ปริญญานิพนธ์ปีการศึกษา 2536

ภาควิชา อิเล็คทรอนิกส์

คณะ วิศวกรรมศาสตร์ สถาบันเทคโนโลยี พระจอมเกล้าเจ้าคุณทหารลาดกระบัง

เรื่อง การส่งข้อมูลผ่านทางอินฟราเรด (DATA TRANSMITTING BY INFRARED)

ผู้จัดทำ

นายพิเศษ บุญเลี้ยง รหัส 33100259

นายภุชงค์ พันมานิมิตร รหัส 33100288

นายมนตรี วรรณา รหัส 33100292

อาจารย์ที่ปรึกษา

(ดิเรก ธีรกุล)

033359

สารบัญ

หน้า

บทที่ 1 บทนำ

- Abstract 1
- บทคัดย่อ 2
- การสื่อสารข้อมูลผ่านพอร์ตอนุกรม 3
- มาตรฐาน RS-232 17

บทที่ 2 การประยุกต์ใช้ในการสั่งอาหาร

- การทำงานภาคส่ง 23
- การทำงานภาครับ 27
- ภาคส่งรับอินฟราเรด 28
- โปรแกรมภาครับ 33
- โปรแกรมภาคส่ง 41
- รูปแบบวงจร 62

บทที่ 3 การสร้างและการทดสอบ

- การทดลองและการทดสอบ 69
- รายการอุปกรณ์ 71
- ปัญหาที่พบและการนำไปปรับปรุงพัฒนา 72

บทสรุป 73

บรรณานุกรม

บทที่ 1 บทนำ

- Abstract

- บทคัดย่อ

- การสื่อสารข้อมูลผ่านพอร์ตอนุกรม

- มาตรฐาน RS-232

ABSTRACT

This project shows the method of transmitting data between two CPUs ie. 8031 and i286. The transmitter is 8031 CPU and reciever is i286 (or higher). 8031 CPU used for recieve data from user and keep in its memory then send to i286 CPU that used for decode data and shows output to user through monitor. The way for transfer data can use by infrared or transmission lines. The applications of this project we can use in the restaurant for sending menu order or use to check stock in supermarket.

บทคัดย่อ

ในปัจจุบันการสื่อสารและการเคลื่อนย้ายข้อมูลมีความสำคัญมากและมีการพัฒนาขึ้นเรื่อยๆ โปรเจคนี้เป็นการเคลื่อนย้ายข้อมูลระหว่าง CPU 2 ตัวคือ 8031 CPU จะทำหน้าที่ส่งข้อมูลออกทางพอร์ตอนุกรมไปให้ PC ซึ่งมี CPU เบอร์ 80286 ขึ้นไป โดยผ่านทางอินฟราเรด เราสามารถประยุกต์ใช้ในการสั่งอาหารตามร้านอาหารได้ โดยจะมี COMPUTER รอรับรายการอาหารอยู่ในห้องครัว หรือเราจะประยุกต์ใช้ในการ CHECK จำนวนสินค้าตาม SUPERMARKET เพื่อนำไปหัก STOCK โดยเราจะป้อนจำนวนสินค้าเข้าไปในตัวส่งของเรา ซึ่งมีลักษณะเป็น portable สามารถถือไปมาได้ พอป้อนเรียบร้อยแล้วก็นำไปต่อเข้ากับ computer ซึ่ง run program รับข้อมูลแล้วนำไปหักกับจำนวนล่าสุด หรือจะใช้อินฟราเรดในการส่งก็ได้ ผู้ที่สนใจจะนำไปทำเป็นโปรเจค ขอแนะนำให้ปรับปรุงภาคส่งอินฟราเรดให้ดีขึ้น

SERIAL DATA COMMUNICATION

ในการส่งของมูลจะใช้ PROGRAM MODE 1: STANDARD 8 - BIT UART MODE รูปแบบการส่งจะส่งข้อมูล 8 บิท เป็น ASCII CODE 7 BIT และ PARITY 1 BIT หรือจะละ PARITY BIT ทิ้งก็ได้ก็จะกลายเป็นการส่งข้อมูล 8 บิท ในการส่งจะเป็นแบบ ASYNCHORNOUS MODE DATA TRANSMISSION ซึ่งต้องใช้ START BIT และ STOP BIT เพื่อให้วงจรทางด้านรับ DETECT BIT เริ่มต้นและบิทสุดท้ายได้ เพราะฉะนั้นข้อมูลที่ต้องส่งไปจริงทั้งหมดมี 10 บิท ประกอบด้วย ASCII CODE 7 BIT PARITY 1 BIT START และ STOP BIT อีก 2 BIT

การส่งจะเริ่มเมื่อมี DATA ถูกเขียนเข้าไปยัง SBUF และการส่ง BYTE ต่อไปจะต้องแน่ใจว่า ข้อมูลชุดก่อนหน้าถูกส่งไปเรียบร้อยแล้ว ซึ่งเราสามารถทราบได้โดยการตรวจดู บิท T_i ใน SFR SCON ว่าอยู่ในสถานะ HIGH หรือไม่ การเคลื่อนย้ายข้อมูลจะเริ่มจาก START BIT ซึ่งจะทำให้เกิดการเปลี่ยนแปลง ของสถานะของขา TXD ซึ่งจะเปลี่ยนจาก HIGH เป็น LOW จะเป็นสัญญาณบอกให้ CIRCUIT ทางด้านรับทรายว่าจะมีข้อมูลอีก 8 บิท และตามด้วย STOP BIT อีก 1 บิท โดยข้อมูลบิทต่ำ (SLB) จะถูกส่งไปก่อนสุดท้ายจะเป็น (MSB) และ STOP BIT จะเป็นบิทสุดท้ายที่จะส่งไป และเป็นสัญญาณบอกให้ทางด้านรับทราบว่าสิ้นสุดการส่งข้อมูล 1 บิท และถ้ามีข้อมูล BYTE ต่อไปตามมาทันทีทันใด START BIT ของข้อมูลชุดหลังนี้จะถูกส่งออกไปทางขา TXD หลังจาก STOP BIT ของข้อมูล BYTE ก่อนหน้าถูกส่งไปแล้ว 1 PERIOD ของ STOP BIT ของข้อมูลชุดก่อน แต่ในทางตรงข้ามถ้ายังไม่มีข้อมูลส่งตามมาขา TXD จะคงสถานะ HIGH ไว้ เรียกสถานะนี้ว่า IDLE

ความกว้างของ PULSE ที่ส่งไปจะถูกควบคุมโดย BAUD RATE CLOCK ที่ใช้ ซึ่ง BAUD RATE ของตัวรับจะต้องตรงกับ BAUD RATE ทางด้านส่ง ถ้าไม่ตรงกันจะทำให้การรับข้อมูลผิดไป

SFR ที่เกี่ยวข้องกับการส่งข้อมูลออกทาง SERIAL PORT

1).SCON (Serial Port Control Register) ตำแหน่งหน่วยความจำภายใน 98H รีจีสเตอร์ SCON มีขนาด 8 บิท ใช้สำหรับควบคุมการส่งและการรับข้อมูลผ่านทาง Serial Port แต่ละบิทของข้อมูลใน Register นี้มีความหมายเฉพาะดังแสดงในรูป

SCON: SERIAL PORT CONTROL REGISTER BIT ADDRESSABLE.

SM0	SM1	SM2	REN	TB8	RB8	TI	RI

RI Receive Interrupt Flag

bit นี้จะถูกกำหนดโดย Hardware ให้มีค่าเป็น 0 หรือ 1 โดยที่ในการรับข้อมูลโหมด 0 นั้น บิท RB8 จะมีค่าเป็น 1 เมื่อมีข้อมูลเข้ามาครบทั้ง 8 บิท ส่วนใน MODE อื่น BIT RB8 จะเป็น 1 ก็ต่อเมื่อข้อมูลเข้ามาถึงเวลาครึ่งหนึ่งของ STOP BIT (ยกเว้นบางกรณีจะกล่าวไว้ใน SM2) บิทนี้จะสามารถเคลียร์ให้เป็น 0 ได้โดยใช้คำสั่ง clr bit โดยค่าตำแหน่งของบิทมีค่าเท่ากับ 98H บิทนี้มีประโยชน์ให้รู้ว่าข้อมูลได้เข้ามามาอยู่ใน SBUF ครบทั้งชุดแล้วพร้อมที่ CPU จะอ่านไปเก็บไว้ยังหน่วยความจำต่อไปหรืออาจกำหนดค่าใน REGISTER IE และ IP เมื่อมีข้อมูลเข้ามาทางพอร์ตอนุกรมแล้วจะทำให้เกิดการขัดจังหวะ (INTERRUPT) การทำงานของโปรแกรมหลัก (MAIN PROGRAM) แล้วกระโดดไปทำงานในโปรแกรมตอบสนองการขัดจังหวะ (INTERRUPT SERVICE ROUTINE)

TI TRANSMIT INTERRUPT FLAG

ค่าในบิท TI จะถูกกำหนดให้เป็น 1 หรือ 0 ด้วย HAEDWARE โดยในการส่งข้อมูลแบบอนุกรม MODE 0 BIT นี้จะเป็น 1 เพื่อจะบอกว่าการส่งข้อมูลใน REGISTER SBUF ออกไปทาง PORT อนุกรมครบทั้ง 8 บิทแต่ถ้าเป็นการส่งข้อมูลแบบอนุกรมในโหมดอื่น จะทำให้ BIT TI เป็น 1 เมื่อเริ่มการส่ง STOP BIT ข้อมูลบิทนี้สามารถ CLEAR ให้เป็น 0 ได้ด้วยคำสั่ง CLR BIT โดยที่ค่าตำแหน่งของบิทนี้เท่ากับ 99H บิทนี้ยังมีประโยชน์เพื่อบอกว่าการส่งข้อมูลจาก SBUF ออกไปทางพอร์ตอนุกรมนั้นสิ้นสุดแล้วพร้อมที่จะให้โปรแกรมเขียนข้อมูลลงไปยัง SBUF นอกจากนี้การกำหนดค่าใน REGISTER IE และ IP ยังสามารถที่จะกำหนดให้เกิดการขัดจังหวะการทำงานของโปรแกรมได้เมื่อบิทนี้ถูก HARDWARE ทำให้เป็น 1

PB8

เมื่อมีการกำหนดให้รับข้อมูลในโหมด 2 และ 3 จะใช้บิทนี้สำหรับเก็บข้อมูลบิทที่ 9 ที่เข้ามาทางพอร์ต RXD ส่วนในโหมด 1 นั้น บิทนี้จะเก็บ STOP BIT ซึ่งมีค่าเป็น 1 ค่าตำแหน่งประจำบิทคือ 9AH

TB8

ในการส่งข้อมูลแบบอนุกรมโหมด 2 และ 3 จะใช้บิทนี้เก็บข้อมูลบิทที่ 9 ส่วนโหมดอื่นจะไม่ใช้งานบิทนี้ ค่าในบิทนี้สามารถกำหนดได้โดยใช้คำสั่ง CLR bit ค่าตำแหน่งบิทนี้คือ 9BH

REN RECEIVE ENABLE

เป็นบิทที่จะใช้กำหนดให้ทำการรับข้อมูลเข้ามาจากทางพอร์ตอนุกรม (SERIAL PORT) หรือไม่ถ้าบิทนี้เป็น 1 จะรับข้อมูลเข้ามา แต่ถ้าเป็น 0 ก็จะไม่รับข้อมูลเข้ามาแต่ถ้าเป็น 0 ก็จะไม่รับข้อมูลที่ขา RXD เข้ามาการทำให้บิทนี้เป็น 1 หรือ 0 ทำได้โดยใช้คำสั่ง SETB BIT หรือ CLR bit ค่าตำแหน่งของบิทนี้คือ 9CH

SM2

เป็นบิทสำหรับควบคุมการทำงาน ของ HARDWARE ที่จะทำให้ RI เป็น 1 หรือไม่ในกรณีที่บิท SM2 เป็น 0 ค่าในบิท RI ก็จะเป็นไปตามที่ได้อธิบายมาแล้วในเรื่องบิท RI แต่ถ้า SM2 = 1

ในโหมด 1 บิท RI มีค่าเป็น 1 เมื่อข้อมูล STOP BIT เข้ามายังพอร์ตอนุกรมถูกต้องแต่ถ้า STOP BIT ไม่เข้ามายังพอร์ตอนุกรมอันอาจเกิดจากปัญหาในการส่งข้อมูลแล้วบิท RI จะมีค่าเป็น 0

SMO , SMI

เป็น 2 บิท ที่ใช้งานร่วมกัน เพื่อกำหนดโหมดของการรับ-ส่งข้อมูลของพอร์ตอนุกรม ค่าใน 2 บิท นี้จะกำหนดโหมดได้ดังนี้

SMO	SMI	MODE	DESCRIPTION
0	0	0	SHIFT REGISTER
0	1	1	8-BIT UART
1	0	2	9-BIT UART
1	1	3	9-BIT UART

SERIAL PORT SET-UP

MODE	SCON	SM2 VARIATION
0	10H	
*1	50H	SINGLE PROCESSOR
2	90H	ENVIRONMENT
3	D0H	SM2 = 0

TIMER Register TH0,TL0,TH1,TL1

ตำแหน่งหน่วยความจำภายในเท่ากับ 8CH , 8AH ,8DH ,8BH

ใน 8051จะมีวงจร timer o และ timer 1(8052 จะมี timer 2 อีก 1 ชุด) ใน timer แต่ละชุดจะมี register ขนาด 8 บิทอยู่ 2 ตัว เพื่อเก็บค่าของการนับของ timer ได้สูงสุดถึง 16 บิทใน timer 0 register นี้คือ TH0 , TL0 และใน timer 1 คือ register TH1 , TL1 TLx (x หมายถึง 0 หรือ 1)จะเก็บค่าของการนับ 8 บิทล่าง และ THx จะเก็บค่าของการนับ 8 บิทบน ผู้ใช้จะสามารถ์การทำงานของวงจร timer ในโมด timer หรือโมด Counter ได้โดยการกำหนดใน register ชื่อ TMOD (timer register) การทำงานเป็น timer นั้นจะให้ register ใน timer 0 หรือ 1 ทำการนับจำนวนไซเคิล (cycle) ของสัญญานนาฬิกาที่ผ่านวงจรหาร 12 ดังรูปที่ 5.7 เมื่อมีการนับครบถึงค่าสูงสุดที่ register TLx และ THx จะเก็บได้ คือค่า

FFFFH แล้วยังนับต่อไปค่าที่ได้จากการนับจะเป็น 000H ทำให้เกิดการ Set บิทบางบิทใน register TCON เพื่อบอกสภาวะ timer overflow นี้ ในการให้วงจร timer ทำงานเป็น counter ก็คือการใช้register THx และ TLx ทำการนับจำนวนไซเคิลของสัญญาณที่เข้ามาทางขา T0 หรือ T1 ของ 8051 สัญญาณที่เข้ามาทางขา T0 หรือ T1 อาจมาจากอุปกรณ์ตรวจจับ (sensor) ก็ได้แต่สภาวะของสัญญาณนี้จะต้องมีระดับโวลเตจของสภาวะลอจิก 0 หรือ 1 เป็นแบบ TTL คือลอจิก 0 จะต้องมีโวลเตจไม่เกิน 0.6 โวลท์ และลอจิก 1 จะต้องมีโวลเตจมากกว่า 2.4 โวลท์

TMOD Time / Counter mode register

ตำแหน่งหน่วยความจำภายในเท่ากับ 89H

TMOD เป็นregister ขนาด 8 บิทมีหน้าที่ทำการควบคุมการทำงานของ timer 0 และ timer 1 แต่ละบิทใน register นี้ความหมายเฉพาะดังรูปที่ 5.6

GATE	C/T	M1	M0	GATE	C/T	M1	M0
TIMER 1				TIMER 0			

GATE when TRx [in TCON] is set and GATE = 1.timer/counter Rx will run onty while INTx pin is high [hardware control].When GATE = 0 Timer/Counter Rx will run only while TRx = [Software control].

C/T Timer or Counter serector.Cleared for Timer operation [Input from imtermal system clock.]Set for Counter operation [input from Tx input pin].

M1 Mode selector bit [NOTE 1]

M0 Mode selector bit [NOTE 1]

NOTE 1:

M1	M0	Operating Mode
0	0	0 13 - bit Timer
0	1	1 16 - bit Timer/Counter
1	0	2 8 - bit Auto - Relode Timer/Counter controlled by the standard Timer 0 Control bits. THO is an 8 bit Timer and is controlled by Timer 1 control bits.
1	1	3 [Timer 1]Timer/Counter 1 stopped.

