ระบบควบคุมแผงแสดงผล

Control Board Display System

นาย ธีระ วิทิตวิทยา

นาย อรรถพล อับดุลเลาะ

ปริญญานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาอุตสาหกรรมศาสตรบัณฑิต

สาขาเทคโนโลยีอิเล็กทรอนิกส์ ภาควิชาเทคนิคอุตสาหกรรม

คณะวิศวกรรมศาสตร์

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

ปีการศึกษา 2544

Control Board Display System

Mr. Teera Witittwittaya

Mr. Adtaphol Abdullor

A THESIS SUBMITTED IN PARTIAL FULFILLMENT

OF THE REQUIREMENT FOR THE DEGREE OF

MACHELOR OF THE TECHNOLOGY ELECTRONICS

FACULTY OF ENGINEERING

KING MONGKUT'S INSTITUTE OF TECHNOLOGY LADKRABANG

2001

หัวข้อปริญญานิพนธ์	ระบบควบคุมแผงแสดงผล Control Board Display System
นักศึกษา	นาย ธีระ วิธิตวิทยา เลขประจำตัว 42015555 นาย อรรถพล อับดุลเลาะ เลขประจำตัว 42015586
อาจารย์ที่ปรึกษา	อาจารย์ภุชงค์ หงษ์สุวรรณ
ภาควิชา	เทคนิคอุตสาหกรรม
ปีการศึกษา	2544

บทคัดย่อ

ปริญญานิพนธ์ฉบับนี้ได้นำเสนอการประยุกต์การนำคอมพิวเตอร์ส่วนบุคคล (Personal Computer) มาใช้ควบคุมการแสดงผลเป็นตัวอักษรภาษาไทยหรือตัวอักษรภาษาอังกฤษ โดยตัวอักษรจะมีรูปแบบการวิ่งจากซ้ายไปขวา ขวาไปซ้าย บนลงล่าง และล่างขึ้นบน และยังสามารถแสดงผลเป็นสีแดงหรือเขียวได้ สำหรับการเปลี่ยนข้อความและรูปแบบนั้นจะทำการเปลี่ยนแปลงที่คอมพิวเตอร์เพียงอย่างเดียว

Thesis Title Control Board Display System

Student Mr.Teera Wititwittaya ID 42015555
Mr. Adtaphol Abdullor ID 42015586

Advisor Mr.Puchong Hongsuwan

Academic Year 2001

ABSTRACT

This thesis presents the application of controlling LED-matrix display by personal computer. The Led-matrix display can be shown to Thai or English letter which they move from left to right, right to left, up to down and down to up. It can selected color to shown, red or green. Exchanced this form or letter on personal computer.

กิตติกรรมประกาศ

ปริญญานิพนธ์ฉบับนี้สำเร็จลุล่วงได้อย่างดี ด้วยคำแนะนำและคำปรึกษาเกี่ยวกับการเขียนปริญญานิพนธ์ จาก อาจารย์ภุชงค์ หงษ์สุวรรณ ซึ่งเป็นอาจารย์ที่ปรึกษา คณะผู้จัดทำรู้สึกซาบซึ้งในความอนุเคราะห์จากท่าน และขอกราบขอบพระคุณเป็นอย่างสูง

ขอขอบพระคุณ เพื่อน ทศพร ภิรมวงศ์ ปรัชญา ลางคุญเสนที่ช่วยค้นคว้าหาข้อมูล ซึ่งมีส่วนทำให้คู่มือฉบับนี้สำเร็จลงได้

คุณค่าและประโยชน์อันพึงมีจากคู่มือฉบับนี้ ผู้จัดทำขอมอบแด่ผู้มีพระคุณทุกท่าน

นาย ธีระ วิทิตวิทยา

นาย อรรถพล อับดุลเลาะ

สารบัญ

สารบัญรูป

หน้า

สารบัญตาราง

บทที่ 1
บทนำ

1.1 บทนำ

โครงงานนี้เป็นการประยุกต์นำเอาคอมพิวเตอร์ส่วนบุคคล (Personal Computer) มาควบคุมการแสดงผลสำหรับแผงแสดงผล ในรูปแบบต่างๆ ตามที่กำหนดให้ เนื่องจากในปัจจุบันซอร์ฟแวร์ที่ใช้ควบคุมแผงแสดงผลเป็นแบบที่ไม่สามารถเปลี่ยนแปลงได้โดยตรง ต้องทำการแก้ไขข้อมูลในซอร์ฟแวร์เพื่อเปลี่ยนแปลงการแสดงผล ซอร์ฟแวร์ที่พัฒนาขึ้นจะช่วยในการแก้ไขข้อมูลและการควบคุมการแสดงผลได้ง่ายขึ้น เพื่อให้บุคคลทั่วไปสามารถนำซอร์ฟแวร์สำเร็จรูปไปใช้งานได้ โดยผู้ใช้สามารถป้อนข้อมูลที่จะแสดงผลและรูปแบบการแสดงผลได้จากโปรแกรมโดยตรงจากเครื่องคอมพิวเตอร์

1.2 วัตถุประสงค์ของโครงงาน

1. เพื่อศึกษาการทำงานการเชื่อมต่อคอมพิวเตอร์ส่วนบุคคลและไมโครคอนโทรลเลอร์ (Microcontroller) ได้
2. เพื่อศึกษาโปรแกรมที่สามารถควบคุมอินพุทและเอาท์พุทและส่งข้อมูลไปยังไมโครคอนโทรลเลอร์ได้
3. เพื่อศึกษาการควบคุมตัวอักษรของแผงด้วยไมโครคอนโทรลเลอร์

เพื่อศึกษาการออกแบบของแผงแสดงผล

1.3 ขอบเขตของโครงงาน

1. เขียนโปรแกรมการแสดงผลเป็นตัวอักษรภาษาไทยและตัวอักษรภาษาอังกฤษ โดยให้มีรูปแบบการแสดงผลดังนี้

- ตัวอักษรวิ่งไปทางซ้าย
- ตัวอักษรวิ่งไปทางขวา
- ตัวอักษรวิ่งขึ้น
- ตัวอักษรวิ่งลง
- เลือกรูปแบบของตัวอักษรได้
- แสดงข้อความได้ 2 ชุดโดยเรียงทีละชุด

2. สร้างชุดวงจรส่วนของการเชื่อมโยงระบบ (Interface)มาตรฐาน RS-232 เพื่อทำหน้าที่เป็นส่วนรับ-ส่งข้อมูลและสัญญาณการควบคุมต่างๆระหว่างส่วนรับข้อมูลและส่วนควบคุม (Control) กับคอมพิวเตอร์ส่วนบุคคล

บทที่ 2

ทฤษฎี MCS-51

2.1 บทนำ

ไมโครคอนโทรลเลอร์แบบชิพเดี่ยวตระกูล MCS-51 นี้ผลิตโดยบริษัทอินเทลมีอยู่ด้วยกันหลายเบอร์ซึ่งมีรายละเอียดดังตาราง

ตารางที่ 2.1 MCS-51 family

Device	ROM less Version	EPROM Version	ROM	RAM	8-Bit I/O Ports	16-Bit Timer/ Counters	PCA	UART	SEP	GSC	DMA CH	A/D CH	Interrupt Sources/ Vectors	Power Downand Idle Modes
8051	8031	-	4K	128	4	2		/					6/5	
8051AH	8031AH	8751H 8751BH	4K	128	4	2		/					6/5	
8052AH	8032AH	8752BH	8K	256	4	3		/					8/6	
80C51BH	80C31BH	87C51	4K	128	4	2		/					6/5	/
80C52	80C32	-	8K	256	4	3		/					8/6	/
83C51FA	80C51FA	87C51FA	8K	256	4	3	/	/					14/7	/
83C51FB	80C51FB	87C51FB	16K	256	4	3	/	/					14/7	/
83C152JA	80C152JA	-	8K	256	5	2		/		/	2		19/11	/
-	80C152JB	-	-	256	7	2		/		/	2		19/11	/
83C152JC	80C152JC	-	8K	256	5	2		/		/	2		19/11	/
-	80C152JD	-	-	256	7	2		/		/	2		19/11	/
83C452	80C452	87C452P	8K	256	5	2		/					9/8	/

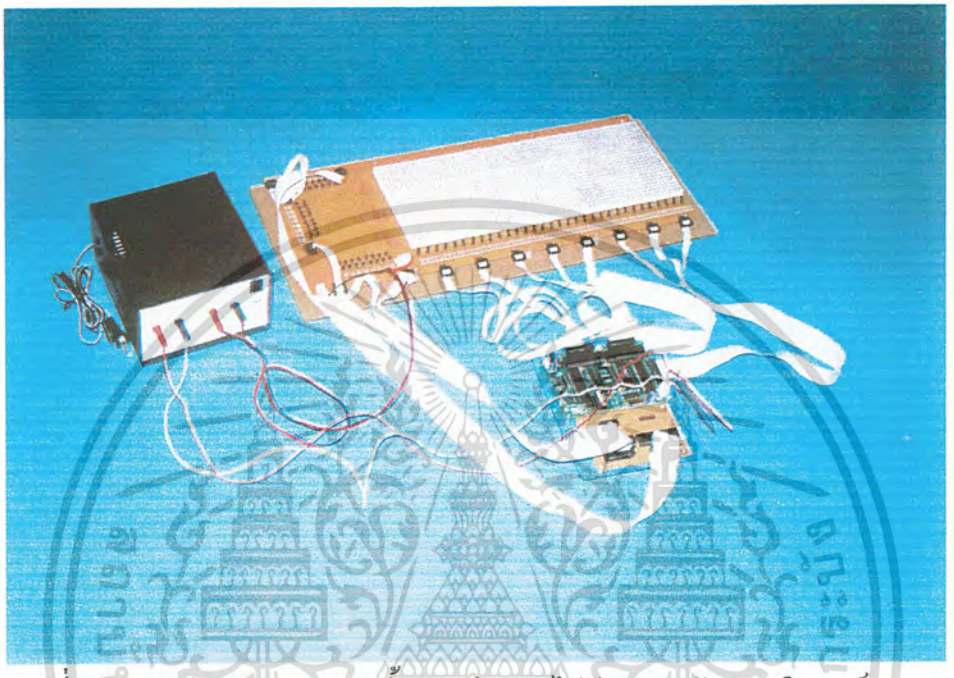
คุณสมบัติของไมโครคอนโทรลเลอร์ตระกูล MCS-51

- ต้องการแหล่งจ่ายไฟ + 5V ชุดเดียว
- มีหน่วยความจำโปรแกรม (Program Memory) ขนาด 4 กิโลไบต์สำหรับเบอร์ 8051 และ 8031 สำหรับเบอร์ 8052
- มีหน่วยความจำสำหรับเก็บข้อมูล (Data Memory) ขนาด 128 ไบต์ สำหรับเบอร์ 8052 ขึ้นไปมีถึง 256 ไบต์
- หน่วยความจำสำหรับเก็บโปรแกรมและข้อมูลแยกจากกันอย่างละ 64 กิโลไบต์
- มีไทม์เมอร์เคาน์เตอร์ ขนาด 16 บิต 2 ชุด (สำหรับ 8052 มี 3 ชุด) ทำงานได้ 4 โหมด
- รับอินเตอร์รัพท์ได้ 6 แหล่ง 5 เวคเตอร์ สำหรับเบอร์ 8052 ขึ้นไปมี 8 แหล่ง 6 เวคเตอร์
- มีพอร์ทรับส่งข้อมูลอนุกรม (UART) 2 พอร์ทแบบ Full Duplex เลือกรูปได้ 4 โหมด
- มีคำสั่งในการทำ AND,OR หรือ Complement ได้ทั้งแบบ 8 บิตและ 1 บิต

บทที่ 3

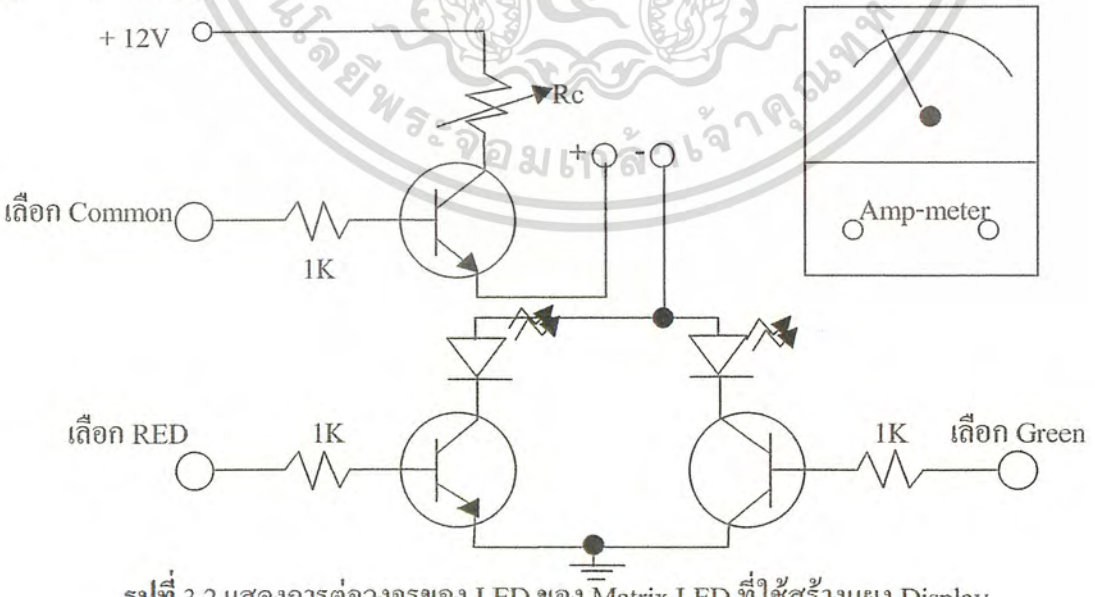
การทดลองและข้อกำหนดการใช้งานของวงจร

3.1 ลักษณะและรูปร่างของชิ้นงาน



รูปที่ 3.1 แสดงส่วนประกอบของชิ้นงานก่อนนำไปประกอบกับคอมพิวเตอร์

3.2 ส่วนของ Supply



รูปที่ 3.2 แสดงการต่อวงจรของ LED ของ Matrix LED ที่ใช้สร้างแผง Display

บทที่ 4

ผลการทดลอง

จากการทดลองโดยที่เราทำการใส่ข้อความที่ต้องการแล้วเลือกการแสดงผลแบบต่างๆให้ผลดังนี้

- ในการเลือกการเลื่อนอักษรนั้น การเลือกจะเป็นไปตามที่เราเลือก โดยจะแสดงการเลื่อนตัวอักษรจากแบบที่ 1 – 4 จนครบ แล้วจะมาทำการเลื่อนแบบที่ 1 ใหม่อีกครั้ง
- ในการเลือกความเร็วในการเลื่อนตัวอักษรนั้น ถ้าเลือก FAST การเลื่อนของตัวอักษรจะเร็ว แต่ถ้าเลือก SLOW การเลื่อนของตัวอักษรจะช้า
- ในการเลือกสีของตัวอักษรนั้น ถ้าเลือก RED จะได้ตัวอักษรสีแดง ถ้าเลือก Green จะได้ตัวอักษรสีเขียว
- ในการเปลี่ยนค่าแต่ละครั้งต้องทำการกดปุ่ม Transfer ทุกครั้ง

บทที่ 5

สรุปผลการทดลอง ปัญหาและข้อเสนอแนะ

โครงงานนี้ ผู้จัดทำได้ออกแบบวงจรและทดลองการทำงานตามรายละเอียดที่กล่าวมาข้างต้นและสามารถทำงานได้ตามวัตถุประสงค์ของโครงงาน จากการทดลองพบว่ายังมีปัญหาเกิดขึ้น อาทิเช่น

- ปัญหาความสว่างของแผงแสดงผลไม่สว่างเท่าตอนที่ทำการทดลอง สาเหตุเนื่องจากเมื่อประกอบส่วนประกอบต่างๆ เข้าด้วยกัน มีการ Load ของอุปกรณ์แต่ละตัวมากเกินไป จึงต้องทำการเปลี่ยนแปลง Power Supply ให้มีแรงดันและกระแสเพิ่มขึ้นอีกระดับหนึ่ง
- ปัญหาการแสดงผลของข้อความไม่ค่อยสม่ำเสมอเท่าที่ควร สาเหตุอาจเกิดตัว Matrix LED แต่ละตัว มี Characteristic ที่ผิดเพี้ยนกันไปบ้าง ทำให้ในการเขียน โปรแกรม Delay ควบคุม จึงเกิดข้อผิดพลาด
- ปัญหาการจำกัดทางด้านงบประมาณ ทำให้ได้แผงแสดงผลที่มีขนาดไม่ใหญ่มาก ทำให้ความสมบูรณ์ของการแสดงผลไม่สมบูรณ์เท่าที่ควร

โครงงานนี้แม้จะทำงานได้ผล แต่ก็ยังไม่สมบูรณ์มากนักเพราะทางผู้จัดทำนั้น ได้เน้นในเรื่องการประหยัด ทำให้ได้แผงแสดงผลที่เล็กตัวหนังสือมีความไม่สมบูรณ์เท่าที่ควร สำหรับผู้ที่สนใจถ้าต้องการให้ตัวหนังสือมีความสมบูรณ์มากกว่านี้ ก็ควรเพิ่มขนาดของแผงแสดงผลให้มีขนาดใหญ่กว่านี้ แล้วแต่ความต้องการของผู้สนใจ

บทที่ 1
บทนำ

1.1 บทนำ

โครงงานนี้เป็นการประยุกต์นำเอาคอมพิวเตอร์ส่วนบุคคล (Personal Computer) มาควบคุมการแสดงผลสำหรับแผงแสดงผล ในรูปแบบต่างๆ ตามที่กำหนดให้ เนื่องจากในปัจจุบันซอร์ฟแวร์ที่ใช้ควบคุมแผงแสดงผลเป็นแบบที่ไม่สามารถเปลี่ยนแปลงได้โดยตรง ต้องทำการแก้ไขข้อมูลในซอร์ฟแวร์เพื่อเปลี่ยนแปลงการแสดงผล ซอร์ฟแวร์ที่พัฒนาขึ้นจะช่วยในการแก้ไขข้อมูลและการควบคุมการแสดงผลได้ง่ายขึ้น เพื่อให้บุคคลทั่วไปสามารถนำซอร์ฟแวร์สำเร็จรูปไปใช้งานได้ โดยผู้ใช้สามารถป้อนข้อมูลที่จะแสดงผลและรูปแบบการแสดงผลได้จากโปรแกรมโดยตรงจากเครื่องคอมพิวเตอร์

1.2 วัตถุประสงค์ของโครงงาน

1. เพื่อศึกษาการทำงานการเชื่อมต่อคอมพิวเตอร์ส่วนบุคคลและไมโครคอนโทรลเลอร์ (Microcontroller) ได้

2. เพื่อศึกษาโปรแกรมที่สามารถควบคุมอินพุทและเอาท์พุทและส่งข้อมูลไปยังไมโครคอนโทรลเลอร์ได้

3. เพื่อศึกษาการควบคุมตัวอักษรของแผงด้วยไมโครคอนโทรลเลอร์
เพื่อศึกษาการออกแบบของแผงแสดงผล

1.3 ขอบเขตของโครงงาน

1. เขียนโปรแกรมการแสดงผลเป็นตัวอักษรภาษาไทยและตัวอักษรภาษาอังกฤษ โดยให้มีรูปแบบการแสดงผลดังนี้

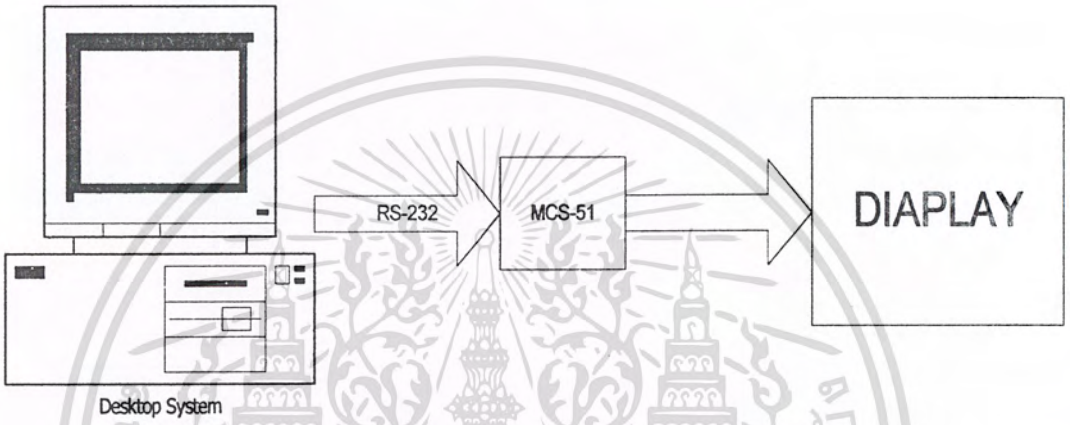
- ตัวอักษรวิ่งไปทางซ้าย - เลือกรูปแบบของตัวอักษรได้
- ตัวอักษรวิ่งไปทางขวา - แสดงข้อความได้ 2 ชุดโดยเรียงทีละชุด
- ตัวอักษรวิ่งขึ้น
- ตัวอักษรวิ่งลง

2. สร้างชุดวงจรส่วนของการเชื่อมโยงระบบ (Interface)มาตรฐาน RS-232 เพื่อทำหน้าที่เป็นส่วนรับ-ส่งข้อมูลและสัญญาณการควบคุมต่างๆระหว่างส่วนรับข้อมูลและส่วนควบคุม (Control) กับคอมพิวเตอร์ส่วนบุคคล

3. เขียนโปรแกรมแอสเซมบลี (Assemby) ให้กับไมโครคอนโทรลเลอร์ เบอร์ 8051 เพื่อทำการรับส่งข้อมูลและควบคุม (Control) แผงแสดงผล

4. สร้างชุดแผงแสดงผลโดยใช้ LED แบบ Matrix ขนาด 8x8 ดวง มี 2 สี คือ สีแดงและสีเขียว โดยมีขนาดของแผงแสดงผล 24x64 ดวง

1.4 ส่วนประกอบของโครงงาน



รูปที่ 1.1 Block Diagram

โครงงานนี้จะประกอบไปด้วย 3 ส่วนสำคัญ คือ

1. ชุดควบคุมและแสดงผลด้วยคอมพิวเตอร์(Control and Display)
2. ชุดวงจรไมโครคอนโทรลเลอร์ (Microcontroller)
3. ชุดแผงแสดงผล LED ขนาด 24 x 64 ดวง

บทที่ 2
ทฤษฎี MCS-51

2.1 บทนำ

ไมโครคอนโทรลเลอร์แบบชิพเดี่ยวตระกูล MCS-51 นี้ผลิตโดยบริษัทอินเทลมีอยู่ด้วยกันหลายเบอร์ซึ่งมีรายละเอียดดังตาราง

ตารางที่ 2.1 MCS-51 family

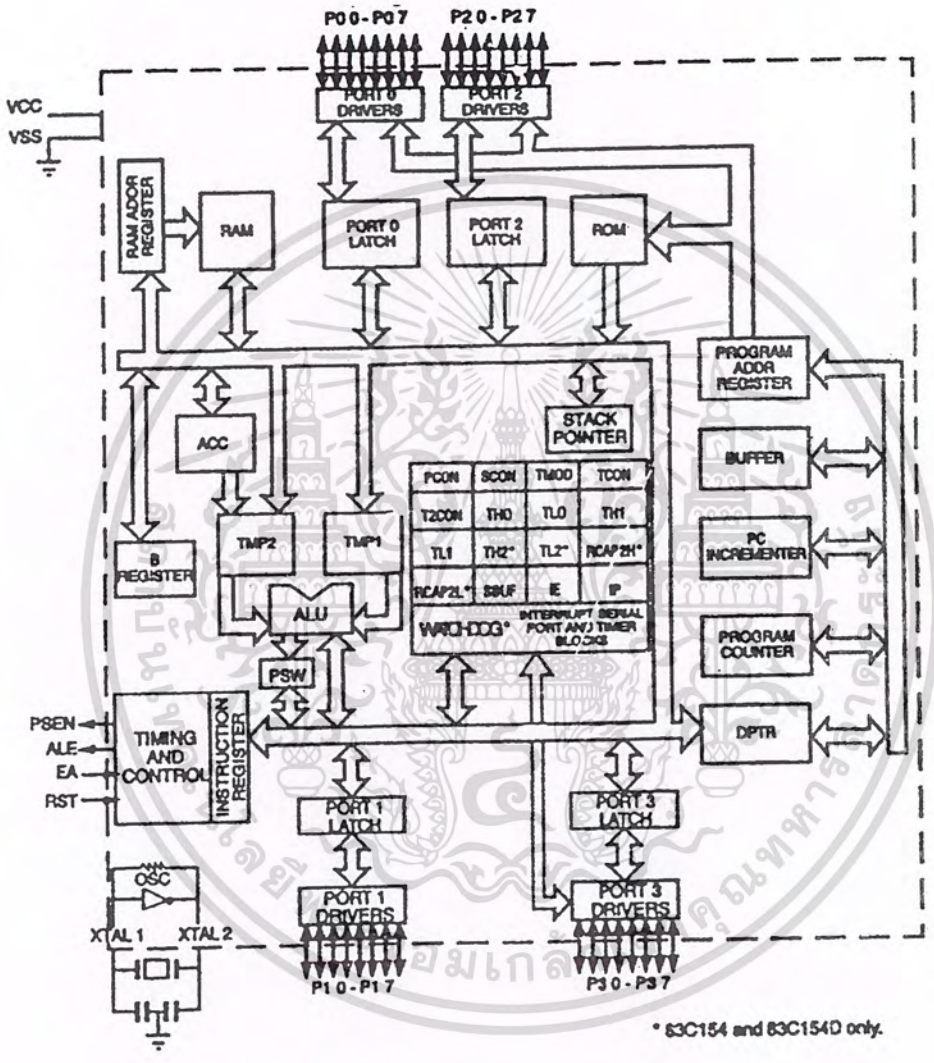
Device	ROM less Version	EPROM Version	ROM	RAM	8-Bit I/O Ports	16-Bit Timer/ Counters	PCA	UART	SEP	GSC	DMA CH	A/D CH	Interrupt Sources/ Vectors	Power Downand Idle Modes
8051	8031	-	4K	128	4	2		/					6/5	
8051AH	8031AH	8751H 8751BH	4K	128	4	2		/					6/5	
8052AH	8032AH	8752BH	8K	256	4	3		/					8/6	
80C51BH	80C31BH	87C51	4K	128	4	2		/					6/5	/
80C52	80C32	-	8K	256	4	3		/					8/6	/
83C51FA	80C51FA	87C51FA	8K	256	4	3	/	/					14/7	/
83C51FB	80C51FB	87C51FB	16K	256	4	3	/	/					14/7	/
83C152JA	80C152JA	-	8K	256	5	2		/		/	2		19/11	/
-	80C152JB	-	-	256	7	2		/		/	2		19/11	/
83C152JC	80C152JC	-	8K	256	5	2		/		/	2		19/11	/
-	80C152JD	-	-	256	7	2		/		/	2		19/11	/
83C452	80C452	87C452P	8K	256	5	2		/					9/8	/

