ระบบวีดีโอที่ซ่อนภาพบรรยายได้

นาย นันทกฤษณ์ วัฒนเลี้ยงใจ

วิทยานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิศวกรรมศาสตรมหาบัณฑิต
สาขาวิชาวิศวกรรมไฟฟ้า ภาควิชาวิศวกรรมไฟฟ้า
คณะวิศวกรรมศาสตร์ จุฬาลงกรณ์มหาวิทยาลัย
ปีการศึกษา 2543

ISBN 974-346-392-5

A CLOSED PICTURE CAPTION VIDEO SYSTEM

Mr. Nuntakrit Vattanaliangjai

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 2000

ISBN 974-346-392-5

หัวข้อวิทยานิพนธ์	ระบบวีดีโอที่ซ่อนภาพบรรยายได้
โดย	นายนันทกฤษณ์ วัฒนเลี้ยงใจ
ภาควิชา	วิศวกรรมไฟฟ้า
อาจารย์ที่ปรึกษา	รองศาสตราจารย์ ดร. เอกชัย ลีลารัศมี

คณะวิศวกรรมศาสตร์ จุฬาลงกรณ์มหาวิทยาลัย อนุมัติให้นับวิทยานิพนธ์ฉบับนี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญามหาบัณฑิต

..... คณบดีคณะวิศวกรรมศาสตร์
(ศาสตราจารย์ ดร. สมศักดิ์ ปัญญาแก้ว)

คณะกรรมการสอบวิทยานิพนธ์

..... ประธานกรรมการ
(ศาสตราจารย์ ดร. ประสิทธิ์ ประพิณมงคลการ)

..... อาจารย์ที่ปรึกษา
(รองศาสตราจารย์ ดร. เอกชัย ลีลารัศมี)

.....กรรมการ
(อ. สุวิทย์ นาคพีระยุทธ)

นันทกฤษณ์ วัฒนเลี้ยงใจ : ระบบวีดีโอที่ซ่อนภาพบรรยายได้. (A Closed Picture Caption Video System) อ. ที่ปรึกษา : รศ.ดร.เอกชัย ลีลารัศมี, 158 หน้า. ISBN 974-346-392-5.

วิทยานิพนธ์นี้นำเสนอระบบที่ใช้ช่วงไร้ภาพของสัญญาณวีดิทัศน์ระบบ PAL เพื่อแทรกภาพเคลื่อนไหวซึ่งเป็นภาพขาวดำขนาด 112X128 จุด การบีบอัดข้อมูลจะใช้วิธีตามมาตรฐาน CCITT T.4 2-D ซึ่งสามารถลดข้อมูลของแต่ละภาพได้ 5-10 เท่าโดยไม่สูญเสียข้อมูล นอกจากนี้ขั้นตอนวิธีคลายข้อมูลยังสามารถทำในเวลาจริงได้โดยใช้ไมโครคอนโทรลเลอร์ความเร็วสูงขนาด 8 บิต เครื่องแทรกข้อมูลภาพและเครื่องรับข้อมูลภาพได้ถูกพัฒนาขึ้นเพื่อทดสอบระบบนี้ เครื่องแทรกข้อมูลภาพถูกออกแบบให้แทรกข้อมูลจำนวน 48 ไบต์ที่บีบอัดด้วยเครื่องคอมพิวเตอร์ส่วนบุคคล ข้อมูลจะถูกแทรกในช่วงเส้นที่ 7-22 ของสัญญาณวีดิทัศน์ระบบ PAL เครื่องรับข้อมูลภาพจะแยกข้อมูลเหล่านี้มาคลาย และแสดงภาพที่มุมขวาด้านล่างของจอโทรทัศน์ การวิเคราะห์เบื้องต้นพบว่าระบบนี้สามารถแสดงภาพเคลื่อนไหวได้ในอัตรา 6 ภาพต่อวินาที ดังนั้นระบบนี้จึงมีศักยภาพดีพอสำหรับการบริการภาพบรรยายภาษามือแบบซ่อนได้

ภาควิชา วิศวกรรมไฟฟ้า ลายมือชื่อนิสิต .....

สาขาวิชา วิศวกรรมไฟฟ้า ลายมือชื่ออาจารย์ที่ปรึกษา .....

ปีการศึกษา 2543 ลายมือชื่ออาจารย์ที่ปรึกษาร่วม .....

4170367021 : MAJOR ELECTRICAL ENGINEERING

KEY WORD: PAL VIDEO SYSTEM / VIDEO SIGNAL / COMPRESSION / CCITT T.4 2-D

NUNTAKRIT VATTANALIANGJAI: A CLOSED PICTURE CAPTION VIDEO SYSTEM. THESIS ADVISOR : ASSO. PROF. EKACHAI LEELARASMEE, Ph.D., 158 pp. ISBN 974-346-392-5.

This dissertation proposes a system that uses the vertical blanking intervals of a PAL video signal for transmitting and receiving a sequence of animation pictures, each of which consists of 112 X 128 dots of either black or white. A data compression technique based on the CCITT T.4 2-D standard is adopted by the system to reduce the amount of transmitting data per picture by an average of 5-10 times with no loss of information. Yet, its decompression algorithm is still simple enough to be implemented at the receiver in real time by a high speed 8 bit microcontroller. Two pieces of hardware, i.e. an inserter and a decoder, are developed to test the proposed system. The inserter is designed to hide 48 bytes of compressed data, generated by a personal computer, in lines 7 to 22 of a PAL video signal. The decoder can extract these data, decompress and display the hidden picture on the lower right position of a normal TV screen. Preliminary analysis has indicated that this system can support an animation rate of 6 hidden picture frames per second. Therefore, it has a good potential for providing a sign language closed captioning service.

Department Electrical Engineering . Student's signature .....

Field of study Electrical Engineering . Advisor's signature .....

Academic Year 2000 Co-advisor's signature .....

กิตติกรรมประกาศ

วิทยานิพนธ์ฉบับนี้สำเร็จลุล่วงด้วยความช่วยเหลืออย่างดียิ่งของ รศ.ดร. เอกชัย ลีลารัศมี อาจารย์ที่ปรึกษาวิทยานิพนธ์ ซึ่งได้ให้คำแนะนำและข้อคิดเห็นต่างๆ พร้อมทั้งจัดหาอุปกรณ์ที่ใช้ในการทำวิจัย จึงใคร่ขอกราบขอบพระคุณมา ณ โอกาสนี้ด้วย

ข้าพเจ้าขอขอบคุณห้องปฏิบัติการวิจัยระบบเชิงเลข ซึ่งเป็นสถานที่ทำการวิจัย และเพื่อนพี่น้องนิสิตห้องปฏิบัติการวิจัยระบบเชิงเลขทุกท่าน ที่ให้คำแนะนำ แสดงข้อคิดเห็น และให้กำลังใจตลอดการทำงานวิจัย

ท้ายนี้ข้าพเจ้าขอขอบคุณบิดา มารดา และครอบครัวของข้าพเจ้าที่ให้โอกาสในการศึกษาแก่ข้าพเจ้า ตลอดจนให้กำลังใจและเอาใจใส่ข้าพเจ้าเสมอมา จนสำเร็จการศึกษา

สารบัญ

หน้า

สารบัญ (ต่อ)

หน้า

สารบัญ (ต่อ)

หน้า

สารบัญภาพ

หน้า

สารบัญภาพ (ต่อ)

หน้า

สารบัญภาพ (ต่อ)

หน้า

สารบัญตาราง

หน้า

บทที่ 1

บทนำ

1.1 แนวเหตุผลในการทำวิทยานิพนธ์

ในปัจจุบันมีสื่อมากมายที่ให้บริการข้อมูลข่าวสาร โทรทัศน์เป็นอีกสื่อหนึ่งที่มีการใช้อย่างแพร่หลายในคนหลายๆ กลุ่ม ผู้พิการทางหูเป็นคนกลุ่มหนึ่งที่รับชมสื่อทางด้านนี้มากเช่นกัน ดังนั้น การรับชมรายการต่างๆจึงต้องมีระบบที่อำนวยความสะดวกแก่ผู้พิการทางหูเพื่อให้ได้ข้อมูลต่างๆ ครบถ้วน ตัวอย่างของระบบดังกล่าวคือ ระบบคำบรรยายภาพแบบซ่อนได้ (Closed Caption) ซึ่งเป็นระบบที่แสดงคำบรรยายเป็นตัวอักษรที่สอดคล้องกับเสียงที่เกิดขึ้น ระบบดังกล่าวมีการใช้อย่างแพร่หลายในต่างประเทศเพื่อช่วยผู้พิการทางหูให้ได้ข่าวสารในการรับชมรายการทางโทรทัศน์อย่างครบถ้วน

ภาษาที่ผู้พิการทางหูใช้ในการสื่อสารคือ ภาษามือ (Sign Language) ดังนั้น การใช้ภาพภาษามือบรรยายแทนเสียงจะทำให้ผู้พิการทางหูมีความเข้าใจเช่นเดียวกับคำบรรยายภาพ วิทยานิพนธ์นี้จึงเสนอแนวทางพัฒนาระบบวีดิทัศน์ที่ซ่อนภาพบรรยายขึ้น ระบบดังกล่าวมีการพัฒนามาจากระบบคำบรรยายภาพแบบซ่อนได้ที่ใช้กับสัญญาณวีดิทัศน์ระบบ PAL กล่าวคือ ระบบคำบรรยายภาพแบบซ่อนได้จะแทรกข้อมูลลงในช่วงไร้ภาพแนวตั้ง (Vertical Blanking Interval) จำนวน 1 เส้นต่อ 1 เฟรม แต่ระบบวีดิทัศน์ที่ซ่อนภาพบรรยายจะใช้จำนวนเส้นสัญญาณภาพในช่วงไร้ภาพแนวตั้งสำหรับการแทรกข้อมูลมากกว่าเนื่องจากมีข้อมูลภาพบรรยายมีจำนวนมาก ระบบนี้จะแสดงภาพบรรยายเป็นภาพเคลื่อนไหวของภาษามือ ภาพเคลื่อนไหวดังกล่าวเกิดจากการนำภาพภาษามือที่มีอิริยาบถต่างๆกันมาแสดงต่อเนื่องหลายๆ ภาพภายในเวลา 1 วินาที

ระบบวีดิทัศน์ที่ซ่อนภาพบรรยายที่นำเสนอในวิทยานิพนธ์นี้จะแสดงภาพขาว-ดำที่มุมขวาล่างของจอโทรทัศน์ คล้ายกับรายการโทรทัศน์บางรายการที่มีการแสดงภาษามือ แต่ผู้ชมสามารถเปิดปิดภาพบรรยายดังกล่าวได้ตามต้องการ ระบบนี้จึงเป็นต้นแบบสำหรับการแทรกภาพบรรยายแบบซ่อนได้ในประเทศไทยอีกด้วย

1.2 วัตถุประสงค์

1. เพื่อพัฒนากรรมวิธีการแทรกสัญญาณภาพแบบจุดขาวดำในช่วงไร้ภาพแนวตั้ง โดยการบีบอัดข้อมูลตามมาตราฐาน CCITT T.4 แบบ 2 มิติ
2. เพื่อพัฒนาต้นแบบของเครื่องแทรกสัญญาณภาพและเครื่องรับสำหรับถอดสัญญาณจากเครื่องแทรกดังกล่าว

1.3 ขอบเขตของงานวิจัย

1. พัฒนาระบบแทรกภาพแบบซ่อนได้ในสัญญาณวีดิทัศน์แบบ PAL ที่มีสมบัติดังนี้
   1.1 แทรกข้อมูลภาพในช่วงไร้ภาพแนวตั้งของสัญญาณวีดิทัศน์เส้นที่ 7 - 22 ด้วยอัตราการแทรก 500 kbit/sec
   1.2 แทรกภาพขาวดำขนาด 112 X 128 จุด
   1.3 ใช้วิธีบีบอัดข้อมูลตามมาตราฐาน CCITT T.4 แบบ 2 มิติ ในการบีบอัดข้อมูลที่ใช้รับส่ง
2. สร้างต้นแบบอุปกรณ์ที่ใช้กับระบบตามข้อ 1 ซึ่งประกอบด้วย
   2.1 อุปกรณ์เครื่องแทรกสัญญาณภาพแบบซ่อนได้
   2.2 อุปกรณ์เครื่องรับสัญญาณภาพแบบซ่อนได้
   2.3 โปรแกรมคอมพิวเตอร์ส่วนบุคคลสำหรับบีบอัดข้อมูลและส่งมาให้เครื่องแทรกเพื่อแทรกเข้าในสัญญาณวีดิทัศน์

1.4 ขั้นตอนการดำเนินงานวิจัย

1. ออกแบบและปรับปรุงเครื่องแทรกข้อมูลภาพทีละส่วนจนครบ
2. สร้างต้นแบบอุปกรณ์ของเครื่องแทรกข้อมูลภาพ
3. ออกแบบและปรับปรุงเครื่องรับทีละส่วนจนครบ
4. สร้างต้นแบบอุปกรณ์เครื่องรับ
5. พัฒนาโปรแกรมบีบอัดข้อมูลตามมาตราฐาน CCITT T.4 แบบ 2 มิติบนเครื่องคอมพิวเตอร์ส่วนบุคคลสำหรับส่งข้อมูลให้เครื่องส่ง

6. ทดสอบและปรับปรุงอุปกรณ์ต้นแบบ
7. สรุปผลการทดสอบและเขียนวิทยานิพนธ์

1.5 ประโยชน์ที่คาดว่าจะได้รับ

1. เป็นจุดเริ่มต้นในการพัฒนาระบบวีดิทัศน์ที่ซ่อนภาพบรรยายได้ในประเทศไทย
2. เพื่อประโยชน์ในการแทรกภาษามือสำหรับผู้พิการทางหู

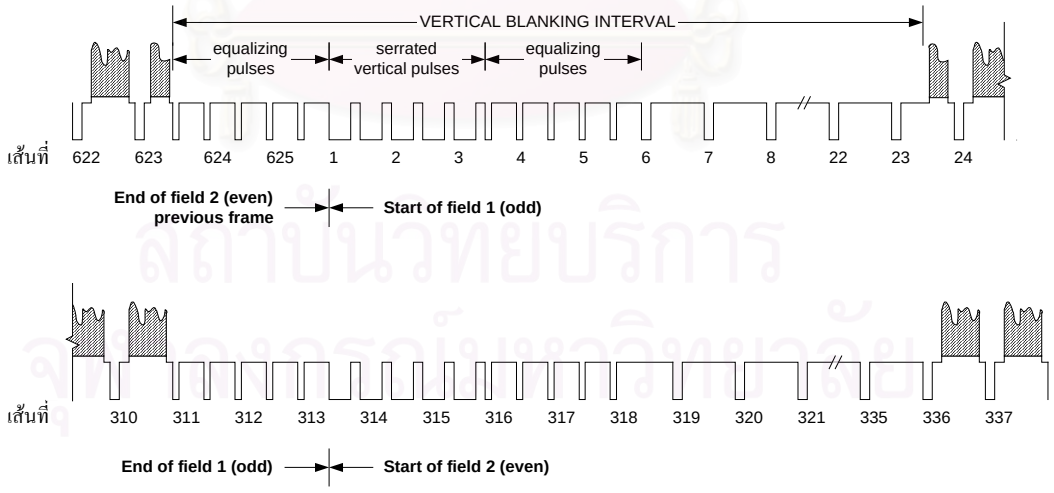
บทที่ 2

สัญญาณวีดิทัศน์ระบบ PAL และการบีบอัดข้อมูล

ระบบวีดิทัศน์ที่ซ่อนภาพบรรยายสำหรับสัญญาณวีดิทัศน์ระบบ PAL ได้มีการพัฒนามาจากระบบคำบรรยายภาพแบบซ่อนได้ โดยแทรกข้อมูลภาพลงในสัญญาณวีดิทัศน์ในช่วงไร้ภาพแนวตั้งเหมือนกัน แต่ใช้จำนวนเส้นสัญญาณวีดิทัศน์มากกว่าเนื่องจากข้อมูลมีจำนวนมาก ดังนั้นข้อมูลที่แทรกลงในช่วงไร้ภาพแนวตั้งจึงต้องมีการบีบอัดข้อมูลเพื่อลดขนาดข้อมูลที่จะส่งไปยังเครื่องรับ บทนี้จะกล่าวถึงรายละเอียดของสัญญาณวีดิทัศน์ระบบ PAL และการบีบอัดข้อมูลแบบ CCITT T.4 แบบ 2 มิติ ดังต่อไปนี้

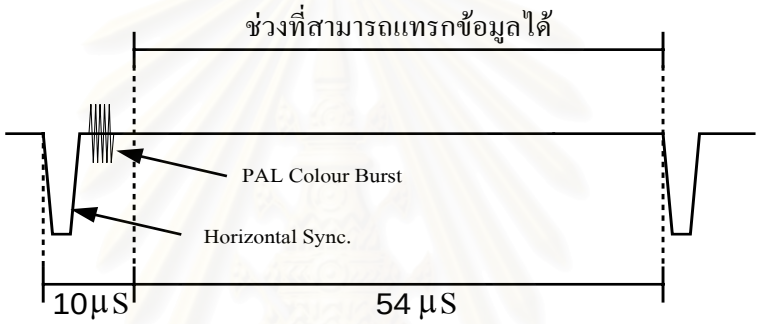
2.1 สัญญาณวีดิทัศน์ระบบ PAL

ภาพที่แสดงบนจอโทรทัศน์ของสัญญาณวีดิทัศน์ระบบ PAL เกิดจากการแสดงเฟรมภาพทั้งหมด 25 เฟรมในเวลา 1 วินาที ทำให้ตาผู้ชมไม่สามารถแยกภาพที่แสดงได้ทัน จึงเห็นเป็นภาพดังกล่าวเป็นภาพเคลื่อนไหว ลักษณะของเฟรมภาพ 1 เฟรมแสดงดังรูปที่ 2.1



รูปที่ 2.1 ส่วนประกอบของเฟรมภาพ 1 เฟรม

เฟรมภาพ 1 เฟรมประกอบด้วยฟิลด์ภาพจำนวน 2 ฟิลด์คือ ฟิลด์คู่และฟิลด์คี่ ทั้ง 2 ฟิลด์จะสลับกันแสดงบนจอภาพ โดย 1 ฟิลด์จะประกอบด้วยจำนวนเส้นสัญญาณวีดิทัศน์ 312.5 เส้น ดังนั้นเฟรมภาพ 1 เฟรมจะประกอบด้วยเส้นสัญญาณวีดิทัศน์จำนวน 625 เส้น ช่วงต้นและช่วงท้ายของแต่ละฟิลด์จะเรียกว่า ช่วงไร้ภาพแนวตั้ง ดังแสดงในรูปที่ 2.1 ซึ่งสามารถนำไปใช้แทรกข้อมูลของระบบต่างๆ เช่น ระบบ Teletext [1] และ ระบบคำบรรยายภาพแบบซ่อนได้ [1],[2] เป็นต้น การแทรกข้อมูลสามารถกระทำได้ตั้งแต่เส้นวีดิทัศน์ที่ 6 ถึง 22 ของแต่ละฟิลด์โดยไม่มีผลต่อการแสดงภาพรายการปกติบนจอโทรทัศน์ ลักษณะของเส้นสัญญาณวีดิทัศน์ระหว่างเส้นที่ 6 ถึง 22 เป็นดังรูปที่ 2.2



รูปที่ 2.2 ลักษณะเส้นสัญญาณวีดิทัศน์เส้นที่ 6 ถึง 22

มาตรฐานของ ETSI [3]กำหนดให้สัญญาณวีดิทัศน์รวม (Composite Video Signal) 1 เส้นใช้เวลาทั้งหมด 64 μS โดยในช่วง 10 μS แรกจะประกอบด้วยซิงก์แนวราบ และเบิร์ตสี ส่วน 54 μS ที่เหลือเป็นที่ว่างที่ไม่มีสัญญาณภาพ (สัญญาณอยู่ที่ระดับสีดำ) ดังนั้นข้อมูลต่างๆ จะถูกแทรกลงในช่วงเวลา 54 μS นี้

2.2 การบีบอัดข้อมูล

การบีบอัดข้อมูลเป็นการลดขนาดของข้อมูลให้เล็กลงเพื่อให้ใช้ที่เก็บน้อยลงหรือลดเวลาส่งข้อมูล การบีบอัดข้อมูลแบ่งออกเป็น 2 ประเภทคือ

2.2.1. การบีบอัดข้อมูลที่ยอมเสียข้อมูลบางส่วน (Lossy compression)

การบีบอัดข้อมูลแบบนี้จะตัดข้อมูลบางส่วนออกเพื่อให้สามารถบีบอัดข้อมูลได้มากที่สุด และเมื่อนำข้อมูลที่บีบอัดมาคลายกลับเป็นข้อมูลแล้วมีความผิดพลาดจากเดิมในเกณฑ์ที่ยอมรับได้ ตัวอย่างการบีบอัดข้อมูลชนิดนี้ที่นิยมใช้กันอย่างแพร่หลายในปัจจุบัน ได้แก่ MPEG, JPEG และ JBIG เป็นต้น

2.2.2. การบีบอัดข้อมูลที่ไม่สูญเสียข้อมูล (Lossless compression)

การบีบอัดข้อมูลแบบนี้จะรักษาความถูกต้องข้อมูลทั้งหมดและในกระบวนการบีบอัดข้อมูลจะไม่มีการตัดข้อมูลออก จึงทำให้บีบอัดข้อมูลได้น้อยกว่าแบบแรก ตัวอย่างของการบีบอัดข้อมูลแบบนี้ ได้แก่ การเข้ารหัสแบบ Run-lengths และ การบีบอัดข้อมูลแบบ Facsimile เป็นต้น

2.3 การบีบอัดข้อมูลตามมาตรฐาน CCITT T.4 แบบ 2 มิติ

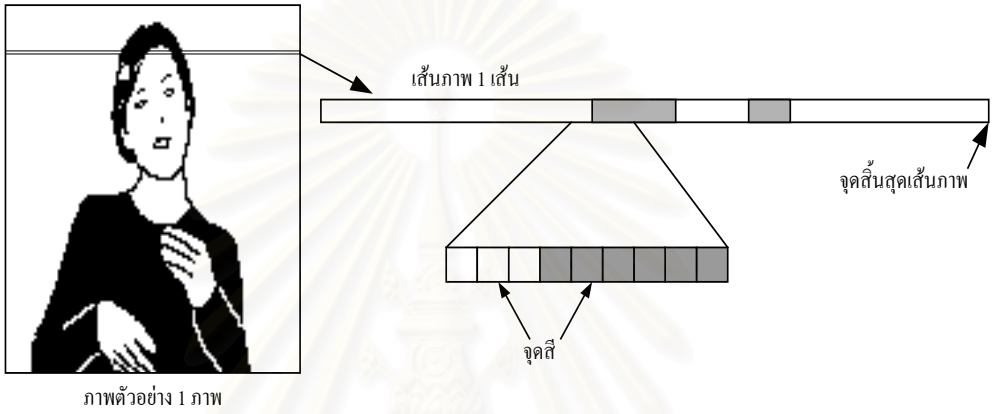
มาตรฐานการบีบอัดข้อมูลนี้ได้รับการพัฒนาขึ้นโดย Consultative Committee of the International Telephone and Telegraph (CCITT) และ International Standardization Organization (ISO) จัดเป็นวิธีบีบอัดแบบไม่สูญเสียข้อมูล และใช้กับการบีบอัดข้อมูลที่มีจุดสีเพียง 2 ระดับคือ ขาวและดำ มาตรฐานนี้ได้ถูกออกแบบเพื่อส่งภาพข้อมูล FAX ผ่านทางเครือข่ายสายโทรศัพท์ [5]

การบีบอัดข้อมูลสามารถลดขนาดข้อมูลโดยเฉลี่ยได้ 6-10 เท่า เช่น ภาพตัวอย่างขนาด 112 X 128 จุด จะมีข้อมูล 1792 ไบต์ เมื่อบีบอัดข้อมูลจะมีขนาดลดลงเหลือประมาณ 300 ไบต์ การบีบอัดจะเริ่มจากเส้นบนสุด (เส้นที่ 1) ไปจนถึงเส้นล่างสุด (เส้นที่128) และจากด้านซ้าย (จุดที่ 1) ถึง ด้านขวา (จุดที่112) โดยการเลื่อนจุดอ้างอิง ดังจะกล่าวต่อไป

2.3.1. หลักการเบื้องต้นของการบีบอัดข้อมูล

ภาพที่จะบีบอัดจะถูกแบ่งเป็นเส้นในแนวนอนที่มีความสูง 1 จุด (Pixel) เรียกว่า เส้นภาพ แต่ละเส้นภาพประกอบด้วยจุดสีที่มีความกว้าง 1 จุดเรียงต่อกันดังแสดงในรูปที่ 2.3 ข้อมูลจะถูกนำมาบีบอัดครั้งละ 1 เส้นภาพ โดยพิจารณาความสัมพันธ์ของตำแหน่งเปลี่ยนสีของจุดทั้งในแนวนอนและแนวตั้ง กล่าวคือ พิจารณาตำแหน่งเปลี่ยนสีของจุดในเส้นภาพและพิจารณาความสัมพันธ์ระหว่างตำแหน่งเปลี่ยนจุดสีในเส้นภาพที่ถูกบีบอัดกับตำแหน่งเปลี่ยนจุดสีของเส้นอ้างอิง

ซึ่งได้แก่ เส้นภาพที่เหนือขึ้นไป 1 เส้น สำหรับเส้นภาพเส้นแรกจะใช้เส้นภาพที่เป็นประกอบด้วยจุดขาวทั้งหมดเป็นเส้นอ้างอิง ตำแหน่งเปลี่ยนจุดสีจะถูกพิจารณาทีละจุดจากด้านซ้ายไปด้านขวาจนถึงจุดสิ้นสุดเส้นภาพเพื่อนำมาบีบอัดข้อมูล เมื่อบีบอัดเส้นภาพเรียบร้อยแล้ว เส้นภาพดังกล่าวจะเป็นเส้นอ้างอิงสำหรับการบีบอัดข้อมูลของเส้นภาพต่อไป ซึ่งอยู่ด้านล่างของเส้นภาพที่บีบอัดแล้ว การบีบอัดสิ้นสุดเมื่อเส้นภาพทุกเส้นถูกบีบอัด

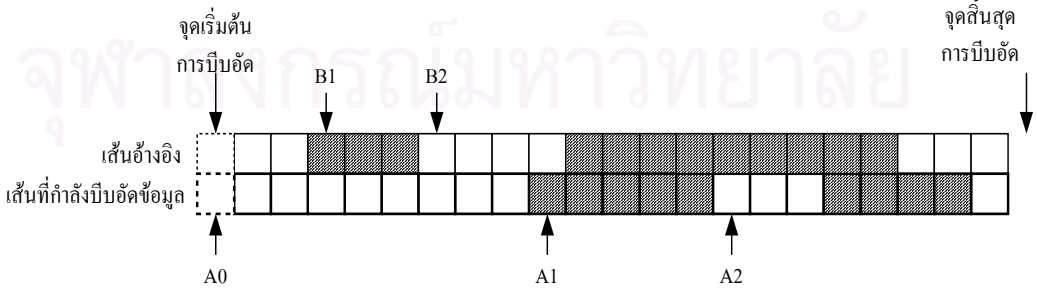


รูปที่ 2.3 ลักษณะของจุดสีและเส้นภาพ

2.3.2. นิยามของจุดเปลี่ยนสีในภาพ

จุดเปลี่ยนสี คือจุดที่มีสี (ขาวหรือดำ) ต่างจากจุดสีก่อนหน้าที่อยู่ติดกันบนเส้นภาพเส้นเดียวกัน [5]

จุดเปลี่ยนสีที่มีความสำคัญต่อการบีบอัดข้อมูลแบบ CCITT T.4 แบบ 2 มิติมี 5 จุดดังแสดงในรูปที่ 2.4 ได้แก่



รูปที่ 2.4 จุดเปลี่ยนสีที่ใช้ในการบีบอัดข้อมูล

A0 คือ จุดอ้างอิงหรือจุดเริ่มต้นที่ใช้บีบอัดข้อมูลของเส้น ที่จุดเริ่มต้นของแต่ละเส้นภาพจะสมมติให้ A0 ชี้ที่จุดสีสีขาวที่วางอยู่หน้าจุดสีจุดแรกของเส้นภาพ และตำแหน่งของ A0 จะมีการเปลี่ยนแปลงไปตามการบีบอัดข้อมูลแบบต่างๆ

A1 คือ ตำแหน่งจุดเปลี่ยนสีที่อยู่ถัดไปทางขวาของ A0 บนเส้นภาพ

A2 คือ ตำแหน่งจุดเปลี่ยนสีที่อยู่ถัดไปทางขวาของ A1 บนเส้นภาพ

B1 คือ ตำแหน่งจุดเปลี่ยนสีตำแหน่งแรกบนเส้นอ้างอิง ที่อยู่ทางขวาของ A0 และมีสีตรงข้ามกับ A0

B2 คือ ตำแหน่งจุดเปลี่ยนสีที่อยู่ถัดไปทางขวาของ B1 บนเส้นอ้างอิง

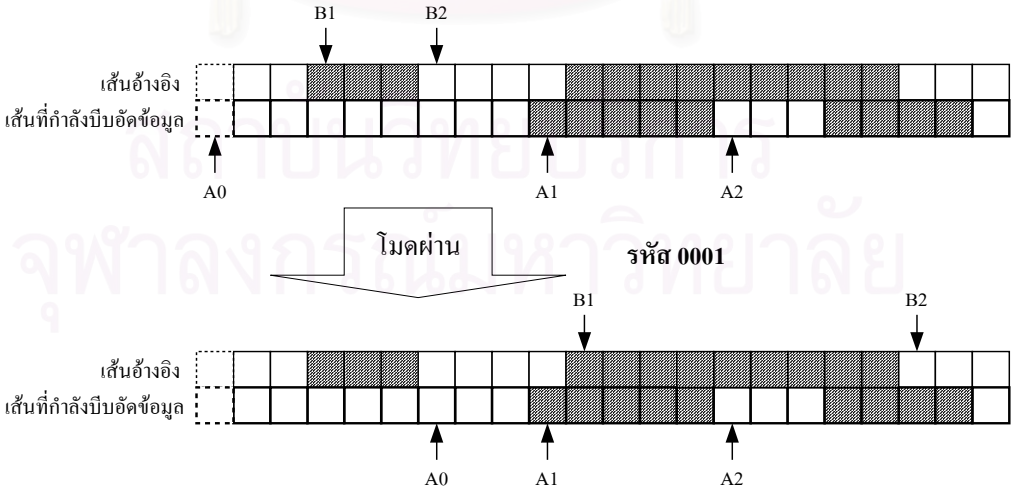
ตำแหน่งจุดเปลี่ยนสีทั้ง 5 จุด จะนำไปใช้บีบอัดข้อมูลในโมดต่างๆ ดังนี้

2.3.3. การบีบอัดข้อมูลในโมดต่างๆ

มาตรฐานการบีบอัดข้อมูลแบบ CCITT T.4 แบบ 2 มิติ มีโมดการบีบอัด 3 โมด [5] โดยเรียงลำดับตามความสำคัญ ดังนี้

ก. โมดผ่าน (Pass mode)

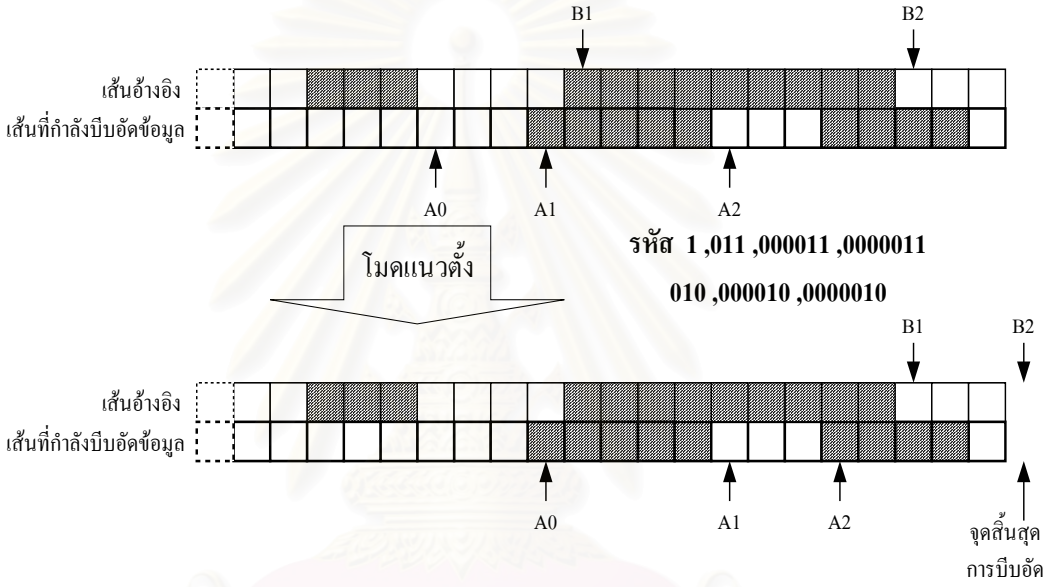
ตำแหน่งของจุด A1 จะอยู่ทางขวาของ B2 แต่กรณีที่ A1 อยู่ตำแหน่งเดียวกับ B2 จะไม่ถือว่าอยู่ในโมดนี้ เมื่อผ่านการบีบอัดข้อมูลของโมดผ่านแล้วจะได้รหัส "0001" และตำแหน่งของ A0 จะย้ายไปอยู่ตำแหน่งเดียวกับตำแหน่งของ B2 ดังแสดงรูปที่ 2.5



รูปที่ 2.5 ลักษณะการบีบอัดข้อมูลแบบโมดผ่าน

ข. โมดแนวตั้ง (Vertical mode)

ตำแหน่ง A1 ห่างจาก B1 ไม่เกิน 3 จุด กล่าวคือ ตำแหน่งของ A1 สามารถอยู่ตำแหน่งเดียวกับ B1 หรือ เยื้องไปทางขวาหรือทางซ้ายจาก B1 ได้ไม่เกิน 3 จุด เมื่อผ่านการบีบอัดข้อมูลในโมดนี้จะได้รหัสที่เป็นได้ทั้งหมด 7 ค่าขึ้นกับตำแหน่งของ A1 ดังที่จะกล่าวต่อไป สำหรับตำแหน่งของ A0 จะย้ายไปอยู่ที่ตำแหน่งของ A1 ดังแสดงในรูปที่ 2.6



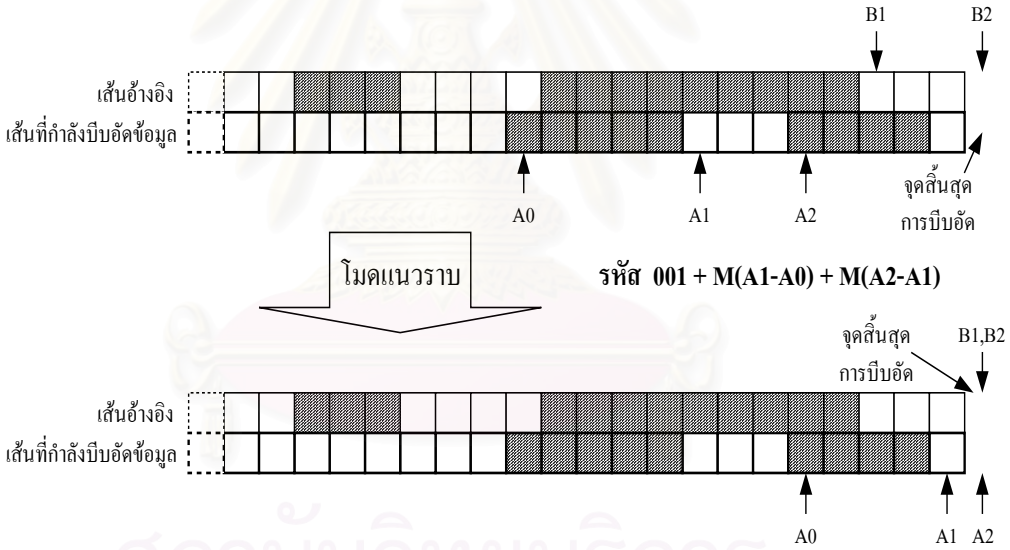
รูปที่ 2.6 ลักษณะการบีบอัดข้อมูลแบบโมดแนวตั้ง

ความสัมพันธ์ระหว่างตำแหน่งของ A1 กับ B1 จะให้รหัสต่างๆ ดังนี้

A1 อยู่ตรงกับ B1	ได้รหัส	"1"
A1 เยื้องทางขวา B1 ไป 1 จุด	ได้รหัส	"011"
A1 เยื้องทางขวา B1 ไป 2 จุด	ได้รหัส	"000011"
A1 เยื้องทางขวา B1 ไป 3 จุด	ได้รหัส	"0000011"
A1 เยื้องทางซ้าย B1 ไป 1 จุด	ได้รหัส	"010"
A1 เยื้องทางซ้าย B1 ไป 2 จุด	ได้รหัส	"000010"
A1 เยื้องทางซ้าย B1 ไป 3 จุด	ได้รหัส	"0000010"

ค. โมดแนวราบ (Horizontal mode)

การบีบอัดโมดนี้เกิดขึ้นเมื่อความสัมพันธ์ของตำแหน่งจุดเปลี่ยนสีทั้ง 5 จุดไม่สอดคล้องกับการบีบอัดทั้ง 2 โมดก่อนหน้านี้ โมดนี้จะนำการบีบอัดข้อมูลแบบ Run-lengths มาเป็นส่วนหนึ่งของคำรหัส ลักษณะของรหัสที่ได้จากการบีบอัดในโมดนี้คือ "001 + M(A1-A0) + M(A2-A1)" เมื่อ M( ) คือฟังก์ชั่นของการบีบอัดข้อมูลแบบ Run-lenghts, + หมายถึง การต่อรหัส และ A1-A0, A2-A1 คือ จำนวนจุดสี(ขาวหรือดำ)จากตำแหน่ง A0 ถึง A1 และ A1 ถึง A2 ตามลำดับ ซึ่งค่าฟังก์ชั่น M( ) ได้แสดงไว้ดังตารางที่ 2.2 และ 2.3 [5] การเลือกชุดรหัสของฟังก์ชั่น M( ) ว่าจะใช้ชุดรหัสของจุดสีขาวหรือจุดสีดำขึ้นอยู่กับตำแหน่งของ A0 ว่าอยู่ที่จุดสีใด หาก A0 อยู่ที่จุดสีดำฟังก์ชั่น M(A1-A0)จะเลือกใช้ชุดรหัสสีดำและฟังก์ชั่น M(A2-A1) จะเลือกใช้ชุดรหัสสีขาว ตัวอย่างการบีบอัดแสดงในรูปที่ 2.7 รหัสที่ได้ คือ 001 + 0011 + 1000



รูปที่ 2.7 ลักษณะการบีบอัดข้อมูลแบบโมดแนวราบ

กรณีที่จำนวนจุดสีจากจุด A0 ถึง A1 หรือ A1 ถึง A2 มีค่าไม่ตรงกับจำนวนจุดในชุดรหัส จำนวนจุดจะถูกแบ่งออกเป็นกลุ่มย่อย โดยแต่ละกลุ่มมีจำนวนจุดที่มีค่าตรงกับในชุดรหัส เช่น จุดสีดำ 95 จุด แบ่งเป็น สีดำ 64 จุด และ สีดำ 32 จุด รหัสที่ได้ คือ 001 + ชุดรหัสสีดำ 64 จุด + ชุดรหัสสีขาว 0 จุด + 001 + ชุดรหัสสีดำ 32 จุด + ชุดรหัสสีขาว 0 จุด

นอกจากการบีบอัดข้อมูลทั้ง 3 โมดแล้ว ยังมีรหัสเพิ่มเติมได้แก่ "0000001XXX" ค่า XXX คือ ค่าของบิตที่จะเติมเข้าไปเพื่อสร้างรหัสเอาไว้ใช้เฉพาะกับงานที่ได้นำการบีบอัดข้อมูลวิธีนี้ไปประยุกต์ใช้

รหัสที่ได้จากการบีบอัดข้อมูลสรุปได้ดังตารางที่ 2.1

โมด	**เงื่อนไข**	**คำรหัส**	**ตำแหน่งใหม่ของ A0 หลังบีบอัด**
ผ่าน	B2 < A1	0001	B2
แนวราบ	B2 > A1 และ |A1 - B1| > 3	001 + M(A1-A0) + M(A2-A1)	A2
แนวตั้ง	A1-B1 = 0	1	A1
	A1-B1 = 1	011	
	A1-B1 = 2	000011	
	A1-B1 = 3	0000011	
	B1-A1 = 1	010	
	B1-A1 = 2	000010	
	B1-A1 = 3	0000010	
ส่วนเพิ่มเติม		0000001xxx	

ตารางที่ 2.1 ตารางสรุปคำรหัส

White run length	Code word M( )	Black run length	Code word M( )
0	00110101	0	0000110111
1	000111	1	010
2	0111	2	11
3	1000	3	10
4	1011	4	011
5	1100	5	0011
6	1110	6	0010
7	1111	7	00011
8	10011	8	000101
9	10100	9	000100
10	00111	10	0000100
11	01000	11	0000101
12	001000	12	0000111
13	000011	13	00000100
14	110100	14	00000111
15	110101	15	000011000
16	101010	16	0000010111
17	101011	17	0000011000
18	0100111	18	0000001000
19	0001100	19	00001100111
20	0001000	20	00001101000
21	0010111	21	00001101100
22	0000011	22	00000110111
23	0000100	23	00000101000
24	0101000	24	00000010111
25	0101011	25	00000011000
26	0010011	26	000011001010
27	0100100	27	000011001011
28	0011000	28	000011001100
29	00000010	29	000011001101
30	00000011	30	000001101000
31	00011010	31	000001101001
32	00011011	32	000001101010
33	00010010	33	000001101011
34	00010011	34	000011010010
35	00010100	35	000011010011
36	00010101	36	000011010100
37	00010110	37	000011010101
38	00010111	38	000011010110
39	00101000	39	000011010111
40	00101001	40	000001101100
41	00101010	41	000001101101
42	00101011	42	000011011010
43	00101100	43	000011011011
44	00101101	44	000001010100
45	00000100	45	000001010101
46	00000101	46	000001010110
47	00001010	47	000001010111
48	00001011	48	000001100100
49	01010010	49	000001100101

ตารางที่ 2.2 คำรหัส Run-lengths จุดสีขาวและดำ หรือ ฟังก์ชั่น M( )

White run length	Code word (M( ))	Black run length	Code word (M( ))
50	01010011	50	000001010010
51	01010100	51	000001010011
52	01010101	52	000000100100
53	00100100	53	000000110111
54	00100101	54	000000111000
55	01011000	55	000000100111
56	01011001	56	000000101000
57	01011010	57	000001011000
58	01011011	58	000001011001
59	01001010	59	000000101011
60	01001011	60	000000101100
61	00110010	61	000001011010
62	00110011	62	000001100110
63	00110100	63	000001100111

ตารางที่ 2.2 (ต่อ) คำรหัส Run-lengths จุดสีขาวและดำ หรือ ฟังก์ชั่น M( )

White run length	Code word (M( ))	Black run length	Code word (M( ))
64	11011	64	0000001111
128	10010	128	000011001000
192	010111	192	000011001001
256	0110111	256	000001011011
320	00110110	320	000000110011
384	00110111	384	000000110100
448	01100100	448	000000110101
512	01100101	512	0000001101100
576	01101000	576	0000001101101
640	01100111	640	0000001001010
704	011001100	704	0000001001011
768	011001101	768	0000001001100
832	011010010	832	0000001001101
896	011010011	896	0000011110010
960	011010100	960	0000001110011
1024	011010101	1024	0000001110100
1088	011010110	1088	0000001110101
1152	011010111	1152	0000001110110
1216	011011000	1216	0000001110111
1280	011011001	1280	0000001010010
1344	011011010	1344	0000001010011
1408	011011011	1408	0000001010100
1472	010011000	1472	0000001010101
1536	010011001	1536	0000001011010
1600	010011010	1600	0000001011011
1664	011000	1664	0000001100100
1728	010011011	1728	0000001100101

ตารางที่ 2.3 คำรหัสเพิ่มเติมของจำนวนจุดระหว่าง 64 ถึง 1728 หรือ ฟังก์ชั่น M( )

2.3.4. กระบวนการบีบอัดข้อมูล

ผังงานการบีบอัดข้อมูลแสดงดังรูปที่ 2.8 [6] และมีรายละเอียดโดยย่อดังนี้

1. สร้างเส้นอ้างอิงสีขาวเพื่อใช้ในการอ้างอิงสำหรับการเข้ารหัสเส้นภาพเส้นแรก และกำหนดตำแหน่ง A0 ลงบนเส้นภาพที่ต้องการบีบอัดข้อมูลโดยไว้ที่ตำแหน่งก่อนหน้าจุดแรกของเส้นภาพ

2. ตรวจหาตำแหน่ง A1, B1 และ B2

3. ตรวจสอบการบีบอัดโมดผ่าน ซึ่งเกิดขึ้นในกรณีที่ ตำแหน่ง B2 อยู่ทางซ้ายของตำแหน่ง A1 ถ้าตรวจสอบไม่พบโมดผ่านจะตรวจสอบโมดต่อไป (ข้อ 4) หลังจากบีบอัดโมดผ่านแล้วตำแหน่งของ A0 จะย้ายมาที่ตำแหน่งของ B2 และกลับไปเริ่มทำงานที่ข้อ 2

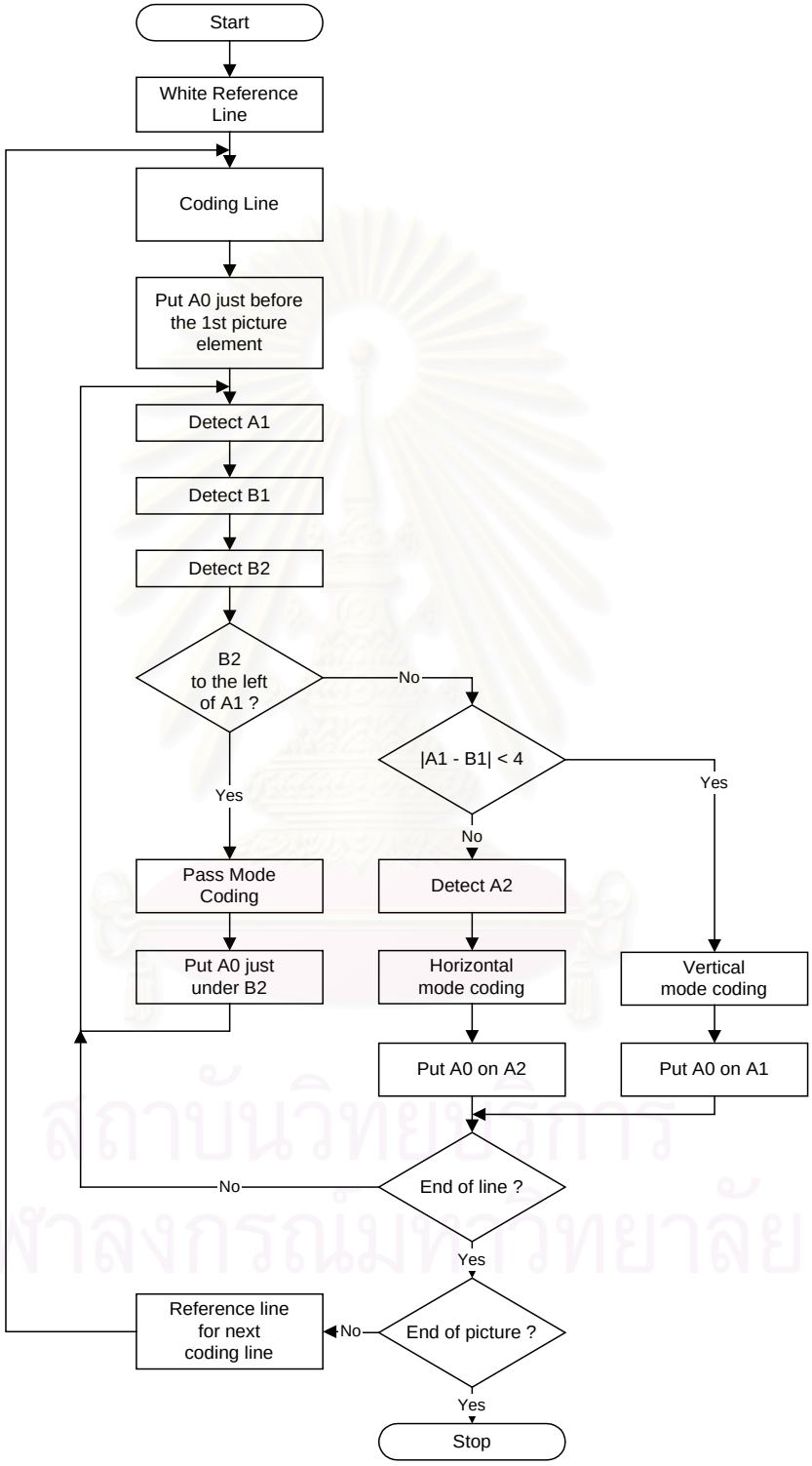
4. ตรวจสอบระยะห่างระหว่าง A1 กับ B1 หากมีค่าไม่เกิน 3 จะเป็นโมดแนวตั้ง และตำแหน่งของ A0 จะย้ายไปที่ตำแหน่งของ A1

5. หากระยะห่างของ A1 กับ B1 มีค่ามากกว่า 3 จะเป็นโมดแนวราบ ซึ่งต้องหาตำแหน่งของ A2 เพื่อนำไปใช้คำรหัส หลังเข้ารหัสตำแหน่งของ A0 จะย้ายไปที่ตำแหน่ง A2

6. เมื่อบีบอัดข้อมูล 1 ครั้ง จะต้องตรวจสอบว่าข้อมูลมาถึงจุดสิ้นสุดของเส้นภาพหรือยัง ถ้ายังไม่สิ้นสุดจะกลับไปเริ่มทำงานที่ข้อ 3

7. ถ้าการบีบอัดถึงจุดสิ้นสุดของเส้นภาพ จะต้องตรวจสอบว่าสิ้นสุดการบีบอัดข้อมูลของภาพหรือยัง หากยังไม่สิ้นสุด จะเปลี่ยนเส้นอ้างอิงมาเป็นเส้นภาพที่ผ่านการบีบอัดข้อมูลเส้นล่าสุดและกลับไปเริ่มทำงานที่ข้อ 2

8. ถ้าพบการสิ้นสุดของข้อมูลภาพก็จะหยุดการบีบอัดข้อมูล และรอทำการบีบอัดข้อมูลภาพต่อๆไป



รูปที่ 2.8 ผังงานการบีบอัดข้อมูล

5. ตัวอย่างการบีบอัดข้อมูลตามมาตรฐาน CCITT T.4 แบบ 2 มิติ

ภาพตัวอย่างที่ใช้บีบอัดข้อมูล มีขนาด 112 X 128 จุดแสดงดังรูปที่ 2.9

รูปที่ 2.9 รูปตัวอย่างที่ใช้บีบอัดข้อมูล

การบีบอัดที่แสดงใช้เส้นภาพจำนวน 55 เส้นจากภาพตัวอย่างในรูปที่ 2.8 มาแปลงเป็นข้อมูลดิจิตอลโดย '0' แทนสีดำ '1' แทนสีขาว จะได้ข้อมูลดังนี้

```
1111111111111111111111111111111111111111111111110000000000011111111111111111111111111111111111111111111111111111
1111111111111111111111111111111111111111111111000000000000000111111111111111111111111111111111111111111111111111
1111111111111111111111111111111111111111111111000000000000000111111111111111111111111111111111111111111111111111
1111111111111111111111111111111111111111111111000000000000000111111111111111111111111111111111111111111111111111
1111111111111111111111111111111111111111111100000000000000000001111111111111111111111111111111111111111111111111
1111111111111111111111111111111111111111111100000000000000000001111111111111111111111111111111111111111111111111
1111111111111111111111111111111111111111111100000000000000000000111111111111111111111111111111111111111111111111
1111111111111111111111111111111111111111111000000000000000000000011111111111111111111111111111111111111111111111
1111111111111111111111111111111111111111100000000000011111000000011111111111111111111111111111111111111111111111
1111111111111111111111111111111111111111100000000000011111110000111111111111111111111111111111111111111111111111
1111111111111111111111111111111111111111000000000000111111110000011111111111111111111111111111111111111111111111
1111111111111111111111111111111111111111000100000001111111111100001111111111111111111111111111111111111111111111
1111111111111111111111111111111111111111000000000001111111111100000111111111111111111111111111111111111111111111
1111111111111111111111111111111111111110000000000001111111111100000111111111111111111111111111111111111111111111
1111111111111111111111111111111111111111010101000011111111111100000111111111111111111111111111111111111111111111
1111111111111111111111111111111111111111010101000011111111111100000111111111111111111111111111111111111111111111
1111111111111111111111111111111111111111010101000011111111111100000111111111111111111111111111111111111111111111
1111111111111111111111111111111111111110001010101111111111111110000111111111111111111111111111111111111111111111
1111111111111111111111111111111111111110001100001111111111110011100001111111111111111111111111111111111111111111
1111111111111111111111111111111111111110001100001111111111011111000011111111111111111111111111111111111111111111
1111111111111111111111111111111111111110001000001111111111011111100001111111111111111111111111111111111111111111
1111111111111111111111111111111111111110000000001111111111111111100000111111111111111111111111111111111111111111
1111111111111111111111111111111111111110000000011001111111110001000001111111111111111111111111111111111111111111
1111111111111111111111111111111111111110010000010111111111101101100001111111111111111111111111111111111111111111
1111111111111111111111111111111111111110010000011111111111110011100001111111111111111111111111111111111111111111
1111111111111111111111111111111111111110000001111111111111111111111100011111111111111111111111111111111111111111
1111111111111111111111111111111111111110000001110001111111111111111100111111111111111111111111111111111111111111
1111111111111111111111111111111111111100000010010111111111111111111100011111111111111111111111111111111111111111
1111111111111111111111111111111111111100000010001111111111111111111100011111111111111111111111111111111111111111
1111111111111111111111111111111111111100000011111111111111111111111100011111111111111111111111111111111111111111
1111111111111111111111111111111111111100000011111111111111111111111100011111111111111111111111111111111111111111
1111111111111111111111111111111111111100000011111111111111111111111100011111111111111111111111111111111111111111
1111111111111111111111111111111111111100000011111111111111111111111100111111111111111111111111111111111111111111
1111111111111111111111111111111111111100000011111111111111011111111101111111111111111111111111111111111111111111
1111111111111111111111111111111111111100000011111111111111011111111101111111111111111111111111111111111111111111
1111111111111111111111111111111111111100000011111111111111111111111101111111111111111111111111111111111111111111
1111111111111111111111111111111111111110011011111111111111111111111101111111111111111111111111111111111111111111
1111111111111111111111111111111111111110011011111111111111111111111101111111111111111111111111111111111111111111
1111111111111111111111111111111111111110001111111111111111111111111101111111111111111111111111111111111111111111
1111111111111111111111111111111111111110001111111111111000000111001111111111111111111111111111111111111111111111
1111111111111111111111111111111111111110000111111111111001101111110111111111111111111111111111111111111111111111
1111111111111111111111111111111111111111000111111111110111110111110111111111111111111111111111111111111111111111
1111111111111111111111111111111111111111000011111111110000001111110111111111111111111111111111111111111111111111
1111111111111111111111111111111111111111000011111111111111111111110111111111111111111111111111111111111111111111
1111111111111111111111111111111111111111100010011111111111111111110111111111111111111111111111111111111111111111
1111111111111111111111111111111111111111100010011111111111111111110111111111111111111111111111111111111111111111
1111111111111111111111111111111111111111111100000111111111111111101111111111111111111111111111111111111111111111
1111111111111111111111111111111111111111111110000111111111111111101111111111111111111111111111111111111111111111
1111111111111111111111111111111111111111111110000111111111111111101111111111111111111111111111111111111111111111
1111111111111111111111111111111111111111111111110011111111111111011111111111111111111111111111111111111111111111
1111111111111111111111111111111111111111111111111001111111111100111111111111111111111111111111111111111111111111
1111111111111111111111111111111111111111111111111001111111111100111111111111111111111111111111111111111111111111
1111111111111111111111111111111111111111111111111010111111111001011111111111111111111111111111111111111111111111
1111111111111111111111111111111111111111111111111011011111110011101111111111111111111111111111111111111111111111
1111111111111111111111111111111111111111111111111011100000000111011111111111111111111111111111111111111111111111
```

ข้อมูลเส้นแรกที่นำมาบีบอัดคือ

|---------- 50 จุด ----------|-- 10 จุด --|---------- 52 จุด ----------|

11111111111111111111111111111111111111111111111111111000000000011111111111111111111111111111111111111111111111111111111111111

เส้นอ้างอิงของเส้นภาพนี้คือ เส้นสีขาว (ข้อมูล '1' ทั้งหมด) เนื่องจากเป็นเส้นภาพเส้นแรก จุดเปลี่ยนสีต่างๆมีค่าดังนี้

A0 มีค่าเท่ากับ 0

A1 มีค่าเท่ากับ 50

A2 มีค่าเท่ากับ 60

B1, B2 มีค่าเท่ากับ 113 (จุดสิ้นสุดเส้นภาพ)

ตำแหน่ง B2 อยู่ทางขวาของ A1 และ ระยะห่างของ A1 กับ B1 มีมากกว่า 3 จึงเข้าการบีบอัดในโมดแนวราบ คือ 001 + M(A1-A0) + M(A2-A1) ซึ่งได้แก่ 001 + คำรหัส Run-lengths สีขาว 50 จุด + คำรหัส Run-lengths สีดำ 10 จุด ซึ่งจะได้คำรหัส ดังนี้

" 001 + 01010011 + 0000100 "

หลังเข้ารหัสแล้ว A0 จะมาอยู่ที่จุด A2 ซึ่งได้แก่ จุดเปลี่ยนสีจุดที่ 2 ทำให้ได้ตำแหน่งใหม่ของ A1, B1 และ B2 อยู่ที่จุดสิ้นสุดเส้นภาพ เมื่อบีบอัดต่อ จะพบว่า A1 อยู่ตำแหน่งเดียวกับ B1 จึงเป็นการบีบอัดในโมดแนวตั้งที่มีระยะห่างของ A1 กับ B1 เป็น 0 ซึ่งได้คำรหัส เป็น " 1 "

ดังนั้นผลการบีบอัดของเส้นภาพเส้นแรก คือ

" 00101010011000010011 "

เมื่อทำการบีบอัดข้อมูลต่อไปจะได้คำรหัสทั้งหมดออกมาดังตารางที่ 2.4

รหัสข้อมูลจากการบีบอัดเส้นภาพแต่ละเส้น	**จำนวนบิต**	**ขนาดลดลง (เท่า)**
001010100110000100,1	19	5.89
000010,000011,1	13	8.62
1,1,1	3	37.33
1,1,1	3	37.33
000010,000011,1	13	8.62
1,1,1	3	37.33
1,011,1	5	22.40
010,011,1	7	16.00
000010,001000010111110,1,1	22	5.09
010,1,000011,1,1	12	9.33

ตารางที่ 2.4 ผลการบีบอัดตัวอย่างภาพของแต่ละเส้นภาพ

รหัสข้อมูลจากการบีบอัดเส้นภาพแต่ละเส้น	จำนวนบิต	ขนาดลดลง (เท่า)
1,010,1,011,1	9	12.44
1,00110000111,010,011,1,1	20	5.60
1,0001,1,011,000011,1	16	7.00
010,1,1,1,1	7	16.00
1,001010000111,001010000111,001010000111,010,1,1,1	48	2.33
1,1,1,1,1,1,1,1,1,1,1	11	10.18
1,1,1,1,1,1,1,1,1,1,1	11	10.18
1,0001,1,1,1,1,0000010,001000111010,011,1,1	33	3.39
1,1,011,0001,1,0010100011,011,011,1	27	4.15
1,1,1,1,010,000010,1,1,1	16	7.00
1,1,010,1,010,010,1,1,1	15	7.47
1,0001,010,0001,1,011,1	17	6.59
1,010,001011111,0011010010,1,1,1	26	4.31
1,00111000111,1,010,000010,010,0000010,0000010,1,011,1,1	45	2.49
1,1,1,1,0001,011,010,1,1,1	17	6.59
1,0001,010,0001,011,1,1	17	6.59
1,1,001100010,011,1,1	16	7.00
010,010,000010,0000010,001000111010,010,1,1	36	3.11
1,1,1,011,1,1,1	9	12.44
1,1,0001,1,1,1	9	12.44
1,1,1,1,1	5	22.40
1,1,1,1,1	5	22.40
1,1,011,1,1	7	16.00
1,1,001000011010,1,010,1	19	5.89
1,1,1,1,1,1,1	7	16.00
1,1,0001,1,1,1	9	12.44
011,0000010,0010111010,1,1,1	23	4.87
1,1,1,1,1,1,1	7	16.00
1,011,0001,1,1,1	11	10.18
1,1,0010000110010,000010,010,1	25	4.48
1,011,1,001110111,010,1,010,1	22	5.09
011,1,010,000010,1,1,1,1,1	18	6.22
1,011,1,0001,1,1,1,1	13	8.62
1,1,0001,1,1,1	9	12.44
011,1,001000011111,1,1,1	18	6.22
1,1,1,1,1,1,1	7	16.00
000011,0001,011,010,010,1	20	5.60
011,1,1,1,1	7	16.00
1,1,1,1,1	5	22.40
0000011,011,010,010,1	17	6.59
011,011,000010,010,1	16	7.00
1,1,1,1,1	5	22.40
1,010,001000111010,010,010,001000111010,1	35	3.2
1,1,011,0010101110,000010,1,1,1	24	4.67
1,1,011,010,1,1,1	11	10.18
รวม	850	7

ตารางที่ 2.4 (ต่อ) ผลการบีบอัดตัวอย่างภาพของแต่ละเส้นภาพ

อัตราการบีบอัดของภาพแต่ละเส้นในตาราง 2.4 มีค่าไม่เท่ากันเนื่องจากข้อมูลของเส้นภาพไม่เหมือนกัน บางเส้นบีบอัดได้ถึง 37 เท่าเพราะมีจุดเปลี่ยนสีน้อยและจุดเปลี่ยนสีอยู่ตำแหน่งเดียวกับจุดเปลี่ยนสีของเส้นอ้างอิง แต่บางเส้นบีบอัดได้เพียง 3 เท่า

เพราะมีจุดเปลี่ยนสีหลายจุดและตำแหน่งของจุดเปลี่ยนสีอยู่ห่างกับจุดเปลี่ยนสีของเส้นอ้างอิงมากจึงใช้โมดแนวราบบีบอัด ซึ่งจะได้รหัสยาวกว่าโมดอื่น

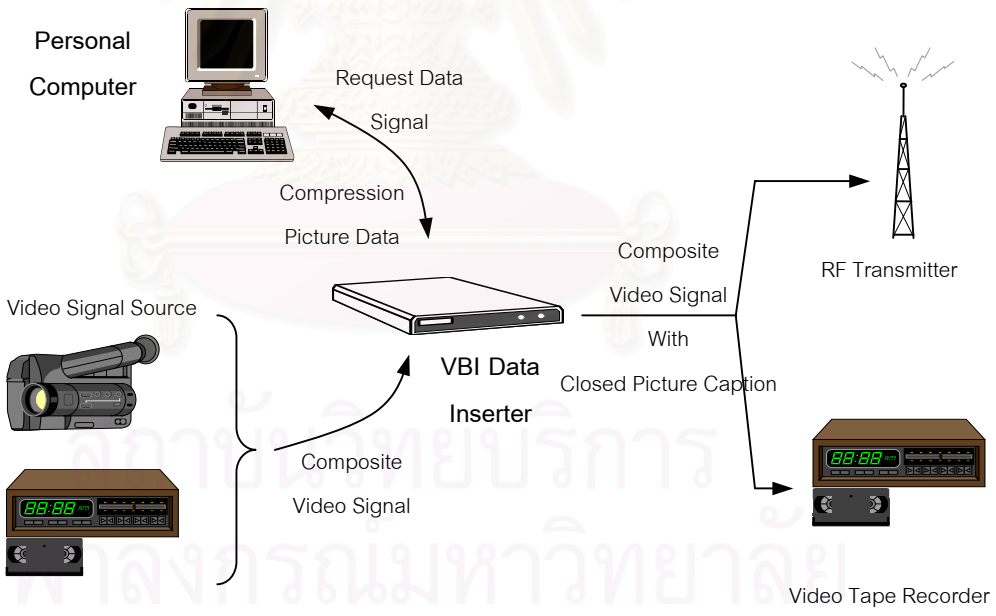
ผลการบีบอัดจะได้ว่าข้อมูลมีขนาดลดลงจาก 6160 บิต เหลือ 850 บิต คิดเป็นอัตราประมาณ 7:1 ซึ่งอัตราการบีบอัดจะมากหรือน้อยขึ้นกับลักษณะภาพที่บีบอัดว่ามีจุดเปลี่ยนสีและความต่อเนื่องของลายเส้นอย่างไร ภาพที่มีจุดเปลี่ยนสีน้อยลายเส้นต่อเนื่องจะบีบอัดได้มาก ภาพที่จุดเปลี่ยนสีมากและลายเส้นไม่ต่อเนื่องจะบีบอัดได้น้อย

บทที่ 3

ระบบวีดิทัศน์ที่ซ่อนภาพบรรยายได้

3.1 โครงสร้างรวมของระบบวีดิทัศน์ที่ซ่อนภาพบรรยายได้

ระบบวีดิทัศน์ที่ซ่อนภาพบรรยายได้เป็นระบบที่แทรกภาพบรรยายเข้าไปในช่วงไร้ภาพแนวตั้งของสัญญาณวีดิทัศน์ระบบ PAL ซึ่งผู้ชมสามารถเลือกที่จะเปิดหรือปิดภาพบรรยายที่แทรกเข้าไปในสัญญาณวีดิทัศน์ได้ ระบบนี้ประกอบด้วยด้านส่งและด้านรับ ระบบด้านส่งประกอบด้วยส่วนประกอบหลัก 2 ส่วน คือ เครื่องแทรกข้อมูลภาพและเครื่องคอมพิวเตอร์ส่วนบุคคล ดังแสดงในรูปที่ 3.1

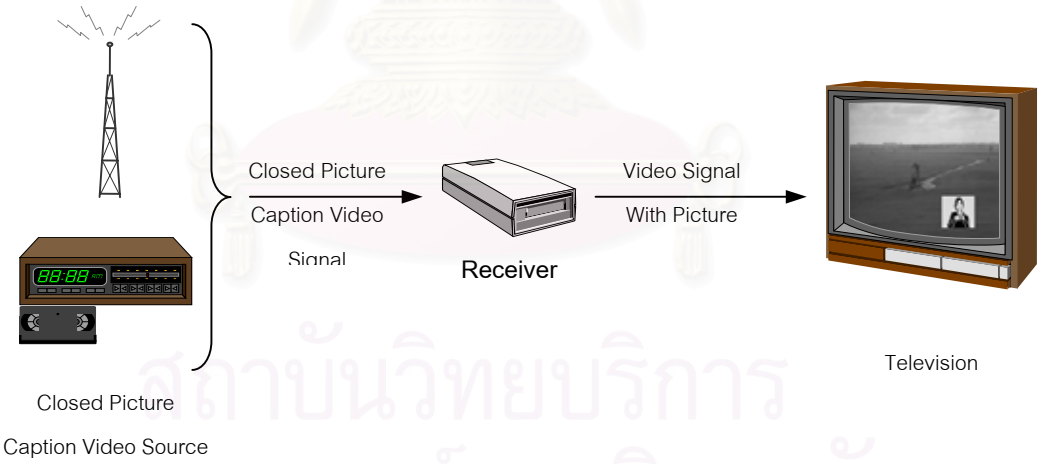


รูปที่ 3.1 โครงสร้างด้านส่งของระบบวีดิทัศน์ที่ซ่อนภาพบรรยายได้

ข้อมูลภาพที่แทรกลงในสัญญาณวีดิทัศน์รวมจะถูกบีบอัดตามมาตรฐาน CCITT T.4 แบบ 2 มิติด้วยโปรแกรมบีบอัดข้อมูลบนเครื่องคอมพิวเตอร์ส่วนบุคคล (Personal Computer) และจัดเก็บไว้รอการส่งข้อมูลให้กับเครื่องแทรกข้อมูลภาพ (VBI Data Inserter) เมื่อเริ่มแทรกข้อมูลภาพ

เครื่องแทรกข้อมูลภาพจะส่งสัญญาณขอข้อมูล (Request Data Signal) ผ่านทางพอร์ตอนุกรม (Serial Port) ตามมาตรฐาน RS232 ให้เครื่องคอมพิวเตอร์ส่วนบุคคลทุกๆ ฟิลด์ โปรแกรมบนเครื่องคอมพิวเตอร์ส่วนบุคคลจะตรวจจับสัญญาณขอข้อมูล และส่งข้อมูลภาพที่ถูกบีบอัดแล้วให้กับเครื่องแทรกข้อมูลภาพ เส้นสัญญาณ 1 เส้นสามารถส่งข้อมูลได้ 3 ไบต์ และใน 1 ฟิลด์ใช้จำนวนเส้นสัญญาณ 16 เส้น ดังนั้นการส่งข้อมูล 1 ฟิลด์จะส่งข้อมูล 48 ไบต์ ข้อมูลจะส่งผ่านทางพอร์ตอนุกรมด้วยอัตราการส่ง 28.8 kbit/sec เครื่องแทรกข้อมูลภาพจะนำข้อมูลเก็บในหน่วยความจำภายใน (Internal RAM) เมื่อเครื่องแทรกข้อมูลภาพได้รับสัญญาณวีดิทัศน์รวมจากแหล่งกำเนิดสัญญาณวีดิทัศน์ (Video Signal Source) หรือรายการที่ต้องการแทรกคำบรรยายภาพ เครื่องแทรกข้อมูลภาพจะนำข้อมูลที่เก็บอยู่ในหน่วยความจำภายในแทรกลงในช่วงไร้ภาพแนวตั้งของสัญญาณวีดิทัศน์ และส่งเป็นสัญญาณวีดิทัศน์ที่มีข้อมูลของภาพบรรยายซ่อนอยู่ได้ไปยังเครื่องบันทึกสัญญาณวีดิทัศน์หรือออกอากาศต่อไป

สำหรับระบบด้านรับจะมีเครื่องรับเป็นส่วนประกอบหลัก เครื่องรับมีหน้าที่ในการดึงข้อมูลภาพที่ถูกบีบอัดจากช่วงไร้ภาพแนวตั้งในสัญญาณวีดิทัศน์, คลายข้อมูลภาพ และแสดงผลขึ้นบนจอภาพ รายละเอียดของด้านรับแสดงดังรูปที่ 3.2



รูปที่ 3.2 ด้านรับของระบบวีดิทัศน์ที่ซ่อนภาพบรรยายได้

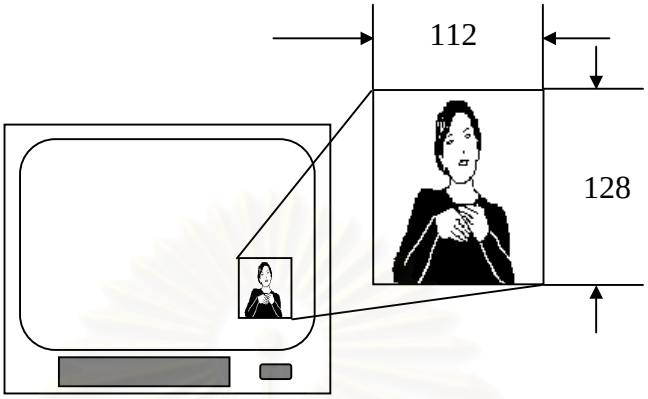
สัญญาณวีดิทัศน์ที่มีข้อมูลของภาพบรรยายแทรกอยู่จะถูกส่งมายังเครื่องรับ เพื่อแยกข้อมูลภาพที่ถูกบีบอัดและข้อมูลที่ใช้ควบคุมการแสดงภาพอื่นๆ ออกมา ข้อมูลเหล่านี้จะถูกนำไปเก็บไว้ในหน่วยความจำภายในเพื่อรอการคลายข้อมูล เมื่อรับข้อมูลครบทั้ง 48 ไบต์แล้ว ข้อมูลจะถูกนำมาผ่านกระบวนการคลายข้อมูลให้ได้ภาพบรรยายที่พร้อมจะแสดงบนจอภาพ ภาพบรรยายที่ได้

นี้จะถูกเก็บไว้ที่หน่วยความจำภายนอก ซึ่งมีอยู่ 2 แห่ง การเก็บภาพบรรยายจะเก็บข้อมูลภาพที่กำลังคลายข้อมูลในหน่วยความจำที่ยังไม่มีการแสดงผลและแสดงภาพบรรยายบนจอภาพทันทีที่การคลายข้อมูลเสร็จเรียบร้อย นอกจากมีการส่งรหัสควบคุมให้เก็บภาพบรรยายที่คลายข้อมูลเสร็จแล้วเอาไว้ เพื่อรอการแสดงผลในครั้งถัดไป ในการรับชมภาพบรรยายผู้ชมสามารถเปิดหรือปิดภาพบรรยายที่แทรกมากับสัญญาณวีดิทัศน์ได้โดยการเปิด-ปิดสวิตช์เลือกการรับชมภาพบรรยายที่เครื่องรับ

3.2 ข้อมูลภาพที่แทรกในสัญญาณวีดิทัศน์

ภาพบรรยายที่แทรกในสัญญาณวีดิทัศน์เป็นภาพภาษามือขาวดำที่มีลายเส้นต่อเนื่องขนาด 112 X 128 จุด มีขนาดข้อมูล 1792 ไบต์ ดังภาพตัวอย่างในรูปที่ 3.3 การแสดงภาพบรรยายบนจอโทรทัศน์จะใช้จำนวนเส้นสัญญาณวีดิทัศน์จำนวน 128 เส้น จำนวนเส้นดังกล่าวมีขนาดเท่ากับความสูงของรูป โดยแบ่งเป็นฟิลด์ละ 64 เส้น คิดเป็นอัตราส่วนเส้นภาพประมาณ 1 ต่อ 4.5 และใช้จำนวนจุด 112 จุดในสัญญาณวีดิทัศน์ 1 เส้น แต่ละจุดใช้ความถี่สำหรับแสดง 12 MHz คิดเป็นเวลา 0.083 μS (1/12MHz) ดังนั้นการแสดงภาพแนวนอนจะใช้เวลาทั้งหมด 9.33 μS คิดเป็นอัตราส่วนเวลาประมาณ 1 ต่อ 6 จากขนาดภาพดังกล่าวทำให้การแสดงผลบนจอภาพใช้พื้นที่ประมาณ 1 ใน 27 ของจอภาพ ภาพบรรยายจะแสดงที่มุมขวาล่างของจอภาพดังแสดงในรูปที่ 3.4

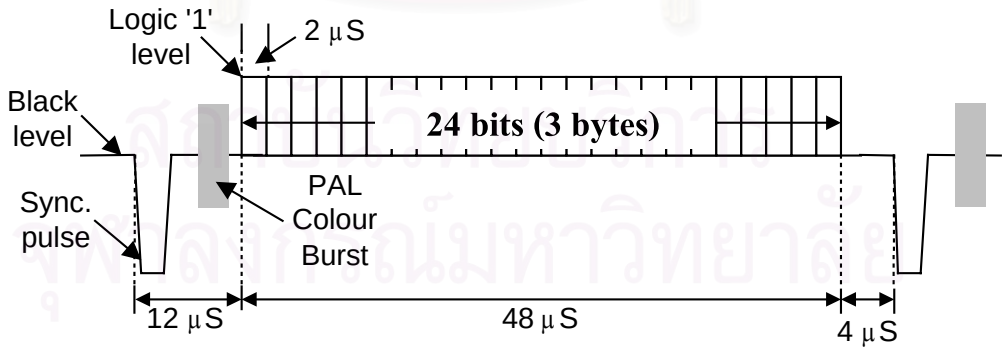
รูปที่ 3.3 ตัวอย่างภาพที่นำมาแทรก



รูปที่ 3.4 ภาพบรรยายเมื่อแสดงบนจอภาพ

3.3 เนื้อที่ในสัญญาณวีดิทัศน์สำหรับแทรกข้อมูลภาพ

ข้อมูลภาพที่ถูกบีบอัดแล้วสามารถนำมาแทรกในสัญญาณวีดิทัศน์ได้ตั้งแต่เส้นที่ 6 ถึง 22 ของแต่ละฟีลด์ ในการพัฒนาโครงงานนี้จะแทรกข้อมูลภาพที่ถูกบีบอัดแล้วลงในเส้นที่ 7 ถึง 22 คิดเป็นจำนวนเส้นสัญญาณทั้งหมด 16 เส้น แต่ละเส้นจะมีเวลาในการส่งข้อมูลได้ 54 μS ข้อมูลจะส่งที่อัตรา 500 kbit/sec ทำให้ส่งข้อมูลได้ 27 บิต แต่ในการส่งจริงจะส่งข้อมูลเพียง 24 บิตหรือ 3 ไบต์ ลักษณะของข้อมูลที่แทรกอยู่ในช่วงไร้ภาพแนวตั้งแสดงดังรูป 3.5



รูปที่ 3.5 ข้อมูลภาพในช่วงไร้ภาพแนวตั้ง

เนื่องจากเส้นสัญญาณวีดิทัศน์ 1 เส้นสามารถแทรกข้อมูลได้ 3 ไบต์ ดังนั้นใน 1 ฟีลด์ สามารถส่งข้อมูลได้ 48 ไบต์ และ 1 เฟรมประกอบด้วยฟีลด์ 2 ฟีลด์คือ ฟีลด์คู่และฟีลด์คี่ 1 เฟรม

