เครื่องเข้ารหัสสัญญาณเพจโดยใช้แบบ POCSAG CODE

ENCODER PAGER FOR POCSAG CODE

โดย

นายกิตติวัชร์ ทิพากร 39012041

นายวสันต์ แย้มดี 39012065

ปริญญานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาอุตสาหกรรมศาสตรบัณฑิต

สาขาเทคโนโลยีโทรคมนาคม

คณะวิศวกรรมศาสตร์

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

ปีการศึกษา 2540

หัวข้อปริญญานิพนธ์ เครื่องเข้ารหัสสัญญาณเพจเจอร์แบบ POCSAG CODE

ENCODER PAGER FOR POCSAG CODE

โดย

นาย กิตติวัชร์ ทิพากร 39012041

นาย วสันต์ แย้มดี 39012065

อาจารย์ที่ปรึกษา อ.กฤดากร กล่อมการ

ภาควิชา เทคนิคอุตสาหกรรม

ปีการศึกษา 2540

คณะวิศวกรรมศาสตร์ สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง อนุมัติให้นับเป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาอุตสาหกรรมศาสตรบัณฑิต

คณะกรรมการสอบปริญญานิพนธ์

..... ประธานกรรมการ

( )

..... กรรมการ

( )

..... กรรมการ

( )

..... กรรมการ

( )

..... กรรมการ

( )

เครื่องเข้ารหัสสัญญาณเพจเจอร์แบบ POCSAG CODE

โดย นาย กิตติวัชร์ ทิพากร 39012041

นาย วสันต์ แย้มดี 39012065

อาจารย์ที่ปรึกษา อ. กฤดากร กล่อมการ

ปีการศึกษา 2540

บทคัดย่อ

ปริญญานิพนธ์ ชุดนี้ประกอบด้วยรายละเอียดของการศึกษา และวิจัยเครื่องส่งเพจเจอร์ รวมทั้งการสร้างเลียนแบบเครื่องส่ง ที่ใช้งานได้กันอย่างแพร่หลายในปัจจุบัน ตัวโครงงานนี้ทำเป็นการ์ดคอมพิวเตอร์ทำหน้าที่เป็นตัวส่งสัญญาณเพจเจอร์ การทำงานของเครื่องจะอาศัยทั้งชุดวงจร และโปรแกรมคอมพิวเตอร์ที่สร้างด้วยภาษา C โดยตัววงจรจะทำหน้าที่จัดเรียงข้อมูลเพื่อให้สามารถเก็บไว้ใน RAM EMUNILATOR ซึ่งเป็นที่พักข้อมูล และสร้างสัญญาณบอกให้แก่คอมพิวเตอร์รับข้อมูลเหล่านั้น ไปวิเคราะห์และตีความจาก POCSAG (โปรโตรคอลของสัญญาณเพจเจอร์) เป็นข่าวสารที่เข้าใจได้โดยโปรแกรมที่สร้างขึ้น

ENCODER PAGER FOR POCSAG CODE

BY MR.KITTIWAT TIPAKORN 39012041
MR.WASAN YAMDEE 39012065

ADVISOR MR.KITDAKORN KLOMKRAN

YEAR 1997

ABSTRACT

The thesis concerns about study and research description of pager transmitter as well as detail of simulating such pager transmitter used widely today. The project is a computer card used to sent the pager data temporarily. The project consists of hardware, which reorders data, and software, which is created with C language.

กิตติกรรมประกาศ

ปริญญานิพนธ์ฉบับนี้สำเร็จลุล่วงด้วยดี ข้าพเจ้าผู้จัดทำต้องขอขอบพระคุณท่านอาจารย์ กฤดากร กล่อมการ เป็นอย่างยิ่ง ซึ่งเป็นอาจารย์ที่ปรึกษา ที่ได้ประสิทธิประสาทวิชาให้คำแนะนำ และคำปรึกษาเรื่องต่างๆ พร้อมทั้งสนับสนุนห้องทดลอง และเครื่องมือที่ใช้ในการทดลองต่างๆมา โดยตลอด และที่ขาดไม่ได้ก็คือ สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบังศูนย์ นนทบุรี ซึ่งเป็นที่ปฏิบัติงานของผู้จัดทำ

ขอขอบคุณ คุณอุดมพร สุนันทชัยกุล สำหรับการสละเวลาเพื่อให้คำแนะนำและคำปรึกษา ให้แก่ผู้จัดทำมาโดยตลอด

คุณความดีใดๆที่เกิดจากปริญญานิพนธ์ฉบับนี้ ขอมอบแด่ บิดา มารดา และครูบาอาจารย์ ที่มีพระคุณกับผู้จัดทำตลอดมา

คณะผู้จัดทำ

สารบัญ

สารบัญรูปภาพ

สารบัญตาราง

บทที่ 1

บทนำ

ทฤษฎีทั่วไปของวิทยุติดตามตัว (Pager)

ในระบบการติดต่อสื่อสารจากอดีตถึงปัจจุบัน ได้มีการพัฒนาอุปกรณ์ที่ใช้ในการสื่อสารมากมายหลายชนิด ตั้งแต่อุปกรณ์แบบมีสายจนกระทั่งพัฒนาเป็นแบบไร้สาย และจากเจริญก้าวหน้าทางเทคโนโลยี ก็ได้มีการพัฒนาอุปกรณ์เหล่านี้ เพื่ออำนวยความสะดวกแก่ผู้ใช้มากขึ้น ไม่ว่าจะเป็นการเพิ่มรัศมีในการติดต่อสื่อสาร หรือการลดขนาดเพื่อให้เล็กกระทัดรัด

วิทยุติดตามตัวหรือเพจเจอร์ (Pager) เป็นวิทยุเครื่องมือสื่อสารชนิดไร้สายแบบเคลื่อนที่แบบหนึ่ง เป็นที่นิยมสำหรับผู้ใช้ที่มีภาระหน้าที่ไม่ค่อยอยู่ประจำที่ แต่จำเป็นต้องมีการติดต่อสื่อสารได้อย่างตลอดเวลา เช่นเดียวกับวิทยุโทรศัพท์แบบรวงผึ้ง แต่จะมีลักษณะการสื่อสารในทิศทางเดียว คือจากผู้ส่งไปยังผู้รับ

การติดต่อระหว่างผู้ส่งกับผู้รับ อาจจะเป็นการแจ้งให้ผู้รับติดต่อกลับไปยังผู้ส่ง หรืออาจจะเป็นการส่งข่าวสารที่ต้องการไปยังผู้รับโดยตรงก็ได้ โดย เพจเจอร์จะมีการแบ่งออกเป็น 5 แบบตามลักษณะคือ

1. Voice Pager จะทำการส่งข่าวสารแบบเสียงพูดไปยังผู้รับ

2. Digital Display Pager จะทำการส่งข่าวสารเป็นตัวเลข ซึ่งตัวเลขนี้จะแทนข่าวสารที่มีความหมายเป็นที่เข้าใจระหว่างผู้ส่งกับผู้รับ

3. Alpha Numerlic Pager แบบนี้สามารถที่จะทำการส่งข่าวสารเป็นได้ทั้งตัวเลขและตัวอักษร ซึ่งเป็นการช่วยส่งข่าวสารได้ละเอียดขึ้น แต่จำเป็นต้องใช้ระบบที่พิเศษเพิ่มมากขึ้นมาอีก

4. Tone-Alert Pager เป็นการส่งสัญญาณเสียง เพื่อเป็นการบอกให้ผู้รับติดต่อไปยังศูนย์เพื่อรับข่าวสารอีกทีหนึ่ง

5. Dual Address Pager จะส่งสัญญาณเสียงได้ 2 ลักษณะเพื่อให้ผู้รับทราบว่าจะติดต่อไปยังที่ใด

เวลาที่ใช้ในการส่งข่าวสารด้วยเพจเจอร์ในแต่ละแบบ จะแตกต่างกันคือเพจเจอร์แบบใช้เสียงพูดที่ ใช้เวลาที่ตั้งแต่ 10 วินาทีลงมา ส่วนเพจเจอร์แบบที่ไม่ใช้เสียงพูด จะใช้เวลาการส่งเป็นมิลลิวินาที

ข้อดีของเพจเจอร์เมื่อเปรียบเทียบกับระบบการสื่อสารเคลื่อนที่ชนิด 2 ทิศทางคือ

1. สามารถพกติดตัวได้สะดวก เนื่องจากขนาดเล็กลงและมีน้ำหนักเบา และไม่ต้องมีแหล่งจ่ายไฟภายนอก
2. ค่าใช้จ่ายในการเช่าใช้ถูกกว่ามาก
3. มีจำนวนผู้เช่าใช้ต่อหนึ่งความถี่ที่มากกว่า

4. เครื่องรับสามารถใช้ร่วมกันได้โดยส่วนใหญ่จากหลายๆยี่ห้อ

5. ที่ตัวเพจเจอร์แทบไม่ต้องมีการบำรุงรักษา

6. ไม่ต้องมีการติดตั้งใดๆในส่วนของผู้ใช้

7. สามารถบันทึกข่าวสารไว้ได้ในตัวเครื่อง

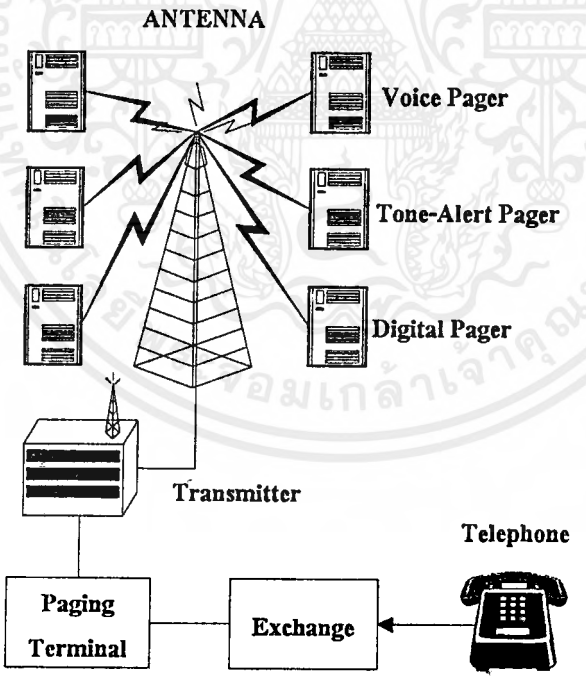
1.1 โครงสร้างของระบบเพจเจอร์

เพจเจอร์ใช้การสื่อสารด้วยวิทยุเช่นเดียวกับระบบวิทยุโทรศัพท์ ดังนั้นโครงสร้างของระบบจึงไม่แตกต่างกันนักดังแสดงในรูปที่ 1.1 ซึ่งประกอบด้วยส่วนประกอบๆ ดังนี้

1. ระบบโทรศัพท์ เป็นส่วนที่ผู้ส่งสามารถส่งข่าวสารผ่านทางโทรศัพท์ไปยังศูนย์ เพื่อส่งต่อไปยังผู้รับที่ต้องการ

2. ศูนย์กลางของระบบ (Pager Terminal) เป็นศูนย์กลางในการติดต่อสื่อสาร โดยมีคอมพิวเตอร์เป็นผู้จัดการทุกอย่าง เกี่ยวกับการให้บริการการส่งข่าวสาร จากผู้ส่งไปยังผู้รับ ซึ่งถือว่าเป็นส่วนสำคัญของระบบ

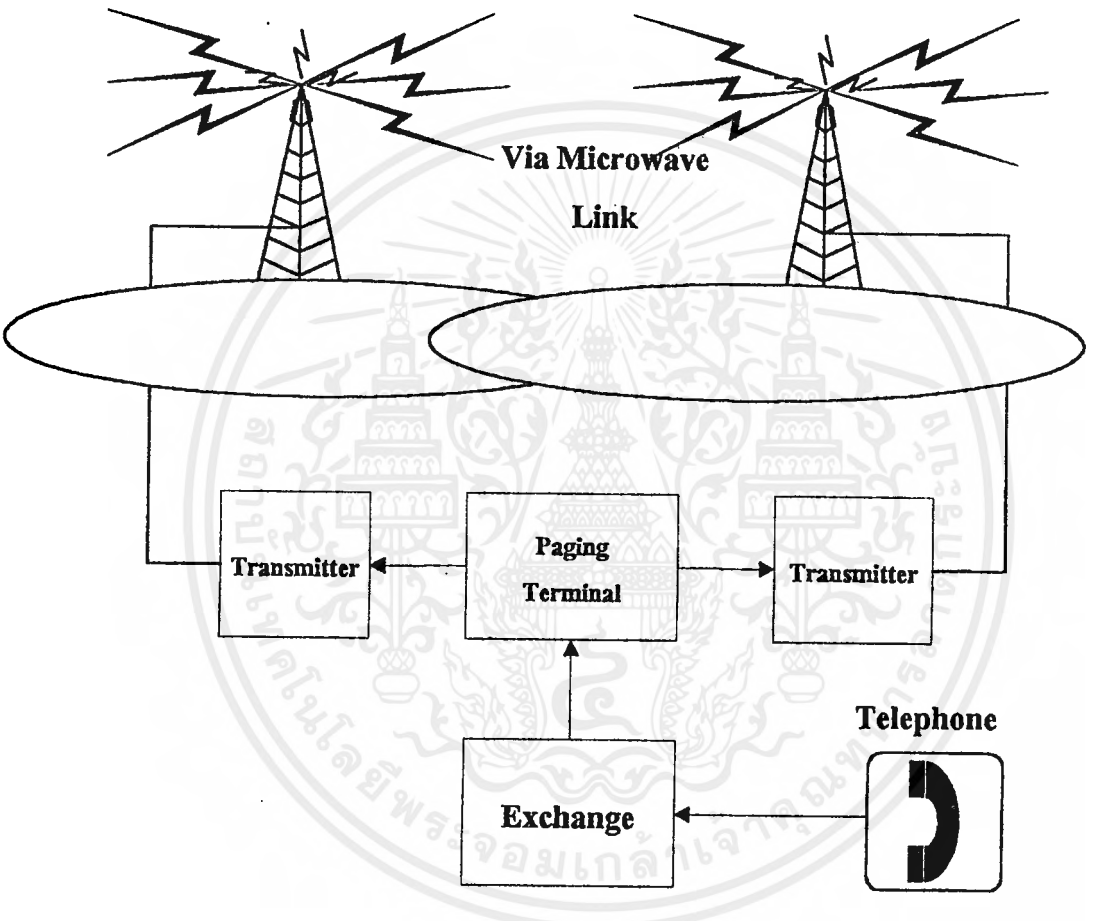
3. เครื่องส่ง ทำหน้าที่ส่งข่าวสารในรูปของคลื่นวิทยุผ่านสายอากาศไปยังเครื่องรับ ซึ่งเครื่องส่งอาจตั้งอยู่ที่ศูนย์ หรือที่อื่นก็ได้



รูปที่ 1.1 โครงสร้างของวิทยุติดตามตัว

1.2 การขยายพื้นที่ของระบบ

เพื่อให้การบริการแก่ผู้ใช้กว้างออกไป จำเป็นต้องเพิ่มสถานีส่งกระจายออกไปตามจุดต่างๆ ให้ครอบคลุมอาณาบริเวณที่ต้องการ โดยความถี่ที่ส่งออกไปในเครื่องส่งแต่ละเครื่องยังคงเป็นความถี่เดิม ส่วนข่าวสารจะเป็นการติดต่อถึงกันด้วยคลื่นความถี่วิทยุอีกความถี่หนึ่ง ซึ่งใช้ระบบไมโครเวฟ ปัญหาในเรื่องความถี่เครื่องส่งที่ความถี่เดียวกัน ในพื้นที่เหลี่ยมกันนั้นจะใช้วิธีปรับเฟสของคลื่นที่ส่งออกไปให้มีเฟสตรงกัน เมื่อทำตามระบบดังกล่าวจะได้รูปแบบการวางเสาส่งของระบบดังรูปที่ 1.2



รูปที่ 1.2 ลักษณะการขยายพื้นที่ของระบบ

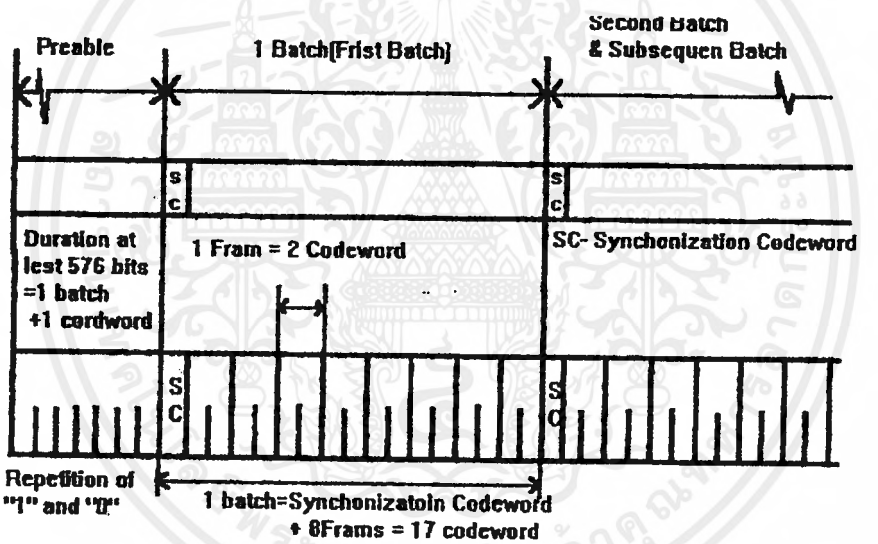
บทที่ 2

POCSAG

(Post Office Code Standardization Advisory Group)

2.1 รูปแบบของสัญญาณเพจเจอร์

สัญญาณข่าวสารที่จะส่งไปยังผู้รับนั้น เป็นสัญญาณรหัสข้อมูลเลขฐาน 2 เรียงต่อเนื่องกันไป ซึ่งรหัสข้อมูลที่ใช้เป็นรหัสของ POCSAG (Post Office Code Standardization Advisory Group) ซึ่งเป็นรูปแบบ และรหัสมาตรฐานสำหรับวิทยุติดตามตัวที่ถูกกำหนดโดย CCIR รูปแบบของสัญญาณข่าวสารที่ใช้ส่งประกอบไปด้วยสัญญาณพัลส์ส่วนหน้า (Preamble) จำนวน 576 บิท และ ตามด้วยรหัสคำตั้งแต่ 1 ชุดขึ้นไป (Batch Structure) ซึ่งแต่ละชุดจะเริ่มต้นด้วยรหัสการซิงโครไนท์ (Synchronous) รูปแบบสัญญาณดังรูปที่ 2.1



รูปที่ 2.1 รูปแบบของสัญญาณต่างๆ

2.2 สัญญาณพัลส์ส่วนหน้า (Preamble)

ประกอบด้วยรูปแบบที่มีลักษณะของบิทตรงข้ามคือ " 1010.....1010 " จะถูกส่งเป็นจำนวนอย่างน้อย 576 บิท (มีค่าเท่ากับ 1 Batch บวกกับอีก 1 Codeword) สัญญาณพัลส์ส่วนหน้านี้ ถูกใช้เป็นตัวช่วยในการตรวจหาจุดเริ่มต้นของการส่งเพจเจอร์

