โพรโทคอล X.25 / HDLC

X.25 / HDLC PROTOCOL

โดย

นาย มานิจ	หาดทวายกาญจน์	34131221
นาย สุทัศน์	พะสุรัมย์	34131238
นาย สุทิน	พิพัฒน์จำเริญกุล	34131239

ปริญญานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาอุตสาหกรรมศาสตรบัณฑิต

สาขาวิชาเทคโนโลยี อีเล็กทรอนิกส์

คณะวิศวกรรมศาสตร์

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหาร ลาดกระบัง

ปีการศึกษา 2536

ปริญญานิพนธ์ปีการศึกษา 2536

ภาควิชาเทคนิคอุตสาหกรรม

คณะวิศวกรรมศาสตร์ สถาบันเทคโนโลยีพระจอมเกล้า เจ้าคุณทหารลาดกระบัง

เรื่อง การสื่อสารข้อมูลทาง X.25 / HDLC PROTOCOL

ผู้จัดทำ

1. นาย มานิจ	หาดทวายกาญจน์	34131221
2. นาย สุทัศน์	พะสุรัมย์	34131238
3. นาย สุทิน	พิพัฒน์จำเริญกุล	34131239

คณะกรรมการสอบปริญญานิพนธ์

..... อาจารย์ที่ปรึกษา
( )

..... กรรมการ
( )

..... กรรมการ
( )

..... กรรมการ
( )

..... กรรมการ
( )

..... กรรมการ
( )

กิตติกรรมประกาศ

ปริญญานิพนธ์นี้สำเร็จลุล่วงได้ด้วยดี เนื่องจากได้รับความกรุณาจาก อาจารย์ กฤดากร กล่อมการ ซึ่งเป็นอาจารย์ที่ปรึกษาที่ได้ให้ความช่วยเหลือแนะนำในการทำปริญญานิพนธ์นี้

และคณะผู้จัดทำต้องขอขอบพระคุณอย่างสูง สำหรับเจ้าหน้าที่ ของกองสื่อสารข้อมูล (Data Communication Divtions) เจ้าหน้าที่ชุมสายสื่อสารข้อมูล (Thaipak) การสื่อสารแห่งประเทศไทย ที่ได้ให้ คำแนะนำ และทำการทดลองต่าง ๆ ตลอดมา

โพรโทคอล X.25 / HDLC

นาย มานิจ หาดทวายกาญจน์ 34131221
นาย สุทัศน์ พะสุรัมย์ 34131238
นาย สุทิน พิพัฒน์จำเริญกุล 34131239
อาจารย์ กฤดากร กล่อมการ
อาจารย์ที่ปรึกษา
ปีการศึกษา 2536

บทคัดย่อ

โครงงานวิจัยนี้มีจุดประสงค์ เพื่อศึกษาการทำงานของการสื่อสารข้อมูลแบบ PROTOCOL X.25/ HDLC โดยได้ดำเนินการดังนี้

1. ศึกษาเครือข่ายคอมพิวเตอร์

ระบบเครือข่ายคอมพิวเตอร์ แบ่งตามอัตราความเร็วในการส่งข้อมูล, ระยะทางและลักษณะการให้บริการออกเป็น 2 ประเภทใหญ่ ๆ คือ เครือข่ายเฉพาะบริเวณ (LOCAL AREA NETWORK : LAN) และเครือข่ายบริเวณกว้าง (WIDE AREA NETWORK : WAN) ซึ่งในโครงงานนี้จะ ศึกษาเฉพาะ WAN ที่ใช้ระบบมาตรฐาน X.25 / HDLC ตามข้อกำหนด ของ ISO

2. ศึกษา PROTOCOL HDLC

โพรโทคอล HDLC เป็นโพรโทคอลชั้นที่ 2 ที่ถูกกำหนดโดย ISO ซึ่งมีรูปแบบการเชื่อมต่อทางกายภาพ , หน้าที่การทำงาน, รูปแบบการส่งผ่านข้อมูล, รูปแบบของเฟรม, ลักษณะการส่งเฟรมโต้ตอบของโพรโทคอล HDLC

3. ศึกษา 8273 กับ PORT ที่ใช้ในการตรวจสอบโพรโทคอล HDLC /SDLC เป็นการศึกษาถึงส่วนประกอบต่าง ๆ ภายในของ CHIP 8273 , ซอฟต์แวร์สั่งการ , ซอฟต์แวร์สั่งงานปฏิบัติการ , ซอฟต์แวร์คำสั่งผลลัพท์และลักษณะของคำสั่งของ 8273

4. ออกแบบวงจรและประกอบวงจรลงปริ๊นท์ จำนวน 2 ชุด

5. เขียนโปรแกรมควบคุมการทำงานด้วยภาษาแอสแซมบลี้

6. ลำดับขั้นการทดลอง

- ใช้เครื่องคอมพิวเตอร์ 2 เครื่อง โดยนำวงจรที่สร้างเสร็จแล้ว ใส่เข้ากับเครื่องคอมพิวเตอร์ เครื่องละ 1 ชุด
- ต่อสายสัญญาณถึงกันระหว่าง คอมพิวเตอร์ทั้ง 2 เครื่อง
- ป้อนคำสั่งสำหรับควบคุมการทำงานของเครื่อง

จากผลการทดลองปรากฏว่าวงจรที่ได้ออกแบบ และนำมาทดลองนั้น สามารถติดต่อสื่อสารกันได้ แต่มีความเร็วในการรับ-ส่งข้อมูลต่ำ ยังไม่เหมาะสมที่จะนำมาใช้งานจริง ซึ่งต้องมีการปรับปรุง ให้มีความเร็วในการ รับ-ส่ง ข้อมูลได้สูงกว่านี้ ถึงจะมีความเหมาะสมในการนำมาใช้งานจริง

PROTOCOL X.25/HDLC

MR. MANICH HADTAVAYKAN

MR. SUTAT PASURAM

MR. SUTIN PIPATCHAMREONKUL

ADVISOR:

MR. KITDAKORN KLOMKARN

ACADEMIC YEAR 1993

ABSTRACT

The purpose of this project is to study the working of typical of Protocol X.25/HDLC data communications. The way of study are as follows:

1. To study computer network

The computer network system is divided into 2 categories according to the speed of data transmission, distance and type of service. They are:

a) LAN (Local Area Network)

b) WAN (Wide Area Network)

For this project WAN (WAN X.25/HDLC according to ISO standard) is studied.

2. To study Protocol

Protocol HDLC is second level specified by ISO and it connected by physical lavel and it working in data communications sending / receiving by a frame. The Chip 8273 is control that.

3. To study 8273 and Port while are used for checking Protocol HDLC/SDLC

That means the study of structure of the IC 8273 chip, instruction solfware, operation solfware, and 8273 instruction useage.

4. To design and compose 2 circuit boards.

5. To write the operation control program by using the Assembly Language.

6. The Experimental Step

- To install each of 2 circuit board mentioned above in each of 2 sets of computers.
- To connect the cables between two computer.
- To enter the operating instruction into the computers.

The result of this experiment is that the two circuit board can communicate each other. But it is not suitable for using them because of the low speed of receiving and transmitting data . It is successory to improve the speed to be high.

สารบัญ

บทที่ 1
เครือข่ายคอมพิวเตอร์

1.1 บทนำ

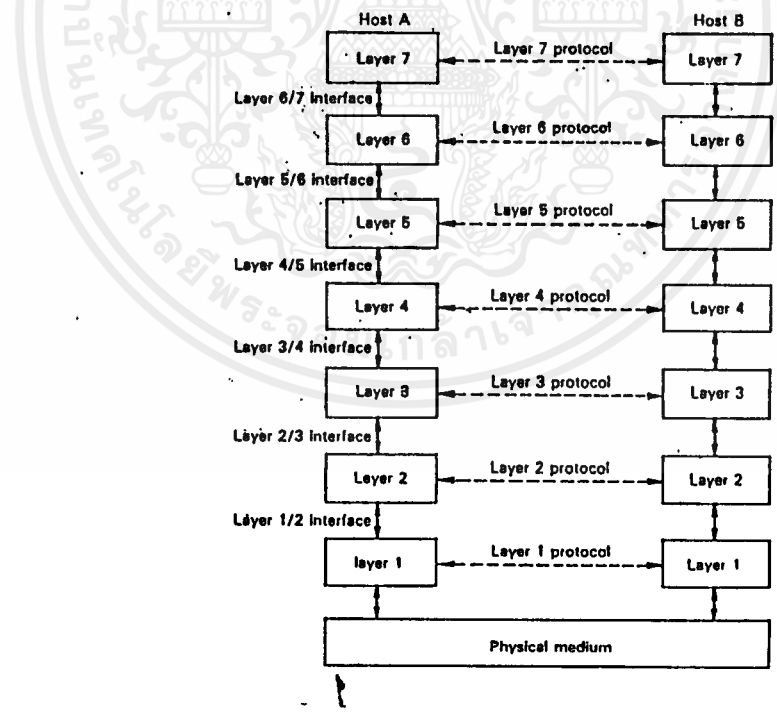
ระบบเครือข่าย เมื่อ พิจารณาตามอัตราความเร็วในการส่งผ่านข้อมูล ระยะทาง และลักษณะการให้บริการ จะแบ่งเป็น 2 ประเภทใหญ่ ๆ คือ เครือข่าย เฉพาะบริเวณ (Local Area Network : LAN) และ เครือข่ายบริเวณกว้าง (Wide Area Network : WAN) โดยเครือข่ายเฉพาะบริเวณ จะมี อัตราในการส่งผ่าน ข้อมูล ค่อนข้างสูง คือ ประมาณ 4 - 100 เมกะบิต / วินาที ระยะทางอยู่ในช่วงประมาณ 2.5 - 200 กิโลเมตรสำหรับการให้บริการนั้นผู้ที่เป็นเจ้าของระบบจะเป็นผู้กำหนด ซึ่งอาจจะเป็นเอกชนหรือรัฐบาลก็ได้ มาตรฐาน และเครือข่ายเฉพาะบริเวณ เช่น IEEE 802.3 , 802.4 , 802.5 และ FDDI (Fiber Distributed Data Interface) เป็นต้น เครือข่ายบริเวณกว้าง จะมีอัตราเร็วในการส่งข้อมูลเมื่อถึงผู้ใช้บริการไม่สูงมาก คือประมาณ 1,200 บิต / วินาที ถึง 2 เมกะบิต/วินาที (2.048 เมกะบิต/วินาที) ระยะทางที่ใกล้โดยเฉลี่ย 8 กิโลเมตรส่วนระยะทางไกล ก็อาจจะถึงหลายหมื่น กิโลเมตร สำหรับการให้บริการนั้น จะครอบคลุมภายในประเทศ และ ติดต่อกับต่างประเทศ จึงจำเป็นต้องมี องค์กรของรัฐ เข้ามาเกี่ยวข้องด้วยมาตรฐานของเครือข่ายบริเวณกว้าง เช่น X.25 , X.75 เป็นต้น สำหรับ ในบทนี้จะกล่าวถึงมาตรฐานที่เกี่ยวข้องเฉพาะเครือข่าย โดยจะเน้นที่ คำจำกัดความสถาปัตยกรรมลำดับชั้น ตาม ข้อกำหนด ของ ISO และ ในบทที่ 2 จะ กล่าวถึงรูปแบบโพรโทคอล HDLC ซึ่งเป็น โพรโทคอล ชั้นที่ 2 ที่ถูกกำหนดโดย ISO

1.2 สถาปัตยกรรมลำดับชั้น

ถ้าพิจารณาเหตุการณ์ ที่เกิดขึ้น ในระหว่าง การส่งผ่านแฟ้มข้อมูล ระหว่างเครื่องคอมพิวเตอร์ ต่างเครื่องหมายการค้ากัน โดยผ่านเครือข่ายคอมพิวเตอร์ จะ พบว่ามีปัญหา เกิดขึ้นหลายประการด้วยกัน เช่น โปรเซสที่อยู่บนเครื่องคอมพิวเตอร์ต้นทางจะหาตำแหน่ง และ เส้นทางของโปรเซสที่ต้องการจะติดต่อด้วยที่ปลายทาง ได้อย่างไร ถ้าระบบเกิดความผิดพลาดขึ้น ขณะที่มีการส่งผ่านข้อมูลซึ่งอาจ เนื่องมาจากเครื่องคอมพิวเตอร์ทำงานผิดพลาดจนต้องหยุดการทำงานของตัวเอง จะมีวิธีการอย่างไรเพื่อที่จะทำให้ระบบกลับคืนสู่สภาพเดิมและพร้อมกันนั้นจะแน่ใจได้อย่างไรว่าข้อมูลที่ส่งผ่านกันมีการเรียงลำดับ

ของข้อมูลตลอดจนเรียงลำดับ บิตก่อนหลังได้ถูกต้องจะเห็นได้ว่าปัญหาดังกล่าวมีขนาดใหญ่มาก และจะเป็นการยากยิ่งถ้าจะ พยายามสร้างวัตถุ (object) ใด ๆ ขึ้นมา วัตถุ หนึ่งเพื่อแก้ปัญหาทั้งหมดที่เกิดขึ้นดังนั้น จึงได้เกิดแนวความคิด ที่จะแบ่งหน้าที่การแก้ปัญหาออกเป็นส่วนย่อย หลาย ๆ ส่วน พร้อมกันนั้นก็จัดระเบียบของปัญหาที่เกิดขึ้นให้อยู่ในกลุ่มเดียวกัน

จากหลักการนี้เอง จึงได้เกิดสถาปัตยกรรมลำดับชั้นขึ้น โดยแบ่งระดับชั้นของการเชื่อมต่อ ระหว่าง คอมพิวเตอร์ออกเป็น n ชั้น แต่ละชั้นจะมีหน้าที่เฉพาะอย่างดังแสดงในรูปที่ 1 การ เชื่อมต่อจะเกิดขึ้นใน 2 ลักษณะ คือ แนวนอนและแนวตั้งแนวนอนจะเป็นการสื่อสาร ระหว่าง ต้นทาง กับ ปลายทางที่อยู่ในระดับเดียวกันโดยการใช้โพรโทคอลระดับ นั้น ๆ เป็นตัวจัดการส่วนแนวตั้งจะเป็นการขอและให้บริการผ่านจุดเชื่อมต่อระหว่างชั้นที่อยู่ติดต่อกันเท่านั้น



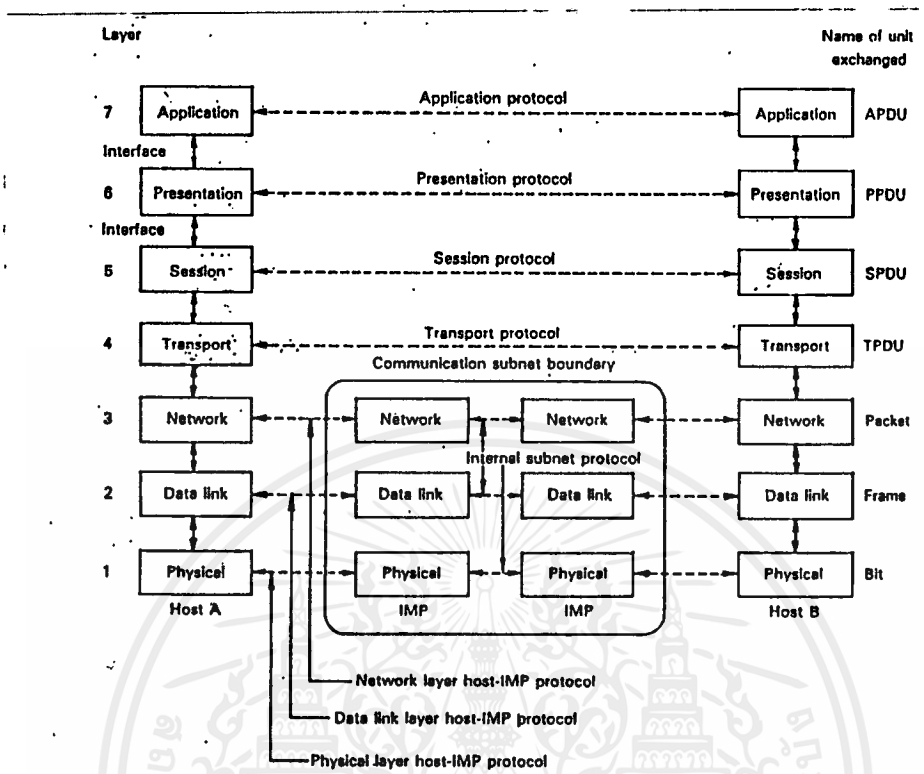
รูปที่ 1.1 สถาปัตยกรรมลำดับชั้น แต่ละชั้นจะมีโพรโทคอลประจำชั้น ซึ่งใช้ในการสื่อสารในระดับที่เท่ากันเท่านั้น แต่ละชั้นจะมีการสื่อสารถึงกันโดยผ่านจุดเชื่อมต่อระหว่างชั้น

1.3 รูปแบบอ้างอิงมาตรฐาน (Reference Model)

คณะกรรมการของ ISO ชุดที่ 9 7 (Technical Committee 97 : TC97) ที่ ดูแลทางด้านการประมวลผลระบบสารสนเทศ ได้ตระหนักถึงความต้องการ ที่จะเชื่อมต่อเครื่องคอมพิวเตอร์เพื่อสื่อสารถึงกันมากขึ้นเป็นเงาตามตัว ดังนั้นจึงได้ตั้งคณะทำงาน ที่เรียกว่า คณะกรรมการย่อยชุดที่ 16 (Subcomittee 16 : SC16) เพื่อศึกษาและกำหนดรูปแบบอ้างอิงมาตรฐาน และ ในช่วงปี ค.ศ. 1983 ได้มี การประกาศ รูปแบบอ้างอิงมาตรฐาน ของ ISO โดยใช้รหัสว่า ISO 7498 ซึ่งเป็น รูปแบบที่กล่าวถึง การเชื่อมต่อเครื่องคอมพิวเตอร์เข้าด้วยกันในลักษณะที่เรียกว่า ระบบเปิด (Open System Interconnection : OSI) ซึ่งประกอบด้วย 2 ส่วนใหญ่ ๆ คือ องค์ประกอบ ของสถาปัตยกรรม ลำดับชั้น ซึ่งกล่าวถึงส่วนประกอบที่ถูกกำหนดหรือสร้างขึ้นมาในลำดับ ชั้นต่าง ๆ และสถาปัตยกรรม ลำดับชั้น 7 ชั้น ซึ่งกล่าวถึง ฟังก์ชั่น และการให้บริการ ในแต่ละชั้น และในที่นี้ขอยกขึ้นมากล่าวเฉพาะ สถาปัตยกรรม ลำดับ ชั้น 7 ชั้น ซึ่งจะกล่าวถึงลำดับชั้นที่ 1 ถึง 3 โดยจะเน้นลำดับชั้นที่ 2 โดยเฉพาะสถาปัตยกรรมลำดับชั้น 7 ชั้นสถาปัตยกรรม ลำดับชั้น 7 ชั้น คือการแบ่งโพรโทคอลที่ใช้ในการควบคุมการส่งผ่านข้อมูลออกเป็นชั้นย่อย ๆ 7 ชั้น ดังแสดงในรูปที่ 1.2 โดยแต่ละชั้นจะมีหน้าที่การทำงานแยกจากกันอย่างอิสระ ในกรณีที่ทำงานร่วม กันสามารถติดต่อกันได้ โดยผ่านจุดเชื่อมต่อระหว่างชั้นที่ติดกันเท่านั้น

ก. Physical Layer เป็นชั้นที่ 1 จะกล่าวถึงการเชื่อมต่อทางกายภาพระหว่างเครื่องคอมพิวเตอร์ด้วยกัน เครื่องคอมพิวเตอร์กับเทอร์มินัล หรือเทอร์มินัล กับ เทอร์มินัล (DTE / DTE , DTE / DCE) มาตรฐานของชั้นนี้จะกล่าวทางด้านคุณสมบัติ ทางกล (mechanical) คุณสมบัติทางไฟฟ้า (eletrical) หน้าที่ของการทำงาน (function) และขั้นตอนในการทำงาน (procedural) ของการเชื่อมต่อสำหรับข้อมูลในชั้นนี้จะอยู่ในรูปแบบที่เป็นบิต (bit stream)

ข. Data Link Layer เป็นชั้นที่ 2 จะกล่าวถึงการควบคุม และตรวจสอบความผิดพลาดของข้อมูล การควบคุมการส่งผ่านข้อมูลและการให้บริการในลักษณะ ต่าง ๆ เช่น Connection Oriented, Connectionless สำหรับข้อมูลชั้นนี้จะถูกจัดให้เป็นกลุ่ม ซึ่งเรียกโดยทั่ว ๆ ไปว่า "เฟรม (Frame)"



รูปที่ 1.2 สถาปัตยกรรมลำดับชั้นตามรูปแบบของการเชื่อมต่อระบบเปิด

ค. Network Layer เป็นชั้นที่ 3 จะกล่าวถึงการควบคุมการทำงานของชุมสายที่ใช้ในการสวิตช์ข้อมูลจากต้นทางไปยังปลายทางการแก้ปัญหาของการมีข้อมูลภายในระบบมากเกินกว่าที่จะให้บริการได้ สำหรับข้อมูลในชั้นนี้จะถูกจัดให้เป็นกลุ่มซึ่งเรียกโดยทั่ว ๆไปว่า " กลุ่มข้อมูล "(packet)"

1.4 ข้อกำหนดของ OSI สำหรับโพรโทคอลชั้นที่ 2

จุดมุ่งหมายของ OSI ได้กำหนดว่า โพรโทคอลในชั้นที่ 2 จะต้องให้ความเชื่อถือได้ในการส่งผ่านข้อมูลของผู้ใช้ระหว่างระบบ จากข้อกำหนดนี้เองทำให้สามารถอธิบาย ฟังก์ชั่นและบริการของโพรโทคอลชั้นที่ 2 อย่างคร่าว ๆ ดังนี้

1.5 ฟังก์ชั่นของโพรโทคอลในชั้นที่ 2

ก. การเริ่มต้นการเชื่อมต่อ (initalization) ใช้สำหรับ การสถาปนา การเชื่อมต่อระหว่าง entity ในชั้นที่ 2 ระหว่างระบบไปบนเส้นทางในการส่งผ่านข้อมูลที่ได้ถูกกำหนดขึ้นมาแล้ว ซึ่งเส้นทางนี้อาจจะมีค่ามากกว่าหนึ่งเส้นทางก็ได้

ข.การกำหนดจุดปลายทางระหว่างระบบ (identification) ใช้สำหรับกำหนดผู้รับและผู้ส่งข้อมูลระหว่างระบบ เพื่อให้ข้อมูลส่งผ่านกันได้อย่างถูกต้อง

ค. การเข้าจังหวะระหว่างระบบ (Synchronization) ใช้สำหรับปรับและสร้างความคงอยู่ให้ผู้รับและผู้ส่งระหว่างระบบ มีกลไกในการเข้ารหัสและถอดรหัสระหว่างกันได้อย่างถูกต้อง

ง. การแบ่งส่วนและการจำกัดขอบเขต (segmenting and delimiting) ใช้สำหรับแบ่งข้อมูลของผู้ใช้ออกเป็นส่วนย่อย ๆ มีขนาดที่แน่นอนและเหมาะสมที่จะใช้ในการส่งผ่านระหว่างระบบเพราะว่าการส่งผ่านข้อมูล ไปบนสื่อนำสัญญาณอาจจะได้รับการรบกวนจากสัญญาณภายนอก ทำให้ข้อมูลไม่สามารถถูกนำไปใช้งานได้ ทำให้ต้องมีการส่งผ่านข้อมูลเดิมจนกว่าจะสามารถนำไปใช้งานได้ ถ้าข้อมูลมีขนาดยาวเกินไป จะทำให้เสียเวลามากในการส่งผ่านข้อมูลซ้ำ แต่ถ้าข้อมูลมีขนาดเล็กจนเกินไปก็ทำให้ไม่มีประสิทธิภาพ

จ. ความโปร่งใสของข้อมูลต่อผู้ใช้ (transparency) ทำให้ข้อมูลในทุกๆ รูปแบบ (รหัส) ของผู้ใช้งานสามารถส่งผ่านถึงกันได้โดยไม่ต้องคำนึงถึงรหัสพิเศษต่าง ๆ ที่ ใช้ในชั้นที่ 2 ว่าจะเป็นอุปสรรคต่อการส่งผ่านข้อมูลระหว่างระบบ

ฉ. การควบคุมการไหลของข้อมูล (flow control) ทำให้ผู้รับสามารถจัดระเบียบการไหลของข้อมูลที่รับจากผู้ส่งเพื่อป้องกันไม่ให้ผู้ส่งส่งข้อมูลมามากเกินไปจนกระทั่งผู้รับไม่สามารถรับหรือตรวจสอบข้อมูลได้ทัน ซึ่ง มีผลทำให้เสียเวลา ในการเริ่มการ รับส่งข้อมูลระหว่างกันใหม่

ช. การควบคุมลำดับของข้อมูลและตรวจสอบความผิดพลาด (error and sequence control) ใช้สำหรับตรวจสอบความผิดพลาดของข้อมูลที่เกิดขึ้นระหว่างการส่งผ่าน ทำให้สามารถแยกแยะข้อมูลที่ผิดออกจากข้อมูลที่ถูกได้อย่างถูกต้องนอกจาก นี้ยังสามารถตอบกลับหรือแจ้งกลับไปยังผู้ส่งได้ว่า มีเหตุการณ์อะไรเกิดขึ้นเกี่ยวกับข้อมูลนั้น ๆ

ซ. การกลับคืนสู่สถานะเดิม (abnormal condition recovery) ใช้สำหรับตรวจสอบและการกลับคืนสู่สถานะเดิมได้อย่างถูกต้อง เช่น ในกรณีที่ระบบไม่ได้รับการ ตอบรับจากฝ่ายตรงข้าม การส่งผ่านข้อมูลที่มีลำดับไม่ถูกต้อง เป็นต้น วิธีการแก้ไขที่นิยมใช้กันมากคือ การตรวจสอบสถานะภายในเวลาที่กำหนด (time-out)

ฌ. การยกเลิกการเชื่อมต่อ (termination) ใช้สำหรับสิ้นสุด การส่งผ่าน ข้อมูลระหว่างระบบการยกเลิกนี้เป็นการยกเลิกการเชื่อมต่อเชิงตรรกเท่านั้นจะไม่รวมถึงการยกเลิกเส้นทางทางกายภาพที่ใช้ในการส่งผ่านข้อมูล

ญ. การบริหารการเชื่อมโยง (link mangment) ใช้สำหรับดูแลและจัดการการเชื่อมต่อและส่งผ่านข้อมูล เช่น การสถาปนา (establish) การ ปลดปล่อย (release) และ ลำดับของข้อมูล (sequence) เป็นต้น นอกจากนี้ ยังใช้ในการจัดการกับระบบที่มีรูปแบบของการเชื่อมต่อในลักษณะจุดต่อจุดและหลายจุดด้วย สถานีในความหมาย ของโพรโทคอลในชั้นที่ 2 ถูกแบ่งออกเป็น 3 ประเภท คือ สถานีปฐมภูมิ (primary station) เป็นสถานีที่มีสิทธิพิเศษมากที่สุดในโทโพโลยีหนึ่ง ๆ (แบบจุดต่อจุด หรือ หลายจุด) เช่น สามารถกำหนดให้สถานีที่เหลือทำงานตามคำสั่งสามารถส่งหรือรับ ข้อมูล กับสถานีที่เหลือทั้งหมดได้ ประเภทที่สองคือสถานีทุติยภูมิ (secondary station) เป็นสถานีที่ถูกควบคุมการทำงาน โดย สถานีปฐมภูมิ ในกรณีที่ต้องการส่งผ่านข้อมูลระหว่างสถานีทุติยภูมิด้วยกันเองจะต้องส่งผ่านสถานีปฐมภูมิก่อนเสมอ สถานีประเภทสุดท้ายคือ สถานีผสม (Combined station) เป็นสถานีที่รวมความสามารถ ของ สถานีปฐมภูมิและสถานีทุติยภูมิไว้ในสถานีเดียว ดังนั้นในโทโพโลยีที่มีการต่อ สถานีผสมเข้าหากัน จึงไม่มีสถานีใดมีสิทธิเหนือกว่ากัน

บทที่ 2
Protocol HDLC

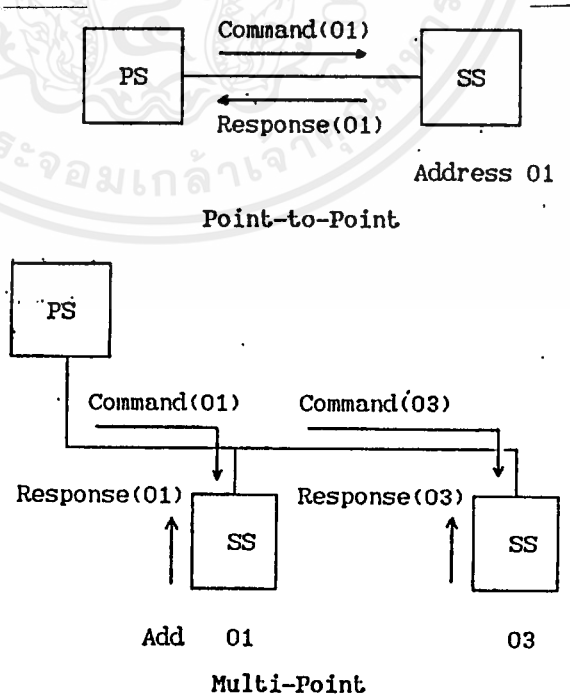
2.1 บทนำ

โพรโทคอล HDLC (High-level Data Link Control Protocol)

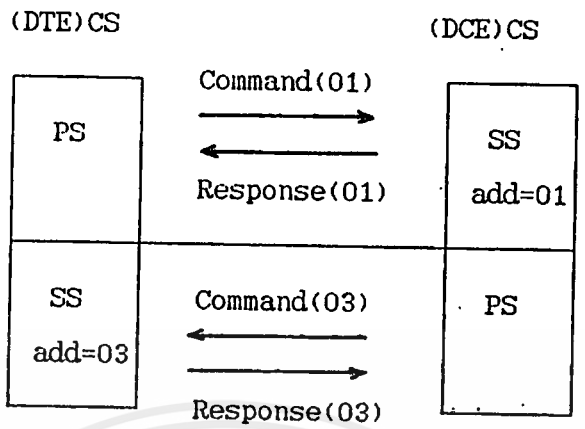
โพรโตคอล HDLC เป็นโปรโตคอลชั้นที่ 2 ที่ถูกกำหนดขึ้นโดย ISO โดยมีรหัส คือ " ISO 4335 " สามารถประยุกต์เข้ากับหลายรูปแบบของการเชื่อมต่อทางกายภาพ และมีหน้าที่การทำงานดังนี้

- ใช้ได้กับการเชื่อมต่อแบบจุดต่อจุดและหลายจุด
- ใช้ได้กับการส่งผ่านข้อมูลระยะใกล้และไกล
- ใช้ได้กับการส่งผ่านข้อมูลแบบฮาล์ฟดูเพล็กซ์และฟูลดูเพล็กซ์
- ใช้ได้กับระบบที่มีการกำหนดประเภทของสถานีทั้ง 3 ประเภท

อนึ่ง ระบบที่มีการกำหนดสถานีปฐมภูมิ และ สถานีทุติยภูมิ เรียกว่า ระบบอสมดุล (unbalanced system) ซึ่งการเชื่อมต่อเป็นไปได้ทั้งจุดต่อจุดและหลายจุด สำหรับระบบที่ประกอบด้วยสถานีผสมทั้งหมดเรียกว่า ระบบสมดุล (balance system) ซึ่งการเชื่อมต่อเป็นได้เฉพาะจุดต่อจุดเท่านั้น ดังแสดงในรูปที่ 2.1 และรูปที่ 2.2 ตามลำดับ



รูปที่ 2.1 แสดงระบบอสมดุล



PS = Primary Station (Master)

SS = Secondary Station (Slave)

CS = Combine Station

รูปที่2.2 แสดงระบบสมดุล

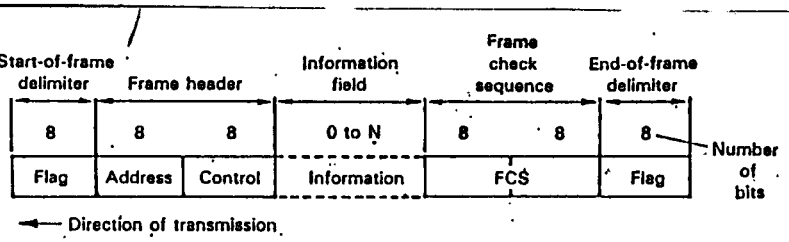
รูปแบบของการส่งผ่านข้อมูลตามข้อกำหนดของโพรโทคอล HDLC แบ่งออกเป็น 3 ลักษณะคือ

1. NRM (Normal Response Mode) ใช้กับระบบอสมดุลสถานีปฐมภูมิต้องการจะติดต่อสถานีทุติยภูมิในกรณีใดก็ได้ แต่สถานี ทุติยภูมิ จะมีสิทธิ์ติดต่อกับสถานีปฐมภูมิได้ก็ต่อเมื่อสถานีปฐมภูมิอนุญาตเท่านั้น

2. ARM (Asynchronous Response Mode) ใช้กับระบบอสมดุลสถานีทุติยภูมิสามารถจะติดต่อกับปฐมภูมิได้โดยไม่จำเป็นต้องให้สถานีปฐมภูมิอนุญาตก่อน

3. ABM (Asynchronous Balanced Mode) ใช้กับระบบสมดุล แต่ละสถานีสามารถส่งผ่าน ข้อมูล ถึงกันได้ โดยไม่จำเป็นต้องได้รับอนุญาตจากสถานีที่ต้องการจะติดต่อด้วยเสียก่อน

2.2 รูปแบบของเฟรม (Frame format)



รูปที่2.3 รูปแบบเฟรมของ HDLC

รูปแบบเฟรมของโพรโทคอล HDLC ประกอบด้วย 5 เขต ดังแสดงในรูปที่ 2.3 คือ เขตแฟล็ก (flag field) เขตแอดเดรส (address field) เขตควบคุม (control field) เขตข้อมูล (imformation field) และ เขตตรวจสอบเฟรม (Frame Check Sequence field (FCS)) แต่ละเขตสามารถอธิบายได้ดังนี้

2.3 เขตแฟล็ก

เขตแฟล็กมีขนาด 8 บิต เป็นเขตที่ใช้บอกถึงจุดเริ่มต้นและจุดสิ้นสุด ของแฟรม ซึ่งเรียกว่า แฟล็กเปิด (opening flag) และแฟล็กปิด (closing flag) มีรหัสคงที่คือ 01111110_B ($7E_H$)

2.4 เขตแอดเดรส

เขตแอดเดรส เป็นเขตที่ใช้บอกถึงหมายเลขของสถานีซึ่งขึ้นอยู่กับว่า เป็นระบบอสมดุล หรือสมดุลในระบบอสมดุลเมื่อสถานีปฐมภูมิส่งเฟรมไปยังสถานีทุติยภูมิ แอดเดรสนี้จะเป็นหมายเลขของ สถานี ทุติยภูมิ ถ้าเฟรมนี้ถูกส่งจาก สถานี ทุติยภูมิ แอดเดรส นี้จะเป็นหมายเลขของ สถานี ทุติยภูมิ นั้น ในระบบสมดุลถ้าเฟรมที่ส่งเป็นเฟรมคำสั่ง (command frame) แอดเดรสนี้จะเป็นหมายเลขของสถานีปลายทางแต่ถ้าเฟรมที่ส่งเป็นเฟรมตอบสนอง (response frame) แอดเดรส นี้จะเป็นหมายเลขของสถานีที่ส่งเฟรมนั้นเขตแอดเดรสมีขนาด 8 บิตหรือจำนวนเท่าของ 8 บิต โดยมีข้อกำหนดว่าถ้ามีบิตที่มีนัยสำคัญน้อย (Least Significant Bit (LSB)) มีค่าเป็น "0 หมายความว่า แอดเดรสถูกขยายออกไปอีก 8 บิต แต่ถ้ามีค่าเป็น 1" หมายถึงเขตของแอดเดรสสิ้นสุดลงที่ 8 บิต นอกจากนี้ถ้าแอดเดรสมีค่าเป็น 11111111_B (FF_H) จะหมายความว่า เป็นแอดเดรสของทุก ๆ สถานีที่ต่ออยู่ ใช้ประโยชน์ในการกระจายข้อมูลให้กับทุกสถานี โดยใช้แอดเดรสเพียงหมายเลขเดียว

2.5 เขตควบคุม

เขตควบคุม เป็นเขตที่ใช้บอกถึงประเภทของเฟรมซึ่งแบ่งออกเป็น 3 ประเภท ดังแสดงในรูปที่ 2.4 คือ เฟรมข้อมูล (information frame) ซึ่งถูกกำหนดโดยบิตที่มี นัยสำคัญน้อย บิต แรกของขอบเขตควบคุมจะต้องมีค่าเป็น "0" เป็นเฟรมที่ใช้ในการรับส่งข้อมูล ประเภทที่สองคือ เฟรมควบคุม และ ดูแล (supervisory frame) ซึ่ง

ถูกกำหนดโดยบิตที่มีนัยสำคัญน้อยสองบิตแรกจะต้องมีค่าเป็น 10 เป็นเฟรมที่ใช้ในการสอบถาม และแสดงสถานภาพการรับข้อมูลของแต่ละสถานี และประเภทสุดท้ายคือ เฟรม Unnumbered ซึ่งถูกกำหนดโดยบิตที่มีนัยสำคัญน้อย สองบิตแรกจะต้องมีค่าเป็น "11" เป็นเฟรมที่ใช้ในการส่งผ่านคำสั่ง หรือ คำตอบรับในการควบคุมระบบ เช่น การสถาปนาการเชื่อมต่อการยกเลิกการเชื่อมต่อเป็นต้น สำหรับส่วนต่าง ๆ ที่อยู่ภายในเขตควบคุม สามารถอธิบายได้ดังนี้

	CONTROL FIELD BITS							
BIT ORDER	1	2	3	4	5	6	7	8
I frame format.	0	N(S)			P/F	N(R)		
S frame format	1	0	S	S	P/F	N(R)		
U frame format	1	1	M	M	P/F	M	M	M

9 (ก)

	EXTENDED CONTROL FIELD BITS															
	1st Octet									2nd Octet						
BIT ORDER	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
I frame format	0	N(S)							P	N(R)						
S frame format	1	0	S	S	0	0	0	0	/	N(R)						
U frame format	1	1	M	M	*	M	M	M	F	0	0	0	0	0	0	0

