ปีการศึกษา 2531

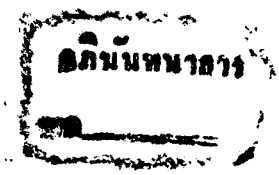
เครื่องนับคะแนนโบว์ลิ่ง

โดย

นายสุรสิทธิ์ อังคุศรีวงศ์
นายอนุวัติ อนุศักดิ์พิทยา

อาจารย์ที่ปรึกษา

อ.สมยศ จุณณะปิยะ



ปริญญานิพนธ์ปีการศึกษา 2531

ภาควิชา วิศวกรรมโทรคมนาคม

คณะวิศวกรรมศาสตร์ สถาบันเทคโนโลยีพระจอมเกล้า เจ้าคุณทหาร ลาดกระบัง

เรื่อง เครื่องนับคะแนนโบว์ลิ่ง

ผู้จัดทำ

1. นายสุรสิทธิ์ อังศุศรีวงศ์ 281285
2. นายอนุวัติ อนุศักดิ์พิทยา 281311

..... อาจารย์ที่ปรึกษา

(อ.สมยศ จุณณะปิยะ)

เครื่องนับคะแนนโบว์ลิ่ง

สุรสิทธิ์ อังศุครีวงศ์
อนุวัติ อนุศักดิ์พิทยา
อ.สมยศ จุณณะบิยะ อาจารย์ที่ปรึกษา
ปีการศึกษา 2531

บทคัดย่อ

ปริญญานิพนธ์ฉบับนี้เสนอการนำระบบไมโครโปรเซสเซอร์ไปประยุกต์ใช้งานกับเครื่องคิดคะแนนโบว์ลิ่ง โดยใช้ไมโครโปรเซสเซอร์เป็นตัวควบคุมการทำงานของระบบ และในภาคตรวจจับแทนระบบเก่า ซึ่งทำให้สามารถลดค่าใช้จ่ายในการนำเข้าวงจรของระบบจากต่างประเทศมาใช้ แนวความคิดนี้ยังผลให้สามารถพัฒนาระบบไว้ใช้ในงานด้านอื่น ๆ ได้อีก การทำงานของระบบที่ได้สร้างขึ้นมีลักษณะการทำงานเป็นไปอย่างมีขั้นตอน ขนาดเล็ก ไม่ผิดพลาด และสามารถดัดแปลง แก้ไข ซ่อมแซม เคลื่อนย้ายได้ง่ายอีกทั้งราคาถูกและมีขีดความสามารถสูง

BOWLING SYSTEM

Surasit Aungsuseewong

Anuwat Anusakpitaya

Somyot Chunnapiyar Advisor

1988

Abstract

This thesis presents the application of microprocessor control to the Bowling System. The system control by microprocessor and a part of sensors replace last system. The new system isn't import system from abroad and cheap. This idea can be use to develop this system for the any system. The operation system is method smaller size, error free operation, greater flexibility and adaptability.

บทที่ 1

บทนำ (INTRODUETION)

ปัจจุบันกีฬาโบว์ลิ่งเป็นกีฬาที่คนนิยมเล่นมากชนิดหนึ่ง เพราะให้ความสนุกและใช้ในการแข่งขันได้ด้วย แต่การจดนับคะแนนยังไม่สะดวกนัก เพราะยังต้องใช้คนจดนับและรวมคะแนน ซึ่งการนับและรวมคะแนนอาจผิดพลาดได้ง่าย และยังเปลืองแรงงานอีกด้วย

วัตถุประสงค์ของปริญญานิพนธ์นี้จึงคิดที่จะแก้ไขปัญหาข้างต้น โดยการนำไมโครคอมพิวเตอร์ควบคุมการทำงานของเครื่องนับคะแนนโบว์ลิ่ง เนื่องจากในปัจจุบันไมโครคอมพิวเตอร์ราคาไม่แพงเหมือนแต่ก่อน และประสิทธิภาพในการทำงานก็สูง

ปริญญานิพนธ์ฉบับนี้ใช้อุปกรณ์ส่วนต่าง ๆ ในการประกอบเป็นเครื่องนับคะแนนโบว์ลิ่งดังนี้

1 วงจรตรวจจับลูกโบว์ลิ่งว่าลูกไหนล้มบ้าง โดยใช้วงจรตรวจจับแบบแสงอินฟราเรด

2. ใช้ MICROPROCESSOR MpF1 ในการประมวลผล (นับคะแนน)

3. วงจรที่แปลงการส่งข้อมูลแบบขนานเป็นการส่งข้อมูลแบบอนุกรม โดยใช้ไอซี เบอร์ 8251

4. ใช้ MICRO COMPUTER ในการรวมคะแนนของผู้เล่นพร้อมทั้งแสดงคะแนนรวมทั้งรายละเอียดว่าทำการโยนกี่ครั้ง มีการ STRIKE และ SPARE หรือไม่ออกทางจอภาพให้ผู้เล่นเห็นคะแนนที่ตัวเองและคู่แข่งขันทำได้

วิทยานิพนธ์ฉบับนี้มีเนื้อเรื่องแยกเป็นบท ๆ ดังนี้

บทที่ 1 กล่าวถึงแนวความคิดของวิทยานิพนธ์ฉบับนี้

บทที่ 2 เป็นส่วนทฤษฎีต่าง ๆ ที่เกี่ยวข้อง

บทที่ 3 เป็นการอธิบายขั้นตอนการคำนวณและการออกแบบในส่วนของฮาร์ดแวร์

บทที่ 4 การทดลองและผลการทดลองที่ได้คำนวณและออกแบบในบทที่ 3

บทที่ 5 สรุปและวิจารณ์วิทยานิพนธ์ฉบับนี้ถึงผลการทดลองและปัญหาต่าง ๆ รวมทั้งแนวความคิดและข้อเสนอในการพัฒนาและแก้ไขปัญหาต่าง ๆ ที่ผู้ทำได้พบเห็น

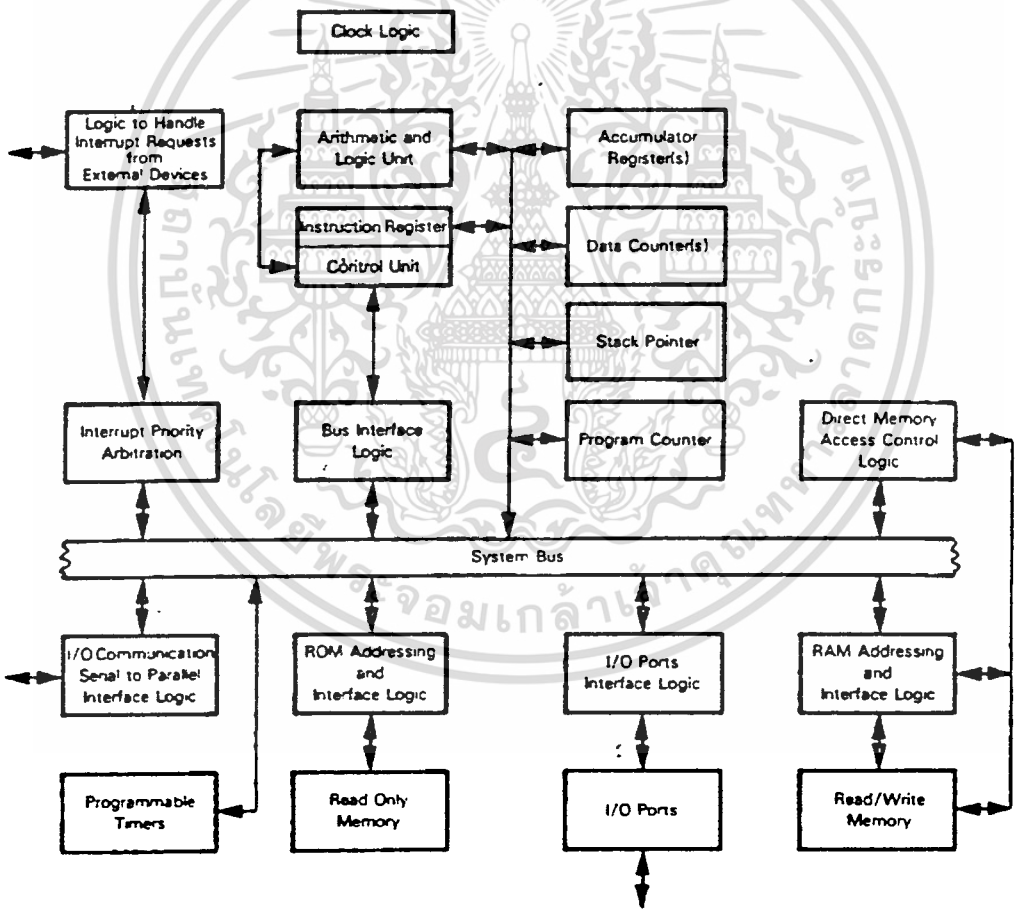
3. หน่วยรับส่งข้อมูล (I/O DATA UNIT)

2.1 หน่วยประมวลผล

ในระบบใช้ Z-80 เป็นซีพียู (CENTRAL PROCESSING UNIT) ซึ่งเป็นไมโครโปรเซสเซอร์ขนาด 8 บิทที่นิยมใช้แพร่หลายในปัจจุบันเป็นการนำเอาวงจรอิเลคทรอนิกส์ที่ยุ่งยากและซับซ้อนมาบรรจุลงบนแผ่น วงจรเดียวที่มีขนาดเล็กมากซึ่งเรียกว่า LSI (LAST SCALE INTEGRATED CIRCUIT) บรรจุอยู่ในตัวถังซึ่งต่อขาออกมา เพื่อใช้ในการติดต่อกับวงจรภายนอกหรืออุปกรณ์สนับสนุน (CHIP SUPPORT) ต่าง ๆ

2.1.1 โครงสร้างโดยทั่วไปของ Z-80

Z-80 ซีพียูจะประกอบไปด้วยส่วนต่าง ๆ ดังบล็อกไดอาแกรม



รูปที่ 2.1.1 แสดงบล็อคไดอะแกรมของ Z-80

ซึ่งแต่ละบล็อกมีลักษณะการทำงานดังต่อไปนี้ คือ

1. ARITHMETICS LOGIC UNIT (ALU) เป็นหน่วยที่ทำหน้าที่ในการคำนวณฟังก์ชันพื้นฐานทางคณิตศาสตร์ และการกระทำฟังก์ชันทางลอจิก เช่น AND และOR, ALU จะสามารถทำหน้าที่ได้อย่างมีประสิทธิภาพมากเพียงใดนั้นขึ้นอยู่กับการออกแบบวงจรภายในของ ALU

2. CONTROL UNIT เป็นหน่วยที่ทำหน้าที่ในการส่งสัญญาณไปควบคุมอุปกรณ์ต่าง ๆ ที่ต่อเชื่อมกับซีพียูให้ทำงานร่วมกันได้อย่างถูกต้อง

3. DATA BUS เป็นบัสสองทิศทาง (BI-DIRECTIONAL) ที่ใช้ในการส่งผ่านข้อมูลระหว่างซีพียูกับอุปกรณ์อื่น ๆ ภายในระบบ จำนวนเส้นของบัสข้อมูล (DATA BUS) จะขึ้นอยู่กับชนิดของซีพียู เช่นในกรณีของ Z80 CPU จะส่งผ่านข้อมูลทีละ 8 บิท ดังนั้นจะมีจำนวนเส้นของบัสข้อมูล 8 เส้น

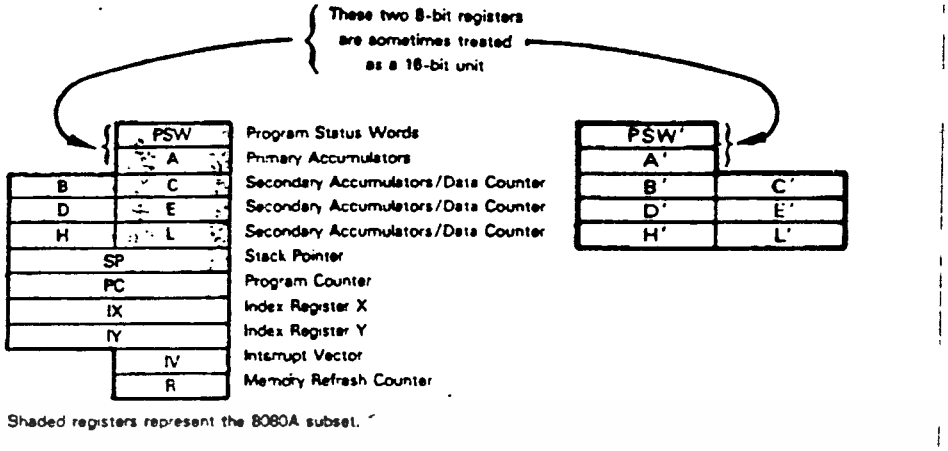
4. CONTROL BUS หรือบัสควบคุม เป็นบัสทางเดียว (UNI-DIRECTIONAL BUS) ที่ใช้ในการส่งผ่านสัญญาณควบคุมให้กับอุปกรณ์ต่าง ๆ ในระบบ

5. ADDRESS BUS เป็นบัสทางเดียว ใช้ส่งผ่านค่าแอดเดรสจากซีพียูออกไปยังหน่วยความจำเพื่อระบุตำแหน่งที่ต้องการรับหรือส่งข้อมูล หรือใช้ระบุตำแหน่งของพอร์ท I/O (INPUT/OUTPUT PORT) ที่ซีพียูต้องการติดต่อด้วย

ซีพียูที่ใช้จะทำงานร่วมกับอุปกรณ์อื่น ๆ อีก 2 ส่วน คือ หน่วยความจำ(MEMORY) และหน่วยรับส่งข้อมูลเข้าออก (I/O DEVICE) ซึ่งในการทำงานตามคำสั่งจากโปรแกรมที่ป้อนเข้ามา ซีพียูจะทำการโอนย้ายคำสั่งหรือข้อมูลระหว่างหน่วยความจำกับรีจิสเตอร์ (REGISTER) ซึ่งจะกล่าวถึงหน้าที่และการทำงานรีจิสเตอร์ภายในซีพียูก่อน

2.1.2 รีจิสเตอร์ต่าง ๆ ในซีพียู

ซีพียูจะประกอบด้วยรีจิสเตอร์ 22 ตัว ซึ่งจะแบ่งออกได้เป็น 2 กลุ่ม คือ รีจิสเตอร์ที่ทำหน้าที่ทั่ว ๆ ไป และรีจิสเตอร์ที่ทำหน้าที่เฉพาะงาน



รูปที่ 2.1.2 แสดงรีจิสเตอร์ต่าง ๆ ภายใน Z-80

1. รีจิสเตอร์ที่ทำหน้าที่ทั่ว ๆ ไป แบ่งเป็นรีจิสเตอร์หลัก ได้แก่ A, B, C, D, E, H และ L มีความจุขนาด 8 บิท รีจิสเตอร์เหล่านี้ใช้เก็บข้อมูลชั่วคราว นอกจากนี้ยังสามารถรับข้อมูลจากหน่วยความจำหรืออาจจะทำการย้ายข้อมูลไปเก็บในหน่วยความจำก็ได้ และรีจิสเตอร์สำรอง ได้แก่ A', B', C', D', E', H' และ L' ซึ่งเป็นรีจิสเตอร์ที่ทำหน้าที่เก็บข้อมูลที่มาจากรีจิสเตอร์หลักในกรณีที่ต้องใช้รีจิสเตอร์หลักในการทำงานอย่างอื่นก่อน ดังนั้นรีจิสเตอร์กลุ่มนี้จึงไม่สามารถกระทำขบวนการทางคณิตศาสตร์และลอจิกได้

รีจิสเตอร์ เรียกว่า แอกคิวมูเลเตอร์ (ACCUMULATOR) ทำหน้าที่เก็บข้อมูลชั่วคราวที่ได้จากการทำขบวนการทางคณิตศาสตร์ เช่น บวกหรือลบข้อมูล 2 จำนวน ผลลัพท์ที่ได้จะเก็บไว้ในรีจิสเตอร์ A นี้ นอกจากนี้ในการปฏิบัติตามคำสั่งที่ใช้กับข้อมูลขนาด 16 บิท ซีพียูจะนำเอารีจิสเตอร์แฟลก "F" (FLAG REGISTER) มาใช้ร่วมกับรีจิสเตอร์ A เรียกว่า คู่รีจิสเตอร์AF ซึ่งมีขนาด 16 บิท นอกจากนี้ยังมีคู่รีจิสเตอร์ 16 บิท อื่น ๆ อีก คือ BC, DE และ HL

2. รีจิสเตอร์ที่ใช้งานเฉพาะอย่าง ได้แก่ รีจิสเตอร์ I, R, IX, IY, SPและ PC ซึ่งทำหน้าที่ต่าง ๆ ดังนี้

รีจิสเตอร์ I (INTERRUPT PAGE ADDRESS REGISTER) เมื่อมีการอินเทอร์รัพท์เกิดขึ้นจำเป็นต้องบอกตำแหน่งของหน่วยความจำที่เก็บโปรแกรมตอบสนองการอินเทอร์รัพท์ รีจิสเตอร์ I จะทำหน้าที่เก็บค่า 8 บิทบนของตำแหน่งข้อมูลในหน่วยความจำ ส่วนค่า 8 บิทล่างจะป้อนจากภายนอกให้แก่ซีพียูค่าทั้งสองจะประกอบกันเป็นค่าแอดเดรสที่ระบุตำแหน่งของโปรแกรมการตอบสนองการอินเทอร์รัพท์

รีจิสเตอร์ R (MEMORY REFRESH REGISTER) เป็นรีจิสเตอร์ขนาด 7 บิทที่ถูกใช้ในการรีเฟรช (REFRESH) DYNAMIC RAM และค่ารีจิสเตอร์ R จะเพิ่มขึ้นเองโดยอัตโนมัติ ในทุกๆ ครั้งที่มีการเฟทช์คำสั่งจากหน่วยความจำ

รีจิสเตอร์ IX และ IY (INDEX REGISTER) เป็นรีจิสเตอร์ที่มีขนาด 16 บิท มีประโยชน์ใช้บ่งบอกตำแหน่งในหน่วยความจำแบบ INDEX ADDRESSING MODE โดยจะกำหนดให้ค่าใน INDEX REGISTER เป็นค่าอ้างอิง แล้วใช้คำสั่งบ่งบอกว่าตำแหน่งของข้อมูลที่ต้องการอยู่ห่างจากค่าอ้างอิงนี้เท่าใด โดยจะบอกค่าระยะห่างในรูปของ TWO COMPLEMENT

รีจิสเตอร์ SP (STACK POINTER) มีขนาด 16 บิท ในหน่วยความจำชนิดRAM จะมีส่วนหนึ่งที่ถูกกำหนดให้เป็นที่เก็บข้อมูลชั่วคราว ส่วนนี้เรียกว่าสแตก (STACK) ซึ่งมีลักษณะการเก็บข้อมูลแบบ LIFO (LAST IN FIRST OUT) เราสามารถที่จะเก็บข้อมูลลงบนสแตกโดยใช้คำสั่ง PUSH และเมื่อต้องการดึงข้อมูลออกจากสแตกต้องใช้คำสั่ง POP

รีจิสเตอร์ PC (PROGRAM COUNTER) เป็นรีจิสเตอร์ขนาด 16 บิทที่ใช้ในการเก็บตำแหน่งของหน่วยความจำที่ซีพียูจะเฟทช์ (FETCH) คำสั่งหลังจากที่เฟทช์คำสั่งเรียบร้อยแล้ว ค่าในรีจิสเตอร์ PC จะเพิ่มขึ้น และจะชี้ไปยังตำแหน่งของคำสั่งถัดไป สามารถเปลี่ยนแปลงค่าในCALLได้โดยใช้คำสั่ง CALL หรือ JUMP

รีจิสเตอร์ F (FLAG REGISTER) ประกอบด้วย

SIGN FLAG (S) : แฟลกเครื่องหมาย

ZERO FLAG (Z) : แฟลกศูนย์

HALF CARRY FLAG (H) : แฟลกหดครึ่ง

PARITY/OVERFLOW FLAG (P/V) : แฟลกพาริตีหรือโอเวอร์โฟล์ว

SUBTRACT FLAG (N) : แฟลกลบ

CARRY FLAG (C) : แฟลกตัวทด

นำเอาแฟลกเหล่านี้มาประกอบร่วมกับ บิทว่าง (X : ไม่มีความหมาย) อีก 2 บิทเพื่อทำเป็นรีจิสเตอร์ขนาด 8 บิท สำหรับรายละเอียดของแฟลกเหล่านี้จะไม่ขอกล่าวถึง