รูปที่ 5.6 TMOD Timer/Counter.Mode Register

ในรูป 5.6 M0 เป็นชื่อของบิท 0 และ GATE ทางซ้ายสุดเป็นชื่อของบิท 7 รีจีสเตอร์นี้แบ่งข้อมูลออกเป็น 2 ชุด ชุดละ 4 บิท คือ บิท 0-3 ใช้สำหรับควบคุมการทำงานของ Timer 0 และบิท 4-7 ใช้ควบคุมการทำงานของ Timer ของแต่ละบิทที่มีชื่อเดียวกันจะเหมือนกัน

GATE เป็นบิทที่ใช้ควบคุมให้ Timer ทำงานหรือไม่ ถ้าบิทนี้ของ Timer x ถูกตั้งเป็น 1 จะทำให้ Timer ทำงานก็ต่อเมื่อที่ขา INTx มีสภาวะลอจิกเป็น 1 และบิท TRx ใน register TCON เป็น 1 ด้วย

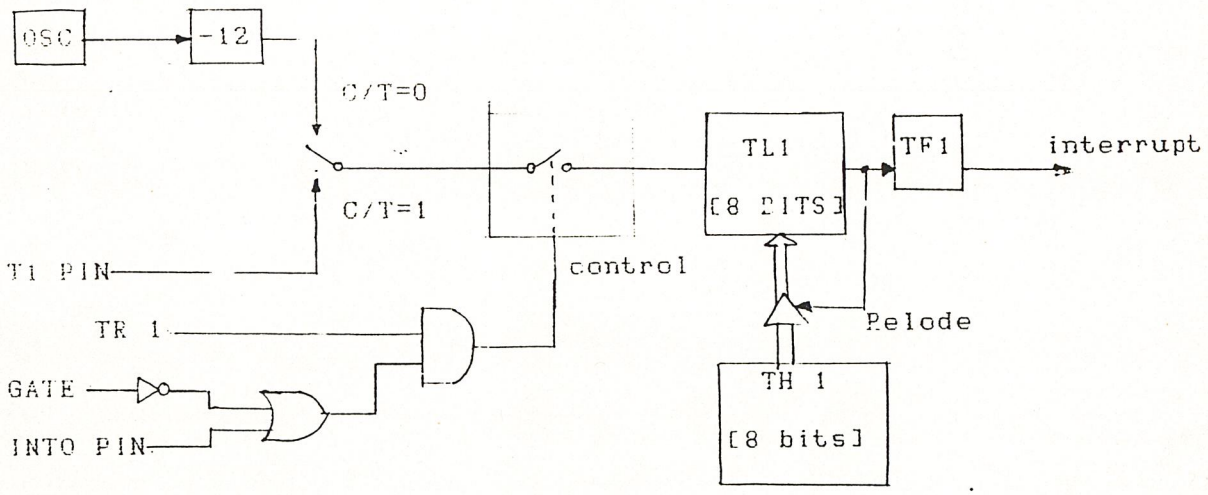
C/T บิทนี้ใช้สำหรับเลือกการทำงานของ Timer ว่าจะใช้เป็น Timer หรือ Counter ถ้าบิทนี้เป็น 1ก็หมายความว่า เลือกการทำงานเป็น Counter ซึ่งจะนับจำนวนไซเคิลของสัญญาณที่เข้ามาทางขา Tx

M1,M0 บิทที่ใช้ร่วมกันเพื่อเลือกโมดการทำงานของ Timer การทำงานโมด 0,1 และ 2 ของ Timer 0 จะเหมือนกับ Timer 1 แต่ในโมด 3 การทำงานของทั้งสองจะต่างกัน ค่าใน M1 และ M0 จะเลือกโมดการทำงานดังนี้

ห้องสมุดคณะวิศวกรรมศาสตร์ จุฬาลงกรณ์มหาวิทยาลัย

M1	M0	การทำงาน
0	0	โมด 0 รีจิสเตอร์ THx และ TLx ทำตัวเป็นตัวนับ 13 บิท ค่าจากการนับ 8 บิทบน มาจาก 8 บิท ของ THx และอีก 5 บิทล่างมาจากค่า 5 บิทล่างของรีจิสเตอร์ TLx โดยที่ 3 บิทบน ของ TLx จะไม่ต้องสนใจเลย
0	1	โมด 1 รีจิสเตอร์ THx และ TLx ทำตัวเป็นตัวนับ 16 บิทค่าจากการนับ 8 บิทบนอยู่ในรีจิสเตอร์ THx และค่าจากการ 8 บิทล่างอยู่ในรีจิสเตอร์ TLx
1	0	โมด 2 ในการนับของ register TLx ขนาด 8 บิทเมื่อนับถึงค่าสูงสุดคือ FFH เมื่อทำการนับต่อไปจะเกิดการ overflow แล้วก็จะ Relode เอาข้อมูลจาก THx เข้าไปยัง TLxเพื่อเป็นค่าเริ่มต้นในการนับครั้งต่อไป
1	1	โมด 3 การทำงานของ Timer 0 และ Timer 1 จะต่างกันดังที่จะกล่าวต่อไป

โมด 2 Auto Relode



รูปที่ 5.8 Timer mode 2

033359

ในรูป 5.8 เป็นไดอะแกรมของวงจร Timer 8051 ที่ทำงานโมด 2 Timer 0 และ Timer 1 มีการทำงานในโมด 2 เหมือนกันโดยสามารถกำหนดให้ทำหน้าที่เป็น Timer หรือ Counter ได้โดยบิท C/T และการควบคุมการนับได้โดยข้อมูลในบิท TR1 และ GATE ในรีจีสเตอร์ TMOD กับสัญญาณที่ขา INTx เมื่อเริ่มการทำงานข้อมูลในรีจีสเตอร์ TH1 จะถูกโหลด ไปยังรีจีสเตอร์ TL1 ทำให้รีจีสเตอร์ TH1 และ TL1 มีการนับเหมือนกัน เมื่อเกิดการนับจำนวนไซเคิลของสัญญาณที่ออกจากสวิตช์ Control จะทำให้ค่าจากการนับในรีจีสเตอร์ TL1 เพิ่มขึ้นไปเรื่อย ๆ ที่ละ 1 ไปจนถึง OFFH ในการนับครั้งต่อไปจะทำให้บิท TF1 ในรีจีสเตอร์ TCON ไม่เป็น 1 และข้อมูลในรีตีสเตอร์ TH1 จะถูกโหลดไปยังรีจีสเตอร์ TH1 จะถูกโหลดไปยังรีจีสเตอร์ TL1 เพื่อเป็นค่าเริ่มต้นการนับต่อไป

TCON TIMER CONTROL REGISTER

ตำแหน่งหน่วยความจำภายในเท่ากับ 088H

รีจีสเตอร์ขนาด 8 บิท นี้ใช้ควบคุมการทำงานและบอกสภาวะของ Timer 0 และ Timer 1 แต่บิทของรีจีสเตอร์นี้จะทำงานต่างกันดังรูปที่ 5.10

TF1	TR1	TF0	TR0	IE1	IT1	IE0	IT0

TF1 TCON.7 Timer 1 is overflow flag Set by hardware When the Timer/Counter 1 overflows.Cleared by hardware as proceesor vectors to the interupt service routine.

TR1 TCON.6 Timer 1 run control bit. Set/cleared by software to turn Timer/Counter1 ONOFF.

TF0 TCON.5 Timer 0 overflow flag.Set by hardware when the Timer/Counter 0 overflow cleared by hardware as proceesor vectors to the service routine.

TR0 TCON.4 Timer 0 run control bit.Set/cleared by software to turn Timer/Counter 0 ON/OFF.

IE1 TCON.3 External Interrupt 1 edge flag.Set by hardware when External interrupt edge is detected cleared by hardware when interrupt is processed

IT1	TCON.2	Interrupt 1 type control bit.Set/cleared by triggered External Interrupt.
IE0	TCON.1	External Interrupt 0 edge flag.Set by hardware When Externternal Interrupt edge detected Cleared by hardware when interrupt is processed
IT0	TCON.0	Interrupt 0 type control bit Set/cleared by software to specify falling edge/low level triggered External Interrupt.

รูปที่ 5.10 TCON Timer Control register

ในรูปที่ 5.10 T0 เป็นชื่อของบิท 7 ในรีจีสเตอร์ TCON แต่ละบิทมีหน้าที่การทำงานดังนี้

IT0 Interrupt 0 เป็นบิทที่จะใช้กำหนดวิธีการขัดจังหวะโปรแกรม อันเนื่องจากสภาวะของสัญญาณที่เข้ามาทางขา INT0

ถ้า IT0 เป็น 0 จะเกิดการขัดจังหวะโปรแกรม ถ้าสภาวะของสาญญาที่ขาINT0 เปลี่ยนจาก 1 เป็น 0

ถ้า IT0 เป็น 0 จะเกิดการขัดจังหวะโปรแกรม ถ้าสภาวะของสัญญาณที่ขา INT0 เป็น 0

IE0 บิทนี้จะเป็น 1 ถ้าสัญญาณที่เข้ามาทางขา INT0 มีสภาวะลอจิกของสัญญาณ ตามที่กำหนดในบิท IT0 แล้วทำให้เกิดการขัดจังหวะโปรแกรม เมื่อเกิดการกระโดด ไปทำงานยังโปรแกรมตอบสนองการขัดจังหวะแล้ว จะทำให้บิทนี้กลับเป็น 0

IT1 Interrupt 1 เป็นบิทที่จะใช้กำหนดวิธีการขัดจังหวะโปรแกรมอันเนื่องจากสภาวะของสัญญาณที่เข้ามาทางขา INT1

ถ้า IT1 เป็น 1 จะเกิดการขัดจังหวะโปรแกรม ถ้าสภาวะของสัญญาณที่ขา INT1 เปลี่ยนจาก 1 เป็น 0

ถ้า IT1 เป็น 0 จะเกิดการขัดจังหวะโปรแกรม ถ้าสภาวะของสัญญาณที่ขา INT1 เป็น 0

IE1 บิทนี้จะเป็น 1 ถ้ามีสัญญาณเข้ามาทาง INT1 มีสภาวะลอจิกของสัญญาณ ตามที่กำหนดในบิท IT1 แล้วทำให้เกิดการขักจังหวะโปรแกรมเหมือนกับบิท IT0 ที่ทำงานกับสัญญาณ INT0

TR0 Timer 0 run control bit บิทนี้ถ้าเป็น 0 Timer 0ไม่ทำการนับสัญญานไม่ว่ากรณีใด ๆทั้งสิ้น แต่ถ้าบิทนี้เป็น 1 จะทำให้ Timer 0 ทำงาน โดยขึ้นสัญญาน GATE,INT0 ดังใน บทที่ 5.1.10 ข้อมูลในบิทนี้จะสามารถ Set เป็น 1 หรือ clear เป็น 0

TF0 Timer 0 overflow flag บิทนี้จะ 1 เมื่อการนับของ registerของTimer0 [TL0 หรือ TH0 ขึ้นกับโมดการทำงาน] เกิด overflow ขึ้น คือเอาการนับเพิ่มไปจนถึงค่าสูงสุดแล้วนับต่อไป ทำให้ค่าการนับกลับมาเริ่มต้นใหม่ที่ 0 หรือค่า Relode เมื่อ 8051 กระโดดไปทำงานที่โปรแกรมตอบสนอง การขัดจังหวะจะทำให้บิทนี้กลับเป็น

TR1 Timer 1 Run Control Bit การทำงานจะเหมือนกันการทำงานของบิท TR0 แต่บิทนี้จะทำงานกับ Timer 1

TF1 Timer 1 Overflow Flag บิทนี้เหมือนกับบิท TF0 ต่างกันที่ขึ้นกับการทำงาน Timer 1

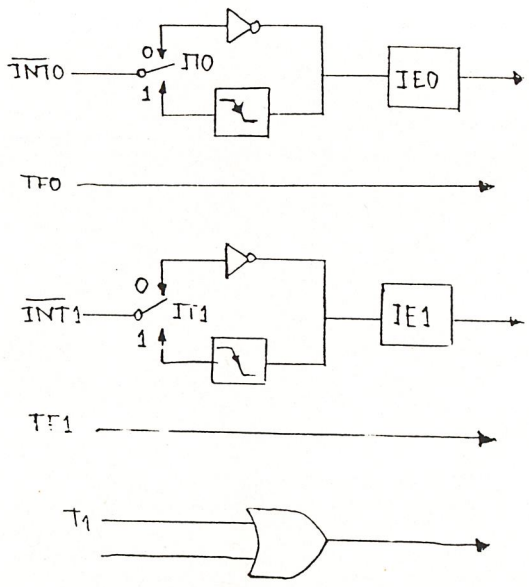
4 บิทแรกที่กล่าวมานั้นจะเกี่ยวข้องกับการขัดจังหวะ [Interrupt] ส่วนที่ 4 บิทหลังนั้นได้กล่าว มาแล้วอย่างละเอียดในเรื่องโมดการทำงานของ Timer

ในขณะที่ Timer ทำงานในโมดของ Timer นั้น รีจิสเตอร์ที่ทำหน้าที่เป็นตัวนับจะมีค่าเพิ่มขึ้น 1 ทุกๆ 1 ไซเคิลของเครื่อง ซึ่งเท่ากับ 12 คาบของสัญญานจากออสซิลเลเตอร์ ในกรณีที่ Timer ทำงานเป็น Counter เพื่อนับจำนวนไซเคิลของสัญญานที่เข้ามาทางขา T0 หรือ T1 รีจิสเตอร์จะเพิ่มค่าไป 1 เมื่อมีการเปลียนสถานะของสัญญานดังกล่าวจาก 1 เป็น 0 โดยวงจรภายใน 8051 จะตรวจสอบสถานะของสัญญานที่ขาดังกล่าวในช่วงเวลาเฟส 2 ของState 5 (55P) ในทุก ๆ 1 ไซเคิลของเครื่อง เช่นในเวลา 55P2ครั้งหนึ่งพบว่าสัญญานที่ขาT0 มีสภาวะลอจิกเป็น 1 และในเวลา 55P2 ของไซเคิลของเครื่องถัดมาพบว่าสัญญานที่ขา T0 มีสภาวะลอจิกเป็น 1 ก็จะทำให้ค่าในรีจิสเตอร์เป็นตัวนับเพิ่มค่าไป 1 แต่ละไซเคิลของเครื่องจะเท่ากับ 12 ไซเคิลของสัญญานจากออสซิลเลเตอร์ ดังนั้นสัญญานที่จะนับได้จะต้องเป็น 1 อย่างน้อยให้ถูกจับได้ใน 1 ไซเคิลของเครื่องและเป็น 0 อย่างน้อยไม่ต้องให้ถูกจับได้ใน 1 ไซเคิลของเครื่องเช่นกัน ทำให้ความถี่สูงสุดของสัญญานที่ขา T0 หรือ T1 ที่จะนับได้ต้องไม่เกิน 1/24 เท่าของความถี่สัญญานจากออสซิลเลเตอร์

IE INTERRUPT ENABLE REGISTER

ตำแหน่งหน่วยความจำภายในเท่ากับ 0A8H

การขัดจังหวะการทำงาน (interrupt) เป็นการที่มีสัญญาณหนึ่งหรือคำสั่งหนึ่งที่ (ไม่ใช่คำสั่ง CALL หรือ JMP) ที่จะทำให้การทำงานปกติของโปรแกรมถูกขัดจังหวะแล้วข้ามไปทำงานยังตำแหน่งหนึ่งตำปหน่งใดที่กำหนดไว้ เมื่อทำงานในโปรแกรมขัดจังหวะเสร็จสิ้นก็จะกลับมาทำงานในโปรแกรมที่ตำแหน่งก่อนที่จะไปทำงานยังโปรแกรมขัดจังหวะ ฝปรแกรมที่ถูกกระโดดไปทำงาน เรียกว่า โปรแกรมตอบสนองการขัดจังหวะ [Interrupt Service Routine] ใน 8051 จะสามารถขัดจังหวะด้วยสัญญาณจาก 6 แหล่ง ดังรูปที่ 5.11 ถ้าเป็น 8052 หรือ 83154 จะสามารถขัดจังหวะได้ด้วยสัญญาณจาก 8 แหล่งคือสัญญาณในชุดล่างสุดของรูปที่ 5.11



รูปที่ 5.11 Interrupt source

สัญญาณขัดจังหวะที่ 5 ในรูปที่ 5.11 จะสามารถทำให้เกิดการขัดจังหวะได้ 2 วิธี คือ มีข้อมูลเข้ามาทางพอร์ทอนุกรมเก็บอยู่ที่รีจีสเตอร์ SBUF และในกรณีที่ข้อมูลใน SBUF ส่งออกไปทางพอร์ทอนุกรมหมดแล้ว ไม่ว่าเกิดกรณีใด ๆ ก็ทำให้เกิดการขัดจังหวะขึ้น

สัญญาณภายนอกที่เข้ามายัง 8051 ทางขา INT0 และ INT1 จะสามารถทำให้เกิดการขัดจังหวะการทำงาน 8051 ได้ (สัญญาณที่ 1 และ 3 ในรูปที่ 5.11) โดยสภาวะของสัญญาณนั้นเปลี่ยนจาก 1 เป็น 0 หรือ เมื่อสัญญาณนั้นเป็น 0 แล้วแต่การกำหนดในบิท IT0 และ IT1 ของรีจีสเตอร์ TCON จะทำให้บิท IE0 กับ IE1 เป็นตัวสร้างสัญญาณขัดจังหวะต่อไป

จาก Timer 0 และ Timer 1 เมื่อค่าการนับในแต่ละโมดถึงค่าสูงสุดในโมดนั้นแล้วเมื่อทำการนับต่อไป ค่าการนับต่อไปจะเป็น 0 (หรืออาจเป็นค่าที่ Relode จาก THx ในโมด 2) และทำให้บิท TF0,TF1 เป็น 1 ซึ่งสัญญาณจาก 2 บิทนี้จะสามารถทำให้เกิดการขัดจังหวะได้เช่นกันดังเช่นสัญญาณขัดจังหวะที่ 2 และ 4 ในรูปที่ 5.11

แหล่งกำเนิดสัญญาณทั้ง 6 ที่สามารถทำให้เกิดการขัดจังหวะได้ 5 แบบ นี้ผู้ใช้สามารถกำหนดให้สัญญาณใดบ้างขัดจังหวะเรียกว่า Enable หรือไม่ให้เกิดการขัดจังหวะเรียกว่า Disable โดยการกำหนดในรีจีสเตอร์ IE [Interrupt Enable Register] ซึ่งมี 8 บิท แต่ละบิทสามารถ Enable ให้ขัดจังหวะได้จากแต่ละสัญญาน ดังรูปที่ 5.12

MSB LSB

EA	X	ET2	ES	ET1	EX1	ET0	EX0

SYMBOL	POSITION	FUNCTION
EA	IE.7	disables all interrupts. if IA=0, no interrupt will be acknowledged if EA=1 each interrupt source is individually enabled or disabled by setting or clearing its enable bit.
-	IE.6	reverseed
ET2	IE.5	enables or disable the Timer 2 Overflow or capture interrupt .if ET2=0.the Serial Port interrupt is dissabled.