จึงส่งข้อมูลได้ 96 ไบต์ แต่ข้อมูลภาพบรรยาย 1 ภาพมีขนาด 1792 ไบต์ ดังนั้นถ้าไม่มีการบีบอัดข้อมูลการส่งภาพ 1 ภาพจะใช้จำนวนเฟรมทั้งหมด 19 เฟรม หรือคิดเป็นเวลาประมาณ 0.76 วินาที เพื่อลดเวลาที่ใช้ส่งข้อมูลจึงต้องลดจำนวนข้อมูลโดยนำข้อมูลที่จะส่งไปผ่านการบีบอัดข้อมูลก่อน ข้อมูลที่ผ่านการบีบอัดข้อมูลแล้วจะมีขนาดลดลงเหลือประมาณ 300 ไบต์หรือลดลงประมาณ 6 เท่า เนื้อที่ในการส่งจึงน้อยลงทำให้สามารถส่งข้อมูลภาพบรรยาย 1 ภาพได้โดยใช้จำนวนเฟรมเพียง 3 เฟรมกับ 1 ฟีลด์ หรือคิดเป็นเวลาประมาณ 0.14 วินาที ทำให้ใช้เวลาที่ใช้ส่งข้อมูลลดลงประมาณ 5.4 เท่า ซึ่งทำให้สามารถส่งภาพต่อเนื่องที่มีความเร็วประมาณ 7 ภาพต่อวินาที

3.4 รูปแบบของข้อมูลที่ส่ง

ข้อมูลภาพบรรยายที่ถูกบีบอัดแล้วเป็นส่วนหนึ่งของข้อมูลทั้งหมดที่จะส่งให้เครื่องรับเพื่อนำไปใช้คลายข้อมูล ข้อมูลที่เพิ่มเติมเข้าจะนำไปใช้ประโยชน์สำหรับคลายข้อมูลและการแสดงผลบนจอภาพ ข้อมูลต่างๆที่ส่งให้เครื่องรับแบ่งออกเป็นส่วนที่สำคัญ 5 ส่วน คือ

- ไบต์เริ่มต้น (Start Byte)
- ไบต์ควบคุม (Control Byte)
- ไบต์ข้อมูล (Data Byte)
- ไบต์ขนาด (Size Byte)
- ไบต์สิ้นสุดข้อมูล (Stop Byte)

ข้อมูลเหล่านี้จะถูกเพิ่มลงไปด้วยโปรแกรมบีบอัดข้อมูลที่ทำงานงานบนเครื่องคอมพิวเตอร์ส่วนบุคคล ซึ่งมีรูปแบบการจัดวางตำแหน่งของไบต์ต่างๆ ดังแสดงในรูปที่ 3.6 ข้อมูลดังกล่าวจะถูกส่งจากเครื่องคอมพิวเตอร์ส่วนบุคคลไปยังเครื่องแทรกข้อมูลภาพครั้งละ 48 ไบต์เพื่อให้เครื่องแทรกข้อมูลภาพนำข้อมูลดังกล่าวไปแทรกลงในช่วงไร้ภาพแนวตั้งระหว่างเส้นที่ 7 ถึง 22 เส้นละ 3 ไบต์

Start						Stop
00000011	00000011	00000011	Control	Size(n)	Compressed Data	00000000
	3 Bytes		1 Byte	2 Bytes	n Bytes	1 Byte

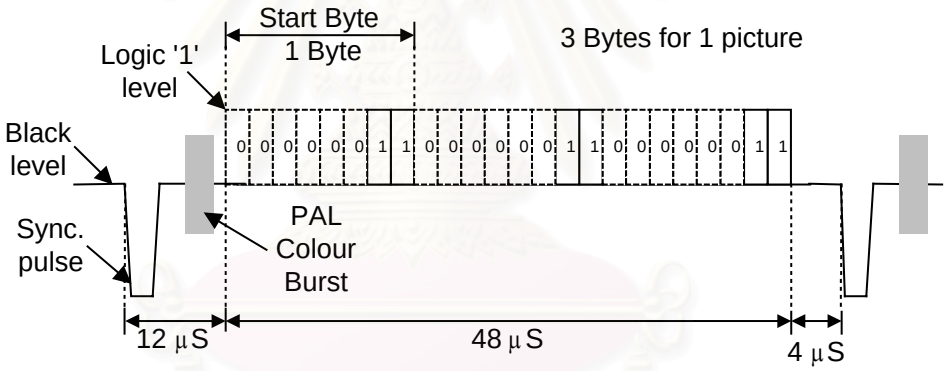
รูปที่ 3.6 รูปแบบของข้อมูลที่แทรกในช่วงไร้ภาพแนวตั้ง

รายละเอียดของข้อมูลชนิดต่างๆ มีดังนี้

ก. ไบต์เริ่มต้น

ไบต์เริ่มต้นเป็นส่วนสำคัญที่บอกให้เครื่องรับทราบว่ามีข้อมูลที่ถูกบีบอัดชุดใหม่เข้ามา เพื่อให้เครื่องรับเตรียมค่ารีจิสเตอร์ (Register) ที่ใช้คลายข้อมูล ข้อมูลจะเริ่มเก็บต่อจากไบต์เริ่มต้นและนำไปประมวลผลในกระบวนการคลายข้อมูล เช่น ไบต์ควบคุม และไบต์ขนาด ซึ่งรายละเอียดของข้อมูลแต่ละประเภทจะได้กล่าวในลำดับต่อไป

ข้อมูลไบต์เริ่มต้นได้พัฒนาขึ้นมาจากรหัสเพิ่มเติมของการบีบอัดข้อมูลตามมาตรฐาน CCITT T.4 แบบ 2 มิติ คือ "0000001xxx" ให้เป็น "00000011" คำรหัสที่ได้จะไม่ซ้ำกับคำรหัสที่ได้จากการบีบอัดข้อมูลในโมดต่างๆ การแทรกไบต์เริ่มต้นจะแทรกข้อมูลลงในช่วงไร้ภาพแนวตั้งของเส้นสัญญาณวิทัศน์จำนวน 3 ไบต์ติดกัน และให้อยู่ในเส้นสัญญาณวีดิทัศน์ 1 เส้น ดังแสดงในรูปที่ 3.7

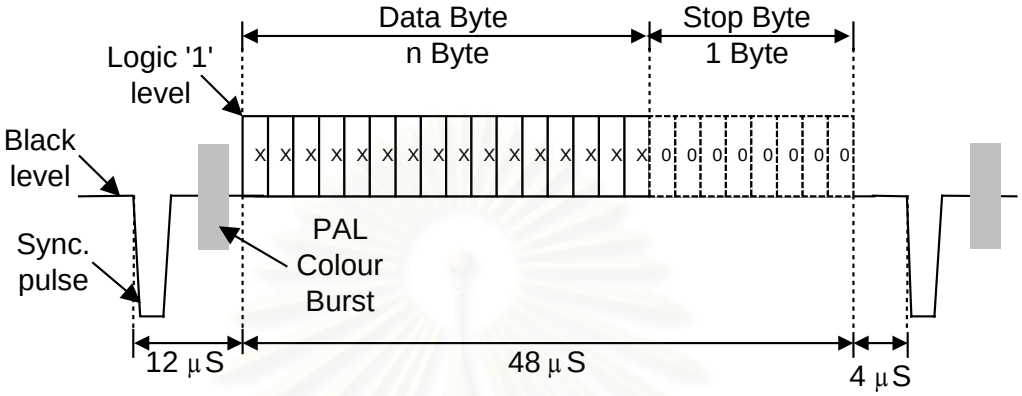


รูปที่ 3.7 ไบต์เริ่มต้นเมื่อแทรกลงในสัญญาณวีดิทัศน์

ข. ไบต์สิ้นสุดข้อมูล

ไบต์สิ้นสุดข้อมูลจะบอกเครื่องรับให้ทราบว่าสิ้นสุดการส่งข้อมูลของภาพบรรยายภาพนั้น เมื่อเครื่องรับข้อมูลภาพได้รับไบต์สิ้นสุดข้อมูลจะตรวจสอบขนาดข้อมูลที่ได้รับ ถ้าข้อมูลที่ได้ไม่ครบถ้วนเครื่องรับจะไม่แสดงผลภาพที่ได้รับมา และจะรับข้อมูลภาพชุดต่อไปมาทับลงบนข้อมูลภาพชุดเดิมที่ไม่สมบูรณ์ ไบต์สิ้นสุดพัฒนาจากรหัสเพิ่มเติมของการบีบอัดข้อมูลตามมาตรฐาน CCITT T.4 แบบ 2 มิติเช่นกัน คำรหัสที่ได้จะมีขนาด 1 ไบต์ซึ่งลักษณะคำรหัสที่ได้จะไม่ซ้ำกับคำรหัสที่ได้จากากรบีบอัดข้อมูลในโมดต่างๆ

ลักษณะคำรหัสของไบต์สิ้นสุดข้อมูลคือ "00000000" การแทรกไบต์สิ้นสุดข้อมูลจะแทรกต่อท้ายจากข้อมูลภาพที่ถูกบีบอัด ดังแสดงในรูปที่ 3.8

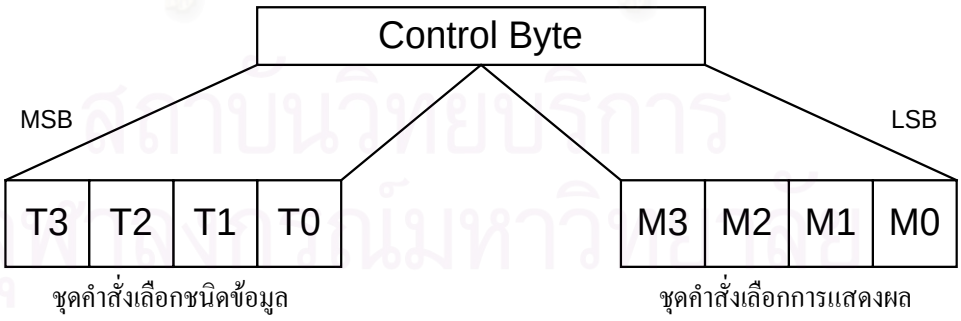


รูปที่ 3.8 ไบต์สิ้นสุดในเส้นสัญญาณวีดิทัศน์

ค. ไบต์ควบคุม

ไบต์ควบคุมเป็นข้อมูลสำคัญที่ใช้ควบคุมการทำงานต่างๆ ของเครื่องรับข้อมูลภาพ เช่น การแสดงผลบนจอภาพ และชนิดของข้อมูลเป็นต้น การควบคุมการทำงานเหล่านี้สามารถเลือกได้จากโปรแกรมที่ทำงานบนเครื่องคอมพิวเตอร์ส่วนบุคคล

ไบต์ควบคุมจะถูกส่งต่อจากไบต์เริ่มต้น และมีขนาด 1 ไบต์ โดยแยกออกเป็นกลุ่มย่อยขนาด 4 บิต 2 กลุ่มเพื่อใช้ในการควบคุมงานที่แตกต่างกัน ดังรูปที่ 3.9 กล่าวคือ



รูปที่ 3.9 โครงสร้างของไบต์ควบคุม

1. กลุ่มที่ใช้เลือกชนิดข้อมูล ได้แก่ บิตที่ 4 ถึง 7 ของไบต์ควบคุม จึงเลือกชนิดข้อมูลได้ทั้งหมด 16 แบบ บิตควบคุมกลุ่มนี้ออกแบบสำหรับการส่งข้อมูลที่ไม่ใช่ข้อมูลที่บีบอัดตามมาตรฐาน CCITT T.4 แบบ 2 มิติ โดยจะต้องส่งด้วยอัตราการส่งข้อมูลเดียวกัน สำหรับค่าของบิตควบคุมที่ใช้เมื่อชนิดข้อมูลที่ส่งเป็นข้อมูลที่บีบอัดตามมาตรฐาน CCITT T.4 แบบ 2 มิติ คือ "0000" สำหรับค่าของข้อมูลชนิดอื่นๆ สามารถกำหนดเพิ่มเติมได้ในภายหลัง

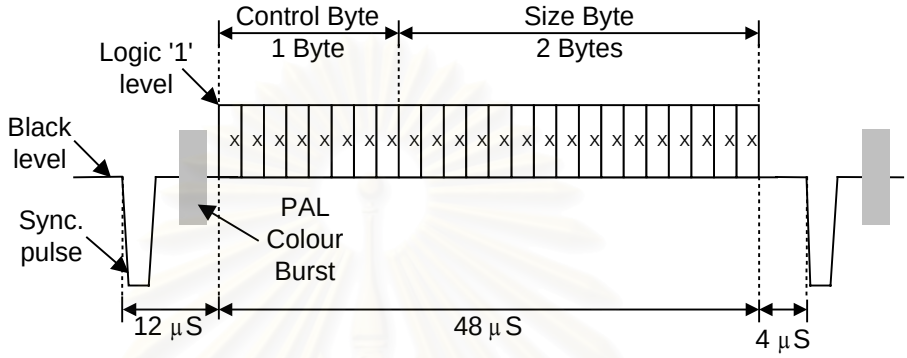
2. กลุ่มที่ใช้เลือกการแสดงผล ได้แก่ บิตที่ 0 ถึง 3 จึงเลือกการแสดงภาพบรรยายบนจอภาพของเครื่องรับได้ 16 แบบ แต่ในโครงงานนี้มีเพียง 6 แบบ ดังตารางที่ 3.1

ค่ารหัส	ผลการแสดงภาพ
0000	แสดงภาพใหม่ทันทีเมื่อคลายข้อมูลภาพเสร็จ
0001	ยังไม่มีการแสดงภาพใหม่เมื่อมีการคลายข้อมูลภาพใหม่เสร็จ
0010	แสดงภาพเก่าที่ยังไม่ได้แสดงและเก็บภาพใหม่ไว้ในหน่วยความจำ
0011	แสดงภาพเก่าโดยที่ไม่มีการส่งข้อมูลภาพใหม่
0100	ซ่อนภาพบรรยายที่กำลังแสดงอยู่
0101	เปิดภาพบรรยายที่ถูกปิดด้วยคำสั่งซ่อนภาพ
0110…1111	ออกแบบสำหรับการแสดงผลแบบอื่นๆ

ตารางที่ 3.1 รหัสเลือกการแสดงผลแบบต่างๆ

ง. ไบต์ขนาด

ไบต์ขนาดจะบอกจำนวนบิตของข้อมูลภาพที่ถูกบีบอัดแล้ว ไบต์ขนาดจะส่งต่อจากไบต์ควบคุมซึ่งจะอยู่ในเส้นสัญญาณวีดิทัศน์เดียวกัน ดังแสดงในรูปที่ 3.10



รูปที่ 3.10 ไบต์ควบคุมและไบต์ขนาดบนสัญญาณวีดิทัศน์เส้นเดียวกัน

ขนาดข้อมูลภาพที่บีบอัด จะเก็บด้วยไบต์ขนาดจำนวน 2 ไบต์ ดังนี้

ขนาดข้อมูล 315 ไบต์ มีไบต์ขนาด คือ 0000000100111011

แบ่งออกเป็นไบต์แรก 00000001 และ ไบต์ที่สอง 00111011

นอกจากนี้ในกรณีที่มีการส่งเพียงคำสั่งให้แสดงภาพเพียงอย่างเดียว ไบต์ขนาดจะมีค่าเป็น " 00000000 00000000 "

จ. ไบต์ข้อมูล

ไบต์ข้อมูลเป็นชุดของคำรหัสที่มาจากการบีบอัดข้อมูลตามมาตรฐาน CCITT T.4 แบบ 2 มิติ การส่งข้อมูลจะนำข้อมูลที่ถูกบีบอัดมาเรียงต่อกัน และตัดข้อมูลออกมาครั้งละ 8 บิต และหากข้อมูลที่เหลือไม่ครบ 8 บิต จะนำ 0 มาเติมให้ครบ 8 บิตแล้วจึงทำการส่ง เช่น

ข้อมูลที่เหลือคือ 0110 จะนำ 0000 มาเติมให้ครบ 8 บิตได้ 0110**0000** เป็นส่วนเติมแล้วจึงส่งข้อมูลออกไป

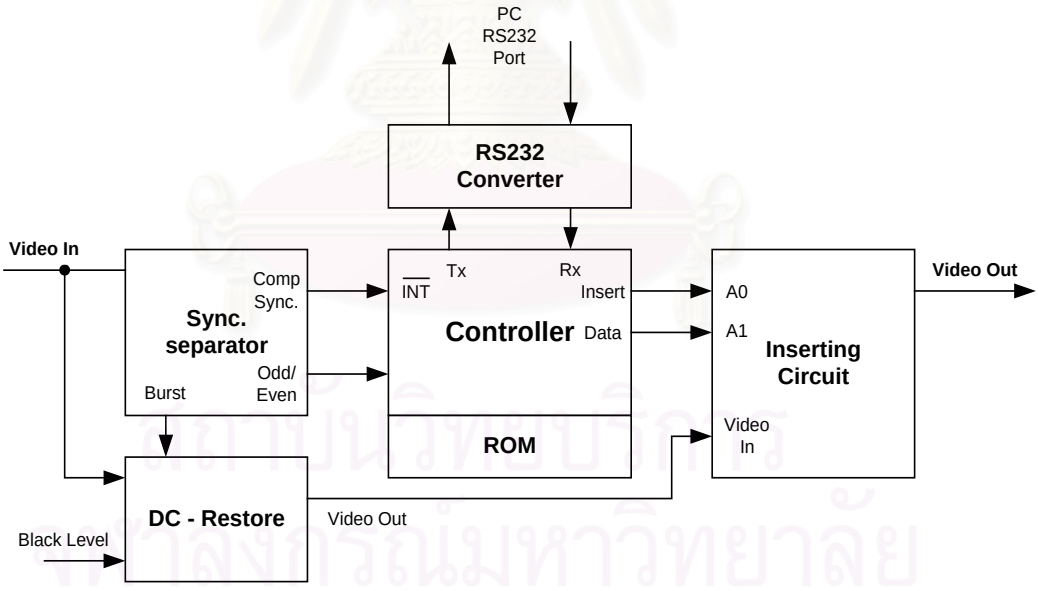
บทที่ 4

โครงสร้างและการทำงานของเครื่องแทรกข้อมูลภาพ

ดังที่ได้กล่าวมาในบทที่ 3 เกี่ยวกับการทำงานของระบบด้านส่งของระบบวีดีโอที่ซ่อนภาพบรรยายได้ เครื่องแทรกข้อมูลภาพและคอมพิวเตอร์ส่วนบุคคลเป็นส่วนประกอบที่สำคัญของระบบด้านส่ง ในบทนี้จะกล่าวถึงโครงสร้างและการทำงานโดยละเอียดของเครื่องแทรกข้อมูลภาพ

4.1 โครงสร้างภายในของเครื่องแทรกข้อมูลภาพ

เครื่องแทรกข้อมูลภาพมีโครงสร้างภายใน ดังแสดงในรูปที่ 4.1



รูปที่ 4.1 โครงสร้างภายในของเครื่องแทรกข้อมูลภาพ

เครื่องแทรกข้อมูลภาพประกอบด้วยส่วนสำคัญคือ

1. ตัวควบคุม (Controller) มีหน้าที่ควบคุมการทำงานของเครื่องแทรกข้อมูลภาพทั้งหมด เช่น การแทรกข้อมูลภาพ การติดต่อกับเครื่องคอมพิวเตอร์ส่วนบุคคล เป็นต้น
2. วงจรแยกซิงก์ (Sync. Separator) มีหน้าที่แยกสัญญาณที่ใช้ควบคุมการทำงานจากสัญญาณวีดิทัศน์ส่งให้กับตัวควบคุม
3. วงจรรักษาระดับแรงดัน (DC-Restore) มีหน้าที่รักษาแรงดันของระดับสีดำในสัญญาณวีดิทัศน์ให้อยู่ที่ค่าที่ต้องการ
4. วงจรแทรกข้อมูลภาพ (Inserting Circuit) มีหน้าที่แทรกข้อมูลภาพลงในช่วงไร้ภาพแนวตั้งของสัญญาณวีดิทัศน์
5. วงจรติดต่อพอร์ตอนุกรม (RS232 Converter) มีหน้าที่เป็นตัวกลางสำหรับการติดต่อระหว่างเครื่องแทรกข้อมูลภาพและคอมพิวเตอร์ส่วนบุคคล

การทำงานของเครื่องแทรกข้อมูลภาพเริ่มจากรับสัญญาณวีดิทัศน์จากแหล่งกำเนิดสัญญาณวีดิทัศน์ที่ต้องการแทรกภาพบรรยาย สัญญาณวีดิทัศน์จะถูกแบ่งออกเป็น 2 ส่วน คือ ส่วนที่ต่อเข้ากับวงจรแยกซิงก์และส่วนที่ต่อเข้ากับวงจรรักษาระดับแรงดัน วงจรแยกซิงก์จะแยกสัญญาณซิงก์รวม (Composite Sync.) และสัญญาณบอกฟิลด์ (Odd/Even) ส่งให้กับตัวควบคุมเพื่อใช้ประมวลผล ตัวควบคุมติดต่อกับคอมพิวเตอร์ส่วนบุคคลผ่านทางพอร์ตอนุกรมเพื่อรับชุดข้อมูลภาพที่ถูกบีบอัดจากเครื่องคอมพิวเตอร์ส่วนบุคคล ข้อมูลจะถูกเก็บไว้ที่หน่วยความจำภายในตัวควบคุม แล้วจึงแทรกข้อมูลลงในช่วงไร้ภาพแนวตั้งของสัญญาณด้วยวงจรแทรกข้อมูลภาพ

4.2 ตัวควบคุม

ตัวควบคุมที่ใช้เป็นไมโครคอนโทรลเลอร์ตระกูล 8051 เบอร์ DS80C320 [7] ซึ่งมีความเร็วในการทำงานสูงกว่าไมโครคอนโทรลเลอร์ตระกูลเดียวกันประมาณ 3 เท่า หน่วยความจำภายในมีขนาด 128 ไบต์ และสามารถทำงานได้ที่สัญญาณนาฬิกาสูงสุด 33 MHz แต่ในเครื่องแทรกข้อมูลภาพจะทำงานที่ความเร็วสัญญาณนาฬิกาเพียง 24 MHz ไมโครคอนโทรลเลอร์เบอร์นี้สามารถ

เลือกส่งข้อมูลทางพอร์ตอนุกรมได้ 2 พอร์ต คือ ผ่านทาง TXD0, RXD0 กับ TXD1, RXD1 พอร์ตอนุกรม TXD0, RXD0 จะใช้ตัวสร้างอัตราการส่งเป็นตั้งเวลาตัวที่ 1 (Timer 1) หรือตัวตั้งเวลาที่ 2 (Timer2) ก็ได้ ในขณะที่ TXD1,RXD1 สามารถใช้เพียงตัวตั้งเวลาตัวที่ 1 เป็นตัวสร้างอัตราการส่งเท่านั้น

อัตราการส่งข้อมูลที่เครื่องแทรกข้อมูลภาพรับจากคอมพิวเตอร์ส่วนบุคคลนั้นคิดจาก การส่งข้อมูล 48 ไบต์ โดยแต่ละไบต์จะต้องใช้ข้อมูลทั้งหมด 10 บิต คือ รวมบิตเริ่มต้น (Start Bit) และบิตหยุด (Stop Bit) ดังนั้นในการส่งข้อมูล 1 ชุดจะมีข้อมูลทั้งหมด 480 บิต ตัวควบคุมจะต้องเก็บข้อมูลให้ได้ภายใน 1 ฟิลด์เพื่อนำไปแทรกลงในช่วงไร้ภาพแนวตั้งของเส้นสัญญาณวีดิทัศน์ในฟิลด์ถัดไป ตัวควบคุมจะสามารถรับข้อมูลได้ตั้งแต่เส้นสัญญาณวีดิทัศน์เส้นที่ 24 ถึงเส้นที่ 312 ซึ่งคิดเป็นเวลา 18.496 ms ดังนั้นอัตราการส่งข้อมูลต่ำสุดที่จะต้องใช้คือ

480 บิต / (18.496 ms) = 25.95 kBit/sec

แต่เครื่องคอมพิวเตอร์สามารถส่งข้อมูลผ่านทางพอร์ตอนุกรมได้ด้วยอัตราการส่งข้อมูลค่าต่างๆ คือ 28800, 38400, 56000 [8] และไมโครคอนโทรลเลอร์สามารถตั้งค่าอัตราส่งที่ความถี่สัญญาณนาฬิกา 24 MHz ได้ดังนี้

ตัวตั้งเวลาตัวที่ 1 สามารถตั้งอัตราส่งที่ใกล้เคียง 28800 ได้ดังนี้คือ 25000 และ 31250

ตัวตั้งเวลาตัวที่ 2 สามารถตั้งอัตราส่งที่ใกล้เคียง 28800 ได้ดังนี้คือ 27777 และ 28846

ดังนั้นค่าที่ใกล้เคียงมากที่สุดคือ 28846 ซึ่งอัตราการส่งข้อมูลนี้สามารถส่งข้อมูลระหว่างเครื่องคอมพิวเตอร์ส่วนบุคคลและไมโครคอนโทรลเลอร์ได้ถูกต้อง ดังการพิสูจน์ต่อไปนี้

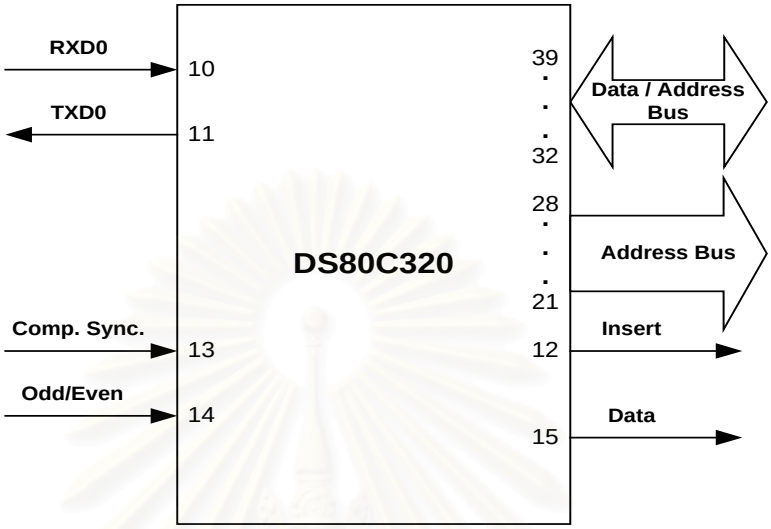
อัตราการส่งข้อมูล 28800 bit/sec ใช้เวลาส่งข้อมูล 1 บิต 34.72 μS

อัตราการส่งข้อมูล 28846 bit/sec ใช้เวลาส่งข้อมูล 1 บิต 34.67 μS

ส่วนต่างของเวลาที่ส่งข้อมูล 1 บิต คือ 0.05 μS การสุ่มข้อมูล 10 บิต จะทำให้เวลาที่สุ่มของบิตสุดท้ายเร็วขึ้น 0.5 μS ซึ่งยังอยู่ในช่วงของบิตสุดท้าย จึงทำให้ได้ข้อมูลบิตสุดท้ายถูกต้อง ดังนั้น การส่งข้อมูลด้วยอัตราข้อมูล 28800 kbit/sec จะสุ่มด้วยอัตราข้อมูล 28846 kbit/secc ได้

ดังนี้น อัตราการส่งข้อมูลที่ต้องการคือ 28846 kbit/sec จึงต้องใช้ตัวตั้งเวลาที่ 2 เพราะสามารถสร้างค่าอัตราการส่งได้ใกล้เคียงกับอัตราการส่งได้มากที่สุด พอร์ตอนุกรมที่ใช้ส่งข้อมูลคือ TXD0, RXD0

ลักษณะการต่อสัญญาณที่สำคัญต่างๆเข้าที่ไมโครคอนโทรลเลอร์แสดงดังรูปที่ 4.2

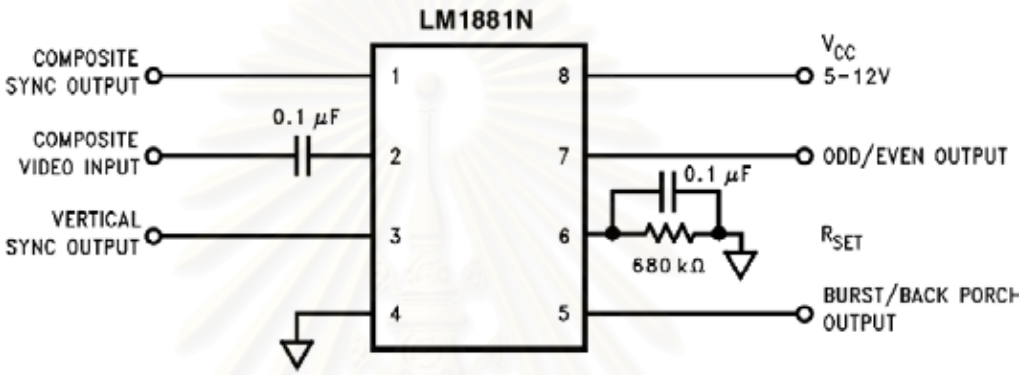


รูปที่ 4.2 การต่อสัญญาณของไมโครคอนโทรลเลอร์

ไมโครคอนโทรลเลอร์จะใช้ขาสัญญาณ TXD0 กับ RXD0 สำหรับติดต่อกับเครื่องคอมพิวเตอร์ส่วนบุคคลผ่านทางพอร์ตอนุกรม การแทรกข้อมูลเริ่มเมื่อสัญญาณวีดิทัศน์ถูกส่งเข้ามาที่วงจรแยกซิงก์ ไมโครคอนโทรลเลอร์ได้รับสัญญาณซิงก์รวมเข้ามาที่ขาสัญญาณ Comp. Sync. ซึ่งเป็นขาที่ทำให้เกิดการขัดจังหวะ (Interrupt) ในขณะที่สัญญาณบอกฟิลด์จะเข้ามาที่ขาสัญญาณ Odd/Even ตัวควบคุมจะนำสัญญาณทั้ง 2 ชนิดไปนับเส้นสัญญาณวีดิทัศน์เพื่อเข้าไปทำงานในกระบวนการต่างๆ ซึ่งจะกล่าวโดยละเอียดอีกครั้งในการอธิบายทำงานของโปรแกรม ขาสัญญาณ Insert และ Data จะใช้แทรกข้อมูลที่ถูกบีบอัดลงในช่วงไร้ภาพแนวตั้งในเส้นสัญญาณวีดิทัศน์ โดยสัญญาณดังกล่าวจะส่งไปให้วงจรแทรกข้อมูลภาพ โดยสัญญาณ Insert ใช้บอกการเริ่มต้นการแทรกข้อมูล สัญญาณ Data คือสัญญาณของข้อมูลที่ต้องการแทรกซึ่งจะส่งออกมาเป็นข้อมูลที่มีอัตราส่ง 500 kbit/sec สำหรับขาสัญญาณ Data Bus ใช้ในการรับส่งข้อโปรแกรมควบคุมการทำงานจาก ROM

4.3 วงจรแยกซิงก์

วงจรแยกซิงก์มีหน้าที่ดึงสัญญาณซิงก์รวมและสัญญาณบอกฟิลด์ออกจากสัญญาณวีดีโอรวม ภายในวงจรแยกซิงก์ประกอบด้วยวงจรรวมเบอร์ LM1881 หรือ EL4581 ซึ่งมีการต่อวงจร ดังแสดงในรูปที่ 4.3

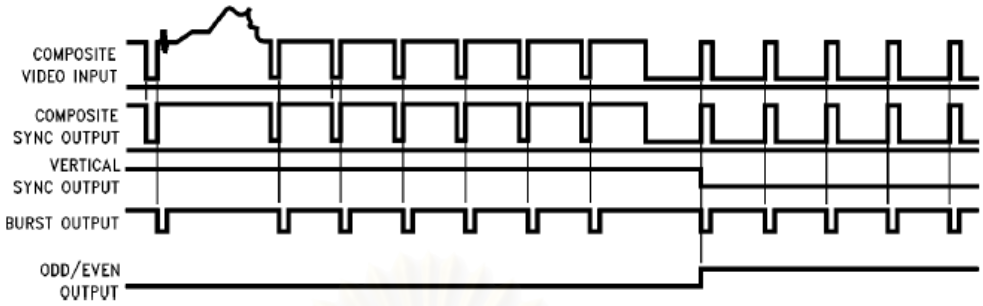


รูปที่ 4.3 วงจรแยกซิงก์

สัญญาณวีดีทัศน์จะผ่านตัวเก็บประจุขนาด 0.1 μF ก่อนแล้วจึงผ่านเข้าไปที่วงจรรวมเบอร์ LM1881 (Video Sync. Separator) สัญญาณที่ได้ออกมาจากวงจรรวมมี 4 สัญญาณคือ

1. สัญญาณซิงก์รวม
2. สัญญาณซิงก์แนวตั้ง
3. สัญญาณบอกฟิลด์
4. สัญญาณเบิรสต์และไหล่หลัง (Burst/Back Porch)

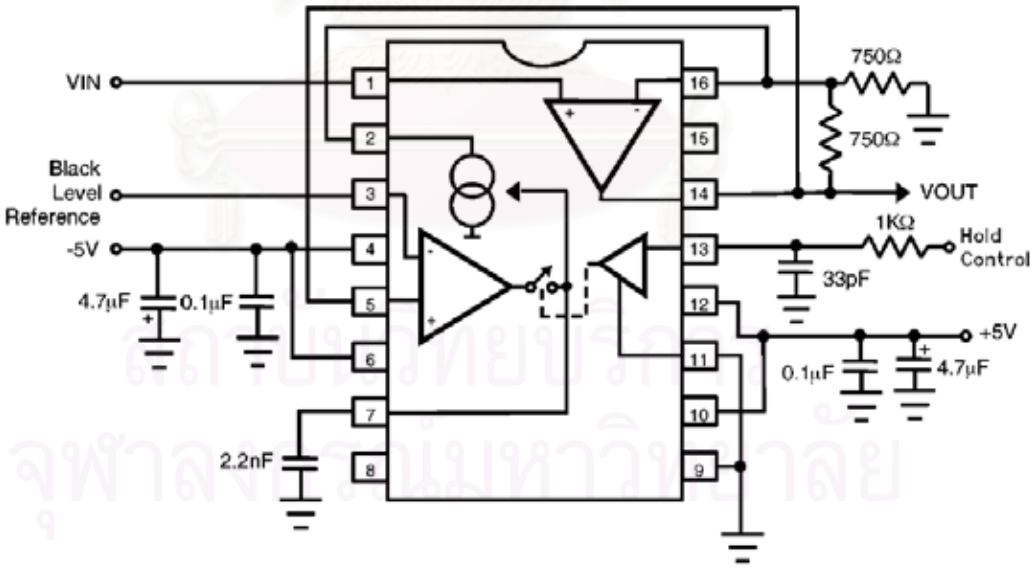
รูปสัญญาณต่างๆ แสดงดังรูปที่ 4.4 โดยสัญญาณที่ใช้ควบคุมการทำงานของไมโครคอนโทรลเลอร์คือ สัญญาณซิงก์รวมซึ่งใช้เป็นสัญญาณนับเส้นสัญญาณวีดิทัศน์ สัญญาณบอกฟิลด์ซึ่งใช้บอกว่าขณะนั้นเป็นฟิลด์คู่หรือฟิลด์คี่ ซึ่งสัญญาณทั้ง 2 สัญญาณนี้จะต่อเข้ากับขาสัญญาณ Comp. Sync. และ Odd/Even ตามลำดับ สำหรับสัญญาณเบิรสต์จะต่อเข้ากับวงจรรักษาระดับแรงดันเพื่อใช้รักษาระดับแรงดันของระดับสีดำในสัญญาณวีดิทัศน์



รูปที่ 4.4 สัญญาณจาก LM1881

4.4 วงจรรักษาระดับแรงดัน

วงจรรักษาระดับแรงดันมีหน้าที่รักษาระดับแรงดันที่ต้องการให้มีค่าคงที่ค่าหนึ่ง ซึ่งตั้งเป็นระดับอ้างอิงไว้ ภายในวงจรประกอบด้วยวงจรรวมเบอร์ EL4093C (300 MHz DC-Restored Video Amplifier) มีการต่อใช้งาน ดังแสดงในรูปที่ 4.5



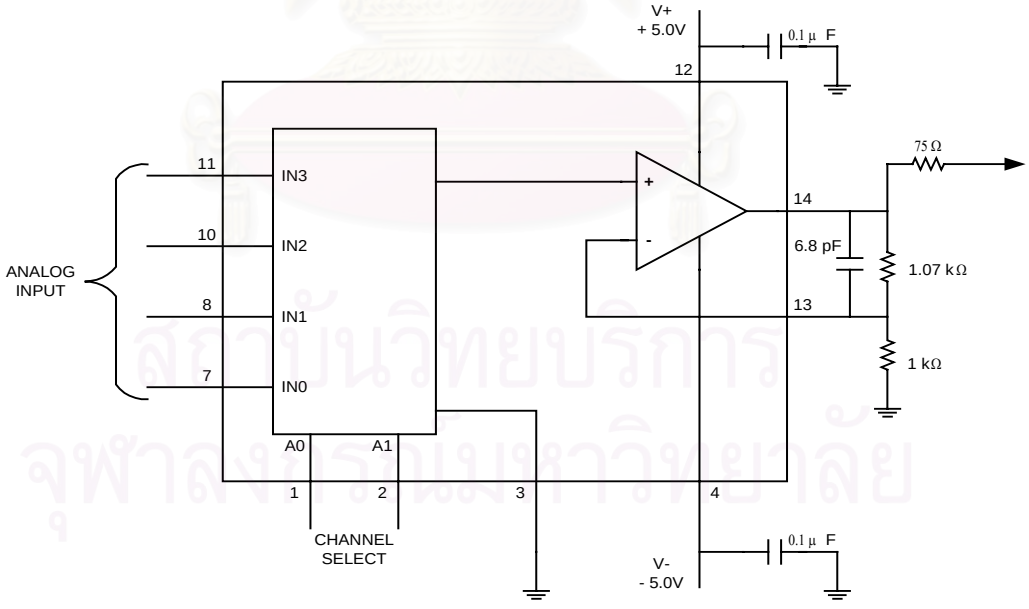
รูปที่ 4.5 วงจรรักษาระดับแรงดัน

สัญญาณวีดิทัศน์จะส่งเข้ามาที่ขาสัญญาณ VIN (ขาที่ 1) และสัญญาณจะออกที่ขาสัญญาณ VOUT (ขาที่ 14) โดยจะมีขนาดสัญญาณเป็น 2 เท่าของสัญญาณขาเข้า ระดับอ้างอิง

จะต่อเข้าที่ขาสัญญาณ Black Level (ขาที่ 3) ซึ่งระดับอ้างอิงนี้คือ ระดับสีดำของสัญญาณวีดิทัศน์ที่ต้องการให้คงค่าไว้ การรักษาระดับแรงดันจะต้องให้ระดับสัญญาณ '1' ที่ขาสัญญาณ Hold Control (ขาที่ 13) ดังนั้นการรักษาระดับสีดำของสัญญาณวีดิทัศน์จึงใช้สัญญาณเบิรสต์จากวงจรแยกซิงก์ ซึ่งมีลักษณะสัญญาณดังแสดงในรูปที่ 4.4 สัญญาณขาออกจะถูกนำมาเปรียบเทียบกับระดับแรงดันอ้างอิงและส่งสัญญาณออกที่ขาที่ 2 เพื่อไปปรับค่าแรงดันขาออกผ่านทางตัวต้านทานค่า 750 โอห์มทั้ง 2 ตัว แต่การปรับระดับสัญญาณจะสามารถกระทำได้ในช่วงที่สัญญาณ Hold มีค่าอยู่ในลอจิก '1' เท่านั้น นั่นคือ การรักษาระดับแรงดันของระดับสีดำตามที่ต้องการ

4.5 วงจรแทรกข้อมูลภาพ

วงจรแทรกข้อมูลภาพมีหน้าที่นำข้อมูลภาพที่เก็บไว้ที่หน่วยความจำภายในของเครื่องส่งแทรกลงในช่วงไร้ภาพแนวตั้งในเส้นสัญญาณวีดิทัศน์ครั้งละ 3 ไบต์ด้วยอัตราข้อมูล 500 kBit/sec ภายในวงจรประกอบด้วยวงจรรวมเบอร์ MAX 454 (CMOS Video Multiplexer / Amplifier) มีการต่อวงจร ดังแสดงในรูปที่ 4.6



รูปที่ 4.6 วงจรรวมเบอร์ MAX454

วงจรรวมนี้เป็นตัวมัลติเพลกซ์ (Multiplexer) แบบเข้า 4 ออก 1 ในการเลือกช่องสัญญาณสามารถเลือกได้ด้วยขาสัญญาณ A0 และ A1 ดังตารางที่ 4.1

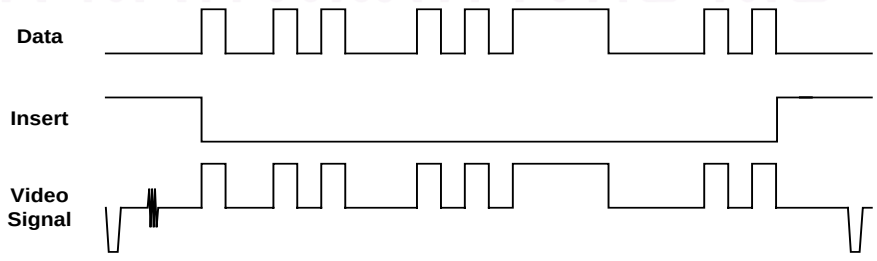
A0	A1	ช่องสัญญาณ
L	L	0
L	H	1
H	L	2
H	H	3

ตารางที่ 4.1 การเลือกช่องสัญญาณ

ขาสัญญาณ A0 จะต่อเข้ากับสัญญาณ Insert จากตัวควบคุม เพื่อเลือกแทรกข้อมูลภาพ ส่วนสัญญาณ A1 จะต่อเข้ากับสัญญาณ Data จากตัวควบคุม ซึ่งบอกว่าข้อมูลที่แทรกเป็นลอจิก '0' หรือ ลอจิก '1' สำหรับช่องสัญญาณทั้ง 4 ช่องจะมีการต่อกับสัญญาณต่างๆดังนี้

1. ช่องสัญญาณ 0 ต่อกับสัญญาณระดับสีดำที่เป็นระดับอ้างอิงของวงจรรักษาระดับแรงดันเพื่อใช้เป็นลอจิก '0'
2. ช่องสัญญาณ 1 ต่อกับระดับแรงดัน 1.8 V. เพื่อใช้เป็นลอจิก '1' ของข้อมูล
3. ช่องสัญญาณ 3 และ 4 จะต่อกับสัญญาณวีดิทัศน์ที่ได้จากวงจรรักษาระดับแรงดันที่มีระดับสีดำ 0 V.

การแทรกข้อมูลเริ่มเมื่อตัวควบคุมส่งสัญญาณ Insert เป็นลอจิก '0' ทำให้ MAX454 เลือกสัญญาณที่จะส่งจากช่องสัญญาณ 0 และ 1 เท่านั้น เมื่อสัญญาณ Inst เป็นลอจิก '0' ตัวควบคุมจะส่งข้อมูลที่ต้องการแทรกด้วยอัตราข้อมูล 500 kBit/sec มาที่ขาสัญญาณ A1 ดังแสดงในรูปที่ 4.7



รูปที่ 4.7 การแทรกข้อมูลลงในช่วงไร้ภาพแนวตั้ง

4.6 วงจรติดต่อพอร์ตอนุกรม

วงจรติดต่อพอร์ตมีหน้าที่เป็นตัวกลางสำหรับติดต่อทางพอร์ตอนุกรม (RS232) ระหว่างเครื่องแทรกข้อมูลภาพและคอมพิวเตอร์ส่วนบุคคล เนื่องจากระดับแรงดันลอจิกของพอร์ตอนุกรมของเครื่องคอมพิวเตอร์ส่วนบุคคลกับระดับแรงดันลอจิกของเครื่องแทรกข้อมุลภาพมีค่าไม่เท่ากัน

4.7 โปรแกรมการทำงานของเครื่องแทรกข้อมูลภาพ

เครื่องแทรกข้อมูลภาพจะทำงานตามโปรแกรมที่เก็บไว้ในหน่วยความจำ โดยตัวควบคุมจะเป็นตัวทำงานตามโปรแกรมคำสั่งที่เขียนไว้ โปรแกรมนี้พัฒนาขึ้นจากภาษาแอสเซมบลี (Assembly) โดยมีขอบเขตการทำงานของโปรแกรม คือ

1. สามารถส่งติดต่อรับ-ส่งกับคอมพิวเตอร์ส่วนบุคคลได้
2. สามารถแทรกข้อมูลลงในช่วงไร้ภาพแนวตั้งของสัญญาณวีดิทัศน์ได้

การทำงานของโปรแกรมจะอธิบายเป็นภาษาบรรยายโปรแกรม (Program Description Language: PDL) ซึ่งสามารถอธิบายการทำงานได้ง่ายกว่า ภาษาบรรยายโปรแกรมแบ่งออกเป็น 3 ระดับ ในแต่ละระดับจะมีรายละเอียดของโปรแกรมเพิ่มขึ้นตามลำดับ

1. ภาษาบรรยายโปรแกรมระดับที่ 1 (Program Description Language 1: PDL1)

```
Module: Main
 Initialization
 Repeat
      Wait for update new line
      Case (N) * is                    ;* N = Number of video scan line
            When 6<N<23     => Insert data in Vertical Blanking Interval
            When N = 23     => Send request to personal computer
            When N = 24     => Receiver data from personal computer
                               and wait new field when end 48 bytes
                               of data
```

When others => No operation

Until Forever

END;

โปรแกรมเริ่มทำงานงานจากการกำหนดเริ่มต้น (Initial) จากนั้นจะนับเส้นสัญญาณจนถึงเส้นที่ 23 เพื่อส่งสัญญาณร้องขอข้อมูล (Send Request) และรับข้อมูลทางพอร์ตอนุกรมในช่วงเส้นสัญญาณที่ 24 ถึง 312 มาเก็บไว้ที่หน่วยความจำภายใน และรอขึ้นฟิลด์ใหม่ เพื่อแทรกข้อมูลภาพลงในช่วงไร้ภาพแนวตั้งของสัญญาณวีดิทัศน์เส้นที่ 7 ถึง 22 และทำวนไปเรื่อยๆ

ภาษาบรรยายโปรแกรมระดับที่ 2 จะกล่าวถึงรายละเอียดของโปรแกรมเกี่ยวกับการตั้งค่าเริ่มต้นและการทำงานอื่นๆ เพิ่มเติม

2. ภาษาบรรยายโปรแกรมระดับที่ 2 (Program Description Language 2: PDL2)

Module: Main

Initialization

Clear internal memory

Initialize serial port

Initialize interrupt

Initialize register

Repeat

Wait for update new line

Increase line number

Case (N) is

When 6<N<23 =>

Insert data in Vertical Blanking Interval

Get data from internal memory

Insert data 3 bytes in Vertical Blankig Interval of scan line

```
            When N = 23        =>
                Send request to personal computer
                        Send character 'R' to personal computer
            When N = 24        =>
                Receiver data from personal computer
                Wait new field when end 48 bytes of data
            When others        =>
                No operation
  Until Forever
END;
```

3. ภาษาบรรยายโปรแกรมระดับที่ 3 (Program Description Language 3: PDL3)

```
Module: Main
  Initialization
      Clear internal memory
        Write 00h to internal memory
      Initialize serial port
        Select serial port and buad rate generator
        Set baud rate from Timer2
        Clear serial buffer
      Initialize interrupt
        Enable interrupt
        Set active for Interrupt1and Serial Interrupt
      Initialize register
        Set data pointer to start address
```

```
Repeat
    Wait for update new line
        Repeat
            No operation
        Until detect new line
        Increase line number
    Case (N) is
        When 6<N<23     =>
            Insert data in Vertical Blanking Interval
                Repeat
                    Get data from internal memory
                        Move data from internal memory to accumulator
                    Insert data 3 bytes in Vertical Blankig Interval of scan line
                        Repeat
                            Shift data at rate 500 kbit/sec
                            Shift data bit to dataport
                        Until end 8 bits of data
                Until end 3 bytes of data

        When N = 23       =>
            Send request to personal computer
                    Send character 'R' to personal computer
                        Move 'R' to serial buffer
                        Clear transmitted flag in SCON register
```

```
When N = 24        =>
    Set data pointer to start address
    Repeat
        Receiver data from personal computer
            Move data from serial buffer to accumulator
            Move data from accumulator to internal RAM
            Increase data pointer
        Wait new field when end 48 bytes of data
            Set line number = 0
            Set data pointer to start address
    Until end of data
When others        =>
    No operation
 Until Forever
END;
```

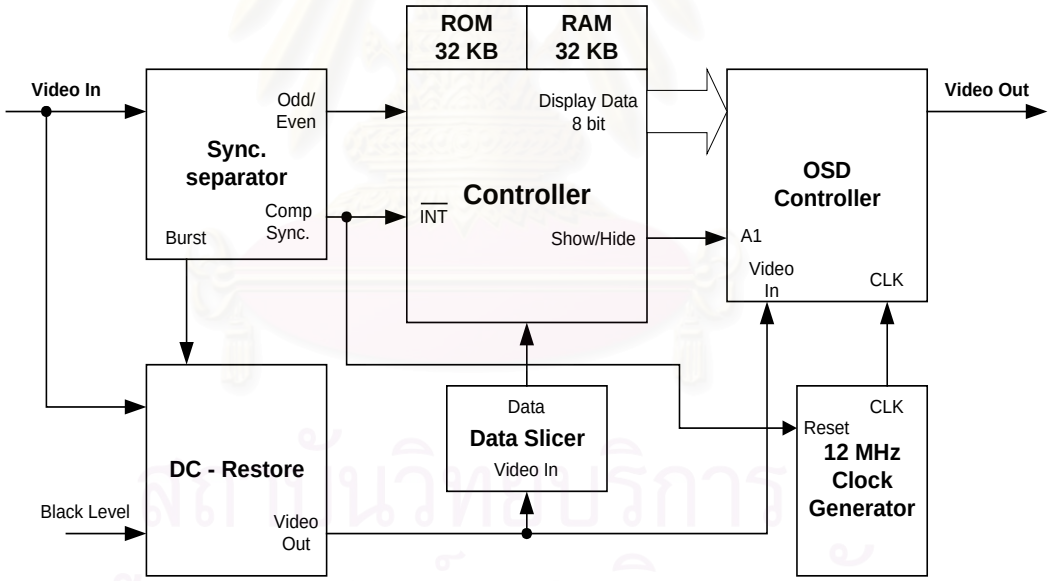
บทที่ 5

โครงสร้างและการทำงานของเครื่องรับข้อมูลภาพ

เครื่องรับข้อมูลภาพเป็นส่วนประกอบหลักของระบบด้านรับ โดยมีหน้าที่คลายข้อมูลภาพที่ถูกบีบอัด และแสดงผลภาพบรรยายบนจอโทรทัศน์ ในบทนี้จะกล่าวถึงโครงสร้างและการทำงานโดยละเอียดของเครื่องรับข้อมูลภาพ

5.1 โครงสร้างภายในของเครื่องรับข้อมูลภาพ

เครื่องรับข้อมูลภาพมีโครงสร้างบางส่วนที่คล้ายกับเครื่องแทรกข้อมูลภาพ โดยเครื่องรับข้อมูลภาพมีโครงสร้างดังรูปที่ 5.1



รูปที่ 5.1 โครงสร้างภายในของเครื่องรับข้อมูลภาพ

เครื่องรับข้อมูลประกอบด้วยส่วนประกอบสำคัญคือ

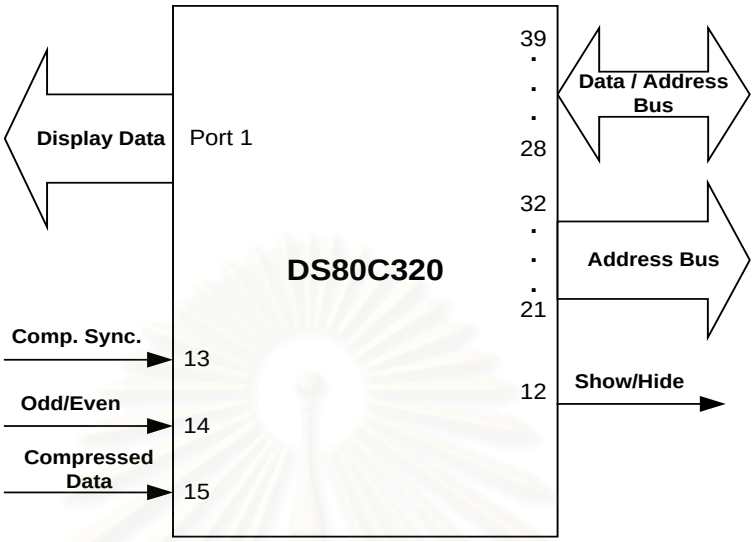
1. ตัวควบคุม (Controller) มีหน้าที่คลายข้อมูลภาพที่ถูกบีบอัด, แสดงผลภาพบนจอโทรทัศน์ และ ควบคุมการทำงานส่วนอื่นๆ

2. วงจรแยกซิงก์ (Sync. Separator) มีหน้าที่แยกสัญญาณที่ใช้ควบคุมการทำงานจากสัญญาณวีดิทัศน์ส่งให้กับตัวควบคุม
3. วงจรรักษาระดับแรงดัน (DC-Restore) มีหน้าที่รักษาแรงดันของระดับสีดำในสัญญาณวีดิทัศน์ให้อยู่ที่ค่าที่ต้องการ
4. วงจรแยกข้อมูลภาพ (Data Slicer) มีหน้าที่ดึงข้อมูลภาพที่ถูกบีบอัดออกจากช่วงไร้ภาพแนวตั้งของเส้นสัญญาณวีดิทัศน์
5. วงจรกำเนิดสัญญาณนาฬิกา (Clock Generator) มีหน้าที่สร้างสัญญาณนาฬิกาให้กับวงแสดงผลบนจอภาพ เพื่อใช้แสดงผลภาพในแนวตั้ง
6. วงจรควบคุมการแสดงผลบนจอภาพ (On screen Display Controller) มีหน้าที่นำข้อมูลภาพที่คลายข้อมูลแล้วมาแสดงบนจอภาพ ที่ความละเอียด 112 X 128 จุด
7. หน่วยความจำ (Memory) หน่วยความจำภายนอกที่ใช้มีขนาด 32 Kbyte ใช้สำหรับเก็บภาพที่ต้องการแสดงบนจอภาพ

เครื่องรับข้อมูลภาพจะแยกสัญญาณวีดิทัศน์ที่ได้รับออกเป็น 2 ส่วน คือ ส่วนที่ต่อไปยังวงจรรักษาระดับแรงดันและส่วนที่ต่อไปยังวงจรแยกซิงก์ โดยวงจรแยกซิงก์จะแยกสัญญาณซิงก์รวมและสัญญาณบอกฟิลด์ส่งให้กับตัวควบคุมเพื่อใช้ควบคุมการทำงาน สำหรับสัญญาณวีดิทัศน์จากวงจรรักษาระดับแรงดันจะแบ่งเป็น 2 ส่วน ส่วนแรกต่อเข้ากับวงจรแยกข้อมูลภาพเพื่อแยกข้อมูลที่ถูกบีบอัดจากช่วงไร้ภาพแนวตั้งของเส้นสัญญาณวีดิทัศน์ส่งให้กับตัวควบคุม อีกส่วนหนึ่งจะต่อเข้ากับวงจรควบคุมการแสดงผลบนจอภาพ ตัวควบคุมจะได้ข้อมูลมาฟิลด์ละ 48 ไบต์ ข้อมูลดังกล่าวจะถูกเก็บไว้ในหน่วยความจำภายในของไมโครคอนโทรลเลอร์เพื่อรอการคลายข้อมูล หลังจากคลายข้อมูลแล้ว ภาพจะถูกเก็บที่หน่วยความจำภายนอก ตัวควบคุมส่งข้อมูลเหล่านี้ครั้งละ 1 ไบต์ ให้กับ วงจรควบคุมการแสดงผลบนจอภาพเพื่อแสดงภาพบนจอโทรทัศน์

5.2 ตัวควบคุม

ตัวควบคุมที่ใช้เป็นไมโครคอนโทรลเลอร์ตระกูล 8051 เบอร์ DS80C320 ทำงานที่ความถี่ของสัญญาณนาฬิกา 24 MHz ซึ่งมีลักษณะการต่อขาสัญญาณที่สำคัญ ดังแสดงในรูปที่ 5.2



รูปที่ 5.2 การต่อสัญญาณของไมโครคอนโทรลเลอร์

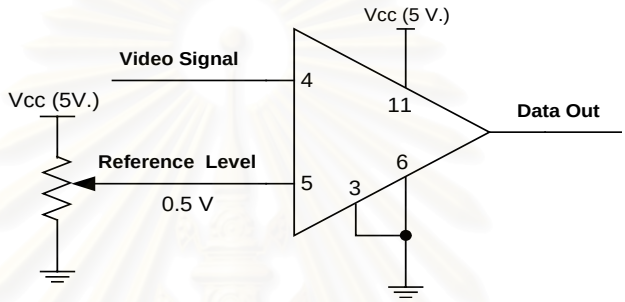
ขาสัญญาณซิงก์รวม (Composite Sync.) และ ขาสัญญาณบอกฟิลด์ (Odd/Even) มีหน้าที่เหมือนในเครื่องแทรกข้อมูลภาพ คือ ใช้นับเส้นสัญญาณวีดิทัศน์เพื่อเข้าไปทำงานในกระบวนการต่างๆ สำหรับขาสัญญาณข้อมูลที่ถูกบีบอัด (Compressed Data) จะรับข้อมูลจากวงจรแยกข้อมูล โดยข้อมูลที่เข้ามาจะมีความถี่ 500 kbit/sec ไมโครคอนโทรลเลอร์จะสุ่มข้อมูลด้วยความถี่เดียวกัน และเก็บไว้ในหน่วยความจำภายใน ข้อมูลที่ได้ในแต่ละฟิลด์จะมีจำนวน 48 ไบต์ เมื่อรับข้อมูลครบแล้วไมโครคอนโทรลเลอร์จะคลายข้อมูลทั้งหมด และ ส่งข้อมูลภาพที่คลายแล้วไปเก็บที่หน่วยความจำภายนอก เมื่อถึงเส้นสัญญาณวีดิทัศน์ที่ต้องการแสดงภาพบนจอโทรทัศน์ ตัวควบคุมจะนำข้อมูลภาพที่เก็บไว้ในหน่วยความจำภายนอกส่งให้วงจรควบคุมการแสดงผลทางขาสัญญาณข้อมูลภาพ (Display Data) จากนั้นตัวควบคุมจะส่งสัญญาณออกที่ขาสัญญาณแสดงภาพ (Show/Hide) ไปยังวงจรควบคุมการแสดงผลบนจอภาพเพื่อแสดงข้อมูลภาพดังกล่าวบนจอโทรทัศน์

5.3 วงจรแยกซิงก์และวงจรรักษาระดับแรงดัน

การทำงานและโครงสร้างของทั้ง 2 วงจรจะเหมือนกับเครื่องแทรกข้อมูลภาพ จึงไม่ขอกล่าวรายละเอียดในบทนี้

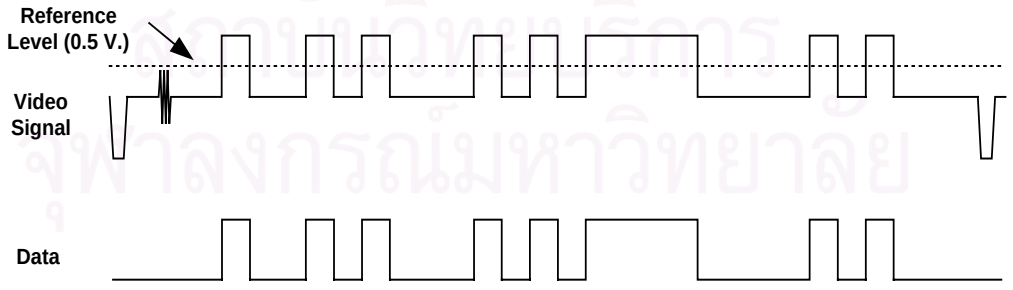
5.4 วงจรแยกข้อมูลภาพ

วงจรแยกข้อมูลภาพมีหน้าที่แยกข้อมูลจากช่วงไร้ภาพแนวตั้งของสัญญาณวีดิทัศน์ที่ได้รับจากวงจรรักษาระดับแรงดัน ภายในวงจรประกอบด้วยวงจรรวมเบอร์ LM319 (High Speed Dual Comparator) ดังแสดงในรูปที่ 5.3



รูปที่ 5.3 วงจรแยกข้อมูลภาพ

สัญญาณวีดิทัศน์จะถูกนำมาเปรียบเทียบกับระดับแรงดันอ้างอิง (ประมาณ 0.5 V.) ถ้าหากสัญญาณวีดิทัศน์มีระดับแรงดันมากกว่าระดับแรงดันอ้างอิง ขาสัญญาณข้อมูลขาออก (Data Out) จะมีค่าเป็นลอจิก '1' แต่ถ้ามีระดับต่ำกว่าจะมีค่าเป็นลอจิก '0' ข้อมูลจะถูกแยกตลอดเวลา ดังแสดงในรูปที่ 5.4 แต่ตัวควบคุมจะเก็บข้อมูลเข้าไปเฉพาะในช่วงไร้ภาพแนวตั้งเท่านั้น เนื่องจากตัวควบคุมจะรู้ว่าที่เส้นสัญญาณวีดิทัศน์เส้นใดที่มีข้อมูล



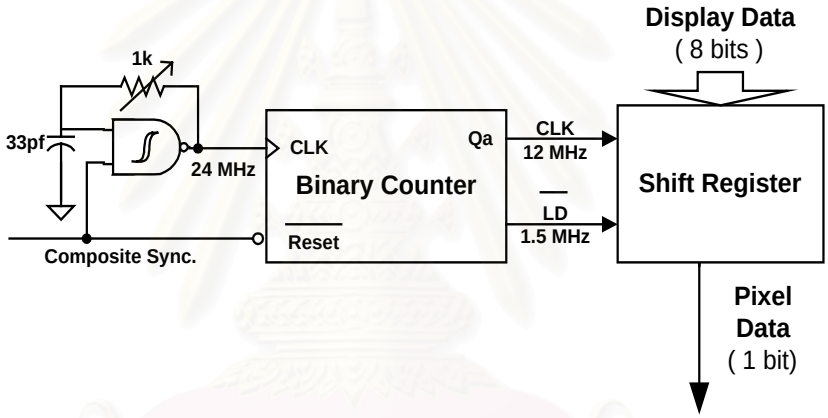
รูปที่ 5.4 การแยกข้อมูลภาพจากสัญญาณวีดิทัศน์

5.5 วงจรกำเนิดสัญญาณนาฬิกา

การแสดงผลภาพบรรยายบนจอโทรทัศน์จะใช้ความถี่ 12 MHz เพื่อแสดงผลในแนวนอน โดยความถี่นี้เกิดจากวงจรกำเนิดสัญญาณนาฬิกา ภายในวงจรกำเนิดสัญญาณนาฬิกาประกอบด้วย

1. วงจรรวมเบอร์ 74132 (NAND Schmitt Trigger)
2. วงจรรวมเบอร์ 74193 (Binary Counter)
3. วงจรรวมเบอร์ 74165 (Shift Register)

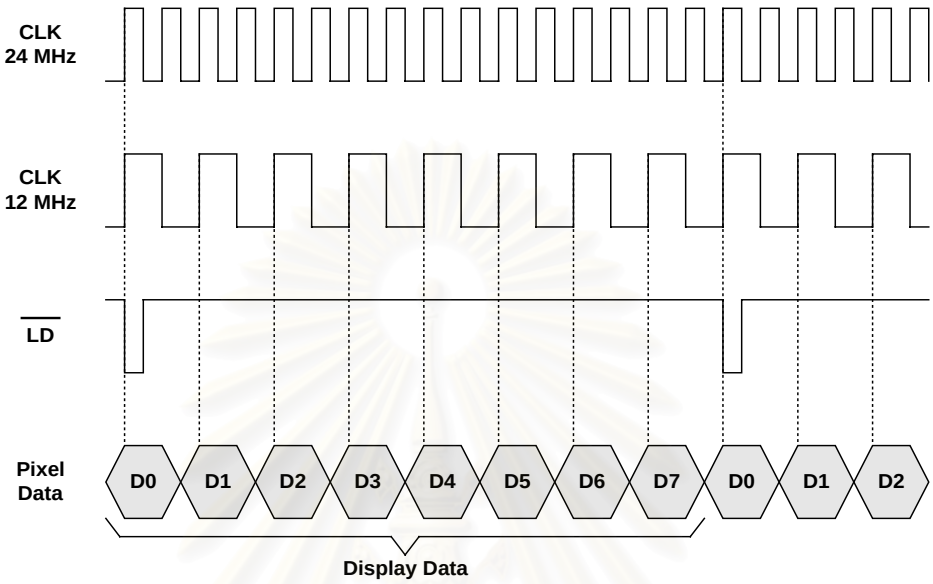
วงจรกำเนิดสัญญาณนาฬิกามีโครงสร้างดังรูปที่ 5.5



รูปที่ 5.5 วงจรกำเนิดสัญญาณนาฬิกา

เมื่อนำวงจรรวมเบอร์ 74132 มาต่อกับตัวเก็บประจุและตัวต้านทานดังแสดงในรูปที่ 5.5 จะสามารถสร้างความถี่ 24 MHzได้ สัญญาณซิงก์รวมจะต่อเข้ามาที่วงจรเพื่อทำให้วงจรสร้างสัญญาณนาฬิกาเริ่มต้นใหม่เมื่อมีการขึ้นสัญญาณวีดิทัศน์เส้นใหม่ สัญญาณนาฬิกาดังกล่าวจะส่งเข้าไปที่วงจรรวมเบอร์ 74193 โดยสัญญาณซิงก์รวมจะต่อเข้ามาที่วงจรรวมนี้เช่นกันเพื่อให้เริ่มนับใหม่เมื่อขึ้นเส้นสัญญาณวีดิทัศน์เส้นใหม่ สัญญาณที่ออกจากขาสัญญาณ Qa ของวงจรรวม 74193 คือ สัญญาณนาฬิกาความถี่ 12 MHz นอกจากนี้ยังมีสัญญาณ LD ที่มีความถี่ 1.5 MHz เพื่อสั่งให้มีการนำข้อมูลภาพชุดใหม่เพื่อนำไปแสดงผล นอกจากนี้สัญญาณ LD ยังนำไปใช้ในวงจรควบคุมการแสดงผลบนจอภาพ ดังจะกล่าวต่อไป สัญญาณจาก 74193 จะส่งไปยังวงจรรวม 74165 ซึ่งมีหน้าที่เลื่อนข้อมูลแบบขนานขนาด 8 บิต ออกไปเป็นข้อมูลแบบอนุกรมที่ความถี่

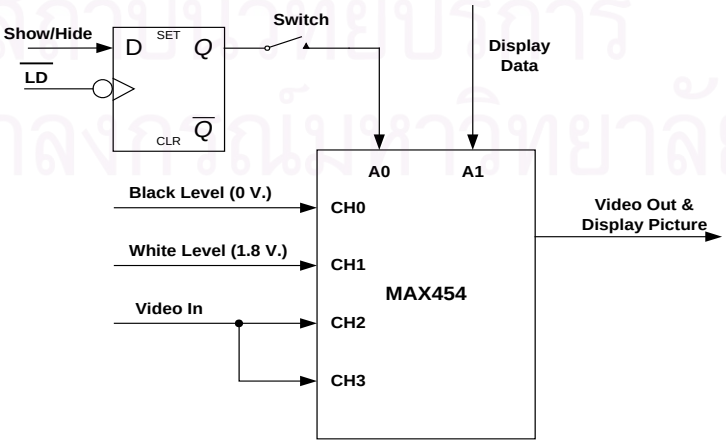
12 MHz และส่งข้อมูลดังกล่าวให้กับวงจรควบคุมการแสดงผลบนจอภาพ มีผลทางเวลา ดังแสดงในรูปที่ 5.6



รูปที่ 5.6 การแสดงผลทางเวลาของวงจรกำเนิดสัญญาณนาฬิกา

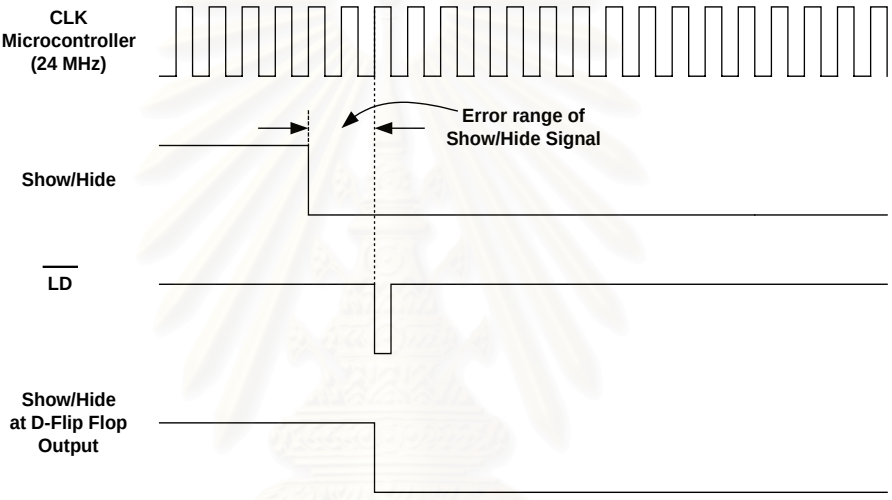
5.6 วงจรควบคุมการแสดงผลบนจอภาพ

วงจรควบคุมการแสดงผลมีหน้าที่นำข้อมูลจากวงจรกำเนิดสัญญาณนาฬิกามาแสดงบนจอภาพ ภายในวงจรประกอบด้วยวงจรรวมเบอร์ MAX 454 และวงจรรวมเบอร์ 7474 (D-Type Positive-Edge-Trigged Flip Flop) ดังแสดงในรูปที่ 5.7



รูปที่ 5.7 วงจรภายในของวงจรควบคุมการแสดงผลบนจอภาพ

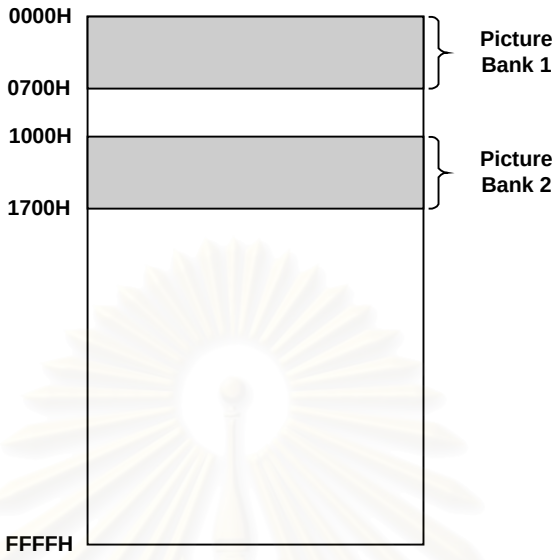
สัญญาณ Show/Hide จากตัวควบคุม จะต่อเข้าฟลิปฟล็อปชนิดดี (D-Flipflop) และสวิตช์สำหรับเปิดปิดภาพบรรยายก่อนต่อเข้า MAX454 เนื่องจากสัญญาณ Show/Hide เกิดจากการนับเวลาจากจุดเริ่มต้นเส้นสัญญาณวีดิทัศน์ โดยใช้การขัดจังหวะของตัวควบคุมบอกตำแหน่งเริ่มต้น แต่การขัดจังหวะของตัวควบคุม (DS80C320) จะตรวจสอบทุก 2 คาบของสัญญาณนาฬิกาทำให้การส่งสัญญาณ Show/Hide อาจช้าไป 2 คาบสัญญาณนาฬิกา จึงแก้โดยนำสัญญาณ Show/Hide ผ่านฟลิปฟล็อปและใช้สัญญาณ LD เป็นสัญญาณนาฬิกา ซึ่งสัญญาณ Show/Hide ที่ผ่านฟลิปฟล็อปจึงเกิดที่เวลาใกล้เคียงกัน ดังแสดงในรูปที่ 5.8



รูปที่ 5.8 ผลทางเวลาของสัญญาณ Show/Hide

5.7 หน่วยความจำ

ภาพบรรยาย 1 ภาพมีขนาดข้อมูล 1792 ไบต์ และการแสดงภาพจะแสดงสลับกันทีละ 1 ภาพ ดังนั้นหน่วยความจำจะต้องมี 2 ส่วนเพื่อเก็บภาพบรรยายแต่ละภาพ หน่วยความจำแต่ละส่วนจะมีขนาด 2 Kbyte สำหรับเก็บข้อมูลภาพบรรยาย 1 ภาพ โดยหน่วยความจำส่วนแรกจะเริ่มจากตำแหน่ง 0000H ของหน่วยความจำ ส่วนที่สองเริ่มจากตำแหน่ง 1000H การเก็บข้อมูลของแต่ละภาพจะเริ่มเก็บตั้งแต่ตำแหน่งเริ่มต้นไปจนสิ้นสุดข้อมูลภาพ ดังแสดงในรูปที่ 5.9



รูปที่ 5.9 ตำแหน่งข้อมูลภาพบรรยายในหน่วยความจำภายนอก

5.8 การคลายข้อมูล

ข้อมูลจะส่งมาในช่วงไร้ภาพแนวตั้งของสัญญาณวีดิทัศน์เส้นละ 3 ไบต์ จำนวน 16 เส้นต่อ 1 ฟีลด์ ดังนั้นข้อมูลที่ใช้คลายจะมีทั้งหมด 48 ไบต์ และเก็บไว้ในหน่วยความจำภายในของไมโครคอนโทรลเลอร์ ข้อมูลจะถูกนำมาคลายทีละ 1 ไบต์ จนครบ 48 ไบต์

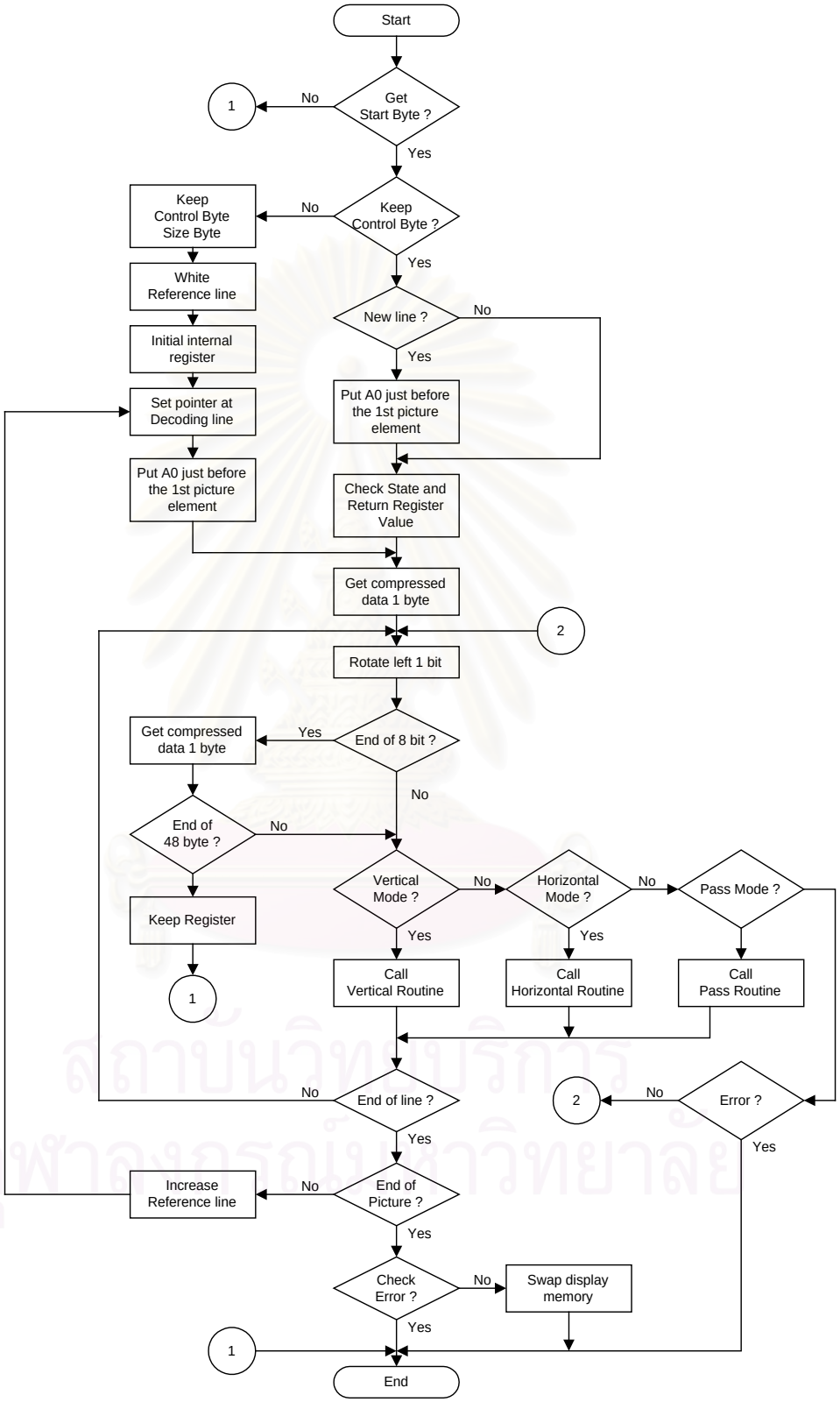
ตัวแปรสำคัญที่ใช้คลายข้อมูล ได้แก่

1. จำนวนเส้นภาพ (Number of line)นับจำนวนเส้นภาพของภาพบรรยายขณะคลายข้อมูล มีค่าตั้งแต่ 1-128
2. จำนวนจุดสี (Number of pixel) นับจำนวนจุดสีในเส้นภาพของภาพบรรยายขณะคลายข้อมูล มีค่าตั้งแต่ 1- 112
3. ตำแหน่งจุดเปลี่ยนสี A0 ซึ่งเป็นจุดอ้างอิงเพื่อหาตำแหน่งจุดอื่นๆ ที่ใช้คลายข้อมูลในแต่ละโมด
4. สถานะจุดสี (State) บอกสีของจุดที่ A0 ชี้ โดยค่า 1 หมายถึง สีขาว และ 0 หมายถึง สีดำ

