



การ์ดเข้ารหัสเพจเจอร์บนไมโครคอมพิวเตอร์
PAGER ENCODER CARD ON MICROCOMPUTER

โดย

นายวิศักดิ์ พงษ์พัฒนาชาติ
นายณรัตน์ ศรีประโชติ
นายสมศักดิ์ เดชสุชน

๒๕๓๘
วัน เดือน ปี..... ๒๕๓๘
เลขทะเบียน..... ๐๓๘๐๘๐
เลขเรียกหนังสือ..... ๓๘๐๘๐ ๓๘๘๐

ปริญญานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาอุตสาหกรรมศาสตรบัณฑิต
สาขาเทคโนโลยีโทรคมนาคม
คณะวิศวกรรมศาสตร์
สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง
ปีการศึกษา ๒๕๓๘

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า
ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุก ๐๓๘๐๘๐ ใช้



หัวข้อปริญญานิพนธ์ การ์ดเข้ารหัสเพจเจอร์บนไมโครคอมพิวเตอร์
Pager Encoder Card on Microcomputer

ชื่อนักศึกษา	นายทวีศักดิ์ พงศ์พัฒนาชาติ	37012052
	นายณพรัตน์ ศรีประโชติ	37012056
	นายสมศักดิ์ เศรษฐชน	36012076

อาจารย์ที่ปรึกษา อ.กฤดากร กล่อมการ

ภาควิชา เทคนิคอุตสาหกรรม

ปีการศึกษา 2538

คณะวิศวกรรมศาสตร์ สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง อนุมัติ
ให้รับปริญญานิพนธ์ฉบับนี้ เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรอุตสาหกรรมศาสตรบัณฑิต

คณะกรรมการสอบปริญญานิพนธ์

.....ประธานกรรมการ
(.....)
.....กรรมการ
(.....)
.....กรรมการ
(.....)
.....กรรมการ
(.....)
.....กรรมการ
(.....)

ลิขสิทธิ์ของคณะวิศวกรรมศาสตร์ สถาบันเทคโนโลยีพระจอมเกล้าลาดกระบัง

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า
ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

Pager Encoder Card on Microcomputer.

BY MR.TAWEESAK PHONGPHATANACHAT 37012052
MR.NOPPARAT SEPRACHOTE 37012056
MR.SOMSAK DETSUCHON 36012076

ADVISER MR. KITDARORN KLOMKARN

YEAR 1996

ABSTRACT

This thesis is described about construction pager encoder card on microcomputer. The function of this card is simulation pager signal. It send basedband pager from memory at bit rate about 1200 Hz. The program implement by C language for control a format of data in POCSAG before keep in memory.

การเข้ารหัสเพจเจอร์บนไมโครคอมพิวเตอร์

โดย	นาย ทวีศักดิ์ พงศ์พัฒนาชาติ	37012052
	นาย นพรัตน์ ศรีประโชติ	37012056
	นาย สมศักดิ์ เดชสุชน	36012076

อาจารย์ที่ปรึกษา อ.กฤดากร กล่อมการ

ปีการศึกษา 2538

บทคัดย่อ

ปริญญานิพนธ์นี้เสนอเกี่ยวกับการเข้ารหัสเพจเจอร์บนไมโครคอมพิวเตอร์ การเข้ารหัสเพจเจอร์เป็นการเขียนแบบเครื่องส่งเพจเจอร์ การเข้ารหัสเพจเจอร์จะส่งสัญญาณเบสแบนด์เป็นแบบอนุกรมที่บิตเรท 1200 Hz โดยการออกแบบซอฟต์แวร์ใช้ภาษา C เป็นตัวควบคุมและจัดรูปแบบให้อยู่ในรหัส POCSAG ก่อนจะเก็บลงในหน่วยความจำ

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

กิตติกรรมประกาศ

โครงการปริญญานิพนธ์สำเร็จได้ด้วยดี โดยได้รับคำที่ปรึกษาและคำแนะนำ ตลอดจนเครื่องมือต่างๆจาก อาจารย์กฤตากร กล่อมการ รวมทั้งคำติชมจากอาจารย์ประจำภาควิชาทุกท่าน , สถาบันพระจอมเกล้าเจ้าคุณทหารลาดกระบัง ศูนย์นนทบุรี เป็นสถานที่วิจัยและที่พักอาศัย และที่ลืมไม่ได้คือเพื่อนห้อง 2 M ที่คอยให้กำลังใจทางคณะผู้จัดทำจึงขอขอบพระคุณมา ณ โอกาสนี้ด้วย



สารบัญ

บทที่ 1 บทนำ	1
โครงสร้างของระบบ	2
การขยายพื้นที่	3
บทที่ 2 POCSAG	4
รูปแบบของสัญญาณเพจเจอร์	4
สัญญาณพัลส์ส่วนหน้า	4
โครงสร้างของแบทช์ข้อมูล	5
รหัสคำการชิงโครโนซ์	6
รหัสคำหมายเลขเรียกขาน	7
รหัสคำของข่าวสาร	8
รหัสคำเทียม	12
บทที่ 3 ภาครับสัญญาณเพจเจอร์	13
หลักการทั่วไปของภาครับเพจเจอร์	14
วงจรลิมิตเตอร์	16
วงจรกู้สัญญาณนาฬิกา	17
วงจรตรวจจับเฟรมชิง	18
SIPO	21
การกำเนิดสัญญาณอ่าน, เขียนแรมและสัญญาณนับแอดเดรส	21
แรมอีเอ็มแวลเตอร์	24
ภาคตรวจจับหมายเลขเรียกขาน	27
ภาคสร้างสัญญาณอินเตอร์รัพท์แอดเดรสปลักโคค	27
คอมพิวเตอร์	31
บทที่ 4 ภาคส่งสัญญาณเพจเจอร์	34
การทำงานของภาคส่งเพจเจอร์	34
PISO	35
ภาคอ้างอิงสัญญาณนาฬิกา	36
ภาคนับแอดเดรส	37
ภาคเลือกพอร์ตและอินาเปิลเซน	39

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า
ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

RAM	40
การจัดข้อมูลเพื่อการส่ง	42
การทำงานของโปรแกรม	43
โปรแกรม menu.c	48
โปรแกรม P.c	68
โปรแกรม puta.c	69
โปรแกรม putb.c	75
โปรแกรม putdata.c	81
บทที่ 5 ผลการทดลอง	82

ภาคผนวก รายละเอียดไอซี
บรรณานุกรม



บทที่ 1

บทนำ

ทฤษฎีทั่วไปของวิทยุติดตามตัว (Pager)

ในระบบการติดต่อสื่อสารจากอดีตถึงปัจจุบันได้มีการพัฒนาอุปกรณ์ที่ใช้ในการสื่อสารมากมายหลายชนิดตั้งแต่อุปกรณ์สื่อสารแบบมีสายจนกระทั่งถึงอุปกรณ์แบบไร้สายและจากความเจริญก้าวหน้าทางเทคโนโลยีก็ได้มีการพัฒนาอุปกรณ์เหล่านี้เพื่ออำนวยความสะดวกแก่ผู้ใช้งานขึ้นไม่ว่าจะเป็นการเพิ่มรัศมีในการติดต่อสื่อสาร, การลดขนาดเพื่อให้เล็กลงกระทัดรัด

วิทยุติดตามตัวหรือเพจเจอร์ (Pager) เป็นเครื่องมือสื่อสารชนิดเคลื่อนที่แบบหนึ่งที่นิยมใช้สำหรับผู้ที่มีภาระหน้าที่ไม่ค่อยประจำที่แต่จำเป็นต้องมีการติดต่อได้ตลอดเวลาเช่นเดียวกับวิทยุโทรศัพท์แบบพกพาตั้งแต่มีลักษณะการสื่อสารในทิศทางเดียวจากผู้ส่งไปยังผู้รับ

การติดต่อระหว่างผู้ส่งกับผู้รับอาจเป็นการแจ้งให้ผู้รับติดต่อกลับไปยังผู้ส่งหรืออาจเป็นการส่งข่าวสารที่ต้องการไปยังผู้รับโดยตรงก็ได้ เพจเจอร์ที่ใช้แบ่งออกเป็น 5 แบบ

- 1.Voice Pager จะทำการส่งข่าวสารเป็นเสียงพูดไปยังผู้รับ
- 2.Digital Display Pager จะทำการส่งข่าวสารเป็นตัวเลขซึ่งตัวเลขนี้จะแทนข่าวสารที่มีความหมายเป็นที่เข้าใจกันระหว่างผู้รับกับผู้ส่ง
- 3.Alpha numeric Pager แบบนี้สามารถที่จะส่งข่าวสารเป็นได้ทั้งตัวเลขและอักษรซึ่งเป็นการช่วยให้ส่งข่าวสารได้ละเอียดขึ้นแต่จำเป็นต้องใช้ระบบที่พิเศษ
- 4.Tone-alert Pager เป็นการส่งสัญญาณเสียงเพื่อเป็นการบอกให้ผู้รับติดต่อไปยังศูนย์เพื่อรับข่าวสารอีกทีหนึ่ง
- 5.Dual Address Pager จะส่งสัญญาณเสียงได้ 2 ลักษณะเพื่อให้ผู้รับทราบว่าติดต่อไปยังที่ใด

เวลาที่ใช้ในการส่งข่าวสารด้วยเพจเจอร์ในแต่ละแบบจะแตกต่างกันคือเพจเจอร์แบบใช้เสียงพูดจะใช้เวลาตั้งแต่ 10 วินาทีลงมา ส่วนเพจเจอร์แบบที่ไม่ใช้เสียงพูดจะใช้เวลาในการส่งเป็นมิลลิวินาที

ข้อดีของเพจเจอร์เมื่อเปรียบเทียบกับระบบการสื่อสารเคลื่อนที่ชนิด 2 ทิศทางคือ

- 1.สามารถพกติดตัวได้สะดวกเนื่องจากขนาดเล็กลงและมีน้ำหนักเบาและไม่ต้องมีแหล่งจ่ายไฟฟ้าจากภายนอก
- 2.ค่าใช้จ่ายในการเช่าใช้ถูกกว่ามาก

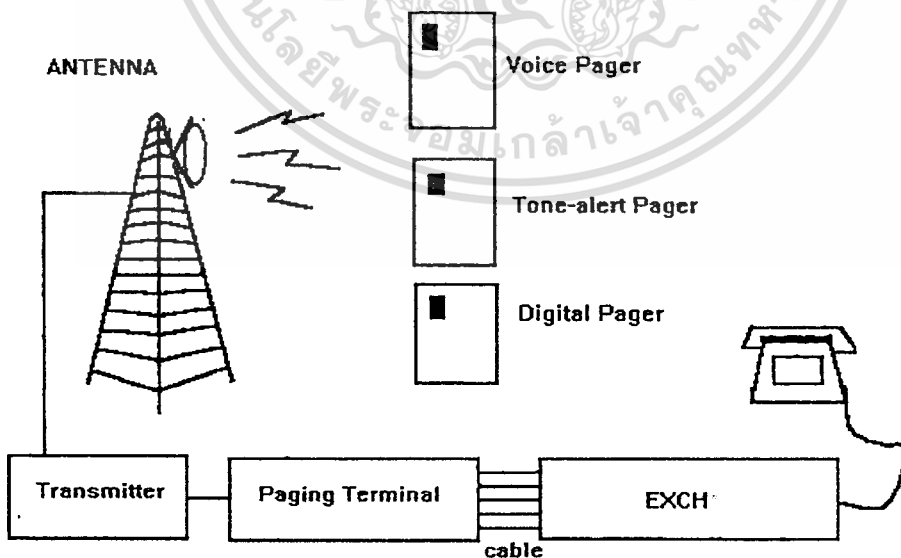
เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้าไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

3. มีจำนวนผู้เข้าใช้ต่อหนึ่งความถี่ที่มากกว่า
4. เครื่องรับสามารถใช้ร่วมกันได้โดยส่วนใหญ่จากหลายๆยี่ห้อ
5. ที่ตัวเพจเจอร์แทบไม่ต้องการบำรุงรักษา
6. ไม่ต้องการติดตั้งใดๆในส่วนของผู้ใช้
7. สามารถบันทึกข่าวสารไว้ได้ในตัวเครื่อง

โครงสร้างของระบบเพจเจอร์

เพจเจอร์ใช้การสื่อสารด้วยความถี่วิทยุเช่นเดียวกับระบบวิทยุโทรศัพท์ ดังนั้น โครงสร้างของระบบจึงไม่แตกต่างกันนักดังแสดงในรูปที่ 1.1 ซึ่งประกอบด้วยส่วนต่างๆดังนี้

1. ระบบโทรศัพท์ เป็นส่วนที่ผู้ส่งสามารถส่งข่าวสารผ่านทางระบบโทรศัพท์ไปยังศูนย์เพื่อส่งต่อไปยังผู้รับที่ต้องการ
2. ศูนย์กลางของระบบ (Pager Terminal) เป็นศูนย์กลางในการติดต่อสื่อสาร โดยมีคอมพิวเตอร์เป็นผู้จัดการทุกอย่างเกี่ยวกับการให้บริการการรับส่งสารจากผู้ส่งไปยังผู้รับ ซึ่งถือว่าเป็นส่วนที่สำคัญของระบบ
3. เครื่องส่งทำหน้าที่ส่งข่าวสารในรูปของคลื่นวิทยุผ่านสายอากาศไปยังเครื่องรับ ซึ่งเครื่องส่งอาจตั้งอยู่ที่ศูนย์หรือที่อื่นก็ได้

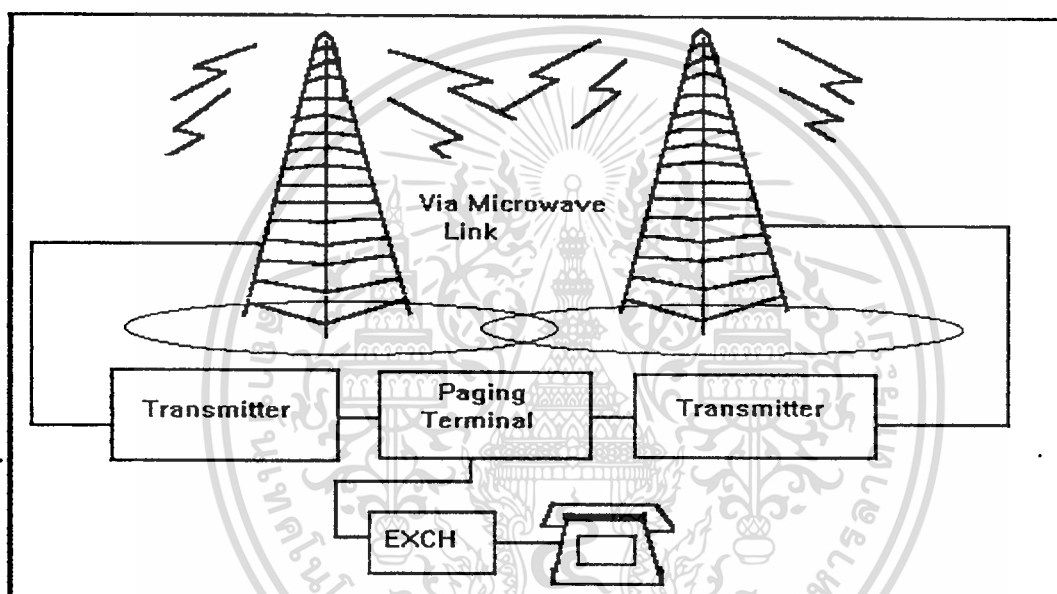


รูปที่ 1.1 โครงสร้างของวิทยุติดตามตัว

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

การขยายพื้นที่ของระบบ

เพื่อให้พื้นที่ในการให้บริการแก่ผู้ใช้กว้างออกไป จำเป็นต้องเพิ่มสถานีส่งกระจายออกไปตามจุดต่างๆ ให้ครอบคลุมอาณาบริเวณที่ต้องการ โดยความถี่ที่ส่งออกไปในเครื่องส่งแต่ละเครื่องยังคงเป็นความถี่เดิม ส่วนข่าวสารจะมีการติดต่อถึงกันด้วยคลื่นวิทยุอีกความถี่หนึ่งหรือใช้ระบบไมโครเวฟปัญหาในเรื่องการรบกวนของเครื่องส่งที่ส่งความถี่เดียวกันในพื้นที่เหลื่อมกันนั้นใช้วิธีปรับเฟสของคลื่นที่ส่งออกอากาศให้มีเฟสตรงกัน เมื่อคลื่นมาถึงพื้นที่ดังกล่าวลักษณะของการขยายระบบเป็นดังรูปที่ 1.2



รูปที่ 1.2 ลักษณะการขยายพื้นที่ของระบบ

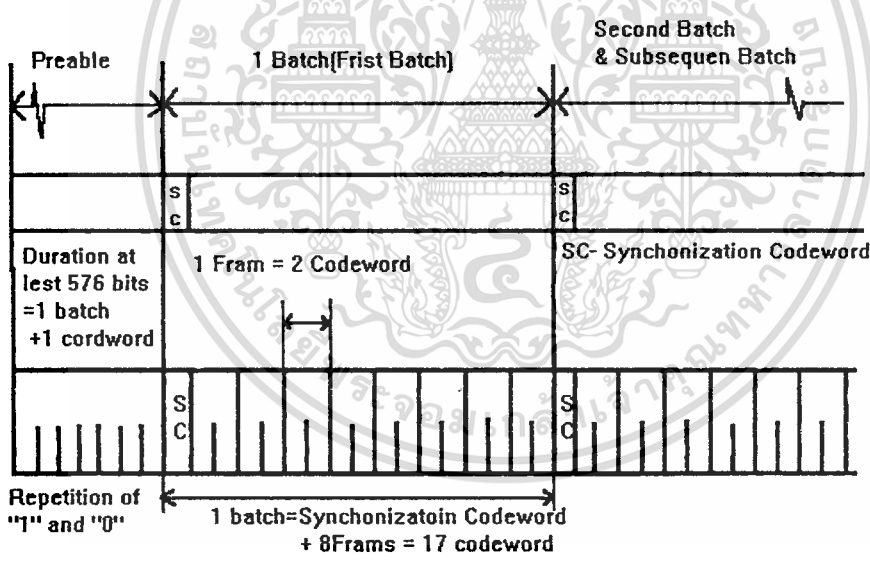
บทที่ 2

POCSAG

(Post Office Code Standardization Advisory Group)

รูปแบบของสัญญาณเพจเจอร์

สัญญาณข่าวสารที่จะส่งไปยังผู้รับนั้นเป็นสัญญาณรหัสข้อมูลเลขฐานสองเรียงต่อเนื่องกัน ซึ่งรหัสข้อมูลที่ใช้เป็นรหัสของ POCSAG (Post Office Code Standardization Advisory Group) ซึ่งเป็นรูปแบบและรหัสมาตรฐานสำหรับวิทยุติดตามตัวที่ถูกกำหนดโดย CCIR รูปแบบของสัญญาณข่าวสารที่ใช้ส่งประกอบไปด้วยสัญญาณพัลส์ส่วนหน้า (preamble) จำนวน 576 บิตและตามด้วยรหัสคำตั้งแต่หนึ่งชุดขึ้นไป (batch structure) ซึ่งแต่ละชุดจะเริ่มด้วยรหัสการซิงโครไนซ์ (synchronization codeword) รูปแบบของสัญญาณแสดงได้ดังรูปที่ 2.1



รูปที่ 2.1 รูปแบบของสัญญาณต่างๆ

สัญญาณพัลส์ส่วนหน้า (Preamble)

ประกอบด้วยรูปแบบที่มีลักษณะของการบิตตรงข้ามกันคือ "1010.....1010" จะถูกส่งเป็นจำนวนอย่างน้อย 576 บิต (มีค่าเท่ากับ 1 batch บวก 1 codeword) สัญญาณพัลส์ส่วนหน้านี้ถูกใช้เป็นตัวช่วยในการตรวจหาจุดเริ่มต้นของการส่งเพจเจอร์

โครงสร้างของแบทช์ข้อมูล (Batch Structure)

รหัสคำจะถูกส่งลงในแบทช์โดยที่เฟรมแรกเป็นเฟรม 0 และเฟรมสุดท้ายเป็นเฟรมที่ 7
รหัสคำที่บรรจุอยู่ในแบทช์แบ่งออกเป็น 4 ชนิดดังนี้

- 1.รหัสคำการซิงโครไนซ์ (Synchronization Codewords)
- 2.รหัสคำของหมายเลขเรียกขาน (Address Codewords)
- 3.รหัสคำของข่าวสาร (Message Codewords)
- 4.รหัสคำเทียม (Idel Codewords)

รหัสคำทั้ง 4 ชนิดนี้แต่ละชนิดประกอบด้วยเลขฐานสองจำนวน 32 บิต แต่ละชนิดมีรูปแบบ ซึ่งแสดงได้ดังตารางที่ 2.1

BIT NO	SYNCHRONIZATION 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
17	0	Address Bit	Message Bit	1

BIT NO	SYNCHROINIZATION CODEWORD	ADDRESS CODEWORD	MESSAGE CODEWORD	IDEL CODEWORD
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	Even Parity	Even Parity	1

ตารางที่ 2.1 รูปแบบของรหัสคำ

รหัสคำการซิงโครไนซ์ (Synchronization Codeword)

เป็นรหัสคำที่มีรูปแบบเฉพาะเป็นเอกลักษณ์และไม่มีการซ้ำจากรหัสคำอื่นๆ รหัสคำชนิดนี้จะอยู่เป็นอันดับแรกของแต่ละแบทช์จึงทำหน้าที่กำหนดจุดเริ่มต้นของแบทช์ด้วย โดยมีรูปแบบของบิตดังตารางที่ 2.2

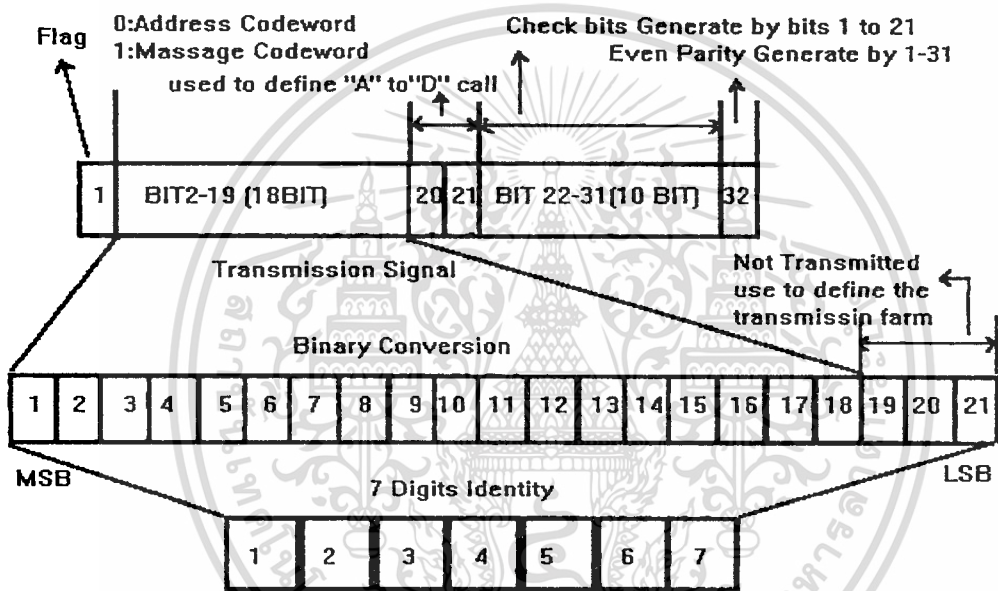
BIT NO	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1	2
BIT	0	1	1	1	1	1	0	0	1	1	0	1	0	0	1	0	0	0	0	1	0	1	0	1	1	1	0	1	1	0	0	0

ตารางที่ 2.2 รูปแบบของบิตของรหัสคำการซิงโครไนซ์

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

รหัสคำของหมายเลขเรียกขาน (Address Codewords)

รหัสคำชนิดนี้จะประกอบไปด้วยเลขฐานสองจำนวน 32 บิต ซึ่งบิตแรกจะเป็น "0" เสมอ รูปแบบของรหัสคำนี้ถูกแสดงไว้ในตารางที่ 2.1 บิตที่ 2 ถึงบิตที่ 19 เป็นรหัสหมายเลขเฉพาะที่กำหนดให้กับเพจเจอร์แต่ละเครื่อง ซึ่งถูกแปลงจากเลขฐานสิบจำนวน 7 หลักมาเป็นเลขฐานสอง (BCD) จำนวน 21 บิต ซึ่งเลขฐานสองที่ถูกแปลงมานั้นจะใช้เพียง 18 บิตที่มีความสำคัญมาก (MSB) ส่งไปกับรหัสของหมายเลขเรียกขาน ส่วนอีก 3 บิตที่มีความสำคัญน้อย (LSB) จะไม่ถูกส่งไปด้วย แต่จะถูกใช้ในการส่งรหัสของหมายเลขเรียกขานออกไปว่าเป็นเฟรมที่เท่าใดของแบทช์



รูปที่ 2.2 รูปแบบของรหัสหมายเลขเรียกขาน

รหัสคำของข่าวสาร (Message Codeword)