* The value of this bit is undefined

LEGEND

N(S) = Send Sequence Number
N(R) = Receive Sequence Number
P/F = Poll/Final Bit
S = Supervisory Bits
M = Modifier Bits

9 (ข)

รูปที่ 2.4 การเคลื่อนที่ของหน้าต่าง (ก)เขตควบคุม 8 บิต (ข)เขตควบคุม 16 บิต

ก. N (S) (send sequence number) ใช้ในการควบคุม การไหลของข้อมูล โดยใช้เป็นค่าที่บอกว่าเป็นเฟรมที่เท่าไร

ข. N (R) (receive sequence number) ใช้ในการควบคุมการไหลของข้อมูล โดยฝ่ายรับใช้เป็นค่าที่บอกว่าคาดว่า จะได้รับเฟรมต่อไปเป็นเฟรมที่เท่าไร หรือ เป็นค่าที่บอกว่าเฟรมที่ไม่สามารถรับได้ เป็นเฟรมที่เท่าไร สำหรับคำตอบรับ หรือ คำตอบปฏิเสธ ตามลำดับ

ค. P / F (Poll / Final) การ poll หมายถึงการที่สถานีใดสถานีหนึ่งส่งคำสั่งไปให้อีกสถานีหนึ่งเพื่อต้องการการตอบกลับ การ final หมายถึงการที่สถานีที่ถูกทำการตอบรับต่อการ poll นั้น ๆ จะเห็นได้ว่าการ poll จะเกิดกับเฟรมที่ เป็นคำสั่ง

เสมอ และจะต้องกำหนดบิต P / F ให้มีค่าเป็น "1" แต่ถ้าเป็น "0" จะไม่เกิดการ poll สำหรับการ final จะเกิดกับเฟรมที่เป็นคำตอบรับเสมอ และจะต้องกำหนดบิต P / F ให้มีค่าเป็น "1" แต่ถ้าเป็น "0" จะไม่เกิดการ final

ง. S (Supervisory code) เป็นรหัสที่กำหนดชุดคำสั่ง และคำตอบรับดังแสดงในรูปที่ 2.5 ของเฟรมควบคุมและดูแล

S FRAMES	BITS	
COMMANDS/RESPONSES	3	4
RR — Receive Ready	0	0
REJ — Reject	0	1
RNR — Receive Non Ready	1	0
SREJ — Selective Reject	1	1

U FRAMES		CONTROL FIELD BITS				
COMMANDS	RESPONSES	3	4	6	7	8
SNRM		0	0	0	0	1
SNRME		1	1	0	1	1
SARM	DM	1	1	0	0	0
SARME		1	1	0	1	0
SABM		1	1	1	0	0
SABME		1	1	1	1	0
DISC	RD	0	0	0	1	0
	UA	0	0	1	1	0
SIM	RIM	1	0	0	0	0
TEST	TEST	0	0	1	1	1
XID	XID	1	1	1	0	1
UI	UI	0	0	0	0	0
	FRMR	1	0	0	0	1

รูปที่ 2.5 รหัสชุดคำสั่งและคำตอบรับ

จ. M (Unnumbered code) เป็นรหัสที่กำหนดชุดคำสั่งและคำตอบรับ ดังแสดงในรูปที่ 2.5 ของเฟรม unnumbered

อนึ่ง เขตควบคุมจะมีขนาด 8 หรือ 16 บิตก็ได้ขึ้นอยู่กับการกำหนดของระบบนั้น ๆ ในกรณีที่เป็นแบบ 8 บิต จะทำให้สถานีสามารถรับส่งข้อมูลได้สูงสุด 7 เฟรม ติดต่อกัน (modulo-8) โดยไม่ต้องรอการตอบกลับจากสถานีตรงข้ามและในกรณีที่เป็นแบบ 16 บิต จะทำให้สถานีสามารถรับส่งข้อมูลกันได้ สูงสุด 127

เฟรมติดต่อกัน . (modulo-128) โดย ไม่ต้องรอการตอบกลับจากสถานีตรงกันข้าม

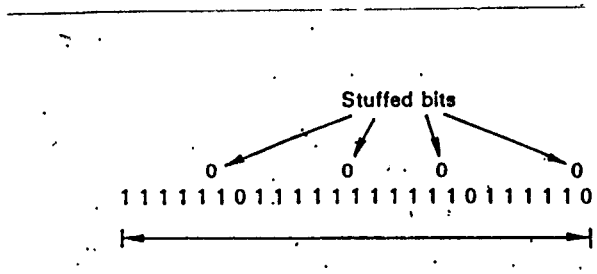
2.6 เขตข้อมูล

เขตข้อมูล มีขนาดเป็นจำนวนเท่าของ 8 บิต ใช้สำหรับบรรจุข้อมูลซึ่งก็คือ PDU ของโพรโทคอลที่อยู่ในชั้นที่สูงกว่านั้นเอง

2.7 เขตตรวจสอบเฟรม

เขตตรวจสอบเฟรม มีขนาด 16 บิต เป็นเขตที่ใช้ในการตรวจสอบความถูกต้องของเฟรมที่ได้รับที่ปลายทางโดยใช้หลักการของ CRC (Cyclic Redundancy Code) สำหรับโพรโทคอล HDLC ใช้ $X^6 + X^{12} + X^5 + 1$ เป็น generator polynomial

จากข้างต้นจะเห็นว่า รหัสแฟล็กมีค่าคงที่ ดังนั้นรหัสใดๆ จะเหมือนกับแฟล็กไม่ได้เพราะจะทำให้การรับส่งโพรโทคอลผิดพลาดได้และยังไม่สอดคล้องกับข้อกำหนดที่กล่าวถึง ความโปร่งใสของข้อมูลต่อผู้ใช้ การแก้ไขปัญหาทำได้โดยวิธีที่เรียกว่า การแทรกบิตศูนย์ (Zero bit insertion) โดยมีข้อกำหนดว่าถ้ารหัสที่ไม่ใช่แฟล็ก (ยกเว้นรหัส abort และ idle) เมื่อมีบิตที่มีค่าเป็น 1 เกิดขึ้นอย่างน้อย 5 บิตติดต่อกัน จะทำการแทรกบิต "0" ลงไปในตำแหน่งที่ 6 เสมอ และบิตที่อยู่ในตำแหน่งถัดไปก็จะถูกเลื่อนไป 1 ตำแหน่งทุก ๆ บิต ดังแสดงในรูปที่ 2.6 เมื่อปลายทางตรวจพบลักษณะเช่นนี้ ก็จะดึง บิต "0" ออกและเลื่อนบิตในตำแหน่งถัดไปเข้า 1 บิต ก็จะได้ข้อมูลเดิมกลับมา สำหรับ รหัส abort และ idle คือรหัสที่บิต "0" และตามด้วย "1" เป็นจำนวน 7 และอย่าง น้อย 15 บิตติดต่อกันตามลำดับ



รูปที่ 2.6 ลักษณะของการแทรกและดึงบิต "0"

2.8 ลักษณะการส่งเฟรมโต้ตอบของโพรโทคอล HDLC

U- Frame (Unnumbered frame) ใช้สำหรับ Link Control ได้แก่

1. SABM (Set Asynchronous Balance Mode) เป็น Command ที่ใช้สำหรับ Set up หรือ Initial ในระดับ Frame Level (X.25 Level 2) เพื่อเปลี่ยนจาก Disconnect Mode มาเป็น Operational Mode ที่ใช้ในการรับส่งข้อมูลกันต่อไป นอกจากนี้ ยังทำให้ Sending และ Receiving Counter ถูกรีเซตเป็น "0" ด้วยเพื่อเริ่มต้นกันใหม่

2. UA (Unnumbered Acknowledge) เป็น Response ที่ตอบรับ SABM เพื่อเข้าสู่ Operational Mode ดังกล่าว

3. Disc (Disconnect) เป็น Command ที่ทำให้เปลี่ยนจาก Operational-Mode มาเป็น Disconnect Mode ซึ่งไม่สามารถรับส่งข้อมูลกันได้

4. DM (Disconnect Mode) เป็น Response มีหน้าที่ 2 อย่างคือ

4.1 ขอให้สถานีปฐมภูมิ ส่ง Command SABM มาเพื่อเปลี่ยนเข้าสู่ Operational Mode

4.2 เป็น Response ที่บอกว่าไม่สามารถเข้าสู่ Operational-Mode ได้

5. FRMR (Frame Reject) เป็น Response อยู่ใน Operational Mode เพื่อเป็น Reject frame ที่ผิดปกติในลักษณะ

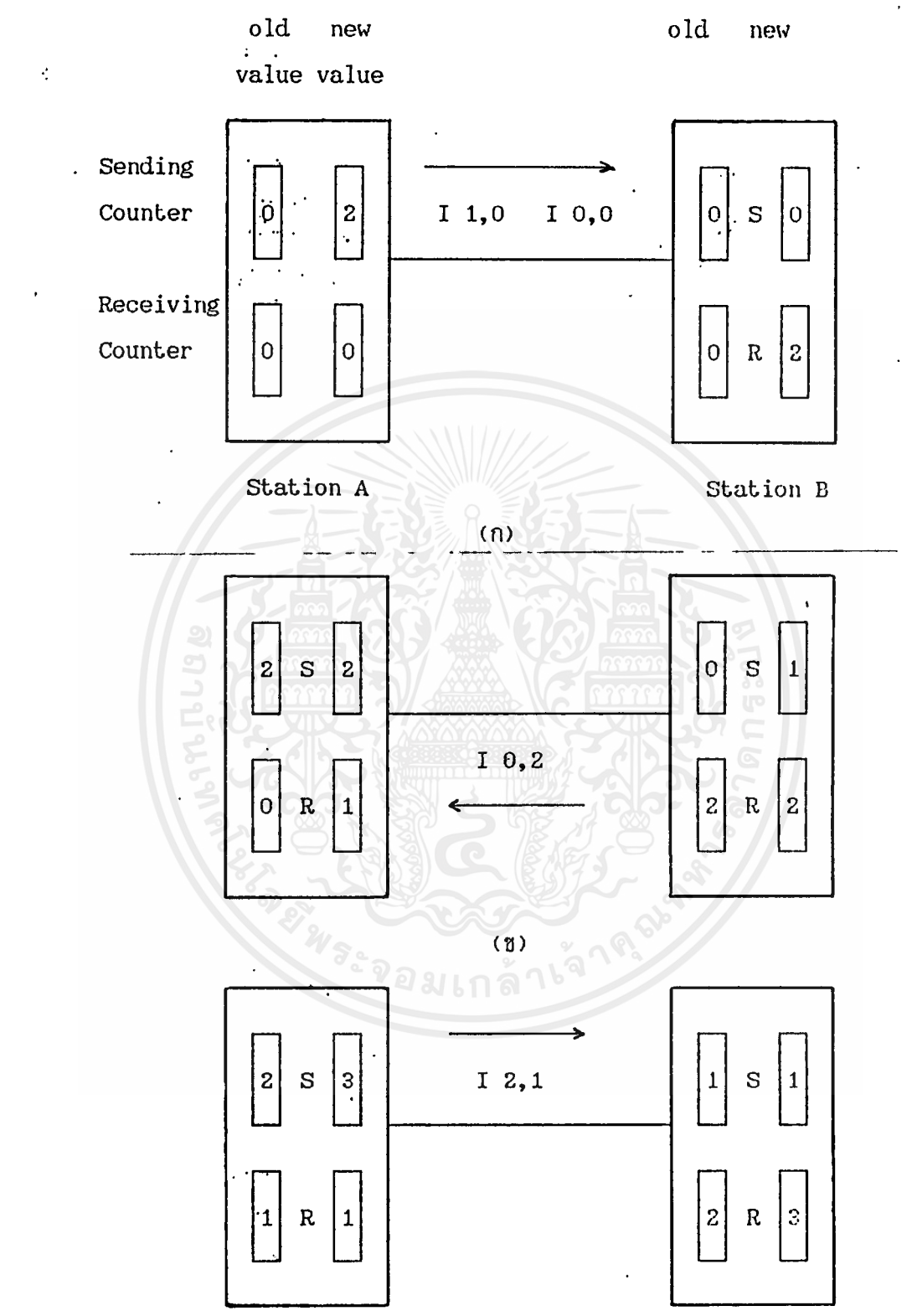
5.1 Control field ที่ผิดปกติ

5.2 มี Information field ที่ยาวเกินไป

5.3 Frame ที่ไม่ควรมี information field แต่กลับมีเช่น SABM เป็นต้น

5.4 Frame ที่มี Flow control (counter) ผิดปกติ

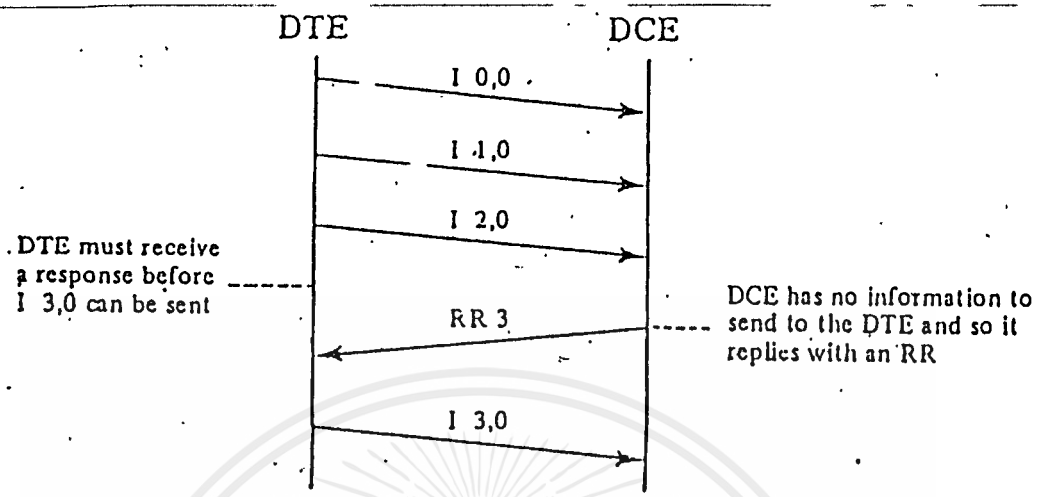
Information frame เป็น Frame ที่มีข้อมูลที่จะรับส่งบรรจุอยู่ และมี Counter ในการตอบรับคือ N (R) และในการส่งคือ N (S) เพื่อใช้ใน Flow and Error Control ดังตัวอย่างในรูปที่ 2.7 ก, ข, และ ค



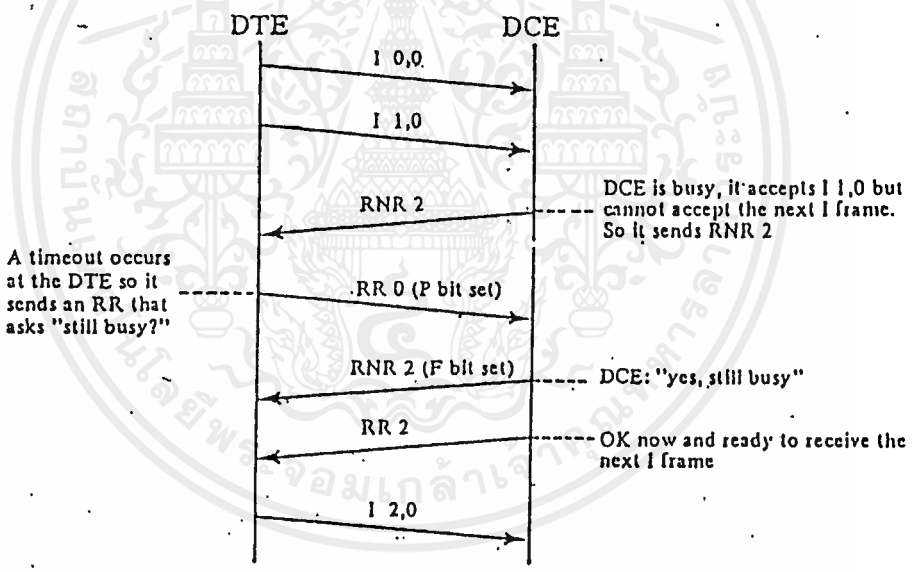
รูปที่ 2.7

* Receiving Counter จะเพิ่มขึ้นเมื่อทางด้านรับ frame ที่ไม่มี error

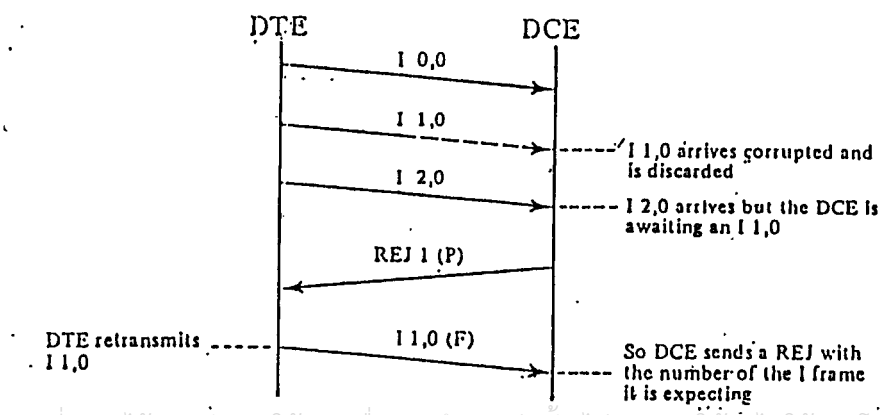
Supervisory frame (S-frame) ใช้สำหรับ Flow & Error Control



การใช้ RR สำหรับแจ้งว่าได้รับ I Frames แล้ว

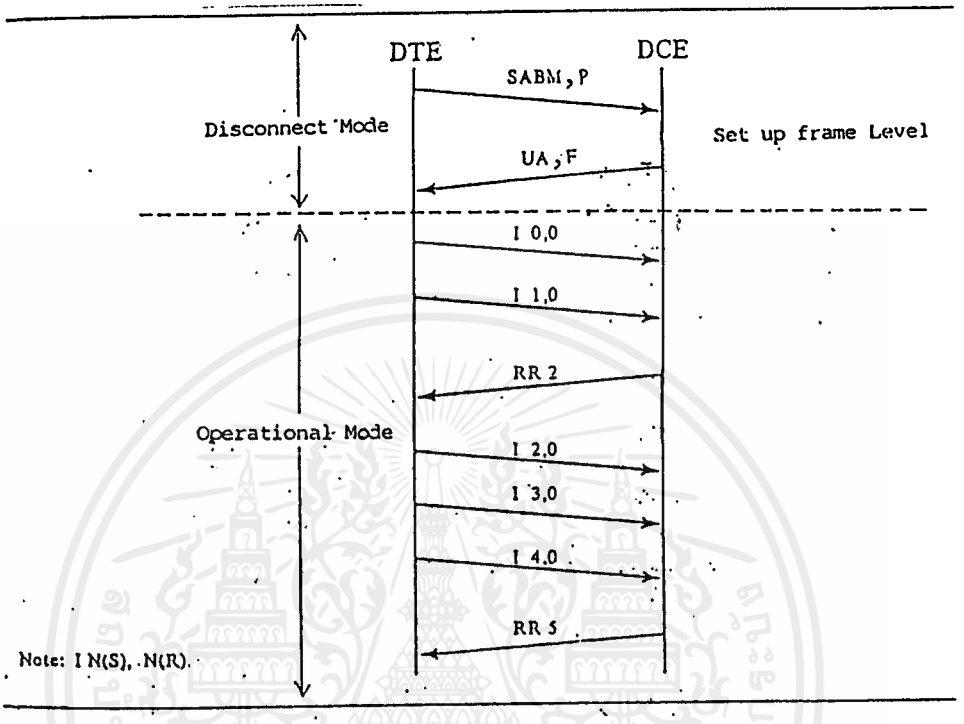


เฟรม RNR เช่นในเงื่อนไข Busy



การใช้เฟรม REJ

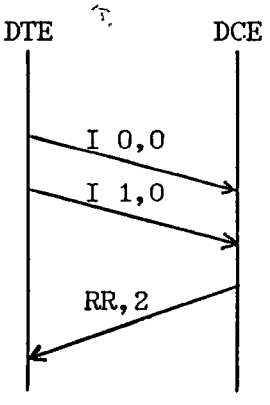
ตัวอย่างการถ่ายโอนข้อมูล



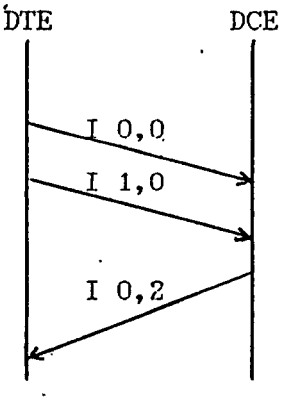
การถ่ายโอนข้อมูลโดยใช้เฟรม RR

2.9 การตอบรับชุดข้อมูล (Acknowledgement)

โดยทั่วไปในการ Acknowledge I - frame จะใช้ S - frame คือ RR , RNR, หรือ REJ แต่ในบางครั้งอาจจะใช้ I - frame ส่งกลับมา Acknowledge ก็ได้ ซึ่งเรียกว่า Imply Acknowledge



Normal Acknowledge



Imply Acknowledge

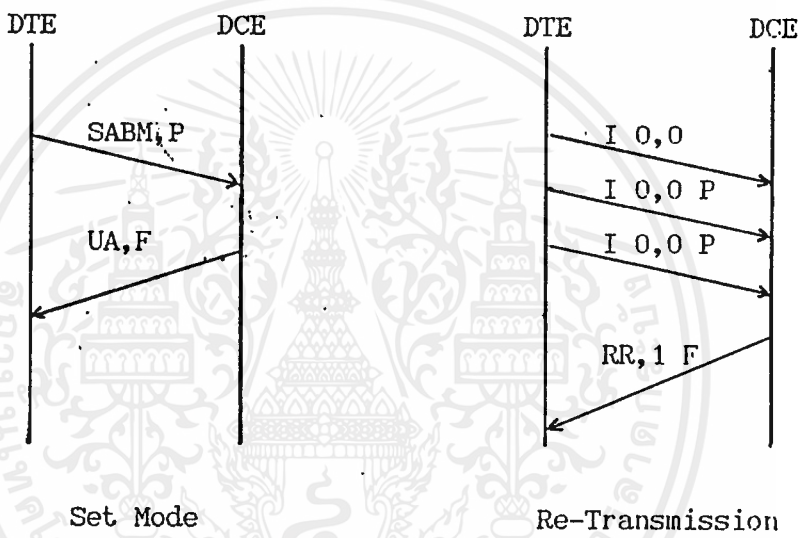
<u>บิต Poll / Final</u>

บิต Poll ใช้ใน Command ที่ต้องการได้รับ Response ตอบกลับมา

บิต Final ใช้ใน Response ที่ตอบรับ Command ที่ส่งบิต Poll มา

<u>การใช้บิต Poll / Final</u> จะใช้ใน 2 ลักษณะคือ

1. การ Set Mode
2. การ Re - Transmission



<u>พารามิเตอร์ K, T_0, T_1, N_1, N_2</u>

K = จำนวน Outstanding frame ที่ส่งไปโดยไม่ต้องรอ Acknowledge

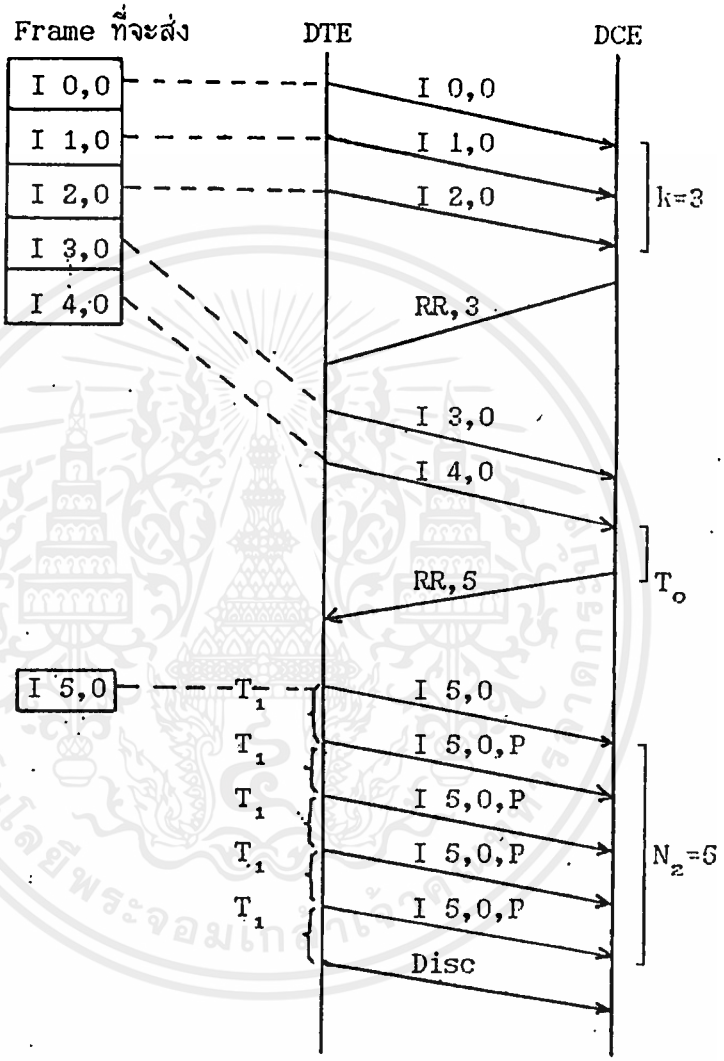
T_0 = เป็น Timer สำหรับคอย Outstanding frame ที่กำหนดโดยค่า K ก่อนจะตอบ Acknowledge กลับไป

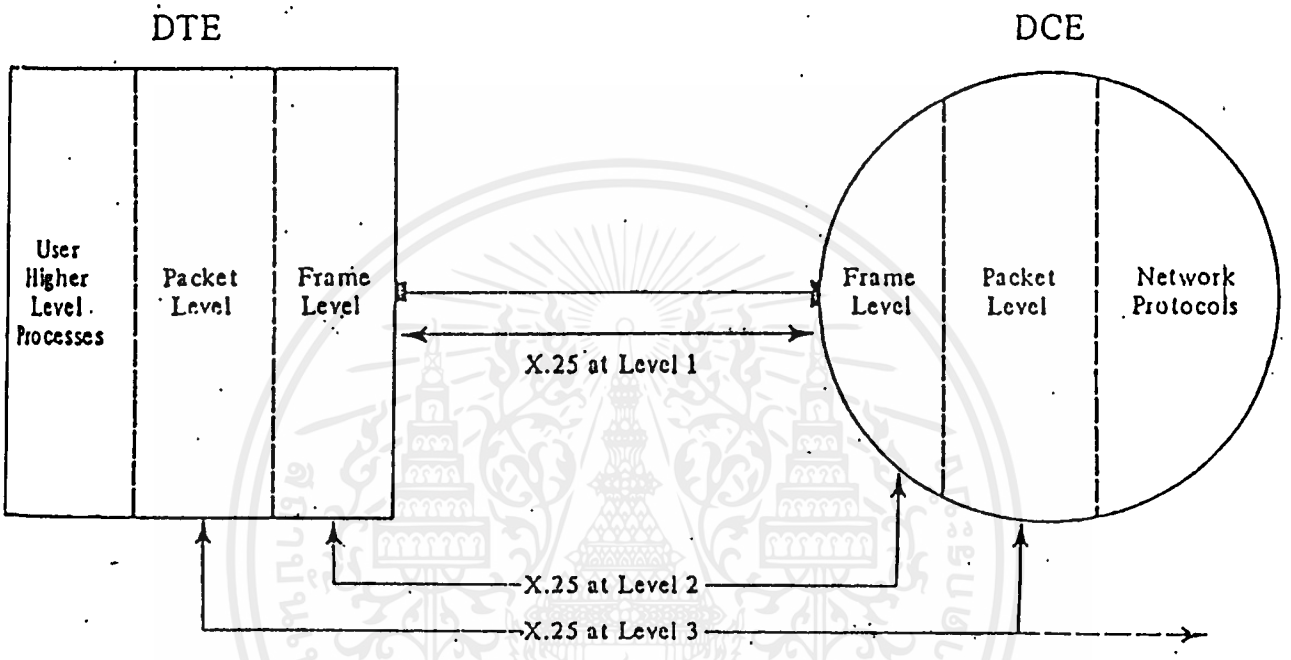
T_1 = เป็น Timer สำหรับคอยการตอบ Acknowledge กลับมา โดยที่ได้ส่ง Outstanding frame ครบตามกำหนดค่า K แล้ว

N_1 = เป็นจำนวนบิตทั้งหมดที่อยู่ใน Information frame (ไม่รวม Flag) จะมากหรือน้อยขึ้นอยู่กับ Information field

N_2 = จำนวน Re - Transmission frame ที่เกิดขึ้น (รวม frame แรกด้วย) หลังจากไม่ได้รับ Acknowledge ตามเวลา T_1 ที่กำหนด

ตัวอย่าง





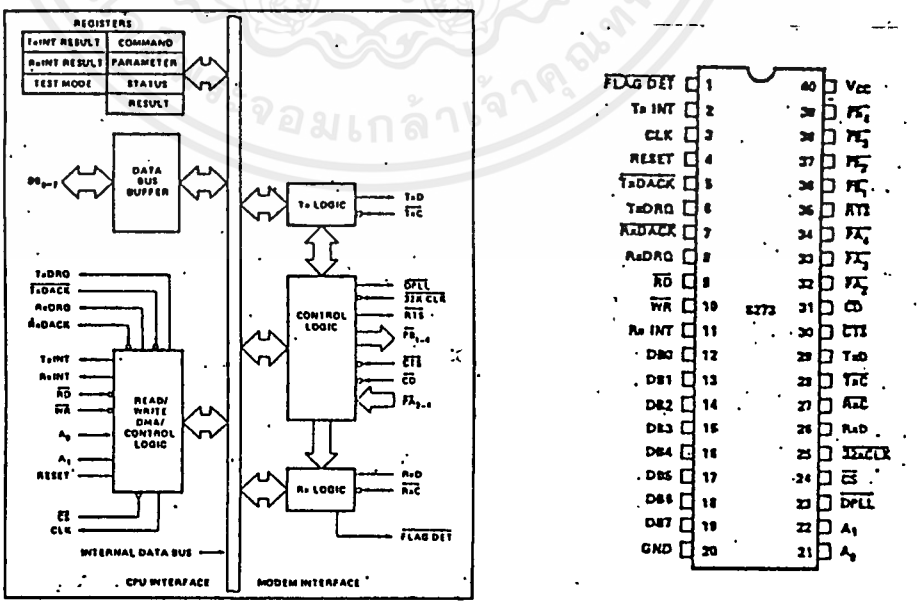
ลำดับชั้น 2: การเชื่อมโยง DTE/DCE

บทที่ 3
8273 กับพอร์ตที่ใช้ในการตรวจสอบ โพรโทคอล HDLC / SDLC

3.1 บทนำ

ดังที่ได้กล่าวมาแล้วว่าโพรโทคอล SDLC มีรูปแบบที่เป็นมาตรฐานซึ่งประกอบไปด้วยเขตที่ต่างชนิดกัน ทำให้การรับส่งข้อมูลมีได้หลายรูปแบบรวมทั้ง สถานะ idel หรือ active ที่มีการส่งแฟลกอยู่ตลอดเวลา รวมทั้งวิธีการแทรกบิตศูนย์ลงไปในข้อมูล ภาระต่าง ๆ นี้ถูกจัดการโดยซีพียู (CPU) ทั้งหมดก็จะมีความยุ่งยากมาก เนื่องจากซีพียูใช้กับงานควบคุมหรือประมวลผลทั่วๆ ไปถ้าจะนำมาใช้กับการตรวจสอบโพรโทคอลนี้จะต้องใช้เวลาในการแยกแยะและจัดการเกี่ยวกับข้อมูลเป็นอันมาก เช่น การดึงบิตศูนย์ออกจากการแทรกมาจากต้นทาง การตรวจสอบ FCS ฯลฯ แต่ถ้าใช้พอร์ตที่มีหน้าที่เฉพาะอย่างจะทำให้การทำงานสะดวกขึ้นและจะลดภาระของซีพียูลง ทำให้ซีพียูมีเวลาที่จะไปทำงานต่าง ๆ ในระบบได้

ในที่นี้จะขอกล่าวถึงไอซี (IC) เบอร์ 8273 ซึ่งถูกนำมาใช้เป็นพอร์ตในการตรวจสอบและแยกแยะส่วนต่าง ๆ ของโพรโทคอล HDLC / SDLC และรับส่งข้อมูลระหว่างไมโครโปรเซสเซอร์กับอุปกรณ์อินพุตเอาต์พุตแบบอนุกรม 8273 เป็นชิปขนาด 40 ขา มีการจัดขาและโครงสร้างภายใน ดังแสดงในรูปที่ 3.1



รูปที่ 3.1 โครงสร้างภายในและการจัดขาของไอซี 8273

จากสัญญาณต่าง ๆ ในรูปที่ 3.1 สามารถสรุปหน้าที่ได้ดังตารางที่ 3.1

ตารางที่ 3.1 สรุปสัญญาณต่าง ๆ ของ 8273

สัญลักษณ์	หมายเลขขา	ชื่อสัญญาณและหน้าที่	ชนิดของสัญญาณ
Vcc	40	Power Supply: ไฟเลี้ยง +5V	
GND	20	Ground:กราวน์	
RESET	4	Reset: สัญญาณนี้จะแอคตีฟที่ลอจิก"1"และจะบังคับให้ชิป 8273 อยู่ในสภาวะไอเดิลจนกว่าซีพียูจะปล่อยคำสั่งออกมา ส่วนทางด้านโมเดมสัญญาณเอาต์พุตจะถูกบังคับให้เป็น "1"ด้วย	อินพุต
CS	24	Chip Select:เป็นสัญญาณที่ใช้เลือกชิป 8273ซึ่งแอคตีฟที่ลอจิก"0"ในการใช้งานจะใช้ร่วมกันสัญญาณ A1,A0,RD,WR	อินพุต
DB7 - DB0	19-12	Data Bus: เป็นสัญญาณที่จะต่อกับระบบบัสไมโครโปรเซสเซอร์	สองทิศทาง
WR	10	Write Input:สัญญาณนี้แอคตีฟที่ลอจิก"0"เป็นสัญญาณเขียน ที่ใช้ในการควบคุมการถ่ายโอน ข้อมูลหรือคำสั่งจากซีพียูไปยัง 8273	อินพุต
RD	9	Read Input:สัญญาณนี้แอคตีฟที่ลอจิก"0"เป็นสัญญาณอ่านที่ใช้ในการควบคุมการถ่ายโอนข้อมูลหรือสถานภาพจากชิป 8273 ไปยัง ซีพียู	อินพุต
TxINT	2	Transmitter Interrupt: เป็นสัญญาณขัดจังหวะของภาคส่ง ที่ใช้ร้องขอเพื่อที่จะใช้บริการ	เอาต์พุต
RxINT	11	Receiver Interrupt:เป็นสัญญาณขัดจังหวะภาครับ ที่ใช้ร้องขอเพื่อที่ภาครับจะใช้บริการ	เอาต์พุต
TxDRQ	6	Transmitter Data Request:สัญญาณร้องขอให้ถ่ายโอนข้อมูลระหว่างหน่วยความจำและ8273 สำหรับการทำงานในการส่ง	เอาต์พุต

ตารางที่ 3.1(ต่อ) สรุปสัญญาณต่าง ๆ ของ 8273

สัญลักษณ์	หมายเลขขา	ชื่อสัญญาณและหน้าที่	ชนิดของสัญญาณ
RxDRQ	8	Receiver DMA Request:สัญญานร้องขอให้ถ่ายโอนข้อมูลระหว่าง 8273 และหน่วยความจำ สำหรับการทำงานในการรับ	เอาต์พุต
TxDACK	5	Transmitter DMA Acknowledge:เป็นสัญญาณตอบรับ DMA ภาคส่งที่แจ้งให้ 8273 ทราบนั้นคือ ได้อนุญาตโดยจัดไซเกิล TxDMAให้8273	อินพุต
RxDACK	7	Receiver DMA Acknowledge:เป็นสัญญา ณตอบรับ DMA ภาครับที่แจ้งให้ 8273 นั้นคือได้อนุญาตโดยจัดไซเกิล RxDMA ให้ชิป 8273	อินพุต
A1 - A0	22-21	Address:ซีพียูจะใช้สายทั้ง 2 เส้นนี้เชื่อมโยงการเลือก รีจิสเตอร์ภายใน 8273	อินพุต
TxD	29	Transmitter Data: สายเส้นนี้ใช้สำหรับส่งข้อมูลแบบอนุกรมไปยังช่องสัญญาณสื่อสาร	เอาต์พุต
TxC	28	Transmitter Clock:เป็นสัญญาณนาฬิกาฐานเวลาที่ใช้ในการซิงโครนัสกับข้อมูลที่ส่งออกไป	อินพุต
RxD	26	Receiver Data:สายเส้นนี้ใช้สำหรับรับข้อมูลแบบอนุกรมจากช่องสัญญาณสื่อสาร	อินพุต
RxC	27	Receiver Clock:เป็นสัญญาณนาฬิกาที่ใช้ในการซิงโครนัสกับข้อมูลจากภายนอกที่รับเข้ามา	อินพุต
32xCLK	25	32x Clock:เป็นสัญญาณนาฬิกาที่เตรียมไว้ให้ใช้ในการกู้คืน เมื่อโมเดมอะซิงโครนัสนำมาใช้ สัญญาณนาฬิกา 1x ของสถานีลูปสามารถดำเนินการคลาดเคลื่อนได้ ดังนั้นการใช้งานในรูปแบบลูปจึงใช้สัญญาณนาฬิกา 32xร่วมกันกับเอาต์พุต DPLL (ขานี้จะต้องถูกต่อลงกราวน์-เมื่อไม่ได้ใช้)	อินพุต
DPLL	23	Digital Phase Locked Loop:เป็นเอาต์พุตที่สามารถต่อเข้ากับ RxC และหรือTxC เมื่อสัญญาณนาฬิกา 1x ไม่สามารถหาได้DPLL จะถูกนำมาใช้กับสัญญาณนาฬิกา 32x	เอาต์พุต