2.1.3 รายละเอียดของขา Z 80 (Z80 PIN OUTS)

A0-A15 (ADDRESS BUS) : เป็นขาสัญญาณเอาท์พุทแบบ TRI-STATE ใช้บ่งบอกตำแหน่งหน่วยความจำได้ถึง 2^{16} = 65536 ตำแหน่ง A0-A7 จะแสดงตำแหน่งของพอร์ทที่ซีพียูต้องการติดต่อด้วย นอกจากนี้ขา A0-A6 จะให้ค่ารีเฟรชแอดเดรสออกมาขณะที่ซีพียูให้สัญญาณรีเฟรช

D0-D7 (DATA BUS) : เป็นขาสัญญาณอินพุท/เอาท์พุท TRI-STATE แบบสองทิศทาง ซึ่งเป็นทางผ่านของข้อมูลระหว่างซีพียูกับหน่วยความจำและอุปกรณ์ I/O

$\overline{M1}$ (MACHINE CYCLE ONE) : เป็นขาเอาท์พุทแอดทีฟที่ลอจิก "0" ขา $\overline{M1}$ นี้จะแอดทีฟขณะที่ซีพียูทำการเฟทช์ออฟโคดของคำสั่ง ในกรณีที่คำสั่งที่จะเฟทช์เข้ามานั้นมีขนาด 2 ไบท์ $\overline{M1}$ จะแอดทีฟในทุก ๆ ไซเคิลการเฟทช์แต่ละไบท์

$\overline{MREQ}$ (MEMORY REQUEST) : เป็นสายเอาท์พุทแบบ TRI-STATE แอดทีฟที่ลอจิก "0" เพื่อเป็นการบ่งบอกว่าซีพียูกำลังกระทำการติดต่อกับหน่วยความจำ

$\overline{IORQ}$ (INPUT/OUTPUT REQUEST) : เป็นสายเอาท์พุทแบบ TRI-STATE จะแอดทีฟที่ลอจิก "0" เพื่อเป็นการบ่งบอกว่าซีพียูกำลังทำการติดต่อกับอุปกรณ์ I/O และเมื่อ $\overline{IORQ}$ และ $\overline{M1}$ แอดทีฟทั้งคู่จะเป็นการบ่งบอกการตอบรับการอินเทอร์รัพท์ (INTERRUPT ACKNOWLEDGE)

$\overline{RD}$ (MEMORY READ) : เป็นขาเอาท์พุท TRI-STATE จะแอดทีฟที่ลอจิก "0" เมื่อซีพียูต้องการอ่านข้อมูลจากหน่วยความจำหรืออุปกรณ์ I/O และซีพียูจะรับข้อมูลจากบัสข้อมูลเข้าไปเมื่อสัญญาณนี้เปลี่ยนระดับลอจิกจาก "0" เป็น "1"

$\overline{WR}$ (MEMORY WRITE) : เป็นขาเอาท์พุทแบบ TRI-STATE จะแอดทีฟที่ลอจิก "0" เมื่อซีพียูต้องการส่งข้อมูลออกไปให้หน่วยความจำหรืออุปกรณ์ I/O

$\overline{RFSH}$ (REFRESH) : เป็นขาเอาท์พุท จะแอดทีฟเมื่อ 7 บิทล่าง (A0-A6) ของบัสแอดเดรสให้ค่ารีเฟรชออกมา

$\overline{HALT}$ (HALT STATE) : เป็นขาเอาท์พุท จะแอดทีฟที่ลอจิก "0" เมื่อซีพียูอยู่ในสภาวะของการ $\overline{HALT}$ คือ ซีพียูจะทำคำสั่ง NOP (NO OPERATION) เพื่อให้เกิดการรีเฟรชได้และซีพียูจะหลุดพ้นจากสภาวะการ HALT เมื่อได้รับการรีเซ็ทหรือถูกอินเทอร์รัพท์

$\overline{WAIT}$: เป็นขาอินพุทแอทีฟที่ลอจิก "0" และจะมีการตรวจสอบสัญญาณนี้ที่ขอบขาลงของคล็อกลูกที่ 2 ของทุก ๆ MACHINE CYCLE เมื่อมีการตรวจพบว่าขาอินพุทนี้แอคทีฟจะมีการแทรก WAIT STATE ให้กับแต่ละ MACHINE CYCLE เพื่อเป็นการรอให้อุปกรณ์ภายนอกทำงานให้ทันกับการทำงานของซีพียูและซีพียูจะแทรก WAIT STATE จนกว่าจะมีการตรวจสอบพบว่าขา WAIT จะมีลอจิกเป็น "1"

$\overline{\text{INT}}$ (INTERRUPT REQUEST) : เป็นขาอินพุทแอคทีฟที่ลอจิก"0" ซีพียูจะตรวจสอบระดับสัญญาณนี้ขณะทุก ๆการสิ้นสุดสัญญาณของ INSTRUCTION CYCLE (LAST STATE)

$\overline{\text{NMI}}$ (NON MASKABLE INTERRUPT) : เป็นขาอินพุทแอคทีฟที่ลอจิก "0" สัญญาณ NON MASKABLE INTERRUPT เป็นสัญญาณที่มีระดับความสำคัญในการขออินเทอร์รัพท์สูงกว่าสัญญาณ INTERRUPT REQUEST ซีพียูจะตอบรับการอินเทอร์รัพท์ชนิดนี้เสมอโดยที่ไม่สามารถ DISABLE ได้ด้วย SOFTWARE

$\overline{\text{RESET}}$: เป็นขาอินพุทแอคทีฟที่ลอจิก "0" สัญญาณนี้จะทำการ INITIALIZE CPU โดยทำการรีเซ็ท INTERRUPT FLIP-FLOP และเซ็ทค่าในโปรแกรมเคาน์เตอร์ (PROGRAM COUMTER) ให้เป็น OOOOH และในสภาวะการรีเซ็ทนี้ บัสแอดเดรสและบัสข้อมูลจะอยู่ในสภาวะ HIGH IMPEDANCE และสัญญาณควบคุมต่าง ๆ จะอยู่ในสภาวะ INACTIVE

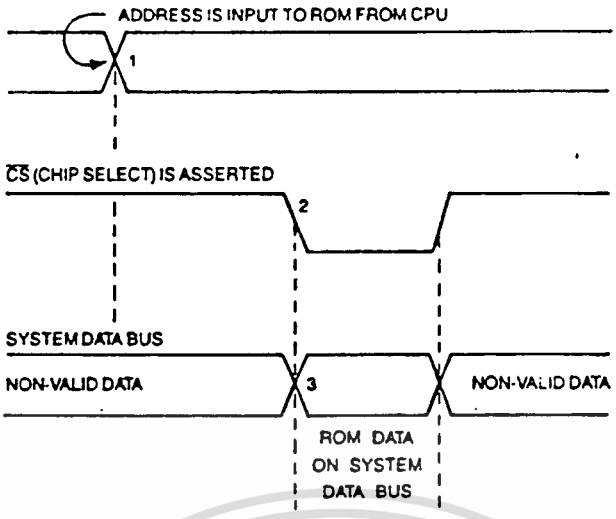
$\overline{\text{BUSRQ}}$ (BUS REQUEST) : เป็นขาอินพุทแอคทีฟที่ลอจิก "0" สัญญาณ BUS REQUEST เป็นสัญญาณที่มีลำดับความสำคัญสูงกว่าสัญญาณ NON MASKABLE INTERRUPT และมีการตรวจสอบสัญญาณนี้ทุก ๆ การสิ้นสุดของ MACHINE CYCLE อุปกรณ์ภายนอกจะให้สัญญาณนี้แก่ซีพียู เมื่อต้องการใช้บัสข้อมูลและบัสแอดเดรสโดยเปรียบเสมือนว่าเป็นการถอดซีพียู ออกจากระบบบัส

$\overline{\text{BUSAK}}$ (BUS ACKNOWLEDGE) : เป็นขาเอาท์พุทแอคทีฟที่ลอจิก "0" ขานี้จะแอคทีฟเมื่อซีพียูตอบสนองการต่อสัญญาณ BUS REQUEST และจะทำให้บัสข้อมูล บัสควบคุมและบัสแอดเดรสมีสภาวะเป็น HIGH IMPEDANCE ซึ่งทำให้อุปกรณ์ภายนอกใช้บัสเหล่านี้ได้โดยไม่มีผลต่อซีพียู

จากที่ได้กล่าวมานำซีพียูไปใช้งานในระบบซึ่งจะทำงานร่วมกับอุปกรณ์สนับสนุนอื่น ๆ ซึ่งได้แก่ EPROM 2716 ซึ่งใช้เป็นหน่วยความจำ Z-80 CTC ซึ่งใช้เป็นตัวสร้างความถี่ที่ใช้ในการส่ง (BAUD RATE GENERATOR) พอร์ทข้อมูลต่าง ๆ ของระบบซึ่งจะกล่าวโดยต่อไป

2.2 หน่วยความจำ

ในระบบใช้หน่วยความจำ EPROM (ERASABLE PROGEAMMABLE ROM) ซึ่งเป็นหน่วยความจำแบบ ROM (READ ONLY MEMORY) ชนิดหนึ่งที่ใช้สำหรับเก็บข้อมูลโดยจะถูกโปรแกรมโดยการให้สัญญาณที่มีแรงดันสูง (HIGH VOLTAGE SIGNAL) ผ่านเข้าไปในตัว EPROM และสามารถเปลี่ยนแปลงได้โดยการลบข้อมูลเดิมออกก่อนแล้วโปรแกรมเข้าไปใหม่ ข้อมูลที่ถูกโปรแกรมเข้าไปจะคงสภาพอยู่เดิมไม่สูญหายเมื่อไม่มีการจ่ายไฟเลี้ยงแก่ระบบ



รูปที่ 2.2.2 แสดงไทยะแกรมของสัญญาเที่ใช้ในการอ่านข้อมูล

1. ซีพียูส่งค่าแอดเดรสให้กับหน่วยความจำ
2. สัญญาเลือกเพอร์ทถูกส่งให้กับหน่วยความจำ
3. ข้อมูลจากหน่วยความจำถูกส่งออกมายังบัสข้อมูล

ลำดับขั้นตอนที่เกิดขึ้นในการอ่านข้อมูลแต่ละครั้งออกจากหน่วยความจำ เป็นลำดับขั้นตอนการทำงานทั่ว ๆ ไปโดยไม่คำนึงถึงชนิดของซีพียู ที่ใช้

1. ค่าแอดเดรสจะถูกป้อนเข้าไปยังหน่วยความจำ โดยซีพียูค่าแอดเดรสนี้จะกำหนดตำแหน่งของข้อมูลที่ต้องการอ่าน โดยข้อมูลจะถูกอ่านออกมาเพียงครั้งละ 1 ไบท์เท่านั้น

2. ซีพียูจะคอยอยู่ช่วงเวลาหนึ่ง (WAIT STATE) เรียกว่า ACCESS TIME ประมาณ 100-300 NANOSECONDS ขึ้นอยู่กับชนิดของหน่วยความจำ ซึ่งเป็นเวลาที่หน่วยความจำใช้ในการถอดรหัสของแอดเดรส (DECODE ADDRESS) ของข้อมูลที่ต้องการจะอ่านออกมาที่เอาท์พุทของหน่วยความจำ

3. CHIP SELECT LINE จะถูกทำให้แอคทีฟ (ACTIVE) เพื่อให้ข้อมูลออกมาที่บัสข้อมูลของระบบได้ ต่อจากนั้นซีพียูจะสโตรบ (ACTIVE) ข้อมูลเข้าไปเก็บไว้ในรีจิสเตอร์ภายในซีพียู

4. CHIP SELECT LINE จะถูกสั่งให้เลิกทำงาน (สถานะ INACTIVE) เพื่อทำ

ให้ข้อมูลที่อยู่บนบัสข้อมูลของระบบหายไป

ลำดับการทำงานทั่ว ๆ ไปที่กล่าวมานี้ จะเกิดขึ้นทุกครั้งที่ซีพียูอ่านข้อมูลจาก ROM ในรูป 2.2.2 แสดงไดอะแกรมเวลา (TIMING DIAGRAM) ทั่ว ๆ ไปของลำดับการทำงาน

ค่าแอดเดรสที่ตำแหน่ง 0000H จะเป็นแอดเดรสเริ่มต้นของหน่วยความจำในระบบนี้ เมื่อซีพียูถูกรีเซ็ทจากภายนอกหรือเป็นการรีเซ็ทที่เกิดจากการจ่ายไฟเลี้ยงให้แก่ระบบในครั้งแรก ซีพียูจะกระโดด (JUMP) ไปทำงานที่แอดเดรส 0000H ของ EPROM

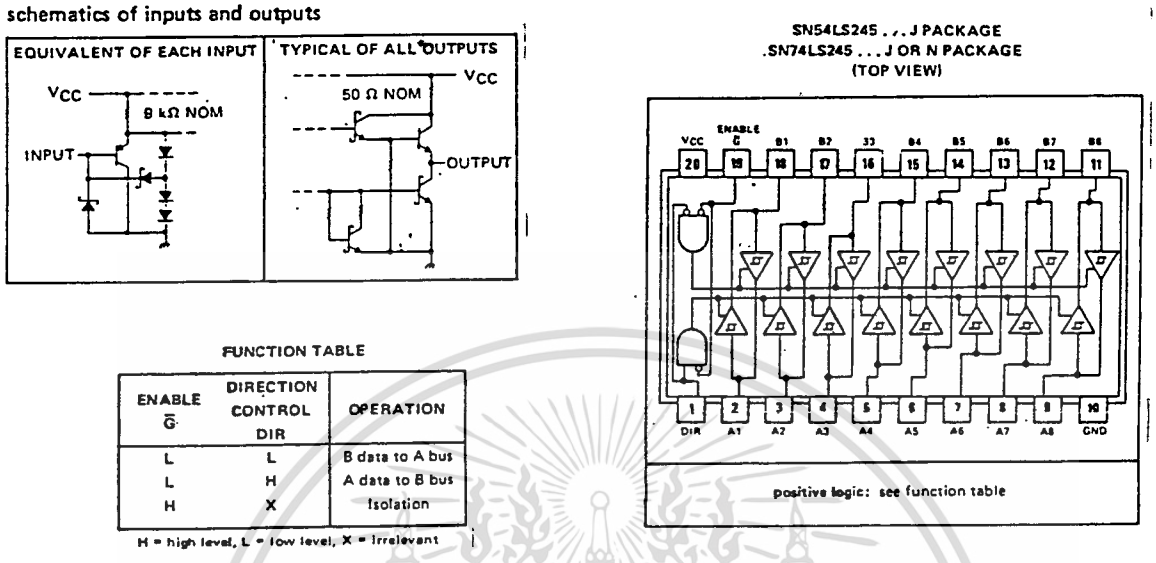
2.2.3 ผังแอดเดรส

ไนซีพียูเป็นไมโครโปรเซสเซอร์ที่มีบัสแอดเดรส 16 เส้น (A_0-A_{15}) ซึ่งสามารถติดต่อกับแอดเดรสต่าง ๆ โดยตรงได้ถึง 2^{16} สำหรับในระบบนี้ EPROM 2716 ซึ่งมีความจุประมาณ 2K ตำแหน่งแอดเดรส 2048 ตำแหน่งซึ่งก็เป็นการเพียงพอในระบบหน่วยความจำของระบบ โดยจะทำงานร่วมกับซีพียู โดยทำการต่อเชื่อมกับขาต่างๆ ของซีพียู โดยทำการเลือกสัญญาณเป็น CHIP SELECT ที่เหมาะสมก่อนที่จะทำการสร้างระบบทางฮาร์ดแวร์ตามโครงสร้างของผังหน่วยความจำ

2.3 หน่วยรับส่งข้อมูล

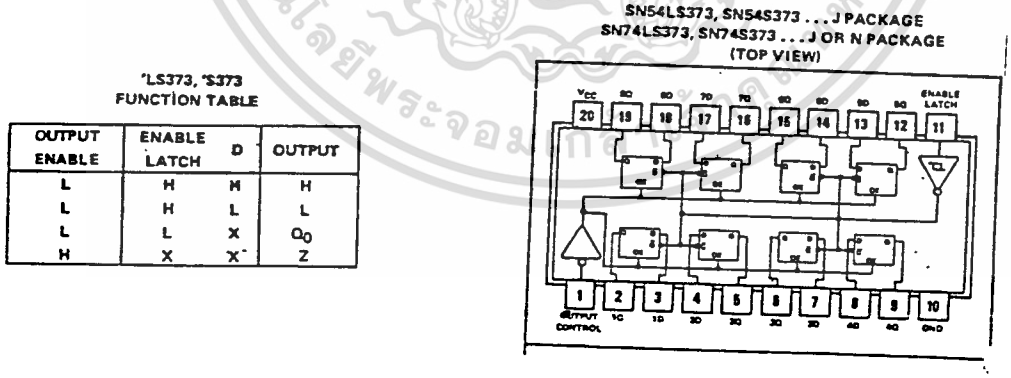
ในขบวนการทำงานของระบบใช้ไมโครโปรเซสเซอร์ทำการติดต่อกับหน่วยรับข้อมูลทางพอร์ทโดยเลือกใช้แอดเดรสและสัญญาณต่าง ๆ ที่ถูกสร้างขึ้นจากโปรแกรมควบคุมการทำงานไมโครโปรเซสเซอร์นำมาถอดรหัส (DECCDER) เพื่อสร้างเลือกพอร์ทให้กับพอร์ทต่าง ๆ โดยใช้คุณสมบัติการทำงานของไอซีที่ใช้ในการเลือกพอร์ทติดต่อ

สำหรับอุปกรณ์อินพุทใช้ไอซีเบอร์74245 เป็นพอร์ทในการรับข้อมูลเข้ามาในระบบจากสถานะตรวจจับทั้งของพินและลูกโบว์ลิ่ง โค้ดของราง เข้ามาประมวลผลเบื้องต้นก่อนที่จะส่งข้อมูลที่ได้ออกไปยังอุปกรณ์เอาท์พุท



รูปที่ 2.3.1 แสดงการทำงานไอซีเบอร์ 74245

ส่วนสำหรับอุปกรณ์เอาท์พุทใช้ไอซีเบอร์ 74373 เป็นพอร์ทในการส่งข้อมูลของสถานะตรวจจับของพินออกมา

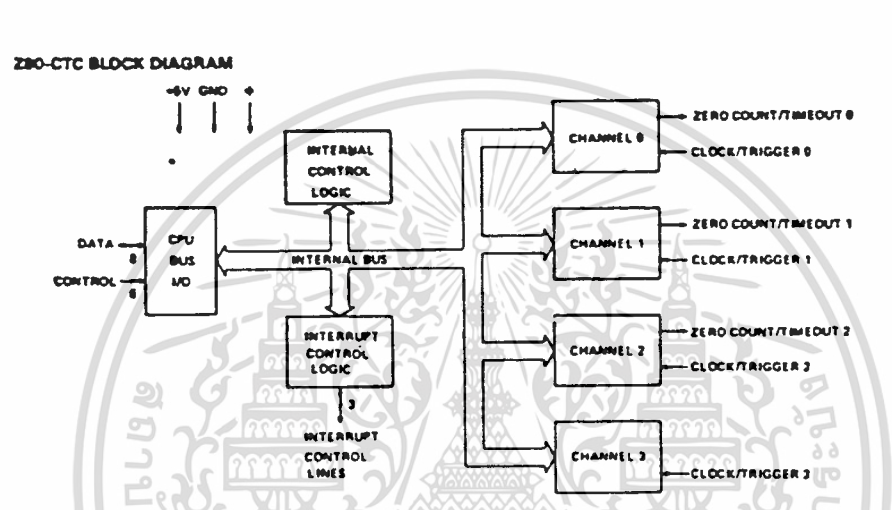


รูปที่ 2.3.1 แสดงการทำงานไอซีเบอร์ 74373

2.4 Z-80 CTC

2.4.1 บล็อกไดอาแกรมของ CTC

Z-80 COUNTER TIMER CHIP หรือเรียกอีกอย่างหนึ่งว่า Z-80 CTC เป็นชิพที่ถูกออกแบบมาสำหรับใช้ร่วมกับ Z-80 ซีพียูโดยเฉพาะ มีความคล่องตัวในการนำมาใช้งานสูงมากสามารถนำมาทำงานร่วมกับระบบโดยใช้โปรแกรมในการควบคุม การเชื่อมต่อซีทีซีเข้ากับไมโครโปรเซสเซอร์เพื่อนำมาใช้ในหน้าที่ที่ต้องการได้ การทำงานของซีทีซีและรีจิสเตอร์ภายในมีรายละเอียดดังนี้



