เครื่องทดสอบไอซี PLD

PLD Tester

โดย

นายพูลศักดิ์ ขจรกลิ่น 43015740

นายสัณหภาส สะอาดดี 43015753

ปริญญานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรอุตสาหกรรมศาสตร์บัณฑิต

ภาควิชาวิศวกรรมสารสนเทศ

คณะวิศวกรรมศาสตร์

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

ปีการศึกษา 2544

หัวข้อปริญญานิพนธ์	เครื่องทดสอบไอซี PLD PLD Tester	
โดย	นายพูลศักดิ์	ขจรกลิ่น 43015740
	นายสัณหภาส	สะอาดดี 43015753
อาจารย์ที่ปรึกษา	ผศ.ไพศาล	สิทธิโยภาสกุล
ภาควิชา	วิศวกรรมสารสนเทศ	
ปีการศึกษา	2544	

ปริญญานิพนธ์ฉบับนี้ได้รับการอนุมัติให้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรอุตสาหกรรมศาสตร์บัณฑิต คณะวิศวกรรมศาสตร์ สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

(.....)

อาจารย์ผู้ควบคุมปริญญานิพนธ์

หัวข้อปริญญานิพนธ์	**เครื่องทดสอบไอซี PLD** **PLD Tester**	
โดย	**นายพูลศักดิ์ ขจรกลิ่น**	**43015740**
	นายสัณหภาส สะอาดดี	**43015753**
อาจารย์ที่ปรึกษา	**ผศ. ไพศาล สิทธิโยภาสกุล**	

บทคัดย่อ

ในปัจจุบันนี้จะสังเกตได้ว่าอุปกรณ์ประเภทไอซีได้ถูกนำมาใช้ประกอบในวงจรอิเล็กทรอนิกส์อย่างมาก เนื่องจากมีขนาดเล็กประหยัดเนื้อที่ของการประกอบวงจรทำให้วงจรอิเล็กทรอนิกส์ที่ประกอบด้วยไอซีมีขนาดที่เล็กลง

ดังนั้นจึงเป็นเหตุผลที่ทำให้ทางกลุ่มโครงงานของเราได้คิดค้นทำเครื่องทดสอบไอซีขึ้นมา ซึ่งก็คือเครื่องทดสอบไอซี PLD (PLD Tester) โดยขีดความสามารถคือใช้ทดสอบไอซีตระกูล PLD โดยไอซีที่ใช้ทดสอบนี้มีขาได้สูงสุด 24 ขา เครื่องทดสอบไอซี PLD ประกอบด้วย 2 ส่วน คือ ส่วนของฮาร์ดแวร์สำหรับตรวจสอบสถานะของไอซีที่นำมาทดสอบแล้วนำผลที่ได้มาประมวลผลที่คอมพิวเตอร์ ส่วนที่สองคือ ซอฟท์แวร์สำหรับประมวลผลสถานะของไอซีที่ทดสอบแล้วแสดงผลในรูปของ Timing Diagram ซึ่งการติดต่อระหว่างฮาร์ดแวร์กับคอมพิวเตอร์นั้นจะติดต่อผ่านทางพอร์ตอนุกรม

THESIS	**PLD Tester**		
NAME	**Poonsak**	**Khajonklin**	**43015740**
	Sanhapas	**Sa-arddee**	**43015753**
ADVISER	**Asst.Prof. Paisan Sitiyopasakul**		
COURSE	**Bachelor of Industrial Technology in Electronics**		
DEPARTMENT	**Information Engineering**		
YEAR	**2001**		

Abstract

In the present we can found the kind of IC component. It was fitted in the electronic circuits. Because of smaller and save area for fitted circuits. As replace electronic circuits with IC component are smaller.

So that, It is the important reason which we taked design something to test IC that is PLD Tester. It can used to test the group of IC PLD, and IC that used to test have highest 24 pin with combined two parts. Part one is Hardware that used to check IC status and take a result to process on computer. Part two is software for process status of IC after test. It show a result in Timing Diagram. The connection of Hardware between Computer are connected serial port.

กิตติกรรมประกาศ

ปริญญานิพนธ์ฉบับนี้สำเร็จเรียบร้อยได้ ด้วยความกรุณาในการให้คำแนะนำคำปรึกษา การเสนอแนะแนวทางแก้ไขและข้อบกพร่องต่าง ๆ จากอาจารย์ไพศาล สิทธิโยภาสกุล อาจารย์ที่ปรึกษาในการทำโครงงานในครั้งนี้ที่ให้ความช่วยเหลือตั้งแต่แรกเริ่มศึกษารวมทั้งซอฟท์แวร์ที่สำคัญคณะผู้จัดทำปริญญานิพนธ์ทุกคนขอกราบขอบพระคุณอย่างสูง

นอกจากนี้คณะผู้จัดทำปริญญานิพนธ์ขอขอบคุณรุ่นพี่และเพื่อนๆที่ช่วยแนะนำ ข้อมูลและหนังสือในการเขียนโปรแกรมและทดสอบโปรแกรม พร้อมทั้งให้ข้อเสนอแนะที่เป็นประโยชน์อย่างยิ่งจนทำให้โปรเจคชิ้นนี้ทำงานได้อย่างมีประสิทธิภาพ

สุดท้ายนี้คณะผู้จัดทำปริญญานิพนธ์ขอกราบขอบพระคุณ คุณพ่อ คุณแม่ ผู้เป็นที่รักและเคารพอย่างสูง พี่น้องทุกคนที่ให้ความช่วยเหลือด้านทุนทรัพย์และเป็นกำลังใจด้วยดี รวมถึงขอบคุณเพื่อน ๆ ทุกคนที่ช่วยเขียนและทดสอบโปรแกรม พร้อมทั้งให้ข้อเสนอแนะที่เป็นประโยชน์อย่างยิ่ง

นายพูลศักดิ์ ขจรกลิ่น 43015740
นายสัณหภาส สะอาดดี 43015753

คณะผู้จัดทำปริญญานิพนธ์

สารบัญ

เอกสารอ้างอิง

ภาคผนวก

ก. Software Program

ข. วงจรและลายทองแดงของฮาร์ดแวร์

ค. Data Sheet

สารบัญรูป

หน้า

สารบัญตาราง

หน้า

บทที่ 1
บทนำ

1.1 ความเป็นมาของหัวข้อวิทยานิพนธ์

วงจรดิจิตอลทั่วไปจะมีสภาวะการทำงาน 2 สภาวะคือ High และ Low ทางไฟฟ้าสามารถแทนด้วยระดับแรงดันไฟสูงและแรงดันไฟต่ำ ในกรณีที่ไม่ได้ต่อสายสัญญาณหรือสายสัญญาณของวงจรถูกตัดขาดจะเรียกสภาวะนี้ว่า Hi Impedance ระบบดิจิตอลจะประกอบด้วยลอจิกเกท โดยจะเป็นวงจรอิเล็กทรอนิกส์ที่ใช้ระดับไฟเป็นตัวแปรทางลอจิกในการทำงานของวงจรทั้งสัญญาณเข้าและสัญญาณออก เกทที่ใช้งานทั่วไปคือ AND Gate, OR Gate, NOT Gate, NAND Gate, NOR Gate และ Exclusive-OR Gate การออกแบบวงจรลอจิกในปัจจุบันโดยมากจะใช้ไอซีประเภท Programmable Logic Devices (PLDs) ซึ่งจะมีส่วนประกอบของวงจรอยู่ในรูปของ Sum of Product From คือเอาเทอมผลคูณที่เกิดขึ้นจากการ AND ระหว่างตัวแปรแล้วนำมา OR กัน จากหลักการนี้ทำให้เกิดแนวคิดในการคิดทำเครื่องทดสอบไอซี PLD - ขึ้นมา

ดังนั้นจึงสามารถกล่าวถึงการทำงานในหน้าที่ต่าง ๆ ได้ดังนี้

1.1.1 การวิเคราะห์และทดสอบไอซี PLD

การอ่านและวิเคราะห์หาสมการจาก ไอซีโปรแกรม PAL จะเป็นส่วนที่ทำหน้าที่คล้ายกับโลจิกอนาไลเซอร์ โดย PAL ที่ต้องการอ่านจะต้องเสียบอยู่บนซ็อกเก็ต (Test Socket) และตัวโปรแกรมจะทำการส่งอินพุทจากค่าต่ำสุดไปจนถึงค่าสูงสุด ซึ่งค่าสูงสุดนี้จะขึ้นอยู่กับจำนวนอินพุทที่ป้อนเข้าไป เมื่อโปรแกรมส่งค่าไปให้กับ PAL หนึ่งครั้ง ก็จะทำการอ่านค่าเอาท์พุทของ PAL มาหนึ่งครั้งจนกว่าจะครบโดยค่าที่ได้นี้จะถูกเก็บไว้ใน Array หลังจากนั้นโปรแกรมจะนำข้อมูลที่เก็บไว้มาวิเคราะห์หาสมการทางลอจิกภายใต้การวิเคราะห์ด้วยซอฟท์แวร์ โดยจะสามารถวิเคราะห์หาสมการได้ทั้งรูปแบบของ Minterm ตามลักษณะการโปรแกรมของไอซี ซึ่งจะสามารถวิเคราะห์หาสมการได้สูงสุด 8 เอาท์พุท จากการรับค่าอินพุทได้สูงสุด 16 อินพุท

1.3 การวิเคราะห์ระดับสัญญาณทางลอจิก

สำหรับในส่วนของการวิเคราะห์ระดับสัญญาณทางลอจิกนั้น สามารถวัดได้สูงสุด 24 ขาและสามารถวิเคราะห์ระดับสัญญาณได้ 2 ระดับคือ ระดับลอจิก LOW และลอจิก HIGH โดยแสดงให้เห็นในรูปของ LED ติดและดับที่หน้าจอมอนิเตอร์และยังแสดงให้เห็นในรูปของกราฟ Timing Diagram โดยสามารถขยายให้รูปคลื่นมีคาบเวลาที่กว้างขึ้นและเลื่อนดูรูปคลื่นที่เวลาต่างๆ ได้

บทที่ 2
ทฤษฎีที่ใช้ในโครงงาน

2.1 Programmable Array Logic (PAL)

2.1.1 แนวความคิดที่นำ PAL มาใช้งาน

PAL (Programable Array Logic) เป็นอุปกรณ์ที่ถูกพัฒนามาเพื่อช่วยให้ สามารถทำฟังก์ชั่นลอจิกที่ยุ่งยาก ซับซ้อน ซึ่งประกอบด้วยหลาย ๆ เอาท์พุท ให้สามารถใช้ PAL เพียงตัวเดียว โดยทราบเพียงสถานะของอินพุท และเอาท์พุท ของฟังก์ชั่นลอจิก ก็สามารถจะทำให้ PAL สามารถเขียนแบบฟังก์ชั่นลอจิก ที่ยุ่งยากได้ โดยไม่ต้องอาศัยลอจิกเกท (Logic Gate) หลาย ๆ ตัวมาประกอบกัน เพื่อให้ได้วงจรที่มีคุณสมบัติเหมือนกับฟังก์ชั่นลอจิก การพัฒนาที่รวดเร็วในเทคโนโลยี LSI (Large Scale Integration) สามารถนำไปสู่มาตรฐานที่กว้างขวางขึ้น

ปัจจุบัน IC (Intergate Circuit) เพียงตัวเดียว สามารถทำหน้าที่แทนวงจรสำเร็จของเดิมได้ ในขณะเดียวกันกับที่การพัฒนา LIS ได้พัฒนาขึ้นมากทำให้สามารถผลิตได้ โดยใช้ต้นทุนต่ำ ถึงแม้กระนั้นการใช้อุปกรณ์พวก LIS ก็ยังต้องใช้ร่วมกับอุปกรณ์ SSI/MSI (Small Scale Intergration / Medium Scale Intergration) เพื่อให้สามารถเชื่อมต่อกับระบบผู้ใช้งาน ผู้ออกแบบสามารถที่จะออกแบบให้หลากหลายเพื่อใช้งานได้

PAL จึงเป็นอุปกรณ์ที่เป็นทางออกที่ดีที่สุดในปัจจุบัน ที่จะนำมาใช้ในการออกแบบ Function Logic จากเดิมที่เราใช้อุปกรณ์พวก LSI และ MSI เพราะสามารถออกแบบได้ง่ายไม่ยุ่งยาก PAL เป็นทางเลือกใหม่ในการออกแบบวงจรคอมไบเนชั่นลอจิก และซีเควนเชียลลอจิกมีความกะทัดรัด ยืดหยุ่นและง่ายต่อการใช้งานทั้งการออกแบบใหม่ และแทนที่วงจรเก่า ซึ่งในอนาคต PAL จะมีราคาถูกลง และประสิทธิภาพการใช้งานที่สูงขึ้นกว่าปัจจุบัน

2.1.2 การ AND และการ OR

โดยส่วนใหญ่แล้ว PAL ถูกสร้างในรูปของผลคูณลอจิก (Sum Of Products Logic) โดยการทำโปรแกรม AND array ซึ่งค่าของเอาท์พุทที่ได้นำไปยัง OR array ที่ไม่ได้มีการ เปลี่ยนแปลง ซึ่งผลรวมของผลคูณสามารถหาได้จากการเปลี่ยนแปลงสมการบูลีน (Boolean) โดย PAL สามารถใช้ได้อย่างไม่มีขีดจำกัด โดยสามารถจะเปลี่ยนแปลงค่าใน AND - OR array โดยที่

โดยที่ PAL จะให้ข้อแตกต่างในด้านประสิทธิภาพสูงสุด รูปที่ 2.1 แสดงโครงสร้างพื้นฐานของ PAL ที่มี 2 อินพุทกับ 1 เอาท์พุท โดยในที่นี้ สมการลอจิกทั่ว ๆ ไปคือ

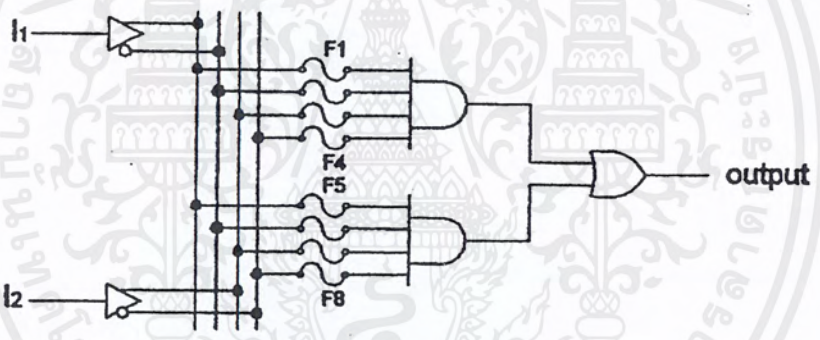
$$\text{Output}=(i1+\bar{f}1)(\bar{i}1+\bar{f}2)(i2+\bar{f}3)(\bar{i}2+\bar{f}4)+$$
$$(i1+\bar{f}5)(\bar{i}1+\bar{f}6)(i2+\bar{f}7)(\bar{i}2+\bar{f}8)$$

โดยที่ f คือ สถานะปัจจุบันของเส้นฟิวส์ใน PAL ซึ่งเส้นฟิวส์ที่ไม่ถูกทำลายจะมีค่าลอจิก "1" ดังนั้น

ฟิวส์ที่ถูกทำลาย : f=0

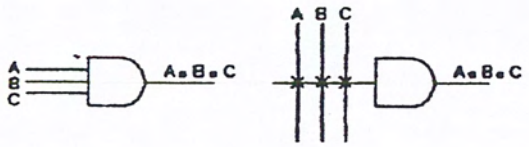
ฟิวส์ที่ไม่ถูกทำลาย : f=1

PAL ที่ยังไม่ได้ทำโปรแกรมฟิวส์ทั้งหมดจะถูกต่อไว้คงที่สมบูรณ์

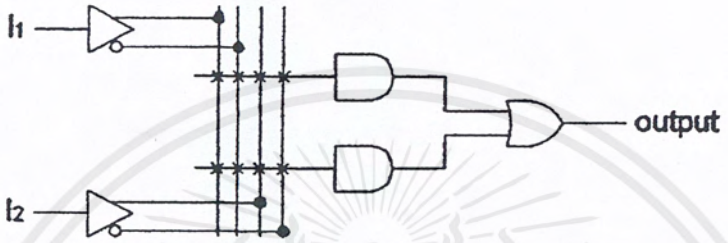


รูปที่ 2.1 แสดงโครงสร้างพื้นฐานของ PAL

ในสมการลอจิกจะมีความสะดวกในการออกแบบสำหรับระบบเล็ก แต่ความยุ่งยากจะเพิ่มขึ้นอย่างมากมายในระบบที่ใหญ่ขึ้นเพื่อที่จะลดความสับสนในสมการลอจิกที่ซับซ้อนมักจะใช้ลอจิกไดอะแกรม และตารางความจริง รูปที่ 2.2 แสดงแบบแผนของโลจิกที่นำมาใช้เก็บโลจิก PAL แบบง่าย ๆ ต่อการเข้าใจและใช้งาน เครื่องหมาย "x" ในรูปแสดงถึงจุดที่ถูกต่อโดยฟิวส์ สัญลักษณ์ในรูป มักใช้ในขบวนการผลิตไอซีอย่างไม่เป็นทางการ เพราะค่อนข้างจะดูได้ง่าย และชัดเจนระหว่างโครงสร้างของชิป และลอจิกไดอะแกรม มันยังเป็นการรวบรวมระหว่างลอจิกไดอะแกรมและตารางความจริง ซึ่งสามารถอ่านเข้าใจได้ง่าย และกะทัดรัด จากรูปที่ 2.2 ที่มีอินพุท 2 อินพุท และมีเอาท์พุท 1 เอาท์พุทสามารถเขียนใหม่ได้ดังรูปที่ 2.3



