เอกสารอ้างอิง

1. Andrew S. Tanenbaum, Computer Network, Printice-Hall Inc., Englewood Cliffs, New Jersey, 1981.
2. IBM, "IBM Synchronous Data Link Control." IBM Corporation, North Carolina, 1979.
3. Elizabeth A. Nichols, Joseph C. Nichols, Keith R. Mussion, Data Communications for Microcomputers with Practical Applications and Experiments, McGraw-Hill Book Company, New York, 1982.
4. Lewis C. Eggebrecht, Interfacing to the IBM personal computer, Howard W. Sams & Co., Inc., Indiana, 1983.
5. IBM, "Technical Reference," IBM Corporation, revised edition, 1983.
6. Kai Hwang, Faye A. Briggs, Computer Architecture and Parallel Processing, pp. 1-27, McGraw-Hill Book Company, New York, 1985.
7. ยืน ภู่วรวรรณ, วัฒนา เชียงกูล, ไมโครโปรเซสเซอร์ ไมโครคอมพิวเตอร์, บริษัท ซีเอ็ดยูเคชั่น จำกัด, กรุงเทพมหานคร, พิมพ์ครั้งที่ 1, 2524.
8. SGS, "Z-80 Microprocessor Family Databook," SGS-ATES group of Companies, Italy, 1982.
9. Intel, "Microprocessors and Peripherals volume 1," Intel Corporation, Santa Clara, California, 1985.
10. Intel, "Microcommunication Handbook," Intel Corporation, Santa Clara, California, 1987.
11. Lance A. Leventhal, Z-80 Assembly Language Programming, McGraw-Hill Book Company, Osborne/McGraw-Hill, Berkeley, California, 1986.

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

13. Borland, "Turbo Pascal Version 3.0 Reference Manual," Borland International Inc., Scotts Valley, California, 1985.

14. Paul M. Dunphy, "IBM PC Interrupt Service Routines," Byte Inside The IBM PC, 223-227, 1985.

ภาคผนวก ก

พอร์ทที่ใช้ในการตรวจสอบโพรโทคอล SDLC

ดังจะเห็นได้จากบทที่ 2 แล้วว่าโพรโทคอล SDLC มีรูปแบบที่เป็นมาตรฐานซึ่งประกอบไปด้วยเขตที่ต่างชนิดกัน ทำให้การรับส่งข้อมูลมีได้หลายรูปแบบรวมทั้งสถานะ idle หรือ active ที่มีการส่งแฟลกอยู่ตลอดเวลารวมทั้งวิธีการแทรกบิตศูนย์ลงไปในข้อมูล ถ้าภาระต่างๆนี้ถูกจัดการโดยซีพียู (CPU) ทั้งหมดจะมีความยุ่งยากมาก เนื่องจากซีพียูใช้กับงานควบคุมหรือประมวลผลทั่ว ๆ ไป ถ้าจะนำมาใช้กับการตรวจสอบโพรโทคอลนี้จะต้องใช้เวลาในการแยกแยะและจัดการเกี่ยวกับข้อมูลเป็นอันมาก เช่น การดึงบิตศูนย์ออกจากการแทรกมาจากต้นทาง การตรวจสอบ FCS ฯลฯ แต่ถ้าใช้พอร์ทที่มีหน้าที่เฉพาะอย่างจะทำให้การทำงานสะดวกขึ้นและจะลดภาระของซีพียูลงทำให้ซีพียูมีเวลาพอที่จะไปทำงานต่าง ๆ ในระบบได้

ในบทนี้จะขอกล่าวถึงไอซี (IC) เบอร์ 8273 ซึ่งถูกนำมาใช้เป็นพอร์ทในการตรวจสอบและแยกแยะส่วนต่าง ๆ ของโพรโทคอลชนิด SDLC และผลที่ได้จะถูกส่งไปยังซีพียูเพื่อประมวลผลต่อไป

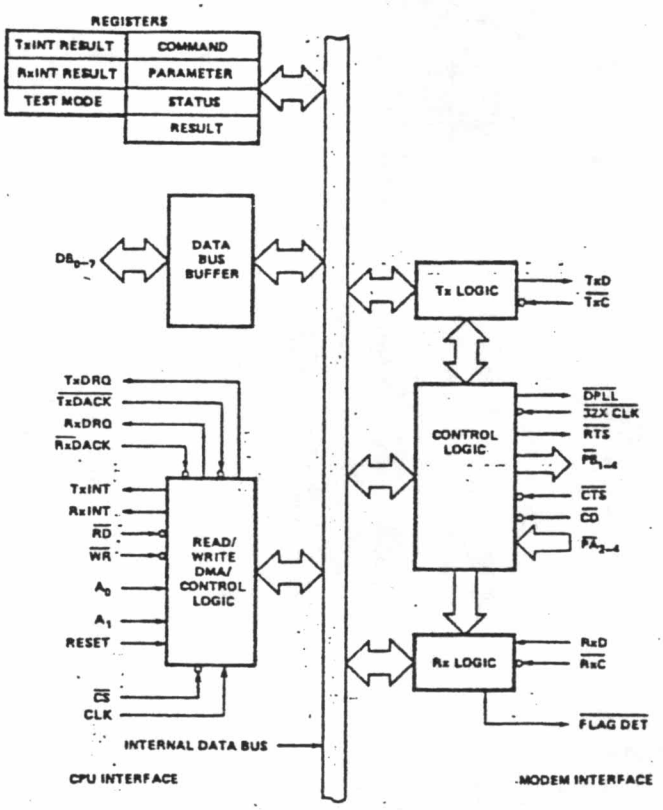
ก.1 ส่วนประกอบของ 8273

ไอซี (IC) เบอร์ 8273 เป็นอุปกรณ์สนับสนุนที่มีความสามารถสูงทางด้านโพรโทคอลชนิด SDLC ดังนั้นภาระต่าง ๆ ที่เกี่ยวข้องกับโพรโทคอลนี้ 8273 จึงรับหน้าที่ไปทำทั้งหมด 8273 จะถูกควบคุมโดยการเขียนคำสั่งลงในรีจิสเตอร์ (register) ต่าง ๆ ซึ่งอยู่ภายใน 8273 ส่วนประกอบภายใน 8273 แสดงเป็นแผนภาพกรอบ ดังรูปที่ ก.1

ก.2 รีจิสเตอร์ภายใน 8273

ภายใน 8273 ประกอบด้วยรีจิสเตอร์ทั้งหมด 7 รีจิสเตอร์ซึ่งแสดงดังรูปที่ ก.1 แต่ละรีจิสเตอร์มีหน้าที่การทำงานดังนี้

1. command register เป็นรีจิสเตอร์ที่ถูกใช้ในการเขียนชุดคำสั่งที่ใช้ควบคุมการทำงานของ 8273



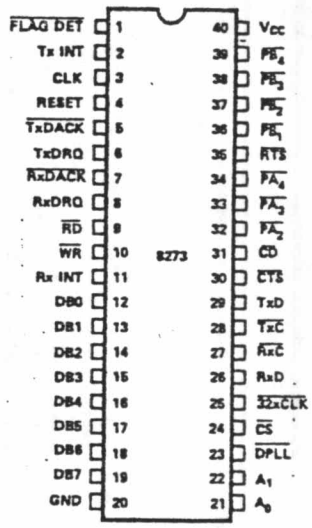


Figure 2. Configuration

รูปที่ ก.1 แสดงส่วนประกอบภายใน 8273

2. Parameter register คำสั่งบางคำสั่งถูกเขียนลงใน command register แต่ต้องการข้อมูลเพิ่มเติมจึงต้องถูกเขียนลงใน register นี้

3. Status register ใช้ในการแสดงสถานภาพที่เกิดขึ้นเกี่ยวกับการป้อนคำสั่งหรือการขัดจังหวะ (interrupt)

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

5. TxINT register (Transmit Interrupt Result Register) ใช้แสดงผลที่เกิดจากขบวนการส่งข้อมูล

6. RxINT register (Receive Interrupt Result Register) ใช้แสดงผลที่เกิดจากขบวนการรับข้อมูล

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

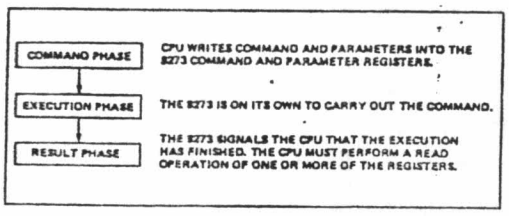
การอ้างถึงรีจิสเตอร์ต่าง ๆ นี้จะอ้างโดยขาแอดเดรส A0 และ A1 ดังตารางที่ ก.1

A_1	A_0	TxDACK	RxDACK	CS	RD	WR	Register
0	0	1	1	0	1	0	Command
0	0	1	1	0	0	1	Status
0	1	1	1	0	1	0	Parameter
0	1	1	1	0	0	1	Result
1	0	1	1	0	1	0	Reset
1	0	1	1	0	0	1	TxINT Result
1	1	1	1	0	1	0	—
1	1	1	1	0	0	1	RxINT Result
X	X	0	1	1	1	0	Transmit Data
X	X	1	0	1	0	1	Receive Data

ตารางที่ ก.1 ตำแหน่งของรีจิสเตอร์

ก.3 การทำงานของ 8273

ประกอบไปด้วยการทำงาน 3 ขั้นตอนดังรูปที่ ก.2 ซึ่งมีรายละเอียดดังนี้คือ



รูปที่ ก.2 แสดงขั้นตอนการทำงานของ 8273

5. TxINT register (Transmit Interrupt Result Register) ใช้แสดงผลที่เกิดจากขบวนการส่งข้อมูล

6. RxINT register (Receive Interrupt Result Register) ใช้แสดงผลที่เกิดจากขบวนการรับข้อมูล

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

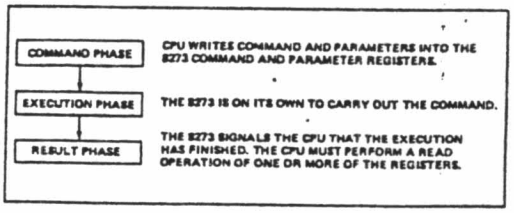
การอ้างถึงรีจิสเตอร์ต่าง ๆ นี้จะอ้างโดยขาแอดเดรส A0 และ A1 ดังตารางที่ ก.1

A_1	A_0	TxDACK	RxDACK	CS	RD	WR	Register
0	0	1	1	0	1	0	Command
0	0	1	1	0	0	1	Status
0	1	1	1	0	1	0	Parameter
0	1	1	1	0	0	1	Result
1	0	1	1	0	1	0	Reset
1	0	1	1	0	0	1	TxINT Result
1	1	1	1	0	1	0	—
1	1	1	1	0	0	1	RxINT Result
X	X	0	1	1	1	0	Transmit Data
X	X	1	0	1	0	1	Receive Data

ตารางที่ ก.1 ตำแหน่งของรีจิสเตอร์

ก.3 การทำงานของ 8273

ประกอบไปด้วยการทำงาน 3 ขั้นตอนดังรูปที่ ก.2 ซึ่งมีรายละเอียดดังนี้คือ



รูปที่ ก.2 แสดงขั้นตอนการทำงานของ 8273

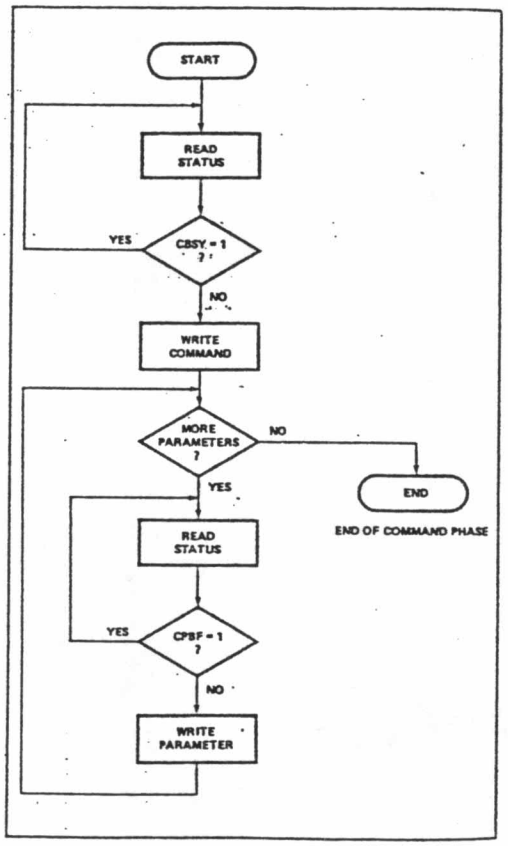
1. Command phase เป็นขั้นตอนที่เขียนคำสั่ง (command) และ พารามิเตอร์ (parameter) ลงในรีจิสเตอร์ทั้งสองตามลำดับ แต่การเขียนจะต้องมีข้อแม้ว่าจะต้องเขียนคำสั่งแรกให้เรียบร้อยก่อนแล้วจึงเขียนคำสั่งต่อไปได้ แผนภูมิสายงาน (flowchart) ในรูปที่ ก.3 แสดงถึงขั้นตอนในการเขียน command phase ถ้าการเขียนไม่เป็นไปตามลำดับจะทำให้เกิดการทำงานผิดพลาดขึ้นได้

2. Execution phase เป็นขั้นตอนที่ 8273 ทำการประมวลผลกับคำสั่งต่าง ๆ ที่ถูกเขียนลงไปและจะเริ่มทำการรับส่งข้อมูลตั้งแต่แฟลกเริ่มต้นจนถึงแฟลกปิดท้าย