2.3 โครงสร้างของแบทช์ข้อมูล (Batch Structure)

รหัสคำจะถูกส่งลงแบทช์โดยที่เฟรมแรกจะเป็นเฟรมที่ 0 และเฟรมสุดท้ายจะเป็นเฟรมที่ 7 รหัสคำที่บรรจุอยู่ในแบทช์ข้อมูลแบ่งออกเป็น 4 ชนิดดังนี้

1. รหัสคำการซิงโครไนท์ (Synchronous Codeword)
2. รหัสคำของหมายเลขเรียกขาน (Address Codeword)
3. รหัสคำของข่าวสาร (Message Codeword)
4. รหัสคำเทียม (Idel Codeword)

รหัสคำทั้ง 4 ชนิดนี้แต่ละชนิดประกอบด้วยเลขฐาน 2 จำนวน 32 บิท แต่ละชนิดมีรูปแบบซึ่งจะแสดงไว้ในดังตารางที่ 2.1

BIT NO.	SYCHRONOUS CODEWORD	ADDRESS CODEWORD	MESSAGE CODEWORD	IDEL CODEWORD
01	0	Address flag = 0	Message flag = 1	0
02	1	Address Bit	Message Bit	1
03	1	Address Bit	Message Bit	1
04	1	Address Bit	Message Bit	1
05	1	Address Bit	Message Bit	1
06	1	Address Bit	Message Bit	0
07	0	Address Bit	Message Bit	1
08	0	Address Bit	Message Bit	0
09	1	Address Bit	Message Bit	1
10	1	Address Bit	Message Bit	0
11	0	Address Bit	Message Bit	0
12	1	Address Bit	Message Bit	0
13	0	Address Bit	Message Bit	1
14	0	Address Bit	Message Bit	0
15	1	Address Bit	Message Bit	0
16	0	Address Bit	Message Bit	1

BIT NO.	SYNCHRONOUS CODEWORD	ADDRESS CODEWORD	MESSSAGE CODEWORD	IDEL CODEWORD
17	0	Address Bit	Message Bit	1
18	0	Address Bit	Message Bit	1
19	0	Address Bit	Message Bit	0
20	1	Function Bit	Message Bit	0
21	0	Function Bit	Message Bit	0
22	1	Check Bit	Check Bit	0
23	0	Check Bit	Check Bit	0
24	1	Check Bit	Check Bit	1
25	1	Check Bit	Check Bit	1
26	1	Check Bit	Check Bit	0
27	0	Check Bit	Check Bit	0
28	1	Check Bit	Check Bit	1
29	1	Check Bit	Check Bit	0
30	0	Check Bit	Check Bit	1
31	0	Check Bit	Check Bit	1
32	0	Check Bit	Check Bit	1

ตารางรูปที่ 2.1 รูปแบบของรหัสคำ

2.4 รหัสคำการซิงโครไนท์ (Synchronous Codeword)

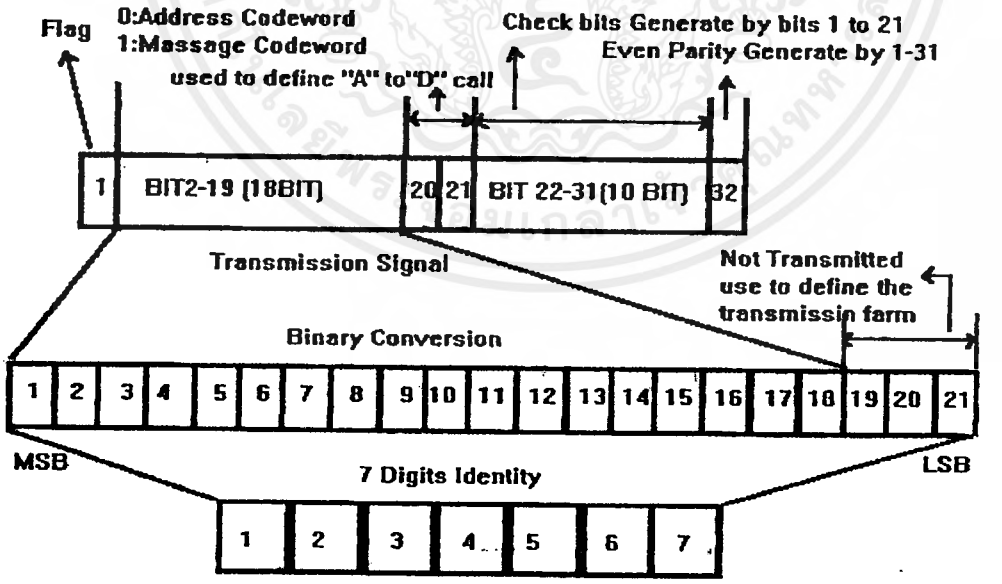
เป็นรหัสคำที่มีรูปแบบเฉพาะเป็นเอกลักษณ์ และไม่มีการซ้ำกับรหัสคำอื่นๆ รหัสคำชนิดนี้จะอยู่เป็นอันดับแรกของแต่ละแบทช์ข้อมูล และทำหน้าที่กำหนดจุดเริ่มต้นของแบทช์ข้อมูลด้วย โดยมีรูปแบบของบิทในตารางที่ 2.2

BIT NO	1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2
BIT	0 1 1 1 1 1 0 0 1 1 0 1 0 0 1 0 0 0 0 1 0 1 0 1 1 1 0 1 1 0 0 0

ตารางที่ 2.2 รูปแบบของบิทของรหัสคำการซิงโครไนท์

2.5 รหัสคำของหมายเลขเรียกขาน (Address Codeword)

รหัสคำชนิดนี้จะประกอบด้วยตัวเลขฐาน 2 จำนวน 32 บิท ซึ่งบิทแรกจะเป็น " 0 " เสมอ รูปแบบของรหัสนี้ถูกแสดงไว้ในตารางที่ 2.1 บิทที่ 2 ถึง บิทที่ 19 เป็นรหัสหมายเลขเฉพาะที่กำหนดให้เพจเจอร์แต่ละเครื่อง ซึ่งจะถูกแปลงจากเลขฐาน 10 จำนวน 7 หลักเป็นเลขฐาน 2 (BCD) จำนวน 21 บิท ซึ่งเลขฐาน 2 จะใช้เพียงแค่ 18 บิทที่มีความสำคัญมาก (MSB) ส่งไปกับรหัสของหมายเลขเรียกขาน ส่วนอีก 3 บิทที่มีความสำคัญน้อย (LSB) จะไม่ถูกส่งไปด้วย แต่จะถูกส่งใช้ในการกำหนดว่ารหัสหมายเลขเรียกขานควรจะอยู่ที่ใดในเฟรมไหนของแบทช์ ตัวอย่างเช่น ต้องการส่งข่าวสารให้กับเครื่องที่มีหมายเลขเรียกขาน 0210570 ก่อนอื่นจะต้องแปลงเลขฐาน 10 ดังกล่าวเป็นเลขฐาน 2 ดังตารางที่ 2.3 ซึ่งมีทั้งหมด 21 บิทดังนี้ " 000110011011010001010 " แต่ในการส่งจริงจะใช้เพียงแค่ 18 บิทที่มีความสำคัญมากเท่านั้น ส่วน " 010 " ที่เหลือจะเป็นตัวกำหนดหมายเลขของเฟรม ที่จะส่งรหัสหมายเลขเรียกขานชุดนี้ออกไป จากตัวอย่างนี้ $010_2 = 2_{10}$ ดังนั้นรหัสคำนี้จะอยู่ในเฟรมที่ 2 ของแบทช์ ดังนั้นเครื่องรับเพจเจอร์ที่มีหมายเลขเรียกขาน 021570 ก็จะมองหารหัสคำหมายเลขเรียกขานเฉพาะเฟรมที่ 2 ของแบทช์เท่านั้น วิธีการส่งลักษณะนี้จะช่วยลดจำนวนบิทของหมายเลขเรียกขานให้น้อยลงไปได้ถึง 3 บิท และถ้ามีบิทใดใน 3 บิทนี้เกิดการผิดพลาดไปในการรับส่งข้อมูล ก็จะไม่มีผลกระทบใดๆต่อรหัสหมายเลขเรียกขานเลย จึงช่วยเพิ่มประสิทธิภาพของเครื่องรับให้ดีขึ้นด้วย



รูปที่ 2.2 รูปแบบของรหัสหมายเลขเรียกขาน

2.6 รหัสคำของข่าวสาร (Message Codeword)

รหัสคำของข่าวสารเพจเจอร์รุ่นตัวเลข

รหัสคำของข่าวสารประกอบด้วยเลขฐาน 2 จำนวน 32 บิท โดยบิทแรกจะเป็น " 1 " เสมอ และรหัสคำนี้จะตามหลังรหัสของหมายเลขเรียกขาน รหัสคำของข่าวสารจะสิ้นสุดลง เมื่อมีการส่วนรหัสคำของหมายเลขเรียกขานตัวต่อไป หรือมีการส่งรหัสคำเทียมหรือเมื่อการส่งรหัสได้มีการสิ้นสุดลง แต่ถ้าข่าวสารที่มีการส่งยาวเลยแบทช์นั้นขึ้นไป ซึ่งนั้นก็หมายความว่าจะถูกแทรกด้วยรหัสคำซิงโครไนท์ ก่อนแล้วข่าวสารส่วนที่เหลือจึงจะถูกส่งในแบทช์ต่อไปจนหมด โดยไม่ต้องเริ่มด้วยรหัสคำหมายเลขเรียกขานอีก ข่าวสารในวิทยุติดตามตัวรุ่นตัวเลขจะมีทั้งข่าวสารที่เป็นตัวเลขฐาน 10 , เครื่องหมาย hyphen (-) , เครื่องหมาย [,] , ช่องว่าง (spacc) และ สัญลักษณ์ urgency " u " ซึ่งข้อมูลที่มีการแสดงออกมาที่หน้าจอของวิทยุติดตามตัวก็จะใช้รูปแบบนี้ รูปแบบของข่าวสารนี้จะใช้เลขฐาน 2 จำนวน 4 บิทแทน 1 อักขระ ชุดอักขระสำหรับข่าวสารถูกแสดงในตารางที่ 2.3 การส่งบิทของข่าวสารของอักขระแต่ละตัวจะส่งตามลำดับ โดนเริ่มจากบิทที่ 1 ซึ่งเป็นบิทที่มีความสำคัญน้อยที่สุดก่อนแล้วจึงตามด้วยบิทที่ 2 , 3 , 4 ตามลำดับอักขระตัวต่อไปก็จะถูกส่งตามจนครบ 5 ตัว หรือ 20 บิทต่อ 1 รหัสคำข่าวสาร

4- bit Combination Bit No. 4321	Displayed Character
0000	0
0001	1
0010	2
0011	3
0100	4
0101	5
0110	6
0111	7
1000	8
1001	9
0101	-
0110	U(Urgency Ondicator)

1100	Space
1101	Hyphen
1110	]
1111	[

ตารางที่ 2.3 ชุดอักขระของข่าวสาร

2.7 รหัสคำเทียม (Idel Codeword)

ถ้าข่าวสารที่ส่งมาหมดลงก่อนการสิ้นสุดเฟรม รหัสคำเทียมนี้จะถูกส่งแทนรหัสคำปกติ เพื่อให้เฟรมนั้นมีครบ 2 รหัสคำ (64 บิท) และในกรณีที่ข่าวสารที่จะส่งหมดลงก่อนที่จะสิ้นสุดแบทช์ รหัสคำเทียมก็จะได้รับการใส่ไว้เพื่อให้ครบเฟรมเดียวกัน รูปแบบของรหัสคำเทียมแสดงไว้ในตารางที่ 2.5

BIT NO	1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2
BIT	0 1 1 1 1 0 1 0 1 0 0 0 1 0 0 1 1 1 0 0 0 0 0 1 1 0 0 1 0 1 1 1

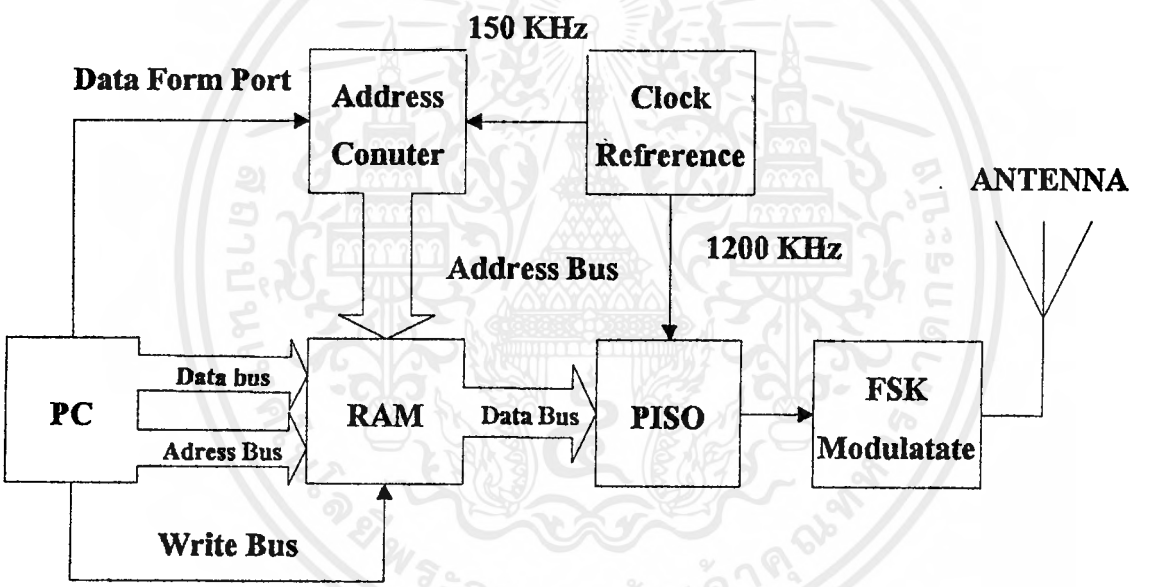
ตารางที่ 2.4 รูปแบบของรหัสคำเทียม

บทที่ 3

ภาคส่งสัญญาณเพจเจอร์

ภาคส่งสัญญาณเพจเจอร์ประกอบด้วยส่วนต่างๆ แสดงเป็นบล็อกไดอะแกรมดังรูปที่ 3.1

1. PISO
2. ภาคอ้างอิงสัญญาณนาฬิกา (Reference Clock)
3. ภาคนับแอสเดรส (Address Counter)
4. ภาคเลือก PORT และ Enable Sent (Port Select and Enable Sent)
5. RAM



รูปที่ 3.1 บล็อกไดอะแกรมหลักการทำงานของการส่งสัญญาณเพจเจอร์

3.1 การทำงานของ Pager Terminal Card

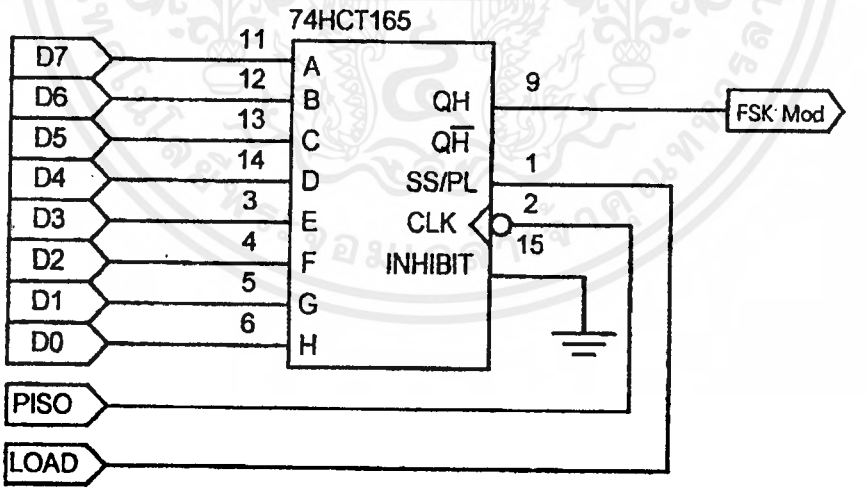
เมื่อข้อมูลของ Pager ที่พร้อมจะทำการส่ง จะถูกเขียนลงใน RAM ในตำแหน่ง DE00: ด้วยคำสั่งทาง Solfware จากนั้นข้อมูลที่จะส่งก็จะอยู่ใน RAM พร้อมที่จะถูกส่งออกไปได้ตลอดเวลา ซึ่งการส่งออกไปจะใช้คำสั่งจาก Solfware เช่นเดียวกัน

เมื่อทำการส่งให้ Pager Card ทำการส่งข้อมูล คอมพิวเตอร์จะส่งข้อมูลไปสั่งให้วงจรภาค Address Counter ให้เริ่มนับ (Access Memory) ทาง Port หมายเลข $3FO_H$ ให้เริ่มนับ (Access

Memory) ทาง Port หมายเลข $3F0_H$ ให้เริ่มนับ Access Memory จากนั้น วงจรดังกล่าวจะเริ่มทำงาน ส่งค่าAddress ผ่าน Address Bus เข้าสู่ RAM หลังจากนั้น RAM ก็จะส่งข้อมูลขนาด 8 บิท ตาม Address นั้นๆผ่านทาง data bus ไปส่งวงจรภาค PISO ทำให้วงจรภาค PISO ทำงานและส่งข้อมูลออกทีละบิท ทางขาเอาท์พุทจนครบ 8 บิท Address Counter ก็จะเพิ่มค่า Address ให้แก่วงจรภาค RAM อีก 1 Address และ RAM ก็จะส่งข้อมูลต่อไปให้กับ PISO อีก จนกระทั่งครบทุก Address แล้ว Address Counter ก็จะทำการ RESET ตัวเอง และก็จะหยุดนับจนกว่าจะมีข้อมูลใหม่เข้ามาอีกครั้ง

จากการทำงานดังกล่าวจะต้องอาศัยวงจร Reference Clock ควบคุมให้แก่วงจรทุกภาคทำงานให้มีการทำงานที่พร้อมเพรียงกันอย่างมี STEP จาก Block Diagram จะเห็นว่า Reference Clock จะสร้างความถี่ขึ้นมา 2 ความถี่ คือ 150 HZ กับ 1200 HZ เพราะเหตุผลดังนี้ คือ การส่งระบบ POSAG นั้นมาตรฐานในการส่งข้อมูลความเร็วในระดับ 1200 HZ ดังนั้นจึงต้องสร้าง Clock ส่งให้แก่ภาค PISO เพื่อที่จะทยอยส่งข้อมูลออกไปให้ได้ตามมาตรฐานดังกล่าว และที่ต้องสร้างความถี่ 150 HZ ขึ้นมาด้วยก็เพราะว่าภาค Address Counter จะต้องเพิ่ม Address ขึ้นมา 1 Address หลังจากที่ PISO ทำการส่งข้อมูลครบ 8 บิทเรียบร้อยแล้ว ดังนั้นความถี่ Clock ที่จะควบคุมภาค Address Counter นั้นจะต้องถูกหาร 8 ลงจาก 1200 HZ เหลือ 150 HZ เพื่อให้การทำงานของระบบเป็นไปโดยพร้อมเพรียงกัน