1. รหัสคำของข่าวสารเพจเจอร์รุ่นตัวเลข

รหัสคำของข่าวสารจะประกอบด้วยเลขฐานสองจำนวน 32 บิต โดยที่บิตแรกจะเป็น "1" เสมอ และรหัสนี้จะตามหลังรหัสของหมายเลขเรียกขาน รหัสคำของข่าวสารจะสิ้นสุดลงเมื่อมีการส่งรหัสคำของหมายเลขเรียกขานตัวต่อไปหรือมีการส่งรหัสคำเทียมหรือเมื่อการส่งสัญญาณได้มีการสิ้นสุดลง แต่ถ้าข่าวสารที่จะส่งยาวเลยแบทช์นั้นขึ้นไป ซึ่งนั่นก็หมายความว่า จะถูกแทรกด้วยรหัสคำการชิงค่อนแล้วข่าวสารส่วนที่เหลือจึงจะถูกส่งในแบทช์ถัดไปจนหมด โดยที่ไม่ต้องเริ่มด้วยรหัสคำหมายเลขเรียกขานอีกข่าวสารในวิทยุติดตามตัวรุ่นตัวเลขจะมีทั้งข่าวสารที่เป็นเลขฐานสิบ, เครื่องหมาย hyphen (-), เครื่องหมาย [, ช่องว่าง (space) และสัญลักษณ์ urgency "u" ซึ่ง

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

ข้อมูลที่แสดงออกที่หน้าจอของวิทยุติดตามตัวก็จะใช้รูปแบบนี้ รูปแบบของข่าวสารนี้จะใช้เลขฐานสองจำนวน 4 บิตแทน 1 อักขระ ชุดของอักขระสำหรับข่าวสาร ชุดอักขระสำหรับข่าวสารถูกแสดงในตารางที่ 2.3 การส่งบิตข่าวสาร b ของอักขระ แต่ละตัวจะส่งเป็นลำดับ โดยเริ่มจากบิตที่ 1 ซึ่งเป็นบิตที่มีความสำคัญน้อยที่สุดก่อนแล้วตามด้วยบิตที่ 2 , 3 , 4 ตามลำดับอักขระตัวต่อไปก็就会被ส่งตามกันไปจนครบ 5 ตัวหรือ 20 บิตต่อ 1 รหัสคำข่าวสารโดยที่ 20 บิตเหล่านี้จะเรียงจากบิตที่ 21 ของรหัสคำข่าวสาร

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
1010	-
1001	U (Urgency indicator)
1100	Space
1101	Hyphen
1110	]
1111	[

ตารางที่ 2.3 ชุดอักขระของข่าวสาร

2. รหัสคำของข่าวสารเพจเจอร์รุ่นตัวอักษร-ตัวเลข (Alpha-Numeric)

สำหรับรหัสคำข่าวสารชนิดนี้จะมีลักษณะ โครงสร้างทางบิตและรูปแบบเหมือนกันกับรหัสคำข่าวสารของรุ่นตัวเลข แต่จะมีความแตกต่างกันตรงจำนวนบิตเนื่องจากเป็นรุ่นตัวอักษร-ตัวเลข จึงจำเป็นต้องใช้จำนวนบิตเพิ่มมากขึ้น ตามมาตรฐานของ POSSAG 1 ตัวอักษร หรือ 1 อักขระ จะใช้จำนวนบิต 7 บิต ด้วยกัน ซึ่งจะต่างกับจำนวนบิตของรุ่นตัวเลขที่มีเพียง 4 บิตเท่านั้น และรุ่นตัวอักษร-ตัวเลขนี้ รหัสข่าวสารที่มีจะมีถึง 80 รูปแบบ โดยมีตัวเลข 0-9 อักษรภาษาอังกฤษพิมพ์ใหญ่และพิมพ์เล็ก รวมทั้งสัญลักษณ์พิเศษต่างๆ ลักษณะตำแหน่งของข่าวสารในบิตข่าวสารนี้จะมีการเลื่อนไปของข้อมูลตามความยาวของข่าวสารที่ส่งมา หมายความว่าในบิตข่าวสารของรหัสคำข่าวสารแรกนั้นจะบรรจุได้ 3 ตัวอักษร แต่อักขระตัวที่ 3 จะมีเพียง 6 บิต ส่วนบิตสุดท้ายของอักขระตัวที่ 3 จะไปอยู่ในบิตข่าวสารของรหัสคำข่าวสารถัดไป ถ้ามีการส่งเป็นข้อความยาวๆ จำนวนบิตของอักขระแต่ละตัวจะถูกเลื่อนไป เนื่องจากรูปแบบของบิตข่าวสารกำหนดมีความยาว 20 บิต หรือ 2 ตัวกับอีก 6 บิต หมายความว่าจะมีอักขระบางตัวที่อยู่คร่อมระหว่างบิตข่าวสารของรหัสคำข่าวสาร 2 รหัสคำ ส่วนรหัสของอักขระแต่ละตัวอักษรแสดงไว้ในตารางที่ 2.4

7-bits Combination Bit No. 7654321	Displayed Character
0000110	0
1000110	1
0100110	2
1100110	3
0010110	4
1010110	5
0110110	6
1110110	7
0000110	8
1001110	9
0101110	:
1101110	;

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้าไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

7-bits Combination Bit No. 7654321	Displayed Character
0011110	<
1011110	=
0111110	>
1111110	?
0000001	@
1000001	A
0100001	B
1100001	C
0010001	D
1010001	E
0110001	F
1110001	G
0000001	H
1001001	I
0101001	J
1101001	K
0011001	L
1011001	M
0111001	N
1111001	O
0000101	P
1000101	Q
0100101	R
1100101	S
0010101	T
1010101	U
0110101	V

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า
ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

7-bits Combination Bit No. 7654321	Displayed Character
1110101	W
0000101	X
1001101	Y
0101101	Z
1101101	[
0011101	
1011101	]
0111101	^
1111101	-
0000011	“
1000011	a
0100011	b
1100011	c
0010011	d
1010011	e
0110011	f
1110011	g
0000011	h
1001011	i
0101011	j
1101011	k
0011011	l
1011011	m
0111011	n
1111011	o
0000111	p
1000111	q

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า
ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

7-bits Combination Bit No. 7654321	Displayed Character
0100111	r
1100111	s
0010111	t
1010111	u
0110111	v
1110111	w
0000111	x
1001111	y
0101111	z
1101111	{
0011111	
1011111	}
0111111	~
1111111	#

ตารางที่ 2.4 ชุดอักขระของข่าวสารเพจเจอร์รุ่นตัวอักษร-ตัวเลข

รหัสคำเทียม (Idle Codeword)

ถ้าข่าวสารที่ส่งมาหมดลงก่อนการสิ้นสุดของเฟรม รหัสคำเทียมนี้จะถูกใส่แทนรหัสคำปกติเพื่อให้เฟรมนั้นมีครบสองรหัสคำ (64 บิต) 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
Bit	0	1	1	1	1	0	1	0	1	0	0	0	1	0	0	1	1	1	0	0	0	0

ตารางที่ 2.5 รูปแบบของรหัสคำเทียม

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

หลักการทั่วไปของภาครับเพจเจอร์

การรับส่งข้อมูลข่าวสารของเพจเจอร์ที่ใช้กันอยู่ในปัจจุบัน ใช้การมอดูเลทแบบ FSK (Frequency Shift Keying Modulation) ซึ่งมีย่านความถี่ต่างที่ใช้ดังนี้

- 1.VHF LOW BAND (33-37,40-50 MHz)
- 2.VHF HIGH BAND (138-174 MHz)
- 3.UFH LOW BAND (406-422,435-512 MHz)
- 4.UFH HIGH BAND (920-932 Mhz)

วงจรภาครับสัญญาณ (Receiver Circuit)

สำหรับภาครับของสัญญาณเพจเจอร์ที่ใช้ทดสอบมีด้วยกัน 2 ชนิด

1.Field Strength meter

ประกอบด้วยส่วนต่างๆซึ่งคล้ายคลึงกับเครื่องรับวิทยุ FM ทั่วไปคือมีภาค amplifier mixer , bandpass filter , IF amplifier และ demodulator ข้อแตกต่างของเครื่องรับแบบ FM และ FSK ก็คือเครื่องรับ FM จะมีการดีมอดูเลทที่ค่อนข้างเป็นเชิงเส้น (linearity) มากกว่าแบบ FSK ทั้งนี้เนื่องจาก FSK มีความถี่ของการส่งเพียงสองความถี่สำหรับระดับของแรงดันสองระดับ ("0"และ"1") เท่านั้นถ้าพิจารณาจากรูปที่ 3.2 ความถี่ทั้งสองก็คือ f_1 และ f_2 การที่จะทำให้เครื่องรับมีคุณลักษณะดังกล่าวจะต้องใช้วงจรกรองความถี่ที่เหมาะสม ซึ่งในเครื่องรับเพจเจอร์จริงๆจะใช้วงจร กรองความถี่แบบผลึกสองขั้ว (Two-Pole Crystal Filter) ที่มีค่า Q สูงมากๆ และมีแบนด์วิดท์แคบมากๆโดยมีความถี่กลาง (center frequency) หรือ IF ตรงกับ f_0 ในรูปที่ 3.2 นั้นเอง วงจรกรองความถี่ชนิดนี้มีแบนด์วิดท์แคบประมาณ 12 KHz เท่านั้นและประกอบด้วยขดสองขดหรือสองขั้วดังนี้

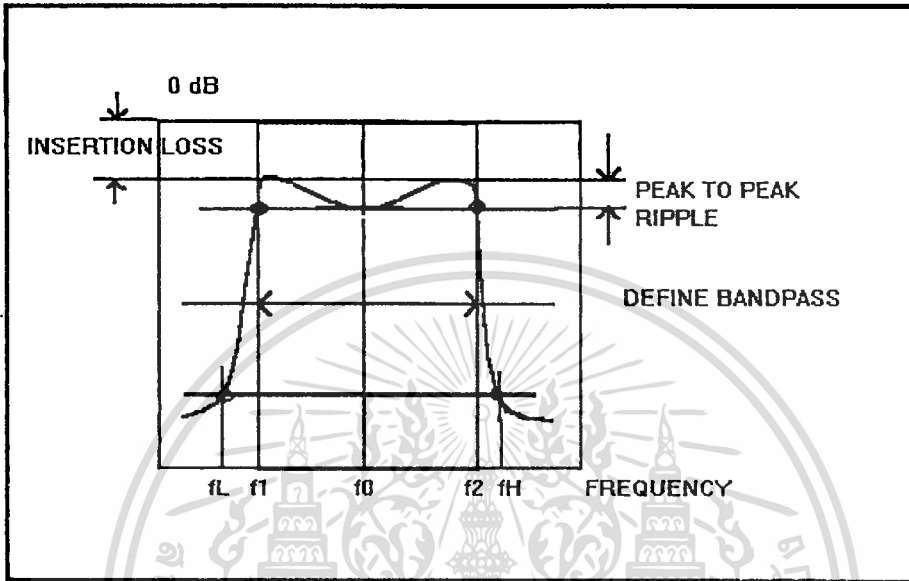
$$f_1 = f_0 - 6 \text{ KHz}$$

และ
$$f_2 = f_0 + 6 \text{ KHz}$$

สัญญาณ IF จะผ่านวงจรกรองและวงจรขยายหลายชั้นที่เดียวเพื่อเพิ่มความคมของการกรองความถี่ให้ดีขึ้นและแคบขึ้นสาเหตุที่จะต้องกรองความถี่อื่นๆทั้งก็เพื่อเป็นการลดสัญญาณรบกวนซึ่งแฝงมากับความถี่ต่างๆออกไปและช่วยให้สัญญาณข้อมูล (baseband signal) ที่ถูกดีมอดูเลทออกมามีความถูกต้องมากขึ้น

อีกภาคหนึ่งที่มีความสำคัญมากก็คือภาคดีมอดูเลเตอร์ภาคนี้จะต้องมีช่วงการตอบสนองความถี่ที่เหมาะสมไม่แคบจนเกินไป เพราะสัญญาณสแควร์เวฟที่ผ่านการดีมอดูเลทไม่ได้มีเพียง

ความถี่แรกเท่านั้นแต่จะต้องประกอบด้วยความถี่อื่นๆอีกบางช่วงเพื่อให้ได้สัญญาณสแควร์เวฟที่สมบูรณ์มากขึ้น ดังนั้นคิโมดุลเลเตอร์จึงต้องมีคุณสมบัติที่ต่างจาก FM คิโมดุลเลเตอร์ทั่วไป



รูปที่ 3.2 การตอบสนองของ Two-Pole Crystal Bandpass Filter

เนื่องจากไม่ได้มีการสร้างเครื่องรับที่สร้างขึ้นมา จึงใช้เครื่องวัดความแรงของสัญญาณ Field Strength meter แทนในส่วนของการรับและได้ตั้งค่าต่างๆของเครื่องเพื่อให้ได้ใกล้เคียงกับคุณสมบัติของเครื่องรับ FSK ที่ใช้งานได้จริงโดยจะตั้งค่า BW (Band Width) ไว้ที่ 15 KHz และตั้งค่าสวิตช์ FM/AM ไว้ที่ FM การใช้เครื่องวัดความแรงของสัญญาณแทน Rx ข้อดีก็คือสามารถเลือกย่านความถี่ได้และยังสามารถทดสอบการรับข้อมูลจริงที่ส่งจากบริษัทต่างๆที่ทำการส่งเพจเจอร์ได้อีกด้วย

2.FSK Demodulator (Single tune)

วงจรการรับสัญญาณเป็นวงจรที่อยู่ในภาคแรกๆของเครื่องรับสัญญาณ Pager ซึ่งทำหน้าที่รับสัญญาณในช่องความถี่ของการรับและส่งสัญญาณ Pager โดยใช้การมอดูเลตแบบ FSK (Frequency Shift Keying) ในปัจจุบันการรับส่งข้อมูลข่าวสารของ Pager ส่วนใหญ่จะอยู่ในย่านความถี่ต่อไปนี้

1.VHF LOW BAND (33-37,40-50 MHz)

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้าไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

2.UFH LOW BAND (406-422,435-512 MHz)

คุณสมบัติในขั้นแรกของวงจรภาครับสัญญาณที่สมควรจะมีได้แก่

- 1.มีความไวในการรับสัญญาณสูง
- 2.ใช้แรงดันไฟในการทำงานต่ำ
- 3.มีการตัดเทคจุดผ่านศูนย์ของสัญญาณ (Zero Crossing Dertection) สำหรับใช้กับการรับส่งด้วยระบบ FM FSK ซึ่งมีอัตราเร็วข้อมูลสูง
- 4.วงจรควรมีขนาดเล็กเพื่อเป็นการประหยัดพื้นที่ในการต่อวงจรรวมกับภาคอื่นบนแผ่นวงจรพิมพ์เดียวกัน

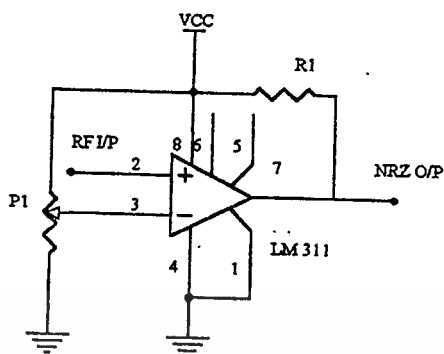
เมื่อภาครับสัญญาณทำการรับสัญญาณที่ส่งมาในช่องความถี่ของ Pager ได้แล้วก็จะทำการตัดเทคเพื่อให้ได้เป็นสัญญาณ Digital Baseband และจะส่งสัญญาณนี้ไปทำการกู้สัญญาณนาฬิกา และจัดเฟรมเพื่อถอดรหัสคำหมายเลขเรียกขานต่อไป อุปกรณ์ที่เป็นสิ่งสำคัญของภาครับสัญญาณนี้ก็คือ IC เบอร์ MC 3362 ของ MOTOROLA ซึ่งเป็น IC ภาครับ FM ระบบ Dual Conversion คือ ภาค IF มีการแปลงความถี่ 2 ครั้ง จากความถี่ที่รับเข้ามาได้จะแปลงลงมาเป็น 10.7 MHz ก่อนแล้วค่อยลดลงเป็น 455 MHz อีกครั้งหนึ่งตามลำดับ ซึ่งเป็นวิธีที่ใช้กันโดยทั่วไปกับวิทยุรับ/ส่งในปัจจุบัน

วงจรลิมิตเตอร์ (Limiter)

ธรรมชาติของสัญญาณสแควร์เวฟจะประกอบด้วยฮาร์โมนิกแรก (1^{st} harmonic) และฮาร์โมนิกอื่นๆ อีกมากมายซึ่งเป็นฮาร์โมนิกที่ สามารถแสดงได้ในรูปของอนุกรมฟูเรียร์ (Fourier serie) ของสแควร์เวฟที่มี Duty cycle 50% ได้ดังนี้

$$V(t) = A[1/2 + \sin at + 1/3 \sin 3 nt + 1/5 \sin 5nt + \dots]$$

แต่เครื่องรับโดยเฉพาะภาคขยายสัญญาณ, วงจรกรองต่างๆ และดีมอดูเลเตอร์จะลดทอนฮาร์โมนิกที่สูงๆ ออกไปอย่างมีอาจจะหลีกเลี่ยงได้ ดังนั้นสัญญาณ NRZ ที่ได้จากเครื่องรับจะมีรูปร่างผิดไปจากความเป็นจริง จะทำให้การตีความของ TTL ผิดพลาดเนื่องจากไม่รู้ว่าเป็นระดับสัญญาณหรือสูงต่ำ (High or Low logic) สิ่งนี้สามารถแก้ไขได้โดยให้สัญญาณที่ออกจากเครื่องรับผ่านวงจรลิมิตเตอร์ก่อน โดยวงจรลิมิตเตอร์จะทำหน้าที่เปลี่ยนระดับสัญญาณที่ได้ให้เป็นสัญญาณ NRZ ที่มี Duty cycle 50% ซึ่งวงจรลิมิตเตอร์สร้างจากออปแอมป์ที่ถูกต่อแบบคอมพาราเตอร์ (Comparater) โดยอัตราขยายสูงมากๆ ไม่มีการป้อนกลับสัญญาณถูกต่อ เข้าขานอนอินเวอร์ตติง เพื่อเปรียบเทียบกับแรงดันอ้างอิง (Reference voltage) ที่ขาอินเวอร์ตติงดังรูปที่ 3.3



รูปที่ 3.3 วงจรลิมิตเตอร์

ออปแอมป์ที่มีใช้ในวงจรนี้คือ LM 311 เนื่องจากเป็นออปแอมป์ที่ใช้แหล่งจ่ายไฟเดียว ซึ่งเหมาะสมกับวงจรโดยรวมที่มีเฉพาะแรงดัน TTL เท่านั้น จึงไม่ต้องการแหล่งจ่ายไฟลบเพิ่มเติม

วงจรกู้สัญญาณนาฬิกา (Clock Recovery)

เนื่องจากการส่งข้อมูลของเพจเจอร์เป็นการส่งแบบอะซิงโครนัส จึงไม่มีสัญญาณนาฬิกาควบคุมด้วย ทางด้านรับจึงต้องสร้างสัญญาณนาฬิกาขึ้นมาใหม่จากสัญญาณ NRZ ที่ได้รับเพื่อใช้ควบคุมการทำงานของส่วนต่างๆ ในภาครับให้ทำงานพร้อมเพรียงกัน (Synchronous) สัญญาณนาฬิกาที่สร้างขึ้นใหม่จะต้องมีความถี่ตรงกับบิตเรต (Bit rate) ของ NRZ และยังต้องซิงโครนัส (Synchronous) กับสัญญาณ NRZ ด้วยรูปที่ 3.4 เป็นวงจรกู้สัญญาณนาฬิกาเริ่มจากออสซิลเลเตอร์ (Oscillator) อ้างอิงโดยใช้คริสตอล (X-tal) 18.432 MHz ต่อรวมกับตัวเก็บประจุและ 74HC4060 ทำหน้าที่ป้อนไฟให้คริสตอล รวมทั้งหารความถี่จากคริสตอลด้วย 64 ไปในตัวสัญญาณนาฬิกาจาก 4060 จะถูกหารด้วย 15 ด้วย 74LS161 โดยใช้เทคนิคการตั้งค่าหารเริ่มต้น (Present) ที่ 1 แล้วดึงสัญญาณจากขา ripple ผ่านนอตเกต เพื่อโหลดตัวมันเองดังนั้น 74LS161 นี้จะเริ่มนับ 1 ถึง 15 แล้ววนกลับมาเริ่มที่ 1 ใหม่ หลังจากนั้นสัญญาณนาฬิกาจากขา ripple จะถูก 74LS161 หารด้วย 16 เพื่อให้ได้สัญญาณนาฬิกาความถี่ 1200 บิตต่อวินาที

สัญญาณ NRZ ที่ผ่านลิมิตเตอร์มาแล้วถูกใช้เป็นตัวควบคุมให้สัญญาณนาฬิกาซิงโครนัสกับตัวเองโดยนำ NRZ มาหน่วงเวลาด้วยนอตเกต 3 ตัว เพื่อให้ขอบขาขึ้นและขอบขาลงของ NRZ หน่วงเวลาไว้เหมือนกับ NRZ เดิม แล้วเทียบสัญญาณทั้งสองด้วยเอ็กซ์คลูซิฟออร์ (XOR) จึงทำให้เกิดพัลส์ “0” เล็กๆ ทุกๆ การเปลี่ยนแปลงของสัญญาณ NRZ แต่ละครั้ง และจะเป็น “1”

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

ในช่วงเวลาอื่นพัลส์ที่ได้นี้สามารถใช้เคลีย 74LS161 เพื่อให้สัญญาณนาฬิกาที่ได้ซิงค์กับ NRZ ตลอดเวลาพัลส์ที่ได้นี้สามารถใช้เคลียร์ ดังรูปที่ 3.4

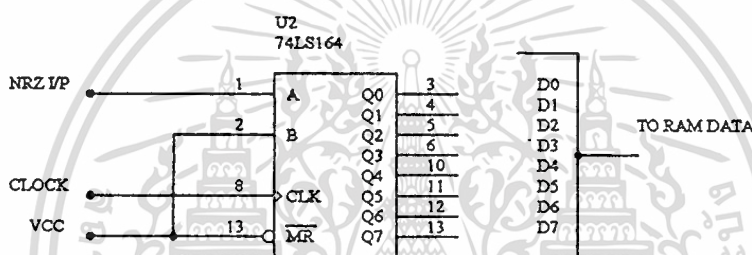
วงจรตรวจจับเฟรมซิงค์ (Frame Sync Detector)

สัญญาณเฟรมซิงค์นับว่ามีความสำคัญในการส่งข้อมูลพธเจรอย่างมกเพราะจะเป็นตัวบอกให้ทราบจุดเริ่มต้นของการส่งข้อมูลในแต่ละเบทซ์และเครื่องรับยังอาศัยสัญญาณนี้สำหรับใช้ในการตรวจสอบระยะเวลาการส่งว่าสิ้นสุดลงเมื่อใดอีกด้วย สัญญาณเฟรมซิงค์จะถูกสร้างจากรหัสค่าการซิงค์ทั้ง 32 บิตซึ่งเป็นเอกลักษณ์เฉพาะตัวอยู่แล้ว วิธีการตรวจจับสัญญาณเฟรมซิงค์จะเริ่มจากการนำสัญญาณ NRZ ผ่านดีฟลิปฟลอป (D-Flipflop) เพื่อหน่วงสัญญาณไปหนึ่งบิต สาเหตุที่ต้องการหน่วงก็เพราะต้องการให้สัญญาณ NRZ ถูกเลื่อน (Shift) จากเดิม 32 บิต ไป 33 บิต (1 บิตดีฟลิปฟลอปและ 32 บิตที่ชิพรีจิสเตอร์แบบ 8 บิต 4 ตัว) พร้อมกับที่บิตแรกซึ่งถัดจากรหัสค่าซิงค์เริ่มต้นนับแอดเดรส (Address)

จากตารางที่ 2.1 จะเริ่มวาร์หัสค่าการซิงค์ประกอบด้วย “0” 16 ตัว “1” 16 ตัว สัญญาณเฟรมซิงค์ถูกกำหนดให้เกิดขึ้นที่บิตสุดท้ายของรหัสค่าการซิงค์ ดังนั้นจึงต้องต่อขาเอาต์พุตของ 74LS164 ที่คาดว่าจะเป้น “0” ในช่วงที่เกิดเฟรมซิงค์พอดีนนำไปผ่านน้อทเกตแล้วนำไปเข้าแนทเกต 74LS30 ส่วนขาเอาต์พุตที่เหลือของ 74LS164 ถูกต่อเข้าแนทเกตโดยตรงดังรูปที่ 3.5 หลังจากนั้นสัญญาณจากขา 8 ของแนตเกตทั้ง 4 จะถูกออร์ (OR) กันจนได้สัญญาณเฟรมซิงค์ออกมาทุกๆบิตสุดท้ายของรหัสค่าการซิงค์ ดังแสดงในรูปที่ 3.5

SIPO (Serice In parallal Out)

