การส่งข้อมูลดิจิตอลแบบเป็นลำดับ

DATA SEQUENTIAL COLLECTING

โดย

นางสาวธีรนาฏ เจียมพจมาน

นายธีระ กลิ่นกำธรกุล

นางสาวนงนุช สุขจิตร

ปริญญานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิศวกรรมศาสตรบัณฑิต

สาขาวิชาวิศวกรรมอิเล็กทรอนิกส์

คณะวิศวกรรมศาสตร์

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

ปีการศึกษา 2541

การส่งข้อมูลดิจิตอลแบบเป็นลำดับ
DATA SEQUENTIAL COLLECTING

โดย

นางสาวธีรนาฏ เจียมพจมาน 38014208

นายธีระ กลิ่นกำธรกุล 38014212

นางสาวนงนุช สุขจิตร 38014218

อาจารย์ที่ปรึกษา

ผศ. พลผดุง ผดุงกุล

ปริญญานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิศวกรรมศาสตร์บัณฑิต

สาขาวิชาวิศวกรรมอิเล็กทรอนิกส์

คณะวิศวกรรมศาสตร์

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

ปีการศึกษา 2541

ปริญญานิพนธ์ปีการศึกษา 2541

ภาควิชาอิเล็กทรอนิกส์

คณะวิศวกรรมศาสตร์ สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

เรื่อง ระบบการส่งข้อมูลแบบเป็นลำดับ (Data Sequencial Collecting System)

ผู้จัดทำ

1. นางสาวธีรนาฏ เจียมพจมาน เลขประจำตัว 38014208
2. นายธีระ กลิ่นกำธรกุล เลขประจำตัว 38014212
3. นางสาวนงนุช สุขจิตร เลขประจำตัว 38014218

ลงชื่อ.....อาจารย์ที่ปรึกษา

(ผศ. พลผดุง ผดุงกุล)

การส่งข้อมูลดิจิตอลแบบเป็นลำดับ

DATA SEQUENTIAL COLLECTION

นางสาวธีรนาฎ	เจียมพจมาน	เลขประจำตัว	38014208
นายธีระ	กลิ่นกำธรกุล	เลขประจำตัว	38014212
นางสาวนงนุช	สุขจิตร	เลขประจำตัว	38014218

โครงงานนี้ได้รับการตรวจสอบแล้ว พร้อมที่จะทำการสอบได้

(ผศ. พลผดุง ผดุงกุล)

อาจารย์ที่ปรึกษา

การส่งข้อมูลดิจิตอลแบบเป็นลำดับ

น.ส. ธีรนาฏ เจียมพจมาน
นาย ธีระ กลิ่นกำธรกุล
น.ส. นงนุช สุขจิตร
อ. พลผดุง ผดุงกุล (อาจารย์ที่ปรึกษา)
ปีการศึกษา 2541

บทคัดย่อ

ปริญญานิพนธ์ฉบับนี้เป็นการศึกษาการส่งข้อมูลดิจิตอลแบบเป็นลำดับ เป็นการประยุกต์การใช้งานการควบคุมและแสดงผลแบบพัลส์วิดท์ดิฟเฟอเรนเชียลซีเควนเชียลคอนโทรล ซึ่งจะใช้การส่งข้อมูลแบบดิฟเฟอเรนเชียลโหมดเพื่อลดผลของสัญญาณรบกวนและสามารถนำสัญญาณในสายนำสัญญาณได้ยาวขึ้น โดยในระบบการส่งข้อมูลดิจิตอลแบบเป็นลำดับนี้ประกอบด้วย 3 ส่วนหลัก ๆ คือ ส่วนของฮาร์ดแวร์ตัวแม่ ซึ่งจะทำหน้าที่ควบคุมและจัดการเกี่ยวกับการรับข้อมูลดิจิตอล 8 บิตจากฮาร์ดแวร์ตัวลูก ส่วนของฮาร์ดแวร์ตัวลูกจะทำหน้าที่รับสัญญาณอินพุทของระบบ แล้วทำการแปลงสัญญาณอนาลอกเป็นดิจิตอลและส่งข้อมูลกลับไปยังฮาร์ดแวร์ตัวแม่ ซึ่งฮาร์ดแวร์ตัวลูกนี้จะมีจำนวนได้สูงสุด 31 ตัว ส่วนสุดท้ายคือส่วนของเครื่องคอมพิวเตอร์ ซึ่งจะทำการติดต่อกับฮาร์ดแวร์ตัวแม่ด้วยมาตรฐาน RS-232 โดยใช้แสดงความสัมพันธ์ระหว่างข้อมูลกับเวลาในรูปของกราฟ

Data Sequential Collecting

Miss. Teeranard Jaimphotjamarn
Mr. Teera Klinkamthornkul
Miss. Nongnuch Sukchit
Mr. Ponpadung Padungkun (Advisor)
1998

Abstract

This thesis describes about Data Sequential Collecting System. It's application of Pulse Width Differential Sequential Control and Monitoring, that use the differential technique for reduce noise and increase transmission length. In Data Sequencial Collecting System, it consist of 3 parts. The first one is the Master unit to control and manage about receiving 8 bits digital data from Slave unit. In the part of Slave unit the analog signal input are received and convert to digital signal before transmit to the Master unit. The maximum number of Slave unit is 31. The third part is PC that connects with the Master unit by RS-232 standard to monitor the relation between the data and time in graph.

สารบัญ

หน้า

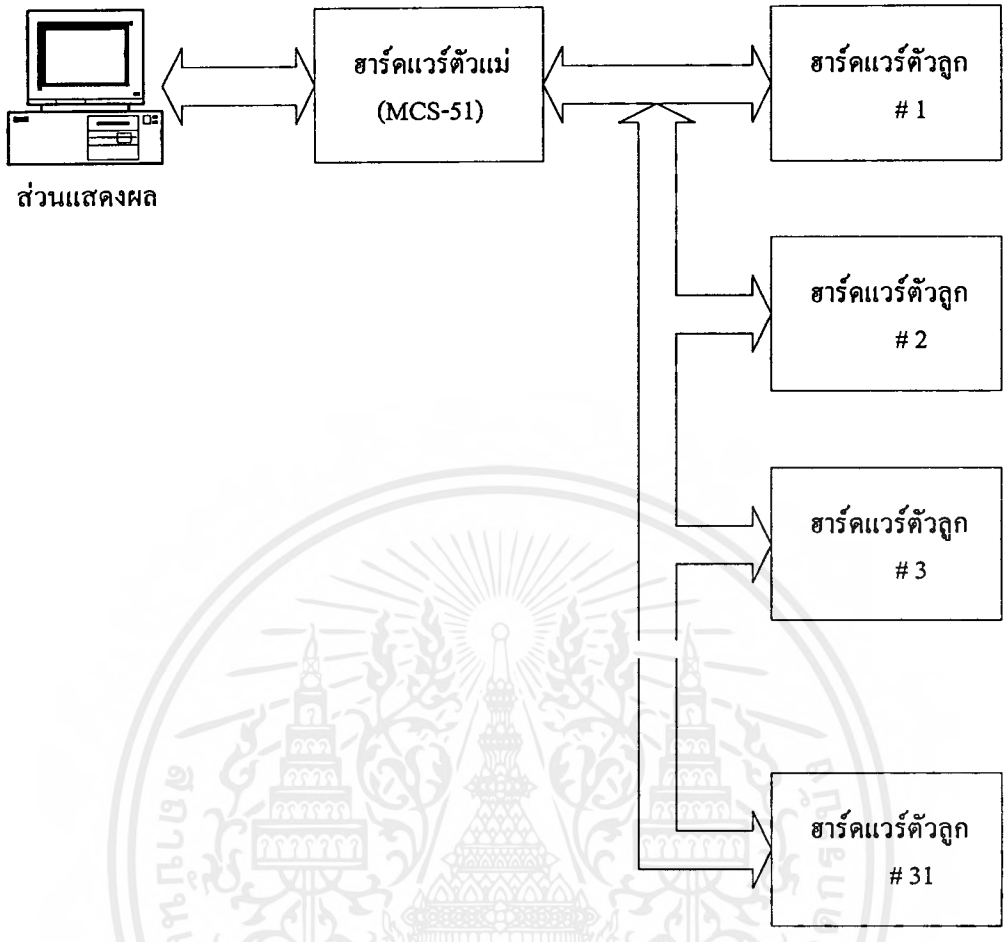
สารบัญรูป

บทที่1
บทนำ

การที่เราต้องการที่จะศึกษาระบบใดระบบหนึ่งนั้นจะต้องมีการเก็บข้อมูลเพื่อที่จะนำข้อมูลมาประมวลผลหรือเป็นการเก็บข้อมูลไว้เปรียบเทียบ ดังนั้นปริญญานิพนธ์การส่งข้อมูลดิจิตอลแบบเป็นลำดับ (Data Sequential Collecting System) จะเป็นรูปแบบของการส่งข้อมูลของระบบที่เราสนใจไปแสดงผลบนเครื่องคอมพิวเตอร์

ในระบบควบคุมและแสดงผลระยะไกลแบบเก่าซึ่งใช้สัญญาณ 3 ระดับคือ 12 V 5 V และ 0 V และเป็นการส่งในสายส่งแบบซิงเกิลเอนด์ (Single End) ข้อเสียของระบบเก่าคือเมื่อต่อสายนำสัญญาณยาวมากๆจะทำให้การรับสัญญาณอาจเกิดการผิดพลาดได้ง่าย ดังนั้นในระบบการส่งข้อมูลดิจิตอลแบบเป็นลำดับนี้จึงได้ทำการประยุกต์การใช้งานระบบส่งสัญญาณซีเควนเชียลคอนโทรล (Sequencial Control) ซึ่งจะเป็นการส่งสัญญาณในแบบดิฟเฟอเรนเชียล (Differential) และการส่งข้อมูลแบบพัลส์วิดท์มอดูเลชั่น (Pulse Width Modulation) การตรวจจับข้อมูลในสายนำสัญญาณจะเป็นการเปรียบเทียบผลต่างของสัญญาณในสายส่งแบบดิฟเฟอเรนเชียล ซึ่งถึงแม้ว่าสัญญาณที่ส่งมาจะเปลี่ยนแปลงไป แต่ผลที่ได้จากวงจรตรวจจับสัญญาณก็ยังคงได้สัญญาณเหมือนกับสัญญาณต้นทางและยังสามารถป้องกันสัญญาณรบกวนจากภายนอกได้ดี

ระบบการส่งข้อมูลดิจิตอลแบบเป็นลำดับนี้ประกอบด้วย 3 ส่วนหลักๆ คือ ส่วนของฮาร์ดแวร์ตัวแม่ (Master Unit) ,ส่วนของฮาร์ดแวร์ตัวลูก (Slave Unit) และส่วนของเครื่องคอมพิวเตอร์ซึ่งใช้ในการแสดงผล ในปริญญานิพนธ์นี้ MCS-51 จะทำหน้าที่เป็นฮาร์ดแวร์ตัวแม่ใช้สำหรับการสร้างสัญญาณพัลส์ควบคุมการทำงานโดยใช้การส่งสัญญาณแบบดิฟเฟอเรนเชียลไปยังฮาร์ดแวร์ตัวลูก จากนั้นฮาร์ดแวร์ตัวลูกก็จะทำการส่งข้อมูลแบบพัลส์วิดท์ กลับมายังฮาร์ดแวร์ตัวแม่ เพื่อจะให้ฮาร์ดแวร์ตัวแม่ทำการจักการรวบรวมข้อมูลเหล่านั้นส่งไปแสดงผลยังเครื่องคอมพิวเตอร์ โดยใช้มาตรฐานการส่งแบบ RS-232 ที่เครื่องคอมพิวเตอร์จะแสดงข้อมูลและกราฟความสัมพันธ์ระหว่างข้อมูลของฮาร์แวร์ตัวลูกแต่ละตัวกับเวลาพร้อมทั้งยังสามารถบันทึกข้อมูลเก็บเป็นไฟล์ข้อมูล เพื่อนำมาประมวลผล หรือใช้ในการวิเคราะห์ข้อมูลในภายหลังได้



รูปที่ 1.1 แสดงรูปแบบการส่งข้อมูลดิจิตอลแบบเป็นลำดับ

บทที่2
ทฤษฎี

2.1 ไมโครคอนโทรลเลอร์ตระกูล MCS-51

หน่วยการทำงานพื้นฐานของไมโครคอนโทรลเลอร์ตระกูล MCS-51 ซึ่งแสดงในรูปที่ 2.1 นี้จะประกอบด้วย

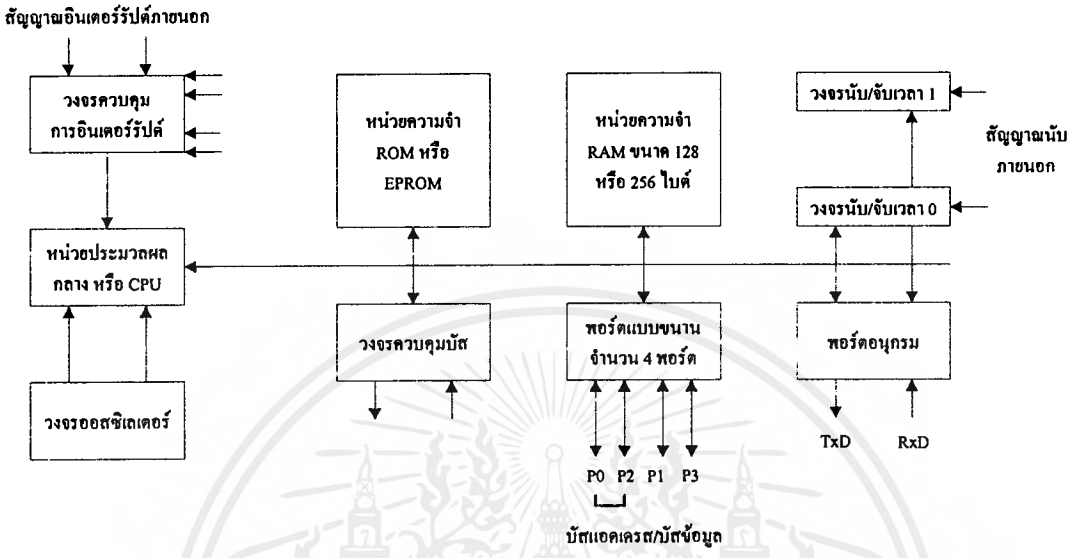
- หน่วยประมวลผลกลางขนาด 8 บิต
- หน่วยประมวลผลสำหรับข้อมูลแบบบิต (Boolean Processor)
- ความสามารถในการอ้างถึงตำแหน่งของหน่วยความจำโปรแกรม 64 กิโลไบต์
- ความสามารถในการอ้างตำแหน่งของหน่วยความจำข้อมูล 64 กิโลไบต์
- หน่วยความจำโปรแกรมภายในขนาด 4 กิโลไบต์ แบบอีพรอม (เบอร์ 8751) หรือแบบรอม(เบอร์ 8051)
- หน่วยความจำแบบรอมภายในจำนวน 128 ไบต์
- พอร์ตอินพุต/เอาต์พุตแบบขนานจำนวน 32 เส้น ซึ่งสามารถแยกทำงานได้อย่างอิสระ
- วงจรนับ/จับเวลาขนาด 16 บิต จำนวนสองวงจร
- วงจรสื่อสารแบบอนุกรมแบบฟูลดูเพล็กซ์ (Full Dulplex)
- วงจรควบคุมการอินเตอร์รัปต์จากแหล่งกำเนิดสัญญาณ 6 ประเภท พร้อมการกำหนดลำดับความสำคัญได้สองระดับ
- วงจรออสซิเลเตอร์ภายใน

2.1.1 วงจรจับเวลา

MCS-51 ประกอบด้วยรีจีสเตอร์ขนาด 16 บิตจำนวนสองตัว คือ T0 (timer 0) และ T1 (Timer1) ซึ่งสามารถนำไปใช้งานได้อย่างอิสระ โดยสามารถควบคุมให้ทำหน้าที่เป็นตัวจับเวลา (Timer) เพื่อนับจำนวนพัลส์สัญญาณนาฬิกาภายใน หรือควบคุมให้ทำหน้าที่เป็นตัวนับ (Counter) เพื่อนับจำนวนของพัลส์ของระบบได้

ภายในรีจีสเตอร์แต่ละตัวยังสามารถแยกออกได้เป็นรีจีสเตอร์ขนาด 8 บิต คือ TH0 กับ TL0 สำหรับรีจีสเตอร์ T0 และ TH1 กับ TL1 สำหรับรีจีสเตอร์ T1 โดยการทำงานของรีจีสเตอร์ทั้งสอง

ตัวนี้มีผลมาจากการกำหนดค่าของบิตที่อยู่ภายในรีจีสเตอร์ TMOD (Time mode control register) และรีจีสเตอร์ TCON (Timer/counter cintrol register)



รูปที่ 2.1 แผนภาพบล็อกแสดงหน่วยทำงานพื้นฐานของ MCS-51

เมื่อกำหนดให้ทำงานเป็นตัวจับเวลา รีจีสเตอร์จะทำการเพิ่มค่าขึ้นทีละหนึ่งในทุก ๆ แมชชีนไซเคิลการทำงานของซีพียู ดังนั้นอาจจะกล่าวได้ในอีกลักษณะว่าการทำงานเป็นตัวจับเวลาก็เป็นการนับหน่วยเวลาซึ่งสร้างมาจากวงจรออสซิลเลเตอร์ของซีพียูเอง การคำนวณค่าระยะเวลาของหนึ่งแมชชีนไซเคิลจะใช้เวลานับเท่ากับคาบเวลาของออสซิลเลเตอร์จำนวน 12 คาบ หรือคิดเป็นค่าอัตราการนับในแต่ละครั้งจะใช้เวลาเท่ากับ 1/12 เท่าของความถี่ออสซิลเลเตอร์

จากกระบวนการทำงานของวงจรนับ/จับเวลาของ MCS-51 จำเป็นต้องทำการกำหนดค่าเริ่มต้นให้กับรีจีสเตอร์ T0 หรือ T1 ค่านี้จะเป็นค่าจำนวนของพัลส์ภายในที่ต้องการจะให้นับหรือค่าของจำนวนพัลส์ภายนอกที่เข้ามาทางขาสัญญาณ T0 และ T1 ค่าตัวเลขภายในรีจีสเตอร์นี้จะต้องลดให้มีค่าน้อยกว่าค่าที่ต้องการอยู่หนึ่งค่า ทั้งนี้เนื่องจากการทำงานของรีจีสเตอร์จะเพิ่มค่าจากที่กำหนดไปเรื่อยๆ จนถึงค่าสูงสุดของรีจีสเตอร์ และกลับไปเป็นค่าศูนย์เมื่อมีพัลส์สุกท้ายเกิดขึ้นซึ่งเรียกว่ามี การโอเวอร์โฟลว์ เกิดขึ้น ทำให้เกิดการกำหนดค่าของแฟล็กเพื่อแจ้งให้ซีพียูได้รับทราบ ดังนั้นโปรแกรมโดยทั่วไปจึงมักจะใช้สภาวะของแฟล็กนี้ (TF0 และ TF1) ซึ่งเป็นบิตอยู่ภายในรีจิสเตอร์ TCON เพื่อตรวจสอบว่ากระบวนการนับได้เสร็จสิ้นลงแล้ว หรือใช้เพื่อทำการอินเตอร์รัปต์โปรแกรมต่อไป

การจับเวลาในโหมด 0 : การจับเวลาในโหมด 0 นี้ วงจรนับ/จับเวลาจะทำหน้าที่เป็นตัวนับขนาด 13 บิต (โดยใช้รีจีสเตอร์ TH0 หรือ TH1 เป็นตัวนับขนาด 8 บิต และรีจีสเตอร์ TL0 หรือ TL1 มีขนาด 5 บิต)

การจับเวลาในโหมด 1 : การทำงานในโหมด 1 มีความคล้ายคลึงกับโหมด 0 มากเพียงแตกต่างกับเฉพาะจำนวนบิตของการนับเท่านั้น ซึ่งรีจีสเตอร์ของวงจรนับ/จับเวลาจะถูกใช้เต็มทุกบิตในลักษณะของการนับแบบ 16 บิตล้วน โดยค่าภายในรีจีสเตอร์ TH0 (หรือ TH1) จะเก็บค่าไบต์บน (High-order byte) ของตัวเลข

การจับเวลาในโหมด 2 : การทำงานในโหมด 2 ของวงจรนับ/จับเวลาจะมีความพิเศษต่างออกไป กล่าวคือจะมีเพียงการใช้รีจีสเตอร์ TL0 (หรือ TL1) เป็นตัวนับขนาด 8 บิตเท่านั้น ส่วนรีจีสเตอร์ TH0 (หรือ TH1) ใช้สำหรับทำหน้าที่เก็บค่าเริ่มต้นของการนับไว้ เมื่อรีจีสเตอร์ TL0 (หรือ TL1) เกิดการโอเวอร์โฟลว์จากค่า 0FFH เป็น 00H ระบบจะทำการนำค่าจากรีจีสเตอร์ TH0 (หรือ TH1) กลับมาใส่ให้โดยอัตโนมัติ (Autonomatic reload)

ชื่อบิต : TMOD **ตำแหน่ง : 89H** **ค่าบิตเริ่มต้น : 0000 0000**

GATE1	C/T1	M1	M0	GATE0	C/T0	M1	M0

ชื่อบิต	ตำแหน่ง	ความหมาย
GATE1	TMOD.7	บิตควบคุม GATE สำหรับ Timer 1
C/T1	TMOD.6	บิตกำหนดการทำงานแบบตัวนับหรือจับเวลาของ timer 1 โดยถ้าเป็นค่า 0 จะทำหน้าที่เป็นตัวจับเวลา
M1	TMOD.5	บิตบนสำหรับการกำหนดโหมดทำงานของ Timer 1
M0	TMOD.4	บิตล่างสำหรับการกำหนดโหมดทำงานของ Timer 1
GATE0	TMOD.3	บิตควบคุม GATE สำหรับ Timer 0
C/T0	TMOD.2	บิตกำหนดการทำงานแบบตัวนับหรือจับเวลาของ timer 0 หากเป็นค่า 0 จะทำหน้าที่เป็นตัวจับเวลา
M1	TMOD.1	บิตบนสำหรับการกำหนดโหมดทำงานของ Timer 0
M0	TMOD.0	บิตล่างสำหรับการกำหนดโหมดทำงานของ Timer 0

รูปที่ 2.2 บิตต่างๆ ภายในรีจีสเตอร์ TMOD (Timer/Counter Mode Control)

