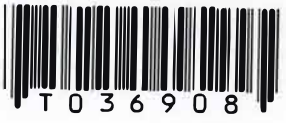
แสดงตัวอักษรควบคุมโดยคอมพิวเตอร์

Projector used PC control



โดย

นาย ภาณุ เกตุแก้ว

นาย อุดม ภูห้องเพชร

ปริญญานิพนธ์นี้ เป็นส่วนหนึ่งของการศึกษาตามหลักสูตร อุตสาหกรรมศาสตร์บัณฑิต

สาขาวิชาเทคโนโลยีอิเล็กทรอนิกส์

ภาควิชาเทคนิคอุตสาหกรรม

คณะวิศวกรรมศาสตร์

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

ปีการศึกษา 2542

หัวข้อปริญญานิพนธ์	แสดงตัวอักษรควบคุมโดยคอมพิวเตอร์
โดย	นาย ภาณุ เกตุแก้ว
	นาย อุดม ภูห้องเพชร
อาจารย์ที่ปรึกษา	อาจารย์ มยุรี เลิศเวชกุล
	อาจารย์ บุณย์ชนะ ภู่ระหงษ์
ภาควิชา	เทคนิคอุตสาหกรรม
ปีการศึกษา	2542

คณะวิศวกรรมศาสตร์ สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง อนุมัติให้นับปริญญานิพนธ์ฉบับนี้ เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาอุตสาหกรรมศาสตร์บัณฑิต

คณะกรรมการสอบปริญญานิพนธ์

.....ประธานกรรมการ
( )
.....กรรมการ
( )
.....กรรมการ
( )
.....กรรมการ
( )
.....กรรมการ
( )

ลิขสิทธิ์ของคณะวิศวกรรมศาสตร์
สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

แสดงตัวอักษรควบคุมโดยคอมพิวเตอร์

นาย ภาณุ เกตุแก้ว
นาย อุดม ภูห้องเพชร

อาจารย์ที่ปรึกษา
อ.มยุรี เลิศเวชกุล
อ. บุณย์ชนะ ภู่ระหงษ์

บทคัดย่อ

ปริญญานิพนธ์นี้ได้มีการนำเอาคอมพิวเตอร์มาใช้งาน เพื่อควบคุมการสแกนของแสงเลเซอร์ให้เกิดเป็นตัวอักษรขึ้นบนฉาก ซึ่งสามารถแสดงข้อความยาวๆ ได้อย่างต่อเนื่อง จะเป็นประโยชน์อย่างยิ่งในการโฆษณาสินค้าต่างๆ ที่มีการแข่งขันกันอย่างมากมายในปัจจุบัน และโครงงานที่สร้างขึ้นมีขนาดเล็กทำให้สามารถเคลื่อนย้ายได้สะดวก

ปริญญานิพนธ์นี้เป็นการสร้างเครื่องฉายข้อความโดยอาศัยแสงเลเซอร์เป็นตัวฉาย ซึ่งประกอบด้วยส่วนที่เป็นเครื่องกลไก, ส่วนกำเนิดแสงเลเซอร์ซึ่งใช้เลเซอร์ไดโอดจำนวน 8 ตัวด้วยกัน, ส่วนของบอร์ดไมโครคอนโทรลเลอร์ และส่วนของโปรแกรมใช้งาน การแสดงตัวอักษรจะสามารถแสดงได้ครั้งละ 8 ตัวอักษร สามารถควบคุมลักษณะการสแกนได้

Projector used PC control

MR. PHANU KETKEAW
MR. UDOM PHUHONGPET

ADVISER
MS. MAYULEE LIRDWETCHAKUL
MR. BOONCHANA PURAHONG

ABSTRACT

Computer system was used in preparation of this thesis to control laser scaned before projected several characters on the blackground, that can be project the continuous word. Thus it had the most useful for promoted the goods. This projector has the small size, it can easily move to any location.

This thesis made projector for projects the word by laser diode applied. It has each part that is mechanical part, 8-laser beam producted part, microcontroled part, and used program part. It can be projected 8 character per one scaned and can controled its products.

กิตติกรรมประกาศ

โครงงานนี้จะไม่สำเร็จลงได้ถ้าไม่ได้คำแนะนำและความช่วยเหลือจากบุคคลดังต่อไปนี้ อ.มยุรี เลิศเวชกุล และอ.บุญชนะ ภู่ระหงษ์ ที่ให้คำปรึกษามาโดยตลอด คุณพ่อและคุณแม่ที่ได้ส่งเสริมและคอยเป็นกำลังใจทำให้มาจนถึงวันนี้ได้ เพื่อนๆทุกคน และบุคคลอีกหลายๆท่านที่คอยเป็นกำลังใจและสุดท้าย กองทุนเงินกู้ยืมเงินเพื่อการศึกษาที่ให้กู้เงินเพื่อใช้ในการศึกษา

คณะผู้จัดทำ

สารบัญ

สารบัญรูปภาพ

หน้า

สารบัญตาราง

บทที่ 1
บทนำ

1.1 ความเป็นมา

เนื่องจากในปัจจุบันนี้มีการแข่งขันทางด้านธุรกิจกันอย่างกว้างขวาง สิ่งหนึ่งที่เป็นปัจจัยสำคัญที่ควบคู่การเจริญเติบโตของธุรกิจ นั่นคือการโฆษณา ยิ่งมีการแข่งขันกันมากเท่าใด การโฆษณาก็ยิ่งเป็นสิ่งจำเป็นมากตามตัว เพราะการโฆษณาจะทำให้กลุ่มเป้าหมายหรือผู้ใช้บริการรู้จักสินค้าหรือผลิตภัณฑ์นั้นมากขึ้น โครงงานนี้จะเป็นประโยชน์อย่างยิ่งสำหรับการโฆษณาสินค้า เพื่อดึงดูดลูกค้า เพราะโครงงานนี้เป็นสิ่งที่แปลกใหม่ และมีความสะดวกในการเคลื่อนย้ายไปในที่ต่างๆ

1.2 วัตถุประสงค์

การโฆษณาที่แปลกใหม่และทันสมัยเป็นสิ่งที่สามารถดึงดูดความสนใจจากผู้พบเห็นได้เป็นอย่างดี ผู้เขียนจึงได้คิดค้นโครงงานนี้ขึ้นมา โดยใช้อุปกรณ์ต่างๆ ที่มีขายทั่วไปและราคาไม่แพงจนเกินไป และสามารถเคลื่อนย้ายไปมาได้สะดวก

1.3 ขอบเขตของปริญญานิพนธ์

เครื่องมือที่สร้างขึ้นมานี้ เป็นการสร้างเครื่องที่ใช้ประโยชน์ในด้านการเป็นสื่อโฆษณา โดยจะใช้แทนแบบเดิมที่ใช้ LED ที่มีจำนวนหลายหลอด และมีราคาแพง เครื่องมือชิ้นนี้สามารถที่จะเก็บข้อมูลตัวอักษรอย่างน้อย 1000 ตัวอักษร ภายในหน่วยความจำขนาด 32 กิโลไบท์ ซึ่งขึ้นอยู่กับลักษณะของตัวอักษรที่ได้ป้อนให้ ซึ่งลักษณะตัวอักษรนั้นจะได้จากการออกแบบโดยซอร์ฟแวร์

บทที่2

ทฤษฎีและหลักการ

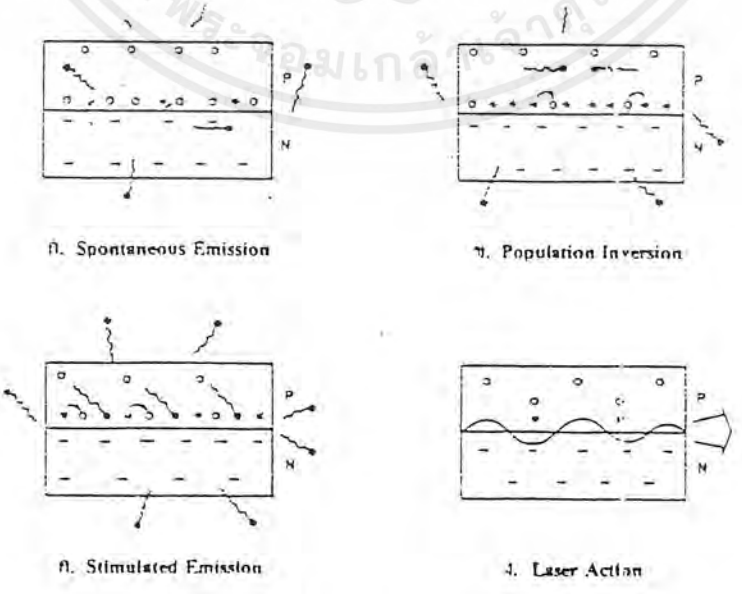
2.1 เลเซอร์ไดโอด (LASER DIODE: LD)

เลเซอร์ไดโอด ก็คือ ไดโอดมีขนาดประมาณเท่าตัวต้านทาน 1-2 วัตต์ หลักการทำงานก็คล้ายกับ LED ต่างกันที่ตรงที่ เมื่อมีการไบแอสถูกวิธีจะเปล่งแสงออกมาเป็นแสงเลเซอร์กำลังงานต่ำ แทนที่จะเป็นแสงธรรมดาอย่าง LED ทั่วไป การใช้งานทั่วไปในปัจจุบันมักใช้ในการอ่านข้อมูลทางดิจิตอล เช่น ในเครื่องเสียงชนิดคอมแพคดิสค์ หรือใช้เป็นเครื่องวัดระยะทางในการทหาร เพื่อหาระยะเป้าหมายของข้าศึกได้ หรือใช้ชี้ตำแหน่งบนบอร์ดในการสัมนาต่าง ๆ เป็นต้น

โครงสร้างและหลักการและการเกิดแสง

โดยทั่วไปเลเซอร์ไดโอดจะทำด้วยสารกึ่งตัวนำพวก GaAs (gallium arsenidde) ทำเป็นรอยต่อ P-N เหมือนไดโอดทั่วไป รวมทั้งคุณสมบัติทางไฟฟ้า โดยแสงเลเซอร์ส่วนใหญ่ที่เปล่งออกจะเป็นแสงย่านอินฟาเรดซึ่งตามองไม่เห็น หลักการในการเกิดแสงเลเซอร์เป็นดังรูปที่ 2.1

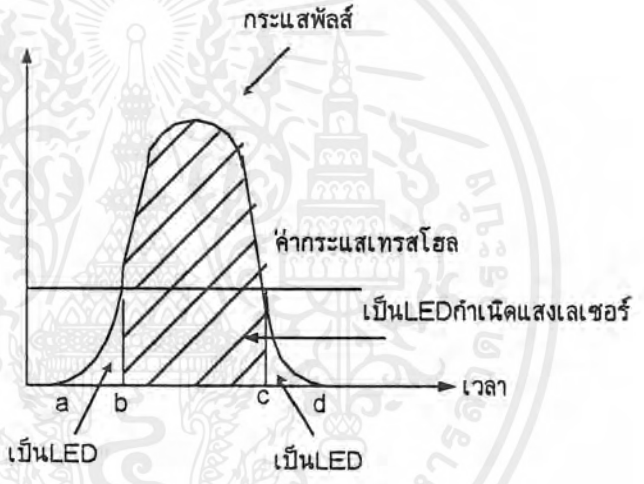
เริ่มจากรูป ก. เราทำการไบแอสกระแสให้กับเลเซอร์ไดโอด จะทำให้มีการทำงานคล้ายกับ LED คือ จะมีแสงที่เกิดจากโฟรตอน (proton) ปล่อยออกมาอย่างสะเปะสะปะ เรียกสภาวะนี้ว่า spontaneous emission เมื่อเพิ่มกระแสให้สูงขึ้นดังอีกรูป ข เลเซอร์ไดโอดจะก้าวไปสู่สภาวะ population inversion ซึ่งมีการคายโฟรตอนมากขึ้น โฟรตอนเหล่านี้ จะชนกับอิเลคตรอนอื่น ๆ ทำให้มีการเพิ่มโฟรตอนมากขึ้นอีก จนเข้าสู่สภาวะ stiimulated emission ดังรูป ค. โฟรตรอนตัวใหม่ที่เกิดขึ้นนี้จะมีเฟส (Phase) เหมือนโฟรตอนที่เข้าชนอิเลคตรอน เมื่อโฟรตอนที่มีเฟสเดียวกันมีมากขึ้น จะทำให้



รูปที่2.1.1 ขั้นตอนการเกิดแสงเลเซอร์

ทำให้ เกิด Standing wave ขึ้นระหว่างรอยต่อของสาร P-N ทำให้แสงที่เปล่งออกมามีลักษณะเป็นแสงเลเซอร์ ค่ากระแสที่เราป้อนให้กับตัวเลเซอร์ไดโอดจนสามารถเปล่งแสงเลเซอร์ออกมาได้นี้ต้องมีค่ามากกว่ากระแสเทรสโฮล (threshold curreng) ของเลเซอร์ไดโอดเอง (ค่ากระแสเทรสโฮล คือ ค่ากระแสต่ำสุดที่เลเซอร์ไดโอดจะเปล่งแสงออกมาได้เป็นแสงเลเซอร์)

กล่าวคือ ถ้าจ่ายกระแสพัลส์ (Pulse current) 1 ลูกป้อนให้กับตัวเลเซอร์ไดโอดดังรูปที่ 2.1.2 ในช่วงเวลาจาก a-d ซึ่งกระแสพัลส์มีค่าต่ำกว่ากระแสเทรสโฮล เลเซอร์ไดโอดจะทำงานเป็น LED ตัวหนึ่ง ช่วงเวลาจากd-c กะแสพัลส์มีค่ามากกว่ากระแสเทรดโอล เลเซอร์ไดโอดจะเปล่งแสงออกมาเป็นแสงเลเซอร์ และเลเซอร์ในช่วง c ถึง d กระแสพัลส์ลดลงต่ำกว่ากระแสเทรสโฮล เลเซอร์ไดโอดก็จะทำงานเป็นแบบ LED (คือแสงกระจายอย่างสะเปะสะปะ) อีกครั้งหนึ่ง



รูปที่2.1.2 แสดงกระแสพัลส์และการทำงานของเลเซอร์ไดโอด

โดยปกติการที่จะให้เลเซอร์ไดโอดทำงานได้ เราจะต้องป้อนกระแสเป็นพัลส์ดังนั้นจึงต้องคำนึงถึงค่าช่วงเวลาขึ้น(rise Time)และช่วงเวลาลง(fall time)ของกระแสพัลส์ ต้องมีค่าน้อยที่สุด เพราะถ้ามีค่ามากเกินไป จะทำเลเซอร์ไดโอดเกิดความร้อนที่รอยต่อ เมื่อเกิดความร้อนขึ้นค่ากระแสเทรสโฮลจะมีค่ามากขึ้นอีก ทำให้กระแสมีค่าไม่ถึงกระแสเทรสโฮล เลเซอร์ไดโอด ก็จะไม่ทำงาน หรือความร้อนจากรอยต่อถ้าเกิดมากเกินไปจะทำให้เสียหายให้เกิดกับเลเซอร์ไดโอดได้

รายละเอียดและคุณสมบัติทางไฟฟ้าของเลเซอร์ไดโอด แสดงไว้ในรูปที่ 1.1.3 ซึ่งเป็นของบริษัท Laser Diode Ladoratories แห่งอเมริกา

ข้อควรระวังในการใช้งาน

ในการใช้งานเลเซอร์ไดโอดจะต้องมีความระมัดระวังเป็นอย่างมาก เพราะถ้าเกิดความผิดพลาดขึ้นจะทำความเสียหายให้กับตัวเลเซอร์ไดโอดได้ ฉะนั้นจึงควรคำนึงถึงสิ่งต่าง ๆ ต่อไปนี้

1. ในการทำงานที่อุณหภูมิห้อง (25*) ต้องป้อนกระแสในลักษณะพัลส์
2. ความกว้างของพัลส์ (pulse width) ต้องไม่เกินค่าที่กำหนดของตัวเลเซอร์ไดโอด(ดูจากสเปค)โดยทั่วไปมีค่าประมาณไม่เกิน 200 nS โดยวัดที่ 50% ของกระแสใช้งานสูงสุด
3. ค่าของ duty factorต้องไม่เกินค่าที่กำหนด โดยที่

duty factor =ความกว้างของพัลส์/ระยะห่างระหว่างพัลส์

เช่น ค่าของ duty factor เป็น 0.1 ถ้าใช้พัลส์ที่มีความกว้าง 200 nS ค่าความห่างระหว่างพัลส์ต้องมีค่าไม่ต่ำกว่า 2000 nS

4. ค่ากระแสไบเอสตรง (forword bias current) เป็น peak สูงสุดไม่เกินค่าที่กำหนด
5. ค่าสูงสุดของกระแสไบแอส ต้องไม่ต่ำกว่าค่ากระแสเทรสโฮล
6. ค่าสูงของแรงดันไฟฟ้าที่ตกคร่อมตัวเลเซอร์ไดโอดต้องไม่เกินค่าที่กำหนด ปกติมีค่าอยู่ในช่วง 5-8 โวลท์ที่กระแสทำงานสูงสุด

ข้อกำหนดนี้ต้องปฏิบัติตามโดยเคร่งครัด มิฉะนั้นเลเซอร์ไดโอดซึ่งมีราคาประมาณตัวละ 2000 บาท ขึ้นไปอาจเสียหายได้ การทดสอบให้แน่ใจควรใช้ตัวต้านทาน 1 โอมห์ต่อเป็นโหลดเทียม(dummy load) แทนตัวเลเซอร์ไดโอดเสียก่อน เมื่อทดสอบจนมั่นใจแล้ว จึงค่อยเปลี่ยนมาใช้เลเซอร์ไดโอดจริง ๆ

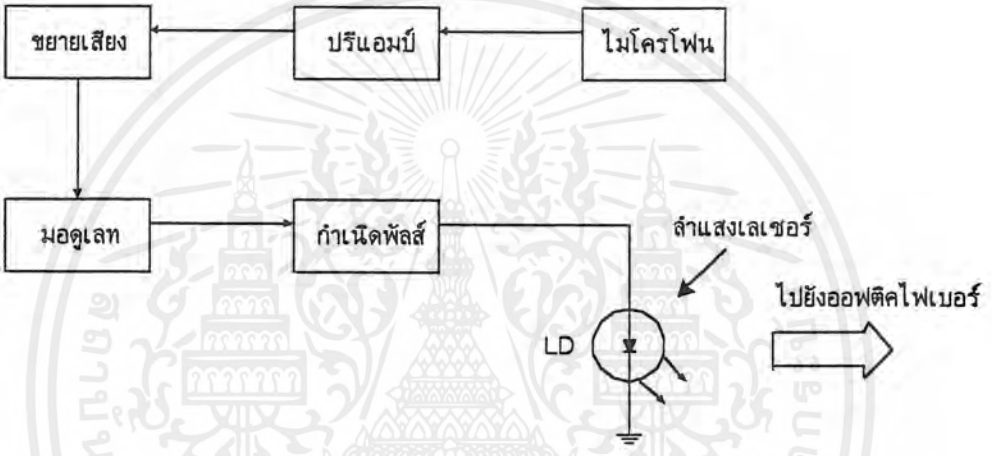
ดังตารางที่2.1.1 ตัวอย่างรายละเอียดและคุณสมบัติทางไฟฟ้าของเลเซอร์ไดโอด

		LD-60	LD-61	LD-62	LD-63	LD-65	LD-66	LD-67	LD-68	Units
Total Peak Radiant Flux	Min.	2	1	5	5	10	8	16	16	Watts
at max. rated I_{fm}	Typ.	2.3	1.5	6	6	12	9.5	20	20	Watts
Maximum Peak Forward Current	I_{fm}	10	10	20	25	40	40	60	75	Amps.
Typical Threshold Current	I_{th}	3	3.5	6	7	10	12	16	18	Amps.
Typical Peak	@ I_{fm}	5.0	5.0	5.8	6.5	6.7	6.7	7.0	8.0	
Forward Voltage	@ 50mA	1.2	1.2	1.2	1.2	1.2	1.2	1.2	1.2	Volts
Emitting Area		3×.08	3×.08	6×.08	6×.08	9×.08	9×.08	16×.08	16×.08	Mils.

	Symbol	Min.	Typical	Max.	Units
Wavelength of Peak Intensity	λ		904		nm
Spectral Width @ 50% points	$\Delta\lambda$		3.5		nm
Rise Time of Radiant Flux -10% to 90% pts.	T_r		< 0.5		nS
Pulse Width -50% point @ I_{fm}	T_p			200	nS
Duty Factor @ I_{fm}				0.1	
Storage Temperature	T_s	-196		+150	°C
Operating Temperature	T_c	-196		+75	°C

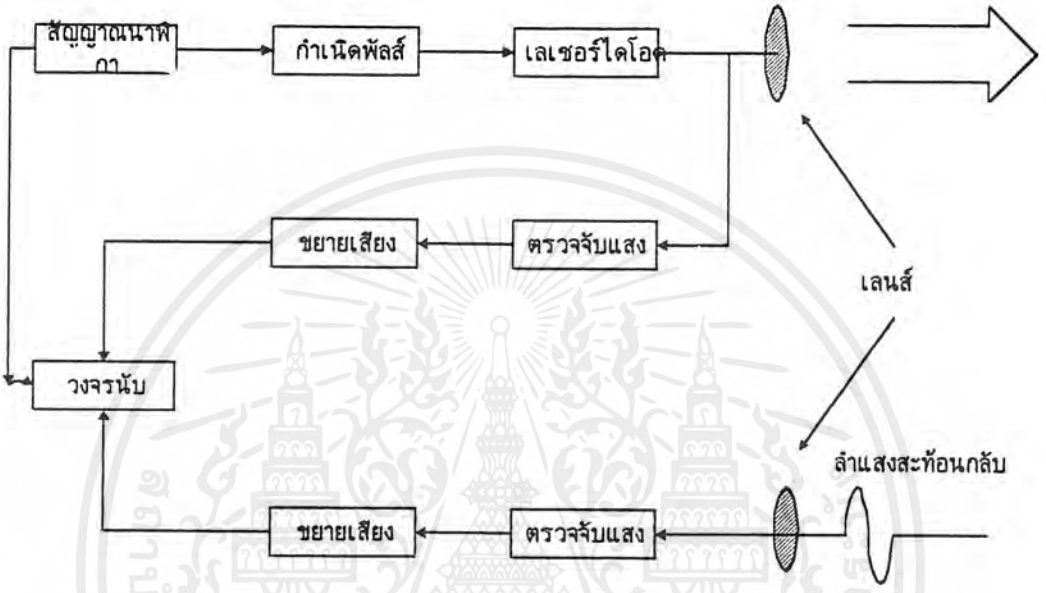
การประยุกต์ใช้งาน

เนื่องจากแสงเลเซอร์เป็นแสงที่มีความเข้มสูงกว่าแสงธรรมดามาก อีกทั้งเลเซอร์ไดโอดเป็นอุปกรณ์ที่ขนาดเล็ก จึงได้มีการนำใช้งานการสื่อสารผ่านท่อไฟเบอร์ออพติค โดยเลเซอร์ เป็นตัวกลางในการเปลี่ยนข้อมูลทางไฟฟ้าให้เป็นข้อมูลทางแสงการใช้งานเลเซอร์ไดโอดสามารถส่งข้อมูลได้ถูกต้องและไกลกว่าแสงธรรมดามาก หลักการง่าย ๆ ในการใช้งานแสดงดังรูปที่ 1.1.3



รูปที่2.1.3 การสื่อสารที่ใช้เลเซอร์ไดโอด

ในรูปที่2.1.4เป็นการนำเอาเลเซอร์ไดโอดมาประยุกต์ใช้ในการวัดระยะทางโดยมีสัญญาณนาฬิกาเป็นตัวควบคุมจำนวนพัลส์ที่ป้อนให้กับเลเซอร์ไดโอด พร้อมกับเป็นตัวรีเซตวงจรนับ เมื่อมีการป้อนพัลส์ 1 ลูก ให้กับเลเซอร์ไดโอดจะเกิดแสงขึ้น 1 ครั้ง ส่งออกไปยังเป้าหมาย ขณะเดียวกันแสงเดียวกันนี้จะทำหน้าที่เริ่มต้นวงจรนับให้ทำงานเมื่อแสงกระทบเป้าหมายแล้วสะท้อนกลับมายังภาครับจะทำให้วงจรนับหยุดทำงาน ก็จะได้ระยะเวาลาการเดินทางไปกับ จากจุดกำหนดแสงกับเป้าหมาย นำเวาลาที่ได้มาคำนวนโดยวงจรอิเลคทรอนิคส์ ก็จะทราบระยะทางออกมาได้ ถ้าเป้าหมายมีการเคลื่อนที่ก็สามารถหาความเร็วออกมาได้ เหมาะกับการใช้ตรวจจับความเร็วของรถยนต์บนถนนที่ขับเร็วเกินกำหนด



รูปที่ 2.1.4 หลักการของเครื่องวัดระยะทางด้วยแสงเลเซอร์

นอกจากนี้ยังมีการใช้เลเซอร์ไดโอดในคอมแพคดิสค์ไปในการรับส่งข้อมูลจากแผ่นดิสค์ ซึ่งเราจะสังเกตได้ว่าเสียงจากคอมแพคดิสค์มีคุณภาพที่ดี เกี่ยวกับการแพทย์ ก็ได้นำเอาเลเซอร์ไดโอดมาใช้ในการฝังเข็ม โดยอาศัยพลังงานแสงเลเซอร์ที่กระตุ้นจุดฝังเข็มแทนการฝังเข็มจริงบนร่างกาย และได้นำเลเซอร์ไดโอดมาใช้ในการลบจุดด่างดำบนร่างการของมนุษย์ได้อีกด้วย

