start bit (มีค่าเป็น 0) 8 data bits (รับและส่งบิทต่ำก่อน) และ 1 stop bit (มีค่าเป็น 1) โดยขณะทำการรับ ค่าของ stop bitจะไปอยู่ในบิท RB8 ของ SFR SCON ค่า BAUD Rate ของการรับและส่งข้อมูลในโหมดนี้สามารถเปลี่ยนแปลงได้ ดังจะได้กล่าวในรายละเอียดต่อไป

MODE 2 : ในการทำงานโหมด 2 มีการรับและส่งข้อมูลครั้งละ 11 บิท โดยข้อมูลถูกส่งผ่านทางขา TXD และรับเข้ามาผ่านทางขา RXD ข้อมูลทั้ง 11 บิทที่รับและส่งนั้นประกอบด้วย 1 start bit (เป็น 0) 8 data bits (รับหรือส่งบิทต่ำก่อน) ส่วนบิทที่ 9 เป็นบิทที่สามารถโปรแกรมให้มีค่าเป็นศูนย์หรือหนึ่งก็ได้ (Programmable 9th Data bit) และบิทสุดท้ายคือ stop bit (เป็น 1)

ในขณะทำการส่งข้อมูล บิทข้อมูลบิทที่ 9 (the 9th data bit) ซึ่งจะเป็นบิท TB8 ใน SFR SCON สามารถถูกกำหนดให้เป็น 0 หรือ 1 โดยส่วนใหญ่จะใช้เป็นบิทสำหรับไว้ตรวจสอบข้อมูลที่รับหรือส่ง (parity bit) ในการใช้งานจริงๆ จะนำเอาบิท P ในรีจิสเตอร์ PSW) ไปไว้ในบิท TB8 ส่วนในขณะรับข้อมูล บิทข้อมูลบิทที่ 9 (the 9th data bit) จะไปอยู่ในบิท RB8 ของรีจิสเตอร์ SCON โดยไม่สนใจ stop bit ค่า Baud Rate ในโหมดนี้สามารถตั้งให้เป็น 1/32 หรือ 1/64 ของความถี่ออสซิลเลเตอร์ที่ใช้ในระบบ

MODE 3 : ข้อมูลจำนวน 11 บิทถูกส่งผ่านขา TXD และถูกรับผ่านขา RXD โดยทั้ง 11 บิทจะประกอบไปด้วย 1startbit (0) 8 data bits (LSB First) ส่วนบิทที่ 9 จะเป็นบิทที่โปรแกรมได้เหมือนในโหมด 2 (Programmable 9th bit) และบิทสุดท้ายคือ 1 stop bit (1) ส่วนค่า Baud Rate สามารถเปลี่ยนแปลงได้ ดังจะศึกษาในรายละเอียดต่อไป ซึ่งจะเห็นว่าการรับส่งข้อมูลในโหมด 3 นี้จะเหมือนกับโหมด 2 ทุกอย่าง ยกเว้นค่า Baud Rate ที่เปลี่ยนแปลงได้ของโหมดนี้เท่านั้น

การทำงานของ Serial Port ทั้ง 4 โหมดที่กล่าวมานี้ การส่งข้อมูลจะเริ่มขึ้นต้นเมื่อมีคำสั่งที่ใช้รีจิสเตอร์ SBUF เป็นรีจิสเตอร์ปลายทาง (Destination Register) ส่วนการรับข้อมูลจะเริ่มต้นโดยมีเงื่อนไขดังนี้

- ในโหมด 0 บิท RI = 0 และ REN = 1
- ในโหมดอื่นๆ การรับข้อมูลเริ่มเมื่อ MCS-51ได้รับ start bit เข้ามา ถ้า REN = 1

MULTIPROCESSOR COMMUNICATIONS

การทำงานของ Serial Port ในโหมด 2 และ 3 มีรูปแบบการใช้งานพิเศษนอกเหนือจากการรับส่งข้อมูลธรรมดาตามที่ได้กล่าวมาแล้ว การใช้งานพิเศษที่ว่านี้คือ การติดต่อสื่อสารระหว่างซีพียูด้วยกันเอง (Multiprocessor communications)

การทำงานแบบ MULTIPROCESSOR จะใช้เมื่อซีพียูในระบบมีจำนวนหลายตัว โดยในการทำงาน

แบบนี้ซีพียูหลักจะสามารถทำการติดต่อกันระหว่างซีพียูแต่ละตัวได้โดยตรง และสามารถเลือกการติดต่อกับตัวใดก็ได้ การทำงานแบบนี้ บิทที่ 9 ที่รับเข้ามาจะถูกนำไปไว้ในบิท RB8 จากนั้น stop bit ก็จะถูกรับตามเข้ามาเหมือนการรับส่งข้อมูลตามที่กล่าวมาแล้ว แต่เราสามารถโปรแกรม Serial Port เพื่อที่เมื่อ stop bit ถูกรับเข้ามาแล้ว จะมีผลไปกระตุ้นให้อาร์ดแวร์ในส่วน Serial Port Interrupt ทำงาน เพื่ออินเทอร์รัพท์ซีพียูก็ต่อเมื่อบิท RB8 = 1 ซึ่งเราสามารถโปรแกรมให้ Serial Port ของ MCS-51 ให้ทำงานเช่นนี้ได้โดยการเซ็ทบิท SM2 ในรีจิสเตอร์ SCON รายละเอียดและแนวทางในการใช้งาน Serial Port แบบ Multiprocessor มีดังนี้

เมื่อซีพียูตัวแม่หรือหน่วยประมวลผลหลัก (Master Processor) ต้องการส่งข้อมูลจำนวนหนึ่งไปยังหน่วยประมวลผลย่อย (Slave) ตัวหนึ่งจากที่มีหลายตัวในระบบ ในขั้นแรก หน่วยประมวลผลหลักจะต้องส่งข้อมูลขนาดขนาด 1 ไบท์ ซึ่งจะเป็นค่าที่บอกตำแหน่ง (Address Byte) ของหน่วยประมวลผลเป้าหมายที่หน่วยประมวลผลหลักต้องการติดต่อด้วย ค่า Address byte ที่ส่งไปจะมีข้อแตกต่างจากข้อมูลที่รับส่งกันจริงๆ ระหว่างหน่วยประมวลผลซึ่งมีชื่อเรียกว่า Data byte ตรงที่บิทที่ 9 จะเป็น 1 เมื่อข้อมูลนั้นเป็น Address Byte และจะเป็น 0 เมื่อข้อมูลนั้นเป็น Data Byte

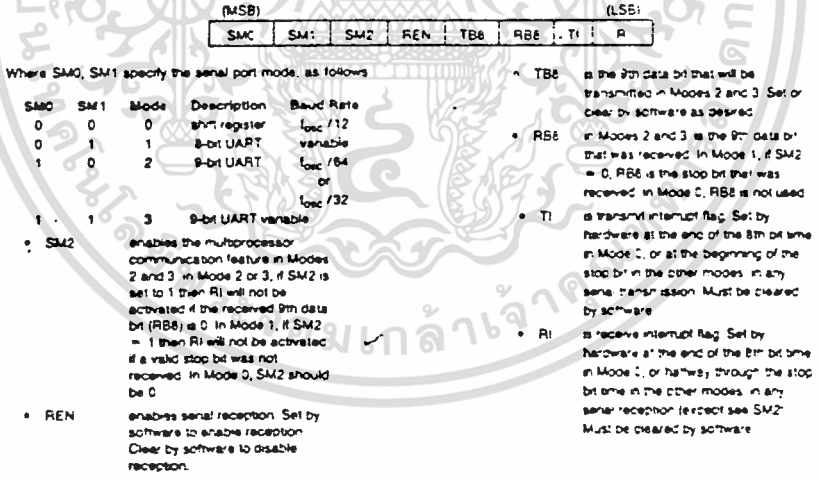
หากในหน่วยประมวลผลย่อยมีการเซ็ทบิท SM2 = 1 แล้ว ถ้าข้อมูลที่รับเข้ามาเป็น Data Byte จะไม่สามารถอินเทอร์รัพท์ซีพียูได้ แต่ถ้าข้อมูลที่ได้รับเป็น Address Byte (บิทที่ 9 มีค่าเป็น 1)หน่วยประมวลผลย่อยทุกตัวจะถูกอินเทอร์รัพท์ การทำเช่นนี้เพื่อที่หน่วยประมวลผลย่อยทุกตัวจะสามารถตรวจสอบได้ว่าข้อมูล Address Byte ที่ได้รับเข้ามามีค่าตรงกับตำแหน่งของตัวมันเองหรือไม่ โดยหน่วยประมวลผลย่อยที่มีค่าตำแหน่งของตัวเองตรงกับข้อมูล Address Byte ที่ได้รับเข้ามา จะทำการเคลียร์บิท SM2 และเตรียมรับข้อมูลที่เป็น Data Bye ซึ่งจะตามเข้าหลังจากรับ Address Byte เรียบร้อยแล้ว ส่วนหน่วยประมวลผลย่อยตัวอื่นที่ตรวจสอบข้อมูลดูแล้วปรากฏว่าไม่ตรงกับแอดแดรสของตัวมันเองจะยังคงปล่อยให้บิท SM2 ของมันถูกเซ็ทต่อไป และกลับไปทำงานที่ค้างอยู่ก่อนได้รับการอินเทอร์รัพท์ต่อ โดยไม่สนใจข้อมูลที่เป็น Data Bye ซึ่งตามเข้ามาหลัง Address Bye

บิท SM2 จะไม่มีผลในการทำงานของ Serial Port ในโหมด 0 แค่การทำงานในโหมด 1 บิท SM2 สามารถถูกใช้เพื่อที่จะตรวจสอบ stop bit (Validity of the stop bit) โดยในการรับข้อมูลของโหมด 1 ถ้าบิท SM2 = 1 การเกิด Receive Interrupt จะยังไม่ทำงานจนกว่า stop bit ที่รับเข้ามามีค่าเป็น 1

Serial Port Control Register

Serial Port Control And Status Register เป็นรีจิสเตอร์ตัวหนึ่งใน SFR ซึ่งมีชื่อย่อว่า SCON โดยค่าของแต่ละบิทจะใช้ในการควบคุมและตรวจสอบการทำงานของ Serial Port ใน MCS-51 ก่อนใช้งาน Serial Port ผู้เขียนโปรแกรมจำเป็นต้องทราบถึงความหมายของบิทต่างๆ ในรีจิสเตอร์ตัวนี้ โดยค่าข้อมูลแต่ละบิทของรีจิสเตอร์ SCON จะมีความหมายดังแสดงในรูปที่ 4.22

รีจิสเตอร์ตัวนี้ไม่เพียงแค่ใช้ควบคุมการทำงานของ Serial Port ในโหมดต่างๆ เท่านั้น เพราะมันยังใช้เป็นรีจิสเตอร์ที่เก็บข้อมูลซึ่งเป็นบิทที่ 9 สำหรับการรับและการส่งข้อมูลในโหมด 2 และ 3 (บิท TB8 และ RB8) และนอกจากนี้รีจิสเตอร์ SCON ยังมีบิทที่ทำหน้าที่อินเทอร์รัพท์ซีเรียล (Serial Port Interrupt) คือบิท TI และ RI อีกด้วย



(MSB)							(LSB)
SM0	SM1	SM2	REN	TB8	RB8	TI	RI

Where SM0, SM1 specify the serial port mode, as follows:

SM0	SM1	Mode	Description	Baud Rate
0	0	0	shift register	f_{osc} /12
0	1	1	8-bit UART	variable
1	0	2	9-bit UART	f_{osc} /64 or f_{osc} /32
1	1	3	9-bit UART	variable

- SM2 enables the multiprocessor communication feature in Modes 2 and 3. In Mode 2 or 3, if SM2 is set to 1 then RI will not be activated if the received 9th data bit (RB8) is 0. In Mode 1, if SM2 = 1 then RI will not be activated if a valid stop bit was not received. In Mode 0, SM2 should be 0.
- REN enables serial reception. Set by software to enable reception. Clear by software to disable reception.
- TB8 is the 9th data bit that will be transmitted in Modes 2 and 3. Set or clear by software as desired.
- RB8 in Modes 2 and 3 is the 9th data bit that was received. In Mode 1, if SM2 = 0, RB8 is the stop bit that was received. In Mode 0, RB8 is not used.
- TI is transmit interrupt flag. Set by hardware at the end of the 8th bit time in Mode 0, or at the beginning of the stop bit in the other modes, in any serial transmission. Must be cleared by software.
- RI is receive interrupt flag. Set by hardware at the end of the 8th bit time in Mode 0, or halfway through the stop bit time in the other modes, in any serial reception (except see SM2). Must be cleared by software.

รูปที่ 4.22 SCON : Serial Port Control Register

Baud Rate

Baud Rete หมายความถึงอัตราเร็วในการรับหรือส่งข้อมูล โดยใน MCS-51 ค่า Baud Rate สำหรับการรับหรือส่งข้อมูลจะมีค่าเท่าใดก็ขึ้นอยู่กับการทำงานในแต่ละโหมดของ Serial Port ดังนี้

- ในโหมด 0 Baud Rate = (ความถี่ออสซิลเลเตอร์ที่ใช้)/12

- ในโหมด 2 ค่า Baud Rate ขึ้นอยู่กับค่าของบิท SMOD ซึ่งอยู่ในรีจิสเตอร์ PCON โดย ถ้าบิท SMOD = 0 (ซึ่งเป็นค่านี้เมื่อเกิดการรีเซ็ท) Baud Rate จะเป็น 1/64 ของความถี่ออสซิลเลเตอร์ที่ใช้ แต่ถ้าบิท SMOD = 1 Baud Rate จะเป็น 1/32 ของความถี่ออสซิลเลเตอร์ หรืออาจเขียนเป็นสูตรได้ว่า

$$\text{Baud Rate ในโหมด 2} = [2^{(SMOD)} * (\text{Oscillator Frequency})]/64$$

-- Baud Rate ในโหมด 1 และ 3 จะถูกกำหนดโดยอัตราการเกิด Over flow ของ Timer 1 (Timer 1 Overflow Rate) แต่ถ้าเป็น 8052 ซึ่งมี Timer/Counter เพิ่มมาอีก 1 ตัว เราสามารถใช้ Timer 2 ที่มีเพิ่มมานี้เป็นตัวกำหนด Baud Rate ได้ ทำให้มี Timer จำนวน 2 ตัว (Timer 1 และ Timer 2) ที่สามารถนำมากำหนด Baud Rate โดยอาจใช้ตัวใดตัวหนึ่งในการกำหนด Baud Rate การรับ ส่วนอีกตัวหนึ่งสำหรับการส่งข้อมูล ทำให้การรับและการส่งมีค่า Baud Rate ที่แตกต่างกันได้

<u>การใช้ Timer 1 เป็นตัวสร้าง Baud Rates</u>

เมื่อ Timer 1 ถูกใช้เป็นตัวกำหนด Baud Rate (Baud Rate Generator) สำหรับการทำงานของ Serial Port ในโหมด 1 และ 3 ค่าของ Baud Rate ที่ได้จะถูกกำหนดด้วยอัตราการเกิด Over flow ของ Timer 1 และขึ้นอยู่กับบิท SMOD ในรีจิสเตอร์ PCON ซึ่งเราอาจเขียนเป็นสมการที่ใช้คำนวณหา Baud Rate ได้ดังนี้

$$\text{Baud Rate โหมด 1,3} = [2^{(SMOD)} * (\text{Timer 1 Overflow Rate})]/32$$

เนื่องจากเมื่อเกิด Overflow ใน Timer ตัวใด จะทำให้ซีพียูถูกอินเทอร์รัพท์ ดังนั้นเมื่อเรานำ Timer 1 มาเป็นตัวสร้าง Baud Rate จึงควรห้ามไม่ให้เกิดอินเทอร์รัพท์ขึ้นในระหว่างการรับส่งข้อมูล และตัว Timer เองยังสามารถถูกกำหนดให้ทำงานเป็น Timer หรือ Counter อย่างใดอย่างหนึ่ง ซึ่งมีโหมดการทำงานย่อยลงไปอีก 3 โหมด ดังได้กล่าวมาแล้วในเรื่อง Time/Counter ในการใช้งานที่พบบ่อยที่สุดนั้น Timer 1 ถูกกำหนดให้อยู่ในโหมด Auto-Reload (ค่า 4 บิทบนของ TMOD คือ 0010B) ในกรณีนี้ Baud Rate จะถูกกำหนดโดยสมการดังนี้

$$\text{Baud Rate Mode 1,3} = 2^{SMOD} * \text{Oscillator Frequency}$$

$$32 \qquad 12 * [256 - (TH1)]$$

เราสามารถที่จะสร้าง Baud Rate ค่าต่ำๆ ได้ด้วย Timer 1 โดยการปล่อยให้ Timer 1 สามารถอินเทอร์รัพท์ซีพียูได้ และกำหนดการทำงานเป็น Timer ขนาด 16 บิท (โหมด 1 ค่าบิทบนของ TMOD คือ 0010B) และใช้ Timer 1 อินเทอร์รัพท์ซีพียูเพื่อทำการโหลดค่าเองใหม่ด้วยซอฟท์แวร์ขณะเกิด Overflow เนื่องจากในโหมด 16 บิท ไม่มีการทำงานแบบ Auto-Reload นั่นเอง