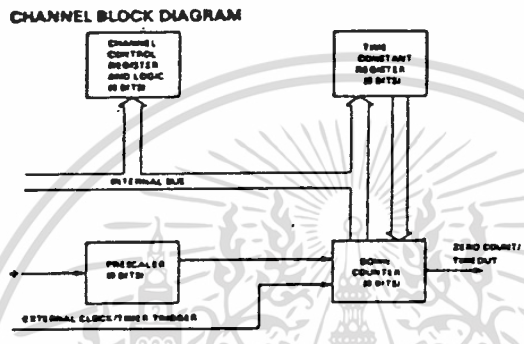
รูปที่ 2.4.1 บล็อกไดอะแกรมของซีทีซี

ซีทีซีมีชื่อที่มาจากลักษณะการทำงานพื้นฐาน 2 ประการของชิพคือ การนับและการเป็นฐานเวลา (COUNTER/TIMER) โดยภายในของชิพนี้มีเคาน์เตอร์และไทม์เมอร์ที่เป็นอิสระต่อกันอยู่ 4 แชลแนล ประกอบอยู่บนตัวถังแบบ DIP (DUAL-IN-LINE PACKAGE) 24 ขาเป็นอิสระต่อกัน เรียกว่า Ch0,Ch1,Ch2 Ch3 ซึ่งใน 3 แชลแนลแรกมีเส้นสัญญาณใช้สำหรับเชื่อมต่อกับระบบภายนอกอยู่ 2 เส้นเรียกว่า ZERO COUNT/TIME OUT CLOCK/TRIGGER ซึ่งเป็นเอาท์พุทและอินพุท ส่วนในแชลแนล 3 มีเพียงคล็อกและทริกเกอร์เพียงเส้นเดียว ทั้งนี้เพราะว่าถูกจำกัดด้วยจำนวนขาของตัวถังแบบดิพ 24 ขา จากรูปบล็อกไดอาแกรมจะเห็นว่ามีบล็อกที่มีชื่อว่า INTERNAL CONTROL LOGIC ทำหน้าที่ควบคุมการส่งผ่านข้อมูลบนบัสภายในซีทีซีให้ถูกต้อง และบล็อกของ INTERRUPT CONTROL LOGIC ทำหน้าที่ควบคุม การอินเทอร์รัพท์

สำหรับบล็อกสุดท้าย คือ ซีพียูบัสไอโอ ซึ่งใช้ในการเชื่อมต่อระหว่างซีพียูกับซีทีซี ประกอบด้วยบัสข้อมูล 8 เส้น และบัสควบคุม 6 เส้น การติดต่อระหว่างซีพียูกับซีทีซีจะต้องผ่านบล็อกนี้ ดังนั้นบล็อกนี้สามารถที่จะถูกโปรแกรมให้ทำตามจุดประสงค์ของผู้โปรแกรมได้

2.4.2 รายละเอียดของแต่ละแชลแนลบล็อก

เพื่อที่จะทำการโปรแกรมและใช้ซีทีซีให้มีประสิทธิภาพ โดยพิจารณาจากบล็อกแชนแนล 0 เท่ากับพิจารณาแชลแนลทั้งหมดด้วย ยกเว้นแชลแนล 3 เพราะว่าในแชนแนล 3 ไม่มีสายเอาท์พุท



รูปที่ 2.4.2 บล็อกไดอะแกรมแสดงโครงสร้างภายในของเคาน์เตอร์

จากรูปเป็นการแสดงบล็อกไดอาแกรมของภายใน แต่ละแชนแนล ซึ่งประกอบไปด้วย CHANNDL CONTROL REGISTER, TIME CONSTANT REGISTER AND COUNTER

บล็อก CHANNEL CONTROL REGISTER เป็นรีจิสเตอร์ที่ผู้โปรแกรมใช้เขียนข้อมูลเข้าไป เพื่อกำหนดลักษณะการทำงานของแชนแนล

บล็อก TIME CONSTANT REGISTER เป็นรีจิสเตอร์ที่มีขนาด 8 บิท ซึ่งมีค่าได้ตั้งแต่ OOH ไปจนถึง OFFH ค่าที่อยู่ในรีจิสเตอร์นี้จะนำไปใช้สำหรับเซ็ทค่าที่ใช้เริ่มต้นการนับจำนวนคล็อกของ DOWN COUNTER

บล็อก DOWN COUNTER เป็นเคาน์เตอร์ที่มีขนาด 8 บิท มีลักษณะการนับแบบนับลง ที่บล็อกนี้มีเส้นสัญญาณอินพุทชื่อว่า EXTERNAL CLOCK/TIMER TRIGGER สัญญาณนี้เป็นสัญญาณคล็อกที่จะป้อนให้กับ DOWN COUNTER โดยตรง หรือ อาจจะใช้เป็นสัญญาณ ENABLE สำหรับให้ PRESCALER ทำงานก็ได้ ขึ้นอยู่กับการโปรแกรม CTC ส่วนเส้นสัญญาณ ZERO COUNT/TIME OUT ใช้เป็นเอาท์พุท

ที่จะแอกทีฟเมื่อค่าที่นับในเคาน์เตอร์มีค่าเป็น OOH

ยังมีอีกบล็อกหนึ่งอยู่หน้าบล็อก DOWN COUNTER มีชื่อว่า PRESCALER ซึ่งมีขนาด 8 บิท หน้าที่ของบล็อกนี้ จะใช้สำหรับเป็นตัวหารเริ่มแรกให้กับสัญญาณคล็อกที่จะป้อนให้กับ DOWN COUNTER โดยสามารถเลือกได้ 2 ค่าคือ 16 หรือ 256 ซึ่งขึ้นกับการโปรแกรมของผู้ใช้

2.4.3 ขาสัญญาณต่าง ๆ ของซีทีซี

D0-D7 : เป็นขาที่ใช้สำหรับรับและ ส่งข้อมูลระหว่างซีทีซีกับ Z-80 CPU โดยมี D0 เป็น บิทที่มีนัยสำคัญต่ำ

CS1 , CS0 : เป็นขาที่ใช้เลือกแชนแนล จากที่ได้กล่าวมาแล้วว่าซีทีซี มีอยู่ด้วยกัน 4 แชนแนล ฉะนั้นเมื่อต้องการเลือกให้แชนแนลใดทำการเขียนหรืออ่านข้อมูลจากอุปกรณ์ I/O ก็ทำได้ โดยให้ลอจิกที่ถูกต้องแก่ CS1 และ CS0 สำหรับแชนแนลนั้น ๆ ดังตารางข้างล่างนี้ (โดยทั่ว ๆ ไป แล้วขา CS1 และ CS0 จะต่อกับขาแอดเดรส A1 และ A0 ของ Z-80 ซีพียู)

CS1	CS0	ACTIVE CHANNEL
0	0	0
0	1	1
1	0	2
1	1	3

$\overline{CE}$: ขานี้เรียกว่า CHIP ENABLE INPUT, Z-80 CPU จะเลือกใช้ ซีทีซี โดยใช้ขานี้เป็นตัวกำหนด ถ้าซีพียูต้องการติดต่อกับซีทีซีแล้ว จะทำให้ขา $\overline{CE}$ นี้มีลอจิกเป็น "0" (ACTIVE LOW) ซึ่งลอจิกที่ป้อนให้กับขานี้ โดยทั่ว ๆ ไปจะได้มาจากการถอดรหัสค่าแอดเดรส A7-A2 ของ Z-80 บัสแอดเดรส

CLOCK : เป็นขาที่ใช้สำหรับป้อนสัญญาณคล็อก ซึ่งเป็นสัญญาณเดียวกันกับที่ป้อนให้กับ Z-80 CPU โดย CTC จะใช้สำหรับให้การเคลื่อนย้ายข้อมูลภายในมีความสัมพันธ์กัน ฉะนั้นขานี้สามารถต่อโดยตรงเข้ากับขา CLK ของ Z-80 CPU

$\overline{M1}$: เป็นขาที่รับสัญญาณ $\overline{M1}$ ซึ่งได้มาจาก Z-80 CPU สัญญาณ $\overline{M1}$ นี้เมื่อใช้ร่วมกับสัญญาณ $\overline{RD}$ จะใช้เป็นตัวกำหนดการเพ็ทช์ออพโคดจากระบบหน่วยความจำและเมื่อใช้ร่วมกับ $\overline{IORQ}$ จะใช้เป็นตัวกำหนดสภาพการตอบรับการขออินเทอร์รัพท์

$\overline{\text{IORQ}}$: ขานี้ใช้เป็นขาอินพุทของสัญญาณ $\overline{\text{IORQ}}$ ที่ได้มาจากขา $\overline{\text{IORQ}}$ ของ Z-80 CPU สัญญาณนี้จะส่งมาให้ CTC เพื่อบอกให้ CTC รู้ว่า Z-80 มีความต้องการอ่านหรือเขียนข้อมูลต่าง ๆ กับ CTC

$\overline{\text{RD}}$: ขานี้ต่อกับขา $\overline{\text{RD}}$ ของ Z-80 โดยตรง เพื่อบอกให้รู้ว่าZ-80 กำลังอ่านข้อมูลจากหน่วยความจำหรืออุปกรณ์ I/O จะสังเกตได้ว่า CTC นี้ไม่มีขา $\overline{\text{WR}}$ (WRITE INPUT) เมื่อ Z-80 ต้องการเขียนข้อมูลให้กับ CTC จะต้องทำให้ขา $\overline{\text{CE}}$ ลอจิกเป็น "0" ,$\overline{\text{RD}}$ มีลอจิกเป็น "1" และ $\overline{\text{IORQ}}$ เป็นลอจิก "0"

IEI (INTERRUPT ENABLE INPUT) : ขานี้เป็นขาอินพุทที่ใช้กับการอินเทอร์รัพท์ โดยแอคทีฟที่ลอจิก "1" กล่าวคือเมื่อมีลอจิก "1" ป้อนเข้ามาที่ขานี้ CTC จะให้สัญญาณการขออินเทอร์-รัพท์ออกไปที่ขา $\overline{\text{INT}}$ OUTPUT แต่เมื่อมีลอจิก "0" เข้ามาที่ขา IEI จะทำให้สัญญาณการขออินเทอร์-รัพท์ที่ขา $\overline{\text{INT}}$ หมดไป

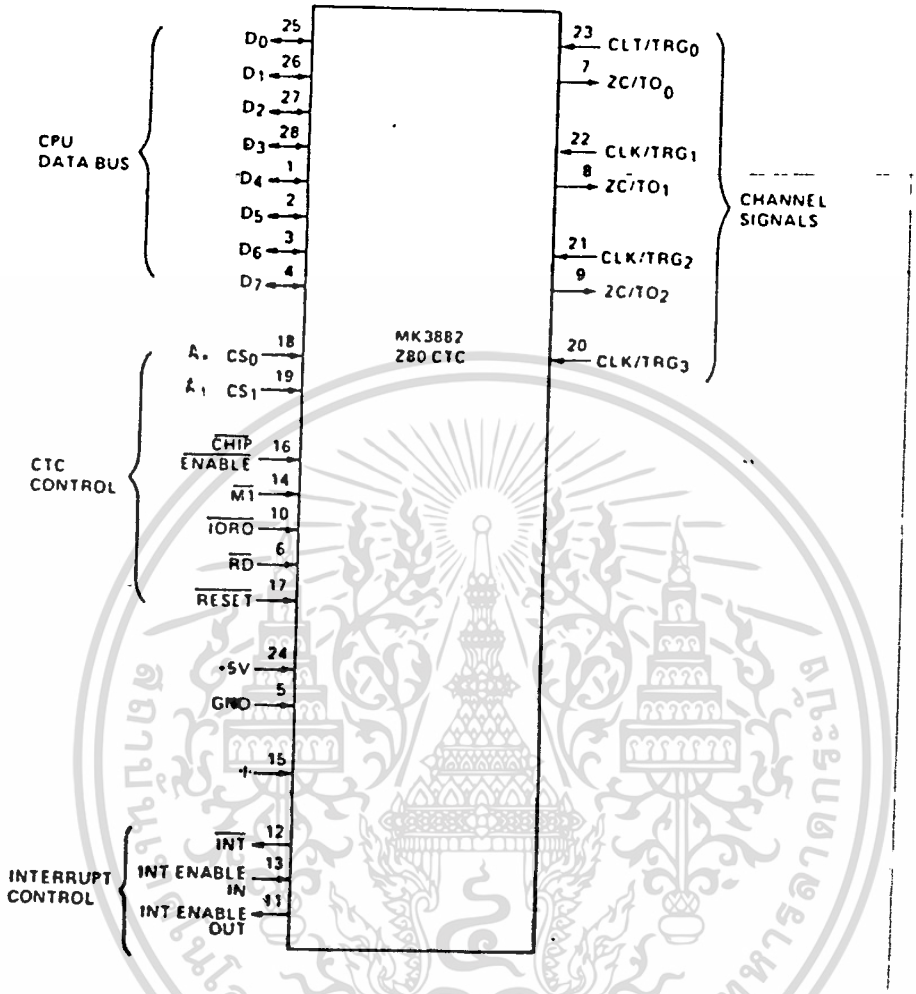
IEO (INTERRUPT ENABLE OUTPUT) : ขานี้เป็นขาเอาท์พุทและแอคทีฟที่ลอจิก "1" และเป็นขาที่ CTC ใช้แสดงสถานะของการอินเทอร์รัพท์ คือ เมื่อขานี้มีลอจิกเป็น "1" แชนแนลภายใน (INTERNAL CHANNEL) จะไม่สามารถขออินเทอร์รัพท์ต่อ CTC ได้อีก และขานี้ยังใช้ต่อกับขา IEI เพื่อทำ PRIORITY INTERRUPT DAISY CHAIN อย่างง่าย ๆ และมีประสิทธิภาพได้ด้วย

$\overline{\text{INT}}$: เป็นขาที่ให้สัญญาณการขออินเทอร์รัพท์ ลักษณะทางเอาท์พุทของขานี้เป็นแบบ OPEN DRAIN ดังนั้นสามารถทำการ WIRE ANDED กับขา $\overline{\text{INT}}$ OUTPUT ขาอื่น ๆในระบบได้

$\overline{\text{RESET}}$: เมื่อขานี้มีลอจิก "0" จะทำให้ CTC ถูก SET ให้อยู่ในสถานะหนึ่ง ซึ่งในสถานะนี้การนับในทุก ๆ แชลแนลจะหยุดหมด และบิททุกบิทที่ใช้ในการอินเทอร์รัพท์ใน CONTROL REGISTER ทั้งหมดจะถูกรีเซ็ททำให้ CTC ไม่สามารถรับการร้องขออินเตอร์รัพท์จากระบบอื่นได้นอกจากนั้นสาย OUTPUT, ZC/TO และ INT จะถูกทำให้อยู่ในสถานะ HIGH IMPEDANCE และ OUTPUT DRIVERS ของบัสข้อมูลก็จะถูกทำให้เป็นสถานะ HIGH IMPEDANECด้วย

CLK/TRG3-CLK/TRG0 : เป็นขาอินพุทที่ใช้รับสัญญาณ TRIGGER และคล็อกจากภายนอก

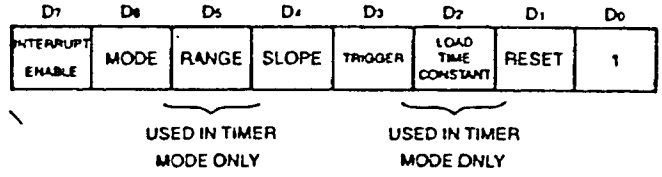
ZC/TO2-ZC/TO0 : เป็นขา ZERO COUNT/TIMEOUT ที่ใช้เป็นเอาท์พุท (ACTIVE HIGH)



รูปที่ 2.4.3 แสดงการจัดขาของ Z-80 CTC

2.4.4 การโปรแกรม CHANNEL CONTROL REGISTER

การโปรแกรม CHANNEL CONTROL REGISTER โดยทำการพิจารณา DATA BIT ของ CHANNEL CONTEROL REGISTER กันก่อน จะสังเกตได้ว่ามีบิทบางบิทที่ถูกกำหนดให้ใช้งานใน TIMER MODE โดยเฉพาะ



รูปที่ 2.4.4 แสดงรายละเอียดของบิทต่าง ๆ ใน CHANNEL CONTROL REGISTER ของ Z-80 CTC

จากรูปแสดงบิทต่าง ๆ ของ CHANNEL CONTROL REGISTER ซึ่งรายละเอียดของแต่ละบิทเป็นดังต่อไปนี้

D0 บิทนี้เป็นบิทที่ใช้แยกแยะคำสั่งควบคุม (CONTROL WORD) ออกจากข้อมูลอื่น ๆ โดยถ้าให้ D0 มีลอจิกเป็น "1" บิทต่าง ๆ จะประกอบกันเป็น CONTROL WORD

D1 (CHANNEL RESET INPUT) เมื่อให้บิทนี้มีลอจิกเป็น "1" จะทำให้การนับหรือ การแสดงฐานเวลา (TIMING)ของแชนแนลนั้น ๆ หยุดลง โดยบิททุกบิทในCHANNEL REGISTER ไม่มีการเปลี่ยนแปลง และการทำงานของแชนแนลจะกลับเป็นปกติเมื่อมีการโหลดค่า TIME XONSTANT เข้าไปใน TIME CONSTANT REGISTER

D2 ถ้าบิทนี้มีลอจิกเป็น "1" จะเป็นการบอกให้ CTC รู้ว่า WORD ที่ตามมาจะเป็นค่าของ TIME CONSTANT ที่ให้กับ TIME CONSTANT REGISTER ของแชนแนลนั้น ๆ

D3 บิทนี้จะถูกใช้งานในโหมด TIMER เท่านั้น โดยถ้าให้ลอจิกเป็น "1" จะเป็นการบ่งว่า สัญญาณทริกเกอร์จากภายนอก (EXTERNAL TRIGGER) จะเป็นสิ่งกระตุ้นให้ TIMER เริ่มทำงาน (RUN)และถ้าบิทนี้มีลอจิกเป็น "0" TIMER จะเริ่มทำงานทันทีที่มีการโหลด TIME CONSTANT REGISTER

D4 บิทนี้เป็นบิทที่กำหนดขอบของสัญญาเข้ามาที่ขา CLK/TRG ว่าขอบไหนเป็นขอบที่แอกทีฟ โดยกำหนดไว้ดังนี้

สำหรับ TIMER MODE : D4 = 1 ทำให้ TIMER เริ่มต้นทำงานที่ขอบขาขึ้น (POSITIVE EDGE) ของสัญญาเทริกเกอร์

D4 = 0 ทำให้ TIMER เริ่มต้นทำงานที่ขอบขาลง (NEGATIVE EDGE) ของสัญญาณทริกเกอร์

COUNTER MODE : D4 = 1 ทำให้เคาน์เตอร์นับสัญญาณจากภายนอกที่ขอบขาขึ้น (POSITIVE EDGE)

D4 = 0 ทำให้เคาน์เตอร์นับสัญญาณจากภายนอกที่ขอบขาลง (NEGATIVE EDGE)

D5 บิทนี้ถูกใช้ใน TIMER MODE เท่านั้น ใช้สำหรับเลือกตัวหารเริ่มแรกเพื่อหารคล็อกที่ป้อนเข้ามาด้วย 16 หรือ 256 โดยถ้าบิทนี้เป็นลอจิก "1" คล็อกจะถูกหารด้วย 256 และถ้าเป็นลอจิก "0" จะถูกหารด้วย 16

