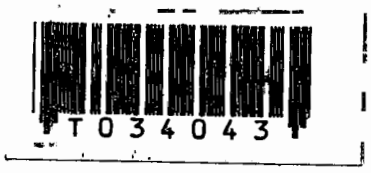
บอร์ดแสดงผลโดย LED

LED DISPLAY BOARD



โดย

นางสาววิไล แม้นถาวรสิริ

นายศักดิ์ชัย เลปนะวัฒน์

นายสรณชนม์ นิชพรกุล

ปริญญานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิศวกรรมศาสตรบัณฑิต

สาขาวิชาอิเล็กทรอนิกส์

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

ปีการศึกษา 2541

บอร์ดแสดงผลโดย LED

LED DISPLAY BOARD

โดย

นางสาววิไล แม้นถาวรสิริ เลขประจำตัว 38014471

นายศักดิ์ชัย เลปนะวัฒน์ เลขประจำตัว 38014493

นายสรณชนม์ นิชพรกุล เลขประจำตัว 38014537

อาจารย์ที่ปรึกษา

อาจารย์ชินภัทร นันทจิวากรชัย

ปริญญานิพนธ์สำหรับปริญญาวิศวกรรมศาสตรบัณฑิต

สาขาวิชาอิเล็กทรอนิกส์

คณะวิศวกรรมศาสตร์

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

ปีการศึกษา 2541

ปริญญานิพนธ์ ปีการศึกษา 2541

ภาควิชา อิเล็กทรอนิกส์

คณะวิศวกรรมศาสตร์ สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

เรื่อง บอร์ดแสดงผลโดย LED

ผู้จัดทำ

1. นางสาววิไล แม้นถาวรสิริ
2. นายศักดิ์ชัย เลปนะวัฒน์
3. นายสรณชนม์ นิชพรกุล

.....อาจารย์ที่ปรึกษา

(อาจารย์ชินภัทร นันทจิวากรชัย)

บอร์ดแสดงผลโดย LED

LED DISPLAY BOARD

นางสาววิไล แม้นถาวรสิริ เลขประจำตัว 38014471

นายศักดิ์ชัย เลปนะวัฒน์ เลขประจำตัว 38014493

นายสรณชนม์ นิชพรกุล เลขประจำตัว 38014537

โครงงานได้รับการตรวจสอบแล้ว พร้อมที่จะทำการสอบได้

.....

(อาจารย์ชินภัทร นันทจิวากรชัย)

อาจารย์ที่ปรึกษา

บอร์ดแสดงผลโดย LED

นางสาววิไล แม้นถาวรสิริ
นายศักดิ์ชัย เลปนะวัฒน์
นายสรณชนม์ นิชพรกุล
อ.ชินภัทร นันทจิวากรชัย อาจารย์ที่ปรึกษา
ปีการศึกษา 2541

บทคัดย่อ

ในปริญญานิพนธ์ฉบับนี้ เป็นการศึกษาเกี่ยวกับการการแสดงผลตัวอักษร รูปภาพผ่านทาง LED โดยวงจรที่ได้ออกแบบจะประกอบไปด้วยส่วนต่างๆที่จำเป็นต้องศึกษาคือ วงจรสร้างสัญญาณนาฬิกา โดยใช้ ไอซีเบอร์ LM555 เป็นตัวกำเนิดความถี่ , วงจรนับ ใช้ไอซีเบอร์ 74HC4040 เป็นตัวนับและสร้างแอดเดรสให้แก่แรม (Ram) , วงจรถอดรหัส ใช้เป็นวงจรเลือกสัญญาณทางด้านแนวนอนและแนวตั้ง , วงจรพักข้อมูล ใช้พักข้อมูลขณะแสดงผล , วงจรเก็บข้อมูลภาพ ใช้แรมขนาด 2 กิโลไบร์ท ขนาด 8 บิท ในการเก็บข้อมูล 1 ภาพ , วงจรขับกระแส เป็นส่วนจ่ายไฟ และเป็นเหมือนสวิตช์เลือก LED , ส่วนแสดงผล ประกอบไปด้วย LED ขนาด 60*91 ต่อกันแบบเมตริก, และส่วนไมโครคอลโทรลเลอร์ (Micricontroller)ใช้ในการรับส่งข้อมูลภาพจากคอมพิวเตอร์ส่วนบุคคล และควบคุมการแสดงภาพในรูปแบบต่างๆ

LED DISPLAY BOARD

Vilai Manthavornsiri

Sakchai Lepnawat

Sornachon Nichaporngoon

Chinapat Nantajivagornchai Advisor

1998

ABSTRACT

This thesis is studying about display graphic and text by LED's Board. The designing circuit consists of many part. The important of this thesis consist of Clock Generator Circuit is used IC LM555 to generate clock signal. Counter Circuit is used IC 74HC4040 to count and create address for RAM. Decoder Circuit is used to select column and row trick signal. Latch Buffer Circuit is used to latch data to display. RAM is used size 2 Kbyte 8 bit to record graphic data. Drive Current Circuit is used to drive current and switch LED. Display Board consists of LEDs size 60*91 dot matrix. The Microcontroller is used to receive graphic data and control processing display graphic.

สารบัญ

สารบัญรูปและตาราง

หน้าที่

บทที่1
บทนำ

เนื่องจากในปัจจุบันนี้เทคโนโลยีทางด้านสารสนเทศเข้ามามีบทบาทเป็นอย่างมากในชีวิตประจำวัน เพราะปัจจุบันเป็นโลกของข่าวสารข้อมูล ที่เป็นหนึ่งเดียวกันทั่วโลก โดยเฉพาะ อย่างยิ่งด้านธุระกิจและการโฆษณา ดังนั้นในปัจจุบันจึงต้องการสื่อที่สามารถแสดงข่าวสารข้อมูลรวมทั้งภาพการเคลื่อนไหวต่างๆ เพื่อให้สามารถดึงดูดความสนใจ และสามารถสื่อสารให้ทุกคนได้เข้าใจในข้อมูลข่าวสารนั้นๆ

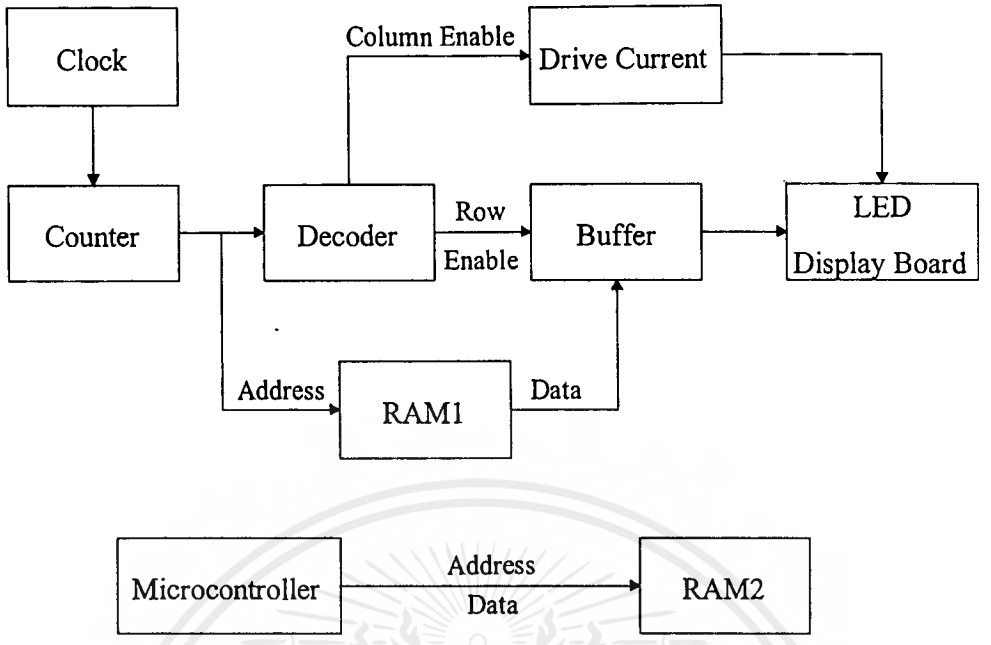
กระดาน หรือแผ่นป้ายแสดงข่าวสาร (Display Board) เป็นอีกสื่อหนึ่งที่เป็นที่นิยม ใช้กันมากในปัจจุบัน ซึ่งสามารถพบเห็นได้ตามแหล่งชุมชน ย่านธุรกิจ ศูนย์การค้า โรงพยาบาลและตามบริษัทห้างร้านต่างๆ ซึ่งในอดีตยังคงใช้ระบบการแสดงข้อมูลแบบเป็นเซเวนเซกเมนต์ (7-Segments) และต่อมาจึงมีการพัฒนาเป็นแบบจุดแสดงผล (Dot Matrix Display) ซึ่งจะมีความละเอียดของภาพ (Resolution) มากกว่าแบบเดิม

การแสดงผลมีทั้งแบบตัวอักษรภาษาไทย ภาษาอังกฤษ รวมทั้งรูปภาพกราฟฟิกต่างๆ นอกจากนี้แล้วสามารถดัดแปลงรูปแบบการแสดงผลแบบอินเวอร์ส โดยใช้สีพื้น (Back Ground) เป็นสีต่างๆ แล้วให้ภาพที่ต้องการเป็นดำ หรือดับนั้นเอง

นอกจากนี้แล้วปัจจุบันป้ายโฆษณาที่มีลักษณะเป็นตัวอักษรเลื่อนก็เป็นที่นิยม ดังจะเห็นได้จากตามห้างสรรพสินค้าทั่วไป หรือแม้แต่ร้านค้าใหญ่ มักจะมีป้ายโฆษณาในลักษณะนี้ติดตั้งอยู่มากมาย สามารถดึงดูดความสนใจได้มากกว่าป้ายโฆษณาทั่วๆ ไป เนื่องจากมองเห็นได้เด่นชัด มีรูปแบบในการแสดงผลที่น่าสนใจ จึงมีผู้นิยมใช้กันมาขึ้นเรื่อยๆ ลงทุนเพียงครั้งเดียวสามารถใช้ได้ตลอดและสามารถปรับเปลี่ยนข้อความหรือรูปได้ ตามความต้องการ

1.1 ความมุ่งหมายของโครงงาน

ดังนั้นในโครงงานนี้ เราจึงนำหลอด LED ชนิด Dot Matrix ขนาด 5 x 7 มาประยุกต์ใช้มาทำเป็นบอร์ดแสดงผลแบบจุด (Dot Matrix Display Board) เป็นบอร์ดแสดงข่าวสาร ซึ่งจะได้เปรียบบอร์ดแสดงข่าวสารแบบเก่าซึ่งสามารถแสดงข้อมูลได้เพียงรูปแบบเดียวเป็นสามารถแสดงผลภาพหรือตัวอักษร โดยใช้การควบคุมด้วยไมโครโพรเซสเซอร์ และยังเป็นการศึกษาการทำงานของวงจรที่ใช้ในการแสดงผลอีกด้วย



รูปที่ 1.1 บล็อกไดอะแกรม (Block diagram) ของวงจรโดยรวม

โครงงานที่ทำมีส่วนประกอบทางฮาร์ดแวร์ที่สำคัญ ได้แก่

ส่วนแสดงผล (Display Board)

ส่วนตัวขับ (Driver Board)

ส่วนควบคุมและพักข้อมูล (Control / Latch Data Board)

ในโครงงานนี้ก็จะเป็นการออกแบบทางด้านฮาร์ดแวร์ คือส่วนของตัวบอร์ดที่ใช้แสดงผลซึ่งจะทำออกมาเป็นชิ้นงานส่วนแสดงผล 1 บอร์ด ขนาดความละเอียด 60X91 จุด โดยใช้ไมโครคอนโทรลเลอร์เป็นตัวควบคุมการแสดงผล และในส่วนของซอฟแวร์ (Software) จะทำการเขียนโปรแกรมที่จะใช้ควบคุมการแสดงผล 1 บอร์ด โดยการใช้งานโปรแกรมนี้ให้สามารถใช้ คอมพิวเตอร์ส่วนบุคคล (Personal Computer หรือ PC) ส่งข้อมูลภาพและรูปแบบที่ต้องการแสดงผลไปยังไมโครคอนโทรลเลอร์ เพื่อให้ไมโครคอนโทรลเลอร์ประมวลผลและทำการควบคุมการแสดงผลอีกขั้นตอนหนึ่ง

บทที่2
ทฤษฎี

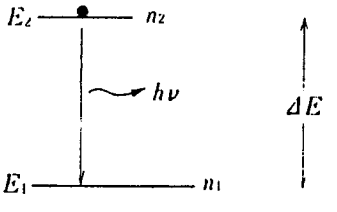
ในการแสดงผลในโครงงานนี้ ส่วนแสดงผลจะประกอบด้วย LED เป็นหัวใจสำคัญ ดังนั้นจึงต้องทำความเข้าใจเกี่ยวกับคุณสมบัติและการทำงานของตัว LED ให้เข้าใจจากนั้นจึงทำการศึกษาทางการแสดงผลโดย LED

2.1 คุณสมบัติของหลอดไดโอดนำแสง (LED)

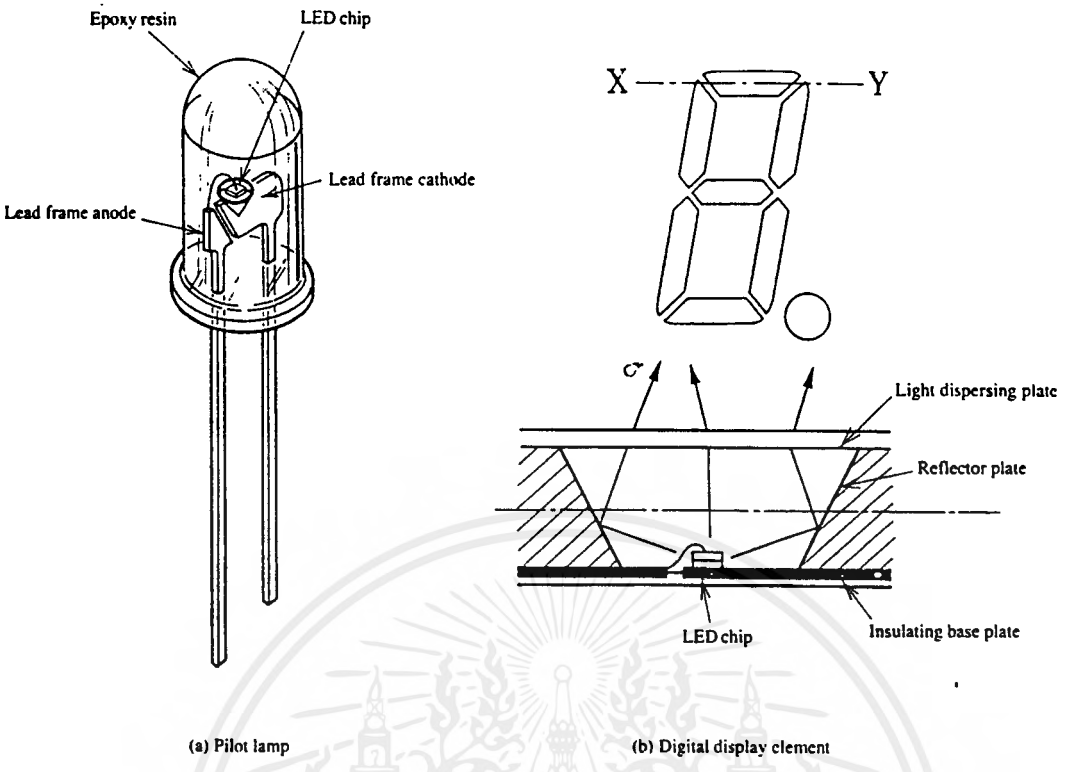
LED เป็นไดโอดชนิดหนึ่งซึ่งสามารถเปล่งแสงได้ โดยโครงสร้างของ LED จะมีลักษณะเป็นสารกึ่งตัวนำชนิดพี (P-Type) และชนิดเอ็น (N-Type) ต่อกันเป็นรอยต่อ พี-เอ็น (P-N Junction) LED จะเปล่งแสงออกมาได้ก็ต่อเมื่อมีการจ่ายกระแสไบอัสตรง (Forward Bias) ให้กับมัน กระแสไบอัสตรงนี้จะไปกระตุ้นอิเล็กตรอน (Electron) และโฮล (Hole) ข้าม รอยต่อ พี-เอ็น เพื่อมารวมตัวกัน ในการรวมตัวกันนี้จะมีการปลดปล่อยพลังงานออกมาในรูปของโฟตอน (Photon) ซึ่งเป็นอนุภาคของแสง ซึ่งต่างจากอุปกรณ์อย่างอื่นๆ ที่ปลดปล่อยพลังงานออกมาในรูปของความร้อน สำหรับสารกึ่งตัวนำที่นิยมนำมาสร้าง LED จะใช้แกลเลี่ยมอาร์เซไนด์ฟอสไฟด์ (Galliumarsenidephosphide : GaAsP) ซึ่งสารทั้งสองชนิดนี้จะใช้กระแสไฟฟ้าไม่มากในการไบอัสเพื่อให้เกิดการปลดปล่อยโฟตอนออกมา การให้แสงของ LED โดยการจ่ายกระแสไฟฟ้า เรียกว่าอิเล็กโตรลูมีเนเซนส์ (Electroluminescence)

คุณสมบัติเด่นของ LED คือ

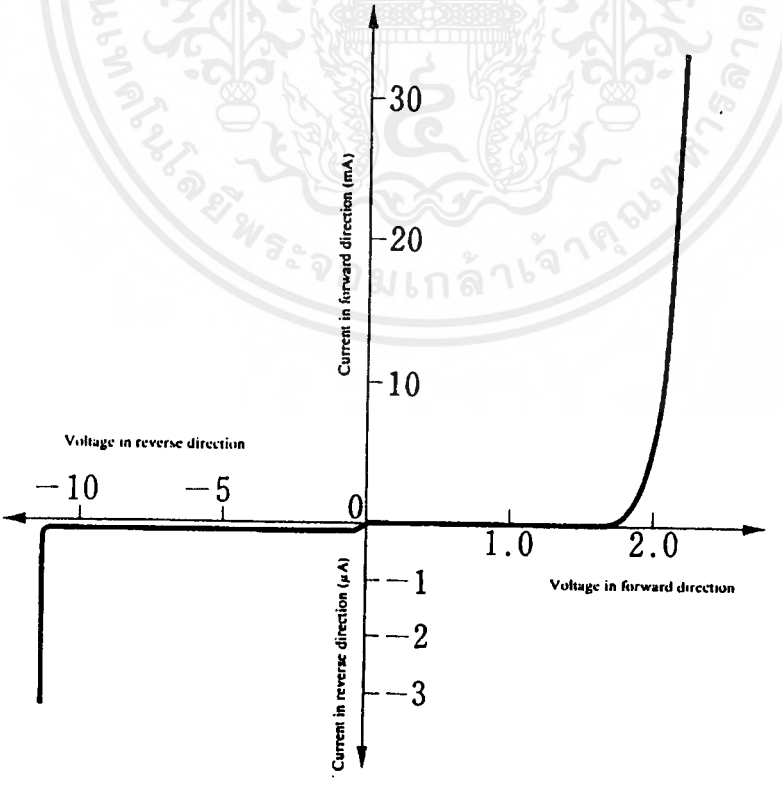
1. เป็นแหล่งกำเนิดแสงขนาดเล็กแต่มีความทนทานสูง
2. ใช้แรงดันที่มาไบอัส LED ต่ำ (ประมาณ 2 V)
3. สามารถนำมาใช้งานในการมอดดูเลชั่น ความเร็วสูงได้



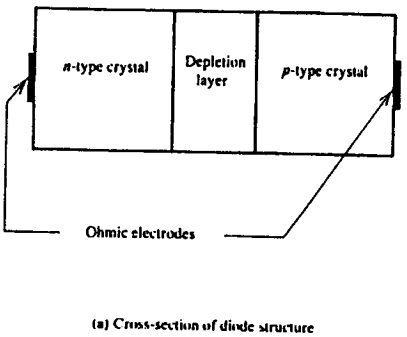
รูปที่ 2.1 แสดงระดับพลังงานของการรวมตัวของอิเล็กตรอน



รูปที่2.2 โครงสร้างทั่วไปของ LED

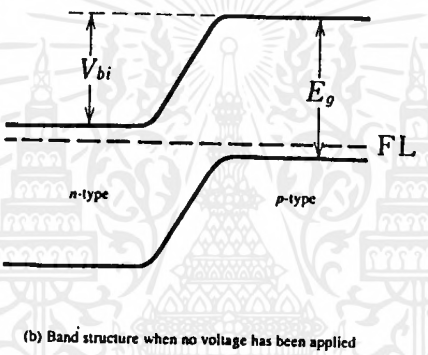


รูปที่ 2.3 กราฟแสดงคุณลักษณะของ LED



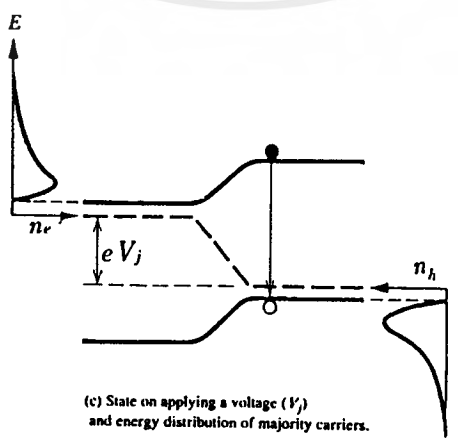
(a) Cross-section of diode structure

รูปที่ 2.4 ภาพตัดขวางของรอยต่อ พี-เอ็น ของ LED



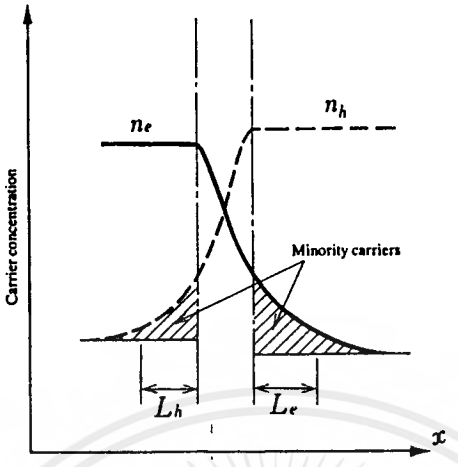
(b) Band structure when no voltage has been applied

รูปที่ 2.5 แสดงแถบพลังงานของ LED เมื่อไม่มีการป้อนศักดาไฟฟ้าให้ LED



(c) State on applying a voltage (V_j) and energy distribution of majority carriers.