ตารางที่ 4 เป็นค่า Baud Rate ค่าต่างๆ ที่ใช้กันมากและบอกว่าสามารถใช้ได้จาก Timer 1 อย่างไร

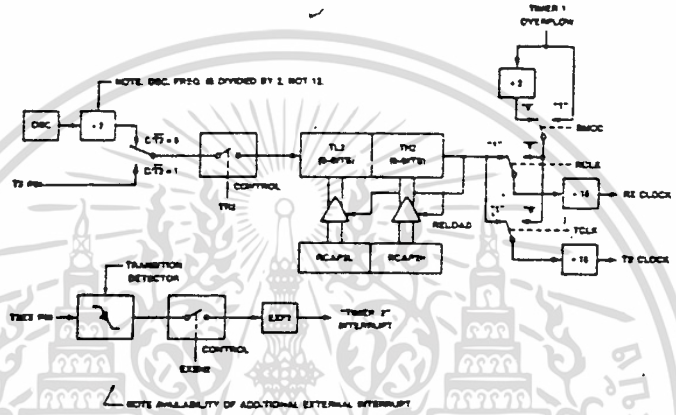
ตารางที่ 4 Timer 1 Generated Commonly Used Baud Rate

Baud Rate	f_{osc}	SMOD	Timer 1		
			C/T	Mode	Reload Value
Mode 0 Max: 1 MHZ	12 MHZ	X	X	X	X
Mode 2 Max: 375K	12 MHZ	1	X	X	X
Modes 1, 3: 62.5K	12 MHZ	1	0	2	FFH
19.2K	11.059 MHZ	1	0	2	FDH
9.6K	11.059 MHZ	0	0	2	FDH
4.8K	11.059 MHZ	0	0	2	FAH
2.4K	11.059 MHZ	0	0	2	F4H
1.2K	11.059 MHZ	0	0	2	E8H
137.5	11.986 MHZ	0	0	2	1DH
110	6 MHZ	0	0	2	72H
110	12 MHZ	0	0	1	FEEBH

การใช้ Timer 2 ในการสร้าง Baud Rate

ใน 8052 ซึ่งมี Timer 2 เพิ่มขึ้นมาให้ผู้ใช้อีก 1 ตัว ซึ่งสามารถถูกเลือกให้มีการทำงานเป็นตัวกำหนด Baud Rate (Baud Rate Generator) ได้โดยการเซ็ทบิท TCLK และ/หรือ RCLK ในรีจิสเตอร์ T2CON (รูปที่ 4.16 ในเรื่อง Timer) จากรูปจะเห็นว่าบิท TCLK , RCLK จะเป็นสวิทช์เลือกระหว่างการใช้อัตราการเกิด overflow ของ Timer 2 หรือ Timer 1 เป็นตัวสร้าง Baud Rate ดังนั้น Baud Rate สำหรับการส่งและการรับสามารถที่จะต่างกันได้ในเวลาเดียวกัน (ให้บิท RLCK มีค่า

เป็น 1 ส่วน TCLK มีค่าเป็น 0 หรือกลับกัน) การกำหนด RCLK และ/หรือ TCLK ให้เป็น 1 จะทำให้ Timer 2 ถูกเลือกใช้เป็นตัวสร้าง Baud Rate และอยู่ในโหมดการสร้าง Baud Rate ดังแสดงในรูปที่ 4.23



รูปที่ 4.23 Timer 2 in Baud Rate Generator Mode

Baud Rate Generator Mode มีการทำงานเหมือนโหมด Auto-Reload ตรงที่การเกิด Rollover ในรีจิสเตอร์ TH2 ทำให้ Timer 2 รีจิสเตอร์ (TL2 และ TH2) ถูกโหลดด้วยค่าขนาด 16 บิทจากรีจิสเตอร์ RCAP2L และ RCAP2H ซึ่งผู้ใช้สามารถตั้งค่าไว้ล่วงหน้าด้วยซอฟท์แวร์ได้

เมื่อ Timer 2 ทำงานในโหมดนี้ Baud Rate ของการรับหรือการส่งข้อมูลในโหมด 1 และ 3 จะถูกกำหนดโดยอัตราการเกิด Overflow ของรีจิสเตอร์ Timer 2 ตามสมการ

Mode 1,3 Baud Rate = [Timer 2 Overflow Rate]/16

Timer สามารถถูกกำหนดสำหรับการทำงานเป็น Timer หรือ Counter อย่างใดอย่างหนึ่ง ซึ่งการทำงานในโหมด Timer ของ Timer 2 จะแตกต่างออกไปเล็กน้อยเมื่อมันถูกใช้เป็น Baud Rate Generator กล่าวคือ ปกติ Timer จะถูกเพิ่มค่าทุกๆ machine cycle (ดังนั้นความถี่จึงเป็น 1/12 ของความถี่ออสซิลเลเตอร์) แต่เมื่อใช้เป็นตัวสร้าง Baud Rate มันจะเพิ่มค่าทุกๆ state time (ใน 1 machine cycle มี 6 state) นั่นคือใช้ความถี่ 1/2 ของออสซิลเลเตอร์ ในกรณีนี้ Baud Rate

จะถูกกำหนดโดยสมการ

Mode 1,3 Baud Rate = [Oscillator Frequency]/[32*(65535-(RCAP2H,RCAP2L)]

เมื่อ RCAP2L , RCAP2H เป็นค่าของรีจิสเตอร์ RCAP2L และ RCAPH ตามลำดับ ซึ่งถูกกำหนดเป็นจำนวนเต็ม ไม่มีเครื่องหมาย ขนาด 16 บิท

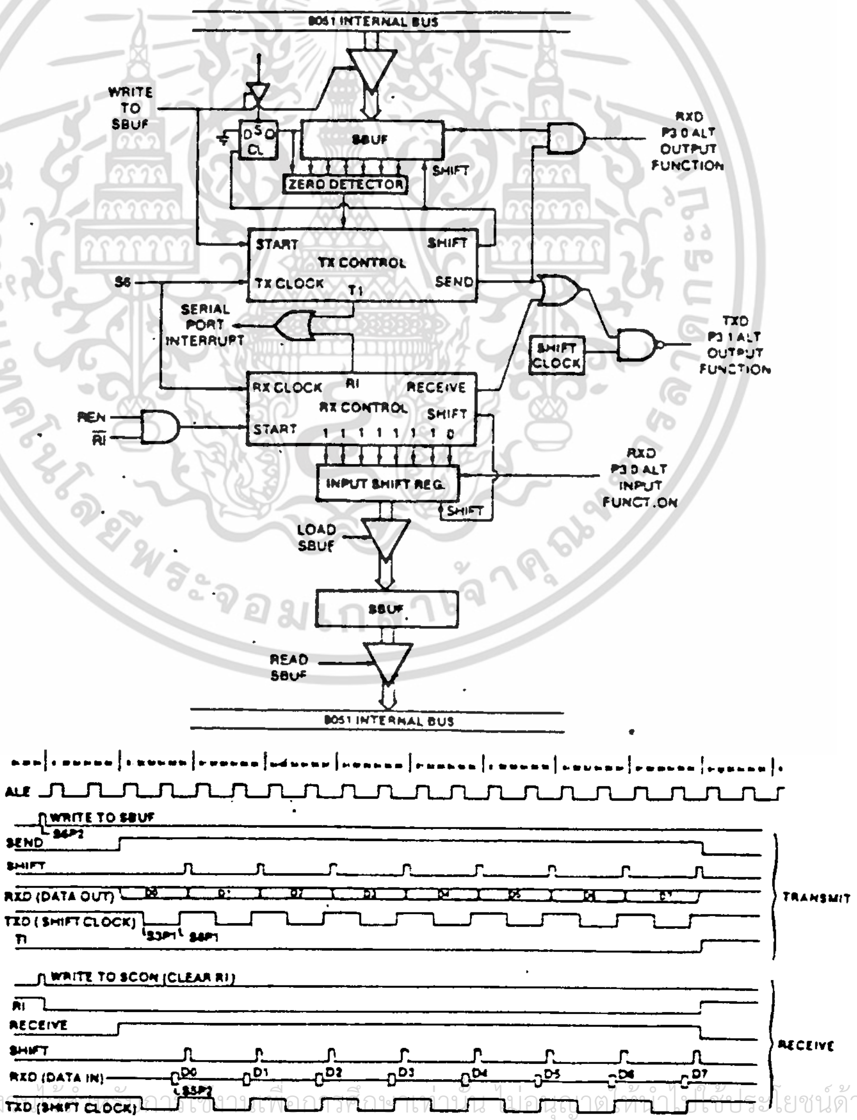
Timer 2 ในการทำงานเป็น Baud Rate Generator แสดงในรูปที่ 3 จากรูปนี้จะเห็นได้ว่า Baud Rate จะถูกสร้างขึ้นก็ต่อเมื่อบิท RCLK,TCLK ตัวใดตัวหนึ่งมีค่าเป็น 1 (RCLK+TCLK = 1) ใน T2CON และเมื่อเปรียบเทียบกับภาพแสดงการทำงานของ Timer 2 จะสังเกตุได้ว่าการเกิด Rollover ใน TH2 จะไม่เซ็ทบิท TF2 จึงไม่ทำให้เกิดอินเทอร์รัพท์ ดังนั้นผู้ใช้จึงไม่จำเป็นต้องทำการ disable interrupt เมื่อ Timer 2 อยู่ใน Baud Rate Generator Mode และจากด้านล่างของรูปที่ 3 จะเห็นว่าถ้า EXEN2 ถูกเซ็ทเป็น 1 การเปลี่ยนสถานะของสัญญาณจาก 1 เป็น 0 (1 to 0 transition) ที่ขา T2EX จะมีผลไปเซ็ทบิท EXF2 ทำให้สามารถอินเทอร์รัพท์ซีพียูได้หากต้องการ โดยไม่มีการโหลดค่าจากรีจิสเตอร์ RCAP2L และ RCAP2H ไปยังรีจิสเตอร์ Timer 2 เลย ทั้งนี้เพื่อให้ผู้ใช้สามารถนำขา T2EX มาทำเป็น External Interrupt พิเศษในระหว่างใช้งานเป็น Baud Rate Generator ได้ถ้าต้องการ และจากรูปจะเห็นว่าบิท SMOD ไม่ได้มีผลในการเพิ่มค่า Baud Rate เมื่อใช้ Timer 2 เป็นตัวกำหนด Baud Rate

สิ่งหนึ่งที่ควรระลึกไว้เสมอในการใช้งาน Timer 2 เป็น Baud Rate Generator นั่นคือ เมื่อ Timer 2 กำลังทำงาน (บิท TR2 = 1) ในโหมด Timer เพื่อสร้าง Baud Rate ผู้ใช้ไม่ควรพยายามที่จะอ่านหรือเขียนรีจิสเตอร์ของ Timer 2 (TL2 , TH2) ทั้งนี้เพราะภายใต้สภาพการทำงานเช่นนี้ Timer จะถูกเพิ่มค่าทุกๆ state time ทำให้ข้อมูลที่ได้จากการอ่านหรือเขียนไม่เที่ยงตรง แต่ผู้ใช้อาจจะเขียนหรืออ่านข้อมูลจาก RCAP Register (RCAP2L,RCAP2H) แต่ถึงอย่างไรก็ไม่ควรมีการเขียนข้อมูลไปยังรีจิสเตอร์ทั้ง 2 นี้ เพราะข้อมูลที่เขียนลงไปอาจไปทับการโหลดค่าใหม่ครั้งต่อไปพอดี ทำให้การเขียนหรือการโหลดค่าไปยังรีจิสเตอร์ของ Timer 2 เกิดการผิดพลาดขึ้น (ตัวอย่างเช่นผู้ใช้ต้องการเปลี่ยน Baud Rate) ในกรณีนี้ ผู้ใช้ควรจะเคลียร์บิท TF2 เสียก่อน หลังจากนั้นจึงค่อยกระทำการใดๆ กับรีจิสเตอร์นี้ตามต้องการ

อธิบายการทำงานในโหมด 0 เพิ่มเติม

การทำงานในโหมดนี้ ดังได้กล่าวมาแล้วว่าข้อมูลถูกรับเข้าและส่งออกผ่านขา RXD ส่วนขา TXD เป็นตัวสร้าง Shift Clock ข้อมูลถูกส่งและรับโดยเริ่มจาก start bit (0) data ขนาด 8 บิท โดยเริ่มที่บิทต่ำสุดก่อน (LSB First) และตามมาด้วย stop bit (1) โดยในการรับข้อมูล stop bit จะถูกนำไปไว้ในบิท RB8 ของรีจิสเตอร์ SCON ส่วน Baud Rate จะคงที่เท่ากับ 1/12 ของความถี่ออสซิลเลเตอร์

รูปที่ 4.24 แสดงถึงโครงสร้างในการทำงานคร่าวๆ ของ Serial Port ในโหมด 0 รวมทั้ง Timing diagram ที่เกี่ยวข้องสำหรับการส่งและการรับข้อมูล



รูปที่ 4.24 Serial Port Mode 0

จากรูปที่ 4.24 การส่งข้อมูลถูกทำให้เริ่มต้นโดยคำสั่งใดๆ ที่ใช้รีจิสเตอร์ SBUF เป็น destinati register สัญญาณ write to SBUF ซึ่งเกิดขึ้นขณะ S6P2 จะเป็นการโหลดค่า 1 ไปที่ตำแหน่งที่ 9 ของ transmit shift register และบอกให้ TX Control Block เริ่มทำการส่งข้อมูลที่มีอยู่ในรีจิสเตอร์ SBUF (ข้อมูลที่อยู่ใน SBUF ได้จากการใช้คำสั่งที่ใช้รีจิสเตอร์ SBUF เป็น destination register ดังที่ได้กล่าวมาแล้ว) แผนผังเวลาที่แสดงการส่งข้อมูลจะเห็นได้ว่ามี machine cycle 1 machine เต็มอยู่ระหว่างช่วงสัญญาณ write to SBUF และสัญญาณ SEND ขณะเริ่ม active ซึ่งได้มาจาก TX Control (ดูในภาพที่ 4.24)

สัญญาณ SEND ที่ถูก enable (มีค่าเป็น 1) จะเป็นสัญญาณออกจากส่วน TX control เพื่อไปทำการ enable output ของ shift register ให้ผ่านออกไปยังขา RXD ซึ่งตรงกับ P3.0 และ enable ให้ shift clock ผ่านออกไปทางขา TXD ซึ่งตรงกับ P3.1 โดย shift clock จะเป็นสัญญาณซึ่งมีค่าเป็น low ระหว่าง S3,S4,S5 ของทุก machine และมีค่า high ระหว่าง S6,S1,S2 ของทุก machine cycle ดังแสดงให้เห็นในภาพ โดยที่เมื่อถึง S6P2 ของทุกๆ machine cycle ซึ่งสัญญาณ SEND ยังคงถูก enable อยู่ ข้อมูลของ transmit shift register จะถูก shifted ไปทางขวา 1 ตำแหน่ง ทำให้ข้อมูลที่ได้ออกมาทางขา RXD ทีละ 1 บิท โดยเริ่มจากบิทต่ำสุดก่อน