เนื่องจากข้อมูลที่ได้รับได้จากสัญญาณพเจเจอร์เป็นข้อมูลแบบอนุกรม จึงจำเป็นต้องแปลงข้อมูลเหล่านั้นให้เป็นแบบขนาด 8 บิต เพื่อให้สามารถเขียนลงแรมได้ หลักการของ SIPO ก็คือการเลื่อนบิตของข้อมูลไปเรื่อยๆ และทุกๆ 8 บิต จะถูกอ่านจากแรม 1 ครั้ง ซึ่งวงจรที่ใช้ก็คือ 74LS164 (8 bit shift register) และใช้สัญญาณนาฬิกาเป็นตัวกำหนดการเลื่อนบิตของข้อมูลทีอื่น พุทขา 1 ของ 74LS164 จะเลื่อนไปที่ Q_0 (ขา 3) เมื่อสัญญาณนาฬิกาเป็นขอบขาขึ้นและข้อมูลจะถูกเลื่อนจาก Q_0 ไป Q_1 Q_2 Q_3 Q_4 Q_5 Q_6 Q_7 แล้วจึงมีการอ่านข้อมูลจากแรม 1 ครั้ง ดังนั้นสัญญาณจาก Q_0 - Q_7 ดังรูปที่ 3.6



รูปที่ 3.6 วงจร PISO

การกำเนิดสัญญาณอ่าน/เขียนแรม และสัญญาณนับแอดเดรส

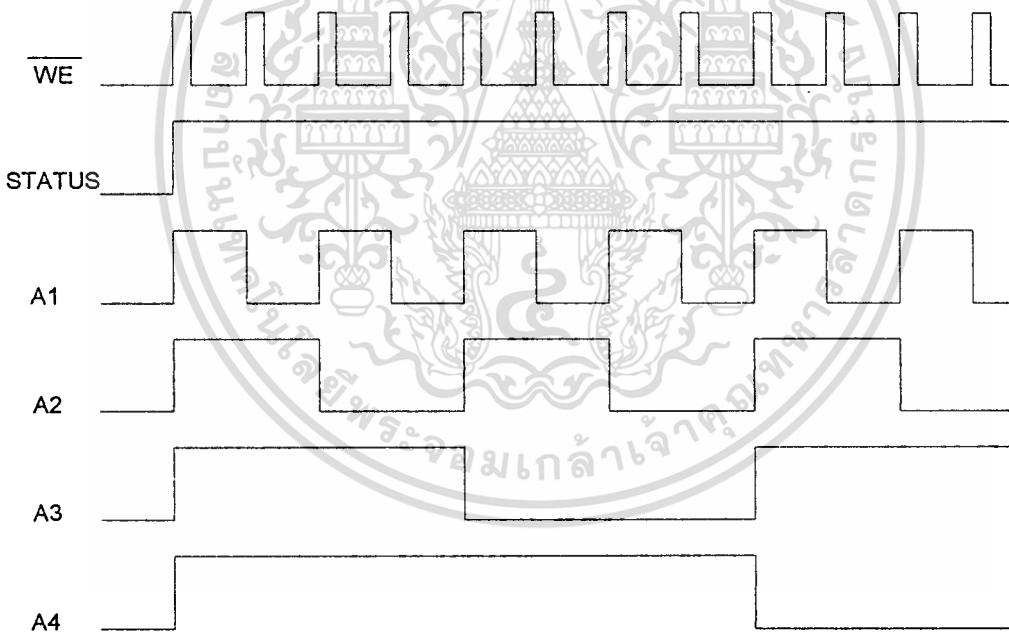
(Read/Write Clock Generator and Address Counter)

ในช่วงแรกของภาคนี้นี้จะกำเนิดสัญญาณการอ่านและเขียนข้อมูลให้กับขา WE (ขา 27) ของแรม 6264 ซึ่งจะเป็นพัลส์เล็กๆ ซึ่งมีความกว้างเท่ากับพัลส์ของสัญญาณนาฬิกา โดยในขั้นแรกสัญญาณนาฬิกาที่มีบิตเรท 1200 บิตต่อวินาที จะถูกหารด้วย 2,4,8,16 ดังในรูปที่ 3.8 แล้วจึงนำเอาที่พุทจากขา Q_0, Q_1, Q_2 สัญญาณนาฬิกาและสัญญาณ STATUS มาทำการแนคกันหมด และจะได้พัลส์ “0” แคบๆ ทุกๆ 8 บิตของสัญญาณนาฬิกาหรือสัญญาณ NRZ นั้นเอง ดังนั้นสัญญาณการเขียนนี้จะเกิดขึ้นเพื่อให้แรมถูกเขียนข้อมูลทุกๆ 8 บิตหรือ 1 ไบท์ของพเจเจอร์อีก ส่วนหนึ่งของภาคนี้นี้เป็นตัวนับแอดเดรสสำหรับแรม 6264 ซึ่งมีแอดเดรสทั้งหมด 13 เส้น (A_0 - A_{12}) แต่เนื่องจากวงจรของแรมนี้ติดต่อกับคอมพิวเตอร์ผ่านทางบัสบนเมนบอร์ด (bus slot on main board) โดยใช้บัสข้อมูล (data bus) เพียง 8 บิต (D_0 - D_7) ขณะที่คอมพิวเตอร์ที่ใช้ในการทดลองเป็นแบบ 16 บิต (CPU 80286) คือมีบัสข้อมูล 16 เส้น (D_0 - D_{15}) จึงทำให้ CPU ไม่สามารถอ่านข้อมูลจาก D_8 - D_{15} ได้ วิธีการแก้ไขก็คือการใช้แอดเดรสคู่ของแรมในการ

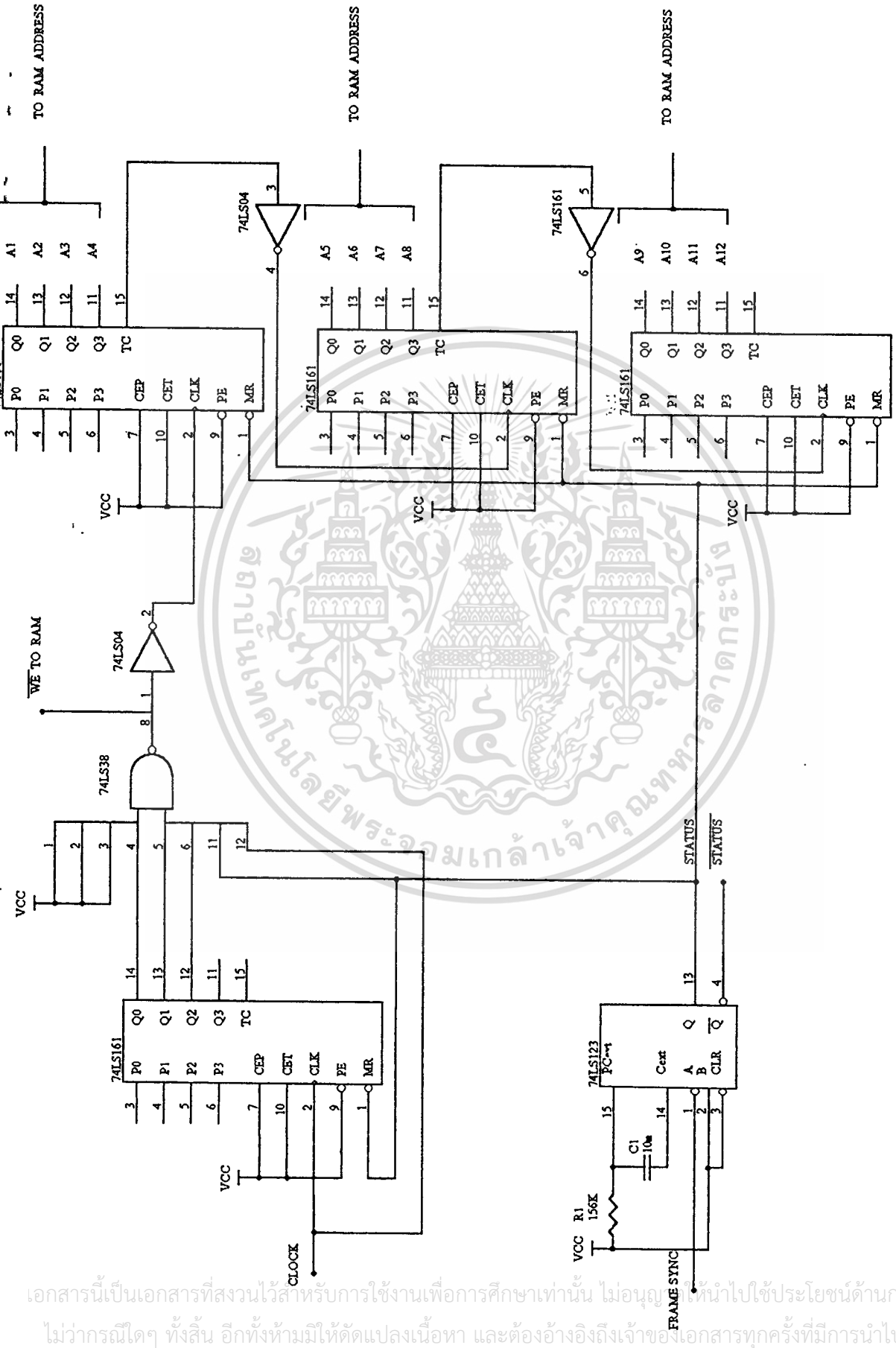
เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า

ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

บันทึกและอ่านข้อมูลเพจเจอร์เท่านั้น ซึ่งหมายความว่า CPU จะต้องอ่านข้อมูลจาก $D_0 - D_7$ (ไบต์ล่าง) ของบัสข้อมูลโดย ไม่สนใจข้อมูลจาก $D_0 - D_{15}$ ในทางปฏิบัติจึงได้ทำการต่อขา A_0 (ขา 10) ของ 6264 ลงกราวด์และเหลือเฉพาะ $A_1 - A_{12}$ ที่จะเป็นตัวกำหนดแอดเดรส วงจรตัวนับแอดเดรส (Address Counter) จึงมีเอาท์พุทเหลือเพียง 12 เส้น สำหรับ $A_1 - A_{12}$ ของแรม การนับแอดเดรสของแรมจะเป็นคู่เท่านั้น (0,2,4,8,...) ในรูปที่ 3.8 แสดงถึงการสร้างแอดเดรสโดยใช้วงจรหารทำการหารพลัส อ่าน/เขียน ที่ออกจาก 74LS161 โดยถูกกลับเฟสก่อนเพื่อให้ A_1 เปลี่ยนแปลงพร้อมกับที่มีการเขียนข้อมูลลงแรมดังรูปที่ 3.7 จะสังเกตเห็นว่าขา 1 ของ 74LS161 ทุกตัวจะถูกต่อกับสัญญาณ STATUS เพื่อเริ่มการทำงาน เฉพาะตอนที่มีการส่งสัญญาณเพจเจอร์มาเท่านั้นและมีการนับ แอดเดรสทุกเส้นเริ่มต้นพร้อมๆกัน ดังนั้น 74LS161 ตัวบนจะกำเนิดสัญญาณสำหรับแอดเดรส $A_1 - A_4$, 74LS161 ตัวกลางสำหรับ $A_5 - A_6$ และ 74LS161 ตัวล่างสำหรับ $A_9 - A_{12}$ ดังรูปที่ 3.8



รูปที่ 3.7 Timing Diagram ของตัวนับแอดเดรส



รูปที่ 3.8 วงจรกำเนิดสัญญาณอ่าน,เขียนแรมและตัวนับแอดเดรส

แรมอีมูเลเตอร์ (RAM Emulator)

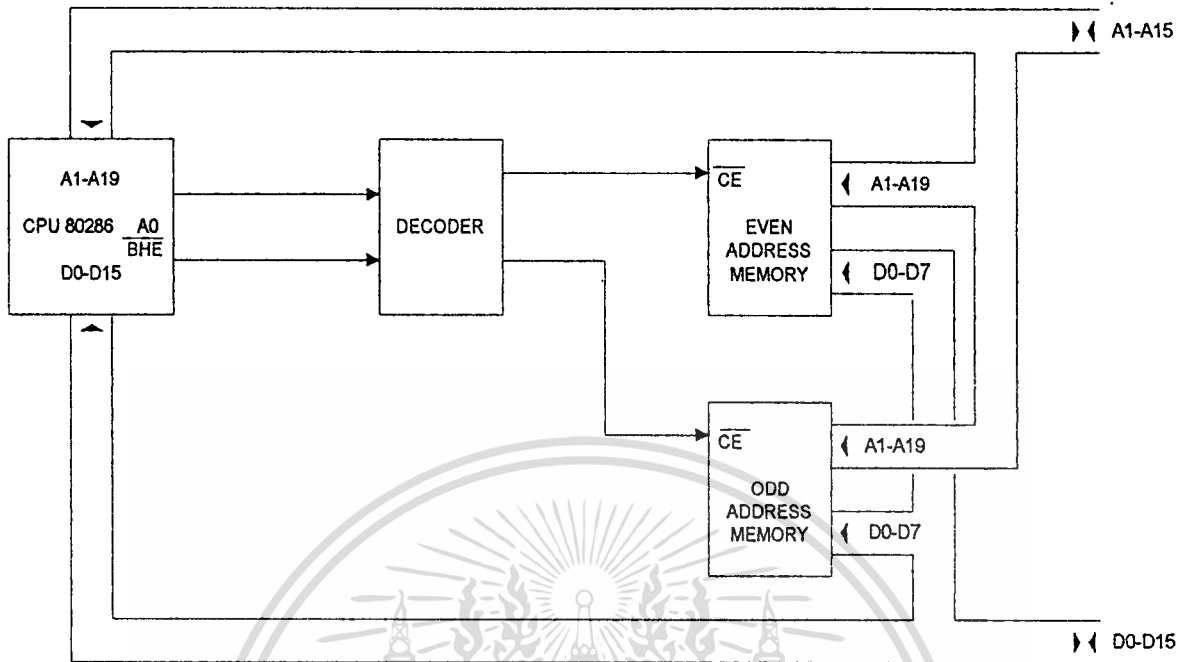
แรมในส่วนนี้ทำหน้าที่เป็นที่พักข้อมูลเพจเจอร์ เพื่อให้คอมพิวเตอร์สะดวกในการอ่านข้อมูล โดยคอมพิวเตอร์จะมองว่า แรมอีมูเลเตอร์เป็นหน่วยความจำส่วนหนึ่งของเครื่องและจะอ่านข้อมูลผ่านทางบัสข้อมูลของสล็อตบนเมนบอร์ด

-สแตติกแรม

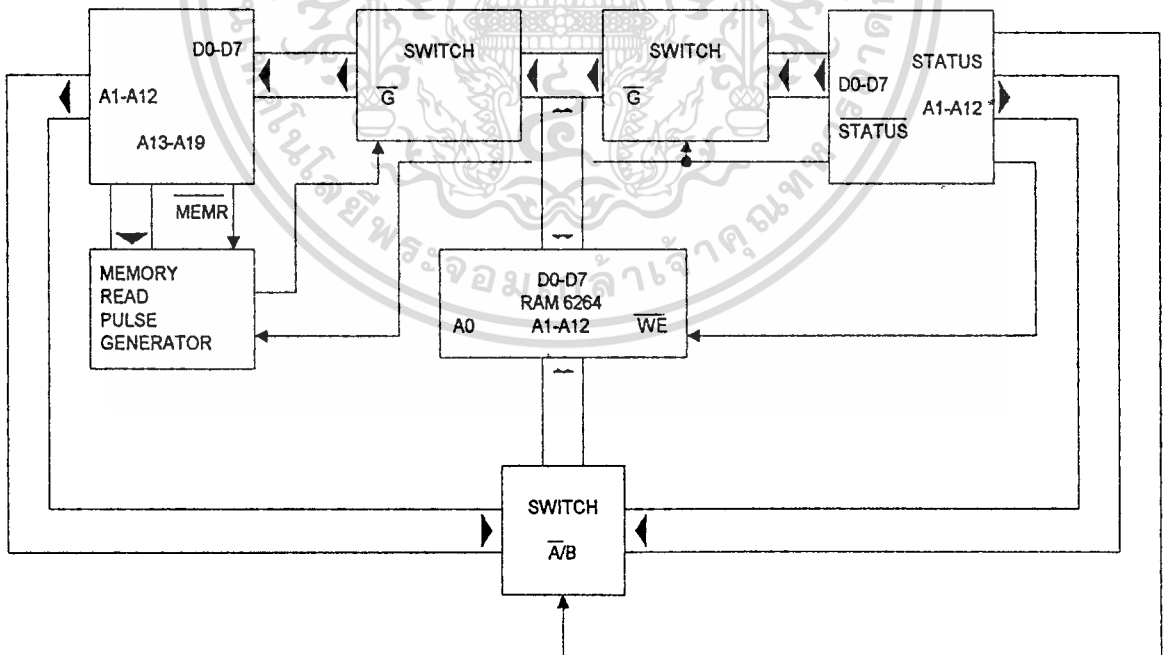
ใช้แรม 9294 ที่มีความจุ 8 กิโลไบต์แต่คอมพิวเตอร์จะอ่านเฉพาะข้อมูลในแอดเดรสคู่เท่านั้นจึงต้องต่อ A_0 (ขา 10) ลงกรานด์ตลอดทำให้เหลือความจุเพียงครึ่งหนึ่ง (4 กิโลไบต์) เท่านั้น แต่ก็เพียงพอสำหรับการบันทึกข้อมูลเพจเจอร์ในการส่งแต่ละครั้ง ขา WE (ขา 27) ได้รับสัญญาณการเขียนทุกๆ 8 บิต ของข้อมูลโดยใช้พัลส์ “0” สั่งให้แรมบันทึกข้อมูลส่วนช่วงเวลาอื่นสัญญาณที่ขา WE นี้เป็น “1” ตลอดที่ขา A_1-A_{12} จะถูกต่อกับแอดเดรสที่สร้างจากสัญญาณนาฬิกาในภาคตัวนับแอดเดรส แต่ในช่วงจังหวะที่คอมพิวเตอร์ทำการอ่านข้อมูลจากแรม 6264 ขาแอดเดรส A_1-A_{12} ของบัสแอดเดรสบนเมนบอร์ดตัวเลือกแอดเดรส (Address Selector) จะใช้ 74LS157 เป็นตัวเลือกที่ถูกควบคุมด้วยสัญญาณ STATUS ที่ขา A/B (ขา 1) เมื่อมีสัญญาณเพจเจอร์ส่งเข้ามา STATUS จะเป็น “1” ทำให้ 74LS157 ต่อ A_1-A_{12} ของแรมกับขาแอดเดรส A_1-A_0 ของคอมพิวเตอร์แทนและจึงทำให้แรมพร้อมที่จะถูกอ่านข้อมูลโดยคอมพิวเตอร์ ตัวตัดต่อบัสข้อมูลใช้ 74LS244 จำนวน 2 ตัว โดยตัวแรก ทำหน้าที่ต่อบัสข้อมูลจากภาค SIPO เข้ากับแรมซึ่งจะต่อบัสเฉพาะตอนที่มีการส่งข้อมูลเพจเจอร์เท่านั้น สัญญาณที่ใช้ควบคุมคือ STATUS ซึ่งต่อกับ 1G และ 2G (ขา 1 และขา 19) ของ 74LS244 จะเสมือนเปิดวงจร (high impedance) ส่วน 244 อีกตัวหนึ่งทำหน้าที่ตัดต่อบัสข้อมูลของแรมกับคอมพิวเตอร์ โดยอาศัย สัญญาณควบคุมจากการถอดรหัส $A_{13}-A_{19}$, MENU และ STATUS มาแนบกันสาเหตุที่นำ $A_{13}-A_{19}$ มาต่อไว้ก็เพื่อเป็นการถอดรหัสให้ 74LS244 ต่อบัสข้อมูลของแรม เข้ากับของคอมพิวเตอร์เฉพาะตอนที่คอมพิวเตอร์ทำการอ่านหน่วยความจำส่วนนี้เท่านั้น โดยที่แรมอีมูเลเตอร์ถูกตั้งให้มีแอดเดรสดังตารางที่ 3.1

Address	A_{19}	A_{18}	A_{17}	A_{16}	A_{15}	A_{14}	A_{13}	A_{12}	A_{11}		A_0
Logic	1	1	0	1	1	1	1	X	X		X

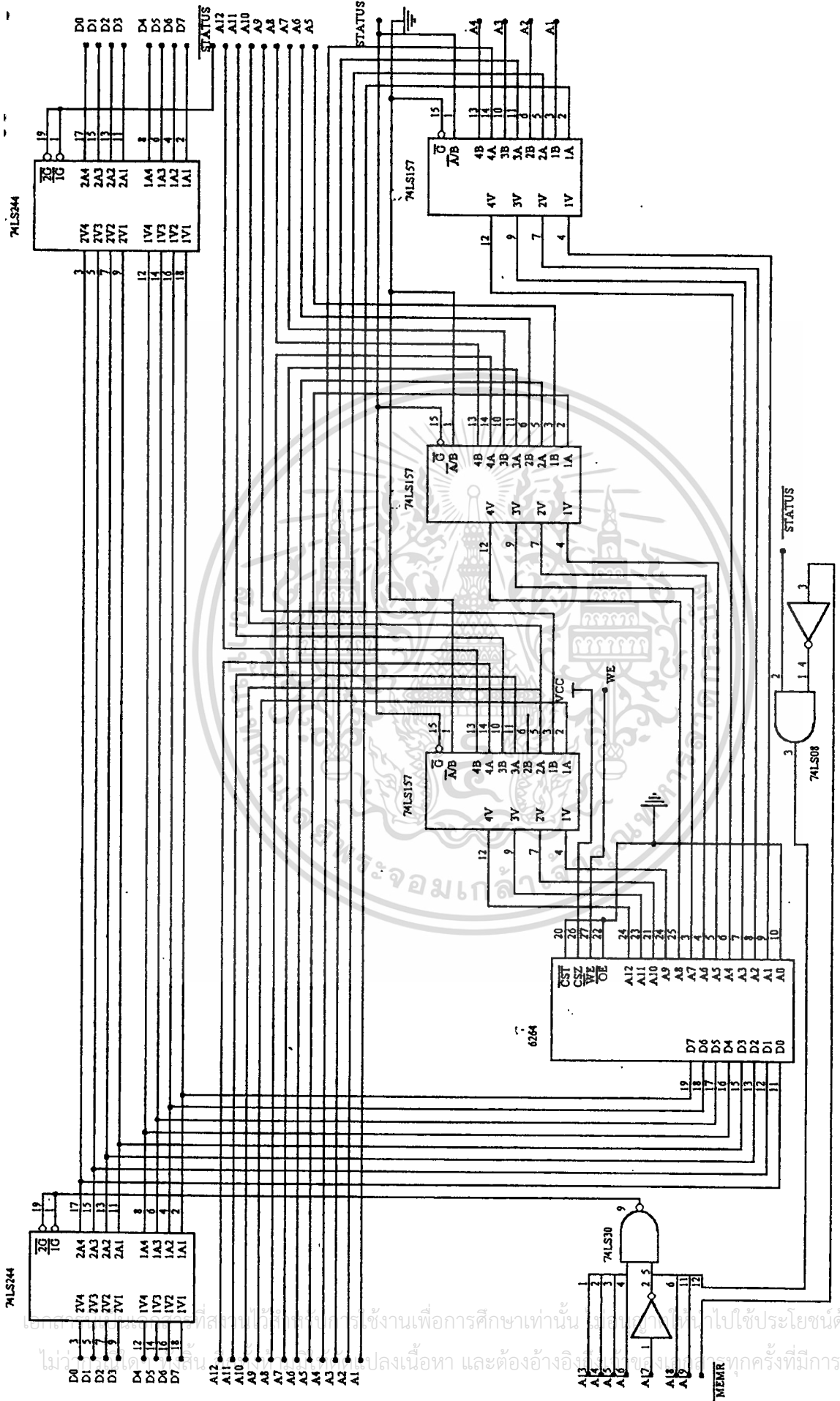
ตารางที่ 3.1 บัสแอดเดรสของแรมอีมูเลเตอร์



รูปที่ 3.9 การแบ่งหน่วยความจำของ CPU 80286



รูปที่ 3.10 บล็อกโคอะแกรมของแรม



รูปที่ 3.11 ภาครวมอีเลเตอร์

กรณีบัสแอดเดรสเส้นใดเป็น “0” ต้องผ่าน not gate ก่อน จึงจะเข้าวงจรถอดรหัสได้ ส่วนสัญญาณ MEMR จะต้องมีไว้เพื่อให้ 74LS244 ตัวแรกต่อบัสข้อมูลเฉพาะตอนที่คอมพิวเตอร์ต้องการอ่านมิฉะนั้นข้อมูลจะไหลเข้าบัสข้อมูลของเมนบอร์ดคอมพิวเตอร์ในช่วงที่ไม่ต้องการอ่านและอาจจะทำให้คอมพิวเตอร์หยุดทำงาน (hank) ได้ เนื่องจากบัสข้อมูลถูกใช้ร่วมกันทั้งจากแรมอิมูเลเตอร์และจากส่วนอื่นๆ ของคอมพิวเตอร์ด้วยสัญญาณอีกเส้นที่จำเป็นด้วยคือ STATUS เพื่อที่จะทำให้คอมพิวเตอร์อ่านข้อมูลเฉพาะตอนที่ไม่มี การส่งข้อมูลเพจเจอร์มา ดังนั้นข้อมูลในบัสข้อมูลของแรมจะมีเพียงอย่างเดียวอย่างหนึ่งในเวลาเดียวกันเท่านั้น