รูปที่ 2.2 แสดงการเก็บลอจิกในไอซี PAL

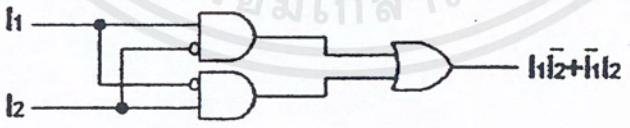


รูปที่ 2.3 แสดงโครงสร้างไอซี PAL 2 อินพุท 1 เอาท์พุท

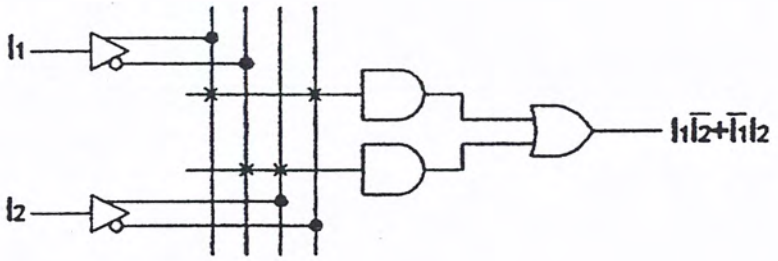
จากตัวอย่างเป็นลอจิกฟังก์ชั่นอย่างง่ายพิจารณาทรานเฟอร์ฟังก์ชั่นจะได้

$$\text{Output} = I1\bar{I}2 + \bar{I}1I2$$

โดยปกติการรวมกันของลอจิกไดอะแกรม สำหรับฟังก์ชั่นนี้คือ รูปที่ 2.4 ซึ่งแสดงวงจรสมมูลทางลอจิกแสดงดังรูปที่ 2.5



รูปที่ 2.4 แสดงวงจรสมมูล

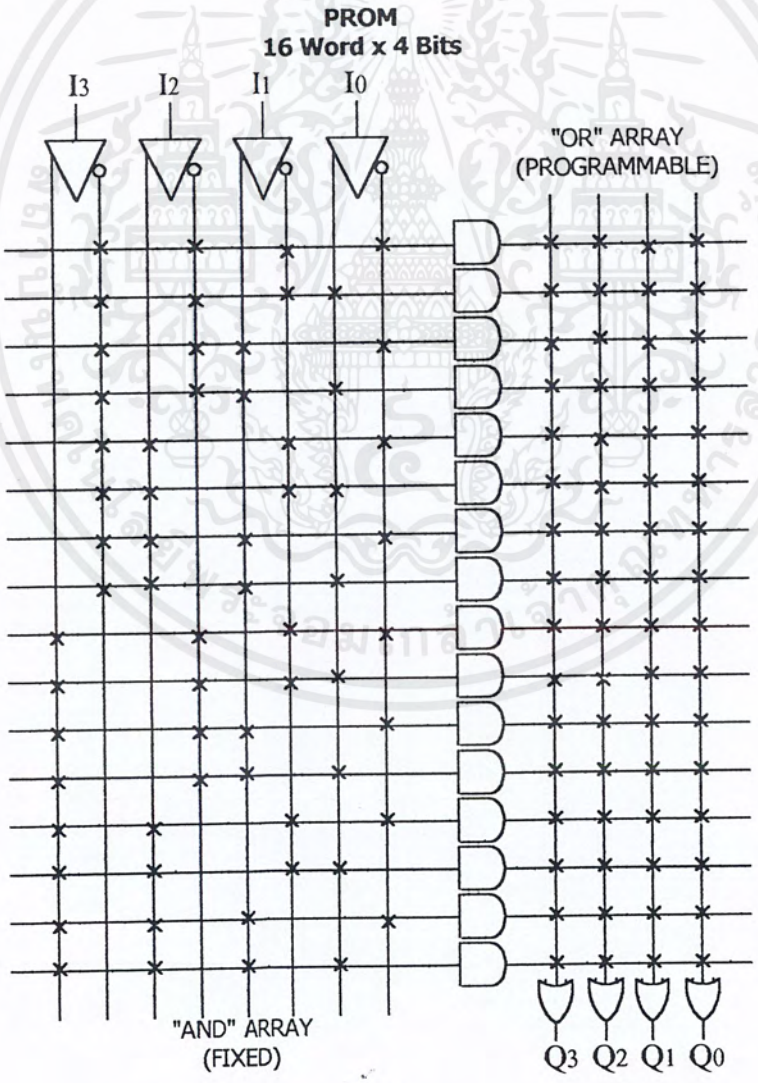


รูปที่ 2.5 แสดงเกทที่อยู่ในไอซี PAL

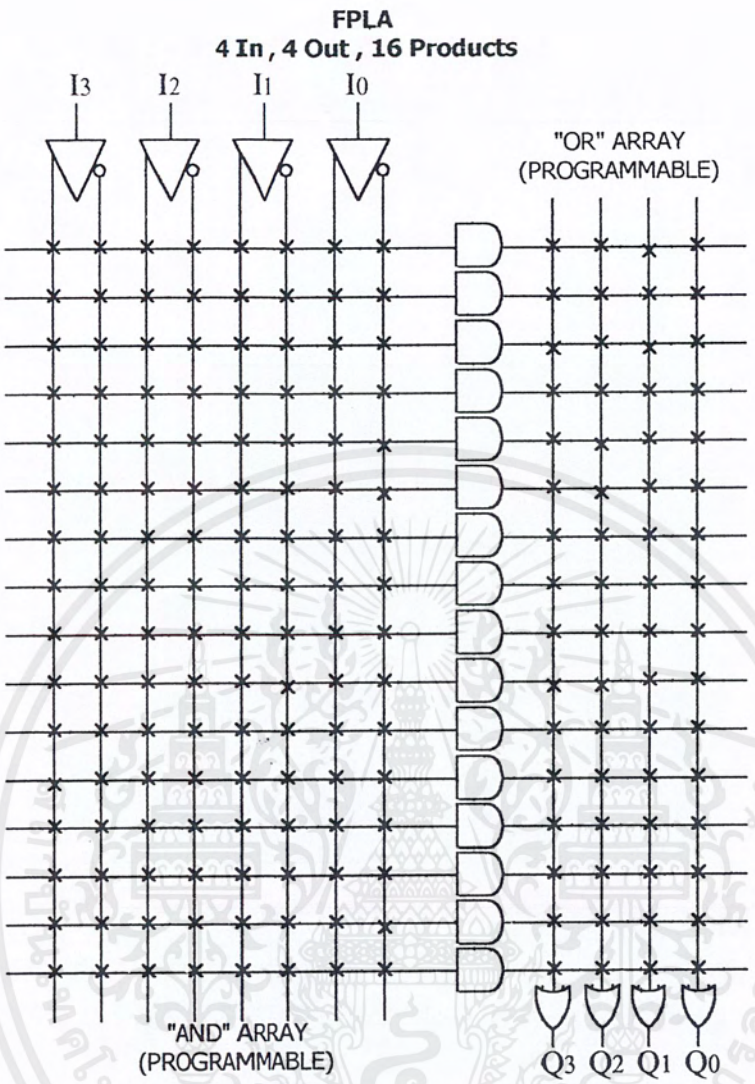
การใช้การรรวมกันของลอจิกในปัจจุบันเป็นไปได้ที่จะเปรียบเทียบ โครงสร้างของ PAL กับโครงสร้างของหลาย ๆ ตระกูลของ PROM กับ PLA ซึ่งโครงสร้างพื้นฐานทางลอจิกของ PROM ประกอบไปด้วยขบวนการ AND ซึ่งเอาท์พุทจะถูกป้อนเข้าสู่ระบบของขบวนการ OR ดังรูปที่ 2.6 PROM ราคาต่ำ, โปรแกรมง่าย และเหมาะสำหรับใช้เปลี่ยนขนาดและการจัดระบบ โดยธรรมดาส่วนมากนิยมใช้เก็บโปรแกรมคอมพิวเตอร์และข้อมูล ในการประยุกต์ใช้งานนี้ได้ระบุอินพุทตามตำแหน่งหน่วยความจำของคอมพิวเตอร์ เอาท์พุทคือตัวบรรจุตำแหน่งหน่วยความจำ โครงสร้างพื้นฐานทางลอจิกของ PLA ประกอบด้วยการโปรแกรมโดยขบวนการ AND ซึ่งเอาท์พุทได้จากการป้อนข้อมูลผ่านขบวนการ OR ดังรูปที่ 2.7 ตั้งแต่ที่นักออกแบบสามารถควบคุมทั้งอินพุทและเอาท์พุทได้อย่างสมบูรณ์ PLA ไม่สามารถที่จะทำการโปรแกรมได้อีก ถูกออกแบบให้ใช้อย่างกว้างขวางในการประยุกต์ใช้งาน อย่างไรก็ดีหลักการสร้าง PLA โดยทั่ว ๆ ไปจะแพง ค่อนข้างยากต่อการเข้าใจ และค่าโปรแกรมแพง โครงสร้างพื้นฐานทางลอจิกของ PAL ประกอบด้วยการโปรแกรมโดยขบวนการ AND ซึ่งเอาท์พุทถูกระบุไว้อย่างแน่นอนด้วยขบวนการ OR ดังรูปที่ 2.8 PAL ได้รวมเอาข้อดีของ PLA ด้วยราคาที่ต่ำและการโปรแกรมง่ายของ PROM ตารางที่ 2.1 สรุปคุณสมบัติของ PROM, PLAและ PAL

	AND	OR	OUTPUT OPTION
PROM	Fixed	Programmable	TS OC
FPLA	Programmable	Programmable	TS OC Fusible Polarity
FPGA	Programmable	None	TS OC Fusible Polarity
FPLS	Programmable	Programmable	TS Registered Feedback, I / O
PAL	Programmable	Fixed	TS Registered Feedback, I / O

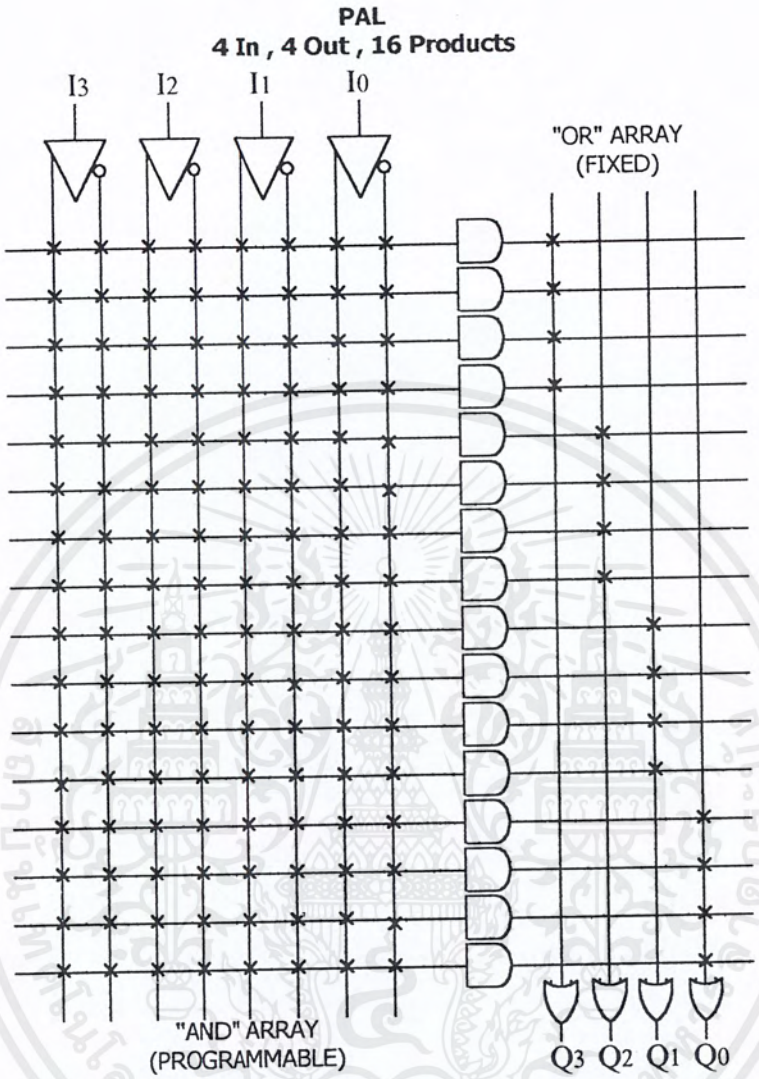
ตารางที่2.1แสดงคุณสมบัติของ PROM, PLA และ PAL



รูปที่ 2.6 แสดงโครงสร้างพื้นฐานทางลอจิกของ PROM



รูปที่ 2.7 แสดงโครงสร้างพื้นฐานทางลอจิกของ PLA

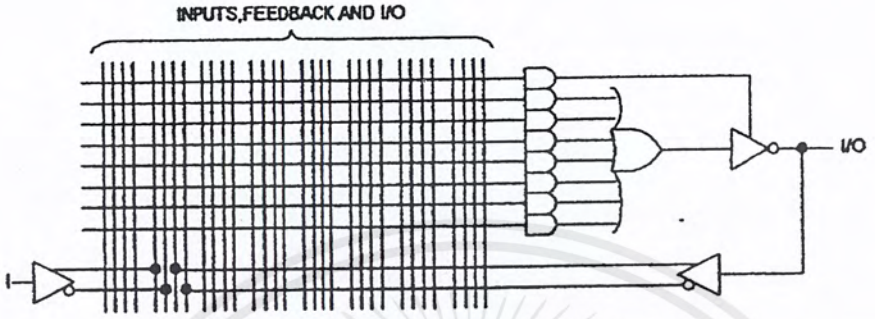


รูปที่ 2.8 แสดงโครงสร้างพื้นฐานทางลอจิกของ PAL

2.1.3 การโปรแกรม อินพุท / เอาท์พุท

ลักษณะของสมาชิกที่สุดยอดของตระกูล PAL คือการโปรแกรม อินพุท / เอาท์พุท นี่คือการยอมรับเงื่อนไขการผลิตซึ่งควบคุมเอาท์พุทโดยตรงของ PAL ดังแสดงในรูปที่ 2.9 เงื่อนไขการผลิตอันดับแรกคือกำหนดให้ตัวกั้น หรือ Buffer ทำการได้ 3 สภาวะ ซึ่งในการบังคับสวิตซ์เลื่อนรวมเข้ากับเอาท์พุท PIN เอาท์พุทจะย้อนกลับไปยังกระบวนการทางอินพุทของ PAL ดังนั้น PAL จะขับอินพุท / เอาท์พุท PIN เมื่อสวิตซ์ 3 สภาวะให้อยู่ในสภาวะที่สามารถต่อเชื่อมได้, อินพุท/เอาท์พุท PIN คือ อินพุทของขบวนการของ PAL และเมื่อสวิตซ์ 3 สภาวะให้อยู่ในสภาวะที่ไม่สามารถที่จะต่อเชื่อมได้ ลักษณะนี้สามารถใช้ PIN สำหรับจัดเตรียมเนื้อที่สำหรับ