5. ตัวชี้ข้อมูลของเส้นภาพแถวที่ถูกคลาย แบ่งออกเป็น 2 ชนิด ได้แก่
    1. ตัวชี้ข้อมูลระดับไบต์ บอกตำแหน่งของหน่วยความจำภายในที่เก็บข้อมูลที่ถูกบีบอัด
    2. ตัวชี้ข้อมูลระดับบิต บอกตำแหน่งบิตภายในไบต์ที่กำลังคลายข้อมูล
6. ตัวชี้ข้อมูลแถวอ้างอิง แบ่งออกเป็น 2 ชนิด ได้แก่
    1. ตัวชี้ข้อมูลระดับไบต์ บอกตำแหน่งของหน่วยความจำภายในที่เก็บข้อมูลแถวอ้างอิง
    2. ตัวชี้ข้อมูลระดับบิต บอกตำแหน่งบิตภายในไบต์ของแถวอ้างอิง
7. ตัวชี้ข้อมูลภาพบรรยายตัว (Display Pointer) ใช้บอกค่าตำแหน่งเลขที่อยู่ (Address) ของภาพบรรยายที่กำลังคลายข้อมูลในหน่วยความจำภายนอก ว่าคลายถึงตำแหน่งใดของภาพ
8. จำนวนข้อมูล บอกจำนวนข้อมูลที่ได้คลายไปแล้ว มีค่า 0 - 48
9. ขนาดข้อมูลภาพ บอกขนาดข้อมูลที่ถูกบีบอัดทั้งหมดล

การคลายข้อมูลมีการทำงานดังรูปที่ 5.10 และมีขั้นตอนดังนี้

1. ตรวจจับไบต์เริ่มต้นเพื่อเริ่มคลายข้อมูล
2. ตรวจสอบไบต์ควบคุมและไบต์ขนาด หากจัดเก็บแล้วจะตรวจสอบว่าเริ่มต้นเส้นใหม่หรือไม่ เพื่อตั้งค่าตำแหน่ง A0 จากนั้นจะตั้งค่าตัวแปรที่ใช้คลายข้อมูล
3. หลังการเก็บไบต์ควบคุมและไบต์ขนาด จะต้องตั้งค่าเริ่มต้นให้กับตัวแปรที่สำคัญ เลือกเส้นอ้างอิงเป็นเส้นภาพสีขาว และตั้งค่า A0 ให้เป็น 0
4. นำข้อมูลที่บีบอัด 1 ไบต์จากหน่วยความจำภายใน และเพิ่มค่าตัวชี้ระดับไบต์ของข้อมูล
5. เลื่อนบิตไปทางซ้ายครั้งละ 1 บิต และเพิ่มค่าตัวชี้ระดับบิตของข้อมูล
6. ตรวจสอบว่าข้อมูลถูกเลื่อนครบ 8 บิตหรือไม่ ถ้าครบจะนำข้อมูลอีก 1 ไบต์จากหน่วยความจำ และเพิ่มค่าตัวชี้ระดับไบต์ของข้อมูล หากจำนวนไบต์ที่นำมาใช้เกิน 48 ไบต์ จะเก็บค่าตัวแปรต่างๆ และออกจากการทำงาน
7. การเลื่อนบิต 1 ครั้งจะนำมาตรวจสอบโมดการบีบอัด และแยกไปทำงานตามกระบวนงาน (procedure) ย่อยต่างๆ หากไม่พบจะตรวจสอบความผิดพลาด ถ้าไม่พบจะกลับไปทำข้อ 5 อีกครั้ง ถ้าพบจะหยุดการทำงานและไม่แสดงภาพบรรยายที่ผิดพลาด



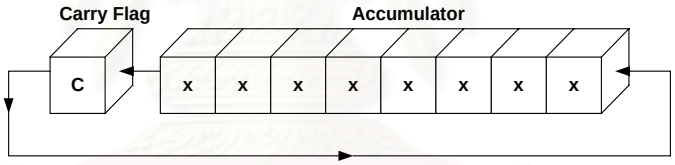
รูปที่ 5.10 ผังงานการคลายข้อมูล

8. เมื่อคลายข้อมูลในแต่ละโมดแล้วจะตรวจสอบว่าสิ้นสุดเส้นภาพหรือหากไม่ถึงจุดสิ้นสุดเส้นภาพจะกลับไปทำข้อ 5 อีกครั้ง
9. หลังคลายข้อมูลได้ 1 เส้นภาพ จะตรวจสอบว่าครบภาพบรรยายหรือไม่ (128 เส้น) หากไม่ครบจะนำเส้นภาพที่คลายแล้วเป็นเส้นอ้างอิง
10. เมื่อคลายภาพบรรยายครบ 1 ภาพ จะเปลี่ยนตัวชี้ข้อมูลภาพบรรยายตัวที่ 2 ให้ชี้ที่ภาพที่คลายภาพแล้ว เพื่อรอการแสดงภาพบนจอโทรทัศน์

กระบวนงานย่อยที่สำคัญของการคลายข้อมูล แบ่งได้ดังนี้

1. กระบวนงานตรวจสอบโมดบีบอัด

กระบวนงานนี้มีหน้าที่นำข้อมูลที่ถูกบีบอัดมาเลื่อนทีละ 1 บิตเพื่อหาโมดบีบอัดหรือ ความผิดพลาดต่างๆ ข้อมูลจะถูกนำมาใส่ตัวสะสม (Accumulator) ครั้งละ 1 ไบต์ และบิตข้อมูลจะถูกเลื่อนไปทางซ้ายและนำไปเก็บในตัวบ่งชี้ตัวทด (Carry Flag) ด้วยคำสั่ง " RLC A" ดังแสดงในรูปที่ 5.11



รูปที่ 5.11 การเลื่อนข้อมูล

โมดแนวตั้งที่แสดงในผังงานมีเพียง 4 โมด เนื่องจากรหัสเลือกโมดจะบอกระยะห่างของจุดเปลี่ยนสีในเส้นภาพกับเส้นอ้างอิงตามด้วยทิศทางที่ห่างออกไป (ซ้ายหรือขวา) เช่น รหัสของโมดแนวตั้งที่มีระยะห่าง 2 จุดสี คือ "00001X" ถ้าบิตต่อมาคือ "1" หมายถึง ทางขวา "0" หมายถึง ทางซ้าย ดังนั้นรหัส "000011" หมายถึง รหัสของโมดแนวตั้งที่มีระยะห่างไปทางขวา 2 จุดสี การตรวจสอบโมดบีบอัดจึงหาเพียงระยะห่างของจุดเปลี่ยนสี ซึ่งมีค่า 0,1,2 และ 3 เท่านั้น ส่วนทิศทางจะแบ่งเป็นโมดย่อยหลังจากเข้าโมดแนวตั้งแล้ว

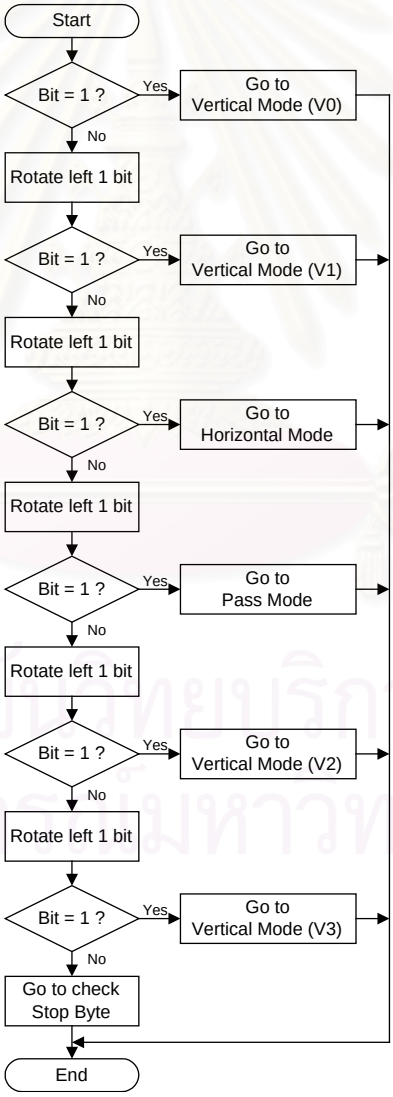
ตัวอย่างการเลื่อนข้อมูลเพื่อเลือกโมดบีบอัด

ถ้าข้อมูลที่ได้รับ คือ "00001111 01001111 …"

จะแบ่งออกเป็นโมด ได้ดังนี้

000011	โมดแนวตั้ง (ระยะห่าง 2 จุด)
1	โมดแนวตั้ง (ตำแหน่งตรงกัน)
1	โมดแนวตั้ง (ตำแหน่งตรงกัน)
010	โมดแนวตั้ง (ระยะห่าง 1 จุด)
011	โมดแนวตั้ง (ระยะห่าง 1 จุด)
1	โมดแนวตั้ง (ตำแหน่งตรงกัน)
1	โมดแนวตั้ง (ตำแหน่งตรงกัน)

การตรวจสอบโมดบีบอัดสรุปเป็นผังงาน ดังแสดงในรูปที่ 5.12



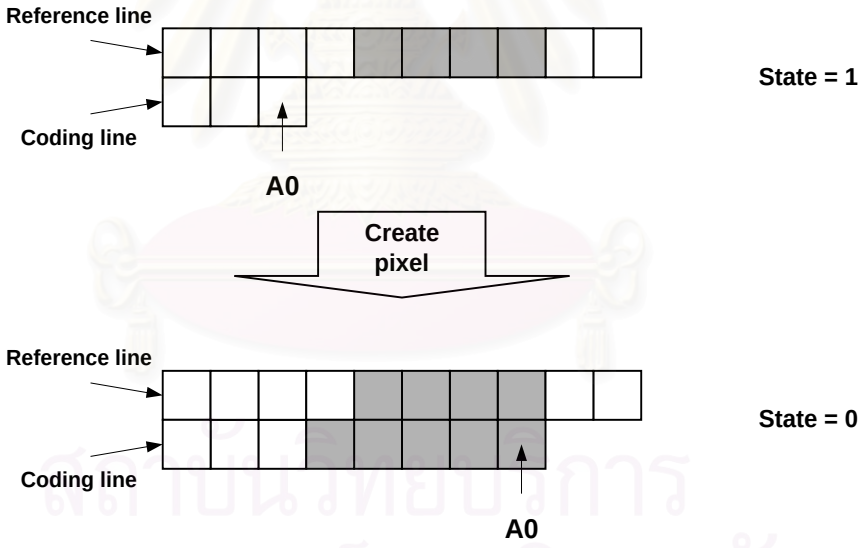
รูปที่ 5.12 ผังงานการเลือกโมดบีบอัดข้อมูล

2. กระบวนงานคลายโมดแนวตั้ง (Vertical mode decompression routine)

การคลายแนวตั้ง ประกอบด้วยโมดภายใน 7 โมด คือ โมดแนวตั้งที่ตำแหน่งจุดเปลี่ยนสีของเส้นภาพและเส้นอ้างอิงตรงกันจำนวน 1 โมด และโมดแนวตั้งที่ตำแหน่งจุดเปลี่ยนสีมีระยะห่างไม่เกิน 3 จุดสีจำนวน 6 โมด ซึ่งแบ่งเป็นเยื้องทางซ้ายและทางขวาแบบละ 3 โมด

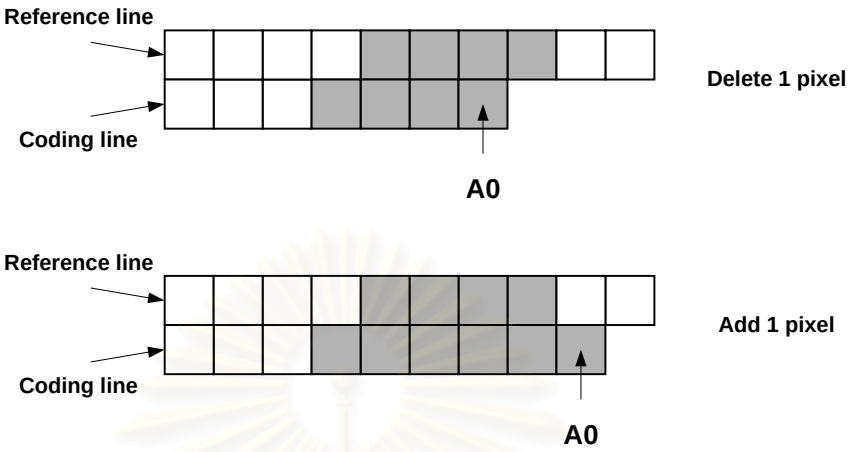
ภาพบรรยายเกิดจากจุดสีที่สร้างขึ้นจากข้อมูลที่ได้ โดยสีของจุดที่สร้างจะดูจากสถานะจุดสี ซึ่งเก็บสีของจุดที่ A0 ชี้ โดยค่า "1" คือ จุดสีขาว และ "0" คือ จุดสีดำ จุดที่สร้างจะมีสีตรงข้ามกับจุดที่ A0 ชี้

โมดแนวตั้งทุกโมดจะใช้ชุดคำสั่งประจำหลัก (Main Routine) เดียวกัน ซึ่งมีหน้าที่หาตำแหน่งจุดเปลี่ยนสีที่อยู่ทางขวา A0 และมีสีตรงข้ามกับ A0 บนเส้นอ้างอิงและสร้างจุดสีไปถึงตำแหน่งดังกล่าว ดังแสดงในรูปที่ 5.13 เมื่อได้จุดสีที่ตำแหน่งตรงกับจุดเปลี่ยนสีของแถวอ้างอิงแล้วจึงปรับข้อมูลจุดสีตามโมดภายในที่ตรวจสอบได้



รูปที่ 5.13 การสร้างจุดสีของเส้นภาพ

การปรับตำแหน่งจุด คือ การลบหรือเพิ่มจุดสีเพิ่มลงในเส้นภาพ โดยอ้างอิงตำแหน่ง A0 ว่าจะลบหรือเพิ่มจุดสีกี่จุด (ไม่เกิน 3 จุด) ดังแสดงในรูปที่ 5.14



รูปที่ 5.14 การปรับตำแหน่งจุด

เมื่อปรับตำแหน่งจุดสี ตำแหน่ง A0 จะเลื่อนไปตามจุดสีจุดสุดท้ายของเส้นภาพ ดังแสดงในรูปที่ 5.14 เพื่อเป็นตำแหน่งอ้างอิงต่อไป

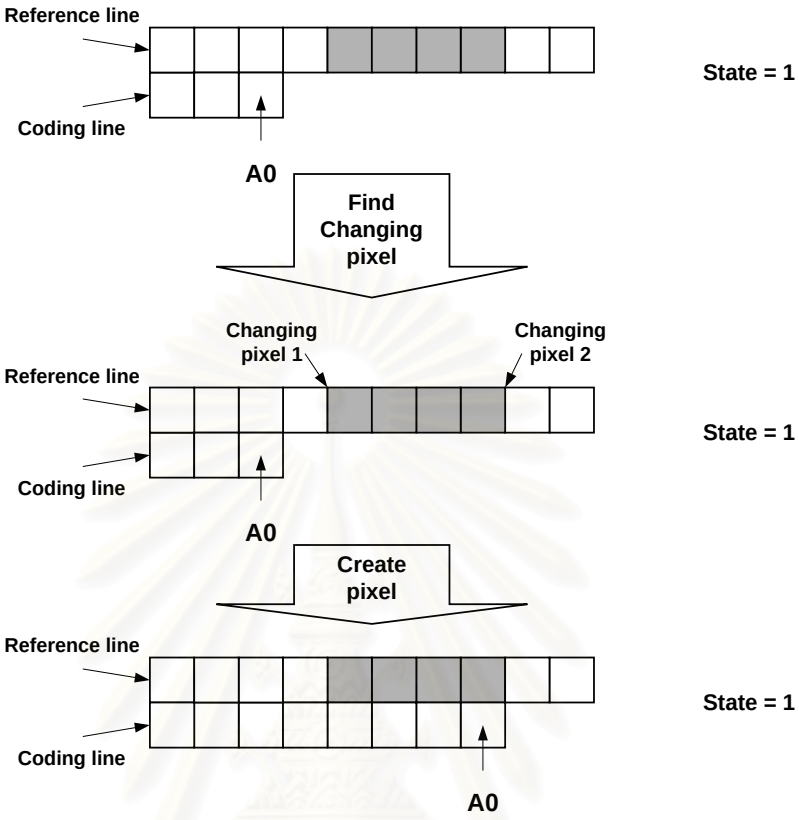
โปรแกรมจะใช้หน่วยความภายในไมโครคอนโทรลเลอร์เก็บข้อมูลเส้นอ้างอิงและเส้นภาพที่ตำแหน่งเดียวกัน เพื่อเพิ่มความเร็วของกระบวนงานคลายข้อมูลแบบโมดแนวตั้ง เนื่องจากการคลายแบบนี้เส้นภาพจะมีจุดสีคล้ายกับเส้นอ้างอิงมากที่สุด การลบและเพิ่มจุดสีจึงเกิดที่จุดเปลี่ยนสีเท่านั้น เวลาที่ใช้สร้างเส้นภาพจึงลดลง

สรุปขั้นตอนการคลายแบบโมดแนวตั้งได้ดังนี้

1. ตรวจหาจุดเปลี่ยนสีที่อยู่ทางขวา A0 และมีสีตรงข้ามกับ A0 บนเส้นอ้างอิง
2. สร้างจุดสีที่มีสีตรงข้ามกับค่าสถานะสี
3. ตรวจสอบโมดเพื่อลบหรือเพิ่มจุดสี
4. เลื่อนค่าตำแหน่ง A0 ไปจุดสีจุดสุดท้าย
5. เปลี่ยนค่าสถานะจุดสีเป็นค่าตรงข้ามกับค่าเดิม

3.กระบวนงานคลายโมดผ่าน (Pass mode decompression rouitne)

การคลายโมดผ่านจะหาตำแหน่งจุดเปลี่ยนสีบนเส้นอ้างอิง 2 ครั้ง โดยครั้งแรกจะหาตำแหน่งจุดเปลี่ยนสีทางขวาของ A0 ครั้งที่สองจะหาตำแหน่งจุดเปลี่ยนสีถัดไปทางขวาของจุดแรก จุดสีที่สร้างบนเส้นภาพจะมีสีเหมือนกับจุดที่ A0 ชี้ โดยสร้างจาก A0 ถึงจุดเปลี่ยนสีจุดที่สอง ดังแสดงในรูปที่ 5.15



รูปที่ 5.15 การคลายข้อมูลโมดผ่าน

การคลายโมดผ่านมีขั้นตอนการทำงานคล้ายกับโมดแนวตั้ง แต่จุดสีที่สร้างจะมีสีเหมือนค่าสถานะสี และไม่มีการเปลี่ยนค่าสถานะสีเมื่อคลายเสร็จ

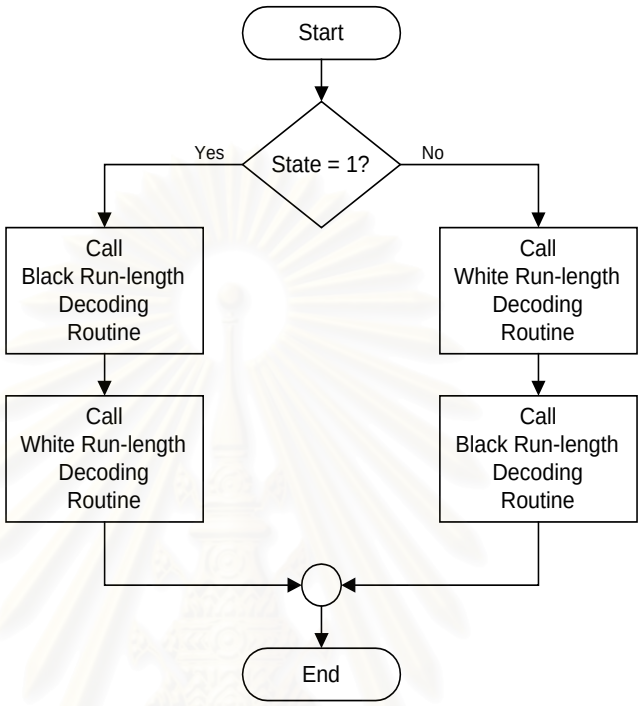
สรุปขั้นตอนการคลายโมดผ่านได้ดังนี้

1. ตรวจหาจุดเปลี่ยนสีที่ 1 และ 2
2. สร้างจุดสีที่มีสีเหมือนค่าสถานะสี
3. เลื่อนตำแหน่ง A0 ไปจุดสีจุดสุดท้าย
4. คงค่าสถานะจุดสี

4.กระบวนงานคลายโมดแนวราบ (Horizontal mode decompression routine)

การคลายโมดแนวราบจะต่างจาก 2 โมดข้างต้น เพราะโมดแนวราบสร้างจุดสีจากรหัสที่ต่อจากรหัสบอกโมด โดยไม่ใช้ความสัมพันธ์ของเส้นอ้างอิง รหัสโมดแนวราบ 1 ชุดสร้างจุดสี 2 สี คือ จุดสีดำขาว หรือ ขาวดำ เรียงต่อกันไม่เหมือนโมดแนวตั้งและโมด

ผ่านที่สร้างจุดสี 1 สี จุดสีที่สร้างจะเริ่มด้วยสีขาวหรือสีดำก่อนขึ้นกับค่าสถานะจุดสีของ A0 ดังแสดงในรูปที่ 5.16



รูปที่ 5.16 ผังงานการคลายโมดแนวราบ

เมื่อทราบสถานะของจุดสีที่ตำแหน่ง A0 จะทราบลำดับการเรียกกระบวนงานย่อย ซึ่งประกอบด้วย 2 กระบวนงานคือ

1. กระบวนงานสร้างจุดสีขาว (White run-length decoding routine)
2. กระบวนงานสร้างจุดสีดำ (Black run-length decoding routine)

กระบวนงานสร้างจุดสีขาวและสีดำจะนำข้อมูลที่ถูกบีบอัดมาเปรียบเทียบกับค่าในตารางเพื่อหาจำนวนจุดสีที่ต้องสร้างบนเส้นภาพ เนื่องจากความยาวของรหัส Run-length แต่ละค่ามีความยาวรหัสไม่เท่ากันจึงต้องปรับความยาวให้เท่ากัน เพื่อการคลายข้อมูลที่เร็วขึ้น ตารางสำหรับคลายข้อมูลถูกดัดแปลงจากตารางรหัส Run-length สีขาวและดำ (ตารางที่2.2) โดยเติมบิต ‘0’ ต่อท้ายรหัสเดิมได้รหัสใหม่มีขนาด 8 บิตสำหรับจุดสีขาว และ 12 บิตสำหรับจุดสีดำ ซึ่งเป็นความยาวสูงสุดของรหัสสีขาวและสีดำ ตัวอย่างตารางที่เปลี่ยนแปลง ดังแสดงในตารางที่ 5.1 และ 5.2

รหัสเดิม	รหัสใหม่
1100	11000000
10011	10011000
000011	00001100
0100111	01001110
00110011	00110011

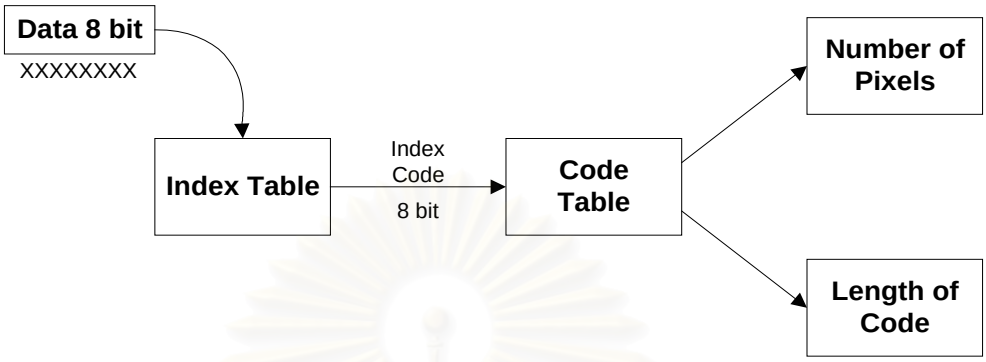
ตารางที่ 5.1 รหัสใหม่สำหรับจุดสีขาว

รหัสเดิม	รหัสใหม่
11	110000000000
011	011000000000
0011	001100000000
00011	000110000000
000101	000101000000
00000111	000001110000
000011000	000011000000
0000010111	000001011100
00001100111	000011001110
000011001011	000011001011

ตารางที่ 5.2 รหัสใหม่สำหรับจุดสีดำ

รหัสใหม่จะเป็นข้อมูลของตารางตัวชี้ (Index Table) เพื่อบอกรหัสตัวชี้ (Index code) โดยรหัสนี้จะนำไปใช้ในตารางรหัส (Code Table) ซึ่งบอกจำนวนจุดสีและขนาดความยาวจริงของรหัส

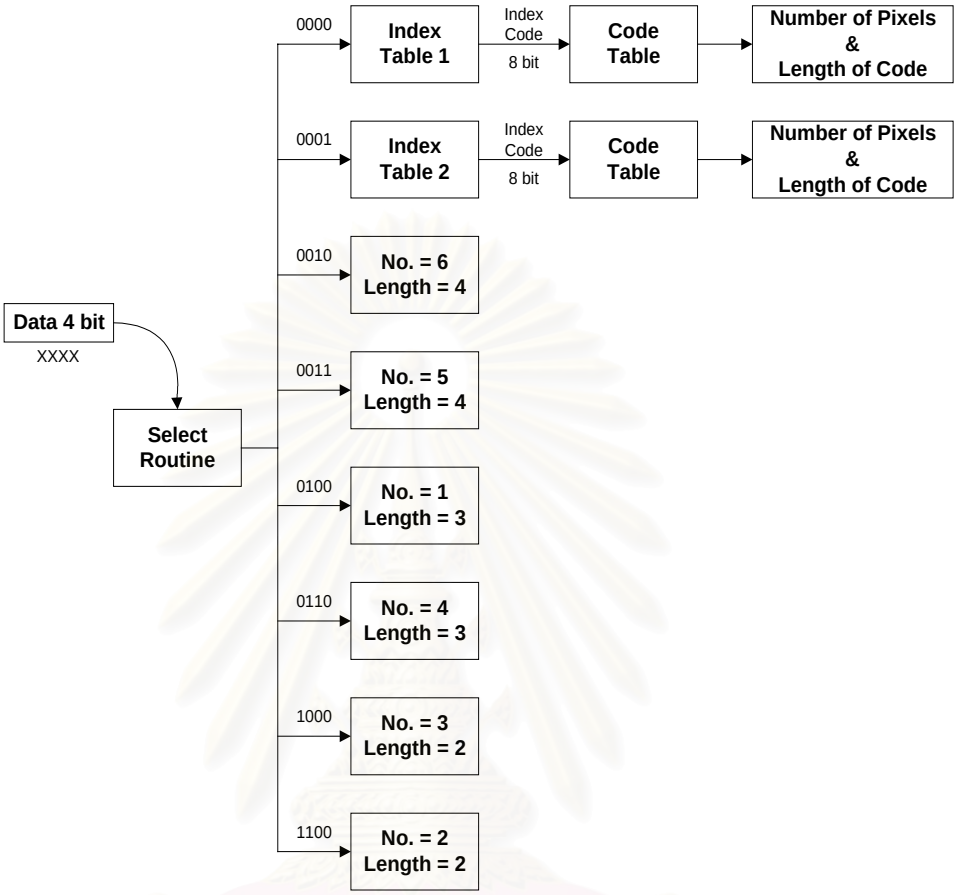
การสร้างจุดสีขาวใช้ข้อมูลครั้งละ 8 บิต โดยนำข้อมูลเปรียบเทียบกับค่ารหัสตัวชี้ของตารางตัวชี้ เพื่อหาจำนวนจุดสีและความยาวรหัสเดิม ดังแสดงในรูปที่ 5.17



รูปที่ 5.17 การหาจำนวนจุดสีและความยาวรหัสสำหรับสีขาว

การสร้างจุดสีใช้ข้อมูลครั้งละ 12 บิต แต่ไมโครคอนโทรลเลอร์รับส่งข้อมูลได้ 8 บิต จึงแบ่งข้อมูลเป็น 2 ชุด คือ 4 บิต และ 8บิต ข้อมูล 4 บิตใช้เลือกกระบวนงานย่อย ซึ่งแบ่งเป็น 8 กระบวนงานย่อย ดังแสดงในรูปที่ 5.18 และมีรายละเอียดดังนี้

1. ข้อมูล 4 บิต คือ '0000' นำข้อมูลอีก 8 บิตมาเป็นข้อมูลของตารางตัวชี้ที่ 1 เพื่อบอกรหัสตัวชี้ของตารางรหัส ซึ่งบอกจำนวนจุดและความยาวรหัส
2. ข้อมูล 4 บิต คือ '0001' นำข้อมูลอีก 8 บิตมาเป็นข้อมูลของตารางตัวชี้ที่ 2 เพื่อบอกรหัสตัวชี้ของตารางรหัส ซึ่งบอกจำนวนจุดและความยาวรหัส
3. ข้อมูล 4 บิต คือ '0010' ได้จำนวนจุด 6 จุด ความยาวรหัส 4 บิต
4. ข้อมูล 4 บิต คือ '0011' ได้จำนวนจุด 5 จุด ความยาวรหัส 4 บิต
5. ข้อมูล 4 บิต คือ '0100' ได้จำนวนจุด 1 จุด ความยาวรหัส 3 บิต
6. ข้อมูล 4 บิต คือ '0110' ได้จำนวนจุด 4 จุด ความยาวรหัส 3 บิต
7. ข้อมูล 4 บิต คือ '1000' ได้จำนวนจุด 3 จุด ความยาวรหัส 2 บิต
8. ข้อมูล 4 บิต คือ '1100' ได้จำนวนจุด 2 จุด ความยาวรหัส 2 บิต



รูปที่ 5.18 การหาจำนวนจุดสีและความยาวรหัส

เมื่อทราบจำนวนจุดและความยาวรหัสแล้ว โปรแกรมจะสร้างจุดสีตามจำนวนจุดที่ต้องการ และปรับจำนวนข้อมูลที่ใช้คลายจริง เช่น การสร้างจุดสีขาวดึงข้อมูลมาใช้ 8 บิต ผลจากตารางได้ความยาวรหัส 5 บิต ดังนั้นจะลดตัวชี้ข้อมูลระดับบิตลงมา 3 บิต

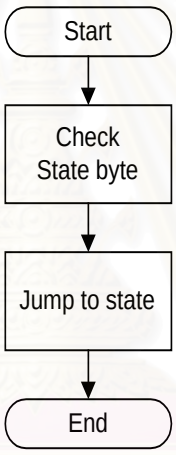
สรุปขั้นตอนการคลายโมดแนวราบได้ดังนี้

1. ตรวจสอบสถานะจุดสีตำแหน่ง A0 เพื่อเลือกกระบวนงานว่าสร้างจุดสีขาวหรือดำก่อน
2. กระบวนงานสร้างจุดขาว ใช้ข้อมูลความยาว 8 บิตสำหรับคลายข้อมูล
3. กระบวนงานสร้างจุดสีดำ ใช้ข้อมูล 12 บิตคลายข้อมูล

4. สำหรับการสร้างจุดสีดำ ข้อมูล 4 บิตแรกใช้เลือกกระบวนงานย่อย และ 8 บิตถัดไปใช้สำหรับบางกระบวนงานย่อย
5. นำจำนวนจุดที่ได้มาสร้างจุดสี และเลื่อนตัวชี้ข้อมูลระดับบิตไปตำแหน่งข้อมูลสุดท้ายตามค่าความยาวรหัส
6. เลื่อน A0 มาที่ตำแหน่งสุดท้ายของจุดสี
7. ตั้งค่าสถานะจุดสีตามตำแหน่งจุดที่ A0 ชี้

5.กระบวนงานตรวจสอบตำแหน่งการทำงาน

การทำงานของกระบวนงานแสดงดังรูปที่ 5.19



รูปที่ 5.19 ผังงานกระบวนงานสร้างจุดสี

ข้อมูลที่บีบอัดถูกแยกส่งมาครั้งละ 48 ไบต์ต่อฟีลด์ การทำงานจึงหยุดที่โมดต่างกันเมื่อข้อมูลหมด ดังนั้นก่อนออกจากกระบวนงานจึงเก็บค่าโมดที่กำลังทำงานไว้ในไบต์สถานะ (State Byte) เมื่อเริ่มเข้ากระบวนงานคลายข้อมูลในฟีลด์ต่อไป ไบต์สถานะจะบอกว่าต้องไปทำงานที่กระบวนงานใด เพื่อให้ได้ข้อมูลภาพที่สมบูรณ์

5.9 โปรแกรมการทำงานของเครื่องรับข้อมูลภาพ

โปรแกรมควบคุมไมโครคอนโทรลเลอร์พัฒนาด้วยภาษาแอสเซมบลี โดยมีขอบเขตการทำงานของโปรแกรมดังนี้

1. สามารถรับข้อมูลจากช่วงไร้ภาพแนวตั้งได้
2. สามารถคลายข้อมูล 48 ไบต์เสร็จก่อนแสดงภาพบนจอภาพ
3. แสดงผลบนมุมขวาล่างของจอภาพ

การทำงานของโปรแกรมอธิบายด้วยภาษาบรรยายโปรแกรม แบ่งออกเป็น 3 ระดับ โดยแต่ละระดับจะมีความละเอียดเพิ่มขึ้นตามลำดับ

1.ภาษาบรรยายโปรแกรมระดับที่ 1

```
Module: Main
  Initialization
  Repeat
        Wait for update new line
        Case (N) * is                    ; * N = Number of video scan line
            When 6<N<23       => Read 3 bytes of data in the video line and
                                 store in internal RAM
            When N = 23       => Call decompression routine
            When 205<N<269 => On screen display one line of picture
                                 (112 dots) from the display RAM in the
                                 video line
            When others       => No operation
  Until Forever
END;
```

Module: Decompression

Initialization

Decompress data and store the results in the Non-display RAM

END;

โปรแกรมจะเริ่มทำงานโดยกำหนดค่าเริ่มต้น และรับข้อมูลภาพในช่วงสัญญาณที่ 7 ถึง 22 โดยสุ่มตรงกลางของข้อมูลด้วยความถี่ข้อมูล 500 kbit/sec หลังสุ่มข้อมูลแล้วจะเก็บไว้ในหน่วยความจำภายใน เมื่อถึงเส้นสัญญาณที่ 23 จะเริ่มคลายข้อมูลในหน่วยความจำภายใน หลังคลายข้อมูลภาพบรรยายจะเก็บในหน่วยความจำภายนอก ภาพบรรยายจะถูกแสดงบนจอภาพในช่วงเส้นสัญญาณที่ 205 - 269 การทำงานจะวนกลับไปเริ่มต้นใหม่ และทำไปไม่สิ้นสุด

2. ภาษาบรรยายโปรแกรมระดับที่ 2

Module: Main

Initialization

Clear internal memory

Clear display memory

Initialize interrupt

Initialize register

Repeat

Wait for update new line

Increase line number

Case (N) * is

When 6<N<23 =>

Read 3 bytes of data in the video line and store in internal RAM

Sampling data at rate 500 kbit/sec

Store data to the internal

When N = 23 =>

Call decompression routine

```
            When 205<N<269  =>
                On screen display one line of picture (112 dots) from the display
                RAM in the video line
                        Move data from external RAM to internal RAM
                        Send data to Port1 at rate 1.5 Mbyte/sec
            When others          =>
                        No operation
  Until Forever
END;

Module Decompression
  Initialization
        Initialize register
        Set data pointer
  Decompress data and store the results in the Non-display RAM
        Repeat
                Check compression mode
                Decompression data in each mode
                If end of decoding linethen
                        Move data to external RAM
                End if
        Until end of data
        Set display pointer for on screen display
End;
```

3. ภาษาบรรยายโปรแกรมระดับที่ 3

```
Module: Main
 Initialization
      Clear internal memory
        Write 00h to internal memory
      Clear display memory
        White 00h to internal memory
      Initialize interrupt
        Enable interrupt
        Set active interrupt
      Initialize register
        Set data pointer to start address
        Set display pointer to first bank address
        Write 00h to Accumulator
 Repeat
      Wait for update new line
          Repeat
             No operation
          Until detect new line
          Increase line number
      Case (N) * is
          When 6<N<23      =>
                Read 3 bytes of data in the video line and store in internal RAM
                     Sampling data at rate 500 kbit/sec
                          Repeat
                               Get data from Compressed data signal to
                               Carry flag
                               Shift carry flag to accumulator
```

```
                    Until get 1 byte of data
                Store data to the internal
                    Move data in accumulator to internal RAM
                    Increase data pointer
        When N = 23         =>
            Call decompression routine
        When 205<N<269  =>
            On screen display one line of picture (112 dots) from the display
            RAM in the video line
                Move data from external RAM to internal RAM
                    Set data pointer at start of internal address
                    Repeat
                        Move data from external to internal RAM
                        Increase data pointer
                        Increase display pointer
                    Until end of 14 bytes
                Send data to Port1 at rate 1.5 Mbyte/sec
                    Set data pointer at start of internal address
                    Repeat
                        Move data internal RAM to Port1
                        Increase data pointer
                    Until end of 14 bytes
        When others     =>
                No operation
  Until Forever
END;
```

```
Module Decompression
  Initialization
      Initialize register
          Set A0 pointer to 00h
          Write decoding line at 00h

      Set data pointer
          Set decompressed data pointer at start address
          Set compressed data pointer at start address
          Set display data pointer at start address
  Decompress data and store the results in the Non-display RAM
      Repeat
          Check compression mode
              Move compressed data 1 byte to accumulator
              Repeat
                  Shift data from accumulator 1 bit
                  check compression mode
              Until end of byte
              Increase compressed data pointer
              Go to check compression mode
          Decompression data in each mode
          If end of decoding linethen
              Move data to external RAM
                  Repeat
                      Move data from decoding line to external
                      RAM
                      Increase data decompressed pointer
                      Increase display data pointer
                  Until end of 14 bytes
```

```
                    Set data decompressed pointer at start address
                End if
            Until end of data
            Set display pointer for on screen display
                If (decompression is finished) then
                    Swap display memory to another bank
                End if
                Set display memory at start address
End;
```

บทที่ 6

การพัฒนาโปรแกรมแทรกภาพบรรยาย

โปรแกรมแทรกภาพบรรยายเป็นโปรแกรมที่สร้างขึ้นเพื่อนำภาพบรรยายที่จะแทรกลงในสัญญาณวีดิทัศน์มาบีบอัดตามมาตราฐาน CCITT T.4 แบบ 2 มิติ และใส่ไบต์ข้อมูล ก่อนส่งให้เครื่องแทรกข้อมูล โปรแกรมจะรับสัญญาณขอข้อมูลทุกฟีลด์ผ่านทางพอร์ตอนุกรม (RS232) และนับเวลาที่ต้องส่งข้อมูลแล้วจึงส่งข้อมูลให้เครื่องแทรกข้อมูลภาพครั้งละ 48 ไบต์

6.1 การเลือกใช้ตัวแปลโปรแกรม

โปรแกรมที่สามารถทำงานได้ดังที่กล่าวมาแล้วข้างต้นเป็นโปรแกรมที่พัฒนาขึ้นมาเองเพื่อใช้ติดต่อเครื่องแทรกข้อมูลภาพ การพัฒนาโปรแกรมในวิทยานิพนธ์นี้เลือกใช้ Visual Basic Version 6.0 เป็นตัวแปลโปรแกรม (Complier) เนื่องด้วยเหตุผล 2 ประการคือ

1. โปรแกรมที่เขียนต้องทำงานภายใต้ระบบปฏิบัติการวินโดวส์ และมีการเขียนโปรแกรมเป็นแบบวิชวล (Visual Programming)
2. Visual Basic สามารถพัฒนาต่อได้ง่าย เนื่องจากโครงสร้างภาษาไม่ซับซ้อน

เหตุผลที่เขียนโปรแกรมให้ทำงานภายใต้ระบบปฏิบัติการวินโดวส์เนื่องจากระบบปฏิบัติการดังกล่าวนิยมใช้อย่างแพร่หลาย และโปรแกรมแบบวิชวลมีความสะดวกในการพัฒนา ทำให้ใช้เวลาในการพัฒนาน้อย

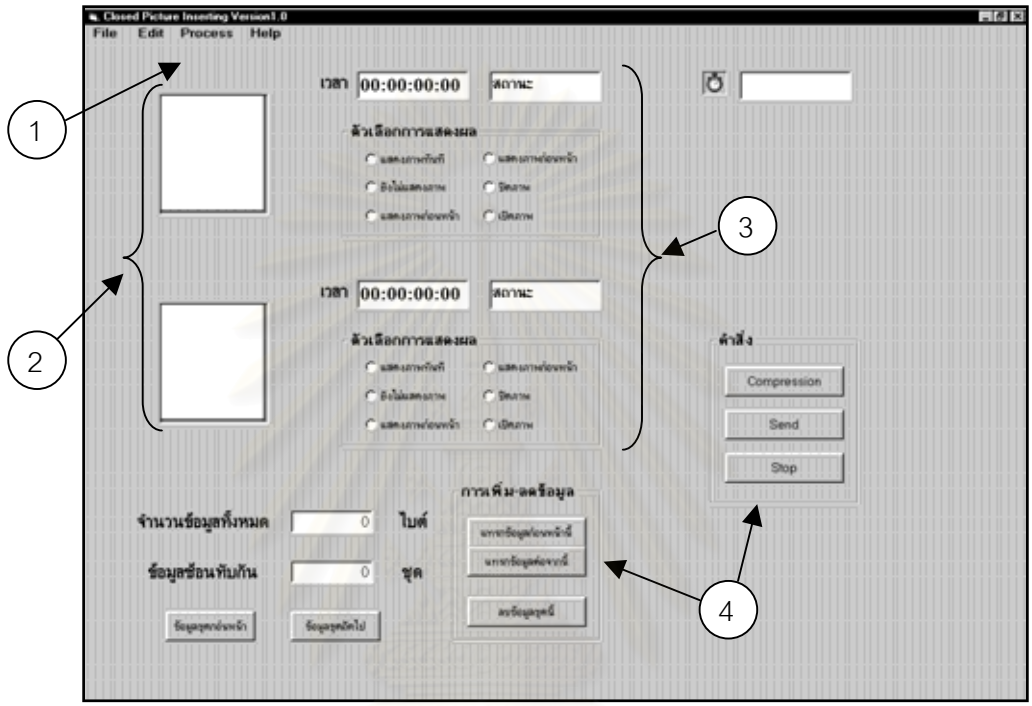
6.2 ขั้นตอนการพัฒนาโปรแกรมแทรกภาพบรรยาย

การพัฒนาโปรแกรมจะกำหนดความสามารถของโปรแกรม ดังนี้

1. สามารถบีบอัดภาพขนาด 112 X 128 จุด ตามมาตรฐาน CCITT T.4 แบบ 2 มิติ
2. สามารถเพิ่มข้อมูลที่สำคัญ คือ ไบต์เริ่มต้น, ไบต์ควบคุม,ไบต์ขนาด และไบต์สิ้นสุด รวมกับข้อมูลที่ถูกบีบอัดแล้ว
3. สามารถแบ่งข้อมูลภาพที่ต้องการส่งออกเป็นส่วนย่อย ส่วนละ 48 ไบต์ และสามารถรับสัญญาณขอข้อมูลจากเครื่องแทรกข้อมูลภาพได้
4. สามารถส่งข้อมูลที่เตรียมผ่านทางพอร์ตอนุกรมด้วยอัตราข้อมูล 28800 bit/sec

6.3 การใช้งานโปรแกรมแทรกภาพบรรยาย

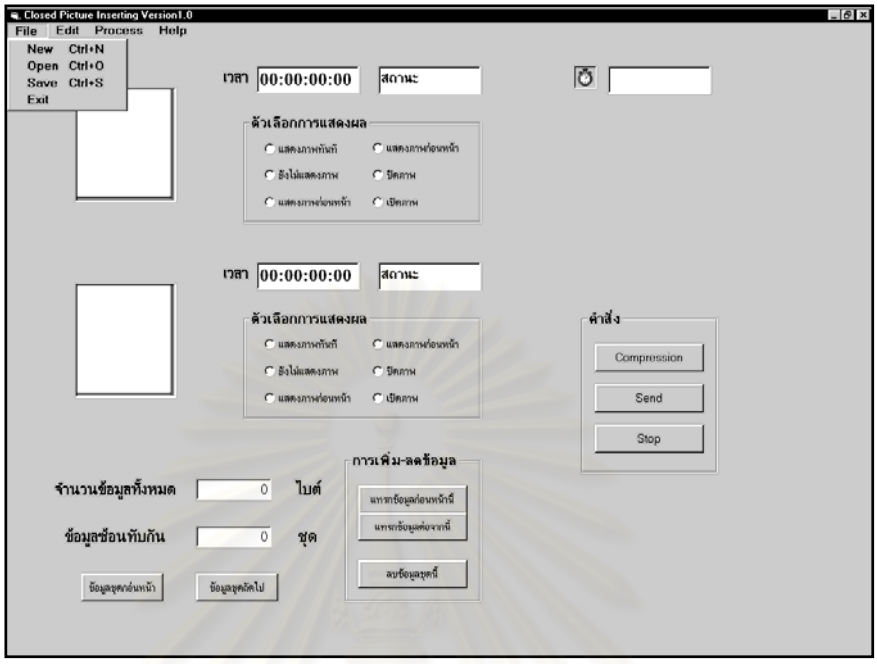
หน้าจอของโปรแกรมแทรกภาพบรรยายมีลักษณะ ดังแสดงในรูปที่ 6.1



รูปที่ 6.1 หน้าจอโปรแกรมแทรกภาพบรรยาย

หน้าจอโปรแกรมแทรกภาพบรรยาย แบ่งออกเป็น 4 ส่วนคือ

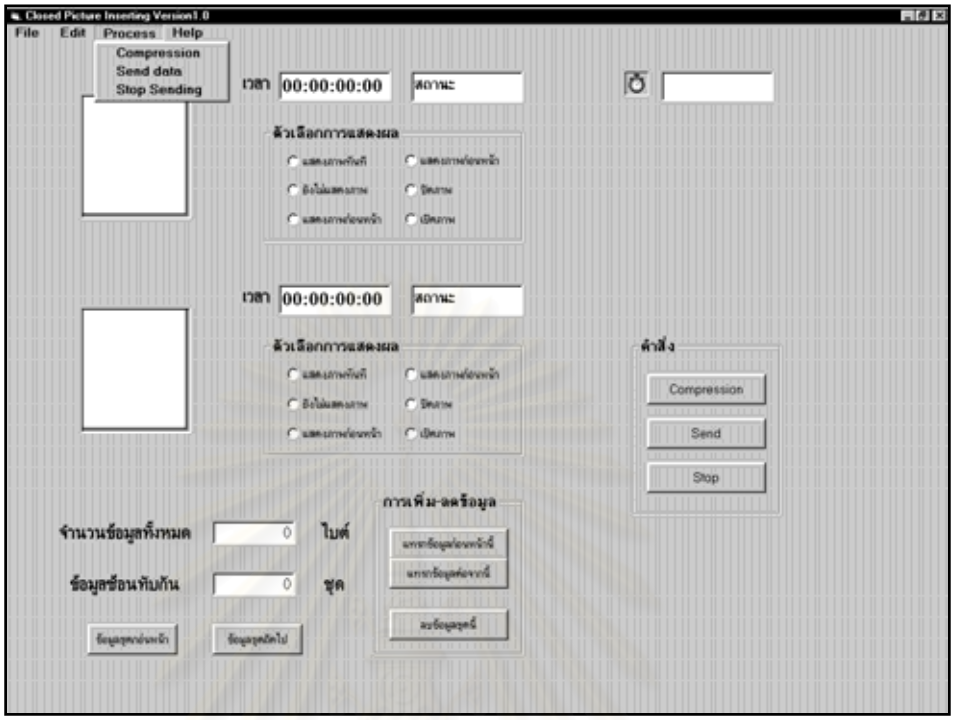
1. เมนู ประกอบด้วยเมนูย่อย 3 เมนูคือ
    1. File เป็นเมนูที่จัดการเกี่ยวกับแฟ้มโปรแกรม ดังแสดงในรูปที่ 6.2 ประกอบด้วยคำสั่งย่อย 4 คำสั่งดังนี้
        1) New ใช้สำหรับสร้างข้อมูลใหม่
        2) Open ใช้เปิดแฟ้มข้อมูลเก่า
        3) Save ใช้บันทึกข้อมูลภาพบรรยายที่สร้างขึ้นเก็บเป็นแฟ้มข้อมูล
        4) Exit ใช้ออกจากโปรแกรม



รูปที่ 6.2 เมนู File

2. Process เป็นเมนูที่จัดการเกี่ยวกับการทำงานของโปรแกรมกับภาพบรรยาย ดังแสดงในรูปที่ 6.3 ประกอบด้วยคำสั่งย่อย 3 คำสั่งคือ

1) Compression ใช้บีบอัดข้อมูลภาพบรรยายที่ต้องการส่ง พร้อมใส่ส่วนประกอบอื่นที่จำเป็น ได้แก่ ไบต์เริ่มต้น, ไบต์ขนาด, ไบต์ควบคุม และไบต์สิ้นสุด
2) Send data ใช้ส่งข้อมูลภาพบรรยายผ่านทางพอร์ตอนุกรม
3) Stop Sending ใช้ยกเลิกการส่งข้อมูลทางพอร์ตอนุกรม



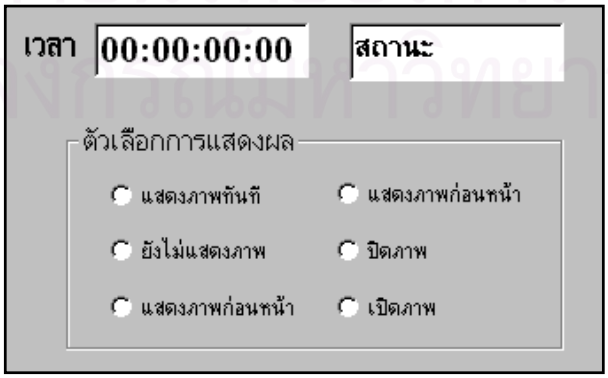
รูปที่ 6.3 เมนู Process

2. ส่วนแสดงภาพ

แสดงภาพบรรยายที่ต้องการแทรกลงในสัญญาณวีดิทัศน์ ครั้งละ 2 ภาพ คือ ภาพปัจจุบันและภาพถัดไป โดยภาพปัจจุบันแสดงอยู่ตำแหน่งบน และภาพถัดไปอยู่ตำแหน่งล่าง ดังแสดงในรูปที่ 6.1

3. ส่วนแสดงรายละเอียดของภาพบรรยาย

หน้าจอของส่วนนี้แสดงดังรูปที่ 6.4



รูปที่ 6.4 ส่วนแสดงรายละเอียดของภาพบรรยาย

ส่วนนี้จะแสดงข้อมูลที่ใช้ประกอบการแทรกภาพบรรยาย ได้แก่

1. เวลา ใช้ระบุเวลาที่ต้องการแทรกภาพลงในสัญญาณวีดิทัศน์ มีรูปแบบดังนี้

   HH:MM:SS:FF

   โดย HH คือ ชั่วโมง มีค่า 0 ถึง 23

   MM คือ นาที มีค่า 0 ถึง 59

   SS คือ วินาที มีค่า 0 ถึง 59

   FF คือ เฟรม มีค่า 0 ถึง 24

2. สถานะ ใช้บอกการซ้อนทับของภาพ 2 ภาพ ถ้าเวลาที่ระบุมีค่าใกล้กันมากเกินไป กล่าวคือ ภาพบรรยายก่อนหน้านี้ยังส่งข้อมูลไม่ครบ จึงไม่สามารถส่งข้อมูลภาพใหม่ได้ สามารถแก้ไขโดยเพิ่มเวลาที่ส่ง
3. ตัวเลือกการแสดงผล ใช้เลือกลักษณะการแสดงผล ดังแสดงในรูปที่ 6.4

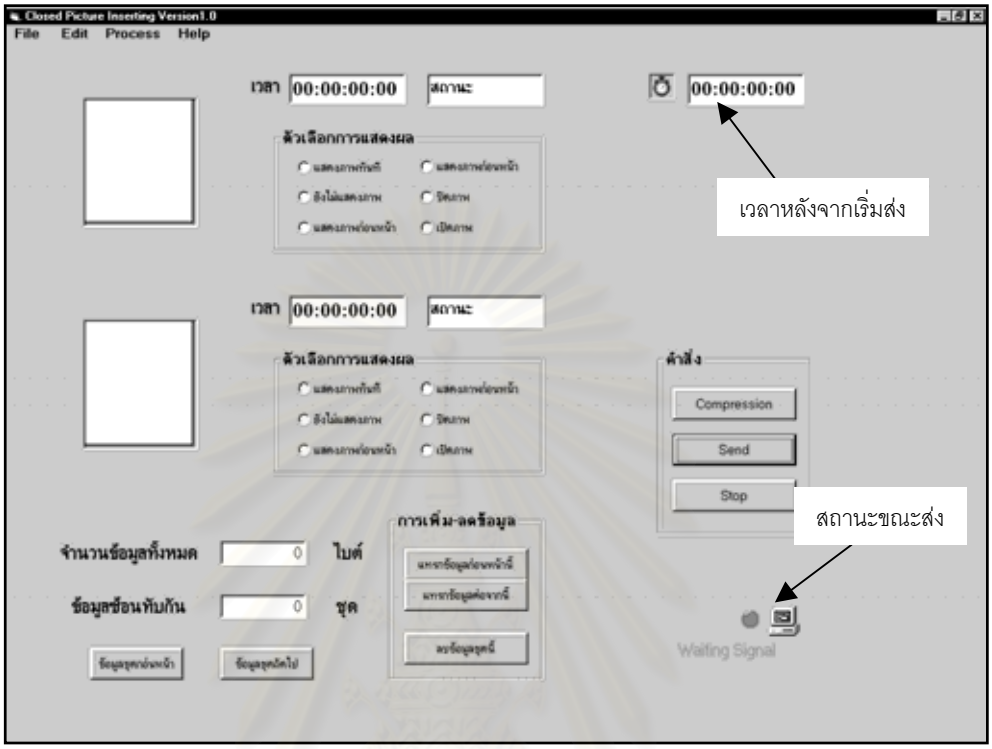
4. คำสั่ง

ใช้แสดงคำสั่งที่ใช้จากเมนู Edit และ Process เพื่อความสะดวกของผู้ใช้โปรแกรม แต่มีการเพิ่มคำสั่งอีก 2 คำสั่ง ได้แก่ คำสั่งเลื่อนดูภาพบรรยายก่อนหน้า และคำสั่งเลื่อนดูภาพบรรยายถัดไป

นอกจากนี้บนหน้าจอโปรแกรมยังประกอบด้วยส่วนสำคัญอื่นๆ ซึ่งจะแสดงเมื่อใช้คำสั่งบางคำสั่ง ได้แก่ คำสั่ง Send data เมื่อกดส่งข้อมูลทางพอร์ตอนุกรม จะปรากฏหน้าจอ ดังแสดงในรูปที่ 6.5

โปรแกรมจะบอกสถานะของการส่งตลอดเวลาระหว่างเริ่มส่งจนการส่งข้อมูลสิ้นสุด โดยประกอบด้วย 3 สถานะ

1. Waiting Signal ข้อความนี้จะขึ้นหลังการส่งข้อมูลทันที หากเครื่องแทรกข้อมูลภาพยังไม่เปิดหรือสายหลุด และเวลาที่ส่งจะมีค่าเป็น 00:00:00:00
2. Sending Process บอกให้ผู้ใช้ทราบว่ากำลังส่งข้อมูลให้กับเครื่องแทรกข้อมูลภาพ
3. No Signal ข้อความนี้เกิดขึ้นขณะที่ส่งข้อมูล โดยบอกให้ผู้ใช้ทราบว่าสายพอร์ตอนุกรมหลุด หรือ เครื่องแทรกข้อมูลภาพหยุดทำงาน และเวลาที่แสดงจะหยุดเดิน



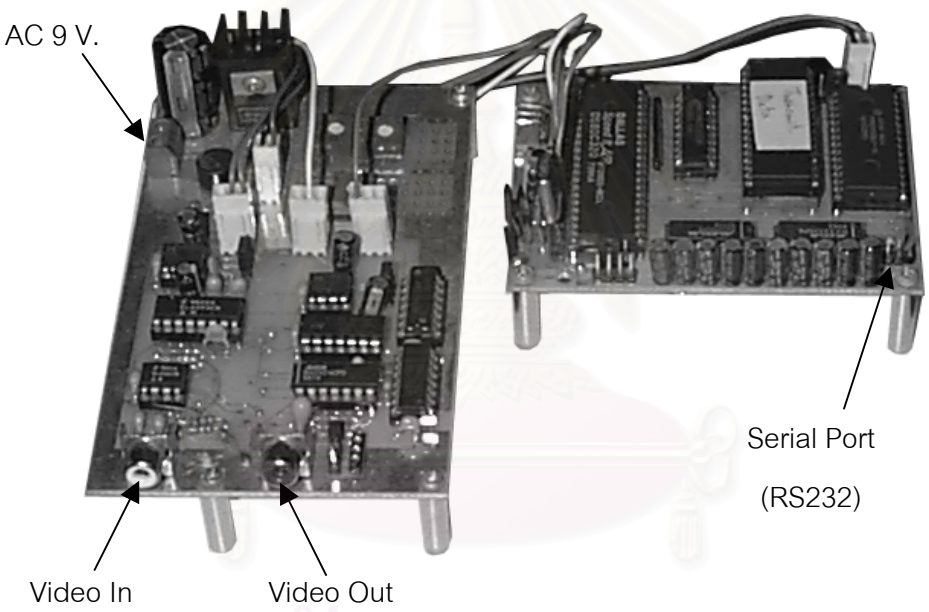
รูปที่ 6.5 หน้าจอโปรแกรมเมื่อส่งข้อมูลทางพอร์ตอนุกรม

บทที่ 7

การทดสอบและสรุปผล

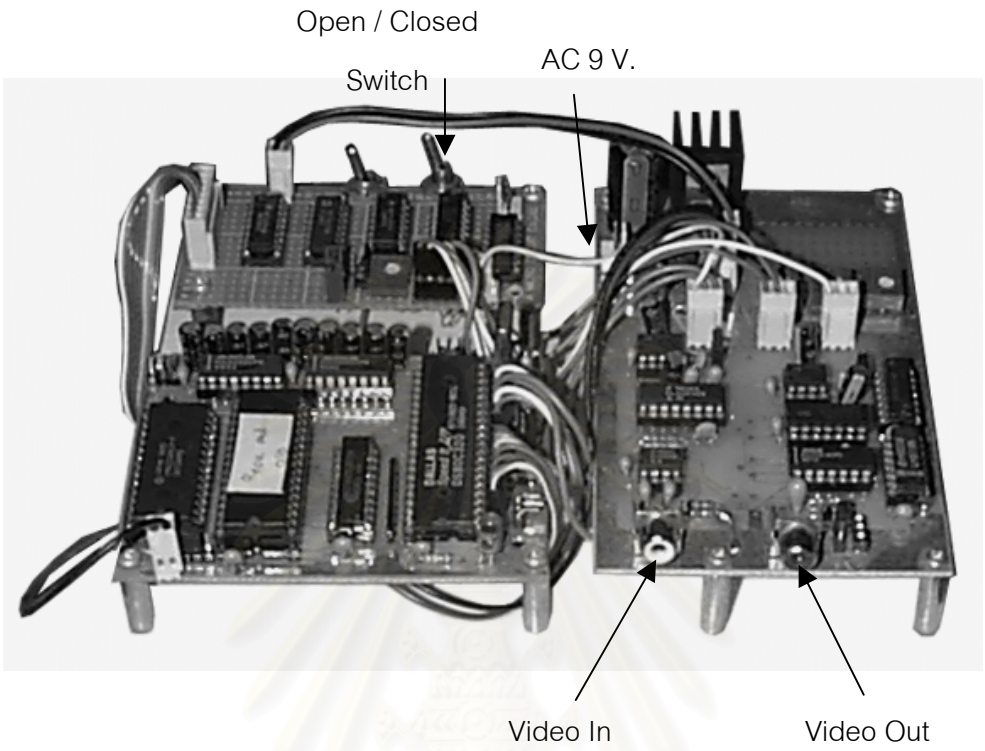
7.1 การทดสอบการทำงาน

เมื่อออกแบบระบบแทรกภาพบรรยายเรียบร้อยแล้ว ได้ประกอบเครื่องแทรกข้อมูลภาพดังแสดงในรูปที่ 7.1 เครื่องรับข้อมูลภาพดังแสดงในรูปที่ 7.2 และเขียนโปรแกรมควบคุมการทำงานของเครื่องแทรกและเครื่องรับข้อมูลภาพ

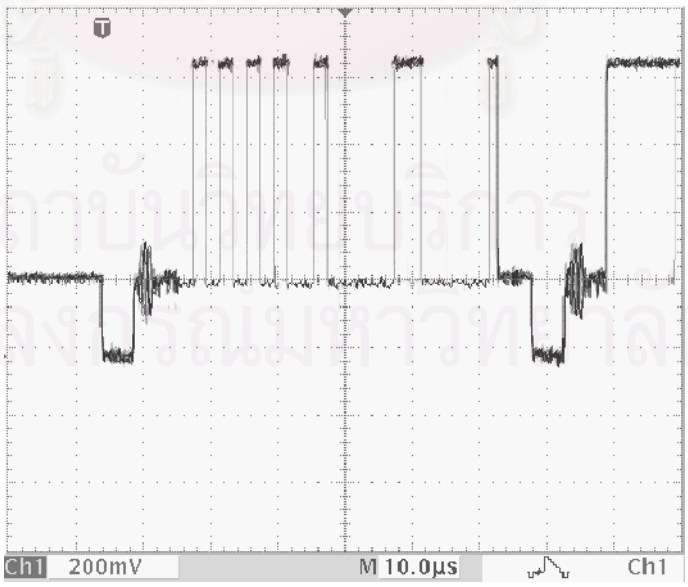


รูปที่ 7.1 บอร์ดต้นแบบเครื่องแทรกข้อมูลภาพ

เมื่อทดสอบการทำงานของเครื่องแทรกข้อมูลภาพ โดยต่อเครื่องแทรกข้อมูลภาพกับเครื่องคอมพิวเตอร์ส่วนบุคคลและให้สัญญาณวีดิทัศน์ที่ช่องสัญญาณ Video In ของเครื่องแทรกข้อมูลภาพ เพื่อแทรกข้อมูลลงในช่วงไร้ภาพแนวตั้งของสัญญาณวีดิทัศน์เส้นที่ 7-22 จะได้ลักษณะสัญญาณวีดิทัศน์ ดังแสดงในรูปที่ 7.3



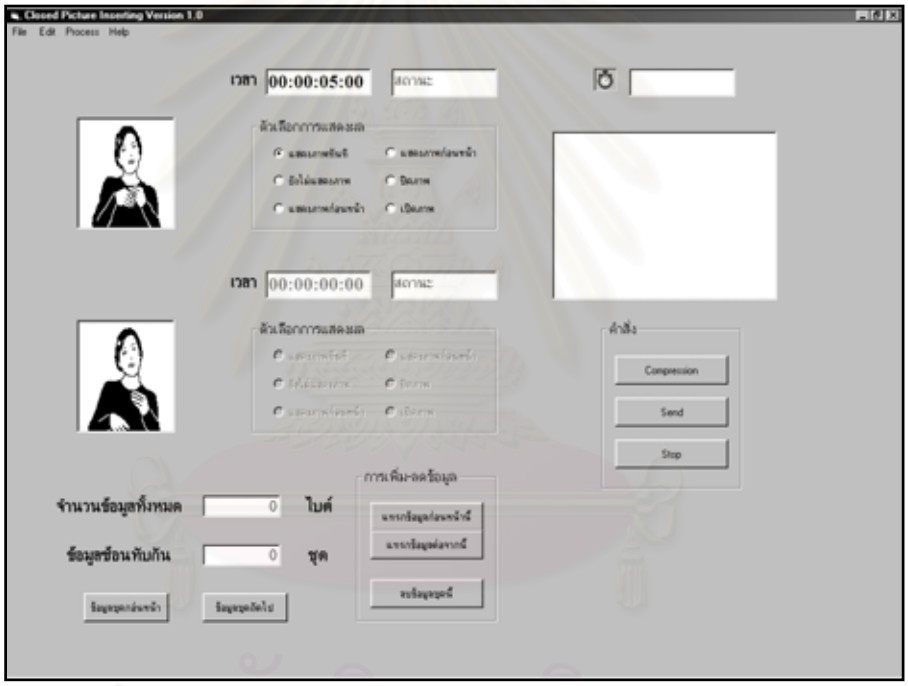
รูปที่ 7.2 บอร์ดต้นแบบเครื่องแทรกข้อมูลภาพ



รูปที่ 7.3 สัญญาณวีดิทัศน์ที่แทรกข้อมูลภาพ

การแทรกข้อมูลภาพใช้โปรแกรมแทรกข้อมูลภาพบนเครื่องคอมพิวเตอร์ส่วนบุคคล ซึ่งมีหน้าจอ ดังแสดงในรูปที่ 7.4 ภาพที่ต้องการแทรกจะถูกเปิดขึ้นมาที่โปรแกรมและบีบอัดข้อมูลโดยการเลือกเมนู Process แล้วเลือก Compression หรือ กดปุ่ม Compression บนหน้าจอ จากนั้นสามารถส่งข้อมูลที่บีบอัดโดยเลือกเมนู Process แล้วเลือก Send หรือ กดปุ่ม Send ข้อมูลจะถูกส่งไปที่เครื่องแทรกข้อมูลภาพเพื่อแทรกข้อมูลภาพลงในสัญญาณวีดิทัศน์

เครื่องรับข้อมูลภาพจะนำสัญญาณวีดิทัศน์ที่ได้จากเครื่องแทรกข้อมูลภาพมาคลายข้อมูลและแสดงภาพบนจอโทรทัศน์ ดังแสดงในรูปที่ 7.5



รูปที่ 7.4 การเตรียมภาพบรรยายบนโปรแกรมแทรกภาพบรรยาย

รูปที่ 7.5 หน้าจอโทรทัศน์เมื่อเครื่องรับข้อมูลภาพคลายข้อมูลและแสดงผลบนจอโทรทัศน์

7.2 ปัญหาในการทำงาน

1. ความเร็วของไมโครคอนโทรลเลอร์ในเครื่องรับข้อมูลภาพมีความเร็วไม่มากทำให้การเขียนโปรแกรมคลายข้อมูลต้องคำนึงถึงเวลาที่ใช้ จึงต้องปรับปรุงโปรแกรมคลายข้อมูลหลายครั้ง
2. ไมโครคอนโทรลเลอร์ในเครื่องแทรกข้อมูลไม่สามารถสร้างอัตรารับข้อมูลผ่านทางพอร์ตอนุกรมได้ตรงกับอัตราส่งของเครื่องคอมพิวเตอร์ส่วนบุคคลได้ จึงต้องใช้อัตราข้อมูลที่ใกล้เคียงในการส่ง

7.3 สรุป

1. โปรแกรมแทรกภาพบรรยายสามารถบีบอัดข้อมูลตามมาตรฐาน CCITT T.4 แบบ 2 มิติ และส่งข้อมูลให้เครื่องแทรกข้อมูลภาพด้วยอัตราข้อมูล 28800 bit/sec ได้
2. เครื่องแทรกข้อมูลภาพที่ออกแบบสามารถแทรกข้อมูลลงในช่วงไร้ภาพแนวตั้งเส้นที่ 7-22 ของสัญญาณวีดิทัศน์ ด้วยอัตราข้อมูล 500 kbit/sec โดยผู้ใช้สามารถแทรกข้อมูลภาพได้ด้วยโปรแกรมแทรกภาพบรรยายบนเครื่องคอมพิวเตอร์ส่วนบุคคล
3. เครื่องรับข้อมูลภาพสามารถนำข้อมูลจากเส้นสัญญาณวีดิทัศน์เส้นที่ 7 - 22 มาคลายข้อมูลและแสดงผลบนจอโทรทัศน์ โดยภาพบรรยายสามารถเปิด/ปิดได้ตามความต้องการของผู้ชม

7.4 ข้อเสนอแนะ

1. การแสดงผลบนจอโทรทัศน์ของเครื่องรับข้อมูลภาพสามารถปรับปรุงให้แสดงได้หลายตำแหน่งบนจอโทรทัศน์ได้
2. โปรแกรมแทรกภาพบรรยายบนเครื่องคอมพิวเตอร์ส่วนบุคคลสามารถปรับปรุงเพิ่มความสามารถในการส่งข้อมูล เช่น เลือกส่งข้อมูลแบบอื่น เป็นต้น
3. ข้อมูลที่ส่งควรมีการตรวจสอบข้อผิดพลาดโดยเพิ่มรหัสวัดความผิดพลาด (Error detecting code) และรหัสแก้ข้อผิดพลาด (Error correcting code) เพราะข้อมูลผิดพลาด 1 บิตจะทำให้ข้อมูลภาพเสียไปทั้งภาพ
4. การเพิ่มจำนวนภาพบรรยายที่แสดงใน 1 เฟรมสามารถทำได้โดยลดจำนวนข้อมูลภาพที่บีบอัด เช่น ใช้การบีบอัดที่อ้างอิงภาพก่อนหน้าเพราะภาพที่ส่งมีความต่อเนื่องกัน แต่การบีบอัดข้อมูลได้มากจะทำให้ไมโครคอนโทรลเลอร์ที่ใช้ต้องทำงานเร็วขึ้นจึงควรใช้ไมโครคอนโทรลเลอร์ 16 บิต หรือไมโครคอนโทรลเลอร์ที่ทำงานได้เร็วกว่าในวิทยานิพนธ์นี้
5. การแทรกภาพบรรยายอาจแทรกแบบเวลาจริง โดยใช้กล้องถ่ายภาพคนทำท่าภาษามือ และใช้โปรแกรมแยกภาพเป็นเฟรมย่อยๆ แล้วแปลงเป็นภาพขาวดำเพื่อบีบอัดและส่งให้เครื่องแทรกในเวลาจริง

รายการอ้างอิง

1. K. Blair Benson, Jerry Whitaker. Television Engineering Handbook. (n.p.), 1992.
2. Report and order on GEN docket No. 91-1, Federal Communications commission United States of Americe, 1991.
3. อัษฎางค์ แทนสถิตย์. การพัฒนาทีวีไมโครคอนโทรลเลอร์ที่สามารถถอดรหัสคำบรรยายภาพไทย-อังกฤษแบบซ่อนได้. ปริญญาวิศวกรรมศาสตรมหาบัณฑิต ภาควิชาวิศวกรรมไฟฟ้า คณะวิศวกรรมศาสตร์ จุฬาลงกรณ์มหาวิทยาลัย, 2541.
4. ETS 300 731 - Enhanced 625-line Phased Alternate line (PAL) television : PALplus. ETSI, 1997.
5. ITU-T. Facsimile coding schemes and coding control functions for group 4 fasimile appartus, 1988.
6. Rafael C. Gonzalez, Richard E. Woods. Digital Image Processing : Addison Wesley, 1993.
7. Dallas Semicomductor. DS80C320/DS80C323 High-Speed/Low-Power Micro, 1998.
8. Michael Tischer. PC Intern System Programing : Abacus, 1995.

ภาคผนวก

ภาคผนวก ก

บทความตีพิมพ์ใน การประชุมวิชาการทางวิศวกรรมไฟฟ้าครั้งที่ 23
(23rd Electrical Engineering Conference)

ระบบวีดีโอที่ซ่อนภาพบรรยายได้แบบราคาประหยัด

A Low Cost Closed Picture CaptionVideo System

นันทกฤษณ์ วัฒนเลี้ยงใจ และ เอกชัย ลีลารัศมี
ภาควิชาวิศวกรรมไฟฟ้าคณะวิศวกรรมศาสตร์ จุฬาลงกรณ์มหาวิทยาลัย
ถนน พญาไท กรุงเทพฯ 10330
โทร (02)2186537 E-mail : b0367021@student.chula.ac.th

บทคัดย่อ

บทความนี้นำเสนอระบบแทรกภาพในสัญญาณวีดีโอระบบ PAL ที่ผู้รับชมสามารถจะปิดหรือเปิดภาพที่แทรกเข้ามาได้ตามความต้องการ โดยภาพที่แทรกเป็นภาพขาวดำ ขนาด 112 X 128 จุด ที่แปลงเป็นข้อมูลและผ่านกระบวนการบีบอัดข้อมูลตามมาตรฐาน CCITT T.4 2-D. ก่อนจะนำไปแทรกไว้ในช่วงไร้ภาพแนวตั้ง ระบบโดยรวมประกอบด้วยเครื่องแทรกข้อมูลภาพ และ เครื่องรับข้อมูลภาพ โดยเครื่องแทรกจะแทรกข้อมูลภาพด้วยความถี่ของข้อมูล 500 kbit/sec ทำให้สามารถบันทึกข้อมูลลงในเทปได้ เครื่องแทรกและเครื่องรับข้อมูลภาพนี้พัฒนาขึ้นสามารถนำไปใช้ประโยชน์ในการแทรกภาพภาษามือลงในสัญญาณวีดีโอ

คำสำคัญ : ระบบ PAL, สัญญาณวีดีโอ, การบีบอัดข้อมูล

Abstract

A PAL video system for inserting small pictures is presented. The viewer will be able to close or open a picture with a resolution of 112 X 128 dots. Each picture data are compressed by using CCITT T.4 2-D algorithms before being inserted in the vertical blanking interval at the rate of 500 kbit/sec. The developed hardware consists of an inserter and a decoder and could be used for inserting a sign language caption in the video signal.