ตารางที่ 3.1(ต่อ) สรุปสัญญาณต่าง ๆ ของ 8273

สัญลักษณ์	หมายเลขขา	ชื่อสัญญาณและหน้าที่	ชนิดของสัญญาณ
FLAG-DET	1	Flag Detect: เป็นสัญญาณเอาต์พุตที่ได้จากการตรวจสอบแฟลก จะแอคตีฟเมื่อเครื่องรับมีการรับแฟลก(01111110)จำนวน 1 แฟลก	เอาต์พุต
RTS	35	Request To Send:เป็นสัญญาณเอาต์พุตของ8273 ที่พร้อมจะส่งข้อมูล	เอาต์พุต
CTS	30	Clear To Send: เป็นสัญญาณอินพุตจากภายนอกหรือโมเดม ที่พร้อมจะรับข้อมูลจาก 8273	อินพุต
CD	31	Carrier Detect:เป็นสัญญาณอินพุตของ8273ที่ใช้ในการตรวจจับการเริ่มต้นการส่งผ่านในสาย และ8273 สามารถเริ่มต้นในการสุ่มข้อมูลบนสาย RxD	อินพุต
PA_{2-4}	32 - 34	General purpose Input parts: ระดับ ลอจิกบนสายเหล่านี้สามารถอ่านได้โดย ซีพียู ผ่านบัฟเฟอร์บัสข้อมูล	อินพุต
PB_{1-4}	36 - 39	General purpose Output parts: ซีพียู สามารถเขียนเอาต์พุตลงบนสายเหล่านี้โดยผ่านทางบัฟเฟอร์บัสข้อมูล	อาต์พุต
CLK	3	Clock:สัญญาณนาฬิกา TTL	อินพุต

ส่วนประกอบของชิป 8273

ไอซีเบอร์ 8273 เป็นอุปกรณ์สนับสนุนที่มีความสามารถสูงทางด้านโพรโทคอลชนิด SDLC ดังนั้นภาระต่าง ๆ ที่เกี่ยวข้องกับโพรโทคอลนี้ 8273 จึงรับหน้าที่ไปทำทั้งหมด 8273 จะควบคุมโดยการเขียนคำสั่งลงในรีจีสเตอร์ (register) ต่าง ๆ ซึ่งอยู่ภายใน 8273 ส่วนประกอบภายใน 8273 แสดงเป็นแผนภาพกรอบ ดังรูปที่ 3.1

รีจีสเตอร์ภายในชิป 8273

ภายใน 8273 ประกอบด้วยรีจีสเตอร์ทั้งหมด 7 รีจีสเตอร์ซึ่งแสดงดังรูปที่ 3.1 แต่ละรีจีสเตอร์มีหน้าที่การทำงานดังนี้

1. Command register เป็นรีจีสเตอร์ที่ถูกใช้ในการเขียนชุดคำสั่งที่ใช้ควบคุมการทำงานของ 8273

2. Parameter register คำสั่งบางคำสั่งถูกเขียนลงใน Command register แต่ต้องการข้อมูลเพิ่มเติมจึงต้องถูกเขียนลงใน register นี้

3. Status register ใช้ในการแสดงสถานภาพที่เกิดขึ้นเกี่ยวกับการป้อนคำสั่งหรือการขัดจังหวะ (interrupt)

4. Result register ใช้แสดงผลของ immediate result ซึ่งเกิดจากการใช้คำสั่งเกี่ยวกับไอโอ (I / O) พอร์ตของ 8273

5. TxINT register (Transmit Interrupt Result Register) ใช้แสดงผลลัพธ์ที่เกิดจากขบวนการส่งข้อมูล

6. RxINT register (Receive Interrupt Result Register) ใช้ แสดงผลลัพธ์ที่เกิดจากขบวนการรับข้อมูล

7. Test mode register ใช้ในการ reset 8273 โดยขบวนการทางซอฟต์แวร์ (software)

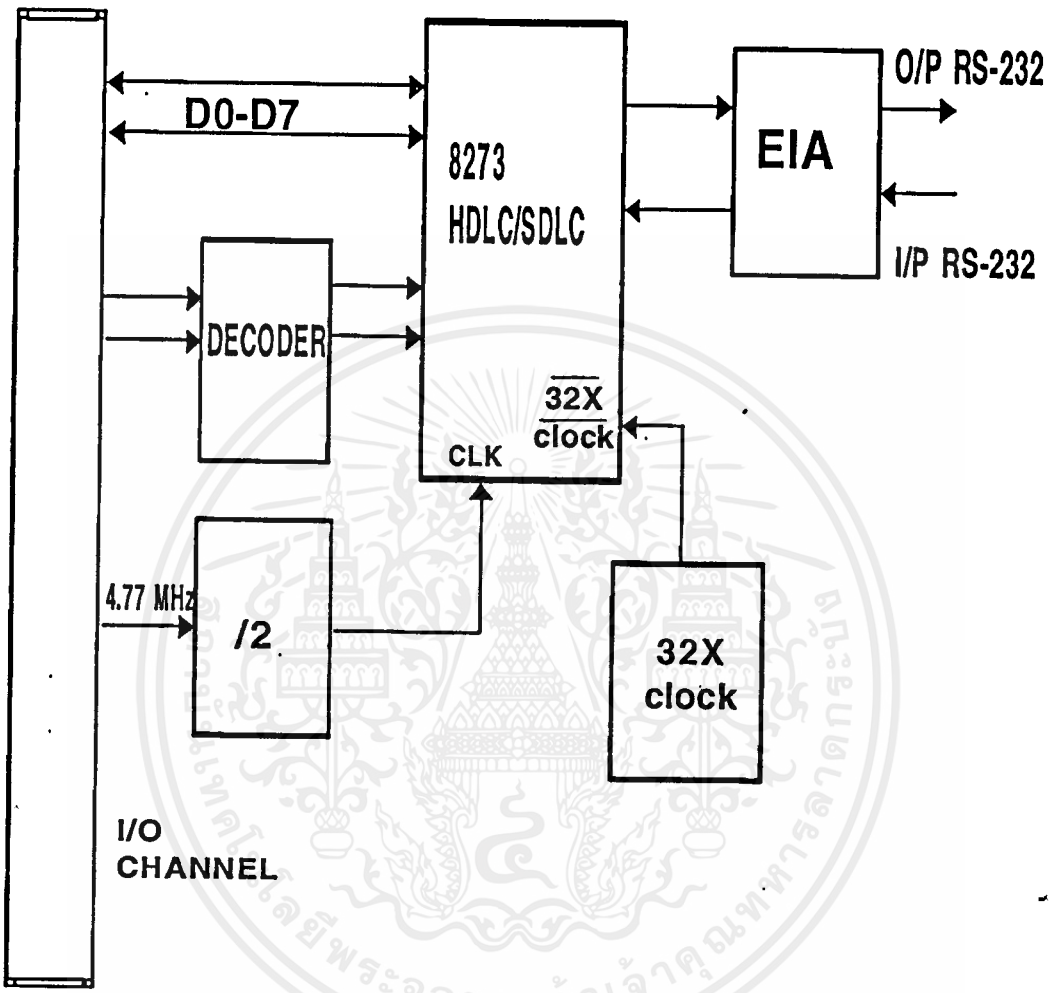
การอ้างถึงรีจีสเตอร์ต่าง ๆ นี้จะอ้างโดยขาแอดเดรส A0 และ A1 ดังตารางที่ 3.2

A1	A0	TxDACK	RxDACK	CS	RD	WR	REGISTER
0	0	1	1	0	1	0	Command Register
0	0	1	1	0	0	1	Status Register
0	1	1	1	0	1	0	Parameter Register
0	1	1	1	0	0	1	Result Register
1	0	1	1	0	1	0	Reset Register
1	0	1	1	0	0	1	TxINT Result Register
1	1	1	1	0	1	0	-
1	1	1	1	0	0	1	RxINT Result Register
x	x	0	1	1	1	0	Transmit Data
x	x	1	0	1	0	1	Receive Data

ตารางที่ 3.2 ตำแหน่งของรีจีสเตอร์

การต่อชิป 8273 เข้ากับระบบไมโครคอมพิวเตอร์

การเชื่อมต่อระบบไมโครคอมพิวเตอร์เข้ากับชิป 8273 เพื่อทำหน้าที่โพรโทคอล HDLC/SDLC ดังแสดงด้วยบล็อกไดอะแกรมในรูปที่ 3.2



รูปที่ 3.2

จากรูปที่ 3.2 ได้แสดงบล็อกไดอะแกรม ของ HDLC / SDLC INTERFACE CARD ซึ่งประกอบด้วย 5 ส่วนหลักดังนี้คือ

- 8273 ซึ่งทำหน้าที่เป็นโพรโทคอล HDLC/SDLC

- DECODER ทำหน้าที่กำหนดหมายเลขพอร์ต อินพุต / เอาต์พุต ให้กับระบบไมโครคอมพิวเตอร์กับ 8273

- / 2 ทำหน้าที่หารความถี่ 4.77 MHz เพื่อใช้ในการควบคุมกำหนดจังหวะในการติดต่อระหว่าง ซีพียู กับ 8273

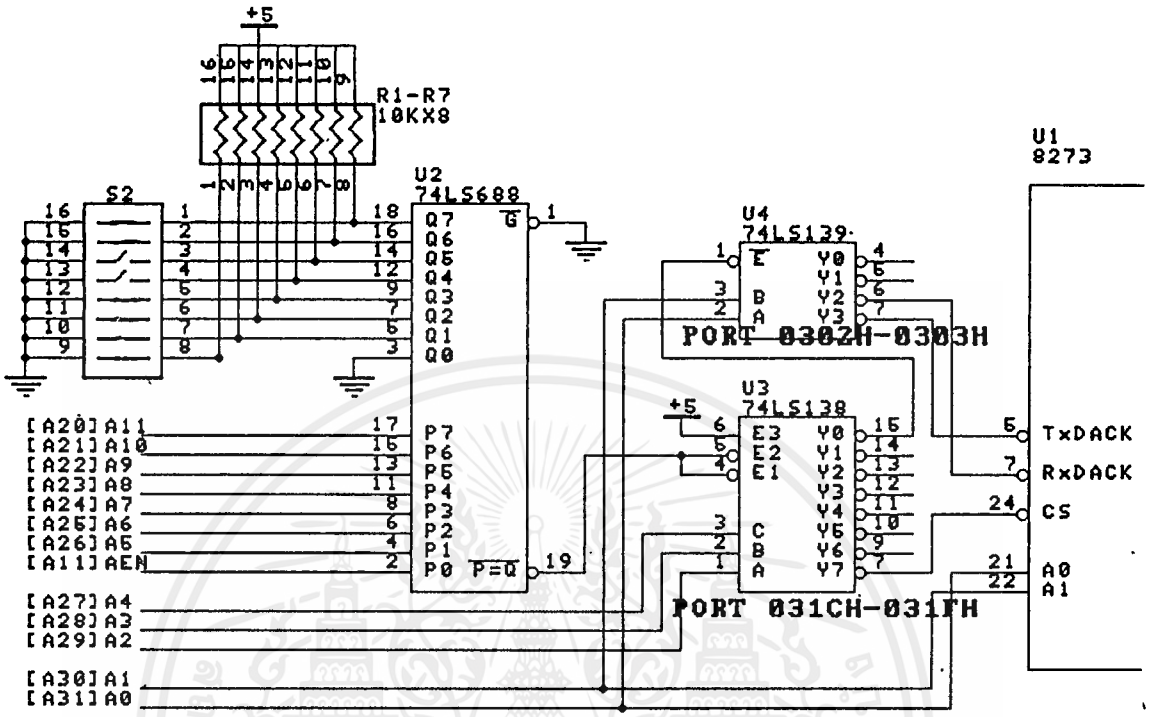
- 32xCLOCK จะผลิตความถี่ 32x ของอัตราการส่งข้อมูลเพื่อกำหนดอัตรา

การส่งข้อมูลและร่วมกับ DPLL ในการกู้ข้อมูลที่ได้รับ

- EIA ปรับระดับ / ลดระดับแรงดัน ตามระดับแรงดันของ RS-232

3.2 การต่อ 8273 เข้ากับ I/O CHANNEL

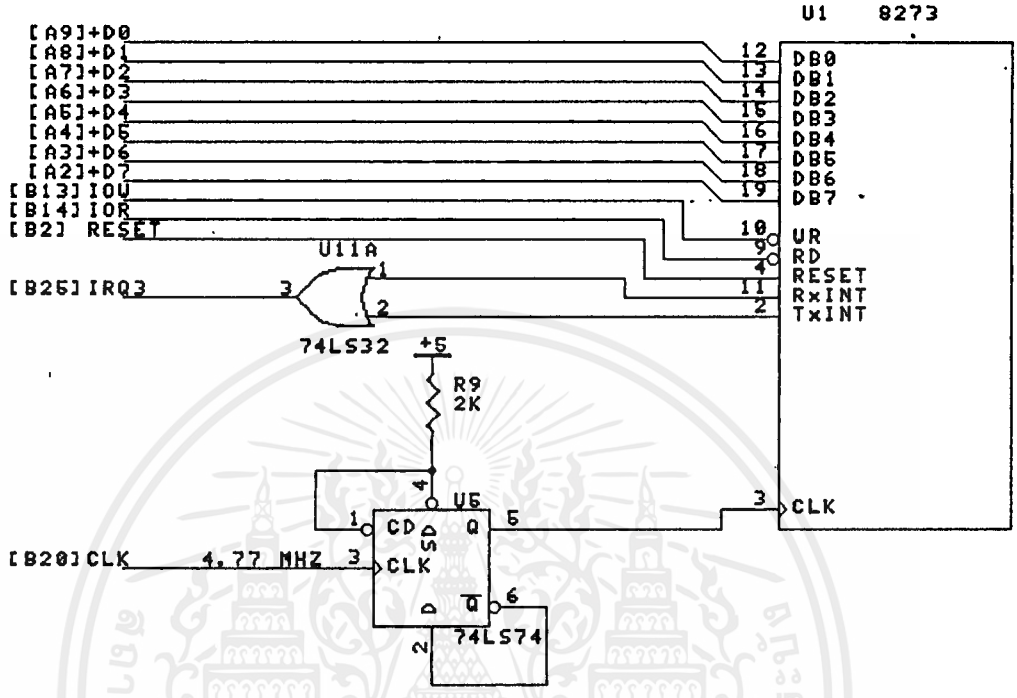
การเลือกชิป 8273 จะผ่านเข้ามาทางสัญญาณ CS ซึ่งได้มาจากการดีโค้ดของ 74LS138 ในการดีโค้ดในที่นี้จะใช้สวิตช์เลือกซึ่งสามารถเปลี่ยนค่าแอดเดรสได้โดยเพียง แต่เปลี่ยนตำแหน่งของสวิตช์ (ในที่นี้คือ DIP switch) ที่เซตไว้ในวงจรเท่านั้น สำหรับหน้าที่ของ 74LS688 นี้จะทำการเปรียบเทียบค่าของอินพุต 2 ชุดที่ถูกส่งเข้ามาทางขา P_0 - P_7 และ Q_0 - Q_7 ถ้าอินพุตทั้ง 2 ชุดเท่ากันแล้ว เอาต์พุตที่ขา P = Q จะให้เอาต์พุตเป็นลอจิก "0" จากในวงจร P_1 - P_7 ของ 74LS688 ต่อกับแอดเดรสบิต A_5 - A_{11} ในขณะที่ขา Q_1 - Q_7 ต่อกับความต้านทานที่ทำหน้าที่ เป็น Pull up (ค่ารักษาระดับแรงดันให้เป็นลอจิก "1" ไว้ในกรณีที่ไม่มีอินพุตเข้ามา)และขา Q_1 - Q_7 นี้จะต่อกับอีกปลายด้านหนึ่งของ DIP switch ด้วยส่วนปลายอีกด้านหนึ่งของ DIP switch นั้นจะต่อลงกราวน์ (ลอจิก "0") ไว้ดังนั้นถ้าเราทำการ "ON" DIP switch ที่ต่อขาใดขานั้นก็จะได้รับลอจิก "0" ในขณะที่ถ้า DIP switch ที่ต่อกับขาใดถูก "OFF" ขานั้นก็จะได้รับลอจิก "1" และเนื่องจากอินพุตที่ขา P_1 - P_7 (แอดเดรส A_5 - A_{11}) ต้องเท่ากับอินพุตที่ขา Q_1 - Q_7 ดังนั้น ถ้าเราเปลี่ยนแปลง การเซ็ต DIP switch เหล่านี้ก็จะทำให้แอดเดรสบิต A_5 - A_{11} ซึ่งต่อกับขา P_1 - P_7 นั้นเปลี่ยนแปลงตามไปด้วยจึงทำให้เอาต์พุตของ 74LS688 แอคตีฟได้ทำให้สามารถเปลี่ยนแปลงค่าแอดเดรสที่ต้องการจะดีโค้ดได้ง่ายกว่า วิธีการดีโค้ดแบบ Fixed ส่วนขา P_0 จะต่อกับสัญญาน AEN โดยขา Q_0 ต่อกับลอจิก " 0 " การต่อในลักษณะนี้ ก็เพื่อป้องกันไม่ให้ 74LS688 ทำการดีโค้ดในระหว่างขบวนการ DMA นับเอาเอาท์พุทจาก M P = Q ของ 74LS688 นี้ จะถูกนำไปใช้ในการ อีนาเบิ้ล 74LS138 ซึ่งในที่นี้ DIP switch ขา 3 และ 4 OFF ซึ่งดีโค้ดร่วมกับ 74LS138 และ 74LS139 ตั้งแต่พอร์ต 0300H ถึง 031FH ซึ่ง 8273 ซึ่ง 8273 จะใช้พอร์ต 0302H - 0303H สำหรับเขียนอ่านข้อมูลและพอร์ต 031CH - 031F H สำหรับเขียนอ่านรีจีสเตอร์ภายในดังแสดงในรูปที่ 3.3



รูปที่ 3.3

3.3 สัญญาณควบคุมและบัสข้อมูล

บัสข้อมูลของ I / O channel สามารถเชื่อมต่อเข้ากับ DB_0 - DB_7 ของ 8273 ได้เลย ส่วนของ 8273 จะติดต่อกับซีพียูได้นั้น โดยอาศัยการ อินเตอร์รัพต์ ซึ่งในวงจรจะนำเอาสัญญาณที่ขา TxINT กับ RxINT มาผ่าน OR Gate ไปเข้า IRQ3 ของระบบสำหรับการจำแนกของสัญญาณ Interrupt อาศัยโดยการตรวจสอบสถานะ Register ภายใน 8273 ในการจำแนกว่าเป็น TxINT หรือ RxINT ในส่วน ชิป 74LS74 ซึ่งทำหน้าที่เป็นวงจรหาร 2 ความถี่ 4.77 MHz เพื่อกำหนดไซเกิลการทำงานของ 8273 ส่วนของสัญญาณ IOW, IOR, RESET นั้นสามารถต่อเข้าโดยตรงกับ I / O channel เพื่อใช้ในการควบคุมการอ่านเขียนพอร์ต และรีเซ็ตระบบซึ่งสายสัญญาณดังกล่าวทั้งหมดได้แสดงในรูปที่ 3.4



รูปที่ 3.4

3.4 สัญญาณที่ติดต่อกับ โมเดม

สัญญาณที่ติดต่อกับโมเดมตามรูปที่ 3.5 ประกอบด้วย

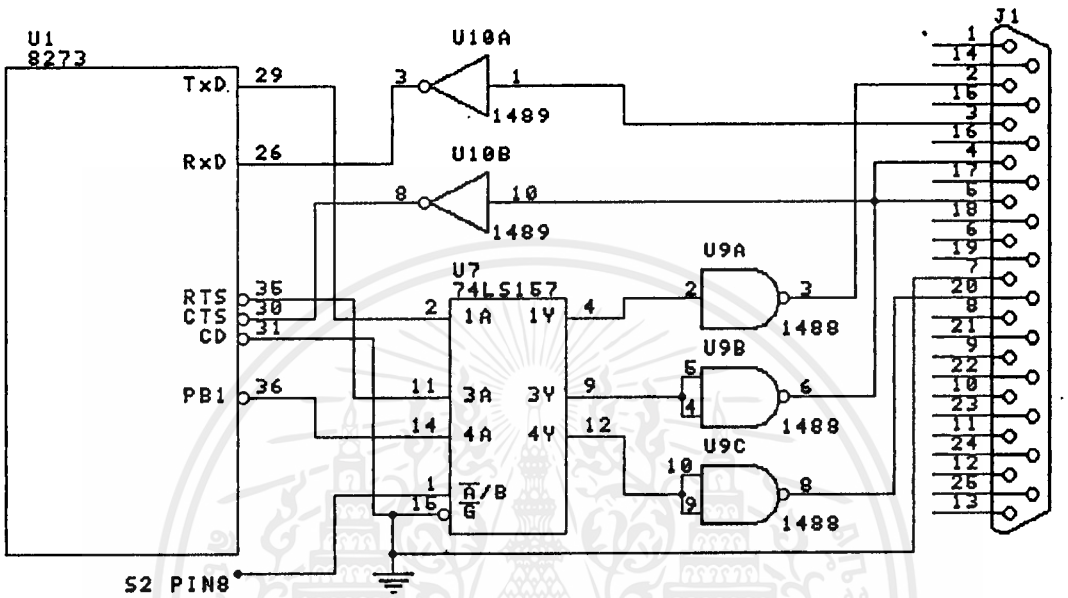
TxD จะส่งผ่านข้อมูลไปยัง U9A เพื่อปรับระดับลอจิกให้ตรงกับมาตรฐาน ของ RS-232

RxD จะรับข้อมูลโดยผ่าน U10A เพื่อ ลดระดับสัญญาณให้ตรง กับระดับลอจิก ของ TTL

CTS เป็นสัญญาณอินพุตจากภายนอก ที่พร้อมจะรับข้อมูลจาก 8273 จะผ่าน U10B เพื่อลดระดับเช่นเดียวกัน

CD จะถูกต่อลง ground เพื่อที่ภาครับจะได้แอคตีฟตลอดเวลา ในกรณีที่เลือก โหมด FLAG แทน IDLE

RTS เป็นสัญญาณเอาท์พุตของ 8273 ที่พร้อมจะส่งข้อมูล จะถูกส่งผ่าน U9B เพื่อปรับระดับลอจิก TTL ให้ตรงกับมาตรฐาน RS-232

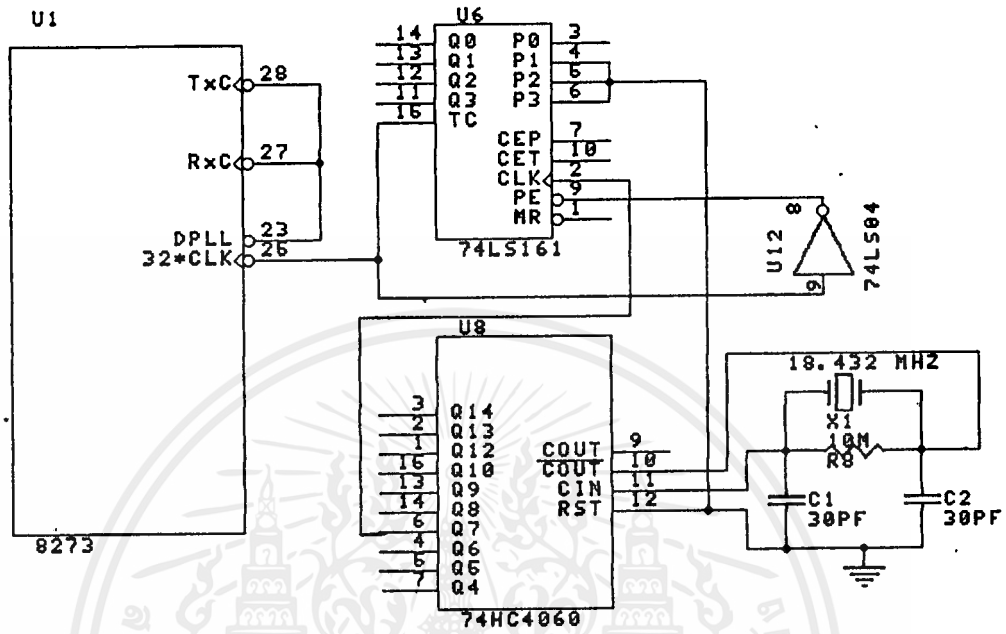


รูปที่ 3.5 แสดงถึงส่วนของสัญญาณที่ติดต่อกับโมเดม

3.5 ส่วนของ 32xCLOCK

ในส่วนนี้จะประกอบด้วย ชิป 74HC4060 ซึ่งทำหน้าที่เป็นวงจรนับไบนารี่และออสซิลเลเตอร์ ซึ่งควบคุมโดย R-C หรือ คริสตอล 18.432 MHz ซึ่งในที่นี้ต้องการอัตราการส่งข้อมูลเท่ากับ 300 ดังนั้น ความถี่ที่จะผลิต 32x300 เท่ากับ 9600 และวงจรนับที่ต้องการคือ $18.432 \times 10^6 / 9600 = 1920 = 128 \times 15$

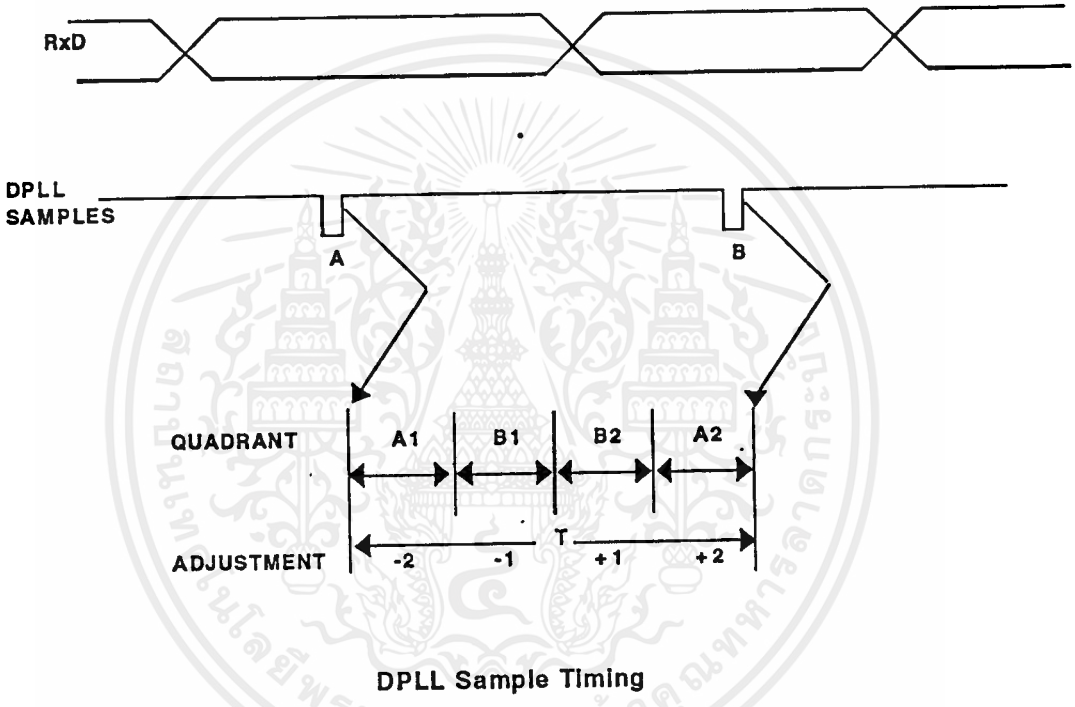
เพราะฉะนั้น วงจรนับ 128 ซึ่งสามารถต่อสายเอาต์พุตได้ที่ขา 6 ของ74HC4060 ส่วนวงจรนับ 15 นั้นอาศัย ชิป 74LS161 ร่วมกับ 74LS04 ทำหน้าที่นับ 15 และเอาต์พุท ขา 15 ของ 74LS161 จะต่อเข้ากับขา 26 ของ 8273 ดังแสดงในรูปที่ 3.6



3.6 <u>การทำงานของ Digital Phase Looked Loop ในการประยุกต์แบบอะซิงโครนัส</u>

การได้มาของคล็อกข้อมูลบนการไหลที่ต่อเนื่องของข้อมูลบนขา RxD ของ 8273 นั้น จะใช้หลักการของ ดิจิตอลเฟสล็อกลูป (DPLL) ซึ่ง DPLL ต้องการคล็อกอินพุตที่ 32x อัตราการส่งข้อมูล ข้อมูลที่ได้รับ (RxD) จะถูกสุ่มด้วย 32xCLK ซึ่งขา DPLL จะผลิตพัลส์สุ่มตัวอย่างบริเวณกึ่งกลางของบิทเซลส์ RxD DPLL มีโครงสร้างภายในที่จะลดความไวที่เกิดจาก line noise และ bit distortion ซึ่งสามารถทำได้โดยการปรับแก้ในการเพิ่มแบบดีสครีสของความคลาดเคลื่อนเฟสในเมื่อพัลส์ที่สร้างระบุให้เกิดที่จำนวนนับ 32 ของ 32xCLK จำนวนนับนี้จะถูกบวกลบขึ้นอยู่กับควอดแรนท์ของขอบข้อมูลที่เกิดขึ้นภายในสี่ควอดแรนท์เฟสที่คลาดเคลื่อน ยกตัวอย่าง เช่น ถ้าขอบ RxD ถูกตรวจพบในควอดแรนท์ที่ A1 ซึ่งจะเห็นได้ว่า DPLL แซมเปิลพัลส์ " A " จะถูกวางอยู่ในตำแหน่งใกล้ขอบขาลงของเซลส์ข้อมูลมาก ดังนั้นแซมเปิลพัลส์ " B " จะถูกวางที่ตำแหน่ง (T nominal - 2 COUNT) = นับ 30 ของ 32xCLOCK ซึ่งเป็นการเลื่อนพัลส์ " B " ไปข้างหน้าในบิทเซลส์ข้อมูลต่อไป ขอบของข้อมูลที่เกิดขึ้นในควอดแรนท์ B1 จะเป็น

สาเหตุให้การแก้ไขทางเฟสน้อยมากด้วยจำนวนนับเท่ากับ 31 ของ 32xCLK โดยการใช้เทคนิคพัลส์ DPLL จะสามารถเข้ายึดบริเวณตรงกลางของบิทเซลส์ข้อมูลภายในเวลา 12 บิทเซลส์ข้อมูล ดังแสดงในรูปที่ 3.7



DPLL Sample Timing

รูปที่ 3.7 แสดง DPLL Sample Timing

จากที่ได้กล่าวมาแล้วในบทนี้ จะเห็นได้ว่า ชิป 8273 ซึ่งทำหน้าที่เป็นโฟรโทคอล HDLC / SDLC มีคุณลักษณะ ไม่ว่าจะเป็นคุณสมบัติทางไฟฟ้า ความสามารถในการต่อร่วมกับอุปกรณ์อื่นที่หาง่าย ตลอดจนความคล่องตัวในการทำงานไม่ว่าจะเป็นการสื่อสารข้อมูลแบบอะซิงโครนัสหรือซิงโครนัส ทำให้สามารถนำมาใช้ประโยชน์ได้อย่างกว้างขวาง และง่ายต่อการออกแบบวงจรในการทำงานร่วมกับระบบไมโครคอมพิวเตอร์

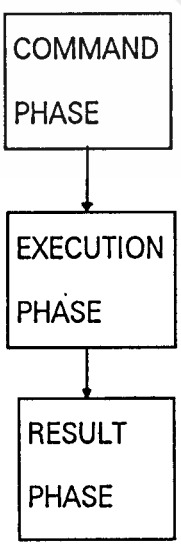
บทที่ 4
ซอฟแวร์ของ 8273

4.1 รูปลักษณ์ซอฟต์แวร์ของ 8273

รูปลักษณ์ซอฟต์แวร์ของ 8273 เกี่ยวข้องกับการสื่อสารในการออกคำสั่งจากซีพียูไปยัง 8273 และการส่งกลับคืนผลลัพธ์ของคำสั่งเหล่านั้นจาก 8273 ไปยังซีพียู เนื่องจากสถาปัตยกรรมโปรเซสเซอร์ภายในของ 8273 การสื่อสารระหว่างซีพียู กับ 8273 จึงเป็นรูปแบบพื้นฐานของการสื่อสารระหว่างโปรเซสเซอร์ โดยปกติการสื่อสารเช่นนี้ ต้องการรูปแบบโฟรโทคอลของตัวมันเอง โฟรโทคอลนี้ได้รับการสนับสนุนโดยผ่านการใช้การให้อาณัติสัญญาณในรีจีสเตอร์สถานะของ 8273 ซึ่งจะได้นำขึ้นมากล่าวในที่นี้

4.2 ลำดับขั้นตอนการทำงานของ 8273

การทำงานของ 8273 ประกอบด้วยขั้นตอนดังต่อไปนี้



ซีพียูจะทำการเขียนคำสั่งและพารามิเตอร์เข้าไปในรีจีสเตอร์คำสั่งและรีจีสเตอร์พารามิเตอร์

8273 เริ่มปฏิบัติการตามคำสั่ง

8273 ส่งสัญญาณผลลัพธ์สิ้นสุดการทำงานไปยังซีพียู ซีพียูจะต้องอ่านผลลัพธ์ของการทำงานจากรีจีสเตอร์ผลลัพธ์

4.3 ซอฟต์แวร์ช่วงสั่งการ (Command Phase Software)

ซีพียู จะเริ่มช่วงสั่งการ โดยการเขียน ไบท์ คำสั่ง เข้าไปใน รีจีสเตอร์ คำสั่ง (Command register) ของ 8273 ถ้า 8273 ต้องการข้อมูลเพิ่มเติมเกี่ยวกับคำสั่งมากกว่านี้ ซีพียู ก็จะเขียนข้อมูลส่วนนี้เข้าไปในรีจีสเตอร์พารามิเตอร์ (Parameter register) รูปที่ 4.3 ได้แสดงแผนผังของช่วงสั่งการ เป็นที่น่าสังเกตุว่า บิท CBSY และ CPBF ของรีจีสเตอร์สถานะ (Staus register) ได้นำมาใช้ในการให้อาณัติสัญญาณกับไบท์คำสั่ง และพารามิเตอร์ นอกจากนี้แผนผังยังได้แสดงถึง คำสั่งไม่สามารถ ปล่อยออกมาได้ถ้ารีจีสเตอร์สถานะของ 8273 แสดงสถานะไม่ว่าง (CBSY=1) ถ้าคำสั่งถูกปล่อยในขณะที่ CBSY = 1 คำสั่งเดิมที่อยู่ในรีจีสเตอร์คำสั่งจะถูกเขียนทับและสูญหาย (คงยังจำได้ว่า บิท CBSY นั้นแสดงถึง ช่วงสังการอยู่ในการรุดหน้าและไม่ใช่การปฏิบัติการที่แท้จริงของคำสั่ง)และแผนผังยังได้รวมการตรวจสอบบัฟเฟอร์พารามิเตอร์เต็ม ซีพียูจะต้องคอยจนกระทั้ง CPBF = 0 ก่อนที่จะเขียนพารามิเตอร์ให้กับรีจีสเตอร์พารามิเตอร์ ถ้าปล่อยพารามิเตอร์ในขณะที่ CPBF= 1 พารามิเตอร์ก่อนหน้านี้จะถูกเขียนทับและเกิดการสูญหายขึ้น 8273 เป็นอุปกรณ์ฟูลดูเพลกซ์ นั่นคือ ภาคส่ง และภาครับทั้งคู่ อาจจะปฏิบัติการคำสั่ง หรือทำการผ่านผลลัพธ์อินเตอร์รัพต์ ณ เวลาที่กำหนดให้ใดๆ (การแยกขาอินเตอร์รัพต์ Rx และ Tx และรีจีสเตอร์ผลลัพธ์ ซึ่งจัดให้สำหรับเหตุผลนี้) ถึงอย่างไรก็ตามรีจีสเตอร์คำสั่มีเพียงตัวเดียวเท่านั้น ดังนั้นรีจีสเตอร์คำสั่งจะถูกใช้ที่ละคำสั่งตามลำดับที่เวลาหนึ่งๆและ ภาคส่งและภาครับไม่เคยที่จะอยู่ในช่วงสั่งการในเวลาเดียวกัน รายละเอียดของคำสั่งและพารามิเตอร์จะได้นำมาบรรยายในส่วนที่ตามมา

4.4 รีจีสเตอร์สถานะ

นิยามบิทของรีจีสเตอร์สถานะได้แสดงในรูปที่ 4.2

CBSY : Command Busy - CBSY แสดงถึง 8273 อยู่ในช่วงสั่งการ CBSY เป็น "1" เมื่อซีพียูเขียนคำสั่งเข้าไปในรีจีสเตอร์คำสั่ง นั้นคือการเริ่มต้นของช่วงสั่งการ มันจะเป็น "0" เมื่อพารามิเตอร์สุดท้ายได้ฝากไว้ในรีจีสเตอร์พารามิเตอร์และได้ยอมรับโดย 8273 ซึ่งเป็นการสิ้นสุดของช่วงสั่งการ

CBF : Command Buffer Full - เมื่อบิทนี้ เซ็ต (1) หมายถึง ภายในรีจีสเตอร์คำสั่งขณะนี้มีข้อมูลอยู่ 1 ไบท์ โดยปกติแล้วบิทนี้จะไม่นำมาใช้

CPBF : Command Parameter Buffer Full - บิท นี้แสดงถึง พารามิเตอร์ที่บรรจุอยู่ในรีจีสเตอร์พารามิเตอร์ CPBF เป็น " 1 " เมื่อ ซีพียูฝากพารามิเตอร์ไว้ในรีจีสเตอร์พารามิเตอร์และจะเป็น "0" เมื่อ 8273 รับพารามิเตอร์

CRBF : Command Result Buffer Full - บิทนี้เป็น " 1 " เมื่อ 8273 วางผลลัพธ์จากการกระทำคำสั่งประเภทอิมมิเดียตในรีจีสเตอร์ผลลัพธ์มันจะเป็น " 0 " เมื่อซีพียูอ่านผลลัพธ์จากรีจีสเตอร์ผลลัพธ์

RxINT : Receiver Interrupt - บิทนี้จะสะท้อนสถานะของขา RxINT เมื่อไรก็ตามที่ภาครับต้องการจะใช้บริการ RxINT จะเป็น " 1 " โดย 8273 และจะเป็น " 0 " เมื่อ ซีพียูอ่านผลลัพธ์ หรือ กระทำการถ่ายโอนข้อมูล

