เครื่องตรวจสอบโปรโตคอล X.28

X.28 PROTOCOL ANALYZER

โดย

นายกิจวัตร ปานดำรง

นายนันทวิทย์ แก้วถาวร

นายอภิชาติ สวรรค์คำธรณ์

ปริญญานิพนธ์ฉบับนี้

เป็นส่วนหนึ่งของการศึกษาตามหลักสูตร

ปริญญาอุตสาหกรรมศาสตร์บัณฑิต

สาขาเทคโนโลยีคอมพิวเตอร์อุตสาหกรรม

สถาบันเทคโนโลยีพระจอมเกล้า

เจ้าคุณทหารลาดกระบัง

ปีการศึกษา 2535

ภาควิชา เทคโนโลยีการวัดคุมทางอุตสาหกรรม

สาขา เทคโนโลยีคอมพิวเตอร์อุตสาหกรรม

คณะวิศวกรรมศาสตร์ สถาบันเทคโนโลยีพระจอมเกล้า เจ้าคุณทหาร ลาดกระบัง

เรื่อง เครื่องตรวจสอบโปรโตคอล X.28

X.28 PROTOCOL ANALYZER

ผู้จัดทำ

1.	นายกิจวัตร	ปานดำรง	34162201
2.	นายนันทวิทย์	แก้วถาวร	34162208
3.	นายอภิชาติ	สวรรค์คำธรณ์	34162244

.....อาจารย์ที่ปรึกษา

(อาจารย์ พิพัฒน์ เลาหสงคราม)

หัวข้อปริญญานิพนธ์	X.28 PROTOCOL ANALYZER		
นักศึกษา	นายกิจวัตร	ปานดำรง	34162201
	นายนันทวิทย์	แก้วถาวร	34162208
	นายอภิชาติ	สวรรค์คำธรณ์	34162244
อาจารย์ที่ปรึกษา	อ.พิพัฒน์	เลาหสงคราม	
ระดับการศึกษา	อุตสาหกรรมศาสตร์บัณฑิตทางเทคโนโลยีการวัดคุมอุตสาหกรรม		
ปีการศึกษา	พ.ศ. 2535		

บทคัดย่อ

ปริญญานิพนธ์ฉบับนี้เป็นการกล่าวถึงการนำเอา PERSONAL COMPUTER มาประยุกต์ใช้เป็นอุปกรณ์ช่วยตรวจสอบข้อมูลสำหรับโปรโตคอล X.28 ซึ่งมีประสิทธิภาพสูงเทียบได้กับอุปกรณ์มาตรฐานซึ่งมีราคาสูงมาก PROJECT นี้มีการดัดแปลงเพียงเล็กน้อยจากอุปกรณ์มาตรฐานบน IBM PC อีกทั้ง PC ดังกล่าวยังใช้งานอื่น ๆ ได้ตามปกติในขณะที่มิได้ใช้เป็น PROTOCOL ANALYZER

THESIS TITLE	X.28 Protocol Analyzer		
NAME	KITJAWAT	PARNDOMRONG	34162201
	NUNTAWIT	KAEWTHAVORN	34162208
	APICHAT	SWANKOMTORN	34162244
THESIS ADVISOR	PIPAT	LALHASONGKRAM	Assist.Prof.
LEVEL OF STUDY	Bachelor's Degree of Computer Technology		
	Instrument Department		
ACADEMIC YEAR	1992		

ABSTRACT

This thesis is to introduce how to adapt personal computer for protocol analyzer to trace and analyze X.28 protocol. In high efficiency but low cost compare with standard protocol analyzer. That available in the market, for this project. The personal computer still maintain all functions of personal computer but by modifying some hard ware and software, this personal computer (IBM PC) will work as protocol analyzer as well.

กิตติกรรมประกาศ

ปริญญานิพนธ์ฉบับนี้สำเร็จ ลุล่วงไปได้ด้วยดี เพราะได้รับความช่วยเหลือชี้แนะข้อคิดเห็นต่าง ๆ ได้เรียนคอร์สหนึ่งจากอาจารย์ที่ปรึกษา จึงกราบขอบคุณ อาจารย์ ผศ.พิพัฒน์ เลาหสงคราม

ประโยชน์อันเกิดจากปริญญานิพนธ์ฉบับนี้ขอมอบความดี ให้ผู้มีพระคุณของอาจารย์ ผศ.พิพัฒน์ เลาหสงคราม ผู้ที่ให้ คำปรึกษาาคำปรึกษา มาโดยตลอด

สารบัญ

หน้าที่

บทนำ

ในปัจจุบันจะเห็นได้ว่าการสื่อสารข้อมูล มีความเจริญก้าวหน้าและมีบทบาทอย่างมากในวงการ ธุรกิจต่างๆ ซึ่งทราบโดยทั่วไป เช่น ระบบออนไลน์ (on line system) หรือระบบสื่อสารข้อมูลต่างๆ การรับส่งข้อมูลกระทำโดยใช้คู่สายโทรศัพท์ หรือช่องสัญญานในย่านไมโครเวฟเมื่อระบบสื่อสารมีความก้าวหน้ามากขึ้น อุปกรณ์ที่ใช้ในระบบรวมทั้งตัวระบบเองก็มีความสลับซับซ้อนมากขึ้นสิ่งที่ตามมาคือความยุ่งยากในการติดต่อสื่อสาร ถ้าไม่มีการจัดระบบที่ดีพอ ดังนั้นจึงได้มีการกำหนดมาตรฐานของ ระบบสื่อสารข้อมูลขึ้นโดยกำหนดให้การรับส่งข้อมูลมีรูปแบบที่ แน่นอนชนิด หนึ่งเรียกว่า protocal ขึ้นมาใช้ในระบบ สื่อสารข้อมูล มารตฐาน ในการสื่อสารข้อมูล นี้กำหนดขึ้นหลายมาตรฐานแต่สำหรับโครงงานนี้จะกล่าวถึง x.28 Protocal อย่างไรก็ตามการใช้ Protocalเป็น Formatในการรับส่งข้อมูลก็ย่อมมีข้อผิดพลาดเกิดขึ้นได้ซึ่งอาจจะเนื่องมาจาก สัญญานรบกวน (noise)ทำให้ผู้รับรับข้อมูลที่ผิดพลาดไป ดังเมื่อเกิดความผิดพลาด ขึ้นเราจะทราบได้อย่างไรว่าความผิดพลาดนั้นเกิดขึ้นที่จุดใด มีสาเหตุมาจากอะไร ถ้าเราใช้ เครื่อง มืออย่างเช่น Oscilloscope หรือ Logic Analyzer มาตรวจ สอบ โปรโตรคอล ก็ ไม่สามารถ อ่านออกได้ปัญหาอย่างแรก ก็คือข้อมูลที่ส่งมาในโปรโตรคอลมีลักษณะ ไม่เป็นจำนวน คาบและทั้งนี้ทั้งนั้น การแสดง ผลของ Oscilloscope นั้น เป็น แบบ (Real time) ดังนั้นจึงต้องใช้เครื่องมือเฉพาะ ในการตรวจสอบโปรโตรคอล ซึ่งสามารถบอก ได้ว่าใน 1 เฟรมของโปรโตรคอล ประกอบด้วยอะไรบ้างช่วงใดเป็นการเริ่มต้นหรือสิ้นสุดของข้อมูล ความผิดพลาดที่เกิดขึ้นและแสดงให้เห็นว่าเฟรมใดที่รับมา เมื่อทราบถึง ข้อมูลเหล่านี้ก็สามารถวิเคราะห์ถึงความผิดพลาด ที่เกิดขึ้น ในระบบอย่าง มีประสิทธิภาพ และในภาวะ ปกติเราก็สามารถตรวจสอบขั้นตอนในการรับส่งโปรโตรคอลได้

จากที่กล่าวมาแล้วนั้นจะเห็นได้ว่าเครื่องมือ ตรวจสอบ Protocal เป็นเครื่องมือพื้นฐานที่สำคัญมากในระบบสื่อสารข้อมูลเพราะทำให้ เราทราบถึงความเป็นไป ในระบบในกรณีที่มีข้อผิดพลาด เกิดขึ้นก็สามารถวิเคราะห์ถึงปัญหา และการแก้ไขได้อย่าง รวดเร็วและมีประสิทธิภาพ ทำให้ลดการสูญเสียทางเวลา และการติดต่อสื่อสารได้

ปริญญานิพนธ์นี้ จะกล่าวถึงการออกแบบ และ สร้างเครื่องมือ ตรวจสอบโปรโตรคอลชนิด Asynchronous ซึ่งเป็นโปรโตรคอล ที่มีการ ใช้กว้างขวางที่สุดในประเทศไทยในขณะนี้โดยมีวัตถุประสงค์ และขอบเขตของการวิจัยและประโยชน์ที่คาดว่าจะได้รับดังนี้

วัตถุประสงค์

เพื่อออกแบบ สร้างเครื่อง มือตรวจสอบ โพรโตคอลชนิด Asynchronous ที่สามารถใช้ในการตรวจสอบดังนี้

1. ใช้ตรวจสอบการทำงานระหว่างอุปกรณ์ ข้อมูลปลายทางและอุปกรณ์สื่อสารข้อมูล
2. ใช้ตรวจสอบคำสั่งและ ผลตอบในการรับส่งข้อมูล
3. แสดงความผิดพลาดที่เกิดขึ้นในการรับส่งข้อมูล
4. สามารถใช้เป็น Terminal Emulator

ขอบเขตของปริญญานิพนธ์

เครื่องมือตรวจสอบโพรโตคอลนี้ประกอบด้วย

1. ส่วนอ่านและวิเคราะห์ ข้อมูล เป็นส่วนที่รับข้อมูลมาและทำการแยกชนิดของข้อมูลว่าเป็นแบบใดแล้วจึงนำไปเก็บในหน่วยความจำ ความจุของหน่วยความจำที่ใช้ในการเก็บข้อมูลมีขนาด 64 กิโลไบต์ ส่วนอ่านและวิเคราะห์ข้อมูลนี้จะแบ่งเป็น 2 ชุดโดยชุดแรกใช้อ่านข้อมูลที่มาจากอุปกรณ์ข้อมูลปลายทาง (DTE)ส่วนที่สองใช้อ่านข้อมูลที่มาจากอุปกรณ์สื่อสารข้อมูล (DCE)

2. ส่วนแสดงผลและบันทึกข้อมูลส่วนนี้จะใช้ ไมโครคอมพิวเตอร์ IPM PC มาเก็บในหน่วยความ จำคอมพิวเตอร์และหลังจากนั้นก็นำผล ไปแสดงบนจอภาพ ส่วนการบันทึกข้อมูลจะบันทึกลงจานแม่เหล็ก

ความสำคัญและประโยชน์ที่คาดว่าจะได้รับจากปริญญานิพนธ์

1. จะเป็นการ พัฒนาเทคนิคการออกแบบและสร้างเครื่อง มือตรวจสอบโปรโตรคอล

2. จะกระตุ้นให้เกิดอุตสาหกรรมทางด้านการสร้าง เครื่องมือตรวจสอบโปรโตรคอล

3. เครื่องมือที่สร้างขึ้นนี้สามารถใช้ประโยชน์ทางด้านการสื่อสารข้อมูล

PROTOCAL X.28

X.28 เป็น PROTOCAL ที่ใช้ในการสื่อสารข้อมูลซึ่งเป็นข้อตกลงระหว่างประเทศ ซึ่งประชุมกันตกลงรายละเอียดของ PROTOCAL แต่ละชนิดทุก ๆ ปี หน่วยงานนี้มี ชื่อว่า (THE INTERNATIONAL TELEGRAPH AND TELEPHONE CONSULTATIVE COMMITTEE)

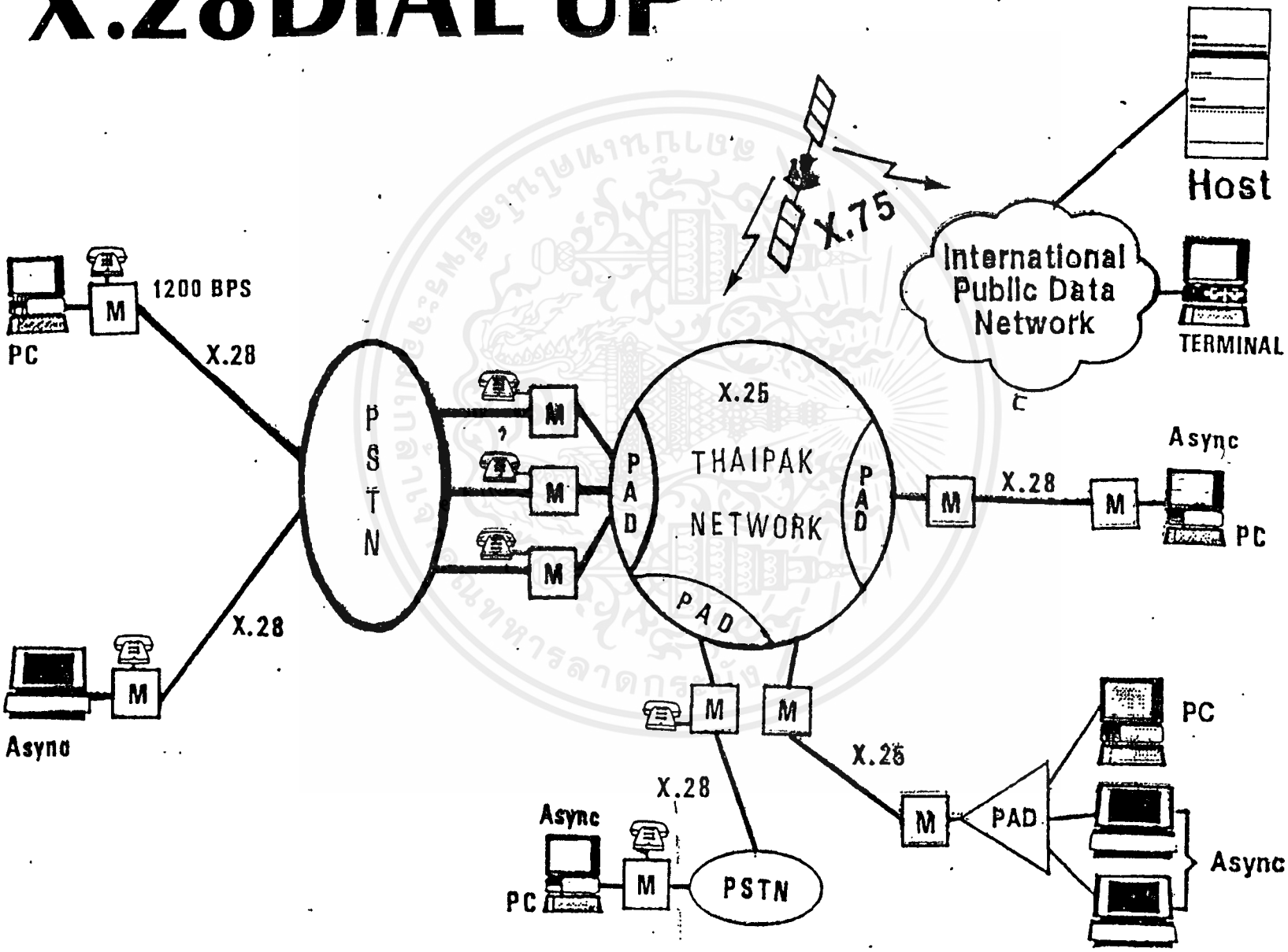
ลักษณะโครงสร้างพื้นฐานของ X.28 คือเป็นลักษณะของการสื่อสารข้อมูลอนุกรม ที่ใช้ความเร็วไม่สูงมากนักโดยมีความเร็วระหว่าง 300 - 9600 bps ในลักษณะ การส่งข้อ มูลแบบ อะซิงโครนัส (ASYNCHRONOUS MODE) ซึ่งมีการใช้ START-STOP BITS การ ตรวจสอบ PARITY ในส่วนหนึ่งซึ่งเป็นส่วนสำคัญที่จะกล่าวถึงคือการใช้งาน PROTOCAL X.28 โดยสื่อสารข้อมูลผ่านทาง PACKET SWITCHING โดยทั่วไประบบชุมสายสื่อ สารข้อมูล (PSDN : PUBBLIC SWITCHING DATA NETWORK) เป็นระบบ ที่ใช้การสื่อสารข้อมูลระหว่างกันเป็นแบบ SYNCHRONOUS MODE ที่ PROTOCAL X.25 และ X.75 เป็นหลักใหญ่และมีการให้บริการ X.28 เป็นรายย่อยของลูกค้า

ดังนั้นในการที่จะให้ X.28 สามารถติดต่อกับ PROTOCAL อื่น ๆ ในระบบชุมสาย ซึ่ง เป็นลักษณะ STORE & FORWARD โดยใช้ X.25 PROTOCAL ซึ่งเป็น PACKET MODE จึงจำเป็นต้องมีอุปกรณ์ที่ใช้ในการสับเปลี่ยนลักษณะของข้อมูลให้ตรงกับที่ฝ่ายหนึ่ง ซึ่งเรียกว่า PAD (PACKET ASSEMBLY/DISASSEMBLY) PAD ดังกล่าว จะทำหน้าที่เปลี่ยนข้อมูลที่ส่งมาจาก X.28 ให้เป็น PACKET MODE เพื่อส่งผ่าน X.25 PROTOCALไปยังจุดหมายปลายทางและทำการ เปลี่ยนข้อมูลจากปลายทางที่รับมาในรูปของ PACKET ให้ เป็นลักษณะข้อมูลของ X.28 MODE เพื่อส่งไปยังต้นทาง PAD ดังกล่าวจึงมี PARAMETER ต่างๆ ที่ USER สามารถ กำหนดได้เพื่อให้ได้คุณลักษณะตามต้องการที่จะใช้กับการส่งข้อมูล แต่ละชนิด

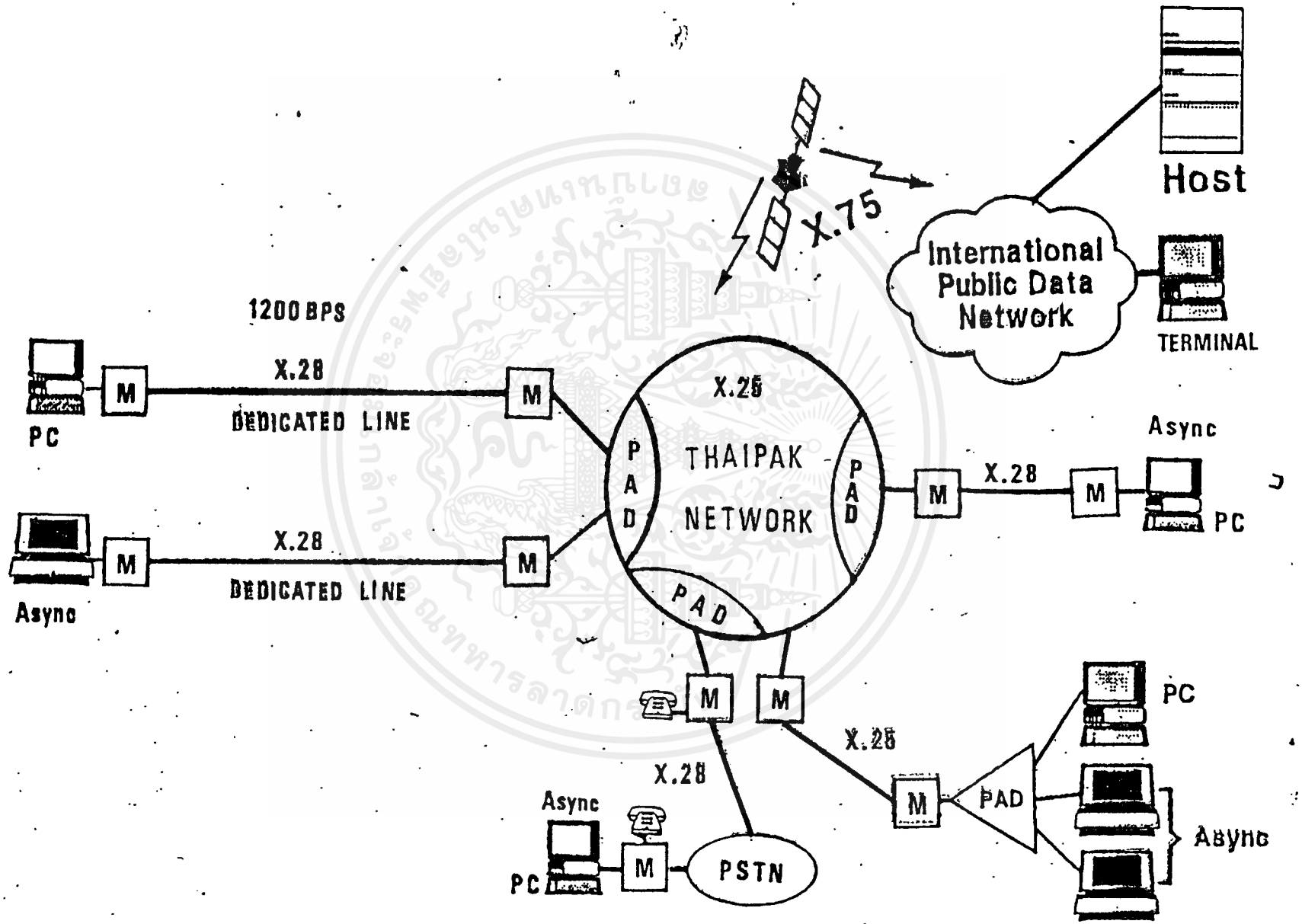
ลักษณะการใช้งาน X.28 PROTOCAL สามารถ แบ่งออกได้เป็น 2 ชนิดคือ

1. DEDICATED CIRCUIT
2. DIAL-UP CIRCUIT

X.28DIAL UP


Host
X.75
International
Public Data
Network
TERMINAL
1200 BPS
PC
X.28
M
PSTN
X.25
THAIPAK
NETWORK
PAD
Async
X.28
PC
X.28
Async
X.28
PC
X.25
PAD
PC
Async
Async
PC
PSTN

X.28 DEDICATED LINE



DEDICATED CIRCUIT เป็นลักษณะของการสื่อสารข้อมูลที่มีวงจรสายตรง (LEASED LINE)จากชุมสายสื่อสารข้อมูลไปยังลูกข่าย (USERS) ซึ่งใช้ในลักษณะ การรับส่งข้อมูลที่มีปริมาณข้อมูล มาก และในระยะทาง ที่ไม่ไกล จนเกินไปนักเฉลี่ยระยะทางระหว่าง 0-3 กิโลเมตรซึ่งก็จะขึ้นอยู่กับข้อจำกัดของวงจรที่ใช้ด้วย เช่น หากเป็น สายเคเบิล ปกติไม่ควรเกิน 3 กิโลเมตรหาก ใช้ลักษณะของวงจรที่ดีกว่า เช่น ใช้เป็น วงจร PCM ซึ่งอาจจะทำให้ได้ระยะทางมากกว่าเดิมในลักษณะของ X.28 DEDICATED CIRCUIT เป็นลักษณะ การสื่อสารข้อมูลแบบ 2 ทางดังกล่าวคือ สามารถเรียกขานและรับการเรียกขานได้เช่นเดียว กับลักษณะของการใช้โทรศัพท์โดยทั่วไป

กล่าวคือทางชุมสายสื่อสารข้อมูลจะกำหนดหมายเลขประจำวงจร ลูกข่ายทุกตัวไว้ เมื่อมีการเรียกขานก็จะเชื่อมต่อวงจรดังกล่าวไปยังหมายเลขที่ลูกข่ายนั้นๆ

DIAL-UP CIRCUIT คือ การสื่อสาร ข้อมูลโดยอาศัย วงจรโทรศัพท์ เป็นพาหะนำ สัญญานโดยชุมสายสื่อสารข้อมูลจะนำเอาวงจร X.28 ต่อเข้ากับ DCE (DATA COMMUNICATION EQUIPMENT) ซึ่งก็คือ MODEM โดย MODEM นี้ในสาย สัญญาน ANALOG จะถูกต่อกับวงจร โทรศัพท์ มายังหมายเลขที่เรา เชื่อมต่อไว้กับชุมสาย ในลักษณะ เดียว ลูกข่ายก็จะใช้ อุปกรณ์ MODEM ต่อเข้ากับ หมายเลขโทรศัพท์เพื่อใช้ในการเรียกออก ในการใช้งานลูกข่าย ก็จะเรียก DIALโทรศัพท์ไปยังชุมสายสื่อสารข้อมูลในขณะที่ MODEM ทั้งสองต่อ ถึงกันโดยมีวงจรโทรศัพท์เป็นสื่อกลางในขณะนั้นก็คือ อุปกรณ์ทั้งสองสามารถ ส่ง สัญญาน ANALOG รับส่งกันได้ DTE ของลูกข่ายก็สามารถ รับส่ง ข้อมูลโดยการเรียกขาน DTE ADDRESS ที่ ต้องการติดต่อด้วยได้ ลักษณะการ สื่อสารข้อมูล ในลักษณะนี้เป็นแบบ UNDIRECTIONAL กล่าวคือลูกข่ายเท่านั้นที่สามารถเป็น ฝ่ายเรียกไปยังปลายทางได้เท่านั้นเนื่องจากชุมสาย ไม่ถูกกำหนดให้หมุน หมายเลขโทรศัพท์มายัง MODEM ของลูกข่ายได้ลักษณะการใช้งานลักษณะนี้จึงใช้กับปริมาณข้อมูลที่รับส่งน้อยและความเร็วในการรับส่งต่ำเนื่องจากคุณภาพของวงจรโทรศัพท์หรือความ เชื่อถือมีน้อยกว่า วงจรสายตรง (LEASED LINE)

PROTOCAL ANALYZER คืออะไร

PROTOCAL ANALYZER คืออุปกรณ์ ที่ใช้ในการตรวจสอบข้อผิดพลาดใน การรับส่งข้อมูลใน PROTOCAL หรือ PROTOCAL นั้นๆ ที่อุปกรณ์ดัง กล่าวกำหนดไว้ซึ่งในการตรวจสอบนั้นสามารถแบ่งลักษณะการเข้าถึงได้ 2 ระดับ คือ

1.ระดับทางกายภาพ (PHYSICAL LEVEL) หมายถึง การเชื่อมต่อกันทางสัญญานไฟฟ้าซึ่งก็คือ การส่งสัญญานกันระหว่าง DTE และ DCE ซึ่งโดย รวมเราสามารถเรียกเป็นสัญญาน HAND SHAKE ซึ่งโดยทั่วไปส่วนใหญ่จะใช้กันใน มาตรฐานของ RS232-C ซึ่งได้แก่สัญญานดังนี้

DTR (DATA TERMINAL READY)เป็นสัญญานที่ TERMINAL หรือ DTEส่งไปยังDCE หรือ MODEM เพื่อให้ MODEM รับทราบว่าขณะนี้ TERMINAL ได้เริ่ม ACTIVE แล้ว

DSR (DATA SET READY) เป็นสัญญานที่ MODEM ตอบไปยัง TERMINAL เมื่อได้รับสัญญาน DTR จาก TERMINAL เพื่อแสดงความ พร้อมเช่นกัน

RTS (REQUEST TO SEND)เป็นสัญญานที่ DTE หรือ TERMINAL แสดงให้ MODEM รับรู้ว่าขณะนี้ TERMINAL พร้อมที่จะส่งข้อมูลไปยัง MODEM แล้ว

CTS(CLEAR TO SEND)เป็นสัญญานที่ DCE หรือ MODEM ตอบรับสัญญาน RTS จากTERMINAL เพื่อแสดงให้รู้ว่า MODEM พร้อมที่จะรับส่งข้อมูลเช่นกัน

CD(CARIER DETECT) สัญญานนี้จะ ACTIVE ก็ต่อเมื่อสัญญาน MODEM สามารถรับสัญญานพาหะ (CARIER)จาก(MODEM)อีกตัวที่ต่ออยู่ได้

SQ(SIGNAL QUALITY)สัญญานนี้จะACTIVE ก็ต่อ เมื่อสัญญานทาง ANALOG ที่ MODEM รับมาได้เป็นสัญญานที่มีความถี่ถูกต้องและมี ระดับสัญญาน สูงพอในการ นำไปใช้ถอดระหัสข้อมูล

RI(RINGING) สัญญานนี้จะ ACTIVE เมื่อถูกใช้ในการต่อกับโทรศัพท์ โดยจะ ACTIVE เมื่อมีสัญญานกระดิ่ง(RING)จากสายโทรศัพท์เข้ามา

TD(TRANSMITTED DATA)เป็นสัญญานที่ใช้ส่งข้อมูล

RD(RECIEVED DATA) เป็นสัญญานที่ใช้รับข้อมูล

2.ระดับข้อมูล (DATA LEVEL) เป็นระดับที่อยู่สูงกว่า PHYSICAL กล่าวคือ จะมีการตรวจสอบข้อมูลในระดับนี้ได้ก็ต่อเมื่อระดับทางกายภาพพร้อมสมบูรณ์เสียก่อน ระดับการตรวจสอบข้อมูลนี้จะเป็นการตรวจสอบข้อมูล ทั้งหมดที่มีการรับส่งกันใน LINE เช่น การตรวจสอบดูว่ามีข้อมูลที่เกิด ERROR หรือไม่ การเรียกขานถูกต้องหรือไม่ เมื่อเกิดการยกเลิกการเรียกขาน(CLEAR CALLED)เราสามารถนำข้อมูลที่ปรากฏมาพิจารณาถึงสาเหตุของการ CLEAR ได้และใช้ในการ ตรวจสอบ อักขระควบคุม(CONTROL CHARACTER) ซึ่งมิใช่ข้อมูลแต่เป็นอักขระที่ใช้สื่อความหมายกันในระดับเครื่องต่อเครื่องด้วยตนเอง ซึ่งใช้ในการควบคุมลักษณะการรับส่งข้อมูลระหว่างกันอีกทั้งยังใช้ในการตรวจสอบผลจากการเซ็ตค่าพารามิเตอร์ต่าง ๆ

ในการใช้งานจริง PROTOCAL ANALYZER มีความจำเป็นมากเปรียบเสมือนอุปกรณ์เครื่องมือวัดที่ใช้ ตรวจสอบ ซ่อมอุปกรณ์ทาง HARDWARE ต่าง ๆ เพราะหากขาดอุปกรณ์นี้แล้ว OPERATER ไม่สามารถ ทราบได้เลยว่าสัญญาน ที่เชื่อมต่อกันอยู่นั้นเป็นอย่างไรบ้างและสาเหตุที่ผิดพลาดอยู่ที่ขั้นตอนไหนอย่างไร ดังนั้นชุมสายสื่อสาร ข้อมูล ทุกแห่งจึงต้องมีอุปกรณ์ดังกล่าวประจำอยู่เพื่อใช้ตรวจสอบข้อผิดพลาดและใช้ในการ LINE UP วงจรต่าง ๆ

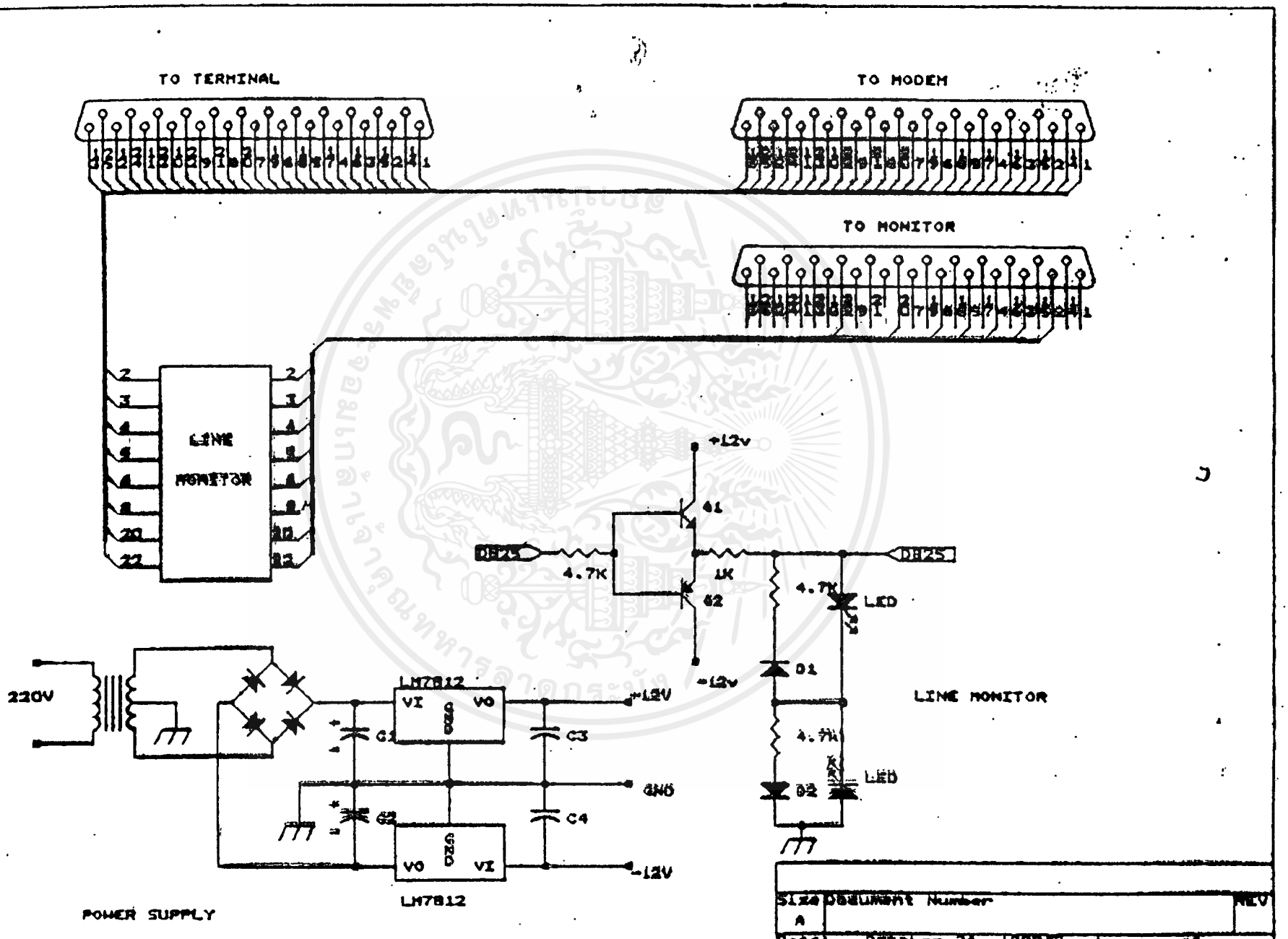
X.28 PROTOCAL ANALYZER

อุปกรณ์ดังกล่าว สร้างขึ้นเพื่อใช้ในการตรวจสอบ PROTOCAL X.28 เท่านั้น มิได้ครอบคลุมไปถึง PROTOCAL อื่นๆ สาเหตุเนื่องจาก PROTOCAL อื่นๆ มี โครงสร้างที่ แตกต่างกันออกไปมากและมีความซับซ้อนทาง SOFTWARE ต่างกัน อุปกรณ์ดังกล่าวถูกสร้าง ขึ้นมาโดยการนำอุปกรณ์ CHIP SUPPORT ของ CPU มาใช้โดย การออกแบบ กำหนด ให้ สามารถใช้งานได้กับ PC COMPATIBLE ทั่ว ๆ ไปได้ โดยในอุปกรณ์ดังกล่าวนี้ได้แบ่งชิ้นส่วนทาง HARD WARE ออกเป็น 2 ส่วนคือ

1. LINE MONITOR

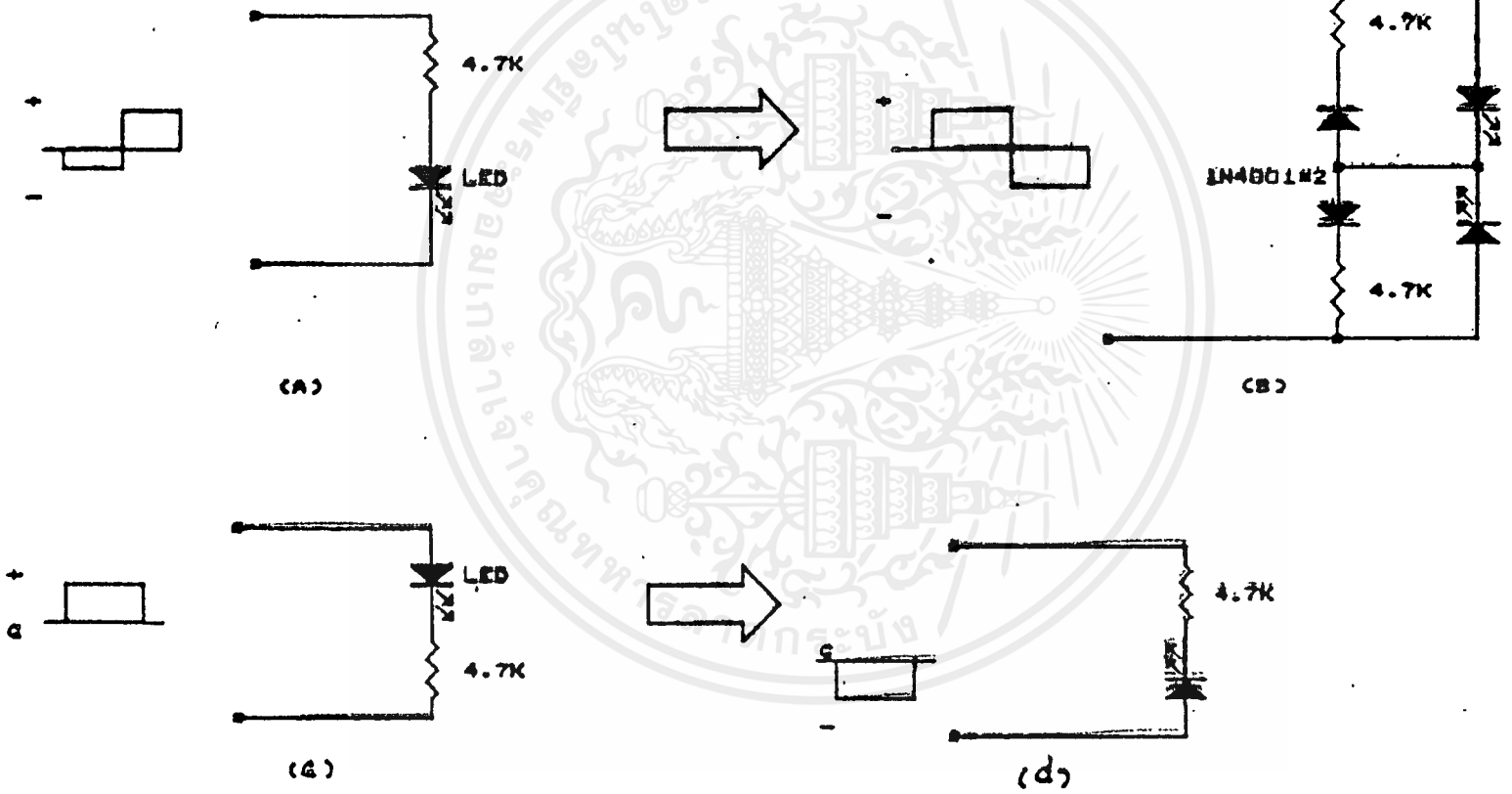
2. PROTOCAL CARD

1.LINE MONITOR เป็นส่วนที่ใช้ในการตรวจสอบสัญญาณทางไฟฟ้าและขยายสัญญาณใน LINE เพื่อใช้ในการตรวจ สอบข้อมูลโดยไม่ทำให้สัญญาณใน LINE ลดระดับความแรง ลงมากเกินไปลักษณะของวงจรเป็นดังในรูปข้างล่างนี้


TO TERMINAL
TO MODEM
TO MONITOR
LINE
MONITOR
+12v
Q1
DB25
4.7K
1K
Q2
-12v
4.7K
LED
D1
4.7K
LED
D2
LINE MONITOR
220V
LM7812
VI
VO
GND
+12V
C1
C3
GND
C2
C4
VO
VI
-12V
LM7812
POWER SUPPLY
Size
A
Document Number
REV

จากวงจรจะเห็นได้ว่าที่ INPUT ถูกจำกัดไว้ให้เป็น HI IMPEDANCE เพื่อไม่ให้ LOAD สัญญาน ที่ใช้งาน โดยมี TRANSISTOR Q_A และ Q_B ทำหน้าที่เป็น SWICTH โดยเมื่อสัญญานเป็น HI Q_A จะทำการ SWICTH ให้ +V ออกทาง OUTPUT เป็น HI LOGIC และในขณะต่อมาหากสัญญานเป็น LOW Q_B ก็จะ SWICTH ให้สัญญาน -V ออกทาง OUTPUT

ในการ DISPLAY สัญญาน โดยทั่วไปจะใช้ LED เป็นตัวแสดงซึ่งอุปกรณ์ที่มี ขายในท้องตลาดจะไม่มีส่วนขยายของสัญญานในบางครั้งหาก LEVEL ที่ใช้งานอยู่ในระดับ ต่ำเมื่อต่ออุปกรณ์ ดังกล่าว อาจทำให้สัญญาน เกิดการผิดพลาดขึ้นได้อุปกรณ์ที่ใช้ในท้องตลาด เป็นอุปกรณ์ที่ นำเข้าจากต่าง ประเทศ LED ที่ใช้เป็น LED ที่สามารถให้แสงได้ ทั้ง FORWARD และ REVERS BIASซึ่งจะให้สีที่แตกต่างกัน แต่อุปกรณ์ดังกล่าวในประเทศไทยยังไม่มีขาย สำหรับโครงงานนี้ ผู้จัดทำได้นำ LED 3 ขาซึ่งมีใช้ในท้องตลาดทั่วไปนำมาดัดแปลง จากรูป
ข้างล่างนี้จะเห็นได้ว่ามีการใช้อุปกรณ์ต่อเพิ่มเข้ามา คือ diode อีก 2 ตัว



Title	protocal analyzer
Size	Document Number
	REV

EQUIVALENT CIRCUIT ในรูป C เป็นการแสดงถึงขณะที่มี HI LOGIC เป็น INPUT กระแสจะไหลผ่าน LED1 และ DIODE ตัวล่างซึ่ง เป็น FORWARD BIAS ผ่าน R 4.7K จะลง GND ครบวงจรและในรูป D เมื่อเป็น INPUT ลบ กระแสจะไหลผ่าน R 4.7K และ DIODE ตัวบนผ่านมายัง LED2 ครบวงจรเช่นกัน โดย DIODE และ R อีก จุดหนึ่งเป็น HI-IMPEDANCE เนื่องจากเป็น REVERSE กับแรงดันที่ผ่านเข้ามาจากวงจร ดังกล่าวจะ ต้องมี POWER SUPPLY เพื่อจ่ายแรงดันให้กับวงจรทั้งหมด เนื่องจากแรงดันใน สาย สัญญานมีกระแสที่ต่ำอยู่แล้ว ไม่สามารถนำมาใช้จ่ายให้กับวงจรได้

2. PROTOCAL CARD อุปกรณ์ตัวนี้ถูก ออกแบบให้ อยู่บนแผ่น บริษท์ ซึ่ง สามารถต่อลงบน SLOT ของ IBM PC ซึ่งส่วนที่ทำ หน้าที่รับ สัญญานจาก LINE MONITORมา เพื่อทำการDISPLAY ออกทางจอ MONITOR อุปกรณ์ตัวนี้ได้ถูกออกแบบให้ใช้งานได้ 3 ลักษณะคือ

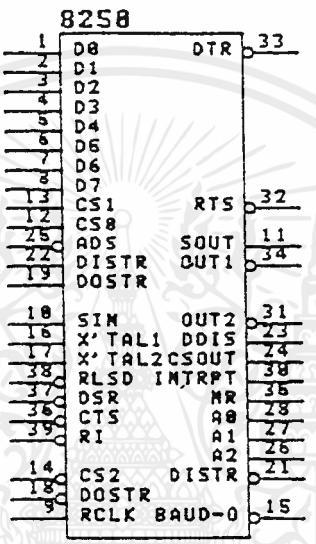
- EMULATED DTE เป็นการสร้าง PORT ดังกล่าวให้เป็นเสมือน TERMINAL ตัวหนึ่งเพื่อใช้ในการตรวจสอบ DCE หรือ MODEM
- EMULATED DCE คือการสร้าง PORT ดังกล่าว ให้เป็นเสมือน DCE หรือ MODEM เพื่อใช้ใน การตรวจสอบDTE หรือTERMINAL

3 MONITOR คือการใช้ INPUT ของ PORT ทั้งสองของวงจรเป็นเป็นตัวนำข้อมูลทั้งด้าน TD และ RD ของการเชื่อมต่อกันระหว่าง MODEM และ TERMINAL มาทำการ DISPLAY เพื่อตรวจสอบข้อมูลทั้งหมด

จากวงจรจะเห็นได้ว่าหัวใจหลักของวงจรอยู่ที่ CHIP 8250 จึงจะขออธิบายถึงรายละเอียดของ CHIP 8250 เสียก่อน

ชิพ 8250

8250 เป็นไอซีขนาด 40 ขามีการทำงานเพื่อควบคุมสิ่งต่าง ๆ ที่เกี่ยวกับการ สื่อสารแบบอนุกรมได้หมด โครงสร้างทางฮาร์ดแวร์ของอะแดปเตอร์การ์ดนี้จึงไม่ยุ่งยากมากนัก การจัดวางขาของไอซี 8250 เป็นดังรูปที่ 3 ขาต่าง ๆ แต่ละขามีรายละเอียดดังนี้



รูปที่ 3 การจัดวางขาของไอซี 8250

สัญญาณอินพุต

ขาเลือกชิพ CS_0, CS_1, CS_2 (chip select) ขา 12 - 14 เป็นสัญญาณเลือกชิพโดยที่เมื่อต้องการเลือกชิพจะให้CS_0CS_1 เป็น "1" และ CS_2 เป็น "0" สัญญาณ เลือกชิพนี้จะมีไว้เพื่อให้ซีพียูติดต่อกับ 8250

สโตรบ ข้อมูลอินพุต

DISTR, DISTR data input strobe ขา 22, 21 เมื่อสัญญาณที่ DISTR เป็น "1" และ DISTR เป็น "0" ในขณะที่มีการเลือกชิพ เป็นขณะที่ซีพียูจะอ่านข้อมูลจากรีจิสเตอร์ภายในที่ได้รับการกำหนดไว้แล้วมายังซีพียู สัญญาณนี้จึง เป็นสัญญาณอ่านข้อมูลหรือ read นั่นเอง

ขาสโตรบข้อมูลเอาต์พุต

DOSTR, DOSTR ขา 19 และขา 18 เป็นสัญญาณที่แอคตีฟขึ้นเพื่อให้ซีพียูเขียนข้อ มลลงมายังรีจิสเตอร์ของ 8250

ขาสโตรบแอดเดรส

ADS ขา 25 เมื่อมีค่าเป็น "o" จะแอคทีฟเพื่อแลตช์ค่า A_o - A_2 เลือกรีจิส เตอร์ภายใน การเลือกรีจิสเตอร์จะทำขณะที่การเลือกชิพแอคทีฟอยู่

ขาเลือกรีจิสเตอร์

A_O, A_1, A_2 ขา 26 - 28 เป็นตัวกำหนดแอดเดรสของรีจิสเตอร์ภายใน เพื่อให้ซีพียูทำการติดต่อกับ 8250 ตามค่ารีจิสเตอร์ที่กำหนด เพื่อการเขียนหรืออ่าน อย่างไร ก็ตามการทำงานของ A_o - A_2 นี้ก็ขึ้น กับสัญญาณเลือกตัวหาร DLLAB divisor latch access bit ซึ่งกำหนดค่ารีจิสเตอร์ได้ดังนี้

DLAB	A_2	A_1	A_0	รีจิสเตอร์
0	0	0	0	บัฟเฟอร์ตัวรับ (อ่าน)
0				รีจิสเตอร์โฮลดิ้ง (เขียน)
0	0	0	1	อินเตอร์รัพต์อีนาเบิ้ล
X	0	1	0	ตัวกำหนดอินเตอร์รัพต์
X	0	1	1	ควบคุมสายสื่อสาร
X	1	0	0	ควบคุมโมเด็ม
X	1	0	1	สถานะสายสื่อสาร
X	1	1	0	สถานะโมเด็ม
X	1	1	1	ไม่ใช้
1	0	0	0	แลคซ์ตัวหาร (LSB)
1	0	0	1	แลคซ์ตัวหาร (MSB)