คุณสมบัติของไมโครคอนโทรลเลอร์ตระกูล MCS-51

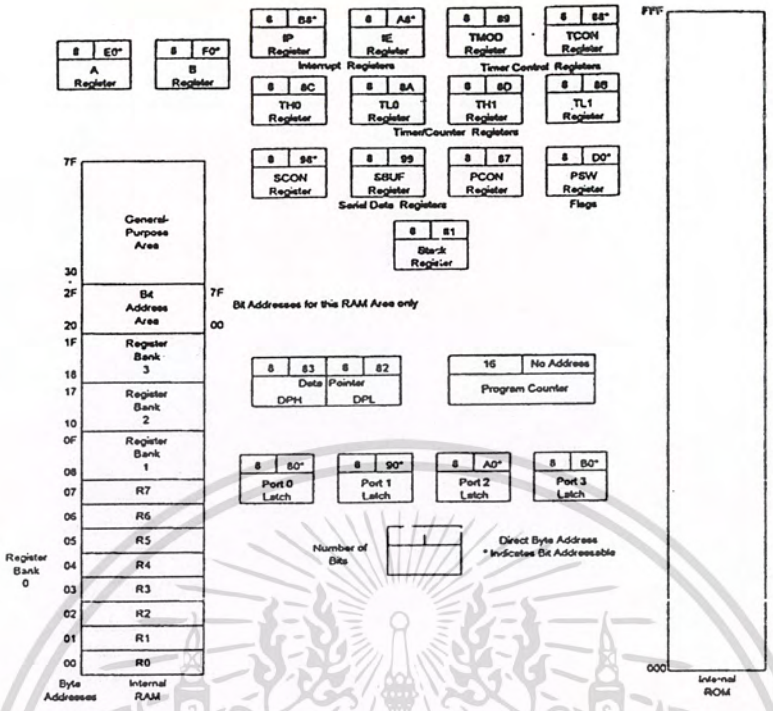
- ต้องการแหล่งจ่ายไฟ + 5V ชุดเดียว
- มีหน่วยความจำโปรแกรม (Program Memory) ขนาด 4 กิโลไบต์สำหรับเบอร์ 8051 และ 8031 สำหรับเบอร์ 8052
- มีหน่วยความจำสำหรับเก็บข้อมูล (Data Memory) ขนาด 128 ไบต์ สำหรับเบอร์ 8052 ขึ้นไปมีถึง 256 ไบต์
- หน่วยความจำสำหรับเก็บโปรแกรมและข้อมูลแยกจากกันอย่างละ 64 กิโลไบต์
- มีไทม์เมอร์เคาน์เตอร์ ขนาด 16 บิต 2 ชุด (สำหรับ 8052 มี 3 ชุด) ทำงานได้ 4 โหมด
- รับอินเตอร์รัพท์ได้ 6 แหล่ง 5 เวคเตอร์ สำหรับเบอร์ 8052 ขึ้นไปมี 8 แหล่ง 6 เวคเตอร์
- มีพอร์ทรับส่งข้อมูลอนุกรม (UART) 2 พอร์ทแบบ Full Duplex เลือกรูปได้ 4 โหมด
- มีคำสั่งในการทำ AND,OR หรือ Complement ได้ทั้งแบบ 8 บิตและ 1 บิต

2.2 โครงสร้างภายในของ 8051

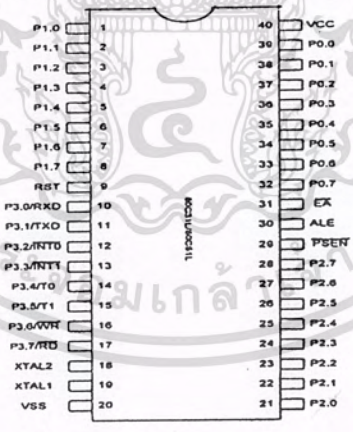
MCS-51 ใช้เทคโนโลยีในการผลิตเป็นแบบ NMOS และ CMOS เบอร์ 8032 และ 8052 จะมี ROM BASIC อยู่ภายในจึงสะดวกสำหรับโปรแกรมเมอร์ที่จะเขียนโปรแกรมด้วยภาษาแบสิก โครงสร้างภายในสำหรับเบอร์ 8051 ดังแสดงในรูป 2.1 (a) และ 2.1 (b)



รูปที่ 2.1(a) 8051 บล็อกไดอะแกรมของ MCS-51



รูปที่ 2.1 (b) ตำแหน่งของรีจีสเตอร์ต่างๆ และหน่วยความจำภายใน

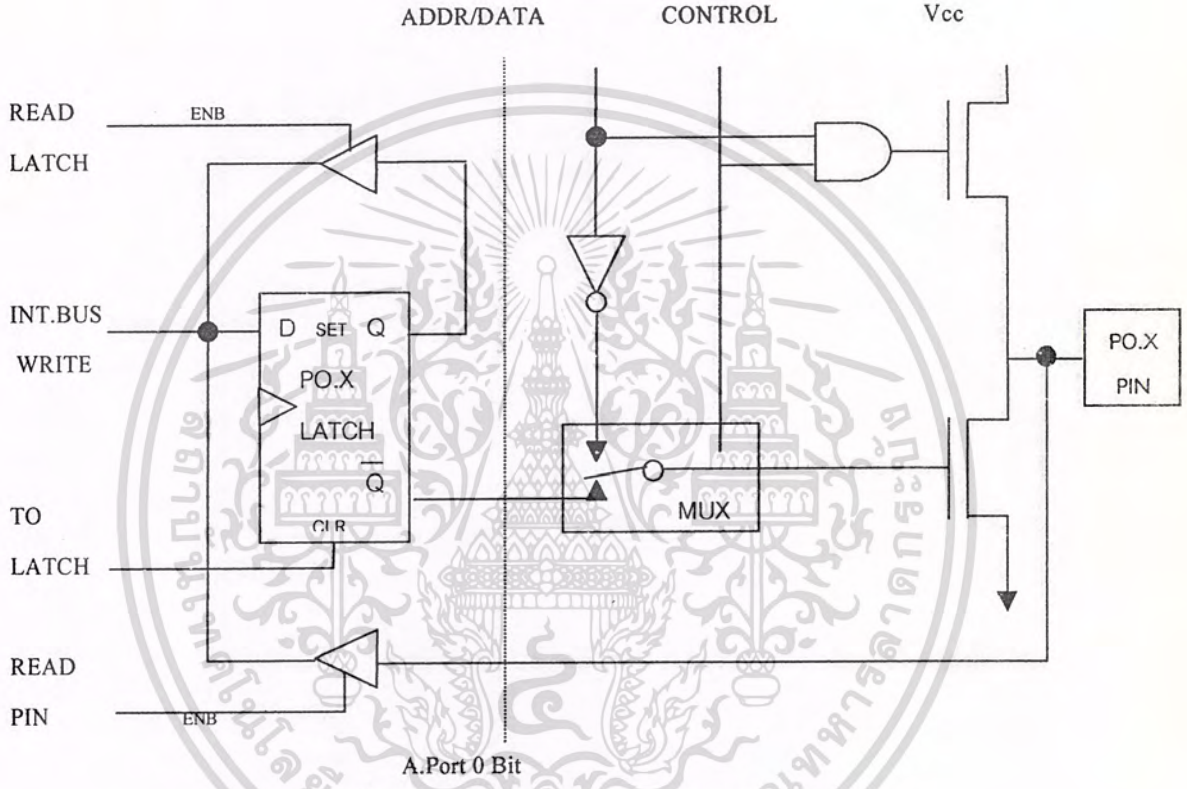


รูปที่ 2.2 การจัดวางขาของ 8051

2.3 พอร์ทของ 8051

8051 เป็นไมโครคอนโทรลเลอร์ขนาด 40 ขา ซึ่งมีขาต่างๆดังนี้

- Vcc (ขา 40) ต่อกับ +5V
- Vss (ขา 20) เป็นขา GND
- พอร์ท 0 (ขา 32-39) มีทั้งหมด 8 บิต คือ (P0.7-P0.0) มีโครงสร้างแบบ Open-Drain Bi-directional ดังแสดงในรูป 2.3



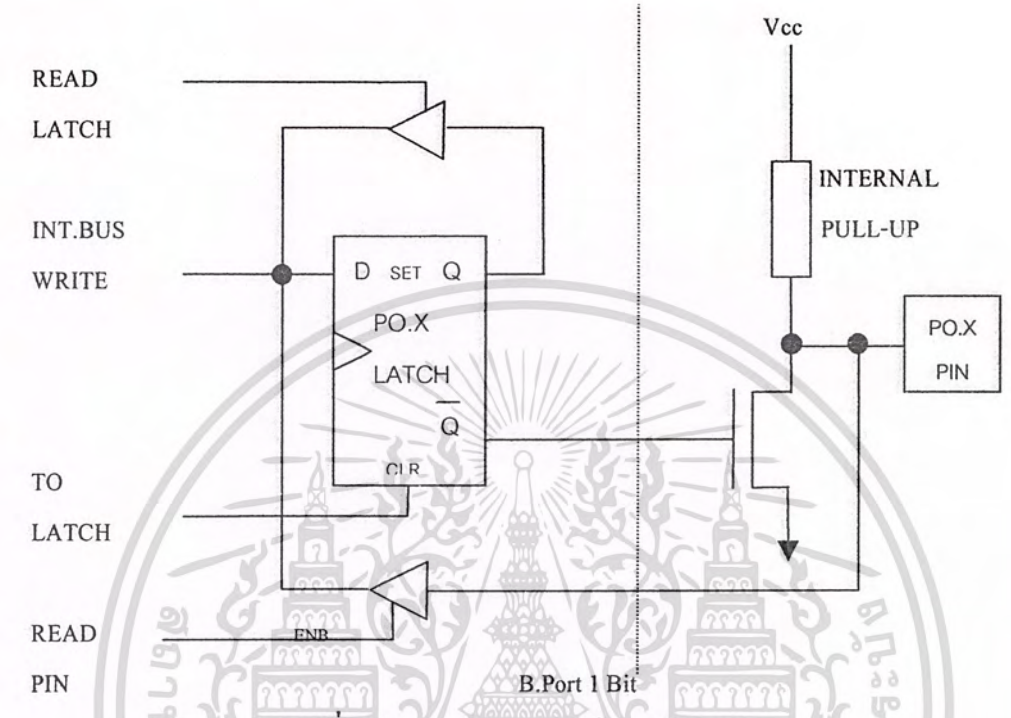
รูปที่ 2.3 แสดงโครงสร้าง พอร์ท 0 (บิต)

- พอร์ท 0 (ขา 32-39) มีทั้งหมด 8 บิต คือ (PO.7-PO.0) ใช้งานได้ 2 หน้าที่ คือแอดเดรสบัสและดาต้าบัสเมื่อต้องการติดต่อกับหน่วยความจำภายนอกหรือเป็นไอโอพอร์ท และถ้าต้องการให้ทำงานเป็นอินพุทพอร์ทต้องส่งลอจิก "1" ไปยังพอร์ทนี้ จะมีผลให้ Q ของ D-FF เป็น "0" ทำให้ FET ตัวล่างมีสถานะ OFF สัญญาณที่ใช้อ่านอินพุทพอร์ทแลทซ์โดยสัญญาณ READ LATCH ไปกระตุ้นที่ Tri-State Buffer ตัวบนและการ อ่าน Port (pin) จะใช้สัญญาณ Read (pin)

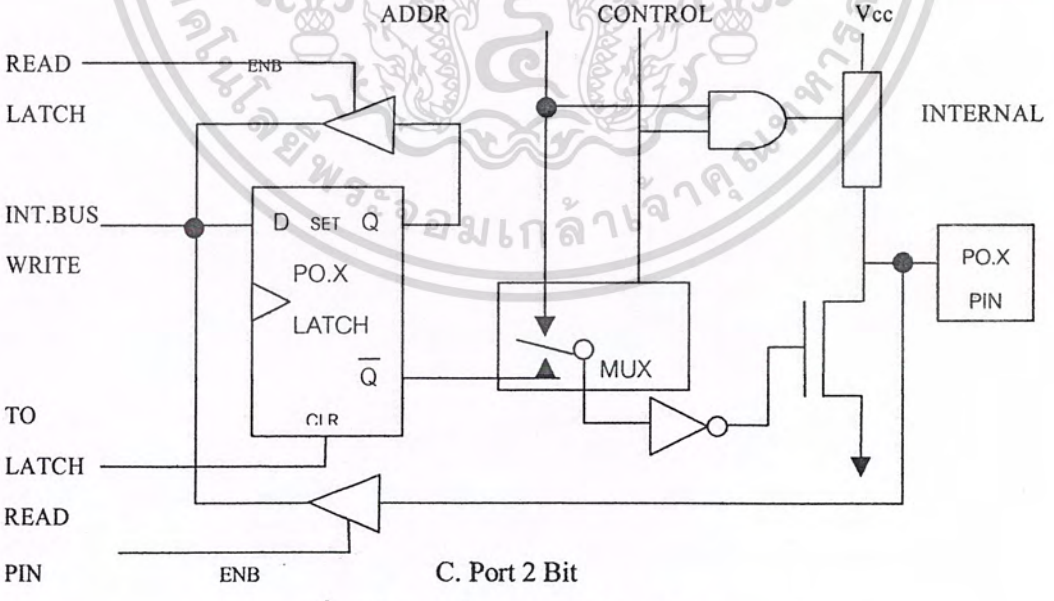
- พอร์ท 1 (ขา 1-8) มีทั้งหมด 8 บิต คือ (P1.0-P1.7) มีโครงสร้างคล้าย พอร์ท 0 แต่จะใช้ความต้านทานภายในพลูอัพแทน Internal Pull up Register มีโครงสร้างดังรูปที่ 2.4

- พอร์ท 2 (ขา 21-28) มีทั้งหมด 8 บิต คือ ขา (P2.7-P2.0) มีโครงสร้างคล้าย พอร์ท 0 โดยมี FET ตัวล่างตัวเดียวส่วนด้านบนใช้ความต้านทานพูลอัพแทน (Internal pull up) พอร์ทนี้ทำงาน 2

หน้าที่ คือสามารถใช้เป็นแอดเดรสบัสขนาด 8 บิต (A15-A8) และเป็นไอโอพอร์ทใช้งานทั่วไปเมื่อจะใช้งานเป็นอินพุทพอร์ทต้องส่งลอจิก "1" มาที่พอร์ทนี้ก่อนเพื่อบังคับให้ FET อยู่ในสภาวะ OFF ดังแสงในรูปที่ 2.5



รูปที่2.4 โครงสร้างของพอร์ท 1 (บิต)



รูปที่ 2.5 โครงสร้างของ พอร์ท 2 (บิต)

- **พอร์ท 3 (ขา 10-17) มีทั้งหมด** 8 บิต คือ ขา (P3.7-3.0) มีโครงสร้างคล้ายพอร์ท 1 ทำงานได้ 2 หน้าที่คือเป็นไอโอพอร์ทถ้าจะโปรแกรมให้เป็นอินพุตพอร์ทต้องส่งลอจิก "1"มาที่พอร์ทนี้ก่อนและอีกหน้าที่หนึ่งก็คือใช้ส่งสัญญาณควบคุมออกมาและรับสัญญาณเข้าไป สัญญาณต่างๆมีดังนี้

P3.0/RXD (Serial Input Port) เป็นขาที่ใช้รับข้อมูลแบบอนุกรม (UART)

P3.1/TXD (Serial Output Port) เป็นขาที่ใช้ส่งข้อมูลแบบอนุกรม (UART)

P3.2/$\overline{\text{INT0}}$ (External Interrupt 0) ใช้รับสัญญาณการขัดจังหวะจากภายนอกเบอร์ 0

P3.3/$\overline{\text{INT1}}$ (External Interrupt 1) ใช้รับสัญญาณการขัดจังหวะจากภายนอกเบอร์ 1

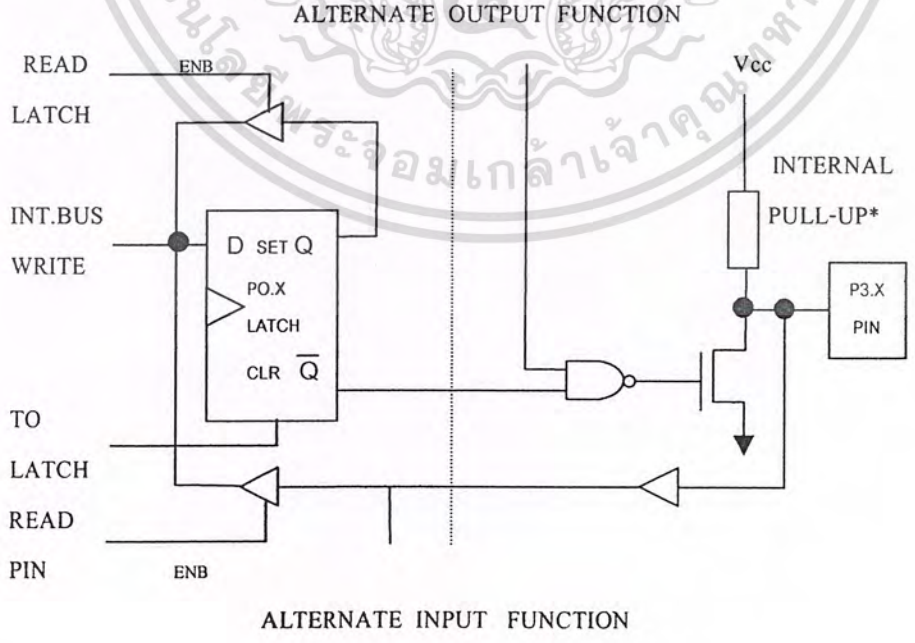
P3.4/T0 (Counter 0 Extrenal Input) ขารับสัญญาณพัลซ์อินพุทเข้าไปยังวงจร Counter 0 (เป็นอินพุทโหมดเคาน์เตอร์)

P3.5/T1 (Counter 1 External Input) ขารับสัญญาณพัลซ์อินพุทเข้าไปยังวงจร Counter 1 (เป็นอินพุทโหมดเคาน์เตอร์)

P3.6/$\overline{\text{WR}}$ (External Data Memory Write Strobe) ขาสัญญาณควบคุมการเขียนข้อมูลข้อมูลลงหน่วยความจำข้อมูลภายนอก

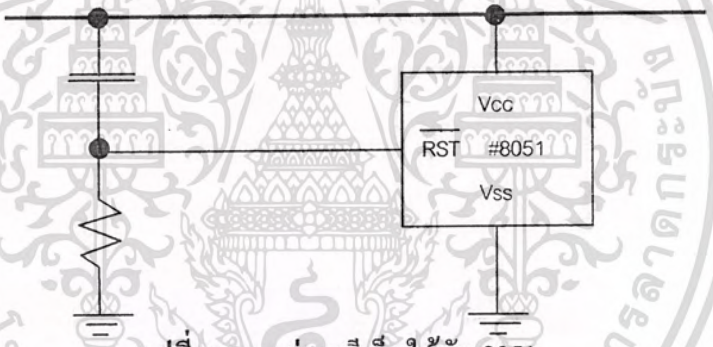
P3.7/$\overline{\text{RD}}$ (External Data Memory Read Strobe) ขาสัญญาณควบคุมการอ่านข้อมูลจากหน่วยความจำข้อมูลภายนอก

โครงสร้างของ (พอร์ท 3) ดังแสดงในรูป 2.6



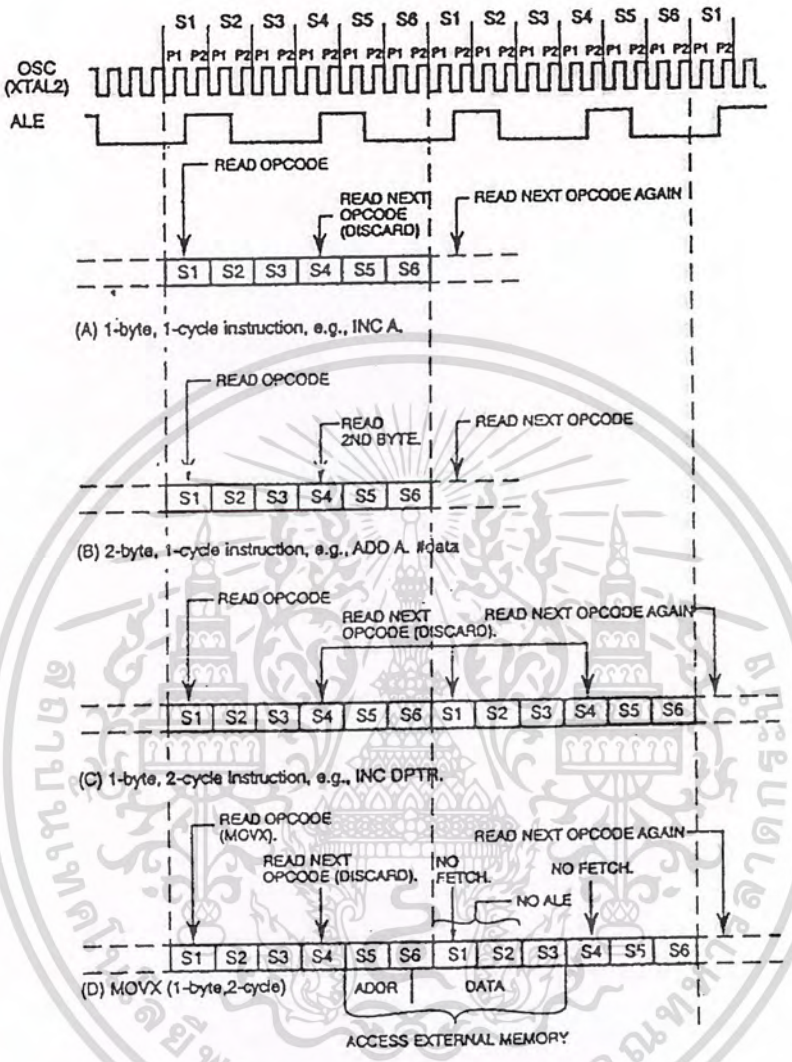
รูปที่ 2.6 โครงสร้างของพอร์ท 3 (บิต)

- **ALE (ขา 30)** เป็นขาส่งสโตรบสำหรับใช้ในการแลตซ์แอดเดรสไบต์ต่ำ (A7-A0) ที่ส่งออกมาจาก(พอร์ท 0)สัญญาณนี้จะแอตทีฟทุกๆ 2 ครั้งใน 1 แมชชีนไซเคิล
- **$\overline{\text{PSEN}}$ (ขา 29)** เป็นขาสโตรปที่ใช้สำหรับอ่านข้อมูลจาก Program Memory ภายนอก สัญญาณนี้จะส่งออกมา 2 ครั้งในแต่ละแมชชีนไซเคิลแต่ถ้าเป็นการอ่าน Internal Program Memory จะไม่มีสัญญาณออกที่ขานี้
- **$\overline{\text{EA}}$ (ขา 30)** ใช้เลือกหน่วยความจำโปรแกรมภายนอก
  - ป้อน "0" จะอ่านโปรแกรมจากภายนอกชิพ
  - ป้อน "1" จะอ่านโปรแกรมจากภายในชิพ
- **RST (ขา 9)** ขารีเซ็ท จะรีเซ็ทได้ก็ต่อเมื่อป้อนลอจิก "1" เข้าที่ขานี้นานอย่างน้อย 2 แมชชีนไซเคิล
- **XTAL1 (ขา 19)** ใช้ต่อคริสตอลภายนอกโดยเป็นอินพุทเข้าสู่วงจรออสซิสเลเตอร์ภายใน
- **XTAL2 (ขา 18)** ใช้ต่อคริสตอลภายนอกโดยเป็นเอาท์พุทของวงจรออสซิสเลเตอร์ภายใน



รูปที่ 2.7 การต่อขารีเซ็ทให้กับ 8051

2.4 ผังเวลาของซีพียู (CPU Timing)



