ระบบเรียนรู้ไมโครคอนโทรลเลอร์โดยใช้ภาษาเบสิก

นายอมรพงษ์ ชุมสาย ณ อยุธยา

วิทยานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิศวกรรมศาสตรมหาบัณฑิต
สาขาวิชาวิศวกรรมไฟฟ้า ภาควิชาวิศวกรรมไฟฟ้า
คณะวิศวกรรมศาสตร์ จุฬาลงกรณ์มหาวิทยาลัย
ปีการศึกษา 2542

ISBN 974-333-745-8

A MICROCONTROLLER LEARNING SYSTEM USING THE BASIC LANGUAGE

Mr. Amonpong Chumsai Na Ayuttaya

A Thesis Submitted in Partial Fulfillment of the Requirements

for the Degree of Master of Engineering in Electrical Engineering

Department of Electrical Engineering

Faculty of Engineering

Chulalongkorn University

Academic Year 1999

ISBN 974-333-745-8

หัวข้อวิทยานิพนธ์	ระบบเรียนรู้ไมโครคอนโทรลเลอร์โดยใช้ภาษาเบสิก
โดย	นายอมรพงษ์ ชุมสาย ณ อยุธยา
ภาควิชา	วิศวกรรมไฟฟ้า
อาจารย์ที่ปรึกษา	รองศาสตราจารย์กฤษดา วิศวธีรานนท์
อาจารย์ที่ปรึกษาร่วม	ศาสตราจารย์ ดร.มงคล เตชนครินทร์

คณะวิศวกรรมศาสตร์ จุฬาลงกรณ์มหาวิทยาลัย อนุมัติให้นับวิทยานิพนธ์ฉบับนี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญามหาบัณฑิต

_____คณบดีคณะวิศวกรรมศาสตร์
(ศาสตราจารย์ ดร.สมศักดิ์ ปัญญาแก้ว)

คณะกรรมการสอบวิทยานิพนธ์

_____ประธานกรรมการ
(รองศาสตราจารย์ ดร.เอกชัย ลีลารัศมี)

_____อาจารย์ที่ปรึกษา
(รองศาสตราจารย์กฤษดา วิศวธีรานนท์)

_____อาจารย์ที่ปรึกษาร่วม
(ศาสตราจารย์ ดร.มงคล เตชนครินทร์)

_____กรรมการ
(อาจารย์ ดร.ฐิต ศิริบูรณ์)

อมรพงษ์ ชุมสาย ณ อยุธยา : ระบบเรียนรู้ไมโครคอนโทรลเลอร์โดยใช้ภาษาเบสิก (A Microcontroller Learning System Using The BASIC Language) อ. ที่ปรึกษา : รศ.กฤษดา วิศวธีรานนท์ อ. ที่ปรึกษาร่วม : ศ.ดร.มงคล เตชนครินทร์, 134 หน้า. ISBN 974-333-745-8.

วิทยานิพนธ์นี้เสนอการออกแบบและสร้างระบบเรียนรู้ไมโครคอนโทรลเลอร์โดยใช้ภาษาเบสิก ในรูปแบบของชุดอุปกรณ์ฝึกทดลองสำหรับผู้ศึกษา ชุดทดลองนี้สามารถใช้เขียนและทดสอบโปรแกรมควบคุมได้โดยไม่ต้องใช้ไมโครคอมพิวเตอร์มาช่วยแต่อย่างใด เพราะมีตัวแปลภาษาเบสิกเป็นภาษาเครื่องติดตั้งอยู่แล้วจึงทำงานได้อย่างอิสระและใช้งานรับคำสั่งรูปแบบภาษาเบสิกได้ทำให้ผู้ใช้สามารถเข้าใจคำสั่งได้ง่ายและรวดเร็ว แผงทดลองของชุดอุปกรณ์มีขนาดเล็กและราคาถูก เหมาะสำหรับผู้ศึกษาที่มีงบประมาณน้อย ชุดฝึกทดลองแบ่งออกเป็น 2 ส่วน คือ ส่วนแผงควบคุม และส่วนแผงทดลองที่มีการทดลองย่อย

ชุดอุปกรณ์ฝึกทดลองมีชุดทดลองย่อยอยู่ในแผงทดลอง ผู้เรียนสามารถเรียนรู้หลักการพื้นฐานในการควบคุม คือ การแปลงสัญญาณจากแอนะล็อกเป็นดิจิทัล การแปลงสัญญาณดิจิทัลเป็นสัญญาณแอนะล็อก อินพุตและเอาต์พุตพอร์ตผ่านแบบขนาน ไฟวิ่ง 64 ดวงแบบเมทริกซ์ อินพุตแผงแป้นอักขระ 12 ปุ่ม การแสดงผลแบบแอลซีดี การควบคุมมอเตอร์กระแสตรงและสเต็ปปิงมอเตอร์ และการควบคุมแอลซีดีแบบเจ็ดส่วน

ภาควิชา วิศวกรรมไฟฟ้า	**ลายมือชื่อนิสิต**
สาขาวิชา วิศวกรรมไฟฟ้า	**ลายมือชื่ออาจารย์ที่ปรึกษา**
ปีการศึกษา 2542	**ลายมือชื่ออาจารย์ที่ปรึกษาร่วม**

AMONPONG CHUMSAI NA AYUTTAYA : A MICROCONTROLLER LEARNING SYSTEM USING THE BASIC LANGUAGE.
THESIS ADVISOR : ASSOC. PROF.KRISADA VISAVATEERANON.
THESIS COADVISOR : PROF. MONGKOL DETNAKARINTRA, Ph.D. ,134 pp.
ISBN 974-333-745-8.

This thesis presents the design and construction of a microcontroller learning system in the form of an experiment board set for students. The experiment set can be used to write and test a control program without using a microcomputer. This is because the learning system contains a BASIC interpreter, can be used independently, and can run a BASIC program. This facilitates the user's learning of the BASIC language. The experiment boards are small and inexpensive, this is suitable for students with a limited fund. The boards include two parts: The control board and the experiment board.

The experiment board can run many experiments which are fundamentals in control work, namely, analog-to-digital conversion, digital-to-analog conversion, Inputs and Outputs via a parallel port, a matrix LED display, a 12-key keyboard input, a character LCD display, DC- and Stepping Motor controls, a seven-segment display.

ภาควิชา วิศวกรรมไฟฟ้า ลายมือชื่อนิสิต .....
สาขาวิชา วิศวกรรมไฟฟ้า ลายมือชื่ออาจารย์ที่ปรึกษา .....
ปีการศึกษา 2542 ลายมือชื่ออาจารย์ที่ปรึกษาร่วม .....

กิตติกรรมประกาศ

วิทยานิพนธ์ฉบับนี้สำเร็จลุล่วงไปได้ด้วยความช่วยเหลืออย่างดียิ่งของ รองศาสตราจารย์กฤษดา วิศวธีรานนท์ อาจารย์ที่ปรึกษา และศาสตราจารย์ ดร.มงคล เดชนครินทร์ อาจารย์ที่ปรึกษาร่วม ซึ่งได้ให้คำแนะนำและข้อคิดเห็นต่างๆ พร้อมทั้งติดต่อกับหน่วยงานต่าง ๆ เพื่อหาข้อมูลในการทำวิทยานิพนธ์ จึงใคร่ขอขอบพระคุณมา ณ โอกาสนี้

ข้าพเจ้าขอขอบคุณคณาจารย์ทุกท่านที่ได้อบรมสั่งสอนศิษย์เรื่อยมา ตลอดจนเพื่อนๆ พี่ๆ และน้องๆ ห้องปฏิบัติการออกแบบวงจรอิเล็กทรอนิกส์ ที่ได้ให้ความช่วยเหลือในการให้ยืมเครื่องมือต่างๆที่ใช้ในการทดลอง

ท้ายที่สุดนี้ ข้าพเจ้าขอกราบขอบพระคุณบิดามารดา และผู้เกี่ยวข้องที่ได้สนับสนุนในด้านต่างๆ ทุกๆ ด้านและให้กำลังใจให้แก่ข้าพเจ้าเสมอจนสำเร็จการศึกษา

อมรพงษ์ ชุมสาย ณ อยุธยา

สารบัญ

หน้า

สารบัญภาพ

หน้า

สารบัญตาราง

หน้า

บทที่ 1

บทนำ

1.1 แนวเหตุผล

ในปัจจุบันเทคโนโลยีทางอิเล็กทรอนิกส์ได้ก้าวหน้าไปมาก จากระบบควบคุมแบบเก่าที่ใช้แอนะล็อก (Analog) ก็พัฒนามาใช้ระบบดิจิทัล (Digital) ทำให้มีความแม่นยำสูงซึ่งต้องใช้ระบบไมโครคอนโทรลเลอร์มาควบคุมและมีความยืดหยุ่นในการโปรแกรมสูง ไม่ยึดติดกับฮาร์ดแวร์ แต่ระบบไมโครคอมพิวเตอร์ที่ใช้ในการควบคุมนั้นเป็นระบบใหญ่ ฉะนั้นการเรียนรู้ควรเริ่มรู้จากระบบเล็ก ๆ และง่าย ๆ ก่อน ซึ่งก็คือ ไมโครคอนโทรลเลอร์อันเป็นพื้นฐานในการศึกษาระบบใหญ่ ๆ ต่อไป

ปัญหาและที่มาของวิทยานิพนธ์

ในปัจจุบันได้มีหลายบริษัทพยายามแข่งขันในการสร้างผลิตภัณฑ์ที่เกี่ยวกับไมโครคอนโทรลเลอร์ออกมาจำหน่าย โดยสินค้าที่จำหน่ายมีทั้งไมโครคอนโทรลเลอร์แผงเดี่ยวที่รวมเอาทั้งส่วนแสดงผลและส่วนป้อนโปรแกรมจัดอยู่บนแผงเดียวกัน สามารถใช้งานได้ทันทีโดยอิสระ ป้อนโปรแกรมเป็นภาษาแอสเซมบลี และแสดงผลออกหน้าจอแอลซีดีหรือจอแอลอีดีแบบ 7 ส่วน (7 Segments) และอีกประเภทหนึ่งคือ แผงไมโครคอนโทรลเลอร์ การใช้งานแผงไมโครคอนโทรลเลอร์ต้องอาศัยไมโครคอมพิวเตอร์ในการพัฒนาโปรแกรมคือ ใช้โปรแกรมเทอร์มินัลบนคอมพิวเตอร์ติดต่อกับแผงไมโครคอนโทรลเลอร์เพื่อเขียนพัฒนาโปรแกรมต่าง ๆ ทางพอร์ต RS-232 แต่ด้วยวิธีใช้งานสินค้าทั้ง 2 ประเภทนี้ เกิดปัญหาที่เป็นอุปสรรคในการเรียนรู้คือ

- ไมโครคอนโทรลเลอร์แผงเดี่ยว ใช้ภาษาแอสเซมบลีในการพัฒนาโปรแกรม โดยภาษาระดับต่ำนี้จะเป็นเอกลักษณ์ ซึ่งขึ้นอยู่กับเบอร์และบริษัทของผู้ผลิตซีพียูซึ่งแตกต่างกันออกไป ดังนั้นหากผู้ศึกษามีความประสงค์จะเปลี่ยนบริษัทผู้ผลิตจำเป็นต้องเรียนรู้เกี่ยวกับชุดคำสั่งใหม่ ๆ ของผู้ผลิตนั้น ๆ ทำให้ต้องเสียเวลาในการศึกษาโครงสร้างภายในและรายละเอียดในแต่ละชุดคำสั่ง
- แผงไมโครคอนโทรลเลอร์ ต้องใช้ไมโครคอมพิวเตอร์ในการพัฒนาในหลาย ๆ สถาบันการศึกษาที่มีนักศึกษาจำนวนมากมีจำนวนไมโครคอมพิวเตอร์จำนวนในการสอนไม่เพียงพอ รวมทั้งผู้เรียนไม่มีไมโครคอมพิวเตอร์ใช้ เนื่องจากราคาไมโครคอมพิวเตอร์ในปัจจุบันมีราคาสูง ดังนั้นจึงทำให้การเรียนรู้ไม่เข้าถึงนักศึกษาเหล่านี้ ปัญหานี้จึงเป็นอีกปัญหาหนึ่งที่เป็นอุปสรรคในการเรียนรู้

ดังนั้นปัญหาต่าง ๆ ที่กล่าวมาผู้วิจัยได้ทำการคิดค้นหาทางแก้ไขปัญหาและอุปสรรคในการพัฒนาโปรแกรมไมโครคอนโทรลเลอร์ และนี่คือแหล่งที่มาของวิทยานิพนธ์ฉบับนี้

แนวคิดของวิทยานิพนธ์

จากปัญหาข้างต้นที่ได้กล่าวมาแล้ว ผู้วิจัยได้มีแนวคิดที่จะนำเอาภาษาระดับสูงที่เป็นมาตรฐานในกันอย่างแพร่หลาย และมีคำสั่งพิเศษที่ช่วยในการพัฒนาไมโครคอนโทรลเลอร์ ภาษาที่มีขนาดโปรแกรมขนาดเล็กที่สามารถบรรจุในตัวซีพียูก็คือ ภาษาเบสิก (BASIC) ชุดคำสั่งมีไม่มากและใกล้เคียงกับภาษามนุษย์ในชีวิตประจำวัน ติดตั้งบนแผงควบคุม ซึ่งผู้วิจัยขอเรียกว่า แผงอีดีแอลเบสิก (EDL BASIC) และจำลองไมโครคอมพิวเตอร์ที่ต่อกับแผงอีดีแอลเบสิกที่ใช้ในการพัฒนาโปรแกรมมาเป็นแผงอีดีแอลเทอร์มินัลแทนไมโครคอมพิวเตอร์ แผงอีดีแอลเทอร์มินัล (EDL terminal board) จะมีต่อกับแผงแป้นอักขระของคอมพิวเตอร์ซึ่งหาซื้อได้ง่ายและราคาถูก พร้อมทั้งมีจอแสดงผลแบบแอลซีดีที่สามารถแสดงผลได้อย่างหลากหลาย และยังได้จัดทำแผงฝึกทดลองที่ใช้ร่วมกับแผงแอลอีดีเบสิกในการทดลองต่าง ๆ โดยมีแบบเรียนแนวใหม่สำหรับใช้ทดลองไปด้วย ซึ่งเป็นชุดฝึกทดลองแนวใหม่ที่ยังไม่มีในประเทศไทยและต่างประเทศ โดยรายละเอียดจะกล่าวในบทต่อไป

1.2 วัตถุประสงค์

1.2.1 เพื่อศึกษาและออกแบบระบบเรียนรู้ไมโครคอนโทรลเลอร์

1.2.2 สร้างและทดสอบระบบเรียนรู้นี้และสร้างบทเรียนเพื่อนำไปใช้งานจริง

1.3 ขั้นตอนการดำเนินงาน

1.3.1 ศึกษาและรวบรวมข้อมูลเกี่ยวกับชุดฝึกทดลองที่มีใช้กันอยู่ในปัจจุบัน

1.3.2 ศึกษาและทำความเข้าใจทฤษฎีเกี่ยวกับระบบพัฒนาโปรแกรมบนไมโครคอนโทรลเลอร์

1.3.3 กำหนดขอบเขตของระบบที่ต้องการจะสร้าง

1.3.4 ออกแบบฮาร์ดแวร์ (Hardware) ที่เกี่ยวข้องกับบอร์ดไมโครคอนโทรลเลอร์ MCS-51

1.3.5 ออกแบบซอฟต์แวร์ (Software) ของโปรแกรมมอนิเตอร์ภาษาเบสิก

1.3.6 ทดสอบฮาร์ดแวร์ของบอร์ดไมโครคอนโทรลเลอร์

1.3.7 ทดสอบซอฟต์แวร์กับฮาร์ดแวร์ที่ออกแบบให้ทำงานจริง

1.3.8 ออกแบบโมดูลการทดลองต่าง ๆ

1.3.9 ทดสอบโมดูลสำหรับต่อทดลอง

1.3.10 ปรับปรุงแก้ไขข้อบกพร่องของระบบทั้งหมด

1.3.11 สรุปและประเมินผล

1.3.12 เขียนวิทยานิพนธ์

1.3.13 เขียนแบบเรียนแนวใหม่สำหรับการทดลองแต่ละการทดลอง

1.4 ขอบเขตของวิทยานิพนธ์

1.4.1 สร้างและพัฒนาซอฟต์แวร์ตัวแปลภาษาเบสิกแนวใหม่และชุดบอร์ดคอนโทรลเลอร์

1.4.2 สร้างบอร์ดทดลองที่ใช้กับระบบเรียนรู้แนวใหม่ 5 การทดลอง

1.5 ประโยชน์ที่คาดว่าจะได้รับ

1.5.1 ได้ชุดฝึกทดลองภาษาเบสิกแนวใหม่ซึ่งอาจสามารถนำไปใช้งานจริง

1.5.2 ได้ชุดบทเรียนแนวใหม่ที่จะนำไปใช้กับบอร์ดไมโครคอนโทรลเลอร์ที่มีโปรแกรมตัวแปลภาษาเบสิก

บทที่ 2

ระบบพัฒนาโปรแกรมไมโครคอนโทรลเลอร์

ระบบไมโครโทรลเลอร์ (Microcontroller System) ถูกนำมาใช้อย่างแพร่หลายในงานควบคุมต่าง ๆ ตั้งแต่งานควบคุมง่าย ๆ ไปจนถึงงานที่มีความซับซ้อน เช่น การควบคุมวงจรลำดับ การควบคุมกระบวนการผลิตในโรงงานอุตสาหกรรม การควบคุมแขนกล ฯลฯ ทั้งนี้เนื่องจากว่า มีคุณสมบัติต่าง ๆ ที่เป็นจุดเด่นหลายประการ เช่น

1. สามารถแปลงข้อมูล ให้อยู่ในรูปแบบต่าง ๆ เช่น แปลงข้อมูล จากคีย์บอร์ดไปแสดงผลบนจอภาพ แปลงข้อมูลจากคีย์บอร์ดไปเป็นข้อมูลในหน่วยความจำ ซึ่งทำให้สามารถเก็บข้อมูลได้ จากไอโอ (I/O) ไปพิมพ์ออกที่เครื่องพิมพ์ เป็นต้น
2. สามารถควบคุมการทำงานโดยใช้โปรแกรม ทำให้มีความยืดหยุ่นสูง ไม่ติดอยู่กับฮาร์ดแวร์
3. โปรแกรมสามารถเขียนเป็นภาษาที่มนุษย์เข้าใจง่าย ทำให้ง่ายต่อการเข้าใจการทำงาน
4. มีหน่วยความจำ ทำให้สามารถบันทึกข้อมูลได้
5. ซีพียูมีความสามารถในการดำเนิน ฟังก์ชัน ทางคณิตศาสตร์ ทำให้สามารถประมวลผลข้อมูลที่ซับซ้อนได้
6. สามารถทำงานได้รวดเร็ว ภายใน 1 วินาที อาจปฏิบัติคำสั่งได้นับล้านคำสั่ง

การเรียนรู้การนำระบบไมโครโพรเซสเซอร์มาใช้ในการควบคุมจึงเป็นสิ่งสำคัญ อุปกรณ์ที่ใช้ในการเรียนรู้ คือ ชุดฝึกทดลอง

2.1 ชุดฝึกทดลอง

ชุดฝึกทดลอง คือ เครื่องมือที่ใช้ในการเรียนรู้วิธีการนำระบบไมโครโพรเซสเซอร์มาใช้ในการควบคุม ซึ่งจะประกอบด้วยส่วนหลัก ๆ 2 ส่วน ส่วนแรกคือระบบไมโครโพรเซสเซอร์หรือระบบไมโครคอมพิวเตอร์ ที่ทำหน้าที่ในการควบคุม ซึ่งจะมีส่วนของการป้อนข้อมูลเข้าเพื่อรับโปรแกรมในการควบคุม ส่วนที่สองเป็นอุปกรณ์ต่าง ๆ ที่ต่อเข้ากับระบบไมโครโพรเซสเซอร์เพื่อเป็นเป้าหมายของการควบคุมเพื่อใช้ในการเรียนรู้วิธีการควบคุมแบบต่าง ๆ

2.2 ความเป็นมาของชุดฝึกทดลอง

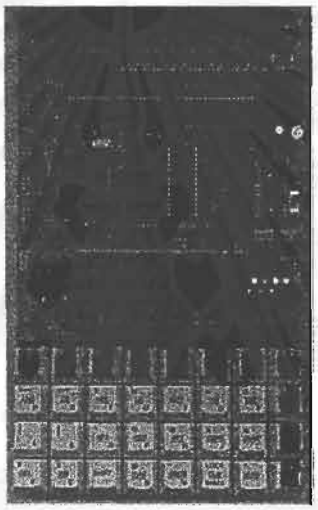
ชุดฝึกทดลองไมโครโพรเซสเซอร์ในสมัยแรกมักจะเป็นแบบ ไมโครโพรเซสเซอร์บอร์ดเดี่ยว (Single-Board Microprocessor) ซึ่งใช้ซีพียูขนาด 8 บิต ป้อนโปรแกรมด้วยภาษาเครื่อง มีหน่วยความจำขนาดเล็ก เช่น ET-Board ของบริษัทอีทีที (ETT)[2] เป็นต้น ต่อมาได้มีการพัฒนาให้สามารถต่อเข้ากับเทอร์มินัล และใช้เทอร์มินัลในการป้อนโปรแกรมและแสดงผล โดยใช้โปรแกรมมอนิเตอร์ หลังจากนั้นได้มีการพัฒนาให้สามารถเขียน

โปรแกรมด้วยภาษาระดับสูงได้เช่น ภาษาเบสิกและภาษาซี ในที่นี้ไมโครโพรเซสเซอร์บอร์ดเดี่ยว หมายความรวมถึงกลุ่มของไมโครคอนโทรลเลอร์ด้วย

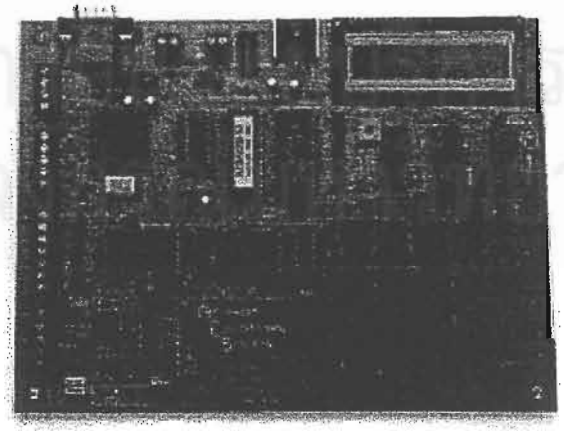
ไมโครโพรเซสเซอร์บอร์ดเดี่ยวมีจุดเด่นที่มีราคาถูกและมีขนาดเล็ก และมีระบบที่ไม่ซับซ้อน ทำให้เรียนรู้ได้ง่าย และเป็นพื้นฐานที่ดีในการเรียนรู้ระบบที่ซับซ้อนยิ่ง ๆ ขึ้นไป

ตัวอย่างไมโครโพรเซสเซอร์ และ ไมโครคอนโทรลเลอร์บอร์ดเดี่ยว รุ่นต่าง ๆ ที่มีจำหน่ายในท้องตลาด ได้แก่

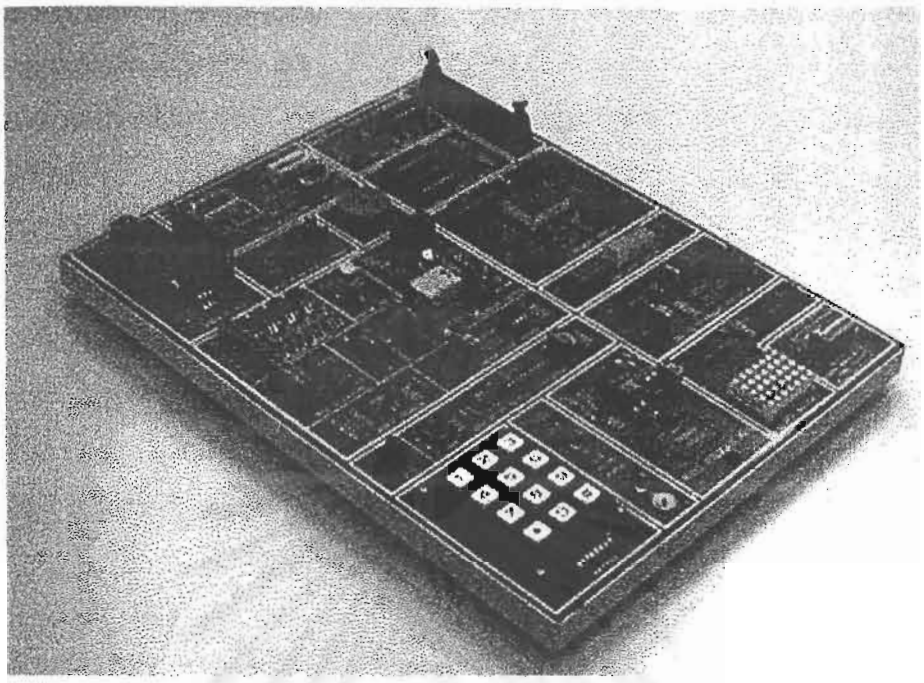
- ET-Board Version 5.0 ของบริษัทอีทีที ตามรูปที่ 2.1
- AES-51 ของบริษัท Advanced Electronic System (AES)[9] ตามรูปที่ 2.2
- Multi Interface ของบริษัทแอนาดิจิท ตามรูปที่ 2.3



รูปที่ 2.1 ชุดฝึกทดลอง ET-BOARD ของบริษัทอีทีที



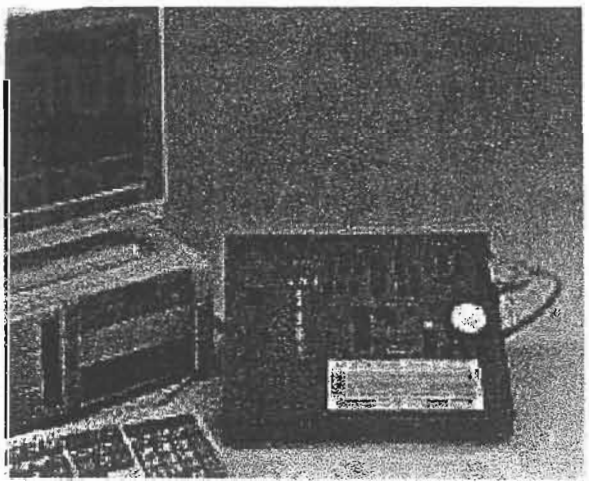
รูปที่ 2.2 ชุดฝึกทดลอง AES-51



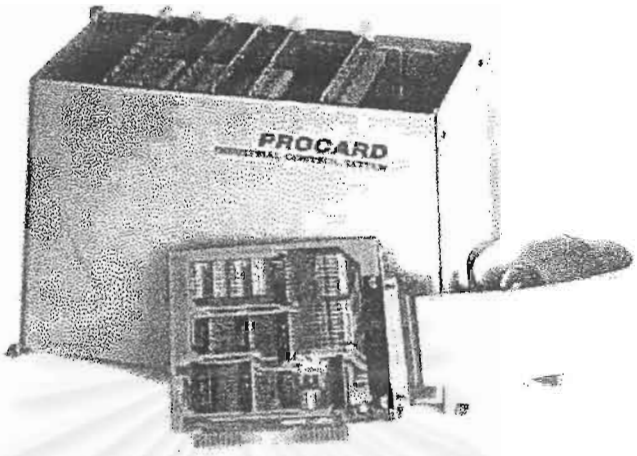
รูปที่ 2.3 ชุดฝึกทดลอง Multi Interface Board ของบริษัทแอนนาดิจิท

ในระยะหลัง ๆ เครื่องไมโครคอมพิวเตอร์มีราคาถูกลงอย่างมาก และใช้งานเป็นที่แพร่หลาย อีกทั้งวิทยาการทางด้านไมโครคอมพิวเตอร์ก็เป็นที่เปิดกว้าง จึงได้มีการพัฒนาชุดฝึกทดลองที่ใช้ไมโครคอมพิวเตอร์เป็นตัวควบคุมแทนการใช้ไมโครโพรเซสเซอร์บอร์ดเดี่ยว ตัวอย่างชุดฝึกทดลองชนิดนี้ได้แก่

- CIC-100 ของบริษัท King Instrument Electronics[17] ตามรูปที่ 2.4
- ICS ของบริษัทแอนนาดิจิท[7] ตามรูปที่ 2.5



รูปที่ 2.4 ชุดฝึกทดลอง CIC-100



รูปที่ 2.5 ชุดฝึกทดลอง ICS

2.3 ประเภทของชุดฝึกทดลอง

ชุดฝึกทดลองที่ใช้ในการเรียนรู้การควบคุมด้วยระบบไมโครโพรเซสเซอร์ สามารถแบ่งได้เป็น 2 ประเภท คือ ชุดฝึกทดลองไมโครโพรเซสเซอร์บอร์ดเดี่ยว และ ชุดฝึกทดลองไมโครคอมพิวเตอร์ มีรายละเอียดดังนี้

2.3.1 ชุดฝึกทดลองไมโครโพรเซสเซอร์บอร์ดเดี่ยว (Single-Board Microprocessor)

ชุดฝึกทดลองไมโครโพรเซสเซอร์บอร์ดเดี่ยว คือ ชุดฝึกทดลองที่ใช้บอร์ดไมโครโพรเซสเซอร์เป็นตัวควบคุมหลัก ได้แก่ AES-51[23] ของบริษัท AES, Z 80 Interface Training ZI-1 ของบริษัทแอนนาดิจิท, V-50 CPU-Board และชุดฝึกทดลองแบบอื่น ๆ

1. AES-51 System with Interface

ชุดฝึกทดลองนี้ ซึ่งผลิตโดยบริษัท AES มีลักษณะ ดังนี้

- CPU 8032
- I/O parallel 48 line 8255 x 2
- ใช้ timer 8253
- อินเทอร์รัปต์ 5 ระดับ
- โมดูลการทดลอง : โมดูลแผงแป้นอักขระ, โมดูลแสดงผล, โมดูล DAC คู่, แผงใช้โมดูล DAC, โมดูล DAC แบบความชันคู่, โมดูลลิฟต์ และ ตัวควบคุมตรรกะ

2. Z-80 Interface Training ZI-1

ผลิตโดยบริษัทแอนนาดิจิทกรุ๊ป จำกัด มีลักษณะดังนี้

- ต่อเข้ากับบอร์ดไมโครโพรเซสเซอร์ Z-80 โดยใช้ STD Bus
- โมดูลการทดลอง 6 โมดูล ประกอบด้วย Switch & Display, TTL I/O, Parallel I/O, Serial I/O, Digital to Analog, Analog to Digital
- มีการทดลองประมาณ 20 การทดลอง

3. V-50 CPU-Board

V-50 CPU-Board มีลักษณะดังนี้

- PPI 8255 x 4
- DIP Switch 8 Bit x 2
- LED x 8
- 7 Segment LED x 4
- No-Chattering Switch x 2
- Frequency Counter, Photo- Interrupter
- Stepping Motor, A/D Converter (8 Channels), D/A Converter

4. ไมโครโพรเซสเซอร์บอร์ดเดี่ยว (Single-Board Microprocessor) อื่น ๆ เช่น

4.1) 8032 Single-Board Microprocessor ของบริษัท อีทีที[2] ใช้งานร่วมกับ ET-01 ซึ่งประกอบด้วย

- A/D, D/A ขนาด 8 บิต การควบคุมสเตปปิงมอเตอร์ (Stepping Motor) และมอเตอร์กระแสตรง
- Solid-State Relay 220 V, Dot Matrix LED ขนาด 8x8 จุด
- และอุปกรณ์อื่น ๆ เพิ่มเติม เช่น ET-RS232 ใช้สำหรับรับ-ส่งข้อมูลทางพอร์ตอนุกรมที่ระดับสัญญาณของ RS-232
- 72 IO Z-80 เป็น Port -8255 x 3 ใช้เป็น พอร์ตรับ-ส่งข้อมูลเข้า/ออกได้ 72 บิต
- ET-AD ใช้สำหรับแปลงสัญญาณจากแอนะล็อกเป็นดิจิทัล
- RTC เป็นบอร์ดสำหรับสร้างฐานเวลาที่ใช้ในชีวิตประจำวัน

4.2) MCS-51 Single-Board ของบริษัทแอนนาดิจิทกรุ๊ป ซึ่งใช้ร่วมกับแผงทดลองอื่น ๆ เช่น LED ขนาด 8x8 จุด และ RTC

2.3.2 ชุดฝึกทดลองไมโครคอมพิวเตอร์

ชุดฝึกทดลองไมโครคอมพิวเตอร์ คือ ชุดฝึกทดลองที่ใช้ไมโครคอมพิวเตอร์เป็นตัวควบคุม ประกอบด้วยอุปกรณ์ภายนอก ที่ต่อเข้ากับเครื่องไมโครคอมพิวเตอร์ ชุดฝึกทดลองประเภทนี้จะสังเกตได้ว่าไม่มีซีพียูที่อยู่ภายนอก เนื่องจากใช้ซีพียูที่อยู่ภายในเครื่องไมโครคอมพิวเตอร์เป็นตัวควบคุม

กลุ่มของชุดฝึกทดลองไมโครคอมพิวเตอร์ ได้แก่ CIC-100[17], Industrial Control System (ICS) [25] โดยมีรายละเอียดของแต่ละชุดฝึกทดลอง ดังนี้

1) CIC-100

เป็นชุดฝึกทดลองของบริษัท K&H มีลักษณะ คือ

- ต่อกับ IBM PC XT/ATหรือ เครื่องที่เข้ากันได้
- ใช้ภาษาเบสิก, แอสเซมบลี หรือ ภาษาซี ในการควบคุม
- ต่อกับ IBM PC โดยผ่านแผงวงจรที่เสียบกับร่องเสียบ (slot)
- ประกอบด้วย 11 การทดลองย่อย
- ตัวขับสัญญาณเข้า/สัญญาณออก, รีเลย์, LED, สวิตช์, ลำโพง
- Analog Signal Input : Light, Temperature, Sound, External Signal
- 2-PPI (Programmable Peripharal Interface) ซึ่งเป็นไอซีเบอร์ 8255A
- โมดูลประกอบด้วย ตัวควบคุมสเตปปิงมอเตอร์, Key-Pad Matrix and Traffic-light Module, ADC Module, DAC Module, VFC Module และ FVC Module

2) Industrial Control System (ICS)

ผลิตโดยบริษัทแอนนาดิจิทกรุ๊ป จำกัด มีลักษณะดังนี้

- ต่อกับ IBM PC ผ่านทางแผงวงจรที่เสียบกับร่องเสียบ
- ควบคุมด้วยภาษาเบสิก, ปาสคาล, แอสเซมบลี หรือภาษาซี
- โมดูลประกอบด้วย Relay Control Board, Opto-Isolator Board, A/D Converter Board, D/A Converter Board (Voltage), D/A Converter Board (Current), I/O Board, Simpro Bus Interface, Switch & Display Module, Joystick Module, DC Motor Module, Stepping Motor Module, Temperature Control Module, Traffic Module

3) การนำแผงเสียบต่าง ๆ มาใช้งานร่วมกัน

เป็นการนำแผงเสียบที่ผู้ผลิตรายต่าง ๆ ได้ผลิตขึ้นเพื่อวัตถุประสงค์อื่น ๆ มาประยุกต์ใช้ในการ

ทดลอง โดยแผงเสียบแผงหนึ่ง ๆ อาจครอบคลุมหัวข้อ 1-2 การทดลอง เช่น ในหนังสือ Laboratory Automations ใช้แผงเสียบ DT-2801 ในการทดลองเรื่อง A/D, D/A และใช้บอร์ด DACA (Data Acquisition and Control Adapter) ในการทดลองเรื่อง ตัวจับเวลาและตัวนับ A/D, D/A

นอกจากนี้ ยังมีแผงวงจร A/D, D/A ของบริษัทแอนนาดิจิทที่ใช้ผ่านร่องเสียบของ IBM PC แผงวงจรเหล่านี้ออกแบบมาเพื่อใช้ในงานอุตสาหกรรม

ไมโครโพรเซสเซอร์บอร์ดเดี่ยว และ ไมโครคอมพิวเตอร์ มีข้อแตกต่างกัน ดังนี้

ตารางที่ 2.1 ข้อแตกต่างระหว่างไมโครโพรเซสเซอร์บอร์ดเดี่ยว และ ไมโครคอมพิวเตอร์

ไมโครโพรเซสเซอร์บอร์ดเดี่ยว	ไมโครคอมพิวเตอร์
1. มีระบบที่ง่าย	1. มีระบบที่ซับซ้อน
2. ใช้โปรแกรมมอนิเตอร์	2. ใช้ระบบปฏิบัติการ
3. มีราคาถูก	3. มีราคาแพง
4. มีโปรแกรมสนับสนุนน้อย	4. มีโปรแกรมสนับสนุนมาก

2.4 วิเคราะห์ข้อดีและข้อเสียของชุดฝึกทดลอง

2.4.1 กลุ่มของชุดฝึกทดลองไมโครโพรเซสเซอร์บอร์ดเดี่ยว (Single-Board Microprocessor)

กลุ่มของชุดฝึกทดลองไมโครโพรเซสเซอร์บอร์ดเดี่ยว มีลักษณะที่พิจารณาแล้ว สรุปได้ดังนี้

1) ใช้ภาษาแอสเซมบลี จึงทำให้เกิดความยากลำบากในการเขียนและพัฒนาโปรแกรมที่มีความซับซ้อน และทำให้เกิดการเรียนรู้การใช้งานในวงแคบ คือ ในขอบเขตของซีพียูเบอร์นั้น ๆ เท่านั้น ไม่สามารถนำไปใช้กับเครื่องอื่น ๆ ได้
2) การป้อนโปรแกรม มักจะต้องป้อนเป็นภาษาเครื่อง ผ่านทางแผงแป้นกดของฝึกทดลอง ซึ่งมีขนาดเล็ก (โดยทั่วไปอยู่ระหว่าง 24-32 แป้น)
3) ถ้าไม่ใช้วิธีป้อนโปรแกรมด้วยภาษาเครื่อง มักจะต้องใช้ร่วมกับเทอร์มินัล ซึ่งปัจจุบันมักใช้ IBM PC มาดัดแปลงเป็นเทอร์มินัล ทำให้ลดสมรรถนะการใช้งานของ IBM PC ลง
4) ซีพียูที่ใช้บนแผงไมโครโพรเซสเซอร์บอร์ดเดี่ยวมักเป็นซีพียูรุ่นเก่า ๆ หรือเป็นรุ่นที่ถูกพัฒนาขึ้นมา แต่มีแนวโน้มที่จะเลิกใช้งานในอนาคต สังเกตได้ว่ามีการพัฒนาโดยการเปลี่ยนแปลงเล็กน้อยอยู่บ่อยครั้ง
5) การแสดงผล มักใช้ตัวแสดงผลแบบ 7 ส่วน (7-Segment Display) หรือจอแอลซีดีทำให้ไม่สามารถแสดงผลได้อย่างหลากหลาย
6) กรณีใช้งานร่วมกับเทอร์มินัล ทำให้ไม่สามารถติดตามการทำงานของระบบแบบเวลาจริงได้ ถ้าต้องการผลดังกล่าว ต้องใช้ ICE (In-Circuit Emulator) ร่วมกับแผงไมโครโพรเซสเซอร์

บอร์ดเดี่ยว ซึ่งตัวจำลองการทำงานมีราคาแพงมาก ตัวอย่างเช่น ชุดฝึกทดลองของ AES-51 ของ AES[9] ที่ใช้ร่วมกับ ICE-51

7) พื้นที่หน่วยความจำมักมีขนาดเล็ก เช่น มี RAM ขนาด 8K และ ROM ขนาด 32K ทำให้ไม่สามารถพัฒนาโปรแกรมขนาดใหญ่ หรือเก็บข้อมูลจำนวนมาก ๆ ได้

2.4.2 กลุ่มของชุดฝึกทดลองไมโครคอมพิวเตอร์ (Microcomputer)

กลุ่มของชุดฝึกทดลองไมโครคอมพิวเตอร์ คือ กลุ่มที่ใช้ไมโครคอมพิวเตอร์แทนระบบไมโครโพรเซสเซอร์ในการควบคุมการทำงานหลัก เช่น เครื่องไมโครคอมพิวเตอร์ IBM PC เป็นต้น

ต่อไปนี้จะกล่าวถึงชุดฝึกทดลองไมโครคอมพิวเตอร์ ที่จำหน่ายในท้องตลาดที่ใช้กับเครื่องไมโครคอมพิวเตอร์ IBM PC

1) CIC-100 ของ K&H

- ใช้งานผ่านแผงเสียบผ่านร่องเสียบของเครื่อง IBM PC โดยที่ร่องเสียบที่ใช้เป็นแบบ ISAซึ่งมีแนวโน้มจะเลิกใช้งานบนเครื่อง IBM PC ในอนาคต และการใช้งานผ่านร่องเสียบทำให้เกิดความยุ่งยากในการติดตั้ง
- ผลิตภัณฑ์ดังกล่าว ผลิตจากต่างประเทศ จึงมีราคาสูง
- ชุดทดลองมีขนาดใหญ่ ไม่สะดวกในการเคลื่อนย้าย

2) ICS ของบริษัทแอนนาดิจิทกรุ๊ป

- ใช้งานผ่านแผงวงจรที่เสียบกับร่องเสียบของเครื่อง IBM PC โดยมีเหตุผลเหมือนกับ CIC-100
- ชุดทดลองมีขนาดใหญ่ ไม่สะดวกในการเคลื่อนย้าย

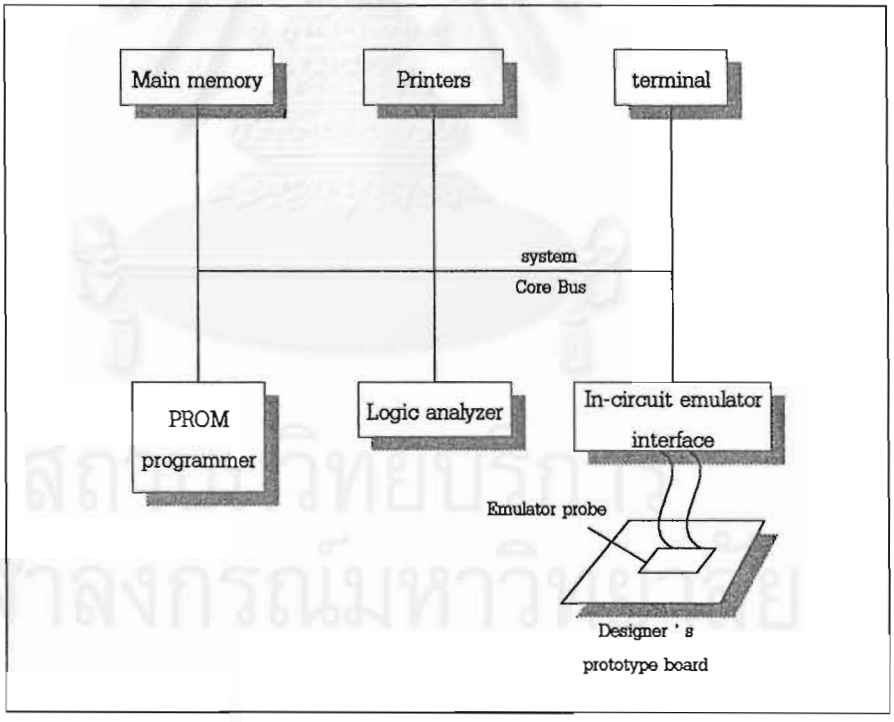
3) พวกที่นำแผงเสียบต่าง ๆ มาใช้งานร่วมกัน

กลุ่มนี้มีราคาค่อนข้างแพงเนื่องจากถูกผลิตขึ้นมาเพื่อใช้ในงานอุตสาหกรรม ที่มีความเที่ยงตรงสูง ไม่เหมาะในการนำมาใช้เป็นชุดฝึกทดลอง สามารถสรุปข้อเสียได้ดังนี้

- มีราคาแพง
- ไม่มีคู่มือการทดลองประกอบ
- กินเนื้อที่หลายร่องเสียบของเครื่อง IBM PC หรือ ต้องเปลี่ยนแผงเสียบเข้าและออกบ่อย ๆ
- ไม่ครอบคลุมหัวข้อการเรียนรู้

2.5 ระบบพัฒนาโปรแกรมไมโครคอนโทรลเลอร์

ระบบพัฒนาโปรแกรมไมโครคอนโทรลเลอร์ หรือ Microcontroller Development System (MDS) เป็นระบบที่จำเป็นในการออกแบบและพัฒนาระบบที่ใช้ไมโครคอนโทรลเลอร์ ยิ่งระบบไมโครคอนโทรลเลอร์ที่จะออกแบบมีความซับซ้อนมากเท่าไร ความจำเป็นที่ต้องมีระบบช่วยในการพัฒนาซอฟต์แวร์และเทคนิคในการแก้จุดบกพร่องของฮาร์ดแวร์ก็ยิ่งมีมากขึ้นเท่านั้น ระบบพัฒนาโปรแกรมไมโครคอนโทรลเลอร์ที่ใช้จะมีตั้งแต่ขนาดเล็กราคาไม่แพงนัก และมีขีดความสามารถจำกัดซึ่งใช้พัฒนาได้เฉพาะทางด้านซอฟต์แวร์ จนถึงระบบพัฒนาโปรแกรมไมโคร-คอนโทรลเลอร์ขนาดใหญ่ที่สามารถสนับสนุนการพัฒนาได้ทั้งทางด้านฮาร์ดแวร์และซอฟต์แวร์จากผู้ใช้หลาย ๆ คนในขณะเวลาเดียวกัน ผู้ที่สร้างระบบพัฒนาโปรแกรมไมโครคอนโทรลเลอร์จะมีอยู่ 2 กลุ่ม คือ กลุ่มผู้ผลิตอุปกรณ์ และกลุ่มผู้ที่ใช้อุปกรณ์ ระบบพัฒนาโปรแกรมที่สร้างจากกลุ่มผู้ผลิตไมโครคอนโทรลเลอร์ จะใช้ได้เฉพาะการพัฒนาระบบที่ใช้ไมโครคอนโทรลเลอร์เบอร์นั้น ๆ (Nonuniversal System) ส่วนระบบพัฒนาโปรแกรมที่สร้างจากกลุ่มผู้ผลิตหรือกลุ่มผู้ใช้ จะสามารถรองรับการพัฒนาได้หลายเบอร์ (Universal Systems) ระบบพัฒนาโปรแกรมไมโครคอนโทรลเลอร์ระบบหนึ่ง ๆ ประกอบด้วยฮาร์ดแวร์ที่จำเป็นในการออกแบบระบบไมโครคอนโทรลเลอร์ และชุดของโปรแกรมหรือซอฟต์แวร์ที่จำเป็นในการทำงานของฮาร์ดแวร์ ระบบพัฒนาโปรแกรมไมโครคอนโทรลเลอร์ส่วนที่เป็นฮาร์ดแวร์แสดงได้ดังรูปที่ 2.6 ได้แก่



รูปที่ 2.6 องค์ประกอบที่เป็นฮาร์ดแวร์ของระบบพัฒนาโปรแกรมไมโครคอนโทรลเลอร์

1. หน่วยความจำหลัก (Main Memory) เป็นหน่วยความจำที่ตัวประมวลผลของระบบพัฒนาโปรแกรมเข้าถึงได้โดยตรงซึ่งจะใช้เก็บซอฟต์แวร์ของการพัฒนาโปรแกรม เช่น เอดิเตอร์ (Editor) แอสเซมเบลอร์ (Assembler) และตัวเลียนแบบ (Emulator) เป็นต้น หน่วยความจำหลักมักมีขนาดตั้งแต่ 32 K จนถึง 64

K และประกอบด้วยรอม (ROM) และแรม (RAM) โดยปกติรอมจะมีโปรแกรมมอนิเตอร์ (Monitor) ซึ่งใช้ในตัวอุปสาน (Interface) ซอฟต์แวร์ของระบบกับฮาร์ดแวร์ของระบบ

2. เทอร์มินัล (Terminal) เป็นตัวที่ใช้ในการอินเทอร์เฟซระหว่างผู้ใช้กับ OS โดยผู้ใช้ป้อนอินพุตผ่านทางแผงแป้นอักขระ (CRT Keyboard) และ OS แสดงข้อมูลบนจอ (CRT Screen) ข้อมูลที่แสดงบนจอได้แก่ สถานะของตัวประมวลผล, โปรแกรมที่แก้จุดบกพร่องและข้อความที่แสดงข้อผิดพลาดในระหว่างการแก้จุดบกพร่องเป็นต้น ปกติเทอร์มินัลจะเชื่อมต่อกับระบบพัฒนาการรับส่งอนุกรม ซึ่งมีอัตราเร็วโดยเฉลี่ยประมาณ 9,600 - 19,200 บิตต่อวินาที