ภาคตรวจจับหมายเลขเรียกขาน (Address Codeword Detector)

สัญญาณหมายเลขเรียกขานเกิดจากรหัสหมายเลขเฉพาะที่กำหนดให้เพจเจอร์แต่ละเครื่อง ซึ่งมีความสำคัญอย่างมาก เพราะเพจเจอร์แต่ละเครื่องสามารถที่จะรับข่าวสารจากผู้ส่งได้อย่างถูกต้องและตรงได้จากการตรวจสอบสัญญาณหมายเลขเรียกขานและยังเป็นสัญญาณที่บ่งบอกถึงจุดเริ่มต้นของข่าวสาร (MESSAGE) ที่ส่งมาให้เพจเจอร์แต่ละเครื่องด้วย

การตรวจจับสัญญาณหมายเลขเรียกขาน จากวงจรในรูปที่ 3.12 จากสัญญาณ NRZ ที่ถูก DELEY ไป 1 บิต ถูกป้อนให้กับ 74LS164 โดยที่กำหนดให้สัญญาณหมายเลขเรียกขานเกิด OUTPUT ที่บิตสุดท้ายของรหัสคำหมายเลขเรียกขาน (ซึ่งหมายถึงจะเกิด OUTPUT หลังจากบิตที่ 19 ของรหัสคำหมายเลขเรียกขาน) ดังนั้นจึงต้องต่อขา OUTPUT ของ 74LS164 ที่คาดว่าจะ เป็น “0” ในช่วงเวลาที่เกิดสัญญาณหมายเลขเรียกขานพอดิผ่าน NOT GATE ซึ่งจากวงจร สามารถที่จะเลือกตั้งค่าได้โดยใช้ดิปสวิทช์จากนั้นสัญญาณ OUTPUT แต่ละตัว รวมทั้งสัญญาณ OUTPUT ของ 74LS164 ที่เหลือด้วย จะถูกต่อเข้ากับ NAND GATE ที่มี INPUT 8 ขาสัญญาณ OUTPUT ที่ได้จาก NAND GATE จะถูกต่อเข้ากับ OR GATE ซึ่งจะได้ OUTPUT เป็นสัญญาณ หมายเลขเรียกขานออกมาทุกครั้งที่มีการรับข่าวสารต่อการส่งสัญญาณ 1 ครั้ง

ภาคสร้างสัญญาณอินเตอร์รัพท์แอดเดรสปลั๊กโคด (Interrupt Address Plugcode)

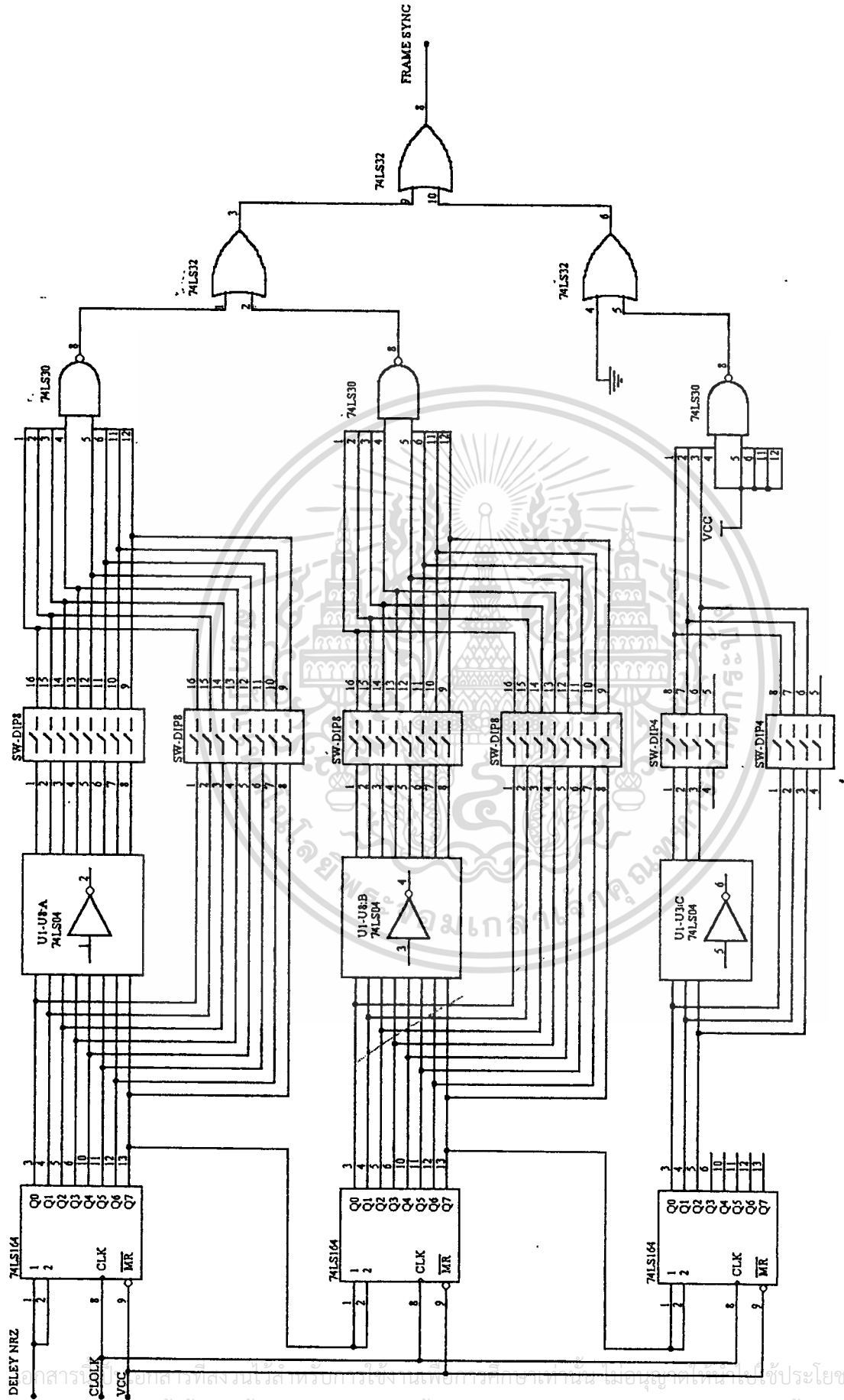
จากการที่มีการกำหนดให้หมายเลขเรียกขานของเครื่องรับเพจเจอร์ แต่ละเครื่องนั้นแตกต่างกันมีผลทำให้สัญญาณหมายเลขเรียกขานของแต่ละเครื่องถูกส่งมาอยู่ในเฟรม (FRAME) ที่แตกต่างกันด้วย ซึ่งจะสามารถทราบตำแหน่งของเฟรม (FRAME) ที่ส่งหมายเลขเรียกขานได้โดยการนำเอาหมายเลข PLUGCODE ที่เป็นหมายเลขประจำตัวเครื่องรับเพจเจอร์ ซึ่งเป็นเลขฐาน 10 แปลงให้อยู่ในรูปเลขฐาน 2 จะได้ทั้งหมด 21 บิต แต่ 3 บิตหลังที่มีความสำคัญน้อย (LSB) คือ

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

บิตที่ 19-21 ที่ไม่ได้ทำการส่ง จะเป็นตัวกำหนด เฟรม (FRAME) ที่ใช้ในการส่งหมายเลขเรียกขานของแต่ละเครื่องรับเพจเจอร์

วงจรนับเฟรมของหมายเลขเรียกขาน จากวงจรในรูปที่ 3.13 สัญญาณนาฬิกาที่มีความถี่ 1200 bps และสัญญาณ FRAME SYNC จะถูกป้อนเข้าที่ 74LS161 ตัวแรกซึ่งสัญญาณนาฬิกาจะถูกนับ 16 ได้ OUTPUT ออกมา 1 ครั้ง จากนั้นสัญญาณ OUTPUT ที่ได้จะถูกต่อผ่าน NOT GATE และป้อนเข้าที่ 74LS161 ตัวที่สองเพื่อทำการนับ 2 ได้ OUTPUT ออกมา 1 ครั้ง จากนั้นนำเอา OUTPUT ที่ได้ของ 74LS164 ทั้งสองมาผ่าน AND GATE เพื่อให้ได้สัญญาณ OUTPUT เป็น “1” ทุกๆ 32 บิต หรือ เท่ากับ 1 CODEWORD ส่วน 74LS161 ตัวที่สองจะนำเอา OUTPUT ที่ได้จาก AND GATE มาทำการนับอีกครั้ง เพื่อให้ได้สัญญาณ OUTPUT ออกมาในช่วงจังหวะที่มีการส่งสัญญาณหมายเลขเรียกขาน ซึ่งสามารถเลือกตั้งค่านับ ได้โดยใช้คิปลสวิทช์ และค่าที่เลือกใช้ในการนับเพื่อให้เกิด OUTPUT ในช่วงที่มีการส่งสัญญาณหมายเลขเรียกขานได้ตรงกับเครื่องรับเพจเจอร์ของแต่ละเครื่อง มีค่าเท่ากับค่าของ บิตที่ 19-21 ของหมายเลขเรียกขานและจะถูกทำการ RESET ให้ทำการนับใหม่ที่ 9 ด้วยสัญญาณที่เกิดจากขาที่ 15 ของ 74LS161 ตัวที่สองและสัญญาณ FRAME SYNC จะทำการ RESET ค่าการนับของ 74LS161 ทุกๆ ตัว เพื่อทำให้เกิดการนับที่เป็นไปอย่างถูกต้อง

จากนั้นนำเอาสัญญาณ OUTPUT ที่ได้จากการตั้งค่านับและสัญญาณ OUTPUT ที่ได้จากภาคตรวจจับสัญญาณหมายเลขเรียกขานมาผ่าน AND GATE จะได้ OUTPUT เป็นสัญญาณที่ใช้ในการ INTERRUPT PC



รูปที่ 3.12 วงจรตรวจจับแอดเดรสหลักโคด

คอมพิวเตอร์ (COMPUTER)

ถือเป็นส่วนสำคัญซึ่งถูกใช้สำหรับการอ่านและแสดงข่าวสารของสัญญาณเพจเจอร์ การทดลองโครงงานนี้ใช้คอมพิวเตอร์แบบ 16 บิต CPU80286 ดังนั้นบัสข้อมูลของระบบจึงมี 16 เส้น แบ่งเป็นไบท์บน 8 เส้น ($D_8 - D_{15}$) และไบท์ล่าง 8 เส้น ($D_0 - D_7$) ซึ่งแต่ละกลุ่มของบัสข้อมูลจะต่อเข้ากับกลุ่มของหน่วยความจำ (memory bank) ที่แตกต่างกันในเครื่อง AT แบบ 16 บิตจะแบ่งหน่วยความจำเป็น 2 กลุ่ม ซึ่งเรียกว่าแบงก์ โดยมีขนาดเท่ากันทั้ง 2 แบงก์ หน่วยความจำแบงก์แรกจะต่อกับไบท์ล่างของบัสข้อมูล ($D_0 - D_7$) โดยมีแอดเดรสเป็นคู่ทั้งหมด (0,2,4,6,...) ส่วนอีกแบงก์หนึ่งจะต่อกับไบท์บนของบัสข้อมูล ($D_8 - D_{15}$) ซึ่งมีแอดเดรสเป็นคี่ทั้งหมด (1,3,5,7,...) เช่นกัน ดังรูปที่ 3.7 สัญญาณที่ถูกใช้สำหรับการถอดรหัสเพื่อเลือกแบงก์หน่วยความจำก็คือสัญญาณ A_0 และ BHE ของ CPU 80286 ในกรณีที่ $A_0 = "0"$ แสดงว่ามีการเลือกแบงก์คู่ แต่ถ้า $A_0 = "1"$ จะเป็นการเลือกแบงก์คี่ ส่วนสัญญาณ BHE จะเป็นตัวบอกว่า CPU ยอมให้มีการใช้ไบท์บนของบัสข้อมูลหรือไม่ในกรณีที่ $BHE = "0"$ แสดงว่ายอมให้ใช้ $D_0 - D_{15}$ แต่ถ้า $BHE = "1"$ แสดงว่าไม่ยอมให้ใช้ $D_8 - D_{15}$ เมื่อเรานำสัญญาณ A_0 กับ BHE มาใช้รวมก็จะก่อให้เกิด 4 สถานะขึ้นตามตารางที่ 3.2

คำสั่งการอ่านหน่วยความจำในภาษา	BHE	A_0	หน้าที่การทำงาน
Peek (แอดเดรสคู่)	0	0	รับส่งข้อมูลเป็นเวิร์ด ($D_0 - D_{15}$)
Peekb (แอดเดรสคี่)	0	1	รับส่งข้อมูลทางไบท์บน ($D_8 - D_{15}$)
Peekb (แอดเดรสคู่)	1	0	รับส่งข้อมูลทางไบท์บน ($D_0 - D_7$)
	1	1	

ตารางที่ 3.2 การอ่านข้อมูลจากหน่วยความจำด้วยวิธีต่างๆ

ภาษา C มีฟังก์ชันสำหรับการอ่านข้อมูลจากหน่วยความจำอยู่ 2 ชนิด คือ peek สำหรับการอ่านหน่วยความจำที่ละเวิร์ด (16 บิต) และ peekb สำหรับการอ่านหน่วยความจำที่ละไบท์ (8 บิต) ในกรณีที่ใช้ฟังก์ชัน peek CPU จะอ่านข้อมูลจากหน่วยความจำทั้ง 16 เส้น เช่นถ้าสั่ง peek (OXDE00.0) CPU ก็อ่านข้อมูลจาก DE00:0 ผ่านทาง ($D_0 - D_7$) และอ่านข้อมูลจาก DE00:1 ผ่านทาง ($D_8 - D_{15}$) รวมเป็น 1 เวิร์ดเช่นกัน ส่วน peekb เป็นการอ่านหน่วยความจำทีละไบท์ตามแอดเดรสที่กำหนดให้โดยตรงเช่น peekb (OXDE00.0) CPU ก็อ่านข้อมูลจาก DE00:1 ผ่าน

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้าไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

ทาง ($D_8 - D_{15}$) ข้อจำกัดนี้ก็ยังคงมีอยู่ แม้แต่ภาษา (assemble) ก็ไม่มีข้อยกเว้น เช่น ถ้าเราส่ง `MOV AX,WORD [DE00H:1]` ก็จะมีผลเช่นเดียวกับ `peek (OXDE00.1)` นั่นคือการอ่านข้อมูลจาก DE0 0:1 ผ่านทาง ($D_8 - D_{15}$) และอ่านข้อมูลจาก DE00:2 ผ่านทาง $D_0 - D_7$ เช่นกัน ด้วยข้อจำกัดนี้เองเราจึงไม่สามารถอ่านข้อมูลจากหน่วยความจำที่มีแอดเดรสคี่ผ่านทาง $D_0 - D_7$ ซึ่งต่ออยู่กับแรมอีมูเลเตอร์ได้โดยตรง ทางออกสำหรับปัญหานี้มี 2 วิธีคือ

1.บันทึกข้อมูลลงทุกแอดเดรสของแรมอีมูเลเตอร์ และดัดแปลงวงจรเพื่อถอดรหัสบัสแอดเดรสของคอมพิวเตอร์ โดยทำการหารแอดเดรสคอมพิวเตอร์ด้วย 2 ก่อนที่จะต่อกับแอดเดรสของแรมอีมูเลเตอร์ดังตารางที่ 3.3

แอดเดรสจากคอมพิวเตอร์	แอดเดรสของแรมอีมูเลเตอร์
DE00:0	DE00:0
DE00:2	DE00:1
DE00:4	DE00:2

ตารางที่ 3.3 การเปลี่ยนรหัสแอดเดรสของคอมพิวเตอร์มาเป็นแอดเดรสของแรมอีมูเลเตอร์

แม้วิธีการนี้จะใช้แรมได้อย่างมีประสิทธิภาพ แต่ก็เพิ่มความยุ่งยากในการออกแบบและการสร้างวงจรมากขึ้น อีกทั้งข้อมูลเพจเจอร์ที่รับเข้ามาจากการส่งแต่ละครั้งก็ไม่มากจนถึงกับต้องใช้แรมให้ครบ 8 กิโลไบต์ ดังนั้นจึงนำมาสู่วิธีแก้ถัดมาคือ

2.บันทึกข้อมูลเฉพาะแอดเดรสคู่ของแรมอีมูเลเตอร์เท่านั้นและให้คอมพิวเตอร์อ่านเฉพาะข้อมูลแอดเดรสคู่เหล่านั้นผ่าน $D_0 - D_7$ ได้ ซึ่งวิธีการนี้ถูกใช้อยู่ในวงจรการทดลองจริงดังจะเห็นได้จากรูปที่ 3.11 ว่าหา A_0 ของ 6264 ถูกต่อลงกราวด์หรือเป็น “0” ตลอดและหา A_0 จากคอมพิวเตอร์จึงไม่จำเป็นต้องใช้ สำหรับโปรแกรมที่ใช้ถูกเขียนและคอมไพล์ (compile) บนเทอร์โบซี (Turbo C Version 2.0) สาเหตุที่เลือกใช้ภาษาซีในการเขียนโปรแกรมก็เนื่องจากเป็นภาษาที่มีโครงสร้างกระชับและมีฟังก์ชันต่างๆ ให้ใช้งานมาก อีกทั้งมีคุณสมบัติพิเศษในการทำงานกับข้อมูลที่เป็นบิต อาศัยหลักการของ บิตฟิลด์ (bit field) ซึ่งเหมาะสมกับโครงงานนี้ เนื่องจากข้อมูลเพจเจอร์ที่ถูกอ่านเข้ามานั้นตัวโปรแกรมจะต้องถอดรหัสจาก 1 บิตของข้อมูล, 4 บิต หรือ 18 บิต ซึ่งภาษาคอมพิวเตอร์บางตัวไม่สามารถทำได้และค่อนข้างยุ่งยาก ลักษณะเด่นอีกประการหนึ่งของภาษาซีก็คือ มีการทำงานที่รวดเร็วใกล้เคียงกับภาษาแอสเซมบลีมากกว่าภาษาอื่นๆ ซึ่งนับว่าจำเป็นมากสำหรับการทำงานในลักษณะตามเวลาจริง หมายความว่า โปรแกรมจะ

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

ต้องอ่านข้อมูลจากแรมอิเมเลเตอร์ให้เสร็จทัน ก่อนที่การส่งสัญญาณเพจเจอร์ครั้งต่อไปจะมาถึง และการแสดงข่าวสารจะต้องเกิดขึ้นเกือบจะทันทีที่สัญญาณส่วนที่เป็นข่าวสารนั้นๆมาถึงและเพื่อความสะดวกในการเขียนโปรแกรมได้มีการสร้างเฮดเดอร์ไฟล์ (header file) ขึ้นมาต่างหาก สำหรับการกำหนดตัวแปรชนิดพิเศษ รวมทั้งค่าคงที่ต่างๆ ซึ่งถูกเรียกจากโปรแกรมหลักเป็นประจำ

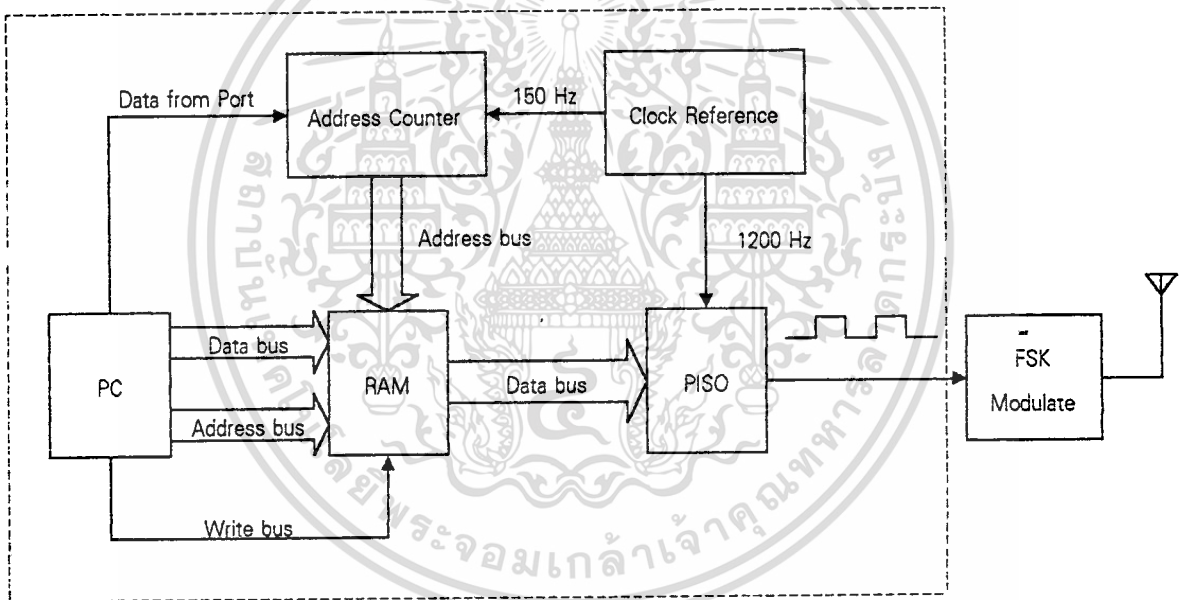


บทที่ 4

ภาคส่งสัญญาณเพจเจอร์

ภาคส่งสัญญาณเพจเจอร์ประกอบด้วยส่วนต่างๆแสดงเป็นบล็อกไดอะแกรมดังรูปที่ 4.1

1. PISO
2. ภาคอ้างอิงสัญญาณนาฬิกา (Reference Clock)
3. ภาคนับแอดเดรส (Address Counter)
4. ภาคเลือกพอร์ตและอินาเบิลเซนต์ (Port Select and Enable Sent)
5. แรมอีมูเลเตอร์ (RAM EMULATOR)



รูปที่ 4.1 แสดงบล็อกไดอะแกรมหลักการทำงานของภาคส่งสัญญาณเพจเจอร์

การทำงานของ Pager Transmitter Card

เมื่อข้อมูลของ Pager พร้อมที่จะทำการส่ง จะถูกเขียนลงแรมในตำแหน่ง DE00: ด้วยคำสั่งทาง software จากนั้นข้อมูลที่จะส่งก็จะอยู่ในแรมพร้อมที่จะถูกส่งออกไปได้ตลอดเวลา ซึ่งการส่งออกไปจะใช้คำสั่งทาง software อีกเช่นกัน

เมื่อทำการส่งให้ Pager Card ทำการส่งข้อมูล คอมพิวเตอร์จะส่งข้อมูลไปสั่งให้วงจร

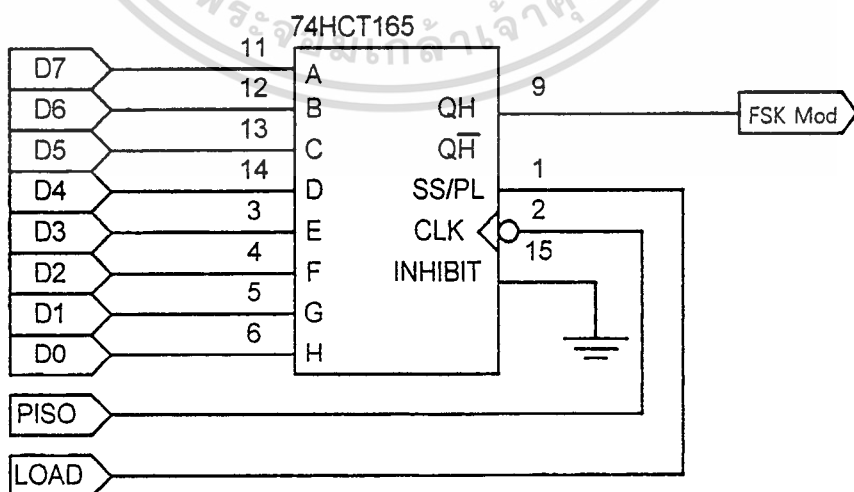
ภาค Address Counter ให้เริ่มนับ (Access Memory) ทาง port หมายเลข 3F0_H ให้เริ่มนับ Access

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

Memory จากนั้น วงจรดังกล่าวจะเริ่มทำงาน ส่งค่าแอดเดรสผ่าน address bus เข้าสู่แรมหลังจากนั้น แรมก็จะส่งข้อมูลขนาด 8 บิต ตามแอดเดรสนั้นๆ ผ่านทาง data bus ไปส่งวงจรภาค PISO ทำให้วงจรภาค PISO ทำงานและส่งออกข้อมูลที่ละบิตออกทางขาเอาต์พุตจนครบ 8 บิต Address Counter ก็จะเพิ่มค่า address ขึ้นอีก 1 ให้กับวงจรภาค RAM EMULATOR อีกแล้ว แรมก็จะส่งข้อมูลให้กับ PISO ต่อไปอีก จนครบทุกตำแหน่งแอดเดรสแล้ว Address Counter ถึงจะหยุดนับ แอดเดรสจนกว่าจะมีการสั่งให้ส่งข้อมูลใหม่อีกครั้ง