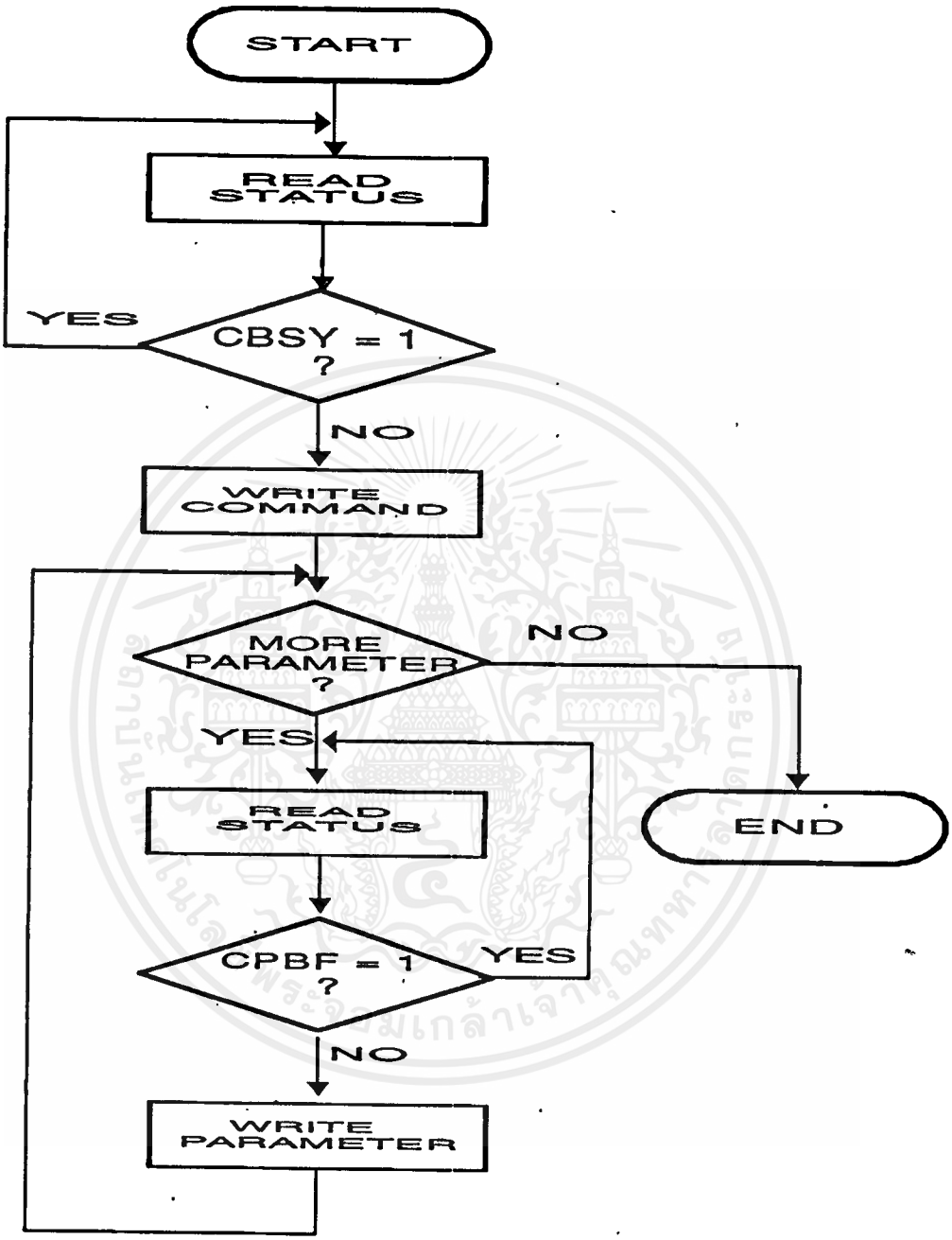
TxINT : Transmitter Interrupt - บิท นี้จะเหมือนกันกับ RxINT ยกเว้น การกระทำเริ่มที่ขา TxINT

RxIRA : Receiver Interrupt Result Available - RxINT เป็น "1" เมื่อ 8273 วางไบท์ผลลัพธ์เข้าไปในรีจีสเตอร์ RxI/R และเป็น " 0 " เมื่อ ซีพียูอ่าน ผลลัพธ์จากรีจีสเตอร์ RxI/R

TxIRA : Transmitter Interrupt Result Available - TxIRA เป็น บิทที่สอดคล้องกับผลลัพธ์ที่สามารถหาได้สำหรับภาคส่ง มันจะเป็น " 1 " เมื่อ 8273 ว่าง ไบท์ผลลัพธ์อินเตอร์รัพต์ในรีจีสเตอร์ TxI/R และจะเป็น " 0 " . เมื่อซีพียูอ่านรีจีสเตอร์ ความสำคัญของแต่ละบิทเหล่านี้จะเป็นที่เด่นชัดในเร็วๆ นี้เมื่อความต้องการซอฟต์แวร์แต่ละช่วงของ 8273 เป็นสิ่งสำคัญที่ต้องการอิสระ แต่ละช่วงถูกแยกครอบคลุม

D7	D6	D5	D4	D3	D2	D1	D0
CBSY	CBF	CPBF	CRBF	RxINT	TxINT	RxIRA	TxIRA

รูปที่ 4.2 รูปแบบรีจีสเตอร์ S STATUS



Command Phase Flowchart

รูปที่ 4.3 แผนผังของช่วงสั่งการ

4.4 ซอฟต์แวร์ช่วงการปฏิบัติการ (Execution Phase Software)

ในระหว่างของช่วงการปฏิบัติการการทำงานตามระบุโดยช่วงสั่งการได้ดำเนินการ ถ้าระบบใช้การถ่ายโอนข้อมูลแบบ DMA จะไม่มีผลต่อ ซีพียูในระหว่างช่วงนี้ และไม่ต้องการซอฟต์แวร์อีกด้วย ถ้าใช้การถ่ายโอนข้อมูลแบบ non-DMA ไม่ว่าจะเป็นแบบอินเตอร์รัพต์ หรือ โพลลิ่ง ต้องการสัญญาณในการถ่ายโอนข้อมูลสำหรับการถ่ายโอนข้อมูลแบบ การขับอินเตอร์รัพต์ 8273 จะสร้างสัญญาณขึ้นบนขาที่ได้จัดไว้ให้เฉพาะ เมื่อทำการตอบสนองต่ออินเตอร์รัพต์แล้ว ซีพียูต้องกำหนดลงไปว่า มันเป็นการเรียกร้องถ่ายโอนข้อมูลหรือเป็นการสัญญาณอินเตอร์รัพต์สิ้นสุด การทำงานหรือผลลัพธ์ที่มีได้ การกำหนดสาเหตุเหล่านี้โดยซีพียูทำการอ่านรีจีสเตอร์ สถานะและสอบถามบิท IRA (Interrupt Result Available) ที่เกี่ยวเนื่อง (TxIRA สำหรับ TxINT และ RxIRA สำหรับ RxIRA) ถ้า IRA = 0 อินเตอร์รัพต์ เป็น การเรียกร้องการถ่ายโอนข้อมูล ถ้า IRA = 1 เป็นการสิ้นสุดการทำงานและรีจีสเตอร์ ผลลัพธ์อินเตอร์รัพต์ที่เกี่ยวเนื่องด้วย จะถูกอ่านเพื่อกำหนดสถานะการสิ้นสุดการทำ งาน (ถูกต้อง/ผิดพลาด/และอื่นๆ) เมื่อใช้การถ่ายโอนข้อมูลแบบโพลลิ่ง รูทีนโพลลิ่ง จะอ่านรีจีสเตอร์สถานะเฝ้ามอง บิท INT บิท ใดบิทหนึ่งเซ็ต เมื่อพบบิท INT เซ็ตบิท IRA ที่สอดคล้องจะถูกพิจารณา ซึ่งเหมือนกับกรณีของการขับอินเตอร์รัพต์ ถ้า IRA = 0 เป็นความต้องการถ่ายโอนข้อมูล ถ้า IRA = 1 เป็นการสิ้นสุดการทำงาน และผลลัพธ์จะถูกอ่านจาก รีจีสเตอร์ผลลัพธ์อินเตอร์รัพต์

4.5 ซอฟต์แวร์ช่วงผลลัพธ์ (Result Phase Software)

ในระหว่างช่วงผลลัพธ์ 8273 จะแจ้งผลลัพธ์ของคำสั่งไปยังซีพียู ซึ่งเริ่มขึ้นโดยการทำงาน ที่ต่อเนื่อง ได้สิ้นสุดลงหรือตรวจพบความผิดพลาดในระหว่างปฏิบัติการ คำสั่งบางคำสั่งอย่างเช่น คำสั่งการอ่านหรือเขียนพอร์ท อินพุท/เอาท์พุท จัดให้ผลลัพธ์โดยตรงนั้นคือ ไม่มีการหน่วงจากการออกคำสั่ง และผลลัพธ์ที่มีได้ ส่วนคำสั่งอื่นอย่างเช่น คำสั่งการส่งเฟรม เวลาสิ้นสุดนั้นผลลัพธ์ไม่ได้จัดให้โดยตรง รีจีสเตอร์ผลลัพธ์ต่าง ๆ เหล่านี้จัดให้กับการจำแนกประเภทคำสั่งทั้งสอง และหลีกเลี่ยงการจัดการอินเตอร์รัพต์สำหรับผลลัพธ์ง่าย ๆ คำสั่งประเภทนอนอิมมิเดียตทั้งหมด จัดการ

เกี่ยวกับ ภาคส่ง หรือ ภาครับ ผลลัพธ์ จากคำสั่งเหล่านี้ได้จัดให้ในรีจีสเตอร์ TxI/R (Transmit Interrupt Result) และ RxI/R (Receive Interrupt Result) ตามลำดับ และถูกลำเลียงไปยังซีพียูโดย บิท TxIRA และ RxIRA ของรีจีสเตอร์สถานะ ผลลัพธ์คำสั่งประเภทนอนอิมมิเดียตประกอบด้วย รหัสอินเตอร์รัพต์ผลลัพธ์ 1 ไบท์ ที่แสดงเงื่อนไข สำหรับอินเตอร์รัพต์ และ จัดส่งข้อมูลเพิ่ม 1 ไบท์ หรือมากกว่า ถ้าต้องการรายละเอียด รหัส อินเตอร์รัพต์ และ จุดมุ่งหมายของผลลัพธ์ ได้ครอบคลุมในประเภทคำสั่งรูปที่ 4.7 และรูปที่ 4.8 ได้แสดงแผนผังช่วงผลลัพธ์ของ ภาคส่ง และภาครับ ตามลำดับ

4.6 ลักษณะคำสั่ง 8273

รายละเอียดแต่ละคำสั่งได้นำขึ้นมาอธิบายในส่วนนี้สัญลักษณ์ย่อในลำดับกรุณาอ้างอิงรหัสคำสั่งได้ในตารางที่ 1 8273 ใช้ประโยชน์จากคำสั่งถึง 5 ชนิดด้วยกัน คือ คำสั่งการเริ่มต้นหรือรูปลักษณ์ ,รับ , ส่ง , รีเซ็ต , และควบคุมโมเด็ม

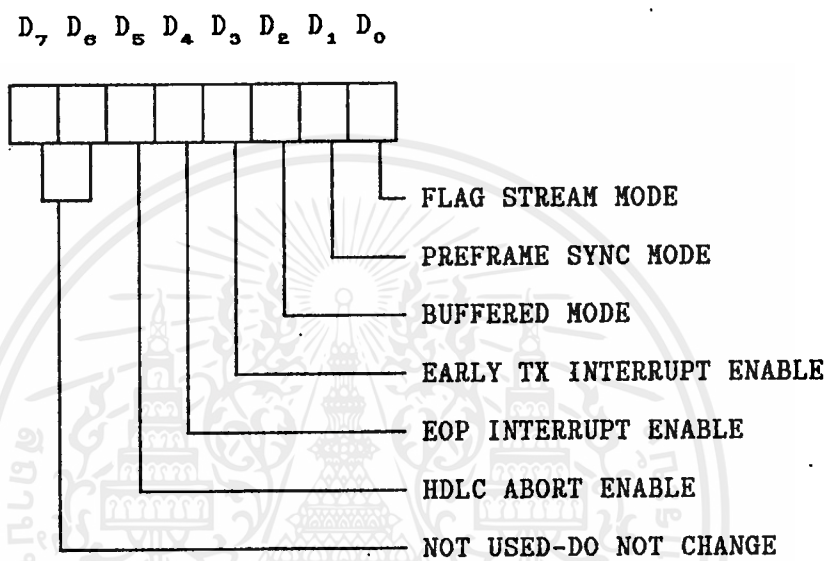
4.7 คำสั่งการเริ่มต้นหรือรูปลักษณ์

คำสั่งการเริ่มต้นหรือรูปลักษณ์ต่างๆ จะจัดการกับรีจีสเตอร์ภายในเพื่อให้ 8273 กำหนดโหมดการทำงาน คำสั่งเหล่านี้เป็นการเซ็ต หรือ รีเซ็ตบิทที่ระบุในรีจีสเตอร์ซึ่งขึ้นอยู่กับชนิดของคำสั่งและต้องการ 1 พารามิเตอร์ คำสั่งเซ็ตจะกระทำการทำงานแบบลอจิก OR ของพารามิเตอร์ (mask) และภายในรีจีสเตอร์ ซึ่ง mask นี้บรรจุ " 1 " ตำแหน่งบิทรีจีสเตอร์ที่ต้องการเซ็ต และ " 0 " ใน mask จะไม่มีการเปลี่ยนแปลงบิทในรีจีสเตอร์ คำสั่งรีเซ็ตจะกระทำการทำงานแบบลอจิก AND ของพารามิเตอร์ (mask) และภายในรีจีสเตอร์นั้นคือ mask ที่เป็น " 0 " จะรีเซ็ตบิทรีจีสเตอร์และ " 1 " จะไม่มีการเปลี่ยนแปลงก่อนที่จะอธิบายคำสั่งนิยามบิทรีจีสเตอร์จะได้นำขึ้นมากล่าวรีจีสเตอร์โหมด ทำงาน

4.8 รีจีสเตอร์โหมด ทำงาน (Operating Mode Register) รูปที่ 4.4

D7 - D6 : Not Used- บิทเหล่านี้ต้องไม่ได้รับจัดการโดยคำสั่งใดๆ นั้นคือ D7 - D6 ต้องเป็น " 0 " สำหรับคำสั่งเซ็ต และ " 1 " สำหรับ คำสั่งรีเซ็ต

D5 : HDLC Abort- เมื่อบิทนี้เป็น เซ็ต 8273 จะอินเตอร์รัพต์โดยภาครับแอคตีฟ เมื่อรับ "01111111" (HDLC Abort) เมื่อ รีเซ็ต SDLC Abort (011111111) จะเป็นสาเหตุการอินเตอร์รัพต์



รูปที่ 4.4 Operating Mode Register

D4 : EOP Interrupt- เมื่อบิทนี้เป็น เซ็ต การรับอักขระ EOP (011111111) จะเป็นสาเหตุให้ 8273 อินเตอร์รัพต์ซีพียูสถานีควบคุมลูปจะใช้โหมดนี้เป็นสัญญาณสอบถามการครบลูปของเฟรมเมื่อบิทนี้เป็น รีเซ็ต จะไม่มี EOP อินเตอร์รัพต์กำเนิด

D3 : Early Tx Interrupt- . บิทนี้กำหนดเวลาซึ่งภาคส่งจะ กำจัดสัญญาณอินเตอร์รัพต์ท้ายเฟรม ถ้าบิทนี้เป็นเซ็ต สัญญาณอินเตอร์รัพต์ถูกกำเนิดขึ้นเมื่ออักขระข้อมูลตัวสุดท้ายได้มีการส่งผ่านไปยัง 8273 ถ้าซอฟต์แวร์ของผู้ใช้สามารถปล่อยคำสั่งส่งของอีกคำสั่งหนึ่งภายในเวลา 2 ไบท์ สัญญาณอินเตอร์รัพต์ท้ายแฟล็กจะไม่เกิดและเฟรมใหม่จะถูกส่งต่อโดยใช้แฟล็กเพียง 1 แฟล็กเท่านั้นที่จะแยกถ้าการกำหนดนี้ไม่พบการแยกระหว่างเฟรมจะมากกว่า 1 แฟล็ก และสัญญาณอินเตอร์รัพต์ท้ายเฟรมจะถูกกำเนิดหลังแฟล็กปิด ถ้ารีเซ็ตบิทนี้จะมีเฉพาะ สัญญาณอินเตอร์รัพต์สิ้นสุด

เฟรมเกิด เมื่อเซ็ตนี้จะยินยอมให้เฉพาะมีแฟล็กเดี่ยวเท่านั้นที่แยกเฟรมติดกัน

D2 : Buffered Address and Control- เมื่อ เซ็ต ฟิลด์ ควบคุม และ แอดเดรสของเฟรมที่รับจะถูกบัฟเฟอร์ไว้ใน 8273 และผ่านไปยังซีพียูตามผลลัพธ์ภายหลังจากที่อินเตอร์รัพต์เฟรมที่ได้รับ (ฟิลด์เหล่านี้ไม่ได้ถ่ายโอนไปยังหน่วยความจำ กับฟิลด์ข้อมูล) ทางด้านส่งฟิลด์ A และ C ผ่านไปยัง 8273 ในฐานะพารามิเตอร์ โหมดนี้ทำให้ง่ายในการบริหารบัฟเฟอร์ เมื่อรีเซ็ต บิทนี้ ฟิลด์ A และ C จะถูกผ่านไปยังหรือออกจากหน่วยความจำเป็น 2 ไบท์แรกของการถ่ายโอนข้อมูล

D1 : Preframe Sync- เมื่อ เซ็ต 8273 จะนำหน้าการส่งเฟรมแต่ละครั้งด้วยอักขระ 2 ตัวก่อนแฟล็กเปิด อักขระทั้ง 2 จัดให้ 16 การเปลี่ยนแปลงในการยินยอมให้เกิดการซิงโครนัสกับภาครับตรงข้ามเพื่อเป็นที่รับประกัน 16 การเปลี่ยนแปลง ดังนั้นอักขระทั้ง 2 คือ 55H - 55H สำหรับโหมด non - NRZI หรือ 00H - 00H สำหรับโหมด NRZI เมื่อ รีเซ็ต จะไม่มีอักขระนำหน้าส่ง

D0 : Flag Stream- เมื่อ เซ็ต ภาคส่งเริ่มทำการส่งอักขรแฟล็กเร็วพอกับที่มันเป็นไอเดิล นั่นคือ ทันทีที่ไอเดิลเมื่อคำสั่งถูกปล่อยหรือหลังการส่งโดยภาคส่งแอคตีฟ เมื่อ รีเซ็ต ภาคส่งจะเริ่มส่งอักขระไอเดิลเหนือบริเวณเขตแดนอักขระตัวต่อ ไปถ้าไอเดิลพร้อม หรือ สิ้นสุดการส่ง ถ้าแอคตีฟ

ตารางที่ 4.1 COMMAND SUMMARY KEY

B0 , B1	- LSB AND MSB OF RECEIVE BUFFER LENGTH
R0 , R1	- LSB AND MSB OF RECEIVED FRAME LENGTH
L0 , L1	- LSB AND MSB OF TRANSMIT FRAME LENGTH
A1 , A2	- MATCH ADDRESSES FOR SELECTIVE RECEIVE
RIC	- RECEIVER INTERRUPT RESULT CODE
TIC	- TRANSMITTER INTERRUPT RESULT CODE
A	- ADDRESS FIELD OF RECEIVED FRAME
C	- CONTROL FIELD OF RECEIVED FRAME

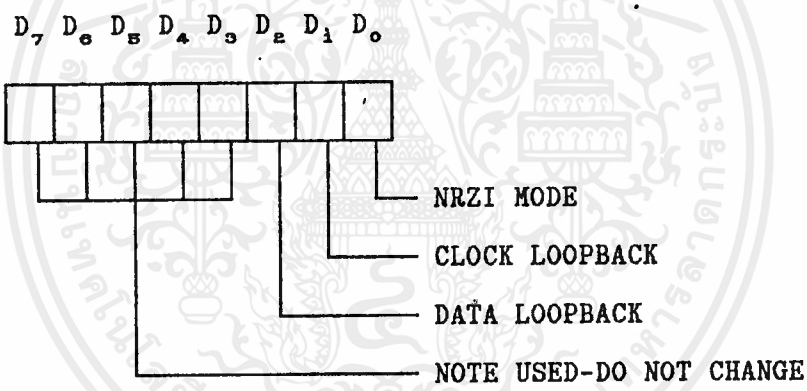
4.9 <u>รีจีสเตอร์โหมดอนุกรม I/O (Serial I/O Mode Register)</u>

D7 - D3 : Not used- บิทเหล่านี้จะต้องเป็น "0" สำหรับคำสั่งเซ็ต และ "1" สำหรับคำสั่งรีเซ็ต

D2 : Data Loopback- เมื่อ"เซ็ต"ข้อมูลที่ส่ง (TxD) ถูกเชื่อมโยงเส้นทางภายในกับวงจรข้อมูลด้านรับ เมื่อ "รีเซ็ต" TxD และ RxD จะเป็นอิสระ

D1 : Clock Loopback- เมื่อ"เซ็ต" TxC ถูกเชื่อมโยงเส้นทางภายในกับ RxC เมื่อ"รีเซ็ต" สัญญาณนาฬิกาจะเป็นอิสระ

D0 : NRZI (Non - Return to Zero Inverted)- เมื่อ"เซ็ต" 8273 จะถือว่าข้อมูลที่รับได้นั้นเป็นรหัส NRZI และเข้ารหัส NRZI ข้อมูลที่ส่ง

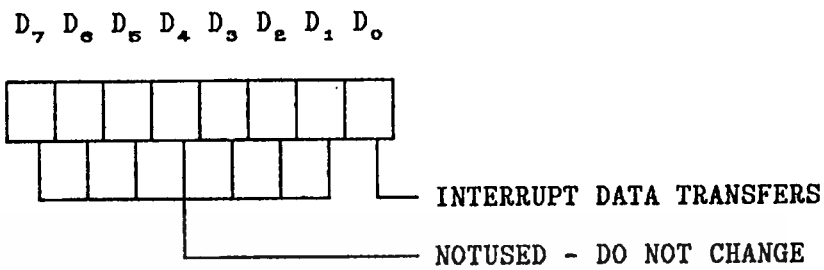


รูปที่ 4.5 Serial I/O Mode Register

4.10 <u>รีจีสเตอร์โหมดถ่ายโอนข้อมูล (Data Transfer Mode Register)</u>

D7 - D1 : Not used- บิท เหล่านี้จะต้องเป็น "0" สำหรับคำสั่งเซ็ตและ " 1 " สำหรับคำสั่งรีเซ็ต

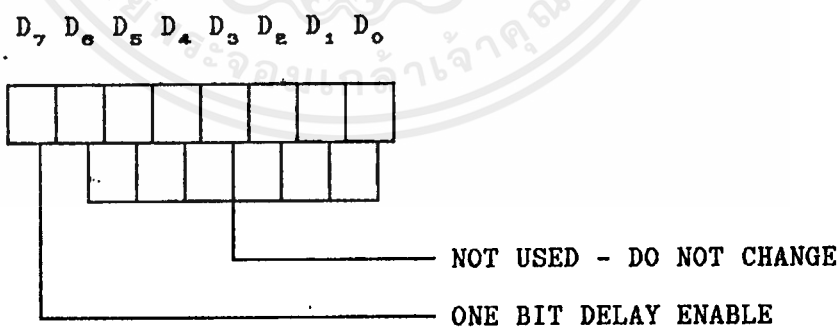
D0 : Interrupt Data Transfer- เมื่อ"เซ็ต 8273 จะอินเตอร์รัพต์ซีพียูเมื่อมีความต้องการถ่ายโอนข้อมูล (สอดคล้องกับบิทรีจีสเตอร์สถานะ IRA = 0 บอกถึงอินเตอร์รัพต์การถ่ายโอนข้อมูลมากกว่าจะเป็นอินเตอร์รัพต์ช่วงผลลัพธ์) เมื่อ "รีเซ็ต" การถ่ายโอนข้อมูลดำเนินการโดยผ่านการร้องขอ DMA บนขา DRQ ปราศจากอินเตอร์รัพต์ซีพียู



4.11 <u>รีจีสเตอร์ดีเลย์ 1 บิท (One Bit Delay Register)</u>

D7 : One Bit Delay- เมื่อ "เซ็ต" 8273 จะทำการส่งข้อมูลต่อเนื่องที่รับได้ใหม่โดยการหน่วง 1 บิท โหมดนี้จะเริ่มและออกจากบริเวณเขตอักขระที่ได้รับเมื่อ "รีเซ็ต" ข้อมูลที่ส่ง และรับ จะอิสระ โหมดนี้ใช้ประโยชน์สำหรับการทำงานแบบลูปและจะได้นำขึ้นมากล่าวในส่วนต่อมา

D6 - D0 : Not Used- บิทเหล่านี้จะเป็น "0" สำหรับคำสั่งเซ็ต และเป็น "1" สำหรับคำสั่งรีเซ็ต



รูปที่ 4.7 One Bit Delay Mode Register

ตารางที่ 4.2 ได้แสดงคำสั่งเซ็ตและรีเซ็ตที่เกี่ยวข้องกับรีจีสเตอร์ข้างบน mask ซึ่ง"เซ็ต"หรือ"รีเซ็ต"บิทที่ต้องการถือเป็นพารามิเตอร์เดี่ยว คำสั่งเหล่านี้ไม่มีการอินเตอร์

รัพต์หรือจัดให้ผลลัพธ์ในระหว่างช่วงผลลัพธ์ ภายหลังจากรีเซ็ต 8273 บิทเหล่านี้จะรีเซ็ตโดยปริยาย

REGISTER	COMMAND	HEX CODE	PARAMETER
ONE BIT DELAY MODE	SET	A4	SET MASK
	RESET	64	RESET MASK
DATA TRANSFER MODE	SET	97	SET MASK
	RESET	57	RESET MASK
OPERATING MODE	SET	91	SET MASK
	RESET	51	RESET MASK
SERIAL I/O MODE	SET	A0	SET MASK
	RESET	60	RESET MASK

ตารางที่ 4.2 สรุปคำสั่ง Initialization/Configuration

4.12 <u>คำสั่งรับ (Receive Commands)</u>

8273 สนับสนุนคำสั่งรับข้อมูลถึง 2 คำสั่งกับฟังก์ชั่นดีสเอเบิลภาครับ 1 ฟังก์ชั่น

4.13 <u>คำสั่งรับทั่วไป (General Receive Command)</u>

เมื่อคำสั่ง ตรงกับ คำสั่งรับทั่วไป 8273 จะผ่านเฟรม ทั้งหมด ไปยังหน่วยความจำ (โหมด DMA) หรือ ซีพียู (โหมด non - DMA) โดยไม่สนใจข้อมูลที่บรรจุในฟิลด์แอดเดรสของเฟรม คำสั่งนี้ เป็นคำสั่งที่ใช้กับสถานีปฐมภูมิ และ สถานีควบคุมลูป ต้องการ 2 พารามิเตอร์ คือ B0 และ B1 พารามิเตอร์เหล่านี้ คือ LSB และ MSB ของขนาดบัฟเฟอร์ภาครับการกำหนดให้ 8273 เช่นนี้จะช่วยลดภาระซีพียูในการตรวจสอบการไหลของข้อมูลล้นบัฟเฟอร์ 8273 จะอินเตอร์รัพต์ซีพียู ถ้าเฟรมที่ได้รับพยายามที่จะใส่ในที่ว่างบัฟเฟอร์มากเกินไป

4.14 คำสั่งรับแบบเจาะจง (Selective Receive Command)

คำสั่งรับแบบเจาะจง ต้องการพารามิเตอร์เพิ่มนอกเหนือจาก B0 และ B1 อีก 2 พารามิเตอร์ คือ A1 และ A2 พารามิเตอร์เหล่านี้เป็น 2 ไบท์แมตช์แอดเดรสเมื่อคำสั่งเป็นการรับแบบเจาะจง 8273 จะผ่านเฉพาะเฟรมที่มีฟิลด์แอดเดรสแมตช์กับ A1 หรือ A2 ไปยังหน่วยความจำหรือซีพียู โดยปกติแล้ว คำสั่งนี้ใช้กับสถานีทุติยภูมิโดย A1 เป็นแอดเดรสทุติยภูมิ และ A2 เป็นแอดเดรสของกลุ่มถ้าต้องการไบท์แมตช์เฉพาะเพียง 1 ไบท์ A1 และ A2 ต้องเท่ากันเช่นเดียวกับคำสั่งรับทั่วไป 8273 จะนับไบท์ข้อมูลที่เข้ามาและอินเตอร์รัพต์ ซีพียูถ้ามากเกิน B0, B1 คำสั่งทั้งสองจัดให้ผลลัพธ์อินเตอร์รัพต์ เมื่อเฟรมที่ได้รับปราศจากความผิดพลาดนั่นคือ FCS ถูกต้องและ CD (Carrier Detect) แอคตีฟตลอดเฟรมหรือไม่พยายามที่จะทำให้บัฟเฟอร์ถูกใส่มากเกินไป 8273 จะอินเตอร์รัพต์ซีพียูตามหลังแฟล็กปิดเพื่อผ่านผลลัพธ์ความสำเร็จในลำดับผลลัพธ์เหล่านี้ คือ รหัสผลลัพธ์อินเตอร์รัพต์ภาครับ (RIC) และไบท์ความยาวฟิลด์ข้อมูลของเฟรม ที่ได้รับ (R0 , R1) ถ้าเลือกโหมด บัฟเฟอร์ฟิลด์แอดเดรสและควบคุมจะถูกผ่านเพิ่มในฐานะผลลัพธ์ถ้าโหมดบัฟเฟอร์ไม่ได้ รับเลือกฟิลด์แอดเดรสและควบคุมจะถูกผ่านในฐานะถ่ายโอนข้อมูล และ R0 , R1 จะสะท้อนความยาวฟิลด์ข้อมูลเพิ่มอีกสอง

4.15 คำสั่งดีสเอเบิลการรับ (Receive Disable Command)

ภาครับสามารถทำให้ดีสเอเบิลได้โดยการใช้คำสั่ง ดีสเอเบิล การรับ คำสั่งนี้จะสิ้น สุดการทำงานการรับทันที ไม่ต้องการพารามิเตอร์ และ ไม่มีผลลัพธ์ส่งกลับ รายละเอียดสำหรับคำสั่งรับได้แสดงในรูปที่ 4.8 กุญแจรหัสผลลัพธ์ได้แสดงในรูปที่ 4.9 การอธิบายของ รหัสผลลัพธ์ เหล่านี้ได้จัดไว้เฉพาะบางอย่างที่เหมาะสม รหัสผลลัพธ์อินเตอร์รัพต์ในรีจีสเตอร์ RxI/R จะเป็นไบท์แรกที่ถูกผ่านไปให้ซีพียู ในระหว่างช่วงผลลัพธ์ บิท D4 - D0 จะกำหนดสาเหตุการอินเตอร์รัพต์ของภาครับ ในเมื่อแต่ละรหัสผลลัพธ์มีส่วนเกี่ยวข้องเฉพาะ ซึ่งจะได้นำขึ้นมา อธิบายแต่ละ รหัสดังข้างล่างนี้

COMMAND	HEX CODE	PARAMETER	RESULT RxI/R
GENERAL RECEIVE	C0	B0,B1	RIC , R0 , R1 , A , C
SELECTIVE RECEIVE	C1	B0 , B1 , A1 , A2	RIC , R0 , R1 , A , C
SELECTIVE LOOP RECEIVE	C2	B0 , B1 , A1 , A2	RIC , R0 , R1 , A , C
DISABLE RECEIVER	C5	NONE	NONE

รูปที่ 4.8 สรุปคำสั่งภาครับ

รหัสผลลัพธ์ 2 อันดับแรก เป็นผลลัพธ์จากการรับเฟรมที่ error - free หลังจาก คำสั่งรับทั่วไป (General receive command) แล้วถ้าเฟรมรับได้อย่างถูกต้องผลลัพธ์แรกจะถูกส่งกลับ ถ้าใช้คำสั่งแบบเจาะจง (Selective receive command) ไม่ว่าจะเป็นแบบธรรมดาหรือลูป รหัสผลลัพธ์แรกจะกำเนิดถ้าแมตช์กับ A1 และรหัสผลลัพธ์ที่สองจะกำเนิดถ้าแมตช์กับ A2 ในแต่ละกรณี ภายหลังจากอินเตอร์รัพต์ภาครับยังคงแอคตีฟ อย่างไรก็ตามตัวนับขนาดบัฟเฟอร์ภายในไม่ได้รีเซ็ตถ้าคำสั่งรับบอก 100 ไบท์ นั้นคือ เป็นการจัดสรรให้กับบัฟเฟอร์รับ (B0 , B1) และ 1 เฟรม 80 ไบท์ รับได้อย่างถูกต้อง ค่าสูงสุดขนาดของเฟรมต่อไปที่สามารถรับได้โดยปราศจากการใช้คำสั่ง ภาครับใหม่ (การ รีเซ็ต B0 และ B1 คือ 20 ไบท์ ดังนั้นในทางปฏิบัตินิยมใช้คำสั่งภาครับใหม่หลังจากการรับ เฟรมแต่ละครั้ง โดยปกติแล้ว พอยต์เตอร์ DMA หรือหน่วยความจำ จะได้รับข้อมูลใหม่ๆ ณ เวลานี้ (จะเห็นได้ว่าผู้ใช้นั้นแกไม่ต้องการถือเอาข้อได้เปรียบลักษณะการจัดการบัฟเฟอร์ ของ 8273 เช่นนี้ อาจจะใช้แบบง่ายๆ B0 , B1 = FFH สำหรับคำสั่งการรับแต่ละครั้ง ดังนั้น 65 กิโลไบท์ของบัฟเฟอร์สามารถรับได้ปราศจากความผิดพลาดการไหลล้นบัฟเฟอร์

RIC HEX CODE	RECEIVER INTERRUPT RESULT CODE	Rx STATUS AFTER INT
X0	A1 MATCH OR GENERAL RECEIVE	ACTIVE
X1	A2 MATCH	ACTIVE
03	CRC ERROR	ACTIVE
04	ABORT DETECTED	ACTIVE
05	IDLE DETECTED	DISABLED
06	EOP DETECTED	DISABLED
07	FRAME <32 BITS	ACTIVE
08	DMA OVERRUN	DISABLED
09	MEMORY BUFFER OVERFLOW	DISABLED
0A	CARRIER DETECT FAILURE	DISABLED
0B	RECEIVER INTERRUPT OVERRUN	DISABLED
X	PARTIAL BYTE RECEIVED	
E	ALL 8 BITS OF LAST BYTE	
0	ID	
8	D1 - D0	
4	D2 - D0	
C	D3 - D0	
2	D4 - D0	
A	D5 - D0	
6	D6 - D0	

รูปที่ 4.9 รหัสผลลัพธ์อินเตอร์รัพต์ภาครับ

4.16 <u>คำสั่ง Transmit</u>

ภาคส่งของ 8273 ได้รับการสนับสนุนโดยคำสั่ง Transmit 3 คำสั่ง และ

คำสั่ง Abort ที่สอดคล้อง 3 คำสั่ง ดังนี้คือ

4.18 <u>คำสั่ง Transmit Frame</u>

คำสั่ง Transmit Frame ใช้สำหรับส่งเฟรม 1 เฟรมโดยตรง เมื่อเลือกโหมดบัฟเฟอร์ต้องการ 4 พารามิเตอร์และ 2 พารามิเตอร์เมื่อไม่ใช่โหมดบัฟเฟอร์ ในแต่ละกรณี 2 พารามิเตอร์แรกคือ ไบท์นัยสำคัญต่ำสุดและสูงสุดของความยาวเฟรมที่ต้องการ (L0 , L1) ในโหมดบัฟเฟอร์ L0 และ L1 เท่ากับความยาวไบท์ของฟิลด์ข้อมูลที่ต้องการในขณะที่โหมดนอนบัฟเฟอร์ L0 และ L1 เท่ากับความยาวไบท์ของฟิลด์ข้อมูลบวกกับอีก 2 ไบท์ (L0 และ L1 กำหนดจำนวนของการถ่ายโอนข้อมูลที่ดำเนินการ) ในโหมดบัฟเฟอร์ นั้น ฟิลด์แอดเดรสและควบคุมได้จัดให้กับภาคส่งในฐานะพารามิเตอร์ที่ 3 และ 4 ตาม ลำดับ ส่วนโหมดนอนบัฟเฟอร์ ฟิลด์แอดเดรสและควบคุมจะต้องผ่านในฐานะเป็น 2 ไบท์ ๆ แรกของการถ่ายโอนข้อมูลเมื่อปล่อยคำสั่ง Transmit Frame 8273 จะสร้าง RTS (Request - to - send) แอคตีฟ และในขณะเดียวกันมันก็จะคอยจนกว่า CTS (Clear - to - send) กลายเป็นแอคตีฟก่อนที่จะเริ่มต้นเฟรม ถ้าบิท Preframe Sync ในรีจีสเตอร์โหมดทำงานเป็น 1 ภาคส่งจะนำหน้าแฟล็กเปิดด้วย 2 ตัวอักษร (16 transitions) ถ้าบิท Flag Stream ในรีจีสเตอร์โหมดทำงานเป็น 1 เฟรม (รวมด้วย Preframe Sync ถ้าเลือก) จะเริ่ม บนบริเวณเขตแดนแฟล็ก อีกนัยหนึ่งเฟรมจะเริ่มบริเวณเขตแดนตัวอักษร ที่ท้ายของเฟรมภาคส่งจะอินเตอร์รัพต์ซีพียู (ผลลัพธ์อินเตอร์รัพต์จะอธิบายสั้นๆ)และกลับสู่สถานะ Idle หรือ Flag Stream ซึ่งขึ้นอยู่กับบิท Flag Stream ของรีจีสเตอร์โหมดทำงาน ถ้า RTS แอคตีฟก่อนหน้าคำสั่ง Transmit 8273 จะไม่ทำการเปลี่ยนแปลงมัน ถ้ามันไม่อยู่ในสถานะ แอคตีฟ 8273 จะจัดการมันภายในเวลา 1 ตัวอักษร

4.18 <u>คำสั่ง Loop Transmit</u>

คำสั่ง Loop transmit มีลักษณะคล้ายกับคำสั่ง Transmit Frame (การกำหนดพารามิเตอร์ทำนองเดียวกัน) แต่ในเมื่อมันจัดการกับรูปลักษณ์ลูป โหมดดีเลย์ 1 บิท จะต้องได้รับการเลือกถ้าภาคส่งไม่อยู่ในโหมด Flag Stream เมื่อคำสั่งนี้ถูก