รูปที่ 2.8 ผังเวลาการทำงานของแต่ละคำสั่ง

การทำงานใน 1 คำสั่งต่ำสุดจะกินเวลาเพียง 1 uS เช่น คำสั่ง INC A ซึ่งเป็นคำสั่ง 1 ไบต์ 1 แมชชีนไซเคิล ซึ่งจะใช้คล็อคไปเท่ากับ 12 ลูก โดยคล็อคลูกที่ 1 และ 2 จะอยู่ในช่วง S1 P1 และ S1 P2 และคล็อคลูกที่ 12 ก็จะอยู่ในช่วง S6P2 นั่นเอง (ปรกติแล้ว ซีพียูจะ RUN ด้วยความเร็วเท่ากับ 12 MHz ดังนั้น คล็อค 12 ลูกจะกินเวลาเท่ากับ 1 uS

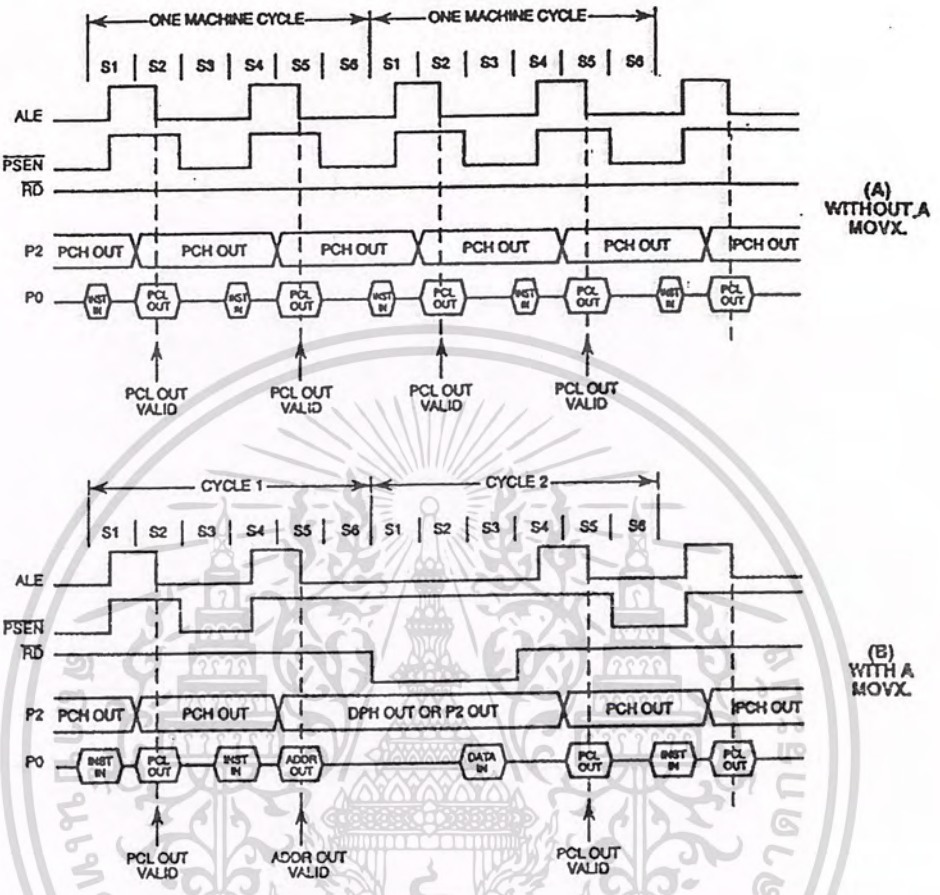
*** คำว่า 1 แมชชีนไซเคิล คือช่วงการทำงานตั้งแต่ S1 จนถึง S6 ***

รูป 2.8(a) การทำงานของคำสั่ง INC A ซึ่งเป็นคำสั่ง 1 ไบต์ทำงานเสร็จภายใน 1 แมชชีนไซเคิล

รูป 2.8(b) การทำงานของคำสั่ง ADD A.#Data ซึ่งเป็นคำสั่ง 2 ไบต์ทำงานเสร็จใน1แมชชีนไซเคิล

รูป 2.8(c) การทำงานของคำสั่ง INC DPTR ซึ่งเป็นคำสั่ง 1 ไบต์ แต่ทำงานเสร็จใน2แมชชีนไซเคิล

รูป 2.8 (d) การทำงานของคำสั่ง MOVX ซึ่งเป็นคำสั่ง 1 ไบต์ แต่ทำงานเสร็จใน 2 แมชชีนไซเคิล



รูปที่ 2.9 แสดงผังเวลาการติดต่อกับหน่วยความจำภายนอก

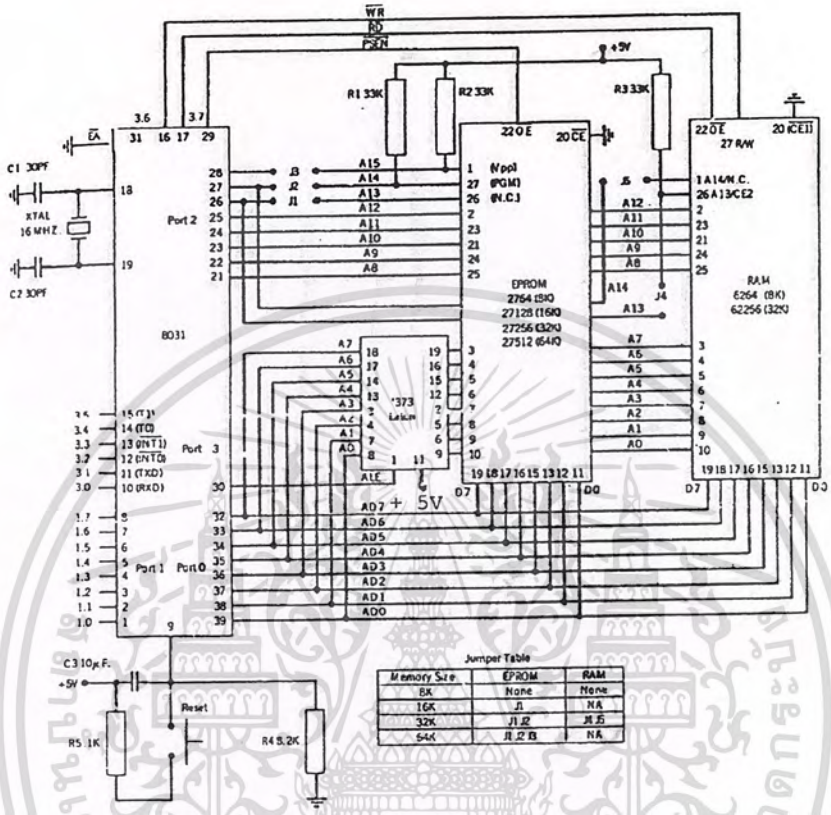
รูป 2.9 (a) เป็นผังเวลาของสัญญาณซึ่งเกี่ยวข้องกับเฟิทซ์เมื่อส่วนของ Program Memory อยู่ภายนอก ดังนั้น สัญญาณที่จะนำไปใช้อ่านออปโค้ด จาก Program Memory ก็คือ $\overline{PSEN}$ ซึ่งจะแอคทีฟ 2 ครั้งใน 1 แมชชีนไซเคิล ดังนั้น สัญญาณที่ใช้อ่านข้อมูลจาก Program Memory จะใช้สัญญาณ $\overline{PSEN}$

รูป 2.9 (b) เป็นผังเวลาของสัญญาณที่ใช้การอ่านข้อมูลจาก Data Memory สัญญาณ $\overline{PSEN}$ จะมีเพียง 1 ลูก เพราะช่วงเวลาถัดมาจะเป็นช่วงเวลาในการอ่านข้อมูลจาก Data Memory โดยใช้สัญญาณ $\overline{RD}$

(การอ่านข้อมูลจาก Program Memory จะใช้สัญญาณ $\overline{PSEN}$ และการอ่านข้อมูลจาก Data Memory จะใช้สัญญาณ $\overline{RD}$ ส่วนสัญญาณ ALE คือ สัญญาณที่ใช้ในการ Latch Address A7-A0 นั่นเอง)

2.5 การต่อหน่วยความจำ Program Memory และ Data Memory

การต่อหน่วยความนำดังแสดงในรูป 2.10

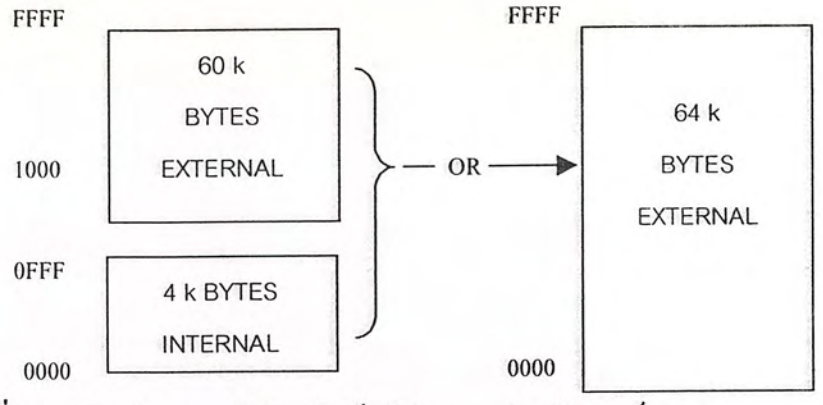


รูปที่ 2.10 การต่อหน่วยความจำโปรแกรม และหน่วยความจำข้อมูลภายนอกชิพ

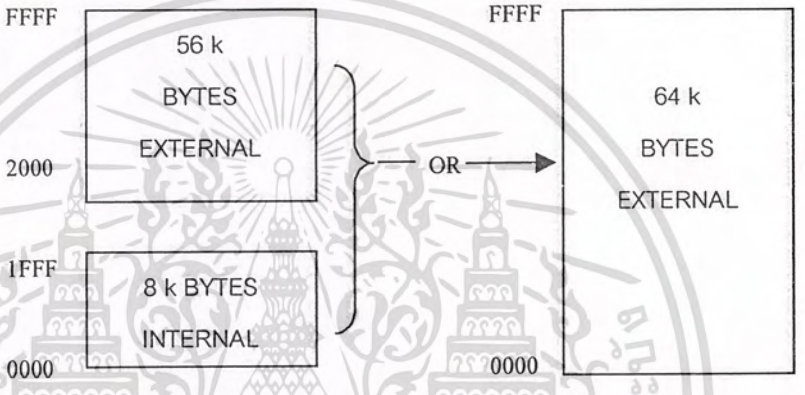
2.6 การแบ่งประเภทของหน่วยความจำ

2.6.1 หน่วยความจำที่ใช้กับ MCS-51 มีอยู่ด้วยกัน 2 ชนิด คือ

2.6.1.1 หน่วยความจำสำหรับเก็บโปรแกรม (Program Memory) เป็นหน่วยความจำที่ใช้เก็บโปรแกรมสั่งงานบรรจุอยู่ในชิพ 8051 ส่วนที่เป็น Program Memory ก็คือ ROM ขนาด 4 กิโลไบต์นั่นเอง แต่ถ้าเป็นเบอร์ 8052 จะมี ROM ขนาด 8 กิโลไบต์ ดังแสดงในรูป 2.11 และ 2.12

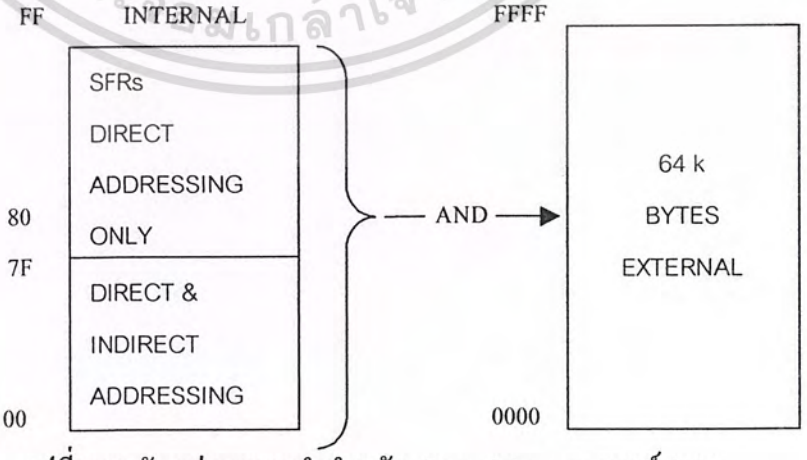


รูปที่ 2.11 ผังหน่วยความจำสำหรับเก็บโปรแกรมสำหรับเบอร์ 8051

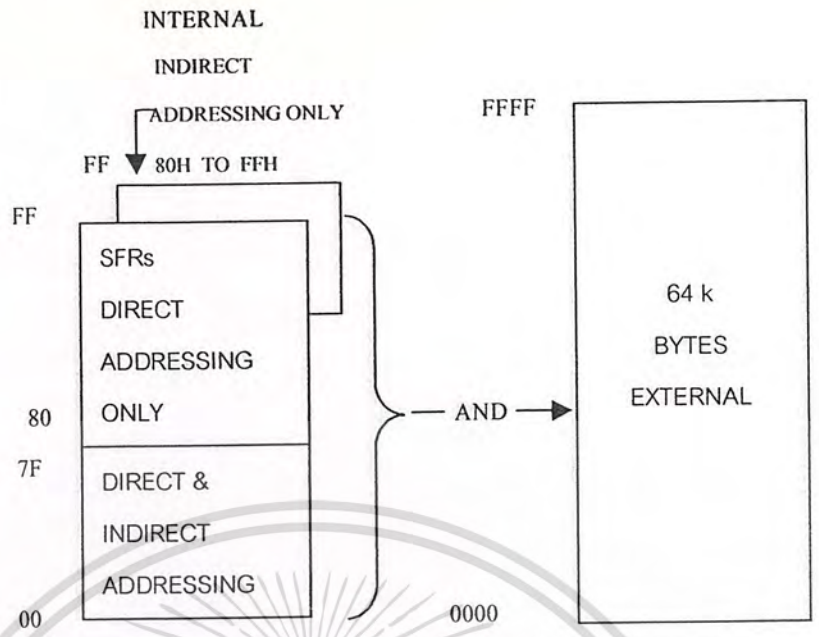


รูปที่ 2.12 ผังหน่วยความจำสำหรับเก็บโปรแกรมสำหรับเบอร์ 8052

2.6.1.2 Data Memory (RAM) แบ่งเป็น 2 ส่วนคือ หน่วยความจำข้อมูลภายในชิพมีเพียง 128 ไบต์ สำหรับเบอร์ 8051 มีจำนวน 256 ไบต์ สำหรับเบอร์ 8052 ขึ้นไปและหน่วยความจำข้อมูลภายนอกชิพมีความจุ 64 กิโลไบต์ ดังแสดงในรูปที่ 2.14 และ 2.15



รูปที่ 2.13 ผังหน่วยความจำสำหรับ Data Memory เบอร์ 8051



รูปที่ 2.14 ผังหน่วยความจำสำหรับ Program Memory ของ 8052

บางครั้งอาจจะสงสัยว่าตำแหน่งของหน่วยความจำสำหรับโปรแกรมและดาต้ามีตำแหน่งที่ซ้อนกันซีพียูจะรู้ได้อย่างไรว่าติดต่อกับหน่วยความจำโปรแกรมและหน่วยความจำข้อมูล บริษัทอินเทลได้ออกแบบแยกคำสั่งออกเป็น 3 ส่วน คือ

MOV ใช้ติดต่อกับ RAM ภายใน

MOVC ใช้ติดต่อ Program Memory

MOVX ใช้ติดต่อ Data Memory ภายนอกชิพ โดยระบุตำแหน่งผ่าน DPTR และ PC

* ชิพเบอร์ 8052 จะมีพื้นที่บริเวณ 80h-FFh ซึ่งถ้าจะเขียนอ่านข้อมูล ณ บริเวณนี้จะเข้าถึงข้อมูลโดยอ้อมเท่านั้น ดังผังหน่วยความจำดังรูป 2.15 *

2.6.2 พื้นที่หน่วยความจำที่เข้าถึงข้อมูลโดยทางอ้อมเท่านั้น (Indirect Address Area)

พื้นที่หน่วยความจำบริเวณ (80h - FFh) ตามรูป 2.15 เป็นพื้นที่ซ้อนกันอยู่อย่างละ 128 ไบต์ โดยส่วนแรกจะเป็น SFR แอดเดรสและ Indirect Address Area ดังนั้นผู้เขียนโปรแกรมถ้าจะติดต่อกับ SFR จะต้องใช้คำสั่งแบบเข้าถึงข้อมูลโดยตรงเท่านั้น (Direct Address Area) ส่วนพื้นที่อีกส่วนหนึ่งจะเข้าถึงข้อมูลแบบทางอ้อมเท่านั้น (Indirect Address Area) ส่วนตำแหน่ง (00h – 7Fh) จะเข้าถึงข้อมูลได้ทั้ง 2 แบบ

2.6.3 พื้นที่หน่วยความจำที่เข้าถึงข้อมูลโดยตรงและทางอ้อม (Direct and Indirect Address Area)

2.6.3.1 รีจิสเตอร์ แบงค์ (Register Banks 0-3)

ตั้งแต่ตำแหน่ง (00h-1Fh) จะเป็นส่วนของรีจิสเตอร์แบงค์ (0-3) โดยแบ่งเป็นแบงค์ละ 8 ไบต์รวมแล้วได้ 32 ไบต์ (แต่ละแบงค์จะมีรีจิสเตอร์ R0, R1, R2, R3, R4, R5, R6, R7) ถ้าซีพียูทำงานอยู่ที่แบงค์ 3 เมื่อถูกรีเซ็ทก็จะกลับมาทำงานที่แบงค์ 0 เสมอ และ SP จะมาเริ่มต้นที่ตำแหน่ง 07h ทันที

2.6.3.2 บริเวณหน่วยความจำที่ใช้คำสั่งอ่านเขียนทีละบิตได้ (Bit Addressable Area)

พื้นที่ตั้งแต่แอดเดรส (20h-7Fh) จำนวน 16 ไบต์หรือแบ่งเป็นบิตจะได้เท่ากับ 128 บิต ซึ่งตำแหน่งบิตมีดังนี้ 00, 01, 02, 03, 04, 05, 06, 07 จนถึง7FH

เช่น บิต 00 ก็คือ D0 ของหน่วยความจำตำแหน่งที่ 20h

บิต 01 ก็คือ DI ของหน่วยความจำตำแหน่งที่ 20h

ดูรูปที่ 2.15 ประกอบ เช่นต้องการเซ็ทบิต 00 ต้องเขียนคำสั่งว่า SET 00h

2.6.5 ไทม์เมอร์/เคาน์เตอร์

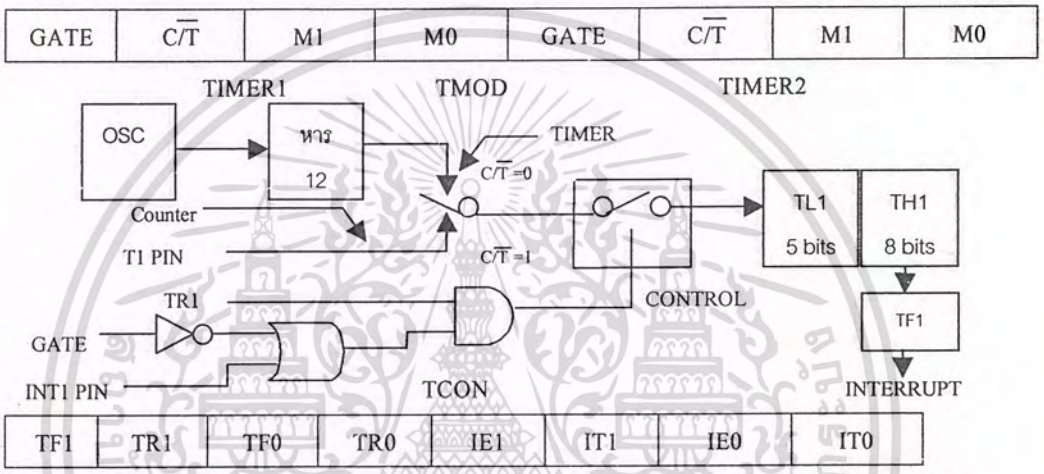
2.6.5.1 ไทม์เมอร์/เคาน์เตอร์ สามารถเลือกให้มีการทำงานเป็นไทม์เมอร์ หรือเคาน์เตอร์อย่างใดอย่างหนึ่ง โดยเลือกที่บิต $\overline{C/T}$ ในรีจิสเตอร์ใช้งานเฉพาะ TMOD โหมดไทม์เมอร์และเคาน์เตอร์จะใช้ up Counter Register (TH_x, TL_x) ตัวเดียวกันซึ่งเป็นแบบนับขึ้น

2.6.5.2 โหมดไทม์เมอร์ up Counter Register (TH_x, TL_x) จะถูกเพิ่มค่าทุกๆ 1 แมชชีนไซเคิล (12 คาบเวลาของ CPU osc) โหมดนี้ไม่ต้องป้อนสัญญาณจากภายนอกเข้ามาแต่จะใช้สัญญาณ CPU osc

2.6.5.3 โหมดเคาน์เตอร์ up Counter Register (TH_x, TL_x) จะถูกเพิ่มค่าทีละหนึ่งเมื่อสัญญาณนาฬิกาจากภายนอกเข้ามา 1 ลูกเข้ามาทางขา T0 (pin) หรือ T1 (pin) อยู่ที่ขา 14 และ 15 ตามลำดับโดยไม่สนใจคาบเวลาของพัลซ์แต่ละลูกการตรวจสอบสัญญาณที่เข้ามาทางขานี้โดยจะตรวจสอบทุกๆ S5P2 ของแต่ละแมชชีนไซเคิล ดังนั้นการตรวจสอบสัญญาณนาฬิกา 1 ลูกจะต้องใช้ถึง 2 แมชชีนไซเคิล (1/24 คาบเวลาของ CPU osc)

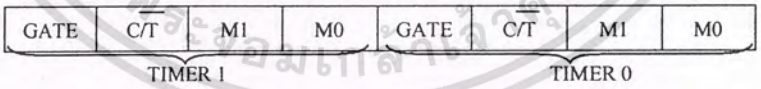
2.6.5.4 โครงสร้างของไทม์เมอร์และเคาน์เตอร์ มีโครงสร้างดังนี้

- เคาน์เตอร์ แบบนับขึ้น up Counter Register (TH_x, TL_x)
- ส่วนเลือกโหมดไทม์เมอร์ และ เคาน์เตอร์ เลือกที่บิต $\overline{C}/T$
- ส่วนควบคุมการนับ และหยุดนับ (Start Counter) ควบคุมที่บิต TR_x, GATE และสัญญาณจากภายนอกที่ขา $\overline{INTx}$ (pin)
- ในโหมดเคาน์เตอร์จะรับอินพุตพับซ์จากภายนอกที่ขา Tx (pin)
- ในโหมดไทม์เมอร์จะรับอินพุตพัลซ์จากคล็อคซีพียูที่หารด้วย 12 แล้ว
- ในโหมดเคาน์เตอร์และไทม์เมอร์ใช้เคาน์เตอร์ตัวเดียวกันเป็นแบบนับขึ้น



รูปที่ 2.15 ผังการทำงานของไทม์เมอร์/เคาน์เตอร์1(โหมด 0) 13-bit Counter และรีจิสเตอร์ควบคุม

2.6.5.5 Timer/Counter Mode Control Register (TMOD) อยู่ใน SFR ตำแหน่งที่ (89H)



GATEx - เป็นบิตเลือกการสตาร์ทไทม์เมอร์เคาน์เตอร์ x

0 ควบคุมโดย Software (Internal Control)

1 ควบคุมโดย Hardware (External Control โดยการทริกจากภายนอก)

$\overline{C}/T$ - บิตเลือกการทำงานเป็นไทม์เมอร์หรือเคาน์เตอร์โดยเลือกดังนี้

ถ้า $\overline{C}/T = 0$ เป็นการเลือกโหมดไทม์เมอร์ (นับจำนวนแมชชีนไซเคิล)

ถ้า $\overline{C}/T = 1$ เป็นการเลือกโหมดเคาน์เตอร์ (นับจำนวนพัลซ์จากภายนอก)

M1,M2 - เลือกโหมดการทำงานได้ 4 โหมด

M1	M2	โหมด	การทำงาน
0	0	0	13 บิต ไทม์เมอร์หรือเคาน์เตอร์
0	1	1	16 บิต ไทม์เมอร์หรือเคาน์เตอร์
1	0	2	8 บิต ไทม์เมอร์หรือเคาน์เตอร์แบบโหลดซ้ำอัตโนมัติ
1	1	3	8 บิต ไทม์เมอร์หรือเคาน์เตอร์ โดยใช้ TL0
1	1	3	8 บิต ไทม์เมอร์โดยใช้ TH0

2.6.5.6 Timer/Counter Control Register (TCON) อยู่ใน SFR ตำแหน่งที่ (088H)

TF1	TR1	TF0	TR0	IE1	IT1	IE0	IT0

บิต TF1 - แฟลกซ์แสดงการเกิดโอเวอร์โฟลว์ของไทม์เมอร์ 1 จะเซ็ทเมื่อไทม์เมอร์ 1 โอเวอร์โฟลว์และจะถูกเคลียร์เองเมื่อซีพียูย้ายไปที่โปรแกรมบริการอินเตอร์รัปต์หรือใช้คำสั่ง CLR TF1

บิต TR1 - บิตควบคุมการนับของไทม์เมอร์ เคาเตอร์ 1 ควบคุมจากโปรแกรม

1 ไทม์เมอร์ หรือ เคาน์เตอร์ 1 **เริ่มทำงาน**

0 ไทม์เมอร์ หรือ เคาน์เตอร์ 1 **หยุดทำงาน**

(กรณีนี้ต้องเซ็ทหรือรีเซ็ทบิต GATE1 ใน TMOD ก่อน)

บิต TF0 - แฟลกซ์แสดงการเกิด โอเวอร์โฟลว์ ของไทม์เมอร์ 0 ถูกเซ็ทเมื่อไทม์เมอร์ 0 เกิดโอเวอร์โฟลว์ เช่นเดียวกับ TF1