ข้อควรระวังขณะใช้งาน

แสงเลเซอร์จากเลเซอร์ไดโอดไม่อาจมองตรงด้วยตาเปล่า เพื่อความปลอดภัยจะอันตรายซึ่งเกิดจากการทำลายด้วยตาจากแสงเลเซอร์ จึงต้องระมัดระวังขณะใช้งาน คือ อย่าจ้องมองไปตรง ยังตัวมันหรือ มองในระยะใกล้ ๆ นาน ๆ

2.2มอเตอร์กระแสตรง

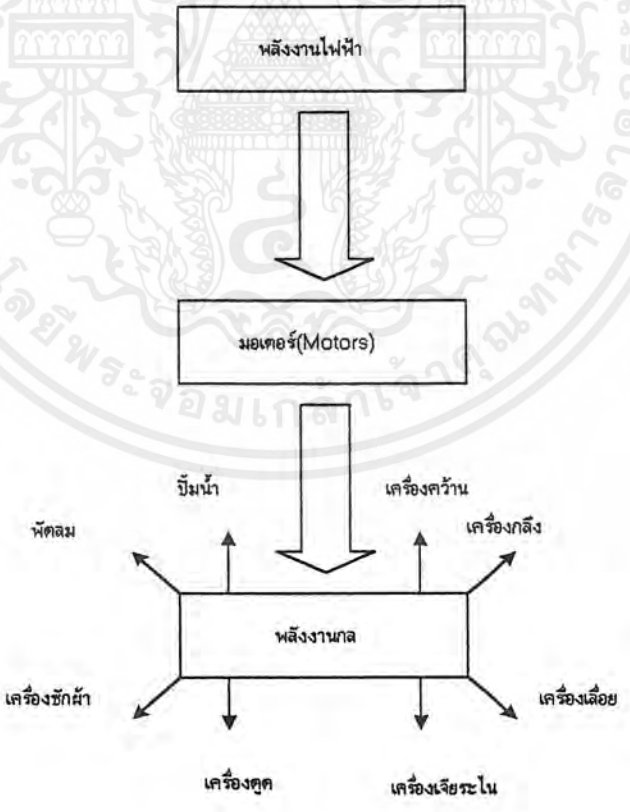
มอเตอร์

เครื่องใช้ไฟฟ้าประเภทเครื่องกล ที่ใช้งานกันอยู่โดยทั่วไปภายในอาคารบ้านเรือนและโรงงานอุตสาหกรรม ได้แก่ พัดลม เครื่องบด เครื่องปั่น เครื่องซักผ้า ปั๊มน้ำ เครื่องกลึง เครื่องใส เครื่องเจาะ เครื่องคว้าน เครื่องเลื่อย เครื่องเจียระไน ฯลฯ ต่างก็ทำงานด้วยการหมุนขับของมอเตอร์ ดังนั้นมอเตอร์จึงเป็นเครื่องกลไฟฟ้าที่ให้กำเนิดพลังงานกลที่จำเป็นและสำคัญยิ่งประเภทหนึ่ง

หลักการเบื้องต้นของมอเตอร์

มอเตอร์คือเครื่องกลไฟฟ้าที่เปลี่ยนพลังงานไฟฟ้า ให้เป็นพลังงานกล

วงจรไฟฟ้าของมอเตอร์



รูป 2.2.1 มอเตอร์ คือเครื่องกลไฟฟ้าที่เปลี่อนพลังงานไฟฟ้าเป็นพลังงกล

มอเตอร์กระแสตรงมีด้วยกัน 3แบบคือ

1.มอเตอร์กระแสตรงแบบอนุกรม

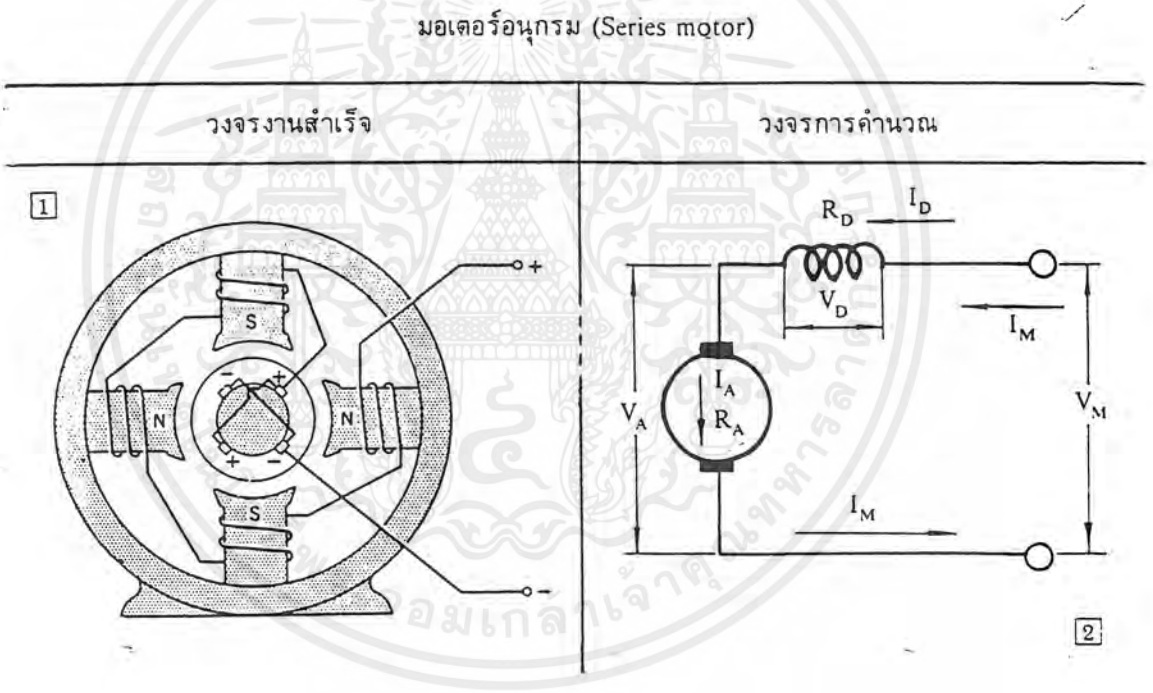
2.มอเตอร์กระแสตรงแบบขนาน

3.มอเตอร์กระแสตรงแบบผสม

แต่ในโครงงานนี้จะกล่าวถึงเฉพาะแบบอนุกรมเท่านั้นและจะอธิบายไว้ในหัวข้อต่อไป

วงจรไฟฟ้าของมอเตอร์

มอเตอร์อนุกรม (Series motor)



รูปที่ 2.2.2 วงจรไฟฟ้าของมอเตอร์อนุกรม

ในวงจรการคำนวณรูปที่ 2.2.2 [2] กำหนดค่าต่าง ๆ ให้ดังนี้ :

Vm = แรงดันเมน

V_m = แรงดันเมน

V_m = แรงดันเมน

ΔV = แรงดันตกคร่อมสายต่อเข้ามอเตอร์

V_D = แรงดันตกคร่อมขดลวดชุดอนุกรม

V_A = แรงดันตกคร่อมขดลวดอาร์เมเจอร์

I_m = กระแสมอเตอร์

I_D = กระแสขดลวดอนุกรม

I_A = กระแสอาร์เมเจอร์

R_D = ความต้านทานขดลวดอนุกรม

R_A = ความต้านทานอาร์เมเจอร์

R_L = ความต้านทานของสายต่อเข้ามอเตอร์

P_m = กำลังของมอเตอร์

$$I_A = I_D = I_M \qquad 2.2\text{-}1$$

$$V_D = I_D . R_D = I_M \ R_D = I_A \ R_D \qquad 2.2\text{-}2$$

$$V_A = I_A . R_A = I_D . R_A = I_M . R_A \qquad 2.2\text{-}3$$

วงจรมอเตอร์ในงานแบบไฟฟ้า

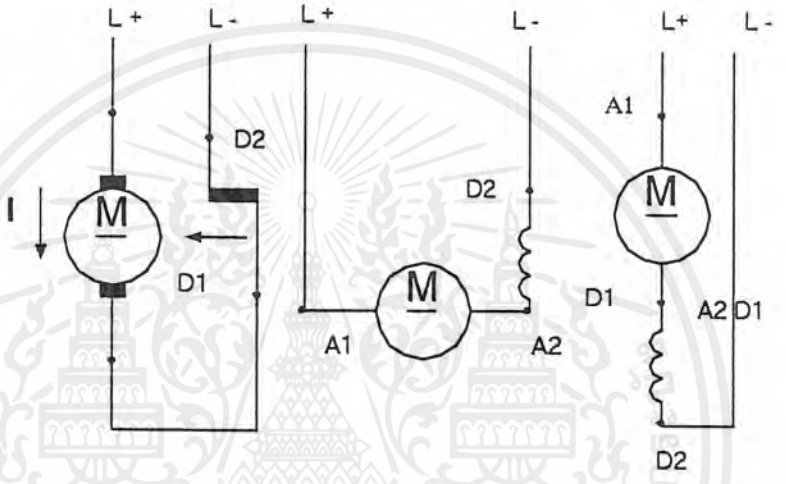
วงจรมอเตอร์ในงานไฟฟ้าได้แก่ แบบงานสำเร็จ แบบงานติดตั้ง และแบบงานควบคุม เขียนวงจรแทนด้วยสัญลักษณ์ไฟฟ้าทั้งหมดดังนี้ :

สัญลักษณ์ของขดลวดนามแม่เหล็ก	
ชนิดของขดลวด	**สัญลักษณ์**
ขดลวดอาร์เมเจอร์ (Armature Winding)	A_1 - A_2
ขดลวดสนามแม่เหล็กช่วย (Interpole)	B_1 - B_2
ขดลวดสนามแม่เหล็กเสริม (Compensating Winding)	C_1 - C_2

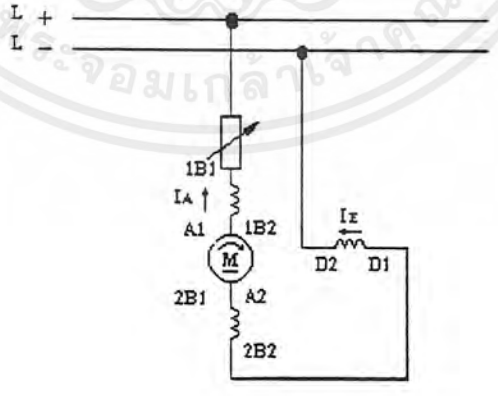
ขดลวดสนามแม่เหล็กชุดอนุกรม (Series field Winding)	D_1 - D_2
ขดลวดสนามแม่เหล็กชุดขนาน(Shunt field Winding)	E_1 - E_2
ขดลวดสนามแม่เหล็กไฟฟ้ากระตุ้นภายนอก (Separately excited field Winding)	F_1 - F_2

รูปที่2.2.3 สัญลักษณ์ไฟฟ้าของขดลวดสนามแม่เหล็กของเครื่องกลไฟฟ้ากระแสตรง

ตัวอย่างวงจรมอเตอร์อนุกรม



รูปที่ 2.2.4 มอเตอร์อนุกรม



รูปที่2.2.5 มอเตอร์อนุกรมในวงจรงานสำเร็จ

คุณสมบัติของมอเตอร์

คุณสมบัติของเครื่องกลไฟฟ้าในที่นี้สามารถแบ่งออกเป็น 2 ลักษณะ คือ คุณสมบัติทั่วไป และคุณสมบัติทั่วไปและคุณสมบัติทางเทคนิค ดังนี้ :

คุณสมบัติทั่วไป :

เป็นคุณสมบัติประจำตัวของเครื่องกลไฟฟ้าแต่ละประเภทที่ควรจะทราบอย่างกว้าง ๆ โดยมิได้เจาะลึกเข้าไปในเนื้อหาเชิงวิชาการแต่อย่างใด ได้แก่ ลักษณะสร้าง ลักษณะงาน ลักษณะของวงจร เช่น คุณสมบัติทั่วไปของมอเตอร์อนุกรม คือ : **ลักษณะสร้าง :** ประกอบด้วยขดลวดสนามแม่เหล็กที่มีความต้านทานต่ำมาก (พันด้วยลวดทองแดงเส้นใหญ่น้อยรอบบนแกนขั้วแม่เหล็ก) ต่อเป็นอนุกรมกับอาร์เมเจอร์และต่อโดยตรงกับแรงดันเมน **ลักษณะวงจร :** A1 - A2: เป็นอาร์เมเจอร์ต่อเป็นอนุกรมกับขดลวดสนามแม่เหล็กชุดอนุกรม: D1 - D2 และต่อโดยตรงกับสายเมน L+, L- และลักษณะงาน: ให้ความเร็วรอบสูงเมื่อโหลดลดลง และความเร็วรอบจะยิ่งสูงมากเมื่อโหลดลดลงมาก ขณะเดียวกันเมื่อโหลดเพิ่มขึ้น ความเร็วรอบจะลดลง จึงเป็นมอเตอร์ที่หมุนไม่คงที่ ความเร็วรอบเปลี่ยนแปลงไปตามโหลด เหมาะสมอย่างยิ่งที่จะใช้เป็นมอเตอร์สตาร์ตรถยนต์ มอเตอร์ขับรถไฟฟ้า และรถราง เครนยกของ หรือลิฟท์

คุณสมบัติทางเทคนิค :

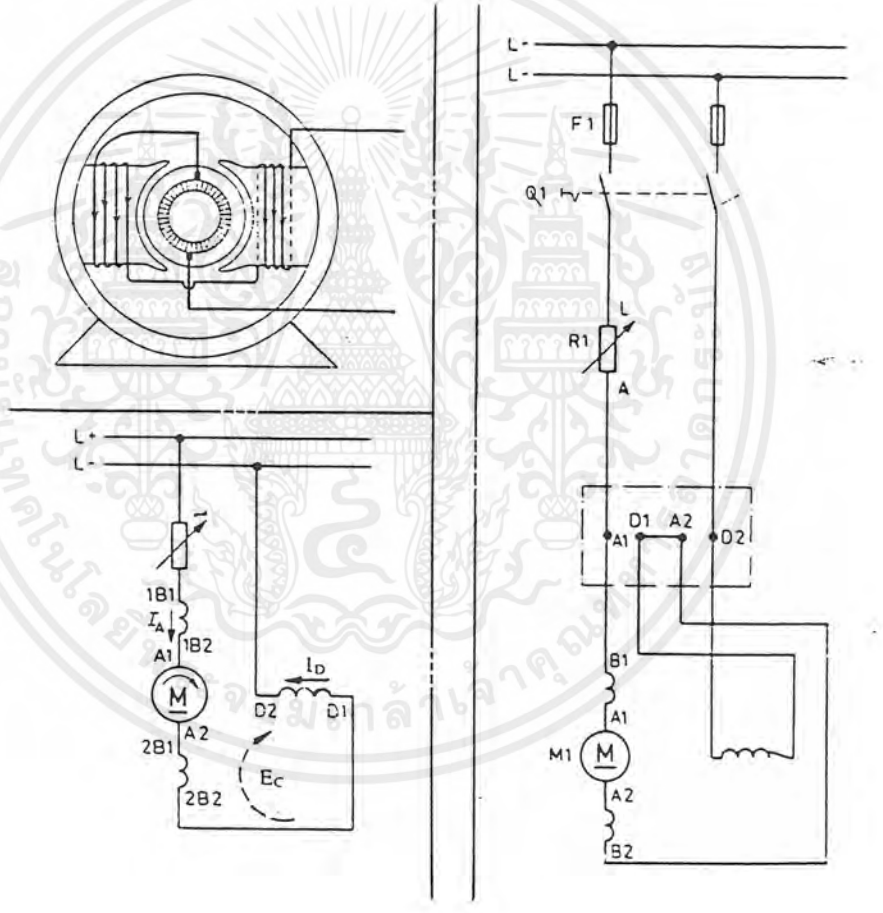
เป็นคุณสมบัติประจำตัวของเครื่องกลไฟฟ้าแต่ละประเภทเช่นเดียวกัน ที่ให้รายละเอียดซึ้งลึกซึ้งเจาะลึกเข้าไปถึงในเนื้อหาเชิงวิชาการ สามารถทดสอบและวัดด้วยเครื่องมือวัดได้ด้วยวิธีประลองในห้องปฏิบัติการประลอง ส่วนใหญ่จะแสดงด้วยกราฟเพื่อแสดงให้เห็นความสำคัญระหว่างค่าหนึ่งกับอีกค่าหนึ่ง เช่น สมรรถนะในการให้กำเนิดแรงเคลื่อนไฟฟ้าของเครื่องกำเนิดไฟฟ้า แสดงด้วย "กราฟแม่เหล็กอิ่มตัว (Saturarion หรือ Magnetization curve)" สมรรถนะในการจ่ายโหลดของเครื่องกำเนิดไฟฟ้ามาแสดงด้วย "External Characteristic" ส่วนคุณสมบัติทางเทคนิคของมอเตอร์มาแสดงด้วย Performance Curve" ซึ่งได้แก่ "สมรรถนะในการหมุนขับโหลด (Speed - load Curves หรือ Speed - load Characteristics)" แสดงให้เห็นความสัมพันธ์ระหว่างความเร็วรอบกับกระแสมอเตอร์ (N = ความเร็วรอบให้อยู่บนแกน Y หรือ Ordinate และ I_A =กระแสอาร์เมเจอร์ให้อยู่บนแกน X หรือ abscissae หรืออาจให้แสดงความสัมพันธ์ระหว่างความเร็วรอบ (n เป็น ordinate หรือแกน Y) กับทอร์ค หรือกำลังที่หมุนขับงาน (T = ทอร์ค , P = กำลังเป็นวัตต์หรือกิโลวัตต์ ให้อยู่บนแกน X หรือ abscissae) จุดประสงค์เพื่อต้องการแสดงให้เห็นถึงความเปลี่ยนแปลงของความเร็วรอบของมอเตอร์ที่หมุนขับโหลดว่าจะมีการเปลี่ยนแปลงไปอย่างไรเมื่อโหลดเปลี่ยนแปลง

2.2.1มอเตอร์อนุกรม

คุณสมบัติทั่วไป

ลักษณะสร้าง ประกอบด้วยขดลวดสนามแม่เหล็กที่มีความต้านทานต่ำ พันด้วยขดลวดทองแดงเส้นใหญ่บนแกนขั้วแม่เหล็กจำนวนน้อยรอบ เช่นเดียวกับขดลวดกระแสแอมมิเตอร์ ต่อเป็นอนุกรมกับอาร์เมเจอร์

และแรงดันเมน



รูปที่ 1.2.6 วงจรมอเตอร์อนุกรม

ลักษณะของวงจร

V_M = แรงดันเมน

E_C = แรงดันไฟฟ้าเหนี่ยวนำกลับ

R_A = ความต้านทานขดลวดอาร์เมเจอร์

R_D = ความต้านทานขดลวดสนามแม่เหล็กชุดอนุกรม

I_M = กระแสมอเตอร์

I_D = กระแสขดลวดสนามแม่เหล็ก

I_A = กระแสอาร์เมเจอร์

$$I_A = I_D = I_M$$

สัญลักษณ์

$A_1 - A_2$: ขดลวดอาร์เมเจอร์

$B_1 - B_2$: ขดลวดสนามแม่เหล็กช่วย

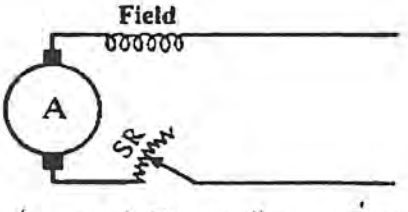
$D_1 - D_2$: ขดลวดสนามแม่เหล็กชุดอนุกรม

คุณสมบัติทางเทคนิค

สมรรถนะในการหมุนขับโหลด : มอเตอร์อนุกรมตามรูป : อาร์เมเจอร์ต่ออนุกรมกับขดลวดสนามแม่เหล็กและต่ออนุกรมกับแรงดันเมน เมื่อโหลดเปลี่ยนแปลง (I) กระแสอาร์เมเจอร์(I) และเส้นแรงแม่เหล็กในสนามแม่เหล็กจะเปลี่ยนแปลงไปด้วย มีผลทำให้ความเร็วรอบมอเตอร์เปลี่ยนแปลงไปในที่สุด ดังนั้นจึงกล่าวได้ว่า ความเร็วรอบของมอเตอร์อนุกรมจะเปลี่ยนแปลงไปตามการเปลี่ยนแปลงของโหลด ด้วยเหตุนี้จึงเรียกมอเตอร์อนุกรมนี้ว่า "Variable-Speed Machine" และว่าการเปลี่ยนแปลงของโหลดเพียงเล็กน้อยจะมีผลทำให้ความเร็วรอบของมอเตอร์เปลี่ยนแปลงไปได้อย่างมาก คือ :

-โหลดเพิ่มขึ้น ความเร็วรอบลดลง

-โหลดลดลง ความเร็วรอบเพิ่มขึ้น



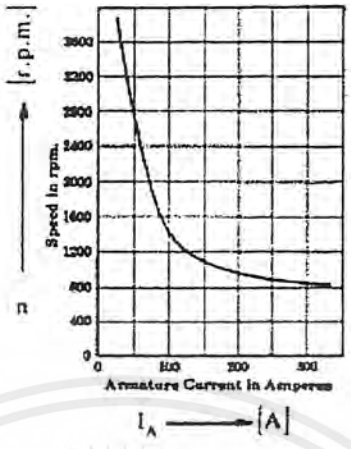
รูปที่ 2.2.7 มอเตอร์อนุกรมต่อกับความต้านทานเริ่มหมุน SR

มอเตอร์อนุกรมที่หมุนขับงานกลด้วยแรงดันเมนคงที่ ขณะเพิ่มโหลดกระแสอาร์เมเจอร์เพิ่มขึ้น ทอร์คของมอเตอร์จะเพิ่มขึ้นด้วย ขณะเดียวกันแรงดันไฟฟ้าเหนี่ยวนำกลับจะลดลง [$E_C=(V_M - I_A R_A) / R_A$] ความสัมพันธ์ตามที่กล่าวมานี้อธิบายได้ว่า : เส้นแรงแม่เหล็กเพิ่มขึ้นตามการเพิ่มขึ้นของกระแสอาร์เมเจอร์ แรงดันไฟฟ้าเหนี่ยวนำกลับจึงลดลงมีผลทำให้ความเร็วรอบของมอเตอร์ลดลงและการลดลงของความเร็วรอบนี้พบว่า จะลดลงของความเร็วรอบนี้พบว่า จะลดลงมากกว่าการเพิ่มขึ้นของเส้นแรงแม่เหล็กในสนามแม่เหล็กอีกด้วย

ด้วยเหตุที่เส้นแรงแม่เหล็กเป็นปฏิภาคโดยตรงกับกระแสอาร์เมเจอร์ ถ้าเพิ่มโหลดให้กระแสอาร์เมเจอร์(I_A) เพิ่มขึ้นเป็น 2 เท่าแล้ว เส้นแรงแม่เหล็กจะเพิ่มขึ้นเป็น 2 เท่าด้วย ความเร็วรอบจะลดลงมากกว่า 2 เท่าของความเร็วรอบเดิม ในทางตรงกันข้ามถ้าลดกระแสอาร์เมเจอร์ลงให้เหลือเพียงครึ่งหนึ่งเส้นแรงแม่เหล็กก็จะลดลงเหลือเพียงครึ่งหนึ่งด้วย ความเร็วรอบจะเพิ่มขึ้นมากกว่าเดิม 2 เท่าเช่นเดียวกันและความสัมพันธ์ที่กล่าวมานี้จะเป็นจริงก็ต่อเมื่อให้มอเตอร์หมุนขับโหลดเบาๆ ทั้งนี้เพราะว่าที่โหลดหนักๆจะมีผลกระทบเนื่องจากจุดอิ่มตัวของวงจรแม่เหล็กและอาร์เมเจอร์รีแอคชัน

มอเตอร์อนุกรมแตกต่างจากมอเตอร์ขนานตรงที่ความเร็วรอบขณะหมุนตัวเปล่า (No-load Speed)ของมอเตอร์อนุกรมไม่มีพิกัดที่กัดที่แน่นอน ไม่มีทอร์ค (หรืออาจมีแต่น้อยมาก) ความเร็วรอบจึงสูงมาก สูงจนกระทั่งมอเตอร์ให้กำเนิดแรงดันเหนี่ยวนำกลับเท่ากับแรงดันเมน ขณะนี้มอร์เตอร์กินกระแสอาร์เมเจอร์น้อยมาก เส้นแรงแม่เหล็กในสนามแม่เหล็กมีค่าน้อยมากเช่นเดียวกัน เป็นผลสืบเนื่องทำให้มอเตอร์หมุนด้วยความเร็วรอบที่สูงมาก ($n=E_C / K'$, เมื่อ $E_C= V_M$) สูงจนกระทั่งสามารถทำให้เกิดแรงเหวี่ยงมากพอที่จะเหวี่ยงให้ตัวนำอาร์เมเจอร์หลุดออกจากสล็อตของแกนอาร์เมเจอร์ได้ อันจะเป็นอันตรายอย่างยิ่งต่อมอเตอร์