3. Result phase เป็นขั้นตอนที่ 8273 แจ้งถึงผลที่ได้จากการรับส่งข้อมูลไปยังซีพียูโดยวิธีขัดจังหวะ ซึ่งผลที่ได้นั้นอาจจะสมบูรณ์หรือไม่ก็ได้ ผลที่ 8273 แจ้งต่อซีพียู แบ่งเป็น 2 ประเภทคือ

ก) immediate result เป็นผลที่ได้จากการที่ 8273 ทำคำสั่งเกี่ยวกับพอร์ท A และพอร์ท B ผลที่ได้จะนำไปเก็บไว้ใน result register

ข) non-immediate result เป็นผลที่เกิดจาก 8273 ทำการขัดจังหวะซีพียู ผลลัพธ์ชนิดนี้จะถูกนำไปเก็บใน TxINT หรือ RxINT register ขึ้นอยู่กับการขัดจังหวะว่ามาจากภาคส่งหรือภาครับ



รูปที่ ก.3 การเขียน command phase

ก.4 หน้าที่ของแต่ละบิตภายใน command register

จะเห็นได้จากรูปที่ ก.3 ว่าการเขียนคำสั่งลงใน 8273 ต้องขึ้นอยู่กับสถานะที่เกิดขึ้นภายใน 8273 ด้วย เช่น CBSY, CPBF ฯลฯ ตำแหน่งของบิตต่าง ๆ ที่อยู่ใน status register แสดงในรูปที่ ก.4

D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0
CBSY	CBF	CPBF	CRBF	RxINT	TxINT	RxIRA	TxIRA

รูปที่ ก.4 บิตที่ใช้แสดงสถานะของ 8273

1. Bit 7 Command busy (CBSY) เป็นบิตที่ใช้แสดงว่า command register เต็มหรือไม่ ถ้า CBSY เท่ากับหนึ่งแสดงว่าเต็ม CBSY จะถูก reset ก็ต่อเมื่อการทำงานในช่วงของ command phase ได้เสร็จสิ้นแล้ว อนึ่งขณะที่ CBSY ยังคงมีค่าเท่ากับหนึ่ง ถ้ามีการเขียนคำสั่งลงใน command register อีกจะทำให้เกิดการผิดพลาดขึ้นได้

2. Bit 6 Command buffer full (CBF) เป็นบิตที่ใช้แสดงว่า command register เต็มหรือไม่ ถ้า CBF เท่ากับหนึ่งแสดงว่าเต็ม CBF จะถูก reset หลังจาก 8273 ยอมรับคำสั่งที่ถูกเขียนลงไปแต่จะไม่ยืนยันว่า 8273 ได้เข้าสู่ execution phase แล้ว

3. Bit 5 Command parameter buffer full (CPBF) CPBF เท่ากับหนึ่งเมื่อ parameter register เต็มและจะถูก reset เมื่อ 8273 ยอมรับพารามิเตอร์ที่ถูกเขียนลงไป ในกรณีที่คำสั่งบางชนิดต้องการพารามิเตอร์มากกว่า 1 ค่า ซีพียูจะทำการตรวจสอบบิตนี้ก่อนที่จะเขียนพารามิเตอร์ตัวต่อไป

4. Bit 4 Command result buffer full (CRBF) CRBF เท่ากับหนึ่งเมื่อ immediate result ถูกเก็บใน result register เรียบร้อยแล้วและจะถูก reset เมื่อซีพียูอ่าน result register

5. Bit 3 Receiver interrupt (RxINT) เป็นบิตที่ใช้บอกว่าขณะนี้ได้มีการขัดจังหวะเกิดขึ้นที่ภาครับ บิตนี้จะแสดงสถานภาพเชิงตรรกเช่นเดียวกับขาที่ 11 และจะถูก reset หลังจากซีพียูอ่านผลลัพธ์ที่เกิดขึ้นจาก RxINT register หรือซีพียูอ่านข้อมูลจาก receive data register

6. Bit 2 Transmitter interrupt (TxINT) เป็นบิตที่ 8273 ใช้บอกว่าขณะนี้ได้มีการขัดจังหวะ เกิดขึ้นทางภาคส่ง บิตนี้จะแสดงสถานภาพเชิงตรรกเช่นเดียวกับขาที่ 2 และจะถูก reset หลังจากซีพียูอ่านผลลัพธ์ที่เกิดจาก TxINT register หรือซีพียูทำการส่งข้อมูลจากหน่วยความจำไปยัง transmit data register

7. Bit 1 Receiver interrupt result available (RxIRA) RxIRA เท่ากับหนึ่งเมื่อผลลัพธ์ที่ได้จากการขัดจังหวะทางภาครับถูกเก็บลงใน RxINT register เรียบร้อยแล้วและจะถูก reset หลังจากซีพียูอ่านผลลัพธ์นั้นจาก RxINT register

8. Bit 0 Transmitter interrupt result available (TxIRA) TxIRA เท่ากับหนึ่งเมื่อผลลัพธ์ที่ได้จากการขัดจังหวะทางภาคส่งถูกเก็บลงใน TxINT register เรียบ

ร้อยแล้วและจะถูก reset หลังจากซีพียูอ่านผลลัพธ์นั้นจาก TxINT register

ก.5 การเขียนและลบคำสั่งที่ใช้ควบคุม 8273

คำสั่งที่ใช้ควบคุมการทำงานของ 8273 สามารถแบ่งออกได้ดังนี้คือ

1. Set one bit delay มีรหัสของคำสั่งและพารามิเตอร์เป็น A4H และ 80H ตามลำดับ ถ้า 8273 ถูกกำหนดให้อยู่ในโมด (mode) นี้ 8273 จะทำการส่งข้อมูลที่รับมาได้โดยจะ delay ระยะเวลาไปเท่ากับหนึ่งบิต โมดนี้ใช้เฉพาะการติดต่อประเภท loop เท่านั้น

2. Reset one bit delay มีรหัสดังนี้ CMD (command) = 64H, PAR (parameter) = 7FH คำสั่งนี้จะควบคุมให้ 8273 หยุดการส่งข้อมูลตามลักษณะในข้อ 1

3. Set data transfer mode มีรหัสคือ CMD = 97H, PAR = 01H เมื่อ 8273 ถูกกำหนดให้อยู่ในโมดนี้ 8273 จะทำการขัดจังหวะซีพียูเมื่อต้องการจะส่งข้อมูลหรือมีข้อมูลมาคอยอยู่ใน receive data register ในโมดของการส่งรับตามลำดับ ถ้ามีการขัดจังหวะเกิดขึ้นทางภาคส่งและสถานะแสดงว่าไม่มีผลเกิดขึ้นจากการส่งข้อมูล (TxIRA = 0) การขัดจังหวะนี้แสดงว่า ต้องการให้ซีพียูส่งข้อมูลที่จะทำการส่งมายัง 8273 ถ้าการขัดจังหวะเกิดขึ้นทางภาครับและสถานะแสดงว่าไม่มีผลเกิดจากการรับข้อมูล (RxIRA = 0) กรณีนี้จะเป็นการขัดจังหวะที่ต้องการให้ซีพียูอ่านข้อมูลไปจาก 8273

4. Reset data transfer mode มีรหัสคือ CMD = 57H, PAR = 0FEH ถ้า 8273 ถูกกำหนดให้อยู่ในโมดนี้จะไม่มีการขัดจังหวะโดยตรงต่อซีพียู แต่จะทำให้ 8273 ติดต่อกับหน่วยความจำโดยวิธีการ DMA (Direct Memory Access)

5. Set operating mode มีรหัสคือ CMD = 91H, PAR ขึ้นอยู่กับการเลือกโมดต่าง ๆ ซึ่งแบ่งได้ดังนี้

ก) bit D5 HDLC mode ถ้า D5 = "0" 8273 จะถูกกำหนดให้มีการรับส่งข้อมูลเป็นโพรโทคอลชนิด SDLC ถ้า D5 = "1" จะเป็นโพรโทคอลชนิด HDLC

ข) bit D4 EOP interrupt mode ถ้า D4 = "1" 8273 จะทำการขัดจังหวะซีพียูเมื่อทางภาครับตรวจพบรหัส EOP (01111111) คำสั่งนี้จะถูกใช้ในการรับส่งข้อมูลแบบ loop

ค) bit D3 transmitter early interrupt mode ถ้า D3 = "1" 8273 จะสร้างสัญญานขัดจังหวะขณะที่ข้อมูลตัวสุดท้ายได้ผ่าน 8273 ออกไป จะมีประโยชน์ในกรณีที่ต้องการส่งข้อมูลติดต่อกันโดยที่มีเพียง 1 แฟลกเท่านั้นที่คั่นระหว่างข้อมูลทั้งสองเฟรม แต่ถ้า D3 = "0" 8273 จะสร้างสัญญานขัดจังหวะหลังจากแฟลกปิดท้ายได้ถูกส่งออกไปแล้ว

ง) bit D2 buffered mode ถ้า D2 = "1" ข้อมูลสองไบต์แรกที่อยู่ต่อมาจากแฟลกเปิดจะถูกเก็บในบัฟเฟอร์ (buffer) เพราะว่าข้อมูลทั้งสองไบต์นี้คือ address และ control field ถ้า D2 = "0" 8273 จะถือว่าทั้งสองไบต์นี้เป็นข้อมูลทั่ว ๆ ไป

จ) bit D1 preframe sync mode ถ้า D1 = "1" 8273 จะส่งตัวอักษรออกไปสองตัวก่อนที่จะส่งแฟลกเปิด ในกรณีที่ข้อมูลในรหัสของ Non-Return to Zero Invert (NRZI) ตัวอักษรนี้คือ 00H แต่ถ้าเป็นแบบ Non-Return to Zero (NRZ) ตัวอักษรนี้คือ 55H ถ้า D1 = "0" 8273 จะไม่ส่งตัวอักษรก่อนแต่จะส่งแฟลกทันที

ฉ) bit D0 flag stream mode ถ้า D0="1" สถานภาพของการส่งข้อมูลจะเป็นไปตามตารางที่ ก.2 แต่ถ้า D0 = " 0 " จะเป็นไปตามตารางที่ ก.3

ตารางที่ ก.2 แสดงการส่งข้อมูลเมื่อ D0 = "1"

Transmitter State	Action
Idle	Send Flags Immediately.
Transmit or Transmit Transparent Active	Send Flags After the Transmission Complete
Loop Transmit Active	Ignore Command.
1 Bit Delay Active	Ignore Command.

ตารางที่ ก.3 แสดงการส่งข้อมูลเมื่อ D0 = "0"

Transmitter State	Action
IDLE	Sends Idles on Next Character boundary.
Transmit or Transmit-Transparent Active	Send Idles after the Transmission is Complete.
Loop Transmit Active	Ignore Command.
1 Bit Delay Active	Ignore Command.

6. Reset operating mode มีรหัสคือ CMD = 51H,PAR ขึ้นอยู่กับความต้องการของการใช้งานถ้าจะ reset การทำงานแบบใดก็ให้บิตนั้น = "0"

7. Set serial I/O mode มีรหัสคือ CMD = A0H,PAR ขึ้นอยู่กับการเลือกโมดต่าง ๆ ซึ่งแบ่งได้ดังนี้

ก) bit D2 loopback ถ้า D2 = "1" ข้อมูลที่ถูกส่งออกไปจะถูกผ่านภายในไปยังภาครับของ 8273 ด้วย แต่ถ้า D2 = "0" จะไม่มีการผ่านภายในของข้อมูล

ข) bit D1 TxC --> RxC ถ้า D1 = "1" สัญญาณนาฬิกาที่ใช้ในการส่งข้อมูลจะถูกผ่านไปยังภาครับสัญญาณนาฬิกาของ 8273 ด้วย แต่ถ้า D1 = "0" จะไม่มีการผ่านภายในของสัญญาณนาฬิกา

ค) bit D0 NRZI mode ถ้า D0 = "1" 8273 จะถูกเข้ารหัสและถอดรหัสในการส่งและรับข้อมูลเป็นแบบ NRZI แต่ถ้า D0 = "0" 8273 จะรับส่งข้อมูลในแบบธรรมดาคือ NRZ

8. Reset serial I/O mode มีรหัสคือ CMD = 60H,PAR ขึ้นอยู่กับความต้องการของการใช้งานถ้าจะ reset แบบใดก็ให้บิตนั้น = "0"

9. Reset device command มีรหัสคือ TMR = 01H, TMR = 00H ที่ใช้ TMR เนื่องจากว่าในการสั่ง reset 8273 จะต้องเขียนคำสั่งลงใน reset register ซึ่งมีตำแหน่งอยู่ที่ 02H การ reset 8273 โดยใช้ซอฟต์แวร์นี้จะมีผลกระทบต่อ 8273 ดังนี้คือ

- สัญญาณที่ใช้ในการควบคุมโมเดม (modem) จะถูกกำหนดให้เป็น "1"
- สถานภาพของ status register จะถูก reset = "0"
- คำสั่งต่าง ๆ ที่ถูกเขียนลงใน 8273 จะถูกเลิกกระทำทันที
- 8273 จะอยู่ในสถานะ idle จนกระทั่งถูกเขียนคำสั่งใหม่ลงไป
- คำสั่งที่อยู่ในโมดของ serial I/O operating ถูกกำหนดให้เท่ากับศูนย์และจะถูกกำหนดให้อยู่ในโมดของ DMA
- 8273 ถูกกำหนดให้ทำงานแบบ non-loop SDLC

ก.6 คำสั่งที่ใช้ในการรับข้อมูล (Receive Commands)