ES	IE.4	enables or disables the Serial Port interrupt .if ES=0.the Timer 1 interrupt is disable.
ET1	IE.3	enables or disables the Timer 1 Overflow interrupt is disabled.
EX1	IE.2	enables or disables External interrupt1. if EX1=0 .External interrupt 1 is disables.
ET0	IE.1	enables or disables the Timer 0 Overflow interrupt .if ET0=0.the Timer 0 interrupt is disabled.
EX0	IE.0	enables or disables External interrupt if EX0=0.External interrupt 0 is disabled

รูปที่ 5.2 Interrupt Enable Register

ถ้าต้องการ Enable บิทใดก็ให้โปรแกรมกำหนดค่าในบิทนั้นเป็น 1 ถ้าค่าในบิทนั้นเป็น 0 หมายถึง Disable การ Disable จะทำให้ไม่มีการขัดจังหวะการทำงานของโปรแกรมเนื่องจากสัญญานขอขัดจังหวะนั้น ๆ EX0 เป็นชื่อบิท 0 และ EA เป็นชื่อของบิท 7

EX0 บิทนี้ใช้สำหรับการ Enable สัญญานที่เข้ามาทางขา INT0 ให้เกิดการขัดจังหวะหรือไม่

ET0 Timer 0 Interrupt Enable Bit ข้อมูลบิทนี้จะใช้ Enable หรือ Disable สัญญานขัดจังหวะจาก Timer 0 [TF1]

EX1 บิทนี้จะใช้ Enable หรือ Disable สัญญานที่เข้ามาทางขา INT1 ให้เกิดการ ขัดจังหวะหนือไม่

ET1 Timer 1 Interrupt Enable Bit บิทนี้จะใช้ Enable หรือ Disable สัญญานขัดจังหวะจาก Timer 1 [TF1]

ES ข้อมูลในบิทนี้จะ Disable หรือ Enable การขัดจังหวะจาก Serial Port อันเนื่องจากมีข้อมูลเข้ามายัง SBUF หรือข้อมูลจาก SBUF ได้ส่งออกไปทาง Serial Port หมดแล้ว

ET2 Timer 2 Internal Enable Bitจะใช้งานเฉพาะใน 8052 และ 83152 เท่านั้นที่บิทจะใช้ Enable หรือ Disable สัญญานขอขัดจังหวะที่มาจาก Timer 2 (สัญญานที่ 6 ในรูปที่ 5.11)

EA บิทนี้จะควบคุมทั้ง 6 บิทที่กล่าวมาแล้วถ้าข้อมูลในบิทนี้เป็น 0 จะเป็นการ Disable ทุกบิทที่กล่าวมาแล้ว ทำให้ไม่เกิดการขัดจังหวะโปรแกรมได้เลย แต่ถ้าบิทนี้เป็น 1 การ Enable/Disable ใน 6 บิทที่กล่าวมาแล้วจะขึ้นกับข้อมูล ในแต่ละบิทนั้น

หมายเหตุ: บิทอื่นนอกจากนี้จะไม่มีการใช้งาน

การกำหนดให้บิทใด Enable หรือ Disable นั้นจะเป็นไปโดยอิสระไม่ขึ้นแก่กัน จึงสามารถกำหนดให้บิทใดหรือมากกว่า 1 บิท Enable ก็ได้ ดังนั้น 8051 จึงมีรีจีสเตอร์อีกตัวที่ใช้เลือกว่าถ้ามีสัญญานขอการขัดจึงหวะโปรแกรมเข้ามาพร้อมกันมากกว่า 1 แล้วจะทำโปรแกรมตอบสนองการขัดจังหวะอันใดก่อน รีจีสเตอร์นั้นคือ IP Interrupt Pri-ority Register

การรับ - ส่งข้อมูลทางพอร์ทอนุกรม

ในการรับ - ส่งข้อมูลแบบอนุกรมนั้น จะต้องมีการกำหนดโมดการทำงานในรีจีสเตอร์ TCON และในบางโมดของการทำงานจะสามารถกำหนดอัตราการส่งข้อมูลได้โดยการโปรแกรมใน Timer ข้อมูลที่จะส่งออกหรือรับเข้าทางพอร์ทอนุกรมจะอยู่ที่รีจีสเตอร์ SBUF การทำงานของวงจรภายในแต่ละโมดมีดังนี้

การทำงานโมด 1

การส่งข้อมูล

จากรูปที่ 5.18 บิท SMOD จะเป็นตัวเลือกว่า สัญญาน Timer 1 Overflow ที่ส่งไปยังวงจรหาร 16 จะถูกหาร 2 ก่อนหรือไม่ ถ้า SMOD เป็น 1 สัญญาน Timer 1 Overflow จะถูกหาร 2 ก่อนที่จะเข้าวงจรหาร 16 การส่งข้อมูลจะเริ่มจากการที่มีคำสั่งเขียนข้อมูลไปยังรีจีสเตอร์ SBUF จะมีสัญญาน Write to SBUF เกิดขึ้นเพื่อรับข้อมูลจาก

InternalBusด้านบนไปเก็บไปยังรีจิสเตอร์SBUFและทำให้ outputของ D FLIP FLOP ทางซ้ายของ SBUF มีค่าเป็น 1 และเป็นบิทที่ 9 ของการส่งข้อมูล สัญญาน Write to SBUF ยังส่งไปยัง TX control ด้วย ขณะนี้ข้อมูลในวงจรหาร 16 มีค่าเป็นอะไรไม่ทราบจึงจะรอจนกว่าข้อมูลในวงจรหาร 16 นับเพิ่มขึ้นจนถึงค่าสูงสุดแล้ววนกลับเป็น 0 คือเกิดการวนกลับทำให้เริ่มการส่งข้อมูลที่เวลา S1P1 ของไซเคิลเครื่องถัดไป (การส่งข้อมูลออกจะสัมพันธ์กับการเกิด Overflow ในวงจรหาร 16) สัญญาน SEND จาก TX Control เปลี่ยนสภาวะลอจิกเป็น 0 แล้วเริ่มส่งข้อมูลที่เป็น Start bit (0) ออกไป เมื่อส่งออกไปแล้ววงจร TX Control ก็ทำให้สัญญาน DATA เป็น 1 เพื่อเลื่อนข้อมูลใน SBUF ออกไป เริ่มจากบิท 0 จนถึงบิทที่ 7 การส่งข้อมูลถูกเลื่อนออกไปนั้นจะมี 0 เลื่อนเข้ามาทางซ้ายของรีจิสเตอร์ SBUF เมื่อข้อมูลเลื่อนออกไปทั้ง 8 บิทแล้วบิทที่ 9 ซึ่งเป็น 1 และตอนต้นอยู่ทางซ้ายสุดจะถูกเลื่อนมาอยู่ในตำแหน่งสุดท้ายทางขวาของรีจิสเตอร์ SBUF และทางซ้ายของหลักนี้จะมี 0 อยู่ทั้ง 8 บิทใน SBUF ทำให้ Zero Detector รู้ว่าเป็นข้อมูลบิทสุดท้ายแล้วที่ส่งออกโดยจะมีสัญญานมาบอกกับวงจร Tx Control ด้วย เมื่อ Tx Controlส่งสัญญาน Shift ออกไปเป็นการส่งข้อมูลบิทสุดท้าย (บิท 7) ออกไป ก็จะรออีก 1 Tx Clock [Bit Clock] ก็จะทำให้ขา TXD ส่งข้อมูล Stop Bit (1) ออกมา สัญญาน DATA ซึ่งมีสภาวะลอจิกเป็น 1 มาตั้งแต่เริ่มส่งข้อมูลบิท 0 ก็จะกลับเป็น 0 และบิท T1 ก็จะเป็น 1 เพื่อบอกการสิ้นสุดการส่งข้อมูลทั้งหมดเมื่อสัญญาน TX Clock ไซเคิลที่ 10 นับตั้งแต่สัญญาน SEND เปลี่ยนสภาวะลอจิกเป็น 0

$$\text{Baud rate} = \frac{2^{SMOD}}{32} * (\text{Timer 1 Overflow Rate})$$

ในขณะที่ใช้ Timer 1 เป็นตัวกำหนด Baud Rate นี้จะต้อง Disable ไม่ให้เกิดการขัดจังหวะเนื่องมาจากการ Overflow Timer 1 อาจใช้ในโมดของ Timer หรือ Counter ก็ได้ ซึ่งเมื่อการนับโดยรีจิสเตอร์ตัวนับมีค่าสูงสุดแล้วกลับมาเป็น 0 ก็จะเกิด Overflow เช่นเดียวกัน แต่โดยปกติแล้วจะใช้ Timer 1 นี้ในโมดของ Timer ที่มีการทำงานแบบ Auto Relode โมด 2 เพื่อว่าเมื่อค่าในการนับโดยรีจิสเตอร์ TL1 ถึงค่าสูงสุดก็จะโหลดค่าในรีจิสเตอร์ TH1 มาไว้ใน TL1 สำหรับเป็นค่าเริ่มต้นการนับต่อไป ซึ่ง Baud rate จะมีค่า

$$\text{Baud rate} = \frac{2^{SMOD}}{32} * \frac{\text{Oscillator frequency}}{(12 * 1256 - [TH\ 1])}$$

โดยที่ SMOD เป็นบิทหนึ่งในรีจีสเตอร์ PCON

เช่นความถี่ของออสซิลเลเตอร์เท่ากับ 11.059 MHz บิท SMOD = 0 รีจีสเตอร์ TH1 มีค่า E8H,Timer 1 ทำงานในโมด 2 จะได้ว่าอัตราการส่ง-รับข้อมูลแบบอนุกรม

$$= \frac{2^{0}}{32} * \frac{11.059 * 10^{6}}{12 * (256 - 232)}$$

$$= \frac{1}{32} * \frac{11.059 * 10}{12 * 24}$$

= 1200 บิท / วินาที

มาตราฐาน RS-232

พอรต์อนุกรมส่วนใหญ่ จะมีรูปร่างขึ้นอยู่กับมาตรฐาน RS-232คือมี 25 ขา(แต่ IBM AT จะมีเพียง9 ขา) แต่พอร์ตส่วนใหญ่จะมีสัญญาณไม่เหมือนสัญญาณไม่เหมือนสัญญาณของ RS-232 ทั้งหมดเพราะว่าบางสัญญาณไม่จำเป็นต้องใช้สัญญาณพื้นฐานของ RS-232 แสดงดังตารางที่ 6.1

สัญญาณ	ชื่อย่อ	หมายเลขขา
REQUEST TO SEND	RTS	4
CLEAR TO SEND	CTS	5
DATA SEND READY	DSR	6
DATA TERMINAL READY	DTR	2O
TRANSMIT DATA	TXD	2
RECEIVE DATA	RXD	3
GROUND	GND	7

สัญญาณทั้งหมดมีมากกว่านี้ เพราะว่าเริ่มแรกนั้น พอร์ตอนุกรมถูกออกแบบมาเพื่อใช้งานร่วมกับโมเด็ม ดังนั้นเมื่อนำไปใช้กับอุปกรณ์อื่น บางสัญญาณจึงไม่จำเป็น เพราะสัญญาณเหล่านี้มีไว้เพื่อเป็นข้อตกลงระหว่างโมเด็ม และคอมพิวเพอร์ว่า

1) คอมพิวเตอร์จะไม่ส่งข้อมูลให้แก่โมเด็ม ก่อนที่โมเด็มจะพร้อมส่งข้อมูล

2) คอมพิวเตอร์จะไม่อ่านข้อมูลจากโมเด็ม ก่อนที่โมเด็มจะพร้อม

FRAMING ERROR

คือความผิดพลาด ของการส่งข้อมูลที่เกิดจากสัญญาณนาฬิกา ที่ควบคุมการทำงานของอุปกรณ์ทั้ง 2 ด้านที่ไม่เท่ากัน เพราะว่าจากกการทำงานของพอร์ตอนุกรม เมื่อพอร์ตได้รับบิทเริ่มต้นก็จะสุ่มอ่านคำจากส่วนรับข้อมูล 1 ครั้งต่อ 1 รอบ เพื่ออ่านบิทต่อไป ซึ่งระยะเวลาในการสุ่มอ่านแต่ละรอบกำหนดได้จาก BAUD RATE ถ้าหากว่า COMPUTER ทั้ง 2 เครื่องมีสัญญาณนาฬิกาไม่ตรงกัน CPU ทางด้านรับจะอ่านข้อมูลจากส่วนรับข้อมูลของตน ช้าเกินไปหรือเร็วเกินไป ก่อนที่ข้อมูลจะถูกส่งจากคอมพิวเตอร์ด้านส่ง ทำให้เกิด FRAMING ERROR ขึ้น

ปัญหาของการสื่อสาร

ในการส่งและรับข้อมูลความจริงแล้วจะต้องใช้ขาสัญญาณหลาย ๆ ขาเพื่อใช้ในการบอกสถานะของฝ่ายส่งและฝ่ายรับ ว่าพร้อมที่จะรับหรือส่งหรือยัง แต่เพื่อที่จะลดค่าใช้จ่ายเรื่องสายส่ง จะลดขาสัญญาณเหลือเพียง TXD , RXD และ GND (แต่ใน PROJECT นี้จะใช้เฉพาะ TXD กับ GND) ซึ่งจะทำให้เกิดปัญหาการเขียนทับของข้อมูล (OVERRUN ERROR)

เมื่อการติดต่อผ่านพอร์ตอนุกรมใช้สายส่งเพียง 2 สายนั้นจะต้องมีกรรมวิธีพิเศษเล็กน้อยเพื่อให้พอร์ตของตัวส่งเข้าใจว่าพอร์ตด้านรับพร้อมที่จะรับข้อมูลตลอดเวลา โดยการต่อขา 6 , 8 และขา 20 ของ CONNECTOR เข้าด้วยกัน วิธีการเช่นนี้เป็นการตัดฮาร์ดแวร์แฮนด์เชคกิ้งไปแต่การทำเช่นนี้จะทำให้เกิดปัญหาข้อมูลที่ส่งไปชุดก่อนหน้าไม่เสร็จทำให้เกิดปัญหาข้อมูลถูกเขียนทับได้ง่าย ซึ่งเกิดจากการส่งข้อมูลชุดใหม่ถูกเขียนทับข้อมูลชุดก่อนหน้า

การใช้งาน PORT อนุกรมสามารถทำได้ 3 วิธีด้วยกันคือ

1) ผ่านทางดอส

2) ผ่านทาง BIOS

3) การเขียนโปรแกรมควบคุมพอรต์อนุกรมโดยตรง

หมายเหตุ : 1) การเรียกใช้พอร์ตอนุกรมผ่านทางดอสนั้นไม่เหมาะสมเท่าใดนักเพราะว่าดอสไม่มีวิธีที่จะตรวจดูสถานะของการรับส่งและตัวพอร์ตอนุกรมว่าการรับส่งข้อมูลถูกต้องหรือไม่เพียงใด

2) การเขียนโปรแกรมควบคุมการทำงานของพอร์ตอนุกรมโดยตรงนั้นคงไม่จำเป็นเพราะมีวิธีที่ดีกว่าและง่ายกว่า คือการเรียกใช้ผ่าน BIOSอินเทอรัพที่หมายเลข 14