Keyword : Pal Video System, Video Signal, Compression

1. บทนำ

ปัจจุบันคนพิการให้ความสนใจในการรับชมสื่อทางโทรทัศน์มากขึ้น การเพิ่มบริการเพื่ออำนวยความสะดวกแก่คนเหล่านี้จึงมีมากขึ้น เช่น Closed Caption [1], [2] ที่ช่วยให้คนหูหนวกเข้าใจสิ่งที่รับชมได้ดีขึ้น แต่คนหูหนวกส่วนใหญ่จะคุ้นเคยกับการใช้ภาษามือมากกว่าคำบรรยายจึงมีการนำระบบ Closed Caption มาพัฒนาเป็นระบบแทรกภาพ ซึ่งแทรกข้อมูลภาพลงในช่วงไร้ภาพแนวตั้งของสัญญาณวีดีโอระบบ PAL เหมือนกับ ระบบ Closed Caption และ ระบบ Teletext [1], [3] แต่เนื่องจากภาพ 1 ภาพจะใช้ข้อมูลในการส่งมากกว่า Closed Caption จึงต้องบีบอัดข้อมูลและใช้เส้นสัญญาณในช่วงไร้ภาพแนวตั้งมากกว่าการส่งข้อมูลของ Closed Caption ภาพที่แทรกเป็นภาพ Bitmap ขาวดำขนาด 112 X 128 จุด ซึ่งใช้เนื้อที่ประมาณ 1 ใน 25 ส่วนของจอภาพทั้งหมด

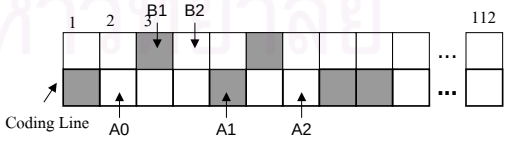
2.การบีบอัดและการคลายข้อมูล

2.1 มาตรฐานการบีบอัดข้อมูลแบบ CCITT T.4 2-D

การบีบอัดข้อมูลแบบ CCITT T.4 2-D [4] เป็นการบีบอัดที่ใช้บีบอัดข้อมูล Fax โดยลักษณะข้อมูลมีเพียงสีดำและสีขาว และมีความต่อเนื่องของลายเส้น ซึ่งมีคุณสมบัติเช่นเดียวภาพที่ต้องการส่ง และการคลายข้อมูลมีกระบวนการที่ไม่ซับซ้อนทำให้สามารถใช้ไมโครคอนโทรลเลอร์ 8 บิตซึ่งมีราคาถูกในการคลายข้อมูลได้ จึงนำมาตรฐานนี้มาใช้บีบอัดข้อมูลภาพ

การบีบอัดจะพิจารณาจุดเปลี่ยนสีเป็นหลัก และพิจารณาว่าจุดเปลี่ยนสีของแถวที่เข้ารหัสมีความสัมพันธ์อย่างไรกับจุดเปลี่ยนสีของแถวอ้างอิง โดยแถวอ้างอิง หมายถึง แถวที่อยู่ด้านบนของแถวที่กำลังเข้ารหัส สำหรับแถวอ้างอิงของแถวแรกจะแทนด้วยแถวที่มีจุดสีขาวทุกจุด

ในการบีบอัดจะมีจุดอ้างอิงที่สำคัญ 5 จุด [4] คือ



รูปที่ 1 จุดเปลี่ยนสีที่ใช้ในการเข้ารหัส

A0 : ตำแหน่งอ้างอิงหรือจุดที่เริ่มการเปลี่ยนสีของแถวที่เข้ารหัส และ A0 จะเลื่อนตำแหน่งไปตามการบีบอัดแบบต่างๆ

A1 : จุดเปลี่ยนสีถัดมาทางขวาของ A0 บนแถวที่เข้ารหัส

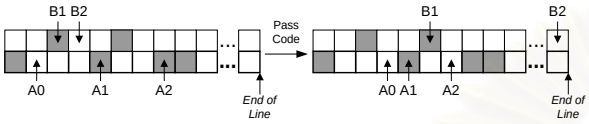
A2 : จุดเปลี่ยนสีถัดมาทางขวาของ A1 บนแถวที่เข้ารหัส

B1 : จุดเปลี่ยนสีจุดแรกทางขวาของ A0 บนแถวอ้างอิงและมีสีตรงข้ามกับ A0

B2 : จุดเปลี่ยนสีถัดมาทางขวาของ B1 ของแถวอ้างอิง

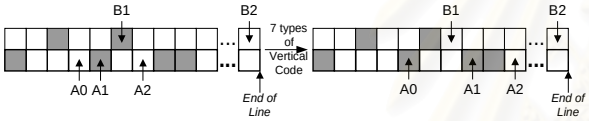
โมดการบีบอัดมี 3 แบบคือ

2.2.1 Pass mode เกิดขึ้นเมื่อ B2 อยู่ทางซ้ายของ A1 และหลังเข้ารหัส A0 จะย้ายไปที่ตำแหน่งตรงกับ B2 ของแถวที่เข้ารหัสและให้รหัส 0001



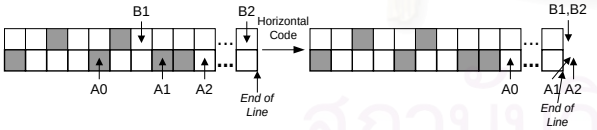
รูปที่ 2 Pass mode

2.2.2 Vertical mode เกิดขึ้นเมื่อระยะห่างของ A1 และ B1 มีค่าไม่เกิน 3 และหลังเข้ารหัส A0 จะย้ายไปที่ตำแหน่ง A1 ของแถวที่เข้ารหัสและให้รหัส 7 แบบขึ้นกับระยะห่างระหว่าง A1 กับ B1



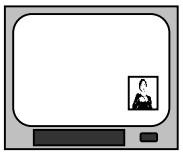
รูปที่ 3 Vertical mode

2.2.3 Horizontal mode เกิดขึ้นเมื่อการเปลี่ยนแปลงจุดสีไม่อยู่ในกรณีของ 2 แบบข้างต้น และหลังเข้ารหัส A0 จะย้ายไปที่ตำแหน่ง A2 ของแถวที่เข้ารหัสและให้รหัส 001 + M(A1-A0) + M(A2-A1) เมื่อ M(A1-A0) และ M(A2-A1) คือรหัสแทนความยาวและสีของระยะห่างระหว่าง A0 กับA1 และ A1 กับ A2 ตามลำดับ



รูปที่ 4 Horizontal mode

ภาพที่แสดงบนจอโทรทัศน์จะใช้จำนวนเส้นสัญญาณวีดีโอทั้งหมด 64 เส้นใน 1 ฟิลด์และจำนวนจุดในแนวนอน 112 จุดโดยแต่ละจุดใช้เวลา 83 nS ดังนั้นภาพที่แทรกจะใช้เนื้อที่ประมาณ 1 ใน 25 ของจอภาพ ดังแสดงในรูปที่ 5



รูปที่ 5 ลักษณะภาพที่แสดงบนจอภาพ

ภาพ 1 ภาพที่แทรกมีขนาด 112 X 128 จุด คิดเป็นข้อมูลทั้งหมดจำนวน 1792 ไบต์ เมื่อบีบอัดจะมีขนาดลดลงประมาณ 6 เท่า เหลือประมาณ 300 ไบต์ แต่ถ้าภาพซับซ้อนมาก เช่น ภาพจุดตารางหมากรุกที่มีการเปลี่ยนสีทุกจุดสี จะได้ขนาดเพิ่มขึ้นประมาณ 2 เท่า ซึ่งกรณีนี้จะไม่เกิดในชิ้นเพราะภาพที่ส่งเป็นภาพลายเส้นดำที่มีความต่อเนื่อง การบีบอัดภาพ 1 ภาพใช้เวลาส่งลดลงเหลือประมาณ 4 เฟรม ทำให้ได้ภาพต่อเนื่อง 6 ภาพต่อ 1 วินาที ซึ่งถือว่าเป็นภาพเคลื่อนไหวที่ใช้ได้

2.2 การคลายข้อมูลในเครื่องรับ

การคลายข้อมูลจะมีจุดอ้างอิงที่สำคัญ 3 จุด ดังนี้

A0 : จุดอ้างอิงบอกตำแหน่งของจุดที่ได้คลายข้อมูลแล้ว

B1 : จุดที่มีสีตรงข้ามกับ A0 จุดแรกที่อยู่ทางขวาของ A0 ในแถวอ้างอิง

B2 : จุดเปลี่ยนสีถัดจาก B1 บนเส้นอ้างอิง

ข้อมูลภาพที่ถูกบีบอัดจะถูกนำมาพิจารณาทีละ 1 บิต เพื่อหาลักษณะการบีบอัดข้อมูล ซึ่งมีผลดังตารางนี้

ลักษณะข้อมูล	ลักษณะการคลายข้อมูล		
	สี	จำนวนจุด	ตำแหน่ง A0
1	เหมือน B1	B1 - A0	B1
011	เหมือน B1	B1 - A0 +1	B1 +1
000011	เหมือน B1	B1 - A0 +2	B1 + 2
0000011	เหมือน B1	B1 - A0 +3	B1 + 3
010	เหมือน B1	B1 - A0 - 1	B1 - 1
000010	เหมือน B1	B1 - A0 - 2	B1 - 2
0000010	เหมือน B1	B1 - A0 - 3	B1 - 3
001 *	ขึ้นกับรหัส	ขึ้นกับรหัส	ขึ้นกับรหัส
0001	เหมือน B2	B2 - A0	B2
0000001	สิ้นสุดการคลายข้อมูลภาพ		

* <u>หมายเหตุ</u> รหัส 001 คือ Horizontal mode การคลายข้อมูลจะใช้รหัสต่อท้ายที่จะสร้างจุดสีขาว - ดำ หรือ ดำ - ขาว ต่อกัน จึงมีสีและตำแหน่ง A0 ไม่แน่นอน

ตารางที่ 1 ผลการคลายข้อมูล

3. ข้อมูลภาพที่ส่ง

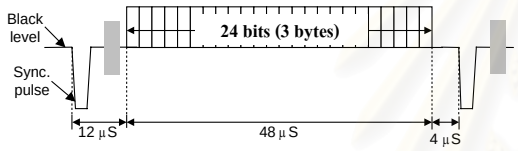
ภาพบรรยาย 1 ภาพจะต้องนำข้อมูลมาเพิ่มส่วนหัวและ ส่วนท้ายเฟรมก่อน ซึ่งมีรายละเอียด ดังแสดงในรูปที่ 6

Start				Stop
00000011	Control	Size(n)	Compressed Data	00000000
1 Byte	1 Byte	2 Bytes	n Bytes	1 Byte

รูปที่ 6 ลักษณะข้อมูลที่จะส่งเมื่อเพิ่มข้อมูลแล้ว

ส่วนหัวของเฟรมประกอบด้วย Start Byte (00000011) ขนาด 1 ไบต์ทำหน้าที่บอกเครื่องรับว่าข้อมูลถัดจากนี้เป็นข้อมูลของภาพใหม่, Control Byte ขนาด 1 ไบต์บอกคำสั่งที่จะทำหลังคลายข้อมูลเสร็จ เช่น สั่งให้แสดงภาพเมื่อคลายข้อมูลเสร็จ, Size Byte ขนาด 2 ไบต์บอกขนาดข้อมูลที่ส่งว่ามีขนาดเท่าไร และ Stop Byte (00000000) ขนาด 1 ไบต์ใช้บอกการสิ้นสุดของข้อมูลเพื่อให้เครื่องรับเตรียมเนื้อที่ในหน่วยความจำไว้เก็บข้อมูลภาพถัดไป

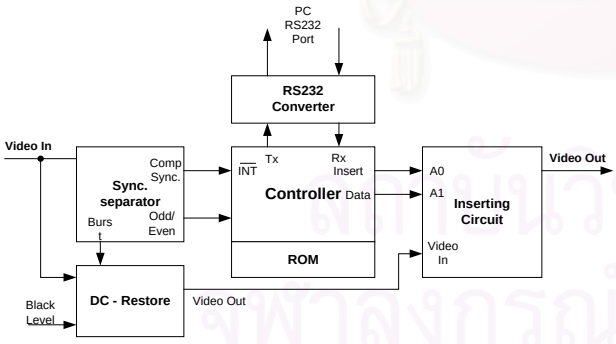
ข้อมูลภาพที่บีบอัดแล้วจะถูกส่งจากเครื่องคอมพิวเตอร์ส่วนบุคคลมาแทรกที่เส้นสัญญาณช่วงเส้นที่ 7 – 22 ข้อมูลจะถูกแทรกหลังจากเกิดสัญญาณซิงก์แนวราบ 12 μS ข้อมูลจะแทรกด้วยความถี่ข้อมูล 500 kbit/sec เพื่อให้สามารถบันทึกข้อมูลลงในวีดีโอเทปได้ ดังนั้นใน 1 เส้นสัญญาณวีดีโอสามารถแทรกข้อมูลได้ 3 ไบต์ ดังแสดงในรูปที่ 7



รูปที่ 7 ข้อมูลภาพที่แทรกลงในสัญญาณวีดีโอ

4. โครงสร้างและการทำงานของเครื่องแทรกข้อมูลภาพ

เครื่องแทรกข้อมูลภาพควบคุมด้วยไมโครคอนโทรลเลอร์ 8 บิต ตระกูล 8051 และประกอบด้วยส่วนต่างๆ ดังแสดงในรูปที่ 8



รูปที่ 8 โครงสร้างของเครื่องแทรกข้อมูลภาพ

วงจรหลักอื่นๆของเครื่องแทรกประกอบด้วย

4.1 วงจร RS 232 Converter ประกอบด้วย MAX232 ใช้แปลงข้อมูลที่ได้รับจากพอร์ตอนุกรมของเครื่องคอมพิวเตอร์ส่วนบุคคลและส่งต่อให้ไมโครคอนโทรลเลอร์เพื่อเก็บในหน่วยความจำ

4.2 วงจร Sync. Separator มีหน้าแยกสัญญาณ Composite Sync. และ Odd/Even ออกจากสัญญาณวีดีโอ ภายในวงจรประกอบด้วย LM1881 Video Sync Separator

4.3 วงจร DC-Restore Video Amplifier ประกอบด้วย EL4093C ที่ใช้รักษาระดับสีดำของสัญญาณวีดีโอให้คงที่ โดยสามารถตั้งระดับสีดำจากการเปลี่ยนค่าแรงดันอ้างอิงที่จ่ายให้กับ EL4093C

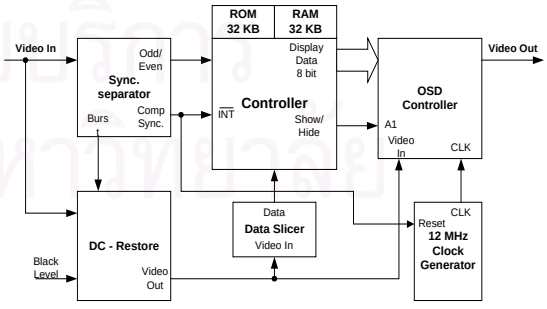
4.4 วงจร Insertering ภายในวงจรประกอบด้วย MAX454 CMOS Video Multiplexer / Amplifier เพื่อสั่งแทรกข้อมูลภาพ โดยสัญญาณ Select ใช้เลือกว่าต้องการแทรกข้อมูลหรือไม่ ส่วนสัญญาณ Data จะบอกระดับของข้อมูลที่แทรกว่าเป็นลอจิก ‘0’ หรือ ‘1’

ข้อมูลภาพทุกภาพจะถูกบีบอัดและเก็บไว้ในโปรแกรมในคอมพิวเตอร์ส่วนบุคคลก่อนส่งให้เครื่องแทรกครั้งละ 48 ไบต์ผ่านทางพอร์ตอนุกรม โดยการส่งครั้งแรกจะมีสัญญาณจากเครื่องแทรกไปที่คอมพิวเตอร์เพื่อขอให้ส่งข้อมูลและเครื่องแทรกจะเก็บข้อมูลให้เสร็จภายในฟิลด์นั้น และรอจนถึงเส้นที่ 6 ถึง 22 ของฟิลด์ถัดไป ไมโครคอนโทรลเลอร์จะส่งสัญญาณ Select และ Data ให้กับวงจร Data Inserter เพื่อแทรกข้อมูลภาพลงในสัญญาณวีดีโอ จากนั้นจะรอเส้นสัญญาณที่ 23 และส่งสัญญาณไปขอข้อมูลชุดต่อไป ดังสรุปการทำงานด้านล่าง

เส้นที่ 7 ถึง 22	แทรกข้อมูล 3 ไบต์ลงในสัญญาณวีดีโอ
เส้นที่ 24	ส่งสัญญาณขอข้อมูลไปยัง PC
เส้นที่ 25	เก็บข้อมูลลงในหน่วยความจำ

5.ต้นแบบของเครื่องรับสัญญาณภาพ

เครื่องรับสัญญาณภาพจะควบคุมการทำงานด้วยไมโครคอนโทรลเลอร์ 8051 และประกอบด้วยส่วนต่างๆ ดังแสดงในรูปที่ 9

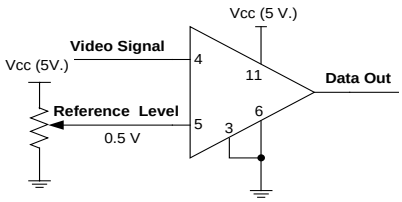


รูปที่ 9 โครงสร้างของเครื่องรับข้อมูลภาพ

เครื่องรับสัญญาณภาพจะประกอบด้วยวงจรหลักอื่นๆ คือ

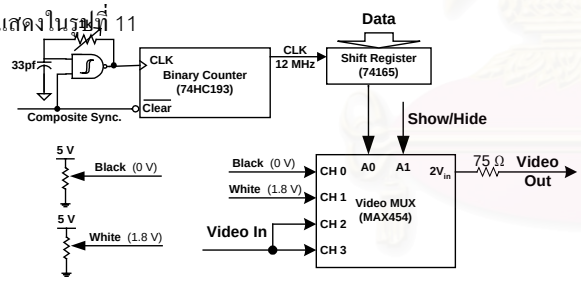
5.1 วงจร Video Clamp, DC Restore และ Data Slicer ภายในวงจรประกอบด้วยโครงสร้างคล้ายกับวงจรของเครื่องแทรกแต่จะมี

ส่วนของ Data Slicer เพื่อใช้แยกข้อมูลภาพที่แทรกออกจากสัญญาณวีดีโอและส่งให้ไมโครคอนโทรลเลอร์นำไปเก็บในหน่วยความจำ มีวงจรดังแสดงในรูปที่ 10



รูปที่ 10 วงจร Data Slicer

5.2 วงจรกำเนิดสัญญาณนาฬิกาและวงจร OSD Controller มีหน้าที่แสดงภาพบรรยายบนจอภาพ โดยข้อมูลภาพจะส่งมาที่ Shift Register และเลื่อนข้อมูลออกครั้งละบิตด้วยความถี่ 12 MHz โดยสัญญาณนาฬิกามาจากวงจรกำเนิดสัญญาณนาฬิกาที่สร้างสัญญาณนาฬิกาความถี่ 24 MHz ให้กับ Binary Counter และต่อสัญญาณนาฬิกา 12 MHz จากขาสัญญาณ Qa ของ Binary Counter ภายในวงจรจะ Reset ด้วยสัญญาณ Composite Sync. เพื่อให้สัญญาณนาฬิกาเริ่มต้นพร้อมกันทุกเส้น การเลือกชมภาพที่แทรกบนจอภาพจะถูกกำหนดด้วยสัญญาณ Show/Hide จากไมโครคอนโทรลเลอร์ ซึ่งสัญญาณนี้จะต่อผ่านสวิตช์ที่ใช้เลือกรับชมภาพที่แทรก ข้อมูลจาก Shift Register จะเลือกว่าจุดที่แสดงคือจุดสีขาวหรือสีดำ วงจร OSD Controller มีรายละเอียด ดังแสดงในรูปที่ 11



รูปที่ 11 วงจรสร้างสัญญาณนาฬิกาและวงจร OSD Controller

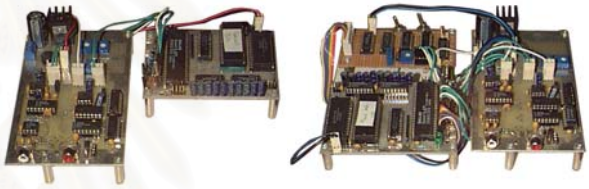
ไมโครคอนโทรลเลอร์จะรับสัญญาณ Interrupt จากวงจร Sync. Separater เพื่อนับเส้นสัญญาณวีดีโอ เมื่อถึงเส้นที่ 7 ถึง 22 ของแต่ละฟิลด์ ไมโครคอนโทรลเลอร์จะแยกข้อมูลจำนวน 3 ไบต์ในแต่ละเส้นสัญญาณวีดีโอด้วยวงจร Data Slicer และนำไปเก็บในหน่วยความจำภายในเพื่อรอการคลายข้อมูล เมื่อถึงเส้นที่ 23 ไมโครคอนโทรลเลอร์จะคลายข้อมูลที่เก็บไว้และนำภาพบรรยายที่คลายได้เก็บที่หน่วยความจำภายนอก ซึ่งจะเก็บในหน่วยความจำ 2 แห่งสลับกัน คือ หน่วยความจำช่องที่1 จะเก็บภาพแรกที่แสดงบนจอภาพ ช่องที่ 2 จะเก็บข้อมูลภาพที่กำลังคลายในขณะที่ยังแสดงผลของช่องที่ 1 อยู่ เมื่อคลายข้อมูลในช่องที่ 2 เรียบร้อยแล้วจึงนำข้อมูลจากช่องที่ 2 แทรกบนจอภาพ และคลายข้อมูลลงในช่องที่ 1 สลับกันไปเรื่อยๆ การแสดงภาพบนจอภาพจะเริ่มแสดงที่เส้นสัญญาณที่ 205 ถึง 269 ในแต่ละฟิลด์ โดยจะใช้ข้อมูลจากหน่วยความจำที่เก็บภาพบรรยายครั้งละ 1 ไบต์และส่งไปให้วงจร OSD Controller เพื่อแสดงภาพบนจอภาพ

กระบวนการทำงานของเครื่องรับสรุปโดยรวมได้ดังนี้

เส้นที่ 7 ถึง 22	ดึงข้อมูล 3 ไบต์ มาเก็บไว้ในหน่วยความจำภายใน
เส้นที่ 23	เริ่มคลายข้อมูล 48 ไบต์ที่รับเข้ามา
เส้นที่ 205 ถึง269	แสดงภาพบนจอภาพ

6. ผลการดำเนินงาน

เมื่อออกแบบเครื่องแทรกและเครื่องรับข้อมูลภาพแล้ว ได้สร้างเครื่องต้นแบบขึ้นมาดังแสดงในรูปที่ 13



รูปที่ 13 เครื่องแทรกและเครื่องรับข้อมูลภาพ

ตัวอย่างภาพที่แทรกลงในสัญญาณวีดีโอมีขนาด 1792 ไบต์ เมื่อผ่านการบีบอัดจะมีขนาด 315 ไบต์ (บีบอัดได้ 5.68 เท่า) โดยใช้เวลาในการส่ง 4 เฟรม ข้อมูลนี้จะส่งไปยังเครื่องรับเพื่อคลายข้อมูลและแสดงผลที่จอภาพ ซึ่งจะเห็นภาพที่เราแทรกเข้าไปในสัญญาณวีดีโอปรากฏเป็นภาพเคลื่อนไหวขนาดเล็ก 6 ภาพต่อวินาทีบนจอโทรทัศน์ ดังแสดงในรูปที่ 14

รูปที่ 14 ภาพขนาดเล็กเมื่อนำมาแทรกบนจอโทรทัศน์

7. สรุป

บทความนี้ได้นำเสนอระบบการแทรกข้อมูลภาพในช่วงไร้ภาพแนวตั้ง ซึ่งภาพที่แทรกเป็นภาพขาวดำขนาด 112 X 128 จุดที่ทำการบีบอัดข้อมูลภาพตามมาตรฐาน CCITT T.4 2-D ก่อนที่จะแทรกลงไปในสัญญาณวีดีโอด้วยความถี่ 500 kHz ในการพัฒนาขั้นต้นนี้ได้พัฒนาให้

แสดงเฉพาะภาพต่อเนื่องขาวดำ ซึ่งมีประโยชน์ในการแทรกภาพภาษามือสำหรับคนพิการ

เอกสารอ้างอิง

[1] K.Blair Benson and Jerry Whitaker, "Television Engineering Handbook", 1992

[2] อัษฎางค์ แทนสถิตย์, "การพัฒนาทีวีไมโครคอนโทนลเลอร์ที่สามารถถอดรหัสคำบรรยายไทย – อังกฤษแบบซ่อนได้", 2541

[3] สายัณห์ ธีรปัญญาวัฒน์, "การพัฒนาเครื่องรับเทเลเท็กซ์แสดงผลตัวไทย/อังกฤษ", 2536

[4] ITU-T, "Facsimile coding schemes and coding control functions for group 4 fasimile appartus", 1988

ภาคผนวก ข

บทความตีพิมพ์ใน 2000 IEEE Asia-Pacific Conference On Circuit And System (Electronic Communication Systems)

A Video System For Transmitting Hidden Pictures Based On CCITT T.4 2-D Compression

Ekachai Leelarasmee and Nuntakrit Vattanaliangjai
Electrical Engineering Department, Chulalongkorn University
Phayathai Road, Bangkok, Thailand 10330
Tel. (662) 2186488 E-mail: ekachai.l@chula.ac.th

Abstract

The use of vertical blanking intervals in a PAL video signal for transmitting an animation of 112x128 dots of black and white picture is proposed. Data compression based on CCITT T.4 2-D standard and low data rate of 500 kbit/sec are chosen to make the hardware design of an inserter and decoder economically feasible by using an 8-bit microcontroller. The inserter is designed to hide 48 bytes of compressed data, generated by a personal computer, within lines 7 to 22 of a PAL video signal. The decoder can extract these data, decompress and display the hidden picture on the lower right position of a normal TV screen. The basic building blocks of these two devices will be described. This system can be used to provide a Sign Language Closed Caption service.

I. Introduction

Vertical blanking interval (VBI) [1] in each field of analog video systems, such as PAL and NTSC, is used primarily for picture frame synchronization. These empty lines can be used to transmitted hidden data, such as those found in Teletext [1] and Closed Caption [2], without interrupting normal picture viewing. Whereas most VBI data are intended for text and still image transmission, this work will propose the use of VBI for transmitting a small low-resolution animation picture. Each hidden picture consists of 112x128 dots of black or white color and is primarily intended for representing an image of a sign language as shown in Fig. 1

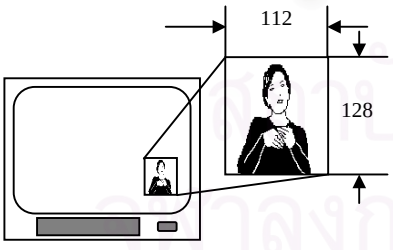


Fig. 1 An on screen display of the hidden picture

A prototype development of this modified video system will be described with special emphasis on a low volume and low cost implementation. This includes the selection of a compression technique, the design of a VBI data format, the implementation of a inserting and decoding hardware.

II. The CCITT T.4 2-D compression standard

For ease of hardware implementation, the CCITT T.4 2-D [3] standard for fax data compression is selected for compressing the hidden picture on a frame by frame basis. This standard is known to achieve 6-10 times data reduction for an ordinary image. Therefore, an 112 x 128 dots picture consisting of 1792 bytes of original information can be reduced to about 300 bytes before being transmitted. Compression is carried out as a coding procedure. It proceeds from top (1st line) to bottom (128th line) lines and from left (1st pixel) to right (112th pixel) positions by moving a reference position, called A0, which is set to zero at the start of each coding line. To output a code and advance A0, four more positions related to A0 are needed as shown in Fig. 2. They are:

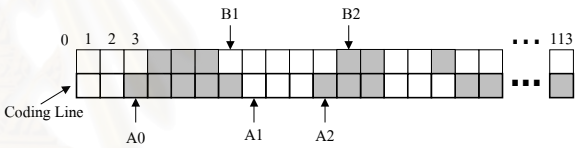


Fig. 2 Five reference positions for coding

A1: The next color changing position to the right of A0 on the coding line.
A2: The next color changing position to the right of A1 on the coding line.
B1: The first color changing position on the preceding line to the right of A0 and of opposite color to A0.
B2: The next color changing position on the preceding line to the right of B1.

These values are set to 113, or end of line, when they cannot be detected. Each line coding stops when A0 equal to 113. For the top line, its preceding line is assumed to be all white which is also the default color at the 0th position of each line. Table I summarizes the coding algorithm and how A0 is advanced after coding.

Since the compression process is not complicated and uses only 2 consecutive lines at a time, so is the decompression. Therefore these processes can be easily implemented and quickly executed by using an 8-bit microcontroller.

Mode	Condition	Code output	New A0 after coding
Pass	B2 < A1	0001	B2
Horizontal	B2 > A1 and |A1-B1|>3	001+M(A1-A0) +M(A2-A1) *	A2
Vertical	A1 - B1 = 0	1	A1
	A1 - B1 = 1	011	
	A1 - B1 = 2	000011	
	A1 - B1 = 3	0000011	
	B1 - A1 = 1	010	
	B1 - A1 = 2	000010	
	B1 - A1 = 3	0000010	

* M( ) is a tabulated run length coding function [3]
+ is a concatenation function

Table 1 Coding Table

III. VBI data format and signals

To facilitate the detection of each picture at the receiver, each compressed picture data must be framed as shown in Fig. 3, with 00000011 at its beginning and 00000000 at the end before being transmitted.

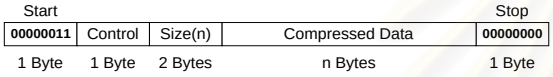


Fig.3 VBI Data Format

These two patterns are guaranteed not to exist in any part of the compressed data and, therefore, are called the Start and Stop bytes. The amount of data, in terms of bytes, is also stored as a two bytes Size parameter. This implies that the compressed data must be byte-justified at its end by zero bits. A Control byte is also added to provide controlling commands such as turn on/off the picture display at the receiver and to facilitate future extension such as multi color pictures.

These framed data are inserted in VBI scan lines at a rate of 500 kbit/sec and the timing of the VBI data signal is shown in Fig. 4, i.e. only 3 bytes can be inserted in each line.

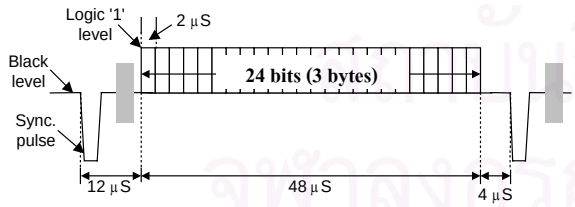


Fig.4 Timing waveform of a VBI data signal

This slightly low data rate is chosen so that the data signal can be accurately generated and detected by using an 8-bit microcontroller. It also ensures that the data signal can be recorded in a normal VHS videotape with no significant distortion. With only 16 VBI scan lines, i.e. 7th line to 22nd line, available in each field for data insertion, the total capacity for data transmission in each 40 ms frame of PAL video signal is equal to (2 fields/frame) x (16 lines/field) x (3 bytes/lines) or 96 bytes. Hence for an ordinary image with a compressed data size of about 300 bytes, an average animation picture of 3-video frames/image or 8.3 images/sec can be supported by the proposed system.

IV. A personal computer (PC) based data inserter

Fig.5 shows a simple implementation of a PAL video transmitting system that supports the hidden picture data in its VBI scan lines. The system requires a specialize hardware, called a VBI data inserter, for accurately placing the picture data in the incoming video signal to produce an output signal that can either be recorded by a video tape recorder or broadcast through an RF transmitter. This inserter is designed to work with a PC and in synchronization with the video signal according to a simplified working chart shown in Fig.6

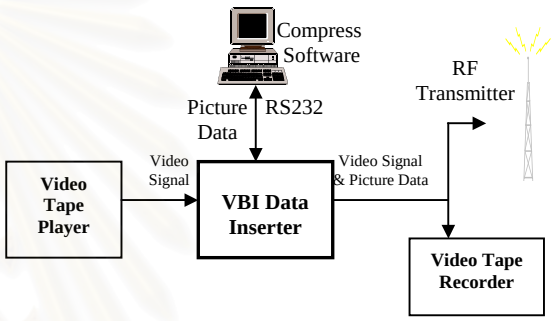


Fig. 5 Inserting system overview

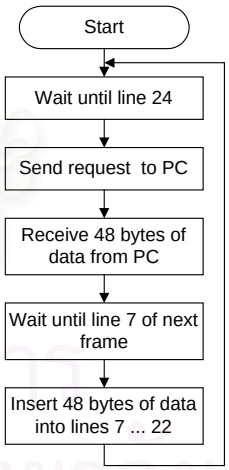


Fig. 6 Inserter working chart

An asynchronous serial communication (RS232) between a PC and the inserter is used and the baud rate is set at 48kbit/sec to guarantee that 48 bytes of data can be successfully transferred before the next video frame starts. The PC also plays an important role in preparing the compressed picture as well as their display timing in order to generate the data for the inserter. However, its operations are implemented in software that is beyond the scope of this paper.

The internal structure of the VBI data inserter is shown in Fig. 7 and briefly described as follows:

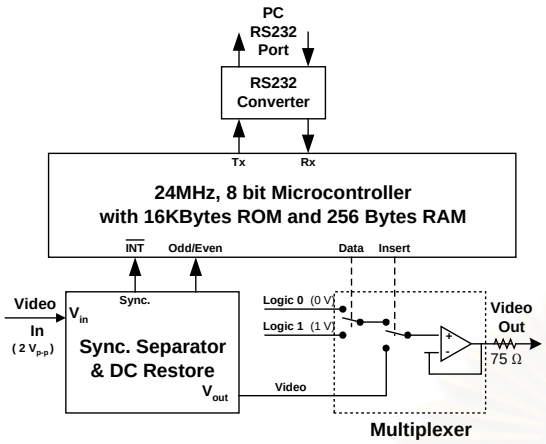


Fig.7 Structure of the VBI data inserter

A video sync separator, e.g. [4], is used to detect synchronizing pulses and the odd/even fields in the incoming video signal. These detected signals, i.e. sync and odd/even, are used by the microcontroller, e.g. [5], to locate the beginning of a video scan line and its line number within a field. The microcontroller is programmed to use these lines information for executing various operations within each cycle as shown in Fig.6.

During each cycle, 48 bytes of data are transferred from the PC and stored in the internal memory (RAM) of the microcontroller which later shifts them serially out at its output port to control the data insertion at the video multiplexer, e.g. [6],.

A dc restore circuit, e.g. [7], is used to set the black level of each video scan line at zero volt. This enables the multiplexer to select a one volt when a '1' bit must be inserted and the restored video signal in the case of a '0' bit. Since one bit of data is only 2 μS wide, the use of microcontroller operating at 24 MHz clock guarantees the timing precision of data insertion to within ± 0.5% error.

V. A low cost hidden picture decoder

To view the hidden picture on a normal TV screen, a decoding circuit is needed as shown in Fig.8.

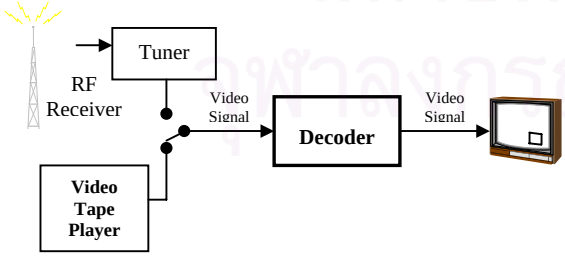


Fig. 8 Decoding system overview

The decoder is designed to display the picture at lines 205 to 269 of each field (312.5 lines) for a total of 64x2 = 128 rows. A 12 MHz dot frequency is chosen for displaying 112 horizontal dots per each 64 μS scan line. Hence, the decoded picture can be located at the lower right position of the TV screen as shown in Fig.2. The internal structure of this decoder, shown in Fig.9, is the same as that of the inserter with some additional components.

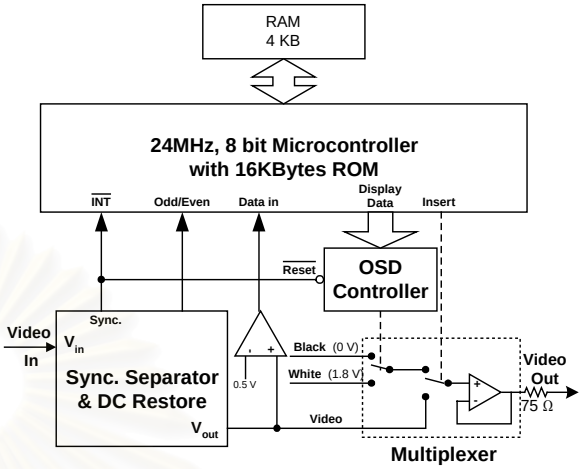


Fig. 9 Structure of the decoder

A high speed comparator, e.g.[8], is used to extract the hidden data signal by slicing the restored video signal at 0.5 V which is the middle of the '0' and '1' levels. The microcontroller performs various operations with respect to the incoming video lines according to the following pseudo-code.

Hardware initialization
Repeat
  Wait until the beginning of a video line;
  Case line number **of**
    6..22: Read 3 bytes of data in each video line and all 48 bytes store in the internal RAM;
    23: Decompress all 48 bytes of data and store the decompressed pixel data in the non-display RAM;
      If stop byte is found then switch the display and non-display RAM;
    205..269: On screen display one line of picture (112 dots) from the display RAM in the video line.
  End case;
Until forever;

By operating at 24 MHz, the microcontroller can accurately sample the 500-kbit/sec data at the middle point. Decompressing 48 bytes of sampled data, which starts at line 23, will also be completed before line 205. Decompressed data are stored in an external RAM organized as two pages of 2 Kbytes, each of which is used to store one 1792 bytes of picture data. Only one page is selected for on screen display (OSD) by the control and stop bytes. It is named the display page, while the other is called the non-display page.

During an OSD process, data in the display page are moved to an 8-bit OSD controller, which shifts them

out at 12Mbit/sec by using a 12 MHz clock generator as shown in Fig.10. This generator is always reset at the start of each scan line to generate a stable display. It frequency, however, needs not be accurate as this will effect only the width of the displayed picture.

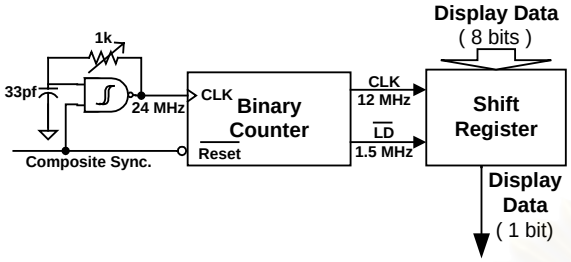


Fig. 10 An OSD controlling circuit

VI. Test results and conclusion

The hidden picture shown in Fig. 2 was used to test the developed inserter and decoder. This 1792 bytes of picture data was compressed by using a personal computer software. The size of its compressed data is 315 bytes, corresponding to a compression ratio of 5.68. Therefore, 4 video frames were needed to hide this picture in its VBIs. After receiving this data, the decoder decompresses and displays the picture on a TV screen as shown in Fig. 14.

Fig. 14 Actual display of a hidden picture on a TV screen

In conclusion, the potential of using VBI scan lines of an analog video system for transmitting pictures has been demonstrated. The CCITT T.4 2-D compression algorithm and a low data rate of 500 kbit/sec are adopted simply to minimize the hardware development cost by using a low cost 8-bit microcontroller.

The presented system can support a picture of 112x128 black and white dots with an update rate of 6.25 frame/sec. This should be good enough for showing a sign language animation picture. More sophisticated compression algorithm, faster data rate and more advanced hardware can significantly improve its performance and therefore should be further investigated.

VII. References

[1] Television Engineering Handbook, K. Blair Benson, Jerry Whitaker, 1992

[2] Report and order on GEN docket No. 91-1, Federal Communications commission United States of America, 1991

[3] CCITT T4 2-Dimension Compression Standard, ITU-T, 1988

[4] LM1881 Video Sync. Separator, National semiconductor

[5] DS80C320 High-speed/Low-Power Microcontroller, Dallas Semiconductor

[6] MAX454 CMOS Video Multiplexer/Amplifier, Maxim Integrated Products

[7] EL4093C 300 MHz DC-Restored Video Amplifier, Elantec

[8] LM319 High Speed Dual Comparator, National Semiconductor

ภาคผนวก ค

รายละเอียดโปรแกรมควบคุมการทำงานเครื่องแทรกข้อมูลภาพ

```
;Status Bit
FIELD_T         EQU     00H
RS232_READY     EQU     02H
SYNC_RECV       EQU     03H
;Defind Register
SYNC_COUNT      EQU     31H
;I/O Port
DATA_O          EQU     P3.5
FIELD           EQU     P3.4
INST            EQU     P3.2
;Specific Register
TL2             EQU     0CCH
TH2             EQU     0CDH
RCAP2L          EQU     0CAH
RCAP2H          EQU     0CBH
T2CON           EQU     0C8H
SCON1           EQU     0C0H
WDCON           EQU     0D8H

;Defind Constant
TOP             EQU     0CH ;12
BOTTOM          EQU     1BH ;27
START_RECV      EQU     40H ; START RECV
ADDRESS
;RAM Start ADD
PAGE1           EQU     0000H
PAGE2           EQU     1000H
;-----------------------------------------
;Defind Macro
;---------------------------------------------------------
; CHECK S > D ?
; S IS ACC OR Ri AND D IS ANY TYPE OF DATA ,
; BUT IF S IS Ri D IS NOT Ri OR ACC
IFGT    macro  S,D,ELSE1
        local NOTEQU,THEN
        CJNE S,D,NOTEQU
        JMP ELSE1
NOTEQU:
        JNC THEN
        JMP ELSE1
THEN:
        endm
;---------------------------------------------------------
; CHECK S >= D ?
; S IS ACC OR Ri AND D IS ANY TYPE OF DATA ,
; BUT IF S IS Ri D IS NOT Ri OR ACC
IFGE    macro  S,D,ELSE1
        local NOTEQU,THEN
        CJNE S,D,NOTEQU
        JMP THEN
NOTEQU:
        JNC THEN
        JMP ELSE1
THEN:
        endm
;---------------------------------------------------------
; CHECK S < D ?
; S IS ACC OR Ri AND D IS ANY TYPE OF DATA ,
; BUT IF S IS Ri D IS NOT Ri OR ACC
IFLT   macro  S,D,ELSE1
        local NOTEQU,THEN
        CJNE S,D,NOTEQU
        JMP ELSE1
NOTEQU:
        JC THEN
        JMP ELSE1
THEN:
        endm
;---------------------------------------------------------
; CHECK S <= D ?
; S IS ACC OR Ri AND D IS ANY TYPE OF DATA ,
; BUT IF S IS Ri D IS NOT Ri OR ACC
IFLE   macro  S,D,ELSE1
        local NOTEQU,THEN
        CJNE S,D,NOTEQU
        JMP THEN
NOTEQU:
        JC THEN
        JMP ELSE1
THEN:
        endm
;---------------------------------------------------------
; CHECK S = D ?
; S IS ACC OR Ri AND D IS ANY TYPE OF DATA ,
; BUT IF S IS Ri D IS NOT Ri OR ACC
IFEQ   macro  S,D,ELSE1
        local NOTEQU,THEN
        CJNE S,D,NOTEQU
        JMP THEN
NOTEQU:
        JMP ELSE1
THEN:
        endm
;---------------------------------------------------------
; CHECK S != D ?
; S IS ACC OR Ri AND D IS ANY TYPE OF DATA ,
; BUT IF S IS Ri D IS NOT Ri OR ACC
IFNEQ   macro  S,D,ELSE1
        local THEN
        CJNE S,D,THEN
        JMP ELSE1
THEN:
        endm
;---------------------------------------------------------
; CAUTION !  YOU MUST SAVE THE ACC, BEFORE
USE THIS MACRO.
; SWITCH MACRO MAKE CHANGE THE ACC TO
NEW VALUE REFER TO PARAMETER(DATA).
;SWITCH  macro DATA
;       MOV A,DATA
;       endm
;---------------------------------------------------------
ENDSWITCH      macro LINE
               LABEL LINE
                endm
;---------------------------------------------------------
DEC_AND_JUMP  macro COUNT,LABEL_TO_GO
              local
DEC_AND_JUMP1,END_DEC_AND_JUMP
                        DJNZ COUNT,DEC_AND_JUMP1
              JMP END_DEC_AND_JUMP
DEC_AND_JUMP1:
              JMP LABEL_TO_GO
END_DEC_AND_JUMP:
              endm
;---------------------------------------------------------
; CASE MACRO , USE THE ACC TO COMPARE!
THEN YOU MUST USE SWITCH MACRO FIRST!.
CASE    macro D,ELSE1
        local NOTEQU,THEN
        CJNE A,D,NOTEQU
        JMP THEN
NOTEQU:
        JMP ELSE1
THEN:
        endm
;---------------------------------------------------------
DEFAULT macro
        endm
;---------------------------------------------------------
; FOR DECARE THE LABEL
LABEL   macro LINE
LINE:
        endm
;---------------------------------------------------------
; FOR DECARE THE ELSE1
```

```
ELSE1   macro LINE
LINE:
        endm
;-------------------------------------------------------
BREAK   macro LINE
        JMP LINE
        endm
;-------------------------------------------------------
SEND    macro PARA
        local SEND4,SEND3,SEND41
SEND41:
        MOV SBUF,PARA
    ;   clr es
SEND4:
        JBC TI,SEND3
        JMP SEND4
SEND3:
     ;  setb es
        endm

;=========================================
====
; SEND THE DATA TO CUTALK3.0
; FIRST, YOU MUST SET THE BAUD RATE IN THE
MAIN
;       ROUTINE.
;=========================================
====
SEND2   macro PARA
        local SEND24,SEND23
;       push acc
        MOV SBUF1,PARA
SEND24:
        MOV A,SCON1
        JNB ACC.1,SEND24
SEND23:
;       MOV SCON1,#0
;       pop acc
        endm

;-----------------------------------------------
; THIS IS MARCO FOR SET BAUD RATE INTO 9600
; NONE PARITY,ONE STOP BIT , ONLY DALLA 80320
; AT 24MHZ , 48 = 0F3H AND DOUBLE 48 TO 96 BY
; SET PCON = 1xxx xxxxB AT THE 5th LINE.
;-----------------------------------------------

SET_BAUD_RATE   macro
             ; SET BAUD RATE OF SERIAL2
            MOV SCON,#50H
            MOV TMOD,#022H
            MOV TH1,#0f3h         ; #0FdH = 11.59MHz
            MOV TL1,#0f3h         ; #0FdH
            MOV WDCON,#80H        ; SET DOUBLE
BAUD RATE of SERIAL2.
            ; SET BAUD RATE OF SERIAL1
            ;MOV SCON1,#50h
            MOV SCON1,#00h
            MOV T2CON,#34H
            MOV TH2,#0FFH         ; 24Mhz  =
0ffh,32Mhz = 0fbh
            MOV RCAP2H,TH2
            MOV TL2,#0E9H         ; #0D9H =
11.0592MHz(9.6K),24Mhz = 0E9h (38.4K)
                                  ; 32Mhz = 0eeh(9.6K)
            MOV RCAP2L,TL2
      ;     SETB TR1              ; FOR SERIAL1 TO
RECEIVE DATA.
            endm
;-------------------------------------------------------------------------
                ORG 00H
                JMP INITIAL
                ORG 13H
                JMP NEW_LINE
                ORG 23H
                JMP SERIAL
                ORG 100H
;-------------------------------------------------------------------------
INITIAL:        MOV DPTR,#00H
                MOV A,#00H
                MOV R2,#0FFH
CLEAR_L1:       MOV R3,#0FFH
CLEAR_L2:       MOVX @DPTR,A
                INC DPTR
                DJNZ R3,CLEAR_L2
                DJNZ R2,CLEAR_L1
                MOV R0,#START_RECV
                MOV R4,#30H
                MOV A,#00H
CLEAR_IN_RAM:   MOV @R0,A
                INC R0
                DJNZ R4,CLEAR_IN_RAM

MAIN:           MOV SYNC_COUNT,#00H
                MOV IE,#0CH
                SETB IT1
                SET_BAUD_RATE
                MOV DPTR,#00H
                MOV R5,#30H
                MOV R0,#START_RECV
                MOV R1,#START_RECV
;-------------------------------------------------------------------------
WAIT_OE:        JB FIELD,$   ;WAIT ODD EVEN
                JNB FIELD,$
                SETB FIELD_T
                MOV SYNC_COUNT,#00H
                SETB EA
                SETB EX1
;-------------------------------------------------------------------------
WAIT_NEW_LINE:  MOV PCON,#1H
                JMP WAIT_NEW_LINE
;-------------------------------------------------------------------------
GO_END_SPACE:   JMP END_SPACE

CHK_LINE23:     CJNE A,#1CH,GO_END_SPACE
                CLR EX1
                CALL BACK
                SETB ES
                MOV R1,#START_RECV
                MOV R0,#START_RECV
                MOV A,#'R'
                JMP END_SPACE

BACK:           RETI

NEW_LINE:       INC SYNC_COUNT
                MOV A,SYNC_COUNT
                IFGE A,#TOP,NOT_REACH
                IFLE A,#BOTTOM,CHK_LINE23
INSERT:         MOV R4,#05H
DELAY_LOOP:     NOP
                DJNZ R4,DELAY_LOOP
                NOP
                NOP
                NOP

                CLR DATA_O

                NOP
                MOV A,@R0
                RLC A
                MOV DATA_O,C
                CLR INST
```

```
                REPT 11
                NOP
                ENDM
                REPT 6
                RLC A
                MOV DATA_O,C
                REPT 9
                NOP
                ENDM
                ENDM
                RLC A
                MOV DATA_O,C
                REPT 3
                NOP
                ENDM
                INC R0

                REPT 4
                NOP
                ENDM
                MOV A,@R0
                REPT 7
                RLC A
                MOV DATA_O,C
                REPT 9
                NOP
                ENDM
                ENDM
                RLC A
                MOV DATA_O,C
                REPT 3
                NOP
                ENDM
                INC R0
                REPT 4
                NOP
                ENDM
                MOV A,@R0
                REPT 7
                RLC A
                MOV DATA_O,C
                REPT 9
                NOP
                ENDM
                ENDM
                RLC A
                MOV DATA_O,C
                REPT 3
                NOP
                ENDM
                INC R0

                NOP
                NOP
                NOP
                SETB INST
                SETB DATA_O
                SETB EX1
                RETI

NOT_REACH:      SETB EX1
                RETI
;-------------------------------------------------------------------------
SERIAL:         CLR EA
                CLR TI
                JBC RI,RECV_DATA
                JMP RET_SERIAL

RECV_DATA:      MOV A,SBUF
                MOV @R1,A
                INC R1

RET_SERIAL:     SETB EA
                RETI
;-------------------------------------------------------------------------
END_SPACE:      CLR EX1
                MOV SYNC_COUNT,#00H
                JB FIELD_T,NEXT_E
                JMP NEXT_O

;-------------------------------------------------------------------------
NEXT_E:         CLR FIELD_T
                MOV SBUF,A
                JB FIELD,$
                SETB EX1
                CLR ES
                RET
;-------------------------------------------------------------------------
NEXT_O:         SETB FIELD_T
                MOV SBUF,A
                JNB FIELD,$
                SETB EX1
                CLR ES
                RET
;-------------------------------------------------------------------------
                END
```