ปล่อยภาคส่งจะต้องคอยจนกระทั้งอักษร EOP ที่มันได้รับถูกเปลี่ยนเป็น Flag (กระทำโดยอัตโนมัติ) ก่อนที่จะทำการส่งถ้าภาคส่งอยู่ในโหมด Flag Stream (ในระหว่างการส่งเส้นทางดีเลย์ 1 บิท จะถูกลอยตัว) ตามที่ผลลัพธ์ของเฟรมที่ได้รับแบบเจาะจงในระหว่างคำสั่ง Selective Loop Receive และการส่งผ่านจะเริ่มต้นที่บริเวณเขตแดนแฟล็กตัวต่อไปสำหรับโหมดบัฟเฟอร์ หรือบริเวณเขตแดนแฟล็กที่ 3 สำหรับโหมดนอนบัฟเฟอร์ความไม่ลงรอยกันเช่นนี้ จะต้องยินยอมให้มีเวลาทำการถ่ายโอนข้อมูลให้เต็มบัฟเฟอร์ภายในภาคส่งเพียงพอ เกิด ที่ท้ายลูปของการส่งโหมดดีเลย์ 1 บิทจะกลับมาเริ่มต้นใหม่และโหมด Flag Stream เป็น 0 รายละเอียดการทำงานของลูปจะครอบคลุมในตอนต่อไป

4.19 <u>คำสั่ง Transmit Transparent</u>

คำสั่ง Transmit Transparent จะทำให้ 8273 นำไปสู่การส่งแถวข้อมูลเป็นบล็อกๆ ข้อมูลเหล่านี้จะไม่มีโพรโทคอล SDLC ดังนั้น ไม่มีการแทรกบิทศูนย์,แฟล็ก หรือ FCS ดังนั้น มันเป็นไปได้ที่จะสร้างและส่งข่าวสารใน Bi - Sync สำหรับการสวิทช์โปรเซสเซอร์ในส่วนหน้าหรือสร้างและส่งข่าวสาร SDLC โดยไม่ต้องตรวจสอบความถูกต้อง FCS เพื่อจุดประสงค์วินิจฉัย ต้องการเฉพาะพารามิเตอร์ L0 และ L1 เท่านั้นในเมื่อไม่มีฟิลด์ต่าง ๆ ในโหมดนี้ (8273 จะไม่มีคำสั่ง Receive Transparent สนับสนุน)

4.20 <u>คำสั่ง Abort</u>

แต่ละคำสั่ง Transmit ตามที่ได้กล่าวข้างบนนี้จะมีคำสั่ง Abort ร่วมด้วย คำสั่ง Abort Frame Transmit จะเป็นสาเหตุให้ภาคส่งทำการส่ง 11111111 (ไม่มีบิท 0 แทรก) ทันทีและในขณะนั้นจะกลับคืนสู่ Idle หรือ Flag Stream ซึ่งขึ้นอยู่กับ Flag Stream ในรีจีสเตอร์โหมดทำงาน (ตามที่ อักษร Abort 11111111 นี้สามารถเข้ากันได้กับ SDLC หรือ HDLC)

สำหรับ Loop Transmit คำสั่ง Abort Loop Transmit จะเป็นสาเหตุให้ภาคส่งทำการส่งแฟล็ก 1 แฟล็กและในเวลานั้นมันจะกลับสู่สถานะดีเลย์ 1 บิท

โพรโทคอลของลูปจะขึ้นอยู่กับความผิดพลาด FCS ในการตรวจจับ Abort ของเฟรม

คำสั่ง Abort Transmit Transparent เป็นสาเหตุโดยตรงที่ทำให้ภาคส่งกลับสู่สภาพ Idle หรือ Flag ตามฟังก์ชั่นของโหมด Flag Stream ที่กำหนด คำสั่ง Abort ไม่ต้องการพารามิเตอร์ อย่างไรก็ตามมันยังคงกำเนิดสัญญาณอินเตอร์รัพต์และผ่านผลลัพธ์กลับคืนเมื่อเสร็จสิ้น สรุปคำสั่ง Transmit ได้แสดงในรูปที่ 4.10 รูปที่ 4.11 ได้แสดงรหัสผลลัพธ์อินเตอร์รัพต์ของการส่งต่าง ๆ ดังเช่นในการทำงานของภาครับ ภาคส่งจะกำเนิดอินเตอร์รัพต์พื้นฐาน ไม่ว่าจะเป็นการสิ้นสุดการทำงานเนื่องจากประสพความสำเร็จ หรือเงื่อนไขความผิดพลาดในการนำไปสู่ช่วงผลลัพธ์

COMMAND	HEX CODE	PARAMETERS	RESULTS TX I/R
TRANSMIT FRAME	C8	L0 , L1 , A , C	TIC
ABORT	CC	NONE	TIC
LOOP TRANSMIT	CA	L0 , L1 , A , C	TIC
ABORT	CE	NONE	TIC
TRANSMIT TRANSPARENT	C0	L0 , L1	TIC
ABORT	CD	NONE	TIC

* A และ C ถูกส่งผ่านในฐานะเป็น PARAMETER เฉพาะใน BUFFER MODE เท่านั้น

รูปที่ 4.10 สรุปคำสั่งของภาคส่ง

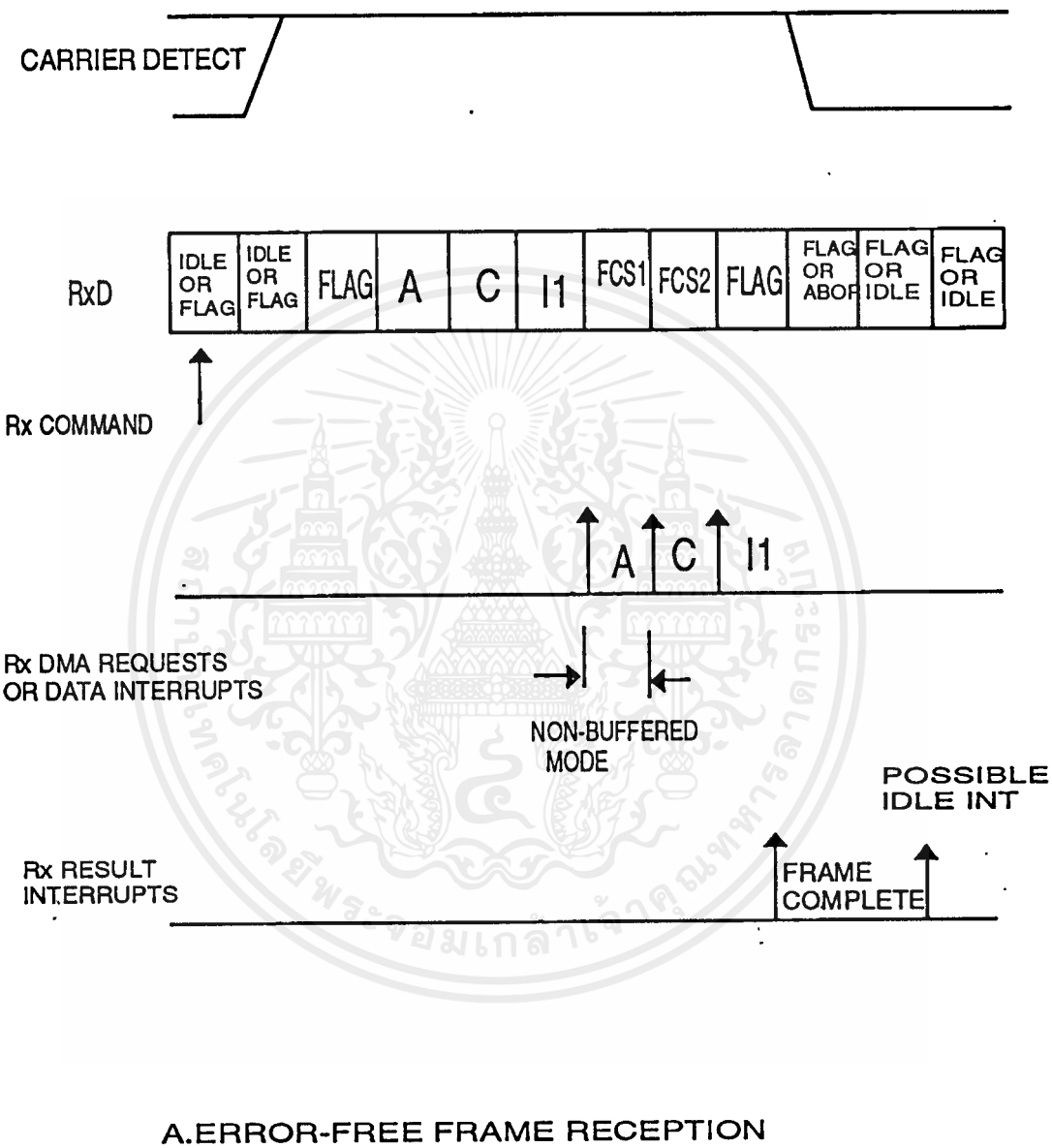
TIC HEX CODE	TRANSMITTER INTERRUPT RESULT CODE	TX STATUS AFTER INT
0C	EARLY Tx INTERRUPT	ACTIVE
0D	FRAME Tx COMPLETE	IDLE OR FLAGS
0E	DMA UNDERRUN	ABORT
0F	CLEAR TO SEND ERROR	ABORT
10	ABORT COMPLETE	IDLE OR FLAGS

รูปที่ 4.11 รหัสผลลัพธ์ TRANSMITTER INTERRUPT

4.21 <u>คำสั่ง Reset</u>

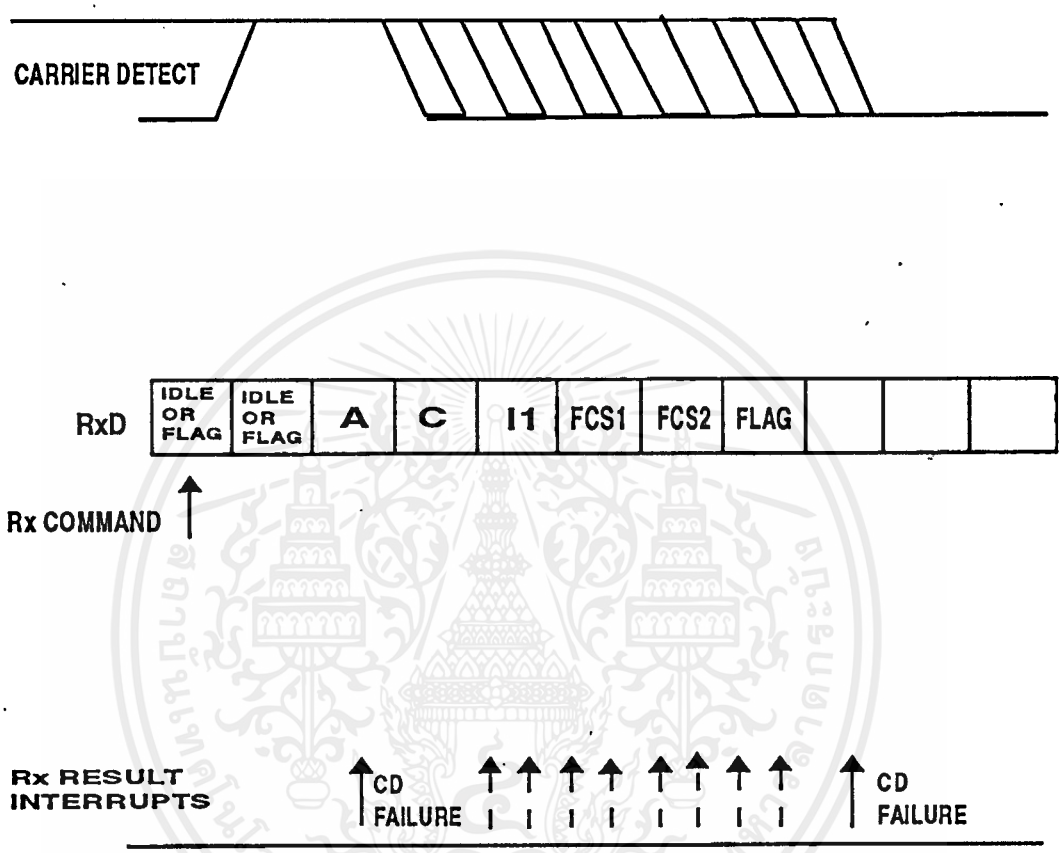
คำสั่ง Reset จัดให้รีเซ็ททางด้านซอฟต์แวร์สำหรับ 8273 ซึ่งเป็นกรณีพิเศษและนำมาใช้เป็นคำสั่งเชื่อมโยงในสภาวะปกติ รีจีสเตอร์ Test Mode จัดไว้ให้ในความสะดวกในการรีเซ็ทการรีเซ็ท 8273 นั้น โดยการส่ง 01H แล้วตามด้วย 00H เข้าไปในรีจีสเตอร์ Test Mode การเขียน 01H โดยการตาม 00H นั้นเป็นความต้องการที่จะเรียนแบบการรีเซ็ททางฮาร์ดแวร์ ในเมื่อ 8273 ต้องการเวลาในกรรมวิธีรีเซ็ทภายใน ดังนั้นสัญญาณนาฬิกาอย่างน้อย 10 ไซเกิลของ CLK ที่จะต้องเกิดในระหว่างที่ทำการเขียน 01H และ 00H การกระทำเช่นนี้จะถือเป็นแบบอย่างเดียว กันกับการรีเซ็ททางฮาร์ดแวร์ กล่าวคือ

1. เอาท์พุทที่ควบคุมโมเด็มจะถูกกระตุ้นให้เป็นสถานะสูง (inactive)
2. รีจีสเตอร์ Status จะถูกเคลีย
3. คำสั่งใดๆ ในระหว่างดำเนินการจะยุติ
4. 8273 จะเข้าสู่สถานะ idle จนกระทั้ง คำสั่งต่อไปจะถูกปล่อย



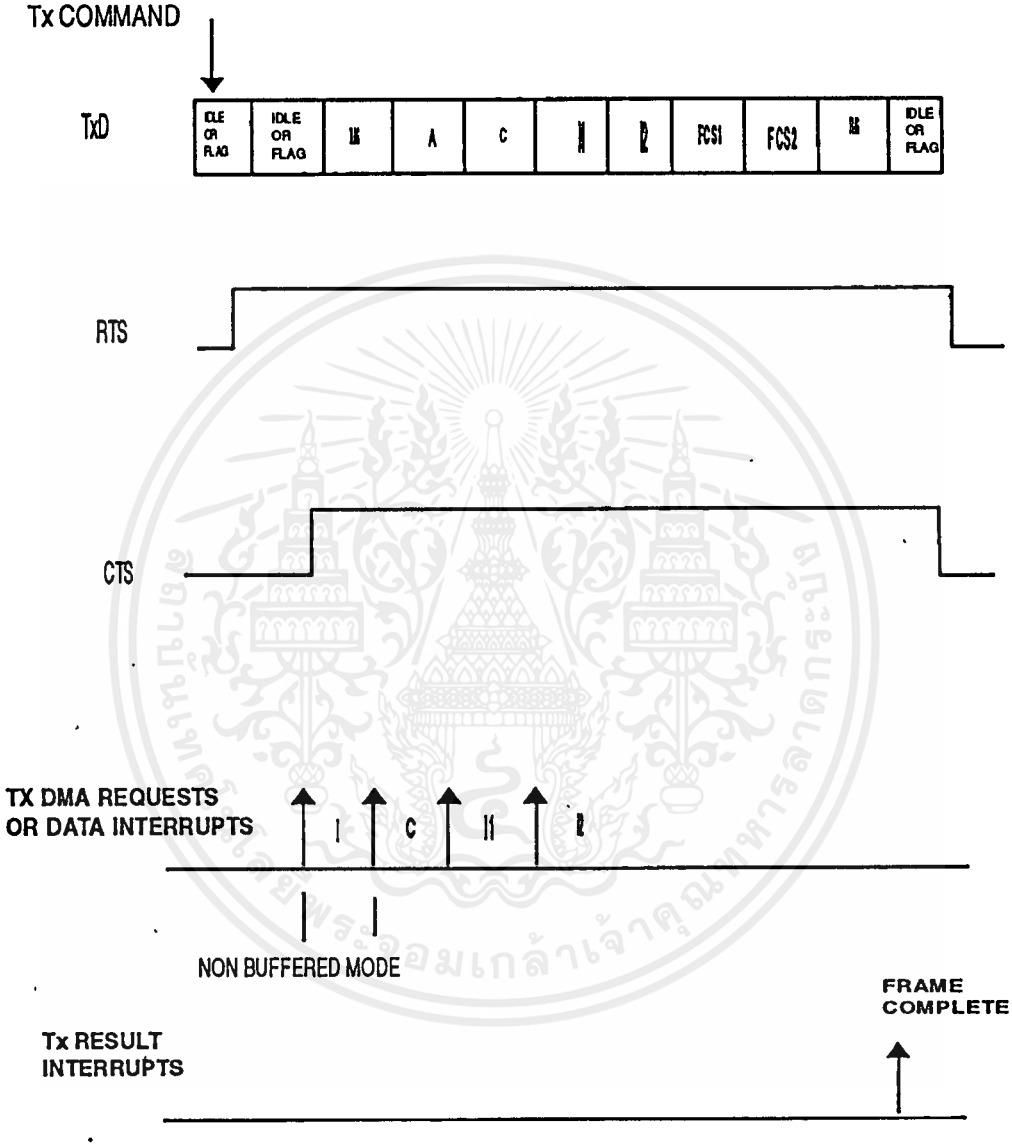
A.ERROR-FREE FRAME RECEPTION

Figure 3.12 Sample Receiver Timing Diagram



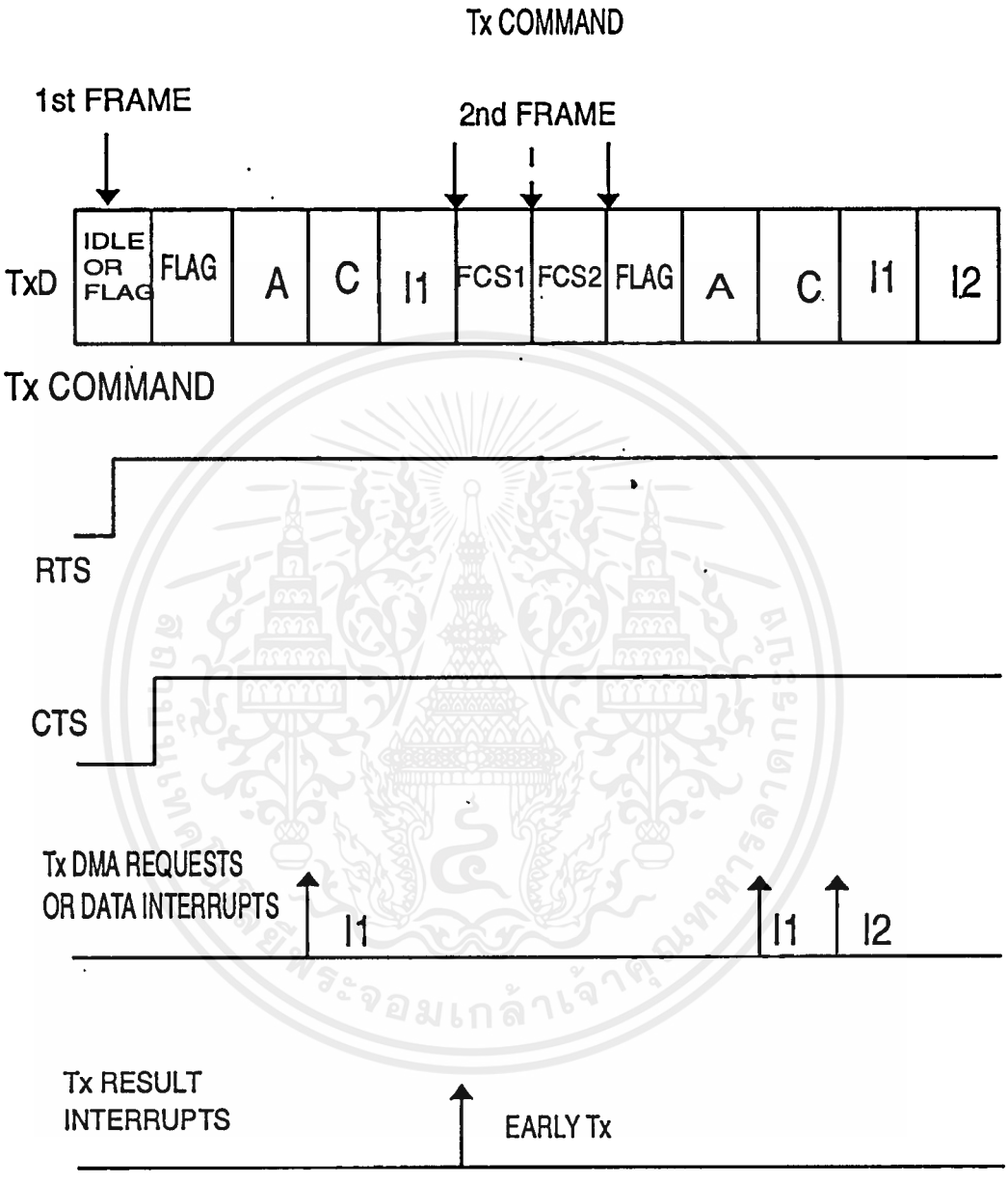
B.CARRIER DETECT FAILURE DURING FRAME RECEPTION

FIGURE 4.12 Sample Receiver Timing Diagram.



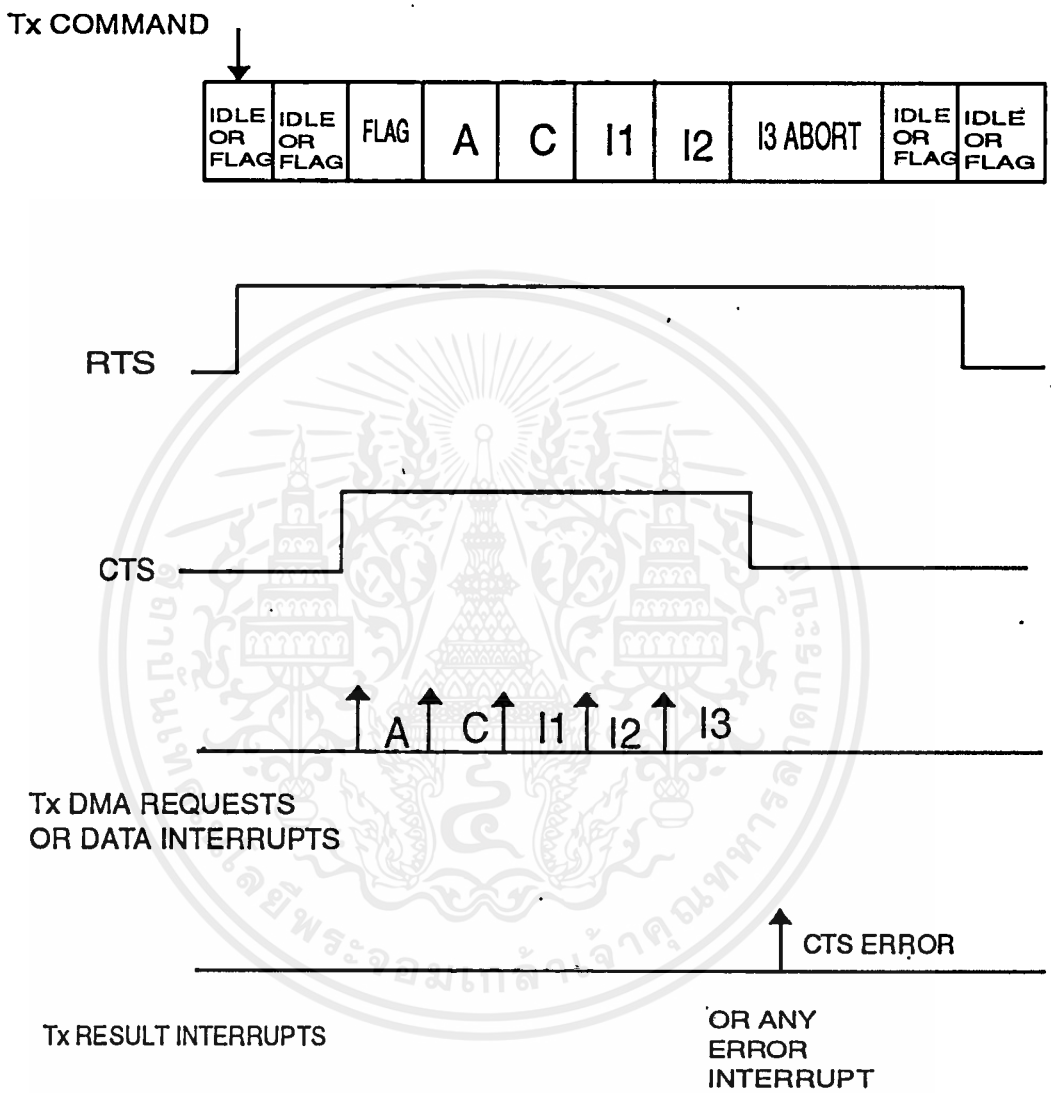
A. ERROR-FREE FRAME TRANSMISSION

รูปที่ 4.13 ตัวอย่าง Transmitter Timing Diagrams



B.DIAGRAM SHOWING Tx COMMAND QOEING AND EARLY Tx INTERRUPT (SINGLE FLAG BETWEEN FRAMES) BUFFERED MODE IS ASSUMED

รูปที่ 4.13 ตัวอย่าง Transmitter Timing Diagrams



C.CTS FAILURE (OR OTHER ERROR) DURING TRANSMISSION

รูปที่ 4.13 ตัวอย่าง Transmitter Timing Diagrams

รูปที่ 4.14 a จนถึงรูปที่ 4.14 i ได้แสดงผังงานที่นำมาใช้สำหรับการพัฒนาโปรแกรม 8273 ส่วนโปรแกรมจริงนั้น ได้รวบรวมไว้ในภาคผนวก รูปที่ 4.14 เป็นลูปโพลสถานะหลัก ภายหลังจากที่อุปกรณ์ทั้งหมดได้มีการเซ็ทค่าเริ่มต้นและได้แสดงผลอักขระ Prompt แล้ว ลูปมีทางเข้าที่ LOOPIT ลูปนี้จะคอยตรวจสอบการเปลี่ยนแปลงสถานะภายในบัฟเฟอร์ RESULT หรืออักขระที่รับจากคีย์บอร์ด หรือถ้ามีการรับเฟรมโพลเข้ามา ถ้าเงื่อนไขใด ๆ เหล่านี้ถูกตรวจพบโปรแกรมก็จะถูกแยกไปยังรูทีนที่ได้จัดไว้ให้ หรือกล่าวได้ว่าลูปจะถูกขวางอีกครั้ง

บัฟเฟอร์ RESULT เป็น CIRCULAR BUFFER ขนาด 255 ไบท์ซึ่งสนับสนุนโดยสองพอยน์เตอร์คือ CNADR และ LDADR CNADR เป็นพอยน์เตอร์คอนโซลที่จะชี้ไปยังตำแหน่งที่ว่างในบัฟเฟอร์ซึ่งอินเตอร์รัพต์แฮนเดิลจะใช้เป็นที่บรรจุผลลัพธ์ต่อไป ผลลัพธ์ของภาคส่งและภาครับจะใช้บัฟเฟอร์เดียวกัน LOOPIT จะพิจารณาพอยน์เตอร์เหล่านี้ โดยการตรวจจับ เมื่อ CNADR ไม่เท่ากับ LDADR ซึ่งจะหมายถึงบัฟเฟอร์ได้บรรจุผลลัพธ์ไว้ ซึ่งยังไม่ได้แสดงผลเมื่อเกิดกรณีเช่นนี้ ขึ้น โปรแกรมจะถูกแยกไปยังรูทีน DISPLY

DISPLY จะกำหนดแหล่งที่มาของผลลัพธ์ที่ยังไม่ได้แสดงผล โดยการตรวจสอบผลลัพธ์แรก ผลลัพธ์แรกมีความสำคัญมาก เพราะเป็นรหัสผลลัพธ์อินเตอร์รัพต์ถ้าผลลัพธ์นี้มีค่าเท่ากับหรือมากกว่า 0CH แสดงผลลัพธ์ มาจากอินเตอร์รัพต์ภาคส่ง ถ้าตรงข้ามจะมาจากแหล่งภาครับดังนั้นรหัสผลลัพธ์ของ แหล่งจ่ายจะถูกนำมาแสดงผลบนคอนโซล และตามด้วยผลลัพธ์ตัวต่อไป อีก 4 ผลลัพธ์จากบัฟเฟอร์ถ้าแหล่งจ่ายมาจากอินเตอร์รัพต์ภาคส่ง รูทีนเพียงแต่นำ CNADR ขึ้นมาชี้ใหม่เท่านั้น สำหรับแหล่งจ่ายภาครับข้อมูลในบัฟเฟอร์ข้อมูลภาครับ จะถูกนำมาแสดงผลในส่วนที่เพิ่มกับรหัสอินเตอร์รัพต์ภาครับก่อนที่จะกลับเข้าสู่ LOOPIT

ถ้าพอยน์เตอร์บัฟเฟอร์ RESULT บอกถึงบัฟเฟอร์ว่างเปล่าอักขระจากคีย์บอร์ดจะถูก สอบถามต่อไป ถ้ามีการกดอักขระเข้ามารูทีน GETCMD จะถูกเรียกอักขระจะถูกนำเข้ามาตรวจสอบตามกฎเกณฑ์ ถ้าอักขระผิดกฎเกณฑ์จะเป็นสาเหตุให้กลับมา Prompt อีกครั้ง อักขระที่ถูกต้องจะบอกถึงการ เริ่มของคำสั่งอินพุท ส่วนมากคำสั่งที่รู้จักจะประกอบด้วย 2 อักขระซึ่งจะบอกถึงการกระทำ อย่างเช่น GR หมายถึง

General Receiver ซอฟต์แวร์จะรู้จักรหัสคำสั่ง 2 อักขระนี้และถือปฏิบัติตามที่ได้จัดไว้ สำหรับคำสั่งประเภทที่ไม่ใช่ Transmit เลขฐานสิบหกเสมือนของคำสั่งจะถูกวางไว้ในรีจีสเตอร์ CL และจำนวนของพารามิเตอร์ที่เกี่ยวเนื่องกับคำสั่งนั้นจะถูกวางเข้าไปในรีจีสเตอร์ CH และโปรแกรมจะถูกแยกไปยังรูทีน COMM

รูทีน COMM จะสร้างบัฟเฟอร์คำสั่งโดยการอ่านจำนวนของ พารามิเตอร์ที่ต้องการจากคีย์บอร์ด และทำการวางเข้าไปในบัฟเฟอร์ซึ่งถูกชี้ โดย CMDBUF ในเวลานั้นรูทีน COMM2 จะเป็นตัวปล่อยออกจากบัฟเฟอร์คำสั่งไปยัง 8273

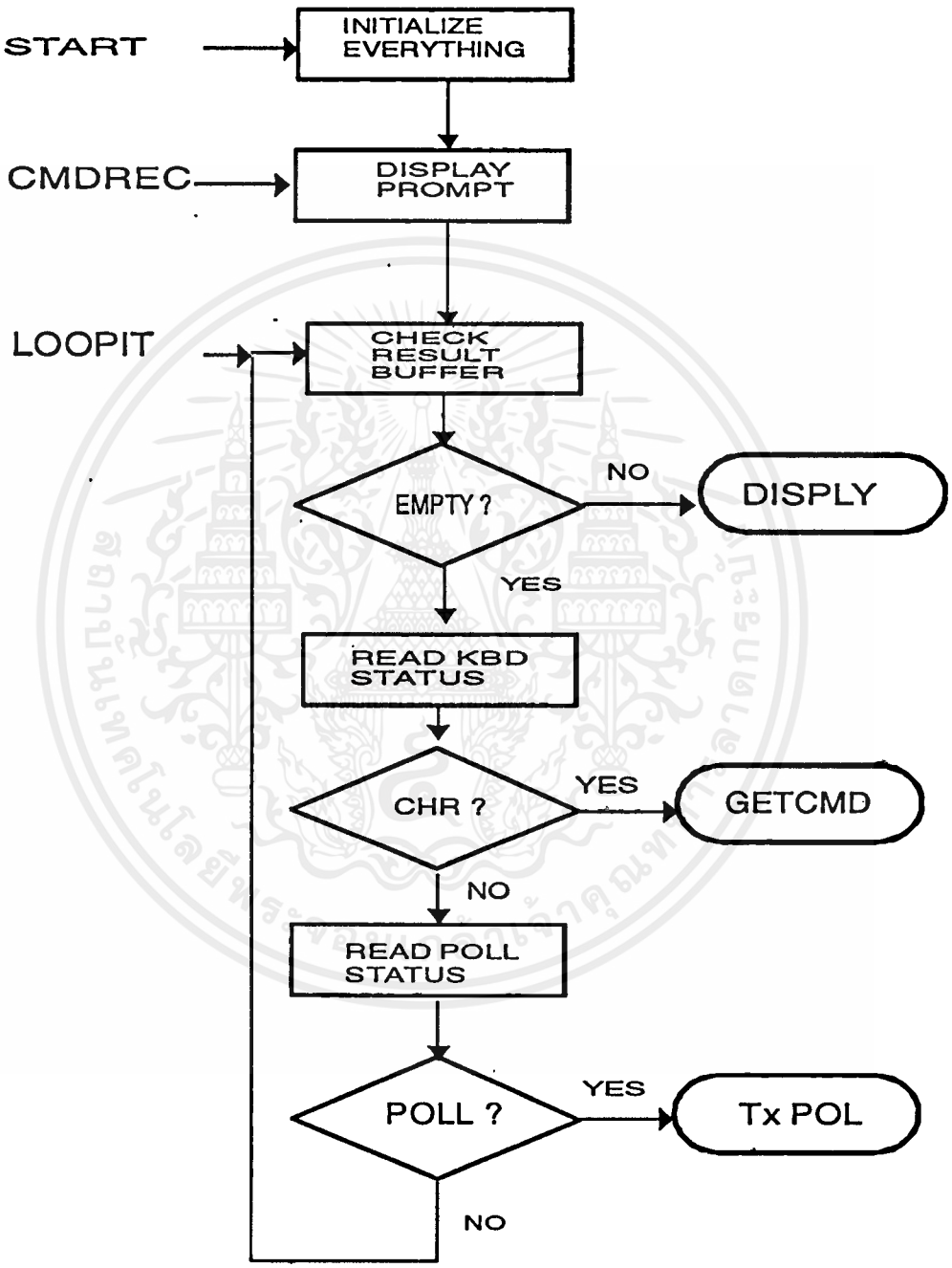
ถ้าคำสั่งระบุประเภท Transmit บัฟเฟอร์คำสั่งจะถูกสร้างคล้าย ๆ กับรูทีน COMM อย่างไรก็ตาม ในเมื่อข้อมูลของฟิลด์ข่าวสารจะถูกป้อนมาจากคีย์บอร์ดรูทีนชั่วคราว TF จะถูกเรียก TF จะโหลดข้อมูลที่จะใช้ส่งเข้าไปในบัฟเฟอร์ที่ถูกชี้โดย TXBUF มันจะนับจำนวนไบท์ข้อมูลที่ป้อนเข้าไปใน บัฟเฟอร์ และ จะทำการบรรจุจำนวนที่นับได้นี้เข้าไปในบัฟเฟอร์คำสั่งในฐานะเป็น L0 , L1 คำสั่งจะถูกปล่อยไปยัง 8273 โดย COMM2

มีอีกคำสั่งหนึ่ง ที่ไม่มีผลลัพธ์ในการกระทำโดยตรง ในขณะที่ปล่อยคำสั่งไปยัง 8273 คือ คำสั่ง Z ทำงานเป็นซอฟต์แวร์ฟลิบฟลอบ ซึ่งจะถูกเลือก เมื่อซอฟต์แวร์จะตอบสนองแบบ อัตโนมัติ กับการได้รับโพลลิ่งเฟรมถ้าโหมด Poll - Response ถูกเลือกอักขระ Prompt จะถูกเปลี่ยนเป็น '+' ถ้าเฟรมที่ได้รับบรรจุโพล ที่ได้จัดไว้ล่วงหน้าในฟิลด์ควบคุม หน่วยความจำตำแหน่ง POLIN จะถูกทำไม่ให้เป็น ศูนย์ โดยตัวควบคุมอินเตอร์รัพต์ ภาครับ LOOPIT จะทำการตรวจสอบ ณ ตำแหน่งนี้ และถ้าไม่เป็นศูนย์จะเป็นสาเหตุให้แยกออกไปยัง รูทีน TxPOL รูทีน TxPOL จะเคลีย POLIN แล้ว เซ็ทพอยน์เตอร์ไปยังบัฟเฟอร์คำสั่งพิเศษที่ CMDBUF1 แล้วปล่อยคำสั่งโดยทาง COMM2 บัฟเฟอร์คำสั่งพิเศษจะบรรจุเฟรม Response สำหรับการรับโพลเฟรม การกระทำเช่นนี้จะเกิดขึ้นเฉพาะเมื่อ คำสั่ง Z มีการเปลี่ยน Pormpt ไปเป็น '+' ถ้า Prompt ปกติ '-' เฟรมโพลลิ่งจะถูกแสดงผลในฐานะปกติและไม่มี Response ถูกส่งโหมด Poll - response ถูกนำมาใช้ในระหว่างทดสอบ IBM