3.2 PISO (Parallal In Serie Out)



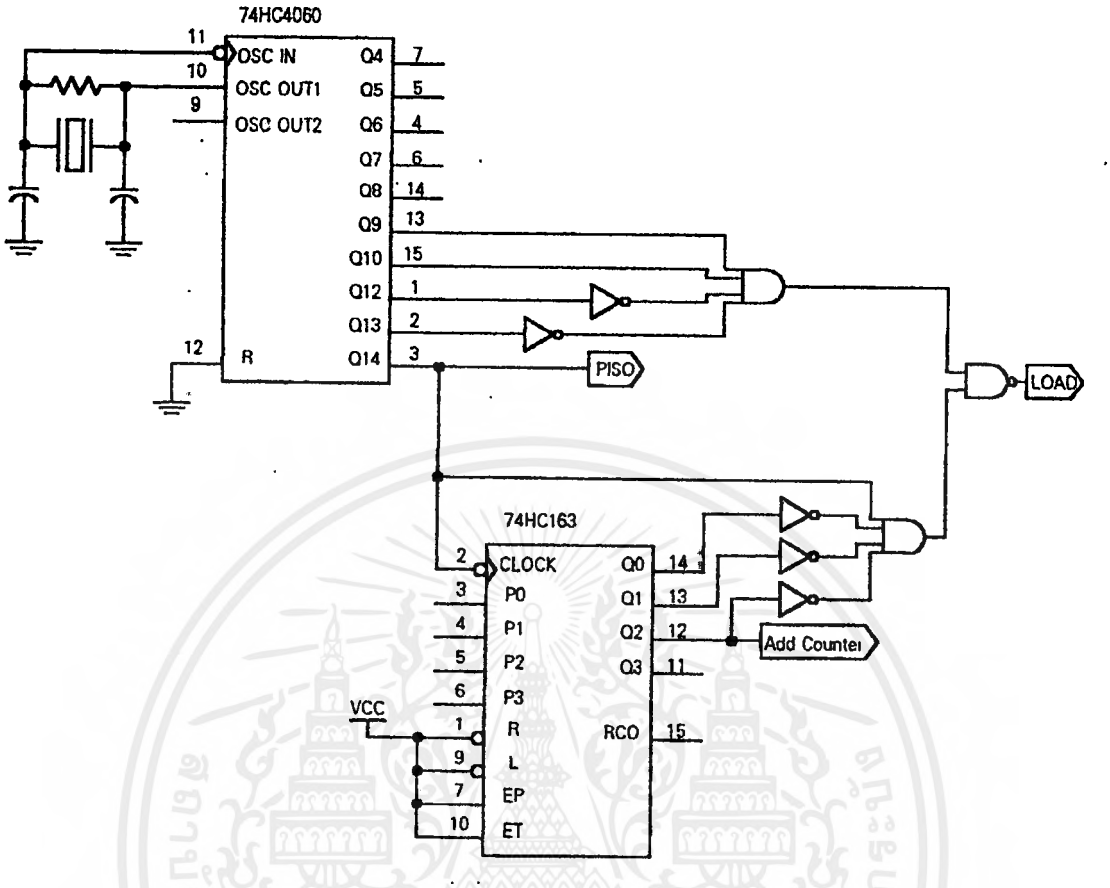
รูปที่ 3.2 วงจร PISO

เนื่องจากข้อมูลนั้นที่ส่งมาเป็นข้อมูลแบบขนาน แต่ในการที่จะส่งไปยังเครื่องรับจะต้องเป็นข้อมูลอนุกรมเท่านั้น (เพื่อที่จะนำไป Mod กับคลื่นพาหะแล้วจึงส่งสัญญาณออกไป) แต่การ์ดนั้นจะต่อกับ Port COM ซึ่งมีการอ่าน และเขียนข้อมูลแบบขนาน จึงต้องนำข้อมูลที่ส่งนั้นมาแปลงเป็นข้อมูลแบบอนุกรมก่อน โดยจะใช้วงจรดังรูปที่ 3.2

หลักการของ PISO คือ การเลื่อนบิทข้อมูลออกไปทีละบิทจนครบ 8 บิท และทุกๆ 8 บิทจะถูกอ่านจาก RAM หนึ่งครั้ง ซึ่งใช้ 74HCT165 (8 Bit Shift Register) โดยใช้สัญญาณนาฬิกาจากภาค Reference Clock เป็นสัญญาณในการ LOAD จะอ่านข้อมูลออกมาทีละ 1 Address (8 บิท) แล้วนำไป LOAD ข้อมูลใน Address ต่อไปจนครบ 4K โดยมีสัญญาณ PISO เป็นตัวชิพข้อมูลที่ได้จากการ LOAD ส่งออกไป

3.3 ภาคอ้างอิงสัญญาณนาฬิกา (Reference Clock)

การทำงานของภาคดังกล่าวจะใช้ 19.0608 MHZ ผลิตความถี่อ้างอิงให้กับ IC เบอร์ 4060 จะได้เอาท์พุทจากขา Q14 (หารด้วย 2^{14}) ซึ่งมีสัญญาณนาฬิกาเป็น 1200 HZ เพื่อนำสัญญาณนี้ไปอ้างอิงให้แก่วงจร PISO และนำสัญญาณ 1200 HZ ไปหาร 8 โดยใช้ 74HC163 ทำให้ได้สัญญาณนาฬิกา 150 HZ เพื่อนำสัญญาณนาฬิกานี้ไปอ้างอิงให้แก่วงจรภาค Address Counter ส่วนสัญญาณ LOAD นั้น จะนำไปอ้างอิงให้กับภาค PISO เช่นกันแต่จะเป็นสัญญาณพัลส์ลบแคบๆทุกๆ 8 ลูกของสัญญาณนาฬิกา 1200 HZ

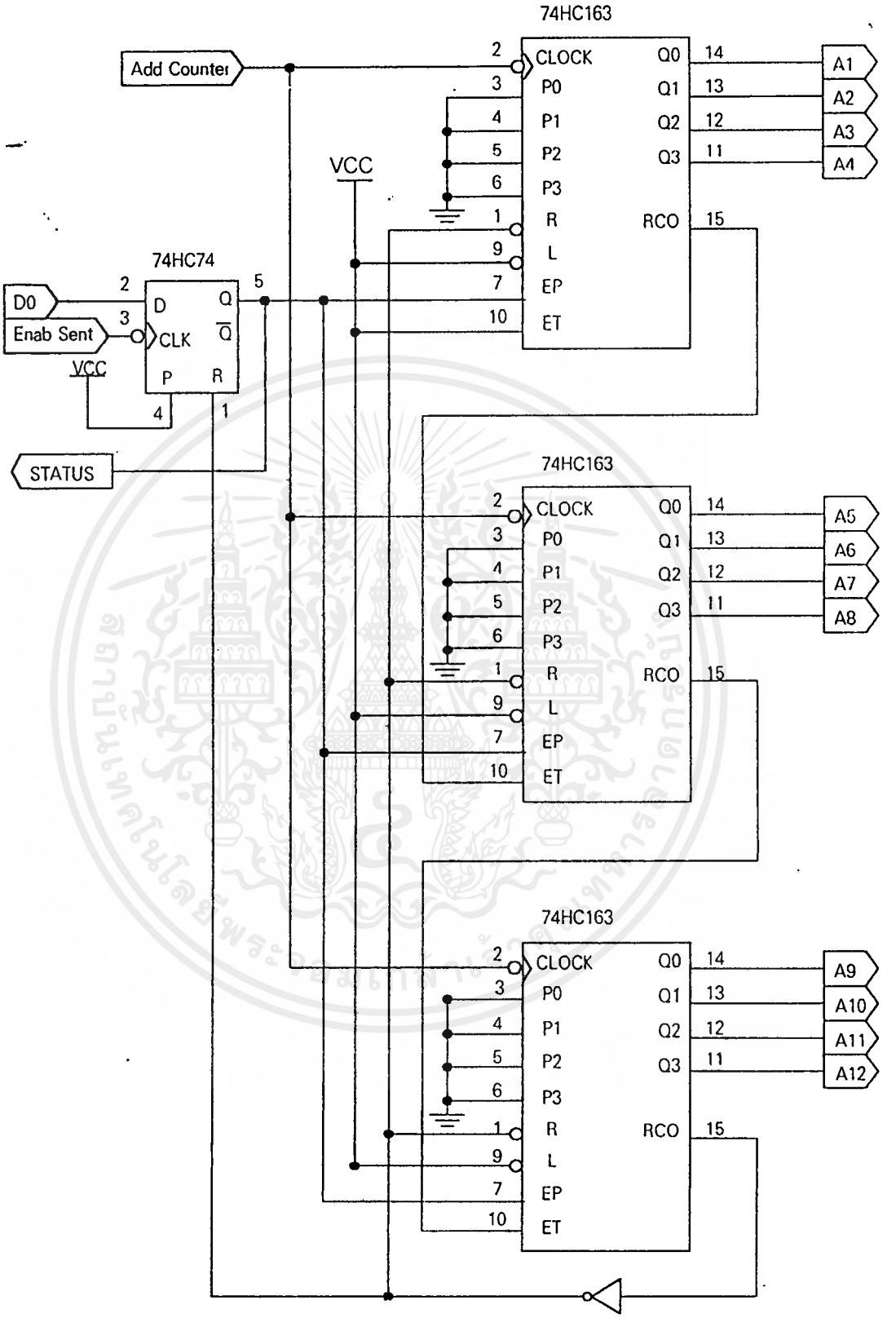


รูปที่ 3.3 วงจรอ้างอิงสัญญาณนาฬิกา

3.4 ภาคนับแอสเดรส (Address Counter)

วงจรนี้ใช้ IC เบอร์ 74HC163 3 ตัวเป็นตัวนับ Address โดยมี Address ทั้งหมด 12 เส้น ดังนั้นจึงสามารถที่จะจับ Address ได้ทั้งหมด 4K ($2^{12} = 4096$) โดยจะใช้ Clock 150 KHZ จากภาค Reference Clock เป็นสัญญาณอ้างอิง เมื่ออยู่ในสภาวะที่ไม่มีข้อมูลสั่งเข้ามา ขาเอาท์พุทของ Address จะเป็น High Impedance อยู่ตลอดเวลา

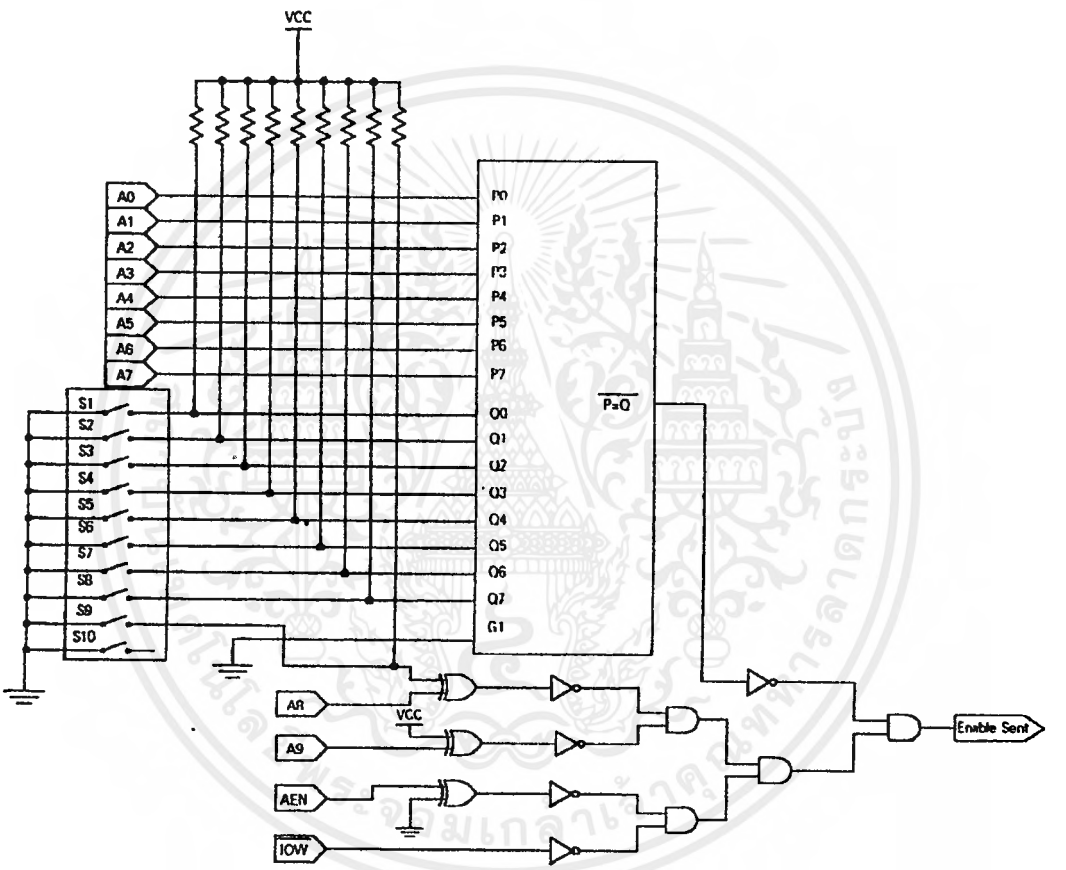
เมื่อทีการสั่งให้ข้อมูล โดยส่งข้อมูลออกทาง D0 ของ Port COM IC 74LS 74 จะทำการเก็บค่าดังกล่าวไว้ เอาท์พุทของ 74LS74 จะไปทำการ Enable อินพุทของ 74HC163 ทำให้เริ่มมีการนับ Address ออกทางเอาท์พุทจนครบ 4K แล้ววงจรจะทำการ RESET ตัวเอง โดยการ RESET ค่าที่เก็บไว้ใน Flip Flop ทำให้ 74HC163 ถูก Disable ไม่มี Address ออกทางเอาท์พุท



รูปที่ 3.4 วงจร Address Counter

3.5 ภาคเลือกพอร์ท และอีนาเบิลเซนต์(Port Select And Enable Sent)

เป็นวงจรใช้สำหรับเลือก Port Address ในการทำงาน เพื่อเป็น Port ในการส่งข้อมูลจากคอมพิวเตอร์มาควบคุมการส่งของสัญญาณ โดยปกติ Port ของคอมพิวเตอร์มีทั้งหมด 1 K จึงใช้เส้น Address ทั้งหมด 10 เส้น คือ A0-A9 โดยใช้ IC เบอร์ 74LS688 ในการ Set Port โดยทำการเปรียบเทียบจาก Address A0-A7 กับ DIP SW. ที่ต่ออยู่ ถ้ามี Address ตรงกันจะมีสัญญาณพัลส์ออกมาที่ขา 19 ของ 74LS688 ส่วน A8 และ A9 โดยต่อเปรียบเทียบกับสัญญาณ " 1 " เอาไว้ ดังนั้นจึงสามารถเลือก Port ต่างๆ โดยใช้ DIP SW. ได้ 256 Address



รูปที่ 3.5 วงจร Enable Sent

จากวงจรที่ใช้กำหนดให้เป็น Port 3F0 จึงต้องปรับ DIP SW. โดยใช้สวิทช์ S1 , S2 , S3 ไว้ที่สัญญาณ " 0 " และให้สวิทช์ S4 , S5 , S6 , S7 , S8 ไว้ที่สัญญาณ " 1 " และสัญญาณจากขา AEN เปรียบเทียบกับสัญญาณ " 0 " เพื่อให้แน่ใจว่าเอาท์พุทที่ออกมาจะไม่เป็น Direct Memory Access (ถ้ามีการ Direct Memory Access สัญญาณจากขา AEN = 1) แล้วนำสัญญาณแต่ละสัญญาณมา AND กันดังวงจรที่ 3.5 ซึ่งจะได้สัญญาณ Enable Sent ออกมาที่เอาท์พุท โดยสัญญาณนี้จะเกิดเมื่อมีการเลือก Port 3F0

โดยมีลักษณะสัญญาณเป็นพัลส์ช่วงสั้นๆ แล้วนำเอาท์พุทที่ได้ไปเข้าขา Clock ของ Flip Flop ในภาค Address Counter

3.6 RAM

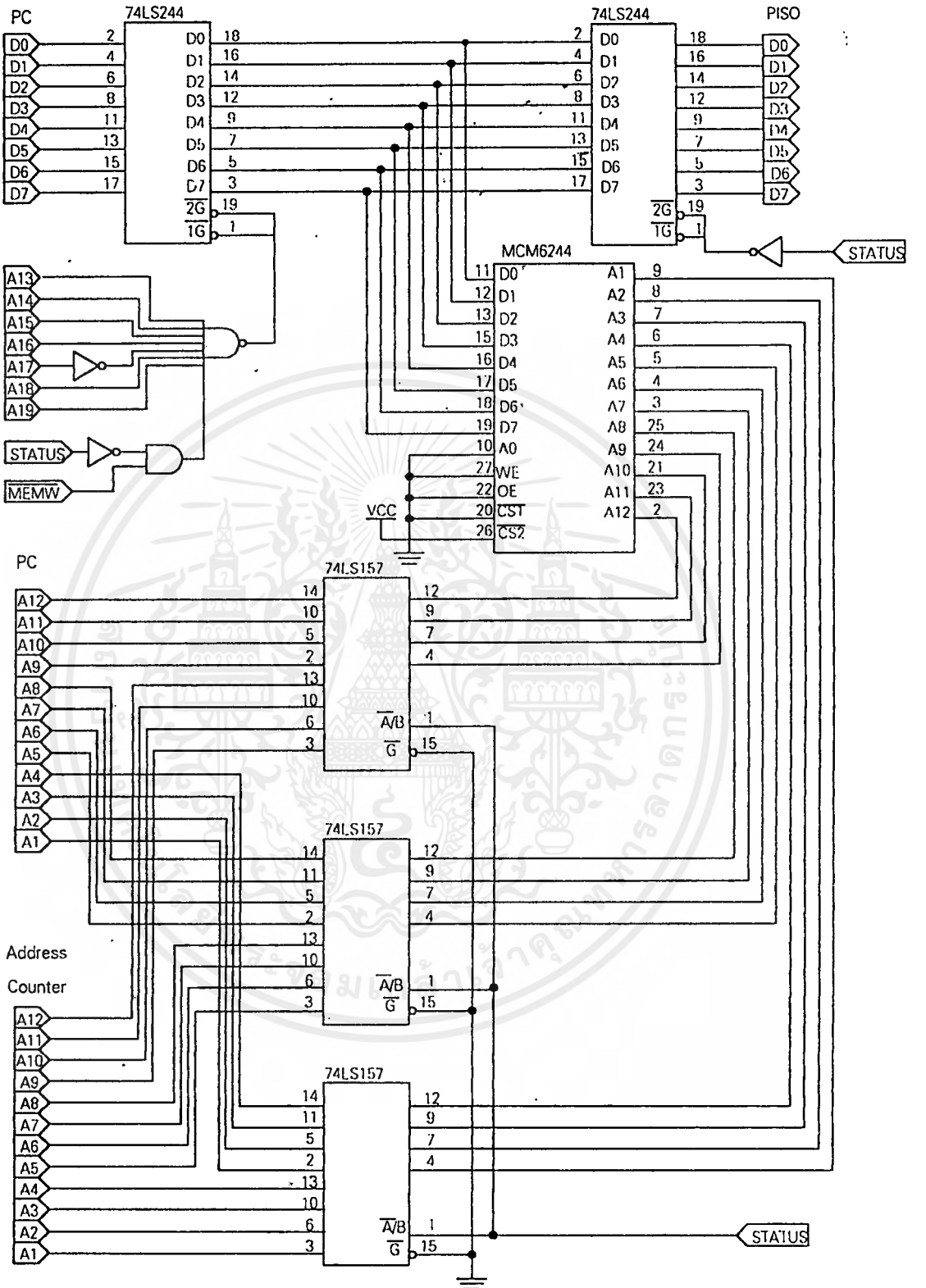
RAM ทำหน้าที่เป็น 2 Way RAM Emulator (RAM 2 ทาง) โดยทางบิทแรกจะติดต่อกับทางคอมพิวเตอร์ ส่วนทางที่ 2 จะติดต่อกับตัวการ์ด ดังนั้น Address ในการที่จะเข้าถึง RAM จึงต้องมาจากคอมพิวเตอร์ หรือว่าการ์ดโดยจะใช้ IC เบอร์ 74LS157 3 ตัว ในการเลือกว่าจะมีการติดต่อกับ RAM โดยผ่านทาง Address จากคอมพิวเตอร์ หรือจาก Address จากการ์ดสัญญาณจาก Status จะเป็นตัวเลือกว่าจะใช้ Address จากส่วนใดโดยถ้า Status เป็น " 0 " จะหมายความว่าขณะนั้น RAM ได้ติดต่อกับคอมพิวเตอร์ แต่ถ้าหาก Status เป็น " 1 " ก็หมายความว่าขณะนั้น RAM ได้ติดต่อกับการ์ด จากวงจรจะมีขา Apping ตำแหน่ง RAM ไว้ที่ DE00 ซึ่งตำแหน่งดังกล่าวนี้ เส้น Address A15-A19 จากคอมพิวเตอร์จะต้องมีลอจิกตามตาราง