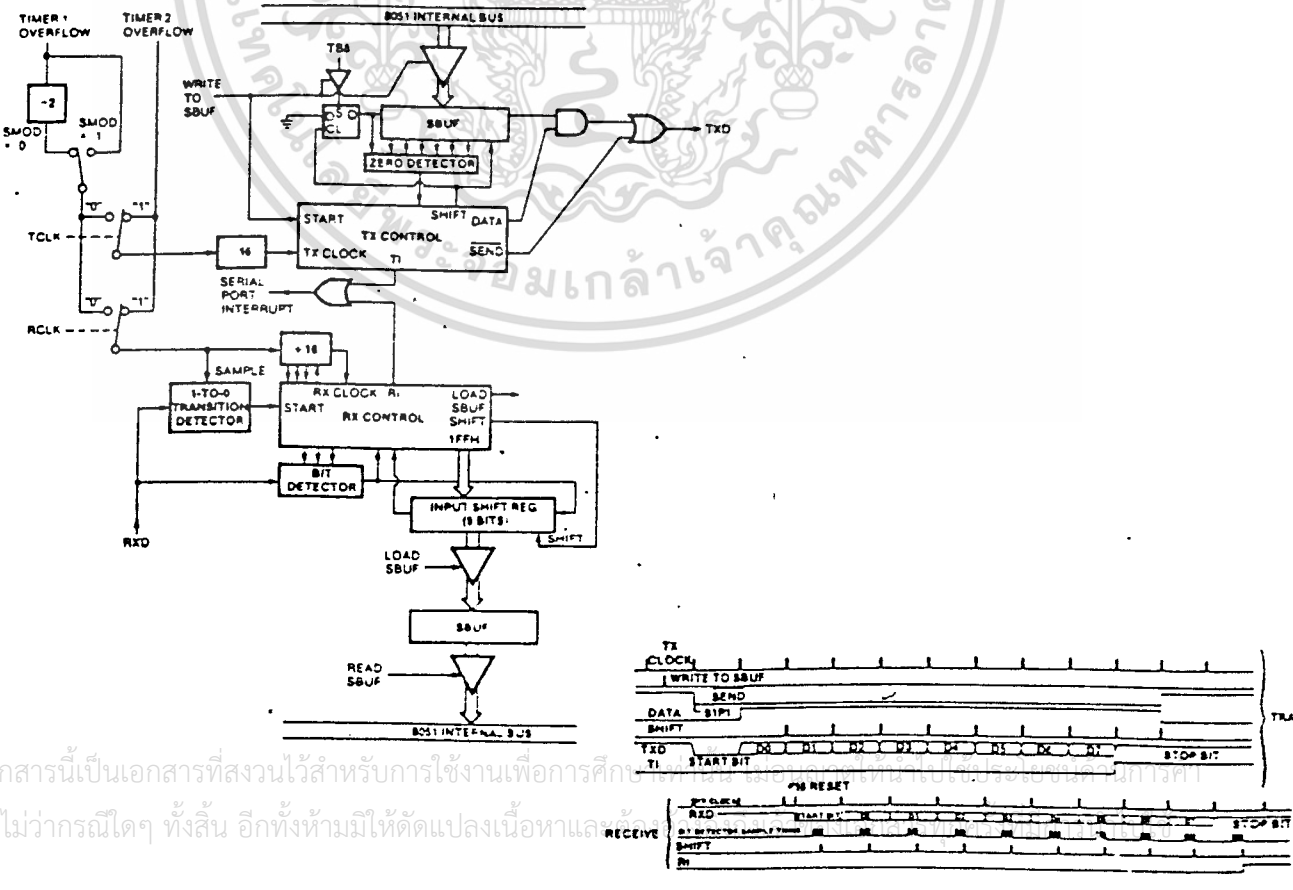
ขณะที่ข้อมูลซึ่งถูก shift ออกมาทางขวาทีละบิทนี้ ค่า 0 จะถูกนำมาแทนที่ตำแหน่งซึ่งถูก shift ออกไป โดยเข้ามาทางด้านซ้ายของ transmit shift Register การส่งข้อมูลจะดำเนินไปเรื่อย ๆ จนกระทั่งข้อมูลซึ่งเป็น MSB ของ data byte ถูกเลื่อนไปอยู่ที่ output position ของ shift register และค่า 1 ที่เคยถูกโหลดไปที่ตำแหน่งที่ 9 ในตอนแรกนั้น จะอยู่ถัดจาก MSB มาทางซ้าย โดยทุกตำแหน่งทางซ้ายของมันจะมีค่าเป็น 0 หมด (มีศูนย์ทางซ้ายรวม 7 ตัว) ในสภาวะเช่นนี้จะส่งผลให้วงจรตรวจจับศูนย์ (ZERO Dector) ซึ่งตรวจจับเพียงแค่ 7 บิทเริ่มทำงานโดยการส่งสัญญาณไปบอก TX control block ให้ทำการ shift ครั้งสุดท้ายอีก 1 ครั้ง เพื่อส่ง MSB ออกไป แต่บิทที่ 9 จะไม่ถูกส่งตามออกมา จะยังคงอยู่ที่ output position ของ shift register เท่านั้น จากนั้นจึงค่อยหยุดการส่ง (deactived SEND) และเซ็ทบิท TI เพื่ออินเทอร์รัพท์ซีพียู การกระทำทั้งสองนี้จะเกิดขึ้นที่ขณะ S1P1 ของ machine cycle ที่ 10 หลังจากที่มีสัญญาณ Write to SBUF

การรับข้อมูลในโหมด 0 เกิดขึ้นได้ก็ต่อเมื่อบิท REN = 1 และ RI = 0 (พิจารณาจากรูปที่ 4) โดยขณะ S6P2 ของ machine cycle ถัดไป RX control block จะทำการเขียนข้อมูล 11111110 ไปที่ Receive shift register และใน clock phase ถัดไป ก็จะเริ่มรับข้อมูล

(activated RECEIVE)

สัญญาณ RECEIVE ที่ได้มาจาก RX control block จะทำการ enable shift clock ไปที่ขา TXD ซึ่งมีการเปลี่ยนสถานะที่ทุกๆ S3P1 และ S6P1 ของทุกๆ machine cycle โดยขณะ S6P2 ของแต่ละ machine cycle ซึ่งสัญญาณ RECEIVE ยังคงทำงานอยู่ ข้อมูลของ RECEIVE Shift Register จะถูก shift มาทางซ้าย 1 ตำแหน่ง ค่าที่เข้ามาทางขวาจะเป็นข้อมูลซึ่งได้รับการตรวจสอบ (sampled) ที่ขา RXD ขณะ S5P2 ของ machine cycle เดียวกัน

ขณะที่ข้อมูลเข้ามาทางขวาทีละบิท ค่าของ 1 ซึ่งถูกนำไปไว้ใน RECEIVE shift register ในตอนแรกแล้ว จะถูกเลื่อนออกไปทางซ้าย (shift out) เมื่อ 0 ซึ่งถูกโหลดไปไว้ที่บิทต่ำสุดตั้งแต่ตอนแรกเช่นกัน ถูกเลื่อนมาอยู่ในตำแหน่งซ้ายสุดของ shift register มันจะส่งสัญญาณที่บ่งบอกถึงการรับข้อมูลเสร็จสิ้นแล้วไปยัง RX control block เพื่อให้ทำการรับข้อมูลอีก 1 บิทโดยการ shift มาทางซ้ายอีก 1 ครั้ง และส่งสัญญาณ LOAD SBUF เพื่อเปิดให้ข้อมูลเข้าสู่รีจิสเตอร์ SBUF การกระทำทั้งหมดนี้เกิดขึ้นในช่วง S1P1 ของ machine cycle ที่ 10 หลังจากที่มีการ write ไปที่ SCON ซึ่งเป็นการเคลียร์บิท RI เมื่อการรับข้อมูลสิ้นสุดลงแล้ว RECEIVE จะถูกเคลียร์ และบิท RI จะถูกเซ็ท



รูปที่ 4.25 Serial Port MODE 1 TCLK,RCLK and TIMER 2 are Present in the 8052 Onl

อธิบายการทำงานโหมด 1 เพิ่มเติม

การทำงานของ serial port ในโหมด 1 นั้น ดังที่ได้กล่าวมาแล้วว่าข้อมูล 10 บิทถูกส่งผ่านขา TXD และรับเข้าทางขา RXD โดยข้อมูลจะประกอบไปด้วย 1 start bit (0) 8 data bits (รับหรือส่งบิทต่ำสุดก่อน) ตามด้วย 1 stop bit (1) ในการรับ stop bit จะถูกนำไปไว้ในบิท RB8 ของรีจิสเตอร์ SCON ใน 8051 Baud Rate ถูกกำหนดโดยอัตราการเกิด Overflow ของ Timer 1. (Timer 1 Overflow Rate) ส่วนใน 8052 มันจะถูกกำหนดโดยอัตราการเกิด Overflow ของ Timer 1 หรือ Timer 2 อย่างใดอย่างหนึ่ง หรือทั้ง 2 อย่าง (ตัวหนึ่งสำหรับการส่ง และอีกตัวหนึ่งสำหรับการรับ) โดยใช้บิท RCLK และบิท TCLK เป็นสวิทช์เลือก (รูปที่ 4.23) รูปที่4.26 แสดงการทำงานของ Serial Port ในโหมดนี้อย่างคร่าวๆ รวมทั้งแผนผังเวลาที่เกี่ยวข้องสำหรับการส่งและการรับข้อมูล

การส่งข้อมูลจะเริ่มต้นโดยคำสั่งใดๆ ที่ใช้รีจิสเตอร์ SBUF เป็น Destination Register สัญญาณ write to SBUF จะเป็นการโหลด 1 ไปที่ตำแหน่งที่ 9 ของ transmit shift register และจะไปกระตุ้นให้ TX control block ทราบว่าขณะนี้กำลังต้องการส่งข้อมูล การส่งข้อมูลจริงๆ จะเกิดขึ้นที่ S1P1 ของ machine cycle ที่ถัดจากการเกิด Rollover ครั้งถัดไปใน Counter ซึ่งถูกหารด้วย 16 (Counter ซึ่งถูกเพิ่มค่าเมื่อเกิด rollover 16 ครั้ง ดังในรูปที่ 5) ดังนั้นจังหวะการส่งข้อมูลแต่ละบิทจะซิงโครไนซ์กับค่าของ Counter หาร 16 ไม่ใช่กับสัญญาณ write to SBUF

การส่งเริ่มต้นด้วยสัญญาณ SEND* ถูก Active และทำการใส่ start bit ไปยังขา TXD เมื่อเวลาของการส่งบิทแรกผ่านไป ข้อมูลจะถูกส่งตามมาโดยการ enable output bit ของ transmit shift register ไปที่ขา TXD shift pulse ลูกที่ 1 เกิดขึ้นภายหลังการส่งข้อมูลไป 1 บิท

ขณะที่ data bit shift out ไปทางขวา ค่า 0 จะถูก clock เข้ามาทางซ้าย เมื่อ MSB ของ data bit ไปอยู่ที่ตำแหน่ง output ของ shift register แล้ว 1 ซึ่งถูกโหลดไปไว้ในตำแหน่งที่ 9 ในตอนแรกจะอยู่ถัดจาก MSB ไปทางซ้าย ส่วนตำแหน่งอื่นๆ ทางซ้ายทั้งหมดมีค่าเป็นศูนย์ ซึ่งในสภาวะเช่นนี้ ตัวตรวจจับศูนย์ (ขนาด 7 บิท) จะทำงานโดยการส่งสัญญาณไปบอก TX control block ให้ทำการ shift ครั้งสุดท้ายอีก 1 ครั้ง แล้วจึงหยุดการส่งข้อมูล (deactivate SEND*) พร้อมทั้งเซ็ทบิท TI ซึ่งเหตุการณ์ที่เกิดขณะการส่งข้อมูลเสร็จสิ้นลงจะเกิดขึ้นที่ขณะเกิด rollover ครั้งที่ 10 ของ counter ที่ถูกหาร 16 หลังจากสัญญาณ write to SBUF เกิดขึ้น

การรับข้อมูลเริ่มต้นเมื่อตรวจพบการเปลี่ยนสถานะจาก 1 เป็น 0 ที่ขา RXD ซึ่งการตรวจจับสถานะนี้จะถูก sampled ด้วยอัตรา 16 ครั้ง (อัตราเดียวกับการเกิด rollover ซึ่งถูกส่งต่อไปให้ counter หาร 16 ต่อไป) ไม่ว่าค่า Baud Rate จะถูกกำหนดเป็นเท่าใดก็ตาม เมื่อการเปลี่ยนสถานะของสัญญาณถูกตรวจพบ Counter หาร 16 จะถูกรีเซ็ทในทันที และข้อมูล 1FFH จะถูกนำไปไว้ใน input shift register การรีเซ็ท Counter หาร 16 ก็เพื่อตั้งการ rollover ให้เริ่มตรงกับการเริ่มต้นของบิทที่ได้รับเข้ามา

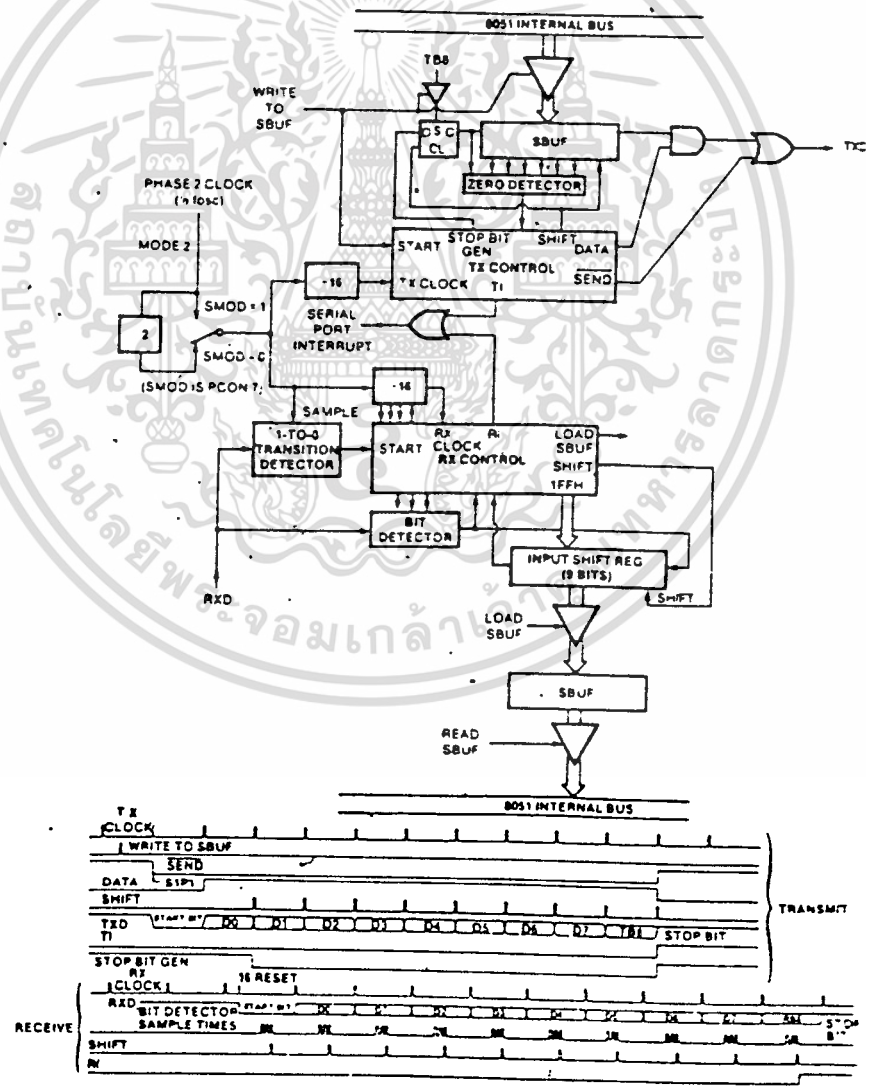
จากรูป Counter จะถูกนับขึ้นไป 1 ก็ต่อเมื่อมีการเกิด overflow ขึ้น 16 ครั้ง หรือกล่าวง่ายๆ ได้ว่า เมื่อเกิด overflow 16 ครั้ง ค่าของ counter จะถูกเพิ่มขึ้น 1 เสมอ ซึ่งเวลาในการรับข้อมูลแต่ละบิท จะนำเอาอัตราการเกิด overflow มาเป็นตัวตรวจสอบ นั่นคือแบ่งเวลาในการรับข้อมูลแต่ละบิทออกเป็น 16 state โดยมีตัวตรวจสอบข้อมูลแต่ละบิท (BIT DETECTOR) ทุกๆ state ที่ 7,8 9 ของเวลาในข้อมูลแต่ละบิท ตัวตรวจสอบบิทจะตรวจสอบค่าสถานะที่ขา RXD ซึ่งมีข้อมูลรออยู่ ค่าที่รับเข้ามาจะถือว่าเป็นบิทข้อมูลที่ถูกต้องก็เมื่อการตรวจสอบพบว่าเป็นค่าเดียวกันอย่างน้อย 2 ใน 3 ของ state 7,8,9 ที่ต้องทำเช่นนี้เพื่อเป็นการป้องกันสัญญาณรบกวน (noise rejection) แต่ในช่วงของการรับบิทแรก (start bit) ถ้าค่าที่ถูกตรวจสอบได้ใน state 7,8,9 ปรากฏว่าไม่เป็น 0 วงจรที่ทำหน้าที่รับข้อมูลจะถูกรีเซ็ทและระบบจะเริ่มกลับไปตรวจหาการเปลี่ยนสถานะจาก 1 เป็น 0 ที่ขา RXD ต่อไป ที่ต้องทำเช่นนี้ก็เพื่อยกเลิกการรับข้อมูลหาก start bit ไม่ถูกต้อง แต่ถ้า start bit ถูกตรวจสอบแล้วปรากฏว่ามีค่าเป็น 1 การรับข้อมูลบิทต่อไปจะเริ่มต้นทันทีโดย start bit จะถูก shift ไปไว้ใน input shift register และบิทข้อมูลที่เหลือจะถูกรับตามมา