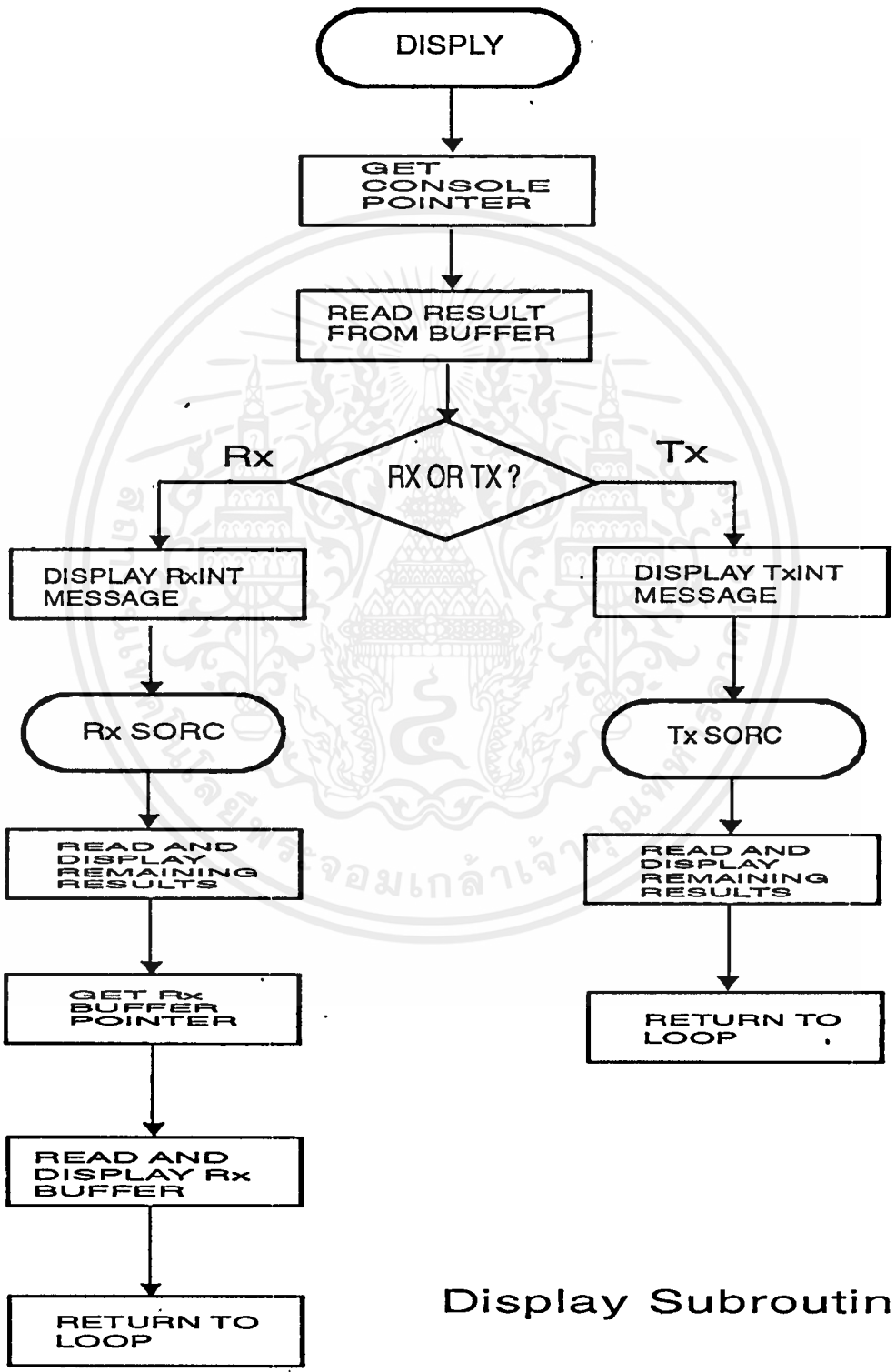
สองรูทีนซอฟต์แวร์สุดท้าย คือ อินเตอร์แฮนเดิลภาคส่ง และ ภาครับ เมื่อเกิดอินเตอร์รัพต์ โปรแกรมจะต้องมีการตรวจจับแหล่งที่มา ของอินเตอร์รัพต์ นั่นคือมา

จากภาคส่งหรือภาครับ จากรีจีสเตอร์สถานะ ซึ่งจะสามารถทำการแฮนเดิลได้ถูกต้อง TXI เป็นอินเตอร์รัพต์ภาคส่งเนื่องจากอินเตอร์รัพต์มีทั้งอินเตอร์รัพต์ผลลัพธ์และข้อมูลรีจีสเตอร์ STATUS จะถูกนำมาตรวจสอบอีกครั้งในบิท IRA ถ้า IRA = 0 จะหมายถึงเป็นอินเตอร์รัพต์ถ่ายโอนข้อมูลและถ้า IRA = 1 จะหมายถึง อินเตอร์รัพต์ผลลัพธ์ ถ้าเป็นการถ่ายโอนข้อมูล ข้อมูลจะถูกเขียนจาก TXBUF ซึ่งชี้โดยพอยน์เตอร์ XPNT ไปยัง 8273 โดยผ่านพอร์ท TXDACK ถ้า IRA = 1 จะมีการตรวจสอบบัฟเฟอร์ผลลัพธ์เต็ม ถ้าผลลัพธ์จะล้นบัฟเฟอร์โปรแกรมจะออกจากปฏิบัติการและควบคุมไปยัง Monitor ถ้าบัฟเฟอร์ผลลัพธ์ยังว่างพอ ผลลัพธ์จะถูกอ่านจากรีจีสเตอร์ TXI/R และจะถูกบรรจุเข้าไปในบัฟเฟอร์ผลลัพธ์ที่ชี้โดย LDADR และ โปรแกรม จะกลับไปสู่ตำแหน่งก่อนที่จะมีการอินเตอร์รัพต์เข้ามา

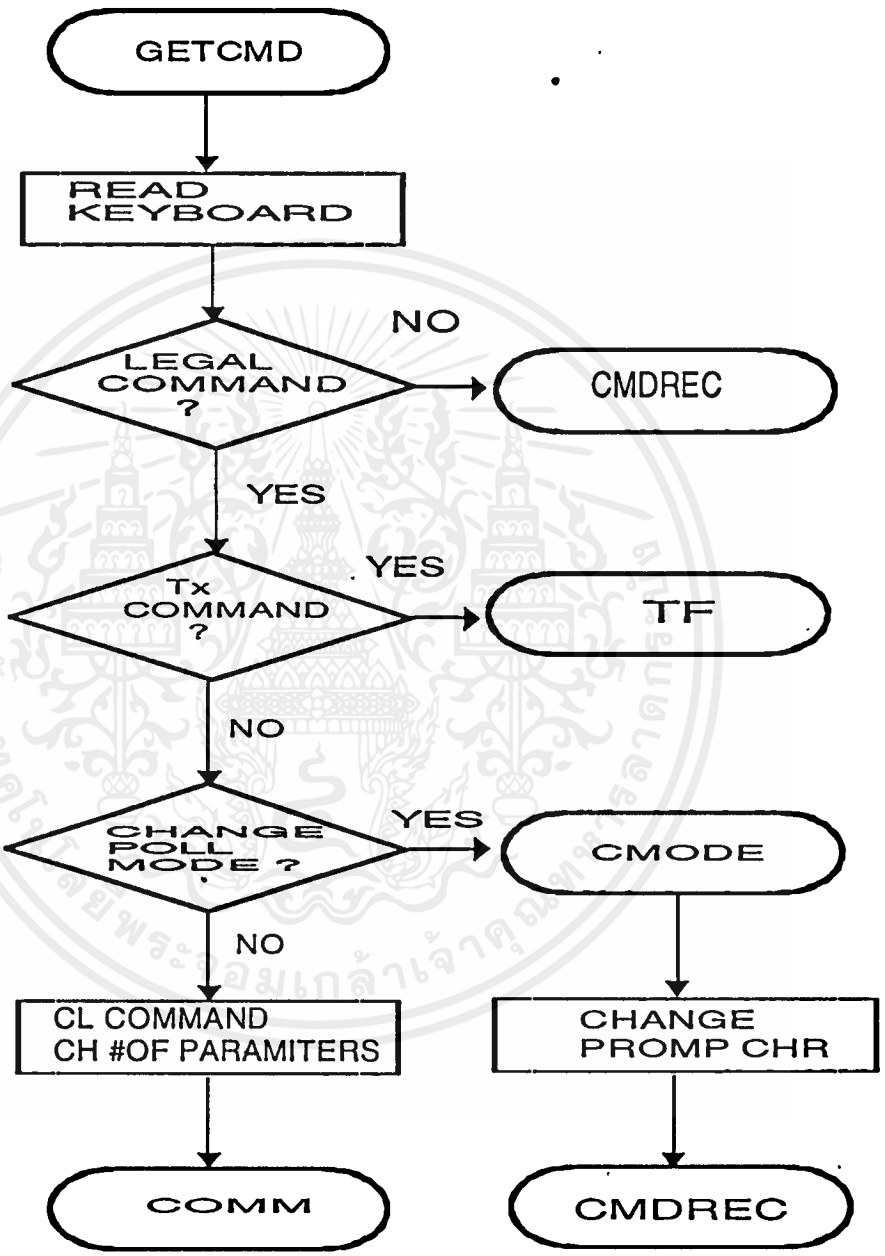
RXI เป็นแฮนเดิลภาครับ เช่นเดียวกันการตรวจสอบ IRA ในรีจีสเตอร์ STATUS ถ้า IRA = 0 ข้อมูลจะถูกอ่านจาก 8273 โดยผ่านพอร์ต RXDACK แล้วไปเก็บไว้ใน RXBUF ซึ่งใช้ RXPNT เป็นอินพุทพอยน์เตอร์ ถ้า IRA = 1 จะมีการตรวจสอบบัฟเฟอร์ผลลัพธ์เช่นเดียวกันกับ TXI ถ้าผลลัพธ์บัฟเฟอร์ยังไม่เต็ม ผลลัพธ์จะถูกอ่านจาก RXI/R และบรรจุเข้าไปในบัฟเฟอร์ผลลัพธ์ ณ จุดนี้อักขระ Prompt จะได้รับการพิจารณา ถ้าเป็นโหมด Poll - Response ถูกเลือกดังนั้นฟิลด์ควบคุมจะได้รับ การเปรียบเทียบในความเป็นไปได้ 2 กรณีของการโพลลิ่ง ฟิลด์ควบคุม ถ้าเกิดการตรงกันเกิดขึ้น บัฟเฟอร์คำสั่งพิเศษจะถูกโหลด และตัวชี้บอกโพล POLIN จะถูกทำไม่ให้เป็น ศูนย์ ถ้าไม่มีการตรงกันจะไม่มีการกระทำใดๆ และโปรแกรมกลับไปสู่ตำแหน่งที่มี การอินเตอร์รัพต์



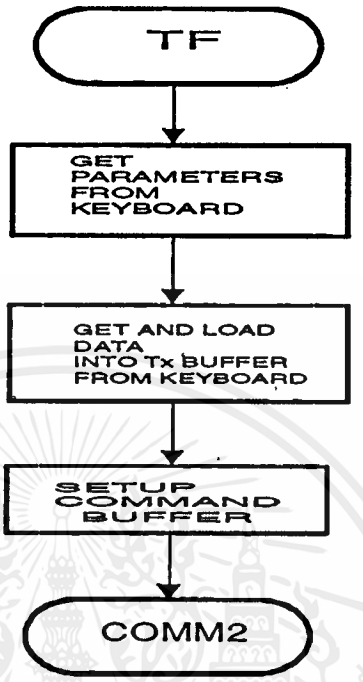
4.14 Main Status Poll Loop



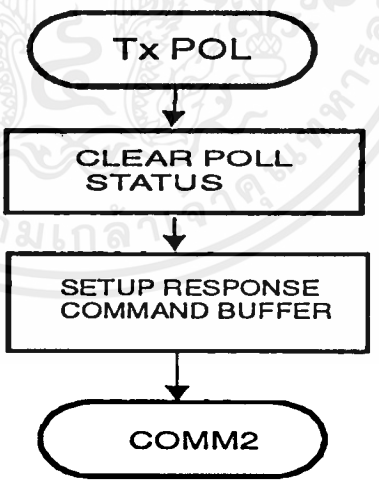
Display Subroutine



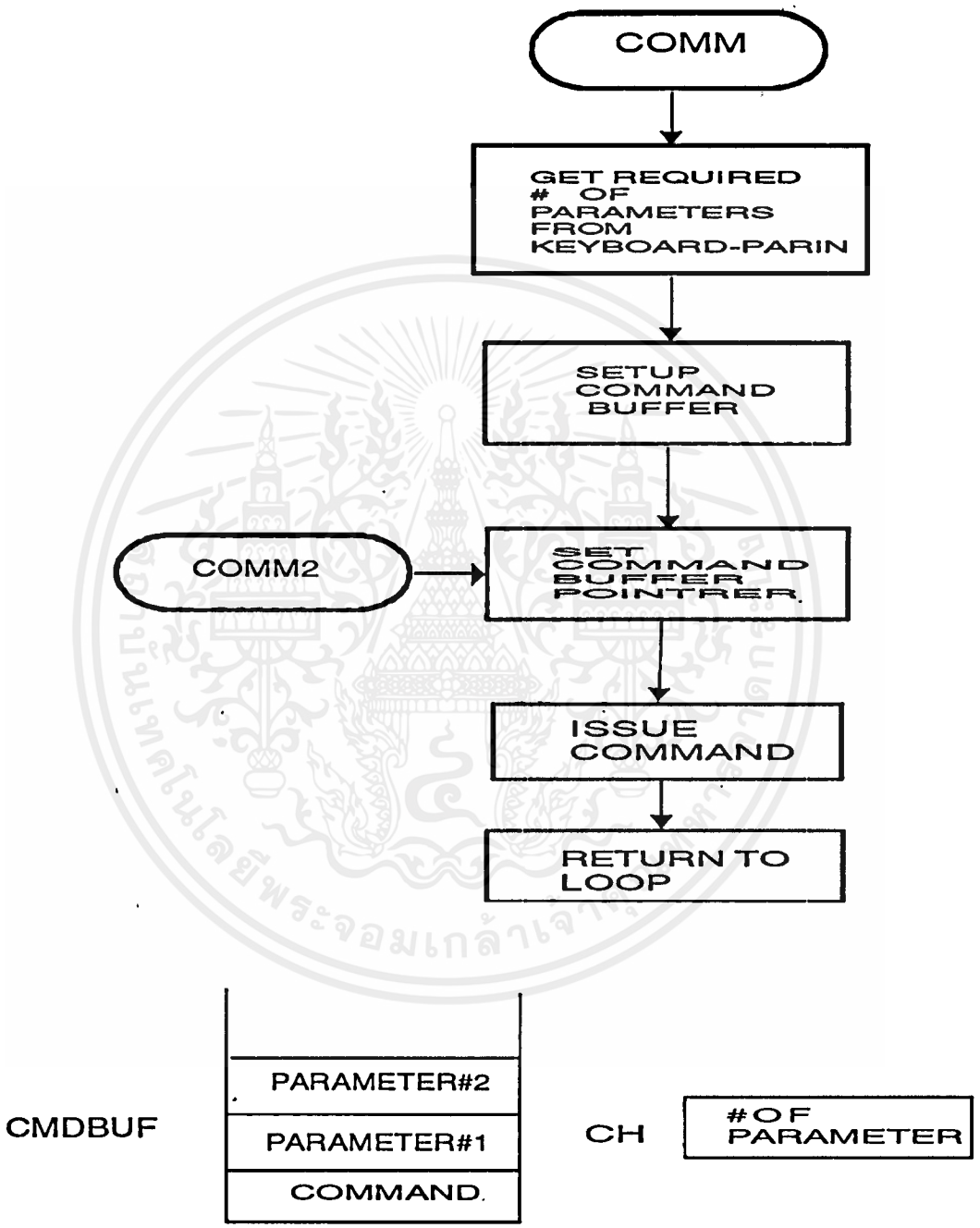
GETCMD Subroutine



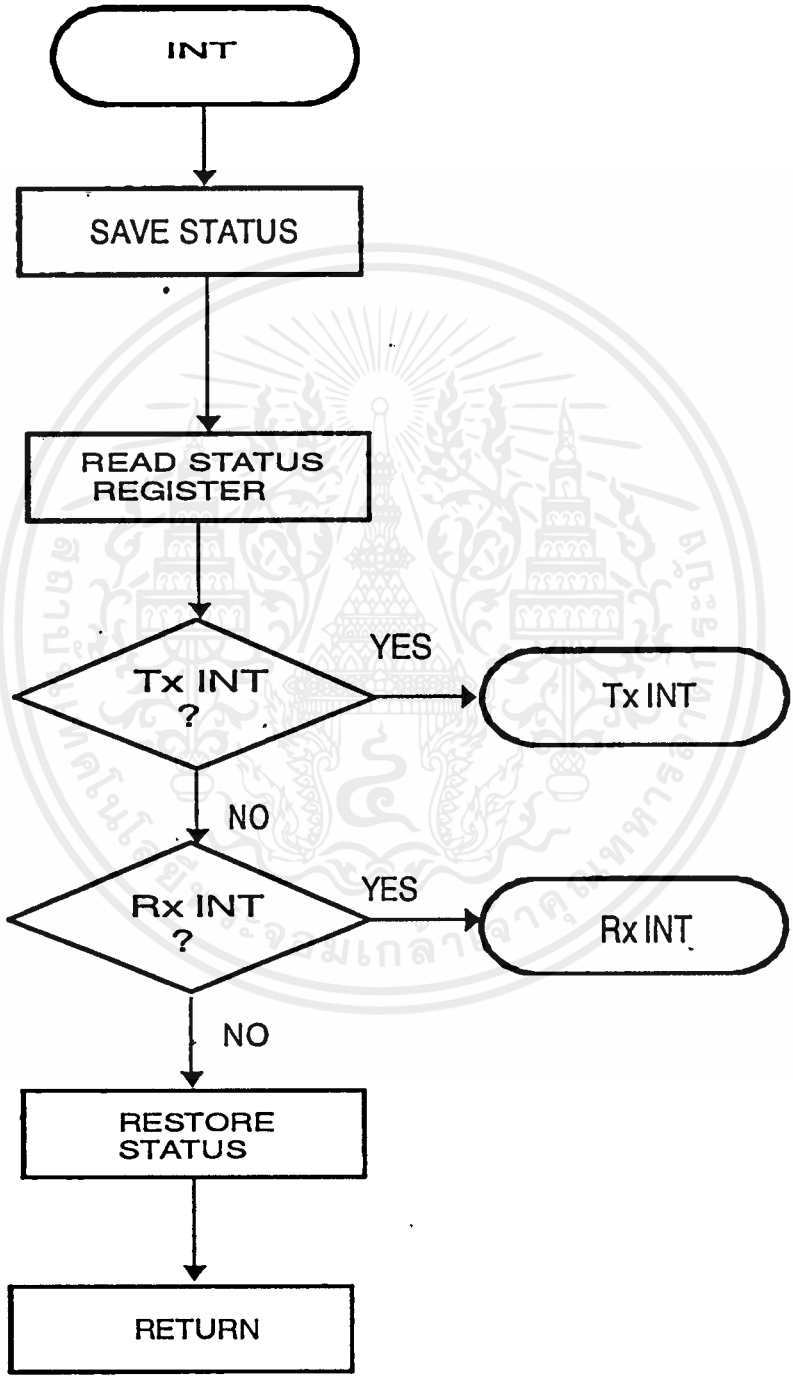
4.14d TF Subroutine



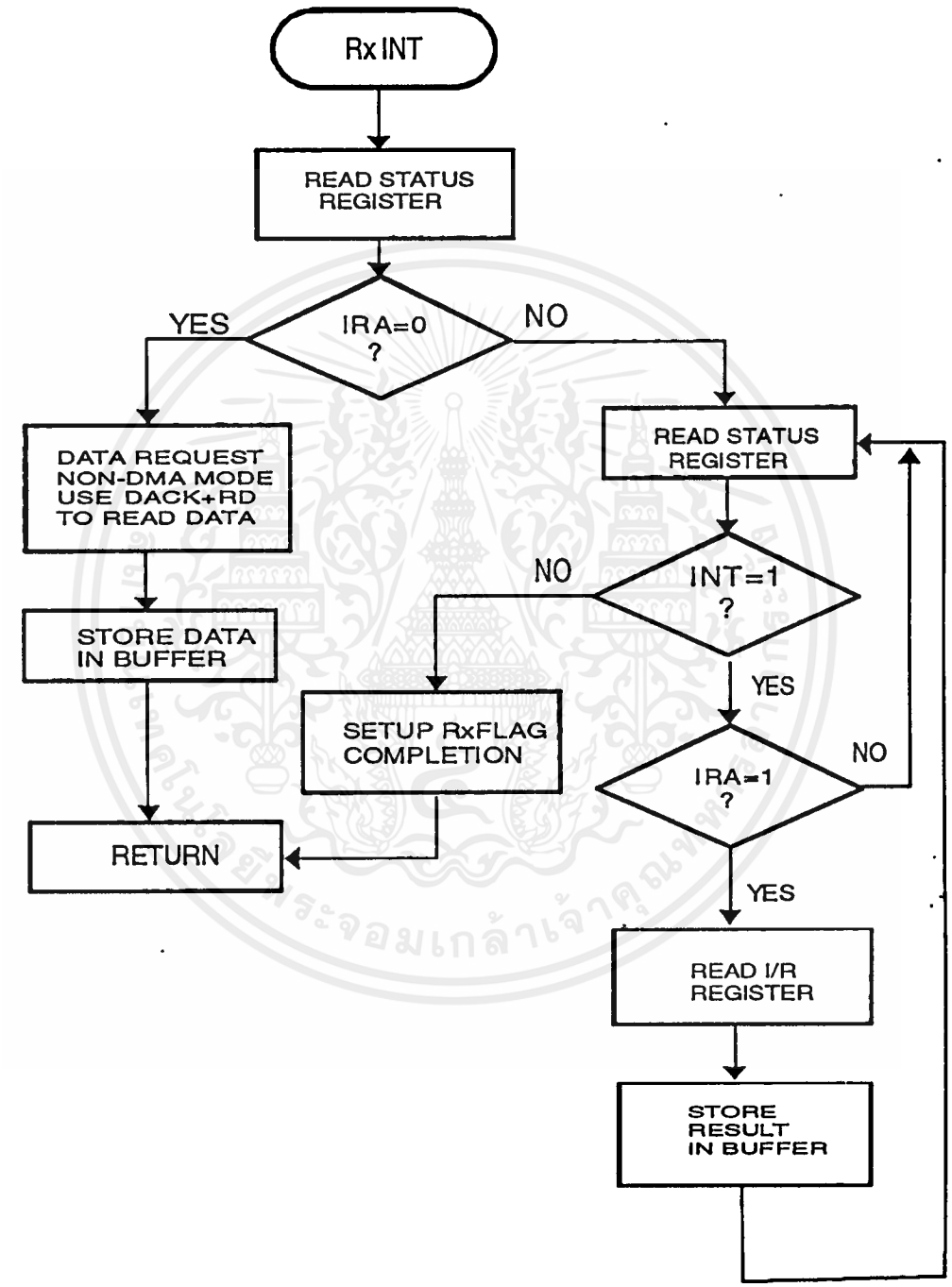
4.14e Tx POL Subroutine



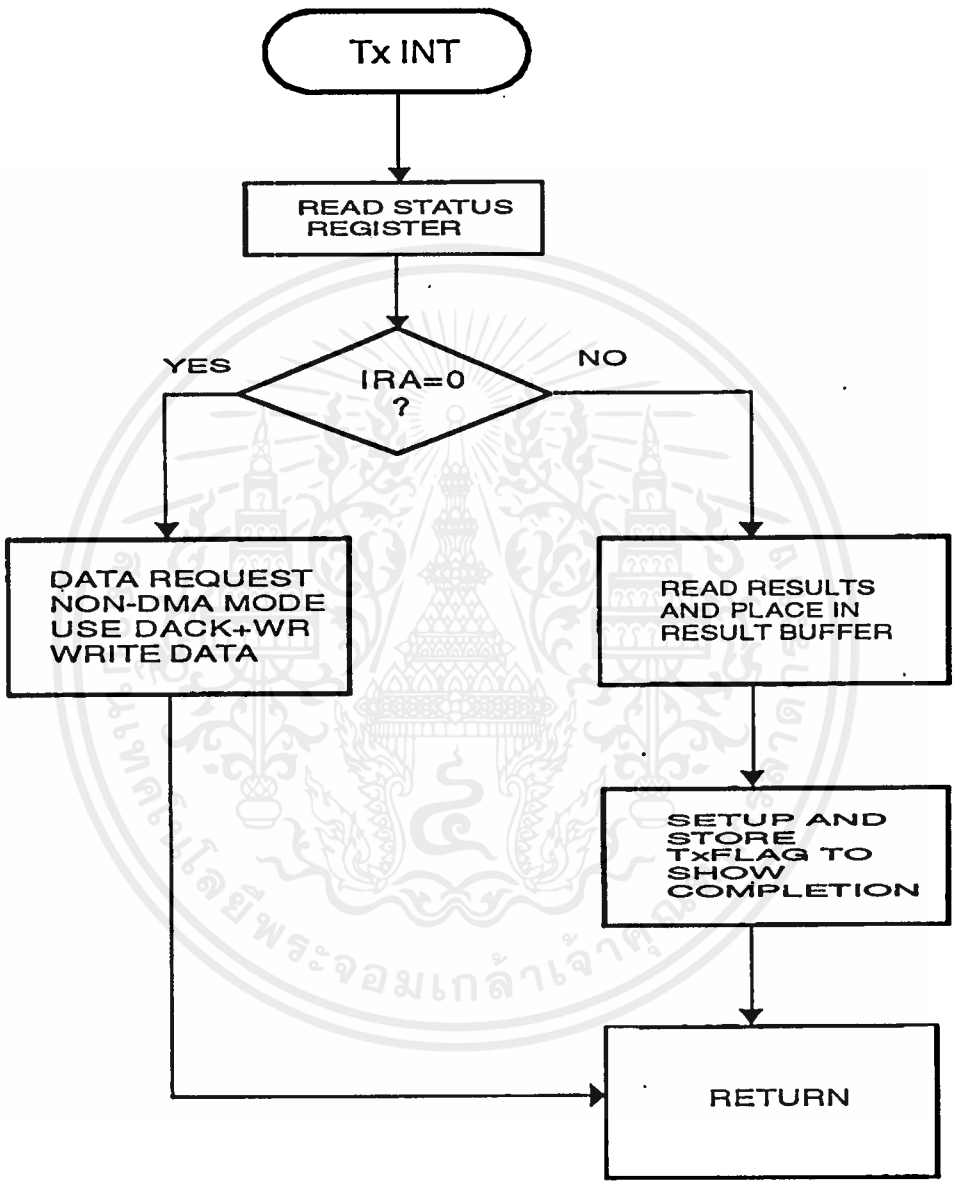
4.14fCOMM Subroutin with command Buffer Format



1.14 g



1.14 h RXI (Receiver Interrupt) Routine



4.14 i Tx Interrupt Routine

บทที่ 5
การสร้าง HDLC / X.25

5.1 การสร้างวงจร

โดยอาศัย PC XT CARD แบบสั้น ต่อตามวงจรที่ได้ออกแบบขึ้นมาตามลำดับดังนี้คือ

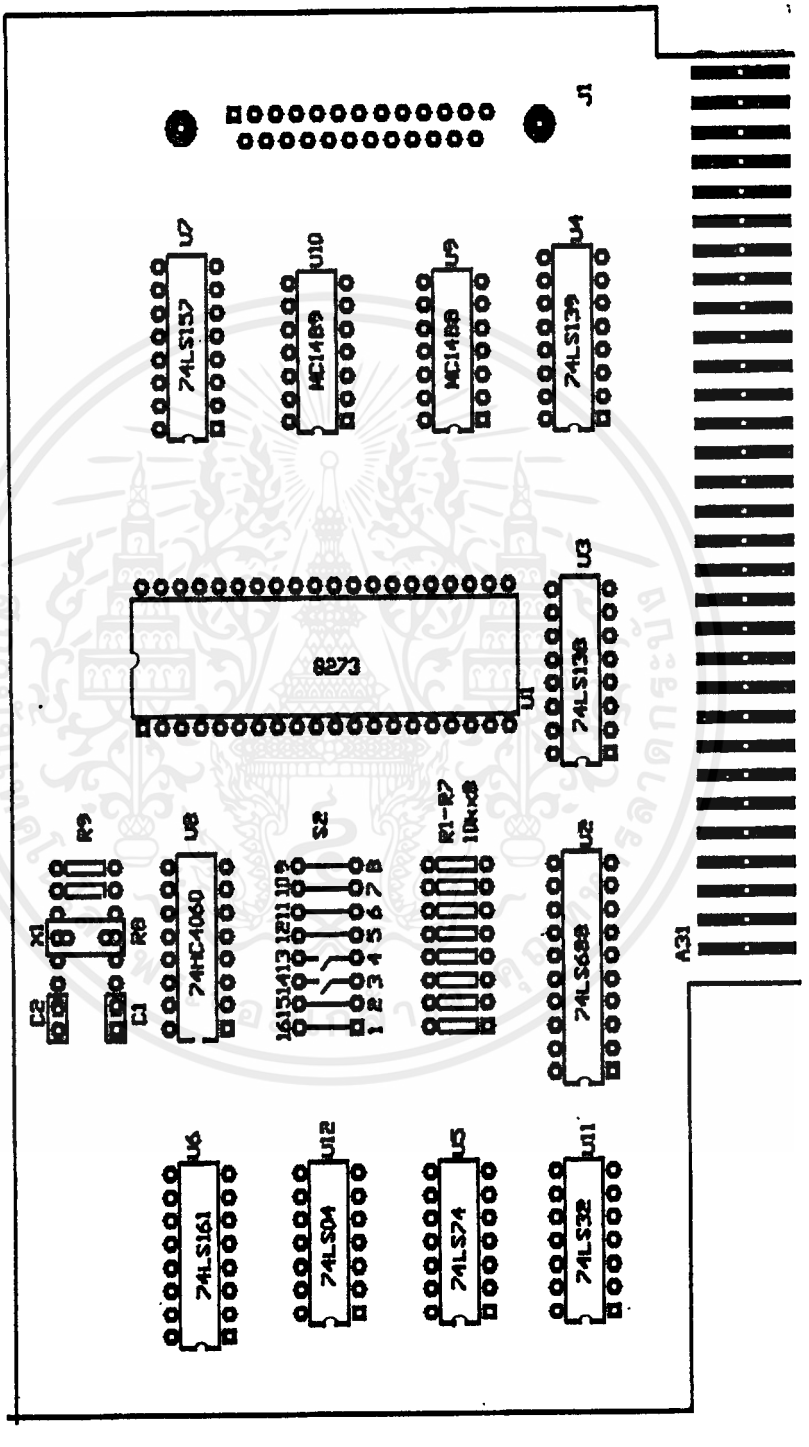
1. นำ IC Socket ชนิดรูกลมขายาว วางบนแผ่น CARD ตามรูปที่ 1 และในขณะเดียวกัน วางแผ่น Wrap - ID ด้านล่างของ CARD ให้ตรงตามขา IC Socket เพื่อให้สะดวกต่อการทำ

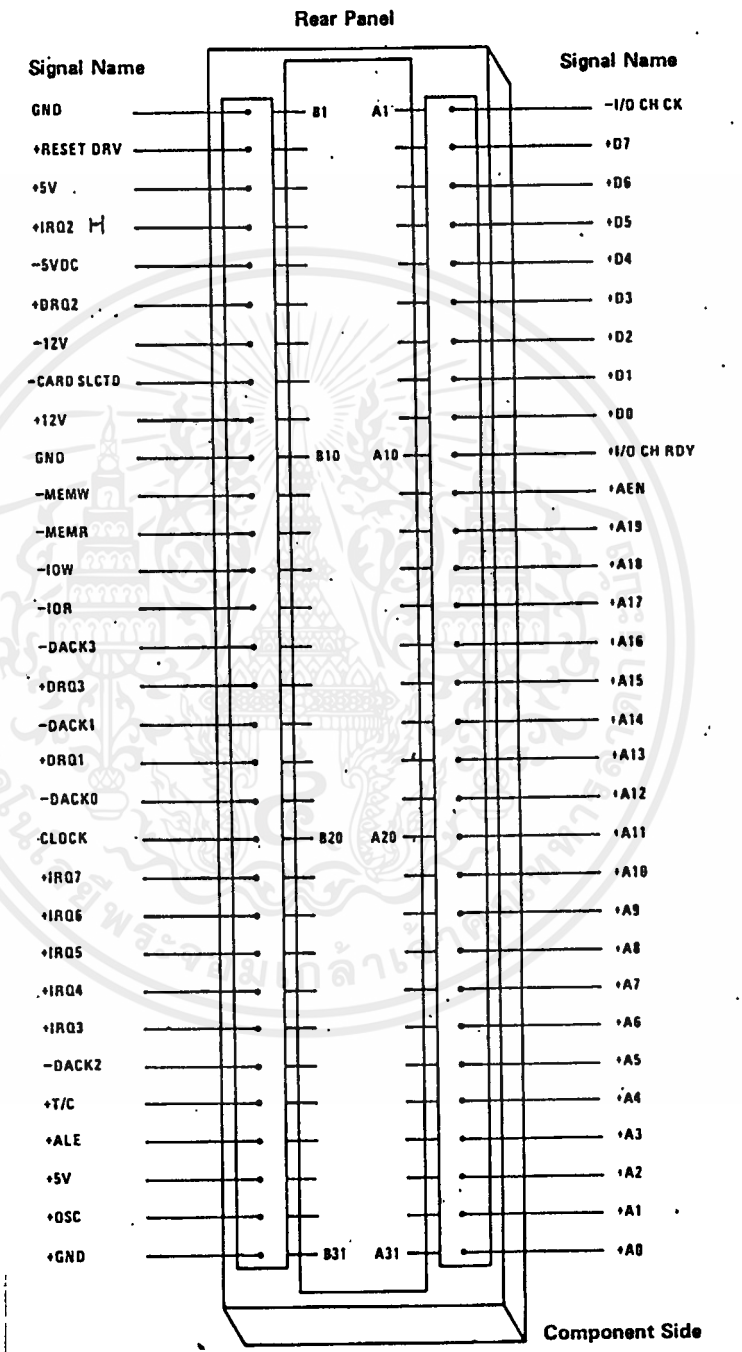
2. Wiring สาย Wire Wrapping ตามวงจร HDLC / X.25 Interface CARD ตามที่ได้ออกแบบไว้ โดยให้ สายสีแดงแทนแรงดันไฟเลี้ยง สายสีดำแทนกราวด์ นอกนั้นแทนด้วยสายสีเหลือง

3. นำอุปกรณ์ต่างๆ เสียบลงบน IC Socket ตามรูปที่ 5.1

4. ติดตั้ง J1 (DB 25)

อนึ่ง ควรสำรวจอีกครั้งทุกขั้นตอน เพื่อให้แน่ใจว่าถูกต้อง





I/O Channel Diagram

5.2 รายการอุปกรณ์

รายการอุปกณ์ ของ HDLC / X.25 Interface CARD มีดังนี้

ตัวต้านทาน ขนาด 1 / 4 วัตต์ 5 %

R1 - R8	10 K	8 ตัว	R9	10 M	1 ตัว
R10	2 K	1 ตัว			

ตัวเก็บประจุ

C1 - C2	30 pF Ceramic	2 ตัว			

อุปกณ์สารกึ่งตัวนำ

U1	8273	1 ตัว	U2	74LS688	1 ตัว
U3	74LS138	1 ตัว	U4	74LLS139	1 ตัว
U5	74LS74	1 ตัว	U6	74LS161	1 ตัว
U7	74LS157	1 ตัว	U8	74HC4060	1 ตัว
U9	MC1488	1 ตัว	U10	MC1489	1 ตัว
U11	74LS32	1 ตัว	U12	74LS04	1 ตัว

IC Socket รูกลม ชนิดขายาว

14	ขา	6 ตัว	40	ขา	6 ตัว
10	ขา	6 ตัว	20	ขา	6 ตัว

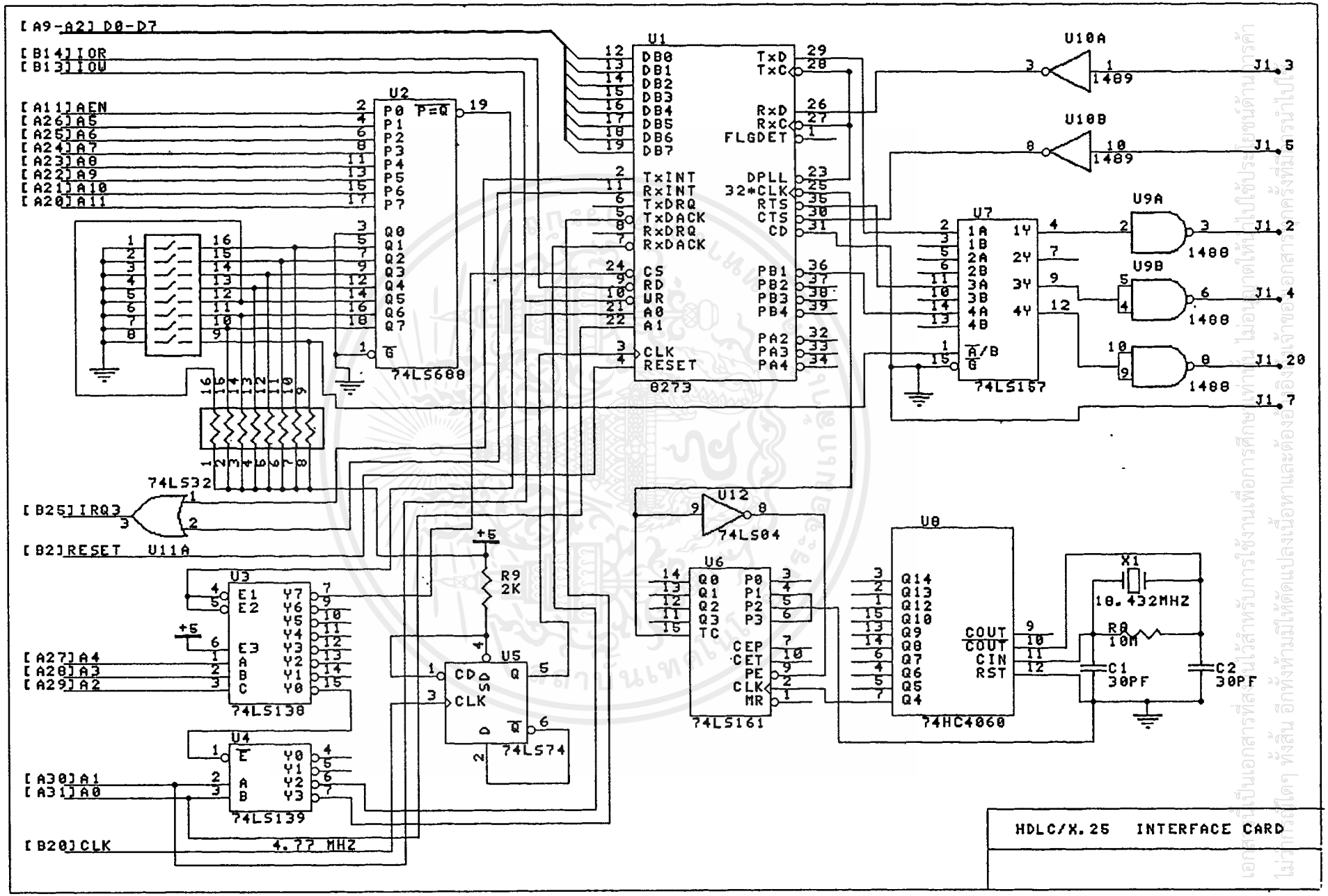
CARD XT แบบสั้น	1 แผ่น	
Wire - Wrapping Wire	3 สี	