การเตรียมสถานะเริ่มต้นของพอร์ต

ในการเตรียมสถานะของพอร์ตอนุกรมเราสามารถทำได้ โดยผ่าน INTERRUPT หมายเลข 14 (INT 14) ฟังก์ชั่นหมายเลข 0 โดยมี REGISTER AH เป็นตัวเก็บตัวผ่านค่าหมายเลขของฟังก์ชั่น (ในที่นี้คือ 0) REGISTER AL ผ่านค่ารหัสที่จะตั้งสถานะของพอร์ตอนุกรมโดยมีความหมายดังตาราง 6.2

หมายเลขประจำบิท	ความหมาย
7 , 6 , 5	อัตรารับส่งข้อมูล 000 = 110 BAUD 001 = 150 BAUD 010 = 300 BAUD 011 = 600 BAUD 100 = 1200 BAUD 101 = 2400 BAUD 110 = 4800 BAUD 111 = 9600 BAUD
4 , 3	PARITY BIT 00 หรือ 01 = ไม่มีพาริตี้ 01 = พาริตี้คี่ 11 = พาริตี้คู่
2	จำนวนของบิทสิ้นสุด 0 = 1 STOPBIT 1 = 2 STOPBIT
1 , 0	จำนวนบิทในข้อมูล 1 BYTE 10 = 7 BIT 11 = 8 BIT

สำหรับโปรเจคนี้มีการเตรียมพอร์ตอนุกรมไว้ดังนี้คือ อัตราการรับส่ง 2,400 BUAD ไม่มีพาริตี้มี 1 STOP BIT และใช้ 8 BIT ต่อ 1 BYTE ข้อมูล

บิทที่	7	6	5	4	3	2	1	0
รหัส	1	0	1	0	0	0	1	1

เครื่องคอมพิวเตอร์ทั่วไปจะมี PORT อนุกรมได้มากถึง 7 พอร์ต โดยหมายเลขพอร์ตที่จะใช้สามารถกำหนดผ่าน รีจีสเตอร์ DX โดยพอร์ตแรกมีหมายเลข 0 และ PORT ต่อไปหมายเลข 1 เช่นนี้ต่อไปเรื่อย ๆ

```
   /* ฟังก์ชันนี้ทำหน้าที่เตรียมสถานะเริ่มต้นของพอร์ต */
void port_init(port,code)
int port;
unsigned char code;
{
  union REGS r;
  r.x.dx = port;      /* หมายเลขพอร์ตอนุกรม */
  r.h.ah = 0;         /* เรียกฟังก์ชันหมายเลข 0 ทำหน้าที่เตรียมสถานะ */
  r.h.al = code;      /* รหัสตั้งสถานะเริ่มต้น */
  int86(0x14,&r,&r);
}
```

<u>การรับข้อมูล 1 BYTE ผ่านทางพอร์ตอนุกรม</u>

INTERRUPT หมายเลข 14 FUNCTION หมายเลข 2 ของ BIOS ทำให้หน้าที่รับข้อมูล 1 BYTE ทางพอร์ตอนุกรม โดยมี REGISTOR DX ใช้สำหรับกำหนดหมายเลขพอร์ตข้อมูลที่อ่านได้จะอยู่ใน REGISTER AL เมื่ออ่านข้อมูลเสร็จแล้ว สถานะของข้อมูลและพอร์ตจะสามารถตรวจดูได้ที่บิต 7 ของ REGISTER AH

```
   /* ฟังก์ชันนี้ทำหน้าที่อ่านตัวอักษรจากพอร์ตอนุกรม */
rport(port)
int port;  /* หมายเลขพอร์ต */
{
  union REGS r;
  /* รอจนกว่าจะได้รับตัวอักษร */
  while(!(check_stat(PORT)&256))
    if(khbit())  {    /* ยกเลิกเลิกเมื่อมีการกดคีย์ */
        getch();
        exit(1);
    }
  r.x.dx = port;    /* กำหนดหมายเลขพอร์ต */
  r.h.ah = 2;       /* ฟังก์ชันหมายเลข 2 ทำหน้าที่อ่านตัวอักษร */
  int86(0x14,&r,&r);
  if(r.h.ah & 128)
     printf("resd error detected in serial port");
  return r.h.al;
}
```

การตรวจสอบสถานะของพอร์ตอนุกรม

ฟังก์ชั่นหมายเลข 3 ของINTERRUPT 14 ของ BIOS ไว้ตรวจดูสถานะของพอร์ตอนุกรม REGISTER DX ใช้สำหรับเก็บหมายเลขพอร์ต ที่จะตรวจสอบสถานะของพอร์ตสามารถแปลความหมายได้จากรหัสที่ปรากฏใน REGISTER AH ดังตารางที่ 6.3

สถานะของสายส่ง (AH)	
ความหมายของแต่ละบิทเมื่อมีค่าเป็น 1	ตำแหน่งของบิต
DATA READY	0
OVERRUN ERROR	1
PARITY ERROR	2
FRAMING ERROR	3
BREAK-DETECT ERROR	4
TRANSFER HOLD-REGISTER EMPTY	5
TRANSFER SHIFT-REGISTER EMPTY	6
TIME-OUT ERROR	7

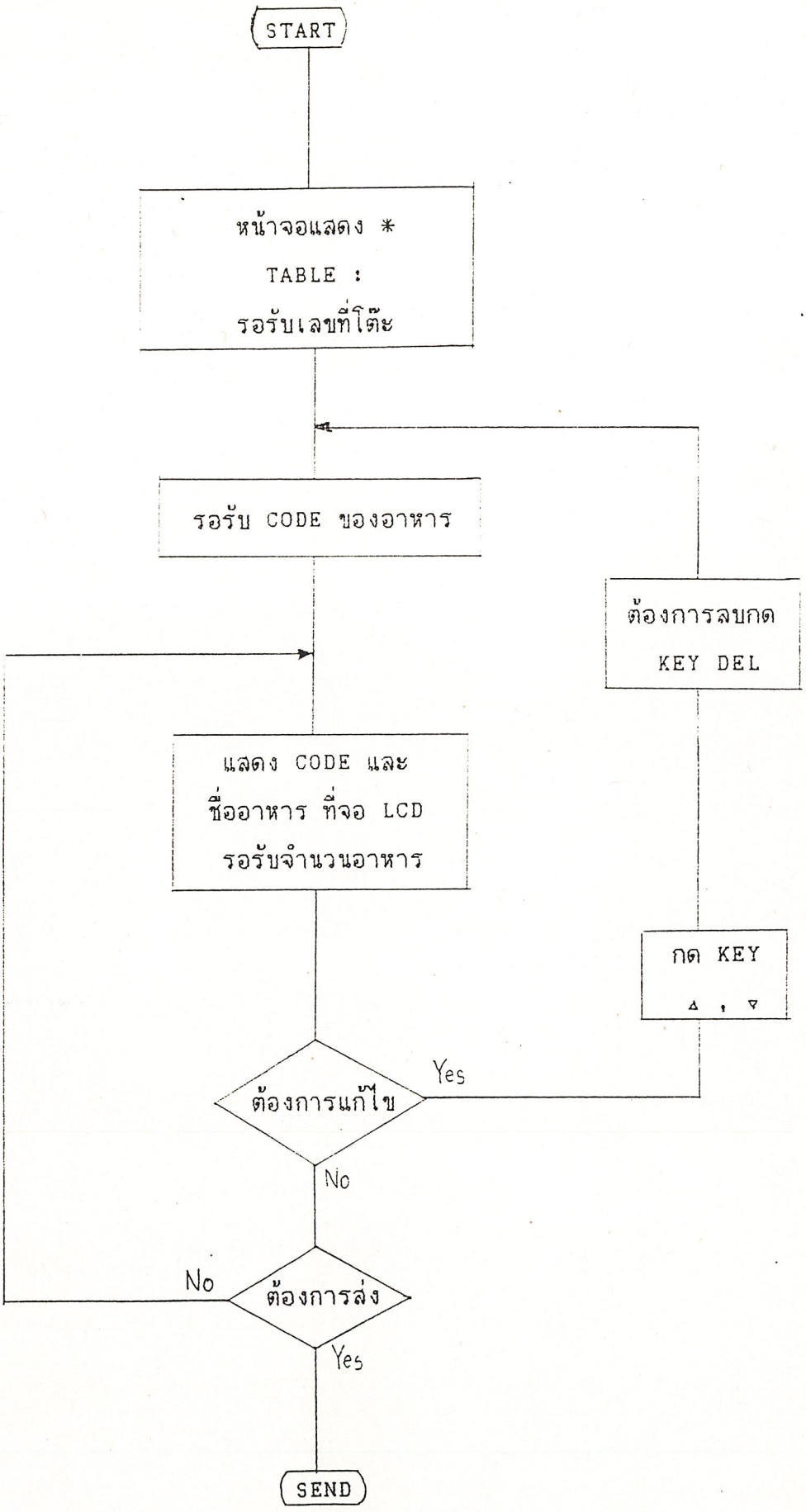
ตัวรับเราจะใช้ตัวรับแบบ MODULE ของ SONY signal ที่ออกจากขา signal จะมีสถานะเป็น high ขณะที่ไม่มีสัญญาณมาจากตัวส่ง และจะมีสถานะ low เมื่อมีสัญญาณมาจากตัวส่ง ภายใน MODULE จะมีวงจร PHASE LOCK LOOP ซึ่งสามารถ lock frequency ได้ในช่วง 20 - 60 KHz

ปัญหาของการส่งด้วยอินฟาเรด

เนื่องจาก PORT RS-232 ของ computer มีสัญญาณรบกวนสูง จะออกมากวนตัวรับ IR ทำให้ ตัวรับส่งสัญญาณออกมาทางขา signal ทำให้ computer รับสัญญาณที่ไม่ถูกต้องไป และคลื่นที่ออกจากหลอก fuorescent สามารถ กวนตัวรับได้วิธีแก้ คือต้องออกแบบวงจรเพื่อให้ลด noise ให้ดี

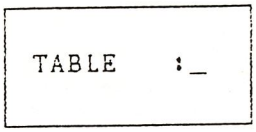
บทที่ 2 การประยุกต์ใช้ในการสั่งอาหาร

- การทำงานภาคส่ง
- การทำงานภาครับ
- ภาคส่งและรับอินฟราเรด
- โปรแกรมภาครับ
- โปรแกรมภาคส่ง
- รูปแบบวงจร


START
หน้าจอแสดง *
TABLE :
รอรับเลขที่โต๊ะ
รอรับ CODE ของอาหาร
ต้องการลบกด
KEY DEL
แสดง CODE และ
ชื่ออาหาร ที่จอ LCD
รอรับจำนวนอาหาร
กด KEY
▵ , ▿
ต้องการแก้ไข
Yes
No
ต้องการส่ง
No
Yes
SEND

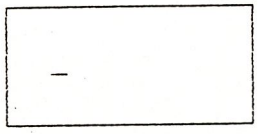
การทำงานภาคส่ง

เริ่มต้น เปิดเครื่อง หรือ reset ที่จอ LCD จะปรากฏ



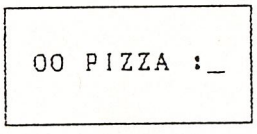
รูปที่ 1

cursor จะกระพริบหลัง : เป็นการรอการป้อนเลขที่โต๊ะ หลังจากป้อนเลขที่โต๊ะแล้วหน้าจอ LCD จะ CLEAR . cursor จะกระพริบอยู่ซ้ายสุด ซึ่งรอรับ code ของอาหาร



รูปที่ 2

หลังจากป้อนรหัสของอาหารแล้วหน้าจอจะ show รหัสของอาหารและตามด้วยชื่ออาหาร แล้ว cursor จะกระพริบอยู่หน้า : ซึ่งรอรับจำนวนที่จะสั่ง



รูปที่ 3

พอป้อนจำนวนอาหารแล้วหน้าจอจะ CLEAR ดังรูปที่ 2 เพื่อรอรับรหัสอาหารชุดต่อไป นอกจากนี้ยังมี KEY UP ▵ , DOWN ▿ เพื่อเลื่อนไปดูรายการที่สั่งแล้ว ถ้าต้องการลบก็ใช้ KEY ▵ , ▿ เลื่อนไปหารายการอาหารนั้นแล้วกด KEY DEL จะเห็นว่าจำนวนของอาหารจะหายไปซึ่งเราสามารถใส่จำนวนอาหารใหม่เข้าไปได้ แต่ถ้าไม่ใส่จำนวนอาหารใหม่เข้าไป แต่กด DEL ซ้ำอีกครั้ง รายการอาหารนั้นจะถูกลบออกไป โดยที่หน้าจอ LCD จะกระพริบ 2 ครั้ง แล้วแสดงคำว่า DELETE เป็นเวลาประมาณ 30 วินาที แล้วจะโชว์รายการอาหารถัดไป

หลังจากแก้ไขเรียบร้อยแล้วถ้าต้องการส่งก็กด KEY SEND ซึ่งหน้าจอจะโชว์

TOTAL ORDER : Σn

; Σn คือ จำนวนรายการทั้งหมด

ข้อมูลก็จะถูกส่งออกไป

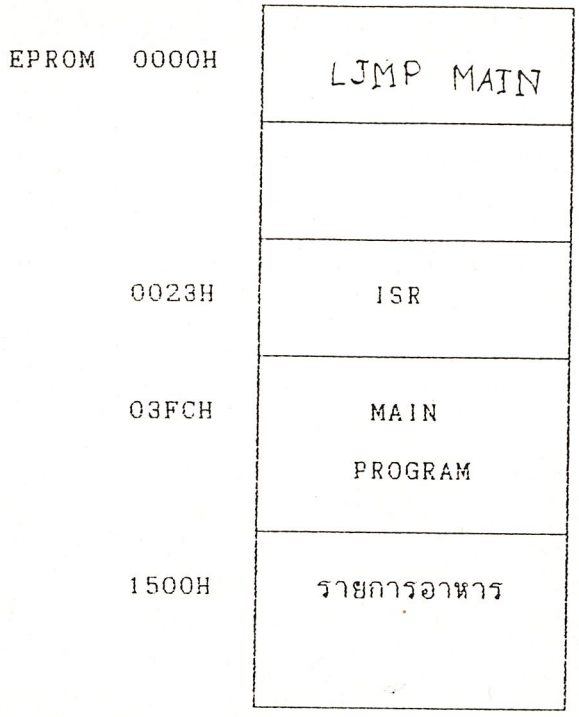
ตำแหน่ง KEY

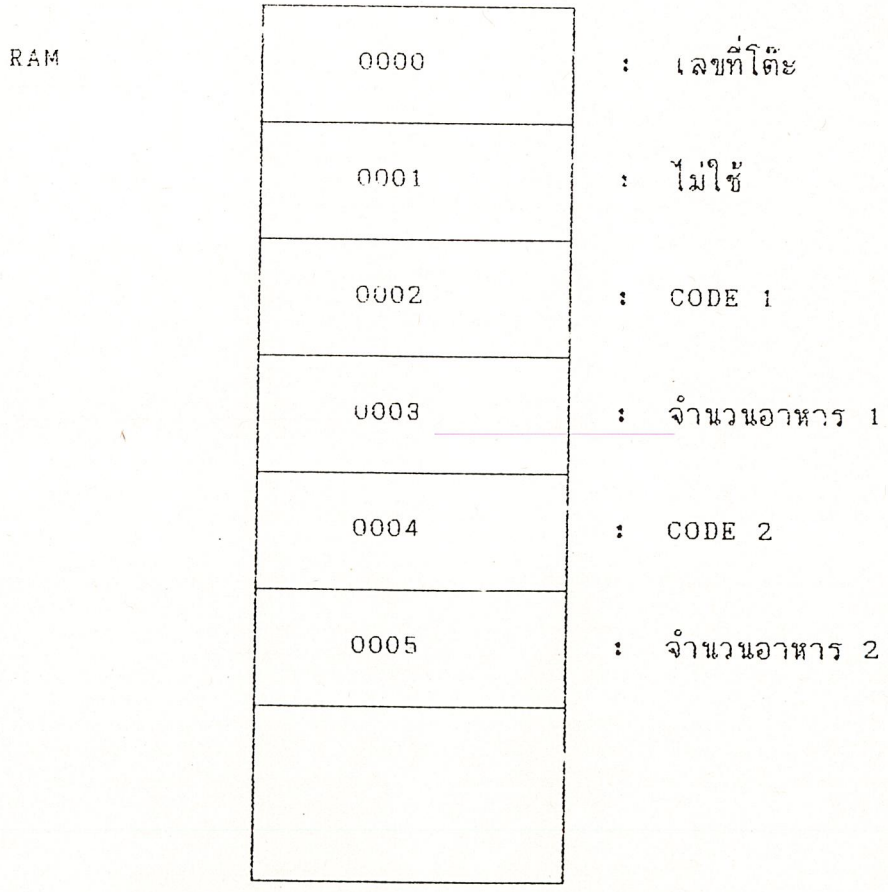
△	▽	SEND	RESET
0	1	2	3
4	5	6	7
8	9	ENTER	DEL