รูปที่ 2.6 แสดงแถบพลังงานของ LED เมื่อมีการจ่ายศักดาไฟฟ้าให้ LED



รูปที่ 2.7 แผนภาพแสดงการกระจายของประจุพาหะส่วนมากและส่วนน้อย

หมายเหตุ : E_g : Forbidden bandwidth

V_{bi} : Internal potential difference at junction

n_e : Electron density

n_h : Positive hole density

L_h : Positive hole diffusion length

L_e : Electron diffusion length

FL : Fermi level

E : Energy

V_j : Voltage applied to junction

2.2 การสแกน (Scanning)

ภาพที่เราเห็นหรือตัวอักษรต่างในป้ายโฆษณานั้น จะประกอบไปด้วยจุดเล็กๆ จำนวนหนึ่งที่ทำให้เกิดขึ้นโดยเส้นแนวนอนและแนวตั้ง โดยจะเป็นออกเป็นส่วนๆ ส่วนละเท่าๆกัน จุดเหล่านี้ก็คือ LED หนึ่งดวงนั้นเอง การที่เราจะมองเห็นว่าเป็นภาพได้ ก็คือการควบคุมให้ LED สว่างติดตามต้องการ แต่เราไม่สามารถทำให้ LED เหล่านี้สว่างติดตลอดเวลาได้ ดังนั้นจึงต้องใช้เทคนิคในการสแกน

ปัญหาที่ยุ่งยากอันหนึ่งคือเราจำเป็นต้องควบคุมการติดดับของ LED ให้สามารถมองเห็นเป็นภาพนิ่งหรือภาพที่เคลื่อนไหวได้ โดยปกติแล้ว LED ที่ใช้กันอยู่จะมีความเข้มของการส่องสว่างในระดับ มิลลิคาเดลลา (mcd) ซึ่งเป็นหน่วยการวัดความเข้มของการส่องสว่าง เช่น LED สีแดงให้ความส่องสว่าง 20 mcd ความส่องสว่างเอาท์พุทเมื่อป้อนกระแสตรง 10 มิลิแอมป์ (mA) จะได้เพียง 0.7 mcd ในขณะที่ ถ้าป้อนเป็นพัลส์ที่มีกระแสสูงสุด 100 mA ดิวตี้ไซเคิล t1/T = 1/10 ซึ่งก็ได้กระแสเฉลี่ย 10 mA เช่นกันจะได้ความส่องสว่างเอาท์พุทเฉลี่ย 2 mcd ซึ่งจะเห็นได้ว่าถ้าใช้กระแสพัลส์ จะได้กำลังแสงเอาท์พุทเพิ่มขึ้นเป็น 3 เท่า เมื่อใช้กระแสเท่ากัน และการทำให้มองเห็นเป็นแสงต่อเนื่องได้ จะต้องป้อนกระแสพัลส์มีความถี่มากกว่า 30 Hz ตาคนเราจึงจะมองเห็นแสงเอาท์พุท LED เป็นแสงต่อเนื่อง แต่ถ้าเราต้องการให้เห็นเป็นภาพต่อเนื่องหรือตัวอักษรวิ่งที่ไม่สามารถสังเกตเห็นการกระพริบได้เลย เราควรป้อนพัลส์ที่มีความถี่ประมาณ 50-60 Hz จึงจะไม่เห็นภาพเกิดการพริ้ว และหน้าที่ของการสแกนก็คือการเลือก LED ที่สว่างติดให้ได้ภาพที่ชัดเจนตามต้องการ การสแกนหมายถึง จำนวนเส้นการสแกนต่อหนึ่งภาพ และจำนวนภาพที่ส่งออกไปต่อวินาที ถ้าเราส่งจำนวนต่อวินาทีมากมายเท่าไร การกระพริบของภาพก็จะลดลงเท่านั้น

2.2.1 หลักการของการสแกน

ในการทำไฟวิ่งมีหลักการอยู่ที่การสแกน ซึ่งการสแกนนี้สามารถทำได้ 2 วิธี คือ สแกนทางคอลัมน์ (Column) การสแกนทางโรว์ (Row)

2.2.2 การสแกนทางคอลัมน์

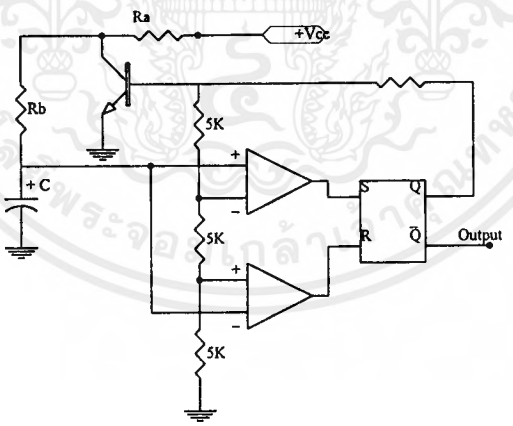
การสแกนทางคอลัมน์จะทำการส่งข้อมูลออกไปทางโรว์ โดยส่งข้อมูลตัวที่ 1 ออกไป แล้วให้คอลัมน์ที่ 1 แอกทีฟ (Active) จากนั้นก็ทำการส่งข้อมูลตัวที่ 2 ออกไปแล้วให้คอลัมน์ที่ 2 แอกทีฟทำเช่นนี้ไปจนกระทั่งข้อมูลถูกส่งออกไปครบหมดทุกคอลัมน์ก็จะเป็นการสแกนครบ 1 รอบ ดังนั้นถ้าจำนวนหลักที่จะแสดงผลออกมาเป็นตัวอักษรมีจำนวนหลายหลัก วิธีนี้จะไม่เหมาะสมนักที่จะนำมาใช้งาน เพราะว่าเมื่อให้ LED ในคอลัมน์ที่ 1 ติด กว่าที่ LED ที่คอลัมน์สุดท้ายจะติดต้องใช้เวลานาน

2.2.3 การสแกนทางโรว์

ส่วนการสแกนทางโรว์จะทำการส่งข้อมูลออกไปจนครบทุกหลักก่อนแล้วให้โรว์ที่1 แอกทีฟ จากนั้นก็ทำการส่งข้อมูลชุดถัดไปออกไปจนครบหมดทุกหลัก แล้วให้โรว์ที่ 2 แอกทีฟ ทำเช่นนี้จนกระทั่งข้อมูลถูกส่งออกไปครบหมดทุกโรว์ ก็จะเป็นการสแกนครบ 1 รอบ วิธีนี้มีข้อดีคือ สามารถแสดงผลเป็นตัวอักษรพร้อมกันได้หลายหลักและถ้าจัดเวลาให้เหมาะสมแล้ว เวลาทำการสแกนจะไม่เกิดอาการพริ้ว แต่มีข้อเสียคือ การเขียนโปรแกรมควบคุมให้ตัวอักษรเลื่อนทำได้ยากว่าแบบแรก

จากรูปวงจรนี้จะสามารถอธิบายการทำงานของวงจรได้คือ ในขั้นแรกเราสมมุติให้เอาท์พุท Q มีสถานะเป็นสูง (High "+Vcc") ซึ่งก็จะทำให้ทรานซิสเตอร์อิ่มตัว (Saturate) ดังนั้นตัวเก็บประจุ C ก็จำถูกช็อตลงกราวด์ ค่าของความต่างศักย์ที่ตัวเก็บประจุก็จะเป็น 0 V เนื่องจากตัวเก็บประจุไม่สามารถถูกชาร์ท (Charge) ได้ เราจะนิยามค่าของความต่างศักย์ที่ขานอนอินเวอร์ติ้ง (Noninverting Input) ของ ออปแอมป์ ในวงจรดังรูปเป็นค่า แรงดันขีดเริ่ม (Threshold Voltage) และในทำนองเดียวกัน ค่าความต่างศักย์ที่ขา อินเวอร์ติ้ง (Inverting Input) เราจะนิยามว่า แรงดันควบคุม (Control Voltage) เราก็จะสามารถกล่าวได้ว่าเมื่อ RS-FF ถูกเซ็ท ทรานซิสเตอร์จะอิ่มตัว ซึ่งก็จะทำให้ค่าแรงดันขีดเริ่มมีค่า 0 V ค่าของแรงดันควบคุมซึ่งมีค่าเท่ากับ 10 V จากการแบ่งแรงดัน ก็จะทำให้เอาท์พุทของออปแอมป์ มีค่าเป็น 0 V ถ้าเราป้อนลอจิกสูง (High) เข้าที่ขา R ซึ่งก็จะเป็นการรีเซ็ท RS-FF ซึ่งเอาท์พุท Q ก็จะมีสถานะเป็นต่ำ (Low) และทรานซิสเตอร์ก็จะคัทออฟ (Cutoff) ตัวเก็บประจุ C ก็จะถูกชาร์ท ผ่านตัวต้านทาน R ค่าของแรงดันขีดเริ่ม ก็จะมีค่าเพิ่มขึ้น เมื่อค่าของแรงดันขีดเริ่มเพิ่มขึ้นจนมากกว่าค่าของแรงดันควบคุม (+10 V) นิดหน่อยเอาท์พุทของออปแอมป์ ก็จะมีสถานะเป็นสูง (+15V) ซึ่งก็จะทำให้ RS-FF ถูกเซ็ท เอาท์พุท Q ก็จะเป็นสูง ทรานซิสเตอร์ก็จะอิ่มตัว ตัวเก็บประจุ C ก็จะถูกดิสชาร์ท (Discharge) ผ่านทรานซิสเตอร์นั่นเอง ในการใช้งานทั่วไปขาสัญญาณควบคุมนี้มักจะไม่ได้ใช้

Astable Opreation



รูปที่ 2.9 วงจรสร้างสัญญาณแบบ Astable

จากรูปที่ 2.9 เป็นการแสดงถึงการต่อไอซี LM555 ทำงานในแบบ Astable หรือ free-running เอาท์พุทที่ได้ของวงจรจะเป็นสัญญาณคลื่นสี่เหลี่ยมที่ต่อเนื่องกันออกมา เราจะสามารถอธิบายการทำงานได้คือ เมื่อเอาท์พุท Q มีสถานะเป็นต่ำ ทรานซิสเตอร์ก็จะคัทออฟ ตัวเก็บประจุ C จะถูกชาร์ทผ่านตัวต้านทานรวมของ $R_A + R_B$ ดังนั้นค่าเวลาคงตัว (Time constant) ก็จะเป็น ($R_A + R_B$) C เมื่อตัวเก็บประจุถูกชาร์ท ค่าแรงดันขีดเริ่มก็จะมีค่าเพิ่มขึ้น เมื่อค่าแรงดันขีดเริ่มมีค่าเพิ่มขึ้นจนมากกว่าค่า

2.2.4 หลักการสแกนโดยใช้ฮาร์ดแวร์เข้าช่วย

โดยทั่วไปในการสแกนไม่ว่าจะเป็นการสแกนทางโรว์หรือการสแกนทางคอลัมน์ เราต้องควบคุมการสแกน และข้อมูลที่ส่งไปเองทุกอย่าง จึงทำให้เกิดปัญหาในกรณีที่เราต้องการแสดงผลที่มีความละเอียดสูง จะทำให้จำนวนคอลัมน์และโรว์มีค่ามากทั้งคู่ ทำให้การแสดงผลตั้งแต่แถวแรก จนถึงแถวสุดท้ายใช้เวลานาน นอกจากนี้ส่วนควบคุมยังก็ต้องใช้เวลามากขึ้น ในการเคลื่อนย้ายข้อมูลเพื่อทำให้ภาพเคลื่อนไหว ทำให้ช่วงนี้ไม่สามารถควบคุมการแสดงผลได้ดี

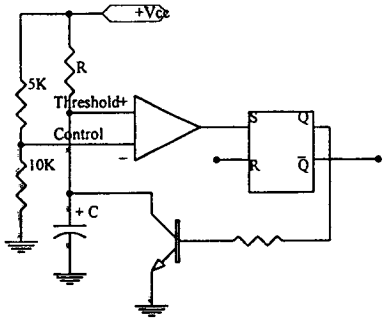
ดังนั้นจึงมีการใช้วงจรทางฮาร์ดแวร์เข้าช่วย โดยใช้ฮาร์ดแวร์นี้ทำการเก็บข้อมูลที่แสดงผลและทำการสแกนออกมาโดยอัตโนมัติ โดยไม่ต้องใช้หน่วยควบคุมเข้าช่วย ทำให้หน่วยควบคุมนี้มีเวลาที่จะเคลื่อนย้ายข้อมูลเพื่อทำภาพเคลื่อนไหวได้โดยไม่ส่งผลกระทบต่อภาพที่กำลังแสดงอยู่ นอกจากนี้เนื่องจากการใช้ฮาร์ดแวร์เข้าช่วยนี้ เป็นการสแกนโดยอัตโนมัติ เราจึงสามารถที่จะแบ่งการสแกนผลที่มีความละเอียดสูงออกเป็นส่วนย่อยๆ เพื่อทำการสแกนด้วยตัวมันเอง ทำให้ลดช่องว่างของเวลาในการแสดงผลระหว่างแถวแรกถึงแถวสุดท้าย ทำให้ภาพที่ได้มีคุณภาพดีขึ้น

ตัวอย่างการสแกนโดยใช้ฮาร์ดแวร์เข้าช่วย อย่างเช่น การใช้แรมแยกออกต่างหาก แล้วทำการส่งข้อมูลออกมาโดยใช้ไอซีเคาท์เตอร์ (IC Counter) ช่วยในการแสดง ซึ่งเป็นวิธีที่ใช้ในปริญญานิพนธ์เล่มนี้ ดังจะได้กล่าวต่อไป

2.3 วงจรสร้างสัญญาณนาฬิกา

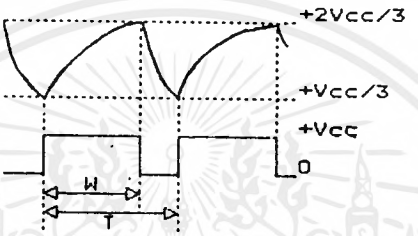
วงจรสร้างสัญญาณนาฬิกาจะใช้ไอซีเบอร์ LM555 ซึ่งเป็น ไอซี 8 ขา ซึ่งจะเอาไว้ต่อกับอุปกรณ์ภายนอกเพื่อที่จะให้ไอซีทำงานในแบบต่างๆ เช่น Monostable หรือ Astable

โครงสร้างภายในของไอซี LM555 เป็นดังรูป

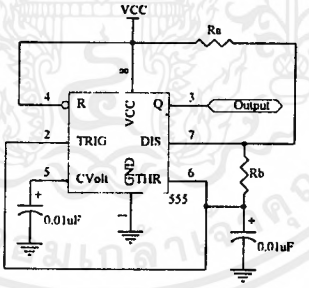


รูปที่ 2.8 โครงสร้างภายในของ LM555

+2Vcc/3 ออปแอมม์ตัวบนก็จะมีเอาท์พุทเป็นสูง ทรานซิสเตอร์ก็จะอิ่มตัว ซึ่งก็เหมือนกับการช็อตขา 7 ของไอซีลงกราว์ด ขณะนี้ตัวเก็บประจุ C ก็จะถูกดิสชาร์ท โดยผ่านทาง R_B ดังนั้นค่าเวลาคงตัว ของการดิสชาร์ทก็คือ R_BC เมื่อค่าของ แรงดันตกคร่อมตัวเก็บประจุ C มีค่าลดลงจนน้อยกว่า +Vcc/3 ออปแอมป์ตัวล่างก็จะมีเอาท์พุทเป็นสูงซึ่งจะ รีเซ็ท RS-FF จากรูป 2.10 จะแสดงถึงลักษณะสัญญาณที่จุดต่างๆ จะเห็นได้ว่าการชาร์ทและดิสชาร์ท ของตัวเก็บประจุจะเป็นแบบเอ็กโปเนนเชียล และสัญญาณเอาท์พุทที่ได้จะเป็นคลื่นสี่เหลี่ยม แต่เนื่องจากค่าเวลาคงตัวของการชาร์ทนั้นมีค่ามากว่าค่า เวลาคงตัวของการดิสชาร์ท สัญญาณเอาท์พุทที่ได้จึงไม่สมมาตร ซึ่งค่าคาบเวลาของสัญญาณในขณะที่มีสถานะสูง จะยาวนานกว่าในขณะที่มีสถานะต่ำดังรูป



รูปที่ 2.10 ลักษณะสัญญาณที่จุดต่างๆ



รูปที่ 2.11 วงจรการต่อ LM555 แบบ Astable

ในการที่จะกำหนดความไม่สมมาตราของสัญญาณเอาท์พุท เราจะนิยามค่าว่า duty cycle ซึ่งจะกำหนดโดย

$$D = (W/T) * 100\%$$

ยกตัวอย่างเช่นถ้า W=2 ms และT=2.5ms ดังนั้นค่า duty cycle คือ

$$D = (2\ ms / 2.5\ ms) * 100\% = 80\ \%$$

ค่า Duty cycle ของวงจรนี้ก็จะมีค่าอยู่ระหว่าง 50-100% ทั้งนี้ก็ขึ้นอยู่กับค่าของ R_A และ R_B

จากการชาร์ทและดิสชาร์ท เราสามารถคำนวณหาค่าของความถี่ของสัญญาณเอาท์พุทคือ

$$f = 1.44 / (R_A + 2R_B) * C$$

และค่าของ duty cycle จะคำนวณได้จาก

$$D = (R_A + R_B) / (R_A + 2R_B) * 100\%$$

จากสูตรจะเห็นได้ว่าถ้าค่า R_A มีค่าน้อยกว่า R_B มากๆ แล้วค่า duty cycle จะประมาณ 50 % และจากรูปที่ 2.11 จะเป็นรูปวงจรที่นิยมเขียนโดยทั่วไปของ วงจรAstable 555 โดยเราต้องต่อขา 4 ของไอซีเข้ากับ +Vcc และเช่นกัน ขาสัญญาณควบคุม ก็ควรต่อตัวเก็บประจุค่าน้อย (โดยมากใช้ 0.01 F) เอาไว้ดังรูป

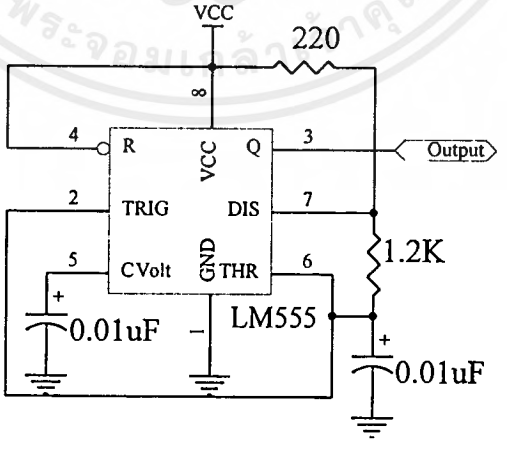
บทที่ 3

หลักการออกแบบ

จากหลักการที่ได้กล่าวมาแล้วในภาคทฤษฎี เราจะความรู้ที่ได้ศึกษามาออกแบบวงจรรวมทั้งหมด โดยจะทำการแยกการออกแบบทีละส่วนเพื่อให้ง่ายในการทำความเข้าใจ และการปรับปรุงเปลี่ยนแปลง ซึ่งในโครงงานนี้ เราได้เลือกใช้ไอซีในตระกูลซี-มอส (C-MOS) เป็นส่วนใหญ่ เนื่องจากในโครงงานนี้มีความจำเป็นต้องไอซีเป็นจำนวนมากและไอซีในตระกูลนี้ใช้กระแสไฟฟ้าในการทำงานน้อยมาก นอกจากนี้แรงดันที่ได้จากไอซีตระกูลนี้ก็มีค่าที่ค่อนข้างคงที่ไม่เปลี่ยนแปลงมาก ซึ่งการรักษาระดับแรงดันนี้มีความจำเป็นในการควบคุมวงจรในส่วนขับกระแสมาก และในปัจจุบันนี้ไอซีในตระกูลซี-มอสนี้ก็มีราคาไม่แตกต่างจากไอซีตระกูลอื่นมากนัก และยังมีให้เลือกใช้มากมายหลายแบบอีกด้วย โดยการออกแบบทั้งหมดนี้เราจะแยกอธิบายออกเป็นส่วนๆดังต่อไปนี้

3.1 วงจรสร้างสัญญาณนาฬิกา

วงจรสร้างสัญญาณ จะใช้ ไอซีเบอร์ LM555 เพื่อความสะดวกในการใช้งานและออกแบบได้ง่าย รวมทั้งสามารถปรับแต่งความถี่ที่ต้องการได้ตามต้องการโดยง่าย โดยปรับค่าความต้านทานในวงจรเท่านั้น ดังรูปที่ 3.1



รูปที่ 3.1 การต่อวงจรสร้างสัญญาณนาฬิกาโดยใช้ไอซีเบอร์ LM555

ในการแสดงภาพนิ่งหรือภาพเคลื่อนไหวจะต้องการใช้ความถี่ในการแสดงภาพอยู่ในช่วง 50-60 ครั้งต่อวินาที (50-60 Hz) ตามที่ได้กล่าวในทฤษฎี เพื่อที่จะให้ภาพที่ปรากฏในบอร์ดแสดงผลไม่เกิดการสั่นพริ้ว หรือเกิดการกระพริบได้

จากรูป วงจรแผงไฟ จะเห็นว่าเราได้ออกแบบขนาดของบอร์ดแสดงผลไว้ให้มีความละเอียดขนาด 91 x 60 จุด ถ้าเราทำการสแกนภาพตามแนวคอลัมน์และในแต่ละแถวใช้บัฟเฟอร์ 12 ตัว (12 x8 = 96 บิท) และเราใช้ วงจรถอดรหัสขนาด 64 x 16 เพราะฉะนั้นจะสามารถคำนวณ สัญญาณนาฬิกาเพื่อให้เกิดความถี่ตามต้องการได้เป็น

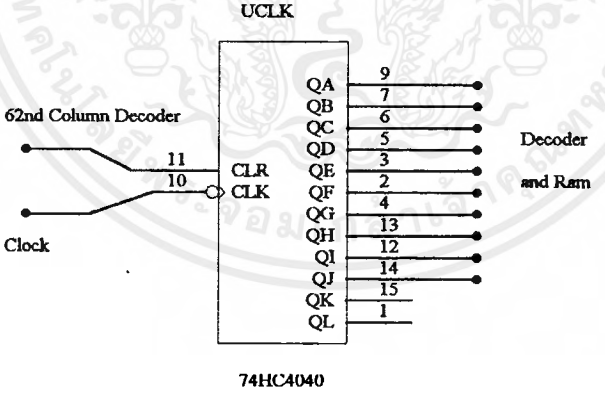
$$\text{Min} = 64 \times 16 \times 50 = 51200 \text{ Hz}$$

$$\text{Max} = 64 \times 16 \times 60 = 61440 \text{ Hz}$$

ความถี่ที่สร้างขึ้นจะต้องมีค่าอยู่ในช่วง Min และ Max จึงจะทำให้มนุษย์สามารถสังเกตเป็นภาพนิ่งได้ และสัญญาณความถี่ที่ได้นี้ก็จะนำไปเป็นอินพุทให้กับวงจรนับต่อไป

3.2 วงจรนับ

วงจรนับนี้จะใช้ไอซีเบอร์ 74HC4040 ซึ่งเป็นไอซีเคาท์เตอร์ขนาด 12 บิทและมีลักษณะของขาสัญญาณดังรูปที่ 3.2



รูปที่ 3.2 แสดงลักษณะของขาสัญญาณของไอซีเบอร์ 74HC4040

ซึ่งไอซีนี้จะใช้นับสัญญาณนาฬิกาจากวงจรสร้างสัญญาณนาฬิกาซึ่งป้อนเข้าที่ขา CLKของไอซี โดยจะทำการนับที่ขอบขาลงของสัญญาณนาฬิกา ให้ออกมาเป็นไบนารี่ขนาด 12 บิตที่ขา Q_A-Q_L ตามลำดับ โดยเมื่อสัญญาณได้ทำการนับจนถึงค่าสุดท้าย (111111111111) ก็จะวนกลับมาที่ค่าเริ่มต้นอีกครั้งหนึ่ง หรือใช้สัญญาณป้อนเข้าที่ขา CLR เพื่อให้การนับกลับมาที่ค่าเริ่มต้นโดยไม่ต้องรอให้การ

นับถึงค่าสุดท้าย ซึ่งในงานนี้จะใช้ขาสัญญาณจากวงจรถอดรหัสที่ควบคุมในแนวคอลัมน์เส้นที่ 62 มาใช้ในการเคลียร์การนับเพื่อที่จะให้วงจรสแกนไม่เสียเวลาในการสแกนคอลัมน์ที่ 63 และ 64 ซึ่งเป็นคอลัมน์ที่ไม่ได้ใช้งาน

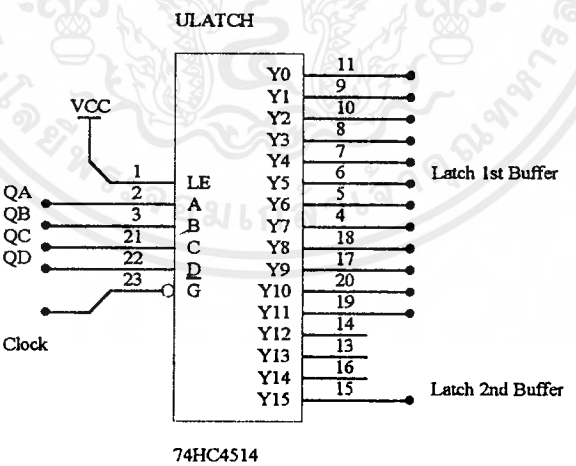
ค่าที่ได้จากวงจรนับนี้จะนำไปใช้ในการกำหนดแอดเดรสให้กับแรมตัวที่จะส่งข้อมูลไปควบคุมการติดดับของ LED ในขณะเดียวกันค่าที่ได้จาก วงจรนับนี้ ก็จะส่งค่าไปควบคุมวงจรรักษาระดับข้อมูล และวงจรถอดรหัส ให้ทำการเก็บข้อมูลจากแรมที่ส่งข้อมูลออกมาและทำการสแกนทางแนวคอลัมน์อย่างอัตโนมัติ ให้สัมพันธ์กับเวลาที่ข้อมูลถูกส่งออกมาจากแรม