ขณะที่ data bit เข้ามาทางขวา 1 (1FFH ที่ถูกโหลดไว้ก่อน) จะถูก shift ไปทางซ้ายจนกระทั่งเมื่อ start bit ถูกเลื่อนไปถึงตำแหน่งซ้ายสุดใน shift register (ในโหมด 1 รีจิสเตอร์นี้จะมีขนาด 9 บิท) มันจะส่งสัญญาณไปบอก RX control block ให้ทำการ shift ครั้งสุดท้ายอีก 1 ครั้ง เพื่อรับ stop bit ไปไว้ในบิท RB8 ในรีจิสเตอร์ SCON จากนั้นจึงส่งสัญญาณไปโหลดข้อมูลจาก input shift register เพื่อนำไปเก็บไว้ในรีจิสเตอร์ SBUF และ RX control block ก็จะส่งสัญญาณ RI (RI ถูกเซ็ท) เพื่ออินเทอร์รัพท์ซีพียู แต่สัญญาณที่จะโหลดข้อมูลไปไว้ในรีจิสเตอร์ SBUF และรับ stop bit ไปไว้ในบิท RB8 รวมทั้งการเซ็ทบิท RI จะถูกสร้างขึ้นก็ต่อเมื่อเงื่อนไขต่อไปนี้เป็นจริงขณะสัญญาณ shift pulse ลูกสุดท้ายถูกสร้างขึ้น

- บิท RI ถูกเคลียร์ (RI = 0) และ

- บิท SM2 = 0 และ stop bit = 1

ถ้าเงื่อนไขอย่างใดอย่างหนึ่งของทั้งสองอย่างนี้ไม่ตรง กลุ่มข้อมูลที่รับเข้ามาจะหายไปโดยไม่สา มารถเรียกคืนได้ ถ้าเงื่อนไขทั้งสองถูกต้อง stop bit จะถูกนำไปไว้ในบิท RB8 ในรีจิสเตอร์ SCON ส่วนบิทที่เป็นข้อมูลจำนวน 8 บิท (data bit) จะถูกโหลดไปไว้ในรีจิสเตอร์ SBUF และบิท RI จะถูก เซ็ท เมื่อถึงตอนนี้ไม่ว่าเงื่อนไขข้างต้นจะตรงหรือไม่ ระบบการรับส่งข้อมูลจะถือว่าการรับข้อมูลจำนวน 1 ไบท์ (รับข้อมูลเข้ามา 10 บิท แต่เป็น data จริงๆ เพียง 8 บิท) เสร็จสิ้นลงแล้ว และเริ่มรีเซ็ท ตัวเองเพื่อกลับไปรอรับการเปลี่ยนสถานะจาก 1 เป็น 0 ที่ขา RXD เพื่อรับข้อมูลไบท์ต่อไป



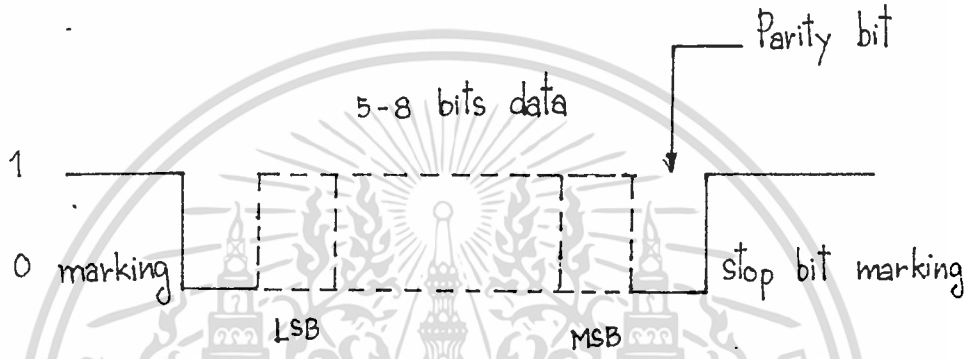
รูปที่ 4.26 Serial Port MODE 2

บทที่5

พอร์ทมาตราฐานและการโปรแกรมรับโดยภาษาปาสคาล

การสื่อสารข้อมูลแบบอะซิงโครนัส

เป็นการสื่อสารที่พัฒนามาจากการส่งโทรพิมพ์ ลักษณะของสัญญาณดังรูป



รูปที่5.1แสดงฟอร์แมท(FORMAT)ของการส่งแบบอะซิงโครนัส

เพื่อเพิ่มกลไกในการรับส่งให้ถูกต้อง สัญญาณอะซิงโครนัสจะระกอบด้วย

1.บิทเริ่มต้น (START BIT)

2.ข้อมูล (DATA)

3.บิทพาริตี้ (PARITY BIT)

4.บิทสุดท้าย (STOP BIT)

ในขณะที่เครื่องไม่มีข้อมูลส่งออกมา เรียกว่า สถานะการส่งแบบว่าง(IDLE)จะมีสัญญาณ หรือมีแรงดันตลอดเวลาเพื่อความแน่ใจว่าฝ่ายรับยังติดต่อกับฝ่ายส่งอยู่เมื่อเริ่มต้นส่งข้อมูล สัญญาณช่องอะซิงโครนัสจะเป็น"0" หนึ่งช่วงของสัญญาณนาฬิกา บิทนี้เรียกว่า บิทเริ่มต้น (START BIT)ต่อจากบิทเริ่มต้นก็เป็นข้อมูล1ตัวอักษรซึ่งอาจมีตั้งแต่5-8บิท โดยบิทต่ำ(LSB)จะถูกส่งออกมาก่อนไปจนถึงบิทสูงสุด(MSB) การเข้ารหัสแบบนี้มักใช้ASCII-CODE แต่เดิมในงานโทรพิมพ์ใช้รหัส

BAUDOTซึ่งใช้5บิท ต่อจากข้อมูลก็เป็นพาริตี้บิท ซึ่งอาจจะใช้หรือไม่ก็ได้ โดยมีหน้าที่ตรวจสอบความถูกต้องของสัญญาณที่รับได้ อาจเป็นพาริตี้คู่(EVEN PARITY)หรือพาริตี้คี่(ODD PARITY) หมายความว่าหากเป็นพาริตี้คู่ จำนวนของ"1"ในช่วงของข้อมูลรวมกับบิทพาริตี้แล้วต้องเป็นจำนวนคู่ สำหรับพาริตี้คี่ก็ทำนองเดียวกัน พาริตี้บิทนี้ผู้ส่งจะต้องตรวจสอบข้อมูลแล้วใส่พาริตี้เอง ฝ่ายรับเมื่อรับได้แล้วก็จะตรวจสอบว่าเป็นจริงดังที่กำหนดไว้หรือไม่ หลังจากพาริตี้บิทก็เป็นบิทสุดท้าย ซึ่งมีความกว้างเป็น 1,1.5,2พัลส์(PULSE)ของสัญญาณนาฬิกาแล้วแต่ผู้รับและผู้ส่ง ดังนั้นการเริ่มใช้พอร์ตอนุกรม(SERIAL PORT)จึงจำเป็นต้องตั้งค่าต่างๆสำหรับเป็นการส่งข้อมูลแบบอนุกรมอันได้แก่

1.ความเร็วในการส่ง

2.ความยาวรหัส1อักขระ

3.บิทพาริตี้

4.จำนวนบิทสุดท้าย

การส่งแบบอะซิงโครนัสนี้มีลักษณะไปทีละอักขระ จำนวนพัลส์ของสัญญาณที่ส่งออกมายังมี บางส่วนใช้ในการควบคุมการส่งอันได้แก่บิทเริ่มต้น,พาริตี้บิทและบิทสุดท้าย ทำให้เกิดการส่งอักขระ ต่อวินาทีน้อยลง การส่งสัญญาณด้วยความเร็ว 300 บอด(BAUD)สำหรับการเข้ารหัส7บิทไม่ได้ หมายความว่าส่งได้ 300/7 อักขระ/วินาที(CHARACTER PER SECOND)

<u>ความเร็วในการสื่อสารข้อมูลแบบอนุกรม</u>

ความเร็วของการสื่อสารข้อมูลแบบอนุกรมหน่วยวัดเป็นบิทต่อนาที(BPS) หน่วยที่บรรยายถึงการเปลี่ยนแปลงของสัญญาณใน1วินาที เรียกว่าบอดเรต(BAUD RATE) ในการเปลี่ยนแปลงของสัญญาณ1ครั้งอาจจะแสดงถึงการส่งข้อมูลแบบอนุกรมมากกว่า1บิท ถ้าเขียนในรูปของสะการจะได้ว่า

BIT RATE(BPS) = BAUD RATE x (บิทใน1บอด)

<u>การควบคุมการส่งข้อมูล</u>

1.การมีบัฟเฟอร์ในการสื่อสารข้อมูล

บัฟเฟอร์ สำหรับการสื่อสารก็คือ หน่วยความจำในคอมพิวเตอร์ซึ่งแบ่งแยกออกมาจากหน่วยความจำหลักสำหรับเก็บข้อมูลชั่วคราว ส่วนมากใช้เฉพาะฝ่ายรับเท่านั้น เนื่องจากฝ่ายรับต้องตามฝ่ายส่งให้ทัน

2.การควบคุมโดยใช้ XON/XOFF

ในบางครั้งการส่งถ่ายข้อมูลด้วยความเร็วสูงและขนาดของข้อมูลมีขนาดใหญ่ ซึ่งอาจจะมีบัฟเฟอร์ไม่พอก็จำเป็นต้องควบคุมการส่งโดยการบอกให้ฝ่ายส่งหยุดชั่วคราว(XOFF) จนกว่าฝ่ายรับจะจัดการเอาข้อมูลออกจากบัฟเฟอร์ส่อสารหมดเสียก่อน จึงบอกให้ฝ่ายส่งจัดการส่งต่อไป(XON)

3.การใช้โปรโตคอล

เทคนิคอีกอย่างในการควนคุมการรับส่ง คือการใช้โปรโตคอล(PROTOCOL TRANSFER) เทคนิคนี้จำเป็นต้องมีเหมือนกันทั้งฝ่ายรับและส่งโดยการใช้อักขระควบคุมในตารางของASCII-CODE สำหรับควบคุมการส่งข้อมูลออกมาเป็นกลุ่มที่มีขนาดคงที่

อักขระที่ใชช้เป็นโปรโตคอล ในการควบคุมการส่งจากตารางASCIIมีดังนี้

ETB = END OF TRANSMISSION BLOCK

มีค่า 23 เป็นการบอกฝ่ายรับว่าบระนี้สิ้นสุดการส่งข้อมูลกลุ่มหนึ่งแล้ว

ETB = END OF TEXT

มีค่า 03 เป็นการบอกฝ่ายรับว่าขณะนี้สิ้นสุดการส่งแล้ว

ENQ = ENQUIRY

มีค่า 05 เป็นอักขระที่ส่งมาจากฝ่ายรับบอกให้ฝ่ายส่งส่งข้อมูล

NACK = NEGATIVE ACKNOWLEDGE

มีค่า 21 เป็นการบอกฝ่ายส่งว่าข้อมูลที่รับมาผิดพลาด

ACK = ACKNOWLEDGE

มีค่า 06 เป็นการบอกฝ่ายส่งว่าข้อมูลถูกต้องแล้ว

<u>พอร์ทมาตราฐาน RS-232C</u>

พอร์ต RS-232Cนี้ทำหน้าที่รับและส่งข้อมูลในแบบอนุกรม ซึ่งเป็นที่นิยมกันมากสำหรับเครื่องไมโครคอมพิวเตอร์

ความหมายของRS-232Cเป็นดังนี้

RS - RECOMMENED STANDARD

232 - เป็นหมายเลขบ่งบอกของมาตราฐานนี้

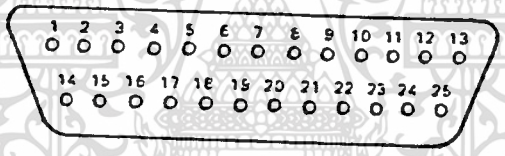
C - เป็นมาตราฐานของฉบับสุดท้ายของมาตราฐานนี้

จุดประสงค์ของมาตราฐานตัวนี้เพื่อบรรยายลักษณะของการเชื่อมต่ออุปกรณ์รับส่งข้อมูลปลายทาง

(DATA TERMINAL EQUIPTMENT -DTE)กับอุปกรณ์สื่อสารข้อมูล (DATA COMMUNICATION EQUIPTMENT) สำหรับผู้ใช้ไมโครคอมพิวเตอร์ DTE คือ ไมโครคอมพิวเตอร์และ CDE หมายถึง MODEM, อุปกรณ์อื่นๆเช่น PRINTER ที่รับสัญญาณแบบขนุกรมเป็นต้น ในการเชื่อมต่อ RS-232C สามารถส่งถ่ายข้อมูลได้จาก 0-20000 บิทต่อวินาที ซึ่งเป็นการเพียงพอสำหรับเครื่องไมโครคอมพิวเตอร์ที่มีบอดเรท 110-9600 ความยาวของสายเชื่อมต่อโดยสัญญาณตามมาตราฐานของRS-232Cจำกัดอยู่เพียง 50 ฟุต

การกำหนดจุดต่อของ RC-232C

มาตราฐานของ RS-232C กำหนดข้อต่อโดยคอนเนคเตอร์(CONNECTOR) แบบ DB-25 แต่ละขาของข้อต่อกำหนดไว้ดังรูปต่อไปนี้



Pin Number	EIA Name	RS-232-C Common Mnemonic	Description
1	AA		Protective ground
2	BA	TxD	Data transmitted from terminal
3	BB	RxD	Data received from modem
4	CA	RTS	Request to send
5	CB	CTS	Clear to send
6	CC	DSR	Data set ready
7	AB		Signal ground
8	CF	DCD	Carrier detector
9	*		Reserved for Data Set Testing
10	*		Reserved for Data Set Testing
11	*		Unassigned
12	SCF		Secondary carrier detector
13	SCB		Secondary clear to send
14	SBA		Secondary transmitted data
15	DB		Transmitted bit clock, from DCE
16	SBB		Secondary received data
17	DD		Received bit clock
18	*		Unassigned
19	SCA		Secondary request to send
20	CD	DTR	Data terminal ready
21	CG		Signal quality detector
22	CE		Ring indicator (used by auto answer equipment)
23	CH/CI		Data signal rate selector
24	DA		Transmitted bit clock, from DTE
25	*		Unassigned

*Undefined

รูปที่5.2แสดงRS-232C COCECTOR และหน้าที่ของขาสัญญาณ

หน้าที่ของขาสัญญาณแต่ละขามีรายละเอียดดังนี้

TxD - TRANSMIT DATA เป็นขาของสัญญาณที่ออกจากไมโครคอมพิวเตอร์ไปยังโมเด็ม (MODEM) หรือต่อเข้าโดยตรงกับไมโครคอมพิวเตอร์ตัวอื่น ในสภาวะที่ไม่มีการส่งสัญญาณออก ขานี้จะมีลอจิก "1"