จากการทำงานดังกล่าวจะต้องอาศัยวงจร Clock Reference ควบคุมให้วงจร ทุกภาค ทำงาน ร่วมกันอย่างพร้อมเพรียงอย่างมี step จาก Block diagram จะเห็นว่า Clock Reference จะสร้างความถี่ clock ขึ้นมา 2 ความถี่ คือ 150 Hz กับ 1200 Hz เพราะมีเหตุผลดังนี้ คือ ในการส่งระบบ POCSAG นั้นมาตรฐานการส่งข้อมูลจะได้ความเร็วในระดับ 1200 Hz ดังนั้นจึงต้องสร้าง clock ส่งให้กับภาค PISO ให้ทยอยส่งข้อมูลออกไปให้ได้ตามมาตรฐานดังกล่าว และที่ต้องสร้างความถี่ 150 Hz ขึ้นมาด้วยก็เพราะว่า ภาค Address Counter จะต้องนับ address เพิ่มขึ้น 1 address หลังจากที PISO ส่งข้อมูลออกไปทีละบิตจนครบ 8 บิต ดังนั้นความถี่ Clock ที่จะควบคุมภาค Address Counter นั้นจะต้องถูกหาร 8 ลงจาก 1200 Hz เหลือ 150 Hz เพื่อให้การทำงานของระบบเป็นไปโดยพร้อมเพรียงกัน

PISO (Parallal In Serie Out)



รูปที่ 4.2 แสดงวงจร PISO

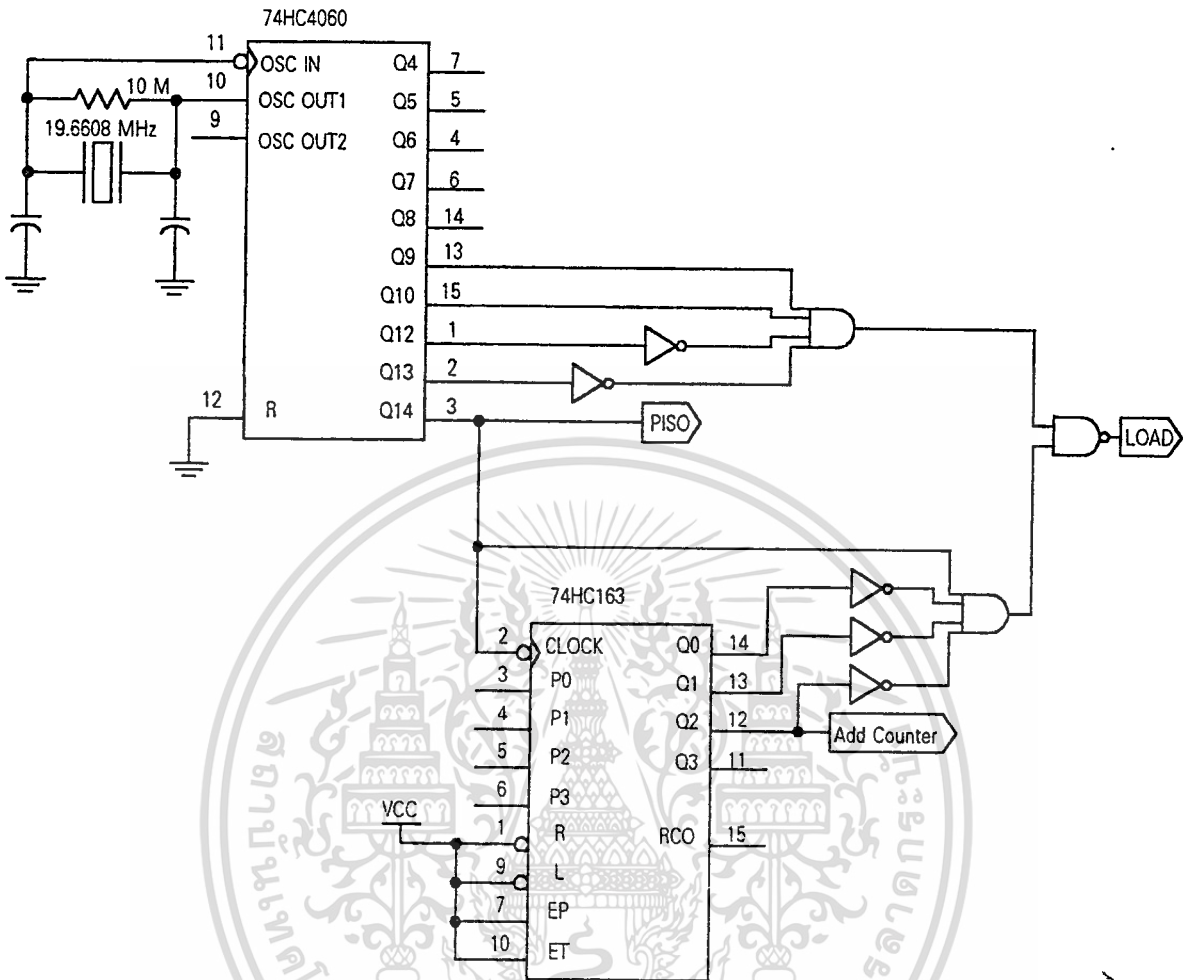
เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้าไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

เนื่องจากข้อมูลที่จะส่งนั้น ต้องเป็นข้อมูลแบบอนุกรม (เพื่อนำไป Mod เข้ากับคลื่นพาหะแล้วจึงส่งสัญญาณออกไป) แต่การคั่นจะต่อกับ Port COM ซึ่งจะมีการอ่านและเขียนข้อมูลแบบขนานจึงต้องนำข้อมูลที่จะส่งนั้นทำให้เป็นข้อมูลในแบบอนุกรมก่อน โดยจะใช้วงจรดังรูปที่ 4.2

หลักการของ PISO คือ การเลื่อนบิตข้อมูลไปเรื่อยๆ และทุกๆ 8 บิต จะถูกอ่านจากแรม 1 ครั้ง ซึ่งใช้ 74HCT165 (8 bit shift register) โดยใช้สัญญาณนาฬิกาจากภาค Reference Clock เป็นสัญญาณในการ Load จะอ่านข้อมูลออกมาทีละ 1 แอดเดรส (8 บิต) แล้วจะนำไป LOAD ข้อมูลในแอดเดรสต่อไปจนครบ 4K โดยมีสัญญาณ PISO เป็นตัวรับข้อมูลที่ได้จากการ LOAD ส่งออกไป

ภาคอ้างอิงสัญญาณนาฬิกา (Reference Clock)

การทำงานของภาคดังกล่าวจะใช้ 19.6608 MHz ผลิตความถี่อ้างอิงให้กับ IC เบอร์ 4060 จะได้เอาท์พุทจากขา Q14 (หารด้วย 64) ซึ่งมีสัญญาณนาฬิกาเป็น 1200 Hz เพื่อนำสัญญาณนี้ไปอ้างอิงให้กับวงจร PISO และนำสัญญาณ 1200 Hz นี้ไปหารด้วย 8 โดยใช้ 74HC163 ทำให้ได้สัญญาณนาฬิกา 150 Hz เพื่อนำสัญญาณนี้ไปอ้างอิงให้กับวงจรภาค Address Counter ส่วนสัญญาณ LOAD นั้น จะนำไปอ้างอิงให้กับภาค PISO เช่นกันแต่จะเป็นสัญญาณพัลส์ลบแคบๆ ทุกๆ 8 ลูกของสัญญาณนาฬิกา 1200 Hz



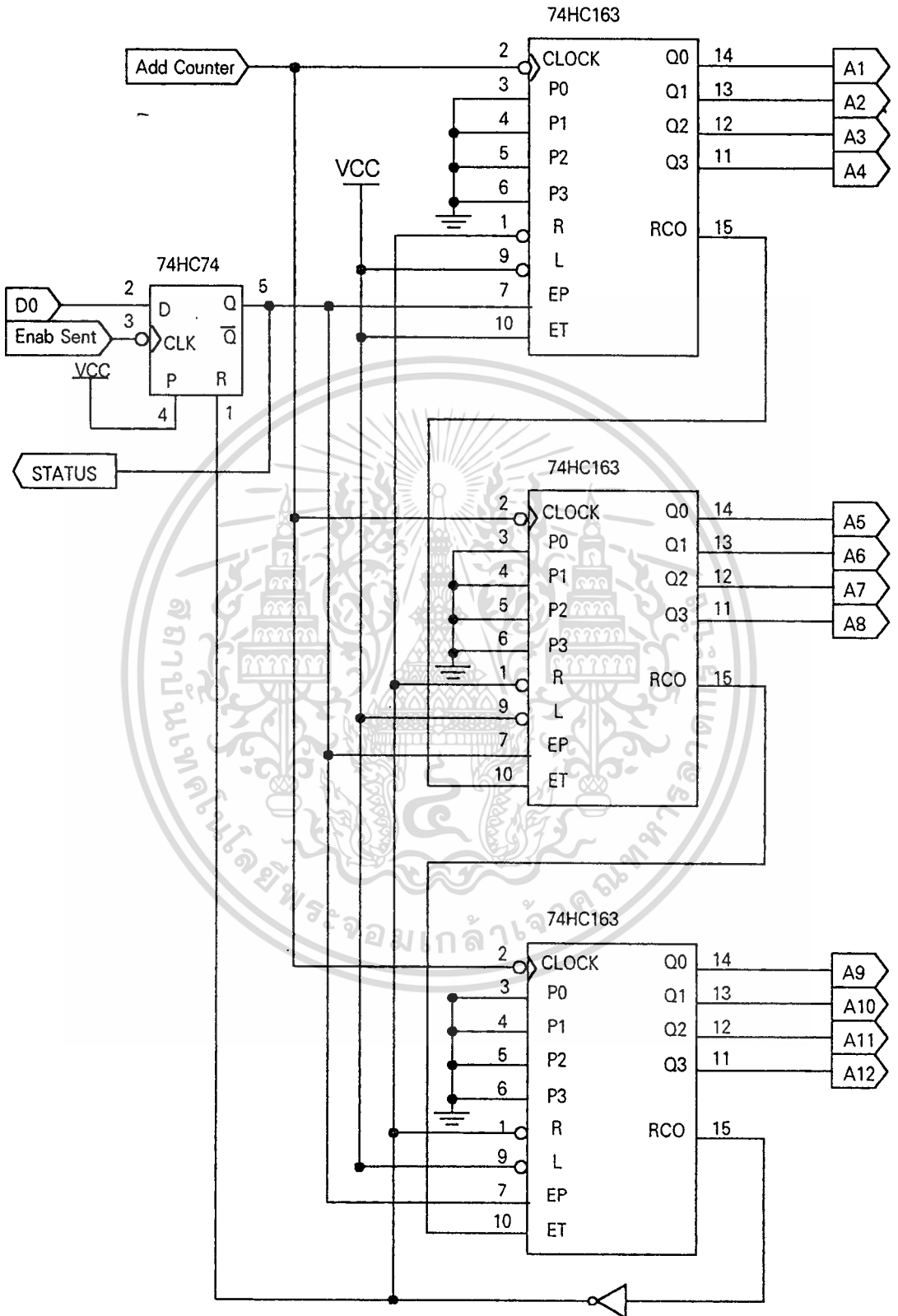
รูปที่ 4.3 วงจรอ้างอิงสัญญาณนาฬิกา

ภาคนับแอดเดรส (Address Counter)

วงจรจะใช้ IC เบอร์ 74HC163 3 ตัวเป็นตัวนับ Address โดยจะมี Address ทั้งหมด 12 เส้น ดังนั้นจึงสามารถที่จะจับ Address ได้ทั้งหมด $4k$ ($2^{12} = 4096$) โดยใช้ clock 150 Hz จากภาค Reference Clock เป็นสัญญาณอ้างอิง เมื่ออยู่ในสถานะที่ยังไม่มีการสั่งให้ส่งข้อมูลหาเอาต์พุตของ Address จะเป็น High Impedance อยู่ตลอดเวลา

เมื่อมีการสั่งให้ส่งข้อมูล โดยส่งข้อมูลออกมาทาง D0 ของ Port COM IC 74LS74 จะทำการเก็บค่าดังกล่าวไว้ เอาต์พุตของ 74LS74 จะไปทำการ Enable อินพุตของ 74HC163 ทำให้เริ่มมีการนับ Address ออกทางเอาต์พุตจนครบ $4k$ แล้ววงจรจะทำการ Reset ตัวเอง โดยการ Reset ค่าที่เก็บไว้ใน Flip Flop ทำให้ 74HC163 ถูก Disable ไม่มี Address ออกทางเอาต์พุต

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้



รูปที่ 4.4 วงจร Address Counter

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า
ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

เปรียบเทียบกับสัญญาณ “0” เพื่อให้แน่ใจว่าเอาต์พุตที่ออกมาจะไม่เป็นการ Direct Memory Access (ถ้ามีการ Direct Memory Access สัญญาณจากขา AEN = 1) แล้วเอาสัญญาณแต่ละอันมาแอนด์ (and) กันดังวงจรที่ 4.5 ซึ่งจะได้สัญญาณ Enable Sent ออกมาที่เอาต์พุต โดยสัญญาณนี้จะเกิดขึ้นเมื่อมีการเลือกพอร์ท 3F0 โดยมีลักษณะสัญญาณเป็นพัลส์ช่วงสั้นๆ แล้วนำเอาต์พุตที่ได้ไปเข้าขา Clock ของ Flip Flop ในภาค Address Counter

แรมอีมูเลเตอร์ (RAM EMULATOR)

แรม จะทำหน้าที่เป็น 2 WAY RAM EMULATOR (แรม 2 ทาง) โดยทางแรกจะติดต่อกับคอมพิวเตอร์ทางที่ 2 ติดต่อกับตัวการ์ดเองดังนั้นแอดเดรสในการที่จะเข้าถึงแรมจึงต้องมาจากคอมพิวเตอร์หรือว่าจากการ์ด โดยจะใช้ IC เบอร์ 74LS157 3 ตัว ในการเลือกว่าจะติดต่อกับแรมโดยผ่านทางแอดเดรสจากคอมพิวเตอร์หรือจากแอดเดรสของการ์ดสัญญาณจาก status จะเป็นตัวเลือกว่าจะใช้แอดเดรสจากส่วนใดโดยถ้า status เป็น “0” จะหมายถึงว่าขณะนั้นแรมติดต่อกับคอมพิวเตอร์ แต่ถ้าหาก status เป็น “1” แสดงว่าขณะนั้นแรมติดต่อกับการ์ด จากวงจรจะมีการหา mapping ตำแหน่งแรมไว้ที่ DE00 ซึ่งตำแหน่งดังกล่าวนี้ เส้นแอดเดรส $A_{15}-A_{19}$ จากคอมพิวเตอร์จะต้องมีลอจิกตามตาราง

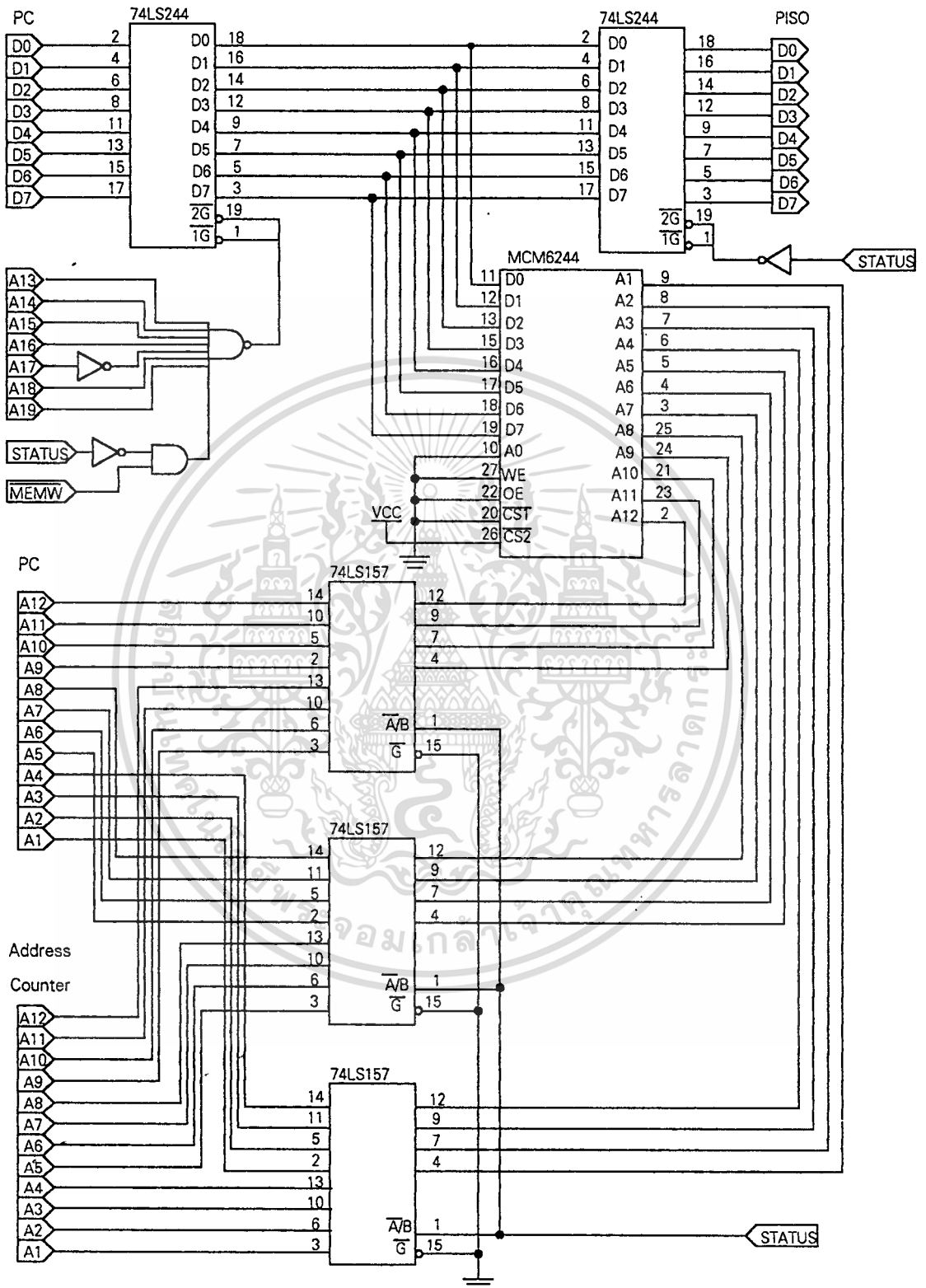
Address	A_{19}	A_{18}	A_{17}	A_{16}	A_{15}	A_{14}	A_{13}	A_{12}		A_1	A_0
Logic	1	1	0	1	1	1	1	X		X	0

ตารางที่ 4.1 แสดงลอจิกที่แอดเดรสเมื่อแรมอยู่ที่ DE00

การติดต่อของแรมกับคอมพิวเตอร์จะเป็นรูปแบบทางเดียว คือ จะส่งข้อมูลมาจากคอมพิวเตอร์ เพื่อเขียนลงแรมได้เท่านั้น ไม่สามารถอ่านข้อมูลจากแรมไปให้คอมพิวเตอร์ได้ เนื่องจากออกแบบโดยใช้ 74LS244 เป็น Buffer ให้กับ databus (74LS244 เป็น Buffer ทางเดียว) ดังนั้นเมื่อแรมติดต่อกับคอมพิวเตอร์เมื่อใดก็หมายถึงจะมีข้อมูลเขียนลงในแรมสัญญาณจาก $\overline{MEMW}$ จะเป็นตัวที่ทำให้แรมยอมรับที่จะเขียนข้อมูล โดยจะใช้สัญญาณลอจิก “0” จาก $\overline{MEMW}$ ต่อเข้าขา $\overline{ME}$ ของแรมเพื่อให้แรมเขียนข้อมูล

ส่วนการติดต่อของแรมกับการ์ดจะถูกต่อเมื่อ status เป็น “1” และเมื่อใดก็ตามเมื่อ status เป็น “1” ก็หมายถึงว่ามี การอ่านแรมออกไปจาก Address Counter จากการ์ดจะเป็นตัวเข้าถึงตำแหน่งของแรมและข้อมูลที่ออกจาก data bus ของแรมจะถูกส่งให้กับภาค PISO โดยผ่าน buffer (IC เบอร์ 74LS244)

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้



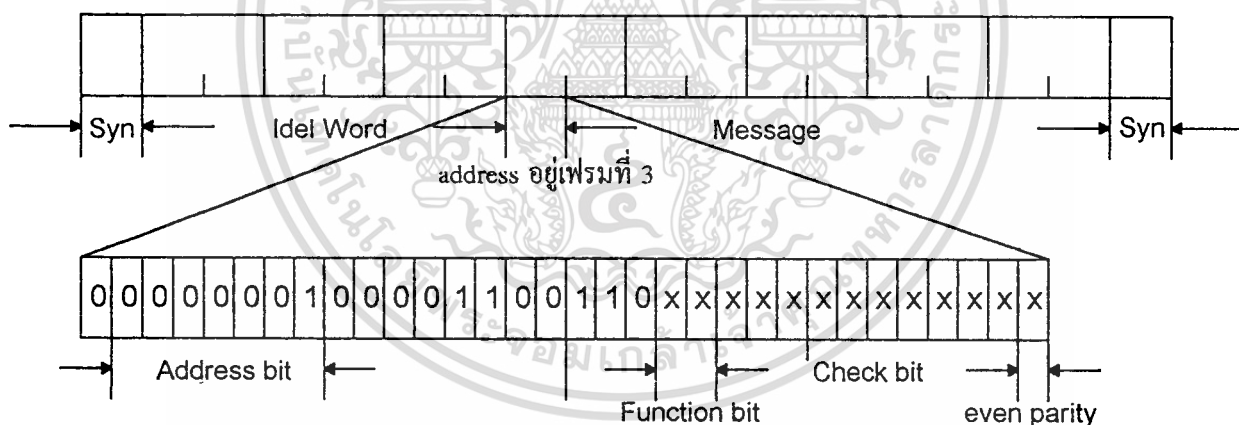
รูปที่ 4.6 แสดงวงจรของแรม

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า
ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

การจัดข้อมูลเพื่อการส่ง

ในการจัดรูปแบบเกี่ยวกับข้อมูลของ pager มีรูปแบบเฉพาะของมันเอง ดังได้กล่าวรายละเอียดไปแล้วและรหัสดังกล่าวเรียกว่า POCSAG (Post Office Code Santandard Advisory Grope) ประกอบด้วย Preable 576 บิต และ Batch Structure ภายใน Bath Structure จะประกอบด้วย Synchronization Code Word , Address Code Word , Message Code Word และ Idle Code Word การจัดรูปแบบที่จะส่งนี้ เราใช้ Software ในการจัด อย่างเช่น หากเราต้องการส่งข้อมูลว่า kmitl - tech ออกไปที่เครื่องหมายเลข Plugcode 17203 จะมีขั้นตอนดังนี้

1. แปลง Plugcode 17203 เป็นรหัส binary จะได้ 0100001100110011
2. รหัส binary จาก Plugcode ที่ได้ให้ตัด 3 บิต หลังออกจะเหลือ 01000011001110
3. รหัส binary จาก Plugcode 3 บิต หลัง คือ $011_2 = 3_{10}$ จะเป็นตัวกำหนดว่าจะนำ binary จาก Plugcode ที่เหลือไปไว้ในเฟรมที่ 3 ของเบทซ์แรก ซึ่งเฟรมจะมีตั้งแต่เฟรมที่ 0-7 จากนั้นข้อความที่เหลือไปไว้ในเฟรมถัดไปของเบทซ์แรก เมื่อนำเอาผลที่ได้มาลงในเบทซ์แรกจะได้ดังนี้



รูปที่ 4.7 แสดงถึงตำแหน่งของแอดเดรสในรหัสคำ

4. ต่อจากแอดเดรสก็จะเป็นข้อความที่จะต้องจัดคือ kmitl-tech แปลงตัวอักษรทั้งหมดให้อยู่ในรูป binary จะได้

k = 1101011

m = 1011011

i = 1001011

t = 0010111

l = 0011011

- = 1111101

t = 0010111

e = 1010011

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

$c = 1100011$

$h = 0000011$

นำบิตของข้อความมารวมกันทั้งหมดจะเป็น

1	1	0	1	0	1	1	1	0	1	1	0	1	1	1	0	0	1	0	1	1	0	0	1	0	1	1	1	0	0	1	1	0	1	1
1	1	1	1	1	0	1	0	0	1	0	1	1	1	1	0	1	0	0	1	1	1	1	0	0	0	1	1	0	0	0	0	0	1	1