D6 ถ้าบิทนี้มีลอจิกเป็น "1" แชนแนลนี้จะถูกเลือกให้ทำงานในโหมดของเคาน์เตอร์ที่มีลักษณะการนับแบบนับลง (DOWN COUNTER) โดยสัญญาณคล็อกจะถูกป้อนเข้ามาทางขา CLK/TRG ของแชนแนล และถ้ามีลอจิกเป็น "0" การทำงานแชนแนลจะถูกเลือกให้อยู่ในโหมดของไทม์เมอร์ โดยมีสัญญาณคล็อกเป็นอินพุท และจะได้พัลส์ทางเอาท์พุทที่มีช่วงระยะเท่ากับ คาบสัญญาณคล็อกของระบบ×ค่าตัวหารเริ่มต้น (PRESCALE FACTOR) ที่มีค่าเป็น 16 หรือ 256×ค่า TIME CONSTANT

D7 เป็นบิทที่ใช้กับการอินเทอร์รัพท์ถ้าให้บิทนี้มีลอจิกเป็น "1" จะทำให้เคาน์เตอร์นับลงจนถึง 0 และถ้าบิทนี้มีลอจิกเป็น "0" จะไม่สามารถออินเทอร์รัพท์ได้

2.4.5 การโปรแกรม TIME CONSTANT REGISTER

แชนแนลต่าง ๆ ของ CTC จะไม่สามารถเริ่มทำงานได้จนกว่าจะมีการโหลดค่า TIME CONSTANE ให้กับ TIMECONSTANT REGISTER เสียก่อน โดยป้อนเข้าไปในแชนแนลหลังจากที่ป้อน WORD ที่ D2 ถูกเซ็ทให้เป็นลอจิก "1" เข้าไปแล้ว

ข้อมูลที่เขียนให้กับ TIME CONSTANT REGISTER จะมีค่าอยู่ระหว่าง 0 ถึง 255 ถ้าทุกบิทมีลอจิกเป็น "0" หมด จะทำให้การนับของเคาน์เตอร์ทำงานที่ 256 และถ้ามี TIME CONSTANT ค่าใหม่ถูกโหลดระหว่างที่เคาน์เตอร์อยู่ในขบวนการนับ ค่าใหม่นี้จะไม่ถูกใช้ จนกว่าเคาน์เตอร์จะนับถึง 0 เสียก่อน

โดยขออธิบายตามระบบคล็อกที่ใช้ ในระบบต้องการให้ซีทีซีป้อนความถี่ค่าออกมาเป็น 2400 H_Z จากความถี่ของคล็อกที่ใช้ในระบบที่ป้อนเข้ามามีค่า 1.79 MH_Z ซึ่งมีคาบประมาณจาก PRESCALER ของซีทีซีสามารถกำหนดค่าตัวหารความถี่ให้เป็น 16 หรือ 256 ในที่นี้ 16 จะได้ค่าไทม์คอนสแตนส์ค่า 46 หรือ 2E ในระบบเลขฐานสิบหก เริ่มแรกจะต้องเซ็ทคอนโทรลเวิลก์ก่อน โดยจะใช้แชนแนล 0 ของซีทีซี ค่าของบิทในคอนโทรลเวิลก์จะถูกเซ็ทดังต่อไปนี้

บิท 7 = 0 : ไม่มีการอินเทอร์รัพท์

บิท 6 = 0 : เป็นฟรีรันไทม์เมอร์

บิท 5 = 0 : ตั้ง PRESCALE FACTOR = 16

บิท 4 = 0 : ไม่สนใจ เพราะไม่ได้ใช้เป็นทริกเกอร์

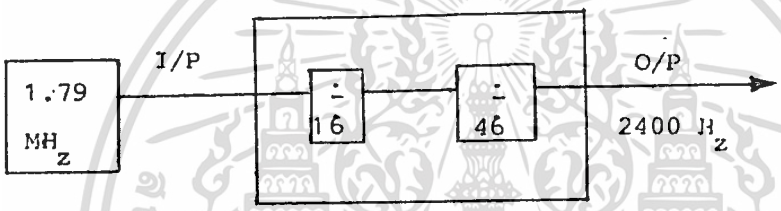
บิท 3 = 0 : ให้เริ่มต้นทำงานเมื่อไทม์เอาต์พุตถูกโหมด

บิท 2 = 1 : จะมีค่าไทม์เอาต์พุตก็ตามเหลังคอนโทรลเวิลด์

บิท 1 = 1 : แซนแนลถูกรีเซ็ท

บิท 0 = 1 : เป็นบิทที่กำหนดข้อมูลทั้งหมดเป็นคอนโทรลเวิลด์

โดยจากที่กล่าวมาซีทีซีจะมีการทำงานดังรูป

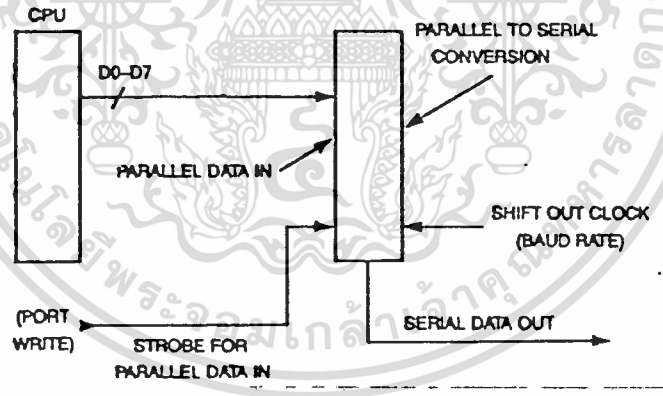


2.5 การรับส่งข้อมูล

การรับส่งข้อมูลจากระบบให้กับเครื่อง PC เพื่อใช้ในการประมวลผลในขั้นสุดท้ายนับว่ามีความสำคัญมากในการส่งข้อมูลเป็นระยะทางไกล จำเป็นจะต้องทำการเปลี่ยนข้อมูลที่รับได้แบบขนาน (PARALLET DATA) ไปเป็นการส่งข้อมูลแบบอนุกรม (SERIAL DATA) เพื่อที่จะป้องกันการลดทอนของสัญญาณให้ระยะทางไกล ส่งผลให้สัญญาณที่รับได้ปลายทางผิดเพี้ยนไปจากต้นทาง อีกทั้งยังเป็นการประหยัดค่าใช้จ่ายในการวางสาย โดยไม่จำเป็นได้อีกด้วย แต่ระบบนี้มีความยุ่งยากกว่าแบบขนานก็ตาม ในระบบใช้ IC แบอร์ 8251 เป็นตัวเปลี่ยนข้อมูล ซึ่งมีหลักการทำงานโดยทั่ว ๆ ไปดังนี้

2.5.1 การทำงานของ 8251 USART (UNIVERSAL SYNCHRONOUS/ASYNCHRONOUS RECEIVER /TRANSMITTER)

IC 8251 เป็นตัวเปลี่ยนรูปแบบของข้อมูลจากแบบขนานไปเป็นแบบอนุกรมก่อนที่จะทำการส่งข้อมูลออกไปตามสายส่ง มีหลักโดยทำการเก็บข้อมูลแบบขนานซึ่งมีจำนวน 8 บิทไว้ใน SHIFT REGISTER แล้วทำการค่อยเลื่อนข้อมูลทั้ง 8 บิทออกไปให้อุปกรณ์รับข้อมูลที่ละบิท โดยที่จะทำการส่งข้อมูลแต่ละบิทออกไปด้วยอัตราเดียวกับ BAUD RATE ที่ป้อนให้โดยจะส่ง D_0 เป็นบิทแรกและบิท D_7 เป็นบิทสุดท้าย



รูปที่ 2.5.1 บล็อกไดอะแกรมของการเปลี่ยนข้อมูลจากขนานเป็นอนุกรม

START BIT

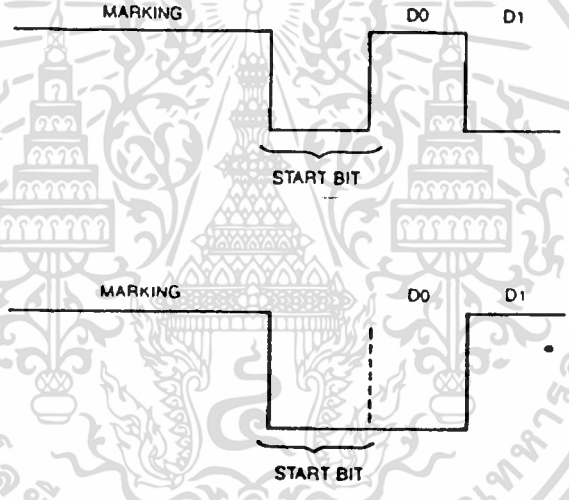
ในการส่งผ่านข้อมูลแบบอนุกรมนี้ จำเป็นที่จะต้องทำให้อุปกรณ์ที่จะรับข้อมูลทราบว่า ข้อมูลที่ส่งมานั้นเริ่มต้นที่จุดใด ดังนั้นจึงจำเป็นที่จะต้องเพิ่มข้อมูล 1 บิทลงไปก่อนหน้าข้อมูลจริง (ACTUAL

DATA) ที่จะทำการส่ง คือทำการเพิ่มบิทนี้ลงไปหน้าบิท D0 นั่นเอง และเรียกบิทนี้ว่า "START BIT"

หน้าที่ของ START BIT นี้นอกจากจะใช้ในการบอกว่าข้อมูลนั้นเริ่มต้นที่ใดแล้ว ยังทำงานร่วมกับ STOP BIT (ซึ่งจะกล่าวถึงต่อไป) เพื่อช่วยในการแยกข้อมูลแต่ละชุดออกจากกัน และความกว้างของบิทนี้จะเท่ากับความกว้างของบิทอื่น ๆ ในข้อมูลที่จะส่ง

เมื่ออุปกรณ์ที่จะส่งข้อมูลยังไม่ได้ทำการส่งข้อมูลใด ๆ ออกมานั้น สายส่งจะอยู่ในสภาวะที่เรียกว่า "MARKING" ซึ่งเป็นสภาวะที่ไม่มีการรับส่งข้อมูลใด ๆ เกิดขึ้นในที่นี้ MARKING ของสายส่งเป็นลอจิก "1" START BIT ที่จะเพิ่มเข้าไปนี้จะมีลอจิกที่ตรงข้ามกับลอจิกของ MARKING ดังนั้น START BIT จะมีลอจิกเป็น "0"

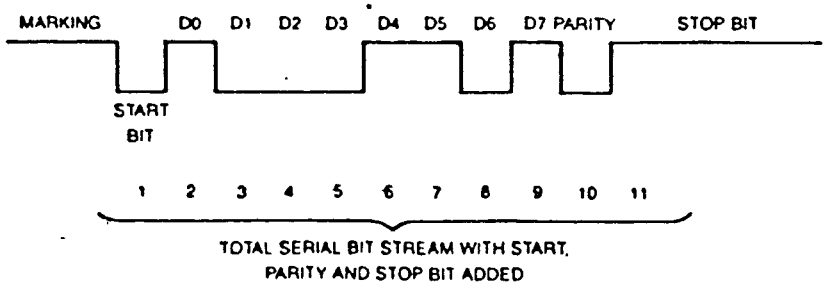
สำหรับ START BIT นี้จะมีความกว้างเท่ากับ 1 บิทของข้อมูลดังรูป



รูป การเพิ่ม START BIT เข้าไปก่อนหน้าบิท D0 ในกรณีที่บิท D0 เป็น "1" และ "0" ตามลำดับ

STOP BIT

เป็นบิทสุดท้ายที่เพิ่มเข้าไป จะใช้ในการตรวจสอบจุดสิ้นสุดของข้อมูล บิทนี้จะถูกเพิ่มเข้าไปหลัง PARITY BIT (ซึ่งจะไม่ขอกล่าวถึงเพราะไม่ได้ใช้) ถ้าอุปกรณ์รับข้อมูลตรวจไม่พบบิทนี้ก็จะแสดงว่าข้อมูลที่ได้รับเข้ามานั้นมีความผิดพลาดเกิดขึ้น สำหรับ STOP BIT นี้อาจมีจำนวนบิทเป็น 1, 1.5 หรือ 2 บิทก็ได้ จะแสดงข้อมูลทั้ง 8 บิทที่ส่งออกมารวมทั้ง START, STOP และ PARITY BIT ด้วย ซึ่งจะเห็นว่าสิ่งที่ส่งออกมาในแต่ละไบทนั้นไม่ได้มีเพียงข้อมูล 8 บิทเท่านั้น แต่อาจจะมีได้ถึง 12 บิท (กรณีที่ส่ง STOP BIT ออกมา 2 บิท)



รูปแบบของข้อมูลแต่ละไบท์ในการรับส่งข้อมูลแบบอนุกรม

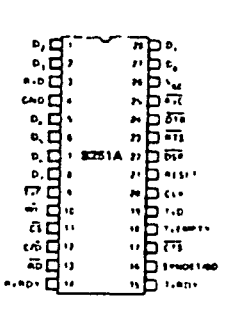
2.5.2 การเปลี่ยนข้อมูลจากแบบขนานเป็นข้อมูลแบบอนุกรม

โดยทั่วไปแล้ว การรับส่งข้อมูลภายในระบบมักจะเป็นการรับส่งข้อมูลแบบขนาน เนื่องจากมีความเร็วในการส่งผ่านข้อมูลที่สูงกว่าแบบอนุกรมมาก และยังมีความยุ่งยากน้อยกว่าอีกด้วย ดังนั้นในการรับส่งข้อมูลในระยะทางไกล ๆ ที่จำเป็นจะต้องใช้การส่งผ่านข้อมูลแบบอนุกรม จึงจำเป็นที่จะต้องทำการเปลี่ยนรูปแบบของข้อมูลจากแบบขนานไปเป็นแบบอนุกรมก่อนที่จะทำการส่งข้อมูลออกไปตามสายส่ง สำหรับหลักการง่าย ๆ ที่ใช้ในการเปลี่ยนรูปแบบของข้อมูลนั้นมีขั้นตอนดังต่อไปนี้ คือ

1. ทำการเก็บข้อมูลแบบขนาน (ในที่นี้มีจำนวน 8 บิท) ไว้ใน SHIFT REGISTER
2. เลื่อนข้อมูลทั้ง 8 บิทออกไปให้กับอุปกรณ์รับข้อมูลทีละบิท โดยที่จะทำการส่งข้อมูลแต่ละบิทออกไปด้วยอัตราเดียวกับ BAUD RATE ที่ได้กำหนดไว้

จากรูปจะแสดงบล็อกไดอะแกรมของการทำงานทั้ง 2 ขั้นตอน คือ ข้อมูลแบบขนานนั้นจะถูกส่งจาก CPU ให้กับ SHIFT REGISTER จากนั้นจึงทำการเลื่อนข้อมูลออกที่ละบิทด้วยอัตราของ BAUD RATE ที่กำหนด โดยเลื่อนบิท D0 ก่อน และ D7 เป็นบิทสุดท้าย

2.5.3 การจัดเรียงขาและหน้าที่ในการใช้งานในระบบ



รูปที่ 2.5.3 การจัดเรียงขาบน 8251

ไอซี 8251 เป็นไอซีที่มี 28 ขา สามารถแบ่งขาของไอซีตามลักษณะการนำไปใช้งานได้เป็นกลุ่ม ๆ ดังนี้ คือ

กลุ่มที่ใช้ติดต่อกับซีพียู

D0-D7 : ใช้ในการติดต่อกับ DATA BUS ของ CPU โดยตรง ซึ่งจะทำหน้าที่ในการรับส่งข้อมูลและคำสั่งต่าง ๆ ระหว่าง 8251 กับ ซีพียู

$\overline{\text{RESET}}$: 8251 จะถูกรีเซ็ทเมื่อขานี้ได้รับลอจิก "1" ซึ่งอาจจะต่อมาจากขา $\overline{\text{RESET}}$ ของ Z80 โดยผ่าน INVERTET ก่อนก็ได้

CLK (CLOCK) : ใช้ในการควบคุมช่วงเวลาการทำงานภายในของ 8251 สำหรับการใช้งานนั้นจะต่อเข้าโดยตรงกับระบบ อย่างไรก็ตามสัญญาเตขา CLK นี้ไม่เกี่ยวข้องกับ อัตราการรับส่งข้อมูลหรือ BAUD RATE แต่อย่างใด

$\overline{\text{RD}}$: เมื่อขานี้ได้รับลอจิก "0" 8251 จะทำการส่งข้อมูลแบบขนานออกมาที่ DATA BUS เพื่อส่งให้กับ ซีพียู

$\overline{\text{WR}}$: เมื่อขานี้ได้รับลอจิก "0" 8251 จะทำการรับข้อมูลแบบขนานจาก DATA BUS ของระบบ

C/$\overline{\text{D}}$ (CONTROL/DATA) : ขา C/$\overline{\text{D}}$ นี้จะใช้ในการทำให้ 8251 ทราบว่า CPUต้องการที่จะติดต่อกับ CONTROL REGISTER หรือ DATA REGISTER โดยที่ถ้าขานี้ได้รับลอจิก "1" ก็แสดงว่า CPU ต้องการที่จะติดต่อกับ CONTROL REGISTER แต่ถ้าได้รับลอจิก "0" ก็แสดงว่า CPU ต้องการที่จะติดต่อกับ DATA REGISTER

$\overline{\text{CS}}$ (CHIP SELECT) : ในกรณีที่ขานี้ได้รับลอจิก "0" ก็จะเป็นการ ENABLE 8251 โดยทั่วไปแล้วสัญญาเตขานี้จะได้มาจากการถอดรหัสแอดเดรส ดังที่ใช้กับ CHIP SUPPORT อื่นๆ

กลุ่มที่ใช้ในการติดต่อกับ MODEM

$\overline{\text{DSR}}$ (DATA SET READY) : ขานี้เป็นขาที่ใช้ในการรับสัญญาณจากอุปกรณ์ภายนอก ซึ่ง CPU สามารถที่จะตรวจสอบสัญญาณที่ขานี้ได้ โดยการอ่านค่าในรีจิสเตอร์สถานะ และระดับของสัญญาณที่ขานี้จะใช้ในการแสดงว่าอุปกรณ์ภายนอกพร้อมที่จะทำการติดต่อด้วยหรือยัง

$\overline{\text{DTR}}$ (DATA TERMINAL READY) : ขานี้เป็นเอาท์พุทที่ใช้ในการบอกให้อุปกรณ์ภายนอกทราบว่า CPU พร้อมที่จะทำการติดต่อด้วย

$\overline{\text{CTS}}$ (CLEAR TO SEND) : ขานี้เป็นขาอินพุทที่ใช้ในการทำให้ 8251 เริ่มทำการส่งข้อมูลได้ สิ่งที่ต้องระวังในการใช้งานขานี้ก็คือ เมื่อไม่ได้ใช้งานขานี้จะต้องถูกต่อเข้ากับลอจิก "0" ถ้าไม่เช่นนั้น 8251 จะทำการส่งข้อมูลไม่ได้

$\overline{\text{RTS}}$ (READY TO SEND) : ขานี้เป็นเอาท์พุทที่ CPU จะเป็นผู้ควบคุมสัญญาณที่ขานี้เอง (ขา $\overline{\text{DTR}}$ ก็ถูกควบคุมโดย CPU เช่นกัน)

กลุ่มที่ใช้ในการส่งข้อมูล

T×D (TRANSMIT DATA OUTPUT) : เป็นขาที่ใช้ในการส่งข้อมูลไปตามสายส่ง

T×C (TRANSMIT BAUD RATE CLOCK) : ขานี้เป็นขาที่ใช้ในการส่งสัญญาณคล็อกที่ใช้ในการส่งข้อมูล ซึ่งก็คือความถี่ที่ใช้ในการกำหนด BAUD RATE นั่นเอง โดยปกติแล้วจะต้องช้ากว่าสัญญาณคล็อกของระบบไม่น้อยกว่า 30 เท่า

T×RDY : ขานี้จะใช้ในการทำให้ CPU ทราบว่า 8251 พร้อมที่จะรับข้อมูลจาก CPU เพื่อที่จะทำการส่งต่อไปแล้วหรือยัง และขานี้อาจจะนำไปใช้ในการขออินเทอรัพท์ก็ได้