3.3 วงจรถอดรหัส

วงจรถอดรหัสนี้จะออกเป็น 2 ส่วนคือ ส่วนที่ใช้สำหรับควบคุมการสแกนในแนวคอลัมน์และและส่วนที่ใช้ควบคุมการสแกนในแนวโรว์

3.3.1ส่วนควบคุมการสแกนในแนวโรว์

ในส่วนนี้สัญญาณจาก 4 บิทล่างของวงจรนับ (ขา Q_A-Q_D) มาป้อนให้กับไอซีถอดรหัสเบอร์ 74HC4514 ซึ่งเป็นไอซีถอดรหัสเข้า 4 ออก 16 และทำงานที่สถานะสูง (IC 4-16 Decoder Active High) ที่ขา A-D ตามลำดับ และมีลักษณะของขาสัญญาณดังรูปที่ 3.3



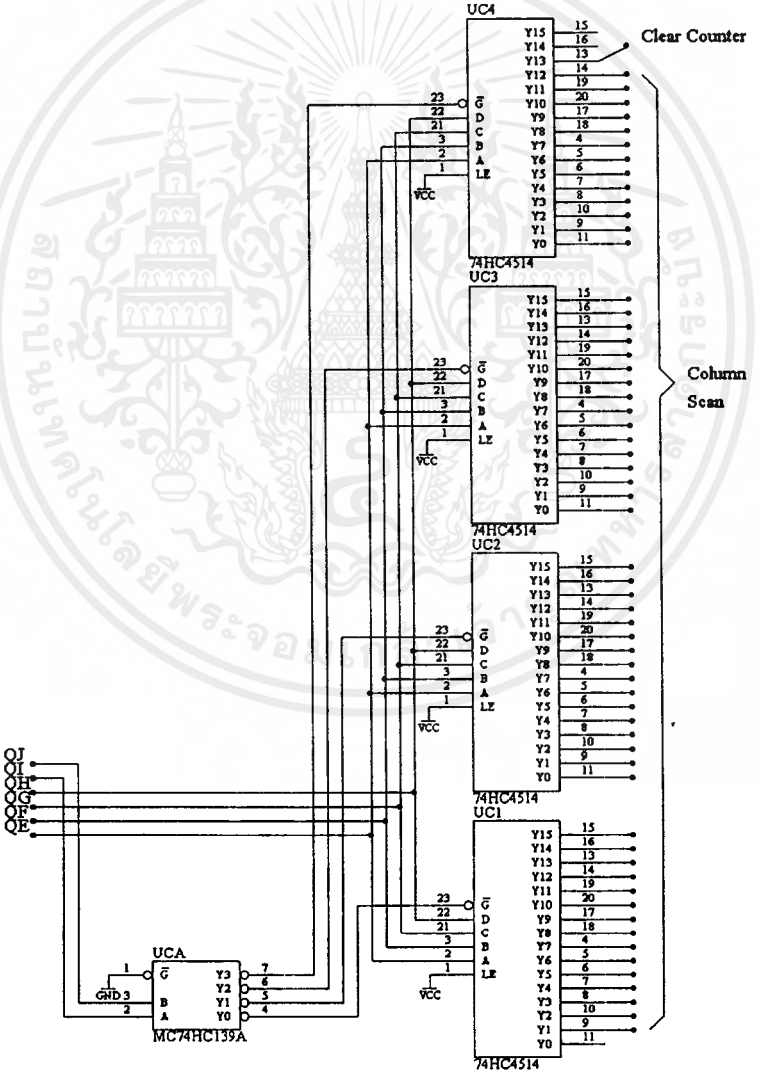
รูปที่ 3.3 แสดงลักษณะของขาสัญญาณของไอซีเบอร์ 74HC4514

ในวงจรนี้ เราจะต่อขาที่ 1 ลงไฟเลี้ยง เพื่อให้สัญญาณออกเปลี่ยนไปตามค่าสัญญาณอินพุทตลอดเวลา และต่อขาที่ 23 กับสัญญาณนาฬิกา เพื่อให้ไอซีทำงานในช่วงครึ่งของการแลทช์ข้อมูล เพื่อป้องกันการหน่วงของเวลาที่ไม่เท่ากันของข้อมูลที่ออกมาจากแรมและสัญญาณที่ใช้ในการแลทช์ไอซี

บัฟเฟอร์ โดยขาของสัญญาณเอาท์พุทที่ได้นี้ขา Y0-Y11จะนำไปใช้ในการแลทช์ข้อมูลของไอซีบัฟเฟอร์ชุดที่หนึ่ง และจะต่อขาสัญญาณ Y15 ในการแลทช์ข้อมูลของบัฟเฟอร์ชุดที่สอง ดังจะได้กล่าวในส่วนของวงจรรักษาระดับข้อมูล

3.3.2 ส่วนควบคุมการสแกนในแนวคลอลัมน์

ในการสแกนในแนวนี้ จะใช้สัญญาณจากวงจรนับใน 4 บิทถัดไป (ขา Q_E -Q_H) มาป้อนเป็นอินพุทให้กับไอซีเบอร์ 74HC4514 จำนวน 4 ตัว โดยต่อร่วมกับไอซีเบอร์ 74HC139A ที่ได้รับอินพุทจากกวงจรถอดรหัส 2 บิทถัดไป (ขา Q_I–Q_J) โดยไอซีนี้เป็นไอซีถอดรหัสเข้า 2 ออก 4 จำนวน 2 ชุด ทำงานที่สถานะต่ำ (IC Dual 2-4 Decoder Active Low) แต่เราใช้เพียงชุดเดียวเท่านั้น และมีลักษณะการต่อร่วมกันดังรูปที่ 3.4



รูปที่ 3.4 แสดงลักษณะการต่อของวงจรถอดรหัสในแนวคอลัมน์

จากการต่อวงจรในลักษณะดังรูป ทำให้เราได้วงถอดรหัสที่ทำการขยายเป็นวงจรถอดรหัสเข้า 6 ออก 64 วงจรส่วนที่ได้นี้จะเป็นวงจรสแกนในแนวคอลัมน์ที่ละคอลัมน์เลื่อนไปเรื่อยๆ โดยจะเลื่อนทุกๆครั้งที่การสแกนในแนวโรว์ทำการสแกนครบ 16 ครั้ง

เนื่องจากสัญญาณที่ได้จากวงจรนี้ จะออกมาจากไอซีโดยตรง ทำให้ไม่สามารถที่จะจ่ายกระแสจำนวนมากได้ เราจึงจำเป็นต้องนำสัญญาณเอาท์พุทที่ได้นี้ไปต่อร่วมกับวงจรขับกระแส โดยการต่อเราจะใช้คอลัมน์ที่ 2-61 ในการต่อกับวงจรขับกระแส เนื่องจากในขณะที่ทำการสแกนในคอลัมน์ที่ 1 ข้อมูลจากวงจรรักษาระดับข้อมูลจะยังไม่มีป้อนเข้ามา และส่วนของวงจรขับกระแสจะได้อธิบายในส่วนต่อไป

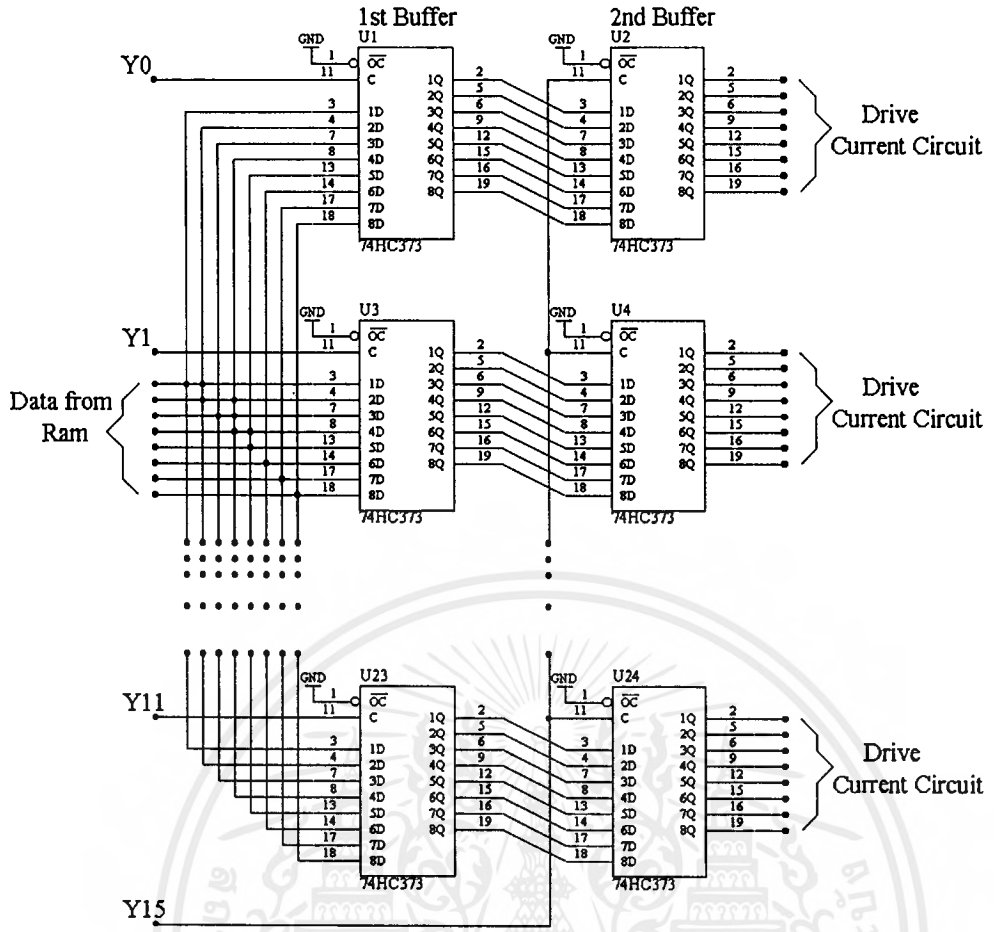
ในวงจรที่ได้นี้เราสามารถที่ใช้นำมาขับกระแสของ LED ได้ถึง 64 คอลัมน์ แต่ในโครงงานนี้เราใช้เพียงแค่ 61 คอลัมน์เท่านั้น (คอลัมน์ที่ 1 ไม่ได้ใช้งาน) ดังนั้นเราจึงนำสัญญาณที่ได้จากคอลัมน์ที่ 62 ไปใช้ในการเคลียร์วงจรนับ เพื่อให้วงจรนับเริ่มนับใหม่และวงจรสแกนก็จะเริ่มทำการสแกนในแนวคอลัมน์ที่ 1 และโรว์ที่ 1 ใหม่ ทำให้วงจรไม่ต้องเสียเวลาในการสแกนคอลัมน์ที่ 63 และ 64 โดยเปล่าประโยชน์

3.4 วงจรรักษาระดับข้อมูลโดยใช้ไอซีบัฟเฟอร์

ในการแสดงภาพแต่ละภาพ ถ้าเราใช้ตัวเก็บข้อมูลหรือรักษาข้อมูลทุกจุด จะเป็นการสิ้นเปลืองอุปกรณ์มาก เราจึงใช้ไอซีบัฟเฟอร์แบบมีแลทช์ (IC Buffer Noninverting with Latch เพื่อรักษาข้อมูลเอาไว้ตัวละ 8 บิต เพื่อรักษาข้อมูลที่ได้รับมาจากแรมเอาไว้ ซึ่งในโครงงานนี้เราได้เลือกใช้ไอซีเบอร์ 74HC373 เนื่องจากเป็นไอซีที่สามารถหาได้ง่ายทั่วไป และมีราคาไม่แพง

ในการใช้ไอซีบัฟเฟอร์ เพื่อรักษาข้อมูลเอาไว้นั้น ถ้าหากมีไอซีบัฟเฟอร์ เพียง 1 ชุด(12 ตัว) ไอซีบัฟเฟอร์ ที่ได้รับข้อมูลจากแรมก่อนก็สามารถส่งข้อมูลให้ LED ติดสว่างได้นานกว่าไอซีบัฟเฟอร์ ที่ได้รับข้อมูลทีหลัง เนื่องจากเมื่อแรมส่งข้อมูลครบ 16 ครั้ง วงจรก็จะเริ่มทำการสแกนในแนวคอลัมน์ถัดไปทันที ทำให้ ไอซีบัฟเฟอร์ที่ได้รับข้อมูลทีหลังสามารถส่งข้อมูลได้เพียงชั่วครู่เท่านั้น ทำให้เกิดปัญหาว่า LED จะสว่างไม่เท่ากันในแต่ละแถว นอกจากนี้หากเราทำการแลทช์ข้อมูลเอาไว้ในขณะที่ไอซีบัฟเฟอร์ตัวแรกๆทำการแลทช์ข้อมูลในคอลัมน์ปัจจุบันอยู่ ไอซีตัวปัฟเฟอร์ตัวหลังจะยังมีข้อมูลของคอลัมน์ที่แล้วอยู่จะทำให้ข้อมูลที่ส่งออกมาเกิดการผิดพลาดได้

จากปัญหาที่ได้กล่าวมา เราจึงใช้ไอซีบัฟเฟอร์เพิ่มอีกหนึ่งชุดเพื่อใช้ในการรักษาระดับข้อมูล ดังแสดงในรูปที่ 3.5

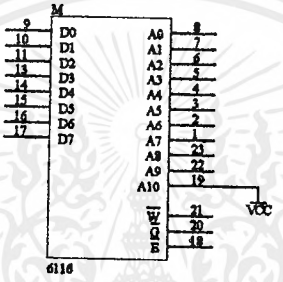


รูปที่ 3.5 การต่อวงจรรักษาระดับข้อมูลโดยใช้ไอซีบัฟเฟอร์สองชุด

ในวงจรส่วนนี้ขาที่ 1 ของไอซีบัฟเฟอร์ทุกตัวจะถูกต่อลงกราวด์นตลอดเวลาทุกตัวเพื่อให้ไอซีทำงานตลอดเวลา โดยสามารถแบ่งส่วนของวงจรนี้ได้เป็นไอซีบัฟเฟอร์ชุดที่ 1 จะทำหน้าที่รับข้อมูลจากแรม โดยข้อมูลที่ได้รับมาจะเป็นข้อมูลที่ใช้ทำการแสดงในแนวคอลัมน์ถัดไป โดยขาแลทช์ (ขาที่ 11) ของไอซีบัฟเฟอร์แต่ละตัวจะได้รับสัญญาณมาจากวงจรถอดรหัสในแนวโรว์ จากขา Y0-Y12 ตามลำดับ เพื่อจะเป็นการรับข้อมูลจากแรมในแอดเดรสที่ xx0H ถึง xxCH ส่วนไอซีบัฟเฟอร์ชุดที่ 2 จะทำหน้าที่รักษาระดับข้อมูลที่กำลังแสดงอยู่ในแนวคอลัมน์ขณะนั้น โดยขาแลทช์ของไอซีชุดที่ 2 ทุกตัวจะต่ออยู่กับวงจรถอดรหัสในแนวโรว์ จากขา Y16 เพื่อรับข้อมูลมาจากไอซีบัฟเฟอร์ชุดที่ 1 ที่ได้รับข้อมูลจากแรมครบทั้ง 12 ตัวแล้ว ดังนั้นข้อมูลในไอซีบัฟเฟอร์ชุดที่ 2 จะเปลี่ยนไปทุกๆครั้งที่วงจรถอดรหัสเส้นที่ 16 ทำงาน ทำให้LED สว่างได้นานเท่ากันและยังสว่างได้นานกว่าการใช้ไอซีบัฟเฟอร์ เพียงชุดเดียวอีกด้วย ซึ่งไอซีบัฟเฟอร์ชุดที่หนึ่งนั้น จะรับข้อมูลจากหน่วยความจำแรม ขนาด 2 K x 8 Bit โดยการส่งข้อมูลจากหน่วยความจำจะอธิบายในหัวข้อต่อไป

3.5 หน่วยความจำ

ในโครงงานนี้เราได้ออกแบบให้บอร์ดแสดงผลแต่ละบอร์ดมีความละเอียด 91x60 จุด และใช้วงจรถอดรหัสในแนวโรว์และคอลัมน์เป็น 16x64 ดังนั้นเราจำเป็นต้องใช้หน่วยความจำสำหรับเก็บข้อมูลภาพเป็น 16x64 = 1024 ไบท์ (Byte) ซึ่งเป็นขนาดของหน่วยความจำที่เราจำเป็นต้องใช้ แต่ในปัจจุบันไอซีแรมที่มีขนาด 1 กิโลไบท์ หาได้ยากมากในท้องตลาดและมีราคาไม่ต่างจากไอซีแรมขนาด 2 กิโลไบท์ มากนัก เราจึงเลือกใช้ไอซีแรมเบอร์ 6116 ซึ่งมีขนาด 2 กิโลไบท์แทน โดยทำการต่อขาแอดเดรส A10 ลงกราวด์นเพื่อใช้หน่วยความจำเพียงแค่ 1 กิโลไบท์ แทนดังรูปที่ 3.6



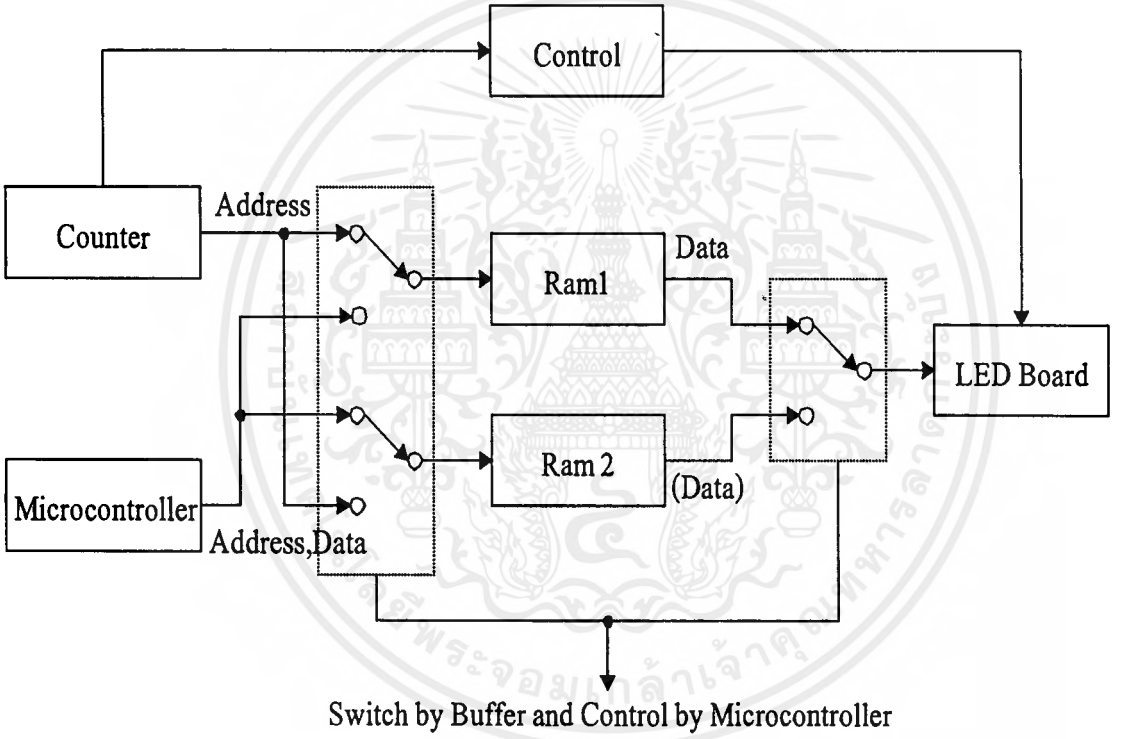
รูปที่ 3.6 ลักษณะของขาสัญญาณแรมเบอร์ 6116

วงจรส่วนนี้เราจะแบ่งหน่วยความจำออกเป็น 2 ตัวคือ

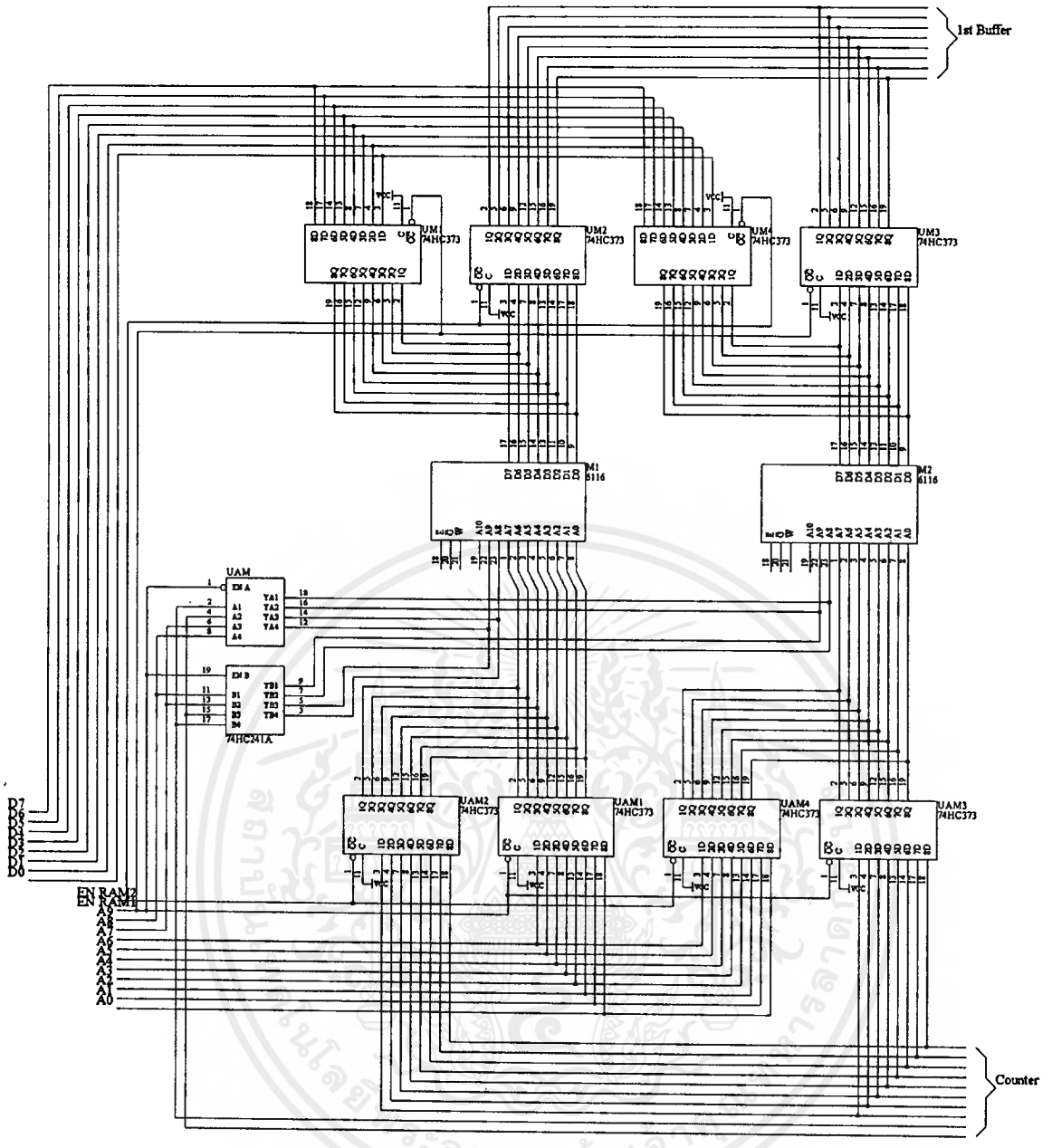
1. หน่วยความจำ ขณะส่งข้อมูลภาพปัจจุบัน
2. หน่วยความจำ รับข้อมูลภาพถัดไป จากไมโครคอนโทรลเลอร์