บิต TR0 - เช่นเดียวกับ TR1 แต่ใช้ควบคุมไทม์เมอร์ 0

บิต IE1 - แฟลกซ์แสดงการเกิดสัญญาณอินเตอร์รัปต์ภายนอกหมายเลข 1 เมื่อมีสัญญาณอินเตอร์รัปต์เข้ามาที่ขา $\overline{\text{INT 0}}$ และถูกเคลียร์เองโดยคำสั่ง RETI ในโปรแกรมส่วนบริการอินเตอร์รัปต์

บิต IT1 - แฟลกซ์เลือกประเภทการตรวจสอบสัญญาณอินเตอร์รัปต์ที่เกิดขึ้นที่ขา $\overline{\text{INT 1}}$ โดย

1 จะตรวจสอบการเปลี่ยนระดับแบบขอบขาลงที่ขา $\overline{\text{INT 1}}$

0 จะตรวจสอบระดับศูนย์ของสัญญาณที่ขา $\overline{\text{INT 1}}$

บิต IE0 - แฟลกซ์แสดงการเกิดสัญญาณอินเตอร์รัปต์ภายนอกหมายเลข 0 เมื่อมีสัญญาณอินเตอร์รัปต์เข้ามาที่ขา $\overline{\text{INT 1}}$ และถูกเคลียร์เอง โดยคำสั่ง RETI ที่อยู่ในโปรแกรม ส่วนบริการอินเตอร์รัปต์

บิต IT0 - เช่นเดียวกับ IT1

ตารางที่ 2.2 โหมดการทำงานของ ไทม์เมอร์ 2

RCLK+TCLK	$\overline{CP/RL2}$	TR2	MODE
0	0	1	16-bit auto-reload
0	1	1	16-bit capture
1	X	1	Baud rate generator
X	X	0	(off)

บิต TF2 - แฟลกซ์แสดงการเกิดโอเวอร์โฟลว์ของ ไทม์เมอร์ 2 จะเซ็ทเมื่อไทม์เมอร์ 2 เกิดโอเวอร์โฟลว์และจะถูกเคลียร์โดยใช้ซอร์ฟแวร์เท่านั้นโดยใช้คำสั่ง CLR TF2 เท่านั้น บิต TF2 จะไม่ถูกเซ็ทถ้า บิต RCLK=1 และ TCLK=1

บิต EXF2 - แฟลกซ์แสดงการเกิดอินเตอร์รัพต์จากภายนอกเบอร์ 2 เมื่อมีสัญญาณทริกจากภายนอกแบบขอบขาลงที่ขา T2EX pin และ บิต EXEN2 ต้อง =1 การเคลียร์จะใช้ซอร์ฟแวร์เท่านั้นเช่น CLR EXF2

บิต RCLK - ใช้เลือกแหล่งกำเนิดสัญญาณคล้อคสำหรับกำหนดบอดเรทสำหรับพอร์ทสื่อสารทางด้านรับ

1 = เลือกจาก ไทม์เมอร์ 2

0 = เลือกจาก ไทม์เมอร์ 1

บิต TCLK - ใช้เลือกแหล่งกำเนิดสัญญาณคล้อคสำหรับกำหนดบอดเรทสำหรับพอร์ทสื่อสารทางด้านส่ง

1 = เลือกจาก ไทม์เมอร์ 2

0 = เลือกจาก ไทม์เมอร์ 1

บิต EXEN2 - บิตเลือก อนุญาตหรือไม่อนุญาตสัญญาณอินเตอร์รัปต์ภายนอกที่ขา T2EX pin

1 = อนุญาต

0 = ไม่อนุญาต

บิต TR2 - บิตควบคุมการสตาร์ทของไทม์เมอร์ 2 ควบคุมจากโปรแกรม

1 ไทม์เมอร์ หรือ เคาน์เตอร์ 2 **เริ่มทำงาน**

0 ไทม์เมอร์ หรือ เคาน์เตอร์ 2 **หยุดทำงาน**

บิต $C/\overline{T2}$ - บิตเลือกโหมดการทำงาน ไทม์เมอร์ หรือ เคาน์เตอร์ 2

1 ไทม์เมอร์ หรือ เคาน์เตอร์ 2 **ทำงาน** โหมดเคาน์เตอร์โดยทริกแบบขอบขาลงที่ที่ขา T2EX pin และต้องเซ็ทบิตEXEN2ไว้ล่วงหน้า

0 ไทม์เมอร์ หรือ เคาน์เตอร์ 2 **เลือกการทำงานโหมดเคาน์เตอร์**

บิต CP/RL2 - บิตเลือกโหมดการทำงาน Capture หรือ Reload

1 ไทม์เมอร์ หรือ เคาน์เตอร์ 2 **ทำงาน** Capture mode โดยทริกแบบขอบขาลงที่ขา T2EX pin และต้องเซ็ทบิตEXEN2ไว้ล่วงหน้า

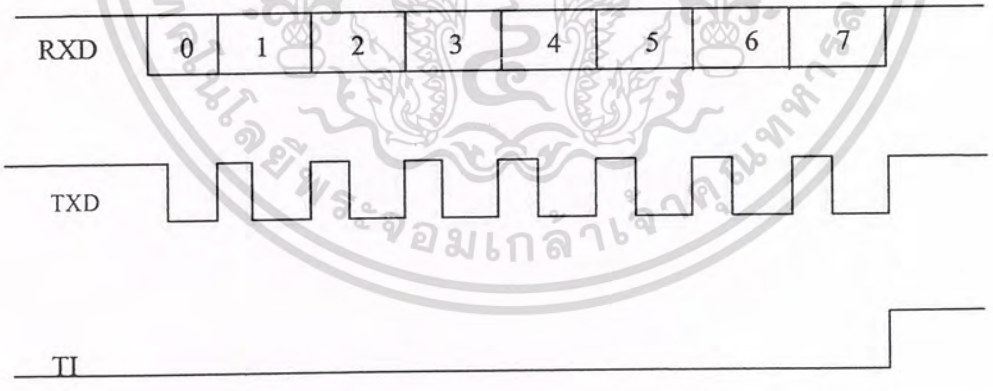
0 ไทม์เมอร์ หรือ เคาน์เตอร์ 2 **เลือกการทำงานแบบ** Reload mode โหลดค่าซ้ำอัตโนมัติเมื่อ TF2= 1 หรือ โดยการทริกแบบขอบขาลงที่ขา T2EX pin และต้องเซ็ทบิต EXEN2 ไว้ก่อน

2.6.6 การรับส่งข้อมูลอนุกรม (UART) ของ 8051 และ 8052

พอร์ทสื่อสารอนุกรมของ 8051, 8052 มีโครงสร้างการทำงานในแบบที่เรียกว่า ฟูลดูเพล็กซ์ (Full Duplex) ในการรับและส่งข้อมูลอนุกรมได้ในเวลาเดียวกัน โดยทางด้านส่งใช้ขา TxD (พอร์ท 3.1) ทางด้านรับใช้ขา RxD (พอร์ท 3.0) SBUF ใช้เป็นบัฟเฟอร์สำหรับรับและส่งข้อมูลอนุกรม

พอร์ทสื่อสารอนุกรมของ 8051 สามารถโปรแกรมการทำงานได้หลายโหมดด้วยกันโดยเลือกที่บิต SM1 และ SM0 ซึ่งอยู่ในรีจิสเตอร์ควบคุม SCON การทำงานทั้ง 4 โหมด ของพอร์ทสื่อสารอนุกรม มีดังนี้

โหมด 0 : พอร์ทสื่อสารอนุกรม 8 บิต โดยการส่งจะเลื่อนออกทีละบิตโดยส่งบิต D0 ออกไปก่อนทาง RxD และไม่มีการส่ง start bit แต่จะส่ง shift clock ทางขา TxD (ความเร็ว 1/12 เท่าของ CPU Clock)



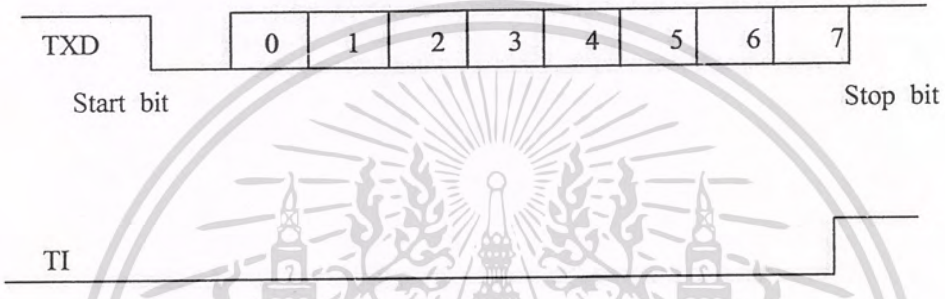
โหมด 1 : พอร์ทสื่อสารอนุกรม 10 บิต ข้อมูล 8 บิต 1 start bit และ 1 stop bit และสามารถเปลี่ยนแปลงความเร็วในการส่งข้อมูลได้ โดยขึ้นกับบิต SMOD ใน PCON และอัตราโอเวอร์โฟลว์ของ Timer 1)

$$\text{Baud Rate Mode 1} = \frac{K \times \text{Oscillator Freq.}}{32 \times 12 \times [256 - (TH1)]}$$

สำหรับ 8032,8052 โดยใช้ Timer 1

$$\text{Baud Rate Mode 1} = \frac{\text{Oscillator Freq.}}{32 \times [65536 - (RCAP2H,RCAP2L)]}$$

สำหรับ 8032,8052 โดยใช้ Timer 2

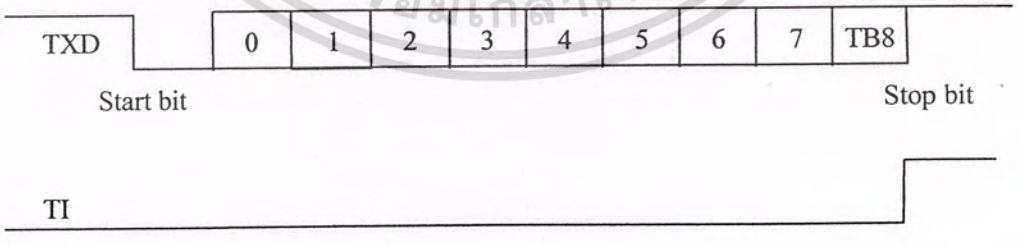


โหมด 2 : พอร์ทสื่อสารอนุกรม 11 บิต ใช้ข้อมูล 9 บิต 1 start bit และ 1 stop bit (TB8 นิยมนำมาใช้ส่ง Parity bit

(ความเร็วในการรับส่งข้อมูลเท่ากับ 1/32 และ 1/64 ของ CPU clock โดยขึ้นกับบิต SMOD ใน PCON)

Baud Rate Mode2 = 1/(32 Osc Freq) เมื่อ SMOD = 1

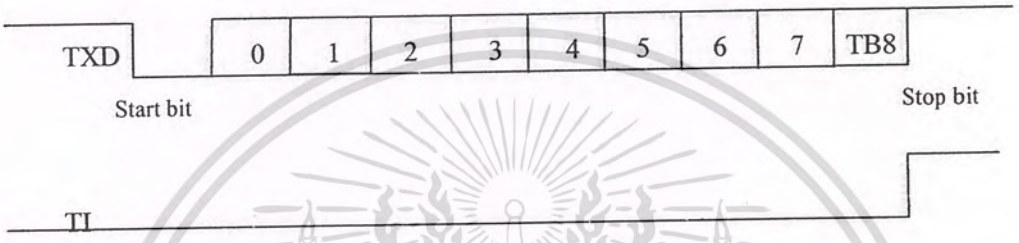
Buad Rate Mode2 = 1/(64 Osc Freq) เมื่อ SMOD = 0



โหมด 3 : พอร์ทสื่อสารอนุกรมแบบ 11 bit UART โดยส่งข้อมูล 9 บิต 1 start bit และ 1stop bit เหมือนโหมด 2 ยกเว้นอัตราความเร็วจะขึ้นกับบิต SMOD ใน PCON และอัตราโอเวอร์โฟลว์ของ Timer 1 สำหรับ 8051 หรือขึ้นกับ อัตราโอเวอร์โฟลว์ของ Timer 2 สำหรับ 80C154D

$$\text{Buad Rate Mode3} = \frac{\text{K x Oscillator Freq.}}{32 \times 12 \times [256 - (\text{TH1})]}$$ สำหรับ 8031,8051 โดยใช้ Timer 1

$$\text{Buad Rate Mode3} = \frac{\text{Oscillator}}{32 \times [65536 - (\text{RCAP2H,RCAP2L})]}$$ สำหรับ 8032,8052 80154,80154D โดยใช้ Timer



2.6.7 การเชื่อมต่อไมโครโปรเซสเซอร์เพื่อรับส่งข้อมูลอนุกรม (UART)

มีอยู่ 2 โหมดด้วยกันคือ

Single Processor Mode

Multiprocessors Mode

Single Processor Mode : ในโหมดนี้เราจะใช้ไมโครคอนโทรลเลอร์ 2 ตัวเชื่อมเข้าหากัน

Multiprocessors Mode : ในโหมดนี้เราจะใช้ไมโครคอนโทรลเลอร์ 1 ตัวเป็นตัวแม่ (Master) และอีก 256 ตัวลูก (Slave) รีจิสเตอร์ที่ใช้ควบคุมการรับส่งข้อมูลอนุกรม

ตารางที่ 2.3 แสดงการทำงานของ SM1 และ SM0 ในโหมดต่างๆ

SM1	SM0	โหมด	การทำงาน
0	0	0	Shift register อัตราเร็วในการรับหรือส่งข้อมูลเท่ากับ ½ ของความถี่ออสซิลเลเตอร์ของ CPU
0	1	1	8 bit UART อัตราเร็วในการรับหรือส่งข้อมูลขึ้นกับ Timer 1, 2 และ บิต SMOD
1	0	2	9 bit UART อัตราเร็วในการรับหรือส่งข้อมูล=1/32 หรือ 1/64 ของความถี่ออสซิลเลเตอร์ ขึ้นกับบิต SMOD ใน PCON
1	1	3	9 bit UART อัตราเร็วในการรับหรือส่งข้อมูลกำหนดที่ Timer 1, 2 และ บิต SMOD

SM2 บิตเลือกการทำงานแบบ Single Processor Mode หรือ Multiprocessor Mode

1 : เลือก Multiprocessor Mode ใช้ได้กับโหมด 2,3

0 : เลือก Single Processor Mode ใช้ได้กับทุกโหมด

เมื่อเลือกการทำงานรับข้อมูลแบบ Multiprocessor Mode แล้ว

ถ้าข้อมูลบิตที่ 9 ที่รับได้มีค่าเป็น **1** RI จะเซ็ท

ถ้ารับข้อมูลบิตที่ 9 ที่รับได้มีค่าเป็น **0** RI จะเคลียร์

REN (Receive Enable) บิตควบคุมให้รับหรือไม่รับข้อมูล

1 : ให้รับข้อมูลได้

0 : ห้ามรับข้อมูล

TB8 (Transmit bit D8) ข้อมูลบิตที่ 9 ที่จะส่งออกไปในโหมด 2, 3 ให้ใส่ในบิตนี้

RB8 (Receive bit D8) ข้อมูลบิตที่ 9 ที่รับเข้ามาจะเก็บในบิตนี้

(ข้อมูลบิตที่ 9ก็คือค่าใน TB8 ทางด้านส่งนั่นเอง)

TI แฟลกซ์ TI จะเป็น 1 เมื่อสิ้นสุดการส่งข้อมูลบิตสุดท้าย

RI แฟลกซ์ RI จะเป็น 1 เมื่อรับข้อมูลเสร็จบิตสุดท้ายเข้ามา (บิต RI,TI ผู้เขียนโปรแกรมจะต้องเคลียร์เอง)

ตารางที่ 2.4 ตารางการใช้ ไทม์เมอร์ 1 กำหนดบอดเรท

Baud Rate	Fosc	SMOD	TIMER 1		
			$\overline{C/T}$	MODE	Reload Value
(MODE 0) Max : 1 MHz	12 MHz	X	X	X	X
(MODE 2) Max : 375 KHz	12 MHz	1	X	X	X
(MODE 2) Min : 187.5 KHz	12 MHz	0	X	X	X
MODE 1, 3 : 62.5 KHz	12 MHz	1	0	2	FFH
19.2 KHz	11.059 MHz	1	0	2	FDH
9.6 KHz	11.059 MHz	0	0	2	FDH
4.8 KHz	11.059 MHz	0	0	2	FAH
2.4 KHz	11.059 MHz	0	0	2	F4H
KHz	11.059 MHz	0	0	2	E8H
137.5	11.059 MHz	0	0	2	1DH
110	6 MHz	0	0	2	72H
110	12 MHz	0	0	2	FEEBH
		0	0	1	

2.6.8 การอินเตอร์รัปต์

คือ การขัดจังหวะโปรแกรมชั่วขณะ แล้วมาทำโปรแกรมบริการอินเตอร์รัปต์ (Interrupt Service Routine; ISR) การตรวจสอบสัญญาณการต้องขออินเตอร์รัปต์จะตรวจสอบที่ตำแหน่ง S5P2 ของทุกๆ แมชชีนไซเคิลเมื่อพบแล้วในช่วงแมชชีนไซเคิลที่ 2 จะเป็นการตรวจสอบว่าเป็นของอุปกรณ์ใดและแมชชีนไซเคิลที่ 3 จะกระโดดไปทำโปรแกรมบริการอินเตอร์รัปต์

ตารางที่ 2.5 อินเตอร์รัปต์เวคเตอร์ของ MCS-51 และลำดับความสำคัญของการอินเตอร์รัปต์

ลำดับ	ชื่อสัญญาณอินเตอร์รัปต์	Vector Address	Priority
1	$\overline{\text{INT0}}$	0003H	Highest
2	TF0	000BH	
3	$\overline{\text{INT1}}$	0013H	
4	TF1	001BH	
5	TI+RI	0023H	
6	TF2+EXF2	002B	Lowest

* ถ้ามีอินเตอร์รัปต์เข้ามาพร้อมกัน $\overline{\text{INT0}}$ จะถือว่ามี Priority สูงสุด *

2.6.8.1 Interrupt Enable Register (IE) อยู่ใน SFR ตำแหน่งที่ (0A8H)

ใช้ควบคุมอินเตอร์รัปต์ได้ 8 แหล่ง ดูตารางที่ 2.6 ประกอบเราสามารถสั่งห้ามหรือไม่ห้ามการอินเตอร์รัปต์ได้จากรีจิสเตอร์ชุดนี้ดังมีรายละเอียดดังรูปที่ 2.26 เราสามารถสั่งห้ามไม่ให้เกิดการขัดจังหวะทั้งหมดก็ได้เพียงแต่ไปรีเซ็ทบิต $\overline{\text{EA}}$ ใน Interrupt Enable Register (IE) ถ้าต้องการ Enable อินเตอร์รัปต์จากอุปกรณ์ตัวไหนก็เพียงแต่ไปเซ็ทบิตของอุปกรณ์ตัวนั้นไว้ แต่อย่าลืมเซ็ทบิต $\overline{\text{EA}}$

ตารางที่ 2.6 แสดงรายละเอียดของรีจีสเตอร์ IE

บิต ชื่อบิต	การทำงาน
IE.7 $\overline{EA}$	=1 หมายถึงยอมให้เลือกการทำอินเตอร์รัปต์จากอินเตอร์รัปต์จากแหล่งได้ =0 หมายถึงไม่ยอมให้ทำอินเตอร์รัปต์จากแหล่งใดๆ ทั้งสิ้น
IE.6 X	ไม่ได้ใช้งาน
IE.5 IE.2	=1 หมายถึง ยอมให้ทำอินเตอร์รัปต์เมื่อ TF2 เกิดโอเวอร์โฟลว์ =0 หมายถึง ไม่ยอมให้ทำอินเตอร์รัปต์เมื่อ TF2 เกิดโอเวอร์โฟลว์
IE.4 ES	=1 หมายถึง ยอมให้ทำอินเตอร์รัปต์จากพอร์ทสื่อสารอนุกรมได้ =0 หมายถึง ไม่ยอมให้ทำอินเตอร์รัปต์จากพอร์ทสื่อสารอนุกรมได้
IE.3 ET1	=1 หมายถึง ยอมให้ทำอินเตอร์รัปต์เมื่อ TF1 เกิดโอเวอร์โฟลว์ =0 หมายถึง ไม่ยอมให้ทำอินเตอร์รัปต์เมื่อ TF1 เกิดโอเวอร์โฟลว์
IE.2 EX1	=1 หมายถึง ยอมให้ทำอินเตอร์รัปต์จากสัญญาณภายนอกหมายเลข 1 =0 หมายถึง ไม่ยอมให้ทำอินเตอร์รัปต์จากสัญญาณภายนอกหมายเลข 1
IE.1 ET0	=1 หมายถึง ยอมให้ทำอินเตอร์รัปต์เมื่อ TF0 เกิดโอเวอร์โฟลว์ =0 หมายถึง ไม่ยอมให้ทำอินเตอร์รัปต์เมื่อ TF0 เกิดโอเวอร์โฟลว์
IE.0 EX0	=1 หมายถึง ยอมให้ทำอินเตอร์รัปต์จากสัญญาณภายนอกได้ (ขา $\overline{INT0}$) =0 หมายถึง ไม่ยอมให้ทำอินเตอร์รัปต์จากสัญญาณภายนอก (ขา $\overline{INT0}$)

ความหมายของสัญญลักษณ์

E = Enable หรือ External

T = Timer

0,1,2 = Channel 0,Channel 1, Channel 2

- อินเตอร์รัปต์ภายใน MCS-51 ได้จาก Timer 0, Timer 1 โดยตรวจสอบที่ TF0 และ TF1 และอินเตอร์รัปต์จาก Serial Port โดยตรวจสอบที่ TI และ RI
- ขา INT และ INT ใช้งาน 2 หน้าที่ โดยที่

  เมื่อทำอินเตอร์รัปต์จะเป็นขา External Interrupt Input

  เมื่อไม่ทำอินเตอร์รัปต์ จะใช้ Start Counter หรือที่เราเรียกว่า Hardware Start

  การขัดจังหวะของการอินเตอร์รัปต์เราสามารถกำหนดลำดับความสำคัญได้จาก Interrupt Priority Register (IP) ดังรายละเอียดดังรูปที่ 2.16

2.6.8.2 Interrupt Priority Register (IP) อยู่ใน SFR ตำแหน่งที่ (0B8H)

ใช้กำหนดลำดับความสำคัญของการอินเตอร์รัปต์

PCT	-	PT2	PS	PT1	PX1	PT0	PX0

PCT IP.7 Defines the priority level for all the source interrupt (83C154 and 83C154D only).

IP.6 Not implemented, reseved for future use*.

PT2 IP.5 Defines the Timer2 interrupt priority level (80C52 and 83C154D only).

PS IP.4 Defines the Serial port interrupt priority level.

PT1 IP.3 Defines the Timer 1 interrupt priority level.

PX1 IP.2 Defines External Interrupt priority level.

PT0 IP.1 Defines the Timer 0 interrupt priority level.

PX0 IP.0 Defines the External interrupt 0 pririty level.

* User software should not write 1s to reserved bits. These bits may be used in future MHS C51 products to invoke new features. In that case, the reset or inactive value of the now bit will be 0, and its active value will be 1.