T×EMPTY : ขานี้จะใช้ในการแสดงว่าข้อมูลที่ CPU ส่งให้กับ 8251 นั้นได้ถูกส่งออกไปให้อุปกรณ์อื่น ๆ หมดแล้ว โดยที่ 8251 จะทำให้ขานี้เป็น "1" และเมื่อ CPU ทำการส่งข้อมูลชุดต่อไปให้กับ 8251 ขา T×EMPTY ก็จะเป็น "0" จนกว่า 8251 จะทำการส่งข้อมูลนี้ออกไปหมด 8251 ก็จะทำให้ขานี้กลับเป็น "0" อีกครั้ง

กลุ่มที่ใช้ในการรับข้อมูล

R×D : ใช้ในการรับข้อมูลแบบอนุกรมจากสายส่ง

R×C : เป็นขาที่ใช้ในการรับสัญญาณคล็อกที่ใช้ในการรับข้อมูล โดยเหตุผลแล้วจะทำการต่อเข้ากับ T×C โดยตรง

R×RDY : จะใช้ในการแสดงว่า 8251 พร้อมที่จะส่งข้อมูลให้กับ CPU และขานี้อาจจะใช้ในการขออินเทอร์รัพท์ได้เช่นเดียวกับขา T×RDY

SYNDET : ขานี้จะใช้ในการรับข้อมูลแบบ SYNCHRONOUS เท่านั้น โดยที่เราสามารถที่จะโปรแกรมให้ขานี้เป็นอินพุทหรือเอาท์พุทก็ได้ โดยที่เมื่อขาSYNDET นี้ถูกโปรแกรมเป็นเอาท์พุทนั้น ขา SYNDET จะให้ลอจิก "1" เมื่อ 8251 สามารถที่จะตรวจจับ SYNC CHARACTER ได้ และจะให้ลอจิก "0" เมื่อ CPU ทำการอ่านรีจิสเตอร์สถานะสำหรับขา SYNDET นี้จะให้ลอจิก "1" ในอีกกรณีหนึ่งคือ เมื่อ 8251 ได้รับข้อมูลจากสายส่งเป็น "0" หมดตั้งแต่ START BIT จนถึง STOP BIT

ในกรณีที่ขา SYNDET ถูกโปรแกรมให้เป็นอินพุทนั้น ถ้าขานี้ได้รับสัญญาณขึ้น (สัญญาณเปลี่ยนจากลอจิก "0" เป็น "1") 8251 ก็จะถือว่าข้อมูลที่ขา R×D เป็นข้อมูลทันที และสามารถที่จะทำให้ลอจิกที่ขานี้กลับเป็น "0" ได้ในสัญญาณ R×D ลูกต่อไป

กลุ่มไฟเลี้ยงของ 8251

8251 ใช้ไฟเลี้ยงเพียงชุดเดียว คือ +5V กับ GND เท่านั้น ดังนั้นขาไฟเลี้ยงของ 8251 จึงมีเพียง 2 ขา คือ VCC กับ GND

2.5.4 การโปรแกรม 8251

การรับส่งข้อมูลบน 8251 ที่ใช้มีอยู่ด้วยกัน 2 โหมด คือ โหมดอะซิงโครนัส และโหมดซิงโครนัส (ASYNCHRONOUS AND SYNCHRONOUS MODE) ในที่นี้จะขอกล่าวเฉพาะแต่โหมดแรก ซึ่งเป็นโหมดที่ระบบใช้อยู่

จากรูปแสดงการจัดเรียงบิทในคอนโทรลเวิลด์ในโหมดอะซิงโครนัส ซึ่งโหมดเวิลด์นี้จะถูกส่งให้กับรีจิสเตอร์ควบคุมโดยการอ้างถึงพอร์ทแอดเดรสพร้อมกับการให้สัญญาณ $\overline{WR}$ กับ 8251 ซึ่งสามารถที่จะทำได้โดยการใช้คำสั่ง OUT โดยที่ค่าในรีจิสเตอร์ A เป็นค่าของ MODE WORD และ 8251 จะถือว่าข้อมูลที่ส่งให้กับรีจิสเตอร์ควบคุมเป็นโหมดเวิลด์ในกรณีที่ข้อมูลนี้เป็นข้อมูลไบท์แรกที่ถูกส่งให้กับ 8251 หลังจากที่ถูกรีเซ็ท

ในที่นี้จะสมมติว่า 8251 ได้รับการรีเซ็ทแล้วและจะทำการโปรแกรมให้ทำงานในโหมดอะซิงโครนัส (บิท 0 และ 1 จะกำหนดว่าจะให้ 8251 ทำงานในโหมดใด ในกรณีที่บิททั้งสองนี้เป็น "0" ทั้งคู่เท่านั้น จึงจะเป็นการเลือกให้ 8251 ทำงานในโหมดซิงโครนัส ดังนั้นจึงต้องทำการอ้างถึงพอร์ท กำหนดการทำงานของ 8251 ในระบบ ดังนี้

1. กำหนดให้ใช้ค่า BAUD RATE เท่ากับ 1200 BAUD ในกรณีที่ใช้ความถี่ของคล็อกที่ขา T×C และ R×C เท่ากับ BAUD RATE พอดี ก็เลือกโปรแกรมให้บิท 0 และ 1 เป็น "1" และ "0" ตามลำดับ แต่ในกรณีที่ความถี่ของคล็อกที่ใช้มีค่ามากกว่า BAUD RATE ที่ใช้ 16 หรือ 64 เท่า ก็จะต้องทำการโปรแกรมให้ BAUD RATE FACTOR เป็น 16× หรือ 64× ตามลำดับ

2. กำหนดให้ข้อมูลที่จะส่งออกไปมีจำนวน 8 บิท (บิท 2 และ 3 เป็น "1") ซึ่งอาจจะโปรแกรมให้เป็น 5,6 หรือ 7 บิทก็ได้ ในกรณีที่ข้อมูลที่ต้องการที่จะส่งออกไปมีน้อยกว่า 8 บิท 8251 จะทำการตัดบิทสูงทิ้งไป เช่น ถ้าเลือกให้ส่งเพียง 5 บิท 8251 ก็จะทำการตัดบิท D7-D5 ทิ้ง

3. ไม่ใช้ PARITY BIT (บิท 4 และ 5 เป็น "0")

4. เลือกใช้ STOP BIT จำนวน 2 บิท (บิท 7 และ 6 เป็น "1")

ดังนั้นข้อมูลที่จะทำการส่งให้กับพอร์ท 7DH จะเป็น 11001101 หรือ OCDH สำหรับชุดคำสั่งของ Z80 ที่จะใช้ในการโปรแกรมสั่งงาน 8251 แสดงได้ดังนี้

```
LD A, OCDH
OUT (CCH), A
```

หลังจากที่ได้ส่ง MODE WORD ให้กับ 8251 แล้ว ข้อมูลไบท์ต่อไปที่ถูกส่งให้กับรีจิสเตอร์ควบคุมจะถือว่าเป็น COMMAND WORD ทั้งสิ้น จนกว่า 8251 จะได้รับการรีเซ็ทอีก จึงจะถือว่าข้อมูลที่ส่งมาให้กับรีจิสเตอร์ควบคุมนั้นเป็น MODE WORD

ในที่นี้กำหนดให้ 8251 ทำการรับและส่งข้อมูลได้ (บิท 0 และ 2 เป็น "1") ทำการ

รีเซ็ท ERROR FLAG ในรีจิสเตอร์สถานะคือPE ,FE และ OE สำหรับการทำ INTERNAL RESET (IR) นั้นสามารถที่จะทำได้ โดยการทำให้บิท 6 (IR) ของ COMMAND WORD เป็น "1" แต่ในที่นี้จะยังไม่ทำ INTERNAL RESET ดังนั้น COMMAND WORD ที่จะส่งให้กับ 8251 ก็คือ 00010101B หรือ 15H สำหรับการส่ง CONNAND WORD นี้ให้กับรีจิสเตอร์ควบคุมของ 8251 นั้นสามารถที่จะทำได้โดยใช้วิธีเดียวกับการส่ง MODE WORD คือส่งออกไปที่พอร์ท CCH

2.5.5 การเชื่อมต่อกับสายส่งข้อมูล

ระดับสัญญาณที่ใช้ในวงจร (+5v) ไม่สามารถที่จะส่งไปได้ไกลนัก ก่อนที่จะส่งข้อมูลไปในสายส่งจึงต้องเปลี่ยนระดับแรงดันของสัญญาณใหม่ เพื่อให้สามารถที่จะส่งสัญญาณได้ไกลขึ้นในระบบใช้ตามมาตรฐานการรับ+ส่งข้อมูล RS-232 ซึ่งเป็นที่นิยมในปัจจุบัน

ในมาตรฐาน RS-232 จะใช้ระดับแรงดันในสายส่งประมาณ +/- 12 VOLTS โดยเปลี่ยนระดับแรงดันของสัญญาณจากระดับสัญญาณที่ใช้กับอุปกรณ์พวก TTL ไปเป็นระดับสัญญาณที่ใช้กับมาตราฐาน RS-232 ในการส่งข้อมูลอุปกรณ์ที่ใช้ในการทำหน้าที่นี้คือ IC เบอร์ MC 1488 จากรูปขา $\overline{\text{CTS}}$ จะถูกต่อกับลอจิก "0" ด้วย จึงทำการส่งเพียง 2 เส้น-คือ T×D และ GND เท่านั้น

บทที่ 3

การออกแบบและคำนวณ

จากบทที่แล้วเป็นการอธิบายรายละเอียดของการทำงานของไอซีเบอร์ต่าง ๆ ที่ใช้ในระบบ สำหรับในระบบนี้จะขออธิบายถึงการออกแบบและการคำนวณนำมาใช้ในระบบ ซึ่งถูกสร้างขึ้นเพื่อใช้ในงานจุดประสงค์นี้ ซึ่งมีรายละเอียดดังต่อไปนี้

ในระบบที่สร้างขึ้นไว้รองรับการใช้งานสำหรับ 2 ราง การทำงานจะเริ่มขึ้นตั้งแต่เมื่อจ่ายไฟเลี้ยงให้เลี้ยงให้ระบบ ระบบในภาคประมวลผลจะทำการอ่านค่าสัญญาณจากวงจรสร้างพัลส์ที่ได้จากเซนเซอร์ตรวจจับลูกโบว์ลิ่งอยู่ตลอดเวลา ซึ่งในขณะที่ไม่มีลูกผ่านสัญญาณจะมีสถานเป็น "0" ตลอดเมื่อมีลูกผ่านสัญญาณจะเปลี่ยนจาก "0" เป็น "1" จากวงจรสร้างพัลส์ จะมีผลทำให้ซีพียูทำการอ่านค่าโค้ดของราง และสถานะของพินทั้งสิบในรางนำมาประมวลผลเป็นรหัสแอสกี้ 2 ไบท์ส่งไปไปยังเครื่องคอมพิวเตอร์ต่อไป โดยในไบท์แรกที่ส่งจะส่งหมายเลขโค้ดของราง และไบท์ที่สองจะเป็นข้อมูลของพินโบว์ลิ่งที่ล้มลง โดยการทำงานต่าง ๆ เป็นลำดับอย่างนี้จะถูกควบคุมด้วยโปรแกรมการทำงาน ซึ่งรายละเอียดของวงจรในส่วนต่าง ๆ มีดังนี้

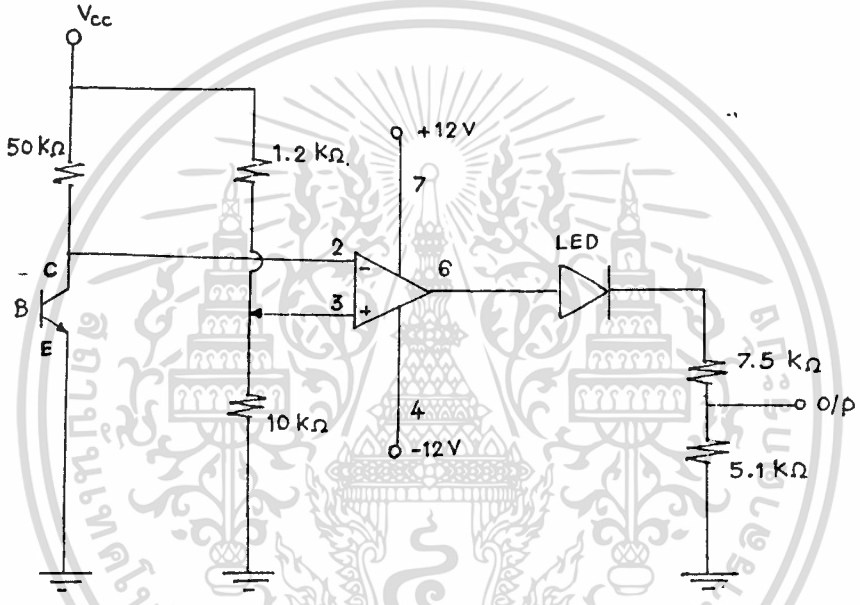
3.1 ตัวตรวจจับ (SENSOR)

วงจรส่วนนี้มี 2 แบบ

3.1.1 ตัวตรวจจับลูกโบว์ลิ่ง

ตัวตรวจจับพินโบว์ลิ่ง

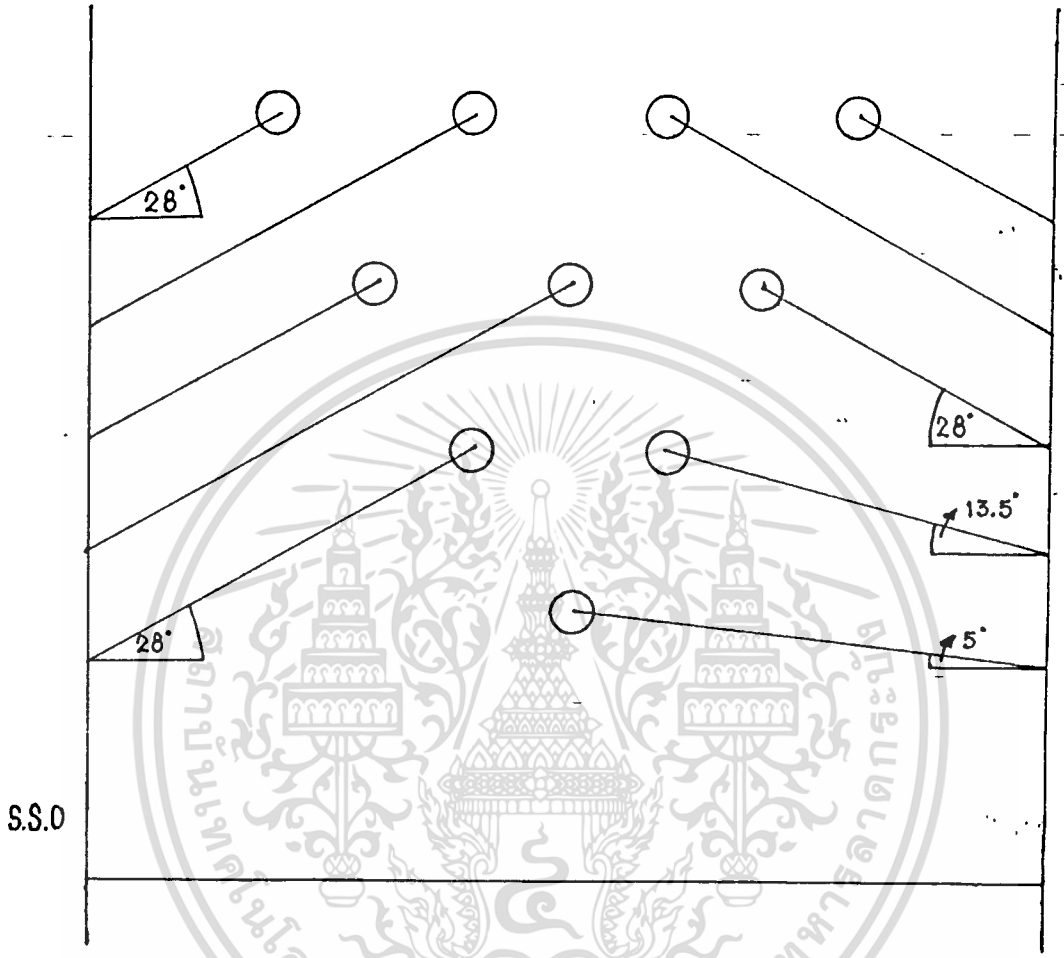
ตัวตรวจจับ ทั้งสองแบบนี้เหมือนกันเกือบทุกส่วน แตกต่างตรงที่ตัวตรวจจับลูกโบว์ลิ่งสัญญาณเข้าจะเข้า (INPUT) ที่ขา 3 และระดับแรงดันไฟฟ้าอ้างอิงจะเข้าที่ขา 2 แต่ตัวตรวจจับพินโบว์ลิ่งจะกลับตรงข้ามกัน คือ สัญญาณเข้าที่ขา 2 และระดับแรงดันไฟฟ้าอ้างอิงจะเข้าที่ขา 3 ดังรูป



รูปที่ 3.1.1 แสดงวงจรตรวจจับพินโบว์ลิ่งโดยใช้ตัวรับแสงอินฟราเรด

การตรวจจับพินโบว์ลิ่งว่าพินไหนล้มข้าง เราใช้โฟโต้ทรานซิสเตอร์เบอร์MRD 370 เป็นตัวรับแสงแล้วเปลี่ยนเป็นสัญญาณไฟฟ้า เพื่อป้อนให้แก่วงจรเปรียบเทียบระดับสัญญาณ ซึ่งประกอบด้วยออปแอมป์เบอร์ LF.351 ทำหน้าที่เปรียบเทียบระดับสัญญาณอินพุทกับศักดาอ้างอิงที่กำหนดไว้จากความต้านทานแบบปรับค่าได้ที่ขา 3 ของIC เบอร์ LF.351 ถ้าระดับสัญญาณอินพุทที่ขา 2 ต่ำกว่าศักดาอ้างอิงที่ขา 3 จะทำให้ให้เอาท์พุทที่ขา 6 เป็นระดับสัญญาณไฟตรงประมาณ +12 โวลต์ แล้วทำการแบ่งโวลต์เตดให้ได้เอาท์พุทประมาณ 5 โวลต์ โดยการต่อตัวต้านทาน ขา 7.5kΩ และ 5.1kΩ ดังรูป ในทางตรงกันข้ามถ้าระดับสัญญาณอินพุทที่ขา 2 สูงกว่าศักดาอ้างอิงที่ขา 3 จะทำให้ให้เอาท์พุทที่ขา 6 เป็นระดับสัญญาณไฟตรงประมาณ -12 โวลต์

การออกแบบการวางตัวตรวจจับเพื่อตรวจจับพินโบว์ลิ่ง

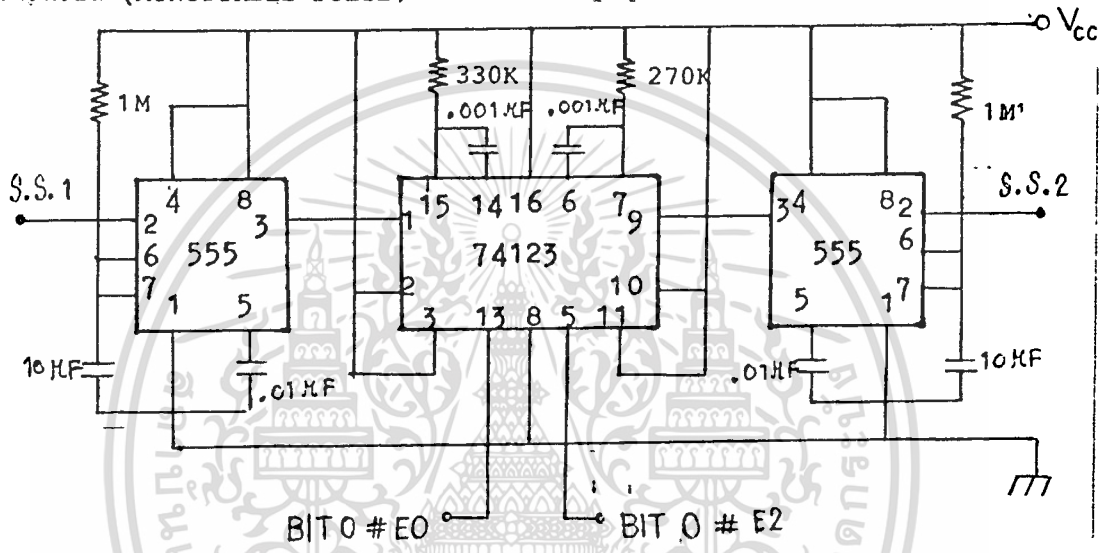


รูปที่ 3.1.1-ข - แสดงการวางตำแหน่งตัวตรวจจับเพื่อตรวจจับพินโบว์ลิ่ง

ในการตรวจจับว่าพินโบว์ลิ่งอันไหนล้มบ้าง จะทำการตรวจจับหลังจากที่ตัวตรวจจับลูกโบว์ลิ่งทำการตรวจจับได้ว่ามีลูกโบว์ลิ่งผ่านมาแล้วเป็นเวลา 5 วินาที การตรวจจับว่าพินโบว์ลิ่งพินไหนล้มบ้าง จะใช้ตัวรับแสงอินฟราเรด 1 ชุดต่อหนึ่งพินโดยการตรวจจับนี้ต้องทำการวางตำแหน่งของตัวรับแสงอินฟราเรดในทิศทางที่จะรับแสงที่สะท้อนจากพินโบว์ลิ่งลูกที่ตัวตรวจจับตัวนั้นต้องการตรวจจับให้แน่นอน ซึ่งต้องทำการวัดมุมที่จะวางตัวตัวตรวจจับดังในรูปเพื่อให้ตัวรับแสงอินฟราเรดรับแสงได้มากที่สุด และเพื่อป้องกันการรบกวนเนื่องจากแสงอินฟราเรดที่สะท้อนจากพินโบว์ลิ่งพินอื่นเข้าตัวรับแสงอีกด้วยดังแสดงในรูปที่ 3.1.2