ด้วยเหตุนี้การหมุนขับโหลดของมอเตอร์อนุกรมจึงห้ามหมุนขับด้วยระบบสายพานอย่างเด็ดขาดให้ต่อโดยตรงกับโหลดที่ต้องการหมุนขับเท่านั้น มอเตอร์ขนาดเล็กที่มีกำลังน้อยกว่า 7.5 กิโลวัตต์ จะให้กำลังสูญเสียเนื่องจากความต้านทาน ความเสียดทาน และอื่นๆ มากพอที่จะทำให้มอเตอร์หมุนตัวเปล่า ด้วยความเร็วรอบที่ไม่สูงเกินนัก พอที่จะควบคุมให้อยู่ในพิกัดที่ต้องการได้



รูปที่ 2.2.8 กราฟแสดงคุณสมบัติในการหมุนขับโหลดของมอเตอร์อนุกรม

กราฟแสดงสมรรถนะในการหมุนขับโหลด : กราฟรูปที่ 1.2.8 เป็นกราฟแสดงสมรรถนะในการหมุนขับโหลดของมอเตอร์อนุกรม (Speed-Load Characteristic) แสดงให้เห็นถึงความสัมพันธ์ระหว่างความเร็วรอบกับกระแสโหลดของมอเตอร์ ซึ่งจะเห็นได้ว่าความเร็วรอบขณะหมุนตัวเปล่าสูงมาก สูงจนกระทั่งไม่สามารถแสดงในกราฟได้ และเมื่อโหลดเพิ่มขึ้นความเร็วรอบจะลดลงอย่างรวดเร็วและลดลงเรื่อยๆ จนกว่าจะหมุนขับเต็มโหลด และวงจรแม่เหล็กจะถึงจุดอิ่มตัว

กราฟแสดงสมรรถนะในการหมุนขับโหลด (Speed-Load Curve) ของมอเตอร์อนุกรมนี้อาจหาได้การต่อโหลด (Mechanical Load) กับแกนมอเตอร์โดยตรง แล้วปรับโหลดที่หมุนขับจากเต็มโหลดน้อยที่สุดลงมายังโหลดต่ำสุด ทั้งนี้เพื่อมิให้มอเตอร์หมุนด้วยความเร็วรอบอันตราย ทุกระดับของโหลดให้วัดความเร็วรอบ ความสัมพันธ์ระหว่างความเร็วรอบกับกระแสโหลดนำมาเขียนกราฟ: $n = f(I_A)$ จะได้กราฟแสดงสมรรถนะในการหมุนขับโหลดของมอเตอร์อนุกรมตามต้องการ

2.3 ไมโครคอลโทรลเลอร์ MCS-51

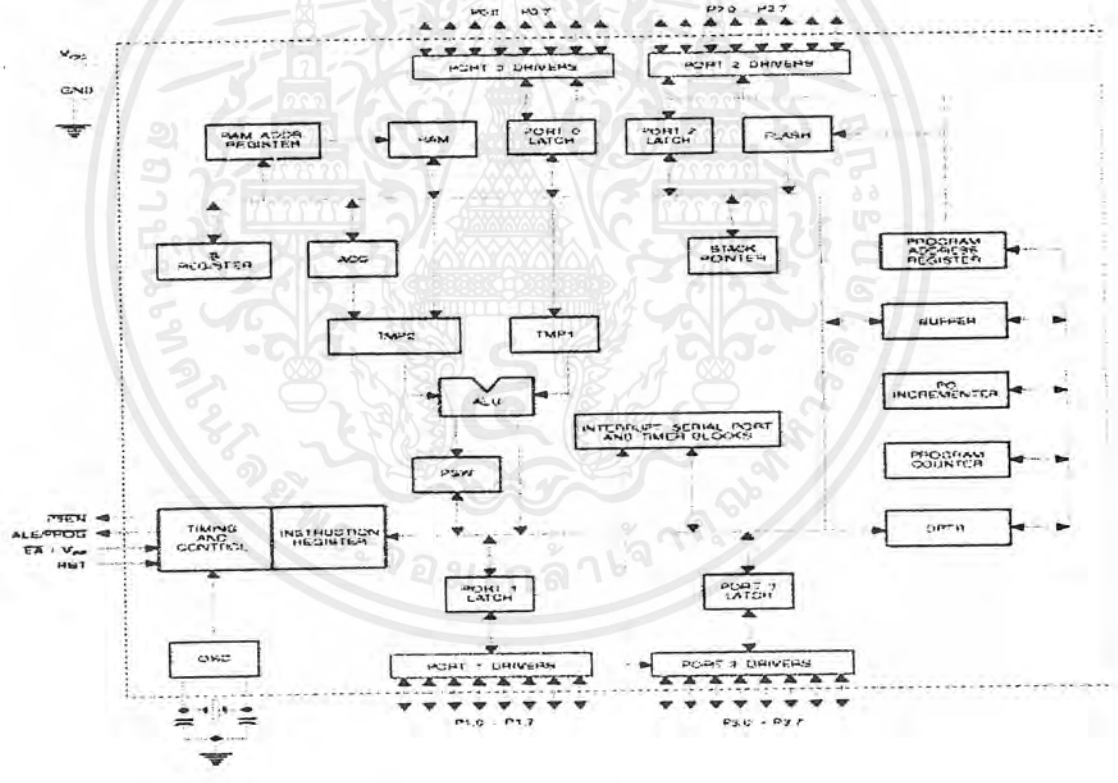
คุณสมบัติของ MCS-51

ต้องการแหล่งจ่ายไฟ + 5V ชุดเดียวมีหน่วยความจำโปรแกรม (Program Memory) ขนาด ไบต์สำหรับเบอร์ 8051 และ 8031, 8032 ไม่มีหน่วยความจำชุดนี้ ส่วน 8052 มีหน่วยความจำถึง 8 กิโลไบต์ มีหน่วยความจำสำหรับเก็บข้อมูล (Data Memory) ขนาด 128 ไบต์ สำหรับ 8052 มีถึง 256

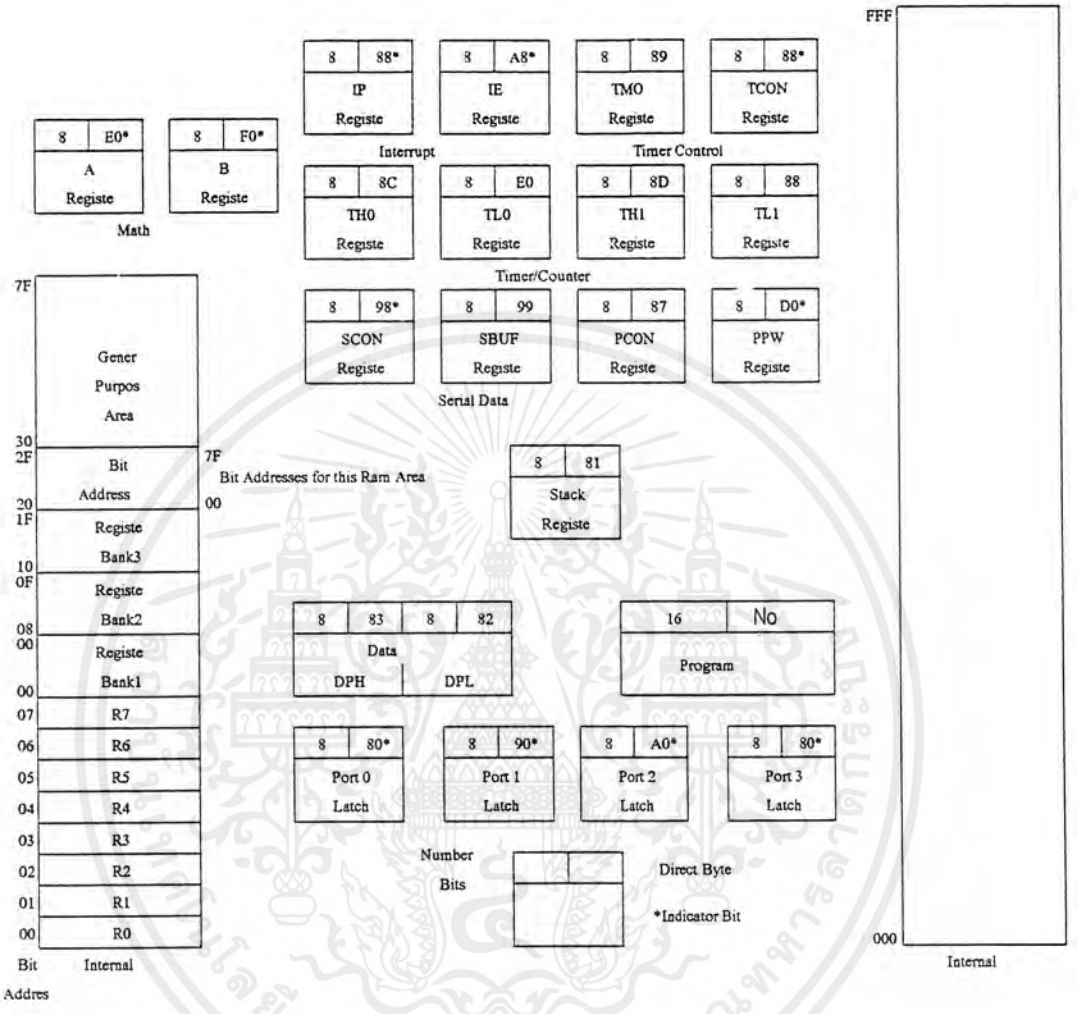
ไบต์หน่วยความจำสำหรับโปรแกรมและข้อมูล (Program Memory) และ Data Memory แยกจากกันอย่างละ 64 กิโลไบต์ คำสั่งที่ใช้เวลาน้อยที่สุดประมาณ 1 uS เมื่อทำงานที่ความถี่ 12 Mhz มี Timer/Counter ขนาด 16 บิต 2 ชุดรับอินเตอร์รัพท์ได้ 6 แหล่ง 5 เวคเตอร์ มีพอร์ตสำหรับส่งข้อมูลอนุกรม (UART) 2 พอร์ต ทั้งรับและส่งในเวลาเดียวกันได้ (Full Duplex) เลือกรูปแบบการส่งข้อมูลได้ 4 โหมดมีคำสั่งในการทำ AND, OR หรือ COMPLEMENT ได้ทั้งแบบ 8 บิต และ 1 บิต

โครงสร้างภายในของ 8051

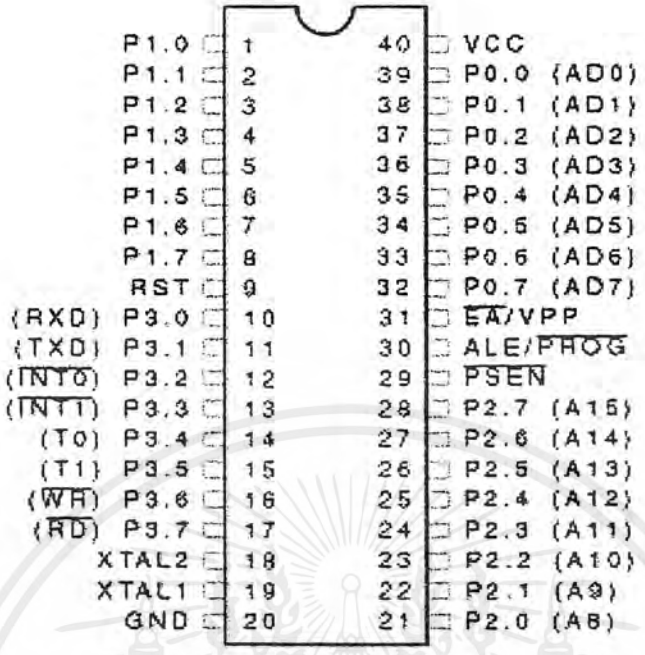
MCS-51 ใช้เทคโนโลยีในการผลิตแบบ NMOS และ CMOS เบอร์ 8032 และ 8052 จะมี ROM BASIC อยู่ภายในจึงสะดวกที่จะเขียนโปรแกรมด้วยภาษาเบสิค โครงสร้างภายในสำหรับเบอร์ 8051 ดังแสดงรูปที่ 2.3.1 และ 2.3.2



รูปที่ 2.3.1โครงสร้างภายในของ MCS-51



รูปที่ 2.3.2 ตำแหน่งต่าง ๆ ของรีจิสเตอร์ต่าง ๆ และหน่วยความจำ



รูปที่ 2.3.3 การจัดวางขาของ 8051

พอร์ตของ 8051

8051 เป็นไมโครคอนโทรลเลอร์ขนาด 40 ขา ซึ่งมีขาต่าง ๆ ดังนี้

- ไฟเลี้ยง (ขา 40) ต่อกับ +5V
- ไฟเลี้ยง (ขา 20) เป็นขา GND
- พอร์ต 0 (ขา 32-39) มีทั้งหมด 8 บิต คือ (P0.0-P0.7) ใช้งานได้ 2 หน้าที่แอดเดรสและข้อมูลออกไปให้หน่วยความจำภายนอกเมื่อทำการเขียนข้อมูลลงในหน่วยความจำภายนอกและอีกหน้าที่หนึ่งคือเป็นพอร์ อินพุทและเอาท์พุท ถ้าต้องการให้ทำงานเป็นอินพุทพอร์ตต้องส่งลอจิก 1 ไปยังพอร์ตนี้
- พอร์ต 1 (ขา 1 – 8) มีทั้งหมด 8 บิตคือ (P1.0-P1.7) มีโครงสร้างคล้าย พอร์ต 0 แต่จะใช้ความต้านทานภายในพูลอัพแทน (Internal pull Up Register)
- พอร์ต 2 (ขา21-28)มีทั้งหมด 8 บิต คือขา (P2.0-P2.7) มีโครงสร้างคล้าย พอร์ต 0 โดยมี FET ตัวล่างตัวเดียวส่วนด้านบนใช้ความต้านทานพูลอัพแทน พอร์ตนี้ทำงาน 2 หน้าที่ คือ สามารถใช้เป็นพอร์ตสำหรับส่งแอดเดรส 8 บิตบน (A8-A15) และเป็นอินพุท,เอาท์พุทพอร์ตใช้งานทั่วไป เมื่อจะใช้งานเป็นอินพุทพอร์ต ต้องส่งลอจิก 1 มาที่พอร์ตนี้ก่อน
- พอร์ต 3 (ขา 10-17) มีทั้งหมด 8 บิท คือขา (P3.0-P3.7) มีโครงสร้างคล้ายพอร์ต1 พอร์ตนี้ทำหน้าที่เป็นอินพุทและเอาพุทพอร์ต ถ้าจะให้พอร์ตนี้เป็นอินพุทพอร์ตก็ให้ส่งลอจิก

'1' มาที่พอร์ตนี้ก่อนและอีกหน้าที่ หนึ่งคือ ส่งสัญญาณควบคุมออกมาและรับสัญญาณเข้าไป สัญญาณต่าง ๆ มีดังนี้

P3.0/RXD (Serial Input Port) เป็นขาที่ใช้รับข้อมูลแบบอนุกรม

P3.1/RXD (Serial Output Port) เป็นขาที่ใช้ส่งข้อมูลแบบอนุกรม

P3.2/INT0 (External Interrupt) ใช้รับสัญญาณขัดจังหวะจากภายนอก

P3.3/INT1 (External Interrupt) ใช้รับสัญญาณขัดจังหวะจากภายนอก

P3.4/T0 (Timer/Counter 0 External Input) ขารับสัญญาณเข้าไปในวงจร Timer/Counter 0 ที่ทำหน้าที่นับจำนวนไซเคิลของสัญญาณ T1 นี้หรือสัญญาณนาฬิกาก็ได้

P3.5/T1 (Timer/Counter 1 External input) ขารับสัญญาณเข้าไปยังTimer/Counter 1 ซึ่งมีการทำงานเหมือนกับ T0

P3.6/WR (External Data Memory Write strobe) ขาสัญญาณควบคุมการเขียนข้อมูลไปยังหน่วยความจำสำหรับข้อมูลภายนอก 8051

P3.7/RD (External Data Memory Read strobe) ขาสัญญาณควบคุมการอ่านข้อมูลจากหน่วยความจำสำหรับข้อมูลภายนอก

- ALE (ขา 30)เป็นขาส่ง สโตรบสำหรับใช้ในการแลตช์แอดเดรสไบต์ต่ำ (A0-A7) ที่ส่งออกมาจาก (พอร์ต 0) สัญญาณนี้จะแอคทีฟทุก ๆ ครั้งใน 1 แมชชีนไซเคิล (1/16 ของสัญญาณนาฬิกา)
- PSEN (ขา 29) เป็นขาที่ส่งสโตรบสำหรับอ่านข้อมูลจาก Program Memory ภายนอก (หน่วยความจำประเภท ROM EPROM) สัญญาณนี้จะส่งออกมา 2 ครั้ง ในแต่ละแมชชีนไซเคิลแต่ถ้าเป็นการอ่าน Internal Program Memory จะไม่มีสัญญาณ นาฬิกาออกที่ขานี้
- EA (ขา 30)ถ้าป้อนลอจิก 0 เข้าที่ขานี้ ซีพียูจะอ่านค่าจาก Program Memory ภายนอกชิพเท่านั้น แต่ถ้าถูกป้อนด้วยลอจิก 1 ก็จะอ่านโปรแกรมภายในชิพ
- RST (ขา 9) เป็นขารีเซ็ทซีพียู จะรีเซ็ทได้ก็ต่อเมื่อป้อนลอจิก 1 เข้าที่ขานี้นานอย่างน้อย 2 แมชชีนไซเคิล เมื่อ ซีพียูถูกรีเซ็ทค่าต่าง ๆ ในรีจิสเตอร์ใด ๆ จะมีคำสั่งดังตารางที่ 2.3.1
- XTAL1 (ขา 19) ใช้ต่อคริสตัลภายนอกโดยเป็นอินพุทเข้าสู่วงจรออสซิลเลเตอร์
- XTAL2 (ขา 18) ใช้ต่อคริสตัลภายนอกโดยเป็นเอาท์พุทของวงจรออสซิลเลเตอร์

ตารางที่ 2.3.1 แสดงค่าของรีจีสเตอร์เมื่อทำการรีเซทแล้ว

รีจีสเตอร์	
PC	0000H
ACC	00H
B	00H
PSW	00H
SP	00H
DPTR	OOOOH
P0-P3	0FFH
IP	00H
IE	0X000000B
TMOD	00H
TCON	00H
TH0,TL0	00H
TH1,TL1	00H
SCON	00H
SBUF	Indeterminate

การแบ่งประเภทของหน่วยความจำ

หน่วยความจำที่ใช้กับ MCS –51 มีอยู่ด้วยกัน 2 ชนิดคือ Program Memory และ Data Memory Program Memoryซึ่งเป็นหน่วยความจำที่ใช้เก็บโปรแกรมสั่งงานบรรจุอยู่ในชิพ 8051 ส่วนที่เป็น Program Memory ก็คือ ROM ขนาด 4 กิโลไบต์นั่นเอง แต่ถ้าเป็นเบอร์ 8052 ก็คือ ROM ขนาด 8 กิโลไบต์

Data Memory เป็นหน่วยความจำที่ใช้เก็บข้อมูลหน่วยความจำนี้ สามารถเขียนข้อมูลลงไปและ อ่านข้อมูลออกมาได้ ซึ่งเป็นหน่วยความจำภายในชิพมีเพียง 128 ไบต์ สำหรับเบอร์ 8051 และ 256 ไบต์ สำหรับเบอร์8052 ส่วนหน่วยความจำภายนอกชิพมี 64 กิโลไบต์

พื้นที่หน่วยความจำที่เข้าถึงข้อมูลโดยตรงและทางอ้อม (Direct and Indirect Address Area) พื้นที่ 128 ไบต์ ล่างสุดจะแบ่งเป็น 3 ส่วน

1. รีจิสเตอร์แบงก์ (Register Banks 0-3) ตั้งแต่ตำแหน่ง (00H-1FH) จะเป็นส่วนของรีจิสเตอร์แบงก์ (0-3) โดยแบ่งเป็นแบงก์ละ 8 ไบต์รวมแล้วได้ 32 ไบต์ ถ้าซีพียูทำงานอยู่ที่

แบงก์ 3 เมื่อถูกรีเซ็ทก็จะกลับมาทำงานที่แบงก์ 0 เสมอ และ SP จะมาเริ่มต้นที่ตำแหน่ง 07Hทันที

2. บริเวณหน่วยความจำที่ใช้คำสั่งเขียนอ่านเกี่ยวกับบิตได้ (Bit Addressable Area) พื้นที่ตั้งแต่แอดเดรส (20H-7FH) จำนวน 16 ไบต์ หรือถ้านับเป็นบิตจะได้เท่ากับ 128 บิต ซึ่งตำแหน่งบิต 00,01,02,03,04,05,06,07 ก็คือตำแหน่งหน่วยความจำตำแหน่ง 20H ที่บิต 0,1,2,3,4,5,6,7, ตามลำดับดูตาราง 1.3.2เช่นถ้าต้องการเซทบิต D0 ของตำแหน่ง 20H ก็จะต้องเขียนคำสั่งว่า SETB 00H

ตารางที่ 2.3.2 แสดงตำแหน่งของหน่วยความจำภายใน

RAM	MSB							LSB
7FH	7F	7E	7D	7C	7B	7A	79	78
2FH	77	76	75	74	73	72	71	70
2EH	6F	6E	6D	6C	6B	6A	69	68
2DH	67	66	65	64	63	62	61	60
2CH	5F	5E	5D	5C	5B	5A	59	58
2BH	57	56	55	54	53	52	51	50
2AH	4F	4E	4D	4C	4B	4A	49	48
29H	47	46	45	44	43	42	41	40
28H	3F	3E	3D	3C	3B	3A	39	38
27H	37	36	35	34	33	32	31	30
26H	2F	2E	2D	ก) 2C	2B	2A	29	28
25H	27	26	25	24	23	22	21	20
24H	1F	1E	1D	ข) 1C	1B	1A	19	18
23H	17	16	15	14	13	12	11	10
22H	0F	0E	0D	0C	0B	0A	09	08
21H	07	06	05	04	03	02	01	00
20H								

3. บริเวณหน่วยความจำที่ใช้งานทั่วไป (Scratch Pad Area) พื้นที่ตั้งแต่ (30H-7FH) จะเขียนข้อมูลได้ทีละไบต์เท่านั้น ไม่สามารถใช้คำสั่งเกี่ยวกับบิตได้

- ช่วง 78H-37H คือช่วง SCRATCH PAD AREA
- ช่วง 28H-27H คือช่วง BIT ADDRESSABLE SEGMENT
- ช่วง 18H-07H คือช่วง REGISTER BANK

ส่วนการสร้างฐานเวลาและการนับ (Timer /Counter)

ในไมโครคอนโทรลเลอร์ 8051 จะมีตัวจับเวลาอยู่ภายในชิพ รีจีสเตอร์ที่เกี่ยวข้องกับการใช้งาน Timer/Counter แสดงไว้ดังตารางที่ 2.3.3

ตารางที่ 2.3.3 รีจีสเตอร์ที่เกียวข้องกับการใช้ Timer/Counter

รีจีสเตอร์	หน้าที่	ตำแหน่ง	สามารถอ้างอิงตำแหน่งบิต
TCON	Control	88H	Yes
TMODE	Mode	89H	No
TL0	Timer 0 Low-byte	8AH	No
TL1	Timer 1 Low-byte	8BH	No
TH0	Timer 0 High-byte	8CH	No
TH1	Timer 1 High-byte	8DH	No

Timer Mode Register (TMODE)

เป็นรีจีสเตอร์ที่ใช้สำหรับควบคุม Timer/Counter ไม่สามารถอ้างอิงข้อมูลระดับบิตได้ โดยสี่บิตล่างจะทำหน้าที่ควบคุม Timer/Counter 0 และสี่บิตบนใช้สำหรับ Timer/Counter 1

GATE	C/T	M1	M0	GATE	C/T	M1	M0

MSB LSB

- GATE เป็นบิตสำหรับควบคุม Timer/Counter ถ้าเป็นลอจิก 1 Timer/Counter นั้นจะทำงาน เมื่อขาสัญญาณที่ขา INTx เป็นลอจิก 1 เท่านั้น ที่บิต TRx ใน TCON ต้องเป็นลอจิก 1 ด้วย ซึ่งจะเป็นการควบคุมด้วย ฮาร์ดแวร์ เมื่อเป็นลอจิก 0 การควบคุม Timer/Counter นั้นๆ จะเป็นการควบคุมด้วย โปรแกรมที่บิต TRx ของรีจีสเตอร์ TCON เท่านั้น

- C/T เป็นบิตสำหรับเลือกการทำงานของ Timer/Counter เป็น Timer หรือ Counter โดย ถ้าเป็นลอจิก 1 จะเป็น Counter

- M0,M1 เป็นบิตสำหรับเลือกการทำงานของ Timer/Counter

ตารางที่ 2.3.4 แสดงการใช้งาน Timer/Counter ในโหมดต่างๆ

M1	M0	mode	ความหมาย
0	0	0	ใช้เป็น Timer แบบ 13บิต
0	1	1	ใช้เป็น Timer แบบ 16บิต
1	0	2	ใช้เป็น Timer แบบ 8 บิต สามารถตั้งค่าใหม่อัตโนมัติ
1	1	3	Timer 0 จะถูกแยกออกเป็น Timer 8บิตสองตัวคือ TL0 และ TH0 โดยไม่ใช้ Timer1

TCON

เป็นรีจีสเตอร์ ใช้สำหรับควบคุมการทำงานชอง Timer/Counter และระดับการตอบรับอินเตอร์รัพท์ ซึ่งสามารถอ้างอิงระดับบิตได้