8273 จะแบ่งการกำหนดคำสั่งในการรับข้อมูลเป็น 3 แบบคือ

ก.6.1 General receive มีรหัสคือ CMD = C0H การรับข้อมูลในโมดนี้จะไม่คำนึงถึงแอดเดรสที่แจ้งมาในเฟรม คำสั่งนี้ใช้กับสถานีปฐมภูมิหรือสถานีควบคุม loop ส่วนพารามิเตอร์จะแบ่งเป็นสองไบต์ใช้ในการกำหนดขนาดของบัฟเฟอร์ในการรับข้อมูลโดยไบต์แรกจะแทน Least Significant Byte (LSB) และไบต์ที่สองแทน Most Significant Byte (MSB)
ข้อสังเกต

1. ถ้าในการรับข้อมูลนั้นถูกกำหนดให้อยู่ใน buffer mode R0,R1 จะแทนจำนวนของไบต์ข้อมูลที่รับได้

2. ถ้าในการรับข้อมูลนั้นถูกกำหนดให้อยู่ใน non-buffer mode R0,R1 แทนจำนวนไบต์ที่รับได้ทั้งหมดซึ่งจะรวม address และ control byte ด้วย

3. FCS จะไม่ถูกส่งไปยังหน่วยความจำ

4. เฟรมที่มีจำนวนบิทระหว่างแฟลกเริ่มต้นและแฟลกปิดท้ายน้อยกว่า 32 บิตจะไม่ถูกสนใจโดย 8273 ในกรณีที่อยู่ใน buffer mode

5. ในกรณีที่อยู่ใน non-buffer mode ถ้า 8273 ได้รับเฟรมที่มีจำนวนบิตน้อยกว่า 32 บิตจะส่งสัญญาณขัดจังหวะไปยังซีพียู

6. 8273 จะหยุดรับข้อมูลทุกครั้งที่รับรหัส idle ได้หลังจากที่รับเฟรมที่สมบูรณ์เรียบร้อยแล้ว ดังนั้นถ้าต้องการให้ 8273 รับข้อมูลใหม่จะต้องใช้คำสั่ง re-enable

7. ถ้ารหัสของ abort character อยู่ระหว่างแฟลกปิดท้ายกับ idle 8273 จะไม่ส่งสัญญาณขัดจังหวะไปยังซีพียู

8. ถ้ารหัส abort character ไม่ถูกนำหน้าด้วยแฟลกและตามด้วย idle 8273 จะส่งสัญญาณขัดจังหวะไปยังซีพียูและตามด้วย idle interrupt ในกรณีที่ 8273 รับ abort character ได้มันจะหยุดรับข้อมูลทันที

ก.6.2 Selective receive มีรหัสคือ CMD = C1H การรับข้อมูลในโมดนี้จะไม่สนใจถ้าเฟรมนั้นมีแอดเดรสไม่ตรงกับแอดเดรสที่ถูกกำหนดไว้ที่สถานีนั้น พารามิเตอร์ที่ใช้มีทั้ง

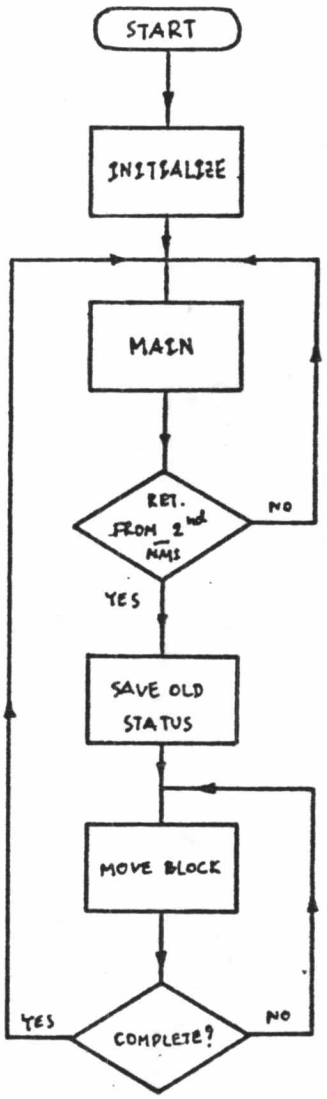
หมด 4 ไบต์โดยที่สองไบต์แรกเป็นตัวบอกขนาดของบัฟเฟอร์ ไบต์ที่สามใช้กำหนดแอดเดรสที่ต้องการจะรับข้อมูล ไบต์ที่สี่ใช้ในการรับข้อมูลชนิด broadcast ในกรณีที่ไม่ต้องการรับข้อมูลชนิด broadcast จะกำหนดให้ไบต์ที่สี่มี แอดเดรสเช่นเดียวกับไบต์ที่สาม

ก.6.3 Selective loop receive มีรหัสคือ CMD = C2H การรับข้อมูลในโมดนี้จะเหมือนกับ selective receive และการกำหนดพารามิเตอร์ก็เช่นกันแต่จะมีข้อแตกต่างอยู่ที่การรับข้อมูลในโมดนี้จะทำการตรวจสอบรหัส End Of Poll (EOP) เมื่อตรวจพบ EOP แล้วก็จะทำการส่งข้อมูลทันที

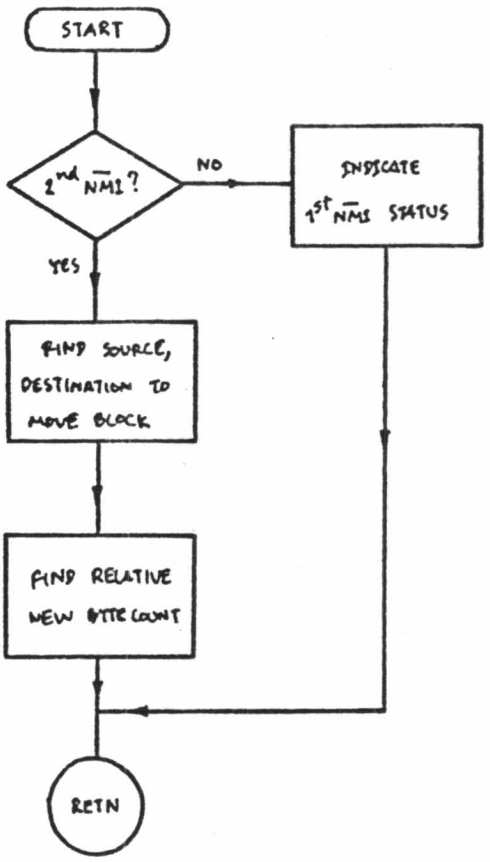
ก.6.4 Receive disable มีรหัสคือ CMD = C5H พารามิเตอร์ไม่ต้องใช้ เมื่อใช้คำสั่งนี้ 8273 จะหยุดการทำงานของภาครับทันที

เนื่องจากการตรวจสอบโพรโทคอลเป็นการรับข้อมูลอย่างเดียว ดังนั้นฟังก์ชันที่ใช้กำหนดการทำงานของ 8273 จึงเป็นฟังก์ชันที่เกี่ยวกับการรับข้อมูลเท่านั้น