การจับเวลาในโหมด 3 : การทำงานในโหมด 3 จะสามารถใช้ได้เฉพาะกับ Timer 0 เท่านั้น หากว่านำไปใช้กำหนดให้กับ Timer 1 จะทำให้หยุดการทำงานไป เมื่อ Timer 0 ได้รับการกำหนดทำงานในโหมด 3 จะมีผลทำให้รีจีสเตอร์ของมันแยกกันทำงานเป็นอิสระ โดยรีจีสเตอร์ TL0 จะถูกควบคุมจากบิตภายในรีจีสเตอร์ TCON และขาสัญญาณ INT0 เมื่อมีการโอเวอร์โฟลว์เกิดขึ้นจากค่า 0FFH เป็น 00H ก็จะมีผลให้แฟล็ก TF0 มีการเปลี่ยนแปลงเกิดขึ้น สำหรับรีจีสเตอร์ TH0 จะถูกกำหนดให้ทำงานในแบบของตัวจับเวลา ภายใต้การควบคุมของบิต TR1 ในรีจีสเตอร์ TCON เท่านั้น และหากเกิดการโอเวอร์โฟลว์ขึ้นจะมีผลเฉพาะต่อแฟล็ก TF1 ในส่วนของ Timer 1 ขณะเมื่อ Timer 0 ถูกกำหนดให้ทำงานในโหมด 3 ก็ยังจะสามารถทำงานในโหมดอื่นๆ ที่ไม่ใช่โหมด 3 ได้เช่นเดิม ยกเว้นจะไม่มีการอินเตอร์รัปต์เกิดขึ้นเท่านั้น

ชื่อบิต : TCON **ตำแหน่ง : 88H** **ค่าบิตเริ่มต้น : 0000 0000**

TF1	TR1	TF0	TR0	IE1	IT1	IE0	IT0

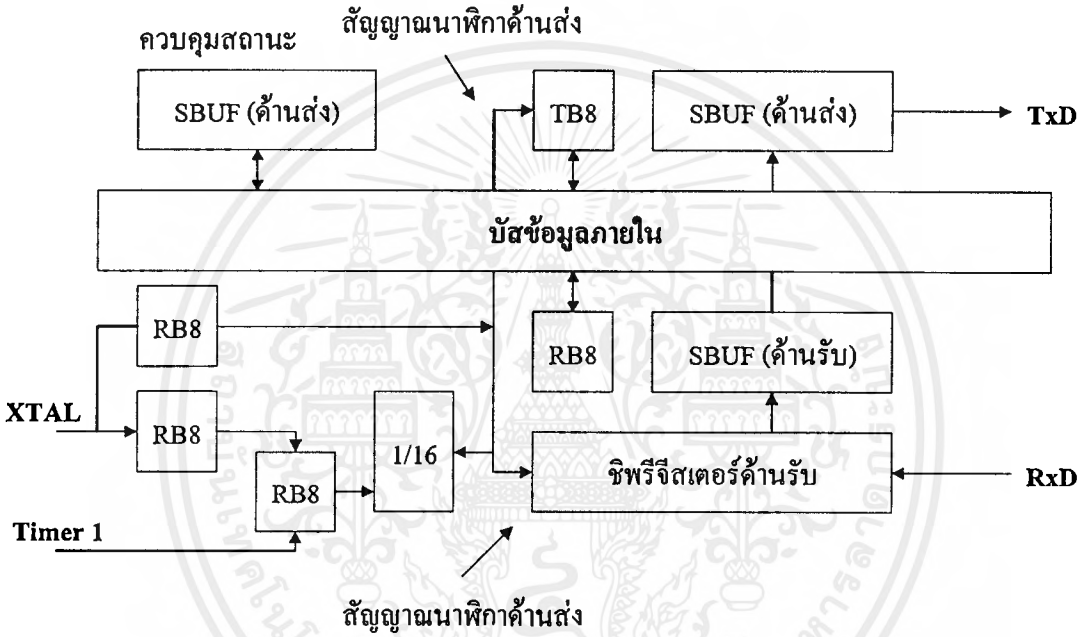
ชื่อบิต	ตำแหน่ง	ความหมาย
TF1	TCON.7	แฟล็กแสดงการอินเตอร์รัปต์ของ Timer 1
TR1	TCON.6	บิตเลือกประเภทสัญญาณอินเตอร์รัปต์ Timer 1
TF0	TCON.5	แฟล็กแสดงการอินเตอร์รัปต์ของ Timer 1
TR0	TCON.4	บิตเลือกประเภทสัญญาณอินเตอร์รัปต์ Timer 0
IE1	TCON.3	แฟล็กแสดงการอินเตอร์รัปต์ของ Timer 0
IT1	TCON.2	บิตเลือกประเภทสัญญาณอินเตอร์รัปต์ของ INT1
IE0	TCON.1	แฟล็กแสดงการอินเตอร์รัปต์ของ INT0
IT0	TCON.0	บิตเลือกประเภทสัญญาณอินเตอร์รัปต์ INT0

รูปที่ 2.3 บิตต่างๆ ภายในรีจีสเตอร์ TCON (Timer/Counter Control)

2.1.2 การจัดการข้อมูลอนุกรมของ MCS-51

พอร์ตอนุกรมของ MCS-51 มีโครงสร้างการทำงานในแบบที่เรียกว่า ฟูลดูเพล็กซ์ (Full Duplex) ซึ่งหมายถึงความสามารถในการรับและส่งข้อมูลอนุกรมได้ในเวลาเดียวกัน จากรูปที่ 2.4 แสดงให้เห็นถึงแผนภาพการทำงานอย่างง่ายของวงจรส่วนจัดการข้อมูลอนุกรมของ MCS-51 โดย

ทางด้านวงจรของตัวส่ง (Transmitter) ประกอบด้วยรีจีสเตอร์ SBUF ทำหน้าที่เก็บข้อมูลที่จะส่งออก การใช้คำสั่งเขียนหรือโอนย้ายข้อมูลมายังรีจีสเตอร์นี้ จะเป็นการส่งข้อมูลนั้นออกไปยังพอร์ตอนุกรมทางขาสัญญาณ TxD (พอร์ต 3.1) โดยอัตโนมัติ ส่วนวงจรทางด้านตัวรับ (Receiver) ประกอบด้วยรีจีสเตอร์ SUBF เช่นเดียวกัน แต่หน้าที่เก็บข้อมูลที่นำมาจากส่วนของวงจรเลื่อนบิตหรือชิพรีจีสเตอร์ (Shift register) ของวงจรจัดการข้อมูลอนุกรมภายใน สัญญาณข้อมูลอนุกรมที่รับเข้าผ่านมาทางขาสัญญาณ RxD (พอร์ต 3.0)



รูปที่ 2.4 แผนภาพแสดงการทำงานของวงจรส่วนการรับและส่งข้อมูลอนุกรมของ MCS-51

จากแผนภาพในรูปที่ 2.4 ชิพรีจีสเตอร์ภายในตัวส่งจะทำหน้าที่ในการเลื่อนบิตข้อมูลออกไปภายนอกโดยไม่มีการบัฟเฟอร์ และเมื่อใดที่มีการเขียนข้อมูลในกับรีจัสเตอร์ SBUF แสดงว่ามีความต้องการที่จะส่งข้อมูลนี้ออกไปแบบอนุกรม สำหรับรีจีสเตอร์ทางด้านรับจะทำการเลื่อนบิตข้อมูลที่รับเข้ามาเก็บไว้ เมื่อบิตของข้อมูลที่รับเข้ามาครบถ้วนตามจำนวนที่กำหนดไว้ตามลักษณะโหมดการทำงานต่างๆ แล้ว จะถูกย้ายไปเก็บยังรีจีสเตอร์ SBUF ต่อไป อย่างไรก็ตามการย้ายข้อมูลนี้จะเกิดขึ้นก็ต่อเมื่อรีจีสเตอร์ SBUF นั้นไม่มีข้อมูลที่จะทำการส่งหรือได้ส่งข้อมูลออกไปเสร็จสิ้นแล้ว

พอร์ตอนุกรมของ MCS-51 สามารถโปรแกรมให้ทำหน้าที่ในรูปแบบต่างๆ กันสี่แบบ (หรือเรียกว่าโหมดการทำงาน) โดยการกำหนดบิต SM0 และ SM1 ซึ่งอยู่ภายในรีจีสเตอร์ควบคุมและบอกสถานะ SCON ดังแสดงในรูปที่ 2.5 โหมดการทำงานทั้ง 4 แบบของพอร์ตอนุกรม มีดังนี้

โหมดทำงาน	คำอธิบาย
โหมด 0	เป็นการขยายพอร์ตอินพุตเอาร์พุต โดยทำงานร่วมกับไอซีชิพรีจีสเตอร์ภายนอกประเภททีทีแอลหรือซีมอส
โหมด 1	ใช้สำหรับการเชื่อมต่ออนุกรมแบบ UART (Univerrsail asynchronous receiver/transmitter) โดยการใช้กลุ่มข้อมูลแบบ 10 บิต และสามารถเปลี่ยนแปลงอัตราเร็วในการส่งข้อมูลได้
โหมด 2	ใช้สำหรับการเชื่อมต่ออนุกรมแบบ UART โดยการใช้กลุ่มข้อมูลแบบ 11 บิต และกำหนดอัตราความเร็วในการส่งข้อมูลคงที่
โหมด 3	ใช้สำหรับการเชื่อมต่ออนุกรมแบบ UART โดยการใช้กลุ่มข้อมูลแบบ 11 บิต และสามารถเปลี่ยนแปลงอัตราเร็วในการส่งข้อมูลได้

ชื่อบิต : SCON **ตำแหน่ง : 98H** **ค่าบิตเริ่มต้น : 0000 0000**

SM0	SM1	SM2	REN	TB8	RB8	TI	RI

ชื่อบิต	ตำแหน่ง	ความหมาย
SM0	SCON.7	บิตเลือกโหมดการทำงาน
SM1	SCON.6	บิตเลือกโหมดการทำงาน
SM2	SCON.5	แฟล็กกำหนดการทำงานแบบมัลติโปรเซสเซอร์
REN	SCON.4	แฟล็กยอมให้มีการรับข้อมูล
TB8	SCON.3	ค่าของบิตที่ 9 สำหรับการส่งข้อมูลออก
RB8	SCON.2	ค่าของบิตที่ 9 ของข้อมูลที่รับเข้า
TI	SCON.1	แฟล็กแสดงการอินเตอร์รัปต์ภายหลังการส่งข้อมูล
RI	SCON.0	แฟล็กแสดงการอินเตอร์รัปต์เมื่อมีข้อมูลรับเข้า

รูปที่ 2.5 รีจีสเตอร์ควบคุมการทำงานและบอกสถานะการสื่อสารข้อมูลอนุกรม SCON

เนื่องจากการส่งหรือรับข้อมูลอนุกรมในการส่งข้อมูลไบต์หนึ่งๆ ค่อนข้างจะใช้เวลานานหลายมิลลิวินาที ดังนั้นเพื่อให้การจัดการเกี่ยวกับการสื่อสารแบบนี้เป็นไปอย่างมีประสิทธิภาพ MCS-51 จึงได้กำหนดให้บิตหรือแฟล็กสถานะที่เกี่ยวข้องทั้งหมดจัดรวมอยู่ภายในรีจีสเตอร์ SCON เท่านั้น เช่น แฟล็ก TI ซึ่งมีค่าเป็น 1 เมื่อมีข้อมูลได้ทำการส่งออกไปภายนอกเสร็จสิ้นแล้ว และแฟล็ก RI ซึ่งจะมีค่าเป็น 1 เพื่อแจ้งให้ทราบว่ารับข้อมูลผ่านเข้ามาทางพอร์ตอนุกรม เมื่อแฟล็กตัวใดตัวหนึ่งนี้มีค่าเป็น 1 จะมีผลทำให้เกิดการอินเตอร์รัปต์ขึ้น ดังนั้นภายในโปรแกรมจะต้องทำการตรวจสอบจากสภาวะของแฟล็กเหล่านี้เองว่ามีการอินเตอร์รัปต์ขึ้นด้วยสาเหตุใด จากนั้นจึงค่อยทำการกำหนดค่า 0 ให้กับแฟล็กนั้น ลักษณะดังกล่าวนี้จะมีความแตกต่างไปจากการอินเตอร์รัปต์สัญญาณอื่นๆ เช่น วงจรนับ/จับเวลา เป็นต้น ซึ่งจะมีการกำหนดค่า 0 ให้กับแฟล็กสถานะที่เกี่ยวข้องโดยอัตโนมัติ ภายหลังจากที่ได้เข้าไปทำงานยังส่วนของโปรแกรมย่อยบริการอินเตร์รัปต์

การส่งข้อมูลออกทางพอร์ตอนุกรมของ MCS-51 จะเริ่มต้นขึ้นภายหลังเมื่อมีการเขียนข้อมูลลงในรีจีสเตอร์ SBUF ข้อมูลนี้จะถูกจัดการด้วยวิธีการทางด้านฮาร์แวร์ในการเลื่อนบิตและส่งสัญญาณออกไปภายนอกโดยอัตโนมัติ เมื่อข้อมูลเหล่านี้ได้ส่งออกครบถ้วนแล้ว จึงจะทำการกำหนดค่าแฟล็ก TI ให้เป็น 1 เพื่อแจ้งให้ทราบว่าขณะนี้รีจีสเตอร์ SBUF ว่าง และพร้อมที่จะส่งข้อมูลไบต์ต่อไปแล้ว ในกรณีที่ผู้ใช้เขียนข้อมูลใหม่ลงในรีจีสเตอร์ SBUF โดยไม่รอให้แฟล็ก TI มีค่าเป็น 1 ก่อนจะมีผลทำให้ข้อมูลที่ส่งออกไปผิดพลาดได้

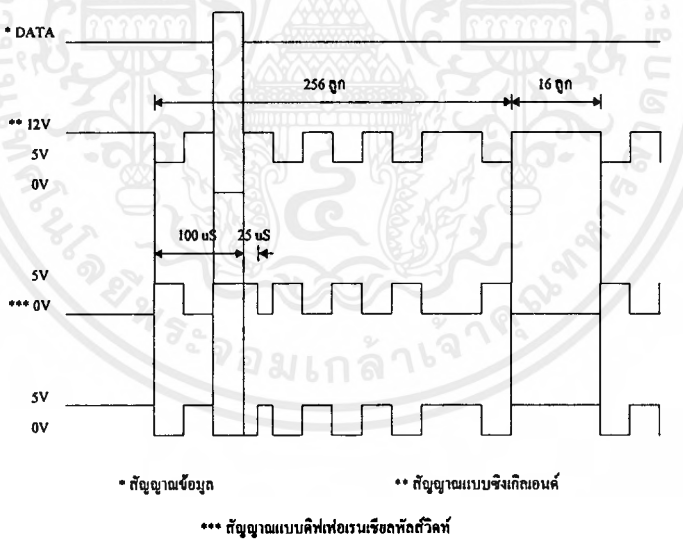
สำหรับการรับข้อมูลจากพอร์ตอนุกรมจะต้องเริ่มต้นโดยการกำหนดค่าบิต REN (Receiver Enable) ให้มีค่าเป็น 1 ก่อน หลังจากนั้นเมื่อมีบิตของข้อมูลถูกส่งเข่ามาจากภายนอกระบบฮาร์แวร์ของ MCS-51 จึงจะทำการเลื่อนบิตเหล่านี้เข้ามาโดยอัตโนมัติ และเมื่อบิตสุดท้ายถูกเลื่อนเข้ามาเรียบร้อยแล้ว ข้อมูลนั้นจะถูกย้ายมาเก็บไว้ยังรีจีสเตอร์ SBUF และทำการกำหนดให้แฟล็ก RI ให้มีค่าเป็น 1 ซึ่งมีผลทำให้เกิดการอินเตอร์รัปต์โปรแกรมขึ้น

ในการเชื่อมต่อแบบอนุกรมเข้ากับอุปกรณ์คอมพิวเตอร์ต่างๆ เช่น คอมพิวเตอร์ เทเลกซ์ หรือโทรพิมพ์ เป็นต้น มักจะกำหนดใช้การเชื่อมต่อมาตรฐาน RS-232C ทั้งนี้เพื่อให้มีการใช้งานเส้นสัญญาณหรือรูปแบบของตัวเชื่อมต่อที่สอดคล้องกัน จะได้ลดปัญหาการเข้ากันไม่ได้ระหว่างสัญญาณของอุปกรณ์ที่มาเชื่อมต่อกันทั้งสองด้านให้น้อยลง เนื่องจากระดับโวลเตจที่ใช้และการแทนความหมายของระดับลอจิกตามมาตรฐานนี้แตกต่างออกไปจากที่ใช้งานกันในระบบดิจิตอลทั่งไป โดยระดับสัญญาณของ RS-232C เป็นแบบไบโพล่าร์ (Bopolar) ระดับโวลเตจทางด้านช่วงลบ -3V ถึง -20V แทนค่าลอจิก 1 และโวลเตจทางด้านบวกช่วง +3V ถึง +20V แทนค่าลอจิก 0 ดังนั้นจะเห็นได้ว่ามีความจำเป็นจะต้องเพิ่มเติมอุปกรณ์หรือวงจรพิเศษเข้าไป เพื่อเปลี่ยนระดับโวลเตจ

จากระบบ 0V ถึง +5V จากขาสัญญาณของ MCS-51 เป็นระดับโวลเตจที่สูงกว่าค่า +3.0V หรือต่ำกว่า -3.0V

2.2 การควบคุมและแสดงผลแบบพัลส์วิดท์ดิฟเฟอเรนเชียลซีเควนเชียลคอนโทรล (Pulse Width Differential Sequential Control and Monitoring)

จากระบบควบคุมและแสดงผลระยะไกลจะเป็นการส่งสัญญาณ 3 ระดับคือ 12V ,5V และ 0V ซึ่งจะเป็นสัญญาณนาฬิกาที่ระดับ 12V ,5V และข้อมูลที่ระดับ 0V จะเป็นการส่งสัญญาณนาฬิกา 256 ลูก และเป็นสัญญาณว่างที่ระดับ 12V ที่มีความกว้างเท่ากับ 16ลูก โดยมีสายส่งสัญญาณ 2 เส้นคือสายสัญญาณและกราวด์ของระบบ ซึ่งเป็นสายส่งสัญญาณแบบซิงเกิลเอนด์ สำหรับการส่งสัญญาณแบบดิฟเฟอเรนเชียลที่พัฒนาขึ้นมาใหม่นี้ จะมีการส่งสัญญาณ 2 ระดับคือ 5V และ 0V ซึ่งจะมีการส่งสัญญาณนาฬิกา 256 ลูกและเป็นสัญญาณว่างที่มีความกว้างเท่ากับ 16 ลูก ในระดับ 0V ส่วนในการส่งข้อมูลจะเป็นการเพิ่มความกว้างให้กับสัญญาณ Pulse ที่ต้องการดังรูปที่ 2.6

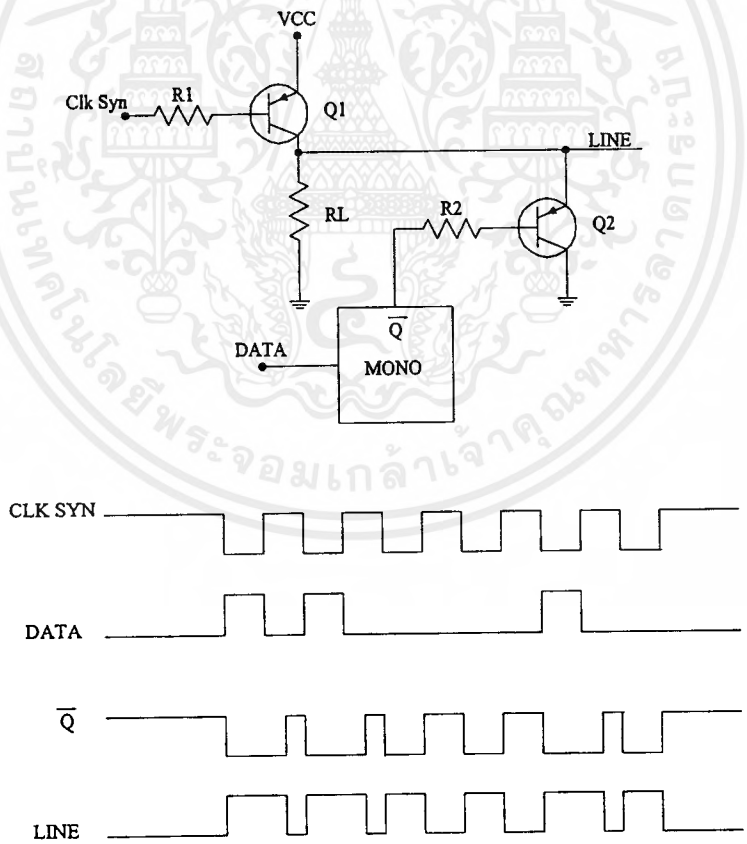


รูปที่ 2.6 แสดงลักษณะการส่งสัญญาณแบบ Sequential แบบเก่ากับแบบดิฟเฟอเรนเชียลที่พัฒนาขึ้น

จากรูปที่ 2.7 เป็นวงจรสำหรับส่งข้อมูลพัลส์วิทดิ์มอดูเลชั่นซีเครนเชียลคอนโทรลแบบซิงเกิลเอนด์ ซึ่งเป็นการส่งข้อมูลแบบง่ายๆ ที่ใช้ในการพัฒนาในการส่งข้อมูล จากรูปเมื่อป้อนสัญญาณ Clf Syn. ให้อินพุท Q1 ทางเอาท์พุทจะได้สัญญาณที่กลับเฟสกับสัญญาณทางอินพุท ซึ่งสัญญาณทางเอาท์พุทที่ได้นี้จะเป็นสัญญาณเพื่อใช้ในการซิงโครไนส์ (Syncronize) ระบบทั้งหมด

โดยจะถูกส่งไปในสายนำสัญญาณทำให้ได้สัญญาณ (Line) และ Monostable จะถูก Trig จากขอบขาขึ้นของสัญญาณ Data ทำให้ได้สัญญาณ Q ที่มีความกว้างของพัลส์กว้างกว่าสัญญาณ Clk Syn ดังกล่าว สัญญาณ Q ที่ได้มานี้จะเอาไปป้อนให้กับ Q2 ซึ่งทำหน้าที่สร้างสัญญาณ Data เข้าไปในสัญญาณ Clk Syn. ในสัญญาณ Line ในสายสัญญาณจะเป็นดังรูปสัญญาณที่แสดงไว้ในรูปที่ 2.6

