ชุดพัฒนาไมโครคอนโทรลเลอร์ 80C535

80C535 Microcontroller Develop Kit

นายอนุพงศ์ พานิชสรรพ์ 34161238

นายอภิชาติ สืบปรุ 34161240

อาจารย์ที่ปรึกษา

อ.วิทยา ทิพย์สุวรรณนร

ปริญญานิพนธ์สำหรับปริญญาอุตสาหกรรมศาสตร์บัณฑิต

สาขาเทคโนโลยีการวัดคุมทางอุตสาหกรรม

ภาควิชาเทคโนโลยีการวัดคุมทางอุตสาหกรรม

คณะวิศวกรรมศาสตร์

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

ปีการศึกษา 2535

ภาควิชา เทคโนโลยีการวัดคุมทางอุตสาหกรรม

สาขา เทคโนโลยีการวัดคุมทางอุตสาหกรรม

คณะวิศวกรรมศาสตร์ สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

เรื่อง ชุดพัฒนาไมโครคอนโทรลเลอร์ 80C535

80C535 Microcontroller Develop Kit

ผู้จัดทำ

1. นายอนุพงศ์ พานิชสรรพ์ 34161238
2. นายอภิชาติ สิบปรุ 34161240

.....อาจารย์ที่ปรึกษา

(อ.วิทยา ทิพย์สุวรรณพร)

บทคัดย่อ

โครงงานนี้แบ่งเป็น 2 ส่วนคือ

1. ฮาร์ดแวร์ โดยการสร้างการ์ดขึ้นมาเพื่อเสียบบนสล็อตของเครื่อง ไอบีเอ็ม XT/AT ให้มีการติดต่อกับผู้ใช้โดยที่เครื่องไอบีเอ็ม จะเป็นส่วนจัดการเกี่ยวกับการรับส่งข้อมูลผ่านทางคีย์บอร์ดและจอภาพ โดยที่คีย์บอร์ดจะทำหน้าที่เป็นส่วนอินพุท และจอภาพจะทำหน้าที่เป็นส่วนเอาท์พุท การ์ดที่สร้างขึ้นนั้นจะมีหน่วยความจำสแตติคแรมขนาด 32 กิโลไบท์ 8 บิท จำนวน 2 ตัว เพื่อที่จะใช้เป็นหน่วนความจำของ SAB 80C535 ซึ่งเป็นไมโครคอนโทรเลอร์ขนาด 8 บิท ข้อมูลและ 16 บิท แอสเดรส และยังต้องมีส่วนมัลติเพลกซ์ส่วนควบคุมการทำงานของแอสเดรสบัส ส่วนการทำอินเตอร์เฟส ส่วนควบคุมบัสข้อมูลและบัฟเฟอร์ โดยการนำวงจรแต่ละส่วนที่ได้ออกแบบมาต่อร่วมเข้าด้วยกันเป็นระบบขึ้นมา

2. ซอฟท์แวร์ โดยต้องเขียนซอฟท์แวร์ขึ้นเพื่อสนับสนุนการใช้งานของการ์ด ซึ่งมีโปรแกรมโหลดเดอร์ โปรแกรมดีบักเกอร์ ส่วนโปรแกรมแอสเซมเบลอร์นั้นจะใช้โปรแกรมที่สนับสนุนชีพียูตระกูล 8051 ที่มีอยู่ในท้องตลาดตัวไหนก็ได้ โดยที่โปรแกรมแอสเซมเบลอร์มีหน้าที่ในการแปลภาษาแอสเซมบลีของ 80C835 ให้เป็นภาษาเครื่อง โปรแกรมโหลด โหลดเดอร์ใช้สำหรับทำการโหลดข้อมูลในหน่วยความจำของเครื่องไอบีเอ็ม ไปไว้ในหน่วนความจำบนการ์ดของ 80C535 และโปรแกรมดีบักเกอร์เป็นโปรแกรมที่ช่วยอำนวยความสะดวก ในการพัฒนาโปรแกรมภาษาแอสเซมบลี ให้มีความง่ายและสะดวกสะบายในการแก้ไขและตรวจสอบ

ไมโครคอนโทรลเลอร์ 80C535 ก็เป็นหนึ่งในจำนวนไมโครคอนโทรลเลอร์ ที่มีความสามารถและคำสั่งพื้นฐานเหมือนกับ ไมโครคอนโทรลเลอร์ตระกูล MCS-51 แต่จะมีความสามารถพิเศษที่เพิ่มขึ้นมามากมาย เช่นมีตัวแปลงสัญญาณอนาลอกเป็นดิจิตอล และมีตัวป้องกันไม่ให้ไมโครคอนโทรลเลอร์หยุดทำงานรวมถึงพอร์ทสองทางถึงหกชุด และสิ่งที่เพิ่มเติมนอกจากนี้จะขอกล่าวไว้ในปริญญานิพนธ์นี้และ ได้นำความสามารถเหล่า นี้มาทำเป็นชุดพัฒนาบนไมโครคอมพิวเตอร์ ในรูปแบบของไมโครคอนโทรลเลอร์ที่สามารถนำไปใช้ในงานควบคุม หรือจะใช้ทำเป็นชุดพัฒนาทางซอฟแวร์และฮาร์ดแวร์โดย มีทั้งซอฟแวร์ดีบักเกอร์และฮาร์ดแวร์ดีบักเกอร์ เพื่อไว้ใช้ในการพัฒนาซอฟแวร์ในงานควบคุม ซึ่งตัวไมโครคอนโทรลเลอร์นี้นั้นสามารถจะทำงานอยู่บนตัวไมโครคอมพิวเตอร์ หรือจะทำงานโดยอิสระด้วยตัวของมันเองโดยผ่านทาง RS-232 หรือจะใช้ EPROM ในการเก็บโปรแกรมที่ได้พัฒนาไว้แล้วและต้องใช้แหล่งจ่ายไฟภายนอก

ABSTRACT

This Project is have two parts as

1. HARDWARE

By build The 80535 Development card for working on IBM.PC slot or stand alone working. The user can communicate with this card by key board and the monitor of IBM. PC or compatable. The card's memory is have two 32 Kbyte 8 Bit static ram and have the multiplex and demultiplex of data and address parts.

2. SOFTWARE

By creat program for manage the card memory.For software it have the loader program, debugger program.The assembler procedure cad use any 8051 family utility program for convert the assembly program to machine code.

The loader program is use for down load from IBM.PC to the card memory and the debugger program is tha utility of the assembly development program.

The 80C535 CPU. Is the microcontroller that have basic command as the MCS-51 family but it hame more than function by include analog to digital converter (ADC) function, watchdog timmer, and 6 user ports.

The purpose of this thesis is for build and use 80535 card for use in the control processing or use for softward development. For this card able to run on IBM PC slot or stand alone working bt use RS-232 communication.

สารบัญ

หน้า

บทคัดย่อ

Abstract

บทนำ

วัตถุประสงค์ของปริญญานิพนธ์

ขอบเขตของงานวิจัย

บทที่ 1 ความรู้ทั่วไปเกี่ยวกับไมโครคอนโทรลเลอร์เบอร์ 80C535.....1

1.1 คุณสมบัติของไมโครคอนโทรลเลอร์.....1

1.2 การทำหนดขาสัญญาณและพอทต์ต่างๆ ของ 80C535.....5

1.3 การจัดหน่วยความจำภายในและภายนอก.....12

1.4 การอินเตอร์รัพต์.....15

บทที่ 2 การสร้างฮาร์แวร์ของ 80C535 ไมโครคอนโทรลเลอร์.....17

2.1 คุณสมบัติของ 80C535 ไมโครคอนโทรลเลอร์.....17

2.2 แนวทางการออกแบบ.....18

2.3 การสร้างระบบไมโครคอลโทรเลอร์.....18

2.4 การทำงานของวงจร.....24

2.5 การติดตั้ง JUMPER.....29

บทที่ 3 การออกแบบส่วนซอฟต์แวร์.....31

3.1 บทนำ.....31

3.2 ซอฟต์แวร์ดีบักเกอร์.....31

3.2.1 แนวทางการออกแบบ.....31

3.2.2 การทำงานของโปรแกรม.....46

3.2.3 คำสั่งการใช้งานดีบักเกอร์.....46

บทที่ 4 บทสรุป

3.1 สรุปผลการทดลอง.....49

3.2 ปัญหาที่เกิดขึ้น.....49

กิตติกรรมประกาศ.....51

หนังสืออ้างอิง.....52
ภาคผนวก ก ชุดคำสั่งของ SAB 80C535.....53
ภาคผนวก ข คุณลักษณะของ SAB 80C535.....86

บทนำ

ในปัจจุบันเทคโนโลยีสมัยใหม่ที่มีความสำคัญและมีอิทธิพลต่อความเป็นอยู่ในชีวิตประจำวันอย่างหลีกเลี่ยงไม่ได้ และมีแนวโน้มว่าจะเพิ่มความสำคัญยิ่งขึ้นไปอีกในอนาคต ทั้งงานทางด้านอุตสาหกรรม ด้านการแพทย์ตลอดจนถึงงานในบ้านที่ต้องการความสะดวกรวดเร็ว เพื่อให้สอดคล้องกับความเจริญก้าวหน้าในปัจจุบัน แต่ในทางกลับกันเป็นเรื่องที่มีความยุ่งยากมากในการที่จะศึกษาและทำความเข้าใจกับสิ่งประดิษฐ์ใหม่ๆ ที่เกิดขึ้นมาในท้องตลาดโดยเฉพาะเทคโนโลยีทางด้านไมโครโปรเซสเซอร์และ ไมโครคอนโทรลเลอร์ที่เกิดขึ้นมาใหม่ๆ ซึ่งการที่จะนำตัวมันและโปรแกรมของมันมาใช้งานได้จริงๆ นั้นเป็นสิ่งที่ต้องใช้เวลาในการศึกษา บางทีอาจจะมีการแก้ไขซ้ำแล้วซ้ำอีก โดยเฉพาะทางด้านการเขียนโปรแกรมอาจจะทำเป็นสิบๆ ครั้ง จึงได้มีหลายๆฝ่ายได้สร้างเป็นชุดพัฒนาเฉพาะสำหรับไมโครโปรเซสเซอร์ตัวนั้นเลย ซึ่งส่วนมากก็จะเป็นไมโครโปรเซสเซอร์เบอร์ที่ออกมาสู่ตลาดได้พอสมควรและได้รับความนิยม แต่สำหรับไมโครโปรเซสเซอร์ที่ออกมาใหม่ๆ จึงเป็นการยากที่จะศึกษา ซึ่งสำหรับในปริญญานิพนธ์นี้ขอเสนอการนำไมโครคอนโทรลเลอร์ใหม่ๆ มาทำเป็นชุดพัฒนาทั้งทางด้านซอฟแวร์ และฮาร์ดแวร์ โดยใช้ความสามารถของไมโครคอมพิวเตอร์มาช่วย ในการเพิ่มความสามารถและความคล่องตัวของชุดพัฒนา

วัตถุประสงค์ของปริญญานิพนธ์

ตามที่ได้กล่าวมาแล้วว่าในปริญญานิพนธ์นี้เป็นการศึกษาการทำงาน ของไมโครคอนโทรลเลอร์เบอร์ 80C535 ซึ่งพัฒนาบนไมโครคอมพิวเตอร์ เพื่อนำไปทำเป็นชุดพัฒนางานในด้านต่างๆ ได้หลายด้านและเป็นการศึกษาถึงรูปแบบทั่วๆ ไปของวงจรดิจิตอลและทางด้านไมโครโปรเซสเซอร์และ เป็นการศึกษาการติดต่อกับไมโครคอมพิวเตอร์กับไมโครโปรเซสเซอร์เบอร์อื่นๆ และกับอุปกรณ์ทั้งทางด้านอนาลอกและดิจิตอลและ ที่สำคัญนั้นคือได้ศึกษาโปรแกรมที่ช่วยในงานพัฒนางานทางด้านภาษาแอสแซมบลี้ตามรายการต่อไปนี้ คือ โปรแกรม Cross-32 Meta-Assembler ของ Universal Cross-Assemblrs ซึ่งเป็นโปรแกรมแปลภาษาแอสแซมบลี้ให้เป็นภาษาเครื่อง,8051SIM ของ Hitech Equipment Corporation เป็นโปรแกรมจำลองให้เครื่องไมโครคอมพิวเตอร์ให้มี ไมโครโปรเซสเซอร์เป็นเบอร์ 8051 และโปรแกรม AvCase ของ AVOCET SYSTEMS,INC. เป็นโปรแกรมพัฒนาภาษาแอสแซมบลี้จากภาษา ซี ซึ่งคิดว่าจะเป็นประสพการณ์และเป็นประโยชน์แก่งานในอนาคตข้างหน้าต่อไป

ขอบเขตของงานวิจัย

ในส่วนหลักของปริญญานิพนธ์จะเป็นการทำไมโครคอนโทรลเลอร์ บนเครื่องคอมพิวเตอร์ซึ่งเป็นวงจรที่ประกอบไปด้วยอุปกรณ์ที่มีประสิทธิภาพสูง โดยใช้เครื่องไอบีเอ็มเป็นตัวจัดการทางด้านอินพุท/เอ้าท์พุท ดังมีรายละเอียดดังต่อไปนี้

ในส่วนของฮาร์ดแวร์ จะเป็นไมโครคอนโทรลเลอร์บอร์ดที่ต้องทำงานโดยเสียบอยู่บนสล๊อลของเครื่องคอมพิวเตอร์ หรือจะทำงานอยู่นอกเครื่องคอมพิวเตอร์โดยผ่านทาง RS-232 และใช้แหล่งจ่ายไฟภายนอก ในส่วนของวงจรนั้นจะประกอบไปด้วยหน่วยความจำ (RAM) พื้นฐานที่ 8 Kbyte แต่สามารถขยายได้สูงสุด 64 Kbyte และใช้พอร์ทที่จะนำไปใช้งานควบคุมจะใช้คอนเน็คเตอร์ขนาด 37 ขา

ในส่วนของซอฟแวร์ จะประกอบด้วยการสร้างโปรแกรมดีบักเกอร์ทางด้านซอฟแวร์และทางฮาร์ดแวร์ เพื่อช่วยในการเขียนโปรแกรมภาษาแอสเซมบลี้ ให้ง่ายและสะดวกรวดเร็วขึ้น โดยจะมีความผิดพลาดน้อยลง และโปรแกรมโหลดเดอร์ที่จะทำหน้าที่ในการจัดการนำข้อมูลในหน่วยความจำ บนเครื่องไอบีเอ็มลงสู่หน่วนความจำบนการ์ดของซีพียู 80C535 เพื่อเป็นโปรแกรมระบบ

ประโยชน์ที่คาดว่าจะได้รับ

โครงงานนี้ สามารถที่จะนำไปใช้เป็นเครื่องมือในการพัฒนาระบบที่ใช้ไมโครโปรเซสเซอร์ และไมโครคอลโทรเลอร์ในตระกูล 8051 โดย เฉพาะไมโครคอลโทรเลอร์เบอร์ 80C535 โดยถือว่าเป็นจุดเริ่มต้นในการพัฒนา และเป็นแนวทางสำหรับผู้สนใจที่จะนำความรู้และแนวความคิดต่างๆ ที่ได้จากโครงงานนี้ไปสร้างการ์ดที่มี SAB 80C535 เป็นซีพียูขึ้นมาและในโครงงานนี้เราจะได้เขียนซอฟท์แวร์ขึ้นมาด้วย เพื่อเป็นการสนับสนุนการใช้งานของการ์ดดังนั้นผู้ที่สามารถนำไปเป็นแนวทางได้ โดยผู้ที่มีเครื่องไอบีเอ็มอยู่แล้วสามารถนำการ์ดที่สร้างขึ้นมาไปใช้งานได้เลย

บทที่ 1

ความรู้ทั่วไปเกี่ยวกับไมโครคอนโทรลเลอร์เบอร์ 80C535

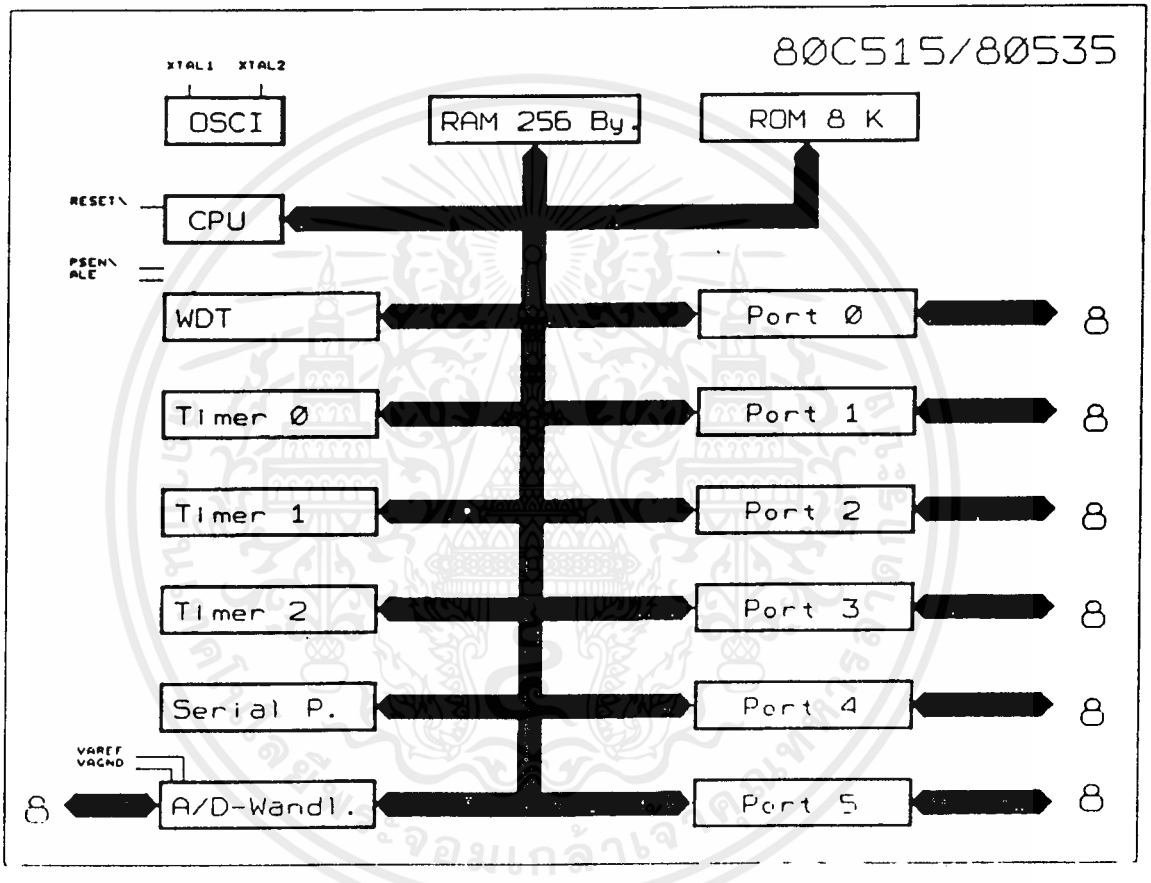
ไมโครคอนโทรลเลอร์เบอร์ 80C535 นี้เป็นอุปกรณ์ที่ออกแบบมาสนองตามความต้องการของผู้ใช้ ที่มีความต้องการความสามารถที่เหนือกว่าไมโครคอนโทรลเลอร์ในตระกูลเอ็มซีเอส 51 คือมีสายอินพุตและเอาต์พุตภายในตัวเอง พอร์ตของอินพุตและเอาต์พุตบัฟเฟอร์อินเตอร์เฟส และสายควบคุมอื่น ๆ ที่ใช้สำหรับแยกข้อมูลกับแอดเดรสและยังมีชุดคำสั่งเพิ่มขึ้นเป็นพิเศษ เพื่อจัดการข้อมูลทั้งทางด้านพอร์ต และพอร์ตแปลงสัญญาณอนาลอกเป็นดิจิตอล แถมท้ายด้วยวงจรตั้งเวลากับวงจรนับด้วย(ปกติวงจรนับจะสามารถทางานเป็นวงจรตั้งเวลาได้ด้วย จึงเรียกควบคู่กันไปคือ วงจรตั้งเวลา/วงจรนับ) ตารางที่ 1 เป็นรายการของไมโครคอนโทรลเลอร์ 80C535 ซึ่งแสดงถึง จำนวนของหน่วยความจำ วงจรตั้งเวลา/วงจรนับ พอตร์แปลงสัญญาณอนาลอกเป็นดิจิตอล และระดับความสำคัญของการอินเตอร์รัพต์

1.1 คุณสมบัติของไมโครคอนโทรลเลอร์เบอร์ 80C535

- มีความสามารถพื้นฐานเหมือนกับ (compatable) เอ็มซีเอส-51
- หนึ่งวัฏจักรคำสั่งใช้เวลา 1 ไมโครวินาที
- มีหน่วยความจำภายใน (RAM) ขนาด 256 ไบต์
- มีหน่วยความจำภายใน (ROM) ขนาด 8 กิโลไบต์ และภายนอกขนาดสามารถขยายได้ถึง 64 กิโลไบต์
- มีรีจิสเตอร์ที่ทำหน้าที่พิเศษ (SFR) 41 ตัว
- โครงสร้างอินเตอร์รัพต์ทำได้ 12 แหล่ง และการจัดระดับความสำคัญ (Priority) ได้ 4 ระดับ
- พอร์ตแบบอนุกรมสามารถที่จะโปรแกรมการรับส่งข้อมูลแบบ Full Duplex ที่ความเร็วสูง
- วงจร Watchdog-Timer ขนาด 16 บิท
- วงจร Timer/Counter ขนาด 16 บิท 3 ตัว

- พอร์ตแบบ อนาลอก/ดิจิตอล โดยที่มีความเร็วในการส่งผ่านข้อมูล 15 ไมโครวินาที
- รูปร่างเป็นแบบ PLCC 68 ขา

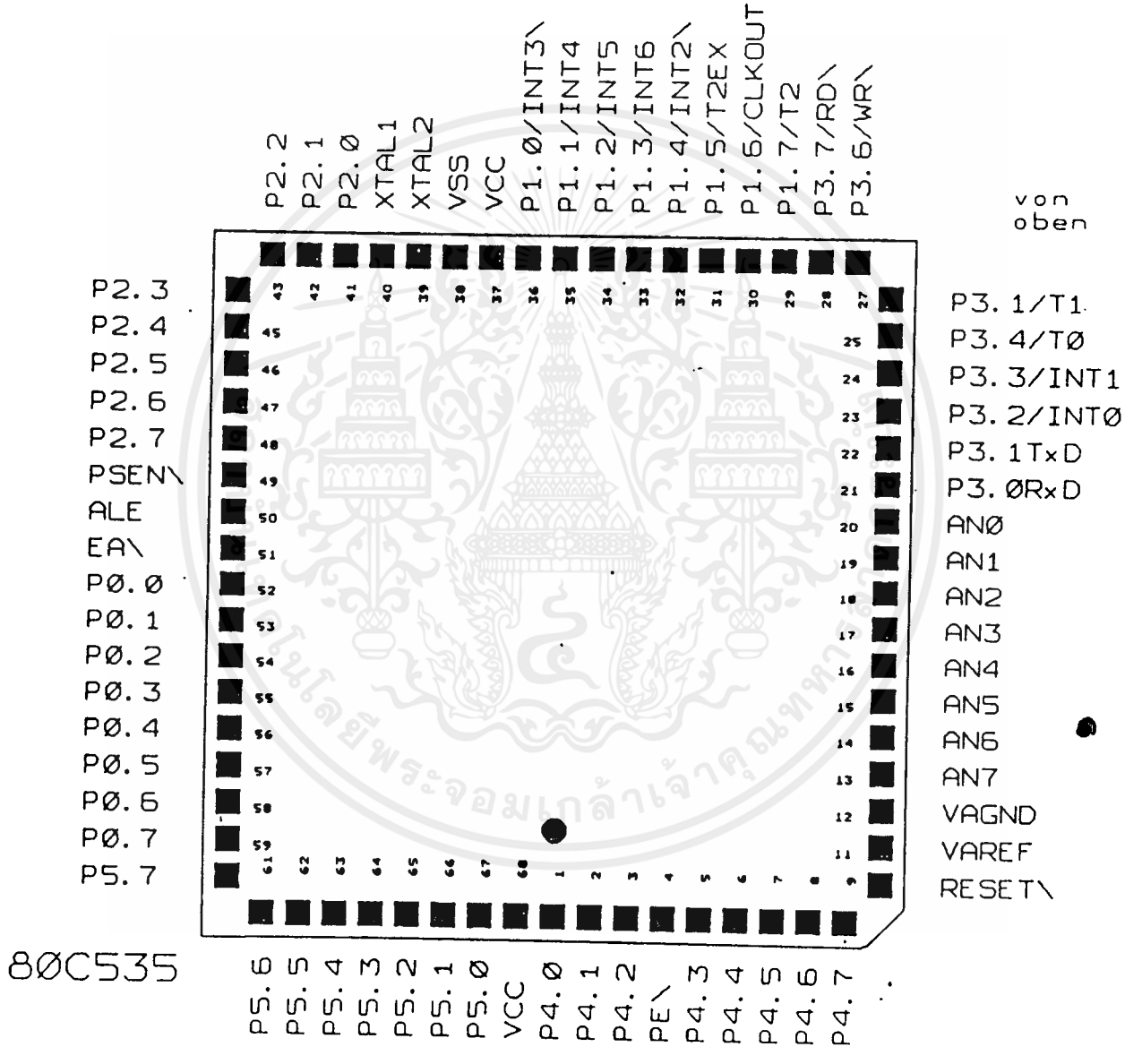
โครงสร้างทั่วไปของ Block Diagram ของ 80C535 แสดงให้เห็นในรูปที่ 1.1