รีเซต MR - master reset

ขา 35 เมื่อมีค่าเป็น "1" จะรีเซตการทำงานของชิพ 8250 โดยทำให้ค่า ต่าง ๆ ในรีจิสเตอร์ถูกเคลียร์หมด (ยกเว้นบัฟเฟอร์ของตัวรับตัวส่งและตัวหาร) ขณะทำ การรีเซตจะมีผลต่อสัญญาณเอาต์พุตด้วย ผลของการรีเซตแสดงไว้ในตารางที่ 3

สัญญาณนาฬิกาตัวรับ

RCLK - receiver clock ขา 9 เป็นขาที่ตัวรับสัญญาณนาฬิกาเพื่อกำหนดอัตราบอดสัญญาณนาฬิกานี้จะมีค่าเป็น 16 เท่าของที่นำมาใช้

ขาอินพุตข้อมูลอนุกรม

SIN - serial input ขา 10 เป็นขารับข้อมูลอนุกรมจากสายส่งในการเชื่อมโยงการติดต่อสื่อสาร

ขาเคลียทูเซนด์

CTS - clear to send ขา 36 เป็นสัญญาณ ที่ใช้ในการ ติดต่อกับโมเด็ม เงื่อนไขของสัญญาณนี้สามารถเก็บไว้ภายในชิพ 8250 ที่จะให้ซีพียูอ่านไปตรวจสอบได้ โดยเก็บไว้ที่บิต 4 ของรีจิสเตอรร์ สถานะโมเด็ม ส่วนบิต 0 ของ รีจิสเตอร์ แสดงสถานะจะ เป็นตัวบอกว่า CTS ได้เปลี่ยนสถานะไปหลังจากการอ่านครั้งก่อนแล้วหรือไม่

ขาดาต้าเซตรีดี้

DSR - data set ready ขา 37 เมื่อ เป็น "0" จะแสดงว่าโมเด็มหรือ ข้อมูลได้รับการเซ๖เตรียมพร้อมแล้วสำหรับการเชื่อมต่อกับสายสื่อสาร และส่งข้อมูลระหว่าง 8250 กับโมเด็ม สัญญาณ DSR เป็นสัญญาณอินพุตของ 8250 ที่ซีพียูสามารถอ่านไปดูได้ ทางบิตที่ 5 ของรีจิสเตอร์แสดงสถานะ ส่วนบิต 1 ของรีจิสเตอร์แสดงสถานะจะเห็นตัว บอกว่าสัญญาณ DST ได้เปลี่ยนสถานะไปหลังจากที่อ่านครั้งก่อนแล้วหรือไม่

หมายเหตุ

ทั้ง CTS และ DSR เมื่อมีการเปลี่ยนสถานะและถ้าได้รับการอีนาเบิ้ลที่ modem status interrupt จะส่งผลในการสร้างสัญญาณอินเตอร์รัพต์

ตรวจสอบสายสื่อสาร

RLSD - received line signal detect ขา 38 ถ้าเป็น "0" หมายถึง แอคทีฟ คือ 8250 รับสัญญาณตรวจสอบสัญญาณพาหะจาก โมเด็ม ว่า โมเด็มตรวจ สอบสัญญาณพาหะจากโมเด็ม ว่า โมเด็มตรวจสอบได้แล้ว หรือข้อมูลได้รับการเซตแล้ว ซีพียู สามารถตรวจ สอบสัญญาณ นี้ทางบิต 7 ของ รีจิสเตอร์แสดงสถานะ ส่วนบิต 3 จะเป็นบิตที่

แสดงสถานะว่าสัญญาณนี้ได้รับการเปลี่ยนแปลงหลังจากอ่านไปแล้วหรือยัง

ตารางที่ 3 ค่าเริ่มต้นและเอาต์พุตของ 8250

รีจิสเตอร์/สถานะ	การควบคุม	สถานะเมื่อรีเซต
อินเตอร์รัพต์รีจิสเตอร์ภายใน	มาสเตอร์รีเซต	ทุกบิตเป็น "0" (0-3 ถูกกำหนด 4-7 จะเป็นถาวร)
รีจิสเตอร์ interrupt identification	มาสเตอร์รีเซต	บิต 0 เป็น "1" บิต 1-2 เป็น "0" บิต 3-7 เป็น "0" ถาวร
รีจิสเตอร์ควบคุมสาย	มาสเตอร์รีเซต	ทุกบิตเป็น "0"
รีจิสเตอร์ควบคุมโมเด็ม	มาสเตอร์รีเซต	ทุกบิตเป็น "0"
รีจิสเตอร์แสดงสถานะสาย	มาสเตอร์รีเซต	ยกเว้นบิต 5 และ 6 เป็น "1"
รีจิสเตอร์แสดงสถานะโมเด็ม	มาสเตอร์รีเซต	บิต 0-3 เป็น "0" บิต 4-7 เป็นสัญญาณอินพุต
SOUT	มาสเตอร์รีเซต	เป็น "1"
INTRPT (RCVR Errors)	Read LSR/มาสเตอร์-รีเซต	เป็น "0"
INTRPT (RCVR Data Ready)	Read RBR/มาสเตอร์-รีเซต	เป็น "0"
INTRPT (RCVR Data Ready)	Read IIR/Write THR มาสเตอร์รีเซต	เป็น "0"
INTRPT (เปลี่ยนสถานะ-โมเด็ม	Read MSR/มาสเตอร์-รีเซต	เป็น "0"
OUT 2	มาสเตอร์รีเซต	เป็น "1"
RTS	มาสเตอร์รีเซต	เป็น "1"
DTR	มาสเตอร์รีเซต	เป็น "1"
OUTI	มาสเตอร์รีเซต	เป็น "1"

ขาแสดงวงจรเรียก RI - ring indicator

ขา 39 สัญญาณนี้แอคทีฟด้วยลอจิก "0" เป็นสัญญาณที่ส่งมาจากโมเด็ม โมเด็ม ตรวจสอบสัญญาณการเรียก (ringing)สัญญาณ นี้ตรวจสอบได้ทางบิต 6และดูสถานะการเปลี่ยนหลังจากอ่านแล้วจากบิต 2

ขาไฟเลี้ยง V_{cc} ขา 40 V_{ss} ขา 20

เป็นสัญญาณจากแหล่งจ่ายไฟเลี้ยง 5 โวลต์และกราวด์

Request to Send RTS

ขา 32 เริ่มขานี้มีลอจิกเป็น "0" หมายความว่า 8250 พร้อมที่จะส่งข้อมูล แล้ว สัญญาณขานี้จะได้รับการเซตให้แอคทีฟด้วยการโปรแกรมค่าลงไปในรีจิสเตอร์ควบคุมที่บิต 1

เอาต์พุต 1

OUT1 ขา 34 เป็นขาที่ผู้ใช้สามารถโปรแกรมให้แอคทีฟเป็น "0" ด้วยการโปรแกรมลงไปในบิต 2 ของรีจิสเตอร์ควบคุมโมเด็ม

เอาต์พุต 2

OUT2 ขา31 เป็นขาที่ผู้ใช้สามารถโปรแกรมให้แอคทีฟเป็น "0" ด้วยการโปรแกรมค่าลงในรีจิสเตอร์ควบคุมโมเด็มทางบิต 3

เลือกชิพเอาต์

CSOUT - chip select out ขา 24 เมื่อมีค่าเป็น "1" จะบอกว่าชิพนี้ได้รับการเลือกชิพโดยซีพียูทางขา CS_0, CS_1 และ CS_2

ไดร์ฟเวอร์ดีสเอเบิ้ล

DDIS - driver disable ขา 23 เป็นลอจิก "0" เมื่อซีพียู กำลังอ่านข้อมูลจาก 8250 สัญญาณ DDIS เป็น "1" มีไว้ สำหรับการ ดิสเอเบิ้ล การรับส่งภายนอก ในกรณีที่ใช้ 8250 กับซีพียู ผ่านทางบิต D_0 - D_7 เพื่อ บอกเวลาว่าซซีพียยูเป็น 8250 ติดต่อกันอย่างไร

สัญญาณบอดอต

BAUDOUT ขา 15 เป็นสัญญาณนาฬิกาที่มีความถี่เป็น 16 เท่าของสัญญาณนาฬิกา แล้วหารด้วยค่าที่โปรแกรมกำหนดในตัวหาร

อินเตอร์รัพต์

INTRPT - ขา 30 เมื่อเป็น "1"" เป็นการ ส่งสัญญาณ อินเตอร์รัพต์ออกไปจาก 8250

ข้อมูลเอาต์พุต

SOUT ขา 11 เป็นขาที่ใช้ส่ง ข้อมูลอนุกรมออกไปยังสายสื่อสารสัญญาณ

อินพุตเอาต์พุต

ข้อมูล D_6 - D_7 เป็นสัญญาณต่อเชื่อมกับบัสข้อมูลของระบบ

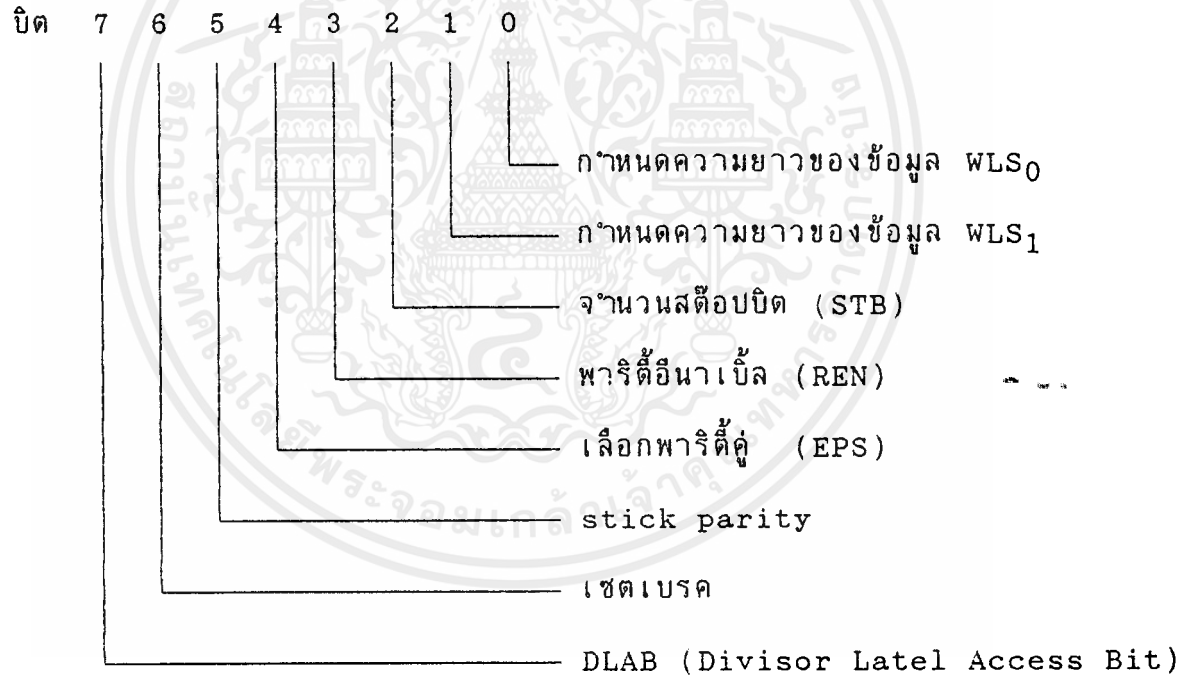
ขาสัญญาณ $XTAL_1$, $XTAL_2$ ขษ 16 - 17 เป็นขาต่อกับคริสตอล เพื่อ สร้างสัญญาณนาฬิกา

รีจิสเตอร์ต่าง ๆ บน 8250

เพียงจากการใช้ งาน บอร์ดอะแดปเตอร์ สื่อสารที่ จะต้อง โปรแกรมค่าไมโครโค้ดเข้าไปใน 8250 ก่อน ดังนั้นผู้ใช้งาน 8250 ก่อนดังนั้น ผู้ใช้งาน 8250 จำเป็น ต้องเข้าใจส่ารีจิสเตอร์ของ 8250 มีความหมายอย่างไร

รีจิสเตอร์ควบคุมสายสื่อสาร (line control register)

ในการควบคุมรูปแบบของข้อมูลแบบอะซิงโครนัสนั้น ผู้โปรแกรมจะ ต้องกำหนดค่าลงในรีจิสเตอร์ควบคุมสายสื่อสาร รีจิสเตอร์ ตัวนี้มี 8 บิต โดยแต่ละบิตมี ความหมายดังนี้



บิต 0 และ 1 เป็นตัวกำหนดความยาวของข้อมูลในการรับส่ง โดยที่

บิต 0	บิต 1	
0	0	หมายถึง ข้อมูลขนาด 5 บิต
0	1	หมายถึง ข้อมูลขนาด 6 บิต
1	0	หมายถึง ข้อมูลขนาด 7 บิต
1	1	หมายถึง ข้อมูลขนาด 8 บิต

บิต 2 เป็น บิตที่ใช้ในการ กำหนดจำนวน สต๊อบบิต ถ้าเป็น "0" หมายถึง ใช้ สต๊อบบิต1บิต แต่ถ้าบิต2 เป็น "1" ในกรณีส่งแบบ 5 บิต จะมีความยาวของสต๊อบบิตเป็น 1.5 บิต แต่ถ้าส่งแบบ 6, 7 หรือ 8 บิต ความยาวของสต๊อบบิตจะเป็น 2

บิต 3 บิต นี้เป็นบิตแสดงการอีนาเบิ้ลให้มีการตรวจสอบพาริตี้ โดยถ้าบิตนี้มีค่าเป็น 1 จะมีการเพิ่มพาริตี้

บิต 4 บิต มีค่าเป็น "0" และบิต 3 มีค่าเป็น "1" จะมี การกำหนดเป็น พาริตี้คี่ แต่ถ้าบิตนี้มีค่าเป็น "1" จะเป็นพาริตี้คู่

บิต 5 เมื่อบิต 3 มีค่าเป็น "1" และบิต 5 มีค่าเป็น "1" และบิต 4 มีค่าเป็น "1" จะมีการแทรกหรือตรวจสอบพาริตี้ด้วยเงื่อนไจกำหนดให้เป็น"0" และถ้าบิต 4 มีค่าเป็น "0" บิต 3 มีค่าเป็น "1" และบิต 5 มีค่าเป็น "1" จะมีการกำหนดบิตพาริตี้เป็น "1"

บิต 6 เป็นบิต ที่ควบคุมการเบรคเมื่อบิต 6 มีค่าเป็น "1" ส่วนของ SOUT จะได้รับการกำหนดให้เป็น "0" ตลอด

บิต 7 บิตนี้ทำหน้าที่เป็น DLAB บิตที่จะมีผลต่อการแลตช์ตัวหารดังที่กล่าวมาแล้วในตารางที่ 1 และ 2

การโปรแกรมอัตราบอด (boud rate generator)

อัตราบอดได้รับการกำหนดเทียบกับสัญญาณนาฬิกา 1.8432 MHz และสามารถโปรแกรมตัวหารได้ตั้งแต่ 1-(2^{16}-1) ค่าความถี่ เอาต์พุตของตัวกำหนดอัตราบอดมี ค่าเท่ากับ 16 x อัตราบอด ดังนั้นตัวหาร = ความถี่สัญญาณนาฬิกา/(อัตราบอด x 16) การกำหนดอัตราบอดด้วยการกำหนดตัวหารนี้ ตัวหารจึงเป็นค่าที่กำหนดในรีจิสเตอร์ 2 ตัว ตัวหารนี้จะต้องถูก กำหนดค่า ก่อนแล้วโปรแกรมลงใน รีจิสเตอร์นี้ การกำหนดต้องให้ DLAB = 1 แล้วให้ ลดลงมาในรีจิสเตอร์ $3F_8$ ซึ่งเรียงกันเป็น LSB ของตัวหาร ส่วน $3F_9$ เมื่อ DLAB = 1 จะเป็นค่าของตัวหาร MSB ค่าของตัวหารเมื่อเทียบกับสัญญาณ 1.8432 MHz เป็นดังตารางที่ 4

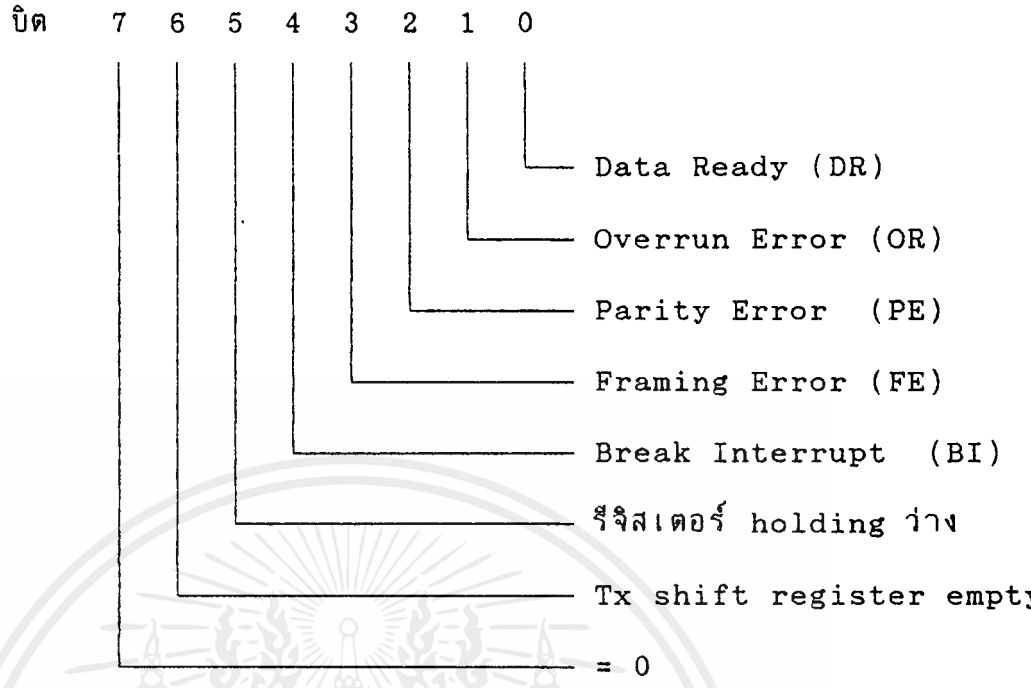
ตารางที่ 4 ค่าตัวหารสำหรับการกำหนดอัตราบอด

อัตราบอด	ตัวหาร		ค่าผิดพลาด
	ฐานสิบ	ฐานสิบหก	
50	2304	900	-
75	1536	600	-
110	1047	417	0.026
134.5	857	359	0.058
150	768	300	-
300	384	180	-
600	192	0C0	-
1200	96	060	-
1800	64	040	-
2000	58	03A	0.69
2400	48	030	-
3600	32	020	-
4800	24	018	-
7200	16	010	-
9600	12	00C	-

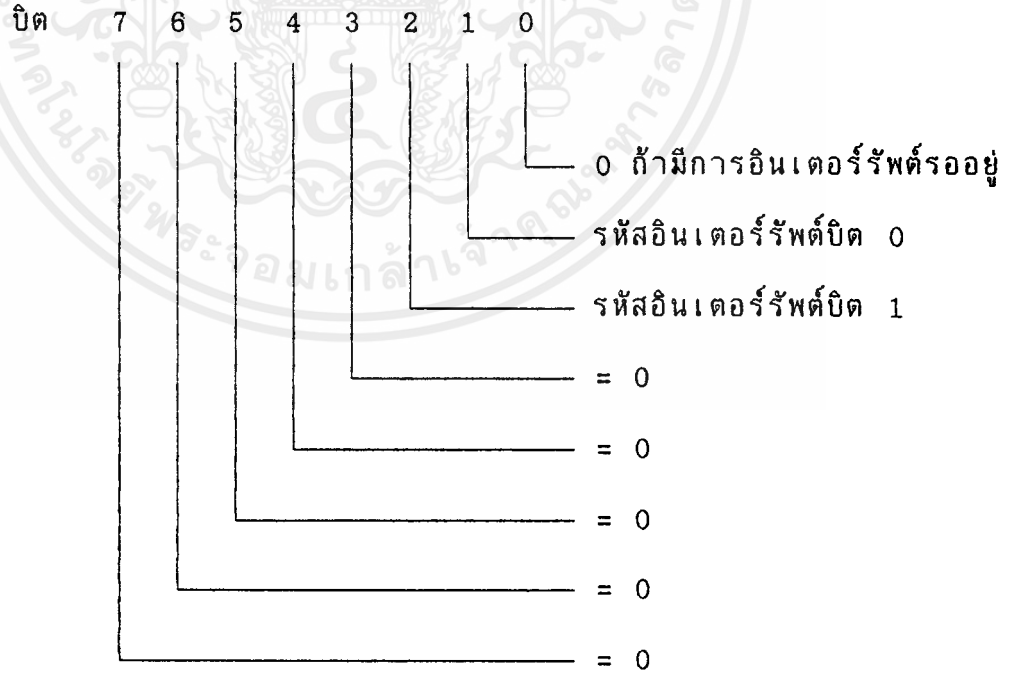
รีจิสเตอร์แสดงสถานะสายสื่อสาร (line status register)

รีจิสเตอร์ตัวนี้เป็นรีจิสเตอร์ที่จะให้ข้อมูลแก่ซีพียูที่เกี่ยวกับการสื่อสารข้อมูลในสายสื่อสาร ค่าของบิตต่าง ๆ ในรีจิสเตอร์เป็นดังนี้

แอดเดรส 3FD



แอดเดรส 3FA



บิต 0 บิตนี้เป็นบิตที่บอกสถานะการรับข้อมูล ถ้าบิตนี้เป็น "I" แสดงว่า การรับข้อมูลเข้า มาในบัฟเฟอร์ได้ครบทุกบิตแล้ว บิตนี้จะได้รับการรีเซตให้เป็น "0" เมื่อซีพียูได้อ่านข้อมูลใน บัฟเฟอร์ไปแล้ว หรือจะให้ซีพียู เขียนข้อมูล กลับมายังรีจิสเตอร์นี้ก็ได้ บิต1บิตนี้ถ้ามีค่าเป็น "I" แสดงว่าเกิด overrun error กล่าวคือขณะที่มีข้อมูลที่ บัฟเฟอร์แต่ซีพียูยังไม่ได้อ่านไป ปรากฏว่ามี ข้อมูลชุดใหม่ มาเขียนทับบนบัฟเฟอร์นี้ บิตนี้จะรีเซตโดยซีพียูเมื่อซีพียูอ่านค่าจากรีจิสเตอร์นี้ไปแล้ว

บิต 2 บิตนี้ถ้ามีค่าเป็น "I" แสดงว่าเกิด parity error กล่าวคือถ้ามีการตรวจสอบบิตพาริตี้แล้วไม่เป็นไปตามที่กำหนดไว้ บิตนี้จะได้รับการรีเซตโดยซีพียูเมื่อซีพียูอ่านค่าจากรีจิสเตอร์นี้ไปแล้ว

บิต 3 บิตนี้ถ้ามีค่าเป็น "I" แสดงว่าเฟรมของข้อมูลไม่เป็นไปตามที่กำหนด เช่นตรวจสอบจำนวนบิตโดยดูที่พาริตี้และสต๊อป บิตไม่เป็นไปตามที่กำหนด

บิต 4 บิตนี้เรียกว่า break interrupt (BI) บิตนี้จะได้รับการเซตให้มีค่าเป็น "I" ถ้าหากว่ารับข้อมูลอินพุตเป็น "0" เป็นเวลายาวนานกว่าเวิร์ดของการสื่อสาร

บิต 5 บิตนี้เป็นบิตที่บอกว่า 8250 พร้อมที่จะรับข้อมูลจากสายสื่อสาร บิตนี้จะได้รับการเซตให้มีค่าเป็น "I" บิตนี้ยังคงสร้างสัญญาณอินเตอรัพต์ เพื่อส่งไปบอกซีพียูด้วยบิตนี้จะมีสถานะเซตเมื่อมีการส่งถ่ายข้อมูลจากโฮลด์รีจิสเตอร์ไปยังชิพรีจิสเตอร์เพื่อพร้อมที่ จะส่ง

บิต 6 เป็นบิตที่จะบอกว่าชิพรีจิสเตอร์ว่างเปล่า บิตนี้จะได้รับการเซตให้มีค่าเป็น "I" เพื่อบอกว่าพร้อมส่งแล้ว

บิต 7 จะเป็น "0" ตลอด

รีจิสเตอร์กำหนดอินเตอร์รัพต์ (IIR-Interrupt Identification Register)

ไอซี 8250 มีขีดความสามารถในการส่งอินเตอร์รัพต์ภายในชิพ เพื่อว่าให้การทำงานระหว่าง 8250 กับซีพียูเป็นไปอย่างมีประสิทธิภาพสูง และ เพื่อให้ ผู้เขียนซอฟต์แวร์เขียนซอฟต์แวร์ได้ง่ายและ สั้นลงได้มาก 8250 กำหนดความ สำคัญของอินเตอร์รัพต์ไว้ 4ระดับ คือ ระดับแรก-สถานะการรับ ข้อมูลจาก สายสื่อสาร ระดับสอง-การพร้อมรับข้อมูล ระดับสาม-ขณะรีจิสเตอร์โฮลดิ้งสำหรับส่งข่าว ระดับสี่-สัญญาณสถานะโมเด็ม ในขณะที่มีความต้องการ อินเตอร์รัพต์หลาย ระดับพร้อมกัน 8250 จะให้ระดับที่มีความสำคัญน้อยกว่ารอไว้ก่อนโดยเก็บสถานะการอินเตอร์รัพต์นี้ไว้ใน interrupt indentification register

ความหมายของรีจิสเตอร์นี้มีดังนี้

บิต 0 เป็นบิตที่ใช้แสดงว่ามีอินเตอร์รัพต์เกิดขึ้นหรือไม่ ซึ่งสามารถให้ซีพียู ตรวจสอบ ดูด้วยวิธีการ polling ได้ถ้าบิตนี้เป็น "1" หมายถึง ไม่มีอินเตอร์รัพต์เกิดขึ้น

บิต 1 - 2 เป็นบิตที่แสดงความหมายบอกว่าการอินเตอร์รัพต์ที่เกิดขึ้นนั้นมาจากอินเตอร์รัพต์ตามฟังก์ชันใด

บิต 3 - 7 มีค่าเป็น "0"

รีจิสเตอร์อีนาเบิ้ลอินเตอร์รัพต์ (IN-TRPT-Interrupt Enable Register)

ใน COM_1 เมื่อให้ DLAB = 0 พอร์ต $3F_9$ จะเป็นรีจิสเตอร์อีนาเล้ิล อินเตอร์รัพต์ ผู้ใช้สามารถกำหนดให้เกิดอินเตอร์รัพต์หรือไม่ก็ได้ โดยการกำหนดค่าลงในรีจิสเตอร์นี้ จากที่กล่าวแล้วว่าการอินเตอร์รัพต์ของ 8250 นี้มี 4 แบบ ดังนั้นจึงต้องกำหนด การอีนาเบิ้ลได้ทั้ง 4 แบบ โดยการใช้ข้อมูลแต่ละบิตของรีจิสเตอร์นี้เพื่อการกำหนดการอีนาเบิ้ลข้อมูลที่อยู่ในรีจิสเตอร์นี้มีความหมายดังนี้

บิต 0 บิตนี้ได้รับการเซตเป็น "1" เมื่อต้องการอีนาเบิ้ลอินเตอร์รัพต์การพร้อมรับข้อมูล

บิต 1 บิตนี้ได้รับการเซตเป็น "1" เมื่อต้องการอีนาเบิ้ลอินเตอร์รัพต์โฮลดิ้งรีจิสเตอร์ว่าง

บิต 2 บิตนี้ได้รับการเซตเป็น "1"เมื่อต้องการอีนาเบิ้ลอินเตอร์รัพต์จากสถานะการรับข้อมูลจากสายสื่อสาร

บิต 3 บิตนี้ได้ รับการเซตเป็น"1"เมื่อต้องการ อีนาเบิ้ลอินเตอร์รัพต์จากสถานะโมเด็ม

บิต 4 - 7ได้รับการกำหนดให้เป็น 0 เสมอ

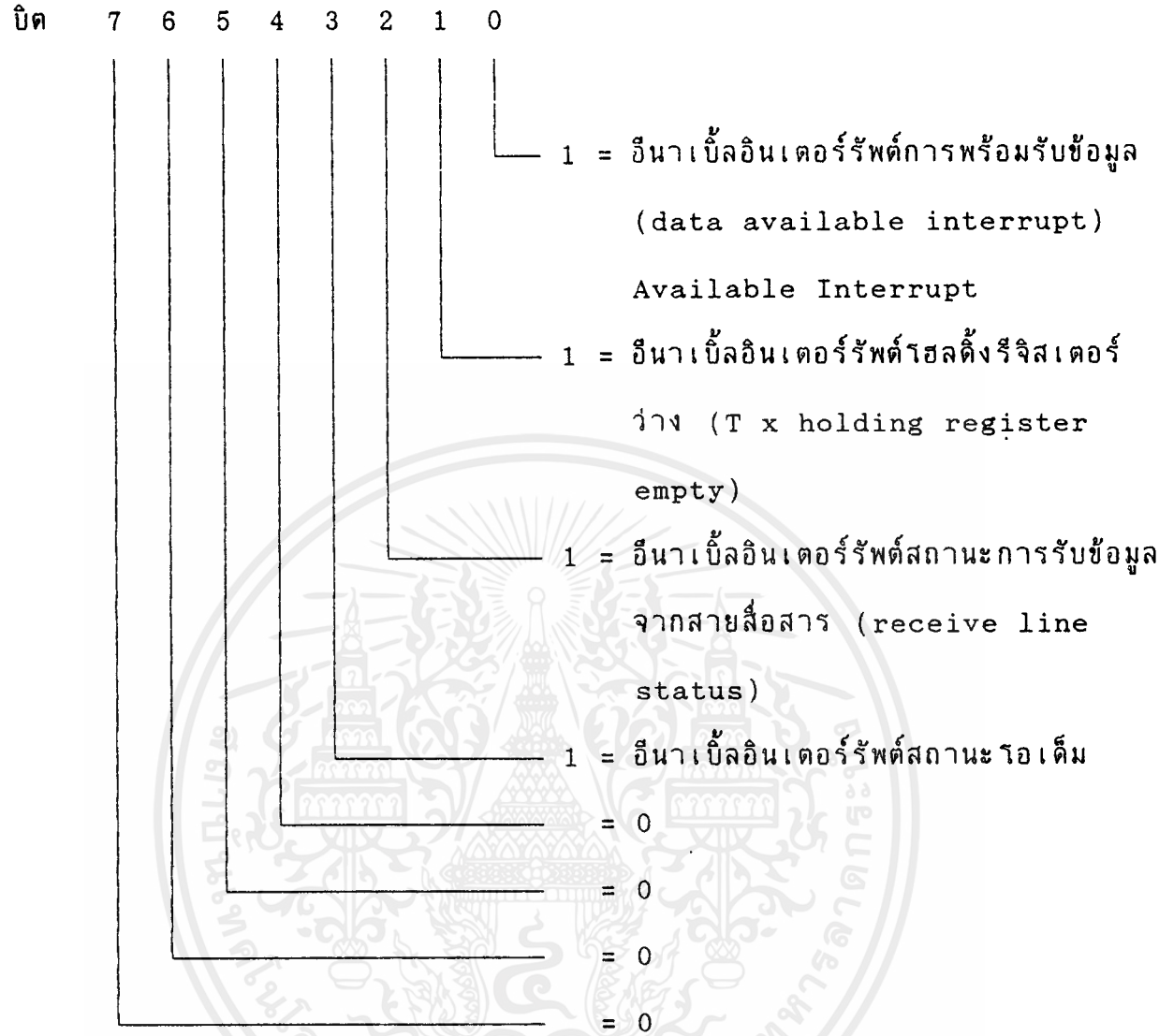
รีจิสเตอร์ควบคุมโมเด็ม (modem control-register)

รีจิสเตอร์ตัวนี้มีไว้ให้ซีพียูส่งผ่านข้อมูลมาเก็บเพื่อเป็นรหัสสำหรับควบคุมการทำงานของโมเด็ม การกำหนดพอร์ตของรีจิสเตอร์รัวนี้ คือ 3FC ข้อมูลต่าง ๆ ที่ มีในรีจิสเตอร์ ตัวนี้มีความหมายดังนี้

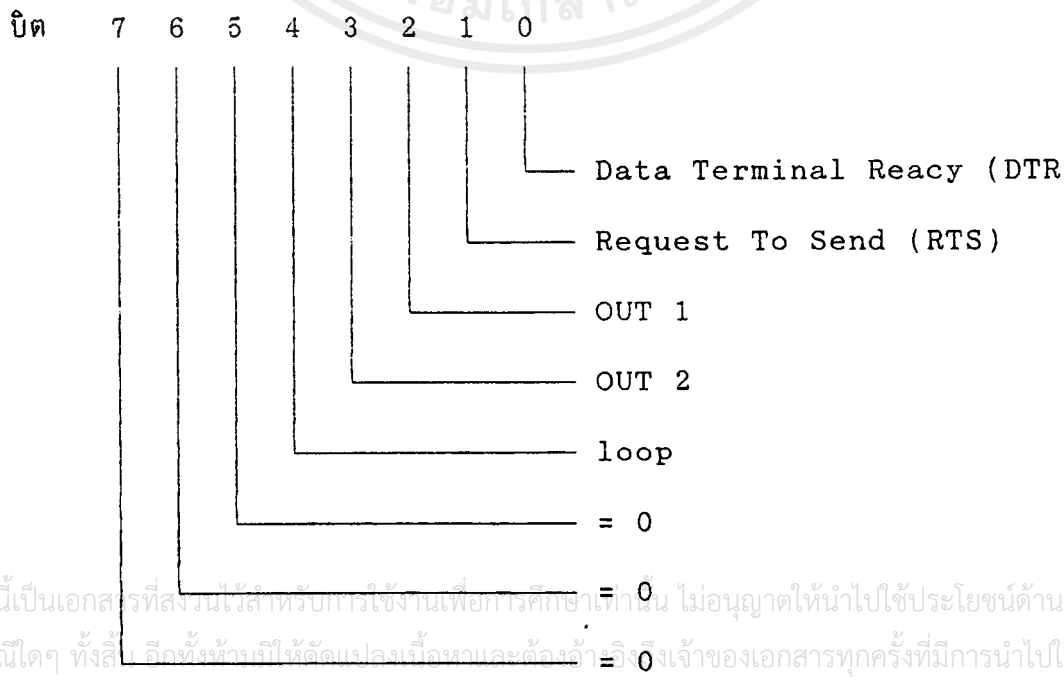
ตารางที่ 5 แสดงฟังก์ชันการอินเตอร์รัพต์รหัสอินเตอร์รัพต์

บิต 2	บิต 1	บิต 0	ระดับความสำคัญ	ชนิดของอินเตอร์รัพต์	แหล่งเกิดอินเตอร์รัพต์	การรีเซตควบคุมอินเตอร์รัพต์
0	0	1	สูงสุด	ไม่เกิด	ไม่เกิด	อ่านข้อมูลจาก
1	1	0		สถานะการรับข้อมูลจากสายส่งสื่อสาร	Overrun Error Parity Error Framming Error Break interrupt	รีจิสเตอร์ สถานะสายสื่อสาร
1	0	0	ที่สอง	การพร้อมรับข้อมูล	มีข้อมูลที่ตัวรับ	การอ่านข้อมูลจากบัฟเฟอร์
0	1	0	ที่สาม	โฮลดิ้งรีจิสเตอร์สำหรับส่งข่าว	โฮลดิ้งรีจิสเตอร์สำหรับส่งข่าว	อ่านรีจิสเตอร์กำหนดอินเตอร์รัพต์ IIR หรือเขียนลงไปยังโฮลดิ้งรีจิสเตอร์สำหรับส่ง
0	0	0	ที่สี่	สถานะโมเด็ม	CLS DSR RI ตรวจสอบสายส่งโดยตรง	อ่านรีจิสเตอร์แสดงสถานะของโมเด็ม

กำหนดพอร์ต 3F9 DLAB = 0



พอร์ตหมายเลข 3FC



บิต 0 บิตนี้ มีความหมาย ถึงการควบคุมสัญญาณ DTR เมื่อบิต นี้มีค่า เป็น "1" เอาต์พุตที่ DTR จะได้รับการกำหนดให้เป็น "0" เอาต์พุตที่ DTR จะได้รับการกำหนดให้เป็น "1"

บิต 1 บิตนี้มีความหมายถึงสัญญาณ RTS ซึ่งจะมีผลเหมือนกับบิต 0 ในกรณีของ DTR

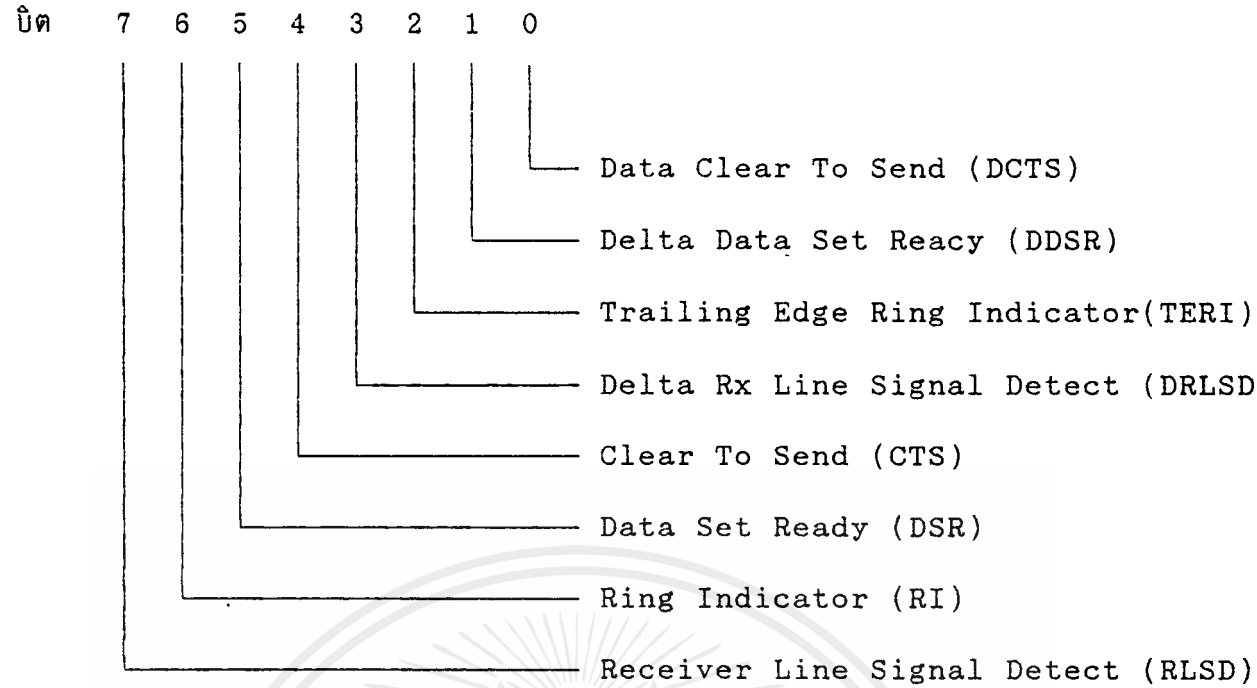
บิต 2 บิตนี้ใช้ควบคุมขาเอาต์พุต 1 (OUT 1) ซึ่งจะมีผลเหมือนบิต 0

บิต 3 บิตนี้ใช้ควบคุมเอาต์พุต 2 (OUT 2) ซึ่งจะมีผลเหมือนบิต 0

บิต 4 บิตนี้จะใช้สำหรับการกำหนดวงรอบสำหรับการตรวจสอบ 8250 เมื่อบิต 4 นี้ได้รับการเซตเป็น "1" สิ่งที่จะเกิดขึ้นเป็นดังนี้ ข้อมูลที่ SOUT จะได้รับการเซต ให้เป็นลอจิก "1" ขาข้อมูลอินพุต SIN จะได้รับการ แยกตัวออก ข้อมูลเอาต์พุตชิพรีจิสเตอร์ จะได้รับการป้อนกลับมายังรีจิสเตอร์ข้อมูลอินพุตส่วนสัญญาณ CTS, DSR, RLSD และ RI จะ ได้รับการแยกออกจากระบบแต่ สัญญาณควบคุมโมเด็มคือ DTR, RTS, OUT 1 และ OUT 2
จะต่อเข้ากับสัญญาณทั้งสี่ที่เป็นอินพุต ดังนั้นจึงตรวจสอบระบบการทำงานได้

รีจิสเตอร์แสดงสถานะโมเด็ม

รีจิสเตอร์ตัวนี้จะเป็นตัวที่รับสถานะจากโมเด็มมาเก็บไว้ เพื่อให้ซีพียูสามารถอ่านตรวจสอบดูได้ สถานะของข้อมูลจะแอคทีฟเมื่อมีข้อมูลเป็น "1" และจะได้รับการรี เซตเมื่อซีพียู อ่านข้อมูลในรีจิสเตอร์นี้ไป พอร์ตที่ใช้กำหนดเป็นพอร์ตหมายเลข 3FE ข้อมูลภายในรีจิสเตอร์นี้เป็นดังนี้



บิต 0 นี้ใช้สำหรับแสดงการเปลี่ยนแปลงของสัญญาณ CTS กล่าวคือ เมื่อขา CTS ของ 8250 ได้เปลี่ยนสถานะหลังจากที่ซีพียูได้อ่านสถานะ นี้ไปแล้ว บิตนี้ก็จะบอกด้วยการเซตและเมื่อซีพียูอ่านก็จะได้รับการรีเซ๖ ("0") เมื่อมีการเปลี่ยนสถานะที่ขา CTS

บิต 1 เหมือนบิต 0 แต่เป็นบิตที่แสดงสถานะการเปลี่ยนแปลงของขา DSR

บิต 2 บิตนี้เป็นบิตแสดงว่าสัญญาณ RI ซึ่งเป็นอินพุตของ 8250 ได้รับการเปลี่ยนจากออน ("1") มาเป็นออฟ ("0")

บิต 3 บิตนี้เหมือนบิต 0 แต่เป็นบิตแสดงสถานะการเปลี่ยนแปลงของ line signal detector ซึ่งเป็นขาอินพุต RLSD

บิต 4 บิตนี้เก็บสัญญาณคอมพลีเมนต์กับสัญญาณที่ขา CTS

บิต 5 บิตนี้เก็บสัญญาณคอมพลีเมนต์กับสัญญาณที่ขา DSR

บิต 4 บิตนี้เก็บสัญญาณคอมพลีเมนต์กับสัญญาณที่ขา RI

บิต 4 บิตนี้เก็บสัญญาณคอมพลีเมนต์กับสัญญาณที่ขา RLSD

อนึ่งถ้าบิต 4 ของ MCR ได้รับการเซตหรือให้ทำ Loop ตรวจสอบข้อมูลใน บิต 5จะเหมือนกับ RTS ใน MCR ข้อมูลในบิต 5 จะเหมือนกับ DTR ใน MCR ข้อมูลใน
บิต 6 จะเหมือนกับ OUT_1 ใน MCR ข้อมูลในบิต 7 จะเหมือนกับ OUT_2 ใน MCR

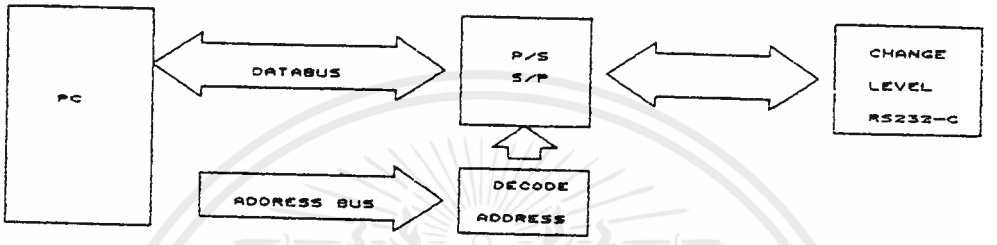
บัฟเฟอร์รีจิสเตอร์สำหรับตัวรับข้อมูล (receiver buffer register)

เป็นรีจิสเตอร์สำหรับการรับข้อมูลที่มาจากสายสื่อสารสัญญาณ พอร์ตที่กำหนดคือหมายเลขแอดเดรส 3F8 ขณะที่ DLAB = 0 หากซีพียูอ่านข้อมูลที่รีจิสเตอร์นี้ก็หมายถึงได้ อ่านข้อมูลที่มาจากสายสัญญาณสื่อสารนั่นเอง

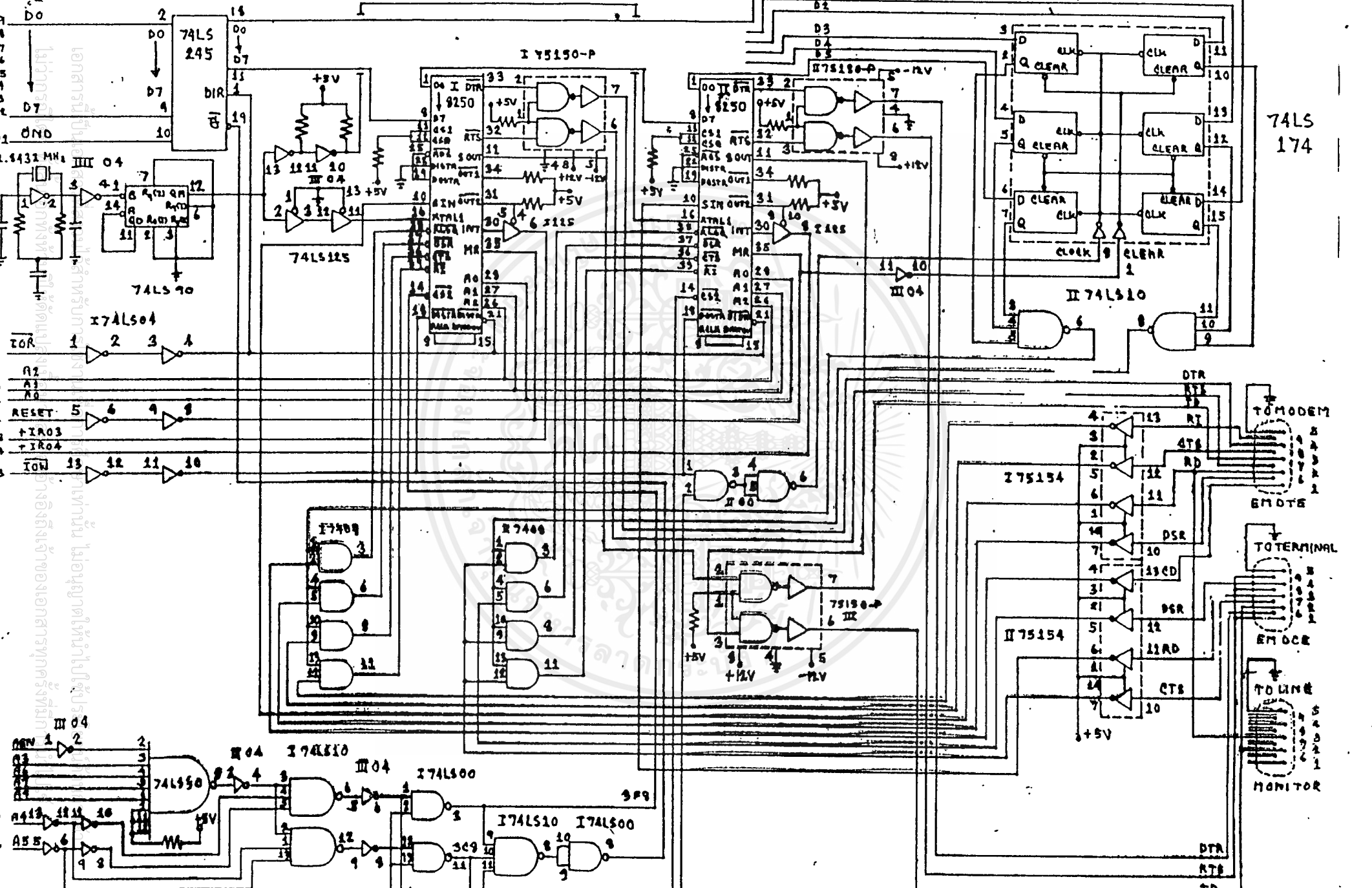
รีจิสเตอร์โฮลดิ้งสำหรับตัวส่งข้อมูล (transmitter holding register)

เป็นรีจิสเตอร์บัฟเฟอร์สำหรับส่งข้อมูล รีจิสเตอร์นี้จะรับข้อมูลจากซีพียู โดยที่ กำหนดพอร์ตเป็นหมายเลข 3f8 เมื่อ DLAB = 0 ข้อมูลของซีพียูที่เอาต์พุตมาที่พอร์ตนี้ก็เพื่อจะส่งต่อออกไปยังสายสารข้อมูล