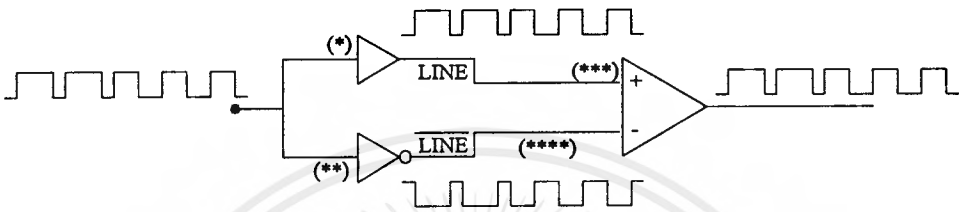
การส่งสัญญาณแบบดิฟเฟอเรนเชียล จะเป็นการสร้างสัญญาณที่มีระดับของสัญญาณตรงข้ามกันส่งไปในสายนำสัญญาณ 2 เส้น ส่วนในการตรวจสัญญาณจะใช้วงจรเปรียบเทียบระดับสัญญาณ (Comparator) ในการตรวจจับสัญญาณผลต่างของสัญญาณในสายนำสัญญาณ ดังรูปที่ 2.8 ซึ่งสัญญาณที่ตรวจจับได้จะเหมือนสัญญาณอินพุทก่อนที่จะส่งเข้ามาในสายนำสัญญาณ ข้อดีของการส่งแบบนี้คือไม่ว่าสัญญาณในสายส่งจะมีการเปลี่ยนแปลงอย่างไรก็ตาม สัญญาณที่ตรวจจับได้ก็ยังคงเหมือนกับสัญญาณที่ต้นทางทั้งยังป้องกันสัญญาณรบกวนภายนอกได้ดีเนื่องจากสัญญาณภายในสายส่งเป็นแบบดิเฟอเรนเชียล



รูปที่ 2.7 วงจรส่งข้อมูลและแผนภาพสัญญาณ

พัลส์วิดท์มอดูเลชั่นซีเควนเชียลคอนโทรลแบบซิงเกิลเอนด์

ดังนั้นการพัฒนาการส่งข้อมูลแบบซีเครนเชียลคอนโทรลนี้ จะสามารถแก้ปัญหาที่เกิดขึ้นในระบบสายส่งสัญญาณในระบบเก่าที่ระดับของสัญญาณจะลดลงเมื่อต่อกับสายสัญญาณที่ยาวมากๆ ซึ่งเทคนิคการส่งแบบดิฟเฟอเรนเชียลนี้ สามารถนำไปใช้ในการส่งข้อมูลในระบบควบคุมและแสดงผลระยะไกลได้ หรือจะใช้แทนการส่งข้อมูลแบบเนกกาทีฟพัลส์ก็ได้



* วงจรขับสัญญาณแบบบวก ** วงจรขับสัญญาณแบบลบ

*** วงจรเปรียบเทียบระดับสัญญาณ **** สายส่งสัญญาณ

รูปที่ 2.8 แผนภาพของระบบส่งแบบดิฟเฟอเรนเชียลและการตรวจจับสัญญาณ

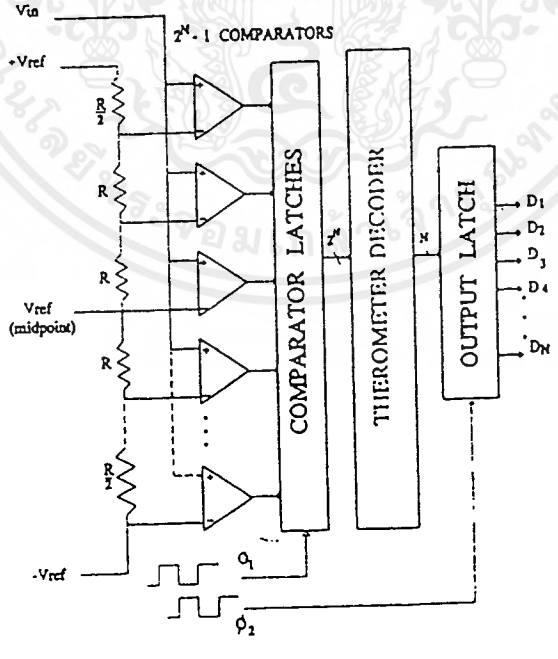
บทที่ 3
วงจรแปลงสัญญาณอนาลอกเป็นดิจิตอล

วงจรเปลี่ยนสัญญาณอนาลอกเป็นดิจิตอลมีหลายแบบด้วยกันคือ

3.1 แบบใช้วงจรเปรียบเทียบขนานหรือ “แฟลช”

(Parallel Comparator Simultaneous “Flash” A/D Converter)

วงจร A/D แบบนี้ใช้หลักการง่ายๆอีกทั้งยังเป็นวิธีที่รวดเร็วที่สุด คือใช้วงจรเปรียบเทียบที่ต่อขนานกัน ดังรูปที่ 3.1 ซึ่งประกอบด้วยออปแอมป์ที่ต่อเป็นวงจรเปรียบเทียบ และตัวต้านทานที่ต่อไว้เพื่อแบ่งแรงดันที่ขาอินพุทแบบกลับ (Inverting)ให้มีขนาดต่างๆกัน จากหลักการของวงจรเปรียบเทียบทั่วไป เมื่อแรงดันอินพุทที่ขาอินพุทแบบไม่กลับ (Non-inverting) มีค่าสูงกว่าที่ขาอินพุทแบบกลับ เอาพุทต์จะได้แรงดันค่าสูง



รูปที่3.1 โครงสร้างของแฟลช A/D Converter

เมื่อต้องการวงจรที่มีความละเอียดสูงขึ้น จำเป็นต้องใช้วงจรเปรียบเทียบเพิ่มขึ้น เช่น ถ้าต้องการความละเอียด 3 บิต ต้องใช้วงจรเปรียบเทียบ 7 ตัว และถ้าต้องการความละเอียด 4 บิต ก็ต้องใช้วงจรเปรียบเทียบ 15 ตัว (16 ระดับ) โดยหาจำนวนวงจรเปรียบเทียบได้จาก 2^N-1 เมื่อ N แทนจำนวนบิตหรือความละเอียดที่ต้องการ

จะเห็นได้ว่าที่ความละเอียด 8 บิต ต้องใช้วงจรเปรียบเทียบมากถึง 255 ตัว ซึ่งเป็นข้อเสียของวงจร A/D แบบนี้ และข้อเสียอีกประการหนึ่งคือ เอาพุทต์ที่ไม่ได้เป็นเลขฐานสองต้องมีวงจรเพิ่มเติมไปทำการเข้ารหัส

ส่วนข้อดีของวงจร A/D ขนานนี้คือมีความเร็วในการทำงานสูงมาก บางครั้งจึงเรียกวงจร A/D แบบนี้ว่า "แฟลช" (Flash Type A/D Converter) โดยใช้เวลาในการแปลงได้ถึงระดับนาโนวินาทีทีเดียว

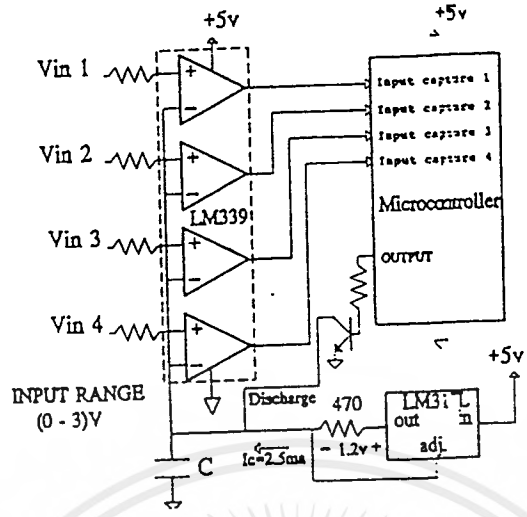
3.2 วงจร A/D ที่ใช้การอินทิเกรต

วงจร A/D ที่ใช้หลักการการอินทิเกรตมีอยู่ 3 แบบด้วยกันคือ

3.2.1 แบบสโลปเดี่ยวหรือแบบแรมป์ (Single Ramp หรือ Single Slope A/D Converter)

วงจร A/D แบบนี้แสดงไว้ดังรูปที่ 3.2 ซึ่งประกอบด้วยวงจรกำเนิดสัญญาณแรมป์ ,วงจรเปรียบเทียบ ,วงจรนับ BCD หรือ วงจรนับเลขฐานสอง

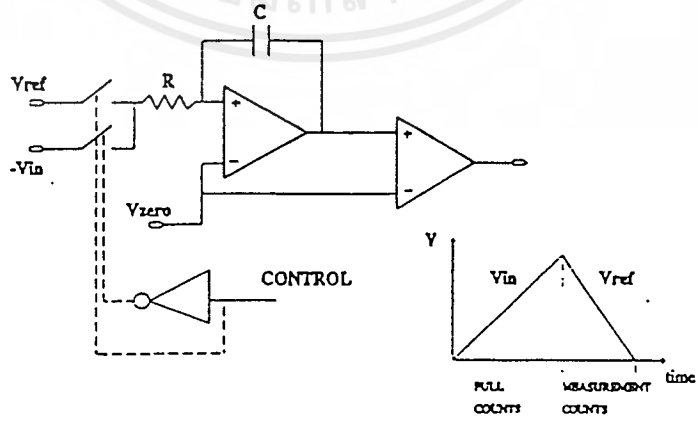
เมื่อเริ่มทำการเปลี่ยนสัญญาณ สัญญาณแรมป์และวงจรนับจะถูกรีเซตให้เป็นศูนย์ แรงดันอนาลอกจะถูกป้อนไปยังวงจรเปรียบเทียบทางขาอินพุทแบบไม่กลับ (Non-inverting) เมื่อแรงดันอินพุทที่ขานี้เป็นบวกมากกว่าขาอินพุทแบบกลับ (Inverting) วงจรเปรียบเทียบจะให้เอาพุตท์เป็นระดับ "High" ทำให้แอนด์เกตปล่อยสัญญาณนาฬิกาไปยังวงจรนับได้ และทำให้เริ่มเกิดสัญญาแรมป์ สัญญาณแรมป์มีแรงดันเป็นบวกขึ้นเรื่อยๆ จนมากกว่าระดับแรงดันอินพุท เอาพุตท์ ของวงจรเปรียบเทียบ ก็ตกลงมาเป็นระดับ "Low" ทำให้แอนด์เกตถูกปิด จึงไม่มีสัญญาณผ่านไปให้วงจรนับ วงจรนับจะหยุดนับและเก็บค่าไว้ที่วงจรแลต จากนั้นจึงทำการรีเซตวงจรนับและวงจรกำเนิดสัญญาณแรมป์ วงจรแบบนี้เป็นหลักการเบื้องต้นของดิจิตอลโวลต์มิเตอร์ ซึ่งถ้าใช้วงจรนับเลขฐานสองแทนแบบ BCD เอาพุทต์ก็จะอ่านเป็นได้เลขฐานสอง



รูปที่ 3.2 Low cost , 4 channel , Single Slope A/D Converter

3.2.2 แบบสโลปคู่ (Dual Slope A/D Converter)

วงจรของ A/D แบบสโลปคู่แสดงได้ดังรูปที่ 3.3 ซึ่งวงจรส่วนใหญ่คล้ายกับแบบสโลปเดี่ยว แต่มีสวิตซ์ที่อินพุทเพิ่มขึ้นเพื่อทำการเลือกระหว่างแรงดันอินพุทกับแรงดันอ้างอิง ข้อดีที่เหนือกว่าแบบสโลปเดี่ยวคือค่าที่ได้ไม่ขึ้นกับความถี่ของรอบการทำงาน มีความถูกต้องสูง ราคาถูก เสถียรภาพทางด้านอุณหภูมิดี แต่มีข้อเสียคือความเร็วในการทำงานต่ำ



รูปที่ 3.3 วงจร A/D แบบสโลปคู่

3.2.3 แบบชาร์จบาลานซ์ (Charge Balance A/D Converter)

วงจร A/D แบบนี้ใช้วงจรคล้ายกับแบบสโลปคู่ แต่แทนที่จะให้อินพุทสวิตซ์ไปมาระหว่างแรงดันที่ไม่รู้ค่ากับแรงดันอ้างอิง ก็ทำการแทรกพัลส์ของกระแสอ้างอิงมาตรงๆที่จุดรวมของวงจรอินทิเกรเตอร์ในช่วงเวลาที่คงที่ โดยที่จำนวนของพัลส์จะเป็นสัดส่วนโดยตรงกับแรงดันอินพุทที่ไม่รู้ค่า ประโยชน์ของเทคนิคนี้คือ แรงดันตกคร่อมตัวเก็บประจุของวงจรอินทิเกรเตอร์จะมีค่าใกล้เคียงศูนย์โวลต์ ดังนั้นจึงไม่เกิดความผิดพลาดจากผลของกระแสรั่วไหล A/D แบบนี้จึงมีความถูกต้องสูงกว่าแบบสโลปคู่

3.2.4 แบบเดลต้า-ซิกมา (Delta- Sigma)

วงจรของ A/D แบบนี้แสดงได้ดังรูปที่ 3.4 เมื่อมีแรงดันอินพุทป้อนเข้าไปที่วงจรอินทิเกรเตอร์เอาพุทต์ที่ได้จะไปเข้าวงจรเปรียบเทียบ เพื่อทำการเปรียบเทียบกับแรงดันคงที่ (จากรูปคือกราวด์) พัลส์ของกระแสที่ได้ขึ้นอยู่กับเอาพุทต์ของวงจรเปรียบเทียบ โดยสวิตซ์ที่ทำขึ้นจากเฟตจะควบคุมให้กระแสเข้าไปยังที่จุดรวมหรือลงกราวด์ไป ส่วนวงจรนับจะนับจำนวนพัลส์ด้วยหลักการที่คล้ายกัน

3.3 วงจร A/D ที่ใช้วงจรนับและวงจร D/A ประกอบกัน

3.3.1แบบวงจรนับเดี่ยว (Single Counter)

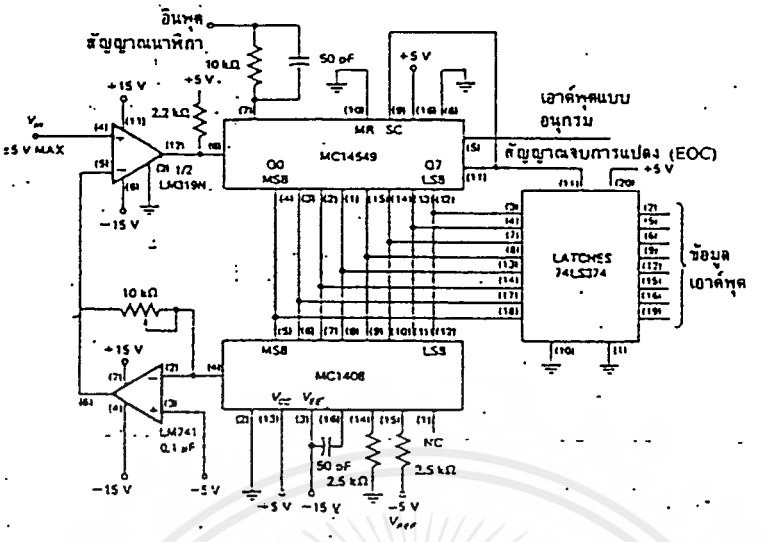
แท้ที่จริงแล้วสัญญาณแรมป์เชิงเส้นอาจประกอบขึ้นด้วยสัญญาณขั้นบันไดเล็กๆจำนวนมากที่เกิดจากการต่อเอาพุทต์ของวงจรนับเข้ากับวงจรแปลง D/A โดยขนาดของขั้นบันไดแต่ละขั้นขึ้นอยู่กับจำนวนบิตหรือความละเอียดของวงจร D/A นั้นๆ

3.3.2แบบแทรคกิ้ง (Tracking A/D Converter)

การทำงานจะคล้ายกับแบบใช้วงจรนับเดี่ยว แต่การนับจะไม่ได้เริ่มจากศูนย์แต่จะทำการนับขึ้นหรือนับลงจากค่าล่าสุดไปยังค่าใหม่แล้วแต่ว่าแรงดันอินพุทในรอบใหม่มีค่าสูงกว่าหรือต่ำกว่าค่าที่แล้ว ข้อดีของ A/D แบบนี้คือทำได้เร็วขึ้น

3.4 วงจร A/D ที่ใช้การประมาณค่า (Successive Aproximation A/D Converter)

วงจร A/D แบบนี้มีข้อดีคือได้เปรียบทางด้านความละเอียด เพราะความละเอียด N บิต สามารถกำหนดได้จากค่าสัญญาณนาฬิกา N ลูก ถ้าเราต้องการความละเอียด 8 บิต จะต้องการพัลส์ของสัญญาณนาฬิกา 8 ลูก ในขณะที่ใช้แบบวงจรนับต้องใช้ถึง 256 ลูก วงจร SA (Successive Approximation) นี้แสดงไว้ในรูปที่ 3.4 หัวใจของวงจรคือ Successive Approximation Register (SAR) เช่นเบอร์ MC14549 ที่มีการทำงานดังต่อไปนี้



รูปที่ 3.4 วงจร A/D แบบ Successive Approximation

เมื่อเริ่มทำการเปลี่ยนสัญญาณ พัลส์ลูกแรกจะทำการส่งบิตที่มีนัยสำคัญสูงสุดไปยัง D/A เบอร์ MC1408 โดย SAR จะรอสัญญาณจากวงจรเปรียบเทียบ LM319 ซึ่งทำการตรวจสอบว่าเอาพุทต์ของวงจร D/A มากกว่าหรือน้อยกว่าแรงดันอินพุท Vin ถ้าเอาพุทต์ของวงจรเปรียบเทียบมีระดับ "high" เอาพุทต์ของ D/A จึงต่ำกว่า Vin ของ SAR ก็จะทำการเก็บบิตที่มีนัยสำคัญสูงสุดไว้ ถ้าเอาพุทต์ของวงจรเปรียบเทียบเป็นระดับ "low" เอาพุทต์ของวงจรเปรียบเทียบจึงมากกว่า Vin ของ SAR ก็จะทำการรีเซตบิตที่มีนัยสูงสุดนั้น

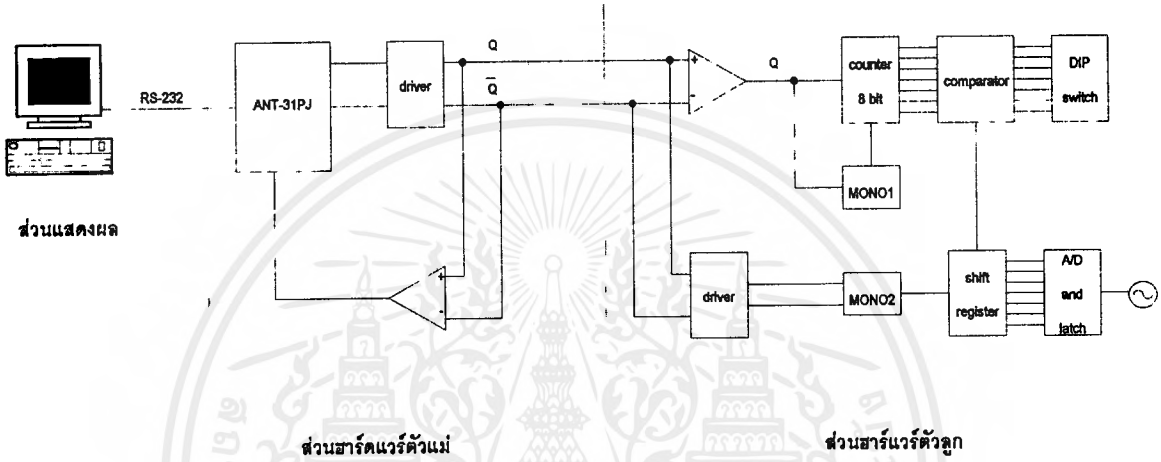
พัลส์ลูกต่อมาก็ทำเช่นเดียวกัน โดยบิตที่ได้คือบิตที่มีนัยสำคัญรองลงมา SAR ทำงานแบบนี้ไปจนถึงบิตที่มีนัยสำคัญต่ำสุด แต่ละบิตใช้สัญญาณนาฬิกาลูกเดียวครบทุกบิต แล้ว SAR ทำการส่งสัญญาณ EOC (End Of Conversion) ออกไป สัญญาณ EOC เป็นตัวบอกว่าสายสัญญารเอาพุทต์ที่ขนานกันมาทุกเส้นมีข้อมูลดิจิตอลของสัญญาณอินพุตครบถ้วนแล้ว ถ้าสัญญาณ EOC ถูกต่อไปยังอินพุทที่เป็นจุดเริ่มการเปลี่ยนสัญญาณ การเปลี่ยนสัญญาณก็จะเกิดขึ้นอย่างต่อเนื่อง วงจรA/D ชนิดนี้มีความเร็วและความละเอียดสูงจึงเป็นวงจรที่นำมาใช้กันอย่างแพร่หลาย

บทที่ 4

การออกแบบ

4.1 การทำงานโดยรวมของระบบ

พิจารณาวงจรการทำงานอย่างง่ายของระบบดังรูปที่ 3.1



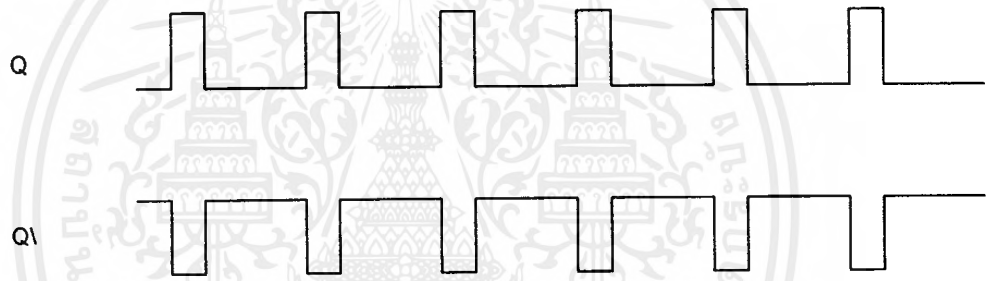
รูปที่ 4.1 ภาพวงจรโดยรวมของระบบ

ในระบบนี้จะแบ่งออกเป็น 3 ส่วนใหญ่ๆคือ ส่วนของฮาร์ดแวร์ตัวแม่ ฮาร์ดแวร์ตัวลูก และส่วนเครื่องคอมพิวเตอร์ โดยฮาร์ดแวร์ตัวแม่จะเป็นส่วนที่ใช้ในการควบคุมและจัดการการทำงานทั้งหมดของระบบ ในการเชื่อมต่อกันของฮาร์ดแวร์ตัวแม่และฮาร์ดแวร์ตัวลูกนั้นจะทำการเชื่อมต่อกันโดยใช้สายนำสัญญาณที่มีความยาวมากๆได้ นอกจากนี้ยังสามารถต่อกับฮาร์ดแวร์ตัวลูกได้หลายตัว โดยจะต่อได้มากที่สุดจำนวน 31 ตัว ฮาร์ดแวร์ตัวแม่จะทำหน้าที่ในการสร้างสัญญาณนาฬิกาเพื่อเป็นการจัดลำดับการส่งข้อมูลให้กับฮาร์ดแวร์ตัวลูกแต่ละตัว สัญญาณนาฬิกาที่ทางฮาร์ดแวร์ตัวลูกจะตรงกับสัญญาณนาฬิกาที่ฮาร์ดแวร์ตัวแม่ เมื่อถึงลำดับการส่งข้อมูลของตัวเองจะส่งข้อมูลซึ่งได้แปลงสัญญาณอนาลอกเป็นสัญญาณดิจิตอลเรียบร้อยแล้วส่งกลับไปยังสายนำสัญญาณ เมื่อฮาร์ดแวร์ตัวแม่ได้รับข้อมูลแล้วก็จะทำการรวบรวมข้อมูลเหล่านั้นเก็บลงในหน่วยความจำ เพื่อรอเวลาที่จะส่งไปแสดงผลยังเครื่องคอมพิวเตอร์ที่ติดต่อกันภายใต้มาตรฐาน RS-232 ที่เครื่องคอมพิวเตอร์เมื่อมีการรับข้อมูลผิดพลาดก็จะทำการร้องขอให้ฮาร์ดแวร์ตัวแม่ทำการส่งข้อมูลนั้นใหม่ (Retransmit) ภายในช่วงระยะเวลาที่กำหนด การแสดงผลบนเครื่องคอมพิวเตอร์นี้จะใช้โปรแกรมสำเร็จรูปที่ได้จัดทำขึ้นชื่อ Data Sequencial Collecting Display ซึ่งจะแสดงข้อมูลและ