RxD - RECEIVE DATA เป็นขาสัญญาณที่รับข้อมูลเข้า เมื่อไม่มีการรับสัญญาณจะมีลอจิก "1"

RTS - REQUEST TO SEND เป็นขาสัญญาณที่ใช้ในการแฮนด์เช็ค(HANDSHAKING)เป็นการเรียกร้องที่จะนำสัญญาณออกทางTxD สัญญาณนี้ใช้คู่กับสัญญาณ CTS อุปกรณ์รับหากได้รับสัญญาณ RTS จะตรวจสอบตัวเองว่าพร้อมที่จะรับข้อมูลหรือไม่ หากพร้อมก็จะตอบไปทาง CTS

CTS - CLEAR TO SEND อธิบายไว้ในส่วน RTS แล้ว

DSR - DATA SET READY เป็นขาสัญญาณที่ส่งมาเพื่อบอกฝ่ายรับว่าทำการรับข้อมูลเรียบร้อยแล้ว

DTR - DATA TERMINAL READY เป็นขาสัญญาณที่ส่งมาเพื่อบอกฝ่ายรับว่าพร้อมที่จะทำการติดต่อด้วย

CD - CARRIER DETECT เป็นขาสัญญาณที่ฝ่ายรับบอกฝ่ายส่งว่าได้รับสัญญาณเรียบร้อยแล้ว

SD - SIGNAL GROUND เป็นขาที่ทำหน้าที่เป็นกราวด์(GROUND)ของทุกสายสัญญาณ

บริการอินเทอร์รัพท์14ของไบออส(BIOS)

ในส่วนของโปรแกรมรับเพื่อแสดงผลออกหน้าจอคอมพิวเตอร์นั้นได้นำภาษาปาสคาลมาประยุกต์ใช้ซึ่งต้องอาศัยอินเทอรัพท์14ของไบออสช่วย ดังนั้นจึงขออธิบายการใช้บริการอินเทอร์รัพท์14เพียงสังเขปดังนี้

รีจิสเตอร์ AH = 0 เป็นการอินนิเชียลไลซ์(INITIALIZE)พอร์ทการสื่อสารแบบอนุกรม

รีจิสเตอร์ AL มีการเซทบิทต่างๆเพื่อการอินนิเชียลไลซ์(INITIALIZE)ดังนี้

- บิท 1,0 ใช้เซทความยาวของคำโดย

1 0 = 7บิท

1 1 = 8บิท

- บิท 2 ใช้เซทบิทสุดท้าย(STOP BIT)โดย

  0 = 1บิท

  1 = 2บิท

- บิท 4,3 ใช้เซตบิทพาริตี้(PARITY BIT)โดย

  X 0 = ไม่มีบิทพาริตี้

  0 1 = พาริตี้คี่

  1 1 = พาริตี้คู่

- บิท 7,6,5 ใช้เซทบอดเรท(BAUD RATE)

  0 0 0 = 110

  0 0 1 = 150

  0 1 0 = 300

  0 1 1 = 600

  1 0 0 = 1200

  1 0 1 = 2400

  1 1 0 = 4800

  1 1 1 = 9600

รีจิสเตอร์ AH = 1 เป็นการส่งอักขระ(CHARACTER)ใน ALไปพอร์ทสื่อสาร (COMMUNICATION PORT) บิท7ของรีจิสเตอร์AHจะถูกเซตถ้ามีความผิดพลาดในการส่ง(ERROR)ถ้าบิท7ของรีจิสเตอร์AHไม่ถูกเซทก็จะแสดงให้เห็นถึงสถานะปัจจุบัน(CURRENT STATUS)

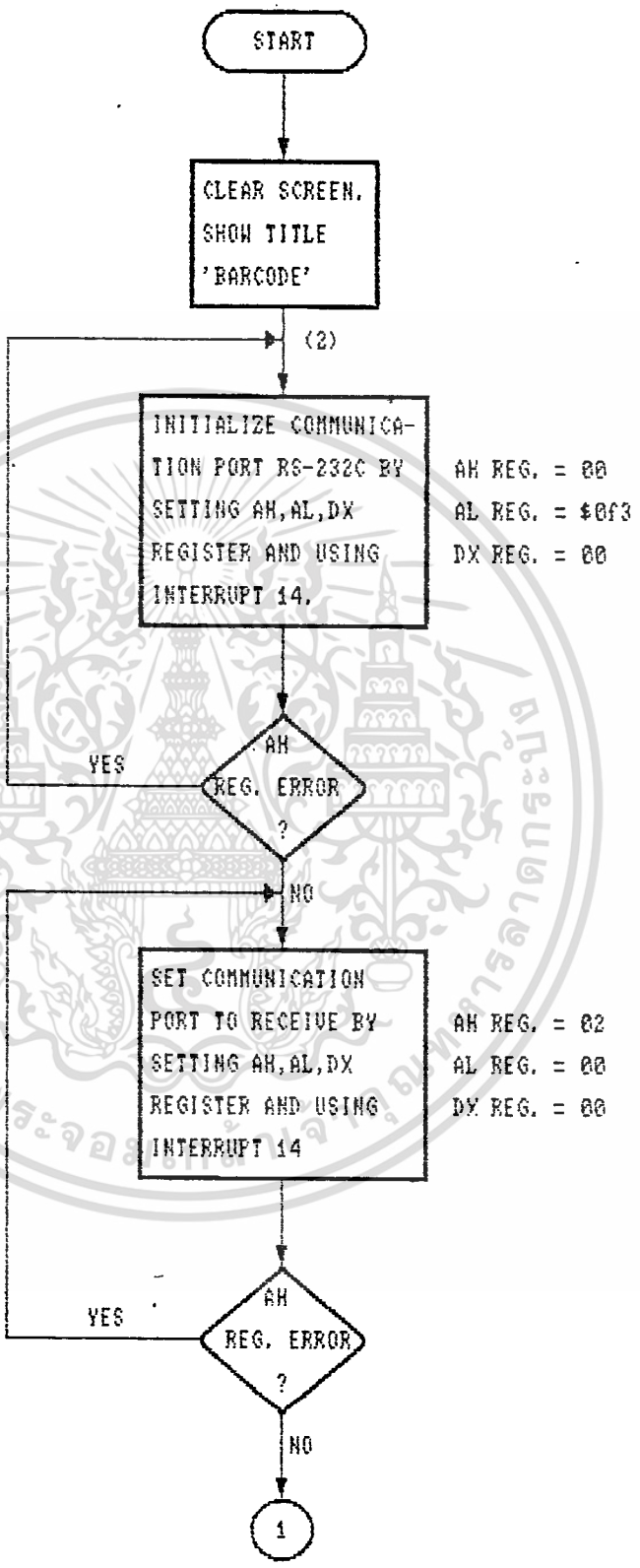
รีจิสเตอร์ AL = 2 เป็นการรับอักขระ(CHARACTER)ในซึ่งส่งผ่านมาจากพอร์ตสื่อสาร(COMMUNICATION PORT) AHจะไม่เป็นOเพียงเมื่อมีความผิดพลาด(ERROR)เกิดขึ้นเท่านั้น

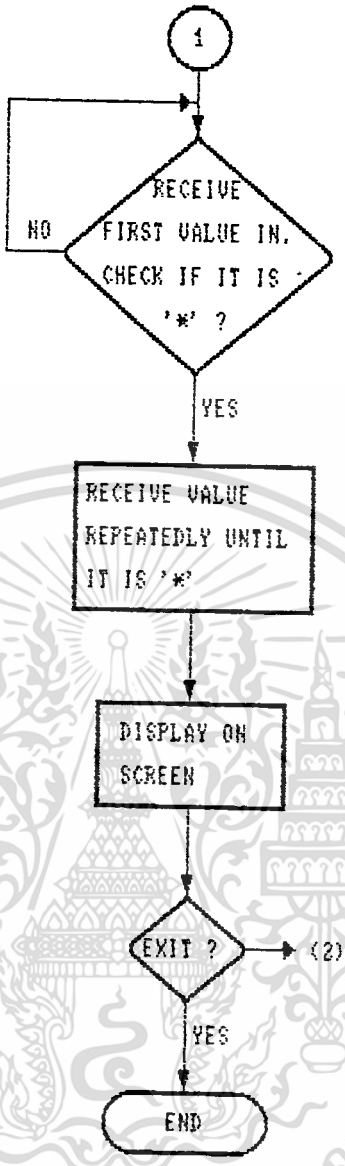
รีจิสเตอร์ AH = 3 จะคืนค่าสถานะของพอร์ทสื่อสารในรีจิสเตอร์AH

รีจิสเตอร์ DX ใช้เซทว่าใช้พอร์ทใด ถ้าเป็น 0 ใช้ พอร์ทคอม1(COM1) ถ้าเป็น 1ใช้พอร์ทคอม2(COM2)

ซึ่งได้แสดงโฟลวชาร์ต(FLOWCHART)ของโปรแกรมรับไว้ดังหน้าถัดไป

RECEIVE PROGRAM

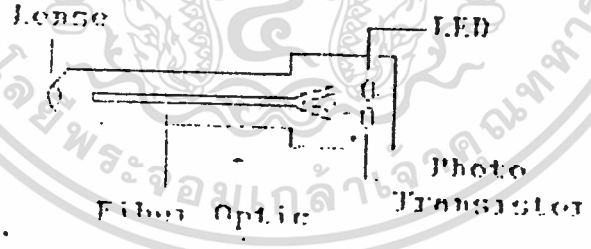




บทที่ 6

การออกแบบวงจรหัวอ่าน

หัวอ่านจะถูกใช้ในการสแกนผ่านรหัสแถบ แหล่งแสงจะส่งแสงผ่านเลนส์ไปกระทบแถบที่ถูกลากผ่าน ตัวตรวจจับแสงจะรอรับการสะท้อนของแสงกลับมา ในการออกแบบได้ใช้อุปกรณ์ส่วนนี้ จากหัวอ่าน National รุ่นVEQ 0542 ดังรูป 6.1 ซึ่งภายในมีเส้นใยนำแสงต่อเชื่อมระหว่าง แอลอีดี(LED) กับ โฟโตทรานซิสเตอร์(Photo transistor) โดยจะมีเลนส์รวมแสงอยู่ที่ปลายหัวอ่าน กำหนดให้แอลอีดีปล่อยแสงออกไปอย่างคงที่ ระยะห่างระหว่างรหัสที่จะสแกนกับหัวอ่านคงที่ ความเข้มของแสงที่จะสะท้อนกลับเข้ามาจะขึ้นอยู่ปัจจัยเพียงประการเดียว คือ คุณสมบัติการสะท้อนแสงของแถบ

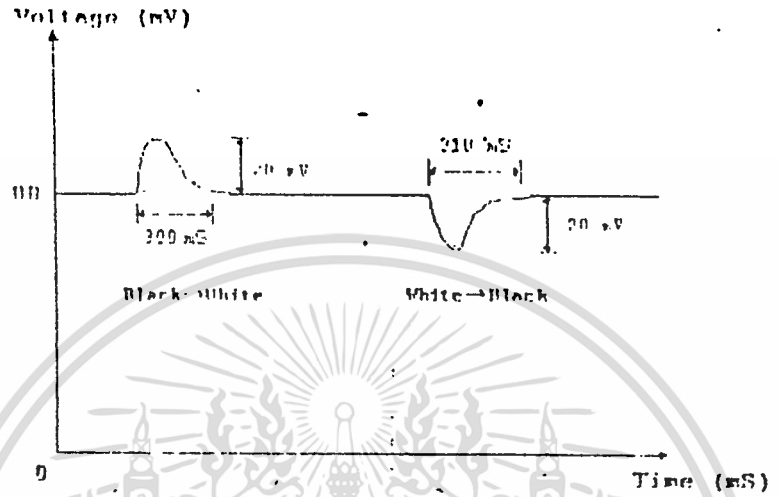


รูปที่ 6.1โครงสร้างของหัวอ่านรหัสแถบ

อย่างไรก็ตาม สัญญาณที่ได้จากโฟโตทรานซิสเตอร์ยังมีขนาดและลักษณะไม่เหมาะสมที่จะนำมาใช้งาน จึงต้องมีวงจรแปลงสัญญาณให้อยู่ในรูปของสัญญาณดิจิตอล ดังรูป6.2

สัญญาณที่ได้จากโฟโตทรานซิสเตอร์มีลักษณะพิเศษ คือ จะมีการเปลี่ยนแปลงของสัญญาณในลักษณะพัลส์เมื่อมีการเปลี่ยนจากดำเป็นขาวหรือ ขาวเป็นดำ โดยที่ตามปกติแล้วสัญญาณจะเป็นแรงดันไฟตรงขนาด 80mV ซึ่งค่านี้จะขึ้นอยู่กับการจัดไบอัสให้แก่ แอลอีดีและ

โฟโตทรานซิสเตอร์ ลักษณะของสัญญาณจะเป็นดังรูป 6.3



รูปที่ 6.3 ลักษณะของสัญญาณที่ออกจากโฟโตทรานซิสเตอร์

เนื่องให้การเชื่อมต่อกับส่วนควบคุมเป็นไปได้โดยสะดวก สัญญาณที่ออกจากวงจรส่วนหัวอ่านนี้ ควรจะเป็นสัญญาณดิจิตอลขนาด 0-5โวลต์ โดยที่ระดับของสัญญาณที่เวลาใดๆ ควรจะขึ้นอยู่กับสีของรหัส กำหนดให้ ค่า 5 โวลต์แทนแถบสีดำ ค่า 0โวลต์แทนแถบสีขาว ดังนั้น วงจรถูกแยกเป็น 3ส่วน คือ 1.ส่วนขยายสัญญาณ

2.ส่วนเปรียบเทียบ เพื่อตรวจจับการเปลี่ยนแปลงของสัญญาณ

3.ส่วนแลทช์สัญญาณ เพื่อคงสถานะของสัญญารไว้

ในส่วนของวงจรเปรียบเทียบสัญญาณ จะประกอบไปด้วยคอมพาราเตอร์ 2ตัว คอยทำหน้าที่ตรวจสอบระดับสัญญาณว่ามีการเปลี่ยนแปลงขึ้นหรือลง ดังนั้นคอมพาราเตอร์ทั้ง 2ตัว จะสลับกันทำงาน ตัวแรกจะคอยตรวจสอบว่าสัญญาณมีการเปลี่ยนจากขาวเป็นดำหรือไม่ ตัวที่2 ตรวจสอบการเปลี่ยนแปลงจากสีดำเป็นสีขาว สัญญาณที่ออกจากวงจรเปรียบเทียบทั้ง2สัญญาณนี้ จะนำไปป้อนเข้าฟลิปฟลอป เพื่อให้แลทช์ค่าเอาท์พุทไว้จนกว่าจะมีการเปลี่ยนแปลงของสัญญาณครั้งต่อไป โดยป้อนเข้าขาพรีเซท(Preset)และเคลียร์(Clear) ตามลำดับ

การทำงานของวงจรรูปที่ 6.2

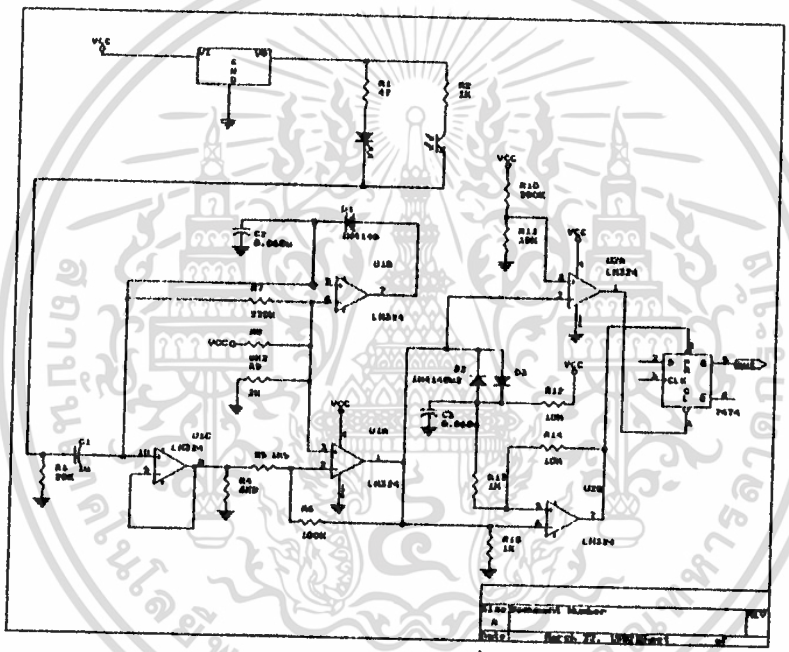
จากรูป 5.2 U_0 เป็นวงจรรักษาระดับแรงดัน ทำหน้าที่จ่ายแรงดันแก่ LED และ PHOTO TRANSISTOR โดยมี R_1, R_2 เป็นตัวจำกัดกระแส สัญญาณที่ได้จาก PHOTO TRANSISTOR จะถูกป้อนให้กับวงจรขยายสัญญาณ ซึ่งประกอบไปด้วย $U_{1A,B,C}$ LM324 ซึ่งเป็นออปแอมป์ 4ตัวในตัวถังเดียว (Quad OP AMP)