3.1.2 วงจรหน่วงเวลาและสร้างพัลซ์ (PULSE)

ในขั้นแรกหลังจากผู้เล่นได้ทำการเริ่มเล่น ไฟเลี้ยงจะถูกจ่ายไปยังวงจรส่วนย่อยต่าง ๆ ของระบบทั้งหมดเครื่องก็จะเริ่มทำงาน หลังจากผู้เล่นได้เริ่มทำการโยนลูกโบว์ลิ่งวิ่งตรงในรางก่อนที่จะชนเข้ากับพินทั้งสิบของโบว์ลิ่งที่ตั้งอยู่ ลูกโบว์ลิ่งก็จะถูกตรวจจับจากเซนเซอร์ตรวจจับลูกโบว์ลิ่งผ่านซึ่งจะถูกติดตั้งอยู่ในรางเพื่อตรวจจับลูกโบว์ลิ่งก่อนที่ตั้งของพินแรกสัญญาณจะถูกส่งไปยังวงจรส่วนหน่วงเวลา ซึ่งประกอบด้วยไอซีเบอร์ 555 ทำหน้าที่หน่วงเวลา และไอซีเบอร์ 74123 ทำหน้าที่สร้างพัลซ์โมโนสเตเบิ้ล (MONOSTABLE PULSE)



รูปที่ 3.1.2 วงจรหน่วงเวลาและสร้างพัลซ์

จากรูปไอซีเบอร์ 555 จะตรวจจับสถานะของสัญญาณจากเซนเซอร์ที่เข้ามาบังขา 2 ถ้าสัญญาณที่รับได้มีค่าต่ำกว่าประมาณ $\frac{1}{3} V_{cc}$ ซึ่งเกิดขึ้นในกรณีเซนเซอร์ตรวจจับได้ว่ามีลูกผ่านก็จะสร้างสัญญาณสถานะ HIGH ออกมาเป็นระยะเวลาประมาณ 5S สามารถคำนวณได้จากค่า R และ C ที่ใช้ในวงจร ซึ่งจะได้คาบของสัญญาณประมาณ 1.1 RC

จากวงจร R = 1M และ C = 4.7 uF

จะได้ $t = 1.1 \times 1 \times 10^{6} \times 4.7 \times 10^{-6}$ S

$= 517$ S

จากที่กล่าวมาเป็นวงจรนอนรีทริกซ์(NON-RETRIG)เมื่อมีการทริกซ์ขึ้นในช่วงสร้าง

สัญญาณสถานะ HIGH อยู่จะไม่มีผลต่อการทำงานของวงจร จากนั้นสัญญาณ HIGH ก็จะถูกตรวจจับสัญญาณขอบขาลงจากไอซีเบอร์ 74123 ซึ่งเป็น DUAL RETRIGGERABLE MONOSTABLE MULTIVIBRATOR WITH CLEAR IC ก็จะสร้างพัลซ์ไมโนสเตเบิ้ลขึ้นมีช่วงพัลซ์กว้างอยู่ในช่วงการทำงานของซีพียูที่สามารถอ่านค่าได้เพียงรอบเดียวจะถูกกำหนดโดยโปรแกรมที่ใช้ควบคุมการทำงานนำมาสร้างวงจร

จากวงจรเราใช้ R = 270 K และ C = 0.001 uF

จะได้ มีค่าประมาณ $t = 0.29\ RC$

$= 0.29 \times 270 \times 10^{3} \times 0.001 \times 10^{-6}\ S$

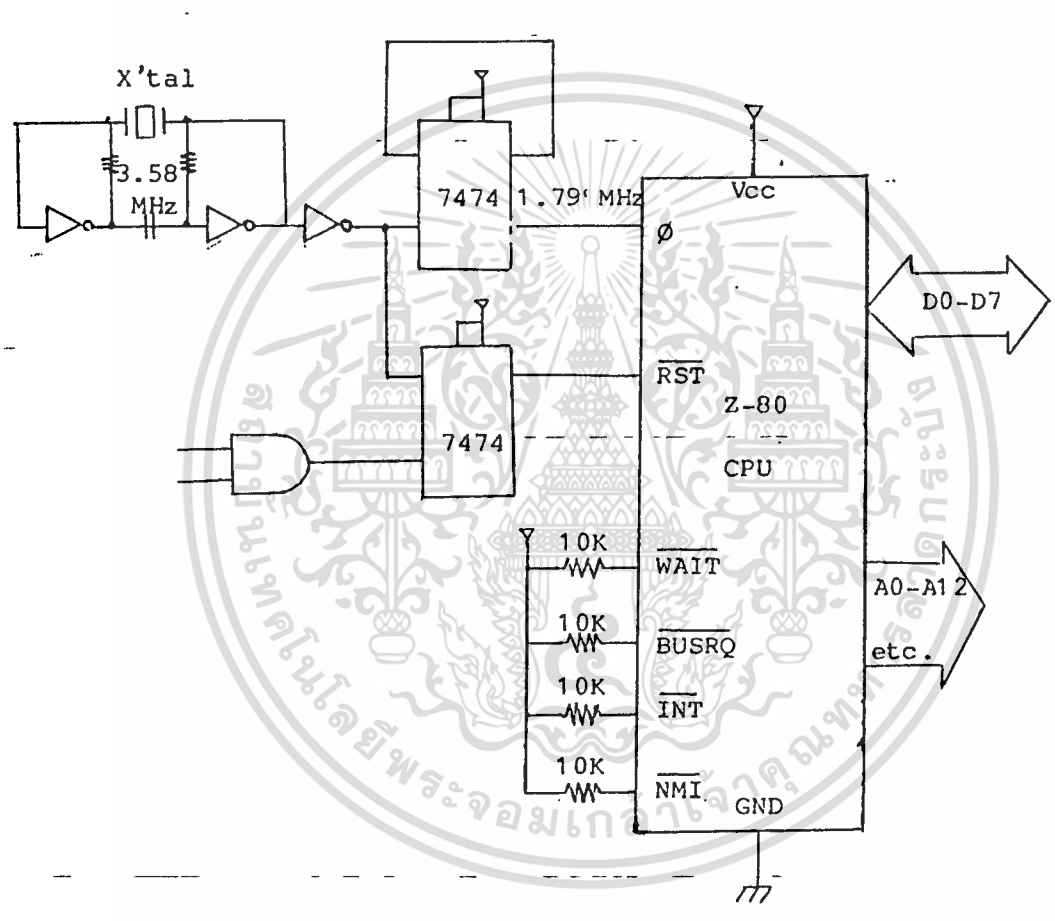
$= 7.83\ 10^{-5}\ S$

หรือมีค่าเท่ากับ 80uS โดยประมาณสำหรับรางแรกและประมาณ 100 uS สำหรับในรางที่สองเพื่อที่จะแก้ปัญหาในกรณีเกิดการเล่นพร้อมกันทั้งสองราง ลูกโบว์ลิ่งอาจจะถูกตรวจจับลูกผ่านในเวลาพร้อมกันสามารถทำให้ซีพียูรับค่าเข้าไปได้ทั้งสองค่า ซึ่งกรณีนี้โอกาสจะเกิดขึ้นเป็นไปได้น้อยมาก สัญญาณพัลซ์ไมโนสเตเบิ้ลที่ถูกสร้างขึ้นจะถูกส่งต่อไปยังภาควงจรไมโครโปรเซสเซอร์ต่อไป

3.2 ภาคประมวลผล

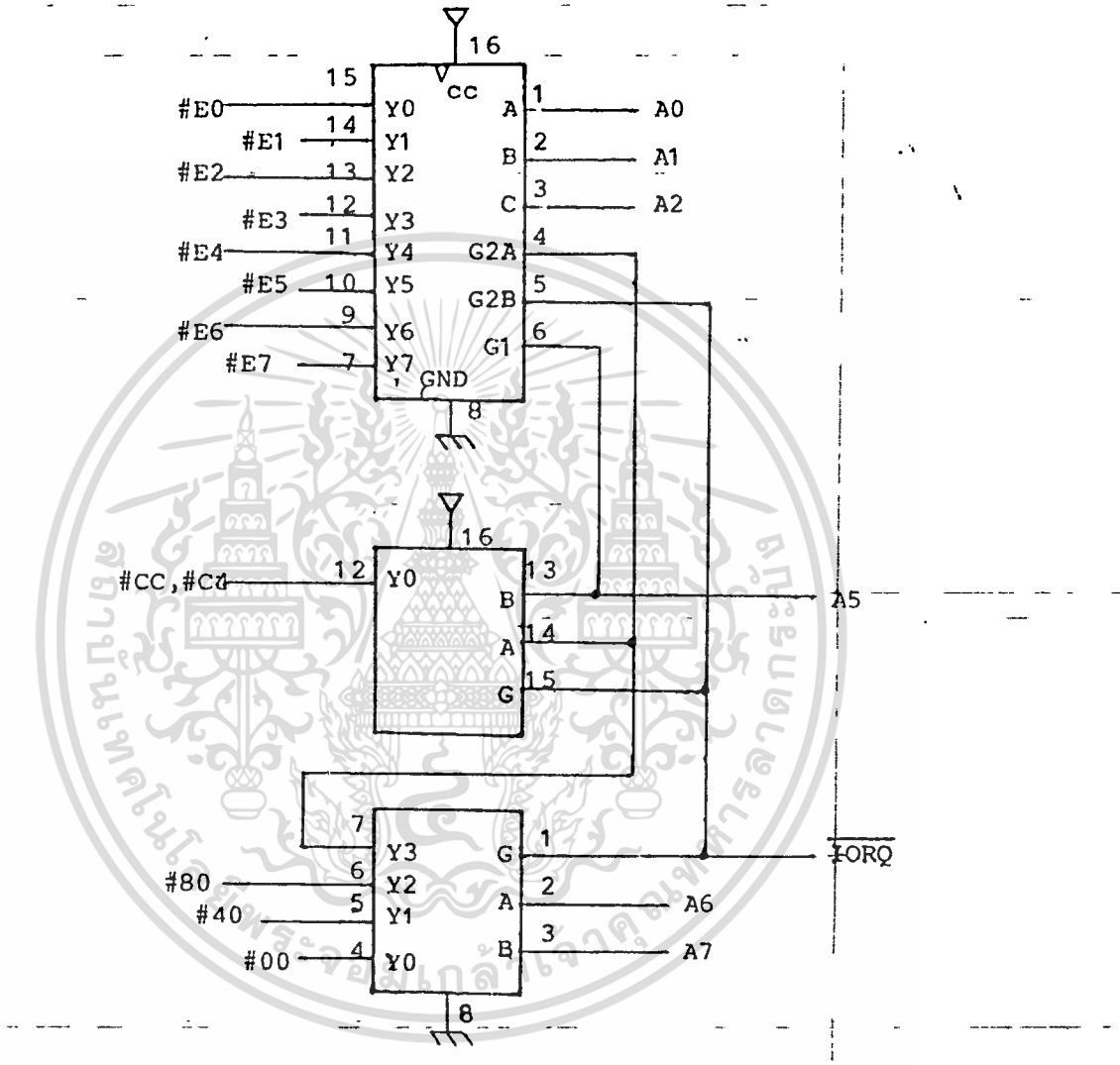
3.2.1 หน่วยประมวลผล

จากที่กล่าวในบทก่อนในระบบใช้ Z-80 ซีพียูเป็นหน่วยประมวลผล โดยที่ขาสัญญาเต้งอินพุทและเอาท์พุทของซีพียูต่อเข้ากับส่วนต่าง ๆ ในระบบดังรูป ขาคล็อกของระบบต่อเข้ากับวงจรสร้างคล็อกซึ่งใช้ X'tal-3.58 MH_z ผ่านอินเวอร์เตอร์ชับกระแสเข้าไปยัง D-FLIP FLOP เพื่อสร้างสัญญาเคล็อกที่มีความถี่สัญญาณ 1.79 MH_z ป้อนให้กับระบบ -



รูปที่ 3.2.1 แสดงหน่วยประมวลผลและวงจรคล็อกของระบบ

นอกจากนี้ระบบยังใช้ขาของแอดเดรส A_0-A_7 และขาสัญญาณ $\overline{IORQ}$ เป็นตัวเลือกพอร์ท เพื่อทำการกำหนดการช่วงเวลาการทำงานของอุปกรณ์ ซึ่งการออกแบบและต่อวงจร เพื่อทำการเลือกพอร์ทได้แสดงไว้ดังรูป



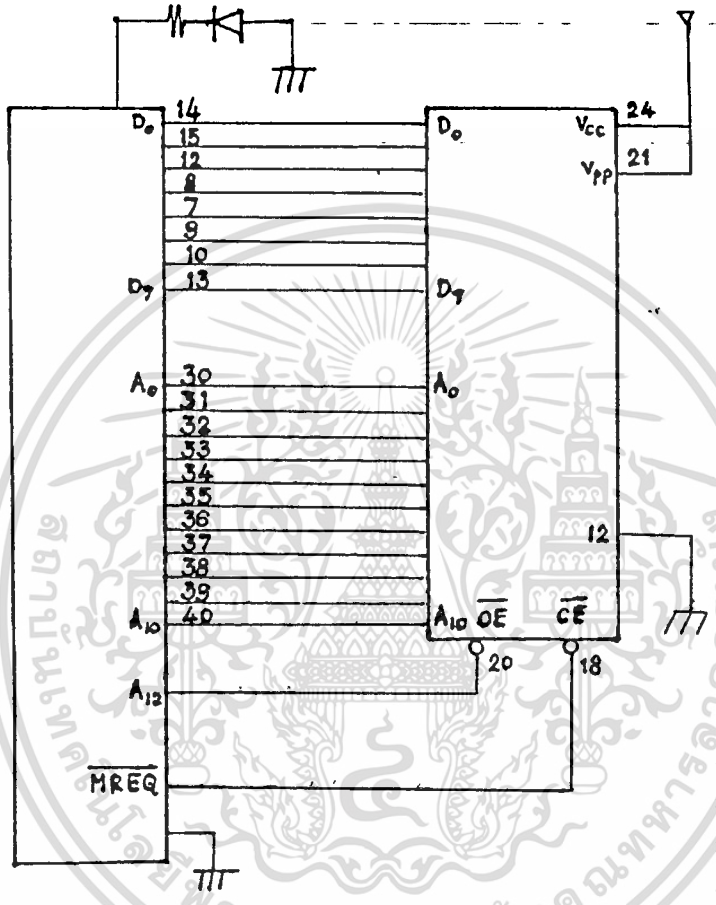
รูป 3.2.18 แสดงวงจรถอดรหัส

ซึ่งมีผังแอดเดรสการทำงานตามตาราง

A7	A6	A5	A4	A3	A2	A1	A0			
0	0	0	0	0	0	0	0	#00	:	EXT. PORT
0	1	0	0	0	0	0	0	#40	:	CTC
1	0	0	0	0	0	0	0	#80	:	EXT. PORT
1	1	0	0	1	X	0	0	#CC, C4	:	8251
1	1	1	0	0	0	0	0	#E0	:	I/P PORT
1	1	1	0	0	0	0	1	#E1	:	I/P PORT
1	1	1	0	0	0	1	0	#E2	:	I/P PORT
1	1	1	0	0	0	1	1	#E3	:	I/P PORT
1	1	1	0	0	1	0	0	#E4	:	O/P PORT
1	1	1	0	0	1	0	1	#E5	:	O/P PORT
1	1	1	0	0	1	1	0	#E6	:	O/P PORT
1	1	1	0	0	1	1	1	#E7	:	O/P PORT

3.2.2 หน่วยความจำ

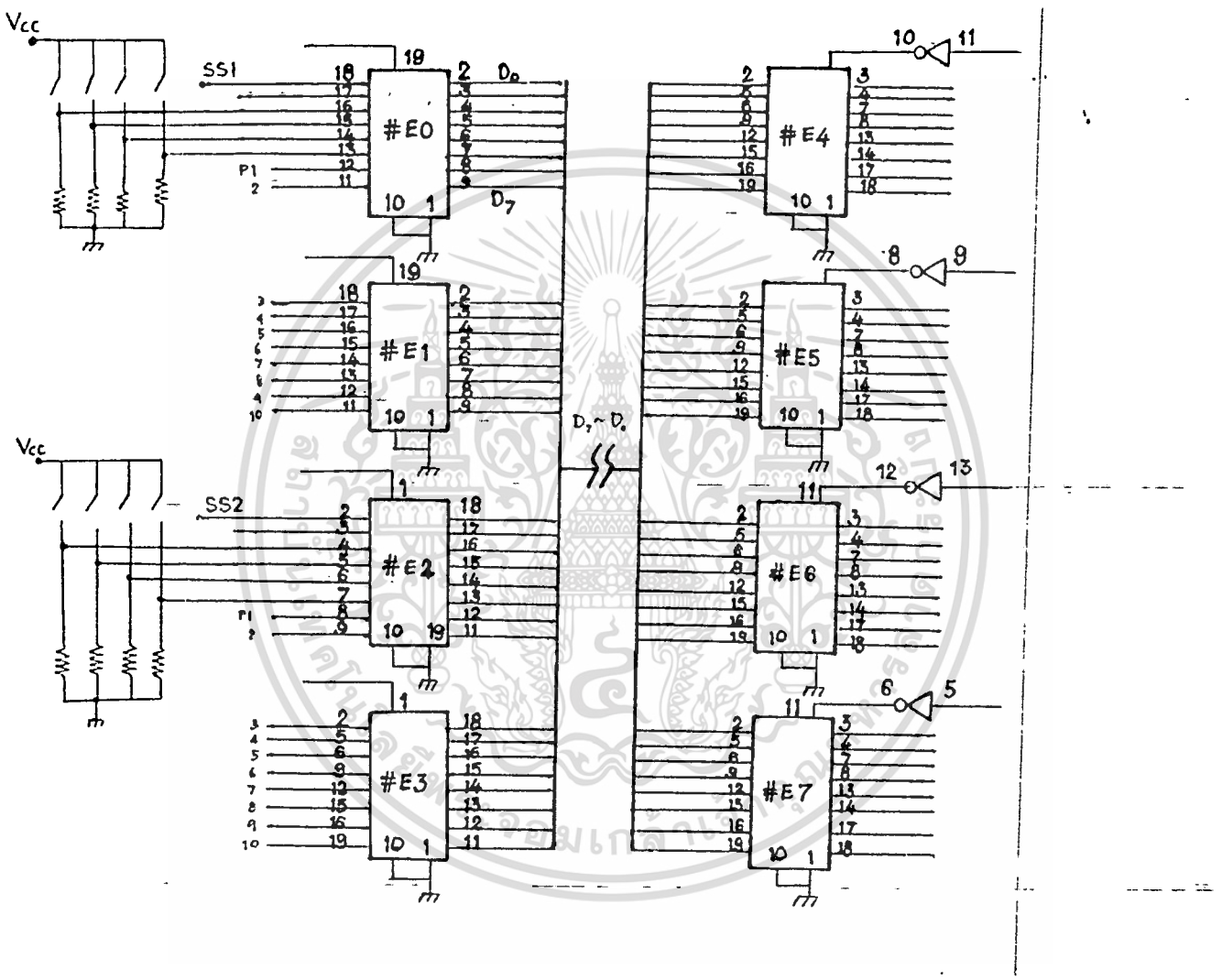
ในระบบใช้หน่วยความจำEPROM เบอร์ 2716 โดยต่อขาสัญญาณข้อมูลและขาแอดเดรส A_0-A_{10} และใช้สัญญาณ $\overline{MREQ}$ เป็นตัวเลือกพอร์ทดังแสดงไว้ดังรูป โปรแกรมการทำงานต่าง ๆ ของระบบจะถูกโปรแกรมไว้ภายในเพื่อสั่งและควบคุมการทำงานให้เป็นไปตามต้องการ