กราฟความสัมพันธ์ระหว่างข้อมูลกับเวลาของฮอาร์ดแวร์ตัวลูกแต่ละตัวพร้อมด้วยตัวเลือก (Option) ต่างๆที่จะช่วยให้การวิเคราะห์ประเมินผลและเก็บข้อมูลเป็นไปได้สะดวกยิ่งขึ้น

4.2 การสร้างสัญญาณพัลส์วิดส์ซีเควนเชียลคอนโทรล

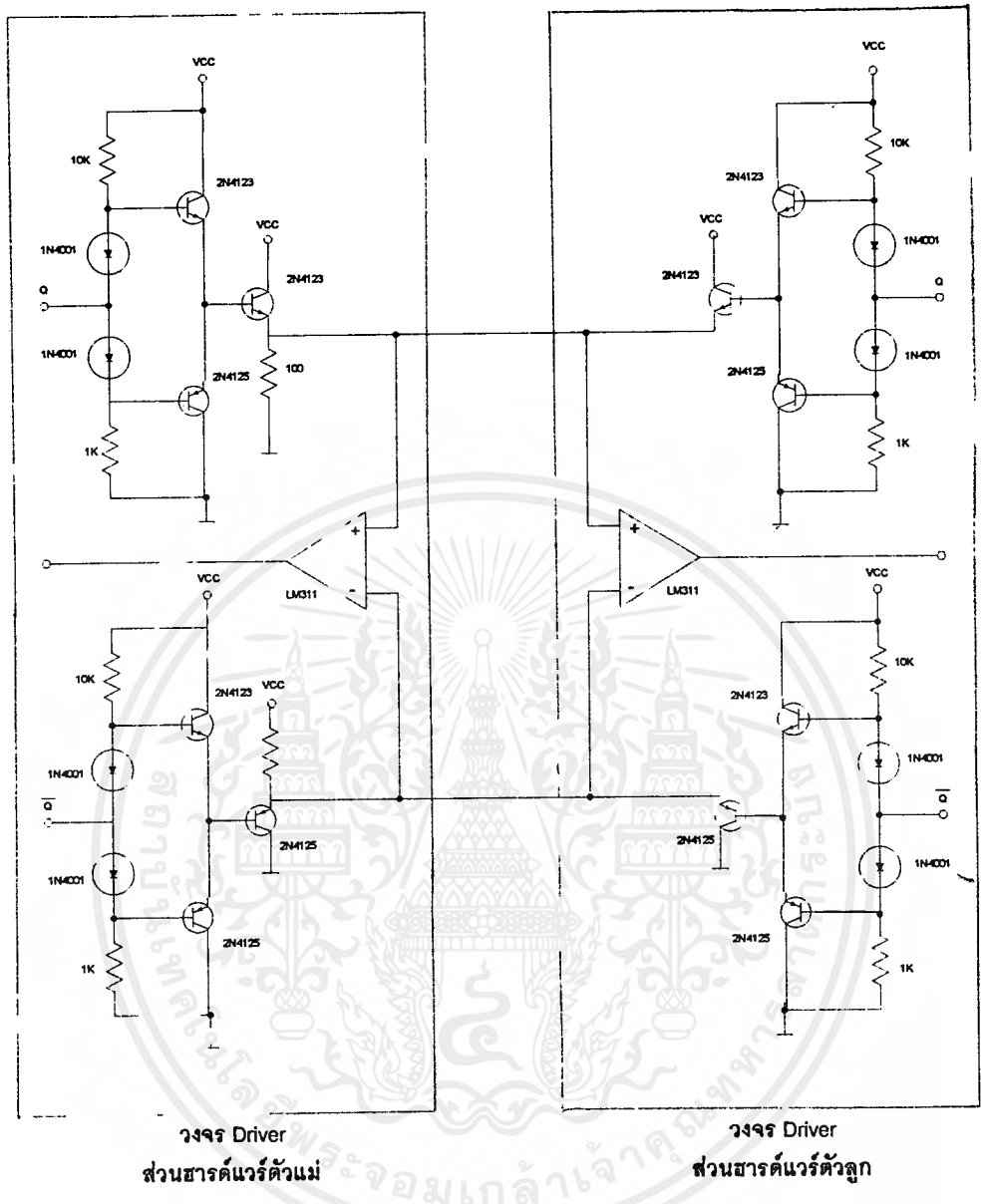
สัญญาณพัลส์วิดส์ซีเควนเชียลคอนโทรลนี้จะถูกสร้างขึ้นโดย Timer ภายในตัวไมโครคอนโทรลเลอร์ MCS-51 ซึ่งเป็นส่วนหนึ่งของฮาร์ดแวร์ตัวแม่ สัญญาณที่สร้างขึ้นนี้จะมีคาบเวลาประมาณ 1 ms และมีดิวตี้ไซเคิลประมาณ 25% รูปแบบของสัญญาณที่ใช้จะเป็นไปในลักษณะของพัลส์ 256 ลูก แล้วตามด้วยสัญญาณแบงค์ช่วงหนึ่ง ซึ่งจะเป็นลักษณะอย่างนี้ตลอดช่วงระยะเวลาการทำงานของระบบ สัญญาณเหล่านี้จะถูกส่งไปยังส่วนของฮาร์ดแวร์ตัวลูกผ่านสายนำสัญญาณในลักษณะดิฟเฟอเรนเชียลคือ มีทั้ง Q และ Q\ ดังรูป



รูปที่ 4.2 ภาพแสดงลักษณะสัญญาณของ Q และ Q\

เมื่อฮาร์ดแวร์ตัวลูกรับสัญญาณ Q และ Q\ ก็จะทำการเปรียบเทียบสัญญาณโดยใช้วงจรคอมพาราเตอร์ สัญญาณเอาพุทต์ที่ได้จะมีรูปร่างเหมือนสัญญาณ Q ทุกประการ ซึ่งสัญญาณนี้จะถูกนำไปเป็นสัญญาณนาฬิกาให้กับฮาร์ดแวร์ตัวลูกต่อไป สัญญาณพัลส์จำนวน 256 ลูกนี้ฮาร์ดแวร์ตัวลูกแต่ละตัวจะใช้สัญญาณนี้เพียงแค่ 8 ลูกเท่านั้น ดังนั้นจึงทำให้เราสร้างฮาร์ดแวร์ตัวลูกได้ถึง 32 ตัว แต่เนื่องจากเรานำสัญญาณพัลส์ 8 ลูกแรกไปใช้ในการรีเซตเคาเตอร์ของฮาร์ดแวร์ตัวลูก จึงทำให้จำนวนของฮาร์ดแวร์ตัวลูกลดลงเหลือเพียง 31 ตัวเท่านั้น

เนื่องจากเราต้องการที่จะส่งสัญญาณผ่านสายนำสัญญาณไปได้ระยะทางไกลจึงทำให้ฮาร์ดแวร์ตัวแม่จะต้องมีส่วนของวงจรไดรเวอร์เพื่อทำหน้าที่สร้างกระแสให้เพียงพอ ในส่วนของฮาร์ดแวร์ตัวลูกก็เช่นเดียวกันที่จะต้องมีวงจรไดรเวอร์เพื่อที่จะทำการส่งข้อมูลกลับไปที่สายนำสัญญาณซึ่งใช้ร่วมกันระหว่างฮาร์ดแวร์ตัวแม่และฮาร์ดแวร์ตัวลูก ซึ่งลักษณะของวงจรไดรเวอร์จะเป็นดังรูปข้างล่าง



รูปที่ 4.3 วงจรไดรเวอร์ในการส่งสัญญาณระหว่างฮาร์ดแวร์ตัวแม่และฮาร์ดแวร์ตัวลูก

4.3 การทำงานของฮาร์ดแวร์ตัวลูก

การทำงานของฮาร์ดแวร์ตัวลูกจะแบ่งออกเป็น 2 ส่วนใหญ่ๆคือ

1) การเปรียบเทียบหมายเลขประจำตัวของฮาร์ดแวร์ตัวลูก

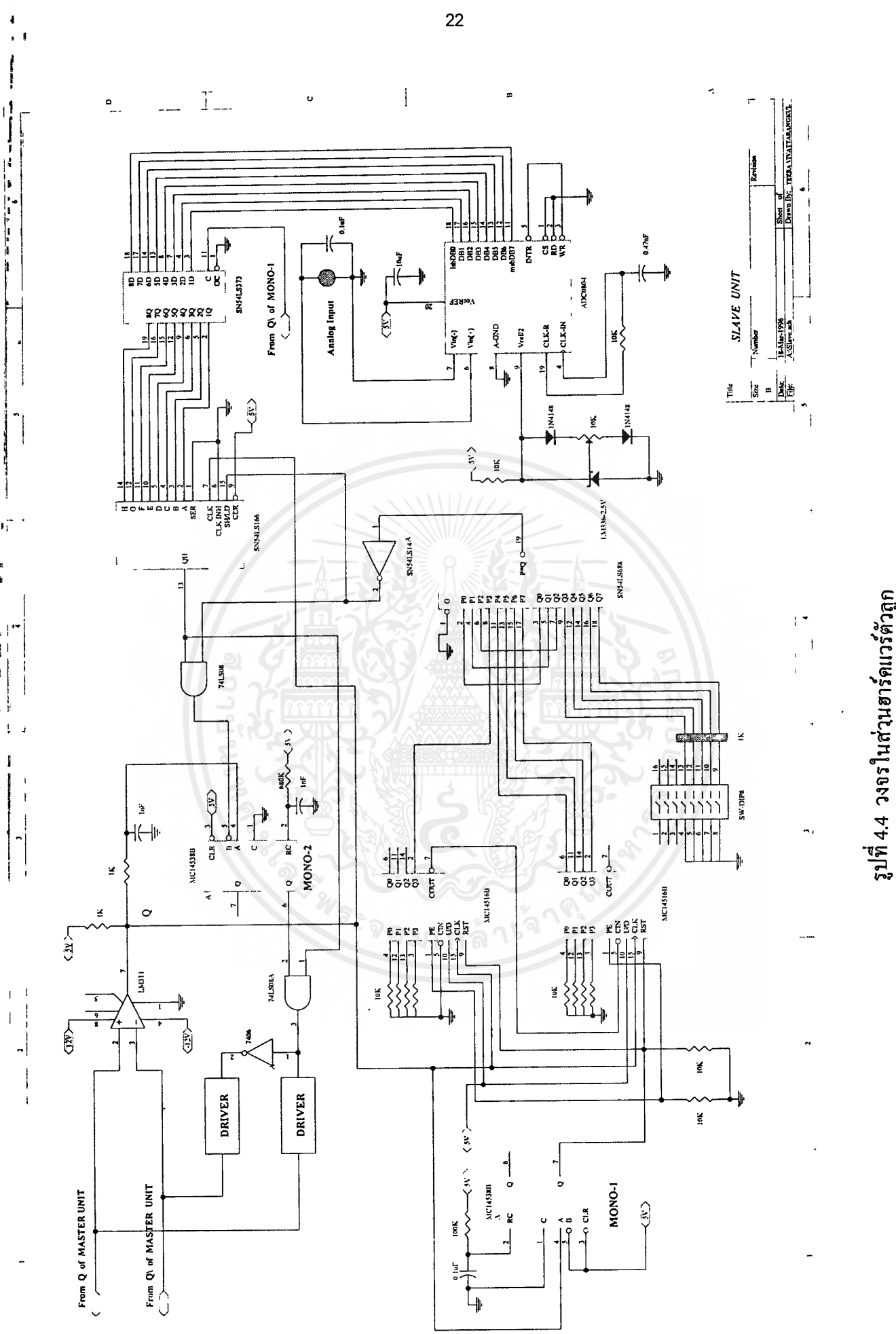
2) การส่งข้อมูลกลับไปให้ฮาร์ดแวร์ตัวแม่

ในส่วนแรกนั้นเมื่อฮาร์ดแวร์ตัวลูกได้รับสัญญาณนาฬิกาจากวงจรคอมพาราเตอร์แล้ว ก็จะนำสัญญาณนี้ไปผ่านวงจรเคาเตอร์เพื่อทำการนับสัญญาณพัลส์ 256 ลูก เอาพุทต์ของตัวเคาเตอร์นั้นจะเชื่อมต่อกับดิจิตอลคอมพาราเตอร์เพื่อทำการเปรียบเทียบกับ DIP-SW ซึ่งค่าของ DIP-SW จะใช้

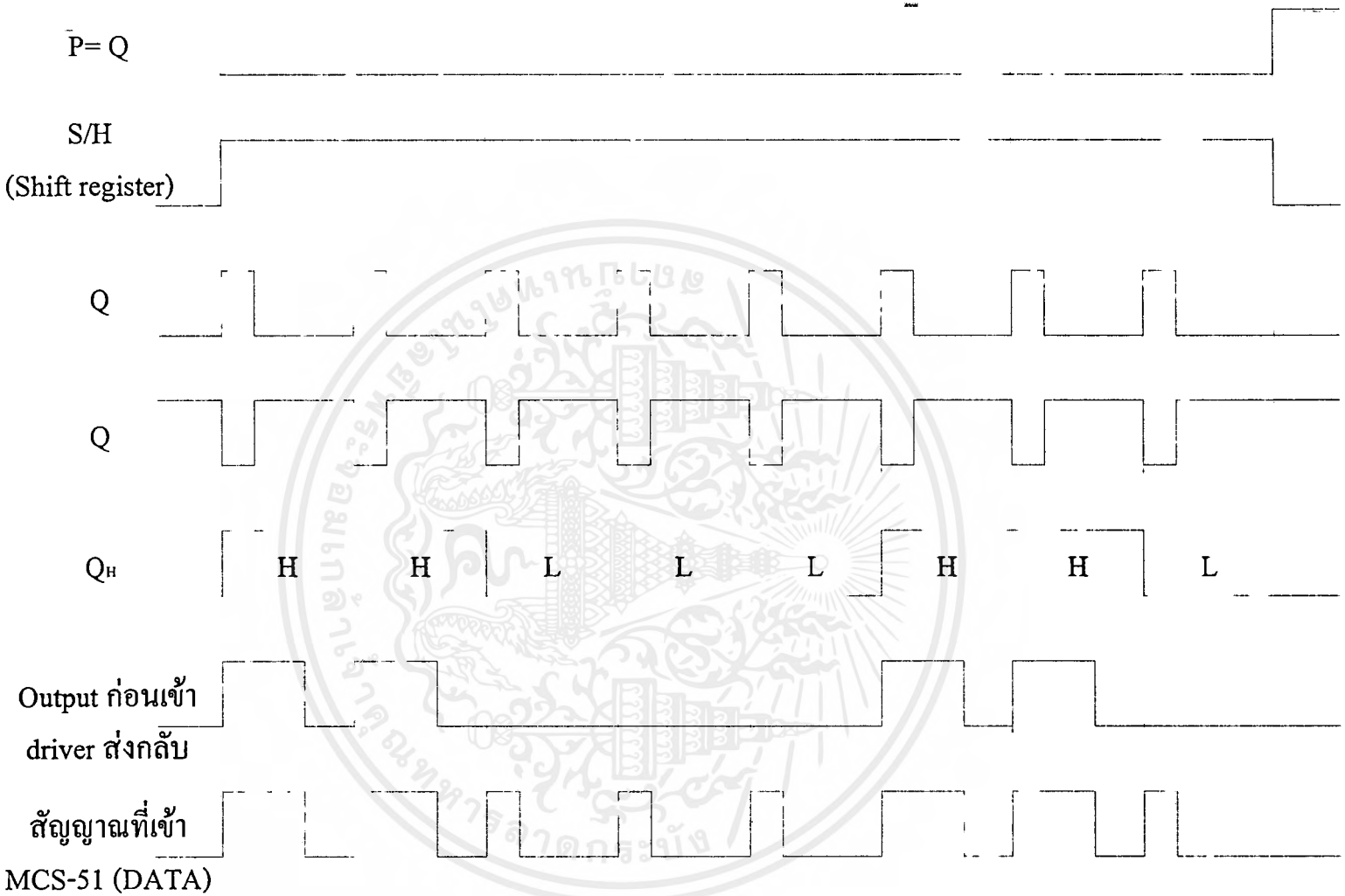
เป็นหมายเลขประจำตัวของฮาร์ดแวร์ตัวลูกแต่ละตัว จากที่กล่าวมาแล้วว่าฮาร์ดแวร์ตัวลูกมีทั้งหมด 31 ตัว เราจึงใส่ค่าให้กับ DIP-SW เพียงแค่ 5 บิต เมื่อดิจิตอลคอมพาราเตอร์ทำการเปรียบเทียบและพบว่าเอาพุทต์ของเคาเตอร์และ DIP-SW มีค่าตรงกันก็จะทำการรีเซตขา P=Q ทันที เราจะนำสัญญาณที่ได้จากขา P=Q นี้เป็นตัวชิฟข้อมูลของฮาร์ดแวร์ตัวลูกเพื่อทำการส่งกลับไปให้ฮาร์ดแวร์ตัวแม่ต่อไป

ในส่วนของโมโนสเตเบิล (MONO1) จะทำหน้าที่ในการรีเซตการทำงานของเคาเตอร์ ซึ่งจะเกิดขึ้นในช่วงของสัญญาณแบงค์ โมโนสเตเบิลที่ใช้นั้นเป็นแบบ Retrigable โดยเราได้ตั้งเวลาในการทริกไว้ที่ประมาณ 3 ms หรือประมาณ 3 คาบของสัญญาณนาฬิกา ในสภาวะปกติเมื่อไม่มีสัญญาณนาฬิกาเคาเตอร์จะอยู่ในสถานะถูกรีเซต แต่เมื่อใดก็ตามที่มีสัญญาณนาฬิกาแบบ positive edge เข้ามาทริกเคาเตอร์ก็จะทำการนับทันทีจนเมื่อถึงช่วงสัญญาณแบงค์เคาเตอร์ก็จะถูกรีเซตเป็นศูนย์อีกครั้งหนึ่ง

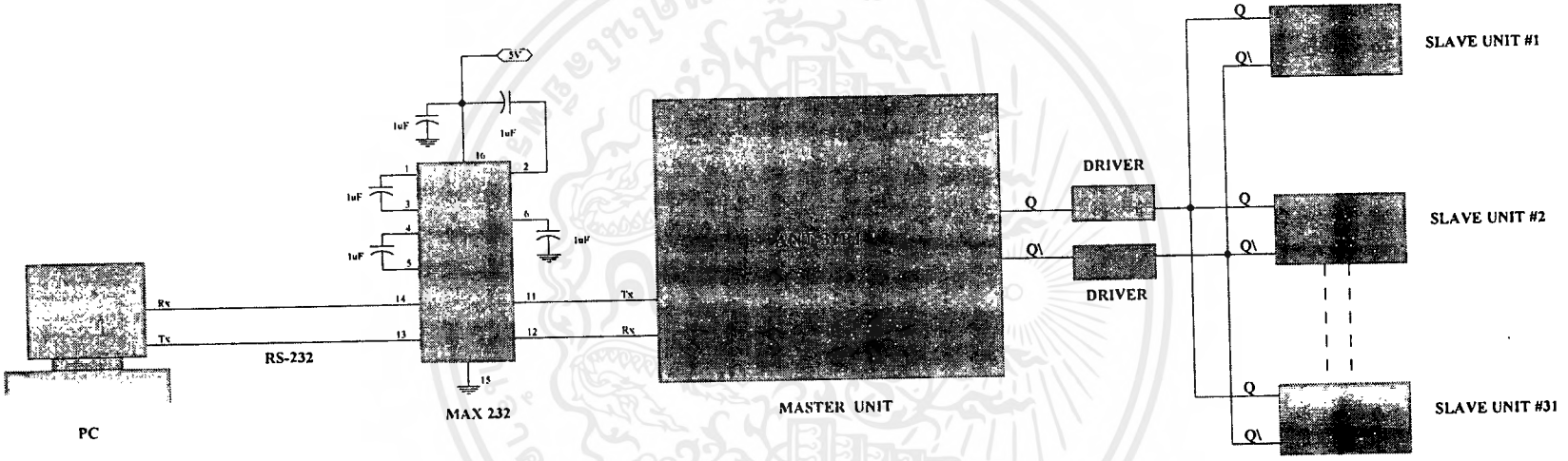
วงจรของ A/D จะทำหน้าที่แปลงสัญญาณจากอนาลอกอินพุทอันได้แก่ อุณหภูมิ ความชื้น ศักดาไฟฟ้า ไปเป็นสัญญาณแบบดิจิตอล ซึ่งในโครงงานนี้ได้จำลองสัญญาณอนาลอกโดยใช้ตัวต้านทานที่ปรับค่าได้ สัญญาณที่ได้จาก A/D จะถูกนำไปผ่าน Latch ซึ่งเมื่อมีสัญญาณ Q\ ของโมโนสเตเบิล (MONO1) ที่เป็นลอจิก 1 มา enable ก็จะทำให้ Latch ส่งผ่านข้อมูลไปให้ชิฟรีจิสเตอร์ต่อไป และเมื่อถึงช่วงจังหวะของฮาร์ดแวร์ตัวลูกตัวนั้นๆก็จะทำการส่งข้อมูลต่อไปให้โมโนสเตเบิล (MONO2) ในแบบอนุกรม ถ้าหากโมโนสเตเบิล (MONO2) ตรวจสอบพบว่าบิตที่รับมามีค่าลอจิกเป็น 1 ก็จะทำการยืดสัญญาณนาฬิกาออกจนมีดิวตี้ไซเคิลประมาณ 75% แต่ถ้าเป็นค่าลอจิก 0 ก็จะไม่มีการยืดสัญญาณนาฬิกาออก ซึ่งข้อมูลเหล่านี้จะถูกส่งเข้าสู่วงจรไดรเวอร์เพื่อนำข้อมูลส่งกลับไปยังสายส่งแบบดิฟเฟอเรนเชียลเส้นเดียวกับที่ฮาร์ดแวร์ตัวแม่ใช้ส่งสัญญาณนาฬิกาต่อไป ฮาร์ดแวร์ตัวลูกตัวใดที่ยังไม่ถึงช่วงเวลาของตนที่จะทำการส่งข้อมูล สัญญาณข้อมูลของฮาร์ดแวร์ตัวนั้นภายในสายตัวนำสัญญาณจะมีค่าเป็นศูนย์ ทำให้ไม่มีผลกระทบต่อข้อมูลของฮาร์ดแวร์ตัวลูกที่ทำการส่งข้อมูล