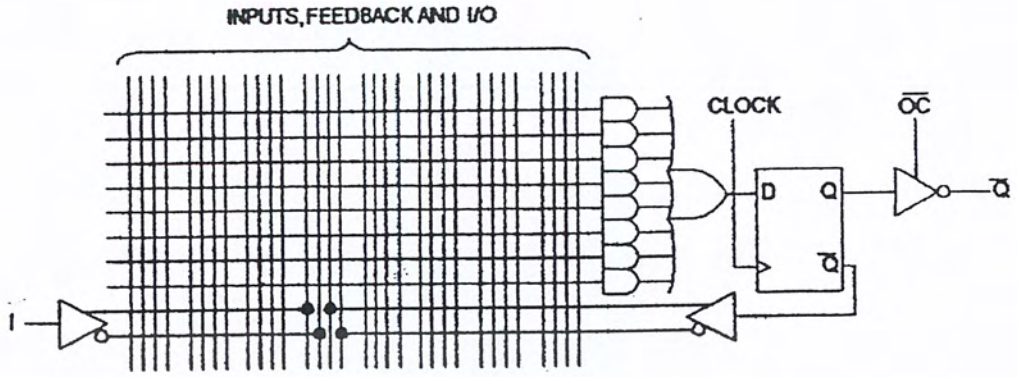
ใช้ในการประมวลผลของฟังก์ชั่นอินพุท / เอาท์พุท หรือกำหนดเอาท์พุท PIN ที่สามารถติดต่อโดยตรงได้ 2 ทาง สำหรับทำงาน เช่น การเลื่อนหรือการหมุน ข้อความแบบอนุกรม



รูปที่ 2.9 แสดงการกำหนดอินพุทเอาท์พุท

2.1.4 หน่วยความจำย่อยทางเอาท์พุทกับการป้อนกลับ

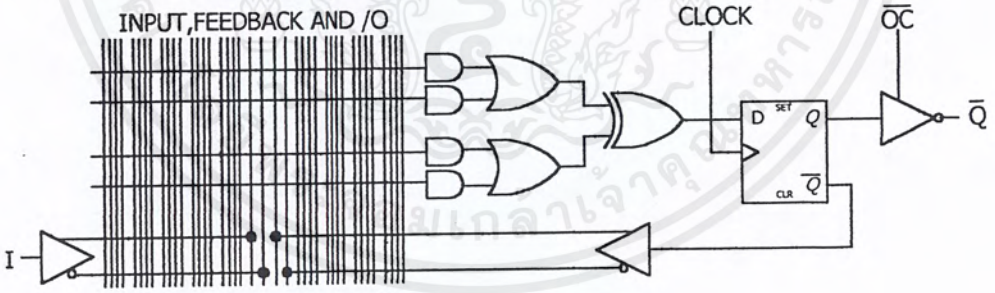
ลักษณะอื่นของความสุดยอดของตระกูล PAL คือข้อมูลในหน่วยความจำย่อยที่เอาท์พุทกับการป้อนกลับของหน่วยความจำย่อย แต่ละเงื่อนไขการผลิตจะเก็บค่าไว้ในฟลิปฟลอบชนิด D ในช่วงขอบขาขึ้นของระบบนาฬิกา ดังรูปที่ 2.10 เอาท์พุท Q ของฟลิปฟลอบผ่านสวิตซ์ไปยังเอาท์พุท PIN โดยสามารถเชื่อมต่อโดยการทำงานเมื่อเป็น Low ของตัวกั้นหรือ Buffer 3 สภาวะรวมไปถึงความเหมาะสมในการส่งข้อมูล เอาท์พุท Q จะถูกส่งกลับไปยังขบวนการของทางอินพุทของ PAL การป้อนกลับนี้ทำให้ PAL จดจำสภาวะก่อนหน้า และยังสามารถดัดแปลงฟังก์ชั่นพื้นฐานบนสภาวะนั้นได้ การยินยอมดังกล่าวทำให้นักออกแบบสามารถเลือกใช้อุปกรณ์ต่าง ๆ เพื่อสนับสนุนการทำงานของระบบคอมพิวเตอร์ ด้วยการเรียงลำดับสภาวะ ซึ่งสามารถโปรแกรมให้ได้ผลสำเร็จ เช่น การกระทำพื้นฐาน, การนับขึ้น, การนับข้าม, การเลื่อน และการแยกไปทำงานตามที่โปรแกรมกำหนดไว้ การกระทำดังกล่าวสามารถทำให้เป็นผลสำเร็จได้โดยหน่วยความจำย่อยของ PAL ด้วยอัตราสูงขึ้นถึง 20 MHz



รูปที่ 2.9 แสดงหน่วยความจำกับการป้อนกลับ

2.1.5 ขบวนการ OR แบบพิเศษของ PAL

XOR PAL ชนิดนี้มีลักษณะการกระทำการ OR แบบพิเศษ เป็นการรวมของผลคูณเป็นส่วนหนึ่งในการรวมกัน 2 ครั้ง ซึ่ง XOR ที่อินพุทของฟลิปฟลอบชนิด D ดังรูปที่ 2.11 ลักษณะทั้งหมดของหน่วยความจำของ PAL คือการรวมใน XOR PAL การกระทำ XOR กำหนดอุปกรณ์ง่าย ๆ ซึ่งการทำงานใช้ในการนับและการจัดเรียงลำดับอื่น ๆ



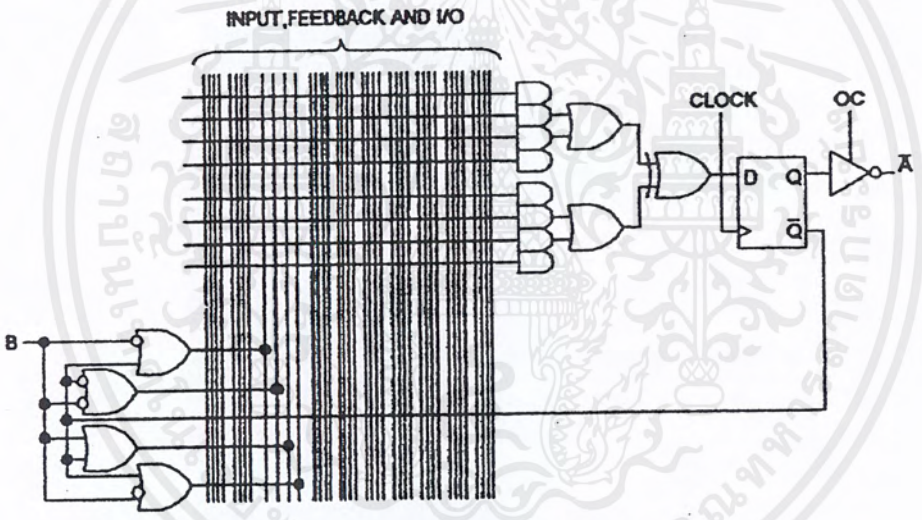
รูปที่ 2.11 แสดงการ OR แบบพิเศษ

2.1.6 การป้อนกลับเกี่ยวกับ เรขาคณิต ของวงจรตรรก

การกระทำทางคณิตศาสตร์ (+,-,>,<) สามารถทำให้เกิดผลโดยการรวมกันของการป้อนกลับของวงจรตรรกในลักษณะของ XOR PAL ที่อินพุทของฟลิปฟลอบชนิด D ยอมผลักดันให้

การทำงานในสภาวะก่อนของ XOR ด้วยการรวมการเปลี่ยนแปลง 2 ส่วน ผลิตโดยกระบวนการของ PAL เอาท์พุท Q ของฟลิปฟลอบที่ป้อนกลับไปยังเกทพื้นฐานโดยอินพุทของเงื่อนไข A ดังรูปที่ 2.12 วงจรเกทพื้นฐานนี้ได้จากกรรมวิธีการลดรูปโดยอาศัยกรรมวิธีการลดรูปแบบคาร์นอร์ (Marnaugh Map) ดังรูปที่ 2.13 รูปที่ 2.14 แสดงขบวนการของ PAL ที่สามารถโปรแกรมได้ 16 วิธีการปฏิบัติการ การกำหนดลักษณะนี้เพื่อประโยชน์ในการทำงานหลาย ๆ อย่างของการกระทำ 2 ตัวแปร และทำให้สะดวกต่อการเปรียบเทียบที่จำเป็นเพื่อให้การทำงานทางคณิตศาสตร์เร็วขึ้น

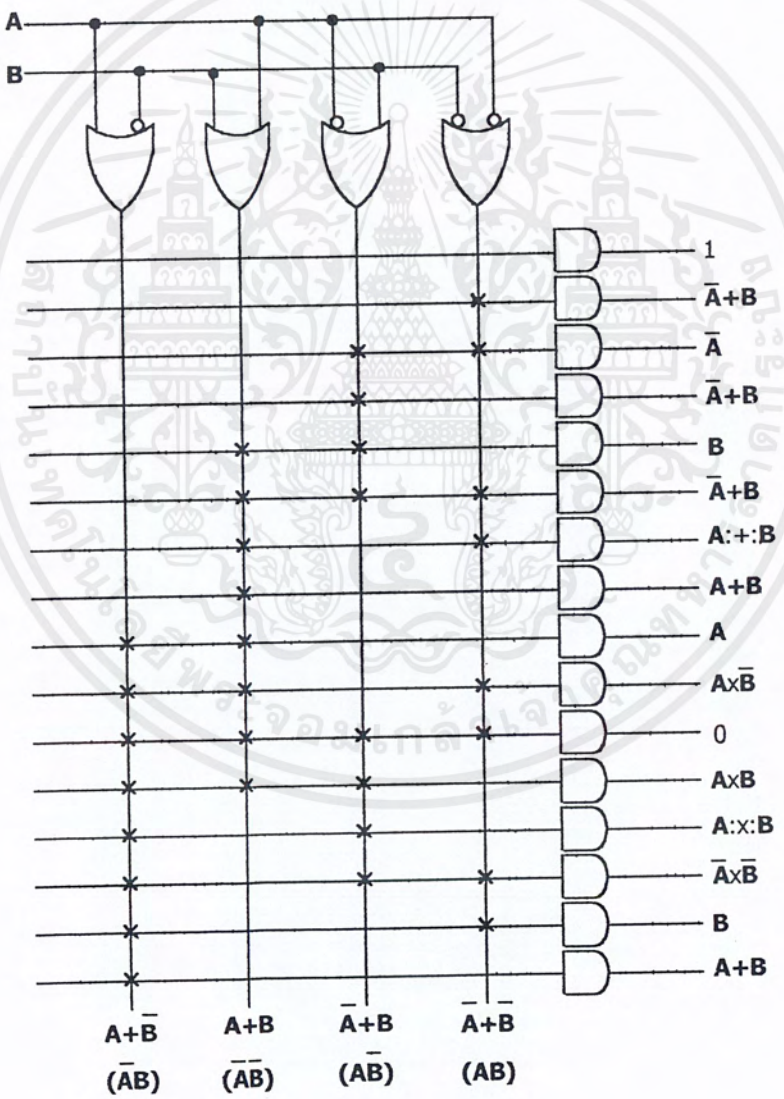
มันทำให้เราเข้าใจอย่างชัดเจนว่า PAL สามารถแทนที่ระบบ SSI หรือ Small-Scale Intergrated ที่ใช้กันอยู่ทุกวันนี้ ดังนั้นต้นทุนการผลิตซึ่งต่ำลงและให้ความคล่องตัวสูงขึ้นในการทำงานของเครื่องมือทางลอจิก



รูปที่ 2.12 แสดงการป้อนกลับเกี่ยวกับเลขคณิต

(Ā+B),(Ā+B̄) / (A+B̄),(A+B)	— —	— X	X X	X —
— —	1	Ā+B̄	Ā	Ā+B
— X	A+B	A:+:B	Ā•B	B
X X	A	A•B̄	0	A•B
X —	A+B̄	B	Ā•B̄	A:•:B

รูปที่ 2.13 แสดงการลดรูปแบบ Karnaugh Map



รูปที่ 2.14 แสดงการโปรแกรม PAL 16วิธี

สมาชิกของตระกูล PAL และคุณสมบัติสามารถสรุปได้ดังตารางที่ 2.2 ซึ่งถูกออกแบบให้ครอบคลุมการกระทำของลอจิก เพื่อลดราคา และขนาดโปรแกรมสำเร็จรูป นี่คือการยอมรับของการยอมรับของนักออกแบบว่า PAL เหมาะสมที่สุดที่จะนำมาประยุกต์ใช้งาน PAL จึงเข้ามาเป็นองค์ประกอบพื้นฐานต่อไป

ตารางที่ 2.2 แสดงสมาชิกและคุณสมบัติของไอซีตระกูล PAL

PART NUMBER	I/P	I/O	PROGRAMMABLE I/O'S	FEEDBACK REGISTER	OUTPUT POLARITY	FUNCTION	PERFORMANCE STD	A	-2	-3
PAL10H8	10	8			AND-OR	AND-OR Gate Array	x			
PAL12H6	12	6			AND-OR	AND-OR Gate Array	x			
PAL14H4	14	4			AND-OR	AND-OR Gate Array	x			
PAL16H2	16	2			AND-OR	AND-OR Gate Array	x			
PAL16C1	16	2			BOTH[1]	AND-OR Gate Array	x			
PAL20C1	20	2			BOTH[1]	AND-OR Gate Array	x			
PAL10L8	10	8			AND-NOR	AND-OR Invert Gate Array	x			
PAL12L6	12	6			AND-NOR	AND-OR Invert Gate Array	x			
PAL14L4	14	4			AND-NOR	AND-OR Invert Gate Array	x			
PAL16L2	16	2			AND-NOR	AND-OR Invert Gate Array	x			
PAL12L10	12	10			AND-NOR	AND-OR Invert Gate Array	x			
PAL14L8	14	8			AND-NOR	AND-OR Invert Gate Array	x			
PAL16L6	16	6			AND-NOR	AND-OR Invert Gate Array	x			
PAL18L4	18	4			AND-NOR	AND-OR Invert Gate Array	x			
PAL20L2	20	2			AND-NOR	AND-OR Invert Gate Array	x			
PAL16L8	10	2	6		AND-NOR	AND-OR Invert Gate Array	x	x	x	x
PAL20L10	12	2	8		AND-NOR	AND-OR Invert Gate Array	x			
PAL16R8	8	8		8	AND-NOR	AND-OR Invert Gate Array w/Reg's	x	x	x	X
PAL16R6	8	6	2	6	AND-NOR	AND-OR Invert Gate Array w/Reg's	x	x	x	X
PAL16R4	8	4	4	4	AND-NOR	AND-OR Invert Gate Array w/Reg's	x	x	x	X
PAL20x10	10	10		10	AND-NOR	AND-OR Invert Gate Array w/Reg's	x			
PAL20x8	10	8	2	8	AND-NOR	AND-OR Invert Gate Array w/Reg's	x			
PAL20x4	10	4	4	4	AND-NOR	AND-OR Invert Gate Array w/Reg's	x			
PAL16x4	8	4	4	4	AND-NOR	AND-OR Invert Gate Array w/Reg's	x			
PAL16A4	8	4	4	4	AND-NOR	AND-CARRY-OR-XOR Invert Gate Array w/Reg's	x			

[1] Simultaneous AND-OR and AND-NOR output

2.1.7 การแบ่งแต่ละส่วนของเบอร์ PAL

แต่ละส่วนของเบอร์ PAL จะถูกระบุในส่วนที่เป็นรหัสของเบอร์ ซึ่งจะแสดงความหมายของการทำงานลอจิก ระบบของรหัสของ PAL ถูกแสดงดังรูปที่ ? ตัวอย่างเช่น PAL เบอร์ PAL14LACN จะมี 14 อินพุท, ทำงานเมื่อเป็น Low, บอกอัตราอุณหภูมิ, ขนาด 20 ขา, ทำจากพลาสติก

ตัวอย่าง PAL เบอร์ PAL14L4-2CJ883BPO1234

PAL บอกกระบวนการหลักว่าเป็นโลจิกตระกูลใด

14 บอกกระบวนการทางอินพุท

L ชนิดของเอาท์พุท

L= ทำงานที่สภาวะ

C= ส่วนแระกอบให้สมบูรณ์

X=ตัวที่เก็บข้อมูลแบบ OR แบบพิเศษ

A= ตัวเก็บข้อมูลแบบคณิตศาสตร์

4 จำนวนของเอาท์พุท

-2 ความเร็ว/กำลัง

A=ความเร็วสูง

-2=ให้กำลังครึ่งหนึ่ง

-4=ให้กำลัง 1/4

C อัตราของอุณหภูมิ

C= 0 ถึง 75 องศา

M= -55 ถึง 125 องศา

J รูปแบบของตัวไอซี

N= ชนิดพลาสติก

J= ชนิดเซรามิค

F= ชนิดที่มีลักษณะแบน

883B กระบวนการ Hi-Rel ที่มีให้เลือก

883B MIL-STD-883

METHOD 5004 & 5005 LEVEL B

883C MIL-STD-883

METHOD 5004 & 5005 LEVEL C

B= MIL-STD-883

METHOD 5004 EQUIVALENT

C= MIL-STD-883

METHOD 5004 EQUIVALENT

PO1234 จำนวนรูปแบบของบิท

2.1.8 ข้อได้เปรียบของการใช้ PAL

PAL มีลักษณะเฉพาะอย่างในโลกของการออกแบบลอจิก ไม่เท่านั้นมันยังมีข้อได้เปรียบมากมาย เหนือกว่าการใช้ลอจิกโดยทั่วไป ด้วยลักษณะมากมายที่ไม่สามารถพบได้อีกในตระกูลของ PAL