3. ตัวเลียนแบบในวงจร (In-Circuit Emulator: ICE) ระบบพัฒนาโปรแกรมจะมี ICE อย่างน้อยหนึ่งตัว ICE เป็นเครื่องมือที่ก้าวหน้ามากที่สุด สำหรับการพัฒนาฮาร์ดแวร์ของไมโครคอนโทรลเลอร์ให้เป็นเครื่องมือทดสอบ ในการใช้งานเราจะถอดเอาตัวไมโครคอนโทรลเลอร์ของระบบที่พัฒนา (Target Controller) ออก และนำเอาสายต่อจากอีมูเลเตอร์ไปเสียบแทนที่ อีมูเลเตอร์จะทำงานเลียนแบบการทำงานของตัวไมโครคอนโทรลเลอร์ถอดออกไปภายใต้การควบคุมของซอฟต์แวร์ของระบบพัฒนาโปรแกรม เครื่องสามารถทำงานร่วมกับฮาร์ดแวร์ของระบบที่จะพัฒนา (Target System) และแสดงข้อมูลของสถานะต่าง ๆ เกี่ยวกับกระบวนการทำงานได้ ทั้งยังสั่งให้ทำงานทีละคำสั่งได้อีกด้วย ผู้ใช้ ICE สามารถหาจุดบกพร่องของฮาร์ดแวร์และซอฟต์แวร์อย่างรวดเร็ว ซึ่งอาจต้องใช้เวลาหลายชั่วโมงหากใช้อุปกรณ์ทดสอบอื่น ๆ

4. เครื่องวิเคราะห์ตรรกะ (Logic Analyzer) เป็นเครื่องมือในการแก้จุดบกพร่องฮาร์ดแวร์ที่ดีมากอีกอย่างหนึ่ง ซึ่งผู้ใช้สามารถดูผลของระดับตรรกะที่จะเกิดขึ้นในการทำงานจริงของฮาร์ดแวร์ได้ เครื่องวิเคราะห์ตรรกะยังสามารถตรวจจับและแสดงสัญญาณรบกวนต่าง ๆ ที่เป็นสาเหตุให้ฮาร์ดแวร์ไม่ทำงานด้วยโดยจะสุ่มสัญญาณอินพุตซึ่งปกติมี 16 ถึง 32 เส้นที่จังหวะสัญญาณนาฬิกาต่าง ๆ และเก็บเป็นค่าฐานสอง (Binary) ในหน่วยความจำ เพื่อจะนำมาแสดงเป็นลำดับของสัญญาณ "1" และ "0" บนจอ CRT

5. เครื่องโปรแกรมอีพรอม (EPROM Programmer) หลังจากพัฒนาและแก้จุดบกพร่องซอฟต์แวร์ของ ระบบที่จะพัฒนาก็จำเป็นต้องมีที่เก็บโปรแกรมเพื่อใช้ระบบที่พัฒนาขึ้นมาในงานจริง เครื่องโปรแกรมอีพรอมจะนำเอาโปรแกรมภาษาเครื่องไปโปรแกรมลงในอีพรอม

6. เครื่องพิมพ์ (Line Printer) โปรแกรมที่พัฒนาและข้อมูลต่าง ๆ จะถูกพิมพ์ออกมาทางเครื่องพิมพ์ เพื่อการเรียกดูและเก็บเป็นข่าวสารข้อมูล

ส่วนซอฟต์แวร์ของระบบพัฒนาไมโครคอนโทรลเลอร์ได้แก่

1. ระบบปฏิบัติการ (Operating System) OS เป็นตัวจัดการเกี่ยวกับซอฟต์แวร์ที่ใช้พัฒนาระบบไมโครคอนโทรลเลอร์โดยจะติดต่อกับฮาร์ดแวร์ของระบบผ่านทางโปรแกรมย่อยต่าง ๆ OS จะประกอบด้วยไมโครคอนโทรลเลอร์ที่จัดการเกี่ยวกับหน่วยความจำ โปรแกรมที่ติดต่อกับผู้ใช้และโปรแกรมช่วยต่าง ๆ (Utility Program) ที่ให้ผู้ใช้สามารถจัดการเกี่ยวกับแฟ้มเก็บข้อมูล (file) ได้ เช่น สร้างแฟ้มข้อมูล ลบแฟ้มข้อมูล ทำสำเนาแฟ้มข้อมูล หรือสาระบบ (directory) แฟ้มข้อมูล

2. บรรณาธิกรณ์ (Editor) เป็นโปรแกรมแรกที่ใช้ในการพัฒนาซอฟต์แวร์ ผู้ใช้จะป้อนรหัสต้นทาง Source code ซึ่งเขียนด้วยภาษาแอสเซมบลีหรือภาษาชั้นสูงอื่น ๆ ไปยังระบบที่จะพัฒนาโดยผ่านทางบรรณาธิกรณ์ จะมีคำสั่งสำหรับเพิ่มเติมหรือลบแก้ไขโปรแกรมด้วย

3. แอสเซมเบลอร์ (Assembler) และคอมไพเลอร์ (Compiler) เป็นโปรแกรมที่แปลงรหัสต้นทาง Source code จากป้อนรหัสต้นทาง ซึ่งถูกเก็บไว้ในรูปแบบของตัวอักษร ASCII มาเป็นรหัสจุดหมาย Object code หรือรหัสเครื่อง Machine code ซึ่งเก็บในลักษณะเป็น Relocatable file

4. ตัวเชื่อมโยง (Linker) เป็นโปรแกรมที่เปลี่ยนแฟ้มจุดหมาย Object file ซึ่งเป็นแฟ้มย้ายที่อยู่ได้ ให้เป็นแฟ้มสัมบูรณ์ Absolute file ซึ่งมีรหัสเครื่องที่กำหนดเลขที่อยู่ (Address) แน่นอนแล้ว

5. โปรแกรมบรรจุ (Loader) เป็นโปรแกรมที่นำเอาแฟ้มสัมบูรณ์ Absolute file มาบรรจุลงสู่หน่วยความจำของระบบที่ต้องการจะพัฒนา

6. โปรแกรมตรวจสอบจุดบกพร่อง (Debugger) เป็นโปรแกรมที่ทำให้ผู้ใช้สามารถสั่งดำเนินโปรแกรมภายใต้การควบคุมเงื่อนไขต่าง ๆ และให้ผู้ใช้สามารถควบคุมแบบจำลองของเครื่องต้นแบบได้ โดยทั่วไปจะมีความสามารถ หรือเครื่องมือที่ช่วยในการแก้จุดบกพร่องซอฟต์แวร์ ดังนี้

- Single Step Facility เป็นความสามารถในการทำงานทีละคำสั่งเพื่อติดตามขั้นตอนการทำงานของโปรแกรมอย่างใกล้ชิด
- Breakpoint Facility เป็นความสามารถในการหยุดทำงาน ณ จุดที่กำหนดเพื่อให้สามารถทดลองรันโปรแกรมที่ความเร็วจริงแล้วหยุดพื่อตรวจสอบค่าต่าง ๆ ได้
- Register Dump Facility มีโปรแกรมส่วนที่ให้ผู้ใช้สามารถตรวจสอบแก้ไขค่าในเรจิสเตอร์ผ่านทาง CRT ส่วนเฝ้าคุม (Console)
- Memory Dump Facility มีโปรแกรมส่วนที่ให้ผู้ใช้สามารถตรวจสอบแก้ไขค่าในหน่วยความจำผ่านทาง CRT คอนโซล
- Simulator Program มีโปรแกรมส่วนที่ให้ผู้ใช้สามารถทดสอบรันโปรแกรมและเก็บค่าพารามิเตอร์ต่าง ๆ ไว้เช่น ค่าเลขที่อยู่ ข้อมูลหรือตัวบ่งชี้ (Flag) เป็นต้น

7. มอนิเตอร์ (Monitor) เป็นโปรแกรมที่อำนวยการทำงานทั้งหมดของระบบพัฒนาโดยมีโปรแกรมย่อยที่เป็นตัวเชื่อมต่อระหว่างโปรแกรมต่าง ๆ กับอุปกรณ์ไอโอในระบบ

บทที่ 3

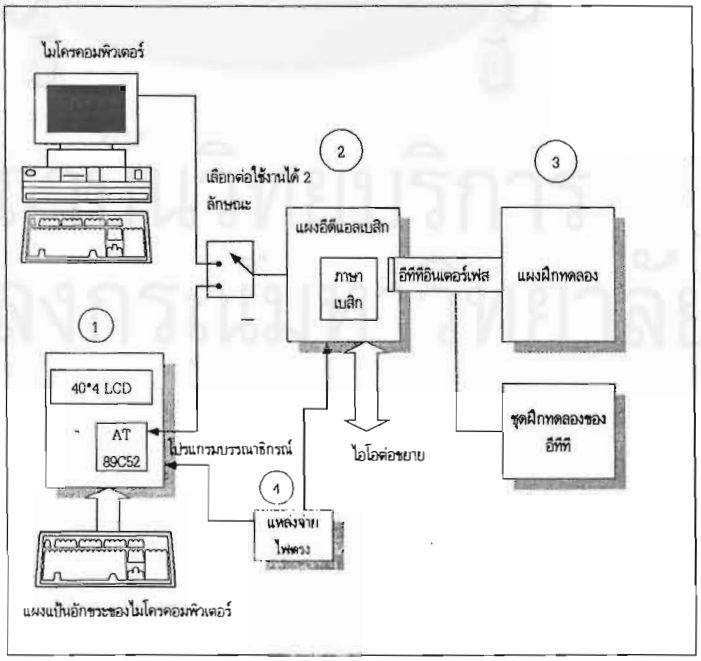
แนวคิดการออกแบบระบบชุดฝึกทดลอง

3.1 แนวคิดการออกแบบระบบชุดฝึกทดลอง

ชุดฝึกทดลองที่ออกแบบขึ้น เป็นระบบไมโครคอนโทรลเลอร์ใช้ภาษาเบสิกในการควบคุมและแก้จุดบกพร่องโปรแกรม โดยในตอนแรกได้กำหนดคุณสมบัติของเครื่องที่จะออกแบบเพื่อให้สามารถใช้งานได้จริง ดังต่อไปนี้

1. ต้องสามารถป้อนข้อมูลเป็นภาษาเบสิกสำหรับควบคุมได้
2. สามารถเก็บข้อมูลเป็นภาษาเบสิกโดยไม่ต้องใช้ไฟเลี้ยง
3. สามารถใช้งานแบบอิสระ (Stand Alone) หรือต่อเข้ากับไมโครคอมพิวเตอร์ได้
4. มีคำสั่งพิเศษเพื่อช่วยในการแก้จุดบกพร่องโปรแกรมและการใช้งาน เช่น LOAD, BRKP, CONT, XBY( ), DBY( ), CBY( ), PROG
5. สามารถเก็บโปรแกรมภาษาเบสิกได้สูงสุด 614 บรรทัด
6. มีจอแสดงผลที่กว้างและสามารถแสดงผลได้หลากหลายขนาด 40 ตัวอักษร 4 บรรทัด
7. ราคาถูกและง่ายต่อการจัดหาอุปกรณ์มาต่อที่อินพุต คือ การนำแผงแป้นอักขระของไมโครคอมพิวเตอร์ทั่วไปมาใช้
8. มีบทเรียนในการศึกษาและทดลองการใช้งานที่เข้าใจง่าย

ระบบทั้งหมดสามารถเขียนได้ดังรูปที่ 3.1



รูปที่ 3.1 ส่วนประกอบของระบบทั้งหมด

จากปัญหาและแนวคิดในบทที่ 1 สามารถแบ่งส่วนประกอบได้เป็น 4 ส่วน ดังนี้

1) แผงอีดีแอลเบสิก

แผงอีดีแอลเบสิกจะบรรจุโปรแกรมตัวแปลภาษาเบสิก (BASIC Interpreter) ขนาด 8 KB ในซีพียู AT89C52 มีหน้าที่รับข้อมูลจาก RS-232 ในรูปแบบของภาษาระดับเบสิกเข้ามาแปลเป็นภาษาเครื่องแล้วกระทำคำสั่ง การใช้งานแผงอีดีแอลเบสิกสามารถใช้เลือกงานพัฒนาโปรแกรมได้ 2 ลักษณะคือ

- ใช้งานร่วมกับแอลอีดีเทอร์มินัลและแผงฝึกทดลอง
- ใช้งานร่วมกับไมโครคอมพิวเตอร์ที่มีโปรแกรมสื่อสารข้อมูลทาง RS-232 พอร์ต เช่น Procomm plus เป็นต้น

แผงอีดีแอลเบสิกนี้ จะมีอีดีแอลเบสิกที่มีคุณสมบัติสำหรับเขียนโปรแกรมที่ใช้ในงานควบคุม เพราะได้เพิ่มฟังก์ชันพิเศษที่ช่วยในการเขียนโปรแกรมเข้าไป จึงแตกต่างจากเบสิกธรรมดาทั่วไป แผงอีดีแอลเบสิกจะมีพอร์ตไอโอสำหรับขยายใช้งานบนแผงผ่าน 8255 และพอร์ตใช้กับแผงฝึกทดลองเพื่อทดลองตามการทดลองต่าง ๆ จะเห็นว่าลักษณะเด่นของส่วนนี้จะเน้นตรงส่วนที่ใช้ภาษาระดับสูง อีดีแอลเบสิกเพื่อแก้ไขปัญหาการเขียนโปรแกรมภาษาแอสเซมบลีซึ่งมีความซับซ้อนและมีขนาดใหญ่ ในการเลือกชิป AT89C52 นี้ เพราะ AT89C52 เป็นชิปที่อยู่ในตระกูล MCS-51 ไมโครคอนโทรลเลอร์ หลาย ๆ ผู้ผลิตได้นำเอารูปแบบของไมโครคอนโทรลเลอร์ตระกูลนี้ไปทำการสร้างซีพียูเป็นของตนเอง ดังนั้น MCS-51 ไมโครคอนโทรลเลอร์จึงมีใช้กันอย่างกว้างขวางและยังมีใช้งานต่อไปอีกนาน

2) แอลซีดีเทอร์มินัล

ส่วนแอลซีดีเทอร์มินัลนี้ จะเป็นส่วนที่นำมาแทนไมโครคอมพิวเตอร์ ลักษณะการทำงานคล้ายกับไมโครคอมพิวเตอร์คือ รับส่งข่อมูลผ่านทางพอร์ตอนุกรม RS-232 แอลซีดีเทอร์มินัลจะมีส่วนที่เชื่อมต่อกับแผงแป้นอักขระของไมโครคอมพิวเตอร์ (PC Keyboard) สำหรับป้อนโปรแกรม ทั้งนี้เพราะแผงแป้นอักขระของไมโครคอมพิวเตอร์นี้หาซื้อได้ง่ายและมีราคาถูก สามารถนำไปประยุกต์ใช้งานด้านอื่น ๆ ได้อีกมากมาย แอลซีดีเทอร์มินัลยังเชื่อมต่อกับส่วนแสดงผล บนจอแอลซีดีขนาด 40 ตัวอักษร 4 บรรทัด สำหรับการแสดงผลโปรแกรม จอแอลซีดีสามารถแสดงผลได้หลากหลาย ใช้กระแสไฟฟ้าต่ำ มีขนาดเล็ก เหมาะสำหรับนำไปใช้ในการแสดงผล

3) แผงฝึกทดลอง

แผงฝึกทดลองที่ผู้วิจัยได้คิดทำขึ้นนี้ จะใช้สำหรับทำการเรียนรู้วิธีในการควบคุมทางอิเล็กทรอนิกส์ต่าง ๆ ผ่านทางพอร์ตที่เป็นรูปแบบมาตรฐานของบริษัทอีทีที จำกัด ผู้เรียนรู้สามารถนำแผงฝึกทดลองอื่น ๆ นอกจากแผงฝึกทดลองของบริษัทอีทีที จำกัด มาต่อทดลองเพิ่มเติมได้ แผงฝึกทดลองนี้จะประกอบด้วยการทดลองย่อย ๆ พร้อมใบงานทั้งหมด 7 ใบงานคือ

- การทดลองพื้นฐานอินพุต
- การทดลองใช้งาน 8255

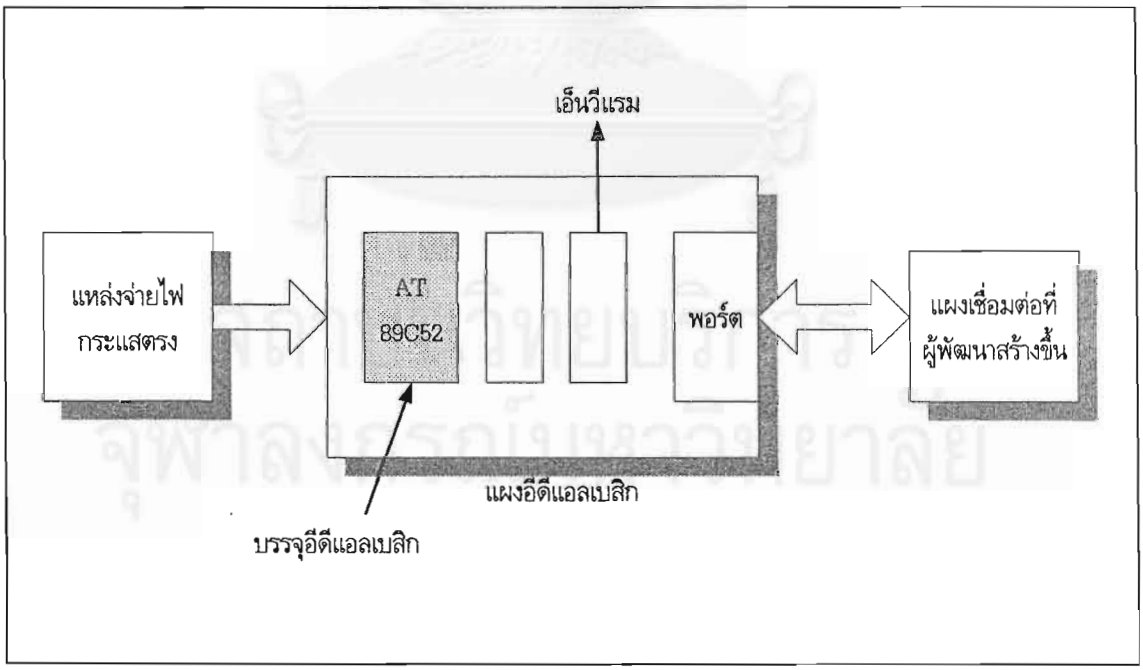
- การทดลองส่วนแสดงผลโดยใช้แอลอีดี 8 แถว และ 8 หลัก
- การทดลองมอเตอร์กระแสตรงและสเต็ปปิงมอเตอร์
- การทดลองอินพุตและแผงแป้นอักขระ
- การทดลองแสดงผลบนจอแอลซีดี
- การทดลองการแปลงผันสัญญาณ D/A และ A/D

4) ชุดแหล่งจ่ายไฟกระแสตรง (DC Power supply)

ชุดแหล่งจ่ายไฟกระแสตรงจะรับแรงไฟจากหม้อแปลงขนาดเล็ก 12 โวลต์ เปลี่ยนเป็นแรงดัน 5 โวลต์ แล้วจ่ายให้กับแอลซีดีเทอร์มินัลและแผงอีดีแอลเทอร์มินัล โดยมีตัวรักษาระดับแรงไฟ (Regulator) ช่วยในการรักษาระดับแรงไฟ

3.2 แผงควบคุมขั้นสุดท้ายสำหรับใช้งานจริง

จากที่ได้กล่าวมาจากการอ้างอิงรูปที่ 3.1 หลังจากการเรียนรู้บทเรียนและสามารถเขียนโปรแกรมพัฒนาด้วยภาษาเบสิกที่ผู้วิจัยได้ขอใช้เรียกภาษาเบสิกในวิทยานิพนธ์นี้ว่า อีดีแอลเบสิก เป็นผลสำเร็จแล้ว ผู้ใช้จะไม่จำเป็นที่จะใช้งานแอลซีดีเทอร์มินัลอีกต่อไป ทั้งนี้เป็นเพราะว่าระบบสามารถเก็บโปรแกรมที่พัฒนาเสร็จสมบูรณ์ และสามารถทำงานได้โดยอัตโนมัติ โดยปราศจากแอลซีดีเทอร์มินัลและคอมพิวเตอร์ใด ๆ ดังรูปที่ 3.2

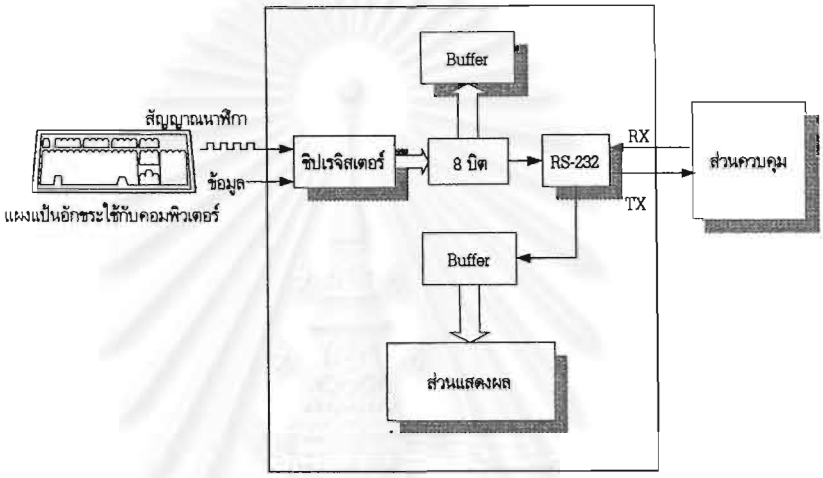


รูปที่ 3.2 แผงอีดีแอลเบสิกขั้นสุดท้ายสำหรับใช้งานจริง

บทที่ 4

แอลซีดีเทอร์มินัล

4.1 หลักการแอลซีดีเทอร์มินัล



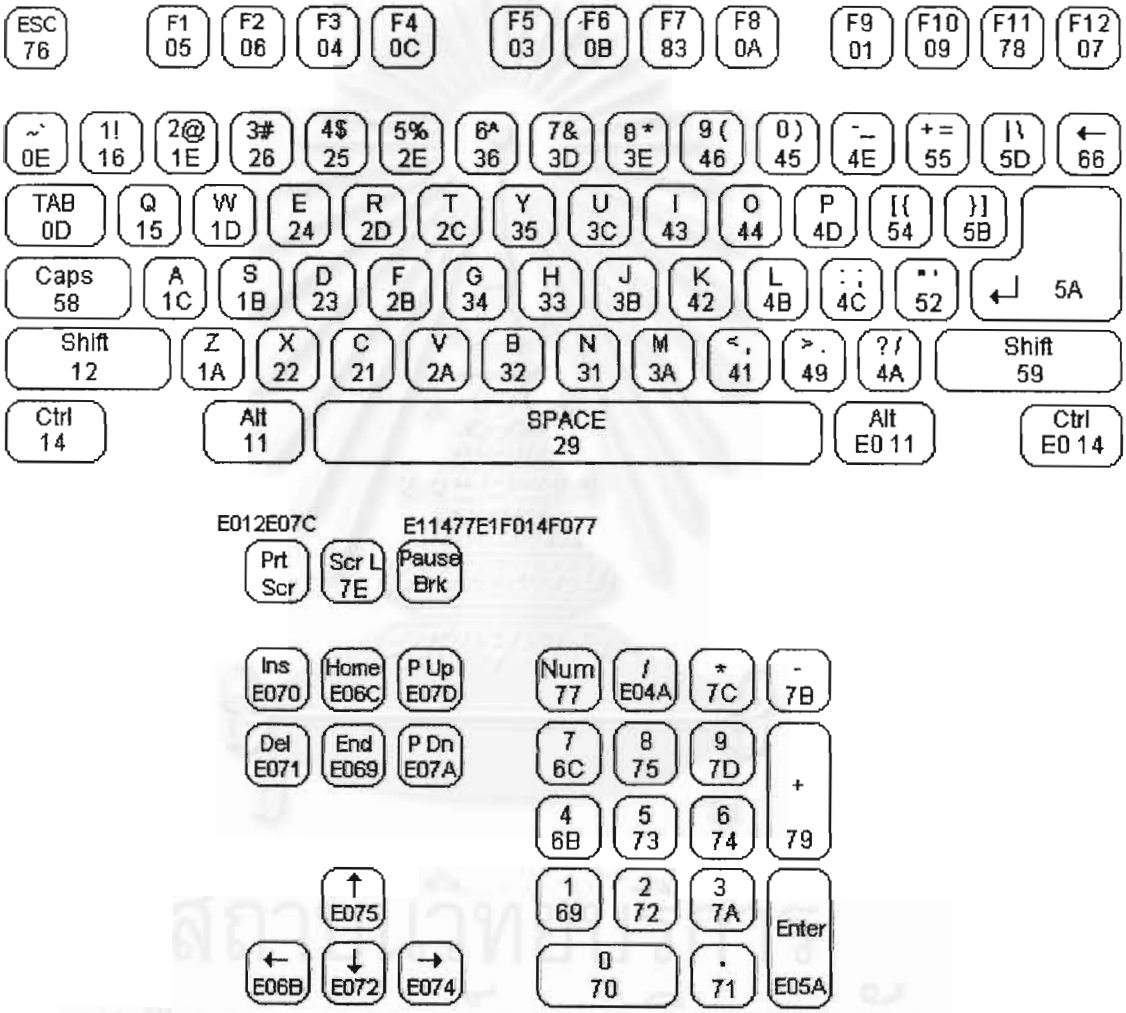
รูปที่ 4.1 หลักการของแอลซีดีเทอร์มินัล

เมื่อมีการกดแป้นกดบนแผงแป้นอักขระ แผงแป้นอักขระจะส่งรหัสกราดตรวจ (Scan code) และสัญญาณนาฬิกาออกมา เพื่อทำการซิงค์ให้กับจังหวะในการรับข้อมูลรหัสกราดตรวจ 8 บิต เทอร์มินัลพอร์ตจะรับข้อมูลและเลื่อนข้อมูลออก เก็บค่าข้อมูลลงในหน่วยความจำจนครบ 8 บิต แล้วเก็บรูป 8 บิตไว้ในบัฟเฟอร์ (Buffer) จากนั้นส่งออกทางพอร์ตที่ต่อกับส่วนของพอร์ตอนุกรม (RS 232) ไปยังส่วนควบคุม ส่วนควบคุมจะส่งสัญญาณสะท้อน (Echo Signal) มาที่พอร์ตอนุกรม (Serial Port) พอร์ตเทอร์มินัลจะรับข้อมูลแล้วส่งออกทางส่วนแสดงผล ดังรูปที่ 4.1

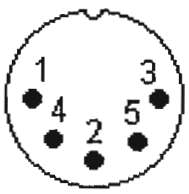
แอลซีดีเทอร์มินัลส่วนรับส่งข้อมูลโดยอุปกรณ์ที่ต่อเข้าทางแอลซีดีเทอร์มินัลเป็นอินพุต คือ แผงแป้น-อักขระแบบเอทีซึ่งเป็นที่นิยมในประเทศไทย โดยที่แผงแป้นอักขระมักมีรหัส (Code) เฉพาะของแต่ละแป้น (key) เมื่อเรากดแผงแป้นอักขระรหัสจะถูกส่งไปยังพอร์ตอนุกรม (Serial Port) เพื่อบอกไบออส (Bios) ของแผงแป้นอักขระในพีซีให้ตรวจสอบว่าแป้น (Key) ใดกำลังถูกกด นอกจากตัวอักษรบนแผงแป้นอักขระแล้วก็ยังมีคำสั่งบนแผงแป้นอักขระอีก ซึ่งปุ่มคำสั่งก็จะมีรหัสเฉพาะเพื่อบอกว่าจะให้พีซี (PC) ทำคำสั่งอะไรอีก เช่น

- รหัส EDH จะกำหนดตำแหน่งของแอลอีดี (LED) ที่จะติดหรือดับ เช่น Num Lock, Caps Lock และ Scroll Lock หลังจากส่ง EDH ไปแผงแป้นอักขระจะรอรหัสจำนวนเลขฐาน 16 ตอบกลับมาแล้วจะส่งบิตไปกำหนดสถานะแอลอีดี (LED)

- รหัส F0H เป็นรหัสที่กำหนดรหัสกราดตรวจ (Scan code) แผงแป้นอักขระจะส่ง F0H จากนั้นแผงแป้นอักขระจะรอรหัส FAH ตอบกลับมา และรอรหัส 00H ตอบกลับมาเพื่อกำหนดการย้อนกลับเพื่อลบตัวอักษร
- รหัส F5H จะกำหนดการปิดทาง (Disable) และการตั้งค่าใหม่จากแผงแป้นอักขระ
- รหัส FEH จะส่งคำสั่งในคำสั่งที่แล้วซ้ำอีก



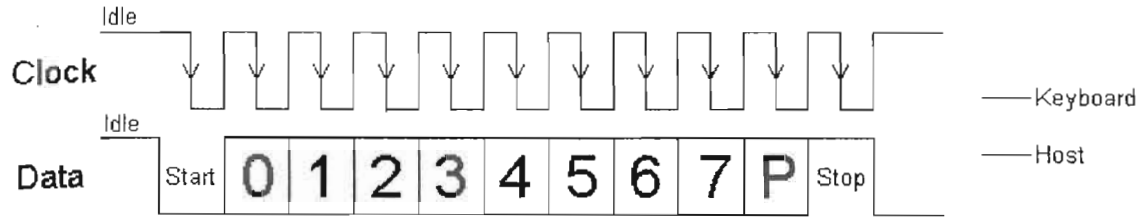
รูปที่ 4.2 รหัสของแต่ละปุ่มบนแผงแป้นอักขระ [13]



1. KBD Clock
2. KBD Data
3. N/C
4. GND
5. +5V

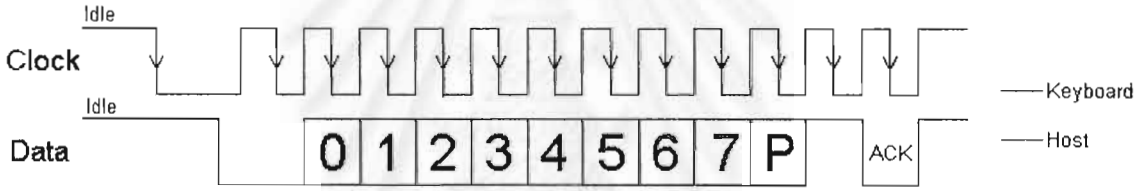
รูปที่ 4.3 แผงแป้นอักขระบนตัวเชื่อมต่อ [13]

โพรโทคอลของแผงแป้นอักขระของไมโครคอมพิวเตอร์



รูปที่ 4.4 รูปคลื่นสัญญาณของแผงแป้นอักขระไปยังคอมพิวเตอร์แม่งาน (host) [12]

1) จากแผงแป้นอักขระไปยังคอมพิวเตอร์แม่งาน จะส่งในลักษณะเป็นแบบประสานเวลา (synchronous) คือมีสัญญาณนาฬิกาออกมากับข้อมูลจากรูปที่ 4.4 จะเห็นได้ว่า รูปแบบข้อมูลมีทั้งหมด 11 บิต คือ 1 บิตเริ่มต้น และ 8 บิตข้อมูลตามด้วยภาวะคู่หรือคี่ (parity) และบิตสิ้นสุด

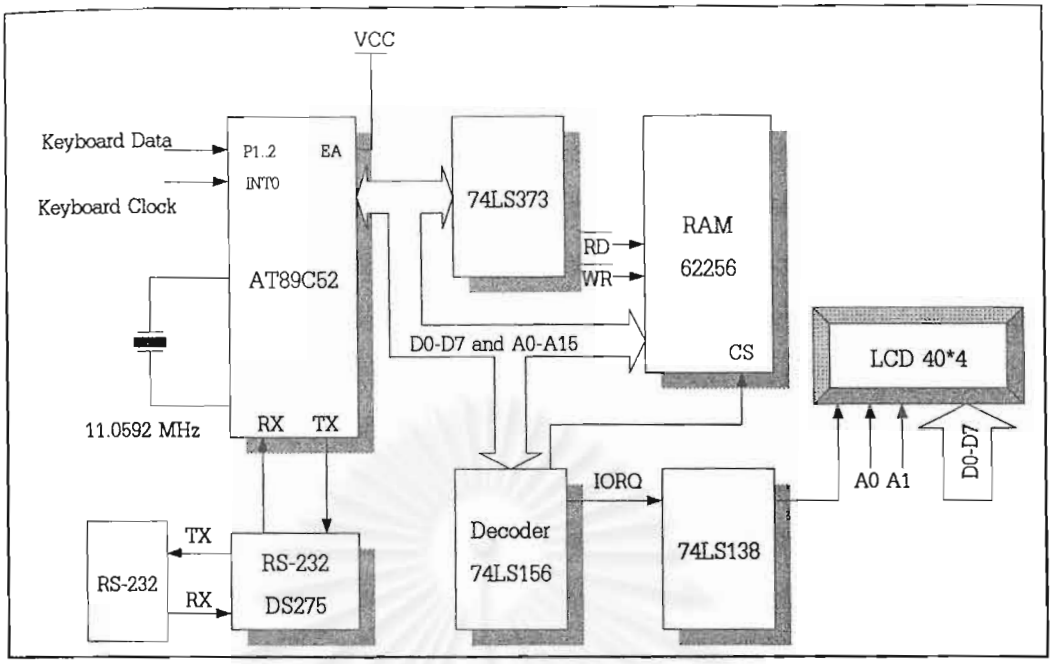


รูปที่ 4.5 รูปคลื่นสัญญาณของคอมพิวเตอร์แม่งานไปยังแผงแป้นอักขระ [12]

ข้อมูลจากคอมพิวเตอร์แม่งานไปยังแผงแป้นอักขระจะมีทั้งหมด 12 บิต ประกอบด้วย บิตเริ่มต้น 1 บิต บิตข้อมูล 8 บิตข้อมูลและบิตภาวะคู่หรือคี่ต่อด้วยบิตหยุดแล้วสุดท้าย บิตตอบรับ (ACK bit) ดังรูปที่ 4.5 (ข้างบน)

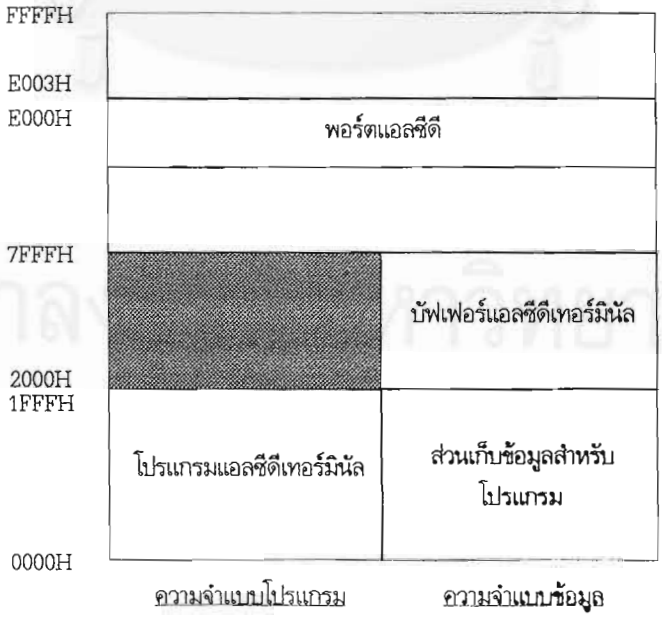
4.2 ส่วนประกอบของแอลซีดีเทอร์มินัล

แอลซีดีเทอร์มินัล เป็นส่วนที่ใช้รับจากแผงแป้นอักขระของไมโครคอมพิวเตอร์ทั่วไปซึ่งส่งข้อมูลและสัญญาณนาฬิกา เข้ามาทางอินพุตทีละบิต เข้ามาเก็บในหน่วยความจำแล้วเลื่อนหน่วยความจำไปทางขวา รับเข้ามาทีละ 1 บิต แล้วเลื่อนทุก ๆ ครั้ง ที่ทำการรับข้อมูล 1 ครั้ง จนครบ 8 บิต จากนั้นอ่านข้อมูลเข้ามาเพื่อทำการแปลงจากรหัสกราดตรวจแผงแป้นอักขระเป็นรหัสแอสกี ให้ตรงตามแผงแป้นอักขระที่ถูกกด จากนั้นส่งออกทางพอร์ต RS-232 แบบอะซิงโครนัส (Asynchronous) แบบ 8 บิต 1 บิตเริ่มต้น (Start bit) และ 1 บิตสิ้นสุด (Stop bit) ไม่มีบิตภาวะคู่หรือคี่ ในขณะเดียวกันก็รอรับข้อมูลทาง RS-232 พอร์ต เพื่อไปทำการแสดงผลยังแอลซีดี (LCD) ขนาด 40 ตัวอักษร 4 บรรทัด ดังรูปที่ 4.6



รูปที่ 4.6 ส่วนประกอบของแอลซีดีเทอร์มินัล

แอลซีดีเทอร์มินัล เป็นส่วนที่ใช้รับจากแผงแป้นอักขระของไมโครคอมพิวเตอร์ทั่วไปซึ่งส่งข้อมูลและสัญญาณนาฬิกา เข้ามาทางอินพุตทีละบิต เข้ามาเก็บในหน่วยความจำแล้วเลื่อนหน่วยความจำไปทางขวา รับเข้ามาทีละ 1 บิต แล้วเลื่อนทุก ๆ ครั้ง ที่ทำการรับข้อมูล 1 ครั้ง จนครบ 8 บิต จากนั้นอ่านข้อมูลเข้ามาเพื่อทำการแปลงจากรหัสกราดตรวจแผงแป้นอักขระเป็นรหัสแอสกี ให้ตรงตามแผงแป้นอักขระที่ถูกกด จากนั้นส่งออกทางพอร์ต RS-232 แบบอะซิงโครนัส (Asynchronous) แบบ 8 บิต 1 บิตเริ่มต้น (Start bit) และ 1 บิตสิ้นสุด (Stop bit) ไม่มีบิตภาวะคู่หรือคี่ ในขณะเดียวกันก็รอรับข้อมูลทาง RS-232 พอร์ต เพื่อไปทำการแสดงผลยังแอลซีดี (LCD) ขนาด 40 ตัวอักษร 4 บรรทัด ดังรูปที่ 4.6 (ข้างบน)



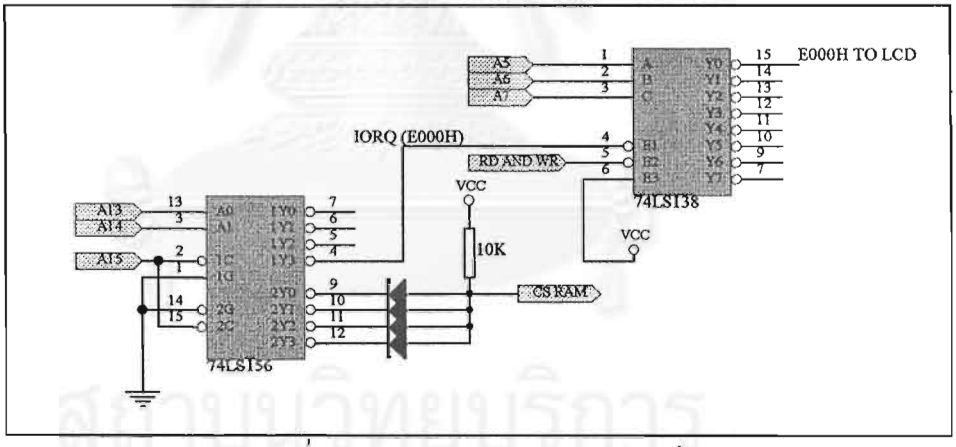
รูปที่ 4.7 แผนผังแสดงตำแหน่งในหน่วยความจำของแอลซีดีเทอร์มินัล

ในรูปที่ 4.6 ใช้ขา P1.2 ของพอร์ต 1 เป็นขาสำหรับบิตข้อมูล และขา INT0 (Interrupt 0) เป็นขารับสัญญาณนาฬิกาจากแผงแป้นอักขระ โดยจะเริ่มทำงานเมื่อเปลี่ยนจากระดับตรรกะ 1 เป็นตรรกะ 0 ซีพียูจะถูกขัดจังหวะให้กระโดดไปทำงานยังโปรแกรมรับข้อมูลจากแผงแป้นอักขระ และนำข้อมูลที่ถูกแปลงเป็นรหัสแอสกีส่งออกทาง RS-232 ในแนวเดียวกัน ซีพียูจะรอรับการขัดจังหวะทางพอร์ตอนุกรมอีกครั้ง (Serial port Interrupt) แต่โปรแกรมจะกำหนดให้ INT0 มีความสำคัญสูงกว่าการขัดจังหวะทางพอร์ตอนุกรม สำหรับแผนผังแสดงตำแหน่งในหน่วยความจำของพอร์ตเทอร์มินัล จะแสดงได้ดังรูปที่ 4.7 (ข้างบน)

บัฟเฟอร์ของโปรแกรมในพอร์ตเทอร์มินัลจะเริ่มที่ตำแหน่ง 2000H-7FFFH เนื้อที่ความจุเท่ากับ 5FFFH หรือ 24575 ตัวอักษร แต่แอลซีดี 1 บรรทัด มี 40 ตัวอักษร ดังนั้นจะใช้เนื้อที่ 40 ไบต์ต่อ 1 บรรทัด ดังนั้นบัฟเฟอร์ของเทอร์มินัลพอร์ตสามารถเก็บข้อมูลได้สูงสุดเท่ากับ 24575/40 = 614 บรรทัด เมื่อเขียนถึงบรรทัดที่ 615 จะกลับไปเริ่มต้นที่บรรทัดที่ 1 ใหม่ ลักษณะการเขียนโปรแกรมข้อมูลแบบวงแหวน (Ring Buffer)

ส่วนถอดรหัสของแอลซีดีเทอร์มินัล

ตัวถอดรหัส (Decoder) ใช้ 74LS156 ถอดรหัสที่ตำแหน่ง 0000H ถึง 7FFFH เพื่อกำหนดการทำงานให้แรม 32 กิโลไบต์ และสร้างสัญญาณ IORQ (I/O Request) เพื่อส่งไปควบคุมพอร์ตไอ/โอให้กับแอลซีดีที่ตำแหน่ง E000H ถึง E003H ดังรูปที่ 4.8



รูปที่ 4.8 วงจรถอดรหัสของแอลซีดีเทอร์มินัล

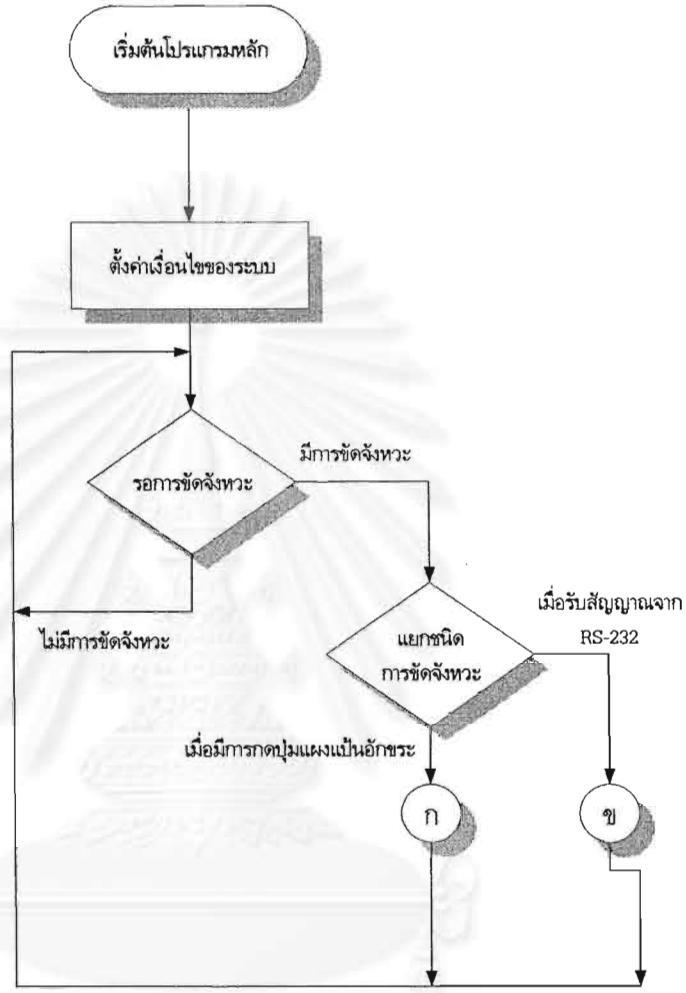
โดยตำแหน่งของพอร์ตแอลซีดีจะแสดงในตารางที่ 4.1

ตารางที่ 4.1 ตำแหน่งของแอลซีดี[18]

ชนิดแอลซีดี (LCD)	หน้าที่การทำงานของแอลซีดี (LCD)	หมายเลขพอร์ต
ด็อตเมตริกซ์	เขียนคำสั่งให้กับด็อตเมตริกซ์ LCD	E000H
	อ่านตำแหน่ง และ Busy Flag	E001H
	เขียนข้อมูลให้ CG&DD RAM	E002H
	อ่านข้อมูลจาก CG&DD RAM ที่ Cursor ชี้อยู่	E003H

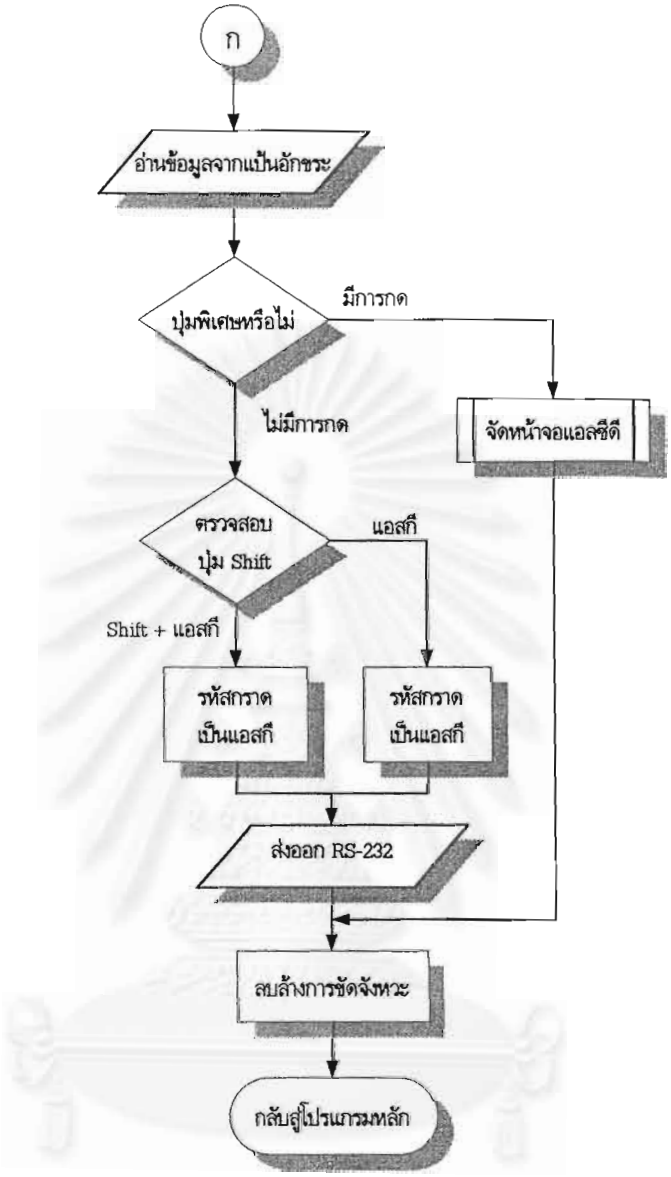
4.3 ส่วนซอฟต์แวร์ของแอลซีดีเทอร์มินัล

การทำงานอย่างคร่าว ๆ ของซอฟต์แวร์ของแอลซีดีเทอร์มินัล สามารถเขียนเป็นแผนภาพการทำงานได้ดังนี้



รูปที่ 4.9 แผนภาพการทำงานของโปรแกรมหลัก

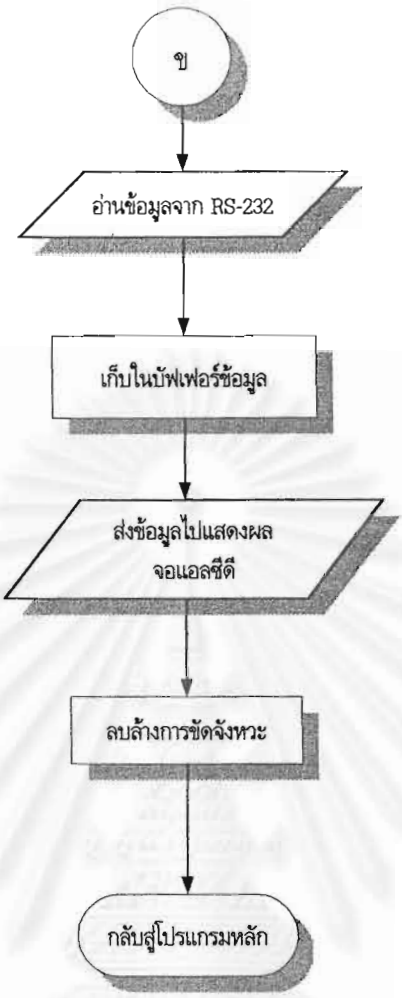
การทำงานของโปรแกรมหลัก เริ่มต้นระบบจะตั้งค่าเงื่อนไขการทำงานของโปรแกรม จากนั้นจะเข้าสู่สถานะรอการขัดจังหวะ (Interrupt) ถ้าไม่มีการขัดจังหวะก็จะวนรอไปเรื่อย ๆ แต่ถ้ามีการขัดจังหวะเกิดขึ้นจะมีการตรวจสอบว่าเป็นการขัดจังหวะชนิดใด ในตัวโปรแกรมจะมีการขัดจังหวะ 2 อย่างคือ การขัดจังหวะเมื่อมีการกดแป้นอักขระและขัดจังหวะเมื่อรับรหัสแอสกีเข้ามาทางพอร์ตอนุกรม RS-232 โปรแกรมจะให้ความสำคัญการขัดจังหวะทางจากแป้นอักขระสูงกว่าซอฟต์แวร์จะตรวจสอบชนิดการขัดจังหวะ แล้วทำการกระโดดไปทำโปรแกรมตามชนิดของการขัดจังหวะต่อ