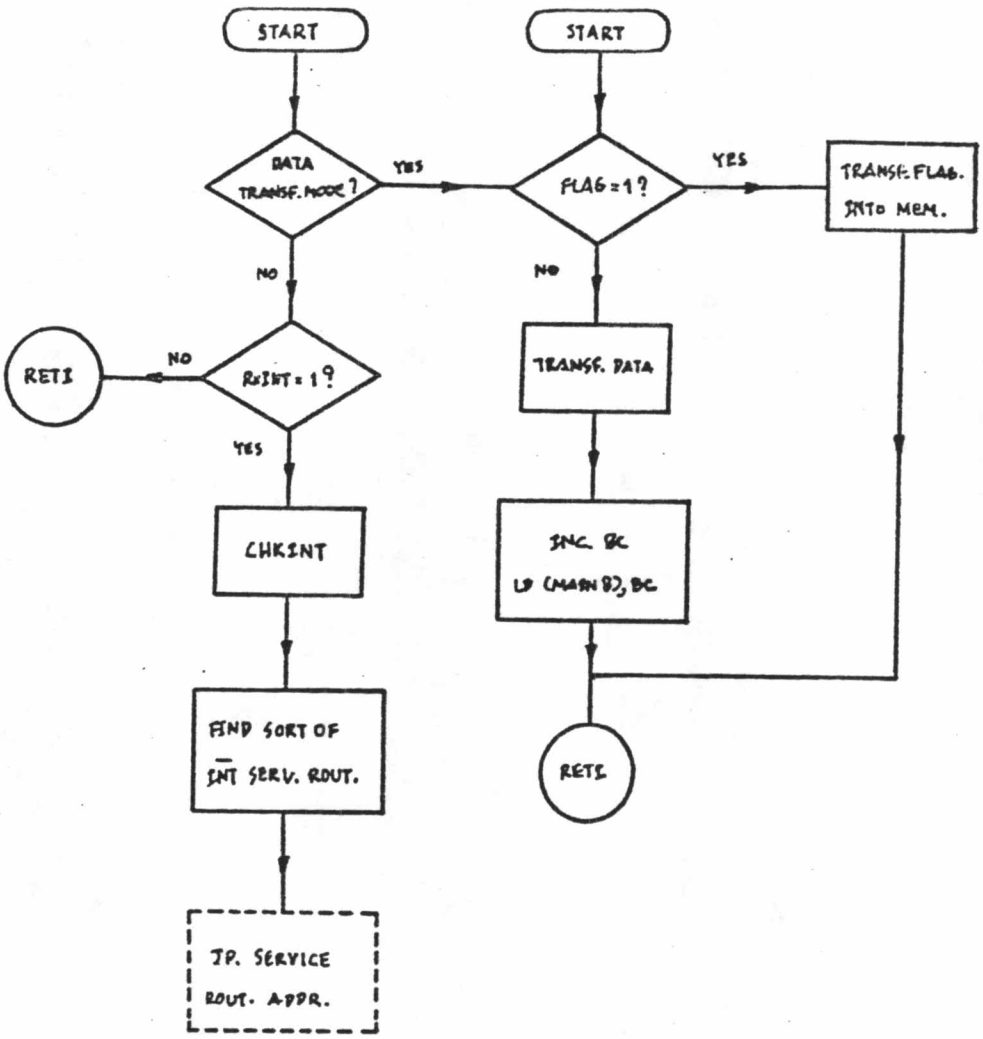
ภาคผนวก ข



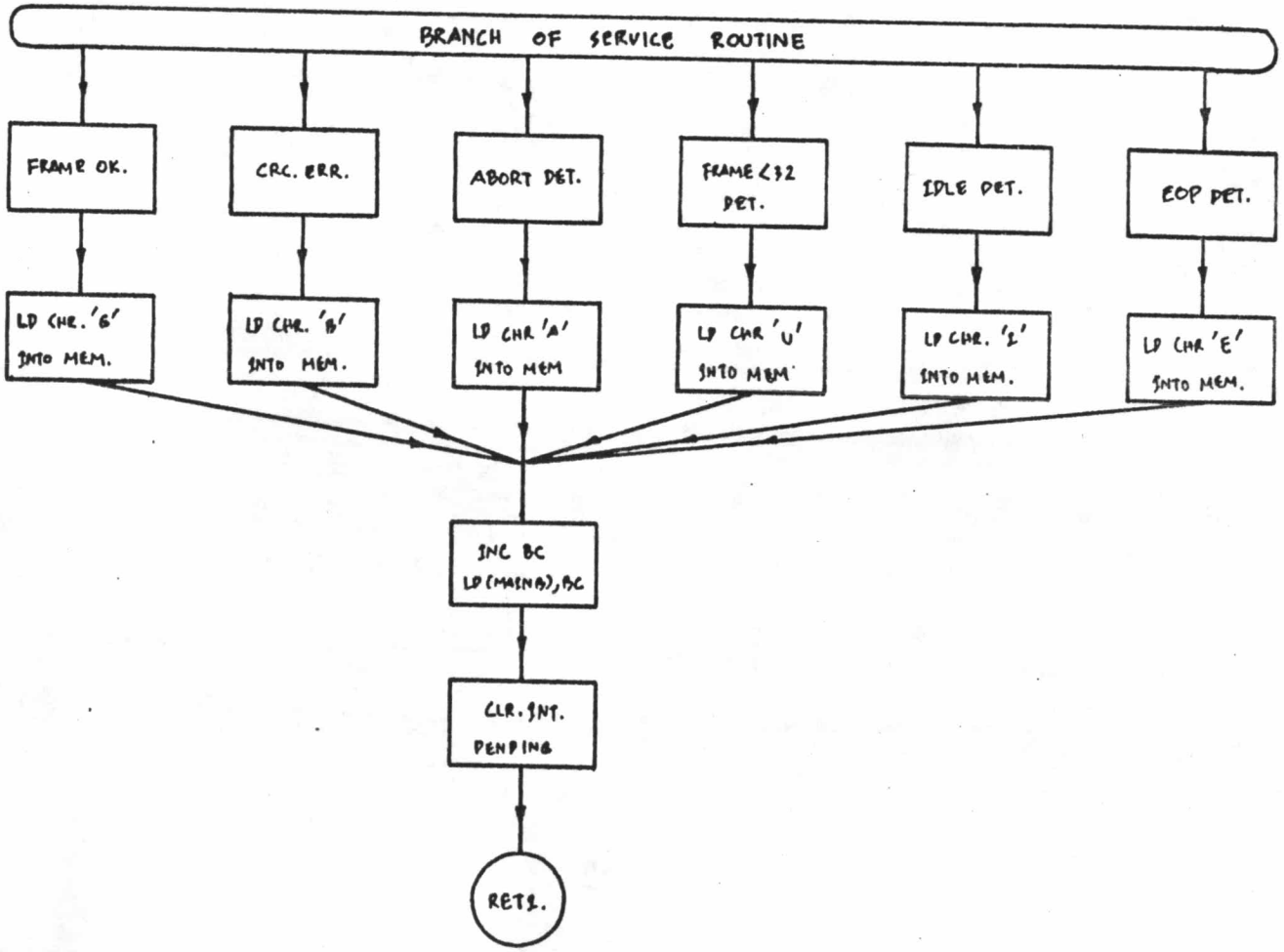
แผนภูมิสายงานโปรแกรมหลัก



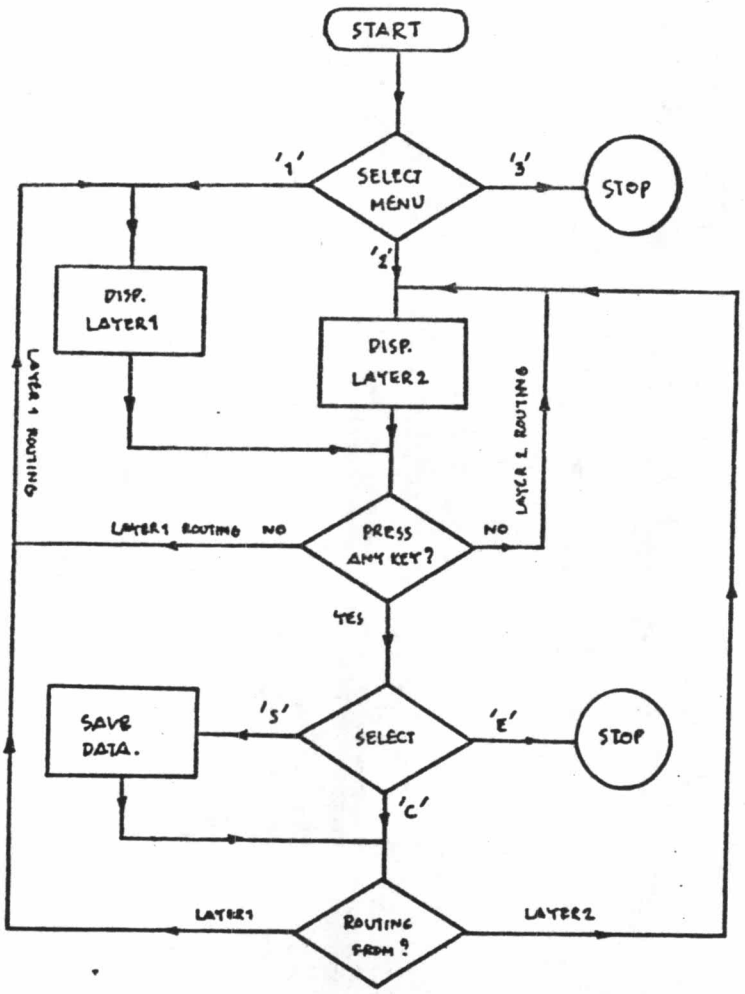
แผนภูมิสายงาน Non-Maskable Interrupt service routine



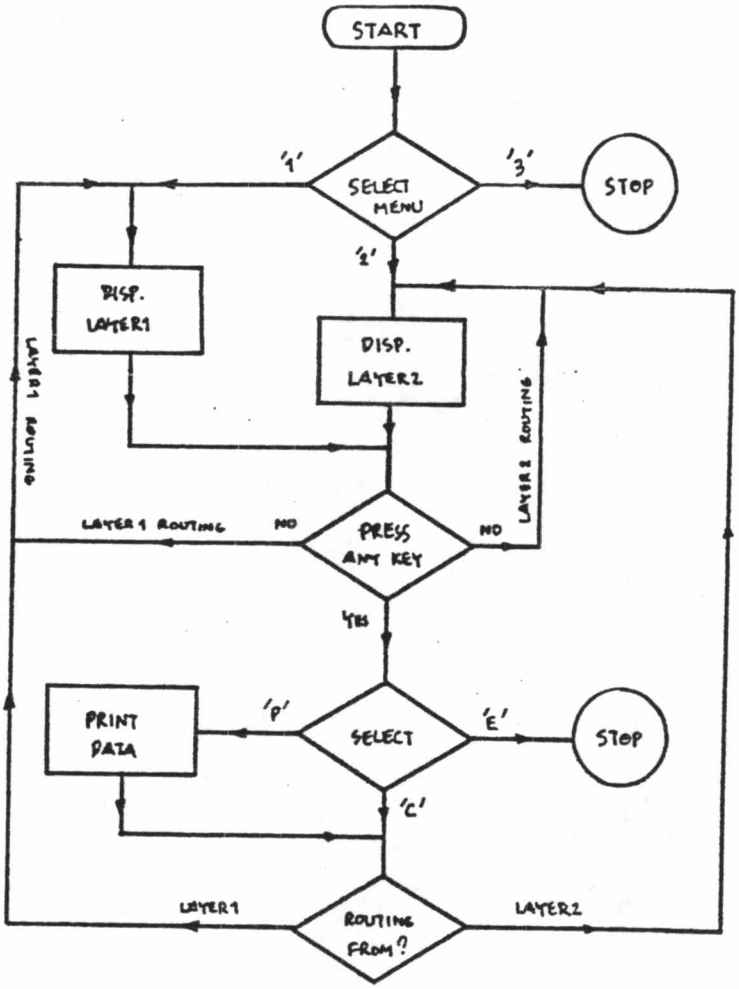
แผนภูมิสายงาน Maskable Interrupt service routine