Address	A19	A18	A17	A16	A15	A14	A13	A12	.....	A1	A0
Logic	1	1	0	1	1	1	1	X	.....	X	0

ตารางที่ 3.1 แสดงลอจิกที่ Address เมื่อ RAM อยู่ที่ DE00

การติดต่อของ RAM กับคอมพิวเตอร์จะเป็นไปได้รูปแบบเพียงทางเดียว คือ จะส่งข้อมูลมาจากคอมพิวเตอร์ เพื่อเขียนลง RAM ได้เท่านั้น ไม่สามารถอ่านข้อมูลจาก RAM ไปให้คอมพิวเตอร์ได้ เนื่องจากการออกแบบโดยใช้ 74LS244 เป็น Buffer ให้กับ data bus ซึ่งเป็น Buffer ทางเดียว ดังนั้นเมื่อ RAM ติดต่อกับคอมพิวเตอร์เมื่อใดก็หมายถึง จะมีข้อมูลเขียนลง RAM จาก MEMW จะเป็นตัวที่ทำให้ RAM ยอมรับที่จะเขียนข้อมูล โดยจะใช้สัญญาณลอจิก " 0 " จาก MEMW ต่อเข้ากับขา ME ของ RAMเพื่อให้ RAM เขียนข้อมูล

ส่วนการติดต่อช่องแรมกับการ์ด จะติดต่อกันก็ต่อเมื่อ Status เป็น " 1 " และเมื่อใดก็ตามเมื่อ Status เป็น " 1 " ก็หมายถึงขามีการอ่าน RAM ออกไปจาก Address Counter จากการ์ดจะเป็นตัวเข้าถึงตำแหน่งของ RAM และข้อมูลที่ออกจาก data bus ของ RAM จะถูกส่งให้กับภาค PISO โดยผ่าน Buffer (IC เบอร์ 74LS244)

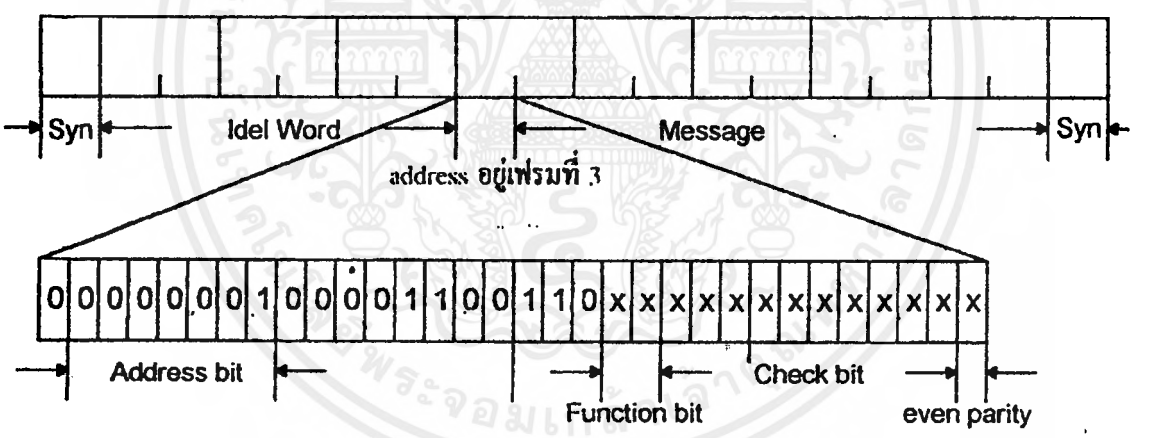


รูปที่ 3.6 วงจรของ RAM

3.7 การจัดข้อมูลในการส่ง

ในการจัดข้อมูลในการส่ง รูปแบบข้อมูลของเพจเจอร์ มีรูปแบบเฉพาะของมันดังกล่าว ได้กล่าวรายละเอียดไปแล้ว และรหัสดังกล่าวเรียกว่า POSAC (Post Office Code Standard Advisory Group) ประกอบด้วย Preamble 576 บิท และ Batch Structure จะประกอบด้วย Synchronous Codeword , Address Codeword , Message Codeword , Idel Codeword การจัดรูปแบบที่จะส่งนี้ เราใช้ Solfware ในการจัด อย่างเช่น หากเราต้องการส่งข้อมูลว่า 4471743 ออกไปที่เครื่องหมายเลข Plug Code 17203 จะมีขั้นตอนดังนี้

- แปลง Plug Code 17203 เป็นรหัส Binary จะได้ 0100001100110011
- รหัส Binary จาก Plug Code ที่ได้ให้ทำการตัดออก 3 บิทหลังออก ก็จะเหลือ 0100001100110
- รหัส Binary จาก Plug Code 3 บิทหลัง คือ $011_2 = 3_{10}$ จะเป็นตัวกำหนดว่าจะนำ Binary จาก Plug Code ที่เหลือไปใส่ไว้ในเฟรมที่ 3 ของแบทช์แรก ซึ่งเฟรมที่ 0-7 จากนั้นข้อความที่เหลือไปไว้ในเฟรมถัดไปของแบทช์แรก เมื่อนำเอาผลที่ได้มาลงแบทช์แรกจะได้ดังนี้



รูปที่ 3.7 แสดงถึงตำแหน่งของ Address ในรหัสคำ

- ต่อจาก Address ก็จะเป็นข้อความที่จะต้องจัด คือ 4471743 แปลงตัวเลขเป็นตัว Binary จะได้ดังนี้

4 = 0100	**7 = 0111**
4 = 0100	**4 = 0100**
7 = 0111	**3 = 0011**
1 = 0001	

นำบิททั้งหมดมาเรียงรวมกันจะได้

0100010001110001011101000011

จะได้บิทรวมทั้งหมด 7 X 4 = 28 บิท แต่เนื่องจาก 1 รหัสคำ (Codeword) สามารถส่งได้แค่เพียง 20 บิท หรือ 5 อักขระ จึงจำเป็นจะต้องมีการเลื่อนข้อมูลบางส่วนไปใส่ยังแบทช์ต่อไปดังนี้

0100010001110001011 1 < เฟรมที่ 4 Codewordที่ 1

01000011 < เฟรมที่ 4 Codewordที่ 2

บทที่ 4

การทำงานของโปรแกรม

การทำงานของโปรแกรมจะมีอยู่ 3 โปรแกรมหลักๆ คือ

4.1 โปรแกรมสำหรับแสดงผล (Message)

ส่วนของโปรแกรมแสดงผล Message จะใช้สำหรับแสดงข้อมูลที่เป็น Message สำหรับข้อมูลมีการรับมาเป็นที่เรียบร้อยแล้วจาก Receiver Card จะแสดงผลข้อมูลออกทางจอภาพ โดยจะสามารถแสดงผลเฉพาะ Message หรือแสดงผลข้อมูลเป็น Codeword ได้ โดยใช้โปรแกรม File menu.exe

ภายในโปรแกรมเมนูจะมีฟังก์ชั่นย่อยๆ ที่จะทำหน้าที่ดังนี้

save_screen()	เก็บหน้าจอ
get_screen()	คืนหน้าจอ
tell_you()	กำหนดขนาดของ Windows
popup_menu()	แสดงเมนูแบบ popup
put_menu()	แสดงเมนูครั้งแรกเมื่อ RUN โปรแกรม
put_string()	แสดงข้อความออกแบบจอภาพ
border_line()	ตีกรอบ Windows
put_one-char()	แสดงตัวอักษร 1 ตัวออกทางจอภาพ
get_resp()	ตอบสนองการเลือกเมนู
one_()	แสดงข้อมูลออกทางเพจเจอร์
Send_()	ส่งเพจเจอร์
Loop_delay()	หน่วงเวลา
print_alpha()	แสดงข้อมูลเพจเจอร์แบบ Character
monitor()	แสดงข้อความ
current_directory	แสดง Directory ปัจจุบัน
type_bits()	แสดงเพจเจอร์ออกมาเป็นบิทๆ
swab_bit()	สลับบิทของ Plug Code
transfer_codeword	จัดการเกี่ยวกับโครงสร้างข้อมูล
change_to_ascii()	แปลง Code ตัวเลขเป็น Ascii Code

4.2 โปรแกรม Write Memory

ส่วนของโปรแกรม Write Memory จะเป็นข้อมูลที่อยู่ในมาตรฐานของ POCSAG ลงใน Memory ของ Transmitter เพื่อที่จะรอการส่งโดยโปรแกรมในส่วนนี้แยกออกเป็น 2 ประเภทคือ

- โปรแกรมที่ Write ข้อมูลของ Pager จริงๆที่ได้ทำการบันทึกที่ได้จากเครื่องส่ง โดยใช้ File putdata.exe รันเพื่อทำการโหลด File Data อีกทีหนึ่ง

- โปรแกรมที่ใช้ Simulate ข้อมูลขึ้นมาเอง puta.exe , putb.exe

4.3 โปรแกรมสำหรับให้เริ่มส่ง Message

โปรแกรมสำหรับสั่งให้ข้อมูลออกทาง Port 3F0 เพื่อส่งให้ (Hard Ware) Pager Transmitter Card ทำการส่งข้อมูลในเครื่องรับไว้ใน Memory ของไปคือ File P.exe และ File P.exe นี้ได้ได้นำมารวมใน Menu ของโปรแกรม menu.exe ด้วย ดังนั้นจึงสามารถเรียกใช้ File P.exe ได้โดยเลือก menu Send