-ใช้การโปรแกรมแทนที่ลอจิกของ TTL ทั่ว ๆ ไป

-ลดความสำคัญของไอซีและง่ายต่อการควบคุม

-ลดไอซีของวงจรนับให้น้อยที่สุดคือ 4 ถึง 1

-ความสะดวกและปราศจากสิ่งขัดขวางของเครื่องต้นแบบและโครงการ

-ขนาดของรูปร่างไอซีจะบางและมีขนาด 24 ขา และ 20 ขา

-ความเร็วสูง, ขนาดช่วงหน่วงเวลาที่ใช้กันแพร่หลายคือขนาด 20 nS

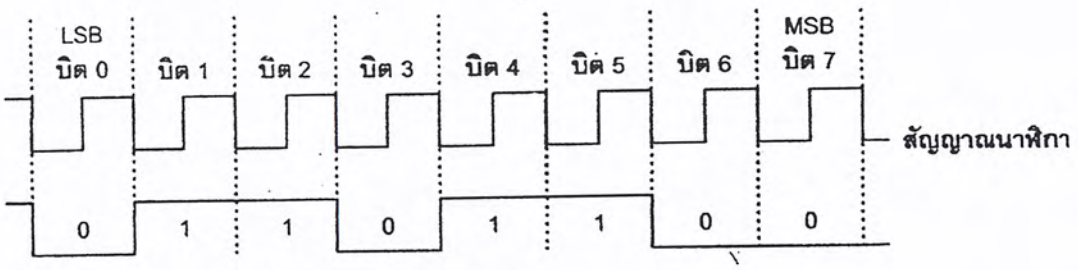
-ขนาดมาตรฐานของ PROM

-โปรแกรมเอาท์พุท 3 สถานะ

ลักษณะทั้งหมดเหล่านี้รวมกันอยู่ในการพัฒนาการผลิตให้ต่ำลง และเพิ่มผลกระทบทางราคาของการผลิต ซึ่งเป็นวิธีการประหยัดเงินของ PAL

2.2 พอร์ตอนุกรม

การสื่อสารแบบอนุกรมนั้นจะแบ่งออกได้เป็น 2 แบบคือการสื่อสารอนุกรมแบบซิงโครนัสและการสื่อสารอนุกรมแบบอะซิงโครนัสการสื่อสารแบบซิงโครนัสจะมีสัญญาณนาฬิการ่วมอยู่กับการรับและส่งสัญญาณด้วยตัวอย่างการส่งข้อมูลแบบซิงโครนัสก็คือคีย์บอร์ดของคอมพิวเตอร์ซึ่งสายเส้นหนึ่งจะเป็นสายของสัญญาณนาฬิกาส่วนสายอีกเส้นจะเป็นสายของข้อมูล ดังนั้นการติดต่อกันแบบซิงโครนัสนี้จะต้องใช้สายในการเชื่อมต่ออย่างน้อยที่สุด3 เส้นคือ สัญญาณนาฬิกา, ข้อมูล และกราวด์ รูปที่ 2.15 แสดงให้เห็นถึงไทมิ่งไดอะแกรมของการส่งข้อมูลแบบซิงโครนัส



รูปที่ 2.15 แสดงรูปแบบข้อมูลอนุกรม

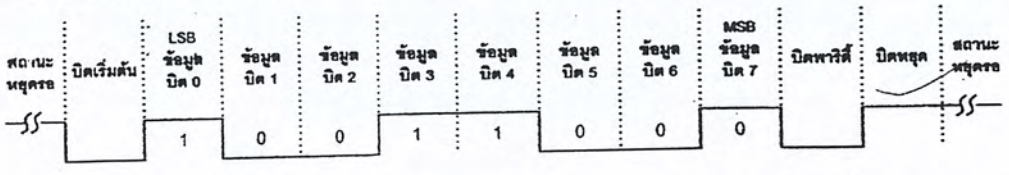
2.2.1 การสื่อสารข้อมูลแบบอะซิงโครนัส

การสื่อสารข้อมูลแบบอะซิงโครนัสคือการรับส่งข้อมูลไปในสายโดยไม่จำเป็นต้องมีสัญญาณนาฬิการ่วมด้วยเหมือนกับการรับส่งข้อมูลแบบซิงโครนัส แต่จะใช้การกำหนดค่าสัญญาณนาฬิกาทั้งภาครับและภาคส่งให้มีค่าเท่ากัน ซึ่งเรียกสัญญาณนาฬิกาที่ใช้ในการกำหนดค่าให้ภาครับและภาคส่งนี้ว่า อัตราการถ่ายทอดข้อมูล หรือ บอดเรต (baudrate) มีหน่วยเป็น บิตต่อวินาที (bit per secone : bps)

รูปแบบของข้อมูลที่ใช้ในการรับส่งแบบอะซิงโครนัสประกอบด้วย 4 ส่วนด้วยกันคือ

1. บิตเริ่มต้น (Start Bit) ซึ่งจะมีขนาด 1 บิต
2. บิตข้อมูลแบบอนุกรมจะมีขนาด 5, 6, 7 หรือ 8 บิต
3. บิตตรวจสอบพาริตี้ (Parity Bit) จะมีขนาด 1 บิตหรือไม่มี
4. บิตปิดท้าย (Stop Bit) จะมีขนาด 1,1.5 หรือ 2 บิต

รูปที่ 2.16 แสดงรูปแบบของข้อมูลอนุกรมแบบอะซิงโครนัส ซึ่งเมื่อไม่มีข้อมูลที่จะส่งขา DATAจะมีสภานะลอจิก "1" ซึ่งเรียกสถานะนี้ว่าสถานะหยุดรอ (waiting stage)จะเริ่มจากการให้ขา DATA มีลอจิก "0" ด้วยช่วงระยะเวลา 1 บิต ซึ่งจะเรียกบิตนี้ว่าบิตเริ่มต้น จากนั้นบิตข้อมูลจะถูกส่งออกไป โดยเริ่มจากบิตที่มีนัยสำคัญต่ำสุด (LBS) ก่อนซึ่งข้อมูลในไบต์ที่จะส่งอาจจะมีจำนวนบิต 5,6,7 หรือ 8 บิตก็ได้ จากนั้นจะตามด้วยบิตพาริตี้ ซึ่งใช้เพื่อตรวจสอบความผิดพลาดที่เกิดขึ้นจากการส่งข้อมูล บิตสุดท้ายที่จะส่งคือบิตปิดท้าย ซึ่งจะให้ขาดดาต้ามีสถานะลอจิก 1 อีกครั้งด้วยระยะเวลาอย่างน้อย 1 บิต, 1.5 บิต หรือ 2 บิต เพื่อเป็นการแสดงว่าสิ้นสุดข้อมูลแล้ว



รูปที่ 2.16 แสดงรูปแบบข้อมูลอนุกรมแบบอะซิงโครนัส

อุปกรณ์พิเศษที่ได้รับการออกแบบมาสำหรับการรับและส่งข้อมูลแบบอะซิงโครนัสเรียกว่า Universal Asynchronous Receiver/Transmitter หรือ UART อัตราความเร็วในการรับและส่งข้อมูลของการรับส่งข้อมูลแบบอะซิงโครนัสคือ ค่าบอดเรต ซึ่งก็คือค่าจำนวนบิตต่อวินาทีที่ใช้ในการรับและส่งข้อมูล บอดเรตมาตรฐานที่ใช้สำหรับพอร์ตอนุกรม RS-232 ได้แก่ 110,150,300,600,1200,2400,4800,9600 และ 19200 บิตต่อวินาที และมีค่าเพิ่มมากขึ้นตามเทคโนโลยีของคอมพิวเตอร์ซึ่งการรับส่งแบบอนุกรมโดยไม่ผ่านโมเด็มอาจจะสามารถกำหนดค่าบอดเรตได้สูงถึง 115200 บิตต่อวินาที เนื่องจากบอดเรตคือจำนวนบิตของข้อมูลที่สามารถถ่ายทอดได้ภายใน 1 วินาที ยกตัวอย่าง ข้อมูลอนุกรมถูกส่งในลักษณะ 8 บิต ไม่มีการตรวจสอบพาริตี้ มีบิตเริ่มต้น 1 บิต และบิตปิดท้าย 1 บิต ความยาวของข้อมูลที่รับส่งนี้เท่ากับ 10 บิต ถ้าใช้บอดเรตในการส่งข้อมูลเท่ากับ 9600 บิตต่อวินาที ก็จะสามารถรับส่งข้อมูลได้ด้วยความเร็ว 960 ไบต์ต่อวินาที และถ้ามีการใช้พาริตี้ความเร็วในการรับส่งข้อมูลจะเหลือเป็น 872 ไบต์ต่อวินาที

การตรวจสอบพาริตี้สามารถกำหนดให้เป็นแบบคี่ (odd), แบบคู่ (even) หรือไม่มีการตรวจสอบพาริตี้ก็ได้ การตรวจสอบพาริตี้เป็นการตรวจสอบจำนวนรวมของบิตที่เป็นลอจิก “1” ภายในข้อมูลที่ส่งไป 1 ไบต์ว่ามีจำนวนรวมเป็นเลขคู่หรือเลขคี่โดยต้องรวมบิตพาริตี้เข้าไปด้วย ยกตัวอย่างข้อมูลที่จะทำการส่งมีขนาด 8 บิตและมีค่าเท่ากับ 99 ฐานสิบหก หรือ 10011001 ฐานสอง จะเห็นว่าข้อมูลในไบต์นี้มีจำนวนลอจิก “1” จำนวน 4 ตัวซึ่งเป็นเลขคู่ ดังนั้นถ้ากำหนดหนดค่าพาริตี้เป็นคู่ค่าในพาริตี้ จะต้องมีลอจิกเป็น “0” แต่ถ้าพาริตี้เป็นคี่ ค่าที่บิตพาริตี้จะต้องเป็น “1” เพื่อให้ข้อมูล 1 ไบต์รวมทั้งบิตพาริตี้มีจำนวนบิตที่เป็นลอจิก “1” มีจำนวนรวมกันเป็นเลขคี่ ในตารางที่ 2.3 แสดงตัวอย่างของพาริตี้ในการรับส่งข้อมูลอนุกรม

ข้อมูล	บิตพาริตี้คู่	บิตพาริตี้คี่
00000000	0	1
00000001	1	0
00000010	1	0
00000011	0	1
00000100	1	0
11111110	0	1
11111111	1	0

ตารางที่ 2.3 แสดงบิตพาริตี้ของข้อมูล

บิตพาริตี้ถูกสร้างขึ้นจากภาคส่งข้อมูลของ UART ซึ่งภาครับจะต้องทำการกำหนดคุณสมบัติการตรวจสอบพาริตี้คี่หรือพาริตี้คู่ จากนั้นภาครับของ UART จะทำกรตรวจสอบค่าพาริตี้ที่เกิดขึ้นว่าเป็นคู่หรือเป็นคี่ โดยการนับจำนวนลอจิก "1" ทั้งหมดรวมทั้งบิตพาริตี้ด้วย ถ้ากำหนดพาริตี้ไว้เป็นคู่แต่อ่านค่าตัวเลขในการนับออกมาได้ตัวเลขเป็นคี่ ทางภาครับจะแสดงข้อผิดพลาดออกมาให้ผู้ใช้ทราบ นับเป็นการตรวจสอบความผิดพลาดที่เกิดขึ้นในการถ่ายข้อมูลที่ง่ายที่สุด แต่จะเชื่อถือได้เมื่อมีบิตข้อมูลที่ทำการส่งผิดพลาดเพียงบิตเดียวเท่านั้น ถ้าข้อมูลที่ทำการส่งบิตที่ผิดพลาดมากกว่า 1 บิต การตรวจสอบด้วยวิธีนี้จะไม่ได้ผล สำหรับการตั้งพาริตี้บิตเป็น NONE นั้นทั้งภาครับและภาคส่ง จะไม่มีการตรวจสอบพาริตี้

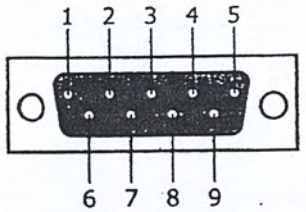
คอมพิวเตอร์ในรุ่น AT เกือบทั้งหมดจะใช้ UART เบอร์ 16450 และ 16550 ส่วนคอมพิวเตอร์รุ่น XT ใช้ UART เบอร์ 8250 UART ชิปเหล่านี้มีระดับแรงดันเป็นแบบทีทีแอล (0 และ +5V) แต่เพื่อให้มีแรงดันเป็นไปตามมารฐาน RS – 232 และเพื่อให้การรับส่งข้อมูลสามารถทำได้ที่ระยะทางไกลมากขึ้นระดับแรงดันทีทีแอลจะถูกแปลงเป็นระดับแรงดันที่สูงขึ้น โดยลอจิก "0" มีระดับแรงดัน +3V ถึง +12V ในขณะที่มีลอจิก "1" มีดับแรงดัน -3V จนถึง-12V

2.2.2 ระดับแรงดันที่ใช้สำหรับพอร์ตอนุกรม RS-232

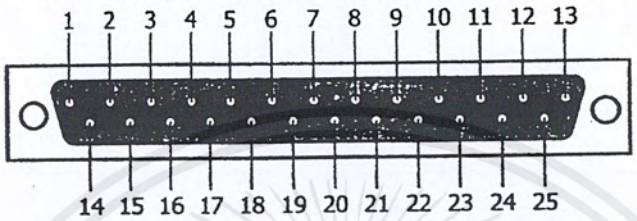
มาตรฐานการสื่อสารข้อมูลของพอร์ตอนุกรม ได้ระบุช่วงระดับแรงดันสำหรับการทำงานของพอร์ตอนุกรมไว้ว่า ที่ลอจิก "0" จะมีระดับสัญญาณ +3V ถึง +15V ส่วนลอจิก "1" จะมีระดับสัญญาณ –3V ถึง -15V ระดับสัญญาณนี้ทำให้ไม่สามารถที่จะนำขาเอาต์พุตใด ๆ ต่อเข้ากับลอจิกเกตเพื่อใช้งานได้โดยตรง จะต้องผ่านวงจรเพื่อเปลี่ยนระดับแรงดันเสียก่อน โดยปกติจะใช้ไอซีจำพวก RS-232 transceiver ที่นิยมมากคือ MAX232 หรือ ICL232 ไอซีกลุ่มนี้จะทำหน้าที่แปลงระดับแรงดันของ RS232 ให้อยู่ในระดับทีทีแอล โดยลอจิก "0" ซึ่งเดิมมีระดับสัญญาณ +3V ถึง +15V จะถูกแปลงเป็น 0V ส่วนลอจิก "1" ซึ่งเดิมมีระดับสัญญาณ –3V ถึง -15V จะแปลงเป็น +5V ทั้งนี้เพื่อให้สามารถเชื่อมต่อกับอุปกรณ์ดิจิตอลอื่นที่ใช้ระดับแรงดันทีทีแอล

2.2.3 คอนเน็กเตอร์สำหรับพอร์ต RS-232 และการเชื่อมต่อ

มาตรฐานการเชื่อมต่อแบบ RS-232 จะใช้คอนเน็กเตอร์แบบ DB-25 ตัวผู้หรือ DB-9 ตัวผู้ซึ่งคอนเน็กเตอร์แบบ DB—25 จะมีขาต่อใช้งานเพียง 9 เส้นเช่นเดียวกับคอนเน็กเตอร์แบบ DB-9 เนื่องจากขาอื่น ๆ ที่เคยใช้งานในอดีต ปัจจุบันมีการใช้งานไม่มากนัก จึงถูกยกเลิกไปโดยแสดงรูปร่างและตำแหน่งขาในรูปที่ 2.17



(ก) คอนเน็กเตอร์อนุกรม 9 ขาหรือแบบ DB-9 (มองจากด้านหลังคอมพิวเตอร์)



(ข) คอนเน็กเตอร์อนุกรม 25 ขาหรือแบบ DB-25 (มองจากด้านหลังคอมพิวเตอร์)

คอนเน็กเตอร์ DB-9	คอนเน็กเตอร์ DB-25	ชื่อของสายสัญญาณ	ชนิดของสายสัญญาณ
1	8	Data Carrier Detect : DCD	อินพุต
2	3	Received Data : RxD	อินพุต
3	2	Transmitted Data : TxD	เอาต์พุต
4	20	Data Terminal Ready : DTR	เอาต์พุต
5	7	Signal Ground : GND	-
6	6	Data Set Ready : DSR	อินพุต
7	4	Request To Send : RTS	เอาต์พุต
8	5	Clear To Send : CTS	อินพุต
9	22	Ring Indicator : RI	อินพุต