รูปที่ 4.4 วงจรในส่วนฮาร์ดแวร์ตัวถูก



รูปที่ 4.5 Timimg Diagram ของระบบ



Title: *MASTER UNIT*

Size: B | Number | Revision

Date: 18-Mar-1998 | Sheet of

File: A:\MASTER.SCH | Drawn by:

4.4 การรับข้อมูลของฮาร์ดแวร์ตัวแม่และการส่งข้อมูลไปที่เครื่องคอมพิวเตอร์

ฮาร์ดแวร์ตัวแม่นอกจากจะใช้ในการสร้างสัญญาณนาฬิกาแล้ว ยังใช้ในการรับข้อมูลจากฮาร์ดแวร์ตัวลูกอีกด้วย โดยจะนำสัญญาณนาฬิกา Q และ Q\ ผ่านวงจรคอมพาราเตอร์ซึ่งเป็นส่วนหนึ่งของฮาร์ดแวร์ตัวแม่ สัญญาณเอาพุทต์ที่ได้จะมีลักษณะเหมือน Q ทุกประการ แล้วจึงทำการตรวจสอบระดับของสัญญาณนาฬิกาที่ดิวตี้ไซเคิล 50% ว่ามีค่าลอจิกอะไร ซึ่งนั่นก็คือค่าของข้อมูลที่ส่งมาจากฮาร์ดแวร์ตัวลูก ข้อมูลขนาด 8 บิตที่ได้จากฮาร์ดแวร์ตัวลูกแต่ละตัวจะถูกนำไปเก็บไว้ในหน่วยความจำของฮาร์แวร์ตัวแม่ เมื่อถึงเวลาในช่วงแบงค์ของสัญญาณก็จะนำข้อมูลทั้งหมดนี้มาหาพาริตี้บิต เพื่อให้เครื่องคอมพิวเตอร์ได้ตรวจสอบว่าข้อมูลที่ส่งขึ้นไปมีความถูกต้องหรือไม่ จากนั้นจึงทำการส่งข้อมูลไปยังเครื่องคอมพิวเตอร์ ผ่านสายส่งสัญญาณแบบอนุกรม RS-232 ถ้าข้อมูลที่เครื่องคอมพิวเตอร์อ่านได้ไม่ถูกต้องก็จะมีการส่งสัญญาณอินเตอรัพท์จากเครื่องคอมพิวเตอร์ลงมาเพื่อร้องขอให้ฮาร์ดแวร์ตัวแม่ส่งข้อมูลขึ้นไปใหม่อีกครั้งหนึ่ง โดยการติดต่อกับเครื่องคอมพิวเตอร์นี้จะกระทำอยู่ภายในช่วงสัญญาณแบงค์นี้เท่านั้น

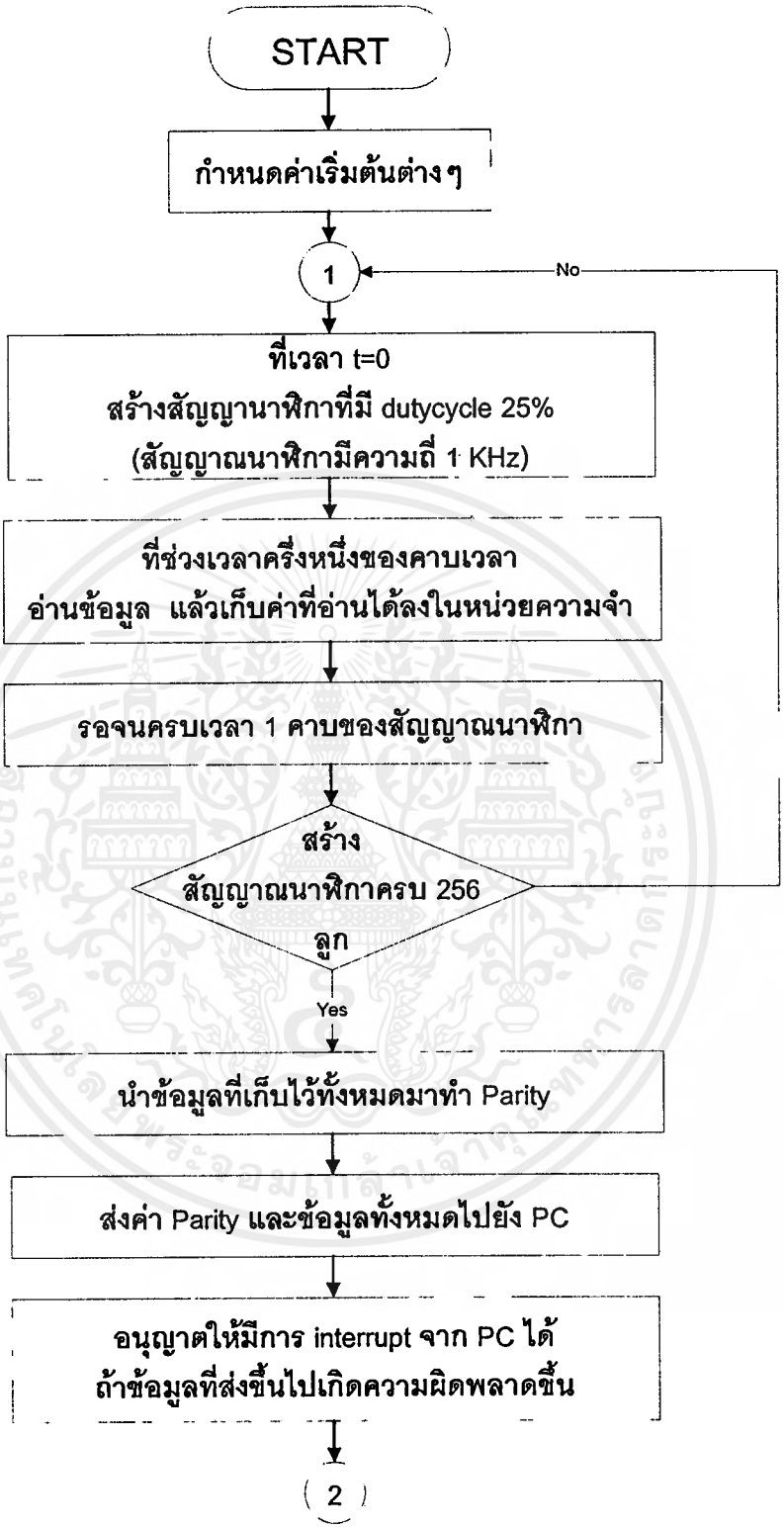
4.5 การแสดงผลที่หน้าจอเครื่องคอมพิวเตอร์

ข้อมูลที่เครื่องคอมพิวเตอร์อ่านได้จากฮาร์ดแวร์ตัวแม่นั้น จะถูกนำไปแสดงผลในรูปของกราฟและตาราง ในส่วนของกราฟนั้นสามารถที่จะแสดงกราฟของตัวลูกได้พร้อมกันถึง 5 ตัวบนกราฟเดียวกัน รวมทั้งยังสามารถที่จะย่อขยายกราฟและปรับเปลี่ยนค่าศักดาสูงสุดของสัญญาณอนาลอกได้อีกด้วย ในส่วนของตารางสามารถตรวจสอบข้อมูลของฮาร์ดแวร์ตัวลูกได้ทั้ง 31 ตัว ซึ่งการแสดงผลทั้งสองรูปแบบจะมีเวลากำกับอยู่เสมอทำให้เราสามารถที่จะทำการวิเคราะห์และประเมินผลที่เวลาต่างๆได้อย่างง่ายดาย รวมทั้งยังสามารถที่จะเก็บข้อมูลในรูปของไฟล์ได้อีกด้วย

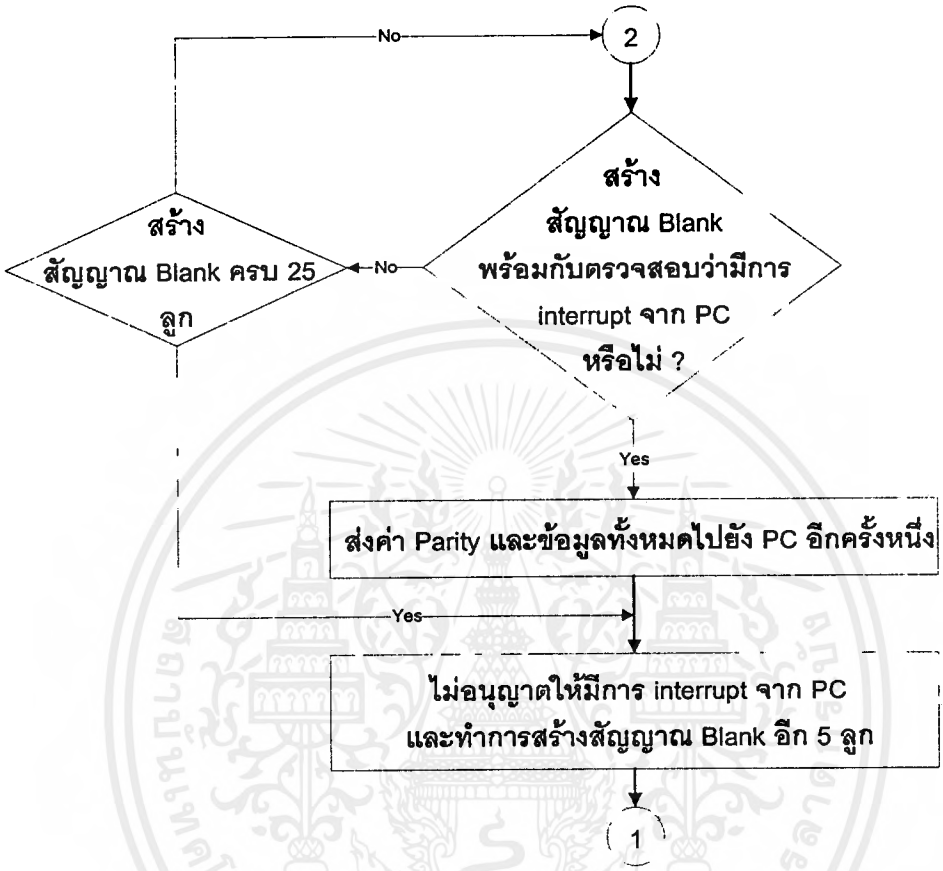
4.6 การทำงานของโปรแกรมในส่วนต่างๆ

โปรแกรมที่ใช้ในการควบคุมการทำงานในระบบการส่งข้อมูลดิจิตอลแบบเป็นลำดับนี้ จะแบ่งออกเป็น 2 ส่วน คือ ส่วนของโปรแกรมที่ใช้ในฮาร์ดแวร์ตัวแม่ และส่วนของโปรแกรมที่ใช้ในการแสดงผลบนเครื่องคอมพิวเตอร์

ส่วนของโปรแกรมที่ใช้ในฮาร์ดแวร์ตัวแม่ เป็นโปรแกรมที่ใช้เพื่อควบคุมการทำงานของไมโครคอนโทรลเลอร์ โดยในการเขียนโปรแกรมจะใช้ C for 51 โปรแกรมในส่วนนี้จะมีโฟล์วชาร์ตแสดงการทำงานดังนี้



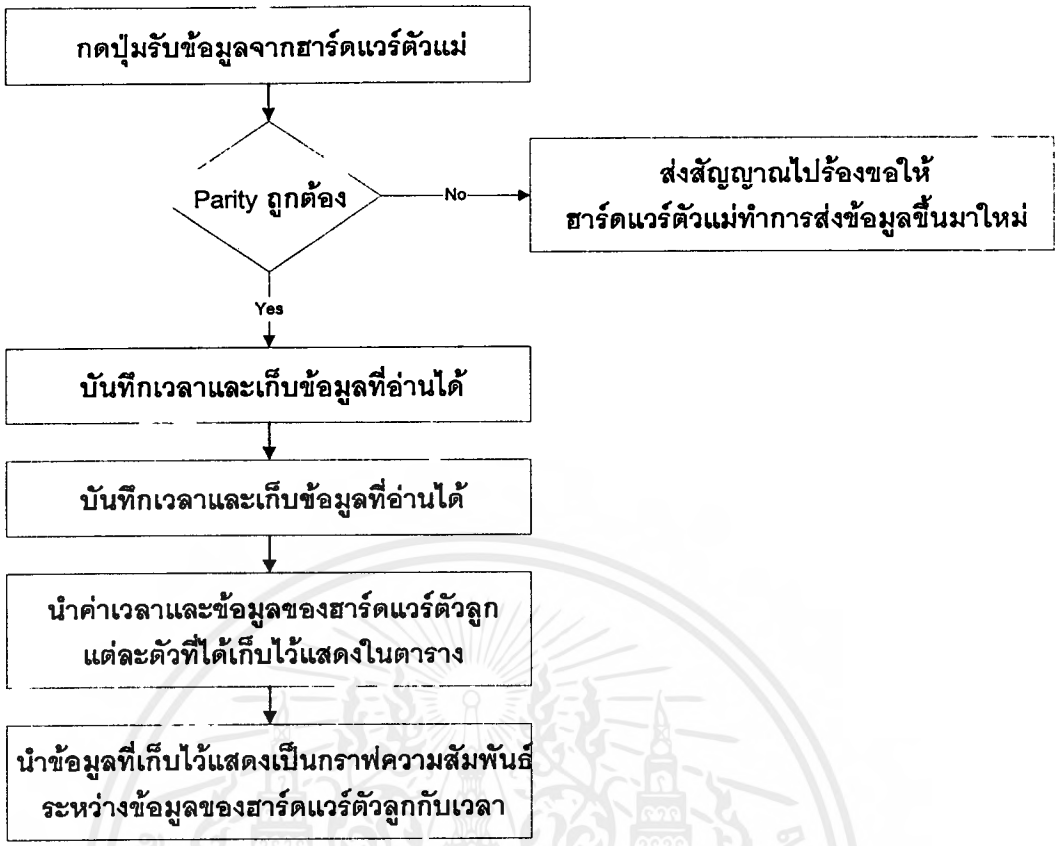
รูปที่ 3.7 โฟล์วชาร์ตแสดงการทำงานของโปรแกรมที่ใช้งานในฮาร์ดแวร์ตัวแม่



รูปที่ 3.7 (ต่อ) โฟล์วชาร์ตแสดงการทำงานของโปรแกรมที่ใช้งานในฮาร์ดแวร์ตัวแม่

ส่วนของโปรแกรมที่ใช้ในการแสดงผลบนเครื่องคอมพิวเตอร์ ได้ใช้การเขียนโปรแกรมด้วย Delphi โดยโปรแกรมในส่วนแสดงผลนี้จะมีหน้าที่การทำงานหลักๆ ดังนี้

- การรับข้อมูลจากฮาร์แวร์ตัวแม่ ผ่านทางพอร์ตอนุกรมภายใต้มาตรฐาน RS-232 ซึ่งมีโฟล์วชาร์ตการทำงานดังนี้



รูปที่3.8 โฟล์วชาร์ตแสดงการทำงานของโปรแกรมในการรับข้อมูลจากฮาร์ดแวร์ตัวแม่

- การใช้กราฟในหารแสดงผล ในโปรแกรมนี้ กราฟ 1 รูป สามารถแสดงค่าข้อมูลของฮาร์ดแวร์ตัวลูกได้พร้อมกันจำนวนสูงสุด 5 ตัว ซึ่งผู้ใช้สามารถเลือกที่จะแสดงหรือไม่แสดงกราฟของฮาร์ดแวร์ตัวใดก็ได้ โดยกราฟที่แสดงจะเป็นกราฟความสัมพันธ์ระหว่างข้อมูลของฮาร์ดแวร์ตัวลูกกับเวลาที่เครื่องคอมพิวเตอร์รับข้อมูลนั้นเข้ามาได้ ลักษณะของกราฟจะเป็นกราฟเคลื่อนไหวตามเวลาและค่าของข้อมูลที่เปลี่ยนไป นอกจากนี้ผู้ใช้ยังสามารถกำหนด cursor ให้คงที่อยู่ที่ค่าใดค่าหนึ่งได้เพื่อความสะดวกในการวิเคราะห์การฟของข้อมูล โดยจะสามารถกำหนด cursor ได้สูงสุด 4 ค่า และในรูปกราฟที่มีค่าต่ำมากๆ ผู้ใช้สามารถที่จะทำการ Zoom In หรือ Zoom Out เพื่อที่จะดูค่าของข้อมูลได้อย่างสะดวกยิ่งขึ้น

- การบันทึกเป็นไฟล์ข้อมูลและการเปิดไฟล์ข้อมูล ข้อมูลของฮาร์ดแวร์ตัวลูกแต่ละตัวที่เวลาต่างๆ สามารถเก็บบันทึกเป็นไฟล์ข้อลูกแบบเท็กซ์ไฟล์ (Text File) ได้ นอกจากนี้ยังสามารถนำไฟล์ข้อมูลที่ได้เก็บไว้มาเปิดแสดงในภายหลัง ซึ่งวิธีการในการใช้งานการแสดงกราฟนี้จะ

ทำได้เช้นเดียวกับการแสดงกราฟตามปกติ พร้อมทั้งยังได้แสดงค่าข้อมูลที่เก็บไว้ในรูปแบบของตาราง เพื่อความสะดวกในการนำข้อมูลมาวิเคราะห์และประมวลผลต่อไป

- การตั้งค่าแรงดันสูงสุด เนื่องจากในโครงงานนี้ได้ทำการจำลองข้อมูลของฮาร์ดแวร์ตัวลูกด้วยสัญญาณอนาลอกอินพุทในรูปของค่าแรงดัน ซึ่งในโปรแกรมนี้ได้กำหนดค่าเริ่มต้นของค่าแรงดันของข้อมูลของฮาร์ดแสร์ตัวลูกไว้ที่ 5 Volt ดังนั้นก่อนที่จะทำการรับข้อมูลจึงจำเป็นจะต้องกำหนดค่าแรงดันสูงสุดของฮาร์ดแวร์ตัวลูกแต่ละตัวให้ถูกต้องเสียก่อนเพื่อที่จะให้สามารถแสดงค่าแรงดันได้ถูกต้องตามที่เป็นจริง

- การจัดการหน้าต่างบนหน้าจอคอมพิวเตอร์ โปรแกรมนี้จะมีส่วนที่ช่วยในการจัดการหน้าต่างให้เป็นไปตามรูปแบบต่างๆ คือ การ Cascade , Tile หน้าต่าง ,การย้ายไปทำงานในหน้าต่างถัดไป (Next) , การปิดหน้าต่างย่อยทั้งหมด (Close All) ทั้งนี้เพื่อที่จะทำให้การใช้งานต่างๆ ในโปรแกรมสะดวกยิ่งขึ้น

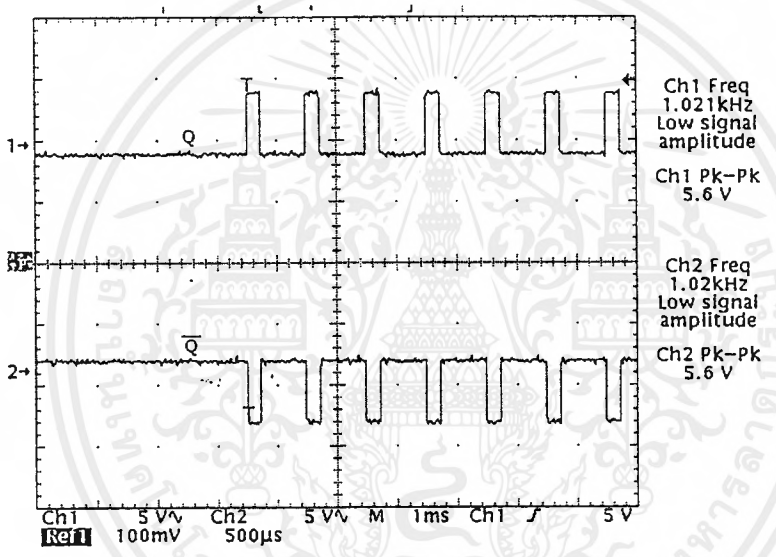
บทที่ 5

ผลการทดลอง

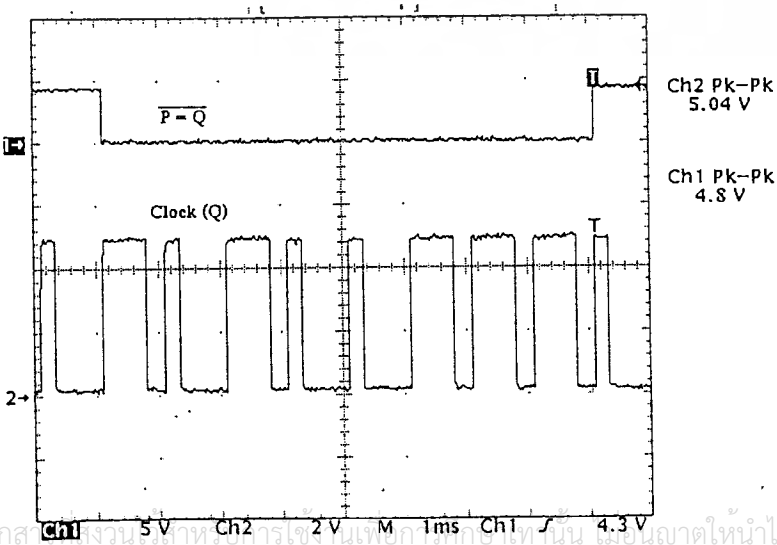
5.1 การทดลองเก็บภาพของสัญญาณต่างๆ ระหว่างฮาร์ดแวร์ตัวแม่และฮาร์แวร์ตัวลูก