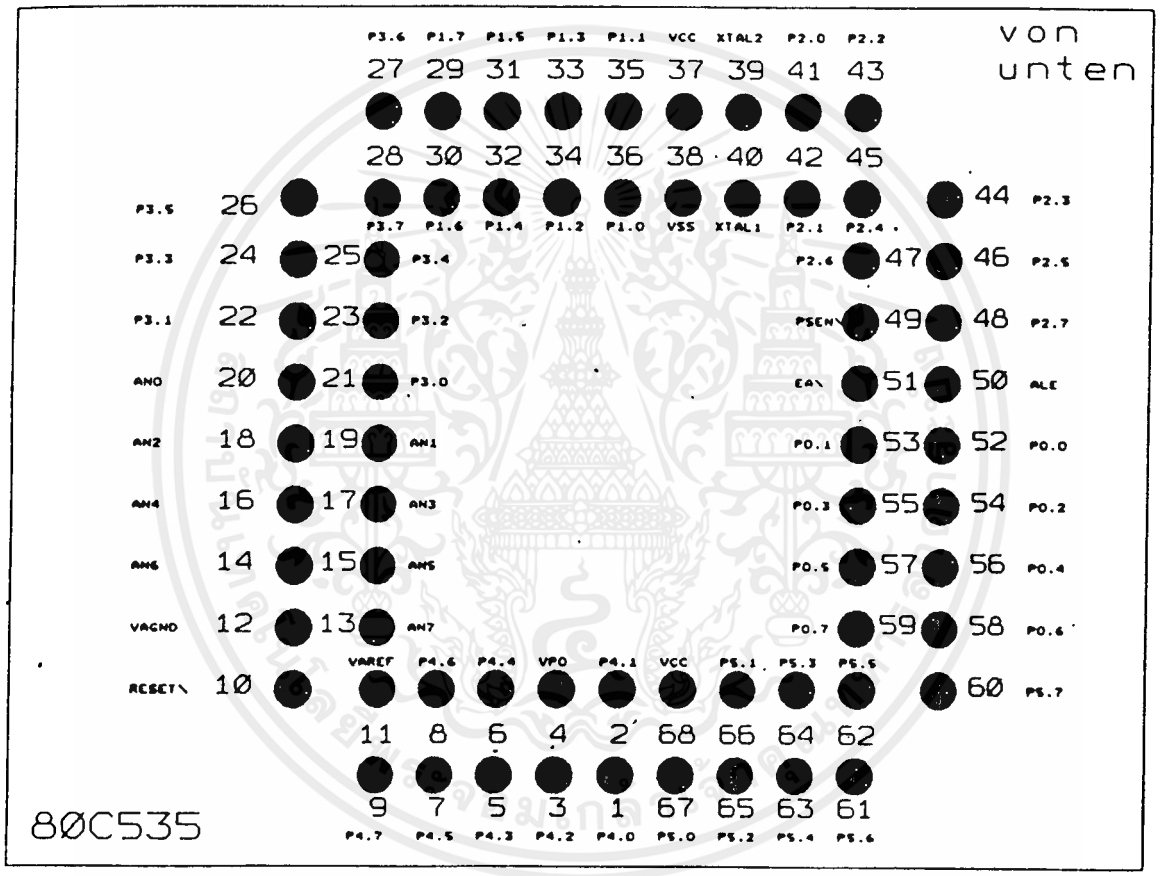
รูปที่ 1.1 แสดง Block Diagram ของ 80C535

โครงสร้างทั่วไปทางด้านฟิสิกส์ได้แสดงให้เห็นในรูปที่ 1.2 และ รูปที่ 1.3

จากรูปที่ 1.2 นั้นแสดงจำนวนขาและหน้าที่ของ 80C535 ซึ่งเป็นแบบ PLCC ที่มีจำนวนขาถึง 68 ขา และมีความเหมือนกันกับ 80C515 ในตระกูลของ SEIMENS จะเห็นได้ว่าชิพเบอร์นี้มีความสามารถมากกว่าชิพในตระกูล เอมซีเอส 51 มากทีเดียวตัวอย่างเช่น มีพอร์ตถึง 6 พอร์ต มีพอร์ต 8 บิต สำหรับแปลงสัญญาณอนาลอกเป็นดิจิตอล และมีวงจรตั้งเวลาและวงจรนับ 3 วงจร เป็นต้น



รูปที่ 1.2 แสดงรูปของ SAB 80C535



รูปที่ 1.3 แสดงการวางขาของ SAB 80C535 ทางด้านล่าง

1.2 การกำหนดขาสัญญาณและพอทต่างๆ ของ 80C535

ขาที่	สัญญลักษณ์	อินพุท/เอาท์พุท	หน้าที่
		Port 0	
52-59	P0.0-0.7	อินพุท/เอาท์พุท	สามารถเป็นได้ทั้ง อินพุท และเอาท์พุทขนาด 8 บิท และมีหน้าที่พิเศษ
		Port 1	
29-36	P1.7-1.0	อินพุท/เอาท์พุท	สามารถเป็นได้ทั้ง อินพุท และเอาท์พุทขนาด 8 บิท
29	P1.7	อินพุท/เอาท์พุท	อินพุทของ Timer T2
30	P1.6	อินพุท/เอาท์พุท	เอาท์พุทของ CLKOUT ของระบบ
31	P1.5	อินพุท/เอาท์พุท	อินพุทภายนอกของ T2EX
32	P1.4	อินพุท/เอาท์พุท	อินพุทของ Interrupt2 INT2
33	P1.3	อินพุท/เอาท์พุท	อินพุทของ INT6/CC3 Interrupt 6
34	P1.3	อินพุท/เอาท์พุท	อินพุทของ INT6/CC3 Interrupt 5
35	P1.3	อินพุท/เอาท์พุท	อินพุทของ INT6/CC3 Interrupt 4
36	P1.3	อินพุท/เอาท์พุท	อินพุทของ INT6/CC3 Interrupt 3

ขาที่	สัญญลักษณ์	อินพุท/เอาท์พุท	หน้าที่
		Port 2	
41-48	P2.0-2.7	อินพุท/เอาท์พุท	พอร์ต 2 มี 8 บิท และพอร์ตสองทาง ทำหน้าที่แอสเดรสทาด้านไบท์สูง
		Port 3	
21-28	P1.7-1.0	อินพุท/เอาท์พุท	สามารถเป็นได้ทั้ง อินพุท และเอาท์พุทขนาด 8 บิท
21	P3.0	อินพุท/เอาท์พุท	RDX Serial Port E/A sync
22	P3.1	อินพุท/เอาท์พุท	TXD Serial Port A/D async
23	P3.2	อินพุท/เอาท์พุท	INT0\,Timer 0 เป็นอินพุท
24	P3.3	อินพุท/เอาท์พุท	INT1\,Timer 1 เป็นอินพุท
25	P3.4	อินพุท/เอาท์พุท	อินพุท Timer 0
26	P3.5	อินพุท/เอาท์พุท	อินพุท Timer 1
27	P3.6	อินพุท/เอาท์พุท	WR\ สัญญานสำหรับเขียน RAM ภายนอก
28	P3.7	อินพุท/เอาท์พุท	RD\ สัญญานสำหรับอ่าน RAM ภายนอก
		Port 4	
1-3	P4.0-4.2	อินพุท/เอาท์พุท	สามารถเป็นได้ทั้ง อินพุท
5-9	P4.3-4.7		และเอาท์พุทขนาด 8 บิท

ขาที่	สัญญาลักษณ์	อินพุท/เอาท์พุท	หน้าที่
		Port 5	
60-67	P5.7-5.0	อินพุท/เอาท์พุท	สามารถเป็นได้ทั้ง อินพุท และเอาท์พุทขนาด 8 บิท
		Port 6	
13-20	AN7-AN0	อินพุท	เป็นอนาลอกอินพุท
สัญญานควบคุม และระบบไฟ			
39	XTAL2		Oscilator
40	XTAL1		Oscilator
10	RESET\	อินพุท	เป็นสัญญาน RESET CPU Active LOW
49	PSEN\	เอาท์พุท	Program Store Enable
50	ALE	เอาท์พุท	Address Latch Enable
4	PE\	อินพุท	
51	EA	อินพุท	LOW=ext,HIGH=int
11	VAREF		5V Referent สำหับ A/D
12	VAREF		0V Referent สำหับ A/D
68	VCC		Supply สำหรับ CPU 5V
38	VSS		Supply สำหรับ CPU GND 0V

1.2 สถาปัตยกรรมทางด้านการคำนวณของ SAB 80C535

SAB 80C535 นั้นเป็นซีพียูขนาด 8 บิท ข้อมูล 16 บิท แอสเดรสและมีหน่วยความจำภายในตัวซีพียู (RAM) มีรีจิสเตอร์พิเศษ (SFR) ถึง 41 ตัว โดยที่มี 14 รีจิสเตอร์ที่ที่เกี่ยวข้องคือ

		Adress [H]
ADCON A/D	ควบคุมการเปลี่ยน A/D	D8*
CCEN	ควบคุม Capture Compare	C1
DAPR	โปรแกรมรีจิสเตอร์ A/D	DA
IEN0	Interrupt Enable 0	A8*
IEN1	Inetrrupt Enable 1	B8*
IP0	ระดับความสำคัญที่ 1 ของ Interrupt	A9
IP1	ระดับความสำคัญที่ 2 ของ Interrupt	B9
IRCON	ควบคุมความต้องการขอ Interrupt	C0*
PCON	ควบคุม Power	87
PSW	โปรแกรมแสดงสถานะคำ	D0
SCON	ควบคุมพอร์ตอนุกรม	98*
T2CON	ควบคุม Timer 2	C8
TCON	ควบคุม Timer	88*
TMOD	Timer Mode	89

* = bitaddress

โดยแสดงรายระเอียดของรีจิสเตอร์หน้าที่พิเศษ (SFR) ตามลำดับตัวอักษรตามตารางข้างนี้

สัญญาลักษณ์	แอสเดรส	คำอธิบาย
ACC	E0	แอคคิวมูเลเตอร์
ADCON	D8	ควบคุมการเปลี่ยน A/D
ADDAT	D9	การเปลี่ยนรีจิสเตอร์ A/D
B	F0	รีจิสเตอร์ B
CCEN	C1	Compare Capture Enable Register
CCH1	C3	Compare Capture Register 1 (H)
CCH2	C5	Compare Capture Register 2 (H)
CCH3	C7	Compare Capture Register 3 (H)
CCL1	C2	Compare Capture Register 1 (L)
CCL2	C4	Compare Capture Register 2 (L)
CCL3	C6	Compare Capture Register 3 (L)
CRCH	C9	Compare Reload Capture Register (H)
CRCL	CA	Compare Reload Capture Register (L)
DAPR	DA	โปรแกรมการเปลี่ยน A/D
DPH	83	Datapointer (H)
DPL	82	Datapointer (L)
IEN0	A8	Interrupt Enable Register 0
IEN1	B8	Interrupt Enable Register 1
IP0	A9	รีจิสเตอร์ระดับความสำคัญของ Interrupt 0
IP1	B9	รีจิสเตอร์ระดับความสัมคัญของ Interrupt 1
IRCON	C0	การควบคุมความต้องการขอ Interrupt
P0	80	พอร์ต 0
P1	90	พอร์ต 1
P2	A0	พอร์ต 2
P3	B0	พอร์ต 3

ตารางที่ 1.1 แสดงรายระเอียดของรีจิสเตอร์หน้าที่พิเศษ

สัญลักษณ์	แอดเดรส	คำอธิบาย
P4	E8	พอร์ต 4
P5	F8	พอร์ต 5
PCON	87	Power Control Register
PSW	D0	โปรแกรมแสดงสถานะ Word Register
SBUF	99	บัฟเฟอร์รีจิสเตอร์สำหรับพอร์ตอนุกรม
SCON	98	การควบคุมรีจิสเตอร์ของพอร์ตอนุกรม
SP	81	Stackpointer
T2CON	C8	Timer 2 Control Register
T2H	CD	Timer 2 (H)
T2L	CC	Timer 2 (L)
TCON	88	Timer Control Register
TH0	8C	Timer 0 (H)
TH1	8D	Timer 1 (L)
TL0	8A	Timer 0 (L)
TL1	8B	Timer 1 (H)
TMOD	89	โหมด Timer Register

ตารางที่ 1.1 แสดงรายระเอียดของรีจิสเตอร์หน้าที่พิเศษ (ต่อ)

SFR ADR								
ACC	ACC.7	ACC.6	ACC.5	ACC.4	ACC.3	ACC.2	ACC.1	ACC.0
E0	E7	E6	E5	E4	E3	E2	E1	E0
ADCON	BD	CLK	-	BSY	ADM	MX2	MX1	MX0
D8	DF	DE	DD	DC	DB	DA	D9	D8
B	B.7	B.6	B.5	B.4	B.3	B.2	B.1	B.0
F0	F7	F6	F5	F4	F3	F2	F1	F0
IEN0	EAL	WDT	ET2	ES	ET1	EX1	ET0	EX0
A8	AF	AE	AD	AC	AB	AA	A9	A8
IEN1	EXEN2	SWDT	EX6	EX5	EX4	EX3	EX2	EADC
B8	BF	BE	BD	BC	BB	BA	B9	B8
IRCON	EXF2	TF2	IEX6	IEX5	IEX4	IEX3	IEX2	IACD
C0	C7	C6	C5	C4	C3	C2	C1	C0
P0	P0.7	P0.6	P0.5	P0.4	P0.3	P0.2	P0.1	P0.0
80	87	86	85	84	83	82	81	80
P1	P1.7	P1.6	P1.5	P1.4	P1.3	P1.2	P1.1	P1.0
90	97	96	95	94	93	92	91	90
P2	P2.7	P2.6	P2.5	P2.4	P2.3	P2.2	P2.1	P2.0
A0	A7	A6	A5	A4	A3	A2	A1	A0
P3	P3.7	P3.6	P3.5	P3.4	P3.3	P3.2	P3.1	P3.0
B0	B7	B6	B5	B4	B3	B2	B1	B0
P4	P4.7	P4.6	P4.3	P4.4	P4.3	P4.2	P4.1	P4.0
E8	EF	EE	ED	EC	EB	EA	E9	E8
P5	P5.7	P5.6	P5.5	P5.4	P5.3	P5.2	P5.1	P5.0
F8	FF	FE	FD	FC	FB	FA	F9	F8
PSW	CY	AC	F0	RS1	RS0	OV	F1	P
D0	D7	D6	D5	D4	D3	D2	D1	D0
SCON	SM0	SM1	SM2	REN	TB8	RB8	TI	RI
98	9F	9E	9D	9C	9B	9A	99	98
T2CON	T2PS	I3FR	I2FR	T2R1	T2R0	T2CM	T2I1	T2I0
C8	CF	CE	CD	CC	CB	CA	C9	C8
TCON	TF1	TR1	TF0	TR0	IE1	IT1	IE0	IT0
88	8F	8E	8D	8C	8B	8A	89	88

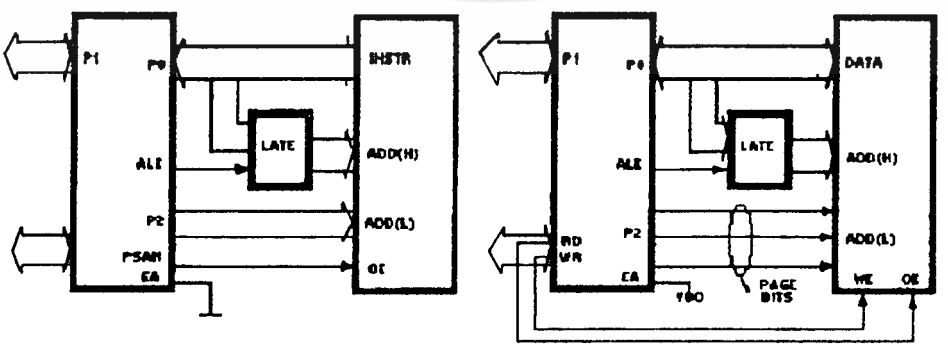
รูปที่ 1.4 แสดงบิทแอสเดรสในรีจิสเตอร์หน้าที่พิเศษ

1.3 การจัดหน่วยความจำภายในและภายนอก

ไมโครคอนโทรลเลอร์ 80C535 จะแบ่งตามพื้นที่หน่วยความจำของการกำหนดเลขที่อยู่แอดเดรส ได้เป็นส่วนที่ประกอบด้วยเนื้อที่ของหน่วยความจำโปรแกรมและหน่วยความข้อมูล โดยจะมีการแบ่งหน่วยความจำเหมือนกับซีพียูทั่วๆไป คือจะแบ่งเป็น 2 ลักษณะตามชนิดของข้อมูลที่เก็บดังนี้

- หน่วยความจำข้อมูล (Data memory)
- หน่วยความจำโปรแกรม (Program memory)

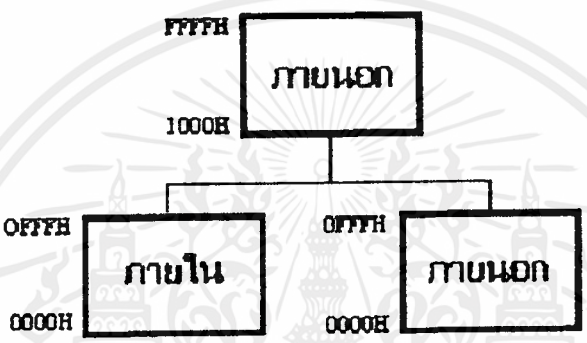
ถ้าจะพูดกันง่ายๆ หน่วยความจำข้อมูลก็หมายถึงหน่วยความจำส่วนที่เป็นแรม(RAM) ซึ่งเราสามารถอ่านหรือเขียนข้อมูลเปลี่ยนแปลงได้ตลอดเวลา แต่ไม่สามารถรันโปรแกรมบนหน่วยความจำส่วนนี้ได้ ส่วนหน่วยความจำโปรแกรม หมายถึงหน่วยความจำที่อ่านได้อย่างเดียว(ROM) ซึ่งบรรจุโปรแกรมที่จะทำให้ SAB 80C535 ทำงาน โดยหน่วยความจำทั้ง 2 ประเภทนี้จะถูกแยกออกจากกันด้วยคำสั่งทางซอฟต์แวร์และลักษณะการติดต่อทางฮาร์ดแวร์ด้วย กล่าวคือ จะมีคำสั่งเฉพาะสำหรับการติดต่อกับหน่วยความจำชนิดใดชนิดหนึ่งและจัดสัญญาณสโตรบในการติดต่อกับหน่วยความจำแต่ละชนิดแยกต่างหากกันด้วย ดังรูปที่ 1.5



รูปที่ 1.5 การเชื่อมต่อกับหน่วยความจำโปรแกรมและหน่วยความจำข้อมูล

1.3.1 หน่วยความจำโปรแกรม

หน่วยความจำโปรแกรมจะประกอบด้วย ส่วนภายในและนอกชิพ ถ้าความยาวของโปรแกรมมีความยาวไม่เกิน 8 กิโลไบต์ สำหรับหน่วยความจำโปรแกรมภายในและสามารถใช้หน่วยความจำโปรแกรมภายนอกได้ถึง 64 กิโลไบต์ โดยอาจจะใช้ร่วมกับหน่วยความนำข้อมูลก็ได้ โดยการควบคุมที่ขา EA\ (ขาที่ 51) หรือทำการต่อลงกราวด์ ดังแสดงในรูปที่ 1.6



รูปที่ 1.6 การจัดพื้นที่หน่วนความจำโปรแกรมภายในและภายนอก

1.3.2 หน่วยความจำข้อมูล

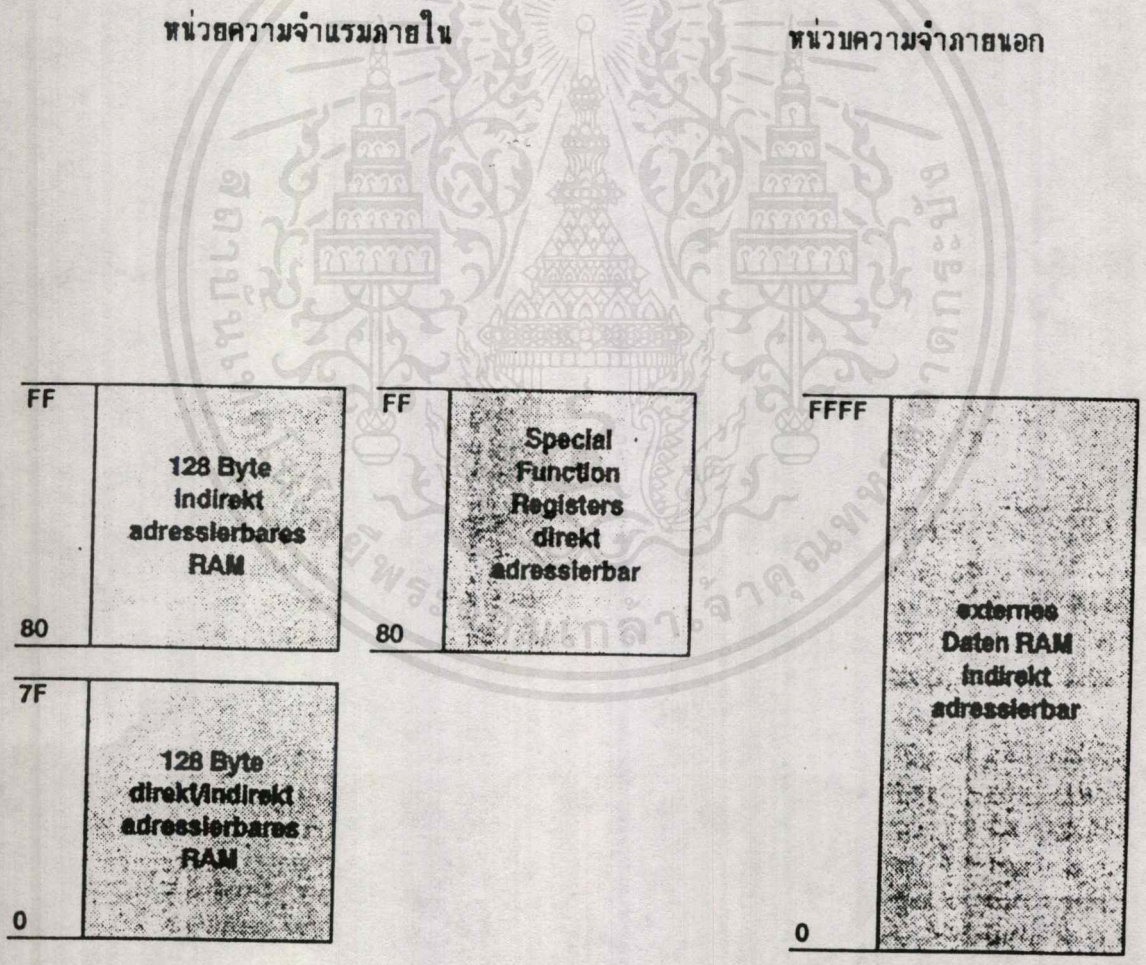
จากรูปที่ 1.7 เป็นการจัดหน่วยความจำภายในของ 80C535 ซึ่งจะเห็นได้ว่าหน่วยความจำข้อมูลภายในจะแบ่งเป็นลักษณะงานดังนี้ คือ จำนวน 128 กิโลไบต์ ของบริเวณตำแหน่งล่างในเนื้อที่แรมภายใน และอีก 128 กิโลไบต์ เป็นของบริเวณตำแหน่งบนของแรมภายในและส่วนของ 128 กิโลไบต์ อีกบริเวณหนึ่งใช้เป็นรีจิสเตอร์ฟังชันก์พิเศษ

จากรูปที่ 1.8 จะเห็นว่าพื้นที่หน่วยความจำข้อมูลภายในส่วน 128 ไบต์ล่าง (ตำแหน่ง 00H-7FH) จะถูกแบ่งเป็น 3 ส่วนคือ ส่วนของรีจิศเตอร์แบงค์ (00H-1FH) ส่วนพิเศษ ที่สามารถเข้าถึงตำแหน่งบิตได้โดยตรง (20H-2FH) และส่วนที่ใช้งานทั่วไป (30H-7FH)

ส่วนของรีจิสเตอร์แบงค์มีทั้งหมด 4 แบงค์ แต่เราสามารถใช้ได้ครั้งละแบงค์เท่านั้น การเลือกใช้แบงค์ไหนอยู่ที่เรากำหนดค่าในรีจิสเตอร์ PSW คำสั่งที่ใช้ในการติดต่อกับหน่วยความจำข้อมูล ซึ่งจะแบ่งลักษณะการกำหนดเลขที่อยู่ของหน่วยความจำข้อมูลได้ 4 โหมดด้วยกันคือ

- โหมดการกำหนดเลขที่อยู่รีจิสเตอร์
- โหมดการกำหนดเลขที่อยู่โดยอ้อม
- โหมดการกำหนดเลขที่อยู่โดยตรง
- โหมดการกำหนดเลขที่อยู่รีจิสเตอร์โดยทันที

รูปที่ 1.8 จะแสดงถึงแผนที่ของหน่วยความจำข้อมูลโดยผ่านเป็น 4 แบงค์ ในแต่ละแบงค์มีรีจิสเตอร์ 8 ตัว มีตำแหน่งตั้งแต่ 0 ถึง 31 ในบริเวณส่วนล่างของแรม