รูป 2.17 แสดงการจัดขาของคอนเน็กเตอร์พอร์ตอนุกรมตามมาตรฐาน RS-232

สำหรับรายละเอียดที่การทำงานในแต่ละขาของพอร์ตอนุกรม RS-232 มีดังนี้

Data Carrier Detect (DCD) หรืออาจเรียกว่า Carrier Detect (CD) ขานี้จะแอกทีฟเมื่อมีการส่งสัญญาณพาห์จากอุปกรณ์สื่อสารข้อมูล เช่น โมเด็ม สำหรับการใช้งานปกติ ขานี้จะไม่ได้ถูกใช้งานมากนัก

Receive Data (RD) หรือ (RxD) ขานี้ใช้เพื่อรับสัญญาณอนุกรมเข้ามายังคอมพิวเตอร์ โดยนำข้อมูลที่อ่านได้เก็บไว้ในรีจิสแตอร์ บัฟเฟอร์

Transmitted Data (TD) หรือ (TxD) ขานี้ใช้เพื่อส่งข้อมูลออกจากคอมพิวเตอร์ โดยนำข้อมูลที่เก็บอยู่ในบัฟเฟอร์สำหรับส่งข้อมูลส่งออกไป

Data Terminal Ready (DTR) เป็นขาสัญญาณที่ส่งออกจากคอมพิวเตอร์เพื่อให้อุปกรณ์ปลายทางรับรู้ว่า ต้องการติดต่อด้วย โดยขา DTR นี้จะต้องเชื่อมต่อกับขา DSR ของอุปกรณ์ปลายทาง และขา DTR ของอุปกรณ์ปลายทางจะต้องเชื่อมต่อกับขา DSR ของคอมพิวเตอร์ ถ้าใช้การเชื่อมต่อเป็นแบบ Null Modem ซึ่งใช้สายในการเชื่อมต่อเพียง 3 เส้น จะต้องต่อขา DTR และ DSR ของตัวมันเองเข้าด้วยกันและต้องต่อกับขา DCD ด้วยในกรณีที่โปรแกรมสื่อสารที่ใช้มีการตรวจจับสัญญาณพาห์

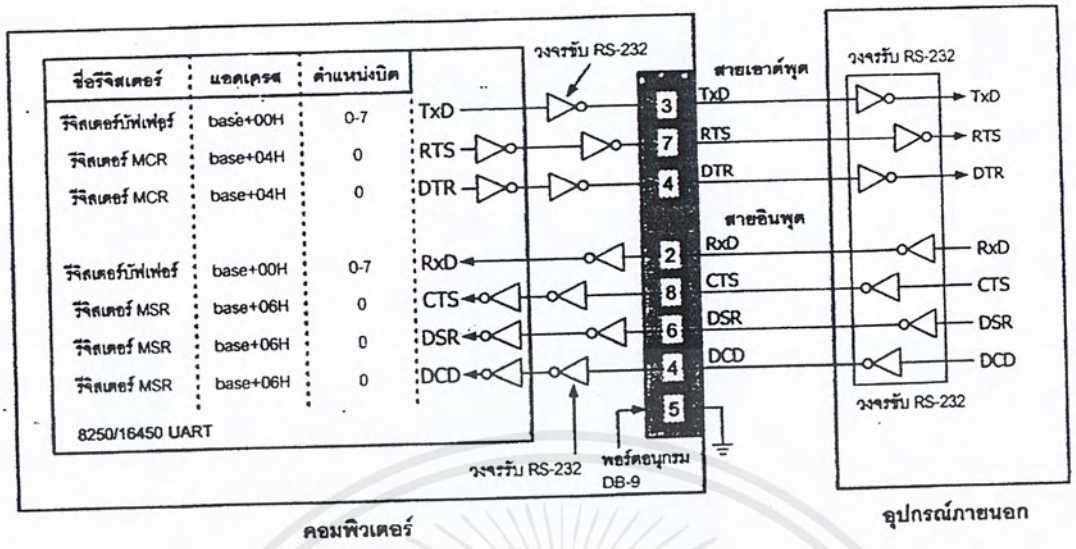
Signal Ground (GND) ขากราวด์ของระบบ

Data Set Ready (DSR) ขานี้จะใช้คู่กับขา DTR เพื่อตรวจสอบการเชื่อมต่อกันระหว่างคอมพิวเตอร์กับอุปกรณ์ปลายทาง ซึ่งขา DSR นี้จะเป็นขาสำหรับรับข้อมูลจากภายนอกซึ่งถูกส่งมาจากขา DTR

Request To Semd (RTS) เป็นขาสำหรับส่งสัญญาณร้องขอให้ทางอุปกรณ์ปลายทางส่งข้อมูลกลับมายังคอมพิวเตอร์ โดยขาที่รับสัญญาณ RTS ก็คือขา CTS ในกรณีที่ใช้การเชื่อมต่อแบบ Null modem 3 สาย จะต้องเชื่อมต่อขา RTS และ CTS ของตัวมันเองเข้าด้วยกัน เพื่อจะให้การรับและส่งข้อมูลสามารถเกิดขึ้นได้ตลอดเวลา

Clear To Send (CTS) ขานี้จะคอยรับสัญญาณจากขา RTS เมื่อรับสัญญาณได้ ข้อมูลที่ขา TxD จะถูกส่งออกไป ดังนั้นขานี้จึงถูกใช้เพื่อตรวจสอบอุปกรณ์ต่อพ่วงว่าพร้อมที่จะรับข้อมูลหรือไม่

Ring Indicator (RI) ใช้แสดงสถานะสัญญาณเรียกจากสายโทรศัพท์ ปกติในการสื่อสารโดยทั่วไปสายนี้จะไม่ถูกใช้งานก็ต่อเมื่อมีการเชื่อมต่อกับโมเด็มและโปรแกรมมีการตรวจสอบสัญญาณนี้เท่านั้น



รูปที่ 2.18 แสดงโครงสร้างทางฮาร์ดแวร์ของพอร์ตอนุกรม

2.2.4 แอดเดรสของพอร์ตอนุกรม

แอดเดรสพื้นฐานของพอร์ตอนุกรมมี 4 ตำแหน่งดังนี้คือ

COM1 : 3F8H

COM2 : 2F8H

COM3 : 3E8H

COM4 : 2E8H

เมื่อเริ่มเปิดแครื่องเพื่อใช้งานคอมพิวเตอร์ ไบออสภายในคอมพิวเตอร์จะทำการตรวจสอบแอดเดรสของพอร์ตอนุกรมทั้งหมด ถ้าไบออสตรวจพบแอดเดรสของพอร์ตอนุกรม ไบออสจะนำแอดเดรสที่ตรวจพบไปเก็บไว้ในหน่วยความจำขนาด 2 ไบต์ สำหรับพอร์ตอนุกรม COM1 จะเก็บไว้ที่แอดเดรส 0000:0400H และ 0000:0401H ส่วนตำแหน่งอื่น ๆ มีรายละเอียดดังนี้

COM2 = 0000:0402H – 0000:0403H

COM3 = 0000:0404H – 0000:0405H

COM4 = 0000:0406H – 0000:0407H

นอกจากนี้ที่หน่วยความจำแอดเดรส0000:0411H ยังใช้สำหรับแสดงจำนวนของพอร์ตอนุกรมที่มีใช้อยู่ในคอมพิวเตอร์อีกด้วยโดยมีรายละเอียดดังแสดงใน

ตารางที่2.4

บิต 3	บิต 2	บิต 1	จำนวนพอร์ต
0	0	0	ไม่มีพอร์อนุกรม
0	0	1	มีพอร์ตอนุกรม 1 พอร์ต
0	1	0	มีพอร์ตอนุกรม 2 พอร์ต
0	1	1	มีพอร์ตอนุกรม 3 พอร์ต
1	0	0	มีพอร์ตอนุกรม 4 พอร์ต

ตารางที่2.4 แสดงข้อมูลในแอดเดรส 0000:0411H ที่ใช้บอกจำนวนพอร์ต

2.2.5 ระดับแรงดันที่ใช้สำหรับพอร์ตอนุกรม RS-232

มาตรฐานการสื่อสารข้อมูลของพอร์ตอนุกรม ได้ระบุช่วงระดับแรงดันสำหรับการทำงานของพอร์ตอนุกรมไว้ว่า ที่ลอจิก "0" จะมีระดับสัญญาณ +3V ถึง +15V ส่วนลอจิก "1" จะมีระดับสัญญาณ –3V ถึง -15V ระดับสัญญาณนี้ทำให้ไม่สามารถที่จะนำขาเอาต์พุตใด ๆ ต่อเข้ากับลอจิกเกตเพื่อใช้งานได้โดยตรง จะต้องผ่านวงจรเพื่อเปลี่ยนระดับแรงดันเสียก่อน โดยปกติจะใช้ไอซีจำพวก RS-232 transceiver ที่นิยมมากคือ MAX232 หรือ ICL232 ไอซีกลุ่มนี้จะทำหน้าที่แปลงระดับแรงดันของ RS232 ให้อยู่ในระดับทีทีแอล โดยลอจิก "0" ซึ่งเดิมมีระดับสัญญาณ +3V ถึง +15V จะถูกแปลงเป็น 0V ส่วนลอจิก "1" ซึ่งเดิมมีระดับสัญญาณ –3V ถึง -15V จะแปลงเป็น +5V ทั้งนี้เพื่อให้สามารถเชื่อมต่อกับอุปกรณ์ดิจิตอลอื่นที่ใช้ระดับแรงดันทีทีแอล

บทที่3
การทำงานของส่วนต่างๆของเครื่องทดสอบไอซี PLD

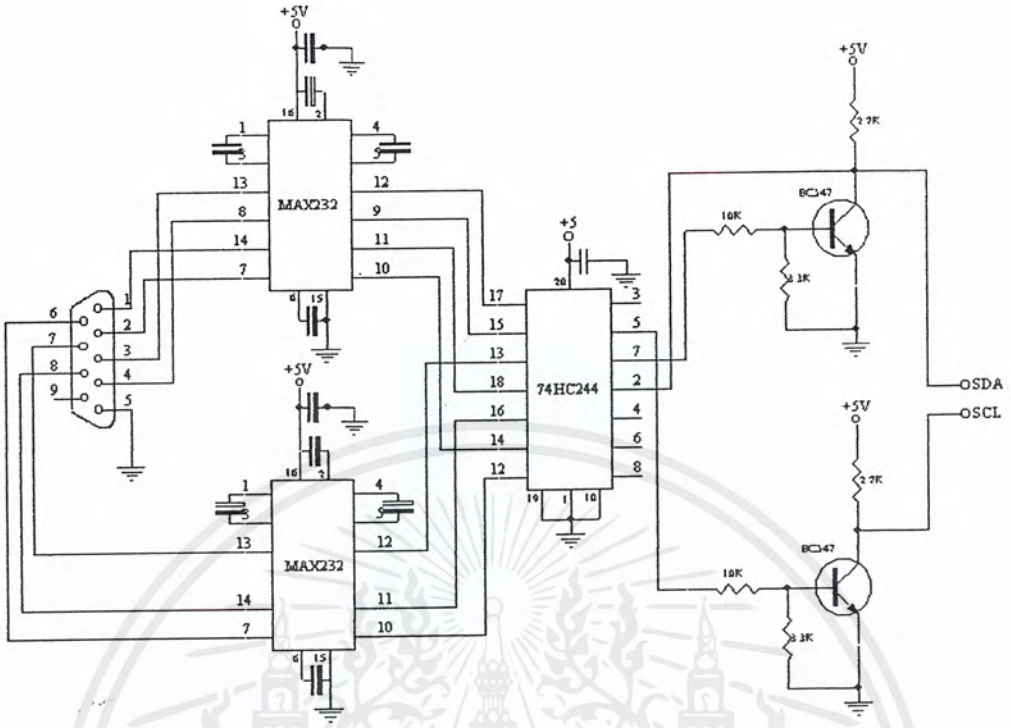
ในส่วนของฮาร์ดแวร์นั้นประกอบด้วยวงจรอยู่ 4 ส่วนคือ

1. การเชื่อมต่อพอร์ตอนุกรม
2. การเชื่อมต่อและขยายพอร์ตอนุกรมกับระบบบัส I^2C
3. การเชื่อมต่อสัญญาณอะนาลอกกับพอร์ตอนุกรมผ่านบัส I^2C
4. บัฟเฟอร์
5. ภาคจ่ายไฟ

3.1 การเชื่อมต่อพอร์ตอนุกรม

เริ่มจากคอมพิวเตอร์เชื่อมต่อฮาร์ดแวร์ผ่านทางคอนเน็กเตอร์แบบ DB-9 สัญญาณจากคอมพิวเตอร์จะมีระดับแรงดันตามมาตรฐาน RS-232 คือมีระดับแรงดัน ± 3 V ถึง ± 12 V เพื่อให้สามารถทำงานเข้ากับวงจรได้ซึ่งมีระดับเป็นแบบทีทีแอลจึงต้องต่อผ่านไอซี MAX232 หรือ ICL232 เพื่อแปลงระดับแรงดันให้อยู่ในระดับทีทีแอลเสียก่อนซึ่งเปรียบเสมือนว่าไอซี MAX232 หรือ ICL232 ทำหน้าที่ปรับระดับแรงนั่นเอง สัญญาณที่ผ่านไอซี MAX232 มาแล้วจะถูกส่งผ่านไปที่ไอซีบัฟเฟอร์เบอร์ 74HC244 ทำหน้าที่ขยายกระแสให้กับขาสัญญาณทั้งหมดและป้องกันความเสียหายที่อาจเกิดความผิดพลาดจากวงจรวงจรแสดงนรูปที่ 3.1จากไอซีบัฟเฟอร์ 74HC244 จะใช้สายสัญญาณ 3 เส้นคือ DTR RTS DCD เพื่อนำมาเข้า ระบบบัส I^2C ' ซึงสายสัญญาณทั้ง 3 เส้นนี้มีความสำคัญในการรับส่งข้อมูลของฮาร์ดแวร์โดยที่ระบบบัส I^2C จะมีขั้นตอนการทำงานดังนี้คือ

ขา SDA ระบบบัส I^2C ทำหน้าที่เป็นขาข้อมูลซึ่งต้องมีทั้งการส่งและรับสัญญาณ จะใช้ขา RTS ในการส่งสัญญาณและรับสัญญาณผ่านทางขา DCD โ ดยทำงานผ่านทรานซิสเตอร์ Q1 ที่ทำหน้าที่เป็นสวิตช์ ในขณะที่ขา SCL เป็นขาสัญญาณนาฬิกาของระบบบัส I^2C โดยจะใช้ขา DTR ส่งสัญญาณผ่านทางทรานซิสเตอร์ Q2 ที่ทำหน้าที่เป็นสวิตช์เช่นกัน



รูปที่3.1 แสดงวงจรการเชื่อมต่อพอร์ตอนุกรม

3.2 การเชื่อมต่อพอร์ตอนุกรมกับระบบบัส I^2C

I^2C ย่อมาจาก Inter-IC Communication หมายถึง การติดต่อสื่อสารระหว่างไอซี โดยบัส I^2C ได้รับการพัฒนาขึ้นโดยฟิลิปส์ (Phillips) ด้วยจุดมุ่งหมายหลักคือ ต้องการให้ไอซีหรือโมดูลสามารถติดต่อ สั่งงาน และควบคุมภายใต้สายสัญญาณเพียง 2 เส้น เส้นหนึ่งคือ สายข้อมูล อีกเส้นหนึ่งคือสายสัญญาณนาฬิกาที่ใช้ในการกำหนดจังหวะการทำงาน การต่อร่วมกันของอุปกรณ์บนบัส I^2C ทำได้ง่ายมาก เพียงต่อสายข้อมูลและสายสัญญาณนาฬิกาของอุปกรณ์แต่ละตัวขนานหรือพ่วงกันไป สาวนการกำหนดแอดเดรสหรือตำแหน่งสำหรับการติดต่ออุปกรณ์แต่ละตัวจะใช้รหัสข้อมูลและการกำหนดสภาวะลอจิกที่ขาแอดเดรสของอุปกรณ์แต่ละตัว