แผนภูมิสายงานการตรวจสอบสถานะต่าง ๆ



แผนภูมิสายงานโปรแกรมแสดงผลและบันทึกข้อมูล



แผนภูมิสายงานโปรแกรมพิมพ์ผล

ภาคผนวก ค

```
program SDLC_monitoring;
 type
  ebc = array [$0..$ff] of string[3];
  box = array [0..32767] of byte;
  result = record
            ax,bx,cx,dx,bp,si,di,ds,es,flags : integer;
           end;
 var
  oldvector1,oldvectorh,am1,am2,count1,count2:integer;
  x,y:string[75];z:string[78];typ:string[4];ch,alph:string[1];
  pf,ns,nr,imr,imr1,mask,addr,fcs,ind,openf1,openf2,flg1,flg2,check1,check2:byte;
  box1:box absolute $2000:$0000;
  box2:box absolute $2000:$8000;
  zbox1:box absolute $4000:$0000;
  zbox2:box absolute $4000:$8000;
  i88:byte absolute $2000:$0000;
  z80:byte absolute $5000:$7000;
  z801:byte absolute $5000:$b000;
  z802:byte absolute $5000:$3000;
  z803 : byte absolute $5000:$d000;
  z804 : byte absolute $5000:$f000;
  dsave:integer absolute cseg:$0006;
  linind,i,offset,segment,first_word,second_word:integer;
  dest1,dest2:file;
  res:result;
 const
  ebd:ebc
  = ('NUL','SOH','STX','ETX','PF','HT','LC','DEL','','',
  'SMM','VT','FF','CR','SO','SI','DLE','DC1','DC2','TM',
  'RES','NL','BS','IL','CAN','EM','CC','CU1','IFS','IGS',
  'IRS','IUS','DS','SOS','FS','','BYP','LF','ETB','ESC',
  '','','SM','CU2','','ENQ','ACK','BEL','','',
  'SYN','','PN','RS','UC','EOT','','','','CU3',
  'DC4','NAK','','SUB','SP','','','','','',
  '','','','','','','','','(','(','+',
  '|','&','','','','','','','','','',
  '','!','$','*',')',';','^','-','/','',
  '','','','','','','','','','','',
  '','','','','','','','','','','',
  '','','?','','','','','','','','',
  '','','','',':','#','@','''','=','"',
  '','a','b','c','d','e','f','g','h','i',
  '','','','','','','','j','k','l',
  'm','n','o','p','q','r','','','','',
  '','','','','s','t','u','v','w','x',
  'y','z','','','','','','','','',
  '','','','','','','','','','',
  '','','','','','A','B','C','D',
  'E','F','G','H','I','','','','','',
  '','J','K','L','M','N','O','P','Q',
  'R','G','K','','','','','','','S',
  'T','U','V','W','X','Y','Z','F','B','0',
  'I','E','U','0','1','2','3','4','5','6',
  '7','8','9','','','','','','');
```

```
  {.pa}
procedure setd;
 begin
  move(i88,z804,1);delay(200);
  move(i88,z802,1);delay(200);
  move(i88,z803,1);
 end;
procedure amnt;
 begin
  am1 := (box1[32767]*256 + box1[32766]) - $2001;
  am2 := (box2[32767]*256 + box2[32766]) - $2001;
 end;
procedure disp;
 begin
  gotoxy(13,3);writeln('This software is used for SDLC protocol monitoring set');
 end;
procedure frame;
 begin
  clrscr;
  for i:= 12 to 67 do
  begin
   gotoxy(i,1);write('-');
   gotoxy(i,5);write('-');
  end;
  for i:= 2 to 4 do
  begin
   gotoxy(11,i);write('¦');
   gotoxy(68,i);write('¦');
  end;
  begin
   gotoxy(11,1);write('+');
   gotoxy(68,1);write('+');
   gotoxy(11,5);write('+');
   gotoxy(68,5);write('+');
  end;
 end;
procedure select;
 begin
  gotoxy(15,7);write('Display format');
  gotoxy(15,9);write(' 1) Layer 1 : DTE X...XX');
  gotoxy(27,11);write(': DCE X...XX');
  gotoxy(15,13);write(' 2) Layer 2 : FROM  ADDR.  N(S)  P/F  N(R)  TYPE  FCS');
  gotoxy(15,15);write(' 3) Quit');
  gotoxy(15,17);write('Select [ 1 , 2 or 3 ]');
 end;
procedure restorevector;
 begin
  memw[$0000:first_word] := oldvector1;
  memw[$0000:second_word] :=oldvectorh;
 end;
procedure i_s_r;  {interrupt service routine}
 begin
  inline($fb/$1e/$50/$53/$51/$52/$57/$56/$06);
```

```
  inline($8c/$c8/$8e/$d8/$a1/dsave/$8e/$d8);
  {.pa}
  move(i88,z80,1);
  move(zbox1,box1,$8000);
  move(zbox2,box2,$8000);
  move(i88,z801,1);
  move(i88,z802,1);
  move(i88,z802,1);
  restorevector;ind := 1;count1:=0;
  count2:=0;openf1:=0;openf2:=0;linind:=0;amnt;clrscr;
  port[$0020] := $20;
  inline($07/$5e/$5f/$5a/$59/$5b/$58/$1f/$cf);
 end;
procedure enable_IR0x;
 begin
  mask := not(1 shl 3);
  imr  := port[$21];imr1 := imr;
  imr  := imr and mask;
  port[$21] := imr;
 end;
procedure set_IVT; {set interrupt vector}
 begin
  offset := ofs(i_s_r) + 7;
  segment := cseg;
  first_word := (3 + 8)*4;
  second_word := first_word + 2;
  oldvector1:=  memw[$0000:first_word];
  oldvectorh:=  memw[$0000:second_word] ;
  memw[$0000:first_word] := offset;
  memw[$0000:second_word] := segment;
 end;
procedure savdat;    {write data to diskette}
 begin
  assign(dest1,'b:DTE.dat');assign(dest2,'b:DCE.dat');
  {$i-} reset(dest1,1); reset(dest2,1) {$i+};
  if ioresult<>0 then
   begin
    rewrite(dest1,1);rewrite(dest2,1);
   end
  else
   begin
    longseek(dest1,longfilesize(dest1));longseek(dest2,longfilesize(dest2));
   end;
  blockwrite(dest1,box1,am1);blockwrite(dest2,box2,am2);
  close(dest1);close(dest2);
 end;
procedure decis;
 begin
  writeln;writeln;
  write('S':15);lowvideo;
  write('ave');normvideo;
  write('C':8);lowvideo;
  write('ontinue');normvideo;
```

```
   write('E':8);lowvideo;
   write('nd');normvideo;
   {.pa}
   writeln;
   set_ivt;
   repeat
    read(kbd,ch);
    case ch of
     'S','s' : begin
                savdat;
               end;
     'C','c' : begin
               end;
     'E','e' : begin
               end;
    end;
   until upcase(ch) in ['S','C','E'];
 end;
procedure ondte;
 begin
  x:='';check1:=0;
  if count1 <= am1 then
  begin
  repeat
   x:= x+ebd[box1[count1]];
   if (length(x))72)  then
    begin
     delete(x,(length(x)+1)-length(ebd[box1[count1]]),length(ebd[box1[count1]]));
     check1:=1;
    end
   else
    begin
     count1:=count1+1;
    end;
  until (check1=1) or (count1)am1);
  end
  else
   begin
    x:='';
   end;
 end;
procedure ondce;
 begin
  y:='';check2:=0;
  if count2 <= am2 then
  begin
  repeat
   y:= y+ebd[box2[count2]];
   if (length(y))72)  then
    begin
     delete(y,(length(y)+1)-length(ebd[box2[count2]]),length(ebd[box2[count2]]));
     check2:=1;
    end
```

```
    else
     begin
      count2:=count2+1;
      {.pa}
     end;
   until (check2=1) or (count2>am2);
   end
   else
    begin
     y:='';
    end;
 end;
procedure onscr;  {display data : Layer1}
 begin
  clrscr;
  repeat
   set_ivt;
   ondte;
   z:='DTE '+x+chr(13)+chr(10);
   for i := 1 to length(z) do
   begin
    res.ax := ord(copy(z,i,1));
    res.ax := res.ax or $0e00;
    res.bx := 7;
    intr(16,res);
   end;
   ondce;
   z:='DCE '+y+chr(13)+chr(10);
   for i := 1 to length(z) do
   begin
    res.ax := ord(copy(z,i,1));
    res.ax := res.ax or $0e00;
    res.bx := 7;
    intr(16,res);
   end;
   linind := linind+1;
  until  keypressed or (linind=2731) ;
  writeln;write(' linind=  ',linind);
 end;
procedure twoside;
 begin
  clrscr;
  gotoxy(2,1);
  write('   FROM  ADDR.  N(S)  P/F  N(R)  TYPE  FCS ');
  writeln;
 end;
procedure nsend (comm1 : byte);
 begin
  case (comm1 and $0e) of
   $00 : ns := 0;
   $02 : ns := 1;
   $04 : ns := 2;
   $06 : ns := 3;
```

```
    $08 : ns := 4;
    $0a : ns := 5;
    $0c : ns := 6;
    $0e : ns := 7;
    {.pa}
   end;
  end;
 procedure pofi (comm1 : byte);
  begin
   case (comm1 and $10) of
    $00 : pf := 0;
    $10 : pf := 1;
   end;
  end;
 procedure nrec (comm1 : byte);
  begin
   case (comm1 and $e0) of
    $00 : nr := 0;
    $20 : nr := 1;
    $40 : nr := 2;
    $60 : nr := 3;
    $80 : nr := 4;
    $a0 : nr := 5;
    $c0 : nr := 6;
    $e0 : nr := 7;
   end;
  end;
 procedure untyp1 (comm1 : byte);
  begin
   case (comm1 and $ec) of
    $00 : typ := 'UI';
    $04 : typ := 'SIM';
    $0c : typ := 'DM';
    $20 : typ := 'UP';
    $40 : typ := 'DISC';
    $60 : typ := 'UA';
    $80 : typ := 'SNRM';
    $84 : typ := 'FRMR';
    $c4 : typ := 'CFGR';
    $e0 : typ := 'TEST';
    $ec : typ := 'BCN';
   end;
  end;
 procedure untyp2 (comm1 : byte);
  begin
   case (comm1 and $ec) of
    $00 : typ := 'UI';
    $04 : typ := 'RIM';
    $0c : typ := 'DM';
    $20 : typ := 'UP';
    $40 : typ := 'RD';
    $60 : typ := 'UA';
    $80 : typ := 'SNRM';
```

```
    $84 : typ := 'FRMR';
    $c4 : typ := 'CFGR';
    $e0 : typ := 'TEST';
    $ec : typ := 'BCN';
   end;
   {.pa}
  end;
 procedure suptyp (comm1 : byte);
  begin
   case (comm1 and $0c) of
    $00 : typ := 'RR';
    $04 : typ := 'REJ';
    $08 : typ := 'RNR';
   end;
  end;
 procedure classi1 ( comm1 : byte); {find control field}
  begin
   case ( comm1 and $03) of
    $00 : begin
           nsend(comm1);nrec(comm1);pofi(comm1);typ:='INF0';
          end;
    $01 : begin
           nrec(comm1);pofi(comm1);suptyp(comm1);ns:=0;
          end;
    $03 : begin
           pofi(comm1);untyp1(comm1);ns:=0;nr:=0;
          end;
   end;
  end;
 procedure classi2 ( comm1 : byte);
  begin
   case ( comm1 and $03) of
    $00 : begin
           nsend(comm1);nrec(comm1);pofi(comm1);typ:='INF0';
          end;
    $01 : begin
           nrec(comm1);pofi(comm1);suptyp(comm1);ns:=0;
          end;
    $03 : begin
           pofi(comm1);untyp2(comm1);ns:=0;nr:=0;
          end;
   end;
  end;
procedure twoside1;
 begin
  flg1:=0;set_ivt;
  if count1 <= am1 then
   begin
    repeat
    case box1[count1] of
     $ea : begin
            case openf1 of
             $0 : begin
```

```
                openf1:=1;
               end;
          $3 : begin
                fcs := box1[count1-1];
                write('DTE ':9,addr:5,ns:5,pf:6,nr:5,typ:8,ebd[fcs]:4,chr(13),chr(10));
                openf1:=1;flg1:=1;delay(200);
                {.pa}
               end;
         end;
        end;
    $db : begin
         write('DTE ':9,chr(13),chr(10));flg1:=1;openf1:=0;delay(200);
        end;
    $ed : begin
         write('DTE ':9,'IDLE  STATE ':25,chr(13),chr(10));flg1:=1;openf1:=0;delay(200);
        end;
    $ef : begin
         write('DTE ':9,'FRAME < 32 BITS ':25,chr(13),chr(10));flg1:=1;openf1:=0;delay(200);
        end;
    $ec : begin
         fcs := box1[count1];
         write('DTE ':9,addr:5,ns:5,pf:6,nr:5,typ:8,ebd[fcs]:4,chr(13),chr(10));delay(200);
         openf1:=0;flg1:=1;
        end;
    else  begin
         case openf1 of
          $1 : begin
                addr := box1[count1];openf1:=2;
               end;
          $2 : begin
                classi1(box1[count1]);openf1:=3;
               end;
         end;
        end;
   end;
   count1 := count1+1;
   until (flg1=1) or (count1>am1) or keypressed;
  end
 else
  begin
   openf1:=0;
  end;
end;
procedure twoside2;
 begin
 set_ivt;flg2:=0;
 if count2 <= am2 then
  begin
   repeat
   case box2[count2] of
    $ea : begin
          case openf2 of
           $0 : begin
```

```
                openf2:=1;
               end;
          $3 : begin
                fcs := box2[count2-1];
                write('DCE ':9,addr:5,ns:5,pf:6,nr:5,typ:8,ebd[fcs]:4,chr(13),chr(10));
                openf2:=1;flg2:=1;delay(200);
               end;
               {.pa}
         end;
        end;
    $db : begin
         write('DCE ':9,chr(13),chr(10));flg2:=1;openf2:=0;delay(200);
        end;
    $ed : begin
         write('DCE ':9,'IDLE  STATE ':25,chr(13),chr(10));flg2:=1;openf2:=0;delay(200);
        end;
    $ef : begin
         write('DCE ':9,'FRAME ( 32 BITS ':25,chr(13),chr(10));flg2:=1;openf2:=0;delay(200);
        end;
    $ec : begin
         fcs := box2[count2];
         write('DCE ':9,addr:5,ns:5,pf:6,nr:5,typ:8,ebd[fcs]:4,chr(13),chr(10));delay(200);
         openf2:=0;flg2:=1;
        end;
    else  begin
         case openf2 of
          $1 : begin
                addr := box2[count2];openf2:=2;
               end;
          $2 : begin
                classi2(box2[count2]);openf2:=3;
               end;
         end;
        end;
   end;
   count2 := count2+1;
   until (flg2=1) or (count2>am2) or keypressed;
  end
 else
  begin
   openf2:=0;
  end;
 end;
procedure two12;  {display data : Layer2}
 begin
  repeat
   twoside1;
   twoside2;
  until keypressed;
 end;
procedure layer1;
 begin
 setd;
```

```
  repeat until (ind=1);
  repeat
   onscr;
   decis;
  until upcase(ch) in ['E'];
  end;
 procedure layer2;
  begin
  {.pa}
  setd;
  repeat until (ind=1);
  twoside;
  repeat
   two12;
   decis;
  until upcase (ch) in ['E'];
  end;
 procedure runp;
  begin
   repeat
    read(kbd,alph);
    case alph of
     '1' : begin
           write(alph:3);writeln;writeln;
           writeln('                        WAIT A FEW SEC.');
           layer1;
          end;
     '2' : begin
           write(alph:3);writeln;writeln;
           writeln('                        WAIT A FEW SEC.');
           layer2;
          end;
     '3' : begin
           ch:='E';
          end;
    end;
  until upcase(alph) in ['1','2','3'];
  writeln;
 end;

begin  {mainprog}
 dsave:= Dseg;
 count1:=0;count2:=0;linind:=0;
 ind:=0;ch:='';alph:='';
 enable_irqx;set_ivt;
 frame;disp;select;runp;port[$21]:=imr1;
 move(i88,z804,1);delay(100);move(i88,z802,1);delay(100);
end.
```

```
program SDLC_monitoring;
 type
  ebc = array [$0..$ff] of string[3];
  box = array [0..32767] of byte;
  result = record
            ax,bx,cx,dx,bp,si,di,ds,es,flags : integer;
           end;
 var
  am1,am2,count1,count2,linind,i:integer;
  x,y:string[75];z:string[78];typ:string[4];ch,alph:string[1];
  pf,ns,nr,addr,fcs,openf1,openf2,flg1,flg2,check1,check2,pause:byte;
  box1:box absolute $2000:$0000;
  box2:box absolute $2000:$8000;
  dest1,dest2:file;
  res:result;
 const
  ebd:ebc
  = ('NUL','SOH','STX','ETX','PF','HT','LC','DEL','','',
  'SMM','VT','FF','CR','SO','SI','DLE','DC1','DC2','TM',
  'RES','NL','BS','IL','CAN','EM','CC','CU1','IFS','IGS',
  'IRS','IUS','DS','SOS','FS','','BYP','LF','ETB','ESC',
  '','','SM','CU2','','ENQ','ACK','BEL','','',
  'SYN','','PN','RS','UC','EOT','','','','CU3',
  'DC4','NAK','','SUB','SP','','','','','',
  '','','','','','','','(','(','+',
  '|','&','','','','','','','','',
  '','!','$','*',')',';','^','-','/','',
  '','','','','','','','','','',
  '',',','%','_','>','?','','','','','',
  '','','',':','#','@','''','=','"',
  '','a','b','c','d','e','f','g','h','i',
  '','','','','','','','j','k','l',
  'm','n','o','p','q','r','','','','',
  '','','','','s','t','u','v','w','x',
  'y','z','','','','','','','','',
  '','','','','','','','','','',
  '','','','','','A','B','C','D',
  'E','F','G','H','I','','','','','',
  '','J','K','L','M','N','O','P','Q',
  'R','G','K','','','','','','','S',
  'T','U','V','W','X','Y','Z','F','B','A',
  'I','E','U','0','1','2','3','4','5','6',
  '7','8','9','','','','','','');


procedure disp;
 begin
  gotoxy(13,3);writeln('This software is used for DISPLAY data from DISKETTE');
 end;
procedure frame;
 begin
  clrscr;
  for i:= 12 to 65 do
```

```
  begin
   gotoxy(i,1);write('-');
   gotoxy(i,5);write('-');
  end;
  for i:= 2 to 4 do
  begin
   gotoxy(11,i);write('¦');
   gotoxy(66,i);write('¦');
  end;
  begin
   gotoxy(11,1);write('+');
   gotoxy(66,1);write('+');
   gotoxy(11,5);write('+');
   gotoxy(66,5);write('+');
  end;
 end;
procedure select;
 begin
  gotoxy(15,7);write('Display format');
  gotoxy(15,9);write(' 1) Layer 1 : DTE X...XX');
  gotoxy(27,11);write(': DCE X...XX');
  gotoxy(15,13);write(' 2) Layer 2 : FROM  ADDR.  N(S)  P/F  N(R)  TYPE  FCS');
  gotoxy(15,15);write(' 3) Quit');
  gotoxy(15,17);write('Select [ 1 , 2 or 3 ]');
 end;

procedure ondte;
 begin
  x:='';check1:=0;
  if count1<=am1 then
  begin
  repeat
   x:= x+ebd[box1[count1]];
   if (length(x))72)  then
    begin
     delete(x,(length(x)+1)-length(ebd[box1[count1]]),length(ebd[box1[count1]]));
     check1:=1;
    end
   else
    begin
     count1:=count1+1;
    end;
  until (check1=1) or (count1>am1);
  end
  else
   begin
    x:='';
   end;
 end;
procedure ondce;
 begin
  y:='';check2:=0;
  if count2<=am2 then
```

```
  begin
  repeat
   y:= y+ebd[box2[count2]];
   if (length(y)>72)  then
    begin
     delete(y,(length(y)+1)-length(ebd[box2[count2]]),length(ebd[box2[count2]]));
     check2:=1;
    end
   else
    begin
     count2:=count2+1;
    end;
  until (check2=1) or (count2>am2);
  end
  else
   begin
    y:='';
   end;
 end;
procedure onscr; {display data : Layer1}
 begin
  clrscr;
  repeat
   ondte;
   z:='DTE '+x+chr(13)+chr(10);