รูปที่ 4.10 แผนภาพการทำงานโปรแกรมย่อยของแผงแป้นอักขระ

การทำงานโปรแกรมย่อยของแผงแป้นอักขระ เมื่อมีการกดปุ่มซอฟต์แวร์จะรอข้อมูลเข้ามาจนครบ 8 บิต จากนั้นจะอ่านเข้ามาเก็บไว้ในเรจิสเตอร์ ซอฟต์แวร์จะตรวจสอบค่าของข้อมูลที่รับเข้ามาว่ามีการกดปุ่มพิเศษหรือไม่ เช่น การกดเลื่อนซ้าย, ขวา, บน และล่าง เป็นต้น ถ้าตรวจสอบว่ามีการกดปุ่ม ก็จะดูว่าปุ่มอะไรถูกกดแล้วทำงานตามปุ่มที่ถูกกดเกี่ยวกับการจัดหน้าจอแอลซีดี

ถ้าซอฟต์แวร์ตรวจสอบว่าไม่ได้มีการกดปุ่มฟังก์ชันพิเศษ ก็จะตรวจสอบข้อมูลอีกครั้งว่าถูกกดปุ่ม Shift มาด้วยหรือไม่ถ้าใช่จะแปลงรหัสอักขระตามปุ่มที่กด ในตารางของปุ่ม Shift แต่ถ้าไม่ใช่จะแปลงค่าในตารางอักขระที่ไม่ได้กด Shift เมื่อแปลงรหัสกราดอักขระเป็นรหัสแอสกี ก็จะส่งออกทางพอร์ตอนุกรม (RS-232) จากนั้นจะลบล้างค่าที่บันทึกในสแต็กทั้งหมด แล้วกระโดดกลับสู่โปรแกรมหลัก



รูปที่ 4.11 ซอฟต์แวร์โปรแกรมย่อยของส่วนแสดงผล

การทำงานซอฟต์แวร์ ซอฟต์แวร์จะรับข้อมูลทางพอร์ตอนุกรม (RS-232) มาเก็บในบัฟเฟอร์ข้อมูลขนาด 24 KB แล้วส่งข้อมูลออกทางจอแอลซีดี เมื่อแสดงผลเสร็จจะลบล้างค่าต่าง ๆ แล้วกลับเข้าสู่โปรแกรมหลัก

ซอฟต์แวร์ของเทอร์มินัลพอร์ต สามารถแบ่งออกเป็น 4 ส่วนใหญ่ คือ

4.3.1 โมดูลสำหรับการเริ่มต้น

โมดูลนี้จะทำหน้าที่ตั้งค่าเริ่มต้นต่าง ๆ สำหรับส่วนตัวควบคุมการขัดจังหวะ, ตั้งค่าชั้นหน่วยความจำ (Stack), ล้างหน่วยความจำภายในและภายนอก, กำหนดส่วนการติดต่อพอร์ตแบบอนุกรม, กำหนดค่าเริ่มต้นของจอแอลซีดีและข้อความเริ่มต้น

```
Module Inlialization
        Set up      Stack
        Clear       RAM_Data
        Set up      LCD_Screen
        Set up      Serial_Port
        Set up      Key_board_System
        Print       messages
          If (INT0=0) Then
            Call Module Key_board
          End If
End Module
```

4.3.2 โมดูลส่วนรับข้อมูลจากแผงแป้นอักขระ

โมดูลนี้จะทำหน้าที่รอการขัดจังหวะจากสัญญาณนาฬิกาของแผงแป้นอักขระ ที่ขา INT0 ของไมโครคอนโทรลเลอร์ โดยสัญญาณนาฬิกา 1 ลูก จะขัดจังหวะ 1 ครั้ง และรับข้อมูลเข้ามา 1 ครั้ง เข้ามาเก็บในหน่วยความจำทีละบิต แล้วทำการเลื่อนข้อมูลไปทางขวาเมื่อรับบิตต่อไปจนครบ 8 บิต แล้วนำข้อมูลมาแปลงให้เป็นรหัสแอสกี แล้วส่งออกทาง RS-232

```
Module    Key_board
        Do While (INT0 = 1)
          Stay Here
        If (INT0 = 1) Then
          Read (Start_bit)
          Read (Key_board_data)
          Count (Bit_data) = Count (bit_data) + 1
          While Count (bit_data) = 8
            Read Stop_bit
            Call Translate_ASCII
            Send RS_232
        End Do
      End If
End Module
```

4.3.3 โมดูลส่วนรับข้อมูลจาก RS-232

โมดูลส่วนนี้จะทำหน้าที่รอการขัดจังหวะทางพอร์ตอนุกรม แล้วรับข้อมูลที่ส่งมาเก็บไว้ในบัฟเฟอร์ ขนาด 5FFFH ในขณะเดียวกันก็ส่งข้อมูลออกทางแอลซีดีพอร์ต

```
Module        Receive RS_232
        Do While (Rx = 0)
            Stay Here
        If (Rx = 1) Then
            Read (Serial_Buffer)
            Save Buffer
            Out_Put to LCD
        End If
End Module
```

4.3.4 โมดูลส่วนการแปลรหัสกราดตรวจ (Scan code)

โมดูลส่วนนี้จะทำหน้าที่รับข้อมูลรหัสสแกนคีย์บอร์ดนำมาแปลงเป็นรหัสแอสกี โดยข้อมูลจากแผงแป้น-อักขระที่เข้ามาจะมีฟังก์ชันที่พิเศษ เช่น ปุ่ม Enter, ปุ่ม Shift และ ปุ่มบน, ล่าง, ซ้ายและขวา ซึ่งถ้าเป็นฟังก์ชันที่พิเศษจะไม่ถูกนำมาแปลงเป็นแอสกีแล้วส่งไป RS-232 แต่จะนำรหัสไปกราดตรวจว่าเป็นรหัสใด เพื่อควบคุมหน้าจอแอลซีดีต่อไป

```
Module Translate
        Read Key_board_data
        Select Case   Key Press
            Case   Key Press = Enter
                   New_line_LCD
            Case   Key Press = Shift_ASCII
                   Load Shift_ASCII
            Case   Key Press = ASCII
                   Load ASCII
            Case   Key Press = Right
                   Move_Right_ LCD
            Case   Key Press = Left
                   Move_ Left_ LCD
            Case   Key Press = UP
                   Move_ UP_ LCD
```

```
        Case    Key Press = Down
                Move_Down _ LCD
End Module.
```

บทที่ 5

อีดีแอลเบสิก (EDL BASIC)

5.1 หลักการของอีดีแอลเบสิก (EDL BASIC)

ภาษาระดับสูง เบสิก (BASIC) [4] (ย่อมาจากคำว่า Beginner 's All purpose Symbolic Instruction Code) เป็นภาษาระดับสูงที่ใกล้เคียงกับภาษาอังกฤษที่ใช้ในชีวิตประจำวันมากที่สุด ซึ่งเหมาะสำหรับการแก้ปัญหาทางด้านวิทยาศาสตร์และคณิตศาสตร์

โปรแกรมสำหรับแปลงภาษาเบสิกมีอยู่ 2 ชนิด คือ BASIC Interpreter และ BASIC Complier มีข้อแตกต่างกันตรงที่ BASIC Interpreter เป็นโปรแกรมที่มีขนาดเล็ก สามารถติดตั้งลงบนชิปได้ ทำหน้าที่สำหรับรับคำสั่งรูปแบบภาษาเบสิกแล้วทำการแปลงเป็นภาษาเครื่อง (Machine Language) เพื่อส่งไปประมวลผลยังซีพียู (CPU) ส่วน BASIC Complier เป็นโปรแกรมแปลงคำสั่งภาษาเบสิกเป็นภาษาแอสเซมบลี ซึ่งมีขนาดใหญ่ ดังนั้นจึงติดตั้งไว้ในไมโครคอมพิวเตอร์

5.1.1 รูปแบบของภาษาเบสิก

[หมายเลขบรรทัด] [คำสั่ง] [ค่าคงที่และตัวแปลต่าง ๆ]

ตัวอย่าง
```
10 FOR I=1 TO 10
20 PRINT I
30 NEXT I
```

1) หมายเลขบรรทัด (Line Number) เป็นหมายเลขบรรทัดที่ใช้อ้างอิงคำสั่งควบคุมและยังใช้ระบุหมายเลขบรรทัดเพื่อแก้ไขต่อไป

2) คำสั่ง (Command) มี 2 ประเภท คือ คำสั่งของระบบเป็นคำสั่งที่คนสั่งให้เครื่องทำงานโดยตรงไม่ต้องมีหมายเลขบรรทัดอยู่ข้างหน้า และอีกแบบหนึ่งคือคำสั่งประเภทที่ต้องมีหมายเลขบรรทัดอยู่ข้างหน้า (Statement)

3) ค่าคงตัว (Constants) แบ่งเป็น 2 ประเภท คือ ค่าคงตัวที่เป็นตัวเลข (Numeric Constants) และค่าคงตัวที่ไม่ใช่จำนวนเลข (String Constants)

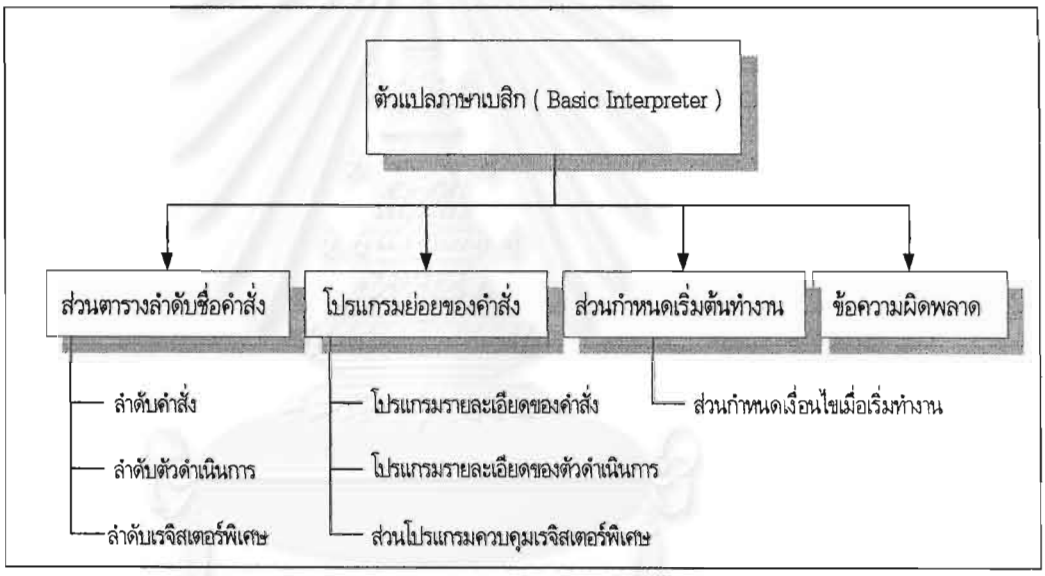
4) ตัวแปร (Variables) คือ ชื่อของหน่วยความจำในคอมพิวเตอร์ ซึ่งจะทำหน้าที่ในการเก็บค่าต่าง ๆ เพื่อใช้ในการทำงานซึ่งตัวแปรแบ่งออกเป็น 2 ประเภท คือ

4.1 ตัวแปรที่ใช้แทนค่าตัวเลข (Numeric Variables) คือตัวแปรที่ใช้แทนตัวเลขที่ใช้ในการคำนวณ การตั้งชื่อตัวแปรชนิดนี้จะขึ้นต้นด้วยตัวอักษร

4.2 ตัวแปรที่ใช้แทนสายอักขระ (String Variables) คือตัวแปรที่ค่าไม่ใช่ตัวเลขและไม่ สามารถนำไปใช้ในการคำนวณได้ การตั้งชื่อตัวแปรชนิดนี้จะมีเครื่องหมาย $ นำหน้าอยู่เสมอ

5.1.2 อีดีแอลเบสิก

อีดีแอลเบสิก เป็นตัวแปลภาษาเบสิก (BASIC Interpreter) ซึ่งดัดแปลงมาจาก MCS51 BASIC-52 ซึ่งเป็นผลงานของบริษัทอินเทลที่บรรจุอยู่ในไมโครคอนโทรลเลอร์ MCS-51 โดยมีขนาดโปรแกรม 8 กิโลไบต์ ได้มีการดัดแปลงโดยโปรแกรมต้นฉบับ คือ BASIC-52.HEX ที่เป็นแฟ้ม (File) นามสกุล . HEX มาแปลงกลับให้อยู่ในรูปแบบภาษาแอสเซมบลี แล้วนำโปรแกรมที่ได้จากการแปลงกลับมาวิเคราะห์หลักการทำงานของโปรแกรมตัวแปลภาษาเบสิก จากนั้นนำโครงสร้างโปรแกรมมาศึกษาเพื่อพัฒนาชุดคำสั่งใหม่ขึ้นมาสำหรับใช้ในงานควบคุม จากรูปที่ 5.1 จะแบ่งส่วนประกอบออกเป็น 4 ส่วนใหญ่ๆ แต่ส่วนที่ต้องมีการเขียนเพิ่มเพื่อเพิ่มชุดคำสั่งเข้าไปก็ คือ ส่วนตารางลำดับคำสั่ง ผู้วิจัยจะต้องทราบตัวเลขลำดับคำสั่งในแต่ละคำสั่งจากในโปรแกรมต้นฉบับ จากนั้นกำหนดตัวเลขลำดับคำสั่งใหม่ไม่ให้ซ้ำของเดิมและอีกส่วนหนึ่ง (ดูจากตารางที่ 5.1) คือ ส่วนโปรแกรมย่อยของคำสั่ง เป็นส่วนที่ผู้วิจัยได้ทำการเขียนเพิ่มเติมโดยจะทำงานสัมพันธ์กับชื่อคำสั่งที่ได้ทำเพิ่มขึ้น ซึ่งโครงสร้างทั้งหมดจะแสดงดังรูปที่ 5.1



รูปที่ 5.1 โครงสร้างของอีดีแอลเบสิกสำหรับไมโครคอนโทรลเลอร์

แต่ละส่วนสามารถอธิบายได้ดังนี้

1. ส่วนตารางลำดับ (Token) ชื่อคำสั่ง ในส่วนนี้จะเก็บคำสั่งโดยมีตัวเลขกำกับในแต่ละคำสั่ง ซึ่งตัวเลขกำกับคำสั่งนี้อาจเป็นตัวเลขที่บอกตำแหน่งที่ให้ชี้ตัวกระโดดไปยังโปรแกรมย่อยของคำสั่งที่รับจากแผงแป้นอักขระ ตัวอย่าง เช่น

<u>ตัวอย่าง</u> การเก็บข้อมูลของตารางลำดับคำสั่ง

```
DB    13H      ; ตัวเลขกำกับคำสั่ง
DB    Display  ; คำสั่ง
DB    00H      ; รหัสปิดท้ายคำสั่ง
```

2. โปรแกรมย่อยคำสั่ง เป็นส่วนที่เก็บโปรแกรมย่อยโดยใช้หมายเลขกำกับคำสั่งเป็นตัวชี้ (Pointer) ให้กระโดดไปยังโปรแกรมนั้น ๆ ตัวอย่าง เช่น

ตัวอย่าง ส่วนที่เก็บโปรแกรมย่อย

```
RUN _ DISPLAY :
                ; User ASM Code for Display Goes here
```

3. ส่วนที่เก็บข้อความใช้งานและข้อความผิดพลาด เป็นส่วนที่เก็บข้อความใช้งานและข้อความผิดพลาดของโปรแกรมดัง ตัวอย่าง เช่น

```
BAD : DB  BAD SYNTAX  ERROR
    ; DB  NO DATA
```

4. ส่วนกำหนดค่าเริ่มต้นเงื่อนไขของระบบ เมื่อโปรแกรมตัวแปลภาษาเบสิกเริ่มทำงานจะมีกำหนดเงื่อนไขเริ่มต้นของโปรแกรมโดยในส่วนนี้จะเก็บข้อมูลเริ่มต้นเมื่อใช้งาน เช่น

- กำหนดสัญญาณนาฬิกาของระบบ
- กำหนดอัตราการส่งข้อมูลทางพอร์ตอนุกรม
- กำหนดชั้นทำงาน (Stack) และ ลบล้างหน่วยความจำในแรมของระบบ

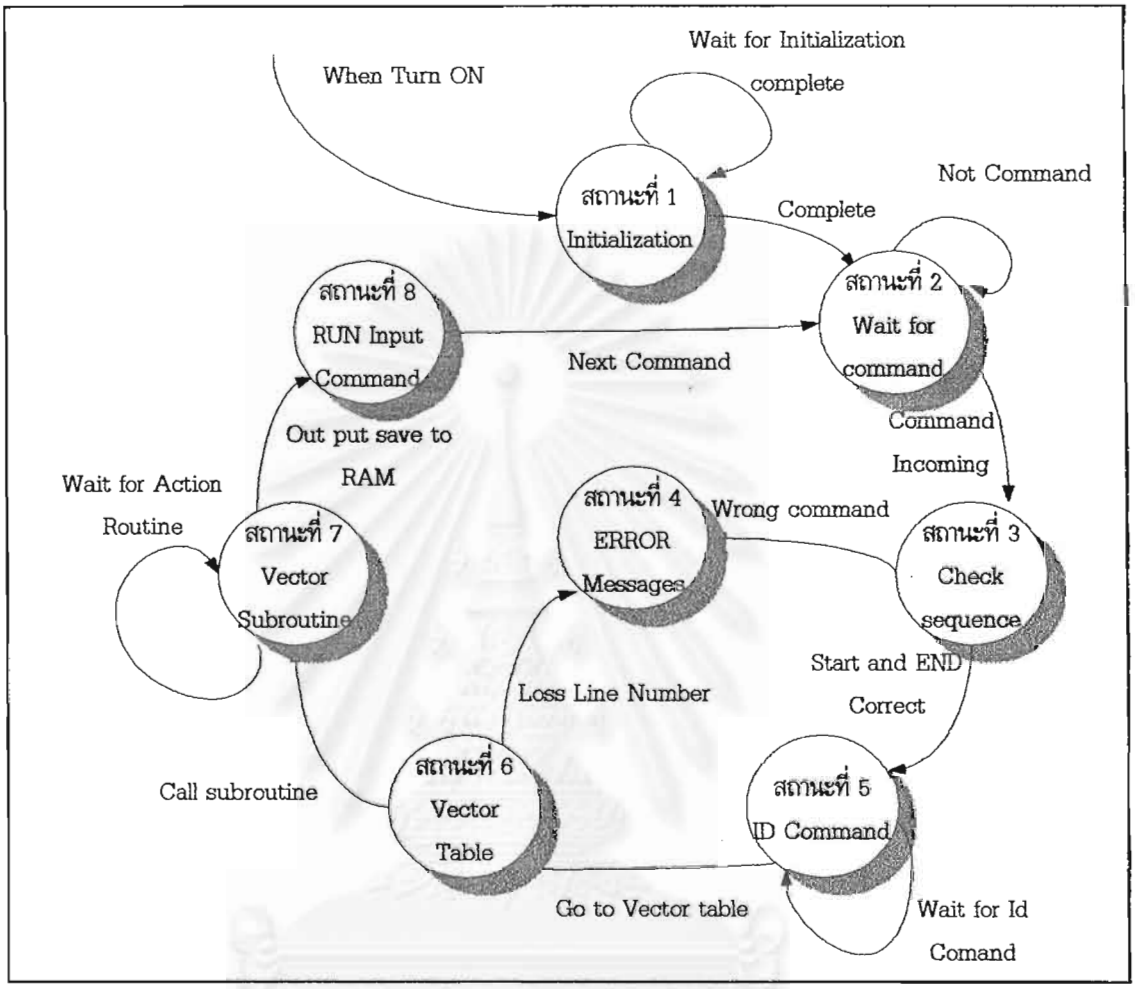
ตารางที่ 5.1 เป็นตารางของคำสั่งทั้งหมดของชุดคำสั่งภาษาเบสิกต้นฉบับและมีคำสั่งที่ได้เพิ่มเข้าไปเพื่อช่วยในการใช้งานควบคุม และในแต่ละคำสั่งก็จะแสดงตัวเลขลำดับคำสั่งกำกับไว้ข้างผู้ศึกษาสามารถนำเอาข้อมูลตัวเลขฐาน 16 จากคำสั่งเหล่านั้นมาแล้วกำหนดค่าใหม่ให้กับคำสั่งใหม่ได้โดยไม่ให้ซ้ำกับลำดับตัวเลขเดิม

ตารางที่ 5.1 ชุดคำสั่งอีดีแอลเบสิก

Command / Token number		Statement / Token number		Operation / Token number	
RUN	0F0H	CLOCK (1&0)	8EH	ADD (+)	0E3H
CONT	0F1H	CLEAR (S&I)	81H	DIVIDE (/)	0E4H
LIST	0F2H	DATA	9CH	EXPONENTIAL (**)	0E1H
NEW	0F3H	READ	9BH	SUBTRACT (-)	0E5H
*RAM	0F4H	RESTORE	95H	MULTIPLY (*)	0E2H
*ROM	0F5H	DIM	8BH	LOGICAL (.AND.)	0E7H
*LOAD	0F6H	DO - WHILE	94H,0A1H	LOGICAL (.OR.)	0E8H
*PROG	0F7H	DO - UNTIL	94H,0A2H	LOGICAL (.XOR.)	0E6H
*PROG2	0F8H	END	0A3H	LOGICAL (NOT)	0E9H
*FBLANK	0F9H	FOR-TO-STEP	0A0H,0A6H	ABS( )	0B0H
		NEXT	97H	INT( )	0B1H
		GOSUB	9FH	SGN( )	0B2H
		GOTO	083H	SQR( )	0B7H
		IF - THEN - ELSE	9EH,0A5H,0A8H	LOG( )	0BBH
		INPUT	9AH	EXP( )	0B9H
		ONEX1	92H	SIN( )	0B6H
		ONTIME	91H	COS( )	0B7H
		PRINT	89H	TAN( )	0B8H
		PH0.	85H	=, <, <=, >, >=, <>	0EAH, EEH,ECH,EFH,0EBHEDH
		PUSH	82H	ASC( )	0D1H
		POP	88H	CHR( )	0D3H
		REM	96H	CBY( )	0B8H
		RETI	9DH	DBY( )	0BCH
		*BRKP	90H	XBY( )	0BDH
				GET( )	0C0H
				IE	0C6H
				IP	0C6H
				*PORT1	0CFH
				*PCON	0D0H
				*RCAP2	0CEH
				*T2CON	0CBH
				*TCON	0CCH
				*TMOD	0CDH
				*TIME	0C5H
				*TIMER0	0C8H

หมายเหตุ (* หน้าคำสั่ง คือ ชุดคำสั่งที่พัฒนาขึ้นมาใหม่เพิ่มขึ้นจากเดิม)

5.1.3 การทำงานซอฟต์แวร์ของอีดีแอลเบสิก



รูปที่ 5.2 แผนภาพสถานะการทำงานของซอฟต์แวร์ [ftp:intel.com, basic52.src]

จากรูปที่ 5.2 ในสถานะแรกโปรแกรมภาษาเบสิกจะกำหนดเงื่อนไขการทำงานของตนเอง เช่น ตั้งค่าสัญญาณนาฬิกา, ตั้งค่าสแต็กข้อมูล, ตั้งค่าเริ่มต้นให้เรจิสเตอร์ต่าง ๆ และอัตราเร็วของการส่งข้อมูลผ่านไปยังพอร์ตอนุกรม สถานะแรกจะรอจนกว่าการตั้งค่าเงื่อนไขเสร็จ เมื่อเสร็จเรียบร้อยแล้วก็จะเปลี่ยนไปเป็นสถานะที่ 2 คือรอคำสั่งเข้ามาทางแผงแป้นอักขระ เมื่อมีคำสั่งเข้ามาทางอินพุตก็จะเปลี่ยนสถานะเป็นสถานะที่ 3 สถานะนี้จะตรวจสอบคำสั่งเข้ามาว่าถูกต้องหรือไม่ ถ้าพบว่าไม่ถูกต้องก็จะเปลี่ยนไปสถานะที่ 4 คือส่งข้อความผิดพลาด (Error message) ออกหน้าจอ แต่ถ้าคำสั่งที่พิมพ์เข้ามาถูกต้อง ก็จะหาเลขกำกับคำสั่งนั้นว่าเป็นคำสั่งที่เท่าไรในสถานะที่ 5 เมื่อหาตัวเลขลำดับคำสั่งได้ก็จะเปลี่ยนเป็นสถานะที่ 6 ซึ่งจะนำตัวเลขลำดับคำสั่งไปเป็นตัวชี้ที่อยู่ของโปรแกรมย่อย สถานะที่ 6 จะตรวจสอบว่ามีการพิมพ์ หมายเลขบรรทัด และตัวแปรอื่น ๆ ด้วยหรือไม่ ถ้าไม่มีก็จะเปลี่ยนเป็นสถานะที่ 4 ถ้ามีก็กระโดดไปทำโปรแกรมย่อยของคำสั่งที่เราพิมพ์เข้าไปคือสถานะที่ 7 สถานะที่ 7 นี้จะรอจนกว่าการกระทำคำสั่งเสร็จแล้วเก็บค่าผลลัพธ์ลงในแรมคือสถานะที่ 8 เมื่อบันทึกลงในแรมเรียบร้อยแล้ว จะวนกลับไปรับคำสั่งต่อไปที่สถานะที่ 2 อีกและจะกระทำต่อเนื่องตามขั้นตอนจนเสร็จเรียบร้อย

5.1.4 รูปแบบการเก็บข้อมูลของโปรแกรมอีดีแอลเบสิก

เมื่อเราพิมพ์คำสั่งเข้าไปในหน่วยความจำของโปรแกรมภาษาเบสิก เก็บในรูปแบบของแอสกี (ASCII) การเก็บข้อมูลจะเริ่มเก็บทีละบรรทัด โดยจะยกตัวอย่างจากคำสั่ง ดังนี้

10 FOR I =1 TO 10 : PRINT I : NEXT I

แต่การเก็บข้อมูลในแรม (RAM) จะเริ่มต้นที่ตำแหน่ง 512 H สามารถเขียนรายละเอียดได้ดังนี้

ตารางที่ 5.2 การเก็บข้อมูลในหน่วยความจำของอีดีแอลเบสิก

ตำแหน่ง	ข้อมูล (ไบต์)	รายละเอียด
512	11H	ค่าความยาวของคำสั่งในบรรทัด = 17
513	00H	ไบต์ด้านสูงของตัวเลขบรรทัด
514	0AH	ไบต์ด้านต่ำของตัวเลขบรรทัด
515	0A0H	ตัวเลขลำดับของ " FOR "
516	49H	รหัสแอสกีของ " I "
517	0EAH	ตัวเลขลำดับของ " = "
518	31H	รหัสแอสกีของตัวเลข " 1 "
519	0A6H	ตัวเลขลำดับของ " TO "
520	31H	รหัสแอสกีของตัวเลข " 1 "
521	30H	รหัสแอสกีของตัวเลข " 0 "
522	3AH	รหัสแอสกีของ " : "
523	89H	ตัวเลขลำดับของ " PRINT "
524	49H	รหัสแอสกีของ " I "
525	3AH	รหัสแอสกีของ " : "
526	97H	ตัวเลขลำดับของ " NEXT "
527	49H	รหัสแอสกีของ " I
528	0DH	รหัสสิ้นสุดของโปรแกรม

ตารางที่ 5.2 เกิดจากการวิเคราะห์โปรแกรมตัวแปลเบสิกว่า เก็บข้อมูลโปรแกรมในรูปแบบใด จากนั้นนำมาเป็นแบบในการพัฒนาคำสั่งเบสิกขึ้นใหม่ที่มีคำสั่งมากขึ้น ข้อมูลที่เก็บในตารางจะอยู่ในหน่วยความจำแบบแรมที่ตำแหน่งเริ่มต้น 512H โดยการเก็บข้อมูลจะเก็บที่ละบรรทัด 1 บรรทัดต่อ 1 เลขลำดับคำสั่ง ในการดำเนินการโปรแกรมของอีดีแอลเบสิกจะแปลและดำเนินการที่ละเลขบรรทัด ในแต่ละเลขบรรทัดคำสั่งจะประกอบด้วยข้อมูลที่เก็บความยาวของบรรทัดนั้นๆขนาด 1 ไบต์ และตามด้วยข้อมูลเลขบรรทัด 2 ไบต์ โดยจะสามารถกำหนดเลขบรรทัดได้ตั้งแต่ 0000 ถึง 65535 (0000H – FFFFH) ตัวเลขบรรทัดนี้จะบอกให้อีดีแอลเบสิกทราบถึงการอ้างอิงข้อมูลจากเลขบรรทัด ลำดับต่อไปเป็นข้อมูลขนาด 1 ไบต์จะเป็นตัวเลขลำดับคำสั่ง (Token) สำหรับอ้างอิงถึงคำสั่งปฏิบัติการ

ให้กระโดดไปยังโปรแกรมย่อยที่รองรับคำสั่งปฏิบัติการได้อย่างถูกต้องในการกระทำโปรแกรม (สามารถดูข้อมูลลำดับคำสั่งได้ในตารางที่ 5.1) คำสั่งในแต่ละคำสั่งจะต้องมีข้อมูลตัวแปรและค่าคงที่เพื่อกำหนดสถานะการทำงานของโปรแกรมให้ถูกต้อง ดังนั้นแต่ละคำสั่งจะต้องมีข้อมูลตัวแปรและค่าคงที่อยู่ด้วย ในส่วนของค่าคงที่และตัวแปรจะเก็บข้อมูลในรูปเลขฐาน 16 โดยอ้างอิงมาจากรหัสแอสกี เมื่อจบ 1 บรรทัดจะตามข้อมูล 0DH เพื่อบอกให้อีดีแอลเบสิกทราบว่าได้สิ้นสุดโปรแกรมที่นี่แล้ว ตัวชี้โปรแกรมก็จะเก็บข้อมูลตำแหน่งในหน่วยความจำแรมสำหรับบันทึกค่าข้อมูลในตัวเลขบรรทัดถัดไป

ในการพัฒนาคำสั่งเพิ่มนี้จึงควรจำเป็นต้องทราบถึงรูปแบบการเก็บข้อมูลในแอลดีแอลเบสิกเพื่อที่จะทำความเข้าใจและดัดแปลงแก้ไขโปรแกรมตัวแปลภาษาเบสิกได้ดียิ่งขึ้น

5.1.5 คำสั่งภาษาเบสิกสำหรับการควบคุม

รูปแบบคำสั่งที่เพิ่มขึ้นจากโปรแกรมภาษาเบสิกทั่วไปมีดังนี้

- BAUD เป็นคำสั่งในการกำหนดอัตราส่งข้อมูลไปยังพอร์ตอนุกรม
- BRKP (Break point) เป็นคำสั่งที่กำหนดให้หยุดดำเนินโปรแกรมที่บรรทัดนั้น ๆ ที่มีคำสั่งนี้อยู่
- RAM เป็นคำสั่งที่เปลี่ยนแบบวิธีการใช้งานของหน่วยความจำเริ่มที่ตำแหน่ง 512 H
- ROM เป็นคำสั่งที่เปลี่ยนแบบวิธีการใช้งานของหน่วยความจำเริ่มที่ตำแหน่ง8000 H
- LOAD เป็นคำสั่งย้ายข้อมูลที่บันทึกอยู่ใน NV RAM ไปไว้ในแรมมอนิเตอร์
- CLEAR (I&S) เป็นคำสั่งเคลียร์และสแต็กข้อมูลในโปรแกรม
- CLOCK 0,1 เป็นคำสั่งที่อนุญาตและไม่อนุญาตให้ Timer/Counter0 ทำงาน
- ONEX เป็นคำสั่งที่อนุญาตให้ขัดจังหวะ 1 ทำงาน
- CONT (Continuous) เป็นคำสั่งสำหรับรันโปรแกรมต่อจากสถานะ Break point
- RETI (Return Interrupt) เป็นคำสั่งที่กำหนดการกระโดดสู่โปรแกรมหลักหลังการทำขัดจังหวะ

รูปแบบของตัวดำเนินการ (Operators) ที่เพิ่มขึ้นจากโปรแกรมภาษาเบสิกทั่วไปมีดังนี้

- CBY เป็นตัวดำเนินการของหน่วยความจำแบบรหัสคือตั้งแต่ 0 ถึง FFFFH
- DBY เป็นตัวดำเนินการของหน่วยความจำภายในแบบข้อมูลคือตั้งแต่ 0 ถึง FFH
- XBY เป็นตัวดำเนินการของหน่วยความจำภายนอกแบบข้อมูลคือตั้งแต่ 0 ถึง FFFFH

นอกจากนี้ยังมีกลุ่มตัวดำเนินการที่เกี่ยวข้องกับเรจิสเตอร์พิเศษดังต่อไปนี้

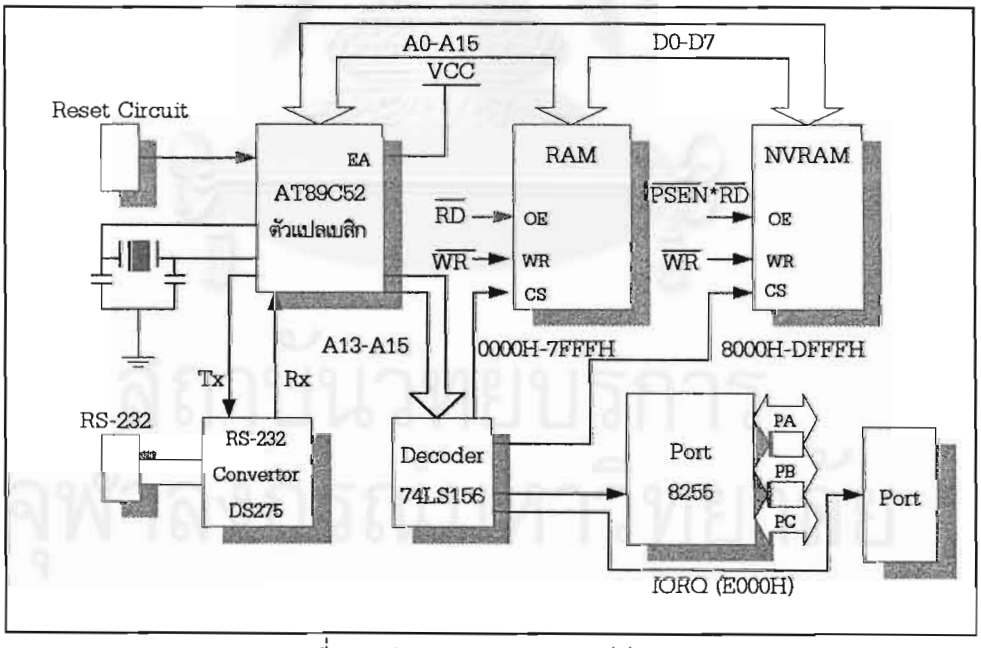
- PORT1 เป็นตัวดำเนินการเรจิสเตอร์ใน PORT1 ของ MCS-51
- PCON เป็นตัวดำเนินการเรจิสเตอร์ใน PCON ของ MCS-51
- RCAP2 เป็นตัวดำเนินการเรจิสเตอร์ใน RCAP2 ของ MCS-51

- T2CON เป็นตัวดำเนินการเรจิสเตอร์ใน T2CON ของ MCS-51
- TCON เป็นตัวดำเนินการเรจิสเตอร์ใน TCON ของ MCS-51
- TM0D ป็นตัวดำเนินการเรจิสเตอร์ใน TMOD ของ MCS-51
- TIMER0 เป็นตัวดำเนินการเรจิสเตอร์ใน TIMER0 ของ MCS-51
- TIMER1 เป็นตัวดำเนินการเรจิสเตอร์ใน TIMER1 ของ MCS-51
- TIMER2 เป็นตัวดำเนินการเรจิสเตอร์ใน TIMER2 ของ MCS-51
- XTAL เป็นตัวดำเนินการเกี่ยวกับค่าคริสทัลในระบบ
- FREE ตัวดำเนินการตรวจสอบเนื้อที่ในหน่วยความจำแรมที่คงเหลือ
- MTOP ตัวดำเนินการตรวจสอบเนื้อที่สูงสุดในหน่วยความจำแรม
- LEN ตัวดำเนินการหาความยาวของโปรแกรมในแรมหน่วยเป็นไบต์

5.2 ส่วนฮาร์ดแวร์ของอีดีแอลเบสิก

5.2.1 ส่วนประกอบของแผงอีดีแอลเบสิก

ส่วนประกอบของแผงอีดีแอลเบสิก เป็นส่วนที่ใช้ควบคุมโดยซีพียู (CPU) ซึ่งจะถูกบันทึกโปรแกรมภาษาเบสิกเข้าไปในหน่วยความจำแบบแฟลช (Flash) ภายใน 8 กิโลไบต์ สามารถอธิบายได้ดังรูปที่ 5.3



รูปที่ 5.3 ส่วนประกอบของแผงอีดีแอลเบสิก

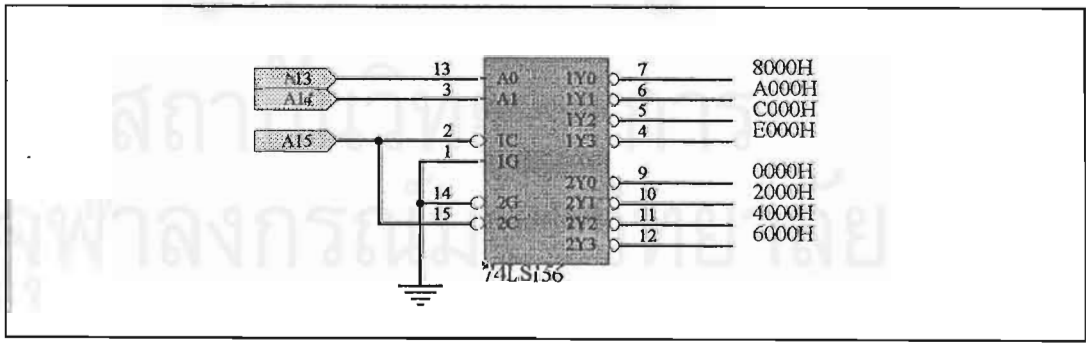
ในรูปที่ 5.3 ผู้วิจัยใช้ไมโครคอนโทรลเลอร์เบอร์ AT89C52 ของบริษัท ATMEL[10] ซึ่งมีหน่วยความจำของโปรแกรมขนาด 8 กิโลไบต์ บรรจุโปรแกรมตัวแปลภาษาเบสิกอยู่ภายในโดยต่อขา EA กับไฟเลี้ยงไมโครคอนโทรลเลอร์จะเลือกดำเนินโปรแกรมจากภายใน ภายในวงจรบรรจุแรมไว้ 2 ตัว คือ แรมมอนิเตอร์สำหรับแก้ไข

โปรแกรมกับแรมที่ทำหน้าที่เก็บข้อมูลพัฒนาโปรแกรมสำเร็จเราใช้เอ็นวีแรม (Nonvolatile RAM) จะถูกกำหนดการทำงานโดยวงจรถอดรหัส (Decoder) วงจรถอดรหัสจะควบคุมการใช้งานของ 8255 และกำหนดสัญญาณ (IORQ) ที่ตำแหน่ง 0E000H จะสามารถแสดงรายละเอียดแผนที่ของหน่วยความจำ (Memory Map) ได้ดังนี้

รูปที่ 5.4 แผนผังหน่วยความจำ (Memory Map) ของแผงอีดีแอลเบสิก

วงจรถอดรหัส (Decoder) ของแผงอีดีแอล

เราใช้ 74LS156 ทำการถอดรหัสพอร์ตอินพุตเอาต์พุต และหน่วยความจำต่าง ๆ แสดงดังรูปที่ 5.5



รูปที่ 5.5 วงจรถอดรหัส (Decoder) ในตำแหน่งต่าง ๆ

ตารางที่ 5.3 ความสัมพันธ์ระหว่างอินพุตและเอาต์พุตของตัวถอดรหัส [15]

อินพุต			เอาต์พุต	ตำแหน่งเอาต์พุต
A15	A14/B	A13/A		
0	0	0	2Y0	0000H ถึง 1FFFH
0	0	1	2Y1	2000H ถึง 3FFFH
0	1	0	2Y2	4000H ถึง 5FFFH
0	1	1	2Y3	6000H ถึง 7FFFH
1	0	0	1Y0	8000H ถึง 9FFFH
1	0	1	1Y1	A000H ถึง BFFFH
1	1	0	1Y2	C000H ถึง DFFFH
1	1	1	1Y3	E000H ถึง FFFFH

จะเห็นได้ว่า 74LS156 เป็นตัวถอดรหัสแบบเข้า 2 ค่า ถอดรหัสออกมา 4 ค่า แต่สามารถเลือกส่วนที่ 1 หรือส่วนที่ 2 มีเอาต์พุตออกได้ โดย A15 ถ้า A15 เป็น 0 จะเลือกส่วนที่ 2 แต่ถ้า A15 เป็น 1 จะเลือกส่วนที่ 1 สามารถดูผลความสัมพันธ์ระหว่างอินพุตและเอาต์พุตได้จากตารางที่ 5.3

ทีทีแอล (TTL) 8255

8255 Programmable Peripheral Interface (PPI) [3] เป็นไอซีแบบที่นิยมใช้งานกันมากมายสำหรับบอร์ดควบคุม ใช้พอร์ต 8255 1 ตัว ทำหน้าที่พอร์ตอินพุต/เอาต์พุต ถึง 24 บิต โดยมีตำแหน่งดังนี้

8255 ตำแหน่ง E000H+ 8255 (Offset Address) = ตำแหน่งจริง

พอร์ต A ตำแหน่ง E000H+ 00H = E000H

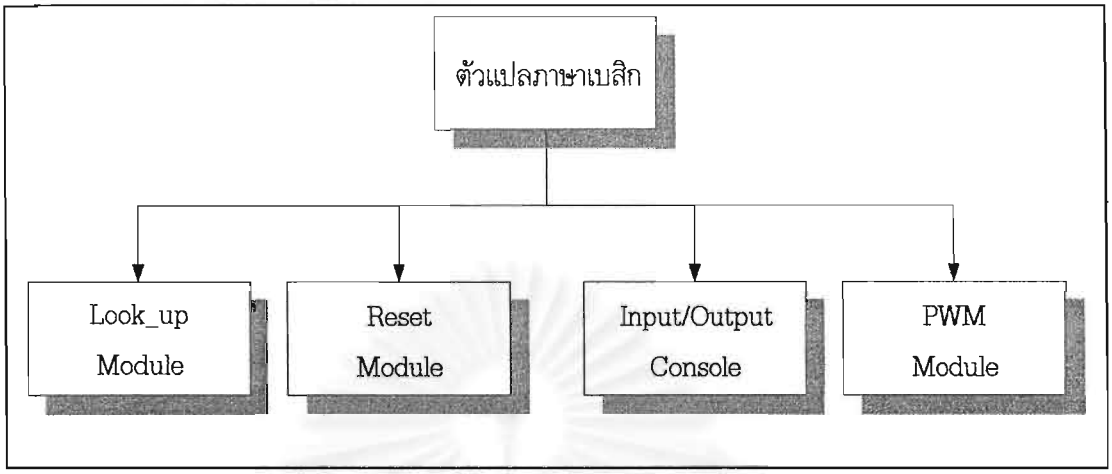
พอร์ต B ตำแหน่ง E000H+ 01H = E001H

พอร์ต C ตำแหน่ง E000H+ 02H = E002H

พอร์ตควบคุม ตำแหน่ง E000H+ 03H = E003H

ก่อนที่จะใช้งานพอร์ต 8255 ผู้ใช้ต้องทำการกำหนดแบบวิธีการทำงาน (Configuration) ของพอร์ต A, B และ C ให้เป็นพอร์ตอินพุตหรือเอาต์พุต โดยทำการเขียนรหัสควบคุม (Control code) ไปที่พอร์ตรับวิธีการทำงาน ซึ่งพอร์ตรับวิธีการทำงานนี้สามารถเขียนได้เท่านั้น ไม่สามารถอ่านได้

5.3 ส่วนรายละเอียดซอฟต์แวร์ของอีดีแอลเบสิก



รูปที่ 5.6 ส่วนประกอบย่อยของโมดูลอีดีแอลเบสิก

ส่วนประกอบของซอฟต์แวร์ของตัวแปลภาษาเบสิก (BASIC Interpreter) สามารถแบ่งย่อยเป็นโมดูลได้ดังรูปที่ 5.6

5.3.1 โมดูลส่วนเก็บคำสั่งและโปรแกรมย่อยคำสั่ง

หน้าที่ของโมดูลนี้ คือ กำหนดค่าเริ่มต้นของเรจิสเตอร์ในแบงก์ (Bank) ต่างๆ รวมถึงบิตตัวบ่งชี้ที่ใช้ตรวจสอบเงื่อนไขต่างๆ และยังเป็นส่วนที่เก็บคำเฉพาะ (Key word) ของคำสั่งที่จะเก็บโดยมีหมายเลขลำดับไว้ข้างหน้า ตัวเลขนี้จะนำไปชี้ตารางเวกเตอร์เพื่อกระโดดไปทำงานยังโปรแกรมย่อย

```
Module  Look_up
        Initialization_Table
            Addressing Register_bank
            Byte_Operation
            Flag_Operation
        User_Table
            Key word_Command
            Key word_Statement
            Key word_Operator
        Vector_Table
            Vector_Command
            Vector_Statement
            Vector_Operator
        Routine_Action
```

```
                    Routine Action_command
                    Routine_Action_statement
                    Routine_Action_Operator
          End Module
```

5.3.2 โมดูลส่วนอินพุตและเอาต์พุตพอร์ตอนุกรม

หน้าที่ของโมดูลนี้ คือ รับส่งรับส่งจากอินพุตเอาต์พุตของเบสิกผ่าน RS-232 โมดูลนี้จะรับข้อมูลจากตัวจับเวลาที่ 2 (Timer/Counter 2) ที่ได้จากการตรวจจับข้อมูลของแป้นในช่วงเริ่มต้นทำงาน แล้วคำนวณค่าอัตราการส่งข้อมูล (Baud Rate) จากนั้นนำค่าอัตราส่งข้อมูลมาใช้กับตัวจับเวลาที่ 1 (Timer 1) เพื่อกำหนดสัญญาณแบบอะซิงโครนัสสำหรับใช้ส่งข้อมูลต่อไป

```
Module  Input/Output Console
        Do While (Rx=0)
            Stay Here
        If (Rx=1) Then
            Read (Rx_buffer)
            Rx_buffer = Tx_buffer
            Do While (Tx=0)
                Stay Here
            If (Tx=1) Then
            End If
        End If
End Module
```

5.3.3 โมดูลส่วนกำหนดเริ่มต้นการทำงานและควบคุมการตั้งค่าใหม่

หน้าที่ของโมดูลนี้ก็ คือ ลบล้างข้อมูลของแรมภายในและแรมภายนอก, ตั้งค่าสแต็กข้อมูล, ตั้งค่าเงื่อนไขเรจิสเตอร์พิเศษ, กำหนดสัญญาณนาฬิกา, ตรวจจับสัญญาณทางพอร์ตอนุกรม เพื่อนำมาคำนวณอัตราเร็วสำหรับส่งข้อมูล และ ข้อความเริ่มต้นใช้งาน

```
Module  Reset_Control
        Setup Stack
        Clear Internal_RAM
        Clear External_RAM
        Setup Special_register
```

```
Load XTAL_value
While (Rx=0)
    Stay Here
If (Rx=1) Then
    Calculate Baud_rate
    If Keypress = Space_bar Then
        Sign_on_messages
    Else  Stay Here
End If
End Do
End Module
```

5.3.4 โมดูลสร้างสัญญาณมอดูเลตความกว้างพัลส์ (Pulse Width Modulation)

โมดูลนี้จะทำหน้าที่สร้างสัญญาณมอดูเลตความกว้างพัลส์ (PWM) โดยจะนำค่าความถี่จากผลึกมาคำนวณ โดยที่ความถี่ของสัญญาณ PWM ที่สร้างขึ้นจะขึ้นอยู่กับความถี่ที่ป้อนข้อมูลทางแผงแป้นอักขระนั้นเอง

```
Module  PWM
    Call Input/Output Console
    Reset_Frequency
    Calculate_PWM
    Out_PWM
End Module
```

ข้อความที่กล่าวมาข้างต้นได้อธิบายโครงสร้างและรายละเอียดในแต่ละโมดูลซอฟต์แวร์ของตัวแปลภาษาเบสิกสำหรับการควบคุม ซึ่งได้เพิ่มเติมคำสั่งบางคำสั่งเข้าไปเพื่อเพิ่มความสามารถในการแก้จุดบกพร่องโปรแกรมให้ดีขึ้นซึ่งจะขอกล่าวดังต่อไปนี้

คำสั่ง BRKP (Break point)

คำสั่งนี้อาศัยหลักการตรวจสอบตัวบ่งชี้จุดพัก (Break Flag) ที่กำหนดขึ้นกับแรมภายใน โดยในขณะที่ดำเนินโปรแกรม เมื่อพบคำสั่งนี้บรรทัดต่อไปจะถูกตั้งค่าตัวบ่งชี้จุดพัก และเมื่อตรวจสอบพบว่าตัวบ่งชี้จุดพักถูกตั้งค่าจะทำให้ข้อมูลตำแหน่งของบรรทัดถัดไปถูกเก็บลงในสแต็ก จากนั้นจะกระโดดออกมาที่โหมดคำสั่ง โมดูลซอฟต์แวร์จะแสดงอยู่ข้างล่าง

```
Module BRKP
    Set BRKP_bit_flag
    Save Line_number
```

```
        Print Next_line_number
        Goto Command_mode
    End Module
```

คำสั่ง CONT (Continuous)

เป็นคำสั่งที่ใช้สั่งให้โปรแกรมทำงาน หลังจากพักการทำงาน ซอฟต์แวร์โมดูลนี้จะลบล้างค่าตัวบ่งชี้จุดพักในหน่วยความจำแบบข้อมูล แล้วทำการบรรจุข้อมูลบรรทัดที่อยู่ในสแต็กออกมาทำการดำเนินโปรแกรม

```
Module  CONT
        Clear BRKP_bit_flag
        Get Line_number
        RUN Next_line_number
End Module
```

คำสั่ง PH0.