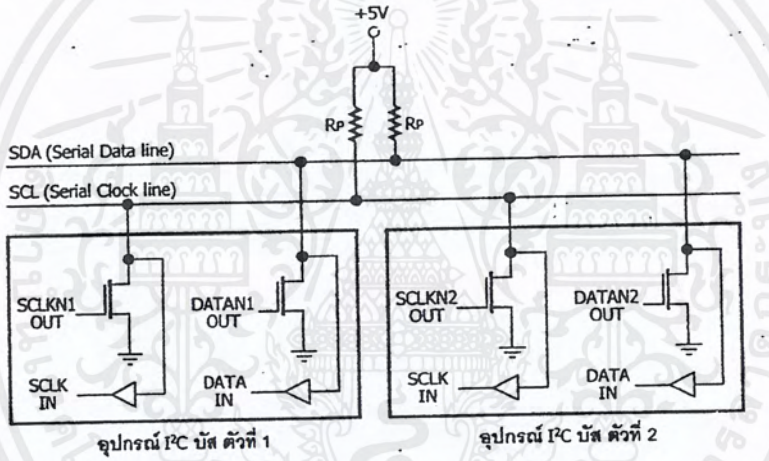
สายข้อมูลบนบัส I^2C มีชื่อเรียกอย่างเป็นทางการว่า สายข้อมูลอนุกรม หรือ SDA (Serial Data line) ส่วนสายสัญญาณนาฬิกามีชื่อเรียกว่า สายสัญญาณนาฬิกาอนุกรม หรือ SCL (Serial Clock line) ในการอธิบายต่อไปนี้จะเรียกสายสัญญาณทั้งสองว่า สาย SDA และ SCL

อุปกรณ์ที่ใช้การเชื่อมต่อด้วยบัส I^2C มีหลากหลาย ไม่ว่าจะเป็นไอซีขยายพอร์ตอินพุตเอ้าต์พุต (I/O Expander), ไอซีแปลงสัญญาณอะนาลอกเป็นดิจิตอล (ADC)และแปลงสัญญาณ

ดิจิตอลเป็นอนาลอก (DAC) , ไอซีรีลไทม์คล็อก (RTC), ไอซีขับโมดูล LCD, หน่วยความจำอีพรอม และไมโครคอนโทรลเลอร์

3.2.1 คุณสมบัติโดยทั่วไปของบัส 1^2C

สาย SDA และ SCL เป็นสายสัญญาณ 2 ทิศทาง (bi-directional line) ต้องมีการต่อตัวต้านทานพูลอัปกับแรงดัน +5V ไว้ตลอดเวลา เพื่อให้สายมีสถานะลอจิกสูงในขณะที่ไม่มีการติดต่อใช้งาน ทั้งยังช่วยในการป้องกันสัญญาณรบกวนที่อาจมีเข้ามาในสายสัญญาณทั้งสองวงจรเอ้าต์พุตของอุปกรณ์ที่ต่ออยู่บนบัส 1^2C ต้องมีลักษณะเป็นวงจรเดรนเปิด (open-drain) หรือคอลเล็กเตอร์เปิด(open-collector) ดังแสดงรายละเอียดในรูป3.2

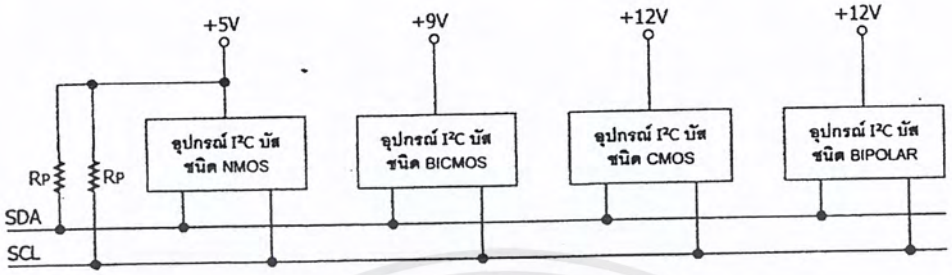


รูปที่ 3.2 แสดงโครงสร้างวงจรเอาท์พุตของอุปกรณ์ที่ใช้ในการเชื่อมต่อบนระบบบัส 1^2C

อัตราการถ่ายทอดข้อมูลบนบัส 1^2C สูงถึง 100 กิโลบิตต่อวินาทีในโหมดปกติ (standard mode) และสูงถึง 400 กิโลบิตต่อวินาทีในโหมดความเร็วสูง (fast mode) อุปกรณ์ที่ต่อร่วมอยู่บนบัส 1^2C จะต้องมีค่าความจุไฟฟ้ารวมที่เกิดขึ้นระหว่างสาย SDA และ SCL ไม่เกิน 400pF การเข้าถึงอุปกรณ์บนบัส 1^2C ใช้ข้อมูลสำหรับการเข้าถึง 2 ค่า คือ 7 บิต (7- bit addressing) หรือ 10 บิต (10-bit addressing)

ข้อเด่นอีกประการหนึ่งของบัส 1^2C คือ สามารถเชื่อมต่ออุปกรณ์ที่ใช้ไฟเลี้ยงไม่เท่ากันให้สามารถติดต่อสื่อสารกันได้โดยอุปกรณ์บนบัส 1^2C ตัวหนึ่งอาจใช้ไฟเลี้ยง +5V ในขณะที่อีกตัวหนึ่งใช้ไฟเลี้ยง +12V การต่อร่วมกันบนบัส 1^2C สามารถกระทำได้ในลักษณะเดียวกันกับกรณีที่

อุปกรณ์ทั้งสองใช้ไฟเลี้ยงเท่ากัน กล่าวคือ ให้ต่อสาย SDA และSCL ของอุปกรณ์แต่ละตัวเข้าด้วยกัน และต้องต่อตัวต้านทานพูลอัป (Rp) เข้ากับแรงดัน +5V ไว้ด้วยเสมอ ดังแสดงในรูปที่ 3.3



รูปที่ 3.3 แสดงการเชื่อมต่ออุปกรณ์บนระบบบัส 1^2C ที่ใช้ไฟเลี้ยงไม่เท่ากัน

3.2.2 หลักการของบัส 1^2C

บัส 1^2C ประกอบด้วยสายสัญญาณ 2 เส้น ดังที่ได้กล่าวมาแล้ว คือ SDA และ SCL อุปกรณ์ที่ต่อพ่วงบนบัสสามารถมีได้มากมาย ดังนั้นจึงต้องมีการกำหนดรูปแบบของการติดต่อบนบัส หรือเรียกว่า โปรโตคอล (protocal) เพื่อให้ผู้ใช้งานทราบว่าขณะนี้อุปกรณ์ใดติดต่อกันอยู่ และอุปกรณ์ตัวใดเป็นตัวรับหรือตัวส่ง ต่อไปนี้จะขออธิบายลักษณะ หน้าที่ และนิยามของอุปกรณ์ที่ต่ออยู่บนบัส 1^2C เพื่อเป็นข้อตกลงพื้นฐานก่อนที่จะอธิบายการทำงานของบัส 1^2C ต่อไป

อุปกรณ์ที่เป็นผู้สร้างข้อมูลหรือส่งข้อมูล เรียกว่าตัวส่ง (transmitter)

อุปกรณ์ที่เป็นผู้รับข้อมูล เรียกว่า ตัวรับ (receiver)

ในอุปกรณ์บนบัส 1^2C สามารถเป็นได้ทั้งตัวรับและตัวส่งบางอุปกรณ์ทำหน้าที่เป็นตัวรับเพียงอย่างเดียว จะไม่มีอุปกรณ์ใดบนบัส 1^2C ที่ทำหน้าที่เป็นตัวส่งเพียงอย่างเดียว

อุปกรณ์ที่ทำหน้าที่ควบคุมจังหวะการทำงานหรือการติดต่อบนบัส 1^2C เรียกว่า มาสเตอร์ (master)

อุปกรณ์ที่ถูกควบคุมหรืออุปกรณ์ที่ต่อพ่วงเข้าไปบนบัส 1^2C เรียกว่า สเลฟ (slave)

ข้อกำหนด 2 ประการสำคัญของการติดตอ่บนบัส 1^2C คือ

1 การถ่ายทอดข้อมูลจะเกิดขึ้นได้เมื่อบัสว่างเท่านั้น

2 ในระหว่างการถ่ายทอดข้อมูล เมื่อใดก็ตามที่สาย SCL มีสถานะเป็นลอจิกสูง สายข้อมูลต้องรักษาข้อมูลไว้ อย่าให้เกิดการเปลี่ยนแปลงขึ้นเด็ดขาด มิฉะนั้น สัญญาณที่เกิดขึ้นจะได้รับการแปลความหมายเป็นสัญญาณควบคุมแทน

3.2.3 สภาวะที่เกิดขึ้นบนบัส I^2C

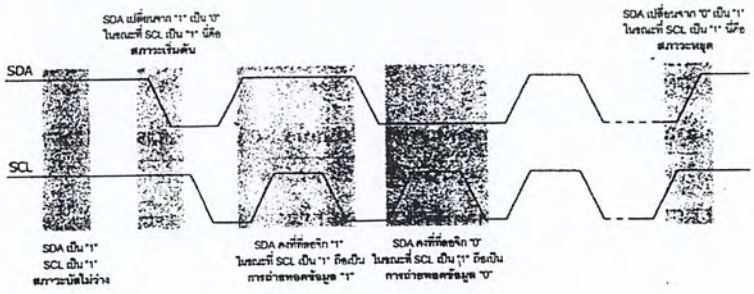
1 บัสว่าง (Bus not busy) สภาวะนี้เกิดขึ้นเมื่อสถานะลอจิกบนสาย SDA และ SCL เป็นลอจิกสูงทั้งคู่ นั่นเมื่อหมายความว่า การถ่ายทอดข้อมูลสามารถเริ่มต้นขึ้นได้

2 เริ่มต้นการถ่ายทอดข้อมูล (strat data transfer) เกิดขึ้นเมื่อสาย SDA มีการเปลี่ยนแปลงระดับลอจิกจากสูงไปต่ำ ในขณะที่สาย SCL มีสถานะลอจิกสูง เรียกสภาวะที่เกิดขึ้นนี้ว่า สภาวะเริ่มต้น (START)

3 ข้อมูลดำรงอยู่บนบัส (data valid) สภาวะนี้เกิดขึ้นถัดจากสภาวะเริ่มต้น โดยสถานะลอจิกที่เกิดขึ้นบนสาย SDA ก็คือข้อมูลที่ทำการถ่ายทอด เมื่อสาย SCL เป็นลอจิกสูง สถานะที่สาย SDA ต้องคงที่ เพื่อให้อุปกรณ์รับรู้ข้อมูลในจังหวะนั้นว่า เป็น "0" หรือ "1" ข้อมูลอาจเกิดการเปลี่ยนแปลงได้ในขณะที่สาย SCL เป็นลอจิกต่ำ แต่เมื่อใดก็ตามที่ต้องการให้เกิดการถ่ายทอดข้อมูลอย่างสมบูรณ์ สถานะลอจิกที่ขา SDA ต้องคงที่ตลอดช่วงเวลาที่สาย SCL มีสถานะลอจิกสูง หากเกิดการเปลี่ยนแปลงสถานะลอจิกในขณะที่สาย SCL มีลอจิกสูงอยู่นั้น อุปกรณ์มาสเตอร์ที่ทำการควบคุมการถ่ายทอดข้อมูลจะแปลความหมายเป็นสภาวะหยุดหรือสภาวะเริ่มต้นก็ได้ ทำให้ข้อมูลที่ทำการถ่ายทอดนั้นเกิดความผิดพลาดขึ้น

4 รับรู้ข้อมูล (acknowledge) เกิดขึ้นหลังจากที่การถ่ายทอดข้อมูลจากตัวส่งมายังตัวรับเกิดขึ้นอย่างสมบูรณ์ โดยตัวส่งจะทำการส่งข้อมูลมา 1 บิต เรียกว่า บิตรับรู้ (acknowledge bit) มีสถานะเป็นลอจิกสูง หลังจากส่งข้อมูลมาครบถ้วน ส่วนอุปกรณ์มาสเตอร์จะทำการส่งสัญญาณรับรู้พิเศษซึ่งสัมพันธ์กับสัญญาณนาฬิกา อุปกรณ์สเลฟที่ถูกอ้างถึงในการติดต่อหรือกำลังติดต่ออยู่ในขณะนั้นก็จะกำเนิดบิตรับรู้ที่มีสถานะลอจิกต่ำเพื่อตอบสนองให้ทราบว่าได้รับข้อมูลเรียบร้อยแล้ว

(5) หยุดการถ่ายทอดข้อมูล (stop data transfer) เกิดขึ้นเมื่อสาย SDA มีการเปลี่ยนแปลงระดับลอจิกจากต่ำไปสูง ในขณะที่สาย SCL มีสถานะลอจิกสูง เรียกสภาวะที่เกิดขึ้นนี้ว่า สภาวะหยุด (STOP)



รูปที่ 3.4 แสดงไดอะแกรมเวลาแสดงสภาวะต่างๆ ที่เกิดขึ้นบนระบบบัส I^2C

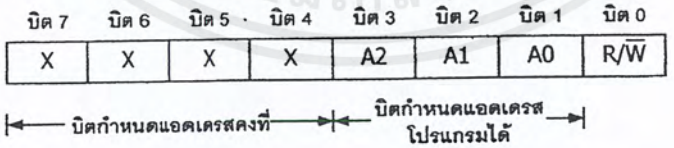
3.2.4 การทำงานบนบัส I^2C

ก่อนที่จะเริ่มต้นการถ่ายทอดข้อมูลระหว่างอุปกรณ์ต่างๆ ที่ต่ออยู่บนบัส ต้องมีการอ้างถึงอุปกรณ์เสียก่อน โดยการอ้างถึงอุปกรณ์บนบัส I^2C นั้นจะใช้การอ้างถึงแบบ 7 บิต หรือ10บิต ในกรณีที่มีอุปกรณ์ต่ออยู่บนบัสไม่มากใช้การอ้างถึงแบบ 7 บิตก็เพียงพอแต่ถ้ามีอุปกรณ์ต่ออยู่บนบัสมากกว่า 127 แอดเดรส จำเป็นต้องใช้การอ้างถึงแบบ 10 บิต หลังจากที่ติดต่ออุปกรณ์แต่ละตัวได้เรียบร้อยแล้ว ก็จะเริ่มต้นการถ่ายทอดข้อมูลกันต่อไป

ในอันดับแรกของการทำงานบนบัส I^2C คือการอ้างถึงอุปกรณ์แต่ละตัว ซึ่งในการอ้างถึงอุปกรณ์สามารถได้ 2 รูปแบบ

การอ้างถึงแบบ 7 บิต

ข้อมูลไบต์แรกที่เกิดขึ้นหลังจากสภาวะเริ่มต้นคือ ข้อมูลที่ใช้ในการอ้างถึงอุปกรณ์ที่ต้องการติดต่อ โดยมีรูปแบบแสดงในรูปที่ 3.5 ใน 7 บิต รวมทั้งบิต MSB ด้วยจะเป็นข้อมูลแอดเดรสของอุปกรณ์สเลฟที่ต้องการติดต่อ โดยแบ่งเป็น บิตกำหนดแอดเดรสคงที่ (fixed address bit) จำนวน 4 บิต ซึ่งข้อมูลนี้อุปกรณ์แต่ละตัวจะถูกกำหนดมาจากผู้ผลิต ไม่สามารถเปลี่ยนแปลงแก้ไขได้ ถัดมาอีก 3 บิต เป็นบิตกำหนดแอดเดรสที่สามารถโปรแกรมได้ (programmable address bit) โดยผู้ใช้งานต้องกำหนดสถานะลอจิกให้แก่ขา A0-A2 ของอุปกรณ์ที่มีการต่อเชื่อมต่อแบบบัส I^2C ส่วนในบิต LSB เป็นบิตที่ใช้กำหนดการอ่านหรือเขียนข้อมูลกับอุปกรณ์สเลฟตัวนั้น ๆ หากบิต LSB เป็น “0” หมายถึงต้องการเขียนข้อมูลไปยังอุปกรณ์นั้น ถ้าเป็น “1” จะเป็นการอ่านข้อมูลจากอุปกรณ์สเลฟ



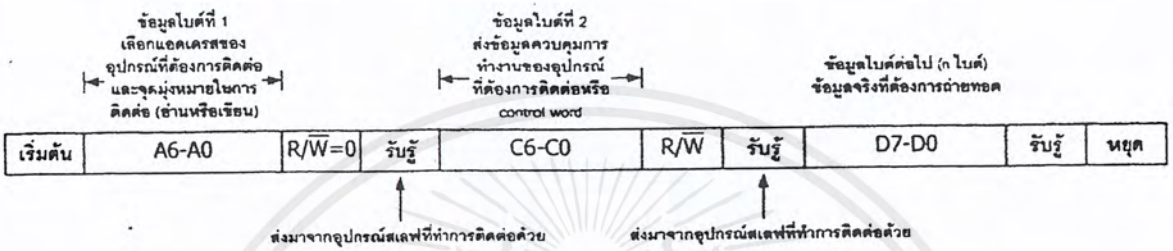
รูปที่ 3.5 แสดงรูปแบบของข้อมูลที่ใช้ในการอ้างถึงอุปกรณ์บนระบบบัส I^2C