การทำงานในส่วนต่างๆของวงจรสามารถอธิบายโดยอาศัยบล๊อกไดอะแกรม ข้างล่างนี้



ส่วนที่อยู่ในเส้นประ คือส่วนที่ทำหน้าที่ PROTOCAL CARD D/S, S/P คือ ส่วนที่ทำ หน้าที่เปลี่ยนขอ้มูลจาก PARALLEL เป็น SERIAL และในทางกลับกันก็เปลี่ยนจาก SERIAL TO PARALLEL ป้อนให้กับ PC DIODE ADRESS จะทำหน้าที่กำหนด ADDRESS ให้กับ PORT แต่ละตัวของชุด PORT SERIAL ส่วนชุดสุดท้ายจะทำหน้าที่เปลี่ยน LEVEL ของระบบ MICRO ให้อยู่ในระบบของ RS232-C เพื่อใช้ติดต่อกับอุปกรณ์ภายนอกได้

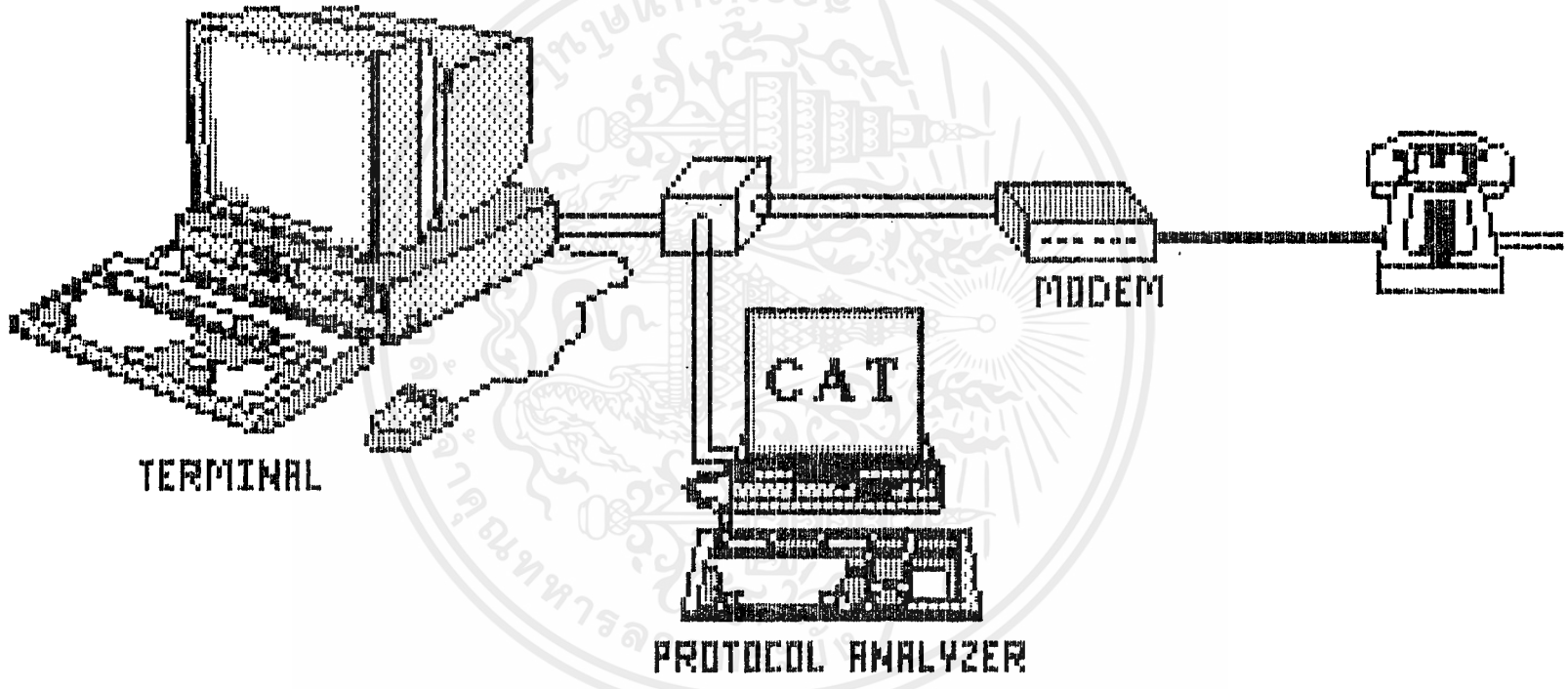

74LS 245
I 75150-P
9250
74LS125
74LS 90
74LS 174
II 74LS10
I74LS04
III 04
I74LS10
I74LS00
74LS50
I 7408
II 7408
II 75154
I 75154
TO MODEM
TO TERMINAL
TO LINE
MONITOR
DTR
RTS
RI
RD
DSR
CD
CTS
RD
+5V
+12V
-12V

จากวงจรส่วนที่ทำหน้าที่ DECODE ADDRESS จะประกอบด้วย TTL ในชุด ล่างของวงจรทำหน้าที่ DECODE ADDRESS ซึ่งใช้กำหนด ให้ ADDRESS 3F8เป็น ของ PORT ตัวที่ 1
และ 2F8 เป็น ADDRESS ของ PORT 2 และ ADDRESS 3C8 ใช้ใน การ LATCH ค่าเพื่อ LOOP ขาต่างๆของ CHIP 8250 เพื่อใช้ในการ MONITOR ข้อมูล

ส่วนที่ทำหน้าที่ปรับระดับสัญญานจาก TTL LEVEL ให้เป็น RS232-C LEVEL ประกอบด้วย 75154 ซึ่งใช้เป็น INPUT ปรับจาก RS232-C เป็น TTL LEVEL และ 75150-Pใช้ในการปรับจาก TTL LEVEL เป็น RS232-C

ส่วนที่ทำหน้าที่เปลี่ยนข้อมูลจาก SERIAL TO PARALLEL และ PARALLEL TO SERIAL คือ CHIP 8250 ทั้ง สองตัวทำหน้าที่เป็น SERIAL PORT1 และ PORT 2 โดย PORT1 นั้นใช้เป็น EMULATED DTE และใช้ RDของ PORT นี้เป็นตัว จับสัญญาน TRANSMITTEDใน LINE ที่เราทำการ MONITOR ส่วน PORT2ใช้ทำหน้า ที่เป็น EMULATED DCEและยังใช้ RD ของ PORT นี้ใช้ในการจับสัญญาน RECIEVED ของ LINE ที่เราทำการ MONITOR ด้วย

ส่วนสำคัญอีกส่วนหนึ่งเพื่อให้อุปกรณ์ดังกล่าวใช้งานได้ คือส่วน ของ SOFT WARE ที่ทำหน้า ที่สั่ง การให้ PC ทำงานตาม ที่เรากำหนด นั่นคือ การนำ ข้อมูล DISPLAY และกำหนดรูปแบบการแสดงผล การแสดงส่วนของสัญญานด้านรับและส่ง การ แยกส่วนของข้อมูลในส่วนที่เป็นข้อมูลและอักขระควบคุม การเก็บข้อมูลที่ MONITOR ไว้ ในหน่วยความจำ ตลอดจนการกหนดความ
เร็วในการ MONITOR และชนิดของ PARITY รวมทั้งขนาดของ STOP BIT ด้วย


CAT
TERMINAL
MODEM
PROTOCOL ANALYZER

หลักการออกแบบ

สิ่งที่ต้องคำนึงถึงในการออกแบบคือจุดที่นำเครื่องมือตรวจสอบโพรโทคอลเข้าไปต่อในที่นี้จุดที่ นำเข้าไปต่อคือจุดที่อยู่ระหว่าง DTE (Data Terminal Equipment) และ DCE (Data Circuit Terminating Equipment)ดังรูป สัญญาณที่อยู่ระหว่างจุดนั้นมีขนาดของแรงดับไฟตรงเท่ากับ 12 โวลต์ เป็นสัญญาณเชิงเลข จุดประสงค์ของการต่ออุปกรณ์ตรวจสอบเข้าที่จุดนี้เพราะ เป็นสัญญาณที่มีความแน่นอนสูงเมื่อมองจาก DTE จะเป็นสัญญาณที่ออกมาจากเครื่องคอมพิวเตอร์โดยตรงแต่ถ้ามองจาก DCE จะเป็นสัญญาณที่ ได้จากการแปลงมาจาก สัญญาณ อะนาล็อก (analog) ส่วนสัญญาณนาฬิกาก็ได้มาจากภายในระบบนั่นเอง ดังนั้นในการสุ่ม (Sampling) ข้อมูลมาตรวจสอบก็สามารถสุ่มได้ กึ่งกลางของข้อมูลซึ่งทำให้เกิดความคลาดเคลื่อนได้ยาก อีกประการหนึ่งก็คือไม่ต้องมีส่วนแปลงสัญญาณ อะนาล็อกเป็นสัญญาณเชิงเลขภายใน เครื่องรวจสอบซึ่งมีความสลับซับซ้อนพอสมควรเพราะว่าจะต้องมีการ Coupling สัญญาณมาจากคู่สายโทรศัพย์และวงจรที่ใช้ภายในเครื่องมือตรวจสอบกับโมเดม (modem) เป็นชุดที่ต่างกันย่อมเกิดความผิดพลาดในการแปลงข้อมูลได้ง่าย

เพื่อที่จะให้เครื่องมือตรวจสอบโพรโทคอลที่สร้างขึ้นนี้มีความสามารถเป็นไปตามข้อที่กำหนดไว้ ในส่วนของการแสดงผล อุปกรณ์ช่วยในการเก็บข้อมูลและพิมพ์ผลออกทางเครื่องพิมพ์ จึงได้ออกแบบให้เครื่องมือตรวจสอบโพรโทคอลที่สร้างขึ้นนี้ทำงานร่วมกับเครื่องไมโครคอมพิวเตอร์ IBM PC/XT จุดประสงค์ที่ออกแบบเช่นนี้เพราะว่า

- ลดฮาร์ดแวร์ในส่วนของการแสดงผล พิมพ์ผล และบันทึกข้อมูล
- สามารถนำเครื่องไมโครคอมพิวเตอร์มาใช้ประโยชน์ได้มากขึ้น
- สถาปัตยกรรมทางด้านฮาร์ดแวร์ของเครื่องไมโครคอมพิวเตอร์รุ่นนี้ได้รับการเปิดเผยพอสมควร
- นำฮาร์ดแวร์บางอย่างที่มีอยู่ในเครื่องไมโครคอมพิวเตอร์มาใช้ประโยชน์ได้
- เครื่องไมโครคอมพิวเตอร์รุ่นนี้เป็นรุ่นที่มีการใช้งานอย่างแพร่หลาย

ดังนั้นเครื่องมือตรวจสอบโพรโทคอลที่สร้างขึ้น จึงได้ถูกออกแบบเป็นการ์ดสำหรับเสียบลงไปบนบอร์ดของเครื่องไมโครคอมพิวเตอร์ โดยจะมีส่วนที่ใช้ในการตรวจสอบโพรโทคอลอยู่บนการ์ด แต่ส่วนที่ใช้ในการแสดงผล พิมพ์ผล และบันทึกข้อมูล ถูกจัดการโดยเครื่องไมโครคอมพิวเตอร์

การออกแบบทางด้านซอฟต์แวร์

โปรแกรมสำหรับควบคุมการรับส่งข้อมูล

เราได้เรียนรู้โครงสร้างต่างๆ ที่จำเป็นสำหรับการสื่อสารข้อมูลของไอบีเอ็มพีซีแล้ว คราวนี้มาลองดูว่าเราจะเขียนโปรแกรมควบคุมการสื่อสารอย่างไรบ้าง โดยไม่กล่าวถึงโปรแกรมสำเร็จรูปที่ใช้ในการสื่อสาร แต่จะกล่าวถึงวิธีการที่จะเขียนโปรแกรมควบคุมการสื่อสาร โดยจะเริ่มตั้งแต่ภาษาระดับสูงไปจนถึงระดับที่เราเข้าไปควบคุมเองทุกอย่าง

ลักษณะของการเขียนโปรแกรมการสื่อสาร

โปรแกรมที่เข้าไปควบคุมการสื่อสารจะต้องรอรับอินพุตอย่างน้อย 2 ทางที่เกิดขึ้นแบบอสมวาร (Asynchronous) คือไม่ทราบแน่ชัดว่าอินพุตทั้งสองจะเข้ามาเมื่อไหร่ อินพุตทั้งสองทางก็คืออินพุตจากคีย์บอร์ดและจากช่องทางการสื่อสาร วิธีการที่จะรอรับอินพุตจากทั้งสองทาง มีอยู่ 2 วิธีคือ Polling และ การใช้อินเตอร์รัพต์ วิธีการทั้งสองมีข้อดีและข้อเสียต่างกัน

วิธีการหยั่งเสียง (Polling) ก็คือการตรวจสอบอินพุตทั้งสองทางตลอดเวลา ถ้ามีอินพุตจากทางใดก็ให้ไปดำเนินการไปทางนั้นเลยให้เสร็จเรียบร้อยแล้วค่อยตรวจสอบอินพุต ในทางอื่นต่อไป วิธีการเช่นนี้ง่ายต่อการเขียนโปรแกรม แต่มีข้อเสียอยู่บ้างตรงที่มีโอกาสที่จะผิดพลาดมีเยอะถ้าหากโปรแกรมกลับเข้าไปตรวจสอบช่องทางสื่อสารไม่ทัน เพราะข้อมูลเข้าจากช่องทางสื่อสารส่งมาตลอดเวลายกตัวอย่างเช่นในขณะที่ซีพียูใช้เวลาในการเขียนข้อมูลลงดิสค์ซึ่งอาจจะใช้เวลา 4-5 วินาที อาจจะมีข้อมูลเข้ามาที่ช่องทางการสื่อสารมากกว่า 2 อักขระแล้วลักษณะเช่นนี้จะทำให้อักขระที่อ่านออกมาไม่ทันหายไป หนทางที่จะแก้ปัญหานี้ก็คือสร้างกฎเกณฑ์ในการติดต่อขึ้นมา (โปรโตคอล) เช่น ให้ฝ่ายที่ต้องการติดต่อส่งอักขระพิเศษ

(Request to Send) มาขอการติดต่อก่อน และรอรับคำตอบว่าจะให้ส่งได้หรือไม่จากฝ่ายตรงกันข้าม เนื่องจากอักขระพิเศษที่ส่งมีอักขระเดียว จึงสามารถค้างอยู่ในรีจิสเตอร์พักข้อมูล RBR ของ 8250 ไปจนกว่าซีพียูจะอ่านข้อมูลออกไป ทางฝ่ายรับเมื่อพบว่าคู่สนทนาต้องการติดต่อด้วยที่ให้ส่งอักขระพิเศษอีกตัวหนึ่ง ตอบกลับไป (Clear to Send) บอกฝ่ายที่เริ่มการติดต่อพร้อมกลับเข้าไปในรูทีนที่ทำหน้าที่ในการรับโดยเฉพาะ ฝ่ายเริ่มการติดต่อหลังจากได้รับสัญญาณ Clear to Send แล้วจึงส่งข้อมูลมาเป็น ชุดถ้าหากเป็นการติดต่อระยะใกล้อาจจะใช้สัญญาณในสาย CTS และ RTSช่วยแทนที่จะใช้อักขระพิเศษ อย่างไรก็ตามการเขียนโปรโตคอลชนิดนี้ต้องคำนึงถึงระยะเวลาที่ต้องคอยคำตอบด้วย ไม่เช่นนั้นฝ่ายหนึ่งอาจจะคอยเก้อ หากฝ่ายตรงข้ามไม่ตอบสนอง

ข้อเสียอีกประการหนึ่งคือ ภาษาที่ใช้ในการเขียนโปรแกรมควบคุมจะต้องมีความเร็วพอที่จะ ตรวจสอบข้อมูลที่ส่งเข้ามาได้ทัน ตัวอย่างเช่น ภาษาเบสิกถ้าหากใช้เขียนให้เข้าไปตรวจสอบช่องทางสื่อสารโดยตรง แล้วจะรับข้อมูลไม่ทันหากความเร็วในการรับส่ง มากกว่า 600 baud

วิธีการเขียนโปรแกรมควบคุมการสื่อสารอีกแบบหนึ่ง คือใช้การขัดจังหวะการทำงานของซีพียู (อินเตอร์รัพต์) เมื่อมีข้อมูลเข้ามาที่ช่องทางการสื่อสารวิธีการเช่นนี้จะทำให้โปรแกรมหลักไม่ต้องพะวงถึง ข้อมูลที่จะเข้ามาทางช่องทางการสื่อสาร เนื่องจากมีโปรแกรมการขัดจังหวะคอยจัดการให้อยู่แล้ว ระบบของไอบีเอ็มพีซีมีระบบการขัดจังหวะสำหรับช่องการสื่อสารไว้พร้อมให้ใช้อยู่แล้วถึงสองช่อง อีกทั้ง 8250 มีระบบการจัดการเกี่ยวกับการขัดจังหวะไว้ให้พร้อมใช้ อย่างไรก็ตามการเขียนโปรแกรมจัดการเกี่ยวกับการขัดจังหวะ (Interrupt Handling Routine) ค่อนข้างจะยุ่งยากและจำเป็นต้องเขียนในรูปแบบของภาษาเครื่องผสมผสานเข้าไปด้วย จึงไม่เหมาะสำหรับผู้เริ่มต้น รายละเอียดของการเขียนอินเตอร์รัพต์จะกล่าวถึงในหัวข้อต่อไป

ภาษาเบสิกมีระบบการจัดการเกี่ยวกับการสื่อสารแบบ การขัดจังหวะให้อยู่แล้วโดยที่เราไม่รู้ตัว ทุกครั้งที่เราเปิดช่องทางการสื่อสารแบบอนุกรม โดยเบสิกจะมีบัฟเฟอร์ สำหรับการสื่อสารไว้ให้ 512 ไบต์ ถึง 32 กิโลไบต์ แล้วแต่เราเลือกเอาว่าจะใช้บัฟเฟอร์การสื่อสารมากน้อยแค่ไหน ในลักษณะเช่นนี้ทุกครั้งที่มีข้อมูลเข้ามาที่ช่องทางการสื่อสาร จะมีการขัดจังหวะดึงเอาข้อมูลไปเก็บไว้ในบัฟเฟอร์โดยอัตโนมัติ ถ้าหากเราไม่อ่านข้อมูลออกจากบัฟเฟอร์ก่อน บัฟเฟอร์เต็มเบสิกจะส่ง Error Massage ออกมาเมื่อเราพยายามอ่านออกจากบัฟเฟอร์ว่า "Buffer overflow" เราสามารถกำหนดให้บัฟเฟอร์มีขนาดใหญ่ได้ถึง 32 K โดยตอนแรกเริ่มเรียกเบสิกมาทำงานให้กำหนดขนาดของบัฟเฟอร์การสื่อสารไว้ด้วย เช่น

```
A > BASIC/C:4096
```

เป็นการบอกว่าเราต้องการบัฟเฟอร์การสื่อสารขนาด 4 กิโลไบต์ ไม่เช่นนั้นเบสิกจะจองพื้นที่สำหรับเป็นบัฟเฟอร์การสื่อสารแค่ 512 ไบต์ ภาษาในระดับสูงอื่นๆ ไม่ได้มีคำสั่งที่จะใช้ช่องการสื่อสารอย่างเต็มที่เหมือนเบสิก ทุกอย่างจะต้องผ่าน Operating System จัดการให้การเข้าไปควบคุมช่องทางสื่อสารจึงจำเป็นต้องเขียนยูทิลิตี้ขึ้นมาใช้เอง

ยูทิลิตี้เกี่ยวกับช่องทางสื่อสารที่ระบบมีให้ใช้

ระดับ Operating System

ดอสมองช่องการสื่อสารเหมือนกับแฟ้ม ๆ หนึ่ง การจัดการทุกอย่างจึงปฏิบัติเสมือนว่าช่องทางการสื่อสารเป็นแฟ้ม ดอสรู้จักช่องทางการสื่อสารภายใต้ชื่อ AUX:, COMO: และ COM1: Command ของดอสที่เกี่ยวกับช่องการสื่อสารมีดังนี้

1) MODE comn:baud,data,parity,stop,p

Mode เป็น external command เป็นคำสั่งที่อยู่ในชื่อแฟ้ม MODE.COM ใช้สำหรับการกำหนดพารามิเตอร์ของอุปกรณ์รอบนอกต่าง ๆ สำหรับดอสในกรณีนี้ใช้ในการตั้งค่าพารามิเตอร์ของช่องทางการสื่อสาร ตัวอย่างเช่นเราต้องการตั้งค่าพารามิเตอร์ของ COM 0 เป็น 9600 band ความยาวของอักขระ 8 บิต, ไม่มีบิตตรวจสอบและบิตหยุดเป็น 1 ดังนี้

```
A >MODE como: 9600,n,8,1,p
```

p ข้างท้ายหมายถึงให้พยายามติดต่อเรื่อย ๆ ไม่มี Time out

2) CTTY evice

เป็นการเปลี่ยน console device จากคีย์บอร์ดและดิสเพลย์มอนิเตอร์ให้ส่งออกไปที่ดีไวซ์แทน เช่น ถ้าเราต้องการจะให้ช่องทางการสื่อสารที่ 1 เข้าไปแทนคีย์บอร์ดและมอนิเตอร์ใช้คำสั่ง

```
A>CTTY COM1:
```

จากนั้นอินพุตและเอาต์พุตทุกอย่างจะผ่านช่องการสื่อสารเท่านั้น คีย์บอร์ดและมอนิเตอร์ไม่ได้ใช้อีกต่อไป

3) การเปลี่ยนทิศทางของอินพุตและเอาต์พุตของดอส

ในดอสเราสามารถเปลี่ยนอินพุตและเอาต์พุตของคำสั่งไปออกที่ดีไวซ์หรือแฟ้มไหนก็ได้โดยให้เครื่องหมาย ">","<",">>" และ "<<"แทนทิศทาง เช่น

```
A >DIR>TEXT
```

แทนที่จะพิมพ์ไดเรคทอรีออกบนหน้าจอก็ไปพิมพ์เก็บไว้ที่แฟ้มชื่อ TEXT

```
A >DIR >> Text
```

ต่างจากตัวอย่างข้างบนตรงที่เอาไปต่อท้ายแฟ้ม TEXT แทนที่จะเขียนทับของเก่า ทำนองเดียวกัน ถ้าเราต้องการจะเปลี่ยนทิศทางของอินพุต/เอาต์พุตให้ออกไปช่องทางการสื่อสารแทนก็ใส่ชื่อ com0: หรือ com1 แทน เช่น

```
A >DIR >COM1:
```

บริการสำหรับการสื่อสารของดอส

ดอสมีบริการสำหรับการสื่อสารให้เรียกใช้เพียง 2 ฟังก์ชันเท่านั้นคือ ฟังก์ชันที่ 3และ4 นอกจากนั้นดอสถือเสมือนว่าช่องการสื่อสารคือแฟ้ม ๆ หนึ่ง ดังนั้นบริการของดอสเกี่ยวกับการอ่านหรือเขียนแฟ้มสามารถนำมาใช้กับช่องทางการสื่อสารได้

ฟังก์ชัน 3 ของดอส (Int 21H,AH =3)
รับอักขระจาก Auxiliary device (หรือช่องการสื่อสารแบบอนุกรม) มาเก็บไว้ในรีจิสเตอร์ AL

ฟังก์ชันที่ 4 ของดอส (Int 21H,AH =4)
ส่งอักขระใน AL ออกไปที่ Auxiliary device

บริการสำหรับการสื่อสารของไบออส (BIOS)

ไบออสมีบริการเกี่ยวกับการสื่อสารให้เลือกใช้คืออินเตอร์รัพต์ที่ 14H โดยมีฟังก์ชันให้ 3 อัน คือ

AH = 0 สำหรับตั้งค่าพารามิเตอร์

AH = 1 ส่งอักขระ AL ออกไปที่ช่องทางออกอนุกรม

AH = 2 รับอักขระจากพอร์ตอนุกรมเข้ามาที่ AL

การพัฒนายูทิลิตี้ สำหรับการสื่อสารข้อมูล

ในภาษาระดับสูงหลายภาษาที่ไม่มีคำสั่งเกี่ยวกับการควบคุมช่องทางสื่อสารให้ หากเราจำเป็นต้องใช้คอมพิวเตอร์ให้ติดต่อกับโลกภายนอกทางช่องทางการสื่อสาร จำเป็นที่เราจะต้องพัฒนายูทิลิตี้ขึ้นมาใช้เอง

ฮาร์ดแวร์อินเตอร์รัพต์

ไอบีเอ็มพีซีมีสัญญาณการขัดจังหวะที่สามารถเข้าไปขัดจังหวะการทำงานของซีพียูได้ 8 ช่องทางโดยใช้ไอซี 8259 สำหรับควบคุมการขัดจังหวะไอบีเอ็มพีซีได้กำหนดการใช้งานของการขัดจังหวะจากฮาร์ดแวร์ทั้ง 8 ไว้ดังนี้

IRQ0 สำหรับการขัดจังหวะของ Timer

IRQ1 สำหรับการขัดจังหวะของคีย์บอร์ด

IRQ2 สำรอง

IRQ3 สำหรับช่องการสื่อสารหมายเลข 1

IRQ4 สำหรับช่องการสื่อสารหมายเลข 0

IRQ5 สำหรับดิสเกตต์คอนโทรลเลอร์

IRQ6 สำหรับดิสค์คอนโทรลเลอร์

IRQ7 สำรอง

สัญญาณจากการขัดจังหวะทั้ง 8 จะทำให้เกิดอินเตอร์รัพต์ที่ 8 บวกหมายเลขของ IRQ ดังนั้น การขัดจังหวะจาก com0 ซึ่งผ่านทาง IRQ4 จึงทำให้เกิดทิศทางการขัดจังหวะที่ 0CH หรือที่ 13 ในเลขฐาน 10

ขั้นตอนของการยอมให้สัญญาณการขัดจังหวะผ่านเข้าไปถึงซีพียู

1. เมื่อมีเหตุการณ์เกิดขึ้นที่ 8250 ตามที่เรากำหนดเหตุการณ์ที่จะทำให้เกิดสัญญาณอินเตอร์รัพต์ที่พอร์ต XF9 เช่น ตั้งค่าไว้ที่ XF9 เป็น 01 สัญญาณ Instr จะออกมาจาก 8250 เมื่อ 8250 รับข้อมูลครบ 1 อักขระ

2. สัญญาณ Instr ที่ออกมาจาก 8250 ยังถูกควบคุมด้วยเกตตัวหนึ่ง ซึ่งสามารถควบคุมได้โดยพอร์ต XFC บิตที่ 3 (สัญญาณ OUT 2)

3. เมื่อ 8259 รับสัญญาณ IRQ4 จะตรวจสอบดู IMR หรือ Interrupt Mask Register ว่าจะรับหรือไม่ 8259 จะรับสัญญาณ IRQ ต่อเมื่อบิตที่ตรงกับหมายเลข IRQ ใน IMR รีจิสเตอร์เป็น 0 เท่านั้น เราเข้าไปควบคุม IMR ได้โดยช่องทางเข้าออกหมายเลข 21H

4. ถ้า 8259 รับสัญญาณ IRQ 8259 จะตรวจสอบลำดับความสำคัญของสัญญาณ IRQ ว่าขณะ นี้ซีพียูกำลังทำงานให้กับอินเตอร์รัพต์ไหนบ้าง โดยตรวจสอบดูที่ รีจิสเตอร์ ISR หรือ Interrupt Service Register ถ้าซีพียูไม่ได้ให้บริการอินเตอร์รัพต์ไหนที่สำคัญกว่า 8259 จะส่งสัญญาณอินเตอร์รัพต์ต่อไปยัง 8088 ผ่านทาง Maskable Interrupt (MI)

5. ทุก ๆ คำสั่งของ 8088 หลังจากปฏิบัติแล้ว จะตรวจดูสัญญาณอินเตอร์รัพต์ที่ขา NMI (Non Maskable Interrupt ซึ่งไอบีเอ็มพีซีต่อเอาไว้กับการตรวจความผิดพลาดของหน่วยความจำ) และ MI สำหรับสัญญาณ MI จะมีผลให้ 8088 ให้ความสนใจต่อเมื่ออินเตอร์รัพต์แฟลกเป็น 1

6. เมื่อ 8088 รับสัญญาณ MI จะส่งสัญญาณไปบอก 8259 ว่า ตกลงรับการร้องขออินเตอร์รัพต์ (Instr.Ack) กลับไปยัง 8259

7. 8259 จะเอาหมายเลขทิศทางการขัดจังหวะ(OCH ในที่นี้)ส่งให้ 8088 ผ่านทางดาต้าบัส พร้อมกับตั้งค่า ISR บิตที่ตรงกับหมายเลขสัญญาณ IRQ ให้เป็น 1 เพื่อเป็นการเตือนตัวเองว่าขณะ นี้ซีพียูกำลังให้บริการอินเตอร์รัพต์ไหนบ้าง

8. 8088 รับเอาหมายเลขทิศทางการขัดจังหวะจากดาต้าบัสแล้วจะเก็บค่าภาวะแฟลกและเซกมนต์ของโปรแกรมและโปรแกรมเคาน์เตอร์ขณะนั้นลงในสแต็กพร้อมกับกระโดดไปยังอินเตอร์รัพต์รูทีนที่ชี้บอกโดย Interrupt Vector Table (ในหน่วยความจำ ตำแหน่ง ที่ 0000:0030H ในที่นี้) พร้อมกับเคลียร์ IF flag ให้เป็น 0ไม่ยอมให้เกิด Maskable Interrupt อีกต่อไป

9. เป็นหน้าที่ของอินเตอร์รัพต์รูทีนที่จะจัดต่อไป

หมายเหตุ สัญญาณ Instr ของ 8250 จะหายไปเมื่อมีการอ่านรีจีสใน 8250 ตามตาราง

เทคนิคการเขียนอินเตอร์รัพต์รูทีนสำหรับการสื่อสาร

1) ตั้งค่าพารามิเตอร์ต่าง ๆ สำหรับการสื่อสาร

2) กำหนดชนิดของการเกิดสัญญาณอินเตอร์รัพต์ของการสื่อสารที่จะเข้ามา (ทิศทางที่ OCH สำหรับ IRQ4 หรือ com0: และ OBH สำหรับ IRQ3 หรือ com1:)

3) กำหนด Vector table สำหรับการอินเตอร์รัพต์ของการสื่อสารที่จะเข้ามา (ทิศทางที่ OCH สำหรับ IRQ4 หรือ com0: และ OBH สำหรับ IRQ3 หรือ com1:)

4) ตั้งค่า IMR ใน 8259 โดยผ่านทางพอร์ต 21H

5) เปิดอินเตอร์รัพต์เกตใน 8250 โดยให้บิตที่ 3 ของ MCR เป็น 1

6) ต้องไม่ลืมนำอินเตอร์รรัพต์เวคเตอร์เก่ากลับคืนเมื่อเราเอาอินเตอร์รัพต์รูทีนออกจากหน่วยความจำหรือเลิกใช้โปรแกรมนี้แล้ว

ตัวรูททีนที่ให้บริการสำหรับการขัดจังหวะสำหรับช่องทางสื่อสารควรจะมีลักษณะดังนี้

1) เซตค่า IF flag ให้เป็น 1 กลับคืนถ้าหากต้องการให้เกิด Maskable interrupt ตัวอื่นที่มีความสำคัญกว่า

2) เก็บค่ารีจิสเตอร์ต่าง ๆ ที่ต้องการใช้ลงในสแต็กให้หมด

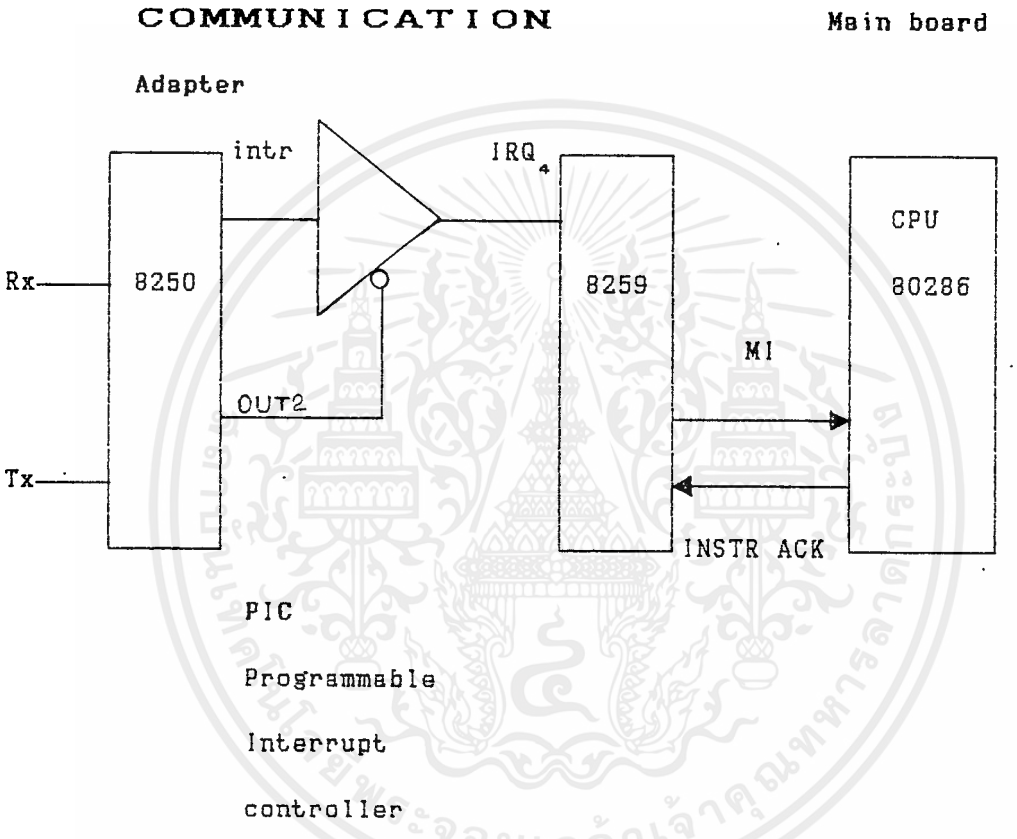
3) อ่านดูรีจิสเตอร์ IIR ใน 8250 ว่าเป็นการเกิดอินเตอร์รัพต์ด้วยเหตุใดในกรณีที่เราตั้งค่าให้เกิดอินเตอร์รัพต์หลายชนิด

4) อ่านอินพุตมาจาก 8250 และจัดการตามขบวนการ

5) เอาค่ารีจิสเตอร์ต่าง ๆ ที่เก็บไว้ในสแต็กกลับคืน

6) ส่งเอาต์พุตไปบอก 8259 ว่าสิ้นสุดการทำงานอินเตอร์รัพต์ที่ขอมาแล้วโดยส่งไปที่พอร์ต 20H ด้วยค่า 20H บวกหมายเลข IRQ ที่ 8259 รับเข้ามา เช่น ถ้าเป็น IRQ4 ก็ให้ส่งค่า 24H ออกไปที่พอร์ต 20H

7) IRET กระโดดกลับไปโปรแกรมหลัก



PORT	PORT	I Flag
XF9 - กำหนดชนิดของการเกิดอินเตอร์รัพต์ XFC - เปิด OUT2 ให้สัญญาณ Instr ส่งต่อไป 8259	21H interrupt Mask ยอมให้สัญญาณ IRQ4 ส่งต่อไป 80286 ถ้าเซตบิตที่ 4เป็น 0	Interrupt flage สำหรับรับหรือไม่รับสัญญาณอินเตอร์รัพต์คำสั่งในการเซตและรีเซต flag คือ STI และ CLI

รูป ขั้นตอนการเดินทางของสัญญาณการขัดจังหวะจากช่องทางสื่อสารอนุกรม

สาเหตุของการเกิดข้อผิดพลาดในการรับ เกิดขึ้นได้ 3 แบบ

1. เกิดการเขียนทับข้อมูลเก่าคืออ่านข้อมูลออกจาก RBR (หรือรีจิสเตอร์พักข้อมูล) ไม่ทันที่ของใหม่เขียนทับเข้ามา ข้อมูลเก่าจะหายไปเมื่อเกิดเหตุการณ์เช่นนี้ สถานภาพของการรับส่งใน LSR รีจิสเตอร์จะเป็นตัวบอกพร้อมกับเกิดสัญญาณอินเตอร์รัพต์ไปบอกซีพียู ถ้าเรายอมให้เกิดอินเตอร์รัพต์ชนิดนี้ (ซึ่งกำหนดได้โดยรีจิสเตอร์ LCR)

2. เกิดการตรวจสอบผิดพบาดในการส่งข้อมูลเราสามารถแทรกบิตตรวจสอบเอาไว้ทุกอักขระที่ส่ง ถ้าหากฝ่ายรับตรวจสอบแล้วไม่ตรงตามที่ส่งมาฝ่ายรับจะถือว่าเกิดการผิดพลาด LSR รีจิสเตอร์จะเป็นผู้บ่งชี้ และเกิดสัญญาณอินเตอร์รัพต์ทำนองเดียวกับการผิดพลาดในแบบที่ 1

3. เมื่อเกิดการผิดพลาดในการหาขอบเขตของอักขระเรียกว่า Faming Error คือหาบิตเริ่มและบิตหยุดไม่พบ

การควบคุมพารามิเตอร์ของการรับส่งสามารถทำได้โดยการกำหนดค่าลงไป LCR รีจิสเตอร์ ซึ่งอยู่ที่หมายเลขช่องทาง XFB โดยค่าใน XFB มีความหมายดังในรูป

นอกเหนือจากนั้น 8250 ยังมีรีจิสเตอร์ที่นำไปใช้ในการควบคุมโมเด็มอีกด้วยโดยรีจิสเตอร์ MCR ที่หมายเลขช่องทางเข้าออกที่ XFC ซึ่งมีความหมายการควบคุม

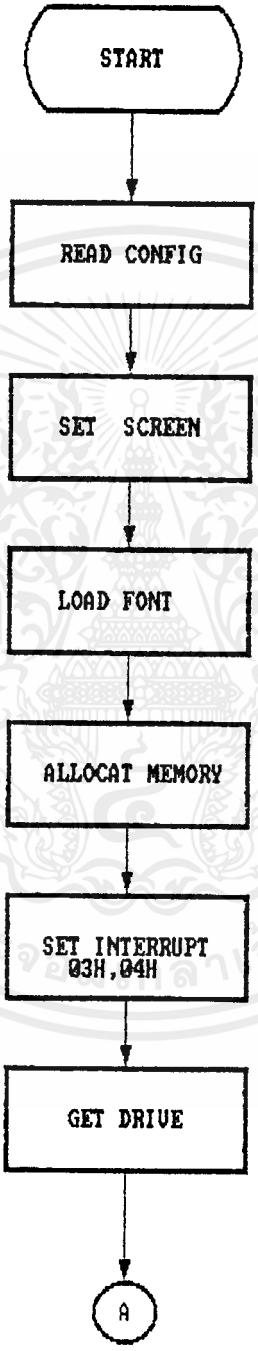
การทำงานของโปรแกรม
CAT PROTOCOL ANALYZER

โปรแกรม CAT นี้จะประกอบไปด้วยไฟล์ต่างๆ 8 ไฟล์ด้วนกันมีดังนี้

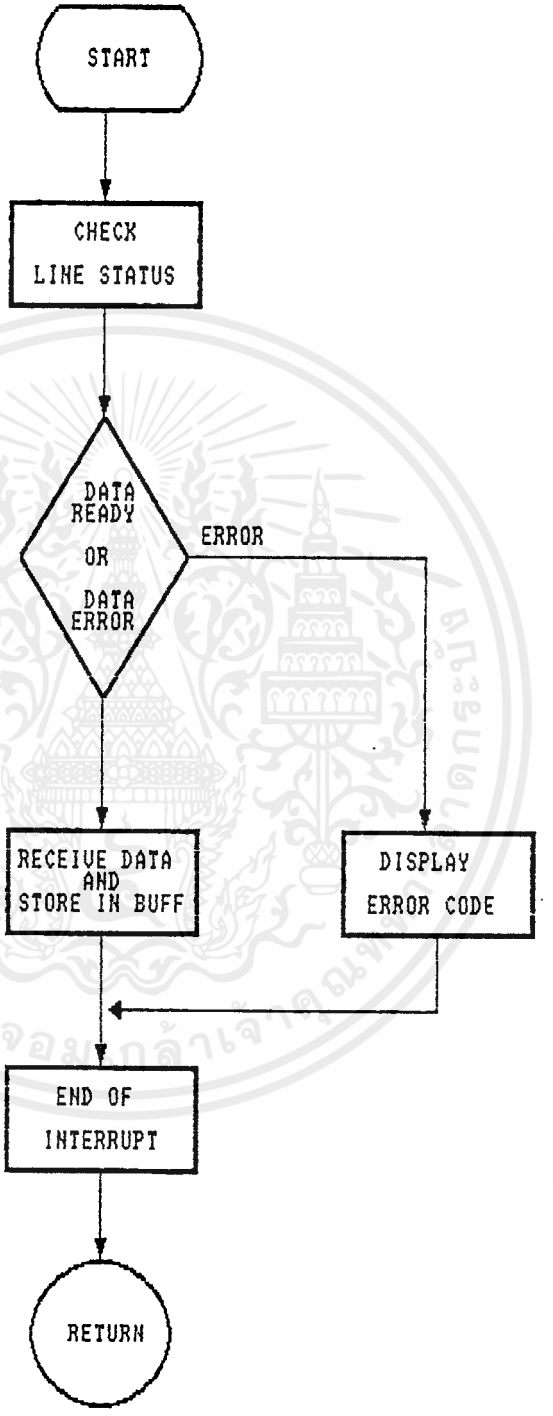
1. CAT.ASM เป็นโปรแกรมหลักในการเรียกใช้โปรแกรมย่อยต่างๆ
2. PTA.ASM โปรแกรมนี้จะทำหน้าที่รับข้อมูลจาก COM1 และ COM2 มาทำการวิเคราะห์ปัญหาต่างๆที่เกิดขึ้นในการรับ-ส่ง
3. VIDEO_IO.ASM ไฟล์นี้จะเก็บ routine ทางด้าน display และ key
4. TERMINAL.ASM เป็นโปรแกรมที่ทำงานทางด้าน Terminal
5. FILECOM.ASM เป็นโปรแกรมที่ใช้ในการรับและส่งข้อมูลเป็นไฟล์
6. SHELL.ASM โปรแกรมนี้จะทำการเรียกใช้งาน COMMAND.COM
7. CATPRN.ASM เป็นโปรแกรมที่ใช้ในการพิมพ์ข้อมูลที่อยู่ใน DISk ออกไปยังเครื่อง Printer
8. CAT.LIB เป็นไฟล์ที่ใช้เก็บ Libraries

ในแต่ละ File ยังประกอบไปด้วย Routine อื่นๆอีกมากมาย ท่านสามารถที่จะดูได้ในตัวโปรแกรมในภาคผนวกท้ายเล่ม ส่วนการทำงานของโปรแกรมต่างๆ ในแต่ละส่วนดูได้จาก Flower chart ต่อไปนี้