รูปที่ 1.7 การจัดพื้นที่หน่วยความจำข้อมูล

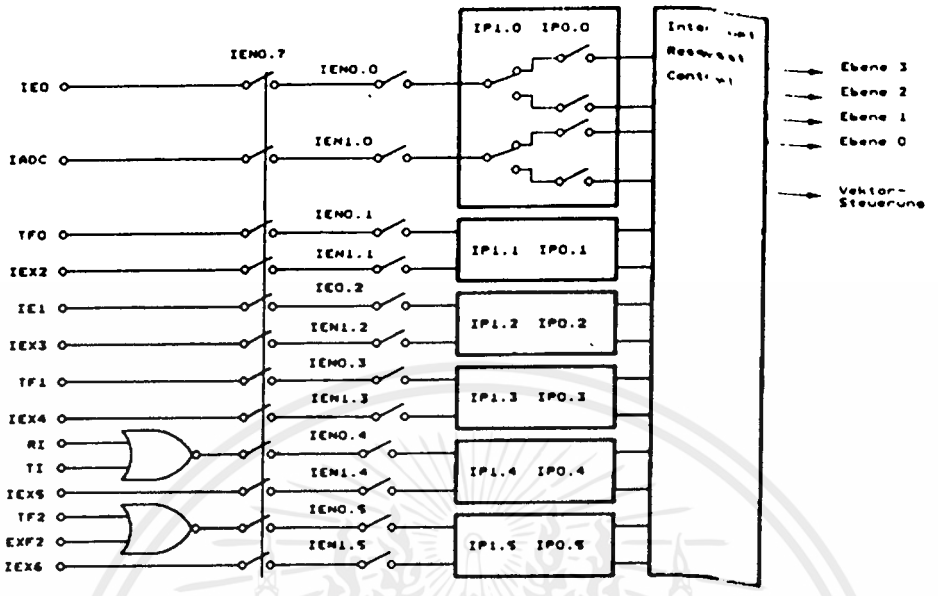
FFH	255		indirekt adressierbar
			"
80H	120		"
7FH	127		direkt und indirekt adr.
			"
30H	48		"
2FH	47	7F 7E 7D 7C 7B 7A 79 78	bitadressierbar
2EH	46	77 76 75 74 73 72 71 70	"
2DH	45	6F 6E 6D 6C 6B 6A 69 68	"
2CH	44	67 66 65 64 63 62 61 60	"
2BH	43	5F 5E 5D 5C 5B 5A 59 58	"
2AH	42	57 56 55 54 53 52 51 50	"
29H	41	4F 4E 4D 4C 4B 4A 49 48	"
28H	40	47 46 45 44 43 42 41 40	"
27H	39	3F 3E 3D 3C 3B 3A 39 38	"
26H	38	37 36 35 34 33 32 31 30	"
25H	37	2F 2E 2D 2C 2B 2A 29 28	"
24H	36	27 26 25 24 23 22 21 20	"
23H	35	1F 1E 1D 1C 1B 1A 19 18	"
22H	34	17 16 15 14 13 12 11 10	"
21H	33	0F 0E 0D 0C 0B 0A ´09 ´08	"
20H	32	´07 ´06 ´05 ´04 ´03 ´02 ´01 ´00	"
1FH	31		Registerbank 4
			"
18H	24	´07 ´06 ´05 ´04 ´03 ´02 ´01 ´0	"
17H	23		Registerbank 2
			"
10H	16	´07 ´06 ´05 ´04 ´03 ´02 ´01 ´0	"
0FH	15		Registerbank 1
			"
08H	8	´07 ´06 ´05 ´04 ´03 ´02 ´01 ´0	"
07H	7		Registerbank 0
			"
00H	0	´07 ´06 ´05 ´04 ´03 ´02 ´01 ´0	"

รูปที่ 1.8 พื้นที่การกำหนดตำแหน่งบิตของหน่วยความจำข้อมูลภายในจากรูปที่ 1.7

1.4 การอินเตอร์รัพต์

ในระบบของ SAB 80C535 จะมีการอันเตอร์รัพต์ได้จาก 6 แหล่ง โดยสามารถตั้งระดับความสำคัญได้ 4 ระดับซึ่งมีแผนภูมิทางฮาร์ดแวร์ของการอินเตอร์รัพต์ ดังรูปที่ 1.9

การอินเตอร์รัพต์ระดับความสำคัญต่ำ สามารถที่จะถูกอินเตอร์รัพต์ตัวอื่นที่มีระดับความสำคัญสูงได้ แต่ไม่สามารถที่จะถูกอันเตอร์รัพต์จากอินเตอร์รัพต์ตัวอื่นที่มีระดับ



รูปที่ 1.9 แผนภูมิทางฮาร์ดแวร์ของแหล่งอินเตอร์รัพต์ชนิดต่างๆ

ของการอินเตอร์รัพต์เดียวกันได้ แต่ถ้าเป็นการอินเตอร์รัพต์จากตัวที่มีความสำคัญสูงจะไม่สามารถถูกอินเตอร์รัพต์จากตัวอื่นได้เลย โดยเราสามารถตั้งระดับความสำคัญของแต่ละแหล่งอินเตอร์รัพต์ได้ที่รีจิสเตอร์ IP และถ้าในเหตุการณ์ที่มีการร้องขออินเตอร์รัพต์ในระดับความสำคัญของแต่ละแหล่ง ดังตารางที่ 1.2

อินเตอร์รัพต์ 0 จากภายนอก	+	อินเตอร์รัพต์ของ A/D
อินเตอร์รัพต์ของตัวจับเวลาตัวที่ 0	+	อินเตอร์รัพต์ 2 จากภายนอก
อินเตอร์รัพต์ 1 จากภายนอก	+	อินเตอร์รัพต์ 3 จากภายนอก
อินเตอร์รัพต์ของตัวจับเวลาตัวที่ 1	+	อินเตอร์รัพต์ 4 จากภายนอก
การรับ/ส่ง อนุกรรม	+	อินเตอร์รัพต์ 5 จากภายนอก
อินเตอร์รัพต์ของตัวจับเวลาตัวที่ 2	+	อินเตอร์รัพต์ 6 จากภายนอก
อินเตอร์รัพต์ตัวที่ 2 จากภายนอก		

ตารางที่ 1.2 ลำดับความสำคัญของแหล่งการอินเตอร์รัพต์ต่าง

บทที่ 2

การสร้างฮาร์ดแวร์ของ 80C535 ไมโครคอนโทรลเลอร์

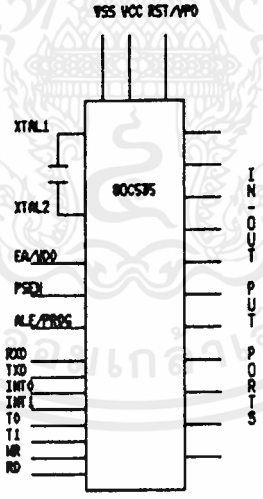
คนที่อยากจะใช้งานชิพตัวใหม่ๆ คงไม่สะดวกกับการติดต่อกับอุปกรณ์นั้น ไม่เหมือนกับชิพที่ได้รับความนิยมแล้วชิพเหล่านี้จะได้ถูกพัฒนาออกมาเป็นชิงเกิ้ลบอร์ด และมีซอฟแวร์สนับสนุนมากมาย ทำให้ชิฟที่ดีมีคุณภาพที่ออกมาใหม่อาจจะไม่เป็นที่รู้จักแพร่หลายโดยทั่วไป เพราะฉะนั้นในที่นี้จะเป็นกล่าวถึงการทำงาน และการสร้างให้ชิฟเหล่านี้ทำงานบนไมโครคอมพิวเตอร์ เพื่อเป็นการสะดวกในการพัฒนาโปรแกรมและความยืดหยุ่นในการใช้งานสูง เพราะมีโปรแกรมหลายตัวที่ใช้ความสามารถเข้าช่วยเสริมการทำงานของมัน

2.1 คุณสมบัติของ 80C535 ไมโครคอนโทรลเลอร์

- ใช้ความถี่ออสซิเลเตอร์ที่ 12 MHz
- ใช้หน่วยความจำขนาด 8 กิโลไบต์ (สามารถขยายได้ถึง 64 กิโลไบต์)
- 8 บิต อนาลอกอินพุท (P6.0-7)
- 16 บิต ดิจิตอลอินพุท/เอาต์พุท (P1.0-7;P3.0,1,4 และ P4.4-7)
- 1 อนาลอกเอ้าต์พุต
- วงจรตั้งเวลา/นับ 3 ตัว (Timer 0,1 และ 2)
- พอร์ต RS-232
- มิอินเตอร์รัพ 2,3,4,5,6
- มีสัญญาณการ RESET จากภายนอก
- มีไฟเลี้ยงที่พอร์ตขนาด 0,+5,+12 และ 12

2.2 แนวทางการออกแบบ

ในการที่คิดจะทำปริญญานิพนธ์เรื่องไมโครคอนโทรลเลอร์ 80C535 นี้ขึ้นมาจากความคิดที่ได้เห็นการพัฒนาซอฟแวร์และฮาร์ดแวร์ในชิพทั่วไปนั้นส่วนใหญ่ จะต้องพัฒนาบนตัวของมันเองโดยการสร้างอยู่ในชุดของชุดคอนโทรลเลอร์ก็ดีหรืออยู่ในลักษณะของ Single board ก็ดีล้วนแต่ไม่มีความยืดหยุ่นในการใช้งานและ ใช้งานลำบากที่จะพัฒนาโปรแกรม เพราะการพัฒนาโปรแกรมบนไมโครคอนโทรลเลอร์นั้น โดยส่วนใหญ่นั้นต้องเขียนเป็นภาษาแอสแซมบลี้ ซึ่งเป็นภาษาที่ยากแก่การพัฒนาเพราะต้องมีการแก้ไขอยู่ตลอดเวลา โดยต้องนำโปรแกรมเหล่านั้นต้องถูกนำไปโหลดลงไปสู่ EPROM ถ้ามีการแก้ไขโปรแกรมก็จะต้องมีลบ EPROM และนำไปโหลดใหม่อีกครั้ง หรือการนำ EPROM EMULATOR มาใช้ก็มีความยุ่งยากอยู่พอสมควร ด้วยเหตุนี้จึงเป็นเหตุให้เกิดปริญญานิพนธ์ฉบับนี้ขึ้นมา เพราะทุกสิ่งทุกอย่างนั้นจะถูกกระทำและพัฒนาอยู่บนเครื่องไมโครคอมพิวเตอร์ทั้งหมด ดังนั้นการพัฒนาจึงมีความยืดหยุ่นสูงและทำการแก้ไขโปรแกรมได้ง่าย เพราะมีเครื่องมือในการพัฒนาที่เรียกว่า DEBUGER



รูปที่ 2.1 แสดงหน้าที่การทำงานของไมโครคอลโทรเลอร์พื้นฐาน

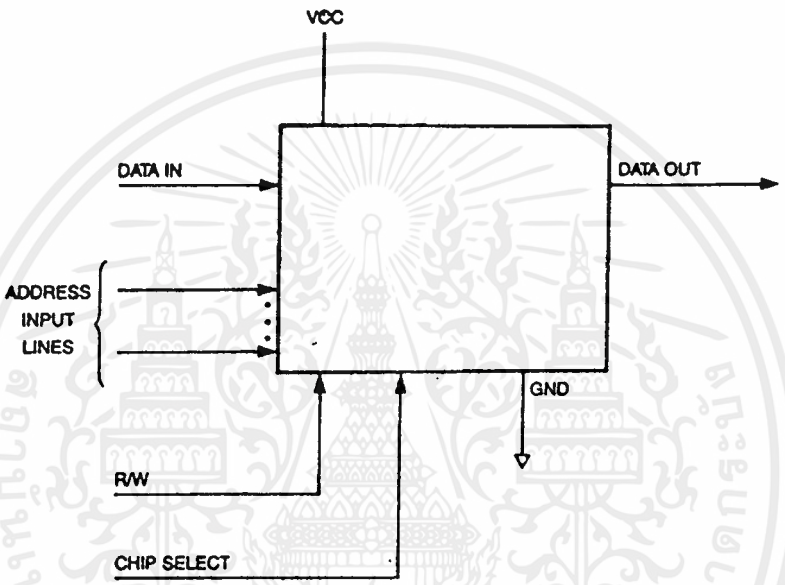
2.2 การสร้างระบบไมโครคอลโทรเลอร์

ระบบไมโครคอลโทรเลอร์นั้น เป็นอุปกรณ์ที่ออกแบบมาเพื่อสนองตามความต้องการของผู้ใช้คือมีสายอินพุตและเอาต์พุทบัฟเฟอร์อินเตอร์เฟส และสายควบคุมอื่น ๆ ที่ใช้สำหรับแยกข้อมูลกับแอดเดรสและยังมีชุดคำสั่งเพิ่มขึ้นเป็นพิเศษเพื่อจัดการข้อมูล และยังมีวงจรตั้งเวลากับวังจรนับ(ปกติวงจรนับจะสามารถทำงานเป็นวงจรตั้งเวลาได้ด้วย จึงเรียก

ควบคู่กันไปคือ วงจรตั้งเวลา/วงจรนับ) และยังมี วงจรเปลี่ยนสัญญานอนาลอกเป็นสัญญานดิจิตอลด้วย

- การออกแบบการใช้หน่วยความจำภายนอก

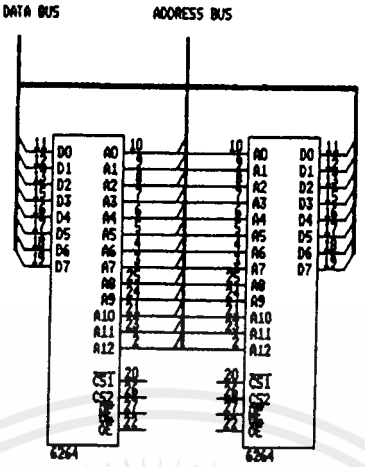
ในการออกแบบ 80C535 การ์ดนั้นในที่นี้เราจะใช้ Static RAM ในการใช้งานเพราะมีความเร็วสูงในการส่งถ่ายข้อมูล และมีความง่ายในการออกแบบโดยมีพื้นฐานในการทำงานดังรูปที่ 2.2



รูปที่ 2.2 แสดงสัญญาณทางไฟฟ้าที่จำเป็นในการใช้

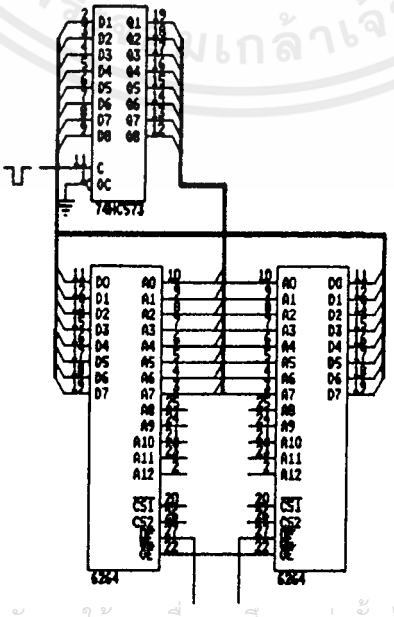
จากรูป "DATA IN" เป็นสายที่ใช้นำข้อมูลที่ต้องการเขียนเข้ามาในแรม ส่วน "DATA OUT" ใช้สำหรับอ่านข้อมูลที่อยู่ในแรมออกไปในแรมตัวหนึ่ง ๆ อย่างน้อยที่สุดจะต้องมี DATA IN หนึ่งสายและ DATA OUT หนึ่งสาย ซึ่งจำนวนที่แน่นอนของสายเหล่านี้จะขึ้นอยู่กับโครงสร้างภายในของแรม แต่ในโครงงานนี้เราได้กำหนดขึ้นเองจากประสพการณ์ที่เกี่ยวกับไมโครโปรเซสเซอร์ให้มีแรมขนาด 8 กิโลไบท์ สองตัวเป็นตัวสำหรับทำการเก็บโปรแกรมของระบบ

ดังนั้นในการทดลองจึงทำการต่อแรมเบอร์ 6264 เข้าด้วยกันดังรูปที่ 2.3 จะเห็นได้ว่ามีการนำขาของข้อมูล(D0-D9) และขาของแอสเดรส(A0-A16) ของทั้งสองตัวมาต่อเข้าด้วยกัน เพื่อเป็นการอ่านเขียนข้อมูลเดียวกันโดยการเลือกการอ่าน หรือเขียนแรมตัวไหนนั้นจะทำการควบคุมโดยขา OE(Output enable) และ WE(write enable) จากตัวไปโครคอลโทรเลอร์



รูปที่ 2.3 แสดงการต่อแรม เบอร์ 6264

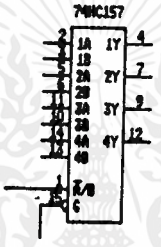
จากการต่อหน่วยความจำภายนอกของ SAB 80C535 จะเห็นได้ว่าในพอร์ท P0 ของซีพียูนั้นจะเป็นทั้งพอร์ทข้อมูลและพอร์ทแอสเดรสอยู่ในพอร์ทเดียวกันดังนั้นจึงต้องหาวิธีแยกข้มูลทั้งสองออกจากกัน โดยใช้วิธีของการแลทช์เอาค่าของแอสเดรสออกไปโดยใช้ไอซีเบอร์ 74HC573 เป็นตัวแลทช์เอาแอสเดรสที่ไบท์สูงออกมา โดยมีการควบคุมการแลทช์ให้มีจังหวะที่สอดคลองกับการทำงานของซีพียู โดยส่งสัญญาณนาฬิกาในจังหวะที่ต้องการดังรูปที่ 2.4



รูปที่ 2.4 แสดงการต่อหน่วยความจำให้กับซีพียู ที่พอร์ท P0

- การเลือกชิพของแรมที่จะทำการอินาเบิ้ล

เมื่อต่อแรมเข้าสู่ระบบ เราจะต้องหาวิธีในการเลือกชิพของแรมได้อย่างถูกต้องตามแอดเดรสที่เรากำหนดไว้ เช่น แรมตัวที่หนึ่งเรากำหนดแอดเดรสไว้ที่แอดเดรส 000 0H ถึง 07FFH ดังนั้นเราจำเป็นต้องมีตัวถอดรหัสเพื่อเลือกแอดเดรสได้ถูกต้อง การถอดรหัสนี้เราจะใช้แอดเดรส ส่วนบนที่เหลือมาทำการถอดรหัสในที่นี้เราจะใช้ 74HC157 ทำการเลือก 2 บิตไปเป็น 4 บิตโดยอินพุต A,B วงจรการถอดรหัสนี้ แสดงได้ดังรูปที่ 2.5



รูปที่ 2.5 การใช้ 74HC157 เป็นตัวถอดรหัสเลือกแรม

- ลำดับการทำงานของสัญญาณต่าง ๆ ระหว่างไอบีเอ็ม พีซี กับ 80C535 การ์ด

ในการทำงานของการส่งสัญญาณเอาต์พุตนั้น ข้อมูลจะต้องได้รับการแลตช์ไว้กับตัวแลตช์ข้อมูล ซึ่งก็คือ D ฟลิปฟลอปนั่นเอง ลักษณะการจัดพอร์ตจึงเสมือนการใช้แลตช์เป็นตัวรับเอาต์พุตให้ข้อมูลคงค้างอยู่ ส่วนการรับอินพุตนั้นโดยทั่วไปใช้หลักการของลอจิกสามสถานะ กล่าวคือข้อมูลที่อินพุตจะได้รับการอินาเบิ้ลผ่านเข้ามายังบัสข้อมูลเพื่อให้ซีพียูอ่านข้อมูลมาเก็บไว้ภายในตามต้องการดังรูปที่ 2.6

1. ลำดับการเขียนเอาต์พุต

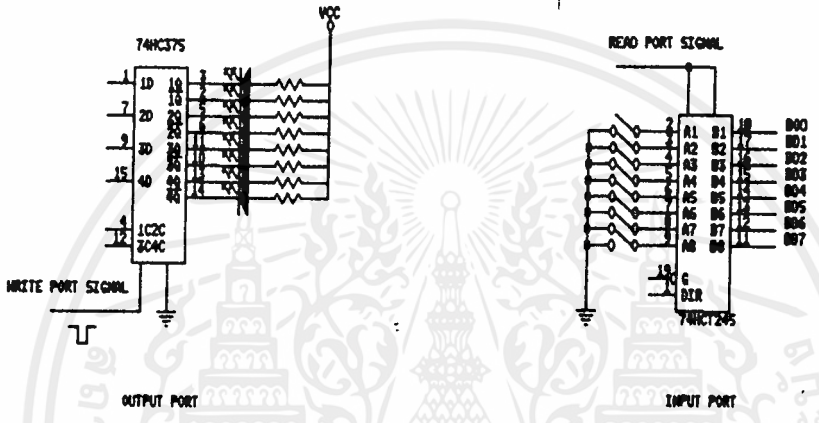
ในการส่งข้อมูลไปยังเอาต์พุตนั้น SAB 80C535 มีลำดับการทำงานดังต่อไปคือเมื่อซีพียูส่งสัญญาณให้มีการเขียนพอร์ตออกมา จะทำให้ไอซี 74HC245 อินาเบิ้ลจึงการทำการแลตช์สัญญาณนั้นออกไปและจะเป็นอย่างนี้ต่อไปเรื่อยๆ ตามที่มันต้องการอ่านข้อมูล

2. ลำดับการอ่านข้อมูลจากอินพุต

ในการอ่านข้อมูลจากอินพุตจำเป็นจะต้องมีลำดับการทำงาน กล่าวคือ ซีพียูจะส่งสัญญาณ RD ออกมาเพื่อบอกว่าต้องการติดต่อกับอินพุต ข้อมูลอินพุตบนบัสจะ

ได้รับการอ่านเข้ามาในรีจิสเตอร์ A

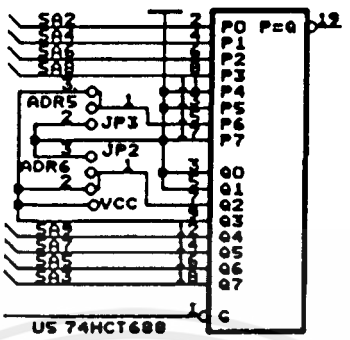
ในรูปที่ 2.6 เป็นส่วนหนึ่งของวงจรแลตช์และลอจิกสามสถานะ วงจรแลตช์ใช้ 74HC375 ซึ่งเป็น D ฟลิปฟลอบ 8 ตัว ส่วนอินพุตใช้ 74HC245 เป็นเกทแบบลอจิกสามสถานะ 8 ตัวเช่นกัน ในการแลตช์ข้อมูลจะใช้สัญญาณ CK ซึ่งก็คือสัญญาณเขียนพอร์ตการเขียนนี้จะกระทำขณะเกิดการเปลี่ยนลอจิกจาก "0" ไปเป็น "1"



รูปที่ 2.6 การใช้แลทช์เป็นเอาต์พุตและลอจิกเกทสามสถานะเป็นอินพุต

- การอินเตอร์เฟส 80C535 เข้ากับไอบีเอ็ม พีซี

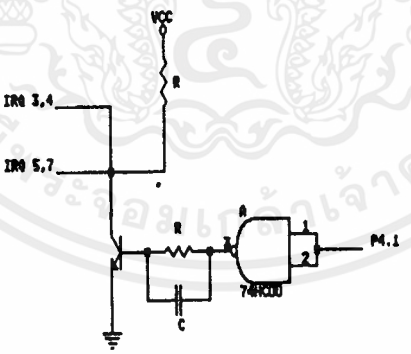
ในการให้ไอบีเอ็ม พีซี ทำการติดต่อกับอุปรณ์ภายนอกนั้นต้องมีการกำหนดค่าของแอสเดรสให้มีค่าตรงกันกับที่เครื่องคอมพิวเตอร์ เพื่อที่จะให้มีการตอบรับและการส่งถ่ายข้อมูลได้อย่างถูกต้อง โดยใช้ 74HCT688 เป็นตัวเปรียบเทียบค่าของแอสเดรสจากเครื่องคอมพิวเตอร์ให้ตรงกับการ์ดที่จะทำการติดต่อด้วย ในโครงงานนี้ได้กำหนดการติดต่อของเครื่องคอมพิวเตอร์กับการ์ด 80C535 ไว้ถึง 4 ช่องทางคือ 100H, 120H, 140H และ 160H ดังรูปที่ 2.7 จะเห็นได้ว่าเราได้ตั้งค่าของแอสเดรสไว้ที่ค่า 160H กล่าวคือเอาท์พุตของ 74HCT688 จะเป็น "0" ก็ต่อเมื่อมีค่าแอสเดรสที่ 160H เท่านั้นแล้วทำการนำเอาท์พุตที่ได้นี้ไปสร้างสัญญาณนาฬิกาสำหรับ ระบบของการ์ด 80C535 ต่อไป



รูปที่ 2.7 แสดงการเลือกค่าแอสเดรสที่จะทำการติดต่อด้วย

- การขออินเตอร์รัพต์ระหว่างไอบีเอ็ม พีซี กับ SAB 80C535

ในการขออินเตอร์รัพต์กับไอบีเอ็ม พีซี นั้นจะใช้กับอินเตอร์รัพต์ 3, 4, 5 และ 7 ของไอบีเอ็ม พีซี เท่านั้น โดยเราได้ตั้งค่าปกติไว้ที่ อินเตอร์รัพต์ 5 และทำได้โดยการใช้ NPN Transistor BC 237 ทำเป็น Open-Collector-Signal ทำเป็นสวิทช์ในการเปิดปิดสัญญาณที่ ซีพียู 80C535 ส่งออกมาเพื่อทำการขออินเตอร์รัพต์ ดังแสดงในรูปที่ 2.8



รูปที่ 2.8 วงจร Open-Collector

จากที่กล่าวมาแล้วในการที่จะทำการสร้างระบบไมโครคอลโทรเลอร์ขึ้นมาที่มีความสมบูรณ์ มีความสามารถสูง และพร้อมที่จะนำไปใช้งานได้นั้นจะประกอบไปด้วยวงจรต่าง ๆ มากมายตามที่กล่าวไว้ข้างต้นและยังมีวงจรอื่นๆที่ไม่ได้กล่าวถึงอีกมากมายเพื่อที่จะประกอบขึ้นมาเป็นระบบไมโครคอลโทรเลอร์ที่มีประสิทธิภาพในการใช้งาน โดยในการสร้างการ์ดนี้ขึ้นมาเราจะทำการแบบวงจรออกเป็น 3 ส่วน ใหญ่คือ ส่วนทางด้านการอินเตอร์เฟสส่วนทางด้านของระบบของไมโครคอลโทรเลอร์ และในส่วนของการต่อไปใช้งานภายนอก