ข้อมูลในไบต์ต่อมาคือ ข้อมูลควบคุม (control byte) ในอุปกรณ์แต่ละตัวมีการกำหนดข้อมูลควบคุมที่แตกต่างกันไป ยกตัวอย่าง ไอซีขยายพอร์ตมีข้อมูลควบคุมที่ใช้กำหนดว่า บิตใด

เป็นอินพุต บิตใดเป็นเอ้าต์พุต ในขณะที่ไอซี ADC/DAC ต้องการข้อมูลควบคุมเพื่อกำหนดให้ทำงานเป็นวงจร ADC หรือ DAC เป็นต้น

ข้อมูลในไบต์ต่อมาคือ ข้อมูลที่ทำการถ่ายทอดจริง (data)

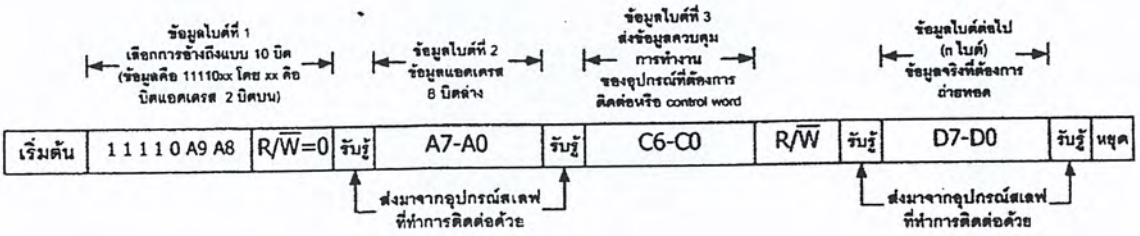
หลังจากที่มีการถ่ายทอดข้อมูลในแต่ละไบต์ อุปกรณ์สเลฟที่ได้รับการติดต่อต้องส่งสัญญาณรับรู้ตอบกลับมาด้วยทุกครั้ง เพื่อให้กระบวนการถ่ายทอดข้อมูลสามารถดำเนินต่อไปได้



รูป 3.6 แสดงรูปแบบข้อมูลอนุกรมที่ใช้ในการติดต่อกับอุปกรณ์บนระบบบัส I^2C แบบ 7 บิต

การอ้างถึงแบบ 10 บิต

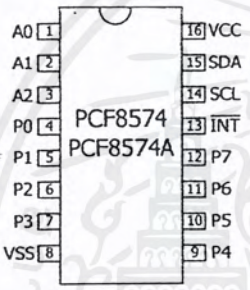
ในการอ้างถึงแบบนี้ ยังคงใช้รูปแบบขอ้มูลอนุกรมที่เหมือนกับแบบ 7 บิต หากแต่จะมีข้อมูลเพิ่มเติมขึ้นมาเล็กน้อย โดยในข้อมูลไบต์แรกหลังจากเกิดสภาวะเริ่มต้น ต้องกำหนดให้ 5 บิตบนมีข้อมูลเป็น 11110 ส่วนอีก 2 บิตถัดมาเป็นบิตแอดเดรสของอุปกรณ์ที่ต้องการติดต่อ ในบิต LSB ของข้อมูลไบต์แรกยังคงเป็นการกำหนดว่าต้องการอ่านหรือเขียนข้อมูลกับอุปกรณ์สเลฟตัวที่ต้องการติดต่อด้วย ข้อมูลไบต์ต่อมาเป็นข้อมูลแอดเดรสในไบต์ที่ 2 ของอุปกรณ์ที่ต้องการติดต่อด้วยข้อมูลไบต์ถัดไปจึงเป็นข้อมูลควบคุมข้อมูลหลังจากนั้นก็จะเป็นข้อมูลจริงที่ใช้ในการติดต่อเช่นเดียวกับการอ้างถึงแบบ 7 บิต หลังจากการถ่ายทอดข้อมูลครบทุกไบต์ ต้องมีสภาวะรับรู้เกิดขึ้น เพื่อให้กระบวนการถ่ายทอดข้อมูลสามรถดำเนินต่อไปได้ ในรูปที่ 3.6 แสดงรูปแบบข้อมูลอนุกรมของการอ้างถึงแบบ 10 บิต



รูปที่ 3.7 แสดงรูปแบบข้อมูลอนุกรมที่ใช้ในการติดต่อกับอุปกรณ์บนระบบบัส I^2C แบบ 10 บิต

3.3 ขยายพอร์ตอินพุตเอาต์พุตให้แก่พอร์ตอนุกรมด้วยระบบบัส I^2C

เพื่อขยายพอร์ตด้วยอุปกรณ์บนระบบบัส I2C เพื่อช่วยให้พอร์ตอนุกรมมีจำนวนพอร์ตใช้งานเพิ่มมากขึ้น โดยใช้สายส่งสัญญาณของพอร์ตอนุกรมเพียง 3 เส้น แต่สามารถขยายพอร์ตอินพุตเอาต์พุตได้มากถึง 128 บิต ไอซีที่ใช้ขยายพอร์ตคือเบอร์ PCF8574 และ PCF8574A โดยไอซีแต่ละเบอร์สามารถขยายพอร์ตอินพุตเอาต์พุตได้ 8 ช่อง หรือ 8 บิต สามารถต่อพ่วงกันได้มากถึง 8 ตัว ทำให้สามารถต่อพอร์ตอินพุตเอาต์พุตได้มากถึง 64 ช่องต่อไอซี 1 เบอร์ ถ้าหากนำ PCF8574 และ PCF8574A มาต่อพ่วงกัน จะเพิ่มจำนวนพอร์ตได้อีก 64 บิต รวมเป็น 128 บิต โดยการเชื่อมต่อนั้นต้องกระทำบนบัส I^2C เท่านั้น

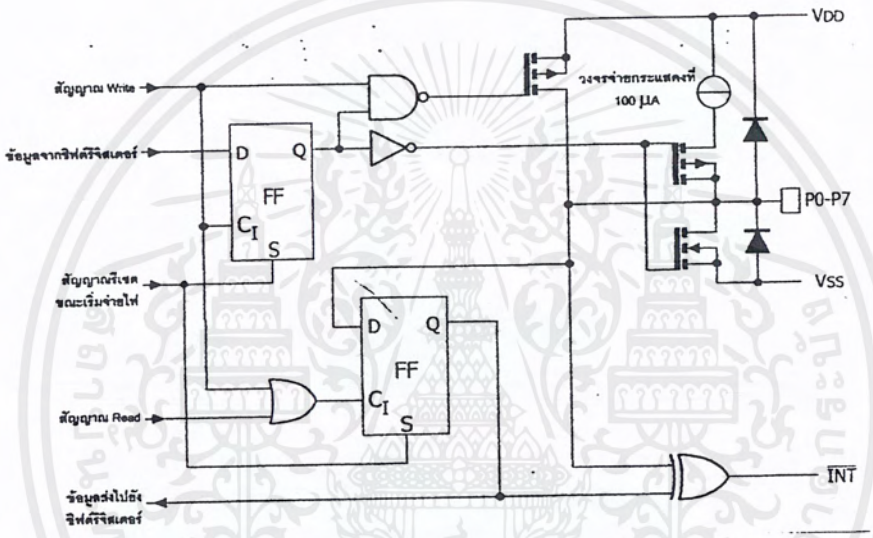


ชื่อ	ตำแหน่งขา	หน้าที่
A0	1	อินพุตแอดเดรสตัวที่ 1
A1	2	อินพุตแอดเดรสตัวที่ 2
A2	3	อินพุตแอดเดรสตัวที่ 3
P0	4	พอร์ตอินพุตเอาต์พุต 2 ทิศทางบิต 0
P1	5	พอร์ตอินพุตเอาต์พุต 2 ทิศทางบิต 1
P2	6	พอร์ตอินพุตเอาต์พุต 2 ทิศทางบิต 2
P3	7	พอร์ตอินพุตเอาต์พุต 2 ทิศทางบิต 3
Vss	8	กราวด์
P4	9	พอร์ตอินพุตเอาต์พุต 2 ทิศทางบิต 4
P5	10	พอร์ตอินพุตเอาต์พุต 2 ทิศทางบิต 5
P6	11	พอร์ตอินพุตเอาต์พุต 2 ทิศทางบิต 6
P7	12	พอร์ตอินพุตเอาต์พุต 2 ทิศทางบิต 7
INT	13	ขาเอาต์พุตอินเตอร์รัปต์(ทำงานที่ลอจิก 0)
SCL	14	ขาสัญญาณนาฬิกาสำหรับ I^2C บัส
SDA	15	ขาข้อมูลสำหรับ I^2C บัส
VCC	16	ไฟเลี้ยง

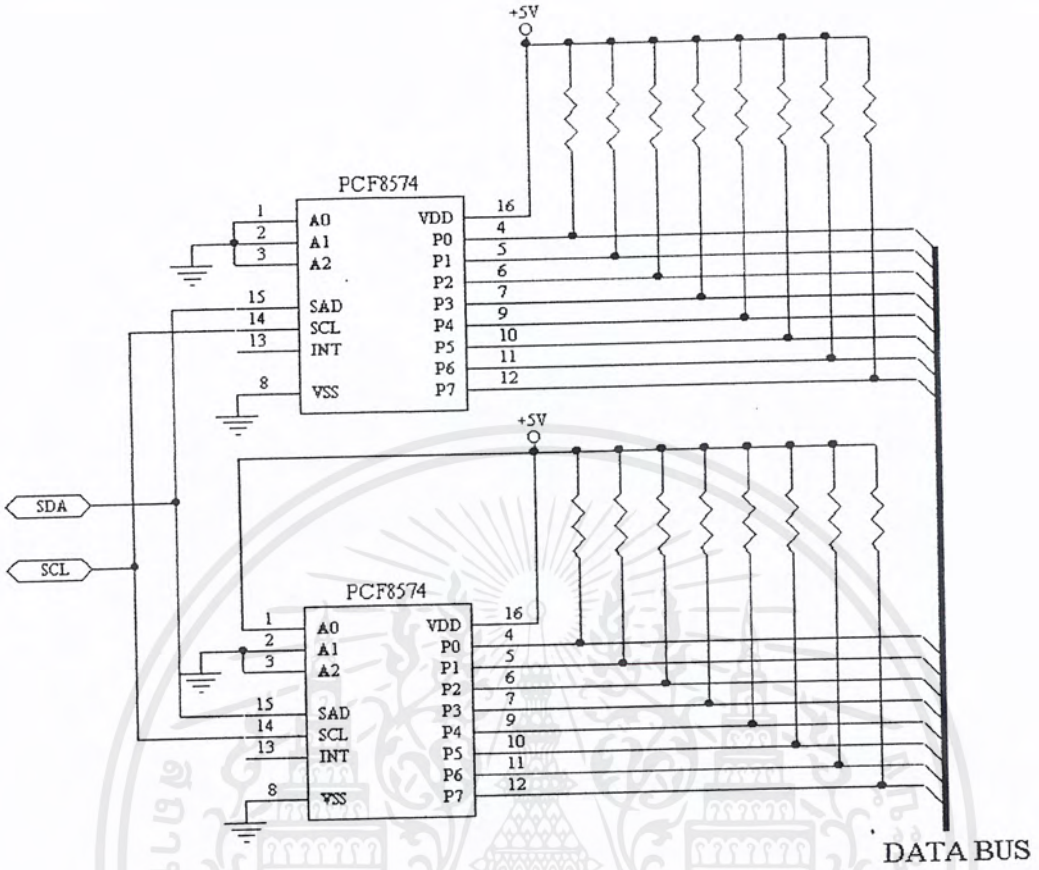
รูปที่ 3.8 แสดงการจัดขาของ PCF8574/8574A และหน้าที่การทำงานของแต่ละขา

ขาพอร์ตทั้ง 8 ขาของ PCF8574 สามารถกำหนดให้เป็นอินพุตหรือเอาต์พุตได้โดยอิสระโดยไม่จำเป็นต้องใช้คำสั่งควบคุมเพื่อเลือกให้เป็นขาเอาต์พุตหรือขาอินพุต ลักษณะวงจรภายในของพอร์ตอินพุตเอาต์พุตแสดงในรูปที่ 3.8 เมื่อจ่ายไฟให้กับ PCF8574 ครั้งแรก ขาพอร์ตทั้ง 8 ขาจะมีลอจิกเป็น "1" ซึ่งจะเป็นการจ่ายกระแสมาจากแหล่งจ่ายกระแสที่แสดงภายในตัวไอซี ทำให้มีกระแสในขณะลอจิก "1"นี้เพียง 100 uA เท่านั้น จึงไม่เหมาะที่จะใช้ไอซี PCF8574 ในการขับกระแสซอร์ส แต่จะเหมาะกับการนำ PCF8574 ไปขับกระแสซิงก์มากกว่า เมื่อต้องการให้ขาพอร์ตเหล่านี้ทำหน้าที่เป็นอินพุตจะต้องส่งสัญญาณให้ขาเหล่านี้มีลอจิก "1" เสียก่อน เมื่อขาอิน

พุตได้รับสัญญาณจากภายนอกป้อนเข้ามา ไอซี PCF8574 จะสร้างสัญญาณอินเตอร์รัปต์(INT) ป้อนให้ไมโครคอนโทรลเลอร์หรือคอมพิวเตอร์รับรู้แทนการต้องคอยตรวจสอบขาอินพุตอยู่ตลอดเวลา สัญญาณอินเตอร์รัปต์นี้จะถูกรีเซตเมื่อมีการอ่านค่าข้อมูลหรือมีการเปลี่ยนค่าของอินพุตไปสู่ค่าเดิม สำหรับ PCF8574 และ PCF8574A มีการทำงานเหมือนกันทุกประการ แตกต่างกันเพียงค่าแอดเดรสที่ใช้ในการติดต่อเท่านั้น ด้วยเหตุนี้จึงสามารถนำไอซีทั้งสองเบอร์นี้มาต่อพ่วงกันเพื่อขยายพอร์ตได้มากถึงเบอร์ละ 8 ตัว รวมทั้งสิ้น 16 ตัว ส่งผลให้สามารถขยายพอร์ตได้สูงสุด 128 บิต



รูปที่ 3.9 แสดงวงจรภายในของขาพอร์ตของไอซี PCF8574 /8574A

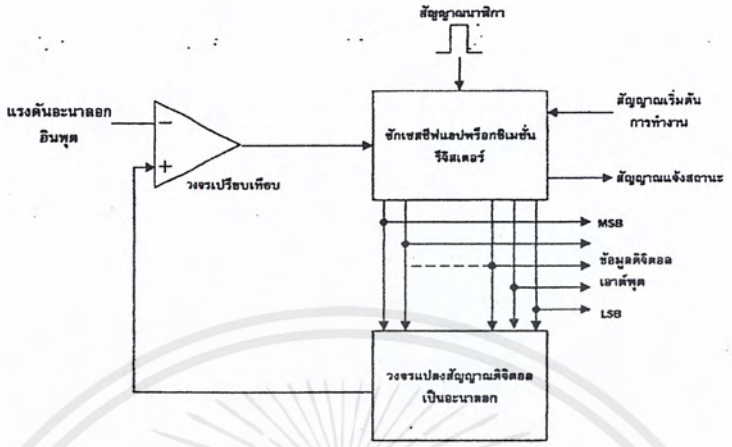


รูปที่ 3.9 แสดงการขยายพอร์ตอินพุทเอาท์พุทด้วย PCF8574 2 ตัว

จากรูปที่ 3.9 ถ้ามีการต่อไอซี PCF8574 เพื่อขยายพอร์ตอินพุทเอาร์พุทต้องมีการอ้างตำแหน่งที่ขา A1 A2 A3 เพื่อให้ไอซีทำงานได้อย่างถูกต้อง โดยการรับส่งข้อมูลจะส่งผ่านทางเส้นสัญญาณ SDA และ SCL ของระบบบัส I^2C

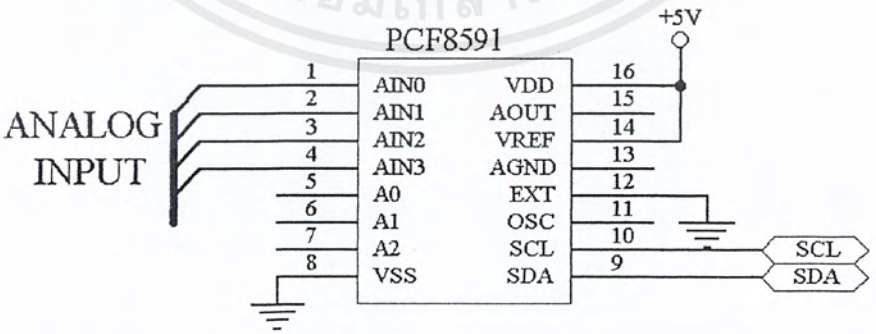
3.4 การเชื่อมต่อสัญญาณอะนาลอกกับพอร์ตอนุกรมผ่านบัส I^2C