4.4 โปรแกรม menu.c

```
/*--------------------------*/
/* Header and Phototype */
/*--------------------------*/
#include<call_alp.h>

#define arrowU          0x4800
#define arrowD          0x5000
#define arrowR          0x4d00
#define arrowL          0x4b00
#define Spacebar        0x3920
#define Enter           0x1c0d
#define Escap           0x011b

void main(void)
void save_screen(int x,int y,int xx,int yy,char vid_buff[2500]);
void get_screen(int x,int y,int xx,int yy,char vid_buff[2500]);
void tell_you(int x,int y,int xx,int yy,char one,int attrib);
int popup_menu(int x,int y,int count,char *menu_line[]);
void put_menu(int x,int y,int count,char *menu_text[]);
void put_string(int x,int y,char *data,int attrib);
void border_line(int x,int y,int xx,int yy);
void put_one_char(char one,int attrib);
void interrupt_print_screen(void);
void get_ resp(int respond);
void one_(int show);
void sent_(void);
void write_(void);
void tttt_(void);
void loop_delay(void);
unsigned int print_alpha(int xx,int attrib);
void monitor(int page);
```

```
char *current_directory(char *path);
void type_bits(DOUBLE_WORDtemp);
DOUBLEswab_bit(DOUBLE_WORDtemp);
CHECK_MESSAGEtransfer_codeword(ENTRY_MESSAGEentry_message1);
char change_to_ascii(int codein);

DOUBLE_WORD          address,data      /* 32 bits */
ENTRY_MESSAGE        entry_message     /* 224 bits */
CHECK_MESSAGE        check_message     /* 140 bits */
FILE *fp;
FILE *fop;
FILE *fhelp;
char vid_buff[2500];
int Pcount=0;
char *menu_text[]={      "Message.",
                         "....All...",
                         ".........",
                         "...Sent...",
                         ".........",
                         "...Exit.....",
                   };

/*--------------------------------------*/
/*   Function Use in Programe     */
/*--------------------------------------*/
char *current_directory(char *path)
{
    strcpy(path, "X:\\");
    path[0]= 'a'+getdisk();
    getcurdir(0,path+3);
    return(path);
}
```

```
/*-----------------------------------------------*/
/*  Function for save Screen & Restore Screen   */
/*-----------------------------------------------*/
void save_screen(int x,int y,int xx,int yy,char vid_buff[2500]);
{
    register int i,j;
int k=0;
for(j=y;j<=yy;j++)
    for(i=x;i<=xx;i++)
      {
            gotox,y(i,j);
            _AH=8;_BH=0;
            geninterrupt(0x10);
            vid_buff[k++]=_AL;
            vid_buff[k++]=_AH;
            put_one_char(' ',30);
      }
}

/*-----------------------------------*/
void get_screen(int x,int y,int xx,int yy,char vid_buff[2500])
{
   register int i,j;
   int k=0;
   for(j=y;j<=yy;j++)
          for(i=x;i<=xx;i++)
            {
                  gotoxy(i,j);
                  _AL=vid_buff[k++]);
                  _BL=vid_buff[k++]);
                  _AH=9;_BH=0;
                  _CX=1;
```

```
                    geninterrupt(0x10);
                }
    }
    /*-----------------------------------*/
    /* Function Use for make Menu and Select menu */
    /*-----------------------------------*/
    void put_menu(int x,int y,int count,char *menu_line[])
    {
        register int i;
        for(i=0;i<count;i++)
         {
                gotoxy(x,y);
                put_string(x,y,menu_line[i],30;
                x+=8;
        }
    }

    /*-----------------------------------*/
    void put_one_char(char one,int attrib)
    {
        _AL=one;
        _BL=attrib;
        _AH=9;
        _BH=0;
        _CX=1;
        geninterrupt(0x10);
    }

    /*--------------------------*/
    void put_string(int x,int y,char *data,int attrib)
    {
```

```
    while(*data)
        {
            gotoxy(x++,y)
            put_one_char(*data++,attrib);
        }
}

/*---------------------------*/
int popup_menu(int x,int y,int count,char *menu_line[])
{
    char curdir[MAXPATH];
    unsigned int arrow;
    register int i=0;
    put_string(x,y,menu_line[i],44);
    current_directory(curdir,28);
    for(;;)
      {
            arrow = bioskey(0);
            put_string(x,y,menu_line[i],30);
            switch(arrow)
             {
                    case  arrowU: i++;x+=8; break;
                    case  arrowR: i++;x+=8; break;
                    case  arrowD: i--;x-=8; break;
                    case  arrowL: i--;x-=8; break;
                    case  spacebar: i++;x+=8; break;
                    case  Enter: return i;
                    case  Escap: return 5;
             }
            if(i<0)             {i=count-1;x=45;}   /*x=((count-1)*8)+5*/
            if(i>=count)        {i=0;x=5}
            put_string(x,y,menu_line[i],44)
```

```
    }
}

/*-------------------------------------*/
void get_resp(int respond)
{
    switch(respond)
      {
            case 0:  one_(0);  break;
            case 1: one_(1);   break;
            case 2: tttt_();   break;
            case 3: sent();    break;
            case 4: write();   break;
            case 5:            break;
      }
}

/*-----------------------------------------*/
void border_line(int x,int y,int xx,int yy)
{
   register int i;
          for(i=x+1;i<xx;i++)
            {
                gotoxy(i,y);put_one_char(205,30);
                gotoxy(i,yy);put_one_char(205,30);
            }
          for(i=y+1;i<yy;i++)
            {
                gotoxy(x,i);put_one_char(179,30);
                gotoxy(xx,i);put_one_char(179,30);
            }
```

```
            gotoxy(x,y);put_one_char(213,30);
            gotoxy(x,yy);put_one_char(212,30);
            gotoxy(xx,y);put_one_char(184,30);
            gotoxy(xx,yy);put_one_char(190,30);
}

/*-------------------------------------------*/
void tell_you(int x,int y,int xx,int yy,char one,int attrib)
{
    register int i,j;
    for(j=y;<=yy;j++)
            for(i=x;i<=xx;i++)
             {
                gotoxy(i,j);
                put_one_char(one,attrib);
             }
}

/*-----------------------------------------------*/
/*                Main Program                   */
/*             Change Interrupt Vector           */
/*      Read Blugcode from keyboard              */
/*    Allocate memory for Stay resident          */
/*-----------------------------------------------*/

/*-------------------------------------------------------------*/
/*     This Program is Resident in interrupt Vector NO.05H     */
/*-------------------------------------------------------------*/
void main(void)
{
   int x,xx,y,yy,count=6;
   register int i;
```

```
        int respond;
        int curx,cury,mode,m;
    if(pcount==0)
    {
        Pcount++;
        _AH=0x0f;
        geninterrupt(0x10);
        mode=_AL;
        if((mode==2)||(mode==3)||(mode==7))
          {
                _AH=3;_BH=0;
                geninterrupt(0x10);
                curx = _DL;cury = _DH;

                x=1;y=1;xx=78;yy=16;
                save_screen(x,y,xx,yy,vid_buff);
                put_string(16,yy, "Pager Emulater Program"
                                  "Version 3 By Pager Group",28);
                border_line(x,y,xx,yy);border_line(x+1,3,xx-1,yy);
                put_string(36,y, "Pager",30);
                put_menu(5,2,count,menu_text);
/*---------- Plug Code ----------*/
tell_you(4,4,75,13, ' ',30);
border_line(25,4,55,6);
put_string(31,4, "Enter NUMBERIC only",157);
put_string(26,5, "Type your BLUGECODE:",30);
scanf("%d",&address);
swab_bit(address);
/*-------------------------------------------------*/
                for(;;)
                {
                        respond=popup_menu(5,2,count,menu_text);
```

```
                if (respond==5) goto exit0;
                get_resp(respond);
            }
            exit0;
            get_screen(x,y,xx,yy,vid_buff);
            gotoxy(curx+1,cury+1);
            }
            Pcount—;
    }
}
/*-------------------------------------------------*/
void sent_(void)
{
    int i;
    tell_you(4,4,75,13, '  ',30);
    put_string(30,6, "Now..Memory Reading",158);
    system("p.exe");
    border_line(25,7,55,9);
    for(i=26;i<55;i++)
    {
        loop_delay();
        put_string(i,8, "R",112);
    }
    put_string(37,10, "OK…",30);
}
/*-------------------------------------------------*/
void loop_delay(void)
{
        int i,j;
        for(i=0;i<1200;i++)
            for(j=0;j<400;j++)
}
```

```
/*-------------------------------------------------------*/
void write_(void)
{
}
/*-------------------------------------------------------*/
void tttt_(void)
{
}
/*-------------------------------------------------------*/
void one_(int show)
{
    unsigned OFSET=0);
    unsigned int getoff;
    char alpha;
    int i,j,k,m=0;
    int code,status,page=0;
    unsigned int dx,dy;
    unsigned int sx=5;
    int attrib=28;
    tell_you(4,4,75,13, ' ',30);
    for(i=0;i<60;i++)
    {
            for(j=0;j<4;j++)
            {
                    data.BYTE[j]=peekb(SEGMENT,OFSET);
                    OFSET+=2;
            }
            if((data.BULGCODE.b2_10==address.BLUGCODE.b2_10)
                &&(data.BLUGCODE.b11_19==addressBLUGCODE.b11_19))
            {
                    for(;;)
                    {
```

```
do{
        for(j=0;j<4;j++)
        {
                data.BYTE[j]=peekb(SEGMENT,OFSET);
                OFSET+=2;
        }
  }while((data.CODEWORD==FRAME_SYN)
        ||(data.CODEWORD==FRAME_SYN-4)
        ||(data.CODEWORD==IDLE_WORD)
        ||(data.CODEWORD==IDEL_WORD-4));

if((data.CODEWORD==EOT)
 ||(data.CODEWORD==CLEAR)
 ||(data.CODEWORD==END))
        m=7;
   else
   {
        entry_message.CODE_WORD[m]=data.CODEWORD;
        m++;
        if(show==1)
        {
                if(page==1)
                {       dx=24;dy=4      }
                else
                {       dx=4;dy=4       }
                border_line(dx,dy,dx+50,dy+8);
                gotoxy(dx+1,dy+m);
                type_bits(data);
        }
   }
   if(m==7)
   {
```

```
                    m=0;
                    for(j=0;j<20;j++)
                    check_message.CHECK_MCW[j]=0;
                    check_message=transfer_codeword(entry_message);
                    sx=print_alpha(sx,attrib);
                    if(page==0)
                    {       page=1;attrib=30;       }
                    else
                    {       page=0;attrib=28;       }
                    if(show==1)
                    {       getoff=bioskey(0);
                            if(getoff==Escap) break;
                    }
                    if(show==0)
                    {
                            if(sx==Escap) break;
                    }
                }
                if((data.CODEWORD==EOT)||(data.CODEWORD==CLEAR)
                  ||(data.CODEWORD==END))
                    goto END_;
            }
        }
    }
    END_;
    OFSET=0;
}

/*-------------------------------------------------*/
unsigned int print_alpha(int sx,int attrib)
{
    int code,i,x,sy=13;
```

```
    unsigned int getoff;
    char alpha;
    for(i=0;i<20;i++)
    {
            code=check_message.CHECK_MCW[i];
            alpha=change_to_ascii(code);
            if(sx>73)
            {
                    sx=5;
                    getoff=bioskey(0);
                    if(getoff==Escap)
                    return getoff);
            }
            gotoxy(sx++,sy);
            put_one_char(alpha,attrib);
    }
    x=wherex();
    return x+1;
}

/*-------------------------------------------------*/
void type_bits(DOUBLE_WORD)data1)
{
    short int m1,m2,m3,m4,m5,m6,m7,m8,m9,m10;
    short int m11,m12,m13,m14,m15,m16,m17,m18,m19,m20;
    short int m21,m22,m23,m24,m25,m26,m27,m28,m29,m30;m31,m32;
    m1=data1.BINARY.b1;          m2=data1.BINARY.b2;
    m3=data1.BINARY.b3;          m4=data1.BINARY.b4;
    m5=data1.BINARY.b5;          m6=data1.BINARY.b6;
    m7=data1.BINARY.b7;          m8=data1.BINARY.b8;
    m9=data1.BINARY.b9;          m10=data1.BINARY.b10;
    m11=data1.BINARY.b11;        m12=data1.BINARY.b12;
```

```
    m13=data1.BINARY.b13;        m14=data1.BINARY.b14;
    m15=data1.BINARY.b15;        m16=data1.BINARY.b16;
    m17=data1.BINARY.b17;        m18=data1.BINARY.b18;
    m19=data1.BINARY.b19;        m20=data1.BINARY.b20;
    m21=data1.BINARY.b21;        m22=data1.BINARY.b22;
    m23=data1.BINARY.b23;        m24=data1.BINARY.b24;
    m25=data1.BINARY.b25;        m26=data1.BINARY.b26;
    m27=data1.BINARY.b27;        m28=data1.BINARY.b28;
    m29=data1.BINARY.b29;        m30=data1.BINARY.b30;
    m31=data1.BINARY.b31;        m32=data1.BINARY.b32;
    printf("%d%d%d%d%d%d%d%d",m1,m2,m3,m4,m5,m6,m7,m8);
    printf("%d%d%d%d%d%d%d%d",m9,m10,m11,m12,m13,m14,m15);
    printf("%d%d%d%d%d%d%d",m16,m17,m18,m19,m20,m21);
    printf("%d%d%d%d%d%d",m22,m23,m24,m25,m26);
    printf("%d%d%d%d%d%d",m27,m28,m29,m30,m31,m32);
    printf("%101u\n",data1.CODEWORD);
}
/*-------------------------------------------------------*/
DOUBLE_WORDswab_bit(DOUBLE_WORDtemp)
{
    address.BINARY.b19 = temp.BINARY.b4;
    address.BINARY.b18 = temp.BINARY.b5;
    address.BINARY.b17 = temp.BINARY.b6;
    address.BINARY.b16 = temp.BINARY.b7;
    address.BINARY.b15 = temp.BINARY.b8;
    address.BINARY.b14 = temp.BINARY.b9;
    address.BINARY.b13 = temp.BINARY.b10;
    address.BINARY.b12 = temp.BINARY.b11;
    address.BINARY.b11 = temp.BINARY.b12;
    address.BINARY.b10 = temp.BINARY.b13;
    address.BINARY.b9 = temp.BINARY.b14;
    address.BINARY.b8 = temp.BINARY.b15;
```

```
    address.BINARY.b7 = temp.BINARY.b16;
    address.BINARY.b6 = temp.BINARY.b17;
    address.BINARY.b5 = temp.BINARY.b18;
    address.BINARY.b4 = temp.BINARY.b19;
    address.BINARY.b3 = temp.BINARY.b20;
    address.BINARY.b2 = temp.BINARY.b21;
}
/*-------------------------------------------------------------*/
CHECK_MESSAGEtransfer_codeword(ENTRY_MESSAGEentry_message1)
{
    CHECK_MESSAGE temp;
    temp.CHECK_MCW1.W1_M1 = entry_message1.MESSAGE_7CW.W1_M1;
    temp.CHECK_MCW1.W1_M2 = entry_message1.MESSAGE_7CW.W1_M2;
    temp.CHECK_MCW1.W1_M3 = entry_message1.MESSAGE_7CW.W1_M3;

    temp.CHECK_MCW1.W2_M3 = entry_message1.MESSAGE_7CW.W2_M3;
    temp.CHECK_MCW1.W2_M4 = entry_message1.MESSAGE_7CW.W2_M4;
    temp.CHECK_MCW1.W2_M5 = entry_message1.MESSAGE_7CW.W2_M5;
    temp.CHECK_MCW1.W2_M6 = entry_message1.MESSAGE_7CW.W2_M6;

    temp.CHECK_MCW1.W3_M6 = entry_message1.MESSAGE_7CW.W3_M6;
    temp.CHECK_MCW1.W3_M7 = entry_message1.MESSAGE_7CW.W3_M7;
    temp.CHECK_MCW1.W3_M8 = entry_message1.MESSAGE_7CW.W3_M8;
    temp.CHECK_MCW1.W3_M9 = entry_message1.MESSAGE_7CW.W3_M9;

    temp.CHECK_MCW1.W4_M9 = entry_message1.MESSAGE_7CW.W4_M9;
    temp.CHECK_MCW1.W4_M10 = entry_message1.MESSAGE_7CW.W4_M10;
    temp.CHECK_MCW1.W4_M11 = entry_message1.MESSAGE_7CW.W4_M11;
    temp.CHECK_MCW1.W4_M12 = entry_message1.MESSAGE_7CW.W4_M12;

    temp.CHECK_MCW1.W5_M12 = entry_message1.MESSAGE_7CW.W5_M12;
    temp.CHECK_MCW1.W5_M13 = entry_message1.MESSAGE_7CW.W5_M13;
```

```
    temp.CHECK_MCW1.W5_M14 = entry_message1.MESSAGE_7CW.W5_M14;
    temp.CHECK_MCW1.W5_M15 = entry_message1.MESSAGE_7CW.W5_M15;

    temp.CHECK_MCW1.W6_M15 = entry_message1.MESSAGE_7CW.W6_M15;
    temp.CHECK_MCW1.W6_M16 = entry_message1.MESSAGE_7CW.W6_M16;
    temp.CHECK_MCW1.W6_M17 = entry_message1.MESSAGE_7CW.W6_M17;
    temp.CHECK_MCW1.W6_M18 = entry_message1.MESSAGE_7CW.W6_M18;

    temp.CHECK_MCW1.W7_M18 = entry_message1.MESSAGE_7CW.W7_M18;
    temp.CHECK_MCW1.W7_M19 = entry_message1.MESSAGE_7CW.W7_M19;
    temp.CHECK_MCW1.W7_M20 = entry_message1.MESSAGE_7CW.W7_M20;
    return temp;
}
*/----------------------------------------------------*/
char change_to_ascii(int codein)
{
    char d;
    if((codein>48)&&(codein<127))
    switch(codein)
    {
            case 48:    d = '0' ;    break;
            case 49:    d = '1' ;    break;
            case 50:    d = '2' ;    break;
            case 51:    d = '3' ;    break;
            case 52:    d = '4' ;    break;
            case 53:    d = '5' ;    break;
            case 54:    d = '6' ;    break;
            case 55:    d = '7' ;    break;
            case 56:    d = '8' ;    break;
            case 57:    d = '9' ;    break;
            case 58:    d = ':' ;    break;
            case 59:    d = ';' ;    break;
```

```
case 60:    d = '<' ;    break;
case 61:    d = '=' ;    break;
case 62:    d = '>' ;    break;
case 63:    d = '?' ;    break;
case 64:    d = '@' ;    break;
case 65:    d = 'A' ;    break;
case 66:    d = 'B' ;    break;
case 67:    d = 'C' ;    break;
case 68:    d = 'D' ;    break;
case 69:    d = 'E' ;    break;
case 70:    d = 'F' ;    break;
case 71:    d = 'G' ;    break;
case 72:    d = 'H' ;    break;
case 73:    d = 'I' ;    break;
case 74:    d = 'J' ;    break;
case 75:    d = 'K' ;    break;
case 76:    d = 'L' ;    break;
case 77:    d = 'M'      break;
case 78:    d = 'N' ;    break;
case 79:    d = 'O' ;    break;
case 80:    d = 'P' ;    break;
case 81:    d = 'Q' ;    break;
case 82:    d = 'R' ;    break;
case 83:    d = 'S' ;    break;
case 84:    d = 'T' ;    break;
case 85:    d = 'U' ;    break;
case 86:    d = 'V' ;    break;
case 87:    d = 'W'      break;
case 88:    d = 'X' ;    break;
case 89:    d = 'Y' ;    break;
case 90:    d = 'Z' ;    break;
case 91:    d = '[' ;    break;
```

```
case 92:    d = ' ' ;    break;
case 93:    d = ']' ;    break;
case 94:    d = '^' ;    break;
case 95:    d = '-' ;    break;
case 96:    d = "" ;     break;
case 97:    d = 'a' ;    break;
case 98:    d = 'b' ;    break;
case 99:    d = 'c' ;    break;
case 100:   d = 'd' ;    break;
case 101:   d = 'e' ;    break;
case 102:   d = 'f' ;    break;
case 103:   d = 'g' ;    break;
case 104:   d = 'h' ;    break;
case 105:   d = 'i' ;    break;
case 106:   d = 'j' ;    break;
case 107:   d = 'k' ;    break;
case 108:   d = 'l' ;    break;
case 109:   d = 'm' ;    break;
case 110:   d = 'n' ;    break;
case 111:   d = 'o' ;    break;
case 112:   d = 'p' ;    break;
case 113:   d = 'q' ;    break;
case 114:   d = 'r' ;    break;
case 115:   d = 's' ;    break;
case 116:   d = 't' ;    break;
case 117:   d = 'u' ;    break;
case 118:   d = 'v' ;    break;
case 119:   d = 'w' ;    break;
case 120:   d = 'x' ;    break;
case 121:   d = 'y' ;    break;
case 122:   d = 'z' ;    break;
case 123:   d = '{' ;    break;
```

```
            case 124:       d = '|' ;       break;
            case 125:       d = '}' ;       break;
            case 126:       d = '~' ;       break;
            case 127:       d = '#' ;       break;
    }
    else d = ' ' ;
    return d;
}

/*-------------------------------------------------------------*/
/*-------------------------------------------------------------*/
/*-------------------------------------------------------------*/
```

4.5 โปรแกรม P.c

```
# include<call_app.h>
void main(void)
{
        int ports,data;
        printf("\nPort number");
        scanf("%d",&ports);
        printf("\nData Control");
        scanf("%d",&data);
        printf("\nPort = %d \n Data = %d\n",port,data);
        outportb(0x3f0,1);
}
```

4.6 โปรแกรม puta.c

```
# include<call_alp.h>
void main(void)
{
        int i,j;
        int OFSET = 0;

        printf("\nPut Premble");
        for(i=0;i<72;i++)
        {
                pokeb(SEGMENT.OFSET,85);
                OFSET+=2;
        }

        printf("\nPut SYN");

                pokeb(SEGMENT,OFSET,62);OFSET+=2;
                pokeb(SEGMENT,OFSET,75);OFSET+=2;
                pokeb(SEGMENT,OFSET,168);OFSET+=2;
                pokeb(SEGMENT,OFSET,27);OFSET+=2;
        printf("\nPut Idle");

        for(i=0;i<6;i++)
        {
                pokeb(SEGMENT,OFSET,94);OFSET+=2;
                pokeb(SEGMENT,OFSET,145);OFSET+=2;
                pokeb(SEGMENT,OFSET,131);OFSET+=2;
                pokeb(SEGMENT,OFSET,233);OFSET+=2;
        }

        printf("\nPut Address");
```

```
        pokeb(SEGMENT,OFSET,128);OFSET+=2;
        pokeb(SEGMENT,OFSET,48);OFSET+=2;
        pokeb(SEGMENT,OFSET,163);OFSET+=2;
        pokeb(SEGMENT,OFSET,4);OFSET+=2;

printf("\nPut AAAAA");
        /*.......WORD.1......*/
        pokeb(SEGMENT,OFSET,131);OFSET+=2;
        pokeb(SEGMENT,OFSET,193);OFSET+=2;
        pokeb(SEGMENT,OFSET,0);OFSET+=2;
        pokeb(SEGMENT,OFSET,0);OFSET+=2;

        /*.......WORD.2......*/
        pokeb(SEGMENT,OFSET,7);OFSET+=2;
        pokeb(SEGMENT,OFSET,13);OFSET+=2;
        pokeb(SEGMENT,OFSET,1);OFSET+=2;
        pokeb(SEGMENT,OFSET,0);OFSET+=2;

        /*.......WORD.3......*/
        pokeb(SEGMENT,OFSET,13);OFSET+=2;
        pokeb(SEGMENT,OFSET,6);OFSET+=2;
        pokeb(SEGMENT,OFSET,3);OFSET+=2;
        pokeb(SEGMENT,OFSET,0);OFSET+=2;

        /*.......WORD.4......*/
        pokeb(SEGMENT,OFSET,25);OFSET+=2;
        pokeb(SEGMENT,OFSET,12);OFSET+=2;
        pokeb(SEGMENT,OFSET,6);OFSET+=2;
        pokeb(SEGMENT,OFSET,0);OFSET+=2;

        /*.......WORD.5......*/
        pokeb(SEGMENT,OFSET,49);OFSET+=2;
```

```
pokeb(SEGMENT,OFSET,24);OFSET+=2;
pokeb(SEGMENT,OFSET,12);OFSET+=2;
pokeb(SEGMENT,OFSET,0);OFSET+=2;

/*.....WORD.6.....*/
pokeb(SEGMENT,OFSET,97);OFSET+=2;
pokeb(SEGMENT,OFSET,48);OFSET+=2;
pokeb(SEGMENT,OFSET,24);OFSET+=2;
pokeb(SEGMENT,OFSET,0);OFSET+=2;

/*.....WORD.7.....*/
pokeb(SEGMENT,OFSET,193);OFSET+=2;
pokeb(SEGMENT,OFSET,96);OFSET+=2;
pokeb(SEGMENT,OFSET,16);OFSET+=2;
pokeb(SEGMENT,OFSET,0);OFSET+=2;

/*.....WORD.1.....*/
pokeb(SEGMENT,OFSET,131);OFSET+=2;
pokeb(SEGMENT,OFSET,193);OFSET+=2;
pokeb(SEGMENT,OFSET,0);OFSET+=2;
pokeb(SEGMENT,OFSET,0);OFSET+=2;

/*.....WORD.2.....*/
pokeb(SEGMENT,OFSET,7);OFSET+=2;
pokeb(SEGMENT,OFSET,13);OFSET+=2;
pokeb(SEGMENT,OFSET,1);OFSET+=2;
pokeb(SEGMENT,OFSET,0);OFSET+=2;

printf("\nPut SYN");

/*.....WORD.3.....*/
pokeb(SEGMENT,OFSET,13);OFSET+=2;
```

```
pokeb(SEGMENT,OFSET,6);OFSET+=2;
pokeb(SEGMENT,OFSET,3);OFSET+=2;
pokeb(SEGMENT,OFSET,0);OFSET+=2;

/*.....WORD.4.....*/
pokeb(SEGMENT,OFSET,25);OFSET+=2;
pokeb(SEGMENT,OFSET,12);OFSET+=2;
pokeb(SEGMENT,OFSET,6);OFSET+=2;
pokeb(SEGMENT,OFSET,0);OFSET+=2;

/*.....WORD.5.....*/
pokeb(SEGMENT,OFSET,49);OFSET+=2;
pokeb(SEGMENT,OFSET,24);OFSET+=2;
pokeb(SEGMENT,OFSET,12);OFSET+=2;
pokeb(SEGMENT,OFSET,0);OFSET+=2;

/*.....WORD.6.....*/
pokeb(SEGMENT,OFSET,97);OFSET+=2;
pokeb(SEGMENT,OFSET,48);OFSET+=2;
pokeb(SEGMENT,OFSET,24);OFSET+=2;
pokeb(SEGMENT,OFSET,0);OFSET+=2;

/*.....WORD.7.....*/
pokeb(SEGMENT,OFSET,193);OFSET+=2;
pokeb(SEGMENT,OFSET,96);OFSET+=2;
pokeb(SEGMENT,OFSET,16);OFSET+=2;
pokeb(SEGMENT,OFSET,0);OFSET+=2;

/*.....WORD.1.....*/
pokeb(SEGMENT,OFSET,131);OFSET+=2;
pokeb(SEGMENT,OFSET,193);OFSET+=2;
pokeb(SEGMENT,OFSET,0);OFSET+=2;
```

```
pokeb(SEGMENT,OFSET,0);OFSET+=2;

/*.....WORD.2.....*/
pokeb(SEGMENT,OFSET,7);OFSET+=2;
pokeb(SEGMENT,OFSET,13);OFSET+=2;
pokeb(SEGMENT,OFSET,1);OFSET+=2;
pokeb(SEGMENT,OFSET,0);OFSET+=2;

/*.....WORD.3.....*/
pokeb(SEGMENT,OFSET,13);OFSET+=2;
pokeb(SEGMENT,OFSET,6);OFSET+=2;
pokeb(SEGMENT,OFSET,3);OFSET+=2;
pokeb(SEGMENT,OFSET,0);OFSET+=2;

/*.....WORD.4.....*/
pokeb(SEGMENT,OFSET,25);OFSET+=2;
pokeb(SEGMENT,OFSET,12);OFSET+=2;
pokeb(SEGMENT,OFSET,6);OFSET+=2;
pokeb(SEGMENT,OFSET,0);OFSET+=2;

/*.....WORD.5.....*/
pokeb(SEGMENT,OFSET,49);OFSET+=2;
pokeb(SEGMENT,OFSET,24);OFSET+=2;
pokeb(SEGMENT,OFSET,12);OFSET+=2;
pokeb(SEGMENT,OFSET,0);OFSET+=2;

/*.....WORD.6.....*/
pokeb(SEGMENT,OFSET,97);OFSET+=2;
pokeb(SEGMENT,OFSET,48);OFSET+=2;
pokeb(SEGMENT,OFSET,24);OFSET+=2;
pokeb(SEGMENT,OFSET,0);OFSET+=2;
```

```
                /*.....WORD.7.....*/
                pokeb(SEGMENT,OFSET,193);OFSET+=2;
                pokeb(SEGMENT,OFSET,96);OFSET+=2;
                pokeb(SEGMENT,OFSET,16);OFSET+=2;
                pokeb(SEGMENT,OFSET,0);OFSET+=2;

                /*.....WORD.1.....*/
                pokeb(SEGMENT,OFSET,131);OFSET+=2;
                pokeb(SEGMENT,OFSET,193);OFSET+=2;
                pokeb(SEGMENT,OFSET,0);OFSET+=2;
                pokeb(SEGMENT,OFSET,0);OFSET+=2;

                /*.....WORD.2.....*/
                pokeb(SEGMENT,OFSET,7);OFSET+=2;
                pokeb(SEGMENT,OFSET,13);OFSET+=2;
                pokeb(SEGMENT,OFSET,1);OFSET+=2;
                pokeb(SEGMENT,OFSET,0);OFSET+=2;

                printf("\nPut Idle");

                for(i=0;i<2;i++)
                {
                        pokeb(SEGMENT,OFSET,94);OFSET+=2;
                        pokeb(SEGMENT,OFSET,145);OFSET+=2;
                        pokeb(SEGMENT,OFSET,131);OFSET+=2;
                        pokeb(SEGMENT,OFSET,233);OFSET+=2;
                }
        printf("\nPut CLEAR");
        for(i=0;i<4,i++)
        {       pokeb(SEGMENT,OFSET,0);OFSET+=2;        }
        printf("\nEnd");
}
```