2.4 การทำงานของวงจร

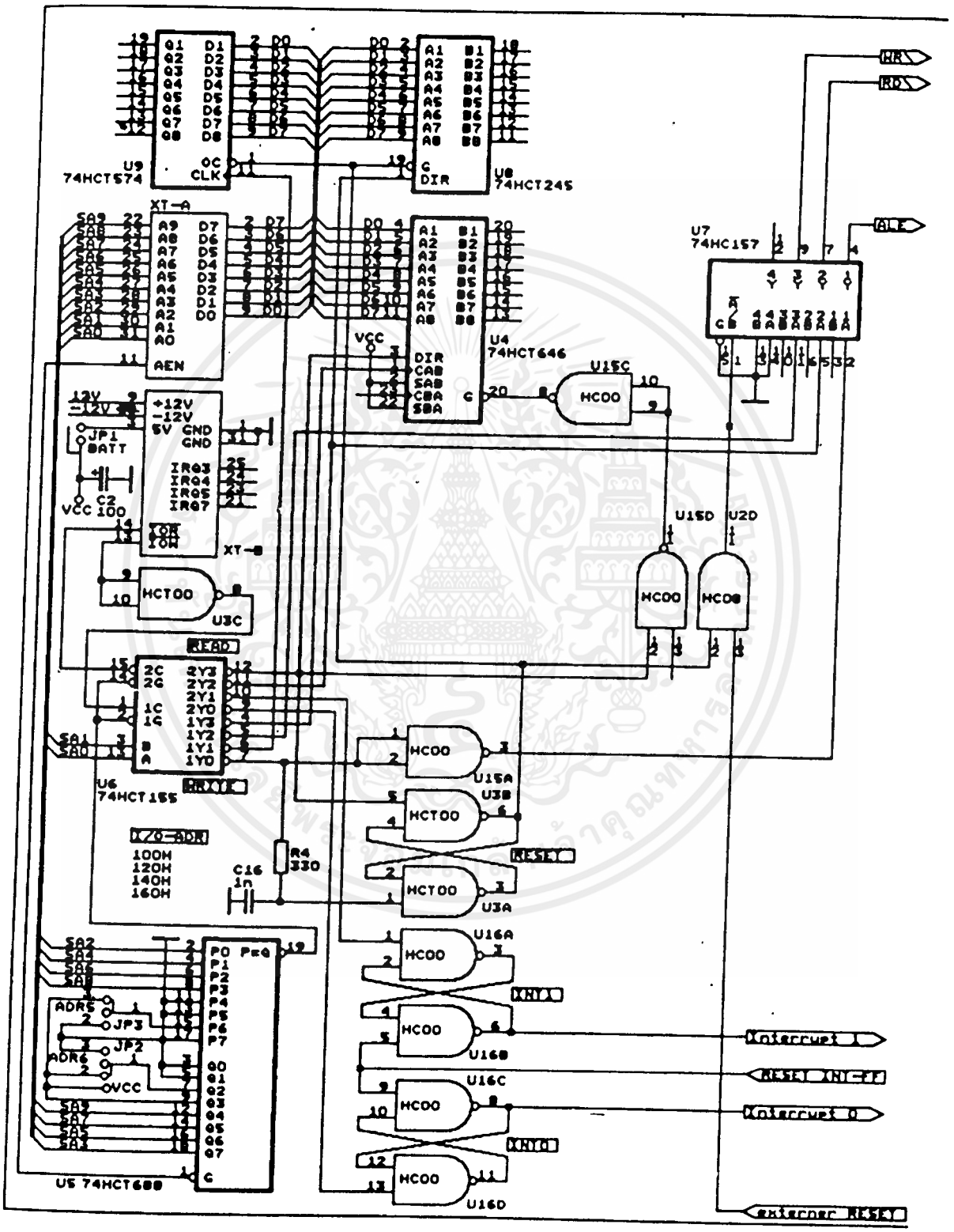
-ทางด้านการอินเตอร์เฟส และการจัดการข้อมูล

วงจรฮาร์ดแวร์ไมโครคอนโทรลเลอร์ 80C535 ในการอินเตอร์เฟสจะแสดงดังรูปที่ 2.9 การอินเตอร์การ์ดนี้เข้ากับเครื่องไมโครคอมพิวเตอร์ จะผ่านทางสล็อตที่ว่างของเครื่องไมโครคอมพิวเตอร์ จากวงจรในรูปที่ 2.9 จะแสดงให้เห็นถึงจำนวนสัญญาณที่ต้องใช้ในการติดต่อกับไมโครคอมพิวเตอร์ คือที่ XT-A และ XT-B เป็นสล๊อตของเครื่องไมโครคอมพิวเตอร์ โดยในขั้นแรกของการทำงานจะเริ่มจากการ Decode ค่าของแอสเดรสให้ตรงกับความต้องการของการ์ด 80C535 โดยใช้ไอซี 74HC688 เป็นตัวเปรียบเทียบค่าของแอสเดรสทั้งทางด้านต่ำและทางด้านสูง เมื่อได้ค่าของแอสเดรสที่ต้องการก็จะสั่งให้ชิพยูทำงานโดยการให้ลอจิก "0" ออกมาทางขา "P" ของ 74HC688 ต่อจากนั้นจะส่งค่าที่ได้นี้ไปให้กับไอซี 74HC155 ซึ่งทำหน้าที่เป็น Decoder/Demultiplexers ทำหน้าที่จ่ายสัญญาณนาฬิกาให้แก่อุปกรณ์ทั้งหมดที่มีในระบบ และเป็นตัวบังคับให้อุปกรณ์ตัวใดทำงานเมื่อใด โดยอยู่ภายใต้การควบคุมของสัญญาณ IOR และ IOW ของไมโครคอมพิวเตอร์ในการที่จะให้การ์ดทำการอ่านหรือเขียนข้อมูลนั้น ๆ โดยจะมีชุดของการควบคุมการอ่านและควบคุมการเขียนโดยใช้ 74HC245 และ 74HC573 ตามลำดับ และใช้ 74HC646 เป็นตัวที่จะทำการควบคุมการอ้างแอสเดรสทางด้านไบท์สูง

- ทางด้านระบบของ SAB 80C535

ในส่วนนี้นั้นจะมีความคล้ายคลึกกับระบบของวงจรไมโครคอลโทรเลอร์ทั่วไป โดยสายต่างๆ ของบัสและพอร์ตจะเห็นโรงสร้างภายในของ SAB 80C535 ในรูปที่ 2.10 เราจะไม่ใช้หน่วยความจำภายในตัวของ SAB 80C535 แต่จะใช้หน่วยความจำจากภายนอกเพียงอย่างเดียว และเมื่อไม่ใช้หน่วยความจำภายใน พอร์ต 0 และ 2 จะถูกใช้เป็นบัสของข้อมูลและแอดเดรส ดังนั้นพอร์ต 2 พอร์ต ยังคงใช้งานเป็นอินพุตและเอาต์พุต โดยที่พอร์ต 2 จะทำหน้าที่เป็นสายสัญญาณแอดเดรส A15-A8 ส่วนพอร์ต 0 จะทำหน้าที่เป็นสายสัญญาณแอดเดรส A7-A0 ออกจากบิตข้อมูล D7-D0

เอาต์พุตของขา RD และ WR มาจากสายเอาต์พุตของพอร์ต 3 โดยโปรแกรมภายในใช้สัญญาณ RD และ WD เพื่อการเขียนและการอ่านข้อมูลกับข้อมูลของหน่วยความนำภายนอก



รูปที่ 2.9 แสดงส่วนของการอินเตอร์รัพต์และการจัดการข้อมูล

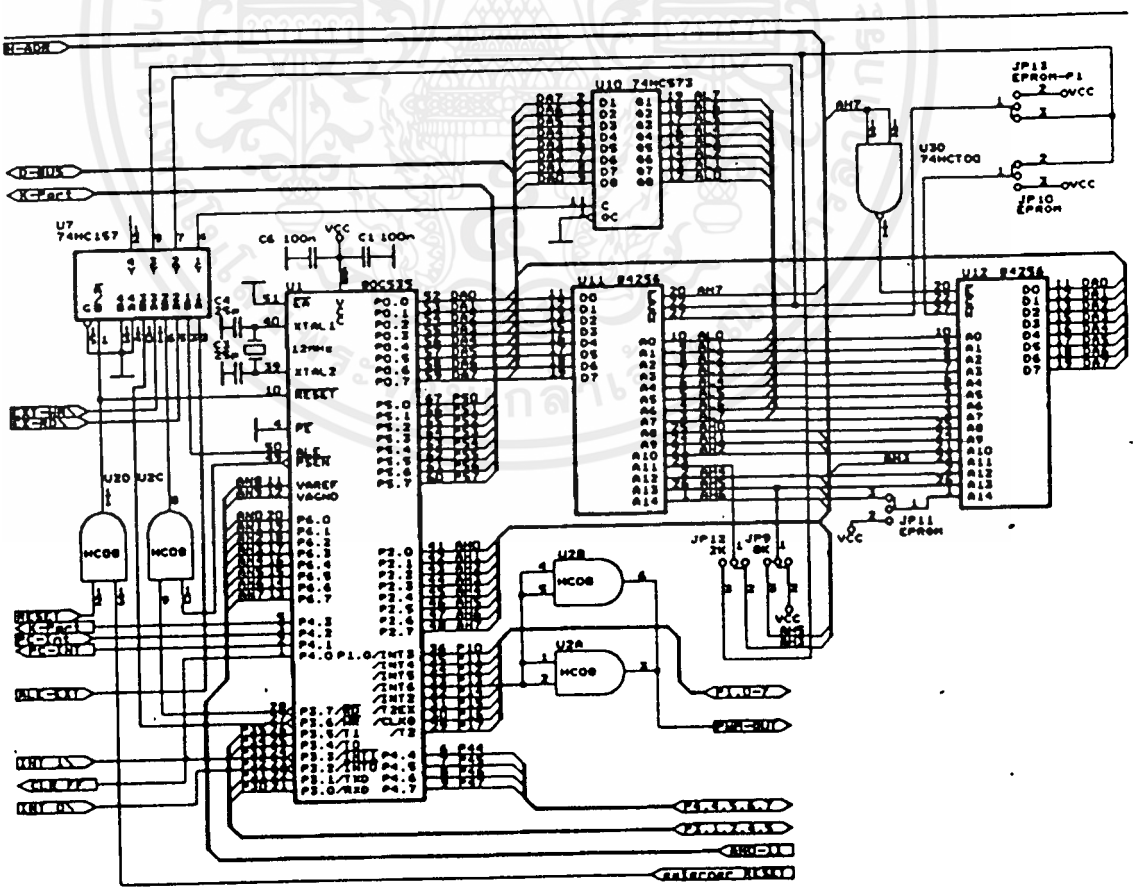
ขา PSEN

เป็นขารับสัญญาณสำหรับเปิดให้มีการอ่านหน่วยความจำภายนอก ถ้าสังเกตทุก ๆ รอลคำสั่งระหว่างการทำงานด้วยโปรแกรมในหน่วยความจำ (RAM,ROM,EPROM) จะเห็นว่าสัญญาณ PSEN ทำงานถึงสองครั้งเหมือนสัญญาณ ALE เพราะว่ามีการอ่านข้อมูลจำนวน 2 ไบท์ ในแต่ละรอบคำสั่ง

ขา PSEN นี้ไม่ทำงานเมื่อมีภาษาเครื่องเก็บอยู่ในหน่วยความจำภายในและถ้าหากหน่วยความจำภายนอกไม่มีข้อมูลบรรจุอยู่ PSEN ก็จะไม่ทำงานด้วยเช่นกัน

ขา EA

เป็นขาอินพุตที่ใช้ร่วมกับแอดเดรสภายนอก โดยจะมีค่าลอจิก "0" เมื่อไมโครคอลโทรเลอร์อ่านคำสั่งจากหน่วยความจำภายนอก(ปกติไมโครคอลโทรเลอร์จะอ่านหน่วยความจำภายในน้อยกว่าอ่านจากหน่วยความจำภายนอก) แต่ในที่เราได้นำขา EA ลงกราวด์เพื่อทำให้เป็นค่าลอจิก "0"



รูปที่ 2.10 แสดงถึงส่วนของระบบไมโครคอลโทรเลอร์

ขา XTAL

ในจุดนี้เราได้ใช้ค่าของออสซิเลเตอร์ 11.0592 เพื่อที่จะทำให้มันทำการติดต่อกับไมโครคอมพิวเตอร์ได้ทั้งทางสล๊อตของเครื่อง และติดต่อท่าง RS-232

ขา ALE

ขานี้จะเป็นสัญญาณแอดเดรสแลตช์อีนาเบิลออกจากตัวซีพียู โดยการบอกอุปกรณ์ภายนอกให้ทำการแลตช์แอดเดรสไปต์ต่ำที่ถูกส่งออกมาในขฯะเฟตช์คำสั่ง

ขา RD

ขานี้เป็นขาสัญญาณการอ่านข้อมูลจากหน่วยความจำข้อมูลภายนอก

จากรูปที่ 2.10 พอร์ต P0 จะทำหน้าที่เป็นทั้งแอดเดรสและดาต้า โดยเมื่อซีพียูต้องการอ่านโปรแกรมจะส่งแอดเดรสไบต์ต่ำออกมาทางพอร์ต P0 และส่งสัญญาณ ALE ให้กับ 74HC573 ทำการแลตช์ เพื่อเป็นแอดเดรสของหน่วยความจำต่อไป ส่วน A8-A15 จะออกทางพอร์ต P2 เมื่อถึงไซเกิลของการเฟตช์คำสั่ง ที่ขา PSEN ของซีพียูจะส่งพัลส์ลอจิก "0" ออกมา ซึ่งจากวงจร ขา PSEN ถูกต่ออยู่กับ 6264 ข้อมูลใน 6264 จะถูกส่งออกไปทาง D0-D7 ไปยังพอร์ต P0 ของซีพียูเพื่อนำคำสั่งไปแปลต่อไป

การต่อแรมในที่นี้นั้นจะอ้างหน่วยความจำได้สูงสุด 64 กิโลไบต์ วิธีการต่อการทำโดยการนำสัญญาณ PSEN และ RD มา AND กันแต่วงจรที่เราทำการออกแบบมานั้นเมื่อนำมาผ่านตัวดีโค้ดอีกทีหนึ่ง และเมื่อตำแหน่งหน่วยความจำอยู่ในหน้าเดียวกันจำเป็นจะต้องกำหนดตำแหน่งของหน่วยความจำของแต่ละตัวว่าอยู่ที่ตำแหน่งใด ในวงจรที่ 2.10 เราจะใช้ 74HC157 เป็นตัวดีโค้ด

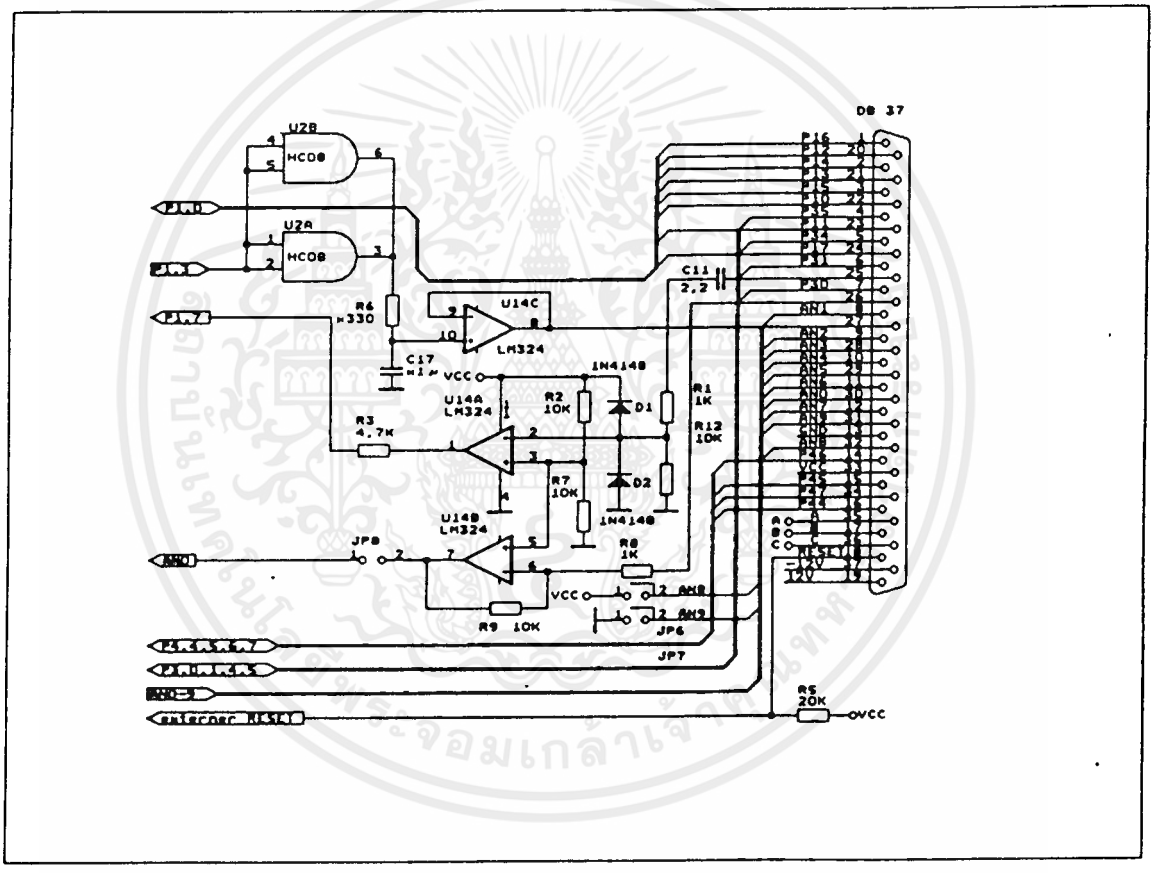
พอร์ตสื่อสารข้อมูลอนุกรม(RS-232) SAB 80C535 มีขา TX และ RX สำหรับส่งและรับข้อมูลทางอนุกรมตามลำดับ ซึ่งอยู่ที่ ขา 21 และขา 22 แต่เนื่องจากสัญญาณที่ขาทั้งสองนี้อยู่ในระดับสัญญาณทาง TTL แต่สัญญาณที่จะต้องใช้ในการส่งข้อมูลของ RS-232 นั้นจะอยู่ในช่วงระหว่าง 15 โวลต์ ทั้งทางบวกและทางลบ ฉะนั้นจึงต้องมีวงจรอินเตอร์เฟส เพื่อจัดระดับสัญญาณให้เหมาะสมจากโดยการใช้ MAX 232 เป็นตัวเพิ่มสัญญาณให้สูงขึ้นที่จะเพียงพอกับการติดต่อ

- ทางด้านการต่อพอร์ตไปใช้งานภายนอก

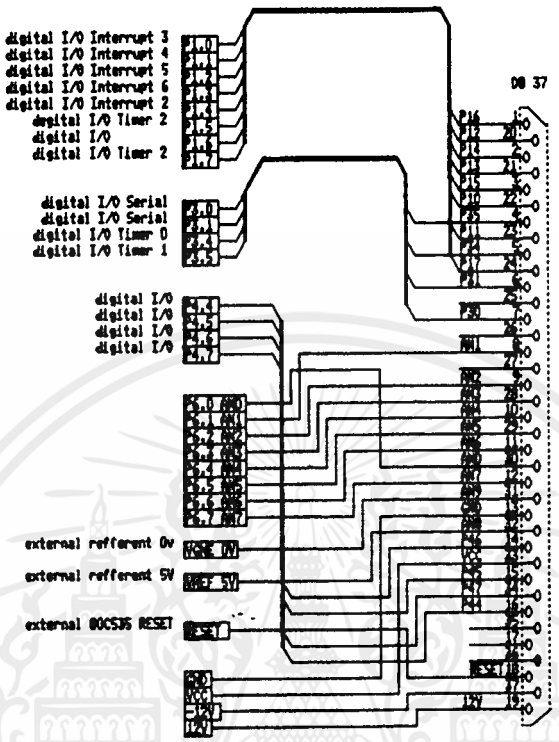
ดังรูปที่ 2.11 จากรูปเป็นการแสดงการต่อพอร์ตและวงจรไดร์เวอร์ไปยังคอนเน็คเตอร์ DB 37 ซึ่งประกอบไปด้วยพอร์ตที่ P1, P3, P4 และ P6 ซึ่งเป็นอนาล๊อกเอ้าต์พุต และสามารถทำการ RESET เครื่องจากภายนอก

จากรูปที่ 2.11 จะประกอบไปด้วยดิจิตอลพอร์ต 3 พอร์ท อนาลอกเอาต์พุตพอร์ต หนึ่งพอร์ต อนาลอกอินพุตพอร์ต 2 พอร์ต ซึ่งอนาลอกเอ้าต์พุตพอร์ตนั้นโดยความจริงแล้วเป็นดิจิตอลพอร์ตแต่เรามาทำการต่อวงจร D/A เอาไว้เพื่อเพิ่มความสามารถของตัวการ์ด โดยใช้วงจร D/A Pulse-width modulated waveform เป็นตัวแปลงสัญญาณดิจิตอลไปเป็นสัญญาณทางอนาลอก

ในรูปที่ 2.12 ได้แสดงจุดต่อและการวางขาของพอร์ตที่จะทำการต่อออกไปใช้งาน



รูปที่ 2.11 แสดงการใช้งานของพอร์ต



รูปที่ 2.12 แสดงจุดต่อพอร์สำหรับต่อไปใช้งาน

2.5 การติดตั้ง JUMPER

การติดตั้ง JUMPER ลงบนบอร์ดไมโครคอนโทรลเลอร์ ให้มีความคล่องตัวสูงและมีความเหมาะสมกับการใช้งานซึ่งแต่งต่างกันไป ได้ให้ความหมายและหน้าที่การทำงานของ JUMPER ไว้ดังนี้

JP1	การใช้งาน 80C535 บนไมโครคอมพิวเตอร์หรือใช้บนตัวมันเอง
JP2,JP3	ตั้งค่าการอ้างแอสเดรสที่ พอร์ตอินพุท/เอาต์พุต
JP4,JP5	ตั้งค่าการอินเตอร์รัพต์ของไมโครคอมพิวเตอร์ 3,4,5 หรือ 7
JP6	กำหนดค่า +5 V ที่พอร์ต (ไฟมาตรฐาน)
JP7	กำหนดค่า 0 V ที่พอร์ต (กราวด์มาตรฐาน)
JP8	เอ้าต์พุตของ ออปแอมป์ที่ AN0
JP9	8 หรือ 16 Kbyte บนบอร์ด

JP10	EPROM/RAM U12
JP11	EEPROM U12
JP12	2 Kbyte บนบอร์ด
JP13	EPROM/ROM U11

บทที่ 3

การออกแบบส่วนซอฟต์แวร์

3.1 บทนำ

ในบทนี้เราจะกล่าวถึงการออกแบบโปรแกรมโหลดเดอร์ และโปรแกรมดีบักเกอร์ ว่ามีขั้นตอนในการออกแบบและพัฒนาอย่างไร กล่าวคือ

โปรแกรมแอสเซมบลี้ เป็นโปรแกรมที่ทำหน้าที่แปลจาก ภาษาแอสเซมบลี้ไปเป็นภาษาเครื่องซึ่งในที่ก็คือภาษาเครื่องของ SAB 80C535 โดยในโครงงานนี้จะนำโปรแกรมแอสเซมเบลอร์ ของ Cross-32 Meta-Assembler ของ Universal Cross-Assemblers เป็นตัวแปลภาษาเอสแซมบลี้ เพื่อสะดวกในการใช้งานและลดเวลาที่จะมาพัฒนาขึ้นมาใช้เอง

โปรแกรมโหลดเดอร์ โปรแกรมนี้จะใช้โหลดโปรแกรมที่ได้เขียนและทำการทดสอบการทำงานแล้วว่าถูกต้องลงบนการ์ด SAB 80C535 เพื่อทดลองการทำงานจริง

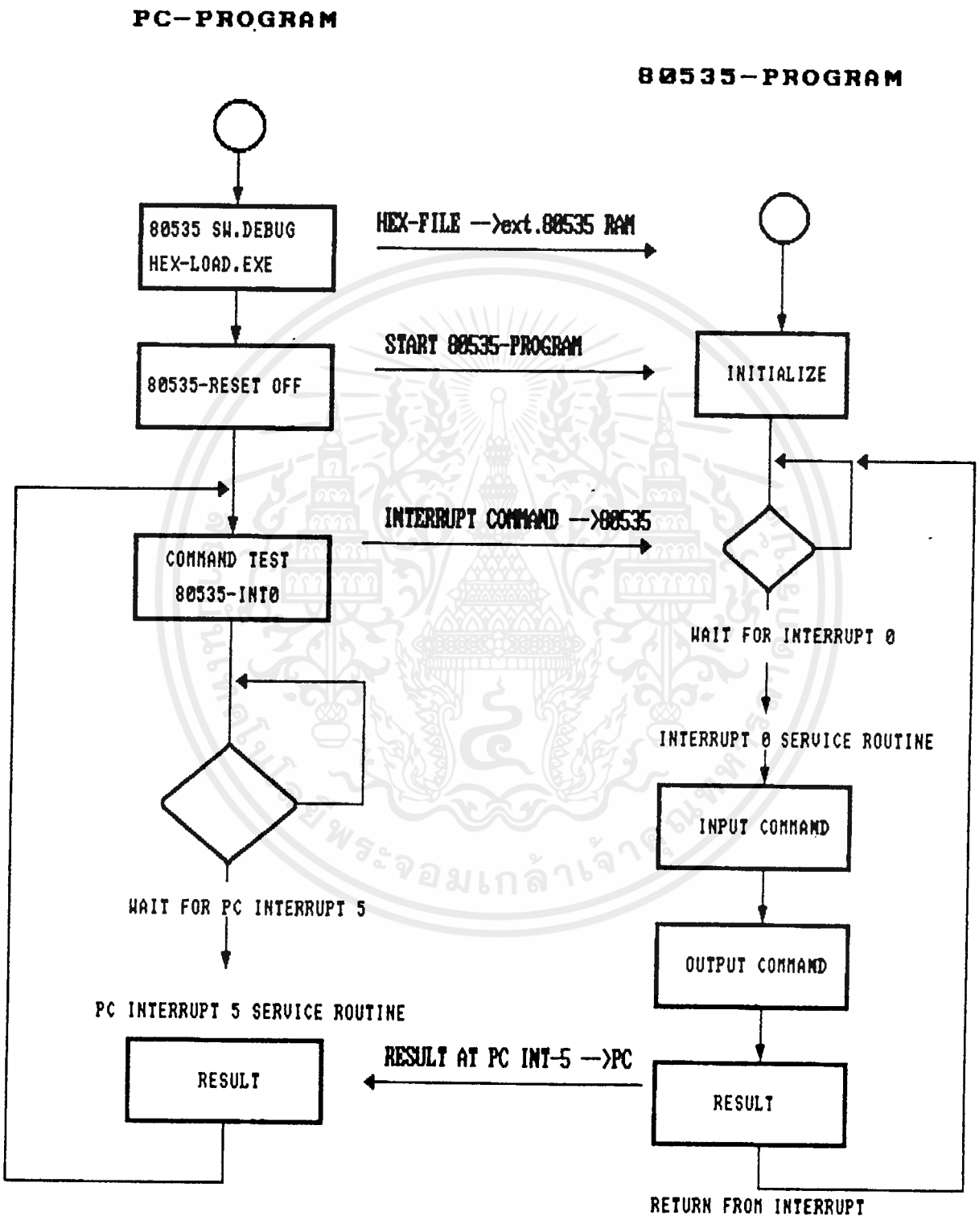
โปรแกรมดีบักเกอร์ ถือได้ว่าเป็นสิ่งจำเป็นที่จะต้องมีในระบบพัฒนา (Development System) เพราะเป็นสิ่งเดียวที่จะช่วยในการติดตามการทำงานของระบบไมโครคอลโทรเลอร์ซึ่งในการพัฒนาระบบต่าง ๆ ขึ้นมานั้นความผิดพลาดย่อมมีโอกาสเกิดขึ้นได้เสมอ ดังนั้นตัวดีบักเกอร์นี้จะเป็นเครื่องมือที่จะช่วยในการหาสาเหตุแห่งความผิดพลาดเหล่านี้ได้