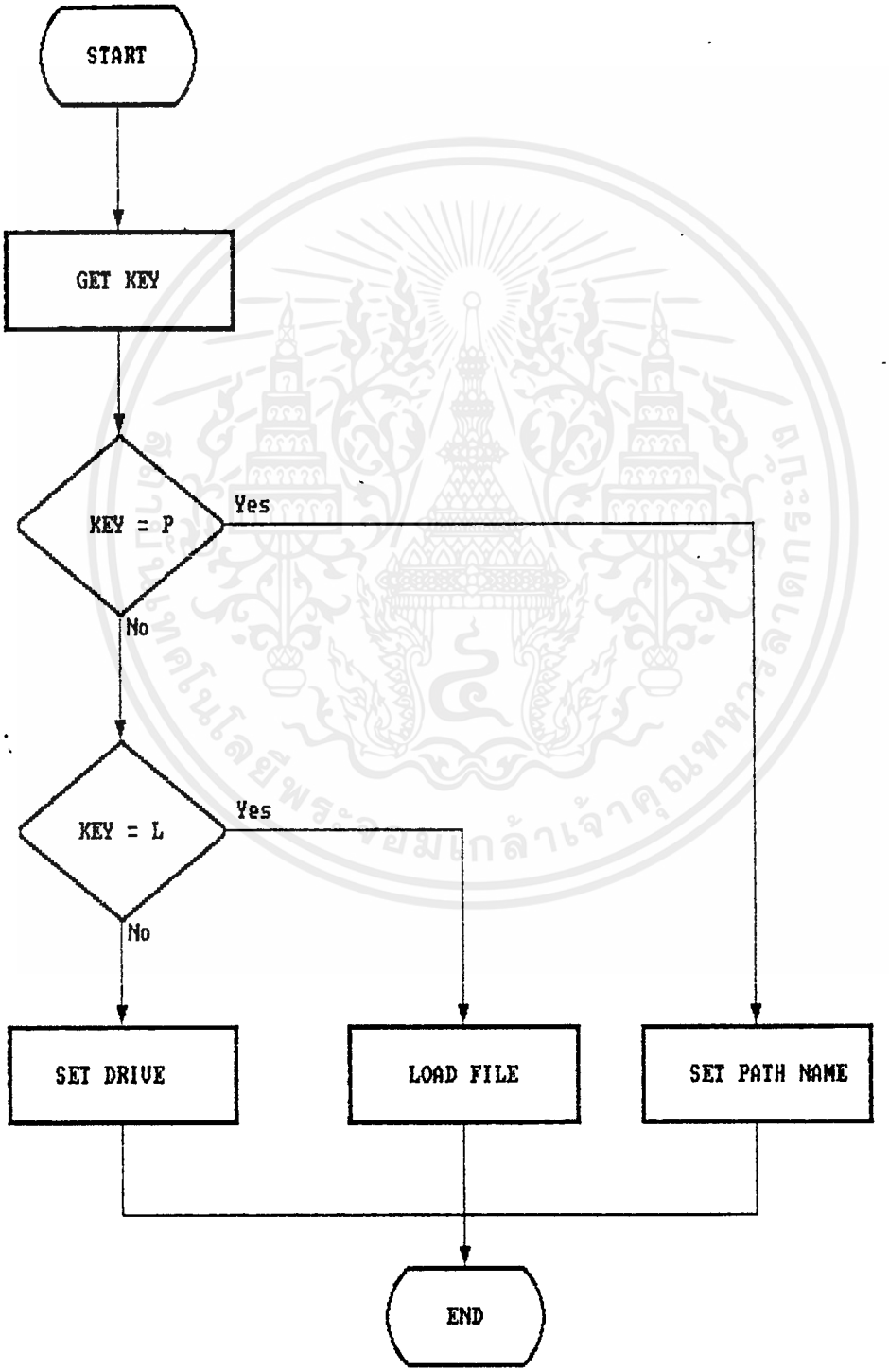
MAIN PROGRAM



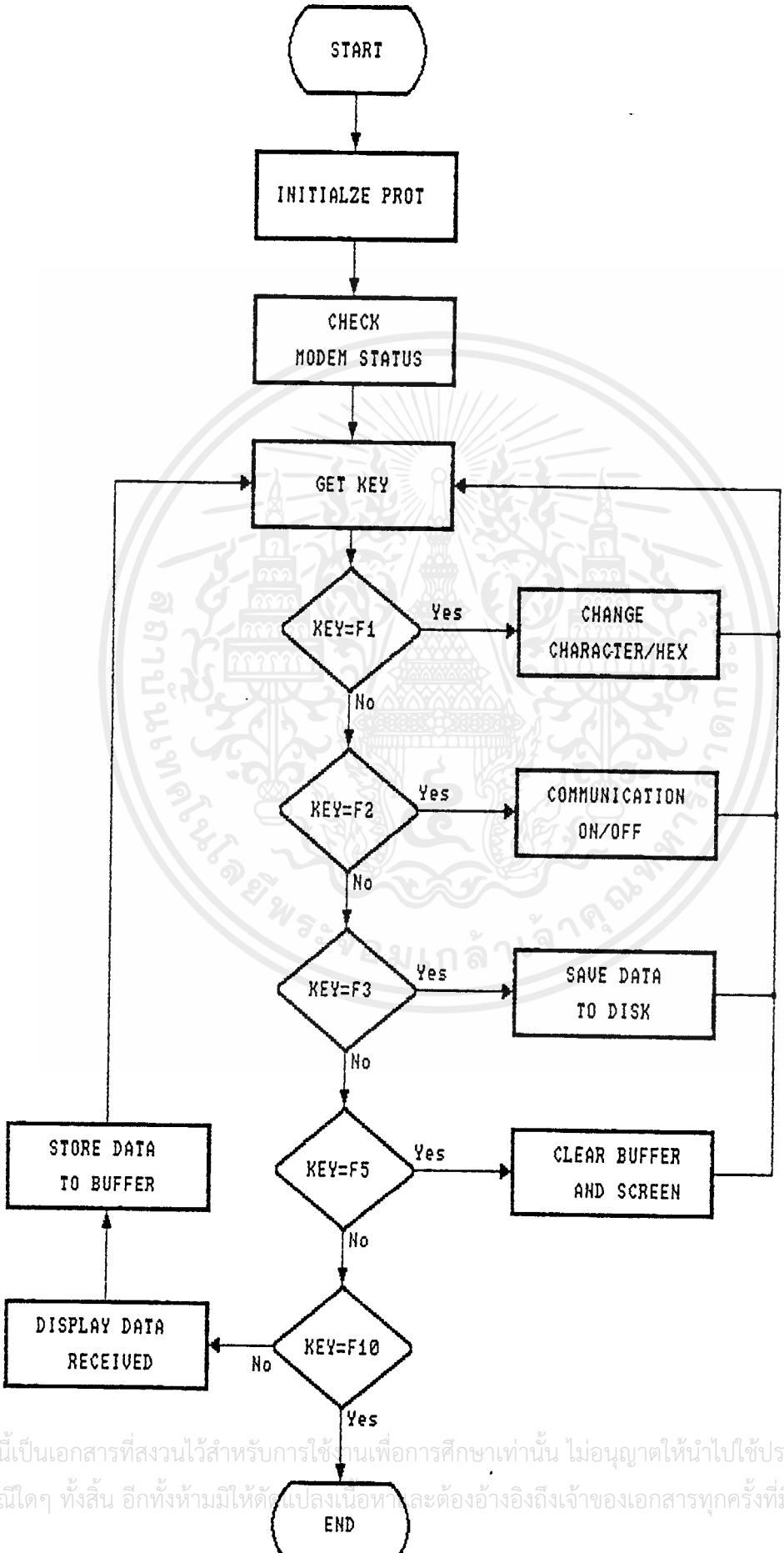
INTERRUPT SERVICE ROUTINE



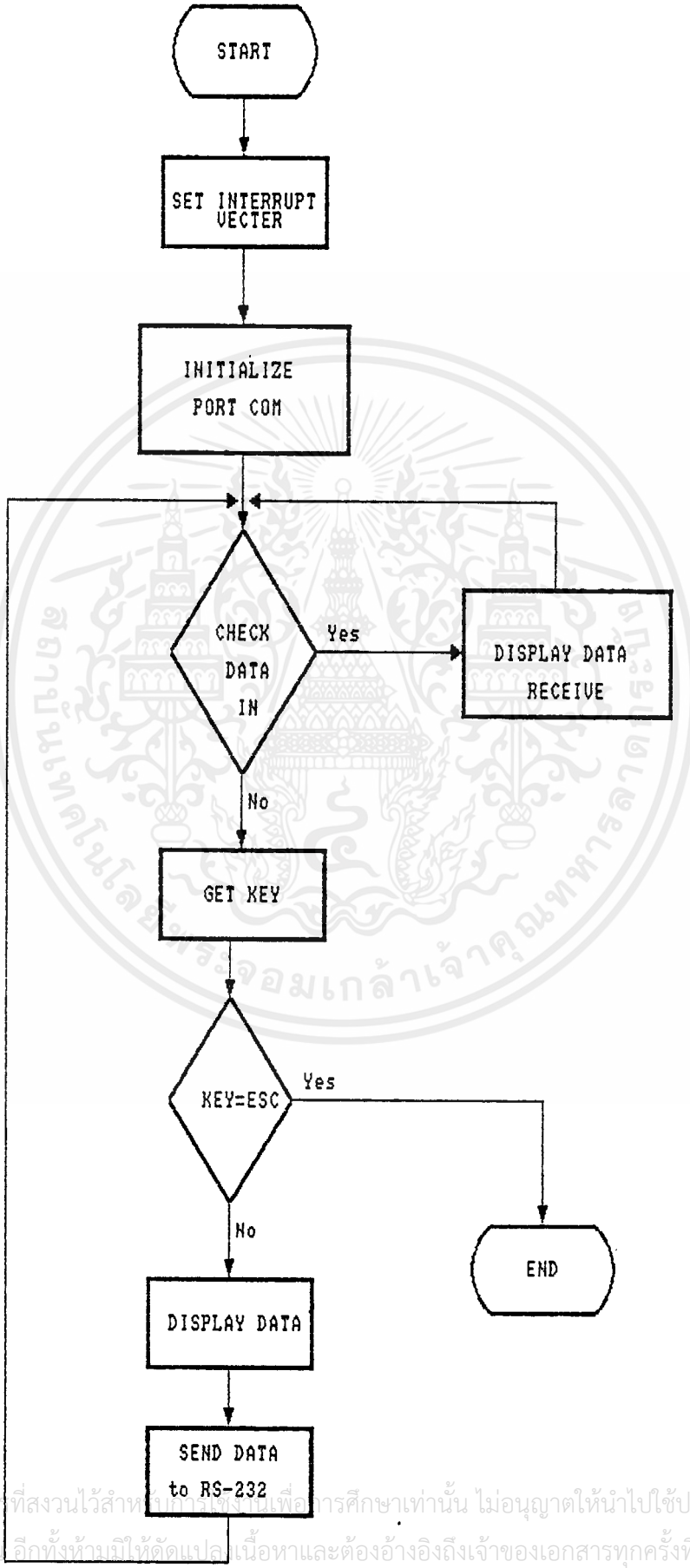
F I L E



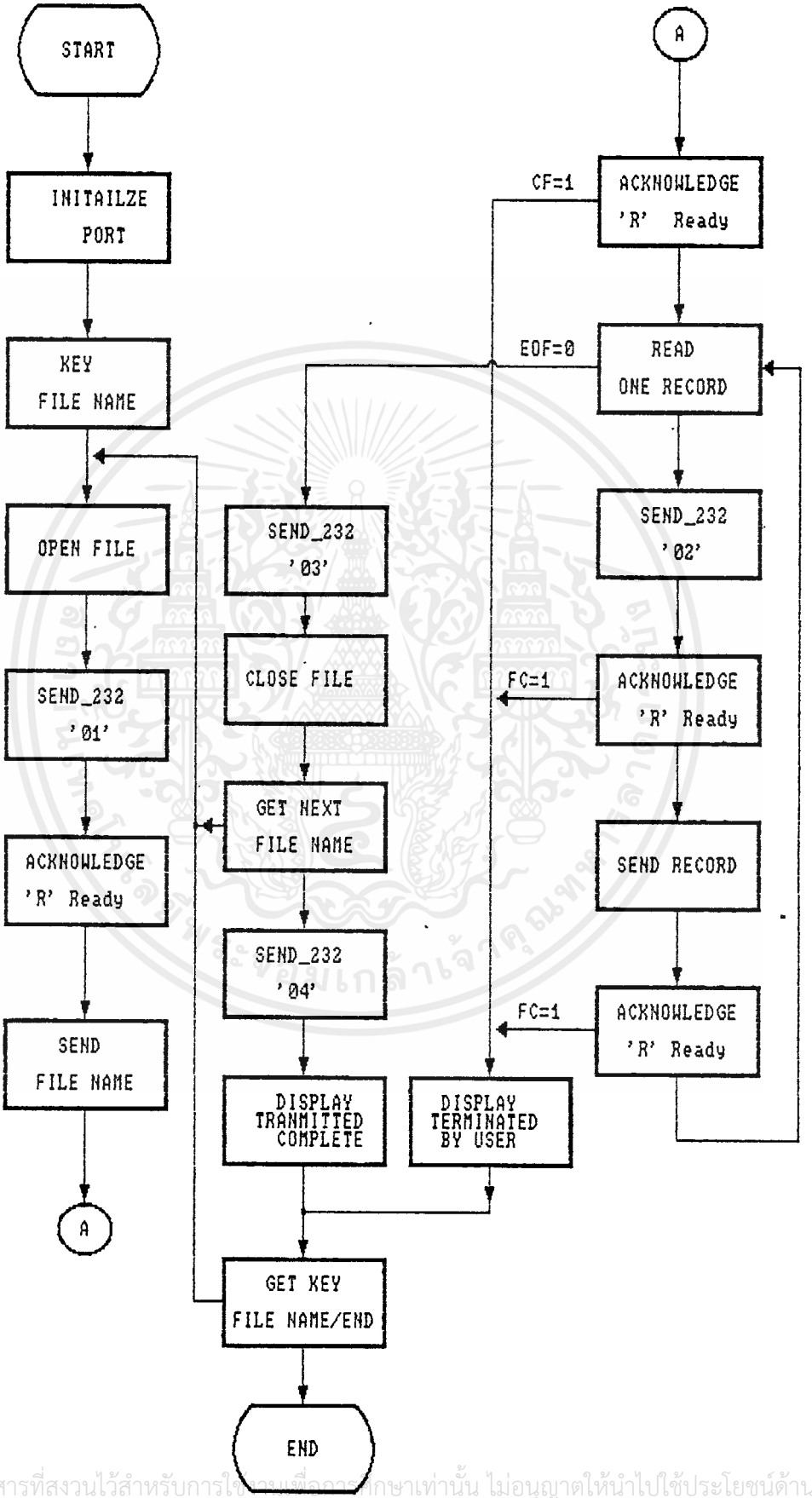
COMMUNICATION TEST



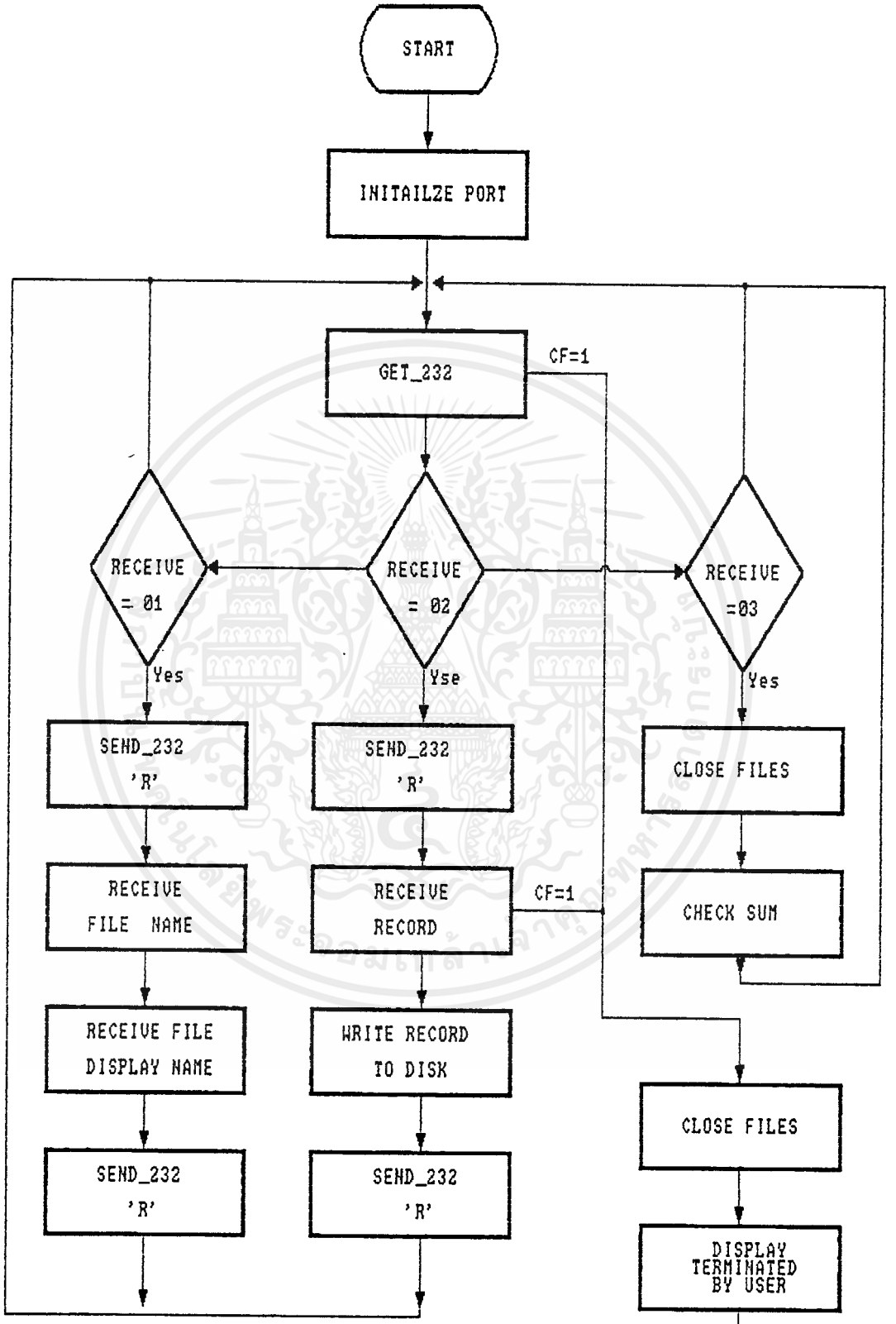
TERMINAL EMULATOR



SAND FILE



RECEIVE FILES



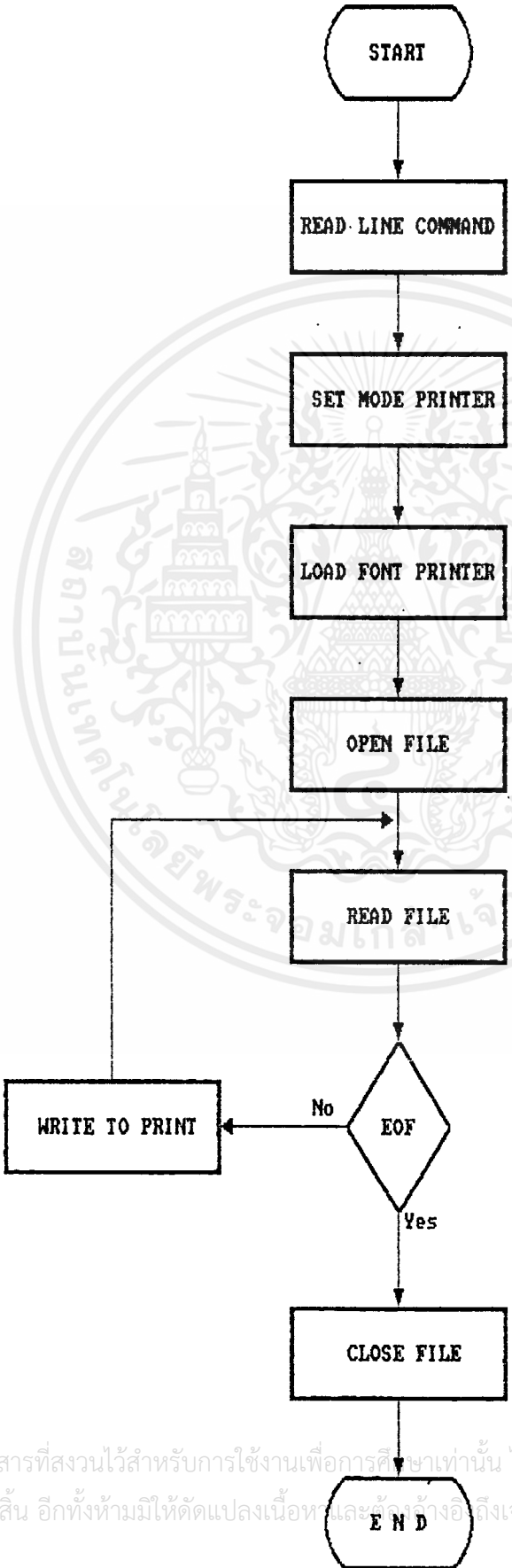
01 = receive path or filename and openfile

02 = receive a 128 characters through the RS-232 line
send R to acknowledge and wait for new code

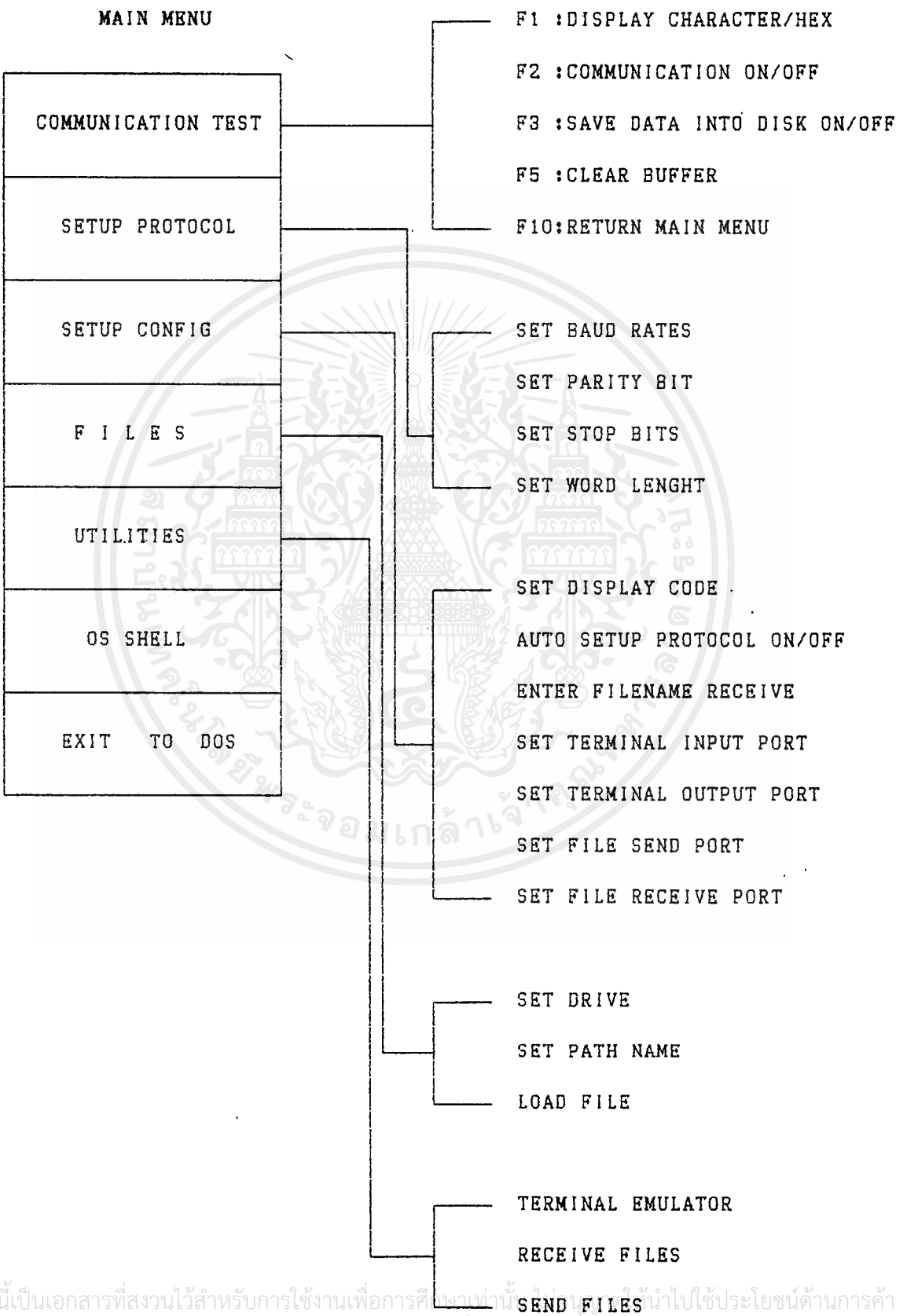
03 = close file,post message and repeat receive

04 = transfer complete

CATPRN.COM



ผังคำสั่งโปรแกรม



สรุปงานที่ทำ

เครื่องมือตรวจสอบโพรโทคอลที่สร้างนี้มีคุณสมบัติในการทำงานดังนี้คือ

โพรโทคอลที่สามารถตรวจสอบได้	- Asynchronous
อัตราเร็วของข้อมูลที่ตรวจสอบได้	- สูงสุด 9,600 บิดต่อวินาที
รหัสข้อมูล	- ASCII
หน่วยความจำที่ใช้ในการเก็บข้อมูล	- 32+32 กิโลไบต์
อุปกรณ์ช่วยในการเก็บข้อมูล	- Diskette
สัญญาณที่ใช้ในการตรวจสอบ	- TX,RX,CTS,DSR,RI,CD และ
	- GND ตามมาตรฐาน RS-232C
รหัสที่ใช้ในการตรวจสอบความผิดพลาด	- CCITT
จอแสดงผล	- จอแสดงผลของเครื่องไมโคร-
	- คอมพิวเตอร์และเครื่องพิมพ์
ความสามารถพิเศษ	- ใช้เป็น Terminal
	- ใช้ในการรับส่งข้อมูลลักษณะ File
โมดของการรับส่งข้อมูลที่ตรวจสอบๆได้	- ฮาล์ฟดูเพลกซ์และฟูลดูเพลกซ์

DEFINITIONS OF CONTROL CHARACTERS

NUL	NULL
SOH	START OF HEADING
STX	START TEXT
ETX	END TEXT
EOT	END OF TRANSMISSION
ENQ	ENQUIRY
ACK	ACKNOWLEDGE
BEL	BELL
BS	BACKSPACE
HT	HORIZONTAL TAB
LF	LINE FEED
VT	VERTICAL TAB
FF	FORM FEED
CR	CARRIAGE RETURN
SO	SHIFT OUT
SI	SHIFT IN
DLE	DATA LINK ESCAPE
DC1	DIRECT CONTROL 1
DC2	DIRECT CONTROL 2
DC3	DIRECT CONTROL 3
DC4	DIRECT CONTROL 4
NAK	NEGATIVE ACKNOWLEDGE

SYN	SYNCHRONOUSIDLE
ETB	END TRANSMISSION BLOCK
CAN	CANCEL
EM	END OF MEDIUM
SUB	SUBSTITUTE
ESC	ESCAPE
FS	FORM SEPARATOR
GS	GROUP SEPARATOR
RS	RECORD SEPARATOR
US	UNIT SEPARATOR

****

คู่มือการใช้ Program

C A T

PROTOCOL ANALYZER

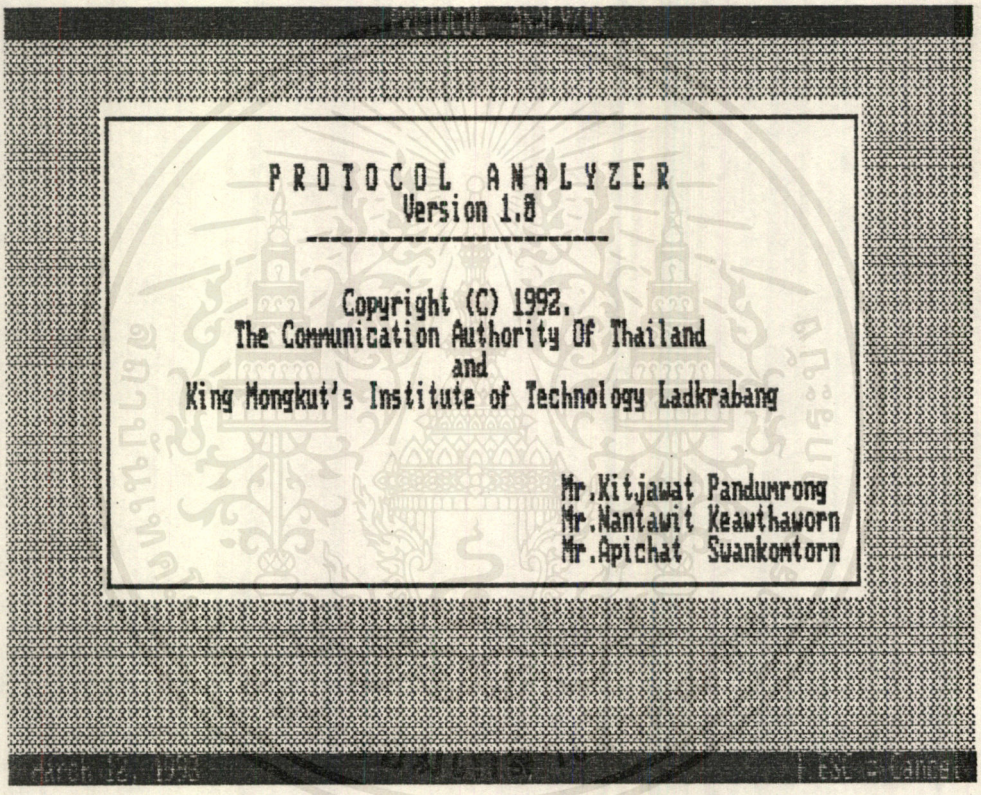
เนื่องจากในปัจจุบันระบบสื่อสารข้อมูลมีความเจริญขึ้นมาก อุปกรณ์ที่ใช้ในระบบรวมทั้งตัวระบบเองก็มีความสลับซับซ้อนมากขึ้นด้วย และในปัจจุบันนี้มีโปรแกรมที่ใช้งานในด้านการติดต่อสื่อสารก็มีมากด้วย ดังนั้นรหัสที่ใช้ในการโต้ตอบอาจจะไม่เหมือนกันจะเห็นได้ว่าปัญหาจะเริ่มเกิดขึ้น สิ่งที่ตามมาก็คือผู้ใช้อุปกรณ์หรือโปรแกรมเหล่านั้นไม่สามารถบอกได้ว่าปัญหาที่เกิดขึ้นนั้นเกิดขึ้นเนื่องจากตัวอุปกรณ์หรือตัวโปรแกรม เพราะผู้ใช้ไม่สามารถมองเห็นข้อมูลที่มีการติดต่อรับส่งอยู่ขณะนั้นได้ แต่เมื่อท่านนำ Protocol Analyzer ไปติดตั้งกับระบบของท่าน ท่านก็สามารถที่อ่านข้อมูลมาทำการวิเคราะห์ได้ว่าปัญหาที่เกิดขึ้นนั้น เนื่องมาจากอะไร และในโปรแกรม CAT PROTOCOL ANALYZER นี้ยังมีความสามารถที่จะให้ท่านใช้ทำงานเป็น TERMINAL EMULATOR และใช้ในการรับ หรือส่งข้อมูลเป็นไฟล์ ๆ ที่เราเก็บไว้ในแผ่น Disk

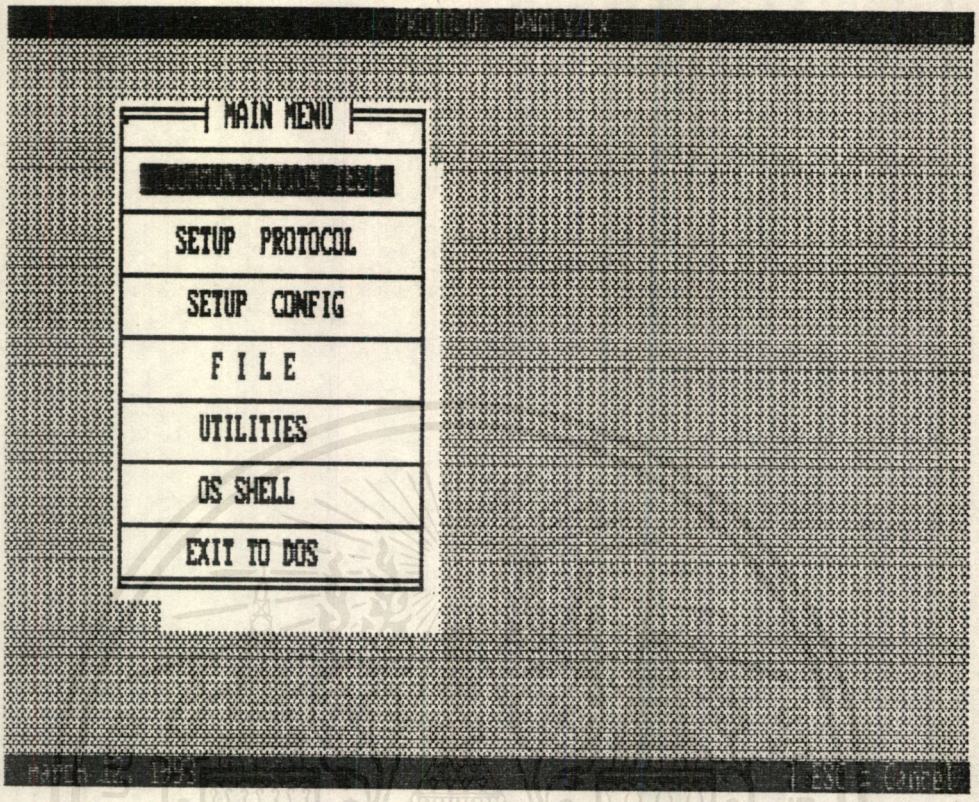
เครื่องไมโครคอมพิวเตอร์ที่สามารถใช้งานโปรแกรม CAT นี้ได้จะต้องประกอบด้วย

- ไมโครคอมพิวเตอร์ รุ่น AT CPU 286,386,486
- ดิสก์ไดรฟ์ 360 KB หรือ 1.2 MB
- ควรเป็น DOS เวอร์ชัน 3.0 ขึ้นไป
- จอภาพแบบ EGA,VGA หรือ SVGA เท่านั้น
- เครื่องพิมพ์ EPSON LX-850

เริ่มต้นการใช้งาน

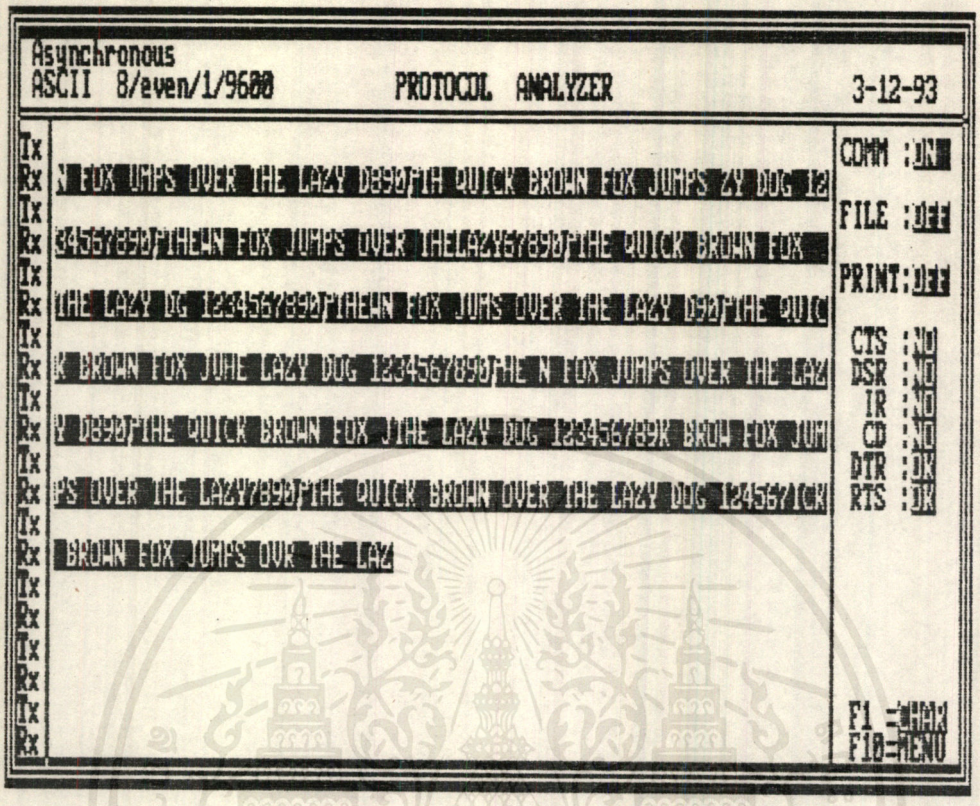
A:> CAT





- COMMUNICATION TEST

- ใช้ตรวจสอบการทำงานระหว่างอุปกรณ์ข้อมูลปลายทางและอุปกรณ์สื่อสารข้อมูล
- ใช้ตรวจสอบคำสั่งและผลตอบรับ ในการรับส่งข้อมูล
- แสดงผลความผิดพลาดที่เกิดขึ้นในการรับส่งข้อมูล
- สำหรับส่วนรหัส CONTROL CHARACTERS จะแสดงเป็นตัวอักษรที่เห็นแล้วเข้าใจได้ง่ายตัวอักษรที่แสดงว่าเป็น CONTROL CHARACTERS ตัวอักษรจะซ้อนกันอยู่



F1 = Character / Hex

F2 = Com on / off

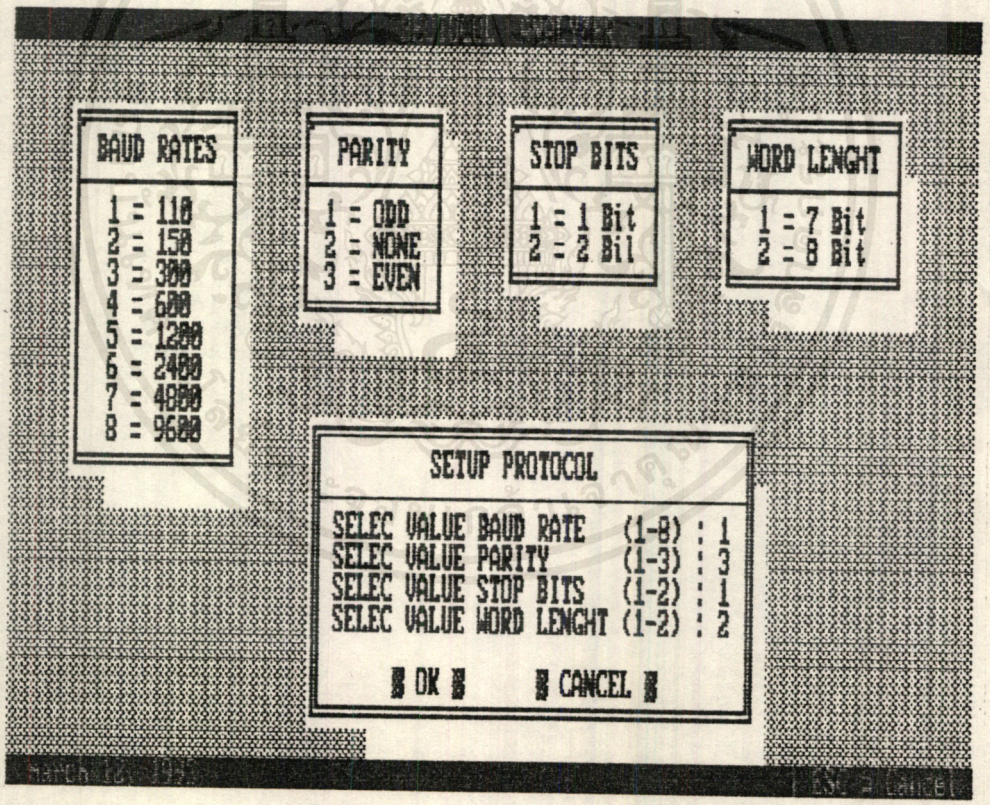
F3 = Save data to disk

F5 = Clear data buffer and screen

F10= Exit to main menu

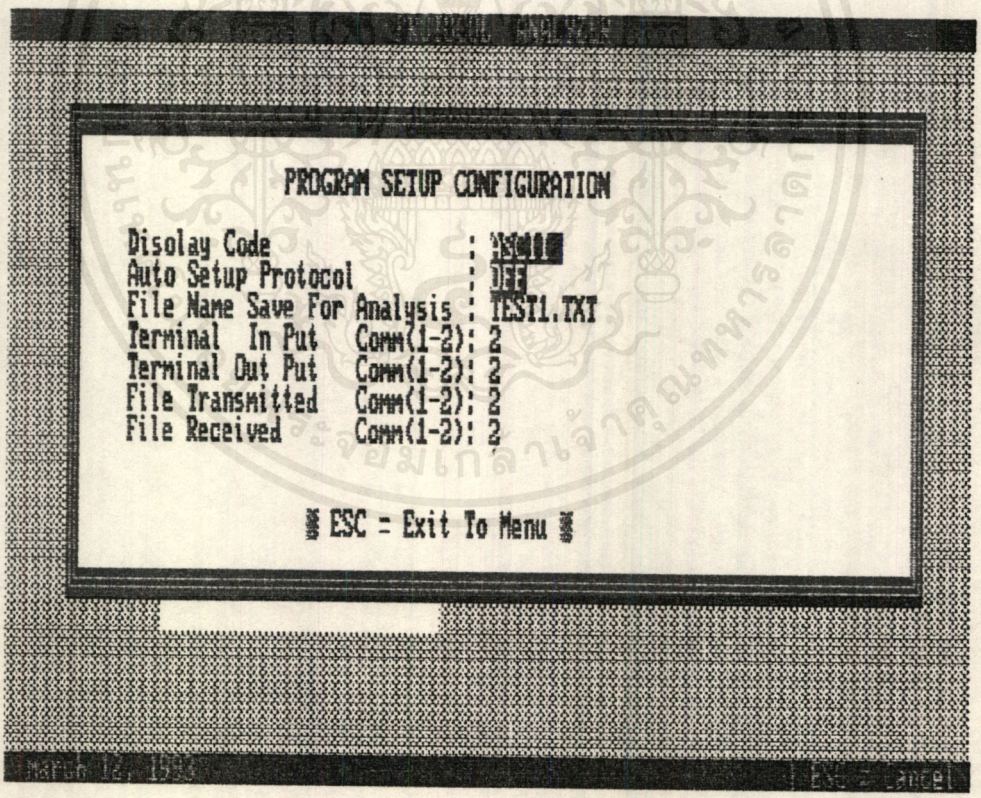
- SETUP PROTOCOL

เป็นการ set ค่า Baud Rate,Parity,Stop Bits และ Word lenght ให้ตรงกับ Protocol อีกฝั่งหนึ่งที่เราต้องการติดต่อด้วย กระทำได้โดยใช้ลูกศรเลื่อนตำแหน่งของ cursor ไปยังจุดที่จะเปลี่ยนแปลงแล้วกดตัวเลขได้ตามที่เราต้องการ เมื่อท่านกำหนดค่าทุกอย่างครบแล้วให้เลื่อนมาที่ "OK" แล้วกด Enter ทุกอย่างก็จะเปลี่ยแปลงใหม่หมด



- SETUP CONFIG

ใช้ในการกำหนดค่าให้โปรแกรมทำงานติดต่อกับ port com1 หรือ com2 ทำการเปลี่ยนแปลงค่าได้โดยให้ผู้ใช้เลื่อนเคอร์เซอร์ไปยังตำแหน่งที่จะเปลี่ยนแปลงแล้วให้กด Enter ค่านั้นก็จะเปลี่ยนแปลงไปและใช้ในการกำหนดชื่อไฟล์ที่จะทำการ Save date ในตอนที่เราทำการเลือก Mode Communication test เพื่อที่จะนำเอาข้อมูลนั้นมาทำการวิเคราะห์ต่อไป เมื่อต้องการที่จะออกจาก Menu นี้ ให้กด Key ESC จะกลับไปที่ Main Menu

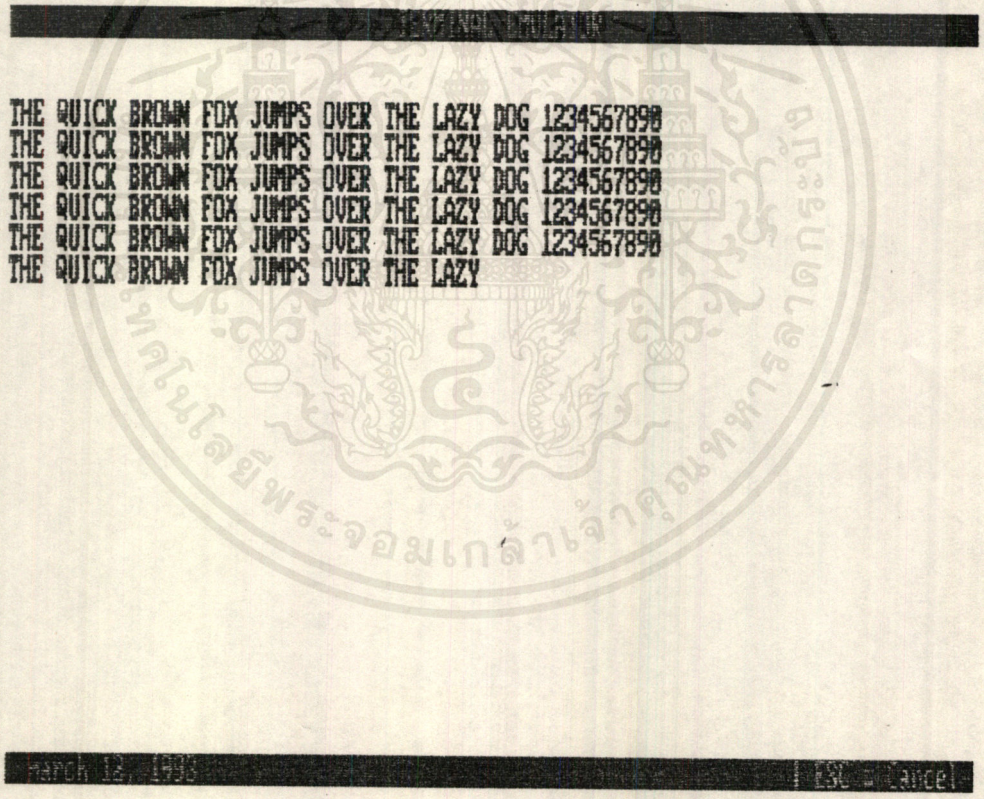


- **FILE**

ใช้ในการกำหนดหรือเปลี่ยนแปลงค่า ของ DISK DRIVE หรือ PATH และใช้ในการ LOAD FILE ที่เราได้ทำการ SAVE ไว้ในตอน COMMUNICATION TEST กลับมาดูที่หน้าจอใหม่อีกครั้งหนึ่ง เมื่อต้องการที่จะออกจาก Menu นี้ ให้กด Key ESC จะกลับไปที่ Main Menu

- UTILITIES

ใน MENU นี้จะมีให้เราทำการเลือกใช้ ให้เครื่องทำงานในลักษณะของ TERMINAL หรือ จะให้ใช้ในการรับ-ส่ง FILE กันระหว่างเครื่อง COMPUTER ด้วยกันและ ถ้าต้องการที่จะใช้ COM1 หรือ COM2 ให้ไปทำการกำหนดใน MENU SETUP CONFIG ใน Mode Terminal Emulator ถ้ากด F1 จะเป็นการส่งข้อความออกไปยังปลายทาง เพื่อใช้ในการทดสอบ



- OS SHELL

ใช้ในการให้เครื่องออกไปทำงานในคำสั่งของดอส แต่ในโปรแกรมนี้ได้เขียนคำสั่งของดอสไว้บ้าง บางคำสั่งซึ่งจะถือว่าเป็นคำสั่งภายในโปรแกรมนี้ ท่านสามารถใช้ได้เลย แต่ถ้าต้องการไปทำงานที่ดอส ให้ท่านพิมพ์คำว่า DOS แล้วกด ENTER โปรแกรมก็จะไปทำงานที่ DOS COMMAND.COM ดังนั้นก็จะใช้งาน DOS ได้ทุกอย่างตามปกติและเมื่อต้องการที่จะกลับมายังที่เดิมให้กด EXIT โปรแกรมก็จะกลับมายังชั้นของ SHELL COMMAND ให้ท่านกด EXIT อีกครั้งหนึ่ง จึงจะกลับเข้ามาที่ใน MAIN MENU

```
        INTERNAL COMMAND

        CLS  =  Clear Screen
        DOS  =  Run command.com
        EXIT =  Exit Dos and OS Shell

(Shell Command): DIR.ASM

 Volume in drive A is CAT_MOND
 Volume Serial Number is 2763-1DF1
 Directory of A:\

CAT      ASM     26958 02-22-93   7:16a
FILECOM  ASM     30078 02-21-93  11:40p
PIA      ASM     57654 12-15-92   5:55p
SHELL    ASM     13452 12-17-92  11:38p
TERMINAL ASM      9973 02-26-93  10:01a
VIDEO_IO ASM     26309 02-22-93   7:12a
       6 File(s)     164424 bytes
                     133120 bytes free

(Shell Command):
```

- EXIT TO DOS

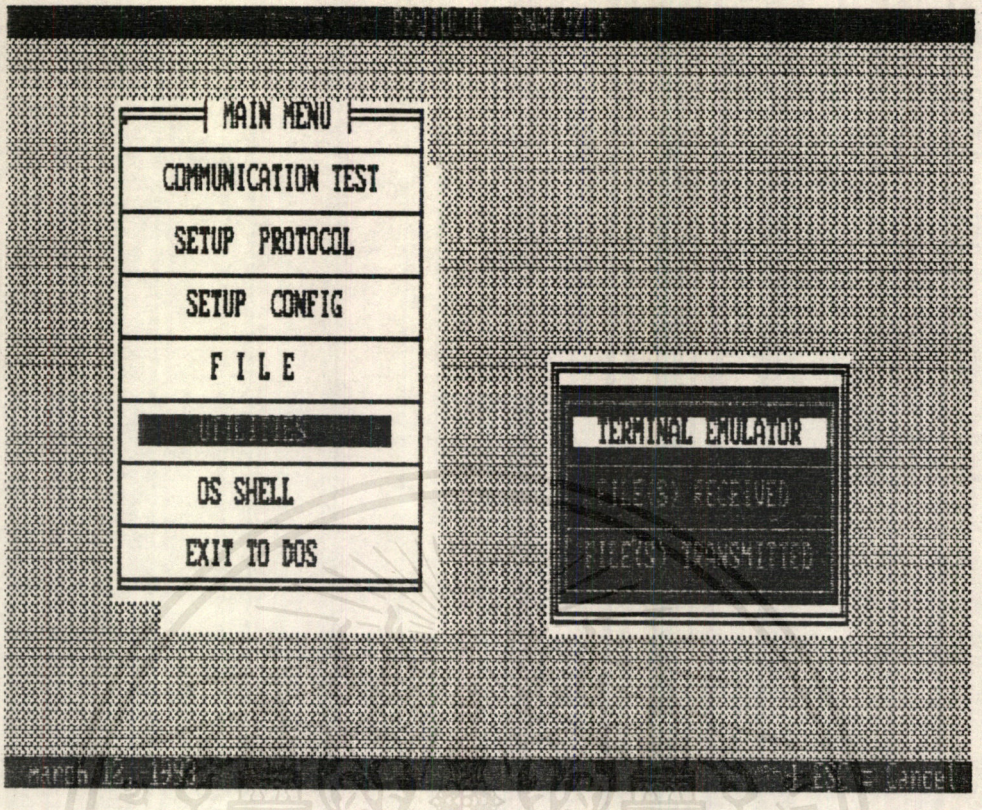
เป็นการเลิกการทำงานของโปรแกรม และโปรแกรมนี้ก็จะทำการเก็บค่าต่างๆ ที่เราทำการ SETUP ไว้ทั้งหมดในโปรแกรมลงใน DISK อีกครั้งหนึ่งในตอนนี้ และเมื่อเราเรียกโปรแกรมนี้ใช้งานใหม่เราก็จะไม่ต้องมาทำการ SETUP ใหม่จะทำให้สะดวกกับการใช้งานได้มาก

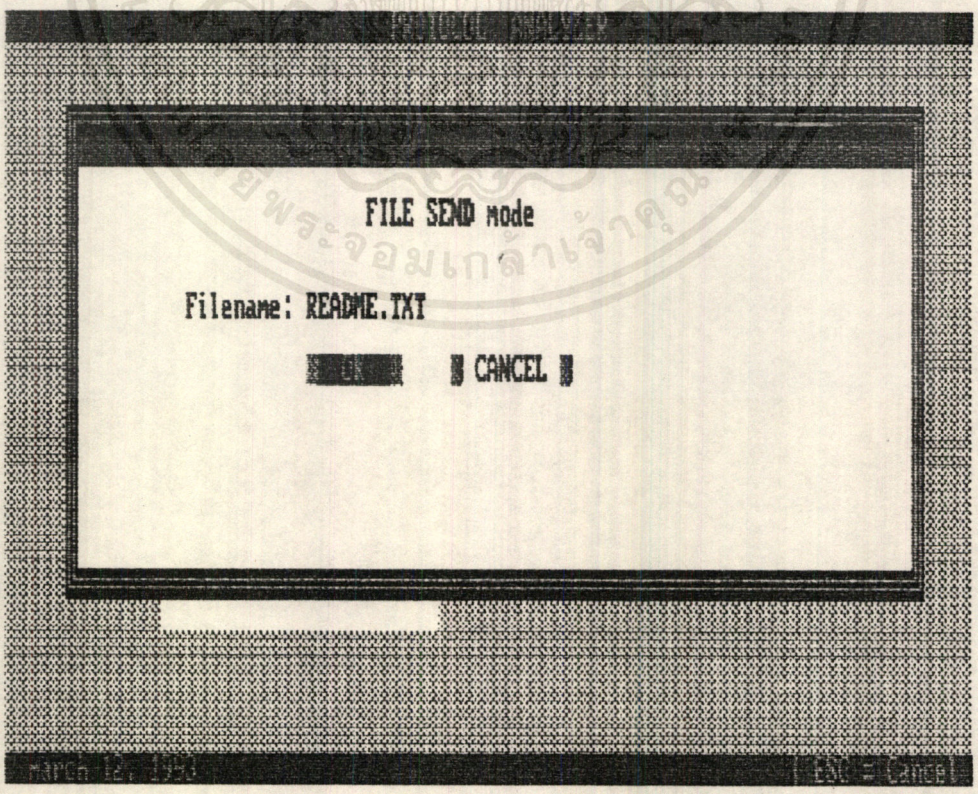
นอกจากนี้ ยังมีโปรแกรม CATPRN.COM ใช้ในการพิมพ์ข้อมูลที่เราได้ทำการเก็บไว้ในตอนที่เราใช้งาน MODE COMMUNICATION TEST และได้ทำการ SAVE DATA ไว้ ดังนั้นเราก็สามารถที่จะนำเอามาทำการพิมพ์ข้อมูลนี้ออกทางเครื่อง PRINTER ได้เพื่อที่จะทำการวิเคราะห์ต่อไป นอกจากนี้ยังสามารถที่จะเลือกรูปแบบการพิมพ์ในลักษณะที่เป็น CHARACTOR หรือ HEX ได้อีกด้วย ลักษณะของการใช้คำสั่งมีดังนี้

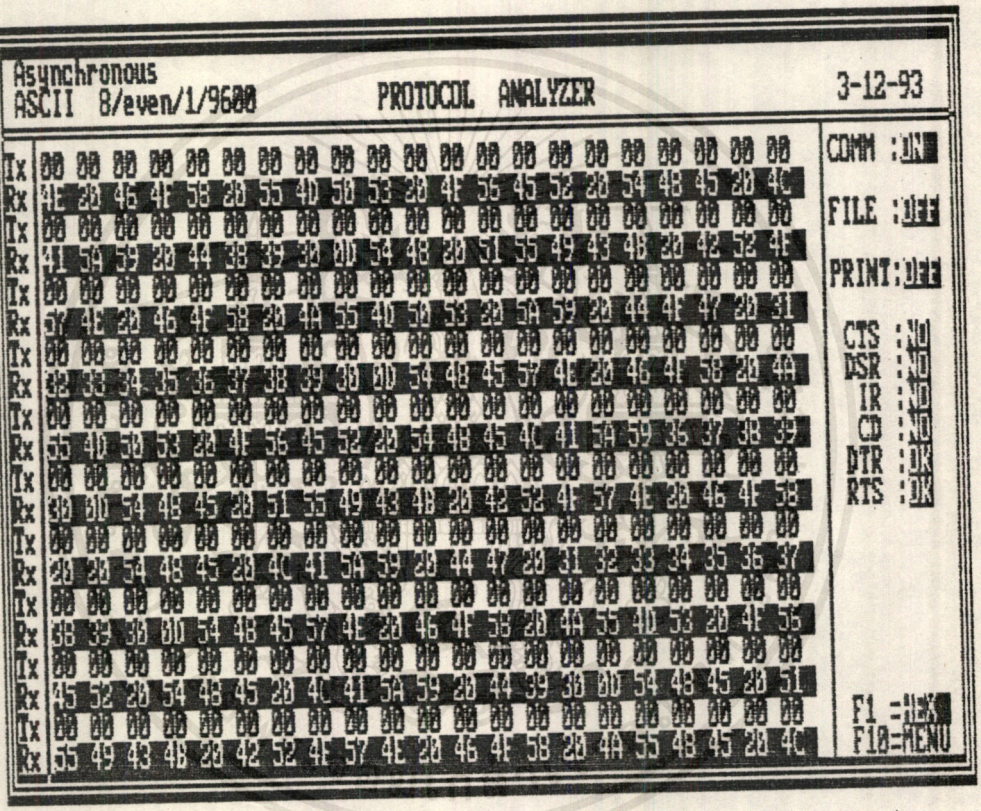
```
A:> CATPRN [filename] /H, /C

    /H = HEX
    /C = CHARACTER
```

เช่น

```
A:> CATPRN TEST.CAT /C
```







เอกสารอ้างอิง

1. Lewis C. Eggebrecht, Interfacing to the IBM personal computer , Howard Q. Sams & Co., Inc., Indiana, 1983.

2. IBM, "Technical Reference", IBM Corporation, revised edition, 1983.

3. Rob Flores, Pete Penninga, Kim Weinman, Taking the Mystery out of Protocol Analysis, Hewlett-Packard Company, Colorado, 1985.