หน่วยความจำทั้งสองส่วนจะทำหน้าที่สลับกันไปในการแสดงภาพแต่ละครั้ง โดยที่ในขณะที่หน่วยความจำตัวหนึ่งอยู่ในช่วงการอ่านข้อมูลและจะส่งข้อมูลให้กับ Buffer เพื่อทำการแสดงผล หน่วยความจำตัวนี้จะได้รับแอดเดรสจากวงจรนับที่จะมีค่าเพิ่มขึ้นตามสัญญาณนาฬิกา ทำให้ข้อมูลจากแรมตัวนี้มีค่าเปลี่ยนไปตามแอดเดรสที่ได้รับ ซึ่งสัมพันธ์กับการ Latch Buffer ที่ควบคุมโดยวงจรถอดรหัส ทำให้มีการแสดงผลเป็นภาพที่ถูกเก็บไว้ในหน่วยความจำตัวนี้ ในขณะเดียวกันหน่วยความจำอีกตัวหนึ่งจะอยู่ในช่วงการเขียนข้อมูลจากไมโครคอนโทรลเลอร์ โดยไมโครคอนโทรลเลอร์จะทำการส่งทั้งข้อมูลและแอดเดรสให้กับแรมตัวนี้เพื่อเขียนข้อมูลภาพที่ต้องการจะแสดงในลำดับต่อไปลงไปในหน่วยความจำ เมื่อไมโครคอนโทรลเลอร์ได้ทำการเขียนข้อมูลลงไปในแรมตัวนี้เสร็จแล้วและแรมตัวแรกได้ทำการแสดงภาพไปเป็นระยะเวลาที่ได้กำหนดไว้แล้ว ไมโครคอนโทรลเลอร์ก็จะทำการสลับให้แรมตัวที่สองทำการส่งข้อมูลออกไปแสดงผลเป็นภาพที่ได้เขียนข้อมูลเอาไว้และทำการ

เขียนข้อมูลที่ต้องการแสดงผลต่อไปลงไปในแรมตัวแรกแทน ทำอย่างนี้สลับกันไปก็จะได้ภาพที่สามารถเปลี่ยนแปลงไปได้เรื่อยๆ ซึ่งการที่จะทำการสลับแรมไปมานี้ก็จำเป็นต้องใช้ Buffer เพื่อที่จะช่วยกำหนดทิศทางของแรมแต่ละตัวว่าต้องการให้แรมตัวไหนติดต่อกับไมโครคอนโทรลเลอร์และแรมตัวไหนทำการส่งข้อมูลไปแสดงผล โดยในวงจรนี้เราจะใช้ไอซีเบอร์ 74HC241A เข้าช่วยด้วย ซึ่งเป็นไอซีบัฟเฟอร์ขนาด 4 บิทสองชุด ในตัวเดียวกัน ในการสลับแรมที่จะส่งข้อมูลไปแสดงผลและทำการเขียนข้อมูล สามารถแสดงเป็นบล็อกไดอะแกรมการสลับแรมได้ดังรูปที่ 3.7 และรูปวงจรการสลับแรมรูปที่ 3.8



รูปที่ 3.7 บล็อกไดอะแกรม (Block diagram) แสดงการสลับแรม



รูปที่ 3.8 วงจรที่ช่วยในการสลับข้อมูลของแรมสองตัว

3.6 วงจรขับกระแส

ในวงจรส่วนนี้จะแบ่งออกเป็นส่วนการขับกระแสตามแนวโรว์ และส่วนสแกนตามแนวคอลัมน์ ส่วนขับกระแสตามแนวโรว์จะต่อจากไอซีบัฟเฟอร์ที่มาจากวงจรรักษาระดุบข้อมูล โดยถ้าเอาท์พุทของไอซีออกเป็นลอจิก 1 หมายถึงหลอดไฟไม่สว่าง และถ้าเอาท์พุทออก เป็น 0 หลอดไฟจะสว่าง โดยวงจรนี้ได้ออกแบบกระแสไว้มากที่สุด 100 มิลิแอมป์ ต่อ LED 1 หลอดโดยจะต้องใช้กระแสไฟขนาด 0.1 X 91 = 9.1 แอมป์ ในการจ่ายไฟทั้งหมดต่อ 1 แถว ได้ออกแบบโดยใช้

ทรานซิสเตอร์เบอร์ BC558B ในการจ่ายกระแสไฟให้ LED แต่ละดวง และจากวงจรจะต้องกำหนดค่าความต้านทาน เพื่อป้องกันความเสียหายต่อทรานซิสเตอร์ โดยคำนวณจากค่าอัตราขยาย (Beta)ของทรานซิสเตอร์ คือ 200 , กระแสไฟที่กำหนดไว้มีค่าเท่ากับ Ic = 100 มิลิแอมป์ โดยใช้ Ib = 500 มิลิแอมป์ โดยที่ทรานซิสเตอร์ เบอร์ BC558B นี้สามารถทนกระแสไฟสูงสุด 100 มิลิแอมป์ และต้องใช้

$$\text{Rbias} = (\text{Vcc}-0.7)/\text{Ib}$$

$$= (5-0.7)/500*10^{-6}$$

$$= 8600 \text{ โอห์ม}$$

ส่วนการขับกระแสตามแนวตั้ง จะรับสัญญาณจากวงจรถอดรหัสในแนวคอลัมน์ (6-64 Output Active High) เพื่อมาขับทรานซิสเตอร์ เบอร์ TIP 100 ซึ่งมีค่าอัตราการขยายขั้นต่ำ 750 โดยต้องใช้

$$\text{Rbias} = (5\text{ V} - 0.7) * h_{fe} / \text{Ic}$$

$$= (5-0.7) * 750 / 9.1$$

$$= 340 \text{ โอห์ม}$$

ค่า V_{CC} จะคำนวณได้จาก

$$V_{CC} = V_{CE1} + V_{CE2} + V_{\text{LED ที่ 100 mA}}$$

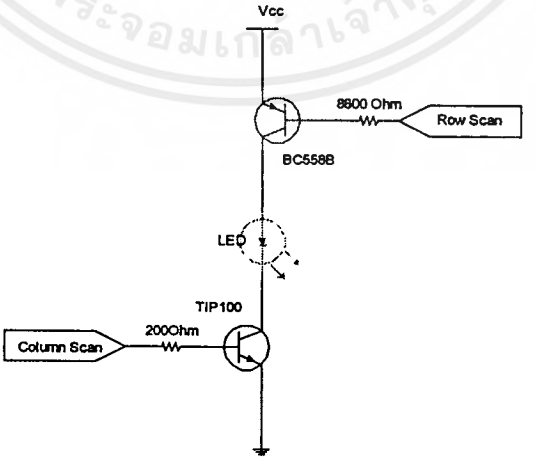
$$= 0.2 + 2.5 + 2.8$$

$$= 5.5 \text{ V}$$

โดยที่สามารถคำนวณได้จากวงจรขับกระแส ไม่เช่นนั้นทรานซิสเตอร์จะทำงานตลอดเวลา

$$V_{CC} < 5\text{ V} + 0.7 = 5.7 \text{ V}$$

ซึ่งจากวงจรและค่าความต้านทานที่คำนวณได้ทั้งหมด สามารถแสดงได้ดังรูปที่ 3.8



รูปที่ 3.9 วงจรในส่วนขับกระแสของบอร์ดแสดงผล

3.7 ส่วนแสดงผล

เรากำหนดขนาดเป็น 91x60 จุด ประมาณความกว้างต่อความยาว มีสัดส่วนเป็นสองต่อสาม และต่อกันแบบเมตริกท์ โดยในโครงานนี้เราเลือกใช้ LED สำเร็จขนาด 5x7 จุดที่ต่อกันในลักษณะเมตริกท์ในตัวของมันเอง มีขนาดเล็ก และยังสะดวกในการเข้าถึงข้อมูล โดยออกแบบลายวงจรเพียงหน้าเดียว เพราะเนื้อที่ในการเขียนลายวงจรมีขนาดจำกัด และอีกด้านหนึ่งของเมตริกท์นี้จะมีกระแสไหลผ่านสูงมาก (ประมาณ 9.5 แอมป์ ตามที่ได้จากการคำนวณ) จึงจำเป็นต้องใช้สายไฟเชื่อมเข้าช่วยเพื่อนำกระแสและยัง เป็นการต่อลายวงจรส่วนที่เหลืออีกด้วย

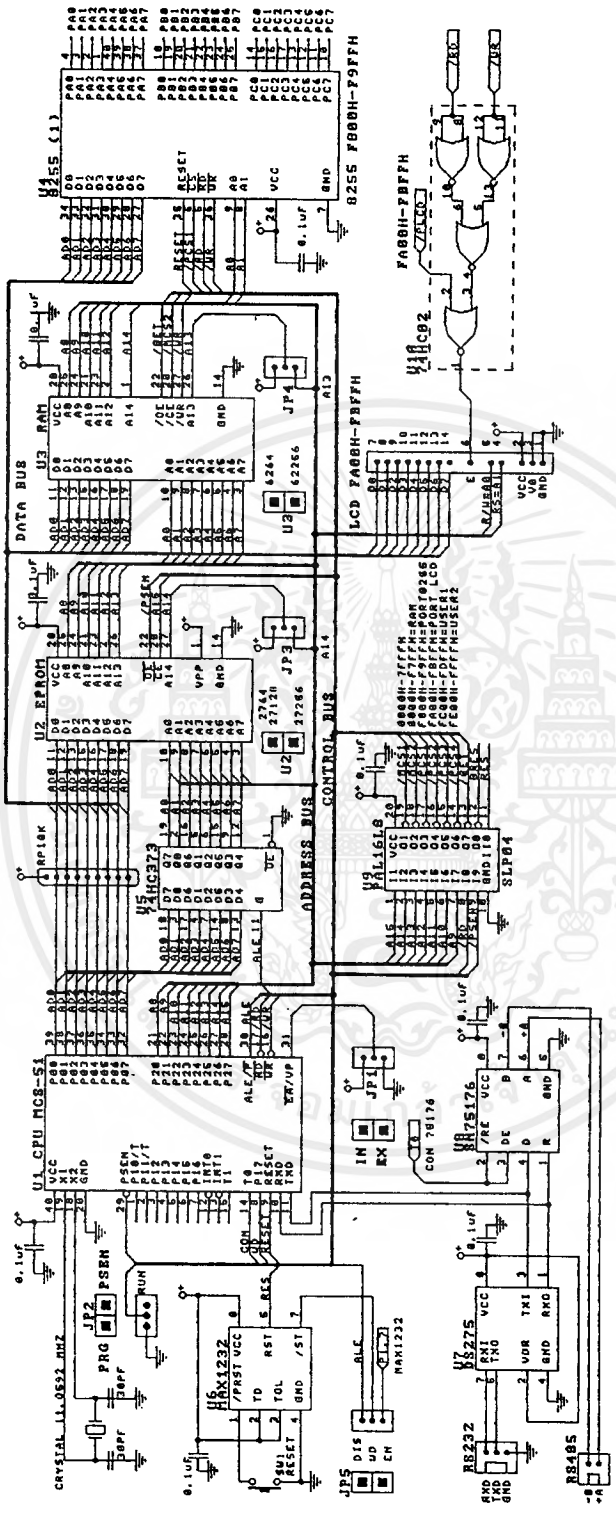
3.8 ส่วนไมโครคอลโทรลเลอร์

เป็นวงจรควบคุมการแสดงผลของบอร์ดแสดงผลไฟวิ่งแบบต่างๆ เช่น เลื่อนภาพขึ้น-ลง เลื่อนภาพไปทางซ้าย-ขวา แสดงการกระพริบของภาพ และแสดงภาพนิ่ง โดยมีการเชื่อมต่อกับส่วนแสดงผล LED บอร์ด ด้วยการต่อพอร์ต 40 ขา ซึ่งจะประกอบไปด้วยขาต่างๆเช่น ขาแอดเดรส (ADDRESS) ขาข้อมูล (DATA) ขาควบคุม (CONTROL)

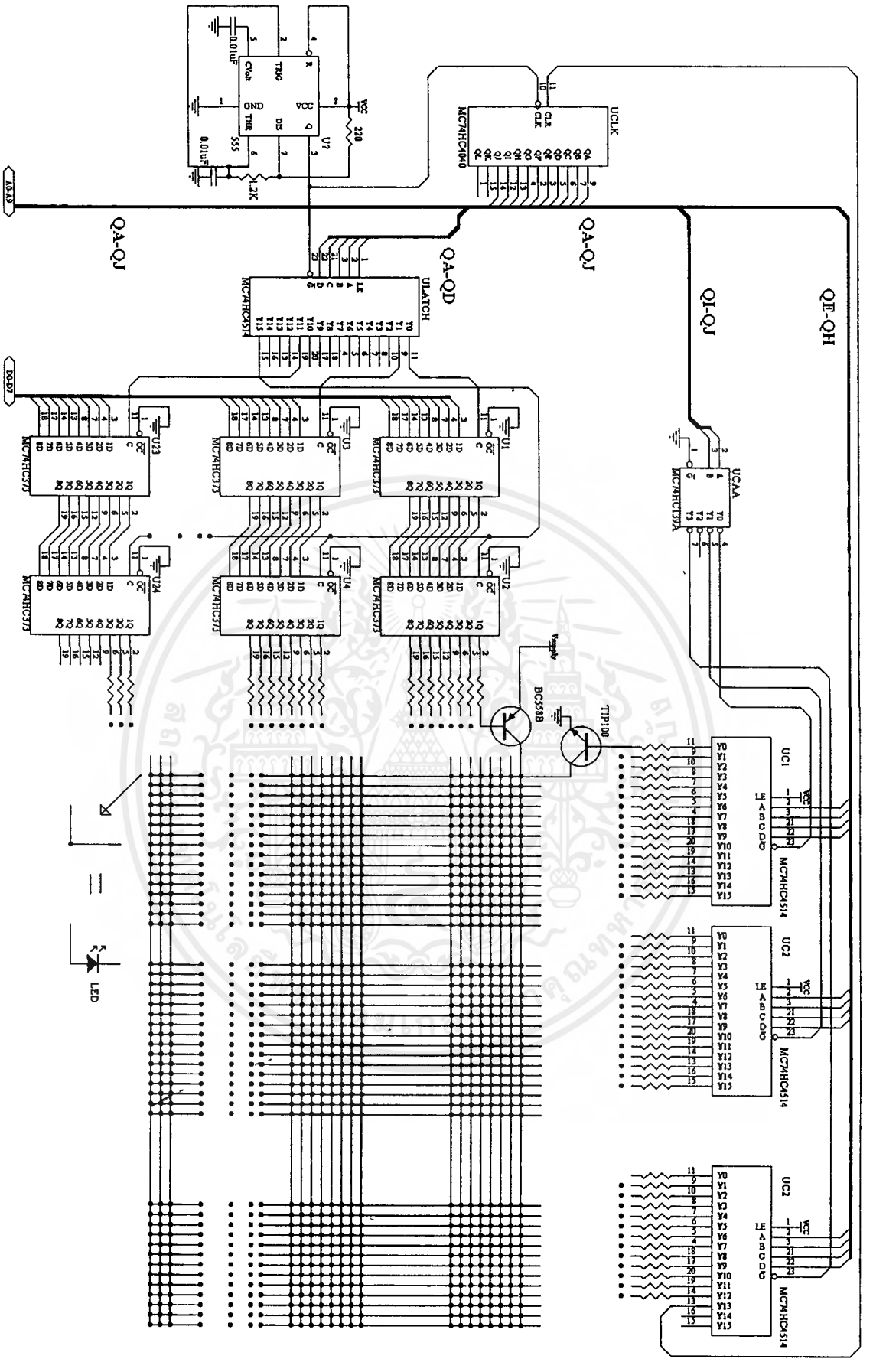
ซึ่งการเชื่อมต่อจะผ่านไอซีเบอร์ 8255 ทำให้ต้องต่อขา ไอซี 8255 จำนวน 2 ตัว โดยมีการกำหนดขาพอร์ตต่าง ๆ ดังนี้

- พอร์ต P1 ของขา 8255 ตัวที่ 1 กำหนดเป็น ขาแอดเดรส A0-A7
- พอร์ต P2 ของขา 8255 ตัวที่ 1 กำหนดเป็น ขาข้อมูล D0-D7
- พอร์ต P3 ของขา 8255 ตัวที่ 1 กำหนดเป็น ขาแอดเดรส A8-A9
- พอร์ต P1 ของขา 8255 ตัวที่ 2 กำหนดเป็น อ่านและเขียนข้อมูล แรม1และ แรม 2
- พอร์ต P2 ของขา 8255 ตัวที่ 2 กำหนดเป็น ขาติดต่อ แรม 1 และ แรม 2

โดยการทำงานได้ออกแบบให้ ใช้ อีพรอมอีมูเลเตอร์(EPROM EMULATOR) ในการรับส่งข้อมูลภาพมาจากการแปลงภาพจากโปรแกรมทางคอมพิวเตอร์ส่วนบุคล โดยใช้การเขียนโปรแกรมแล้วรับส่งข้อมูลภาพผ่านทางพอร์ตอนุกรม



รูปที่ 3.10 วงจรส่วนไมโครคอลโทรลเลอร์



รูปที่ 3.11 วงจรรวม

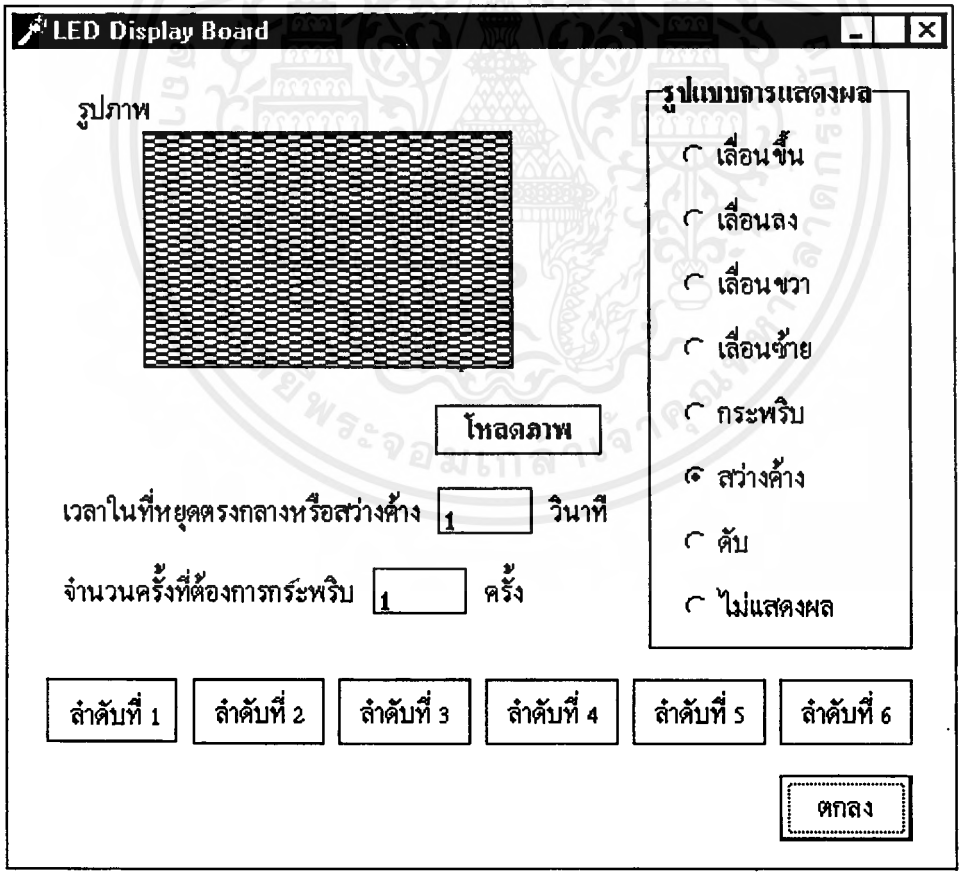
บทที่ 4
ส่วนโปรแกรม

สำหรับโครงงานนี้ได้ทำการแบ่งส่วนของโปรแกรมออกเป็น 2 ส่วนคือ

1. ส่วนโปรแกรมเคลไฟล์
2. ส่วนโปรแกรมไมโครคอลโทรลเลอร์

4.1 โปรแกรมเคลไฟล์

ในส่วนของโปรแกรมเคลไฟล์ จะเป็นส่วนที่ติดต่อการใช้งานกับผู้ใช้โดยตรง โดยเป็นส่วนที่ทำการกำหนดรูปแบบภาพที่ต้องการแสดงผล ซึ่งได้ทำการออกแบบโปรแกรมให้มีลักษณะการใช้งานเป็นดังรูป



รูปที่ 4.1 รูปแสดงการใช้งานโปรแกรมเคลไฟล์

จากการออกแบบ เราจะสามารถเลือกการแสดงผลได้ 6 ลำดับหรือ 6 รูป และในแต่ละรูปที่จะนำมาแสดงต้องทำการเลือกรูปที่ต้องการนำมาแสดงผล โดยการกดที่ปุ่ม " โหลดภาพ " แล้วทำการเลือกไฟล์รูปภาพที่ต้องการ ซึ่งรูปภาพที่สามารถนำมาแสดงได้ต้องมีลักษณะดังนี้

- เป็นไฟล์ตระกูล บิทแมบ (Bitmap file)
- มีชนิดของสีเป็นแบบขาว-ดำ
- มีขนาดความละเอียดของภาพเป็น 91x60 จุด

โดยรูปภาพนี้สามารถสร้างได้จาก โปรแกรมกราฟฟิกที่มีอยู่โดยทั่วไป

ในแต่ละลำดับรูป เราสามารถที่จะเลือกรูปแบบการแสดงผลได้ถึง 7 รูปแบบ คือ

1. เลื่อนขึ้น
2. เลื่อนลง
3. เลื่อนซ้าย
4. เลื่อยขวา
5. กระพริบ
6. สว่างค้าง
7. ดับ

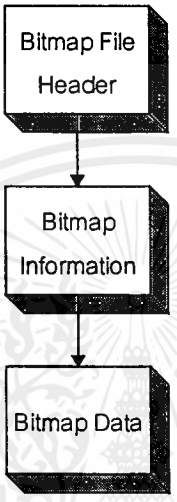
และสามารถที่จะกำหนดค่าเวลาที่ต้องการแสดงผล แสดงจำนวนครั้งของการกระพริบได้ในช่องจำนวนครั้งที่กระพริบและ เวลาที่จะหยุดตรงกลางหรือสว่างค้าง

เมื่อกำหนดค่าต่างๆของแต่ละลำดับตามความต้องการแล้ว ซึ่งอาจจะไม่ครบทั้ง 6 ลำดับก็ได้ ก็ทำการกดปุ่ม " ตกลง " โดยโปรแกรมก็จะทำการแปลงข้อมูลภาพและรูปแบบการแสดงผลแบบต่างๆ เป็นรูปแบบเฮ็กซ์ไฟล์ (Hex file) เพื่อใช้ในการส่งข้อมูลเข้าสู่ส่วนของไมโครคอลโทรลเลอร์ต่อไป โดยในขั้นตอนนี้เราต้องกำหนดชื่อไฟล์ที่ต้องการบันทึก

ส่วนการส่งข้อมูลภาพเข้าสู่บอร์ดไมโครคอนโทรลเลอร์ จะใช้วิธีส่งข้อภาพในรูปแบบเฮ็กซ์ไฟล์ที่ได้จากส่วนโปรแกรมเดลไฟล์ ผ่านอีพรอมอีมูเลเตอร์ที่ต่ออยู่กับไมโครคอนโทรลเลอร์ ซึ่งข้อมูลที่จะส่งผ่านอีพรอมอีมูเลเตอร์นี้จะรวมถึงโปรแกรมของส่วนไมโครคอนโทรลเลอร์ด้วย จากนั้นจึงเป็นหน้าที่ของส่วนไมโครคอนโทรลเลอร์ที่จะทำการประมวลผลและแสดงภาพตามที่ได้กำหนดไว้