เป็นคำสั่งที่ใช้สำหรับดูค่าพารามิเตอร์ ค่าที่จะแสดงจะออกมาในรูปเลขฐาน 16 การทำงานของโมดูลซอฟต์แวร์จะอธิบายได้ดังนี้ ซอฟต์แวร์จะจองเนื้อที่ในหน่วยความจำสำหรับเก็บค่าตัวแปรและใช้ค่าตัวแปรอ้างอิงข้อมูลในหน่วยความจำออกมาแสดง เช่น ตัวแปร A ค่าแอสกีของ A=41H ตำแหน่งที่เก็บข้อมูลที่ตัวแปร A ชี้อยู่จะเท่ากับ 41H บวกด้วยค่าออฟเซตของตัวแปรในแรม แต่ในโปรแกรมกำหนดอยู่ที่ 5000H เพราะฉะนั้นค่าตำแหน่งที่ใช้ชี้ตัวแปร A จะอยู่ที่ 5014H ตัวชี้ข้อมูลจะนำข้อมูลตำแหน่งนี้ออกมาแสดง

```
Module Ph0.
        Read Variable
        Addressing (Variable) = Offset + ASCII (Variable)
        Get_Data from Address
        Call Input/Output Console
End Module
```

คำสั่ง PROG

คำสั่งที่ใช้ในการบันทึกข้อมูลจากแรมมอนิเตอร์ไปยังเอ็นวีแรม

```
Module PROG
        Load Source_program in RAM
        Desination = Source
        Do While (Flag_End = 0)
            Desination = Source + 1
```

End Module

คำสั่ง LOAD

คำสั่งนี้จะใช้ในการโหลดโปรแกรมจากเอ็นวีแรมมายังแรมมอนิเตอร์ ซอฟต์แวร์จะทำงานคล้ายกับคำสั่ง PROG แต่แตกต่างกันตรงที่ตำแหน่งของต้นทาง (Source) และปลายทาง (Destination) ของข้อมูล โมดูลรายละเอียดของซอฟต์แวร์จะแสดงรายรายเอียดอยู่ด้านล่าง

```
Module LOAD
        Load Source_program in NVRAM
        Destination = Source (NVRAM)
            Do While (Flag_End = 0)
                Destination = Source + 1
End Module
```

คำสั่ง ROM และ RAM

คำสั่งนี้จะใช้เลือกแบบวิธี (mode) การทำงานของระบบว่าจะให้ชี้ที่ไหน ถ้าในแบบวิธี (mode) แรมจะชี้ที่ตั้งแต่ตำแหน่ง 512H ขึ้นไปถึง 7FFFH และในแบบวิธี (mode) รอมจะชี้ตำแหน่งของข้อมูลเริ่มที่ 8000H ถึง FFFFH แต่รอมแบบวิธีนั้นสามารถเลือกโปรแกรมที่จะใช้งานได้ เช่น ROM1, ROM2 เป็นต้น

```
Module ROM, RAM
        Read Keypress
        Select Case Keypress
            Case Keypress = RAM
                Data Pointer = 512H
            Case Keypress = ROMn
                Call ROMn
End Module

Module ROMn
            Read (ROMn)
            Count = n
            Data_Pointer = NVRAM
            Do_While (Count < >n)
```

```
                    Search End_of_Text
                    Count = Count + 1
            End Module
```

คำสั่ง DIR

คำสั่ง DIR ซึ่งมาจากคำว่า Directory เป็นคำสั่งที่ใช้แสดงรายการโปรแกรมที่ถูกบันทึกในเอ็นวีแรม การทำงานของโมดูลนี้มีส่วนคล้ายกับโมดูล ROMn คือ มีการค้นหาค่าจำนวนโปรแกรม แต่ DIR จะแสดงรายการทั้งหมดตั้งแต่โปรแกรมแรกไปยังโปรแกรมสุดท้าย

```
        Module DIR
                        Data_pointer = NVRAM
        Loop:           Do While (Data < > End_of_Text)
                            Data_pointer = Data_pointer + 1
                        If  Data = End_of_Text Then
                            Print DIR_Count
                            DIR Count = DIR_Count + 1
                            If Data < > 0 Then
                                Goto Loop
                            End If
                        End If
        End Module
```

ตัวดำเนินการเกี่ยวกับเรจิสเตอร์พิเศษ (Special function register)

ตัวดำเนินการประเภทที่เกี่ยวกับเรจิสเตอร์พิเศษ

TCON, PCON, RCAP2, TIMER2, TIMER1, TIMER0, PORT1, IE, IP, TMOD

ตัวดำเนินการนี้จะมีลักษณะชื่อเหมือนกับเรจิสเตอร์พิเศษใน MCS-51 ไมโครคอนโทรลเลอร์ ซึ่งจหลักการในแต่ละคำสั่งจะคล้ายกันจึงขอสรุปได้ดังนี้

การทำงานของซอฟต์แวร์เรจิสเตอร์พิเศษในอีดีแอลเบสิก ชื่อตัวดำเนินการเรจิสเตอร์พิเศษจะถูกอ้างถึงเข้ากับตำแหน่งเรจิสเตอร์พิเศษในโครงสร้างภายในซีพียู

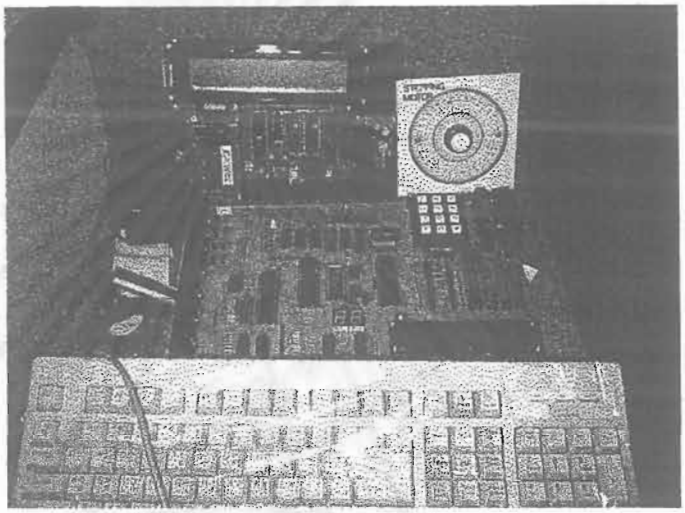
ตัวอย่าง เช่น ในซอฟต์แวร์จะกำหนดชื่อตัวดำเนินการ PORT1 ไว้ที่ตำแหน่งหน่วยความจำแบบโปรแกรมที่ 90H โดยที่ 90H จะตรงกับตำแหน่งเรจิสเตอร์ PORT1 ในซีพียูในไมโครคอนโทรลเลอร์ ดั้งนั้นเวลาที่ตัวดำเนินการถูกพิมพ์ลงไปก็จะอ้างอิงหน่วยความจำแบบโปรแกรมที่ตำแหน่งดังกล่าว ส่วนตัวดำเนินการอื่นก็มีลักษณะแบบเดียวกันนี้

บทที่ 6

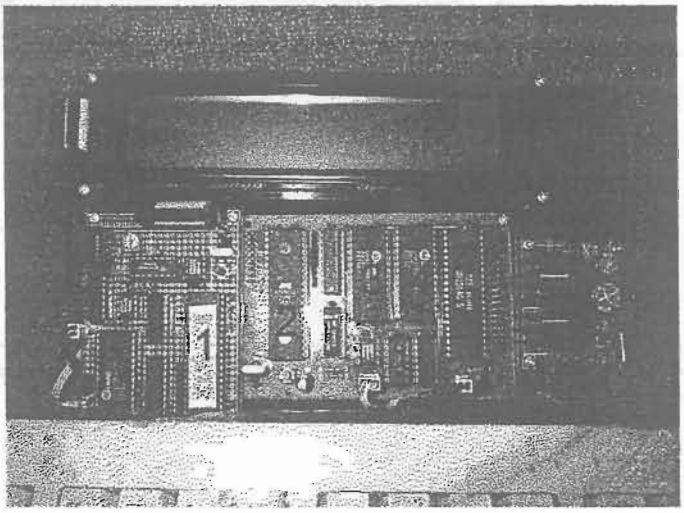
ผลการทดสอบ

6.1 ส่วนฮาร์ดแวร์จริงของระบบทั้งหมด

เมื่อออกแบบส่วนต่าง ๆ ของชุดฝึกทดลองเสร็จเรียบร้อยจะได้รูปร่างของระบบทั้งหมด ดังรูปที่ 6.1, 6.2 และ 6.3



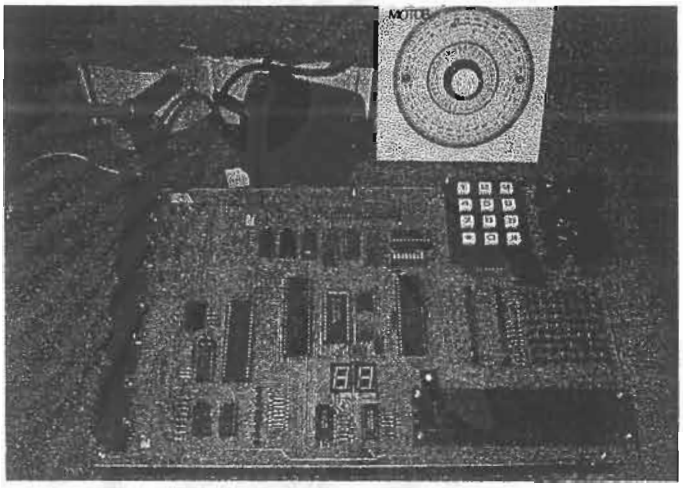
รูปที่ 6.1 ฮาร์ดแวร์จริงที่ได้จากการออกแบบระบบทั้งหมด



รูปที่ 6.2 ฮาร์ดแวร์ของแผงอีดีแอลเบสิกและแอลซีดีเทอร์มินัล

จากรูปที่ 6.2 แต่ละหมายเลขจะแสดงความหมายดังนี้

1. แอลซีดีเทอร์มินัลถูกต่อเชื่อมกับแผงอีดีแอลเบสิกทางพอร์ต RS-232
2. ซีพียูที่บรรจุโปรแกรมอีดีแอลเบสิกไว้ภายใน
3. แรมมอนิเตอร์สำหรับเก็บโปรแกรมในขณะที่ใช้งาน
4. เอ็นวีแรมไว้เก็บโปรแกรมภาษาเบสิกหลังการพัฒนาเสร็จสมบูรณ์
5. 8255 สำหรับขยายไอโอ
6. ชุดจ่ายไฟกระแสตรง

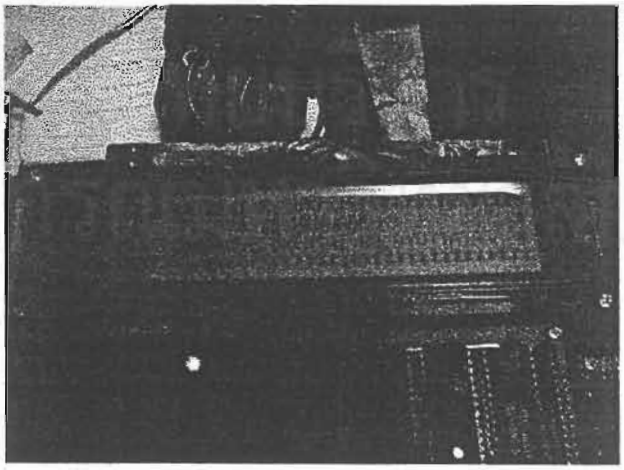


รูปที่ 6.3 แผงชุดทดลองทั้งหมด

6.2 ผลการทดสอบการทำงานของระบบ

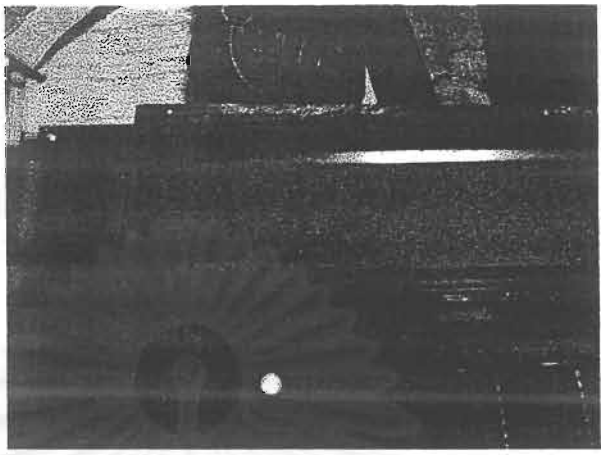
6.2.1 ทดสอบการทำงานเบื้องต้นของระบบ

ผู้วิจัยได้นำแผงอีดีแอลเบสิกมารวมกับแผงแอลซีดีเทอร์มินัลและยึดบนพลาสติกเดียวกัน เพื่อทำการทดสอบต่อแผงเป็นอักขระของคอมพิวเตอร์เข้ากับระบบ แล้วเริ่มจ่ายไฟเข้าเครื่อง



รูปที่ 6.4 หน้าจอแอลซีดีเมื่อจ่ายไฟครั้งแรก

ทดสอบกดปุ่ม Space bar ที่แป้นอักขระของคอมพิวเตอร์ หน้าจอแอลซีดีจะปรากฏประโยคเริ่มต้น ดังรูปที่ 6.5



รูปที่ 6.5 ข้อความที่ปรากฏหลังกดปุ่ม Space bar

ทดสอบหน่วยความจำ เนื่องจากวงจรใช้แรมมอนิเตอร์ 62256 มีขนาด 32 กิโลไบต์ เพราะว่าหน่วยความจำสูงสุดได้ประมาณ 32 กิโลไบต์ ใช้คำสั่ง

> PRINT MTOP

> 32512

ทดสอบความถี่จากผลึกโดยการพิมพ์คำสั่งดังนี้ ควรได้เท่ากับค่าความถี่ที่ใส่เข้าไปในวงจรคือ 11.0529 MHz

> PRINT XTAL

> XTAL = 11095200

ทดสอบการรันโปรแกรม โดยพิมพ์โปรแกรมแล้วทดสอบการดำเนินโปรแกรมแล้วดูผลหน้าจอแอลซีดีดังนี้

> 10 FOR I = 1 TO 10

> 20 PRINT I

> 30 NEXT I

> RUN

ผลการทดสอบปรากฏว่าโปรแกรมจะเริ่มพิมพ์ค่า 1 จนถึงค่า 10 เรียงกันไปตามลำดับ ลองใช้คำสั่ง List เพื่อดูผลการทดสอบโปรแกรมที่พิมพ์เข้าไป

> LIST

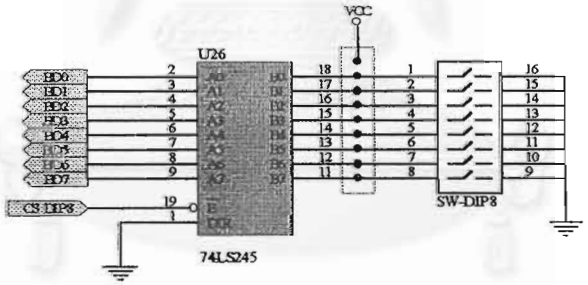
```
> 10 FOR I = 1 TO 10
> 20 PRINT I
> 30 NEXT I
> READY
>
```

6.2.2 ผลการทดสอบโปรแกรมภาษาเบสิกเทียบกับภาษาแอสเซมบลีโดยใช้ไมโครคอมพิวเตอร์

ผู้วิจัยได้นำอีดีแอลเบสิกที่พัฒนาขึ้นมาเพื่อทดสอบการทำงานของโปรแกรม ในการทดสอบนี้จะเป็นการทดสอบร่วมกับไมโครคอมพิวเตอร์ โดยต่อผ่านพอร์ต RS-232 มายังแผงอีดีแอลเบสิก ส่วนโปรแกรมที่จะทดสอบก็เป็นโปรแกรมง่าย ๆ คือ อ่านข้อมูลจากพอร์ตและเขียนข้อมูลไปยังพอร์ต เพราะทั้ง 2 อย่างนี้เป็นหลักสำคัญเกี่ยวกับการส่งข้อมูลไปควบคุมส่วนต่าง ๆ ซึ่งการทดลองได้นำแผงอีดีแอลเบสิกมาต่อแผงฝึกทดลอง โดยจะทดสอบดำเนินการโปรแกรมทั้ง 2 ภาษา คือ ภาษาแอสเซมบลีและภาษาเบสิก วงจรที่จะอ่านข้อมูลดังรูป 6.6

ตำแหน่งพอร์ตอยู่ที่ 0E0A0H คือตำแหน่งของดิปสวิตช์ ผู้วิจัยได้เขียนโปรแกรมเพื่อทดสอบเป็นภาษาแอสเซมบลี โปรแกรมนี้จะอ่านข้อมูลจากพอร์ตแล้วส่งค่าออกไปที่พอร์ต 1 ของไมโครคอนโทรลเลอร์มีดังนี้

ทดสอบอ่านข้อมูล



รูปที่ 6.6 วงจรอ่านข้อมูลจากดิปสวิตช์

ตำแหน่งพอร์ตอยู่ที่ 0E0A0H คือตำแหน่งของดิปสวิตช์ผู้เขียนได้เขียนโปรแกรมเพื่อทดสอบเป็นภาษาแอสเซมบลี โปรแกรมนี้จะอ่านข้อมูลจากพอร์ตแล้วส่งค่าออกไปที่พอร์ต 1 ของไมโครคอนโทรลเลอร์มีดังนี้

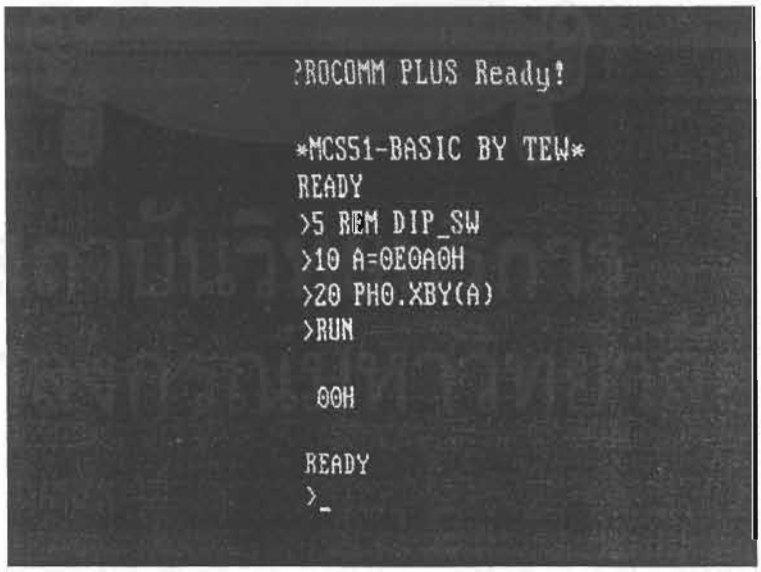
```
;*****************************************************************
;                  โปรแกรมอ่านข้อมูลจากพอร์ต 0E0A0H
;*****************************************************************
DIP_SW8     EQU   0E0A0H

            ORG 0000H
```

```
CALL  START_UP
      ORG 100H
START_UP:   MOV     DPTR, #DIP_SW8
            MOVX    A,@DPTR
            MOV     P1, A
            JMP     $
            END
```

นำโปรแกรมภาษาแอสเซมบลีมาคอมไพล์โดยคอมไพเลอร์แล้วบรรจุลงแผงควบคุมโดยใช้ไมโครคอมพิวเตอร์ต่อเข้ากับอุปกรณ์บรรจุโปรแกรม SPI ของบริษัท ศิลา จำกัด แล้วเชื่อมต่อกับแผงควบคุมอีกทอดหนึ่ง เมื่อบรรจุเสร็จสมบูรณ์ใช้ลอจิกโพรบตรวจสอบผลที่ได้จากพอร์ต 1

จากนั้นเปลี่ยนมาทดสอบกับภาษาเบสิก โดยวิธีการทดสอบจะทดสอบกับไมโครคอมพิวเตอร์ ขั้นตอนการทดสอบจะมีดังนี้

- เปิดเครื่องคอมพิวเตอร์แล้วเข้าโปรแกรม Procomm Plus ดำเนินการแฟ้มข้อมูล PCP. Bat หน้าจอคอมพิวเตอร์จะปรากฏคำว่า "Procomm Plus Ready" ขึ้น
- ตั้งค่าโดยกดปุ่ม ALT-P ตั้งไปที่อัตราการส่งข้อมูล 300 บิตต่อวินาที, 8 บิต, ไม่ใช้พาริตีบิต, บิตหยุด 1 บิต, บิตเริ่มต้น 1 บิต ให้กดเลือกหมายเลข 1 จากนั้นเก็บค่าที่ตั้งโดยการกดปุ่ม ALT-S
- จ่ายกระแสไฟฟ้าเข้าแผงอีดีแอลเบสิกแล้วกดปุ่ม Space bar หน้าจอคอมพิวเตอร์จะปรากฏข้อความดังในรูปที่ 6.7



รูปที่ 6.7 การทดสอบโปรแกรมอ่านข้อมูลโดยไมโครคอมพิวเตอร์

- พิมพ์โปรแกรมเป็นภาษาเบสิก ดังนี้

```
> 10   A=XBY (0E0A0H)        ; นำข้อมูลที่ตำแหน่ง 0E0A0H ไปเก็บที่ตัวแปร
> 20   PH0.XBY(A)            ; พิมพ์ข้อมูลที่อยู่ในตัวแปร A ออกหน้าจอ
> RUN
```

ทดสอบเขียนข้อมูล

ทดสอบเขียนข้อมูลไปยังแผงอีดีแอลเบสิกโดยใช้ทั้งภาษาแอสเซมบลีและภาษาเบสิก โดยจะเขียนข้อมูลไปยังพอร์ต C ของ 8255 ตัวที่ 3 ที่ต่อกับแอลอีดีมอนิเตอร์ 8 ดวง ซึ่งตำแหน่งพอร์ตอยู่ที่ 0E0BBH โดยจะทดสอบกับภาษาแอสเซมบลีก่อนโปรแกรมที่ผู้วิจัยมีดังนี้

```
*************************************************************
*          โปรแกรมเขียนข้อมูลไปยังแอลอีดี                  *
*************************************************************
        PC      EQU   0E0BBH   ; พอร์ต C ของ 8255
        CP      EQU   0E0BCH   ; พอร์ตควบคุมของ 8255
         ORG 0000H
      CALL   START_UP
         ORG 100H
      START_UP: MOV    DPTR,#CP
                MOV    A,#80H       ; กำหนดพอร์ตทั้งหมดเป็นเอาต์พุต
                MOVX   @DPTR,A
                MOV    A,#0FFH      ; ให้แอลอีดีทุกดวงติด
                MOV    DPTR,#PC
                MOVX   @DPTR,A      ; เขียนข้อมูลไปยังพอร์ต C
                JMP    $
                END
```

จากนั้นเปลี่ยนการทดสอบมาเขียนเป็นภาษาเบสิกโดยการทำงานของโปรแกรมยังคงเหมือนเดิมแต่รูปแบบคำสั่งเปลี่ยนไป

```
> 10 PC = 0E0BBH : CP = 0E0BCH
> 20 XBY(CP) = 80H                ; ให้ทุกพอร์ตของ 8255 เป็นเอาต์พุตหมด
> 30 XBY(PC) = 0FFH               ; ให้แอลอีดีทุกดวงติดหมด
> RUN
```

จะเห็นได้ชัดว่าโปรแกรมที่เขียนด้วยภาษาเบสิกนี้สั้นและกะทัดรัดไม่ยุ่งยากเหมือนภาษาแอสเซมบลีซึ่งเวลาอ้างอิงตำแหน่งที่มากกว่า 0FFH หรือ 256 จะต้องใช้เรจิสเตอร์ข้อมูล (DPTR) ทำการอ้างอิงตำแหน่งทุกครั้ง เวลาเขียนหรืออ่านข้อมูลต้องนำข้อมูลมาเก็บในตัวสะสมข้อมูล (Accumulator) ทุกครั้ง ซึ่งยุ่งยากเมื่อเขียนโปรแกรมที่ซับซ้อนและมีขนาดยาว

ฉะนั้นภาษาเบสิกจึงเหมาะสำหรับเรียนรู้ระบบการทำงานพื้นฐานของอินพุตและเอาต์พุตของไมโครคอนโทรลเลอร์เป็นอย่างดี

6.2.3 ทดสอบระบบร่วมกับแอลซีดีเทอร์มินัล

ผู้วิจัยนำแผงอีดีแอลเบสิกมาทดสอบร่วมกับแอลซีดีเทอร์มินัล โดยที่ขั้นตอนการทดสอบมีดังนี้

- จ่ายไฟเข้าระบบ แล้วกดปุ่ม Space bar เพื่อเข้าสู่ระบบ
- พิมพ์โปรแกรมที่จะทดลอง ดังนี้

```
> 5     REM  TIMER A minute
> 10    TIMER = 0  : CLOCK1 : ONTIME1, 100 : D0
> 20    WHILE  TIMER < 10   : END
> 100   A = TIME
> 110 PRINT " TIMER INTERRUPT AT - ", A, " SECONDS "
> 120 ONTIME A+1, 100       : RETI
> RUN
```

จะได้ผลการทดลองดังนี้

```
TIMER INTERRUPT AT - 1 SECONDS
TIMER INTERRUPT AT - 2 SECONDS
        ⋮
        ▼
TIMER INTERRUPT AT - 60 SECONDS
READY
```

- จากนั้นทำการบันทึกโปรแกรมลงในเอ็นวีแรมโดยใช้คำสั่ง PROG2 เพื่อทดสอบการทำงานอัตโนมัติเมื่อเปิดเครื่อง
- ปิดเครื่องแล้วจ่ายกระแสไฟฟ้าเข้าเครื่องอีกครั้งจะได้ผลการทดลอง

จากผลการทดสอบเมื่อเปิดเครื่องอีดีแอลเบสิกจะตรวจสอบข้อมูลที่ 8000H ในเอ็นวีแรมว่าเท่ากับ AAH หรือไม่ ถ้าเท่าก็แสดงระบบจะเรียกโปรแกรมในเอ็นวีแรมขึ้นมาทำงานอย่างอัตโนมัติ โดยไม่ต้องต่อแป้นอักขระใช้งานแต่อย่างใด

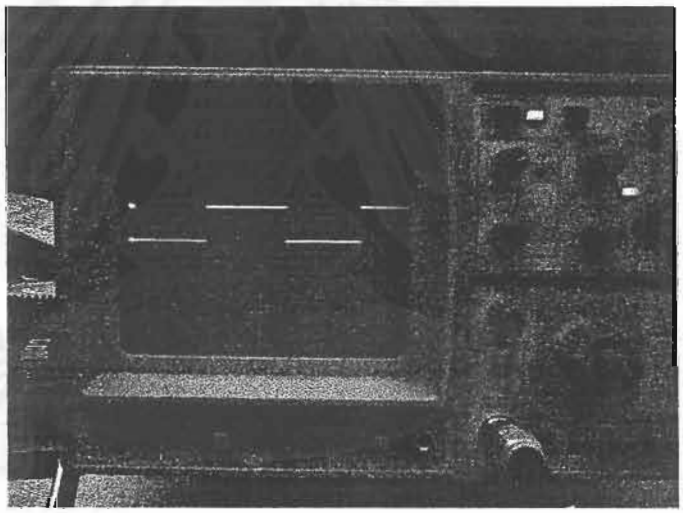
6.2.4 ทดสอบความเร็วในการทำงานของชุดคำสั่ง

ผู้วิจัยได้นำแอลซีดีเทอร์มินัลและอีดีแอลเบสิกมาทดสอบร่วมกัน โดยขั้นตอนการทดสอบดังนี้

- ผู้วิจัยทดสอบเขียนโปรแกรมส่งข้อมูลออก 8255 ที่ตำแหน่ง 0E0BAH ที่บิต 7 เป็นรูปคลื่นสี่เหลี่ยม แล้ววัดค่าความกว้างของพัลส์ในช่วงเปลี่ยนระดับแรงดันเป็น 0 โวลต์ โดยใช้ออสซิลโลสโคป

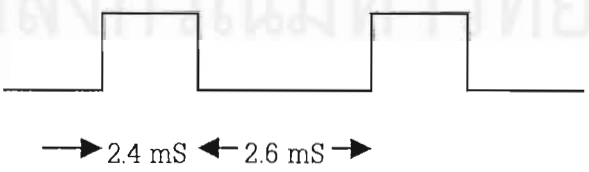
```
> 10  PC = 0E0BAH  : CP = 0F0BBH
> 20   XBY (CP) = 80H    ; กำหนดรูปแบบการทำงาน 8255 เป็นเอาต์พุตทุกพอร์ต
> 30   XBY (PC) = 00H    ; ส่งค่า 0 ออกพอร์ต C บิต 7
> 40   XBY (PC) = 80H    ; ส่งค่า 1 ออกพอร์ต C บิต 7
> 50   GOTO 30           ; วนไปเรื่อย
> PROG
```

ผลการทดลองดังรูปที่ 6.8

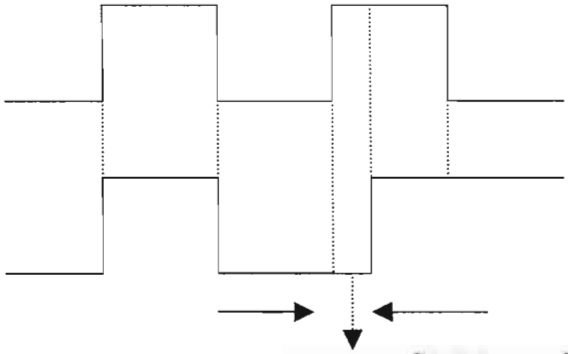


รูปที่ 6.8 รูปคลื่นสี่เหลี่ยมที่ขา 10 ของ 8255

ตั้งสโคปที่ย่าน 1 mS ต่อ 1 ช่องสัญญาณ วัดได้ค่า 5 mS ช่วงแรงดันเป็น 0 เท่ากับ 2.6 mS



- ทดสอบเพิ่มคำสั่งเข้าไประหว่างบรรทัดที่ 30 ถึง 40 ในช่วงพัลส์เป็น 0 แล้วดำเนินการโปรแกรม แล้วทำการวัดรูปคลื่นโดยสโคป จากนั้นนำค่าเวลาช่วงพัลส์เป็น 0 มาลบกับผลการทดลองในช่วงแรก



สามารถเขียนสรุปได้ดังตารางที่ 6.1

ตารางที่ 6.1 ค่าคาบเวลาในการกระทำชุดคำสั่ง

คำสั่ง	ความเร็วในการทำงาน	คำสั่ง	ความเร็วในการทำงาน	คำสั่ง	ความเร็วในการทำงาน
CLEARI	0.2 ms	+	1.8 ms	CBY( )	0.8 ms
CLOCK1	0.2 ms	/	6.8 ms	DBY( )	0.2 ms
CLOCK0	0.2 ms	**	8 ms	XBY( )	3.4 ms
DO -UNTIL	1.8 ms	*	7 ms	IE	0.2 ms
FOR-TO-STEP	5.4 ms	-	1.6 ms	IP	0.2 ms
GOSUB	1 ms	.AND.	0.2 ms	PORT1	0.2 ms
GOTO	3.4 ms	.OR.	0.2 ms	PCON	0.2 ms
IF-THEN-ELSE	1.8 ms	.XOR.	0.2 ms	RCAP2	0.2 ms
LET	1.2 ms	ABS( )	1.9 ms	T2CON	0.2 ms
ONTIME	3 ms	NOT( )	0.2 ms	TCON	0.2 ms
ONEX1-RETI	37.4 ms	INT( )	1 ms	TMOD	0.2 ms
PRINT	47.4 ms	SGN( )	1.8 ms	TIME	2 ms
PH0.	41.4 ms	SQR( )	4.2 ms	TIMER0	0.2 ms
PUSH	0.2 ms	LOG( )	6.2 ms	TIMER1	0.2 ms
POP	0.2 ms	EXP( )	8 ms	TIMER2	0.2 ms
REM	0.2 ms	SIN( )	7.4 ms		
STRING	1.2 ms	COS( )	6.8 ms		
		TAN( )	8.4 ms		

ในตารางที่ 6.1 เป็นตารางที่ได้จากการทดสอบการหาความเร็วของชุดคำสั่งแต่ละคำสั่งในอีดีแอลเบสิก โดยจะเห็นค่าความเร็วสูงสุดของชุดคำสั่งอยู่ที่ 0.2 ms และค่าความเร็วในการประมวลผลของชุดคำสั่งต่ำสุดเท่ากับ 47.4 ms

ต่อไปผู้วิจัยจะทำการทดสอบหาความเร็วในการประมวลในโปรแกรมที่มีฟังก์ชันการทำงานที่เหมือนกันระหว่างภาษาแอสเซมบลี้และอีดีแอลเบสิกโดยจะใช้ฟังก์ชันพื้นฐานในหัวข้อ 6.2.2 เป็นการส่งข้อมูลออกทางพอร์ต

C ของ 8255 เป็นรูปคลื่นสี่เหลี่ยมแล้วใช้ออสซิสโลสโคปวัดหาคาบเวลาในการผลิตสัญญาณสี่เหลี่ยม 1 ลูกแล้วนำมาเปรียบเทียบค่าได้ โปรแกรมที่ทดสอบจะแสดงดังต่อไปนี้

```
**********************************************************
*          โปรแกรมเขียนข้อมูลไปยังแอลอีดี                *
**********************************************************
        PC      EQU   0E0BBH    ; พอร์ต C ของ 8255
        CP      EQU   0E0BCH    ; พอร์ตควบคุมของ 8255
        ORG 0000H
      CALL   START_UP
        ORG 100H
      START_UP: MOV    DPTR,#CP
                MOV    A,#80H       ; กำหนดพอร์ตทั้งหมดเป็นเอาต์พุต
                MOVX   @DPTR,A
       LOOP:    MOV    A,#0FFH      ; ให้แอลอีดีทุกดวงติด
                MOV    DPTR,#PC
                MOVX   @DPTR,A      ; เขียนข้อมูลไปยังพอร์ต C
                MOV    A,#00
                MOVX   @DPTR,A
                JMP     LOOP
                END
```

จากนั้นเปลี่ยนการทดสอบมาเขียนเป็นภาษาเบสิกโดยการทำงานของโปรแกรมยังคงเหมือนเดิมแต่รูปแบบคำสั่งเปลี่ยนไป

```
> 10 PC = 0E0BBH : CP = 0E0BCH
> 20 XBY(CP) = 80H                ; ให้ทุกพอร์ตของ 8255 เป็นเอาต์พุตหมด
> 30 XBY(PC) = 0FFH               ; ให้แอลอีดีทุกดวงติดหมด
> 40 XBY(PC) = 00
> 50 GOTO 30
> RUN
```

จากผลการทดสอบที่ได้จากการวัดรูปคลื่นสัญญาณ 1รอบ (Cycle) ได้ผลสามารถเขียนในตารางที่ 6.2 ดังต่อไปนี้

ตารางที่ 6.2 เปรียบเทียบความเร็วระหว่างภาษาเบสิกและภาษาแอสเซมบลีในโปรแกรมกำเนิดสัญญาณสี่เหลี่ยม

	ภาษาแอสเซมบลี	ภาษาเบสิก
ความถี่ที่วัดได้ต่อ 1 ลูกคลื่น	40 KHz	66.66 Hz
เวลาต่อ 1 ลูกคลื่น	25 ไมโครวินาที	15 มิลลิวินาที

จะเห็นได้ว่าภาษาแอสเซมบลีสามารถผลิตสัญญาณสี่เหลี่ยมได้เร็วกว่าอีดีแอลเบสิกประมาณ 600 เท่า

โปรแกรมต่อไปนี้เป็นโปรแกรมตรวจสอบความเร็วในการทำงาน (Beuch work) ผู้วิจัยได้สร้างโปรแกรมขึ้นมาเองเพื่อเปรียบเทียบความเร็วของการทำงานในภาษาเบสิก เป็นโปรแกรมขับแอลซีดีและมีคำสั่งของภาษาเบสิกที่ใช้งานพื้นฐานทุกคำสั่งมาทดสอบ การวัดเวลาทำงานของโปรแกรมทำได้ดังนี้โดยเริ่มต้นโปรแกรมให้ส่งข้อมูลตรรกะ 1 ออกพอร์ต 1 ของไมโครคอนโทรลเลอร์และสิ้นสุดโปรแกรมให้ส่งค่าตรรกะ 0 ออกไปยังพอร์ต 1 แล้วใช้ออสซิโลสโคปวัดคาบเวลาในช่วงตรรกะเป็น 1 ของทั้ง 2 ภาษาแล้วเปรียบเทียบค่าที่ได้ในตารางที่ 6.3 ส่วนของโปรแกรมภาษาเบสิกจะมีดังนี้

```
5   REM LCD_DISPLAY :PORT1=0FFH: A=0FFH : B=0
10  COM = 0E0ACH : BUSY = COM+1 : WR = COM+2
20  GOSUB 120
30  FOR I=1 TO 6                          ; ตัวอักษร 6 ตัว
40  READ B                                ; อ่านข้อมูล
50  XBY (WR) = B                          ; เขียนไปยังแอลซีดี
60  GOSUB  200
70  NEXT I
80  RESTORE
90  PORT1=00:END
100 DATA 41H, 42H, 43H, 44H, 45H, 46H
120 XBY (COM) = 30H                       ; ตั้งเงื่อนไขแอลซีดี 8 บิต 1 บรรทัด
130  GOSUB  200
140 XBY (COM) = 0FH                       ; เปิดหน้าจอแอลซีดี
150 GOSUB  200
160 XBY (COM) = 6                         ; เพิ่มค่าในแรมขึ้น
170 GOSUB  200
180 XBY (COM) = 1                         ; ลบล้างหน่วยความจำ
190 GOSUB  200
200  RETURN
210 A= XBY (BUSY)                         ; ตรวจสอบสถานะของแอลอีดี
220 IF A > 80H THEN GOTO 200
230 RETURN
```

ตารางที่ 6.3 เปรียบเทียบเวลาการดำเนินการโปรแกรมทดสอบการทำงาน (Beuch work)

	ภาษาแอสเซมบลี	ภาษาเบสิก
เวลาในการกระทำโปรแกรม	28 มิลลิวินาที	0.5 วินาที

ผลการทดลองที่ได้โดยวัดช่วงที่ตรรกะเป็น 1 โดยใช้ออสซิโลโคปแบบบันทึกรูปคลื่นที่ได้แล้วนำมาเปรียบเวลาในการทำงานของโปรแกรมทั้ง 2 ภาษาแอสแซมบลีจะใช้เวลาในการเขียนข้อมูลไปยังแอลซีดีประมาณ 28 มิลลิวินาที และอีดีแอลเบสิกจะใช้เวลาในการเขียนข้อมูลไปยังจอแอลซีดี เท่ากับ 0.5 วินาที ภาษาแอสเซมบลีจะเร็วกว่าอีดีแอลเบสิกอยู่ 17 เท่า

จาก 2 ตัวอย่างที่ได้ทดสอบหาค่าความเร็วในการประมวลผลของโปรแกรมทั้ง 2 ภาษา ภาษาแอส-แซมบลีจะได้เปรียบตรงที่ใช้เวลาในการประมวลผลน้อยกว่า ทั้งนี้เป็นเพราะว่าภาษาแอสเซมบลีเสียเวลาในการแปลคำสั่งให้กับซีพียูน้อยกว่าภาษาเบสิกนั้นเอง แต่อย่างไรก็ตามภาษาเบสิกก็ยังมีข้อได้เปรียบตรงที่รูปแบบภาษานั้นทำความเข้าใจง่ายและมีความกระทัดรัด ไม่ยุ่งยากในการเขียนโปรแกรม เหมาะสำหรับนำไปใช้งานที่ต้องการความเร็วในการประมวลผลไม่สูงมากนัก เช่น เหมาะสำหรับโปรแกรมสแกนแป้นอักขระ และโปรแกรมประเภทที่มีการรอในการขัดจังหวะเพื่อเริ่มทำงาน เพราะว่าอีดีแอลเบสิกสามารถเขียนโปรแกรมกำเนิดสัญญาณสี่เหลี่ยมได้สูงสุดไม่เกินประมาณ 70 Hz ซึ่งไม่สามารถนำไปใช้งานที่การรับส่งข้อมูลในอัตราเร็วสูงๆได้

6.2.5 ทดสอบโดยใช้นิสิตทดลองใช้งานจริง

ผู้วิจัยได้ให้นิสิตระดับประกาศนียบัตรวิชาชีพชั้นสูง (ปวส) ชั้นปีที่ 1 แผนกช่างอิเล็กทรอนิกส์ โรงเรียนเทคโนโลยีกรุงเทพ ทำการเรียนรู้ชุดฝึกทดลองใช้เวลาประมาณ 3 วัน วันละ 3 ชั่วโมงทั้งหมด 8 การทดลอง โดยให้เรียนบทเรียนเพื่อทำการทดลองกับแผงทดลองและฝึกการเขียนโปรแกรมเพื่อควบคุมด้วยภาษาเบสิก จากนั้นผู้วิจัยได้ทดสอบให้นิสิตเขียนโปรแกรมส่งข้อมูลออกแอลอีดีแบบ 7 ส่วนโดยให้แสดงผลตัวเลข 0 ถึงตัวเลข 9 เมื่อถึงเลข 9 ให้วนมาแสดงผลที่ 0 อีกครั้ง ผลปรากฏว่านิสิตสามารถทำความเข้าใจชุดฝึกทดลองและเขียนโปรแกรมภาษาเบสิกได้ภายในระยะเวลา 3 ชั่วโมงได้ ซึ่งใช้เวลาในการทำความเข้าใจและเขียนโปรแกรมเป็นระยะเวลาสั้นกว่าการเรียนรู้โดยใช้ภาษาแอสเซมบลี

บทที่ 7

สรุปและข้อเสนอแนะ

7.1 สรุป

วิทยานิพนธ์นี้ได้นำเสนอระบบการเรียนรู้ไมโครคอนโทรลเลอร์โดยใช้ภาษาเบสิก ที่สามารถใช้งานได้อย่างอิสระ มีขนาดเล็ก น้ำหนักเบาราคาถูก และมีชุดคำสั่งที่จำง่ายไม่มากนัก รายละเอียดของงานวิจัยในวิทยานิพนธ์ฉบับนี้จะขอสรุปดังต่อไปนี้

1. พัฒนาโปรแกรมอีดีแอลเบสิกจาก BASIC-52.SRC โดยเพิ่มคำสั่งสำหรับใช้งานควบคุม เหมาะสำหรับการทดลองและเรียนรู้ตั้งแต่นิสิตระดับ ประกาศณียบัตรวิชาชีพ ไปจนถึงระดับมหาวิทยาลัย

2. ออกแบบระบบด้านฮาร์ดแวร์ ซึ่งประกอบด้วย แอลซีดีเทอร์มินัล, แผงอีดีแอลเบสิก, และ แผงฝึกทดลองที่ใช้สำหรับบทเรียน

3. เขียนบทเรียนที่ใช้ในการทดลองแต่ละการทดลอง ซึ่งสามารถนำมาใช้ในการเรียนรู้ด้วยตนเองสำหรับผู้ที่สนใจทั่วๆไป

4. ได้ทดสอบระบบทั้งหมดโดยให้ใช้งานจริง

ชุดฝึกทดลองนี้มีข้อดีเหนือกว่าชุดฝึกทดลองในท้องตลาดทั้งในและต่างประเทศดังต่อไปนี้

- ทำงานได้อย่างอิสระ ไม่ต้องใช้คอมพิวเตอร์จึงทำให้สะดวกในการใช้งาน
- มีชุดคำสั่งพิเศษที่ช่วยในการแก้จุดบกพร่องโปรแกรม
- สามารถใช้ควบคุมได้ทั้งเป็นเอาต์พุตและอินพุต
- สามารถเก็บข้อมูลโปรแกรมที่พัฒนาโดยไม่ต้องพึ่งไฟเลี้ยง
- มีความสามารถในการเขียนโปรแกรมด้วยความยาวสูงสุด 614 บรรทัด
- มีโมดูลสำหรับทำการต่อทดลอง
- มีคู่มือและบทเรียนในแต่ละการทดลอง
- เหมาะสำหรับงานควบคุมที่ใช้ความเร็วในการควบคุมไม่สูงมาก
- อุปกรณ์ทุกชิ้นหาซื้อในประเทศได้ง่ายและมีราคาถูก เช่น แผงแป้นอักขระของคอมพิวเตอร์สำหรับมาต่อใช้งานป้อนข้อมูล

7.2 ข้อเสนอแนะ

ในการทดสอบของระบบการเรียนรู้ไมโครคอนโทรลเลอร์โดยใช้ภาษาเบสิก ผู้วิจัยได้พบสาเหตุของปัญหาอยู่หลายปัญหา เช่น

1. ความเร็วในการประมวลผลของตัวแปลภาษาเบสิก ซึ่งช้ากว่าภาษาแอสเซมบลีอยู่ประมาณ 20 ถึง 600 เท่า ซึ่งไม่เหมาะสมที่จะเขียนโปรแกรมดำเนินการด้วยความเร็วสูง ๆ

2. ระบบชุดคำสั่ง ที่มีชุดคำสั่งปฏิบัติการโปรแกรมเกี่ยวกับเงื่อนไขของระบบหลาย ๆ เงื่อนไข ตัวอย่างคำสั่ง เช่น DIM (x,y) เป็นต้น ซึ่งทำให้ต้องเขียนโปรแกรมยาว ๆ

3. การเก็บข้อมูลในเอ็นวีแรมที่ไม่ปลอดภัย จากการทดสอบพบว่าเมื่อต่อแผงฝึกทดลองเข้ากับแผงอีดีแอลเบสิก พบว่าเวลาเปิดแผงฝึกทดลองหลังเปิดแผงอีดีแอลเบสิก พบว่ามีแรงดันกระชากค่าต่าง ๆ หนึ่งเข้าไปในระบบของเอ็นวีแรม ทำให้ข้อมูลบางส่วนที่เก็บไว้หายไป ดังนั้นการใช้เอ็นวีแรมเก็บโปรแกรมขณะใช้งานระบบจึงไม่ค่อยปลอดภัยนัก

4. ความล้าหลังทางด้านเทคโนโลยีของแผงฝึกทดลอง ซึ่งทำให้วิสัยทัศน์ในการเรียนรู้ของผู้ใช้ชุดฝึกทดลองนั้นแคบลงไป

5. วิธีการใช้งานของแผงอีดีแอลเบสิก เช่น ชุดคำสั่ง BRKP และ CONT ต้องพิมพ์เข้าไปคั่นกับหมายเลขบรรทัดทุกครั้ง หากเกิดการหยุดการทำงาน 100 จุด ต้องพิมพ์ BRKP 100 ครั้ง ซึ่งมีความซับซ้อนในการใช้งานเป็นอย่างมาก โดยเฉพาะโปรแกรมที่มีความซับซ้อน

ดังนั้นผู้วิจัยจึงขอเสนอแนวทางที่จะแก้ไขปัญหาดังกล่าวนี้ ได้แก่

1. เนื่องจากอีดีแอลเบสิกดำเนินคำสั่งแต่ละคำสั่งได้ช้า ผู้วิจัยได้เสนอแนวความคิดที่จะเพิ่มความเร็วในการดำเนินโปรแกรมให้มีความเร็วมากขึ้น โดยการเปลี่ยนไมโครคอนโทรลเลอร์เป็นตระกูลที่มีความเร็วสูง ตัวอย่าง เช่น ไมโครคอนโทรลเลอร์ 89C51RD มีขนาดหน่วยความจำแบบแฟลช (Flash) ขนาด 64 กิโลไบต์ สามารถใช้สัญญาณนาฬิกาได้ถึง 22 MHz แต่จะต้องเปลี่ยนแปลงแก้ไขค่าพารามิเตอร์ในโปรแกรมของตัวแปลภาษาเบสิกเป็น 22 MHz จึงจะสามารถทำงานได้อย่างมีประสิทธิภาพสูง

2. ผู้วิจัยได้เสนอความคิดที่จะเพิ่มชุดคำสั่ง DIM (x,y) เข้าไปในโปรแกรมเพื่อประโยชน์ในการเขียนโปรแกรมได้สะดวกยิ่งขึ้น

3. เนื่องจากเอ็นวีแรมไม่ปลอดภัยในการเก็บข้อมูล ผู้วิจัยได้เสนอแนวความคิดที่จะเอาแฟลชมาเก็บโปรแกรมแทนเอ็นวีแรม เนื่องจากแฟลชมีราคาถูกและสามารถเก็บรักษาโปรแกรมได้นานและปลอดภัย ดัดแปลง

4. จากปัญหาข้อ 4 ที่ได้กล่าวมา เมื่อใช้คำสั่ง BRKP และคำสั่ง CONT นั้นจะต้องพิมพ์คำสั่งลงไปทุกครั้งเพื่อที่จะกำหนดจุดหยุดและดำเนินการโปรแกรมต่อไปทุกครั้ง ฉะนั้นแนวความคิดในการแก้ปัญหาคือการเขียนโปรแกรมกำหนดปุ่มแต่ละปุ่มให้เป็นปุ่ม BRKP และปุ่ม CONT เช่นปุ่ม BRKP เป็น F1 และ CONT เป็น F2 โดยการแก้ไขโปรแกรมมอนิเตอร์ในแอลซีดีเทอร์มินัล หลักการก็คือ กด F1 ให้ส่งอักษร BRKP ออกไป อีดีแอลภาษาเบสิกจะทำการแปลเอง

5. ในโลกปัจจุบัน เทคโนโลยีทางด้านไมโครคอนโทรลเลอร์ได้พัฒนาไปอย่างรวดเร็ว ฉะนั้นผู้วิจัยจึงได้สร้างมาตรฐานวัดอ้างอิงบริษัทอีทีทีจำกัด เพื่อที่จะสามารถนำแผงฝึกทดลองที่วางจำหน่ายตลาดใหม่ ๆ มาต่อทดลองกับแผงอีดีแอลเบสิกได้ ดังนั้นเมื่อมีการทดลองใหม่เกิดขึ้นผู้ศึกษาก็สามารถจัดหามาทำการวิจัยได้

รายการอ้างอิง

1. กฤษดา วิศวธีรานนท์. เรียน/เล่น/ใช้ไอซีดิจิตอล. กรุงเทพมหานคร : บริษัทซีเอ็ดยูเคชั่น จำกัด, 2532.
2. ก้าวสู่โลกคอมพิวเตอร์ด้วยชิงเกอร์บอร์ด ET-3.5 เครื่องมืออันทรงประสิทธิภาพ. เซมิคอนดัคเตอร์อิเล็กทรอนิกส์. ฉบับที่ 107 (มิถุนายน 2534): 16-19.
3. คู่มือซัพพอร์ตและหน่วยความจำ. ครั้งที่ 1, กรุงเทพมหานคร, : ซีเอ็ดยูเคชั่น จำกัด (2529)
4. จลีพร โกลากุล. หลักภาษาเบสิก. กรุงเทพมหานคร, ศรีสมร จำกัด, 2538 : 27-52.
5. จิติ หนูแก้ว. ไมโครโปรเซสเซสเซอร์และการออกแบบเบื้องต้น. กรุงเทพมหานคร, บริษัทซีเอ็ดยูเคชั่น จำกัด, 2521.
6. ทวีชัย เจริญเศรษฐศิลป์ วิทยานิพนธ์เรื่อง. การออกแบบและพัฒนาชุดฝึกทดลองไมโครคอมพิวเตอร์ โดยใช้ไอบีเอ็มพีซี. กรุงเทพมหานคร, จุฬาลงกรณ์มหาวิทยาลัย, 2539 : 10-28.
7. บริษัท แอนาดิจิท จำกัด. เซมิคอนดัคเตอร์ อิเล็กทรอนิกส์. ฉบับที่ 92 (พฤษภาคม – มิถุนายน 2532): 112.
8. ประพัฒน์ อุทโยภาส. เรียน Applesoft Basic ด้วยตนเอง. กรุงเทพมหานคร, บริษัทซีเอ็ดยูเคชั่น จำกัด, 2538 : 5-28.
9. Advanced Electronic System. AES-51 System User 's Manual . Bangalore, India.
10. ATMEL Inc. Data sheet of 89C52 Microcontroller. http://www.atmel.com
11. Carl E. Wick. A four-Chip Microcomputer for undergraduate Engineering Course. IEEE, Conference, 1993.
12. Craig Precock. Interfacing the PC Keyboard. http://www.geocities.com/silicon.htm.
13. Hans-Peter Messmer. The Indispensable PC Hardware Book. Addison – Wesley Publishing Company, USA, Seccond Edition (1995) : 919-945.
14. J. Kemeny and T. Kurtz. BASIC Programming Language. USA, Holden-Day, Inc.,1973.
15. Jan Axelson. The Microcontroller Idea Book (Circuit, Program, & Applications featuring the 8052- BASIC Microcontroller). USA, Internation Thomson Publishing (ITP), 1997 : 10-58
16. King Instrument Electronic Co., Ltd. Computer Interface Control lab CIC-100 User Guide. Taiwan: King Instrument Electronic Co., Ltd., 1994.
17. Lawrence,P.,D.,and Konrad Mauch. Real Time Microcontroller System Design. Singapore, McGraw – Hill, 1988 : 156-189.
18. Remesh, B.,S., Jamadagni, H, S, and Macro Oligiati. Microprocessor Laboratory Primer . Bangalore, India: CEDT (1987).
19. Richard S. Sanolige. Modern Digital Design. McGraw-Hill Publishing Company, (1990) : 15-150.