รายการอาหาร

CODE	
00	PIZZA
01	SLAD
02	STEAK
03	HOTDOG
04	HAMBERGER
05	CHICKEN PIE
06	APPLE PIE
07	ONION SOUP
08	DONUT
09	HOT COFFEE
10	ICE COFFEE
11	ICE TEA
12	COCA COLA
13	PUNCH
14	JUICE
15	MILK
16	BEER

MEMORY MAP


EPROM
0000H
LJMP MAIN
0023H
ISR
03FCH
MAIN
PROGRAM
1500H
รายการอาหาร


RAM
0000
: เลขที่โต๊ะ
0001
: ไม่ใช้
0002
: CODE 1
0003
: จำนวนอาหาร 1
0004
: CODE 2
0005
: จำนวนอาหาร 2

การทำงานภาครับ

เรียก rec22.exe ซึ่งเป็น file ที่ compile ด้วย Turbo C compiler ที่หน้าจอจะแสดง

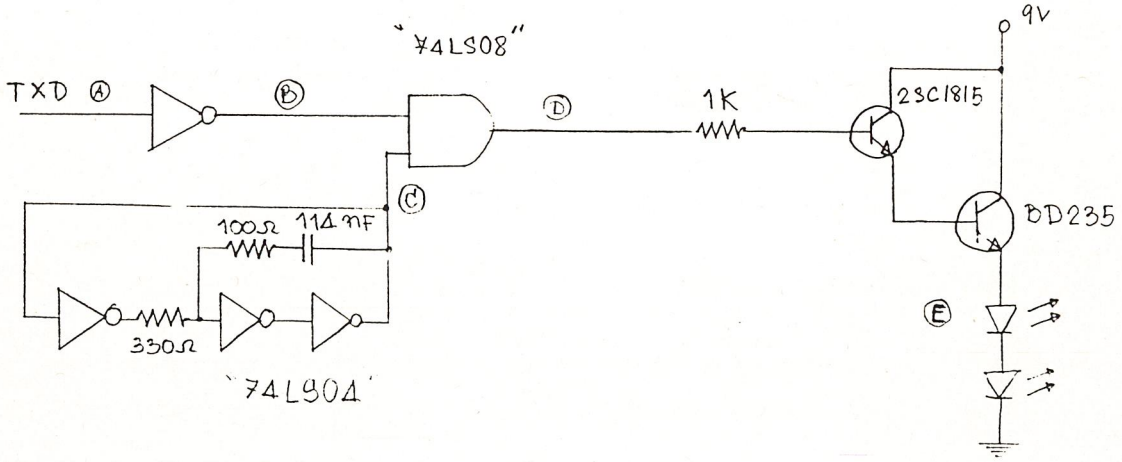
```
... WAITTING FOR DATA...
```

เมื่อตัวส่ง ส่งข้อมูลมาแล้ว ที่หน้าจอจะปรากฏ

```
                TABLE :
-------------------------------------
 food & drink  unit  price/unit  price
-------------------------------------
1.  PIZZA       2        75       150
2.  DONUT       2        14        28
3.  COCA COLA   2        13        26
                                 -----
                        TOTAL    204
                                 -----
```

ภาคส่ง-รับ INFARED

เนื่องจากการใช้อินฟาเรดเป็น carrier เราจะต้องใช้ความถี่ประมาณ 20 KHz ในการ SWITCHING ทรานซิสเตอร์เพื่อขับตัวส่งอินฟาเรด แต่ข้อมูลที่ออกจาก serial port(TXD) มี baud rate 9600 baud จะเห็นว่าความถี่ที่เป็นไปได้สูงสุดแค่ 4800 Hz เพราะฉะนั้นเราต้อง MOD CARRIER เข้าไปกับข้อมูลในการ MOD เราจะนำข้อมูลที่ ออกมาจาก TXD มา and กับ carrier ดังแสดงไว้ดังรูป

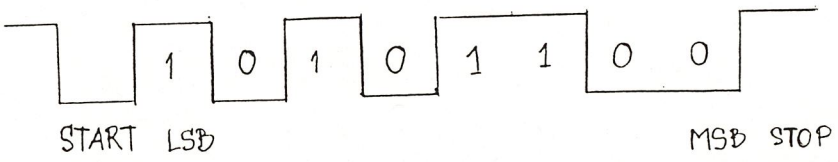


ในการ mod จะใช้ความถี่ประมาณ 20KHz ซึ่งมาจาก RC OSCILLATOR ความถี่ที่ได้เท่ากับ

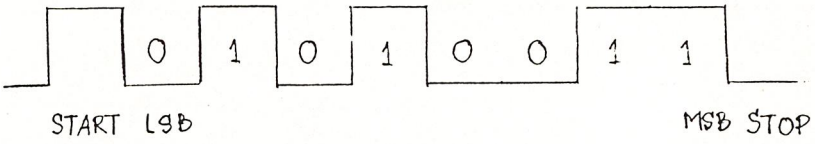
$$F_o = \frac{1}{2C[0.41R_p + 0.7R_1]} \quad ; \quad R_p = \frac{R1*R2}{R1 + R2}$$

สัญญาณที่จุดต่างๆแสดงได้ดังนี้

A สัญญาณที่ออกจาก TXD สมมุติว่าเป็น 35H

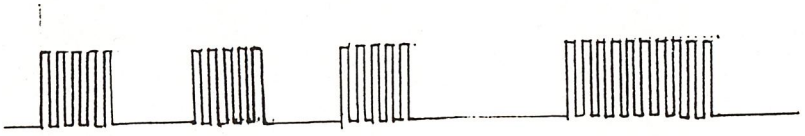


B ข้อมูลที่ผ่าน NOT GATE



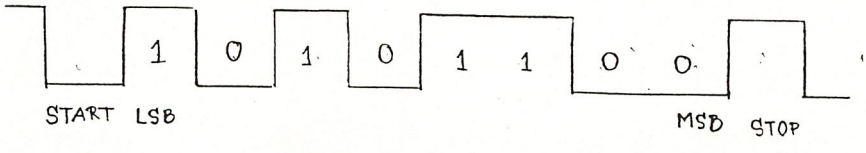
C carrier

D สัญญาณที่ผ่านการ MOD แล้ว



E สัญญาณที่ส่งออกจากตัวส่ง

F สัญญาณที่ได้จากตัวรับ



โปรแกรมภาครับ

_จะเก็บข้อมูลที่รับได้ไว้ที่ file in_fi.dat

_function port_init() เป็น function เซ็ทสถานะเริ่มต้นของ port

_function reprot(); เป็น function ที่รับข้อมูล 1 ไบท์

_function rec_dat(); จะนำข้อมูลที่รับได้ไปเก็บไว้ที่ไฟล์ in_fi.dat

-function decode(); จะทำการเปิดไฟล์ in_fi.dat แล้วตีความ แสดงออกทางหน้าจอ

_function check_stat(); จะตรวจสอบสถานะของพอร์ท

```
#include <dos.h>
#include <stdio.h>
#define  PORT  0
main()
{
   port_init(PORT,195);
   clrscr();
   gotoxy(22,11);
   printf("....WAITING FOR DATA ....");
   rec_dat();
   clrscr();
   gotoxy(4,4);
   printf("--------------------------------------------------");
   gotoxy(13,5);
   printf("FOOD & DRINK     UNIT    PRICE/UNIT     TOTAL PRICE");
   gotoxy(4,6);
   printf("--------------------------------------------------");
   decode();
  }
```

```
/* This function use to receive data */
 rec_dat()
{
    FILE *fp, *fopen();
    int  ch,count;
    char  rec_f_name[] = "in_fi.dat";
    fp = fopen(rec_f_name,"wb");
    if (fp == NULL ){
       printf("ERROR OPEN FILE");
       exit(1);
       }
    else {
        ch = rport(PORT);
       for(count = 0;((count<=100)&(ch!= 255));count++)
       {
        ch = rport(PORT);
        putc(ch,fp);
        if(ferror(fp)){
           printf("error writing file");
           exit(1);
         }
        }
    ch = '\n';
    putc(ch,fp);
    }
        fclose(fp);
    }
```

```
/* This function use for provide */
/* begining status of port */
port_init(port,code)

int port;
unsigned char code;
{

    union  REGS r;

    r.x.dx = port;      /* number of port */
    r.h.ah = 0;         /* call func_0 provide status */
    r.h.al = code;      /* code for set initial status */
}

/* This function use to check status of port */
check_stat(port)
int port;  /* number of port */
{
   union REGS r;
   r.x.dx  = port;     /* number of port */
   r.x.ah  = 3;        /* func_3 */
   int86(0x14,&r,&r);
   return r.x.ax;
}
```

```
/* This function use for read character form serial port */

rport(port)
int port;                    /* number of port */
{
 union REGS r;

 /* wait until character income */
 while(!(check_stat(port)&256))
      if(kbhit())  {         /* cancel if key press */
           getch();
           exit(1);
      }

 r.x.dx = port;              /* define number of port */
 r.h.ah = 2;                 /* func_2 that read character */
 int86(0x14,&r,&r);
 if(r.h.ah & 128)
      printf("read error detected in serial port\n");
 return r.h.al;
 }
```

```
decode()
{
static char *menu[] =
{ "PIZZA",
  "SALAD",
  "STEAK",
  "HOT DOG",
  "HAMBERGER",
  "CHICKEN PIE",
  "APPLE PIE",
  "ONION SOUP",
  "DONUT",
  "HOT COFFEE",
  "ICE COFFEE",
  "ICE TEA",
  "COCA COLA",
  "PUNCH",
  "JUICE",
  "MILK",
  "BEER"};

int price[18]={75,30,70,15,22,18,15,25,8,12,16,16,13,13,12,12,30};
unsigned int data[100],reprice[100],vol[100];
int i,b,k,y,d,m,n,z,j;
FILE *fp1,*fopen();
char fi_name[] = "in_fi.dat";
{
 fp1 = fopen(fi_name,"rb");
 if(fp1==NULL){
      printf("error open file");
      exit(1);
 }
```

```
i=0;
while((data[i]=getc(fp1))!='\n')
     i++;
     data[i]='\0';
     b=i-1;
     gotoxy(27,2);
     printf("TABLE.....:%x",data[0]);
     d=1;
     y=7;
     m=0;
     for(i=2;i<b;i=i+2){
        if (data[i]>=10) {
        data[i] = data[i] - 6;
        }
        gotoxy(16,y);
        printf("%s",menu[data[i]]);
        gotoxy(5,y);
        printf("%d",d);
        gotoxy(42,y);
        printf("%5d",price[data[i]]);
        reprice[m]=price[data[i]];
        y++;
        m++;
        d++;
     }
        reprice[m]='\0';
        y=7;
        n=0;
```

```
  for(i=3;i<=b;i=i+2){
     gotoxy(32,y);
     printf("%x",data[i]);
     vol[n]=data[i];
     y++;
     n++;
  }
  z=0;
  m=0;
  j=0;
  y=7;
for(i=2;i<b;i=i+2)
{
if(vol[z]>=10){
  k=vol[z]&0xf0;
  k=k>>4;
  k=k*6;
  vol[z] =  vol[z]-k;
  reprice[m]=vol[z]*reprice[m];
}
else
  reprice[m]=vol[z]*reprice[m];
  gotoxy(63,y);
  printf("%5d",reprice[m]);
  j=j+reprice[m];
  y++;
  z++;
  m++;
}
```

```
gotoxy(58,y);
printf("-----------------");
y++;
gotoxy(43,y);
printf("TOTAL                    %5d",j);
y++;
gotoxy(58,y);
printf("=================");
}
fclose(fp1);
getchar();
}
```