ตารางที่ 4.2 รูปแบบของรหัสที่ต้องการส่ง

จะได้บิตรวมทั้งหมด $7 \times 10 = 70$ บิต แต่เนื่องจาก 7 code word มี เนื้อที่สำหรับ bit message อยู่ 20 บิต ดังนั้นจึงต้องแบ่ง bit message ออกเป็นกลุ่มละ 20 บิต เพื่อนำลงไปใน code word จะได้ดังตารางที่ 4.2 ถ้าหากข้อมูลยังไม่เต็มแบทช์จะได้ Idle code word เข้าไปจนเต็ม

1	1	0	1	0	1	1	1	0	1	1	0	1	1	1	0	0	1	0	1	1
0	0	1	0	1	1	1	0	0	1	1	0	1	1	1	1	1	1	0	1	
0	0	1	0	1	1	1	1	0	1	0	0	1	1	1	1	0	0	0	1	1
0	0	0	0	0	1	1														

ตารางที่ 4.3 แสดงถึงการแบ่งรหัสคำในแต่ละ code word

การทำงานของโปรแกรม

จะมีโปรแกรมอยู่ 3 โปรแกรมหลักๆคือ

1. โปรแกรมสำหรับแสดงผล (message)

ส่วนของโปรแกรมแสดงผล message จะใช้สำหรับแสดงข้อมูลที่เป็น message สำหรับข้อมูลนี้มีการรับมาเรียบร้อยแล้วจาก Reciver Card จะแสดงผลข้อมูลออกทางจอภาพ โดยจะสามารถแสดงเฉพาะ message หรือแสดงข้อมูลเป็น Codeward ได้ โดยใช้โปรแกรม File menu.exe

ภายในโปรแกรมเมนูจะมีฟังก์ชันย่อยที่จะทำหน้าที่ต่าง ๆ กันดังนี้

save_screen () เก็บหน้าจอ

get_screen () คืนหน้าจอ

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

tell_you ()	กำหนดขนาดของ window
popup_menu ()	แสดงเมนูแบบ popup
put_menu ()	แสดงเมนูครั้งแรกเมื่อ RUN program
put_strint ()	แสดงข้อความออกแบบจอภาพ
border_line ()	ตีกรอบ window
put_one-char ()	แสดงตัวอักษร 1 ตัวทางจอภาพ
get_resp ()	ตอบสนองการเลือกเมนู
one_()	แสดงข้อมูลเพจเจอร์
Sent_()	ส่งเพจเจอร์
loop_delay ()	หน่วงเวลา
print_alpha ()	แสดงข้อมูลเพจเจอร์เป็น charecter
monitor ()	แสดงข้อความ
current_directory ()	แสดงไดเรกทอรีปัจจุบัน
type_bits ()	แสดงเพจเจอร์ออกเป็นบิต
swab_bit ()	สลับบิตของ plugcode
tranfer_codeword ()	จัดการเกี่ยวกับโครงสร้างข้อมูล
chang_to_ascii ()	แปลง code ตัวเลขเป็น ascii code

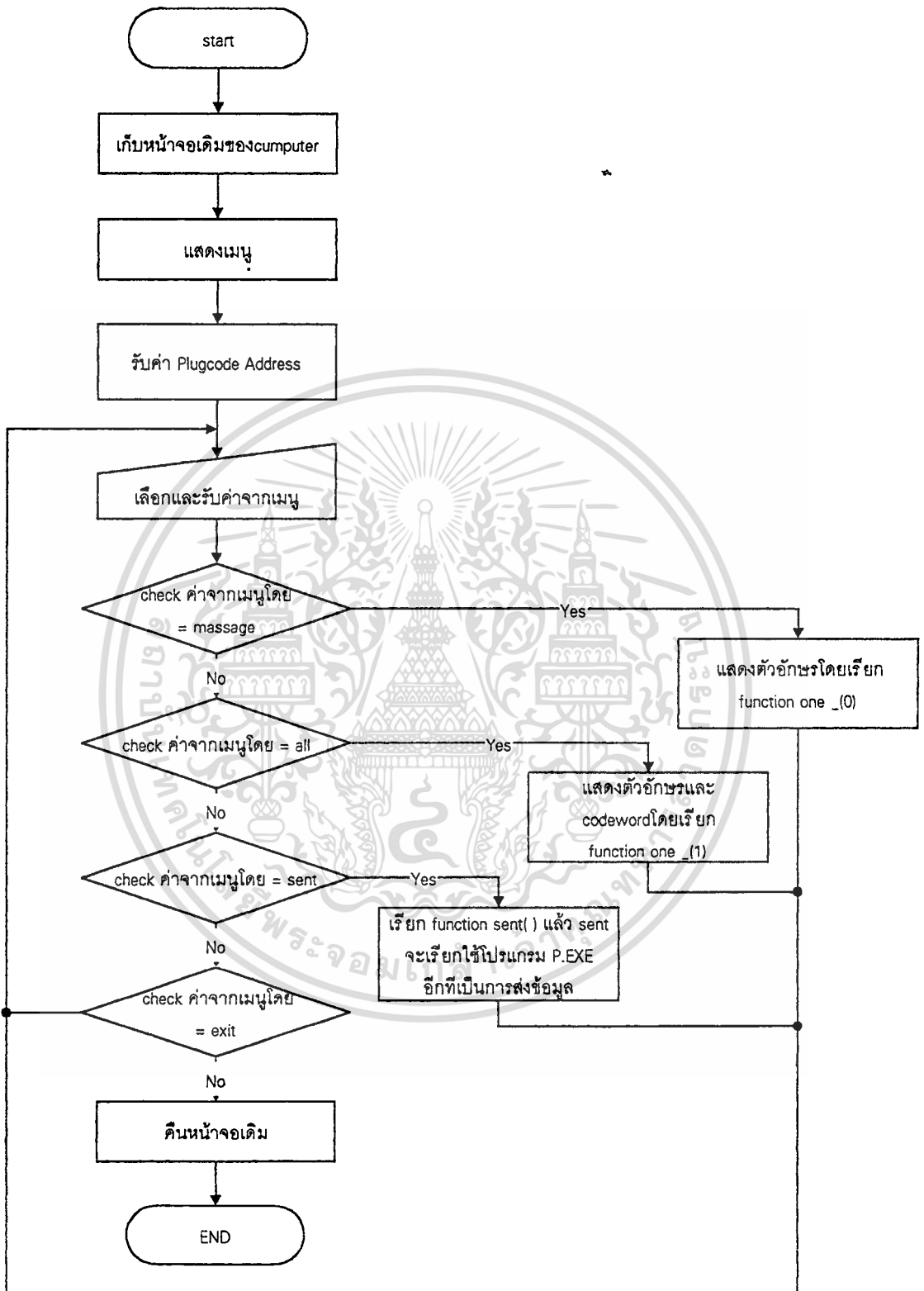
2. โปรแกรม Write memory

ส่วนของโปรแกรม write memory จะเป็นข้อมูลที่อยู่ในมาตรฐานของ POCSAG ลงใน memory ของ transmitter เพื่อที่จะรอการส่งโดยโปรแกรมในส่วนนี้แยกออกเป็น 2 ประเภทคือ

- โปรแกรมที่ write ข้อมูลของ pager จริงๆที่ได้ทำการบันทึกได้จากเครื่องส่ง โดยใช้ File putdata.exe รันเพื่อทำการโหลด File ที่เป็น File Data อีกทีหนึ่ง
- โปรแกรมที่ simulate ข้อมูลขึ้นมาเอง puta.exe, putb.exe

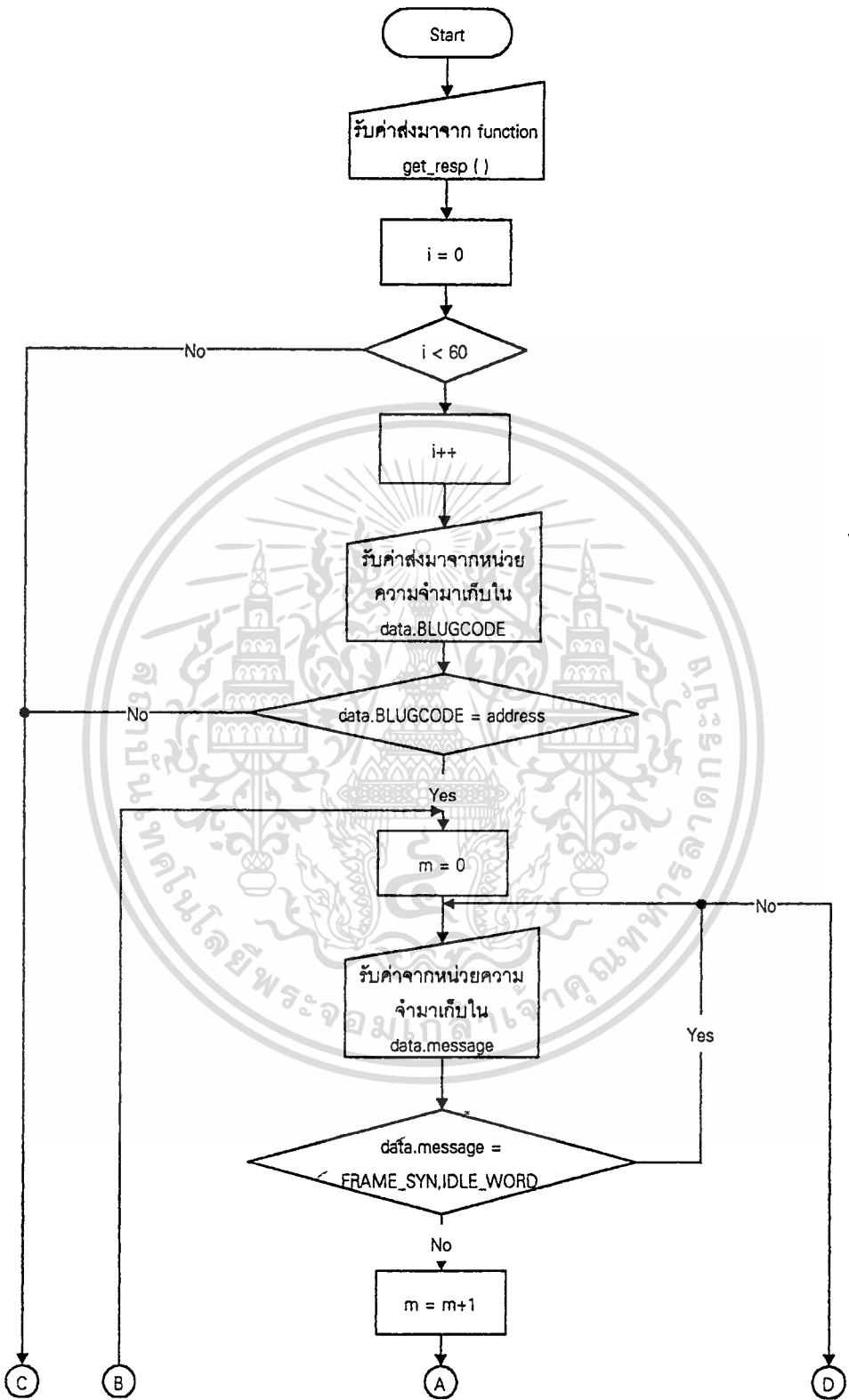
3. โปรแกรมสำหรับสั่งให้เริ่มส่ง message

โปรแกรมสำหรับสั่งให้ส่งข้อมูลออกทาง port 3F0 เพื่อสั่งให้ (Hard ware) Pager Transmitter Card ทำการส่งข้อมูลที่เครื่องไว้ใน memory ของไป คือ File P.exe และ File P.exe นี้ได้นำไปรวมใน menu ของโปรแกรม menu.exe ด้วย ดังนั้นจึงสามารถเรียกใช้ File P.exe ได้โดยเลือก menu Sent

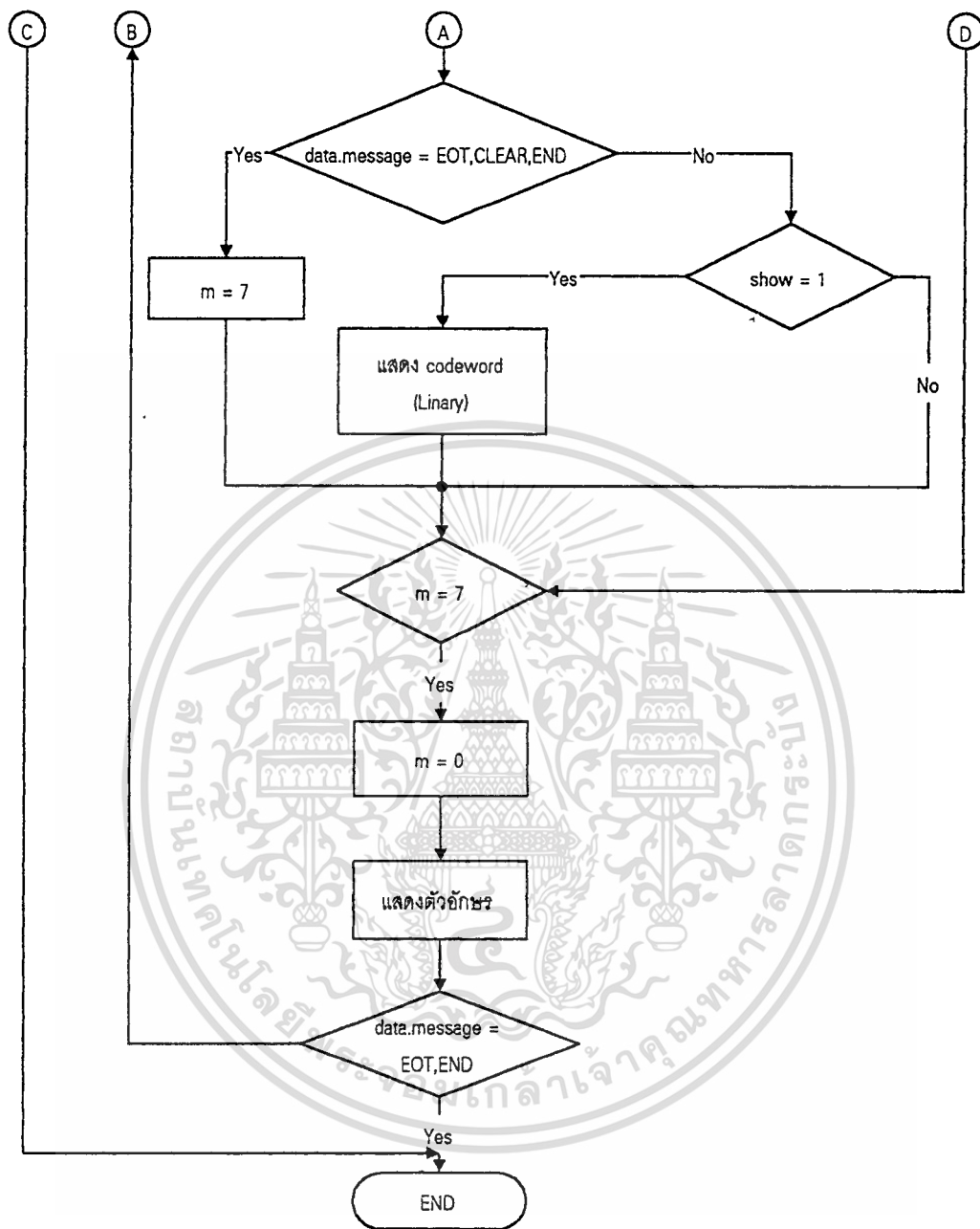


รูปที่ 4.8 flowchart แสดงการทำงานของโปรแกรม menu.exe

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า
ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้



เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า
ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้



รูปที่ 4.9 flowchart การแสดงผลของ function one_()

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);
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า
ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```

unsigned int print_alpha(int sx,int attrib);

void monitor(int page);

char *current_directory(char *path);

void type_bits(DOUBLE_WORD data1);

DOUBLE_WORD swab_bit(DOUBLE_WORD temp);

CHECK_MESSAGE transfer_codeword(ENTRY_MESSAGE entry_message1);

char chang_to_ascii(int codein);

```

```

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

```

```

/*-----*/
/*      Function Use in Programe      */
/*-----*/
/*-----*/

char *current_directory(char *path)

```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า
ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```

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++)
        {
            gotoxy(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])
{

```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า
ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```

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;

```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า
ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```

    _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);
    put_string(10,3,curdir,28);
    for(;;)
    {
        arrow = bioskey(0);
        put_string(x,y,menu_line[i],30);
        switch(arrow)

```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า
ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```

{
    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;
    }
}

/*-----*/

```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```

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;j<=yy;j++)
        for(i=x;i<=xx;i++)
        {
            gotoxy(i,j);
            put_one_char(one,attrib);
        }
}

```

```

    }

}

/*-----*/
/*          Main Programme          */
/* -Chang 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);

```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า
ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```

    curx=_DL;cury=_DH;

    x=1;y=1;xx=78;yy=16;
    save_screen(x,y,xx,yy,vid_buff);
    put_string(16,yy-1,"Pager Emulator Programe "
                "Version 3 By Pager Grope",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 NUMERIC 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--;
}
}

```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

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

```
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)
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า
ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```

{
}

/*-----*/

void one_(int show)
{
    unsigned OFFSET=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,OFFSET);
            OFFSET+=2;
        }
        if((data.BLUGCODE.b2_10==address.BLUGCODE.b2_10)
            &&(data.BLUGCODE.b11_19==address.BLUGCODE.b11_19))
        {
            for(;;)
            {
                do{
                    for(j=0;j<4;j++)
                    {

```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า
ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```

data.BYTE[j]=peekb(SEGMENT,OFFSET);
OFFSET+=2;
}
}while((data.CODEWORD==FRAME_SYN)
||(data.CODEWORD==FRAME_SYN-4)
||(data.CODEWORD==IDLE_WORD)
||(data.CODEWORD==IDLE_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)
{

```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```

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) /*getch()*/
{
    if(sx==Escap) break;
}
if((data.CODEWORD==EOT)||(data.
CODEWORD==CLEAR)
||(data.CODEWORD==END))
goto END_;

}

}

END_:
OFFSET=0;

}

/*-----*/

```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```

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=chang_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;

```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า
ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```

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 %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(" %10lu\n",data1.CODEWORD);
}
/*-----*/
DOUBLE_WORD swab_bit(DOUBLE_WORD temp)
{

```

```

    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;

```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า
ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```

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_MESSAGE transfer_codeword(ENTRY_MESSAGE entry_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_M1;
    temp.CHECK_MCW1.W1_M3 = entry_message1.MESSAGE_7CW.W1_M1;

    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;

```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```

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 chang_to_ascii(int codein)

```

```

{
    char d;
    if((codein>48)&&(codein<127))
    switch(codein)
    {
        case 48:    d = '0';        break;

```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า
ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```

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;

```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า
ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```

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;

```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า
ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```

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;
}
/*-----*/
/*-----*/
/*-----*/

```

โปรแกรม P.c

```
#include<call_alp.h>

void main(void)
{
    /*      int ports,data;
    printf("\nPort number ");
    scanf("%d",&ports);
    printf("\nData Control ");
    scanf("%d",&data);
    printf("\nPort = %d \nData = %d\n",ports,data);

    */      outportb(0x3f0,1);
}
```



โปรแกรม puta.c

```
#include<call_alp.h>
```

```
void main(void)
```

```
{
```

```
    int i,j;
```

```
    int OFFSET=0;
```

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

```
    for(i=0;i<72;i++)
```

```
    {
```

```
        pokeb(SEGMENT,OFFSET,85);
```

```
        OFFSET+=2;
```

```
    }
```

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

```
        pokeb(SEGMENT,OFFSET,62);OFFSET+=2;
```

```
        pokeb(SEGMENT,OFFSET,75);OFFSET+=2;
```

```
        pokeb(SEGMENT,OFFSET,168);OFFSET+=2;
```

```
        pokeb(SEGMENT,OFFSET,27);OFFSET+=2;
```

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

```
    for(i=0;i<6;i++)
```

```
    {
```

```
        pokeb(SEGMENT,OFFSET,94);OFFSET+=2;
```

```
        pokeb(SEGMENT,OFFSET,145);OFFSET+=2;
```

```
        pokeb(SEGMENT,OFFSET,131);OFFSET+=2;
```

```
        pokeb(SEGMENT,OFFSET,233);OFFSET+=2;
```

```
    }
```

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

```
        pokeb(SEGMENT,OFFSET,128);OFFSET+=2;
```

```
        pokeb(SEGMENT,OFFSET,48);OFFSET+=2;
```

```
        pokeb(SEGMENT,OFFSET,163);OFFSET+=2;
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า
ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
pokeb(SEGMENT,OFFSET,4);OFFSET+=2;
```

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

```
/*....WORD.1....*/
```

```
pokeb(SEGMENT,OFFSET,131);OFFSET+=2;
```

```
pokeb(SEGMENT,OFFSET,193);OFFSET+=2;
```

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

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

```
/*....WORD.2....*/
```

```
pokeb(SEGMENT,OFFSET,7);OFFSET+=2;
```

```
pokeb(SEGMENT,OFFSET,131);OFFSET+=2;
```

```
pokeb(SEGMENT,OFFSET,1);OFFSET+=2;
```

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

```
/*....WORD.3....*/
```

```
pokeb(SEGMENT,OFFSET,13);OFFSET+=2;
```

```
pokeb(SEGMENT,OFFSET,6);OFFSET+=2;
```

```
pokeb(SEGMENT,OFFSET,3);OFFSET+=2;
```

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

```
/*....WORD.4....*/
```

```
pokeb(SEGMENT,OFFSET,25);OFFSET+=2;
```

```
pokeb(SEGMENT,OFFSET,12);OFFSET+=2;
```

```
pokeb(SEGMENT,OFFSET,6);OFFSET+=2;
```

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

```
/*....WORD.5....*/
```

```
pokeb(SEGMENT,OFFSET,49);OFFSET+=2;
```

```
pokeb(SEGMENT,OFFSET,24);OFFSET+=2;
```

```
pokeb(SEGMENT,OFFSET,12);OFFSET+=2;
```

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

```
/*....WORD.6....*/
```

```
pokeb(SEGMENT,OFFSET,97);OFFSET+=2;
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
pokeb(SEGMENT,OFFSET,48);OFFSET+=2;
pokeb(SEGMENT,OFFSET,24);OFFSET+=2;
pokeb(SEGMENT,OFFSET,0);OFFSET+=2;
```

```
/*....WORD.7....*/
```

```
pokeb(SEGMENT,OFFSET,193);OFFSET+=2;
pokeb(SEGMENT,OFFSET,96);OFFSET+=2;
pokeb(SEGMENT,OFFSET,16);OFFSET+=2;
pokeb(SEGMENT,OFFSET,0);OFFSET+=2;
```

```
/*....WORD.1....*/
```

```
pokeb(SEGMENT,OFFSET,131);OFFSET+=2;
pokeb(SEGMENT,OFFSET,193);OFFSET+=2;
pokeb(SEGMENT,OFFSET,0);OFFSET+=2;
pokeb(SEGMENT,OFFSET,0);OFFSET+=2;
```

```
/*....WORD.2....*/
```

```
pokeb(SEGMENT,OFFSET,7);OFFSET+=2;
pokeb(SEGMENT,OFFSET,131);OFFSET+=2;
pokeb(SEGMENT,OFFSET,1);OFFSET+=2;
pokeb(SEGMENT,OFFSET,0);OFFSET+=2;
```

```
printf("\nPut SYN");
pokeb(SEGMENT,OFFSET,62);OFFSET+=2;
pokeb(SEGMENT,OFFSET,75);OFFSET+=2;
pokeb(SEGMENT,OFFSET,168);OFFSET+=2;
pokeb(SEGMENT,OFFSET,27);OFFSET+=2;
```

```
/*....WORD.3....*/
```

```
pokeb(SEGMENT,OFFSET,13);OFFSET+=2;
pokeb(SEGMENT,OFFSET,6);OFFSET+=2;
pokeb(SEGMENT,OFFSET,3);OFFSET+=2;
pokeb(SEGMENT,OFFSET,0);OFFSET+=2;
```

```
/*....WORD.4....*/
```

```
pokeb(SEGMENT,OFFSET,25);OFFSET+=2;
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
pokeb(SEGMENT,OFFSET,12);OFFSET+=2;
pokeb(SEGMENT,OFFSET,6);OFFSET+=2;
pokeb(SEGMENT,OFFSET,0);OFFSET+=2;
```

```
/*....WORD.5....*/
```

```
pokeb(SEGMENT,OFFSET,49);OFFSET+=2;
pokeb(SEGMENT,OFFSET,24);OFFSET+=2;
pokeb(SEGMENT,OFFSET,12);OFFSET+=2;
pokeb(SEGMENT,OFFSET,0);OFFSET+=2;
```

```
/*....WORD.6....*/
```

```
pokeb(SEGMENT,OFFSET,97);OFFSET+=2;
pokeb(SEGMENT,OFFSET,48);OFFSET+=2;
pokeb(SEGMENT,OFFSET,24);OFFSET+=2;
pokeb(SEGMENT,OFFSET,0);OFFSET+=2;
```

```
/*....WORD.7....*/
```

```
pokeb(SEGMENT,OFFSET,193);OFFSET+=2;
pokeb(SEGMENT,OFFSET,96);OFFSET+=2;
pokeb(SEGMENT,OFFSET,16);OFFSET+=2;
pokeb(SEGMENT,OFFSET,0);OFFSET+=2;
```

```
/*....WORD.1....*/
```

```
pokeb(SEGMENT,OFFSET,131);OFFSET+=2;
pokeb(SEGMENT,OFFSET,193);OFFSET+=2;
pokeb(SEGMENT,OFFSET,0);OFFSET+=2;
pokeb(SEGMENT,OFFSET,0);OFFSET+=2;
```

```
/*....WORD.2....*/
```

```
pokeb(SEGMENT,OFFSET,7);OFFSET+=2;
pokeb(SEGMENT,OFFSET,131);OFFSET+=2;
pokeb(SEGMENT,OFFSET,1);OFFSET+=2;
pokeb(SEGMENT,OFFSET,0);OFFSET+=2;
```

```
/*....WORD.3....*/
```

```
pokeb(SEGMENT,OFFSET,13);OFFSET+=2;
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า
ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

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