4.1.1 โครงสร้างของภาพ บิทแมบ

ภาพ บิทแมบ (bitmap) จะมีการจัดเก็บตามขนาดของสี ถ้าสีมากจะทำให้ภาพที่จัดเก็บมีขนาดใหญ่ไปด้วย ภาพบิทแมบ จะประกอบไปด้วยส่วนระบุชนิดของภาพ ส่วนข้อมูลของภาพ ขนาดสี อื่นๆ และสุดท้ายจะเป็นจุดภาพที่จะนำมาแสดงผล



รูปที่ 4.2 โครงสร้างของภาพ bitmap

ตารางที่ 4.1 ส่วนหัวของบิทแมบไฟล์ (Bitmap File Header)

Byte #	Data	Details
1-2	File type	Must be ASCII text " BM"
3-6	Size of the file	In double word (32 bit interger)
7-10	Reserved for future use	Must be Zero
11-14	Byte offset to bitmap data	Offset from the Bitmap File Header (i.e.,the start of the file)

ตารางที่ 4.2 รายละเอียดของบิทแมบ (Bitmap Information)

Byte #	Data	Details
1-4	Number of byte in header	ขนาดของส่วนหัว 40 byte
5-8	Width of Bitmap	ความกว้างของภาพ (pixel)
9-12	Height of Bitmap	ความสูงของภาพ (pixel)
13-14	Number of color planes	ต้องเท่ากับ 1
15-16	Number of bit per pixel	ส่วนที่บอกสีของภาพ 1= 2สี , 4=16 สี , 8=256 สี , 24=16 สี
17-20	Type of compression	0 = ไม่มีการอัดภาพ 1 = 8 bit per pixel 2 = 4 bit per pixel
21-24	Size of image	ไม่ได้ใช้งาน
25-28	Horizontal resolution	ไม่ได้ใช้งาน
29-32	Vertical resolution	ไม่ได้ใช้งาน
33-36	Number of color indexes used by the Bitmap	ไม่ได้ใช้งาน
37-40	Number of color indexes important for displaying Bitmap	ไม่ได้ใช้งาน
41	Blue color value	ค่าสีน้ำเงินของภาพ
42	Green color value	ค่าสีเขียวของภาพ
43	Red color value	ค่าสีแดงของภาพ
44	Reserved for future use	ต้องเป็น 0
...	... Remaining color palette entries	บอกค่าสีต่างๆ 4 bit เหมือนกันไปจนหมดภาพ

4.2 โปรแกรมไมโครคอลโทรลเลอร์

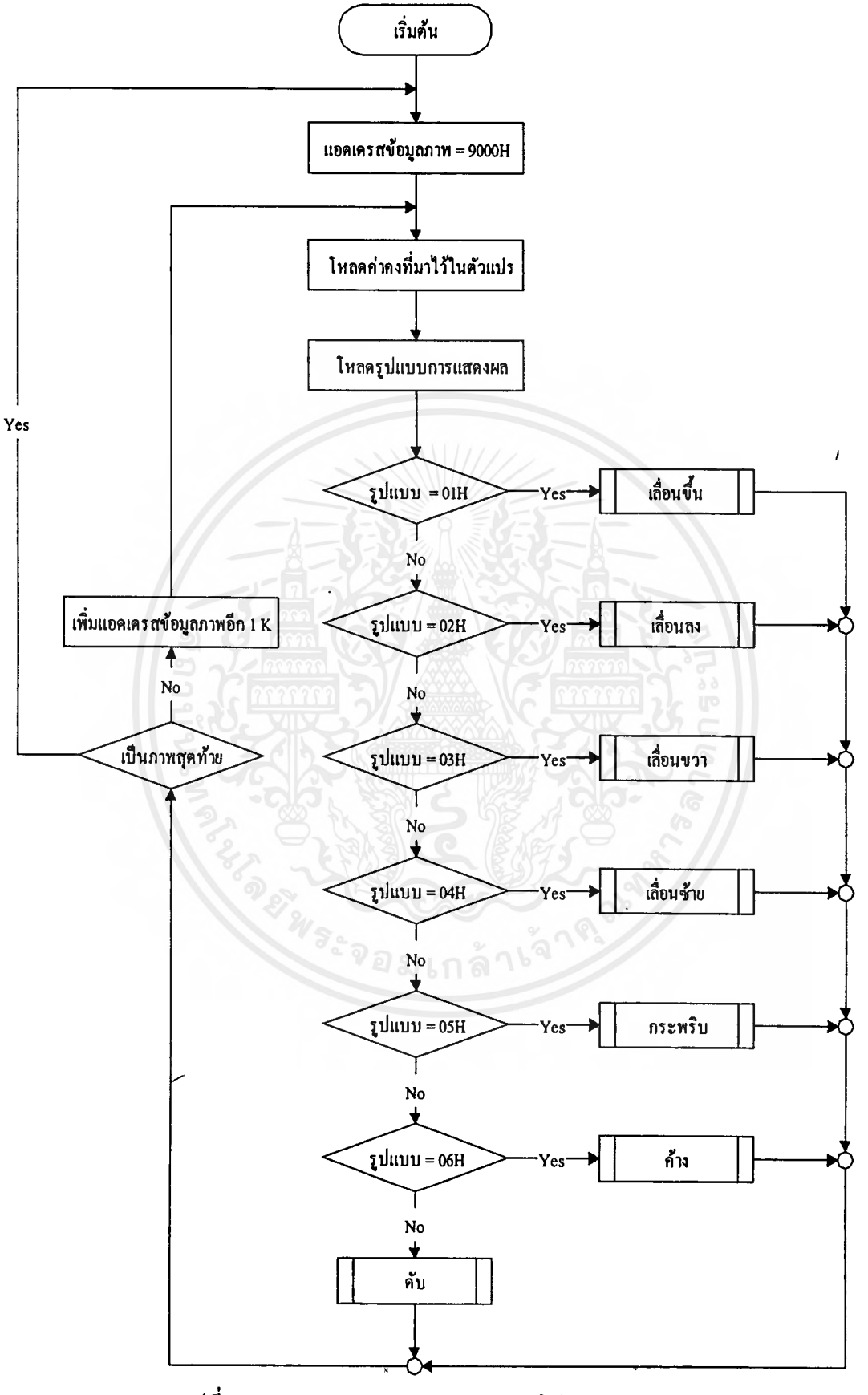
ส่วนโปรแกรมไมโครคอลโทรลเลอร์ คือส่วนที่ทำการรับข้อมูลภาพและรูปแบบของการแสดงผลมาทำการประมวลผลให้ได้ตามที่ได้กำหนดไว้ในส่วนของโปรแกรมเดลไฟล์

สามารถแบ่งส่วนโปรแกรมไมโครคอลโทรลเลอร์ได้เป็นส่วนๆตามหน้าที่ดังนี้

1. โปรแกรมหลัก
2. โปรแกรมย่อยสำหรับการเลื่อนภาพขึ้น
3. โปรแกรมย่อยสำหรับการเลื่อนภาพลง
4. โปรแกรมย่อยสำหรับการเลื่อนภาพไปทางขวา
5. โปรแกรมย่อยสำหรับการเลื่อนภาพไปทางซ้าย
6. โปรแกรมย่อยสำหรับแสดงภาพกระพริบ
7. โปรแกรมย่อยสำหรับแสดงภาพนิ่ง
8. โปรแกรมย่อยสำหรับการไม่แสดงผล

4.2.1 โปรแกรมหลัก

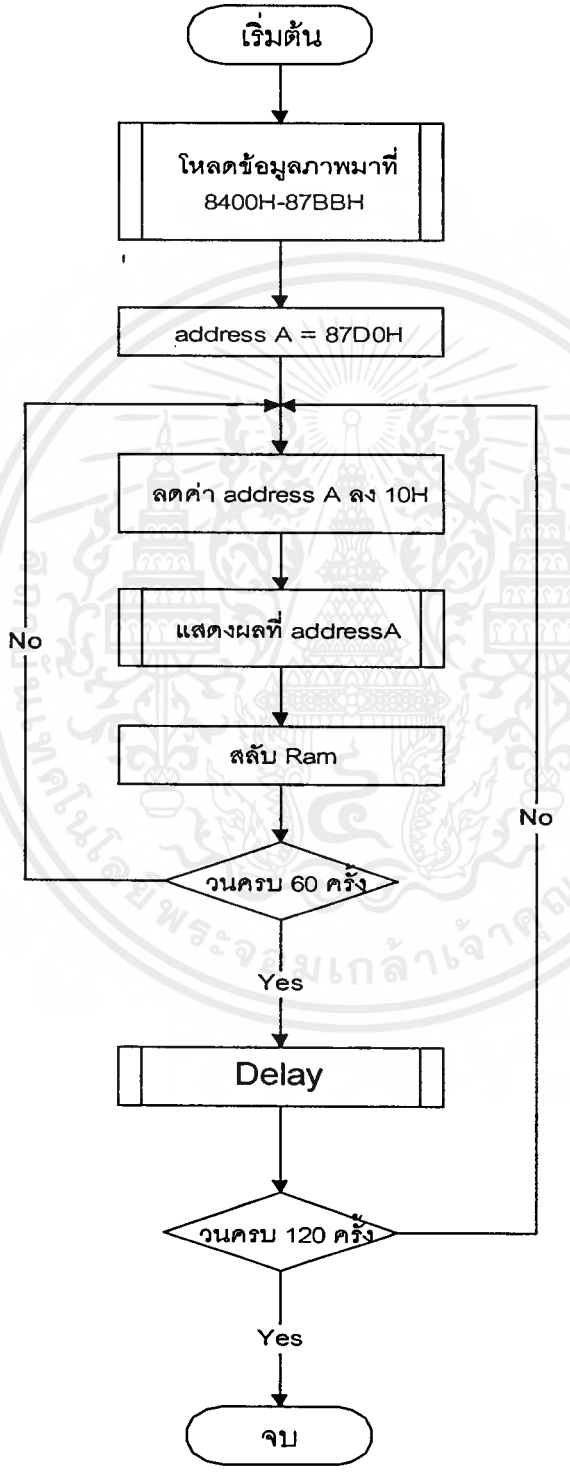
ทำหน้าที่เชื่อมโยงข้อมูลที่ได้รับจากส่วนโปรแกรมเดลไฟล์ และทำการตรวจสอบรูปแบบของการแสดงผล จากนั้นจึงทำการส่งที่อยู่ (Address) ของข้อมูลภาพและค่าเวลาในการแสดงผล ไปยังโปรแกรมย่อยต่างๆ ตามรูปแบบการแสดงผลของข้อมูลภาพนั้นๆ



รูปที่ 4.3 แผนภาพแสดงการทำงานของโปรแกรมหลัก

4.2.2 โปรแกรมย่อยสำหรับการเลื่อนภาพขึ้น

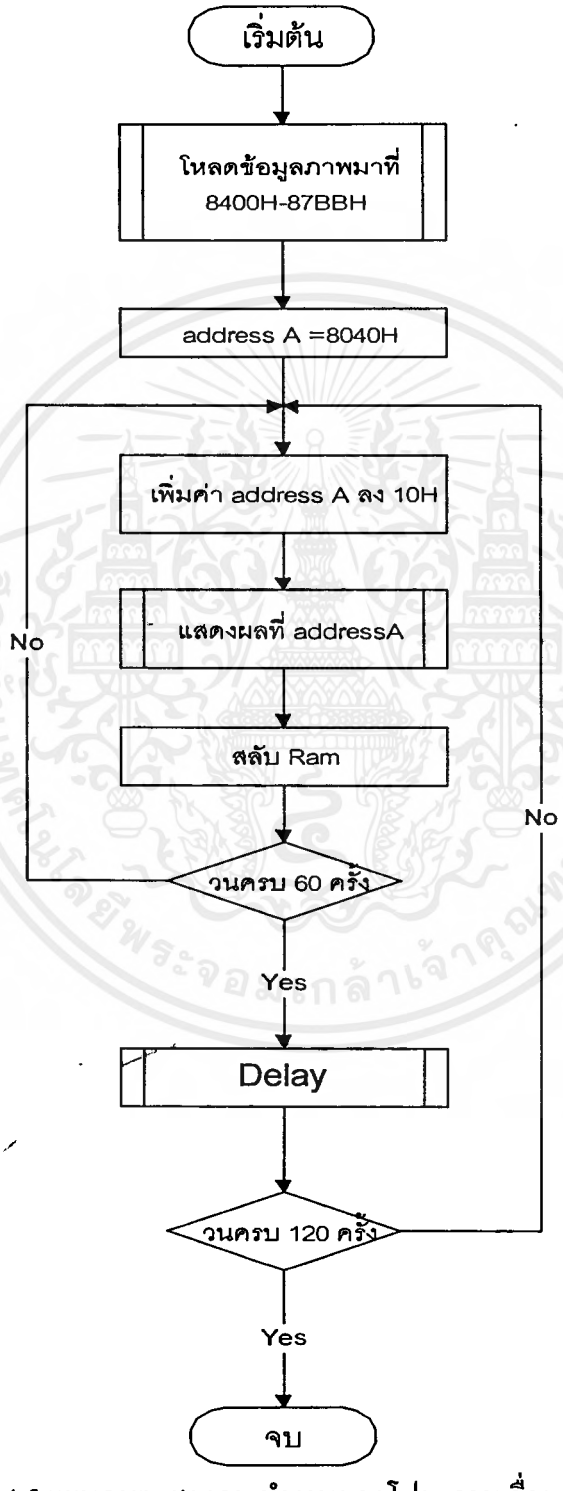
ทำหน้าที่ประมวลผลข้อมูลภาพที่ได้รับให้แสดงผลในลักษณะเลื่อนขึ้นจากด้านล่าง และหยุดที่จุดกึ่งกลางระยะเวลาตามที่ได้กำหนดไว้ และจึงเลื่อนขึ้นต่อไปจนหมดข้อมูลภาพ



รูปที่ 4.4 แผนภาพแสดงการทำงานของโปรแกรมเลื่อนภาพขึ้น

4.2.3 โปรแกรมย่อยสำหรับการเลื่อนภาพลง

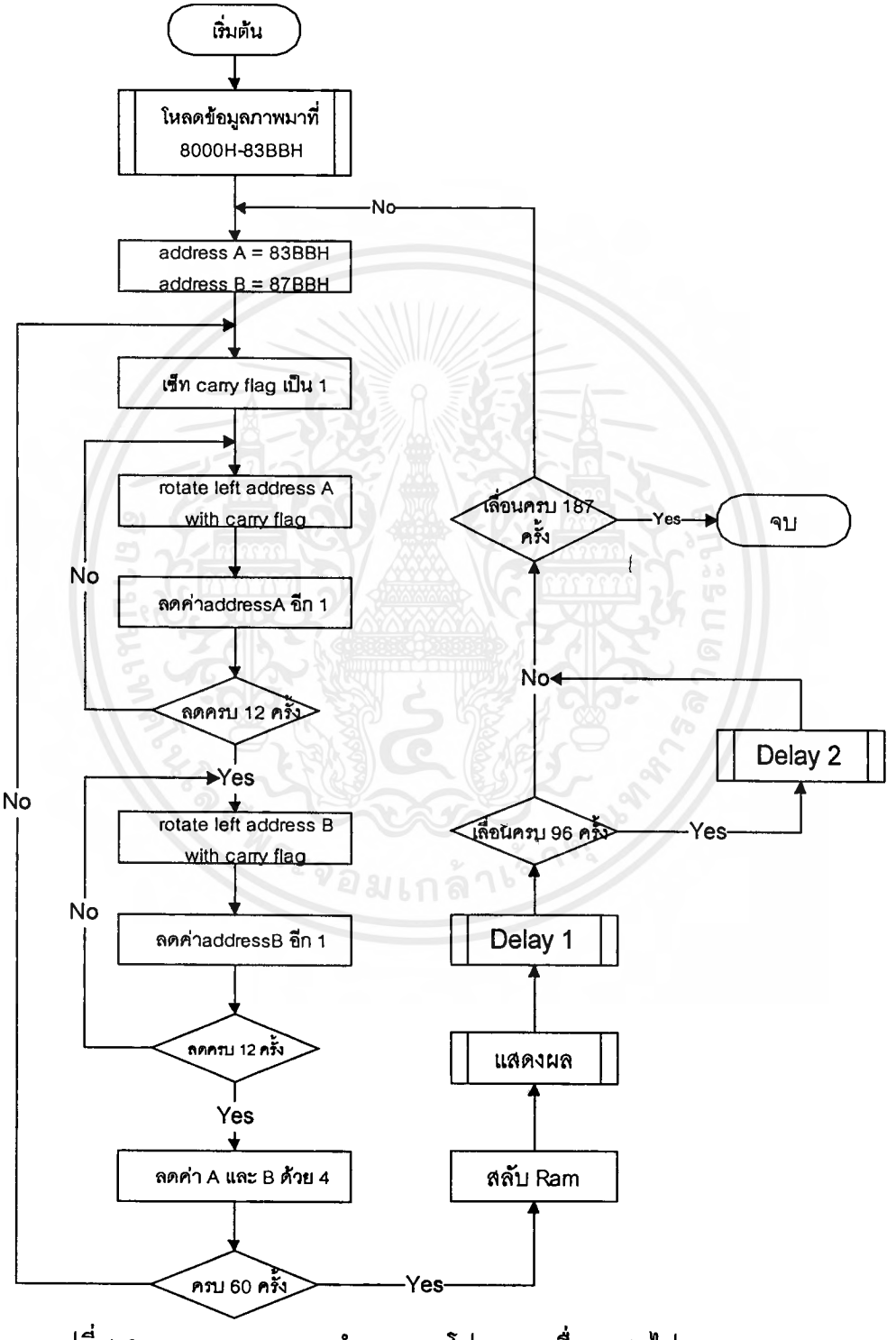
ทำหน้าที่ประมวลผลข้อมูลภาพที่ได้รับให้แสดงผลในลักษณะเลื่อนลงจากด้านบน และหยุดที่จุดกึ่งกลางระยะเวลาตามที่ได้กำหนดไว้ และจึงเลื่อนลงต่อไปจนหมดข้อมูลภาพ



รูปที่ 4.5 แผนภาพแสดงการทำงานของโปรแกรมเลื่อนภาพลง

4.2.4 โปรแกรมย่อยสำหรับการเลื่อนภาพไปทางขวา

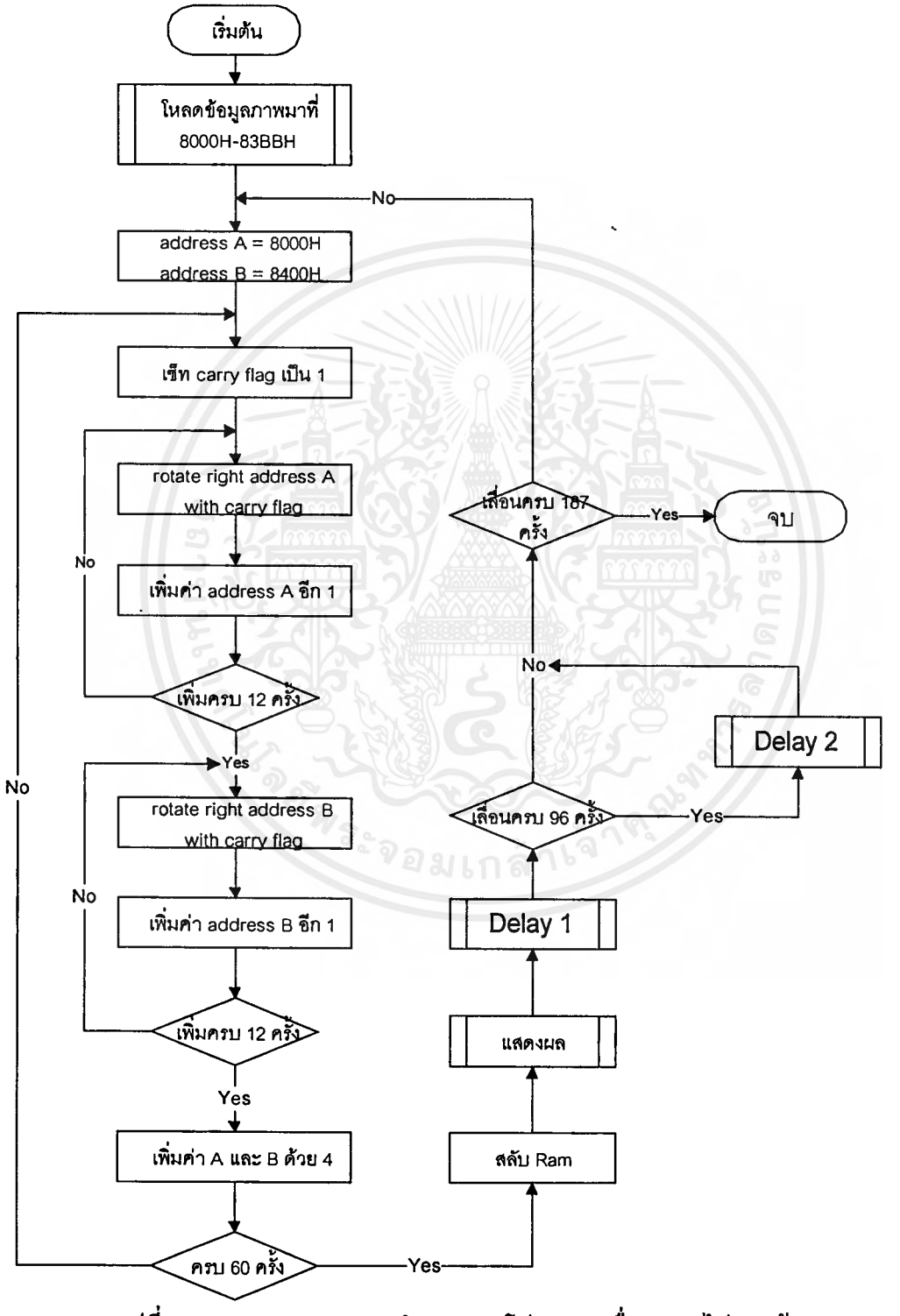
ทำหน้าที่ประมวลผลข้อมูลภาพที่ได้รับให้แสดงผลในลักษณะเลื่อนภาพไปทางขวา และหยุดที่จุดกึ่งกลางระยะเวลาตามที่ได้กำหนดไว้ และจึงเลื่อนภาพต่อไปจนหมดข้อมูลภาพ



รูปที่ 4.6 แผนภาพแสดงการทำงานของโปรแกรมเลื่อนภาพไปทางขวา

4.2.5 โปรแกรมย่อยสำหรับการเลื่อนภาพไปทางซ้าย

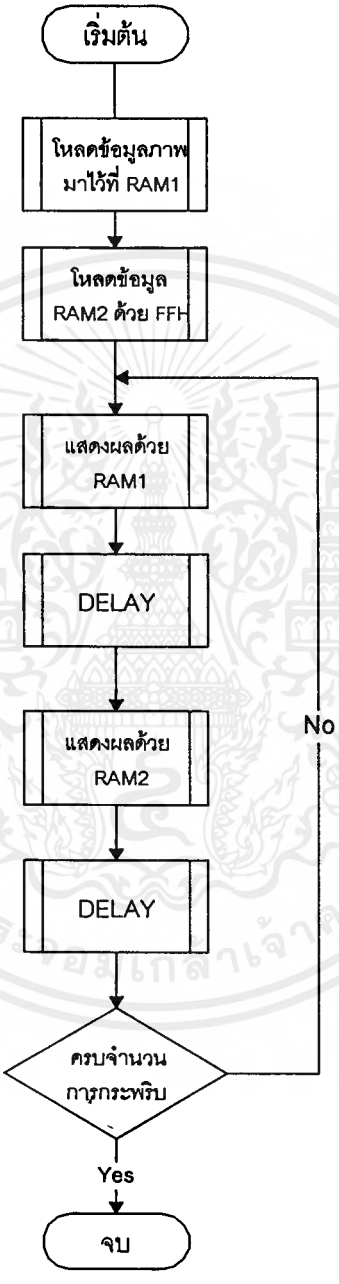
ทำหน้าที่ประมวลผลข้อมูลภาพที่ได้รับให้แสดงผลในลักษณะเลื่อนภาพไปทางซ้าย และหยุดที่จุดกึ่งกลางระยะเวลาตามที่ได้กำหนดไว้ และจึงเลื่อนภาพต่อไปจนหมดข้อมูลภาพ