รูปที่ 2.16 รายละเอียดของ IP

คำอธิบายความหมายในแต่ละบิตใน (IP)

PT2 : 0 Timer 2 มีความสำคัญต่ำสุด

1 Timer 2 มีความสำคัญสูงสุด

PS : 0 พอร์ทสื่อสารอนุกรม UART มีลำดับความสำคัญต่ำสุด

1 พอร์ทสื่อสารอนุกรม มีลำดับความสำคัญสูงสุด

PT1 : 0 Timer 1 มีลำดับความสำคัญต่ำสุด

1 Timer 1 มีลำดับความสำคัญสูงสุด

PT0 : 0 Timer 0 มีลำดับความสำคัญต่ำสุด

1 Timer 0 มีลำดับความสำคัญสูงสุด

PX0 : 0 อินเตอร์รัปต์ภายนอกชนิด 0 มีลำดับความสำคัญต่ำสุด

1 อินเตอร์รัปต์ภายนอกชนิด 0 มีลำดับความสำคัญสูงสุด

PCT : 0 ยอมให้มีการจัดลำดับความสำคัญของการอินเตอร์รัปต์ (priority)

Interrupt Vector ของ MCS-51

มีอยู่ 8 แหล่งดังแสดงในตารางที่ 2.7 ดังนี้

ตารางที่ 2.7 Interrupt Vector

Source	Vector Address
IE0	0003H
TF0	000BH
IE1	0013H
TF1	001BH
RI+TI	0023H
*TF2+EXF2	002BH

* มีเฉพาะในเบอร์ 8052 ขึ้นไป

2.6.8.3 รีจีสเตอร์ที่ใช้ในการประหยัดพลังงาน Power Control Register (PCON) อยู่ใน SFR ตำแหน่งที่ (87H)

SMOD	HPD	RPD	-	GF1	GF0	PD	IDL

บิต SMOD - บิตกำหนดอัตราความเร็วการรับส่งข้อมูลอนุกรม UART

0 = อัตราความเร็ว 1 เท่า

1 = อัตราความเร็ว 2 เท่า

บิต HPD - Hard Power Down bit (มีใน 83C154, 83C154D เท่านั้น) หมายถึงการใช้สัญญาณจากภายนอกมากระตุ้นให้หยุดหรือเริ่ม Power Down mode เมื่อถูกรีเซ็ทจะหยุดการทำงานในโหมดนี้

บิต RPD - Recover Power Down bit (มีใน 83C154, 83C154D เท่านั้น) มันได้ถูกนำไปใช้สำหรับยกเลิก Power-Down/Idle mode

1 = ถ้าการร้องขออินเตอร์รัปต์ถูก Enable ไว้จะกระโดดไปทำโปรแกรมบริการอินเตอร์รัปต์

0 = ถ้าการร้องขออินเตอร์รัปต์ถูก Disable ไว้โปรแกรมจะทำงานต่อหลังจาก Power-Down/Idle Instruction

บิต GF1 - แฟลกซ์ใช้งานทั่วไป

บิต GF2 - แฟลกซ์ใช้งานทั่วไป

บิต PD - Power Down bit

1 = หยุดออสซิเลเตอร์ของซีพียูสัญญาณรีเซ็ทหรืออินเตอร์รัปต์ (83C154, 83C154D เท่านั้น) ที่จะยกเลิกโหมดนี้

บิต IDL - Idle Mode bit

1 = หยุดการทำงานของซีพียู สัญญาณรีเซ็ทหรืออินเตอร์รัปต์เท่านั้นที่จะยกเลิกโหมดนี้ได้

2.7 8255 Programmable Peripheral Interface

IC อีกตัวหนึ่งที่เป็นที่รู้จักกันดีและนิยมนำมาใช้งานร่วมกับ ไมโครคอนโทรลเลอร์ MCS-51 ในลักษณะงานควบคุมระบบที่พอร์ต I/O ของMCU ไม่เพียงพอต่อการใช้งานนั้น IC8255 PPI ตัวนี้ เป็น IC ที่ผลิตมาโดยมีคุณสมบัติในการควบคุมการทำงานที่ง่ายและมีจำนวนพอร์ต I/O ขนาด 8 บิต จำนวน 3 พอร์ต ภายในตัวจึงเป็น IC ที่น่าสนใจที่จะนำมาใช้งานในการเพิ่มขยายพอร์ต I/O ให้กับ MCU โดยรูปแบบโครงสร้างคุณสมบัติต่างๆมีดังนี้

- เป็นชิพขนาด 40 ขา
- มีพอร์ตใช้งาน 3 พอร์ต แบ่งออกเป็นพอร์ต A,B และ C
- พอร์ตใช้งานมีลักษณะเป็นพอร์ตแล็ซ(Latch)คงสภาวะ

รูปที่ 2.17 แสดงลักษณะการจัดขา 8255 PPI

- ควบคุมการทำงานด้วยขา AO และ A1
- สามารถโปรแกรมเลือกลักษณะการใช้งานพอร์ตได้ ทั้งในลักษณะ Input, Output หรือ Bi-Directional พอร์ต

2.8 คุณสมบัติขาสัญญาณต่างๆของ8255

Vcc เป็นขารับไฟเลี้ยงของระบบ +5 V

GND เป็นขาต่อกราวด์ของระบบ

D0-D7 เป็นขา Bi-Directional Bus โดยเป็น Data Bus ของระบบ

RESET เมื่อขานี้มีสภาวะ high จะทำให้ 8255 ถูกรีเซ็ตมีผลทำให้ทุกพอร์ตเปลี่ยนเป็นอินพุตพอร์ตทันที หรือในลักษณะอินพุตโหมด

CS (Chip Select) เป็นขาที่ใช้เป็นตัวควบคุมเลือกจังหวะการทำงานของ 8255จะทำงานเมื่อขานี้อยู่ใน สภาวะ low

RD (Read) เมื่อขานี้มีสภาวะ low MCU จะทำการอ่านข้อมูลผ่านทาง Data Busของ 8255 ได้โดยพร้อมกับขา CS ต้องมีสภาวะ low

WR (Write) เมื่อขานี้มีสภาวะ low 8255 จะทำการรับข้อมูลหรือ control word ที่ส่งมาจาก MCU ได้โดยพร้อมกับขา CS ต้องมีสภาวะ low

A0-A1 ใช้ในการรับสัญญาณควบคุมหรือ เลือกการใช้งานพอร์ต A, B และCของ8255

PA0-PA7 เป็นพอร์ตI/O ขนาด 8 บิต

PB0-PB7 เป็นพอร์ต I/O ขนาด 8 บิต

PC0-PC7 เป็นพอร์ตขนาด I/O ขนาด 8 บิต และมีคุณสมบัติแตกต่างจากพอร์ต A และ B คือ สามารถแบ่งการใช้งานเป็น 4 บิต(C4-C7) และ 4 บิตล่าง(C0-C3)ได้

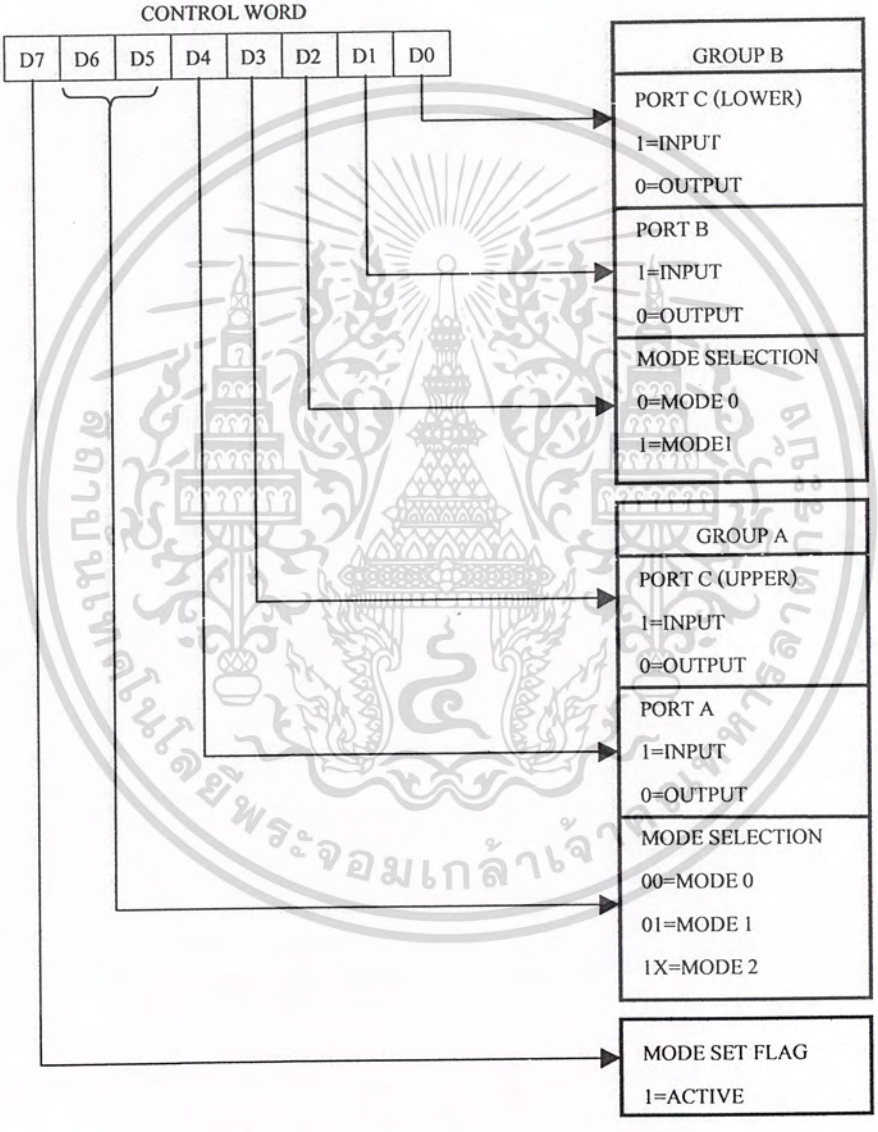
8255 PPI นั้น สามารถที่จะเลือกโหมด การใช้งานได้จากคำสั่งทาง Software จาก MCU ผ่าน Data Bus D0และ D7 ของ 8255โดยโหมดการใช้งาน 8255 มีด้วยกัน 3 โหมด คือ โหมด 0 (Basic I/O) เป็นลักษณะการใช้งาน พอร์ต A, B และC ของ 8255 เป็น I/O ปกติ โหมด 1 (Strobed I/O) จะมีลักษณะการตรวจสอบสัญญาณ Strobe ก่อนที่จะทำงานโหมด 2 (Bi-directional bus) จะใช้งานพอร์ตในลักษณะI/O แบบ 2 ทิศทาง(รับส่งข้อมูล)

ตารางที่ 2.8 ตารางความจริงของ 8255

A1	A0	RD	WR	CS	INPUT OPERATION (READ)
0	0	0	1	0	Port A → Data Bus
0	1	0	1	0	Port B → Data Bus
1	0	0	1	0	Port C → Data Bus
1	1	0	1	0	Control Word → Data Bus
					OUTPUT OPERATION (WRITE)
0	0	1	0	0	Data Bus → Port A
0	1	1	0	0	Data Bus → Port B
1	0	1	0	0	Data Bus → Port C
1	1	1	0	0	Data Bus → Control

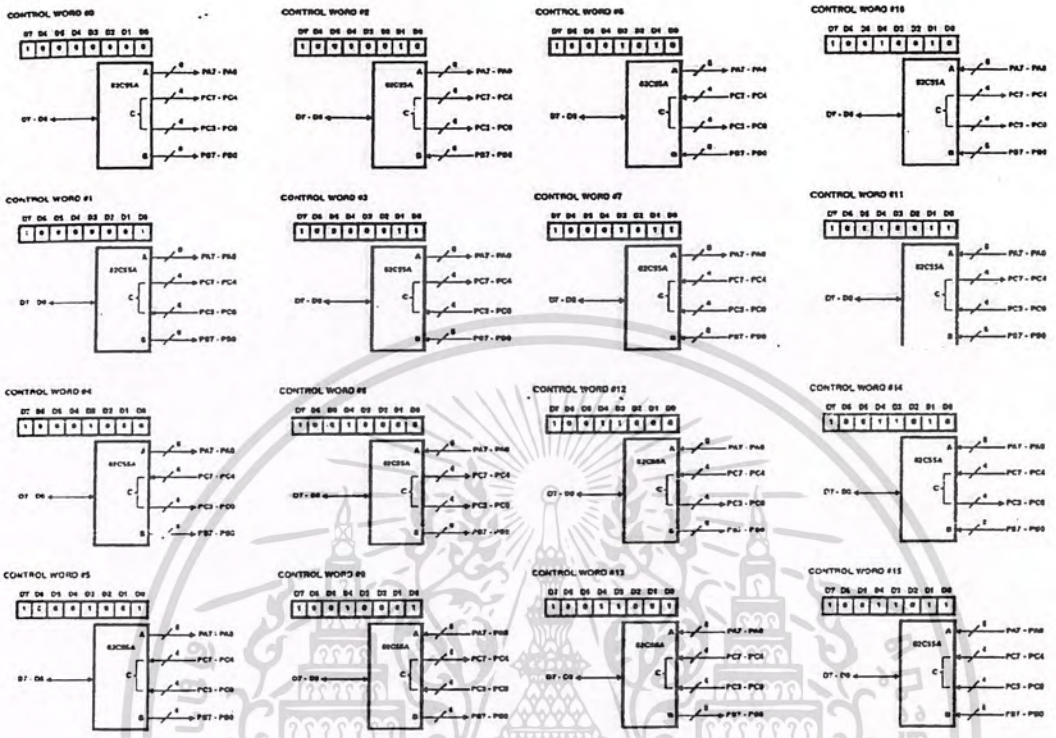
DISABLE FUNCTION					
X	X	X	X	1	Data Bus → Three-State
X	X	1	1	0	Data Bus → Three-State

โดยข้อมูลที่ถูกส่งไปควบคุมการทำงานของ 8255 ผ่านทาง Data bus D0-D7 นั้นจะถูกเรียกว่า Control Word หรือ Control Code โดยมีรูปแบบหรือคำสั่งอยู่ในแต่ละบิตของชุด Control Word ที่ส่งไปควบคุม 8255 ดังรูปที่ 2.18



รูปที่ 2.18 แสดงลักษณะการจัด control word

2.9 ลักษณะการจัด Control Word Mode 0



รูปที่ 2.19 แสดงลักษณะการจัด Control Word Mode 0

เมื่อเริ่มต้นก่อนที่จะใช้งาน 8255 นั้นจำเป็นจะต้องกำหนดโหมดการใช้งาน พอร์ตต่างๆ ของ 8255 ในลักษณะใดด้วย ซึ่งคำสั่งในการกำหนดโหมดการใช้งานรวมถึงลักษณะการใช้งานพอร์ตต่างๆของ 8255 ทั้งหมดที่กล่าวมาจะถูกเรียกว่า Control Word ดังในรูปที่ 2.18 จะแสดงลักษณะการจัด Control Word ในการควบคุม 8255 โดยจะเห็นว่าการใช้งาน 8255 ในโหมด 0 นั้น ชุดคำสั่งของ Control Word ในบิตที่ 2, 5 และ 6 จะต้องเป็น 0 ส่วนบิตที่เหลือจะเป็นบิตที่กำหนดการใช้งาน พอร์ตของ 8255 ในลักษณะ I/O พอร์ต สรุปแล้วก็คือ ชุดคำสั่งหรือ Control Word ที่กล่าวมาได้ถูกสรุปไว้ในตารางที่ 2.18 หรือถ้าจะดูรายละเอียดค่าแต่ละบิตของ Control Word สามารถดูได้จากหัวข้อ 2.19

ตาราง 2.8 เป็นตารางความจริงของ 8255 โดยจะพบว่า A0 และ A1 เป็นตัวกำหนดการใช้งานพอร์ตของ 8255 ว่าจะใช้งานพอร์ตใด ที่นี้ในการที่จะส่ง Control Word ไปยัง 8255 เพื่อให้ 8255 รับรู้ว่าเป็นชุดข้อมูลซึ่งเป็น Control wordไม่ใช่ชุด Data ธรรมดา จะต้องเซ็ตค่าให้ A0 และ

A1 เป็น 1 ทั้งคู่ หลังจากนั้นขา A0 และ A1 ก็จะถูกใช้งานเป็นขาเลือก (Select) ความต้องการใช้พอร์ตของ 8255 ว่าต้องการรับส่งข้อมูลกับพอร์ตใดของ 8255 ต่อไป

ตาราง ที่ 2.9 สรุปการทำงาน 8255 PPI โหมด 0

PORT A (PA0-PA7)	PORT B (PB0-PB7)	PORT C (PC4-PC7)	PORT C (PC0-PC3)	CONTROL WORD(HEX)
OUTPUT	OUTPUT	OUTPUT	OUTPUT	80H
OUTPUT	OUTPUT	OUTPUT	INPUT	81H
OUTPUT	INPUT	OUTPUT	OUTPUT	82H
OUTPUT	INPUT	OUTPUT	INPUT	83H
OUTPUT	OUTPUT	INPUT	OUTPUT	88H
OUTPUT	OUTPUT	INPUT	INPUT	89H
OUTPUT	INPUT	INPUT	OUTPUT	8AH
OUTPUT	INPUT	INPUT	INPUT	8BH
INPUT	OUTPUT	OUTPUT	OUTPUT	90H
INPUT	OUTPUT	OUTPUT	INPUT	91H
INPUT	INPUT	OUTPUT	OUTPUT	92H
INPUT	INPUT	OUTPUT	INPUT	93H
INPUT	OUTPUT	INPUT	OUTPUT	98H
INPUT	OUTPUT	INPUT	INPUT	99H
INPUT	INPUT	INPUT	OUTPUT	9AH
INPUT	INPUT	INPUT	INPUT	9BH

สรุปการใช้งาน 8255 ในโหมด 0 อย่างง่ายๆ ออกเป็น 2 ขั้นตอนคือ

1. ทำการเซ็ต A0 และ A1 ของ 8255 ให้เป็น 1 ทั้งคู่ แล้วทำการส่ง Control Word 8 บิต ไปยัง 8255 เพื่อเริ่มการใช้งาน

2. เลือกใช้งานพอร์ตของ 8255 ได้อย่างอิสระ โดยการควบคุมขา A0 และ A1 ของ 8255 ในตารางที่ 2.9

ที่ผ่านมานั้นเป็นการกล่าวถึงคุณสมบัติการใช้งาน โดยเบื้องต้นของ 8255 เพียงเท่านั้น ซึ่งการนำมาประยุกต์ใช้งาน ก็จะกล่าวเน้นในการใช้งาน 8255 ใน Mode 0 ซึ่งก็คือการใช้งาน 8255 เป็น Basic I/O พอร์ต เพื่อนำไปใช้งานร่วมกับ MCU ให้ใช้งานได้มากยิ่งขึ้น

2.10 การอินเตอร์เฟส MCS-51 กับ 8255 PPI

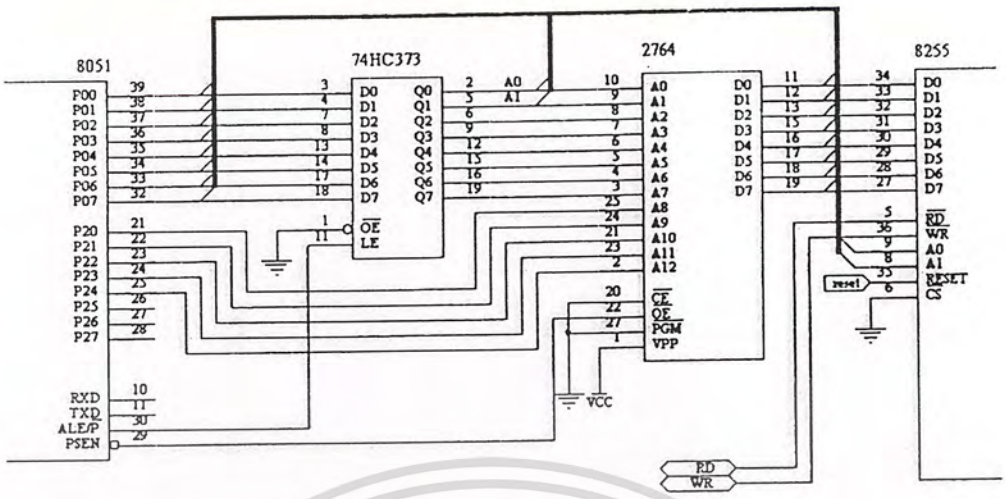
การประยุกต์ใช้งานการอินเตอร์เฟสไมโครคอนโทรลเลอร์ MCS-51 ร่วมกับอุปกรณ์ภายนอก ที่นิยมใช้งานเพื่อเพิ่มประสิทธิภาพในการใช้งานกับ MCU การอินเตอร์เฟส MCS-51 กับ 8255 โดยมีการใช้งานกับ EPROM ภายนอก / ภายในของ MCU ซึ่งจากวงจรดังรูปที่ 4.1 เป็นการต่อวงจรใช้งาน MCU กับ EPROM ภายนอกและมีการขยายพอร์ต I/O เพิ่มขึ้นโดยการต่อร่วมกับ IC 8255 PPI จากวงจรจะเห็นว่าจะมีการใช้ IC 74HC373 ในการ Latch แอดเดรสไบต์ต่ำ A0-A7ในการรับส่งข้อมูลกับ EPROM และ 8255 โดย IC74HC373 ยังถูกใช้งานใน Latch Control Word ,A0 และ A1 ให้กับ 8255 ได้ง่ายโดยมีการจัด Memory Map เพื่อการใช้งาน 8255 จากวงจรในรูปที่ 2.19 ดังต่อไปนี้

2.11 การอินเตอร์เฟส MCS-51 กับ 8255 โดยใช้ EPROM ภายนอก

จากตารางที่ 2.10 จะเป็นการแสดงการจัด Memory Map ของวงจรในรูปที่ 2.19 จะเห็นว่าเมื่อต้องการใช้งาน PORT A ,B และ C หรือส่ง Control Word เพื่อทำการรีเซ็ต 8255 ให้ทำงานตามความต้องการนั้นๆ ก็เพียงแต่อ้างแอดเดรสของหน่วยความจำเท่านั้น เช่น ต้องการใช้งาน PORT A ก็เพียงแต่อ้างแอดเดรสของหน่วยความจำ 00H เท่านั้น ก็จะทำให้ขา A0 และ A1 ของ 8255 มีค่าเป็น ลอจิก 0 หรือ สภาวะ LOW ก็จะเป็นการควบคุม 8255 ในการใช้งาน PORT A นั้นเอง ต่อไปจะขอยกตัวอย่างการใช้งาน 8255 PORT A โดยมีวงจรการใช้งานในรูปที่ 2.19 และมีการจัด Memory Map ตามตารางที่ 2.10

ตารางที่ 2.10 แสดงการจัด Memory Map ของวงจรที่ 2.19

Memory Map	Address	A1	A0
PORT_A	EQU 00H	0	0
PORT_B	EQU 01H	1	1
PORT_C	EQU 02H	0	1
Control Word	EQU 03H	0	1



รูปที่ 2.20 แสดงการอินเตอร์เฟส MCU, 8255 PPI, EPROM ภายนอก

ตัวอย่างที่ 2.20 โปรแกรมตัวอย่างการใช้งาน 8255 PORT_A ของวงจรในรูปที่ 2.20 โดยการทำงานของโปรแกรม จะทำการส่งค่า 00001111B หรือ 0FH ออกไปยัง 8255 PORT _A

```
; ***  Exsemple  For  circuit  fig 2.20  ***
PORT_A          EQU    00H
CTRL_WORD       EQU    03H
ORG             0000H
ACALL           DELAY
ACALL           CTRL_8255
MOV             A,#00001111B
MOV             R1,#PORT_A
MOVX            @R1,A
END
```

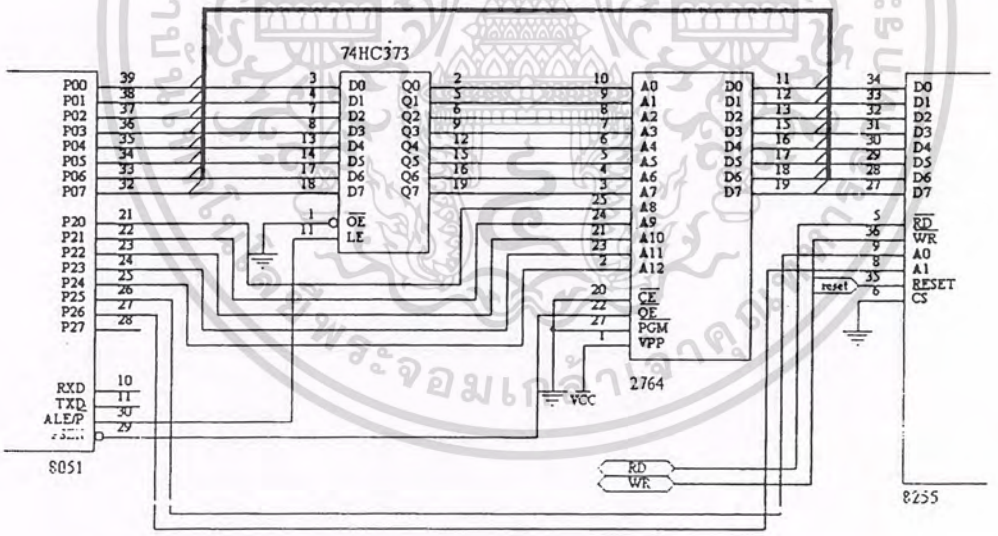
อธิบายการทำงานโปรแกรมตัวอย่างที่ 2.20

จากโปรแกรมที่ 2.20 เริ่มต้นโปรแกรมจะมีการกำหนดตัวแปรค่าคงที่ต่างๆที่จะนำมาใช้ในโปรแกรมโดยจะเห็นว่าพอร์ต A แทน 00Hและ CTRL_WORD แทน 03H จากนั้นจะเข้าสู่ Main Program โดยเริ่มต้นที่ตำแหน่ง 0000H ของหน่วยความจำโปรแกรม จะเห็นว่าเริ่มต้นจะมีการเรียกใช้ Sub Function Delay เพื่อหน่วงเวลาให้กับอุปกรณ์ภายนอกที่ต่อร่วมกับ MCU ให้ทำการเซ็ตตัวเพื่อพร้อมที่จะรับข้อมูลหรือสัญญาณต่างๆ ที่ส่งมาจาก MCU เนื่องจากกรณีการทำงานภายใน

ตัว MCU นั้น มีความเร็วในการทำงานสูงกว่าอุปกรณ์ภายนอกนั้นเองจึงอาจทำให้อุปกรณ์ที่นำมาต่อร่วมภายนอกไม่สามารถเริ่มต้นการทำงานได้ทัน โดยความเร็วของ CPU ในตัว MCU นั้นจะขึ้นอยู่กับ X-TAL ที่นำมาต่อใช้งาน ซึ่งเมื่อ X-TAL ยิ่งมีความถี่สูงมาก (ในหน่วย MHz) ก็อาจจำเป็นที่จะต้องใช้การหน่วงเวลาการทำงานให้กับ CPU ในตัว MCU ให้มากขึ้น

หลังจากการเรียกใช้ Sub Function Delay แล้ว ต่อมาจะมีการเรียกใช้ Sub Function CTRL_8255 เพื่อทำการส่ง CTRL_WORD ไปยัง 8255 เพื่อทำการเซ็ตโหมดและลักษณะพอร์ตใช้งานของ 8255 ถัดมาเป็นคำสั่งในการย้ายค่า 00001111B หรือ 0FH เก็บไว้ในแอกคิวมูเลเตอร์ จากนั้นก็จะเป็นคำสั่งในการย้ายค่า PORT_A (โดยเรากำหนดไว้ในตอนต้นโปรแกรม PORT_A ก็คือ 00H) เก็บไว้ในรีจิสเตอร์ R1(Indirect Addressing) จากนั้นก็ถึงคำสั่งสุดท้ายคือ คำสั่งเคลื่อนย้ายข้อมูลกับอุปกรณ์ภายนอกโดยใช้คำสั่ง MOVX จะเป็นการนำค่าในแอกคิวมูเลเตอร์ออกไปยังตำแหน่งที่รีจิสเตอร์ R1 ชี้อยู่ (00H) หรือ คือ การใช้งาน PORT_A ของ8255 นั้นเอง