TF1	TR1	**TF0**	TR0	IE1	IT1	**IE0**	IT0

MSB LSB

2.4 พอร์ท 8255

8255 เป็นไอซี LSI ขนาด 40ขา ดังรูป ซึ่งแสดงตำแหน่งของขาต่างๆทั้ง40ขา ซึ่ง 8255 นี้มีพอร์ทสำหรับส่งข้อมูลอยู่ด้วยกัน 3 พอร์ท มีชื่อดังนี้ พอร์ทA,B และ C โดยพอร์ท C นี้จะแบ่งออกเป็น 2 ส่วน คือ พอร์ท C บน (CLO) กับพอร์ท C ล่าง (CHI) และยังมีอีกพอร์ทหนึ่ง ซึ่งทำหน้าที่ควบคุมการทำงานของพอร์ท A, B และ C โดยรับคำสั่งมาจาก CPU พอร์ทนี้เรียกว่าพอร์ทควบคุม (Control port) การทำงานของอร์ท จะถูกกำหนดโดย CPU โดย CPU จะส่งข้อมูลทางดาต้าบัส (Data Bus) ให้แก่พอร์ทควบคุมหน้าที่ของขาต่าง ๆ

CS (Chip Select)

ขาใช้ในการเลือกว่าจะให้ 8255 ตัวนี้ทำงานหรือไม่ โดยถ้าได้รับลอจิก 0 จะทำให้ 8255 เชื่อมต่อแข้ากับระบบต่าง ๆ ของ CPU แต่ถ้าเป็นลอจิก 1 จะอยู่ในสภาวะ High Impednce

RD (Read Enable)

ถ้าได้รับลอจิก 0 และ CS ได้รับลอจิก 0 เช่นกัน แสดงว่า 8255 ทำการส่งข้อมูลจากพอร์ทที่ CPU ต้องการติดต่อด้วยนั้นให้แก่ CPU ทางดาต้าบัส

WR (Write Enaqble)

ถ้าได้รับลอจิก 0 พร้อมกับ CS แล้ว 8255 จะส่งข้อมูลจากดาต้าบัสของ CPU ออกไปยังพอร์ทที่ CPU กำหนดไว้

RESET

ทำการ Reset 8255 เข้าสู่โหมดอินพุท ทุก ๆ พอร์ท และเคลียร์สถานะต่าง ๆ ของ 8255

D0-D7

เป็น Data Bus ที่ใช้รับส่งข้อมูลกับ CPU

A0-A1

คือขาแอดเดรสเลือกพอร์ทที่ CPU ต้องการติดต่อ

00 = พอร์ท A

01 = พอร์ท B

10 = พอร์ท C

11 = พอร์ทควบคุม

PA0-PA7

เป็นขาสัญญาณของพอร์ท A

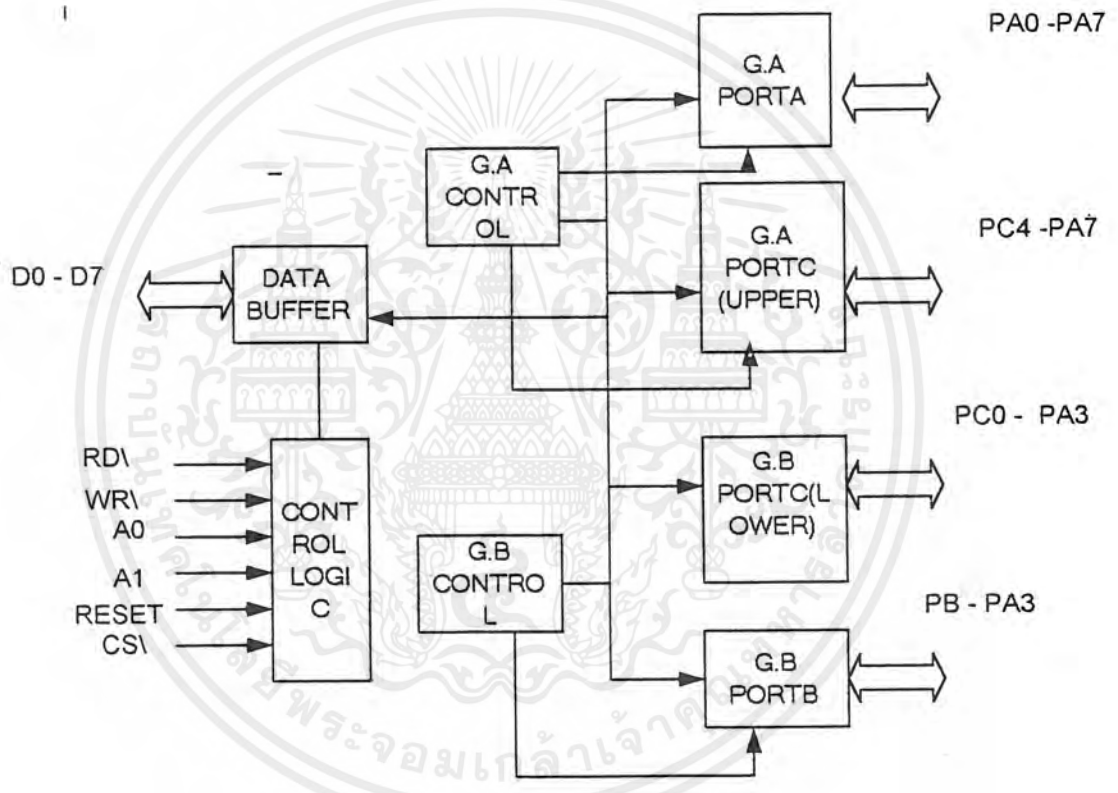
PB0-PB7

เป็นขาสัญญาณของพอร์ท B

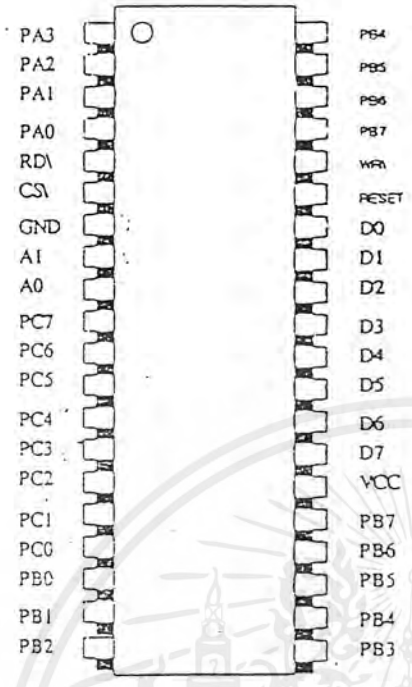
PC0-PC7

เป็นขาสัญญาณของพอร์ท C โดยแยกเป็น PC0-PC3 และ PC4-PC7 โดยสามารถแยกการทำได้โดยอิสระ

BLOCK DIAGRAM



รูป2.4.1 โครงสร้างของ 8255



Pin Names

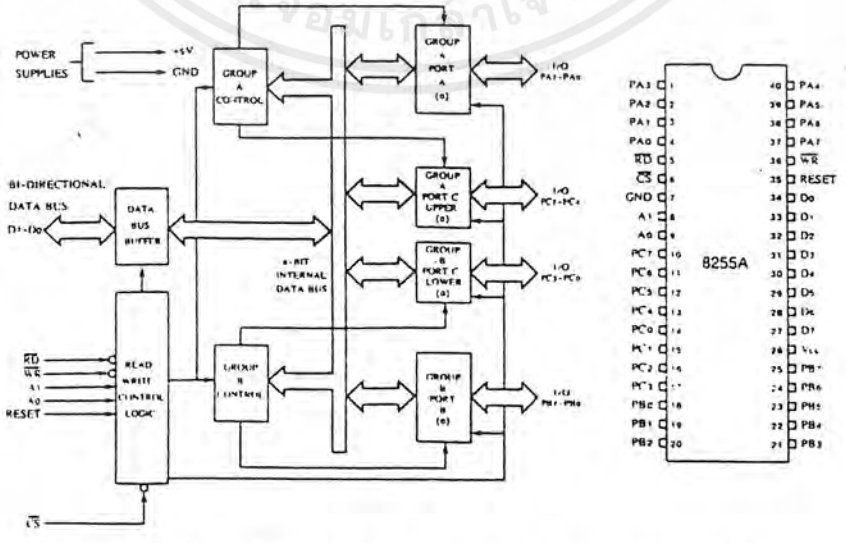
D0 - D7	DATA BUS (BI-DIRECTIONAL)
RESET	RESET INPUT
CS\	CHIP SELECT
RD\	READ INPUT
WR\	WRITE INPUT
A0 - A1	PORT ADDRESS
PA0 - PA7	PORT A
PB0 - PB7	PORT B
PC0 - PC7	PORT C

รูปที่2.4.2 การวางขาของ8255

พอร์ทควบคุม

เป็นพอร์ทการกำหนดการทำงานของ 8255 โดยควบคุมจาก CPU ทำการส่งรหัสควบคุม ผ่านทาง ดาต้าบัสมายังพอร์ทควบคุมของ 8255 รหัสควบคุมนี้มีขนาด 1 ไบท์เรียกว่า Control byte

8255 พอร์ทอินพุท-เอาต์พุตของระบบ



รูปที่ 2.4.3 แสดงแผนผังและการจัดขาของ 8255A-5

8255 เป็นชิปซัปพอร์ทที่ทำหน้าที่เป็นพอร์ทขนาน สามารถรับส่งข้อมูลแบบขนานได้รวดเร็ว มีพอร์ทให้ใช้งาน 3 พอร์ทด้วยกันคือ พอร์ท A, พอร์ท B และพอร์ท C นอกจากนั้นยังมีพอร์ทควบคุม (Control Port) อีก 1 พอร์ท พอร์ท A และ B จะมีขนาด 8 บิต ส่วนพอร์ท C จะถูกแบ่งออกเป็น 4 บิตบนและล่าง โดยที่ 4 บิตบน (Pc4-Pc7) ถูกควบคุมด้วยพอร์ท A ส่วน 4 บิตล่าง (PcO-Pc3) จะถูกควบคุมโดยพอร์ท B

ตารางที่ 2.4.1 หน้าที่ และขาสัญญาณต่าง ๆ ของ 8255

ขาสัญญาณ	หน้าที่
PA,PB,PC (Port A, B, C)	เป็นขาสัญญาณของพอร์ททั้ง 3 ของ 8255 คือ พอร์ท A,B และ C การเลือกใช้งาน 1 ใน 3 พอร์ท จะใช้แอดเดรส A0, A1 เลือกอีกทีหนึ่ง
CS (Chip Select)	สัญญาณเลือกใช้งานชิป 8255 ก่อนจะโปรแกรมอะไรลงไปต้องให้สัญญาณนี้แอกทีฟคือ เป็น "0" ด้วย
RD (Read)	เป็นสัญญาณอินพุตเพื่อใช้อ่านข้อมูลภายในของพอร์ทของ 8255 สัญญาณนี้จะต้องแอกทีฟพร้อมกับ CS
WR (Write)	เป็นสัญญาณอินพุตเพื่อใช้เขียนข้อมูลส่งสู่พอร์ทภายในของ 8255 สัญญาณนี้จะต้องแอกทีฟพร้อมกับสัญญาณ CS
D_0–D_7 (Data Bus)	เป็นขาสัญญาณแบบไบต์ไดเรกแชนแนลคือ 2 ทิศทาง สามารถใช้รับส่งข้อมูลจากซีพียูได้
A_0, A_1(Address)	เป็นสัญญาณอินพุตใช้งานร่วมกับสัญญาณ RD และ WR เพื่อเลือกและควบคุม 1 ใน 3 พอร์ท หรือคอนโทรลเวิร์ดรีจิสเตอร์

ตารางที่ 2.4.1(ต่อ) หน้าที่ และขาสัญญาณต่าง ๆ ของ 8255

ขาสัญญาณ			หน้าที่		
CS	RD	WR	A1	A0	ความหมาย
0	1	0	0	0	เขียนพอร์ท A ซึ่งเป็นข้อมูล
0	0	1	0	0	อ่านพรอร์ท A ซึ่งเป็นข้อมูล
0	1	0	0	1	เขียนพอร์ท B ซึ่งเป็นข้อมูล
0	0	1	0	1	อ่านพอร์ท B ซึ่งเป็นข้อมูล
0	1	0	1	0	เขียนพอร์ท C ซึ่งเป็นข้อมูล
0	0	1	1	0	อ่านพรอ์ท C ซึ่งเป็นข้อมูล
0	1	0	1	1	เขียนข้อมูล ซึ่งเป็นรหัสควบคุม
0	0	1	1	1	อ่านเข้ามาซึ่งไม่มีความหมายใด
RESET			เป็นสัญญาณอินพุตใช้รีเซตแก่ 8255 เพื่อทำการเคลียร์สถานะต่าง ๆ 8255 และทำให้พอร์ท (A,B,C) ทั้ง 3 เป็นอินพุตหมด		

โหมดการทำงานของ 8255

8255 มีโหมดการทำงานอยู่ 3 โหมดด้วยกันคือ

1. Mode 0 (Basic I/O)
2. Mode 1 (Strobed I/O)
3. Mode 2 (Bi-Directional Bus)

แต่ใน โครงงาน นี้ใช้ เฉพาะ Mode 0 ในการทำงาน

การโปรแกรม 8255

การใช้งาน 8255 จะต้องทำการโปรแกรมเสียก่อน โดยการส่งค่าคอนโทรลไบต์ให้แก่พอร์ทควบคุม จะเป็นคำสั่งขนาด 8 บิตคือ 1 ไบต์ ซึ่งแต่ละบิตจะมีความหมายและใช้งานต่างกัน คอนโทรลไบต์นี้จะเป็นคำสั่งกำหนดโหมดการทำงานของ 8255 และการกำหนดให้พอร์ททั้ง 3 (A,B,C) เป็นอินพุต หรือเอาต์พุต ดังแสดงในรูปที่ 2.3

บิต D_0: ข้อมูลในบิตนี้จะกำหนดให้พอร์ท C ล่าง (Pc0-Pc3) เป็นอินพุตหรือเอาต์พุต ถ้า บิตนี้ เป็น "1" จะเป็นอินพุต แต่ถ้าเป็น "0" จะเป็นเอาต์พุต

บิต D_1: ข้อมูลในบิตนี้จะกำหนดให้พอร์ท B เป็นอินพุตหรือเอาต์พุต ถ้าบิตนี้เป็น "1" จะเป็นอินพุต แต่ถ้าเป็น "0" จะเป็นเอาต์พุต

บิต D_2: ข้อมูลในบิตนี้จะกำหนดการเลือกโหมดของกลุ่ม B คือ ถ้าเป็น "1" จะทำงานในโหมด 1 แต่ถ้าเป็น "0" จะทำงานในโหมด "0"

บิต D_3: ข้อมูลในบิตนี้จะกำหนดให้พอร์ท C บน (Pc4-Pc7) เป็นอินพุตหรือเอาต์พุต ถ้าบิตนี้เป็น "1" จะเป็นอินพุต แต่ถ้าบิตนี้เป็น "0" จะเป็นเอาต์พุต

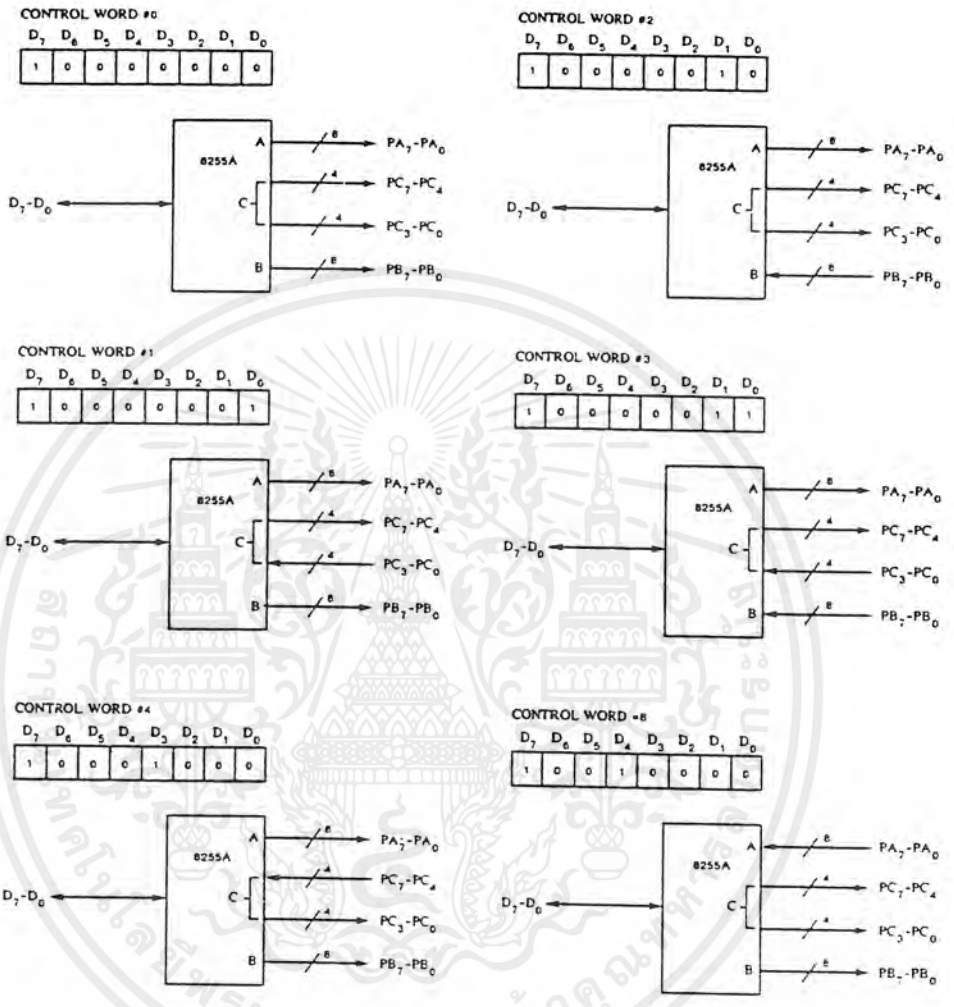
บิต D_4: ข้อมูลในบิตนี้จะกำหนดให้พอร์ท A เป็นอินพุตหรือเอาต์พุต ถ้าบิตนี้เป็น "1" จะเป็นอินพุต แต่ถ้าเป็น "0" จะเป็นเอาต์พุต

บิต D_5, D6: ข้อมูลทั้ง 2 บิตนี้จะเป็นตัวเลือกโหมดการทำงานของกลุ่ม A ถ้ามีค่าเป็น "00" จะทำงานในโหมด 0 ถ้ามีค่าเป็น "01" จะทำงานในโหมด 1 แต่ถ้ามี ค่าเป็น "1X" (10 และ11) จะทำงานในโหมด 2

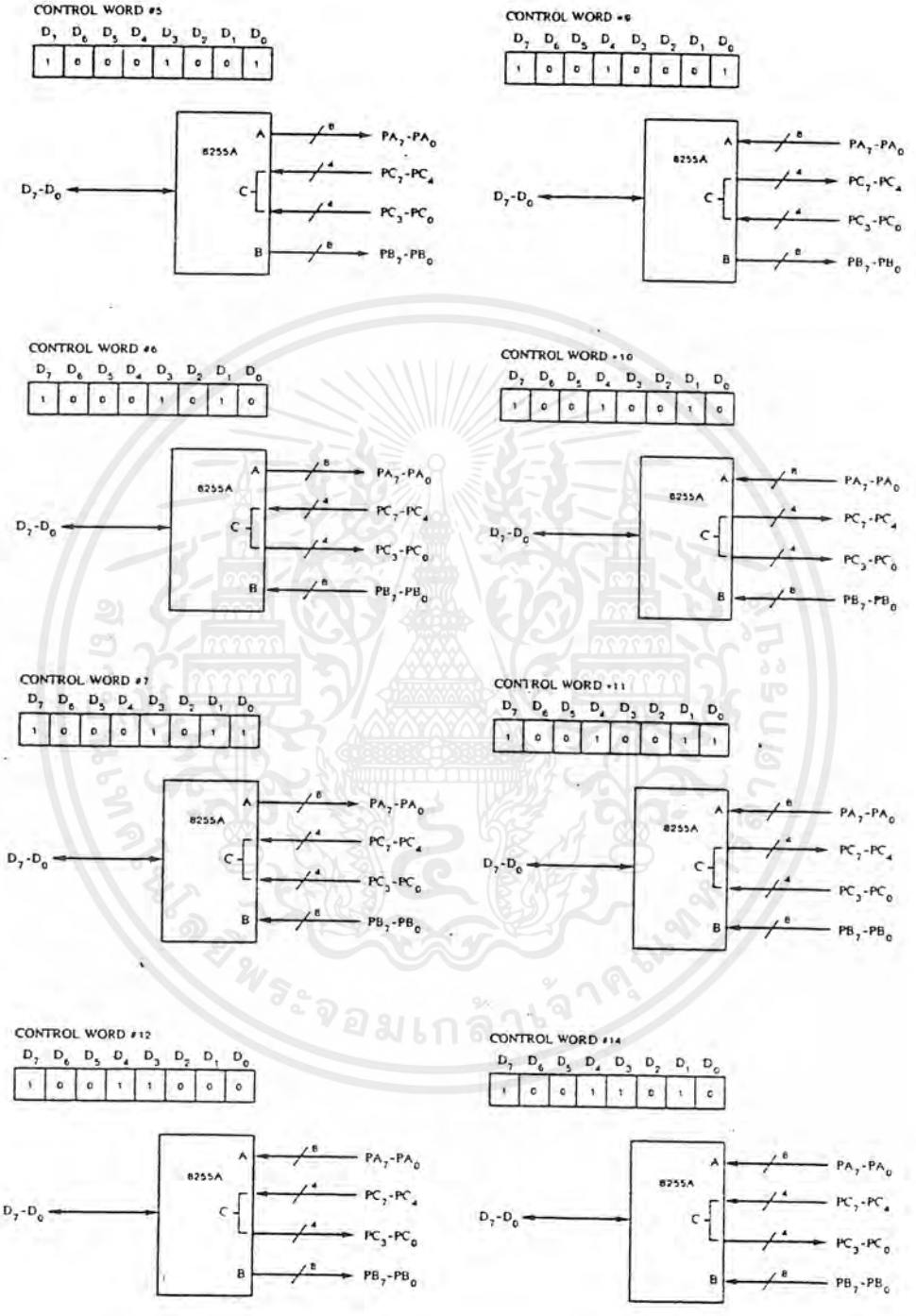
บิต D_7: ข้อมูลในบิตนี้เกี่ยวกับการเซตแฟล็กใน 8255

Mode 0 (Basic I/O)

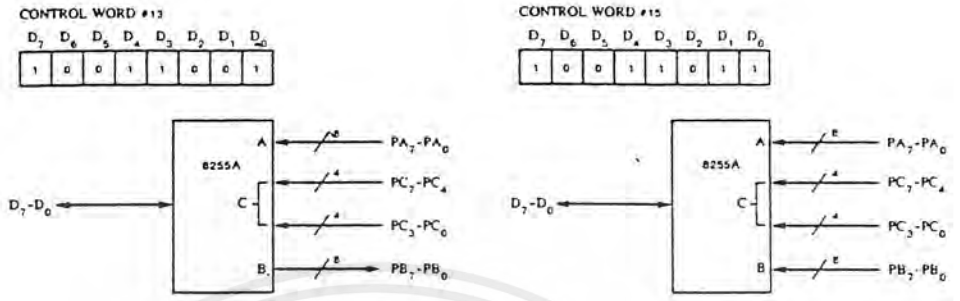
โหมด 0 หรือโหมดอินพุตเอาต์พุตพื้นฐานจะกำหนดให้พอร์ท A, B และ C เป็น อินพุต-เอาต์พุตได้ ไม่มีสัญญาณ "Handshaking" เอาต์พุตที่ออกจากพอร์ต A, B และ C จะแลตช์ค่าไว้ด้วย สามารถจะจัดรูปแบบของอินพุตเอาต์พุตได้ 16 แบบด้วยกัน ดังรูปที่ 2.4.4



รูปที่ 2.4.4แสดงรหัสควบคุมในโหมด 0



รูปที่ 2.4.4(ต่อ) แสดงรหัสควบคุมในโหมด 0



รูปที่ 2.4.4(ต่อ) แสดงรหัสควบคุมโหมด 0

การโปรแกรมโดยส่งค่าคอนโทรลไบต์ทั้ง 8 บิตไปยังพอร์ท A, B และ C เป็นอินพุตเอาต์-พุต สามารถจะเขียนคำสั่งภาษาแอสเซมบลี ภาษาปาสคาล หรือภาษาซีก็ได้ ในระบบฮาร์ดแวร์ของไมโครคอมพิวเตอร์ PC จะกำหนดให้ 8255 มีตำแหน่งหมายเลขพอร์ทตามนี้คือ

หมายเลขพอร์ท	พอร์ท 8255
$F0_H$	PORT A
$F1_H$	PORT B
$F2_H$	PORT C
$F3_H$	Control Port

ถ้าต้องการให้พอร์ท A, C เป็นอินพุตและพอร์ท B เป็นเอาต์พุต ทำงานในโหมด 0 ทั้ง 3 พอร์ทจะต้องส่งค่าคอนโทรลไบต์เป็น 10011001 หรือ 99H ใช้คำสั่งภาษาแอสแซมบลี คือ

```
MOV AL, 99H        (นำค่าคอนโทรลไบต์ส่งให้จิสเตอร์ AL)
OUT F3H, AL        (นำค่าในรีจิสเตอร์ AL คือ 99H ส่งให้พอร์ท F3H)
```