4. ยืน ภู่วรวรรณ, น.ต.ดร.ไพศาล สงวนหมู่ การสื่อสารข้อมูลและไมโครคอมพิวเตอร์ บริษัท ซีเอ็ดยูเคชั่น จำกัด, กรุงเทพมหานคร, 2532

ภาคผนวก ก

CAT.ASM

```
PAGE    60,132
TITLE   PROTOCAL ANALYSIS
DOSSEG

INCLUDE  DOS.INC
INCLUDE  BIOS.INC



EXTRN   ASCII_COM1:WORD
EXTRN   ASCII_COM2:WORD
EXTRN   NUM_CALL:BYTE
EXTRN   ROW:BYTE
EXTRN   COL:BYTE
EXTRN   F2_XON_XOFF:BYTE              ;1 = xoff,FF=xon
EXTRN   OLD_DRIVE:BYTE
EXTRN   NEW_DRIVE:BYTE
EXTRN   CURANT_ADDR:WORD
EXTRN   F_BUFFER:BYTE
EXTRN   FILENAME:BYTE
EXTRN   FHANDLE:WORD
EXTRN   SPEED:BYTE                    ;value setup protocol

PUBLIC  F1_HEX
PUBLIC  MAX
PUBLIC  F_SAVE
F1_HEX      DB  1                                ;1 = display text
    ;ff= display hex
F_SAVE      DB  0FFH                             ;1 = save FF=not save
```

```
MAX         EQU 40                          ;buffer value

PUBLIC  PBUFFER1
PUBLIC  SBUFFER1
PUBLIC  LBUFFER1
PUBLIC  BUFFER1
BUFFER1     LABEL   DWORD
PBUFFER1    DW  0                           ;position in buffer (offse
SBUFFER1    DW  ?                           ;base of buffer (segment)
LBUFFER1    DW  ?                           ;length of buffer

 PUBLIC   SETUP_BYTE
SETUP_BYTE  DB  0BBH                        ;Default value
;
 PUBLIC  CARD_BASE_1
 PUBLIC  CARD_BASE_2
CARD_BASE_1 DW  03F8H                       ;port com 1
CARD_BASE_2 DW  02F8H                       ;port com 2
;
;communication interrupt number
INT_NUM_COM2 DB  0BH                        ;IRQ3 = com2
INT_NUM_COM1 DB  0CH                        ;IRQ4 = com1
;
INT_SEG_COM1 DW  0000H                      ;segment
INT_OFF_COM1 DW  0000H                      ;offset
;
INT_SEG_COM2 DW  0000H                      ;segment
INT_OFF_COM2 DW  0000H                      ;offset
;
```

```
    PUBLIC  CIRC_BUF1
    PUBLIC  DATA_IN1
    PUBLIC  DATA_OUT1
;Circular Buffer and Pointer COMM1
CIRC_BUF1    DB  MAX DUP(00)
     DW  0
DATA_IN1     DW  0                          ;input point comm1
DATA_OUT1    DW  0                          ;out pointercomm2
;
;
    PUBLIC  CIRC_BUF2
    PUBLIC  DATA_IN2
    PUBLIC  DATA_OUT2
;Circular Buffer and Pointer COMM2
CIRC_BUF2   DB  MAX DUP(00)
    DW  0
DATA_IN2    DW  0                          ;input point comm2
DATA_OUT2   DW  0                          ;out point comm2
;
DISPATCH_TABLE  DW  MONITOR_ANALYZER       ;display data comm1,comm2
DW  SETUP_PROTOCAL          ;setup protocol
DW  SHOW_MENU_CONFIG        ;setup display code
DW  MENU_FILE               ;set files
DW  SHOW_UTIL               ;call utility
DW  OS_SHELL                ;call SHELL
DW  EXIT_DOS_6              ;exit to MS DOS

;
;-------------------------------------------------------------------
```

```
;
        .CODE
        EXTRN   FIND_CARD:PROC
        EXTRN   SHOW_SCREENO:PROC
        EXTRN   SHOW_CODE:PROC
        EXTRN   SHOW_PROTOCAL:PROC
        EXTRN   SHOW_ON:PROC
        EXTRN   SHOW_OFF:PROC
        EXTRN   DATE_SET:PROC
        EXTRN   BEEP:PROC
        EXTRN   DISPLAY_DATA_COM:PROC
        EXTRN   RUN_END:PROC
        EXTRN   SHOW_SCREEN:PROC
        EXTRN   SHOW_MENU:PROC
        EXTRN   CLEAR_SCREEN:PROC
        EXTRN   SETUP_PROTOCAL:PROC
        EXTRN   GETKEY:PROC
        EXTRN   PHANTOM_UP:PROC
        EXTRN   PHANTOM_DOWN:PROC
        EXTRN   GETSCRN:PROC
        EXTRN   CURSOR_ON:PROC
        EXTRN   CURSOR_OFF:PROC
        EXTRN   SET_DI:PROC
        EXTRN   SHOW_NO:PROC
        EXTRN   SHOW_OK:PROC
        EXTRN   SET_DTA:PROC
        EXTRN   CLS_ANALY:PROC
        EXTRN   LOAD_BUFFER:PROC
        EXTRN   XON_XOFF:PROC
```

```
        EXTRN   SHOW_F1_HEX:PROC
        EXTRN   SHOW_F3:PROC
        EXTRN   READ_CONFIG:PROC
        EXTRN   SAVE_CONFIG:PROC
        EXTRN   SHOW_TITEL1:PROC
        EXTRN   MENU_FILE:PROC
        EXTRN   PROCESS_PROTOCAL:PROC
        EXTRN   SHOW_MENU_CONFIG:PROC
        EXTRN   BIOS_WRITE_CG:PROC
        EXTRN   SHOW_UTIL:PROC
        EXTRN   OS_SHELL:PROC
        EXTRN   MAIN_AUTO:PROC                  ;in file pta.asm
        EXTRN   STORE_BUFFER:PROC
        EXTRN   LOAD_BUFFER:PROC
        EXTRN   UP_ARROW_KEY:PROC
        EXTRN   DOWN_ARROW_KEY:PROC
        EXTRN   PSTRING:PROC
        EXTRN   DELAY:PROC
START:
        MOV     AX,@DATA
        MOV     DS,AX
        CLI                             ;turn off interrupts
        MOV     SS,AX                   ;Make SS and
        MOV     SP,OFFSET STACK         ;SP relativeto Dgroup
        STI
;
;Adjust memory allocation
        MOV     BX,SP                   ;Convert stack pointer to paragrap
        MOV     CL,4                    ;to get stack size
```

```
    SHR     BX,CL                   ;DIV /16
    ADD     AX,BX                   ;add ss to get end of program
    MOV     BX,ES                   ;get start from end
    SUB     AX,BX                   ;SUBTRACT START FROM END
    @MODBLOK   AX                   ;Release memory after program

;Allocate dynamic memory for file buffer
    @GETBLOK   1000H                ;try to allocate 64k
    MOV     SBUFFER1,AX             ;save buffer segment
    MOV     LBUFFER1,BX             ;save actual length allocated

    PUSHA
    PUSHF
    PUSH    ES
    PUSH    SS

    @SETMODE 3
    CALL    FIND_CARD
    CALL    DATE_SET
    CALL    READ_CONFIG             ;read file config program
    CALL    SHOW_SCREENO            ;
    CALL    SHOW_TITEL1             ;music
    CALL    SHOW_MENU
    CALL    BIOS_WRITE_CG           ;load a new char generator
    CALL    SET_INT_RS232           ;set interrup vector
    CALL    SET_DTA                 ;set disk transfer address
    CALL    PROCESS_PROTOCAL        ;set default comm
    @GetDrv
    MOV     OLD_DRIVE,AL            ;get value drive
```

```
    MOV     NEW_DRIVE,AL

;-------------------------------------------------------------------
;    MAIN MENU
;-------------------------------------------------------------------
;

KB1:        CALL    FLUSH
    CALL    GETKEY
    CMP     AX,1C0DH
    JNE     SPECIAL_KEY
    CALL    DISPATCHER              ;DISPATCH CALL  ROUNTINE
    JC      DOS_EXIT
    JMP     KB1
SPECIAL_KEY:
    CMP     AX,4800H                ;key arrow up
    JNE     KB2
    CALL    PHANTOM_UP
    JMP     KB1
KB2:        CMP     AX,5000H                ;key arrow donw
    JNE     KB3
    CALL    PHANTOM_DOWN
    JMP     KB1
KB3:        CMP     AX,4400H                ;F10 = exit to dos
    JNE     KB1
;
;******************
;*   EXIT       *
;******************
```

```
DOS_EXIT:
    CALL    RUN_END
    CALL    GETKEY
    JC      CNCL_END                ;cancel end program
    CALL    GETSCRN                 ;close windows
    @SetDrv OLD_DRIVE
    CALL    SAVE_CONFIG             ;save config program
    CAL    RESET_INT_RS232         ;get vecter interrupt
    CALL    CLEAR_SCREEN            ;clear screen
    @SETCURSZ 12,13                 ;set cursor size
    @SETCURPOS 00,00                ;set cursor position
    CALL    CURSOR_ON
    CLI
    POP     SS
    POP     ES
    POPF
    POPA
    @FREEBLOK   SBUFFER1            ;release buffer
    @EXIT   0
CNCL_END:
    CALL    GETSCRN                 ;close window
    JMP     KB1
;
;*************************************************************************
;

MONITOR_ANALYZER    PROC
    PUSH    ES
    PUSHA
```

```
    CALL    SHOW_SCREEN


    CALL    MAIN_AUTO
    JNC     OK_AUTO
    JMP     MONITOR_EXIT
OK_AUTO:
    CALL    SHOW_PROTOCAL


    CALL    CURSOR_ON               ;display cursor
    CALL    INITIAL_COMM            ;initialze protocal comm1,2
    CALL    XOFF                    ;Off communication
    CALL    FLUSH                   ;flush keyboard buffer
    CALL    LOAD_BUFFER
;
;*************************************************************************
;*              send and receive characters monitor function key     *
;*************************************************************************
MONITOR:
    CALL    MODEM_STATUS            ;display modem status
    MOV     AH,1                    ;read keyboard status
    INT     16H
    JZ      SER_IMP                 ;Nothing in keyboard buffer
    JMP     SHORT CHAR_TYPED
;Delay loop to allow interrupt to occur
 SER_IMP:    STI                                ;interrupts on
     MOV     CX,50
 DELAYL:     NOP
     NOP
     LOOP    DELAYL
```

```
    CLI                                 ;interrupts off

;----------------------------¦
; test for new data received ¦
;----------------------------¦
    MOV     BX,DATA_OUT1
    CMP     BX,DATA_IN1
    JNE     NEW_DATA
    MOV     BX,DATA_OUT2
    CMP     BX,DATA_IN2
    JNE     NEW_DATA
    STI                                 ;INTERRUPTS ON
    JMP     MONITOR                     ;repeat cycle
;
;----------------¦
;  process char. ¦
;----------------¦
;Retrieve character type from keyboard buffer.
CHAR_TYPED:
    CALL    GETKEY                      ;read keyboard char.

;Test for <F1>,<F2>,<F3> and <F9> keys
;
    CMP     AX,3B00H                    ;<F1>
    JNE     TEST_F2
    CALL    SW_DISPLAY
    JMP     SHORT MONITOR
TEST_F2:
    CMP     AX,3C00H                    ;<F2>
```

```
        JNE     TEST_F3
        CALL    XON_XOFF
        JMP     SHORT MONITOR
TEST_F3:
        CMP     AX,3D00H                ;<F3>
        JNE     TEST_F5
        CALL    F3_SAVE                 ;creat file or close file
        JMP     SHORT MONITOR
TEST_F5:
        CMP     AX,3F00H                ;<F5>
        JNE     TEST_UP
        CALL    F5_CLS                  ;clear buffer
        JMP     SHORT MONITOR
TEST_UP:
        CMP     AX,4800H                ;up arrow key
        JNE     TEST_DOWN
        CALL    UP_ARROW_KEY
        JMP     SHORT MONITOR
TEST_DOWN:
        CMP     AX,5000H                ;down arrow key
        JNE     TEST_F10
        CALL    DOWN_ARROW_KEY
        JMP     SHORT MONITOR
TEST_F10:
        CMP     AX,4400H                ;<F10>
        JE      MONITOR_EXIT
        JMP     SHORT MONITOR
MONITOR_EXIT:
        CALL    XOFF                    ;Off communication
```

```
        MOV     F2_XON_XOFF,1           ;set display OFF
        MOV     F1_HEX,1                ;set display CHAR
        CALL    GETSCRN
        CALL    CURSOR_OFF

        POPA
                        POP     ES

        RET
;
;-------------------;
;New data received ;
;-------------------;
;
NEW_DATA:
        MOV     DX,DATA_OUT1
        CMP     DX,DATA_IN1
        JNE     NEW_DATA_1
        MOV     AX,0700H                ;attr,ascii old 0700h
        INC     DATA_IN1
        INC     DATA_IN1
        CMP     DATA_IN1,MAX
        JNE     NEW_DATA_2
;reset point to start of buffer
        MOV     DATA_IN1,0
        JMP     SHORT NEW_DATA_2
NEW_DATA_1:
        LEA     SI,CIRC_BUF1            ;circular buffer address COMM1
        MOV     BX,DATA_OUT1            ;output pointer
```

```
        ADD     SI,BX                   ;buffer strat+displacement
        MOV     AX,WORD PTR [SI]        ;get character
NEW_DATA_2:
        MOV     ASCII_COM1,AX           ;attr,ascii

        MOV     DX,DATA_OUT2
        CMP     DX,DATA_IN2
        JNE     NEW_DATA_3
        MOV     AX,7000H                ;attr ascii old 7000h
        INC     DATA_IN2
        INC     DATA_IN2
        CMP     DATA_IN2,MAX
        JNE     NEW_DATA_4
        MOV     DATA_IN2,0              ;reset start point
        JMP     SHORT NEW_DATA_4
NEW_DATA_3:
        LEA     SI,CIRC_BUF2            ;circular buffer address COMM2
        MOV     BX,DATA_OUT2
        ADD     SI,BX
        MOV     AX,WORD PTR [SI]
NEW_DATA_4:
        MOV     ASCII_COM2,AX           ;attr,ascii

        INC     BX                      ;update outpot pointer
        INC     BX
        CMP     BX,MAX                  ;point overflows buffer?
        JNE     OK_OUT_PTR
        MOV     BX,0                    ;reset to start of buffer
```

```
OK_OUT_PTR:
    MOV     DATA_OUT1,BX            ;Update
    MOV     DATA_OUT2,BX
;
;display byte taken from buffer
    TI                              ;Interrupt on
    CALL    STORE_BUFFER
    CALL    SAVE_ANALY              ;save data to disk
    CALL    DISPLAY_DATA_COM        ;write to screen
    JMP     MONITOR

MONITOR_ANALYZER    ENDP
;-------------------------------------------------------------------
;
;set vector interrup comm1,comm2

     PUBLIC  SET_INT_RS232
SET_INT_RS232   PROC
    PUSHA                           ;save all register
    PUSH    ES
    @GETINT 0BH
    MOV     INT_SEG_COM1,ES         ;save segment
    MOV     INT_OFF_COM1,BX         ;and offset

    @GETINT 0CH
    MOV     INT_SEG_COM2,ES
    MOV     INT_OFF_COM2,BX
    MOV     AH,25H
```

```
        MOV     AL,INT_NUM_COM1
        MOV     DX,OFFSET CS:RS232_COM1
        PUSH    DS                          ;save ds
        PUSH    CS                          ;cs=ds
        POP     DS
        INT     21H
        POP     DS                          ;get ds

        MOV     AH,25H
        MOV     AL,INT_NUM_COM2
        MOV     DX,OFFSET CS:RS232_COM2
        PUSH    DS
        PUSH    CS
        POP     DS
        INT     21H
        POP     DS

        POP     ES
        POPA                                ;pop all  general registers
        RET
SET_INT_RS232   ENDP
;
;

RESET_INT_RS232     PROC
;Restore original interrupt vector for communication interrupt
        PUSHA                               ;push all general registers
        PUSH    ES
        PUSH    DS
```

```
    MOV     DX,INT_OFF_COM1         ;com1  IRQ3
    MOV     AX,INT_SEG_COM1
    MOV     DS,AX
    MOV     AH,25H                  ;set interrupt
    MOV     AL,INT_NUM_COM1         ;machine interrupt number
    INT     21H
    POP     DS

    PUSH    DS
    MOV     DX,INT_OFF_COM2         ;om2
    MOV     AX,INT_SEG_COM2
    MOV     DS,AX
    MOV     AH,25H
    MOV     AL,INT_NUM_COM2
    INT     21H
    POP     DS
    POP     ES
    POPA                            ;pop all general resiters
    RET

RESET_INT_RS232     ENDP

    PUBLIC  INITIAL_COMM
INITIAL_COMM PROC
    CLI                                     ;interrupt off
;Reset buffer point to start of buffer
    MOV     DATA_IN1,0
    MOV     DATA_OUT1,0
    MOV     DATA_IN2,0
```

```
    MOV     DATA_OUT2,0
;
;Set DX to base address of RS232-C
    MOV     DX,CARD_BASE_1                  ;com1
    CALL    SETUP_RS232
    MOV     DX,CARD_BASE_2                  ;com2
    CALL    SETUP_RS232
    STI
  RET
INITIAL_COMM ENDP
;
;
    PUBLIC  XOFF
XOFF        PROC
;Disable communications interrups by setting bits
;for IRQ3 andIRQ4 line on the interrupt mask register
    IN      AL,21H
    OR      AL,18H                          ;set bit3 and 4
    OUT     21H,AL
    JMP     SHORT $+2
    RET
XOFF        ENDP
;
;
;Enable communication interrupts by resetting the bits
;Corresponding to the IRQ3 and IRQ4 line on the interrupt
;mask register (port address = 21h)
    PUBLIC  XON
XON         PROC
```

```
        IN      AL,21H
        AND     AL,0E7H                 ;reset bit3 and bit4
        OUT     21H,AL                  ;of IC 8259A
        JMP     SHORT $+2
        RET
XON     ENDP
;
;
        PUBLIC  SETUP_RS232
SETUP_RS232   PROC
;initialize modem control register for Data Terminal Ready
;(bit0),Request To Sent(bit1)and OUTPUT2 (bit3)
;DX is still holding port address
        MOV     DL,0FCH                 ;MCR address
        MOV     AL,00001011B            ;bits 0,1 and 3 set
        OUT     DX,AL
        JMP     SHORT $+2
;
;set bit 7 of line control register (DLAB) to access
;the interrupt enable register at XF9H
        MOV     DL,0FBH                 ;XFBH=line control registe
        IN      AL,DX                   ;read byte at port
        JMP     SHORT $+2
        AND     AL,7FH                  ;reset DLAB
        OUT     DX,AL                   ;write to LCR
        JMP     SHORT $+2
;
;Enable interrupt for Data Ready Only
        MOV     DL,0F9H                 ;interrupt enable register
```

```
    MOV     AL,01H                          ;Data Ready interrupt and
    OUT     DX,AL                          ;Modem Status Interrupt
    JMP     SHORT $+2
;
    RET
SETUP_RS232     ENDP
;
;-----------------------------------------------------------------------

     PUBLIC  RS232_COM1
;*************************************************************
;           INTERRUPT SERVICE RUTINE COM1                   *
;*************************************************************
;
RS232_COM1  PROC
    STI                                     ;interrupt on
    PUSH    AX
    PUSH    BX
    PUSH    DX
    PUSH    DI
    PUSH    DS
    PUSH    ES

    MOV     AX,@DATA
    MOV     DS,AX

;Check line status register for reception error and data ready
DATA_CHECK_COM1:
    MOV     DX,CARD_BASE_1
```

```
    MOV     DL,0FDH                     ;line status register
    IN      AL,DX                       ;read port byte
    JMP     SHORT $+2
;Check for data ready on line
    TEST    AL,01H
    JNZ     DATA_READY_COM1
;Check ofr error code
    TEST    AL,1EH
    JNZ     DATA_ERROR_COM1
    JMP     DATA_CHECK_COM1
DATA_ERROR_COM1:
    MOV     AL,'?'                      ;error symbol
    MOV     AH,0F8H                     ;Attr high reves flag
    JMP     SHORT STORE_BYTE_COM1
DATA_READY_COM1:
    MOV     DL,0F8H
    IN      AL,DX                       ;get byte
    JMP     SHORT $+2
    MOV     AH,07H                      ;Attr normal
;place byte in cirular buffer
STORE_BYTE_COM1:
    LEA     DI,CIRC_BUF1
    MOV     BX,DATA_IN1
    ADD     DI,BX
    MOV     WORD PTR[DI],AX             ;store in buffer
;index input pointer reset if point overflows buffer
    INC     BX
    INC     BX
    CMP     BX,MAX                      ;past end of buffer
```

```
    JNE     OK_IN_PTR_COM1
;reset point to start of buffer
    MOV     BX,0
OK_IN_PTR_COM1:
    MOV     DATA_IN1,BX
;Signal end-of-interrupt fo the interrupt command register
    MOV     AL,20H
    OUT     20H,AL                          ;EOI port address
    JMP     SHORT $+2
;restore register from stack
    POP     ES
    POP     DS
    POP     DI
    POP     DX
    POP     BX
    POP     AX
    IRET
RS232_COM1  ENDP
;
;
    PUBLIC   RS232_COM2
;*************************************************************
;           INTERRUPT SERVICE RUTINE COM2                   *
;*************************************************************
;
RS232_COM2  PROC
    STI                                     ;interrupt on
    PUSH    AX
    PUSH    BX
```

```
    PUSH    DX
    PUSH    DI
    PUSH    DS
    PUSH    ES

    MOV     AX,@DATA
    MOV     DS,AX
;Check line status register for reception error and data ready
DATA_CHECK_COM2:
    MOV     DX,CARD_BASE_2
    MOV     DL,0FDH                 ;line status register
    IN      AL,DX                   ;read port byte
    JMP     SHORT $+2
;Check for data ready on line
    TEST    AL,01H
    JNZ     DATA_READY_COM2
;Check ofr error code
    TEST    AL,1EH
    JNZ     DATA_ERROR_COM2
    JMP     DATA_CHECK_COM2
DATA_ERROR_COM2:
    MOV     AL,'?'                  ;error symbol
    MOV     AH,0F8H                 ;Attr
    JMP     SHORT STORE_BYTE_COM2
DATA_READY_COM2:
    MOV     DL,0F8H
    IN      AL,DX                   ;get byte
    JMP     SHORT $+2
    MOV     AH,70H                  ;Attr revers comm2
```

```
STORE_BYTE_COM2:
    LEA     DI,CIRC_BUF2
    MOV     BX,DATA_IN2
    ADD     DI,BX
    MOV     WORD PTR[DI],AX                 ;store in buffer
;index input pointer reset if point overflows buffer
    INC     BX
    INC     BX
    CMP     BX,MAX                          ;past end of buffer
    JNE     OK_IN_PTR_COM2
;reset point to start of buffer
    MOV     BX,0
OK_IN_PTR_COM2:
    MOV     DATA_IN2,BX
;Signal end-of-interrupt fo the interrupt command register
    MOV     AL,20H
    OUT     20H,AL                          ;EOI port address
    JMP     SHORT $+2
;restore register from stack
    POP     ES
    POP     DS
    POP     DI
    POP     DX
    POP     BX
    POP     AX
    IRET
RS232_COM2  ENDP
;--------------------------------------------------------------------
;
```

```
;display modem status of prot com1
MODEM_STATUS    PROC
    MOV     BX,0A4BH
    CALL    SET_DI
    MOV     DX,CARD_BASE_1
    MOV     DL,0FEH                 ;read modem status
    IN      AL,DX
    TEST    AL,10H                  ;CTS Bit4
    JZ      NO_CTS
    CALL    SHOW_OK
    JMP     SHORT DSR
NO_CTS:     CALL    SHOW_NO
DSR:        MOV     BX,0B4BH
    CALL    SET_DI
    TEST    AL,20H                  ;DSR Bit5
    JZ      NO_DSR
    CALL    SHOW_OK
    JMP     SHORT RI
NO_DSR:     CALL    SHOW_NO
RI:         MOV     BX,0C4BH
    CALL    SET_DI
    TEST    AL,40H                  ;RI Bit6
    JZ      NO_RI
    CALL    SHOW_OK
    JMP     SHORT CD
NO_RI:      CALL    SHOW_NO
CD:         MOV     BX,0D4BH
    CALL    SET_DI
    TEST    AL,80H                  ;CD Bit7
```

```
    JZ      NO_CD
    CALL    SHOW_OK
    JMP     SHORT DTR
NO_CD:      CALL    SHOW_NO
;
DTR:        MOV     BX,0E4BH
    CALL    SET_DI
    MOV     DX,CARD_BASE_1
    MOV     DL,0FCH                 ;modem control register
    IN      AL,DX
    TEST    AL,01H                  ;DTR Bit0
    JZ      NO_DTR
    CALL    SHOW_OK
    JMP     SHORT RTS
NO_DTR:     CALL    SHOW_NO
RTS:        MOV     BX,0F4BH
    CALL    SET_DI
    TEST    AL,02H                  ;RTS Bit1
    JZ      NO_RTS
    CALL    SHOW_OK
    JMP     SHORT END_MODEM_STATUS
NO_RTS:     CALL    SHOW_NO
END_MODEM_STATUS:
    RET
MODEM_STATUS    ENDP

;
;--------------------;
;   FLUSH BUFFER     ;
```

```
;---------------------¦
;flush keyboard buffer of old chraracters
    PUBLIC  FLUSH
FLUSH       PROC    NEAR
FLUSH_1:
    MOV     AH,1                    ;read keyboard status
    INT     16H
    JZ      NO_OLD_CHARS
;Flush old character
     CALL    GETKEY                 ;read character from keyboard
    JMP     FLUSH_1
NO_OLD_CHARS:
    RET
FLUSH       ENDP
;
;
    PUBLIC  SET_PROTOCAL
SET_PROTOCAL   PROC
;-------------------¦
;  SET PROTOCOL     ¦
;-------------------¦
;set default communications parameters
;bit mask: 101xxxxx...... Baud = 2400
;          xxx11xxx...... Parity = even
;          xxxxx0xx.......Stop bits = 1
;          xxxxxx11.......Word length = 8
;          10111011 = BBh
;
    MOV     AL,SETUP_BYTE           ;control code
```

```
    MOV     AH,0
    MOV     DX,0                    ;COMM1
    INT     14H

    MOV     AL,SETUP_BYTE
    MOV     AH,0                    ;initialize comm
    MOV     DX,1                    ;comm2
    INT     14H
    RET
SET_PROTOCAL   ENDP
;
;
EXIT_DOS_6      PROC
STC
RET
EXIT_DOS_6      ENDP
;
PUBLIC  SW_DISPLAY
SW_DISPLAY      PROC
NEG     F1_HEX
CALL    CLS_ANALY                   ;clear screen
CALL    LOAD_BUFFER                 ;load data in buffer
CALL    SHOW_F1_HEX                 ;display char or hex
RET
SW_DISPLAY      ENDP
;
;
SAVE_ANALY      PROC
PUSHA
```

```
CMP     F_SAVE,0FFH                     ;ff=not save 1 = save
JE      EXIT_SAVE
MOV     BX,FHANDLE
MOV     CX,4
MOV     DX,OFFSET ASCII_COM1
MOV     AH,40H
INT     21H
JC      ERR_SAVE
CMP     AX,4
JNE     EXIT_SAVE
JMP     SHORT EXIT_SAVE
ERR_SAVE:       CALL    BEEP                    ;error diskful
EXIT_SAVE:      POPA
RET
SAVE_ANALY      ENDP

;
;press key F3 creat file or close file
F3_SAVE         PROC
PUSHA
PUSHF
NEG     F_SAVE                          ;1=save  FF=not save
CALL    SHOW_F3
CMP     F_SAVE,1
JE      CREAT_FILE
@ClosFil FHANDLE                        ;close file
JMP     SHORT EXIT_F3
CREAT_FILE:
@MAKFIL FILENAME,0
```

```
JC      EXIT_F3
MOV     FHANDLE,AX                      ;save file handle
@WRITE  SPEED,15,FHANDLE
EXIT_F3:        POPF
POPA
RET
F3_SAVE         ENDP

;
PUBLIC  F5_CLS
F5_CLS          PROC
MOV     PBUFFER1,0                      ;offset
MOV     CURANT_ADDR,0                   ;
MOV     F_BUFFER,0                      ;flage buffer
CALL    CLS_ANALY
CALL    PROCESS_PROTOCAL
CALL    SHOW_PROTOCAL
MOV     ROW,4
MOV     COL,4
RET
F5_CLS          ENDP
;
DISPATCHER      PROC
PUSH    AX
PUSH    BX
PUSH    CX
PUSH    DX
XOR     BX,BX
MOV     BL,NUM_CALL
```

```
SHL     BX,1
CALL    WORD PTR [BX+DISPATCH_TABLE]
POP     DX
POP     CX
POP     BX
POP     AX
RET
DISPATCHER      ENDP
;
;-----------------------------------------------------------------------

    END     START
```