รูปที่ 4.7 แผนภาพแสดงการทำงานของโปรแกรมเลื่อนภาพไปทางซ้าย

4.2.6 โปรแกรมย่อยสำหรับแสดงภาพกระพริบ

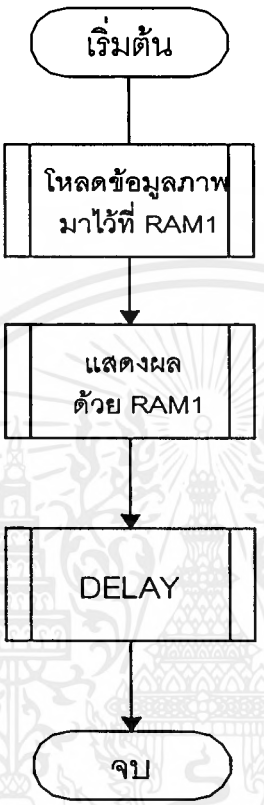
ทำหน้าที่แสดงข้อมูลภาพที่ได้รับเป็นระยะเวลาที่ได้กำหนดไว้ สลับกับไม่แสดงผล ตามจำนวนครั้งที่กำหนด



รูปที่ 4.8 แผนภาพแสดงการทำงานของโปรแกรมแสดงภาพกระพริบ

4.2.7 โปรแกรมย่อยสำหรับการแสดงภาพนิ่ง

ทำหน้าที่แสดงข้อมูลภาพที่ได้รับเป็นระยะเวลาที่ได้กำหนดไว้



รูปที่ 4.9 แผนภาพแสดงการทำงานของโปรแกรมแสดงภาพนิ่ง

4.2.8 โปรแกรมย่อยสำหรับการไม่แสดงผล

ทำหน้าที่ทำให้บอร์ดแสดงผล ไม่แสดงผล เป็นระยะเวลาที่ได้กำหนดไว้ โดยการเขียนข้อมูล #0FFH ลงไปในแรมแทนที่จะเป็นข้อมูลภาพ

บทที่ 5

ผลการทดลอง

5.1 คุณสมบัติ LED

กระแสที่จ่าย (mA)	โวลท์เตจตกคร่อม LED (V)
10	1.8
20	1.9
30	2.0
35	2.1
70	2.4
100	2.8

5.2 ผลการทดลอง

ได้ทำการแบ่งการทดลองออกเป็น 2 ส่วนตามขอบเขตของโครงงานคือ

1. ส่วนฮาร์ดแวร์
2. ส่วนซอฟแวร์

5.2.1 ส่วนฮาร์ดแวร์

จากการออกแบบวงจรทั้งส่วนบอร์ดแสดงผล LED และส่วนควบคุมและขับกระแส เมื่อนำทั้งสองส่วนมาต่อร่วมกัน แล้วทำการทดลองให้แสดงผล พบว่าบอร์ดแสดงผลสามารถแสดงผลได้ตามที่ได้ออกแบบไว้ โดยสามารถควบคุมตำแหน่งของจุดที่ต้องการแสดงผลได้ทุกจุด โดยการกำหนดจากตำแหน่งแอดเดรสของบอร์ดแสดงผลที่ได้กำหนดไว้ตั้งแต่การออกแบบบอร์ด แสงสว่างที่แสดงผลได้ค่อนข้างมาก ภาพมีการพริ้วเกิดขึ้นเล็กน้อย และยังเกิดสัญญาณรบกวนบ้าง

และพบว่าความถี่ที่เหมาะสมของวงจรสร้างสัญญาณนาฬิกาที่ทำให้ภาพที่แสดงผลนิ่งที่สุดทำที่สามารถทำได้คือ 50 Hz

5.2.2 ส่วนซอฟแวร์

สามารถแบ่งออกได้เป็น 2 ส่วนคือ

1. ส่วนโปรแกรมเคลไฟล์

สามารถทำการโหลดไฟล์รูปที่ต้องการทำการแสดงผลและเลือกรูปแบบการแสดงผลได้ โดยจากการออกแบบได้กำหนดให้สามารถทำการกำหนด รูปที่ใช้แสดงผลและรูปแบบแสดงผลได้ครั้งละ 6 รูปและออกแบบให้เลือกรูปแบบการแสดงผลได้ 7 รูปแบบ ปรากฏว่าโปรแกรมสามารถทำการแปลงข้อมูลภาพและรูปแบบการแสดงผลแบบต่างๆเป็นรูปแบบเฮ็กซ์ไฟล์ได้อย่างถูกต้อง สำหรับในส่วนการตรวจสอบการแปลงข้อมูลภาพ ได้ทำการตรวจสอบจากรูปแบบของเฮ็กซ์ไฟล์ที่ทำการแปลงออกมาแล้วว่าถูกต้องจริงหรือไม่ โดยหาความสัมพันธ์ของรูปแบบภาพ และรูปแบบเอ็กซ์ไฟล์ที่ได้

2. ส่วนโปรแกรมไมโครคอลโทรลเลอร์

เนื่องจากเราได้ทำการแบ่งส่วนไมโครคอลโทรลเลอร์ออกเป็นโปรแกรมหลักและโปรแกรมย่อยต่างๆ เราจึงสามารถนำโปรแกรมเหล่านี้มาทำการทดสอบที่ละส่วน ทำให้ง่ายต่อการตรวจเช็ค ก็พบว่าในส่วนโปรแกรมหลักซึ่งทำหน้าที่ในการรับข้อมูลจากส่วนโปรแกรมเคลไฟล์ ทำการตรวจสอบรูปแบบ เพื่อนำไปประมวลผลแล้วส่งไปยังบอร์ดควบคุมการแสดงผลเพื่อทำการแสดงภาพนั้นๆที่บอร์ดแสดงผล LED ก็พบว่าสามารถทำงานได้จริง

สำหรับส่วนโปรแกรมย่อยต่างๆ ที่ทำหน้าที่ประมวลรูปแบบของการแสดงผลแบบต่างๆ ก็สามารถทำการ แสดงผลในรูปแบบต่างๆ ที่กำหนดได้

บทที่6
สรุปผลและวิจารณ์

6.1 ปัญหาที่พบและแนวทางในการแก้ไข

1. การกำหนดขอบเขตการทำงาน มีการเปลี่ยนแปลงหลายครั้งเนื่องจาก LED ที่เลือกใช้เป็นแบบเมตริกท์ จึงต้องมีการคำนวณความละเอียดที่จะเลือกใช้ในบอร์ดแสดงผลเพื่อให้เหมาะสมและสะดวกในการออกแบบ
2. อุปกรณ์ LED ที่ใช้ได้รับความอนุเคราะห์ จากอาจารย์ที่ปรึกษาจัดหามาให้ เนื่องจากต้องใช้จำนวนมาก จึงต้องเสียเวลาในการจัดหาให้ และยังต้องทำการเช็คตัวอุปกรณ์ว่าใช้งาน LED ได้ทุกดวงหรือไม่
3. การออกแบบวงจร มีความล่าช้าเนื่องจากต้องพบกับปัญหาหลายๆ อย่างในการออกแบบ ได้แก่ เรื่องอุปกรณ์ ต้องเลือกคุณสมบัติของอุปกรณ์ให้สามารถที่จะนำมาใช้งานได้อย่างมีประสิทธิภาพ แต่อุปกรณ์ บางเบอร์ไม่มีขาย หรือมีขายแต่เนื่องจากต้องการใช้งานจำนวนมาก จึงต้องมีการเช็คกับทางร้านค้าว่ามีพอหรือไม่ จึงเกิดความล่าช้า และยังต้องคำนึงเรื่องของราคาอีกด้วย จึงต้องมีการปรับเปลี่ยนตัวอุปกรณ์หลายครั้งเพื่อให้เหมาะสม และประหยัดที่สุด
4. การออกแบบลายปริ้นซ์ค่อนข้างทำได้ช้าเนื่องจากวงจรมีความซับซ้อนต้องใช้อุปกรณ์จำนวนมาก และต้องการออกแบบให้มีขนาดเล็กเพื่อเป็นการประหยัดทั้งราคาและเนื้อที่ของฮาร์ดแวร์
5. มีการพบการออกแบบวงจรบางส่วนผิด จึงต้องทำการหาทางแก้ไข เนื่องจากได้มีการต่อวงจรบางส่วนที่เกี่ยวข้องไปบ้างแล้ว
6. จากการทดลองพบว่า เมื่อนำบอร์ดส่วนแสดงผล LED ต่อเข้ากับส่วนขับกระแสและควบคุม LED ภาพที่ได้มีการสั่นพริ้วเล็กน้อย คาดว่าเกิดจากความถี่ที่ใช้งานต่ำเกินไปจึงต้องทำการแก้ไข
7. จากการที่โครงงานนี้จำเป็นต้องใช้ LED เป็นจำนวนมาก ซึ่งบางครั้งมีการเสียหายของ LED เกิดขึ้นในระหว่างการทดลอง แต่เนื่องจากบอร์ดมีขนาดจำกัด ทำให้การสับเปลี่ยน LED ตัวใหม่เป็นไปอย่างยากลำบากเพราะอาจจะทำให้เกิดความเสียหายแก่ LED ตัวข้างเคียงได้

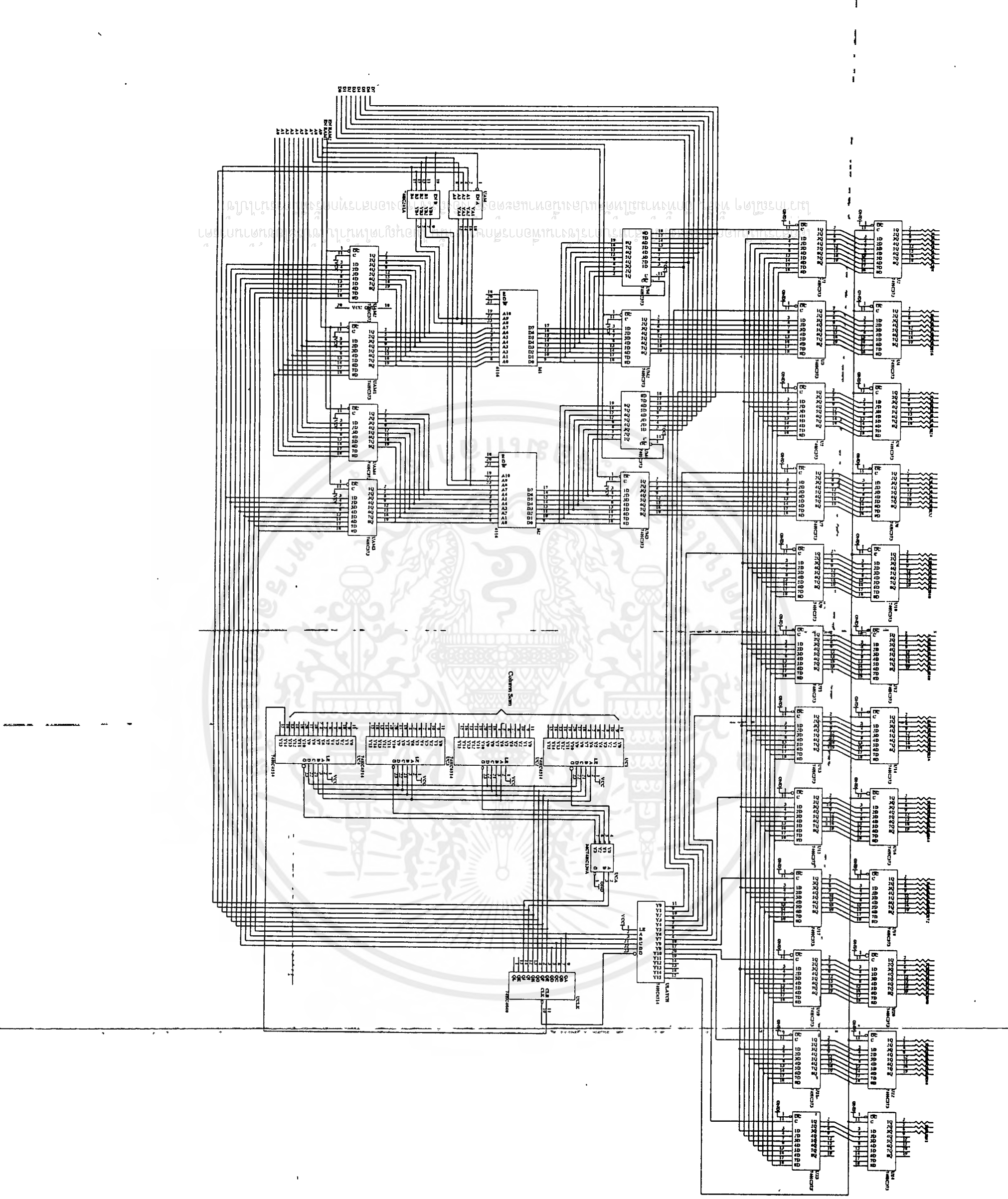
6.2 สรุปผลและวิจารณ์

จากการทดลองทำการแสดงผลด้วยบอร์ด LED แล้วก็พบว่าสามารถแสดงผลๆได้ตามต้องการ ความสว่างที่ได้ค่อนข้างสว่าง สามารถนำมาใช้งานได้จริง แต่ก็ยังเกิดข้อผิดพลาดขึ้นอีกหลายประการดังที่ได้กล่าวไปแล้วในหัวข้อ ปัญหาที่พบและแนวทางแก้ไขปัญหา ก็จะพบว่าปัญหาส่วนใหญ่ก็สามารถทำการปรับปรุงแก้ไขได้ แต่ความยากง่ายในการแก้ปัญหาก็จะแตกต่างกันไป ซึ่งผู้จัดทำก็ได้พยายามแก้ไขจนสุดความสามารถ และจากการที่นำเอาไมโครคอลโทรลเลอร์มาใช้ในการควบคุมร่วมกับโปรแกรมเดลไฟล์ ก็ทำให้ช่วยให้การใช้งานบอร์ดแสดงผลได้ง่ายขึ้น โดยสามารถนำมาใช้ควบคุมการแสดงผลได้ของ LED แสดงผลได้อย่างมีประสิทธิภาพ และยังสามารถติดต่อกับคอมพิวเตอร์ส่วนบุคคลได้ โดยที่เราสามารถควบคุมการติดหรือดับ และสามารถกำหนดลักษณะการแสดงผลเป็นตัวอักษรและรูปกราฟฟิกได้ การใช้งานโปรแกรมเดลไฟล์ก็ออกแบบมาให้เข้าใจวิธีการใช้งานได้ง่ายจึงสามารถจะนำปรับปรุงให้ใช้งานในชีวิตประจำวันได้จริง

สำหรับการเขียนโปรแกรมไมโครคอลโทรลเลอร์ก็ได้มีการแบ่งโปรแกรมออกเป็นโปรแกรมหลัก และโปรแกรมย่อยหลายๆโปรแกรมทำให้ง่ายต่อการเขียนโปรแกรม สะดวกในการตรวจเช็คโปรแกรม และสามารถนำไปพัฒนาปรับปรุงโปรแกรมให้ดียิ่งขึ้นต่อไปได้

ภาคผนวก ก

1. รูปวงจรรวม
2. รายการอุปกรณ์
3. ลายวงจรพิมพ์ของบอร์ดแสดงผล
4. ลายวงจรพิมพ์ของบอร์ดควบคุมด้านบน
5. ลายวงจรพิมพ์ของบอร์ดควบคุมด้านล่าง
6. การวางอุปกรณ์บนแผ่นวงจรพิมพ์
7. รูปชิ้นงาน