   for i := 1 to length(z) do
   begin
    res.ax := ord(copy(z,i,1));
    res.ax := res.ax or $0e00;
    res.bx := 7;
    intr(16,res);
   end;
   ondce;
   z:='DCE '+y+chr(13)+chr(10);
   for i := 1 to length(z) do
   begin
    res.ax := ord(copy(z,i,1));
    res.ax := res.ax or $0e00;
    res.bx := 7;
    intr(16,res);
   end;
   linind := linind+1;
  until  (keypressed) or ((count1>am1) and (count2>am2));
  writeln;write(' linind=  ',linind);
 end;
procedure prnt1; {print data}
 begin
  repeat
   ondte;
   z:='DTE '+x;
   writeln(lst,z);
   ondce;
   z:='DCE '+y;
```

```
   writeln(lst,z);
   linind := linind+1;
  until  keypressed or ((count1>am1) and (count2>am2)) ;
 end;
procedure twoside;
 begin
  clrscr;
  gotoxy(2,1);
  write('  FROM  ADDR.  N(S)  P/F  N(R)  TYPE  FCS ');
  writeln;
 end;
procedure nsend (comm1 : byte);
 begin
  case (comm1 and $0e) of
   $00 : ns := 0;
   $02 : ns := 1;
   $04 : ns := 2;
   $06 : ns := 3;
   $08 : ns := 4;
   $0a : ns := 5;
   $0c : ns := 6;
   $0e : ns := 7;
  end;
 end;
procedure pofi (comm1 : byte);
 begin
  case (comm1 and $10) of
   $00 : pf := 0;
   $10 : pf := 1;
  end;
 end;
procedure nrec (comm1 : byte);
 begin
  case (comm1 and $e0) of
   $00 : nr := 0;
   $20 : nr := 1;
   $40 : nr := 2;
   $60 : nr := 3;
   $80 : nr := 4;
   $a0 : nr := 5;
   $c0 : nr := 6;
   $e0 : nr := 7;
  end;
 end;
procedure untyp1 (comm1 : byte);
 begin
  case (comm1 and $ec) of
   $00 : typ := 'UI';
   $04 : typ := 'SIM';
   $0c : typ := 'DM';
   $20 : typ := 'UP';
   $40 : typ := 'DISC';
   $60 : typ := 'UA';
```

```
        end;
   $03 : begin
          pofi(comm1);untyp2(comm1);ns:=0;nr:=0;
        end;
  end;
 end;
procedure twoside1;
 begin
  flg1:=0;
  if count1 (= am1 then
   begin
    repeat
    case box1[count1] of
     $ea : begin
           case openf1 of
            $0 : begin
                 openf1:=1;
                end;
            $3 : begin
                 fcs := box1[count1-1];
                 write('DTE ':9,addr:5,ns:5,pf:6,nr:5,typ:8,ebd[fcs]:4,chr(13),chr(10));
                 openf1:=1;flg1:=1;delay(200);
                 if fcs = $eb then
                  begin
                   pause:=1;
                  end;
                end;
           end;
          end;
     $db : begin
           write('DTE ':9,chr(13),chr(10));flg1:=1;openf1:=0;delay(200);
          end;
     $ed : begin
           write('DTE ':9,'IDLE  STATE ':25,count1:7,chr(13),chr(10));flg1:=1;openf1:=0;
           delay(200);
          end;
     $ef : begin
           write('DTE ':9,'FRAME ( 32 BITS ':25,chr(13),chr(10));flg1:=1;openf1:=0;
           delay(200);
          end;
     $ec : begin
           fcs:=$ec;
           write('DTE ':9,addr:5,ns:5,pf:6,nr:5,typ:8,ebd[fcs]:4,chr(13),chr(10));
           openf1:=0;flg1:=1;delay(200);pause:=1;
          end;
     else  begin
           case openf1 of
            $1 : begin
                 addr := box1[count1];openf1:=2;
                end;
            $2 : begin
                 classi1(box1[count1]);openf1:=3;
                end;
```

```
          end;
         end;
    end;
    count1 := count1+1;
    until (flg1=1) or (count1>am1) or keypressed;
   end
  else
   begin
    openf1:=0;
   end;
 end;
procedure twoside2;
 begin
 flg2:=0;
 if count2 <= am2 then
  begin
   repeat
   case box2[count2] of
    $ea : begin
          case openf2 of
           $0 : begin
                openf2:=1;
               end;
           $3 : begin
                fcs := box2[count2-1];
                write('DCE ':9,addr:5,ns:5,pf:6,nr:5,typ:8,ebd[fcs]:4,chr(13),chr(10));
                openf2:=1;flg2:=1;delay(200);
                if fcs = $eb then
                  begin
                   pause:=1;
                  end;
               end;
          end;
         end;
    $db : begin
          write('DCE ':9,count2:7,chr(13),chr(10));flg2:=1;openf2:=0;delay(200);
         end;
    $ed : begin
          write('DCE ':9,'IDLE  STATE ':25,count2:7,chr(13),chr(10));flg2:=1;openf2:=0;
          delay(200);
         end;
    $ef : begin
          write('DCE ':9,'FRAME < 32 BITS ':25,chr(13),chr(10));flg2:=1;openf2:=0;
          delay(200);
         end;
    $ec : begin
          fcs:=$ec;
          write('DCE ':9,addr:5,ns:5,pf:6,nr:5,typ:8,ebd[fcs]:4,chr(13),chr(10));
          openf2:=0;flg2:=1;delay(200);pause:=1;
         end;
    else  begin
          case openf2 of
           $1 : begin
```

```
                addr := box2[count2];openf2:=2;
               end;
          $2 : begin
                classi2(box2[count2]);openf2:=3;
               end;
         end;
        end;
   end;
   count2 := count2+1;
   until (flg2=1) or (count2)am2) or keypressed;
  end
 else
  begin
   openf2:=0;
  end;
 end;
procedure lst3;
 begin
  flg1:=0;
  if count1 (= am1 then
   begin
    repeat
    case box1[count1] of
     $ea : begin
           case openf1 of
            $0 : begin
                 openf1:=1;
                end;
            $3 : begin
                 fcs := box1[count1-1];
                 write(lst,'DTE ':9,addr:5,ns:5,pf:6,nr:5,typ:8,ebd[fcs]:4,chr(13),chr(10));
                 openf1:=1;flg1:=1;
                end;
           end;
          end;
     $db : begin
           write(lst,'DTE ':9,chr(13),chr(10));flg1:=1;openf1:=0;
          end;
     $ed : begin
           write(lst,'DTE ':9,'IDLE  STATE ':25,count1:7,chr(13),chr(10));flg1:=1;openf1:=0;
          end;
     $ef : begin
           write(lst,'DTE ':9,'FRAME ( 32 BITS ':25,chr(13),chr(10));flg1:=1;openf1:=0
          end;
     $ec : begin
           fcs:=$ec;
           write(lst,'DTE ':9,addr:5,ns:5,pf:6,nr:5,typ:8,ebd[fcs]:4,chr(13),chr(10));
           openf1:=0;flg1:=1;
          end;
     else  begin
           case openf1 of
            $1 : begin
                 addr := box1[count1];openf1:=2;
```

```
                end;
             $2 : begin
                  classi1(box1[count1]);openf1:=3;
                end;
           end;
          end;
    end;
    count1 := count1+1;
    until (flg1=1) or (count1)am1) or keypressed;
   end
  else
   begin
    openf1:=0;
   end;
 end;
procedure lst4;
 begin
 flg2:=0;
 if count2 (= am2 then
  begin
   repeat
   case box2[count2] of
    $ea : begin
          case openf2 of
           $0 : begin
                 openf2:=1;
                end;
           $3 : begin
                 fcs := box2[count2-1];
                 write(lst,'DCE ':9,addr:5,ns:5,pf:6,nr:5,typ:8,ebd[fcs]:4,chr(13),chr(10));
                 openf2:=1;flg2:=1;
                end;
          end;
         end;
    $db : begin
          write(lst,'DCE ':9,chr(13),chr(10));flg2:=1;openf2:=0;
         end;
    $ed : begin
          write(lst,'DCE ':9,'IDLE  STATE ':25,count2:7,chr(13),chr(10));flg2:=1;openf2:=0
         end;
    $ef : begin
          write(lst,'DCE ':9,'FRAME ( 32 BITS ':25,chr(13),chr(10));flg2:=1;openf2:=0;
         end;
    $ec : begin
          fcs:=$ec;
          write(lst,'DCE ':9,addr:5,ns:5,pf:6,nr:5,typ:8,ebd[fcs]:4,chr(13),chr(10));
          openf2:=0;flg2:=1;
         end;
    else  begin
          case openf2 of
           $1 : begin
                 addr := box2[count2];openf2:=2;
                end;
```

```
          $2 : begin
               classi2(box2[count2]);openf2:=3;
              end;
         end;
        end;
   end;
   count2 := count2+1;
   until (flg2=1) or (count2>am2) or keypressed;
  end
 else
  begin
   openf2:=0;
  end;
 end;
procedure prnt2;   {print data}
 begin
  repeat
   lst3;
   lst4;
  until keypressed or ((count1>am1) and (count2>am2));
 end;
procedure two12;
 begin
  repeat
   twoside1;
   twoside2;
  until keypressed or ((count1>am1) and (count2>am2)) or (pause=1);
 end;
procedure hcopy;
 begin
  count1:=0;count2:=0;linind:=0;
  case alph of
   '1' : begin
          prnt1;
         end;
   '2' : begin
          write(lst,'   FROM  ADDR.  N(S)  P/F  N(R)  TYPE  FCS ');
          writeln(lst);
          prnt2;
         end;
  end;
 end;
procedure decis;
 begin
  writeln;writeln;
  write('P':15);lowvideo;
  write('rinter');normvideo;
  write('C':8);lowvideo;
  write('ontinue');normvideo;
  write('E':8);lowvideo;
  write('nd');normvideo;
  writeln;
  repeat
```

```
   read(kbd,ch);
   case ch of
    'P','p' : begin
               hcopy;
              end;
    'C','c' : begin
              end;
    'E','e' : begin
              end;
   end;
  until upcase(ch) in ['P','C','E'];
 end;
procedure layer1;   {read data from diskette}
 begin
 assign(dest1,'b:DTE.dat');assign(dest2,'b:DCE.dat');
 reset(dest1,1); reset(dest2,1);
 repeat
  blockread(dest1,box1,$7ffd,am1);
  blockread(dest2,box2,$7ffd,am2);
  am1:=am1-1;am2:=am2-1;
  onscr;
  check1:=0;check2:=0;ch:='Z';
  if ((count1<=am1) or (count2<=am2)) or (eof(dest1) and eof(dest2)) then
   begin
    decis;pause:=0;
   end;
  count1:=0;count2:=0;
 until (upcase(ch) in ['E']) {or (eof(dest1) and eof(dest2))};
 close(dest1);close(dest2);
 end;
procedure layer2;  {read data from diskette}
 begin
 assign(dest1,'b:DTE.dat');assign(dest2,'b:DCE.dat');
 reset(dest1,1); reset(dest2,1);
 repeat
  blockread(dest1,box1,$7ffd,am1);
  blockread(dest2,box2,$7ffd,am2);
  am1:=am1-1;am2:=am2-1;
  twoside;
  two12;
  flg1:=0;flg2:=0;ch:='Z';
  if ((count1<=am1) or (count2<=am2)) or (eof(dest1) and eof(dest2) or (pause=1)) then
   begin
    decis;pause:=0
   end;
  count1:=0;count2:=0;
 until (upcase(ch) in ['E']) {or (eof(dest1) and eof(dest2))};
 close(dest1);close(dest2);
 end;
procedure runp;
 begin
  repeat
   read(kbd,alph);
```

```
   case alph of
    '1' : begin
          write(alph:3);writeln;writeln;
          layer1;
         end;
    '2' : begin
          write(alph:3);writeln;writeln;
          layer2;
         end;
    '3' : begin
          ch:='E';
         end;
   end;
  until upcase(alph) in ['1','2','3'];
  writeln;
 end;

begin  {mainprog}
 count1:=0;count2:=0;linind:=0;
 openf1:=0;openf2:=0;pause:=0;
 ch:='';alph:='';
 frame;disp;select;runp;
end.
```

```
                ORG     0H
                DI
                LD      A,00H
                OUT     (0BH),A
                LD      HL,0A000H
                LD      BC,2000H
                LD      DE,1H
AAZZ:           LD      (BC),A
                INC     BC
                SBC     HL,DE
                JP      NZ,AAZZ
                JP      100H

                ORG     38H
                JP      CHECK1
                ORG     66H
                JP      NON1

                ORG     100H
INTID           EQU     0BEFFH                  ;SAVE STATUS OF INT.
MAINB           EQU     0BEFDH                  ;BYTE CNT. OF MAIN PROG.
CNTB            EQU     0BEFBH                  ;BYTE CNT. OF MOVE BLOCK
DESD            EQU     0BEF9H                  ;DEST. BYTE OF MOVE BLOCK
SORH            EQU     0BEF7H                  ;SOURCE BYTE OF MOVE BLOCK
MEMH            EQU     0BEF5H                  ;MEMBF.CNT.INT.

;************* INITIALIZE 8253 & 8273 *************

START:          LD      A,00H                   ;RSET.INT.STATUS
                LD      (INTID),A
                OUT     (0CH),A                 ;RSET. COMMON MEM.
                OUT     (0DH),A
                LD      A,7AH
                OUT     (07H),A
                LD      A,0EH
                OUT     (05H),A
                LD      A,00H
                OUT     (05H),A
                LD      HL,9FFEH        ;SET NUMERATOR
                LD      (MEMH),HL
                LD      BC,2000H        ;SET MAIN ADDR.BYTE CNT.
                LD      (MAINB),BC
                LD      SP,0BFFFH       ;SET STACK POINTER
CM3:            CALL    SPG1                    ;WR.CMD. OF DAT.TRANSF. MODE
                LD      A,97H
                OUT     (00H),A
                CALL    SPG2                    ;WR.PAR.
                LD      A,01H
                OUT     (01H),A
CM4:            CALL    SPG1                    ;WR.CMD. OF OPER.MODE
                LD      A,91H
                OUT     (00H),A
                CALL    SPG2                    ;WR.PAR.
```

```
                LD          A,00H
                OUT         (01H),A
CM5:            CALL        SPG1                    ;WR.CMD. OF SERIAL I/O MODE
                LD          A,0A0H
                OUT         (00H),A
                CALL        SPG2                    ;WR.PAR.
                LD          A,01H
                OUT         (01H),A
                CALL        CM7

;************** MAIN PROGRAM *****************

                IM          1
                EI
MAIN:           IN          A,(10H)                 ;*****************
                BIT         1,A
                JP          Z,MAIN1
                LD          A,(INTID)
                BIT         7,A
                JP          NZ,BRAN2
                LD          BC,(MAINB)
                LD          (9FFEH),BC
                LD          BC,0BEF0H
                LD          (MEMH),BC
                LD          BC,0A000H
                LD          (MAINB),BC
                LD          A,(INTID)
                SET         7,A
                LD          (INTID),A
BRAN2:          OUT         (08H),A
BRAN5:          LD          A,(INTID)
                BIT         7,A
                JP          NZ,BRAN5
MAIN1:          IN          A,(11H)                 ;CHK.ACK. FROM REMOTEZ-80
                BIT         1,A
                JP          Z,LABL1                 ;NO ACK.
                LD          A,(INTID)    ;ACK.
                RES         0,A                     ;DEL.INF. OF LOCALZ-80
                LD          (INTID),A
                OUT         (0DH),A
LABL1:          IN          A,(11H)                 ;CHK.INF. FROM REMOTEZ-80
                BIT         0,A
                JP          Z,LABL2                 ;NO INF.
                LD          A,(INTID)    ;INF.
                BIT         1,A                     ;CHK.REACK. FROM LOCALZ-80
                JP          NZ,LABL3     ;REACK. ALREADY
                SET         1,A                     ;NOTYET
                LD          (INTID),A
                OUT         (0DH),A
                BIT         2,A                     ;CHK. RECEIVING DATA?
                JP          NZ,LABL3     ;YES
                LD          A,0DBH                  ;NO ==) INSERT BLANK
                LD          BC,(MAINB)
```

```
            LD          (BC),A
            INC         BC
            LD          (MAINB),BC
            JP          LABL3
LABL2:      LD          A,(INTID)     ;CHK. REALLY NOINF.?
            BIT         1,A
            JP          Z,LABL3                 ;YES
            RES         1,A                     ;NO
            LD          (INTID),A
            OUT         (0DH),A
LABL3:      LD          A,(INTID)
            BIT         6,A                     ;CHECK RET. FROM 2nd NMI.
            JP          Z,MAIN
            LD          BC,(CNTB)     ;INITILIZE MOVE BLOCK
            LD          DE,(DESD)
            LD          HL,(SORH)
            LDIR
            LD          A,(INTID)     ;RSET.INT. STATUS
            RES         6,A
            LD          (INTID),A
            LD          BC,(MAINB)
            JP          MAIN

;************ MASKABLE INT. SERVICE ROUTINE **************

CHECK1:     PUSH        AF
            PUSH        HL                      ;SAVE 0.STATUS
            PUSH        DE
            PUSH        BC
            LD          BC,(MAINB)    ;LOAD MAIN BYTE CNT.
            IN          A,(00H)                 ;RD. STATUS REG.
            BIT         1,A                     ;TEST RxIRA
            JP          NZ,CHECK2     ;RxIRA<>0
            LD          A,22H                   ;CHECK FLAG
            OUT         (00H),A
CRBF2:      IN          A,(00H)
            BIT         4,A
            JP          Z,CRBF2
            IN          A,(01H)
            BIT         2,A