```
/*....WORD.4....*/
```

```
pokeb(SEGMENT,OFFSET,25);OFFSET+=2;
pokeb(SEGMENT,OFFSET,12);OFFSET+=2;
pokeb(SEGMENT,OFFSET,6);OFFSET+=2;
pokeb(SEGMENT,OFFSET,0);OFFSET+=2;
```

```
/*....WORD.5....*/
```

```
pokeb(SEGMENT,OFFSET,49);OFFSET+=2;
pokeb(SEGMENT,OFFSET,24);OFFSET+=2;
pokeb(SEGMENT,OFFSET,12);OFFSET+=2;
pokeb(SEGMENT,OFFSET,0);OFFSET+=2;
```

```
/*....WORD.6....*/
```

```
pokeb(SEGMENT,OFFSET,97);OFFSET+=2;
pokeb(SEGMENT,OFFSET,48);OFFSET+=2;
pokeb(SEGMENT,OFFSET,24);OFFSET+=2;
pokeb(SEGMENT,OFFSET,0);OFFSET+=2;
```

```
/*....WORD.7....*/
```

```
pokeb(SEGMENT,OFFSET,193);OFFSET+=2;
pokeb(SEGMENT,OFFSET,96);OFFSET+=2;
pokeb(SEGMENT,OFFSET,16);OFFSET+=2;
pokeb(SEGMENT,OFFSET,0);OFFSET+=2;
```

```
/*....WORD.1....*/
```

```
pokeb(SEGMENT,OFFSET,131);OFFSET+=2;
pokeb(SEGMENT,OFFSET,193);OFFSET+=2;
pokeb(SEGMENT,OFFSET,0);OFFSET+=2;
pokeb(SEGMENT,OFFSET,0);OFFSET+=2;
```

```
/*....WORD.2....*/
```

```
pokeb(SEGMENT,OFFSET,7);OFFSET+=2;
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า
ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
pokeb(SEGMENT,OFFSET,131);OFFSET+=2;
pokeb(SEGMENT,OFFSET,1);OFFSET+=2;
pokeb(SEGMENT,OFFSET,0);OFFSET+=2;
```

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

```
for(i=0;i<2;i++)
```

```
{
```

```
    pokeb(SEGMENT,OFFSET,94);OFFSET+=2;
    pokeb(SEGMENT,OFFSET,145);OFFSET+=2;
    pokeb(SEGMENT,OFFSET,131);OFFSET+=2;
    pokeb(SEGMENT,OFFSET,233);OFFSET+=2;
```

```
}
```

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

```
for(i=0;i<4;i++)
```

```
{
```

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

```
}
```

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

```
}
```



โปรแกรม putb.c

```
include<call_alp.h>
```

```
void main(void)
```

```
int i,j;
```

```
int OFFSET=0;
```

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

```
for(i=0;i<72;i++)
```

```
{
```

```
    pokeb(SEGMENT,OFFSET,85);
```

```
    OFFSET+=2;
```

```
}
```

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

```
    pokeb(SEGMENT,OFFSET,62);OFFSET+=2;
```

```
    pokeb(SEGMENT,OFFSET,75);OFFSET+=2;
```

```
    pokeb(SEGMENT,OFFSET,168);OFFSET+=2;
```

```
    pokeb(SEGMENT,OFFSET,27);OFFSET+=2;
```

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

```
for(i=0;i<6;i++)
```

```
{
```

```
    pokeb(SEGMENT,OFFSET,94);OFFSET+=2;
```

```
    pokeb(SEGMENT,OFFSET,145);OFFSET+=2;
```

```
    pokeb(SEGMENT,OFFSET,131);OFFSET+=2;
```

```
    pokeb(SEGMENT,OFFSET,233);OFFSET+=2;
```

```
}
```

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

```
    pokeb(SEGMENT,OFFSET,128);OFFSET+=2;
```

```
    pokeb(SEGMENT,OFFSET,48);OFFSET+=2;
```

```
    pokeb(SEGMENT,OFFSET,163);OFFSET+=2;
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า
ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
pokeb(SEGMENT,OFFSET,4);OFFSET+=2;
```

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

```
/*....WORD.1....*/
```

```
pokeb(SEGMENT,OFFSET,133);OFFSET+=2;
```

```
pokeb(SEGMENT,OFFSET,66);OFFSET+=2;
```

```
pokeb(SEGMENT,OFFSET,1);OFFSET+=2;
```

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

```
/*....WORD.2....*/
```

```
pokeb(SEGMENT,OFFSET,11);OFFSET+=2;
```

```
pokeb(SEGMENT,OFFSET,133);OFFSET+=2;
```

```
pokeb(SEGMENT,OFFSET,2);OFFSET+=2;
```

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

```
/*....WORD.3....*/
```

```
pokeb(SEGMENT,OFFSET,21);OFFSET+=2;
```

```
pokeb(SEGMENT,OFFSET,10);OFFSET+=2;
```

```
pokeb(SEGMENT,OFFSET,5);OFFSET+=2;
```

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

```
/*....WORD.4....*/
```

```
pokeb(SEGMENT,OFFSET,41);OFFSET+=2;
```

```
pokeb(SEGMENT,OFFSET,20);OFFSET+=2;
```

```
pokeb(SEGMENT,OFFSET,10);OFFSET+=2;
```

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

```
/*....WORD.5....*/
```

```
pokeb(SEGMENT,OFFSET,81);OFFSET+=2;
```

```
pokeb(SEGMENT,OFFSET,40);OFFSET+=2;
```

```
pokeb(SEGMENT,OFFSET,20);OFFSET+=2;
```

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

```
/*....WORD.6....*/
```

```
pokeb(SEGMENT,OFFSET,161);OFFSET+=2;
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า
ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
pokeb(SEGMENT,OFFSET,80);OFFSET+=2;
pokeb(SEGMENT,OFFSET,8);OFFSET+=2;
pokeb(SEGMENT,OFFSET,0);OFFSET+=2;
```

```
/*....WORD.7....*/
```

```
pokeb(SEGMENT,OFFSET,67);OFFSET+=2;
pokeb(SEGMENT,OFFSET,161);OFFSET+=2;
pokeb(SEGMENT,OFFSET,16);OFFSET+=2;
pokeb(SEGMENT,OFFSET,0);OFFSET+=2;
```

```
/*....WORD.1....*/
```

```
pokeb(SEGMENT,OFFSET,133);OFFSET+=2;
pokeb(SEGMENT,OFFSET,66);OFFSET+=2;
pokeb(SEGMENT,OFFSET,1);OFFSET+=2;
pokeb(SEGMENT,OFFSET,0);OFFSET+=2;
```

```
/*....WORD.2....*/
```

```
pokeb(SEGMENT,OFFSET,11);OFFSET+=2;
pokeb(SEGMENT,OFFSET,133);OFFSET+=2;
pokeb(SEGMENT,OFFSET,2);OFFSET+=2;
pokeb(SEGMENT,OFFSET,0);OFFSET+=2;
```

```
printf("\nPut SYN");
pokeb(SEGMENT,OFFSET,62);OFFSET+=2;
pokeb(SEGMENT,OFFSET,75);OFFSET+=2;
pokeb(SEGMENT,OFFSET,168);OFFSET+=2;
pokeb(SEGMENT,OFFSET,27);OFFSET+=2;
```

```
/*....WORD.3....*/
```

```
pokeb(SEGMENT,OFFSET,21);OFFSET+=2;
pokeb(SEGMENT,OFFSET,10);OFFSET+=2;
pokeb(SEGMENT,OFFSET,5);OFFSET+=2;
pokeb(SEGMENT,OFFSET,0);OFFSET+=2;
```

```
/*....WORD.4....*/
```

```
pokeb(SEGMENT,OFFSET,41);OFFSET+=2;
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
pokeb(SEGMENT,OFFSET,20);OFFSET+=2;
pokeb(SEGMENT,OFFSET,10);OFFSET+=2;
pokeb(SEGMENT,OFFSET,0);OFFSET+=2;
```

```
/*....WORD.5....*/
```

```
pokeb(SEGMENT,OFFSET,81);OFFSET+=2;
pokeb(SEGMENT,OFFSET,40);OFFSET+=2;
pokeb(SEGMENT,OFFSET,20);OFFSET+=2;
pokeb(SEGMENT,OFFSET,0);OFFSET+=2;
```

```
/*....WORD.6....*/
```

```
pokeb(SEGMENT,OFFSET,161);OFFSET+=2;
pokeb(SEGMENT,OFFSET,80);OFFSET+=2;
pokeb(SEGMENT,OFFSET,8);OFFSET+=2;
pokeb(SEGMENT,OFFSET,0);OFFSET+=2;
```

```
/*....WORD.7....*/
```

```
pokeb(SEGMENT,OFFSET,67);OFFSET+=2;
pokeb(SEGMENT,OFFSET,161);OFFSET+=2;
pokeb(SEGMENT,OFFSET,16);OFFSET+=2;
pokeb(SEGMENT,OFFSET,0);OFFSET+=2;
```

```
/*....WORD.1....*/
```

```
pokeb(SEGMENT,OFFSET,133);OFFSET+=2;
pokeb(SEGMENT,OFFSET,66);OFFSET+=2;
pokeb(SEGMENT,OFFSET,1);OFFSET+=2;
pokeb(SEGMENT,OFFSET,0);OFFSET+=2;
```

```
/*....WORD.2....*/
```

```
pokeb(SEGMENT,OFFSET,11);OFFSET+=2;
pokeb(SEGMENT,OFFSET,133);OFFSET+=2;
pokeb(SEGMENT,OFFSET,2);OFFSET+=2;
pokeb(SEGMENT,OFFSET,0);OFFSET+=2;
```

```
/*....WORD.3....*/
```

```
pokeb(SEGMENT,OFFSET,21);OFFSET+=2;
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า
ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
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 CLREAR");
```

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

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

printf("\nPut CLREAR");
for(i=0;i<4;i++)
{
    pokeb(SEGMENT,OFFSET,0);OFFSET+=2;
}
printf("\nEnd");