3.2 ซอฟต์แวร์ดีบักเกอร์

3.2.1 แนวทางการออกแบบ

การออกแบบโปรแกรมดีบักเกอร์นั้น เราจะใช้ข้อมูลที่เป็นเลขฐานสิบหกหรือไฟล์ของภาษาแอสเซมบลีที่มีนามสกุล .HEX มาทำการดีบัก โดยหลักการของโปรแกรมดีบักเกอร์ในการติดต่อกับเครื่องไมโครคอมพิวเตอร์ รวมไปถึงการเรียกใช้การอินเตอร์รัพต์ด้วย ดังแสดงรายละเอียดในรูปแบบของ Flow chart ในรูปที่ 3.1 โดยเริ่มต้นจากการโหลดโปรแกรมโหลดเดอร์และมันจะทำให้มีการโหลด HEX ไฟล์ลงไปในหน่วยความจำของ 80C535 การ์ด และจะทำการ Reset ค่าต่างๆ ทั้งหมดที่มีอยู่ในการ์ดและในพีซี ที่อยู่ในส่วนของโปรแกรมโหลดเดอร์ ต่อจากนั้นการ์ดจะต้องรอการเกิดการอินเตอร์รัพต์ 0(INT0) เมื่อเกิดมีการอินเตอร์รัพต์ขึ้นมาพีซีก็จะทำการส่งคำสั่งให้กับการ์ด แล้ว 80C535 ก็จะกระทำตามคำสั่ง

ที่ได้มาและส่งผลกลับไปยังพีซีอีกทีหนึ่ง โดยทำการส่งอินเตอร์รัพต์ 5 (INT 5) ของ 80C535 และจะทำงานในลักษณะนี้ไปเรื่อยๆ



รูปที่ 3.1 แสดง Flow chart ของการทำงานในโปรแกรมดีบักเกอร์

การติดต่อสื่อสาร Protocol ระหว่างไมโครคอมพิวเตอร์กับการ์ด

การติดต่อกันระหว่างพีซีกับ 80C535 นั้นทำเป็นลำดับดังนี้ กล่าวคือ มันจะส่งข้อมูลคำสั่งไปให้กับ 80C535 การ์ดโดยการทำอินเตอร์ 0 หรือ 1 เมื่อทำการส่งเรียบร้องแล้วพีซีก็จะต้องคอยการตอบสนองกลับโดยการคอยรับการอินเตอร์รัพต์ 5 ของ 80C535 แต่ในระหว่างนั้นการ์ด 80C535 จะทำการประมวณผลในคำสั่งที่มันได้รับมาเมื่อทำเสร็จเรียบร้อยแล้วจึงค่อยส่งผลลัพท์ที่ได้กลับไปยังพีซีที่ยังคอยรับการอินเตอร์อยู่ซึ่งได้แสดงไว้ในรูปที่ 3.2 แล้ว

Interface-Protocol ระหว่าง PC และ 80C535 Card

Presonal Computer XT/AT	Microcontroller 80C535
1 Command Adr. 103H write	
2 Interrupt 0 (1) Adr. 100H read	
3 Wait และ Interupt 5 Adr. 103H read	1. รับ Command 2. รับ PC-Interrupt 5 3. ให้ PC_Interrupt 5

รูปที่ 3.2 แสดงการติดต่อกันระหว่าง PC กับ 80C535

โดยมีโปรแกรมภาษาแอสเซมบลี้ในส่วนของ SAB 80C535 ดังนี้

```
; SW-DEBUGGER - 80C535

IEN0                DATA    0A8H
IEN1                DATA    0B8H
IP0                 DATA    0A9H
IP1                 DATA    0B9H
IRCON               DATA    0C0H
P4                  DATA    0E8H
P5                  DATA    0F8H

PC_BYTE_1           DATA     70H
PC_BYTE_2           DATA     71H
MARKEN              DATA     72H
MARKE               DATA     73H
FLAGS_0             DATA     74H
START               CODE   1C00H
INT_0               CODE     03H
INT_1               CODE     13H
INT_6               CODE     6BH

START_ADR:
        CSEG    AT      RESET
        LJMP    START
        CSEG    AT      INT_0
        LJMP    INTERRUPT_0
        CSEG    AT      INT_1
        LJMP    INTERRUPT_0
        CSEG    AT      INT_6
        LJMP    INTERRUPT_6
        CSEG    AT      START

ORG     1C00H

PROGRAMM_START:
        MOV     R0,#15H
        MOV     DPTR,#X_USER_BREAKPOINT_1
LOOP_1: MOVX    @DPTR,A
        INC     DPTR
        DJNZ    R0,LOOP_1
LOOP_2: MOV     @R0,A
        DJNZ    R0,LOOP_2
        MOV     SP,#74H
        MOV     A,#7
        MOV     DPTR,#U_SP
        MOVX    @DPTR,A
        ORL     IP0,#20H
        ORL     IP1,#20H
        SETB    TCON.0
        ORL     IEN0,#81H
        SETB    IEN1.5
        CLR     P4.0
        SETB    P4.0
INTERRUPT_LOOP:
        SJMP    INTERRUPT_LOOP
```

รูปที่ 3.3 แสดงโปรแกรมดีบักเกอร์ในส่วนของ 80C535

```
INTERRUPT_0:
        MOV     DPTR,#X_PC_BEFEHL
        MOV     P5,#0FFH
        CLR     P4.3
        MOV     A,P5
        SETB    P4.3
        MOV     P5,A
        MOVX    @DPTR,A
        ORL     FLAGS_0,#01H
INTO_1: MOV     A,MARKEN
        JZ      INTO_BEFEHLE
        JMP     INTO_4

INTO_BEFEHLE:
        MOV     DPTR,#X_PC_BEFEHL
        MOVX    A,@DPTR
        ANL     A,#0FH
        RL      A
        MOV     DPTR,#JMP_TABELLE
        JMP     @A+DPTR
JMP_TABELLE:
        AJMP    B0_GO
        AJMP    B1_SHOW_SFR_BYTE
        AJMP    B2_PROGRAM_COUNTER_TO_PC
        AJMP    B3_SHOW_INT_RAM_BYTE
        AJMP    B4_DASM_ENDE
        AJMP    B5_SHOW_EXT_RAM_BYTE
        AJMP    B6_FOLGE_BYTE_TO_PC
        AJMP    B7_ENTER_SFR
        AJMP    B8_ENTER_INT_RAM
        AJMP    B9_ENTER_EXT_RAM
        AJMP    BA_TRACE
        AJMP    BB_BREAKPOINT_SET
        AJMP    BC_BREAKPOINT_CLEAR
        AJMP    B4_DASM_ENDE
        JMP     BEFEHL_DASM

B0_GO:  ORL     MARKEN,#01H
        JMP     B_W
B1_SHOW_SFR_BYTE:
        ORL     MARKEN,#02H
        MOV     R0,#1H
        SJMP    B_ER
B2_PROGRAM_COUNTER_TO_PC:
        ANL     FLAGS_0,#0FCH
        ORL     FLAGS_0,#0CH
        SJMP    B_ER
B3_SHOW_INT_RAM_BYTE:
        ORL     MARKEN,#04H
        SJMP    B_W
B4_DASM_ENDE:
        ANL     FLAGS_0,#0FEH
        MOV     DPTR,#X_FLAGS_1
        MOVX    A,@DPTR
```

รูปที่ 3.3 (ต่อ) แสดงโปรแกรมดีบักเกอร์ในส่วนของ 80C535

```
          CLR     A
B_Y:      MOVX    @DPTR,A
          SJMP    B_ER
B5_SHOW_EXT_RAM_BYTE:
          ORL     MARKEN,#08H
          SJMP    B_W
B6_FOLGE_BYTE_TO_PC:
          ANL     FLAGS_0,#0FEH
          SJMP    B_ER
B7_ENTER_SFR:
          ORL     MARKEN,#10H
B_W:      MOV     R0,#2H
B_ER:     JMP     INT0_RETURN
B8_ENTER_INT_RAM:
          ORL     MARKEN,#20H
          SJMP    B_X
B9_ENTER_EXT_RAM:
          ORL     MARKEN,#40H
B_X:      MOV     R0,#3H
          SJMP    B_ER
BA_TRACE:
          ANL     FLAGS_0,#0FEH
          JMP     BEFEHL_TRACE
BB_BREAKPOINT_SET:
          ORL     MARKEN,#80H
          SJMP    B_W
BC_BREAKPOINT_CLEAR:
          ANL     FLAGS_0,#0FEH
          MOV     A,MARKE
          JZ      B_ER
          JMP     BEFEHL_BREAKPOINT_CLEAR

INTO_4:   JB      ACC.0,INTO_M0_GO
          JB      ACC.1,INTO_M1_SHOW_SFR_BYTE
          JB      ACC.2,INTO_M2_SHOW_INT_RAM_BYTE
          JB      ACC.3,INTO_M3_SHOW_EXT_RAM_BYTE
          JB      ACC.4,INTO_M4_ENTER_SFR
          JB      ACC.5,INTO_M5_ENTER_INT_RAM
          JB      ACC.6,INTO_M6_ENTER_EXT_RAM
          JMP     INTO_M7_BREAKPOINT_SET

INTO_M0_GO:
          ACALL   PC_BEFEHL_EXT_LESEN
          ANL     MARKEN,#0FEH
          ORL     FLAGS_0,#20H
          JMP     BEFEHL_GO
INTO_M1_SHOW_SFR_BYTE:
          ACALL   PC_BEFEHL_EXT_LESEN
          ANL     MARKEN,#0FDH
          JMP     BEFEHL_SHOW_SFR_BYTE
INTO_M2_SHOW_INT_RAM_BYTE:
          ORL     FLAGS_0,#02H
          ACALL   PC_BEFEHL_EXT_LESEN
          ANL     MARKEN,#0FBH
          JMP     BEFEHL_SHOW_INT_RAM_BYTE
```

รูปที่ 3.3 (ต่อ) แสดงโปรแกรมดีบักเกอร์ในส่วนของ 80C535

```
INTO_M3_SHOW_EXT_RAM_BYTE:
        ORL     FLAGS_0,#02
        ACALL   PC_BEFEHL_EXT_LESEN
        ANL     MARKEN,#0F7H
        JMP     BEFEHL_SHOW_EXT_RAM_BYTE
INTO_M4_ENTER_SFR:
        ACALL   PC_BEFEHL_EXT_LESEN
        ANL     MARKEN,#0EFH
        JMP     BEFEHL_ENTER_SFR
INTO_M5_ENTER_INT_RAM:
        ACALL   PC_BEFEHL_EXT_LESEN
        ANL     MARKEN,#0DFH
        JMP     BEFEHL_ENTER_INT_RAM
INTO_M6_ENTER_EXT_RAM:
        ACALL   PC_BEFEHL_EXT_LESEN
        ANL     MARKEN,#0BFH
        JMP     BEFEHL_ENTER_EXT_RAM
INTO_M7_BREAKPOINT_SET:
        ACALL   PC_BEFEHL_EXT_LESEN
        ANL     MARKEN,#7FH
        JMP     BEFEHL_BREAKPOINT_SET

PC_BEFEHL_EXT_LESEN:
        MOV     DPTR,#X_PC_BEFEHL
        MOVX    A,@DPTR
        CJNE    R0,#1,PC_BYTE_2_3_LESEN
PC_BYTE_1_LESEN:
        MOV     PC_BYTE_1,A
        ANL     FLAGS_0,#0FEH
        SJMP    RETURN_PC_BYTE_LESEN
PC_BYTE_2_3_LESEN:
        CJNE    R0,#2H,PC_BYTE_3_LESEN
        MOV     PC_BYTE_2,A
        SJMP    RETURN_PC_BYTE_LESEN
PC_BYTE_3_LESEN:
        MOV     DPTR,#X_PC_BYTE_3
        MOVX    @DPTR,A
RETURN_PC_BYTE_LESEN:
        DEC     R0
        MOV     A,FLAGS_0
        JNB     ACC.0,NO_INT_RET
        DEC     SP
        DEC     SP
        JMP     INTO_RETURN
NO_INT_RET:
        RET

BEFEHL_SHOW_SFR_BYTE:
        ORL     FLAGS_0,#02
        MOV     A,PC_BYTE_1
        MOV     DPTR,#BSS_10 + 1
        MOVX    @DPTR,A
        CJNE    A,#0E0H,BSS_2
        MOV     DPTR,#U_ACC
BSS_W:  MOVX    A,@DPTR
```


รูปที่ 3.3 (ต่อ) แสดงโปรแกรมดีบักเกอร์ในส่วนของ 80C535

```
            MOV     PC_BYTE_1,A
            SJMP    BSS_E
BSS_2:      CJNE    A,#0D0H,BSS_4
            MOV     DPTR,#U_PSW
            SJMP    BSS_W
BSS_4:      CJNE    A,#082H,BSS_6
            MOV     DPTR,#U_DPL
            SJMP    BSS_W
BSS_6:      CJNE    A,#083H,BSS_8
            MOV     DPTR,#U_DPH
            SJMP    BSS_W
BSS_8:      CJNE    A,#081H,BSS_10
            MOV     DPTR,#U_SP
            SJMP    BSS_W
BSS_10:     MOV     PC_BYTE_1,0
BSS_E:      JMP     INTO_RETURN

BEFEHL_SHOW_INT_RAM_BYTE:
            MOV     R0,PC_BYTE_1
            MOV     A,R0
            JZ      BSI_4
            MOV     PC_BYTE_1,@R0
BSI_2:      JMP     INTO_RETURN
BSI_4:      MOV     DPTR,#U_R0
BSI_6:      MOVX    A,@DPTR
            MOV     PC_BYTE_1,A
            SJMP    BSI_2

BEFEHL_SHOW_EXT_RAM_BYTE:
            MOV     DPL,PC_BYTE_1
            MOV     DPH,PC_BYTE_2
            SJMP    BSI_6

BEFEHL_ENTER_SFR:
            MOV     A,PC_BYTE_2
            MOV     R0,A
            MOV     DPTR,#BES_10 + 1
            MOVX    @DPTR,A
            MOV     A,PC_BYTE_1
            CJNE    R0,#0E0H,BES_2
            MOV     DPTR,#U_ACC
BES_W:      MOVX    @DPTR,A
            SJMP    BES_E
BES_2:      CJNE    R0,#0D0H,BES_4
            MOV     DPTR,#U_PSW
            SJMP    BES_W
BES_4:      CJNE    R0,#082H,BES_6
            MOV     DPTR,#U_DPL
            SJMP    BES_W
BES_6:      CJNE    R0,#083H,BES_8
            MOV     DPTR,#U_DPH
            SJMP    BES_W
BES_8:      CJNE    R0,#081H,BES_10
            MOV     DPTR,#U_SP
            SJMP    BES_W
```


รูปที่ 3.3 (ต่อ) แสดงโปรแกรมดีบักเกอร์ในส่วนของ 80C535

```
BES_10: MOV     0H,A
BES_E:  JMP     INTO_RETURN

BEFEHL_ENTER_INT_RAM:
        MOV     R0,PC_BYTE_2
        MOV     A,PSW
        ANL     A,#18H
        CJNE    A,PC_BYTE_2,BEI_2
        MOV     A,PC_BYTE_1
        MOV     DPTR,#U_R0
        MOVX    @DPTR,A
        SJMP    BEI_4
BEI_2:  MOV     A,PC_BYTE_1
        MOV     @R0,A
BEI_4:  JMP     INTO_RETURN

BEFEHL_ENTER_EXT_RAM:
        MOV     DPTR,#X_PC_BYTE_3
        MOVX    A,@DPTR
        MOV     DPH,A
        MOV     DPL,PC_BYTE_2
        MOV     A,PC_BYTE_1
        MOVX    @DPTR,A
        JMP     INTO_RETURN

BEFEHL_BREAKPOINT_SET:
        MOV     A,MARKE
        ANL     MARKE,#80H
        JZ      SAVE_BREAK_ADR
        ACALL   RECHANGE_BREAK_USER_BYTES
SAVE_BREAK_ADR:
        MOV     DPTR,#X_PROGRAM_COUNTER_H
        MOV     A,PC_BYTE_2
        MOVX    @DPTR,A
        INC     DPTR
        MOV     A,PC_BYTE_1
        MOVX    @DPTR,A
CHANGE_BREAK_USER_BYTES:
        MOV     DPL,PC_BYTE_1
        MOV     DPH,PC_BYTE_2
        MOVX    A,@DPTR
        PUSH    ACC
        MOV     A,#02H
        MOVX    @DPTR,A
        INC     DPTR
        MOVX    A,@DPTR
        PUSH    ACC
        PUSH    DPH
        PUSH    DPL
        MOV     DPTR,#BREAKPOINT_ERREICHT
        MOV     R0,DPL
        MOV     A,DPH
        POP     DPL
        POP     DPH
        MOVX    @DPTR,A
```

รูปที่ 3.3 (ต่อ) แสดงโปรแกรมดีบักเกอร์ในส่วนของ 80C535

```
        INC     DPTR
        MOVX    A,@DPTR
        PUSH    ACC
        MOV     A,R0
        MOVX    @DPTR,A
        MOV     DPTR,#X_U_BYTE_3
        POP     ACC
        MOVX    @DPTR,A
        INC     DPTR
        POP     ACC
        MOVX    @DPTR,A
        INC     DPTR
        POP     ACC
        MOVX    @DPTR,A
        ORL     MARKE,#80H
        JMP     INTO_RETURN

RECHANGE_BREAK_USER_BYTES:
        MOV     DPTR,#X_U_BYTE_3
        MOVX    A,@DPTR
        PUSH    ACC
        INC     DPTR
        MOVX    A,@DPTR
        PUSH    ACC
        INC     DPTR
        MOVX    A,@DPTR
        PUSH    ACC
        MOV     DPTR,#X_PROGRAM_COUNTER_H
        ACALL   SUBSTITUTE
        POP     ACC
        MOVX    @DPTR,A
        INC     DPTR
        POP     ACC
        MOVX    @DPTR,A
        INC     DPTR
        POP     ACC
        MOVX    @DPTR,A
        ANL     MARKE,#07FH
        RET

BREAKPOINT_ERREICHT:
        ACALL   SAVE_USER_REGISTER
        MOV     A,SP
        MOVX    @DPTR,A
        MOV     DPTR,#D_PSW
        MOVX    A,@DPTR
        MOV     PSW,A
        INC     DPTR
        MOVX    A,@DPTR
        MOV     SP,A
        ACALL   RECHANGE_BREAK_USER_BYTES
        JMP     INTO_RETURN

SAVE_USER_REGISTER:
        PUSH    ACC
```


รูปที่ 3.3 (ต่อ) แสดงโปรแกรมดีบักเกอร์ในส่วนของ 80C535

```
        PUSH    DPH
        MOV     A,DPL
        MOV     DPTR,#U_DPL
        MOVX    @DPTR,A
        POP     ACC
        INC     DPTR
        MOVX    @DPTR,A
        MOV     A,PSW
        INC     DPTR
        MOVX    @DPTR,A
        POP     ACC
        INC     DPTR
        MOVX    @DPTR,A
        MOV     DPTR,#U_R0
        MOV     A,R0
        MOVX    @DPTR,A
        INC     DPTR
        RET

BEFEHL_TRACE:
        MOV     DPTR,#X_PROGRAM_COUNTER_L
        MOVX    A,@DPTR
        ADD     A,#-2
        MOVX    @DPTR,A
        MOV     PC_BYTE_1,A
        MOV     DPTR,#X_PROGRAM_COUNTER_H
        MOVX    A,@DPTR
        ADDC    A,#0FFH
        MOVX    @DPTR,A
        MOV     PC_BYTE_2,A
        ACALL   CHANGE_INT6_CODE_MIT_USER_BYTES

BEFEHL_GO:
        MOV     DPTR,#GO_JMP + 1
        MOV     A,PC_BYTE_2
        MOVX    @DPTR,A
        INC     DPTR
        MOV     A,PC_BYTE_1
        MOVX    @DPTR,A

        MOV     DPTR,#D_PSW
        MOV     A,PSW
        MOVX    @DPTR,A
        INC     DPTR
        MOV     A,SP
        MOVX    @DPTR,A

REG_USER_REGISTER:
        MOV     DPTR,#U_PSW
        MOVX    A,@DPTR
        MOV     PSW,A
        MOV     DPTR,#U_R0
        MOVX    A,@DPTR
        MOV     R0,A
        INC     DPTR
```


รูปที่ 3.3 (ต่อ) แสดงโปรแกรมดีบักเกอร์ในส่วนของ 80C535

```
        MOVX    A,@DPTR
        PUSH    ACC
        MOV     DPTR,#U_ACC
        MOVX    A,@DPTR
        PUSH    ACC
        MOV     DPTR,#U_DPL
        MOVX    A,@DPTR
        PUSH    ACC
        INC     DPTR
        MOVX    A,@DPTR
        MOV     DPH,A
        POP     ACC
        MOV     DPL,A
        POP     ACC
        POP     SP
GO_JMP: JMP     0

SUBSTITUTE:
        MOVX    A,@DPTR
        MOV     R0,A
        INC     DPTR
        MOVX    A,@DPTR
        MOV     DPL,A
        MOV     DPH,R0
        RET

BEFEHL_BREAKPOINT_CLEAR:
        ACALL   RECHANGE_BREAK_USER_BYTES
        JMP     INTO_RETURN

CHANGE_INT6_CODE_MIT_USER_BYTES:
        MOV     DPL,PC_BYTE_1
        MOV     DPH,PC_BYTE_2
        MOVX    A,@DPTR
        PUSH    DPH
        PUSH    DPL
        MOV     DPTR,#X_USER_BREAKPOINT_1
        MOVX    @DPTR,A
        POP     DPL
        POP     DPH
        MOV     A,#0D2H
        MOVX    @DPTR,A
        INC     DPTR
        MOVX    A,@DPTR
        PUSH    DPH
        PUSH    DPL
        MOV     DPTR,#X_USER_BREAKPOINT_2
        MOVX    @DPTR,A
        POP     DPL
        POP     DPH
        MOV     A,#0C5H
        MOVX    @DPTR,A
        RET

RECHANGE_USER_BYTES:
```

รูปที่ 3.3 (ต่อ) แสดงโปรแกรมดีบักเกอร์ในส่วนของ 80C535

```
                MOV     DPTR,#X_PROGRAM_COUNTER_H; RESET_USE
                ACALL   SUBSTITUTE
                PUSH    DPL
                PUSH    DPH
                MOV     DPTR,#X_USER_BREAKPOINT_1
                MOVX    A,@DPTR
                POP     DPH
                POP     DPL
                MOVX    @DPTR,A
                INC     DPTR
                PUSH    DPL
                PUSH    DPH
                MOV     DPTR,#X_USER_BREAKPOINT_2
                MOVX    A,@DPTR
                POP     DPH
                POP     DPL
                MOVX    @DPTR,A
                RET

BEFEHL_DASM:
                MOV     DPTR,#X_PROGRAM_COUNTER_H
                ACALL   SUBSTITUTE
                MOVX    A,@DPTR
                PUSH    ACC
                INC     DPTR
                MOVX    A,@DPTR
                PUSH    ACC
                INC     DPTR
                MOVX    A,@DPTR
                MOV     DPTR,#X_MC_BYTE_3
                MOVX    @DPTR,A
                INC     DPTR
                POP     ACC
                MOVX    @DPTR,A
                INC     DPTR
                POP     ACC
                MOVX    @DPTR,A

                ANL     FLAGS_0,#0F0H
                MOV     DPTR,#X_FLAGS_1
                MOVX    A,@DPTR
                ORL     A,#0FH
                MOVX    @DPTR,A

INTO_RETURN:
                MOV     A,FLAGS_0
                JB      ACC.0,PC_5_RETURN
                JBC     ACC.1,PC_5_2
                JBC     ACC.2,PC_5_4
                JBC     ACC.3,PC_5_6
                MOV     DPTR,#X_FLAGS_1
                MOVX    A,@DPTR
                JB      ACC.0,PC_5_8
                SJMP    PC_5_RETURN
PC_5_2:         MOV     FLAGS_0,A
```