PTA.ASM

```
INCLUDE DOS.INC
INCLUDE BIOS.INC
;
EXTRN   BOX_CHAR:BYTE
EXTRN   ATTRIBUTE:BYTE
EXTRN   UL_COL:BYTE
EXTRN   UL_ROW:BYTE
EXTRN   LR_COL:BYTE
EXTRN   LR_ROW:BYTE
EXTRN   ROW:BYTE
EXTRN   COL:BYTE
EXTRN   SETUP_BYTE:BYTE
EXTRN   NUM_CHAR_COLOR:WORD
EXTRN   TRKY_FREQ:WORD
EXTRN   TRKY_TIME:BYTE
EXTRN   TITEL1:BYTE
EXTRN   TITEL2:BYTE
EXTRN   F1_HEX:BYTE
EXTRN   NEW_DRIVE:BYTE
EXTRN   MESS_6:BYTE
EXTRN   CARD_BASE_1:WORD
EXTRN   CARD_BASE_2:WORD
EXTRN   F_SAVE:BYTE
EXTRN   ERMES_1:BYTE
EXTRN   BUFFER1:DWORD
EXTRN   PBUFFER1:WORD
EXTRN   LBUFFER1:WORD
EXTRN   F_BUFFER:BYTE
EXTRN   CURANT_ADDR:WORD
;
```

```
EXTRN   F_BUFFER:BYTE
EXTRN   CURANT_ADDR:WORD
;

TYPE_PROTOCAL   DB  'Asynchronous$'
HEAD            DB  'PROTOCOL  ANALYZER$'
ASCII           DB  'ASCII $'
EBCDIC          DB  'EBCDIC$'
SHOW_COM        DB  'COMM :',24H
SHOW_FILE       DB  'FILE :',24H
SHOW_PRINT      DB  'PRINT:',24H
TITEL_MODEM     DB  'CTS :',24H
DB  'DSR :',24H
DB  ' IR :',24H
DB  ' CD :',24H
DB  'DTR :',24H
DB  'RTS :',24H
;
TX              DB  54H,78H,24H
RX              DB  52H,78H,24H
ON              DB  'ON ',24H
OFF             DB  'OFF',24H
PUBLIC  F2_XON_XOFF
F2_XON_XOFF     DB  1                                   ;1 =xoff,FF=xoff
ERR1            DB  'File too big$'
;-------------------------------------------------
SELECT          DB  'IMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMM;0',24H
DB  ':          SETUP PROTOCOL              :0',24H
DB  'GDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDD6 ',24H
DB  ': SELEC VALUE BAUD RATE    (1-8) :  :',24H
```

```
DB  ': SELEC VALUE PARITY      (1-3) :   : ',24H
DB  ': SELEC VALUE STOP BITS   (1-2) :   : ',24H
DB  ': SELEC VALUE WORD LENGHT (1-2) :   : ',24H
DB  ':                                   : ',24H
DB  ':      2 OK 2      2 CANCEL 2       : ',24H
DB  'HMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMM< ',24H
DB  '00000                               ',24H

MENU_BAUD       DB  'UMMMMMMMMMMM80',24H
DB  '3 BAUD RATES 3 ',24H
DB  'CDDDDDDDDDDD4 ',24H
DB  '3  1 = 110   3 ',24H
DB  '3  2 = 150   3 ',24H
DB  '3  3 = 300   3 ',24H
DB  '3  4 = 600   3 ',24H
DB  '3  5 = 1200  3 ',24H
DB  '3  6 = 2400  3 ',24H
DB  '3  7 = 4800  3 ',24H
DB  '3  8 = 9600  3 ',24H
DB  'TMMMMMMMMMM> ',24H
DB  '000          ',24H

MENU_PARITY     DB  'UMMMMMMMM80',24H
DB  '3  PARITY  3 ',24H
DB  'CDDDDDDDDD4 ',24H
DB  '3 1 = ODD  3 ',24H
DB  '3 2 = NONE 3 ',24H
DB  '3 3 = EVEN 3 ',24H
DB  'TMMMMMMMM> ',24H
DB  '000        ',24H
```

```
MENU_STOP       DB  'UMMMMMMMMM80',24H
DB  '3 STOP BITS 3 ',24H
DB  'CDDDDDDDDDD4 ',24H
DB  '3 1 = 1 Bit 3 ',24H
DB  '3 2 = 2 Bil 3 ',24H
DB  'TMMMMMMMMM> ',24HJ
DB  '000          ',24H

MENU_LENGHT     DB  'UMMMMMMMMMMM80',24H
DB  '3 WORD LENGHT 30',24H
DB  'CDDDDDDDDDDDD4 ',24H
DB  '3  1 = 7 Bit  3 ',24H
DB  '3  2 = 8 Bit  3 ',24H
DB  'TMMMMMMMMMM> ',24H
DB  '0000         ',24H

TITEL_FILE      DB  'UMMMMM;',24H
DB  '3 Drive :',24H
DB  '3 Path  :',24H
DB  '3 Load  :',24H
DB  'TMMMMM<',24H

TITEL_DRIVE     DB  'Drive : ',24H
TITEL_PATH      DB  'Path  : ',24H
TITEL_FNAME     DB  'File Name : ',24H
CHAR_HIGTH      DB  'D',24H
DB  'P',24H
DB  'L',24H
```

```
PUBLIC  PATH_NAME
PATH_IN         DB  63                                  ;***set ma
DB  0
PATH_NAME       DB  64 DUP (00)
F_LNAME         DB  63
DB  0
F_LBUFFER       DB  64 DUP (00)
FSIZE           DW  0                                   ;save size
ERR_PATH        DB  'ERROR PATH NOT FOUND',24H


PUBLIC  CNCL
PUBLIC  SPEED
CNCL            DB  '3 ESC = Cancel',24H
SPEED           LABEL   BYTE
WORD%           DB  '8/'
PAR%            DB  'even/'
STOP%           DB  '1/'
BAUD%           DB  '2400',0,24H
BAUD$$          DB  '110 150 300 600 1200240048009600'
PAR$$           DB  'odd noneeven'
;
;-------------------------------------------------------
PUBLIC  TERMINAL_IN
PUBLIC  TERMINAL_OUT
PUBLIC  SAND_COM
PUBLIC  RECEIVE_COM
PUBLIC  FILENAME
PUBLIC  FHANDLE
;********** FILE CONFIGURATION ***********
SBAUD           DB  '6'                 ;baud    = 2400
```

```
SPARITY         DB  '3'                         ;parity = even
SSTOPBIT        DB  '1'                         ;stopbit= 1
SWORD           DB  '2'                         ;word   = 8

DISP_CODE       DB  01H                         ;1 = ASCII,FFH= EBCDIC
AUTO_SET        DB  01H                         ;1=ON  , FF=OFF
TERMINAL_IN     DB  '1'                         ;terminal port comm (1-2
TERMINAL_OUT    DB  '1'
RECEIVE_COM     DB  '1'                         ;receive file port comm(
SAND_COM        DB  '1'                         ;sand file port comm (1-
NAMEBUF         DB  30,?
FILENAME        DB  30 DUP ('A:TEST.CAT',0)   ;buffer for file name
DB  00
FHANDLE         DW  0000
FILENAME1       DB  'CONFIG.CAT',0              ;file name save config
HANDLE1         DW  0000
;-------------------------------------------------------------
;
AUTO_FLAGE      DB  01H                         ;1=OK ready FF= user sto
SHOW_HEX        DB  'F1 =',24H
F10_MENU        DB  'F10=MENU',24H
TITLE_HEX       DB  'HEX ',24H
TITLE_CHAR      DB  'CHAR',24H
;
TEXT_END        DB  'RUN END PROTOCOL ANALYZER',24H
DB  '         Good Bye !       ',24H
DB  '        Press any key     ',24H
POSITION        DW  0H                              ;row col
;-------------------------------------------------------------
;
```

```
PUBLIC  NUM_CALL
NUM_CALL          DB  0H
PHANTOM_CURSOR    DW  050BH
NUM_CHAR          DW  00H
CURSOR_LINE       DW  0309H
TEXT_MENU         DB  'UMMMMM5 MAIN MENU FMMMMM80',24H
 DB  'CDDDDDDDDDDDDDDDDDDDDDDDD40',24H
 DB  '3   COMMUNICATION TEST   3 ',24H
 DB  'CDDDDDDDDDDDDDDDDDDDDDDDD4 ',24H
 DB  '3    SETUP  PROTOCOL     3 ',24H
 DB  'CDDDDDDDDDDDDDDDDDDDDDDDD4 ',24H
 DB  '3     SETUP  CONFIG      3 ',24H
 DB  'CDDDDDDDDDDDDDDDDDDDDDDDD4 ',24H
 DB  '3       F I L E          3 ',24H
 DB  'CDDDDDDDDDDDDDDDDDDDDDDDD4 ',24H
 DB  '3      UTILITIES         3 ',24H
 DB  'CDDDDDDDDDDDDDDDDDDDDDDDD4 ',24H
 DB  '3      OS SHELL          3 ',24H
 DB  'CDDDDDDDDDDDDDDDDDDDDDDDD4 ',24H
 DB  '3     EXIT TO DOS        3 ',24H
 DB  'TMMMMMMMMMMMMMMMMMMMMMMM> ',24H
 DB  '0000                       ',24H

UTIL_TEXT         DB  ZDDDDDDDDDDDDDDDDDDDDDD?',24H
 DB  '3 TERMINAL EMULATOR  3',24H
 DB  'CDDDDDDDDDDDDDDDDDDDDDD4',24H
 DB  '3 FILE(S) RECEIVED   3',24H
 DB  'CDDDDDDDDDDDDDDDDDDDDDD4',24H
 DB  '3 FILE(S) TRANSMITTED 3',24H
 DB  '@DDDDDDDDDDDDDDDDDDDDDDV',24H
```

```
;
NUM_CALL_SMALL   DB 0
DISPATCH_TABLE1  DW TERMINAL              ;TERMINAL_EMULATOR
 DW MAIN_FILE_RECEIVE   ;FILE_RECEIVED
 DW MAIN_FILE_SAND      ;FILE_TRANSMITTED

SHOW_HEAD_CONFIG DB 'PROGRAM SETUP CONFIGURATION',24H
SHOW_CONFIG      DB  'Disolay Code                :',24H
DB  'Auto Setup Protocol         :',24H
DB  'File Name Save For Analysis :',24H
DB  'Terminal  In Put   Comm(1-2):',24H
DB  'Terminal Out Put   Comm(1-2):',24H
DB  'File Transmitted   Comm(1-2):',24H
DB  'File Received      Comm(1-2):',24H

show_config1    DB  '1 ESC = Exit To Menu 1',24h
;-------------------------------------------------------
;
long_date       db  19  dup (0)             ;long date string
short_date      db  9   dup (0)             ;short date string
thai_date       db  24  dup (0)             ;thai date string
;time_buffer     db  11  dup (0)              ;time buffer
;
month_text  db 7,'January  '                ;English text of month
    db 8,'February '
    db 5,'march    '
    db 5,'April    '
    db 3,'may      '
    db 4,'June     '
    db 4,'July     '
```

```
    db 6,'August   '
    db 9,'September'
    db 7,'October  '
    db 8,'November '
    db 8,'December '
;
;month_thai  db 6,'A!CR$A   '                    ;Thai text of month
;            db 10,'!A@R>Q981'
;            db 6,'AU9R$A   '
;            db 6,'AIRB9    '
;            db 7,'>DI@R$A  '
;            db 8,'AT69RB9  '
;            db 7,'!C!.R$A  '
;            db 7,'JT'KR$A  '
;            db 7,'!Q9BRB9  '
;            db 6,'5ER$A    '
;            db 9,'>DH(T!RB9'
;            db 7,'8Q9GR$A  '
;
already_load db 0                                ;flag for checking alread
day          db ?                                ;store day
month        db ?                                ;store month
year         db ?                                ;store English year
thai_yr      db ?                                ;store thai year
text1        db ', 19'                           ;mask of English year
text2        db ' >.H. 25'                       ;mask of thai year
;
;*************************************************************
;
EXTRN   BOX:PROC
```

```
EXTRN   SAVSCRN:PROC
EXTRN   GETSCRN:PROC
EXTRN   PCHAR:PROC
EXTRN   SET_DI:PROC!
EXTRN   GOTO_XY:PROC
EXTRN   WRITE_CHAR:PROC
EXTRN   LOAD_BUFFER:PROC
EXTRN   COLOR:PROC
EXTRN   CLEAR_SCREEN:PROC
EXTRN   CURSOR_OFF:PROC
EXTRN   CURSOR_ON:PROC
EXTRN   XON:PROC
EXTRN   XOFF:PROC
EXTRN   DISP_ST_0:PROC
EXTRN   SET_PROTOCAL:PROC
EXTRN   FLUSH:PROC
EXTRN   PLAY:PROC
EXTRN   DELAY:PROC
EXTRN   UP_ARROW_KEY:PROC
EXTRN   DOWN_ARROW_KEY:PROC
EXTRN   SW_DISPLAY:PROC
EXTRN   TERMINAL:PROC
EXTRN   MAIN_FILE_RECEIVE:PROC
EXTRN   MAIN_FILE_SAND:PROC
EXTRN   SET_DRIVE:PROC
EXTRN   UPCASE:PROC
EXTRN   BEEP:PROC
EXTRN   SHELL:PROC
;
;*****************************************
```

```
; show screen  moniter data and analysis *
;*****************************************

PUBLIC  SHOW_SCREEN
SHOW_SCREEN     PROC
CALL    BOX_SCRN
CALL    WRITE_HEAD
CALL    LINE_HOR
MOV     DX,0404H
CALL    GOTO_XY
RET
SHOW_SCREEN     ENDP
;
;************************************************
;*  show screen main menu                       *
;************************************************
PUBLIC  SHOW_SCREEN0
SHOW_SCREEN0    PROC
CALL    CURSOR_OFF
MOV     BX,0000H
CALL    SET_DI
CALL    SCREEN1
CALL    SCREEN2
CALL    SCREEN1
MOV     BX,0020H
CALL    SET_DI
MOV     ATTRIBUTE,1EH           ;yel-blu
MOV     SI,OFFSET HEAD
CALL    PSTRING
MOV     BX,1802H
```

```
CALL    SET_DI
CALL    SHOW_LONG_DATE
MOV     BL,41H
CALL    SET_DI
MOV     SI,OFFSET CNCL
CALL    PSTRING
;               MOV     ATTRIBUTE,01H
;               CALL    SHOW_TITEL1
RET
SHOW_SCREEN0    ENDP
;
;
;               PUBLIC  BOX_SCRN
BOX_SCRN        PROC
MOV     ATTRIBUTE,75H                   ;mag-wht
MOV     UL_COL,00H
MOV     UL_ROW,00H
MOV     LR_COL,79
MOV     LR_ROW,24
MOV     BOX_CHAR,0BBH
MOV     BOX_CHAR+1,0C8H
MOV     BOX_CHAR+2,0C9H
MOV     BOX_CHAR+3,0BCH
MOV     BOX_CHAR+4,0BAH
MOV     BOX_CHAR+5,0CDH
;               MOV     ATTRIBUTE,73H                   ;attribute re
CALL    SAVSCRN                         ;save screen
MOV     BX,0300H
CALL    SET_DI
MOV     AL,204                          ;L
```

```
CALL    PCHAR
;               MOV     ATTRIBUTE,01H                   ;
MOV     AL,205                          ;M
MOV     CX,78
N_TIME:
CALL    PCHAR
LOOP    N_TIME
;
MOV     AL,185                          ;9
CALL    PCHAR
MOV     BX,0344H
CALL    SET_DI
MOV     AL,209                          ;Q
CALL    PCHAR
MOV     BX,0303H
CALL    SET_DI
CALL    PCHAR

MOV     ATTRIBUTE,13                    ;magenta color
MOV     DX,0403H
MOV     POSITION,DX
CALL    GOTO_XY
MOV     DL,179                          ;3
MOV     CX,20
CALL    PRINT_HOR
MOV     DX,0444H
MOV     POSITION,DX
CALL    GOTO_XY
MOV     DL,179                          ;3
MOV     CX,20
```

```
CALL    PRINT_HOR

RET
BOX_SCRN        ENDP
;
;-------------------------------------;
;       DL = ASCII                    ;
;       CX = number of time write char ;
;-------------------------------------;
PRINT_HOR       PROC
PUSH    CX
;               MOV     ATTRIBUTE,07            ;attr normal!
N_LOOP:
CALL    WRITE_CHAR                      ;
PUSH    DX                              ;save char

MOV     DX,POSITION                     ;get row col
INC     DH                              ;dec line row
MOV     POSITION,DX                     ;save row col
CALL    GOTO_XY                         ;set cursor

POP     DX                              ;get char
LOOP    N_LOOP
MOV     ATTRIBUTE,01H                   ;attr normal pstring
POP     CX
RET
PRINT_HOR       ENDP
;
;
;               PUBLIC  WRITE_HEAD
```

```
WRITE_HEAD      PROC
MOV     BX,0220H
CALL    SET_DI
MOV     ATTRIBUTE,1EH                   ;yel-blu
MOV     SI,OFFSET HEAD
CALL    PSTRING

.
MOV     ATTRIBUTE,14                    ;yellow
MOV     BX,0102H
CALL    SET_DI
MOV     SI,OFFSET TYPE_PROTOCAL         ;asynch
CALL    PSTRING
MOV     BX,0246H
CALL    SET_DI
CALL    SHOW_SHORT_DATE


MOV     BX,0202H
CALL    SET_DI
CALL    SHOW_CODE                       ;display ASCII or EBCDIC
;               CALL    SHOW_PROTOCAL                   ;display value set
;
MOV     ATTRIBUTE,15                      ;normal display
MOV     BX,0445H
CALL    SET_DI
MOV     SI,OFFSET SHOW_COM                ;status com on/off
CALL    PSTRING
CALL    SHOW_OFF


MOV     BX,0645H
CALL    SET_DI
```

```
MOV     SI,OFFSET SHOW_FILE             ;status file on/off
CALL    PSTRING
CALL    SHOW_F3

MOV     BX,0845H
CALL    SET_DI
MOV     SI,OFFSET SHOW_PRINT            ;status printer on/off
CALL    PSTRING
CALL    SHOW_OFF
;
MOV     CURSOR_LINE,0A46H
MOV     CX,6                            ;number row
MOV     SI,OFFSET TITEL_MODEM
MOV     NUM_CHAR,6
CALL    WRITE_ST_HOR
;
MOV     BX,1646H
CALL    SET_DI
MOV     SI,OFFSET SHOW_HEX
CALL    PSTRING
CALL    SHOW_F1_HEX
MOV     ATTRIBUTE,12                    ;light red
MOV     BX,1746H
CALL    SET_DI
MOV     SI,OFFSET F10_MENU
CALL    PSTRING
;
RET
WRITE_HEAD      ENDP
;
```

```
CALL    PSTRING
MOV     ATTRIBUTE,15                    ;attr normal
RET
SHOW_F1_HEX     ENDP
;
PUBLIC  SHOW_F3
SHOW_F3         PROC
MOV     BX,064BH
CALL    SET_DI
CMP     F_SAVE,1                        ;1=on FF= off
JE      F3_ON
CALL    SHOW_OFF
JMP     SHORT END_SHOW_F3
F3_ON:          CALL    SHOW_ON
END_SHOW_F3:    RET
SHOW_F3         ENDP

;
PUBLIC  SHOW_PROTOCAL
SHOW_PROTOCAL   PROC
MOV     ATTRIBUTE,14                    ;color yellow
MOV     BX,0209H
CALL    SET_DI
MOV     SI,OFFSET WORD%
CALL    PSTRING
RET
SHOW_PROTOCAL   ENDP
;
;
PUBLIC  SHOW_CODE
```

```
SHOW_CODE       PROC
CMP     DISP_CODE,01H                   ;1=ascii FF=ebcdic
JE      DISP_ASCII
MOV     SI,OFFSET EBCDIC
CALL    PSTRING
JMP     SHORT END_SHOW_CODE
DISP_ASCII:     MOV     SI,OFFSET ASCII
CALL    PSTRING
END_SHOW_CODE:  RET
SHOW_CODE       ENDP



;
;
PUBLIC  PSTRING
PSTRING         PROC
PSLOOP:
MOV     AL,BYTE PTR[SI]
CMP     AL,24H
JE      END_LOOP
INC     SI
CALL    PCHAR
JMP     PSLOOP
END_LOOP:       RET
PSTRING         ENDP
;
;
;*****************************
;*   write RX1,TX1           *
;*****************************
```

```
;
LINE_HOR        PROC
MOV     DH,4H                   ;row
MOV     CX,10
HOR_LOOP:
PUSH    CX
MOV     DL,1H                   ;COL
CALL    GOTO_XY
PUSH    DX
@DISPSTR TX
POP     DX
MOV     DL,1H
INC     DH
CALL    GOTO_XY
PUSH    DX
@DISPSTR RX
POP     DX
INC     DH
POP     CX
LOOP    HOR_LOOP
RET
LINE_HOR        ENDP
;
;

PUBLIC  WRITE_CHAR_N_TIMES
;-----------------------------------------------------------------------
;  This procedure writes more than one copy of a character
;
;       AL      Character code
```

```
;       CX      Number of times to write the character
;
; Uses:         WRITE_CHAR
;-------------------------------------------------------------------------
WRITE_CHAR_N_TIMES      PROC
PUSH    CX
N_TIMES1:
CALL    PCHAR                   ;Write one copy of the character
LOOP    N_TIMES1                ;Keep writing for N times
POP     CX
RET
WRITE_CHAR_N_TIMES      ENDP
;
;
    PUBLIC  SCREEN1
SCREEN1     PROC
    MOV     ATTRIBUTE,1         ;blue
    MOV     AL,219
    MOV     CX,80
    CALL    WRITE_CHAR_N_TIMES
    RET
SCREEN1     ENDP
;
;
    PUBLIC  SCREEN2
SCREEN2     PROC
    MOV     BX,0100H
    CALL    SET_DI
    MOV     ATTRIBUTE,15        ;white(high internsitty)
    MOV     AL,176
```

```
    MOV     CX,1840
    CALL    WRITE_CHAR_N_TIMES
    RET
SCREEN2     ENDP
;
;

PUBLIC  SHOW_ON
SHOW_ON         PROC
MOV     ATTRIBUTE,75H               ;mag-wht
MOV     SI,OFFSET ON
CALL    PSTRING
MOV     ATTRIBUTE,15                ;normal
RET
SHOW_ON         ENDP
;
PUBLIC  SHOW_OFF
SHOW_OFF        PROC
MOV     ATTRIBUTE,75H
MOV     SI,OFFSET OFF
CALL    PSTRING
MOV     ATTRIBUTE,15
RET
SHOW_OFF        ENDP

;
;
;read keyboard character
    PUBLIC  GETKEY
GETKEY      PROC    NEAR
```

```
    MOV     AH,0
    INT     16H
    CMP     AX,011BH
    JNE     NO_ESC
    STC
NO_ESC:     RET
GETKEY      ENDP
;
;
;-------------------------------------------------------------------
; BIN2ASCII converts a binary digit less than 100 into its text form.
; Entry: ES:DI - buffer address
;        AL    - byte value
;        BL    - 0 (print zero before), 30h (not print zero)
;-------------------------------------------------------------------
bin2ascii   proc    near
    aam                     ;convert AL to BCD value in AX
    add     ax,3030h        ;binary to ASCII
    cmp     ah,bl           ;not print zero?
    je      bin1            ;yes
    xchg    ah,al           ;get first digit in AL
    stosb                   ;print first
    xchg    ah,al           ;restore
bin1:       stosb                   ;print second
    ret
bin2ascii   endp
;
;
;
;If TimeDate is already installed, ES points to the code segment of Ti
```

```
;switch on,off communication  F2
PUBLIC  XON_XOFF
XON_XOFF        PROC
MOV     BX,044BH                    ;set cursor
CALL    SET_DI
NEG     F2_XON_XOFF                 ;1= xoff
CMP     F2_XON_XOFF,1               ;FF=xon
JE      XOFF1
CALL    SHOW_ON
CALL    XON
RET
XOFF1:          CALL    SHOW_OFF
CALL    XOFF
RET
XON_XOFF        ENDP
;
;switch Char & hex to display screen
PUBLIC  SHOW_F1_HEX
SHOW_F1_HEX     PROC
MOV     ATTRIBUTE,75H               ;attr mag-wht
MOV     BX,164AH                    ;set row,col
CALL    SET_DI
CMP     F1_HEX,1                    ;FF = hex
JE      SW_CHAR
MOV     SI,OFFSET TITLE_HEX
CALL    PSTRING
MOV     ATTRIBUTE,01                ;attr normal
RET
SW_CHAR:        MOV     SI,OFFSET TITLE_CHAR
```

```
        PUBLIC  DATE_SET
    ;
    DATE_SET    PROC
        PUSH    DI
        PUSH    SI
        PUSH    ES
        PUSH    DS
        POP     ES
        mov     ah,2Ah          ;get system date
        int     21h
        mov     ax,cx           ;save year
        sub     cx,1900         ;subtract 1900 from year
        add     ax,543          ;convert to thai
        sub     ax,2500
        mov     thai_yr,al      ;store all data
        mov     month,dh
        mov     day,dl
        mov     year,cl
        cld
    ;
        dec     dh              ;find table offset of month
        mov     al,10
        mul     dh              ;(month-1)*10
        mov     si,ax           ;and it by SI
        add     si,offset month_text
        mov     cl,[si]         ;get length
        inc     si
        xor     ch,ch
        lea     di,long_date    ;point ES:DI to long date
        rep     movsb           ;transfer to buffer
```

```
        mov     al,20h          ;and space
        stosb
        mov     al,dl           ;get day
        mov     bl,30h          ;not print zero
        call    bin2ascii       ;add day to buffer
        lea     si,text1        ;add '19'
        mov     cx,4
        rep  movsb
        mov     al,year
        mov     bl,0
        call    bin2ascii       ;add year to buffer
        mov     al,24H
        stosb                   ;end of string
;
        lea     di,short_date   ;point ES:DI to short date
        mov     al,month        ;get month
        mov     bl,30h
        call    bin2ascii       ;to buffer
        mov     al,'-'
        stosb
        mov     al,day          ;get day
        mov     bl,30h
        call    bin2ascii       ;to buffer
        mov     al,'-'
        stosb
        mov     al,year         ;get year
        mov     bl,0
        call    bin2ascii       ;to buffer
        mov     al,24H          ;end of string
        stosb
```

```
;           lea     di,thai_date    ;point ES:DI to thai data
;           mov     al,day          ;get day
;           mov     bl,30h          ;with not print zero
;           call    bin2ascii       ;to buffer
;           mov     al,20h          ;add ' '
;          stosb
;           mov     dh,month        ;get month
;           dec     dh              ;find table offset of month
;           mov     al,11
;           mul     dh              ;(month-1)*11
;           mov     si,ax           ;and it by SI
;           add     si,offset month_thai
;           mov     cl,[si]         ;get length
;           inc     si
;           xor     ch,ch
;           rep     movsb           ;transfer to buffer
;           lea     si,text2
;           mov     cx,8            ;length
;           rep     movsb
;           mov     al,thai_yr      ;get year
;           mov     bl,0
;           call    bin2ascii       ;to buffer
;           mov     al,0H          ;end of string
;           stosb
    POP     ES
    POP     SI
    POP     DI
    RET
DATE_SET    ENDP
```

```
;
    PUBLIC  SHOW_LONG_DATE
SHOW_LONG_DATE   PROC
    MOV     SI,OFFSET LONG_DATE
    CALL    PSTRING
    RET
SHOW_LONG_DATE  ENDP
;
    PUBLIC  SHOW_SHORT_DATE
SHOW_SHORT_DATE   PROC
    MOV     SI,OFFSET SHORT_DATE
    CALL    PSTRING
    RET
SHOW_SHORT_DATE  ENDP
;
;           PUBLIC  SHOW_THAI_DATE
;SHOW_THAI_DATE   PROC
;           MOV     SI,OFFSET THAI_DATE
;           CALL    PSTRING
;           RET
;SHOW_THAI_DATE  ENDP
;
;
PUBLIC  RUN_END
RUN_END         PROC                                    ;set window
MOV     UL_COL,45
MOV     UL_ROW,14
MOV     LR_COL,75
MOV     LR_ROW,20
MOV     ATTRIBUTE,12                    ;light red
```

```
CALL    SAVSCRN                 ;open windows
;               MOV     ATTRIBUTE,01H           ;attr  narmal
MOV     CURSOR_LINE,1030H       ;row,col
MOV     CX,3                    ;number row
MOV     SI,OFFSET TEXT_END
MOV     NUM_CHAR,27             ;number character per line
CALL    WRITE_ST_HOR
RET

RUN_END         ENDP
;
;
;-------------------------------------------------;
; INPUT  SI OFFSET   = 'STRING',24H                ;
;        CURSOR_LINE = ROW,COL                     ;
;        NUM_CHAR    = NUMBER CHARACTER PER LINE   ;
;              CX    = NUMBER LINE ALL             ;
;-------------------------------------------------;

WRITE_ST_HOR    PROC

MOV     BX,CURSOR_LINE              ;row,col
CALL    SET_DI
VIDEO_1:        PUSH    SI
CALL    PSTRING
MOV     BX,CURSOR_LINE
INC     BH
MOV     CURSOR_LINE,BX
CALL    SET_DI
POP     SI
```

```
ADD     SI,NUM_CHAR                 ;si + number char per line
LOOP    VIDEO_1
RET
WRITE_ST_HOR    ENDP
;
;DISPLAY MAIN MENU BOX
PUBLIC  SHOW_MENU
SHOW_MENU       PROC
MOV     ATTRIBUTE,1EH               ;yel-blu
MOV     CURSOR_LINE,0309H
MOV     CX,17                       ;number all row in menu
MOV     SI,OFFSET TEXT_MENU
MOV     NUM_CHAR,28                 ;number char per line
CALL    WRITE_ST_HOR

PUSH    WORD PTR ATTRIBUTE
MOV     DX,050BH
CALL    GOTO_XY
MOV     ATTRIBUTE,7CH               ;lrd-wht
CALL    COLOR
POP     WORD PTR ATTRIBUTE

RET

SHOW_MENU       ENDP
;
;
;
PUBLIC  MENU_PROTOCAL
MENU_PROTOCAL  PROC
```

```
CALL    SCREEN2
;               MOV     ATTRIBUTE,0FH           ;bwt-blk
MOV     CURSOR_LINE,0305H       ; ROW , CUL
MOV     CX,13                   ;number all row in menu
MOV     SI,OFFSET MENU_BAUD
MOV     NUM_CHAR,16             ;NUMBER CHARACTER PER LINE
CALL    WRITE_ST_HOR
;
MOV     CURSOR_LINE,0318H
MOV     CX,8                    ;number all row in menu
MOV     SI,OFFSET MENU_PARITY
MOV     NUM_CHAR,14
CALL    WRITE_ST_HOR
;
MOV     CURSOR_LINE,0329H
MOV     CX,7                    ;number all row in menu
MOV     SI,OFFSET MENU_STOP
MOV     NUM_CHAR,15
CALL    WRITE_ST_HOR
;
MOV     CURSOR_LINE,033BH
MOV     CX,7                    ;number all row in menu
MOV     SI,OFFSET MENU_LENGHT
MOV     NUM_CHAR,17
CALL    WRITE_ST_HOR
;
MOV     CURSOR_LINE,0D19H
MOV     CX,11
MOV     SI,OFFSET SELECT        ;select Buad/parity/stop/word
```

```
MOV     NUM_CHAR,39
CALL    WRITE_ST_HOR

;               PUSH    WORD PTR ATTRIBUTE
MOV     DX,103BH
CALL    GOTO_XY
MOV     DL,SBAUD
CALL    WRITE_CHAR              ;display value baud rate

MOV     DX,113BH
CALL    GOTO_XY
MOV     DL,SPARITY
CALL    WRITE_CHAR              ;display value parity

MOV     DX,123BH
CALL    GOTO_XY
MOV     DL,SSTOPBIT
CALL    WRITE_CHAR              ;display value stopbits

MOV     DX,133BH
CALL    GOTO_XY
MOV     DL,SWORD
CALL    WRITE_CHAR              ;display value word lenght
;               POP     WORD PTR ATTRIBUTE
RET
MENU_PROTOCAL   ENDP
;
;
;
;************************
```

```
;* set new protocol    *
;************************
;the individual submenus for selection of BAUD RATE,PARITY,
;STOP BITS and WORD LENGTH are posted.the value entered is
;tested for validity and then ORed into the set-up bte.
;Selected values are also inserted into the parameters area
;for display.
;
;-------------------------------------------------------------------

SET_BAUD        PROC
BAUD_RATES:
MOV     DX,103BH                ;row=16, col=59
CALL    GOTO_XY                 ;set cursor position
CALL    GETKEY                  ;get single key input in AL
JC      EXIT_BAUD               ;enter ECS = CNCL
OR      AL,AL
JZ      EXIT_BAUD               ;no charracter read
CMP     AX,0F09H                ;key tab
JE      EXIT_BAUD

;display input at cursor
;valid range is '1'  to '8'
CMP     AL,'1'                  ;carry set if less than 1
JC      BAUD_BEEP               ;repeat selection
CMP     AL,'9'                  ;no carry if larger then 8
JC      BAUD_OK
BAUD_BEEP:      CALL    BEEP
JMP     BAUD_RATES
```

```
BAUD_OK:        MOV     SBAUD,AL                ;save ascii in baud
MOV     DL,AL
CALL    WRITE_CHAR              ;display ascii
CLC                             ;clear carry

EXIT_BAUD:      RET
SET_BAUD        ENDP
;
;---------------------------------------------------------------
;set value parity bit
;
SET_PARITY      PROC
PARITY:
MOV     DX,113BH
CALL    GOTO_XY
CALL    GETKEY                  ;single key input in AL
JC      EXIT_PARITY
OR      AL,AL
JZ      EXIT_PARITY
CMP     AX,0F09H                ;key tab
JE      EXIT_PARITY

;Valid input range is '1' to '3'
CMP     AL,'1'                  ;Carry set if less than 1
JC      PARITY_BEEP             ;repeat selection
CMP     AL,'4'                  ;No carry if larger than 3
JC      PARITY_OK
PARITY_BEEP:    CALL    BEEP
JMP     PARITY
```

```
PARITY_OK:      MOV     SPARITY,AL              ;save ascii
MOV     DL,AL
CALL    WRITE_CHAR              ;display ascii
CLC
EXIT_PARITY:    RET
SET_PARITY      ENDP
;
;-----------------------------------------------------------------------
;set stop bit key
SET_STOPBIT     PROC
STOPBITS:
MOV     DX,123BH
CALL    GOTO_XY
CALL    GETKEY                  ;get single key input in AL
JC      EXIT_STOPBIT
OR      AL,AL
JZ      EXIT_STOPBIT
CMP     AX,0F09H                ;key tab
JE      EXIT_STOPBIT

;Valid input range is '1' or '2'
CMP     AL,'1'                  ;carry set if less than 1
JC      STOPBITS_BEEP           ;repeat selection
CMP     AL,'3'                  ;No carry if larger than 2
JC      STOPBITS_OK
STOPBITS_BEEP:  CALL    BEEP
JMP     STOPBITS

STOPBITS_OK:    MOV     SSTOPBIT,AL             ;save ascii
MOV     DL,AL
```

```
    CALL    WRITE_CHAR              ;display ascii
    CLC
    EXIT_STOPBIT:   RET
    SET_STOPBIT     ENDP
    ;
    ;
    SET_WORD        PROC
    WORD_LENGTH:
    MOV     DX,133BH
    CALL    GOTO_XY
   CALL    GETKEY                   ;single key input in AL
    JC      EXIT_WORD
    OR      AL,AL
    JZ      EXIT_WORD
    CMP     AX,0F09H                ;key   tab
    JE      EXIT_WORD

    ;valid input range is '1' or '2'
    CMP     AL,'1'                  ;carry set if less than 1
    JC      WORD_BEEP
    CMP     AL,'3'                  ;No carry if larger than 2
    JC      WORD_OK
    WORD_BEEP:      CALL    BEEP
    JMP     WORD_LENGTH

    WORD_OK:        MOV     SWORD,AL
    MOV     DL,AL
    CALL    WRITE_CHAR              ;display ascii
    CLC
    EXIT_WORD:      RET
```

```
SET_WORD        ENDP
;
;
OK              PROC
PUSH    WORD PTR NUM_CHAR_COLOR
MOV     DX,1520H                ;position cursor row,col
CALL    GOTO_XY
MOV     ATTRIBUTE,1EH           ;yel-blu
MOV     NUM_CHAR_COLOR,6        ;number at display
CALL    COLOR                   ;exchange attribute

BEGIN_OK:       CALL    GETKEY
JC      EXIT_OK                 ; ESC=cancel
OR      AL,AL
JZ      EXIT_OK                 ;special key
CMP     AX,0F09H                ;key tab
JE      EXIT_OK
PUSH    DX
CMP     AX,1C0DH                ;value enter "OK".
JNE     BEGIN_OK
CALL    PROCESS_PROTOCAL
CALL    SET_PROTOCAL
POP     DX
STC                             ;set carry
EXIT_OK:
MOV     ATTRIBUTE,15
CALL    COLOR
POP     WORD PTR NUM_CHAR_COLOR
RET
OK              ENDP
```

```
;
CANCEL          PROC
PUSH    WORD PTR NUM_CHAR_COLOR
MOV     DX,152CH                ;position cursor row,col
CALL    GOTO_XY
MOV     ATTRIBUTE,1EH           ;yel-blu
MOV     NUM_CHAR_COLOR,10       ;number at display
CALL    COLOR                   ;exchangtribute

BEGIN_CNCL:     CALL    GETKEY
JC      EXIT_CNCL               ; ESC=cancel
OR      AL,AL
JZ      EXIT_CNCL               ;special key
CMP     AX,0F09H                ;key tab
JE      EXIT_CNCL

CMP     AX,1C0DH                ;value enter "OK".
JNE     BEGIN_CNCL
STC                             ;set carry for cancel
EXIT_CNCL:      MOV     ATTRIBUTE,15
CALL    COLOR
POP     WORD PTR NUM_CHAR_COLOR
RET
CANCEL          ENDP

;
;****************************************
;   MAIN EDIT VALUE SETUP PROTOCOL     *
```

```
;*****************************************
EDIT_SETUP_PTA  PROC
PUSHA
PUSH    ES

BAUD_EDIT:      CALL    SET_BAUD
JC      EXIT_EDIT
CMP     AX,4800H                ;key arrow UP
JE      CANCEL_EDIT

PARITY_EDIT:    CALL    SET_PARITY
JC      EXIT_EDIT
CMP     AX,4800H                ;key arrow up
JE      BAUD_EDIT

STOP_EDIT:      CALL    SET_STOPBIT
JC      EXIT_EDIT
CMP     AX,4800H                ;key arrow up
JE      PARITY_EDIT

WORD_EDIT:      CALL    SET_WORD
JC      EXIT_EDIT
CMP     AX,4800H
JE      STOP_EDIT

OK_EDIT:        CALL    OK                      ;process protocal
JC      EXIT_EDIT
CMP     AX,4800H                ;key arrow up
JE      WORD_EDIT
```

```
CANCEL_EDIT:    CALL    CANCEL
JC      EXIT_EDIT
CMP     AX,4B00H                ;key arrow left <--
JE      OK_EDIT
JMP     BAUD_EDIT

EXIT_EDIT:      POP     ES
POPA
RET

EDIT_SETUP_PTA  ENDP

;-----------------------------------------------------------------------

PUBLIC  SETUP_PROTOCAL
SETUP_PROTOCAL  PROC
MOV     UL_COL,09
MOV     UL_ROW,03
MOV     LR_COL,37
MOV     LR_ROW,20
CALL    SAVSCRN                 ;save screen
CALL    MENU_PROTOCAL           ;display menu protocol
CALL    CURSOR_ON               ;display cursor
PUSH    WORD PTR ATTRIBUTE
MOV     ATTRIBUTE,07H           ;attribute normal
CALL    EDIT_SETUP_PTA          ;edit value protocol
POP     WORD PTR ATTRIBUTE
CLC                             ;clear carry
CALL    SCREEN2
```

```
CALL    GETSCRN                 ;return screen
CALL    CURSOR_OFF
RET
SETUP_PROTOCAL  ENDP
;
;
;-------------------------------------------------------------
;PROCESS VALUE BAUD,PARITY,STOPBIT,WORD
;input is in valid range.
PUBLIC  PROCESS_PROTOCAL
PROCESS_PROTOCAL    PROC
PUSH    ES
MOV     AL,SBAUD                ;get ascii
SUB     AL,30H                  ;ASCII to binary
SUB     AL,1                    ;to range 0 to 7

;select 4 byte baud rate message and place in protocol message
;area for display of installed parameters
PUSH    AX
;Multiply binary code by 4 to compute displacement into table
MOV     CL,4
MUL     CL                      ;AL is now in range 0 to 28(offset)
MOV     SI,OFFSET BAUD$$        ;SI-->baud rates message
MOV     AH,0
ADD     SI,AX                   ;SI-->4 ASCII charaters of baud
;move baud rate message into protocol display area
MOV     DI,OFFSET BAUD%
MOV     CX,4                    ;4 byte to move
CLD                             ;forward
PUSH    DS
```

```
PUSH    DS
POP     ES
REP     MOVSB
;
POP     AX                      ;restore control byte
MOV     CL,5                    ;Shift bit conter
SHL     AL,CL                   ;Baud rate bits left 5 bits
MOV     SETUP_BYTE,AL           ;store control word

;-----------------------------------------------------------------
;process value parity

MOV     AL,SPARITY              ;get value parity
SUB     AL,30H                  ;ascii to binary in range 1-3
PUSH    AX                      ;save control bits
SUB     AL,1                    ;to range 0 to 2
MOV     CL,4
MUL     CL                      ;multiply to obtain offset
MOV     SI,OFFSET PAR$$
MOV     AH,0
ADD     SI,AX
MOV     DI,OFFSET PAR%
MOV     CX,4
CLD                             ;forward
PUSH    DS
POP     ES
REP     MOVSB
;
POP     AX                      ;restore control bits
MOV     CL,3                    ;shift bit counter
```

```
SHL     AL,CL                   ;Shift left 3 bits
OR      SETUP_BYTE,AL           ;OR with stored baud rate

;-----------------------------------------------------------------------
;process value stopbits

MOV     AL,SSTOPBIT             ;get value stopbit
;input is in valid renge.Move input digit to display raea
MOV     SI,OFFSET STOP%         ;pointer to display area
MOV     BYTE PTR[SI],AL         ;insert digit
;convert input digit to binary 0 or 1
SUB     AL,31H                  ;ascii to binary minus 1
;Shift left 2 bits
MOV     CL,2
SHL     AL,CL                   ;shift digits
OR      SETUP_BYTE,AL           ;OR with stored parameters
;-----------------------------------------------------------------------
;process value word

MOV     AL,SWORD                ;get value word
PUSH    AX                      ;save input
;input in valid range.Add 6 and move input to display area
ADD     AL,6                    ;1 = 7 and 2 = 8
MOV     SI,OFFSET WORD%         ;DISPLAY AREA
MOV     BYTE PTR [SI],AL
POP     AX                      ;restore input digit
SUB     AL,30H                  ;convert to binary 1 or 2
INC     AL
OR      SETUP_BYTE,AL           ;OR with stores parameters
;
```

```
;install new parameters
;                CALL    SET_PROTOCAL            ;setup protocal new
POP     ES
RET
PROCESS_PROTOCAL   ENDP
;
;-------------------------------------------------------------------
PUBLIC  PHANTOM_UP
PHANTOM_UP      PROC
PUSH    WORD PTR ATTRIBUTE
MOV     DX,PHANTOM_CURSOR       ;set row,col
CALL    GOTO_XY
MOV     ATTRIBUTE,1EH
CALL    COLOR
;
CMP     NUM_CALL,0
JE      PHANTOM_UP1             ;num_call = 0 goto
DEC     NUM_CALL
MOV     DX,PHANTOM_CURSOR
DEC     DH
DEC     DH
JMP     SHORT PHANTOM_UP2
PHANTOM_UP1:
MOV     NUM_CALL,6
MOV     DX,110BH
PHANTOM_UP2:
MOV     PHANTOM_CURSOR,DX
CALL    GOTO_XY
MOV     ATTRIBUTE,7CH
CALL    COLOR
```

```
POP     WORD PTR ATTRIBUTE
RET
PHANTOM_UP      ENDP
;
;
PUBLIC  PHANTOM_DOWN
PHANTOM_DOWN    PROC
PUSH    WORD PTR ATTRIBUTE
MOV     DX,PHANTOM_CURSOR           ;set row,col
CALL    GOTO_XY
MOV     ATTRIBUTE,1EH
CALL    COLOR
INC     NUM_CALL
CMP     NUM_CALL,7
JAE     PHANTOM_DOWN1               ;if num_call >= 6 goto
MOV     DX,PHANTOM_CURSOR
INC     DH
INC     DH
JMP     SHORT PHANTOM_DOWN2
PHANTOM_DOWN1:
MOV     NUM_CALL,0
MOV     DX,050BH
PHANTOM_DOWN2:
MOV     PHANTOM_CURSOR,DX
CALL    GOTO_XY
MOV     ATTRIBUTE,7CH
CALL    COLOR
POP     WORD PTR ATTRIBUTE
RET
PHANTOM_DOWN    ENDP
```

```
;
;
PUBLIC  WAIT_KB
WAIT_KB         PROC
WAIT1:          MOV     AH,1
INT     16H
JZ      WAIT1
RET
WAIT_KB         ENDP
;
;
PUBLIC  SHOW_TITEL1
SHOW_TITEL1     PROC
MOV     ATTRIBUTE,2             ;color green
MOV     UL_COL,08H
MOV     UL_ROW,03H
MOV     LR_COL,70
MOV     LR_ROW,18
CALL    SAVSCRN
MOV     CURSOR_LINE,050FH       ;ROW,COL
MOV     CX,8                    ;number all row
MOV     SI,OFFSET TITEL1
MOV     NUM_CHAR,50             ;number character per line
CALL    WRITE_ST_HOR
;
MOV     CURSOR_LINE,0F2EH       ;row,col
MOV     CX,3
MOV     SI,OFFSET TITEL2
MOV     NUM_CHAR,24             ;number char/line
CALL    WRITE_ST_HOR
```

```
LEA     SI,TRKY_FREQ
LEA     BP,TRKY_TIME
CALL    PLAY

CALL    GETSCRN
RET
SHOW_TITEL1     ENDP
;
;-----------------------------------------------------------------
PUBLIC  WIN_SMALL
WIN_SMALL       PROC
MOV     ATTRIBUTE,1EH           ;yel-blu
MOV     UL_COL,05
MOV     UL_ROW,03
MOV     LR_COL,75
MOV     LR_ROW,18
CALL    SAVSCRN                 ;save screen
RET
WIN_SMALL       ENDP


PUBLIC  SHOW_MENU_CONFIG
SHOW_MENU_CONFIG PROC
CALL    WIN_SMALL
;               MOV     ATTRIBUTE,1EH           ;high display attr
MOV     BX,0517H
CALL    SET_DI                  ;set cursor position
MOV     SI,OFFSET SHOW_HEAD_CONFIG
CALL    PSTRING                 ;print string
```

```
MOV     ATTRIBUTE,7             ;normal display attr
MOV     CURSOR_LINE,070AH       ;START WRITE ROW,COL
MOV     CX,7                    ;number all row
MOV     SI,OFFSET SHOW_CONFIG
MOV     NUM_CHAR,30             ;number character per line
CALL    WRITE_ST_HOR
MOV     BX,1019H
CALL    SET_DI
MOV     SI,OFFSET SHOW_CONFIG1  ;save process cancel
CALL    PSTRING
CALL    DISP_VALUE_CONFIG
CALL    CURSOR_ON
CALL    EDIT_CONFIG
CALL    CURSOR_OFF
CALL    GETSCRN
RET
SHOW_MENU_CONFIG  ENDP
;
;
;
;display value in configuration program to show_menu_config
DISP_VALUE_CONFIG   PROC

 ;              MOV     ATTRIBUTE,0FH           ; hight
MOV     BX,0728H
MOV     DX,BX
PUSH    DX
CALL    SET_DI                  ;set cursor position
MOV     ATTRIBUTE,75H           ;mag-wht
CALL    SHOW_CODE               ;display code
```

```
POP     DX

INC     DH
PUSH    DX
MOV     BX,DX
CALL    SET_DI
CMP     AUTO_SET,01H            ;1=ON, FF= OFF
JE      AUTO_ON
CALL    SHOW_OFF                ;display auto set protocol
JMP     SHORT END_AUTO
AUTO_ON:        CALL    SHOW_ON
END_AUTO:
POP     DX
INC     DH
PUSH    DX
MOV     BX,DX
CALL    SET_DI
;               MOV     ATTRIBUTE,1EH           ;ATTR HIGHT use in pch
MOV     SI,OFFSET FILENAME      ;display file name
CALL    DISP_ST_0
POP     DX

INC     DH
PUSH    DX
CALL    GOTO_XY
;               MOV     ATTRIBUTE,0FH           ;hight use in write_ch
MOV     DL,TERMINAL_IN          ;display port terminal comm
CALL    WRITE_CHAR
POP     DX
```

```
        INC     DH
        PUSH    DX
        CALL    GOTO_XY
        MOV     DL,TERMINAL_OUT         ;display port terminal comm
        CALL    WRITE_CHAR
        POP     DX

        INC     DH
        PUSH    DX
        CALL    GOTO_XY
        MOV     DL,SAND_COM             ;display port sand comm
        CALL    WRITE_CHAR
        POP     DX

        INC     DH
        CALL    GOTO_XY
        MOV     DL,RECEIVE_COM          ;display port receive comm
        CALL    WRITE_CHAR
        ;       MOV     ATTRIBUTE,07H           ;display narmal attr
        RET
DISP_VALUE_CONFIG    ENDP
;
;
ASCII_EBCDIC    PROC
        MOV     DX,0728H
        CALL    GOTO_XY
BEGIN_AE1:      CALL    GETKEY
        JC      EXIT_ASCII_EBCDIC
        OR      AL,AL
        JZ      EXIT_ASCII_EBCDIC
```

```
CMP     AX,1C0DH
JE      SHORT OK_AE
CALL    BEEP
JMP     BEGIN_AE1
OK_AE:          MOV     BX,0728H
CALL    SET_DI
NEG     DISP_CODE                       ;1=ON, FF=OFF
MOV     ATTRIBUTE,75H                   ;mag-wht
CALL    SHOW_CODE                       ;display code ascii or ebcdic
JMP     BEGIN_AE1
EXIT_ASCII_EBCDIC:
RET
ASCII_EBCDIC    ENDP
;
;
;
SET_AUTO        PROC
MOV     DX,0828H
CALL    GOTO_XY
BEGIN_AUTO:     CALL    GETKEY
JC      EXIT_SET_AUTO
OR      AL,AL
JZ      EXIT_SET_AUTO
CMP     AX,1C0DH
JE      OK_AUTO
CALL    BEEP
JMP     BEGIN_AUTO
OK_AUTO:        MOV     BX,0828H
CALL    SET_DI
NEG     AUTO_SET                        ;1=ON, FF=OFF
```

```
CMP     AUTO_SET,01H                ;display auto setup
JE      AUTOON
CALL    SHOW_OFF
JMP     BEGIN_AUTO
AUTOON:         CALL    SHOW_ON
JMP     BEGIN_AUTO
EXIT_SET_AUTO:  RET
SET_AUTO        ENDP

;
;set file name for analysis
GET_FILENAME    PROC
PUSH    WORD PTR NUM_CHAR_COLOR
MOV     DX,0928H
CALL    GOTO_XY
BEGIN_FILE:     CALL    GETKEY
JC      EXIT_GETFILE
OR      AL,AL
JZ      EXIT_GETFILE
CMP     AX,0F09H                ;TAB KEY
JE      EXIT_GETFILE
CMP     AX,1C0DH                ;ENTER KEY
JE      OK_GETFILE
CALL    BEEP
JMP     BEGIN_FILE

OK_GETFILE:     MOV     BX,0928H                ;POSITION CURSOR
CALL    SET_DI
MOV     ATTRIBUTE,74H           ;REVERS HIGHT
MOV     AL,20H                  ;ASCII
```

```
MOV     CX,30
CALL    WRITE_CHAR_N_TIMES

MOV     ATTRIBUTE,78H           ;revers
@GETSTR NAMEBUF,0               ;get respone as ASCIIZ
MOV     DX,0928H
CALL    GOTO_XY                 ;set cursor position
MOV     NUM_CHAR_COLOR,30
MOV     ATTRIBUTE,07H           ;attr normal
CALL    COLOR
EXIT_GETFILE:
POP     WORD PTR NUM_CHAR_COLOR
RET
GET_FILENAME    ENDP
;
;-------------------------------------------------------------------
;
;*********************************
;*   chang value CX = 1 or 2     *
;*    input      CX              *
;*    output     CX              *
;*********************************
PUBLIC  KEY_1_2
KEY_1_2         PROC
BEGIN_KEY_1_2:  CALL    GETKEY
JC      EXIT_KEY_1_2            ;ESC
CMP     AX,1C0DH                ;enter key
JE      ENTER_KEY
OR      AL,AL                   ;function key
JZ      EXIT_KEY_1_2
```

```
CMP     AX,0F09H                ;tab key
JE      EXIT_KEY_1_2
CALL    BEEP
JMP     BEGIN_KEY_1_2
ENTER_KEY:
CMP     CL,31H
JE      CHANG_2
MOV     CL,31H                  ;set cl=1
JMP     SHORT DISP_1_2
CHANG_2:
MOV     CL,32H                  ;set cl=2
DISP_1_2:       MOV     DL,CL
MOV     ATTRIBUTE,1EH           ;attr high
CALL    WRITE_CHAR
JMP     BEGIN_KEY_1_2

EXIT_KEY_1_2:   RET
KEY_1_2         ENDP
;
;
PUBLIC  SAVE_CONFIG
SAVE_CONFIG     PROC
@MakFil FILENAME1,0
JC      EXIT_SAVE_CONFIG
MOV     HANDLE1,AX              ;save handle file
@WRITE  SBAUD,42,HANDLE1
@CLOSFIL HANDLE1
EXIT_SAVE_CONFIG:
RET
SAVE_CONFIG     ENDP
```

```
;
;
;Read file config for program
;
PUBLIC  READ_CONFIG
READ_CONFIG     PROC
@OpenFil FILENAME1,0
JC      EXIT_READ_CONFIG
MOV     HANDLE1,AX
@Read   SBAUD,42,HANDLE1
@ClosFil HANDLE1
EXIT_READ_CONFIG:
RET
READ_CONFIG     ENDP

;
;****************************************
;   MAIN EDIT VALUE SETUP CONFIG       *
;****************************************
EDIT_CONFIG     PROC

CODE_EDIT:
CALL    ASCII_EBCDIC
JC      EXIT_CONFIG
CMP     AX,4800H                ;key arrow UP
JE      RECEIVE_EDIT

AUTO_EDIT:      CALL    SET_AUTO
JC      EXIT_CONFIG
CMP     AX,4800H
```

```
JE      CODE_EDIT

FILENAME_EDIT:  CALL    GET_FILENAME
JC      EXIT_CONFIG
CMP     AX,4800H                ;key arrow up
JE      AUTO_EDIT

TM_IN_EDIT:     MOV     DX,0A28H
CALL    GOTO_XY
MOV     CL,TERMINAL_IN          ;read value port terminal
CALL    KEY_1_2
MOV     TERMINAL_IN,CL          ;write value port
JC      EXIT_CONFIG
CMP     AX,4800H                ;key arrow up
JE      FILENAME_EDIT

TM_OUT_EDIT:    MOV     DX,0B28H
CALL    GOTO_XY
MOV     CL,TERMINAL_OUT
CALL    KEY_1_2
MOV     TERMINAL_OUT,CL
JC      EXIT_CONFIG
CMP     AX,4800H
JE      TM_IN_EDIT

SAND_EDIT:      MOV     DX,0C28H
CALL    GOTO_XY
MOV     CL,SAND_COM
CALL    KEY_1_2
MOV     SAND_COM,CL
```

```
JC      EXIT_CONFIG
CMP     AX,4800H
JE      TM_OUT_EDIT

RECEIVE_EDIT:   MOV     DX,0D28H
CALL    GOTO_XY
MOV     CL,RECEIVE_COM
CALL    KEY_1_2
MOV     RECEIVE_COM,CL
JC      EXIT_CONFIG
CMP     AX,4800H
JE      SAND_EDIT
JMP     CODE_EDIT
EXIT_CONFIG:
RET
EDIT_CONFIG     ENDP

;-------------------------------------------------------------------
;
PHANTOM_UP_SMALL    PROC
CMP     NUM_CALL_SMALL,0            ;start point call
JE      EXIT_PHAN_UP_SMALL
MOV     ATTRIBUTE,78H               ;attr hight revers
CALL    COLOR
DEC     NUM_CALL_SMALL
DEC     DH
DEC     DH
CALL    GOTO_XY
MOV     ATTRIBUTE,07H
CALL    COLOR
```

```
EXIT_PHAN_UP_SMALL:
RET
PHANTOM_UP_SMALL    ENDP
;
;
PHANTOM_DOWN_SMALL  PROC
CMP     NUM_CALL_SMALL,2
JE      EXIT_PHAN_DOWN_SMALL
MOV     ATTRIBUTE,78H
CALL    COLOR
INC     NUM_CALL_SMALL
INC     DH
INC     DH
CALL    GOTO_XY
 MOV     ATTRIBUTE,07H
CALL    COLOR
EXIT_PHAN_DOWN_SMALL:
RET
PHANTOM_DOWN_SMALL  ENDP
;
;
DISPATCHER_SMALL    PROC
PUSH    BX
XOR     BX,BX
MOV     BL,NUM_CALL_SMALL
SHL     BX,1
CALL    WORD PTR [BX+DISPATCH_TABLE1]
POP     BX
RET
DISPATCHER_SMALL    ENDP
```

```
;
;
PUBLIC  SHOW_UTIL
SHOW_UTIL       PROC
PUSH    DX
MOV     UL_COL,45
MOV     UL_ROW,11
MOV     LR_COL,69
MOV     LR_ROW,19
CALL    SAVSCRN                         ;save screen
MOV     ATTRIBUTE,74H                   ;attr hight revers
MOV     CURSOR_LINE,0C2EH               ;write position
MOV     CX,7                            ;number line
MOV     SI,OFFSET UIL_TEXT
MOV     NUM_CHAR,24                     ;number character
CALL    WRITE_ST_HOR

MOV     DX,0D2FH                        ;set row,col
CALL    GOTO_XY
MOV     ATTRIBUTE,07                    ;attr normal
CALL    COLOR
 SMALL_MENU:
CALL    GETKEY
JC      EXIT_SMALL
CMP     AX,1C0DH                        ;enter
JNE     SPEC_KEY_SMALL
CALL    GETSCRN
CALL    DISPATCHER_SMALL                ;call procedure in menu
JMP     SHORT EXIT_SMALL1
SPEC_KEY_SMALL:
```

```
CMP     AX,4800H                    ;key arrow up
JNE     KB_SMALL1
CALL    PHANTOM_UP_SMALL
JMP     SMALL_MENU
KB_SMALL1:      CMP     AX,5000H                    ;key arrow down
JNE     SMALL_MENU
CALL    PHANTOM_DOWN_SMALL
JMP     SMALL_MENU
EXIT_SMALL:
;               CALL    WAIT_KB                     ;wait
CALL    GETSCRN                     ;get screen
EXIT_SMALL1:
MOV     NUM_CALL_SMALL,0            ;reset start point call
POP     DX
RET
SHOW_UTIL       ENDP
;-------------------------------------------------------------------
;*****************************
;*      SUB PROGRAM FILE     *
;*****************************
;
BOX_MENU_FILE   PROC
MOV     ATTRIBUTE,7BH           ;lcy-wht
MOV     UL_COL,9
MOV     UL_ROW,11
MOV     LR_COL,73
MOV     LR_ROW,16
CALL    BOX
RET
BOX_MENU_FILE   ENDP
```

```
;
;
PUBLIC  CLEAR_BOX
CLEAR_BOX       PROC
MOV     AH,06H
MOV     AL,0H
MOV     BH,07H                  ;attr normal
MOV     CH,11                   ;ul_row
MOV     CL,9                    ;ul_col
MOV     DH,16                   ;lr_row
MOV     DL,73                   ;lr_col
INT     10H
RET
CLEAR_BOX       ENDP
;
;
;display information disk dirve and path name

INFOR_DIR       PROC
MOV     ATTRIBUTE,7             ;attr
MOV     BX,0514H
CALL    SET_DI
MOV     SI,OFFSET TITEL_DRIVE
CALL    PSTRING
MOV     AL,NEW_DRIVE            ;0=A,1=B
ADD     AL,41H
CALL    PCHAR                   ;display drive

MOV     BX,0714H
CALL    SET_DI
```

```
PUSH    DI
MOV     CX,55
MOV     AL,20H
CALL    WRITE_CHAR_N_TIMES          ;clear screen path name
POP     DI
MOV     SI,OFFSET TITEL_PATH
CALL    PSTRING

MOV     AH,47H                      ;get path name
MOV     SI,OFFSET PATH_NAME
MOV     DL,NEW_DRIVE
INC     DL
INT     21H
MOV     SI,OFFSET PATH_NAME         ;display path name
CALL    DISP_ST_0
RET
INFOR_DIR       ENDP

;-------------------------------------------------------------------------
;set drive,path,load file for analyzer

PUBLIC  MENU_FILE
MENU_FILE       PROC
PUSHA
PUSHF
CALL    WIN_SMALL
MOV     ATTRIBUTE,74H               ;attr hight revers
MOV     CURSOR_LINE,0406H           ;write position
MOV     CX,5                        ;number line
MOV     SI,OFFSET TITEL_FILE
```

```
MOV     NUM_CHAR,10                     ;number character
CALL    WRITE_ST_HOR
MOV     ATTRIBUTE,1EH                   ;attr high
MOV     CURSOR_LINE,0508H
MOV     CX,3
MOV     SI,OFFSET CHAR_HIGTH
MOV     NUM_CHAR,2
CALL    WRITE_ST_HOR

BEGIN_MENU_FILE:
CALL    CURSOR_OFF
CALL    INFOR_DIR                       ;display drive,path
CALL    CLEAR_BOX
KB_BEGIN:       CALL    FLUSH
CALL    GETKEY
JNC     MENU_START
JMP     END_MENU_FILE

MENU_START:     CALL    UPCASE
CMP     AL,'D'
JNE     SELEC_P
CALL    BOX_MENU_FILE
MOV     DX,0D0AH
CALL    GOTO_XY
@DispStr TITEL_DRIVE
CALL    CURSOR_ON
CALL    GETKEY
JC      BEGIN_MENU_FILE
MOV     ATTRIBUTE,7
MOV     DL,AL
```

```
CALL    WRITE_CHAR
CALL    CURSOR_OFF
CALL    SET_DRIVE                       ;set drive
JMP     BEGIN_MENU_FILE
SELEC_P:
CMP     AL,'P'                          ;set path
JNE     SELEC_L
CALL    BOX_MENU_FILE
MOV     DX,0D0AH
CALL    GOTO_XY
CALL    CURSOR_ON
@DispStr TITEL_PATH
@GetStr  PATH_IN,0
CALL    CURSOR_OFF
@ChDir   PATH_NAME
JC      ERROR_PATH
JMP     BEGIN_MENU_FILE
ERROR_PATH:
MOV     ATTRIBUTE,7EH                   ;yel-wht
MOV     BX,0F20H
CALL    SET_DI
MOV     SI,OFFSET ERR_PATH
CALL    PSTRING
MOV     SI,OFFSET PATH_NAME
MOV     BYTE PTR [SI],00
CALL    BEEP
MOV     AL,0                            ;minutes
MOV     BH,1                            ;wait 1.5 second
MOV     BL,50
CALL    DELAY
```

```
JMP     BEGIN_MENU_FILE
SELEC_L:
CMP     AL,'L'                          ;load file for analysis
JE      LOAD_FILE
JMP     KB_BEGIN
LOAD_FILE:
CALL    BOX_MENU_FILE
MOV     DX,0D0AH
CALL    GOTO_XY
CALL    CURSOR_ON
@DispStr TITEL_FNAME
@GetStr  F_LNAME,0
CALL    CURSOR_OFF
CALL    LOAD_ANALY

;               JMP     BEGIN_MENU_FILE
END_MENU_FILE:
CALL    GETSCRN
POPF
POPA
RET
MENU_FILE       ENDP
;-------------------------------------------------------------------
;**********************************
;*   AUTOMATIC  SET UP PROTOCOL   *
;**********************************
PUBLIC  MAIN_AUTO
MAIN_AUTO       PROC
PUSHA
CLC
```

```
CMP     AUTO_SET,01H                    ;01h=on ffh= off
JNE     OK_MAIN_AUTO
CALL    SUB_SET_AUTO_2
JC      ERROR_MAIN_AUTO
CMP     AUTO_FLAGE,1                    ;1= OK
JE      OK_MAIN_AUTO
MOV     BX,0514H
CALL    SET_DI
MOV     ATTRIBUTE,79H
MOV     SI,OFFSET MESS_6                ;terminated by user
CALL    PSTRING
MOV     AL,0                            ;Minutes
MOV     BH,1                            ;wait 1.5 saconds
MOV     BL,50
CALL    DELAY
ERROR_MAIN_AUTO:
STC                                     ;error
OK_MAIN_AUTO:   POPA
RET
MAIN_AUTO       ENDP
;


;
SUB_SET_AUTO_1  PROC
MOV     SSTOPBIT,31H
SUB_AUTO_1_1:   MOV     SWORD,31H
SUB_AUTO_1_2:   CALL    PROCESS_PROTOCAL
MOV     DX,CARD_BASE_1
```

```
JE      SUB_AUTO_2_3
INC     SPARITY
JMP     SUB_AUTO_2_2

SUB_AUTO_2_3:   CMP     SBAUD,38H
JE      EXIT_SUB_AUTO_2
INC     SBAUD
JMP     SUB_AUTO_2_1
EXIT_SUB_AUTO_2:
STC
END_SUB_AUTO_2: RET
SUB_SET_AUTO_2  ENDP

;
;
PUBLIC  LOAD_ANALY
LOAD_ANALY      PROC
PUSHA
PUSHF
MOV     BX,0F11H
CALL    SET_DI
@OpenFil F_LBUFFER,0
JC      BADFILE
MOV     FHANDLE,AX                      ;save file handle
@GetFilSz FHANDLE                       ;get file size
OR      DX,DX                           ;larger than 64k ?
JNE     BIG
SUB     AX,15                           ;head file 15 byte
MOV     FSIZE,AX                        ;save file size
MOV     CX,4                            ;convert to paragraphs
```

```
SHR     AX,CL                   ;/16
CMP     AX,LBUFFER1
JLE     FILEREAD
BIG:            MOV     SI,OFFSET ERR1
CALL    PSTRING
DISP_ERR:       MOV     AL,0                    ;minutes
MOV     BH,1                    ;wait 1.5 saconds
MOV     BL,50
CALL    DELAY
;               CALL    GETSCRN
POPF
POPA
RET

BADFILE:        MOV     SI,OFFSET ERMES_1
CALL    PSTRING
JMP     SHORT DISP_ERR
FILEREAD:       @READ   SPEED,15,FHANDLE
MOV     PBUFFER1,0
PUSH    DS
@READ   BUFFER1,FSIZE,FHANDLE
POP     DS
JNC     READ_OK
JMP     SHORT BIG
READ_OK:
MOV     AX,FSIZE
MOV     PBUFFER1,AX
MOV     CURANT_ADDR,0A00H       ;position 1 page
CALL    GETSCRN
CALL    SHOW_SCREEN
```

```
@READ   BUFFER1,FSIZE,FHANDLE
POP     DS
JNC     READ_OK
JMP     SHORT BIG
READ_OK:
MOV     AX,FSIZE
MOV     PBUFFER1,AX
MOV     CURANT_ADDR,0A00H           ;position 1 page
CALL    GETSCRN
CALL    SHOW_SCREEN
CALL    SHOW_PROTOCAL
CALL    LOAD_BUFFER
L_KB1:
CALL    GETKEY
CMP     AX,4800H                    ;key arrow up
JNE     L_KB2
CALL    UP_ARROW_KEY
JMP     L_KB1
L_KB2:          CMP     AX,5000H                    ;key arrow down
JNE     L_KB3
CALL    DOWN_ARROW_KEY
JMP     L_KB1
L_KB3:          CMP     AX,3B00H                    ;F1
JNE     L_KB4
CALL    SW_DISPLAY
JMP     L_KB1
L_KB4:          CMP     AX,4400H                    ;F10
JE      L_EXIT
JMP     L_KB1
```

```
L_EXIT:
;               CALL    GETSCRN
POPF
POPA
RET
LOAD_ANALY      ENDP

;

TEST_RECEIVE_READY  PROC
CALL    FLUSH
TEST_1:         MOV     DL,0FDH
IN      AL,DX
JMP     SHORT   $+2
TEST    AL,01H
JNZ     TEST_READY
TEST    AL,0CH
JNZ     TEST_ERROR
MOV     DL,0F8H
IN      AL,DX
PUSH    DX
MOV     AH,1
INT     16H
POP     DX
JZ      TEST_1
MOV     AUTO_FLAGE,0FFH                 ;FF= terminated by user
CLC
RET
```

```
TEST_READY:     MOV     AUTO_FLAGE,1H                   ;1= OK ready
CLC
RET

TEST_ERROR:     STC
RET
TEST_RECEIVE_READY  ENDP
;-------------------------------------------------------------------------
;
PUBLIC  OS_SHELL
OS_SHELL        PROC
PUSHA
PUSH    ES
MOV     UL_COL,00H
MOV     UL_ROW,00H
MOV     LR_COL,79
MOV     LR_ROW,24
CALL    SAVSCRN                 ;save screen
MOV     DX,0000H
CALL    GOTO_XY
CALL    CLEAR_SCREEN
CALL    CURSOR_ON
CALL    SHELL                   ;shell to command.com
CALL    CURSOR_OFF
CALL    GETSCRN                 ;get screen
POP     ES
POPA
RET
OS_SHELL        ENDP
```