ได้ทำการทดลองเก็บภาพของสัญญาณที่จุดต่างๆ โดยให้ฮาร์แวร์ตัวลูกทำการส่งข้อมูล 11100101 ไปยังฮาร์ดแวร์ตัวแม่ ได้ภาพของสัญญาณต่างๆ ดังนี้

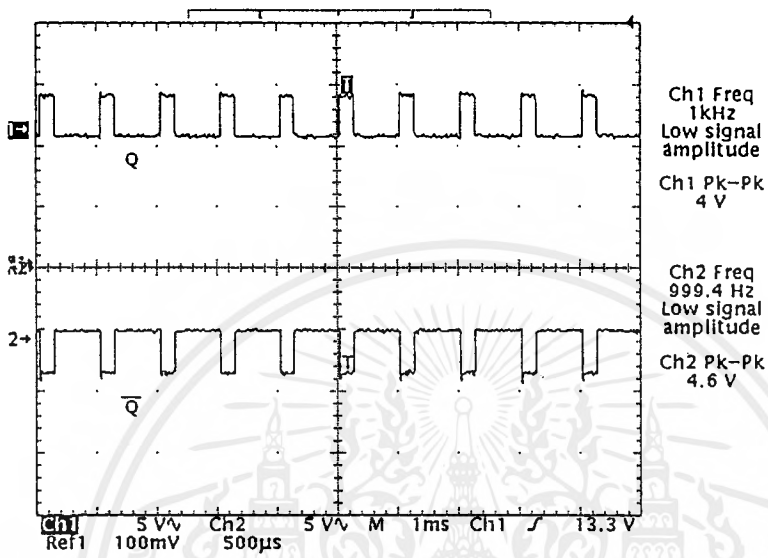
1. ภาพสัญญาณนาฬิกาที่ฮาร์ดแวร์ตัวแม่ส่งออกมา



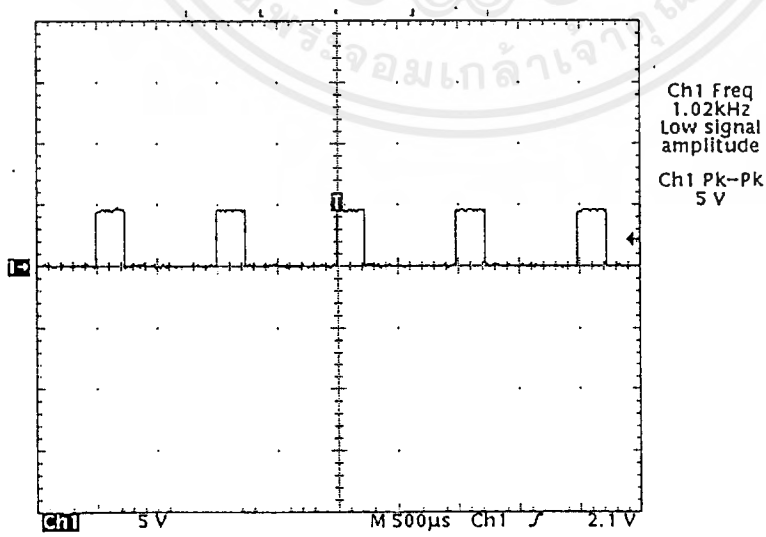
2. ภาพสัญญาณข้อมูล (Clock Q) กับสัญญาณ $P=\overline{Q}$



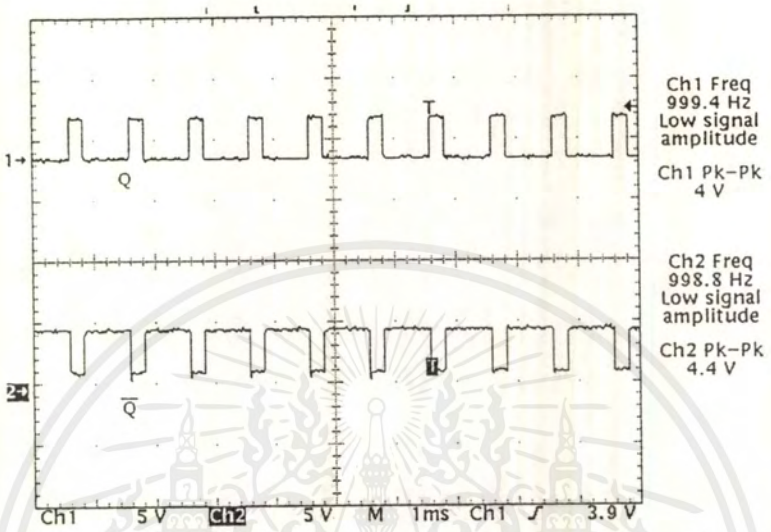
3. ภาพสัญญาณก่อนผ่านเข้าคอมพาราเตอร์ เมื่อใช้สายนำสัญญาณสั้น



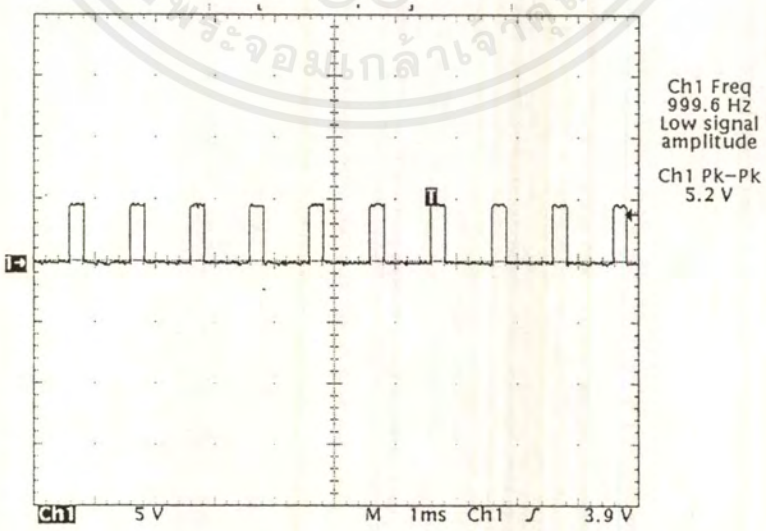
4. ภาพสัญญาณหลังออกจากคอมพาราเตอร์ เมื่อใช้สายนำสัญญาณสั้น



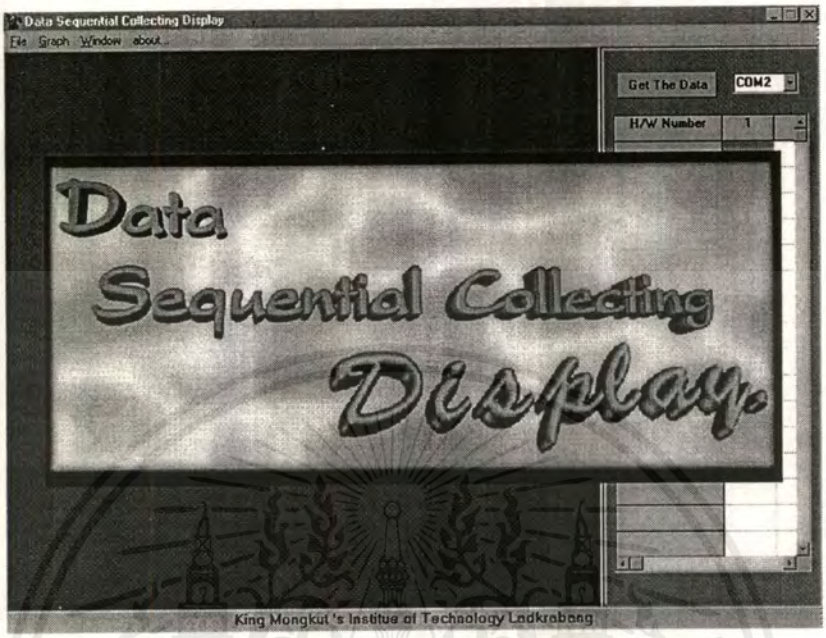
5. ภาพสัญญาณก่อนผ่านคอมพาราเตอร์ เมื่อใช้สายนำสัญญาณยาว (ประมาณ 1 กิโลเมตร)



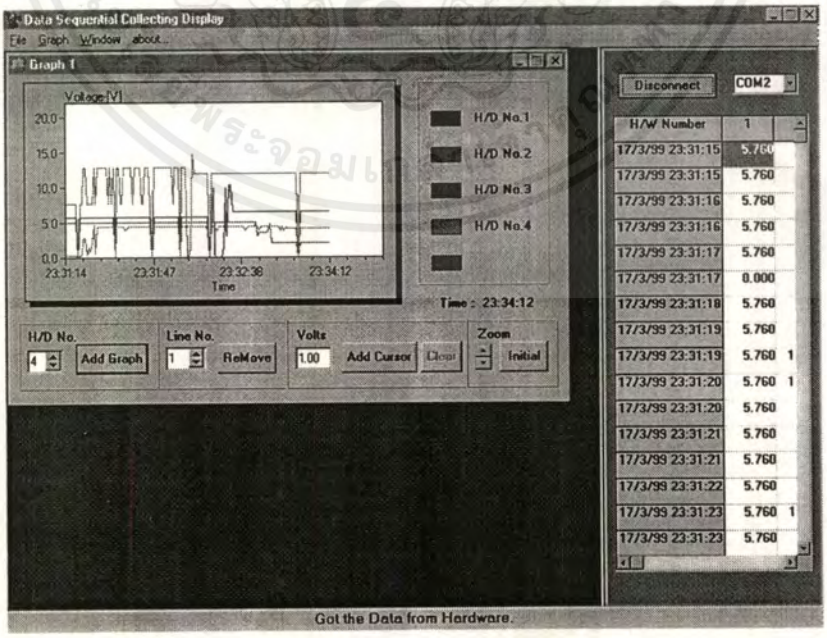
6. ภาพสัญญาณหลังออกจากคอมพาราเตอร์ เมื่อใช้สายนำสัญญาณยาว (ประมาณ 1 กิโลเมตร)



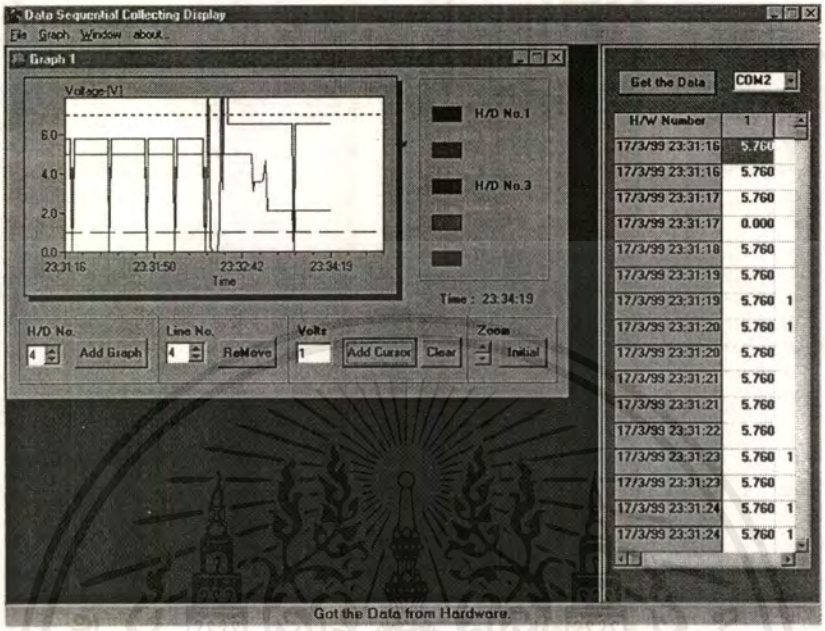
5.2 ตัวอย่างโปรแกรมแสดงผลบนเครื่องคอมพิวเตอร์



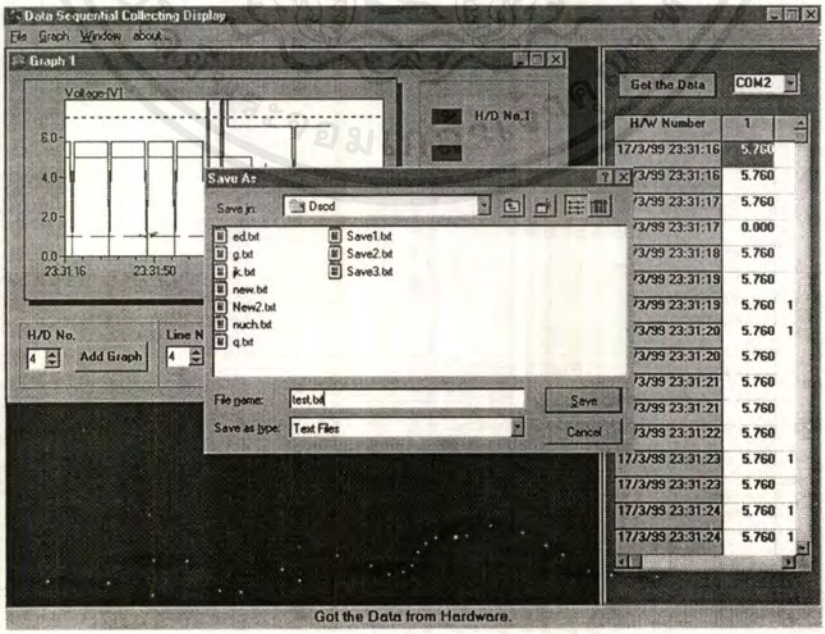
รูปหน้าต่างแรกของโปรแกรมแสดงผลบนเครื่องคอมพิวเตอร์



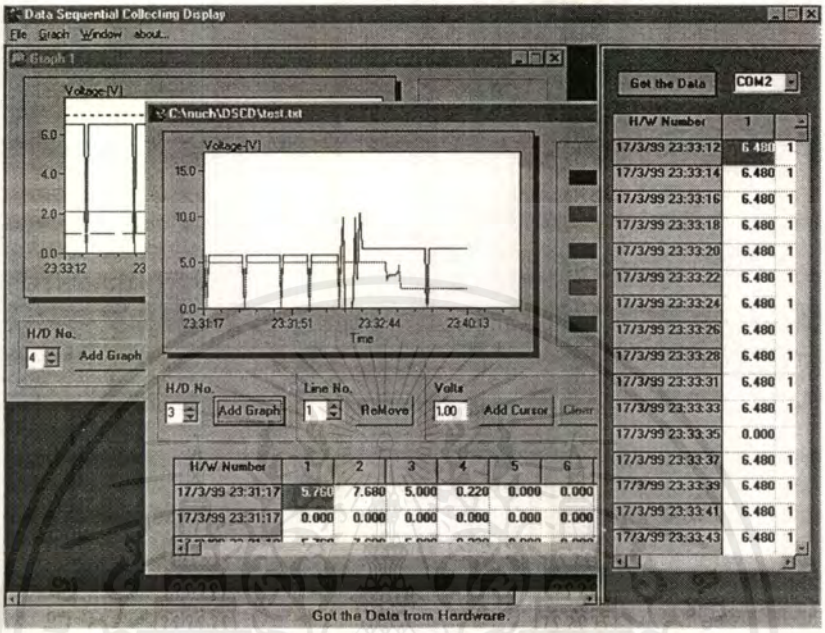
รูปของหน้าจอเครื่องคอมพิวเตอร์ เมื่อทำการรับข้อมูลจากฮาร์ดแวร์ตัวแม่



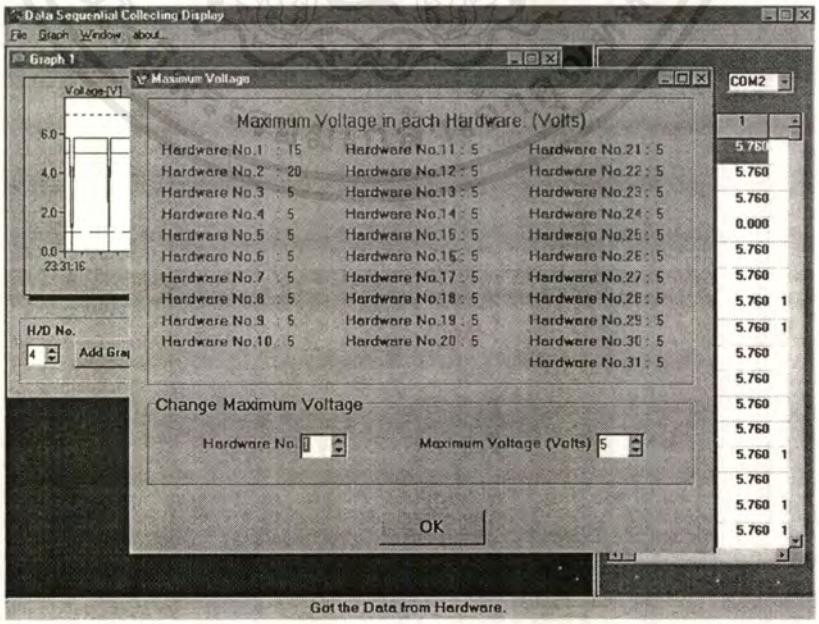
รูปของหน้าจอเครื่องคอมพิวเตอร์ เมื่อทำการ Remove กราฟ ,Add Cursor และ Zoom In



รูปของหน้าจอเครื่องคอมพิวเตอร์ เมื่อทำการบันทึกข้อมูลเก็บเป็นไฟล์ข้อมูล



รูป ของหน้าจอเครื่องคอมพิวเตอร์ เมื่อทำการเปิดไฟล์ข้อมูลที่ได้เก็บไว้แล้ว



รูปของหน้าจอเครื่องคอมพิวเตอร์ เมื่อทำการกำหนดค่าแรงดันสูงสุดให้แก่ฮาร์ดแวร์ตัวลูกแต่ละตัว

บทที่ 6
สรุปผลการทดลอง

ระบบการจัดเก็บข้อมูลแบบเป็นลำดับนี้สำเร็จตามวัตถุประสงค์ที่ได้ตั้งไว้ โดยสามารถที่จะนำข้อมูลจากอนาลอกอินพุทของ A/D ที่แปลงเป็นดิจิตอลแล้วมาเก็บในหน่วยความจำของฮาร์ดแวร์ตัวแม่ แล้วนำขึ้นไปแสดงผลบนเครื่องคอมพิวเตอร์ได้ ปัญหาที่เกิดขึ้นในระหว่างระยะเวลาดำเนินงานมีดังนี้

- สัญญาณนาฬิกาที่เข้าไปยังอินพุทของเคาเตอร์เร็วกว่าการเปลี่ยนระดับสัญญาณจากหนึ่งเป็นศูนย์ หรือจากรีเซตไปเป็นแอคทีฟของโมโนสเตเบิล (MONO1) ทำให้ต้องสูญเสียสัญญาณนาฬิกาไป 1 ลูกในการกระทำดังกล่าว ส่งผลให้ต้องตัด สัญญาณนาฬิกา 8 ลูกแรกทิ้งไป จึงเหลือฮาร์ดแวร์ตัวลูกเพียง 31 ตัว
- สัญญาณนาฬิกาที่เข้าขา A ของโมโนสเตเบิล (MONO2) มีความเร็วในการทำงานมากกว่าขา B ซึ่งเป็นข้อมูลที่ถูกส่งมาจากชิฟรีจิสเตอร์ ทำให้การทริกข้อมูลเกิดการผิดพลาดได้จึงต้องมีการต่อ RC-Delay เข้าไปที่ขา A เพื่อทำให้สัญญาณนาฬิกามาถึงที่ขา A ช้าลงและข้อมูลที่จะถูกยืดพัลส์ออกไปก็จะถูกต้อง
- ข้อมูลดิจิตอลจาก A/D บางบิตมีการเปลี่ยนระดับสัญญาณระหว่างลอจิก 1 และ 0 อยู่ตลอดเวลา ทำให้ไม่สามารถทราบค่าลอจิกที่แน่นอนของบิทนั้นๆได้ ซึ่งเป็นผลมาจาก Ripple และ Noise ที่ขา Vref/2 ของ A/D เอง ภายหลังจากการใช้ Voltage Reference จ่ายไฟเข้าที่ขา Vref/2 แทนการใช้แหล่งจ่ายไฟธรรมดาทั่วไป พบว่าไม่มีการเปลี่ยนระดับของสัญญาณไปมาอีก
- ในการใช้งานหน่วยความจำภายนอกตัวไมโครคอนโทรลเลอร์ในการเก็บข้อมูลของฮาร์ดแวร์ตัวลูก จะทำให้การส่งข้อมูลไปที่เครื่องคอมพิวเตอร์ช้ากว่าการใช้หน่วยความจำภายในตัวไมโครคอนโทรลเลอร์เอง ทั้งนี้เป็นผลเนื่องมาจากการเขียนโปรแกรมและคอมไพเลอร์ที่ใช้งานสำหรับซอฟต์แวร์ในส่วนของไมโครคอนโทรลเลอร์
- โปรแกรมที่ใช้แสดงกราฟบนเครื่องคอมพิวเตอร์จะนำข้อมูลที่ส่งมากฮาร์แวร์ตัว แม่จากทางพอร์ตอนุกรมมาเมื่อเครื่องมีเวลาว่างจากงานหลัก ทำให้บางครั้งก็ดึงข้อมูลมาเร็วบ้างช้าบ้าง เวลาที่แสดงการรับข้อมูลที่หน้าจอคอมพิวเตอร์จึงอาจมีความผิดพลาดเกิดขึ้นได้

- โปรแกรมควบคุมการทำงานของฮาร์ดแวร์ตัวแม่นั้นใช้ภาษา C for 51 ซึ่งการดีบักโปรแกรมทำได้ลำบาก เพราะการวางตำแหน่งของ Statement ก่อนหลังจะมีผลต่อการทำงานของโปรแกรมและรีจิสเตอร์ภายในตัวไมโครคอนโทรลเลอร์เอง