20. Sack and J. Meadow. Entering BASIC. USA, Science Research, 1973.

21. Santa Clara, Calif. MCS BASIC-52 User 's Manual. Intel Corp., (1981) . 20-150.

22. Thomas W, Schuitz. C AND The 8051 Hardware,Modular Programming,and Multitasking. Prentice Hall PTR Upper Saddle Rive, New Jersey 07458 (1998) : 281- 309.

23. Williamson, T. Design Microcontroller System for Electically Noise Enviroments. Intel Application Note AP-125, (February, 1982) : 75-106.

ภาคผนวก

ภาคผนวก ก

คู่มือการใช้งานชุดฝึกทดลองไมโครคอนโทรลเลอร์

ลักษณะทั่วไปของชุดฝึกทดลองไมโครคอนโทรลเลอร์

สามารถแบ่งออกได้เป็น 3 ส่วนใหญ่ ๆ ดังนี้

1. แผงแอลซีดีเทอร์มินัล
2. แผงอีดีแอลเบสิก
3. แผงฝึกทดลอง

รายละเอียดของแต่ละส่วนจะอธิบายต่อไปนี้

1. แผงแอลซีดีเทอร์มินัล

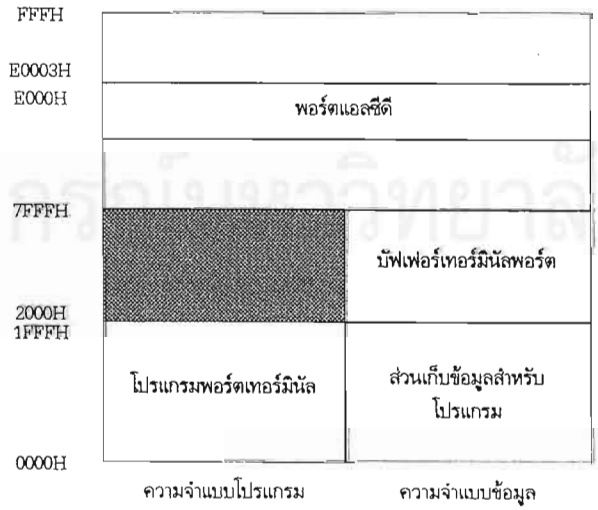
ลักษณะโดยทั่วไปของแผงแอลซีดีเทอร์มินัล

แผงแอลซีดีเทอร์มินัลเป็นแผงไมโครคอนโทรลเลอร์ในตระกูล MCS-51 ออกแบบมาเพื่อนำมาใช้แทนไมโครคอมพิวเตอร์ เนื่องจากมีโปรแกรมบรรณาธิกรณ์บรรจุในซีพียู รับส่งข้อมูลทาง RS-232 อินพุตต่อเข้ากับแผงแป้นอักขระของไมโครคอมพิวเตอร์ (PC Keyboard) และต่อแสดงผลที่จอแอลซีดี

คุณสมบัติของแผงแอลซีดีเทอร์มินัล

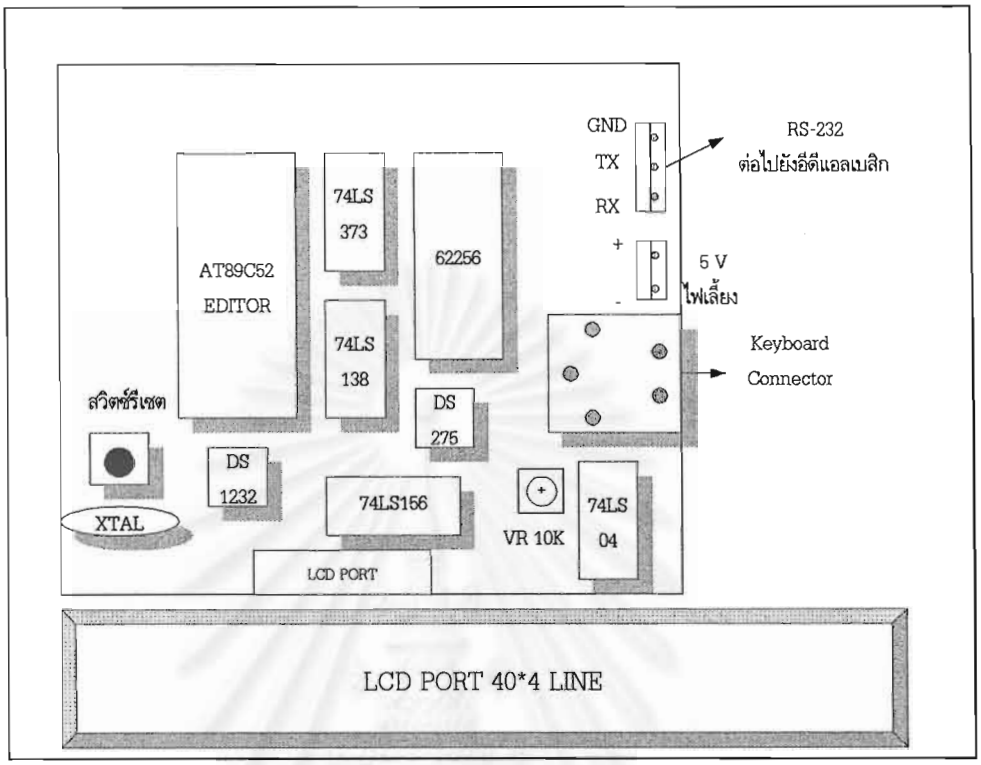
- ซีพียู AT89C52 ดำเนินการ (RUN) ที่ความถี่ 11.059 MHz.
- แสดงผลโปรแกรมสูงสุดได้ 614 บรรทัด (24 KB)
- แอลซีดีขนาด 40 ตัวอักษร 4 บรรทัด (16 ขา)
- พอร์ต RS-232 1 พอร์ต (DS 275)
- อินพุตต่อกับแผงแป้นอักขระคอมพิวเตอร์ (PC Keyboard)
- วงจรการตั้งค่าใหม่ (Reset) (DS 1302)
- แผงขนาด 5.5 cm * 8 cm.

แผนที่หน่วยความจำ



รูปที่ ก.1 แผนผังหน่วยความจำของแอลซีดีเทอร์มินัล

รายละเอียดตำแหน่งอุปกรณ์บนแผงแอลซีดีเทอร์มินัล



รูปที่ ก.2 ตำแหน่งอุปกรณ์บนแผงแอลซีดีเทอร์มินัล

การใช้งานแอลซีดีเทอร์มินัล

ต่อกับแผงแอลซีดีเข้ากับแอลซีดีเทอร์มินัล แล้วต่อไฟเลี้ยงเข้ากับคอนเน็กเตอร์หน้าจอแอลซีดีจะปรากฏคำว่า

```
**********************************************************
*                  BASIC  DEBUGGER                       *
*    BY  AMONPONG  CHUMSAI  NA  AYUTTAYA                 *
**********************************************************
```

การทดสอบต่อแผงแป้นอักขระคอมพิวเตอร์ แล้วนำสายต่อมาต่อขาส่งและขารับเข้าด้วยกันที่พอร์ต RS-232 แล้วจ่ายไฟเข้าแผง ทดสอบกดปุ่มใดปุ่มหนึ่งแล้วดูว่าตัวอักษรที่กดตรงตามอักษรที่แสดงผลหรือไม่

1. แผงอีดีแอลเบสิก

ลักษณะโดยทั่วไปของแผงอีดีแอลเบสิก

แผงอีดีแอลเบสิก เป็นแผงที่ใช้ไมโครคอนโทรลเลอร์ AT89C52 ของ ATMEL บรรจุโปรแกรมอีดีแอลเบสิกขนาด 8 KB ใช้งานร่วมกับไมโครคอมพิวเตอร์ หรือแอลซีดีเทอร์มินัลได้ ด้วยอีดีแอลเบสิกนี้เองทำให้ผู้ใช้

สามารถเรียนรู้และพัฒนาโปรแกรมได้อย่างรวดเร็วและมีประสิทธิผล ตัวแผงมีขนาดเล็กและใช้กระแสไฟฟ้าน้อย ราคาถูกจัดหาอุปกรณ์ในประเทศได้ง่าย

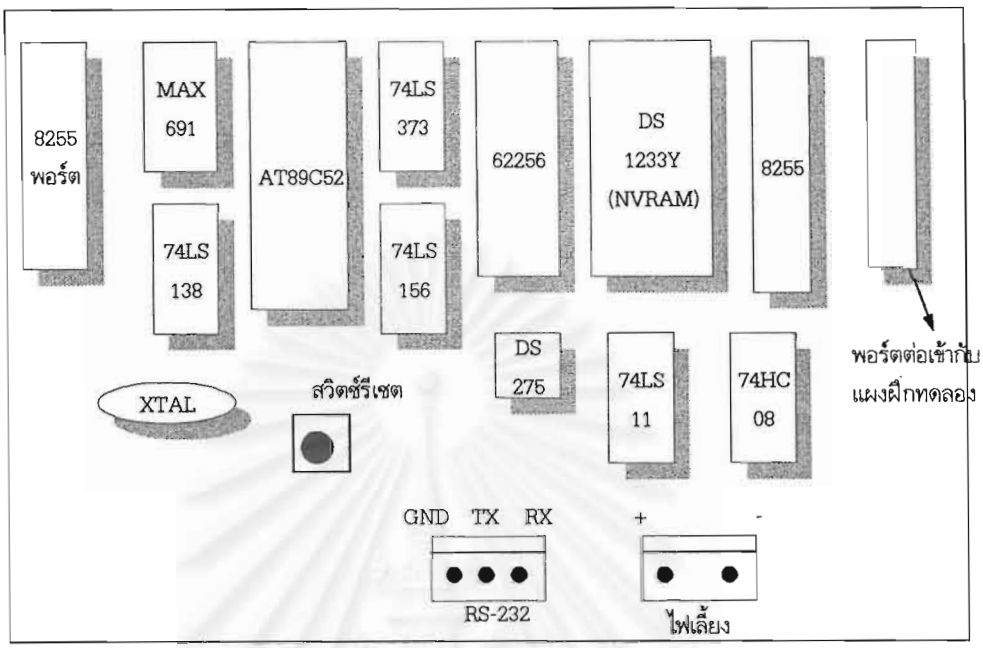
คุณสมบัติของแผงอีดีแอลเบสิก

- ซีพียู AT89C52 ดำเนินการ (RUN) ที่ความถี่ 11.059 MHz.
- 32 KB แรมมอนิเตอร์
- 32 KB เอ็นวีแรม
- 40 ขา อีทีทีบัส
- 24 บิตไอโอ 8255 พอร์ต
- วงจรการตั้งค่าใหม่ (Reset) บนแผง (MAX 691)
- พอร์ตอนุกรม RS-232 (DS 275)
- 8 KB โปรแกรมมอนิเตอร์ภายในซีพียู
- แผ่นวงจรขนาด 11 ซม. * 8 ซม.

แผนที่หน่วยความจำของแผงอีดีแอลเบสิก

รูปที่ ก.3 แผนที่หน่วยความจำของแผงอีดีแอลเบสิก

รายละเอียดตำแหน่งอุปกรณ์บนแผงอีดีแอลเบสิก



รูปที่ ก.4 ตำแหน่งอุปกรณ์บนแผงอีดีแอลเบสิก

การใช้งานอีดีแอลเบสิก

ก่อนเปิดเครื่อง

ต่ออีดีแอลเบสิกเข้าแหล่งจ่ายกระแสไฟตรง ตรวจสอบขั้วและระดับแรงดันให้แน่ใจโดยใช้มิเตอร์ ระดับแรงดันไฟตรงจากวงจรรักษาระดับแรงดันไม่ควรสูงกว่า 5.5 โวลต์ และต่ำกว่า 6.5 โวลต์

เริ่มเปิดเครื่อง

ต่ออีดีแอลเบสิกเข้ากับแอลซีดีเทอร์มินัล หลังจากนั้นเมื่อทุกสิ่งทุกอย่างเรียบร้อยแล้วจึงจ่ายกระแสไฟฟ้าเข้าเครื่อง จากนั้นทดลองการกดปุ่ม Space bar ที่แผงแป้นอักขระ อีดีแอลเบสิกจะแสดงข้อความบนแอลซีดีดังนี้

* MCS-51 BASIC52 BY TEW *

READY

>

ทดลองตรวจสอบค่า XTAL ในโปรแกรมค่าที่ได้ต้องเท่ากับค่าผลึกในวงจรคือ 11.0592 MHz.

> PRINT XTAL

> 11059200

ตรวจสอบค่าหน่วยความจำในแรมมอนิเตอร์ ค่าที่ได้ต้องเท่ากับค่าในหน่วยความจำแรมคือ 32 KB.

> PRINT MTOP

> 32512

ในกรณีที่ใช้อีดีแอลเบสิกกับไมโครคอมพิวเตอร์ให้พิมพ์ดังต่อไปนี้

C:\ PROCOMM\ PCP.BAT

โปรแกรมจะปรากฏขึ้น มีอักษรที่หน้าจอคอมพิวเตอร์ดังนี้

PROCOMM PLUS

จากนั้นกด ALT-P เพื่อตั้งค่าการสื่อสารข้อมูล ตัวอย่างเช่น

อัตราส่งข้อมูล (BAUD RATE) = 1200 bits/Sec

บิตหยุด (Parity bit) = 1 บิต

ข้อมูล = 8 บิต

เมื่อตั้งค่าเรียบร้อยแล้วกด ALT-S เพื่อบันทึกค่าที่เราตั้งค่าลงไป ทดสอบการจ่ายกระแสไฟฟ้าเข้าอีดี-แอลเบสิกแล้วกดปุ่ม Space bar แผงแป้นอักขระจะปรากฏอักษรดังต่อไปนี้

* MCS-51 BASIC52 BY TEW *

ทดสอบการทำงานในลักษณะเดียวกับข้างต้น

3. แผงฝึกทดลอง

ลักษณะโดยทั่วไปของแผงฝึกทดลอง

แผงฝึกทดลองที่สร้างขึ้นถูกออกแบบมาเพื่อให้ใช้กับงานทดลอง โดยในแผงชุดฝึกทดลองจะรวบรวมแต่ละการทดลองไว้ในแผงเดียวกัน สะดวก ใช้งานง่าย อุปกรณ์ที่นำมาประกอบมีราคาถูก จัดหาง่าย เหมาะสำหรับผู้-เริ่มต้น

คุณสมบัติของชุดฝึกทดลอง

- ไฟเลี้ยงกระแสตรง 15 โวลต์ พร้อมวงจรรักษาระดับไฟตรงในตัว
- ใช้มาตรฐานอีทีทีบัส 40 ขา
- วงจรถอดรหัส อินพุตดิปสวิตช์
- วงจรแอลอีดี 8 ดวง, แอลอีดี 7 ส่วน
- แผงแป้นอักขระขนาด 4 หลัก 3 แถว
- วงจรขับแอลอีดีแสดงผลแบบ 8 หลัก 8 แถว
- ตัวควบคุมมอเตอร์กระแสตรง และมอเตอร์แบบสเต็ป
- มีวงจรแอลซีดีในตัว พร้อมแอลอีดีบนแผง
- ตัวแปลงผันสัญญาณแอนะล็อกเป็นดิจิทัล และดิจิทัลเป็นแอนะล็อก
- ขนาดแผ่นวงจรพิมพ์ กว้าง 8 นิ้ว ยาว 13 นิ้ว

ตารางที่ ก.1 ตารางแสดงไอโอของแผงฝึก

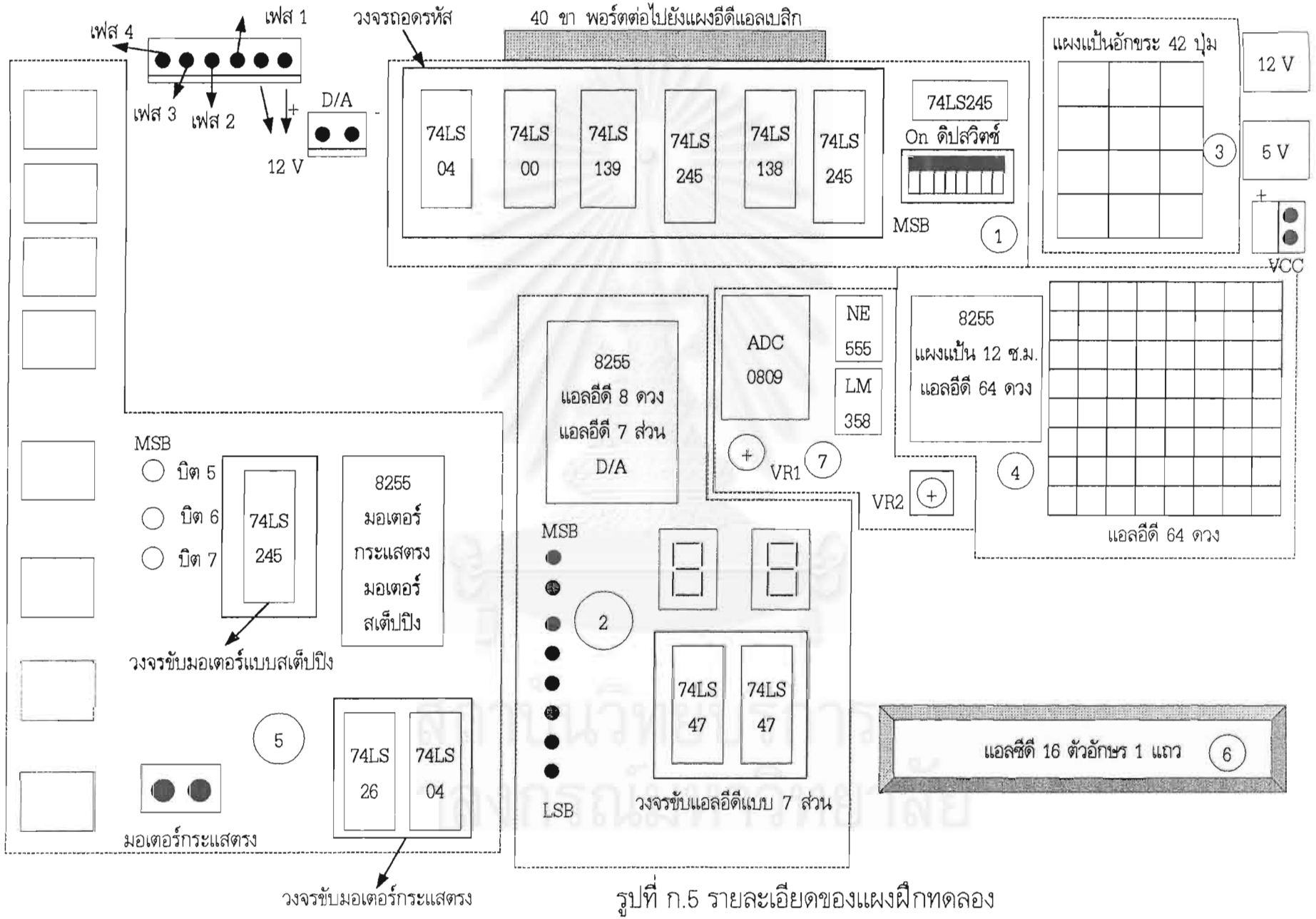
ตำแหน่งของพอร์ต	ตำแหน่งของพอร์ต
CS-IN8 ของ DIP SWITCH	0A0H
OA-ATD ใช้อ่าน ATOD	0A1H
CS-ATD ของ ATOD	0A4H-0ABH
CS-LCD ของ LCD	0ACH-0AFH
8255#1 ของ DISPLAY 8*8 และ KEY BOARD 3*4	0B0H-0B3H
8255#2 ของ MOTOR และ STEPPING	0B6H-0B7H
8255#3 DTOA, SEGMENT และ LED	0B8H-0BBH

แต่ละหมายเลขจะบอกตำแหน่งของแต่ละการทดลอง

1. วงจรถอดรหัส อินพุตดิปสวิตช์
2. วงจรแอลอีดี 8 ดวง, แอลอีดี 7 ส่วน
3. แผงแป้นอักขระขนาด 4 หลัก 3 แถว
4. วงจรขับแอลอีดีแสดงผลแบบ 8 หลัก 8 แถว
5. ตัวควบคุมมอเตอร์กระแสตรง และมอเตอร์แบบสเต็ป
6. มีวงจรแอลซีดีในตัว พร้อมแอลอีดีบนแผง
7. ตัวแปลงผันสัญญาณแอนะล็อกเป็นดิจิทัล และดิจิทัลเป็นแอนะล็อก

VR1 คือ ตัวต้านทานปรับโวลต์อ้างอิงใน A/D

VR2 คือ ตัวต้านทานปรับแรงดัน ช่อง 1 (Channel 1) ของ A/D



รูปที่ ก.5 รายละเอียดของแผงฝึกทดลอง

ภาคผนวก ข

ใบงานการทดลอง

โครงสร้างของบทเรียน

ชุดฝึกทดลองในงานวิทยานิพนธ์นี้ จะประกอบด้วยการทดลองรวมทั้งสิ้น 7 การทดลองโดยมีรายละเอียดของแต่ละการทดลอง ดังตารางข้างล่างนี้

ตารางที่ ข.1 โครงสร้างการทดลอง

ลำดับ	เรื่อง	รายละเอียด
1	พื้นฐานอินพุตและตัวถอดรหัสตำแหน่ง	- ทดสอบดำเนินตามโปรแกรมตัวอย่าง - ทดสอบรับข้อมูลจากดิปสวิตช์
2	8255 และ พื้นฐานเอาต์พุต	- เขียนโปรแกรมตัวอย่าง - ทดสอบส่งข้อมูลออก 8255 ไปยังแอลอีดี
3	แผงแป้นอักขระ	- เรียนรู้และทดสอบโปรแกรมตัวอย่าง - ทดสอบดัดแปลงโปรแกรม
4	การแสดงผลแอลดีอี 8 แถวและ 8 หลัก	- เรียนรู้การทำงานและทดสอบโปรแกรม - ดัดแปลงข้อมูลตัวอักษร
5	มอเตอร์กระแสตรงและสเต็ปปิงมอเตอร์	- เรียนรู้การทำงานและควบคุมมอเตอร์กระแสตรง - ดัดแปลงโปรแกรมสำหรับขับมอเตอร์ในแบบต่างๆ - เรียนรู้หลักการและการใช้งานสเต็ปปิงมอเตอร์ - ทดสอบควบคุมสเต็ปปิงมอเตอร์ในแบบเฟสต่างๆ
6	การแสดงผลบนจอแอลซีดี (LCD)	- เรียนรู้หลักการควบคุมแอลอีดีจากโปรแกรมตัวอย่าง - ดัดแปลงแสดงผลตัวอักษรต่างๆ
7	การแปลงผันสัญญาณดิจิทัลเป็นแอนะล็อกและแอนะล็อกเป็นดิจิทัล	- ทดสอบโปรแกรมตัวอย่างของ D/A - ทดสอบโปรแกรมตัวอย่างของ A/D ขนาด 8 บิต

การทดลองที่ 1 : พื้นฐานอินพุตและตัวถอดรหัสตำแหน่ง

วัตถุประสงค์

1) เพื่อให้เข้าใจการจัดสรรพื้นที่ใช้งานของอุปกรณ์อินพุตเอาต์พุต
2) เพื่อให้เข้าใจการออกแบบวงจรตัวถอดรหัส
3) เพื่อให้เข้าใจการอ่านสัญญาณจากภายนอกมาใช้งาน

ทฤษฎี

ระบบไมโครคอมพิวเตอร์ฮาร์ดแวร์ ประกอบด้วยส่วนสำคัญ 3 ส่วน คือ

1. หน่วยประมวลผลกลาง (CPU = Central Processing Unit) มีหน้าที่ประมวลผลตามคำสั่ง
2. หน่วยความจำ (Memory Map) เป็นหน่วยเก็บรหัสคำสั่งและข้อมูล
3. หน่วยนำข้อมูลเข้า/ออก (Input/Output Unit) ทำหน้าที่ติดต่อกับอุปกรณ์ หรือสัญญาณภายนอก เช่น แผงแป้นอักขระและจอภาพ เป็นต้น

เนื่องจากในระบบคอมพิวเตอร์จะใช้ข้อมูล (Data Bus) และสายตำแหน่งหน่วยความจำ (Address Bus) ร่วมกันทั้งระบบ ฉะนั้นการที่จะติดต่อกับหน่วยความจำหรืออุปกรณ์อินพุตเอาต์พุตตัวใดต้องมีการเลือก ซึ่งเป็นหน้าที่ของวงจรถอดรหัส

วงจรถอดรหัส (Decoder)

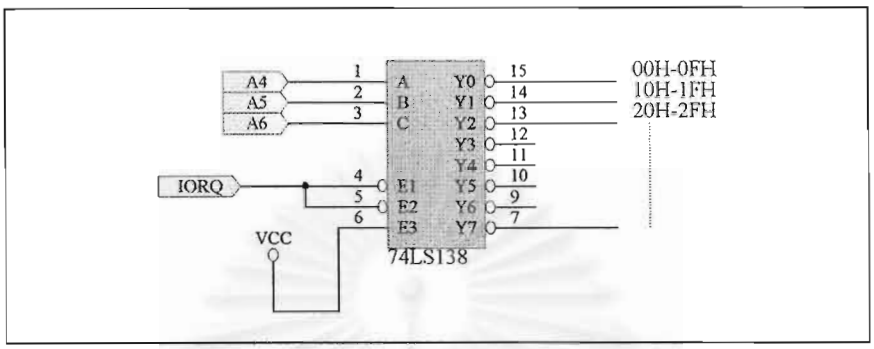
หลักการถอดรหัสไม่ว่าจะเป็นหน่วยความจำหรือหน่วยอินพุต/เอาต์พุต จะใช้หลักการเบื้องต้นคล้ายกันกับซีพียูเบอร์อื่น เนื่องจากซีพียูเป็นตัวจ่ายสัญญาณตำแหน่งที่อยู่ (Address) ไปยังหน่วยความจำหรืออุปกรณ์อินพุต/เอาต์พุต ก่อนที่จะทำการอ่านหรือเขียนข้อมูล เราจึงนำเอาตำแหน่งมาเป็นตัวถอดรหัสเพื่อเลือกตำแหน่งของหน่วยความจำ หรือตำแหน่งของพอร์ตนั้น ๆ หมายความว่าจะมีอุปกรณ์เพียงตัวเดียวเท่านั้นที่ซีพียูทำการติดต่อด้วย ตัวอื่น ๆ จะอยู่ในสถานะลอย (Floating)

การกำหนดจุดของตัวถอดรหัส

โดยเริ่มแรกเราดูว่า เอาต์พุต จะต้องมีจำนวนกี่จุด จากนั้นไอซีตัวถอดรหัสมาใช้โดยดูจำนวนขาอินพุตว่าต้องการกี่เส้น ก็จะทำให้ทราบจำนวนเอาต์พุตได้ โดยจะเป็น 2 ยกกำลังจำนวนของสายอินพุต เช่น อินพุต = 3 จะได้ = 2^3 คือ 8 เอาต์พุต จากนั้นจะให้จุดตัวถอดรหัสแต่ละชุดสามารถอ้างตำแหน่งใช้งานได้เป็น จำนวนเท่าไรขึ้นอยู่กับสายอินพุต ต่ำสุดว่าเลือกใช้สายเส้นใด โดยวิธีการกำหนด BIT ออกเป็นน้ำหนักของเลขยกกำลัง 2 ดังนี้

32768	16384	4096	-----	128	64	32	16	8	4	2	1
A15	A14	A13	-----	A7	A6	A5	A4	A3	A2	A1	A0

เช่น เราต้องการกำหนดจุดถอดรหัสอุปกรณ์ไอโอแต่ละจุดให้ 1 จุด สามารถอ้างตำแหน่งครอบคลุมได้ 16 ตำแหน่ง โดยเลือกใช้ไอซี 74LS138 จะได้ดังรูปที่ ข.1



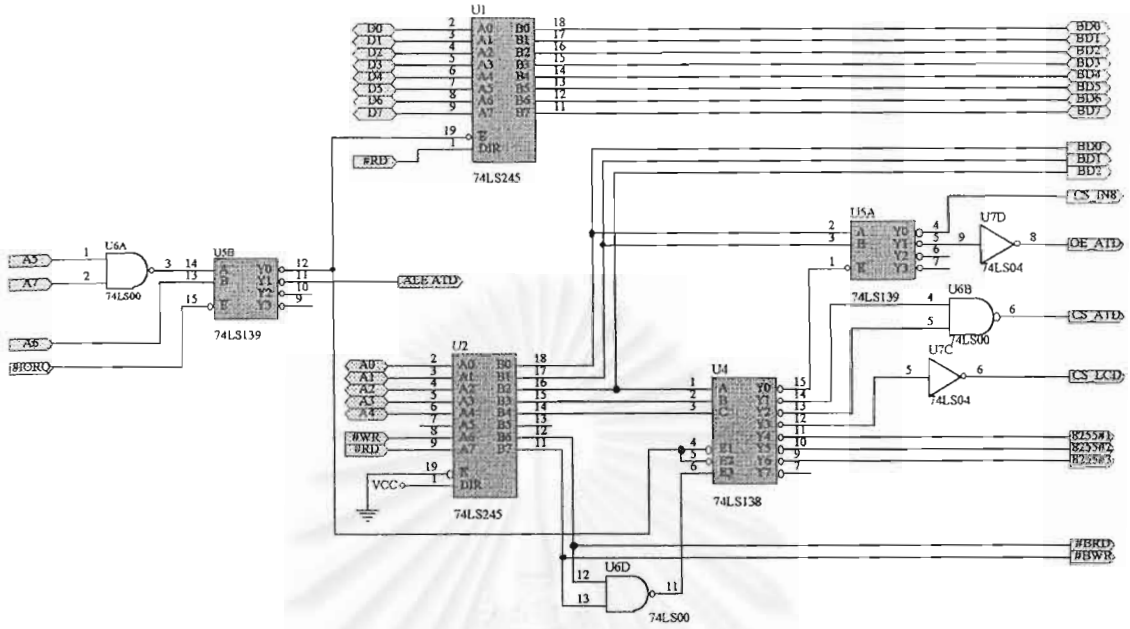
รูปที่ ข.1 วงจรถอดรหัสตำแหน่งอินพุต/เอาต์พุต โดยใช้ 74LS138

ในกรณีที่เราจะดูว่าเอาต์พุต Y2 จะเริ่มทำงานจากเลขที่อยู่สุดท้ายที่เท่าไร ทำได้โดยการกำหนดอินพุต BIT ของ 74LS138 ที่ทำให้ Y2 เปลี่ยนจากระดับตรรกะ 1 เป็นตรรกะ 0 ในที่นี้ A6=0, A5=1, A4=0 โดยบิตที่ไม่สนใจ (Don't care) จะกำหนดให้เป็น 0 หรือ 1 ก็ได้ แต่ถ้าให้ง่าย BIT ที่สูงกว่า สายอินพุตสูงสุดของตัวถอดรหัส กำหนดให้เป็น 0 (A7 - A15) ส่วนบิตที่ต่ำกว่าสายอินพุต ต่ำสุดที่เข้าไอซีตัวถอดรหัส (A0 - A3) โดยกำหนดให้เป็น 2 สภาวะ คือ ครั้งแรกให้เป็น 0 ซึ่งจะหมายถึงขาเอาต์พุต จะเริ่มทำงานที่ตำแหน่งที่อยู่นี้ นั่นก็คือ 20H และจากนั้นจะให้บิต (A0 - A3) เป็น 1 หมดนั่นก็คือตำแหน่งสุดท้ายที่ขาเอาต์พุต นี้ยังทำงานอยู่ในกรณีนี้คือ 2FH

วิธีการดังกล่าว เป็นวิธีแบบง่าย ๆ ที่จริงแล้วสายอินพุตที่เราไม่ได้ต่อจะมีผลต่อจุดตัวถอดรหัส ด้วยทำให้การอ้างตำแหน่งต่าง ๆ เกิดการทับซ้อนกันได้ ซึ่งจะทำให้ตำแหน่งใช้งานสูญเสียไปได้ จากตัวอย่างถ้าเราอ้างตำแหน่งที่ A7 เป็น 1 และ A6, A5 และ A4 ยังเป็น 010 อยู่นั่นจะหมายถึง

ไอโอตำแหน่ง 20H - 2FH ; Y2 ของ 74LS138 ทำงาน
A0H - AFH ; Y2 ของ 74LS138 ก็ทำงาน จึงกลายเป็นตำแหน่งเดียวกัน

กรณีที่เป็นหน่วยความจำตรงขา G2A และ G2B เราก็อาจนำขาควบคุมหน่วยความจำมาต่อแทนหรือในกรณีของการทับซ้อนกัน ก็อาจจะนำจุดที่ถอดรหัส ในส่วนของไบต์สูง ที่ไม่ได้ต่อนำมาควบคุมที่ขานี้ก็ได้



รูปที่ ข.2 ส่วนถอดรหัสและบัฟเฟอร์ของแผง I/O

วงจรขับกระแส

เนื่องจากแผง I/O มีอุปกรณ์จำนวนมากและการต่อระบบจะมีการดึงสายสัญญาณข้อมูล, ตำแหน่ง และสัญญาณควบคุม จากซีพียู มาเข้ากับอุปกรณ์ I/O โดยตรง ซึ่งจะเป็นผลทำให้ ซีพียูต้องรับภาระบรรจุมากเกินไป ซึ่งจะทำให้ซีพียูไม่สามารถทำงานตามที่กำหนดได้ จึงได้มีการต่อบัฟเฟอร์ให้กับระบบโดยไอซี 74LS245 จะเป็นบัฟเฟอร์ข้อมูลซึ่งสัญญาณ จะส่งผ่านก็ต่อเมื่อมีการติดต่อกับอุปกรณ์ I/O ตั้งแต่ตำแหน่ง A0H ถึง 0BFH ถ้าไม่อยู่ในช่วงนี้ก็จะทำให้สายข้อมูลที่เข้าแผง I/O เป็นอิมพีแดนซ์สูงเนื่องจากขา G ของไอซี 74LS245 เป็น HIGH ส่วนขา 1 จะเป็นตัวกำหนดทิศทางของข้อมูลโดยต่อขา RD ไว้ เมื่อมีการอ่านสัญญาณจากทางด้าน B จะได้มาทางด้าน A ไปทางด้าน B และขา G ของไอซี 74LS245 นี้ เราจะต่อกราวด์ เพื่อให้ทำงานตลอดไม่ได้ เพราะบนแผงควบคุมไมโครคอนโทรลเลอร์ไม่มีบัฟเฟอร์ ดังนั้นเมื่อมีการติดต่อกับหน่วยความจำ หรือ I/O จะทำให้มีข้อมูลจากซีพียู หรืออุปกรณ์ที่ไม่ได้ติดต่อด้วยเข้ามาชนกันทำให้ระบบทำงานไม่ได้ นั่นก็คือ บัฟเฟอร์ข้อมูล (U1) เป็นพอร์ตแบบ 2 ทิศทาง ส่วนเลขที่อยู่ของบัฟเฟอร์นั้นกำหนดให้ทำงานตลอด เพราะทิศทางเป็นการเขียนจากซีพียูไปอย่างเดียว (U2)

ส่วนถอดรหัส (Decoder) ของแผงฝึก

U5B จะกำหนดช่วงพื้นที่ใช้งานของอุปกรณ์ไอโอจากตำแหน่ง A0H - BFH โดยดูจากขาเอาท์พุต ที่เข้ามาและเอาท์พุตที่ใช้ ซึ่งเราใช้ Y0 เป็นตัวเปิดการติดต่อของแผงไอโอ ดังนั้นอินพุตจะต้องเป็น 0 เท่านั้น ดังนั้นตำแหน่งจะเป็นดังนี้

A7	A6	A5	A4	A3	A2	A1	A0
1	0	1	X	X	X	X	X

คือ A7 และ A5 ต้องเป็น 1 ทั้งคู่ ขา 14 ของ U5B ถึงจะเป็น 0 เพราะผ่าน NAND และสายแอดเดรสที่ไม่ได้ต่อเข้าที่ตัวถอดรหัสของ 74LS139 ก็จะเป็นไม่สนใจ (Don't care) เริ่มแรกให้เป็น 0 หมด จะได้ 0A0H จากนั้นส่วนที่ไม่สนใจ (Don ' t care) ก็จะให้เป็น 1 หมด จะได้ 0BFH นี่คือจุดถอดหัสเปิดแผงการทำงานของชุดฝึก เปรียบเสมือนสวิตช์ปิดหรือเปิดแผงฝึก และในส่วนของแผงฝึกจะประกอบไปด้วยอุปกรณ์ย่อยอีกจำนวนมาก เราจึงจำเป็นตองแบ่งสรรตำแหน่งที่ใช้เปิดการทำงานของแผงฝึกออกเป็นส่วนย่อย ๆ อีก โดยจะกำหนดการแบ่งสรรเป็นเท่าใดนั้น เราต้องดูอุปกรณ์ที่ติดต่อด้วย ซึ่งแยกได้เป็น

DIP SWITCH	1 จุด Decoder
8255	1 จุด Decoder
LCD	1 จุด Decoder
ATOD	1 จุด Decoder

ซึ่งดูจากจุดนี้จะได้ 8 เอาต์พุต โดยเราเลือก 74LS138 มาใช้ก็น่าจะได้ แต่เราจะต้องดูอีกว่าตำแหน่งที่ได้มาจาก 74LS139 สามารถอ้างตำแหน่งทั้งหมดได้เท่าไหร่

BFH - A0H = 32 ตำแหน่ง

เมื่อนำมาหาร 8 เอาต์พุต ก็จะได้จุดถอดรหัสของ 74LS138 ถอดรหัสทีละ 4 ตำแหน่ง ซึ่งก็น่าจะใช้ได้พอดี โดยเรามาดูที่อุปกรณ์แต่ละชนิดอีกครั้งว่าแต่ละตัว ต้องอ้างตำแหน่งได้เท่าไรบ้าง 8255, LCD และ 8253 จะมี Address A0 และ A1 ต่อใช้งานที่ตัวด้วย 2^2 = 4 ตำแหน่งนั้นก็คือ ทั้ง 3 ตัวที่กล่าวมาจะต้องติดต่อตำแหน่งต่าง ๆ ได้ 4 ตำแหน่ง ซึ่งตรงกับ 74LS138 ที่เราจะถอดรหัสพอดี แต่จะมีปัญหาที่เอทูดีใน 2 จุด คือ 1 ถอดรหัสใช้อ่าน ATOD ส่วนอีก 1 ถอดรหัสที่ใช้เลือกช่องสัญญาณ ซึ่งมีทั้งหมด 8 ช่องสัญญาณ หมายถึงต้องอ้างได้ 8 ตำแหน่ง แต่ตำแหน่งที่เราถอดรหัสได้ต่อ 1 จุด เพียง 4 ตำแหน่ง จึงต้องรวมจุดถอดรหัส 2 จุดเข้าด้วยกัน ซึ่งทำให้ขาดไป 1 จุด ดังนั้นเราจึงต้องย่อย 1 จุดที่ถอดรหัสของ 74LS138 เอาต์พุตใดเอาต์พุตหนึ่งให้ละเอียดขึ้นเมื่อย่อยแล้วอุปกรณ์ไอโอจะต้องใช้ได้ซึ่งก็จะมีดิปสวิตช์ และถอดรหัสที่ใช้อ่านเอทูดี ซึ่งตำแหน่งถอดรหัส 1 จุดอ้างได้เพียงตำแหน่งเดียวก็สามารถทำงานได้แล้ว ดังนั้นแอดเดรสต่ำสุดที่จะเข้า (U4) จะเริ่มที่ A2 จึงจะได้จุด ถอดรหัส 1 จุดได้ 4 ตำแหน่ง และเราจะนำเอาต์พุต Y0 มาย่อยลงโดยใช้ A0 และ A1 ให้กับ U5A เพื่อใช้เลือกดิป-สวิตช์และอ่านเอทูดี นั่นก็หมายความว่า U5B จะมาเปิดการทำงานของ U4 ที่ขา G2A และ G2B โดย U4 จะถอดรหัสจากสายอินพุต BA2-BA4 และขา 6(G1) ของ U4 จะถูกดำเนินการแนนด์ (NAND) ไว้กับ RD และ WR เพื่อให้สัญญาณถอดรหัสออกมาในเวลาที่เหมาะสม

ตารางที่ ข.2 สรุปตำแหน่ง I/O Map ของชุดฝึก

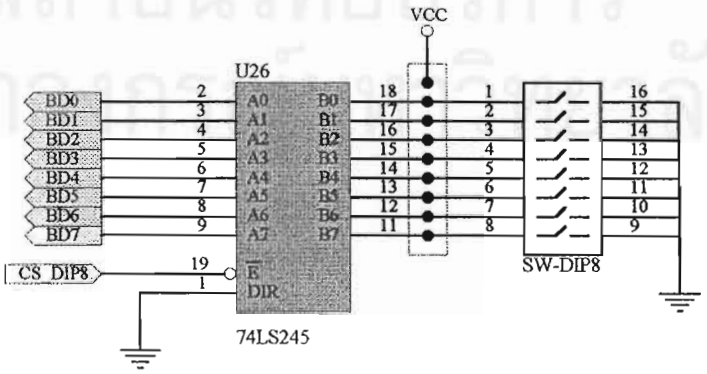
อุปกรณ์ไอโอ	ตำแหน่งของพอร์ต
CS-IN8 ของ DIP SWITCH	0A0H
OA-ATD ใช้อ่าน ATOD	0A1H
CS-ATD ของ ATOD	0A4H-0ABH
CS-LCD ของ LCD	0ACH-0AFH
8255#1 ของ DISPLAY 8*8 และ KEY BOARD 3*4	0B0H-0B3H
8255#2 ของ MOTOR และ STEPPING	0B6H-0B7H
8255#3 DTOA, SEGMENT และ LED	0B8H-0BBH

อุปกรณ์การทดลอง

1. แผงควบคุม
2. แผงชุดฝึกทดลอง
3. แหล่งจ่ายกระแสไฟฟ้า
4. ลอจิกโพรบ

การทดลอง (INPUT)

เมื่อซีพียูพบคำสั่งให้อ่านไอโอ พอร์ตซีพียู จะให้สัญญาณระบุตำแหน่งออกไปทาง ทางเดินข้อมูล (bus) โดยมีตำแหน่งของพอร์ตตามที่กำหนดมาในโปรแกรม ในขณะนี้เองส่วนถอดรหัสก็จะทำงาน และถ้าตำแหน่งของพอร์ตตรงกับตัวถอดรหัสก็จะได้เอาต์พุตจากวงจรถอดรหัส ซึ่งจะส่งต่อให้กับขา CS ของอุปกรณ์ไอโอ เพื่อปล่อยข้อมูลบนทางเดินข้อมูล และซีพียู จะให้สัญญาณอ่าน RD เป็นการอ่านข้อมูลนั้นเข้าสู่ซีพียู



รูปที่ ข.3 วงจรทดลองอ่านข้อมูล

คำสั่งที่ใช้ในการอ่านจากพอร์ต จะใช้คำสั่ง XBY (พอร์ต) ซึ่งตำแหน่งของพอร์ตอยู่ที่ 0E0A0H.