ภาคผนวก ง

รายละเอียดโปรแกรมควบคุมการทำงานเครื่องรับข้อมูลภาพ

```
FIELD_T         EQU     10H
SYNC_FIF        EQU     11H
START_FRAME     EQU     12H
SWAP_BANK       EQU     13H
KEEP_CONT       EQU     14H
COMP_READY      EQU     15H
BANK            EQU     16H
R52             EQU     17H
STATE           EQU     18H
TMP_DATA        EQU     19H
C_TMP           EQU     1AH

;INTERNAL REGISTER
;POINTER AND DISTANT VALUE
SYNC_COUNT      EQU     30H
CODE_LINE_NUM   EQU     31H
A0              EQU     32H
;BIT CODING TMP 2
BIT_C_TMP2      EQU     33H
CODE_TMP2       EQU     34H ;CODE TMP 2
;TELL NUMBER OF DATA BIT
BIT_C           EQU     35H
BIT_C_TMP       EQU     36H
;POINTER FOR REFERENCE LINE INDEX
R5_TMP          EQU     37H
;POINTER FOR CODE LINE INDEX
R5_TMP2         EQU     38H
;STORE CODE THAT IS SHIFTED BEFORE HOR
MODE
CODE_TMP        EQU     39H
R0_TMP          EQU     3AH  ;R0 TMP
R0_TMP2         EQU     3BH  ;R0 TMP 2
INDEX_TMP       EQU     3CH ;INDEX TMP
;KEEP NUMBER OF PIXEL
NUM_PIXEL       EQU     3DH
;NEW REFERENCE BYTE FOR V1-V3 MODE
REF_NBYTE       EQU     3EH
;OLD REF. BYTE FOR V1-V3 MODE
REF_OBYTE       EQU     3FH
R0_START        EQU     4EH ;VALUE  40H-4DH
;STORE STATE OF DECOMPRESS
S_BYTE          EQU     4FH
;STORE CONTROL BYTE
CON_BYTE        EQU     2FH
;STORE MASK TO OR/AND WITH REFERENCE
DATA
MASK_CODE       EQU     2EH
;CHECK END OF VERTICAL MODE
MASK_CHK        EQU     2EH.0
;STORE REF. BYTE WHICH IS DECODING
DECODING_BYTE   EQU     2DH
;CHECK LSB OF REF. DATA
DECODING_BCHK   EQU     2DH.0
R4_TMP          EQU     2AH
TMP_R3          EQU     29H
R1_TMP          EQU     28H
;I/O Port
RECV            EQU     P3.5
SW_MODE         EQU     P3.1
SH              EQU     P3.2
FIELD           EQU     P3.4
;Specific Register
DPL1            EQU     84H
DPH1            EQU     85H
DPS             EQU     86H

;Defind Constant
TOP             EQU     0CH  ;12
BOTTOM          EQU     1BH  ;27
TOP_D           EQU     09FH ;159
BOTTOM_D        EQU     0E1H ;225
COL             EQU     70H  ;112
ROW             EQU     81H ;128
REF_LINE        EQU     40H ;40H - 4DH
;FOR KEEP OSD DATA BEFORE SEND TO SHIFT R.
DATA_PIC        EQU     70H
DATA_RECV       EQU     50H
; 48 BYTES OF DATA
RECV_SIZE       EQU     30H

;RAM Start ADD
PIC1_DISPLAY    EQU     0000H
PIC1_DISPLAY_E  EQU     000EH
PIC2_DISPLAY    EQU     1000H
PIC2_DISPLAY_E  EQU     100EH

;-----------------------------------------
;Defind Macro
;-------------------------------------------------------
; CHECK S > D ?
; S IS ACC OR Ri AND D IS ANY TYPE OF DATA ,
; BUT IF S IS Ri D IS NOT Ri OR ACC
IFGT    macro  S,D,ELSE1
        local NOTEQU,THEN
        CJNE S,D,NOTEQU
        JMP ELSE1
NOTEQU:
        JNC THEN
        JMP ELSE1
THEN:
        endm
;-------------------------------------------------------
; CHECK S >= D ?
; S IS ACC OR Ri AND D IS ANY TYPE OF DATA ,
; BUT IF S IS Ri D IS NOT Ri OR ACC
IFGE    macro  S,D,ELSE1
        local NOTEQU,THEN
        CJNE S,D,NOTEQU
        JMP THEN
NOTEQU:
        JNC THEN
        JMP ELSE1
THEN:
        endm
;-------------------------------------------------------
; CHECK S < D ?
; S IS ACC OR Ri AND D IS ANY TYPE OF DATA ,
; BUT IF S IS Ri D IS NOT Ri OR ACC
IFLT   macro  S,D,ELSE1
        local NOTEQU,THEN
        CJNE S,D,NOTEQU
        JMP ELSE1
NOTEQU:
        JC THEN
        JMP ELSE1
THEN:
        endm
;-------------------------------------------------------
; CHECK S <= D ?
; S IS ACC OR Ri AND D IS ANY TYPE OF DATA ,
; BUT IF S IS Ri D IS NOT Ri OR ACC
IFLE   macro  S,D,ELSE1
        local NOTEQU,THEN
        CJNE S,D,NOTEQU
        JMP THEN
NOTEQU:
        JC THEN
        JMP ELSE1
THEN:
        endm
;-------------------------------------------------------
; CHECK S = D ?
; S IS ACC OR Ri AND D IS ANY TYPE OF DATA ,
; BUT IF S IS Ri D IS NOT Ri OR ACC
IFEQ   macro  S,D,ELSE1
        local NOTEQU,THEN
```

```
        CJNE S,D,NOTEQU
        JMP THEN
NOTEQU:
        JMP ELSE1
THEN:
        endm
;-------------------------------------------------------
; CHECK S != D ?
; S IS ACC OR Ri AND D IS ANY TYPE OF DATA ,
; BUT IF S IS Ri D IS NOT Ri OR ACC
IFNEQ   macro  S,D,ELSE1
        local THEN
        CJNE S,D,THEN
        JMP ELSE1
THEN:
        endm
;-------------------------------------------------------------------
                ORG 00H
                JMP INITIAL
                ORG 13H
                JMP NEW_LINE
                ORG 100H
;-------------------------------------------------------------------
INITIAL:        MOV DPTR,#00H
                MOV A,#00H
                MOV R2,#0FFH
CLEAR_L1:       MOV R3,#0FFH
CLEAR_L2:       MOVX @DPTR,A
                INC DPTR
                DJNZ R3,CLEAR_L2
                DJNZ R2,CLEAR_L1

MAIN:           MOV IE,#80H
;---------INITIAL STATUAS BIT--------------------------
;------- COUNT LINE, RECV AND OSD ------
                CLR SYNC_FIF
                CLR SWAP_BANK
                CLR C
                SETB IT1
                SETB SH
;---------- DECOMPRESS PROCESS -------------
                CLR KEEP_CONT
                CLR STATE
                CLR START_FRAME
;------INITIAL RAM AND REGISTER----------------------
;------- COUNT LINE, RECV DATA AND OSD ------
                MOV IP,#08H
                MOV SYNC_COUNT,#00H
                MOV R0,#DATA_RECV
                MOV DPS,#00H
                MOV DPTR,#PIC1_DISPLAY
                INC DPS
                MOV DPTR,#PIC1_DISPLAY
;---------- DECOMPRESS PROCESS -------------
                MOV CODE_LINE_NUM,#01H
                MOV A0,#00H
                MOV BIT_C,#08H
                MOV R1,#00H
                MOV R1_TMP,#00H
                MOV R4_TMP,#00H
                MOV R0_START,#50H
;-------------------------------------------------------------------
WAIT_OE:        JB FIELD,$ ;WAIT ODD EVEN
                JNB FIELD,$
                SETB FIELD_T
                MOV SYNC_COUNT,#00H
                SETB EX1
;-------------------------------------------------------------------
WAIT_NEW_LINE:  MOV PCON,#1H
                JMP WAIT_NEW_LINE
;-------------------------------------------------------------------
NEW_LINE:       MOV C_TMP,C
                INC SYNC_COUNT
                MOV R3,SYNC_COUNT
                JNB SYNC_FIF,UNDER_FIF
                IFGT R3,#TOP_D,NOT_REACH
                IFLT R3,#BOTTOM_D,
                        END_SPACE_DISP
                JMP OSD
UNDER_FIF:      IFGE R3,#TOP,NOT_REACH
                IFLE R3,#BOTTOM,CHK_LINE24
                AJMP RECV_DATA
CHK_LINE24:     CJNE R3,#1CH,CHK_NEXT
                JMP STORE_DATA
CHK_NEXT:       CJNE R3,#32H,NOT_REACH
                SETB SYNC_FIF
                MOV SYNC_COUNT,#00H
                JMP NOT_REACH
;-------------------------------------------------------------------
NOT_REACH:      SETB EX1
                MOV C,C_TMP
                RETI
;-------------------------------------------------------------------
;---------------------- LINE 7-22 ----------------------------------

RECV_DATA:      JB START_FRAME,RECV1

;-------------- CHECK START FRAME ----------------------
                DEC R0
                DEC R0
                DEC R0
                MOV A,@R0
                CJNE A,#03H,RECV2
                INC R0
                MOV A,@R0
                CJNE A,#03H,RECV3
                INC R0
                MOV A,@R0
                CJNE A,#03H,RECV4
                INC R0
                MOV R0_START,R0
                SETB START_FRAME

CONT_RECV4:     JMP GET_DATA

;--------------- DELAY FOR RECEIVER DATA -------------
RECV1:          MOV R4,#07H
DELAY_RECV1:    DJNZ R4,DELAY_RECV1
                REPT 3
                NOP
                ENDM
                JMP GET_DATA

RECV2:          REPT 6
                NOP
                ENDM
                INC R0

RECV3:          MOV R4,#01H
DELAY_RECV3:    DJNZ R4,DELAY_RECV3
                INC R0

RECV4:          INC R0
                CJNE R0,#80H,CONT_RECV4
                MOV R0,#DATA_RECV
;----------------- RECEIVER DATA --------------------
GET_DATA:       REPT 7
                MOV C,RECV
                RLC A
                REPT 9
                NOP
                ENDM
                ENDM
                MOV C,RECV
                RLC A
```

```
                MOV @R0,A
                INC R0
                REPT 8
                NOP
                ENDM

                REPT 7
                MOV C,RECV
                RLC A
                REPT 9
                NOP
                ENDM
                ENDM
                MOV C,RECV
                RLC A
                MOV @R0,A
                INC R0
                REPT 8
                NOP
                ENDM
                REPT 7
                MOV C,RECV
                RLC A
                REPT 9
                NOP
                ENDM
                ENDM
                MOV C,RECV
                RLC A
                MOV @R0,A
                INC R0
                REPT 8
                NOP
                ENDM
                SETB EX1
                RETI
;=========================================
;----------------------- ON SCEEN DISPLAY -------------------
;------------ MOV DATA TO INTERNAL RAM -----------
OSD:            MOV DPS,#01H
                MOV R0,#DATA_PIC
                MOVX A,@DPTR
                MOV @R0,A
                INC R0
                INC DPTR

                MOVX A,@DPTR
                MOV @R0,A
                INC R0
                INC DPTR

                MOVX A,@DPTR
                MOV @R0,A
                INC R0
                INC DPTR

                MOVX A,@DPTR
                MOV @R0,A
                INC R0
                INC DPTR

                MOVX A,@DPTR
                MOV @R0,A
                INC R0
                INC DPTR

                MOVX A,@DPTR
                MOV @R0,A
                INC R0
                INC DPTR

                MOVX A,@DPTR
                MOV @R0,A
                INC R0
                INC DPTR
                MOVX A,@DPTR
                MOV @R0,A
                INC R0
                INC DPTR
                MOVX A,@DPTR
                MOV @R0,A
                INC R0
                INC DPTR
                MOVX A,@DPTR
                MOV @R0,A
                INC R0
                INC DPTR
                MOVX A,@DPTR
                MOV @R0,A
                INC R0
                INC DPTR
                MOVX A,@DPTR
                MOV @R0,A
                INC R0
                INC DPTR
                MOVX A,@DPTR
                MOV @R0,A
                INC R0
                MOV A,DPL1
                ADD A,#0FH
                MOV DPL1,A
                MOV A,DPH1
                ADDC A,#00H
                MOV DPH1,A
;------------- MOV DATA TO SHIFT REGISTER ------------
                MOV R4,#18H
DELAY_INST:     DJNZ R4,DELAY_INST
                REPT 7
                NOP
                ENDM
                MOV R0,#DATA_PIC
                MOV A,@R0
                CLR SH
                MOV P1,A

                INC R0
                MOV A,@R0
                MOV P1,A
                INC R0
                MOV A,@R0
                MOV P1,A

                INC R0
                MOV A,@R0
                MOV P1,A

                INC R0
                MOV A,@R0
                MOV P1,A
                INC R0
                MOV A,@R0
                MOV P1,A
                NOP
                INC R0
                MOV A,@R0
                MOV P1,A
                INC R0
                MOV A,@R0
                MOV P1,A
                NOP
```

```
                    INC R0
                    MOV A,@R0
                    MOV P1,A
                    NOP

                    INC R0
                    MOV A,@R0
                    MOV P1,A
                    INC R0
                    MOV A,@R0
                    MOV P1,A
                    NOP

                    INC R0
                    MOV A,@R0
                    MOV P1,A
                    INC R0
                    MOV A,@R0
                    MOV P1,A
                    NOP

                    INC R0
                    MOV A,@R0
                    MOV P1,A
                    NOP
                    NOP
                    SETB SH
                    MOV P1,#0FFH
                    SETB EX1
                     RETI
;=======================================
;---------------------------------------------------------------
END_SPACE_DISP:CLR EX1
                    MOV SYNC_COUNT,#00H
                    CLR SYNC_FIF
                    MOV R0,#DATA_RECV
                    JB FIELD_T,NEXT_E
                    JMP NEXT_O
;---------------------- EVEN FIELD -----------------------------
NEXT_E:             CLR FIELD_T
                    JB FIELD,$
                    SETB EX1
                    RETI
;------------------------ ODD FIELD -------------------------------
NEXT_O:             SETB FIELD_T
                    JNB FIELD,$
                    SETB EX1
                    RETI
;=======================================
;--------------------- DECOMPRESSION PROCESS -------
BACK:               RETI

STORE_DATA:         CALL BACK
                    SETB EX1
                    MOV DPS,#01H
                    MOV R1,R1_TMP
                    MOV R4,R4_TMP
                    MOV R0,#DATA_RECV
                    MOV R5,#RECV_SIZE
                    JNB START_FRAME,GO_EXIT
;--- CHECK KEEP CONTROL BYTE AND SIZE BYTE --
                    JB KEEP_CONT,CHK_STATE
                    MOV R0,R0_START
                    DEC R5
                    DEC R5
                    DEC R5
CHK_END:            CJNE R0,#80H,GET_CONT
                    MOV R0_START,#50H
GO_EXIT:            JMP EXIT_PROC
;---------- STORE CONTROL BYTE AND SIZE -----------
GET_CONT:           SETB KEEP_CONT
                    MOV CON_BYTE,@R0
                    INC R0
                    DEC R5
                    MOV A,@R0
                    MOV R7,A
                    INC R0
                    DEC R5
                    MOV A,@R0
                    MOV R6,A
                    INC R0
                    DEC R5
                    MOV A,#0FFH
                    MOV R1,#REF_LINE
;------- WHITE REFERENCE FOR FIRST LINE ----------
                    REPT 14
                    MOV @R1,A
                    INC R1
                    ENDM
;---------------------- INITIAL REGISTER ----------------------
                    MOV A0,#00H
                    MOV R2,#08H
                    MOV R1,#REF_LINE
;-GET 1 BYTE OF COMPRESS DATA AND REF. LINE
                    MOV A,@R0
;GET COMPRESS DATA
;---------- SET DISPLAY BANK ------------------------
                    MOV DPS,#00H
                    JB SWAP_BANK,ST_BANK2
                    MOV DPTR,#PIC1_DISPLAY
                    CJNE R0,#80H,DECOM_DATA
                    JMP EXIT_PROC
ST_BANK2:           MOV DPTR,#PIC2_DISPLAY
                    CJNE R0,#80H,DECOM_DATA
                    JMP EXIT_PROC
;========== DECOMPRESSION PROCESS =======
CHK_STATE:          MOV DPS,#01H
                    MOV DPTR,#STATE_TAB
                ;----------HIGH BYTE----------
                    MOV A,S_BYTE
                    RL A
                    INC A
                    MOVC A,@A+DPTR
                    PUSH ACC
                ;----------LOW BYTE-----------
                    MOV A,S_BYTE
                    RL A
                    MOVC A,@A+DPTR
                    PUSH ACC
                    MOV A,@R0
                    RET

DECOM_DATA:         PUSH ACC
;PUSH COMPRESS DATA TO STACK
                    MOV A,A0
                    CJNE A,#00H,NOT_FIRST
                    CLR STATE
                    POP ACC
;POP COMPRESS DATA TO STACK
                    JMP DECOM_PROC
NOT_FIRST:          POP ACC
;POP COMPRESS DATA TO STACK
;--------------------- SELECT MODE----------------------
DECOM_PROC:         MOV DPS,#01H
                    RLC A
                    DJNZ BIT_C,CONT_C1
                    MOV BIT_C,#08H
                    INC R0
                    DJNZ R6,CONTS_02
                    MOV R6,#0FFH
                    DJNZ R7,CONTS_02
                    JMP END_PIC
CONTS_02:           JNB TMP_DATA,CONTD_01
                    MOV R0,#DATA_RECV
                    MOV R5,#RECV_SIZE
```

```
                CLR TMP_DATA
                MOV A,@R0
                JMP CONT_C1
CONTD_01:       MOV A,@R0
                DJNZ R5,CONT_C1
                JNC S_NC1
                ; GO TO V0_MODE IN NEXT TIME
                MOV S_BYTE,#01H
                JMP END_48_BYTES
S_NC1:          ; GO TO NEXT_C1 IN NEXT TIME
                MOV S_BYTE,#02H
                JMP END_48_BYTES
CONT_C1:        JNC NEXT_C1
                JMP V0_MODE     ;1

NEXT_C1:        RLC A
                DJNZ BIT_C,CONT_C2
                MOV BIT_C,#08H
                INC R0
                DJNZ R6,CONTS_03
                MOV R6,#0FFH
                DJNZ R7,CONTS_03
                JMP END_PIC
CONTS_03:       JNB TMP_DATA,CONTD_02
                MOV R0,#DATA_RECV
                MOV R5,#RECV_SIZE
                CLR TMP_DATA
                MOV A,@R0
                JMP CONT_C2
CONTD_02:       MOV A,@R0
                DJNZ R5,CONT_C2
                JNC S_NC2
                ; GO TO V1_MODE IN NEXT TIME
                MOV S_BYTE,#03H
                JMP END_48_BYTES
S_NC2:          ; GO TO NEXT_C2 IN NEXT TIME
                MOV S_BYTE,#04H
                JMP END_48_BYTES
CONT_C2:        JNC NEXT_C2
                JMP V1_MODE     ;01

NEXT_C2:        RLC A
                DJNZ BIT_C,CONT_C3
                MOV BIT_C,#08H
                INC R0
                DJNZ R6,CONTS_04
                MOV R6,#0FFH
                DJNZ R7,CONTS_04
                JMP END_PIC
CONTS_04:       JNB TMP_DATA,CONTD_03
                MOV R0,#DATA_RECV
                MOV R5,#RECV_SIZE
                CLR TMP_DATA
                MOV A,@R0
                JMP CONT_C3
CONTD_03:       MOV A,@R0
                DJNZ R5,CONT_C3
                JNC S_NC3
        ; GO TO HOR_MODE IN NEXT TIME
                MOV S_BYTE,#05H
                JMP END_48_BYTES
S_NC3:  ; GO TO NEXT_C3 IN NEXT TIME
                MOV S_BYTE,#06H
                JMP END_48_BYTES
CONT_C3         JNC NEXT_C3
                JMP HOR_MODE   ;001

NEXT_C3:        RLC A
                DJNZ BIT_C,CONT_C4
                MOV BIT_C,#08H
                INC R0
                DJNZ R6,CONTS_05
                MOV R6,#0FFH
                DJNZ R7,CONTS_05
                JMP END_PIC
CONTS_05:       JNB TMP_DATA,CONTD_04
                MOV R0,#DATA_RECV
                MOV R5,#RECV_SIZE
                CLR TMP_DATA
                MOV A,@R0
                JMP CONT_C4
CONTD_04:       OV A,@R0
                DJNZ R5,CONT_C4
                JNC S_NC4
        ; GO TO PASS_MODE IN NEXT TIME
                MOV S_BYTE,#07H
                JMP END_48_BYTES
S_NC4:  ; GO TO NEXT_C4 IN NEXT TIME
                MOV S_BYTE,#08H
                JMP END_48_BYTES
CONT_C4:JNC NEXT_C4
                JMP PASS_MODE     ;0001

NEXT_C4:        RLC A
                DJNZ BIT_C,CONT_C5
                MOV BIT_C,#08H
                INC R0
                DJNZ R6,CONTS_06
                MOV R6,#0FFH
                DJNZ R7,CONTS_06
                JMP END_PIC
CONTS_06:       JNB TMP_DATA,CONTD_05
                MOV R0,#DATA_RECV
                MOV R5,#RECV_SIZE
                CLR TMP_DATA
                MOV A,@R0
                JMP CONT_C5
CONTD_05:       MOV A,@R0
                DJNZ R5,CONT_C5
                JNC S_NC5
        ; GO TO V2_MODE IN NEXT TIME
                MOV S_BYTE,#09H
                JMP END_48_BYTES
S_NC5:  ; GO TO NEXT_C5 IN NEXT TIME
                MOV S_BYTE,#0AH
                JMP END_48_BYTES
CONT_C5:        JNC NEXT_C5
                JMP V2_MODE     ;00001

NEXT_C5:        RLC A
                DJNZ BIT_C,CONT_C6
                MOV BIT_C,#08H
                INC R0
                DJNZ R6,CONTS_07
                MOV R6,#0FFH
                DJNZ R7,CONTS_07
                JMP END_PIC
CONTS_07:       JNB TMP_DATA,CONTD_06
                MOV R0,#DATA_RECV
                MOV R5,#RECV_SIZE
                CLR TMP_DATA
                MOV A,@R0
                JMP CONT_C6
CONTD_06:       MOV A,@R0
                DJNZ R5,CONT_C6
                JNC S_NC6
        ; GO TO V3_MODE IN NEXT TIME
                MOV S_BYTE,#0BH
                JMP END_48_BYTES
S_NC6:  ; GO TO NEXT_C6 IN NEXT TIME
                MOV S_BYTE,#0CH
                JMP END_48_BYTES
CONT_C6:        JNC NEXT_C6
                JMP V3_MODE     ;000001

NEXT_C6:        RLC A
```

```
                JNC ERROR
                JMP END_PIC     ;0000001
;00000001 OR 00000000
ERROR:          CPL SWAP_BANK
                JMP END_PIC
;============= V0 MODE =================
GO_V01_PROC:    JMP V01_PROC
V0_MODE:        MOV CODE_TMP,A
;SWAP COMPRESS DATA TO TMP_CODE AND REF.
LINE TO A
        ; ------ GET REF. LINE INDEX -----
                MOV A,A0
                MOV DPTR,#GET_R1
                MOVC A,@A+DPTR
                MOV R1,A ;GET REF. LINE INDEX
        ; ------ GET A00 - A07 -------
                MOV A,A0
                MOV DPTR,#GET_R2
                MOVC A,@A+DPTR
                MOV R2,A ;GET A00-A07

                JB STATE,GO_V01_PROC
; ======= STATE 0(1) PROCESS =======
                SETB STATE
        ; ------ GET A00 - A07 TABLE -----
                MOV DPTR,#A0_WAY_0
                RL A
                INC A
                MOVC A,@A+DPTR
                PUSH ACC
                MOV A,R2
                RL A
                MOVC A,@A+DPTR
                MOV DPH1,A
                POP ACC
                MOV DPL1,A
        ; ------ GET MASK ------
MASK_V00:       MOV A,@R1
                MOV DECODING_BYTE,A
        ;GET DATA AT REF. LINE
                MOVC A,@A+DPTR
                MOV MASK_CODE,A
                MOV DPTR,#A0_VALUE_0
                MOVC A,@A+DPTR
                ADD A,A0
                MOV A0,A
                MOV A,MASK_CODE
                ORL A,@R1
                MOV @R1,A
                JNB MASK_CHK,GO0_V0
                INC R1
                MOV REF_OBYTE,REF_NBYTE
                MOV REF_NBYTE,@R1
                CJNE R1,#4EH,V00_CHK_STATE
                MOV R1,#REF_LINE  ; NEW LINE
                MOV DPS,#00H
                REPT 14
                MOV A,@R1
                MOVX @DPTR,A
                INC DPTR
                INC R1
                ENDM
                MOV R1,#REF_LINE
                MOV A0,#00H
                MOV DPS,#01H
                INC CODE_LINE_NUM
                MOV A,CODE_LINE_NUM
                CJNE A,#ROW,GO0_V0
                JMP END_PIC
V00_CHK_STATE:  JB DECODING_BCHK,V00_MOV
                MOV DPTR,#A0_TAB0
                JMP MASK_V00
GO0_V0:         JMP END_V0
        ;RETURN TO NEW COMPRESS DATA

V00_MOV:        MOV A,@R1
                CJNE A,#0FFH,LAST_V00
                MOV A,A0
                ADD A,#08H
                MOV A0,A
                INC R1
                MOV REF_OBYTE,REF_NBYTE
                MOV REF_NBYTE,@R1
                CJNE R1,#4EH,V00_MOV
                MOV R1,#REF_LINE  ; NEW LINE
                MOV DPS,#00H
                REPT 14
                MOV A,@R1
                MOVX @DPTR,A
                INC DPTR
                INC R1
                ENDM
                MOV R1,#REF_LINE
                MOV DPS,#01H
                MOV A0,#00H
                INC CODE_LINE_NUM
                MOV A,CODE_LINE_NUM
                CJNE A,#ROW,GO0_V0
                JMP END_PIC
LAST_V00:       MOV DPTR,#A0_V_END0
                MOVC A,@A+DPTR
                ADD A,A0
                MOV A0,A
                JMP END_V0
; ======= STATE 1(0) PROCESS =======
V01_PROC:       CLR STATE
        ; ------ GET A00 - A07 TABLE -----
                MOV DPTR,#A0_WAY_1
                RL A
                INC A
                MOVC A,@A+DPTR
                PUSH ACC
                MOV A,R2
                RL A
                MOVC A,@A+DPTR
                MOV DPH1,A
                POP ACC
                MOV DPL1,A
        ; ------ GET MASK ------
MASK_V01:       MOV A,@R1
                MOV DECODING_BYTE,A
                MOVC A,@A+DPTR
                MOV MASK_CODE,A
                MOV DPTR,#A0_VALUE_1
                MOVC A,@A+DPTR
                ADD A,A0
                MOV A0,A

                MOV A,MASK_CODE
                ANL A,@R1
                MOV @R1,A

                JB MASK_CHK,END_V0
                INC R1
                MOV REF_OBYTE,REF_NBYTE
                MOV REF_NBYTE,@R1
                CJNE R1,#4EH,V01_CHK_STATE
                MOV R1,#REF_LINE  ; NEW LINE
                MOV DPS,#00H
                REPT 14
                MOV A,@R1
                MOVX @DPTR,A
                INC DPTR
                INC R1
                ENDM
```

```
                MOV R1,#REF_LINE
                MOV DPS,#01H
                MOV A,#00H
                INC CODE_LINE_NUM
                MOV A,CODE_LINE_NUM
                CJNE A,#ROW,END_V0
                JMP END_PIC
V01_CHK_STATE:  JNB DECODING_BCHK,V01_MOV
                MOV A,@R1
                MOV DPTR,#A0_TAB1
                JMP MASK_V01
END_V0:         MOV A,CODE_TMP
                JMP DECOM_DATA
V01_MOV:        MOV A,@R1
                CJNE A,#00H,LAST_V01
                MOV A,A0
                ADD A,#08H
                MOV A0,A
                INC R1
                MOV REF_OBYTE,REF_NBYTE
                MOV REF_NBYTE,@R1
                CJNE R1,#4EH,V01_MOV
                MOV R1,#REF_LINE  ; NEW LINE
                MOV DPS,#00H
                REPT 14
                MOV A,@R1
                MOVX @DPTR,A
                INC DPTR
                INC R1
                ENDM
                MOV R1,#REF_LINE
                MOV A0,#00H
                MOV DPS,#01H
                INC CODE_LINE_NUM
                MOV A,CODE_LINE_NUM
                CJNE A,#ROW,END_V0
                JMP END_PIC
LAST_V01:       MOV DPTR,#A0_V_END1
                MOVC A,@A+DPTR
                ADD A,A0
                MOV A0,A
                JMP END_V0
;RETURN TO NEW COMPRESS DATA
;================ V1 MODE =================
V1_MODE:        RLC A
                DJNZ BIT_C,SEL0_V1
                MOV BIT_C,#08H
                INC R0
                DJNZ R6,CONTS_08
                MOV R6,#0FFH
                DJNZ R7,CONTS_08
                JMP END_PIC
CONTS_08:       JNB TMP_DATA,CONTD_07
                MOV R0,#DATA_RECV
                MOV R5,#RECV_SIZE
                CLR TMP_DATA
                MOV A,@R0
                JMP SEL0_V1
CONTD_07:       MOV A,@R0
                DJNZ R5,SEL0_V1
                JC S_N0V1
        ; GO TO VL1_MODE IN NEXT TIME
                MOV S_BYTE,#0DH
                JMP END_48_BYTES
S_N0V1:  ; GO TO VR1_MODE IN NEXT TIME
                MOV S_BYTE,#0EH
                JMP END_48_BYTES
GO_VL11_PROC:   JMP VL11_PROC
GO_VR1:         JMP VR1_MODE
SEL0_V1:        JC GO_VR1

VL1_MODE:               MOV CODE_TMP,A
        ; ------ GET REF. LINE INDEX -----
                MOV A,A0
                MOV DPTR,#GET_R1
                MOVC A,@A+DPTR
                MOV R1,A
        ;GET REF. LINE INDEX
        ; ------ GET A00 - A07 -------
                MOV A,A0
                MOV DPTR,#GET_R2
                MOVC A,@A+DPTR
                MOV R2,A
        ;GET A00-A07
                CJNE A,#00H,CONTS_VL1
                MOV REF_OBYTE,REF_NBYTE
                MOV REF_NBYTE,@R1
CONTS_VL1:      JB STATE,GO_VL11_PROC
; ======= STATE 0(1) PROCESS =======
                SETB STATE
        ; ------ GET A00 - A07 TABLE -----
                MOV DPTR,#A0_WAY_0
                RL A
                INC A
                MOVC A,@A+DPTR
                PUSH ACC
                MOV A,R2
                RL A
                MOVC A,@A+DPTR
                MOV DPH1,A
                POP ACC
                MOV DPL1,A
        ; ------ GET MASK ------
MASK_VL10:      MOV A,@R1
                MOV DECODING_BYTE,A
        ;GET DATA AT REF. LINE
                MOVC A,@A+DPTR
                MOV MASK_CODE,A

                MOV DPTR,#A0_VALUE_0
                MOVC A,@A+DPTR
                ADD A,A0
                MOV A0,A

                MOV A,MASK_CODE
                ORL A,@R1
                MOV @R1,A

                JNB MASK_CHK,GO0_VL1
                INC R1
                MOV REF_OBYTE,REF_NBYTE
                MOV REF_NBYTE,@R1
                CJNE R1,#4EH,VL10_CHKSTATE
                MOV R1,#REF_LINE  ; NEW LINE
                MOV DPS,#00H
                REPT 14
                MOV A,@R1
                MOVX @DPTR,A
                INC DPTR
                INC R1
                ENDM
                MOV R1,#REF_LINE
                MOV DPS,#01H
                MOV A0,#00H
                INC CODE_LINE_NUM
                MOV A,CODE_LINE_NUM
                CJNE A,#ROW,GO0_VL1
                JMP END_PIC
VL10_CHK_STATE:JB DECODING_BCHK,VL10_MOV
                MOV A,@R1
                MOV DPTR,#A0_TAB0
                JMP MASK_VL10
GO0_VL1:        JMP END_VL1
;RETURN TO NEW COMPRESS DATA

VL10_MOV:       MOV A,@R1
```

```
                CJNE A,#0FFH,LAST_VL10
                MOV A,A0
                ADD A,#08H
                MOV A0,A
                INC R1
                MOV REF_OBYTE,REF_NBYTE
                MOV REF_NBYTE,@R1
                CJNE R1,#4EH,VL10_MOV
                MOV R1,#REF_LINE  ; NEW LINE
                MOV DPS,#00H
                REPT 14
                MOV A,@R1
                MOVX @DPTR,A
                INC DPTR
                INC R1
                ENDM
                MOV R1,#REF_LINE
                MOV DPS,#01H
                MOV A0,#00H
                INC CODE_LINE_NUM
                MOV A,CODE_LINE_NUM
                CJNE A,#ROW,GO0_VL1
                JMP END_PIC
LAST_VL10:      MOV DPTR,#A0_V_END0
                MOVC A,@A+DPTR
                ADD A,A0
                MOV A0,A
                JMP END_VL1
; ======= STATE 1(0) PROCESS =======
VL11_PROC:      CLR STATE
        ; ------ GET A00 - A07 TABLE -----
                MOV DPTR,#A0_WAY_1
                RL A
                INC A
                MOVC A,@A+DPTR
                PUSH ACC
                MOV A,R2
                RL A
                MOVC A,@A+DPTR
                MOV DPH1,A
                POP ACC
                MOV DPL1,A

        ; ------ GET MASK ------
MASK_VL11:      MOV A,@R1
                MOV DECODING_BYTE,A
                MOVC A,@A+DPTR
                MOV MASK_CODE,A

                MOV DPTR,#A0_VALUE_1
                MOVC A,@A+DPTR
                ADD A,A0
                MOV A0,A

                MOV A,MASK_CODE
                ANL A,@R1
                MOV @R1,A

                JB MASK_CHK,END_VL1
                INC R1
                MOV REF_OBYTE,REF_NBYTE
                MOV REF_NBYTE,@R1
                CJNER1,#4EH,VL11_CHK_STATE
                MOV R1,#REF_LINE  ; NEW LINE
                MOV DPS,#00H
                REPT 14
                MOV A,@R1
                MOVX @DPTR,A
                INC DPTR
                INC R1
                ENDM
                MOV R1,#REF_LINE
                INC CODE_LINE_NUM
                MOV DPS,#01H
                MOV A0,#00H
                MOV A,CODE_LINE_NUM
                CJNE A,#ROW,END_VL1
                JMP END_PIC
VL11_CHK_STATE:JNB DECODING_BCHK,
                      VL11_MOV
                MOV A,@R1
                MOV DPTR,#A0_TAB1
                JMP MASK_VL11
END_VL1:        DEC A0
                MOV A,A0
                MOV DPTR,#GET_R1
                MOVC A,@A+DPTR
                MOV R1,A
                ;GET REF. LINE INDEX
                MOV A,A0
                MOV DPTR,#GET_R2
                MOVC A,@A+DPTR
                MOV R2,A
                MOV DPTR,#VL1_MASK
                RL A
                INC A
                MOVC A,@A+DPTR
                PUSH ACC
                MOV A,R2
                RL A
                MOVC A,@A+DPTR
                PUSH ACC
                RET

VL11_MOV:       MOV A,@R1
                CJNE A,#00H,LAST_VL11
                MOV A,A0
                ADD A,#08H
                MOV A0,A
                INC R1
                MOV REF_OBYTE,REF_NBYTE
                MOV REF_NBYTE,@R1
                CJNE R1,#4EH,VL11_MOV
                MOV R1,#REF_LINE
                 NEW LINE
                MOV DPS,#00H
                REPT 14
                MOV A,@R1
                MOVX @DPTR,A
                INC DPTR
                INC R1
                ENDM
                MOV R1,#REF_LINE
                MOV DPS,#01H
                MOV A0,#00H
                INC CODE_LINE_NUM
                MOV A,CODE_LINE_NUM
                CJNE A,#ROW,GO1_VL1
                JMP END_PIC
LAST_VL11:      MOV DPTR,#A0_V_END1
                MOVC A,@A+DPTR
                ADD A,A0
                MOV A0,A
GO1_VL1:        JMP END_VL1

VL1_MASK0:      MOV A,#10111111B
                ANL A,@R1
                MOV @R1,A
                MOV A,REF_NBYTE
                ANL A,#01000000B
                ORL A,@R1
                MOV @R1,A
                MOV A,CODE_TMP
                JMP DECOM_DATA
VL1_MASK1:      MOV A,#11011111B
```

```
                ANL A,@R1
                MOV @R1,A
                MOV A,REF_NBYTE
                ANL A,#00100000B
                ORL A,@R1
                MOV @R1,A
                MOV A,CODE_TMP
                JMP DECOM_DATA

VL1_MASK2:      MOV A,#11101111B
                ANL A,@R1
                MOV @R1,A
                MOV A,REF_NBYTE
                ANL A,#00010000B
                ORL A,@R1
                MOV @R1,A
                MOV A,CODE_TMP
                JMP DECOM_DATA

VL1_MASK3:      MOV A,#11110111B
                ANL A,@R1
                MOV @R1,A
                MOV A,REF_NBYTE
                ANL A,#00001000B
                ORL A,@R1
                MOV @R1,A
                MOV A,CODE_TMP
                JMP DECOM_DATA

VL1_MASK4:      MOV A,#11111011B
                ANL A,@R1
                MOV @R1,A
                MOV A,REF_NBYTE
                ANL A,#00000100B
                ORL A,@R1
                MOV @R1,A
                MOV A,CODE_TMP
                JMP DECOM_DATA

VL1_MASK5:      MOV A,#11111101B
                ANL A,@R1
                MOV @R1,A
                MOV A,REF_NBYTE
                ANL A,#00000010B
                ORL A,@R1
                MOV @R1,A
                MOV A,CODE_TMP
                JMP DECOM_DATA

VL1_MASK6:      MOV A,#11111110B
                ANL A,@R1
                MOV @R1,A
                MOV A,CODE_TMP
                JMP DECOM_DATA

VL1_MASK7:      MOV A,#01111111B
                ANL A,@R1
                MOV @R1,A
                MOV A,REF_NBYTE
                ANL A,#10000000B
                ORL A,@R1
                MOV @R1,A
                MOV A,CODE_TMP
                JMP DECOM_DATA
        ;=================
GO_VR11_PROC:   JMP VR11_PROC

VR1_MODE:       MOV CODE_TMP,A
        ; ------ GET REF. LINE INDEX -----
                MOV A,A0
                MOV DPTR,#GET_R1
                MOVC A,@A+DPTR
                MOV R1,A ;GET REF. LINE INDEX
        ; ------ GET A00 - A07 -------
                MOV A,A0
                MOV DPTR,#GET_R2
                MOVC A,@A+DPTR
                MOV R2,A  ;GET A00-A07
                CJNE A,#00H,CONTS_VR1
                MOV REF_OBYTE,REF_NBYTE
                MOV REF_NBYTE,@R1
CONTS_VR1:      JB STATE,GO_VR11_PROC
        ; ======= STATE 0(1) PROCESS =======
                SETB STATE
        ; ------ GET A00 - A07 TABLE -----
                MOV DPTR,#A0_WAY_0
                RL A
                INC A
                MOVC A,@A+DPTR
                PUSH ACC
                MOV A,R2
                RL A
                MOVC A,@A+DPTR
                MOV DPH1,A
                POP ACC
                MOV DPL1,A
        ; ------ GET MASK ------
MASK_VR10:      MOV A,@R1
                MOV DECODING_BYTE,A
        ;GET DATA AT REF. LINE
                MOVC A,@A+DPTR
                MOV MASK_CODE,A
                MOV DPTR,#A0_VALUE_0
                MOVC A,@A+DPTR
                ADD A,A0
                MOV A0,A
                MOV A,MASK_CODE
                ORL A,@R1
                MOV @R1,A

                JNB MASK_CHK,
                    END_MASK_VR10
                INC R1
                MOV REF_OBYTE,REF_NBYTE
                MOV REF_NBYTE,@R1
                CJNE R1,#4EH,
                     VR10_CHK_STATE
                MOV R1,#REF_LINE  ; NEW LINE
                MOV DPS,#00H
                REPT 14
                MOV A,@R1
                MOVX @DPTR,A
                INC DPTR
                INC R1
                ENDM
                MOV R1,#REF_LINE
                MOV DPS,#01H
                MOV A0,#00H
                INC CODE_LINE_NUM
                MOV A,CODE_LINE_NUM
                CJNE A,#ROW,GO_END_VR10
                JMP END_PIC

VR10_CHK_STATE:JB DECODING_BCHK,
                  VR10_MOV
                MOV A,@R1
                MOV DPTR,#A0_TAB0
                JMP MASK_VR10

GO_END_VR10:    JMP END_VR1

END_MASK_VR10: MOV A,MASK_CODE
                RR A
                ORL A,MASK_CODE
                ORL A,@R1
                MOV @R1,A
```

```
                INC A0
                JMP END_VR1

VR10_MOV:       MOV A,@R1
                CJNE A,#0FFH,LAST_VR10
                MOV A,A0
                ADD A,#08H
                MOV A0,A
                INC R1
                MOV REF_OBYTE,REF_NBYTE
                MOV REF_NBYTE,@R1
                CJNE R1,#4EH,VR10_MOV
                MOV R1,#REF_LINE  ; NEW LINE
                MOV DPS,#00H
                REPT 14
                MOV A,@R1
                MOVX @DPTR,A
                INC DPTR
                INC R1
                ENDM
                MOV R1,#REF_LINE
                MOV DPS,#01H
                MOV A0,#00H
                INC CODE_LINE_NUM
                MOV A,CODE_LINE_NUM
                CJNE A,#ROW,GO_END_VR10
                JMP END_PIC

LAST_VR10:      MOV DPTR,#A0_V_END0
                MOVC A,@A+DPTR
                CJNE A,#00H,N_MASK_VR10
                INC A0
                MOV A,@R1
                ORL A,#80H
                MOV @R1,A
                JMP END_VR1

N_MASK_VR10:    INC A
                ADD A,A0
                MOV A0,A
                MOV A,@R1
                MOV DPTR,#VR10_MASK_TAB
                MOVC A,@A+DPTR
                ORL A,@R1
                MOV @R1,A
                JMP END_VR1
; ======= STATE 1(0) PROCESS =======
VR11_PROC:      CLR STATE
        ; ------ GET A00 - A07 TABLE -----
                MOV DPTR,#A0_WAY_1
                RL A
                INC A
                MOVC A,@A+DPTR
                PUSH ACC
                MOV A,R2
                RL A
                MOVC A,@A+DPTR
                MOV DPH1,A
                POP ACC
                MOV DPL1,A

        ; ------ GET MASK ------
MASK_VR11:      MOV A,@R1
                MOV DECODING_BYTE,A
                MOVC A,@A+DPTR
                MOV MASK_CODE,A
                MOV DPTR,#A0_VALUE_1
                MOVC A,@A+DPTR
                ADD A,A0
                MOV A0,A

                MOV A,MASK_CODE
                ANL A,@R1
                MOV @R1,A

                JB MASK_CHK,END_MASK_VR11
                INC R1
                MOV REF_OBYTE,REF_NBYTE
                MOV REF_NBYTE,@R1
                CJNE R1,#4EH,
                        VR11_CHK_STATE
                MOV R1,#REF_LINE  ; NEW LINE
                MOV DPS,#00H
                REPT 14
                MOV A,@R1
                MOVX @DPTR,A
                INC DPTR
                INC R1
                ENDM
                MOV R1,#REF_LINE
                MOV DPS,#01H
                MOV A0,#00H
                INC CODE_LINE_NUM
                MOV A,CODE_LINE_NUM
                CJNE A,#ROW,END_VR1
                JMP END_PIC

VR11_CHK_STATE: JNB DECODING_BCHK,
                        VR11_MOV
                MOV A,@R1
                MOV DPTR,#A0_TAB1
                JMP MASK_VR11

END_VR1:        MOV A,CODE_TMP
                JMP DECOM_DATA

END_MASK_VR11:  MOV A,MASK_CODE
                RR A
                ANL A,MASK_CODE
                ANL A,@R1
                MOV @R1,A
                INC A0
                JMP END_VR1

VR11_MOV:       MOV A,@R1
                CJNE A,#00H,LAST_VR11
                MOV A,A0
                ADD A,#08H
                MOV A0,A
                INC R1
                MOV REF_OBYTE,REF_NBYTE
                MOV REF_NBYTE,@R1
                CJNE R1,#4EH,VR11_MOV
                MOV R1,#REF_LINE  ; NEW LINE
                MOV DPS,#00H
                REPT 14
                MOV A,@R1
                MOVX @DPTR,A
                INC DPTR
                INC R1
                ENDM
                MOV R1,#REF_LINE
                MOV DPS,#00H
                MOV A,#00H
                INC CODE_LINE_NUM
                MOV A,CODE_LINE_NUM
                CJNE A,#ROW,END_VR1
                JMP END_PIC

LAST_VR11:      MOV DPTR,#A0_V_END1
                MOVC A,@A+DPTR
                CJNE A,#00H,N_MASK_VR11
                INC A0
                MOV A,@R1
                ANL A,#7FH
                MOV @R1,A
```

```
                JMP END_VR1
N_MASK_VR11:    INC A
                ADD A,A0
                MOV A0,A
                MOV A,@R1
                MOV DPTR,#VR11_MASK_TAB
                MOVC A,@A+DPTR
                ANL A,@R1
                MOV @R1,A
                JMP END_VR1
;============== V2 MODE =================
V2_MODE:        RLC A
                DJNZ BIT_C,SEL0_V2
                MOV BIT_C,#08H
                INC R0
                DJNZ R6,CONTS_09
                MOV R6,#0FFH
                DJNZ R7,CONTS_09
                JMP END_PIC
CONTS_09:       JNB TMP_DATA,CONTD_08
                MOV R0,#DATA_RECV
                MOV R5,#RECV_SIZE
                CLR TMP_DATA
                MOV A,@R0
                JMP SEL0_V2
CONTD_08:       MOV A,@R0
                DJNZ R5,SEL0_V2
                JC S_N0V2
        ; GO TO VL2_MODE IN NEXT TIME
                MOV S_BYTE,#0FH
                JMP END_48_BYTES
S_N0V2:
; GO TO VR2_MODE IN NEXT TIME
                MOV S_BYTE,#10H
                JMP END_48_BYTES

GO_VL21_PROC:   JMP VL21_PROC
GO_VR2:         JMP VR2_MODE
SEL0_V2:        JC GO_VR2

VL2_MODE:       MOV CODE_TMP,A
        ; ------ GET REF. LINE INDEX -----
                MOV A,A0
                MOV DPTR,#GET_R1
                MOVC A,@A+DPTR
                MOV R1,A ;GET REF. LINE INDEX
        ; ------ GET A00 - A07 -------
                MOV A,A0
                MOV DPTR,#GET_R2
                MOVC A,@A+DPTR
                MOV R2,A  ;GET A00-A07
                CJNE A,#00H,CONTS_VL2
                MOV REF_OBYTE,REF_BYTE
                MOV REF_NBYTE,@R1
CONTS_VL2:      JB STATE,GO_VL21_PROC
; ======= STATE 0(1) PROCESS =======
                SETB STATE
        ; ------ GET A00 - A07 TABLE -----
                MOV DPTR,#A0_WAY_0
                RL A
                INC A
                MOVC A,@A+DPTR
                PUSH ACC
                MOV A,R2
                RL A
                MOVC A,@A+DPTR
                MOV DPH1,A
                POP ACC
                MOV DPL1,A
        ; ------ GET MASK ------
MASK_VL20:      MOV A,@R1
                MOV DECODING_BYTE,A
        ;GET DATA AT REF. LINE
                MOVC A,@A+DPTR
                MOV MASK_CODE,A
                MOV DPTR,#A0_VALUE_0
                MOVC A,@A+DPTR
                ADD A,A0
                MOV A0,A

                MOV A,MASK_CODE
                ORL A,@R1
                MOV @R1,A

                JNB MASK_CHK,GO0_VL2
                INC R1
                MOV REF_OBYTE,REF_NBYTE
                MOV REF_NBYTE,@R1
                CJNE R1,#4EH,
                        VL20_CHK_STATE
                MOV R1,#REF_LINE  ; NEW LINE
                MOV DPS,#00H
                REPT 14
                MOV A,@R1
                MOVX @DPTR,A
                INC DPTR
                INC R1
                ENDM
                MOV R1,#REF_LINE
                MOV DPS,#01H
                MOV A0,#00H
                INC CODE_LINE_NUM
                MOV A,CODE_LINE_NUM
                CJNE A,#ROW,GO0_VL2
                JMP END_PIC
VL20_CHK_STATE:JB DECODING_BCHK,VL20_MOV
                MOV A,@R1
                MOV DPTR,#A0_TAB0
                JMP MASK_VL20

GO0_VL2:        JMP END_VL2
                ;RETURN TO NEW COMPRESS DATA
VL20_MOV:       MOV A,@R1
                CJNE A,#0FFH,LAST_VL20
                MOV A,A0
                ADD A,#08H
                MOV A0,A
                INC R1
                MOV REF_OBYTE,REF_NBYTE
                MOV REF_NBYTE,@R1
                CJNE R1,#4EH,VL20_MOV
                MOV R1,#REF_LINE  ; NEW LINE
                MOV DPS,#00H
                REPT 14
                MOV A,@R1
                MOVX @DPTR,A
                INC DPTR
                INC R1
                ENDM
                MOV R1,#REF_LINE
                MOV DPS,#01H
                MOV A0,#00H
                INC CODE_LINE_NUM
                MOV A,CODE_LINE_NUM
                CJNE A,#ROW,GO0_VL2
                JMP END_PIC

LAST_VL20:      MOV DPTR,#A0_V_END0
                MOVC A,@A+DPTR
                ADD A,A0
                MOV A0,A
                JMP END_VL2
; ======= STATE 1(0) PROCESS =======
VL21_PROC:      CLR STATE
```

```
        ; ------ GET A00 - A07 TABLE -----
                MOV DPTR,#A0_WAY_1
                RL A
                INC A
                MOVC A,@A+DPTR
                PUSH ACC
                MOV A,R2
                RL A
                MOVC A,@A+DPTR
                MOV DPH1,A
                POP ACC
                MOV DPL1,A

        ; ------ GET MASK ------
MASK_VL21:      MOV A,@R1
                MOV DECODING_BYTE,A
                MOVC A,@A+DPTR
                MOV MASK_CODE,A
                MOV DPTR,#A0_VALUE_1
                MOVC A,@A+DPTR
                ADD A,A0
                MOV A0,A

                MOV A,MASK_CODE
                ANL A,@R1
                MOV @R1,A

                JB MASK_CHK,END_VL2
                INC R1
                MOV REF_OBYTE,REF_NBYTE
                MOV REF_NBYTE,@R1
                CJNE R1,#4EH,
                      VL21_CHK_STATE
                MOV R1,#REF_LINE  ; NEW LINE
                MOV DPS,#00H
                REPT 14
                MOV A,@R1
                MOVX @DPTR,A
                INC DPTR
                INC R1
                ENDM
                MOV R1,#REF_LINE
                MOV DPS,#01H
                MOV A0,#00H
                INC CODE_LINE_NUM
                MOV A,CODE_LINE_NUM
                CJNE A,#ROW,END_VL2
                JMP END_PIC

VL21_CHK_STATE: JNB DECODING_BCHK,
                      VL21_MOV
                MOV A,@R1
                MOV DPTR,#A0_TAB1
                JMP MASK_VL21

END_VL2:        DEC A0
                DEC A0
                MOV A,A0
                MOV DPTR,#GET_R1
                MOVC A,@A+DPTR
                MOV R1,A
        ;GET REF. LINE INDEX
                MOV A,A0
                MOV DPTR,#GET_R2
                MOVC A,@A+DPTR
                MOV R2,A
                MOV DPTR,#VL2_MASK
                RL A
                INC A
                MOVC A,@A+DPTR
                PUSH ACC
                MOV A,R2
                RL A
                MOVC A,@A+DPTR
                PUSH ACC
                RET

VL21_MOV:       MOV A,@R1
                CJNE A,#00H,LAST_VL21
                MOV A,A0
                ADD A,#08H
                MOV A0,A
                INC R1
                MOV REF_OBYTE,REF_NBYTE
                MOV REF_NBYTE,@R1
                CJNE R1,#4EH,VL21_MOV
                MOV R1,#REF_LINE  ; NEW LINE
                MOV DPS,#00H
                REPT 14
                MOV A,@R1
                MOVX @DPTR,A
                INC DPTR
                INC R1
                ENDM
                MOV R1,#REF_LINE
                MOV DPS,#01H
                MOV A0,#00H
                INC CODE_LINE_NUM
                MOV A,CODE_LINE_NUM
                CJNE A,#ROW,GO1_VL2
                JMP END_PIC

LAST_VL21:      MOV DPTR,#A0_V_END1
                MOVC A,@A+DPTR
                ADD A,A0
                MOV A0,A
GO1_VL2:        JMP END_VL2

VL2_MASK0:      MOV A,#10011111B
                ANL A,@R1
                MOV @R1,A
                MOV A,REF_NBYTE
                ANL A,#01100000B
                ORL A,@R1
                MOV @R1,A
                MOV A,CODE_TMP
                JMP DECOM_DATA

VL2_MASK1:      MOV A,#11001111B
                ANL A,@R1
                MOV @R1,A
                MOV A,REF_NBYTE
                ANL A,#00110000B
                ORL A,@R1
                MOV @R1,A
                MOV A,CODE_TMP
                JMP DECOM_DATA

VL2_MASK2:      MOV A,#11100111B
                ANL A,@R1
                MOV @R1,A
                MOV A,REF_NBYTE
                ANL A,#00011000B
                ORL A,@R1
                MOV @R1,A
                MOV A,CODE_TMP
                JMP DECOM_DATA

VL2_MASK3:      MOV A,#11110011B
                ANL A,@R1
                MOV @R1,A
                MOV A,REF_NBYTE
                ANL A,#00001100B
                ORL A,@R1
                MOV @R1,A
                MOV A,CODE_TMP
                JMP DECOM_DATA
```

```
VL2_MASK4:      MOV A,#11111001B
                ANL A,@R1
                MOV @R1,A
                MOV A,REF_NBYTE
                ANL A,#00000110B
                ORL A,@R1
                MOV @R1,A
                MOV A,CODE_TMP
                JMP DECOM_DATA
VL2_MASK5:      MOV A,#11111100B
                ANL A,@R1
                MOV @R1,A
                JMP DECOM_DATA
VL2_MASK6:      MOV A,#11111110B
                ANL A,@R1
                MOV @R1,A
                MOV A,REF_OBYTE
                ANL A,#00000001B
                ORL A,@R1
                MOV @R1,A
                INC R1
                MOV A,#01111111B
                ANL A,@R1
                MOV @R1,A
                MOV A,REF_NBYTE
                ANL A,#10000000B
                ORL A,@R1
                MOV @R1,A
                MOV REF_NBYTE,REF_OBYTE
                MOV A,CODE_TMP
                JMP DECOM_DATA
VL2_MASK7:      MOV A,#00111111B
                ANL A,@R1
                MOV @R1,A
                MOV A,REF_NBYTE
                ANL A,#11000000B
                ORL A,@R1
                MOV @R1,A
                MOV REF_NBYTE,REF_OBYTE
                MOV A,CODE_TMP
                JMP DECOM_DATA
        ;=================

GO_VR21_PROC:   JMP VR21_PROC
VR2_MODE:       MOV CODE_TMP,A
        ; ------ GET REF. LINE INDEX -----
                MOV A,A0
                MOV DPTR,#GET_R1
                MOVC A,@A+DPTR
                MOV R1,A ;GET REF. LINE INDEX
        ; ------ GET A00 - A07 -------
                MOV A,A0
                MOV DPTR,#GET_R2
                MOVC A,@A+DPTR
                MOV R2,A  ;GET A00-A07
                CJNE A,#00H,CONTS_VR2
                MOV REF_OBYTE,REF_NBYTE
                MOV REF_NBYTE,@R1
CONTS_VR2:      JB STATE,GO_VR21_PROC
; ======= STATE 0(1) PROCESS =======
                SETB STATE
        ; ------ GET A00 - A07 TABLE -----
                MOV DPTR,#A0_WAY_0
                RL A
                INC A
                MOVC A,@A+DPTR
                PUSH ACC
                MOV A,R2
                RL A
                MOVC A,@A+DPTR
                MOV DPH1,A
                POP ACC
                MOV DPL1,A

        ; ------ GET MASK ------
MASK_VR20:      MOV A,@R1
                MOV DECODING_BYTE,A
        ;GET DATA AT REF. LINE
                MOVC A,@A+DPTR
                MOV MASK_CODE,A
                MOV DPTR,#A0_VALUE_0
                MOVC A,@A+DPTR
                ADD A,A0
                MOV A0,A

                MOV A,MASK_CODE
                ORL A,@R1
                MOV @R1,A

                JNB MASK_CHK,
                    END_MASK_VR20
                INC R1
                MOV REF_OBYTE,REF_NBYTE
                MOV REF_NBYTE,@R1
                CJNE R1,#4EH,
                     VR20_CHK_STATE
                MOV R1,#REF_LINE  ; NEW LINE
                MOV DPS,#00H
                REPT 14
                MOV A,@R1
                MOVX @DPTR,A
                INC DPTR
                INC R1
                ENDM
                MOV R1,#REF_LINE
                MOV DPS,#01H
                MOV A0,#00H
                INC CODE_LINE_NUM
                MOV A,CODE_LINE_NUM
                CJNE A,#ROW,GO_END_VR20
                JMP END_PIC

VR20_CHK_STATE: JB DECODING_BCHK,
                    VR20_MOV
                MOV A,@R1
                MOV DPTR,#A0_TAB0
                JMP MASK_VR20

GO_END_VR20:    JMP END_VR2

END_MASK_VR20:  MOV A,MASK_CODE
                RR A
                ORL A,MASK_CODE
                MOV MASK_CODE,A
                ORL A,@R1
                MOV @R1,A
                JB MASK_CHK,L_MASK_VR20
                MOV A,MASK_CODE
                RR A
                ORL A,MASK_CODE
                ORL A,@R1
                MOV @R1,A
                INC A0
                INC A0
                JMP END_VR2
L_MASK_VR20:    INC R1
                MOV REF_OBYTE,REF_NBYTE
                MOV REF_NBYTE,@R1
                MOV A,@R1
                ORL A,#80H
                MOV @R1,A
                INC A0
                INC A0
```

```
GO1_END_VR20:   JMP END_VR2

VR20_MOV:       MOV A,@R1
                CJNE A,#0FFH,LAST_VR20
                MOV A,A0
                ADD A,#08H
                MOV A0,A
                INC R1
                MOV REF_OBYTE,REF_NBYTE
                MOV REF_NBYTE,@R1
                CJNE R1,#4EH,VR20_MOV
                MOV R1,#REF_LINE  ; NEW LINE
                MOV DPS,#00H
                REPT 14
                MOV A,@R1
                MOVX @DPTR,A
                INC DPTR
                INC R1
                ENDM
                MOV R1,#REF_LINE
                MOV DPS,#01H
                MOV A0,#00H
                INC CODE_LINE_NUM
                MOV A,CODE_LINE_NUM
                CJNE A,#ROW,GO1_END_VR20
                JMP END_PIC

LAST_VR20:      CJNE A,#0FEH,CHK_FF_VR20
                MOV DPTR,#A0_V_END0
                MOVC A,@A+DPTR
                INC A
                INC A
                ADD A,A0
                MOV A0,A
                MOV A,@R1
                ORL A,#01H
                MOV @R1,A
                INC R1
                MOV REF_OBYTE,REF_NBYTE
                MOV REF_NBYTE,@R1
                MOV A,@R1
                ORL A,#80H
                MOV @R1,A
                JMP END_VR2

CHK_FF_VR20:    MOV DPTR,#A0_V_END0
                MOVC A,@A+DPTR
                CJNE A,#00H,N_MASK_VR20
                INC A0
                INC A0
                MOV REF_OBYTE,REF_NBYTE
                MOV REF_NBYTE,@R1
                MOV A,@R1
                ORL A,#0C0H
                MOV @R1,A
                JMP END_VR2

N_MASK_VR20:    INC A
                INC A
                ADD A,A0
                MOV A0,A
                MOV A,@R1
                MOV DPTR,#VR20_MASK_TAB
                MOVC A,@A+DPTR
                ORL A,@R1
                MOV @R1,A
                JMP END_VR2
; ======= STATE 1(0) PROCESS =======
VR21_PROC:      CLR STATE
          ; ------ GET A00 - A07 TABLE -----
                MOV DPTR,#A0_WAY_1
                RL A
                INC A
                MOVC A,@A+DPTR
                PUSH ACC
                MOV A,R2
                RL A
                MOVC A,@A+DPTR
                MOV DPH1,A
                POP ACC
                MOV DPL1,A

          ; ------ GET MASK ------
MASK_VR21:      MOV A,@R1
                MOV DECODING_BYTE,A
                MOVC A,@A+DPTR
                MOV MASK_CODE,A
                MOV DPTR,#A0_VALUE_1
                MOVC A,@A+DPTR
                ADD A,A0
                MOV A0,A

                MOV A,MASK_CODE
                ANL A,@R1
                MOV @R1,A

                JB MASK_CHK,END_MASK_VR21
                INC R1
                MOV REF_OBYTE,REF_NBYTE
                MOV REF_NBYTE,@R1
                CJNE R1,#4EH,
                      VR21_CHK_STATE
                MOV R1,#REF_LINE  ; NEW LINE
                MOV DPS,#00H
                REPT 14
                MOV A,@R1
                MOVX @DPTR,A
                INC DPTR
                INC R1
                ENDM
                MOV R1,#REF_LINE
                MOV DPS,#01H
                MOV A0,#00H
                INC CODE_LINE_NUM
                MOV A,CODE_LINE_NUM
                CJNE A,#ROW,END_VR2
                JMP END_PIC
VR21_CHK_STATE: JNB DECODING_BCHK,
                      VR21_MOV
                MOV A,@R1
                MOV DPTR,#A0_TAB1
                JMP MASK_VR21
END_VR2:        MOV A,CODE_TMP
                JMP DECOM_DATA
END_MASK_VR21:  MOV A,MASK_CODE
                RR A
                ANL A,MASK_CODE
                MOV MASK_CODE,A
                ANL A,@R1
                MOV @R1,A
                JNB MASK_CHK,L_MASK_VR21
                MOV A,MASK_CODE
                RR A
                ANL A,MASK_CODE
                ANL A,@R1
                MOV @R1,A
                INC A0
                INC A0
GO_END_VR21:    JMP END_VR2

L_MASK_VR21:    INC R1
                MOV REF_OBYTE,REF_NBYTE
                MOV REF_NBYTE,@R1
                MOV A,@R1
                ANL A,#7FH
                MOV @R1,A
```

```
                INC A0
                INC A0
                JMP END_VR2

VR21_MOV:       MOV A,@R1
                CJNE A,#00H,LAST_VR21
                MOV A,A0
                ADD A,#08H
                MOV A0,A
                INC R1
                MOV REF_OBYTE,REF_NBYTE
                MOV REF_NBYTE,@R1
                CJNE R1,#4EH,VR21_MOV
                MOV R1,#REF_LINE  ; NEW LINE
                MOV DPS,#00H
                REPT 14
                MOV A,@R1
                MOVX @DPTR,A
                INC DPTR
                INC R1
                ENDM
                MOV R1,#REF_LINE
                MOV DPS,#01H
                MOV A0,#00H
                INC CODE_LINE_NUM
                MOV A,CODE_LINE_NUM
                CJNE A,#ROW,GO_END_VR21
                JMP END_PIC

LAST_VR21:      CJNE A,#01H,CHK_00_VR21
                MOV DPTR,#A0_V_END1
                MOVC A,@A+DPTR
                INC A
                INC A
                ADD A,A0
                MOV A0,A
                MOV A,@R1
                ANL A,#0FEH
                MOV @R1,A
                INC R1
                MOV REF_OBYTE,REF_NBYTE
                MOV REF_NBYTE,@R1
                MOV A,@R1
                ANL A,#7FH
                MOV @R1,A
                JMP END_VR2

CHK_00_VR21:    MOV DPTR,#A0_V_END1
                MOVC A,@A+DPTR
                CJNE A,#00H,N_MASK_VR21
                INC A0
                INC A0
                MOV REF_OBYTE,REF_NBYTE
                MOV REF_NBYTE,@R1
                MOV A,@R1
                ANL A,#3FH
                MOV @R1,A
                JMP END_VR2

N_MASK_VR21:    INC A
                INC A
                ADD A,A0
                MOV A0,A
                MOV A,@R1
                MOV DPTR,#VR21_MASK_TAB
                MOVC A,@A+DPTR
                ANL A,@R1
                MOV @R1,A
                JMP END_VR2

;================ V3 MODE =================
V3_MODE:        RLC A
                DJNZ BIT_C,SEL0_V3
                MOV BIT_C,#08H
                INC R0
                DJNZ R6,CONTS_10
                MOV R6,#0FFH
                DJNZ R7,CONTS_10
                JMP END_PIC
CONTS_10:       JNB TMP_DATA,CONTD_09
                MOV R0,#DATA_RECV
                MOV R5,#RECV_SIZE
                CLR TMP_DATA
                MOV A,@R0
                JMP SEL0_V3
CONTD_09:       MOV A,@R0
                DJNZ R5,SEL0_V3
                JC S_N0V3
        ; GO TO VL3_MODE IN NEXT TIME
                MOV S_BYTE,#11H
                JMP END_48_BYTES
S_N0V3:  ; GO TO VR3_MODE IN NEXT TIME
                MOV S_BYTE,#12H
                JMP END_48_BYTES

GO_VL31_PROC:   JMP VL31_PROC

GO_VR3:         JMP VR3_MODE
SEL0_V3:        JC GO_VR3

VL3_MODE:       MOV CODE_TMP,A
        ; ------ GET REF. LINE INDEX -----
                MOV A,A0
                MOV DPTR,#GET_R1
                MOVC A,@A+DPTR
                MOV R1,A ;GET REF. LINE INDEX
        ; ------ GET A00 - A07 -------
                MOV A,A0
                MOV DPTR,#GET_R2
                MOVC A,@A+DPTR
                MOV R2,A  ;GET A00-A07
                CJNE A,#00H,CONTS_VL3
                MOV REF_OBYTE,REF_NBYTE
                MOV REF_NBYTE,@R1
CONTS_VL3:      JB STATE,GO_VL31_PROC
; ======= STATE 0(1) PROCESS =======
                SETB STATE
        ; ------ GET A00 - A07 TABLE -----
                MOV DPTR,#A0_WAY_0
                RL A
                INC A
                MOVC A,@A+DPTR
                PUSH ACC
                MOV A,R2
                RL A
                MOVC A,@A+DPTR
                MOV DPH1,A
                POP ACC
                MOV DPL1,A
        ; ------ GET MASK ------

MASK_VL30:      MOV A,@R1
                MOV DECODING_BYTE,A
        ;GET DATA AT REF. LINE
                MOVC A,@A+DPTR
                MOV MASK_CODE,A
                MOV DPTR,#A0_VALUE_0
                MOVC A,@A+DPTR
                ADD A,A0
                MOV A0,A

                MOV A,MASK_CODE
                ORL A,@R1
                MOV @R1,A
```

```
                JNB MASK_CHK,GO0_VL3
                INC R1
                MOV REF_OBYTE,REF_NBYTE
                MOV REF_NBYTE,@R1
                CJNE R1,#4EH,
                      VL30_CHK_STATE
                MOV R1,#REF_LINE ; NEW LINE
                MOV DPS,#00H
                REPT 14
                MOV A,@R1
                MOVX @DPTR,A
                INC DPTR
                INC R1
                ENDM
                MOV R1,#REF_LINE
                MOV DPS,#01H
                MOV A0,#00H
                INC CODE_LINE_NUM
                MOV A,CODE_LINE_NUM
                CJNE A,#ROW,GO0_VL3
                JMP END_PIC
VL30_CHK_STATE:JB DECODING_BCHK,VL30_MOV
                MOV A,@R1
                MOV DPTR,#A0_TAB0
                JMP MASK_VL30

GO0_VL3:        JMP END_VL3
;RETURN TO NEW COMPRESS DATA

VL30_MOV:       MOV A,@R1
                CJNE A,#0FFH,LAST_VL30
                MOV A,A0
                ADD A,#08H
                MOV A0,A
                INC R1
                MOV REF_OBYTE,REF_NBYTE
                MOV REF_NBYTE,@R1
                CJNE R1,#4EH,VL30_MOV
                MOV R1,#REF_LINE  ; NEW LINE
                MOV DPS,#00H
                REPT 14
                MOV A,@R1
                MOVX @DPTR,A
                INC DPTR
                INC R1
                ENDM
                MOV R1,#REF_LINE
                MOV DPS,#01H
                MOV A0,#00H
                INC CODE_LINE_NUM
                MOV A,CODE_LINE_NUM
                CJNE A,#ROW,GO0_VL3
                JMP END_PIC

LAST_VL30:      MOV DPTR,#A0_V_END0
                MOVC A,@A+DPTR
                ADD A,A0
                MOV A0,A
                JMP END_VL3
; ======= STATE 1(0) PROCESS =======
VL31_PROC:      CLR STATE
          ; ------ GET A00 - A07 TABLE -----
                MOV DPTR,#A0_WAY_1
                RL A
                INC A
                MOVC A,@A+DPTR
                PUSH ACC
                MOV A,R2
                RL A
                MOVC A,@A+DPTR
                MOV DPH1,A
                POP ACC
                MOV DPL1,A
                ; ------ GET MASK ------
MASK_VL31:      MOV A,@R1
                MOV DECODING_BYTE,A
                MOVC A,@A+DPTR
                MOV MASK_CODE,A

                MOV DPTR,#A0_VALUE_1
                MOVC A,@A+DPTR
                ADD A,A0
                MOV A0,A

                MOV A,MASK_CODE
                ANL A,@R1
                MOV @R1,A

                JB MASK_CHK,END_VL3
                INC R1
                MOV REF_OBYTE,REF_NBYTE
                MOV REF_NBYTE,@R1
                CJNE R1,#4EH,
                      VL31_CHK_STATE
                MOV R1,#REF_LINE  ; NEW LINE
                MOV DPS,#00H
                REPT 14
                MOV A,@R1
                MOVX @DPTR,A
                INC DPTR
                INC R1
                ENDM
                MOV R1,#REF_LINE
                MOV DPS,#01H
                MOV A0,#00H
                INC CODE_LINE_NUM
                MOV A,CODE_LINE_NUM
                CJNE A,#ROW,END_VL3
                JMP END_PIC

VL31_CHK_STATE: JNB DECODING_BCHK,
                      VL31_MOV
                MOV A,@R1
                MOV DPTR,#A0_TAB1
                JMP MASK_VL31

END_VL3:        DEC A0
                DEC A0
                DEC A0
                MOV A,A0
                MOV DPTR,#GET_R1
                MOVC A,@A+DPTR
                MOV R1,A
                ;GET REF. LINE INDEX
                MOV A,A0
                MOV DPTR,#GET_R2
                MOVC A,@A+DPTR
                MOV R2,A
                MOV DPTR,#VL3_MASK
                RL A
                INC A
                MOVC A,@A+DPTR
                PUSH ACC
                MOV A,R2
                RL A
                MOVC A,@A+DPTR
                PUSH ACC
                RET

VL31_MOV:       MOV A,@R1
                CJNE A,#00H,LAST_VL31
                MOV A,A0
                ADD A,#08H
                MOV A0,A
                INC R1
                MOV REF_OBYTE,REF_NBYTE
```

```
                MOV REF_NBYTE,@R1
                CJNE R1,#4EH,VL31_MOV
                MOV R1,#REF_LINE  ; NEW LINE
                MOV DPS,#00H
                REPT 14
                MOV A,@R1
                MOVX @DPTR,A
                INC DPTR
                INC R1
                ENDM
                MOV R1,#REF_LINE
                MOV DPS,#01H
                MOV A0,#00H
                INC CODE_LINE_NUM
                MOV A,CODE_LINE_NUM
                CJNE A,#ROW,GO1_VL3
                JMP END_PIC

LAST_VL31:      MOV DPTR,#A0_V_END1
                MOVC A,@A+DPTR
                ADD A,A0
                MOV A0,A
GO1_VL3:        JMP END_VL3

VL3_MASK0:      MOV A,#10001111B
                ANL A,@R1
                MOV @R1,A
                MOV A,REF_NBYTE
                ANL A,#01110000B
                ORL A,@R1
                MOV @R1,A
                MOV A,CODE_TMP
                JMP DECOM_DATA

VL3_MASK1:      MOV A,#11000111B
                ANL A,@R1
                MOV @R1,A
                MOV A,REF_NBYTE
                ANL A,#00111000B
                ORL A,@R1
                MOV @R1,A
                MOV A,CODE_TMP
                JMP DECOM_DATA

VL3_MASK2:      MOV A,#11100011B
                ANL A,@R1
                MOV @R1,A
                MOV A,REF_NBYTE
                ANL A,#00011100B
                ORL A,@R1
                MOV @R1,A
                MOV A,CODE_TMP
                JMP DECOM_DATA

VL3_MASK3:      MOV A,#11110001B
                ANL A,@R1
                MOV @R1,A
                MOV A,REF_NBYTE
                ANL A,#00001110B
                ORL A,@R1
                MOV @R1,A
                MOV A,CODE_TMP
                JMP DECOM_DATA

VL3_MASK4:      MOV A,#11111000B
                ANL A,@R1
                MOV @R1,A
                JMP DECOM_DATA

VL3_MASK5:      MOV A,#11111100B
                ANL A,@R1
                MOV @R1,A
                MOV A,REF_OBYTE
                ANL A,#00000011B
                ORL A,@R1
                MOV @R1,A
                INC R1
                MOV A,#01111111B
                ANL A,@R1
                MOV @R1,A
                MOV A,REF_NBYTE
                ANL A,#10000000B
                ORL A,@R1
                MOV @R1,A
                MOV REF_NBYTE,REF_OBYTE
                MOV A,CODE_TMP
                JMP DECOM_DATA

VL3_MASK6:      MOV A,#11111110B
                ANL A,@R1
                MOV @R1,A
                MOV A,REF_OBYTE
                ANL A,#00000001B
                ORL A,@R1
                MOV @R1,A
                INC R1
                MOV A,#00111111B
                ANL A,@R1
                MOV @R1,A
                MOV A,REF_NBYTE
                ANL A,#11000000B
                ORL A,@R1
                MOV @R1,A
                MOV REF_NBYTE,REF_OBYTE
                MOV A,CODE_TMP
                JMP DECOM_DATA

VL3_MASK7:      MOV A,#00011111B
                ANL A,@R1
                MOV @R1,A
                MOV A,REF_NBYTE
                ANL A,#11100000B
                ORL A,@R1
                MOV @R1,A
                MOV A,CODE_TMP
                JMP DECOM_DATA
                ;=================
GO_VR31_PROC:   JMP VR31_PROC
VR3_MODE:       MOV CODE_TMP,A
                ; ------ GET REF. LINE INDEX -----
                MOV A,A0
                MOV DPTR,#GET_R1
                MOVC A,@A+DPTR
                MOV R1,A ;GET REF. LINE INDEX
                ; ------ GET A00 - A07 -------
                MOV A,A0
                MOV DPTR,#GET_R2
                MOVC A,@A+DPTR
                MOV R2,A  ;GET A00-A07
                CJNE A,#00H,CONTS_VR3
                MOV REF_OBYTE,REF_NBYTE
                MOV REF_NBYTE,@R1
CONTS_VR3:      JB STATE,GO_VR31_PROC
; ======= STATE 0(1) PROCESS =======
                SETB STATE
                ; ------ GET A00 - A07 TABLE -----
                MOV DPTR,#A0_WAY_0
                RL A
                INC A
                MOVC A,@A+DPTR
                PUSH ACC
                MOV A,R2
                RL A
                MOVC A,@A+DPTR
                MOV DPH1,A
                POP ACC
```

```
                MOV DPL1,A
        ; ------ GET MASK ------
MASK_VR30:      MOV A,@R1
                MOV DECODING_BYTE,A
        ;GET DATA AT REF. LINE
                MOVC A,@A+DPTR
                MOV MASK_CODE,A

                MOV DPTR,#A0_VALUE_0
                MOVC A,@A+DPTR
                ADD A,A0
                MOV A0,A

                MOV A,MASK_CODE
                ORL A,@R1
                MOV @R1,A

                JNB MASK_CHK,
                    END_MASK_VR30
                INC R1
                MOV REF_OBYTE,REF_NBYTE
                MOV REF_NBYTE,@R1
                CJNE R1,#4EH,
                      VR30_CHK_STATE
                MOV R1,#REF_LINE  ; NEW LINE
                MOV DPS,#00H
                REPT 14
                MOV A,@R1
                MOVX @DPTR,A
                INC DPTR
                INC R1
                ENDM
                MOV R1,#REF_LINE
                MOV DPS,#01H
                MOV A0,#00H
                INC CODE_LINE_NUM
                MOV A,CODE_LINE_NUM
                CJNE A,#ROW,GO_END_VR30
                JMP END_PIC

VR30_CHK_STATE: JB DECODING_BCHK,
                    VR30_MOV
                MOV A,@R1
                MOV DPTR,#A0_TAB0
                JMP MASK_VR30

END_MASK_VR30: MOV A,MASK_CODE
                RR A
                ORL A,MASK_CODE
                MOV MASK_CODE,A
                ORL A,@R1
                MOV @R1,A
                JB MASK_CHK,L1_MASK_VR30
                MOV A,MASK_CODE
                RR A
                ORL A,MASK_CODE
                MOV MASK_CODE,A
                ORL A,@R1
                MOV @R1,A
                JB MASK_CHK,L2_MASK_VR30
                MOV A,MASK_CODE
                RR A
                ORL A,MASK_CODE
                ORL A,@R1
                MOV @R1,A
                INC A0
                INC A0
                INC A0
                JMP END_VR3

GO_END_VR30:    JMP END_VR3

L1_MASK_VR30:   INC R1
                MOV REF_OBYTE,REF_NBYTE
                MOV REF_NBYTE,@R1
                MOV A,@R1
                ORL A,#0C0H
                MOV @R1,A
                INC A0
                INC A0
                INC A0
                JMP END_VR3
L2_MASK_VR30:   INC R1
                MOV REF_OBYTE,REF_NBYTE
                MOV REF_NBYTE,@R1
                MOV A,@R1
                ORL A,#80H
                MOV @R1,A
                INC A0
                INC A0
                INC A0
GO1_END_VR30:   JMP END_VR3

VR30_MOV:       MOV A,@R1
                CJNE A,#0FFH,LAST_VR30
                MOV A,A0
                ADD A,#08H
                MOV A0,A
                INC R1
                MOV REF_OBYTE,REF_NBYTE
                MOV REF_NBYTE,@R1
                CJNE R1,#4EH,VR30_MOV
                MOV R1,#REF_LINE  ; NEW LINE
                MOV DPS,#00H
                REPT 14
                MOV A,@R1
                MOVX @DPTR,A
                INC DPTR
                INC R1
                ENDM
                MOV R1,#REF_LINE
                MOV DPS,#01H
                MOV A0,#00H
                INC CODE_LINE_NUM
                MOV A,CODE_LINE_NUM
                CJNE A,#ROW,GO1_END_VR30
                JMP END_PIC

LAST_VR30:      CJNE A,#0FCH,CHK_FE_VR30
                MOV DPTR,#A0_V_END0
                MOVC A,@A+DPTR
                INC A
                INC A
                INC A
                ADD A,A0
                MOV A0,A
                MOV A,@R1
                ORL A,#03H
                MOV @R1,A
                INC R1
                MOV REF_OBYTE,REF_NBYTE
                MOV REF_NBYTE,@R1
                MOV A,@R1
                ORL A,#80H
                MOV @R1,A
                JMP END_VR3

CHK_FE_VR30:    CJNE A,#0FEH,CHK_FF_VR30
                MOV DPTR,#A0_V_END0
                MOVC A,@A+DPTR
                INC A
                INC A
                INC A
                ADD A,A0
                MOV A0,A
                MOV A,@R1
```

```
                ORL A,#01H
                MOV @R1,A
                INC R1
                MOV REF_OBYTE,REF_NBYTE
                MOV REF_NBYTE,@R1
                MOV A,@R1
                ORL A,#0C0H
                MOV @R1,A
                JMP END_VR3

CHK_FF_VR30:    MOV DPTR,#A0_V_END0
                MOVC A,@A+DPTR
                CJNE A,#00H,N_MASK_VR30
                INC A0
                INC A0
                INC A0
                MOV A,@R1
                ORL A,#0E0H
                MOV @R1,A
                JMP END_VR3

N_MASK_VR30:    INC A
                INC A
                INC A
                ADD A,A0
                MOV A0,A
                MOV A,@R1
                MOV DPTR,#VR30_MASK_TAB
                MOVC A,@A+DPTR
                ORL A,@R1
                MOV @R1,A
                JMP END_VR3

; ======= STATE 1(0) PROCESS =======
VR31_PROC:      CLR STATE
        ; ------ GET A00 - A07 TABLE -----
                MOV DPTR,#A0_WAY_1
                RL A
                INC A
                MOVC A,@A+DPTR
                PUSH ACC
                MOV A,R2
                RL A
                MOVC A,@A+DPTR
                MOV DPH1,A
                POP ACC
                MOV DPL1,A

        ; ------ GET MASK ------
MASK_VR31:      MOV A,@R1
                MOV DECODING_BYTE,A
                MOVC A,@A+DPTR
                MOV MASK_CODE,A
                MOV DPTR,#A0_VALUE_1
                MOVC A,@A+DPTR
                ADD A,A0
                MOV A0,A

                MOV A,MASK_CODE
                ANL A,@R1
                MOV @R1,A

                JB MASK_CHK,END_MASK_VR31
                INC R1
                MOV REF_OBYTE,REF_NBYTE
                MOV REF_NBYTE,@R1
                CJNE R1,#4EH,
                        VR31_CHK_STATE
                MOV R1,#REF_LINE  ; NEW LINE
                MOV DPS,#00H
                REPT 14
                MOV A,@R1
                MOVX @DPTR,A
                INC DPTR
                INC R1
                ENDM
                MOV R1,#REF_LINE
                MOV DPS,#01H
                MOV A0,#00H
                INC CODE_LINE_NUM
                MOV A,CODE_LINE_NUM
                CJNE A,#ROW,END_VR3
                JMP END_PIC

VR31_CHK_STATE: JNB DECODING_BCHK,
                        VR31_MOV
                MOV A,@R1
                MOV DPTR,#A0_TAB1
                JMP MASK_VR31


END_MASK_VR31:  MOV A,MASK_CODE
                RR A
                ANL A,MASK_CODE
                MOV MASK_CODE,A
                ANL A,@R1
                MOV @R1,A
                JNB MASK_CHK,L1_MASK_VR31
                MOV A,MASK_CODE
                RR A
                ANL A,MASK_CODE
                MOV MASK_CODE,A
                ANL A,@R1
                MOV @R1,A
                JNB MASK_CHK,L1_MASK_VR31
                MOV A,MASK_CODE
                RR A
                ANL A,MASK_CODE
                ANL A,@R1
                MOV @R1,A
                INC A0
                INC A0
                INC A0
                JMP END_VR3

END_VR3:        MOV A,CODE_TMP
                JMP DECOM_DATA

L1_MASK_VR31:   INC R1
                MOV REF_OBYTE,REF_NBYTE
                MOV REF_NBYTE,@R1
                MOV A,@R1
                ANL A,#3FH
                MOV @R1,A
                INC A0
                INC A0
                INC A0
                JMP END_VR3
L2_MASK_VR31:   INC R1
                MOV REF_OBYTE,REF_NBYTE
                MOV REF_NBYTE,@R1
                MOV A,@R1
                ANL A,#7FH
                MOV @R1,A
                INC A0
                INC A0
                INC A0
GO_END_VR31:    JMP END_VR3

VR31_MOV:       MOV A,@R1
                CJNE A,#00H,LAST_VR31
                MOV A,A0
                ADD A,#08H
                MOV A0,A
                INC R1
                MOV REF_OBYTE,REF_NBYTE
```

```
                MOV REF_NBYTE,@R1
                CJNE R1,#4EH,VR31_MOV
                MOV R1,#REF_LINE  ; NEW LINE
                MOV DPS,#00H
                REPT 14
                MOV A,@R1
                MOVX @DPTR,A
                INC DPTR
                INC R1
                ENDM
                MOV R1,#REF_LINE
                MOV DPS,#01H
                MOV A0,#00H
                INC CODE_LINE_NUM
                MOV A,CODE_LINE_NUM
                CJNE A,#ROW,GO_END_VR31
                JMP END_PIC

LAST_VR31:      CJNE A,#03H,CHK_01_VR31
                MOV DPTR,#A0_V_END1
                MOVC A,@A+DPTR
                INC A
                INC A
                INC A
                ADD A,A0
                MOV A0,A
                MOV A,@R1
                ANL A,#0FCH
                MOV @R1,A
                INC R1
                MOV REF_OBYTE,REF_NBYTE
                MOV REF_NBYTE,@R1
                MOV A,@R1
                ANL A,#7FH
                MOV @R1,A
                JMP END_VR3

CHK_01_VR31:    CJNE A,#01H,CHK_00_VR31
                MOV DPTR,#A0_V_END1
                MOVC A,@A+DPTR
                INC A
                INC A
                INC A
                ADD A,A0
                MOV A0,A
                MOV A,@R1
                ANL A,#0FEH
                MOV @R1,A
                INC R1
                MOV A,@R1
                ANL A,#3FH
                MOV @R1,A
                JMP END_VR3

CHK_00_VR31:    MOV DPTR,#A0_V_END1
                MOVC A,@A+DPTR
                CJNE A,#00H,N_MASK_VR31
                INC A0
                INC A0
                INC A0
                MOV A,@R1
                ANL A,#0E0H
                MOV @R1,A
                JMP END_VR3

N_MASK_VR31:    INC A
                INC A
                INC A
                ADD A,A0
                MOV A0,A
                MOV A,@R1
                MOV DPTR,#VR31_MASK_TAB
                MOVC A,@A+DPTR
                ANL A,@R1
                MOV @R1,A
                JMP END_VR3

;=============== HOR MODE ================
GO_HOR_BBW:     JMP HOR_BBW
HOR_MODE:       JB STATE,GO_HOR_BBW

;---------- HOR MODE IN WHITE PIXEL ----------
;---------- WHITE ----------
HOR_WWB:        MOV CODE_TMP,A
                MOV BIT_C_TMP,BIT_C
                MOV R0_TMP,R0
                MOV R5_TMP,R5
                ; ------ GET REF. LINE INDEX -----
                MOV A,A0
                MOV DPTR,#GET_R1
                MOVC A,@A+DPTR
                MOV R1,A ;GET REF. LINE INDEX
                MOV A,CODE_TMP

R_WWB1:         RLC A     ;ROTATE BIT
                XCH A,R4
                RLC A
                XCH A,R4
                DJNZ BIT_C,R_WWB2
                MOV BIT_C,#08H
                INC R0
                JNB TMP_DATA,CONTD_10
                MOV R0,#DATA_RECV
                MOV R5,#RECV_SIZE
                CLR TMP_DATA
                MOV A,@R0
                JMP R_WWB2
CONTD_10:       MOV A,@R0
                DJNZ R5,R_WWB2
                MOV S_BYTE,#13H
                SETB TMP_DATA  ; GO R0TATE2
                JMP END_48_BYTES
R_WWB2:         RLC A     ;ROTATE BIT
                XCH A,R4
                RLC A
                XCH A,R4
                DJNZ BIT_C,R_WWB3
                MOV BIT_C,#08H
                INC R0
                JNB TMP_DATA,CONTD_11
                MOV R0,#DATA_RECV
                MOV R5,#RECV_SIZE
                CLR TMP_DATA
                MOV A,@R0
                JMP R_WWB3
CONTD_11:       MOV A,@R0
                DJNZ R5,R_WWB3
                MOV S_BYTE,#14H
                SETB TMP_DATA
                ; GO R0TATE3
                JMP END_48_BYTES
R_WWB3:         RLC A     ;ROTATE BIT
                XCH A,R4
                RLC A
                XCH A,R4
                DJNZ BIT_C,R_WWB4
                MOV BIT_C,#08H
                INC R0
                JNB TMP_DATA,CONTD_12
                MOV R0,#DATA_RECV
                MOV R5,#RECV_SIZE
                CLR TMP_DATA
                MOV A,@R0
                JMP R_WWB4
CONTD_12:       MOV A,@R0
```

```
                DJNZ R5,R_WWB4
        ; GO R0TATE4
                SETB TMP_DATA
                MOV S_BYTE,#15H
                JMP END_48_BYTES
R_WWB4:         RLC A     ;ROTATE BIT
                XCH A,R4
                RLC A
                XCH A,R4
                DJNZ BIT_C,R_WWB5
                MOV BIT_C,#08H
                INC R0
                JNB TMP_DATA,CONTD_13
                MOV R0,#DATA_RECV
                MOV R5,#RECV_SIZE
                CLR TMP_DATA
                MOV A,@R0
                JMP R_WWB5
CONTD_13:       MOV A,@R0
                DJNZ R5,R_WWB5
        ; GO R0TATE5
                SETB TMP_DATA
                MOV S_BYTE,#16H
                JMP END_48_BYTES
R_WWB5:         RLC A     ;ROTATE BIT
                XCH A,R4
                RLC A
                XCH A,R4
                DJNZ BIT_C,R_WWB6
                MOV BIT_C,#08H
                INC R0
                MOV A,@R0
                DJNZ R5,R_WWB6
        ; GO R0TATE6
                SETB TMP_DATA
                MOV S_BYTE,#17H
                JMP END_48_BYTES
R_WWB6:         RLC A     ;ROTATE BIT
                XCH A,R4
                RLC A
                XCH A,R4
                DJNZ BIT_C,R_WWB7
                MOV BIT_C,#08H
                INC R0
                MOV A,@R0
                DJNZ R5,R_WWB7
        ; GO R0TATE7
                SETB TMP_DATA
                MOV S_BYTE,#18H
                JMP END_48_BYTES
R_WWB7:         RLC A     ;ROTATE BIT
                XCH A,R4
                RLC A
                XCH A,R4
                DJNZ BIT_C,R_WWB8
                MOV BIT_C,#08H
                INC R0
                MOV A,@R0
                DJNZ R5,R_WWB8
        ; GO R0TATE8
                SETB TMP_DATA
                MOV S_BYTE,#19H
                JMP END_48_BYTES
R_WWB8:         RLC A     ;ROTATE BIT
                XCH A,R4
                RLC A
                XCH A,R4
                DJNZ BIT_C,CODE_WWB
                MOV BIT_C,#08H
                INC R0
                MOV A,@R0
                DJNZ R5,CODE_WWB
        ; GO CODE_WWB
                SETB TMP_DATA
                MOV S_BYTE,#1AH
                JMP END_48_BYTES

CODE_WWB:       MOV DPTR,#W_INDEX
        ;GET INDEX OF DATA
                MOV A,R4
                MOVC A,@A+DPTR
                RL A
                MOV TMP_R3,A
        ;------ GET NUMBER OF PIXEL -------
                MOV DPTR,#W_DATA
                MOVC A,@A+DPTR
                MOV NUM_PIXEL,A
        ;------ GET NUMBER OF BIT ---------
                MOV A,TMP_R3
                INC A
                MOVC A,@A+DPTR
                MOV R4,A
        ;---------- SET POSITION ----------
                MOV BIT_C,BIT_C_TMP
                MOV A,CODE_TMP
                MOV R0,R0_TMP
                MOV R5,R5_TMP

                CJNE R0,#7FH,WWB_BIT
WWB_TMP_BIT:    RL A
                DJNZ BIT_C,WWB_TMP_BYTE
                MOV BIT_C,#08H
                MOV R5,#RECV_SIZE
                MOV R0,#DATA_RECV
                CLR TMP_DATA
                MOV A,@R0
                DJNZ R4,WWB_BIT
                JMP SET_WWB
WWB_TMP_BYTE: DJNZ R4,WWB_TMP_BIT
                JMP SET_WWB
WWB_BIT:        RL A
                DJNZ BIT_C,WWB_BYTE
                MOV BIT_C,#08H
                INC R0
                DEC R5
                MOV A,@R0
WWB_BYTE:       DJNZ R4,WWB_BIT
                ;---------- SET PIXEL -------------
SET_WWB:        MOV CODE_TMP,A
                MOV R4,NUM_PIXEL
                ;---------- HIGH BYTE -------------
                MOV A,A0
                MOV DPTR,#SETWWB_TAB
                RL A
                INC A
                MOVC A,@A+DPTR
                PUSH ACC
                ;---------- LOW BYTE -------------
                MOV A,A0
                RL A
                MOVC A,@A+DPTR
                PUSH ACC
                MOV A,A0
                ADD A,R4
                MOV A0,A
                RET
                ;---------- TABLE SET ------------
WWB0:           CJNE R4,#08H,CHK_WWB0
                MOV @R1,#11111111B
                INC R1
                MOV REF_OBYTE,REF_NBYTE
                MOV REF_NBYTE,@R1
                JMP HOR_BWB
CHK_WWB0:       JC LESS_WWB0
                MOV @R1,#11111111B
                INC R1
```

```
                MOV REF_OBYTE,REF_NBYTE
                MOV REF_NBYTE,@R1
                MOV A,R4
                CLR C
                SUBB A,#08H
                MOV R4,A
                JMP WWB0
LESS_WWB0:      MOV A,R4
                MOV DPTR,#LESS8_WWB
                RL A
                INC A
                MOVC A,@A+DPTR
                PUSH ACC
                MOV A,R4
                RL A
                MOVC A,@A+DPTR
                PUSH ACC
                RET
        ;============================
WWB1:           CJNE R4,#07H,CHK_WWB1
                MOV A,@R1
                ORL A,#01111111B
                MOV @R1,A
                INC R1
                MOV REF_OBYTE,REF_NBYTE
                MOV REF_NBYTE,@R1
                JMP HOR_BWB
CHK_WWB1:       JC LESS_WWB1
                MOV A,@R1
                ORL A,#01111111B
                MOV @R1,A
                INC R1
                MOV REF_OBYTE,REF_NBYTE
                MOV REF_NBYTE,@R1
                MOV A,R4
                CLR C
                SUBB A,#07H
                MOV R4,A
                JMP WWB0
LESS_WWB1:      MOV A,R4
                MOV DPTR,#LESS7_WWB
                RL A
                INC A
                MOVC A,@A+DPTR
                PUSH ACC
                MOV A,R4
                RL A
                MOVC A,@A+DPTR
                PUSH ACC
                RET
L7_WWB0:        JMP HOR_BWB
L7_WWB1:        MOV A,@R1
                ORL A,#01000000B
                MOV @R1,A
                JMP HOR_BWB
L7_WWB2:        MOV A,@R1
                ORL A,#01100000B
                MOV @R1,A
                JMP HOR_BWB
L7_WWB3:        MOV A,@R1
                ORL A,#01110000B
                MOV @R1,A
                JMP HOR_BWB
L7_WWB4:        MOV A,@R1
                ORL A,#01111000B
                MOV @R1,A
                JMP HOR_BWB
L7_WWB5:        MOV A,@R1
                ORL A,#01111100B
                MOV @R1,A
                JMP HOR_BWB
L7_WWB6:        MOV A,@R1
                ORL A,#01111110B
                MOV @R1,A
                JMP HOR_BWB
        ;============================
WWB2:           CJNE R4,#06H,CHK_WWB2
                MOV A,@R1
                ORL A,#00111111B
                MOV @R1,A
                INC R1
                MOV REF_OBYTE,REF_NBYTE
                MOV REF_NBYTE,@R1
                JMP HOR_BWB
CHK_WWB2:       JC LESS_WWB2
                MOV A,@R1
                ORL A,#00111111B
                MOV @R1,A
                INC R1
                MOV REF_OBYTE,REF_NBYTE
                MOV REF_NBYTE,@R1
                MOV A,R4
                CLR C
                SUBB A,#06H
                MOV R4,A
                JMP WWB0
LESS_WWB2:      MOV A,R4
                MOV DPTR,#LESS6_WWB
                RL A
                INC A
                MOVC A,@A+DPTR
                PUSH ACC
                MOV A,R4
                RL A
                MOVC A,@A+DPTR
                PUSH ACC
                RET
L6_WWB0:        JMP HOR_BWB
L6_WWB1:        MOV A,@R1
                ORL A,#00100000B
                MOV @R1,A
                JMP HOR_BWB
L6_WWB2:        MOV A,@R1
                ORL A,#00110000B
                MOV @R1,A
                JMP HOR_BWB
L6_WWB3:        MOV A,@R1
                ORL A,#00111000B
                MOV @R1,A
                JMP HOR_BWB
L6_WWB4:        MOV A,@R1
                ORL A,#00111100B
                MOV @R1,A
                JMP HOR_BWB
L6_WWB5:        MOV A,@R1
                ORL A,#00111110B
                MOV @R1,A
                JMP HOR_BWB
        ;============================
WWB3:           CJNE R4,#05H,CHK_WWB3
                MOV A,@R1
                ORL A,#00011111B
                MOV @R1,A
                INC R1
                MOV REF_OBYTE,REF_NBYTE
                MOV REF_NBYTE,@R1
                JMP HOR_BWB
CHK_WWB3:       JC LESS_WWB3
                MOV A,@R1
                ORL A,#00011111B
                MOV @R1,A
                INC R1
                MOV REF_OBYTE,REF_NBYTE
```

```
                MOV REF_NBYTE,@R1
                MOV A,R4
                CLR C
                SUBB A,#05H
                MOV R4,A
                JMP WWB0
LESS_WWB3:      MOV A,R4
                MOV DPTR,#LESS5_WWB
                RL A
                INC A
                MOVC A,@A+DPTR
                PUSH ACC
                MOV A,R4
                RL A
                MOVC A,@A+DPTR
                PUSH ACC
                RET
L5_WWB0:        JMP HOR_BWB
L5_WWB1:        MOV A,@R1
                ORL A,#00010000B
                MOV @R1,A
                JMP HOR_BWB
L5_WWB2:        MOV A,@R1
                ORL A,#00011000B
                MOV @R1,A
                JMP HOR_BWB
L5_WWB3:        MOV A,@R1
                ORL A,#00011100B
                MOV @R1,A
                JMP HOR_BWB
L5_WWB4:        MOV A,@R1
                ORL A,#00011110B
                MOV @R1,A
                JMP HOR_BWB
        ;============================
WWB4:           CJNE R4,#04H,CHK_WWB4
                MOV A,@R1
                ORL A,#00001111B
                MOV @R1,A
                INC R1
                MOV REF_OBYTE,REF_NBYTE
                MOV REF_NBYTE,@R1
                JMP HOR_BWB
CHK_WWB4:       JC LESS_WWB4
                MOV A,@R1
                ORL A,#00001111B
                MOV @R1,A
                INC R1
                MOV REF_OBYTE,REF_NBYTE
                MOV REF_NBYTE,@R1
                MOV A,R4
                CLR C
                SUBB A,#04H
                MOV R4,A
                JMP WWB0
LESS_WWB4:      DJNZ R4,N2_WWB4
                MOV A,@R1
                ORL A,#00001000B
                MOV @R1,A
                JMP HOR_BWB

N2_WWB4:        DJNZ R4,N3_WWB4
                MOV A,@R1
                ORL A,#00001100B
                MOV @R1,A
                JMP HOR_BWB
N3_WWB4:        MOV A,@R1
                ORL A,#00001110B
                MOV @R1,A
                JMP HOR_BWB
        ;============================
WWB5:           CJNE R4,#03H,CHK_WWB5
                MOV A,@R1
                ORL A,#00000111B
                MOV @R1,A
                INC R1
                MOV REF_OBYTE,REF_NBYTE
                MOV REF_NBYTE,@R1
                JMP HOR_BWB
CHK_WWB5:       JC LESS_WWB5
                MOV A,@R1
                ORL A,#00000111B
                MOV @R1,A
                INC R1
                MOV REF_OBYTE,REF_NBYTE
                MOV REF_NBYTE,@R1
                MOV A,R4
                CLR C
                SUBB A,#03H
                MOV R4,A
                JMP WWB0
LESS_WWB5:      DJNZ R4,N2_WWB5
                MOV A,@R1
                ORL A,#00000100B
                MOV @R1,A
                JMP HOR_BWB

N2_WWB5:        MOV A,@R1
                ORL A,#00000110B
                MOV @R1,A
                JMP HOR_BWB
        ;============================
WWB6:           CJNE R4,#02H,CHK_WWB6
                MOV A,@R1
                ORL A,#00000011B
                MOV @R1,A
                INC R1
                MOV REF_OBYTE,REF_NBYTE
                MOV REF_NBYTE,@R1
                JMP HOR_BWB
CHK_WWB6:       JC LESS_WWB6
                MOV A,@R1
                ORL A,#00000011B
                MOV @R1,A
                INC R1
                MOV REF_OBYTE,REF_NBYTE
                MOV REF_NBYTE,@R1
                MOV A,R4
                CLR C
                SUBB A,#02H
                MOV R4,A
                JMP WWB0
LESS_WWB6:      MOV A,@R1
                ORL A,#00000010B
                MOV @R1,A
                JMP HOR_BWB
        ;============================
WWB7:           CJNE R4,#01H,CHK_WWB7
                MOV A,@R1
                ORL A,#00000001B
                MOV @R1,A
                INC R1
                MOV REF_OBYTE,REF_NBYTE
                MOV REF_NBYTE,@R1
                JMP HOR_BWB
CHK_WWB7:       MOV A,@R1
                ORL A,#00000001B
                MOV @R1,A
                INC R1
                MOV REF_OBYTE,REF_NBYTE
                MOV REF_NBYTE,@R1
                MOV A,R4
                CLR C
                SUBB A,#01H
                MOV R4,A
                JMP WWB0
```

```
;-------------- LESS THAN 8 BIT ---------------------
L8_WWB0:        JMP HOR_BWB

L8_WWB1:        MOV A,@R1
                ORL A,#10000000B
                MOV @R1,A
                JMP HOR_BWB

L8_WWB2:        MOV A,@R1
                ORL A,#11000000B
                MOV @R1,A
                JMP HOR_BWB

L8_WWB3:        MOV A,@R1
                ORL A,#11100000B
                MOV @R1,A
                JMP HOR_BWB

L8_WWB4:        MOV A,@R1
                ORL A,#11110000B
                MOV @R1,A
                JMP HOR_BWB

L8_WWB5:        MOV A,@R1
                ORL A,#11111000B
                MOV @R1,A
                JMP HOR_BWB

L8_WWB6:        MOV A,@R1
                ORL A,#11111100B
                MOV @R1,A
                JMP HOR_BWB

L8_WWB7:        MOV A,@R1
                ORL A,#11111110B
                MOV @R1,A
                JMP HOR_BWB

;---------------------- BLACK -------------------------
HOR_BWB:        MOV BIT_C_TMP,BIT_C
                MOV R0_TMP,R0
                MOV R5_TMP,R5
                MOV R4,#00H
        ; ------ GET REF. LINE INDEX -----
                MOV A,A0
                MOV DPTR,#GET_R1
                MOVC A,@A+DPTR
                MOV R1,A ;GET REF. LINE INDEX
                MOV A,CODE_TMP

R_BWB41:        RLC A     ;ROTATE BIT
                XCH A,R4
                RLC A
                XCH A,R4
                DJNZ BIT_C,R_BWB42
                MOV BIT_C,#08H
                INC R0
                MOV A,@R0
                DJNZ R5,R_BWB42
        ; GO R0TATE42
                MOV S_BYTE,#1BH
                JMP END_48_BYTES
R_BWB42:        RLC A     ;ROTATE BIT
                XCH A,R4
                RLC A
                XCH A,R4
                DJNZ BIT_C,R_BWB43
                MOV BIT_C,#08H
                INC R0
                MOV A,@R0
                DJNZ R5,R_BWB43
        ; GO R0TATE43
                MOV S_BYTE,#1CH
                JMP END_48_BYTES
R_BWB43:        RLC A     ;ROTATE BIT
                XCH A,R4
                RLC A
                XCH A,R4
                DJNZ BIT_C,R_BWB44
                MOV BIT_C,#08H
                INC R0
                MOV A,@R0
                DJNZ R5,R_BWB44
        ; GO R0TATE44
                MOV S_BYTE,#1DH
                JMP END_48_BYTES
R_BWB44:        RLC A     ;ROTATE BIT
                XCH A,R4
                RLC A
                XCH A,R4
                DJNZ BIT_C,CODE4_BWB
                MOV BIT_C,#08H
                INC R0
                MOV A,@R0
                DJNZ R5,CODE4_BWB
        ; GO COD4_BWB
                MOV S_BYTE,#1EH
                JMP END_48_BYTES
        ;---------- 4 BIT INDEX TABLE  -----------
CODE4_BWB:      MOV CODE_TMP2,A
                MOV R0_TMP2,R0
                MOV BIT_C_TMP2,BIT_C
                MOV R5_TMP2,R5

                MOV DPTR,#BWB_INDEX
                MOV A,R4
                RL A
                INC A
                MOVC A,@A+DPTR
                PUSH ACC
                MOV A,R4
                RL A
                MOVC A,@A+DPTR
                PUSH ACC
                RET
        ;-------------- 4 BIT LABEL ----------------
        ;------------ 4B 0000 ROUTINE --------------
BWB4_0000:      MOV A,CODE_TMP2
                MOV R4,#00H

R0_BWB81:       RLC A     ;ROTATE BIT
                XCH A,R4
                RLC A
                XCH A,R4
                DJNZ BIT_C,R0_BWB82
                MOV BIT_C,#08H
                INC R0
                MOV A,@R0
                DJNZ R5,R0_BWB82
        ; GO R0TATE82
                SETB TMP_DATA
                MOV S_BYTE,#1FH
                JMP END_48_BYTES
R0_BWB82:       RLC A     ;ROTATE BIT
                XCH A,R4
                RLC A
                XCH A,R4
                DJNZ BIT_C,R0_BWB83
                MOV BIT_C,#08H
                INC R0
                MOV A,@R0
                DJNZ R5,R0_BWB83
        ; GO R0TATE83
                SETB TMP_DATA
```

```
                MOV S_BYTE,#20H
                JMP END_48_BYTES
R0_BWB83:       RLC A     ;ROTATE BIT
                XCH A,R4
                RLC A
                XCH A,R4
                DJNZ BIT_C,R0_BWB84
                MOV BIT_C,#08H
                INC R0
                MOV A,@R0
                DJNZ R5,R0_BWB84
        ; GO R0TATE84
                SETB TMP_DATA
                MOV S_BYTE,#21H
                JMP END_48_BYTES
R0_BWB84:       RLC A     ;ROTATE BIT
                XCH A,R4
                RLC A
                XCH A,R4
                DJNZ BIT_C,R0_BWB85
                MOV BIT_C,#08H
                INC R0
                MOV A,@R0
                DJNZ R5,R0_BWB85
        ; GO R0TATE85
                SETB TMP_DATA
                MOV S_BYTE,#22H
                JMP END_48_BYTES
R0_BWB85:       RLC A     ;ROTATE BIT
                XCH A,R4
                RLC A
                XCH A,R4
                DJNZ BIT_C,R0_BWB86
                MOV BIT_C,#08H
                INC R0
                MOV A,@R0
                DJNZ R5,R0_BWB86
        ; GO R0TATE86
                SETB TMP_DATA
                MOV S_BYTE,#23H
                JMP END_48_BYTES
R0_BWB86:       RLC A     ;ROTATE BIT
                XCH A,R4
                RLC A
                XCH A,R4
                DJNZ BIT_C,R0_BWB87
                MOV BIT_C,#08H
                MOV A,@R0
                DJNZ R5,R0_BWB87
        ; GO R0TATE87
                SETB TMP_DATA
                MOV S_BYTE,#24H
                JMP END_48_BYTES
R0_BWB87:       RLC A     ;ROTATE BIT
                XCH A,R4
                RLC A
                XCH A,R4
                DJNZ BIT_C,R0_BWB88
                MOV BIT_C,#08H
                INC R0
                MOV A,@R0
                DJNZ R5,R0_BWB88
        ; GO R0TATE88
                SETB TMP_DATA
                MOV S_BYTE,#25H
                JMP END_48_BYTES
R0_BWB88:       RLC A     ;ROTATE BIT
                XCH A,R4
                RLC A
                XCH A,R4
                DJNZ BIT_C,CODEBWB_00
                MOV BIT_C,#08H
                INC R0
                MOV A,@R0
                DJNZ R5,CODEBWB_00
        ; GO CODE8_BWB
                SETB TMP_DATA
                MOV S_BYTE,#26H
                JMP END_48_BYTES
                ;---------- GET INDEX ----------
CODEBWB_00:     MOV DPTR,#B_INDEX8B0
                MOV A,R4
                MOVC A,@A+DPTR
                RL A
                MOV TMP_R3,A
                ;------ GET NUMBER OF PIXEL -------
                MOV DPTR,#B_DATA
                MOVC A,@A+DPTR
                MOV NUM_PIXEL,A
                ;------ GET NUMBER OF BIT ---------
                MOV A,TMP_R3
                INC A
                MOVC A,@A+DPTR
                MOV R4,A
                ;---------- SET POSITION ----------
                MOV BIT_C,BIT_C_TMP2
                MOV A,CODE_TMP2
                MOV R0,R0_TMP2
                MOV R5,R5_TMP2
                CJNE R0,#7FH,BWB00_BIT
BWB00_TMP_BIT:  RL A
                DJNZ BIT_C,BWB00_TMP_BYTE
                MOV BIT_C,#08H
                MOV R5,#RECV_SIZE
                MOV R0,#DATA_RECV
                CLR TMP_DATA
                MOV A,@R0
                DJNZ R4,BWB00_BIT
                JMP SET_BWB
BWB00_TMP_BYTE: DJNZ R4,BWB00_TMP_BIT
                MOV R4,NUM_PIXEL
                JMP SET_BWB

BWB00_BIT:      RL A
                DJNZ BIT_C,BWB00_BYTE
                MOV BIT_C,#08H
                INC R0
                DEC R5
                MOV A,@R0
BWB00_BYTE:     DJNZ R4,BWB00_BIT
                MOV R4,NUM_PIXEL
                JMP SET_BWB

                ;------------ 4B 0001 ROUTINE --------------
BWB4_0001:      MOV A,CODE_TMP2
                MOV R4,#00H
R1_BWB31:       RLC A     ;ROTATE BIT
                XCH A,R4
                RLC A
                XCH A,R4
                DJNZ BIT_C,R1_BWB32
                MOV BIT_C,#08H
                INC R0
                MOV A,@R0
                DJNZ R5,R1_BWB32
        ; GO R0TATE32
                SETB TMP_DATA
                MOV S_BYTE,#27H
                JMP END_48_BYTES
R1_BWB32:       RLC A     ;ROTATE BIT
                XCH A,R4
                RLC A
                XCH A,R4
                DJNZ BIT_C,CODEBWB_01
                MOV BIT_C,#08H
```

```
                INC R0
                MOV A,@R0
                DJNZ R5,CODEBWB_01
        ; GO CODE8_BWB
                SETB TMP_DATA
                MOV S_BYTE,#28H
                JMP END_48_BYTES

        ;----------- GET INDEX ------------
CODEBWB_01:     MOV DPTR,#B_INDEX8B1
                MOV A,R4
                MOVC A,@A+DPTR
                RL A
                MOV TMP_R3,A
        ;------ GET NUMBER OF PIXEL -------
                MOV DPTR,#B_DATA
                MOVC A,@A+DPTR
                MOV NUM_PIXEL,A
        ;------ GET NUMBER OF BIT ---------
                MOV A,TMP_R3
                INC A
                MOVC A,@A+DPTR
                MOV R4,A
        ;---------- SET POSITION ----------
                MOV BIT_C,BIT_C_TMP2
                MOV A,CODE_TMP2
                MOV R0,R0_TMP2
                MOV R5,R5_TMP2
                CJNE R0,#7FH,BWB01_BIT
BWB01_TMP_BIT:  RL A
                DJNZ BIT_C,BWB01_TMP_BYTE
                MOV BIT_C,#08H
                MOV R5,#RECV_SIZE
                MOV R0,#DATA_RECV
                CLR TMP_DATA
                MOV A,@R0
                DJNZ R4,BWB01_BIT
                JMP SET_BWB
BWB01_TMP_BYTE: DJNZ R4,BWB01_TMP_BIT
                MOV R4,NUM_PIXEL
                JMP SET_BWB

BWB01_BIT:      RL A
                DJNZ BIT_C,BWB01_BYTE
                MOV BIT_C,#08H
                INC R0
                DEC R5
                MOV A,@R0
BWB01_BYTE:     DJNZ R4,BWB01_BIT
                MOV R4,NUM_PIXEL
                JMP SET_BWB

        ;------------ 4B 0010 ROUTINE --------------
        ;-------------- SET POSITION ---------------
BWB4_0010:      MOV A,CODE_TMP2
                MOV R4,#06H
                JMP SET_BWB
        ;------------ 4B 0011 ROUTINE --------------

BWB4_0011:      MOV A,CODE_TMP2

                MOV R4,#05H
                JMP SET_BWB
        ;------------ 4B 0100 ROUTINE --------------

        ;---------- SET POSITION ----------
BWB4_0100:      MOV BIT_C,BIT_C_TMP
                MOV A,CODE_TMP
                MOV R0,R0_TMP
                MOV R5,R5_TMP

                MOV R4,#03H
                CJNE R0,#7FH,BWB010_BIT
BWB010_TMP_BIT:RL A
                DJNZ BIT_C,BWB010_TMP_BYTE
                MOV BIT_C,#08H
                MOV R5,#RECV_SIZE
                MOV R0,#DATA_RECV
                CLR TMP_DATA
                MOV A,@R0
                DJNZ R4,BWB010_BIT
                MOV R4,#01H
                JMP SET_BWB
BWB010_TMP_BYTE: DJNZ R4,BWB010_TMP_BIT
                MOV R4,#01H
                JMP SET_BWB

BWB010_BIT:     RL A
                DJNZ BIT_C,BWB010_BYTE
                MOV BIT_C,#08H
                INC R0
                DEC R5
                MOV A,@R0
BWB010_BYTE:    DJNZ R4,BWB010_BIT
                MOV R4,#01H
                JMP SET_BWB

        ;------------ 4B 0110 ROUTINE --------------

        ;---------- SET POSITION ----------
BWB4_0110:      MOV BIT_C,BIT_C_TMP
                MOV A,CODE_TMP
                MOV R0,R0_TMP
                MOV R5,R5_TMP

                MOV R4,#03H
                CJNE R0,#7FH,BWB011_BIT
BWB011_TMP_BIT:RL A
                DJNZ BIT_C,BWB011_TMP_BYTE
                MOV BIT_C,#08H
                MOV R5,#RECV_SIZE
                MOV R0,#DATA_RECV
                CLR TMP_DATA
                MOV A,@R0
                DJNZ R4,BWB011_BIT
                MOV R4,#04H
                JMP SET_BWB
BWB011_TMP_BYTE: DJNZ R4,BWB011_TMP_BIT
                MOV R4,#04H
                JMP SET_BWB

BWB011_BIT:     RL A
                DJNZ BIT_C,BWB011_BYTE
                MOV BIT_C,#08H
                INC R0
                DEC R5
                MOV A,@R0
BWB011_BYTE:    DJNZ R4,BWB011_BIT
                MOV R4,#04H
                JMP SET_BWB
        ;------------ 4B 1000 ROUTINE --------------

        ;---------- SET POSITION ----------
BWB4_1000:      MOV BIT_C,BIT_C_TMP
                MOV A,CODE_TMP
                MOV R0,R0_TMP
                MOV R5,R5_TMP

                MOV R4,#02H
                CJNE R0,#7FH,BWB100_BIT
BWB100_TMP_BIT:RL A
                DJNZ BIT_C,BWB100_TMP_BYTE
                MOV BIT_C,#08H
                MOV R0,#DATA_RECV
                MOV R5,#RECV_SIZE
                CLR TMP_DATA
```

```
                MOV A,@R0
                DJNZ R4,BWB100_BIT
                JMP SET_BWB
BWB100_TMP_BYTE: DJNZ R4,BWB100_TMP_BIT
                MOV R4,#03H
                JMP SET_BWB

BWB100_BIT:     RL A
                DJNZ BIT_C,BWB100_BYTE
                MOV BIT_C,#08H
                INC R0
                DEC R5
                MOV A,@R0
BWB100_BYTE:    DJNZ R4,BWB100_BIT
                MOV R4,#03H
                JMP SET_BWB

        ;------------ 4B 1100 ROUTINE --------------

        ;---------- SET POSITION ----------
BWB4_1100:      MOV BIT_C,BIT_C_TMP
                MOV A,CODE_TMP
                MOV R0,R0_TMP
                MOV R5,R5_TMP
                MOV R4,#02H
                CJNE R0,#7FH,BWB110_BIT
BWB110_TMP_BIT:RL A
                DJNZ BIT_C,BWB110_TMP_BYTE
                MOV BIT_C,#08H
                MOV R5,#RECV_SIZE
                MOV R0,#DATA_RECV
                CLR TMP_DATA
                MOV A,@R0
                DJNZ R4,BWB110_BIT
                MOV R4,#02H
                JMP SET_BWB
BWB110_TMP_BYTE: DJNZ R4,BWB110_TMP_BIT
                MOV R4,#02H
                JMP SET_BWB

BWB110_BIT:     RL A
                DJNZ BIT_C,BWB110_BYTE
                MOV BIT_C,#08H
                INC R0
                DEC R5
                MOV A,@R0
BWB110_BYTE:    DJNZ R4,BWB110_BIT
                MOV R4,#02H
        ;---------- SET PIXEL ----------
SET_BWB:        MOV CODE_TMP,A
                MOV A,A0
                MOV DPTR,#SETBWB_TAB
                RL A
                INC A
                MOVC A,@A+DPTR
                PUSH ACC
                MOV A,A0
                RL A
                MOVC A,@A+DPTR
                PUSH ACC
                MOV A,A0
                ADD A,R4
                MOV A0,A
                RET
        ;---------- TABLE SET ----------
BWB0:           CJNE R4,#08H,CHK_BWB0
                MOV @R1,#00000000B
                INC R1
                MOV REF_OBYTE,REF_NBYTE
                MOV REF_NBYTE,@R1
                CJNE R1,#4EH,
                        GO_WB_ENDHOR0
                MOV R1,#REF_LINE  ; NEW LINE
                MOV DPS,#00H
                REPT 14
                MOV A,@R1
                MOVX @DPTR,A
                INC DPTR
                INC R1
                ENDM
                MOV R1,#REF_LINE
                MOV A0,#00H
                MOV DPS,#01H
                INC CODE_LINE_NUM
                MOV A,CODE_LINE_NUM
                CJNE A,#ROW,
                        GO_WB_ENDHOR0
                JMP END_PIC
GO_WB_ENDHOR0: JMP END_HOR
CHK_BWB0:       JC LESS_BWB0
                MOV @R1,#00000000B
                INC R1
                MOV REF_OBYTE,REF_NBYTE
                MOV REF_NBYTE,@R1
                CJNE R1,#4EH,
                        CONT_CHK_BWB0
                MOV R1,#REF_LINE  ; NEW LINE
                MOV DPS,#00H
                REPT 14
                MOV A,@R1
                MOVX @DPTR,A
                INC DPTR
                INC R1
                ENDM
                MOV R1,#REF_LINE
                MOV DPS,#01H
                MOV A0,#00H
                INC CODE_LINE_NUM
                MOV A,CODE_LINE_NUM
                CJNE A,#ROW,
                        GO_WB_ENDHOR0
                JMP END_PIC
CONT_CHK_BWB0: MOV A,R4
                CLR C
                SUBB A,#08H
                MOV R4,A
                JMP BWB0
LESS_BWB0:      MOV A,R4
                MOV DPTR,#LESS8_BWB
                RL A
                INC A
                MOVC A,@A+DPTR
                PUSH ACC
                MOV A,R4
                RL A
                MOVC A,@A+DPTR
                PUSH ACC
                RET
        ;==============================
BWB1:           CJNE R4,#07H,CHK_BWB1
                MOV A,@R1
                ANL A,#10000000B
                MOV @R1,A
                INC R1
                MOV REF_OBYTE,REF_NBYTE
                MOV REF_NBYTE,@R1
                CJNE R1,#4EH,
                        GO_WB_ENDHOR1
                MOV R1,#REF_LINE  ; NEW LINE
                MOV DPS,#00H
                REPT 14
                MOV A,@R1
                MOVX @DPTR,A
                INC DPTR
                INC R1
```

```
                ENDM
                MOV R1,#REF_LINE
                MOV DPS,#01H
                MOV A0,#00H
                INC CODE_LINE_NUM
                MOV A,CODE_LINE_NUM
                CJNE A,#ROW,
                      GO_WB_ENDHOR1
                JMP END_PIC
GO_WB_ENDHOR1: JMP END_HOR
CHK_BWB1:       JC LESS_BWB1
                MOV A,@R1
                ANL A,#10000000B
                MOV @R1,A
                INC R1
                MOV REF_OBYTE,REF_NBYTE
                MOV REF_NBYTE,@R1
                CJNE R1,#4EH,
                      CONT_CHK_BWB1
                MOV R1,#REF_LINE  ; NEW LINE
                MOV DPS,#00H
                REPT 14
                MOV A,@R1
                MOVX @DPTR,A
                INC DPTR
                INC R1
                ENDM
                MOV R1,#REF_LINE
                MOV DPS,#01H
                MOV A0,#00H
                INC CODE_LINE_NUM
                MOV A,CODE_LINE_NUM
                CJNE A,#ROW,
                      GO_WB_ENDHOR1
                JMP END_PIC
CONT_CHK_BWB1: MOV A,R4
                CLR C
                SUBB A,#07H
                MOV R4,A
                JMP BWB0
LESS_BWB1:      MOV A,R4
                MOV DPTR,#LESS7_BWB
                RL A
                INC A
                MOVC A,@A+DPTR
                PUSH ACC
                MOV A,R4
                RL A
                MOVC A,@A+DPTR
                PUSH ACC
                RET
L7_BWB0:        JMP END_HOR
L7_BWB1:        MOV A,@R1
                ANL A,#10111111B
                MOV @R1,A
                JMP END_HOR
L7_BWB2:        MOV A,@R1
                ANL A,#10011111B
                MOV @R1,A
                JMP END_HOR
L7_BWB3:        MOV A,@R1
                ANL A,#10001111B
                MOV @R1,A
                JMP END_HOR
L7_BWB4:        MOV A,@R1
                ANL A,#10000111B
                MOV @R1,A
                JMP END_HOR
L7_BWB5:        MOV A,@R1
                ANL A,#10000011B
                MOV @R1,A
                JMP END_HOR
L7_BWB6:        MOV A,@R1
                ANL A,#10000001B
                MOV @R1,A
                JMP END_HOR
          ;==============================
BWB2:           CJNE R4,#06H,CHK_BWB2
                MOV A,@R1
                ANL A,#11000000B
                MOV @R1,A
                INC R1
                MOV REF_OBYTE,REF_NBYTE
                MOV REF_NBYTE,@R1
                CJNE R1,#4EH,
                      GO_WB_ENDHOR2
                MOV R1,#REF_LINE  ; NEW LINE
                MOV DPS,#00H
                REPT 14
                MOV A,@R1
                MOVX @DPTR,A
                INC DPTR
                INC R1
                ENDM
                MOV R1,#REF_LINE
                MOV DPS,#01H
                MOV A0,#00H
                INC CODE_LINE_NUM
                MOV A,CODE_LINE_NUM
                CJNE A,#ROW,
                      GO_WB_ENDHOR2
                JMP END_PIC
GO_WB_ENDHOR2: JMP END_HOR
CHK_BWB2:       JC LESS_BWB2
                MOV A,@R1
                ANL A,#11000000B
                MOV @R1,A
                INC R1
                MOV REF_OBYTE,REF_NBYTE
                MOV REF_NBYTE,@R1
                CJNE R1,#4EH,
                      CONT_CHK_BWB2
                MOV R1,#REF_LINE  ; NEW LINE
                MOV DPS,#00H
                REPT 14
                MOV A,@R1
                MOVX @DPTR,A
                INC DPTR
                INC R1
                ENDM
                MOV R1,#REF_LINE
                MOV DPS,#01H
                MOV A0,#00H
                INC CODE_LINE_NUM
                MOV A,CODE_LINE_NUM
                CJNE A,#ROW,
                      GO_WB_ENDHOR2
                JMP END_PIC
CONT_CHK_BWB2: MOV A,R4
                CLR C
                SUBB A,#06H
                MOV R4,A
                JMP BWB0
LESS_BWB2:      MOV A,R4
                MOV DPTR,#LESS6_BWB
                RL A
                INC A
                MOVC A,@A+DPTR
                PUSH ACC
                MOV A,R4
                RL A
                MOVC A,@A+DPTR
                PUSH ACC
                RET
```

```
L6_BWB0:        JMP END_HOR
L6_BWB1:        MOV A,@R1
                ANL A,#11011111B
                MOV @R1,A
                JMP END_HOR
L6_BWB2:        MOV A,@R1
                ANL A,#11001111B
                MOV @R1,A
                JMP END_HOR
L6_BWB3:        MOV A,@R1
                ANL A,#11000111B
                MOV @R1,A
                JMP END_HOR
L6_BWB4:        MOV A,@R1
                ANL A,#11000011B
                MOV @R1,A
                JMP END_HOR
L6_BWB5:        MOV A,@R1
                ANL A,#11000001B
                MOV @R1,A
                JMP END_HOR
        ;================================
BWB3:           CJNE R4,#05H,CHK_BWB3
                MOV A,@R1
                ANL A,#11100000B
                MOV @R1,A
                INC R1
                MOV REF_OBYTE,REF_NBYTE
                MOV REF_NBYTE,@R1