โดยผลที่ได้นั้น ที่ PORT_A ของ 8255 จะมีค่าเป็น 00001111B หรือ 0FH ซึ่งเราสามารถใช้ LED ต่อเพื่อตรวจสอบก็ได้หรือจะตรวจสอบด้วยวิธีใช้ Multimeter ตรวจวัดก็ได้เช่นกัน โดย 0 ก็คือ 0 โวลท์ และ 1 ก็คือ 5 โวลท์นั่นเอง



รูปที่ 2.21 แสดงการประยุกต์ใช้ P2.5, P2.6 ควบคุม A0, A1 ของ 8255 โดยตรง

จากวงจรในรูปที่ 2.21 นั้นแสดงการประยุกต์โดยการนำพอร์ต 2 ที่เหลือจากการอ้างแอดเดรสไบต์สูง หรือจากการติดต่อกับ EPROM เบอร์ 2764 ทำให้เหลือขา P2.5, P2.6 และ P2.7 ของพอร์ต 2 โดยขาใช้งานที่เหลือนี้สามารถนำมาประยุกต์ใช้เป็นตัวควบคุม 8255 ที่ขา A0 และA1 โดยการต่อวงจรดังรูปที่ 2.21 ซึ่งการต่อวงจรในลักษณะนี้ในการจัด Memory Map ก็จะแตกต่าง

ไปจาก Memory Map ของวงจรในรูปที่ 2.20 โดยจะมีการจัด Memory Map ดังแสดงในตารางที่ 2.11

ตารางที่ 2.11 แสดงการจัด Memory Map ของวงจรที่ 2.21

Memory	Address	A1(P2,6)	A0(P2,5)
PORT_A	EQU 1FFFH	0	0
PORT_B	EQU 3FFFH	0	1
PORT_C	EQU 5FFFH	1	0
Control Word	EQU 7FFFH	1	1

จาก Memory Map ในตารางที่ 2.11 จะเห็นว่าหลักการต่างๆ ในการอ้างแอดเดรสของของหน่วยความจำก็เพียงเพื่อควบคุมขา A0 และA1 ของ 8255 เท่านั้น ซึ่ง A0 และ A1 ยังคงมีหน้าที่เหมือนเดิมคือ A0 = 0 และ A1= 0 ก็คือใช้งาน PORT_A ของ 8255 โดยพอร์ตอื่นๆ ก็ยังคงมีการควบคุมด้วย A0 และ A1 หมือนเดิมเปลี่ยนแต่เพียงแอดเดรสของหน่วยความจำในการอ้างถึงหรือใช้งาน

เมื่อมีการจัด Memory Map ในตารางที่ 2.11 จะเห็นว่ามีการกำหนดให้ PORT A เป็นตัวแปรที่ใช้แทนค่า 1FFFH ฉะนั้นในโปรแกรมการใช้งานการใช้คำสั่ง MOV DPTR,#PORT_A จะเป็นการให้ DTPR เก็บค่าหรือชี้ค่า 1FFFH เอาไว้ (โดยในตอนนี้ MCU จะมองค่า 1FFFH เป็นค่าคงที่ธรรมดา) จากนั้นเมื่อใช้คำสั่ง MOVX@DPTR,A จะเห็นว่าคำสั่ง MOVX นั้นเป็นคำสั่งในการติดต่อกับอุปกรณ์ภายนอกโดยผ่ายทางพอร์ต Address ของ MCU (พอร์ต 0 และ พอร์ต 2) ซึ่งจากคำสั่งข้างต้นจะเป็นการนำค่าในแอกคิวมูเลเตอร์ไปเก็บไว้ในหน่วยความจำภายนอกที่ชี้โดย DPTR (ตอนนี้ MCU จะเห็นว่าค่า 1FFFH ก็คือตำแหน่งของหน่วยความจำภายนอกนั่นเอง) ฉะนั้นก่อนที่จะทำการนำค่าในแอกคิวมูเลเตอร์ออกไปยังตำแหน่งที่ DPTR ชี้อยู่นั้น ที่พอร์ต Address (พอร์ต 0 และพอร์ต 2) ของ MCU ก็จะมีค่า 1FFFH (PORT_A) หรือ 0001 1111 1111 B ซึ่งเป็นการเซ็ตการใช้งาน 8255 PORT_A นั้นเอง จากนั้นจึงจะนำค่าในแอกคิวมูเลเตอร์ส่งออกไป ซึ่งก็เท่ากับว่าเมื่อต้องการใช้ 8255 PORT_A ก็เพียงแต่อ้างตำแหน่งของหน่วยความจำภายนอกโดยทำการกำหนดเป็นตัวแปรที่เข้าใจง่าย นั่นก็คือ PORT_A, B และ C แทนการใช้งาน 8255 PORT_A, B และ C ตามลำดับซึ่งที่อธิบายมาทั้งหมดก็คือประโยชน์ของการจัด Memory Map

เพื่อให้เข้าใจได้ง่ายขึ้น จากแอดเดรส 1FFFH จะเห็นว่าขา P2.5 , P2.6 และ P2.7 จะมีสภาวะเป็น LOW หรือ 0 ทั้งหมดเมื่อมีการอ้างแอดเดรสนี้ ซึ่งจากวงจรที่ 2.20 จะนำ P2.5 ต่อกับขา A0 และ P2.6 ต่อกับขา A1 ซึ่งเมื่ออ้างแอดเดรสข้างต้น ก็จะทำให้ขา A0 และ A1 ของ 8255 เป็น

0 ทั้งคู่ ซึ่งก็คือการเลือกใช้งาน PORT_A นั่นเอง การอ้างแอดเดรสอื่นๆก็จะมีหลักการดังที่กล่าวมาแล้วทั้งสิ้น

จากวงจรในรูป 2.21 เป็นวงจรตัวอย่างในการประยุกต์ การอ้างแอดเดรสเพื่อให้ผู้อ่านมีความเข้าใจในการจัดตำแหน่งเพื่อสร้าง Memory Map ในการใช้งานต่อวงจรในรูปแบบอื่นๆได้

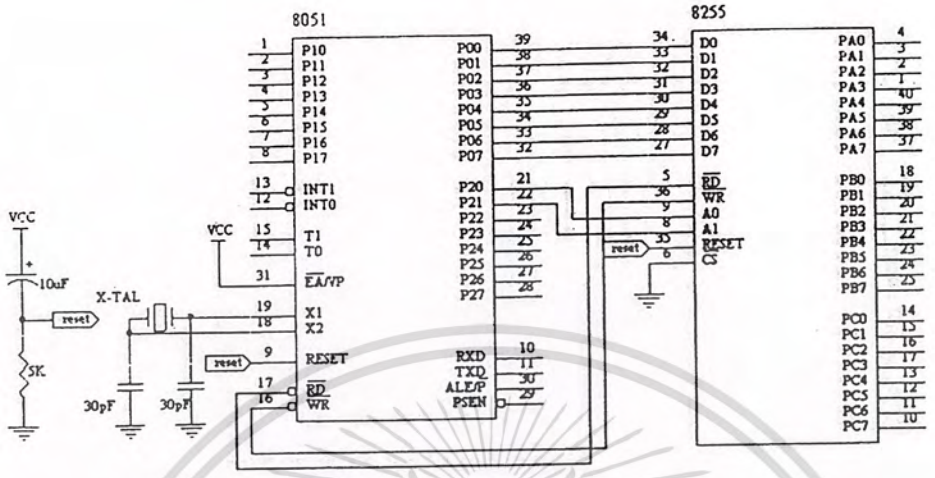
ตัวอย่างที่ 2.21 โปรแกรมตัวอย่างการใช้งาน 8255 PORT_A โดยการทำงานของโปรแกรมจะทำการส่งค่า 00001111B หรือ 0FH ออกไปยัง 8255 PORT_A เช่นเดียวกันกับโปรแกรมตัวอย่างที่ 2.20 แต่เปลี่ยนวงจรใช้งานดังวงจรใน รูปที่ 2.20 และ การจัด Memory Map ตามตารางที่ 2.11

```
; *** Exsemple For circuit fig 2.21 ***
PORT_A          EQU   1FFFH
CTRL_WORD       EQU   7FFFH
ORG             0000H
ACALL           DELAY
ACALL           CTRL_8255
MOV             A,#00001111B
MOV             DPTR,#PORT_A
MOVX            @DPTR,A
END
```

อธิบายการทำงานโปรแกรมตัวอย่างที่ 2.21

จากตัวอย่างโปรแกรมสำหรับวงจรดังรูปที่ 2.21 นั้น จะมีการทำงานเหมือนกับโปรแกรมตัวอย่างที่ 2.20 จะแตกต่างกันเพียงการจัด Memory Map โดยPORT_A จะมีการจัดที่เปลี่ยนค่าจาก 00H ในโปแกรมตัวอย่างที่ 2.20 ไปเป็น 1FFFHเท่านั้น รวมถึง CTRL_WORD

2.12 การอินเตอร์เฟส MCS-51 กับ 8255 โดยใช้ EPROM ภายใน

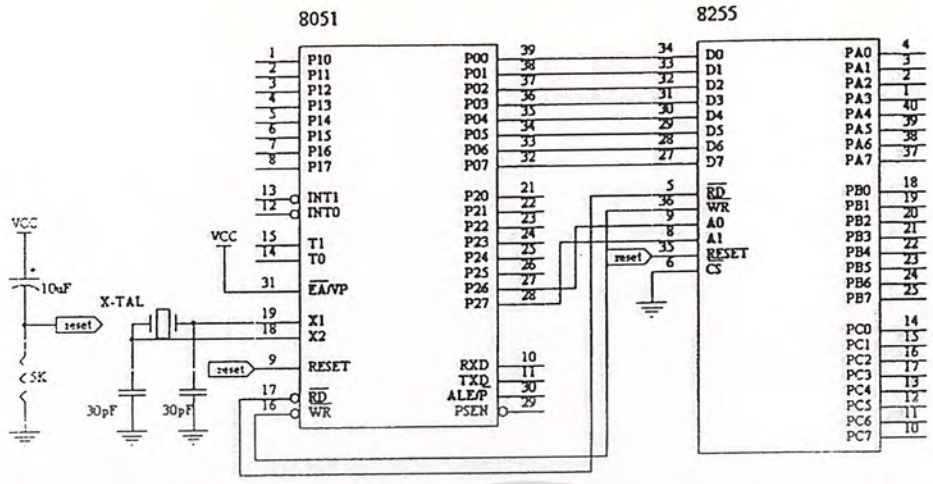


รูปที่ 2.22 แสดงการอินเตอร์เฟส MCU, 8255, EPROM ภายในรูปแบบที่ 1

ตารางที่ 2.12 แสดงการจัด Memory Map

Memory Map	Address	A1(P2,1	A0(P2,0)
PORT_A	EQU 0FFFH	0	0
PORT_B	EQU 1FFFH	0	1
PORT_C	EQU 2FFFH	1	0
Control Word	EQU 3FFFH	1	1

จากวงจรในรูปที่ 2.22 จะเห็นว่า จะใช้ PORT_0 เป็น Data Bus และใช้ PORT_2 เป็นพอร์ตในการควบคุม A0 และ A1 ของ 8255 ซึ่งจะมีการจัด Memory Map ดังตารางที่ 2.12 ซึ่งจะเห็นว่า เมื่อต่อวงจรในลักษณะนี้แล้ว PORT_2 ก็จะถูกใช้งานในการอ้างแอดเดรส ในลักษณะของ Memory Map ให้กับขา A0,A1 ของ 8255 โดยการอ้างแอดเดรสนั้นจะใช้งานทั้งพอร์ตของ PORT_2 คือ P2.0, P2.7 นั่นเองทำให้เกิดข้อจำกัดในการใช้งานพอร์ตลดลง



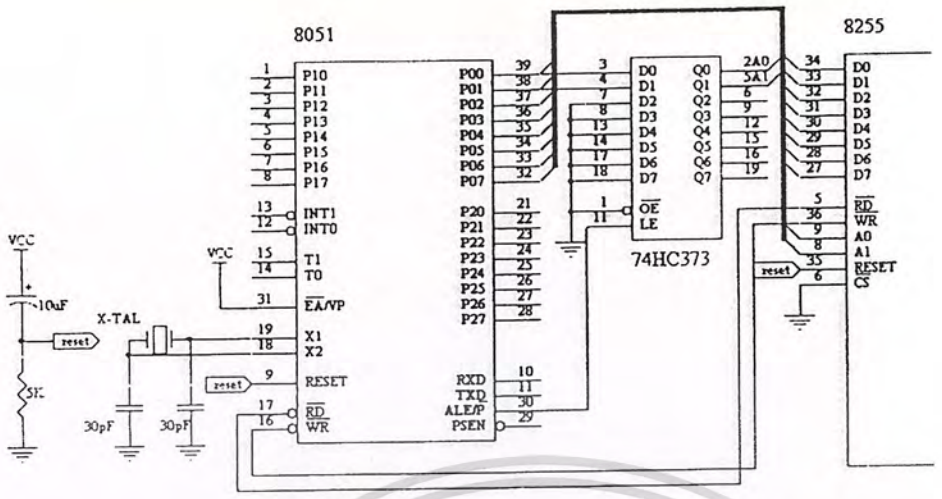
รูปที่ 2.23 แสดงการอินเตอร์เฟส MCU, 8255, EPROM ภายในรูปแบบที่ 2

จากวงจรจะไม่สามารถใช้งาน PORT_0 และPORT_2 ในลักษณะอื่นได้อีก แต่จะมีพอร์ตให้ใช้งานได้ 5 พอร์ต คือ PORT_1,PORT_3,8255 PORT_A, B และ C ซึ่งถ้าเพียงพอต่อการใช้งานแล้วก็ไม่ต้องคำนึงถึง PORT_0 และ PORT_2 อีก สามารถต่อวงจรการใช้งานตามวงจรข้างต้นได้เลย

ตารางที่ 2.13 แสดงการจัด Memory Map ของวงจรที่ 2.23

Memory Map	Address	A1(P2,7)	A0(P2,6)
PORT_A	EQU 0FFFH	0	0
PORT_B	EQU 4FFFH	0	1
PORT_C	EQU 8FFFH	1	0
Control Word	EQU CFFFH	1	1

จากวงจรในรูปที่ 2.23 จะต่างจากวงจรในรูปที่ 2.22 โดยการใช้งานของ P2.6 และ P2.7 แทน P2.0 และ P2.1 ในการควบคุมขา A0 และ A1 ของ 8255 ซึ่งถ้าใช้ P2.7 เป็น A1 และ P2.6 เป็น A0 ก็จะสามารถจัด Memory Map ได้ใหม่ดังตารางที่ 2.23 โดยเปลี่ยนแต่ Memory Map เท่านั้น แต่การทำงานในรูป 2.22 และรูปที่ 2.23 ก็ยังคงเหมือนกันซึ่งในวงจรที่ 2.23 สร้างขึ้นเพื่อให้เข้าใจการใช้ Memory Map ได้ดียิ่งขึ้น



รูปที่ 2.24 แสดงการอินเตอร์เฟส MCU, 8255, EPROM ภายในรูปแบบที่ 3

ตารางที่ 2.14 แสดงการจัด Memory Map ของวงจรที่ 2.24

Memory Map	Address	A1(P2,7)	A0(2,6)
PORT_A	EQU 00H	0	0
PORT_B	EQU 01H	0	1
PORT_C	EQU 02H	1	0
Control Word	EQU 03H	1	1

จากวงจรในรูปที่ 2.24 จะเห็นได้ว่าการนำ IC 74HC373 เข้ามาใช้ในวงจรก็เพื่อใช้ในการ LATCH แอดเดรสไบต์ต่ำ แต่จะ LATCH เพียง 2 บิตเท่านั้น คือ P0.0 และ P0.1 เพื่อใช้เป็นบิตควบคุม A0และ A1ให้กับ 8255 โดยจะทำให้มี Memory Map ในการควบคุม 8255 แสดงดังตารางที่ 2.14 ซึ่งจะต่อวงจรในรูปที่ 2.24 จะทำให้มีพอร์ตใช้งานถึง 6 พอร์ต คือ PORT 1, 2, 3, 8255 PORT A, B และ C ซึ่งการใช้งานแต่ละพอร์ต สามารถใช้งานตามคุณสมบัติของมันได้เต็มที่

จากรูปวงจรหลักๆที่กล่าวมาแล้วทั้งหมดนั้นสามารถนำไปต่อใช้งานได้จริงทุกวงจร แต่ในรูปที่ 2.20และรูปที่ 2.21 นั้น จะใช้ในการทดลองมากกว่าในการใช้งานจริงเนื่องจากสามารถใช้ EPROM Emulator ในการทดลอง (Simulator) การทำงานต่างๆได้ เมื่อทำการ Simulate ได้ผลตามความต้องการหรือบรรลุเป้าหมายแล้ว สามารถนำโปรแกรมนั้นไปทำการเปลี่ยน Memory Map ให้ตรงกับวงจรทางด้าน Hardware ที่ออกแบบ จากนั้นทำการ Down load โปรแกรมที่ถูกต้องสมบูรณ์ลง EPROM ภายในชิป MCS-51ที่มี EPROM ในตัวก็สามารถทำงานได้เหมือนกับโปรแกรมแม่แบบทุกประการ อย่าลืมว่าจากที่กล่าวมาทั้งหมดจะนิยมใช้วงจรในรูปที่ 2.20 และ

2.21 เป็นวงจรทดลองเพราะจำนวน Hardware ที่มาก ทั้งยังมีความซับซ้อนในการออกแบบวงจรพิมพ์ จึงไม่นิยมที่จะนำไปออกแบบในการใช้งานจริง ซึ่งเป็นข้อเสียที่สำคัญในวงจร รูปที่2.20 และ 2.21 โดยจะเห็นว่าวงจรในรูปที่ 2.22 ถึง 2.24 ก็สามารถใช้งานได้เช่นเดียวกันกับวงจรในรูปที่ 2.20 และ 2.21 ทุกประการเช่นกัน เพียงแต่ไม่สามารถใช้ EPROM Emulator ในการ Simulate เท่านั้น แต่เมื่อนำวงจรในรูปที่ 2.22 ถึง 2.24 ไปใช้งานจริงจะทำให้ประหยัดในด้านอุปกรณ์ HARDWARE ทั้งยังออกแบบลายวงจรพิมพ์ได้ง่าย สามารถนำไปใช้งานจริงร่วมกับวงจรอื่นๆได้ง่ายกว่ามาก

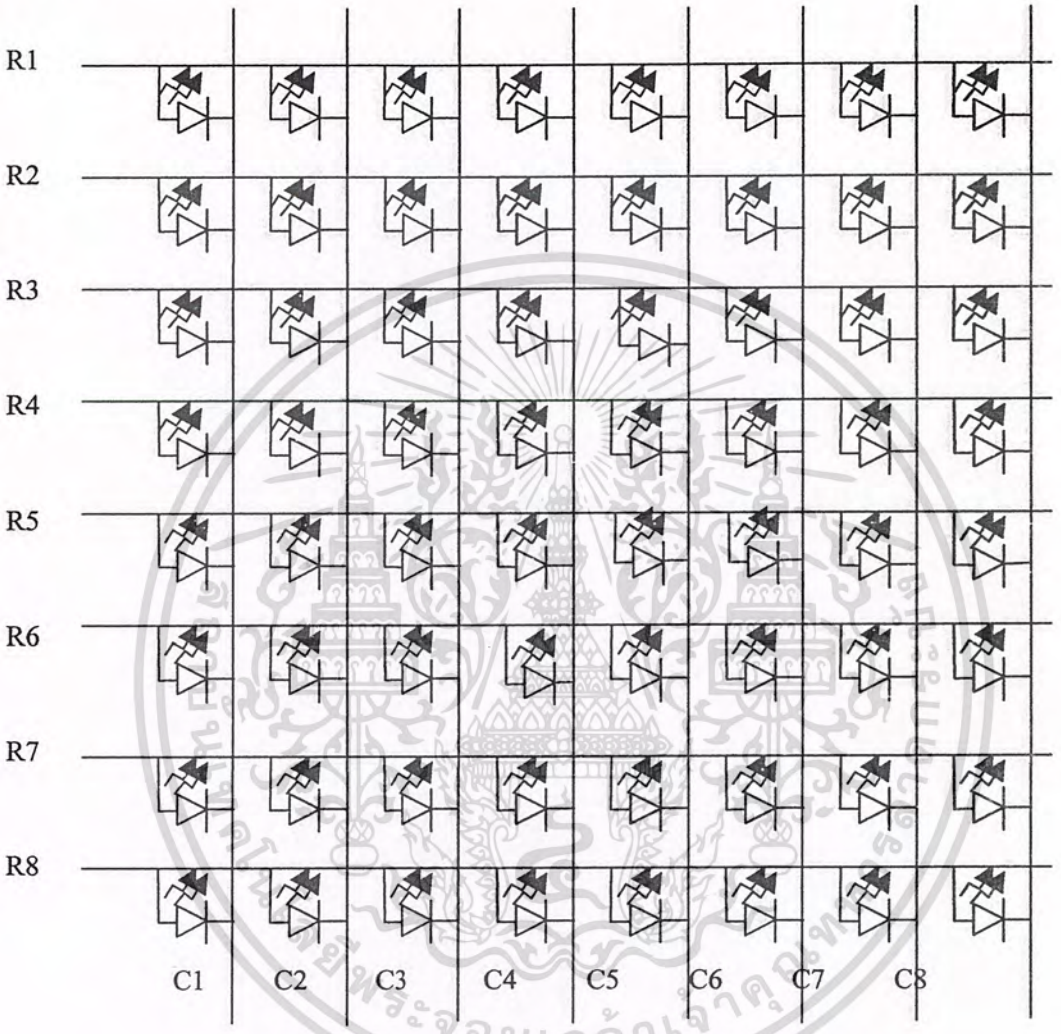
2.13 รายละเอียดของขา DB-9

ตารางที่ 2.15 แสดงรายละเอียดของขา DB-9

9 PIN	EIA RS-232 Circuit	RS-232 Description
5	Gnd	กราวน์ของสัญญาณ
2	RD	รับข้อมูล
3	TD	ส่งข้อมูล
1	CD	ตอบความพร้อมจากอุปกรณ์
4	DTR	แจ้งความพร้อมที่จะรับ-ส่งข้อมูล
6	DSR	พร้อมที่จะรับข้อมูลเป็นชุด
7	RTS	ต้องการจะส่งข้อมูล
8	CTS	Clear ก่อนส่งข้อมูล
9	RI	แสดงว่าอุปกรณ์ได้รับสัญญาณจากแหล่งอื่น

2.14 โครงสร้างของ Matrix LED

LED Matrix 1 ตัวประกอบด้วย LED 8x8 ดวง กล่าวคือ ลักษณะของ LED Matrix เป็นสี่เหลี่ยมจตุรัส มี LED ด้าน Column จำนวน 8 ดวง และมี LED ด้าน Row จำนวน 8 ดวง รวมทั้งสิ้น ใน 1 ตัว LED Matrix มี LED รวม คือ 8x8=64 ดวง

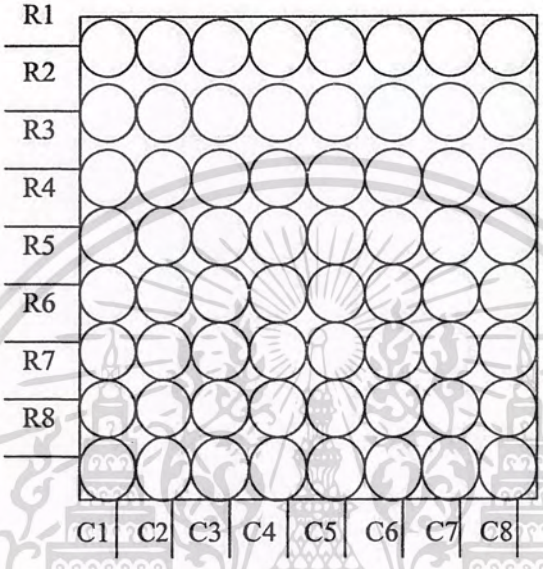


รูปที่ 2.25 แสดงการต่อภายใน Matrix LED 1 ตัว

จากรูปที่ 2.25 จะเห็นว่าทางด้าน Row นั้นต่อกับขา Arnode ของ LED นั่นคือ ทางด้าน Row ต้องการไฟบวก ส่วนทางด้าน Column นั้นต่อกับขา Cathod ของ LED นั่นคือ ทางด้าน Column ต้องการไฟลบ

2.15 การแสดงผล

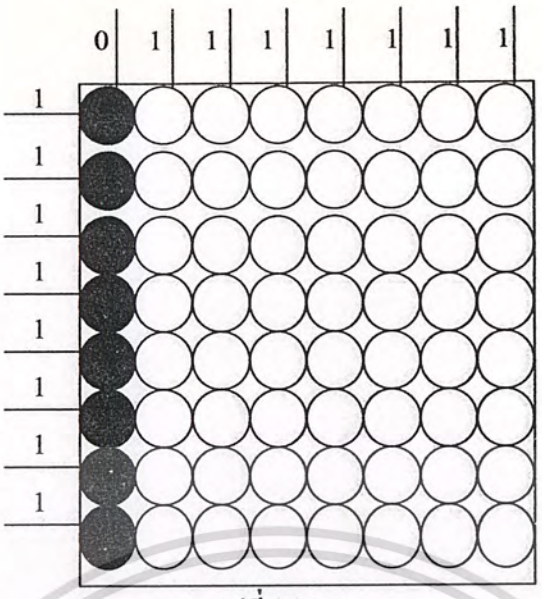
ส่วนแสดงผลของโครงงานนั้นได้นำ Matrix LED มาต่อเป็น Board ขนาดความยาวใช้ 8 ตัว Matrix LED ซึ่งเป็นด้าน Column ใน 1 ตัวมี 8 Column ดังนั้น Board แสดงผลทางด้านความยาวมีทั้งหมด 64 Column ส่วนทางด้านความกว้างใช้ 3 ตัว Matrix LED ซึ่งเป็นด้าน Row มีขนาดทั้งหมด 24 Row