            JP          Z,CRBF3
            CALL        SPG1
            LD          A,63H
            OUT         (00H),A
            CALL        SPG2
            LD          A,00H
            OUT         (01H),A
            LD          A,0EAH                  ;LOAD EBCDIC 'FLAG'
            OUT         (0BH),A                 ;RELEASE INT.SIG.
            JP          CENT
CRBF3:      BIT         3,A
            JP          Z,DATAI
CRBF5:      LD          A,22H                   ;CHECK FLAG
```

```
        OUT     (00H),A
CRBF4:  IN      A,(00H)
        BIT     4,A
        JP      Z,CRBF4
        IN      A,(01H)
        BIT     3,A
        JP      NZ,CRBF5
        LD      A,0EDH
        JP      CENT
DATAI:  LD      A,(INTID)
        SET     2,A
        LD      (INTID),A
        IN      A,(0AH)                 ;RD. DATA
CENT:   LD      (BC),A                  ;MOVE DATA INTO MEM.
        INC     BC
        LD      (MAINB),BC
        LD      A,C
        LD      HL,0BEF5H
        CP      (HL)
        JP      NZ,CENT1
        LD      A,B
        INC     HL
        CP      (HL)
        JP      NZ,CENT1
        LD      (9FFEH),BC      ;INDICATE BYTE_CNT. TO 8088
        LD      BC,0BEF0H       ;SET NUMERATOR OF RES_MEM
        LD      (MEMH),BC
        LD      BC,0A000H       ;LOAD NEW BYTE_CNT.
        LD      (MAINB),BC
        LD      A,(INTID)       ;****************
        SET     7,A
        LD      (INTID),A
        LD      A,01H                   ;IND.BUFF. TO ANOTHER Z-80
        OUT     (0CH),A
CENT1:  POP     BC                      ;RET. 0.STATUS
        POP     DE
        POP     HL
        POP     AF
        EI
        RETI
CHECK2: IN      A,(00H)
        BIT     3,A                     ;TEST RxINT
        JP      NZ,CHKINT       ;RxINT<>0
        JP      CENT1
CHKINT: IN      A,(03H)                 ;RD. RxINT RESULT
        AND     1FH
        ADD     A,A                     ;FIND INT.ADDR.
        LD      HL,TABLE
        ADD     A,L
        LD      L,A
        LD      A,H
        ADC     A,00H
        LD      H,A
```

```
                LD          E,(HL)
                INC         HL
                LD          D,(HL)
                EX          DE,HL
                JP          (HL)                ;JP. TO SPECIAL COND.

;************ SPECIAL CONDITION **************

A1M:            CALL        SPG1
                LD          A,63H
                OUT         (00H),A
                CALL        SPG2
                LD          A,00H
                OUT         (01H),A
                LD          A,(INTID)
                RES         2,A
                SET         0,A
                LD          (INTID),A
                OUT         (0DH),A
                DEC         BC
                LD          A,(BC)
                LD          D,A
                LD          A,0DAH              ;LOAD EBCDIC 'GOOD'
        LD      (BC),A
                INC         BC
                LD          A,D
                JP          CENT
A2M:            JP          CENT1
A3M:            JP          CENT1
CRCER:          CALL        SPG1
                LD          A,63H
                OUT         (00H),A
                CALL        SPG2
                LD          A,00H
                OUT         (01H),A
                LD          A,(INTID)
                RES         2,A
                SET         0,A
                LD          (INTID),A
                OUT         (0DH),A
                LD          A,0EBH              ;'BAD'
                JP          CENT
ABORT:          CALL        SPG1
                LD          A,63H
                OUT         (00H),A
                CALL        SPG2
                LD          A,00H
                OUT         (01H),A
                LD          A,0ECH              ;'ABORT'
                JP          CENT
IDLE:           CALL        CM7
                CALL        SPG1
                LD          A,0A3H
```

```
                OUT         (00H),A
                CALL        SPG2
                LD          A,02H
                OUT         (01H),A
                LD          A,0EDH                  ;'IDLE'
                JP          CENT
EOP:            CALL        CM7
                CALL        SPG1
                LD          A,63H
                OUT         (00H),A
                CALL        SPG2
                LD          A,00H
                OUT         (01H),A
        LD      A,0EEH                  ;'EOP'
                JP          CENT
FRMER:          CALL        SPG1
                LD          A,63H
                OUT         (00H),A
                CALL        SPG2
                LD          A,00H
                OUT         (01H),A
                LD          A,(INTID)
                RES         2,A
                SET         0,A
                LD          (INTID),A
                OUT         (0DH),A
                LD          A,0EFH                  ;'UNSATISFY'
                JP          CENT
DMAOV:          CALL        CM7
                JP          CENT1
MEMBF:          CALL        CM7
                JP          CENT1
CDF:            CALL        CM7
                JP          CENT1
RIOV:           CALL        CM7
                JP          CENT1

;************ NONMASKABLE INT. SERVICE ROUTINE ************

NON1:           PUSH        AF
                PUSH        HL                      ;SAVE 0.STATUS
                PUSH        DE
                PUSH        BC
                LD          A,00H                   ;RSET COMMON  MEM.
                OUT         (0CH),A
                LD          A,(INTID)   ;CHECK ORDER OF NMI.
                BIT         5,A
                JP          Z,NON2                  ;1st NMI.
                RES         5,A                     ;2nd NMI.
                LD          BC,(MAINB)
                LD          H,B
                LD          L,C
                LD          DE,0A000H
```

```
                SBC         HL,DE
                JP          Z,NOTRA
                SET         6,A
NOTRA:          LD          (INTID),A
                LD          (CNTB),HL
                LD          (SORH),DE
                LD          B,H
                LD          C,L
                LD          HL,2000H
                LD          (DESD),HL
                ADD         HL,BC
                LD          B,H
                LD          C,L
                LD          HL,9FFEH        ;SET NUMERATOR OF MAIN_MEM
                LD          (MEMH),HL
                LD          (MAINB),BC
                POP         BC
                POP         DE
                POP         HL
                POP         AF
                RETN
NON2:           RES         7,A
                SET         5,A
                LD          (INTID),A
                POP         BC
                POP         DE
                POP         HL
                POP         AF
                RETN

;************ SUBPROGRAM *************

SPG1:           IN          A,(00H)             ;RD. STATUS REG.
                BIT         7,A                 ;TEST CBSY
                JP          NZ,SPG1             ;CBSY()0
                RET
SPG2:           IN          A,(00H)
                BIT         5,A                 ;TEST CPBF
                JP          NZ,SPG2             ;CPBF()0
                RET
CM7:            CALL        SPG1                ;WR.GEN.REC.CMD.
                LD          A,0C0H
                OUT         (00H),A
                CALL        SPG2                ;SET MEM.BUFF.
                LD          A,0FFH
                OUT         (01H),A
                CALL        SPG2
                LD          A,0FFH
                OUT         (01H),A
                RET

;*********** MAP TABLE *************
```

```
TABLE       DW          A1M
            DW          A2M
            DW          A3M
            DW          CRCER
            DW          ABORT
            DW          IDLE
            DW          EOP
            DW          FRMER
            DW          DMAOV
            DW          MEMBF
            DW          CDF
            DW          RIOV
            END
```

```
            ORG         0H
            DI
            LD          A,00H
            OUT         (0BH),A
            LD          HL,0A000H
            LD          BC,2000H
            LD          DE,1H
AAZZ:       LD          (BC),A
            INC         BC
            SBC         HL,DE
            JP          NZ,AAZZ
            JP          100H

            ORG         38H
            JP          CHECK1
            ORG         66H
            JP          NON1


            ORG         100H
INTID       EQU         0BEFFH                  ;SAVE STATUS OF INT.
MAINB       EQU         0BEFDH                  ;BYTE CNT. OF MAIN PROG.
CNTB        EQU         0BEF8H                  ;BYTE CNT. OF MOVE BLOCK
DESD        EQU         0BEF9H                  ;DEST. BYTE OF MOVE BLOCK
SORH        EQU         0BEF7H                  ;SOURCE BYTE OF MOVE BLOCK
MEMH        EQU         0BEF5H                  ;MEMBF.CNT.INT.

;************* INITIALIZE 8253 & 8273 **************

START:      LD          A,00H                   ;RSET.INT.STATUS
            LD          (INTID),A
            OUT         (0CH),A                 ;RSET. COMMON MEM.
            OUT         (0DH),A
            LD          A,7AH
            OUT         (07H),A
            LD          A,0EH
            OUT         (05H),A
            LD          A,00H
            OUT         (05H),A
            LD          HL,9FFEH        ;SET NUMERATOR
            LD          (MEMH),HL
            LD          BC,2000H        ;SET MAIN ADDR.BYTE CNT.
            LD          (MAINB),BC
            LD          SP,0BFFFH       ;SET STACK POINTER
CM3:        CALL        SPG1                    ;WR.CMD. OF DAT.TRANSF. MODE
            LD          A,97H
            OUT         (00H),A
            CALL        SPG2                    ;WR.PAR.
            LD          A,01H
            OUT         (01H),A
CM4:        CALL        SPG1                    ;WR.CMD. OF OPER.MODE
            LD          A,91H
            OUT         (00H),A
```

```
            CALL        SPG2                    ;WR.PAR.
            LD          A,00H
            OUT         (01H),A
CM5:        CALL        SPG1                    ;WR.CMD. OF SERIAL I/O MODE
            LD          A,0A0H
            OUT         (00H),A
            CALL        SPG2                    ;WR.PAR.
            LD          A,01H
            OUT         (01H),A
            CALL        CM7

;************** MAIN PROGRAM *****************

            IM          1
            EI
MAIN:       IN          A,(10H)                 ;*****************
            BIT         0,A
            JP          Z,MAIN1
            LD          A,(INTID)
            BIT         7,A
            JP          NZ,MAIN1
            LD          BC,(MAINB)
            LD          (9FFEH),BC  ;INDICATE BYTE_CNT. TO 8088
            LD          BC,0BEF0H   ;SET NUMERATOR OF RES_MEM
            LD          (MEMH),BC
            LD          BC,0A000H   ;LOAD NEW BYTE_CNT.
            LD          (MAINB),BC
            LD          A,(INTID)
            SET         7,A
            LD          (INTID),A
            LD          A,02H                   ;IND.BUFF. TO ANOTHER Z_80
            OUT         (0CH),A
MAIN1:      IN          A,(11H)                 ;CHK.ACK.FREMOTEZ_80
            BIT         1,A
            JP          Z,LABL1                 ;NO ACK.
            LD          A,(INTID)   ;ACK.
            RES         0,A                     ;DEL.INF.OLOCALZ_80
            LD          (INTID),A
            OUT         (0DH),A
LABL1:      IN          A,(11H)                 ;CHK.INF.FREMOTEZ_80
            BIT         0,A
            JP          Z,LABL2                 ;NO INF.
            LD          A,(INTID)   ;INF.
            BIT         1,A                     ;CHK.REACK.FLOCALZ_80
            JP          NZ,LABL3    ;REACK.AL.
            SET         1,A                     ;NOTYET
            LD          (INTID),A
            OUT         (0DH),A
            BIT         2,A                     ;CHK.RECINGDATA ?
            JP          NZ,LABL3    ;YES
            LD          A,0DBH                  ;NO ==> INS.BK.
            LD          BC,(MAINB)
            LD          (BC),A
```

```
            INC         BC
            LD          (MAINB),BC
            JP          LABL3
LABL2:      LD          A,(INTID)     ;CHK.REALLY NOINF.?
            BIT         1,A
            JP          Z,LABL3                 ;YES
            RES         1,A                     ;NO
            LD          (INTID),A
            OUT         (0DH),A
LABL3:      LD          A,(INTID)
            BIT         6,A                     ;CHECK RET. FROM 2nd NMI.
            JP          Z,MAIN
            LD          BC,(CNTB)     ;INITILIZE MOVE BLOCK
            LD          DE,(DESD)
            LD          HL,(SORH)
            LDIR
            LD          A,(INTID)     ;RSET.INT. STATUS
            RES         6,A
            LD          (INTID),A
            LD          BC,(MAINB)
            JP          MAIN

;************ MASKABLE INT. SERVICE ROUTINE *************

CHECK1:     PUSH        AF
            PUSH        HL                      ;SAVE O.STATUS
            PUSH        DE
            PUSH        BC
            LD          BC,(MAINB)    ;LOAD MAIN BYTE CNT.
            IN          A,(00H)                 ;RD. STATUS REG.
            BIT         1,A                     ;TEST RxIRA
            JP          NZ,CHECK2     ;RxIRA<>0
            LD          A,22H                   ;CHECK FLAG
            OUT         (00H),A
CRBF2:      IN          A,(00H)
            BIT         4,A
            JP          Z,CRBF2
            IN          A,(01H)
            BIT         2,A
            JP          Z,CRBF3
            CALL        SPG1
            LD          A,63H
            OUT         (00H),A
            CALL        SPG2
            LD          A,00H
            OUT         (01H),A
            LD          A,0EAH                  ;LOAD EBCDIC 'FLAG'
            OUT         (0BH),A                 ;RELEASE INT.SIG.
            JP          CENT
CRBF3:      BIT         3,A
            JP          Z,DATAI
CRBF5:      LD          A,22H                   ;CHECK IDLE
            OUT         (00H),A
```

```
CRBF4:  IN      A,(00H)
        BIT     4,A
        JP      Z,CRBF4
        IN      A,(01H)
        BIT     3,A
        JP      NZ,CRBF5
        LD      A,0EDH
        JP      CENT
DATAI:  LD      A,(INTID)
        SET     2,A
        LD      (INTID),A
        IN      A,(0AH)                 ;RD. DATA
CENT:   LD      (BC),A                  ;MOVE DATA INTO MEM.
        INC     BC
        LD      (MAINB),BC
        LD      A,C
        LD      HL,0BEF5H
        CP      (HL)
        JP      NZ,CENT1
        LD      A,B
        INC     HL
        CP      (HL)
        JP      NZ,CENT1
        LD      (9FFEH),BC      ;INDICATE BYTE_CNT. TO 8088
        LD      BC,0BEF0H       ;SET NUMERATOR OF RES_MEM
        LD      (MEMH),BC
        LD      BC,0A000H       ;LOAD NEW BYTE_CNT.
        LD      (MAINB),BC
        LD      A,(INTID)
        SET     7,A
        LD      (INTID),A
        LD      A,02H                   ;IND.BUFF. TO ANOTHER Z-80
        OUT     (0CH),A
CENT1:  POP     BC                      ;RET. 0.STATUS
        POP     DE
        POP     HL
        POP     AF
        EI
        RETI
CHECK2: IN      A,(00H)
        BIT     3,A                     ;TEST RxINT
        JP      NZ,CHKINT       ;RxINT()0
        JP      CENT1
CHKINT: IN      A,(03H)                 ;RD. RxINT RESULT
        AND     1FH
        ADD     A,A                     ;FIND INT.ADDR.
        LD      HL,TABLE
        ADD     A,L
        LD      L,A
        LD      A,H
        ADC     A,00H
        LD      H,A
        LD      E,(HL)
```