2.5การติดต่อสื่อสารผ่านพอร์ตขนาน (Parallel Port Communication)

พอร์ตข้อมูลแบบขนานนั้นเป็นที่นิยมใช้กันอย่างมาก สำหรับการควบคุม หรือติดต่อกับอุบกรณ์ภายนอก ด้วยเครื่องคอมพิวเตอร์ ซึ่งจะใช้อุปกรณ์ภายนอกต่อร่วมด้วยน้อยมากเมื่อเปรียบเทียบกับพอร์ตข้อมูลแบบอนุกรม พอร์ตข้อมูลแบบขนานนี้สามารถใช้เป็นอินพุทได้ 9 บิต และใช้เป็นเอาท์พุทได้ 12 บิต โดยพอร์ตนี้จะประกอบด้วยสายควบคุม 4 เส้น, สายแสดงสถานะ 5 เส้นและสายข้อมูลอีก 8 เส้น ซึ่งโดยปกติพอร์ตข้อมูลแบบขนานนี้มักจะเป็นแบบ D-Type ตัวเมียขนาด 25 Pin ติดอยู่ที่ด้านหลังของเครื่องคอมพิวเตอร์ ส่วนอีกพอร์ตหนึ่งที่เป็น D-Type ตัวผู้ขนาด 25 Pin เช่นกันนั้นเป็นพอร์ตข้อมูลแบบอนุกรมหรือ RS-232 ซึ่งแตกต่างจากพอร์ตข้อมูลแบบขนานโดยสิ้นเชิง

มาตราฐานของพอร์ตขนานไดมีการกำหนดขึ้นมาใหม่โดย IEEE 1284 ในปี 1994 ซึ่งพอร์ตข้อมูลแบบขนานจะต้องสามารถทำงานได้ตามโหมดต่างๆ ดังนี้

1. Compatibility Mode
2. Nibble Mode
3. Byte Mode
4. EPP Mode (Enhanced Parallel Port)
5. ECP Mode (Extended Capabilities Port)

โดยที่ไดร์เวอร์หรืออุปกรณ์จะต้องสนับสนุนต่อการทำงานในโหมดต่างๆ และยังต้องสนับสนุนต่อการใช้งานของพอร์ตมาตรฐานแบบขนาน (SPP: Standard Parallel Port) ด้วย การทำงานในโหมด Compatibility, Nibble และ Byte นั้นเป็นการทำงานในโหมด EPP และ ECP จะต้องอาศัยอุปกรณ์ภายนอกร่วมด้วย การทำงานในสองโหมดหลังนี้จะทำให้มีความเร็วในการรับส่งข้อมูลสูงขึ้น และยังคงสนับสนุนกับการทำงานของพอร์ตมาตรฐานแบบขนานด้วย

โหมด Compatibility หรือเรียกอีกชื่อหนึ่งว่า Centronics Mode จะสามารถส่งข้อมูลได้ในทิศทางเดียวเท่านั้นซึ่งโดยปกติจะมีความเร็วประมาณ 50 KB/s ในการรับข้อมูลจากพอร์ตขนานจะต้องเปลี่ยนมาทำงานในโหมด Nibble หรือ Bye ในโหมด Nibble จะสามารถรับข้อมูลได้ 4 บิต (Nibble) เช่นการรับข้อมูลจากอุปกรณ์ภายนอกมายังเครื่องคอมพิวเตอร์ ส่วนการทำงานแบบ Byte Mode จะเป็นการทำงานในแบบสองทิศทาง (bi-direction) ซึ่งจะพบในการ์ดบางตัวเทานั้น โดยจะสามารถรับข้อมูลได้ครั้งละ 8 บิต (byte)

สำหรับการทำงานในโหมด EPP และ ECP นั้นจะต้องมีส่วนของอุปกรณ์ภายนอกร่วมด้วย เพื่อสร้างและจัดการกับสัญญาณที่ใช้สำหรับการแฮนด์เชค (handshake) ในการส่งข้อมูลไปยังเครื่องพิมพ์ (printer) จะใช้โหมด Compatibility โดยโปรแกรมที่ส่งข้อมูลจะมีลำดับขั้นการทำงานดังนี้คือ

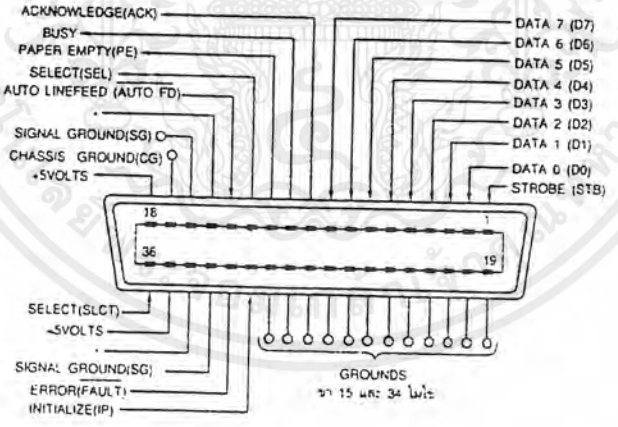
1. ส่งข้อมูลครั้งละ 8 บิตไปยังพอร์ตข้อมูล
2. ตรวจสอบสัญญาณ busy ของเครื่องพิมพ์ กรณีที่สัญญาณ busy active อยู่เครื่องพิมพ์จะไม่รับข้อมูลใดๆ ทั้งสิ้น ดังนั้นหากมีการส่งข้อมูลไปยังเครื่องพิมพ์ในช่วงเวลานี้ ข้อมูลเหล่านั้นก็จะสูญหายไป
3. เปลี่ยนสัญญาณ Strobe เป็น " 0 " เพื่อบอกให้เครื่องพิมพ์รู้ว่าข้อมูลบนพอร์ตข้อมูลเป็นข้อมูลที่ถูกต้อง
4. เปลี่ยนสัญญาณ Strobe เป็น " 1 " อีกครั้งหลังจากที่ได้เปลี่ยนสัญญาณ Strobe เป็น " 0 " แล้วประมาณ 5 μS (หลังจากขั้นตอนที่ 3)

ขั้นตอนต่างๆ เหล่านี้จะเป็นตัวกำหนดความเร็วในการรับส่งข้อมูล แต่ในโหมด EPP และ ECP จะใช้ฮาร์ดแวร์เป็นตัวตรวจสอบว่าเครื่องพิมพ์ว่างอยู่หรือไม่และสร้างสัญญาณ Strobe และสัญญาณในการทำ handshaking จากภายนอก ดังนั้นการทำงานในสองโหมดนี้จึงสามารถรับส่งข้อมูลได้ด้วยความเร็วประมาณ 1-2 MB/s แต่โหมด ECP จะมีการใช้ DMA channel ร่วมกับบัฟเฟอร์แบบ FIFO ซึ่งทำให้มีความเร็วมากว่าโหมด EPP

ลักษณะทางฮาร์ดแวร์

ลักษณะภายนอกของพอร์ตข้อมูลแบบขนานได้แสดงได้ดังรูปที่ 1ซึ่งเป็นคอนเน็คเตอร์แบบ D-Type ขนาด 25 Pin รูป 1(ก) และคอนเน็คเตอร์แบบ Centronics ขนาด 36 Pin รูป 1(ข) ส่วนหน้าที่ของขาสัญญาณต่างๆ ได้แสดงไว้ในตารางที่ 1 โดยทั่วไปแล้วทั่วไปแล้วคอนเน็คเตอร์แบบ D-Type จะติดอยู่กับเครื่องคอนเน็คเตอร์แบบ D-Type ขนาด 25 Pin ซึ่งจะอยู่ที่เครื่องคอมพิวเตอร์ส่วนคอนเน็คเตอร์แบบ Centronics จะอยู่ที่เครื่องพิมพ์ ตามมาตราฐาน IEEE 1284 ได้กำหนดคอนเน็คเตอร์ที่ใช้กับพอร์ตขนานไว้ 3 แบบคือ

1. แบบ 1284 Type A เป็นคอนเน็คเตอร์แบบ D-Type ขนาด 25 Pin ซึ่งจะอยู่ที่เครื่องคอมพิวเตอร์
2. แบบ 1284 Type B เป็นคอนเน็คเตอร์แบบ Centronics ขนาด 36 Pin ซึ่งจะพบที่เครื่องพิมพ์เป็นส่วนใหญ่
3. แบบ 1284 Type B เป็นคอนเน็คเตอร์แบบ Centronics ขนาด 36 Pin เช่นเดียวกับแบบที่ 2 แต่จะเพิ่มสายสัญญาณอีก 2 เส้นเพื่อตรวจสอบว่ามีอุปกรณ์ต่ออยู่หรือไม่



รูปที่ 2.5.1 (ก)คอนเน็คเตอร์แบบ D-Type (ข) คอนเน็คเตอร์แบบ Centronics

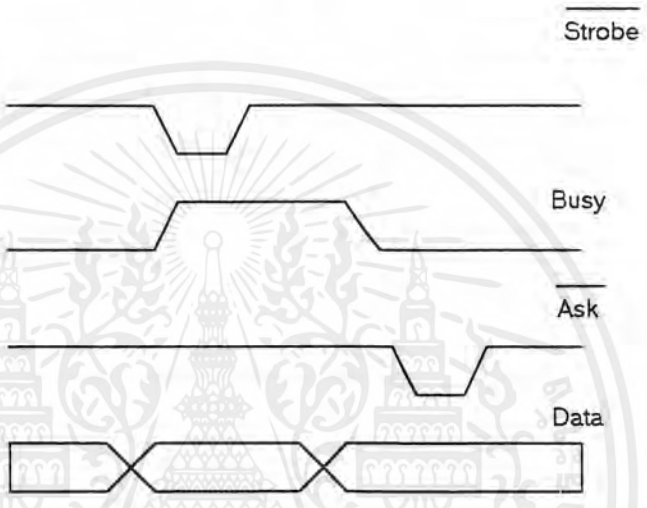
ตารางที่ 2.5.1 กำหนดขาสัญญาณของพอร์ตข้อมูลแบบขนาน

Pin No (D-type 25)	Pin No (Centronics)	SPP Signal	Direction In/Out	Register	Hardware Inverted
1	1	Strobe	In/Out	Control	Yes
2	2	Data0	Out	Data	
3	3	Data1	Out	Data	
4	4	Data2	Out	Data	
5	5	Data3	Out	Data	
6	6	Data4	Out	Data	
7	7	Data5	Out	Data	
8	8	Data6	Out	Data	
9	9	Data7	Out	Data	
10	10	Ack	In	Status	
11	11	Busy	In	Status	Yes
12	12	Paper-Out Paper End	In	Status	
13	13	Select	In	Status	
14	14	Auto-Linefeed	In/Out	Control	Yes
15	32	Error/Fault	In	Status	
16	31	Initialize	In/Out	Control	
17	36	Select-Printer Select-In	In/Out Gnd	Control	Yes
18-25	19-30	Ground			

สัญญาณเอาท์พุตของพอร์ตขนานปกติจะเป็นระดับโลจิก TTL ซึ่งโดยมากจะสร้างเป็นแบบ ASIC (Application Specific Integrated Circuit) ซึ่งสามารถ sink และ source กระแสได้ประมาณ 12 mA ซึ่งในบางการ์ดอาจจะมากหรือน้อยกว่านี้ก็ได้ ดังนั้นในการออกแบบควรจะมีบัฟเฟอร์ไว้เป็นการดีที่สุด

Centronics

Centronics เป็นมาตราฐานแรก ๆ สำหรับการส่งผ่านข้อมูลจากเครื่องคอมพิวเตอร์ไปยังเครื่องพิมพ์ โดยอาศัยวิธีการแฮนด์เชคกับเครื่องพิมพ์ ซึ่งปกติแล้วจะใช้พอร์ตมาตราฐานแบบขนานและโปรแกรมควบคุมการทำงาน ลักษณะการส่งผ่านข้อมูลด้วย Centronics แสดงดังในรูป



รูปที่2.5.2 Centronics Handshake

จากรูปเมื่อต้องการส่งข้อมูลไปยังบัสข้อมูลที่ขา 2ถึง 9 ของพอร์ตขนาน เครื่องคอมพิวเตอร์จะตรวจสอบสัญญาณ Busy ของเครื่องที่พิมพ์ไม่ว่าง สัญญาณนี้จะเป็นโลจิก "0" โปรแกรมจะใส่สัญญาณ Strobe เข้าไปประมาณ 1 μSและจะนำสัญญาณ Strobe ออก ข้อมูลจะถูกอ่านในช่วงขอบขาขึ้นของสัญญาณ Strobe เครื่องพิมพ์จะแสดงสถานะ Busy ผ่านทาง Busy line เมื่อเครื่องพิมพ์รับข้อมูลเรียบร้อยแล้วจะตอบรับด้วยสัญญาณ Ack ซึ่งเป็นพัลส์ช่วงลบประมาณ 5 μS

เพื่อความรวดเร็วในการส่งข้อมูลบางครั้งจะละเลยสัญญาณ $\overline{\text{Ack}}$ เพื่อประหยัดเวลา ในช่วงหลังทีมี Extended Capabilities Port จะพบว่ามีโหมด Centronics ที่เร็วขึ้น ซึ่งจะต้องมีฮาร์ดแวร์ช่วยในการทำแฮนด์เชค

ตำแหน่งพอร์ต

พอร์ตข้อมูลแบบขนานจะใช้พอร์ต 3 พอร์ต ซึ่งได้แสดงไว้บนตารางที่ 2 พอร์ตหมายเลข 3 เป็นพอร์ตขนานที่อยู่บน Video Card ซึ่งใช้กันในช่วงแรก ๆ ตำแหน่งของพอร์ตขนานเราสามารถตรวจสอบได้โดยการอ่านค่าจาก BIOS (Basic Input/Output) ของเครื่องคอมพิวเตอร์

พอร์ตขนานพอร์ตแรกหรือ LPT1 ปกติจะกำหนดให้อยู่ที่หมายเลข 378h ขณะที่ LPT2 จะอยู่ที่พอร์ตหมายเลข 278h ตำแหน่งพอร์ตทั้งสองนี้จะใช้กันมากสำหรับพอร์ตขนาน ดังตัวอักษร h แสดงว่าเป็นเลขฐานสิบหก ตำแหน่งของพอร์ตเหลานี้อาจจะแตกต่างกันไปตามชนิดของคอมพิวเตอร์

ตารางที่ 2.5.2 ตำแหน่งพอร์ต

Address	Notes:
3 BCh	Used for Parallel Ports which were incorporated in to Video Cards and now,commonly an option for Ports controlled by BIOS-Doesn't support ECP address
378h-37Fh	Usual Address for LPT1
278h-27Fh	Usual Address for LPT2

เมื่อเปิดเครื่องคอมพิวเตอร์ BIOS จะกำหนดหมายเลขของพอร์ตที่มีอยู่บนเครื่องและกำหนดชื่อเรียก LPT1, LTP2 & LTP3 ให้กับพอร์ตเหล่านั้นในครั้งแรก BIOS จะตรวจสอบที่ตำแหน่ง 3BCh ถ้าพบพอร์ตที่ตำแหน่งนี้ก็จะกำหนดให้เป็น LPT1 หลังจากนั้นจึงตรวจสอบที่ตำแหน่ง 378h หากพบพอร์ตขนานอีกจะกำหนดด้วยชื่อเรียกของอุปกรณ์ตัวต่อไป ซึ่งอาจทำให้ผู้ใช้เกิดการสับสนขึ้นได้ ในบางการ์ดของพอร์ตขนานจะมีสายจัมพ์ ซึ่งผู้ใช้สามารถกำหนดพอร์ต LPT1, LTP2 & LTP3 ได้ ซึ่งบางครั้งตำแหน่ง 378h อาจจะไม่ใช้ LPT1 ก็ได้

โดยทั่วไปเราจะติดต่อผ่าน LPT1 แต่จะต้องหาตำแหน่ง ที่แน่นอนของพอร์ตให้ได้ ซึ่งสามารถทำได้โดยการตรวจสองผ่าน BIOSเมื่อ BIOS กำหนดตำแหน่งเครื่องพิมพ์แล้วจะเก็บตำแหน่งไว้ในหน่วยความจำ ตารางที่3 แสดงตำแหน่งในหน่วยความจำที่ BIOS เก็บตำแหน่งของเครื่องพิมพ์ไว้

ตารางที่ 2.5.3 แสดงตำแหน่ง LPT ที่เก็บไว้ใน BIOS

Start Address	Function
0000:0408	LTP1's Base Address
0000:040A	LTP2's Base Address
0000:040C	LTP3's Base Address
0000:040E	LTP4's Base Address

*ตำแหน่ง 0000:040E ในBIOS จะใช้เป็น Extended Bios Data Area ในเครื่องคอมพิวเตอร์แบบ PS/2 และ BIOS ของเครื่องคอมพิวเตอร์รุ่นใหม่ ๆ

รีจีสเตอร์ของพอร์ตมาตราฐานแบบขนาน (SPP)

Base address ของพอร์ตขนานปกติจะเป็นพอร์ตข้อมูลหรือ Data Register ซึ่งจะใช้ในการส่งข้อมูลไปยังพอร์ตขนาน (ขา 2-9) รีจีสเตอร์นี้ปกติจะเป็นพอร์ตที่เขียนได้อย่างเดียว แต่ถ้าอ่านข้อมูลจากพอร์ตนี้จะ ได้ข้อมูลจากพอร์นี้ครั้งสุดท้ายที่ส่งออกไปแต่ถ้าพอร์เป็นแบบสองทิศทางก็สามารถอ่านข้อมูลผ่านตำแหน่งนี้ได้เช่นกัน ในตารางที่ 4 แสดงพอร์ตข้อมูลของพอร์ตขนาน

ตารางที่ 2.5.4 พอร์ตข้อมูล

offset	Name	Read/Write	Bit No.	Properties
Base+0	Data Port	*Write	Bit 7	Data 7 (Pin 9)
			Bit 6	Data 6 (Pin 8)
			Bit 5	Data 5 (Pin 7)
			Bit 4	Data 4 (Pin 6)
			Bit 3	Data 3 (Pin 5)
			Bit 2	Data 2 (Pin 4)
			Bit 1	Data 1 (Pin 3)
			Bit 0	Data 0 (Pin 2)

*ถ้าพอร์ตเป็นแบบสองทิศทางจึงจะสามารถอ่านและเขียนข้อมูลได้

ตารางที่ 2.5.5 พอร์ตสถานะ

offset	Name	Read/Write	Bit No.	Properties
Base+1	Status Port	Read Only	Bit 7	Busy
			Bit 6	Ack
			Bit 5	Paper Out
			Bit 4	Select In
			Bit 3	Error
			Bit 2	IRQ (Not)
			Bit 1	Reserved
			Bit 0	Reserved

พอร์ตสถานะ (Base address+1)เป็นพอร์ตสำหรับอ่านเพียงอย่างเดียว ข้อมูลใด ๆ ที่ถูกเขียนลงไปที่พอร์ตนี้จะไม่มีอะไรเกิดขึ้นทั้งสิ้น พอร์ตสถานะจะประกอบด้วยสายอินพุตจำนวน 5 เส้น (ขา 10,11,12,13 และ 15) รีจีสเตอร์แสดงสถานะของ IRQ และอีก 2 บิตที่ถูกสงวนเอาไว้ Bit 7 เป็นอินพุตที่ active low ตัวอย่างเช่นอ่านข้อมูลเข้ามาแล้ว Bit 7 มีค่าเป็นโลจิก "0" หมายความว่าค่าจริง ๆ มีค่าเป็น โลจิก "1" หรือ +5V ที่ขา 11 เช่นเดียวกับ Bit 2 (IRQ) ถ้าบิตนี้มีค่าเป็น "1" แสดงว่าไม่มีการอินเตอร์รัพต์เกิดขึ้น

ตารางที่ 2.5.6 พอร์ตควบคุม

offset	Name	Read/Write	Bit No.	Properties
Base+2	Control	Read /write	Bit 7	Unused
	Port		Bit 6	Unused
			Bit 5	Enable bi-directional Port
			Bit 4	Enable IRQ via Ack Line
			Bit 3	Select Printer
			Bit 2	Initialize Printer (Reset)
			Bit 1	Auto Linefeed
			Bit 0	Strobe

พอร์ตควบคุม (Base address +2)เป็นพอร์ตสำหรับเขียนเพียงอย่างเดียว เมื่อต่อเครื่องพิมพ์เข้ากับพอร์ตขนาน จะใช้สัญญาณควบคุม 4 เส้นด้วยกันคือ Strobe, Auto Linefeed, Initialize และ

Select Printer ซึ่งสายเหล่านี้จะเป็นเอ้าท์พุตแบบopen collector ซึ่งจะมี 2 สถานะคือ ลอจิก "0" และ high impedance (เปิดวงจร)

โดยปกติ I/O การ์ดจะมีตัวต้านทาน Pull-up ซึ่งก็อาจจะไม่มีทั้งหมด เพื่อให้วงจรทำงานได้อย่างถูกต้องจึงควรมีตัวต้านทานภายนอกด้วย โดยค่าที่ใช้กันคือ 4.7 K เมื่อต่อตัวต้านทานแล้วจะทำให้สามารถใช้ 4 บิทของพอร์ตควบคุมเป็นตัวรับส่งข้อมูลในแบบสองทิศทางได้ ซึ่งพอร์ตควบคุมจะต้องกำหนดให้เป็น xxxx0100 เพื่อให้สามารถอ่านข้อมูลได้

บิตที่ 4 และ 5 เป็นการควบคุมจากภายใน บิตที่ 4 จะเป็นตัวอีนาเบิล (Enable) สัญญาณ IRQ ส่วนบิตที่ 5 จะอีนาเบิลพอร์ตให้เป็นพอร์ตแบบสองทิศทาง สำหรับบิตที่ 6 และ 7 ไม่ได้นำมาใช้งาน

ตัวอย่างโปรแกรมการหาตำแหน่งพอร์ตขนาน

```
#Include <stdio.h>
#Include <dos.h>
void main(void)
{
        unsigned int far *prtraddr;
        unsigned int address;
        int a;
        ptraddr=(unsigned int far *)0x00000408;
        for(a=0;a<3;a++)
        {
           address=*ptraddr;
           if(address==0)
                printf("No port found for LPT%d\n",a+1);
           else
                printf("Address assigned to LPT%d is %xH\n",a+1,address);
           *ptraddr++;
        }
}
```

ตัวอย่างโปรแกรมการควบคุมผ่านพอร์ตขนาน

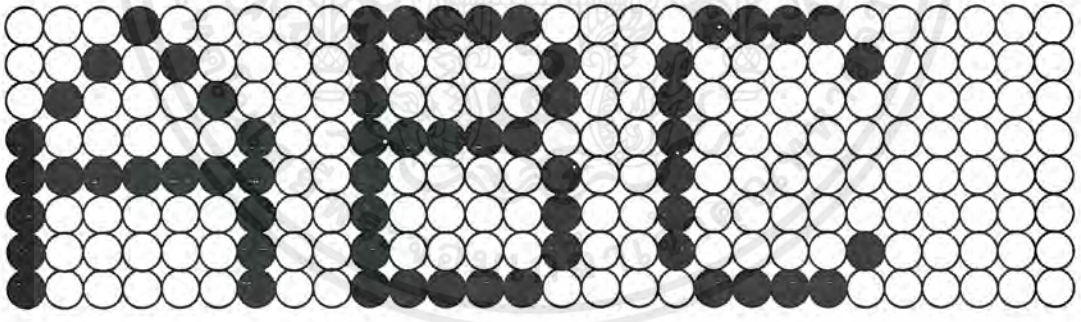
```
#include <dos.h>
#define PORTADDRESS 0x378
#define DATA PORTADDRESS+0
#define STATUS PORTADDRESS+1
#define CONTROL PORTADDRESS+2
void main(void)
{
        outportb(CONTROL,inportb(CONTROL)&&0xDF);
        outportb(DATA,0xFF);
        delay(100);
        outportb(DATA,0xF0);
        delay(100);
        outportb(DATA,0x0F);
        delay(100);
        outportb(DATA,0x00);
        delay(100);
}
```

บทที่ 3
วิธีการดำเนินงาน

เนื่องจากในปัจจุบันการโฆษณาเป็นสิ่งจำเป็นอย่างยิ่งสำหรับธุรกิจที่เกี่ยวข้องกับการขายและการบริการ อุปกรณ์ที่ใช้เป็นสื่อจึงจำเป็นต้องมีความทันสมัย และแปลกใหม่เพื่อดึงดูดความสนใจจากผู้คนให้ได้มากที่สุด ดังนั้นในการทำโครงงานนี้มีจุดมุ่งหมาย เพื่อที่จะสร้างเครื่องมือที่ใช้เป็นสื่อโฆษณาที่แปลกตาแปลกใจ แก่ผู้ที่พบเห็น และมีราคาถูก