ปัญหาต่างๆที่เกิดขึ้นนี้ ส่วนใหญ่ได้ทำการแก้ไขเรียบร้อยแล้ว แต่ยังมีบางปัญหาที่ไม่สามารถแก้ไขได้จริงๆ โดยโครงงานที่จัดทำขึ้นนี้สามารถนำไปประยุกต์ใช้งานได้อย่างกว้างขวาง แต่อาจจะต้องเพิ่มเติมอุปกรณ์ในบางจุดของฮาร์ดแวร์ทั้งตัวแม่และตัวลูก อาทิเช่น ทรานดิวเซอร์ที่ใช้ในการเปลี่ยนสัญญาณธรรมชาติ เช่น ความร้อน อุณหภูมิ ความชื้น ระดับน้ำไปเป็นสัญญาณอนาลอกให้กับตัว A/D รวมทั้งสายส่งที่สามารถส่งได้ในระยะทางไกลๆ เป็นต้น นอกจากนี้ฮาร์ดแวร์ตัวลูกนั้นก็มิได้จำกัดที่ 31 ตัว แต่สามารถที่จะเพิ่มขึ้นได้ ซึ่งจะขึ้นอยู่กับค่าสูงสุดที่เคาเตอร์สามารถนับได้ และจำนวนหน่วยความจำที่อยู่ภายในฮาร์ดแวร์ตัวแม่ เป็นต้น เพื่อที่จะรองรับจำนวนข้อมูลของฮาร์ดแวร์ตัวลูกที่จะเพิ่มขึ้นมา

ภาคผนวก

โปรแกรมการทำงาน

```
program DSCD;

uses
  Forms,
  DSCD_1 in 'DSCD_1.pas' {Form1},
  DSCD_2 in 'DSCD_2.pas' {Form2},
  CSCD_3 in 'DSCD_3.pas' {Form3},
  DSCD_4 in 'DSCD_4.pas' {Form4},
  DSCD_5 in 'DSCD_5.pas' {Form5},
  DSCD_6 in 'DSCD_6.pas' {Form6};

{$R *.RES}

begin
  Application.Initialize;
  Application.CreateForm(TForm1, Form1);
  Application.Run;
end.
```

```
    Next1: TMenuItem;
    Closeall1: TMenuItem;
    PortCombo: TComboBox;
    Comm32: TComm32;
    procedure FormCreate(Sender: TObject);
    procedure Graph1Click(Sender: TObject);
    procedure GetDataButtClick(Sender: TObject);
    procedure MaximumVoltage1Click(Sender: TObject);
    procedure NewGraph1Click(Sender: TObject);
    procedure StringGrid1DrawCell(Sender: TObject; Col, Row: Integer;
      Rect: TRect; State: TGridDrawState);
    procedure Save1Click(Sender: TObject);
    procedure Open1Click(Sender: TObject);
    procedure TimerTitleTimer(Sender: TObject);
    procedure About1Click(Sender: TObject);
    procedure Exit1Click(Sender: TObject);
    procedure Cascade1Click(Sender: TObject);
    procedure Tile1Click(Sender: TObject);
    procedure Next1Click(Sender: TObject);
    procedure Closeall1Click(Sender: TObject);
    procedure Comm32Open(Sender: TObject; ProviderSubType, Error: Integer);
    procedure Comm32Error(Sender: TObject; Errors: Integer);
    procedure Comm32RxFlag(Sender: TObject);
    procedure Comm32Break(Sender: TObject);
    procedure Comm32RxChar(Sender: TObject; Count: Integer);
    procedure Comm32TxEmpty(Sender: TObject);
    procedure FormClose(Sender: TObject; var Action: TCloseAction);
  private
    { Private declarations }
    ConvertData : array[0..31] of real;
    FileStr : TStringList;
    SaveCount : integer;
    Title : TForm5;
    InputData : array[0..1024] of integer;
    CVData : array[0..32] of integer;
```

```
    InputIndex : integer;
    TimeTemp : TDateTime;
    ReTrans : Boolean;
    procedure BuffWrite(InputData : array of real;TimeTemp : TDateTime);
    procedure ConvertToInt(InputData : array of integer);
    procedure CheckData;
    procedure Retransmit;
  public
    { Public declarations }
    BuffData : array[1..150,0..31] of real;
    BuffDTime : array[1..150] of TDateTime;
    MaxVolt : array[1..31] of integer;
    index : integer;
    Graph : array[1..255] of TForm3;
    GraphEn : array[1..255] of boolean;
    MDICount : integer;
    LsPoint : array[1..31] of real;
  end;

var
  Form1: TForm1;

implementation

{$R *.DFM}

procedure TForm1.ConvertToInt(InputData : array of integer);
var
  VLSBin,VMSBin,i : integer;
  VLS,VMS : array[0..15] of real;
begin
  VLS[0] := 0;      VLS[1] := 0.020;  VLS[2] := 0.040;  VLS[3] := 0.060;
  VLS[4] := 0.080;  VLS[5] := 0.100;  VLS[6] := 0.120;  VLS[7] := 0.140;
  VLS[8] := 0.160;  VLS[9] := 0.180;  VLS[10] := 0.200; VLS[11] := 0.220;
  VLS[12] := 0.240; VLS[13] := 0.260; VLS[14] := 0.280; VLS[15] := 0.250;
```

```
  VMS[0] := 0;      VMS[1] := 0.320;  VMS[2] := 0.640;  VMS[3] := 0.960;
  VMS[4] := 1.280;  VMS[5] := 1.600;  VMS[6] := 1.920;  VMS[7] := 2.240;
  VMS[8] := 2.560;  VMS[9] := 2.880;  VMS[10] := 3.200; VMS[11] := 3.520;
  VMS[12] := 3.840; VMS[13] := 4.160; VMS[14] := 4.480; VMS[15] := 4.750;
  for i:=0 to 31 do
  begin
    VLSBin := InputData[i] and 15;
    VMSBin := InputData[i] and 240;
    VMSBin := VMSBin Div 16;
    ConvertData[i] := VMS[VMSBin] + VLS[VLSBin];
  end;
end;

procedure TForm1.BuffWrite(InputData : array of real;TimeTemp : TDateTime);
var
  i,j,Volt : integer;
  s : string[6];
begin
  {--- Remove first point ---}
  if index = 150 then
  begin
    for i:=1 to 31 do
      LsPoint[i] := BuffData[1,i];
    for i:=1 to 150-1  do
    begin
      for j:=0 to 31 do
        BuffData[i,j] := BuffData[i+1,j];
      BuffDTime[i] := BuffDTime[i+1];
    end;
    index := index-1;
    for i:=1 to index do
    begin
      for j:=1 to 31 do
      begin
        Str(BuffData[i,j]:0:3,s);
```

```
          StringGrid1.Cells[j,i] := s;
        end;
        StringGrid1.Cells[0,i] := DateTimeToStr(BuffDTime[i]);
      end;
    end;

    {--- Write data to Buffer and StringGrid ---}
    inc(index);
    BuffData[index,0] := InputData[0];
    for i:=1 to 31 do
    begin
      Volt := MaxVolt[i] div 5;
      BuffData[index,i] := InputData[i]*Volt;
    end;
    for i:=1 to 31 do
    begin
      Str(BuffData[index,i]:0:3,s);
      StringGrid1.Cells[i,index] := s;
    end;
    BuffDTime[index] := TimeTemp;
    StringGrid1.Cells[0,index] := DateTimeToStr(BuffDTime[index]);

    Application.ProcessMessages;
    {--- Draw Graph ---}
    for i:=1 to MDICount do
     if GraphEn[i] then
       Graph[i].WriteGraph;
  end;

  procedure TForm1.FormCreate(Sender: TObject);
  var
    i : integer;
  begin
    MDICount := 0;
    GetDataButt.Tag := 0;
```

```
  StringGrid1.RowCount := 150+1;
  StringGrid1.ColWidths[0] := 105;
  StringGrid1.Cells[0,0] := 'H/W Number';
  for i:=1 to 31 do
  begin
    StringGrid1.Cells[i,0] := IntToStr(i);
    MaxVolt[i] := 5;
  end;
  FileStr := TStringList.Create;
  SaveCount := 0;
  Save1.Enabled := false;
  Title := TForm5.Create(Application);
  OpenDialog1.Filename := 'Title.wav';
  MediaPlayer1.FileName := OpenDialog1.Filename;
  OpenDialog1.Filename := '*.txt';
  MediaPlayer1.Open;
  MediaPlayer1.Play;
  Comm32.DeviceName := PortCombo.Text;
end;

procedure TForm1.Graph1Click(Sender: TObject);
begin
 if index = 0 then {No data in Buffer.}
   NewGraph1.Enabled := false
 else
   NewGraph1.Enabled := true;
end;

procedure TForm1.GetDataButtClick(Sender: TObject);
var
 i,rw,cl : integer;
begin
 Save1.Enabled := true;
 if GetDataButt.Tag = 0 then  {Get the data}
 begin
```

```
    index := 0;
    for i:=1 to 31 do
      LsPoint[i] := 0;
    for rw:=1 to 150 do
      for cl:=0 to 31 do
      begin
        BuffData[rw,cl] := 0;
        StringGrid1.Cells[cl,rw] := '';
      end;
    InputIndex := 0;
    Comm32.DeviceName := PortCombo.Text;
    Comm32.Open;
  end
  else   {Disconnect}
  begin
    GetDataButt.Tag := 0;
    GetDataButt.Caption := 'Get the Data';
    Comm32.Close;
    StatusText.Caption := 'Disconnect with Hardware.';
  end;
end;

procedure TForm1.MaximumVoltage1Click(Sender: TObject);
var
  ChangeMax : TForm2;
begin
  Application.ProcessMessages;
  ChangeMax := TForm2.Create(Application);
  ChangeMax.Caption := 'Maximum Voltage'
end;

procedure TForm1.NewGraph1Click(Sender: TObject);
begin
  Application.ProcessMessages;
  if MDIChildCount = 0 then
```

```
    MDICount := 0;
  inc(MDICount);
  Graph[MDICount] := TForm3.Create(Application);
  Graph[MDICount].Caption := 'Graph ' + IntToStr(MDICount);
  Graph[MDICount].Tag := MDICount;
  GraphEn[MDICount] := true;
end;

procedure TForm1.StringGrid1DrawCell(Sender: TObject; Col, Row: Integer;
  Rect: TRect; State: TGridDrawState);
var
  bufB : array[0..79] of char;
  algn : word;
begin
  if (Row=0) or (Col=0) then
    algn := dt_Center
  else   algn := dt_Right;
  strPCopy(bufB,StringGrid1.Cells[Col,Row]);
  StringGrid1.Canvas.FillRect(Rect);
  DrawText(StringGrid1.Canvas.Handle,bufB,-1,Rect,algn);
end;

procedure TForm1.Save1Click(Sender: TObject);
var
  Row,Col,cl,rw : integer;
begin
  inc(SaveCount);
  SaveDialog1.FileName := 'Untitled-'+IntToStr(SaveCount);
  Comm32.Close;
  if not SaveDialog1.Execute then  Exit;
  Comm32.Open;
  FileStr.Clear;
  Row := StringGrid1.RowCount;
  Col := StringGrid1.ColCount;
  Application.ProcessMessages;
```

```
  for rw := 0 to Row-1 do
    for cl := 0 to Col-1 do
      FileStr.Add(StringGrid1.Cells[cl,rw]);
  Application.ProcessMessages;
  for cl:= 1 to 31 do
    FileStr.Add(IntToStr(MaxVolt[cl]));
  Application.ProcessMessages;
  FileStr.SaveToFile(SaveDialog1.FileName);
end;

procedure TForm1.Open1Click(Sender: TObject);
var
  LoadForm : TForm4;
  cl,rw,itm : integer;
begin
  Comm32.Close;
  if not OpenDialog1.Execute then Exit;
  Comm32.Open;
  FileStr.Clear;
  Application.ProcessMessages;
  FileStr.LoadFromFile(OpenDialog1.FileName);
  Application.ProcessMessages;
  LoadForm := TForm4.Create(Application);
  LoadForm.Caption := OpenDialog1.FileName;
  itm := 0;
  Application.ProcessMessages;
  with LoadForm do
  begin
    for rw:= 0 to StringGrid1.RowCount-1 do
      for cl:= 0 to StringGrid1.ColCount-1 do
      begin
        StringGrid1.Cells[cl,rw] := FileStr[itm];
        inc(itm);
      end;
    for cl:= 1 to 31 do
```

```
    begin
      MVolt[cl] := StrToInt(FileStr[itm]);
      inc(itm);
    end;
    Convert;
  end;
end;

procedure TForm1.TimerTitleTimer(Sender: TObject);
begin
  TimerTitle.Enabled := false;
  Title.Close;
end;

procedure TForm1.About1Click(Sender: TObject);
var
  About : TForm6;
begin
  Application.ProcessMessages;
  About := TForm6.Create(Application);
  About.Caption := 'about...';
end;

procedure TForm1.Exit1Click(Sender: TObject);
begin
  Close;
end;

procedure TForm1.Cascade1Click(Sender: TObject);
begin
  Cascade;
end;

procedure TForm1.Tile1Click(Sender: TObject);
begin
```

```
  Tile;
end;

procedure TForm1.Next1Click(Sender: TObject);
begin
  Next;
end;

procedure TForm1.Closeall1Click(Sender: TObject);
var
  chd : integer;
begin
  Application.ProcessMessages;
  for chd:=MDIChildCount-1 downto 0 do
    MDIChildren[chd].Close;
end;


procedure TForm1.Comm32Open(Sender: TObject; ProviderSubType,
  Error: Integer);
begin
  if Error <> 0 then
    StatusText.Caption := 'Error accessing device: '+Comm32.DeviceName+' (code:
'+IntToStr(Error)+')'
  else
  begin
    StatusText.Caption := 'Port '+Comm32.Devicename+' ready.';
    GetDataButt.Tag := 1;    {--- Disconnect ---}
    GetDataButt.Caption := 'Disconnect';
  end;
end;

procedure TForm1.Comm32Error(Sender: TObject; Errors: Integer);
begin
  if (Errors and CE_BREAK > 0) then
```

```
    StatusText.Caption := 'The hardware detected a break condition.';
  if (Errors and CE_DNS > 0) then
    StatusText.Caption := 'Windows 95 only: A parallel device is not selected.';
  if (Errors and CE_FRAME > 0) then
    StatusText.Caption := 'The hardware deteccted a framing error.';
  if (Errors and CE_IOE > 0) then
    StatusText.Caption := 'An I/O error occurred during communications with the device.';
  if (Errors and CE_MODE > 0) then
    StatusText.Caption := 'The requested mode is not supported, or the hFile parameter is
invalided. If this value is specofoed, it is the only valid error.';
  if (Errors and CE_OOP > 0) then
    StatusText.Caption := 'Windows 95 only: A parallel device signaled that it is out of
paper.';
  if (Errors and CE_OVERRUN > 0) then
    StatusText.Caption := 'A character-buffer overrun has occurred. The next character is
lost.';
  if (Errors and CE_PTO > 0) then
    StatusText.Caption :='Windows 95 only: A time-out occurred on a parallel device.';
  if (Errors and CE_RXOVER > 0) then
    StatusText.CAption := 'An input buffer overflow has occurred. There is either no room
in the input buffer, or a character was received after the end-of-file (EOF) character.';
  if (Errors and CE_RXPARITY > 0) then
    StatusText.Caption := 'The hardware detected a parity error.';
  if (Errors and CE_TXFULL > 0) then
    StatusText.Caption := 'The application tried to transmit a character, but the output
buffer was full.';
end;

procedure TForm1.Comm32RxFlag(Sender: TObject);
begin
  StatusText.Caption := 'Got the Data from Hardware.';
end;

procedure TForm1.Comm32Break(Sender: TObject);
begin
```

```
  StatusText.Caption := 'BREAK SIGNAL';
end;

procedure TForm1.Comm32RxChar(Sender: TObject; Count: Integer);
var
  Buffer : array[0..1024] of char;
  Bytes,i : integer;
begin
  FillChar(Buffer,SizeOf(Buffer),0);
  Bytes := Comm32.Read(Buffer,Count);
  if Bytes = -1 then
    StatusText.Caption := 'Error reading incoming data!'
  else
  begin
    for i:=0 to Bytes-1 do
    begin
      InputData[InputIndex] := ord(Buffer[i]);
      inc(InputIndex);
    end;
  end;
  CheckData;
end;

procedure TForm1.CheckData;
var
  i,Parity : integer;
begin
  if InputIndex > 31 then
  begin
    i := 0;
    repeat
      CVData[i] := InputData[i];
      inc(i);
    until  i=32;
    InputIndex := 0;
```

```
    ReTrans := False;
    {--- Check Parity Bit ---}
    Parity := CVData[1];
    for i:=2 to 31 do
      Parity := Parity xor CVData[i];
    if Parity <> CVData[0] then
    begin
      {--- Retransmiss ---}
      ReTrans := true;
      ReTransmit;
    end
    else
      {--- Data Transmission OK ---}
      if not reTrans then
      begin
        STatusText.Caption := 'Got the Data from Hardware.';
        TimeTemp := now;
        ConvertToInt(CVData);
        BuffWrite(ConvertData,TimeTemp);
      end;
  end;
{  Application.ProcessMessages;}
end;

procedure TForm1.ReTransmit;
var
  s : string;
  count : integer;
begin
  Comm32.PurgeIn;
  s := '0';
  count :=  Comm32.Write(s[1],Length(s));
  if count = -1 then
     StatusText.Caption := 'Retransmit signal Error.';
```

```
end;

procedure TForm1.Comm32TxEmpty(Sender: TObject);
begin
 StatusText.Caption := 'Data error! Retransmit the data request.';
end;

procedure TForm1.FormClose(Sender: TObject; var Action: TCloseAction);
var
 i : integer;
begin
 for i:= 0 to FileStr.Count-1 do
   FileStr.ObjectS[i].Free;
 FileStr.Free;
 MediaPlayer1.Close;
 Comm32.Close;
end;

end.
```

```
unit DSCD_2;

interface

uses
  Windows, Messages, SysUtils, Classes, Graphics, Controls, Forms, Dialogs,
  StdCtrls, Spin, ExtCtrls;

type
  TForm2 = class(TForm)
    Label1: TLabel;
    Label2: TLabel;
    Label3: TLabel;
    Label4: TLabel;
    Label5: TLabel;
    Label6: TLabel;
    Label7: TLabel;
    Label8: TLabel;
    Label9: TLabel;
    Label10: TLabel;
    Label11: TLabel;
    Label12: TLabel;
    Label13: TLabel;
    Label14: TLabel;
    Label15: TLabel;
    Label16: TLabel;
    Label17: TLabel;
    Label18: TLabel;
    Label19: TLabel;
    Label20: TLabel;
    Label21: TLabel;
    Label22: TLabel;
    Label23: TLabel;
    Label24: TLabel;
    Label25: TLabel;
    Label26: TLabel;
    Label27: TLabel;
    Label28: TLabel;
    Label29: TLabel;
    Label30: TLabel;
    Label31: TLabel;
    Label32: TLabel;
    Label33: TLabel;
    Label34: TLabel;
    Label35: TLabel;
    Label36: TLabel;
    Label37: TLabel;
    Label38: TLabel;
    Label39: TLabel;
    Label40: TLabel;
    Label41: TLabel;
    Label42: TLabel;
    Label43: TLabel;
    Label44: TLabel;
    Label45: TLabel;
    Label46: TLabel;
    Label47: TLabel;
    Label48: TLabel;
    Label49: TLabel;
    Label50: TLabel;
    Label51: TLabel;
    Label52: TLabel;
    Label53: TLabel;
    Label54: TLabel;
    Label55: TLabel;
    Label56: TLabel;
```

```
    Label57: TLabel;
    Label58: TLabel;
    Label59: TLabel;
    Label60: TLabel;
    Label61: TLabel;
    Label62: TLabel;
    Label63: TLabel;
    Bevel1: TBevel;
    GroupBox1: TGroupBox;
    SpinEdit1: TSpinEdit;
    SpinEdit2: TSpinEdit;
    Label64: TLabel;
    Label65: TLabel;
    Button1: TButton;
    procedure SpinEdit2Change(Sender: TObject);
    procedure Button1Click(Sender: TObject);
    procedure FormCreate(Sender: TObject);
  private
    { Private declarations }
  public
    { Public declarations }
  end;

var
  Form2: TForm2;
implementation
uses   DSCD_1;

{$R *.DFM}

procedure TForm2.SpinEdit2Change(Sender: TObject);
var
  Temp : TLabel;
  NameLabel : string[9];
begin
  if (SpinEdit2.Value mod 5 = 0 ) and
     (SpinEdit2.Value >= SpinEdit2.MinValue) and
     (SpinEdit2.Value <= SpinEdit2.MaxValue) then
  begin
     NameLabel := 'Label' + IntToStr(SpinEdit1.Value);
     Temp := TLabel(FindComponent(NameLabel));
     Temp.Caption := IntToStr(SpinEdit2.Value);
  end;
end;

procedure TForm2.Button1Click(Sender: TObject);
var
  i : integer;
  Temp : TLabel;
  NameLabel : string[9];
begin
  for i:=1 to 31 do
  begin
     NameLabel := 'Label' + IntToStr(i);
     Temp := TLabel(FindComponent(NameLabel));
     Form1.MaxVolt[i] := StrToInt(Temp.Caption);
  end;
  Close;
end;

procedure TForm2.FormCreate(Sender: TObject);
var
  i : Integer;
  Temp : TLabel;
  NameLabel : string[9];
begin
```

```
  if Form1.GetDataButt.Tag <> 0 then
  begin
    SpinEdit1.Enabled := False;
    SpinEdit2.Enabled := False
  end
  else
  begin
    SpinEdit1.Enabled := true;
    SpinEdit2.Enabled := true;
  end;
  for i:=0 to 31 do
  begin
    NameLabel := 'Label' + IntToStr(i);
    Temp := TLabel(FindComponent(NameLabel));
    Temp.Caption := IntToStr(Form1.MaxVolt[i]);
  end;
end;

end.
```

```
unit DSCD_3;

interface

uses
  Windows, Messages, SysUtils, Classes, Graphics, Controls, Forms, Dialogs,
  Menus, RChart, StdCtrls, ExtCtrls, Spin, ComCtrls;

type
  TForm3 = class(TForm)
    RChart1: TRChart;
    AddButt: TButton;
    CursorButt: TButton;
    AddEdit1: TSpinEdit;
    Label6: TLabel;
    Label7: TLabel;
    Shape1: TShape;
    Label1: TLabel;
    Shape2: TShape;
    Shape3: TShape;
    Label3: TLabel;
    Shape4: TShape;
    Label4: TLabel;
    Shape5: TShape;
    Label5: TLabel;
    Label2: TLabel;
    Bevel3: TBevel;
    Label8: TLabel;
    TimeLabel: TLabel;
    Label9: TLabel;
    Label10: TLabel;
    Label11: TLabel;
    Label12: TLabel;
    Label13: TLabel;
    RemoveButt: TButton;
```

```
    Label14: TLabel;
    RemoveEdit: TSpinEdit;
    Bevel1: TBevel;
    UpDown1: TUpDown;
    NormButt: TButton;
    Label15: TLabel;
    Bevel2: TBevel;
    Bevel4: TBevel;
    Bevel5: TBevel;
    ClearButt: TButton;
    CursorEdit: TEdit;
    procedure FormClose(Sender: TObject; var Action: TCloseAction);
    procedure FormCreate(Sender: TObject);
    procedure AddButtClick(Sender: TObject);
    procedure RemoveButtClick(Sender: TObject);
    procedure CursorButtClick(Sender: TObject);
    procedure ClearButtClick(Sender: TObject);
    procedure NormButtClick(Sender: TObject);
    procedure UpDown1Click(Sender: TObject; Button: TUDBtnType);
  private
    { Private declarations }
    LineNo : integer;
    GraphShow : array[1..31] of boolean;
    GraphNum : array[1..5] of integer;
    GraphColor : array[1..5] of TColor;
    LineLabel : array[1..5] of TLabel;
    LineEn : array[1..5] of boolean;
    LineCount : integer;
    ScaleX : array[1..4] of string;
    Cursor : integer;
    MaxY : integer;
    Zoom : integer;
  public
    { Public declarations }
    procedure WriteGraph;
```

```
  end;

var
  Form3: TForm3;

implementation
uses   DSCD_1;
{$R *.DFM}

procedure TForm3.WriteGraph;
var
  i,Point,Line : integer;
  LastPoint : array[1..5] of real;
begin
  with Form1 do
  begin
    RChart1.ClearGraf;
    {--- Draw Graph ---}
    for i:=1 to 5 do
      LastPoint[i] := LsPoint[GraphNum[i]];
    for Point:=1 to index do
      for Line:= 1 to 5 do
        if LineEn[Line] then
        begin
          RChart1.DataColor := GraphColor[Line];
          RChart1.MoveTo(Point-1,LastPoint[Line]);
          RChart1.DrawTo(Point,BuffData[Point,GraphNum[Line]]);
          LastPoint[Line] := BuffData[Point,GraphNum[Line]];
        end;
    TimeLabel.Caption := TimeToStr(BuffDTime[index]);
    {--- Set Range---}
    if Zoom = 1 then
      NormButt.Click;
    {--- Set scale ---}
    if LineCount <> 0 then
```

```
    begin
      ScaleX[1] := TimeToStr(BuffDTime[1]);
      if index >= 50 then
        ScaleX[2] := TimeToStr(BuffDTime[50])
      else   ScaleX[2] := '';
      if index >=100 then
        ScaleX[3] := TimeToStr(BuffDTime[100])
      else   ScaleX[3] := '';
      if index >= 150 then
        ScaleX[4] := TimeToStr(BuffDTime[150])
      else   ScaleX[4] := '';
    end;
    RChart1.UserTickTextX := ScaleX[1]+'|'+ScaleX[2]+'|'+
                             ScaleX[3]+'|'+ScaleX[4];
  end;
  RChart1.ShowGraf;
  Application.ProcessMessages;
end;

procedure TForm3.FormClose(Sender: TObject; var Action: TCloseAction);
begin
  Form1.GraphEn[Tag] := false;
  Action := caFree;
end;

procedure TForm3.FormCreate(Sender: TObject);
var
  i : integer;
  NameLabel : string[9];
begin
  for i:=1 to 31 do
    GraphShow[i] := false;
  AddEdit1.Value := 1;
  LineNo := 0;
  LineCount := 0;
```

```
      repeat
        inc(i);
      until LineEn[i] = false;
      LineNo := i;
      LineEn[i] := true;
      inc(LineCount);
      GraphNum[LineNo] := AddEdit1.Value;
      LineLabel[LineNo].Caption := 'H/D No.' + IntToStr(GraphNum[LineNo]);
    end;
  RemoveButt.Enabled := true;
  if LineCount>=5 then
    AddButt.Enabled := false;
  WriteGraph;
end;

procedure TForm3.RemoveButtClick(Sender: TObject);
var
  i : integer;
begin
  if (RemoveEdit.Value >= RemoveEdit.MinValue) and
    (RemoveEdit.Value <= RemoveEdit.MaxValue) and
    (LineEn[RemoveEdit.Value]) then
  begin
    GraphShow[GraphNum[RemoveEdit.Value]] := false;
    LineLabel[RemoveEdit.Value].Caption := '';
    LineEn[RemoveEdit.Value] := false;
    AddButt.Enabled := true;
    LineCount := LineCount-1;
  end;
  if LineCount = 0 then
  begin
    RemoveEdit.Enabled := false;
    for i:= 1 to 4 do
      ScaleX[i] := '';
  end;
```

```
  WriteGraph;
end;

procedure TForm3.CursorButtClick(Sender: TObject);
var
  V : real;
  Code : integer;
begin
  Val(CursorEdit.Text,V,Code);
  if Code=0 then
    if (V>=RChart1.RangeLoY) and (V<=RChart1.RangeHiY) and (Cursor<4) then
    begin
      inc(Cursor);
      ClearButt.Enabled := true;
      RChart1.CrossHairPosY[Cursor] := V;
    end;
  if Cursor>=4 then
    CursorButt.Enabled := false;
end;

procedure TForm3.ClearButtClick(Sender: TObject);
var
  i : integer;
begin
  Cursor := 0;
  ClearButt.Enabled := false;
  CursorButt.Enabled := true;
  with RChart1 do
    for i:=1 to 4 do
      CrossHairPosY[i] := -10;
end;

procedure TForm3.NormButtClick(Sender: TObject);
var
  Line : integer;
```

```
begin
 with Form1 do
 begin
   MaxY := 5;
   for Line:=1 to 5 do
     if LineEn[Line] and (MaxVolt[GraphNum[Line]] >= MaxY) then
       MaxY := MaxVolt[GraphNum[Line]];
   RChart1.RangeHiY := MaxY+2;
   RChart1.SetRange(0,0,180,RChart1.RangeHiY);
 end;
 Zoom := 1;
 UpDown1.Position := 1;
 RChart1.ShowGraf;
end;

procedure TForm3.UpDown1Click(Sender: TObject; Button: TUDBtnType);
var
 MidY,ry : single;
begin
 with RChart1 do
 begin
   if UpDown1.Position>Zoom then
   begin
     MidY := (RangeHiY+RangeLoY)/2;
     ry := (RangeHiY-RangeLoY)/2.8;
     SetRange(0,0,180,MidY+ry);
     ShowGraf;
   end
   else
   begin
     MidY := (RangeHiY+RangeLoY)/2;
     ry := 1.4*(RangeHiY-RangeLoY)/2;
     SetRange(0,0,180,MidY+ry);
     showGraf;
   end;
```

```
    end;
    Zoom := UpDown1.Position;
  end;

  end.
```

```
unit DSCD_4;

interface

uses
  Windows, Messages, SysUtils, Classes, Graphics, Controls, Forms, Dialogs,
  StdCtrls, Spin, ExtCtrls, RChart, Grids, ComCtrls;

type
  TForm4 = class(TForm)
    RChart1: TRChart;
    Label6: TLabel;
    Label7: TLabel;
    Shape1: TShape;
    Label1: TLabel;
    Shape2: TShape;
    Label2: TLabel;
    Shape3: TShape;
    Label3: TLabel;
    Shape4: TShape;
    Label4: TLabel;
    Shape5: TShape;
    Label5: TLabel;
    Bevel3: TBevel;
    Label9: TLabel;
    Label10: TLabel;
    Label11: TLabel;
    Label12: TLabel;
    Label13: TLabel;
    Label14: TLabel;
    AddButt: TButton;
    CursorButt: TButton;
    AddEdit1: TSpinEdit;
    RemoveButt: TButton;
    RemoveEdit: TSpinEdit;
    Bevel1: TBevel;
    UpDown1: TUpDown;
    NormButt: TButton;
    Label15: TLabel;
    Bevel2: TBevel;
    Bevel4: TBevel;
    Bevel5: TBevel;
    StringGrid1: TStringGrid;
    ClearButt: TButton;
    CursorEdit: TEdit;
    procedure FormCreate(Sender: TObject);
    procedure StringGrid1DrawCell(Sender: TObject; Col, Row: Integer;
      Rect: TRect; State: TGridDrawState);
    procedure FormClose(Sender: TObject; var Action: TCloseAction);
    procedure AddButtClick(Sender: TObject);
    procedure RemoveButtClick(Sender: TObject);
    procedure CursorButtClick(Sender: TObject);
    procedure ClearButtClick(Sender: TObject);
    procedure NormButtClick(Sender: TObject);
    procedure UpDown1Click(Sender: TObject; Button: TUDBtnType);
  private
    { Private declarations }
    LoadData : array[1..150,1..31] of real;
    LoadTime : array[1..150] of TDateTime;
    LoadIndex : integer;
    LineNo : integer;
    GraphShow : array[1..31] of boolean;
    GraphNum : array[1..5] of integer;
    GraphColor : array[1..5] of TColor;
    LineLabel : array[1..5] of TLabel;
    LineEn : array[1..5] of boolean;
```

```
    LineCount : integer;
    ScaleX : array[1..4] of string;
    Cursor : integer;
    Zoom : integer;
    procedure DrawGraph;
  public
    { Public declarations }
    MVolt : array[1..31] of integer;
    procedure Convert;
  end;

var
  Form4: TForm4;

implementation
uses  DSCD_1;

{$R *.DFM}

procedure TForm4.DrawGraph;
var
  Point,L : integer;
begin
  With RChart1 do
  begin
    ClearGraf;
    {--- Draw Graph ---}
    for L := 1 to 5 do
      if LineEn[L] then
      begin
        MoveTo(0,0);
        for Point:= 1 to LoadIndex do
        begin
          DataColor := GraphColor[L];
          DrawTo(Point,LoadData[Point,GraphNum[L]]);
        end;
      end;
    {--- Set range ---}
    if Zoom = 1 then
      NormButt.Click;
    {--- Set scale ---}
    if LineCount <> 0 then
    begin
      ScaleX[1] := TimeToStr(LoadTime[1]);
      if LoadIndex >= 50 then
        ScaleX[2] := TimeToStr(LoadTime[50]);
      if LoadIndex >=100 then
        ScaleX[3] := TimeToStr(LoadTime[100]);
      if LoadIndex >= 150 then
        ScaleX[4] := TimeToStr(LoadTime[150]);
    end;
    UserTickTextX := ScaleX[1]+'|'+ScaleX[2]+'|'+
                     ScaleX[3]+'|'+ScaleX[4];
  ShowGraf;
  end;
end;


procedure TForm4.Convert;
var
  rw,cl,code : integer;
begin
  with StringGrid1 do
  begin
    rw := 1;
    while (Cells[0,rw]<>'') and (rw<=150) do
```

```
      begin
        LoadTime[rw] := StrToDateTime(Cells[0,rw]);
        for cl:=1 to 31 do
          val(Cells[cl,rw],LoadData[rw,cl],code);
        inc(rw);
      end;
    end;
    LoadIndex := rw-1;
  end;

  procedure TForm4.FormCreate(Sender: TObject);
  var
    rw,cl : integer;
    i : integer;
    NameLabel : string[9];
  begin
    with StringGrid1 do
    begin
      ColWidths[0] := 105;
      for rw:=0 to Row-1 do
        for cl:=0 to Col-1 do
          Cells[cl,rw] := '';
    end;

    for i:=1 to 31 do
      GraphShow[i] := false;
    AddEdit1.Value := 1;
    LineNo := 0;
    LineCount := 0;
    for i:=1 to 5 do
    begin
      NameLabel := 'Label' + IntToStr(i);
      LineLabel[i] := TLabel(FindComponent(NameLabel));
      LineEn[i] := false;
    end;
    GraphColor[1] := clBlue;
    GraphColor[2] := clRed;
    GraphColor[3] := clPurple;
    GraphColor[4] := clFuchsia;
    GraphColor[5] := clGreen;
    for i:=1 to 4 do
      ScaleX[i] := '';
    with RChart1 do
    begin
      CrossHairSetup(1,clNavy,chHoriz,psDot,1);
      CrossHairSetup(2,clPurple,chHoriz,psDash,1);
      CrossHairSetup(3,clMaroon,chHoriz,psDashDot,1);
      CrossHairSetup(4,clOlive,chHoriz,psDashDotDot,1);
    end;
    ClearButt.Click;
    Zoom := 1;
  end;

  procedure TForm4.StringGrid1DrawCell(Sender: TObject; Col, Row: Integer;
    Rect: TRect; State: TGridDrawState);
  var
    bufB : array[0..79] of char;
    algn : word;
  begin
    if (Row=0) or (Col=0) then
      algn := dt_Center
    else    algn := dt_Right;
    strPCopy(bufB,StringGrid1.Cells[Col,Row]);
    StringGrid1.Canvas.FillRect(Rect);
    DrawText(StringGrid1.Canvas.Handle,bufB,-1,Rect,algn);
  end;
```

```
procedure TForm4.FormClose(Sender: TObject; var Action: TCloseAction);
begin
  Action := caFree;
end;

procedure TForm4.AddButtClick(Sender: TObject);
var
  i : integer;
begin
  if (AddEdit1.Value >= AddEdit1.MinValue) and
     (AddEdit1.Value <= AddEdit1.MaxValue) then
     if not GraphShow[AddEdit1.Value] then
     begin
       GraphShow[AddEdit1.Value] := true;
       i := 0;
       repeat
         inc(i);
       until LineEn[i] = false;
       LineNo := i;
       LineEn[i] := true;
       inc(LineCount);
       GraphNum[LineNo] := AddEdit1.Value;
       LineLabel[LineNo].Caption := 'H/D No.' + IntToStr(GraphNum[LineNo]);
     end;
  RemoveButt.Enabled := true;
  if LineCount>=5 then
     AddButt.Enabled := false;
  DrawGraph;
end;

procedure TForm4.RemoveButtClick(Sender: TObject);
var
  i : integer;
begin
  if (RemoveEdit.Value >= RemoveEdit.MinValue) and
     (RemoveEdit.Value <= RemoveEdit.MaxValue) and
     (LineEn[RemoveEdit.Value]) then
  begin
     GraphShow[GraphNum[RemoveEdit.Value]] := false;
     LineLabel[RemoveEdit.Value].Caption := '';
     LineEn[RemoveEdit.Value] := false;
     AddButt.Enabled := true;
     LineCount := LineCount-1;
  end;
  if LineCount = 0 then
  begin
     RemoveEdit.Enabled := false;
     for i:= 1 to 4 do
       ScaleX[i] := '';
  end;
  DrawGraph;
end;


procedure TForm4.CursorButtClick(Sender: TObject);
var
  V : real;
  Code : integer;
begin
  Val(CursorEdit.Text,V,Code);
  if Code=0 then
     if (V>=RChart1.RangeLoY) and (V<=RChart1.RangeHiY) and (Cursor<4) then
     begin
       inc(Cursor);
       ClearButt.Enabled := true;
```

```
      RChart1.CrossHairPosY[Cursor] := V;
    end;
  if Cursor>=4 then
    CursorButt.Enabled := false;
end;

procedure TForm4.ClearButtClick(Sender: TObject);
var
  i : integer;
begin
  Cursor := 0;
  ClearButt.Enabled := false;
  CursorButt.Enabled := true;
  with RChart1 do
    for i:=1 to 4 do
      CrossHairPosY[i] := -10;
end;

procedure TForm4.NormButtClick(Sender: TObject);
var
  MaxY,L : integer;
begin
  MaxY := 5;
  for L:=1 to 5 do
    if LineEn[L] and (MVolt[GraphNum[L]] >= MaxY) then
      MaxY := MVolt[GraphNum[L]];
  RChart1.RangeHiY := MaxY+2;
  RChart1.SetRange(0,0,180,RChart1.RangeHiY);
  Zoom := 1;
  UpDown1.Position := 1;
  RChart1.ShowGraf;
end;

procedure TForm4.UpDown1Click(Sender: TObject; Button: TUDBtnType);
var
  MidY,ry : single;
begin
  with RChart1 do
  begin
    if UpDown1.Position>Zoom then
    begin
      MidY := (RangeHiY+RangeLoY)/2;
      ry := (RangeHiY-RangeLoY)/2.8;
      SetRange(0,0,180,MidY+ry);
      ShowGraf;
    end
    else
    begin
      MidY := (RangeHiY+RangeLoY)/2;
      ry := 1.4*(RangeHiY-RangeLoY)/2;
      SetRange(0,0,180,MidY+ry);
      showGraf;
    end;
  end;
  Zoom := UpDown1.Position;
end;

end.
```

```
unit DSCD_5;

interface

uses
  Windows, Messages, SysUtils, Classes, Graphics, Controls, Forms, Dialogs,
  ExtCtrls;

type
  TForm5 = class(TForm)
    Image1: TImage;
  private
    { Private declarations }
  public
    { Public declarations }
  end;

var
  Form5: TForm5;

implementation

{$R *.DFM}

end.
```

```
unit DSCD_6;

interface

uses
  Windows, Messages, SysUtils, Classes, Graphics, Controls, Forms, Dialogs,
  StdCtrls, Buttons, ExtCtrls;

type
  TForm6 = class(TForm)
    Image1: TImage;
    BitBtn1: TBitBtn;
    Label1: TLabel;
    Label2: TLabel;
    Image2: TImage;
    Image3: TImage;
    Image4: TImage;
    Label3: TLabel;
    Label4: TLabel;
    Label5: TLabel;
    Label6: TLabel;
    Label7: TLabel;
    Label8: TLabel;
    procedure BitBtn1Click(Sender: TObject);
  private
    { Private declarations }
  public
    { Public declarations }
  end;

var
  Form6: TForm6;

implementation

{$R *.DFM}

procedure TForm6.BitBtn1Click(Sender: TObject);
begin
  Close;
end;

end.
```

```
#include <reg51.h>

unsigned char xdata *pA  =0xF800;
unsigned char xdata *pB  =0xF801;
unsigned char xdata *pC  =0xF802;
unsigned char xdata *pCd =0xF803;
unsigned char data  *out;

int data hint=0;

void delay(int d);

void main(void)
{  int   count256,count16,p,i,parity,sixteen;
   int   b=0;
   int   inc=0;

   delay(10);

   /* --- Initial Setting --- */
   *pCd=0x82;  /* port B is input port */
   *pA=0x00;
   *pB=0x00;
   *pC=0x00;
   TMOD=0x22;
   SCON=0x52;
   PCON=0x80;
   TH0=-230;
   TH1=-3;
   TR0=1;
   TR1=1;
   IE=0x00;

   out=0x0051;
   for (i=0; i<32 ;++i)
```

```
{  *out=0;
   out++;
}
out=0x0060;
*out=0;

/* --- Start Working --- */
while(1)
{  out=0x0050;
   inc=0;
   b=0;
   count256=1;
   count16=1;
   hint=0;

   /* --- Pulse 256 (DutyCycle=25%) --- */
   while(count256<=256)
   {  *pA=0x01;
      inc++;
            while(!TF0);
            TF0=0;
      *pA=0x10;
            while(!TF0);
            TF0=0;
      if (count256>=9)
      {  b=b<<1;
         p=*pB;
         b=b|p;
      }
      if (count256==136)
         sixteen=b;
      ++count256;
            while(!TF0);
            TF0=0;
      if ((inc==8) && (count256>=9))
```

```
  { inc=0;
    *out=b;
    out++;
    b=0;
  }
        while(!TF0);
        TF0=0;
}

/* --- Parity bit --- */
out=0x0060;
*out=sixteen;
out=0x0051;
parity=*out;
for (i=2; i<32; i++)
{ ++out;
  parity=parity^(*out);
}
out=0x0050;
*out=parity;

/* --- Transmiss the Data --- */
out=0x004F;
while(++out<=0x006F)
{ while(!TI);
  TI=0;
  SBUF=*out;
}
TI=0;
RI=0;
IE=0x90;

/* --- Test check the Collecting Data --- */
out=0x004F;
for (i=0; i<32 ;i++)
```

```
    { out++;
      if (*out!=0)
      { P1=*out;
      }
    }

    /* --- Pulse blanking wait Transmiss --- */
    while ((count16<=25)&&(!hint))
    { count16++;
          while(!TF0);
          TF0=0;
          while(!TF0);
          TF0=0;
          while(!TF0);
          TF0=0;
          while(!TF0);
          TF0=0;
    }

    IE=0x00;
    count16=1;

    /* --- Pulse blanking --- */
    while(count16<=5)
    { count16++;
          while(!TF0);
          TF0=0;
          while(!TF0);
          TF0=0;
          while(!TF0);
          TF0=0;
          while(!TF0);
          TF0=0;
          while(!TF0);
          TF0=0;
    }
  }
```

```
}

void serial_service(void) interrupt 4
{
  if (RI==1)
  {  hint=1;
     /* --- Retransmiss --- */
     out=0x004F;
     while(++out<=0x006F)
     { while(!TI);
       TI=0;
       SBUF=*out;
     }
     TI=0;
     RI=0;
  }
}

void delay(int d)
{  int i,j;
    for (i=0 ; i<d ;i++)
      for (j=0 ; j<500 ; j++);
}
```