Wrap - ID

40	ขา	6 ตัว	14	ขา	6 ตัว
16	ขา	6 ตัว	20	ขา	6 ตัว

อื่นๆ

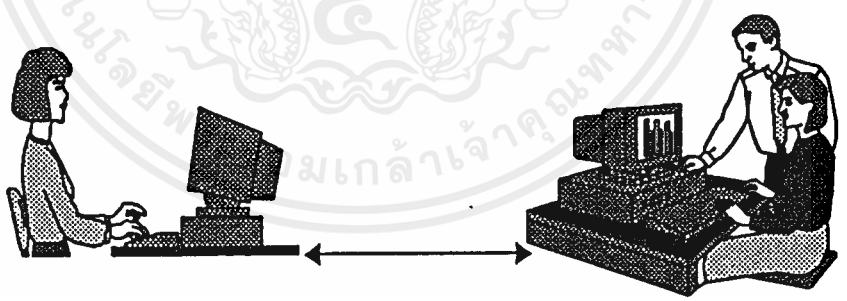
XTAL	18.432 MHz	1 ตัว	S2	DIP Switch 8 จุด	1 ตัว
J1	D325	1 ตัว			


[A9-A2] D0-D7
[B14]IOR
[B13]IOW
[A11]AEN
[A26]A5
[A25]A6
[A24]A7
[A23]A0
[A22]A9
[A21]A10
[A20]A11
U1
8273
DB0
DB1
DB2
DB3
DB4
DB5
DB6
DB7
TxINT
RxINT
TxDRQ
TxDACK
RxDRQ
RxDACK
CS
RD
WR
A0
A1
CLK
RESET
TxD
TxC
RxD
RxC
FLGDET
DPLL
32*CLK
RTS
CTS
CD
PB1
PB2
PB3
PB4
PA2
PA3
PA4
U2
74LS688
P=Q
P0
P1
P2
P3
P4
P5
P6
P7
Q0
Q1
Q2
Q3
Q4
Q5
Q6
Q7
G
U10A
U10B
1489
J1 3
J1 5
U7
74LS157
1A
1B
2A
2B
3A
3B
4A
4B
1Y
2Y
3Y
4Y
A/B
G
U9A
U9B
1408
J1 2
J1 4
J1 20
J1 7
74LS32
[B25]IRQ3
[B2]RESET U11A
U12
74LS04
U3
74LS138
E1
E2
E3
A
B
C
Y7
Y6
Y5
Y4
Y3
Y2
Y1
Y0
[A27]A4
[A28]A3
[A29]A2
+5
R9
2K
U5
74LS74
CD
SD
CLK
D
Q
U6
74LS161
Q0
Q1
Q2
Q3
TC
P0
P1
P2
P3
CEP
CET
PE
CLK
MR
U8
74HC4060
Q14
Q13
Q12
Q10
Q9
Q8
Q7
Q6
Q5
Q4
COUT
COUT
CIN
RST
X1
18.432MHZ
R8
10M
C1
30PF
C2
30PF
U4
74LS139
E
A
B
Y0
Y1
Y2
Y3
[A30]A1
[A31]A0
[B20]CLK
4.77 MHZ
HDLC/X.25 INTERFACE CARD

บทที่ 6
ตัวอย่างการใช้งาน

ตัวอย่าง การทดลอง 8273 Monitor I/O

- นำเอา HDLC/X.25 ที่สร้างขึ้นในบทที่ 5 มาเสียบลงใน I/O channel บน Main board ของไมโครคอมพิวเตอร์
- นำสายมาเชื่อมโยงทาง J1(DB25) ระหว่างเครื่องไมโครคอมพิวเตอร์
- นำเอาแผ่นโปรแกรมที่เตรียมไว้ในแผ่นดีสสอดเข้าในเครื่องขับ
- เรียกโปรแกรมชื่อ SOLF73
- แล้ว เขียนคำสั่งเริ่มต้นดังข้างล่างนี้



รูปที่ 4.22 การเชื่อมต่อแบบจุดต่อจุด

8273 MONITOR V1

- SO 05

- SD 01

- GR 00 10

- TF C2 11 "GOOD FRAME"

-

TxINT- 0D 00 00 00 00

-

RxINT- E0 0A 00 C2 34

ERROR FREE

-

"SO 05"สนับสนุนคำสั่ง SET OPERATING MODE ด้วยพารามิเตอร์ 05 ซึ่งเป็นการเซ็ท โหมด บัฟเฟอร์และ FLAG STREAM

"SD 01" Set 8273 ในโหมด INTERRUPT DATA TRANSFER ซึ่งสอดคล้องกับ IRA=0 โดยใช้คำสั่ง Set Data Transfer Mode

"GR 00 10" เป็นคำสั่ง General Receiver ด้วยขนาดบัฟเฟอร์ 0100H B0 =00, B1 = 10)

คำสั่ง TF จะเป็นสาเหตุให้ 8273 ทำการส่งเฟรม ซึ่งประกอบด้วยฟิลด์แอดเดรส C2, ฟิลด์ควบคุม 11H, ฟิลด์ข้อมูลคือ "GOOD FRAME" คำสั่ง TF มีรูปแบบพิเศษพารามิเตอร์ L0 และ L0 จะถูกคำนวณจากจำนวนไบท์ของฟิลด์ข้อมูลที่ป้อน

ภายหลังจากที่คำสั่ง TF ถูกป้อน 8273 จะส่งเฟรมหลังจากแฟล็กปิด 8273 จะอินเตอร์รัพต์ซีพียูและซีพียูจะอ่านผลลัพธ์อินเตอร์รัพต์และบรรจุมันเข้าไปในบัฟเฟอร์ ซอฟต์แวร์ จะตรวจสอบบัฟเฟอร์สำหรับผลลัพธ์อันใหม่และถ้ามีผลลัพธ์ใหม่ปรากฏอยู่ แหล่งของอินเตอร์รัพต์จะถูกแสดงผลและตามด้วยผลลัพธ์

ในตัวอย่างนี้ ผลลัพธ์ 0DH หมายถึง อินเตอร์รัพต์สิ้นสุดเฟรมด้วยเฟรมที่สมบูรณ์ สำหรับอินเตอร์รัพต์ภาคส่งจะมีเฉพาะ 1 ผลลัพธ์เท่านั้น สำหรับผลลัพธ์ศูนย์ที่ต่อท้ายนั้น ได้รวมเอาไว้เพื่อง่ายต่อการเขียนโปรแกรม

ต่อไปคือ การรับเฟรมผลลัพธ์อินเตอร์รัพต์จะถูกแสดงผลตามลำดับที่อ่านมาจาก 8273 E0H หมายถึง อินเตอร์รัพต์ General Receive ด้วยไบท์สุดท้ายของฟิลด์ข้อมูลที่รับได้ จำนวน 8 บิท ผลลัพธ์ 0A 00 (R0 , R1) แสดงถึง มีขนาด 10 ไบท์ของฟิลด์ข้อมูลที่รับได้ ส่วนที่เหลือคือ C2H ฟิลด์แอดเดรสและ 34H ฟิลด์ควบคุม 10 ไบท์ของฟิลด์ข้อมูลจะถูกนำมาแสดงผลในบรรทัดต่อไป

จะเห็นได้ว่า ชิป 8273 นั้นมีคำสั่งจำนวนมากมายที่สนับสนุน ทำให้สามารถนำเอา 8273 ไปประยุกต์ใช้งานได้อย่างมากมาย โดยเฉพาะอย่างยิ่งงานทางด้านสื่อสาร ข้อมูลที่ใช้โพรโทคอล HDLC/SDLC จัดได้ว่าเป็นชิปที่น่าสนใจมาก

บทที่ 7

บทสรุปและวิจารณ์

ผลจากการศึกษาและทดลองการทำงานของ X.25 / HDLC CARD ในโหมด non -DMA นี้ การถ่ายโอนข้อมูลจากบัฟเฟอร์ไปยัง RS - 232 line หรือ จาก RS-232 line ไปยังบัฟเฟอร์โดยอาศัยฮาร์ดแวร์อินเตอร์รัพต์ผ่าน IRQ3 นั้น การถ่ายโอนข้อมูลระหว่างซีพียูกับชิป 8273 ในด้านความเร็วนั้นยังไม่ได้ถึงจุดที่ดีที่สุด และการอินเตอร์รัพต์ได้นำเอาทั้งเส้นสัญญาณ RxINT และ TxINT มาร่วมผ่าน OR เกท ไม่ได้แยกโดยอิสระ ทำให้การทำงานรองรับได้เฉพาะ ฮาร์ฟดูเพล็ก การถ่ายโอนข้อมูลแต่ละครั้ง จะทำให้ซีพียูต้องรับภาระมาก

ดังนั้นถ้าหากท่านสนใจที่จะศึกษาเกี่ยวกับเรื่องนี้ พวกกระผมขอแนะนำให้ศึกษาและเลือกโหมด DMA ซึ่ง ชิป 8273 นี้มีโหมดการทำงานถ่ายโอนข้อมูลโดยผ่านทาง DMA รวมทั้งชิปได้ออกแบบมาสนับสนุนโหมดนี้ด้วย ซึ่งจะทำให้การถ่ายโอนข้อมูลได้ถึงจุดที่ดีที่สุด

```
;*************************************************************************
;Command   supported   are:
;                              RS - reset serial  I/O mode
;                              SS - set serial  I/O mode
;                              RO - reset operating mode
;                              SO - set operating mode
;                              RD - receiver disable
;                              GR - general receive
;                              SR - serective receive
;                              TF - transmit frame
;                              AF - abort frame
;                              SP - set port B
;                              RP - reset port B
;                              RB - reset one bit delay (PAR = 7F)
;                              SB set one bit delay (PAR = 80)
;                              SL - selective loop receive
;                              TL - transmit loop
;                              Z  - change modes flip/flop
;*************************************************************************
;*************************************************************************

;   Polled mode: When polled mode is selected (denoted by a '+' prompt), if
;            a SNRM-P or RR(0)-P is received, a response frame of NSA-F
;            or RR(0)-F is transmitted. Other commands operate normally.

;*************************************************************************

;*******************************STACK*************************************
STACK           SEGMENT        PARA  STACK 'stack'
```

```
                DW          64 DUP (0)
STACK           ENDS
;*************************************************************************

;*****************************DATA*************************************

DATA            SEGMENT     PARA  PUBLIC  'data'
CMDBUF          DB          16  DUP (0)    ;Command buffer
CMDBF1          DB          7   DUP (0)    ;Response command buffer for poll
;Mode
RESBUF          DB          256 DUP (0)    ;Result buffer
TXBUF           DB          512 DUP (0)    ;Tx buffer
RXBUF           DB          512 DUP (0)    ;Rx buffer
TXFLAG          DB          0              ;Tx FLAG
RXFLAG          DB          0              ;Rx FLAG
PRMPT           DB          0              ;Prompt character storage
POLIN           DB          0              ;Poll mode indicator
SIGNON          DB          CR,'8273 MONITOR V 1',CR,'$'
RXIMSG          DB          CR,'RxINT -','$'
TXIMSG          DB          CR,'TxINT -','$'
INT_NUM         DB          0BH            ;Offset in BIOS 0BH*4 = 2C
O_INT_SEG       DW          0              ;Segment
O_INT_OFF       DW          0              ;Offset
LDADR           DW          0              ;Result buffer load pointer storage
CNADR           DW          0              ;Result buffer console pointer storage
TXPNT           DW          0              ;Tx data pointer
RXPNT           DW          0              ;Rx data pointer
CR              EQU         0DH            ;ASCII character
LF              EQU         0AH            ;ASCII character
```

```
SNRMP     EQU     93H     ;SNRM-P control code
RR0P      EQU     11H     ;RR(0)P control code
NSAF      EQU     73H     ;NSA-F  control code
RR0F      EQU     11H     ;RR(0)-F control code
CNTLC     EQU     03H     ;CNTL-C equivalent

          ; 8273 equates

;STAT73   EQU     031CH   ;Status register
COMM73    EQU     031CH   ;Command register
PARM73    EQU     031DH   ;Parameter register
RESL73    EQU     031DH   ;Result register
TXIR73    EQU     031EH   ;Tx interrupt result register
RXIR73    EQU     031FH   ;Rx interrupt result register
TEST73    EQU     031EH   ;Test mode register
CPBF      EQU     20H     ;Parameter buffer full bit
TXINT     EQU     04H     ;Tx interrupt bit in status register
RXINT     EQU     08H     ;Rx interrupt bit in status register
TXIRA     EQU     01H     ;TxINT result availble bit
RXIRA     EQU     02H     ;RxINT result available bit
INT       EQU     0CH     ;TxINT and RxINT

          ; 8253 equates

MODE53    EQU     0313H   ;8253 mode word register
CNT053    EQU     0310H   ;Counter 0 register
MDCNT0    EQU     36H     ;Mode for counter 0
MDCNT2    EQU     0B6H    ;Mode for counter 2
LKBR1     DB      0       ;8273 baud rate LSB ADR
```

```
LKBR2       DB          0           ;8273 baud rate MSB ADR

; Other direct port

TXDACK      EQU         0303H       ;TX DACK port
RXDACK      EQU         0302H       ;RX DACK port
IN245       EQU         0301H       ;In direct port
OUT373      EQU         0300H       ;Out direct port
DATA        ENDS

;*****************************CODE*****************************************

CODE        SEGMENT     PARA  PUBLIC  'code'
            ASSUME      CS:CODE,DS:DATA,ES:DATA,SS:STACK

START:                                ;Program entry point
            MOV         AX,DATA
            MOV         DS,AX
            MOV         ES,AX         ;DS,ES-->data segment
            MOV         AL,MDCNT0     ;8253 mode set
            MOV         DX,MODE53
            OUT         DX,AL         ;8253 mode port
            MOV         AL,LKBR1      ;Get 8273 baud rate LSB
            MOV         DX,CNT053
            OUT         DX,AL         ;Using counter 0 as baud rate gen
            MOV         AL,LKBR2      ;Get 8273 baud rate MSB
            MOV         DX,CNT053
            OUT         DX,AL         ;Counter 0
            MOV         AL,01H        ;Output 1 followed by a 0
```

```
        MOV     DX,TEST73
        OUT     DX,AL           ;To test mode register
        MOV     AL,00H          ;To reset the 8273
        MOV     DX,TEST73
        OUT     DX,AL
        MOV     AL,'-'          ;Normal mode prompt char.
        MOV     PRMPT,AL        ;Put in storage
        MOV     AL,00H          ;Tx poll response indicator
        MOV     POLIN,AL        ;0 means no special Tx
        MOV     DX,OFFSET SIGNON  ;Signon message ADR
        CALL    TYMSG           ;Display signon
;*************************************************************************
;               Save/install interrupt      *
;*************************************************************************
        MOV     AH,35H          ;DOS service request number
        MOV     AL,INT_NUM      ;Interrupt number (0BH)
        INT     21H
;ES:BX = segment/offset of original handler
        MOV     O_INT_SEG,ES    ;Save segment
        MOV     O_INT_OFF,BX    ;and offset
;Insert address of the interrupt service routine in the BIOS
;label for interrupt service routine is: RS232_INT
        MO      AH,25H          ;Dos service request number
        MOV     AL,INT_NUM      ;Machine interrupt number
        MOV     DX,OFFSET CS:RS232_INT
        PUSH    DS              ;Save program data segment
        PUSH    CS
        POP     DS              ;Set DS to segment base of
```

```
        ;interrupt service routine
                INT             21H
                POP             DS              ;Restore program's DS
        ;Enable communication interrupt by resetting the bit corresponding to the IRQ3
        ;on the interrupt mask register  (PORT ADDRESS 21H)
;               CLI                             ;Interrupts OFF
        ;Reset buffer pointers to start  of  buffer
                MOV             LDADR,0
                MOV             CNADR,0
;*******************************************************************************
                IN              AL,21H          ;Read byte at portT
                JMP             SHORT $+2       ;I/O delay
                AND             AL,0F7H         ;Reset bit 3
                OUT             21H,AL
                JMP             SHORT $+2       ;I/O delay
                STI                             ;Reenable interrupt
                CALL            FLUSH           ;Flush keyboard buffer

;Main program loop checks for change in result pointers,USART status,
;or poll status

CMDREC:
                CALL            CRLF            ;Display CR
                MOV             AL,PRMPT        ;Get current prompt char.
                CALL            ECHO            ;Display it

LOOPIT:
                MOV             AH,1            ;Code for read keyboard status
                INT             16H             ;BIOS service
```

```
                JZ          SER_IMP     ;Nothing in keyboard buffer
                JMP         GETCMD      ;Character in keyboard buffer

        ;Delay loop to allow interrupt to occur
SER_IMP:
                STI                     ;Interrupts on
                MOV         CX,50
DELAY:
                NOP
                NOP
                LOOP        DELAY
;*************************************************************************
;               ***** test for new data received *****
;*************************************************************************
                CLI                     ;Interrupt Off while reading pointer
                MOV         BX,CNADR    ;Compare pointers
                CMP         BX,LDADR
                JNE         DISPY_1     ;New data item or items
                STI                     ;Interrupts on
                MOV         AL,POLIN    ;Get poll mode status
                AND         AL,AL       ;Is it 0?
                JZ          LOOPIT      ;Yes,try again
                JMP         TXPOL       ;No,then poll occurred
DISPY_1:
                JMP         DISPY

        ;command recognizer routine

GETCMD:         CALL        GETCH       ;Get character into AL
```

```
                CALL        ECHO        ;Echo it
                CMP         AL,'R       ;R?
                JNE         GETCMD1     ;Get more
                JMP         RDWN
GETCMD1:
                CMP         AL,'S       ;S?
                JNE         GETCMD2
                JMP         SDWN        ;Get  more
GETCMD2:
                CMP         AL,'G       ;G?
                JNE         GETCMD3
                JMP         GDWN        ;Get more
GETCMD3:
                CMP         AL,'T'      ;T?
                JNE         GETCMD4
                JMP         TDWN        ;Get more
GETCMD4:
                CMP         AL,'A'      ;A?
                JNE         GETCMD5
                JMP         ADWN        ;Get more
GETCMD5:
                CMP         AL,'Z'      ;Z?
                JNE         GETCMD6
                JMP         CMODE       ;Yes,go change mode
GETCMD6:
                CMP         AL,CNTLC    ;Cntl-c?
                JNE         ILLEG
                JMP         DOS_EXIT    ;Exit to monitor
ILLEG:
```

```
                MOV     AL,'?'      ;Print?
                CALL    ECHO        ;Display it
                JMP     CMDREC      ;Loop for command

        ;exit to monitor

DOS_EXIT:
                CALL    COMM_OFF    ;Communications interrupts off
                MOV     AH,25H      ;DOS service request number
                MOV     AL,INT_NUM  ;Machine interrupt number
                MOV     DX,O_INT_SEG ;Segment to DS
                MOV     DS,DX
                MOV     DX,O_INT_OFF ;Offset to DX
                INT     21H
                MOV     AH,4CH      ;DOS service request number
                MOV     AL,0        ;No return code
                INT     21H         ;Exit to DOS

RDWN:
                CALL    GETCH       ;Get next char.
                CALL    ECHO        ;Echo it
                CMP     AL,'O'      ;O?
                JNE     RDWN1
                JMP     ROCMD       ;RO command
RDWN1:
                CMP     AL,'S'      ;S?
                JNE     RDWN2
                JMP     RSCMD       ;RS command
RDWN2:
```

```
                CMP     AL,'D'      ;D?
                JNE     RDWN3
                JMP     RDCMD       ;RD command
RDWN3:
                CMP     AL,'P'      ;P?
                JNE     RDWN4
                JMP     RPCMD       ;RP command
RDWN4:
                CMP     AL,'R'      ;R?
                JNE     RDWN5
                JMP     START       ;Start over
RDWN5:
                CMP     AL,'B'      ;B?
                JNE     RDWN6
                JMP     RBCMD       ;RB command
RDWN6:
                JMP     ILLEG       ;Illegal try again

SDWN:
                CALL    GETCH       ;Get next char.
                CALL    ECHO        ;Echo it
                CMP     AL,'O'      ;O?
                JNE     SDWN1
                JMP     SOCMD       ;SO command
SDWN1:
                CMP     AL,'S'      ;S?
                JNE     SDWN2
                JMP     SSCMD       ;SS command
SDWN2:
```

```
                CMP     AL,'R'      ;R?
                JNE     SDWN3
                JMP     SRCMD       ;SR command
SDWN3:
                CMP     AL,'P'      ;P?
                JNE     SDWN4
                JMP     SPCMD       ;SP command
SDWN4:
                CMP     AL,'B'      ;B?
                JNE     SDWN5
                JMP     SBCMD       ;SB command
SDWN5:
                CMP     AL,'L'      ;L?
                JNE     SDWN6
                MP      CMD         ;SL command
DWN6:
                CMP     AL,'D'      ;D?
                JNE     SDWN7
                JMP     SDCMD       ;SD command
SDWN7:
                JMP     ILLEG       ;Illegal try again

GDWN:
                CALL    GETCH       ;Get next char.
                CALL    ECHO        ;Echo it
                CMP     AL,'R'      ;R?
                JNE     GDWN1
                JMP     GRCMD       ;GR command
GDWN1:
```

```
                JMP         ILLEG       ;Illegal try again

TDWN:
                CALL        GETCH       ;Get next char.
                CALL        ECHO        ;Echo it
                CMP         AL,'F'      ;F?
                JNE         TDWN1
                JMP         TFCMD       ;TF command
TDWN1:
                CMP         AL,'L'      ;L?
                JNE         TDWN2
                JMP         TLCMD       ;TL command
TDWN2:
                JMP         ILLEG       ;Illegal try again

ADWN:
                CALL        GETCH       ;Get next char.
                CALL        ECHO        ;Echo it
                CMP         AL,'F'      ;F?
                JNE         ADWN1
                JMP         AFCMD       ;AF command
ADWN1:
                JMP         ILLEG       ;Illegal try again

        ;reset poll mode response-change prompt char. as indicator

CMODE:
                CLI                     ;Disable interrupts
                MOV         AL,PRMPT    ;Get current prompt
```

```
                CMP         AL,'-'          ;Normal mode?
                JNZ         SW              ;No,change it
                MOV         AL,'+'          ;New prompt
                MOV         PRMPT,AL        ;Store new prompt
                STI                         ;Enable interrupts
                JMP         CMDREC          ;Return to loop
SW:
                MOV         AL,'-'          ;New prompt char.
                MOV         PRMPT,AL        ;Store it
                STI                         ;Enable interrupts
                JMP         CMDREC          ;Return to loop

        ;transmit an6swer to poll setup

TXPOL:
                MOV         AL,00H          ;clear poll indicator
                MOV         POLIN,AL        ;indicator ADR
                MOV         CH,04H          ;get # of parameters ready
                LEA         BX,CMDBF1       ;point to special buffer
                CALL        COMM2           ;jump to command output
                JMP         LOOPIT
;*************************************************************************
;********************** command implementing routines **********************
;*************************************************************************

        ;RO-reset operating mode

ROCMD:
                MOV         CH,01H          ;# of parameter
```

```
        MOV     CL,51H      ;command
        CALL    COMM        ;get parameter and issue command
        JMP     CMDREC      ;get next command

    ;RS-reset serial I/O mode command

RSCMD:
        MOV     CH,01H      ;# of parameter
        MOV     CL,60H      ;command
        CALL    COMM        ;get parameter and issue command
        JMP     CMDREC      ;get next command

    ;RD-receiver disable command

RDCMD:
        MOV     CH,00H      ;# of parameter
        MOV     CL,0C5H     ;command
        CALL    COMM        ;issue command
        JMP     CMDREC      ;get next command

    ;RB-reset one bit delay command

RBCMD:
        MOV     CH,01H      ;# of parameter
        MOV     CL,64H      ;command
        CALL    COMM        ;get parameter and issue command
        JMP     CMDREC      ;get next command
```

```
        ;SB-set one bit delay command

SBCMD:
                MOV     CH,01H      ;# of parameter
                MOV     CL,0A4H     ;command
                CALL    COMM        ;get parameter and issue command
                JMP     CMDREC      ;get next command

        ;SD-set data transfer command

SDCMD:
                MOV     CH,01H      ;# of parameter
                MOV     CL,97H      ;command
                CALL    COMM        ;get parameter and issue command
                JMP     CMDREC      ;get next command

        ;SL-selective loop receive command

SLCMD:
                MOV     CH,04H      ;# of parameter
                MOV     CL,0C2H     ;command
                CALL    COMM        ;get parameter and issue command
                JMP     CMDREC      ;get next command

        ;TL-transmit loop command

TLCMD:
                LEA     BX,CMDBUF   ;set command buffer pointer
                MOV     CH,02H      ;load parameter counter
```

```
        MOV     BYTE PTR[BX],0CAH  ;load command into buffer
        LEA     BX,CMDBUF+2  ;point at ADR and CNTL positions
        JMP     TFCMD1

    ;g mode command

SOCMD:
        MOV     CH,01H      ;# of parameter
        MOV     CL,91H      ;command
        CALL    COMM        ;get parameters and issue command
        JMP     CMDREC      ;get next command

    ;SS-set serial I/O command

SSCMD:
        MOV     CH,01H      ;# of parameter
        MOV     CL,0A0H     ;command
        CALL    COMM        ;get parameter and issue command
        JMP     CMDREC      ;get next command

    ;SR-selective receive command

SRCMD:
        MOV     CH,04H      ;# of parameters
        MOV     CL,0C1H     ;command
        CALL    COMM        ;get parameters and issue command
        JMP     CMDREC      ;get next command
```

```
        ;GR-general receive command

GRCMD:
                MOV     CH,02H      ;# of parameters
                MOV     CL,0C0H     ;command
                CALL    COMM        ;get parameters and issue command
                JMP     CMDREC      ;get next command

        ;AF-abort frame command

AFCMD:
                MOV     CH,00H      ;no parameter
                MOV     CL,0CCH     ;command
                CALL    COMM        ;issue command
                JMP     CMDREC      ;get next command

        ;RP-reset port command

RPCMD:
                MOV     CH,01H      ;# of parameter
                MOV     CL,63H      ;command
                CALL    COMM        ;get parameter and issue command
                JMP     CMDREC      ;get next command

        ;SP-set port command

SPCMD:
                MOV     CH,01H      ;# of parameter
                MOV     CL,0A3H     ;command
```

```
                CALL    COMM        ;get parameter and issue command
                JMP     CMDREC      ;get next command

        ;TF-transmit frame command

TFCMD:
                LEA     BX,CMDBUF   ;set command buffer pointer
                MOV     CH,02H      ;load parameter counter
                MOV     BYTE PTR[BX],0C8H   ;load command into buffer
                LEA     BX,CMDBUF+2 ;point at  ADR and CNTL positions
TFCMD1:
                MOV     AL,CH       ;test parameter count
                AND     AL,AL       ;is it  0?
                JZ      TBUFL       ;yes,load Tx data buffer
                CALL    PARIN       ;get parameter
                JC      ILLEG2      ;illegal char. returned
                INC     BX          ;INC command buffer pointer
                DEC     CH          ;DEC parmeter counter
                MOV  BYTE PTR[BX],AL ;load parameter into command buffer
                JMP     TFCMD1      ;go next parameter
TBUFL:
                LEA     BX,TXBUF    ;load Tx data buffer pointer
                MOV     CX,0000H    ;clear CX-byte counter
TBUFL1:
                PUSH    CX          ;save byte counter
                CALL    PARIN       ;get data,alias parameter
                JC      ENDCHK      ;maybe end if illegal
                MOV     BYTE PTR[BX],AL ;load data byte into buffer
                INC     BX          ;INC  buffer pointer
```

```
                POP     CX          ;restore byte counter
                INC     CX          ;INC byte count
                JMP     TBUFL1      ;get next data
ENDCHK:
                CMP     AL,CR       ;returned illegal char. CR?
                JZ      TBUFFL      ;yes,then Tx buffer full
                POP     CX          ;restore CX to save stack
                JMP     ILLEG       ;illegal char.
TBUFFL:
                POP     CX          ;restore byte counter
                LEA     BX,CMDBUF+1 ;point into command buffer
                MOV     BYTE PTR[BX],CL ;store byte count LSB
                INC     BX          ;INC pointer
                MOV     BYTE PTR[BX],CH ;store byte count MSB
                MOV     CH,04H      ;load parameter counter
                LEA     BX,CMDBUF   ;repoint pointer
                CALL    COMM2       ;issue command
                JMP     CMDREC      ;get next command
ILLEG2:
                JMP     ILLEG
```

```
        ;routine to display result in result buffer when load and console pointers
        ;are different

DISPY:
                MOV     CH,05H      ;CH is result counter
                LEA     SI,RESBUF   ;Result buffer address
                MOV     BX,CNADR    ;Console pointer
                ADD     SI,BX       ;Buffer start + displacement
```

```
        MOV     AL,BYTE PTR[SI] ;Get result interrupt code
        AND     AL,1FH          ;Limit to result code
        CMP     AL,0CH          ;Test if RX or TX source
        JC      RXSORC          ;Carry,then RX source
TXSORC:
        MOV     DX,OFFSET TXIMSG    ;TxINT message
        CALL    TYMSG           ;Display it
DISPY1:
        MOV     BX,CNADR        ;Get console pointer
        LEA     SI,RESBUF       ;Result buffer address
        ADD     SI,BX           ;Buffer start + displacement
        MOV     AL,BYTE PTR [SI] ;Get result
        CALL    NMOUT           ;Convert and display
        MOV     AL,' '          ;SP char.
        CALL    ECHO            ;Display it
        INC     BX              ;Bump
        CMP     BX,256          ;Pointer overflows buffer?
        JNE     DISPY2
        MOV     BX,0            ;Reset to start of buffer
DISPY2:
        MOV     CNADR,BX        ;Update
        DEC     CH              ;Dec result counter
        JNZ     DISPY1          ;Not, done
        STI
        JMP     CMDREC          ;Return to loop

;receiver source-display result and receive buffer contents
```

```
RXSORC:
        MOV     DX,OFFSET RXIMSG  ;RxINT message
        CALL    TYMSG             ;Display message
RXS1:
        MOV     BX,CNADR          ;Get console pointer
        LEA     SI,RESBUF         ;Result buffer address
        ADD     SI,BX             ;Buffer start + displacement
        MOV     AL,BYTE PTR[SI]   ;Retrieve result from buffer
        CALL    NMOUT             ;Convert and display it
        MOV     AL,' '            ;ASCII SP
        CALL    ECHO              ;Display it
        INC     BX                ;INC console pointer
        CMP     BX,256            ;Pointer overflows buffer?
        JNE     RXS6
        MOV     BX,0              ;Reset to start of buffer
RXS6:
        MOV     CNADR,BX          ;Update
        DEC     CH                ;DEC result counter
        MOV     AL,CH             ;Get set to test counter
        CMP     AL,04H            ;Is the result R0?
        JZ      R0PT              ;Yes,go save it
        CMP     AL,03H            ;Is the result R1?
        JZ      R1PT              ;Yes,go save it
RXS2:
        AND     AL,AL             ;Test result counter
        JNZ     RXS1              ;Not done yet,get next result
        CALL    CRLF              ;Display CR
        POP     DX                ;Retrieve received count
```

```
RXS3:
        LEA     DI,RXBUF        ;Pointer at Rx buffer
        MOV     AL,DH           ;Is count  0?
        OR      AL,DL
        JZ      RXS4            ;Yes,go back to loop
        MOV     BX,RXPNT        ;Get data pointer
        ADD     DI,BX           ;Data buffer start + displacement
        MOV     AL,BYTE PTR[DI]  ;No,get char.
        PUSH    DX              ;Save DX
        CALL    NMOUT           ;Convert and display char.
        MOV     AL,' '          ;ASCII SP
        CALL    ECHO            ;Display it to separate data
        POP     DX              ;Restore DX
        DEC     DX              ;DEC count
        INC     BX              ;INC pointer
        CMP     BX,512          ;Pointer overflows buffer?
        JNE     RXS5
        MOV     BX,0            ;Reset to start of buffer
RXS5:
        MOV     RXPNT,BX        ;Update
        JMP     RXS3            ;Get next  char.
ROPT:
        LEA     SI,RESBUF       ;Result buffer address
        ADD     SI,BX           ;Buffer start + displacement
        MOV     DL,BYTE PTR[SI]  ;Get R0 for result buffer
        PUSH    DX              ;Save it
        JMP     RXS2            ;Return
R1PT:
        POP     DX              ;Get R0
```

```
                LEA     SI,RESBUF       ;Result buffer address
                ADD     SI,BX           ;Buffer start + displacement
                MOV     DH,BYTE PTR[SI]  ;Get R1 for result buffer
                PUSH    DX              ;Save it
                JMP     RXS2            ;Return
RXS4:
                STI
                JMP     CMDREC          ;Return to loop

;*************************************************************************
;                       **********procedure**********
;*************************************************************************

PARIN
                PROC    NEAR
                PUSH    CX              ;Save CX
                MOV     DH,01H          ;Set char. counter
                CALL    GETCH           ;Get char.
                CALL    ECHO            ;Echo it
                CMP     AL,' '          ;SP?
                JNZ     PARIN1          ;No,illegal,try again
PARIN3:
                CALL    GETCH           ;Get char. of parameter
                CALL    ECHO            ;Echo it
                CALL    VALDG           ;Is it a valid char?
                JNC     PARIN1          ;No,try again
                CALL    CNVBN           ;Convert it to HEX
                MOV     CL,AL           ;Save it in CL
                MOV     AL,DH           ;Get char.counter
```

```
                AND             AL,AL           ;Is it 0?
                JZ              PARIN2          ;Yes done with this parameter
                DEC             DH              ;DEC char. counter
                XOR             AL,AL           ;Clear carry
                MOV             AL,CL           ;Recover 1 st char.
                ROL             AL,1            ;Rotate left 4 places
                ROL             AL,1
                ROL             AL,1
                ROL             AL,1
                MOV             DL,AL           ;Save it in DL
                JMP             PARIN3          ;Get next char.
PARIN2:
                MOV             AL,CL           ;2 nd char. in AL
                OR              AL,DL           ;Combine both chars.
                POP             CX              ;Restore CX
                RET                             ;Return to calling program
PARIN1:
                STC                             ;Set carry as illegal status
                POP             CX              ;Restore CX
                RET
PARIN           ENDP

;*************************************************************************
                ;   comm procedure      *
;*************************************************************************

COMM            PROC            NEAR
                LEA             BX,CMDBUF       ;Set pointer
                PUSH            CX              ;Save CX
```

```
                MOV         BYTE PTR[BX],CL   ;Load command in buffer
COMM1:
                MOV         AL,CH             ;Check parameter counter
                AND         AL,AL             ;Is it 0?
                JZ          CMDOUT            ;Yes,go issue command
                CALL        PARIN             ;Get parameter
                JC          ILLEG1            ;Illegal char.returned
                INC         BX                ;INC buffer pointer
                DEC         CH                ;DEC parameter counter
                MOV         BYTE PTR[BX],AL   ;Parameter to buffer
                JMP         COMM1             ;Get next parameter
CMDOUT:
                LEA         BX,CMDBUF         ;Set pointer again
                POP         CX
                CALL        COMM2             ;Issue command
                RET
ILLEG1:
                JMP         ILLEG
COMM            ENDP

;*************************************************************************
;                       issue command   *
;*************************************************************************

COMM2           PROC        NEAR
COMM3:
                MOV         DX,STAT73
                IN          AL,DX             ;Read 8273 status
                JMP         SHORT $+2         ;I/O delay
```

```
        RCL     AL,1            ;Rotate CBSY into carry
        JC      COMM3           ;Wait for ok
        MOV     AL,BYTE PTR[BX]  ;Ok,move command into AL
        MOV     DX,COMM73
        OUT     DX,AL           ;Output command
        JMP     SHORT $+2       ;I/O delay
PAR1:
        MOV     AL,CH           ;Get parameter counter
        AND     AL,AL           ;Is it 0?
        JZ      PAR3            ;Yes done return
        INC     BX              ;INC command buffer pointer
        DEC     CH              ;DEC parameter counter
PAR2:
        MOV     DX,STAT73
        IN      AL,DX           ;Read status
        JMP     SHORT $+2       ;I/O delay
        AND     AL,CPBF         ;Is CPBF bit set?
        JNZ     PAR2            ;Wait til it 0
        MOV     AL,BYTE PTR[BX]  ;Ok get parameter from buffer
        MOV     DX,PARM73
        OUT     DX,AL           ;Output parameter
        JMP     SHORT $+2       ;I/O delay
        JMP     PAR1            ;Get next parameter
PAR3:
        RET
COMM2   ENDP
```

```
;*************************************************************************
;                          typed message procedure      *
;*************************************************************************

TYMSG       PROC        NEAR
            MOV         AH,9            ;Service request number
            INT         21H             ;DOS interrupt
            RET
TYMSG       ENDP

;*************************************************************************
;                          flush keyboard procedure      *
;*************************************************************************

FLUSH       PROC        NEAR
FLUSH_1:
            MOV         AH,1            ;BIOS service request code
            INT         16H
            JZ          FLUSH_2
            MOV         AH,0            ;Flush old character
            INT         16H
            JMP         FLUSH_1
FLUSH_2:
            RET
FLUSH       ENDP
```

```
;*************************************************************************
;                              echo procedure    *
;*************************************************************************

ECHO        PROC        NEAR
            PUSH        DX
            MOV         DL,AL          ;Save character
            MOV         AH,02H         ;DOS service request number
            INT         21H            ;DOS service request
            POP         DX
            RET
ECHO        ENDP

;*************************************************************************
;                        valid digit character     *
;*************************************************************************
VALDG       PROC        NEAR
            CMP         AL,'0'         ;Is it a legal digit?
            JB          BAD_DIGIT      ;Nope
            CMP         AL,'9'         ;Not sure yet
            JA          TRY_HEX        ;Might be  HEX digit
            STC                        ;Set the carry
            RET
TRY_HEX:
            CMP         AL,'A'         ;Not sure yet
            JB          BAD_DIGIT      ;Not HEX
            CMP         AL,'F'         ;Not sure yet
            JA          BAD_DIGIT      ;Not HEX
            STC                        ;Set the carry
```

```
                RET
BAD_DIGIT:
                CLC                             ;Clear the carry
                RET
VALDG           ENDP

;*************************************************************************
;                       *****read keyboard procedure *****
;*************************************************************************

GETCH           PROC    NEAR
                MOV     AH,0            ;Code for read keyboard char.
                INT     16H             ;BIOS service
                CMP     AL,'a'          ;See if it is a lowercase letter
                JL      UPPER_CASE      ;Nope
                CMP     AL,'z'
                JG      UPPER_CASE
                SUB     AL,20H          ;Convert to upper letter
UPPER_CASE:
                RET
GETCH           ENDP

;*************************************************************************
;                   CNVBN = convert ASCII to  HEX    *
;*************************************************************************

CNVBN           PROC    NEAR
                CMP     AL,'0'          ;Is it a legal digit?
                JB      NOT_HEX1        ;Not decimal digit
```

```
                CMP     AL,'9'          ;Not sure yet
                JA      TRY_HEX1        ;May be HEX digit
                SUB     AL,'0'          ;Is decimal digit convert to nibble
                RET
TRY_HEX1:
                CMP     AL,'A'          ;Not sure yet
                JB      NOT_HEX1        ;Not hex digit
                CMP     AL,'F'          ;Not sure yet
                JA      NOT_HEX1        ;Not  HEX digit
                SUB     AL,'A'-10       ;Is hex,convert to nibble
NOT_HEX1:
                RET
CNVBN           ENDP

;*************************************************************************
;                       procedure to CR and LF     *
;*************************************************************************

CRLF            PROC    NEAR
                PUSH    CX
                PUSH    DX
                MOV     CX,02H
CRLF_1:
                MOV     DL,0DH          ;Carriage return
                MOV     AH,02H          ;Display function MSDOS
                INT     21H             ;Call  DOS
                MOV     DL,0AH          ;Line feed code
                MOV     AH,02H          ;Display function
                INT     21H             ;Call DOS
```

```
                DEC         CX
                JNZ         CRLF_1
                POP         DX
                POP         CX
                RET
CRLF            ENDP

;*************************************************************************
;       NMOUT = convert binary in AL to HEX on console screen *
;*************************************************************************

NMOUT           PROC        NEAR
                PUSH        CX
                PUSH        BX
                PUSH        DX
                MOV         CH,2        ;Number of digit
                MOV         BL,AL       ;Move to BL
ROTATE:
                MOV         CL,4        ;Set counter to 4 bits
                ROL         BL,CL       ;Left digit to right
                MOV         AL,BL       ;Move to AL
                AND         AL,0FH      ;Mask of left digit
                ADD         AL,30H      ;Convert HEX to ASCII
                CMP         AL,3AH      ;Is it >9?
                JL          PRINTIT     ;Jump if digit = 0 to 9
                ADD         AL,7H       ;Digit is A to F
PRINTIT:
                MOV         DL,AL       ;Put ASCII in DL
                MOV         AH,2        ;Display output function
```

```
                INT             21H             ;Call DOS
                DEC             CH              ;Done 2 digit
                JNZ             ROTATE          ;Not yet
                POP             DX
                POP             BX
                POP             CX
                RET
NMOUT           ENDP

;*************************************************************************
;                       communication line off  procedure       *
;*************************************************************************
COMM_OFF        PROC            NEAR
                IN              AL,21H
                OR              AL,08H          ;Set bit 3
                OUT             21H,AL
                JMP             SHORT $+2
                RET
COMM_OFF        ENDP

;*************************************************************************
;                       interrupt service routine
;*************************************************************************
RS232_INT:
                STI                     ;Interrupt on-except for communicatios
                PUSH            BX              ;Save registers to be used by
                PUSH            DX              ;The service routine
                PUSH            CX
                PUSHF
```

```
        PUSH    AX
        MOV     DX,DATA
        MOV     DS,DX
        ASSUME  DS:DATA
RS232_INT1:
        MOV     DX,STAT73
        IN      AL,DX           ;Read 8273 status
        JMP     SHORT $+2       ;I/O delay
        AND     AL,INT          ;Are TxINT or RxINT set?
        JZ      IEXIT           ;No interrupt exit
        MOV     DX,STAT73
        IN      AL,DX           ;Read 8273 status
        JMP     SHORT $+2       ;I/Odelay
        AND     AL,RXINT        ;Test RxINT
        JNZ     RXIC            ;Yes,go service it
        CALL    TXI             ;Must be  Tx,go service it
        MOV     AL,TXFLAG       ;Get Tx flag
        CMP     AL,01H          ;Was it a  completion?
        JNZ     IEXIT           ;No,so just exit
        JMP     RS232_INT1      ;Try again

RXIC:
        CALL    RXI             ;Go service Rx
        MOV     AL,RXFLAG       ;Get Rx flag
        CMP     AL,01H          ;Was it a completion?
        JNZ     IEXIT           ;No,so just exit
        JMP     RS232_INT       ;Try again

;signal end of interrupt to the interrupt command register
```

```
IEXIT:
                MOV     AL,20H          ;Code
                OUT     20H,AL          ;EOI port address
                JMP     SHORT $+2       ;I/O delay
                POP     AX              ;Restore register from stack
                POPF
                POP     CX
                POP     DX
                POP     BX
                IRET                    ;Return from interrupt
;*************************************************************************
;                       Receiver interrupt procedure    *
;*************************************************************************
RXI             PROC    NEAR
                MOV     DX,STAT73
                IN      AL,DX           ;(*)read 8273 status
                JMP     SHORT $+2       ;(*)I/O delay
                AND     AL,RXIRA        ;(*)test IRA bit
                JNZ     RXI7            ;(*)if 0 data transfer needed
                LEA     DI,RXBUF        ;(*)get data buffer address
                MOV     BX,RXPNT        ;(*)get data buffer pointer
                ADD     DI,BX           ;(*)address start + displacement
                MOV     DX,RXDACK       ;(*)read data via RXDACK
                IN      AL,DX
                JMP     SHORT $+2       ;(*)I/O delay
                MOV     BYTE PTR[DI],AL
                INC     BX              ;(*)bump data pointer
                CMP     BX,512          ;(*)pointer overflows buffer?
                JNE     RXI_1
```

```
                MOV     BX,0            ;(*)Reset to start of buffer

RXI_1:
                MOV     RXPNT,BX        ;(*)Update
                JMP     RXI5
RXI8:
                JMP     DOS_EXIT
RXI7:
                MOV     CH,04H          ;DH is result counter
                MOV     BX,LDADR        ;Get load pointer
                PUSH    BX              ;Save it
                PUSH    BX              ;Save it again
                MOV     DH,BL           ;Save LSB
                MOV     BX,CNADR        ;Get console pointer
RXI1:
                INC     DH              ;Bump load pointer LSB
                MOV     AL,DH           ;Get set to test
                CMP     AL,BL           ;Load = Console?
                JZ      RXI8            ;To monitor
                DEC     CH              ;DEC counter
                JNZ     RXI1            ;Not done,try again
                MOV     CX,05H          ;Set counter
                POP     BX              ;Restore load pointer
RXI2:
                MOV     DX,STAT73
                IN      AL,DX           ;Read status
                JMP     SHORT $+2       ;I/O delay
                AND     AL,RXINT        ;Test RxINT bit
                JZ      RXI3            ;Done,go finish up
```

```
        MOV     DX,STAT73
        IN      AL,DX           ;Read status again
        JMP     SHORT $+2       ;I/Ó delay
        AND     AL,RXIRA        ;Is result ready?
        JZ      RXI2            ;No,Test again
        MOV     DX,RXIR73
        IN      AL,DX           ;Yes,read result
        JMP     SHORT $+2       ;I/O delay
        LEA     DI,RESBUF       ;Result buffer address
        ADD     DI,BX           ;Buffer address + displacement
        MOV     BYTE PTR[DI],AL     ;Store in buffer
        INC     BX              ;INC buffer pointer
        DEC     CX              ;DEC counter
        JMP     RXI2            ;Get more result
RXI3:
        MOV     AX,CX           ;Get set to test
        AND     AX,AX           ;All results?
        JZ      RXI4            ;Yes,so finish up
        LEA     DI,RESBUF       ;Result buffer address
        ADD     DI,BX           ;Buffer address + displacement
        MOV     BYTE PTR[DI],00H    ;No,load 0 til done
        INC     BX              ;Bump pointer
        DEC     CX              ;DEC counter
        JMP     RXI3            ;Go again
RXI4:
        MOV     LDADR,BX        ;Update load pointer
        MOV     AL,01H          ;Test flag to show completion
        MOV     RXFLAG,AL       ;Store Rx FLAG
        MOV     AL,PRMPT        ;Get more indicator
```

```
CMP     AL,'-'          ;Normal mode?
JZ      RXI6            ;Yes,clean up before return

;Poll mode so check control,if control is a poll,set up special
;Tx command buffer and return with poll indicator not 0

POP     BX              ;Get previous load ADR pointer
LEA     DI,RESBUF       ;Result buffer address
ADD     DI,BX           ;Buffer address + displacement
MOV     AL,BYTE PTR[DI]     ;Get  IC byte from buffer
AND     AL,1EH          ;Look at good frame bits
JNZ     RXI5            ;If not 0 interrupt was n't from a
    ;good frame
INC     BL              ;Bypass R0 and R1 in buffer
INC     BL
INC     BL
LEA     DI,RESBUF       ;Get result buffer address again
PUSH    DI              ;Save it
ADD     DI,BX           ;Buffer address + displacement
MOV     DH,BYTE PTR[DI]   ;Get ADR byte and save it in DH
INC     BL
POP     DI              ;Restore DI
ADD     DI,BX           ;Buffer address + displacement
MOV     AL,BYTE PTR[DI]     ;Get CNTL byte from buffer
CMP     AL,SNRMP        ;Was it SNRM-P?
JZ      T1              ;Yes,go set respone
CMP     AL,RR0P         ;Was it RR(0)-P?
JNZ     RXI5    ;Yes,go set response,otherwise return
MOV     DL,RR0F         ;RR(0)-P so set response to RR(0)-F
```

```
                JMP         TXRET       ;Go finish loading special buffer
T1:
                MOV         DL,NSAF     ;SNRM-P so set response to NSA-F
TXRET:
                LEA         BX,CMDBF1   ;Special buffer ADR
                MOV         BYTE PTR[BX],0C8H  ;Load Tx frame command
                INC         BX          ;INC pointer
                MOV         BYTE PTR[BX],00H   ;L0 = 0
                INC         BX          ;INC pointer
                MOV         BYTE PTR[BX],DH    ;Load  RCVD ADR byte
                INC         BX          ;INC pointer
                MOV         BYTE PTR[BX],DL    ;Load response CNTL byte
                MOV         AL,01H      ;Set poll indicator not 0
                MOV         POLIN,AL    ;Load poll indicator
                JMP         RXI5        ;Return

RXI6:
                POP         BX          ;Clean up stack if normal mode
                JMP         RXI5        ;Return

RXI5:
                RET
RXI             ENDP

;*************************************************************************
;               TxINT-interrupt driven result/data handler procedure      *
;*************************************************************************
```

```
TXI     PROC    NEAR
        MOV     DX,STAT73
        IN      AL,DX           ;(*)read 8273 status
        JMP     SHORT $+2       ;(*)I/O delay
        AND     AL,TXIRA        ;(*)test TxIRA bit
        JZ      TXI3            ;(*)if 0,data transfer
        MOV     CH,04H          ;Set counter
        PUSH    CX              ;Store CX counter
        MOV     BX,LDADR        ;Get load pointer
        PUSH    BX              ;Save it
        MOV     DH,BL           ;Save LSB of LDADR in DH
        MOV     BX,CNADR        ;Get console pointer
TXI1:
        INC     DH              ;INC pointer load
        MOV     AL,DH           ;Get set to test
        CMP     AL,BL           ;Load = Console?
        JZ      TXI4            ;Yes,to  DOS monitor
        DEC     CH              ;No,test next location
        JNZ     TXI1            ;Try again
        POP     BX              ;Restore load pointer
        MOV     DX,TXIR73
        IN      AL,DX           ;Read result
        JMP     SHORT $+2       ;I/O delay
        LEA     DI,RESBUF       ;Result buffer address
        ADD     DI,BX           ;Buffer start + displacement
        MOV     BYTE PTR[DI],AL    ;Store in buffer
NOK:
        POP     CX
        INC     BX              ;INC pointer
```