วิธีทดลอง

1. เปิดเครื่องแผงควบคุมและแผงทดลอง พร้อมทั้งต่อสายให้เรียบร้อย
2. กดปุ่มอักษรว่าง (Space Bar) ที่แผงแป้นอักขระ
3. ป้อนโปรแกรมต่อไปนี้

   > 5 REM DIP_SW

   > 10 A= 0E0A0H

   > 20 PH0. XBY(A)

   > RUN
4. อ่านค่าจากแอลซีดี ว่าค่าที่ได้ตรงกับดิปสวิตช์หรือไม่
5. ลองปรับเปลี่ยนค่าที่ดิปสวิตช์ และสังเกตผลการทดลอง
6. ให้เขียนโปรแกรมอ่านข้อมูลจากดิปสวิตช์ และนำค่าที่ได้มาแสดงออกที่พอร์ต 1

คำถามท้ายการทดลอง

1. ให้ทำการออกแบบวงจรถอดรหัส ซึ่งมีจุดถอดรหัส 1 จุด ครอบคลุมถึง 64 ตำแหน่ง
2. จากวงจรทดลองตัวต้านทานออกแบบแถว (R-Pack) ที่ต่ออยู่ทางดิปสวิตช์ต่อไว้เพื่อประโยชน์อะไร
3. ให้บอกถึงการประยุกต์ใช้อินพุตมาสัก 5 ตัวอย่าง

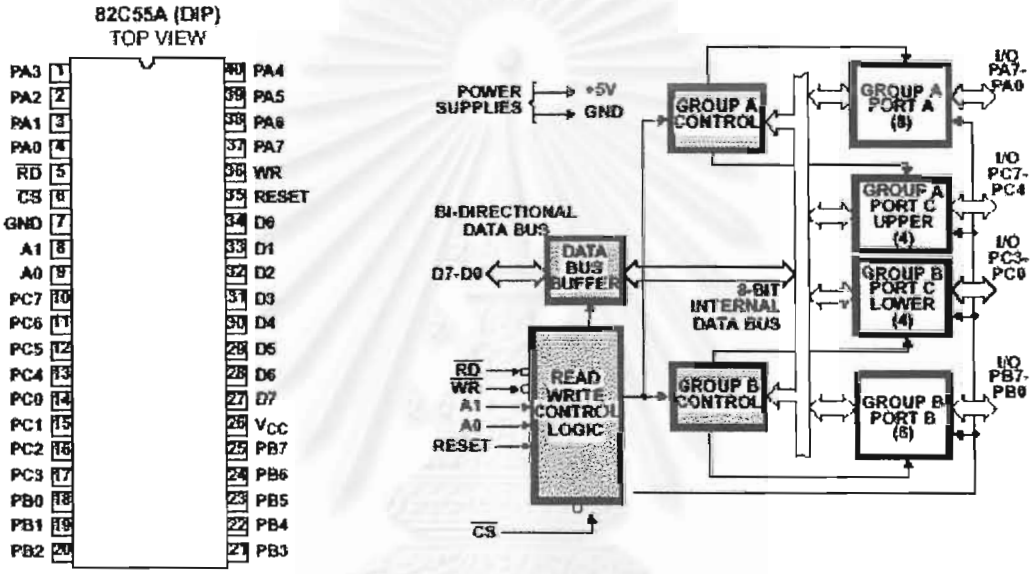
การทดลองที่ 2 : 8255 และพื้นฐานเอาต์พุต

วัตถุประสงค์

1) ให้ทราบถึงหน้าที่ต่าง ๆ ของชิป 8255

2) ให้ทราบถึงการส่งข้อมูลออกไปยังอุปกรณ์เอาต์พุต

ทฤษฎี



รูปที่ ข.4 ไอซี 8255 และโครงสร้างภายใน

ทฤษฎี

8255 สามารถที่จะโปรแกรมให้เป็นอินพุต หรือเอาต์พุตก็ได้ตามต้องการโดยมีไอโอพอร์ตขนาด 8 บิต อยู่ 3 พอร์ต กับอีก 1 พอร์ตควบคุม มีเส้นทางเดินข้อมูล (Data Bus) ที่ใช้ติดต่อกับซีพียูจำนวน 8 เส้น และมีตำแหน่ง A0, A1 เป็นตัวกำหนดการติดต่อกับไอโอ พอร์ตทั้ง 3 คือ พอร์ต A, B และ C ตามลำดับโดยมีการกำหนดตำแหน่ง ดังนี้

ตารางที่ ข.3 ตำแหน่งของพอร์ตใน 8255

A1	A2	ชื่อพอร์ต
0	0	พอร์ต A
0	1	พอร์ต B
1	0	พอร์ต C
1	1	

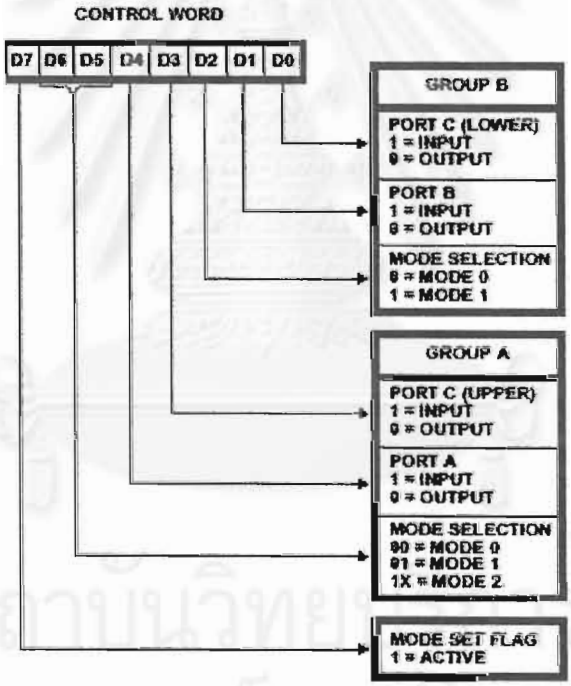
นอกจากนี้ยังมีสัญญาณควบคุมอีก 4 เส้น ดังนี้

1) RD เป็นสัญญาณการอ่านพอร์ต 8255
2) WR เป็นสัญญาณที่เขียนข้อมูลมาที่ตัวพอร์ต
3) CS เป็นขาเปิดให้ 8255 ทำงาน
4) การตั้งค่าซ้ำของฮาร์ดแวร์ (Hardware Reset) ยกเลิกการทำงานของ 8255 ขาการตั้งค่าซ้ำนี้จะทำงานที่ตรรกะมีค่าเป็น 1

การโปรแกรม 8255

การโปรแกรมต้องถูกกระทำก่อนการใช้งานอย่างน้อยที่สุด 1 ครั้ง โดยการเขียนรหัสควบคุมไปที่พอร์ตควบคุม (A1, A0 =1)

รหัสควบคุม (Control Word) ในแต่ละบิตมีความหมายดังนี้



รูปที่ ข.5 รายละเอียดของเรจิสเตอร์ควบคุมภายใน 8255

D7 แสดงถึงรหัสควบคุมให้เริ่มต้นการทำงาน (1=ทำงาน) คือจะมีผลทำให้ 8255 รับรู้สิ่งต่อไปในบิตต่างที่จะกำหนดให้ เพราะฉะนั้นเวลาจะสั่งงานหรือหน้าที่ให้กับ 8255 บิตนี้จะเป็น 1 เสมอ

D6 และ D5 เป็นการเลือกแบบวิธี (Mode) ในการทำงานของพอร์ต A ซึ่งมี 3 แบบ

D4 กำหนดให้ พอร์ต A เป็นอินพุตหรือเอาต์พุต โดย

0 = เอาต์พุตพอร์ต

1 = อินพุตพอร์ต

D3 กำหนดให้ พอร์ต C บน (PC7-PC4) เป็นอินพุตหรือเอาต์พุต โดย

0 = เอาต์พุตพอร์ต

1 = อินพุตพอร์ต

D2 เป็นการเลือกแบบวิธี ให้กับพอร์ต B

0 = แบบวิธีที่ 0 (Mode 0)

1 = แบบวิธีที่ 1 (Mode 1)

D1 กำหนดให้ พอร์ต B เป็นอินพุตหรือเอาต์พุต โดย

0 = เอาต์พุตพอร์ต

1 = อินพุตพอร์ต

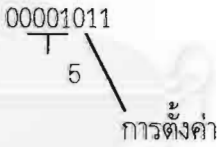
D0 กำหนดให้ พอร์ต C บน (PC0-PC3) เป็นอินพุตหรือเอาต์พุต โดย

0 = เอาต์พุต

1 = อินพุต

และในรหัสควบคุมนี้ยังมีแบบวิธีพิเศษอีก เมื่อให้บิตที่ 7 ของรหัสควบคุมมีค่าเป็น 0 ในแบบวิธีนี้ผู้ใช้งานสามารถตั้งค่าหรือตั้งค่าบิตซ้ำในบิตใดบิตหนึ่งของพอร์ต C ได้โดยมีอิสรภาพ ดังรูปแบบต่อไปนี้

คือ บิตที่ 7 ต้องเป็น 0 ส่วน D6-4 จะเป็นอะไรก็ได้ ในที่นี้กำหนดให้มีค่าเป็น 0 ซึ่งทำให้ 4 บิตสูงเป็น 0 ส่วน D3-D1 จะเป็นตัวกำหนดตำแหน่งบิตที่ต้องการ โดยมีบิตสูงใช้ในการเปิดหรือปิด เช่นต้องการบิต 5 เปิด จะได้



นั่นก็คือ นำค่า 0BH ออกไปยังพอร์ตควบคุม ซึ่งใช้ได้เฉพาะพอร์ต C เท่านั้น และเมื่ออยู่ในแบบวิธี 0 ใช้ในการตั้งค่าหรือการตั้งค่าซ้ำ เฉพาะบิตที่ต้องการเท่านั้น ประโยชน์ใช้ในการปิดเปิดอุปกรณ์ได้ในแต่ละช่องอิสระ ยกตัวอย่าง เช่น นาฬิกาเปิดปิดอุปกรณ์ ช่อง 1 เปิด 11.00 น. ปิด 11.30 น. ช่อง 2 ปิด 11.10 น. และเปิด 11.20 น. ก็ต้องส่งสัญญาณไปปิดช่อง 2 ในขณะเดียวกันก็จะส่งสัญญาณไปเปิดช่อง 1 ไว้ด้วย ในกรณีออกแบบธรรมดา แต่ถ้าใช้เป็นบิตเราไม่ต้องคำนึงถึงช่องอื่น

เช่น การตั้งค่าบิต 4

```
MOV     DPTR,#0E003H
MOV     A,#80H
MOVX    @ DPTR,A
MOV     A,#00001001B
MOVX    @ DPTR,A        ;SET PC4
DEC     A
```

MOVX @ DPTR,A

จากโปรแกรมนี้จะเห็นได้ว่านำไปสร้างจังหวะอิสระได้

วิธีการทำงานของ 8255 มีอยู่ 3 แบบวิธี คือ

แบบวิธี 0 เป็นแบบวิธีอินพุตเอาต์พุตแบบพื้นฐาน ที่ใช้กันโดยทั่วไป โดยแบบวิธีนี้สั่งเป็นอินพุตหรือเอาต์พุต ได้ทั้ง 3 พอร์ต (A, B และ C)

แบบวิธี 1 เป็นแบบวิธีอินพุตเอาต์พุตที่มีการตรวจสอบสัญญาณซึ่งกันและกัน (Hand Shaking) ระหว่าง 8255 กับอุปกรณ์ภายนอก โดยในโหมดนี้จะใช้สั่งงานให้เป็นอินพุตหรือเอาต์พุตได้ที่พอร์ต A และ B เท่านั้น ส่วนพอร์ต C จะใช้เป็นส่วนตรวจสอบสัญญาณความพร้อมซึ่งกันและกัน

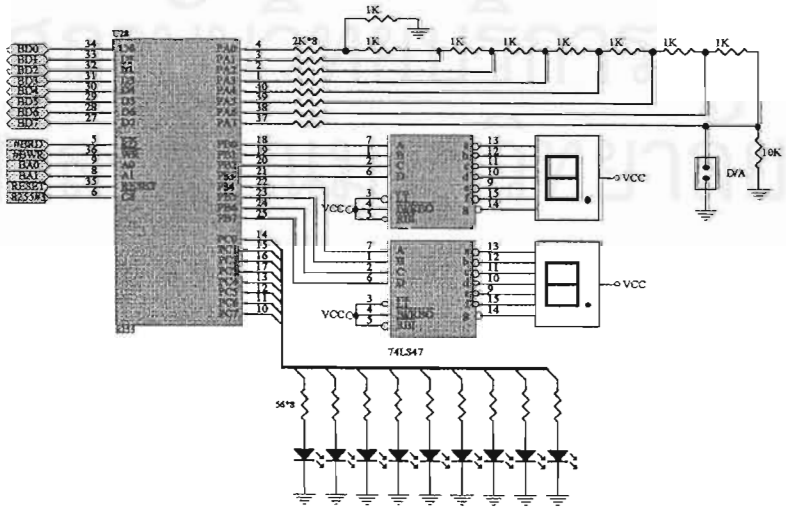
แบบวิธี 2 เป็นแบบวิธีเป็นสองทิศทาง คือพอร์ตเป็นได้ทั้งอินพุตและเอาต์พุตภายในพอร์ตเดียวกัน และในแบบวิธีนี้ยังมีการตรวจสอบความพร้อมโดยใช้พอร์ต C ในการตรวจสอบที่เหลือคือพอร์ต B ที่ผู้ใช้งานจะสั่งให้อยู่ในแบบวิธี 0 หรือ 1 ได้อย่างอิสระ

อุปกรณ์การทดลอง

1. แหล่งจ่ายกระแสไฟตรง
2. แผงควบคุม
3. แผงต่อทดลอง

การทดลอง

การเขียนข้อมูลออกไปยังพอร์ตอุปกรณ์เอาต์พุต โดยซีพียูจะส่งตำแหน่งไปเลือกการทำงานของชิปเอาต์-พุต โดยข้อมูลจะถูกส่งไปยังอุปกรณ์พร้อมกับคำสั่งเขียน



รูปที่ ข.6 วงจรทดลองพอร์ตเอาต์พุต

1. ป้อนโปรแกรมขับแอลอีดีแบบ 7 ส่วน ที่ตำแหน่ง 0E0B9H ดังนี้

```
>  5  REM  7 SEGMENTS
> 10  PB  =  0E0B9H : CP = 0E0BBH
> 15  XBY (CP) = 80H : A= 0
> 20  XBY (PB) = A
> 30  A =  A+1
> 40  FOR I = 1 TO 300  : NEXT I ; หน่วงเวลา
> 50  B = A.AND. 0F0H
> 60  IF  B > 9    THEN A=10 H
> 70  IF  A = 32   THEN GOTO 90
> 80  GOTO 20
> 90  END
```

2. ทดลองดำเนินการ (RUN) และสังเกตผลการทำงานของโปรแกรม
3. ทดลองเปลี่ยนค่าบรรทัดที่ 40 แล้วสังเกตผลการทำงาน
4. ทดลองเขียนโปรแกรมไฟวิ่งที่พอร์ต C ของ 8255 ที่ตำแหน่ง EBAH จากซ้ายไปขวาและจากขวาไปซ้ายตามลำดับ

คำถามท้ายการทดลอง

1. ให้อธิบายถึงเอาต์พุตพอร์ต จะมีคุณสมบัติอย่างไรบ้างจึงเป็นเอาต์พุตพอร์ตได้
2. อธิบายข้อดีและข้อเสียของการนำ 8255 มาทำเป็น อินพุตหรือเอาต์พุต

การทดลองที่ 3 : แผงแป้นอักขระ (Keyboard)

วัตถุประสงค์

1. เพื่อให้ทราบถึงวงจรแผงแป้นอักขระ
2. สามารถทำโปรแกรมจัดการกับแผงแป้นอักขระได้

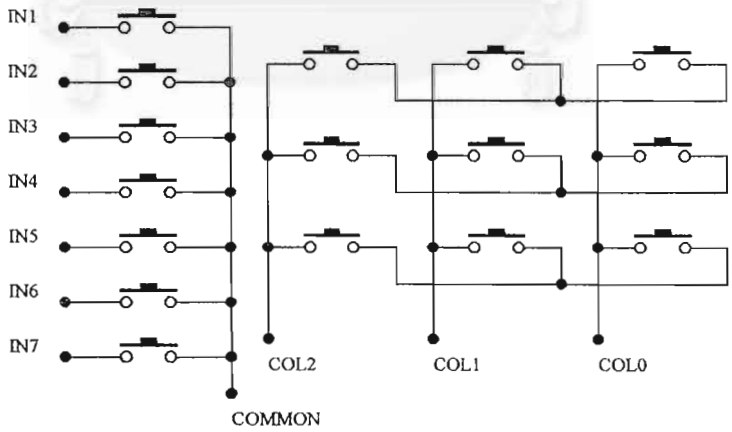
ทฤษฎี

แผงแป้นอักขระ (Keyboard) นับเป็นสิ่งที่สำคัญอย่างหนึ่ง เพราะแผงแป้นอักขระ (Keyboard) เป็นอุปกรณ์อินพุตที่ทำให้ผู้ใช้งานสามารถติดต่อกับเครื่องควบคุมในการทำงานต่าง ๆ โดยแผงแป้นอักขระ (Keyboard) จะประกอบด้วยส่วนสำคัญ 3 ส่วน คือ

1. ส่วนของปุ่มของแผงแป้นอักขระ
2. ส่วนของวงจรเข้ารหัส โดยส่วนนี้จะทำหน้าที่เข้ารหัสเพื่อรับรู้ตำแหน่งแป้น (Key) และสามารถให้ค่าแป้นพิมพ์ของแต่ละตัวเมื่อมีการกดแป้น ที่เรียกว่ารหัสแป้นพิมพ์ (KEYCODE) ได้
3. ส่วนของวงจรถอดรหัสจะเป็นส่วนที่ทำหน้าที่เปลี่ยนรหัสแป้นพิมพ์ ให้เป็นรหัสที่นำไปใช้งานได้เช่น

วงจรแผงแป้นอักขระแบ่งได้เป็น 2 ลักษณะใหญ่ ๆ คือ

1. แบบขาร่วม
2. แบบ เมทริกซ์ (Matrix)



รูปที่ ข.7 วงจรแผงแป้นอักขระในลักษณะต่างๆ

จะเห็นว่าแบบขาร่วมจะเป็นแบบที่เข้าใจง่ายและการรับรู้การกดแป้นพิมพ์จะเป็นไปในลักษณะเฉพาะตัวของแต่ละแป้นพิมพ์ ซึ่งทำให้ง่ายต่อการออกแบบ สะดวกและประหยัดในกรณีที่ใช้ปุ่มไม่มากนัก ส่วนแบบ Matrix จะใช้สายในการต่อเข้ากับวงจรเข้ารหัสได้น้อยกว่าโดยให้จำนวนจุดของแป้นพิมพ์ได้จำนวนมาก ซึ่งจำนวนแป้นจะขึ้นอยู่กับสายทางด้านแถว (Row) และสายทางด้านสดมภ์ (Column) แต่การออกแบบ

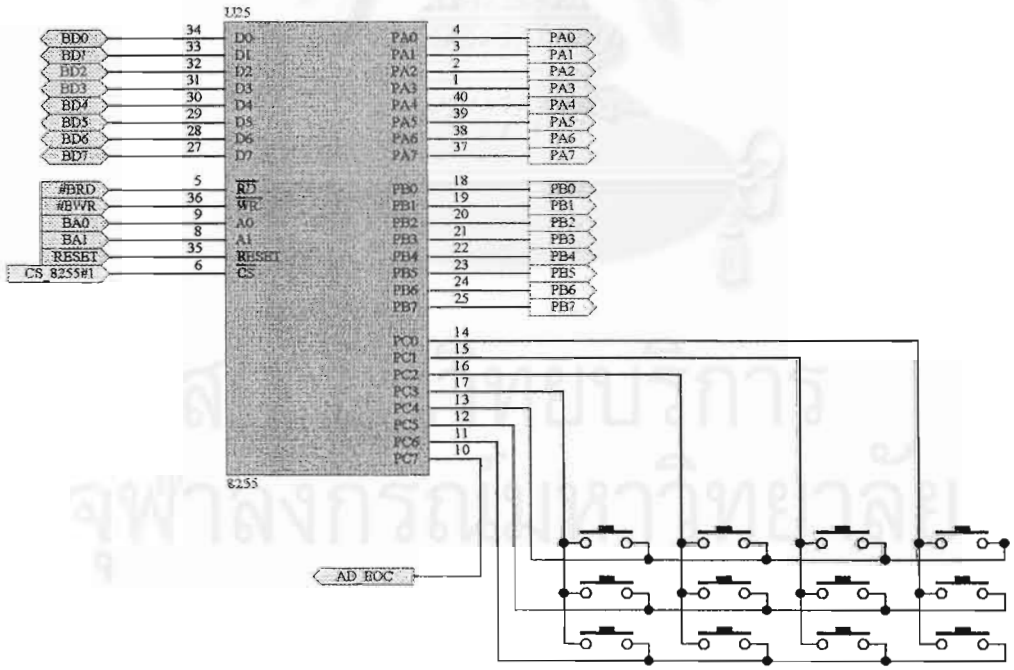
วงจรเข้ารหัสจะซับซ้อนขึ้นโดยสามารถแสดงแผนภาพกรอบของวงจรเข้ารหัสแผงแป้นอักขระ ไว้ดังรูป

จากสัญญาณนาฬิกาจะใช้ตรวจสอบว่าแป้นพิมพ์ใดถูกกด โดยไม่มีการกดแป้นพิมพ์ ตัวกราดตรวจจะส่งสัญญาณกราดตรวจไปที่ปุ่ม เมื่อมีการกดแป้นพิมพ์ สัญญาณนาฬิกาจะหยุดการทำงาน ทำให้ตัวกราดตรวจหยุดการกราดตรวจ วงจรนับจะหยุดนับและคงค่าสุดท้ายไว้ (รหัสแป้นพิมพ์ที่ถูกกด) จนกว่าจะมีการปล่อยแป้นพิมพ์ ถึงจะรับค่าแผงแป้นอักขระตัวใหม่ได้ ซึ่งลักษณะการกราดตรวจแบบเมทริกซ์นี้ จะเริ่มกราดตรวจจากแถวแรกไปยังแถวสุดท้ายและจากสดมภ์แรกไปยังสดมภ์สุดท้ายพร้อมกัน ดังนั้นเมื่อแป้นพิมพ์ใดถูกกดจะทำให้ทราบว่าแป้นพิมพ์ที่ถูกกดอยู่แถวและสดมภ์ใด ได้จากส่วนวงจรนับ

อุปกรณ์การทดลอง

1. แผงควบคุม
2. แผงต่อทดลอง
3. แหล่งจ่ายกระแสตรง
4. ลอจิกโพรบ

วงจรทดลอง



รูปที่ ข.8 วงจรแผงแป้นอักขระขนาด 4*3

ในรูปที่ ข.8 เป็นวงจรแผงแป้นอักขระขนาด 4*3 คือ มีแนวนอนที่ใช้เป็นด้านอินพุต (อ่านข้อมูลทางแผงแป้นอักขระ) 3 เส้นซึ่งจำเป็นต้องมีพูลอัป (R-pull Up) ไว้เพื่อเมื่อยังไม่ได้กดแผงแป้นอักขระจะได้สถานะคงที่อยู่

ค่าหนึ่ง เพราะเราจะกราดตรวจตรรกะ 0 ทางด้านสดมภ์ หลักการก็คือส่งข้อมูลตรรกะ 0 ให้กับสดมภ์แรกและอ่านข้อมูลทางด้านแนวนอน ดูว่าแนวนอนเส้นใดเป็น 0 บ้าง ถ้ามีแสดงว่ามีการกดแผงแป้นอักขระ ก็ไปทำการตรวจสอบว่าเป็นด้านแนวนอนใดและสดมภ์ใด เพื่อทำการเข้ารหัสแผงแป้นอักขระแล้วแปลงเป็นค่าที่ต้องการไปใช้งาน แต่ถ้าอ่านแล้วไม่มีแนวนอนเส้นใดเป็น 0 ก็จะกราดตรวจแนวตั้งถัดไป แล้วอ่านแนวนอนใหม่ ทำอย่างนี้จนกว่าจะหมดสดมภ์

การทดลอง

1. ให้ป้อนโปรแกรมต่อไปนี้ และเมื่อกดแป้นอักขระใด ๆ ค่าแป้นพิมพ์ที่ถูกกดจะถูกเก็บไว้ในตัวแปร C และ D และแสดงค่าที่กดออกหน้าจอแอลซีดี

```
 5  REM  4*3 KEY
10  PC = 0E0B2H, CP = 0E0B3H
20  XBY (CP) = 88H
30  A=0
40  B = 2**A : B = B .XOR. 0FFH
50  XBY (PC) = B
55  FOR I = 1 TO 10 : NEXT 1
60  C = XBY (PC)
70  IF C < 0F0H THEN 150
80  A = A+1
90  IF A = 4 THEN GOTO 20
100 GOTO 30
150 IF A = 0 THEN GOTO 250      ; ROW0
160 IF A = 1 THEN GOTO 300      ; ROW1
170 IF A = 2 THEN GOTO 350      ; ROW2
180 IF A = 3 THEN GOTO 400      ; ROW3
250 C = C ..AND. 70H
260 IF C = 60H THEN PRINT "*"
270 IF C = 50H THEN PRINT "0"
280 IF C = 30H THEN PRINT "#"
290 END
300 C = C . AND. 70H
310 IF C = 60H THEN  D=9
```

```
320  IF C = 50H THEN  D=8
330  IF C = 30H THEN  D=7
340  PRINT D : END
350  C = C ..AND.70H
360  IF C = 60H THEN  D=6
370  IF C = 50H THEN  D=5
380  IF C = 30H THEN  D=4
390  PRINT D : END
400  C = C .AND.70H
410  IF C = 60H THEN  D=3
420  IF C = 50H THEN  D=2
430  IF C = 30H THEN  D=1
440  PRINT D : END
```

2. ลองกดแป้นพิมพ์แล้วตรวจดูค่าว่าถูกต้องหรือไม่ พร้อมทั้งศึกษาการทำงาน
3. เขียนโปรแกรมรับข้อมูลจากแผงแป้นอักขระแล้วนำค่าที่ได้ไปแสดงผลที่แอลอีดี 7 ส่วน

คำถามท้ายการทดลอง

1. ให้อธิบายถึงการโปรแกรมตรวจกราดแป้นกด (Scan Key) แบบเมทริกซ์
2. การอ่านแผงแป้นอักขระแบบเมทริกซ์ ถ้ามีการกดแป้นพิมพ์พร้อมกัน 2 แป้นในแถว (Row) ที่ต่างกัน ซึ่ง ทำให้อ่านข้อมูลผิดพลาด ผู้ใช้งานจะแก้ไขเหตุการณ์นี้ทางซอฟต์แวร์ด้วยวิธีใด

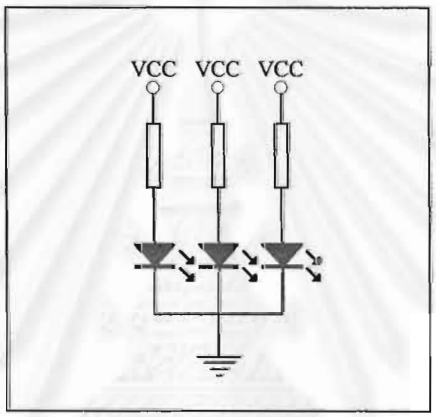
การทดลองที่ 4 : การแสดงผล (Display) แอลอีดี 8 หลักและ 8 แถว

<u>วัตถุประสงค์</u>

1. เพื่อให้รู้จักการแสดงผลอย่างง่าย ๆ
2. เพื่อให้รู้จักการกราดตรวจการแสดงผล
3. เพื่อให้เกิดแนวความคิดในการออกแบบ

ทฤษฎี

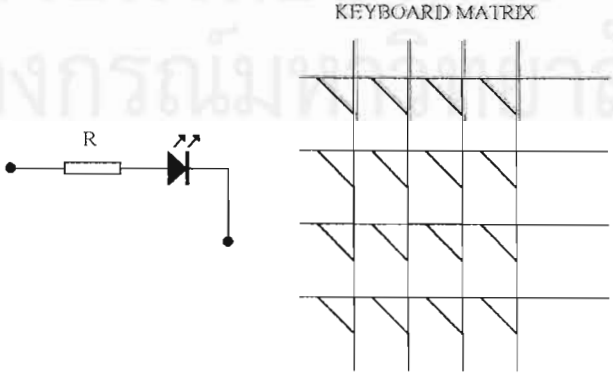
จะขอกล่าวถึงการแสดงผลอย่างง่าย ๆ ก่อนนั่นคือ การแสดงผลแบบแถวเดียว



รูปที่ ข.9 การแสดงผลแบบแถวเดียว

ในรูปที่ ข.9 จะเห็นได้ว่าผู้ใช้งานว่าสามารถควบคุมสัญญาณเพียงด้านเดียว คือส่งแต่ข้อมูล แล้วหน่วงเวลาไว้ระยะหนึ่ง จากนั้นเมื่อส่งสัญญาณตัวต่อไปก็จะทำให้เราเห็นเป็นไฟวิ่งได้

จากนั้นจะสามารถให้เห็นเป็นตัวอักษรหรือรูปภาพได้ ถ้าเพิ่มตรรกะการควบคุมให้กับวงจรแทนที่จะควบคุมการติดดับเพียงด้านเดียวก็เป็น 2 ดังรูปที่ ข.10



รูปที่ ข.10 การต่อแอลอีดีแบบแมทริกซ์

ในรูปที่ ข.10 ถ้ามีการส่งข้อมูลออกไปทางแนวนอนและให้สดมภ์ใดเป็น 0 จะทำให้แอลอีดีถูกไบแอสในทิศทางไปหน้า และติดสว่างตามบิตข้อมูลที่ส่งออกไปทางแถวแนวนอน จากหลักการนี้จึงทำให้สามารถควบคุมแอลอีดีทุกดวง ในแผงเมทริกซ์ให้ติดหรือดับได้ตามต้องการ

หลักการเกิดภาพบนแอลอีดีเมทริกซ์

เนื่องจากการมองเห็นของนัยน์ตาคนเรานั้นจะมีการคงภาพ คือเห็นภาพนั้นอยู่นาน 1/6 วินาที ถึงแม้ว่าภาพนั้นจะไม่ปรากฏแล้วก็ตาม จากหลักการของการมองเห็นนี้ ถ้าสามารถทำให้แอลอีดีติดและดับตามจุดต่าง ๆ ได้เร็วกว่า 1/6 วินาที จะเห็นได้ว่าแอลอีดีที่ตำแหน่งนั้นติดค้างอยู่

แต่เนื่องจากซีพียู ทำงานด้วยความเร็วสูงการที่จะให้แอลอีดีติดและเปลี่ยนตำแหน่งการติดว่างไปเรื่อย ๆ ตามความเร็วของซีพียู แล้วจะทำให้ไม่ปรากฏรูปภาพ แต่จะเห็นได้ว่าแอลอีดีนั้นติดสว่างเรือง ๆ ทุกดวง เหตุที่เป็นเช่นนี้ก็เพราะว่า ความสัมพันธ์ของกระแส ความอิ่มตัวของทรานซิสเตอร์ รวมทั้งกระแสที่ทำให้แอลอีดีติดสว่าง เป็นปัจจัยทำให้เห็นแอลอีดีติดทุกดวง ด้วยเหตุนี้เมื่อทำให้แอลอีดีดวงใดติดแล้ว ควรมีการหน่วงเวลาเพื่อให้มีกระแสเพียงพอที่จะทำให้แอลอีดีติดสว่าง ยิ่งหน่วงเวลานานเท่าไรยิ่งทำให้กระแสเข้าใกล้จุดสูงสุด แต่เนื่องจากต้องกราดตรวจหลาย ๆ สดมภ์ ถ้าทำการหน่วงเวลามากเกินไปจะเกิดผลกระทบต่อแอลอีดีดวงถัดไปคือมีวงรอบในการนำกระแสเป็นช่วง ๆ ซึ่งระยะเวลาการนำกระแสของแอลอีดีแต่ละดวงอาจห่างไกลจากความคงแสงของตามนุษย์ ซึ่งมีผลทำให้เห็นเป็นภาพกะพริบ หรือพลิ้ว ในการกราดตรวจแบบสดมภ์นี้เวลาที่ทำให้แอลอีดีแต่ละดวงติดนานเท่าไรนั้นขึ้นอยู่กับจำนวนสดมภ์ของแอลอีดีด้วย

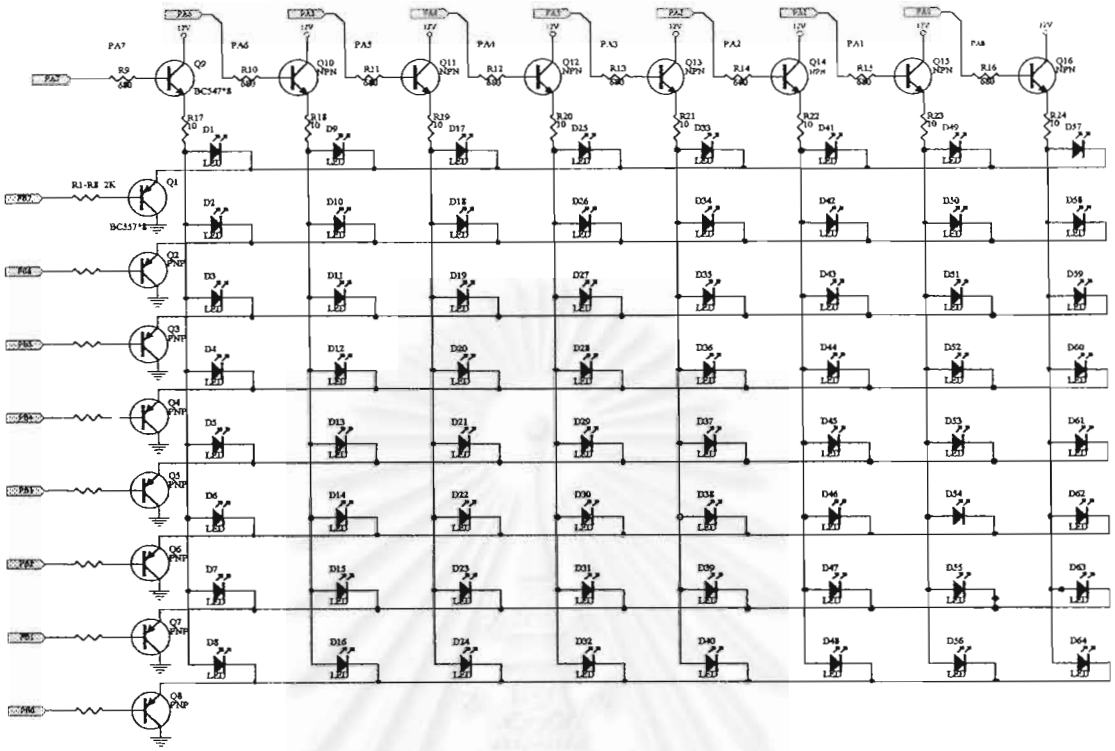
ปัญหาในการกราดตรวจทางสดมภ์คือ ถ้าจำนวนสดมภ์มีมาก เวลาในการนำกระแสของแอลอีดีแต่ละสดมภ์ก็จะมีน้อยตามไปด้วย ทำให้เกิดการพลิ้ว

ทางแก้ไขคือ เมื่อจำนวนสดมภ์มากเกินไปควรเปลี่ยนมาเป็นการกราดตรวจทางแนวนอนแทน ซึ่งทำให้เวลาของแอลอีดีในแต่ละดวง ที่จะติดและความสว่างมีเวลาคงที่มากขึ้น ในการทดลองครั้งนี้คงไม่เห็นความแตกต่างมากนักเพราะแผงทดลองมีแอลอีดีเพียง 8 แถว และ 8 สดมภ์ ซึ่งมีจำนวนเท่ากัน

<u>การคิดตัวอักษร (Font)</u>

การคิดข้อมูลที่จะนำมาแสดงเป็นตัวอักษร มิได้ถูกกำหนดไว้เป็นมาตรฐานหรือเป็นทฤษฎี แต่ขึ้นอยู่กับฮาร์ดแวร์นั้น ๆ เมื่อศึกษาวงจรการแสดงผล 8 คูณ 8 จะพบว่ามีการกำหนดให้พอร์ต A ของ 8255 เป็นพอร์ตสำหรับจ่าย (ผ่าน 74LS245) และทางพอร์ต B เป็นตัวรับ (ผ่านทางซิสเตอร์)

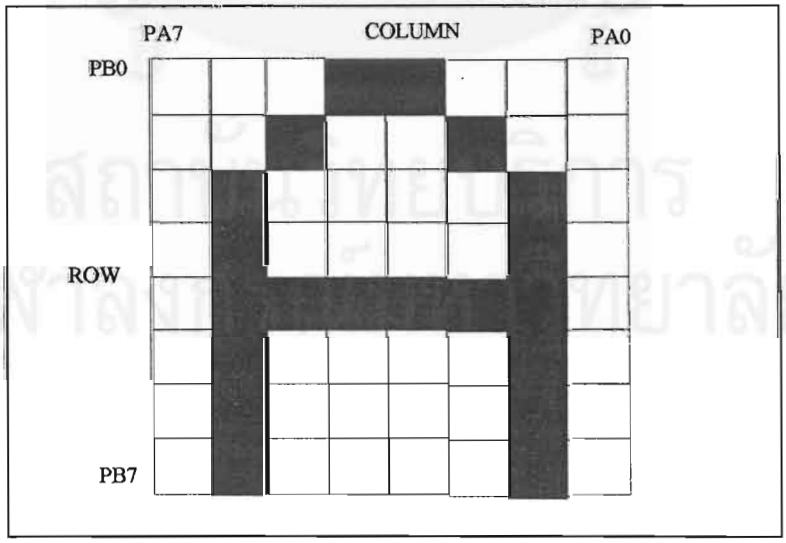
ฉะนั้นข้อมูลที่จะทำให้แอลอีดีติดได้นั้นคือ ข้อมูลที่ออกมาจากพอร์ต A ซึ่งมีค่าเป็นสูง และข้อมูลทางพอร์ต B ต้องมีค่าเป็นต่ำ แต่เนื่องจากพอร์ต B มีทรานซิสเตอร์อยู่ 1 ตัว เป็นตัวรับ จึงต้องคิดข้อมูลเป็นตรรกะบวก คือ 1= แอลอีดี (บวก) ติด



รูปที่ ข.11 วงจรแอลอีดี 8 คูณ 8

ตัวอย่าง

ให้หาข้อมูลที่แสดง ตัวอักษร A บนแผงแสดงผลแอลอีดี โดยให้ข้อมูลที่หานั้นออกทางพอร์ต B และทำการกราดตรวจทางพอร์ต A



รูปที่ ข.12 การสร้างตัวอักษร

ข้อมูลที่ได้ (FONT)	เมื่อทำการกราดตรวจ
PB=00H	PA7=0
PB=FCH	PA6=0
PB=12H	PA5=0
PB=11H	PA4=0
PB=11H	PA3=0
PB=12H	PA2=0
PB=FCH	PA1=0
PB=00H	PA0=0

อุปกรณ์การทดลอง

1. แผงควบคุม
2. แผงต่อทดลอง
3. แหล่งจ่ายกระแสไฟตรง
4. ลอจิกโพรบ

การทดลอง

1. ป้อนโปรแกรมแสดงผลแบบ 8 คูณ 8 แอลอีดี โดยตำแหน่งในแถวแนวนอนที่ 0E0B0H และตำแหน่งใช้กราดตรวจ (Scan) คือ 0E0B1H ดังนี้

```
 5  REM 8*8 LED
10  PA = 0E0B0H : PB = PA+1 : CP = PA+P
20  XBY (CP) = 80H
25  FOR M=1 TO 50
30  B=7 : FOR I=1 TO 8
40  READ A
50  XBY (PA) = A
60  C= 2**B : C=C.XOR.0FFH
70  XBY (PB) = C
80  FOR T= 0 TO 10  : NEXT T
90  B = B-1
100 NEXT I
110 RESTORE
```

```
120  NEXT M
200  DATA 24, 36, 66, 66, 126, 66, 66, 66
```

2. สังเกตผลที่ได้ว่าเป็นตัวอักษรอะไร พร้อมทั้งศึกษาการทำงานของโปรแกรม
3. ให้เขียนโปรแกรมแอลอีดี 8*8 ติดเป็นเส้นทแยงมุม

การทดลองที่ 5 : มอเตอร์กระแสตรงและสเต็ปปิงมอเตอร์

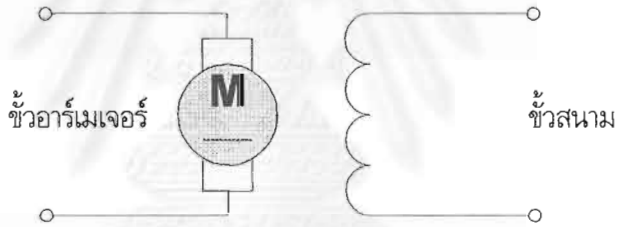
วัตถุประสงค์

1) เพื่อศึกษาความรู้เบื้องต้นของมอเตอร์กระแสตรง
2) เพื่อศึกษาวิธีการควบคุมมอเตอร์กระแสตรง
3) เพื่อให้เกิดแนวความคิดในการนำไปประยุกต์ในการใช้งานจริง

ทฤษฎี

มอเตอร์กระแสตรง (DC Motor) นั้นสามารถจำแนกออกไปได้อีกหลายประเภท ซึ่งขึ้นอยู่กับวิธีการสร้างจึงขอกล่าวรวม ๆ ที่รู้จักกันเป็นส่วนใหญ่ในปัจจุบัน นั่นคือ มอเตอร์กระแสตรงแบบขนาน (Shunt), แบบอนุกรม (Series), แบบผสม (Compound) และ แบบแม่เหล็กถาวร (PM Motor หรือ Permanent Magnet Motor)

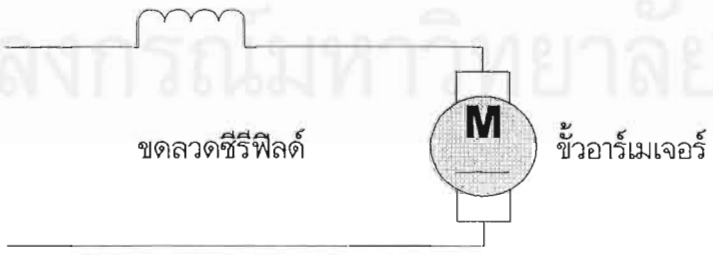
ลักษณะโครงสร้างของมอเตอร์แบบขนาน



รูปที่ ข.13 โครงสร้างของมอเตอร์แบบขนาน

มอเตอร์แบบนี้สามารถปรับเส้นแรงแม่เหล็กได้อย่างอิสระต่อกระแสของอาร์เมเจอร์เป็นผลให้สามารถควบคุมพารามิเตอร์ ให้มีค่าคงที่ได้ตลอดช่วงพิสัยที่กว้าง

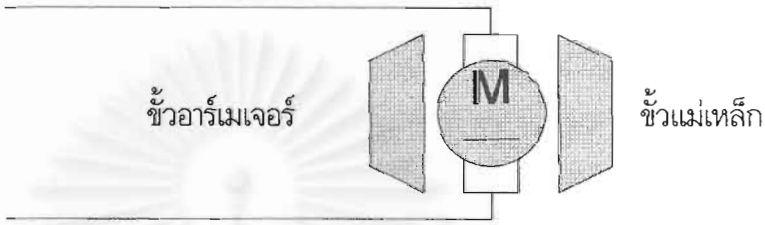
ลักษณะโครงสร้างของมอเตอร์แบบอนุกรม



รูปที่ ข.14 โครงสร้างของมอเตอร์แบบอนุกรม

มอเตอร์แบบนี้จะมีเส้นแรงแม่เหล็กเป็นสัดส่วนกับกระแสดังนั้นเส้นแรงของสนามแม่เหล็กจึงสามารถปรับค่าได้ ซึ่งเราจะได้ความสัมพันธ์ ระหว่างความเร็วและแรงบิดเป็นแบบไม่เชิงเส้น จึงเหมาะที่จะนำไปใช้งานภาวะเฉพาะคือ เมื่อต้องการแรงบิดสูงที่ความเร็วต่ำ และแรงบิดต่ำที่ความเร็วสูง เช่น ระบบการขับเคลื่อนของรถลาก

ลักษณะโครงสร้างของมอเตอร์แบบแม่เหล็กถาวร



รูปที่ ข.15 ลักษณะโครงสร้างของมอเตอร์แบบแม่เหล็กถาวร

มอเตอร์แบบนี้จะใช้ส่วนกระตุ้นสนามของมอเตอร์เป็นแบบแม่เหล็กถาวร ซึ่งต่างกับที่กล่าวมาข้างต้นที่ใช้ขดลวด แบบนี้จะให้เส้นแรงของสนามมีค่าคงที่ ดังนั้น อัตราส่วนระหว่างกระแสของอาร์เมเจอร์และแรงบิดจะมีค่าคงที่ด้วย ซึ่งมีข้อดีคือไม่มีกำลังสูญเสียในสนาม มีประสิทธิภาพสูงกว่าและมีขนาดเล็กกว่า เมื่อเทียบกับมอเตอร์แบบใช้ขดลวดในการกระตุ้นที่มีขนาดกำลังม้าเท่ากัน จึงเหมาะกับงานที่ต้องการแรงบิดของการรับภาระสูง

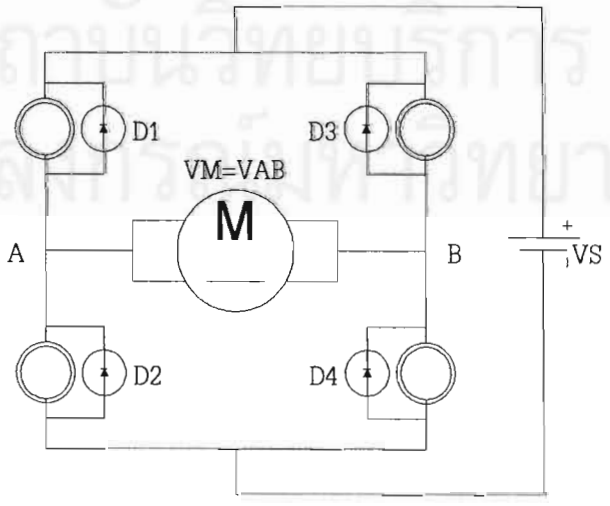
การควบคุมแบบมอดูเลตความกว้างพัลส์ สามารถแบ่งได้เป็น 3 ชนิด คือ

ไบโพลาร์

ยูนิโพลาร์

ลิมิตยูนิโพลาร์

วงจรพื้นฐานจะเป็นดังนี้

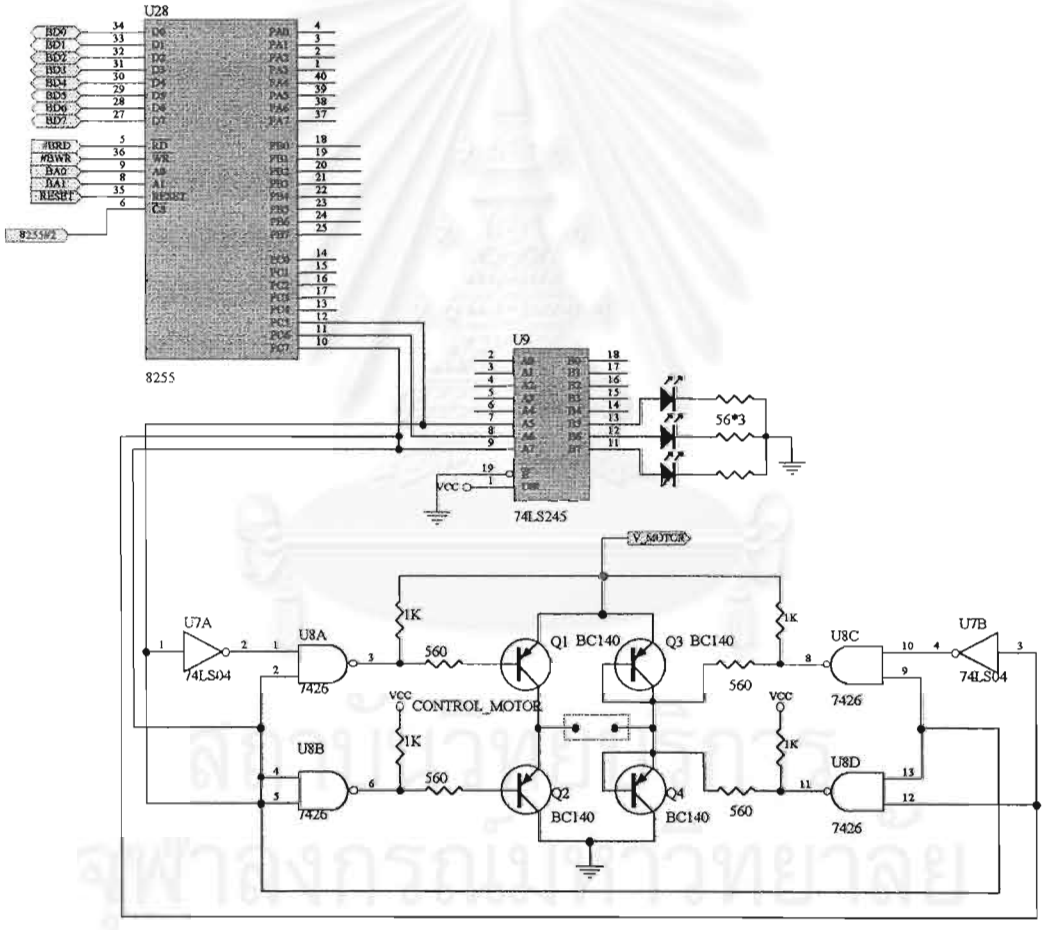


รูปที่ ข.16 วงจรพื้นฐานขับมอเตอร์กระแสตรงแบบบริดจ์

มอเตอร์ทั้ง 3 ชนิด สามารถอธิบายได้ด้วยวงจรพื้นฐานนี้ได้ทั้ง 3 ชนิด ต่างกันตรงที่การควบคุมการทำงานและการหยุดทำงานของทรานซิสเตอร์ ซึ่งในที่นี้เราจะกล่าวเฉพาะแบบไบโพลาร์ เพราะเป็นแบบที่ครอบคลุมและสามารถเข้าใจได้ง่าย คือ