พิจารณา U_{1A} ต่อเป็นบัฟเฟอร์โดยที่ขาอินพุท (+) ของ U_{1C} จะต่อเข้ากับ U_{1B} ซึ่งทำหน้าที่ป้องกันการลดทอนของสัญญาณที่มาจากหัวอ่าน เนื่องจากการที่วงจรนี้ใช้ไฟเลี้ยงแบบSingle Supply ทำให้ที่ขาอินพุท(+) ของ U_{1B} ต้องต่อในลักษณะที่ตั้งแรงดันค่าเริ่มต้นไว้ในวงจรได้กำหนดค่านี้ไว้ประมาณ 0.95 โวลต์ และถ้าเราต้องการให้สัญญาณที่ขาอินพุท(+) ของวงจรมีการเปลี่ยนแปลงตามสัญญาณจากหัวอ่าน ก็จะต้องต่อขาอินพุท(+)ของ U_{1C} ไปยังขาอินพุท(+)ของ U_{1A} แต่ที่ขานี้จะมีการตั้งค่าแรงดันไว้แล้ว จึงทำให้สัญญาณที่จะเข้ามาแล้วผ่านออกไปจาก U_{1C} จะต้องมีแรงดันสูงกว่าค่าที่ตั้งไว้ ดังนั้นจึงได้มีการต่อวงจรของ U_{1B} เข้ามา ซึ่งทำให้ผลนี้หายไป แต่สัญญาณก็จะยังเปลี่ยนแปลงตามสัญญาณที่มาจากหัวอ่าน โดยการต่อคร่อมไว้ด้วย R_7 220k แล้วต่อจากขาอินพุทจากหัวอ่านไปยังขาอินพุท(-)ของ U_{1B} ซึ่งต่อไว้ด้วย C_2 0.068uF (ค่านี้จะน้อยกว่านี้ได้ แต่ต้องไม่ต่ำเกินไป เพราะทำให้รูปลักษณะสัญญาณมีความชันมากเกินไป และค่านี้จะต้องไม่มากไปกว่านี้ เพราะจะทำให้สัญญาณเปลี่ยนแปลงตามสัญญาณเข้าไม่ทัน) ซึ่งจะทำงานในลักษณะที่เปลี่ยนแปลง คือเก็บประจุ และคายประจุ ตามสัญญาณที่ได้จากหัวอ่าน โดยการเก็บประจุจะทำผ่านทางไดโอด D_1 ส่วนการคายประจุ ทำผ่านทาง C_1 1uF และ R_3 20kเพราะที่จุดนี้จะมีแรงดันคร่อมต่ำกว่าเสมอเมื่อดูเทียบกับการคายประจุผ่านทาง R_7 220kไปยังขาอินพุท(+)ของU_{1B} ซึ่งที่จุดนี้จะมีแรงดันตกคร่อมใกล้เคียงกัน การทำในลักษณะนี้จะทำให้ขาอินพุท(+)ของ U_{1C} เสมือนถูกยกระดับแรงดันด้วยค่าแรงดันที่เท่ากับที่ขาอินพุท(+)ของ U_{1A} จึงทำให้เอาท์พุทจาก U_{1A} เป็นสัญญาณที่เกิดจากการเปลี่ยนแปลงของสัญญาณจากการรูดหัวอ่านผ่านรหัสแถบ (ซึ่งถ้าเป็นวงจรแบบที่ไม่มี U_{1B} แล้ว สัญญาณเอาท์พุทที่ออกมาจะมีค่าสูงกว่า 0.95โวลท์) โดยมี R_6 เป็นตัวควบคุมอัตราการขยายสัญญาณ สัญญาณที่ออกจาก U_{1A} จะส่งผ่านต่อไปยังส่วนของวงจรเปรียบเทียบต่อไป

วงจรส่วนเปรียบเทียบสัญญาณประกอบด้วยคอมพาราเตอร์ 2ตัว คือ U_{2A} และ U_{2B} ซึ่งจะถูกกำหนดแรงดันอ้างอิงไว้ต่างกัน ไดโอด D_2 และ D_3 ทำหน้าที่ป้องกันไม่ให้สัญญาณที่เข้ายังขาอินพุทของคอมาราเตอร์ทั้งสองมีค่าแตกต่างกันมากกว่า 0.2โวลท์ พิจารณาในส่วนของคอมพาราเตอร์ U_{2B} จะเห็นว่า มีการต่อวงจรให้เอาท์พุทมีค่าเป็น 0โวลท์ เมื่อสัญญาณที่มาจากส่วนขยายเข้าที่ขาอินพุท(-) มีค่าสูงกว่าค่าแรงดันอ้างอิงที่กำหนดไว้ซึ่งจะเกิดกรณีนี้ได้ก็ต่อเมื่อหัวอ่านมีการเคลื่อนที่ผ่านแถบขาวไปเป็นดำ สัญญาณเอาท์พุทที่ได้จะมีลักษณะเป็นพัลส์ซึ่งควบคุมความกว้างโดย C_9 0.068uF และถูกส่งไปยังวงจรส่วนแลทช์ต่อไป

ในส่วนของคอมพาราเตอร์ U_{2A} จะเป็นลักษณะการต่อแบบพื้นฐาน โดยที่มี R_{10} และ R_{11} เป็นตัวกำหนดแรงดันอ้างอิง เอาท์พุทของ U_{2A} จะเป็น 0โวลท์ ก็ต่อเมื่อสัญญาณจากวงจรขยายมีค่าต่ำกว่าแรงดันอ้างอิง ซึ่งจะเกิดกรณีนี้ได้ก็ต่อเมื่อหัวอ่านมีการเคลื่อนที่ผ่านจากแถบดำไปเป็นขาว สัญญาณเอาท์พุทจะถูกส่งไปยังวงจรส่วนแลทช์ต่อไป

วงจรแลทช์ในที่นี้ใช้ U_{3A} ซึ่งเป็นฟลิปฟลอปแบบดี โดยนำสัญญาณจาก U_{2B} มาป้อนเข้าที่ขาเคลียร์ ซึ่งทำงานแบบ Active Low ดังนั้น เมื่อแถบที่ถูกสแกนเปลี่ยนจากขาวเป็นดำ จะทำให้ฟลิปฟลอปถูกเคลียร์ให้เอาท์พุทเป็น 0โวลท์ และจะคงค่าไปเช่นนั้นเรื่อยๆ จนกว่าจะมีสัญญาณจาก ซึ่งมาป้อนเข้าที่ขาพรีเซท (Preset) เกิดเนื่องจากแถบรหัสที่ถูกสแกนเปลี่ยนจากสีดำเป็นขาว จะทำให้ฟลิปฟลอปให้เอาท์พุทออกมาเป็น 5โวลท์คงที่จนกว่าจะมีสัญญาณเคลียร์เข้ามาอีกครั้งหนึ่ง จะเห็นได้ว่า ลักษณะของสัญญาณที่ออกมาจากเอาท์พุทของฟลิปฟลอปตรงกับลักษณะสัญญาณที่ต้องการ



บทที่ 7

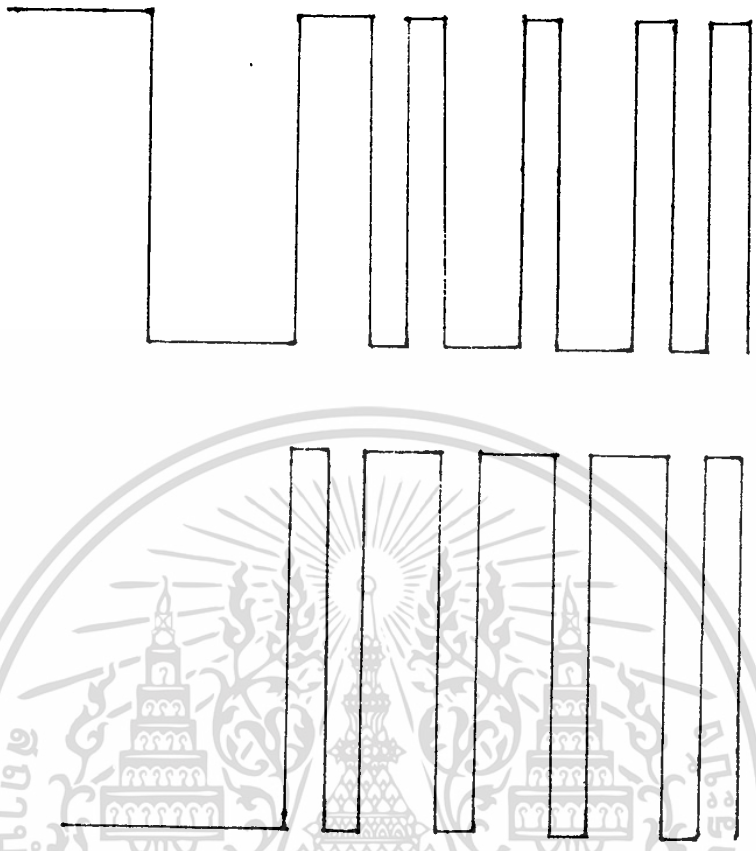
การออกแบบโปรแกรมถอดรหัส

หลักการ

เลือกชนิดของรหัสแถบ เป็นแบบ 3ใน9 อาศัยโครงสร้างของแถบที่ประกอบด้วยแถบขาวที่กว้างและแคบ แถบดำที่กว้างและแคบ นำมาจัดเรียงต่อกันสลับขาว ดำ เริ่มต้นด้วยดำ ไม่ว่าจะขาวหรือดำ ถ้าคิดเทียบเป็นค่าทางลอจิก ให้พิจารณาแถบที่กว้างเป็น 1 และแถบที่แคบเป็น 0 ที่เรียกว่าเป็นรหัส 3ใน9 ก็เพราะว่า 1 อักขระประกอบด้วยแถบดำ 5แถบ แถบขาว 4แถบ รวมทั้งหมด 9แถบแล้วต้องมีแถบกว้างทั้งหมด 3แถบ ดังนั้นในส่วนของซอฟท์แวร์ที่ใช้ตัวจับเวลา (Timer) นับค่าของแถบที่อ่านเข้ามาได้ จะต้องทำการนับในลักษณะเดียวกันไม่ว่าแถบนั้นจะเป็นขาวหรือดำ เนื่องจากหัวอ่านเป็นแบบแฮนด์เฮล(Hand Held) อัตราเร็วของการลากจึงเป็นสิ่งที่ต้องนำมาพิจารณาประกอบกับการสร้างซอฟท์แวร์ ทำให้สังเกตพบว่า การเปรียบเทียบความกว้างแคบของสัญญาณที่อ่านได้ควรกระทำเพียงครั้งละ 1โมดูล (Module) (1 Module = อักขระ) เพื่อลดข้อผิดพลาดที่เกิดจากผู้ใช้ ในการที่จะกำหนดให้แถบใดกว้างหรือแคบต้องพิจารณาเปรียบเทียบกับค่ากำหนด (Threshold) ค่าหนึ่งของแต่ละโมดูล การสร้างค่ากำหนดก็ได้มาจากแถบกว้างแคบในโมดูลที่พิจารณาอยู่นั่นเอง โดยเปรียบเทียบทั้ง 9แถบด้วยกันเอง แล้วเลือกแถบที่กว้างที่สุดกับแถบที่แคบที่สุด นำค่าทั้งสองนี้มาเฉลี่ย แถบใดที่ถูกอ่านได้มีค่ามากกว่าค่ากำหนดก็ให้แถบนั้นแทนลอจิก 1 ถ้าน้อยกว่าให้แทนลอจิก 0 ถึงตอนนี้เราก็จะสามารถแปลงจากแถบขาวดำมาเป็นรหัสในรูปแบบเลขฐานสอง ซึ่งสามารถแปลงเป็นรหัสแอสกี้ได้ง่าย ที่ต้องทำเช่นนี้ก็เพื่อการประมวลผลบนคอมพิวเตอร์ โดยผ่านข้อมูลผ่านสาย RS 232

รายละเอียดของโปรแกรม

หัวอ่านที่พร้อมทำงาน ขณะลอยอยู่หรือวางไว้เฉยๆ เอาท์พุทจะเป็นลอจิก 1 เมื่อพบกับพื้นกระดาษว่างๆ สัญญาณจะเปลี่ยนเป็น 0 ดังนั้นสัญญาณเริ่มต้นที่ตรวจจับด้วยสตอเรจ สโคป (Storage Scope) จะเป็นดังนี้



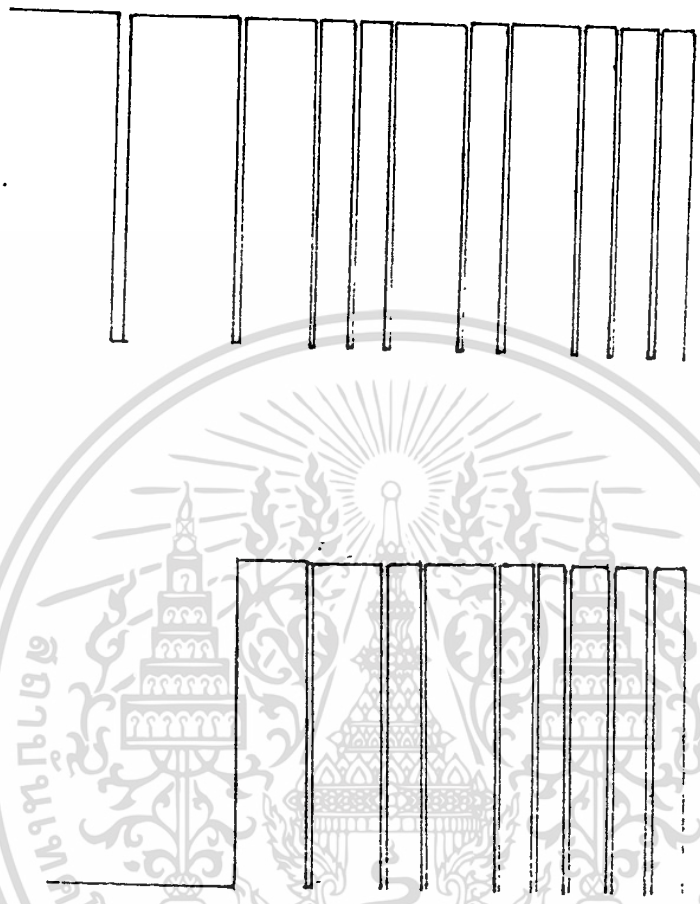
รูปที่ 7.1 แสดงสัญญาณเริ่มต้น

จากการเซทโหมดต่างๆ ตัวจับเวลาที่ 1 โหมด1 เป็นการควบคุมภายใน

" 0 โหมด1 " ภายนอก

ตัวอินเทอร์รัพท์ 0 " แบบทริกขอบ(edge trig)

เราจะใช้ตัวจับเวลา0 ในการนับความกว้างของแถบ ทำให้ต้องแปลงสัญญาณที่อ่านได้ให้เป็นคัง รูปที่ 7.2 เพื่อให้ตัวจับเวลานับค่าได้ นอกจากนี้ยังต้องมีการชดเชยเวลาเนื่องจากการเปลี่ยนสถานะของสัญญาณหัวอ่าน ด้วยการตั้งค่าฐานในการนับที่ตัวจับเวลา0