VIDEO-IO.ASM

```
INCLUDE BIOS.INC
EXTRN   ATTRIBUTE:BYTE
EXTRN   F1_HEX:BYTE
EXTRN   SBUFFER1:WORD
EXTRN   PBUFFER1:WORD
EXTRN   LBUFFER1:WORD

PUBLIC  ROW
PUBLIC  COL
ROW             DB  4
COL             DB  4

PUBLIC  ASCII_COM1
PUBLIC  ASCII_COM2
PUBLIC  CURANT_ADDR
ASCII_COM1      DW  0
ASCII_COM2      DW  0
TAIL            DW  0                                   ;save tail b
CURANT_ADDR     DW  0
PUBLIC  NUM_CHAR_COLOR
NUM_CHAR_COLOR  DW  21
;
PUBLIC  TITEL1
TITEL1          DB  '       P R O T O C O L   A N A L Y Z E R
DB  '                   Version 1.0                     ',24H
DB  '          -------------------------                ',24H
DB  '                                                   ',24H
DB  '             Copyright (C) 1992.                   ',24H
DB  '     The Communication Authority Of Thailand       ',24H
```

```
DB  ´                          and                          ´,24H
DB  "King Mongkut´s Institute of Technology Ladkrabang",24h
;
PUBLIC  TITEL2
PUBLIC  F_BUFFER
TITEL2          DB  ´Mr.Kitjawat Pandumrong ´,24H
DB  ´Mr.Nantawit Keawthaworn´,24H
DB  ´Mr.Apichat  Swankomtorn´,24H
;
OK              DB  ´OK´,24H
NO              DB  ´NO´,24H
F_BUFFER        DB  0                                   ;sing flage full

;-----------------------------------------------------------------------

SOH     DB      00H,3CH,22H,10H,08H,22H,1EH,00H,22H,22H
DB      3EH,22H,22H,00H,00H,00H                                 ;SOH
DB      00H,2CH,22H,10H,08H,22H,1EH,00H,22H,14H
DB      08H,14H,22H,00H,00H,00H                                 ;STX
DB      00H,3EH,20H,3CH,20H,20H,3EH,00H,22H,14H
DB      08H,14H,22H,00H,00H,00H                                 ;ETX
DB      00H,3EH,20H,3CH,20H,20H,3EH,00H,7EH,08H
DB      08H,08H,08H,00H,00H,00H                                 ;EOT
DB      00H,3EH,20H,3CH,20H,20H,3EH,00H,7EH,42H
DB      7AH,78H,7EH,01H,00H,00H                                 ;ENQ
DB      00H,3CH,42H,42H,7EH,42H,42H,00H,4CH,50H
DB      60H,58H,44H,00H,00H,00H                                 ;ACK
DB      00H,7CH,42H,42H,78H,42H,42H,7CH,00H,40H
DB      40H,40H,40H,40H,7EH,00H                                 ;BEL
```

```
DB      00H,7CH,42H,42H,78H,42H,42H,7CH,00H,7EH
DB      40H,7EH,02H,7EH,00H,00H                         ;BS
DB      00H,22H,22H,3EH,22H,22H,22H,00H,7EH,08H
DB      08H,08H,08H,00H,00H,00H                         ;HT
DB      00H,40H,40H,40H,40H,40H,7EH,00H,7EH,40H
DB      7CH,40H,40H,40H,00H,00H                         ;LF
DB      00H,42H,42H,42H,42H,22H,14H,08H,00H,7EH
DB      08H,08H,08H,08H,00H,00H                         ;VT
DB      00H,7EH,40H,40H,7CH,40H,40H,00H,7EH,40H
DB      7CH,40H,40H,40H,00H,00H                         ;FF
DB      00H,3EH,40H,40H,40H,40H,3EH,00H,7EH,42H
DB      7EH,50H,4CH,42H,00H,00H                         ;CR
DB      00H,7EH,40H,40H,7EH,02H,02H,7EH,00H,3CH
DB      46H,4AH,52H,62H,3CH,00H                         ;SO
DB      00H,7EH,40H,40H,7EH,02H,02H,7EH,00H,0CH
DB      1CH,2CH,0CH,0CH,0CH,00H                         ;S1
DB      00H,7CH,42H,42H,42H,42H,7CH,00H,7EH,40H
DB      78H,40H,40H,7EH,00H,00H                         ;DLE
DB      00H,7CH,42H,42H,42H,42H,7CH,00H,0CH,1CH
DB      2CH,0CH,0CH,0CH,00H,00H                         ;DC1
DB      00H,7CH,42H,42H,42H,42H,7CH,00H,7EH,02H
DB      02H,3CH,40H,7EH,00H,00H                         ;DC2
DB      00H,7CH,42H,42H,42H,42H,7CH,00H,7CH,04H
DB      04H,3CH,04H,7CH,00H,00H                         ;DC3
DB      00H,7CH,42H,4H,42H,42H,7CH,00H,44H,44H
DB      44H,7EH,04H,04H,00H,00H                         ;DC4
DB      00H,62H,52H,52H,4AH,4AH,46H,00H,4CH,50H
DB      60H,58H,44H,00H,00H,00H                         ;NAK
DB      00H,7EH,40H,40H,7EH,02H,02H,7EH,00H,62H
```

```
DB      52H,52H,4AH,4AH,46H,00H                                 ;SYN
DB      00H,3EH,20H,3CH,20H,20H,3EH,00H,78H,44H
DB      44H,78H,44H,78H,00H,00H                                 ;ETB
DB      00H,3EH,40H,40H,40H,40H,3EH,00H,62H,52H
DB      52H,4AH,4AH,46H,00H,00H                                 ;CAN
DB      00H,3EH,20H,3CH,20H,20H,3EH,00H,62H,52H
DB      52H,4AH,4AH,46H,00H,00H                                 ;EN
DB      00H,7EH,40H,40H,7EH,02H,02H,7EH,00H,78H
DB      44H,44H,78H,44H,78H,00H                                 ;SUB
DB      00H,3EH,20H,3CH,20H,20H,3EH,00H,3EH,40H
DB      40H,40H,40H,3EH,00H,00H                                 ;ESC
DB      00H,7EH,40H,40H,7CH,40H,40H,00H,7EH,40H
DB      7EH,02H,7EH,00H,00H,00H                                 ;FS
DB      00H,3EH,40H,40H,4EH,42H,3CH,00H,7EH,40H
DB      7EH,02H,7EH,00H,00H,00H                                 ;GS
DB      00H,7EH,42H,7EH,50H,4EH,42H,00H,7EH,40H
DB      7EH,02H,7EH,00H,00H,00H                                 ;RS
DB      00H,42H,42H,42H,42H,42H,3CH,00H,7EH,40H
DB      7EH,02H,7EH,00H,00H,00H                                 ;US

;*****************************************************************************
;
EXTRN   SOUND:PROC
EXTRN   PSTRING:PROC
EXTRN   PCHAR:PROC

PUBLIC  WRITE_HEX
;---------------------------------------------------------------
; This procedure converts the byte in the DL register to hex and wri
```

```
; the two hex digits at the current cursor position.
;
;       DL      Byte to be converted to hex.
;
; Uses:         WRITE_HEX_DIGIT
;------------------------------------------------------------------------
WRITE_HEX       PROC    NEAR            ;Entry point
PUSH    CX                      ;Save registers used in this procedu
PUSH    DX
MOV     DH,DL                   ;Make a copy of byte
MOV     CX,4                    ;Get the upper nibble in DL
SHR     DL,CL
CALL    WRITE_HEX_DIGIT         ;Display first hex digit
MOV     DL,DH                   ;Get lower nibble into DL
AND     DL,0Fh                  ;Remove the upper nibble
CALL    WRITE_HEX_DIGIT         ;Display second hex digit
POP     DX
POP     CX
RET
WRITE_HEX       ENDP

PUBLIC  WRITE_HEX_DIGIT
;------------------------------------------------------------------------
; This procedure converts the lower 4 bits of DL to a hex digit and
; writes it to the screen.
;
;       DL      Lower 4 bits contain number to be printed in hex.
;
; Uses:         WRITE_CHAR
```

```
;-------------------------------------------------------------------
WRITE_HEX_DIGIT         PROC    NEAR
PUSH    DX                      ;Save registers used
CMP     DL,10                   ;Is this nibble <10?
JAE     HEX_LETTER              ;No, convert to a letter
ADD     DL,"0"                  ;Yes, convert to a digit
JMP     Short WRITE_DIGIT       ;Now write this character
HEX_LETTER:
ADD     DL,"A"-10               ;Convert to hex letter
WRITE_DIGIT:
CALL    WRITE_CHAR              ;Display the letter on the screen
CALL    CURSOR_RIGHT
POP     DX                      ;Restore old value of AX
RET
WRITE_HEX_DIGIT         ENDP

PUBLIC  WRITE_CHAR
;-------------------------------------------------------------------
; This procedure outputs a character to the screen using the ROM BIO
; routines, so that characters such as the backspace are treated as
; any other character and are displayed.
;   This procedure must do a bit of work to update the cursor positi
;
;       DL      Byte to print on screen
;
; Uses:         CURSOR_RIGHT ,ATTRIBUTE
;-------------------------------------------------------------------
WRITE_CHAR      PROC    NEAR
PUSH    AX
```

```
PUSH    BX
PUSH    CX
PUSH    DX
MOV     AH,9                    ;Call for output of character/attrib
MOV     BH,0                    ;Set to display page 0
MOV     CX,1                    ;Write only one character
MOV     AL,DL                   ;Character to write
MOV     BL,ATTRIBUTE            ;Normal attribute
INT     10h                     ;Write character and attribute
;       CALL    CURSOR_RIGHT            ;Now move to next cursor pos
POP     DX
POP     CX
POP     BX
POP     AX
RET
WRITE_CHAR      ENDP
;
        PUBLIC  GOTO_XY
;-----------------------------------------------------------------
;  This procedure moves the cursor
;
;       DH      Row (Y)
;       DL      Column (X)
;-----------------------------------------------------------------
GOTO_XY         PROC    NEAR
PUSH    AX
PUSH    BX
PUSH    DX
MOV     BH,0                    ;Display page 0
```

```
MOV     AH,2                    ;Call for SET CURSOR POSITION
INT     10h                     ;Let the ROM BIOS do the work
POP     DX
POP     BX
POP     AX
RET
GOTO_XY         ENDP
;
;-----------------------------------------------------------------
;  input   data com1 in ASCII_COM1 =DW
;               com2    ASCII_COM2 =DW
;-----------------------------------------------------------------
    PUBLIC  DISPLAY_DATA_COM
DISPLAY_DATA_COM    PROC
    PUSHA
    CMP     COL,67
    JLE     DISP_1
    MOV     COL,4
    INC     ROW
    CMP     ROW,22
    JLE     DISP_2
    MOV     ROW,22
    CALL    SCROLL_UP
DISP_1:     MOV     DH,ROW
    MOV     DL,COL
    CALL    GOTO_XY
    JMP     SHORT DISP_3
DISP_2:     INC     ROW
    JMP     SHORT DISP_1
```

```
DISP_3:     CMP     F1_HEX,1
    JE      TEXT
    MOV     DX,ASCII_COM1               ;acsii
    MOV     ATTRIBUTE,DH                ;attr
    CALL    WRITE_HEX
    INC     ROW
    MOV     DH,ROW
    MOV     DL,COL
                                        CALL    GOTO_XY
    MOV     DX,ASCII_COM2
    MOV     ATTRIBUTE,DH
    CALL    WRITE_HEX
    MOV     DL,' '
    CALL    WRITE_CHAR
    @GETCUR                             ;get position cursor
    DEC     DH
    MOV     COL,DL
    MOV     ROW,DH
    CMP     DL,66
    JNE     DISP_4
    INC     COL
    JMP     SHORT DISP_4
TEXT:       MOV     DX,ASCII_COM1
    MOV     ATTRIBUTE,DH
    CALL    WRITE_CHAR
    INC     ROW
    MOV     DH,ROW
    MOV     DL,COL
    CALL    GOTO_XY
```

```
    MOV     DX,ASCII_COM2
    MOV     ATTRIBUTE,DH
    CALL    WRITE_CHAR
    DEC     ROW
DISP_4:     INC     COL
    MOV     DH,ROW
    MOV     DL,COL
    CALL    GOTO_XY
    POPA
    RET
DISPLAY_DATA_COM    ENDP

;-------------------------------------------------------------------
    PUBLIC  SCROLL_UP
SCROLL_UP   PROC    NEAR
    MOV     AH,6
    MOV     AL,2                    ;line number
    MOV     BH,7                    ;attribute
    MOV     CH,04                   ;row top-left
    MOV     CL,04                   ;col top-left
    MOV     DH,23                   ;row bum-right
    MOV     DL,67                   ;col bum-right
    INT     10H
    RET
SCROLL_UP   ENDP
;
    PUBLIC  SCROLL_DOWN
SCROLL_DOWN PROC    NEAR
```

```
SCROLL_DOWN PROC    NEAR
    MOV     AH,7
    MOV     AL,2
    MOV     BH,7
    MOV     CH,04
    MOV     CL,04
    MOV     DH,23
    MOV     DL,67
    INT     10H
    RET
SCROLL_DOWN ENDP
;
;
    PUBLIC  CLS_ANALY
CLS_ANALY   PROC
    PUSHA
    MOV     AH,06H
    MOV     AL,0                    ;clear screen
    MOV     BH,7                    ;attr
    MOV     CX,0404H
    MOV     DH,23                   ;row
    MOV     DL,67                   ;col
    INT     10H
    POPA
    RET
CLS_ANALY   ENDP

;*****************************************************************;
;*  -   let position cursor want exchange                        *;
```

```
;*  -   input      ATTRIBUTE                                    *;
;*                 NUM_CHAR_COLOR                               *;
;*************************************************************;
PUBLIC COLOR
COLOR   PROC
PUSH    AX
PUSH    DX
MOV     CX,NUM_CHAR_COLOR           ;number character at exchan
HERE:   MOV     BX,0                        ;No, move new ATTRI
MOV     AH,8                        ;Prepare to write to screen
INT     10H
PUSH    CX

MOV     CX,1                        ;1 char
MOV     BL,ATTRIBUTE
MOV     AH,9                        ;write char , attr
INT     10H

POP     CX
INC     DL                          ;Move on to next letter.
CALL    GOTO_XY
LOOP    HERE                        ;And go back until we're do
POP     DX
CALL    GOTO_XY                     ;Finish and return.
POP     AX
RET
COLOR   ENDP
;
;
```

```
PUBLIC  CLEAR_SCREEN
;-----------------------------------------------------------------
;  This procedure clears the entire screen.
;-----------------------------------------------------------------
CLEAR_SCREEN    PROC    NEAR
PUSH    AX
PUSH    BX
PUSH    CX
PUSH    DX
XOR     AL,AL                   ;Blank entire window
XOR     CX,CX                   ;Set upper left to (0,0)
MOV     DH,24                   ;Set lower right to (24,79)
MOV     DL,79
MOV     BH,7                    ;Use normal attribute for blanks
MOV     AH,6                    ;Call for scroll function
INT     10h                     ;Let the ROM BIOS do the work
POP     DX
POP     CX
POP     BX
POP     AX
RET
CLEAR_SCREEN    ENDP
;
;
PUBLIC  CURSOR_OFF
CURSOR_OFF  PROC
PUSH    AX
PUSH    DX
```

```
PUSH    ES
PUSH    SI
MOV     SI,0040H                ;area bios
MOV     ES,SI
MOV     AL,0AH                  ;selec registeri1
MOV     DX,WORD PTR ES:0063H    ;base port  (3B4h)
OUT     DX,AL                   ;

INC     DX                      ;port in data (3B5h)
MOV     AL,BYTE PTR ES:0061H    ;prosition cursor end
OR      AL,20H                  ;off cursor
OUT     DX,AL

MOV     AL,0BH                  ;selec register 12
DEC     DX                      ;port register (3B4H)
OUT     DX,AL

INC     DX
MOV     AL,BYTE PTR ES:0060H    ;prosition cursor start
OUT     DX,AL

POP     SI
POP     ES
POP     DX
POP     AX
RET
CURSOR_OFF  ENDP
;
;
```

```
PUBLIC  CURSOR_ON
CURSOR_ON   PROC
PUSH    AX
PUSH    DX
PUSH    ES
PUSH    SI

MOV     SI,0040H                ;area bios
MOV     ES,SI
MOV     AL,0AH                  ;selec register11
MOV     DX,WORD PTR ES:0063H    ;base port  (3B4h)
OUT     DX,AL                   ;

INC     DX                      ;port in data (3B5h)
MOV     AL,BYTE PTR ES:0061H    ;prosition cursor end
AND     AL,0DFH                 ;ON cursor
OUT     DX,AL

MOV     AL,0BH                  ;selec register 12
DEC     DX                      ;port register (3B4H)
OUT     DX,AL

INC     DX
MOV     AL,BYTE PTR ES:0060H    ;prosition cursor start
OUT     DX,AL
@SETCURSZ 8,13                  ;set the cursor size
POP     SI
POP     ES
POP     DX
```

```
POP     AX
RET
CURSOR_ON   ENDP
;
;
PUBLIC  BEEP
BEEP    PROC
PUSHA
MOV     DI,500                  ;set frequncy
MOV     BL,25                   ;TIME
SUB     CL,CL
SUB     BH,BH
CALL    SOUND
POPA
RET
BEEP    ENDP
;
;


PUBLIC  CURSOR_RIGHT
;-------------------------------------------------------------------
; This procedure moves the cursor one position to the right, or to t
; next line if it was at the end of a line.
;
; Uses:
;-------------------------------------------------------------------
CURSOR_RIGHT    PROC    NEAR
PUSH    AX
```

```
PUSH    BX
PUSH    CX
PUSH    DX
MOV     AH,3            ;Read the current cursor position
MOV     BH,0            ;On page zero
INT     10h             ;Read cursor position
MOV     AH,2            ;Set new cursor position
INC     DL              ;Set column to next position
;       CMP     DL,79           ;Did we pass column 79?
;       JBE     MOVE_CURSOR     ;No, then move the cursor
;       CALL    SEND_CRLF       ;Yes, go to next line
;       JMP     DONE_CURSOR_RIGHT
;MOVE_CURSOR:
INT     10h             ;Move the cursor
;DONE_CURSOR_RIGHT:
POP     DX
POP     CX
POP     BX
POP     AX
RET
CURSOR_RIGHT    ENDP


    PUBLIC  SHOW_OK
SHOW_OK     PROC
    PUSH    AX
    MOV     ATTRIBUTE,75H       ;mag-wht
    MOV     SI,OFFSET OK
    CALL    PSTRING
```

```
    MOV     ATTRIBUTE,15
    POP     AX
    RET
SHOW_OK     ENDP


    PUBLIC  SHOW_NO
SHOW_NO     PROC
    PUSH    AX
    MOV     ATTRIBUTE,75H
    MOV     SI,OFFSET NO
    CALL    PSTRING
    MOV     ATTRIBUTE,15
    POP     AX
    RET
SHOW_NO     ENDP
;
;*****************************************
;* Write string terminator null(0)       *
;* input     mov   SI,offset ____        *
;* use       call  pchar                 *
;*****************************************
    PUBLIC  DISP_ST_0
DISP_ST_0   PROC
BEGIN_DISP_0:
    MOV     AL,[SI]
    CMP     AL,0
    JZ      END_DISP_0
    CALL    PCHAR
```

```
    INC     SI
    JMP     BEGIN_DISP_0
END_DISP_0:
    RET
DISP_ST_0   ENDP

;-------------------------------------------------------------------
;save data form RS-323 to buffer
;**************************************************
;*       input ascii_com1,1 = data                *
;*                  sbuffer = segment             *
;*                  pbuffer = offset              *
;**************************************************

    PUBLIC  STORE_BUFFER
STORE_BUFFER    PROC
    PUSHA
    PUSH    ES
    MOV     ES,WORD PTR SBUFFER1        ;segment fo buffer
    MOV     DI,WORD PTR PBUFFER1        ;offset (position)
    MOV     AX,ASCII_COM1
    STOSW
    MOV     AX,ASCII_COM2
    STOSW
    MOV     AX,DI
    MOV     CX,4                        ;convert to paragraphs
    SHR     AX,CL                       ;/16 Byte
    CMP     AX,LBUFFER1
    JLE     ST_BUFFER1
```

```
    MOV     DI,0                        ;begin offset buffer
    MOV     F_BUFFER,1                  ;flage buffer is full
ST_BUFFER1:
    MOV     WORD PTR PBUFFER1,DI        ;save offset position
    MOV     WORD PTR CURANT_ADDR,DI     ;
    POP     ES
    POPA
    RET
STORE_BUFFER    ENDP

;*****************************************
;*   load data form buffer to screen     *
;*       input  row , col                *
;*      output  ascii_com1,ascii_com2    *
;*      call    display_data_com         *
;*****************************************

    PUBLIC  LOAD_BUFFER
LOAD_BUFFER     PROC
PUSHA
CMP     PBUFFER1,0
JZ      EXIT_LOAD_B
MOV     SI,WORD PTR CURANT_ADDR
XOR     AX,AX
XOR     DX,DX
MOV     AL,ROW
SUB     AX,4
MOV     BX,2                        ;(col-4)/2
DIV     BL
```

```
CMP     F1_HEX,1
JE      L_BUF_CHAR

CMP     ROW,22
JNE     L_BUF_1
MOV     BX,21
MUL     BX
MOV     CX,AX
MOV     DL,COL
SUB     DL,4
MOV     AX,DX
MOV     BL,3
DIV     BL
MOV     AH,0
ADD     CX,AX                   ;CX=(row-4)/2*21+(col-4)/3
JMP     SHORT L_BUF_4
L_BUF_1:        MOV     CX,210
JMP     SHORT L_BUF_4
L_BUF_CHAR:
CMP     ROW,22
JNE     L_BUF_3
MOV     BX,64
MUL     BX
MOV     DL,COL
SUB     DL,4
ADD     AX,DX
MOV     CX,AX                   ;CX=(row-4)/2*64+(col-4)
JMP     SHORT L_BUF_4
L_BUF_3:        MOV     CX,640
```

```
L_BUF_4:        PUSH    CX
CALL    CUONT_DOWN
POP     CX
L_BUF_5:        CALL    DISP_AT_TOP
MOV     WORD PTR CURANT_ADDR,SI
EXIT_LOAD_B:    POPA
RET
LOAD_BUFFER     ENDP

;-------------------------------------------------------------------------
;
;input CX = number loop and display

DISP_BUFFER     PROC

DISP_BUF1:      PUSH    DS
MOV     DS,WORD PTR SBUFFER1
LODSW
POP     DS
MOV     ASCII_COM1,AX
PUSH    DS
MOV     DS,WORD PTR SBUFFER1
LODSW
POP     DS
MOV     ASCII_COM2,AX
CALL    DISPLAY_DATA_COM

CMP     SI,PBUFFER1
JE      EXIT_DISP_BUF
```

```
PUSH    CX
MOV     AX,SI                   ;
MOV     CX,4                    ;convert to paragraphs
SHR     AX,CL                   ;/16
POP     CX
CMP     AX,LBUFFER1
JLE     DISP_BUF2
MOV     SI,0
DISP_BUF2:      LOOP    DISP_BUF1
EXIT_DISP_BUF:
RET
DISP_BUFFER     ENDP

;*********************************************
;*   input  CX = number character loop       *
;*   output SI = address buffer              *
;*********************************************

CUONT_DOWN      PROC
CMP     F_BUFFER,1
JNE     CUONT_D1
MOV     BX,PBUFFER1
INC     BX
INC     BX
INC     BX
INC     BX
MOV     TAIL,BX
```

```
CUONT_D1:       DEC     SI
DEC     SI
DEC     SI
DEC     SI
CMP     SI,TAIL
JE      EXIT_CUONT_D
CMP     SI,0
JNE     CUONT_D2
PUSH    CX
MOV     AX,LBUFFER1
MOV     CX,4
SHL     AX,CL                   ;Lbuffer * 16
MOV     SI,AX
POP     CX
CUONT_D2:       LOOP    CUONT_D1
CLC
RET
EXIT_CUONT_D:   STC
RET

CUONT_DOWN      ENDP
;-------------------------------------------------------------------
;**********************************
;*      UP ARROW KEY PRESSED      *
;**********************************
PUBLIC  UP_ARROW_KEY
UP_ARROW_KEY    PROC
PUSHA
MOV     SI,WORD PTR CURANT_ADDR
```

```
CALL    CHECK_PAGE              ;check page full ?
JNC     UP_ARROW_0
JMP     EXIT_UP_ARROW

UP_ARROW_0:     @GETCUR         ;get cursor position
CMP     F1_HEX,1                ;1 =char  FF = hex
JE      UP_KEY_CHAR
CMP     DH,22                   ;row = 22 ?
JNE     UP_ARROW_3
XOR     AX,AX
XOR     CX,CX
SUB     DL,4                    ;(col-4)/3=cx
MOV     AL,DL
MOV     BL,3
DIV     BL
MOV     CL,AL                   ;number char
UP_ARROW_1:     CALL    CUONT_DOWN
MOV     CURANT_ADDR,SI          ;save address buffer
MOV     CX,210
CALL    CUONT_DOWN
JC      UP_ARROW_2
MOV     CX,21
JMP     SHORT UP_ARROW_7
UP_ARROW_2:
MOV     CX,210                  ;display 1 page
CALL    DISP_AT_TOP
MOV     CURANT_ADDR,SI
CALL    BEEP
POPA
```

```
RET
UP_ARROW_3:     MOV     SI,CURANT_ADDR
MOV     CX,21
JMP     SHORT UP_ARROW_1
    ;-----------------------
UP_KEY_CHAR:    CMP     DH,22
JNE     UP_ARROW_6
XOR     CX,CX
SUB     DL,4
MOV     CL,DL
UP_ARROW_4:     CALL    CUONT_DOWN
MOV     CURANT_ADDR,SI
MOV     CX,640
CALL    CUONT_DOWN
JC      UP_ARROW_5
MOV     CX,64
JMP     SHORT UP_ARROW_7
UP_ARROW_5:
MOV     CX,640                  ;display 1page
CALL    DISP_AT_TOP
MOV     CURANT_ADDR,SI
CALL    BEEP
POPA
RET
UP_ARROW_6:     MOV     SI,CURANT_ADDR
MOV     CX,64
JMP     SHORT UP_ARROW_4
UP_ARROW_7:     CALL    DISP_AT_TOP
EXIT_UP_ARROW:  POPA
```

```
RET
UP_ARROW_KEY    ENDP

;-----------------------------------------------------------------
PUBLIC  DOWN_ARROW_KEY
DOWN_ARROW_KEY  PROC
MOV     DX,PBUFFER1
CMP     DX,CURANT_ADDR
JE      EXIT_DOWN_ARROW
CALL    SCROLL_UP
MOV     ROW,16H
MOV     COL,4
MOV     DX,1604H
CALL    GOTO_XY
MOV     SI,CURANT_ADDR
CMP     _HEX,1                  ;1= CHAR
JE      DOWN_CHAR
MOV     CX,21
JMP     SHORT DISP_BOT
DOWN_CHAR:      MOV     CX,64
DISP_BOT:       CALL    DISP_BUFFER
MOV     CURANT_ADDR,SI
EXIT_DOWN_ARROW:
        RET
DOWN_ARROW_KEY  ENDP
;-----------------------------------------------------------------
;*****************************************
;*   return   FC = 1  page full          *
;*            = 0   no page full         *
```

```
;*****************************************

CHECK_PAGE      PROC
CMP     F_BUFFER,1                  ;1 = buffer full
JE      CHECK_0
MOV     DX,CURANT_ADDR
CMP     F1_HEX,1
JE      CHECK_CHAR
CMP     DX,840
JLE     CHECK_1
CHECK_0:        CLC
RET
CHECK_1:        STC
RET

CHECK_CHAR:     CMP     DX,2560
JLE     CHECK_1
JMP     SHORT CHECK_0
CHECK_PAGE      ENDP
;
;
;
DISP_AT_TOP     PROC
PUSH    CX
CALL    SCROLL_DOWN
MOV     ROW,4
MOV     COL,4
MOV     DX,0404H
CALL    GOTO_XY
```

```
POP     CX
CALL    DISP_BUFFER
RET
DISP_AT_TOP     ENDP
;
;
public bios_write_cg
bios_write_cg   proc
push    bp
push    es

lea     bp,soh          ;offset of new bitmap
mov     ax,ds           ;segmant of new bitmap into ES
mov     es,ax
mov     cx,01fh         ;load 31 char only
mov     dx,1            ;starting at char 1
mov     bl,0            ;load it as EGA char set 1
mov     bh,16           ;8x16 char = 16 bytes per char
mov     ah,11h
mov     al,00           ;load custom character set
int     10h             ;perform load

pop     es
pop     bp
ret
bios_write_cg   endp

END
```

TERMINAL.ASM

```
    MOV     AX,WORD PTR [SI]        ;get character,attr
;Update output pointer
    INC     BX
    INC     BX                      ;Bump
    CMP     BX,MAX                  ;pointer overflows buffer?
    JNE     OK_OUT_PTR2
    MOV     BX,0                    ;reset to start of buffer
;
OK_OUT_PTR2:
    MOV     DATA_OUT2,BX            ;update
;
;display byte taken frmo buffer
    STI
    CALL    DISPLAY_TTY
    JMP     MONITOR
;
;**********************
;*  output & display *
;**********************
SHOW_AND_SEND:
;wait loop for transmitter holding register empty
    MOV     CX,2000
    PUSH    AX                      ;save character to transmil
    CMP     TERMINAL_OUT,31H        ;selection com1 or com2
    JE      SAND_COM1
    MOV     DX,CARD_BASE_2
    MOV     CARD_BASE,DX
    JMP     SHORT THRE_WAIT
SAND_COM1:  MOV     DX,CARD_BASE_1
```

```
        MOV     CARD_BASE,DX
THRE_WAIT:
        ADD     DX,5                    ;line status  register
        IN      AL,DX                   ;get byte at port
        JMP     SHORT $+2
        TEST    AL,20H                  ;thre bit set?
        JNZ     OK_2_SEND
        LOOP    THRE_WAIT
;wait period timed out.display error message and exit
        POP     AX
        @DispStr ERR1_MS
        JMP     MONITOR
;
OK_2_SEND:
        POP     AX
;place in transmitter holding register to send
        MOV     DX,CARD_BASE            ;THR register
        OUT     DX,AL                   ;send
        JMP     SHORT $+2               ;I/O delay
;
;display character
        CALL    DISPLAY_TTY
        JMP     MONITOR
;
;
DISPLAY_TTY PROC
        PUSHA
        PUSHF
        CMP     AL,0DH
```

```
        MOV     AH,14
        MOV     BX,0                    ;display page
        INT     10H
        POPA
        RET
TTY     ENDP

SCROLL_UP_TTY   PROC
        PUSHA
        MOV     AH,6
        MOV     AL,1                    ;scroll by 1 line
        MOV     BH,7                    ;normal video attribute
        MOV     CH,1                    ;upper row
        MOV     CL,0                    ;upper col
        MOV     DH,23                   ;lower row
        MOV     DL,79                   ;lower col
        INT     10H
        POPA
        RET
SCROLL_UP_TTY   ENDP

;*****************************************************************
;
        END
```