รูปที่ 3.3 (ต่อ) แสดงโปรแกรมดีบักเกอร์ในส่วนของ 80C535

```
            MOV     A,PC_BYTE_1
            SJMP    PC_5_RETURN
PC_5_4:     MOV     FLAGS_0,A
            MOV     DPTR,#X_PROGRAM_COUNTER_H
            JMP     PC_5_7
PC_5_6:     MOV     FLAGS_0,A
            MOV     DPTR,#X_PROGRAM_COUNTER_L
PC_5_7:     MOVX    A,@DPTR
            SJMP    PC_5_RETURN
PC_5_8:     JBC     ACC.1,PC_5_A
            JBC     ACC.2,PC_5_C
            JBC     ACC.3,PC_5_E
            SJMP    PC_5_RETURN
PC_5_A:     MOVX    @DPTR,A
            MOV     DPTR,#X_MC_BYTE_1
            SJMP    PC_5_7
PC_5_C:     MOVX    @DPTR,A
            MOV     DPTR,#X_MC_BYTE_2
            SJMP    PC_5_7
PC_5_E:     MOVX    @DPTR,A
            MOV     DPTR,#X_MC_BYTE_3
            SJMP    PC_5_7
PC_5_RETURN:
            MOV     P5,A
            CLR     P4.2
            SETB    P4.2
            ANL     P4,#0FCH
            ORL     P4,#003H
            RETI

INTERRUPT_6:
            ACALL   SAVE_USER_REGISTER
            MOV     A,SP
            ADD     A,#-2
            MOVX    @DPTR,A
            ACALL   RECHANGE_USER_BYTES
            MOV     DPTR,#X_PROGRAM_COUNTER_H
            POP     ACC
            MOVX    @DPTR,A
            INC     DPTR
            POP     ACC
            MOVX    @DPTR,A

            MOV     DPTR,#INT6_ZURUECK_VOM_TRACE
            PUSH    DPL
            PUSH    DPH
            RETI

INT6_ZURUECK_VOM_TRACE:
            MOV     DPTR,#D_PSW
            MOVX    A,@DPTR
            MOV     PSW,A
            INC     DPTR
            MOVX    A,@DPTR
            MOV     SP,A
```

รูปที่ 3.3 (ต่อ) แสดงโปรแกรมดีบักเกอร์ในส่วนของ 80C535

```
                JMP     PC_5_RETURN

X_USER_BREAKPOINT_1:    NOP
X_USER_BREAKPOINT_2:    NOP
X_PROGRAM_COUNTER_H:    NOP
X_PROGRAM_COUNTER_L:    NOP
X_PC_BEFEHL:            NOP
X_PC_BYTE_3:            NOP
X_MC_BYTE_3:            NOP
X_MC_BYTE_2:            NOP
X_MC_BYTE_1:            NOP
X_FLAGS_1:              NOP
X_U_BYTE_3:             NOP
X_U_BYTE_2:             NOP
X_U_BYTE_1:             NOP
U_R0:                   NOP
U_SP:                   NOP
U_DPL:                  NOP
U_DPH:                  NOP
U_PSW:                  NOP
U_ACC:                  NOP
D_PSW:                  NOP
D_SP:                   NOP

 END
```

รูปที่ 3.3 (ต่อ) แสดงโปรแกรมดีบักเกอร์ในส่วนของ 80C535

3.2.2 การทำงานของโปรแกรม

ในส่วนนี้จะได้กล่าวถึงการใช้งานกับโปรแกรมดีบักเกอร์ของ การ์ด 80C535 นี้ ซึ่งคำสั่งต่างๆ ที่ใช้นั้นจะเป็นคำสั่งของไมโครคอลโทรเลอร์ในตระกูล MCS 51 ซึ่งโปรแกรมดีบักเกอร์นี้เป็นโปรแกรมสำหรับช่วยในการทดสอบและพัฒนาซอฟต์แวร์ขึ้นบนการ์ด 80C535 โดยมีรายละเอียดดังต่อไปต่อไปนี้

- สามารถเรียกดูข้อมูลที่อยู่ใน หน่วยความจำภายในและภายนอกของ RAM และข้อมูลที่อยู่ในรีจิสเตอร์พิเศษ
- สามารถทำการ RUN และแก้ไขโปรแกรมได้สะดวก
- สามารถทำการ ดีสแอสเซมเบลอร์ได้
- สามารถกำหนดค่าของ Break Point ได้
- สามารถใช้คำสั่ง GO และทำการ Trace ข้อมูลได้

ตัวโปรแกรมของซอฟต์แวร์ดีบักเกอร์นั้นจะกินเนื้อที่ของหน่วยความจำบนเครื่องไอบีเอ็ม พีซี น้อยมากคือเนียงประมาณแค่ 1 กิโลไบท์ เท่านั้น และคุณลักษณะโดยทั่วไปมีดังต่อไปนี้

- ใช้เนื่อที่ในส่วนของหน่วยความจำภายใน โดยกำหนดตั้งแต่แอสเดรสที่ 70H - 7FH
- ใช้เนื้อที่ในส่วนของหน่วยความจำภายนอก โดยกำหนดตั้งแต่แอสเดรสที่ 1C00H - 1FFFH
- ใช้อินเตอร์รัพต์ 6 ในการ Trace ข้อมูลล่วงหน้า
- การติดคต่อระหว่างเครื่องไอบีเอ็ม พีซี กับการ์ดของ 80C535 นี้จะใช้สัญญาณอินเตอร์รัพต์ จากการ์ด 80C535 และอินเตอร์รัพต์ 5 (หรือ 3 ,4 และ 7)

3.2.3 คำสั่งการใช้งานของดีบักเกอร์

OPTIONS เป็นคำสั่งที่ทำหน้าที่เกี่ยวกับไฟล์ต่างๆ และการเข้าออกของโปรแกรม

GO เป็นคำสั่งที่ใช้ในการ RUN โปรแกรม

BREAKPOINT เป็นการตั้งเพื่อให้ โปรแกรมหยุดทำงานตรงตำแหน่งที่เรา

ต้องการ

TRACE เป็นคำสั่งที่ให้โปรแกรมติดตามดูการกระทำของโปรแกรมไปเรื่อยๆ ตลอดคำสั่ง

DISASM เป็นคำสั่งในการให้ทำการแปลงโปรแกรมที่เป็น HEX-File ให้เป็นภาษาแอสเซมบลี้

SHOW เป็นคำสั่งให้โปรแกรมแสดงค่าต่าง ๆ ที่มีอยู่ในซีพียู เช่น ค่าที่อยู่ในหน่วยความจำ ณ. จุดใดจุดหนึ่ง หรือดูค่าของรีจิสเตอร์ชนิดพิเศษ

ENTER เป็นคำสั่งที่ทำหน้าที่ป้อนข้อมูลเข้าสู่หน่วยความจำ (RAM)

OPTIONS

LOAD ทำการโหลดโปรแกรมที่มีนามสกุล "HEX" ลงในหน่วยความของไอบีเอ็ม พีซี

END ทำการจบโปรแกรมดีบักเกอร์

RESET ทำการเรียกโปรแกรมดีบักเกอร์ใหม่อีกครั้ง

COLOR ทำการเปลี่ยนสีบนหน้าจอที่รันโปรแกรมดีบักเกอร์ใหม่

GO

BREAKPOINT

SET ทำการตั้งค่าแอสเดรสขนาด 16 บิท ของ Breakpoint

CLEAR ทำการลบค่า Breakpoint ที่ได้ตั้งไว้ก่อนหน้านี้ออกจากหน่วยความจำ

SHOW ทำการแสดงค่า Breakpoint ที่ได้ตั้งไว้ก่อนหน้านี้ออกจากหน่วยความจำ

TRACE

DISASM

SHOW

SFR ทำการแสดงข้อมูลภายใน รีจิสเตอร์หน้าที่พิเศษเป็นตารางออกมาให้เห็น

INT.RAM ทำการแสดงแอสเดรสที่อยู่ของหน่วนความจำภายใน

EXT.RAM ทำการแสดงแอสเดรสที่อยู่ของหน่วนความจำภายใน

SFR Blk ทำการแสดงขอบเขตของ รีจิสเตอร์หน้าที่พิเศษออกมา

Int.Blk ทำการแสดงขอบเขตของหน่วยความจำ (RAM) ภายในของการ์ด 80C535

Ext.Blk ทำการแสดงหน่วยความจำ (RAM) 256 ไบท์ ที่อยู่ภายนอกการ์ด 80C535

ENTER

SFR ทำการใส่ค่า 8 ไบท์ ลงในรีจิสเตอร์หน้าที่พิเศษ

INT.RAM ทำการใส่ค่า 8 ไบท์ ลงในหน่วยความจำ(RAM) ภายในของการ์ด 80C535

Ext.RAM ทำการใส่ค่า 8 ไบท์ ลงในหน่วยความจำ(RAM) ภายนอกของการ์ด 80C535

บทที่ 4

บทสรุป

4.1 สรุปผลการทดลอง

เนื่องจาเไปโครคอลโทรเลอร์ SAB 80C535 นั้นเป็นไมโครคอลเลอร์ ที่มีความสามารถสูงถ้าสามารถนำไปใช้ในการทำงานทางด้านการควบคุมจะมีประโยชน์ แต่เนื่องจากยังไม่มีเครื่องมือที่ช่วยเหลือ (TOOL) ในการพัฒนาระบบต่าง ๆ ที่จะนำเอาไมโครคอลโทรเลอร์ตัวนี้มาใช้ ดังนั้นโครงงานนี้จึงเป็นการจัดสร้างระบบที่ช่วยเหลือในการพัฒนาระบบที่ใช้ไมโครคอลโทรเลอร์ SAB 80C535 เพื่ออำนวยความสะดวกในการที่จะนำไปใช้พัฒนาระบบที่ใช้ไมโครคอลโทรเลอร์ SAB 80C535 เป็นตัวประมวลต่อไป

ในโครงงานนี้เราได้สร้างการ์ดขึ้นมาใช้ SAB 80C535 เป็นซีพียู และมีวงจรอื่นๆ ประกอบด้วย เช่นหน่วยความจำ วงจรอินเตอร์เฟส วงจรควบคุมการทำการอ้างแอสเดรส วงจรมัลติเพล็กซ์ วงจรควบคุมบัสและบัฟเฟอร์ ซึ่งจะสามารถใช้การได้จริง โดยควบคุมการทำงานจาก ไอบีเอ็ม พีซี ดังนั้นในโครงงานนี้จึงต้องเขียนส่วนซอฟร์แวร์บนเครื่องไอบีเอ็ม พีซี เพื่อทำงานร่วมกับวงจรบนการ์ดด้วยซึ่งจะประกอบด้วย โปรแกรมโหลดเดอร์ และโปรแกรมดีบักเกอร์

4.2 ปัญหาที่เกิดขึ้น

เนื่องจากตัวไมโครคอลโทรเลอร์ SAB 80C535 เป็นซีพียูที่ค่อนข้างใหม่และสลับซับซ้อนมากดังนั้นจึงต้องใช้เวลาในช่วงเทอมแรำทำการศึกษาการทำงานโดยละเอียด ก่อนที่จะทำการออกแบบวงจร ทำให้เสียเวลาในการทดลองและวินิจฉัยข้อผิดพลาด การสร้างการ์ดจะต้องสร้างวงจรย่อยต่างๆ ขึ้นมาประกอบกันเป็นระบบแต่ละวงจรย่อยจะต้องถูกออกแบบและสร้องขึ้นมาให้สัมพันธ์กัววงจรย่อยอื่นๆ เพื่อให้การ์ดสามารถทำงานได้ ซึ่งจากการทดสอบนั้น ปรากฏว่าในส่วนวงจรที่ทำหน้าที่ติดต่อกับเครื่องไอบีเอ็ม พีซี นั้นสามารถทำงานได้โดยการทดสอบส่งข้อมูลออกไปทางพอร์ต และอ่านข้อมูลกลับเข้ามาตรวจสอบและอีกส่วนหนึ่งที่ทำการทดสอบ คือวงจรส่วนควบคุมการแลกเปลี่ยนข้อมูล ซึ่งการทดสอบวงจรส่วนนี้จะต้องเขียนโปรแกรมโหลดเดอร์ขึ้นมาก่อน และทดสอบโยดการลองการส่งข้อมูลไปไวในหน่วยความจำบนการ์ด และทดลองให้ส่งข้อมูลกลับมาตรวจสอบว่าถูกต้องหรือไม่ ซึ่ง

ในครั้งแรกนั้นปรากฏว่าข้อมูลที่ทำการส่งไป แล้วอ่านกลับมานั้นมีค่าไม่ตรงจึงทำการตรวจสอบแก้ไขในส่วนมาก แต่ก็มีบางครั้งที่ยังมีปัญหาข้อมูลบางไบท์มีค่าไม่ตรงและเมื่อทดลองโหลดโปรแกรมที่เป็นออบเจกโค้ดของ SAB 80C535 ไปรัน ปรากฏว่ายังไม่สามารถทำงานให้ถูกต้องได้ ซึ่งจะต้องทำการตรวจสอบและแก้ไปต้อไป แต่เนื่องด้วยเวลาที่ทำโครงงานนี้มีเวลาที่จำกัด และเสียเวลาในการศึกษาและตรวจสอบแก้ไขในส่วนของฮาร์ดแวร์จึงทำให้ผลงานในโครงงานนี้เสร็จไม่สมบูรณ์ตามเป้าหมายดีนัก

ปัญหาที่เกิด โดยส่วนมากเป็นปัญหาที่เกิดจาก การหาอุปกรณ์ไม่ได้ตามที่ต้องการ เพราะอุปกรณ์ที่ต้องใช้นั้นต้องเป็นอุปกรณ์ที่มีประสิทธิภาพสูง เพื่อให้ทำงานประกอบกับการทำงานให้ได้สมบรูณ์ที่สุด แต่ก็ยังเกิดปัญหาที่ทำให้ตัวซีพียูมีความร้อนมาก

อย่างไรก็ตาม โครงงานนี้เป็นเพียงส่วนหนึ่ง ซึ่งสามารถใช้เป็นแนวทางในการพัฒนาระบบ ให้สมบูรณ์ยิ่งขึ้นเพราะจะช่วยแก้ไขปัญหาข้างต้นได้ แต่การใช้งานจริงอาจไม่สะดวกนัก ดังนั้นโครงงานนี้จึงต้องการพัฒนาตามรูปแบบและจุดประสงค์ของผู้ต้องการนำไปใช้งาน

กิตติกรรมประกาศ

ปริญญานิพนธ์ฉบับนี้สำเร็จลุล่วงไปด้วยดี ด้วยความร่วมมือจากหลายฝ่ายเป็นอย่างดีผู้จัดทำขอขอบคุณ G.E.P. co.,ltd. ที่ให้คำปรึกษาและได้เอื้อเฟื้อสถานที่ อุปกรณ์ และเครื่องไปโครคอมพิวเตอร์ พีซี และเครื่องมือวัดต่างๆ

นอกจากนี้ ผู้จัดทำขอขอบคุณอย่างยิ่งสำหรับ อาจารย์ วิทยา ทิพย์สุวรรณพร ที่เป็นอาจารย์ที่ปรึกษาให้โครงงานนี้สำเร็จลุล่วงไปได้ด้วยดี และขอบคุณเพื่อนๆ ที่ได้ให้ความช่วยเหลือในงานด้านต่างๆ

จึงขอขอบคุณไว้ ณ. โอกาสนี้ด้วย

ผู้จัดทำ

หนังสืออ้างอิง

1. John B. Peatman, "Design with Microcontrollers",McGraw-Hall International Editions 1988.
2. Lewis C. Eggebrecht,"INTERFACING TO THE IBM PERSONAL COMPUTER" ,Howard W. Sams & Co., Inc. 1983.
3. "MCS-51 Microcontrollers", ETT co,.ltd.
4. Douglas V. Hall, "MICROPROCESSORS AND INTERFACING Programming and Hardware", McGraw-Hill Book Co., 1986
5. Rodnay Zaks and Austin Lesea, "MICROPROCESSOR INTERFACING TECHNIQUES", Third Edition
6. Microsoft Corporation, "Microsoft Marco Assemly 6.0" 1991
7. Microsoft Corporation, "Microsoft C Compiler Version 7.0" 1992
8. ชูชัย ธนสารตั้งเจริญ, "การใช้งาน Z80", ศูนย์ภาษาคอมพิวเตอร์
9. ปีเตอร์ นอร์ตัน และ จอห์น โซซา, "เทคนิคการเขียนโปรแกรม ภาษาแอสเซมบลี้ สำหรับเครื่อง IBM PC, 1989

ภาคผนวก ก

ชุดคำสั่งของ SAB 80C535

Befehlscode

0	NOP	20	JB badr,rel	40	JC rel
1	AJMP page 0	21	AJMP page 1	41	AJMP page 2
2	LJMP adr16	22	RET	42	ORL dadr,A
3	RR A	23	RL A	43	ORL dadr,#k8
4	INC A	24	ADD A,#k8	44	ORL A,#k8
5	INC adr16	25	ADD A,dadr	45	ORL A,dadr
6	INC @R0	26	ADD A,@R0	46	ORL A,@R0
7	INC @R1	27	ADD A,@R1	47	ORL A,@R1
8	INC R0	28	ADD A,R0	48	ORL A,R0
9	INC R1	29	ADD A,R1	49	ORL A,R1
A	INC R2	2A	ADD A,R2	4A	ORL A,R2
B	INC R3	2B	ADD A,R3	4B	ORL A,R3
C	INC R4	2C	ADD A,R4	4C	ORL A,R4
D	INC R5	2D	ADD A,R5	4D	ORL A,R5
E	INC R6	2E	ADD A,R6	4E	ORL A,R6
F	INC R7	2F	ADD A,R7	4F	ORL A,R7
10	JBC basd,rel	30	JNB badr,rel	50	JNC rel
11	ACALL page 0	31	ACALL page 1	51	ACALL apge 2
12	LCALL adr16	32	RETI	52	ANL dadr,A
13	DEC	33	RLC A	53	ANL dadr,#k8
14	DEC	34	ADDC A,#k8	54	ANL A,#k8
15	DEC	35	ADDC A,dadr	55	ANL A,dadr
16	DEC	36	ADDC A,@R0	56	ANL A,@R0
17	DEC	37	ADDC A,@R1	57	ANL A,@R1
18	DEC	38	ADDC A,R0	58	ANL A,R0
19	DEC	39	ADDC A,R1	59	ANL A,R1
1A	DEC	3A	ADDC A,R2	5A	ANL A,R2
1B	DEC	3B	ADDC A,R3	5B	ANL A,R3
1C	DEC	3C	ADDC A,R4	5C	ANL A,R4
1D	DEC	3D	ADDC A,R5	5D	ANL A,R5
1E	DEC	3E	ADDC A,R6	5E	ANL A,R6
1F	DEC	3F	ADDC A,R7	5F	ANL A,R7

page = 2 K Seite
adr = 8 Bit Adresse
#k8 = 8 Bit Konstante
#K16 = 16- Bit Konstante
rel = relative Adresse = vorzeichenbehafteter 8 Bit Wert, bezogen auf das erste Bbyte des folgenden Befehls

badr = Bitadresse
dadr = Datenadresse
@ = Indirekte Adresse

60	JZ rel	80	SJMP rel	A0	ORL C,badr\
61	AJMP page 3	81	AJMP page 4	A1	AJMP page 5
62	XRL dadr	82	ANL C,badr	A2	MOVC,badr
63	XRL dadr,#k8	83	MOVC A,@A+PC	A3	INC DPTR
64	XRL A,#k8	84	DIV AB	A4	MUL AB
65	XRL A,dadr	85	MOV dadr,dadr	A5	undefiniert
66	XRL A,@R0	86	MOV dadr,@R0	A6	MOV @R0,dadr
67	XRL A,@R1	87	MOV dadr,@R1	A7	MOV @R1,dadr
68	XRL A,R0	88	MOV dadr,R0	A8	MOV R0,dadr
69	XRL A,R1	89	MOV dadr,R1	A9	MOV R1,dadr
6A	XRL A,R2	8A	MOV dadr,R2	AA	MOV R2,dadr
6B	XRL A,R3	8B	MOV dadr,R3	AB	MOV R3,dadr
6C	XRL A,R4	8C	MOV dadr,R4	AC	MOV R4,dadr
6D	XRL A,R5	8D	MOV dadr,R5	AD	MOV R5,dadr
6E	XRL A,R6	8E	MOV dadr,R6	AE	MOV R6,dadr
6F	XRL A,R7	8F	MOV dadr,R7	AF	MOV R7,dadr
70	JNZ rel	90	MOV DPTR,#k16	B0	ANL C,badr\
71	ACALL page 3	91	ACALL page 4	B1	ACALL page 5
72	ORL C,badr	92	MOV badr,C	B2	CPL badr
73	JMP @A+DPTR	93	MOVC A,@A+DPTR	B3	CPL C
74	MOV A,#k8	94	SUBB A,#k8	B4	CJNE A,#k8,rel
75	MOV dadr,#k8	95	SUBB A,dadr	B5	CJNE A,dadr,rel
76	MOV @R0,#k8	96	SUBB A,@R0	B6	CJNE @R0,#k8,rel
77	MOV @R1,#k8	97	SUBB A,@R1	B7	CJNE @R1,#k8,rel
78	MOV R0,#k8	98	SUBB A,R0	B8	CJNE R0,#k8,rel
79	MOV R1,#k8	99	SUBB A,R1	B9	CJNE R1,#k8,rel
7A	MOV R2,#k8	9A	SUBB A,R2	BA	CJNE R2,#k8,rel
7B	MOV R3,#k8	9B	SUBB A,R3	BB	CJNE R3,#k8,rel
7C	MOV R4,#k8	9C	SUBB A,R4	BC	CJNE R4,#k8,rel
7D	MOV R5,#k8	9D	SUBB A,R5	BD	CJNE R5,#k8,rel
7E	MOV R6,#k8	9E	SUBB A,R6	BE	CJNE R6,#k8,rel
7F	MOV R7,#k8	9F	SUBB A,R7	BF	CJNE R7,#k8,rel

C0	PUSH dadr	E0	MOVX A,@DPTR
C1	AJMP page 6	E1	AJMP page 7
C2	CLR badr	E2	MOVX A,@R0
C3	CLR C	E3	MOVX A,@R1
C4	SWAP A	E4	CLR A
C5	XCH A,dadr	E5	MOV A,dadr
C6	XCH A,@R0	E6	MOV A,@R0
C7	XCH A,@R1	E7	MOV A,@R1
C8	XCH A,R0	E8	MOV A,R0
C9	XCH A,R1	E9	MOV A,R1
CA	XCH A,R2	EA	MOV A,R2
CB	XCH A,R3	EB	MOV A,R3
CC	XCH A,R4	EC	MOV A,R4
CD	XCH A,R5	ED	MOV A,R5
CE	XCH A,R6	EE	MOV A,R6
CF	XCH A,R7	EF	MOV A,R7
D0	POP dadr	F0	MOVX @DPTR,A
D1	ACALL page 6	F1	ACALL page 7
D2	SETB badr	F2	MOVX @R0,A
D3	SETB C	F3	MOVX @R1,A
D4	DA A	F4	CPL A
D5	DJNZ dadr,rel	F5	MOV dadr,A
D6	XCHD A,@R0	F6	MOV @R0,A
D7	XCHD A,@R1	F7	MOV @R1,A
D8	DJNZ R0,rel	F8	MOV R0,A
D9	DJNZ R1,rel	FA	MOV R1,A
DA	DJNZ R2,rel	FB	MOV R2,A
DB	DJNZ R3,rel	FC	MOV R3,A
DC	DJNZ R4,rel	FC	MOV R4,A
DD	DJNZ R5,rel	FD	MOV R5,A
DE	DJNZ R6,rel	FE	MOV R6,A
DF	DJNZ R7,rel	FF	MOV R7,A

ACALL adr11

Absoluter Unterprogrammaufruf innerhalb 2 KByte

(PC) <- (PC) + 2
(SP) <- SP) + 1
((SP)) <- (PC7-0)
(SP) <- (SP) + 1
((SP)) <- (P15 - 8)
(PC10-0) <- Page-Adresse (Adressbits A10 ... A0)

ACALL adr11 2 Bytes, 2 Zyklen PSW: -

			1	0	0	0	1
a10	a9	a8					

a7	a6	a5	a4	a3	a2	a1	a0

ACALL Page 0 = 11
ACALL Page 1 = 31
ACALL Page 2 = 51
ACALL Page 3 = 71
ACALL Page 4 = 91
ACALL Page 5 = B1
ACALL Page 6 = D1
ACALL Page 7 = F1

ADD A, <Quellenbyte> - Addition

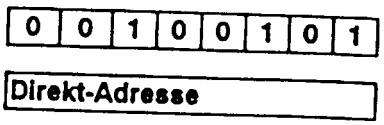
(A) <- (A) + Operand

ADD A, Rr 1 Byte, 1 Zyklus PSW: P, C, A, C, OV

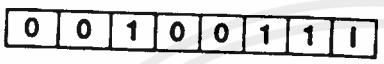
0	0	1	0	1	r	r	r

ADD A, R0 = 28
ADD A, R1 = 29
ADD A, R3 = 2A
ADD A, R3 = 2B
ADD A, R4 = 2C
ADD A, R5 = 2D
ADD A, R6 = 2E
ADD A, R7 = 2F

ADD A, direkt 2 Bytes, 1 Zyklus PSW: P, C, A, C, OV

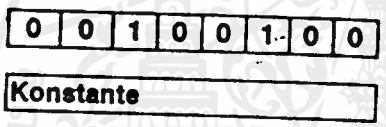


ADD A, @Ri 1 Byte, 1 Zyklus PSW: P, C, A, C, OV



ADD A, @R0 = 26 ADD A,@R1 = 27

ADD A, #Konstante 2 Bytes, 1 Zyklus PSW: P, C, A, C, OV



ADDC A, <Quellenbyte> - Addition + C

(A) <- (A) + (C) + Operand

ADDC A, Rr 1 Byte, 1 Zyklus PSW: P, C, A, C, OV

ADDC A, R0 = 38
ADDC A, R1 = 39
ADDC A, R3 = 3A
ADDC A, R3 = 3B
ADDC A, R4 = 3C
ADDC A, R5 = 3D
ADDC A, R6 = 3E
ADDC A, R7 = 3F