```

โปรแกรม putdata.c

```
#include <call_alp.h>
```

```
void main(void)
```

```
{
    int i;
    char bytes,fname[11];
    unsigned OFFSET=0;
    FILE *fp;
    printf("\n Read file name ");
    gets(fname);
    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,OFFSET,85);
        OFFSET+=2;
    }
    printf("\nBegin Work");
    pokeb(SEGMENT,OFFSET,62);OFFSET+=2;
    pokeb(SEGMENT,OFFSET,75);OFFSET+=2;
    pokeb(SEGMENT,OFFSET,168);OFFSET+=2;
    pokeb(SEGMENT,OFFSET,27);OFFSET+=2;
    printf("\nBegin Work");
    for(i=76;i<204;i++)
    {
        bytes=fgetc(fp);
        pokeb(SEGMENT,OFFSET,bytes);
        OFFSET+=2;
    }
    fclose(fp);
    printf("\n Compleat");
}
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า
ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

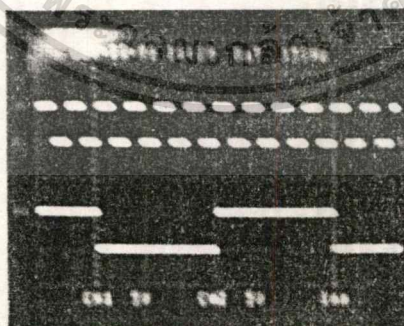
บทที่ 5

สรุปผลการทดลอง

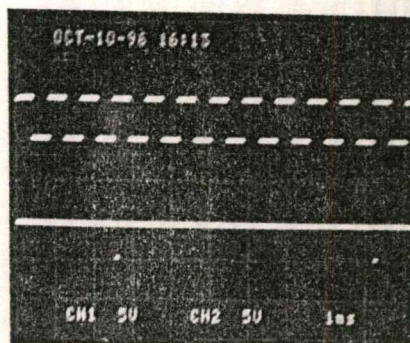
ในบทนี้จะกล่าวถึงการทดลองรับ-ส่งเพจเจอร์ โดยในการทำการส่งข้อมูลจะใช้เครื่องสอบวิทยุ (RF Communication Test Set) เป็นตัวส่งข้อมูลให้ Field Strength Meter เป็นตัวรับความถี่ของสัญญาณส่งผ่านไปยัง Pager Decoder Card ถอดรหัสสัญญาณแล้วเก็บเป็น File เอาไว้ แล้วทำการเขียนข้อมูลจาก File ที่เก็บไว้ลงในแรมของ Pager Encoder Card เสร็จแล้วจึงสั่งให้ Pager Encoder Card ส่งข้อมูล โดยในการทดลองนี้เราใช้วิธีต่อสัญญาณจาก Pager Encoder Card ไปยัง Pager Decoder Card โดยตรงผ่านสายตัวนำ

ในการทดลองจึงได้วัดสัญญาณเอาต์พุตจากภาคต่างๆของวงจรที่ใช้งาน เช่น ภาคอ้างอิง สัญญาณนาฬิกา โดยจะนำสัญญาณเหล่านี้มาเปรียบเทียบ เพื่อทำการวิเคราะห์, แก้ไขและทดสอบ โดยให้แสดงผลออกมาทางหน้าจอคอมพิวเตอร์ว่าตรงตามข้อมูลที่ได้อัปโหลดเข้าไปหรือไม่

จากรูปที่ 5.1 และ 5.2 เป็นรูปสัญญาณที่วัดได้จากการทดลองของภาคอ้างอิงสัญญาณนาฬิกา โดยในรูปที่ 5.1 เป็นรูปสัญญาณนาฬิกา 1200 Hz กับ 150 Hz เปรียบเทียบกันโดยจะนำสัญญาณทั้งสองนี้ไปอ้างอิงให้กับวงจรต่างๆ ส่วนในรูปที่ 5.2 เป็นรูปเปรียบเทียบระหว่างสัญญาณนาฬิกา 1200 Hz กับสัญญาณ LOAD ซึ่งจะเห็นได้ว่าสัญญาณ LOAD จะเกิดขึ้นในทุกๆ 8 พัลส์ของ 1200 Hz เสมอ

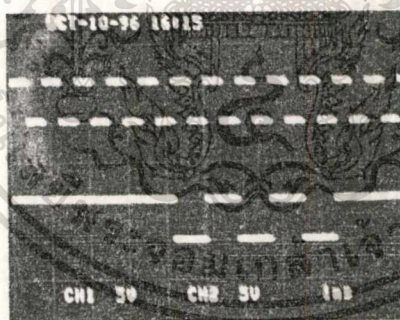


รูปที่ 5.1 เป็นรูปสัญญาณนาฬิกา 1200 Hz กับ 150 Hz



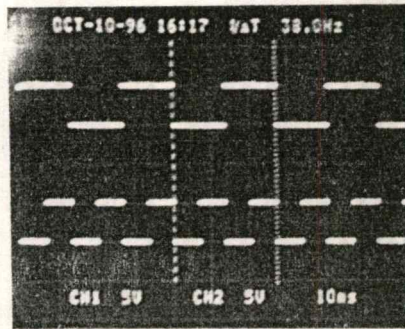
รูปที่ 5.2 เป็นรูปเปรียบเทียบระหว่างสัญญาณนาฬิกา 1200 Hz กับสัญญาณ LOAD

จากรูปที่ 5.3 เป็นรูปสัญญาณที่วัดได้จากการทดลองของภาค PISO โดยได้เปรียบเทียบกันระหว่างสัญญาณนาฬิกา 1200 Hz กับสัญญาณเอาต์พุตของภาค PISO โดยเราไม่สามารถจะสังเกตได้ว่าสัญญาณที่ออกมา (ของภาค PISO) นั้นถูกต้องหรือไม่ ต้องไปสังเกตดูจากจอคอมพิวเตอร์เท่านั้น



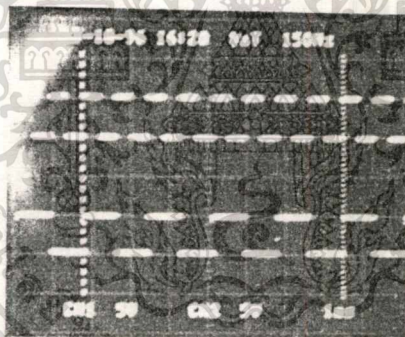
รูปที่ 5.3 เป็นรูปสัญญาณนาฬิกา 1200 Hz กับสัญญาณเอาต์พุตของภาค PISO

จากรูปที่ 5.4 เป็นรูปสัญญาณที่วัดได้จากการทดลองของภาค Address Counter โดยได้เปรียบเทียบกันระหว่างสัญญาณของ A_0 กับสัญญาณของ A_1 โดยสามารถดูได้ว่า A_0 กับ A_1 นั้นจะต่างกันอยู่โดยใช้ 2 ฮารสัญญาณลง โดยที่เอา A_0 เป็นตัวตั้ง

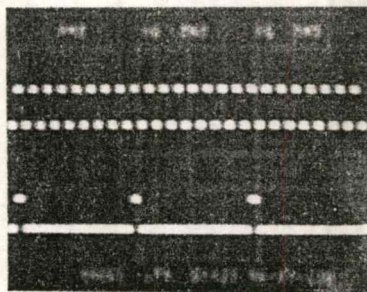


รูปที่ 5.4 เปรียบเทียบกันระหว่างสัญญาณของ A_0 กับสัญญาณของ A_1

จากรูปที่ 5.5 เป็นรูปสัญญาณที่วัดได้จากการทดสอบของภาค Limiter โดยนำสัญญาณ 1200 Hz มาเปรียบเทียบกับสัญญาณ Preamble Pulse ที่ผ่าน Limiter แล้ว



รูปที่ 5.5 สัญญาณ 1200 Hz เปรียบเทียบกับสัญญาณ Preamble Pulse ที่ผ่าน Limiter

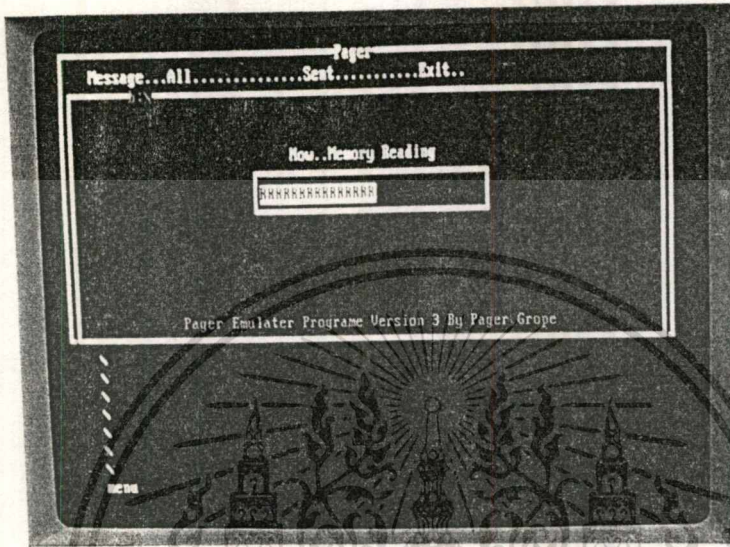


รูปที่ 5.6 สัญญาณ 1200 Hz เปรียบเทียบกับสัญญาณอ่านแรม

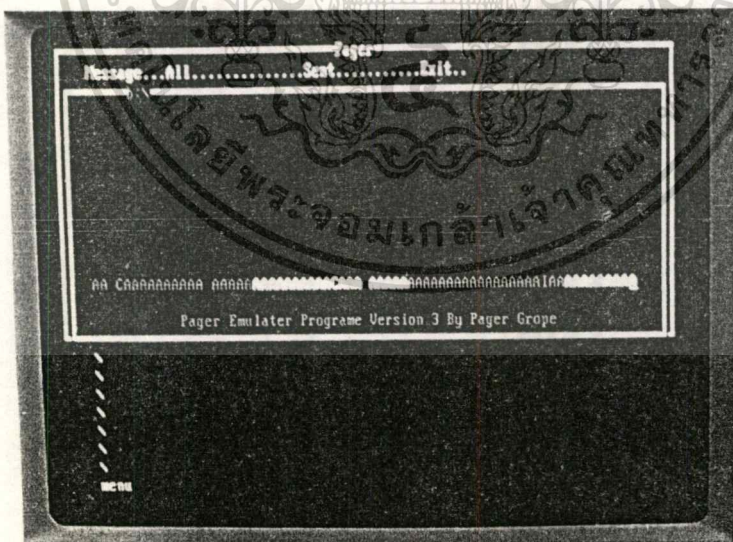
เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

จากรูปที่ 5.6 เป็นรูปสัญญาณที่วัดได้จากการทดลองของภาค RAM Emulator โดยนำสัญญาณ 1200 Hz มาเปรียบเทียบกับสัญญาณอ่านแรม

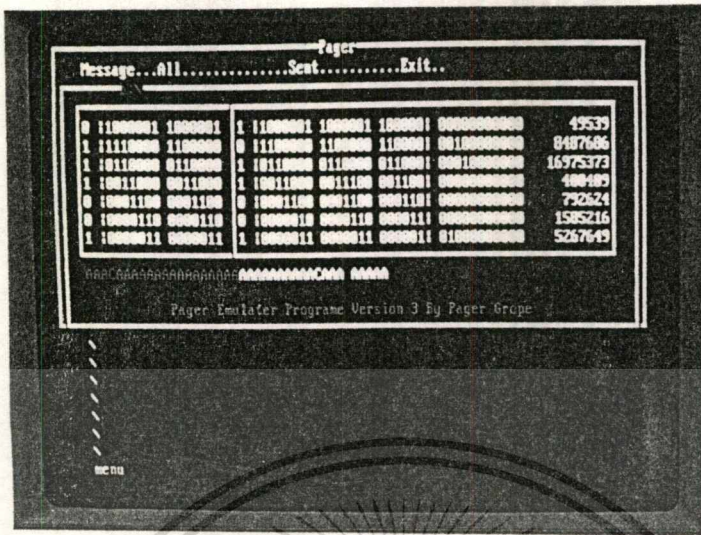
จากรูปที่ 5.7 - 5.9 เป็นรูปของการแสดงผลทางจอคอมพิวเตอร์



รูปที่ 5.7 แสดงให้เห็นว่ากำลังอ่านข้อมูลจากแรมอยู่

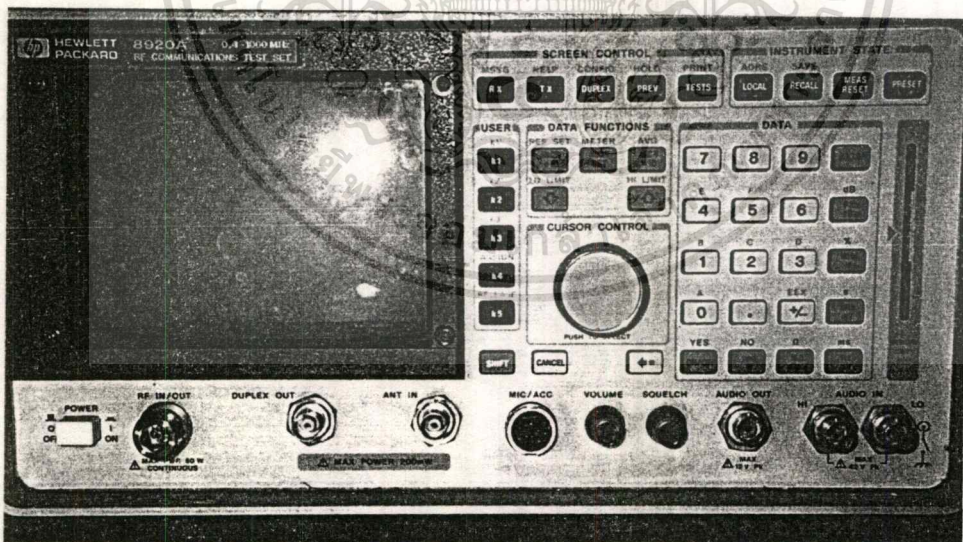


รูปที่ 5.8 แสดงผลของข้อมูลที่ได้รับได้



รูปที่ 5.9 แสดงผลของข้อมูลที่ได้รับได้เป็นแบบตัวอักษรและ Binary

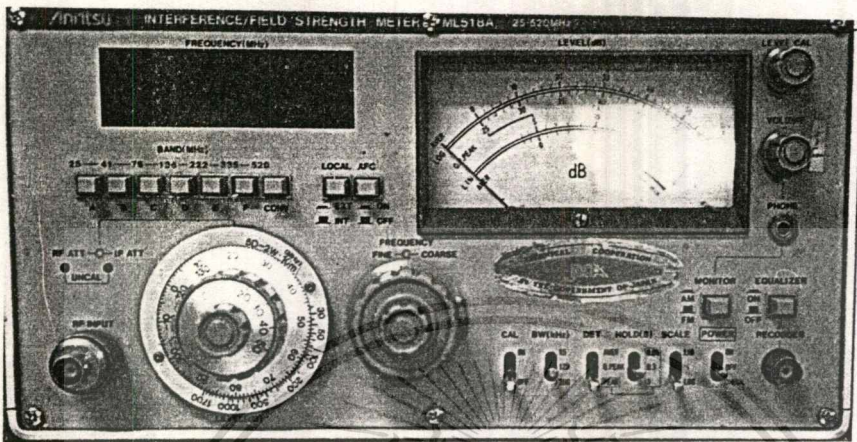
ในการทดลองนั้นได้ทดสอบทั้งการส่งข้อมูลจากเครื่องทดสอบความถี่วิทยุสื่อสาร (RF Communication Test Set) และจากการเขียนโปรแกรม



รูปที่ 5.10 เครื่องทดสอบความถี่วิทยุสื่อสาร (RF Communication Test Set)

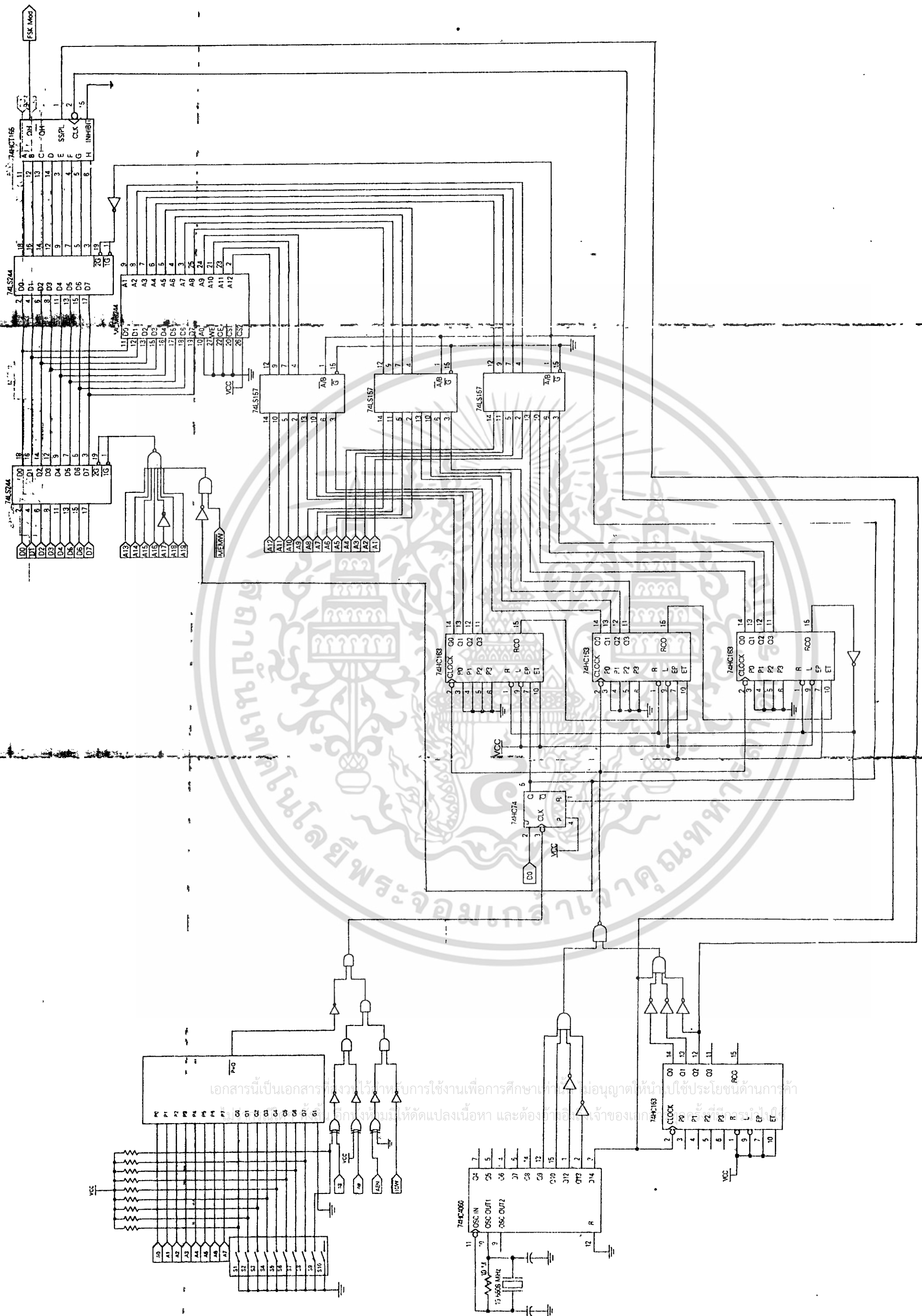
เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า
ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

ในการทดลองนี้ไม่ได้สร้างเครื่องรับขึ้นมาใช้งาน จึงใช้ Field Strength Meter เป็นเครื่องรับเพื่อทดลองในการรับข้อมูลจากภายนอก



รูปที่ 5.11 เครื่องรับ (Field Strength Meter)





เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า
 การใช้งานอื่นที่ผิดัดแปลงเนื้อหา และต้องขออนุญาตจากเจ้าของเอกสารไว้ก่อน



เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า
ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้



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.

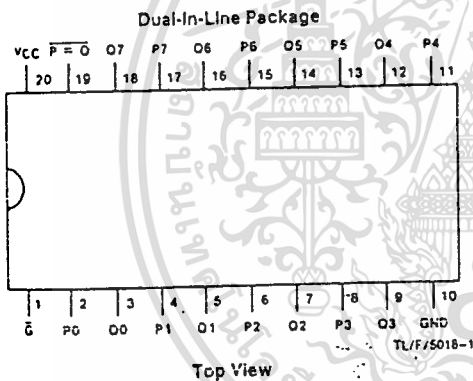
The comparator's output can drive 10 low power Schottky equivalent loads. This comparator is functionally and pin

compatible to the 54LS688/74LS688. All inputs are protected from damage due to static discharge by diodes to V_{CC} and ground.

Features

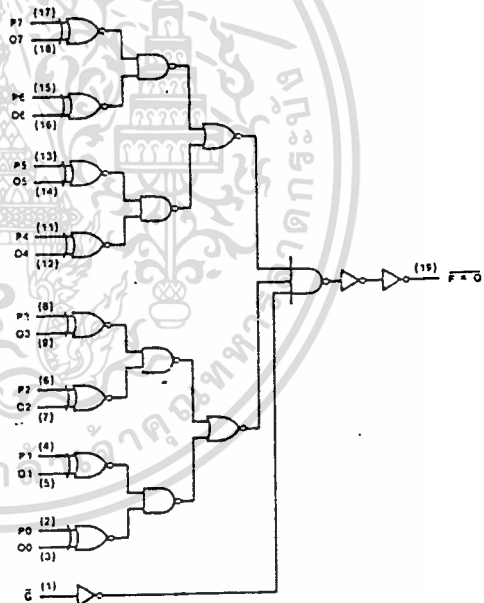
- Typical propagation delay: 20 ns
- Wide power supply range: 2-6V
- Low quiescent current: 80 μ A (74 Series)
- Large output current: 4 mA (74 Series)
- Same as 'HC521

Connection and Logic Diagrams



Order Number MM54HC688* or MM74HC688*

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



TL/F/5015-2

Truth Table

Inputs		$\overline{P=Q}$
Data	Enable	
P,Q	$\overline{G}$	
$P=Q$	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

Supply Voltage (V_{CC})	Min	Max
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	54HC
						T _A = -40 to 85°C	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} ≤ 20 μA	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} ≤ 4.0 mA I _{OUT} ≤ 5.2 mA	4.5V	4.2	3.98	3.84	3.7
			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} ≤ 20 μA	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} ≤ 4.0 mA I _{OUT} ≤ 5.2 mA	4.5V	0.2	0.26	0.33	0.4
			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 125°C.

Note 4: For a power supply of $5V \pm 10\%$ 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 operating 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_{IK}, I_{OK}) 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

$V_{CC} = 5V$, $T_A = 25^\circ C$, $C_L = 15\text{ pF}$, $t_r = t_f = 6\text{ ns}$

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

Electrical Characteristics

$2.0V$ to $6.0V$, $C_L = 50\text{ pF}$, $t_r = t_f = 6\text{ ns}$ (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 _{PLH}	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

*1 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, $I_S = C_{PD} V_{CC} f + I_{CC}$.



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.

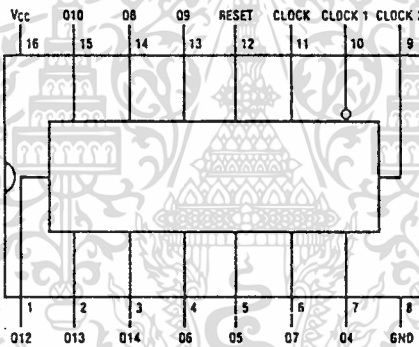
This device is pin equivalent to the CD4060. All inputs are protected from damage due to static discharge by protection diodes to V_{CC} and ground.

Features

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

Connection and Logic Diagrams

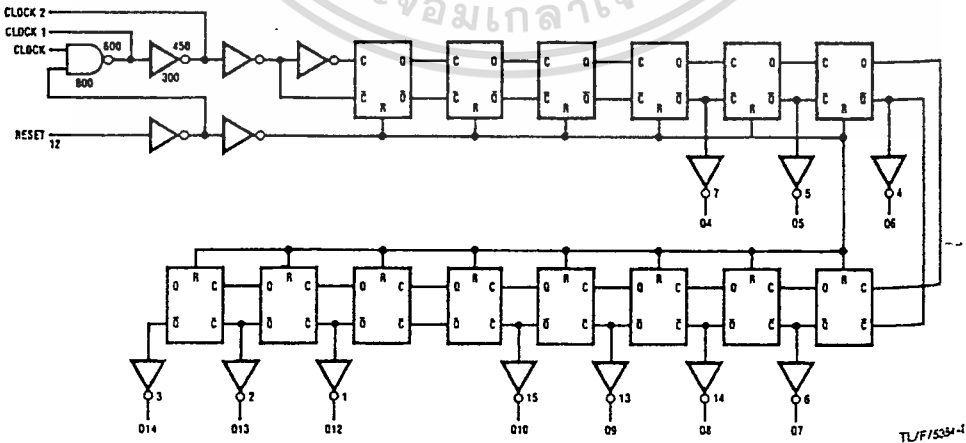
Dual-in-Line Package



Top View

Order Number MM54HC4060* or MM74HC4060*

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



TL/F/5354-1

Absolute Maximum Ratings (Notes 1 & 2)

Primary/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 $V_{CC} + 0.5V$
Input Diode Current (I_{CP})	± 20 mA
Output Current, per pin (I_{OUT})	± 25 mA
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
Package only	500 mW
Temperature (T_J)	
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

Electrical Characteristics (Note 4)

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				
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	
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	
Minimum High Level Output Voltage	V _{IN} = V _{IH} or V _{IL} I _{OUT} ≤ 20 μA	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} I _{OUT} ≤ 4.0 mA I _{OUT} ≤ 5.2 mA	4.5V	4.2	3.98	3.84	3.7	V	
		6.0V	5.7	5.48	5.34	5.2	V	
	Pins 9 & 10 V _{IN} = V _{IH} or V _{IL} I _{OUT} = 0.4 mA I _{OUT} = 0.52 mA			3.98	3.84	3.7	V	
				5.48	5.34	5.2	V	
	Maximum Low Level Output Voltage	V _{IN} = V _{IH} or V _{IL} I _{OUT} ≤ 20 μA	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} I _{OUT} ≤ 4.0 mA I _{OUT} ≤ 5.2 mA		4.5V	0.2	0.26	0.33	0.4	V	
		6.0V	0.2	0.26	0.33	0.4	V	
Pins 9 & 10 V _{IN} = V _{IH} or V _{IL} I _{OUT} = 0.4 mA I _{OUT} = 0.52 mA				0.26	0.33	0.4	V	
				0.26	0.33	0.4	V	
Maximum Input Current		V _{IN} = V _{CC} or GND	6.0V		± 0.1	± 1.0	± 1.0	μA
Maximum Quiescent Supply Current		V _{IN} = V _{CC} or GND I _{OUT} = 0 μA	6.0V		8.0	80	160	μA

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

2 Unless otherwise specified all voltages are referenced to ground.

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

4 For a power supply of 5V $\pm 10\%$ 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 working 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_{CC} and I_{OZ}) occur for CMOS at the higher voltage and so the 6.0V values should be used.

5 Devices 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.

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า
ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

AC Electrical Characteristics

$V_{CC} = 5V$, $T_A = 25^\circ C$, $C_L = 15\text{ pF}$, $t_r = t_f = 6\text{ ns}$

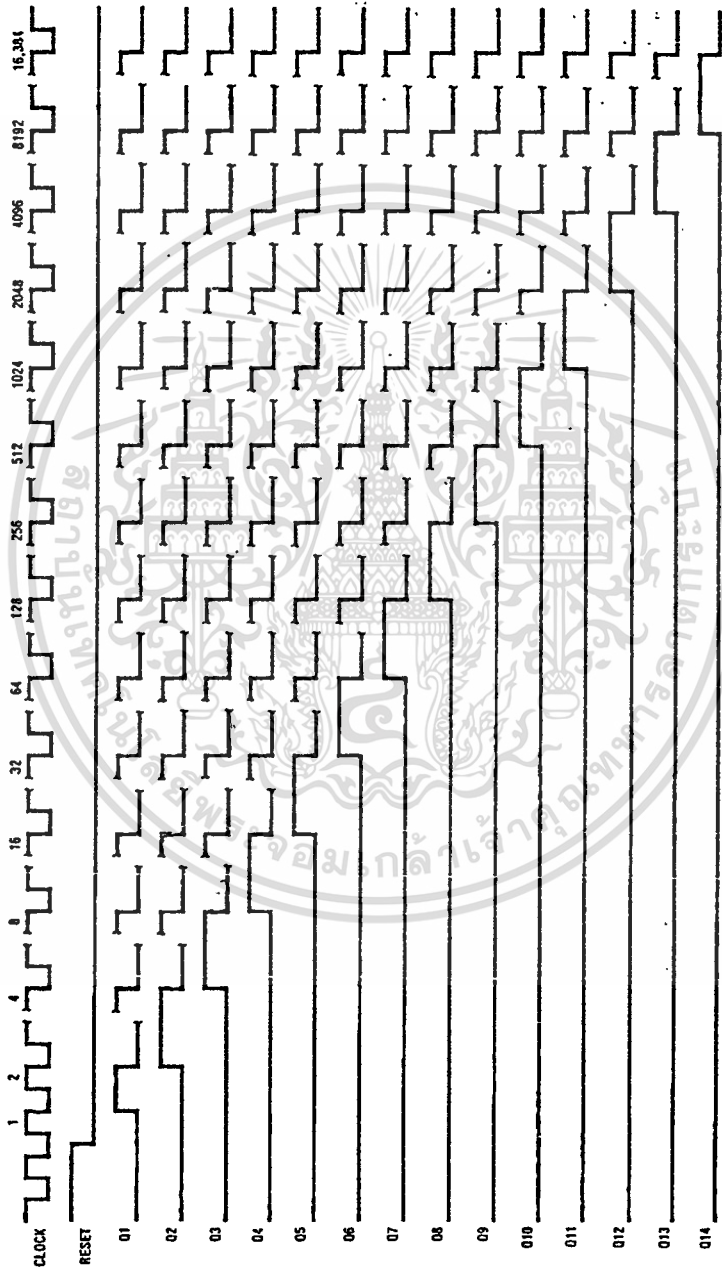
Symbol	Parameter	Conditions	Typ	Guaranteed Limit	Units
f_{MAX}	Maximum Clock Frequency			30	MHz
t_{PHL}, t_{PLH}	Maximum Propagation Delay to Q_4	(Note 5)	40	20	ns
t_{PHL}, t_{PLH}	Maximum Propagation Delay to any Q		16	40	ns
t_{REM}	Minimum Reset Removal Time		10	20	ns
t_W	Minimum Pulse Width		10	16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

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

Symbol	Parameter	Conditions	V _{CC}	T _A = 25°C		T _A 74HC	T _A 54HC
						T _A = -40 to 85°C	T _A = -55 to 125°C
				Typ	Guaranteed Limits		
f _{MAX}	Maximum Operating Frequency		2.0V		6	5	4
			4.5V		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}	Minimum Reset Removal Time		2.0V	100	125	150	
			4.5V	20	25	30	
			6.0V	17	21	25	
t _W	Minimum 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: $t_p = 17 + 12(N-1)\text{ ns}$; where N is the number of the output, Q_w , 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



TL/F/5354-3



MM54HCT688/MM74HCT688
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 indicate 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 combines the low power consumption of CMOS, but inputs are compatible with TTL logic levels, and the output can drive 10 low power Schottky equivalent loads.

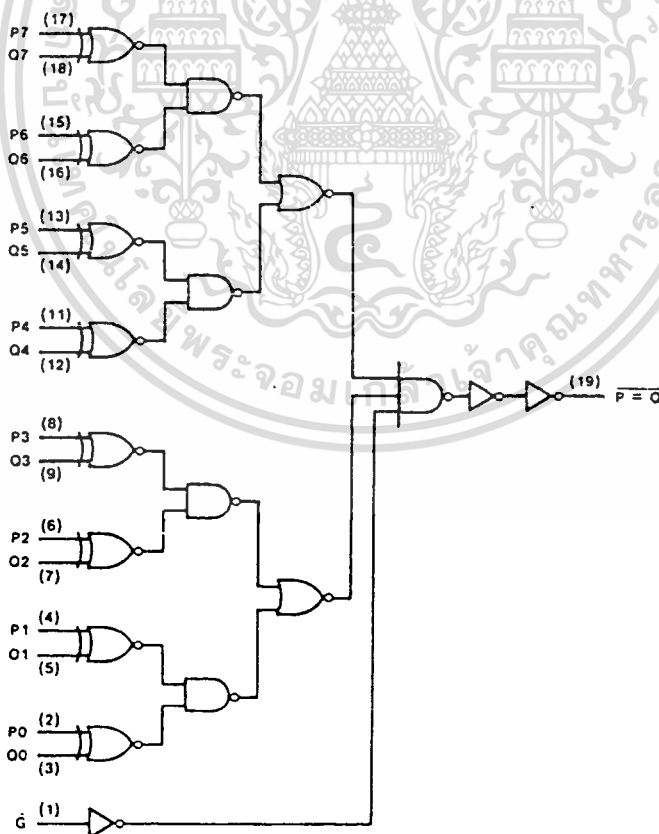
MM54HCT/MM74HCT devices are intended to interface between TTL and NMOS components and standard CMOS devices. These parts are also plug in replacements for LS-TTL devices and can be used to reduce power consumption in existing designs.

All inputs are protected from damage due to static dis-
charge by diodes to V_{CC} and ground.

Features

- TTL input compatible
- Typical propagation delay: 20 ns
- Low quiescent current: 80 μ A maximum (74HCT Series)
- Large output current: 4 mA
- Same as HCT521

Logic Diagrams



TI / F / 5371-2

Absolute Maximum Ratings (Notes 1 & 2)

For 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 $V_{CC} + 0.5V$
Diode Current (I_{IK}, I_{OK})	± 20 mA
Output Current, per pin (I_{OUT})	± 25 mA
Source or GND Current, per pin (I_{CC})	± 50 mA
Storage Temperature Range (T_{STG})	-65°C to +150°C
Power Dissipation (P_D)	
Package 3)	600 mW
Package only	500 mW
Temperature (T_J)	
Soldering 10 seconds)	260°C

Operating Conditions

Supply Voltage (V_{CC})	Min 4.5	Max 5.5	Units V
DC Input or Output Voltage (V_{IN}, V_{OUT})	0	V_{CC}	V
Operating Temp. Range (T_A)			
MM74HCT688	-40	+85	°C
MM54HCT688	-55	+125	°C
Input Rise or Fall Times (t_r, t_f)		500	ns

Electrical Characteristics ($V_{CC} = 5V \pm 10\%$ unless otherwise specified)

Symbol	Parameter	Conditions	$T_A = 25^{\circ}\text{C}$			Units
			74HCT $T_A = -40$ to 85°C	54HCT $T_A = -55$ to 125°C		
			Typ	Guaranteed Limits		
	Minimum High Level Input Voltage		2.0	2.0	2.0	V
	Maximum Low Level Input Voltage		0.8	0.8	0.8	V
	Minimum High Level Output Voltage $V_{IN} = 0.8\text{V}$ or 2.0V $ I_{OUT} = 20\text{ }\mu\text{A}$ $ I_{OUT} = 4.0\text{ mA}$, $V_{CC} = 4.5\text{V}$ $ I_{OUT} = 4.8\text{ mA}$, $V_{CC} = 5.5\text{V}$	V_{CC}	$V_{CC} - 0.1$	$V_{CC} - 0.1$	$V_{CC} - 0.1$	V
			4.2	3.98	3.84	V
			5.7	4.98	4.84	V
						V
	Maximum Low Level Voltage $V_{IN} = 0.8\text{V}$ or 2.0V $ I_{OUT} = 20\text{ }\mu\text{A}$ $ I_{OUT} = 4.0\text{ mA}$, $V_{CC} = 4.5\text{V}$ $ I_{OUT} = 4.8\text{ mA}$, $V_{CC} = 5.5\text{V}$	0	0.1	0.1	0.1	V
			0.2	0.26	0.33	V
			0.2	0.26	0.33	V
						V
	Maximum Input Current $V_{IN} = V_{CC}$ or GND		± 0.1	± 1.0	± 1.0	μA
	Maximum Quiescent Supply Current $V_{IN} = V_{CC}$ or GND $I_{OUT} = 0\text{ }\mu\text{A}$ $V_{IN} = 2.4\text{V}$ or 0.5V (Note 4)		80	80	160	μA
			0.3	0.4	0.5	mA

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

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

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.

4. Measured per pin. All other inputs held at V_{CC} or ground.

AC Electrical Characteristics

V_{CC} = 5V, T_A = 25°C, C_L = 15 pF, t_r = t_f = 6 ns (unless otherwise specified)

Symbol	Parameter	Conditions	Typ	Guaranteed Limit	Units
t _{PHL}	Maximum Propagation Delay - P or Q to Output		19	30	ns
t _{PLH}	Maximum Propagation Delay - P or Q to Output		13	22	ns
t _{PHL}	Maximum Propagation Delay - Enable to Output		13	20	ns
t _{PHL}	Maximum Propagation Delay - Enable to Output		10	18	ns

AC Electrical Characteristics

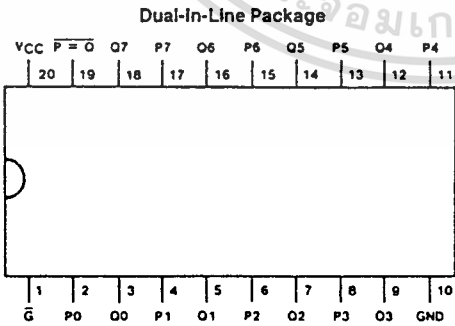
V_{CC} = 5V ± 10%, C_L = 50 pF, t_r = t_f = 6 ns (unless otherwise specified)

Symbol	Parameter	Conditions	T _A = 25°C		74HCT T _A = -40 to 85°C	54HCT T _A = -55 to 125°C	Units
			Typ	Guaranteed Limits			
t _{PHL}	Maximum Propagation Delay - P or Q to Output		23	35	44	53	ns
t _{PLH}	Maximum Propagation Delay - P or Q to Output		16	24	30	36	ns
t _{PHL}	Maximum Propagation Delay - Enable to Output		16	24	30	36	ns
t _{PLH}	Maximum Propagation Delay - Enable to Output		11	20	25	30	ns
t _{THL} , t _{TLH}	Maximum Output Rise and Fall Time		8	15	19	22	ns
C _{PD}	Power Dissipation Capacitance (Note 5)		45				pF
C _{IN}	Maximum Input Capacitance		5	10	10	10	pF

Note 5: 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} + I_{CC}.

Connection Diagram

Truth Table



Inputs		P=Q
Data P, Q	Enable G	
P=Q	L	L
P>Q	L	H
P<Q	L	H
X	H	H

Top View

Order Number MM54HCT688* or MM74HCT688*

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

TL/F/5371-1

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า
ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

บรรณานุกรม

- 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 ของ พงษ์ระพี เตชพาพงษ์
- 5.การอินเทอร์เฟส IBM/PC ของ ธาณินทร์ ถาวรศาสนวงศ์,ทินกร คุ้ม
- 6.การเขียนภาษาซีสำหรับวิศวกรรมของ ธันวา ศรีประโม่ง
- 7.การเขียนชุดคำสั่งภาษาซีของ มัณฑนา ปราการสมุทร