                CJNE R1,#4EH,
                        GO_WB_ENDHOR3
                MOV R1,#REF_LINE ; NEW LINE
                MOV DPS,#00H
                REPT 14
                MOV A,@R1
                MOVX @DPTR,A
                INC DPTR
                INC R1
                ENDM
                MOV R1,#REF_LINE
                MOV DPS,#01H
                MOV A0,#00H
                INC CODE_LINE_NUM
                MOV A,CODE_LINE_NUM
                CJNE A,#ROW,
                        GO_WB_ENDHOR3
                JMP END_PIC
GO_WB_ENDHOR3: JMP END_HOR
CHK_BWB3:       JC LESS_BWB3
                MOV A,@R1
                ANL A,#11100000B
                MOV @R1,A
                INC R1
                MOV REF_OBYTE,REF_NBYTE
                MOV REF_NBYTE,@R1
                CJNE R1,#4EH,
                        CONT_CHK_BWB3
                MOV R1,#REF_LINE  ; NEW LINE
                MOV DPS,#00H
                REPT 14
                MOV A,@R1
                MOVX @DPTR,A
                INC DPTR
                INC R1
                ENDM
                MOV R1,#REF_LINE
                MOV DPS,#01H
                MOV A0,#00H
                INC CODE_LINE_NUM
                MOV A,CODE_LINE_NUM
                CJNE A,#ROW,
                        GO_WB_ENDHOR3
                JMP END_PIC
CONT_CHK_BWB3: MOV A,R4
                CLR C
                SUBB A,#05H
                MOV R4,A
                JMP BWB0
LESS_BWB3:      MOV A,R4
                MOV DPTR,#LESS5_BWB
                RL A
                INC A
                MOVC A,@A+DPTR
                PUSH ACC
                MOV A,R4
                RL A
                MOVC A,@A+DPTR
                PUSH ACC
                RET

L5_BWB0:        JMP END_HOR
L5_BWB1:        MOV A,@R1
                ANL A,#11101111B
                MOV @R1,A
                JMP END_HOR
L5_BWB2:        MOV A,@R1
                ANL A,#11100111B
                MOV @R1,A
                JMP END_HOR
L5_BWB3:        MOV A,@R1
                ANL A,#11100011B
                MOV @R1,A
                JMP END_HOR
L5_BWB4:        MOV A,@R1
                ANL A,#11100001B
                MOV @R1,A
                JMP END_HOR
        ;================================
BWB4:           CJNE R4,#04H,CHK_BWB4
                MOV A,@R1
                ANL A,#11110000B
                MOV @R1,A
                INC R1
                MOV REF_OBYTE,REF_NBYTE
                MOV REF_NBYTE,@R1
                CJNE R1,#4EH,
                        GO_WB_ENDHOR4
                MOV R1,#REF_LINE  ; NEW LINE
                MOV DPS,#00H
                REPT 14
                MOV A,@R1
                MOVX @DPTR,A
                INC DPTR
                INC R1
                ENDM
                MOV R1,#REF_LINE
                MOV DPS,#01H
                MOV A0,#00H
                INC CODE_LINE_NUM
                MOV A,CODE_LINE_NUM
                CJNE A,#ROW,
                        GO_WB_ENDHOR4
                JMP END_PIC
GO_WB_ENDHOR4: JMP END_HOR
CHK_BWB4:       JC LESS_BWB4
                MOV A,@R1
                ANL A,#11110000B
                MOV @R1,A
                INC R1
                MOV REF_OBYTE,REF_NBYTE
                MOV REF_NBYTE,@R1
                CJNE R1,#4EH,
                        CONT_CHK_BWB4
                MOV R1,#REF_LINE  ; NEW LINE
                MOV DPS,#00H
                REPT 14
```

```
                MOV A,@R1
                MOVX @DPTR,A
                INC DPTR
                INC R1
                ENDM
                MOV R1,#REF_LINE
                MOV DPS,#01H
                MOV A0,#00H
                INC CODE_LINE_NUM
                MOV A,CODE_LINE_NUM
                CJNE A,#ROW,
                      GO_WB_ENDHOR4
                JMP END_PIC
CONT_CHK_BWB4: MOV A,R4
                CLR C
                SUBB A,#04H
                MOV R4,A
                JMP BWB0
LESS_BWB4:      DJNZ R4,N2_BWB4
                MOV A,@R1
                ANL A,#11110111B
                MOV @R1,A
                JMP END_HOR
N2_BWB4:        DJNZ R4,N3_BWB4
                MOV A,@R1
                ANL A,#11110011B
                MOV @R1,A
                JMP END_HOR
N3_BWB4:        MOV A,@R1
                ANL A,#11110001B
                MOV @R1,A
                JMP END_HOR
        ;================================
BWB5:           CJNE R4,#03H,CHK_BWB5

                MOV A,@R1
                ANL A,#11111000B
                MOV @R1,A
                INC R1
                MOV REF_OBYTE,REF_NBYTE
                MOV REF_NBYTE,@R1
                CJNE R4,#4EH,
                      GO_WB_ENDHOR5
                MOV R1,#REF_LINE  ; NEW LINE
                MOV DPS,#00H
                REPT 14
                MOV A,@R1
                MOVX @DPTR,A
                INC DPTR
                INC R1
                ENDM
                MOV R1,#REF_LINE
                MOV DPS,#01H
                MOV A0,#00H
                INC CODE_LINE_NUM
                MOV A,CODE_LINE_NUM
                CJNE A,#ROW,
                      GO_WB_ENDHOR5
                JMP END_PIC
GO_WB_ENDHOR5: JMP END_HOR
CHK_BWB5:       JC LESS_BWB5
                MOV A,@R1
                ANL A,#11111000B
                MOV @R1,A
                INC R1
                MOV REF_OBYTE,REF_NBYTE
                MOV REF_NBYTE,@R1
                CJNE R1,#4EH,
                      CONT_CHK_BWB5
                MOV R1,#REF_LINE ; NEW LINE
                MOV DPS,#00H
                REPT 14
                MOV A,@R1
                MOVX @DPTR,A
                INC DPTR
                INC R1
                ENDM
                MOV R1,#REF_LINE
                MOV DPS,#01H
                MOV A0,#00H
                INC CODE_LINE_NUM
                MOV A,CODE_LINE_NUM
                CJNE A,#ROW,
                      GO_WB_ENDHOR5
                JMP END_PIC
CONT_CHK_BWB5: DEC R4
                DEC R4
                JMP BWB0
LESS_BWB5:      DJNZ R4,N2_BWB5
                MOV A,@R1
                ANL A,#11111011B
                MOV @R1,A
                JMP END_HOR

N2_BWB5:        MOV A,@R1
                ANL A,#11111001B
                MOV @R1,A
                JMP END_HOR
        ;==============================
BWB6:           CJNE R4,#02H,CHK_BWB6
                MOV A,@R1
                ANL A,#11111100B
                MOV @R1,A
                INC R1
                MOV REF_OBYTE,REF_NBYTE
                MOV REF_NBYTE,@R1
                CJNE R1,#4EH,
                      GO_WB_ENDHOR6
                MOV R1,#REF_LINE  ; NEW LINE
                MOV DPS,#00H
                REPT 14
                MOV A,@R1
                MOVX @DPTR,A
                INC DPTR
                INC R1
                ENDM
                MOV R1,#REF_LINE
                MOV DPS,#01H
                MOV A0,#00H
                INC CODE_LINE_NUM
                MOV A,CODE_LINE_NUM
                CJNE A,#ROW,
                      GO_WB_ENDHOR6
                JMP END_PIC
GO_WB_ENDHOR6: JMP END_HOR
CHK_BWB6:       JC LESS_BWB6
                MOV A,@R1
                ANL A,#11111100B
                MOV @R1,A
                INC R1
                MOV REF_OBYTE,REF_NBYTE
                MOV REF_NBYTE,@R1
                CJNE R1,#4EH,
                      CONT_CHK_BWB6
                MOV R1,#REF_LINE  ; NEW LINE
                MOV DPS,#00H
                REPT 14
                MOV A,@R1
                MOVX @DPTR,A
                INC DPTR
                INC R1
                ENDM
                MOV R1,#REF_LINE
                MOV DPS,#01H
```

```
                MOV A0,#00H
                INC CODE_LINE_NUM
                MOV A,CODE_LINE_NUM
                CJNE A,#ROW,
                      GO_WB_ENDHOR6
                JMP END_PIC
CONT_CHK_BWB6: DEC R4
                DEC R4
                JMP BWB0
LESS_BWB6:      MOV A,@R1
                ANL A,#11111101B
                MOV @R1,A
                JMP END_HOR
        ;==============================
BWB7:           CJNE R4,#01H,CHK_BWB7
                MOV A,@R1
                ANL A,#11111110B
                MOV @R1,A
                INC R1
                MOV REF_OBYTE,REF_NBYTE
                MOV REF_NBYTE,@R1
                CJNE R1,#4EH,
                      GO_WB_ENDHOR7
                MOV R1,#REF_LINE  ; NEW LINE
                MOV DPS,#00H
                REPT 14
                MOV A,@R1
                MOVX @DPTR,A
                INC DPTR
                INC R1
                ENDM
                MOV R1,#REF_LINE
                MOV DPS,#01H
                MOV A0,#00H
                INC CODE_LINE_NUM
                MOV A,CODE_LINE_NUM
                CJNE A,#ROW,
                      GO_WB_ENDHOR7
                JMP END_PIC
GO_WB_ENDHOR7: JMP END_HOR
CHK_BWB7:       MOV A,@R1
                ANL A,#11111110B
                MOV @R1,A
                INC R1
                MOV REF_OBYTE,REF_NBYTE
                MOV REF_NBYTE,@R1
                CJNE R1,#4EH,
                       CONT_CHK_BWB7
                MOV R1,#REF_LINE  ; NEW LINE
                REPT 14
                MOV A,@R1
                MOVX @DPTR,A
                INC DPTR
                INC R1
                ENDM
                MOV R1,#REF_LINE
                MOV A0,#00H
                INC CODE_LINE_NUM
                MOV A,CODE_LINE_NUM
                CJNE A,#ROW,
                      GO_WB_ENDHOR7
                JMP END_PIC
CONT_CHK_BWB7: DEC R4
                JMP BWB0

;-------------- LESS THAN 8 BIT --------------------
L8_BWB0:        JMP END_HOR

L8_BWB1:        MOV A,@R1
                ANL A,#01111111B
                MOV @R1,A
                JMP END_HOR

L8_BWB2:        MOV A,@R1
                ANL A,#00111111B
                MOV @R1,A
                JMP END_HOR

L8_BWB3:        MOV A,@R1
                ANL A,#00011111B
                MOV @R1,A
                JMP END_HOR

L8_BWB4:        MOV A,@R1
                ANL A,#00001111B
                MOV @R1,A
                JMP END_HOR

L8_BWB5:        MOV A,@R1
                ANL A,#00000111B
                MOV @R1,A
                JMP END_HOR

L8_BWB6:        MOV A,@R1
                ANL A,#00000011B
                MOV @R1,A
                JMP END_HOR

L8_BWB7:        MOV A,@R1
                ANL A,#00000001B
                MOV @R1,A
                JMP END_HOR

;-------------- HOR MODE IN BLACKS PIXEL ----------------
;---------------------- BLACK -------------------------
HOR_BBW:        MOV CODE_TMP,A
                MOV BIT_C_TMP,BIT_C
                MOV R0_TMP,R0
                MOV R5_TMP,R5
                MOV R4,#00H
              ; ------ GET REF. LINE INDEX -----
                MOV A,A0
                MOV DPTR,#GET_R1
                MOVC A,@A+DPTR
                MOV R1,A ;GET REF. LINE INDEX
                MOV A,CODE_TMP

R_BBW41:        RLC A     ;ROTATE BIT
                XCH A,R4
                RLC A
                XCH A,R4
                DJNZ BIT_C,R_BBW42
                MOV BIT_C,#08H
                INC R0
                JNB TMP_DATA,CONTD_14
                MOV R0,#DATA_RECV
                MOV R5,#RECV_SIZE
                CLR TMP_DATA
                MOV A,@R0
                JMP R_BBW42
CONTD_14:       MOV A,@R0
                DJNZ R5,R_BBW42
              ; GO R0TATE42
                SETB TMP_DATA
                MOV S_BYTE,#29H
                JMP END_48_BYTES
R_BBW42:        RLC A     ;ROTATE BIT
                XCH A,R4
                RLC A
                XCH A,R4
                DJNZ BIT_C,R_BBW43
                MOV BIT_C,#08H
                INC R0
                JNB TMP_DATA,CONTD_15
                MOV R0,#DATA_RECV
                MOV R5,#RECV_SIZE
```

```
                CLR TMP_DATA
                MOV A,@R0
                JMP R_BBW43
CONTD_15:       MOV A,@R0
                DJNZ R5,R_BBW43
        ; GO R0TATE43
                SETB TMP_DATA
                MOV S_BYTE,#2AH
                JMP END_48_BYTES
R_BBW43:        RLC A     ;ROTATE BIT
                XCH A,R4
                RLC A
                XCH A,R4
                DJNZ BIT_C,R_BBW44
                MOV BIT_C,#08H
                INC R0
                JNB TMP_DATA,CONTD_16
                MOV R0,#DATA_RECV
                MOV R5,#RECV_SIZE
                CLR TMP_DATA
                MOV A,@R0
                JMP R_BBW44
CONTD_16:       MOV A,@R0
                DJNZ R5,R_BBW44
        ; GO R0TATE44
                SETB TMP_DATA
                MOV S_BYTE,#2BH
                JMP END_48_BYTES
R_BBW44:        RLC A     ;ROTATE BIT
                XCH A,R4
                RLC A
                XCH A,R4
                DJNZ BIT_C,CODE4_BBW
                MOV BIT_C,#08H
                INC R0
                JNB TMP_DATA,CONTD_17
                MOV R0,#DATA_RECV
                MOV R5,#RECV_SIZE
                CLR TMP_DATA
                MOV A,@R0
                JMP CODE4_BBW
CONTD_17:       MOV A,@R0
                DJNZ R5,CODE4_BBW
        ; GO COD4_BBW
                SETB TMP_DATA
                MOV S_BYTE,#2CH
                JMP END_48_BYTES
        ;---------- 4 BIT INDEX TABLE  ----------
CODE4_BBW:      MOV CODE_TMP2,A
                MOV R0_TMP2,R0
                MOV R5_TMP2,R5
                MOV BIT_C_TMP2,BIT_C
                MOV DPTR,#BBW_INDEX
                MOV A,R4
                RL A
                INC A
                MOVC A,@A+DPTR
                PUSH ACC

                MOV A,R4
                RL A
                MOVC A,@A+DPTR
                PUSH ACC
                RET
        ;-------------- 4 BIT LABEL ----------------
        ;------------ 4B 0000 ROUTINE --------------

BBW4_0000:      MOV A,CODE_TMP2
                MOV R4,#00H

R0_BBW81:       RLC A     ;ROTATE BIT
                XCH A,R4
                RLC A
                XCH A,R4
                DJNZ BIT_C,R0_BBW82
                MOV BIT_C,#08H
                INC R0
                MOV A,@R0
                DJNZ R5,R0_BBW82
        ; GO R0TATE82
                SETB TMP_DATA
                MOV S_BYTE,#2DH
                JMP END_48_BYTES
R0_BBW82:       RLC A     ;ROTATE BIT
                XCH A,R4
                RLC A
                XCH A,R4
                DJNZ BIT_C,R0_BBW83
                MOV BIT_C,#08H
                INC R0
                MOV A,@R0
                DJNZ R5,R0_BBW83
        ; GO R0TATE83
                SETB TMP_DATA
                MOV S_BYTE,#2EH
                JMP END_48_BYTES
R0_BBW83:       RLC A     ;ROTATE BIT
                XCH A,R4
                RLC A
                XCH A,R4
                DJNZ BIT_C,R0_BBW84
                MOV BIT_C,#08H
                INC R0
                MOV A,@R0
                DJNZ R5,R0_BBW84
        ; GO R0TATE84
                SETB TMP_DATA
                MOV S_BYTE,#2FH
                JMP END_48_BYTES
R0_BBW84:       RLC A     ;ROTATE BIT
                XCH A,R4
                RLC A
                XCH A,R4
                DJNZ BIT_C,R0_BBW85
                MOV BIT_C,#08H
                INC R0
                MOV A,@R0
                DJNZ R5,R0_BBW85
        ; GO R0TATE85
                SETB TMP_DATA
                MOV S_BYTE,#30H
                JMP END_48_BYTES
R0_BBW85:       RLC A     ;ROTATE BIT
                XCH A,R4
                RLC A
                XCH A,R4
                DJNZ BIT_C,R0_BBW86
                MOV BIT_C,#08H
                INC R0
                MOV A,@R0
                DJNZ R5,R0_BBW86
        ; GO R0TATE86
                SETB TMP_DATA
                MOV S_BYTE,#31H
                JMP END_48_BYTES
R0_BBW86:       RLC A     ;ROTATE BIT
                XCH A,R4
                RLC A
                XCH A,R4
                DJNZ BIT_C,R0_BBW87
                MOV BIT_C,#08H
                INC R0
                MOV A,@R0
                DJNZ R5,R0_BBW87
        ; GO R0TATE87
                SETB TMP_DATA
```

```
                MOV S_BYTE,#32H
                JMP END_48_BYTES
R0_BBW87:       RLC A     ;ROTATE BIT
                XCH A,R4
                RLC A
                XCH A,R4
                DJNZ BIT_C,R0_BBW88
                MOV BIT_C,#08H
                INC R0
                MOV A,@R0
                DJNZ R5,R0_BBW88
        ; GO R0TATE88
                SETB TMP_DATA
                MOV S_BYTE,#33H
                JMP END_48_BYTES
R0_BBW88:       RLC A     ;ROTATE BIT
                XCH A,R4
                RLC A
                XCH A,R4
                DJNZ BIT_C,CODEBBW_00
                MOV BIT_C,#08H
                INC R0
                MOV A,@R0
                DJNZ R5,CODEBBW_00
        ; GO CODE8_BBW
                SETB TMP_DATA
                MOV S_BYTE,#34H
                JMP END_48_BYTES
        ;----------- GET INDEX ------------
CODEBBW_00:     MOV DPTR,#B_INDEX8B0
                MOV A,R4
                MOVC A,@A+DPTR
                RL A
                MOV TMP_R3,A
        ;------ GET NUMBER OF PIXEL -------
                MOV DPTR,#B_DATA
                MOVC A,@A+DPTR
                MOV NUM_PIXEL,A
        ;------ GET NUMBER OF BIT ---------
                MOV A,TMP_R3
                INC A
                MOVC A,@A+DPTR
                MOV R4,A
        ;---------- SET POSITION ----------
                MOV BIT_C,BIT_C_TMP2
                MOV A,CODE_TMP2
                MOV R0,R0_TMP2
                MOV R5,R5_TMP2
                CJNE R0,#7FH,BBW00_BIT
BBW00_TMP_BIT:  RL A
                DJNZ BIT_C,BBW00_TMP_BYTE
                MOV BIT_C,#08H
                MOV R5,#RECV_SIZE
                MOV R0,#DATA_RECV
                CLR TMP_DATA
                MOV A,@R0
BBW00_TMP_BYTE: DJNZ R4,BBW00_TMP_BIT
                MOV R4,NUM_PIXEL
                JMP SET_BBW

BBW00_BIT:      RL A
                DJNZ BIT_C,BBW00_BYTE
                MOV BIT_C,#08H
                INC R0
                DEC R5
                MOV A,@R0
BBW00_BYTE:     DJNZ R4,BBW00_BIT
                MOV R4,NUM_PIXEL
                JMP SET_BBW
        ;------------ 4B 0001 ROUTINE --------------
BBW4_0001:      MOV A,CODE_TMP2
                MOV R4,#00H
R1_BBW31:       RLC A     ;ROTATE BIT
                XCH A,R4
                RLC A
                XCH A,R4
                DJNZ BIT_C,R1_BBW32
                MOV BIT_C,#08H
                INC R0
                MOV A,@R0
                DJNZ R5,R1_BBW32
        ; GO R0TATE32
                SETB TMP_DATA
                MOV S_BYTE,#35H
                JMP END_48_BYTES
R1_BBW32:       RLC A     ;ROTATE BIT
                XCH A,R4
                RLC A
                XCH A,R4
                DJNZ BIT_C,CODEBBW_01
                MOV BIT_C,#08H
                INC R0
                MOV A,@R0
                DJNZ R5,CODEBBW_01
        ; GO CODE8_BBW
                SETB TMP_DATA
                MOV S_BYTE,#36H
                JMP END_48_BYTES

        ;----------- GET INDEX ------------
CODEBBW_01:     MOV DPTR,#B_INDEX8B1
                MOV A,R4
                MOVC A,@A+DPTR
                RL A
                MOV TMP_R3,A
        ;------ GET NUMBER OF PIXEL -------
                MOV DPTR,#B_DATA
                MOVC A,@A+DPTR
                MOV NUM_PIXEL,A
        ;------ GET NUMBER OF BIT ---------
                MOV A,TMP_R3
                INC A
                MOVC A,@A+DPTR
                MOV R4,A
        ;---------- SET POSITION ----------
                MOV BIT_C,BIT_C_TMP2
                MOV A,CODE_TMP2
                MOV R0,R0_TMP2
                MOV R5,R5_TMP2
                CJNE R0,#7FH,BBW01_BIT
BBW01_TMP_BIT:  RL A
                DJNZ BIT_C,BBW01_TMP_BYTE
                MOV BIT_C,#08H
                MOV R5,#RECV_SIZE
                MOV R0,#DATA_RECV
                CLR TMP_DATA
                MOV A,@R0
BBW01_TMP_BYTE: DJNZ R4,BBW01_TMP_BIT
                MOV R4,NUM_PIXEL
                JMP SET_BBW

BBW01_BIT:      RL A
                DJNZ BIT_C,BBW01_BYTE
                MOV BIT_C,#08H
                INC R0
                DEC R5
                MOV A,@R0
BBW01_BYTE:     DJNZ R4,BBW01_BIT
                MOV R4,NUM_PIXEL
                JMP SET_BBW
        ;------------ 4B 0010 ROUTINE --------------
        ;-------------- SET POSITION ---------------
BBW4_0010:      MOV A,CODE_TMP2
```

```
                MOV R4,#06H
                JMP SET_BBW
        ;------------ 4B 0011 ROUTINE --------------
BBW4_0011:      MOV A,CODE_TMP2

                MOV R4,#05H
                JMP SET_BBW
        ;------------ 4B 0100 ROUTINE --------------

        ;---------- SET POSITION ----------
BBW4_0100:      MOV BIT_C,BIT_C_TMP
                MOV A,CODE_TMP
                MOV R0,R0_TMP
                MOV R5,R5_TMP
                MOV R4,#03H
                CJNE R0,#7FH,BBW010_BIT
BBW010_TMP_BIT:RL A
                DJNZ BIT_C,BBW010_TMP_BYTE
                MOV BIT_C,#08H
                MOV R5,#RECV_SIZE
                MOV R0,#DATA_RECV
                CLR TMP_DATA
                MOV A,@R0
BBW010_TMP_BYTE: DJNZ R4,BBW010_TMP_BIT
                MOV R4,#01H
                JMP SET_BBW

BBW010_BIT:     RL A
                DJNZ BIT_C,BBW010_BYTE
                MOV BIT_C,#08H
                INC R0
                DEC R5
                MOV A,@R0
BBW010_BYTE:    DJNZ R4,BBW010_BIT
                MOV R4,#01H
                JMP SET_BBW
        ;------------ 4B 0110 ROUTINE --------------

        ;---------- SET POSITION ----------
BBW4_0110:      MOV BIT_C,BIT_C_TMP
                MOV A,CODE_TMP
                MOV R0,R0_TMP
                MOV R5,R5_TMP
                MOV R4,#03H
                CJNE R0,#7FH,BBW011_BIT
BBW011_TMP_BIT:RL A
                DJNZ BIT_C,BBW011_TMP_BYTE
                MOV BIT_C,#08H
                MOV R5,#RECV_SIZE
                MOV R0,#DATA_RECV
                CLR TMP_DATA
                MOV A,@R0
BBW011_TMP_BYTE: DJNZ R4,BBW011_TMP_BIT
                MOV R4,#04H
                JMP SET_BBW

BBW011_BIT:     RL A
                DJNZ BIT_C,BBW011_BYTE
                MOV BIT_C,#08H
                INC R0
                DEC R5
                MOV A,@R0
BBW011_BYTE:    DJNZ R4,BBW011_BIT
                MOV R4,#04H
                JMP SET_BBW
        ;------------ 4B 1000 ROUTINE --------------
        ;---------- SET POSITION ----------
BBW4_1000:      MOV BIT_C,BIT_C_TMP
                MOV A,CODE_TMP
                MOV R0,R0_TMP
                MOV R5,R5_TMP
                MOV R4,#02H
                CJNE R0,#7FH,BBW100_BIT
BBW100_TMP_BIT:RL A
                DJNZ BIT_C,BBW100_TMP_BYTE
                MOV BIT_C,#08H
                MOV R5,#RECV_SIZE
                MOV R0,#DATA_RECV
                CLR TMP_DATA
                MOV A,@R0
BBW100_TMP_BYTE: DJNZ R4,BBW100_TMP_BIT
                MOV R4,#03H
                JMP SET_BBW

BBW100_BIT:     RL A
                DJNZ BIT_C,BBW100_BYTE
                MOV BIT_C,#08H
                INC R0
                DEC R5
                MOV A,@R0
BBW100_BYTE:    DJNZ R4,BBW100_BIT
                MOV R4,#03H
                JMP SET_BBW

        ;------------ 4B 1100 ROUTINE --------------
        ;---------- SET POSITION ----------
BBW4_1100:      MOV BIT_C,BIT_C_TMP
                MOV A,CODE_TMP
                MOV R0,R0_TMP
                MOV R5,R5_TMP
                MOV R4,#02H
                CJNE R0,#7FH,BBW110_BIT
BBW110_TMP_BIT: RL A
                DJNZ BIT_C,BBW110_TMP_BYTE
                MOV BIT_C,#08H
                MOV R5,#RECV_SIZE
                MOV R0,#DATA_RECV
                CLR TMP_DATA
                MOV A,@R0
BBW110_TMP_BYTE: DJNZ R4,BBW110_TMP_BIT
                MOV R4,#02H
                JMP SET_BBW

BBW110_BIT:     RL A
                DJNZ BIT_C,BBW110_BYTE
                MOV BIT_C,#08H
                INC R0
                DEC R5
                MOV A,@R0
BBW110_BYTE:    DJNZ R4,BBW110_BIT
                MOV R4,#02H
        ;---------- SET PIXEL ----------
SET_BBW:        MOV CODE_TMP,A
                MOV A,A0
                MOV DPTR,#SETBBW_TAB
                RL A
                INC A
                MOVC A,@A+DPTR
                PUSH ACC
                MOV A,A0
                RL A
                MOVC A,@A+DPTR
                PUSH ACC
                MOV A,A0
                ADD A,R4
                MOV A0,A
                RET
        ;---------- TABLE SET ----------
BBW0:           CJNE R4,#08H,CHK_BBW0
                MOV @R1,#00000000B
                INC R1
                MOV REF_OBYTE,REF_NBYTE
                MOV REF_NBYTE,@R1
                JMP HOR_WBW
CHK_BBW0:       JC LESS_BBW0
```

```
                MOV @R1,#00000000B
                INC R1
                MOV REF_OBYTE,REF_NBYTE
                MOV REF_NBYTE,@R1
                MOV A,R4
                CLR C
                SUBB A,#08H
                MOV R4,A
                JMP BBW0
LESS_BBW0:      MOV A,R4
                MOV DPTR,#LESS8_BBW
                RL A
                INC A
                MOVC A,@A+DPTR
                PUSH ACC
                MOV A,R4
                RL A
                MOVC A,@A+DPTR
                PUSH ACC
                RET
        ;==============================
BBW1:           CJNE R4,#07H,CHK_BBW1
                MOV A,@R1
                ANL A,#10000000B
                MOV @R1,A
                INC R1
                MOV REF_OBYTE,REF_NBYTE
                MOV REF_NBYTE,@R1
                JMP HOR_WBW
CHK_BBW1:       JC LESS_BBW1
                MOV A,@R1
                ANL A,#10000000B
                MOV @R1,A
                INC R1
                MOV REF_OBYTE,REF_NBYTE
                MOV REF_NBYTE,@R1
                MOV A,R4
                CLR C
                SUBB A,#07H
                MOV R4,A
                JMP BBW0
LESS_BBW1:      MOV A,R4
                MOV DPTR,#LESS7_BBW
                RL A
                INC A
                MOVC A,@A+DPTR
                PUSH ACC
                MOV A,R4
                RL A
                MOVC A,@A+DPTR
                PUSH ACC
                RET
L7_BBW0:        JMP HOR_WBW
L7_BBW1:        MOV A,@R1
                ANL A,#10111111B
                MOV @R1,A
                JMP HOR_WBW
L7_BBW2:        MOV A,@R1
                ANL A,#10011111B
                MOV @R1,A
                JMP HOR_WBW
L7_BBW3:        MOV A,@R1
                ANL A,#10001111B
                MOV @R1,A
                JMP HOR_WBW
L7_BBW4:        MOV A,@R1
                ANL A,#10000111B
                MOV @R1,A
                JMP HOR_WBW
L7_BBW5:        MOV A,@R1
                ANL A,#10000011B
                MOV @R1,A
                JMP HOR_WBW
L7_BBW6:        MOV A,@R1
                ANL A,#10000001B
                MOV @R1,A
                JMP HOR_WBW
        ;==============================
BBW2:           CJNE R4,#06H,CHK_BBW2
                MOV A,@R1
                ANL A,#11000000B
                MOV @R1,A
                INC R1
                MOV REF_OBYTE,REF_NBYTE
                MOV REF_NBYTE,@R1
                JMP HOR_WBW
CHK_BBW2:       JC LESS_BBW2
                MOV A,@R1
                ANL A,#11000000B
                MOV @R1,A
                INC R1
                MOV REF_OBYTE,REF_NBYTE
                MOV REF_NBYTE,@R1
                MOV A,R4
                CLR C
                SUBB A,#06H
                MOV R4,A
                JMP BBW0
LESS_BBW2:      MOV A,R4
                MOV DPTR,#LESS6_BBW
                RL A
                INC A
                MOVC A,@A+DPTR
                PUSH ACC
                MOV A,R4
                RL A
                MOVC A,@A+DPTR
                PUSH ACC
                RET
L6_BBW0:        JMP HOR_WBW
L6_BBW1:        MOV A,@R1
                ANL A,#11011111B
                MOV @R1,A
                JMP HOR_WBW
L6_BBW2:        MOV A,@R1
                ANL A,#11001111B
                MOV @R1,A
                JMP HOR_WBW
L6_BBW3:        MOV A,@R1
                ANL A,#11000111B
                MOV @R1,A
                JMP HOR_WBW
L6_BBW4:        MOV A,@R1
                ANL A,#11000011B
                MOV @R1,A
                JMP HOR_WBW
L6_BBW5:        MOV A,@R1
                ANL A,#11000001B
                MOV @R1,A
                JMP HOR_WBW
        ;==============================
BBW3:           CJNE R4,#05H,CHK_BBW3
                MOV A,@R1
                ANL A,#11100000B
                MOV @R1,A
                INC R1
                MOV REF_OBYTE,REF_NBYTE
                MOV REF_NBYTE,@R1
                JMP HOR_WBW
CHK_BBW3:       JC LESS_BBW3
                MOV A,@R1
                ANL A,#11100000B
```

```
                MOV @R1,A
                INC R1
                MOV REF_OBYTE,REF_NBYTE
                MOV REF_NBYTE,@R1
                MOV A,R4
                CLR C
                SUBB A,#05H
                MOV R4,A
                JMP BBW0
LESS_BBW3:      MOV A,R4
                MOV DPTR,#LESS5_BBW
                RL A
                INC A
                MOVC A,@A+DPTR
                PUSH ACC
                MOV A,R4
                RL A
                MOVC A,@A+DPTR
                PUSH ACC
                RET

L5_BBW0:        JMP HOR_WBW
L5_BBW1:        MOV A,@R1
                ANL A,#11101111B
                MOV @R1,A
                JMP HOR_WBW
L5_BBW2:        MOV A,@R1
                ANL A,#11100111B
                MOV @R1,A
                JMP HOR_WBW
L5_BBW3:        MOV A,@R1
                ANL A,#11100011B
                MOV @R1,A
                JMP HOR_WBW
L5_BBW4:        MOV A,@R1
                ANL A,#11100001B
                MOV @R1,A
                JMP HOR_WBW
        ;==============================
BBW4:           CJNE R4,#054H,CHK_BBW4
                MOV A,@R1
                ANL A,#11110000B
                MOV @R1,A
                INC R1
                MOV REF_OBYTE,REF_NBYTE
                MOV REF_NBYTE,@R1
                JMP HOR_WBW
CHK_BBW4:       JC LESS_BBW4
                MOV A,@R1
                ANL A,#11110000B
                MOV @R1,A
                INC R1
                MOV REF_OBYTE,REF_NBYTE
                MOV REF_NBYTE,@R1
                MOV A,R4
                CLR C
                SUBB A,#04H
                MOV R4,A
                JMP BBW0
LESS_BBW4:      DJNZ R4,N2_BBW4
                MOV A,@R1
                ANL A,#11110111B
                MOV @R1,A
                JMP HOR_WBW

N2_BBW4:        DJNZ R4,N3_BBW4
                MOV A,@R1
                ANL A,11110011B
                MOV @R1,A
                JMP HOR_WBW
N3_BBW4:        MOV A,@R1
                ORL A,#11110001B
                MOV @R1,A
                JMP HOR_WBW
        ;==============================
BBW5:           CJNE R4,#03H,CHK_BBW5
                MOV A,@R1
                ANL A,#11111000B
                MOV @R1,A
                INC R1
                MOV REF_OBYTE,REF_NBYTE
                MOV REF_NBYTE,@R1
                JMP HOR_WBW
CHK_BBW5:       JC LESS_BBW5
                MOV A,@R1
                ANL A,#11111000B
                MOV @R1,A
                INC R1
                MOV REF_OBYTE,REF_NBYTE
                MOV REF_NBYTE,@R1
                MOV A,R4
                CLR C
                SUBB A,#03H
                MOV R4,A
                JMP BBW0
LESS_BBW5:      DJNZ R4,N2_BBW5
                MOV A,@R1
                ANL A,#11111011B
                MOV @R1,A
                JMP HOR_WBW

N2_BBW5:        MOV A,@R1
                ANL A,#11111001B
                MOV @R1,A
                JMP HOR_WBW
        ;==============================
BBW6:           CJNE R4,#02H,CHK_BBW6
                MOV A,@R1
                ANL A,#11111100B
                MOV @R1,A
                INC R1
                MOV REF_OBYTE,REF_NBYTE
                MOV REF_NBYTE,@R1
                JMP HOR_WBW
CHK_BBW6:       JC LESS_BBW6
                MOV A,@R1
                ANL A,#11111100B
                MOV @R1,A
                INC R1
                MOV REF_OBYTE,REF_NBYTE
                MOV REF_NBYTE,@R1
                MOV A,R4
                CLR C
                SUBB A,#02H
                MOV R4,A
                JMP BBW0
LESS_BBW6:      MOV A,@R1
                ANL A,#11111101B
                MOV @R1,A
                JMP HOR_WBW
        ;==============================
BBW7:           CJNE R4,#01H,CHK_BBW7
                MOV A,@R1
                ANL A,#11111110B
                MOV @R1,A
                INC R1
                MOV REF_OBYTE,REF_NBYTE
                MOV REF_NBYTE,@R1
                JMP HOR_WBW
CHK_BBW7:       MOV A,@R1
                ANL A,#11111110B
                MOV @R1,A
                INC R1
                MOV REF_OBYTE,REF_NBYTE
                MOV REF_NBYTE,@R1
```

```
                MOV A,R4
                CLR C
                SUBB A,#01H
                MOV R4,A
                JMP BBW0
;-------------- LESS THAN 8 BIT ---------------------
L8_BBW0:        JMP HOR_WBW

L8_BBW1:        MOV A,@R1
                ANL A,#01111111B
                MOV @R1,A
                JMP HOR_WBW

L8_BBW2:        MOV A,@R1
                ANL A,#00111111B
                MOV @R1,A
                JMP HOR_WBW

L8_BBW3:        MOV A,@R1
                ANL A,#00011111B
                MOV @R1,A
                JMP HOR_WBW

L8_BBW4:        MOV A,@R1
                ANL A,#00001111B
                MOV @R1,A
                JMP HOR_WBW

L8_BBW5:        MOV A,@R1
                ANL A,#00000111B
                MOV @R1,A
                JMP HOR_WBW

L8_BBW6:        MOV A,@R1
                ANL A,#00000011B
                MOV @R1,A
                JMP HOR_WBW

L8_BBW7:        MOV A,@R1
                ANL A,#00000001B
                MOV @R1,A
;----------------------- WHITE -------------------------
HOR_WBW:        MOV BIT_C_TMP,BIT_C
                MOV R0_TMP,R0
                MOV R5_TMP,R5
        ; ------ GET REF. LINE INDEX -----
                MOV A,A0
                MOV DPTR,#GET_R1
                MOVC A,@A+DPTR
                MOV R1,A ;GET REF. LINE INDEX
                MOV A,CODE_TMP
R_WBW1:         RLC A     ;ROTATE BIT
                XCH A,R4
                RLC A
                XCH A,R4
                DJNZ BIT_C,R_WBW2
                MOV BIT_C,#08H
                INC R0
                MOV A,@R0
                DJNZ R5,R_WBW2
        ; GO R0TATE2
                SETB TMP_DATA
                MOV S_BYTE,#37H
                JMP END_48_BYTES
R_WBW2:         RLC A     ;ROTATE BIT
                XCH A,R4
                RLC A
                XCH A,R4
                DJNZ BIT_C,R_WBW3
                MOV BIT_C,#08H
                INC R0
                MOV A,@R0
                DJNZ R5,R_WBW3
        ; GO R0TATE3
                SETB TMP_DATA
                MOV S_BYTE,#38H
                JMP END_48_BYTES
R_WBW3:         RLC A     ;ROTATE BIT
                XCH A,R4
                RLC A
                XCH A,R4
                DJNZ BIT_C,R_WBW4
                MOV BIT_C,#08H
                INC R0
                MOV A,@R0
                DJNZ R5,R_WBW4
        ; GO R0TATE4
                SETB TMP_DATA
                MOV S_BYTE,#39H
                JMP END_48_BYTES
R_WBW4:         RLC A     ;ROTATE BIT
                XCH A,R4
                RLC A
                XCH A,R4
                DJNZ BIT_C,R_WBW5
                MOV BIT_C,#08H
                INC R0
                MOV A,@R0
                DJNZ R5,R_WBW5
        ; GO R0TATE5
                SETB TMP_DATA
                MOV S_BYTE,#3AH
                JMP END_48_BYTES
R_WBW5:         RLC A     ;ROTATE BIT
                XCH A,R4
                RLC A
                XCH A,R4
                DJNZ BIT_C,R_WBW6
                MOV BIT_C,#08H
                INC R0
                MOV A,@R0
                DJNZ R5,R_WBW6
        ; GO R0TATE6
                SETB TMP_DATA
                MOV S_BYTE,#3BH
                JMP END_48_BYTES
R_WBW6:         RLC A     ;ROTATE BIT
                XCH A,R4
                RLC A
                XCH A,R4
                DJNZ BIT_C,R_WBW7
                MOV BIT_C,#08H
                INC R0
                MOV A,@R0
                DJNZ R5,R_WBW7
        ; GO R0TATE7
                SETB TMP_DATA
                MOV S_BYTE,#3CH
                JMP END_48_BYTES
R_WBW7:         RLC A     ;ROTATE BIT
                XCH A,R4
                RLC A
                XCH A,R4
                DJNZ BIT_C,R_WBW8
                MOV BIT_C,#08H
                INC R0
                MOV A,@R0
                DJNZ R5,R_WBW8
        ; GO R0TATE8
                SETB TMP_DATA
                MOV S_BYTE,#3DH
                JMP END_48_BYTES
R_WBW8:         RLC A     ;ROTATE BIT
                XCH A,R4
                RLC A
```

```
                XCH A,R4
                DJNZ BIT_C,CODE_WBW
                MOV BIT_C,#08H
                INC R0
                MOV A,@R0
                DJNZ R5,CODE_WBW
        ; GO CODE_WBW
                SETB TMP_DATA
                MOV S_BYTE,#3EH
                JMP END_48_BYTES

CODE_WBW:       MOV DPTR,#W_INDEX
;GET INDEX OF DATA
                MOV A,R4
                MOVC A,@A+DPTR
                RL A
                MOV INDEX_TMP,A
        ;------ GET NUMBER OF PIXEL -------
                MOV DPTR,#W_DATA
                MOVC A,@A+DPTR
                MOV NUM_PIXEL,A
        ;------ GET NUMBER OF BIT ---------
                MOV A,INDEX_TMP
                INC A
                MOVC A,@A+DPTR
                MOV R4,A
        ;---------- SET POSITION ----------
                MOV BIT_C,BIT_C_TMP
                MOV A,CODE_TMP
                MOV R0,R0_TMP
                MOV R5,R5_TMP

                CJNE R0,#7FH,WBW_BIT
WBW_TMP_BIT:    RL A
                DJNZ BIT_C,WBW_TMP_BYTE
                MOV BIT_C,#08H
                MOV R5,#RECV_SIZE
                MOV R0,#DATA_RECV
                CLR TMP_DATA
                MOV A,@R0
WBW_TMP_BYTE:   DJNZ R4,WBW_TMP_BIT
                JMP SET_WBW

WBW_BIT:        RL A
                DJNZ BIT_C,WBW_BYTE
                MOV BIT_C,#08H
                INC R0
                DEC R5
                MOV A,@R0
WBW_BYTE:       DJNZ R4,WBW_BIT

        ;---------- SET PIXEL -------------
SET_WBW:        MOV CODE_TMP,A
                MOV R4,NUM_PIXEL
        ;---------- HIGH BYTE -------------
                MOV A,A0
                MOV DPTR,#SETWBW_TAB
                RL A
                INC A
                MOVC A,@A+DPTR
                PUSH ACC
        ;----------- LOW BYTE -------------
                MOV A,A0
                RL A
                MOVC A,@A+DPTR
                PUSH ACC
                MOV A,A0
                ADD A,R4
                MOV A0,A
                RET
        ;----------- TABLE SET ------------
WBW0:           CJNE R4,#08H,CHK_WBW0
                MOV @R1,#11111111B
                INC R1
                MOV REF_OBYTE,REF_NBYTE
                MOV REF_NBYTE,@R1
                CJNE R1,#4EH,
                        GO_BW_ENDHOR0
                MOV R1,#REF_LINE  ; NEW LINE
                MOV DPS,#00H
                REPT 14
                MOV A,@R1
                MOVX @DPTR,A
                INC DPTR
                INC R1
                ENDM
                MOV R1,#REF_LINE
                MOV DPS,#01H
                MOV A0,#00H
                INC CODE_LINE_NUM
                MOV A,CODE_LINE_NUM
                CJNE A,#ROW,
                        GO_BW_ENDHOR0
                JMP END_PIC
GO_BW_ENDHOR0: JMP END_HOR
CHK_WBW0:       JC LESS_WBW0
                MOV @R1,#11111111B
                INC R1
                MOV REF_OBYTE,REF_NBYTE
                MOV REF_NBYTE,@R1
                CJNE R1,#4EH,
                        CONT_CHK_WBW0
                MOV R1,#REF_LINE  ; NEW LINE
                MOV DPS,#00H
                REPT 14
                MOV A,@R1
                MOVX @DPTR,A
                INC DPTR
                INC R1
                ENDM
                MOV R1,#REF_LINE
                MOV DPS,#01H
                MOV A0,#00H
                INC CODE_LINE_NUM
                MOV A,CODE_LINE_NUM
                CJNE A,#ROW,
                        GO_BW_ENDHOR0
                JMP END_PIC
CONT_CHK_WBW0: MOV A,R4
                CLR C
                SUBB A,#08H
                MOV R4,A
                JMP WBW0
LESS_WBW0:      MOV A,R4
                MOV DPTR,#LESS8_WBW
                RL A
                INC A
                MOVC A,@A+DPTR
                PUSH ACC
                MOV A,R4
                RL A
                MOVC A,@A+DPTR
                PUSH ACC
                RET
        ;============================
WBW1:           CJNE R4,#07H,CHK_WBW1
                MOV A,@R1
                ORL A,#01111111B
                MOV @R1,A
                INC R1
                MOV REF_OBYTE,REF_NBYTE
                MOV REF_NBYTE,@R1
                CJNE R1,#4EH,
                        GO_BW_ENDHOR1
                MOV R1,#REF_LINE  ; NEW LINE
                MOV DPS,#00H
```

```
                REPT 14
                MOV A,@R1
                MOVX @DPTR,A
                INC DPTR
                INC R1
                ENDM
                MOV R1,#REF_LINE
                MOV DPS,#01H
                MOV A0,#00H
                INC CODE_LINE_NUM
                MOV A,CODE_LINE_NUM
                CJNE A,#ROW,
                      GO_BW_ENDHOR1
                JMP END_PIC
GO_BW_ENDHOR1: JMP END_HOR
CHK_WBW1:       JC LESS_WBW1
                MOV A,@R1
                ORL A,#01111111B
                MOV @R1,A
                INC R1
                MOV REF_OBYTE,REF_NBYTE
                MOV REF_NBYTE,@R1
                CJNE R1,#4EH,
                      CONT_CHK_WBW
                MOV R1,#REF_LINE  ; NEW LINE
                MOV DPS,#00H
                REPT 14
                MOV A,@R1
                MOVX @DPTR,A
                INC DPTR
                INC R1
                ENDM
                MOV R1,#REF_LINE
                MOV DPS,#01H
                MOV A0,#00H
                INC CODE_LINE_NUM
                MOV A,CODE_LINE_NUM
                CJNE A,#ROW,
                      GO_BW_ENDHOR1
                JMP END_PIC
CONT_CHK_WBW: MOV A,R4
                CLR C
                SUBB A,#07H
                MOV R4,A
                JMP WBW0
LESS_WBW1:      MOV A,R4
                MOV DPTR,#LESS7_WBW
                RL A
                INC A
                MOVC A,@A+DPTR
                PUSH ACC
                MOV A,R4
                RL A
                MOVC A,@A+DPTR
                PUSH ACC
                RET
L7_WBW0:        JMP END_HOR
L7_WBW1:        MOV A,@R1
                ORL A,#01000000B
                MOV @R1,A
                JMP END_HOR
L7_WBW2:        MOV A,@R1
                ORL A,#01100000B
                MOV @R1,A
                JMP END_HOR
L7_WBW3:        MOV A,@R1
                ORL A,#01110000B
                MOV @R1,A
                JMP END_HOR
L7_WBW4:        MOV A,@R1
                ORL A,#01111000B
                MOV @R1,A
                JMP END_HOR
L7_WBW5:        MOV A,@R1
                ORL A,#01111100B
                MOV @R1,A
                JMP END_HOR
L7_WBW6:        MOV A,@R1
                ORL A,#01111110B
                MOV @R1,A
                JMP END_HOR

        ;============================
WBW2:           CJNE R4,#06H,CHK_WBW2
                MOV A,@R1
                ORL A,#00111111B
                MOV @R1,A
                INC R1
                MOV REF_OBYTE,REF_NBYTE
                MOV REF_NBYTE,@R1
                CJNE R1,#4EH,
                      GO_BW_ENDHOR2
                MOV R1,#REF_LINE  ; NEW LINE
                MOV DPS,#00H
                REPT 14
                MOV A,@R1
                MOVX @DPTR,A
                INC DPTR
                INC R1
                ENDM
                MOV R1,#REF_LINE
                MOV DPS,#01H
                MOV A0,#00H
                INC CODE_LINE_NUM
                MOV A,CODE_LINE_NUM
                CJNE A,#ROW,
                      GO_BW_ENDHOR2
                JMP END_PIC
GO_BW_ENDHOR2: JMP END_HOR
CHK_WBW2:       JC LESS_WBW2
                MOV A,@R1
                ORL A,#00111111B
                MOV @R1,A
                INC R1
                MOV REF_OBYTE,REF_NBYTE
                MOV REF_NBYTE,@R1
                CJNE R1,#4EH,
                      CONT_CHK_WBW2
                MOV R1,#REF_LINE  ; NEW LINE
                MOV DPS,#00H
                REPT 14
                MOV A,@R1
                MOVX @DPTR,A
                INC DPTR
                INC R1
                ENDM
                MOV R1,#REF_LINE
                MOV DPS,#01H
                MOV A0,#00H
                INC CODE_LINE_NUM
                MOV A,CODE_LINE_NUM
                CJNE A,#ROW,
                      GO_BW_ENDHOR2
                JMP END_PIC
CONT_CHK_WBW2: MOV A,R4
                CLR C
                SUBB A,#06H
                MOV R4,A
                JMP WBW0
LESS_WBW2:      MOV A,R4
                MOV DPTR,#LESS6_WBW
                RL A
                INC A
                MOVC A,@A+DPTR
                PUSH ACC
```

```
                MOV A,R4
                RL A
                MOVC A,@A+DPTR
                PUSH ACC
                RET

L6_WBW0:        JMP END_HOR
L6_WBW1:        MOV A,@R1
                ORL A,#00100000B
                MOV @R1,A
                JMP END_HOR
L6_WBW2:        MOV A,@R1
                ORL A,#00110000B
                MOV @R1,A
                JMP END_HOR
L6_WBW3:        MOV A,@R1
                ORL A,#00111000B
                MOV @R1,A
                JMP END_HOR
L6_WBW4:        MOV A,@R1
                ORL A,#00111100B
                MOV @R1,A
                JMP END_HOR
L6_WBW5:        MOV A,@R1
                ORL A,#00111110B
                MOV @R1,A
                JMP END_HOR
        ;============================
WBW3:           CJNE R4,#05H,CHK_WBW3
                MOV A,@R1
                ORL A,#00011111B
                MOV @R1,A
                INC R1
                MOV REF_OBYTE,REF_NBYTE
                MOV REF_NBYTE,@R1
                CJNE R1,#4EH,
                      GO_BW_ENDHOR3
                MOV R1,#REF_LINE  ; NEW LINE
                MOV DPS,#00H
                REPT 14
                MOV A,@R1
                MOVX @DPTR,A
                INC DPTR
                INC R1
                ENDM
                MOV R1,#REF_LINE
                MOV DPS,#01H
                MOV A0,#00H
                INC CODE_LINE_NUM
                MOV A,CODE_LINE_NUM
                CJNE A,#ROW,
                      GO_BW_ENDHOR3
                JMP END_PIC
GO_BW_ENDHOR3: JMP END_HOR
CHK_WBW3:       JC LESS_WBW3
                MOV A,@R1
                ORL A,#00011111B
                MOV @R1,A
                INC R1
                MOV REF_OBYTE,REF_NBYTE
                MOV REF_NBYTE,@R1
                CJNE R1,#4EH,
                      CONT_CHK_WBW3
                MOV R1,#REF_LINE  ; NEW LINE
                MOV DPS,#00H
                REPT 14
                MOV A,@R1
                MOVX @DPTR,A
                INC DPTR
                INC R1
                ENDM
                MOV R1,#REF_LINE
                MOV DPS,#01H
                MOV A0,#00H
                INC CODE_LINE_NUM
                MOV A,CODE_LINE_NUM
                CJNE A,#ROW,
                      GO_BW_ENDHOR3
                JMP END_PIC
CONT_CHK_WBW3: MOV A,R4
                CLR C
                SUBB A,#05H
                MOV R4,A
                JMP WBW0
LESS_WBW3:      MOV A,R4
                MOV DPTR,#LESS5_WBW
                RL A
                INC A
                MOVC A,@A+DPTR
                PUSH ACC
                MOV A,R4
                RL A
                MOVC A,@A+DPTR
                PUSH ACC
                RET
L5_WBW0:        JMP END_HOR
L5_WBW1:        MOV A,@R1
                ORL A,#00010000B
                MOV @R1,A
                JMP END_HOR
L5_WBW2:        MOV A,@R1
                ORL A,#00011000B
                MOV @R1,A
                JMP END_HOR
L5_WBW3:        MOV A,@R1
                ORL A,#00011100B
                MOV @R1,A
                JMP END_HOR
L5_WBW4:        MOV A,@R1
                ORL A,#00011110B
                MOV @R1,A
                JMP END_HOR
        ;============================
WBW4:           CJNE R4,#04H,CHK_WBW4
                MOV A,@R1
                ORL A,#00001111B
                MOV @R1,A
                INC R1
                MOV REF_OBYTE,REF_NBYTE
                MOV REF_NBYTE,@R1
                CJNE R1,#4EH,
                      GO_BW_ENDHOR4
                MOV R1,#REF_LINE  ; NEW LINE
                MOV DPS,#00H
                REPT 14
                MOV A,@R1
                MOVX @DPTR,A
                INC DPTR
                INC R1
                ENDM
                MOV R1,#REF_LINE
                MOV DPS,#01H
                MOV A0,#00H
                INC CODE_LINE_NUM
                MOV A,CODE_LINE_NUM
                CJNE A,#ROW,
                      GO_BW_ENDHOR4
                JMP END_PIC
GO_BW_ENDHOR4: JMP END_HOR
CHK_WBW4:       JC LESS_WBW4
                MOV A,@R1
                ORL A,#00001111B
                MOV @R1,A
                INC R1
                MOV REF_OBYTE,REF_NBYTE
                MOV REF_NBYTE,@R1
```

```
                CJNE R1,#4EH,
                      CONT_CHK_WBW4
                MOV R1,#REF_LINE  ; NEW LINE
                MOV DPS,#00H
                REPT 14
                MOV A,@R1
                MOVX @DPTR,A
                INC DPTR
                INC R1
                ENDM
                MOV R1,#REF_LINE
                MOV DPS,#01H
                MOV A0,#00H
                INC CODE_LINE_NUM
                MOV A,CODE_LINE_NUM
                CJNE A,#ROW,
                      GO_BW_ENDHOR4
                JMP END_PIC
CONT_CHK_WBW4: MOV A,R4
                CLR C
                SUBB A,#04H
                MOV R4,A
                JMP WBW0
LESS_WBW4:      DJNZ R4,N2_WBW4
                MOV A,@R1
                ORL A,#00001000B
                MOV @R1,A
                JMP END_HOR

N2_WBW4:        DJNZ R4,N3_WBW4
                MOV A,@R1
                ORL A,#00001100B
                MOV @R1,A
                JMP END_HOR
N3_WBW4:        MOV A,@R1
                ORL A,#00001110B
                MOV @R1,A
                JMP END_HOR
        ;============================
WBW5:           CJNE R4,#03H,CHK_WBW5

                MOV A,@R1
                ORL A,#00000111B
                MOV @R1,A
                INC R1
                MOV REF_OBYTE,REF_NBYTE
                MOV REF_NBYTE,@R1
                CJNE R1,#4EH,
                      GO_BW_ENDHOR5
                MOV R1,#REF_LINE  ; NEW LINE
                MOV DPS,#00H
                REPT 14
                MOV A,@R1
                MOVX @DPTR,A
                INC DPTR
                INC R1
                ENDM
                MOV R1,#REF_LINE
                MOV DPS,#01H
                MOV A0,#00H
                INC CODE_LINE_NUM
                MOV A,CODE_LINE_NUM
                CJNE A,#ROW,
                      GO_BW_ENDHOR5
                JMP END_PIC
GO_BW_ENDHOR5: JMP END_HOR
CHK_WBW5:       JC LESS_WBW5
                MOV A,@R1
                ORL A,#00000111B
                MOV @R1,A
                INC R1
                MOV REF_OBYTE,REF_NBYTE
                MOV REF_NBYTE,@R1
                CJNE R1,#4EH,
                      CONT_CHK_WBW5
                MOV R1,#REF_LINE  ; NEW LINE
                MOV DPS,#00H
                REPT 14
                MOV A,@R1
                MOVX @DPTR,A
                INC DPTR
                INC R1
                ENDM
                MOV R1,#REF_LINE
                MOV DPS,#01H
                MOV A0,#00H
                INC CODE_LINE_NUM
                MOV A,CODE_LINE_NUM
                CJNE A,#ROW,
                      GO_BW_ENDHOR5
                JMP END_PIC
CONT_CHK_WBW5: DEC R4
                DEC R4
                DEC R4
                JMP WBW0
LESS_WBW5:      DJNZ R4,N2_WBW5
                MOV A,@R1
                ORL A,#00000100B
                MOV @R1,A
                JMP END_HOR

N2_WBW5:        MOV A,@R1
                ORL A,#00000110B
                MOV @R1,A
                JMP END_HOR
        ;============================
WBW6:           CJNE R4,#02H,CHK_WBW6

                MOV A,@R1
                ORL A,#00000011B
                MOV @R1,A
                INC R1
                MOV REF_OBYTE,REF_NBYTE
                MOV REF_NBYTE,@R1
                CJNE R1,#4EH,
                      GO_BW_ENDHOR6
                MOV R1,#REF_LINE ; NEW LINE
                MOV DPS,#00H
                REPT 14
                MOV A,@R1
                MOVX @DPTR,A
                INC DPTR
                INC R1
                ENDM
                MOV R1,#REF_LINE
                MOV DPS,#01H
                MOV A0,#00H
                INC CODE_LINE_NUM
                MOV A,CODE_LINE_NUM
                CJNE A,#ROW,
                      GO_BW_ENDHOR6
                JMP END_PIC
GO_BW_ENDHOR6: JMP END_HOR
CHK_WBW6:       JC LESS_WBW6
                MOV A,@R1
                ORL A,#00000011B
                MOV @R1,A
                INC R1
                MOV REF_OBYTE,REF_NBYTE
                MOV REF_NBYTE,@R1
                CJNE R1,#4EH,
                      CONT_CHK_WBW6
                MOV R1,#REF_LINE  ; NEW LINE
                MOV DPS,#00H
                REPT 14
                MOV A,@R1
```

```
                MOVX @DPTR,A
                INC DPTR
                INC R1
                ENDM
                MOV R1,#REF_LINE
                MOV DPS,#01H
                MOV A0,#00H
                INC CODE_LINE_NUM
                MOV A,CODE_LINE_NUM
                CJNE A,#ROW,
                      GO_BW_ENDHOR6
                JMP END_PIC
CONT_CHK_WBW6: DEC R4
                DEC R4
                JMP WBW0
LESS_WBW6:      MOV A,@R1
                ORL A,#00000010B
                MOV @R1,A
                JMP END_HOR
        ;============================
WBW7:           CJNE R4,#01H,CHK_WBW7
                MOV A,@R1
                ORL A,#00000001B
                MOV @R1,A
                INC R1
                MOV REF_OBYTE,REF_NBYTE
                MOV REF_NBYTE,@R1
                CJNE R1,#4EH,
                      GO_BW_ENDHOR7
                MOV R1,#REF_LINE  ; NEW LINE
                MOV DPS,#00H
                REPT 14
                MOV A,@R1
                MOVX @DPTR,A
                INC DPTR
                INC R1
                ENDM
                MOV R1,#REF_LINE
                MOV DPS,#01H
                MOV A0,#00H
                INC CODE_LINE_NUM
                MOV A,CODE_LINE_NUM
                CJNE A,#ROW,
                      GO_BW_ENDHOR7
                JMP END_PIC
GO_BW_ENDHOR7: JMP END_HOR
CHK_WBW7:       MOV A,@R1
                ORL A,#00000001B
                MOV @R1,A
                INC R1
                MOV REF_OBYTE,REF_NBYTE
                MOV REF_NBYTE,@R1
                CJNE R1,#4EH,
                      CONT_CHK_WBW7
                MOV R1,#REF_LINE  ; NEW LINE
                MOV DPS,#00H
                REPT 14
                MOV A,@R1
                MOVX @DPTR,A
                INC DPTR
                INC R1
                ENDM
                MOV R1,#REF_LINE
                MOV DPS,#01H
                MOV A0,#00H
                INC CODE_LINE_NUM
                MOV A,CODE_LINE_NUM
                CJNE A,#ROW,
                      GO_BW_ENDHOR7

                JMP END_PIC
CONT_CHK_WBW7: DEC R4
                JMP WBW0
;---------- LESS THAN 8 BIT ----------
L8_WBW0:        JMP END_HOR

L8_WBW1:        MOV A,@R1
                ORL A,#10000000B
                MOV @R1,A
                JMP END_HOR

L8_WBW2:        MOV A,@R1
                ORL A,#11000000B
                MOV @R1,A
                JMP END_HOR

L8_WBW3:        MOV A,@R1
                ORL A,#11100000B
                MOV @R1,A
                JMP END_HOR

L8_WBW4:        MOV A,@R1
                ORL A,#11110000B
                MOV @R1,A
                JMP END_HOR

L8_WBW5:        MOV A,@R1
                ORL A,#11111000B
                MOV @R1,A
                JMP END_HOR

L8_WBW6:        MOV A,@R1
                ORL A,#11111100B
                MOV @R1,A
                JMP END_HOR

L8_WBW7:        MOV A,@R1
                ORL A,#11111110B
                MOV @R1,A

;---------- END HORIZONTAL ----------
END_HOR:        MOV A,CODE_TMP
                JMP DECOM_DATA
;========== PASS MODE ==========
GO_P1_PROC:     JMP P1_PROC

PASS_MODE:      MOV CODE_TMP,A
        ; ------ GET REF. LINE INDEX -----
                MOV A,A0
                MOV DPTR,#GET_R1
                MOVC A,@A+DPTR
                MOV R1,A ;GET REF. LINE INDEX
        ; ------ GET A00 - A07 -------
                MOV A,A0
                MOV DPTR,#GET_R2
                MOVC A,@A+DPTR
                MOV R2,A            ;GET A00-A07
                JB STATE,GO_P1_PROC
; ======= STATE 0(1) PROCESS =======
        ; ------ GET A00 - A07 TABLE -----
                CLR STATE
                MOV DPTR,#A0_WAY_0
                RL A
                INC A
                MOVC A,@A+DPTR
                PUSH ACC
                MOV A,R2
                RL A
                MOVC A,@A+DPTR
                MOV DPH1,A
                POP ACC
                MOV DPL1,A
        ; ------ GET MASK ------
MASK_P0:        MOV A,@R1
```

```
                MOV DECODING_BYTE,A
        ;GET DATA AT REF. LINE
                MOVC A,@A+DPTR
                MOV MASK_CODE,A
                MOV DPTR,#A0_VALUE_0
                MOVC A,@A+DPTR
                ADD A,A0
                MOV A0,A

                MOV A,MASK_CODE
                ORL A,@R1
                MOV @R1,A

                JNB MASK_CHK,GO_NP0
                INC R1
                MOV REF_OBYTE,REF_NBYTE
                MOV REF_NBYTE,@R1
                JB DECODING_BCHK,P0_MOV
                MOV A,@R1
                MOV DPTR,#A0_TAB0
                JMP MASK_P0

GO_ENDP0:       JMP END_P
;RETURN TO NEW COMPRESS DATA

P0_MOV:         MOV A,@R1
                CJNE A,#0FFH,LAST_P0
                MOV A,A0
                ADD A,#08H
                MOV A0,A
                INC R1
                MOV REF_OBYTE,REF_NBYTE
                MOV REF_NBYTE,@R1
                CJNE R1,#4EH,P0_MOV
                MOV R1,#REF_LINE  ; NEW LINE
                MOV DPS,#00H
                REPT 14
                MOV A,@R1
                MOVX @DPTR,A
                INC DPTR
                INC R1
                ENDM
                MOV R1,#REF_LINE
                MOV DPS,#01H
                MOV A0,#00H
                INC CODE_LINE_NUM
                MOV A,CODE_LINE_NUM
                CJNE A,#ROW,GO_ENDP0
                JMP END_PIC

LAST_P0:        MOV DPTR,#A0_V_END0
                MOVC A,@A+DPTR
                ADD A,A0
                MOV A0,A

GO_NP0:         MOV A,A0
                MOV DPTR,#GET_R2
                MOVC A,@A+DPTR
                MOV R2,A
        ; ------ GET P00 - P07 TABLE -----
                MOV DPTR,#P0_WAY
                RL A
                INC A
                MOVC A,@A+DPTR
                PUSH ACC
                MOV A,R2
                RL A
                MOVC A,@A+DPTR
                MOV DPH1,A
                POP ACC
                MOV DPL1,A

        ; ------ GET MASK ------

MASK_NP0:       MOV A,@R1
                MOVC A,@A+DPTR
                MOV MASK_CODE,A

                MOV DPTR,#A0_VALUE_0
                MOVC A,@A+DPTR
                ADD A,A0
                MOV A0,A

                MOV A,MASK_CODE
                ORL A,@R1
                MOV @R1,A

                JNB MASK_CHK,GO_ENDP01
                MOV DPTR,#P0_TAB0
                INC R1
                MOV REF_OBYTE,REF_NBYTE
                MOV REF_NBYTE,@R1
                CJNE R1,#4EH,MASK_NP0
                MOV R1,#REF_LINE  ;NEW LINE
                MOV DPS,#00H
                REPT 14
                MOV A,@R1
                MOVX @DPTR,A
                INC DPTR
                INC R1
                ENDM
                MOV R1,#REF_LINE
                MOV DPS,#01H
                MOV A0,#00H
                INC CODE_LINE_NUM
                MOV A,CODE_LINE_NUM
                CJNE A,#ROW,GO_ENDP01
                JMP END_PIC

GO_ENDP01:      JMP END_P
;RETURN TO NEW COMPRESS DATA

; ======= STATE 1(0) PROCESS =======

                ; ------ GET A00 - A07 TABLE -----
P1_PROC:        SETB STATE
                MOV DPTR,#A0_WAY_1
                RL A
                INC A
                MOVC A,@A+DPTR
                PUSH ACC
                MOV A,R2
                RL A
                MOVC A,@A+DPTR
                MOV DPH1,A
                POP ACC
                MOV DPL1,A
        ; ------ GET MASK ------
MASK_P1:        MOV A,@R1
                MOV DECODING_BYTE,A
        ;GET DATA AT REF. LINE
                MOVC A,@A+DPTR
                MOV MASK_CODE,A

                MOV DPTR,#A0_VALUE_1
                MOVC A,@A+DPTR
                ADD A,A0
                MOV A0,A

                MOV A,MASK_CODE
                ANL A,@R1
                MOV @R1,A

                JB MASK_CHK,GO_NP1
                INC R1
                MOV REF_OBYTE,REF_NBYTE
                MOV REF_NBYTE,@R1
```

```
                JNB DECODING_BCHK,P1_MOV
                MOV A,@R1
                MOV DPTR,#A0_TAB1
                JMP MASK_P1

P1_MOV:         MOV A,@R1
                CJNE A,#00H,LAST_P1
                MOV A,A0
                ADD A,#08H
                MOV A0,A
                INC R1
                MOV REF_OBYTE,REF_NBYTE
                MOV REF_NBYTE,@R1
                CJNE R1,#4EH,P1_MOV
                MOV R1,#REF_LINE  ; NEW LINE
                MOV DPS,#00H
                REPT 14
                MOV A,@R1
                MOVX @DPTR,A
                INC DPTR
                INC R1
                ENDM
                MOV R1,#REF_LINE
                MOV DPS,#01H
                MOV A0,#00H
                INC CODE_LINE_NUM
                MOV A,CODE_LINE_NUM
                CJNE A,#ROW,GO_ENDP1
                JMP END_PIC
GO_ENDP1:       JMP END_P
LAST_P1:        MOV DPTR,#A0_V_END1
                MOVC A,@A+DPTR
                ADD A,A0
                MOV A0,A

GO_NP1:         MOV A,A0
                MOV DPTR,#GET_R2
                MOVC A,@A+DPTR
                MOV R2,A
        ; ------ GET P10 - P17 TABLE -----
                MOV DPTR,#P1_WAY
                RL A
                INC A
                MOVC A,@A+DPTR
                PUSH ACC
                MOV A,R2
                RL A
                MOVC A,@A+DPTR
                MOV DPH1,A
                POP ACC
                MOV DPL1,A
        ; ------ GET MASK ------
MASK_NP1:       MOV A,@R1
                MOVC A,@A+DPTR
                MOV MASK_CODE,A

                MOV DPTR,#A0_VALUE_1
                MOVC A,@A+DPTR
                ADD A,A0
                MOV A0,A
                MOV A,MASK_CODE
                ANL A,@R1
                MOV @R1,A
                JB MASK_CHK,GO_ENDP1
                MOV DPTR,#P0_TAB1
                INC R1
                MOV REF_OBYTE,REF_NBYTE
                MOV REF_NBYTE,@R1
                CJNE R1,#4EH,MASK_NP1
                MOV R1,#REF_LINE  ;NEW LINE
                MOV DPS,#00H
                REPT 14
                MOV A,@R1
                MOVX @DPTR,A
                INC DPTR
                INC R1
                ENDM
                MOV R1,#REF_LINE
                MOV DPS,#01H
                MOV A0,#00H
                INC CODE_LINE_NUM
                MOV A,CODE_LINE_NUM
                CJNE A,#ROW,END_P
                JMP END_PIC
END_P:          MOV A,CODE_TMP
                JMP DECOM_DATA

;-------------- END 48 BYTES ROUTINE ----------------------
END_48_BYTES:   MOV R1_TMP,R1
                MOV R4_TMP,R4
;=========================================
;---------- SET DPTR FOR OSD SECTION -----------------
-
EXIT_PROC:      SETB EX1
                MOV DPS,#01H
                JB SWAP_BANK,BANK2
                JB FIELD_T,NEXT_EP1
                MOV DPTR,#PIC2_DISPLAY_E
                RETI
NEXT_EP1:       MOV DPTR,#PIC2_DISPLAY
                RETI
BANK2:          JB FIELD_T,NEXT_EP2
                MOV DPTR,#PIC1_DISPLAY_E
                RETI
NEXT_EP2:       MOV DPTR,#PIC1_DISPLAY
                RETI

;---------- CHANGE SWAP_BANK BIT ----------------------
END_PIC:        MOV S_BYTE,#00H
                MOV DPS,#01H
                CLR STATE
                CLR START_FRAME
                CLR KEEP_CONT
                MOV A0,#00H
                MOV BIT_C,#08H
                MOV R1,#00H
                MOV R1_TMP,#00H
                MOV R4_TMP,#00H
                MOV R0_START,#50H
                MOV CODE_LINE_NUM,#01H
                JB SWAP_BANK,CLR_SWAP
                SETB SWAP_BANK
                JMP EXIT_PROC
CLR_SWAP:       CLR SWAP_BANK
                JMP EXIT_PROC

;================ TABLE ZONE ==============

STATE_TAB:
DW
DECOM_DATA,V0_MODE,NEXT_C1,V1_MODE,NEXT
_C2,HOR_MODE,NEXT_C3,PASS_MODE
DW
NEXT_C4,V2_MODE,NEXT_C5,V3_MODE,NEXT_C6,
VL1_MODE,VR1_MODE,VL2_MODE
DW
VR2_MODE,VL3_MODE,VR3_MODE,R_WWB2,R_W
WB3,R_WWB4,R_WWB5,R_WWB6
DW
R_WWB7,R_WWB8,CODE_WWB,R_BWB42,R_BWB4
3,R_BWB44,CODE4_BWB,R0_BWB82
DW
R0_BWB83,R0_BWB84,R0_BWB85,R0_BWB86,R0_B
WB87,R0_BWB88,CODEBWB_00,R1_BWB32
```

```
DW
CODEBWB_01,R_BBW42,R_BBW43,R_BBW44,COD
E4_BBW,R0_BBW82,R0_BBW83,R0_BBW84
DW
R0_BBW85,R0_BBW86,R0_BBW87,R0_BBW88,COD
EBBW_00,R1_BBW32,CODEBBW_01,R_WBW2
DW
R_WBW3,R_WBW4,R_WBW5,R_WBW6,R_WBW7,R_
WBW8,CODE_WBW

;--------- R1 POSITION FOR DETECT B1 TABLE----------
GET_R1:
DB 40H,40H,40H,40H,40H,40H,40H,40H
DB 41H,41H,41H,41H,41H,41H,41H,41H
DB 42H,42H,42H,42H,42H,42H,42H,42H
DB 43H,43H,43H,43H,43H,43H,43H,43H
DB 44H,44H,44H,44H,44H,44H,44H,44H
DB 45H,45H,45H,45H,45H,45H,45H,45H
DB 46H,46H,46H,46H,46H,46H,46H,46H
DB 47H,47H,47H,47H,47H,47H,47H,47H
DB 48H,48H,48H,48H,48H,48H,48H,48H
DB 49H,49H,49H,49H,49H,49H,49H,49H
DB 4AH,4AH,4AH,4AH,4AH,4AH,4AH,4AH
DB 4BH,4BH,4BH,4BH,4BH,4BH,4BH,4BH
DB 4CH,4CH,4CH,4CH,4CH,4CH,4CH,4CH
DB 4DH,4DH,4DH,4DH,4DH,4DH,4DH,4DH
;--------- R2 POSITION FOR DETECT B1 TABLE----------
GET_R2:
DB 00,01,02,03,04,05,06,07
DB 00,01,02,03,04,05,06,07
DB 00,01,02,03,04,05,06,07
DB 00,01,02,03,04,05,06,07
DB 00,01,02,03,04,05,06,07
DB 00,01,02,03,04,05,06,07
DB 00,01,02,03,04,05,06,07
DB 00,01,02,03,04,05,06,07
DB 00,01,02,03,04,05,06,07
DB 00,01,02,03,04,05,06,07
DB 00,01,02,03,04,05,06,07
DB 00,01,02,03,04,05,06,07
DB 00,01,02,03,04,05,06,07
DB 00,01,02,03,04,05,06,07
;----------  STATE 0 A00 - A07 LABEL TABLE --------------
A0_WAY_0:
DW
A0_TAB0,A1_TAB0,A2_TAB0,A3_TAB0,A4_TAB0,A5_
TAB0,A6_TAB0,A7_TAB0

;---------------------- A00 - A07 TABLE SERIES --------------
A0_TAB0:
DB 0FFH,0FFH,0FEH,0FFH,0FCH,0FCH,0FEH,0FFH
DB 0F8H,0F8H,0F8H,0F8H,0FCH,0FCH,0FEH,0FFH
DB 0F0H,0F0H,0F0H,0F0H,0F0H,0F0H,0F0H,0F0H
DB 0F8H,0F8H,0F8H,0F8H,0FCH,0FCH,0FEH,0FFH
DB 0E0H,0E0H,0E0H,0E0H,0E0H,0E0H,0E0H,0E0H
DB 0E0H,0E0H,0E0H,0E0H,0E0H,0E0H,0E0H,0E0H
DB 0F0H,0F0H,0F0H,0F0H,0F0H,0F0H,0F0H,0F0H
DB 0F8H,0F8H,0F8H,0F8H,0FCH,0FCH,0FEH,0FFH
DB 0C0H,0C0H,0C0H,0C0H,0C0H,0C0H,0C0H,0C0H
DB 0C0H,0C0H,0C0H,0C0H,0C0H,0C0H,0C0H,0C0H
DB 0C0H,0C0H,0C0H,0C0H,0C0H,0C0H,0C0H,0C0H
DB 0C0H,0C0H,0C0H,0C0H,0C0H,0C0H,0C0H,0C0H
DB 0E0H,0E0H,0E0H,0E0H,0E0H,0E0H,0E0H,0E0H
DB 0E0H,0E0H,0E0H,0E0H,0E0H,0E0H,0E0H,0E0H
DB 0F0H,0F0H,0F0H,0F0H,0F0H,0F0H,0F0H,0F0H
DB 0F8H,0F8H,0F8H,0F8H,0FCH,0FCH,0FEH,0FFH
DB 80H,80H,80H,80H,80H,80H,80H,80H
DB 80H,80H,80H,80H,80H,80H,80H,80H
DB 80H,80H,80H,80H,80H,80H,80H,80H
DB 80H,80H,80H,80H,80H,80H,80H,80H
DB 80H,80H,80H,80H,80H,80H,80H,80H
DB 80H,80H,80H,80H,80H,80H,80H,80H
DB 80H,80H,80H,80H,80H,80H,80H,80H
DB 80H,80H,80H,80H,80H,80H,80H,80H
DB 0C0H,0C0H,0C0H,0C0H,0C0H,0C0H,0C0H,0C0H
DB 0C0H,0C0H,0C0H,0C0H,0C0H,0C0H,0C0H,0C0H
DB 0C0H,0C0H,0C0H,0C0H,0C0H,0C0H,0C0H,0C0H
DB 0C0H,0C0H,0C0H,0C0H,0C0H,0C0H,0C0H,0C0H
DB 0E0H,0E0H,0E0H,0E0H,0E0H,0E0H,0E0H,0E0H
DB 0E0H,0E0H,0E0H,0E0H,0E0H,0E0H,0E0H,0E0H
DB 0F0H,0F0H,0F0H,0F0H,0F0H,0F0H,0F0H,0F0H
DB 0F8H,0F8H,0F8H,0F8H,0FCH,0FCH,0FEH,0FFH

A1_TAB0:
DB 7FH,7FH,7EH,7FH,7CH,7CH,7EH,7FH
DB 78H,78H,78H,78H,7CH,7CH,7EH,7FH
DB 70H,70H,70H,70H,70H,70H,70H,70H
DB 78H,78H,78H,78H,7CH,7CH,7EH,7FH
DB 60H,60H,60H,60H,60H,60H,60H,60H
DB 60H,60H,60H,60H,60H,60H,60H,60H
DB 70H,70H,70H,70H,70H,70H,70H,70H
DB 78H,78H,78H,78H,7CH,7CH,7EH,7FH
DB 40H,40H,40H,40H,40H,40H,40H,40H
DB 40H,40H,40H,40H,40H,40H,40H,40H
DB 40H,40H,40H,40H,40H,40H,40H,40H
DB 40H,40H,40H,40H,40H,40H,40H,40H
DB 60H,60H,60H,60H,60H,60H,60H,60H
DB 60H,60H,60H,60H,60H,60H,60H,60H
DB 70H,70H,70H,70H,70H,70H,70H,70H
DB 78H,78H,78H,78H,7CH,7CH,7EH,7FH
DB 7FH,7FH,7EH,7FH,7CH,7CH,7EH,7FH
DB 78H,78H,78H,78H,7CH,7CH,7EH,7FH
DB 70H,70H,70H,70H,70H,70H,70H,70H
DB 78H,78H,78H,78H,7CH,7CH,7EH,7FH
DB 60H,60H,60H,60H,60H,60H,60H,60H
DB 60H,60H,60H,60H,60H,60H,60H,60H
DB 70H,70H,70H,70H,70H,70H,70H,70H
DB 78H,78H,78H,78H,7CH,7CH,7EH,7FH
DB 40H,40H,40H,40H,40H,40H,40H,40H
DB 40H,40H,40H,40H,40H,40H,40H,40H
DB 40H,40H,40H,40H,40H,40H,40H,40H
DB 40H,40H,40H,40H,40H,40H,40H,40H
DB 60H,60H,60H,60H,60H,60H,60H,60H
DB 60H,60H,60H,60H,60H,60H,60H,60H
DB 70H,70H,70H,70H,70H,70H,70H,70H
DB 78H,78H,78H,78H,7CH,7CH,7EH,7FH

A2_TAB0:
DB 3FH,3FH,3EH,3FH,3CH,3CH,3EH,3FH
DB 38H,38H,38H,38H,3CH,3CH,3EH,3FH
DB 30H,30H,30H,30H,30H,30H,30H,30H
DB 38H,38H,38H,38H,3CH,3CH,3EH,3FH
DB 20H,20H,20H,20H,20H,20H,20H,20H
DB 20H,20H,20H,20H,20H,20H,20H,20H
DB 30H,30H,30H,30H,30H,30H,30H,30H
DB 38H,38H,38H,38H,3CH,3CH,3EH,3FH
DB 3FH,3FH,3EH,3FH,3CH,3CH,3EH,3FH
DB 38H,38H,38H,38H,3CH,3CH,3EH,3FH
DB 30H,30H,30H,30H,30H,30H,30H,30H
DB 38H,38H,38H,38H,3CH,3CH,3EH,3FH
DB 20H,20H,20H,20H,20H,20H,20H,20H
DB 20H,20H,20H,20H,20H,20H,20H,20H
DB 30H,30H,30H,30H,30H,30H,30H,30H
DB 38H,38H,38H,38H,3CH,3CH,3EH,3FH
DB 3FH,3FH,3EH,3FH,3CH,3CH,3EH,3FH
DB 38H,38H,38H,38H,3CH,3CH,3EH,3FH
DB 30H,30H,30H,30H,30H,30H,30H,30H
DB 38H,38H,38H,38H,3CH,3CH,3EH,3FH
DB 20H,20H,20H,20H,20H,20H,20H,20H
DB 20H,20H,20H,20H,20H,20H,20H,20H
DB 30H,30H,30H,30H,30H,30H,30H,30H
DB 38H,38H,38H,38H,3CH,3CH,3EH,3FH
DB 3FH,3FH,3EH,3FH,3CH,3CH,3EH,3FH
DB 38H,38H,38H,38H,3CH,3CH,3EH,3FH
DB 30H,30H,30H,30H,30H,30H,30H,30H
DB 38H,38H,38H,38H,3CH,3CH,3EH,3FH
```

```
DB 20H,20H,20H,20H,20H,20H,20H,20H
DB 20H,20H,20H,20H,20H,20H,20H,20H
DB 30H,30H,30H,30H,30H,30H,30H,30H
DB 38H,38H,38H,38H,3CH,3CH,3EH,3FH

A3_TAB0:
DB 1FH,1FH,1EH,1FH,1CH,1CH,1EH,1FH
DB 18H,18H,18H,18H,1CH,1CH,1EH,1FH
DB 10H,10H,10H,10H,10H,10H,10H,10H
DB 18H,18H,18H,18H,1CH,1CH,1EH,1FH
DB 1FH,1FH,1EH,1FH,1CH,1CH,1EH,1FH
DB 18H,18H,18H,18H,1CH,1CH,1EH,1FH
DB 10H,10H,10H,10H,10H,10H,10H,10H
DB 18H,18H,18H,18H,1CH,1CH,1EH,1FH
DB 1FH,1FH,1EH,1FH,1CH,1CH,1EH,1FH
DB 18H,18H,18H,18H,1CH,1CH,1EH,1FH
DB 10H,10H,10H,10H,10H,10H,10H,10H
DB 18H,18H,18H,18H,1CH,1CH,1EH,1FH
DB 1FH,1FH,1EH,1FH,1CH,1CH,1EH,1FH
DB 18H,18H,18H,18H,1CH,1CH,1EH,1FH
DB 10H,10H,10H,10H,10H,10H,10H,10H
DB 18H,18H,18H,18H,1CH,1CH,1EH,1FH
DB 1FH,1FH,1EH,1FH,1CH,1CH,1EH,1FH
DB 18H,18H,18H,18H,1CH,1CH,1EH,1FH
DB 10H,10H,10H,10H,10H,10H,10H,10H
DB 18H,18H,18H,18H,1CH,1CH,1EH,1FH
DB 1FH,1FH,1EH,1FH,1CH,1CH,1EH,1FH
DB 18H,18H,18H,18H,1CH,1CH,1EH,1FH
DB 10H,10H,10H,10H,10H,10H,10H,10H
DB 18H,18H,18H,18H,1CH,1CH,1EH,1FH
DB 1FH,1FH,1EH,1FH,1CH,1CH,1EH,1FH
DB 18H,18H,18H,18H,1CH,1CH,1EH,1FH
DB 10H,10H,10H,10H,10H,10H,10H,10H
DB 18H,18H,18H,18H,1CH,1CH,1EH,1FH
DB 1FH,1FH,1EH,1FH,1CH,1CH,1EH,1FH
DB 18H,18H,18H,18H,1CH,1CH,1EH,1FH
DB 10H,10H,10H,10H,10H,10H,10H,10H
DB 18H,18H,18H,18H,1CH,1CH,1EH,1FH

A4_TAB0:
DB 0FH,0FH,0EH,0FH,0CH,0CH,0EH,0FH
DB 08H,08H,08H,08H,0CH,0CH,0EH,0FH
DB 0FH,0FH,0EH,0FH,0CH,0CH,0EH,0FH
DB 08H,08H,08H,08H,0CH,0CH,0EH,0FH
DB 0FH,0FH,0EH,0FH,0CH,0CH,0EH,0FH
DB 08H,08H,08H,08H,0CH,0CH,0EH,0FH
DB 0FH,0FH,0EH,0FH,0CH,0CH,0EH,0FH
DB 08H,08H,08H,08H,0CH,0CH,0EH,0FH
DB 0FH,0FH,0EH,0FH,0CH,0CH,0EH,0FH
DB 08H,08H,08H,08H,0CH,0CH,0EH,0FH
DB 0FH,0FH,0EH,0FH,0CH,0CH,0EH,0FH
DB 08H,08H,08H,08H,0CH,0CH,0EH,0FH
DB 0FH,0FH,0EH,0FH,0CH,0CH,0EH,0FH
DB 08H,08H,08H,08H,0CH,0CH,0EH,0FH
DB 0FH,0FH,0EH,0FH,0CH,0CH,0EH,0FH
DB 08H,08H,08H,08H,0CH,0CH,0EH,0FH
DB 0FH,0FH,0EH,0FH,0CH,0CH,0EH,0FH
DB 08H,08H,08H,08H,0CH,0CH,0EH,0FH
DB 0FH,0FH,0EH,0FH,0CH,0CH,0EH,0FH
DB 08H,08H,08H,08H,0CH,0CH,0EH,0FH
DB 0FH,0FH,0EH,0FH,0CH,0CH,0EH,0FH
DB 08H,08H,08H,08H,0CH,0CH,0EH,0FH
DB 0FH,0FH,0EH,0FH,0CH,0CH,0EH,0FH
DB 08H,08H,08H,08H,0CH,0CH,0EH,0FH
DB 0FH,0FH,0EH,0FH,0CH,0CH,0EH,0FH
DB 08H,08H,08H,08H,0CH,0CH,0EH,0FH
DB 0FH,0FH,0EH,0FH,0CH,0CH,0EH,0FH
DB 08H,08H,08H,08H,0CH,0CH,0EH,0FH
DB 0FH,0FH,0EH,0FH,0CH,0CH,0EH,0FH
DB 08H,08H,08H,08H,0CH,0CH,0EH,0FH
DB 0FH,0FH,0EH,0FH,0CH,0CH,0EH,0FH
DB 08H,08H,08H,08H,0CH,0CH,0EH,0FH

A5_TAB0:
DB 07H,07H,06H,07H,04H,04H,06H,07H
DB 07H,07H,06H,07H,04H,04H,06H,07H
DB 07H,07H,06H,07H,04H,04H,06H,07H
DB 07H,07H,06H,07H,04H,04H,06H,07H
DB 07H,07H,06H,07H,04H,04H,06H,07H
DB 07H,07H,06H,07H,04H,04H,06H,07H
DB 07H,07H,06H,07H,04H,04H,06H,07H
DB 07H,07H,06H,07H,04H,04H,06H,07H
DB 07H,07H,06H,07H,04H,04H,06H,07H
DB 07H,07H,06H,07H,04H,04H,06H,07H
DB 07H,07H,06H,07H,04H,04H,06H,07H
DB 07H,07H,06H,07H,04H,04H,06H,07H
DB 07H,07H,06H,07H,04H,04H,06H,07H
DB 07H,07H,06H,07H,04H,04H,06H,07H
DB 07H,07H,06H,07H,04H,04H,06H,07H
DB 07H,07H,06H,07H,04H,04H,06H,07H
DB 07H,07H,06H,07H,04H,04H,06H,07H
DB 07H,07H,06H,07H,04H,04H,06H,07H
DB 07H,07H,06H,07H,04H,04H,06H,07H
DB 07H,07H,06H,07H,04H,04H,06H,07H
DB 07H,07H,06H,07H,04H,04H,06H,07H
DB 07H,07H,06H,07H,04H,04H,06H,07H
DB 07H,07H,06H,07H,04H,04H,06H,07H
DB 07H,07H,06H,07H,04H,04H,06H,07H
DB 07H,07H,06H,07H,04H,04H,06H,07H
DB 07H,07H,06H,07H,04H,04H,06H,07H
DB 07H,07H,06H,07H,04H,04H,06H,07H
DB 07H,07H,06H,07H,04H,04H,06H,07H
DB 07H,07H,06H,07H,04H,04H,06H,07H
DB 07H,07H,06H,07H,04H,04H,06H,07H
DB 07H,07H,06H,07H,04H,04H,06H,07H
DB 07H,07H,06H,07H,04H,04H,06H,07H

A6_TAB0:
DB 03H,03H,02H,03H,03H,03H,02H,03H
DB 03H,03H,02H,03H,03H,03H,02H,03H
DB 03H,03H,02H,03H,03H,03H,02H,03H
DB 03H,03H,02H,03H,03H,03H,02H,03H
DB 03H,03H,02H,03H,03H,03H,02H,03H
DB 03H,03H,02H,03H,03H,03H,02H,03H
DB 03H,03H,02H,03H,03H,03H,02H,03H
DB 03H,03H,02H,03H,03H,03H,02H,03H
DB 03H,03H,02H,03H,03H,03H,02H,03H
DB 03H,03H,02H,03H,03H,03H,02H,03H
DB 03H,03H,02H,03H,03H,03H,02H,03H
DB 03H,03H,02H,03H,03H,03H,02H,03H
DB 03H,03H,02H,03H,03H,03H,02H,03H
DB 03H,03H,02H,03H,03H,03H,02H,03H
DB 03H,03H,02H,03H,03H,03H,02H,03H
DB 03H,03H,02H,03H,03H,03H,02H,03H
DB 03H,03H,02H,03H,03H,03H,02H,03H
DB 03H,03H,02H,03H,03H,03H,02H,03H
DB 03H,03H,02H,03H,03H,03H,02H,03H
DB 03H,03H,02H,03H,03H,03H,02H,03H
DB 03H,03H,02H,03H,03H,03H,02H,03H
DB 03H,03H,02H,03H,03H,03H,02H,03H
DB 03H,03H,02H,03H,03H,03H,02H,03H
DB 03H,03H,02H,03H,03H,03H,02H,03H
DB 03H,03H,02H,03H,03H,03H,02H,03H
DB 03H,03H,02H,03H,03H,03H,02H,03H
DB 03H,03H,02H,03H,03H,03H,02H,03H
DB 03H,03H,02H,03H,03H,03H,02H,03H
DB 03H,03H,02H,03H,03H,03H,02H,03H
DB 03H,03H,02H,03H,03H,03H,02H,03H
DB 03H,03H,02H,03H,03H,03H,02H,03H
DB 03H,03H,02H,03H,03H,03H,02H,03H

A7_TAB0:
DB 01H,01H,01H,01H,01H,01H,01H,01H
DB 01H,01H,01H,01H,01H,01H,01H,01H
DB 01H,01H,01H,01H,01H,01H,01H,01H
DB 01H,01H,01H,01H,01H,01H,01H,01H
```

```
DB 01H,01H,01H,01H,01H,01H,01H,01H
DB 01H,01H,01H,01H,01H,01H,01H,01H
DB 01H,01H,01H,01H,01H,01H,01H,01H
DB 01H,01H,01H,01H,01H,01H,01H,01H
DB 01H,01H,01H,01H,01H,01H,01H,01H
DB 01H,01H,01H,01H,01H,01H,01H,01H
DB 01H,01H,01H,01H,01H,01H,01H,01H
DB 01H,01H,01H,01H,01H,01H,01H,01H
DB 01H,01H,01H,01H,01H,01H,01H,01H
DB 01H,01H,01H,01H,01H,01H,01H,01H
DB 01H,01H,01H,01H,01H,01H,01H,01H
DB 01H,01H,01H,01H,01H,01H,01H,01H
DB 01H,01H,01H,01H,01H,01H,01H,01H
DB 01H,01H,01H,01H,01H,01H,01H,01H
DB 01H,01H,01H,01H,01H,01H,01H,01H
DB 01H,01H,01H,01H,01H,01H,01H,01H
DB 01H,01H,01H,01H,01H,01H,01H,01H
DB 01H,01H,01H,01H,01H,01H,01H,01H
DB 01H,01H,01H,01H,01H,01H,01H,01H
DB 01H,01H,01H,01H,01H,01H,01H,01H
DB 01H,01H,01H,01H,01H,01H,01H,01H
DB 01H,01H,01H,01H,01H,01H,01H,01H
DB 01H,01H,01H,01H,01H,01H,01H,01H
DB 01H,01H,01H,01H,01H,01H,01H,01H
DB 01H,01H,01H,01H,01H,01H,01H,01H
DB 01H,01H,01H,01H,01H,01H,01H,01H
DB 01H,01H,01H,01H,01H,01H,01H,01H
DB 01H,01H,01H,01H,01H,01H,01H,01H

;--------------- INC A0 VALUE TABLE ---------------------
A0_VALUE_0:
DB
00H,01H,01H,02H,01H,00H,02H,03H,01H,00H,00H,00
H,02H,00H,03H,04H
DB
01H,00H,00H,00H,00H,00H,00H,00H,02H,00H,00H,00
H,03H,00H,04H,05H
DB
01H,00H,00H,00H,00H,00H,00H,00H,00H,00H,00H,00
H,02H,00H,00H,00H
DB
02H,00H,00H,00H,00H,00H,00H,00H,03H,00H,00H,00
H,04H,00H,05H,06H
DB
01H,00H,00H,00H,00H,00H,00H,00H,00H,00H,00H,00
H,00H,00H,00H,00H
DB
00H,00H,00H,00H,00H,00H,00H,00H,00H,00H,00H,00
H,00H,00H,00H,00H
DB
02H,00H,00H,00H,00H,00H,00H,00H,00H,00H,00H,00
H,00H,00H,00H,00H
DB
03H,00H,00H,00H,00H,00H,00H,00H,04H,00H,00H,00
H,05H,00H,06H,07H
DB
01H,00H,00H,00H,00H,00H,00H,00H,00H,00H,00H,00
H,00H,00H,00H,00H
DB
00H,00H,00H,00H,00H,00H,00H,00H,00H,00H,00H,00
H,00H,00H,00H,00H
DB
00H,00H,00H,00H,00H,00H,00H,00H,00H,00H,00H,00
H,00H,00H,00H,00H
DB
00H,00H,00H,00H,00H,00H,00H,00H,00H,00H,00H,00
H,00H,00H,00H,00H
DB
02H,00H,00H,00H,00H,00H,00H,00H,00H,00H,00H,00
H,00H,00H,00H,00H
DB
00H,00H,00H,00H,00H,00H,00H,00H,00H,00H,00H,00
H,00H,00H,00H,00H
DB
03H,00H,00H,00H,00H,00H,00H,00H,00H,00H,00H,00
H,00H,00H,00H,00H
DB
04H,00H,00H,00H,00H,00H,00H,00H,05H,00H,00H,00
H,06H,00H,07H,08H

;-------------- STATE 0 A00 - A07 LABEL TABLE ----------
A0_WAY_1:
DW
A0_TAB1,A1_TAB1,A2_TAB1,A3_TAB1,A4_TAB1,A5_
TAB1,A6_TAB1,A7_TAB1

;---------------------- A00 - A07 TABLE SERIES --------------

A0_TAB1:
DB 00H,01H,03H,03H,07H,07H,07H,07H
DB 0FH,0FH,0FH,0FH,0FH,0FH,0FH,0FH
DB 1FH,1FH,1FH,1FH,1FH,1FH,1FH,1FH
DB 1FH,1FH,1FH,1FH,1FH,1FH,1FH,1FH
DB 3FH,3FH,3FH,3FH,3FH,3FH,3FH,3FH
DB 3FH,3FH,3FH,3FH,3FH,3FH,3FH,3FH
DB 3FH,3FH,3FH,3FH,3FH,3FH,3FH,3FH
DB 3FH,3FH,3FH,3FH,3FH,3FH,3FH,3FH
DB 7FH,7FH,7FH,7FH,7FH,7FH,7FH,7FH
DB 7FH,7FH,7FH,7FH,7FH,7FH,7FH,7FH
DB 7FH,7FH,7FH,7FH,7FH,7FH,7FH,7FH
DB 7FH,7FH,7FH,7FH,7FH,7FH,7FH,7FH
DB 7FH,7FH,7FH,7FH,7FH,7FH,7FH,7FH
DB 7FH,7FH,7FH,7FH,7FH,7FH,7FH,7FH
DB 7FH,7FH,7FH,7FH,7FH,7FH,7FH,7FH
DB 7FH,7FH,7FH,7FH,7FH,7FH,7FH,7FH
DB 00H,01H,03H,03H,07H,07H,07H,07H
DB 0FH,0FH,0FH,0FH,0FH,0FH,0FH,0FH
DB 1FH,1FH,1FH,1FH,1FH,1FH,1FH,1FH
DB 1FH,1FH,1FH,1FH,1FH,1FH,1FH,1FH
DB 3FH,3FH,3FH,3FH,3FH,3FH,3FH,3FH
DB 3FH,3FH,3FH,3FH,3FH,3FH,3FH,3FH
DB 3FH,3FH,3FH,3FH,3FH,3FH,3FH,3FH
DB 3FH,3FH,3FH,3FH,3FH,3FH,3FH,3FH
DB 00H,01H,03H,03H,07H,07H,07H,07H
DB 0FH,0FH,0FH,0FH,0FH,0FH,0FH,0FH
DB 1FH,1FH,1FH,1FH,1FH,1FH,1FH,1FH
DB 1FH,1FH,1FH,1FH,1FH,1FH,1FH,1FH
DB 00H,01H,03H,03H,07H,07H,07H,07H
DB 0FH,0FH,0FH,0FH,0FH,0FH,0FH,0FH
DB 00H,01H,03H,03H,07H,07H,07H,07H
DB 00H,01H,03H,03H,00H,01H,00H,00H

A1_TAB1:
DB 80H,81H,83H,83H,87H,87H,87H,87H
DB 8FH,8FH,8FH,8FH,8FH,8FH,8FH,8FH
DB 9FH,9FH,9FH,9FH,9FH,9FH,9FH,9FH
DB 9FH,9FH,9FH,9FH,9FH,9FH,9FH,9FH
DB 0BFH,0BFH,0BFH,0BFH,0BFH,0BFH,0BFH,0BFH
DB 0BFH,0BFH,0BFH,0BFH,0BFH,0BFH,0BFH,0BFH
DB 0BFH,0BFH,0BFH,0BFH,0BFH,0BFH,0BFH,0BFH
DB 0BFH,0BFH,0BFH,0BFH,0BFH,0BFH,0BFH,0BFH
DB 80H,81H,83H,83H,87H,87H,87H,87H
DB 8FH,8FH,8FH,8FH,8FH,8FH,8FH,8FH
DB 9FH,9FH,9FH,9FH,9FH,9FH,9FH,9FH
DB 9FH,9FH,9FH,9FH,9FH,9FH,9FH,9FH
DB 80H,81H,83H,83H,87H,87H,87H,87H
DB 8FH,8FH,8FH,8FH,8FH,8FH,8FH,8FH
DB 80H,81H,83H,83H,87H,87H,87H,87H
DB 80H,81H,83H,83H,80H,81H,80H,80H
DB 80H,81H,83H,83H,87H,87H,87H,87H
DB 8FH,8FH,8FH,8FH,8FH,8FH,8FH,8FH
DB 9FH,9FH,9FH,9FH,9FH,9FH,9FH,9FH
DB 9FH,9FH,9FH,9FH,9FH,9FH,9FH,9FH
DB 0BFH,0BFH,0BFH,0BFH,0BFH,0BFH,0BFH,0BFH
DB 0BFH,0BFH,0BFH,0BFH,0BFH,0BFH,0BFH,0BFH
DB 0BFH,0BFH,0BFH,0BFH,0BFH,0BFH,0BFH,0BFH
```

```
DB 0BFH,0BFH,0BFH,0BFH,0BFH,0BFH,0BFH,0BFH
DB 80H,81H,83H,83H,87H,87H,87H,87H
DB 8FH,8FH,8FH,8FH,8FH,8FH,8FH,8FH
DB 9FH,9FH,9FH,9FH,9FH,9FH,9FH,9FH
DB 9FH,9FH,9FH,9FH,9FH,9FH,9FH,9FH
DB 80H,81H,83H,83H,87H,87H,87H,87H
DB 8FH,8FH,8FH,8FH,8FH,8FH,8FH,8FH
DB 80H,81H,83H,83H,87H,87H,87H,87H
DB 80H,81H,83H,83H,80H,81H,80H,80H

A2_TAB1:
DB 0C0H,0C1H,0C3H,0C3H,0C7H,0C7H,0C7H,0C7H
DB 0CFH,0CFH,0CFH,0CFH,0CFH,0CFH,0CFH,0CFH
DB 0DFH,0DFH,0DFH,0DFH,0DFH,0DFH,0DFH,0DFH
DB 0DFH,0DFH,0DFH,0DFH,0DFH,0DFH,0DFH,0DFH
DB 0C0H,0C1H,0C3H,0C3H,0C7H,0C7H,0C7H,0C7H
DB 0CFH,0CFH,0CFH,0CFH,0CFH,0CFH,0CFH,0CFH
DB 0C0H,0C1H,0C3H,0C3H,0C7H,0C7H,0C7H,0C7H
DB 0C0H,0C1H,0C3H,0C3H,0C0H,0C1H,0C0H,0C0H
DB 0C0H,0C1H,0C3H,0C3H,0C7H,0C7H,0C7H,0C7H
DB 0CFH,0CFH,0CFH,0CFH,0CFH,0CFH,0CFH,0CFH
DB 0DFH,0DFH,0DFH,0DFH,0DFH,0DFH,0DFH,0DFH
DB 0DFH,0DFH,0DFH,0DFH,0DFH,0DFH,0DFH,0DFH
DB 0C0H,0C1H,0C3H,0C3H,0C7H,0C7H,0C7H,0C7H
DB 0CFH,0CFH,0CFH,0CFH,0CFH,0CFH,0CFH,0CFH
DB 0C0H,0C1H,0C3H,0C3H,0C7H,0C7H,0C7H,0C7H
DB 0C0H,0C1H,0C3H,0C3H,0C0H,0C1H,0C0H,0C0H
DB 0C0H,0C1H,0C3H,0C3H,0C7H,0C7H,0C7H,0C7H
DB 0CFH,0CFH,0CFH,0CFH,0CFH,0CFH,0CFH,0CFH
DB 0DFH,0DFH,0DFH,0DFH,0DFH,0DFH,0DFH,0DFH
DB 0DFH,0DFH,0DFH,0DFH,0DFH,0DFH,0DFH,0DFH
DB 0C0H,0C1H,0C3H,0C3H,0C7H,0C7H,0C7H,0C7H
DB 0CFH,0CFH,0CFH,0CFH,0CFH,0CFH,0CFH,0CFH
DB 0C0H,0C1H,0C3H,0C3H,0C7H,0C7H,0C7H,0C7H
DB 0C0H,0C1H,0C3H,0C3H,0C0H,0C1H,0C0H,0C0H
DB 0C0H,0C1H,0C3H,0C3H,0C7H,0C7H,0C7H,0C7H
DB 0CFH,0CFH,0CFH,0CFH,0CFH,0CFH,0CFH,0CFH
DB 0DFH,0DFH,0DFH,0DFH,0DFH,0DFH,0DFH,0DFH
DB 0DFH,0DFH,0DFH,0DFH,0DFH,0DFH,0DFH,0DFH
DB 0C0H,0C1H,0C3H,0C3H,0C7H,0C7H,0C7H,0C7H
DB 0CFH,0CFH,0CFH,0CFH,0CFH,0CFH,0CFH,0CFH
DB 0C0H,0C1H,0C3H,0C3H,0C7H,0C7H,0C7H,0C7H
DB 0C0H,0C1H,0C3H,0C3H,0C0H,0C1H,0C0H,0C0H

A3_TAB1:
DB 0E0H,0E1H,0E3H,0E3H,0E7H,0E7H,0E7H,0E7H
DB 0EFH,0EFH,0EFH,0EFH,0EFH,0EFH,0EFH,0EFH
DB 0E0H,0E1H,0E3H,0E3H,0E7H,0E7H,0E7H,0E7H
DB 0E0H,0E1H,0E3H,0E3H,0E0H,0E1H,0E0H,0E0H
DB 0E0H,0E1H,0E3H,0E3H,0E7H,0E7H,0E7H,0E7H
DB 0EFH,0EFH,0EFH,0EFH,0EFH,0EFH,0EFH,0EFH
DB 0E0H,0E1H,0E3H,0E3H,0E7H,0E7H,0E7H,0E7H
DB 0E0H,0E1H,0E3H,0E3H,0E0H,0E1H,0E0H,0E0H
DB 0E0H,0E1H,0E3H,0E3H,0E7H,0E7H,0E7H,0E7H
DB 0EFH,0EFH,0EFH,0EFH,0EFH,0EFH,0EFH,0EFH
DB 0E0H,0E1H,0E3H,0E3H,0E7H,0E7H,0E7H,0E7H
DB 0E0H,0E1H,0E3H,0E3H,0E0H,0E1H,0E0H,0E0H
DB 0E0H,0E1H,0E3H,0E3H,0E7H,0E7H,0E7H,0E7H
DB 0EFH,0EFH,0EFH,0EFH,0EFH,0EFH,0EFH,0EFH
DB 0E0H,0E1H,0E3H,0E3H,0E7H,0E7H,0E7H,0E7H
DB 0E0H,0E1H,0E3H,0E3H,0E0H,0E1H,0E0H,0E0H
DB 0E0H,0E1H,0E3H,0E3H,0E7H,0E7H,0E7H,0E7H
DB 0EFH,0EFH,0EFH,0EFH,0EFH,0EFH,0EFH,0EFH
DB 0E0H,0E1H,0E3H,0E3H,0E7H,0E7H,0E7H,0E7H
DB 0E0H,0E1H,0E3H,0E3H,0E0H,0E1H,0E0H,0E0H
DB 0E0H,0E1H,0E3H,0E3H,0E7H,0E7H,0E7H,0E7H
DB 0EFH,0EFH,0EFH,0EFH,0EFH,0EFH,0EFH,0EFH
DB 0E0H,0E1H,0E3H,0E3H,0E7H,0E7H,0E7H,0E7H
DB 0E0H,0E1H,0E3H,0E3H,0E0H,0E1H,0E0H,0E0H
DB 0E0H,0E1H,0E3H,0E3H,0E7H,0E7H,0E7H,0E7H
DB 0EFH,0EFH,0EFH,0EFH,0EFH,0EFH,0EFH,0EFH
DB 0E0H,0E1H,0E3H,0E3H,0E7H,0E7H,0E7H,0E7H
DB 0E0H,0E1H,0E3H,0E3H,0E0H,0E1H,0E0H,0E0H
DB 0E0H,0E1H,0E3H,0E3H,0E7H,0E7H,0E7H,0E7H
DB 0EFH,0EFH,0EFH,0EFH,0EFH,0EFH,0EFH,0EFH
DB 0E0H,0E1H,0E3H,0E3H,0E7H,0E7H,0E7H,0E7H
DB 0E0H,0E1H,0E3H,0E3H,0E0H,0E1H,0E0H,0E0H

A4_TAB1:
DB 0F0H,0F1H,0F3H,0F3H,0F7H,0F7H,0F7H,0F7H
DB 0F0H,0F1H,0F3H,0F3H,0F0H,0F1H,0F0H,0F0H
DB 0F0H,0F1H,0F3H,0F3H,0F7H,0F7H,0F7H,0F7H
DB 0F0H,0F1H,0F3H,0F3H,0F0H,0F1H,0F0H,0F0H
DB 0F0H,0F1H,0F3H,0F3H,0F7H,0F7H,0F7H,0F7H
DB 0F0H,0F1H,0F3H,0F3H,0F0H,0F1H,0F0H,0F0H
DB 0F0H,0F1H,0F3H,0F3H,0F7H,0F7H,0F7H,0F7H
DB 0F0H,0F1H,0F3H,0F3H,0F0H,0F1H,0F0H,0F0H
DB 0F0H,0F1H,0F3H,0F3H,0F7H,0F7H,0F7H,0F7H
DB 0F0H,0F1H,0F3H,0F3H,0F0H,0F1H,0F0H,0F0H
DB 0F0H,0F1H,0F3H,0F3H,0F7H,0F7H,0F7H,0F7H
DB 0F0H,0F1H,0F3H,0F3H,0F0H,0F1H,0F0H,0F0H
DB 0F0H,0F1H,0F3H,0F3H,0F7H,0F7H,0F7H,0F7H
DB 0F0H,0F1H,0F3H,0F3H,0F0H,0F1H,0F0H,0F0H
DB 0F0H,0F1H,0F3H,0F3H,0F7H,0F7H,0F7H,0F7H
DB 0F0H,0F1H,0F3H,0F3H,0F0H,0F1H,0F0H,0F0H
DB 0F0H,0F1H,0F3H,0F3H,0F7H,0F7H,0F7H,0F7H
DB 0F0H,0F1H,0F3H,0F3H,0F0H,0F1H,0F0H,0F0H
DB 0F0H,0F1H,0F3H,0F3H,0F7H,0F7H,0F7H,0F7H
DB 0F0H,0F1H,0F3H,0F3H,0F0H,0F1H,0F0H,0F0H
DB 0F0H,0F1H,0F3H,0F3H,0F7H,0F7H,0F7H,0F7H
DB 0F0H,0F1H,0F3H,0F3H,0F0H,0F1H,0F0H,0F0H
DB 0F0H,0F1H,0F3H,0F3H,0F7H,0F7H,0F7H,0F7H
DB 0F0H,0F1H,0F3H,0F3H,0F0H,0F1H,0F0H,0F0H
DB 0F0H,0F1H,0F3H,0F3H,0F7H,0F7H,0F7H,0F7H
DB 0F0H,0F1H,0F3H,0F3H,0F0H,0F1H,0F0H,0F0H
DB 0F0H,0F1H,0F3H,0F3H,0F7H,0F7H,0F7H,0F7H
DB 0F0H,0F1H,0F3H,0F3H,0F0H,0F1H,0F0H,0F0H
DB 0F0H,0F1H,0F3H,0F3H,0F7H,0F7H,0F7H,0F7H
DB 0F0H,0F1H,0F3H,0F3H,0F0H,0F1H,0F0H,0F0H
DB 0F0H,0F1H,0F3H,0F3H,0F7H,0F7H,0F7H,0F7H
DB 0F0H,0F1H,0F3H,0F3H,0F0H,0F1H,0F0H,0F0H

A5_TAB1:
DB 0F8H,0F9H,0FBH,0FBH,0F8H,0F9H,0F8H,0F8H
DB 0F8H,0F9H,0FBH,0FBH,0F8H,0F9H,0F8H,0F8H
DB 0F8H,0F9H,0FBH,0FBH,0F8H,0F9H,0F8H,0F8H
DB 0F8H,0F9H,0FBH,0FBH,0F8H,0F9H,0F8H,0F8H
DB 0F8H,0F9H,0FBH,0FBH,0F8H,0F9H,0F8H,0F8H
DB 0F8H,0F9H,0FBH,0FBH,0F8H,0F9H,0F8H,0F8H
DB 0F8H,0F9H,0FBH,0FBH,0F8H,0F9H,0F8H,0F8H
DB 0F8H,0F9H,0FBH,0FBH,0F8H,0F9H,0F8H,0F8H
DB 0F8H,0F9H,0FBH,0FBH,0F8H,0F9H,0F8H,0F8H
DB 0F8H,0F9H,0FBH,0FBH,0F8H,0F9H,0F8H,0F8H
DB 0F8H,0F9H,0FBH,0FBH,0F8H,0F9H,0F8H,0F8H
DB 0F8H,0F9H,0FBH,0FBH,0F8H,0F9H,0F8H,0F8H
DB 0F8H,0F9H,0FBH,0FBH,0F8H,0F9H,0F8H,0F8H
DB 0F8H,0F9H,0FBH,0FBH,0F8H,0F9H,0F8H,0F8H
DB 0F8H,0F9H,0FBH,0FBH,0F8H,0F9H,0F8H,0F8H
DB 0F8H,0F9H,0FBH,0FBH,0F8H,0F9H,0F8H,0F8H
DB 0F8H,0F9H,0FBH,0FBH,0F8H,0F9H,0F8H,0F8H
DB 0F8H,0F9H,0FBH,0FBH,0F8H,0F9H,0F8H,0F8H
DB 0F8H,0F9H,0FBH,0FBH,0F8H,0F9H,0F8H,0F8H
DB 0F8H,0F9H,0FBH,0FBH,0F8H,0F9H,0F8H,0F8H
DB 0F8H,0F9H,0FBH,0FBH,0F8H,0F9H,0F8H,0F8H
DB 0F8H,0F9H,0FBH,0FBH,0F8H,0F9H,0F8H,0F8H
DB 0F8H,0F9H,0FBH,0FBH,0F8H,0F9H,0F8H,0F8H
DB 0F8H,0F9H,0FBH,0FBH,0F8H,0F9H,0F8H,0F8H
DB 0F8H,0F9H,0FBH,0FBH,0F8H,0F9H,0F8H,0F8H
DB 0F8H,0F9H,0FBH,0FBH,0F8H,0F9H,0F8H,0F8H
DB 0F8H,0F9H,0FBH,0FBH,0F8H,0F9H,0F8H,0F8H
DB 0F8H,0F9H,0FBH,0FBH,0F8H,0F9H,0F8H,0F8H
DB 0F8H,0F9H,0FBH,0FBH,0F8H,0F9H,0F8H,0F8H
DB 0F8H,0F9H,0FBH,0FBH,0F8H,0F9H,0F8H,0F8H
DB 0F8H,0F9H,0FBH,0FBH,0F8H,0F9H,0F8H,0F8H
DB 0F8H,0F9H,0FBH,0FBH,0F8H,0F9H,0F8H,0F8H
A6_TAB1:
```

```
DB 0FCH,0FDH,0FCH,0FCH,0FCH,0FDH,0FCH,0FCH
DB 0FCH,0FDH,0FCH,0FCH,0FCH,0FDH,0FCH,0FCH
DB 0FCH,0FDH,0FCH,0FCH,0FCH,0FDH,0FCH,0FCH
DB 0FCH,0FDH,0FCH,0FCH,0FCH,0FDH,0FCH,0FCH
DB 0FCH,0FDH,0FCH,0FCH,0FCH,0FDH,0FCH,0FCH
DB 0FCH,0FDH,0FCH,0FCH,0FCH,0FDH,0FCH,0FCH
DB 0FCH,0FDH,0FCH,0FCH,0FCH,0FDH,0FCH,0FCH
DB 0FCH,0FDH,0FCH,0FCH,0FCH,0FDH,0FCH,0FCH
DB 0FCH,0FDH,0FCH,0FCH,0FCH,0FDH,0FCH,0FCH
DB 0FCH,0FDH,0FCH,0FCH,0FCH,0FDH,0FCH,0FCH
DB 0FCH,0FDH,0FCH,0FCH,0FCH,0FDH,0FCH,0FCH
DB 0FCH,0FDH,0FCH,0FCH,0FCH,0FDH,0FCH,0FCH
DB 0FCH,0FDH,0FCH,0FCH,0FCH,0FDH,0FCH,0FCH
DB 0FCH,0FDH,0FCH,0FCH,0FCH,0FDH,0FCH,0FCH
DB 0FCH,0FDH,0FCH,0FCH,0FCH,0FDH,0FCH,0FCH
DB 0FCH,0FDH,0FCH,0FCH,0FCH,0FDH,0FCH,0FCH
DB 0FCH,0FDH,0FCH,0FCH,0FCH,0FDH,0FCH,0FCH
DB 0FCH,0FDH,0FCH,0FCH,0FCH,0FDH,0FCH,0FCH
DB 0FCH,0FDH,0FCH,0FCH,0FCH,0FDH,0FCH,0FCH
DB 0FCH,0FDH,0FCH,0FCH,0FCH,0FDH,0FCH,0FCH
DB 0FCH,0FDH,0FCH,0FCH,0FCH,0FDH,0FCH,0FCH
DB 0FCH,0FDH,0FCH,0FCH,0FCH,0FDH,0FCH,0FCH
DB 0FCH,0FDH,0FCH,0FCH,0FCH,0FDH,0FCH,0FCH
DB 0FCH,0FDH,0FCH,0FCH,0FCH,0FDH,0FCH,0FCH
DB 0FCH,0FDH,0FCH,0FCH,0FCH,0FDH,0FCH,0FCH
DB 0FCH,0FDH,0FCH,0FCH,0FCH,0FDH,0FCH,0FCH
DB 0FCH,0FDH,0FCH,0FCH,0FCH,0FDH,0FCH,0FCH
DB 0FCH,0FDH,0FCH,0FCH,0FCH,0FDH,0FCH,0FCH
DB 0FCH,0FDH,0FCH,0FCH,0FCH,0FDH,0FCH,0FCH
DB 0FCH,0FDH,0FCH,0FCH,0FCH,0FDH,0FCH,0FCH
DB 0FCH,0FDH,0FCH,0FCH,0FCH,0FDH,0FCH,0FCH
DB 0FCH,0FDH,0FCH,0FCH,0FCH,0FDH,0FCH,0FCH

A7_TAB1:
DB 0FEH,0FEH,0FEH,0FEH,0FEH,0FEH,0FEH,0FEH
DB 0FEH,0FEH,0FEH,0FEH,0FEH,0FEH,0FEH,0FEH
DB 0FEH,0FEH,0FEH,0FEH,0FEH,0FEH,0FEH,0FEH
DB 0FEH,0FEH,0FEH,0FEH,0FEH,0FEH,0FEH,0FEH
DB 0FEH,0FEH,0FEH,0FEH,0FEH,0FEH,0FEH,0FEH
DB 0FEH,0FEH,0FEH,0FEH,0FEH,0FEH,0FEH,0FEH
DB 0FEH,0FEH,0FEH,0FEH,0FEH,0FEH,0FEH,0FEH
DB 0FEH,0FEH,0FEH,0FEH,0FEH,0FEH,0FEH,0FEH
DB 0FEH,0FEH,0FEH,0FEH,0FEH,0FEH,0FEH,0FEH
DB 0FEH,0FEH,0FEH,0FEH,0FEH,0FEH,0FEH,0FEH
DB 0FEH,0FEH,0FEH,0FEH,0FEH,0FEH,0FEH,0FEH
DB 0FEH,0FEH,0FEH,0FEH,0FEH,0FEH,0FEH,0FEH
DB 0FEH,0FEH,0FEH,0FEH,0FEH,0FEH,0FEH,0FEH
DB 0FEH,0FEH,0FEH,0FEH,0FEH,0FEH,0FEH,0FEH
DB 0FEH,0FEH,0FEH,0FEH,0FEH,0FEH,0FEH,0FEH
DB 0FEH,0FEH,0FEH,0FEH,0FEH,0FEH,0FEH,0FEH
DB 0FEH,0FEH,0FEH,0FEH,0FEH,0FEH,0FEH,0FEH
DB 0FEH,0FEH,0FEH,0FEH,0FEH,0FEH,0FEH,0FEH
DB 0FEH,0FEH,0FEH,0FEH,0FEH,0FEH,0FEH,0FEH
DB 0FEH,0FEH,0FEH,0FEH,0FEH,0FEH,0FEH,0FEH
DB 0FEH,0FEH,0FEH,0FEH,0FEH,0FEH,0FEH,0FEH
DB 0FEH,0FEH,0FEH,0FEH,0FEH,0FEH,0FEH,0FEH
DB 0FEH,0FEH,0FEH,0FEH,0FEH,0FEH,0FEH,0FEH
DB 0FEH,0FEH,0FEH,0FEH,0FEH,0FEH,0FEH,0FEH
DB 0FEH,0FEH,0FEH,0FEH,0FEH,0FEH,0FEH,0FEH
DB 0FEH,0FEH,0FEH,0FEH,0FEH,0FEH,0FEH,0FEH
DB 0FEH,0FEH,0FEH,0FEH,0FEH,0FEH,0FEH,0FEH
DB 0FEH,0FEH,0FEH,0FEH,0FEH,0FEH,0FEH,0FEH
DB 0FEH,0FEH,0FEH,0FEH,0FEH,0FEH,0FEH,0FEH
DB 0FEH,0FEH,0FEH,0FEH,0FEH,0FEH,0FEH,0FEH
DB 0FEH,0FEH,0FEH,0FEH,0FEH,0FEH,0FEH,0FEH
DB 0FEH,0FEH,0FEH,0FEH,0FEH,0FEH,0FEH,0FEH
;---------------------- INC A0 VALUE TABLE  -----------------
A0_VALUE_1:
DB
08H,07H,00H,06H,00H,00H,00H,05H,00H,00H,00H,00
H,00H,00H,00H,04H
DB
00H,00H,00H,00H,00H,00H,00H,00H,00H,00H,00H,00
H,00H,00H,00H,03H
DB
00H,00H,00H,00H,00H,00H,00H,00H,00H,00H,00H,00
H,00H,00H,00H,00H
DB
00H,00H,00H,00H,00H,00H,00H,00H,00H,00H,00H,00
H,00H,00H,00H,02H
DB
00H,00H,00H,00H,00H,00H,00H,00H,00H,00H,00H,00
H,00H,00H,00H,00H
DB
00H,00H,00H,00H,00H,00H,00H,00H,00H,00H,00H,00
H,00H,00H,00H,00H
DB
00H,00H,00H,00H,00H,00H,00H,00H,00H,00H,00H,00
H,00H,00H,00H,00H
DB
00H,00H,00H,00H,00H,00H,00H,00H,00H,00H,00H,00
H,00H,00H,00H,01H
DB
07H,06H,00H,05H,00H,00H,00H,04H,00H,00H,00H,00
H,00H,00H,00H,03H
DB
00H,00H,00H,00H,00H,00H,00H,00H,00H,00H,00H,00
H,00H,00H,00H,02H
DB
00H,00H,00H,00H,00H,00H,00H,00H,00H,00H,00H,00
H,00H,00H,00H,00H
DB
00H,00H,00H,00H,00H,00H,00H,00H,00H,00H,00H,00
H,00H,00H,00H,01H
DB
06H,05H,00H,04H,00H,00H,00H,03H,00H,00H,00H,00
H,00H,00H,00H,02H
DB
00H,00H,00H,00H,00H,00H,00H,00H,00H,00H,00H,00
H,00H,00H,00H,01H
DB
05H,04H,00H,03H,00H,00H,00H,02H,00H,00H,00H,00
H,00H,00H,00H,01H
DB
04H,03H,00H,02H,00H,00H,00H,01H,03H,02H,00H,01
H,02H,01H,01H,00H
;--  A0 VALUE AT THE END OF BLOCK CODE TABLE-

A0_V_END0:
DB
00H,00H,00H,00H,00H,00H,00H,00H,00H,00H,00H,00
H,00H,00H,00H,00H
DB
00H,00H,00H,00H,00H,00H,00H,00H,00H,00H,00H,00
H,00H,00H,00H,00H
DB
00H,00H,00H,00H,00H,00H,00H,00H,00H,00H,00H,00
H,00H,00H,00H,00H
DB
00H,00H,00H,00H,00H,00H,00H,00H,00H,00H,00H,00
H,00H,00H,00H,00H
DB
00H,00H,00H,00H,00H,00H,00H,00H,00H,00H,00H,00
H,00H,00H,00H,00H
DB
00H,00H,00H,00H,00H,00H,00H,00H,00H,00H,00H,00
H,00H,00H,00H,00H
DB
00H,00H,00H,00H,00H,00H,00H,00H,00H,00H,00H,00
H,00H,00H,00H,00H
DB
00H,00H,00H,00H,00H,00H,00H,00H,00H,00H,00H,00
H,00H,00H,00H,00H
```

```
DB
01H,01H,01H,01H,01H,01H,01H,01H,01H,01H,01H,01
H,01H,01H,01H,01H
DB
01H,01H,01H,01H,01H,01H,01H,01H,01H,01H,01H,01
H,01H,01H,01H,01H
DB
01H,01H,01H,01H,01H,01H,01H,01H,01H,01H,01H,01
H,01H,01H,01H,01H
DB
01H,01H,01H,01H,01H,01H,01H,01H,01H,01H,01H,01
H,01H,01H,01H,01H
DB
02H,02H,02H,02H,02H,02H,02H,02H,02H,02H,02H,02
H,02H,02H,02H,02H
DB
02H,02H,02H,02H,02H,02H,02H,02H,02H,02H,02H,02
H,02H,02H,02H,02H
DB
03H,03H,03H,03H,03H,03H,03H,03H,03H,03H,03H,03
H,03H,03H,03H,03H
DB
04H,04H,04H,04H,04H,04H,04H,04H,05H,05H,05H,05
H,06H,06H,07H,08H

A0_V_END1:
DB
08H,07H,06H,06H,05H,05H,05H,05H,04H,04H,04H,04
H,04H,04H,04H,04H
DB
03H,03H,03H,03H,03H,03H,03H,03H,03H,03H,03H,03
H,03H,03H,03H,03H
DB
02H,02H,02H,02H,02H,02H,02H,02H,02H,02H,02H,02
H,02H,02H,02H,02H
DB
02H,02H,02H,02H,02H,02H,02H,02H,02H,02H,02H,02
H,02H,02H,02H,02H
DB
01H,01H,01H,01H,01H,01H,01H,01H,01H,01H,01H,01
H,01H,01H,01H,01H
DB
01H,01H,01H,01H,01H,01H,01H,01H,01H,01H,01H,01
H,01H,01H,01H,01H
DB
01H,01H,01H,01H,01H,01H,01H,01H,01H,01H,01H,01
H,01H,01H,01H,01H
DB
01H,01H,01H,01H,01H,01H,01H,01H,01H,01H,01H,01
H,01H,01H,01H,01H
DB
00H,00H,00H,00H,00H,00H,00H,00H,00H,00H,00H,00
H,00H,00H,00H,00H
DB
00H,00H,00H,00H,00H,00H,00H,00H,00H,00H,00H,00
H,00H,00H,00H,00H
DB
00H,00H,00H,00H,00H,00H,00H,00H,00H,00H,00H,00
H,00H,00H,00H,00H
DB
00H,00H,00H,00H,00H,00H,00H,00H,00H,00H,00H,00
H,00H,00H,00H,00H
DB
00H,00H,00H,00H,00H,00H,00H,00H,00H,00H,00H,00
H,00H,00H,00H,00H
DB
00H,00H,00H,00H,00H,00H,00H,00H,00H,00H,00H,00
H,00H,00H,00H,00H
DB
00H,00H,00H,00H,00H,00H,00H,00H,00H,00H,00H,00
H,00H,00H,00H,00H
DB
00H,00H,00H,00H,00H,00H,00H,00H,00H,00H,00H,00
H,00H,00H,00H,00H

;--------------------- MASK FOR VL1 - VL3 ---------------------
VL1_MASK: DW
VL1_MASK7,VL1_MASK0,VL1_MASK1,VL1_MASK2,V
L1_MASK3,VL1_MASK4,VL1_MASK5,VL1_MASK6

VL2_MASK: DW
VL2_MASK7,VL2_MASK0,VL2_MASK1,VL2_MASK2,V
L2_MASK3,VL2_MASK4,VL2_MASK5,VL2_MASK6

VL3_MASK: DW
VL3_MASK7,VL3_MASK0,VL3_MASK1,VL3_MASK2,V
L3_MASK3,VL3_MASK4,VL3_MASK5,VL3_MASK6

;--------------------- MASK FOR VR10 - VR31 ----------------
VR10_MASK_TAB:
DB
80H,01H,03H,03H,06H,06H,07H,07H,0CH,0CH,0CH,0
CH,0EH,0EH,0FH,0FH
DB
18H,18H,18H,18H,18H,18H,18H,18H,1CH,1CH,1CH,1
CH,1EH,1EH,1FH,1FH
DB
30H,30H,30H,30H,30H,30H,30H,30H,30H,30H,30H,30
H,38H,38H,38H,38H
DB
38H,38H,38H,38H,38H,38H,38H,38H,3CH,3CH,3CH,3
CH,3EH,3EH,3FH,3FH
DB
60H,60H,60H,60H,60H,60H,60H,60H,60H,60H,60H,60
H,60H,60H,60H,60H
DB
60H,60H,60H,60H,60H,60H,60H,60H,60H,60H,60H,60
H,60H,60H,60H,60H
DB
70H,70H,70H,70H,70H,70H,70H,70H,70H,70H,70H,70
H,70H,70H,70H,70H
DB
78H,78H,78H,78H,78H,78H,78H,78H,7CH,7CH,7CH,7
CH,7EH,00H,7FH,7FH
DB
0C0H,0C0H,0C0H,0C0H,0C0H,0C0H,0C0H,0C0H,0C0
H,0C0H,0C0H,0C0H,0C0H,0C0H,0C0H,0C0H
DB
0C0H,0C0H,0C0H,0C0H,0C0H,0C0H,0C0H,0C0H,0C0
H,0C0H,0C0H,0C0H,0C0H,0C0H,0C0H,0C0H
DB
0C0H,0C0H,0C0H,0C0H,0C0H,0C0H,0C0H,0C0H,0C0
H,0C0H,0C0H,0C0H,0C0H,0C0H,0C0H,0C0H
DB
0C0H,0C0H,0C0H,0C0H,0C0H,0C0H,0C0H,0C0H,0C0
H,0C0H,0C0H,0C0H,0C0H,0C0H,0C0H,0C0H
DB
0E0H,0E0H,0E0H,0E0H,0E0H,0E0H,0E0H,0E0H,0E0H
,0E0H,0E0H,0E0H,0E0H,0E0H,0E0H,0E0H
DB
0E0H,0E0H,0E0H,0E0H,0E0H,0E0H,0E0H,0E0H,0E0H
,0E0H,0E0H,0E0H,0E0H,0E0H,0E0H,0E0H
DB
0F0H,0F0H,0F0H,0F0H,0F0H,0F0H,0F0H,0F0H,0F0H,
0F0H,0F0H,0F0H,0F0H,0F0H,0F0H,0F0H
DB
0F8H,0F8H,0F8H,0F8H,0F8H,0F8H,0F8H,0F8H,0FCH,
0FCH,0FCH,0FCH,0FEH,0FEH,0FFH,0FFH

VR11_MASK_TAB:
DB
00H,00H,01H,01H,03H,03H,03H,03H,07H,07H,07H,07
H,07H,07H,07H,07H
```

```
DB
0FH,0FH,0FH,0FH,0FH,0FH,0FH,0FH,0FH,0FH,0FH,0
FH,0FH,0FH,0FH,0FH
DB
1FH,1FH,1FH,1FH,1FH,1FH,1FH,1FH,1FH,1FH,1FH,1
FH,1FH,1FH,1FH,1FH
DB
1FH,1FH,1FH,1FH,1FH,1FH,1FH,1FH,1FH,1FH,1FH,1
FH,1FH,1FH,1FH,1FH
DB
3FH,3FH,3FH,3FH,3FH,3FH,3FH,3FH,3FH,3FH,3FH,3
FH,3FH,3FH,3FH,3FH
DB
3FH,3FH,3FH,3FH,3FH,3FH,3FH,3FH,3FH,3FH,3FH,3
FH,3FH,3FH,3FH,3FH
DB
3FH,3FH,3FH,3FH,3FH,3FH,3FH,3FH,3FH,3FH,3FH,3
FH,3FH,3FH,3FH,3FH
DB
3FH,3FH,3FH,3FH,3FH,3FH,3FH,3FH,3FH,3FH,3FH,3
FH,3FH,3FH,3FH,3FH
DB
80H,80H,81H,81H,83H,83H,83H,83H,87H,87H,87H,87
H,87H,87H,87H,87H
DB
8FH,8FH,8FH,8FH,8FH,8FH,8FH,8FH,8FH,8FH,8FH,8
FH,8FH,8FH,8FH,8FH
DB
9FH,9FH,9FH,9FH,9FH,9FH,9FH,9FH,9FH,9FH,9FH,9
FH,9FH,9FH,9FH,9FH
DB
9FH,9FH,9FH,9FH,9FH,9FH,9FH,9FH,9FH,9FH,9FH,9
FH,9FH,9FH,9FH,9FH
DB
0C0H,0C0H,0C1H,0C1H,0C3H,0C3H,0C3H,0C3H,0C7
H,0C7H,0C7H,0C7H,0C7H,0C7H,0C7H,0C7H
DB
0CFH,0CFH,0CFH,0CFH,0CFH,0CFH,0CFH,0CFH,0C
FH,0CFH,0CFH,0CFH,0CFH,0CFH,0CFH,0CFH
DB
0E0H,0E0H,0E1H,0E1H,0E3H,0E3H,0E3H,0E3H,0E7H
,0E7H,0E7H,0E7H,0E7H,0E7H,0E7H,0E7H
DB
0F0H,0F0H,0F1H,0F1H,0F1H,0F1H,0F1H,0F1H,0F8H,
0F8H,0F9H,0F9H,0FCH,0FCH,0FFH,7FH

VR20_MASK_TAB:
DB
0C0H,01H,03H,03H,07H,07H,07H,07H,0EH,0EH,0EH,
0EH,0FH,0FH,0FH,0FH
DB
1CH,1CH,1CH,1CH,1CH,1CH,1CH,1CH,1EH,1EH,1EH
,1EH,1FH,1FH,1FH,1FH
DB
38H,38H,38H,38H,38H,38H,38H,38H,38H,38H,38H,38
H,3CH,3CH,3CH,3CH
DB
3CH,3CH,3CH,3CH,3CH,3CH,3CH,3CH,3EH,3EH,3EH
,3EH,3FH,3FH,3FH,3FH
DB
70H,70H,70H,70H,70H,70H,70H,70H,70H,70H,70H,70
H,70H,70H,70H,70H
DB
70H,70H,70H,70H,70H,70H,70H,70H,70H,70H,70H,70
H,70H,70H,70H,70H
DB
78H,78H,78H,78H,78H,78H,78H,78H,78H,78H,78H,78
H,78H,78H,78H,78H
DB
7CH,7CH,7CH,7CH,7CH,7CH,7CH,7CH,7EH,7EH,7EH
,7EH,7FH,7FH,7FH,7FH
DB
0E0H,0E0H,0E0H,0E0H,0E0H,0E0H,0E0H,0E0H,0E0H
,0E0H,0E0H,0E0H,0E0H,0E0H,0E0H,0E0H
DB
0E0H,0E0H,0E0H,0E0H,0E0H,0E0H,0E0H,0E0H,0E0H
,0E0H,0E0H,0E0H,0E0H,0E0H,0E0H,0E0H
DB
0E0H,0E0H,0E0H,0E0H,0E0H,0E0H,0E0H,0E0H,0E0H
,0E0H,0E0H,0E0H,0E0H,0E0H,0E0H,0E0H
DB
0E0H,0E0H,0E0H,0E0H,0E0H,0E0H,0E0H,0E0H,0E0H
,0E0H,0E0H,0E0H,0E0H,0E0H,0E0H,0E0H
DB
0F0H,0F0H,0F0H,0F0H,0F0H,0F0H,0F0H,0F0H,0F0H,
0F0H,0F0H,0F0H,0F0H,0F0H,0F0H,0F0H
DB
0F0H,0F0H,0F0H,0F0H,0F0H,0F0H,0F0H,0F0H,0F0H,
0F0H,0F0H,0F0H,0F0H,0F0H,0F0H,0F0H
DB
0F8H,0F8H,0F8H,0F8H,0F8H,0F8H,0F8H,0F8H,0F8H,
0F8H,0F8H,0F8H,0F8H,0F8H,0F8H,0F8H
DB
0FCH,0FCH,0FCH,0FCH,0FCH,0FCH,0FCH,0FCH,0F
EH,0FEH,0FEH,0FEH,0FFH,0FEH,0FFH,0FFH

VR21_MASK_TAB:
DB
00H,00H,00H,00H,01H,01H,01H,01H,03H,03H,03H,03
H,03H,03H,03H,03H
DB
07H,07H,07H,07H,07H,07H,07H,07H,07H,07H,07H,07
H,07H,07H,07H,07H
DB
0FH,0FH,0FH,0FH,0FH,0FH,0FH,0FH,0FH,0FH,0FH,0
FH,0FH,0FH,0FH,0FH
DB
0FH,0FH,0FH,0FH,0FH,0FH,0FH,0FH,0FH,0FH,0FH,0
FH,0FH,0FH,0FH,0FH
DB
1FH,1FH,1FH,1FH,1FH,1FH,1FH,1FH,1FH,1FH,1FH,1
FH,1FH,1FH,1FH,1FH
DB
1FH,1FH,1FH,1FH,1FH,1FH,1FH,1FH,1FH,1FH,1FH,1
FH,1FH,1FH,1FH,1FH
DB
1FH,1FH,1FH,1FH,1FH,1FH,1FH,1FH,1FH,1FH,1FH,1
FH,1FH,1FH,1FH,1FH
DB
1FH,1FH,1FH,1FH,1FH,1FH,1FH,1FH,1FH,1FH,1FH,1
FH,1FH,1FH,1FH,1FH
DB
80H,80H,80H,80H,81H,81H,81H,81H,83H,83H,83H,83
H,83H,83H,83H,83H
DB
87H,87H,87H,87H,87H,87H,87H,87H,87H,87H,87H,87
H,87H,87H,87H,87H
DB
8FH,8FH,8FH,8FH,8FH,8FH,8FH,8FH,8FH,8FH,8FH,8
FH,8FH,8FH,8FH,8FH
DB
8FH,8FH,8FH,8FH,8FH,8FH,8FH,8FH,8FH,8FH,8FH,8
FH,8FH,8FH,8FH,8FH
DB
0C0H,0C0H,0C0H,0C0H,0C1H,0C1H,0C1H,0C1H,0C3
H,0C3H,0C3H,0C3H,0C3H,0C3H,0C3H,0C3H
DB
0C7H,0C7H,0C7H,0C7H,0C7H,0C7H,0C7H,0C7H,0C7
H,0C7H,0C7H,0C7H,0C7H,0C7H,0C7H,0C7H
DB
0E0H,0E0H,0E0H,0E0H,0E1H,0E1H,0E1H,0E1H,0E3H
,0E3H,0E3H,0E3H,0E3H,0E3H,0E3H,0E3H
```

```
DB
0F0H,0F0H,0F0H,0F0H,0F1H,0F1H,0F1H,0F1H,0F8H,
0F8H,0F8H,0F8H,0FCH,0FCH,0FEH,3FH

VR30_MASK_TAB:
DB
0E0H,03H,03H,03H,07H,07H,07H,07H,0FH,0FH,0FH,0
FH,0FH,0FH,0FH,0FH
DB
1EH,1EH,1EH,1EH,1EH,1EH,1EH,1EH,1FH,1FH,1FH,
1FH,1FH,1FH,1FH,1FH
DB
3CH,3CH,3CH,3CH,3CH,3CH,3CH,3CH,3CH,3CH,3CH
,3CH,3EH,3EH,3EH,3EH
DB
3EH,3EH,3EH,3EH,3EH,3EH,3EH,3EH,3FH,3FH,3FH,
3FH,3FH,3FH,3FH,3FH
DB
78H,78H,78H,78H,78H,78H,78H,78H,78H,78H,78H,78
H,78H,78H,78H,78H
DB
78H,78H,78H,78H,78H,78H,78H,78H,78H,78H,78H,78
H,78H,78H,78H,78H
DB
7CH,7CH,7CH,7CH,7CH,7CH,7CH,7CH,7CH,7CH,7CH
,7CH,7CH,7CH,7CH,7CH
DB
7EH,7EH,7EH,7EH,7EH,7EH,7EH,7EH,7FH,7FH,7FH,
7FH,7FH,7FH,7FH,7FH
DB
0F0H,0F0H,0F0H,0F0H,0F0H,0F0H,0F0H,0F0H,0F0H,
0F0H,0F0H,0F0H,0F0H,0F0H,0F0H,0F0H
DB
0F0H,0F0H,0F0H,0F0H,0F0H,0F0H,0F0H,0F0H,0F0H,
0F0H,0F0H,0F0H,0F0H,0F0H,0F0H,0F0H
DB
0F0H,0F0H,0F0H,0F0H,0F0H,0F0H,0F0H,0F0H,0F0H,
0F0H,0F0H,0F0H,0F0H,0F0H,0F0H,0F0H
DB
0F0H,0F0H,0F0H,0F0H,0F0H,0F0H,0F0H,0F0H,0F0H,
0F0H,0F0H,0F0H,0F0H,0F0H,0F0H,0F0H
DB
0F8H,0F8H,0F8H,0F8H,0F8H,0F8H,0F8H,0F8H,0F8H,
0F8H,0F8H,0F8H,0F8H,0F8H,0F8H,0F8H
DB
0F8H,0F8H,0F8H,0F8H,0F8H,0F8H,0F8H,0F8H,0F8H,
0F8H,0F8H,0F8H,0F8H,0F8H,0F8H,0F8H
DB
0FCH,0FCH,0FCH,0FCH,0FCH,0FCH,0FCH,0FCH,0F
CH,0FCH,0FCH,0FCH,0FCH,0FCH,0FCH,0FCH
DB
0FEH,0FEH,0FEH,0FEH,0FEH,0FEH,0FEH,0FEH,0FF
H,0FFH,0FFH,0FFH,0FFH,0FFH,0FFH,0FFH

VR31_MASK_TAB:
DB
00H,00H,00H,00H,00H,00H,00H,00H,01H,01H,01H,01
H,01H,01H,01H,01H
DB
03H,03H,03H,03H,03H,03H,03H,03H,03H,03H,03H,03
H,03H,03H,03H,03H
DB
07H,07H,07H,07H,07H,07H,07H,07H,07H,07H,07H,07
H,07H,07H,07H,07H
DB
07H,07H,07H,07H,07H,07H,07H,07H,07H,07H,07H,07
H,07H,07H,07H,07H
DB
0FH,0FH,0FH,0FH,0FH,0FH,0FH,0FH,0FH,0FH,0FH,0
FH,0FH,0FH,0FH,0FH
DB
0FH,0FH,0FH,0FH,0FH,0FH,0FH,0FH,0FH,0FH,0FH,0
FH,0FH,0FH,0FH,0FH
DB
0FH,0FH,0FH,0FH,0FH,0FH,0FH,0FH,0FH,0FH,0FH,0
FH,0FH,0FH,0FH,0FH
DB
0FH,0FH,0FH,0FH,0FH,0FH,0FH,0FH,0FH,0FH,0FH,0
FH,0FH,0FH,0FH,0FH
DB
80H,80H,80H,80H,80H,80H,80H,80H,81H,81H,81H,81
H,81H,81H,81H,81H
DB
83H,83H,83H,83H,83H,83H,83H,83H,83H,83H,83H,83
H,83H,83H,83H,83H
DB
87H,87H,87H,87H,87H,87H,87H,87H,87H,87H,87H,87
H,87H,87H,87H,87H
DB
87H,87H,87H,87H,87H,87H,87H,87H,87H,87H,87H,87
H,87H,87H,87H,87H
DB
0C0H,0C0H,0C0H,0C0H,0C0H,0C0H,0C0H,0C0H,0C1,
0C1H,0C1H,0C1H,0C1H,0C1H,0C1H,0C1H
DB
0C3H,0C3H,0C3H,0C3H,0C3H,0C3H,0C3H,0C3H,0C3
H,0C3H,0C3H,0C3H,0C3H,0C3H,0C3H,0C3H
DB
0E0H,0E0H,0E0H,0E0H,0E0H,0E0H,0E0H,0E0H,0E1H
,0E1H,0E1H,0E1H,0E1H,0E1H,0E1H,0E1H
DB
0F0H,0F0H,0F0H,0F0H,0F0H,0F0H,0F0H,0F0H,0F8H,
0F8H,0F8H,0F8H,0FCH,0FCH,0FEH,1FH

;------------- STATE 0 A00 - A07 LABEL TABLE ----------
P0_WAY:
DW
P0_TAB0,P1_TAB0,P2_TAB0,P3_TAB0,P4_TAB0,P5_
TAB0,P6_TAB0,P7_TAB0

;---------------------- P00 - P07 TABLE SERIES --------------
P0_TAB0:
DB 0FFH,0FEH,0FCH,0FCH,0F8H,0F8H,0F8H,0F8H
DB 0F0H,0F0H,0F0H,0F0H,0F0H,0F0H,0F0H,0F0H
DB 0E0H,0E0H,0E0H,0E0H,0E0H,0E0H,0E0H,0E0H
DB 0E0H,0E0H,0E0H,0E0H,0E0H,0E0H,0E0H,0E0H
DB 0C0H,0C0H,0C0H,0C0H,0C0H,0C0H,0C0H,0C0H
DB 0C0H,0C0H,0C0H,0C0H,0C0H,0C0H,0C0H,0C0H
DB 0C0H,0C0H,0C0H,0C0H,0C0H,0C0H,0C0H,0C0H
DB 0C0H,0C0H,0C0H,0C0H,0C0H,0C0H,0C0H,0C0H
DB 80H,80H,80H,80H,80H,80H,80H,80H
DB 80H,80H,80H,80H,80H,80H,80H,80H
DB 80H,80H,80H,80H,80H,80H,80H,80H
DB 80H,80H,80H,80H,80H,80H,80H,80H
DB 80H,80H,80H,80H,80H,80H,80H,80H
DB 80H,80H,80H,80H,80H,80H,80H,80H
DB 80H,80H,80H,80H,80H,80H,80H,80H
DB 80H,80H,80H,80H,80H,80H,80H,80H
DB 00H,00H,00H,00H,00H,00H,00H,00H
DB 00H,00H,00H,00H,00H,00H,00H,00H
DB 00H,00H,00H,00H,00H,00H,00H,00H
DB 00H,00H,00H,00H,00H,00H,00H,00H
DB 00H,00H,00H,00H,00H,00H,00H,00H
DB 00H,00H,00H,00H,00H,00H,00H,00H
DB 00H,00H,00H,00H,00H,00H,00H,00H
DB 00H,00H,00H,00H,00H,00H,00H,00H
DB 00H,00H,00H,00H,00H,00H,00H,00H
DB 00H,00H,00H,00H,00H,00H,00H,00H
DB 00H,00H,00H,00H,00H,00H,00H,00H
DB 00H,00H,00H,00H,00H,00H,00H,00H
DB 00H,00H,00H,00H,00H,00H,00H,00H
DB 00H,00H,00H,00H,00H,00H,00H,00H
DB 00H,00H,00H,00H,00H,00H,00H,00H
DB 00H,00H,00H,00H,00H,00H,00H,00H
```

```
P1_TAB0:
DB 7FH,7EH,7CH,7CH,78H,78H,78H,78H
DB 70H,70H,70H,70H,70H,70H,70H,70H
DB 60H,60H,60H,60H,60H,60H,60H,60H
DB 60H,60H,60H,60H,60H,60H,60H,60H
DB 40H,40H,40H,40H,40H,40H,40H,40H
DB 40H,40H,40H,40H,40H,40H,40H,40H
DB 40H,40H,40H,40H,40H,40H,40H,40H
DB 40H,40H,40H,40H,40H,40H,40H,40H
DB 00H,00H,00H,00H,00H,00H,00H,00H
DB 00H,00H,00H,00H,00H,00H,00H,00H
DB 00H,00H,00H,00H,00H,00H,00H,00H
DB 00H,00H,00H,00H,00H,00H,00H,00H
DB 00H,00H,00H,00H,00H,00H,00H,00H
DB 00H,00H,00H,00H,00H,00H,00H,00H
DB 00H,00H,00H,00H,00H,00H,00H,00H
DB 00H,00H,00H,00H,00H,00H,00H,00H
DB 7FH,7EH,7CH,7CH,78H,78H,78H,78H
DB 70H,70H,70H,70H,70H,70H,70H,70H
DB 60H,60H,60H,60H,60H,60H,60H,60H
DB 60H,60H,60H,60H,60H,60H,60H,60H
DB 40H,40H,40H,40H,40H,40H,40H,40H
DB 40H,40H,40H,40H,40H,40H,40H,40H
DB 40H,40H,40H,40H,40H,40H,40H,40H
DB 40H,40H,40H,40H,40H,40H,40H,40H
DB 00H,00H,00H,00H,00H,00H,00H,00H
DB 00H,00H,00H,00H,00H,00H,00H,00H
DB 00H,00H,00H,00H,00H,00H,00H,00H
DB 00H,00H,00H,00H,00H,00H,00H,00H
DB 00H,00H,00H,00H,00H,00H,00H,00H
DB 00H,00H,00H,00H,00H,00H,00H,00H
DB 00H,00H,00H,00H,00H,00H,00H,00H
DB 00H,00H,00H,00H,00H,00H,00H,00H

P2_TAB0:
DB 3FH,3EH,3CH,3CH,38H,38H,38H,38H
DB 30H,30H,30H,30H,30H,30H,30H,30H
DB 20H,20H,20H,20H,20H,20H,20H,20H
DB 20H,20H,20H,20H,20H,20H,20H,20H
DB 00H,00H,00H,00H,00H,00H,00H,00H
DB 00H,00H,00H,00H,00H,00H,00H,00H
DB 00H,00H,00H,00H,00H,00H,00H,00H
DB 00H,00H,00H,00H,00H,00H,00H,00H
DB 3FH,3EH,3CH,3CH,38H,38H,38H,38H
DB 30H,30H,30H,30H,30H,30H,30H,30H
DB 20H,20H,20H,20H,20H,20H,20H,20H
DB 20H,20H,20H,20H,20H,20H,20H,20H
DB 00H,00H,00H,00H,00H,00H,00H,00H
DB 00H,00H,00H,00H,00H,00H,00H,00H
DB 00H,00H,00H,00H,00H,00H,00H,00H
DB 00H,00H,00H,00H,00H,00H,00H,00H
DB 3FH,3EH,3CH,3CH,38H,38H,38H,38H
DB 30H,30H,30H,30H,30H,30H,30H,30H
DB 20H,20H,20H,20H,20H,20H,20H,20H
DB 20H,20H,20H,20H,20H,20H,20H,20H
DB 00H,00H,00H,00H,00H,00H,00H,00H
DB 00H,00H,00H,00H,00H,00H,00H,00H
DB 00H,00H,00H,00H,00H,00H,00H,00H
DB 00H,00H,00H,00H,00H,00H,00H,00H
DB 3FH,3EH,3CH,3CH,38H,38H,38H,38H
DB 30H,30H,30H,30H,30H,30H,30H,30H
DB 20H,20H,20H,20H,20H,20H,20H,20H
DB 20H,20H,20H,20H,20H,20H,20H,20H
DB 00H,00H,00H,00H,00H,00H,00H,00H
DB 00H,00H,00H,00H,00H,00H,00H,00H
DB 00H,00H,00H,00H,00H,00H,00H,00H
DB 00H,00H,00H,00H,00H,00H,00H,00H

P3_TAB0:
DB 1FH,1EH,1CH,1CH,18H,18H,18H,18H
DB 10H,10H,10H,10H,10H,10H,10H,10H
DB 00H,00H,00H,00H,00H,00H,00H,00H
DB 00H,00H,00H,00H,00H,00H,00H,00H
DB 1FH,1EH,1CH,1CH,18H,18H,18H,18H
DB 10H,10H,10H,10H,10H,10H,10H,10H
DB 00H,00H,00H,00H,00H,00H,00H,00H
DB 00H,00H,00H,00H,00H,00H,00H,00H
DB 1FH,1EH,1CH,1CH,18H,18H,18H,18H
DB 10H,10H,10H,10H,10H,10H,10H,10H
DB 00H,00H,00H,00H,00H,00H,00H,00H
DB 00H,00H,00H,00H,00H,00H,00H,00H
DB 1FH,1EH,1CH,1CH,18H,18H,18H,18H
DB 10H,10H,10H,10H,10H,10H,10H,10H
DB 00H,00H,00H,00H,00H,00H,00H,00H
DB 00H,00H,00H,00H,00H,00H,00H,00H
DB 1FH,1EH,1CH,1CH,18H,18H,18H,18H
DB 10H,10H,10H,10H,10H,10H,10H,10H
DB 00H,00H,00H,00H,00H,00H,00H,00H
DB 00H,00H,00H,00H,00H,00H,00H,00H
DB 1FH,1EH,1CH,1CH,18H,18H,18H,18H
DB 10H,10H,10H,10H,10H,10H,10H,10H
DB 00H,00H,00H,00H,00H,00H,00H,00H
DB 00H,00H,00H,00H,00H,00H,00H,00H
DB 1FH,1EH,1CH,1CH,18H,18H,18H,18H
DB 10H,10H,10H,10H,10H,10H,10H,10H
DB 00H,00H,00H,00H,00H,00H,00H,00H
DB 00H,00H,00H,00H,00H,00H,00H,00H
DB 1FH,1EH,1CH,1CH,18H,18H,18H,18H
DB 10H,10H,10H,10H,10H,10H,10H,10H
DB 00H,00H,00H,00H,00H,00H,00H,00H
DB 00H,00H,00H,00H,00H,00H,00H,00H

P4_TAB0:
DB 0FH,0EH,0CH,0CH,08H,08H,08H,08H
DB 00H,00H,00H,00H,00H,00H,00H,00H
DB 0FH,0EH,0CH,0CH,08H,08H,08H,08H
DB 00H,00H,00H,00H,00H,00H,00H,00H
DB 0FH,0EH,0CH,0CH,08H,08H,08H,08H
DB 00H,00H,00H,00H,00H,00H,00H,00H
DB 0FH,0EH,0CH,0CH,08H,08H,08H,08H
DB 00H,00H,00H,00H,00H,00H,00H,00H
DB 0FH,0EH,0CH,0CH,08H,08H,08H,08H
DB 00H,00H,00H,00H,00H,00H,00H,00H
DB 0FH,0EH,0CH,0CH,08H,08H,08H,08H
DB 00H,00H,00H,00H,00H,00H,00H,00H
DB 0FH,0EH,0CH,0CH,08H,08H,08H,08H
DB 00H,00H,00H,00H,00H,00H,00H,00H
DB 0FH,0EH,0CH,0CH,08H,08H,08H,08H
DB 00H,00H,00H,00H,00H,00H,00H,00H
DB 0FH,0EH,0CH,0CH,08H,08H,08H,08H
DB 00H,00H,00H,00H,00H,00H,00H,00H
DB 0FH,0EH,0CH,0CH,08H,08H,08H,08H
DB 00H,00H,00H,00H,00H,00H,00H,00H
DB 0FH,0EH,0CH,0CH,08H,08H,08H,08H
DB 00H,00H,00H,00H,00H,00H,00H,00H
DB 0FH,0EH,0CH,0CH,08H,08H,08H,08H
DB 00H,00H,00H,00H,00H,00H,00H,00H
DB 0FH,0EH,0CH,0CH,08H,08H,08H,08H
DB 00H,00H,00H,00H,00H,00H,00H,00H
DB 0FH,0EH,0CH,0CH,08H,08H,08H,08H
DB 00H,00H,00H,00H,00H,00H,00H,00H
DB 0FH,0EH,0CH,0CH,08H,08H,08H,08H
DB 00H,00H,00H,00H,00H,00H,00H,00H
DB 0FH,0EH,0CH,0CH,08H,08H,08H,08H
DB 00H,00H,00H,00H,00H,00H,00H,00H

P5_TAB0:
DB 07H,06H,04H,04H,00H,00H,00H,00H
DB 07H,06H,04H,04H,00H,00H,00H,00H
DB 07H,06H,04H,04H,00H,00H,00H,00H
DB 07H,06H,04H,04H,00H,00H,00H,00H
DB 07H,06H,04H,04H,00H,00H,00H,00H
DB 07H,06H,04H,04H,00H,00H,00H,00H
DB 07H,06H,04H,04H,00H,00H,00H,00H
DB 07H,06H,04H,04H,00H,00H,00H,00H
DB 07H,06H,04H,04H,00H,00H,00H,00H
```

```
DB 07H,06H,04H,04H,00H,00H,00H,00H
DB 07H,06H,04H,04H,00H,00H,00H,00H
DB 07H,06H,04H,04H,00H,00H,00H,00H
DB 07H,06H,04H,04H,00H,00H,00H,00H
DB 07H,06H,04H,04H,00H,00H,00H,00H
DB 07H,06H,04H,04H,00H,00H,00H,00H
DB 07H,06H,04H,04H,00H,00H,00H,00H
DB 07H,06H,04H,04H,00H,00H,00H,00H
DB 07H,06H,04H,04H,00H,00H,00H,00H
DB 07H,06H,04H,04H,00H,00H,00H,00H
DB 07H,06H,04H,04H,00H,00H,00H,00H
DB 07H,06H,04H,04H,00H,00H,00H,00H
DB 07H,06H,04H,04H,00H,00H,00H,00H
DB 07H,06H,04H,04H,00H,00H,00H,00H
DB 07H,06H,04H,04H,00H,00H,00H,00H
DB 07H,06H,04H,04H,00H,00H,00H,00H
DB 07H,06H,04H,04H,00H,00H,00H,00H
DB 07H,06H,04H,04H,00H,00H,00H,00H
DB 07H,06H,04H,04H,00H,00H,00H,00H
DB 07H,06H,04H,04H,00H,00H,00H,00H
DB 07H,06H,04H,04H,00H,00H,00H,00H
DB 07H,06H,04H,04H,00H,00H,00H,00H
DB 07H,06H,04H,04H,00H,00H,00H,00H

P6_TAB0:
DB 03H,02H,00H,00H,03H,02H,00H,00H
DB 03H,02H,00H,00H,03H,02H,00H,00H
DB 03H,02H,00H,00H,03H,02H,00H,00H
DB 03H,02H,00H,00H,03H,02H,00H,00H
DB 03H,02H,00H,00H,03H,02H,00H,00H
DB 03H,02H,00H,00H,03H,02H,00H,00H
DB 03H,02H,00H,00H,03H,02H,00H,00H
DB 03H,02H,00H,00H,03H,02H,00H,00H
DB 03H,02H,00H,00H,03H,02H,00H,00H
DB 03H,02H,00H,00H,03H,02H,00H,00H
DB 03H,02H,00H,00H,03H,02H,00H,00H
DB 03H,02H,00H,00H,03H,02H,00H,00H
DB 03H,02H,00H,00H,03H,02H,00H,00H
DB 03H,02H,00H,00H,03H,02H,00H,00H
DB 03H,02H,00H,00H,03H,02H,00H,00H
DB 03H,02H,00H,00H,03H,02H,00H,00H
DB 03H,02H,00H,00H,03H,02H,00H,00H
DB 03H,02H,00H,00H,03H,02H,00H,00H
DB 03H,02H,00H,00H,03H,02H,00H,00H
DB 03H,02H,00H,00H,03H,02H,00H,00H
DB 03H,02H,00H,00H,03H,02H,00H,00H
DB 03H,02H,00H,00H,03H,02H,00H,00H
DB 03H,02H,00H,00H,03H,02H,00H,00H
DB 03H,02H,00H,00H,03H,02H,00H,00H
DB 03H,02H,00H,00H,03H,02H,00H,00H
DB 03H,02H,00H,00H,03H,02H,00H,00H
DB 03H,02H,00H,00H,03H,02H,00H,00H
DB 03H,02H,00H,00H,03H,02H,00H,00H
DB 03H,02H,00H,00H,03H,02H,00H,00H
DB 03H,02H,00H,00H,03H,02H,00H,00H
DB 03H,02H,00H,00H,03H,02H,00H,00H
DB 03H,02H,00H,00H,03H,02H,00H,00H

P7_TAB0:
DB 01H,00H,01H,00H,03H,00H,01H,00H
DB 01H,00H,01H,00H,03H,00H,01H,00H
DB 01H,00H,01H,00H,03H,00H,01H,00H
DB 01H,00H,01H,00H,03H,00H,01H,00H
DB 01H,00H,01H,00H,03H,00H,01H,00H
DB 01H,00H,01H,00H,03H,00H,01H,00H
DB 01H,00H,01H,00H,03H,00H,01H,00H
DB 01H,00H,01H,00H,03H,00H,01H,00H
DB 01H,00H,01H,00H,03H,00H,01H,00H
DB 01H,00H,01H,00H,03H,00H,01H,00H
DB 01H,00H,01H,00H,03H,00H,01H,00H
DB 01H,00H,01H,00H,03H,00H,01H,00H
DB 01H,00H,01H,00H,03H,00H,01H,00H
DB 01H,00H,01H,00H,03H,00H,01H,00H
DB 01H,00H,01H,00H,03H,00H,01H,00H
DB 01H,00H,01H,00H,03H,00H,01H,00H
DB 01H,00H,01H,00H,03H,00H,01H,00H
DB 01H,00H,01H,00H,03H,00H,01H,00H
DB 01H,00H,01H,00H,03H,00H,01H,00H
DB 01H,00H,01H,00H,03H,00H,01H,00H
DB 01H,00H,01H,00H,03H,00H,01H,00H
DB 01H,00H,01H,00H,03H,00H,01H,00H
DB 01H,00H,01H,00H,03H,00H,01H,00H
DB 01H,00H,01H,00H,03H,00H,01H,00H
DB 01H,00H,01H,00H,03H,00H,01H,00H
DB 01H,00H,01H,00H,03H,00H,01H,00H
DB 01H,00H,01H,00H,03H,00H,01H,00H
DB 01H,00H,01H,00H,03H,00H,01H,00H
DB 01H,00H,01H,00H,03H,00H,01H,00H
DB 01H,00H,01H,00H,03H,00H,01H,00H
DB 01H,00H,01H,00H,03H,00H,01H,00H
DB 01H,00H,01H,00H,03H,00H,01H,00H

;-----------------  STATE 0 P00 - P07 LABEL TABLE -------
P1_WAY:
DW
P0_TAB1,P1_TAB1,P2_TAB1,P3_TAB1,P4_TAB1,P5_
TAB1,P6_TAB1,P7_TAB1

;---------------------- P00 - P07 TABLE SERIES -------------
P0_TAB1:
DB 0FFH,0FFH,0FFH,0FFH,0FFH,0FFH,0FFH,0FFH
DB 0FFH,0FFH,0FFH,0FFH,0FFH,0FFH,0FFH,0FFH
DB 0FFH,0FFH,0FFH,0FFH,0FFH,0FFH,0FFH,0FFH
DB 0FFH,0FFH,0FFH,0FFH,0FFH,0FFH,0FFH,0FFH
DB 0FFH,0FFH,0FFH,0FFH,0FFH,0FFH,0FFH,0FFH
DB 0FFH,0FFH,0FFH,0FFH,0FFH,0FFH,0FFH,0FFH
DB 0FFH,0FFH,0FFH,0FFH,0FFH,0FFH,0FFH,0FFH
DB 0FFH,0FFH,0FFH,0FFH,0FFH,0FFH,0FFH,0FFH
DB 0FFH,0FFH,0FFH,0FFH,0FFH,0FFH,0FFH,0FFH
DB 0FFH,0FFH,0FFH,0FFH,0FFH,0FFH,0FFH,0FFH
DB 0FFH,0FFH,0FFH,0FFH,0FFH,0FFH,0FFH,0FFH
DB 0FFH,0FFH,0FFH,0FFH,0FFH,0FFH,0FFH,0FFH
DB 0FFH,0FFH,0FFH,0FFH,0FFH,0FFH,0FFH,0FFH
DB 0FFH,0FFH,0FFH,0FFH,0FFH,0FFH,0FFH,0FFH
DB 0FFH,0FFH,0FFH,0FFH,0FFH,0FFH,0FFH,0FFH
DB 0FFH,0FFH,0FFH,0FFH,0FFH,0FFH,0FFH,0FFH
DB 7FH,7FH,7FH,7FH,7FH,7FH,7FH,7FH
DB 7FH,7FH,7FH,7FH,7FH,7FH,7FH,7FH
DB 7FH,7FH,7FH,7FH,7FH,7FH,7FH,7FH
DB 7FH,7FH,7FH,7FH,7FH,7FH,7FH,7FH
DB 7FH,7FH,7FH,7FH,7FH,7FH,7FH,7FH
DB 7FH,7FH,7FH,7FH,7FH,7FH,7FH,7FH
DB 7FH,7FH,7FH,7FH,7FH,7FH,7FH,7FH
DB 7FH,7FH,7FH,7FH,7FH,7FH,7FH,7FH
DB 3FH,3FH,3FH,3FH,3FH,3FH,3FH,3FH
DB 3FH,3FH,3FH,3FH,3FH,3FH,3FH,3FH
DB 3FH,3FH,3FH,3FH,3FH,3FH,3FH,3FH
DB 3FH,3FH,3FH,3FH,3FH,3FH,3FH,3FH
DB 1FH,1FH,1FH,1FH,1FH,1FH,1FH,1FH
DB 1FH,1FH,1FH,1FH,1FH,1FH,1FH,1FH
DB 0FH,0FH,0FH,0FH,0FH,0FH,0FH,0FH
DB 07H,07H,07H,07H,03H,03H,01H,00H

P1_TAB1:
DB 0FFH,0FFH,0FFH,0FFH,0FFH,0FFH,0FFH,0FFH
DB 0FFH,0FFH,0FFH,0FFH,0FFH,0FFH,0FFH,0FFH
DB 0FFH,0FFH,0FFH,0FFH,0FFH,0FFH,0FFH,0FFH
DB 0FFH,0FFH,0FFH,0FFH,0FFH,0FFH,0FFH,0FFH
DB 0FFH,0FFH,0FFH,0FFH,0FFH,0FFH,0FFH,0FFH
DB 0FFH,0FFH,0FFH,0FFH,0FFH,0FFH,0FFH,0FFH
DB 0FFH,0FFH,0FFH,0FFH,0FFH,0FFH,0FFH,0FFH
DB 0FFH,0FFH,0FFH,0FFH,0FFH,0FFH,0FFH,0FFH
DB 0BFH,0BFH,0BFH,0BFH,0BFH,0BFH,0BFH,0BFH
DB 0BFH,0BFH,0BFH,0BFH,0BFH,0BFH,0BFH,0BFH
DB 0BFH,0BFH,0BFH,0BFH,0BFH,0BFH,0BFH,0BFH
```

```
DB 0BFH,0BFH,0BFH,0BFH,0BFH,0BFH,0BFH,0BFH
DB 9FH,9FH,9FH,9FH,9FH,9FH,9FH,9FH
DB 9FH,9FH,9FH,9FH,9FH,9FH,9FH,9FH
DB 8FH,8FH,8FH,8FH,8FH,8FH,8FH,8FH
DB 87H,87H,87H,87H,83H,83H,81H,80H
DB 0FFH,0FFH,0FFH,0FFH,0FFH,0FFH,0FFH,0FFH
DB 0FFH,0FFH,0FFH,0FFH,0FFH,0FFH,0FFH,0FFH
DB 0FFH,0FFH,0FFH,0FFH,0FFH,0FFH,0FFH,0FFH
DB 0FFH,0FFH,0FFH,0FFH,0FFH,0FFH,0FFH,0FFH
DB 0FFH,0FFH,0FFH,0FFH,0FFH,0FFH,0FFH,0FFH
DB 0FFH,0FFH,0FFH,0FFH,0FFH,0FFH,0FFH,0FFH
DB 0FFH,0FFH,0FFH,0FFH,0FFH,0FFH,0FFH,0FFH
DB 0FFH,0FFH,0FFH,0FFH,0FFH,0FFH,0FFH,0FFH
DB 0BFH,0BFH,0BFH,0BFH,0BFH,0BFH,0BFH,0BFH
DB 0BFH,0BFH,0BFH,0BFH,0BFH,0BFH,0BFH,0BFH
DB 0BFH,0BFH,0BFH,0BFH,0BFH,0BFH,0BFH,0BFH
DB 0BFH,0BFH,0BFH,0BFH,0BFH,0BFH,0BFH,0BFH
DB 9FH,9FH,9FH,9FH,9FH,9FH,9FH,9FH
DB 9FH,9FH,9FH,9FH,9FH,9FH,9FH,9FH
DB 8FH,8FH,8FH,8FH,8FH,8FH,8FH,8FH
DB 87H,87H,87H,87H,83H,83H,81H,80H


P2_TAB1:
DB 0FFH,0FFH,0FFH,0FFH,0FFH,0FFH,0FFH,0FFH
DB 0FFH,0FFH,0FFH,0FFH,0FFH,0FFH,0FFH,0FFH
DB 0FFH,0FFH,0FFH,0FFH,0FFH,0FFH,0FFH,0FFH
DB 0FFH,0FFH,0FFH,0FFH,0FFH,0FFH,0FFH,0FFH
DB 0DFH,0DFH,0DFH,0DFH,0DFH,0DFH,0DFH,0DFH
DB 0DFH,0DFH,0DFH,0DFH,0DFH,0DFH,0DFH,0DFH
DB 0CFH,0CFH,0CFH,0CFH,0CFH,0CFH,0CFH,0CFH
DB 0C7H,0C7H,0C7H,0C7H,0C3H,0C3H,0C1H,0C0H
DB 0FFH,0FFH,0FFH,0FFH,0FFH,0FFH,0FFH,0FFH
DB 0FFH,0FFH,0FFH,0FFH,0FFH,0FFH,0FFH,0FFH
DB 0FFH,0FFH,0FFH,0FFH,0FFH,0FFH,0FFH,0FFH
DB 0FFH,0FFH,0FFH,0FFH,0FFH,0FFH,0FFH,0FFH
DB 0DFH,0DFH,0DFH,0DFH,0DFH,0DFH,0DFH,0DFH
DB 0DFH,0DFH,0DFH,0DFH,0DFH,0DFH,0DFH,0DFH
DB 0CFH,0CFH,0CFH,0CFH,0CFH,0CFH,0CFH,0CFH
DB 0C7H,0C7H,0C7H,0C7H,0C3H,0C3H,0C1H,0C0H
DB 0FFH,0FFH,0FFH,0FFH,0FFH,0FFH,0FFH,0FFH
DB 0FFH,0FFH,0FFH,0FFH,0FFH,0FFH,0FFH,0FFH
DB 0FFH,0FFH,0FFH,0FFH,0FFH,0FFH,0FFH,0FFH
DB 0FFH,0FFH,0FFH,0FFH,0FFH,0FFH,0FFH,0FFH
DB 0DFH,0DFH,0DFH,0DFH,0DFH,0DFH,0DFH,0DFH
DB 0DFH,0DFH,0DFH,0DFH,0DFH,0DFH,0DFH,0DFH
DB 0CFH,0CFH,0CFH,0CFH,0CFH,0CFH,0CFH,0CFH
DB 0C7H,0C7H,0C7H,0C7H,0C3H,0C3H,0C1H,0C0H
DB 0FFH,0FFH,0FFH,0FFH,0FFH,0FFH,0FFH,0FFH
DB 0FFH,0FFH,0FFH,0FFH,0FFH,0FFH,0FFH,0FFH
DB 0FFH,0FFH,0FFH,0FFH,0FFH,0FFH,0FFH,0FFH
DB 0FFH,0FFH,0FFH,0FFH,0FFH,0FFH,0FFH,0FFH
DB 0DFH,0DFH,0DFH,0DFH,0DFH,0DFH,0DFH,0DFH
DB 0DFH,0DFH,0DFH,0DFH,0DFH,0DFH,0DFH,0DFH
DB 0CFH,0CFH,0CFH,0CFH,0CFH,0CFH,0CFH,0CFH
DB 0C7H,0C7H,0C7H,0C7H,0C3H,0C3H,0C1H,0C0H

P3_TAB1:
DB 0FFH,0FFH,0FFH,0FFH,0FFH,0FFH,0FFH,0FFH
DB 0FFH,0FFH,0FFH,0FFH,0FFH,0FFH,0FFH,0FFH
DB 0EFH,0EFH,0EFH,0EFH,0EFH,0EFH,0EFH,0EFH
DB 0E7H,0E7H,0E7H,0E7H,0E3H,0E3H,0E1H,0E0H
DB 0FFH,0FFH,0FFH,0FFH,0FFH,0FFH,0FFH,0FFH
DB 0FFH,0FFH,0FFH,0FFH,0FFH,0FFH,0FFH,0FFH
DB 0EFH,0EFH,0EFH,0EFH,0EFH,0EFH,0EFH,0EFH
DB 0E7H,0E7H,0E7H,0E7H,0E3H,0E3H,0E1H,0E0H
DB 0FFH,0FFH,0FFH,0FFH,0FFH,0FFH,0FFH,0FFH
DB 0FFH,0FFH,0FFH,0FFH,0FFH,0FFH,0FFH,0FFH
DB 0EFH,0EFH,0EFH,0EFH,0EFH,0EFH,0EFH,0EFH
DB 0E7H,0E7H,0E7H,0E7H,0E3H,0E3H,0E1H,0E0H
DB 0FFH,0FFH,0FFH,0FFH,0FFH,0FFH,0FFH,0FFH
DB 0FFH,0FFH,0FFH,0FFH,0FFH,0FFH,0FFH,0FFH
DB 0EFH,0EFH,0EFH,0EFH,0EFH,0EFH,0EFH,0EFH
DB 0E7H,0E7H,0E7H,0E7H,0E3H,0E3H,0E1H,0E0H
DB 0FFH,0FFH,0FFH,0FFH,0FFH,0FFH,0FFH,0FFH
DB 0FFH,0FFH,0FFH,0FFH,0FFH,0FFH,0FFH,0FFH
DB 0EFH,0EFH,0EFH,0EFH,0EFH,0EFH,0EFH,0EFH
DB 0E7H,0E7H,0E7H,0E7H,0E3H,0E3H,0E1H,0E0H
DB 0FFH,0FFH,0FFH,0FFH,0FFH,0FFH,0FFH,0FFH
DB 0FFH,0FFH,0FFH,0FFH,0FFH,0FFH,0FFH,0FFH
DB 0EFH,0EFH,0EFH,0EFH,0EFH,0EFH,0EFH,0EFH
DB 0E7H,0E7H,0E7H,0E7H,0E3H,0E3H,0E1H,0E0H
DB 0FFH,0FFH,0FFH,0FFH,0FFH,0FFH,0FFH,0FFH
DB 0FFH,0FFH,0FFH,0FFH,0FFH,0FFH,0FFH,0FFH
DB 0EFH,0EFH,0EFH,0EFH,0EFH,0EFH,0EFH,0EFH
DB 0E7H,0E7H,0E7H,0E7H,0E3H,0E3H,0E1H,0E0H
DB 0FFH,0FFH,0FFH,0FFH,0FFH,0FFH,0FFH,0FFH
DB 0FFH,0FFH,0FFH,0FFH,0FFH,0FFH,0FFH,0FFH
DB 0EFH,0EFH,0EFH,0EFH,0EFH,0EFH,0EFH,0EFH
DB 0E7H,0E7H,0E7H,0E7H,0E3H,0E3H,0E1H,0E0H

P4_TAB1:
DB 0FFH,0FFH,0FFH,0FFH,0FFH,0FFH,0FFH,0FFH
DB 0F7H,0F7H,0F7H,0F7H,0F3H,0F3H,0F1H,0F0H
DB 0FFH,0FFH,0FFH,0FFH,0FFH,0FFH,0FFH,0FFH
DB 0F7H,0F7H,0F7H,0F7H,0F3H,0F3H,0F1H,0F0H
DB 0FFH,0FFH,0FFH,0FFH,0FFH,0FFH,0FFH,0FFH
DB 0F7H,0F7H,0F7H,0F7H,0F3H,0F3H,0F1H,0F0H
DB 0FFH,0FFH,0FFH,0FFH,0FFH,0FFH,0FFH,0FFH
DB 0F7H,0F7H,0F7H,0F7H,0F3H,0F3H,0F1H,0F0H
DB 0FFH,0FFH,0FFH,0FFH,0FFH,0FFH,0FFH,0FFH
DB 0F7H,0F7H,0F7H,0F7H,0F3H,0F3H,0F1H,0F0H
DB 0FFH,0FFH,0FFH,0FFH,0FFH,0FFH,0FFH,0FFH
DB 0F7H,0F7H,0F7H,0F7H,0F3H,0F3H,0F1H,0F0H
DB 0FFH,0FFH,0FFH,0FFH,0FFH,0FFH,0FFH,0FFH
DB 0F7H,0F7H,0F7H,0F7H,0F3H,0F3H,0F1H,0F0H
DB 0FFH,0FFH,0FFH,0FFH,0FFH,0FFH,0FFH,0FFH
DB 0F7H,0F7H,0F7H,0F7H,0F3H,0F3H,0F1H,0F0H
DB 0FFH,0FFH,0FFH,0FFH,0FFH,0FFH,0FFH,0FFH
DB 0F7H,0F7H,0F7H,0F7H,0F3H,0F3H,0F1H,0F0H
DB 0FFH,0FFH,0FFH,0FFH,0FFH,0FFH,0FFH,0FFH
DB 0F7H,0F7H,0F7H,0F7H,0F3H,0F3H,0F1H,0F0H
DB 0FFH,0FFH,0FFH,0FFH,0FFH,0FFH,0FFH,0FFH
DB 0F7H,0F7H,0F7H,0F7H,0F3H,0F3H,0F1H,0F0H
DB 0FFH,0FFH,0FFH,0FFH,0FFH,0FFH,0FFH,0FFH
DB 0F7H,0F7H,0F7H,0F7H,0F3H,0F3H,0F1H,0F0H
DB 0FFH,0FFH,0FFH,0FFH,0FFH,0FFH,0FFH,0FFH
DB 0F7H,0F7H,0F7H,0F7H,0F3H,0F3H,0F1H,0F0H
DB 0FFH,0FFH,0FFH,0FFH,0FFH,0FFH,0FFH,0FFH
DB 0F7H,0F7H,0F7H,0F7H,0F3H,0F3H,0F1H,0F0H
DB 0FFH,0FFH,0FFH,0FFH,0FFH,0FFH,0FFH,0FFH
DB 0F7H,0F7H,0F7H,0F7H,0F3H,0F3H,0F1H,0F0H
DB 0FFH,0FFH,0FFH,0FFH,0FFH,0FFH,0FFH,0FFH
DB 0F7H,0F7H,0F7H,0F7H,0F3H,0F3H,0F1H,0F0H

P5_TAB1:
DB 0FFH,0FFH,0FFH,0FFH,0FBH,0FBH,0F9H,0F8H
DB 0FFH,0FFH,0FFH,0FFH,0FBH,0FBH,0F9H,0F8H
DB 0FFH,0FFH,0FFH,0FFH,0FBH,0FBH,0F9H,0F8H
DB 0FFH,0FFH,0FFH,0FFH,0FBH,0FBH,0F9H,0F8H
DB 0FFH,0FFH,0FFH,0FFH,0FBH,0FBH,0F9H,0F8H
DB 0FFH,0FFH,0FFH,0FFH,0FBH,0FBH,0F9H,0F8H
DB 0FFH,0FFH,0FFH,0FFH,0FBH,0FBH,0F9H,0F8H
DB 0FFH,0FFH,0FFH,0FFH,0FBH,0FBH,0F9H,0F8H
DB 0FFH,0FFH,0FFH,0FFH,0FBH,0FBH,0F9H,0F8H
DB 0FFH,0FFH,0FFH,0FFH,0FBH,0FBH,0F9H,0F8H
DB 0FFH,0FFH,0FFH,0FFH,0FBH,0FBH,0F9H,0F8H
DB 0FFH,0FFH,0FFH,0FFH,0FBH,0FBH,0F9H,0F8H
DB 0FFH,0FFH,0FFH,0FFH,0FBH,0FBH,0F9H,0F8H
DB 0FFH,0FFH,0FFH,0FFH,0FBH,0FBH,0F9H,0F8H
DB 0FFH,0FFH,0FFH,0FFH,0FBH,0FBH,0F9H,0F8H
DB 0FFH,0FFH,0FFH,0FFH,0FBH,0FBH,0F9H,0F8H
DB 0FFH,0FFH,0FFH,0FFH,0FBH,0FBH,0F9H,0F8H
DB 0FFH,0FFH,0FFH,0FFH,0FBH,0FBH,0F9H,0F8H
DB 0FFH,0FFH,0FFH,0FFH,0FBH,0FBH,0F9H,0F8H
DB 0FFH,0FFH,0FFH,0FFH,0FBH,0FBH,0F9H,0F8H
```