โปรแกรมภาคส่ง

```
000000                          ORG     0000H
000000                          CPU     "8051.TBL"
                                ;
00008000    =                   PDAT:   EQU     8000H
00008001    =                   PSIG:   EQU     8001H
00008003    =                   PCON:   EQU     8003H
0000002F    =                   COUNT1: EQU     2FH
0000002E    =                   COUNT2: EQU     2EH
00000083    =                   DPH:    EQU     83H
00000082    =                   DPL:    EQU     82H
000000E7    =                   ACC.7:  EQU     0E7H
00000098    =                   SCON:   EQU     98H
00000099    =                   SCON.1: EQU     99H
00000099    =                   SBUF:   EQU     99H
0000008D    =                   TH1:    EQU     8DH
0000008B    =                   TL1:    EQU     8BH
00000089    =                   TMOD:   EQU     89H
00000088    =                   TCON:   EQU     88H
000000A8    =                   IE:     EQU     0A8H
00000087    =                   PCON.7: EQU     87H
000000E0    =                   A:      EQU     0E0H
000000E2    =                   ACC.2:  EQU     0E2H
00000081    =                   SP:     EQU     81H
00000090    =                   P1:     EQU     90H
000000B4    =                   P3.4:   EQU     0B4H
00000000    =                   R0:     EQU     00H
00000007    =                   R7:     EQU     07H
00000006    =                   R6:     EQU     06H
000000F0    =                   B:      EQU     0F0H
                                ;
```

```
000000 02002E                       LJMP    MAIN
                                    ;
                                    ;This is a serial interrupt
000023                              ORG     0023H
                                    ;
000023 109903                       JBC     SCON.1,TRANS
000026 02002D                       LJMP    EXIT
000029 E0               TRANS:      MOVX    A,@DPTR
00002A F599                         MOV     SBUF,A
00002C A3                           INC     DPTR
00002D 32               EXIT:       RETI
                                    ;
                                    ;Main program begin here
                                    ;
00002E 758130           MAIN:       MOV     SP,#30H
000031 751700                       MOV     17H,#00H
000034 751800                       MOV     18H,#00H
000037 120130                       LCALL   INITLCD
00003A 9004F4                       MOV     DPTR,#TAB
00003D 120190                       LCALL   DISP
000040 1201DA                       LCALL   ENTDAT
000043 900000                       MOV     DPTR,#0000H
000046 F0                           MOVX    @DPTR,A
000047 752F01                       MOV     COUNT1,#01H
00004A 752E01                       MOV     COUNT2,#01H
00004D 7F00                         MOV     R7,#00H
00004F 7E02                         MOV     R6,#02H
000051 8F18                         MOV     18H,R7
000053 8E17                         MOV     17H,R6
000055 120130                       LCALL   INITLCD
000058 1200F1           ORI:        LCALL   SCAN
00005B 120256                       LCALL   CHECK
00005E BF0003                       CJNE    R7,#00H,CHUP
```

```
000061 02006A                    LJMP    NEXTL
000064 B80CF1          CHUP:     CJNE    RO,#0CH,ORI
000067 020098                    LJMP    COM1
00006A 120268          NEXTL:    LCALL   ONELINE
00006D E57F                      MOV     A,7FH
00006F B4000F                    CJNE    A,#00H,COMBACK
000072 E57E                      MOV     A,7EH
000074 B4000A                    CJNE    A,#00H,COMBACK
000077 120130                    LCALL   INITLCD
00007A 052F                      INC     COUNT1
00007C 052E                      INC     COUNT2
00007E 1202BF                    LCALL   INCADD
000081 1200F1          COMBACK:  LCALL   SCAN
000084 120256                    LCALL   CHECK
000087 BF0008                    CJNE    R7,#00H,COMM
00008A E52F                      MOV     A,COUNT1
00008C B52EF2                    CJNE    A,COUNT2,COMBACK
00008F 02006A                    LJMP    NEXTL
000092 B80A03          COMM:     CJNE    RO,#0AH,COM1
000095 020081                    LJMP    COMBACK
000098 B80C13          COM1:     CJNE    RO,#0CH,COM2
00009B E52E                      MOV     A,COUNT2
00009D B40003                    CJNE    A,#00H,CNTI1
0000A0 020081                    LJMP    COMBACK
0000A3 120130          CNTI1:    LCALL   INITLCD
0000A6 1202EF                    LCALL   UP
0000A9 152E                      DEC     COUNT2
0000AB 020081                    LJMP    COMBACK
0000AE B80D23          COM2:     CJNE    RO,#0DH,COM3
0000B1 C3                        CLR     C
0000B2 E52F                      MOV     A,COUNT1
0000B4 952E                      SUBB    A,COUNT2
0000B6 40C9                      JC      COMBACK
```

```
0000B8 C3                       CLR     C
0000B9 B4010B                   CJNE    A,#01H,CNTI2
0000BC 1202BF                   LCALL   INCADD
0000BF 120130                   LCALL   INITLCD
0000C2 052E                     INC     COUNT2
0000C4 020081                   LJMP    COMBACK
0000C7 40B8             CNTI2:  JC      COMBACK
0000C9 120130                   LCALL   INITLCD
0000CC 120305                   LCALL   DOWN
0000CF 052E                     INC     COUNT2
0000D1 020081                   LJMP    COMBACK
                                ;
0000D4 B80B0E           COM3:   CJNE    RO,#0BH,COM4
0000D7 E52F                     MOV     A,COUNT1
0000D9 B52E03                   CJNE    A,COUNT2,CR
0000DC 020081                   LJMP    COMBACK
0000DF 12030C           CR:     LCALL   CLEAR
0000E2 020081                   LJMP    COMBACK
0000E5 B80E99           COM4:   CJNE    RO,#0EH,COMBACK
0000E8 1204C3                   LCALL   TOTA
0000EB 12049C                   LCALL   SEND
0000EE 020081                   LJMP    COMBACK
                                ;
                                ;
0000F1 7FEE             SCAN:   MOV     R7,#0EEH
0000F3 8F90             NEWR:   MOV     P1,R7
0000F5 D2B4                     SETB    P3.4
0000F7 C2B4                     CLR     P3.4
0000F9 7590FF                   MOV     P1,#0FFH
0000FC E590                     MOV     A,P1
0000FE C4                       SWAP    A
0000FF FA                       MOV     R2,A
```

```
000100 B4FF06                    CJNE    A,#0FFH,PRESS
000103 EF                        MOV     A,R7
000104 23                        RL      A
000105 FF                        MOV     R7,A
000106 0200F3                    LJMP    NEWR
000109 120180           PRESS:   LCALL   DELAY
00010C 7590FF                    MOV     P1,#0FFH
00010F E590                      MOV     A,P1
000111 B4FFF5                    CJNE    A,#0FFH,PRESS
000114 EF               DECODE:  MOV     A,R7
000115 7B04                      MOV     R3,#04H
000117 7C00                      MOV     R4,#00H
000119 D3               DEROW:   SETB    C
00011A 13                        RRC     A
00011B 5006                      JNC     DECOL
00011D 0C                        INC     R4
00011E 0C                        INC     R4
00011F 0C                        INC     R4
000120 0C                        INC     R4
000121 DBF6                      DJNZ    R3,DEROW
000123 EA               DECOL:   MOV     A,R2
000124 7B04                      MOV     R3,#04H
000126 D3               DECOL1:  SETB    C
000127 13                        RRC     A
000128 5003                      JNC     FINI
00012A 0C                        INC     R4
00012B DBF9                      DJNZ    R3,DECOL1
00012D A804             FINI:    MOV     R0,04H
00012F 22                        RET
                                 ;
000130 7480             INITLCD: MOV     A,#80H
000132 908003                    MOV     DPTR,#PCON
000135 F0                        MOVX    @DPTR,A
```

```
000136 7400
000138 908001
00013B F0
00013C 7438
00013E 908000
000141 F0
000142 7400
000144 12016F
000147 120180

00014A 740F
00014C 908000
00014F F0
000150 7400
000152 12016F

000155 7406
000157 908000
00015A F0
00015B 7400
00015D 12016F

000160 7401
000162 908000
000165 F0
000166 7400
000168 12016F
00016B 120180
00016E 22
```

```
MOV     A,#00H          ;FU
MOV     DPTR,#PSIG
MOVX    @DPTR,A
MOV     A,#38H
MOV     DPTR,#PDAT
MOVX    @DPTR,A
MOV     A,#00H
LCALL   EPLUSE
LCALL   DELAY
;
MOV     A,#00001111B    ;DI
MOV     DPTR,#PDAT
MOVX    @DPTR,A
MOV     A,#00H
LCALL   EPLUSE
;
MOV     A,#00000110B    ;EN
MOV     DPTR,#PDAT
MOVX    @DPTR,A
MOV     A,#00H
LCALL   EPLUSE
;
MOV     A,#00000001B    ;CL
MOV     DPTR,#PDAT
MOVX    @DPTR,A
MOV     A,#00H
LCALL   EPLUSE
LCALL   DELAY
RET
;
```

```
00016F D2E2          EPLUSE:   SETB    ACC.2
000171 908001                  MOV     DPTR,#PSIG
000174 F0                      MOVX    @DPTR,A
000175 7E00                    MOV     R6,#00H
000177 DEFE          EP1:      DJNZ    R6,EP1
000179 C2E2                    CLR     ACC.2
00017B 908001                  MOV     DPTR,#PSIG
00017E F0                      MOVX    @DPTR,A
00017F 22                      RET
                               ;
000180 750964        .DELAY:   MOV     09H,#100D
000183 75080A        DEL1:     MOV     08H,#10D
000186 00            DEL2:     NOP
000187 00                      NOP
000188 00                      NOP
000189 D508FA                  DJNZ    08H,DEL2
00018C D509F4                  DJNZ    09H,DEL1
00018F 22                      RET
                               ;
000190 7402          DISP:     MOV     A,#02H
000192 7C02                    MOV     R4,#02D
000194 7B06                    MOV     R3,#06D
000196 C083          NEXT:     PUSH    DPH
000198 C082                    PUSH    DPL
00019A 1201BC                  LCALL   GOTO
00019D D082                    POP     DPL
00019F D083                    POP     DPH
0001A1 7400          REOUT:    MOV      A,#00H
0001A3 93                      MOVC    A,@A+DPTR
0001A4 C083                    PUSH    DPH
0001A6 C082                    PUSH    DPL
0001A8 1201CC                  LCALL   WRBYTE
0001AB D082                    POP     DPL
```

```
0001AD D083                          POP     DPH
0001AF A3                            INC     DPTR
0001B0 DBEF                          DJNZ    R3,REOUT
0001B2 7440                          MOV     A,#40H
0001B4 DC01                          DJNZ    R4,NEXTL1
0001B6 22                            RET
0001B7 7B06                NEXTL1:   MOV     R3,#06H
0001B9 020196                        LJMP    NEXT
                                     ;
0001BC D2E7                GOTO:     SETB    ACC.7
0001BE 908000                        MOV     DPTR,#PDAT
0001C1 F0                            MOVX    @DPTR,A
0001C2 7400                          MOV     A,#00H
0001C4 908001                        MOV     DPTR,#PSIG
0001C7 F0                            MOVX    @DPTR,A
0001C8 12016F                        LCALL   EPLUSE
0001CB 22                            RET
0001CC 908000              WRBYTE:   MOV     DPTR,#PDAT
0001CF F0                            MOVX    @DPTR,A
0001D0 7401                          MOV     A,#01H
0001D2 908001                        MOV     DPTR,#PSIG
0001D5 F0                            MOVX    @DPTR,A
0001D6 12016F                        LCALL   EPLUSE
0001D9 22                            RET
                                     ;
0001DA 1200F1              ENTDAT:   LCALL   SCAN
0001DD 120256              ENTDAT1:  LCALL   CHECK
0001E0 757E00                        MOV     7EH,#00H
0001E3 BF00F4                        CJNE    R7,#00H,ENTDAT
0001E6 8811                          MOV     11H,R0
0001E8 120261                        LCALL   DISPC
```

```
0001EB 1200F1          WRONG:   LCALL   SCAN
0001EE 8810                     MOV     10H,R0
0001F0 B80A06                   CJNE    R0,#0AH,DEL
0001F3 85112D                   MOV     2DH,11H
0001F6 02024F                   LJMP    ENTER
0001F9 B80B3B          DEL:     CJNE    R0,#0BH,CHAR
0001FC 120130          ADEL:    LCALL   INITLCD
0001FF E517                     MOV     A,17H
000201 2518                     ADD     A,18H
000203 B40009                   CJNE    A,#00H,NOTA
000206 9004F4                   MOV     DPTR,#TAB
000209 120190                   LCALL   DISP
00020C 0201DA                   LJMP    ENTDAT
00020F 851883          NOTA:    MOV     DPH,18H
000212 851782                   MOV     DPL,17H
000215 E0                       MOVX    A,@DPTR
000216 F515                     MOV     15H,A
000218 54F0                     ANL     A,#0F0H
00021A C4                       SWAP    A
00021B F8                       MOV     R0,A
00021C 120261                   LCALL   DISPC
00021F E515                     MOV     A,15H
000221 540F                     ANL     A,#0FH
000223 F8                       MOV     R0,A
000224 120261                   LCALL   DISPC
000227 1202CB                   LCALL   DISPS
00022A 1200F1                   LCALL   SCAN
00022D B80BAD                   CJNE    R0,#0BH,ENTDAT1
000230 757EFF                   MOV     7EH,#0FFH
000233 120130                   LCALL   INITLCD
000236 22                       RET
```

```
000237 120256          CHAR:     LCALL   CHECK
00023A BF00AE                    CJNE    R7,#00H,WRONG
00023D 120261                    LCALL   DISPC
000240 E511                      MOV     A,11H
000242 C4                        SWAP    A
000243 2510                      ADD     A,10H
000245 F52D                      MOV     2DH,A
000247 1200F1          LOAD:     LCALL   SCAN
00024A B80A03                    CJNE    R0,#0AH,CD
00024D E52D                      MOV     A,2DH
00024F 22              ENTER:    RET
000250 B80BF4          CD:       CJNE    R0,#0BH,LOAD
000253 0201FC                    LJMP    ADEL
                                 ;
000256 7FFF            CHECK:    MOV     R7,#0FFH
000258 C3                        CLR     C
000259 E8                        MOV     A,R0
00025A 940A                      SUBB    A,#0AH
00025C 5002                      JNC     C0
00025E 7F00                      MOV     R7,#00H
000260 22              C0:       RET
                                 ;
000261 E8              DISPC:    MOV     A,R0
000262 2430                      ADD     A,#30H
000264 1201CC                    LCALL   WRBYTE
000267 22                        RET
                                 ;
000268 E8              ONELINE:  MOV     A,R0
000269 C4                        SWAP    A
00026A F514                      MOV     14H,A
00026C 120261                    LCALL   DISPC
00026F 757F00                    MOV     7FH,#00H
                                 ;
```

```
000272 1200F1          BEG1:    LCALL   SCAN
000275 120256                   LCALL   CHECK
000278 BF001A                   CJNE    R7,#00H,CHDEL
00027B E514                     MOV     A,14H
00027D 28                       ADD     A,R0
00027E F514                     MOV     14H,A
000280 851415                   MOV     15H,14H
000283 120261                   LCALL   DISPC
                                ;
000286 851883                   MOV     DPH,18H
000289 851782                   MOV     DPL,17H
00028C E514                     MOV     A,14H
00028E F0                       MOVX    @DPTR,A
00028F 1202CB                   LCALL   DISPS
000292 02029F                   LJMP    NEX
                                ;
000295 B80BDA          CHDEL:   CJNE    R0,#0BH,BEG1
000298 120130                   LCALL   INITLCD
00029B 757FFF                   MOV     7FH,#0FFH
00029E 22                       RET
                                ;
00029F 1200F1          NEX:     LCALL   SCAN
0002A2 B80B07                   CJNE    R0,#0BH,NODEL
0002A5 120130                   LCALL   INITLCD
0002A8 757FFF                   MOV     7FH,#0FFH
0002AB 22                       RET
0002AC 1201DD          NODEL:   LCALL   ENTDAT1
0002AF E57E                     MOV     A,7EH
0002B1 B4000A                   CJNE    A,#00H,BACK
0002B4 E52D                     MOV     A,2DH
0002B6 851883                   MOV     DPH,18H
0002B9 851782                   MOV     DPL,17H
0002BC A3                       INC     DPTR
```

```
0002BD F0                       MOVX    @DPTR,A
0002BE 22           BACK:       RET
                                ;
0002BF C3           INCADD:     CLR     C
0002C0 0517                     INC     17H
0002C2 0517                     INC     17H
0002C4 E518                     MOV     A,18H
0002C6 3400                     ADDC    A,#00H
0002C8 F518                     MOV     18H,A
0002CA 22                       RET
                                ;
0002CB E515         DISPS:      MOV     A,15H
0002CD 75F00C                   MOV     B,#0CH
0002D0 C3                       CLR     C
0002D1 940A                     SUBB    A,#0AH
0002D3 5005                     JNC      CONV
0002D5 E515                     MOV     A,15H
0002D7 0202DF                   LJMP    MULTI
0002DA E515         CONV:       MOV     A,15H
0002DC C3                       CLR     C
0002DD 9406                     SUBB    A,#06H
0002DF A4           MULTI:      MUL     AB
0002E0 C3                       CLR     C
0002E1 2400                     ADD     A,#00H
0002E3 F582                     MOV     DPL,A
0002E5 E5F0                     MOV     A,B
0002E7 2415                     ADD     A,#15H
0002E9 F583                     MOV     DPH,A
0002EB 120190                   LCALL   DISP
0002EE 22                       RET
                                ;
```

```
0002EF 12041C          UP:      LCALL   DECADD
0002F2 E52E                     MOV     A,COUNT2
0002F4 B4010A                   CJNE    A,#01H,UPE
0002F7 9004F4                   MOV     DPTR,#TAB
0002FA 120190                   LCALL   DISP
0002FD 120428                   LCALL   SHOW1
000300 22                       RET
000301 12043E          UPE:     LCALL   UPDWE
000304 22                       RET
                                ;
000305 1202BF          DOWN:    LCALL   INCADD
000308 12043E                   LCALL   UPDWE
00030B 22                       RET
                                ;
00030C 120130          CLEAR:   LCALL   INITLCD
00030F E518                     MOV     A,18H
000311 2517                     ADD     A,17H
000313 B40009                   CJNE    A,#00H,NONTA
000316 9004F4                   MOV     DPTR,#TAB
000319 120190                   LCALL   DISP
00031C 020328                   LJMP    CHDAT
00031F 120428          NONTA:   LCALL   SHOW1
000322 851915                   MOV     15H,19H
000325 1202CB                   LCALL   DISPS
000328 1200F1          CHDAT:   LCALL   SCAN
00032B 120256                   LCALL   CHECK
00032E BF003C                   CJNE    R7,#00H,CON1
000331 8811                     MOV     11H,R0
000333 120261                   LCALL   DISPC
000336 1200F1          WRONG1:  LCALL   SCAN
000339 8810                     MOV     10H,R0
00033B B80A05                   CJNE    R0,#0AH,CHAR1
00033E E511                     MOV     A,11H
```

```
000340 020351                    LJMP    ENTER1
000343 120256          CHAR1:    LCALL   CHECK
000346 BF00ED                    CJNE    R7,#00H,WRONG1
000349 120261                    LCALL   DISPC
00034C E511                      MOV     A,11H
00034E C4                        SWAP    A
00034F 2510                      ADD     A,10H
000351 A92E            ENTER1:   MOV     R1,COUNT2
000353 B90009                    CJNE    R1,#00H,INCA
000356 851883                    MOV     DPH,18H
000359 851782                    MOV     DPL,17H
00035C 020366                    LJMP    OVE
00035F 851883          INCA:     MOV     DPH,18H
000362 851782                    MOV     DPL,17H
000365 A3                        INC     DPTR
000366 F0              OVE:      MOVX    @DPTR,A
000367 7420                      MOV     A,#20H
000369 1201CC                    LCALL   WRBYTE
00036C 22                        RET
00036D B80B3D          CON1:     CJNE    R0,#0BH,OUT1
000370 E517                      MOV     A,17H
000372 2518                      ADD     A,18H
000374 B40003                    CJNE    A,#00H,NET
000377 020328                    LJMP    CHDAT
00037A 85171C          NET:      MOV     1CH,17H
00037D 85181D                    MOV     1DH,18H
000380 752302                    MOV     23H,#02H
000383 120130          REFLA:    LCALL   INITLCD
000386 120477                    LCALL   OFF
000389 120467                    LCALL   LDELAY
00038C 120130                    LCALL   INITLCD
00038F 12043E                    LCALL   UPDWE
000392 120467                    LCALL   LDELAY
```

```
000395 D523EB                       DJNZ    23H,REFLA
000398 120488                       LCALL   SHODEL
00039B 120467                       LCALL   LDELAY
00039E 120467                       LCALL   LDELAY
0003A1 120467                       LCALL   LDELAY
0003A4 851C17                       MOV     17H,1CH
0003A7 851D18                       MOV     18H,1DH
0003AA 1203AE                       LCALL   CLRL
0003AD 22                   OUT1:   RET
                                    ;
0003AE E52F                 CLRL:   MOV     A,COUNT1
0003B0 C3                           CLR     C
0003B1 952E                         SUBB    A,COUNT2
0003B3 14                           DEC     A
0003B4 F51B                         MOV     1BH,A
0003B6 F51E                         MOV     1EH,A
0003B8 85181D                       MOV     1DH,18H
0003BB 85171C                       MOV     1CH,17H
0003BE B40003                       CJNE    A,#00H,NLDAT
0003C1 0203EE                       LJMP    LDAT
0003C4 1203F4               NLDAT:  LCALL   DUMP
                                    ;
0003C7 C3                           CLR     C
0003C8 851C17                       MOV     17H,1CH
0003CB 0517                         INC     17H
0003CD E51D                         MOV     A,1DH
0003CF 3400                         ADDC    A,#00H
0003D1 F518                         MOV     18H,A
0003D3 851E1B                       MOV     1BH,1EH
0003D6 1203F4                       LCALL   DUMP
0003D9 152F                         DEC     COUNT1
0003DB 851C17                       MOV     17H,1CH
0003DE 851D18                       MOV     18H,1DH
```

```
0003E1 120130                    LCALL   INITLCD
0003E4 12043E                    LCALL   UPDWE
0003E7 851C17                    MOV     17H,1CH
0003EA 851D18                    MOV     18H,1DH
0003ED 22                        RET
0003EE 120130           LDAT:    LCALL   INITLCD
0003F1 152F                      DEC     COUNT1
0003F3 22                        RET
                                 ;
0003F4 C3               DUMP:    CLR     C
0003F5 AE17                      MOV     R6,17H
0003F7 0E                        INC     R6
0003F8 0E                        INC     R6
0003F9 E518                      MOV     A,18H
0003FB 3400                      ADDC    A,#00H
0003FD FF                        MOV     R7,A
0003FE 8F83                      MOV     DPH,R7
000400 8E82                      MOV     DPL,R6
000402 E0                        MOVX    A,@DPTR
000403 C083                      PUSH    DPH
000405 C082                      PUSH    DPL
000407 851883                    MOV     DPH,18H
00040A 851782                    MOV     DPL,17H
00040D F0                        MOVX    @DPTR,A
00040E D082                      POP     DPL
000410 D083                      POP     DPH
000412 858217                    MOV     17H,DPL
000415 858318                    MOV     18H,DPH
000418 D51BD9                    DJNZ    1BH,DUMP
00041B 22                        RET
                                 ;
00041C C3               DECADD:  CLR     C
00041D 1517                      DEC     17H
```