FILECOM.ASM

```
        .286C
        .MODEL  SMALL
        INCLUDE BIOS.INC
        INCLUDE DOS.INC

        .DATA
        EXTRN   UL_COL:BYTE
        EXTRN   UL_ROW:BYTE
        EXTRN   LR_COL:BYTE
        EXTRN   LR_ROW:BYTE
        EXTRN   CARD_BASE:WORD
        EXTRN   CARD_BASE_1:WORD
        EXTRN   CARD_BASE_2:WORD
        EXTRN   ATTRIBUTE:BYTE
        EXTRN   NUM_CHAR_COLOR:WORD
        EXTRN   PATH_NAME:BYTE
        EXTRN   SAND_COM:BYTE
        EXTRN   RECEIVE_COM:BYTE

        PUBLIC  MESS_6
MESS_1  DB  ' DISK  FILE  TRANSFER $'
MESS_2  DB  'Filename: $'
MESS_3  DB  'FILE SEND mode$'
MESS_4  DB  'FILE RECEIVE mode$'
MESS_5  DB  'FILE RECEIVED - CHECKSUM OK -$'
MESS_6  DB  'TERMINATED BY USER$'
MESS_7  DB  'RECEIVING FILE: $'
MESS_8  DB  'FILE(S) TRANSMITTED complaes $'
MESS_9  DB  '******* CHECKSUM ERROR *******$'
;
```

```
;*************************
;     disk data areas    *
;*************************
FILE_IN     DB  63              ;Maximum buffered input
    DB  0               ;Actual input received
F_NAME      DB  64 DUP('README.TXT',0)
HANDLE_3    DW  0               ;File handle
;
F_BUFFER    DB  129 DUP(00H)
;
;Scratch pad DTA to be used by DOS inthe search for matching
;filenames for global characters
DTA_BUFFER  DB  30 DUP (00H)
NEXT_NAME   DB  13 DUP (00H)
    DB  85 DUP (00H)
;
;*************************
;         other data     *
;*************************
    PUBLIC  OLD_DRIVE
OLD_DRIVE   DB  0
    PUBLIC  NEW_DRIVE
NEW_DRIVE   DB  0
CHK_ERR     DB  0                       ;Checksum error control
OK_CNCL     DB  '2  OK  2    2 CANCEL 2$'
TITEL_WAIT  DB  'WAIT$'
CHAR_WAIT   DB  '-\3/'
COURNT      DB  0
SIZEREAD    DB  0
```

```
;****************************
;       ERROR MESSAGES    *
;****************************
    PUBLIC  ERMES_1
ERMES_1     DB  'ERROR-Path or file not found...$'
ERMES_2     DB  'ERROR - Cannot write sector....$'
ERMES_3     DB  'ERROR - Cannot open for read...$'
ERMES_4     DB  'ERROR - cannot open for write..$'
ERR_DRIVE   DB  'ERROR - Invalid Drive...$'
;
;--------------------  CODE  ----------------------------
;
    .CODE

    EXTRN   SAVSCRN:PROC
    EXTRN   GETSCRN:PROC
    EXTRN   CURSOR_ON:PROC
    EXTRN   CURSOR_OFF:PROC
    EXTRN   PSTRING:PROC
    EXTRN   WRITE_CHAR_N_TIMES:PROC
    EXTRN   GOTO_XY:PROC
    EXTRN   FLUSH:PROC
    EXTRN   COLOR:PROC
    EXTRN   SET_DI:PROC
    EXTRN   GETKEY:PROC
    EXTRN   BEEP:PROC
    EXTRN   PCHAR:PROC
    EXTRN   DELAY:PROC
    EXTRN   WAIT_KB:PROC
    EXTRN   WIN_SMALL:PROC
```

```
        EXTRN   CLEAR_BOX:PROC
        EXTRN   WRITE_CHAR:PROC
        EXTRN   BOX:PROC
;*****************************************************************************
;

WIN_FILECOM     PROC

        CALL    WIN_SMALL
        MOV     BX,0406H
        CALL    SET_DI
        MOV     ATTRIBUTE,75H                   ;high revers
        MOV     AL,20H
        MOV     CX,69
        CALL    WRITE_CHAR_N_TIMES
        MOV     BX,041BH
        CALL    SET_DI
        MOV     SI,OFFSET MESS_1                ;display head menu
        CALL    PSTRING
        RET
WIN_FILECOM  ENDP
;
;
DISP_WAIT   PROC
    PUSHA
    PUSH    WORD PTR ATTRIBUTE
    PUSH    DI
    MOV     ATTRIBUTE,12
    MOV     UL_COL,0FH
    MOV     UL_ROW,0DH
```

```
        MOV     LR_COL,14H
        MOV     LR_ROW,10H
        CALL    BOX
        MOV     BX,0E10H
        CALL    SET_DI
        MOV     SI,OFFSET TITEL_WAIT
        CALL    PSTRING
        MOV     DX,0F11H
        CALL    GOTO_XY
        MOV     ATTRIBUTE,14
        MOV     SI,OFFSET CHAR_WAIT
        MOV     BX,00
        MOV     BL,COURNT
        MOV     DL,[SI+BX]
        CALL    WRITE_CHAR                      ;display -\|/

        INC     BX
        CMP     BX,4
        JNE     NO_CLEAR
        MOV     BX,0
NO_CLEAR:   MOV     COURNT,BL
        POP     DI
        POP     WORD PTR ATTRIBUTE
        POPA
        RET
DISP_WAIT   ENDP

;
        PUBLIC  SET_DTA
SET_DTA     PROC
```

```
    @SetDTA  DTA_BUFFER
    RET
SET_DTA     ENDP
;
;convert lowwercase character to uppercase
    PUBLIC UPCASE
UPCASE      PROC
    CMP     AL,'a'
    JB      ON_CONV
    CMP     AL,'z'
    JA      ON_CONV
    SUB     AL,'a'-'A'
ON_CONV:    RET
UPCASE      ENDP
;
;
;set drive A,B,C,D and read path name

    PUBLIC  SET_DRIVE
SET_DRIVE   PROC
    PUSH    DX
    CALL    UPCASE
    SUB     AL,41H                  ;0=A 1=B
    CMP     AL,0                    ;
    JB      ERR_SET_DRIVE           ;al<0
    CMP     AL,3
    JA      ERR_SET_DRIVE           ;al>3
    MOV     NEW_DRIVE,AL            ;al=A,B,C,D
    @SetDrv NEW_DRIVE               ;set drive
```

```
    MOV     AH,47H                          ;get path name
    MOV     SI,OFFSET PATH_NAME
    MOV     BYTE PTR [SI],00
    MOV     DL,NEW_DRIVE
    INC     DL
    INT     21H
    JNC     END_SET_DRIVE
ERR_SET_DRIVE:
    MOV     BX,0F20H
    CALL    SET_DI
    MOV     ATTRIBUTE,7AH
    CALL    BEEP
    MOV     SI,OFFSET ERR_DRIVE
    CALL    PSTRING                         ;display error drive
    MOV     AL,0
    MOV     BH,1
    MOV     BL,0
    CALL    DELAY
END_SET_DRIVE:
    POP     DX
    RET
SET_DRIVE   ENDP
;
;set file name for sand files
GET_FILE        PROC
PUSH    WORD PTR NUM_CHAR_COLOR
MOV     DX,0919H
CALL    GOTO_XY
CALL    CURSOR_ON
BEGIN_FILESAND:
```

```
CALL    GETKEY
JC      EXIT_FILESAND
OR      AL,AL
JZ      EXIT_FILESAND
CMP     AX,0F09H                ;TAB KEY
JE      EXIT_FILESAND
CMP     AX,1C0DH                ;ENTER KEY
JE      OK_FILE_SAND
CALL    BEEP
JMP     BEGIN_FILESAND

OK_FILE_SAND:   MOV     BX,0919H                ;POSITION CURSOR
CALL    SET_DI
MOV     ATTRIBUTE,74H           ;REVERS HIGHT
MOV     AL,20H                  ;ASCII
MOV     CX,45
CALL    WRITE_CHAR_N_TIMES

MOV     ATTRIBUTE,78H           ;revers
@GETSTR FILE_IN,0               ;get respone as ASCIIZ
MOV     DX,0919H
CALL    GOTO_XY                 ;set cursor position
MOV     NUM_CHAR_COLOR,45
MOV     ATTRIBUTE,07H           ;attr normal
CALL    COLOR

EXIT_FILESAND:  PUSHA
PUSHF
CALL    CURSOR_OFF
POPF
```

```
POPA
POP     WORD PTR NUM_CHAR_COLOR
RET
GET_FILE        ENDP
;
;
OK_FILESAND     PROC
PUSH    WORD PTR NUM_CHAR_COLOR
MOV     DX,0B19H                ;position cursor row,col
CALL    GOTO_XY
MOV     ATTRIBUTE,75H           ;revers
MOV     NUM_CHAR_COLOR,8        ;number at display
CALL    COLOR                   ;exchange attribute
BEGIN_OK_FILE:
;               CLC                             ;clear carry(clear err
CALL    GETKEY
JC      EXIT_OK_FILE            ; ESC=cancel
OR      AL,AL
JZ      EXIT_OK_FILE            ;special key
CMP     AX,0F09H                ;key tab
JE      EXIT_OK_FILE
CMP     AX,1C0DH                ;value enter "OK".
JNE     BEGIN_OK_FILE
CALL    FILE_SAND               ;sand files to RC232c
JC      BEGIN_OK_FILE           ;ERROR
EXIT_OK_FILE:   PUSHA
PUSHF
MOV     DX,0B19H
CALL    GOTO_XY
MOV     ATTRIBUTE,07H           ;attribute character normal
```

```
CALL    COLOR

MOV     AX,0600H                ;clear win
MOV     BH,0H
MOV     CX,0C09H
MOV     DX,1049H
INT     10H

POPF
POPA
POP     WORD PTR NUM_CHAR_COLOR
RET
OK_FILESAND     ENDP

;
;
CNCL_FILE       PROC
PUSH    WORD PTR NUM_CHAR_COLOR
MOV     DX,0B25H                ;position cursor row,col
CALL    GOTO_XY
MOV     ATTRIBUTE,75H           ;revers
MOV     NUM_CHAR_COLOR,10       ;number at display
CALL    COLOR                   ;exchange attribute
BEGIN_CNCLFILE:
CALL    GETKEY
JC      EXIT_CNCL_FILE          ; ESC=cancel
OR      AL,AL
JZ      EXIT_CNCL_FILE          ;special key
CMP     AX,0F09H                ;key tab
JE      EXIT_CNCL_FILE
```

```
CMP     AX,1C0DH                ;value enter "OK".
JNE     BEGIN_CNCLFILE
STC                             ;set carry for exit
EXIT_CNCL_FILE:
MOV     ATTRIBUTE,07H           ;attribute character normal
CALL    COLOR
POP     WORD PTR NUM_CHAR_COLOR
RET
CNCL_FILE       ENDP
;-------------------------------------------------------------------------
;
SAND_FILE_MENU  PROC
MENU_FILE:      CALL    GET_FILE
JC      EXIT_SAND_FILE
CMP     AX,4800H                ;key arrow up
JE      CNCL_FILESAND1

FILESAND_OK:    CALL    OK_FILESAND
JC      EXIT_SAND_FILE
CMP     AX,4800H
JE      MENU_FILE
CNCL_FILESAND1:
CALL    CNCL_FILE
JC      EXIT_SAND_FILE
CMP     AX,4800H
JE      FILESAND_OK
JMP     MENU_FILE
EXIT_SAND_FILE:
RET
```

```
;
FILE_SAND   PROC    NEAR
;    CALL    SET_DRV              ;set default drive
SEND_MODE:
    CALL    GET_FIRST            ;input filename
    JC      FILE_SEND_END
;if procedure returns, a first match for the filename has been
;found. the input can contain global filename characters
;The file hand for the matching filename is stored in handle_1
;************************************
;   send filename (first or next)  *
;************************************
NEXT_ENTRY:
    MOV     AL,01                ;code to receive filename
    CALL    SEND_232
    CALL    ACK_R                ;wait for receive to be ready
    JC      FILE_SEND_END        ;for filename
;Receive mode read ready,send the filename and 0FFh terminator
    MOV     SI,OFFSET NEXT_NAME
SEND_NAME:
    MOV     AL,[SI]              ;filename character
    CALL    SEND_232             ;SEND CHARACTER THROUGH LINE
    CMP     AL,0H                ;test for null byte
    JE      NAME_SENT            ;all filename transmitted
    INC     SI                   ;bump pointer
    JMP     SEND_NAME            ;and continue
NAME_SENT:
;send terminator for filename
    MOV     AL,0FFH              ;code recognized by the receiver
    CALL    SEND_232
```

```
        CALL    ACK_R                   ;wait for acknowledge code
        JC      FILE_SEND_END           ;before transmitting record
;
;**********************
;  READ ONE RECORD    *
;**********************
;read one 128-byte disk record using handle and place in buffer
READ_RECORD:
        MOV     AH,63                   ;bios service request number
;to read file (handle mode)
        MOV     BX,HANDLE_3             ;file handle from open function
        MOV     CX,128                  ;128 byte to read (one record)
        MOV     DX,OFFSET F_BUFFER
        INT     21H
;determine if EOF (code=03) or send sector (code=02)
;AX = number of bytes read. AX = 0 if EOF
        MOV     SIZEREAD,AL             ;save number read
        CMP     AX,0                    ;test AX for EOF
        JE      READ_END                ;ax = 0, EOF encountered
;disk sector in F_BUFFER is ready to send
        MOV     AL,02                   ;code to receive record
        CALL    SEND_232                ;send through serial line
        CALL    ACK_R                   ;wait for receive ready
        JC      FILE_SEND_END           ;to start sending record
        CALL    SEND_RECORD             ;send 128 byte to receiver
        CALL    ACK_R
        JC      FILE_SEND_END
        JMP     READ_RECORD             ;read new record and send
;
;**************************
```

```
;       end of read       *
;*************************
READ_END:
;send code to close file
    MOV     AL,03               ;code close file
    CALL    SEND_232            ;through serial line
    CALL    CLOSE_FILE
;
;*****************************
; next matching filename  *
;*****************************
;check for a next filename that satisfies the optional global
;characters in the original input.
    CALL    GET_NEXT
;carry set if no next file match found
    JC      EXIT_FILE
;if no carry a matching filename is in f_buffer
    JMP     NEXT_ENTRY
;******************
; file send      *
;******************
EXIT_FILE:
    MOV     AL,04               ;code send complete
    CALL    SEND_232
    MOV     ATTRIBUTE,79H
    MOV     BX,0E1AH
    CALL    SET_DI
    MOV     SI,OFFSET MESS_8    ;file(s) tranmitted
    CALL    PSTRING
FILE_SEND_END:
```

```
    RET
FILE_SAND   ENDP
;-------------------------------------------------------------------
;*********************************
;*  RECEIVE FILE MAIN PROGRSM   *
;*********************************
;
    PUBLIC  MAIN_FILE_RECEIVE
MAIN_FILE_RECEIVE  PROC
    PUSHA
    CALL    WIN_FILECOM
    CMP     RECEIVE_COM,31H          ;31h=com1, 32=com2
    JE      M_SETCOM1
    MOV     DX,CARD_BASE_2
    MOV     CARD_BASE,DX
    JMP     SHORT M_RECEIVE
M_SETCOM1:  MOV     DX,CARD_BASE_1
    MOV     CARD_BASE,DX
M_RECEIVE:
    MOV     BX,061EH
    CALL    SET_DI
    MOV     ATTRIBUTE,1EH            ;attr hight
    MOV     SI,OFFSET MESS_4         ;display mode
    CALL    PSTRING
    MOV     BX,090AH
    CALL    SET_DI
    MOV     ATTRIBUTE,7              ;attr normal
    MOV     SI,OFFSET MESS_7         ;receiving file :
    CALL    PSTRING
    CALL    FILE_RECEIVE             ;procedure receive file
```

```
        CALL    FLUSH
        MOV     ATTRIBUTE,7
        CALL    WAIT_KB
        CALL    GETSCRN
        POPA
        RET
MAIN_FILE_RECEIVE ENDP
;
;******************
;    receive     *
;******************
FILE_RECEIVE    PROC   NEAR
RECEIVE_MODE:
    MOV     CHK_ERR,0           ;clear checksum error control
    JMP     SHORT NO_R_ENTRY
;
;******************
;  MONITOR CODE   *
;******************
MONITOR_CODE:
;monitor the RS-232 line for a control code and execute
;according to the following
;No carry, AL has control code:
;           01H = receive path or filename and open file
;           02H = receive a 128 characters through the RS-232C line,
;                 send R to acknowledge and wait for new code
;           03H = close file,post message and repeat receive
;carry set,keyboard abort
;
;send acknowledge code to sender
```

```
    MOV     AL,'R'              ;code
    CALL    SEND_232            ;through RS-232C line
;monitor line
NO_R_ENTRY:
    CALL    GET_232             ;get character from RS-232C
    JNC     EXECUTE_CODE        ;carry set if key pressend
;key pressed abort operation,post TERMINATED BY USER
RECEIVE_ABORT:
;           CALL    CLOSE_FILE          ;close open files
    MOV     BX,0E20H
    CALL    SET_DI
    MOV     ATTRIBUTE,79H
    MOV     SI,OFFSET MESS_6    ;terminated by user
    CALL    PSTRING
    JMP     FILE_RECEIVE_END    ;exit receive mode
EXECUTE_CODE:
    CMP     AL,01               ;code to receive filename
    JNE     TST_02              ;continue testing
    JMP     RECEIVE_FNAME
TST_02:     CMP     AL,02               ;code to receive a disk sector
    JNE     TST_03              ;continue testing
    JMP     SHORT RECEIVE_128   ;routine to receive sector
TST_03:     CMP     AL,03               ;close and continue
    JNE     TST_04
    JMP     SHORT END_RECEIVE
TST_04:     CMP     AL,04               ;code transfer completed
    JNE     MONITOR_CODE        ;legal code,ignore

;-------------------------------------------------------------------
;*********************************
```

```
;*    code 04 transfer coplete    *
;*************************************
    MOV     BX,0E1AH
    CALL    SET_DI
    MOV     ATTRIBUTE,79H
    MOV     SI,OFFSET MESS_8
    CALL    PSTRING
    JMP     FILE_RECEIVE_END

;
;*****************************
;  code 03 close disk file *
;*****************************
END_RECEIVE:
;close file,post file recevied ok and repeat receive mode
;if no checksum error
    CALL    CLOSE_FILE
    CMP     CHK_ERR,1
    JNE     OK_CHKSUM           ;no error
    MOV     BX,0E1AH
    CALL    SET_DI
    MOV     ATTRIBUTE,79H
    MOV     SI,OFFSET MESS_9    ;** CHECKSUM ERROR **
    CALL    PSTRING
    JMP     RECEIVE_MODE
;
OK_CHKSUM:
    MOV     BX,0E1AH            ;--CHECKSUM OK---
    CALL    SET_DI
    MOV     ATTRIBUTE,79H
```

```
        MOV     SI,OFFSET MESS_5
        CALL    PSTRING
        JMP     RECEIVE_MODE
    ;
    ;bridge to lable
    GO_TO_ABORT:
        JMP     RECEIVE_ABORT
    ;
    ;**************************************
    ;code 02 receive sector and checksum *
    ;**************************************
    ;get 128 characters from communication line and place in DTA
    RECEIVE_128:
        MOV     AL,'R'              ;signal receive ready
        CALL    SEND_232
        CALL    GET_232
        MOV     SIZEREAD,AL
        MOV     BX,0                ;clear checksum register
        MOV     DI,OFFSET F_BUFFER
        XOR     CX,CX
        MOV     CL,SIZEREAD
    REP_R_128:
        CALL    GET_232
        JC      GO_TO_ABORT         ;key pressed abort
        MOV     [DI],AL             ;place in receiveing buffer
        ADD     BL,AL               ;add-inn to checksum
        INC     DI                  ;bump buffer pointer
        LOOP    REP_R_128           ;repeat until end of sector
    ;receive checksum and compare with value in bl
        CALL    GET_232             ;checksum of sender
```

```
    JC      GO_TO_ABORT         ;key pressed
    CMP     AL,BL               ;test with sum of received chars.
    JE      OK_SUM              ;checksum ok
    MOV     CHK_ERR,1           ;set checksum error switch
;
;************************
;       write sector    *
;************************
OK_SUM:
    MOV     AH,64               ;dos service request number
;to write file (handle mode)
    MOV     BX,HANDLE_3         ;file handle
    MOV     CL,SIZEREAD         ;bytes to write
    MOV     DX,OFFSET F_BUFFER
    INT     21H
;carry flag set if write error occurred
    JNC     OK_WRITE_OP
;write error
    CALL    CLOSE_FILE
    MOV     BX,0E1AH
    CALL    SET_DI
    MOV     ATTRIBUTE,79H
    MOV     SI,OFFSET ERMES_2       ;error:cannot write sector
    CALL    PSTRING
    JMP     SHORT FILE_RECEIVE_END  ;EXIT RECEIVE MODE
OK_WRITE_OP:
    JMP     MONITOR_CODE            ;send r and wait for next
      ;control code from sender
;
;************************
```

```
;code 01 receive filename *
;**************************
;send mode will transmit up to 64 charcters in an ASCIIZ string
;that contains the filename.the code 0FFH will signal that all
;filename has been transmitted.
RECEIVE_FNAME:
    MOV     AL,'R'              ;acknowledge:ready for name
    CALL    SEND_232
;receive filename until 0FFH terminator
    MOV     DI,OFFSET F_NAME
NAME_CHAR:
    CALL    GET_232             ;receive character though line
    JNC     OK_NAME
    JMP     RECEIVE_ABORT       ;key pressed,abort operstion
OK_NAME:
    CMP     AL,0FFH             ;test for filename end
    JE      NAME_END            ;end of filename code received
    MOV     [DI],AL             ;store character
    INC     DI                  ;bump pointer
    JMP     NAME_CHAR           ;next character
NAME_END:
;set terminator and save address of end of filename
    MOV     BYTE PTR [DI],00H   ;next character
    PUSH    DI                  ;save address of terminator
;create file
    MOV     AH,3CH              ;BIOS service request number
    MOV     CX,0                ;create with normal attribute
    MOV     DX,OFFSET F_NAME
    INT     21H
;carry flag is set if create function failed
```

```
    JNC     OK_CREATE
;restore stack,post error message and exit
    POP     DI                      ;afjust stack
    MOV     BX,0E1AH
    CALL    SET_DI
    MOV     ATTRIBUTE,79H
    MOV     SI,OFFSET ERMES_4       ;error:cannot open for write
    CALL    PSTRING                 ;abort with error message
    JMP     SHORT FILE_RECEIVE_END
OK_CREATE:
    MOV     HANDLE_3,AX             ;save mandle

;substitute 00H terminator in ASCIIZ filename string for the $
;to use DOS string display funtion
    POP     DI                      ;address of null byte in stack
    MOV     BYTE PTR[DI],'$'
;display filename string
    MOV     BX,091AH
    CALL    SET_DI
    PUSH    DI
    MOV     AL,20H
    MOV     CX,15
    MOV     ATTRIBUTE,01H
    CALL    WRITE_CHAR_N_TIMES  ;clear screen receive file name
    POP     DI
    MOV     ATTRIBUTE,1EH
    MOV     SI,OFFSET F_NAME
    CALL    PSTRING
    JMP     MONITOR_CODE
FILE_RECEIVE_END:
```

```
    RET
FILE_RECEIVE   ENDP
;
;*************************************************************
;                DISK ACCESS PROCEDURES                      *
;*************************************************************
CLOSE_FILE      PROC    NEAR
;Close disk file using handle
    MOV     AH,62               ;dos service request number
    MOV     BX,HANDLE_3         ;file handle
    INT     21H
;no test for error retur on close
    RET
CLOSE_FILE      ENDP
;
;
GET_FIRST       PROC    NEAR
;input file or part name and find first matching filename in
;the current or specified directory
;Return CF = 1  is error
;       CF = 0  on error
;
;-----------------------¦
;search for first match  ¦
;-----------------------¦

    MOV     AH,78               ;to find first match
    MOV     CX,0                ;normal attribute for file
    MOV     DX,OFFSET F_NAME
    INT     21H
```

```
;carry flag is set if no matched filename found
    JC      NO_MATCH
;matched filename can be found as an asciiz string in the
;buffer nameed NEXT_NAME.use this name to open file and get handle.
    CLC
    MOV     AH,61               ;to opend file (handle mode)
    MOV     AL,2                ;read/write access
    MOV     DX,OFFSET F_NAME    ;NEXT_NAME
    INT     21H
;carry set if opend failed
    JC      NO_OPEN
    MOV     HANDLE_3,AX         ;save handle for new operations
    RET                         ;OK open file
;
NO_MATCH:
    MOV     BX,0E1AH
    CALL    SET_DI
    MOV     ATTRIBUTE,79H
    MOV     SI,OFFSET ERMES_1   ;error:path or file not found
    CALL    PSTRING
    STC                         ;error set carry
    RET
;
NO_OPEN:
    MOV     BX,0E1AH
    CALL    SET_DI
    MOV     ATTRIBUTE,79H
    MOV     SI,OFFSET ERMES_3   ;error:cannot open file
    CALL    PSTRING
    STC                         ;error set carry
```

```
        RET
    ;
    GET_FIRST   ENDP
    ;
    GET_NEXT    PROC    NEAR
    ;get next matching filename,if one can be found that matches
    ;the optional global characters fo the input line.
        MOV     AH,79               ;for next match search
        INT     21H
    ;carry flag is set if no next matched fiename found
        JC      NO_NEXT_MATCH
    ;matched fileanme can be found as an asciiz string in the
    ;buffer named next_name. use thia name to open file and get.handle.
        MOV     AH,61               ;to open file(handle mode)
        MOV     AL,2                ;read/write access
        MOV     DX,OFFSET NEXT_NAME
        INT     21H
    ;carry set if open failed
        JC      NO_OPEN             ;error exit
        MOV     HANDLE_3,AX         ;save handle for new operations
        CLC                         ;return code in carry flag
        RET
    ;
    NO_NEXT_MATCH:
        STC                         ;carry is code no next match
        RET
    GET_NEXT    ENDP
    ;
    ;*************************************************************
    ;*          COMMUNICATIONS PROCEDURES                        *
```

```
;*****************************************************************
;send one disk record(128 bytes) through the RS-232C line
;and word size checksum
;
SEND_RECORD     PROC    NEAR
    MOV     AL,SIZEREAD
    CALL    SEND_232
    MOV     BL,0                ;clear checksum register
    MOV     SI,OFFSET F_BUFFER  ;pointer to source buffer
    XOR     CX,CX
    MOV     CL,SIZEREAD         ;total bytes to send
;1 logical record
REP_S_128:  MOV     AL,[SI]             ;get byte to send
    ADD     BL,AL               ;add-in checksum
    CALL    SEND_232            ;send character through line
    INC     SI                  ;bump pointer
    LOOP    REP_S_128           ;repeat until end of record
;
;send checksum
    MOV     AL,BL
    CALL    SEND_232            ;transmit checksum of sector
    RET
SEND_RECORD ENDP
;
SEND_232    PROC    NEAR
;send AL character through the RS 232C line
    PUSH    AX                  ;save character to send
    MOV     DX,CARD_BASE        ;SERIAL CARD BASE ADDRESS
    ADD     DX,5                ;status register
CHK_THRE:
```

```
        IN      AL,DX                   ;status
        JMP     SHORT $+2               ;I/O delay
        TEST    AL,20H                  ;transmitter holding register
;empty?
        JZ      CHK_THRE                ;No
;ready to send
        POP     AX                      ;restore character to send
        MOV     DX,CARD_BASE            ;transmitter holding register
        OUT     DX,AL                   ;send character
        JMP     SHORT $ + 2             ;I/O delay
        RET
SEND_232    ENDP
;
;************************
;*   RS-232C receive    *
;************************
;receive one character through RS 232C line after flushing
;keyboard buffer
GET_232     PROC    NEAR
        CALL    FLUSH
        MOV     DX,CARD_BASE            ;serial card base address
        ADD     DX,5                    ;status register
CHK_STAT:
        CALL    DISP_WAIT
        IN      AL,DX                   ;character ready?
        TEST    AL,1                    ;data ready bit on status reg
        JNZ     D_READY                 ;Get dat
        TEST    AL,00011110B            ;bits 1,2,3, and 4 are error bits
        JNZ     DATA_ERROR              ;go if error bit set
;test for key pressed
```

```
        PUSH    DX                      ;save card base address
        MOV     AH,1                    ;for keyboard status
        INT     16H
        POP     DX
        JZ      CHK_STAT                ;No key pressed,continue
    ;key was pressed during line monitoring
        STC                             ;carry flag return code
        RET
    ;
    ;
    GET_DATA    PROC    NEAR
    D_READY:
        MOV     DX,CARD_BASE            ;receive register
        IN      AL,DX                   ;read character  into AL
        CLC                             ;Normal exit
        RET
    DATA_ERROR:
        MOV     AL,0A8H                 ;inverted? = error received
        CLC                             ;normal exit
        RET
    ;
    GET_DATA    ENDP
    GET_232     ENDP
    ;
    ;wait for receive acknowledge code R or abort if a key is pressed
    ;return carry flag = 1  = error  key is pressed
    ;                  = 0  = NO error
    ;
    ACK_R       PROC    NEAR
        PUSHA
```

```
        CALL    FLUSH
GET_STATUS:
        CALL    DISP_WAIT
        MOV     DX,CARD_BASE        ;serial card base address
        ADD     DX,5                ;status register
CHK_4_DATA:
        IN      AL,DX               ;get status
        TEST    AL,1                ;data ready bit of status register
        JNZ     DATA_READY          ;go if set
;check keyboard for a key pressed
        MOV     AH,1
        INT     16H                 ;get keyboard status
        JZ      GET_STATUS          ;no key pressed
;abort sending
        MOV     BX,0E20H
        CALL    SET_DI
        MOV     ATTRIBUTE,79H
        MOV     SI,OFFSET MESS_6
        CALL    PSTRING             ;dispaly terminated by user
        STC                         ;set carryflag is error
        POPA
        RET
DATA_READY:
        CALL    GET_DATA
        CMP     AL,'R'              ;test for acknowledge code
        JNE     GET_STATUS          ;repeat if wrong code
        CLC                         ;clear carry flage is no error
        POPA
        RET                         ;return if R received
ACK_R   ENDP
```

SHELL.ASM

```
NAME    SHELL
PAGE    55,132
TITLE   'SHELL.ASM ** MS-DOS shell'
INCLUDE DOS.INC


show_command    db  '       INTERNAL COMMAND',0aH,0dH,0ah,0dh
db  '       CLS  =  Clear Screen',0ah,0dh
db  '       DOS  =  Run command.com',0ah,0dh
db  '       EXIT =  Exit Dos and OS Shell',0ah,0dh,24h

stdin   equ     0                       ;standard input device
stdout  equ     1                       ;standard Output device
stderr  equ     2                       ;standard error device

cr      equ     0dh                     ;ASCII carriage return
lf      equ     0ah                     ;ASCII line feed
blank   equ     20h                     ;ASCII blank code
e_sc    equ     01bh                    ;ASCII SDVII escape code

    ;these variables must be in Code Seg:
stk_seg dw      0                       ;original SS contents
stk_ptr dw      0                       ;original SP contents

int23h_seg  dw  0                       ;segment ctrl + c
int23h_off  dw  0                       ;offset ctrl + c

shell_exit  db  00                      ;status end program
commands equ $                          ;table of "intrinsic" commands
    ;each entry is a null-terminated
```

```
    ;string, followed by the offset
    ;of the procedure to be executed
    ;for that command.
db      'CLS',0
dw      cls_cmd
db      'DOS',0
dw      dos_cmd
db      'EXIT',0
dw      exit_cmd
db      0                       ;table terminated with null string

com_var         db      'COMSPEC=',0        ;Environment Block variabl

    ;filespec of COMMAND.COM moved
com_spec db     80 dup (0)              ;here from the Environment Block

nultail db      0,cr                    ;a "Null" command tail for invokin
    ;COMMAND.COM as another shell

cmd_tail db     0,' /C '                ;command tail invoking COMMAND.COM
    ;as a transient command processor

inp_buf db      80 dup (0)              ;command line from Standard Input

inp_buf_length    equ    $-inp_buf
cmd_tail_length   equ    $-cmd_tail-1

prompt  db      cr,lf,'<Shell Command>: '       ;the shell's prompt t
prompt_length     equ    $-prompt
```

```
env_seg dw      0                       ;segment of Environment Block

msg1    db      cr,lf
db      'Unable to de-allocate memory.'
db      cr,lf
msg1_length       equ    $-msg1

msg2    db      cr,lf
db      'EXEC of COMMAND.COM failed.'
db      cr,lf
msg2_length       equ    $-msg2

msg3    db      cr,lf
db      'No COMSPEC variable in Environment.'
db      cr,lf
msg3_length       equ    $-msg3

cls_str db      e_sc,'[2J'              ;this is the ANSI Standard contra
cls_str_length  equ     $-cls_str       ;sequence to clear the screen.

par_blk equ $                           ;Parameter Block fo EXEC call
dw  0                           ;segment address, environment block
par_cmd dw      offset cmd_tail     ;address of command line
dw      seg cmd_tail
dd      -1                      ;address of default FCB #1
dd      -1                      ;address of default FCB #2

;
;-------------------------------------------------------------------
;
```

```
extrn   delay:proc

public  shell
shell   proc                            ;at entry DS = ES = PSP
pusha                           ;push all generol register
mov     shell_exit,0            ;initail status exit
mov     ax,es:[002ch]           ;get segment of Environment Block
mov     env_seg,ax              ;from PSP and save it.
  ;now release unneeded memory.

call    get_comspec             ;get file spec for COMMAND.COM.
jnc     shell2                  ;jump if it was found ok.
mov     dx,offset msg3          ;COMSPEC variable not found in
mov     cx,msg3_length          ;Environment block, print error
jmp     short shell4            ;message and exit.
shell2:
@getint 23h
mov     int23h_seg,es           ;get interrupt 23h segment and
mov     int23h_off,bx           ;offset

mov     dx,offset shell3        ;set Ctrl-C vector <Int 23H>
mov     ax,cs                   ;so that shell can keep control.
mov     ds,ax                   ;DS:DX = addr of Ctrl-C handler
mov     ax,2523h                ;Function 25H = Set Interrupt
int     21h
mov     ax,@data                ;make DS and ES point to our
mov     ds,ax                   ;data segment again.
mov     es,ax
@DispStr show_command
shell3:                                 ;main loop of command interpreter
```

```
cmp     shell_exit,1
je      exit_e                  ;end program
call    get_cmd                 ;get a command from user.
call    intrinsic               ;check if intrinsic function.
jnc     shell3                  ;yes, it was processed.
call    extrinsic               ;no, pass it to COMMAND.COM
jmp     shell3                  ;then get another COMMand.
shell4:                                 ;COME here if error detected, with
    ;DS:DX = message addr, CX = length.
mov     bx,stderr               ;print error message on
mov     ah,40h                  ;Standard Error Device
int     21h
mov     al,0                    ;Minutes
mov     bh,1                    ;wait 1.5 saconds
mov     bl,50
call    delay                   ;exit to DOS with return code = 1
exit_e:
push    ds
mov     dx,int23h_off
mov     ax,int23h_seg
mov     ds,ax
mov     ah,25h                  ;set interrupt
mov     al,23h                  ;ctrl-c
int     21h
pop     ds

popa                            ;pop all register
ret                             ;indicating an error condition.
shell   endp
;
```

```
popa                    ;pop all register
ret                     ;indicating an error condition.
shell   endp
;
;*****************************************************************************
;
intrinsic proc  near                ;Decode a user command against
    ;the table "commands". If match
    ;found, run the subnroutine, and
    ;return Carry = False.  If no match,
    ;return Carry = True.
mov     si,offset commands   ;let SI = start of command table.
intr1:  cmp     byte ptr [si],0     ;is COMMand table exhausted?
je      intr7               ;jump, end of table found.
mov     di,offset inp_buf   ;no, let DI = addr of user input.
intr2:  cmp     byte ptr [di],blank ;scan off any leading blanks.
jne     intr3
inc     di
jmp     intr2
intr3:  mov     al,[si]             ;get next command char.
or      al,al               ;check if end of string.
jz      intr4               ;jump, entire string matched.
cmp     al,[di]             ;compare to next input char.
jnz     intr6               ;jump, found mismatch.
inc     si                  ;go to next char in strings.
inc     di
jmp     intr3
intr4: cmp      byte ptr [di],cr    ;make sure user's command
je      intr5               ;is the same length,next char
cmp     byte ptr [di],blank ;must be blank or return.
```

```
inc     si                      ;go to next char in strings.
inc     di
jmp     intr3
intr4: cmp      byte ptr [di],cr    ;make sure user's command
je      intr5                   ;is the same length,next char
cmp     byte ptr [di],blank ;must be blank or return.
jne     intr6
intr5:  call    word ptr [si+1]     ;run the command routine
clc                             ;then return carry flag = false
ret                             ;as success flag
intr6:  lodsb                           ;look for end of this command st
or      al,al
jnz     intr6                   ;not end yet, loop.
add     si,2                    ;skip over command routine offset.
jmp     intr1                   ;try to match next command.
intr7:  stc                             ;no match on command, exit
ret                             ;with Carry = True.
intrinsic endp



extrinsic proc  near                    ;process an extrinsic command
    ;by passing it to COMMAND.COM
    ;with a " /C " command tail.
mov     al,cr                   ;find length of the command
mov     cx,cmd_tail_length  ;by scanning for Carriage Return.
mov     di,offset cmd_tail+1
cld
repnz scasb
mov     ax,di                   ;calc length of command tail,
```

```
sub     ax,offset cmd_tail+2 ;not including the Carriage Return.
    ;store length of synthesized
mov     cmd_tail,al          ;command tail for EXEC function.
mov     par_cmd,offset cmd_tail ;address of command tail
call    exec                 ;call the EXEC function to pass
ret                          ;command line to COMMAND.COM.
extrinsic endp

get_cmd proc    near                 ;Prompt user and get a command.
mov     dx,offset prompt     ;display the shell prompt
mov     cx,prompt_length     ;on the Standard Output Device.
mov     bx,stdout
mov     ah,40h               ;Function 40H = Write file or device
int     21h
mov     dx,offset inp_buf    ;get a line from the Standard
mov     cx,inp_buf_length    ;Input Device and place in our
mov     bx,stdin             ;command line buffer.
mov     ah,3fh               ;Function 3fh = Read file or device
int     21h
mov     si,offset inp_buf    ;fold all lowercase characters
mov     cx,inp_buf_length    ;in the command line to uppercase.
gcmd1:  cmp     byte ptr [si],'a'
Jb      gcmd2
cmp     byte ptr [si],'z'
ja      gcmd2
sub     byte ptr [si],'a'-'A'
gcmd2:  inc     si
loop    gcmd1
ret                          ;back to caller
```

```
get_cmd endp

;
;

get_comspec proc    near

    ;Get file specification of COMMAND.COM

    ;from Environment "COMSPEC=" variable.

    ;Returns Carry - False if COMSPEC found.

    ;Returns Carry = True if no COMSPEC.

mov     si,offset com_var   ;let DS:SI = string to match.

call    get_env             ;go search Environment Block.

    ;if environment variable not found,

jc      gcsp2               ;return Carry - True as failure code.

    ;if var found, ES:DI points past "=".

mov     si,offset com_spec  ;copy Env variable to our data segment.

gcsp1:  mov     al,es:[di]          ;transfer null-terminated string

mov     [si],al

inc     si

inc     di

or      al,al               ;null found yet? (turns off Carry)

jnz     gcsp1               ;no, get next character.

    ;success, return Carry Flag = False.

gcsp2:  ret

get_comspec endp



get_env proc    near                ;Search Environment Block

    ;Call DS:SI = "NAME=" to match.

    ;Uses contents of "ENV_SEG".

    ;Returns Carry = False and ES:DI
```

```
        ; pointing to parameter if found,
        ;Returns Carry = True if no match.
mov     es,env_seg          ;get segment of Environment Blo.
xor     di,di               ;initialize offset to Env Block.
genv1:  mov     bx,si               ;initialize pointer to pattern.
cmp     byte ptr es:[di],0  ;end of Environment block?
jne     genv2               ;jump if end not found.
stc                         ;return Carry = True as failure flag.
ret
genv2:  mov     al,[bx]             ;get character from pattern.
or      al,al               ;end of pettern? (turns off Carry)
jz      genv3               ;yes, entire match succeeded.
cmp     al,es:[di]          ;compare to char in Environment Block.
jne     genv4               ;jump if match failed.
inc     bx
inc     di
jmp     genv2
genv3:                              ;ALL matched, return ES:DI point
ret                         ;to parameter Carry Flag = False.
genv4:  xor     al,al               ;scan forward in Environment Blo
mov     cx,-1               ;for zero byte.
cld
repnz   scasb
jmp     genv1               ;go compare next string.
get_env endp


exec    proc    near                ;call MS-DOS EXEC function
        ;to run COMMAND.COM.
```

```
push    ds                      ;save data segments.
push    es
mov     cs:stk_seg,ss           ;save copy of SS:SP for use
mov     cs:stk_ptr,sp           :after return from overlay.
mov     dx,offset com_spec      ;now load and execute COMMAND.COM.
mov     bx,offset par_blk
mov     ah,4bh                  ;function 4BH = EXEC
mov     al,0                    ;subfunction 0 = load and execute
int     21h
mov     ss,cs:stk_seg           ;restore stack segment
mov     sp,cs:stk_ptr           ;and stack pointer.
pop     es                      ;restore data segments.
pop     ds
jnc     exec1                   ;jump if no errors.
mov     dx,offset msg2          ;EXEC failed, print error
mov     cx,msg2_length          ;message.
mov     bx,stderr
mov     ah,40h
int     21h
exec1:  ret                             ;back to caller
exec    endp


cls_cmd proc    near                    ;intrinsic CLS command
    ;  = clear the screen
mov     dx,offset cls_str       ;send the ANSI control sequence
mov     cx,cls_str_length       ;to clear the screen.
mov     bx,stdout
mov     ah,40h
```

```
int     21h
ret
cls_cmd endp


dos_cmd proc   near                    ;intrinsic DOS command
    ;  = run COMMAND.COM
mov     par_cmd,offset nultail ;set command tail to null string.
call    exec                   ;now EXEC the COMMAND.COM.
ret
dos_cmd endp

exit_cmd proc   near                   ;intrinsic EXIT Command
mov     shell_exit,1           ;  = leave this shell
ret                            ;call DOS terminate function
    ;with return code of zero.
exit_cmd endp

end
```

CATPRN.ASM

```
; Program CATPRN.ASM
; Purpose Text file displayer
; Input   File name from command line or prompt
; Output  Display file to printer
; By      Mr.Kitjawat Pandumrong

  .286C
  include dos.inc
  .MODEL  small
  .CODE
  ORG     100H
start:    jmp     main

font  db 0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
      db 0,0,0,0,0,0
      db 0,0,0,50,127,128,73,8,0,73,8,0,73,8,0,73,8,0,73,8,0,73,8,0,
      db 8,0,38,127,128,0,0,0,0,0,0
      db 0,0,0,50,65,0,73,34,0,73,20,0,73,8,0,73,8,0,73,8,0,73,34,0,
      db 65,0,0,0,0,0,0,0
      db 0,0,0,0,127,65,0,73,34,0,73,20,0,73,8,0,73,8,0,73,8,0,73,20
      db 73,34,0,38,65,0,0,0,0,0,0
      db 0,0,0,127,64,0,73,64,0,73,64,0,73,64,0,73,127,128,73,64,0,7
      db 0,73,64,0,73,64,0,0,0,0,0,0,0
      db 0,0,0,127,30,0,73,34,0,73,66,128,73,64,128,73,64,128,73,64,
      db 64,128,73,33,0,73,30,128,0,0,0,0,0,0
      db 0,0,0,15,127,128,20,4,0,36,4,0,68,4,0,68,10,0,68,16,0,36,33
      db 0,0,15,64,128,0,0,0,0,0,0
      db 0,0,0,127,127,128,73,0,128,73,0,128,73,0,128,73,0,128,73,0,
      db 0,128,73,0,128,55,1,128,0,0,0,0,0,0
```

```
db 0,0,0,127,25,128,73,36,128,73,68,128,73,68,128,73,68,128,73
db 73,68,128,73,68,128,55,51,0,0,0,0,0,0,0
db 0,0,0,127,96,0,8,64,0,8,64,0,8,64,0,8,127,128,8,64,0,8,64,0
db 127,96,0,0,0,0,0,0,0
db 0,0,0,127,127,128,1,72,0,1,72,0,1,72,0,1,72,0,1,72,0,1,72,0
db 7,96,0,0,0,0,0,0,0
db 0,0,0,96,64,0,16,64,0,4,64,0,2,64,0,1,127,128,2,64,0,4,64,0
db 0,96,64,0,0,0,0,0,0,0
db 0,0,0,127,127,128,72,72,0,72,72,0,72,72,0,72,72,0,72,72,0,7
db 72,72,0,64,64,0,0,0,0,0,0,0
db 0,0,0,60,127,128,66,68,0,65,68,0,65,68,0,65,68,0,65,68,0,65
db 65,37,0,34,28,128,0,0,0,0,0,0
db 0,0,0,50,63,73,64,128,73,64,128,73,64,128,73,64,128,73,64,1
db 73,64,128,73,64,128,38,63,0,0,0,0,0,0,0,0
db 0,0,0,54,0,0,73,0,0,73,0,0,73,64,128,73,127,128,73,64,128,7
db 73,0,0,54,0,0,0,0,0,0,0,0
db 0,0,0,127,127,128,65,0,128,65,0,128,65,0,128,65,0,128,65,0,
db 0,128,34,0,128,31,1,128,0,0,0,0,0,0
db 0,0,0,127,0,0,65,0,0,65,16,128,65,32,128,65,127,128,65,0,12
db 128,33,0,0,31,0,0,0,0,0,0,0,0
db 0,0,0,127,57,128,65,66,128,65,66,128,65,66,128,65,66,128,65
db 65,68,128,33,57,128,31,0,0,0,0,0,0,0,0
db 0,0,0,127,17,0,65,32,128,65,64,128,64,64,128,64,68,128,64,6
db 64,68,128,33,37,128,31,27,0,0,0,0,0,0,0
db 0,0,0,127,3,0,65,5,0,65,9,0,65,17,0,65,33,0,65,65,0,65,65,0
db 121,31,1,0,0,0,0,0,0,0
db 0,0,0,127,127,128,32,4,0,16,8,0,8,0,0,0,20,0,4,0,0,2,34,0,0
db 127,6,128,0,0,0,0,0,0
db 0,0,0,50,64,0,73,32,0,73,16,0,73,8,0,73,7,128,73,8,0,73,16,
```

```
        db 0,22,64,0,0,0,0,0,0,0
        db 0,0,0,127,127,128,73,68,128,73,68,128,73,68,128,73,68,128,7
        db 73,68,128,73,68,128,99,59,128,0,0,0,0,0,0
        db 0,0,0,28,127,128,34,0,0,65,32,0,65,16,0,65,8,0,65,4,0,65,2,
        db 54,127,128,0,0,0,0,0,0
        db 0,0,0,127,127,128,73,32,0,73,16,0,73,8,0,73,4,0,73,8,0,73,1
        db 0,65,127,128,0,0,0,0,0,0,0
        db 0,0,0,54,127,128,73,68,128,73,68,128,73,68,128,73,68,128,73
        db 73,68,128,73,68,128,54,59,128,0,0,0,0,0,0
        db 0,0,0,127,25,0,73,36,128,73,68,128,73,68,128,73,68,128,73,6
        db 68,128,73,66,128,65,49,0,0,0,0,0,0,0,0
        db 0,0,0,127,25,0,80,36,128,80,68,128,80,68,128,80,68,128,80,6
        db 68,128,64,66,128,64,49,0,0,0,0,0,0,0,0
        db 0,0,0,62,25,0,65,36,128,65,68,128,65,68,128,73,68,128,73,68
        db 68,128,73,66,128,62,49,0,0,0,0,0,0,0,0
        db 0,0,0,127,25,0,72,36,128,72,68,128,72,68,128,72,68,128,72,6
        db 68,128,72,66,128,55,49,0,0,0,0,0,0,0,0
        db 0,0,0,126,25,0,1,36,128,1,68,128,1,68,128,1,68,128,1,68,128
        db 1,66,128,126,49,0,0,0,0,0,0,0,0
;-------------------------------------------------------------------

err1    DB      13,10,'File not found:.....',13,10,24h
helpmsg DB      13,10,'**** PRINTER DATA COMMUNICATION ****',13,10
  DB    'A:>CATPRN [[b:][path]filename] [/C] ',13,10
  DB    'A:>CATPRN [[b:][path]filename] [/H]',13,10
  DB    'C = Print Character',13,10
  DB    'H = Print Hex',13,10,24h
h_tx    DB      ':Tx3 '
h_rx    DB      ':Rx3 '
```

```
s_186     DB      ' :'
cr_lf     DB      13,10
type1     DB      ' Asynchronous'
head      DB      'PROTOCOL ANALYZER'
t_code    DB      ' ASCII '
buf_prn   DB      ?
p_handle  DW      4                                   ;PRN
fchar     DB      'C'
fname     DB      66 dup (0)
fhandle   DW      0
speed     DB      15 DUP (0)
buf_com1  DB      64 DUP (0)
buf_com2  DB      64 DUP (0)
buf_data1 DW      0
buf_data2 DW      0
f_eof     DB      0                                   ;flage EOF
p_count   DW      0
graph_m   db      1bH,74H,01h                         ;Graphics mo
set_left  db      1bh,6ch,05h                         ;set left ma

nlq       db      1bh,"x",1                           ;select NLQ
defin_chr db      1bh,"&",0,";;",0,12,0               ;define uses
nor_set   db      1bh,"%",0                           ;select use-
def_set   db      1bh,"%",1
print_com db      ?
;-------------------------------------------------------------------

main:
; Try to open command line file
```

```
    mov     si,81h
    lodsb
    cmp     al,0dh                                  ;enter
    jne     checkins
error:  mov     ah,09h
    mov     dx,offset helpmsg
    int     21h
    int     20h

checkins: mov     si,82h
    mov     di,offset fname
    mov     al,es:[si]
    inc     si
copy:   mov     [di],al
    inc     di
    mov     al,es:[si]
    inc     si
    cmp     al,20h
    loopne  copy
    mov     al,0
    mov     [di],al
    mov     bl,es:[80h]
    sub     bh,bh
    cmp     byte ptr es:[bx+7Fh],'/'
    jne     error
    mov     al,byte ptr es:[bx+80h]
    mov     fchar,al
    mov     ax,3d00h                        ;open file
```

```
  mov     dx,offset fname              ;and read access
  int     21h
  jc      badfile
  mov     fhandle,ax
  jmp     short opened
badfile:
  @DispCh 07h                          ;beep
  mov     ah,09h
  mov     dx,offset err1
  int     21h
  int     20h
opened:
  mov     ah,3fh                       ;read file
  mov     bx,fhandle
  mov     cx,15
  mov     dx,offset speed
  int     21h
  jc      error
  mov     cx,3
  mov     dx,offset graph_m
  call    prn_bytes
  mov     cx,3
  mov     dx,offset set_left
  call    prn_bytes
  @Write  nlq,3,p_handle
  @Write  def_set,3,P_handle
p_head:
  call    print_head
  mov     cx,26                        ;number line per 1 page
```

```
l_again:
  call    prn_line                    ;print 1line
  jc      quit
  cmp     f_eof,1
  je      quit
  loop    l_again                     ;print 20 line
  call    line_buttom
  jmp     p_head
quit:     mov     ah,3eh                        ;close file
  mov     bx,fhandle
  int     21h
  call    line_buttom
  int     20h                         ; Quit
;-------------------------------------------------------------

PUBLIC  WRITE_HEX

; This procedure converts the byte in the DL register to hex and wri
; the two hex digits at the current cursor position.
;
;       DL      Byte to be converted to hex.
;
; Uses:         WRITE_HEX_DIGIT
;-------------------------------------------------------------
WRITE_HEX       PROC    NEAR            ;Entry point
PUSH    CX                      ;Save registers used in this procedu
PUSH    DX
MOV     DH,DL                   ;Make a copy of byte
MOV     CX,4                    ;Get the upper nibble in DL
```

```
        SHR     DL,CL
        CALL    WRITE_HEX_DIGIT         ;Display first hex digit
        MOV     DL,DH                   ;Get lower nibble into DL
        AND     DL,0Fh                  ;Remove the upper nibble
        CALL    WRITE_HEX_DIGIT         ;Display second hex digit
        POP     DX
        POP     CX
        RET
WRITE_HEX       ENDP

        PUBLIC  WRITE_HEX_DIGIT
;-------------------------------------------------------------------
; This procedure converts the lower 4 bits of DL to a hex digit and
; writes it to the screen.
;
;       DL      Lower 4 bits contain number to be printed in hex.
;
; Uses:         WRITE_CHAR
;-------------------------------------------------------------------
WRITE_HEX_DIGIT         PROC    NEAR
        PUSH    DX                      ;Save registers used
        CMP     DL,10                   ;Is this nibble <10?
        JAE     HEX_LETTER              ;No, convert to a letter
        ADD     DL,"0"                  ;Yes, convert to a digit
        JMP     Short WRITE_DIGIT       ;Now write this character
HEX_LETTER:
        ADD     DL,"A"-10               ;Convert to hex letter
WRITE_DIGIT:
        MOV     BUF_PRN,DL
```

```
CALL    PRN_1                   ;Display the letter on the screen
POP     DX                      ;Restore old value of AX
RET
WRITE_HEX_DIGIT         ENDP

;-------------------------------------------------------------------

print_head  proc
    mov     dx,offset cr_lf
    mov     cx,2
    call    prn_bytes
    mov     buf_prn,201                 ;I
    call    prn_1
    mov     cx,69
    mov     buf_prn,205                 ;M
print_h_1:  call    prn_1
    loop    print_h_1
    mov     buf_prn,187                 ;;
    call    prn_1
    mov     cx,2
    mov     dx,offset cr_lf
    call    prn_bytes
    mov     buf_prn,186                 ;:
    call    prn_1
    mov     cx,13
    mov     dx,offset type1             ;Asyn
    call    prn_bytes
    mov     cx,56
    mov     buf_prn,20h
print_h_2:  call    prn_1
```

```
print_head  endp



prn_1       proc
    push    cx
    @Write  buf_prn,1,p_handle
    pop     cx
    ret
prn_1       endp

prn_bytes   proc
    mov     bx,WORD PTR p_handle
    mov     ah,40h
    int     21h
    ret
prn_bytes   endp

;*****************************************
;*    print 64 character per 1 line      *
;*****************************************
prn_line  proc
  push    cx
;clear buffer com1 and com2
  cmp     fchar,'C'
  je      C_count
  mov     p_count,21
  jmp     set_21
C_count:  mov     p_count,64                          ;set number ch
set_21:   mov     ax,0
```

```
    mov     cx,32
    mov     di,offset buf_com1                  ;clear buffer com1
    repe    stosw
    mov     cx,32
    mov     di,offset buf_com2
    repe    stosw                               ;clear buffer com2
    mov     cx,p_count
    mov     si,offset buf_com1
    mov     di,offset buf_com2
r_again:  push    cx
    mov     ah,3fh                               ;read file
    mov     bx,fhandle
    mov     cx,4
    mov     dx,offset buf_data1
    int     21h
    pop     cx
    jnc     r_ok
    jmp     prn_error                           ;read error
r_ok:     cmp     ax,0                                ;EOF
    je      eof_p
    mov     ax,word ptr buf_data1               ;convert code data c
    cmp     fchar,'H'                           ;comp si char or hex
    je      p_com1                              ;go = hex
    cmp     al,0
    jne     p_com1                              ;n_zero1
    mov     al,20h
p_com1:   mov     byte ptr [si],al
    mov     ax,word ptr buf_data2               ;convert code data co
    cmp     fchar,'H'
```

```
    je      p_com2
    cmp     al,0
    jne     p_com2                                  ;n_zero2
    mov     al,20h
    jmp     short p_com2
p_com2: mov     byte ptr [di],al
    inc     si
    inc     di
    loop    r_again
    jmp     short p_con
eof_p:  mov     f_eof,1
p_con:  @write  cr_lf,2,p_handle
    mov     dx,offset h_tx
    mov     cx,5
    call    prn_bytes

;-------------------------------------------------------------
    cmp     fchar,'C'
    je      p_char1
    mov     cx,p_count
    mov     di,offset buf_com1                      ;print hex data com1
p_hex1: mov     dl,byte ptr [di]
    call    write_hex
    inc     di
    mov     buf_prn,20h
    call    prn_1
    loop    p_hex1
    call    prn_1
    jmp     short p_hex2
```

```
p_char1:
  mov     cx,p_count
  mov     si,offset buf_com1                    ;print com1
p_loop1:  mov     al,byte ptr [si]
  call    prn_font
  inc     si
  loop    p_loop1
p_hex2:
  mov     cx,2
  mov     dx,offset s_186                       ;:
  call    pr_bytes
  mov     dx,offset cr_lf
  mov     cx,2
  call    prn_bytes
  mov     cx,5
  mov     dx,offset h_rx
  call    prn_bytes
  cmp     fchar,'C'
  je      p_char2
  mov     cx,p_count
  mov     di,offset buf_com2                    ;print hex data com2
p_hex3:   mov     dl,byte ptr [di]
  call    write_hex
  mov     buf_prn,20h
  call    prn_1
  inc     di
  loop    p_hex3
  call    prn_1
  jmp     short p_hex4
```

```
p_char2:
  mov     cx,p_count
  mov     si,offset buf_com2                    ;print com2
p_loop2:  mov     al,byte ptr [si]
  call    prn_font
  inc     si
  loop    p_loop2
p_hex4:
  mov     cx,2
  mov     dx,offset s_186                       ;:
  call    prn_bytes
  jmp    short end_prn_1
prn_error:
  stc
end_prn_1:
  pop     cx
  ret
prn_line  endp

;-------------------------------------------------------------
line_buttom proc
    mov     dx,offset cr_lf
    mov     cx,2
    call    prn_bytes
    mov     buf_prn,200                         ;H
    call    prn_1
    mov     buf_prn,205
    mov     cx,69                               ;M
buttom_1:   call    prn_1
```

```
        loop    buttom_1
        mov     buf_prn,188                     ;<
        call    prn_1
        mov     buf_prn,0ch                     ;form feed
        call    prn_1
        ret
line_buttom endp

;check ascii is 1-32 and load font, print character
prn_font    proc
        pusha
        cmp     al,1
        jc      prn_font1
        cmp     al,31
        jnc     prn_font1
        push    ax
        @Write  defin_chr,8,p_handle
        pop     ax
        mov     bl,36
        mul     bl
        mov     dx,offset font
        add     dx,ax
        mov     cx,36
        call    prn_bytes                       ;load font to printe
        mov     al,";"
prn_font1:
        mov    print_com,al
        @Write  print_com,1,P_handle
        popa
```

```
    ret
prn_font    endp

  END    start
```