4.7 โปรแกรม putb.c

```
# include<call_alp.h>
void main(void)
{
        int i,j;
        int OFSET = 0;

        printf("\nPut Premble");
        for(i=0;i<72;i++)
        {
                pokeb(SEGMENT.OFSET,85);
                OFSET+=2;
        }

        printf("\nPut SYN");

                pokeb(SEGMENT,OFSET,62);OFSET+=2;
                pokeb(SEGMENT,OFSET,75);OFSET+=2;
                pokeb(SEGMENT,OFSET,168);OFSET+=2;
                pokeb(SEGMENT,OFSET,27);OFSET+=2;
        printf("\nPut Idle");

        for(i=0;i<6;i++)
        {
                pokeb(SEGMENT,OFSET,94);OFSET+=2;
                pokeb(SEGMENT,OFSET,145);OFSET+=2;
                pokeb(SEGMENT,OFSET,131);OFSET+=2;
                pokeb(SEGMENT,OFSET,233);OFSET+=2;
        }

        printf("\nPut Address");
```

```
        pokeb(SEGMENT,OFSET,128);OFSET+=2;
        pokeb(SEGMENT,OFSET,48);OFSET+=2;
        pokeb(SEGMENT,OFSET,163);OFSET+=2;
        pokeb(SEGMENT,OFSET,4);OFSET+=2;

printf("\nPut BBBBB");
        /*.......WORD.1......*/
        pokeb(SEGMENT,OFSET,133);OFSET+=2;
        pokeb(SEGMENT,OFSET,66);OFSET+=2;
        pokeb(SEGMENT,OFSET,1);OFSET+=2;
        pokeb(SEGMENT,OFSET,0);OFSET+=2;

        /*.......WORD.2......*/
        pokeb(SEGMENT,OFSET,11);OFSET+=2;
        pokeb(SEGMENT,OFSET,133);OFSET+=2;
        pokeb(SEGMENT,OFSET,2);OFSET+=2;
        pokeb(SEGMENT,OFSET,0);OFSET+=2;

        /*.......WORD.3......*/
        pokeb(SEGMENT,OFSET,21);OFSET+=2;
        pokeb(SEGMENT,OFSET,10);OFSET+=2;
        pokeb(SEGMENT,OFSET,5);OFSET+=2;
        pokeb(SEGMENT,OFSET,0);OFSET+=2;

        /*.......WORD.4......*/
        pokeb(SEGMENT,OFSET,4);OFSET+=2;
        pokeb(SEGMENT,OFSET,20);OFSET+=2;
        pokeb(SEGMENT,OFSET,10);OFSET+=2;
        pokeb(SEGMENT,OFSET,0);OFSET+=2;

        /*.......WORD.5......*/
        pokeb(SEGMENT,OFSET,81);OFSET+=2;
```

```
pokeb(SEGMENT,OFSET,40);OFSET+=2;
pokeb(SEGMENT,OFSET,20);OFSET+=2;
pokeb(SEGMENT,OFSET,0);OFSET+=2;

/*.....WORD.6.....*/
pokeb(SEGMENT,OFSET,161);OFSET+=2;
pokeb(SEGMENT,OFSET,80);OFSET+=2;
pokeb(SEGMENT,OFSET,8);OFSET+=2;
pokeb(SEGMENT,OFSET,0);OFSET+=2;

/*.....WORD.7.....*/
pokeb(SEGMENT,OFSET,67);OFSET+=2;
pokeb(SEGMENT,OFSET,16);OFSET+=2;
pokeb(SEGMENT,OFSET,16);OFSET+=2;
pokeb(SEGMENT,OFSET,0);OFSET+=2;

/*.....WORD.1.....*/
pokeb(SEGMENT,OFSET,133);OFSET+=2;
pokeb(SEGMENT,OFSET,66);OFSET+=2;
pokeb(SEGMENT,OFSET,1);OFSET+=2;
pokeb(SEGMENT,OFSET,0);OFSET+=2;

/*.....WORD.2.....*/
pokeb(SEGMENT,OFSET,11);OFSET+=2;
pokeb(SEGMENT,OFSET,133);OFSET+=2;
pokeb(SEGMENT,OFSET,2);OFSET+=2;
pokeb(SEGMENT,OFSET,0);OFSET+=2;

printf("\nPut SYN");

/*.....WORD.3.....*/
pokeb(SEGMENT,OFSET,21);OFSET+=2;
```

```
pokeb(SEGMENT,OFSET,10);OFSET+=2;
pokeb(SEGMENT,OFSET,5);OFSET+=2;
pokeb(SEGMENT,OFSET,0);OFSET+=2;

/*.....WORD.4.....*/
pokeb(SEGMENT,OFSET,41);OFSET+=2;
pokeb(SEGMENT,OFSET,20);OFSET+=2;
pokeb(SEGMENT,OFSET,10);OFSET+=2;
pokeb(SEGMENT,OFSET,0);OFSET+=2;

/*.....WORD.5.....*/
pokeb(SEGMENT,OFSET,81);OFSET+=2;
pokeb(SEGMENT,OFSET,40);OFSET+=2;
pokeb(SEGMENT,OFSET,20);OFSET+=2;
pokeb(SEGMENT,OFSET,0);OFSET+=2;

/*.....WORD.6.....*/
pokeb(SEGMENT,OFSET,161);OFSET+=2;
pokeb(SEGMENT,OFSET,80);OFSET+=2;
pokeb(SEGMENT,OFSET,8);OFSET+=2;
pokeb(SEGMENT,OFSET,0);OFSET+=2;

/*.....WORD.7.....*/
pokeb(SEGMENT,OFSET,67);OFSET+=2;
pokeb(SEGMENT,OFSET,161);OFSET+=2;
pokeb(SEGMENT,OFSET,16);OFSET+=2;
pokeb(SEGMENT,OFSET,0);OFSET+=2;

/*.....WORD.1.....*/
pokeb(SEGMENT,OFSET,133);OFSET+=2;
pokeb(SEGMENT,OFSET,66);OFSET+=2;
pokeb(SEGMENT,OFSET,1);OFSET+=2;
```

```
pokeb(SEGMENT,OFSET,0);OFSET+=2;

/*.......WORD.2......*/
pokeb(SEGMENT,OFSET,11);OFSET+=2;
pokeb(SEGMENT,OFSET,133);OFSET+=2;
pokeb(SEGMENT,OFSET,2);OFSET+=2;
pokeb(SEGMENT,OFSET,0);OFSET+=2;

/*.......WORD.3......*/
pokeb(SEGMENT,OFSET,21);OFSET+=2;
pokeb(SEGMENT,OFSET,10);OFSET+=2;
pokeb(SEGMENT,OFSET,5);OFSET+=2;
pokeb(SEGMENT,OFSET,0);OFSET+=2;

/*.......WORD.4......*/
pokeb(SEGMENT,OFSET,41);OFSET+=2;
pokeb(SEGMENT,OFSET,20);OFSET+=2;
pokeb(SEGMENT,OFSET,10);OFSET+=2;
pokeb(SEGMENT,OFSET,0);OFSET+=2;

/*.......WORD.5......*/
pokeb(SEGMENT,OFSET,81);OFSET+=2;
pokeb(SEGMENT,OFSET,40);OFSET+=2;
pokeb(SEGMENT,OFSET,20);OFSET+=2;
pokeb(SEGMENT,OFSET,0);OFSET+=2;

/*.......WORD.6......*/
pokeb(SEGMENT,OFSET,161);OFSET+=2;
pokeb(SEGMENT,OFSET,80);OFSET+=2;
pokeb(SEGMENT,OFSET,8);OFSET+=2;
pokeb(SEGMENT,OFSET,0);OFSET+=2;
```

```
printf("\nPut CLEAR");

for(i=0;i<4,i++)
{
    pokeb(SEGMENT,OFSET,0);OFSET+=2;
}

for(i=0;i<2;i++)
{
    pokeb(SEGMENT,OFSET,94);OFSET+=2;
    pokeb(SEGMENT,OFSET,145);OFSET+=2;
    pokeb(SEGMENT,OFSET,131);OFSET+=2;
    pokeb(SEGMENT,OFSET,233);OFSET+=2;
}

printf("\nPut SYN");

    pokeb(SEGMENT,OFSET,62);OFSET+=2;
    pokeb(SEGMENT,OFSET,75);OFSET+=2;
    pokeb(SEGMENT,OFSET,168);OFSET+=2;
    pokeb(SEGMENT,OFSET,27);OFSET+=2;

for(i=0;i<2;i++)
    {
        pokeb(SEGMENT,OFSET,94);OFSET+=2;
        pokeb(SEGMENT,OFSET,145);OFSET+=2;
        pokeb(SEGMENT,OFSET,131);OFSET+=2;
        pokeb(SEGMENT,OFSET,233);OFSET+=2;
    }
printf("\nPut CLEAR");
```

```
        for(i=0;i<4,i++)
        {
                pokeb(SEGMENT,OFSET,0);OFSET+=2;
        }

        printf("\nEnd");
}
```

4.8 โปรแกรม putdata.c

```
# include<call_alp.h>
void main(void)
{
        int i;
        char bytes,fname[11];
        unsigned OFSET = 0;
        FILE *fp;
        printf("\n Read file name");
        gets(fnames);
        if((fp=fopen(fname, "rb"))==NULL)
    {
        printf("\n Can not open file");
        exit(0);
    }
        printf("\nBegin Work");
        for(i=0;i<72;i++)
    {
        pokeb(SEGMENT,OFSET,85);
        OFSET+=2;
    }

printf("\nBegin Work");
        pokeb(SEGMENT,OFSET,62);OFSET+=2;
        pokeb(SEGMENT,OFSET,75);OFSET+=2;
        pokeb(SEGMENT,OFSET,168);OFSET+=2;
        pokeb(SEGMENT,OFSET,27);OFSET+=2;
printf("\nBegin Work");
    for(i=76;i<204;i++)
    {
        bytes = fgetc(fp);
        pokeb(SEGMENT,OFSET,bytes);
```

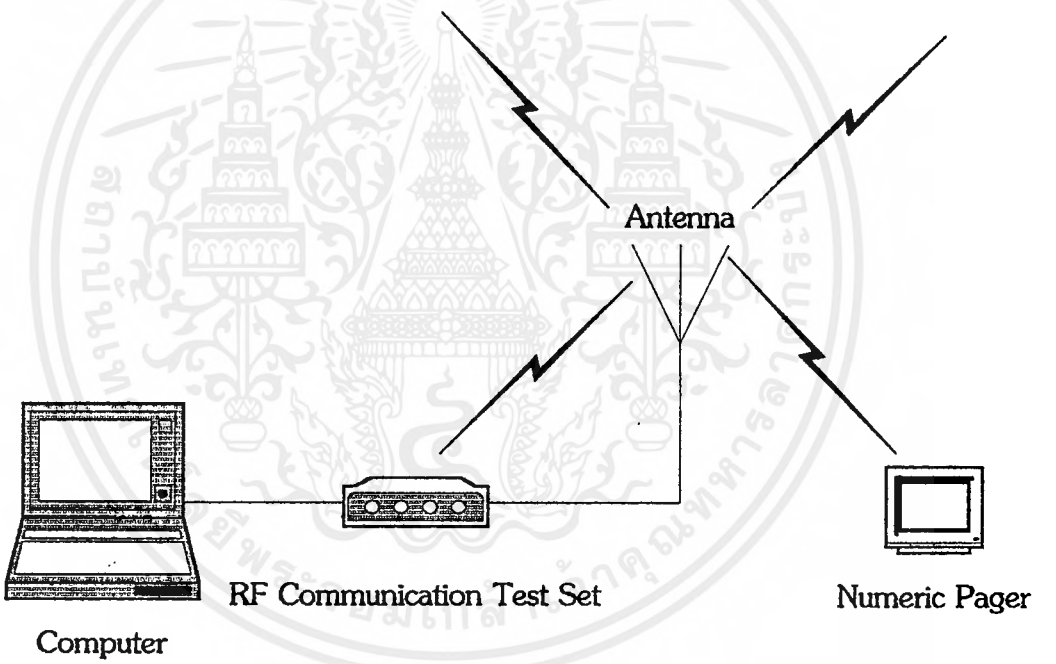
```
            OFSET+=2;
        }
        fclose(fp);
        printf("\n Compleat");
    }
```

บทที่ 5

สรุปผลการทดลอง

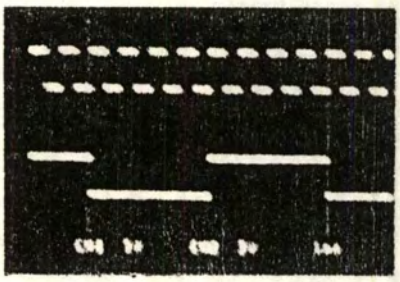
ในบทนี้จะกล่าวถึงการทดลองส่งเพจเจอร์ โดยทำการส่งข้อมูลจะใช้เครื่องทดสอบความถี่วิทยุสื่อสาร (RF Communication Test Set) เป็นตัวส่งข้อมูลออกไป วิธีการเริ่มต้นคือทำการเขียนจาก File ที่เก็บลงในแรมของ Pager Encoder Card แล้วจึงส่งต่อไปยังเครื่องทดสอบความถี่วิทยุสื่อสารเพื่อทำการ Modulate ให้เป็นการส่งแบบ FSK และทำการส่งไปยังเพจเจอร์แบบ Numeric เพื่อทำการรับข้อมูลต่อไป

ในการทดลองจึงได้วัดสัญญาณเอาท์พุทจากภาคต่างๆของวงจรที่ใช้งาน เช่น ภาคอ้างอิงสัญญาณนาฬิกา โดยจะนำเอาสัญญาณเหล่านี้มาเปรียบเทียบเพื่อทำการวิเคราะห์แก้ไข และตรวจสอบโดยให้แสดงผลออกทางหน้าจอคอมพิวเตอร์ ว่าตรงตามข้อมูลที่ได้ป้อนไปหรือไม่

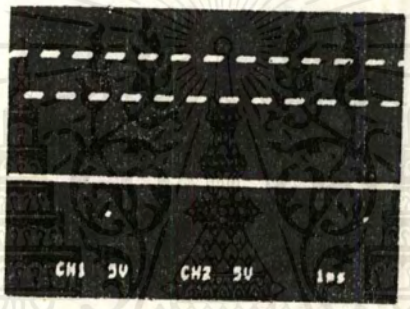


รูปที่ 5.1 รูปการส่งสัญญาณจากเครื่องส่งไปยังเครื่องรับ

จากรูปที่ 5.2 และ 5.3 เป็นรูปสัญญาณที่วัดได้จากการทดลองของภาคอ้างอิงสัญญาณนาฬิกา โดนรูปที่ 5.2 เป็นสัญญาณนาฬิกา 1200 Hz กับ 150 Hz เปรียบเทียบกันโดยจะนำสัญญาณทั้ง 2 นี้มาอ้างอิงกับวงจรต่าง ส่วนในรูปที่ 5.3 เป็นรูปเปรียบเทียบระหว่างสัญญาณนาฬิกา 1200 Hz กับสัญญาณ LOAD ซึ่งจะเห็นได้ว่าสัญญาณ LOAD จะเกิดขึ้นทุกๆ 8 Pulse ของ 1200 Hz เสมอ