รูปที่ 3.2.2 แสดงหน่วยความจำ

3.2.3 หน่วยรับส่งข้อมูล

ในวงจรที่สร้างไว้ในพอร์ทสำหรับอินพุทข้อมูลในระบบได้กำหนดการทำงานไว้สำหรับ 2 รางสำหรับในรางแรกใช้พอร์ทอินพุทข้อมูลด้วยกัน 2 พอร์ท คือ พอร์ท #E0 และพอร์ท #E1 ซึ่งในแต่ละบิททั้ง 2 พอร์ทมีหน้าที่ในการอินพุทข้อมูลต่าง ๆ กัน จากรูปอธิบายการทำงานได้ดังนี้



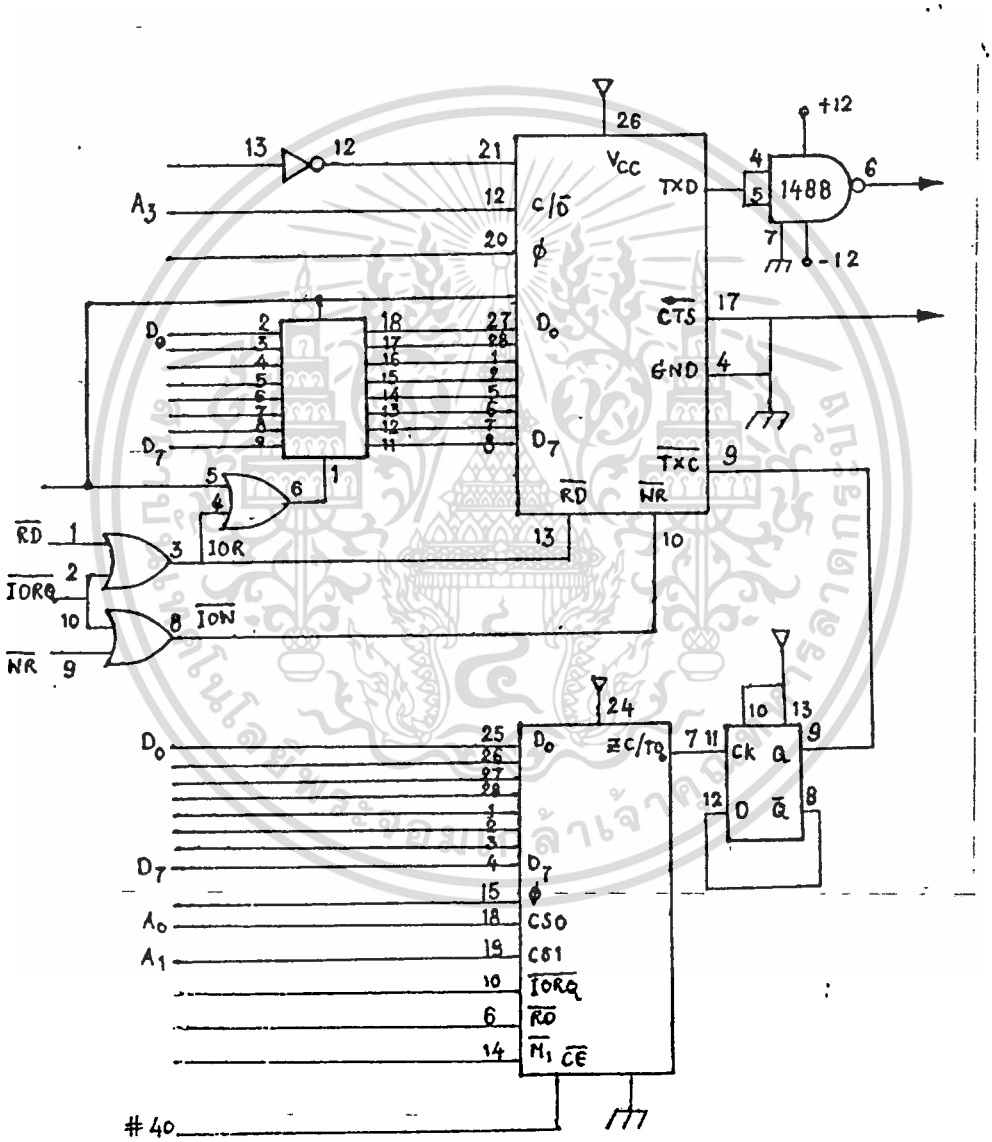
รูปที่ 3.2.3 แสดงหน่วยรับส่งข้อมูล

ในพอร์ท #E_0 ในบิทแรก (D_0)จะรับสัญญาฯจากวงจรสร้างพัลส์ ซึ่งสร้างสัญญาณมาจากวงจรหน่วงเวลา ซึ่งต่อมาจากตัวเซนเซอร์จับลูกโบว์ลิ่ง เพื่อนำไปประมวลผลตามโปรแกรมสำหรับใน D_2-D_5 จะเป็นตัวรับโค้ดของราง ซึ่งสามารถเซ็ทรหัสโค้ดของรางได้จากดิพสวิทช์ ซึ่งต่อไว้ในแผงวงจร ซึ่งในแผ่นการ์ดวงจรสามารถทำการเซ็ทโค้ดได้ถึง 2^4 หรือ 16 โค้ด ซึ่งสามารถทำไว้ขยายระบบในการใช้รางได้ในภายหลัง ส่วนใน D_6-D_7 ในพอร์ทแรกและบิท D_0-D_7 ในพอร์ทต่อไป (พอร์ท #E1) จะเป็นตัวรับสัญญาณข้อมูลจากเซนเซอร์สถานะของพินเข้ามา ซึ่งในการทำงานของพอร์ท และพอร์ท #E3 จะทำงานเหมือนกับพอร์ททั้งสองที่ได้กล่าวมา แต่เป็นของรางที่ 2

สำหรับในพอร์ทเอาท์พุทใช้บิท D6-D7 ของพอร์ท #E4 และบิท D0-D7 ของพอร์ท #E5 สำหรับเอาท์พุทสัญญาณสถานะของพินโบว์ลิ่งในรางแรกออกมา และเหมือนกันพอร์ท #E6 และ #E7 ก็ทำงานเหมือนกันแต่เป็นข้อมูลสถานะที่ได้ในรางที่สอง

3.2.4 ภาคส่งข้อมูล

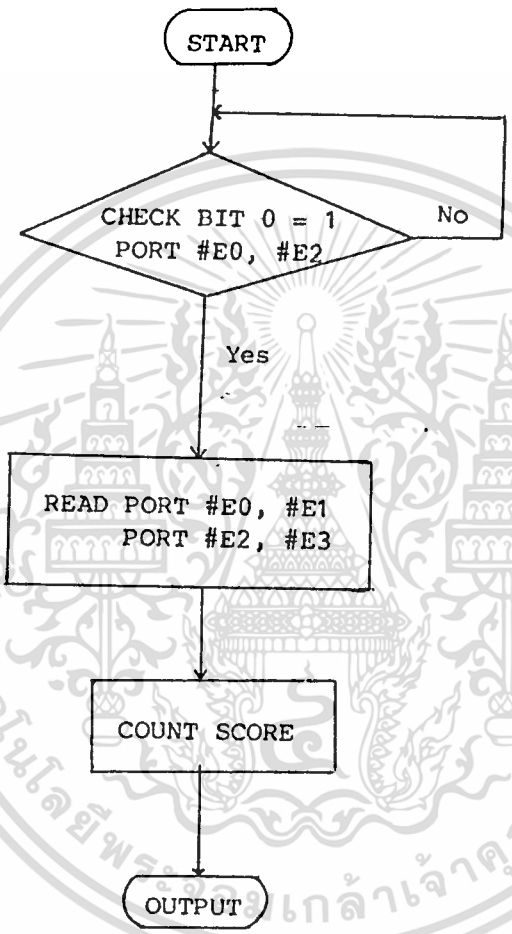
ในระบบใช้ไอซี 8251 เป็นตัวเปลี่ยนข้อมูลที่ได้รับเข้ามาจากพัฟเฟอร์ ซึ่งข้อมูลจะถูกส่งผ่านเข้ามาจากการส่งงานของโปรแกรม โดยต่อขาสัญญาณของ 8251 เข้ากับขาสัญญาณต่าง ๆ ของซีพียูดังรูป โดยความถี่ที่ใช้ในการส่งจะส่งมาจากซีพีซี ซึ่งจะถูกเซ็ทโปรแกรมให้สร้างความถี่ไว้ที่ 2400 เฮิรท์ส่งผ่านฟลิบปล็อบเหลือความถี่ของสัญญาณ 1200เฮิรท์ ซึ่งใช้ในการส่งผ่านข้อมูลไปยังเครื่องพีซีต่อไป



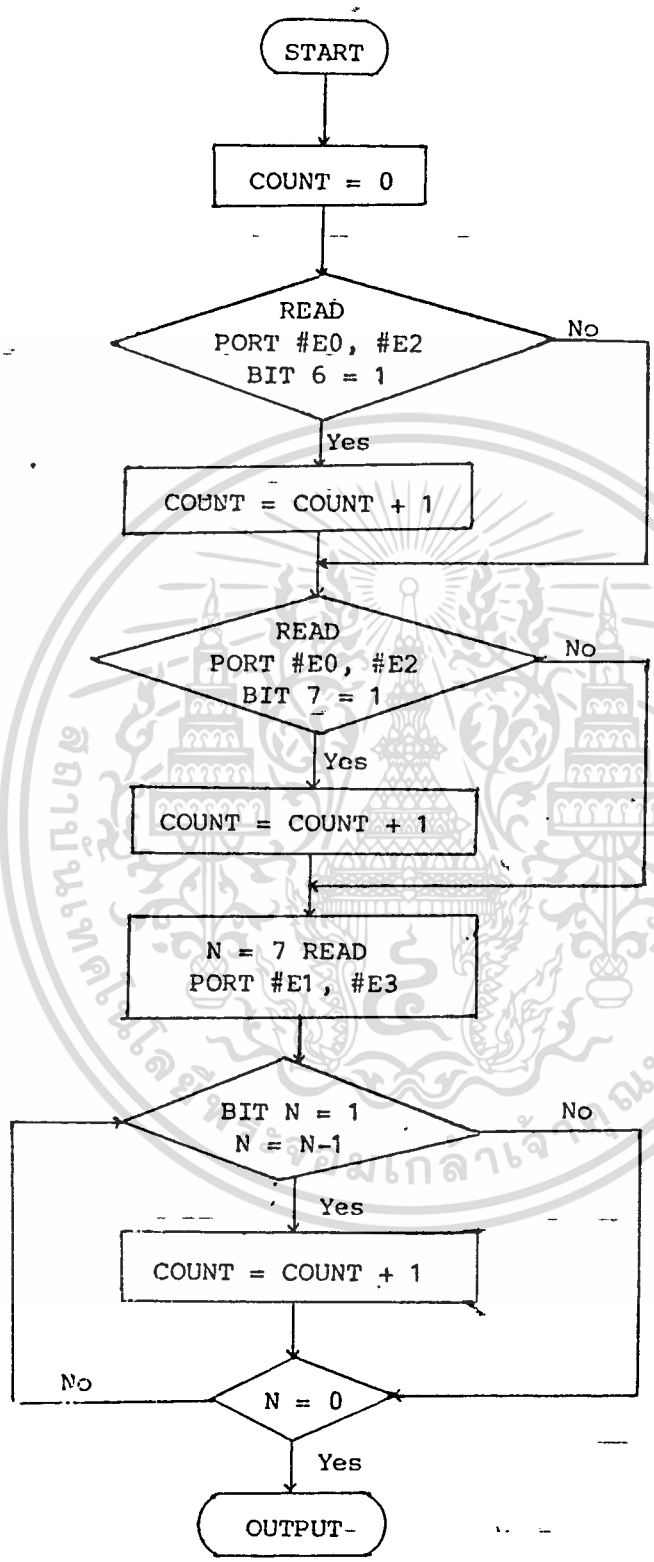
รูปที่ 3.2.4 แสดงการส่งข้อมูล

3.2.5 การทำงานของโปรแกรมควบคุม

การทำงานของโปรแกรมควบคุมสำหรับเครื่องนับคะแนนโบว์ลิ่ง ซึ่งเป็นโปรแกรมที่แสดงรายระเอียดการทำงานของซีพียูในระบบ สามารถแสดงเป็นโฟลว์ชาร์ตการทำงานดังรูป



: COUNT SCORE FLOW CHART



:PROGRAM CONTROL SYSTEM:

ADDRESS	OBJ CODE	MNEMONIC	COMMENT
0000	3E 07	LD A, 07	: control CTC
0002	D3 40	OUT (40H), A	2400 H_z
0004	3E 2E	LD A, 2E	
0006	D3 40	OUT (40H), A	
0008	3E CD	LD, CD	: control 8251
000A	D3 CC	OUT (CCH), A	command word
000C	DB E0	IN A, (E0H)	
000E	CB 47	BIT 0, A	
0010	C4 50 00	CALL NZ, 0050	
0013	DB E0	IN A, (E0H)	
0015	D3 E4	OUT (E4H), A	
0017	DB E1	IN A, (E1H)	
0019	D3 E5	OUT (E5H), A	
001B	DB E2	IN A, (E2H)	
001D	CB 47	BIT 0, A	
0020	C4 60 00	CALL NZ, 0060	
0023	DB E2	IN A, (E2H)	
0025	D3 E6	OUT (E6H), A	
0027	DB E3	IN A, (E3H)	
0029	D3 E7	OUT (E7H), A	
002B	C3 0C 00	JP 000C	
002e	76	HALT	

ADDRESS	OBJ CODE	MNEMONIC	COMMENT
0050	3E 00	LD A, 00	
0052	CD 70 00	CALL 0070	
0055	CD 90 00	CALL 0090	
0058	CD 70 00	CALL 0070	
005B	C9	RET	
0060	3E FF	LD A, FF	
0062	CD 70 00	CALL 0070	
0065	CD D0 00	CALL 00D0	
0068	CD 70 00	CALL 0070	
006B	C9	RET	
0070	D3 C4	OUT (C4H), A	
0072	3E 15	LD A, 15	
0074	D3 CC	OUT (CCH), A	
0076	DB CC	IN (CCH), A	
0078	CB 47	BIT 0, A	
007A	C3 72 00	JP Z, 0072	
007D	C9	RET	

ADDRESS	OBJ CODE	MNOMONIC	COMMENT
0090	21 30 01	LD HL, 0130	
0093	DB E0	IN A, (E0H)	
0095	CB 77	BIT 6, A	
0097	C4 20 01	CALL NZ, 0120	
009A	CB 7F	BIT 7, A	
009C	C4 20 01	CALL NZ, 0120	
009F	DB E1	IN A, (E1H)	
00A1	CB 47	BIT 0, A	
00A3	C4 20 01	CALL NZ, 0120	
00A6	CB 4F	BIT 1, A	
00A8	C4 20 01	CALL NZ, 0120	
00AB	CB 57	BIT 2, A	
00AD	C4 20 01	CALL NZ, 0120	
00B0	CB 5F	BIT 3, A	
00B2	C4 20 01	CALL NZ, 0120	
00B5	CB 67	BIT 4, A	
00B7	C4 20 01	CALL NZ, 0120	
00BA	CB 6F	BIT 5, A	
00BC	C4 20 01	CALL NZ, 0120	
00BF	CB 77	BIT 6, A	
00C1	C4 20 01	CALL NZ, 0120	
00C4	CB 7F	BIT 7, A	
00C6	C4 20 01	CALL NZ, 0120	
00C9	7E	LD A, (HL)	
00CA	C9	RET	

ADDERSS	OBJ CODE	MNCMONIC	COMMENT
00D0	21 30 01	LD HL, 0130	
00D3	DB E2	IN A, (E2H)	
00D5	CB 77	BIT 6, A	
00D7	C4 20 01	CALL NZ, 0120	
00DA	CB 7F	BIT 7, A	
00DC	C4 20 01	CALL NZ, 0120	
00E0	DB E3	IN A, (E3H)	
00E2	CB 47	BIT 0, A	
00E4	C4 20 01	CALL NZ, 0120	
00E7	CB 4F	BIT 1, A	
00E9	C4 20 01	CALL NZ, 0120	
00EC	CB 57	BIT 2, A	
00EF	C$ 20 01	CALL NZ, 0120	
00F2	CB 5F	BIT 3, A	
00F4	C4 20 01	CALL NZ, 0120	
00F7	CB 67	BIT 4, A	
009F	C4 20 01	CALL NZ, 0120	
00FC	CB 6F	BIT 5, A	
00FE	C4 20 01	CALL NZ, 0120	
0101	CB 77	BIT 6, A	
0103	C4 20 01	CALL NZ, 0120	
0106	CB 7F	BIT 7, A	
0108	C4 20 01	CALL NZ, 0120	
010B	7E	LD A, (HL)	
010C	C9	RET	

ADDERSS	OBJ CODE	MNUMONIC	COMMENT
0120	23	INC HL	
0121	C9	RET	
0130	30	30	
0131	31	31	
0132	32	32	
0133	33	33	
0134	34	34	
0135	35	35	
0136	36	36	
0137	37	37	
0138	38	38	
0139	39	39	
0140	41	41	

บทที่ 4

ผลการทดลอง

ในภาคตรวจจับพินโบวลึ่ง เราใช้ตัวรับแสงอินฟราเรดในการตรวจจับนั้นจากการทดลองระยะในการตรวจจับได้ประมาณ 1 เมตร ซึ่งเพียงพอที่จะตรวจจับพินโบว์ลึ่งทุดพินแล้ว และได้ทำการคิดค้นครอบปิดตัวรับแสงอินฟราเรดทำให้มีมในการรับแสงดีขึ้นด้วย

ในภาคประมวลผล จากการออกแบบและคำนวณวงจรที่ใช้จากผลการทดลองที่ได้นั้นเป็นไปตามที่ต้องการ ทั้งแต่การป้อนความถี่คล็อกให้กับระบบ การทำงานของไมโครโปรเซสเซอร์เบอร์ Z-80ซีพียู ตลอดจนการสร้างความถี่ที่ใช้ในการส่งข้อมูลจากความถี่คล็อก โดยอาศัยการทำงานของชิพซับพอท (CHIP SUPPORT) Z-80ซีทีซี ความที่ใช้ในภาคประมวลผลไปเข้ายังเครื่องไมโครคอมพิวเตอร์ ความถี่ที่ใช้สามารถกำหนดได้จากโปรแกรมควบคุม โดยค่าความถี่ที่ได้ออกมามีค่าใกล้เคียง 2400 เฮิร์ทซ ซึ่งเมื่อนำมาผ่านฟลิบฟลอบ (FLIP-FLOP) เพื่อปรับปรุงสัญญาณความถี่ที่ได้ถูกหารลงเหลือครึ่งหนึ่งทำให้ได้ความถี่เป็น 1200 เฮิร์ทซ ในระบบความถี่ที่ใช้ส่งในช่วงนี้เป็น 1190เฮิร์ทซ

ในส่วนของโปรแกรมที่ใช้ควบคุมการแสดงผล(คะแนน)ออกทางจอมอนิเตอร์นั้นก็ใช้ได้ แต่ควรพัฒนาทางด้านนี้อีกเล็กน้อยก็จะทำให้การแสดงผลสมบูรณ์ขึ้น