เมื่อให้มอเตอร์อยู่ในเฟสทำงาน ดังนั้น T1 กับ T4 จะทำงาน และ T2 กับ T3 จะหยุดทำงาน ดังนั้น กระแสไหลจาก Vs ผ่าน T1, มอเตอร์ และ T4 ลงกราวด์ เพราะฉะนั้น Vm = Vs (มอเตอร์จะหมุนตามเข็มนาฬิกา)

เมื่อให้มอเตอร์อยู่ในเฟสไม่ทำงาน (Off) T2 กับ T3 จะทำงาน และ T1 กับ T4 จะหยุดทำงาน ดังนั้น กระแสไหลจาก Vs ผ่าน T3 ขั้วลบ, มอเตอร์ และ T2 ลงกราวด์ เพราะฉะนั้น Va = Vs (มอเตอร์จะหมุนทวนเข็มนาฬิกา)



รูปที่ ข.17 วงจรขับมอเตอร์กระแสตรง

อุปกรณ์การทดลอง

1. แผงควบคุม
2. แผงต่อทดลอง
3. แหล่งจ่ายกระแสไฟตรง

4. มิเตอร์

การทดลอง

1. ป้อนโปรแกรมควบคุมมอเตอร์กระแสตรง โดยใช้พอร์ต C ของ 8255 ที่ตำแหน่ง 0E0B6H

```
5   REM  DC MOTOR
10   PA = 0E0B6H :  CP = 0E0B7H
20   XBY (CP) = 80H : A=50
30   XBY (PC) = 0C0H
40   FOR I=1 TO 50  : NEXT T  ; TON
50   XBY (PC) = 40H
60   FOR I=1 TO 20              ; TOFF
70   A=A-1
80   IF A=0 THEN GOTO 100
90   GOTO 30
100  END
```

2. ทดลองให้บิต PC7=1, PC6=0 และ PC =1 สังเกตผลที่ได้จากการทดลอง
3. ใช้มิเตอร์วัดคร่อมระหว่างขั้วมอเตอร์ แล้วสังเกตค่าที่ได้กับการทำงานของโปรแกรม
4. เปลี่ยนค่าในบรรทัดที่ 40 และ 60 แล้วนำมิเตอร์วัดคร่อมมอเตอร์ สังเกตค่าที่ได้รับกับการทำงานของโปรแกรม

สเต็ปปิงมอเตอร์

<u>วัตถุประสงค์</u>

1) เพื่อให้ทราบเกี่ยวกับการสั่งงานสเต็ปปิงมอเตอร์
2) เพื่อให้เข้าใจเกี่ยวกับการสั่งงานของสเต็ปปิงมอเตอร์
3) เพื่อให้ทราบเกี่ยวกับวงจรขับเคลื่อนและการปรับปรุงให้มีประสิทธิภาพ
4) เพื่อให้เกิดแนวความคิดที่จะนำสเต็ปปิงไปใช้งาน

ทฤษฎี

สเต็ปปิงมอเตอร์ที่เห็นกันในปัจจุบันแบ่งได้เป็น 3 แบบ คือ

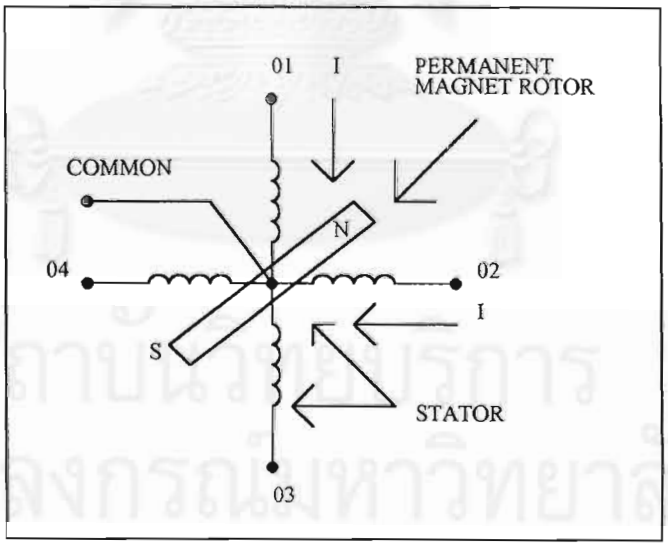
1. แบบแม่เหล็กถาวร (PM = Permanent Magnet)
2. แบบแปรค่ารีลักแตนซ์ (VR = Variable Reluctance)

3. แบบลูกผสม (H = Hybrid)

สเต็ปปิงมอเตอร์แบบ PM นั้นจะมีตัวอยู่กับที่ (Stator) ที่พันขดลวดไว้หลาย ๆ ขั้ว โดยมีตัวหมุน (Rotor) เป็นรูปทรงกระบอกฟันเลื่อย และตัวหมุนทำด้วยแม่เหล็กถาวร เมื่อป้อนกระแสตรงให้แก่ขดลวดตัวอยู่กับที่จะทำให้เกิดแรงผลักต่อตัวหมุน (Rotor) ด้วยแรงทางแม่เหล็กไฟฟ้าจะทำให้มอเตอร์หมุนมอเตอร์แบบ PM จะเกิดแรงฉุดยึดให้ตัวหมุน (Rotor) หยุดอยู่กับที่ แม้ว่าจะไม่ได้ป้อนไฟเข้าขดลวด

ส่วนมอเตอร์แบบ VR จะมีการหมุนตัวหมุนได้อย่างอิสระ แม้จะไม่ได้จ่ายไฟให้ตัวหมุนของมันจะทำจากสารเฟอร์โรแมกเนติกขนาดกำลังอ่อน และมีลักษณะเป็นฟันเลื่อยรูปทรงกระบอก โดยจะมีความสัมพันธ์โดยตรงกับจำนวนขั้วในตัวอยู่กับที่ และจึงทำหน้าที่กำหนดมุมที่หมุนไป แต่ละครั้งเมื่อป้อนไฟเข้าขดลวดตัวอยู่กับที่แรงบิดที่เกิดขึ้นจะไปหมุนตัวหมุนให้หมุนไปในเส้นทางของอำนาจแม่เหล็ก ที่มีค่ารีลักแตนซ์ต่ำที่สุด ตำแหน่งที่เกิดขึ้นแน่นอนและมีเสถียรภาพ แต่จะเกิดขึ้นได้หลาย ๆ จุด ดังนั้นเมื่อป้อนกระแสไฟฟ้าเข้าขดลวดต่าง ๆ ในมอเตอร์แตกต่างขดกันไป จะทำให้มอเตอร์หมุนไปตำแหน่งต่าง ๆ กัน ตัวหมุนของ VR จะมีความเฉื่อยของตัวหมุนน้อย จึงมีความเร็วสูงกว่ามอเตอร์แบบ PM

มอเตอร์แบบผสม (Hybrid) จะเป็นแบบลูกผสมของ VR กับ PM โดยมีตัวอยู่กับที่คล้ายกับที่ใช้ใน VR สำหรับตัวหมุนมีหมวกหุ้มปลาย ซึ่งมีลักษณะของสารแม่เหล็กที่มีกำลังสูง โดยการควบคุมขนาดรูปร่างของหมวกแม่เหล็กอย่างดี ทำให้ได้มุมการหมุนแต่ละครั้งน้อยและแม่นยำ ข้อดีก็คือ ให้แรงบิดสูง มีขนาดกะทัดรัดและให้แรงฉุดยึดตัวหมุนนิ่งกับที่เมื่อยังไม่ต่อกับแหล่งจ่ายไฟ



รูปที่ ข.18 โครงสร้างจำลองของสเต็ปปิงมอเตอร์

แสดงโครงสร้างจำลองของสเต็ปปิงมอเตอร์ ชนิด 4 เฟซ และตำแหน่งของตัวหมุนขณะจ่ายกระแสไฟแก่ 01 และ 02

ตารางที่ ข.4 มุมของตัวหมุนเทียบกับกระแสไฟฟ้าต่าง ๆ 8 ตำแหน่ง

เฟสที่จ่ายกระแสไฟฟ้า	01	0102	02	0203	03	0304	04	0401
ตำแหน่งตัวหมุน								

จากลักษณะของมุมตัวหมุนหมุนกับกระแสไฟฟ้าต่าง ๆ เราสามารถสั่งงานให้สเต็ปปิงมอเตอร์หมุนได้ 3 อย่าง คือ

1) แบบจ่ายกระแสไฟฟ้าให้เฟสเดียววนเวียนกันไป คือ 01, 02, 03, 04 การหมุนแบบนี้แรงบิดจะน้อย

2) แบบจ่ายกระแสไฟฟ้าพร้อมกันทีเดียว 2 เฟส คือ 0102, 0203, 0304, 0401 หมุนเวียนกันไป แบบนี้แรงบิดจะมาก

3) แบบจ่ายกระแสไฟฟ้าให้ทีละเฟสสลับกับ 2 เฟส แต่แบบนี้จำนวนขั้น (Step) จะเพิ่มขึ้นเป็น 2 เท่า ของสองแบบแรก แต่แรงบิดเฉลี่ยจะนอก

จากการจ่ายกระแสไฟฟ้าให้เฟสทั้ง 3 อย่าง เราก็สามารถสั่งให้สเต็ปปิงมอเตอร์หมุนทวนเข็มได้โดยมองการจ่ายกระแสให้กับเฟสย้อนกลับ สามารถดูได้จากตารางดังนี้

ตารางที่ ข.5 ลำดับข้อมูลที่กำหนดให้มอเตอร์หมุนในทิศทางต่าง ๆ

04	03	02	01	
0	0	0	1	01 H
0	0	1	0	02 H
0	1	0	0	04 H
1	0	0	0	08 H

ตามเข็ม จะเป็น 01 02 04 08

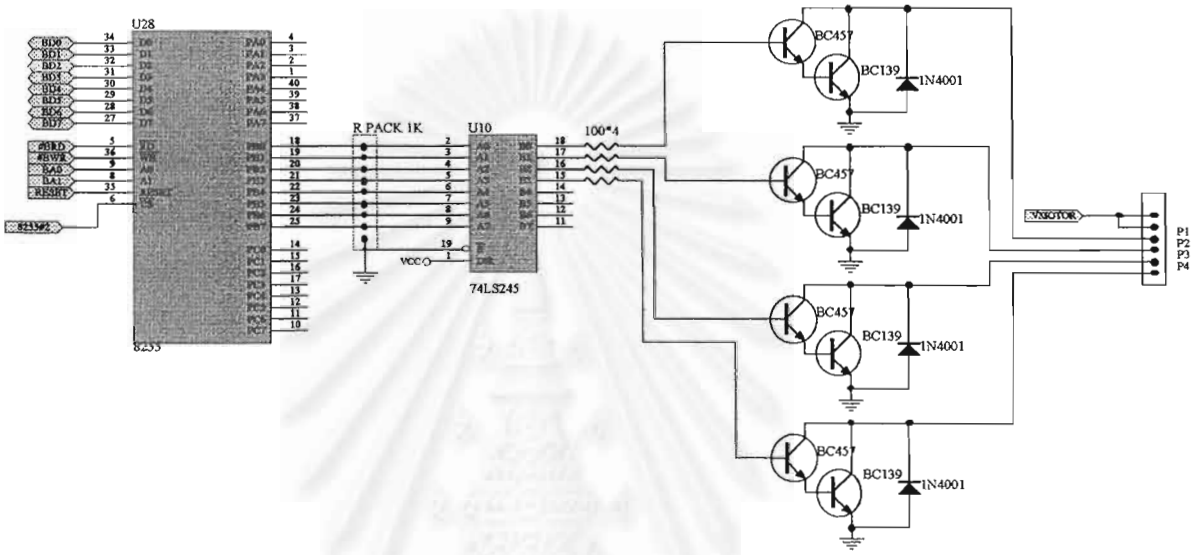
ทวนเข็ม จะเป็น 08 04 02 01

อุปกรณ์การทดลอง

1. แผงควบคุม
2. แผงทดลอง
3. แหล่งจ่ายไฟกระแสตรง
4. ลอจิกโพรบ

5. สเต็ปปิง
6. มัลติมิเตอร์

การทดลอง



รูปที่ ข.19 วงจรขับสเต็ปปิงมอเตอร์

1. ป้อนโปรแกรมดังต่อไปนี้ ทดสอบดูการทำงานของโปรแกรมใช้มิเตอร์วัดแรงดันที่เฟสหนึ่ง แล้วบันทึกไว้ จากนั้นให้บอกว่ามีการส่งข้อมูลแบบใดออกไปที่สเต็ปปิงมอเตอร์

```
5   REM STEPPING MOTOR
10   PC = 0E0B6H : CP = 0E0B7H
15   XBY (PC) = 80H  : A=0
20   FOR T=1 TO 4
40  FOR I=1 TO 50  :  NEXT I
50  NEXT T
60  DATA 3H, 6H, 0CH, 9H
```

2. ให้ดัดแปลงโปรแกรมที่ผ่านมาให้สเต็ปปิง หมุนในทิศทางตรงกันข้าม
3. เมื่อเอาต์เฟสไปทำการหน่วงเวลาแล้วจึงสั่งออฟเฟส เพื่อจุดประสงค์ใด
4. ให้เปลี่ยนข้อมูลในบรรทัดที่ 60 เป็น

   01, 02, 04, 08

   แล้วสังเกตการทำงานของโปรแกรม พร้อมทั้งบอกได้ว่าเป็นการเอาต์เฟสแบบใด

คำถามท้ายการทดลอง

1. จงอธิบายการทำงานของสเต็ปปิงมอเตอร์
2. ทำไมสเต็ปปิงจึงไม่จำเป็นที่จะต้องการการป้อนกลับไม่ว่าจะเป็นการควบคุมตำแหน่งหรือความเร็ว
3. จงบอกการนำสเต็ปปิงไปใช้งานมาซัก 5 อย่าง

การทดลองที่ 6 : การแสดงผลบนจอแอลซีดี (LCD)

วัตถุประสงค์

1) เพื่อให้รู้จักโครงสร้างของแอลซีดี
2) เพื่อให้สามารถสั่งงานแอลซีดีได้
3) เพื่อนำแอลซีดีไปประยุกต์ใช้งาน

ทฤษฎี

ปัจจุบันแอลซีดีได้ถูกนำมาใช้ในการแสดงผลมากขึ้น เนื่องจากราคาถูกลงและใช้งานง่าย ซึ่งอาจจะพบเห็นได้โดยทั่วไปในรูปของนาฬิกา, เครื่องคิดเลข, เกมส์กด จนถึงเครื่องคอมพิวเตอร์

ข้อดีของแอลซีดี

1. บางเบาสามารถพกพาได้สะดวก
2. ใช้พลังงานน้อย เช่น อาจใช้แรงดันต่ำขนาด 2 หรือ 3V และกระแสเพียงไม่กี่มิลลิแอมป์ สามารถใช้งานได้
3. ราคาถูก
4. ใช้งานในช่วงอุณหภูมิที่กว้างได้

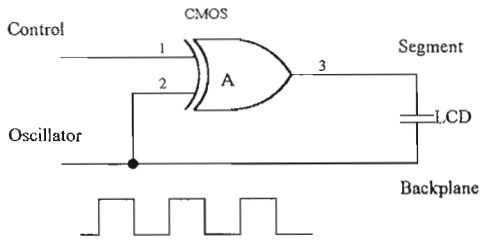
แบบต่าง ๆ ของการแสดงผล

แอลซีดีสามารถแสดงผลให้เห็นได้ โดยมีหลักการ 3 แบบ คือ

1) แบบสะท้อน (Reflective Mode) จะมีสารประเภทโลหะเคลือบอยู่ที่แผ่นหลังของแอลซีดี แบบนี้เหมาะสำหรับที่ที่มีแสงสว่างเพียงพอ
2) แบบส่งผ่าน (Transmissive Mode) แบบนี้จะวางหลอดไฟไว้ด้านบนหลัง ทำให้สามารถอ่านค่าและแสดงผลได้อย่างชัดเจน
3) แบบส่งผ่าน/สะท้อน (Tranflective Mode) เป็นการรวมกันระหว่าง 2 แบบที่ผ่านมาเข้าด้วยกัน

การขับแอลซีดี

การใช้งานไม่สามารถจ่ายแรงดันไฟตรงให้กับแอลซีดีได้ เพราะจะทำให้เกิดปฏิกริยาเคมีไฟฟ้า ด้วยเหตุนี้จะทำให้แอลซีดีมีอายุการใช้งานสั้นลง จึงจำเป็นต้องมีการป้อนสัญญาณไฟสลับให้ด้วยกำลังงานกระแสตรง โดยความถี่ต้องไม่ต่ำกว่า 30 MHz เพื่อป้องกันการกระพริบของตัวอักษรสลับเฟสของการขับโดยตรง เราสามารถใช้เกต XOR ต่อกับส่วนต่าง ๆ ได้ดังรูป



รูปที่ ข.20 วงจรแอลซีดีแบบด็อตเมทริกซ์

สามารถแบ่งได้เป็น

1. Character LCD Module
2. Graphic LCD Module
3. Segment Display Type LCD Module

โดยในแต่ละแบบนี้จะมีส่วนประกอบใหญ่ ๆ แบ่งได้เป็น

1. ด็อตเมทริกซ์แอลซีดี เป็นตัวแสดงผลให้สามารถมองเห็น ในลักษณะการปิดและเปิดตัวเองกับแสง นั่นก็คือ ส่วนของที่เป็นตัวกระจกบรรจุผลึก
2. ตัวขับเป็นตัวรับสัญญาณมาจากส่วนควบคุมเพื่อมาขับผลึกแอลซีดี
3. ตัวควบคุม เป็นตัวรับข้อมูลจากอุปกรณ์ภายนอกมาและจัดการควบคุมแอลซีดีโมดูลให้ทำงานแสดงผลต่าง ๆ เช่นการลบจอภาพ, การเกิดตัวอักษร, การเลื่อนข้อมูลบนหน้าจอแอลซีดี เป็นต้น โดยมีเบอร์ไอซีที่นิยมกันคือ HD44780 ซึ่งจะใช้ในแบบ Character LCD Module คือแอลซีดีที่แสดงข้อมูลในรูปตัวอักษร ส่วนเบอร์ HD61830 ซึ่งจะใช้ในแบบ Graphic LCD Module คือแอลซีดีที่แสดงข้อมูลในลักษณะของรูปภาพหรือตัวอักษรก็ได้

ขาต่าง ๆ ในการใช้ต่องาน HD44780

RS (Registor Selection) จะเป็นขาเลือกเรจิสเตอร์ภายในซึ่งมีอยู่ 2 ตัว คือ

Instruction Registor (IR) และ Dat Registor (DR) โดยถ้าเป็น 1 จะเป็นการเลือกข้อมูล แต่ถ้าเป็น 0 จะเป็นการเลือกโครงสร้าง

R/W (Read/Write) เป็นตัวเลือกว่าจะเขียนหรืออ่านข้อมูล

E (Enable Signal) เป็นขากำหนดสภาพการรับการเขียนหรือการอ่านข้อมูล

DB0-DB7 เป็นขารับส่งข้อมูลจากตัวไอซี

VDD ไฟเลี้ยวตัววงจร

VSS เป็นขารับกราวด์

VO เป็นขารับแรงดันในการขับแอลซีดีให้สว่างหรือมืด

ตารางที่ ข.6 ชุดคำสั่ง

POWERTIP

Instructions

Instruction	Code										Description	Executed Time(max.)
	RS	R/W	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0		
Clear Display	0	0	0	0	0	0	0	0	0	1	Clears all display and returns the cursor to the home position(Address 0).	1.64mS
Cursor At Home	0	0	0	0	0	0	0	0	1	*	Returns the cursor to the home position(Address 0). Also returns the display being shifted to the original position. DDRAM contents remain unchanged.	1.64mS
Entry Mode Set	0	0	0	0	0	0	0	1	I/D	S	Sets the cursor move direction and specifies or not to shift the display. These operations are performed during data write and read.	40 μ S
Display On/off Control	0	0	0	0	0	0	1	D	C	B	Sets ON/OFF of all display(D) cursor ON/OFF (C), and blink of cursor position character(B).	40 μ S
Cursor/Display Shift	0	0	0	0	0	1	S/C	R/L	*	*	Moves the cursor and shifts the display without changing DDRAM contents.	40 μ S
Function Set	0	0	0	0	1	DL	N	F	*	*	Sets interface data length(DL) number of display lines(L) and character font(F).	40 μ S
CGRAM Address Set	0	0	0	1	ACG						Sets the CGRAM address. CGRAM data is sent and received after this setting.	40 μ S
DDRAM Address Set	0	0	1	ADD							Sets the DDRAM address. DDRAM data is sent and received after this setting.	40 μ S
Busy Flag/ Address Read	0	1	BF	AC							Reads Busy flag(BF) indicating internal operation is being performed and reads address counter contents.	0 μ S
CGRAM/DDRAM Data Write	1	0	WRITE DATA								Writes data into DDRAM or CGRAM.	40 μ S
CGRAM/DDRAM Data Read	1	1	READ DATA								Reads data from DDRAM or CGRAM.	40 μ S

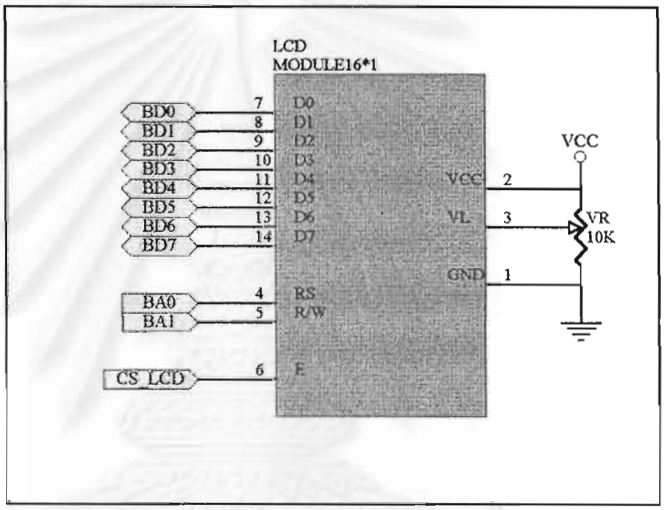
Code		Description	Executed Time(max)
I/D=1:Increment I/D=0:Decrement S=1:With display shift S/C=1:Display shift S/C=0:Cursor movement R/L=1:Shift to the right R/L=0:Shift to the left DL=1:8-bit	DL=0:4-bit N=1:2lines N=0: 1line F=1:5 × 10dots F=0:5 × 7dots BF=1:Internal operation is being performed BF=0:Instruction acceptable	DDRAM:Display Data RAM CGRAM:Character Generator RAM ACG:CGRAM Address ADD:DDRAM Address Corresponds to cursor address. AC:Address Counter used for both DDRAM and CGRAM *:Invalid	tcp or tosc=250kHz However, when frequency changes, execution time also changes Ex. If tcp or tosc is 270kHz, 40 μ S × 250/270=37 μ S

วงจรที่ใช้ในการทดลอง

สิ่งสำคัญในการใช้ LCD DOT MATRIX แบบ Character คือการเข้าใจตารางชุดคำสั่ง จากตารางข้างต้นก็สามารถใช้งานแอลซีดีได้อย่างง่ายดาย โดยที่เริ่มแรกต้องส่งรหัสควบคุมให้แอลซีดีรู้จักตัวเองเสียก่อน คือ กำหนดว่าแอลซีดีเป็นแบบกี่เส้นและการรับข้อมูลเป็นแบบ 4 บิตหรือ 8 บิต ให้มีหรือไม่ และเมื่อมีการเขียนตัวอักษรไปที่แอลซีดีแล้วให้ตำแหน่งตัวอักษรที่จะเขียนใหม่เพิ่มขึ้นหรือไม่ เป็นต้น ส่วนการอ่านหรือการเขียนนั้นเหมือนการส่งข้อมูลไปยังพอร์ตคือ ให้คำสั่งเขียนไปยังพอร์ตเขียน และใช้คำสั่งอ่านไปยังพอร์ตอ่าน ก็จะได้ข้อมูลมายังซีพียูหรือข้อมูลที่ถูกเขียนไปปรากฏที่หน้าจอแอลซีดี โดยจากวงจรทดลอง สามารถกำหนดตำแหน่งพอร์ตในการติดต่อกับแอลซีดีได้ดังนี้

RS = A1	RW = A0	ตำแหน่งพอร์ต	หน้าที่ของพอร์ต
0	0	0ACH	เขียนคำสั่งของแอลซีดี
0	1	0ADH	อ่าน Busy flag
1	0	0AEH	เขียนข้อมูล
1	1	0AFH	อ่านข้อมูล

การส่งรหัสคำสั่งหรือข้อมูลไปให้แอลซีดีจะต้องตรวจสอบความพร้อมของแอลซีดีเสียก่อน โดยการอ่าน Busy flag ถ้าบิต Busy flag นี้เป็น 1 อยู่ จะไม่สามารถส่งข้อมูลไปที่แอลซีดีได้ เพราะแอลซีดีกำลังทำงานภายในอยู่ ต้องรอจนกว่าบิต Busy นี้จะมีค่าเป็น 0 ถึงจะส่งข้อมูลไปที่แอลซีดีได้



รูปที่ ข.21 วงจรแอลควบคุมซีดี

อุปกรณ์การทดลอง

1. แผงควบคุม
2. แผงทดลอง
3. แหล่งจ่ายไฟกระแสตรง

การทดลอง

1. ป้อนโปรแกรมดังต่อไปนี้ สังเกตดูการทำงานของโปรแกรมและหน้าจอแอลซีดีในชุดทดลอง

```
6   REM LCD_DISPLAY : A=0FFH : B=0
11  COM = 0E0ACH : BUSY = COM+1 : WR = CMO+2
20  GOSUB 120
30  FOR I=1 TO 6            ; ตัวอักษร 6 ตัว
40  READ B                  ; อ่านข้อมูล
```

```
50  XBY (WR) = B              ; เขียนไปยังแอลซีดี
60  GOSUB  200
70  NEXT I
80  RESTORE
90  END
100 DATA 41H, 42H, 43H, 44H, 45H, 46H
120 XBY (COM) = 30H           ; ตั้งเงื่อนไขแอลซีดี 8 บิต 1 บรรทัด
130 GOSUB  200
140 XBY (COM) = 0FH           ; เปิดหน้าจอแอลซีดี
150 GOSUB  200
160 XBY (COM) = 6             ; เพิ่มค่าในแรมขึ้น
170 GOSUB  200
180 XBY (COM) = 1             ; ลบล้างหน่วยความจำ
190 GOSUB  200
200 RETURN
210 A= XBY (BUSY)             ; ตรวจสอบสถานะของแอลอีดี
220 IF A > 80H THEN GOTO 200
230 RETURN
```

2. ให้ดัดแปลงโปรแกรมคำสั่งคำว่า "HELLO" ออกหน้าจอ
3. ให้ดัดแปลงโปรแกรมคำสั่งคำว่า "HELLO" อยู่ตรงกลางหน้าจอ
4. อธิบายการทำงานของโปรแกรมในข้อ 1

คำถามท้ายการทดลอง

1. การตรวจสอบความพร้อมของแอลอีดี ถ้าเราไม่อ่าน BUSY FLAG แต่ใช้การหน่วงเวลาแทนได้หรือไม่ เกิดผลดีหรือผลเสียอย่างไร
2. จงบอกข้อดีและข้อเสียของการใช้แอลอีดีแบบ 7 ส่วน เปรียบเทียบกับแอลซีดี

การทดลองที่ 7 : การแปลงผันสัญญาณดิจิทัลเป็นแอนะล็อก และแอนะล็อกเป็นดิจิทัล

<u>วัตถุประสงค์</u>

1) เพื่อให้เข้าใจเกี่ยวกับการเปลี่ยนสัญญาณดิจิทัลให้เป็นสัญญาณแอนะล็อก
2) เพื่อเป็นแนวทางในการประยุกต์ใช้งานจริง

ทฤษฎี

ไมโครโพรเซสเซอร์จะมีการทำงานในเชิงดิจิทัลเสมอซึ่งมีระดับสัญญาณเป็น 0 และ 1 แต่ในการประยุกต์ใช้งานจริงมักจะต้องเกี่ยวข้องกับสัญญาณที่เป็นแอนะล็อก ซึ่งสัญญาณที่มีความแตกต่างหลายระดับสัญญาณเหล่านี้ได้แก่ อุณหภูมิ, ความดัน, น้ำหนักเสียง ในทางอิเล็กทรอนิกส์สามารถเปลี่ยนแรงดันให้อยู่ในรูปของสัญญาณเหล่านี้ได้ และระบบคอมพิวเตอร์ต้องเปลี่ยนสัญญาณ 0 และ 1 ให้เป็นสัญญาณแรงดันได้เช่นกัน ซึ่งสัญญาณ 0 และ 1 นี้ ไมโครจะอยู่ในรูปของข้อมูลทางคอมพิวเตอร์

คุณสมบัติทั่วไปของ D/A จะกล่าวเป็นข้อ ๆ ดังนี้

1. Resolution คือ ความสามารถในการแบ่งแยกระดับของสัญญาณหรืออีกนัยหนึ่งก็คือจำนวนบิตของสัญญาณดิจิทัลนั่นเอง เช่น ขนาด 8 บิต จะหมายความว่า สามารถแยกสัญญาณแอนะล็อกไว้เป็น 256 ระดับ (2^8) ถ้าขนาด 12 บิต ก็จะแยกสัญญาณได้เป็น 4096 ระดับ (2^12) ค่า Resolution บางครั้งก็จะแสดงในรูปของเปอร์เซ็นต์ (%) โดยขนาด 8 บิต จะมีค่าเท่ากับ 100/256 คือ 0.39%

2. Full Scale Output Voltage คือ ค่าแรงดันสูงสุดที่จะเป็นไปได้ของสัญญาณแอนะล็อก สมมติว่าใช้ D/A ขนาด 8 บิต เราจะเขียนเป็นสูตรได้ดังนี้

$$VO = Vref\left[\left(\frac{A1}{2}\right)+\left(\frac{A2}{4}\right)+\left(\frac{A3}{8}\right)+\left(\frac{A4}{16}\right)+\left(\frac{A5}{32}\right)+\left(\frac{A6}{64}\right)+\left(\frac{A7}{128}\right)+\left(\frac{A8}{256}\right)\right]$$

A1 ถึง A8 คือ สัญญาณดิจิทัลมีค่าเป็น 0 หรือ 1 ตามข้อมูลในที่นี้ถ้าสมมติ Vref = 10V ค่าสูงสุดที่เป็นไปได้คือ ให้ A1 ถึง A8 = 1 ดังนี้

$$VO = 10\left(\frac{255}{256}\right) = 9.9609V$$

นั่นหมายความว่า สัญญาณแอนะล็อกจะมีค่าสูงสุดน้อยกว่า Vref อยู่ 1 ระดับเสียง และในที่นี้ความแตกต่างของแต่ละระดับจะเท่ากับ

$$Vdiff = \frac{10}{256} = 0.03906V$$

3. Accuracy คือ ค่าเปรียบเทียบระหว่าง Volt จริง ๆ ที่ได้กับ VoH ที่กำหนดให้เป็น ถ้า D/A มีคุณสมบัติเป็น 10V และ + 0.15% Accuracy นั่นหมายถึง D/A ชนิดนี้จะมีโอกาสผิดพลาดได้สูงคือ 0.0015*10 = 0.015 V ปกติค่า Accuracy ในทางจุดปกติแล้วไม่ควรจะมีค่ามากกว่าครึ่งหนึ่งของระดับสัญญาณแอนะล็อกหรือเท่ากับ + ½ ของบิตที่ต่ำที่สุด (LSB) ในที่นี้คือ 0.039/2 = 0.0195 V

4. Linearity คือ ค่าความคงที่ในการเปลี่ยนระดับสัญญาณมีหน่วยเป็นเปอร์เซ็นต์ (%) ปกติ D/A ควรจะมีค่าระดับเปลี่ยนแปลงเป็นเส้นตรง แต่ในความจริงอาจจะมีการเบี่ยงเบนเกิดขึ้นได้ในทางจุดปกติไม่ควรมีค่ามากกว่า + ½ LSB ซึ่งก็คือค่าเปอร์เซ็นต์ (%) ของ Resolution หารด้วย 2 คือ 0.39122 = 0.019%

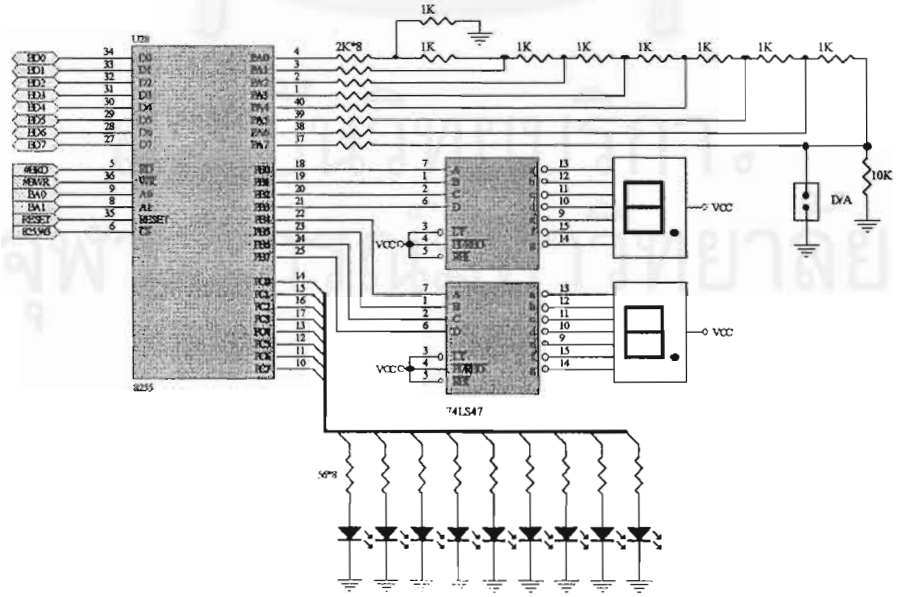
5. Settling Time คือ ค่าเวลาที่ใช้ในการเปลี่ยนระดับสัญญาณเท่ากับ ½ ตัวอย่างเช่น ไอซีเบอร์ DAC 0809 จะมีค่าเท่ากับ 150ns ค่านี้มีความสำคัญต่อความเร็วในการเปลี่ยนหรืออีกนัยหนึ่งก็คือ ความถี่ของสัญญาณแอนะล็อกนั่นเอง

วงจรพื้นฐานของ D/A จะมีอยู่ 2 แบบคือ Weighted – Resistor D/A และ R-2R D/A ในการทดลอง อย่างไรก็ตามในปัจจุบัน เรามักจะใช้ไอซีสำเร็จรูป ซึ่งง่ายต่อการใช้งาน และมีประสิทธิภาพดี

อุปกรณ์ที่ใช้ในการทดลอง

1. แผงควบคุม
2. แผงทดลอง
3. แหล่งจ่ายไฟกระแสตรง
4. มัลติมิเตอร์
5. สโคป

การทดลอง



รูปที่ ข 22 วงจรที่ใช้ทดลอง

วงจร D/A ซึ่งจะใช้พอร์ต A ของ 8255 เป็นวงจรดิจิทัล โดยพอร์ต A มีตำแหน่งอยู่ที่ 0E0B8H และพอร์ตควบคุม อยู่ที่ 0E0B8H

1. ป้อนโปรแกรมต่อไปนี้ แล้วนำมัลติมิเตอร์วัดแรงดันที่เอาต์พุตแล้วบันทึกผล

```
5   REM D/A
10  PA= 0E0B8H : CP = 0E0BBH
20  XBY (CP) = 80H   : A=0
30  XBY (PA) = A
RUN
```

2. เปลี่ยนค่า A จากโปรแกรมให้เป็น 1 และ 0FFH แล้วนำค่ามิเตอร์มาวัดแรงดันเปรียบเทียบแรงดันกับ A=0 บันทึกผล

3. ให้เขียนโปรแกรมกำเนิดรูปสัญญาณสามเหลี่ยม แล้วนำสโคปวัดรูปคลื่นสัญญาณที่เอาต์พุตแล้ววัดสัญญาณที่ได้ประกอบ

4. ดัดแปลงโปรแกรมโดยให้ความถี่ที่เอาต์พุตต่ำลง

การแปลงผันสัญญาณแอนะล็อกเป็นดิจิทัล

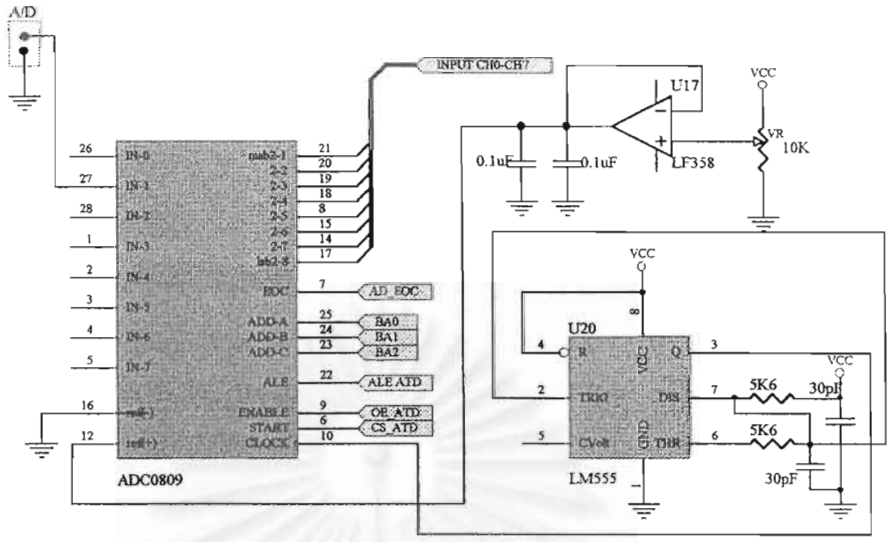
วัตถุประสงค์

1) เพื่อให้เข้าใจเกี่ยวกับการเปลี่ยนสัญญาณแอนะล็อกให้เป็นสัญญาณดิจิทัล
2) เพื่อเป็นแนวทางในการประยุกต์ใช้งานจริง

ทฤษฎี

ลักษณะทั่วไปของ D/A และ D/A จะมีความเกี่ยวข้องกันเป็นอย่างมากเมื่อเราสามารถนำสัญญาณดิจิทัลเปลี่ยนเป็นแอนะล็อกได้แล้ว ในทางกลับกันเราก็ควรจะเปลี่ยนสัญญาณแอนะล็อกเปลี่ยนเป็นสัญญาณดิจิทัลได้ด้วย ซึ่งในจุดนี้เราอ่านข้อมูลทางแอนะล็อกได้เช่น ค่าความดัน, ค่าอุณหภูมิ โดยจะมีประโยชน์อย่างมากต่องานทางด้านวิศวกรรม

คุณสมบัติของ A/D จะมีลักษณะเหมือนกับ D/A และจะมีคุณสมบัติพิเศษอีก 1 อย่างคือช่วงเวลาแปลงผัน ดังนี้หมายถึงช่วงเวลา A/D ใช้ในการแปลงสัญญาณแอนะล็อกเป็นดิจิทัลได้ 1 ค่า A/D ที่ดีควรมีช่วงเวลาแปลงผันน้อยที่สุดเท่าที่จะน้อยได้



รูปที่ ข.23 วงจร A/D

ในรูปที่ ข.23 วงจร A/D จะใช้ IC ADC0808 ซึ่งการเชื่อมต่อเหมือนกับอุปกรณ์ไอโอทั่วไป โดยมีหมายเลขพอร์ตดังนี้

0E0A4H - 0E0ABH เป็นพอร์ตเริ่มต้น (Start) ของการแปลงผัน (Convert) ช่อง 0-7 ตามลำดับ

0E0B2H พอร์ต C ของ 8255 U25 (PC7) อ่าน EOC

0E0A1H พอร์ตสำหรับอ่านค่าที่ได้จากการแปลงผัน

การติดต่อกับ A/D ทำได้ดังนี้

1. เขียนข้อมูลไปที่ช่องสัญญาณที่ต้องการ (เพื่อให้เริ่มต้นการแปลงผัน)
2. อ่านบิต EOC ว่า เป็น 1 หรือไม่ (1= END OF CONVERSION)
3. อ่านไปที่พอร์ต 0E0A1H (ได้ค่าดิจิทัลที่แปลงผัน)

อุปกรณ์การทดลอง

1. แผงควบคุม
2. แผงทดลอง
3. แหล่งจ่ายไฟกระแสตรง

การทดลอง

1. นำแผงทดลองมาแล้วนำมิเตอร์วัดที่ขา 27 ของไอซี ADC0809 ปรับค่าแรงดันให้ต่ำที่สุด จากนั้นป้อนโปรแกรมแล้วบันทึกค่าที่ได้ และปรับให้ค่าแรงดันที่ขา 27 สูงสุดแล้วทำการรำเนินการโปรแกรมและบันทึกผลการทดลอง โปรแกรมตัวอย่างดังนี้

```
10  REM A/D
20  AD_ST = 0E0A9H : AD_OE = 0E0A1H : EOC = 0E0B2H : CP = 0E0B3H
30  XBY(CP) = 9BH
40  XBY(AD_ST) = 1
50  A = XBY(EOC)
60  IF A < 80H THEN GOTO  50
70  B = XBY(AD_OE)
80  PH0.B
```

2. ให้เขียนโปรแกรมอ่าน A/D ทางช่องที่ 1 แล้วนำค่าที่ได้ส่งออก D/A แล้วเปรียบเทียบค่าอินพุตและเอาท์พุตตรงกันหรือไม่
3. วาดรูปคลื่นสัญญาณอินพุตและเอาท์พุตเพื่อเปรียบเทียบกัน

คำถามท้ายการทดลอง

1. Conversion Time ของ A/D หมายถึงอะไร
2. วงจร A/D ที่ใช้กันอยู่ทั่วไปในงานด้านไมโครคอนโทรลเลอร์ คือวงจรแบบไหน
3. ให้ยกตัวอย่างการนำวงจร A/D ไปประยุกต์ใช้งาน 5 อย่าง

ภาคผนวก ค

คู่มือการเขียนโปรแกรมภาษาเบสิก

ลักษณะภาษา/แนะนำภาษา

ภาษาระดับสูงเบสิก (BASIC) ย่อมาจากคำว่า "Beginner 's All purpose Symbolic Instruction Code" เป็นภาษาระดับสูงที่ใกล้เคียงกับภาษาอังกฤษในชีวิตประจำวันซึ่งเหมาะสำหรับการแก้ปัญหาทางด้านวิทยาศาสตร์และคณิตศาสตร์

ปัจจุบันภาษาเบสิก ได้ถูกพัฒนาไปเป็น Visual Basic ที่ใช้งานบนวินโดวส์ (Windows) เวอร์ชันต่าง ๆ สามารถใช้ในการเขียนโปรแกรมประยุกต์ต่าง ๆ หรือคอมไพเลอร์

ณ ที่นี้ จะอธิบายเน้นการใช้งานภาษาเบสิก ในงานการควบคุมโดยอิงกับโปรแกรม MCS-51 BASIC-52 ซึ่งพูดถึงโครงสร้างของภาษาเบสิกและฟังก์ชันต่าง ๆ ที่เป็นประโยชน์ในการเขียนโปรแกรม โดยสมมติว่าผู้อ่านมีความรู้ทางด้านการเขียนโปรแกรมมาบ้างแล้ว

รูปแบบของภาษาเบสิก

[หมายเลขบรรทัด] [คำสั่ง] [ตัวแปร, คำสั่ง, ตัวกระทำทางคณิตศาสตร์, ค่าคงที่]

ตัวอย่าง

```
10 LET A=1
20 LET B=2
30 C = A+B
40 PRINT C
50 END
```

หมายเลขบรรทัด (Line Number)

คำสั่งทุกบรรทัดในภาษาเบสิกจะต้องเริ่มต้นด้วยหมายเลขบรรทัด ด้วยการมีหมายเลขนี้ทำให้เครื่องสามารถรับคำสั่งตามลำดับหมายเลขบรรทัดจากน้อยไปหามาก นอกจากนี้หมายเลขบรรทัดยังใช้อ้างอิงคำสั่งควบคุมและยังใช้ระบุหมายเลขบรรทัดที่ผิดเพื่อให้แก้ไขให้ถูกต้องต่อไป

ในการเขียนหมายเลขบรรทัดจะต้องเขียนเป็นจำนวนเต็มเสมอช่วงตั้งแต่ 0 ถึง 65535 และหมายเลขบรรทัดจะใช้เพียงครั้งเดียวเท่านั้น ใน 1 หมายเลขบรรทัดสามารถเขียนให้รับคำสั่งได้หลายคำสั่ง โดยใส่เครื่องหมาย (:) ข้างหน้าคำสั่งที่ต้องการเพิ่มเข้าไป

คำสั่ง (Command)

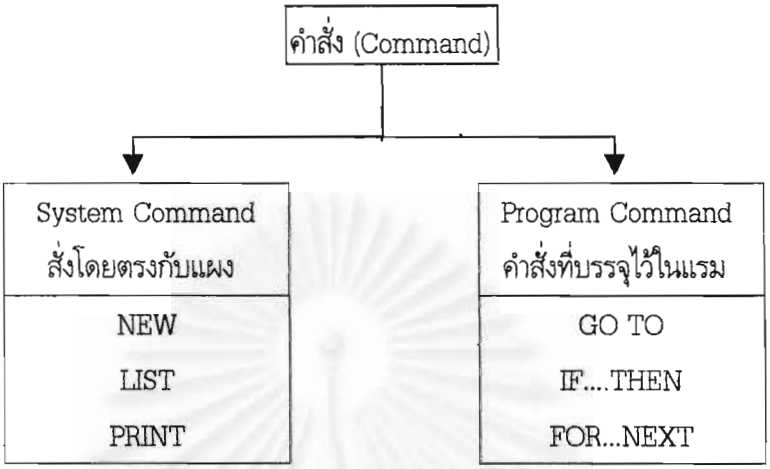
มี 2 แบบ คือ คำสั่งของระบบเป็นคำสั่งให้เครื่องทำงานโดยตรง โดยไม่ต้องมีหมายเลขบรรทัดนำหน้า

NEW - ลบล้างหน่วยความจำแรมและเริ่มต้นเขียนโปรแกรมใหม่

LIST - ให้แสดงโปรแกรมที่เขียนไว้ในแรม

PRINT - แสดงผลออกมาเป็นเลขฐาน 10

และอีกแบบหนึ่งคือ โปรแกรมคำสั่ง เป็นคำสั่งที่บรรจุไว้ในโปรแกรมแล้วให้เครื่องปฏิบัติตามที่วางไว้ตามลำดับ เมื่อสั่งให้เริ่มทำงานตามโปรแกรม เช่น GO TO.., IF.....THEN, FOR....NEXT, INPUT... เป็นต้น



รูปที่ ค.1 ประเภทของคำสั่งในภาษาเบสิก

ค่าคงตัว (Constants) แบ่งออกเป็น 2 ประเภท คือ ค่าคงตัวเป็นตัวเลข (Numeric Constants) และค่าคงตัวไม่ใช่จำนวนเลขเรียกว่า (String Constants) เป็นทั้งค่าบวกและค่าลบ มี 2 ชนิด คือ

1) จำนวนเต็ม (Integer) ได้แก่ 55, -100, 0A000H, 0

2) จำนวนทศนิยม (Floating) ได้แก่ 46.8, -36.79, 7E-09

อย่างไรก็ตาม ค่าคงตัวจะอยู่ในช่วงระยะ (Range) ตั้งแต่ -1E-127 ถึง 0.99999999 E+127 ในภาษาเบสิกนี้จะเขียนเลขยกกำลังได้ดังนี้

2^2 = 2**2

2^I = 2**I

และค่าคงตัวอีกแบบหนึ่งคือ ค่าคงตัวสายอักขระ (String Constants) ได้แก่ ค่าคงตัวเป็นข้อความแตกต่างที่ไม่ใช่การคำนวณอาจจะประกอบด้วยตัวเลข, ตัวอักษรและสัญลักษณ์พิเศษต่าง ๆ เช่น

" EDL LAB "

" PONG82@CHAIYOMAIL.COM "

ตัวแปร (Variables)

ตัวแปร คือ ชื่อหน่วยความจำในคอมพิวเตอร์ ซึ่งจะทำหน้าที่ในการเก็บค่าต่าง ๆ เพื่อใช้ในการทำงานซึ่งตัวแปรแบ่งออกเป็น 2 ชนิด คือ

- ตัวแปรที่ใช้แทนตัวเลข (Numeric Variables) คือ ตัวแปรที่ใช้แทนตัวเลขที่ใช้ในการคำนวณการตั้งชื่อตัวแปรชนิดนี้จะต้องขึ้นต้นด้วยตัวอักษรตัวเดียว เช่น A, B,....., Z หรือมีตัวอักษรห้อยท้าย ได้แก่ A=1, Z = 0FFH
- ตัวแปรที่ใช้แทนตัวอักษร (String Variables) คือ ตัวแปรที่ไม่ใช่ตัวเลขและไม่สามารถนำไปใช้ในการคำนวณได้ การตั้งชื่อตัวแปรชนิดนี้จะต้องมีเครื่องหมาย $ ตามท้ายอยู่เสมอ เช่น

  $ (1) = " EDL_LAB "

  $ (A) = " HELLO "

รายละเอียดของคำสั่ง

- คำสั่ง RUN

หลังจากมีคำสั่ง RUN ถูกตามด้วย Enter โปรแกรมที่เก็บในแรมมอนิเตอร์จะถูกให้ทำงาน (Execute) ตามลำดับบรรทัด โดยเริ่มจากบรรทัดแรก สามารถเลือกดำเนินการที่บรรทัดใดก็ได้โดยพิมพ์