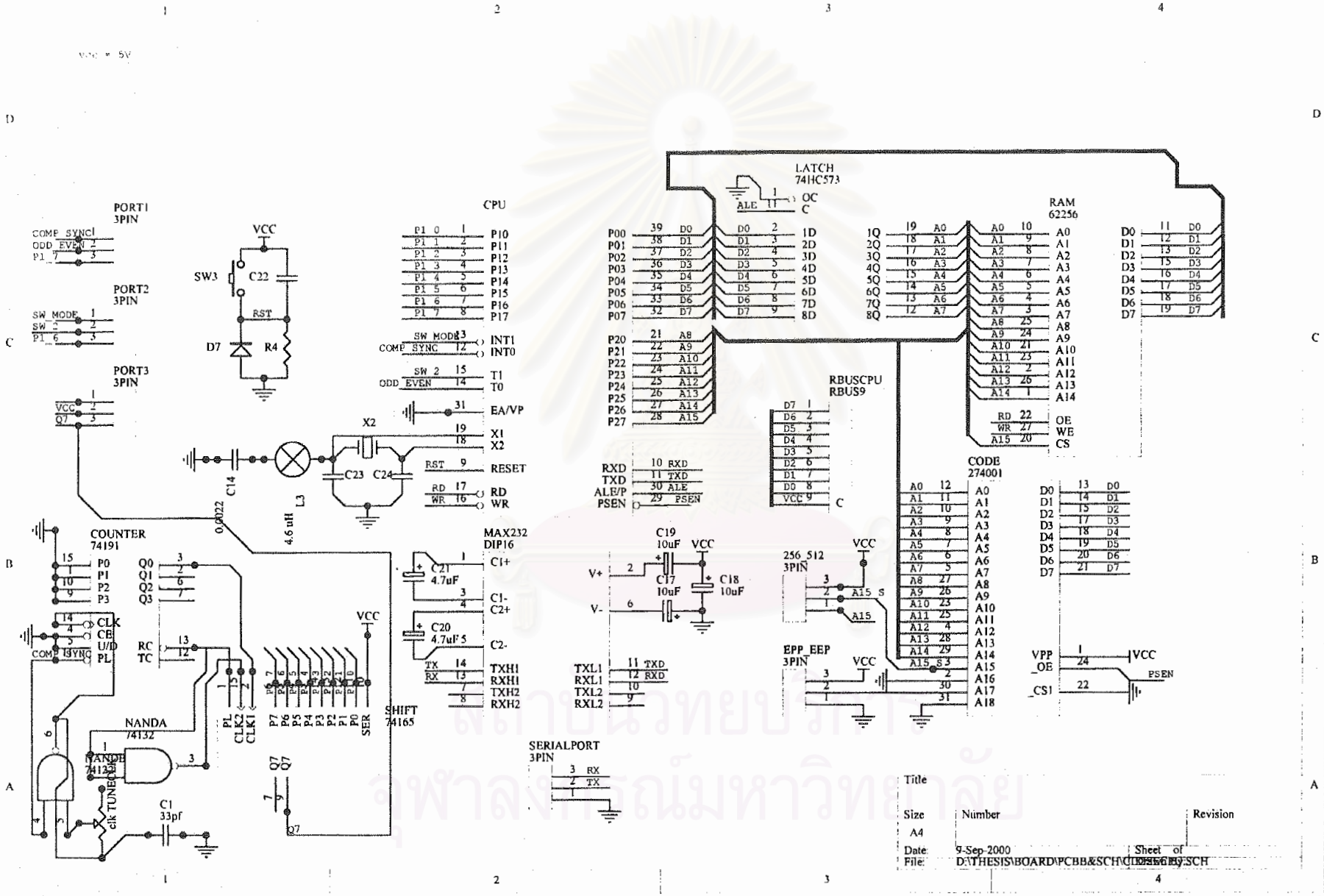
```
DB 0FFH,0FFH,0FFH,0FFH,0FBH,0FBH,0F9H,0F8H
DB 0FFH,0FFH,0FFH,0FFH,0FBH,0FBH,0F9H,0F8H
DB 0FFH,0FFH,0FFH,0FFH,0FBH,0FBH,0F9H,0F8H
DB 0FFH,0FFH,0FFH,0FFH,0FBH,0FBH,0F9H,0F8H
DB 0FFH,0FFH,0FFH,0FFH,0FBH,0FBH,0F9H,0F8H
DB 0FFH,0FFH,0FFH,0FFH,0FBH,0FBH,0F9H,0F8H
DB 0FFH,0FFH,0FFH,0FFH,0FBH,0FBH,0F9H,0F8H
DB 0FFH,0FFH,0FFH,0FFH,0FBH,0FBH,0F9H,0F8H
DB 0FFH,0FFH,0FFH,0FFH,0FBH,0FBH,0F9H,0F8H
DB 0FFH,0FFH,0FFH,0FFH,0FBH,0FBH,0F9H,0F8H
DB 0FFH,0FFH,0FFH,0FFH,0FBH,0FBH,0F9H,0F8H
DB 0FFH,0FFH,0FFH,0FFH,0FBH,0FBH,0F9H,0F8H

P6_TAB1:
DB 0FFH,0FFH,0FDH,0FCH,0FFH,0FFH,0FDH,0FCH
DB 0FFH,0FFH,0FDH,0FCH,0FFH,0FFH,0FDH,0FCH
DB 0FFH,0FFH,0FDH,0FCH,0FFH,0FFH,0FDH,0FCH
DB 0FFH,0FFH,0FDH,0FCH,0FFH,0FFH,0FDH,0FCH
DB 0FFH,0FFH,0FDH,0FCH,0FFH,0FFH,0FDH,0FCH
DB 0FFH,0FFH,0FDH,0FCH,0FFH,0FFH,0FDH,0FCH
DB 0FFH,0FFH,0FDH,0FCH,0FFH,0FFH,0FDH,0FCH
DB 0FFH,0FFH,0FDH,0FCH,0FFH,0FFH,0FDH,0FCH
DB 0FFH,0FFH,0FDH,0FCH,0FFH,0FFH,0FDH,0FCH
DB 0FFH,0FFH,0FDH,0FCH,0FFH,0FFH,0FDH,0FCH
DB 0FFH,0FFH,0FDH,0FCH,0FFH,0FFH,0FDH,0FCH
DB 0FFH,0FFH,0FDH,0FCH,0FFH,0FFH,0FDH,0FCH
DB 0FFH,0FFH,0FDH,0FCH,0FFH,0FFH,0FDH,0FCH
DB 0FFH,0FFH,0FDH,0FCH,0FFH,0FFH,0FDH,0FCH
DB 0FFH,0FFH,0FDH,0FCH,0FFH,0FFH,0FDH,0FCH
DB 0FFH,0FFH,0FDH,0FCH,0FFH,0FFH,0FDH,0FCH
DB 0FFH,0FFH,0FDH,0FCH,0FFH,0FFH,0FDH,0FCH
DB 0FFH,0FFH,0FDH,0FCH,0FFH,0FFH,0FDH,0FCH
DB 0FFH,0FFH,0FDH,0FCH,0FFH,0FFH,0FDH,0FCH
DB 0FFH,0FFH,0FDH,0FCH,0FFH,0FFH,0FDH,0FCH
DB 0FFH,0FFH,0FDH,0FCH,0FFH,0FFH,0FDH,0FCH
DB 0FFH,0FFH,0FDH,0FCH,0FFH,0FFH,0FDH,0FCH
DB 0FFH,0FFH,0FDH,0FCH,0FFH,0FFH,0FDH,0FCH
DB 0FFH,0FFH,0FDH,0FCH,0FFH,0FFH,0FDH,0FCH
DB 0FFH,0FFH,0FDH,0FCH,0FFH,0FFH,0FDH,0FCH
DB 0FFH,0FFH,0FDH,0FCH,0FFH,0FFH,0FDH,0FCH
DB 0FFH,0FFH,0FDH,0FCH,0FFH,0FFH,0FDH,0FCH
DB 0FFH,0FFH,0FDH,0FCH,0FFH,0FFH,0FDH,0FCH
DB 0FFH,0FFH,0FDH,0FCH,0FFH,0FFH,0FDH,0FCH
DB 0FFH,0FFH,0FDH,0FCH,0FFH,0FFH,0FDH,0FCH
DB 0FFH,0FFH,0FDH,0FCH,0FFH,0FFH,0FDH,0FCH
DB 0FFH,0FFH,0FDH,0FCH,0FFH,0FFH,0FDH,0FCH

P7_TAB1:
DB 0FFH,0FEH,0FFH,0FEH,0FFH,0FEH,0FFH,0FEH
DB 0FFH,0FEH,0FFH,0FEH,0FFH,0FEH,0FFH,0FEH
DB 0FFH,0FEH,0FFH,0FEH,0FFH,0FEH,0FFH,0FEH
DB 0FFH,0FEH,0FFH,0FEH,0FFH,0FEH,0FFH,0FEH
DB 0FFH,0FEH,0FFH,0FEH,0FFH,0FEH,0FFH,0FEH
DB 0FFH,0FEH,0FFH,0FEH,0FFH,0FEH,0FFH,0FEH
DB 0FFH,0FEH,0FFH,0FEH,0FFH,0FEH,0FFH,0FEH
DB 0FFH,0FEH,0FFH,0FEH,0FFH,0FEH,0FFH,0FEH
DB 0FFH,0FEH,0FFH,0FEH,0FFH,0FEH,0FFH,0FEH
DB 0FFH,0FEH,0FFH,0FEH,0FFH,0FEH,0FFH,0FEH
DB 0FFH,0FEH,0FFH,0FEH,0FFH,0FEH,0FFH,0FEH
DB 0FFH,0FEH,0FFH,0FEH,0FFH,0FEH,0FFH,0FEH
DB 0FFH,0FEH,0FFH,0FEH,0FFH,0FEH,0FFH,0FEH
DB 0FFH,0FEH,0FFH,0FEH,0FFH,0FEH,0FFH,0FEH
DB 0FFH,0FEH,0FFH,0FEH,0FFH,0FEH,0FFH,0FEH
DB 0FFH,0FEH,0FFH,0FEH,0FFH,0FEH,0FFH,0FEH
DB 0FFH,0FEH,0FFH,0FEH,0FFH,0FEH,0FFH,0FEH
DB 0FFH,0FEH,0FFH,0FEH,0FFH,0FEH,0FFH,0FEH
DB 0FFH,0FEH,0FFH,0FEH,0FFH,0FEH,0FFH,0FEH
DB 0FFH,0FEH,0FFH,0FEH,0FFH,0FEH,0FFH,0FEH
DB 0FFH,0FEH,0FFH,0FEH,0FFH,0FEH,0FFH,0FEH
DB 0FFH,0FEH,0FFH,0FEH,0FFH,0FEH,0FFH,0FEH
DB 0FFH,0FEH,0FFH,0FEH,0FFH,0FEH,0FFH,0FEH
DB 0FFH,0FEH,0FFH,0FEH,0FFH,0FEH,0FFH,0FEH
DB 0FFH,0FEH,0FFH,0FEH,0FFH,0FEH,0FFH,0FEH
DB 0FFH,0FEH,0FFH,0FEH,0FFH,0FEH,0FFH,0FEH
DB 0FFH,0FEH,0FFH,0FEH,0FFH,0FEH,0FFH,0FEH
DB 0FFH,0FEH,0FFH,0FEH,0FFH,0FEH,0FFH,0FEH
DB 0FFH,0FEH,0FFH,0FEH,0FFH,0FEH,0FFH,0FEH
DB 0FFH,0FEH,0FFH,0FEH,0FFH,0FEH,0FFH,0FEH
DB 0FFH,0FEH,0FFH,0FEH,0FFH,0FEH,0FFH,0FEH
DB 0FFH,0FEH,0FFH,0FEH,0FFH,0FEH,0FFH,0FEH

;----------------DEFIND W_INDEX TABLE------------------
W_INDEX:
DB 29,29,29,30,45,46,22,22,23,23,47,48,13,13,13,13
DB 20,20,33,34,35,36,37,38,19,19,31,32,01,01,01,01
DB  12,12,12,12,53,54,26,26,39,40,41,42,43,44,21,21
DB 28,28,61,62,63,00,10,10,10,10,10,10,10,10,10,10
DB 11,11,11,11,11,11,11,11,27,27,59,60,60,60,18,18
DB 24,24,49,50,51,52,25,25,55,56,57,58,58,58,58,58
DB 58,58,58,58,58,58,58,58,58,58,58,58,58,58,58,58
DB 02,02,02,02,02,02,02,02,02,02,02,02,02,02,02,02
DB 03,03,03,03,03,03,03,03,03,03,03,03,03,03,03,03
DB 03,03,03,03,03,03,03,03,08,08,08,08,08,08,08,08
DB 09,09,09,09,09,09,09,09,16,16,16,16,17,17,17,17
DB 04,04,04,04,04,04,04,04,04,04,04,04,04,04,04,04
DB 05,05,05,05,05,05,05,05,05,05,05,05,05,05,05,05
DB 14,14,14,14,15,15,15,15,15,15,15,15,15,15,15,15
DB 06,06,06,06,06,06,06,06,06,06,06,06,06,06,06,06
DB 07,07,07,07,07,07,07,07,07,07,07,07,07,07,07,07;--
--------------DEFIND W_DATA TABLE-------------------
W_DATA:
DB 00,08,01,06,02,04,03,04,04,04,05,04,06,04,07,04
DB 08,05,09,05,10,05,11,05,12,06,13,06,14,06,15,06
DB 16,06,17,06,18,07,19,07,20,07,21,07,22,07,23,07
DB 24,07,25,07,26,07,27,07,28,07,29,08,30,08,31,08
DB 32,08,33,08,34,08,35,08,36,08,37,08,38,08,39,08
DB 40,08,41,08,42,08,43,08,44,08,45,08,46,08,47,08
DB 48,08,49,08,50,08H,51,08,52,08,53,08,54,08,55,08
DB 56,08,57,08,58,08,59,08,60,08,61,08,62,08,63,08
;----------------DEFIND B_INDEX4B TABLE------------------
BWB_INDEX:
DW
BWB4_0000,BWB4_0001,BWB4_0010,BWB4_0011,B
WB4_0100,BWB4_0100,BWB4_0110,BWB4_0110
DW
BWB4_1000,BWB4_1000,BWB4_1000,BWB4_1000,B
WB4_1100,BWB4_1100,BWB4_1100,BWB4_1100
;----------------DEFIND B_INDEX4B TABLE------------------
BBW_INDEX:
DW
BBW4_0000,BBW4_0001,BBW4_0010,BBW4_0011,B
BW4_0100,BBW4_0100,BBW4_0110,BBW4_0110
DW
BBW4_1000,BBW4_1000,BBW4_1000,BBW4_1000,B
BW4_1100,BBW4_1100,BBW4_1100,BBW4_1100
;----------------DEFIND B_INDEX8B0 TABLE-----------------
B_INDEX8B0:
DB 18,18,18,18,18,18,18,18,18,18,18,18,18,18,18,18
DB 18,18,18,18,18,18,18,18,18,18,18,18,18,18,18,18
DB 18,18,18,18,52,52,52,55,56,56,56,59,60,60,24,24
DB 25,25,25,25,25,25,25,53,54,54,54,54,54,54,54,54
DB 13,13,13,13,13,13,13,13,13,13,13,13,13,13,13,13
DB 23,23,50,51,44,45,46,47,57,58,61,61,16,16,16,16
DB 17,17,17,17,48,49,62,63,30,31,32,33,40,41,22,22
DB 14,14,14,14,14,14,14,14,14,14,14,14,14,14,14,14
DB 10,10,10,10,10,10,10,10,10,10,10,10,10,10,10,10
DB 10,10,10,10,10,10,10,10,10,10,10,10,10,10,10,10
DB 11,11,11,11,11,11,11,11,11,11,11,11,11,11,11,11
DB 11,11,11,11,11,11,11,11,11,11,11,11,11,11,11,11
DB 15,15,15,15,15,15,15,15,15,15,26,27,28,29,19,19
DB 20,20,34,35,36,37,38,39,21,21,42,43,00,00,00,00
DB 12,12,12,12,12,12,12,12,12,12,12,12,12,12,12,12
DB 12,12,12,12,12,12,12,12,12,12,12,12,12,12,12,12
;----------------DEFIND B_INDEX8B1 TABLE-----------------
B_INDEX8B1:         DB 09,08,07,07
```