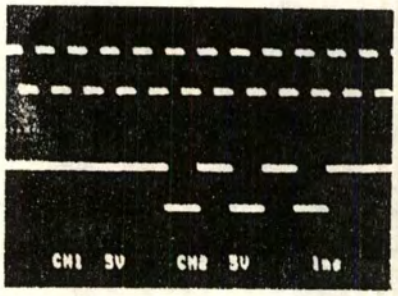


รูปที่ 5.2 เป็นรูปสัญญาณนาฬิกา 1200 Hz กับ 150 Hz



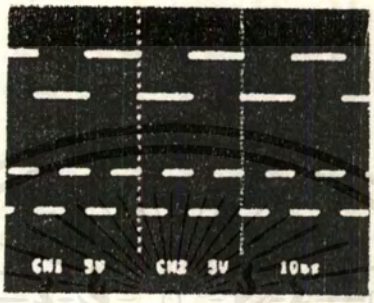
รูปที่ 5.3 เป็นรูปเปรียบเทียบระหว่างสัญญาณนาฬิกา 1200 Hz กับสัญญาณ LOAD

จากรูปที่ 5.4 เป็นการวัดที่ได้จากการทดลองของภาค PISO โดยได้เปรียบเทียบกันกับสัญญาณนาฬิกา 1200 Hz กับสัญญาณเอาท์พุทของภาค PISO โดยเราไม่สามารถจะสังเกตได้ว่าสัญญาณที่ออกมาของภาค PISO นั้นถูกต้องหรือไม่ ต้องไปสังเกตดูจากจอคอมพิวเตอร์เท่านั้น



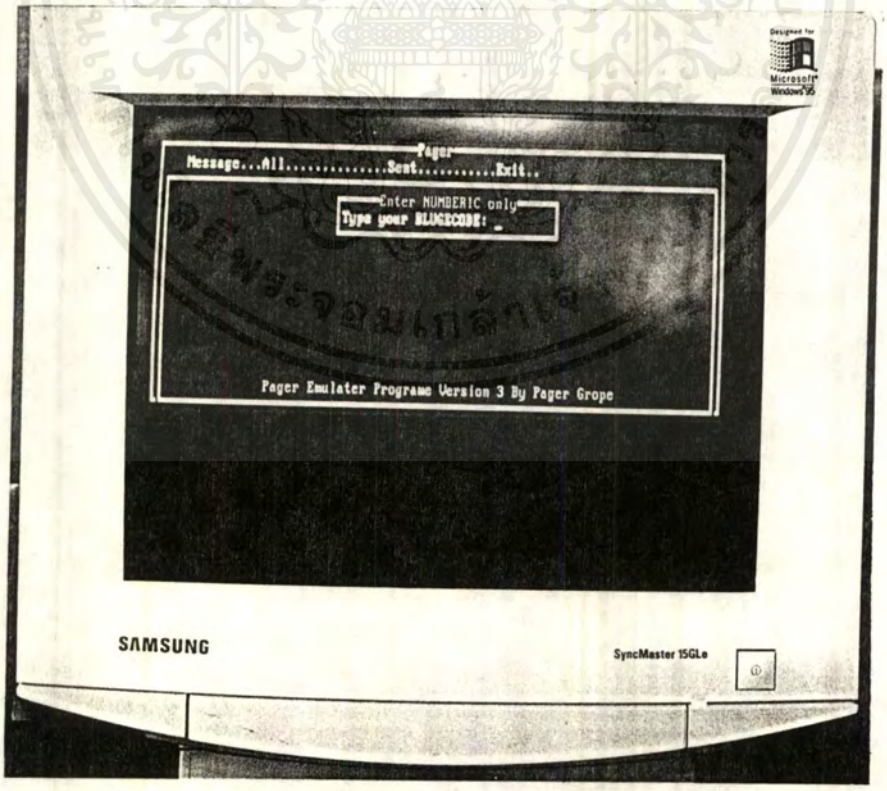
รูปที่ 5.4 เป็นสัญญาณนาฬิกา 1200 Hz กับสัญญาณเอาท์พุทของภาค PISO

จากรูปที่ 5.5 เป็นรูปสัญญาณที่ได้จากการทดลองของภาค Address Counter โดยได้จากการเปรียบเทียบกันระหว่างสัญญาณของ A1 กับ A2 โดยสามารถดูได้ว่า A1 และ A2 นั้นจะต่างกันอยู่โดยใช้ 2 หารสัญญาณลง โดยที่เอา A1 เป็นตัวตั้ง

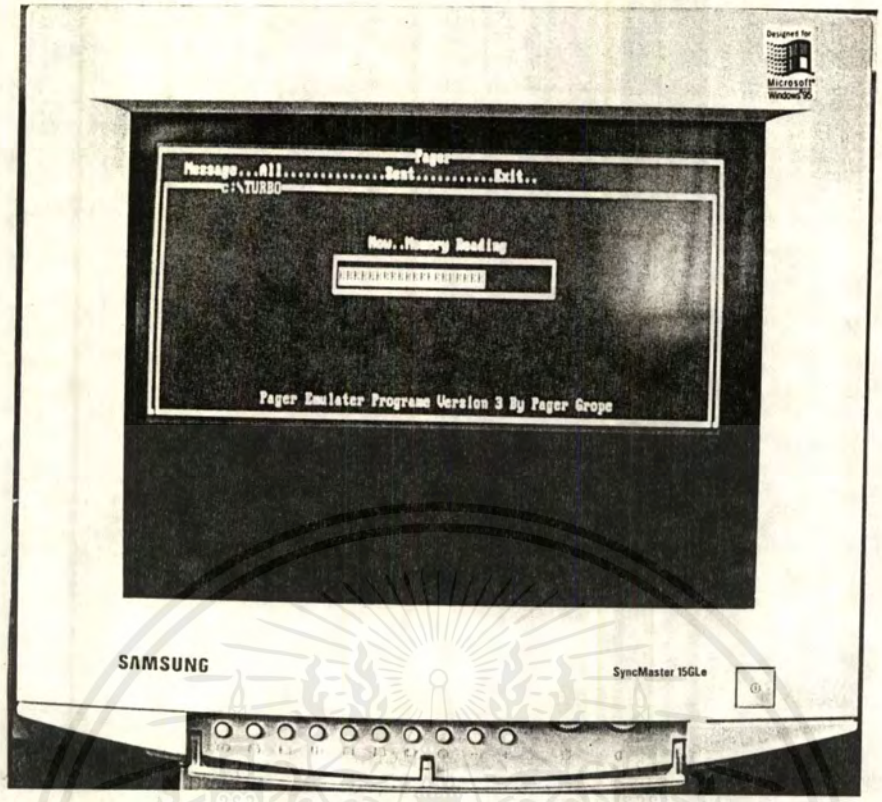


รูปที่ 5.5 เปรียบเทียบระหว่างสัญญาณ A1 กับ สัญญาณ A2

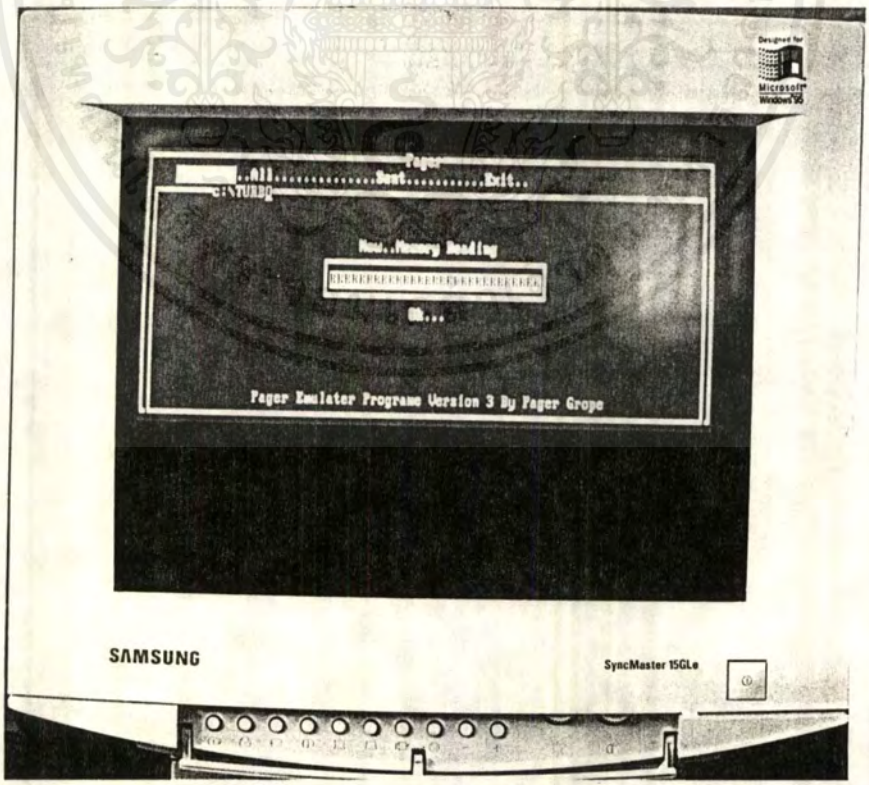
จากรูปที่ 5.6-5.8 เป็นการแสดงผลทางหน้าจอคอมพิวเตอร์



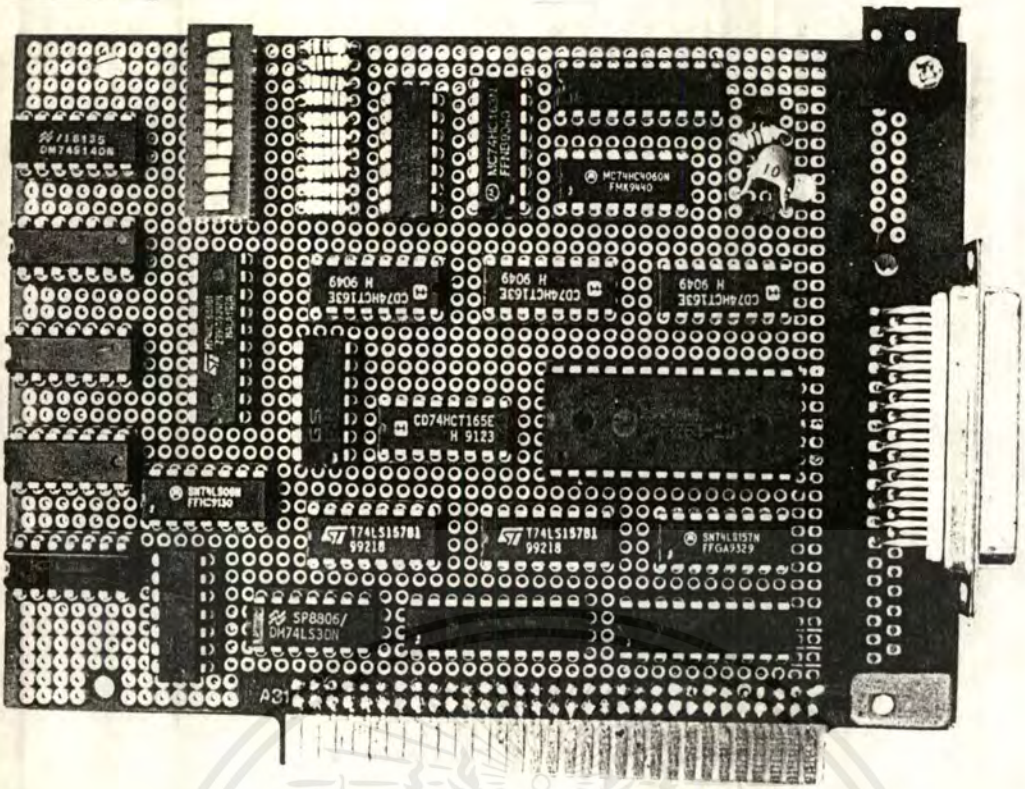
รูปที่ 5.6 เป็นการแสดงให้เห็นถึงหน้าจอที่จะใช้ปฏิบัติงาน



รูปที่ 5.7 แสดงให้เห็นว่ากำลังอ่านข้อมูลจาก RAM อยู่

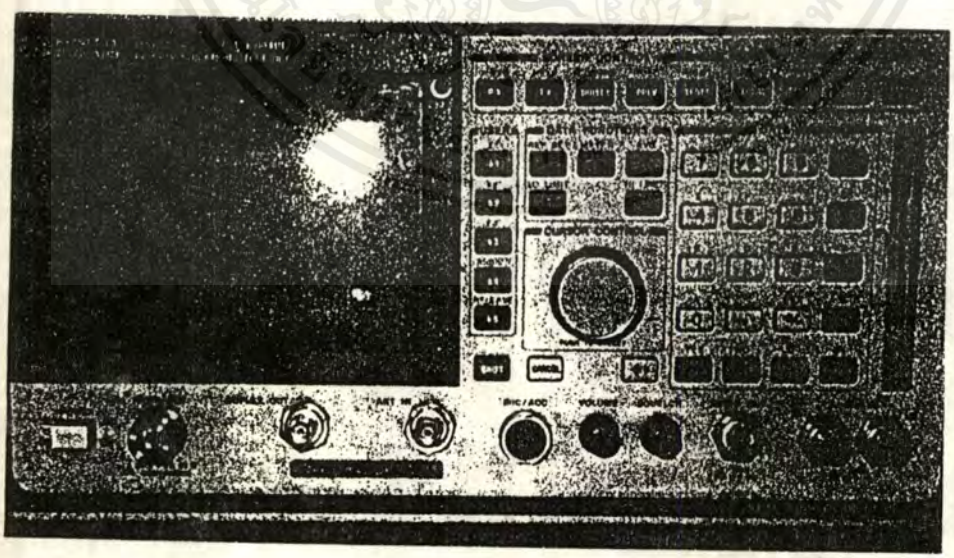


รูปที่ 5.8 เป็นการส่งที่เสร็จแล้ว



รูปที่ 5.9 เป็นรูปของ Encoder Pager Card

ในการทดลองนั้นได้ทดสอบการส่งข้อมูลจากเครื่องทดสอบความถี่วิทยุสื่อสาร (RF Communication Test Set) และจากการเขียนโปรแกรม



รูปที่ 5.10 รูปของเครื่องทดสอบความถี่วิทยุสื่อสาร (RF Communication Test Set)

ภาคผนวก

National Semiconductor

MM54HC688/MM74HC688
8-Bit Magnitude Comparator (Equality Detector)

General Description

This equality detector utilizes advanced silicon-gate CMOS technology to compare bit for bit two 8-bit words and indicates whether or not they are equal. The $\overline{P=Q}$ output indicates equality when it is low. A single active low enable is provided to facilitate cascading of several packages and enable comparison of words greater than 8 bits.

This device is useful in memory block decoding applications, where memory block enable signals must be generated from computer address information.

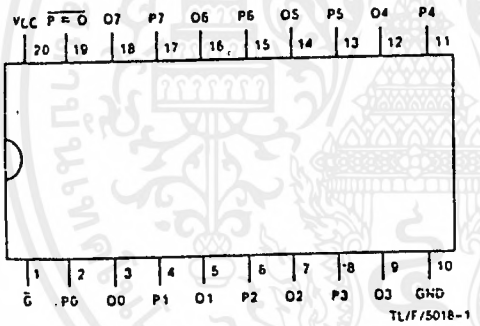
The comparator's output can drive 10 low power Schottky equivalent loads. This comparator is functionally and pin compatible to the 54LS688/74LS688. All inputs are protected from damage due to static discharge by diodes to V_{CC} and ground.

Features

- Typical propagation delay: 20 ns
- Wide power supply range: 2–6V
- Low quiescent current: 80 μA (74 Series)
- Large output current: 4 mA (74 Series)
- Same as 'HC521

Connection and Logic Diagrams

Dual-In-Line Package

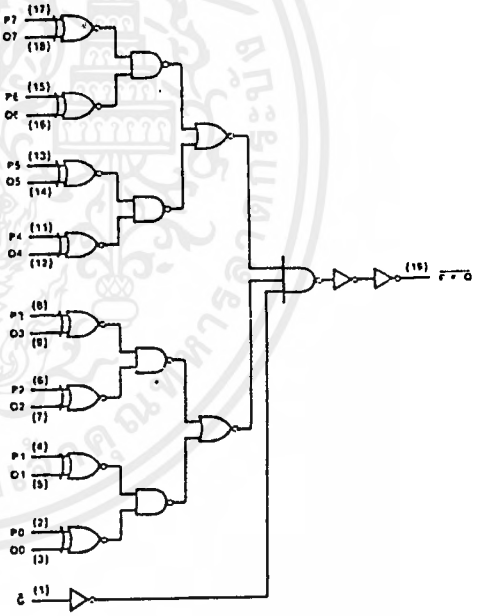


TL/F/5018–1

Top View

Order Number MM54HC688* or MM74HC688*

*Please look into Section 8, Appendix D for availability of various package types.



Truth Table

Inputs		
Data P,Q	Enable $\overline{G}$	$\overline{P=Q}$
P = Q	L	L
P > Q	L	H
P < Q	L	H
X	H	H

Absolute Maximum Ratings (Notes 1 and 2)

If Military/Aerospace specified devices are required, contact the National Semiconductor Sales Office/ Distributors for availability and specifications.

Supply Voltage (V_{CC})	−0.5 to +7.0V
DC Input Voltage (V_{IN})	−1.5 to V_{CC} +1.5V
DC Output Voltage (V_{OUT})	−0.5 to V_{CC} +0.5V
Clamp Diode Current (I_{IK}, I_{OK})	±20 mA
DC Output Current, per pin (I_{OUT})	±25 mA
DC V_{CC} or GND Current, per pin (I_{CC})	±50 mA
Storage Temperature Range (T_{STG})	−65°C to +150°C
Power Dissipation (P_D) (Note 3)	600 mW
S.O. Package only	500 mW
Lead Temp. (T_L) (Soldering 10 seconds)	260°C

Operating Conditions

		Min	Max
Supply Voltage (V_{CC})		2	6
DC Input or Output Voltage (V_{IN}, V_{OUT})		0	V_{CC}
Operating Temp. Range (T_A)			
MM74HC		−40	+85
MM54HC		−55	+125
Input Rise or Fall Times (t_r, t_f)	V_{CC} = 2.0V		1000
	V_{CC} = 4.5V		500
	V_{CC} = 6.0V		400

DC Electrical Characteristics (Note 4)

Symbol	Parameter	Conditions	V_{CC}	T_A = 25°C	74HC T_A = −40 to 85°C	54HC T_A = −55 to 125°C
				Typ	Guaranteed Limits	
V_{IH}	Minimum High Level Input Voltage		2.0V		1.5	1.5	1.5
			4.5V		3.15	3.15	3.15
			6.0V		4.2	4.2	4.2
V_{IL}	Maximum Low Level Input Voltage**		2.0V		0.5	0.5	0.5
			4.5V		1.35	1.35	1.35
			6.0V		1.8	1.8	1.8
V_{OH}	Minimum High Level Output Voltage	V_{IN} = V_{IH} or V_{IL} $\|I_{OUT}\| \le 20\ \mu A$	2.0V	2.0	1.9	1.9	1.9
			4.5V	4.5	4.4	4.4	4.4
			6.0V	6.0	5.9	5.9	5.9
		V_{IN} = V_{IH} or V_{IL} $\|I_{OUT}\| \le 4.0$ mA	4.5V	4.2	3.98	3.84	3.7
		$\|I_{OUT}\| \le 5.2$ mA	6.0V	5.7	5.48	5.34	5.2
V_{OL}	Maximum Low Level Output Voltage	V_{IN} = V_{IH} or V_{IL} $\|I_{OUT}\| \le 20\ \mu A$	2.0V	0	0.1	0.1	0.1
			4.5V	0	0.1	0.1	0.1
			6.0V	0	0.1	0.1	0.1
		V_{IN} = V_{IH} or V_{IL} $\|I_{OUT}\| \le 4.0$ mA	4.5V	0.2	0.26	0.33	0.4
		$\|I_{OUT}\| \le 5.2$ mA	6.0V	0.2	0.26	0.33	0.4
I_{IN}	Maximum Input Current	V_{IN} = V_{CC} or GND	6.0V		±0.1	±1.0	±1.0
I_{CC}	Maximum Quiescent Supply Current	V_{IN} = V_{CC} or GND I_{OUT} = 0 μA	6.0V		8.0	80	160

Note 1: Absolute Maximum Ratings are those values beyond which damage to the device may occur.