รูปที่ 7.2 แสดงการแปลงสัญญาณ

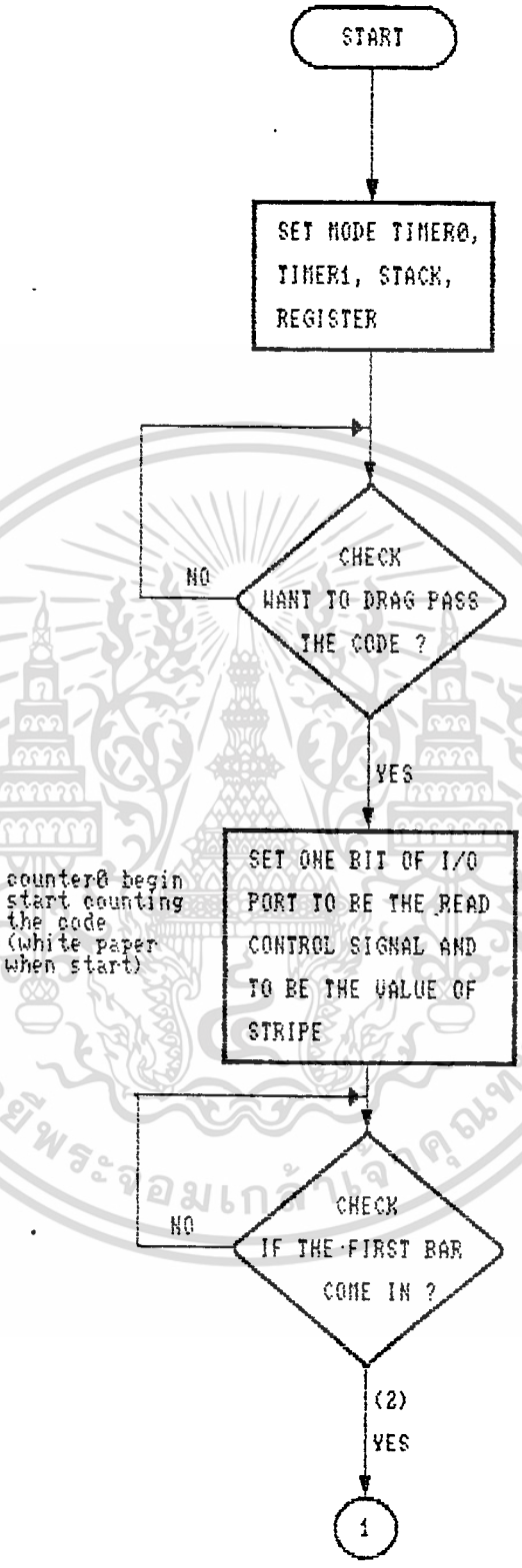
เพื่อให้แน่ใจว่าจะมีการใช้งานจริง จึงต้องมีการตรวจสอบพื้นที่ว่าง(Quiet Zone)ก่อนจะถึงแถบดำที่ 1โดยใช้ตัวจับเวลา0 การตรวจสอบค่าเวลาของแถบย่อยๆก็จำเป็น เพราะอาจจะเกิดการกระตุก หรือการลากที่เร็วเกินไปในบางช่วง หน้าที่นี้จะเป็นของตัวจับเวลา1

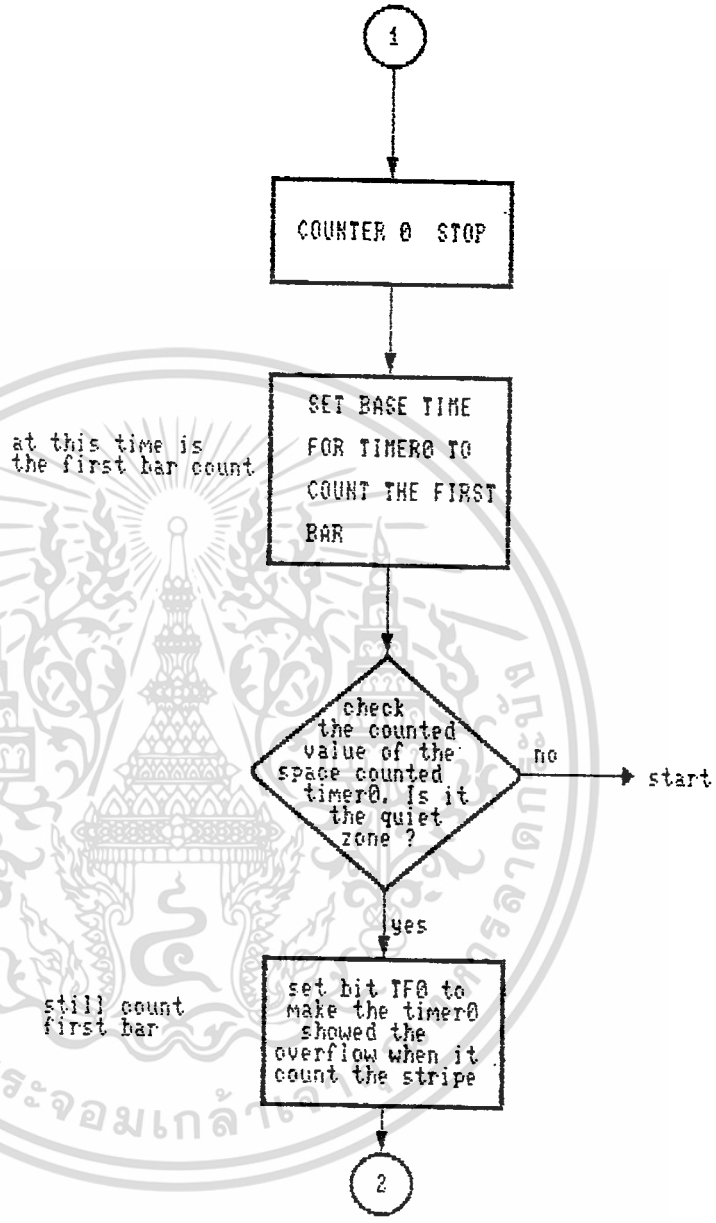
การตั้งค่าฐานในการนับของตัวจับเวลา0 จะพิจารณาจากค่าเวลาของแถบย่อย ซึ่งจะต้องน้อยกว่า 70mS มิฉะนั้นแล้วตัวจับเวลา0จะโอเวอร์โฟลว์ (Overflow) ทั้งนี้เป็นประโยชน์สำหรับการเข้าสู่โปรแกรมถอดรหัส ตัวจับเวลา1จึงตั้งค่าฐานไว้ที่ค่าเดียวกันสำหรับการตรวจสอบขนาดของแถบย่อย ซึ่งความกว้างอย่างน้อยที่สุดจะต้องเท่ากับค่าที่จะใช้แปลงเป็นค่าฐาน ถ้าเล็กกว่านี้เท่าไร การผิดพลาดที่จะเกิดจาก ตัวจับเวลา 1 ก็จะเกิดได้มากเท่านั้น

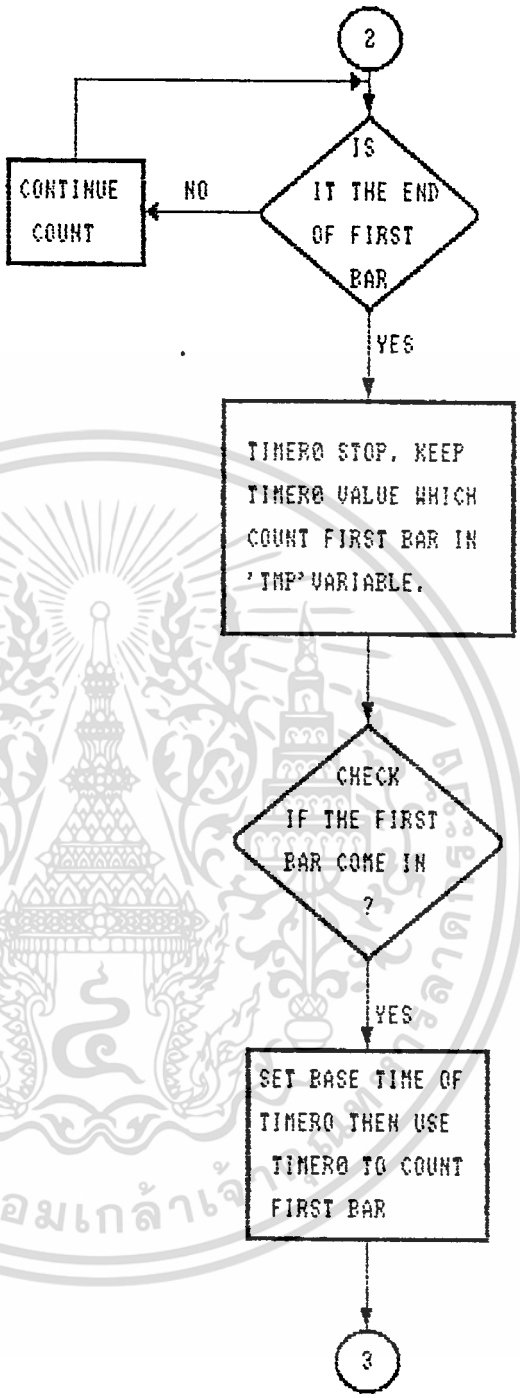
การโอเวอร์โฟลว์จากทั้งตัวจับเวลา0และ1 จะเป็นตัวเปิดไปสู่การถอดรหัสของสัญญาณที่ถูกนับ

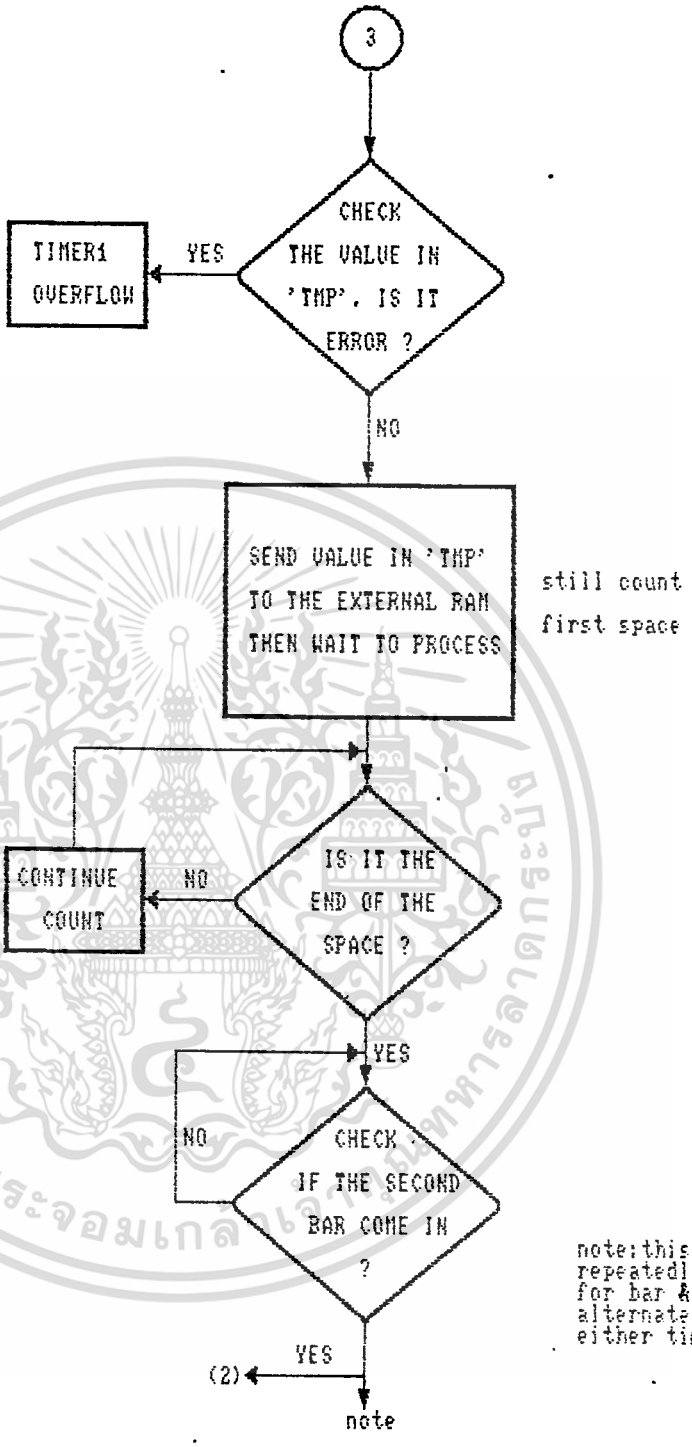
สำหรับส่วนการถอดรหัส จะเป็นการเปรียบเทียบค่าที่นับได้ของแถบย่อยภายใน 1โมดูล คิดแยกแถบดำกับแถบขาวออกจากกัน เป็นแถบดำที่แคบที่สุด กว้างที่สุด แถบขาวที่กว้างที่สุด แคบที่สุด ค่ากำหนดก็คิดแยกกัน จากนั้นทำการเปรียบเทียบระหว่างแถบกับค่ากำหนดภายใน 1โมดูลเพื่อแปลงให้เป็นลอจิก 0 และ 1ตามระบบของการเข้ารหัส 3ใน9 เราสามารถจะแปลงจากลอจิกเหล่านี้เป็นแอสกี้ โดยการสร้างตารางการแปลงขึ้นเอง จะมีอักขระเพียงตัวเดียวเท่านั้นที่จะตรงกับรหัสแถบที่อ่านได้ จากข้อบังคับของรหัสชนิดนี้ จะต้องมีรหัสเริ่มต้น และรหัสสิ้นสุด คือเครื่องหมาย * ดังนั้นการแปลงเป็นแอสกี้จึงต้องตรวจสอบ *เริ่มต้น ก่อนที่จะแปลงอักขระต่อไป และสิ้นสุดการแปลงด้วย * เช่นกัน