ADDC A, direkt 2 Byte, 1 Zyklus PSW: P, C, A, C, OV

0	0	1	1	0	1	0	1

35

Direkt-Adresse

ADDC A, @RI 1 Byte, 1 Zyklus PSW: P, C, A, C, OV

0	0	1	1	0	1	1	I

ADDC A, @R0 = 36 ADDC A,@R1 = 37

ADDC A, #Konstante 2 Bytes, 1 Zyklus PSW: P, C, A, C, OV

0	0	1	1	0	1	0	0

34

Konstante

AJMP adr11

Unbedingter Sprung innerhalb 2 KByte

(PC) <- (PC) + 2
(PC10-0) <- Adresse innerhalb der aktuellen Page

ACALL adr11 2 Bytes, 2 Zyklen PSW: -

a10	a9	a8	0	0	0	0	1

a7	a6	a5	a4	a3	a2	a1	a0

AJMP Page 0 = 01
AJMP Page 1 = 21
AJMP Page 2 = 41
AJMP Page 3 = 61
AJMP Page 4 = 81
AJMP Page 5 = A1
AJMP Page 6 = C1
AJMP Page 7 = 1E

ANL <Zielbyte>,<Quellenbyte>

logisches UND

(A) <- (A) UND Operand

ANL A, Rr — 1 Byte, 1 Zyklus — PSW: P

0	1	0	0	1	r	r	r

ANL A,R0 = 58 — ANL A,R4 = 5C
ANL A,R1 = 59 — ANL A,R5 = 5D
ANL A,R2 = 5A — ANL A,R6 = 5E
ANL A,R3 = 5B — ANL A,R7 = 5F

ANL direkt,#Konstante — 3 Bytes, 2 Zyklen — PSW: P

0	1	0	1	0	0	1	1	53
Direkt-Adresse								
Konstante								

ANL A, @Ri — 1 Byte, 1 Zyklus — PSW: P

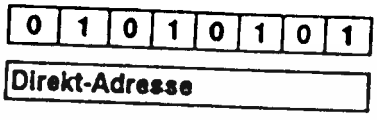
0	1	0	1	0	1	1	i

ANL A, @R0 = 56 — ANL A,@R1 = 57

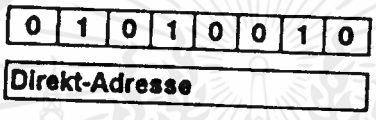
ANL A, #Konstante — 2 Bytes, 1 Zyklus — PSW: P

0	1	0	1	.0	1	0	0	54
Konstante								

ANL A,direkt 2 Bytes, 1 Zyklus PSW: P



ANL direkt,A 2 Bytes, 1 Zyklus PSW: -

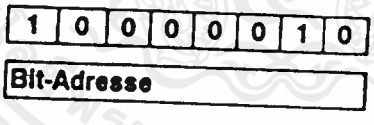


ANL C, <Quellenbit> -

logisches UND mit Bit

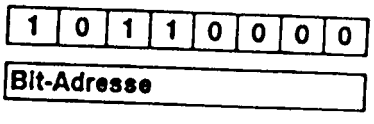
(A) <- (C) UND Bit

ANL C, Bit 2 Bytes, 2 Zyklen PSW: C



(A) <- (C) UND negiertes Bit

ANL C, Bit\ 2 Bytes, 2 Zyklen PSW: C

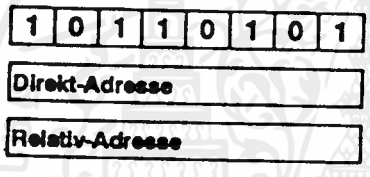


CJNE <Zielbyte>,<Quellenbyte>,rel

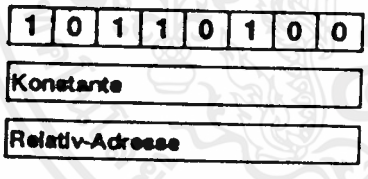
Vergleich und Sprung, wenn nicht gleich

(PC) <- (PC) + 3
wenn (A) <> (direkt)
dann (PC) <- (PC) + relativ Adresse
wenn (A) < (direkt)
(C) <- 1, sonst
(C) <- 0

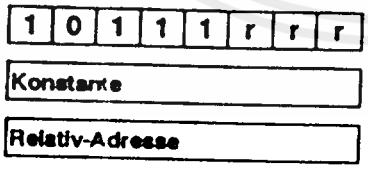
CJNE A,direkt,rel 3 Bytes, 2 Zyklen PSW: C



CJNE A,#Konstante,rel 3 Bytes, 2 Zyklen PSW: C

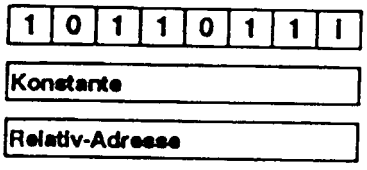


CJNE Rr,#Konstante,rel 3 Bytes, 2 Zyklen PSW: C



CJNE R0,#Daten,rel = B8 CJNE R4,#Daten,rel = BC
CJNE R1,#Daten,rel = B9 CJNE R5,#Daten,rel = BD
CJNE R2,#Daten,rel = BA CJNE R6,#Daten,rel = BE
CJNE R3,#Daten,rel = BB CJNE R7,#Daten,rel = BF

CJNE Rr,#Konstante,rel 3 Bytes, 2 Zyklen PSW: C

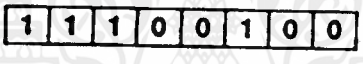


CJNE @R0,#Daten,rel = B6 CJNE @R1,#Daten,rel = B7

CLR A - löscht A

(A) <- 0

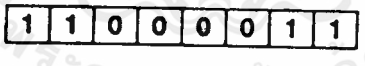
CLR A 1 Byte, 1 Zyklus PSW: P



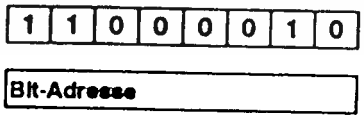
CLR <Bit> - löscht Bit

(Bit) <- 0

CLR C 1 Byte, 1 Zyklus PSW: C



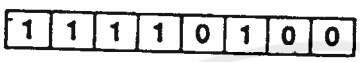
CLR Bit 2 Bytes, 1 Zyklus PSW: P



CPL A - negiert A (Einerkomplement)

(A) <- A\

CPL A 1Byte, 1 Zyklus PSW:

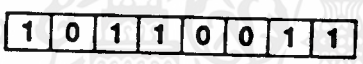


F4

CPL <Bit> - negiert Bit

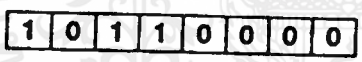
(A) <- Bit\

CPL C 1Byte, 1 Zyklus PSW:.C



B3

CPL Bit 2 Bytes, 1 Zyklus PSW: P



B2

DA A - Dezimalkorrektur für ADD

A enthält BCD-Zahlen

A enthält BCD-Zahlen,
wenn [[(3 - 0) > 9] oder [(AC) = 1]],
dann
(A3 - 0) <- (A3 - 0) + 6
und
wenn [[(A7 - 4) > 9] oder [(C) = 1]],
dann
(A7 - 4) <- (A7 - 4) + 6

DA A — 1 Byte, 1 Zyklus — PSW: P, C, AC

1	1	0	1	0	1	0	0

D4

DEC <Byte> - dekrement Byte

(Byte) <- (Byte) - 1

DEC A — 1 Byte, 1 Zyklus — PSW: P

0	0	0	1	0	1	0	0

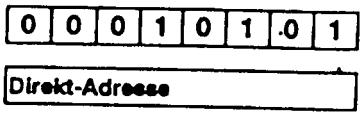
14

DEC Rr — 1 Byte, 1 Zyklus — PSW: -

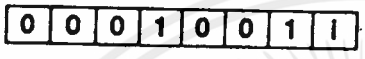
0	0	0	1	1	r	r	r

DEC R0 = 18
DEC R1 = 19
DEC R2 = 1A
DEC R3 = 1B
DEC R4 = 1C
DEC R5 = 1D
DEC R6 = 1E
DEC R7 = 1F

DEC direkt



DEC @RI 1 Byte, 1 Zyklus PSW: -

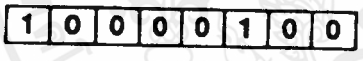


DEC @R0 = 16 DEC @R1 = 17

DIV - A dividiert durch B

(A) <- (A)/(B)
(B) <- Rest von (A)/(B)

DIV AB 1 Bytes, 4 Zyklen PSW: P, C, OV



DJNZ <Byte>,<rel>

dekrementiere und springe wenn <> 0

(PC) <- (PC) + 2
(Rr) <- (Rr) - 1, wenn (Rr) <> 0, dann
(PC) <- (PC) + rel

DJNZ Rr, rel — 2 Bytes, 2 Zyklen — PSW: -

1	1	0	1	1	r	r	r

DJNZ R0,rel = D8 — DJNZ R4.rel = DC
DJNZ R1,rel = D9 — DJNZ R5.rel = DD
DJNZ R2,rel = DA — DJNZ R6.rel = DE
DJNZ R3,rel = DB — DJNZ R7.rel = DF

DJNZ direkt,rel — 3 Bytes, 2 Zyklen — PSW: -

1	1	0	1	0	1	0	1	D5

Direkt-Adresse

Relativ-Adresse

INC <Byte> - inkrement Byte

(Byte) <- (Byte) + 1

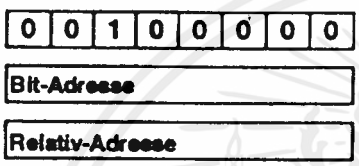
INC A — 1 Byte, 1 Zyklus — PSW: P

0	0	0	0	0	1	0	0	'04

JB rel - springt wenn Bit gesetzt

(PC) <- (PC) +3, wenn (Bit) = 1, dann
(PC) <- (PC) + rel

JB Bit,rel 3 Bytes, 2 Zyklen PSW: -

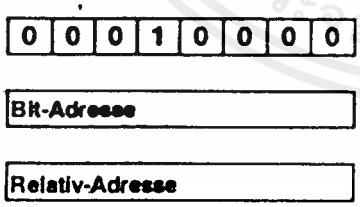


JBC rel - springt wenn Bit gesetzt

springt, wenn Bit gesetzt und löscht es

(PC) <- (PC) +3, wenn (Bit) = 1, dann
(Bit) <- 0
(PC) <- (PC) + rel

JBC Bit,rel 3 Bytes, 2 Zyklen PSW: -

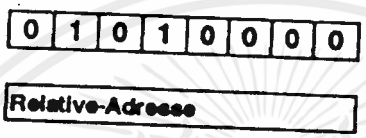


JNC rel - springt wenn C nicht gesetzt

(PC) <- (PC) +2, wenn (C) = 0, dann
(PC) <- (PC) + rel

JNC rel

2 Bytes, 2 Zyklen PSW: -

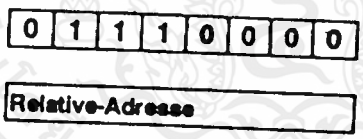


JNZ rel - springt wenn A <> 0

(PC) <- (PC) +2, wenn (A) <> 0, dann
(PC) <- (PC) + rel

JNZ rel

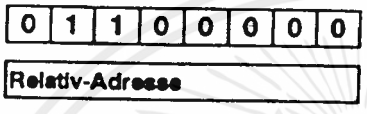
2 Bytes, 2 Zyklen PSW: -



JZ rel - springt wenn A = 0

(PC) <- (PC) +2, wenn (A) <> 0, dann
(PC) <- (PC) + rel

2 Bytes, 2 Zyklen PSW: -

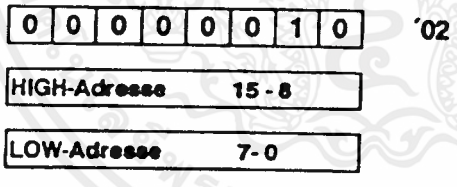


LJMP adr16

Sprung nach 16-Bit-Adresse

(PC) <- adr 15 - 0

LJMP adr16 3 Bytes, 2 Zyklen PSW: P



MOV <Zielbyte>,<Quellenbyte>

MOV A,Rr — 1 Byte, 1 Zyklus — PSW: P

1	1	1	0	1	r	r	r

MOV A, R0 = E8 — MOV A, R4 = EC
MOV A, R1 = E9 — MOV A, R5 = ED
MOV A, R2 = EA — MOV A, R6 = EE
MOV A, R3 = EB — MOV A, R7 = EF

MOV A,direkt — 3 Bytes, 2 Zyklen — PSW: P

1	1	1	0	0	1	0	1

E5

Direkt-Adresse

MOV A,ACC ist unzulässig

MOV A,@Ri — 1 Byte, 1 Zyklus — PSW: P

1	1	1	0	0	1	1	i

MOV A,@R0 = E6 — MOV A,@R1 = E7

MOV A,#Konstante — 2 Bytes, 1 Zyklus — PSW: P

0	1	1	1	0	1	0	0

74

Konstante

MOV Rr,A 1 Byte, 1 Zyklus PSW: P

1	1	1	1	1	r	r	r

MOV A, R0 = F8 MOV A, R4 = FC
MOV A, R1 = F9 MOV A, R5 = FD
MOV A, R2 = FA MOV A, R6 = FE
MOV A, R3 = FB MOV A, R7 = FF

MOV Rr,direkt 2 Bytes, 2 Zyklen PSW: -

1	0	1	0	1	r	r	r

Direkt-Adresse

MOV R0,direkt = A8 MOV R4,direkt = AC
MOV R1,direkt = A9 MOV R5,direkt = AD
MOV R2,direkt = AA MOV R6,direkt = AE
MOV R3,direkt = AB MOV R7,direkt = AF

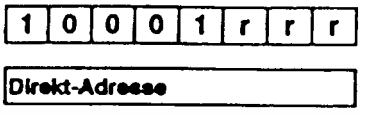
MOV Rr,#Konstante 2 Bytes, 1 Zyklus PSW: -

0	1	1	1	1	r	r	r

Konstante

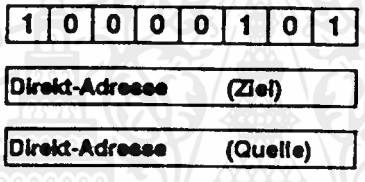
MOV R0,#Konstante = 78 MOV R4,#Konstante = 7C
MOV R1,#Konstante = 79 MOV R5,#Konstante = 7D
MOV R2,#Konstante = 7A MOV R6,#Konstante = 7E
MOV R3,#Konstante = 7B MOV R7,#Konstante = 7F

MOV direkt,Rr **2 Bytes, 2 Zyklen** **PSW: -**

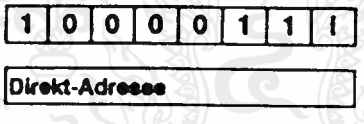


MOV direkt,R0 = 88 MOV direkt,R4 = 8C
MOV direkt,R1 = 89 MOV direkt,R5 = 8D
MOV direkt,R2 = 8A MOV direkt,R6 = 8E
MOV direkt,R3 = 8B MOV direkt,R7 = 8F

MOV direkt,direkt **3 Bytes, 2 Zyklen** **PSW: -**

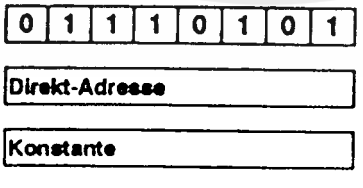


MOV direkt,@Ri **2 Bytes, 2 Zyklen** **PSW: -**

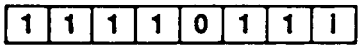


MOV direkt,@R0 = 86 MOV direkt,@R1 = 87

MOV direkt,#Konstante **3 Bytes, 2 Zyklen** **PSW: -**

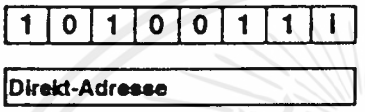


MOV@Ri,A 1Byte, 1 Zyklus PSW: -



MOV @R0,A = F6 MOV @R1,A = F7

MOV @Ri,direkt 2 Bytes, 2 Zyklen PSW: -



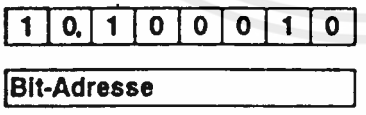
MOV @R0,direkt = A6 MOV @R1,direkt = A7

MOV @Ri,#Daten 2 Bytes, 1 Zyklen PSW: -

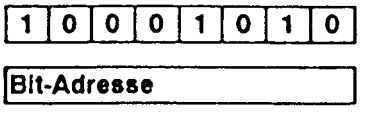
MOV @R0,#Daten = 76 MOV @R1,#Daten = 77

MOV Bit - <Zielbit>,<Quellenbit>

MOV C,Bit 2 Bytes, 1 Zyklus PSW: C



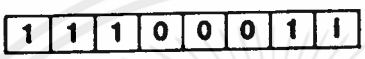
MOV Bit,C 2 Bytes, 2 Zyklen PSW: -



MOVX <Zielbyte>,<Quellenbyte>

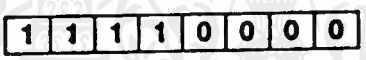
Transport zum/vom externen Speicher

MOVX A,@RI 1 Byte, 2 Zyklen PSW: -



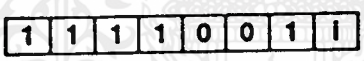
MOVX A,@R0 = E2 MOVX A,@R1 = E3

MOVX A,@DPTR 1 Byte, 2 Zyklen PSW: P



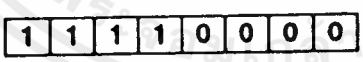
E0

MOVX @RI,A 1 Byte, 2 Zyklen PSW: -



MOV @R0,A = F2 MOV @R1,A = F3

MOVX @DPTR,A 1 Byte, 2 Zyklen PSW: -

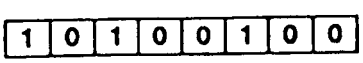


F0

MUL AB - multipliziert A mit B

(A) <- (A) * (B), Bit 7 - 0
(B) <- (A) * (B), Bits 15 - 8

MUL AB 1 Byte, 4 Zyklen PSW: P, OV, C = 0



A4

NOP - Leerbefehl

(PC) <- (PC) + 1

NOP 1 Byte, 1 Zylus PSW: -

0	0	0	0	0	0	0	0	'00

ORL <Zielbyte>,<Quellenbyte>

logisches ODER

(A) <- (A) ODER Operand

ORL A, Rr 1 Byte, 1 Zyklus PSW: P

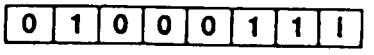
0	1	0	0	1	r	r	r

ORL A,R0 = 48 ORL A,R4 = 4C
ORL A,R1 = 49 ORL A,R5 = 4D
ORL A,R2 = 4A ORL A,R6 = 4E
ORL A,R3 = 4B ORL A,R7 = 4F

ORL direkt,#Konstante 3 Bytes, 2 Zyklen PSW: P

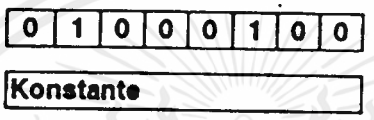
0	1	0	0	0	0	1	1	43
Direkt-Adresse								
Konstante								

ORL A, @Ri 1 Byte, 1 Zyklus PSW: P

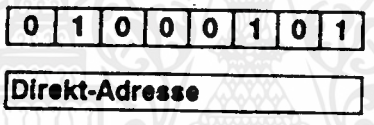


ORL A, @R0 = 46 ORL A,@R1 = 47

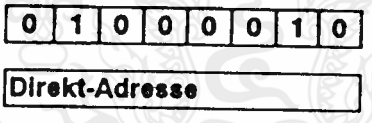
ORL A, #Konstante 2 Bytes, 1 Zyklus PSW: P



ORL A, direkt 2 Bytes, 1 Zyklus PSW: P



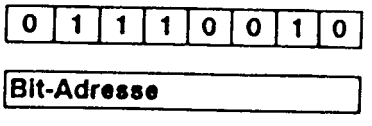
ORL direkt,A 2 Bytes, 1 Zyklus PSW: -



ORL C, <Quellenbit> - logisches ODER

(A) <- (C) ODER Operand

ORL C, Bit 2 Bytes, 2 Zyklen PSW: C



ORL C, Bit\ 2 Bytes, 2 Zyklen PSW: C

1	0	1	0	0	0	0	0	A0
Bit-Adresse								

POP direkt - POP vom STACK

(direkt) <- ((SP))
(SP) <- (SP) - 1

POP direkt 2 Bytes, 2 Zyklen PSW: C bei POP ACC

1	1	0	1	0	0	0	0	D0 PSW: C bei POP ACC
Direkt-Adresse								

PUSH direkt - PUSH zum STACK

(SP) <- (SP) + 1
((SP)) <- (direkt)

PUSH direkt 2 Bytes, 2 Zyklen PSW: -

1	1	0	0	0	0	0	0	C0
Direkt-Adresse								

RL A - linksrotieren von A

(A.n + 1) <- (A.n) n = Bitstelle A.0 bis A.6
(A.0) <- (A.7)

4 mal RL A ist mit SWAP effizienter

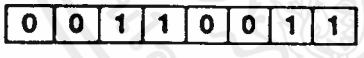
RL A 1 Byte, 1 Zyklus PSW: -

23

RLC A - linksrotieren von A durch C

(A.n + 1) <- (A.n) n = Bitstelle A.0 bis A.6
(A.0) <- (C)
(C) <- (A.7)

RLC A 1 Byte, 1 Zyklus PSW: C, P



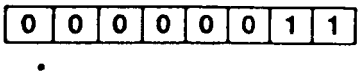
33

RRA - rechtsrotieren von A

(A.n - 1) <- (A.n) n = Bitstelle A.1 bis A.7
(A.7) <- (A.0)

4 mal RL A ist mit SWAP effizienter

RR A 1 Byte, 1 Zyklus PSW: -



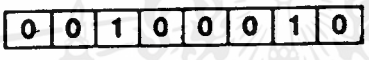
3

RET

Rücksprung vom Unterprogramm

PC(15-8) <- ((SP))
(SP) <- (SP) - 1
(PC7-0) <- ((SP))
(SP) <- (SP) - 1

RET 1 Byte, 2 Zyklen PSW: -



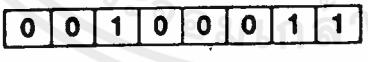
22

RETI

Rücksprung vom Interruptprogramm

PC(15-8) <- ((SP))
(SP) <- (SP) - 1
(PC7-0) <- ((SP))
(SP) <- (SP) - 1

RETI 1 Byte, 2 Zyklen PSW: -

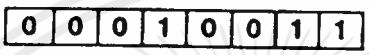


32

RRC A - rechtsrotieren von A durch C

(A.n) <- (An - 1) n = Bittstelle A.1 bis A.7
(A.7) <- (C)
(C) <- (A.0)

RRC A 1 Byte, 1 Zyklus PSW: C, P



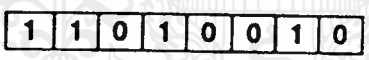
13

SETB Bit - setzt Bit

SETB C 1 Byte, 1 Zyklus PSW: C

D3

SETB Bit 1 Byte, 2 Zyklen PSW: -

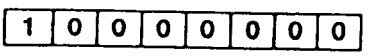


D2

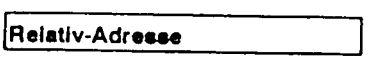
SJMP rel - springt relativ

(PC) <- (PC) + 2
(PC) <- (PC) + rel; -128 bis 127 dezimal

SJMP rel 2 Bytes, 2 Zyklen PSW: -



80



SUBB A,<Quellenbyte>

Subtraktion - C

(A) <- (A) - (C) - Operand

Subtraktion einer Konstanten ohne Carry ist mit ADD A,#-Konstante möglich. C ist dann entgegengesetzt dem von SUBB

SUBB A,Rr — 1 Byte, 1 Zyklus — PSW: P, C, A, C, OV

1	0	0	1	1	r	r	r

SUBB A, R0 = 98
SUBB A, R1 = 99
SUBB A, R2 = 9A
SUBB A, R3 = 9B
SUBB A, R4 = 9C
SUBB A, R5 = 9D
SUBB A, R6 = 9E
SUBB A, R7 = 9F

SUBB A, direkt — 2 Byte, 1 Zyklus — PSW: P, C, A, C, OV

1	0	0	1	0	1	0	1

95

Direkt-Adresse

SUBB A, @Ri — 1 Byte, 1 Zyklus — PSW: P, C, A, C, OV

1	0	0	1	0	1	1	i

SUBB A, @R0 = 96
SUBB A,@R1 = 97

SUBB A, #Konstante — 2 Bytes, 1 Zyklus — PSW: P, C, A, C, OV

1	0	0	1	0	1	0	0

94

Konstante

SWAP

tauscht Nibbles

(A3 - 0) <-> (A7 - 4)

SWAP A

1 Bytes, 1 Zyklen PSW: -

1	1	0	0	0	1	0	0

XCH A,<Quellenbyte>

tauscht Bytes

XCH A, Rr

1 Byte, 1 Zyklus PSW: P

1	1	0	0	1	r	r	r

XCH A,R0 = C8
XCH A,R1 = C9
XCH A,R2 = CA
XCH A,R3 = CB
XCH A,R4 = CC
XCH A,R5 = CD
XCH A,R6 = CE
XCH A,R7 = CF

XCH A,direkt

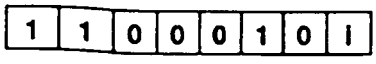
2 Bytes, 1 Zyklus PSW: P

1	1	0	0	1	0	0	1

C5

Direkt-Adresse

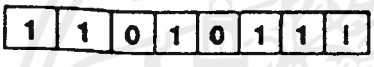
XCH A, @Ri 1 Byte, 1 Zyklus PSW: P



XCH A, @R0 = C6 XCH A,@R1 = C7

XCHD A,@R i- tauscht LOW Nibble

XCHD A,@Ri 1 Byte, 1 Zyklus PSW: B



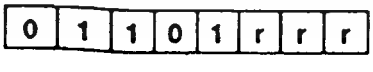
XCHD A,@R0 = D6 XCHD A,@R1 = D7

XRL <Zielbyte>,<Quellenbyte>

logisches EXKLUSIV ODER

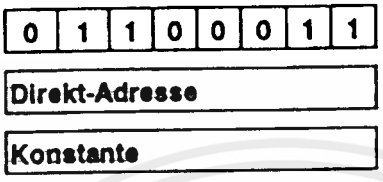
(A) <- (A) EXKLUSIV ODER Operand

XRL A, Rr 1 Byte, 1 Zyklus PSW: P

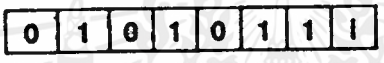


XRL A,R0 = 68 XRL A,R4 = 6C
XRL A,R1 = 69 XRL A,R5 = 6D
XRL A,R2 = 6A XRL A,R6 = 6E
XRL A,R3 = 6B XRL A,R7 = 6F

XRL direkt,#Konstante 3 Bytes, 2 Zyklen PSW:



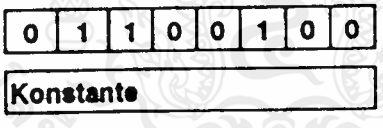
XRL A, @Ri 1 Byte, 1 Zyklus PSW: P



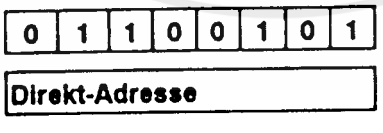
XRL A, @R0 = 66 XRL A,@R1 = 67

XRL A, #Konstante 2 Bytes, 1 Zyklus PSW: P

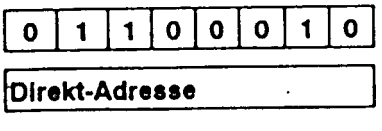
den speziellen Fall XRL A,#0FFH erledigt CPL A mit einem Byte



XRL A,direkt 2 Bytes, 1 Zyklus PSW: P



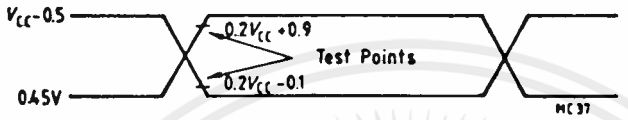
XRL direkt,A 2 Bytes, 1 Zyklus PSW: -



ภาคผนวก ข

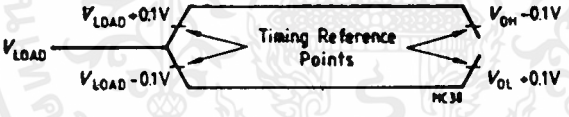
คุณลักษณะของ SAB 80C535

AC Testing: Input, Output Waveforms



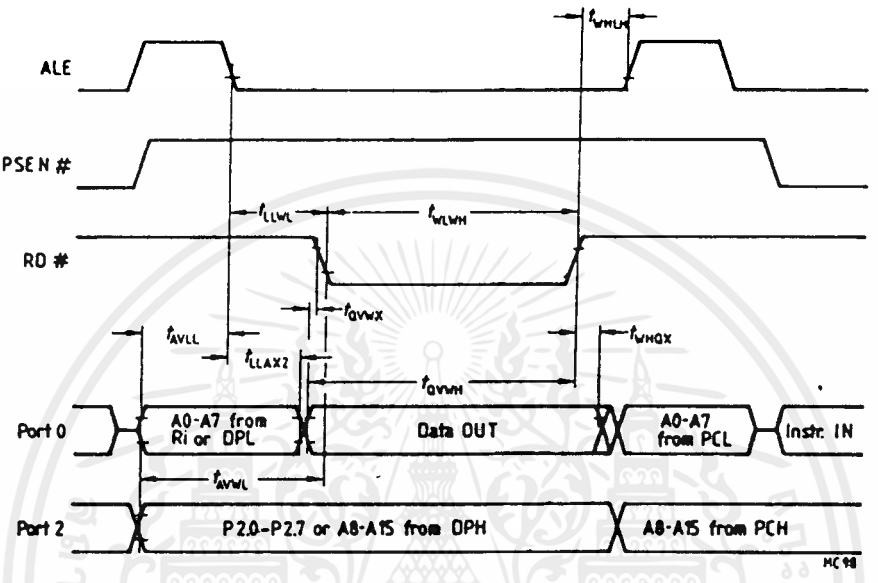
AC inputs during testing are driven at V_{CC} - 0.5 V for a logic '1' and 0.45 V for a logic '0'.
Timing measurements are made at $V_{IH\,min}$ for a logic '1' and $V_{IL\,max}$ for a logic '0'.

AC Testing: Float Waveforms

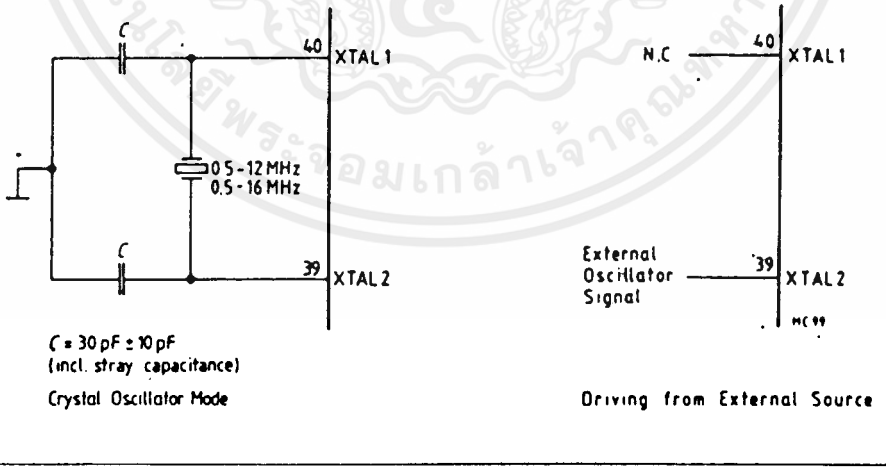


For timing purposes a port pin is no longer floating when a 100 mV change from load voltage occurs and begins to float when a 100 mV deviation from the load voltage V_{OH}/V_{OL} occurs. $I_{OL}/I_{OH} \geq \pm 20$ mA.

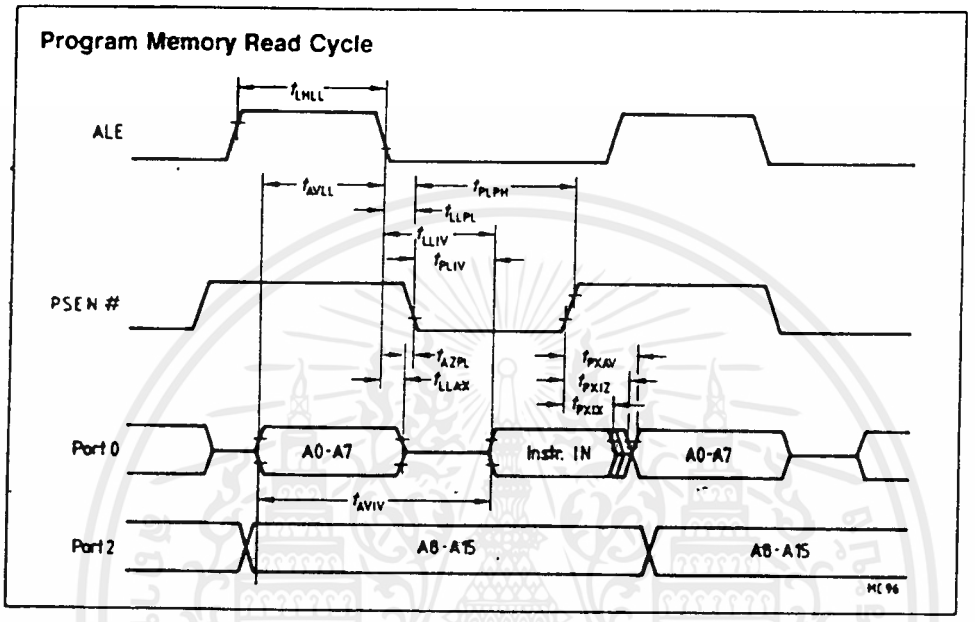
Data Memory Write Cycle

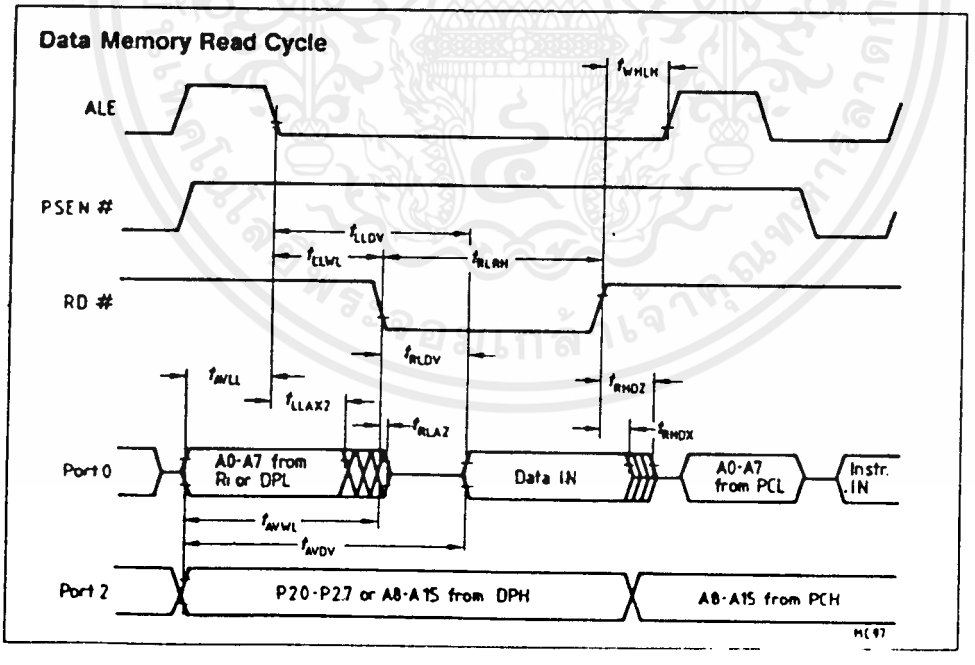


Recommended Oscillator Circuits



Waveforms

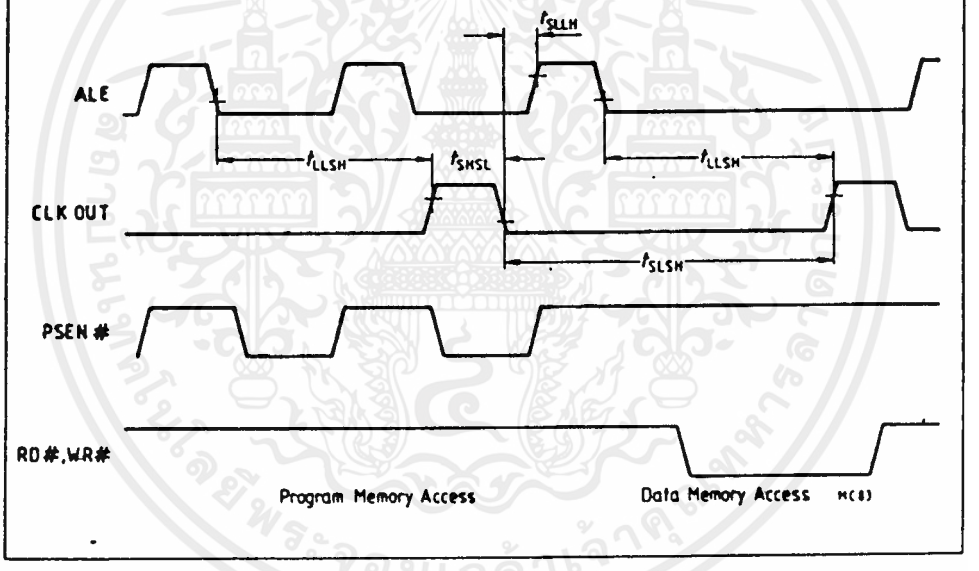




System Clock Timing for SAB 80C515/80C535

Parameter	Symbol	Limit values				Unit
		12 MHz clock		Variable clock $1/t_{CLCL}$ = 0.5 to 12 MHz		
		min.	max.	min.	max.	
ALE to CLKOUT	t_{LLSH}	543	-	$7\ t_{CLCL} - 40$	-	ns
CLKOUT high time	t_{SHSL}	127	-	$2\ t_{CLCL} - 40$	-	ns
CLKOUT low time	t_{SLSH}	793	-	$10\ t_{CLCL} - 40$	-	ns
CLKOUT low to ALE high	t_{SLLH}	43	123	$t_{CLCL} - 40$	$t_{CLCL} + 40$	ns

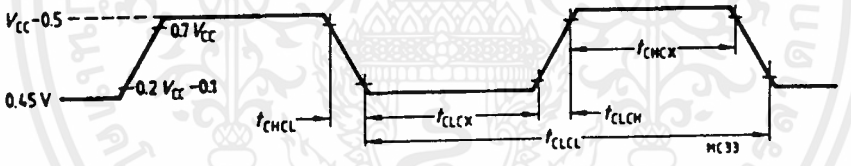
System Clock Timing



AC Characteristics for SAB 80C515/80C535 (cont'd)

Parameter	Symbol	Limit values		Unit
		Variable clock Frequ. = 0.5 to 12 MHz		
		min.	max.	
External Clock Drive				
Oscillator period	t_{CLCL}	83.3	2000	ns
Oscillator frequency	$1/t_{CLCL}$	0.5	12	MHz
High time	t_{CHCX}	20	–	ns
Low time	t_{CLCX}	20	–	ns
Rise time	t_{CLCH}	–	20	ns
Fall time	t_{CHCL}	–	20	ns

External Clock Cycle



AC Characteristics for SAB 80C515/80C535 (cont'd)

V_{CC} = 5 V ± 10 %; V_{SS} = 0 V (C_L for Port 0, ALE and PSEN# outputs = 100pF; C_L for all outputs = 80 pF) T_A = 0 to +70 °C for SAB 80C515/80C535
T_A = – 40 to +85 °C for SAB 80C515/80C535-T40/85
T_A = – 40 to +110 °C for SAB 80C515/80C535-T40/110

Parameter	Symbol	Limit values				Unit
		12 MHz clock		Variable clock $1/t_{CLCL}$ 0.5 to 12 MHz		
		min.	max.	min.	max.	
External Data Memory Characteristics						
RD# pulse width	t_{RLRH}	400	–	6 t_{CLCL} -100	–	ns
WR# pulse width	t_{WLWH}	400	–	6 t_{CLCL} -100	–	ns
Address hold after ALE	t_{LLAX2}	132	–	2 t_{CLCL} -35	–	ns
RD# to valid data in	t_{RLDV}	–	252	–	5 t_{CLCL} -165	ns
DATA hold after RD#	t_{RHDX}	0	–	0		ns
Data float after RD#	t_{RHDZ}	–	97	–	2 t_{CLCL} -70	ns
ALE to valid data in	t_{LLDV}	–	517	–	8 t_{CLCL} -150	ns
Address to valid data in	t_{AVDV}	–	585	–	9 t_{CLCL} -165	ns
ALE to WR# or RD#	t_{LLWL}	200	300	3 t_{CLCL} -50	3 t_{CLCL} + 50	ns
WR# or RD# high to ALE high	t_{WHLH}	43	123	t_{CLCL} -40	t_{CLCL} + 40	ns
Address valid to WR#	t_{AVWL}	203	–	4 t_{CLCL} -130	–	ns
Data valid to WR# transition	t_{QVWX}	33	–	t_{CLCL} -50	–	ns
Data setup before WR#	t_{QVWH}	433	–	7 t_{CLCL} -150	–	ns
Data hold after WR#	t_{WHQX}	33	–	t_{CLCL} -50	–	ns
Address float after RD#	t_{RLAZ}	–	0	–	0	ns

AC Characteristics for SAB 80C515/80C535

V_{CC} = 5 V ± 10 %; V_{SS} = 0 V (C_L for Port 0, ALE and PSEN# outputs = 100pF; C_L for all outputs = 80 pF) T_A = 0 to +70 °C for SAB 80C515/80C535
T_A = – 40 to +85 °C for SAB 80C515/80C535-T40/85
T_A = – 40 to +110 °C for SAB 80C515/80C535-T40/110

Parameter	Symbol	Limit values				Unit
		12 MHz clock		Variable clock $1/t_{CLCL}$ 0.5 to 12 MHz		
		min.	max.	min.	max.	
Program Memory Characteristics						
ALE pulse width	t_{LHLL}	127	–	$2\,t_{CLCL}$ - 40	–	ns
Address setup to ALE	t_{AVLL}	53	–	t_{CLCL} -30	–	ns
Address hold after ALE	t_{LLAX}	48	–	t_{CLCL} -35	–	ns
ALE to valid instruction in	t_{LLIV}	–	233	–	$4\,t_{CLCL}$ -100	ns
ALE to PSEN#	t_{LLPL}	58	–	t_{CLCL} -25	–	ns
PSEN# pulse width	t_{PLPH}	215		$3\,t_{CLCL}$ -35		ns
PSEN# to valid instruction in	t_{PLIV}	–	150	–	$3\,t_{CLCL}$ -100	ns
Input instruction hold after PSEN#	t_{PXIX}	0	–	0	–	ns
Input instruction float after PSEN#	t_{PXIZ} [1]	–	63	–	t_{CLCL} - 20	ns
Address valid after PSEN#	t_{PXAV} [1]	75		t_{CLCL} -8		ns
Address to valid instruction in	t_{AVIV}	–	302	–	$5\,t_{CLCL}$ -115	ns
Address float to PSEN#	t_{AZPL}	0	–	0	–	ns

[1] Interfacing the SAB 80C515 to devices with float times up to 75 ns is permissible. This limited bus contention will not cause any damage to port 0 drivers

A/D Converter Characteristics

V_{CC} = 5 V ± 10 %; V_{SS} = 0 V; V_{AREF} = V_{CC} ± 5 %; V_{AGND} = V_{SS} ± 0.2 V;
$V_{IntAREF} - V_{IntAGND} \geq 1$V; T_A = 0 to + 70° C for SAB 80C515/80C535
T_A = − 40 to + 85°C for SAB 80C515/80C535-T40/85
T_A = − 40 to + 110°C for SAB 80C515/80C535-T40/110

Parameter	Symbol	Limit values			Unit	Test conditions
		min.	typ.	max.		
Analog input voltage	V_{AINPUT}	V_{AGND} - 0.2	–	V_{AREF} + 0.2	V	9)
Analog input capacitance	C_I	–	25	45	pF	7)
Load time	t_L	–	–	2 t_{CY}	µs	–
Sample time (incl. load time)	t_S	–	–	7 t_{CY}	µs	–
Conversion time (incl. sample time)	t_C	–	–	13 t_{CY}	µs	–
Differential non-linearity	DNLE	–	± 1/2	± 1	LSB	$V_{IntAREF}$ =
Integral non-linearity	INLE	–	± 1/2	± 1	LSB	V_{AREF} = V_{CC}
Offset error			± 1/2	± 1	LSB	$V_{IntAGND}$ =
Gain error			± 1/2	± 1	LSB	V_{AGND} = V_{SS}
Total unadjusted error	TUE		± 1	± 2	LSB	7)
V_{AREF} supply current	I_{REF}	–	–	5	mA	8)
Internal reference error	$V_{IntREFERR}$	–	± 5	± 20	mV	8)

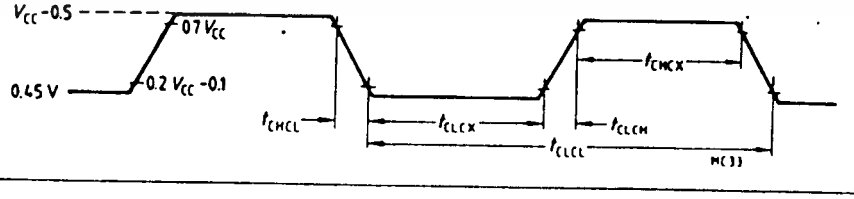
7) The output impedance of the analog source must be low enough to assure full loading of the sample capacitance (C_I) during load time (t_L). After charging of the internal capacitance (C_I) in the load time (t_L) the analog input must be held constant for the rest of the sample time (t_S).

8) The differential impedance r_D of the analog reference voltage source must be less than 1 kΩ at reference supply voltage.

9) Exceeding these limit values at one or more input channels will cause additional current which is sinked / sourced at these channels. This may also affect the accuracy of other channels which are operated within these specifications.

Clock Signal Waveform for I_{CC} Tests In Active and Idle Mode

(t_{CLCH} = t_{CHCL} = 5 ns)



DC Characteristics (cont'd)

Parameter	Symbol	Limit values		Unit	Test condition
		min.	max.		
Logic 0 input current, ports 1, 2, 3, 4, 5	I_{IL}	–	–50	µA	V_{IN} = 0.45 V
Output high voltage (port 0 in external bus mode, ALE, PSEN)	V_{OH1}	2.4 0.9 V_{CC}	– –	V V	I_{OH} = – 400 µA I_{OH} = – 40 µA [2]
Logic 0 input current, ports 1, 2, 3, 4, 5	I_{IL}	–	–50	µA	V_{IN} = 0.45 V
Input low current to RESET# for reset	I_{IL2}	–	–100	µA	V_{IN} = 0.45 V
Logical 1-to-0 transition current, ports 1, 2, 3, 4, 5	I_{TL}	–	–650	µA	V_{IN} = 2 V
Input leakage current (port 0, EA#)	I_{LI}	–	± 10	µA	0.45 < V_{IN} < V_{CC}
Pin capacitance	C_{IO}	–	10	pF	f_C = 1 MHz, T_A = 25 °C
Power-supply current:					
Active mode, 12 MHz [6]	–	–	35	mA	V_{CC} = 5 V [4]
Idle mode, 12 MHz [6]	–	–	13	mA	V_{CC} = 5 V [5]
Active mode, 16 MHz [6]	–	–	46	mA	V_{CC} = 5 V [4]
Idle mode, 16 MHz [6]	–	–	17	mA	V_{CC} = 5 V [5]
Power-down mode	–	–	50	µA	V_{CC} = 2 V to 5.5 V [3]

Notes (for page 35 and 36)

1) Capacitive loading on ports 0 and 2 may cause spurious noise pulses to be superimposed on the V_{OL} of ALE and ports 1, 3, 4, and 5. The noise is due to external bus capacitance discharging into the port 0 and port 2 pins when these pins make 1-to-0 transitions during bus operation.
In the worst case (capacitive loading > 100 pF), the noise pulse on ALE line may exceed 0.8 V.
Then, it may be desirable to qualify ALE with a Schmitttrigger, or use an address latch with a Schmitttrigger strobe input.

2) Capacitive loading on ports 0 and 2 may cause the VOH on ALE and PSEN to momentarily fall below the 0.9 V_{CC} specification when the address bits are stabilizing.

3) Power-down I_{CC} is measured with: EA# = Port 0 = Port 6 = V_{CC};
XTAL1 = N.C.; XTAL2 = V_{SS}; RESET# = V_{CC}; V_{AGND} = V_{SS}; all other pins are disconnected.

4) I_{CC} (active mode) is measured with: XTAL2 driven with the clock signal according to the figure below;
XTAL1 = N.C.; EA# = Port 0 = Port 6 = V_{CC}; RESET# = V_{SS}; all other pins are disconnected. I_{CC} might be slightly higher if a crystal oscillator is used.

5) I_{CC} (idle mode) is measured with: XTAL2 driven with the clock signal according to the figure below; XTAL1 = N.C.; EA# = V_{SS}; Port 0 = Port 6 = V_{CC}; RESET# = V_{CC}; all other pins are disconnected, all on-chip peripherals are disabled.

6) I_{CC} at other frequencies is given by
Active mode: $I_{CC\ max}$ (mA) = 2.67 · f_{OSC} (MHz) + 3.00
Idle mode: $I_{CC\ max}$ (mA) = 0.88 · f_{OSC} (MHz) + 2.50
where f_{OSC} is the oscillator frequency in MHz
$I_{CC\ max}$ is given in mA and measured at V_{CC} = 5 V (see also notes 4 and 5)

Absolute Maximum Ratings

Ambient temperature under bias	0 to + 70°C (SAB 80C515)
	- 40 to + 85°C (SAB 80C515-T40/85)
	- 40 to + 110°C (SAB 80C515-T40/110)
Storage temperature	- 65 to + 150°C
Voltage on any pin with respect to ground (V_{SS})	- 0.5 to V_{CC} + 0.5 V
Voltage on V_{CC} to V_{SS}	- 0.5to + 6.5 V
Power dissipation	2 W

Note: Stresses above those listed under "Absolute Maximum Ratings" may cause permanent damage to the device. Exposure to absolute maximum rating conditions for extended periods may affect device reliability.

DC Characteristics

V_{CC} = 5 V ± 10 %; V_{SS} = 0 V; T_A = 0 to + 70°C for SAB 80C515/80C535
T_A = - 40 to + 85°C for SAB 80C515/80C535-T40/85
T_A = - 40 to + 110°C for SAB 80C515/80C535-T40/110

Parameter	Symbol	Limit values		Unit	Test condition
		min.	max.		
Input low voltage (except EA#)	V_{IL}	- 0.5	0.2 V_{CC} - 0.1	V	-
Input low voltage (EA#)	V_{IL1}	- 0.5	0.2 V_{CC} - 0.3	V	-
Input high voltage (except RESET# and XTAL2)	V_{IH}	0.2 V_{CC} + 0.9	V_{CC} + 0.5	V	-
Input high voltage to XTAL2	V_{IH1}	0.7 V_{CC}	V_{CC} + 0.5	V	-
Input high voltage to RESET#	V_{IH2}	0.6 V_{CC}	V_{CC} + 0.5	V	-
Output low voltage, ports 1, 2, 3, 4, 5	V_{OL}	-	0.45	V	I_{OL} = 1.6 mA [1]
Output low voltage, port 0, ALE, PSEN#	V_{OL1}	-	0.45	V	I_{OL} = 3.2 mA [1]
Output high voltage, ports 1, 2, 3, 4, 5	V_{OH}	2.4 0.9 V_{CC}	- -	V V	I_{OH} = - 80 µA I_{OH} = - 10 µA
Output high voltage (port 0 in external bus mode, ALE, PSEN)	V_{OH1}	2.4 0.9 V_{CC}	- -	V V	I_{OH} = - 400 µA I_{OH} = - 40 µA [2]

for notes see next page