Note 2: Unless otherwise specified all voltages are referenced to ground.

Note 3: Power Dissipation temperature derating — plastic "N" package: −12 mW/°C from 65°C to 85°C; ceramic "J" package: −12 mW/°C from 100°C to

Note 4: For a power supply of 5V ±10% the worst case output voltages (V_{OH}, and V_{OL}) occur for HC at 4.5V. Thus the 4.5V values should be used when de with this supply. Worst case V_{IH} and V_{IL} occur at V_{CC} = 5.5V and 4.5V respectively. (The V_{IH} value at 5.5V is 3.85V.) The worst case leakage current (I_I, I_{OZ}) occur for CMOS at the higher voltage and so the 6.0V values should be used.

**V_{IL} limits are currently tested at 20% of V_{CC}. The above V_{IL} specification (30% of V_{CC}) will be implemented no later than Q1, CY'89.

Electrical Characteristics

5V, $T_A = 25°C$, $C_L = 15$ pF, $t_r = t_f = 6$ ns

Symbol	Parameter	Conditions	Typ	Guaranteed Limit	Units
t_{PLH}	Maximum Propagation Delay, any P or Q to Output		21	30	ns
t_{PHL}	Maximum Propagation Delay, Enable to any Output		14	20	ns

Electrical Characteristics

2.0V to 6.0V, $C_L = 50$ pF, $t_r = t_f = 6$ ns (unless otherwise specified)

Symbol	Parameter	Conditions	V_{CC}	$T_A = 25°C$	74HC $T_A = -40$ to $85°C$	54HC $T_A = -55$ to $125°C$	Units
				Typ	Guaranteed Limits		
t_{PLH}	Maximum Propagation Delay, P or Q to Output		2.0V	60	175	220	263	ns
			4.5V	22	35	44	53	ns
			6.0V	19	30	38	45	ns
t_{PLH}	Maximum Propagation Delay, Enable to Output		2.0V	45	120	150	180	ns
			4.5V	15	24	30	36	ns
			6.0V	13	20	25	30	ns
t_{TLH}	Maximum Output Rise and Fall Time		2.0V	30	75	95	110	ns
			4.5V	8	15	19	22	ns
			6.0V	7	13	16	19	ns
	Power Dissipation Capacitance (Note 5)			45				pF
	Maximum Input Capacitance			5	10	10	10	pF

C_{PD} determines the no load dynamic power consumption, $P_D = C_{PD} V_{CC}^2 f + I_{CC} V_{CC}$, and the no load dynamic current consumption, $I_S = C_{PD} V_{CC} f + I_{CC}$.

National Semiconductor

MM54HC4060/MM74HC4060
14 Stage Binary Counter

General Description

The MM54HC4060/MM74HC4060 is a high speed binary ripple carry counter. These counters are implemented utilizing advanced silicon-gate CMOS technology to achieve speed performance similar to LS-TTL logic while retaining the low power and high noise immunity of CMOS.

The 'HC4060 is a 14-stage counter, which device increments on the falling edge (negative transition) of the input clock, and all their outputs are reset to a low level by applying a logical high on their reset input. The 'HC4060 also has two additional inputs to enable easy connection of either an RC or crystal oscillator.

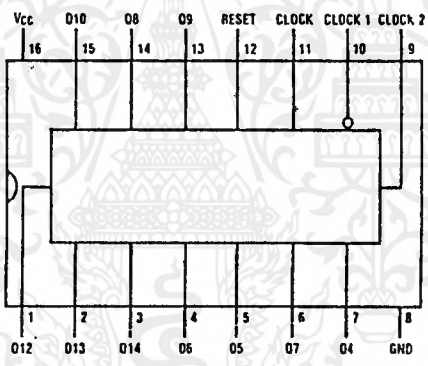
This device is pin equivalent to the CD4060. All inputs are protected from damage due to static discharge by protection diodes to V_{CC} and ground

Features

- Typical propagation delay: 16 ns
- Wide operating voltage range: 2–6V
- Low input current: 1 μA maximum
- Low quiescent current: 80 μA maximum (74 Series)
- Output drive capability: 10 LS-TTL loads

Connection and Logic Diagrams

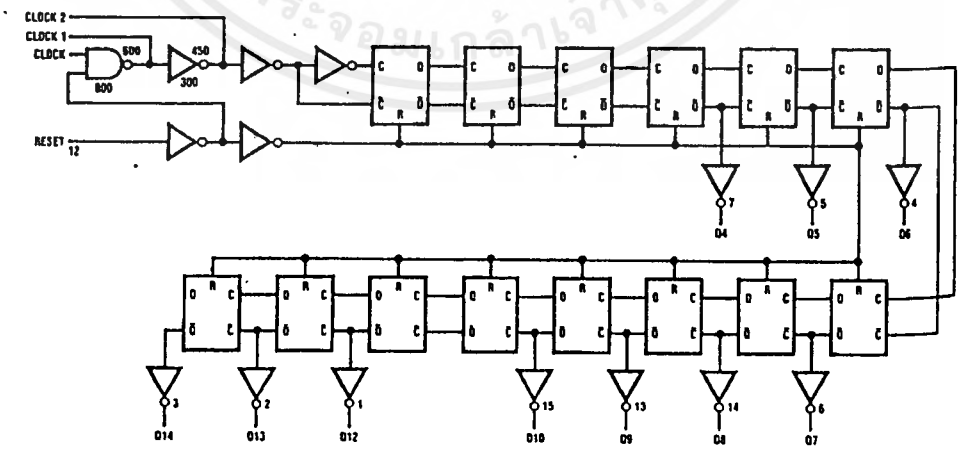
Dual-In-Line Package



Top View

Order Number MM54HC4060* or MM74HC4060*

*Please look into Section 8, Appendix D for availability of various package types.



Absolute Maximum Ratings (Notes 1 & 2)

Military/Aerospace specified devices are required, contact the National Semiconductor Sales Office/ distributors for availability and specifications.

Supply Voltage (V_{CC})	−0.5 to +7.0V
Input Voltage (V_{IN})	−1.5 to V_{CC} +1.5V
Output Voltage (V_{OUT})	−0.5 to V_{CC} +0.5V
Clamp Diode Current (I_{CD})	±20 mA
DC Output Current, per pin (I_{OUT})	±25 mA
DC V_{CC} or GND Current, per pin (I_{CC})	±50 mA
Storage Temperature Range (T_{STG})	−65°C to +150°C
Power Dissipation (P_D)	
(Note 3)	600 mW
S.O. Package only	500 mW
Lead Temperature (T_L)	
(Soldering 10 seconds)	260°C

Operating Conditions

	Min	Max	Units
Supply Voltage (V_{CC})	2	6	V
DC Input or Output Voltage (V_{IN}, V_{OUT})	0	V_{CC}	V
Operating Temp. Range (T_A)			
MM74HCT	−40	+85	°C
MM54HCT	−55	+125	°C
Input Rise or Fall Times (t_r, t_f)			
V_{CC} = 2.0V		1000	ns
V_{CC} = 4.5V		500	ns
V_{CC} = 6.0V		400	ns

DC Electrical Characteristics (Note 4)

Symbol	Parameter		Conditions	V_{CC}	T_A = 25°C	74HC T_A = −40 to 85°C	54HC T_A = −55 to 125°C	Units
					Typ	Guaranteed Limits		
V_{IH}	Minimum High Level Voltage (Not Applicable to Pins 9 & 10)			2.0V		1.5	1.5	1.5	V
				4.5V		3.15	3.15	3.15	V
				6.0V		4.2	4.2	4.2	V
V_{IL}	Maximum Low Level Input Voltage ** (Not Applicable to Pins 9 & 10)			2.0V		0.5	0.5	0.5	V
				4.5V		1.35	1.35	1.35	V
				6.0V		1.8	1.8	1.8	V
V_{OH}	Minimum High Level Output Voltage		V_{IN} = V_{IH} or V_{IL} $\lvert I_{OUT}\rvert \le 20\ \mu A$	2.0V	2.0	1.9	1.9	1.9	V
				4.5V	4.5	4.4	4.4	4.4	V
				6.0V	6.0	5.9	5.9	5.9	V
		Except Pins 9 & 10	V_{IN} = V_{IH} or V_{IL} $\lvert I_{OUT}\rvert \le 4.0$ mA	4.5V	4.2	3.98	3.84	3.7	V
			$\lvert I_{OUT}\rvert \le 5.2$ mA	6.0V	5.7	5.48	5.34	5.2	V
		Pins 9 & 10	V_{IN} = V_{IH} or V_{IL} $\lvert I_{OUT}\rvert$ = 0.4 mA			3.98	3.84	3.7	V
			$\lvert I_{OUT}\rvert$ = 0.52 mA			5.48	5.34	5.2	V
V_{OL}	Maximum Low Level Output Voltage		V_{IN} = V_{IH} or V_{IL} $\lvert I_{OUT}\rvert \le 20\ \mu A$	2.0V	0	0.1	0.1	0.1	V
				4.5V	0	0.1	0.1	0.1	V
				6.0V	0	0.1	0.1	0.1	V
		Except Pins 9 & 10	V_{IN} = V_{IH} or V_{IL} $\lvert I_{OUT}\rvert \le 4.0$ mA	4.5V	0.2	0.26	0.33	0.4	V
			$\lvert I_{OUT}\rvert \le 5.2$ mA	6.0V	0.2	0.26	0.33	0.4	V
		Pins 9 & 10	V_{IN} = V_{IH} or V_{IL} $\lvert I_{OUT}\rvert$ = 0.4 mA			0.26	0.33	0.4	V
			$\lvert I_{OUT}\rvert$ = 0.52 mA			0.26	0.33	0.4	V
I_{IN}	Maximum Input Current		V_{IN} = V_{CC} or GND	6.0V		±0.1	±1.0	±1.0	μA
I_{CC}	Maximum Quiescent Supply Current		V_{IN} = V_{CC} or GND I_{OUT} = 0 μA	6.0V		8.0	80	160	μA

Note 1: Maximum Ratings are those values beyond which damage to the device may occur.

Note 2: Unless otherwise specified all voltages are referenced to ground.

Note 3: Power Dissipation temperature derating: plastic "N" package: −12 mW/°C from 65°C to 85°C ceramic "J" package: −12 mW/°C from 100°C to 125°C

Note 4: For a power supply of 5V ±10% the worst case output voltages (V_{OH}, and V_{OL}) occur for HC at 4.5V. Thus the 4.5V values should be used when designing with this supply. Worst case V_{IH} and V_{IL} occur at V_{CC} = 5.5V and 4.5V respectively. (The V_{IH} value at 5.5V is 3.85V.) The worst case leakage current (I_{IN} and I_{OZ}) occur for CMOS at the higher voltage and so the 6.0V values should be used.

** These are currently tested at 20% of V_{CC}. The above V_{IL} specification (30% of V_{CC}) will be implemented no later than Q1, CY'89.

AC Electrical Characteristics

V_{CC} = 5V, T_A = 25°C, C_L = 15 pF, $t_r = t_f$ = 6 ns

Symbol	Parameter	Conditions	Typ	Guaranteed Limit	Units
f_{MAX}	Maximum Clock Frequency			30	MHz
t_{PHL}, t_{PLH}	Maximum Propagation Delay to Q_4	(Note 5)	40	20	ns
t_{PHL}, t_{PLH}	Maximum Propagation Delay to any Q		16	40	ns
t_{REM}	Minimum Reset Removal Time		10	20	ns
t_W	Minimum Pulse Width		10	16	ns

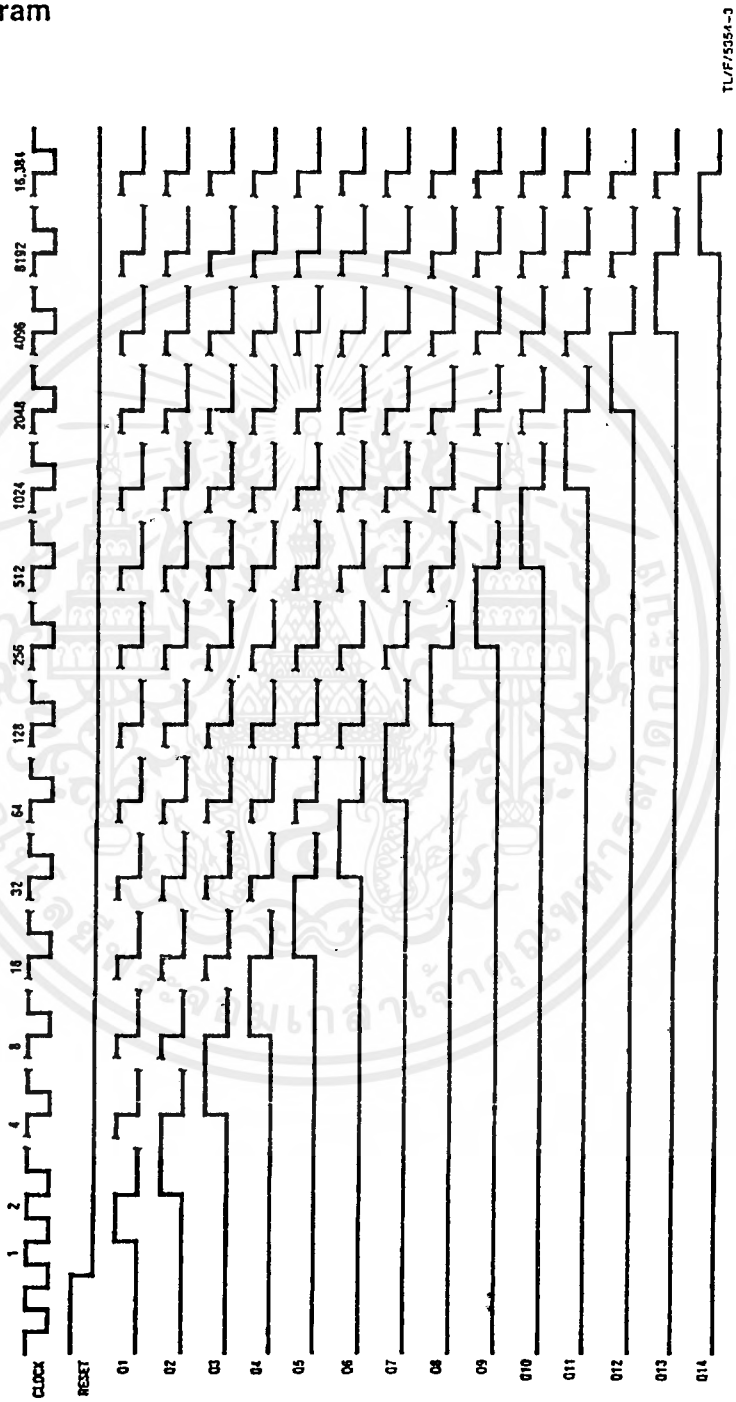
AC Electrical Characteristics V_{CC} = 2.0V to 6.0V, C_L = 50 pF, $t_r = t_f$ = 6 ns (unless otherwise specified)

Symbol	Parameter	Conditions	V_{CC}	T_A = 25°C	T_A74HC T_A = −40 to 85°C	T_A54HC T_A = −55 to 125°C
				Typ	Guaranteed Limits	
f_{MAX}	Maximum Operating Frequency		2.0V		6	5	4
			4.5V0		30	24	20
			6.0V		35	28	24
t_{PHL}, t_{PLH}	Maximum Propagation Delay Clock to Q_4		2.0V	120	380	475	171
			4.5V	42	76	95	114
			6.0V	35	65	81	97
t_{PHL}	Maximum Propagation Delay Reset to any Q		2.0V	72	240	302	358
			4.5V	24	48	60	72
			6.0V	20	41	51	61
t_{PHL}, t_{PLH}	Maximum Propagation Delay Between Stages Q_n to Q_{n+1}		2.0V		125	156	188
			4.5V		25	31	38
			6.0V		21	26	31
t_{REM}	Mimimum Reset Removal Time		2.0V.		100	125	150
			4.5V		20	25	30
			6.0V		17	21	25
t_W	Mimimum Pulse Width		2.0V		80	100	120
			4.5V		16	20	24
			6.0V		14	17	20
t_r, t_f	Maximum Input Rise and Fall Time		2.0V		1000	1000	1000
			4.5V		500	500	500
			6.0V		400	400	400
t_{THL}, t_{TLH}	Maximum Output Rise and Fall Time		2.0V	30	75	95	110
			4.5V	10	15	19	22
			6.0V	9	13	16	19
C_{PD}	Power Dissipation Capacitance (Note 6)	(per package)		55			
C_{IN}	Maximum Input Capacitance			5	10	10	10

Note 5: Typical Propagation delay time to any output can be calculated using: t_p = 17 + 12(N−1) ns; where N is the number of the output Q_N at V_{CC}.

Note 6: C_{PD} determines the no load dynamic power consumption, $P_D = C_{PD} V_{CC}^2 f + I_{CC} V_{CC}$, and the no load dynamic current consumption $I_S = C_{PD} V_{CC} f + I_{CC}$.

Timing Diagram



1. Hewlet Packard. "HP 8920 RF Communication Test Set User's Guide",1991.
2. British Telecom. "A Standard Code for Radio Paging Report of Post Office Code Standard Advisory Group (POCSAG)", June 1978.
3. ปริญญานิพนธ์เรื่อง การ์ดถอดรหัสเพจเจอร์บนไมโครคอมพิวเตอร์ ภาคเทคนิคอุตสาหกรรม คณะวิศวกรรมศาสตร์ สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง
4. Advanced MS-DOS ของพงษ์ระพี เตชพาพงษ์
5. การอินเตอร์เฟส IBM/PC ของธานินทร์ ฉาวรศาสนวงศ์,ทินกร ดุ๊ก
6. การเขียนภาษา C สำหรับวิศวกรรมของ ธันวา ศรีประโมง
7. การเขียนชุดคำสั่งภาษา C ของ มัณฑนา ปราการสมุทร
8. การเขียนโปรแกรมคอมพิวเตอร์ด้วยเทอร์โบซีของ มนตรี พจนารถลาวัณย์
9. การเขียนภาษาซีด้วยตนเองของ วรรณวิภา ติตถะสิริ