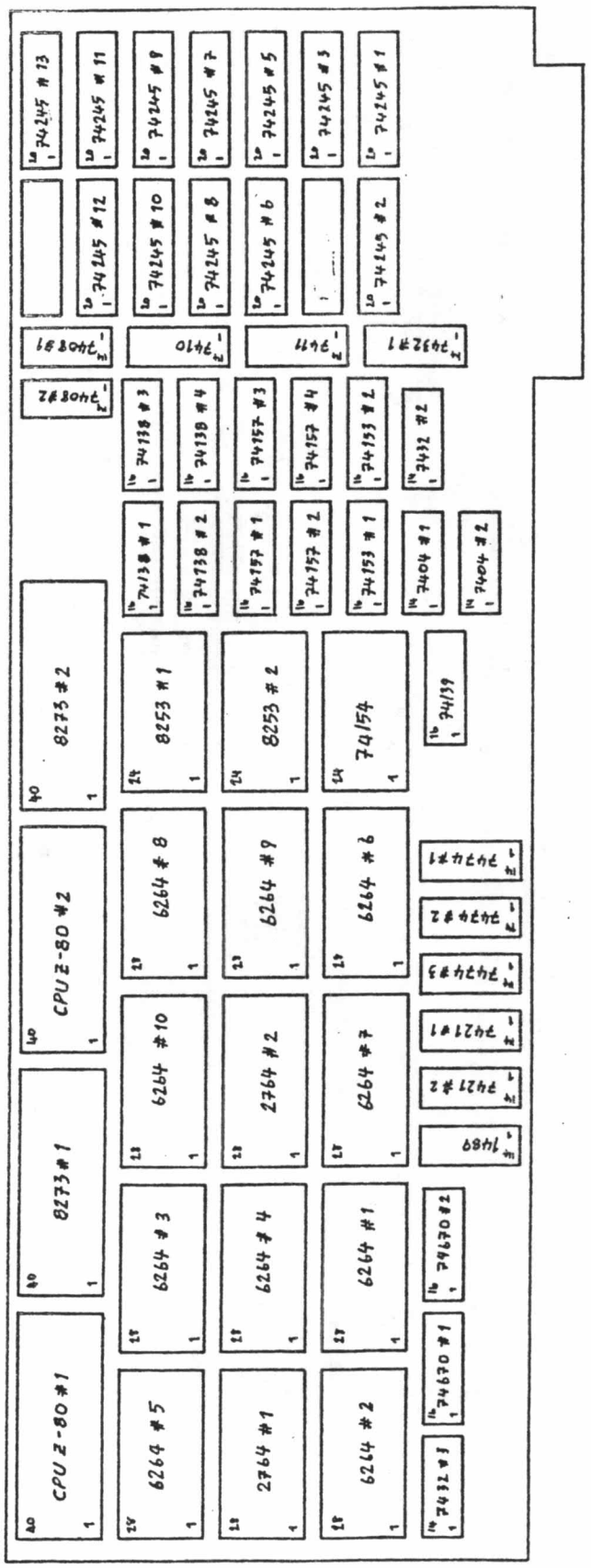
```
                INC         HL
                LD          D,(HL)
                EX          DE,HL
                JP          (HL)                ;JP. TO SPECIAL COND.

;************ SPECIAL CONDITION **************

A1M:            CALL        SPG1
                LD          A,63H
                OUT         (00H),A
                CALL        SPG2
                LD          A,00H
                OUT         (01H),A
                LD          A,(INTID)
                RES         2,A
                SET         0,A
                LD          (INTID),A
                OUT         (0DH),A
                DEC         BC
                LD          A,(BC)
                LD          D,A
                LD          A,0DAH              ;LOAD EBCDIC 'GOOD'
        LD      (BC),A
                INC         BC
                LD          A,D
                JP          CENT
A2M:            JP          CENT1
A3M:            JP          CENT1
CRCER:          CALL        SPG1
                LD          A,63H
                OUT         (00H),A
                CALL        SPG2
                LD          A,00H
                OUT         (01H),A
                LD          A,(INTID)
                RES         2,A
                SET         0,A
                LD          (INTID),A
                OUT         (0DH),A
                LD          A,0EBH              ;'BAD'
                JP          CENT
ABORT:          CALL        SPG1
                LD          A,63H
                OUT         (00H),A
                CALL        SPG2
                LD          A,00H
                OUT         (01H),A
                LD          A,0ECH              ;'ABORT'
                JP          CENT
IDLE:           CALL        CM7
                CALL        SPG1
                LD          A,0A3H
                OUT         (00H),A
```

```
                CALL    SPG2
                LD      A,02H
                OUT     (01H),A
                LD      A,0EDH                  ;'IDLE'
                JP      CENT
EOP:            CALL    CM7
                CALL    SPG1
                LD      A,063H
                OUT     (00H),A
                CALL    SPG2
                LD      A,00H
                OUT     (01H),A
        LD      A,0EEH                  ;'EOP'
                JP      CENT
FRMER:          CALL    SPG1
                LD      A,063H
                OUT     (00H),A
                CALL    SPG2
                LD      A,00H
                OUT     (01H),A
                LD      A,(INTID)
                RES     2,A
                SET     0,A
                LD      (INTID),A
                OUT     (0DH),A
                LD      A,0EFH                  ;'UNSATISFY'
                JP      CENT
DMAOV:          CALL    CM7
                JP      CENT1
MEMBF:          CALL    CM7
                JP      CENT1
CDF:            CALL    CM7
                JP      CENT1
RIOV:           CALL    CM7
                JP      CENT1

;************* NONMASKABLE INT. SERVICE ROUTINE *************

NON1:           PUSH    AF
                PUSH    HL                      ;SAVE 0.STATUS
                PUSH    DE
                PUSH    BC
                LD      A,00H                   ;RSET COMMON  MEM.
                OUT     (0CH),A
                LD      A,(INTID)       ;CHECK ORDER OF NMI.
                BIT     5,A
                JP      Z,NON2                  ;1st NMI.
                RES     5,A                     ;2nd NMI.
                RES     7,A
                LD      BC,(MAINB)
                LD      H,B
                LD      L,C
                LD      DE,0A000H
```

```
                SBC         HL,DE
                JP          Z,NOTRA
                SET         6,A
NOTRA:          LD          (INTID),A
                LD          (CNTB),HL
                LD          (SORH),DE
                LD          B,H
                LD          C,L
                LD          HL,2000H
                LD          (DESD),HL
                ADD         HL,BC
                LD          B,H
                LD          C,L
                LD          HL,9FFEH        ;SET NUMERATOR OF MAIN_MEM
                LD          (MEMH),HL
                LD          (MAINB),BC
                POP         BC
                POP         DE
                POP         HL
                POP         AF
                RETN
NON2:           SET         5,A
                LD          (INTID),A
                POP         BC
                POP         DE
                POP         HL
                POP         AF
                RETN

;************ SUBPROGRAM *************

SPG1:           IN          A,(00H)                 ;RD. STATUS REG.
                BIT         7,A                     ;TEST CBSY
                JP          NZ,SPG1                 ;CBSY()0
                RET
SPG2:           IN          A,(00H)
                BIT         5,A                     ;TEST CPBF
                JP          NZ,SPG2                 ;CPBF()0
                RET
CM7:            CALL        SPG1                    ;WR.GEN.REC.CMD.
                LD          A,0C0H
                OUT         (00H),A
                CALL        SPG2                    ;SET MEM.BUFF.
                LD          A,0FFH
                OUT         (01H),A
                CALL        SPG2
                LD          A,0FFH
                OUT         (01H),A
                RET



;********** MAP TABLE *************
```

```
TABLE       DW          A1M
            DW          A2M
            DW          A3M
            DW          CRCER
            DW          ABORT
            DW          IDLE
            DW          EOP
            DW          FRMER
            DW          DMAOV
            DW          MEMBF
            DW          CDF
            DW          RIOV
            END
```

ภาคผนวก ง



แผนผังแสดงตำแหน่งของไอซีเบอร์ต่าง ๆ

ประวัติผู้เขียน

นาย เขมะทัต วิภาตะวนิช เกิดเมื่อวันที่ 19 พฤศจิกายน พ.ศ. 2505 ที่กรุงเทพ สำเร็จปริญญาตรีวิศวกรรมศาสตร์บัณฑิต สาขาวิศวกรรมไฟฟ้า จากมหาวิทยาลัยเกษตรศาสตร์ เมื่อ พ.ศ. 2526 เคยทำงานในตำแหน่งวิศวกรอันดับหนึ่งแผนกปรับปรุงและพัฒนา กองวางแผนและวิศวกรรมระบบสื่อสาร ฝ่ายระบบสื่อสาร การไฟฟ้าฝ่ายผลิตแห่งประเทศไทย เข้าศึกษาต่อในระดับปริญญาโทที่จุฬาลงกรณ์มหาวิทยาลัย เมื่อ พ.ศ. 2528