> RUN 100

คือ ดำเนินการเริ่มจากบรรทัดที่ 100

- คำสั่ง CONT

ถ้าโปรแกรมถูกพักการทำงานด้วยคำสั่ง BRKP เราสามารถสั่งให้โปรแกรมทำงานต่อหลังจากพักการทำงานได้ โดยพิมพ์ CONT ได้เมื่อเกิดการผิดพลาดในขณะที่พักการทำงาน

```
> 10 FOR I=1 TO 10
> 20 PRINT I
> 25 BRKP
> 30 NEXT I
  1
STOP - IN LINE 20
READY
> PRINT I
  1
> CONT
  2
```

- คำสั่ง LIST (CR)

คำสั่ง LIST เป็นคำสั่งที่ใช้พิมพ์โปรแกรมที่เราเก็บไว้ในแรม เราสามารถจะใช้คำสั่งนี้ช่วยในการแก้จุดบกพร่อง ในการดูคำสั่งที่พิมพ์ลงไป รูปแบบการใช้งานมีดังนี้

> LIST [บรรทัด]

> LIST [บรรทัด]-[บรรทัด]

ตัวอย่าง

```
READY
> LIST
10 PRINT "LOOP PROGRAM"
20 FOR I=1 TO 3
30 PRINT I
40 NEXT I
50 END
READY
> LIST
30 PRINT I
40 NEXT I
50 END
READY
> LIST 20-40
20 FOR I=1 TO 3
30 PRINT I
40 NEXT I
```

- คำสั่ง NEW (CR)

เมื่อคำสั่ง NEW ถูกพิมพ์ลงไป MCS BASIC-52 จะลบล้างโปรแกรมที่เก็บอยู่ภายในแรมมอเตอร์ทั้งหมดและตั้งค่าตัวแปรทุกตัวเท่ากับ 0 สายอักขระจะร้องขอการขัดจังหวะเพื่อลบล้างค่าในสัญญาณนาฬิกาตามเวลาจริง (REAL TIME CLOCK) คำสั่ง NEW จะลบล้างโปรแกรม, ตัวแปร และสายอักขระในแรม

- คำสั่ง RAM และคำสั่ง ROM

รูปแบบคำสั่ง 2 คำสั่งเหล่านี้จะบอก MCS BASIC-52 ให้เลือกดูโปรแกรมจากหน้าจอแอลซีดี โดยแรมจะเริ่มที่ตำแหน่ง 512(200H) ส่วนตำแหน่งโปรแกรมในรอม (ROM) จะเริ่มต้นที่ 8010H

RAM

เมื่อ RAM ถูกกด Enter MCS BASIC-52 จะเลือกบรรจุโปรแกรมจากหน่วยความจำในแรม ROM

เมื่อ ROM ถูกกด Enter MCS BASIC-52 จะเลือกบรรจุโปรแกรมจาก NVRAM ถ้าไม่มีค่าจำนวนจริงหลังคำสั่ง ROM เช่น > ROM จะบรรจุโปรแกรม ROM1 โดยคำสั่ง ROM สามารถเลือกบรรจุได้ตามลำดับที่บันทึกลงใน NVRAM แต่จะไม่สามารถแก้ไขโปรแกรมได้เนื่องจากถูกเก็บใน NVRAM จะเกิดข้อความผิดพลาดขึ้น คือ ERROR : PROM MODE เพราะฉะนั้นผู้เรียนรู้จะต้องนำโปรแกรมจาก NVRAM มาไว้ใน RAM เสียก่อน โดยใช้คำสั่ง LOAD

- คำสั่ง LOAD

คำสั่ง LOAD จะบรรจุโปรแกรมลงใน NVRAM ไปยัง RAM ผู้เรียนรู้จะต้องเปลี่ยนแบบวิธีการทำงานโดยใช้คำสั่ง RAM แบบวิธีนี้สามารถแก้ไขเปลี่ยนแปลงค่าในโปรแกรมได้

- คำสั่ง PROG

คำสั่งนี้จะเป็นคำสั่งเฉพาะที่ใช้สำหรับบันทึกข้อมูลจากแรมลงไปที่เอ็นวีแรม

<u>ตัวอย่าง</u>

```
> LIST
10 FOR I=1 TO 10
20 PRINT I
30 NEXT I
READY
> PROG
READY
> ROM
READY
> LIST
> 10 FOR I=1 TO 10
> 20 PRINT I
> 30 NEXT I
```

- คำสั่ง CLEARI

คำสั่ง CLEARI (Clear Interrupt) เป็นคำสั่งที่ใช้ลบล้างบิต 2 และบิต 3 ของเรจิสเตอร์พิเศษ (IE) ในไมโครคอนโทรลเลอร์ MCS-51 หลังจากกระทำคำสั่ง ONTIME และ ONEX1

■ คำสั่ง CLOCK1 และ CLOCK0

คำสั่ง CLOCK1 เป็นคำสั่งที่อนุญาตให้ตัวจับเวลา/ตัวนับ 0 (Timer/ Counter 0) แบบวิธี 13 บิต หรือแบบวิธี 0 สามารถเกิดการขัดจังหวะได้ทุก ๆ 5 มิลลิวินาที

MCS BASIC-52 จะคำนวณค่าที่ใช้ในการกำหนดบรรจุซ้ำอย่างอัตโนมัติ หลังจากมีการกำหนดค่า XTAL ลงไปแล้ว ถ้าไม่ได้กำหนดค่าจากผลึก MCS BASIC-52 จะใช้ค่า 11.0592 MHz. โดยตัวนับจะนับจาก 0 ถึง 65535.995 วินาที

คำสั่ง CLOCK0 (ศูนย์) เป็นคำสั่งให้หยุดการทำงาน ตัวจับเวลา/ตัวนับ 0 (Timer/ Counter 0) โดยคำสั่งนี้จะไปลบล้างบิตที่ 2 ของเรจิสเตอร์ (IE) ค่าในสัญญาณนาฬิกาตามเวลาจริง (REAL TIME CLOCK)

■ คำสั่ง DATA – READ – RESTORE

คำสั่ง DATA เป็นคำสั่งที่ใช้สำหรับกำหนดข้อมูลเพื่อทำการอ่านทีละ 1 ไบต์ ถ้าหลาย ๆ ไบต์ จะต้องมีเครื่องหมายจุลภาค (,) คั่นกลาง

คำสั่ง READ เป็นคำสั่งที่ใช้สำหรับอ่านข้อมูลทีละบิตจากการกำหนดค่าในคำสั่ง DATA เข้ามาประมวลผล ถ้ามากกว่า 1 ตัวแปร จะต้องมีเครื่องหมายจุลภาค (,) คั่นกลางเสมอ

คำสั่ง RESTORE เมื่อ MCS BASIC-52 พบคำสั่งนี้จะทำการตั้งค่าใหม่ ตัวบ่งชี้ภายในจะเริ่มอ่านข้อมูลจากตำแหน่งเริ่มต้นของไบต์ เริ่มที่ถูกกำหนดโดยคำสั่ง DATA

ตัวอย่าง

```
> 10 FOR I=1 TO 3
> 20 READ A, B
> 30 PRINT I
> 40 NEXT I
> 50 RESTORE
> 60 READ A, B
> 70 PRINT A, B
> 80 DATA 10, 20, 10/2, 20/2, SIN(PI), COS(PI)
> RUN
  10   20
   5   10
   0   -1
  10   20
```

■ คำสั่ง DO - UNTIL

คำสั่ง DO - UNTIL เป็นคำสั่งที่ใช้ควบคุมแบบวงแหวนภายในโปรแกรม MCS BASIC-52 สำหรับสถานะระหว่าง DO และ UNTIL จะกระทำคำสั่งจนกระทั่งสถานะตัวแปรนั้นเป็นจริง

ตัวอย่าง DO - UNTIL

```
> 10 A= 0
> 20 DO
> 30 A=A+1
> 40 PRINT A
> 40 PRINT A
> 50 UNTIL A= 4
> 60 PRINT " DONE "
> RUN
  1
  2
  3
  4
  DONE
  READY
>
```

■ คำสั่ง DO - WHILE

คำสั่ง DO - WHILE ลักษณะการทำงานคล้ายกับ DO - UNTIL ข้างต้น แต่เงื่อนไขของคำสั่ง DO - WHILE ที่นำมาเปรียบเทียบจะเป็นมากกว่าและน้อยกว่า ซึ่งต่างกับ DO - UNTIL

ตัวอย่าง DO - WHILE

```
> 10 DO
> 20 A=A+1
> 30 PRINT A
> 40 WHILE A< 4
> 50 PRINT " DONE "
> RUN
  1
  2
```

```
 3
 4
 DONE
 READY
```

- คำสั่ง END

คำสั่ง END เป็นคำสั่งที่ใช้สำหรับบอกให้ MCS BASIC-52 ทราบว่าจบโปรแกรม คำสั่ง CONT จะไม่สามารถใช้งานได้ หลังจากการกระทำคำสั่ง END

ตัวอย่าง END

```
> 10 FOR I=1 TO 4
> 20 GOSUB 100
> 30 NEXT I
> 40 END
> 100 PRINT I
> 110 RETURN
> RUN
 1
 2
 3
 4
 READY
>
```

- คำสั่ง FOR - TO - STEP - NEXT

คำสั่งนี้เป็นคำสั่งที่ใช้ในการกำหนดขอบเขตในการวนวงแหวนของ MCS BASIC-52 โดยสามารถกำหนดการเริ่มต้นและสิ้นสุดได้ คำสั่ง NEXT เป็นคำสั่งที่สั่งให้โปรแกรมทำงานวนเป็นวงแหวนต่อไป โดยเพิ่มค่าขึ้นตามลำดับที่กำหนดไว้ในโปรแกรม

ตัวอย่าง FOR - TO - STEP - NEXT

```
> 10 FOR I= 0 TO 8 STEP 2
> 20 PRINT I
> 30 NEXT I
> RUN
  0
```

```
 2
 4
 6
 8
READY
```

■ คำสั่ง GOSUB [เลขบรรทัด] – RETURN

คำสั่ง GOSUB เป็นคำสั่งที่ใช้กระโดดข้ามการทำงานไปยังบรรทัดที่กำหนด เหมือนคำสั่ง CALL ในภาษาแอสเซมบลี เมื่อกระโดดมายัง Sub Routine สามารถกำหนดให้กระโดดกลับไปยังบรรทัดที่มาได้ โดยใช้คำสั่ง RETURN

ตัวอย่าง SUBRUTINE

```
> 10 FOR I=1 TO 4
> 20 GOSUB 100
> 30 NEXT I
> 100 PRINT I
> 110 RETURN
> RUN
 1
 2
 3
 4
 5
 READY
>
```

■ คำสั่ง GO TO [เลขบรรทัด]

คำสั่ง GO TO เป็นคำสั่งที่ใช้กำหนดการกระโดดไปกระทำโปรแกรมไปยังบรรทัดต่าง ๆ โดยการกระโดดกลับ คำสั่งนี้คล้ายกับคำสั่ง IMP ในภาษาแอสเซมบลีนั่นเอง

ตัวอย่าง GO TO

```
> 10 FOR I=1 TO 4
> 20 GO TO 100
> 30 NEXT I
```

```
> 100 PRINT I
> 110 GO TO 30
> RUN
 1
 2
 3
 4
 READY
>
```

- คำสั่ง IF – THEN – ELSE

คำสั่ง IF – THEN – ELSE เป็นคำสั่งที่ใช้ตั้งเงื่อนไขในการทดสอบการทำงานของ MCS BASIC-52 ว่าเป็นจริงหรือเท็จ โดยคำสั่ง IF เป็นการตั้งเงื่อนไข คำสั่ง THEN คือเงื่อนไขของสถานะโปรแกรมที่เป็นจริง คำสั่ง ELSE คือเงื่อนไขของสถานะโปรแกรมที่เป็นเท็จ

ตัวอย่าง

```
> 20 IF INT (A) < 10 THEN GOTO 100 ELSE 200
```

- คำสั่ง INPUT

คำสั่ง INPUT เป็นคำสั่งที่อนุญาตให้ผู้ใช้งานป้อนข้อมูลทางแผงแป้นอักขระระหว่างการดำเนินโปรแกรม ผู้ใช้งานสามารถกำหนดตัวแปรได้มากกว่า 1 ใน 1 คำสั่ง INPUT แต่ต้องมีเครื่องหมายจุลภาค (,) คั่นกลางเสมอ

ตัวอย่าง

```
> 10 INPUT " ENTER A NUMBER - ", A
> 20 PRINT SQR (A)
> RUN
   ENTER A NUMBER – 100
10
```

- คำสั่ง LET

คำสั่ง LET คำสั่งนี้ใช้ในการกำหนดค่าให้กับตัวแปรต่าง ๆ ดังตัวอย่าง

ตัวอย่าง LET

```
10 > LET A = 10* SIN (B)/100
20 > LET $ (1) = " THIS IS A STRING "
```

■ คำสั่ง ONEX1 [เลขบรรทัด]

คำสั่ง ONEX1 เป็นคำสั่งที่กำหนดให้ซีพียูตอบสนองการขัดจังหวะที่ INT1 โดยคำสั่งนี้จะตั้งค่าบิต ที่ 2 และ 7 ของเรจิสเตอร์ IF เป็น 1 เพื่อรอการขัดจังหวะ การใช้งานจะต้องกำหนดเลขบรรทัดหลังคำสั่ง เพื่อบอกว่าเมื่อใดมีการขัดจังหวะที่ INT1 จะให้กระโดดไปทำงานบรรทัดใด

ตัวอย่าง ONEX1

```
> 10 ONEX1 100
```

■ คำสั่ง ONTIME

คำสั่ง ONTIME เป็นคำสั่งที่ใช้เป็นตัวนับ คำสั่งนี้จะใช้ร่วมกับคำสั่ง CLOCK1 ดังที่กล่าวมาแล้ว และตัวดำเนินการ TIME โดยรูปแบบคำสั่งสามารถเขียนได้ดังนี้

ONTIME [ค่าคงตัว], [เลขบรรทัด] ค่าคงตัวจะเป็นตัวกำหนดระยะเวลาในการขัดจังหวะขึ้นทุกครั้ง ส่วนเลขบรรทัดจะเป็นตำแหน่งที่ใช้ไปกระโดดไปกระทำคำสั่ง เมื่อเกิดการขัดจังหวะทุก ๆ ค่าคงตัว/วินาที

ตัวอย่าง ONTIME

```
> 10 TIME =0 : CLOCK1 : ONTIME 2, 100 : DO
> 20 WHILE TIME < 10 : END
> 30 PRINT "TIMER INTERRUPT AT- ", TIME, "SEC"
> 110 ONTIME TIME+2, 100 : RETI
> RUN
TIMER INTERRUPT AT- 2.045 SEC
TIMER INTERRUPT AT- 4.045 SEC
TIMER INTERRUPT AT- 6.045 SEC
TIMER INTERRUPT AT- 8.045 SEC
TIMER INTERRUPT AT- 10.045 SEC
READY
```

■ คำสั่ง PRINT

คำสั่ง PRINT เป็นคำสั่งที่ใช้สำหรับพิมพ์ค่าข้อมูลต่าง ๆ ที่ผู้ใช้งานต้องการทราบออกหน้าจอแอลซีดี

ตัวอย่าง

```
> PRINT 10*10, 3*3   > PRINT " BASIC-52 ",      > PRINT 5, 1E3
   100     9                  BASIC-52                  5     1000
```

- คำสั่ง PH0

คำสั่ง PH0 คำสั่งนี้คล้ายกับคำสั่ง PRINT แต่ข้อมูลที่แสดงออกมาจะเป็นเลขฐาน 16

ตัวอย่าง

```
> PH0. 2*2      > PH0. 1000      > PH0. 100
   04H              3E8H             64H
```

- คำสั่ง PUSH และ POP

คำสั่ง PUSH เป็นคำสั่งที่นำข้อมูลมาเก็บไว้ในชั้นหน่วยความจำ แล้วเพิ่มค่าชั้นหน่วยความจำขึ้น 1 ค่า คำสั่ง POP เป็นคำสั่งที่ดึงข้อมูลจากชั้นหน่วยความจำ แล้วลดค่าชั้นหน่วยความจำลง 1 ค่า

ตัวอย่าง PUSH, POP

```
> 10  A= 10
> 20  B= 20
> 30 PRINT A, B
> 40 PUSH A, B
> 50 POP A, B
> 60 PRINT A, B
> RUN
   10        20
   20        10
READY
>
```

- คำสั่ง REM (Remark)

คำสั่ง REM เป็นคำสั่งที่ใช้สำหรับหมายเหตุ MCS BASIC-52 เมื่อตรวจพบคำสั่งนี้จะข้ามไปนำค่าอักษรที่อยู่หลังคำสั่ง REM ไปประมวลผล

ตัวอย่าง REM

```
> 10  INPUT A : REM INPUT ONE VARIABLE
> 20 INPUT B  : REM INPUT ANOTHER VARIABLE
> 30 Z = A*B  : REM MULTIPLY THE TWO
> 40 PRINT Z  : REM PRINT THE ANSWER
```

- คำสั่ง RETI (Return Interrupt)

คำสั่ง RETI เป็นคำสั่งที่ใช้กำหนดให้โปรแกรมกระโดดกลับมายังโปรแกรมหลักหลังจากถูกขัดจังหวะโดยคำสั่ง ONEX1

รายละเอียดของตัวดำเนินการใน MCS BASIC-52

ตัวดำเนินการบวก (+)

ตัวอย่าง (+)

```
> PRINT 3+2
      5
```

ตัวดำเนินการหาร (/)

ตัวอย่าง (/)

```
> PRINT 100/5
      20
```

ตัวดำเนินการคูณ (*)

ตัวอย่าง (*)

```
> PRINT 3*3
      9
```

ตัวดำเนินการยกกำลัง (**)

ตัวอย่าง (**)

```
> PRINT 2**3
      8
```

ตัวดำเนินการลบ (-)

ตัวอย่าง (-)

```
> PRINT 9-6
      3
```

ตัวดำเนินการและ (.AND.)

ตัวอย่าง (.AND.)

```
> PRINT 3 .AND.2
```

2

ตัวดำเนินการ .OR.

ตัวอย่าง .OR.

> PRINT 1.OR.4

5

ตัวดำเนินการ .XOR.

ตัวอย่าง .XOR.

> PRINT 7.XOR.6

1

ตัวดำเนินการค่าสัมบูรณ์ (ABS)

ตัวอย่าง (ABS)

> PRINT ABS (5)

5

> PRINT ABS (-5)

5

ตัวดำเนินการผกผัน (NOT)

ตัวอย่าง (NOT)

> PRINT NOT (6500)

535

> PRINT NOT (0)

65535

ตัวดำเนินการจำนวนเต็ม (INT)

ตัวอย่าง INT.

> PRINT INT (3.7)

3

ตัวดำเนินการเกี่ยวกับเครื่องหมาย (SGN)

ตัวอย่าง SGN

> PRINT SGN (52)

1

> PRINT SGN (0)

0

> PRINT SGN (-8)

-1

ตัวดำเนินการรากที่สอง SQR

ตัวอย่าง SQR

> PRINT SQR (9)

3

ค่าคงที่ PI

PI เป็นค่าคงที่ มีค่าเท่ากับ 3.1415926

ตัวดำเนินการ LOG

ตัวอย่าง LOG

> PRINT LOG (12)

2.484906

ตัวดำเนินการ " e " (EXP)

ตัวอย่าง EXP

> PRINT EXP (1)

2.7182818

> PRINT EXP (LOG(2))

2

นอกจากนี้ยังมีตัวดำเนินการเกี่ยวกับการเปรียบเทียบ ได้แก่

- เท่ากับ (=)
- ไม่เท่ากับ (< >)
- มากกว่า (>)
- น้อยกว่า (<)
- มากกว่าหรือเท่ากับ (>=)
- น้อยกว่าหรือเท่ากับ (<=)

รายละเอียดของตัวดำเนินการพิเศษ

ตัวดำเนินการ CBY ( )

ใช้สำหรับเขียนหรืออ่านข้อมูลจากหน่วยความจำแบบโปรแกรม โดยอยู่ในช่วงระหว่าง 0 ถึง 65535

ตัวอย่าง CBY ( )

> A = CBY (1000)

> CBY (2FH) = A

ตัวดำเนินการ DBY( )

DBY( ) เป็นตัวดำเนินการใช้สำหรับเขียนหรืออ่านข้อมูลจากหน่วยความจำแบบข้อมูล ตั้งแต่ตำแหน่ง 0 ถึง 255

> A = DBY (250) > DBY (250) = A

ตัวดำเนินการ XBY( )

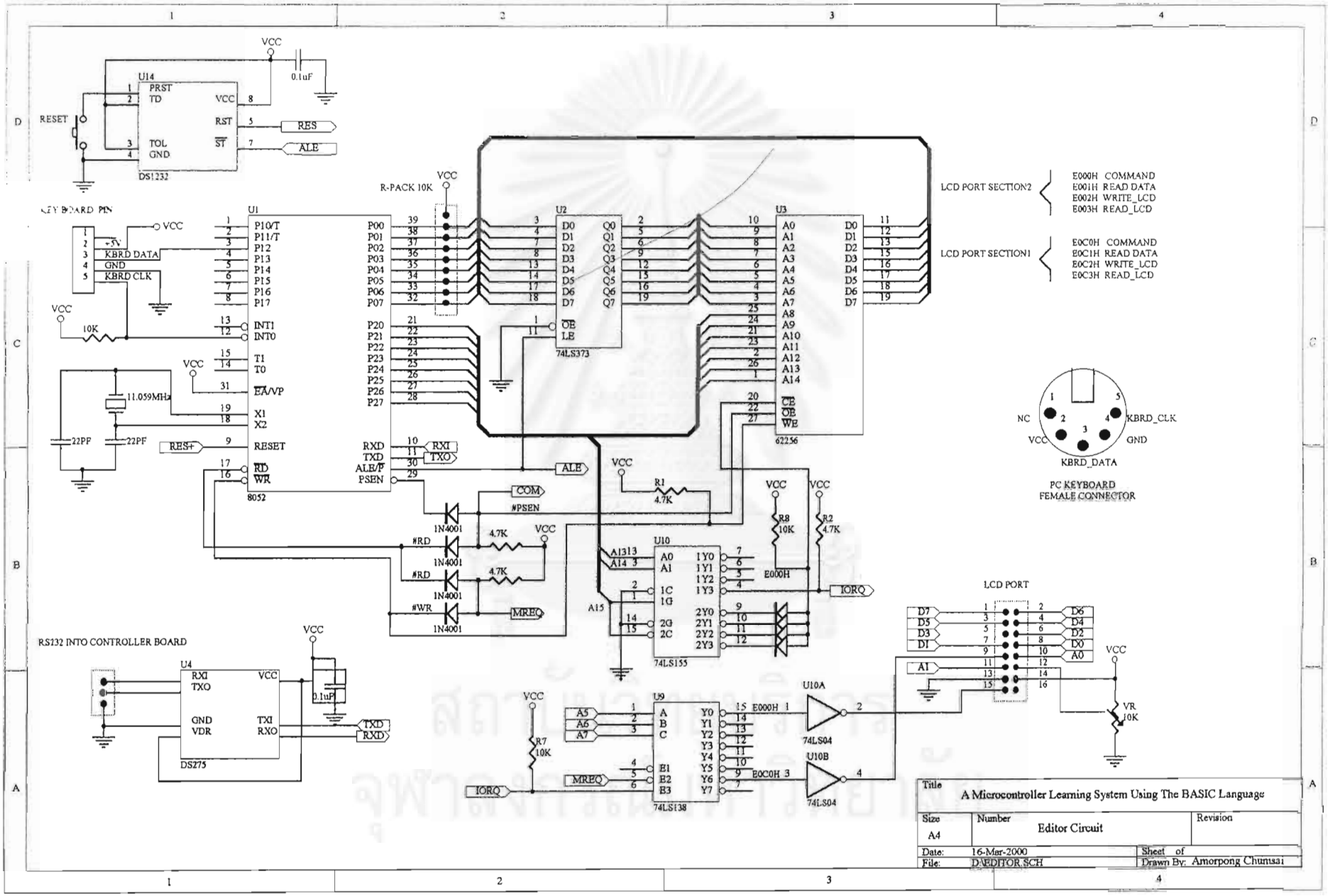
XBY( ) เป็นตัวดำเนินการใช้สำหรับเขียนหรืออ่านข้อมูลจากหน่วยความจำแบบข้อมูล ตั้งแต่ตำแหน่ง 256 ถึง 65535

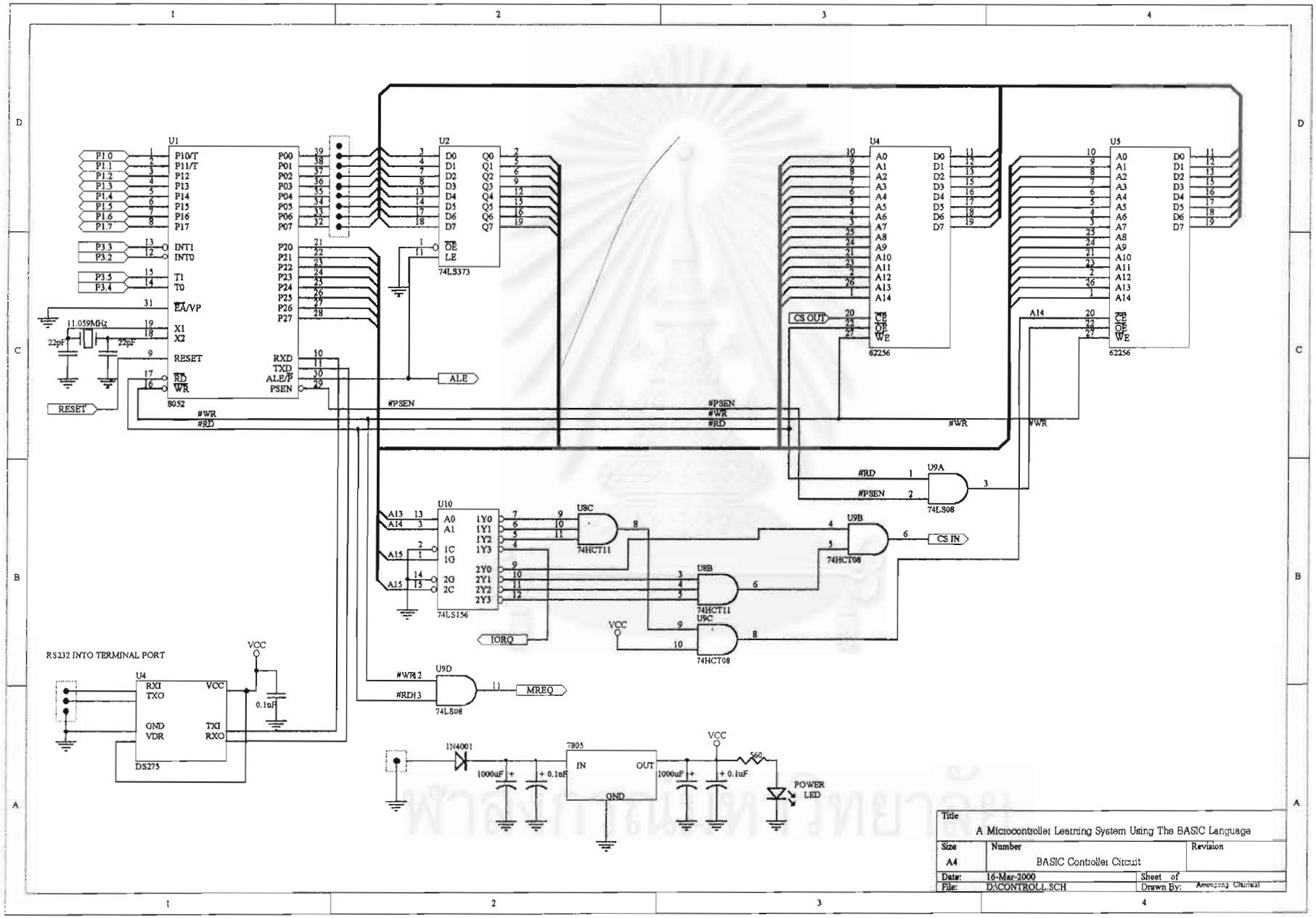
ตัวอย่าง XBY( )

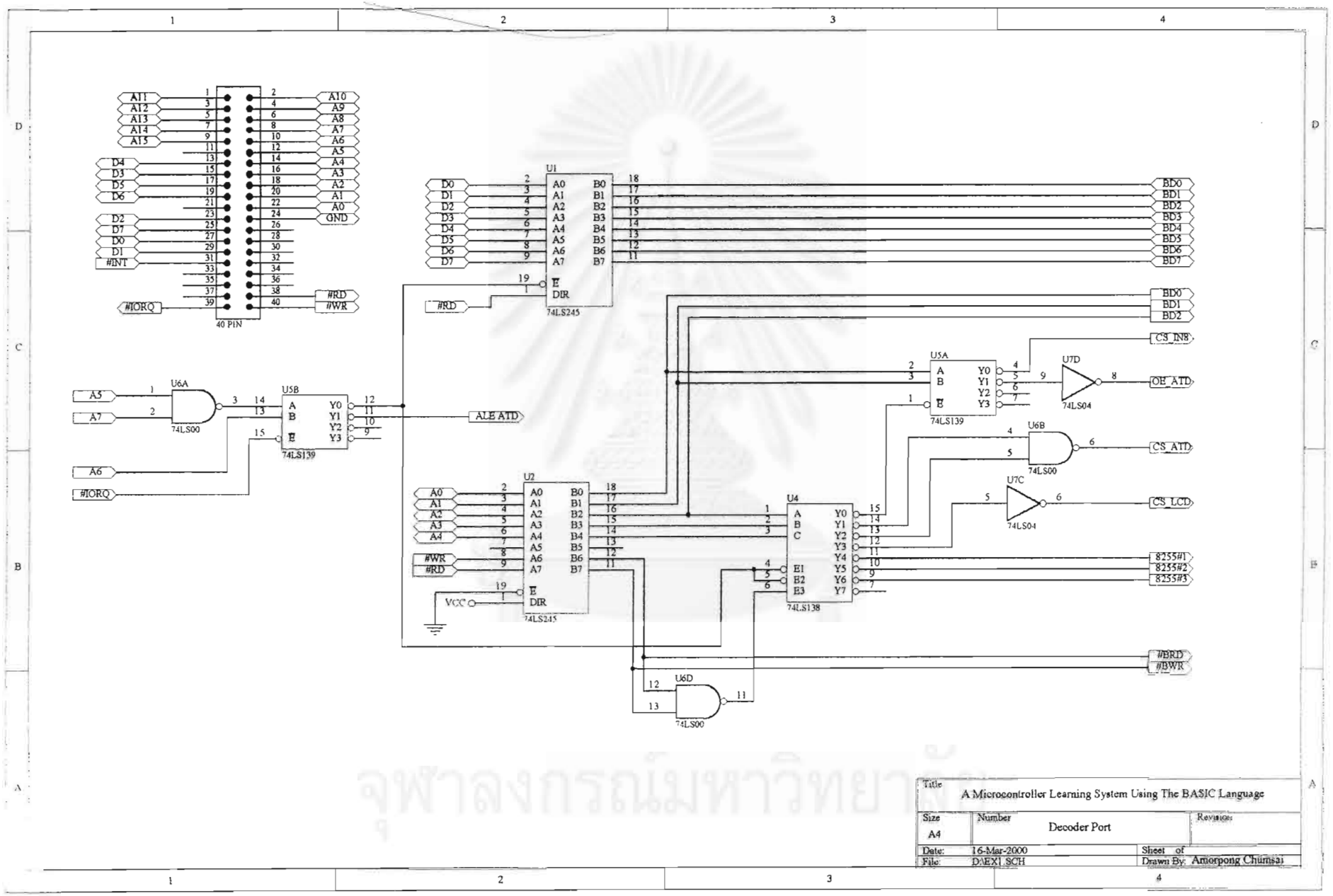
> XBY (4000H) = DBY (100) > A = XBY (0F000H)

นอกจากนี้ยังมีตัวดำเนินการเกี่ยวกับเรจิสเตอร์พิเศษ ได้แก่

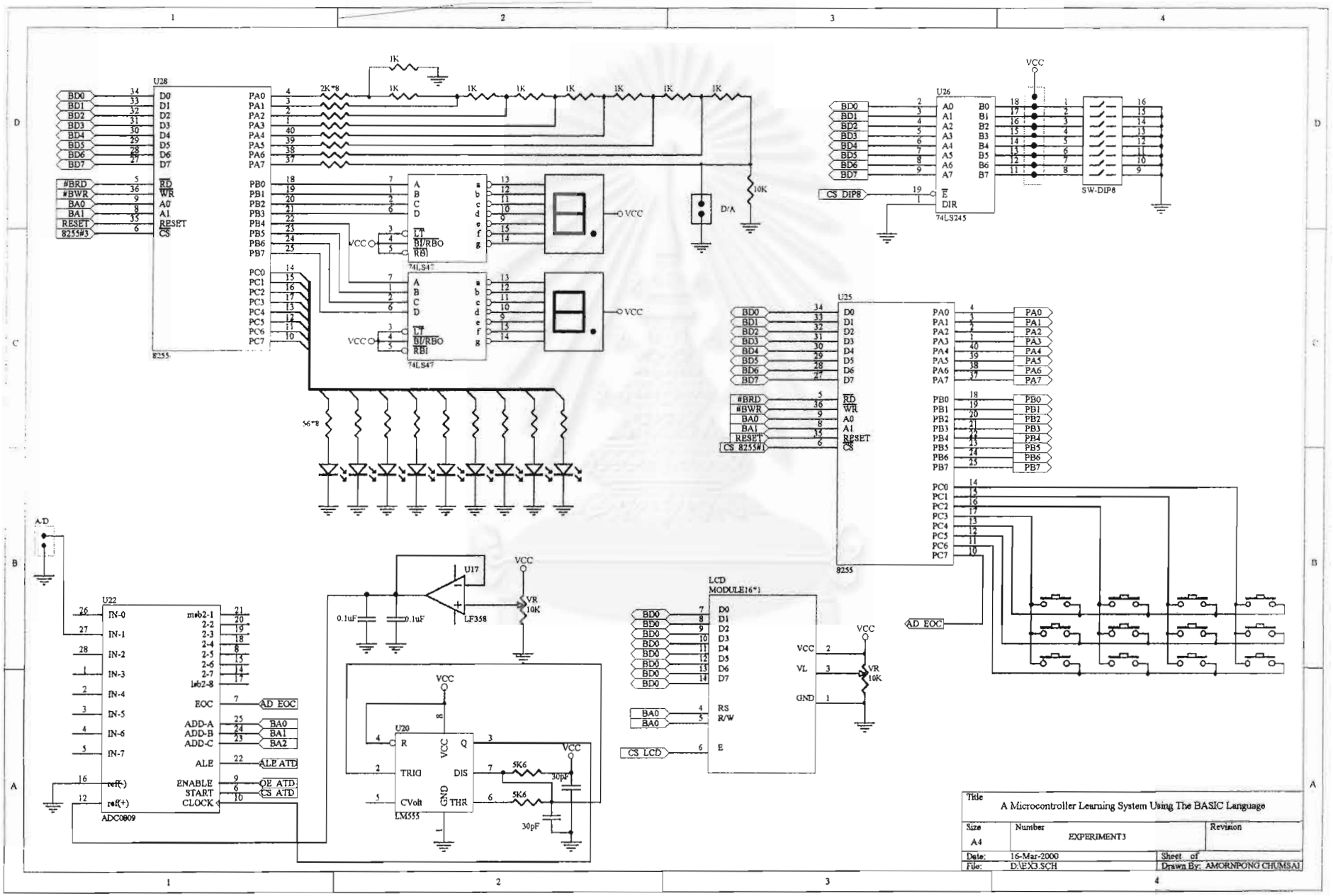
- เรจิสเตอร์ IE
- เรจิสเตอร์ IP
- เรจิสเตอร์ PORT1
- เรจิสเตอร์ PCON
- เรจิสเตอร์ RCAP2
- เรจิสเตอร์ T2CON
- เรจิสเตอร์ TCON
- เรจิสเตอร์ TMOD
- เรจิสเตอร์ TIMER0
- เรจิสเตอร์ TIMER1
- เรจิสเตอร์ TIMER2

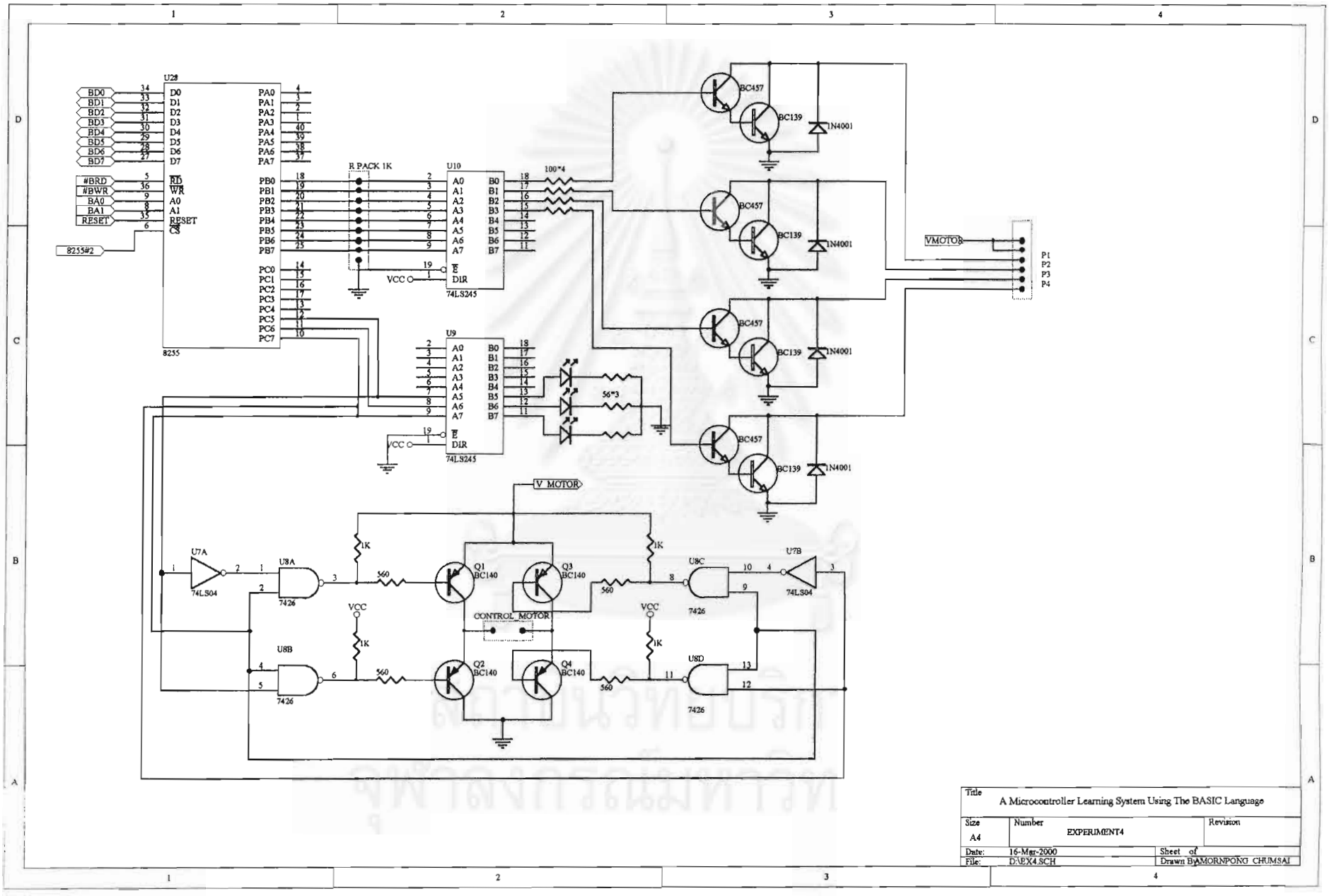

U14
DS1232
RESET
RES
ALE
R-PACK 10K
U1
8052
U2
74LS373
U3
62256
LCD PORT SECTION2
E000H COMMAND
E001H READ DATA
E002H WRITE_LCD
E003H READ_LCD
LCD PORT SECTION1
E0C0H COMMAND
E0C1H READ DATA
E0C2H WRITE_LCD
E0C3H READ_LCD
KEY BOARD PIN
KBRD DATA
KBRD CLK
11.059MHz
22PF
10K
RES+
COM
#PSEN
#RD
#WR
1N4001
4.7K
U10
74LS155
E000H
IORQ
A15
LCD PORT
U4
DS275
RS232 INTO CONTROLLER BOARD
TXD
RXD
U9
74LS138
MREQ
E0C0H
U10A
U10B
74LS04
VR
10K
PC KEYBOARD
FEMALE CONNECTOR
KBRD_CLK
KBRD_DATA
Title
A Microcontroller Learning System Using The BASIC Language
Size
A4
Number
Editor Circuit
Revision
Date: 16-Mar-2000
File: D:\EDITOR.SCH
Sheet of
Drawn By: Amorpong Chumsai


U1
8052
P1.0
P1.1
P1.2
P1.3
P1.4
P1.5
P1.6
P1.7
P3.3
P3.2
P3.5
P3.4
EA/VP
11.059MHz
22pF
22pF
RESET
RD
WR
P00
P01
P02
P03
P04
P05
P06
P07
P20
P21
P22
P23
P24
P25
P26
P27
RXD
TXD
ALE/P
PSEN
INT1
INT0
T1
T0
X1
X2
U2
74LS373
OE
LE
ALE
U4
62256
CS OUT
U5
62256
A14
#PSEN
#WR
#RD
U9A
74LS08
U10
74LS156
A13
A14
A15
1C
1G
2G
2C
1Y0
1Y1
1Y2
1Y3
2Y0
2Y1
2Y2
2Y3
U8C
74HCT11
U9B
74HCT08
CS IN
U8B
74HCT11
U9C
74HCT08
VCC
IORQ
RS232 INTO TERMINAL PORT
U4
DS275
RXI
TXO
GND
VDR
VCC
TXI
RXO
0.1uF
#WR2
#RD13
U9D
74LS08
MREQ
1N4001
7805
IN
OUT
GND
1000uF
+ 0.1uF
560
POWER LED
Title
A Microcontroller Learning System Using The BASIC Language
Size
A4
Number
BASIC Controller Circuit
Revision
Date:
16-Mar-2000
Sheet of
File:
DACONTROLL.SCH
Drawn By:
Amonpong Chamnai

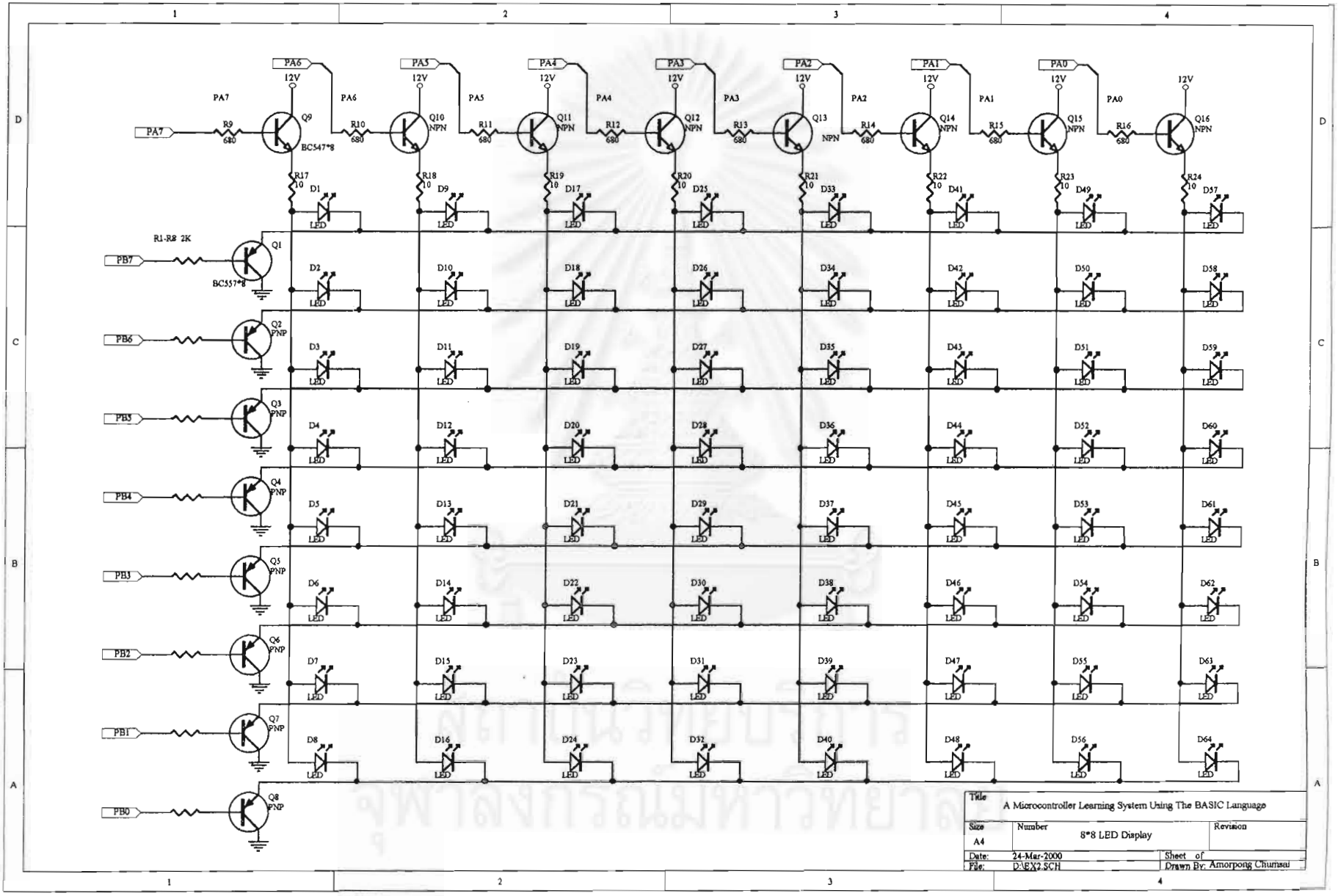


Title	A Microcontroller Learning System Using The BASIC Language		
Size A4	Number Decoder Port		Revision
Date: 16-Mar-2000		Sheet of	
File: D:\EX1.SCH		Drawn By: Amorpong Chumsai	


U28
8255
74LS47
74LS47
U26
74LS245
SW-DIP8
U25
8255
U17
LF358
U22
ADC0809
U20
LM555
LCD
MODULE16*1
A Microcontroller Learning System Using The BASIC Language
Size
A4
Number
EXPERIMENT3
Revision
Date:
16-Mar-2000
File:
D:\EX3.SCH
Sheet of
Drawn By: AMORNPONG CHUMSAI



Title	A Microcontroller Learning System Using The BASIC Language	
Size: A4	Number: EXPERIMENT4	Revision:
Date: 16-Mar-2000		Sheet of
File: D:\EX4.SCH		Drawn By: AMORNPONG CHUMSAI


PA7
PA6
PA5
PA4
PA3
PA2
PA1
PA0
12V
R9 680
R10 680
R11 680
R12 680
R13 680
R14 680
R15 680
R16 680
Q9 BC547*8
Q10 NPN
Q11 NPN
Q12 NPN
Q13 NPN
Q14 NPN
Q15 NPN
Q16 NPN
R17 10
R18 10
R19 10
R20 10
R21 10
R22 10
R23 10
R24 10
D1 D9 D17 D25 D33 D41 D49 D57
R1-R8 2K
PB7
PB6
PB5
PB4
PB3
PB2
PB1
PB0
Q1 BC557*8
Q2 PNP
Q3 PNP
Q4 PNP
Q5 PNP
Q6 PNP
Q7 PNP
Q8 PNP
D2 D10 D18 D26 D34 D42 D50 D58
D3 D11 D19 D27 D35 D43 D51 D59
D4 D12 D20 D28 D36 D44 D52 D60
D5 D13 D21 D29 D37 D45 D53 D61
D6 D14 D22 D30 D38 D46 D54 D62
D7 D15 D23 D31 D39 D47 D55 D63
D8 D16 D24 D32 D40 D48 D56 D64
LED
Title
A Microcontroller Learning System Using The BASIC Language
Size
A4
Number
8*8 LED Display
Revision
Date: 24-Mar-2000
File: D:\EX2.SCH
Sheet of
Drawn By: Amorpong Chumsai

ประวัติผู้เขียน

นายอมรพงษ์ ชุมสาย ณ อยุธยา เกิดวันที่ 25 เมษายน พ.ศ. 2516 ที่จังหวัด กรุงเทพมหานคร สำเร็จการศึกษาระดับปริญญาตรี หลักสูตรวิศวกรรมศาสตรบัณฑิต ภาควิชาวิศวกรรมอิเล็กทรอนิกส์ จากคณะวิศวกรรมศาสตร์ สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง ในปีการศึกษา 2538 เคยทำงานเป็นวิศวกรระบบสื่อสัญญาณ ที่บริษัทไทยเทเลโฟนแอนด์เทเลคอมมิวนิเคชั่น จำกัด (มหาชน) และเข้าศึกษาต่อในหลักสูตร วิศวกรรมศาสตรมหาบัณฑิต คณะวิศวกรรมศาสตร์ ภาควิชาวิศวกรรมไฟฟ้า ณ จุฬาลงกรณ์มหาวิทยาลัย ในปีการศึกษา 2540