THE METHOD TO COUNT THE CODE

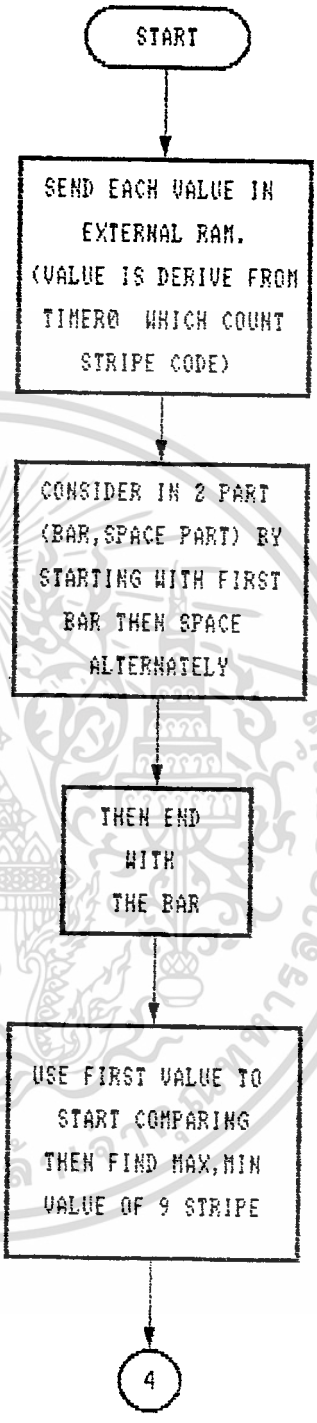


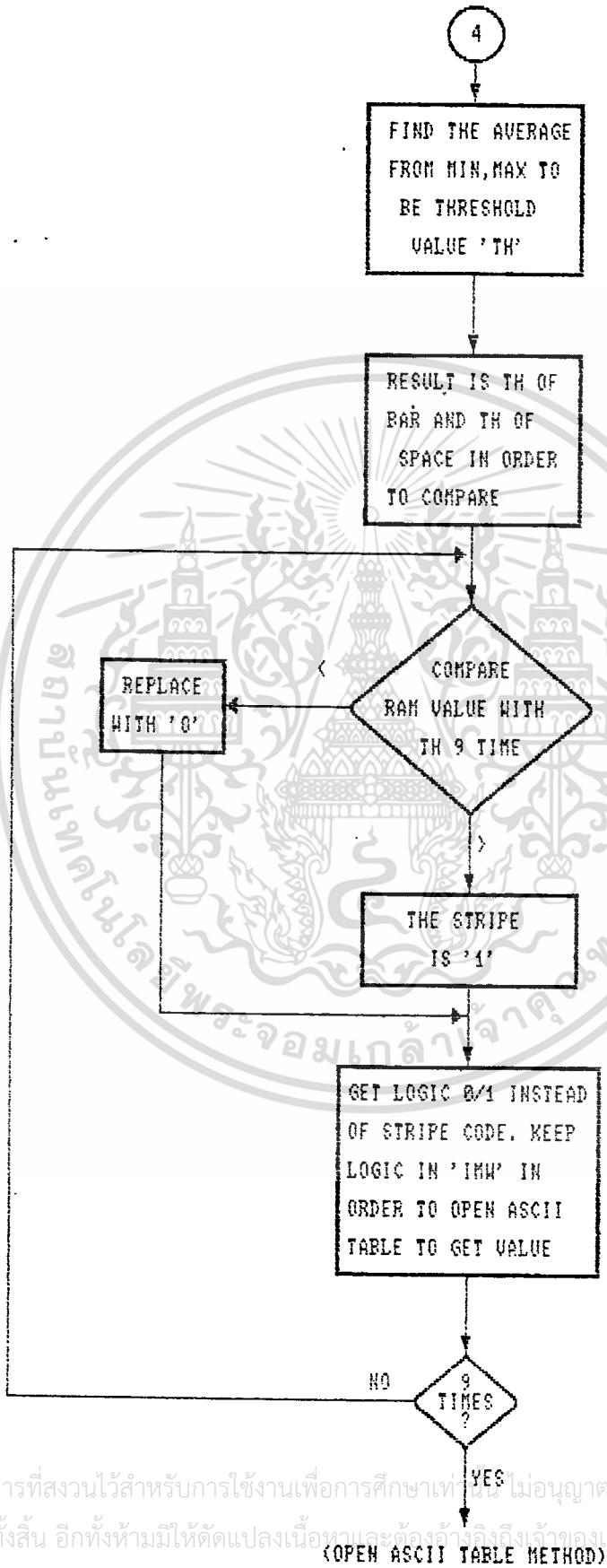


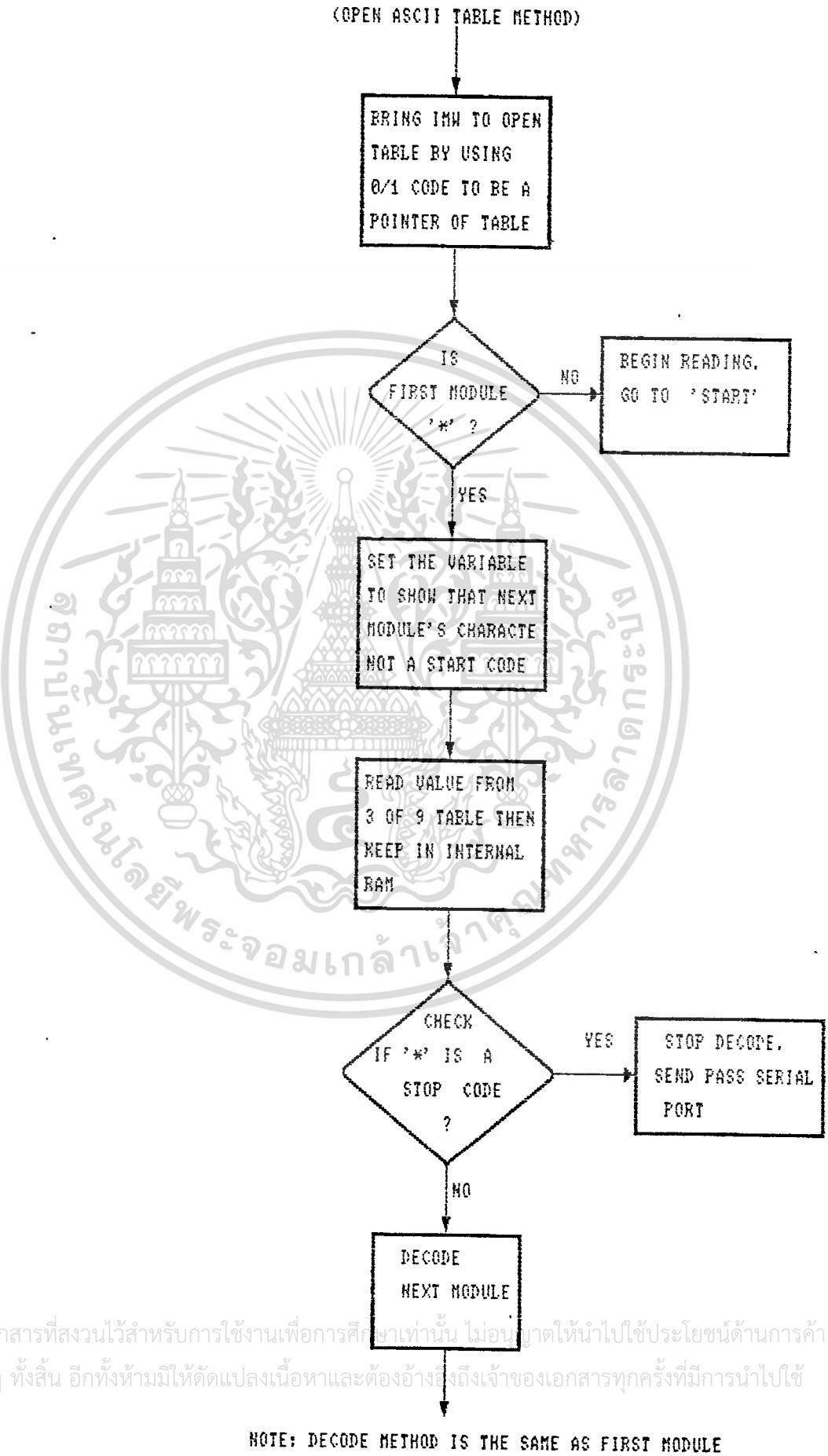




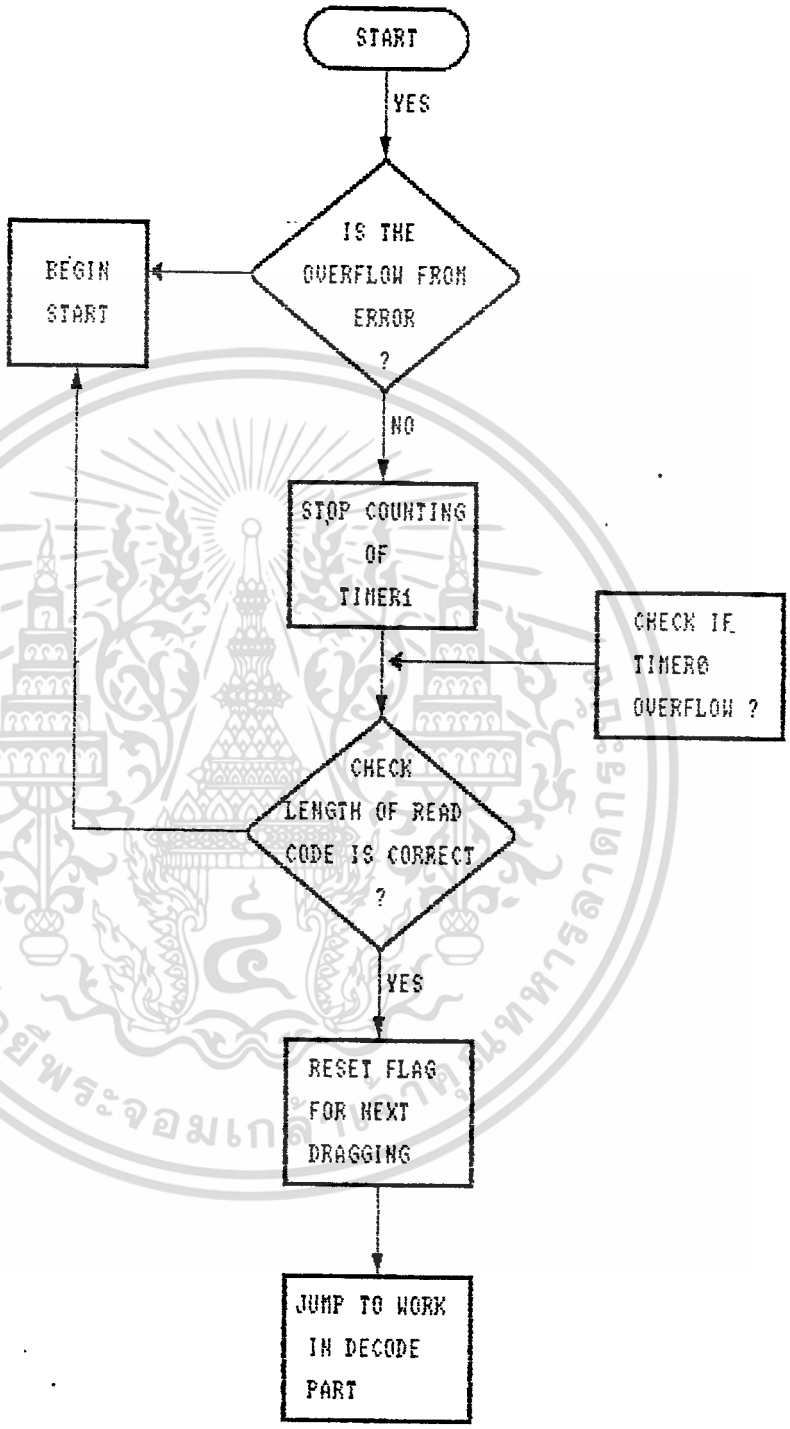
DECODE METHOD







INTERRUPT SERVICE ROUTINE



บทที่ 8

ผลการทดลอง บทสรุปและวิจารณ์

ผลการทดลอง

ได้ทำการวัดความเร็วที่ใช้ในการลากหัวอ่านมือถือแบบ WAND กับรหัสแถบชนิด3OF9 ความยาวต่างๆดังนี้ โดยพิจารณาที่ ณ ความเร็วที่ยังสามารถถอดความหมายของรหัสนั้นได้ ดังนี้

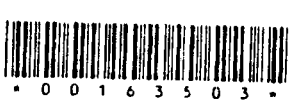
1.ความยาว 1 อักขระ = 0.4ซ.ม.

การลากอย่างช้า = 128 มิลลิเมตรต่อวินาที(mm/S)

การลากอย่างเร็ว = 338 มิลลิเมตรต่อวินาที(mm/S)

การลากโดยเฉลี่ย = 205 มิลลิเมตรต่อวินาที(mm/S)

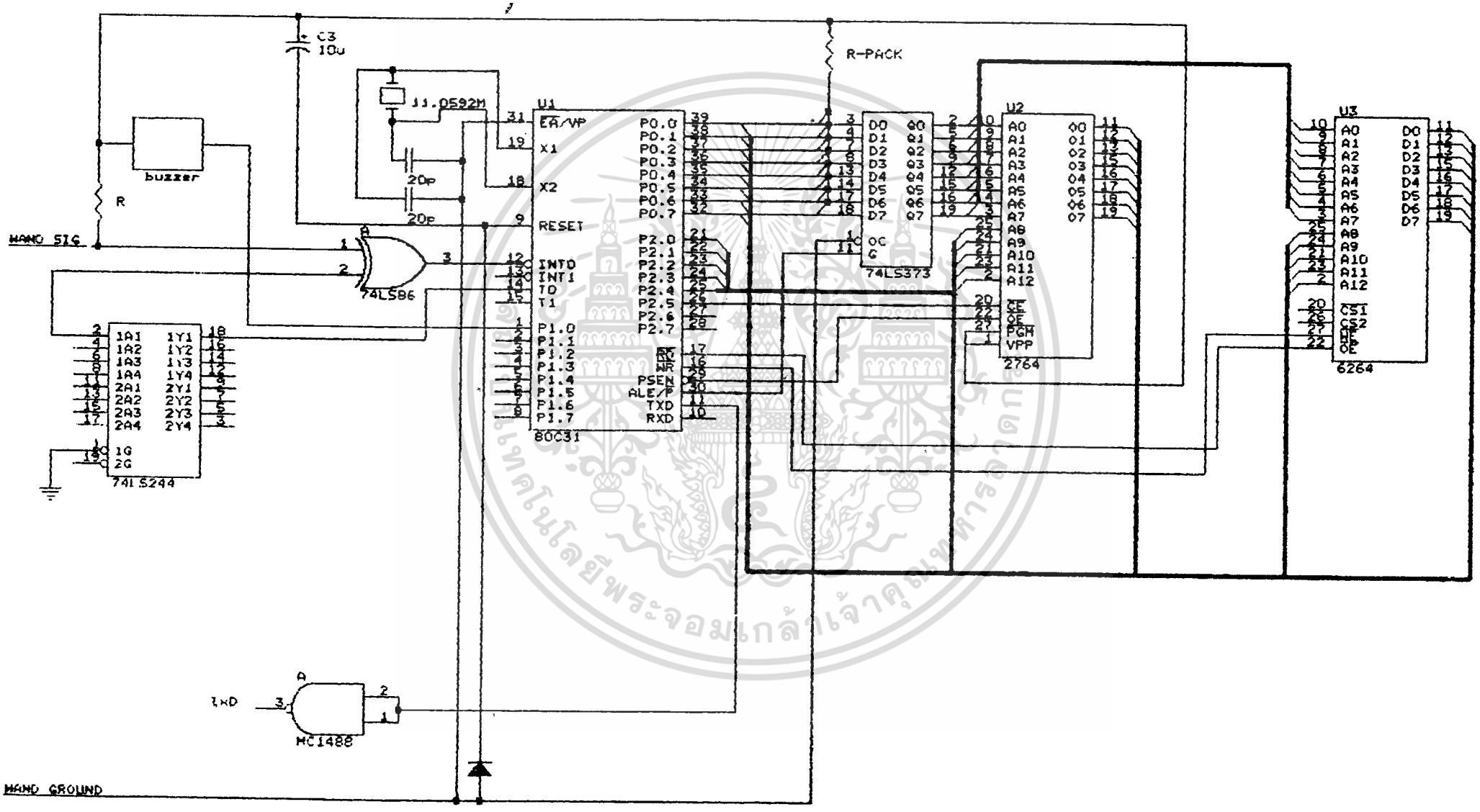
2.ความยาว 1 อักขระ = 0.3ซ.ม.



การลากอย่างช้า = 84 มิลลิเมตรต่อวินาที

การลากอย่างเร็ว = 191 มิลลิเมตรต่อวินาที

การลากโดยเฉลี่ย = 133 มิลลิเมตรต่อวินาที


C3
10u
buzzer
R
HAND SIG
11.0592M
20p
20p
U1
EA/VP
X1
X2
RESET
INT0
INT1
T0
T1
P1.0
P1.1
P1.2
P1.3
P1.4
P1.5
P1.6
P1.7
P0.0
P0.1
P0.2
P0.3
P0.4
P0.5
P0.6
P0.7
P2.0
P2.1
P2.2
P2.3
P2.4
P2.5
P2.6
P2.7
RD
WR
PSEN
ALE/P
TXD
RXD
80C31
74LS86
R-PACK
74LS373
U2
2764
U3
6264
74LS244
MC1488
TXD
HAND GROUND

กิตติกรรมประกาศ

ขอกราบขอบพระคุณท่านอาจารย์รศ.ดร มนัส สังวรศิลป์และอาจารย์สุรพันธ์ เอื้อไพบูลย์ ที่ได้ประสิทธิ์ประสาทวิชาความรู้ ตลอดจนให้คำปรึกษาแนะแนวทางและวิธีการแก้ไขปัญหาต่างๆ รวมทั้งขอกราบขอบพระคุณอาจารย์ทุกท่านที่ให้ความช่วยเหลือ ขอขอบคุณเพื่อนๆทุกคนที่ให้กำลังใจ และให้ความช่วยเหลือในทุกๆด้าน จนทำให้ปริญญานิพนธ์ฉบับนี้สำเร็จลุล่วงไปด้วยดี

ผู้จัดทำ

หนังสืออ้างอิง

1.ROGER C.PALMER,THE BARCODE

1989 HELMERS PUBLISHING INC.

2.IBM,TECHNICAL REFFERENCE

INTERNATIONAL BUSINESS MACHINES CORPORATION,1981,1982,1983

3.MICROPROCESSOR DATA BOOK MCS-51 MICROCONTROLLERS

4.นิตยสารไมโครคอมพิวเตอร์ ฉบับที่42 พ.ศ.2533

สำนักพิมพ์ซีเอ็ดยูเคชั่น