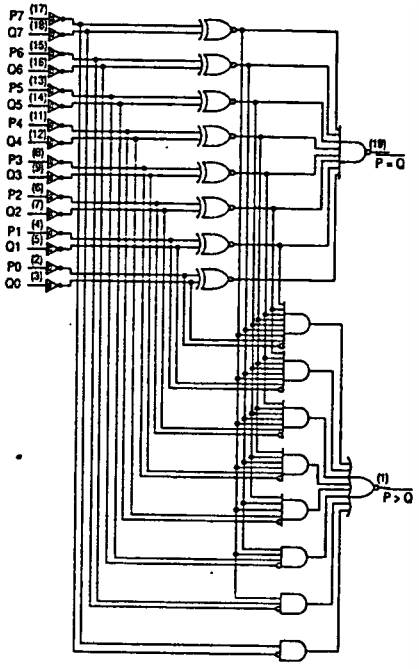
```
        LEA     DI,RESBUF
        ADD     DI,BX
        MOV     BYTE PTR[DI],00H   ;Extra result spots 0
        DEC     CH
        JNZ     NOK
        MOV     LDADR,BX           ;Update load pointer
        MOV     AL,01H             ;Set TxFLAG to show completion
        MOV     TXFLAG,AL          ;Set FLAG

TXI2:
        RET
TXI3:
        LEA     DI,TXBUF           ;(*)Tx data buffer address
        MOV     BX,TXPNT           ;(*)get Tx data pointer
        ADD     DI,BX              ;(*) buffer start + displacement
        MOV     AL,BYTE PTR[DI]    ;(*)get data from buffer
        MOV     DX,TXDACK
        OUT     DX,AL              ;(*)output to 8273 via TxDACK
        JMP     SHORT $+2          ;(*)I/O delay
        INC     BX                 ;(*)bump data pointer
        CMP     BX,512             ;(*)pointer overflows buffer?
        JNE     TXI5
        MOV     BX,0               ;(*)reset to start of buffer

TXI5:
        MOV     TXPNT,BX           ;(*)update
        JMP     TXI2               ;(*)return after store
TXI4:
        JMP     DOS_EXIT
```

```
TXI         ENDP
CODE        ENDS
END         START
```

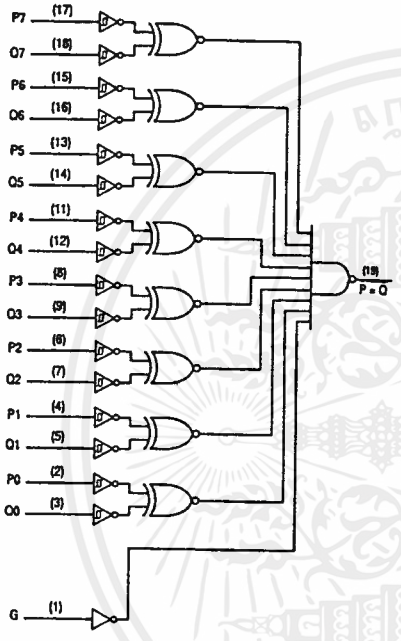
บรรณานุกรม

1. การสื่อสารข้อมูลดิจิตอล , รศ.ดร. วิวัฒน์ กิรานนท์ , คณะวิศวกรรมศาสตร์ สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหาร ลาดกระบัง , โรงพิมพ์ คุรุสภา ลาดพร้าว.
2. คู่มือเทียบเบอร์ไอซี TTL., บ. ซีเอ็ดยูเคชั่น จำกัด .
3. เอกสารประกอบการประชุมสัมมนาทางวิชาการ เพื่ออภิปรายแลกเปลี่ยนความคิดเห็นเกี่ยวกับเทคโนโลยีของเยอรมัน เรื่อง Digital Heirarchy .
4. FAST and LS TTL Data. , MOTOROLA .
5. PACKET SWITCHING ., Bo Lindgren .
6. Technical Reference . , IBM Revised Edition (April 1983) .
7. Using the 8273 SDLC / HDLC Protocol Controller. , John Beaston .

LOGIC DIAGRAMS



SN54/74LS682 thru LS684



SN54/74LS688

AC CHARACTERISTICS (T_A = 25°C)

SN54/74LS682

Symbol	Parameter	Limits Min	Typ	Max	Unit	Test Conditions
t_{PLH} t_{PHL}	Propagation Delay, P to $\overline{P = Q}$		13 15	25 25	ns	V_{CC} = 5.0 V C_L = 45 pF R_L = 667 Ω
t_{PLH} t_{PHL}	Propagation Delay, Q to $\overline{P = Q}$		14 15	25 25	ns	
t_{PLH} t_{PHL}	Propagation Delay, P to $\overline{P > Q}$		20 15	30 30	ns	
t_{PLH} t_{PHL}	Propagation Delay, Q to $\overline{P > Q}$		21 19	30 30	ns	

SN54/74LS684

Symbol	Parameter	Limits Min	Typ	Max	Unit	Test Conditions
t_{PLH} t_{PHL}	Propagation Delay, P to $\overline{P = Q}$		15 17	25 25	ns	V_{CC} = 5.0 V C_L = 45 pF R_L = 667 Ω
t_{PLH} t_{PHL}	Propagation Delay, Q to $\overline{P = Q}$		16 15	25 25	ns	
t_{PLH} t_{PHL}	Propagation Delay, P to $\overline{P > Q}$		22 17	30 30	ns	
t_{PLH} t_{PHL}	Propagation Delay, Q to $\overline{P > Q}$		24 20	30 30	ns	

SN54/74LS688

Symbol	Parameter	Limits Min	Typ	Max	Unit	Test Conditions
t_{PLH} t_{PHL}	Propagation Delay, P to $\overline{P = Q}$		12 17	18 23	ns	V_{CC} = 5.0 V C_L = 45 pF R_L = 667 Ω
t_{PLH} t_{PHL}	Propagation Delay, Q to $\overline{P = Q}$		12 17	18 23	ns	
t_{PLH} t_{PHL}	Propagation Delay, $\overline{G}$, $\overline{G1}$ to $\overline{P = Q}$		12 13	18 20	ns	

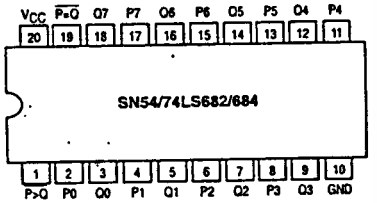
MOTOROLA

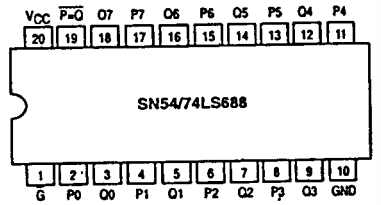
8-BIT MAGNITUDE COMPARATORS

The SN54/74LS682, 684, 688 are 8-bit magnitude comparators. These device types are designed to perform comparisons between two eight-bit binary or BCD words. All device types provide $\overline{P = Q}$ outputs and the LS682 and LS684 have $\overline{P>Q}$ outputs also.

The LS682, LS684 and LS688 are totem pole devices. The LS682 has a 20 kΩ pullup resistor on the Q inputs for analog or switch data.

CONNECTION DIAGRAMS (TOP VIEW)



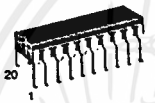


TYPE	$\overline{P = Q}$	$\overline{P > Q}$	OUTPUT ENABLE	OUTPUT CONFIGURATION	PULLUP
LS682	yes	yes	no	totem-pole	yes
LS684	yes	yes	no	totem-pole	no
LS688	yes	no	yes	totem-pole	no

SN54/74LS682
SN54/74LS684
SN54/74LS688

8-BIT MAGNITUDE COMPARATORS

LOW POWER SCHOTTKY



J SUFFIX
CERAMIC
CASE 732-03

N SUFFIX
PLASTIC
CASE 738-03

DW SUFFIX
SOIC
CASE 751D-03

ORDERING INFORMATION

SN54LSXXXJ Ceramic
SN74LSXXXN Plastic
SN74LSXXXDW SOIC

FUNCTION TABLE

INPUTS			OUTPUTS	
DATA	ENABLES			
P, Q	$\overline{G}$, $\overline{GT}$	$\overline{G2}$	$\overline{P = Q}$	$\overline{P > Q}$
P = Q	L	L	L	H
P > Q	L	L	H	L
P < Q	L	L	H	H
X	H	H	H	H

H = HIGH Level, L = LOW Level, X = Irrelevant

GUARANTEED OPERATING RANGES

Symbol	Parameter		Min	Typ	Max	Unit
V_{CC}	Supply Voltage	54 74	4.5 4.75	5.0 5.0	5.5 5.25	V
T_A	Operating Ambient Temperature Range	54 74	−55 0	25 25	125 70	°C
I_{OH}	Output Current — High	54, 74			−0.4	mA
I_{OL}	Output Current — Low	54 74			12 24	mA

DC CHARACTERISTICS OVER OPERATING TEMPERATURE RANGE (unless otherwise specified)

Symbol	Parameter		Limits Min	Typ	Max	Unit	Test Conditions
V_{IH}	Input HIGH Voltage		2.0			V	Guaranteed Input HIGH Voltage for All Inputs
V_{IL}	Input LOW Voltage	54			0.7	V	Guaranteed Input LOW Voltage for All Inputs
		74			0.8		
V_{IK}	Input Clamp Diode Voltage			−0.65	−1.5	V	V_{CC} = MIN, I_{IN} = −18 mA
V_{OH}	Output HIGH Voltage	54	2.5	3.5		V	V_{CC} = MIN, I_{OH} = MAX, V_{IN} = V_{IH} or V_{IL} per Truth Table
		74	2.7	3.5		V	
V_{OL}	Output LOW Voltage	54, 74		0.25	0.4	V	I_{OL} = 12 mA; V_{CC} = V_{CC} MIN, V_{IN} = V_{IL} or V_{IH} per Truth Table
		74		0.35	0.5	V	I_{OL} = 24 mA
I_{IH}	Input HIGH Current				20	μA	V_{CC} = MAX, V_{IN} = 2.7 V
		LS628-Q Inputs			0.1	mA	V_{CC} = MAX, V_{IN} = 5.5 V
		Others			0.1	mA	V_{CC} = MAX, V_{IN} = 7.0 V
I_{IL}	Input LOW Current	LS682-Q Inputs			−0.4	mA	V_{CC} = MAX, V_{IN} = 0.4 V
		Others			−0.2	mA	
I_{OS}	Short Circuit Current (Note 1)		−30		−130	mA	V_{CC} = MAX
I_{CC}	Power Supply Current	LS682			70	mA	V_{CC} = MAX
		LS684			65	mA	
		LS688			65	mA	

Note 1: Not more than one output should be shorted at a time, nor for more than 1 second.

GUARANTEED OPERATING RANGES

Symbol	Parameter		Min	Typ	Max	Unit
V_{CC}	Supply Voltage	54 74	4.5 4.75	5.0 5.0	5.5 5.25	V
T_A	Operating Ambient Temperature Range	54 74	−55 0	25 25	125 70	°C
I_{OH}	Output Current — High	54, 74			−0.4	mA
I_{OL}	Output Current — Low	54 74			4.0 8.0	mA

DC CHARACTERISTICS OVER OPERATING TEMPERATURE RANGE (unless otherwise specified)

Symbol	Parameter		Limits Min	Limits Typ	Limits Max	Unit	Test Conditions	
V_{IH}	Input HIGH Voltage		2.0			V	Guaranteed Input HIGH Voltage for All Inputs	
V_{IL}	Input LOW Voltage	54			0.7	V	Guaranteed Input LOW Voltage for All Inputs	
		74			0.8			
V_{IK}	Input Clamp Diode Voltage			−0.65	−1.5	V	V_{CC} = MIN, I_{IN} = −18 mA	
V_{OH}	Output HIGH Voltage	54	2.5	3.5		V	V_{CC} = MIN, I_{OH} = MAX, V_{IN} = V_{IH} or V_{IL} per Truth Table	
		74	2.7	3.5		V		
V_{OL}	Output LOW Voltage	54, 74		0.25	0.4	V	I_{OL} = 4.0 mA	V_{CC} = V_{CC} MIN, V_{IN} = V_{IL} or V_{IH} per Truth Table
		74		0.35	0.5	V	I_{OL} = 8.0 mA	
I_{IH}	Input HIGH Current				20	μA	V_{CC} = MAX, V_{IN} = 2.7 V	
					0.1	mA	V_{CC} = MAX, V_{IN} = 7.0 V	
I_{IL}	Input LOW Current				−0.4	mA	V_{CC} = MAX, V_{IN} = 0.4 V	
I_{OS}	Short Circuit Current (Note 1)		−20		−100	mA	V_{CC} = MAX	
I_{CC}	Power Supply Current				10	mA	V_{CC} = MAX	

Note 1: Not more than one output should be shorted at a time, nor for more than 1 second.

AC CHARACTERISTICS (T_A = 25°C)

Symbol	Parameter	Levels of Delay	Limits Min	Limits Typ	Limits Max	Unit	Test Conditions
t_{PLH} t_{PHL}	Propagation Delay Address to Output	2 2		13 27	20 41	ns	V_{CC} = 5.0 V C_L = 15 pF
t_{PLH} t_{PHL}	Propagation Delay Address to Output	3 3		18 26	27 39	ns	
t_{PLH} t_{PHL}	Propagation Delay $\overline{E}_1$ or $\overline{E}_2$ Enable to Output	2 2		12 21	18 32	ns	
t_{PLH} t_{PHL}	Propagation Delay E_3 Enable to Output	3 3		17 25	26 38	ns	

AC WAVEFORMS

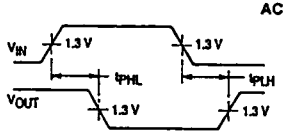


Figure 1

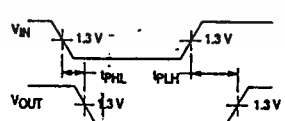


Figure 2

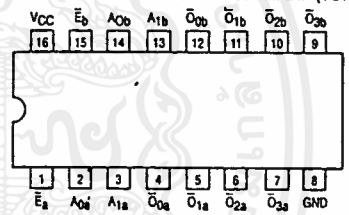
MOTOROLA

DUAL 1-OF-4 DECODER/ DEMULTIPLEXER

The LSTTL/MSI SN54/74LS139 is a high speed Dual 1-of-4 Decoder/Demultiplexer. The device has two independent decoders, each accepting two inputs and providing four mutually exclusive active LOW Outputs. Each decoder has an active LOW Enable input which can be used as a data input for a 4-output demultiplexer. Each half of the LS139 can be used as a function generator providing all four minterms of two variables. The LS139 is fabricated with the Schottky barrier diode process for high speed and is completely compatible with all Motorola TTL families.

- Schottky Process for High Speed
- Multifunction Capability
- Two Completely Independent 1-of-4 Decoders
- Active Low Mutually Exclusive Outputs
- Input Clamp Diodes Limit High Speed Termination Effects
- ESD > 3500 Volts

CONNECTION DIAGRAM DIP (TOP VIEW)



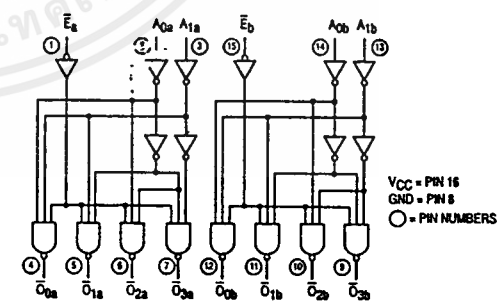
NOTE:
The Flatpak version has the same pinouts (Connection Diagram) as the Dual In-Line Package.

PIN NAMES

		LOADING (Note a) HIGH	LOW
A_0, A_1	Address Inputs	0.5 U.L.	0.25 U.L.
$\overline{E}$	Enable (Active LOW) Input	0.5 U.L.	0.25 U.L.
$\overline{O}_0$–$\overline{O}_3$	Active LOW Outputs (Note b)	10 U.L.	5 (2.5) U.L.

NOTES:
a) 1 TTL Unit Load (U.L.) = 40 μA HIGH/1.6 mA LOW.
b) The Output LOW drive factor is 2.5 U.L. for Military (54) and 5 U.L. for Commercial (74) Temperature Ranges.

LOGIC DIAGRAM



SN54/74LS139

DUAL 1-OF-4 DECODER/ DEMULTIPLEXER

LOW POWER SCHOTTKY

J SUFFIX
CERAMIC
CASE 620-09

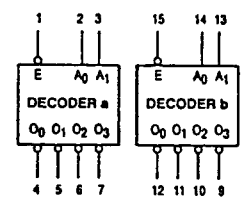
N SUFFIX
PLASTIC
CASE 648-08

D SUFFIX
SOIC
CASE 751B-03

ORDERING INFORMATION

SN54LSXXXJ	Ceramic
SN74LSXXXN	Plastic
SN74LSXXXD	SOIC

LOGIC SYMBOL



VCC = PIN 16
GND = PIN 8

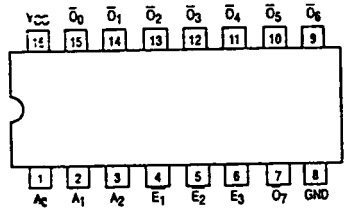
MOTOROLA

1-OF-8 DECODER/ DEMULTIPLEXER

The LSTTL/MSI SN54/74LS138 is a high speed 1-of-8 Decoder/ Demultiplexer. This device is ideally suited for high speed bipolar memory chip select address decoding. The multiple input enables allow parallel expansion to a 1-of-24 decoder using just three LS138 devices or to a 1-of-32 decoder using four LS138s and one inverter. The LS138 is fabricated with the Schottky barrier diode process for high speed and is completely compatible with all Motorola TTL families.

- Demultiplexing Capability
- Multiple Input Enable for Easy Expansion
- Typical Power Dissipation of 32 mW
- Active Low Mutually Exclusive Outputs
- Input Clamp Diodes Limit High Speed Termination Effects

CONNECTION DIAGRAM DIP (TOP VIEW)



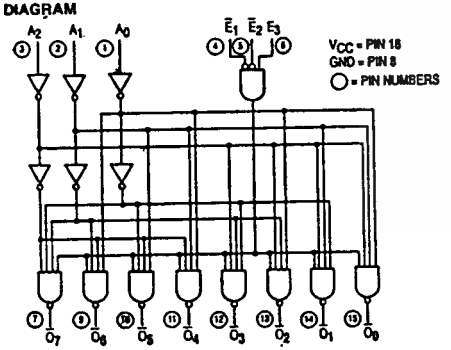
NOTE:
The Flatpak version has the same pinouts (Connection Diagram) as the Dual In-Line Package.

PIN NAMES

		LOADING (Note a) HIGH	LOW
A_0–A_2	Address Inputs	0.5 U.L.	0.25 U.L.
$\overline{E}_1$, $\overline{E}_2$	Enable (Active LOW) Inputs	0.5 U.L.	0.25 U.L.
E_3	Enable (Active HIGH) Input	0.5 U.L.	0.25 U.L.
$\overline{O}_0$–$\overline{O}_7$	Active LOW Outputs (Note b)	10 U.L.	5 (2.5) U.L.

NOTES:
a) 1 TTL Unit Load (U.L.) = 40 µA HIGH/1.6 mA LOW.
b) The Output LOW drive factor is 2.5 U.L. for Military (54) and 5 U.L. for Commercial (74) Temperature Ranges.

LOGIC DIAGRAM



SN54/74LS138

1-OF-8 DECODER/ DEMULTIPLEXER

LOW POWER SCHOTTKY

J SUFFIX
CERAMIC
CASE 620-09

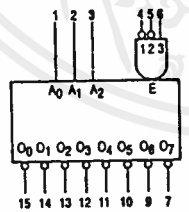
N SUFFIX
PLASTIC
CASE 648-08

D SUFFIX
SOIC
CASE 751B-03

ORDERING INFORMATION

SN54LSXXXJ Ceramic
SN74LSXXXN Plastic
SN74LSXXXD SOIC

LOGIC SYMBOL



FUNCTIONAL DESCRIPTION

The LS138 is a high speed 1-of-8 Decoder/Demultiplexer fabricated with the low power Schottky barrier diode process. The decoder accepts three binary weighted inputs (A_0, A_1, A_2) and when enabled provides eight mutually exclusive active LOW Outputs ($\overline{O}_0$–$\overline{O}_7$). The LS138 features three Enable inputs, two active LOW ($\overline{E}_1$, $\overline{E}_2$) and one active HIGH (E_3). All outputs will be HIGH unless $\overline{E}_1$ and $\overline{E}_2$ are LOW and E_3 is HIGH. This multiple enable function allows easy parallel expansion of the device to a 1-of-32 (5 lines to 32 lines) decoder with just four LS138s and one inverter. (See Figure a.)

The LS138 can be used as an 8-output demultiplexer by using one of the active LOW Enable inputs as the data input and the other Enable inputs as strobes. The Enable inputs which are not used must be permanently tied to their appropriate active HIGH or active LOW state.

TRUTH TABLE

INPUTS						OUTPUTS							
$\overline{E}_1$	$\overline{E}_2$	E_3	A_0	A_1	A_2	$\overline{O}_0$	$\overline{O}_1$	$\overline{O}_2$	$\overline{O}_3$	$\overline{O}_4$	$\overline{O}_5$	$\overline{O}_6$	$\overline{O}_7$
H	X	X	X	X	X	H	H	H	H	H	H	H	H
X	H	X	X	X	X	H	H	H	H	H	H	H	H
X	X	L	X	X	X	H	H	H	H	H	H	H	H
L	L	H	L	L	L	L	H	H	H	H	H	H	H
L	L	H	H	L	L	H	L	H	H	H	H	H	H
L	L	H	L	H	L	H	H	L	H	H	H	H	H
L	L	H	H	H	L	H	H	H	L	H	H	H	H
L	L	H	L	L	H	H	H	H	H	L	H	H	H
L	L	H	H	L	H	H	H	H	H	H	L	H	H
L	L	H	L	H	H	H	H	H	H	H	H	L	H
L	L	H	H	H	H	H	H	H	H	H	H	H	L

H = HIGH Voltage Level
L = LOW Voltage Level
X = Don't Care

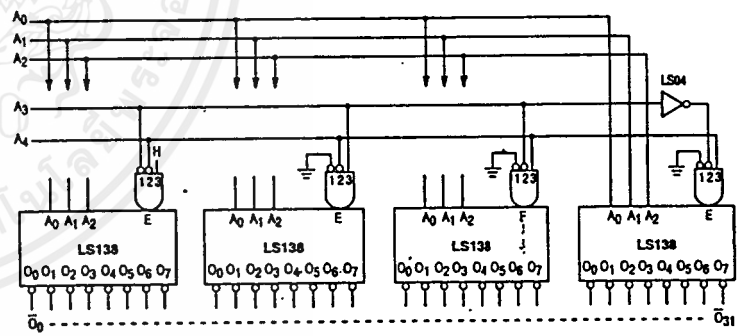


Figure a

DC CHARACTERISTICS OVER OPERATING TEMPERATURE RANGE (unless otherwise specified)

Symbol	Parameter		Limits Min	Typ	Max	Unit	Test Conditions	
V_{IH}	Input HIGH Voltage		2.0			V	Guaranteed Input HIGH Voltage for All Inputs	
V_{IL}	Input LOW Voltage	54			0.7	V	Guaranteed Input LOW Voltage for All Inputs	
		74			0.8			
V_{IK}	Input Clamp Diode Voltage			−0.65	−1.5	V	V_{CC} = MIN, I_{IN} = −18 mA	
V_{OH}	Output HIGH Voltage	54	2.5	3.5		V	V_{CC} = MIN, I_{OH} = MAX, V_{IN} = V_{IH} or V_{IL} per Truth Table	
		74	2.7	3.5		V		
V_{OL}	Output LOW Voltage	54, 74		0.25	0.4	V	I_{OL} = 4.0 mA	V_{CC} = V_{CC} MIN, V_{IN} = V_{IL} or V_{IH} per Truth Table
		74		0.35	0.5	V	I_{OL} = 8.0 mA	
I_{IH}	Input HIGH Current				20	μA	V_{CC} = MAX, V_{IN} = 2.7 V	
					0.1	mA	V_{CC} = MAX, V_{IN} = 7.0 V	
I_{IL}	Input LOW Current				−0.4	mA	V_{CC} = MAX, V_{IN} = 0.4 V	
I_{OS}	Short Circuit Current (Note 1)		−20		−100	mA	V_{CC} = MAX	
I_{CC}	Power Supply Current Total, Output HIGH				6.2	mA	V_{CC} = MAX	
	Total, Output LOW				9.8			

Note 1: Not more than one output should be shorted at a time, nor for more than 1 second.

AC CHARACTERISTICS (T_A = 25°C)

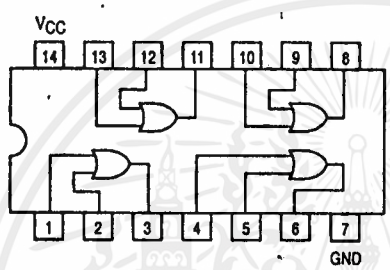
Symbol	Parameter	Limits Min	Typ	Max	Unit	Test Conditions
t_{PLH}	Turn-Off Delay, Input to Output		14	22	ns	V_{CC} = 5.0 V, C_L = 15 pF
t_{PHL}	Turn-On Delay, Input to Output		14	22	ns	

MOTOROLA

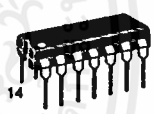
QUAD 2-INPUT OR GATE

SN54/74LS32

QUAD 2-INPUT OR GATE
LOW POWER SCHOTTKY



J SUFFIX
CERAMIC
CASE 632-08



N SUFFIX
PLASTIC
CASE 646-06

D SUFFIX
SOIC
CASE 751A-02

ORDERING INFORMATION

SN54LSXXJ	Ceramic
SN74LSXXN	Plastic
SN74LSXXD	SOIC

GUARANTEED OPERATING RANGES

Symbol	Parameter		Min	Typ	Max	Unit
V_{CC}	Supply Voltage	54 74	4.5 4.75	5.0 5.0	5.5 5.25	V
T_A	Operating Ambient Temperature Range	54 74	−55 0	25 25	125 70	°C
I_{OH}	Output Current — High	54, 74			−0.4	mA
I_{OL}	Output Current — Low	54 74			4.0 8.0	mA