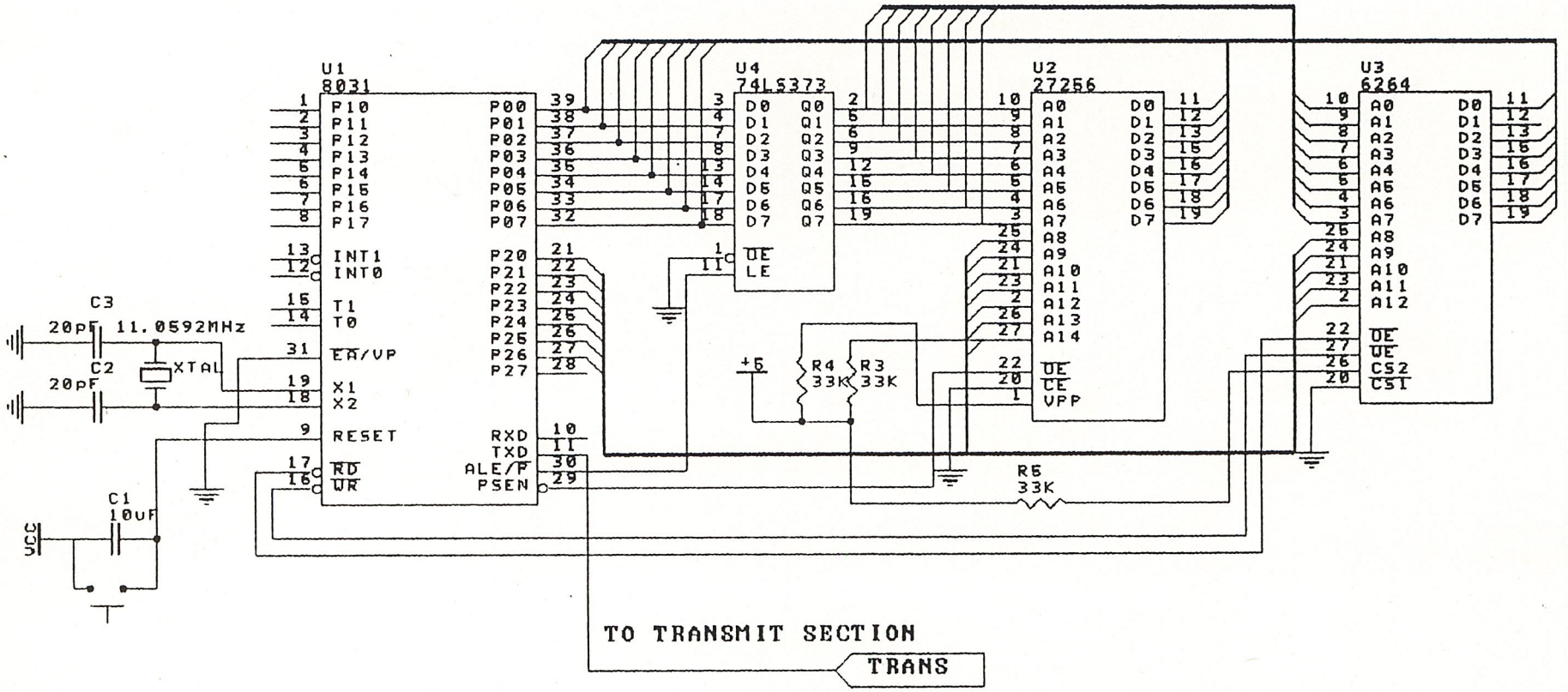
```
00045D 540F          SWA:    ANL     A,#0FH
00045F F51A                  MOV     1AH,A
000461 E519                  MOV     A,19H
000463 54F0                  ANL     A,#0F0H
000465 C4                    SWAP    A
000466 22                    RET
                             ;
000467 7521FF        LDELAY: MOV     21H,#0FFH
00046A 7522EA        LDE1:   MOV     22H,#0EAH
00046D 00            LDE2:   NOP
00046E 00                    NOP
00046F 00                    NOP
000470 D522FA                DJNZ    22H,LDE2
000473 D521F4                DJNZ    21H,LDE1
000476 22                    RET
000477 900500        OFF:    MOV     DPTR,#TAB1
00047A 120190                LCALL   DISP
00047D 7420                  MOV     A,#20H
00047F 1201CC                LCALL   WRBYTE
000482 7420                  MOV     A,#20H
000484 1201CC                LCALL   WRBYTE
000487 22                    RET
                             ;
000488 120130        SHODEL: LCALL   INITLCD
00048B 90050C                MOV     DPTR,#TAB2
00048E 120190                LCALL   DISP
000491 7420                  MOV     A,#20H
000493 1201CC                LCALL   WRBYTE
000496 7420                  MOV     A,#20H
000498 1201CC                LCALL   WRBYTE
00049B 22                    RET
                             ;
```

```
00049C E52F          SEND:    MOV     A,COUNT1
00049E 252F                   ADD     A,COUNT1      ;
0004A0 FB                     MOV     R3,A          ;
0004A1 758300                 MOV     DPH,#00H
0004A4 F582                   MOV     DPL,A
0004A6 74FF                   MOV     A,#0FFH
0004A8 F0                     MOVX    @DPTR,A
0004A9 900000                 MOV     DPTR,#0000H
0004AC 53877F                 ANL     PCON.7,#7FH
0004AF 758DFA                 MOV     TH1,#0FAH     ;
0004B2 759850                 MOV     SCON,#50H
0004B5 758920                 MOV     TMOD,#20H     ;
0004B8 758840                 MOV     TCON,#40H     ;
0004BB 75A890                 MOV     IE,#90H       ;
0004BE D299                   SETB    SCON.1
0004C0 0204C0        LLOOP1:  LJMP    LLOOP1
                              ;
0004C3 120130        TOTA:    LCALL   INITLCD
0004C6 900518                 MOV     DPTR,#TAB3
0004C9 120190                 LCALL   DISP
0004CC AA2F                   MOV     R2,COUNT1
0004CE 1A                     DEC     R2
0004CF EA                     MOV     A,R2
0004D0 C3                     CLR     C
0004D1 940A                   SUBB    A,#0AH
0004D3 500B                   JNC     A6
0004D5 8A00                   MOV     R0,R2
0004D7 120261                 LCALL   DISPC
0004DA 7420                   MOV     A,#20H
0004DC 1201CC                 LCALL   WRBYTE
0004DF 22                     RET
```

```
0004E0 C3                   A6:     CLR     C
0004E1 EA                           MOV     A,R2
0004E2 2406                         ADD     A,#06H
0004E4 FA                           MOV     R2,A
0004E5 C4                           SWAP    A
0004E6 540F                         ANL     A,#0FH
0004E8 F8                           MOV     RO,A
0004E9 120261                       LCALL   DISPC
0004EC EA                           MOV     A,R2
0004ED 540F                         ANL     A,#0FH
0004EF F8                           MOV     RO,A
0004F0 120261                       LCALL   DISPC
0004F3 22                           RET
                                    ;
                                    ;
0004F4 2020544142 4C45      TAB:    DFB "  TABLE    :"
                                    ;
000500 20202020202020       TAB1:   DFB "             "
                                    ;
00050C 202044454C4554       TAB2:   DFB "  DELETE !  "
                                    ;
000518 544F54414C204F       TAB3:   DFB "TOTAL ORDER:"
                                    ;
001500                              ORG     1500H
001500 2050495A5A4120               DFB " PIZZA     :"
00150C 2053414C414420               DFB " SALAD     :"
001518 20535445414B20               DFB " STEAK     :"
001524 20484F54444F47               DFB " HOTDOG    :"
001530 2048414D424552               DFB " HAMBERGER :"
00153C 434849434B454E               DFB "CHICKEN PIE:"
001548 204150504C4520               DFB " APPLE PIE :"
001554 204F4E494F4E20               DFB " ONION SOUP:"
001560 20444F4E555420               DFB " DONUT     :"
```

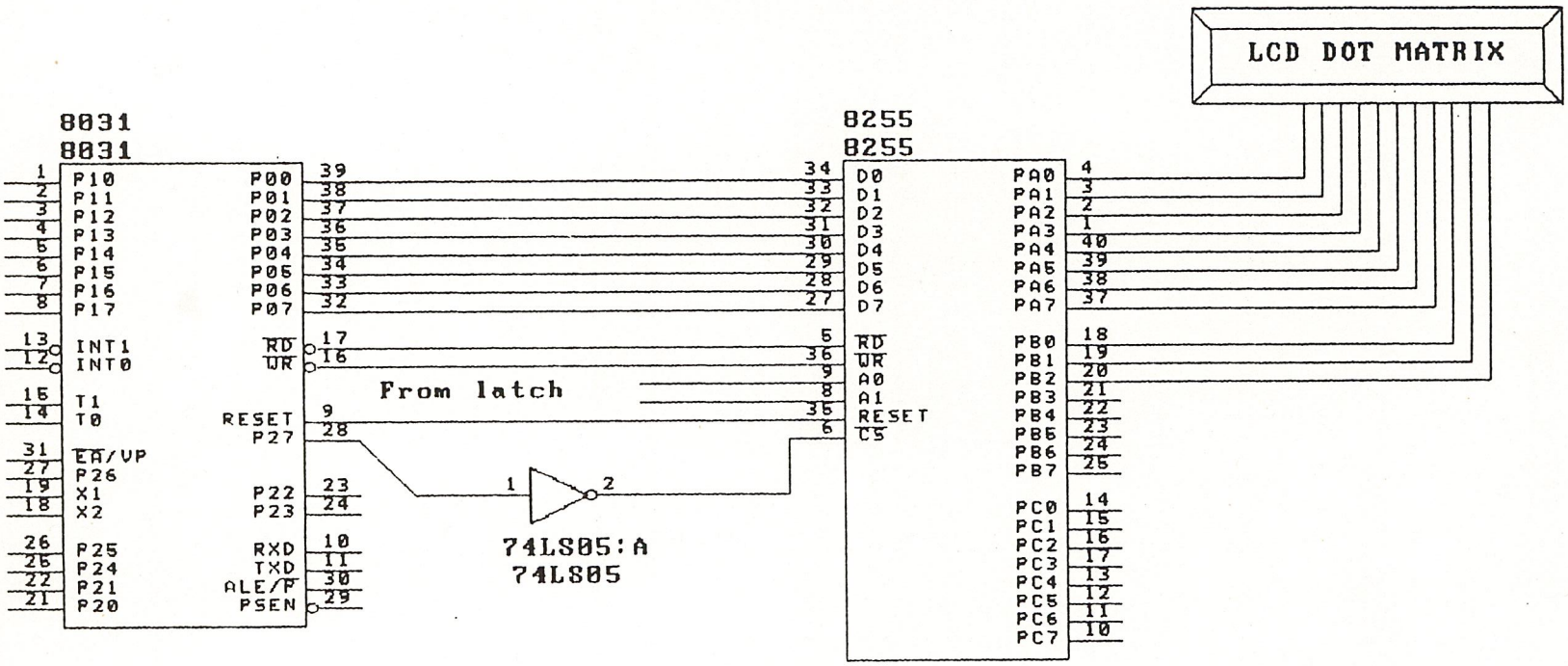
```
001560 20484F5420434F                  DFB " HOT COFFEE:"
001578 2049434520434F                  DFB " ICE COFFEE:"
001584 20494345205445                  DFB " ICE TEA   :"
001590 20434F43412043                  DFB " COCA COLA :"
00159C 2050554E434820                  DFB " PUNCH     :"
0015A8 204A5549434520                  DFB " JUICE     :"
0015B4 204D494C4B2020                  DFB " MILK      :"
0015C0 20424545522020                  DFB " BEER      :"
000000                                 END

000000E0  A             000004E0  A6            000000E2  ACC.
000000E7  ACC.7         000001FC  ADEL          000000F0  B
000002BE  BACK          00000272  BEG1          00000250  CD
00000237  CHAR          00000343  CHAR1         00000328  CHDA
00000295  CHDEL         00000256  CHECK         00000064  CHUP
0000030C  CLEAR         000003AE  CLRL          000000A3  CNTI
000000C7  CNTI2         00000260  CO            00000098  COM1
000000AE  COM2          000000D4  COM3          000000E5  COM4
00000081  COMBACK       00000092  COMM          0000036D  CON1
000002DA  CONV          0000002F  COUNT1        0000002E  COUN
000000DF  CR            0000041C  DECADD        00000114  DECO
00000123  DECOL         00000126  DECOL1        000001F9  DEL
00000183  DEL1          00000186  DEL2          00000180  DELA
00000119  DEROW         00000190  DISP          00000261  DISP
000002CB  DISPS         00000305  DOWN          00000083  DPH
00000082  DPL           000003F4  DUMP          000001DA  ENTD
000001DD  ENTDAT1       0000024F  ENTER         00000351  ENTE
00000177  EP1           0000016F  EPLUSE        0000002D  EXIT
0000012D  FINI          000001BC  GOTO          000000A8  IE
0000035F  INCA          000002BF  INCADD        00000130  INIT
000003EE  LDAT          0000046A  LDE1          0000046D  LDE2
00000467  LDELAY        000004C0  LLOOP1        00000247  LOAD
0000002E  MAIN          000002DF  MULTI         0000037A  NET
000000F3  NEWR          0000029F  NEX           00000196  NEXT
```

```
0000006A  NEXTL
000002AC  NODEL
00000477  OFF
000003AD  OUT1
000000B4  P3.4
00008000  PDAT
00000000  R0
00000383  REFLA
000000F1  SCAN
0000049C  SEND
00000081  SP
00000500  TAB1
00000088  TCON
00000089  TMOD
000002EF  UP
000001CC  WRBYTE
000001B7  NEXTL1
0000031F  NONTA
00000268  ONELINE
00000366  OVE
00008003  PCON
00000109  PRESS
00000006  R6
000001A1  REOUT
00000098  SCON
00000488  SHODEL
0000045D  SWA
0000050C  TAB2
0000008D  TH1
000004C3  TOTA
0000043E  UPDWE
000001EB  WRONG
000003C4  NLDA
0000020F  NOTA
00000058  ORI
00000090  P1
00000087  PCON
00008001  PSIG
00000007  R7
00000099  SBUF
00000099  SCON
00000428  SHOW
000004F4  TAB
00000518  TAB3
0000008B  TL1
00000029  TRAN
00000301  UPE
00000336  WRON
```