3.1 หลักการทำงาน

การทำงานของโครงงานโดยรวมจะประกอบด้วย ส่วนที่เป็นฮาร์ดแวร์และซอฟแวร์ ส่วนของทางด้านฮาร์ทแวร์จะมีหลักการทำงานคือ ทำการรับรหัสรูปแบบตัวอักษรต่างๆ จากเครื่องคอมพิวเตอร์ผ่านทางพอร์ทขนานมาเก็บไว้ในหน่วยความจำ และทำหน้าที่ควบคุมการสแกนของแสงเลเซอร์ โดยนำข้อมูลที่มีอยู่ในหน่วยความจำมาควบคุมการติดดับของ เลเซอร์ไดโอด ทั้ง 8 ตัว ทำให้เกิดเป็นตัวอักษรขึ้นบนฉากที่แสงเลเซอร์ตกกระทบตัวอย่างดังรูป



รูปที่ 3.1 รูปแบบการแสดงตัวอักษรบนฉาก

ในส่วนการทำงานของ ซอร์ฟแวร์ โดยรวมแล้ว ซอร์ฟแวร์มีหน้าที่หลักในการสร้างรหัสตัวอักษรต่างๆ ที่ใช้จะใช้แสดงบนฉาก และควบคุมการทำงานต่างๆ ของส่วนฮาร์ดแวร์โดยผู้ใช้โปรแกรมเอง เช่นการ รีเซต การทดสอบเครื่อง การส่งรหัสตัวอักษรต่างๆ ที่ได้ออกแบบไว้ผ่านพอร์ทขนานไปยัง ฮาร์ดแวร์

เพื่อให้การทำงานของฮาร์ดแวร์ดำเนินไปได้อย่างราบรื่น จึงจำเป็นต้องให้มีการโต้ตอบกันระหว่างฮาร์ดแวร์และซอร์ฟแวร์อยู่เสมอ พื่อบอกสถานะการทำงานของด้านฮาร์ดแวร์ ดังนั้นใน

ส่วนของฮาร์ดแวร์จะต้องมีการส่งข้อมูลแสดงสถานะกลับไปยังเครื่องคอมพิวเตอร์ โดยในโครงงานนี้ได้กำหนดตัวเลขฐานสิบหกแสดงสถานะดังนี้

Status Code	Status
10h	Standby
20h	รับข้อมูล
40h	Self Test
80h	Scan
F0h	Error

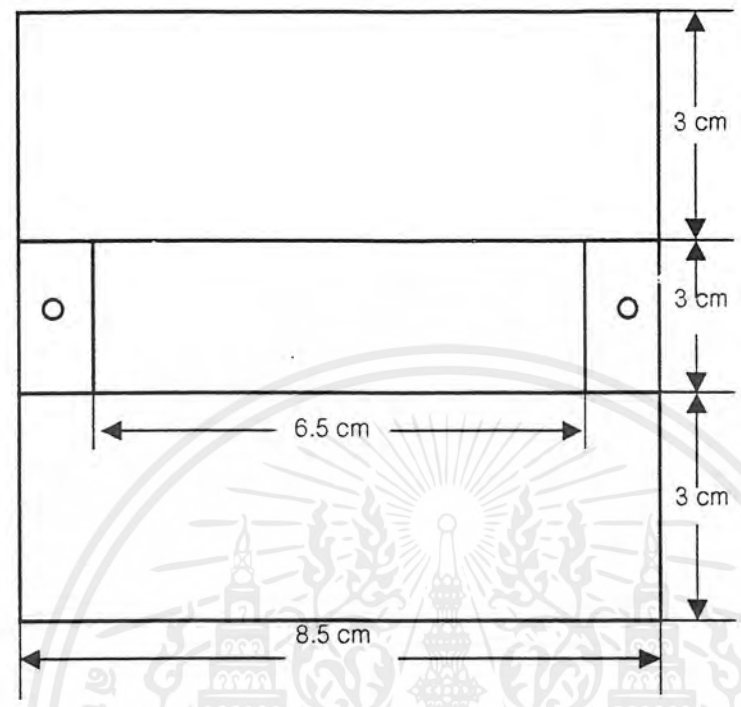
ตารางที่ 3.1 แสดง Status Code

3.2 หลักการออกแบบฮาร์ดแวร์

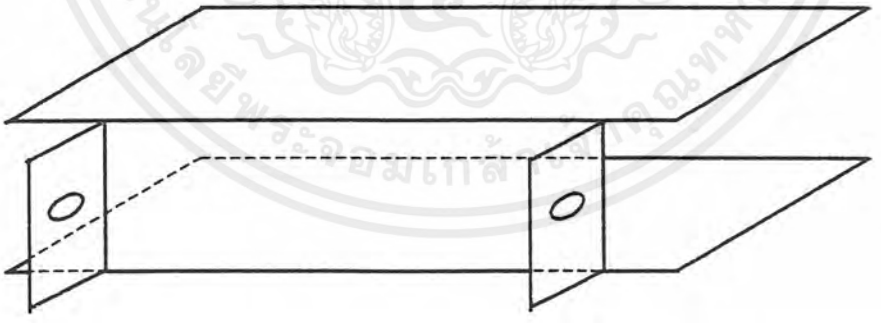
ฮาร์ดแวร์จะเป็นส่วนที่ยากที่สุดในการทำโครงงาน เนื่องจากเป็นสิ่งที่ใหม่และไม่มีการค้นคว้ามาก่อน และต้องอาศัยการทำงานร่วมกันทั้งทางด้าน อิเล็คทรอนิกส์และทางด้านเครื่องกล จึงต้องใช้ทักษะหลายด้านมาประยุกต์ใช้

3.2.1 หลักการออกแบบส่วนของการสแกน

ในส่วนนี้จะต้องใช้วัสดุที่มีขนาดเบา แข็ง เมื่อพับแล้วคงรูปได้ดี ในโครงงานนี้จะใช้อะลูมิเนียม เป็นวัสดุในการทำ การพับจะแยกออกเป็น 2 ส่วน คือส่วนของตัวยึดแผงเลเซอร์ไดโอด และส่วนของตัวที่ใช้ในการส่ายแผงเลเซอร์ไดโอด ประกอบด้วย มอเตอร์และตัวยึดมอเตอร์ มอเตอร์ที่ใช้นี้จะต้องมีความเร็วสูงและน้ำหนักเบา วิธีการพับและขนาดของด้านต่างๆ แสดงไว้ดังรูปที่ 3.2 และ รูปที่ 3.3

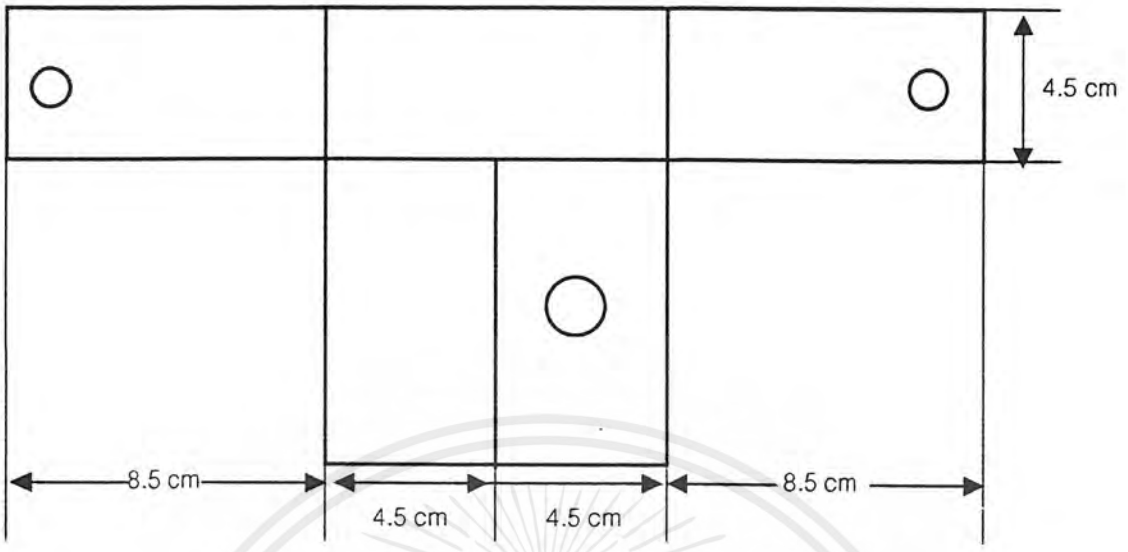


ก. ขนาดด้านต่างๆ

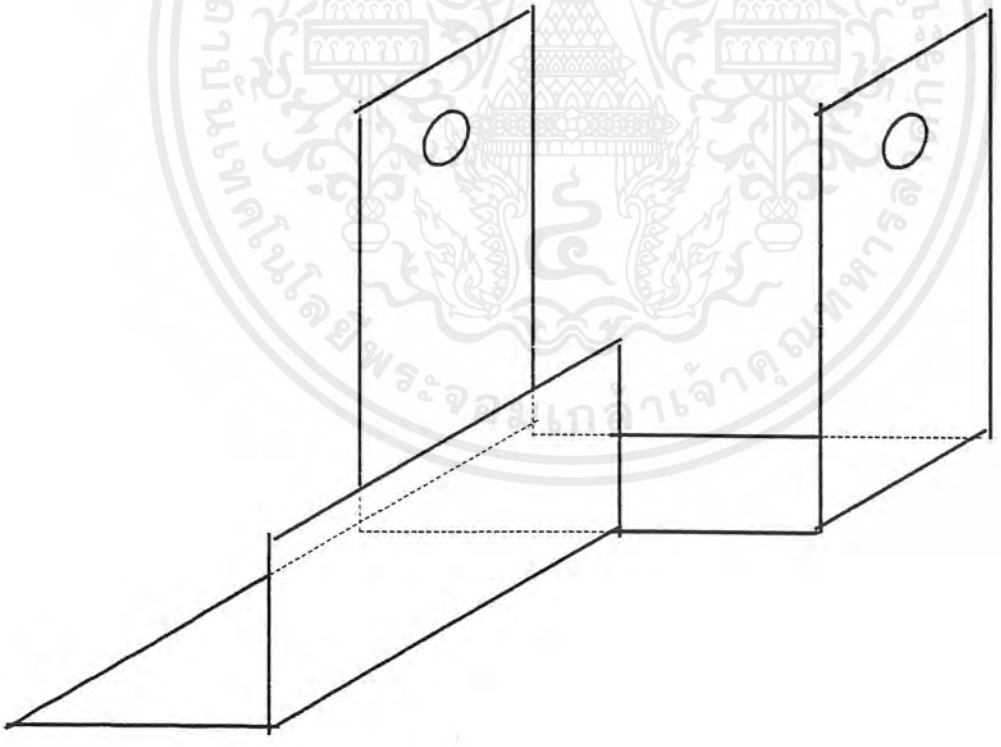


ข. ลักษณะเมื่อพับเสร็จแล้ว

รูปที่ 3.2 แสดง ขนาดและลักษณะของส่วนยึดแผงเลเซอร์ไดโอด



ก. ขนาดด้านต่างๆ



ข. ลักษณะเมื่อพับเสร็จแล้ว

รูปที่ 3.3 แสดง ขนาดและลักษณะของส่วนยึดมอเตอร์

3.2.2 หลักการออกแบบส่วนของ ไมโครคอนโทรเลอร์

ในการออกแบบไมโครคอนโทรเลอร์ เพื่อควบคุมการสแกนและติดต่อเชื่อมโยงกับเครื่องคอมพิวเตอร์ จะต้องคำนึงถึง ความเร็วในการส่งและรับข้อมูล ให้เร็วที่สุด เพื่อที่จะได้ไม่เกิดการกระตุกของตัวอักษรขณะแสดงอยู่ ดั้งนั้นจึงจำเป็นต้องใช้การสื่อสารผ่าน พอร์ทขนาน แทนที่จะใช้การสื่อสารทางพอร์ทอนุกรม และจะต้องคำนึงถึงสัญญาณรบกวนต่างๆ ที่จะเกิดขึ้นจากเครื่องคอมพิวเตอร์ในแผ่นวงจรพิมพ์ จะต้องมีการลดสัญญาณรบกวนโดยการต่อ คาปาซิเตอร์ คร่อมตรงขาไฟเลี้ยงของตัวไอซีทุกตัว

3.3 การออกแบบ ซอร์ฟแวร์

ซอร์ฟแวร์ที่ใช้จะแบ่งออกเป็น 2 ส่วนคือ ส่วนของภาษา MCS-51 Assembly ที่ใช้อ่านข้อมูลและควบคุมการทำงานด้านการสแกน และการติดต่อกับเครื่องคอมพิวเตอร์ และส่วนของภาษา C ที่ใช้ในการออกแบบตัวอักษร การส่งข้อมูล รวมทั้งการควบคุมการทำงานต่างๆ ของทางด้านฮาร์ดแวร์

ส่วนของภาษา Assembly ที่ใช้ในการควบคุมการทำงานของ MCS-51 สามารถแบ่งออกเป็นส่วนต่างๆ ได้ดังต่อไปนี้

- ส่วนของการรับข้อมูลจากเครื่องคอมพิวเตอร์
- ส่วนของการควบคุมการสแกนของแสงเลเซอร์
- ส่วนของการแสดงสถานะให้เครื่องคอมพิวเตอร์ทราบ

ส่วนของภาษา C เป็นส่วนควบคุมการทำงานต่างๆของฮาร์ดแวร์ รวมทั้งการออกตัวอักษรต่างๆ โดยแบ่งออกเป็นส่วนต่างๆ ดังต่อไปนี้

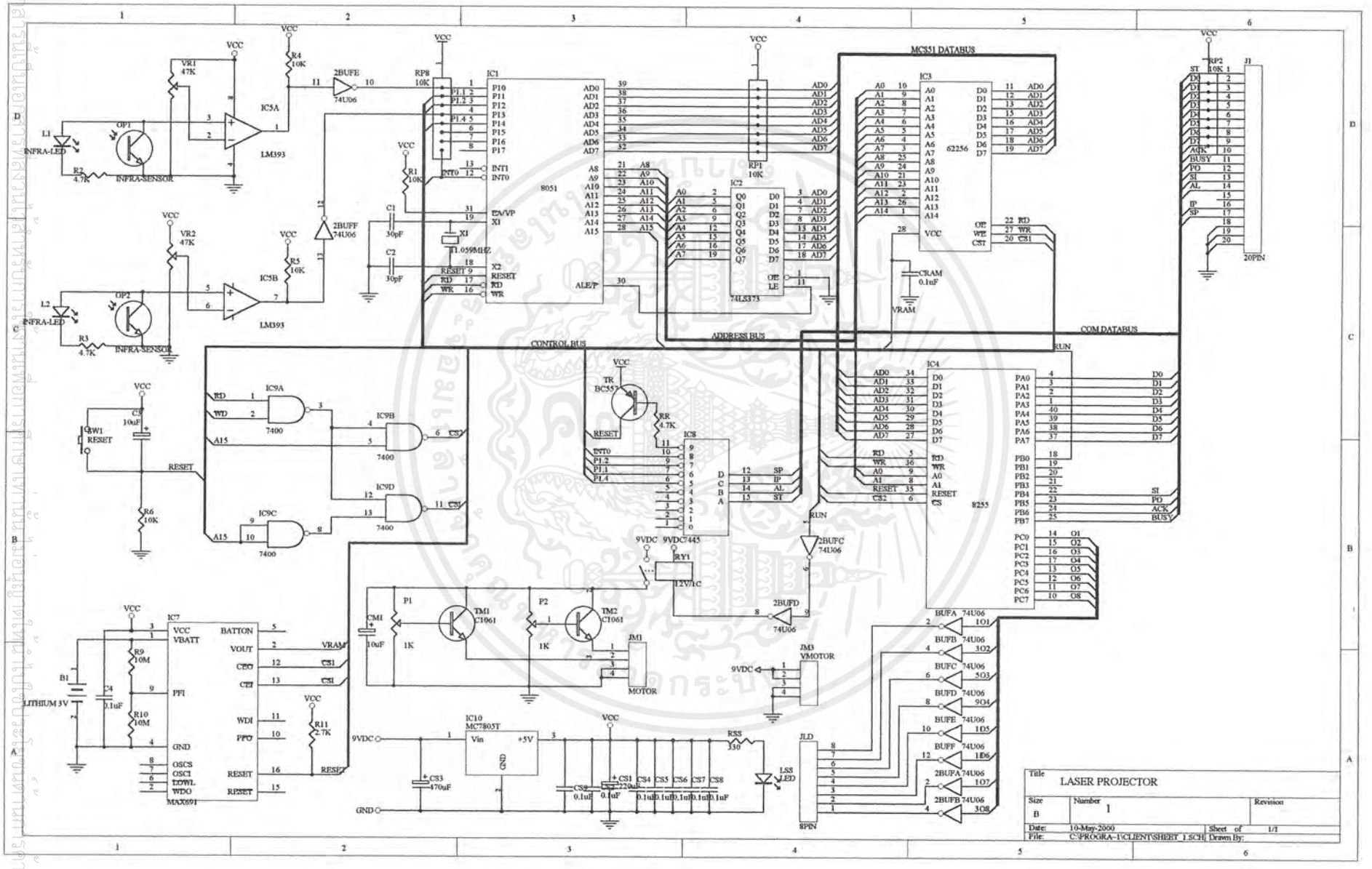
- ส่วนของการออกแบบตัวอักษร
- ส่วนของการแปลงตัวอักษรที่ออกแบบไว้ให้เป็นรหัสที่ฮาร์แวร์เข้าใจได้ แล้วบันทึกลงฮาร์ดดิสก์
- ส่วนของการติดต่อสื่อสารกับฮาร์ดแวร์

เอกสารอ้างอิง

1. ผศ.ศุภชัย สุรินทร์วงค์ เครื่องกลไฟฟ้า 1 ตอน 2 มอเตอร์ไฟฟ้ากระแสตรง สมาคมส่งเสริมเทคโนโลยี(ไทย-ญี่ปุ่น)

2. วรพล ลีลาเกียรติสกุล ,พิเชษฐ์ ช่อผกา,ฐิติรัตน์ สมเพชร ระบบไมโครคอมพิวเตอร์ PC/XT 8088, หน้าที่ 209