รูปที่ 2.26 แสดง Matrix LED 8x8

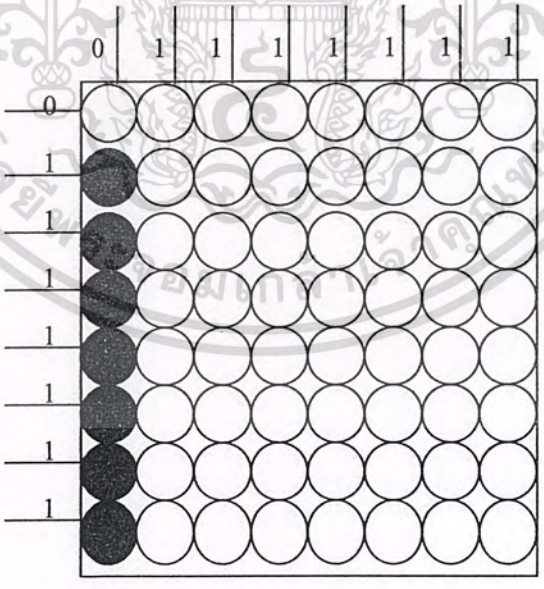
เมื่อต้องการให้ LED Matrix ติดก็ป้อนไฟบวกให้กับขา Row และป้อน ground ให้กับขา Column ซึ่งก็เหมือนกับการป้อน Logic "1" ให้ขา Row และป้อน Logic "0" ให้ขา Column จึงนำหลักการนี้มาทำการแสดงผลของ SCORE BOARD ซึ่งให้ Logic ทาง Row และ Column ควบคุมโดย CPU

ตัวอย่างที่ 2.26 เมื่อต้องการให้ LED Matrix Column ที่ 1 ติด ทั้ง Column ก็จะป้อน Logic ทาง Row เป็น Logic "1" ทุก Row และป้อน Logic "0" ให้ Column ที่ 1 ส่วนขา Column อื่นป้อน Logic "1" ก็จะทำให้ Matrix LED Column ที่ 1 ติดทั้ง Column ดังแสดงในรูปที่ 2.27



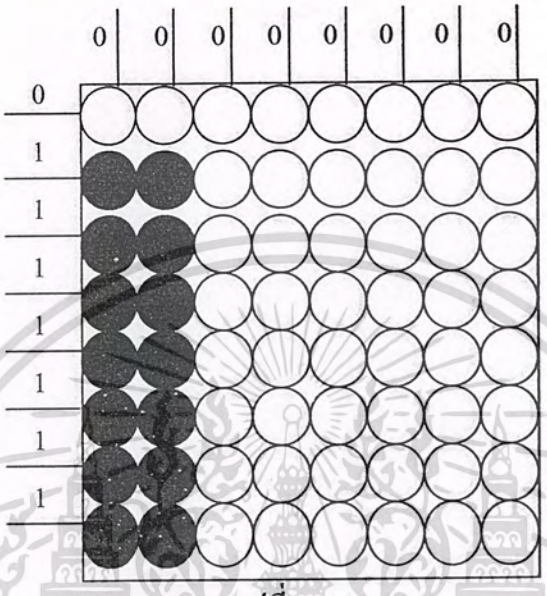
รูปที่ 2.27

จากรูปที่ 2.27 Logic "1" หมดที่ป้อนทาง Row นั้นถ้านำมาเขียนเป็นเลขฐาน 16 ก็จะได้เป็น "FFH" ซึ่งเลขฐาน 16 อันนี้เรียกว่า "Character Generator" ฉะนั้นอยากจะให้ Matrix LED Row ไหนติดหรือดับก็กำหนดได้โดย Character Generator ที่ป้อนทาง Row นี้เอง จากตัวอย่างนี้ถ้าอยากให้ Row ที่ 1 ดับ เราป้อน Character gen เป็น "7FH" ก็จะได้ Matrix LED Row ที่ 1 ไม่ติด ดังรูปที่ 2.28



รูปที่ 2.28

ส่วนทางด้าน Column นั้นเมื่อต้องการให้ Column ไหนติดก็ให้ Logic “0” ที่ Column นั้น จากตัวอย่างที่ 2.25 หากต้องการให้ Column ติดอีก Column ก็ให้ Logic “0” ที่ Column ที่ 2 อีก Column หนึ่ง Column ที่ 2 ก็จะติดดังแสดงในรูปที่ 2.29



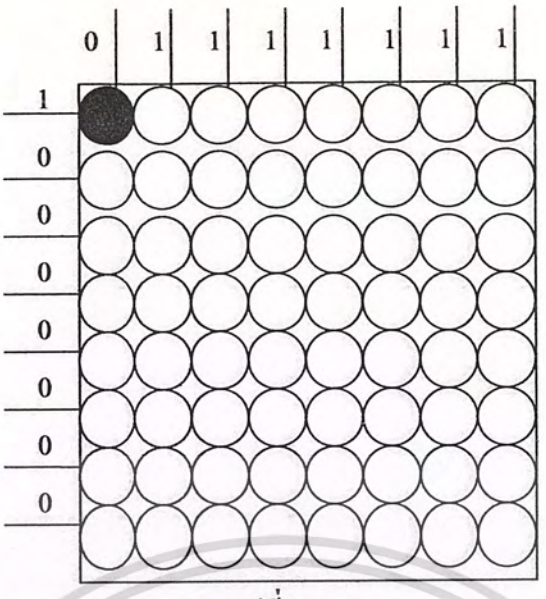
รูปที่ 2.29

จากรูป 2.29 หากว่าจะให้ Column ที่ 1 และ 2 แสดงผลไม่เหมือนกันจะทำไม่ได้ เนื่องจากเราป้อน Character Gen ทาง Row เป็น “7FH” เหมือนกันทั้ง 2 Column

ฉะนั้นจะเห็นว่าเราสามารถ Control การติดของ Matrix LED ได้เพียงทีละ Column เท่านั้น หากจะให้ Column ที่ 1 กับ Column แสดงผลไม่เหมือนกัน ก็ทำได้โดยการป้อน Character Gen ทีละตัวทาง Row แล้วให้ Logic”0” ทางต้น Column ทีละ Column ดังตัวอย่าง

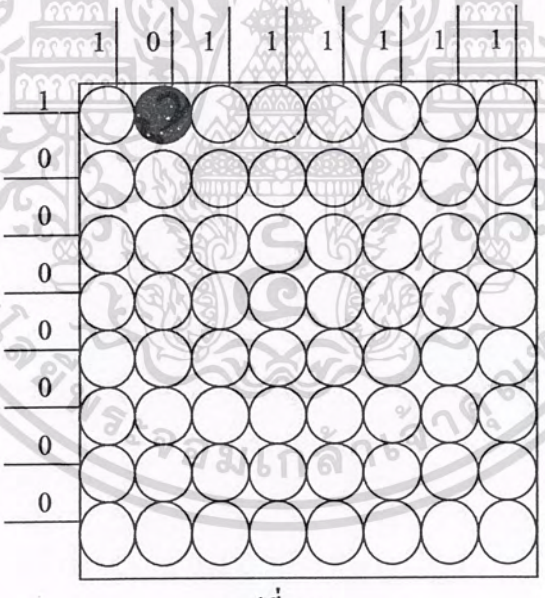
ตัวอย่างที่ 2.26 ให้ Matrix LED เป็นตัวอักษร “T” ก็จะต้องป้อน Character Gen ทาง Row และ Logic “0” ทางด้าน Column เป็นลำดับดังนี้

1. ป้อน Character Gen “01” ทาง Row และให้ Logic “0” Column ที่ 1 Matrix LED ก็จะแสดงผลดังแสดงในรูปที่ 2.30 ซึ่งจะติดกระพริบ แล้วดับไปเลย



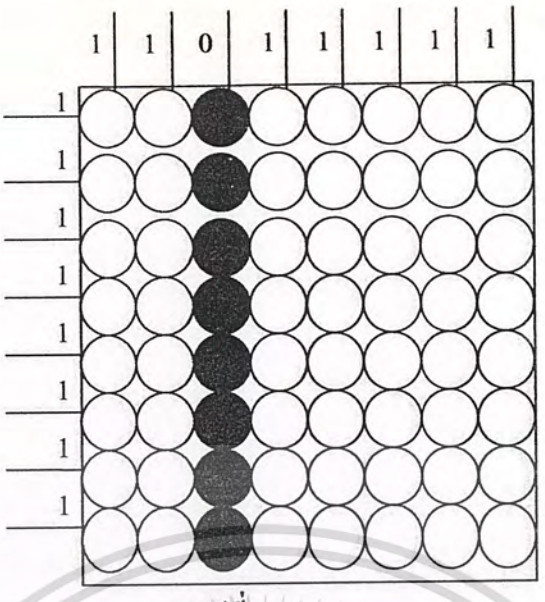
รูปที่ 2.30

2. ป้อน Character Gen “01” ทาง Row และป้อน Logic “0” ที่ Column ที่ 2 Matrix LED ก็จะแสดงผลดังรูปที่ 2.31



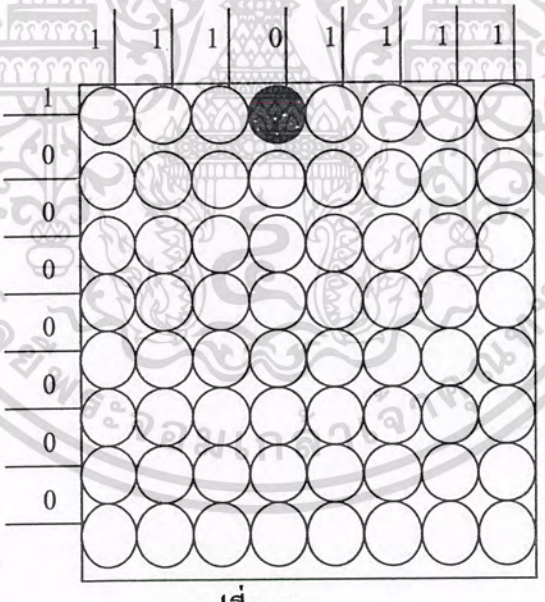
รูปที่ 2.31

3. ป้อน Character Gen “FFH” ทาง Row และป้อน Logic “0” ที่ Column ที่ 3 Matrix LED ก็จะแสดงผลดังรูปที่ 2.32



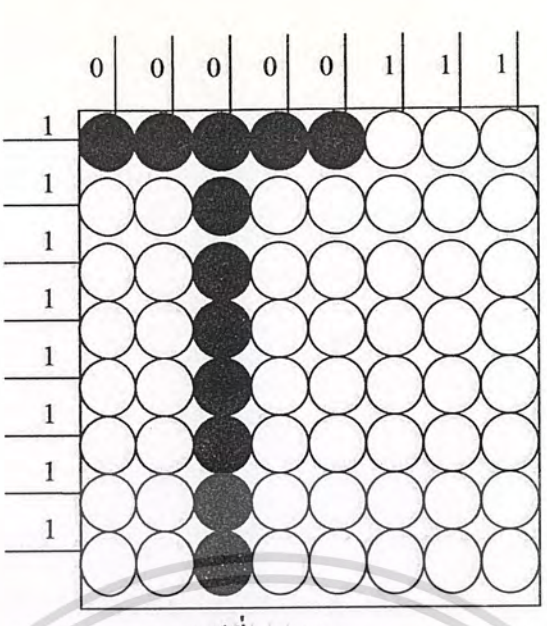
รูปที่ 2.32

4. ป้อน Character Gen "01" ทาง Row และป้อน Logic "0" Column ที่ 4 และ 5 ตามลำดับทีละ Column ก็จะได้ Matrix LED แสดงผลดังรูปที่ 2.33



รูปที่ 2.33

จะเห็นว่า Matrix LED จะติดที่ละ Column เท่านั้น แล้วจะเป็นตัว "T" ได้อย่างไร แต่จะเป็นไปได้เมื่อเราทำขั้นตอนทั้ง 4 ติดต่อกันอย่างพอเหมาะแน่นอน Matrix LED จะยังคงติดทีละ Column เหมือนเดิม ความเร็วในการแสดงผลโดยให้ CPU ควบคุมนั้นเร็วมาก จนตามนุษย์ไม่สามารถจับได้ว่า Matrix ติดทีละ Column แต่ตามนุษย์จะมองเห็นเป็นรูปตัว "T" ดังแสดงในรูปที่ 2.34



รูปที่ 2.34

แต่ก็จะติดอยู่ครู่เดียวก็ดับอีก เพราะเราป้อน Character Gen ตามขั้นตอน 1-4 เพียงเที่ยวเดียว ดังนั้นหากจะให้ติดเป็นรูปตัว "T" อยู่นานจนพอใจก็เพียงแต่ป้อน Character Gen และ Logic ทาง Row และ Column ตามลำดับ 1-4 วนหลายๆ รอบ ก็จะได้รูปตัว "T" ติดนานจนพอใจ

เพราะฉะนั้นจะเก็บ Character Gen ตัวอักษรต่างๆ ไว้ใน Memory เพื่อจะให้แสดงผลได้หลายรอบ ถ้าเป็น Matrix LED ขนาด 24 x 64 ก็จำเป็นต้องมี Character Gen 1ชุด ชุดละ 16 Row ส่วนทางด้าน Column ก็จะต้อง Rotate Logic "0" ไปทีละ Column เรื่อยๆ ทั้ง 64 Column

2.16 รูปแบบการแสดงผล

2.16.1 ตัวอักษรวิ่งไปหน้าหรือถอยหลัง

เมื่อเราสามารถทำให้ Matrix LED แสดงผลเป็นตัวอักษรอยู่กับที่แล้ว และเก็บ Character Gen ไว้ในหน่วยความจำแล้ว การที่จำทำให้ Matrix Led แสดงผลเป็นตัวอักษรวิ่งไปหน้าหรือถอยหลังได้โดยไม่ยาก ในที่นี้จะกล่าวถึงเพียงการวิ่งไปหน้าเพราะการวิ่งถอยหลังก็อาศัยหลักการเดียวกัน

การให้ตัวอักษรวิ่งไปหน้าทำได้จากการที่ Matrix LED แสดงผลเป็นตัวอักษรอยู่กับที่ ได้นานก็เพราะว่าเราป้อน Character Gen วนขบวนการเดิมหลายครั้ง หากว่าจะให้มันวิ่งเราก็เพียงแต่ให้มันวนอยู่ n ครั้งแล้วเลื่อน Character Gen ตัวที่ 2 มาแทนชุดที่ 1 และชุดที่ 3 มาแทน ชุดที่ 2 shift ขึ้นมาเรื่อยๆ

2.16.2 ตัวอักษรเลื่อนขึ้นหรือเลื่อนลง

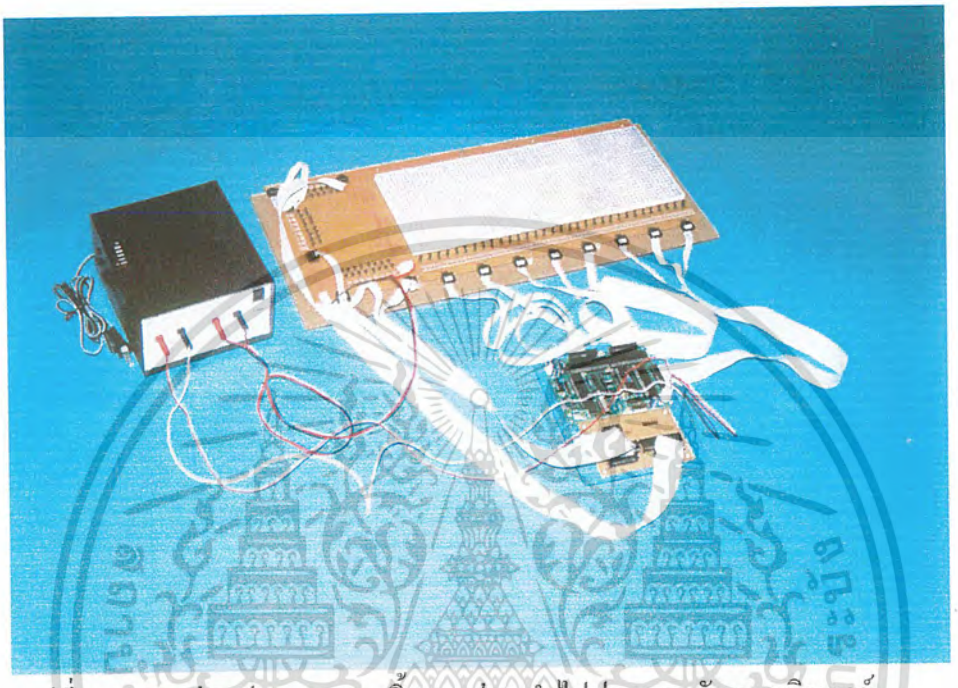
ในตอนการให้ตัวอักษรวิ่งไปหน้าหรือถอยหลัง เราป้อน Character Gen ทาง Row และป้อน Logic "0" ทาง Column เป็น ground ในการที่จะให้ Matrix LED เลื่อนขึ้น หรือลง นี้เราเปลี่ยน Character Gen ไปป้อนทาง Column และป้อน Logic "1" เป็นไฟ ทาง Row โดยที่ Character Gen เดิมนั้น ต้องทำให้เป็น 1's Complement เสียก่อน เนื่องจากทาง Column เป็นขาลบ (Cathode) ของ LED หากให้ดวงไหนติดก็ต้องป้อน Logic "0"

ต่อจากนั้นก็ใช้วิธีการป้อน Character Gen เหมือนกันกับในกรณีการให้ตัวอักษรเลื่อนไปหน้าหรือถอยหลัง ก็จะได้รูปตัวอักษรวิ่งขึ้นหรือลงได้ตามต้องการ

บทที่ 3

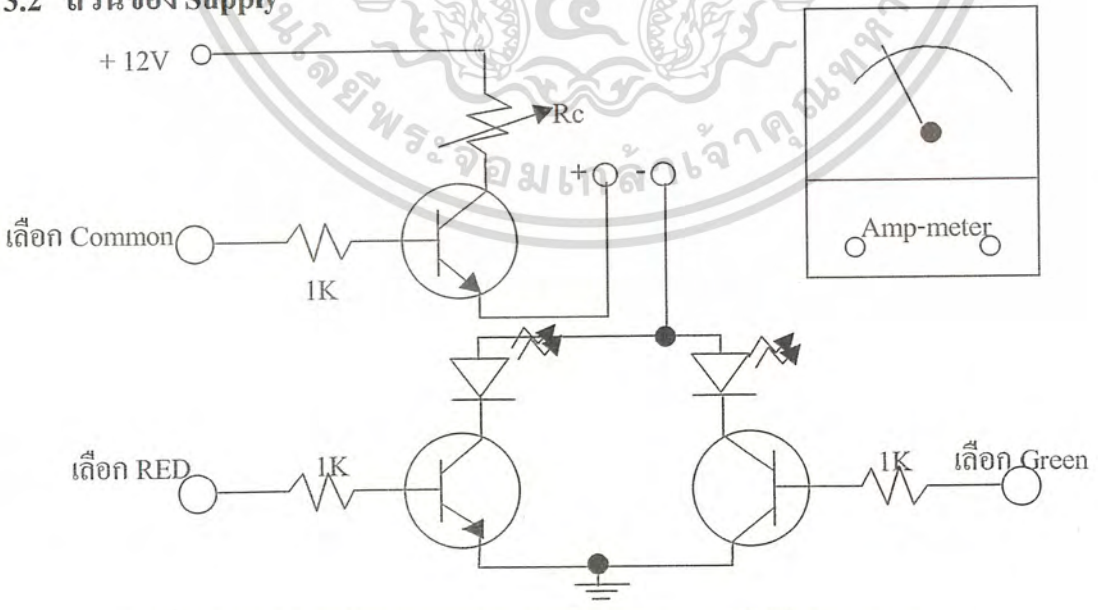
การทดลองและข้อกำหนดการใช้งานของวงจร

3.1 ลักษณะและรูปร่างของชิ้นงาน



รูปที่ 3.1 แสดงส่วนประกอบของชิ้นงานก่อนนำไปประกอบกับคอมพิวเตอร์

3.2 ส่วนของ Supply



รูปที่ 3.2 แสดงการต่อวงจรของ LED ของ Matrix LED ที่ใช้สร้างแผง Display

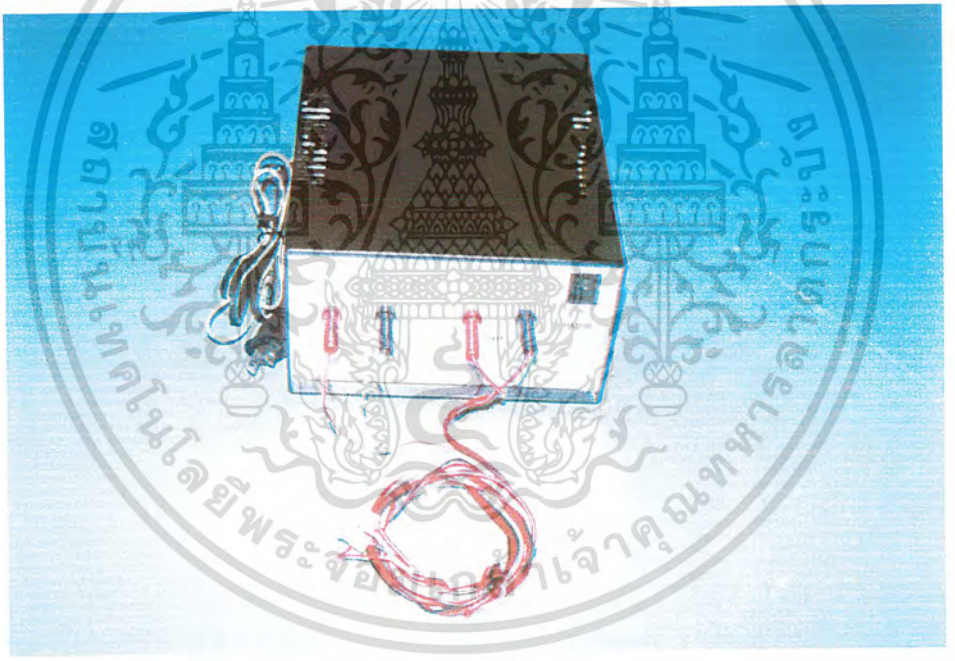
ตารางที่ 3.1 แสดงการทดลองหาค่า Rc ของส่วน Display

แรงดัน	Rc=1KΩ	Rc=2KΩ	Rc=3KΩ	Rc=4KΩ	Rc=5KΩ
9V	7mA	3.5mA	2.5mA	1.7mA	1.4mA
12V	9.5mA	5mA	3.3mA	2.4mA	2mA

จากการทดลองจะได้ว่าภายใน Matrix LED 1ตัวจะมี LED =8x8=64 ดวง แต่ละดวงกินกระแส = 1.4 mA Display นี้ใช้ Matrix LED ขนาด 3x8 จะมีจำนวน LED = 24x64=1536 ดวง ถ้า LED ทั้งหมดติดพร้อมกันจะกินกระแสทั้งหมด = 1536x2mA = 2.15 A

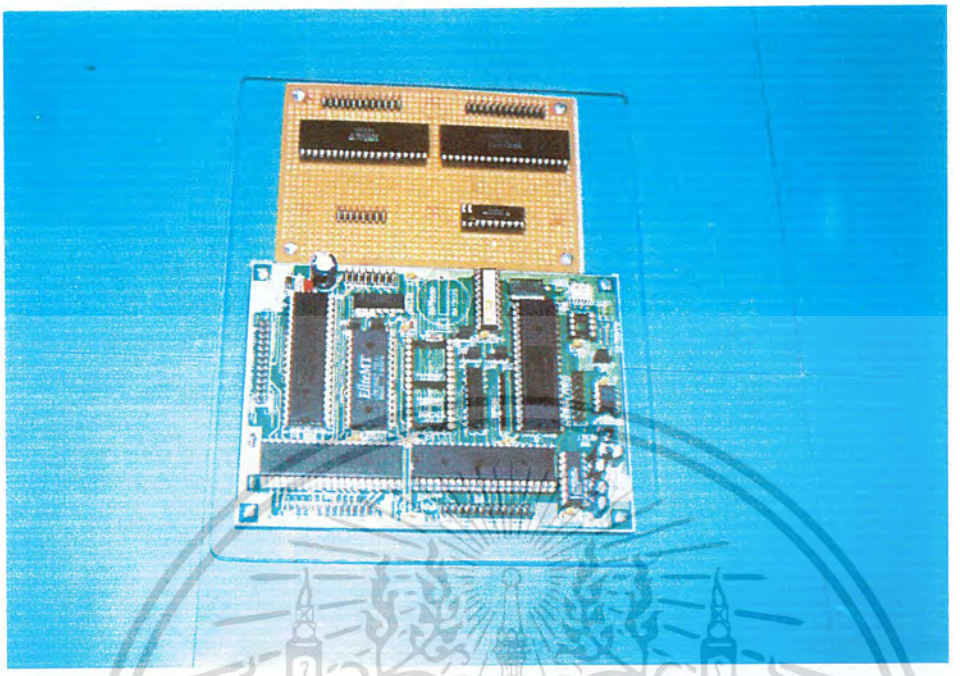
Power Supply จะมี 2 ขนาดคือ 9V 2.15 A สำหรับDisplay และ 5V 0.5A สำหรับเลี้ยงบอร์ดไมโครคอนโทรลเลอร์ MCS-51