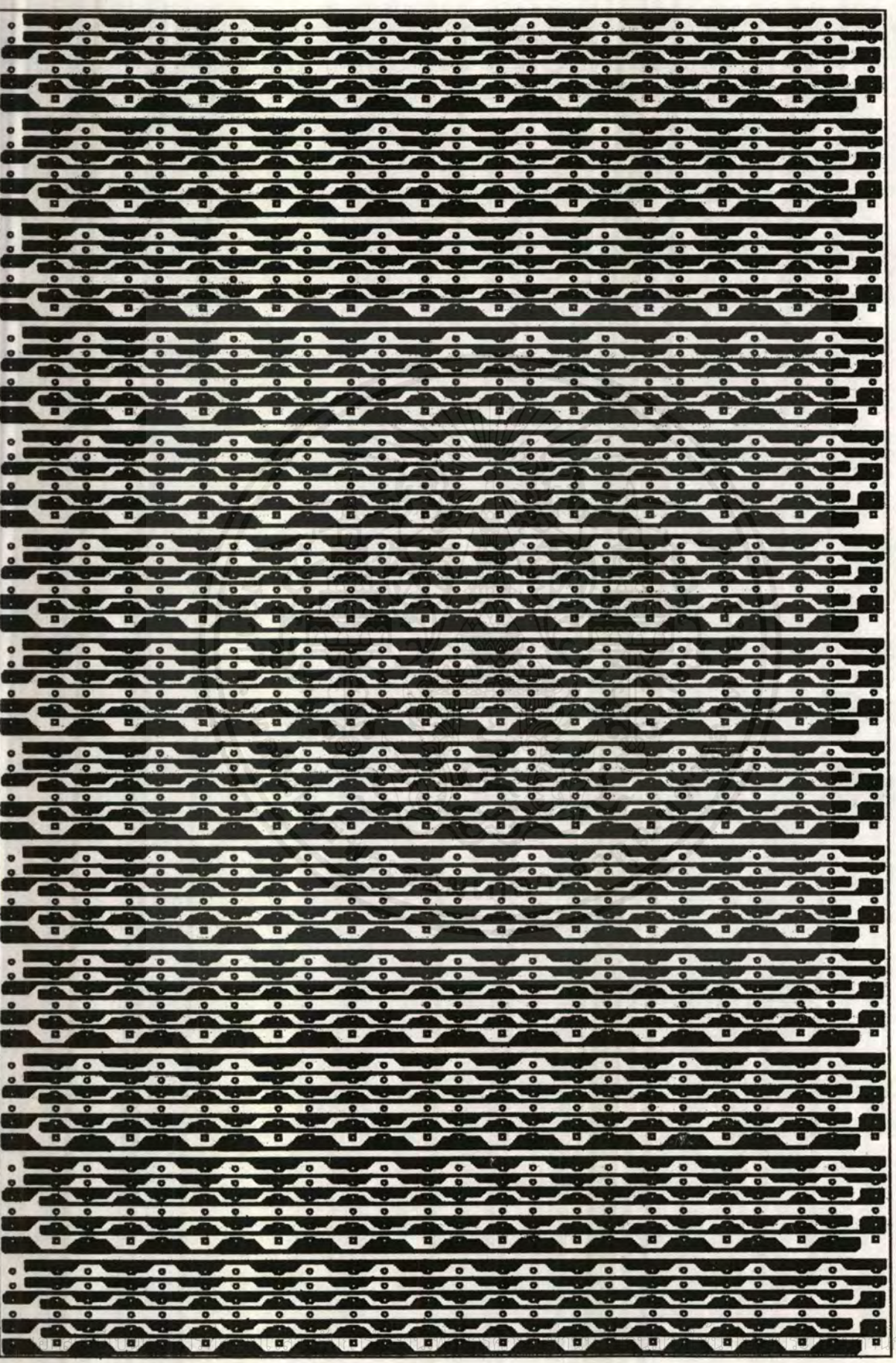


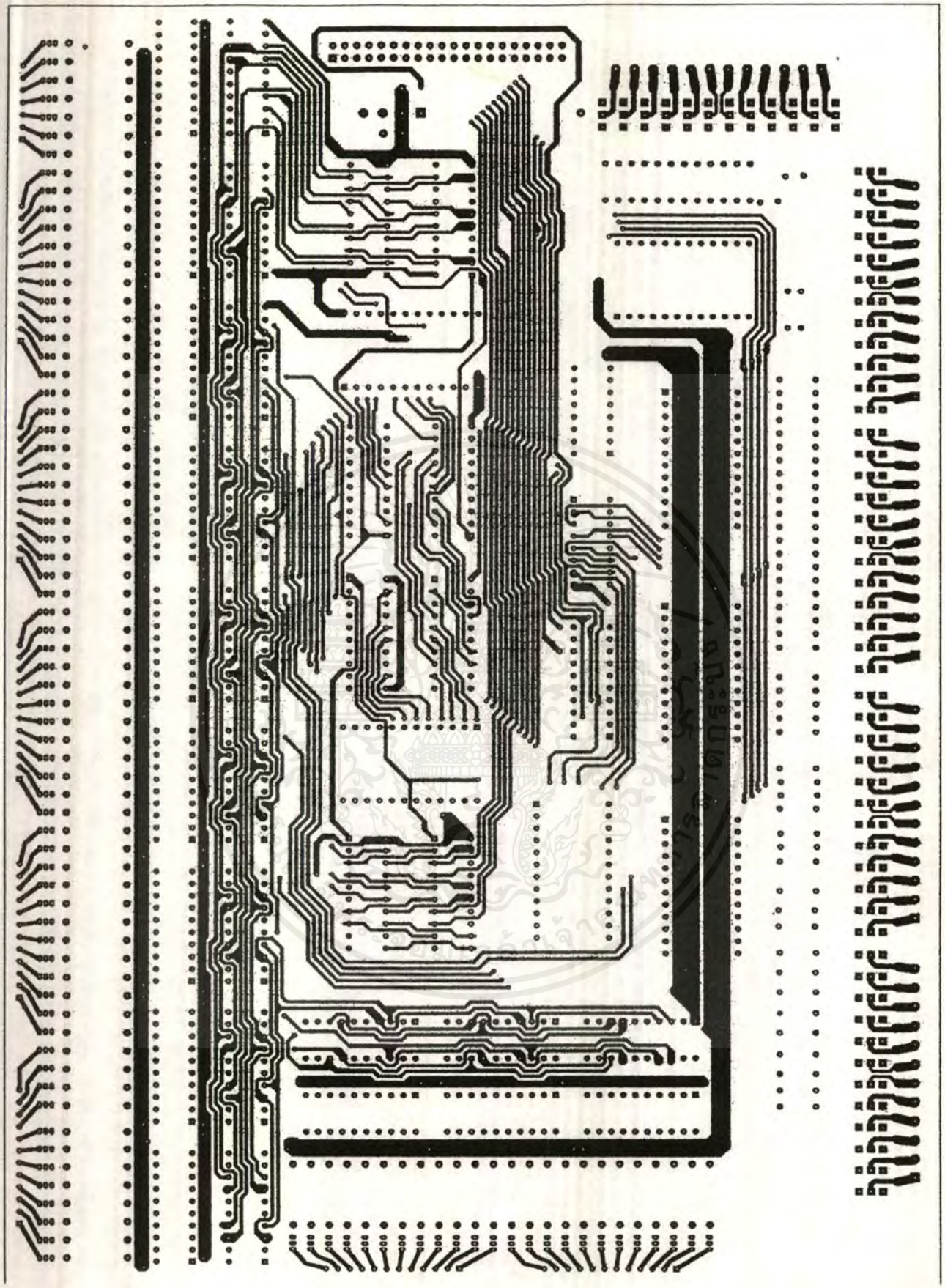
รูปวงจรรวม

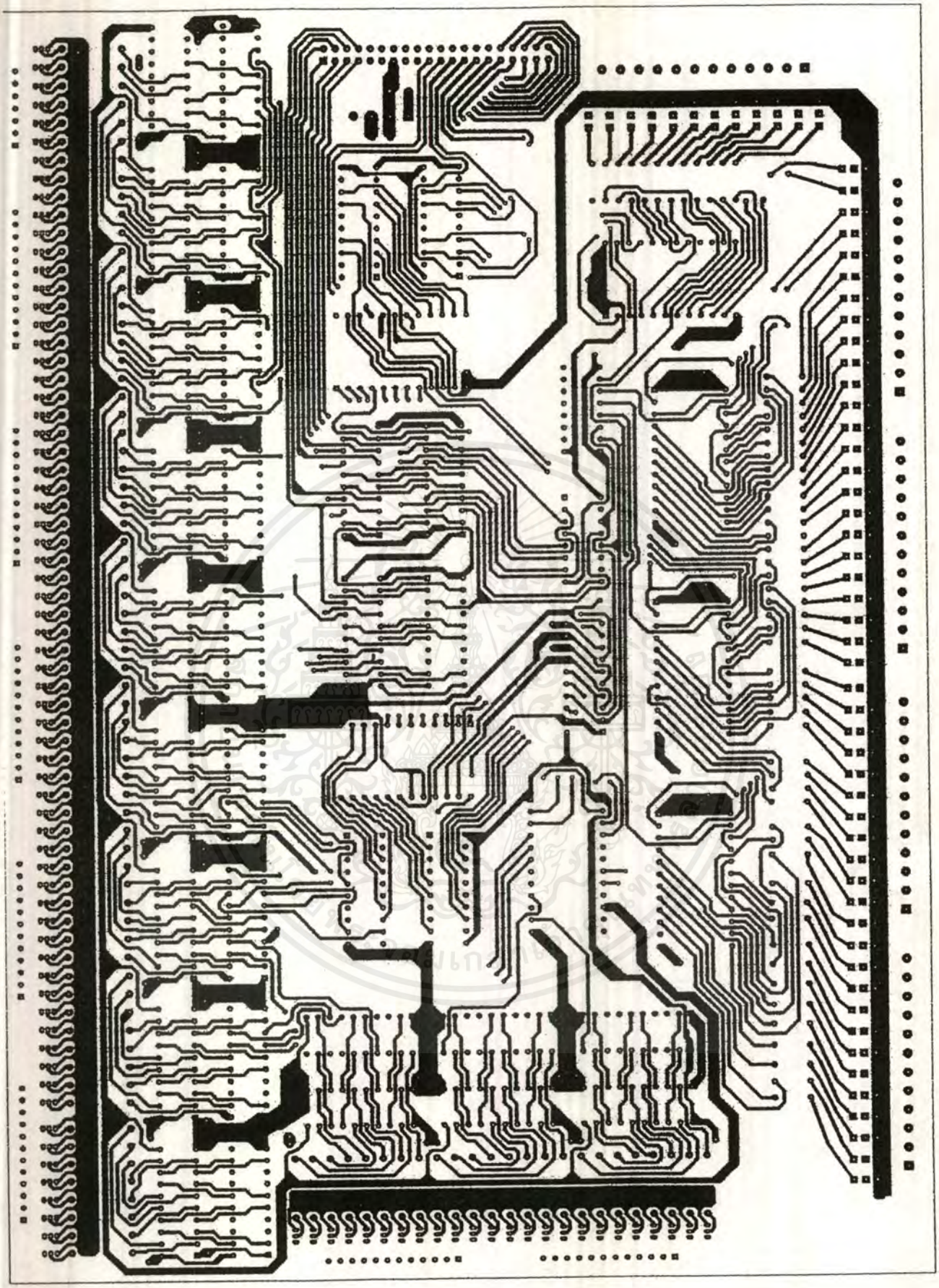
รายการอุปกรณ์

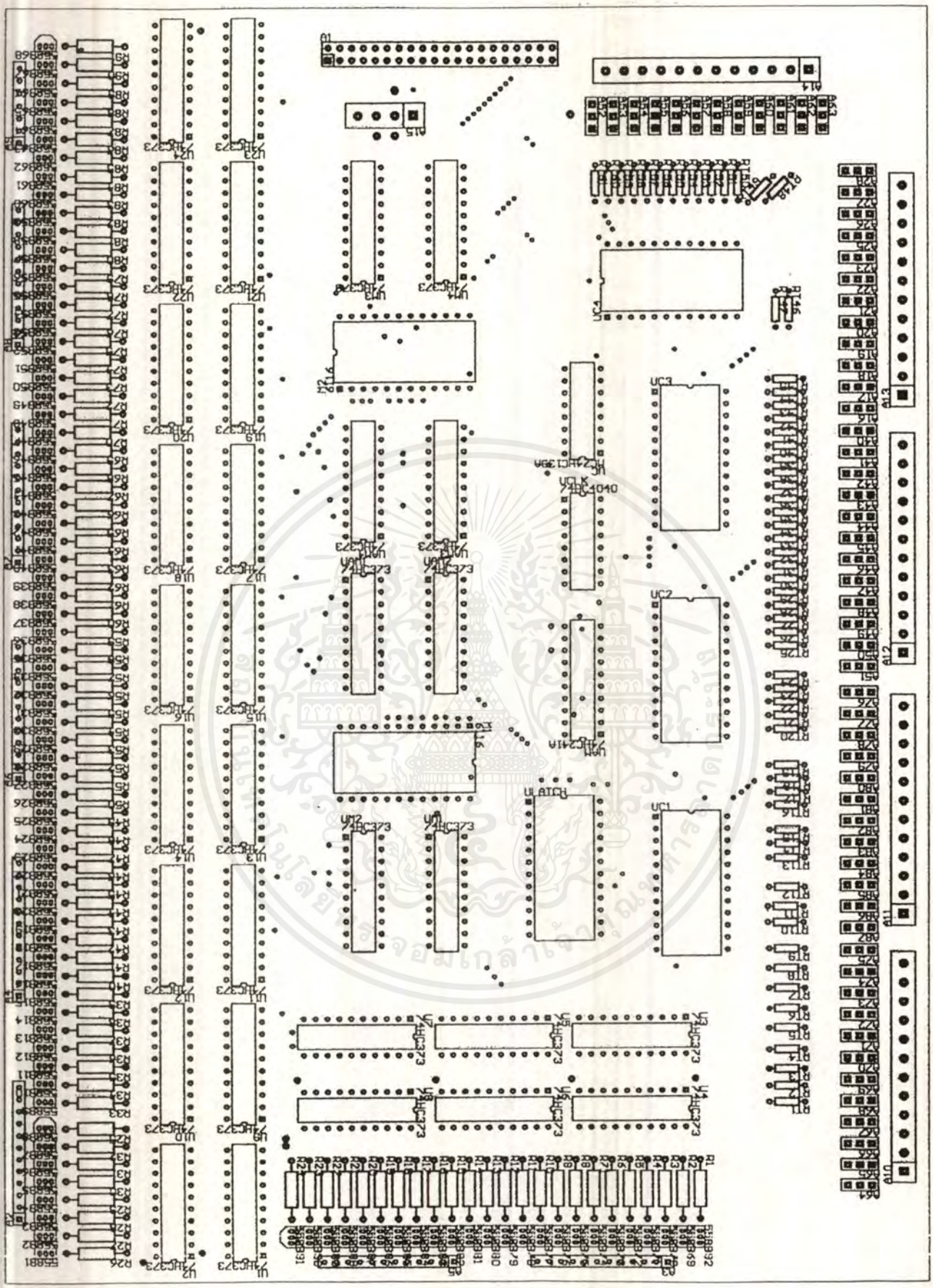
1.	Ram 6116	2	ตัว
2.	74HC373	32	ตัว
3.	74HC4514	5	ตัว
4.	74HC4040	1	ตัว
5.	74HC241A	1	ตัว
6.	74HC139	1	ตัว
7.	74HC154	5	ตัว
8.	LM555	1	ตัว

9. R = 1K ohm ¼ w 91 ตัว
10. R = 200 ohm ¼ w 60 ตัว
11. C = 0.1 F 2 ตัว
12. Connector ใหญ่ 12 ขา 5 ชุด
13. Connector ใหญ่ 4 ขา 2 ชุด
14. Connector เล็ก 12 ขา 7 ชุด
15. Connector เล็ก 7 ขา 1 ชุด
16. หม้อแปลง 0-6 V 1.5A, 0-3.9 V 9.5 A
17. บอร์ดไมโครคอลโทรลเลอร์
18. บอร์ดอีพรอมอีมูเลเตอร์

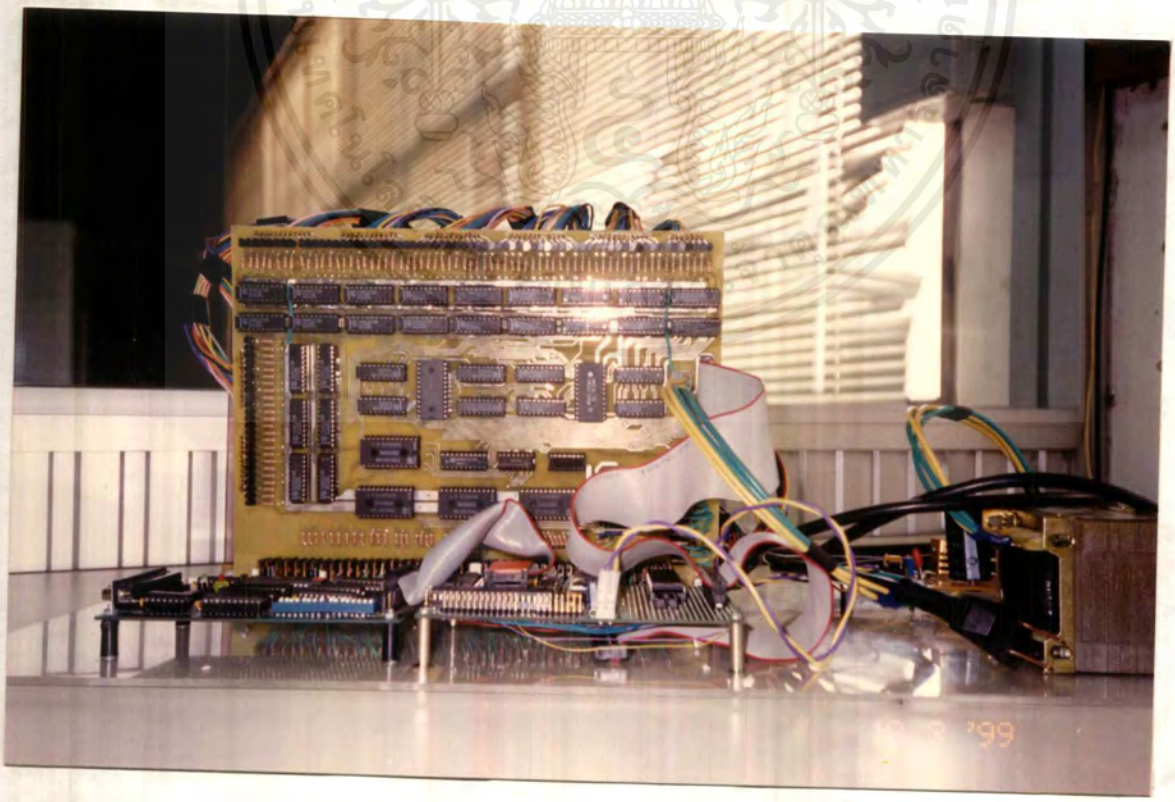








รูปแสดงชิ้นงานด้านหน้า



รูปแสดงชิ้นงานด้านหลัง

ภาคผนวก ข

โปรแกรมเดลไฟล์

```
unit led1;
interface
uses
  Windows, Messages, SysUtils, Classes, Graphics, Controls, Forms, Dialogs,
  ExtCtrls, StdCtrls, Buttons, Menus, ExtDlgs;

type
    TForm1 = class(TForm)
    SpeedButton1: TSpeedButton;
    SpeedButton2: TSpeedButton;
    SpeedButton3: TSpeedButton;
    SpeedButton4: TSpeedButton;
    SpeedButton5: TSpeedButton;
    SpeedButton6: TSpeedButton;
    SpeedButton7: TSpeedButton;
    GroupBox1: TGroupBox;
    Image1: TImage;
    Button1: TButton;
    Button2: TButton;
    OpenPictureDialog1: TOpenPictureDialog;
    Bevel1: TBevel;
    Label1: TLabel;
    Label2: TLabel;
    Edit1: TEdit;
    Edit2: TEdit;
    Label3: TLabel;
    Label4: TLabel;
    GroupBox2: TGroupBox;
    RadioButton1: TRadioButton;
```

```
    RadioButton2: TRadioButton;
    RadioButton3: TRadioButton;
    RadioButton4: TRadioButton;
    RadioButton5: TRadioButton;
    RadioButton6: TRadioButton;
    RadioButton7: TRadioButton;
    RadioButton8: TRadioButton;
    SaveDialog1: TSaveDialog;
    procedure Button1Click(Sender: TObject);
    procedure SpeedButton2Click(Sender: TObject);
    procedure SpeedButton3Click(Sender: TObject);
    procedure SpeedButton4Click(Sender: TObject);
    procedure SpeedButton5Click(Sender: TObject);
    procedure SpeedButton6Click(Sender: TObject);
    procedure SpeedButton7Click(Sender: TObject);
    procedure SpeedButton1Click(Sender: TObject);
    procedure FormCreate(Sender: TObject);
    procedure Button2Click(Sender: TObject);

  private
    procedure ChangeFont(SpStatus: integer);
    procedure Check1();
    procedure Check2();
    function ToHex1(Num: byte):char;
    function Swap(Num2: byte):byte;
    function First_End(Num3: byte):byte;
    { Private declarations }
  public
    { Public declarations }
  end;
```

```
var

  Blinktime: array[1..7] of word;
  Blink_Times: array[20..26] of word;
  code: integer;
  Data_Value: array[1..32] of byte;
  Data_Text: array[1..32] of char;
  Display_Type: array[20..26] of byte;
  Filename: array[1..7] of string[70];
  Form1: TForm1;
  ImageFile: file;
  Image: record
       File_type: array[1..2] of char;
       File_size: array[1..2] of word;
       Not_used1: array[1..12] of byte;
       Width: array[1..2] of word;
       Height: array[1..2] of word;
       Color_planes: word;
       Bits_per_pixel: word;
       Not_used2: array[1..32] of byte;
       Data: array[0..59,1..12] of byte;
       end;
  Image_Test: record
       File_type: array[1..2] of char;
       File_size: array[1..2] of word;
       Not_used1: array[1..12] of byte;
       Width: array[1..2] of word;
       Height: array[1..2] of word;
       Color_planes: word;
       Bits_per_pixel: word;
       end;
```

```
  Output_File: textfile;
  Output_Filename: string[60];
  Picturechange: array[1..7] of boolean;
  Picture_File: array[20..26] of string[60];
  S: string[11];
  Show_Time: array[20..26] of word;
  SpStatus: byte;
  Time: array[1..7] of word;
  Typecheck: array[1..7] of byte;

implementation

{$R *.DFM}

procedure TForm1.FormCreate(Sender: TObject);
var i: integer;
begin
    SpStatus:= 1;
    for i:= 1 to 7 do
    begin
      Typecheck[i]:= 8;
      Time[i]:= 1;
      Blinktime[i]:= 1;
      Picturechange[i]:= false;
      Filename[i]:='c:\begin.bmp';
    end;
    Edit1.text:= '1';
    Edit2.text:= '1';
end;

  procedure TForm1.Button1Click(Sender: TObject);
```

```
begin
    if OpenPictureDialog1.Execute then
    begin
     Filename[Spstatus]:= OpenPictureDialog1.FileName;
    end;
    Image1.Picture.LoadFromFile(Filename[Spstatus]);
end;

procedure TForm1.ChangeFont(SpStatus: integer);
begin
    case SpStatus of
        1: SpeedButton1.Font.Color:= clBlack;
        2: SpeedButton2.Font.Color:= clBlack;
        3: SpeedButton3.Font.Color:= clBlack;
        4: SpeedButton4.Font.Color:= clBlack;
        5: SpeedButton5.Font.Color:= clBlack;
        6: SpeedButton6.Font.Color:= clBlack;
        7: SpeedButton7.Font.Color:= clBlack;
    end;
end;

procedure TForm1.SpeedButton1Click(Sender: TObject);
begin
    Check1();
    val(Edit1.text,Time[SpStatus],code);
    val(Edit2.text,Blinktime[SpStatus],code);
    ChangeFont(SpStatus);
    SpeedButton1.Font.Color:= clBlue;
    SpStatus:= 1;
    Check2();
    str(Time[SpStatus],S);Edit1.Text:= S;
```

```
    str(Blinktime[SpStatus],S);Edit2.Text:= S;
    Image1.Picture.LoadFromFile(Filename[SpStatus]);
end;

procedure TForm1.SpeedButton2Click(Sender: TObject);
begin
    Check1();
    val(Edit1.text,Time[SpStatus],code);
    val(Edit2.text,Blinktime[SpStatus],code);
    ChangeFont(SpStatus);
    SpeedButton2.Font.Color:= clBlue;
    SpStatus:= 2;
    Check2();
    str(Time[SpStatus],S);Edit1.Text:= S;
    str(Blinktime[SpStatus],S);Edit2.Text:= S;
    Image1.Picture.LoadFromFile(Filename[SpStatus]);
end;

procedure TForm1.SpeedButton3Click(Sender: TObject);
begin
    Check1();
    val(Edit1.text,Time[SpStatus],code);
    val(Edit2.text,Blinktime[SpStatus],code);
    ChangeFont(SpStatus);
    SpeedButton3.Font.Color:= clBlue;
    SpStatus:= 3;
    Check2();
    str(Time[SpStatus],S);Edit1.Text:= S;
    str(Blinktime[SpStatus],S);Edit2.Text:= S;
    Image1.Picture.LoadFromFile(Filename[SpStatus]);
end;
```

```
procedure TForm1.SpeedButton4Click(Sender: TObject);
begin
    Check1();
    val(Edit1.text,Time[SpStatus],code);
    val(Edit2.text,Blinktime[SpStatus],code);
    ChangeFont(SpStatus);
    SpeedButton4.Font.Color:= clBlue;
    SpStatus:= 4;
    Check2();
    str(Time[SpStatus],S);Edit1.Text:= S;
    str(Blinktime[SpStatus],S);Edit2.Text:= S;
    Image1.Picture.LoadFromFile(Filename[SpStatus]);
end;

procedure TForm1.SpeedButton5Click(Sender: TObject);
var code:integer;
begin
    Check1();
    val(Edit1.text,Time[SpStatus],code);
    val(Edit2.text,Blinktime[SpStatus],code);
    ChangeFont(SpStatus);
    SpeedButton5.Font.Color:= clBlue;
    SpStatus:= 5;
    Check2();
    str(Time[SpStatus],S);Edit1.Text:= S;
    str(Blinktime[SpStatus],S);Edit2.Text:= S;
    Image1.Picture.LoadFromFile(Filename[SpStatus]);
end;

procedure TForm1.SpeedButton6Click(Sender: TObject);
```

```
begin
    Check1();
    val(Edit1.text,Time[SpStatus],code);
    val(Edit2.text,Blinktime[SpStatus],code);
    ChangeFont(SpStatus);
    SpeedButton6.Font.Color:= clBlue;
    SpStatus:= 6;
    Check2();
    str(Time[SpStatus],S);Edit1.Text:= S;
    str(Blinktime[SpStatus],S);Edit2.Text:= S;
    Image1.Picture.LoadFromFile(Filename[SpStatus]);
end;

procedure TForm1.SpeedButton7Click(Sender: TObject);
begin
    Check1();
    val(Edit1.text,Time[SpStatus],code);
    val(Edit2.text,Blinktime[SpStatus],code);
    ChangeFont(SpStatus);
    SpeedButton7.Font.Color:= clBlue;
    SpStatus:= 7;
    Check2();
    str(Time[SpStatus],S);Edit1.Text:= S;
    str(Blinktime[SpStatus],S);Edit2.Text:= S;
    Image1.Picture.LoadFromFile(Filename[SpStatus]);
end;

procedure TForm1.Check1();
begin
    if RadioButton1.Checked= true then Typecheck[SpStatus]:= 1;
    if RadioButton2.Checked= true then Typecheck[SpStatus]:= 2;
```

```
    if RadioButton3.Checked= true then Typecheck[SpStatus]:= 3;
    if RadioButton4.Checked= true then Typecheck[SpStatus]:= 4;
    if RadioButton5.Checked= true then Typecheck[SpStatus]:= 5;
    if RadioButton6.Checked= true then Typecheck[SpStatus]:= 6;
    if RadioButton7.Checked= true then Typecheck[SpStatus]:= 7;
    if RadioButton8.Checked= true then Typecheck[SpStatus]:= 8;
end;

procedure TForm1.Check2();
begin
    case Typecheck[SpStatus] of
        1: RadioButton1.Checked:= true;
        2: RadioButton2.Checked:= true;
        3: RadioButton3.Checked:= true;
        4: RadioButton4.Checked:= true;
        5: RadioButton5.Checked:= true;
        6: RadioButton6.Checked:= true;
        7: RadioButton7.Checked:= true;
        8: RadioButton8.Checked:= true;
    end;
end;

function TForm1.ToHex1(Num: byte):char;
begin
    case Num of
        0: ToHex1:= '0';
        1: ToHex1:= '1';
        2: ToHex1:= '2';
        3: ToHex1:= '3';
        4: ToHex1:= '4';
        5: ToHex1:= '5';
```

```
        6: ToHex1:= '6';
        7: ToHex1:= '7';
        8: ToHex1:= '8';
        9: ToHex1:= '9';
        10: ToHex1:= 'A';
        11: ToHex1:= 'B';
        12: ToHex1:= 'C';
        13: ToHex1:= 'D';
        14: ToHex1:= 'E';
        15: ToHex1:= 'F';
    end;
end;

function TForm1.Swap(Num2: byte):byte;
begin
    case Num2 of
        0: Swap:= 0;
        1: Swap:= 8;
        2: Swap:= 4;
        3: Swap:= 12;
        4: Swap:= 2;
        5: Swap:= 10;
        6: Swap:= 6;
        7: Swap:= 14;
        8: Swap:= 1;
        9: Swap:= 9;
        10: Swap:= 5;
        11: Swap:= 13;
        12: Swap:= 3;
        13: Swap:= 11;
        14: Swap:= 7;
```

```
        15: Swap:= 15;
    end;
end;

function TForm1.First_End(Num3:byte):byte;
var  Num4,Num5: byte;
begin
    Num4:= Num3 div 16;
    Num5:= Num3 mod 16;
    First_End:= (16-((Num4+Num5)mod 16))mod 16;
end;

procedure TForm1.Button2Click(Sender: TObject);
var s: string;
    ch: char;
    i,j,k,l,m,n,o,sum: byte;
    Imagecheck: boolean;
begin
  Check1();
  val(Edit1.text,Time[SpStatus],code);
  val(Edit2.text,Blinktime[SpStatus],code);
  if SaveDialog1.Execute then
     Output_Filename:= SaveDialog1.FileName;

  n:= 19;
  for o:=1 to 7 do
   if Typecheck[o]<>8 then
   begin
     Imagecheck := true;
     assignfile(ImageFile,Filename[o]);
     reset(ImageFile,1);
```

```
blockread(ImageFile,Image_Test,SizeOf(Image_Test));
closefile(ImageFile);

s:= Image_Test.File_type[1]+Image_Test.File_type[2];
if s<>'BM' then
 begin
  s:=Filename[o]+' is not Bitmap file.';
  showmessage(s);
  Imagecheck:= false;
 end;
if (Image_Test.Width[1]<>91)or(Image_Test.Width[2]<>0) then
 begin
  s:=Filename[o]+' is not 91 pixels width.';
  showmessage(s);
  Imagecheck:= false;
 end;
if (Image_Test.Height[1]<>60)or(Image_Test.Height[2]<>0) then
 begin
  s:=Filename[o]+' is not 60 pixels height.';
  showmessage(s);
  Imagecheck:= false;
 end;
if Image_Test.Bits_per_pixel<>1 then
 begin
  s:=Filename[o]+' is not B/W color.';
  showmessage(s);
  Imagecheck:= false;
 end;
if Imagecheck=true then
 begin
  n:= n+1;
```

```
      Picture_file[n]:= Filename[o];
      Display_Type[n]:= Typecheck[o];
      Show_Time[n]:= Time[o];
      Blink_Times[n]:= Blinktime[o];
    end;
  end;

assignfile(Output_File,Output_Filename);
Rewrite(Output_File);
writeln(Output_File);

if n<>19 then
begin
 for m:=20 to n do
 begin
   assignfile(ImageFile,Picture_File[m]);
   reset(ImageFile,1);
   blockread(ImageFile,Image,SizeOf(Image));
   closefile(ImageFile);

   for i:=0 to 59 do
   begin
     Data_Value[1]:=((1024*m)+(16*i))div(16*16*16);
     Data_Value[2]:=(((1024*m)+(16*i))div 256)mod 16;
     Data_Value[3]:=(((1024*m)+(16*i))mod 256)div 16;
     Data_Value[4]:=((1024*m)+(16*i))mod 16;
     Data_Value[5]:=0;
     Data_Value[6]:=0;
     k:=7;
     for j:=12 downto 2 do
     begin
```

```
 Data_Value[k]:=Swap((Image.Data[i,j]div 32)+((Image.Data[i,j-1]mod 2)*8));
 k:= k+1;
 Data_Value[k]:=Swap((Image.Data[i,j-1]div 2)mod 16);
 k:=k+1;
end;
Data_Value[k]:= Swap((Image.Data[i,1]div 32)+8);
Data_Value[30]:= 15;
sum:=0;

for l:=1 to 15 do
 sum:= sum+Data_Value[(2*l)-1];
Data_Value[31]:=First_End(sum);
sum:=12;
for l:=1 to 15 do
 sum:= sum+Data_Value[2*l];
Data_Value[32]:= (16-(sum mod 16))mod 16;
Write(Output_File,':0C');

 for l:= 1 to 32 do
begin
 Data_Text[l]:= ToHex1(Data_Value[l]);
 Write(Output_File,Data_Text[l]);
end;
 Writeln(Output_File);
end;

Data_Value[1]:=((1024*m)+(16*60))div(16*16*16);
Data_Value[2]:=(((1024*m)+(16*60))div 256)mod 16;
Data_Value[3]:=(((1024*m)+(16*60))mod 256)div 16;
Data_Value[4]:=((1024*m)+(16*60))mod 16;
Data_Value[5]:= 0;
```

```
Data_Value[6]:= 0;
Data_Value[7]:= 0;
Data_Value[8]:= Display_Type[m];
Data_Value[9]:= Show_Time[m] div 16;
Data_Value[10]:= Show_Time[m] mod 16;
Data_Value[11]:= Blink_Times[m] div 16;
Data_Value[12]:= Blink_Times[m] mod 16;

if m=n then temp:= 10 else temp:= 0;
Data_Value[13]:= temp;
Data_Value[14]:= temp;

sum:=0;
for l:=1 to 7 do
  sum:= sum+Data_Value[(2*l)-1];
Data_Value[15]:=First_End(sum);

sum:=4;
for l:=1 to 7 do
  sum:= sum+Data_Value[2*l];
Data_Value[16]:= (16-(sum mod 16))mod 16;
Write(Output_File,':04');

for l:= 1 to 16 do
begin
  Data_Text[l]:= ToHex1(Data_Value[l]);
  Write(Output_File,Data_Text[l]);
end;
Writeln(Output_File);

end;
```

```
    Write(Output_File,':00000001FF');
    Closefile(Output_File);
  end
  else showmessage('Not have any Image match');
end;

end.
```