เอกสารอ้างอิง

1. ไพบูลย์ ธานินทร์สุรัตน์ และ ผศ. พลผดุง ผดุงกุล , " การควบคุมและแสดงผลแบบพัลส์วิดท์ดิฟเฟอเรนเชียลซีเควนเชียลคอนโทรล " , การประชุมวิชาการทางวิศวกรรมไฟฟ้า ครั้งที่ 18 , หน้า 444-447.
2. สุนทร วิทูสุรพจน์ , " การใช้งานไมโครคอนโทรลเลอร์ ตระกูล 8051" , บริษัท ซีเอ็ดยูเคชั่น จำกัด (มหาชน) .

กิตติกรรมประกาศ

ปริญญานิพนธ์ เรื่องการส่งข้อมูลดิจิตอลแบบเป็นลำดับ (Data Sequtial Collecting) ฉบับนี้ สำเร็จลงได้ด้วยดี เพราะได้รับความเอื้อเฟื้อเป็นอย่างดีในด้านต่างๆ จาก อาจารย์พลผดุง ผดุงกุล ซึ่งเป็นอาจารย์ที่ปรึกษาทั้งในด้านการออกแบบวงจร ,การเขียนโปรแกรม ตลอดจนการแก้ไขปัญหาต่างๆ ที่เกิดขึ้น รวมทั้งเพื่อนๆ ที่ให้ความช่วยเหลือในด้านต่างๆ พร้อมให้กำลังใจในการทำงาน ทำให้ปริญญานิพนธ์ฉบับนี้สำเร็จเป็นรูปเล่มที่สมบูรณ์ลงได้ จึงขอขอบพระคุณอาจารย์ และขอบคุณเพื่อนๆ ทุกคนมา ณ. โอกาสนี้ด้วย