ภาคผนวก


MC351 DATABUS
ADDRESS BUS
CONTROL BUS
COM DATABUS
RUN
VCC
GND
IC1
8051
IC2
74LS373
IC3
62256
IC4
8255
IC5A
IC5B
LM393
IC7
MAX691
IC8
IC9A
IC9B
IC9C
IC9D
7400
IC10
MC7805T
2BUFE
74U06
2BUFF
74U06
2BUFC
74U06
2BUFD
74U06
BUFA 74U06
BUFB 74U06
BUFC 74U06
BUFD 74U06
BUFE 74U06
BUFF 74U06
2BUFA 74U06
2BUFB 74U06
VR1 47K
VR2 47K
R1 10K
R2 4.7K
R3 4.7K
R4 10K
R5 10K
R6 10K
R9 10M
R10 10M
R11 2.7K
RR 4.7K
RSS 330
RP1 10K
RP2 10K
RP8 10K
C1 30pF
C2 30pF
C3 10uF
C4 0.1uF
CM1 10uF
CS1 220uF
CS2 0.1uF
CS3 470uF
CS4 0.1uF
CS5 0.1uF
CS6 0.1uF
CS7 0.1uF
CS8 0.1uF
CS9 0.1uF
CRAM 0.1uF
X1 11.059MHZ
P1 1K
P2 1K
TM1 C1061
TM2 C1061
TR BC557
OP1
OP2
INFRA-LED
INFRA-SENSOR
SW1 RESET
B1 LITHIUM 3V
RY1 12V/1C
9VDC
9VDC445
JM1 MOTOR
JM3 VMOTOR
JLD 8PIN
LSS LED
J1 20PIN
Title LASER PROJECTOR
Size B
Number 1
Revision
Date: 10-May-2000
Sheet of 1/1
File: C:\PROGRA~1\CLIENT\SHEET_1.SCH
Drawn By:

```
#include <dos.h>
#include <stdarg.h>
#include <graphics.h>
#include <stdlib.h>
#include <stdio.h>
#include <conio.h>
#include <bios.h>
#include <ctype.h>
#include <string.h>

#define L_ARROW 0x4b00
#define R_ARROW 0x4d00
#define U_ARROW 0x4800
#define D_ARROW 0x5000
#define ENTER   0x1c0d
#define ESC 0x11b
#define ALL 1000


int   GraphDriver;        /* The Graphics device driver       */
int   GraphMode;          /* The Graphics mode value                  */
double AspectRatio;       /* Aspect ratio of a pixel on the screen*/
int   MaxX, MaxY;         /* The maximum resolution of the screen */
int   MaxColors;          /* The maximum # of colors available      */
int   ErrorCode;          /* Reports any graphics errors              */
int midx, midy,num,c=5;
int xradius = c, yradius = c;
int mx=270,my=190;
int tdata[ALL*8];
unsigned int size,base;
void *Circle,*Cir;

void sendto(int,int data[]);
void outfile(int code,int data[]);
void loadfile(void);
void Initialize(void);
int  startwindows(void);
void SayGoodbye(void);
void Pause(void);
void MainWindow(char *header);
void StatusLine(char *msg);
void DrawBorder(void);
void changetextstyle(int font, int direction, int charsize);
void draw(int);
void draw2(void);
void save(int data[]);
void design(void);
void rec_draw(void);
```

```
void send(void);
void reset(void);
void run(void);
int  gprintf(int *xloc, int *yloc, char *fmt, ... );

void main(void)
{
  int sw;
  Initialize();             /* Set system into Graphics mode */
  do{
     sw = startwindows();
     switch(sw)
      {
         case 209: design();      break;
         case 179: send();      break;
         case 239: reset();       break;
         case 269: run();         break;
      }
   }while(sw!=299);
  SayGoodbye();
  closegraph();             /* Return the system to text mode */
}

void Initialize(void)
{
  int xasp, yasp;                   /* Used to read the aspect ratio*/
  unsigned int far *ptraddr;
  unsigned int address;

  ptraddr=(unsigned int far *)0x00000408;  /* DETECT LPT1 */
  base=*ptraddr;
  if(base==0)
   {
    printf("Device has not LPT1.");
    exit(0);
   }

  GraphDriver = DETECT;             /* Request auto-detection   */
  initgraph( &GraphDriver, &GraphMode, "" );
  ErrorCode = graphresult();                /* Read result of initialization*/
  if( ErrorCode != grOk ){              /* Error occured during init */
    printf(" Graphics System Error: %s\n", grapherrormsg( ErrorCode ) );
    exit( 1 );
  }
  MaxColors = getmaxcolor() + 1; /* Read maximum number of colors*/
  MaxX = getmaxx();
  MaxY = getmaxy();                 /* Read size of screen                */
  getaspectratio( &xasp, &yasp );  /* read the hardware aspect           */
```

```
  AspectRatio = (double)xasp / (double)yasp; /* Get correction factor        */
}

int startwindows(void)
 {
  struct viewporttype viewinfo;
  int h, w,key,a,b;
  cleardevice();
  rec_draw();
  size=imagesize(10,10,150,30);
  Cir =malloc(size);
  getimage(10,10,150,30,Cir);
  cleardevice();
  MainWindow( "Main Windows" );
  getviewsettings( &viewinfo );      /* Read viewport settings   */
  changetextstyle( 1, HORIZ_DIR, 2 );
  settextjustify( CENTER_TEXT, CENTER_TEXT );
  setcolor(2);
  h = viewinfo.bottom-340;
  w = viewinfo.right  - viewinfo.left;
  outtextxy( w/2, h, "CHOICE" );
  setcolor(5);
  outtextxy( w/2, h+60, "SEND FONT");
  outtextxy( w/2, h+90, "DESIGN FONT");
  outtextxy( w/2, h+120, "RESET");
  outtextxy( w/2, h+150, "RUN");
  outtextxy( w/2, h+180, "EXIT");
  StatusLine( "LASER PROJECTOR" );
  w=w/2-70;
  b=h+53;
  h=h+53;
    do{
      putimage(w,h,Cir,XOR_PUT);
      while(bioskey(1)==0);
      putimage(w,h,Cir,XOR_PUT);
      key=bioskey(0);
      if(key==D_ARROW)  h+=30;
      if(key==U_ARROW)  h-=30;
      if(h<b)    h=b+120;
      if(h>=b+150) h=b;
      }while(key!=ENTER);
  free(Cir);
  return h;
}
```

```
void rec_draw(void)
{
 setfillstyle(1,1);
 setcolor(1);
 bar( 10, 10, 150,40 );        /* Draw the rectangle                */
 rectangle(10,10,150,40);
}

void design(void)
{

  int mode;
  int key,a=0,b,d,c=5;
  int data_x[640],data_y[200];
  int dx[8]={0,0,0,0,0,0,0,0};
  int dy[8]={0,0,0,0,0,0,0,0};
  draw(2);
  size=imagesize(0,0,2*c,2*c);
  Circle=malloc(size);
  getimage(0,0,2*c,2*c,Circle);
  draw(4);
  size=imagesize(0,0,2*c,2*c);
  Cir=malloc(size);
  getimage(0,0,2*c,2*c,Cir);
  cleardevice();
  MainWindow( "Font Design" );
  StatusLine( "Press Esc to EXIT" );
  draw2();
  midx=mx-5;
  midy=my-5;
   do{
     do{
          putimage(midx,midy,Circle,XOR_PUT);
          delay(100);
          putimage(midx,midy,Circle,XOR_PUT);
          }while(bioskey(1)==0);
     key=bioskey(0);
     if(key==R_ARROW)  midx+=10;
     if(key==L_ARROW)  midx-=10;
     if(key==D_ARROW)  midy+=10;
     if(key==U_ARROW)  midy-=10;
     if(key==ENTER)
          {
          putimage(midx,midy,Cir,XOR_PUT);
          data_x[a]=(midx+5)/10-26;
          data_y[a]=(midy+5)/10-18;
          a++;
          }
```

```
    if(midx<mx-5)    midx=mx-5;
    if(midx>mx+65)   midx=mx+65;
    if(midy<my-5)    midy=my-5;
    if(midy>my+65)   midy=my+65;
    }while(key!=ESC);
free(Cir);
free(Circle);
for(c=0;c<a;c++)
 for(b=c+1;b<a;b++)
 {
  if((data_x[c]==data_x[b])&&(data_y[c]==data_y[b]))
   {
        data_x[b]=1000;
        data_y[b]=1000;
        data_x[c]=1000;
        data_y[c]=1000;
   }
 }
data_x[a]=1000;
data_y[a]=1000;

for(c=0;c<a;c++)
 for(b=c+1;b<a;b++)
  if(data_x[b]<data_x[c])
   {
    d=data_x[c];
    data_x[c]=data_x[b];
    data_x[b]=d;
    d=data_y[c];
    data_y[c]=data_y[b];
    data_y[b]=d;
   }
a=0;
while(data_x[a]!=1000)
 {
  if(data_x[a]==1) dx[0]++;
  if(data_x[a]==2) dx[1]++;
  if(data_x[a]==3) dx[2]++;
  if(data_x[a]==4) dx[3]++;
  if(data_x[a]==5) dx[4]++;
  if(data_x[a]==6) dx[5]++;
  if(data_x[a]==7) dx[6]++;
  if(data_x[a]==8) dx[7]++;
  a++;
 }
d=0;
for(b=0;b<a;b+=dx[d],d++)
 for(c=b;c<dx[d]+b;c++)
```

```
   {
    if(data_y[c]==1) dy[d]|=0x01;        //set bit 0
    if(data_y[c]==2) dy[d]|=0x02;        //set bit 1
    if(data_y[c]==3) dy[d]|=0x04;        //set bit 2
    if(data_y[c]==4) dy[d]|=0x08;        //set bit 3
    if(data_y[c]==5) dy[d]|=0x10;        //set bit 4
    if(data_y[c]==6) dy[d]|=0x20;        //set bit 5
    if(data_y[c]==7) dy[d]|=0x40;        //set bit 6
    if(data_y[c]==8) dy[d]|=0x80;        //set bit 7
   }
restorecrtmode();
  printf( "Font data (HEX): ");
  for(d=0;d<8;d++)
    printf("%XH ",dy[d]);
  printf("\n\nInput font code: " );
do{
  d=key;
  key=bioskey(0);
  gotoxy(18,3);
  printf("%c",key);
  }while(key!=ENTER);
  outfile(d,dy);
  mode = getgraphmode();
  setgraphmode( mode );
 }

void draw(int colr)
{
 setcolor(colr);
 setfillstyle(1,colr);
 fillellipse(c,c ,c,c);
}

void draw2(void)
{
 setcolor(1);
 setfillstyle(1,0);
  for(midx=mx;midx<350;midx+=10)
   for(midy=my;midy<270;midy+=10)
    fillellipse(midx, midy, xradius, yradius);
}

void SayGoodbye(void)
{
  struct viewporttype viewinfo;       /* Structure to read viewport       */
  int h, w;
  MainWindow( "== Finale ==" );
  getviewsettings( &viewinfo );       /* Read viewport settings   */
```

```
  changetextstyle( TRIPLEX_FONT, HORIZ_DIR, 4 );
  settextjustify( CENTER_TEXT, CENTER_TEXT );
  h = viewinfo.bottom - viewinfo.top;
  w = viewinfo.right  - viewinfo.left;
  setcolor(2);
  outtextxy( w/2, h/2, "KMIT'L RESEARCH" );
  StatusLine( "LASER PROJECTOR" );
  delay(1500);
  cleardevice();                    /* Clear the graphics screen       */
}


void MainWindow( char *header )
{
  int height;
  cleardevice();                    /* Clear graphics screen    */
  setcolor( MaxColors - 1 );        /* Set current color to white      */
  setviewport( 0, 0, MaxX, MaxY, 1 );     /* Open port to full screen  */
  height = textheight( "H" );       /* Get basic text height       */
  changetextstyle( DEFAULT_FONT, HORIZ_DIR, 1 );
  settextjustify( CENTER_TEXT, TOP_TEXT );
  outtextxy( MaxX/2, 2, header );
  setviewport( 0, height+4, MaxX, MaxY-(height+4), 1 );
  DrawBorder();
  setviewport( 1, height+5, MaxX-1, MaxY-(height+5), 1 );
}

void StatusLine( char *msg )
{
  int height;
  setviewport( 0, 0, MaxX, MaxY, 1 );     /* Open port to full screen  */
  setcolor( MaxColors - 1 );        /* Set current color to white      */
  changetextstyle( DEFAULT_FONT, HORIZ_DIR, 1 );
  settextjustify( CENTER_TEXT, TOP_TEXT );
  setlinestyle( SOLID_LINE, 0, NORM_WIDTH );
  setfillstyle( EMPTY_FILL, 0 );
  height = textheight( "H" );       /* Detemine current height     */
  bar( 0, MaxY-(height+4), MaxX, MaxY );
  rectangle( 0, MaxY-(height+4), MaxX, MaxY );
  outtextxy( MaxX/2, MaxY-(height+2), msg );
  setviewport( 1, height+5, MaxX-1, MaxY-(height+5), 1 );

}
```

```
void DrawBorder(void)
{
  struct viewporttype vp;
  setcolor( MaxColors - 1 );          /* Set current color to white          */
  setlinestyle( SOLID_LINE, 0, NORM_WIDTH );
  getviewsettings( &vp );
  rectangle( 0, 0, vp.right-vp.left, vp.bottom-vp.top );
}

void changetextstyle(int font, int direction, int charsize)
{
  int ErrorCode;
  graphresult();                      /* clear error code                    */
  settextstyle(font, direction, charsize);
  ErrorCode = graphresult();                /* check result                  */
  if( ErrorCode != grOk ){            /* if error occured                    */
    closegraph();
    printf(" Graphics System Error: %s\n", grapherrormsg( ErrorCode ) );
    exit( 1 );
  }
}

int gprintf( int *xloc, int *yloc, char *fmt, ... )
{
  va_list  argptr;                    /* Argument list pointer      */
  char str[140];                      /* Buffer to build sting into  */
  int cnt;                            /* Result of SPRINTF for return */
  va_start( argptr, fmt );            /* Initialize va_ functions    */
  cnt = vsprintf( str, fmt, argptr ); /* prints string to buffer     */
  outtextxy( *xloc, *yloc, str );     /* Send string in graphics mode */
  *yloc += textheight( "H" ) + 2;      /* Advance to next line        */
  va_end( argptr );                   /* Close va_ functions                 */
  return( cnt );                      /* Return the conversion count     */
}

void send(void)
{
  int a,b,c,dd=0,p,key=0,error,data[10];
  struct viewporttype viewinfo;
  int h,mode,gdata[ALL/2],ddat[ALL/2];
  int P=0,x,y,z;
  mode = getgraphmode();
  restorecrtmode();
  clrscr();
  printf("Out Code       :\n\n");
  printf("Input Font Code :\n\n");
  printf("OK press Enter.\nSEND and EXIT press ESC\n");
```

```
do{
    b=key;
    gotoxy(19,3);
    key=bioskey(0);

    if((key==ENTER)&&(b!=ENTER))
     {
        gotoxy(dd+19,1);
        printf("%c",b);
        gdata[dd]=b;
        dd++;
     }
     printf("%c",key);
    }while((key!=ESC)&&(a<ALL));
  loadfile();

  for(x=0;x<dd;x++)
    for(y=0;y<num;y+=9)
     {

      if(gdata[x]==tdata[y])
         {
          for(p=y+1;p<y+9;p++)
          {
           ddat[P]=tdata[p];
           P++;
          }
         }
     }

  outportb(base+2,0x3);          //INT0
  outportb(base+2,0xc);          //P1.1
  delay(2);
  sendto(P,ddat);
  outportb(base+2,0x4);
  printf("Send Completed.");
  getch();
  setgraphmode( mode );
  cleardevice();
}

void sendto(int P,int data[])
{
  int a,b,c,d,p,error;
  clrscr();
  for(p=0;p<P;p++)
    {
    outportb(base,data[p]);
```

```
    outportb(base+2,0xd);    //P1.2
    delay(1);
    outportb(base+2,0xc);    //P1.1
    delay(1);
    if(!((p+1)%8))
     for(a=0;a<3;a++)
     {
      outportb(base,0);
      outportb(base+2,0xd);
      delay(1);
      outportb(base+2,0xc);
      delay(1);
     }
   }
    outportb(base,0X77);
    outportb(base+2,0xd);
    delay(1);
    outportb(base+2,0xc);
    delay(1);

}


void outfile(int code,int dat[])
{
 FILE *fp;
 int pt,fdata[ALL*8],fcode[ALL];
 int a=0,b=1,n=1,d;
 num=0;
 fp=fopen("font.txt","a+w+r");
 fscanf(fp,"%X",&fcode[a]);
 while(!feof(fp)&&b<ALL*8)
  {
   if(fcode[a]==code)
     {
         fprintf(fp,"\n%X\n",code);
         for(pt=0;pt<8;pt++)
         fprintf(fp,"%X ",dat[pt]);
         n=0;
         code=0;
      }
    if(!(b%9))
    {
     a++;
     fscanf(fp,"%X",&fcode[a]);
     b++;
    }
```

```
    else
     {
      fscanf(fp,"%X",&fdata[b-1]);
      b++;
     }
    num++;
   }
 if(n)
 {
  fprintf(fp,"\n%X\n",code);
  for(pt=0;pt<8;pt++)
  fprintf(fp,"%X ",dat[pt]);
 }
 fclose(fp);
}

void loadfile(void)
{
 FILE *fp;
 int a=0;
 num=0;
 fp=fopen("font.txt","r");
 fscanf(fp,"%X",&tdata[a]);
 while(!feof(fp)&&a<ALL*9)
  {
     a++;
     fscanf(fp,"%X",&tdata[a]);
     num++;
  }
 fclose(fp);
}

void reset(void)
 {
  outportb(base+2,2);
  delay(10);
  outportb(base+2,4);
 }

void run(void)
{
  outportb(base+2,0xe);
  delay(1);
  outportb(base+2,4);
}
```

```
CONP  EQU 0FA03H
PORTA EQU 0FA00H
PORTB EQU 0FA01H
PORTC EQU 0FA02H
A_END EQU 0000H
SHIP  EQU 2H
ORG 0000H

              SJMP START
              NOP
              LJMP GET_D          ;int0 vector 0003

START:        MOV R0,#10H
              MOV R1,#11H
              MOV @R0,#00H         ;ADDRESS MEM
              MOV @R1,#66          ;
              MOV R0,#12H
              MOV R1,#13H
              MOV @R0,#00H
              MOV @R1,#02H
              MOV R0,#64
              MOV DPTR,#A_END+2
              MOV A,#0H
ST:           MOVX @DPTR,A
              INC DPTR
              DJNZ R0,ST
              SETB EA
              SETB EX0
              CLR IE0
              MOV R6,#SHIP
              MOV R5,#00H
              MOV DPTR,#CONP
              MOV A,#90H
              MOVX @DPTR,A
              MOV R3,#0H
              MOV R4,#08H
              MOV R7,#00H

W_RUN:        MOV P1,#0FFH    ;WAIT RUN
              MOV DPTR,#PORTB
              MOV A,#0FFH
              MOVX @DPTR,A
              JB P1.4,W_RUN
              SJMP S3

G_PS:         MOV R3,#0H
              SJMP SS
```

```
S3:     CJNE R3,#64,G_PS
        SJMP PSH

SS:     MOV P1,#0FFH
        MOV R2,#0H
SS2:    DJNZ R6,LSHIP
        LCALL DP
        MOV R6,#SHIP

LSHIP:  MOV R0,#12H
        MOV R1,#13H
        MOV DPH,@R0     ;MOV DPH,R2 BANK0
        MOV DPL,@R1     ;MOV DPH,R3 BANK0

S2:     MOVX A,@DPTR
        CJNE A,#77H,SS3
        MOV A,#00H
SS3:    PUSH DPH
        PUSH DPL

S1:     MOV DPTR,#PORTC
        MOVX @DPTR,A
        MOV DPTR,#PORTB
        MOV A,#0F0H
        MOVX @DPTR,A
        POP DPL
        POP DPH
        INC DPTR

INN:    LCALL DELAY

L2:     MOV P1,#0FFH
        JB P1.3,INCR
        MOV DPTR,#PORTC
        MOV A,#0
        MOVX @DPTR,A
        LCALL DEBUG
        MOV P1,#0FFH
        JNB P1.3,$-3
        SJMP S3

PSH:    MOV DPTR,#PORTC
        MOV A,#0H
        MOVX @DPTR,A
        MOV P1,#0FFH
        JB P1.3,$-3
        MOV P1,#0FFH
        MOV R3,#00H
```

```
                SJMP L2

INCR:           INC R2
                INC R3
                CJNE,R2,#64,S2
                SJMP S3

GET_D:          CLR EX0         ;DISABLE INTRUPT
                PUSH PSW
                PUSH ACC
                PUSH DPH
                PUSH DPL
                SETB RS0
                SETB RS1
                MOV R0,#010H
                MOV R1,#011H
                MOV DPTR,#PORTC
                MOV A,#00H
                MOVX @DPTR,A
RELC:           MOV P1,#0FFH    ;WAIT WR SIGNAL
                JNB P1.4,EXIT
                JB P1.1,RELC
                MOV DPTR,#PORTA
                MOVX A,@DPTR
                MOV DPH,@R0
                MOV DPL,@R1
                MOVX @DPTR,A
                MOV P1,#0FFH
                JNB P1.1,$-3        ;WAIT END WR
                INC DPTR
                MOV @R0,DPH
                MOV @R1,DPL
                MOV P1,#0FFH
                JNB P1.2,RELC
EXIT:           MOV R0,#64
                MOV A,#0H
CLRMEM:         MOVX @DPTR,A
                INC DPTR
                DJNZ R0,CLRMEM
                MOV DPTR,#A_END
                MOV A,@R0
                MOVX @DPTR,A
                INC DPTR
                MOV A,@R1
                MOVX @DPTR,A
                MOV P1,#0FFH
                POP DPL
                POP DPH
```

```
        POP ACC
        POP PSW         ;RETURN OLD RAM BANK
        SETB EX0        ;ENABLE INTERRUPT
        RETI

DP:     MOV R0,#12H
        MOV R1,#13H
        MOV DPH,@R0
        MOV DPL,@R1
        MOVX A,@DPTR
        CJNE A,#77H,NOT_EQ
EQA:    MOV @R0,#00
        MOV @R1,#02
        RET

NOT_EQ: MOV DPH,@R0
        MOV DPL,@R1
        INC DPTR
        MOVX A,@DPTR
        CJNE A,#77H,DP2
        SJMP EQA
DP2:    INC DPTR
        MOVX A,@DPTR
        CJNE A,#77H,DP3
        SJMP EQA
DP3:    MOV @R0,DPH
        MOV @R1,DPL
        RET

DELAY:  MOV R0,#02H
        MOV R1,#50H
        DJNZ,R1,$
        DJNZ,R0,$-4
        RET

DEBUG:  MOV R0,#5H
        MOV R1,#0FFH
        DJNZ,R1,$
        DJNZ,R0,$-4
        RET

DELY:   MOV R0,#01H
        MOV R1,#0AH
        DJNZ,R1,$
        DJNZ,R0,$-4
        RET

        END
```

Features

- Compatible with MCS-51™ Products
- 4K Bytes of Factory Programmable QuickFlash™ Memory
- Fully Static Operation: 0 Hz to 20 MHz
- Three-Level Program Memory Lock
- 128 x 8-Bit Internal RAM
- 32 Programmable I/O Lines
- Two 16-Bit Timer/Counters
- Six Interrupt Sources
- Programmable Serial Channel
- Low Power Idle and Power Down Modes

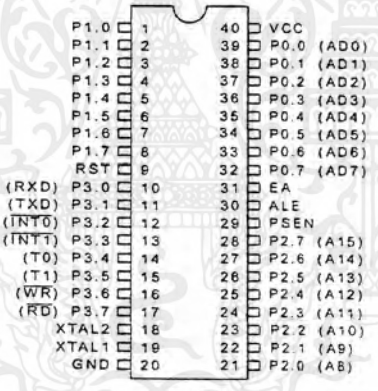
Description

The AT80F51 is a low-power, high-performance CMOS 8-bit microcomputer with 4K bytes of QuickFlash Memory. The device is manufactured using Atmel's high density nonvolatile memory technology and is compatible with the industry standard MCS-51™ instruction set and pinout. The on-chip QuickFlash allows custom codes to be quickly programmed in the factory. By combining a versatile 8-bit CPU with QuickFlash on a monolithic chip, the Atmel AT80F51 is a powerful microcomputer which provides a highly flexible and cost effective solution to many embedded control applications.

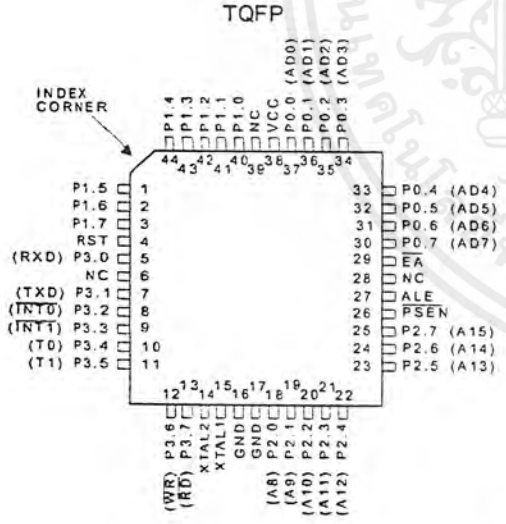
(continued)

Pin Configurations

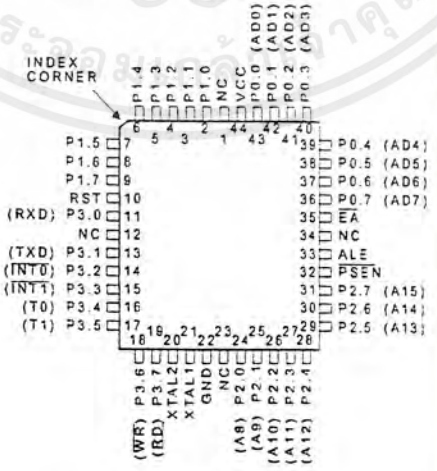
PDIP



TQFP



PLCC

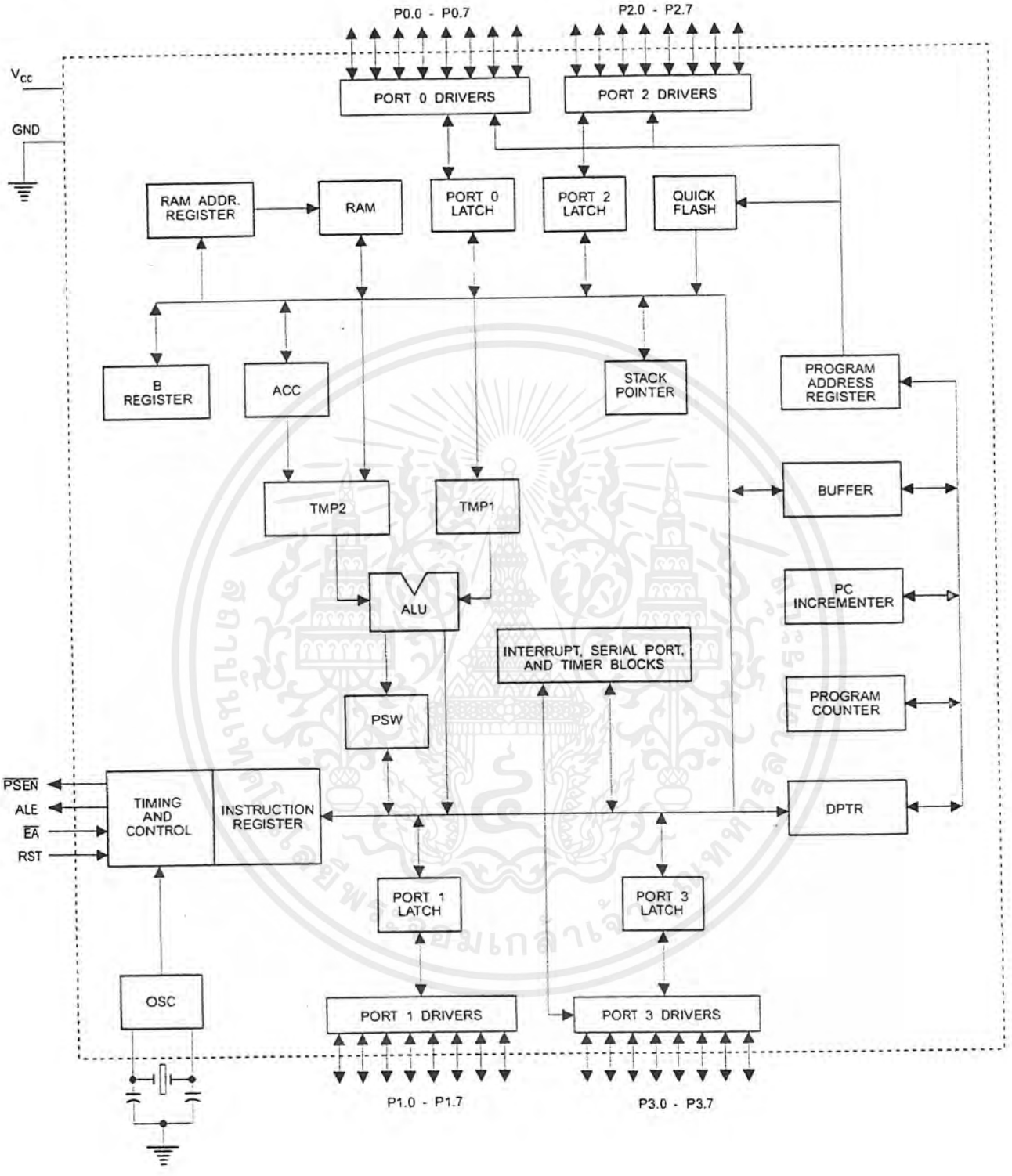


8-Bit Microcontroller with 4K Bytes QuickFlash™ Memory

AT80F51

0979A-A-12/97

lock Diagram



The AT80F51 provides the following standard features: 4K bytes of QuickFlash, 128 bytes of RAM, 32 I/O lines, two 16-bit timer/counters, a five vector two-level interrupt architecture, a full duplex serial port, on-chip oscillator and clock circuitry. In addition, the AT80F51 is designed with static logic for operation down to zero frequency and supports two software selectable power saving modes. The Idle Mode stops the CPU while allowing the RAM, timer/counters, serial port and interrupt system to continue functioning. The Power Down Mode saves the RAM contents but freezes the oscillator disabling all other chip functions until the next hardware reset.

Pin Description

V_{CC}
Supply voltage.

GND
Ground.

Port 0
Port 0 is an 8-bit open drain bidirectional I/O port. As an output port each pin can sink eight TTL inputs. When 1s are written to port 0 pins, the pins can be used as high-impedance inputs.

Port 0 may also be configured to be the multiplexed low-order address/data bus during accesses to external program and data memory. In this mode P0 has internal pullups.

Port 0 also outputs the code bytes during program verification. External pullups are required during program verification.

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

Port 1 also receives the low-order address bytes during QuickFlash verification.

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

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

Port 2 also receives the high-order address bits and some control signals during QuickFlash verification.

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

Port 3 also serves the functions of various special features of the AT80F51 as listed below:

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

Port 3 also receives some control signals for QuickFlash verification.

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

ALE
Address Latch Enable output pulse for latching the low byte of the address during accesses to external memory.

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

If desired, ALE operation can be disabled by setting bit 0 of SFR location 8EH. With the bit set, ALE is active only during a MOVX or MOVC instruction. Otherwise, the pin is weakly pulled high. Setting the ALE-disable bit has no effect if the microcontroller is in external execution mode.

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

When the AT80F51 is executing code from external program memory, $\overline{PSEN}$ is activated twice each machine

cycle, except that two $\overline{PSEN}$ activations are skipped during each access to external data memory.

$\overline{EA}$

External Access Enable. $\overline{EA}$ must be strapped to GND in order to enable the device to fetch code from external program memory locations starting at 0000H up to FFFFH. Note, however, that if lock bit 1 is programmed, $\overline{EA}$ will be internally latched on reset.

$\overline{EA}$ should be strapped to V_{CC} for internal program executions.

XTAL1

Input to the inverting oscillator amplifier and input to the internal clock operating circuit.

XTAL2

Output from the inverting oscillator amplifier.

Oscillator Characteristics

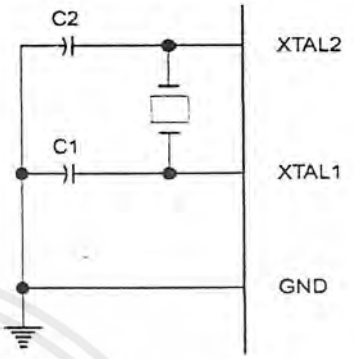
XTAL1 and XTAL2 are the input and output, respectively, of an inverting amplifier which can be configured for use as an on-chip oscillator, as shown in Figure 1. Either a quartz crystal or ceramic resonator may be used. To drive the device from an external clock source, XTAL2 should be left unconnected while XTAL1 is driven as shown in Figure 2. There are no requirements on the duty cycle of the external clock signal, since the input to the internal clocking circuitry is through a divide-by-two flip-flop, but minimum and maximum voltage high and low time specifications must be observed.