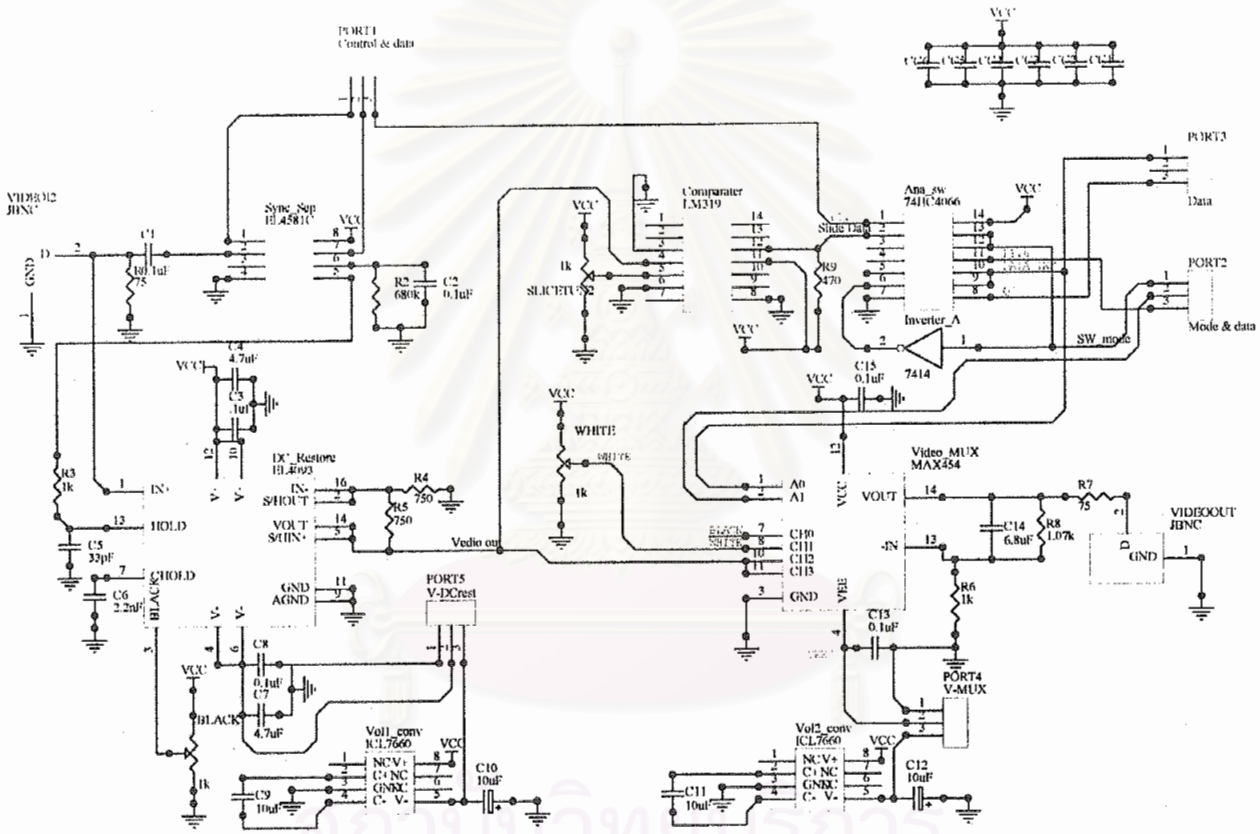
```
;----------------DEFIND B_DATA TABLE-------------------
B_DATA:
DB 00,06,01,03,02,02,03,02,04,03,05,04,06,04,07,01
DB 08,02,09,02,10,03,11,03,12,03,13,04,14,04,15,05
DB 16,06,17,06,18,06,19,07,20,07,21,07,22,07,23,07
DB 24,07,25,07,26,08,27,08,28,08,29,08,30,08,31,08
DB 32,08,33,08,31,08,35,08,36,08,37,08,38,08,39,08
DB 40,08,41,08,42,08,43,08,44,08,45,08,46,08,47,08
DB 48,08,49,08,50,08,51,08,52,08,53,08,54,08,55,08
DB 56,08,57,08,58,08,59,08,60,08,61,08,62,08,63,08
;---------------- DEFIND WHITE FOR WB TABLE -----------
SETWWB_TAB:
DW
WWB0,WWB1,WWB2,WWB3,WWB4,WWB5,WWB6,W
WB7
DW
WWB0,WWB1,WWB2,WWB3,WWB4,WWB5,WWB6,W
WB7
DW
WWB0,WWB1,WWB2,WWB3,WWB4,WWB5,WWB6,W
WB7
DW
WWB0,WWB1,WWB2,WWB3,WWB4,WWB5,WWB6,W
WB7
DW
WWB0,WWB1,WWB2,WWB3,WWB4,WWB5,WWB6,W
WB7
DW
WWB0,WWB1,WWB2,WWB3,WWB4,WWB5,WWB6,W
WB7
DW
WWB0,WWB1,WWB2,WWB3,WWB4,WWB5,WWB6,W
WB7
DW
WWB0,WWB1,WWB2,WWB3,WWB4,WWB5,WWB6,W
WB7
DW
WWB0,WWB1,WWB2,WWB3,WWB4,WWB5,WWB6,W
WB7
DW
WWB0,WWB1,WWB2,WWB3,WWB4,WWB5,WWB6,W
WB7
DW
WWB0,WWB1,WWB2,WWB3,WWB4,WWB5,WWB6,W
WB7
DW
WWB0,WWB1,WWB2,WWB3,WWB4,WWB5,WWB6,W
WB7
DW
WWB0,WWB1,WWB2,WWB3,WWB4,WWB5,WWB6,W
WB7
DW
WWB0,WWB1,WWB2,WWB3,WWB4,WWB5,WWB6,W
WB7
DW
WWB0,WWB1,WWB2,WWB3,WWB4,WWB5,WWB6,W
WB7
DW
WWB0,WWB1,WWB2,WWB3,WWB4,WWB5,WWB6,W
WB7
;------- DEFIND LESS THAN 8 BIT WHITE TABLE -------
LESS8_WWB:
DW
L8_WWB0,L8_WWB1,L8_WWB2,L8_WWB3,L8_WWB
4,L8_WWB5,L8_WWB6,L8_WWB7

LESS7_WWB:
DW
L7_WWB0,L7_WWB1,L7_WWB2,L7_WWB3,L7_WWB
4,L7_WWB5,L7_WWB6

LESS6_WWB:
DW
L6_WWB0,L6_WWB1,L6_WWB2,L6_WWB3,L6_WWB
4,L6_WWB5

LESS5_WWB:
DW
L5_WWB0,L5_WWB1,L5_WWB2,L5_WWB3,L5_WWB
4
;---------------- DEFIND BLACK FOR WB TABLE -----------
SETBWB_TAB:
DW
BWB0,BWB1,BWB2,BWB3,BWB4,BWB5,BWB6,BWB7
DW
BWB0,BWB1,BWB2,BWB3,BWB4,BWB5,BWB6,BWB7
DW
BWB0,BWB1,BWB2,BWB3,BWB4,BWB5,BWB6,BWB7
DW
BWB0,BWB1,BWB2,BWB3,BWB4,BWB5,BWB6,BWB7
DW
BWB0,BWB1,BWB2,BWB3,BWB4,BWB5,BWB6,BWB7
DW
BWB0,BWB1,BWB2,BWB3,BWB4,BWB5,BWB6,BWB7
DW
BWB0,BWB1,BWB2,BWB3,BWB4,BWB5,BWB6,BWB7
DW
BWB0,BWB1,BWB2,BWB3,BWB4,BWB5,BWB6,BWB7
DW
BWB0,BWB1,BWB2,BWB3,BWB4,BWB5,BWB6,BWB7
DW
BWB0,BWB1,BWB2,BWB3,BWB4,BWB5,BWB6,BWB7
DW
BWB0,BWB1,BWB2,BWB3,BWB4,BWB5,BWB6,BWB7
DW
BWB0,BWB1,BWB2,BWB3,BWB4,BWB5,BWB6,BWB7
DW
BWB0,BWB1,BWB2,BWB3,BWB4,BWB5,BWB6,BWB7
DW
BWB0,BWB1,BWB2,BWB3,BWB4,BWB5,BWB6,BWB7
DW
BWB0,BWB1,BWB2,BWB3,BWB4,BWB5,BWB6,BWB7
DW
BWB0,BWB1,BWB2,BWB3,BWB4,BWB5,BWB6,BWB7

;------- DEFIND LESS THAN 8 BIT BLACK TABLE ------
LESS8_BWB:
DW
L8_BWB0,L8_BWB1,L8_BWB2,L8_BWB3,L8_BWB4,L
8_BWB5,L8_BWB6,L8_BWB7

LESS7_BWB:
DW
L7_BWB0,L7_BWB1,L7_BWB2,L7_BWB3,L7_BWB4,L
7_BWB5,L7_BWB6

LESS6_BWB:
DW
L6_BWB0,L6_BWB1,L6_BWB2,L6_BWB3,L6_BWB4,L
6_BWB5

LESS5_BWB:
DW
L5_BWB0,L5_BWB1,L5_BWB2,L5_BWB3,L5_BWB4

;---------------- DEFIND WHITE FOR BW TABLE -----------
--------
SETWBW_TAB:
DW
WBW0,WBW1,WBW2,WBW3,WBW4,WBW5,WBW6,W
BW7
DW
WBW0,WBW1,WBW2,WBW3,WBW4,WBW5,WBW6,W
BW7
```

```
DW
WBW0,WBW1,WBW2,WBW3,WBW4,WBW5,WBW6,W
BW7
DW
WBW0,WBW1,WBW2,WBW3,WBW4,WBW5,WBW6,W
BW7
DW
WBW0,WBW1,WBW2,WBW3,WBW4,WBW5,WBW6,W
BW7
DW
WBW0,WBW1,WBW2,WBW3,WBW4,WBW5,WBW6,W
BW7
DW
WBW0,WBW1,WBW2,WBW3,WBW4,WBW5,WBW6,W
BW7
DW
WBW0,WBW1,WBW2,WBW3,WBW4,WBW5,WBW6,W
BW7
DW
WBW0,WBW1,WBW2,WBW3,WBW4,WBW5,WBW6,W
BW7
DW
WBW0,WBW1,WBW2,WBW3,WBW4,WBW5,WBW6,W
BW7
DW
WBW0,WBW1,WBW2,WBW3,WBW4,WBW5,WBW6,W
BW7
DW
WBW0,WBW1,WBW2,WBW3,WBW4,WBW5,WBW6,W
BW7
DW
WBW0,WBW1,WBW2,WBW3,WBW4,WBW5,WBW6,W
BW7
DW
WBW0,WBW1,WBW2,WBW3,WBW4,WBW5,WBW6,W
BW7
DW
WBW0,WBW1,WBW2,WBW3,WBW4,WBW5,WBW6,W
BW7
DW
WBW0,WBW1,WBW2,WBW3,WBW4,WBW5,WBW6,W
BW7

;------- DEFIND LESS THAN 8 BIT WHITE TABLE -------
LESS8_WBW:
DW
L8_WBW0,L8_WBW1,L8_WBW2,L8_WBW3,L8_WBW
4,L8_WBW5,L8_WBW6,L8_WBW7

LESS7_WBW:
DW
L7_WBW0,L7_WBW1,L7_WBW2,L7_WBW3,L7_WBW
4,L7_WBW5,L7_WBW6

LESS6_WBW:
DW
L6_WBW0,L6_WBW1,L6_WBW2,L6_WBW3,L6_WBW
4,L6_WBW5

LESS5_WBW:
DW
L5_WBW0,L5_WBW1,L5_WBW2,L5_WBW3,L5_WBW
4
;---------------- DEFIND BLACK FOR BW TABLE -----------
SETBBW_TAB:
DW
BBW0,BBW1,BBW2,BBW3,BBW4,BBW5,BBW6,BBW7
DW
BBW0,BBW1,BBW2,BBW3,BBW4,BBW5,BBW6,BBW7
DW
BBW0,BBW1,BBW2,BBW3,BBW4,BBW5,BBW6,BBW7
DW
BBW0,BBW1,BBW2,BBW3,BBW4,BBW5,BBW6,BBW7
DW
BBW0,BBW1,BBW2,BBW3,BBW4,BBW5,BBW6,BBW7
DW
BBW0,BBW1,BBW2,BBW3,BBW4,BBW5,BBW6,BBW7
DW
BBW0,BBW1,BBW2,BBW3,BBW4,BBW5,BBW6,BBW7
DW
BBW0,BBW1,BBW2,BBW3,BBW4,BBW5,BBW6,BBW7
DW
BBW0,BBW1,BBW2,BBW3,BBW4,BBW5,BBW6,BBW7
DW
BBW0,BBW1,BBW2,BBW3,BBW4,BBW5,BBW6,BBW7
DW
BBW0,BBW1,BBW2,BBW3,BBW4,BBW5,BBW6,BBW7
DW
BBW0,BBW1,BBW2,BBW3,BBW4,BBW5,BBW6,BBW7
DW
BBW0,BBW1,BBW2,BBW3,BBW4,BBW5,BBW6,BBW7
DW
BBW0,BBW1,BBW2,BBW3,BBW4,BBW5,BBW6,BBW7
DW
BBW0,BBW1,BBW2,BBW3,BBW4,BBW5,BBW6,BBW7
DW
BBW0,BBW1,BBW2,BBW3,BBW4,BBW5,BBW6,BBW7

;------- DEFIND LESS THAN 8 BIT BLACK TABLE -------
LESS8_BBW:
DW
L8_BBW0,L8_BBW1,L8_BBW2,L8_BBW3,L8_BBW4,L
8_BBW5,L8_BBW6,L8_BBW7

LESS7_BBW:
DW
L7_BBW0,L7_BBW1,L7_BBW2,L7_BBW3,L7_BBW4,L
7_BBW5,L7_BBW6

LESS6_BBW:
DW
L6_BBW0,L6_BBW1,L6_BBW2,L6_BBW3,L6_BBW4,L
6_BBW5

LESS5_BBW:
DW
L5_BBW0,L5_BBW1,L5_BBW2,L5_BBW3,L5_BBW4

                        END
```

ภาคผนวก จ

รายละเอียดแผนภาพวงจรบอร์ดต้นแบบของเครื่องแทรกข้อมูลภาพ และเครื่องรับข้อมูลภาพ


PORT1
3PIN
COMP_SYNC
ODD_EVEN
P1_7
PORT2
3PIN
SW_MODE
SW_2
P1_6
PORT3
3PIN
VCC
Q7
VCC
SW3
C22
RST
D7
R4
X2
C14
0.0022
L3
4.6 uH
C23
C24
CPU
P1_0
P1_1
P1_2
P1_3
P1_4
P1_5
P1_6
P1_7
P10
P11
P12
P13
P14
P15
P16
P17
SW_MODE
COMP_SYNC
INT1
INT0
SW_2
ODD_EVEN
T1
T0
EA/VP
X1
X2
RST
RESET
RD
WR
COUNTER
74191
P0
P1
P2
P3
Q0
Q1
Q2
Q3
CLK
CE
U/D
PL
RC
TC
COMP_SYNC
NANDA
74132
TUNER
clk TUNER
C1
33pf
PL
CLK2
CLK1
P7
P6
P5
P4
P3
P2
P1
P0
SER
SHIFT
74165
Q7
Q7
MAX232
DIP16
C1+
C21
4.7uF
C1-
C2+
C20
4.7uF
C2-
TX
RX
TXH1
RXH1
TXH2
RXH2
LATCH
74HC573
ALE
OC
C
P00
P01
P02
P03
P04
P05
P06
P07
1D
2D
3D
4D
5D
6D
7D
8D
P20
P21
P22
P23
P24
P25
P26
P27
RAM
62256
1Q
2Q
3Q
4Q
5Q
6Q
7Q
8Q
A0
A1
A2
A3
A4
A5
A6
A7
A8
A9
A10
A11
A12
A13
A14
RD
WR
A15
OE
WE
CS
D0
D1
D2
D3
D4
D5
D6
D7
RBUSCPU
RBUS9
RXD
TXD
ALE/P
PSEN
C19
10uF
VCC
C17
10uF
C18
10uF
V+
V-
TXL1
RXL1
TXL2
RXL2
256_512
3PIN
EPP_EEP
3PIN
CODE
274001
VPP
_OE
_CS1
PSEN
SERIALPORT
3PIN
RX
TX
Title
Size
A4
Number
Revision
Date:
9-Sep-2000
Sheet of
File:
D:\THESIS\BOARD\PCBB&SCH\CDSEG80.SCH



Title

Size	Number	Revision
B		
Date:	9-Sep-2000	Sheet of
File:	D:\THESIS\BOARD\PCB\SCH\MAIN.SCH	

ประวัติผู้เขียนวิทยานิพนธ์

นายนันทกฤษณ์ วัฒนเลี้ยงใจ เกิดวันที่ 4 ธันวาคม พ.ศ. 2519 ที่กรุงเทพมหานคร สำเร็จการศึกษาปริญญาตรีวิศวกรรมศาสตรบัณฑิต สาขาวิศวกรรมไฟฟ้า จากจุฬาลงกรณ์มหาวิทยาลัย ในปีการศึกษา 2540 และเข้าศึกษาต่อในหลักสูตรวิศวกรรมศาสตรมหาบัณฑิต สาขาวิศวกรรมไฟฟ้า ที่จุฬาลงกรณ์มหาวิทยาลัยเมื่อ พ.ศ. 2540 ในระหว่างการศึกษาระดับมหาบัณฑิตได้รับทุนผู้ช่วยสอนจากบัณฑิตวิทยาลัยจุฬาลงกรณ์มหาวิทยาลัย