สรุปและวิจารณ์

จากการสร้างและทดลองเครื่องนับคะแนนโบวลิ่งทำให้ทราบถึงจุดที่ต้องพิจารณา คือ ปัญหาที่เกิดขึ้นต่างๆซึ่งส่วนใหญ่จะเป็น

ปัญหาที่เกิดขึ้นจากการตรวจจับของเซนเซอร์ในการตรวจจับสถานะของพินโบว์ลิ่งทั้งสิบ เนื่องจากลักษณะการตั้งของพินโบว์ลิ่งทั้งสิบซึ่งเป็นการตรวจจับโดยอาศัยสัญญาณความถี่ช่วงแสงอินฟราเรดมีทั้งข้อดีและข้อเสียคือในส่วนของข้อดีทำให้สามารถลดค่าใช้จ่ายเกี่ยวกับระบบตรวจจับได้ ส่วนข้อเสียจะเกิดการรบกวนจากสิ่งแวดล้อมภายนอกได้ง่าย ซึ่งได้แก้ไขโดยการติดกัวครอบปิดตัวรับทำให้สามารถรับสัญญาณในตำแหน่งที่ต้องการได้ดีขึ้น แต่ควรปรับปรุงเพื่อเพิ่มความเชื่อถือได้ของระบบ

และความไม่แน่นอนเชื่อถือได้ของระบบ ซึ่งอาจเกิดจากอุปกรณ์ที่ใช้ในโครงงานยังมีคุณภาพต่ำทำให้ระบบเกิดความเชื่อถือได้น้อยหรืออาจผิดพลาดได้ด้วย

ส่วนภาคประมวลผลซึ่งใช้ไมโครโปรเซสเซอร์ z-80 เป็นตัวประมวลผลนั้นจากผลการทดลองก็สามารถใช้งานได้ตามที่ต้องการ

ส่วนซอฟท์แวร์ที่ใช้ในการควบคุมการทำงานส่วนแสดงผล(คะแนน) ออกทางจอมอนิเตอร์ก็ใช้งานได้ค่อนข้างดี

สรุป ถึงแม้ว่าโครงงานนี้จะไม่เสร็จสมบูรณ์แบบยังมีส่วนที่ต้องแก้ไขปรับปรุงก็ตาม แต่ส่วนใหญ่ก็สามารถใช้งานได้ตามวัตถุประสงค์ที่ตั้งไว้ ซึ่งถือได้ว่าเป็นต้นแบบของเครื่องกรอกคะแนนโบวลิ่ง เพื่อเป็นแนวทางในการพัฒนาเครื่องกรอกคะแนนโบวลิ่งนี้ให้มีประสิทธิภาพดียิ่งขึ้นไป

ภาคผนวก

```
program project99;  {new}
var    i,j,a,b,c,d,e,p,q:integer;
       score:array[0..10,1..2] of integer;
       x:array[1..2,1..10,1..2] of integer;
procedure    comm3(d:integer);  {program to receive data by microprocessor}
var
  status   :byte;
procedure set1comparameter;

    begin
       port[$3fb]  :=$87;              {set 8 bit word,2 stop bit,
                                        no parity bit}
       port[$3f9]  :=$00;              {set Data Rate}
       port[$3f8]  :=$60;              {1200 bps}
       port[$3fb]  :=$07;
    end;

procedure receive;
    begin
      status  :=port[$3f8];
      for i := 0 to 0 do
        begin
         status  :=$00;
         while (status=$00) do
             begin
                status :=port[$3fd];
                status :=status and $01;
             end;
             x[i,j,d]:=port[$3f8];
       end;
    end;

begin
```

```
      set]comparameter;
      receive;
end;

procedure    table;
var          x1,y1,x2,y2:integer;
procedure    box(x1,y1,x2,y2:integer);
var          x,y:integer;
BEGIN
      window(1,1,80,25);
      gotoxy(x1,y1);
      write(chr(213));
      for x:=x1+1 to x2-1 do write(chr(205));
      write(chr(184));
      for y:=y1+1 to y2-1 do
          begin
            gotoxy(x1,y);
            write(chr(179));
          end;
      for y:=y1+1 to y2-1 do
          begin
            x1:=4;
            gotoxy(x1+11,y);
            write(chr(179));
            x1:=x1+16;
            while x1<66 do
              begin
                gotoxy (x1,y);
                write(chr(179));
                x1:=x1+5;
              end;
            gotoxy(x1+6,y);
            write(chr(179));
```

```
        end;
    x1:=4;
    gotoxy(x1,y2);
    write(chr(212));
    for x:=x1+1 to x2-1 do write(chr(205));
    write(chr(190));
    gotoxy(9,y1+4);
    window(x1+1,y1+1,x2-1,22);
end;

procedure pause;
var     a:char;
begin
        repeat
          begin
          read(kbd,a);write(a);
          end;
        until ord(a)=32
end;

begin {main}
      clrscr;
      gotoxy(7,4);
      writeln('name1    1   2   3   4   5   6   7   8   9   10   total  ');
      box(4,2,76,10);
      gotoxy(3,5);
      pause;
      gotoxy(3,14);
      writeln('name2    1   2   3   4   5   6   7   8   9   10   total  ');
      box(4,14,76,22);
      gotoxy(3,5);
      window(1,1,80,25);
      pause;
```

```
      gotoxy(17,5);
end;

procedure  strike1(var a:integer);
  begin
    gotoxy((5*j)+12,p);
    case a of
  1 : begin
        write('@10');
        a:=a+1;
        if x[i,(j-1),1]+x[i,(j-1),2]=10 then
           begin
             score[j-1,i]:=score[j-2,i]+20;
             gotoxy((5*(j-1)+12),q);
             write(score[j-1,i]);
           end
        else
           begin
             x[i,j,2]:=0;
           end;
      end;
  2 : begin
        a:=a+1;
        if x[i,j,1]=10 then
          begin
            write('@10');
            score[j-1,i]:=score[j-2,i]+20;
            gotoxy((5*(j-1)+12),q);
            write(score[j-1,i]);
          end
        else
          begin
            write(x[i,j,1],'/');
```

```
            read(kbd,x[i,j,2]);{ comm3(2);}
            write(x[i,j,2]);
            score[j-1,i]:=score[j-2,i]+10+x[i,j,1]+x[i,j,2];
            gotoxy((5*(j-1)+12),q);
            write(score[j-1,i]);
          end;
      end;
   else;
  end;
 end;

procedure  spare1(var b:integer);
begin
  if b=2 then
   begin
    if x[i,(j-1),1]+x[i,(j-1),2]=10 then
       begin
         score[j-1,i]:=score[j-2,i]+10+x[i,j,1];
         if x[i,(j-1),1]=10 then
         score[j-1,i]:=score[j-1,i]+x[i,j,2];
         gotoxy((5*(j-1)+12),q);
         write(score[j-1,i]);
       end;
   end
  else
    begin
      b:=b+1;
    end;
end;

procedure  abc1(i,j:integer);
begin
          read(kbd,x[i,j,1]); { comm3(1);}
```

```
if a=2 then strike1(a);
if x[i,j,1]=10 then strike1(a)
else
 begin
  if a<>3 then
    begin
      gotoxy((5*j)+12,p);
      write(x[i,j,1],'/');
      read(kbd,x[i,j,2]); { comm3(2);}
      write(x[i,j,2]);
    end;
  if a>2 then
     begin
       a:=1;
       if x[i,(j-1),1]=10 then
          begin
            score[j-1,i]:=score[j-2,i]+10+x[i,j,1]+x[i,j,2];
            gotoxy((5*(j-1))+12,q);
            write(score[j-1,i]);
          end;
     end;
  if b>2 then spare1(b);
  if x[i,j,1]+x[i,j,2]=10 then spare1(b)
  else
    begin
      if x[i,(j-1),1]+x[i,(j-1),2]=10 then
         begin
           spare1(b);
           score[j,i]:=score[j-1,i]+x[i,j,1]+x[i,j,2];
           gotoxy((5*j)+12,q);
           write(score[j,i]);
         end
      else
```

```
                begin
                  score[j,i]:=score[j-1,i]+x[i,j,1]+x[i,j,2];
                  gotoxy((5*j)+12,q);
                  write(score[j,i]);
                end;
            end;
        end;
    end;
procedure  strike2(var c:integer);
 begin
   gotoxy((5*j)+12,p);
   case c of
 1 : begin
       write('@10');
       c:=c+1;
       if x[i,(j-1),1]+x[i,(j-1),2]=10 then
          begin
            score[j-1,i]:=score[j-2,i]+20;
            gotoxy((5*(j-1)+12),q);
            write(score[j-1,i]);
          end
       else
          begin
            x[i,j,2]:=0;
          end;
```

```
            write(score[j-1,i]);
          end
        else
          begin
            write(x[i,j,1],'/');
            read(kbd,x[i,j,2]); { comm3(2);}
            write(x[i,j,2]);
            score[j-1,i]:=score[j-2,i]+10+x[i,j,1]+x[i,j,2];
            gotoxy((5*(j-1)+12),q);
            write(score[j-1,i]);
          end;
      end;
    else;
  end;
end;

procedure  spare2(var e:integer);
begin
  if e=2 then
   begin
    if x[i,(j-1),1]+x[i,(j-1),2]=10 then
       begin
         score[j-1,i]:=score[j-2,i]+10+x[i,j,1];
         if x[i,(j-1),1]=10 then
         score[j-1,i]:=score[j-1,i]+x[i,j,2];
         gotoxy((5*(j-1)+12),q);
         write(score[j-1,i]);
       end;
   end
  else
    begin
      e:=e+1;
    end;
```

```
end;

procedure  abc2(i,j:integer);
begin
        read(kbd,x[i,j,1]); { comm3(1);}
        if c=2 then strike2(c);
        if x[i,j,1]=10 then strike2(c)
        else
         begin
          if c<>3 then
            begin
              gotoxy((5*j)+12,p);
              write(x[i,j,1],'/');
              read(kbd,x[i,j,2]); { comm3(2);}
              write(x[i,j,2]);
            end;
         if c>2 then
            begin
              c:=1;
              if x[i,(j-1),1]=10 then
                 begin
                   score[j-1,i]:=score[j-2,i]+10+x[i,j,1]+x[i,j,2];
                   gotoxy((5*(j-1))+12,q);
                   write(score[j-1,i]);
                 end;
            end;
        if e>2 then spare2(e);
        if x[i,j,1]+x[i,j,2]=10 then spare2(e)
        else
          begin
            if x[i,(j-1),1]+x[i,(j-1),2]=10 then
               begin
                 spare2(e);
```

```
                score[j,i]:=score[j-1,i]+x[i,j,1]+x[i,j,2];
                gotoxy((5*j)+12,q);
                write(score[j,i]);
              end
            else
              begin
                score[j,i]:=score[j-1,i]+x[i,j,1]+x[i,j,2];
                gotoxy((5*j)+12,q);
                write(score[j,i]);
              end;
          end;
        end;
      end;

BEGIN {***MAIN***}
      clrscr;
      table;
      a:=1;
      b:=1;
      c:=1;
      e:=1;
      score[0,i]:=0;
      For j:=1 to 10 do
      begin
        For i:=1 to 2 do
        begin
          if i=1 then
            begin
              p:=5; q:=7;
              abc1(i,j);
            end
          else
            begin
```

```
            p:=17; q:=19;
            abc2(i,j);
          end;
      if j=10 then
       begin
         score[j,i]:=score[j-1,i]+x[i,j,1]+x[i,j,2];
         gotoxy((5*j)+12,q);
         write(score[j,i]);
         gotoxy(5*(j+1)+14,q);
         write(score[j,i]);
         sound(1000);
         delay(1000);
         nosound;
       end;
     end;
  end;
END.
```

MOTOROLA

MRD360
MRD370

NPN SILICON HIGH SENSITIVITY PHOTODARLINGTON TRANSISTORS

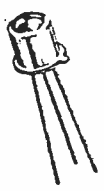
. . . designed for application in industrial inspection, processing and control, counters, sorters, switching and logic circuit or any design requiring very high radiation sensitivity at low light levels.

- Popular TO-18 Type Hermetic Package for Easy Handling and Mounting
- Sensitive Throughout Visible and Near Infrared Spectral Range for Wider Application
- Minimum Light Current 12 mA at H = 0.5 mW/cm^2 (MRD360)
- External Base for Added Control
- Switching Times —
  t_r @ I_L = 1.0 mA peak = 15 μs (Typ) — MRD370
  t_f @ I_L = 1.0 mA peak = 40 μs (Typ) — MRD370

PHOTODARLINGTON TRANSISTORS NPN SILICON

40 VOLTS

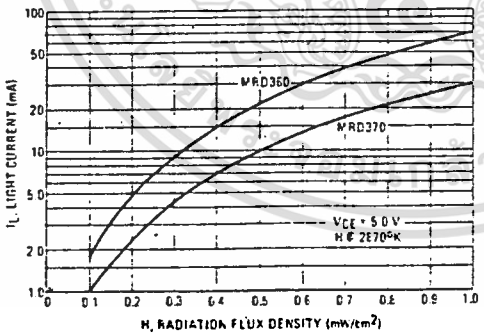
250 MILLIWATTS

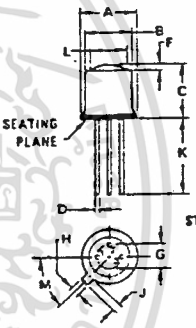


MAXIMUM RATINGS (T_A = 25°C unless otherwise noted).

Rating (Note 1)	Symbol	Value	Unit
Collector-Emitter Voltage	V_{CEO}	40	Volts
Emitter-Base Voltage	V_{EBO}	10	Volts
Collector-Base Voltage	V_{CBO}	50	Volts
Light Current	I_L	250	mA
Total Device Dissipation @ T_A = 25°C Derate above 25°C	P_D	250 1.43	mW mW/°C
Operating and Storage Junction Temperature Range	T_J, T_{stg}	-65 to +200	°C

FIGURE 1 – LIGHT CURRENT versus IRRADIANCE





STYLE 1:
PIN 1. EMITTER
2. BASE
3. COLLECTOR

NOTES:
1. LEADS WITHIN .13 mm (.005) RADIUS OF TRUE POSITION AT SEATING PLANE, AT MAXIMUM MATERIAL CONDITION.
2. PIN 3 INTERNALLY CONNECTED TO CASE.

DIM	MILLIMETERS MIN	MILLIMETERS MAX	INCHES MIN	INCHES MAX
A	5.31	5.84	0.209	0.230
B	4.52	4.95	0.178	0.195
C	4.57	6.48	0.180	0.255
D	0.41	0.48	0.016	0.019
F	–	1.14	–	0.045
G	2.54 BSC		0.100 BSC	
H	0.86	1.17	0.035	0.046
J	0.84	1.22	0.033	0.048
K	12.70	–	0.500	–
L	3.35	4.01	0.132	0.158
M	45° BSC		45° BSC	

CASE 82-05
TO-18 Type

STATIC ELECTRICAL CHARACTERISTICS (T_A = 25°C unless otherwise noted.)

Characteristic	Symbol	Min	Typ	Max	Unit
Collector Dark Current (V_{CE} = 10 V, H ≈ 0) T_A = 25°C	I_{CEO}	–	10	100	nA
Collector-Base Breakdown Voltage (I_C = 100 μA)	$V_{(BR)CBO}$	50	100	–	Volts
Collector-Emitter Breakdown Voltage (I_C = 100 μA)	$V_{(BR)CEO}$	40	80	–	Volts
Emitter-Base Breakdown Voltage (I_E = 100 μA)	$V_{(BR)EBO}$	10	15.5	–	Volts

OPTICAL CHARACTERISTICS (T_A = 25°C unless otherwise noted.)

Characteristic	Device Type	Symbol	Min	Typ	Max	Unit
Light Current V_{CC} = 5.0 V, R_L = 10 Ohms (Note 1)	MRD360 MRD370	I_L	12 3.0	20 10	– –	mA
Collector-Emitter Saturation Voltage (I_L = 10 mA, H = 2 mW/cm² at 2870°K)		$V_{CE(sat)}$	–	0.6	1.0	Volts
Photo Current Rise Time (Note 2) (R_L = 100 ohms, I_L = 1.0 mA peak)	MRD360 MRD370	t_r	– –	15 15	100 100	μs
Photo Current Fall Time (Note 2) (R_L = 100 ohms, I_L = 1.0 mA peak)	MRD360 MRD370	t_f	– –	65 40	150 150	μs

NOTES:

1. Radiation flux density (H) equal to 0.5 mW/cm² emitted from a tungsten source at a color temperature of 2780°K.
2. For unsaturated response time measurements, radiation is provided by pulsed GaAs (gallium-arsenide) light-emitting diode (λ ≈ 0.9 μm) with a pulse width equal to or greater than 500 microseconds (see Figure 6) I_L = 1.0 mA peak.

FIGURE 7 – CONSTANT ENERGY SPECTRAL RESPONSE

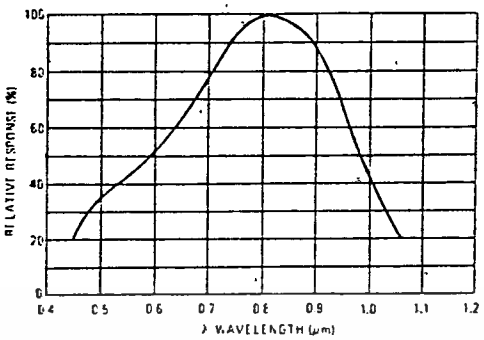
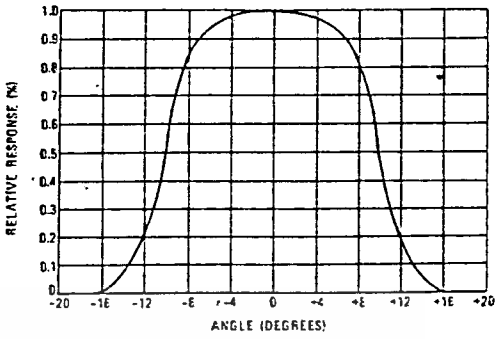


FIGURE 8 – ANGULAR RESPONSE



กิตติกรรมประกาศ

ผลงานปริญญานิพนธ์ชิ้นนี้สามารถสำเร็จลงได้ที่ด้วยความสนับสนุนในด้านต่าง ๆ ทั้งแนวความคิดและคำแนะนำจากอาจารย์สมยศ จุฬะปิยะ อาจารย์ในภาควิชาวิศวกรรมโทรคมนาคม และเจ้าหน้าที่ฝ่ายธุรการทุกท่านที่ได้มีส่วนช่วยเหลือตลอดจนเพื่อน ๆ ในภาคที่ช่วยเหลือและให้ความสนใจสอบถาม

ผู้จัดทำขอขอบพระคุณในความช่วยเหลือของทุก ๆ ท่าน ที่มีส่วนช่วยให้งานปริญญานิพนธ์ชิ้นนี้สำเร็จลุล่วงไปได้ด้วยดี

ผู้จัดทำ

หนังสืออ้างอิง

ก. หนังสืออ้างอิงภาษาอังกฤษ

1. William Barden , Jr. " Z-80 Microcomputer Design Projects "
   Howard W. Sams & Co.,Inc., 1980
2. " Microprocessor MPF-I USER'S MANUAL "
   MULTITECH INDUSTRIAL CORPORATION , 1981
3. TEXAS INSTRUMENT " THE TTL DATABOOK FOR DESIGN ENGINEERS "
   TAXAS INSTRUMENT INCORPORATION. 2nd , 1981
4. NATIONAL SEMICONDUCTOR " LINEAR DATA BOOK "
   NATIONAL SEMICONDUCTOR CORPORATION , 1982

ข. หนังสืออ้างอิงภาษาไทย

1. ชูชัย ธนสารตั้งเจริญ และคณะ " การใช้งาน Z-80 "
   สำนักพิมพ์พิสิกส์เซ็นเตอร์ พ.ศ. 2530
2. ธานิน ถาวรศาสนวงศ์ และ ทินกร คุ้ก " การอินเทอร์เฟส IBM PC "
   สำนักพิมพ์พิสิกส์เซ็นเตอร์ พ.ศ. 2531
3. กลุ่ม CNS " OP AMP "
   สำนักพิมพ์พิสิกส์เซ็นเตอร์ พ.ศ. 2531