Idle Mode

In idle mode, the CPU puts itself to sleep while all the on-chip peripherals remain active. The mode is invoked by software. The content of the on-chip RAM and all the special functions registers remain unchanged during this mode. The idle mode can be terminated by any enabled interrupt or by a hardware reset.

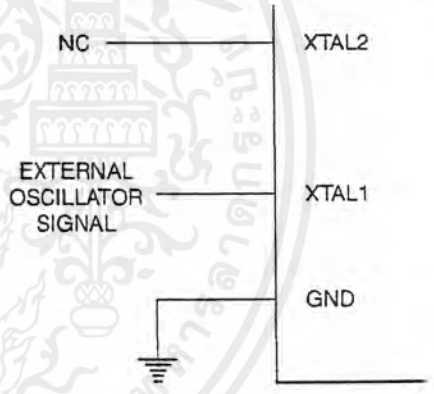
It should be noted that when idle is terminated by a hard ware reset, the device normally resumes program execution, from where it left off, up to two machine cycles before the internal reset algorithm takes control. On-chip hardware inhibits access to internal RAM in this event, but access to the port pins is not inhibited. To eliminate the possibility of an unexpected write to a port pin when Idle is terminated by reset, the instruction following the one that invokes Idle should not be one that writes to a port pin or to external memory.

Figure 1. Oscillator Connections



Note: C1, C2 = 30 pF ± 10 pF for Crystals
= 40 pF ± 10 pF for Ceramic Resonators

Figure 2. External Clock Drive Configuration



Power Down Mode

In the power down mode the oscillator is stopped, and the instruction that invokes power down is the last instruction executed. The on-chip RAM and Special Function Registers retain their values until the power down mode is termi-

Status of External Pins During Idle and Power Down Modes

Mode	Program Memory	ALE	$\overline{PSEN}$	PORT0	PORT1	PORT2	PORT3
Idle	Internal	1	1	Data	Data	Data	Data
Idle	External	1	1	Float	Data	Address	Data
Power Down	Internal	0	0	Data	Data	Data	Data
Power Down	External	0	0	Float	Data	Data	Data

nated. The only exit from power down is a hardware reset. Reset redefines the SFRs but does not change the on-chip RAM. The reset should not be activated before V_{CC} is restored to its normal operating level and must be held active long enough to allow the oscillator to restart and stabilize.

Program Memory Lock Bits

On the chip are three lock bits which can be left unprogrammed (U) or can be programmed (P) to obtain the additional features listed in the table below:

When lock bit 1 is programmed, the logic level at the $\overline{EA}$ pin is sampled and latched during reset. If the device is powered up without a reset, the latch initializes to a random value, and holds that value until reset is activated. It is necessary that the latched value of $\overline{EA}$ be in agreement with the current logic level at that pin in order for the device to function properly.

Lock Bit Protection Modes

Program Lock Bits				Protection Type
	LB1	LB2	LB3	
1	U	U	U	No program lock features.
2	P	U	U	MOVC instructions executed from external program memory are disabled from fetching code bytes from internal memory, $\overline{EA}$ is sampled and latched on reset, and further programming of the QuickFlash is disabled.
3	P	P	U	Same as mode 2, also verify is disabled.
4	P	P	P	Same as mode 3, also external execution is disabled.

Programming/Verifying the QuickFlash

The AT80F51 can only be programmed by Atmel. Customer codes should be submitted in **duplicate** on a floppy disk or uploaded to Atmel's bulletin board or Web site. The code should be in the Intel Hex format. The desired states of the Lock Bits should be specified. Once programmed, the code memory and Lock Bits cannot be erased or reprogrammed.

Please consult the factory or Atmel's representatives for details on submitting custom codes.

Program Verify: If lock bits LB1 and LB2 have not been programmed, the programmed code data can be read back via the address and data lines for verification. The lock bits cannot be verified directly. Verification of the lock bits is achieved by observing that their features are enabled.

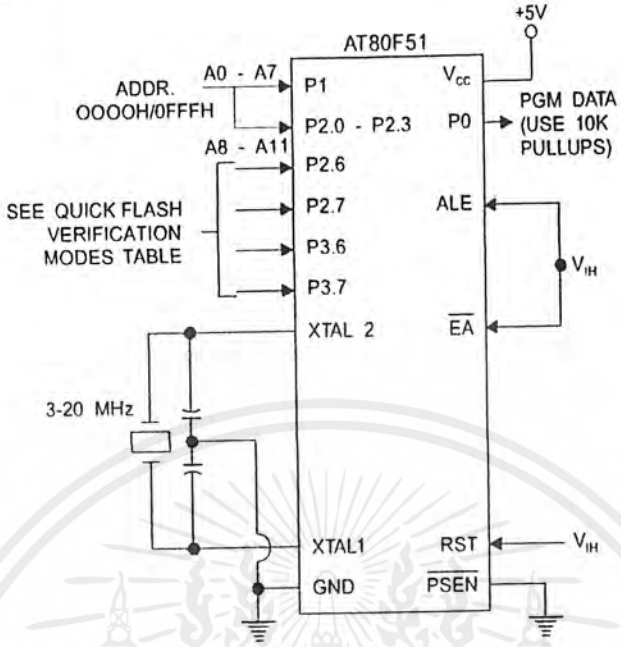
Reading the Signature Bytes: The signature bytes are read by the same procedure as a normal verification of locations 030H, 031H, and 032H, except that P3.6 and P3.7 must be pulled to a logic low. The values returned are as follows.

(030H) = 1EH indicates manufactured by Atmel
(031H) = 80H indicates QuickFlash
(032H) = 01H indicates AT80F51

QuickFlash Verification Modes

Mode	RST	$\overline{PSEN}$	ALE	$\overline{EA}$	P2.6	P2.7	P3.6	P3.7
Read Code Data	H	L	H	H	L	L	H	H
Read Signature Byte	H	L	H	H	L	L	L	L

Figure 3. Verifying the QuickFlash

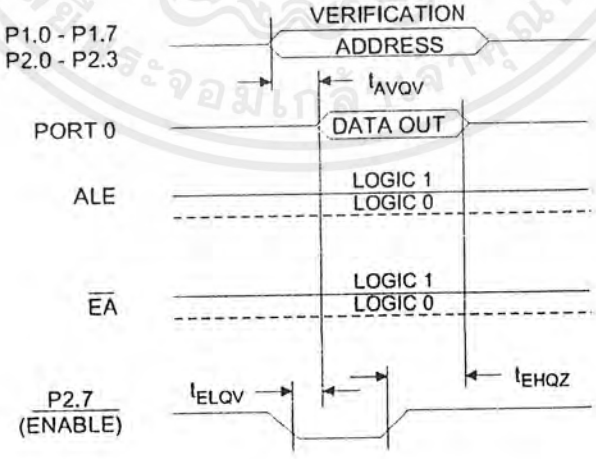


QuickFlash Verification Characteristics

T_A = 0°C to 70°C, V_{CC} = 5.0 ± 10%

Symbol	Parameter	Min	Max	Units
$1/t_{CLCL}$	Oscillator Frequency	3	20	MHz
t_{AVQV}	Address to Data Valid		$48t_{CLCL}$	
t_{ELQV}	$\overline{\text{ENABLE}}$ Low to Data Valid		$48t_{CLCL}$	
t_{EHQZ}	Data Float After $\overline{\text{ENABLE}}$	0	$48t_{CLCL}$	

QuickFlash Verification Waveforms



Absolute Maximum Ratings*

Operating Temperature	-55°C to +125°C
Storage Temperature	-65°C to +150°C
Voltage on Any Pin with Respect to Ground	-1.0V to +7.0V
Maximum Operating Voltage	6.6V
DC Output Current	15.0 mA

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

DC Characteristics

T_A = -40°C to 85°C, V_{CC} = 5.0V ± 20% (unless otherwise noted)

Symbol	Parameter	Condition	Min	Max	Units
V_{IL}	Input Low Voltage	(Except $\overline{EA}$)	-0.5	0.2 V_{CC} - 0.1	V
V_{IL1}	Input Low Voltage ($\overline{EA}$)		-0.5	0.2 V_{CC} - 0.3	V
V_{IH}	Input High Voltage	(Except XTAL1, RST)	0.2 V_{CC} + 0.9	V_{CC} + 0.5	V
V_{IH1}	Input High Voltage	(XTAL1, RST)	0.7 V_{CC}	V_{CC} + 0.5	V
V_{OL}	Output Low Voltage[1] (Ports 1,2,3)	I_{OL} = 1.6 mA		0.45	V
V_{OL1}	Output Low Voltage[1] (Port 0, ALE, $\overline{PSEN}$)	I_{OL} = 3.2 mA		0.45	V
V_{OH}	Output High Voltage (Ports 1,2,3, ALE, $\overline{PSEN}$)	I_{OH} = -60 µA, V_{CC} = 5V ± 10%	2.4		V
		I_{OH} = -25 µA	0.75 V_{CC}		V
		I_{OH} = -10 µA	0.9 V_{CC}		V
V_{OH1}	Output High Voltage (Port 0 in External Bus Mode)	I_{OH} = -800 µA, V_{CC} = 5V ± 10%	2.4		V
		I_{OH} = -300 µA	0.75 V_{CC}		V
		I_{OH} = -80 µA	0.9 V_{CC}		V
I_{IL}	Logical 0 Input Current (Ports 1,2,3)	V_{IN} = 0.45V		-50	µA
I_{TL}	Logical 1 to 0 Transition Current (Ports 1,2,3)	V_{IN} = 2V, VCC = 5V ± 10%		-650	µA
I_{LI}	Input Leakage Current (Port 0, $\overline{EA}$)	0.45 < V_{IN} < V_{CC}		±10	µA
RRST	Reset Pulldown Resistor		50	300	KΩ
C_{IO}	Pin Capacitance	Test Freq. = 1 MHz, T_A = 25°C		10	pF
I_{CC}	Power Supply Current	Active Mode, 12 MHz		20	mA
		Idle Mode, 12 MHz		5	mA
	Power Down Mode[2]	V_{CC} = 6V		100	µA
		V_{CC} = 3V		40	µA

Notes: 1. Under steady state (non-transient) conditions, I_{OL} must be externally limited as follows:
Maximum I_{OL} per port pin: 10 mA
Maximum I_{OL} per 8-bit port: Port 0: 26 mA
Ports 1, 2, 3: 15 mA
Maximum total I_{OL} for all output pins: 71 mA
If I_{OL} exceeds the test condition, V_{OL} may exceed the related specification. Pins are not guaranteed to sink current greater than the listed test conditions.

2. Minimum V_{CC} for Power Down is 2V.

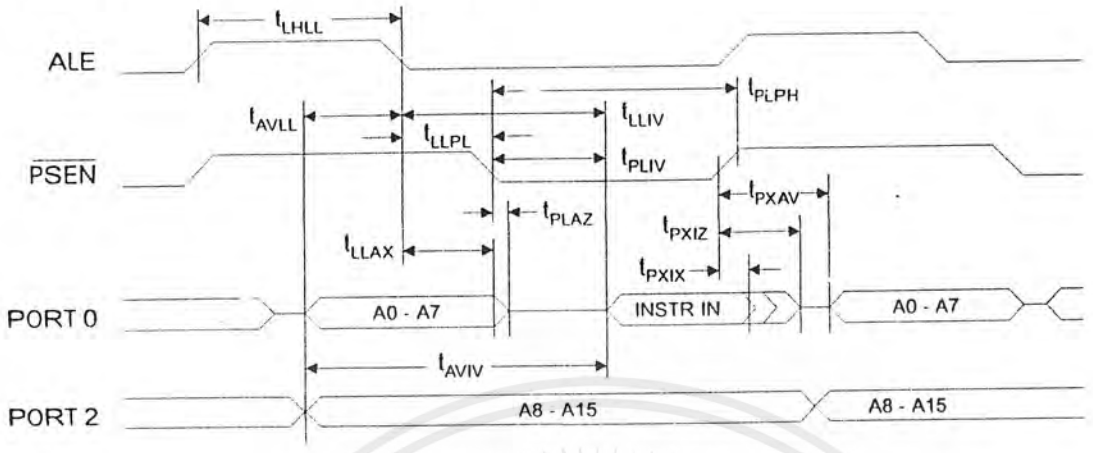
AC Characteristics

(Under Operating Conditions; Load Capacitance for Port 0, ALE, and $\overline{PSEN}$ = 100 pF; Load Capacitance for all other outputs = 80 pF)

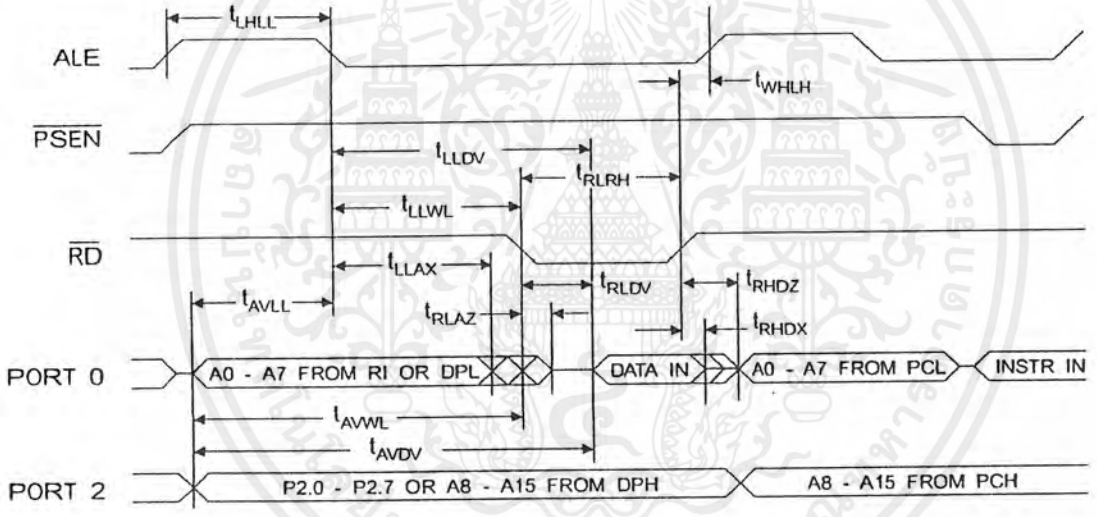
External Program and Data Memory Characteristics

Symbol	Parameter	12 MHz Oscillator		Variable Oscillator		Units
		Min	Max	Min	Max	
$1/t_{CLCL}$	Oscillator Frequency			0	20	MHz
t_{LHLL}	ALE Pulse Width	127		$2t_{CLCL}-40$		ns
t_{AVLL}	Address Valid to ALE Low	43		$t_{CLCL}-13$		ns
t_{LLAX}	Address Hold After ALE Low	48		$t_{CLCL}-20$		ns
t_{LLIV}	ALE Low to Valid Instruction In		233		$4t_{CLCL}-65$	ns
t_{LLPL}	ALE Low to $\overline{PSEN}$ Low	43		$t_{CLCL}-13$		ns
t_{PLPH}	$\overline{PSEN}$ Pulse Width	205		$3t_{CLCL}-20$		ns
t_{PLIV}	$\overline{PSEN}$ Low to Valid Instruction In		145		$3t_{CLCL}-45$	ns
t_{PXIX}	Input Instruction Hold After $\overline{PSEN}$	0		0		ns
t_{PXIZ}	Input Instruction Float After $\overline{PSEN}$		59		$t_{CLCL}-10$	ns
t_{PXAV}	$\overline{PSEN}$ to Address Valid	75		$t_{CLCL}-8$		ns
t_{AVIV}	Address to Valid Instruction In		312		$5t_{CLCL}-55$	ns
t_{PLAZ}	$\overline{PSEN}$ Low to Address Float		10		10	ns
t_{RLRH}	$\overline{RD}$ Pulse Width	400		$6t_{CLCL}-100$		ns
t_{WLWH}	$\overline{WR}$ Pulse Width	400		$6t_{CLCL}-100$		ns
t_{RLDV}	$\overline{RD}$ Low to Valid Data In		252		$5t_{CLCL}-90$	ns
t_{RHDX}	Data Hold After $\overline{RD}$	0		0		ns
t_{RHDZ}	Data Float After $\overline{RD}$		97		$2t_{CLCL}-28$	ns
t_{LLDV}	ALE Low to Valid Data In		517		$8t_{CLCL}-150$	ns
t_{AVDV}	Address to Valid Data In		585		$9t_{CLCL}-165$	ns
t_{LLWL}	ALE Low to $\overline{RD}$ or $\overline{WR}$ Low	200	300	$3t_{CLCL}-50$	$3t_{CLCL}+50$	ns
t_{AVWL}	Address to $\overline{RD}$ or $\overline{WR}$ Low	203		$4t_{CLCL}-75$		ns
t_{QVWX}	Data Valid to $\overline{WR}$ Transition	23		$t_{CLCL}-20$		ns
t_{QVWH}	Data Valid to $\overline{WR}$ High	433		$7t_{CLCL}-120$		ns
t_{WHQX}	Data Hold After $\overline{WR}$	33		$t_{CLCL}-20$		ns
t_{RLAZ}	$\overline{RD}$ Low to Address Float		0		0	ns
t_{WHLH}	$\overline{RD}$ or $\overline{WR}$ High to ALE High	43	123	$t_{CLCL}-20$	$t_{CLCL}+25$	ns

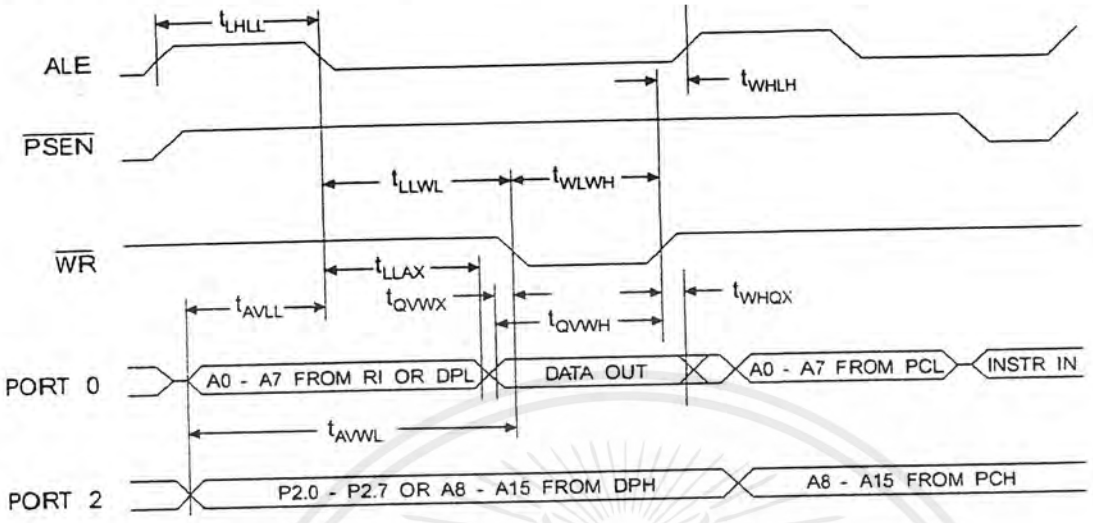
External Program Memory Read Cycle



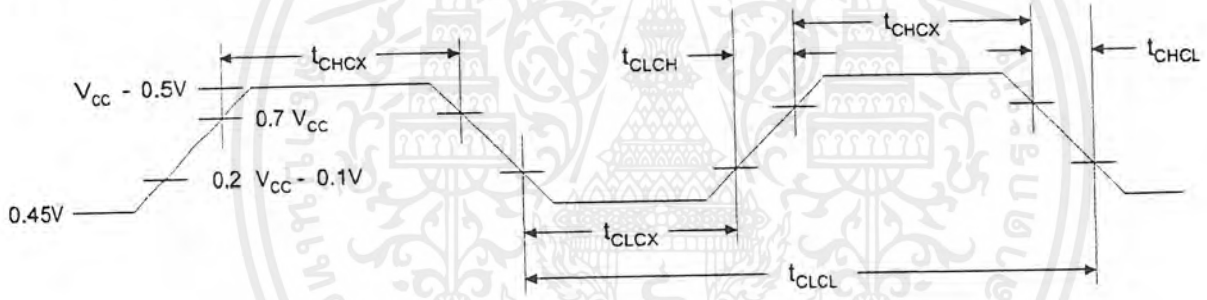
External Data Memory Read Cycle



External Data Memory Write Cycle



External Clock Drive Waveforms



External Clock Drive

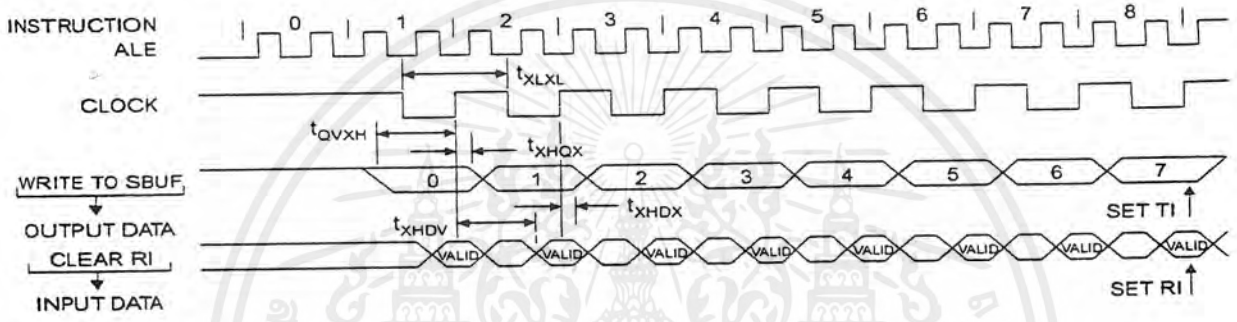
Symbol	Parameter	Min	Max	Units
$1/t_{CLCL}$	Oscillator Frequency	0	20	MHz
t_{CLCL}	Clock Period	41.6		ns
t_{CHCX}	High Time	15		ns
t_{CLCX}	Low Time	15		ns
t_{CLCH}	Rise Time		20	ns
t_{CHCL}	Fall Time		20	ns

Serial Port Timing: Shift Register Mode Test Conditions

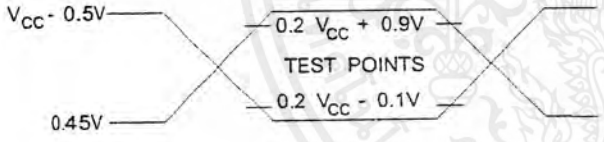
(V_{CC} = 5.0 V ± 20%; Load Capacitance = 80 pF)

Symbol	Parameter	12 MHz Osc		Variable Oscillator		Units
		Min	Max	Min	Max	
t_{XLXL}	Serial Port Clock Cycle Time	1.0		$12t_{CLCL}$		µs
t_{QVXH}	Output Data Setup to Clock Rising Edge	700		$10t_{CLCL}$-133		ns
t_{XHQX}	Output Data Hold After Clock Rising Edge	50		$2t_{CLCL}$-117		ns
t_{XHDX}	Input Data Hold After Clock Rising Edge	0		0		ns
t_{XHDV}	Clock Rising Edge to Input Data Valid		700		$10t_{CLCL}$-133	ns

Shift Register Mode Timing Waveforms

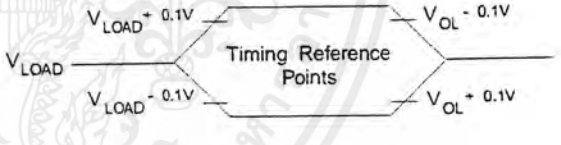


AC Testing Input/Output Waveforms(1)



Note: 1. AC Inputs during testing are driven at V_{CC} - 0.5V for a logic 1 and 0.45V for a logic 0. Timing measurements are made at V_{IH} min. for a logic 1 and V_{IL} max. for a logic 0.

Float Waveforms(1)



Note: 1. For timing purposes, a port pin is no longer floating when a 100 mV change from load voltage occurs. A port pin begins to float when 100 mV change from the loaded V_{OH}/V_{OL} level occurs.

Ordering Information

Speed (MHz)	Power Supply	Ordering Code	Package	Operation Range
12	5V ± 20%	AT80F51-12AC AT80F51-12JC AT80F51-12PC	44A 44J 40P6	Commercial (0°C to 70°C)
		AT80F51-12AI AT80F51-12JI AT80F51-12PI	44A 44J 40P6	Industrial (-40°C to 85°C)
16	5V ± 20%	AT80F51-16AC AT80F51-16JC AT80F51-16PC	44A 44J 40P6	Commercial (0°C to 70°C)
		AT80F51-16AI AT80F51-16JI AT80F51-16PI	44A 44J 40P6	Industrial (-40°C to 85°C)
20	5V ± 20%	AT80F51-20AC AT80F51-20JC AT80F51-20PC	44A 44J 40P6	Commercial (0°C to 70°C)
		AT80F51-20AI AT80F51-20JI AT80F51-20PI	44A 44J 40P6	Industrial (-40°C to 85°C)

Package Type	
44A	44-Lead, Thin Plastic Gull Wing Quad Flatpack (TQFP)
44J	44-Lead, Plastic J-Leaded Chip Carrier (PLCC)
40P6	40-Lead, 0.600" Wide, Plastic Dual Inline Package (PDIP)