ดังนั้นต้องสร้าง Power Supply จ่ายไฟ 9V และ 5V กระแส = 3A



รูปที่ 3.3 Power Supply

3.3 ชุดไมโครคอนโทรลเลอร์ MCS-51



รูปที่ 3.4 แสดงชุดไมโครคอนโทรลเลอร์ MCS-51 และการต่อ RAM เพิ่ม

3.4 แผงแสดงผล

รูปที่ 3.5 แสดงแผงแสดงผล (Display Board)

3.5 หน้าที่และการทำงานของส่วนประกอบต่างๆ

มีด้วยกัน 3 ส่วนประกอบหลักๆ ดังนี้

3.5.1 คอมพิวเตอร์

คอมพิวเตอร์ทำหน้าที่รับคำสั่งจาก Key Borad ผ่านทางทางจอของ Program visual basic แล้วส่ง Data คำสั่ง ไปควบคุม ชุดไมโครคอนโทรลเลอร์ MCS-51 ผ่านทาง Serial Port โดยใช้มาตรฐานการส่ง RS-232

3.5.2 ชุดไมโครคอนโทรลเลอร์

รับคำสั่งจากคอมพิวเตอร์ แล้วส่ง Data ไปแสดงผลทางจอ Display ตามคำสั่งที่รับเข้ามาจากผู้ใช้ คำสั่งที่รับเข้ามาชุดไมโครคอนโทรลเลอร์จะรักษาสภาพการทำงานตามคำสั่งนี้ไปเรื่อยๆ จนกว่าจะมีการแก้ไขคำสั่งการแสดงผลผ่านทางคอมพิวเตอร์ใหม่

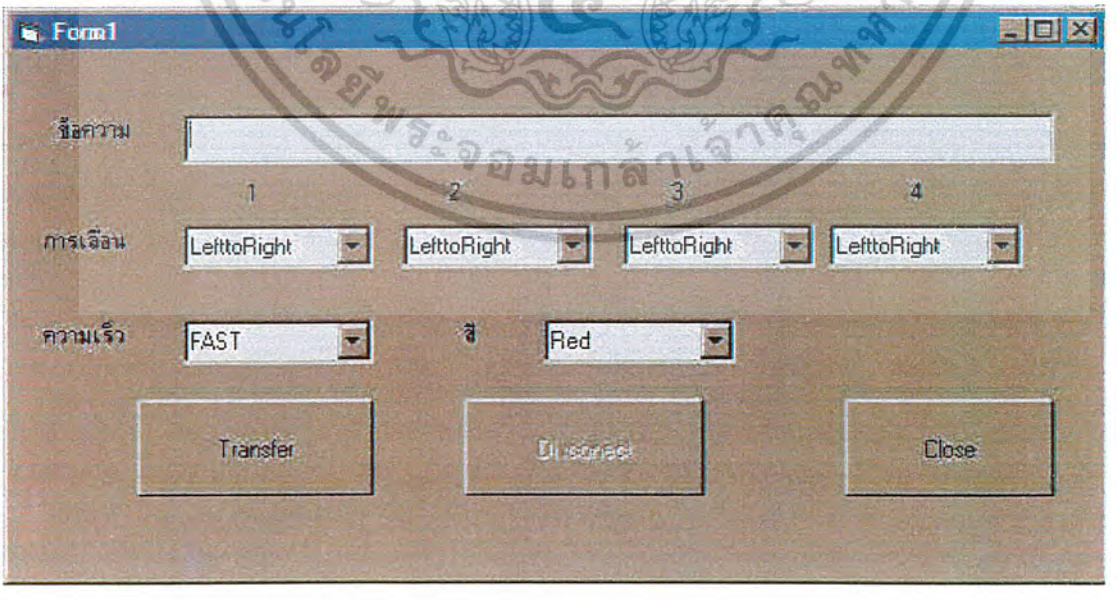
3.5.3 ชุดแผงแสดง LED

แผงแสดงผลขนาด 48x 64 ดวง ทำหน้าที่ แสดงผลตาม Data ที่ชุดไมโครคอนโทรลเลอร์สั่งงานมา

3.6 การใช้งาน

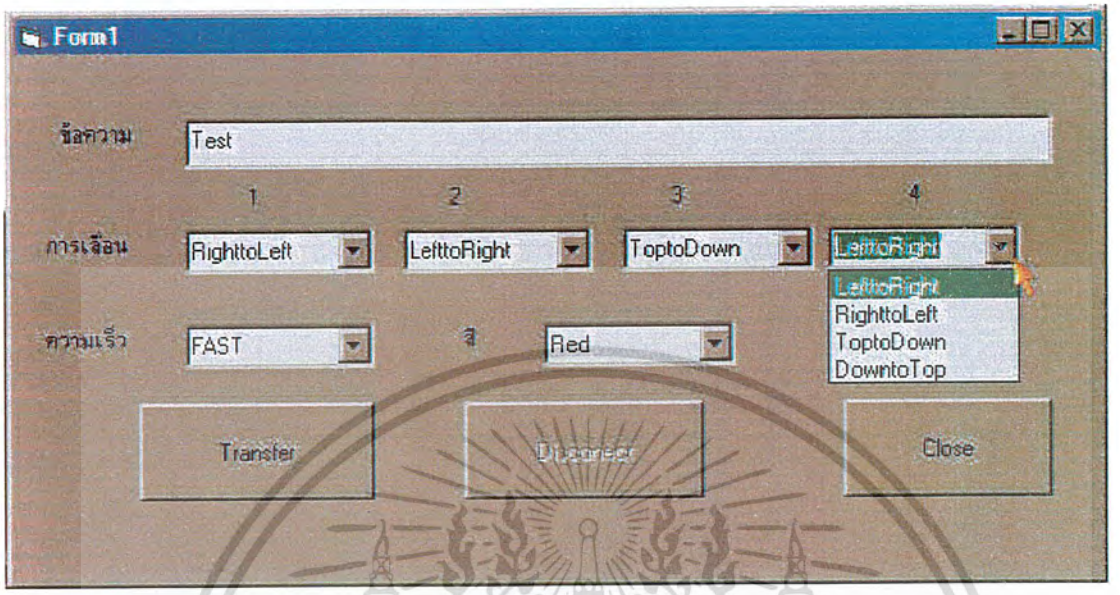
คู่มือการใช้งาน

1. ทำการใส่ข้อความลงในช่องข้อความ



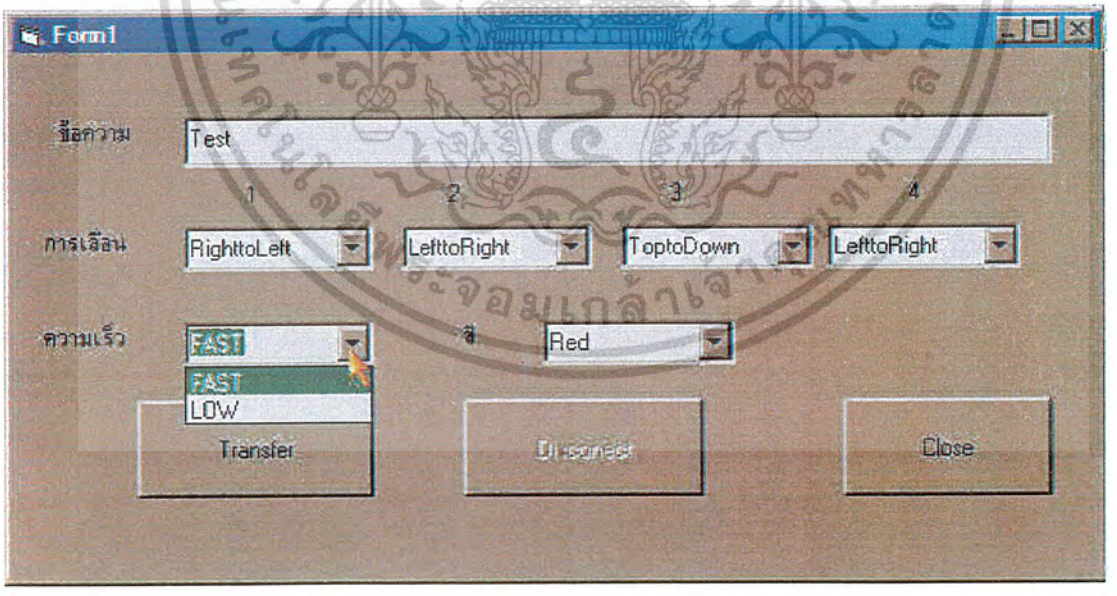
รูปที่ 3.6 แสดงหน้าต่างสำหรับการเปลี่ยนแปลงการแสดงผล

2. ทำการเลือกรูปแบบการเลื่อนของตัวอักษรทั้ง 4 ช่อง โดยการแสดงผลนั้นจะเรียงตามลำดับตัวเลขที่แสดงอยู่เหนือช่อง



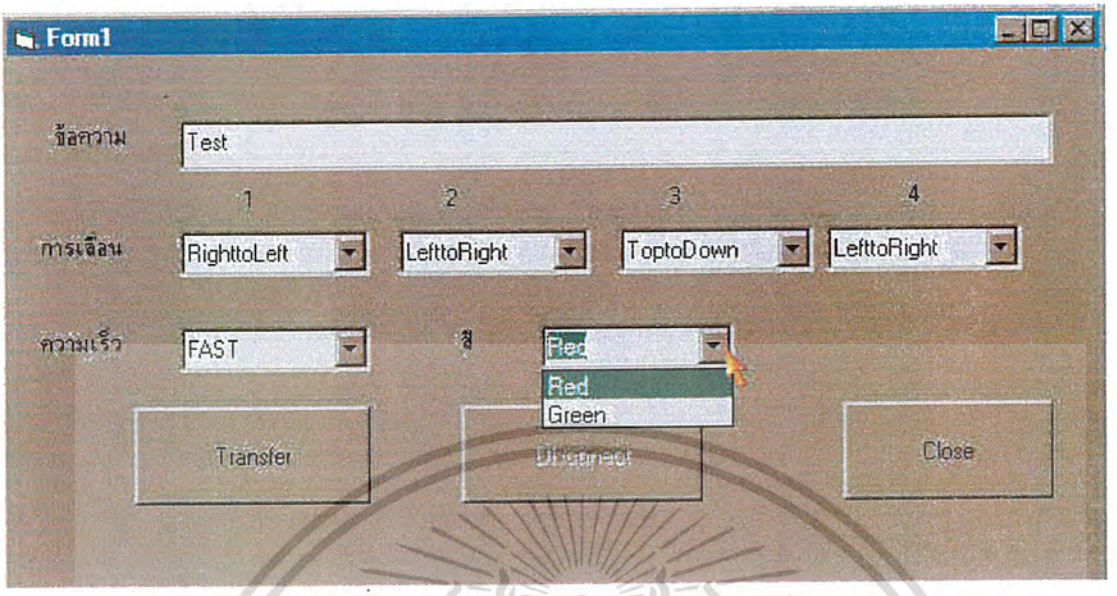
รูปที่ 3.7 แสดงหน้าต่างในการเลือกรูปแบบการเลื่อนของตัวอักษร

3. ทำการเลือกความเร็วในการเลื่อนของตัวอักษร



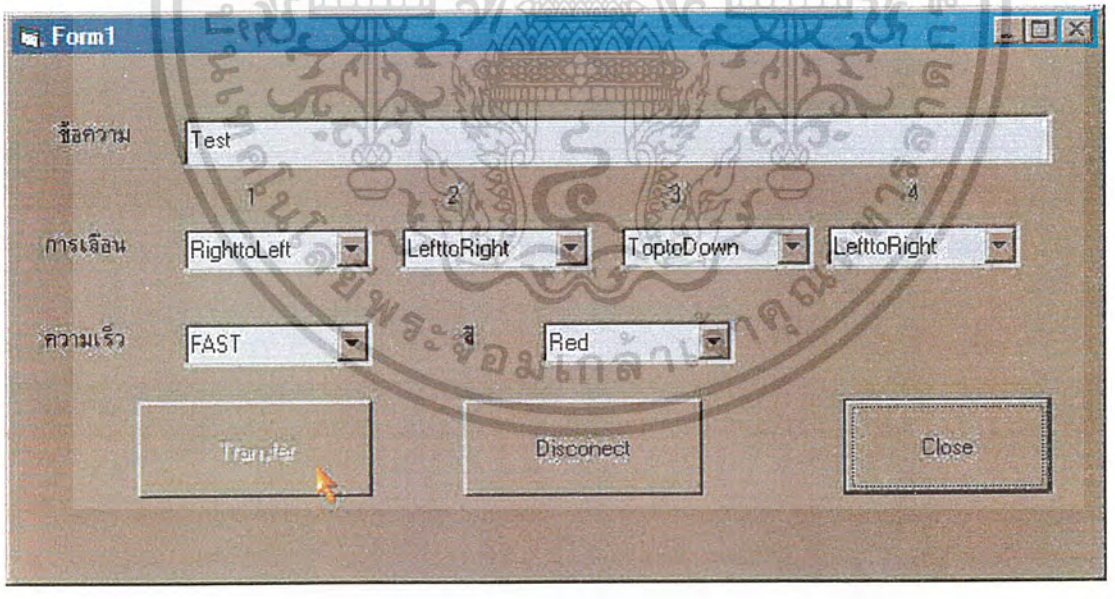
รูปที่ 3.8 แสดงหน้าต่างในการเลือกความเร็ว

4. ทำการเลือกสีของตัวอักษร



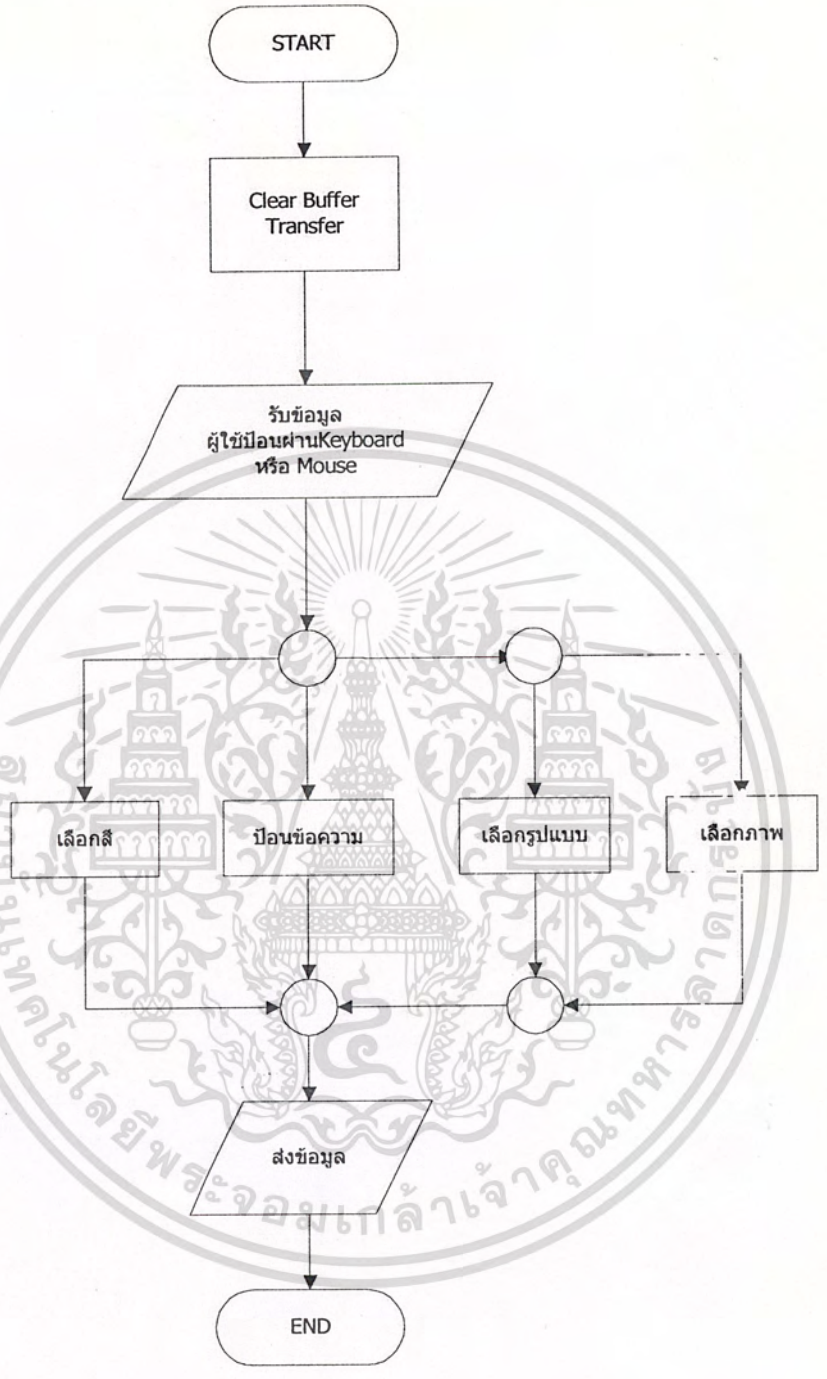
รูปที่ 3.9 แสดงหน้าต่างในการเลือกสี

5. ทำการกดปุ่ม Transfer ในการส่งข้อมูลไปยังแผงสดงผล

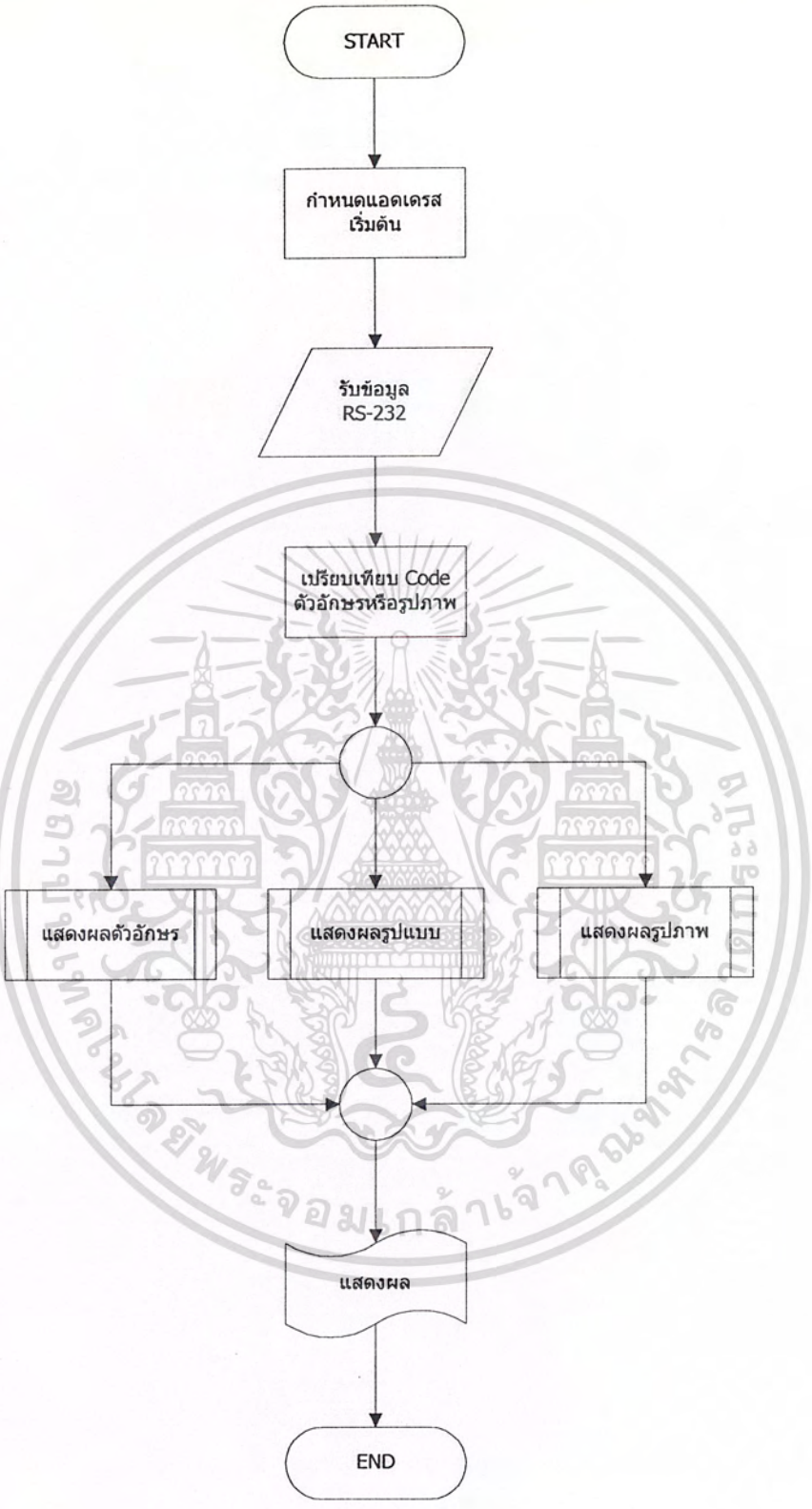


รูปที่ 3.10 แสดงหน้าต่างในการกดปุ่ม Transfer

3.7 Flow Chart



รูปที่ 3.11 flow chart แสดงการทำงานของโปรแกรม VISUAL BASIC



รูปที่ 3.12 flow chart แสดงการทำงานของโปรแกรมหลักของ MCS-51

บทที่ 4

ผลการทดลอง

จากการทดลองโดยที่เราทำการใส่ข้อความที่ต้องการแล้วเลือกการแสดงผลแบบต่างๆให้ผลดังนี้

- ในการเลือกการเลื่อนอักษรนั้น การเลือกจะเป็นไปตามที่เราเลือก โดยจะแสดงการเลื่อนตัวอักษรจากแบบที่ 1 – 4 จนครบ แล้วจะมาทำการเลื่อนแบบที่ 1 ใหม่อีกครั้ง
- ในการเลือกความเร็วในการเลื่อนตัวอักษรนั้น ถ้าเลือก FAST การเลื่อนของตัวอักษรจะเร็ว แต่ถ้าเลือก SLOW การเลื่อนของตัวอักษรจะช้า
- ในการเลือกสีของตัวอักษรนั้น ถ้าเลือก RED จะได้ตัวอักษรสีแดง ถ้าเลือก Green จะได้ตัวอักษรสีเขียว
- ในการเปลี่ยนค่าแต่ละครั้งต้องทำการกดปุ่ม Transfer ทุกครั้ง

บทที่ 5

สรุปผลการทดลอง ปัญหาและข้อเสนอแนะ

โครงงานนี้ ผู้จัดทำได้ออกแบบวงจรและทดลองการทำงานตามรายละเอียดที่กล่าวมาข้างต้นและสามารถทำงานได้ตามวัตถุประสงค์ของโครงงาน จากการทดลองพบว่ายังมีปัญหาเกิดขึ้น อาทิเช่น

- ปัญหาความสว่างของแผงแสดงผลไม่สว่างเท่าตอนที่ทำการทดลอง สาเหตุเนื่องจากเมื่อประกอบส่วนประกอบต่างๆ เข้าด้วยกัน มีการ Load ของอุปกรณ์แต่ละตัวมากเกินไป จึงต้องทำการเปลี่ยนแปลง Power Supply ให้มีแรงดันและกระแสเพิ่มขึ้นอีกระดับหนึ่ง
- ปัญหาการแสดงผลของข้อความไม่ค่อยสม่ำเสมอเท่าที่ควร สาเหตุอาจเกิดตัว Matrix LED แต่ละตัว มี Characteristic ที่ผิดเพี้ยนกันไปบ้าง ทำให้ในการเขียน โปรแกรม Delay ควบคุม จึงเกิดข้อผิดพลาด
- ปัญหาการจำกัดทางด้านงบประมาณ ทำให้ได้แผงแสดงผลที่มีขนาดไม่ใหญ่มาก ทำให้ความสมบูรณ์ของการแสดงผลไม่สมบูรณ์เท่าที่ควร

โครงงานนี้แม้จะทำงานได้ผล แต่ก็ยังไม่สมบูรณ์มากนักเพราะทางผู้จัดทำนั้น ได้เน้นในเรื่องการประหยัด ทำให้ได้แผงแสดงผลที่เล็กตัวหนังสือมีความไม่สมบูรณ์เท่าที่ควร สำหรับผู้ที่สนใจถ้าต้องการให้ตัวหนังสือมีความสมบูรณ์มากกว่านี้ ก็ควรเพิ่มขนาดของแผงแสดงผลให้มีขนาดใหญ่กว่านี้ แล้วแต่ความต้องการของผู้สนใจ

บรรณานุกรม

วันสุระ ศรีใสดี. 2542. **ประยุกต์/อินเตอร์เฟสไมโครคอนโทรลเลอร์ภาคปฏิบัติ.** กรุงเทพฯ. สำนักพิมพ์ดวงกมล จำกัด.

รองศาสตราจารย์ สมยศ จุณณะปิยะ. 2543. **การประยุกต์ใช้งานไมโครคอนโทรลเลอร์ MCS-51.** พิมพ์ครั้งที่ 3. กรุงเทพฯ. สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง.

เชิดชัย บัวพันธ์, สุกิจ อุตะปะละ และอนันต์ สิรันทวิเนติ. 2534. **แผงไฟแสดงผล.** อุตสาหกรรมศาสตร์บัณฑิต สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

Kenneth J. Ayala. **The 8051 Microcontroller Architecture, Programming and Applications.** Vol.1

PHILIPS. **Signatics Microcontroller User's Guide.** 1989

MCS-51 Microcontroller. Vol. I, Inter Coporation. 1988