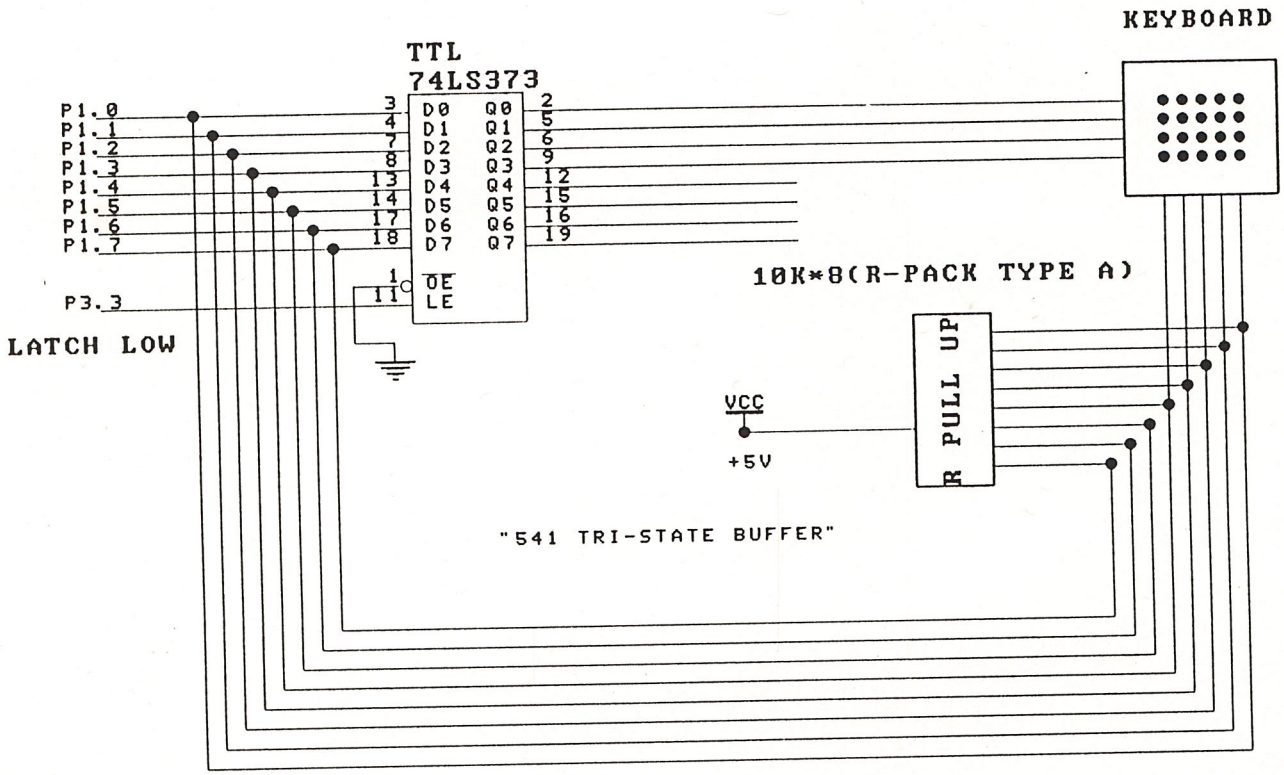


8031 SYSTEM

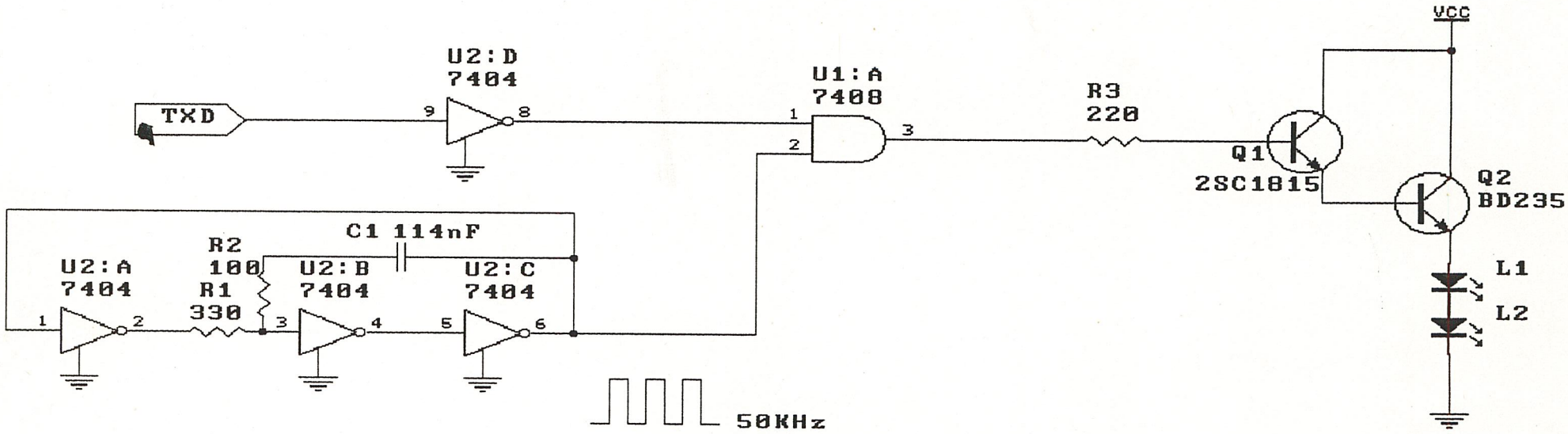
Title		
Size A4	Number	
Date: 11-MAR 1994		Sheet
File: B:PROJECT3/1		Drawn B



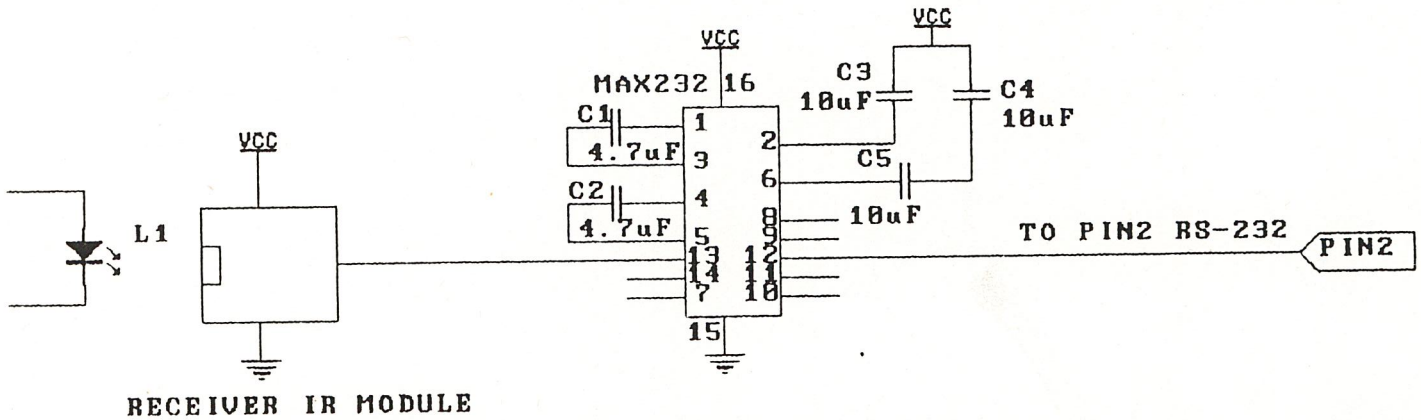
Title		
Size A4	Number	
Date: 11-MAR 1994		Sheet
File: B:PROJECT5/1		Drawn B



The connection of keyboard



TRANSMITING SECTION



RECEIVE SECTION

Title		
Size A4	Number	R
Date: 11-MAR 1994		Sheet
File: B:\RECEIVER/1		Drawn B

บทที่ 3 การสร้างและการทดสอบ

- การทดลองและการทดสอบ

- รายการอุปกรณ์

- ปัญหาที่พบและการนำไปปรับปรุงพัฒนา

การทดลองและการทดสอบ

เริ่มต้นทำโปรเจคนี้ต้องต่อวงจรตามรูป 8031 SYSTEM ซึ่งใช้ 8031 เป็น CPU มี Eprom,Ram,8255,LCD และ Keyboard

วิธีการทดสอบ Hardware

ทำได้โดยใช้ RAMPACK เก็บโปรแกรมแทน EPROM โดยใช้ ETT BOARD.โปรแกรมทดสอบ hardware ง่ายๆ คือเขียนโปรแกรมให้นำข้อมูลไปเก็บไว้ที่ RAM address ไหนก็ได้ แล้ว MOVX คำนั้นกลับมาแล้ว Out ออกไปยัง port A ของ 8255 แสดงได้ดัง

```
                ORG     0000H
                CPU     "8031.TBL"
PORTA:          EQU     8000H
PCONT:          EQU     8003H
                MOV     DPTR,#0000H
                MOV     R0,#2000H
                MOV     A,#55H
RELOAD:         MOVX    @DPTR,A         ; นำค่า 55Hไ ปไว้ใน RAM
                                        ; ตั้งแต่0000H-2000H
                INC     DPTR
                DJNZ    R0,RELOAD
                MOV     DPTR,#8003H
                MOV     A,#80H
                MOVX    @DPTR,A         ; set 8255 as output p
                MOV     DPTR,#1500H
                MOVX    A,@DPTR         ; สุ่มค่าที่ address 1500H
                                        ; ไปไว้ที่ A
                MOV     DPTR,#8000H
                MOVX    @DPTR,A         ; out ค่าใน 1500H ไปที่
                                        ; port A ของ 8255
                END
```

หลังจาก Run Programe แล้วใช้ logic probe วัดที่ PortA ของ 8255 จะต้องได้ 01010101

การทดสอบโปรแกรมรับ

เมื่อเราเขียน programe รับเรียบร้อยแล้ว จะมีวิธีทดสอบโดยใช้ ETT BOARD เป็นตัวส่งข้อมูลเข้ามา ในขณะที่เรา run programe รับรออยู่ ต้องกำหนด BAUD RATE ให้ตรงกันด้วย หลังจากส่งข้อมูลมาแล้ว ถ้าโปรแกรมรับสามารถรับได้จะนำข้อมูลที่ได้ไปเก็บไว้ที่ File in_fi.dat ซึ่งเราสามารถใช้คำสั่ง Type ดูได้ว่าตรงกับข้อมูลในหน่วยความจำของ ETT BOARD ที่ส่งมาหรือไม่ ถ้าตรงแสดงว่าโปรแกรมรับถูกต้อง

การทดสอบโปรแกรมส่ง

หลังจากทราบว่าโปรแกรมรับเรารับได้แล้ว ก็ทดสอบโปรแกรมส่งของเรา เริ่มแรกใช้สายต่อจาก TXD ผ่าน MAX232 เข้าไปยัง RS-232 แล้วใช้คำสั่ง TYPE ดู File in_fi.dat ว่าตรงกับข้อมูลที่เราส่งไปหรือไม่ ถ้าข้อมูลที่ได้ถูกต้อง เราก็ทำการเขียนโปรแกรมต่อโดยนำ File in_fi.dat ไป Decode เพื่อจะแสดงออกทางหน้าจอว่ามีข้อมูลอะไรมาบ้าง ในการ Decode เราจะตกลงกันระหว่างตัวส่งกับตัวรับไว้ว่า BYTE แรกที่มาคือ เลขที่โต๊ะ BYTE ต่อมาจะไม่ใช้ BYTE ต่อมาจะเป็น CODE ของอาหาร เช่น "00" คือ PIZZA , "01" คือ SALAD และ BYTE ต่อมาคือ จำนวนของอาหารอย่างแรก และ BYTE ต่อไปคือ CODE ของอาหารอย่างที่สอง เป็นอย่างนี้เรื่อยๆจนหมดข้อมูล และ BYTE สุดท้ายที่ส่งมาจะเป็น FFH เพื่อบอกให้ตัวรับทราบว่าสิ้นสุดการส่ง

ในการทดสอบการส่ง Infared ก็ทดสอบโดยป้อนไฟให้ตัวรับก่อนแล้วส่งข้อมูลไป โดยใช้ Osciloscope จับดูที่ขา Signal ของตัวรับว่ามี PULSE มาหรือไม่ ถ้ามี PULSE มา ทดลองต่อขา Signal เข้ากับ MAX232 แล้วต่อไปยัง RS232 หลังจากนั้นลอง run โปรแกรมรับ แล้วทำการส่งดู ตรวจสอบ Output ที่ File in_fi.dat ว่ามีความผิดพลาดหรือไม่

รายการอุปกรณ์ที่ใช้

ภาคส่ง

1)	80C31 CPU	1 ตัว
2)	82C55	1 ตัว
3)	LCD MODULE 1 แถว 16 ตัวอักษร	1 ตัว
4)	74LS373	2 ตัว
5)	74LS04	1 ตัว
6)	74LS08	1 ตัว
7)	KEYBOARD 16 KEYS	1 ชุด
8)	6264 RAM	1 ตัว
9)	27265 EPROM	1 ตัว
10)	7805	1 ตัว
11)	XTAL 110.52 MHz	1 ตัว
12)	CAPACITOR 30 pF ceramic	2 ตัว
13)	CAPACITOR 10 uF electrolyte	1 ตัว
14)	CAPACITOR 47 uF electrolyte	1 ตัว
15)	DIODE 1N4001	1 ตัว
16)	R 10K 1/4 W	7 ตัว
17)	VR 10K	1 ตัว
18)	TRANSISTOR 2SC1815	1 ตัว
19)	TRANSISTOR BD235	1 ตัว
20)	ตัวส่ง IR	1 ตัว

ภาครับ

1)	ตัวรับ IR MODULE SONY	1 ตัว
2)	MAX232	1 ตัว
3)	CAPACITOR 4.7 uF electrolyte	2 ตัว
4)	CAPACITOR 10 uF electrolyte	3 ตัว
5)	สาย RS232 พร้อม CONNECTOR	1 ตัว

ปัญหาที่พบและการแก้ไข

1) ในการต่อขา RESET ของ 8255 ถ้าต่อเข้ากับขา RESET ของ 8031 แล้ว 8255 จะไม่ทำงาน แก้ไขโดยการต่อขา RESET ของ 8255 ลง GND 8255 ก็จะทำงาน

2) ในการส่งและรับข้อมูลยาวๆที่มีสัญญาณรบกวนจากภายนอกทำให้ตัวรับๆข้อมูลในขณะที่ยังไม่ได้ส่งข้อมูลไป วิธีแก้คือให้ข้อมูลตัวแรกเป็นตัวบอกว่าจะมีข้อมูลส่งตามมา และสุดท้ายจะมี CODE บอกการสิ้นสุดข้อมูล จะทำให้การรับส่งข้อมูลทำได้แม่นยำยิ่งขึ้น.

ในที่นี้จะใช้ 99H เป็นตัวแรก และ FFH เป็นตัวสุดท้าย

3) ในการส่งด้วย Infared สำคัญที่สุดคือส่วน OSC ที่จะนำไป MOD กับข้อมูลที่ออกจาก TXD ต้องใช้ Function Gen ปรับความถี่ไปเรื่อยๆ เริ่มที่ 15KHz แล้วทดลองส่ง จะได้ความถี่ที่ 20KHz ตัวรับจะรับชัดที่สุด

4) ในการส่งถ้าใช้ BAUD RATE สูงจะทำให้การรับข้อมูลผิดพลาดได้สูง จึงต้องลด BAUD RATE ลงในที่นี้ใช้ 2,400 BAUD

ข้อที่ควรนำไปปรับปรุงพัฒนา

ปรับปรุงให้เพิ่ม BAUD RATE ได้สูงกว่านี้ และให้ได้ไกลกว่าเดิม การที่จะทำให้ BAUD RATE สูงและแม่นยำ จะต้องพัฒนาทั้งภาค Hardware คือปรับปรุงภาค modulation และ วงจรส่ง และ หาตัวส่ง IR ที่มีประสิทธิภาพสูง ส่วนในภาค Software อาจจะมีการส่งข้อมูลซ้ำๆกันหลายๆรอบแล้วโปรแกรมรับก็ทำการรับแล้วนำข้อมูลมาเปรียบเทียบกันและหาข้อมูลที่ถูกต้องที่สุดนำมาใช้ในการประมวลผล

<u>บทสรุป</u> **ทำอะไรบ้างและได้ผลอย่างไร**

โปรเจคของเราชิ้นนี้จะพบว่ามีการดึงกระแสมาใช้ในภาคขับตัวส่งมากมายเหลือเกิน เพราะฉะนั้นเราจึงจำเป็นต้องมีการแยกชุดจ่ายไฟออกเป็น ชุดจ่ายไฟเพื่อเลี้ยงวงจรทั่วไป และชุดจ่ายไฟเพื่อใช้ในการขับภาคส่ง

ในการส่งสัญญาณผ่านคลื่นอินฟราเรดนี้ เราจำเป็นที่จะต้องมีการ encode สัญญาณไว้ด้วยเพื่อไม่ให้เกิดการสับสน เนื่องจากมี Remote ของอุปกรณ์หลายชนิดซึ่งอาจจะเข้ามารบกวนภาครับของเราก็ได้

การทำโปรเจคชิ้นนี้จุดสำคัญก็คือ Software ของทั้งภาคส่งและภาครับ ส่วนด้าน Hardware นั้นเราอาจไม่จำเป็นต้องทำตาม Manual ของ MCS-51 ทั้งหมดเพราะจากการทำโปเจคชิ้นนี้พบว่า เราไม่จำเป็นที่จะต้องใช้ port 8255 ในการ drive ซึ่งถ้าเราตัดทิ้งไปจะสามารถลดการกินกระแสได้อย่างมาก

บรรณานุกรม

1.Blue Ridge Summit,PA [1990],"980 Electronic circuits Volume 2", TAB BOOKS Inc.

2.Kenneth J.Ayala [1991],"8051 Microcontroller",West Publishing Company.

3.Herbert Schildt [1988],"C Power User's Guide",McGraw-Hill Inc.