ภาคผนวก ค

โปรแกรมไมโครคอลโทรลเลอร์

```
ADDRESS_LOW     EQU   0F800H
ADDRESS_HIGH    EQU   0F802H
DATAPORT        EQU   0F801H
READ_WRITE      EQU   0FC00H
CONTROL_PORT    EQU   0FC01H
DELAYTIME1      EQU   5FH
DELAYTIME       EQU   60H
DESTIN_H        EQU   61H
DESTIN_L        EQU   62H
TRANFER_H       EQU   63H
TRANFER_L       EQU   64H
R8              EQU   40H
R9              EQU   41H
R10             EQU   42H
R11             EQU   43H
R12             EQU   44H
R13             EQU   45H
R14             EQU   46H
CONTROL_H       EQU   47H
BLINK_TIME      EQU   48H
TYPE            EQU   49H
SOURCE_L        EQU   4AH
SOURCE_H        EQU   4BH
TIME_60         EQU   4CH
TIME_16         EQU   4DH
TIME_96         EQU   4EH

        MOV     DPTR,#0F803H
```

```
        MOV     A,#80H
        MOVX    @DPTR,A
        MOV     DPTR,#0FC03H
        MOVX    @DPTR,A
        MOV     DPTR,#READ_WRITE
        MOV     A,#03H
        MOVX    @DPTR,A

        LCALL   DELRAM1
        LCALL   DELRAM2

        MOV     DPTR,#5000H
        MOV     R1,#00H
        MOV     R3,#19H
TOI:    MOV     A,#00H
        MOVC    A,@A+DPTR
        MOV     R0,A
        MOV     A,DPH
        ADD     A,#40H
        MOV     DPH,A
        MOV     A,R0
        MOVX    @DPTR,A
        INC     DPTR
        MOV     A,DPH
        ADD     A,#0C0H
        MOV     DPH,A
        DJNZ    R1,TOI
        DJNZ    R3,TOI
```

```
MAIN1:      MOV     SOURCE_H,#90H
MAIN2:      MOV     SOURCE_L,#00H
            MOV     A,SOURCE_H
            ADD     A,#03H
            MOV     CONTROL_H,A
            MOV     DPH,A
            MOV     DPL,#0C0H
            MOVX    A,@DPTR
            MOV     TYPE,A
            INC     DPTR
            MOVX    A,@DPTR
            MOV     DELAYTIME,A
            INC     DPTR
            MOVX    A,@DPTR
            MOV     BLINK_TIME,A
            MOV     A,TYPE
            CJNE    A,#01H,NEXT1
            LCALL   SHIFT_UP
            SJMP    END_SHOW
NEXT1:      CJNE    A,#02H,NEXT2
            LCALL   SHIFT_DOWN
            SJMP    END_SHOW
NEXT2:      CJNE    A,#03H,NEXT3
            LCALL   SHIFT_RIGHT
            SJMP    END_SHOW
NEXT3:      CJNE    A,#04H,NEXT4
            LCALL   SHIFT_LEFT
```

```
                SJMP    END_SHOW
NEXT4:          CJNE    A,#05H,NEXT5
                LCALL   BLINK
                SJMP    END_SHOW
NEXT5:          CJNE    A,#06H,NEXT6
                LCALL   NORMAL
                SJMP    END_SHOW
NEXT6:          LCALL   TURN_OFF
END_SHOW:       MOV     A,CONTROL_H
                MOV     DPH,A
                MOV     DPL,#0C3H
                MOVX    A,@DPTR
                CJNE    A,#0AAH,NOT_END
                LJMP    MAIN1
NOT_END:        INC     CONTROL_H
                MOV     SOURCE_H,CONTROL_H
                SJMP    MAIN2

;;;;;;;;;;    DISPLAY TYPE 1 'SHIFT UP PICTURE '    ;;;;;;;;;;;;

SHIFT_UP:       MOV     6FH,#02H
                MOV     6EH,DELAYTIME
                MOV     6DH,#01H
                MOV     DESTIN_H,#84H
                MOV     DESTIN_L,#00H
                LCALL   FILLPROMT
                LCALL   MPROMT
                MOV     7FH,#01H
```

```
                MOV     TRANFER_H,#87H
                MOV     TRANFER_L,#0A0H
STEPUP4:        MOV     70H,#3BH
STEPUP1:        MOV     DELAYTIME,#01H
                MOV     A,TRANFER_L
                ADD     A,#0F0H
                MOV     TRANFER_L,A
                MOV     A,70H
                JB      ACC.0,STEPU1
                JNB     ACC.0,STEPU2
STEPUP2:        MOV     A,TRANFER_L
                JNZ     STEPUP3
                DEC     TRANFER_H
STEPUP3:        DJNZ    70H,STEPUP1
                MOV     DELAYTIME,6EH
                LCALL   DELAY
                MOV     DELAYTIME,6EH
                LCALL   DELAY
                DJNZ    6FH,STEPUP4
                RET

STEPU1:         LCALL   MTO_RAM1
                LJMP    STEPUP2
STEPU2:         LCALL   MTO_RAM2
                LJMP    STEPUP2

;;;;;;;;      DISPLAY  TYPE  2  SHIFT DOWN PICTURE     ;;;;;;;;;;;;;;;
```

```
SHIFT_DOWN:   MOV     6FH,#02H
              MOV     6EH,DELAYTIME
              MOV     6DH,#01H
              MOV     DESTIN_H,#84H
              MOV     DESTIN_L,#00H
              LCALL   FILLPROMT
              LCALL   MPROMT
              MOV     TRANFER_H,#80H
              MOV     TRANFER_L,#50H
STEPDOWN4:    MOV     70H,#3CH
STEPDOWN1:    MOV     DELAYTIME,#01H
              MOV     A,70H
              JB      ACC.0,STEPD1
              JNB     ACC.0,STEPD2
STEPDOWN2:    MOV     A,TRANFER_L
              ADD     A,#10H
              MOV     TRANFER_L,A
              JNZ     STEPDOWN3
              INC     TRANFER_H
STEPDOWN3:    DJNZ    70H,STEPDOWN1
              MOV     DELAYTIME,6EH
              LCALL   DELAY
              MOV     DELAYTIME,6EH
              LCALL   DELAY
              DJNZ    6FH,STEPDOWN4
              RET

STEPD1:       LCALL   MTO_RAM1
```

```
                LJMP    STEPDOWN2
STEPD2:         LCALL   MTO_RAM2
                LJMP    STEPDOWN2

;;;;;;;;;;' DISPLAY TYPE 3 'SHIFT RIGHT'  ;;;;;;;;;;;;;

SHIFT_RIGHT:    LCALL   FILLPROMT
                MOV     TRANFER_H,#84H
                MOV     TRANFER_L,#00H
                MOV     DESTIN_H,#80H
                MOV     DESTIN_L,#00H
                LCALL   MPROMT
                MOV     TIME_96,#61H
                MOV     DELAYTIME1,DELAYTIME
                MOV     DELAYTIME,#01H
                MOV     34H,#0BBH
                MOV     35H,#0FFH
BEGIN6:         MOV     TIME_60,#3CH
                MOV     R8,#7FH
                MOV     R9,#10H
                MOV     R10,#83H
                MOV     R11,#10H
ADDR6:          MOV     TIME_16,#10H
                INC     R8
                INC     R10
ADDR5:          DEC     R9
                DEC     R9
                DEC     R9
```

```
            DEC     R9
            DEC     R11
            DEC     R11
            DEC     R11
            DEC     R11
            SETB    C
            MOV     30H,#0CH
IMAGE3:     DEC     R9
            MOV     DPH,R8
            MOV     DPL,R9
            MOVX    A,@DPTR
            RLC     A
            MOVX    @DPTR,A
            DJNZ    30H,IMAGE3
            MOV     30H,#0CH
IMAGE4:     DEC     R11
            MOV     DPH,R10
            MOV     DPL,R11
            MOVX    A,@DPTR
            RLC     A
            MOVX    @DPTR,A
            DJNZ    30H,IMAGE4
            MOV     A,R9
            ADD     A,#20H
            MOV     R9,A
            MOV     R11,A
            DJNZ    TIME_60,ADDR7
            SJMP    ADDR8
```

```
ADDR7:          DJNZ        TIME_16,ADDR5
                SJMP        ADDR6
ADDR8:          DJNZ        TIME_96,CHECK_187
                MOV         DELAYTIME,DELAYTIME1
                MOV         R0,#09H
                MOV         A,DELAYTIME
MULTI1:         ADD         A,DELAYTIME
                DJNZ        R0,MULTI1
                MOV         DELAYTIME,A
                LCALL       DISPLAY2
                MOV         DELAYTIME,#01H
CHECK_187:      MOV         A,35H
                CPL         A
                MOV         35H,A
                CJNE        A,#0FFH,SHOW_RAM11
                LCALL       MTO_RAM2
                SJMP        NOT_SHOW1
SHOW_RAM11:     LCALL       MTO_RAM1
NOT_SHOW1:      DJNZ        34H,BEGIN7
                RET
BEGIN7:         LJMP        BEGIN6

;;;;;;;;;;  END DISPLAY TYPE 3 'SHIFT RIGHT'  ;;;;;;;

;;;;;;;;;;;  DISPLAY TYPE 4 'SHIFT LEFT'  ;;;;;;;;;;

SHIFT_LEFT:     LCALL       FILLPROMT
                MOV         TRANFER_H,#84H
```

```
                MOV     TRANFER_L,#00H
                MOV     DESTIN_H,#80H
                MOV     DESTIN_L,#00H
                LCALL   MPROMT
                MOV     TIME_96,#61H
                MOV     DELAYTIME1,DELAYTIME
                MOV     DELAYTIME,#01H
                MOV     34H,#0BBH
                MOV     35H,#0FFH
BEGIN5:         MOV     TIME_60,#3CH
                MOV     R8,#7FH
                MOV     R9,#00H
                MOV     R10,#83H
                MOV     R11,#00H
ADDR1:          MOV     TIME_16,#10H
                INC     R8
                INC     R10
ADDR2:          SETB    C
                MOV     30H,#0CH
IMAGE1:         MOV     DPH,R8
                MOV     DPL,R9
                MOVX    A,@DPTR
                RRC     A
                MOVX    @DPTR,A
                INC     R9
                DJNZ    30H,IMAGE1
                MOV     30H,#0CH
IMAGE2:         MOV     DPH,R10
```

```
                MOV     DPL,R11
                MOVX    A,@DPTR
                RRC     A
                MOVX    @DPTR,A
                INC     R11
                DJNZ    30H,IMAGE2
                INC     R9
                INC     R9
                INC     R9
                INC     R9
                INC     R11
                INC     R11
                INC     R11
                INC     R11
                DJNZ    TIME_60,ADDR0
                SJMP    ADDR4
ADDR0:          DJNZ    TIME_16,ADDR2
                SJMP    ADDR1
ADDR4:          DJNZ    TIME_96,CHECK187
                MOV     DELAYTIME,DELAYTIME1
                MOV     R0,#09H
                MOV     A,DELAYTIME
MULTI2:         ADD     A,DELAYTIME
                DJNZ    R0,MULTI2
                MOV     DELAYTIME,A
                LCALL   DISPLAY2
                MOV     DELAYTIME,#01H
CHECK187:       MOV     A,35H
```

```
                CPL     A
                MOV     35H,A
                CJNE    A,#0FFH,SHOW_RAM1
                LCALL   MTO_RAM2
                SJMP    NOT_SHOW
SHOW_RAM1:      LCALL   MTO_RAM1
NOT_SHOW:       DJNZ    34H,BEGIN5
                RET

;;;;;;;;;;  END DISPLAY TYPE 4 'SHIFT LEFT'  ;;;;;;;;

;;;;;;;;;;;;;  DISPLAY TYPE 5 'BLINK'  ;;;;;;;;;;;;;;

BLINK:          LCALL   DELRAM1
                MOV     R0,#09H
                MOV     A,DELAYTIME
MULTI3:         ADD     A,DELAYTIME
                DJNZ    R0,MULTI3
                LCALL   MTO_RAM2
                DEC     BLINK_TIME
                MOV     TRANFER_H,SOURCE_H
                MOV     TRANFER_L,SOURCE_L
LOOP1:          LCALL   DISPLAY1
                LCALL   DISPLAY2
                DJNZ    BLINK_TIME,LOOP1
                RET
;;;;;;;;;;  END DISPLAY TYPE 5 'BLINK'  ;;;;;;;;;;
```

```
;;;;;;;;;;;;;  DISPLAY TYPE 6 'NORMAL'  ;;;;;;;;;;;;

NORMAL:         MOV     TRANFER_H,SOURCE_H
                MOV     TRANFER_L,SOURCE_L
                MOV     R0,#09H
                MOV     A,DELAYTIME
MULTI4:         ADD     A,DELAYTIME
                DJNZ    R0,MULTI4
                LCALL   MTO_RAM1
                LCALL   DISPLAY1
                RET
;;;;;;;;;;;  END DISPLAY TYPE 6 'NORMAL'  ;;;;;;;;;;

;;;;;;;;;;;;;  DISPLAY TYPE 7 'DARK'  ;;;;;;;;;;;;;;

TURN_OFF:       LCALL   DELRAM1
                MOV     R0,#09H
                MOV     A,DELAYTIME
MULTI5:         ADD     A,DELAYTIME
                DJNZ    R0,MULTI5
                LCALL   DISPLAY1
                RET

;;;;;;;;;;;  END DISPLAY TYPE 7 'DARK'  ;;;;;;;;;;;;

        ;;;;;;;;    MOV DATA FROM PLACE TO DESTINATE  ;;;;;;;;
```

```
MPROMT:     MOV     R0,SOURCE_H
            MOV     R1,SOURCE_L
            MOV     R2,DESTIN_H
            MOV     R3,DESTIN_L
            MOV     72H,#04H
PROMT:      MOV     DPH,R0
            MOV     DPL,R1
            MOVX    A,@DPTR
            MOV     DPH,R2
            MOV     DPL,R3
            MOVX    @DPTR,A
            DJNZ    R3,PROMT1
            INC     R2
PROMT1:     DJNZ    R1,PROMT
            INC     R0
            DJNZ    72H,PROMT
            MOV     DPTR,#87BFH
            MOV     A,#0FFH
PROMT2:     MOVX    @DPTR,A
            INC     DPTR
            DJNZ    R4,PROMT2
            RET

;;;;;;;;  SUBROUTINE SIHFT UP $ DOWN ;;;;;;;

FILLPROMT:  MOV     R0,#80H
            MOV     R1,#00H
            MOV     R3,#0EH
```

```
                MOV     A,#0FFH
LEONAT:         MOV     DPH,R0
                MOV     DPL,R1
                MOVX    @DPTR,A
                DJNZ    R1,LEONAT
                INC     R0
                DJNZ    R3,LEONAT
                RET

;;;;;;;;   MOVE MRAM1 TO 0RAM1   ;;;;;;;;

MTO_RAM1:       MOV     R0,TRANFER_H
                MOV     R1,TRANFER_L
                MOV     R4,#04H
                MOV     R2,#00H
                MOV     R3,#00H
                MOV     DPTR,#CONTROL_PORT
                MOV     A,#05H
                MOVX    @DPTR,A
MTO1:           MOV     DPH,R0
                MOV     DPL,R1
                MOVX    A,@DPTR
                MOV     DPTR,#DATAPORT
                MOVX    @DPTR,A
                MOV     DPTR,#ADDRESS_HIGH
                MOV     A,R2
                MOVX    @DPTR,A
                MOV     DPTR,#ADDRESS_LOW
```

```
                MOV     A,R3
                MOVX    @DPTR,A
                LCALL   WRITE1
                INC     R1
                INC     R1
                DJNZ    R1,MTO2
                INC     R0
MTO2:           INC     R3
                INC     R3
                DJNZ    R3,MTO1
                INC     R2
                DJNZ    R4,MTO1
                LCALL   DISPLAY1
                RET

;;;;;;;;  MOV MRAM2 TO ORAM2  ;;;;;;;;

MTO_RAM2:       MOV     R0,TRANFER_H
                MOV     R1,TRANFER_L
                MOV     R4,#04H
                MOV     R2,#00H
                MOV     R3,#00H
                MOV     DPTR,#CONTROL_PORT
                MOV     A,#0AH
                MOVX    @DPTR,A
MTO3:           MOV     DPH,R0
                MOV     DPL,R1
                MOVX    A,@DPTR
```

```
                MOV     DPTR,#DATAPORT
                MOVX    @DPTR,A
                MOV     DPTR,#ADDRESS_HIGH
                MOV     A,R2
                MOVX    @DPTR,A
                MOV     DPTR,#ADDRESS_LOW
                MOV     A,R3
                MOVX    @DPTR,A
                LCALL   WRITE2
                INC     R1
                INC     R1
                DJNZ    R1,MTO4
                INC     R0
MTO4:           INC     R3
                INC     R3
                DJNZ    R3,MTO3
                INC     R2
                DJNZ    R4,MTO3
                LCALL   DISPLAY2
                RET

;;;;;;;;     WRITE RAM     ;;;;;;;;

WRITE1:         MOV     DPTR,#READ_WRITE
                MOV     A,#02H
                MOVX    @DPTR,A
                MOV     A,#03H
                MOVX    @DPTR,A
```

```
                RET

WRITE2:         MOV     DPTR,#READ_WRITE
                MOV     A,#01H
                MOVX    @DPTR,A
                MOV     A,#03H
                MOVX    @DPTR,A
                RET

;;;;;;;;    DEL RAM1    ;;;;;;;;;;;

DELRAM1:        MOV     DPTR,#CONTROL_PORT
                MOV     A,#05H
                MOVX    @DPTR,A
                MOV     DPTR,#DATAPORT
                MOV     A,#0FFH
                MOVX    @DPTR,A
                MOV     R0,#04H
                MOV     R1,#00H
FILL1:          MOV     DPTR,#ADDRESS_HIGH
                MOV     A,R0
                MOVX    @DPTR,A
FILL2:          MOV     DPTR,#ADDRESS_LOW
                MOV     A,R1
                MOVX    @DPTR,A
                CALL    WRITE1
                DJNZ    R1,FILL2
                DJNZ    R0,FILL1
```

```
                RET

;;;;;;;;    DEL RAM2    ;;;;;;;;;;

DELRAM2:    MOV     DPTR,#CONTROL_PORT
            MOV     A,#0AH
            MOVX    @DPTR,A
            MOV     DPTR,#DATAPORT
            MOV     A,#0FFH
            MOVX    @DPTR,A
            MOV     R0,#04H
            MOV     R1,#00H
FILL3:      MOV     DPTR,#ADDRESS_HIGH
            MOV     A,R0
            MOVX    @DPTR,A
FILL4:      MOV     DPTR,#ADDRESS_LOW
            MOV     A,R1
            MOVX    @DPTR,A
            CALL    WRITE2
            DJNZ    R1,FILL4
            DJNZ    R0,FILL3
            RET

;;;;;;;;    DELAY TIME    ;;;;;;;;

DELAY:      MOV     R2,DELAYTIME
            MOV     R3,DELAYTIME
```

```
                MOV     R4,DELAYTIME
                MOV     R5,DELAYTIME
DELAY1:         DJNZ    R3,DELAY1
DELAY2:         DJNZ    R4,DELAY1
                DJNZ    R5,DELAY2
                RET

;;;;;;;;    DISPLAY1    ;;;;;;;;

DISPLAY1:       MOV     DPTR,#CONTROL_PORT
                MOV     A,#0AH
                MOVX    @DPTR,A
                MOV     R2,DELAYTIME
                CALL    DELAY
                RET

;;;;;;;;    DISPLAY2    ;;;;;;;;

DISPLAY2:       MOV     DPTR,#CONTROL_PORT
                MOV     A,#05H
                MOVX    @DPTR,A
                MOV     R2,DELAYTIME
                CALL    DELAY
                RET

                END
```

กิตติกรรมประกาศ

ผู้จัดทำขอขอบคุณ อ.ชินภัทร นันทจิวากรชัย ที่ได้ให้คำแนะนำและชี้แนะแนวทางต่างๆ ในการ แก้ไขปัญหาจนโครงงานสำเร็จลุล่วงไปได้ด้วยดี และขอขอบคุณเพื่อน ๆ ที่คอยช่วยเหลือ และเอื้อเฟื้ออุปกรณ์ต่าง ๆ แก่ผู้จัดทำตลอดมา

วิไล แม้นถาวรสิริ
.....
(นางสาววิไล แม้นถาวรสิริ)

ศักดิ์ชัย เลปนะวัฒน์
.....
(นายศักดิ์ชัย เลปนะวัฒน์)

สรณชนม์ นิชพรกุล
.....
(นายสรณชนม์ นิชพรกุล)

ผู้จัดทำ

หนังสืออ้างอิง

1. Motorola , Inc. , " High-Speed CMOSS Data " , 1996
2. National Semiconductor, " National Application Specific Analog Products",1995
3. Shoichi Matsumoto , " Electronic Display Device " , John Wiley & Sons 273 p, 1993
4. Texas Instruments Inc. , " The TTL Data Book " , 1992