ข้อมูลที่ใช้ในการติดต่อกับพอร์ตของคอมพิวเตอร์นั้นจะเป็นสัญญาณดิจิตอลเมื่อมีการติดต่อกับสัญญาณอะนาลอกจากภายนอกจะต้องมีการใช้ไอซีในการแปลงสัญญาณอะนาลอกเป็นสัญญาณดิจิตอลซึ่งเรียกว่า ไอซี DAC (Analog to Digital Converter) โดยในส่วนของฮาร์ดแวร์ได้เลือกใช้ไอซีเบอร์ PCF8591 ซึ่งเป็นไอซีที่แปลงระดับสัญญาณแบบประมาณค่าใกล้เคียงบล็อกไดอะแกรมของกระบวนการ DAC แสดงดังรูปที่ 3.10 ซึ่งประกอบด้วยวงจรเปรียบเทียบแรงดัน วงจรแปลงสัญญาณสัญญาณดิจิตอลเป็นอะนาลอก (Digital to Analog Converter) สัญญาณนาฬิกา และส่วนควบคุมลอจิก



รูปที่ 3.10 แสดงไดอะแกรมแสดงการทำงานของวงจร ADC แบบ Successive Appoximation

การใช้งานระบบบัส I^2C ต้องมีการกำหนดแอดเดรสของอุปกรณ์ตัวนั้นอย่างชัดเจน ในการกำหนดแบบ 7 บิท ข้อมูลกำหนดแอดเดรส 4 บิทบนจะเป็นข้อมูลเฉพาะของไอซี PCF8591 มีค่าเท่ากับ 1001 (ฐานสอง) ข้อมูล 3 บิทถัดมาเป็นการกำหนดที่ฮาร์ดแวร์เพื่อเลือกใช้ไอซี PCF8591 ที่ต้องการติดต่อในกรณีที่ใช้ไอซีมากกว่า 1 ตัวจากรูปที่ 3.11 แสดงวงจรที่ใช้งานขา 5,6,7 คือขาสำหรับกำหนดแอดเดรสทางฮาร์ดแวร์ ส่วนบิท LSB ใช้ในการกำหนดว่าต้องการอ่านหรือเขียนข้อมูลกับไอซีตัวนั้นๆ ในกรณีที่ต้องการอ่านข้อมูลจากไอซีขา 1,2,3,4 จะเป็นขาที่ทำหน้าที่รับสัญญาณอะนาลอกเข้ามา ในส่วนออสซิลเลเตอร์ไอซี PCF8591 จะมีวงออสซิลเลเตอร์ภายในเมื่อเลือกใช้ออสซิลเลเตอร์ภายในขา EXT ต้องต่อลงกราวด์



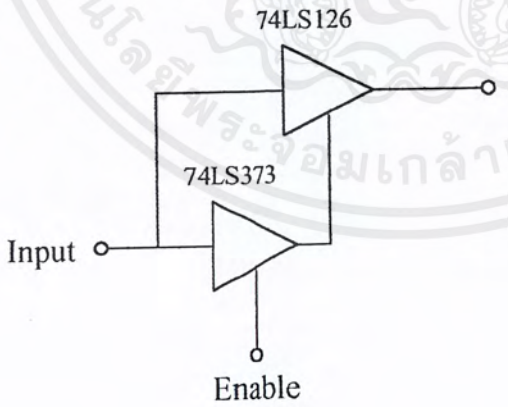
รูปที่3.11 แสดงวงจรใช้งาน PCF8591

3.5 บัฟเฟอร์ของ Socket Test

ในส่วนของ Socket Test ที่สำหรับนำ IC มาทำการทดสอบนั้นจะต้องมีวงจรบัฟเฟอร์เพื่อกันแรงดันไฟที่ส่งมาจาก PCF8574ไม่ให้มารบกวนขาเอาท์พุทของ IC ที่นำมาทดสอบซึ่งโดยปรกติจะมีแรงดันที่ส่งมาจาก PCF8574 ป้อนให้กับ Socket Test ทุกขาเพื่อเป็นแรงดันสำหรับทดสอบ IC แรงดันนี้ถ้าป้อนให้ขาเอาท์พุทของ IC ที่นำมาทดสอบทำให้เกิดการหักล้างของแรงดันเอาท์พุทระดับแรงดันเอาท์ที่ได้มีค่าไม่ถูกต้องซึ่งรูปวงจรบัฟเฟอร์แสดงดังรูปที่ 3.12

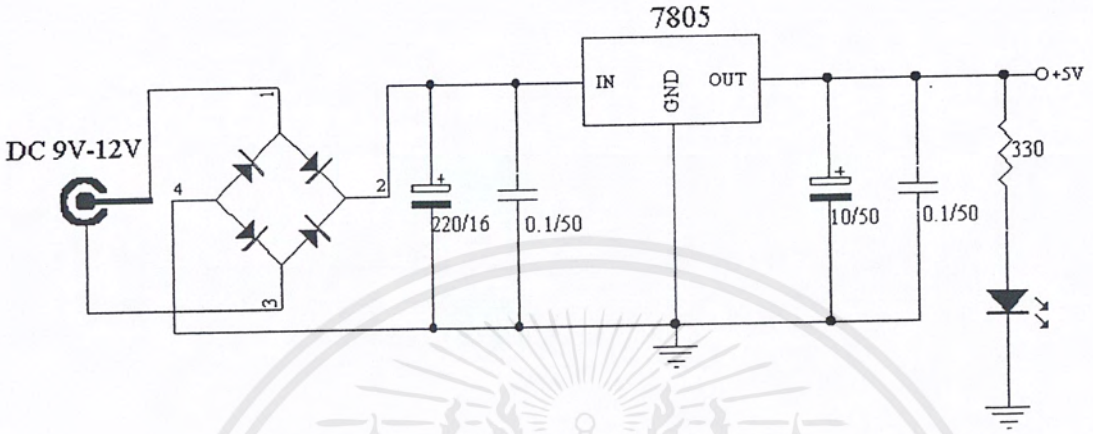
การทำงานของวงจร

การทำงานของวงจรซึ่งดูจากรูปสามารถอธิบายได้ดังนี้ วงจรบัฟเฟอร์ที่เห็นในรูปประกอบด้วย IC เบอร์ 74LS126 และ 74LS373 โดยต้องมีทั้งหมด 22 ชุดยกเว้นขา VCC และขา GND ไม่ต้องมีวงจรบัฟเฟอร์ ทางด้านอินพุทของวงจรบัฟเฟอร์จะต่อกับเอาท์พุทของ PCF8574 ส่วนเอาท์พุทของบฟเฟอร์จะต่อกับ Socket Test ถ้าต้องการให้บัฟเฟอร์ Enable หรือให้ PCF8574 ส่งแรงดันมาที่ Socket Testจะต้องส่งข้อมูลเป็น 1 มาที่อินพุทแล้วป้อนแรงดัน Enable เอาท์พุทของ 74LS373 จะ Latch สภาวะ 1 ไว้เพื่อEnable 74LS126 ให้ทำงานจึงสามารถส่งแรงดันมาที่เอาท์พุทได้ ส่วนการ disable หรือกันไม่ให้ PCF8574 ส่งแรงดันมาที่ Socket Test ต้องทำการส่งข้อมูลที่เป็น 0 มาที่อินพุทของบัฟเฟอร์แล้วป้อนแรงดัน Enable เอาท์พุทของ 74373 จะ Latch สภาวะ 0 ไว้เพื่อ disable 74126 ทำให้ไม่สามารถส่งแรงดันมาที่ Test Socket ได้



รูปที่3.12 แสดงวงจรบัเฟอร์

3.6 ภาคจ่ายไฟ



รูปที่ 3.13 แสดงวงจรภาคจ่ายไฟ

ภาคจ่ายไฟของวงจรในส่วนของฮาร์ดแวร์จะใช้ DC 9V-12V จากอะแดปเตอร์ที่จ่ายกระแส 1000 mA มาเข้าวงจร Bridge Rectifier แล้วผ่านวงจร Regulator โดยใช้ไอซีเบอร์7805 เพื่อให้ได้ไฟ DC คงที่ +5V จ่ายให้วงจรทั้งหมด

บทที่4
การทำงานและการใช้งานโปรแกรม

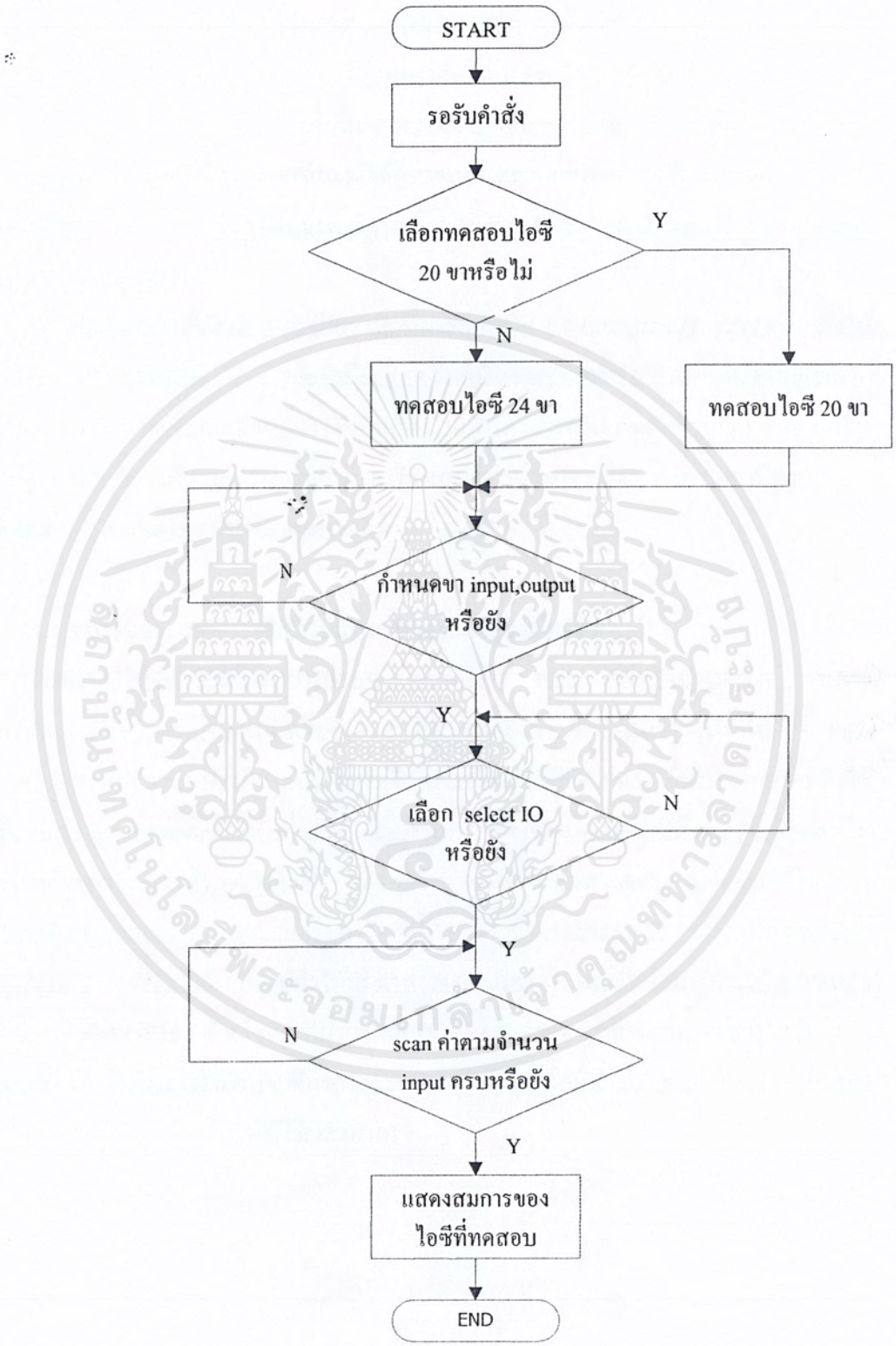
4.1 การทำงานของโปรแกรม

ในโครงงานนี้ต้องการสร้างเครื่องทดสอบไอซี PAL ซึ่งแสดงผลการทดสอบในรูป Timing Diagram และวิเคราะห์หาสมการลอจิกที่อยู่ในไอซี PAL ออกมโดยทางด้าน ฮาร์ดแวร์ทำการตรวจสอบสถานะลอจิกของไอซี PAL ที่นำมาทดสอบแล้วนำผลที่ได้ไปประมวลผลด้วยซอฟร์แวร์ ในส่วนซอฟร์แววว์รของโครงงานนี้เขียนด้วยภาษา Visual Basic ซึ่งแบ่งออกได้เป็น 3 ส่วนคือ

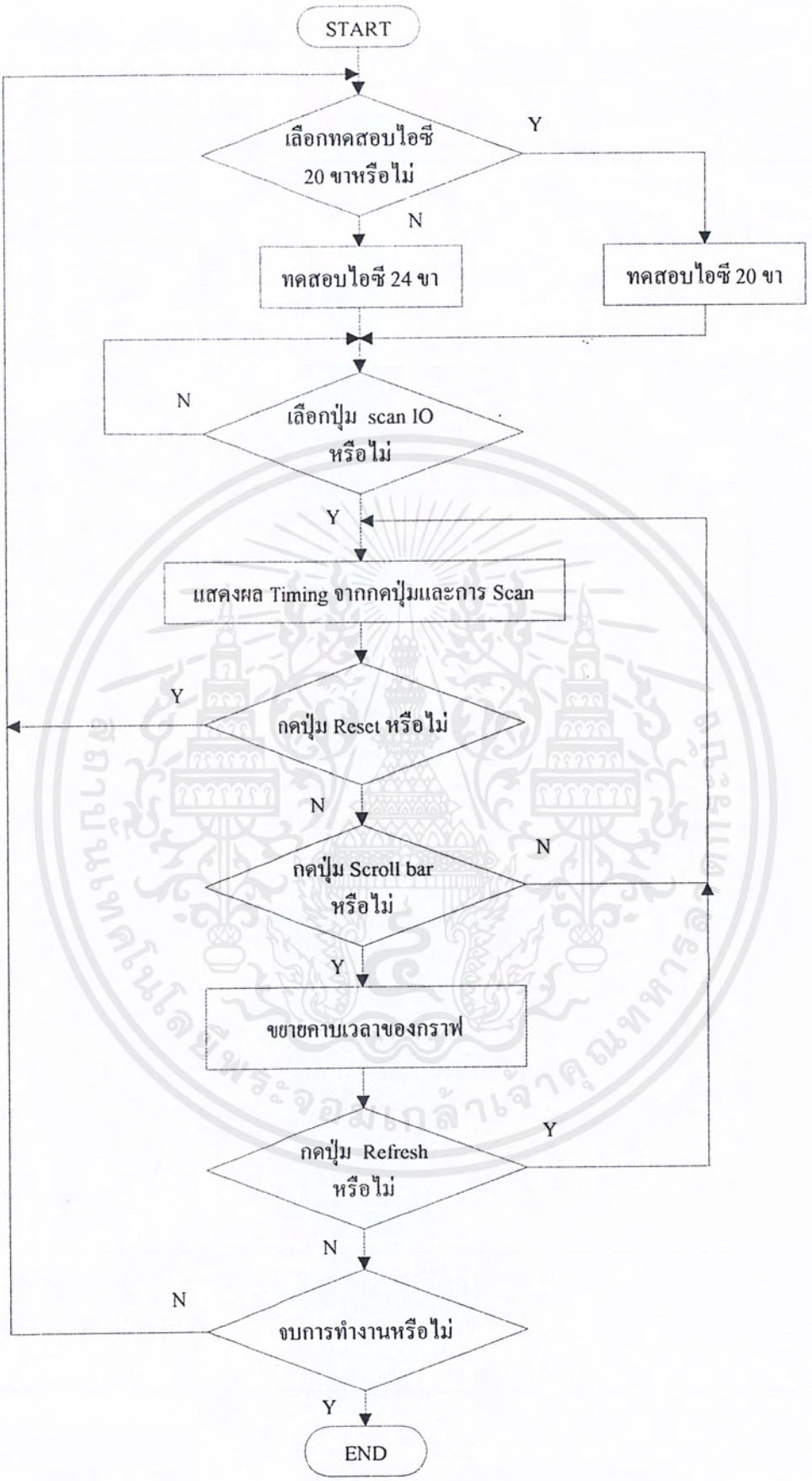
4.1.1 โปรแกรมหลัก

ในส่วนโปรแกรมหลักนี้จะทำหน้าที่สั่งงานในการทดสอบไอซี ว่าจะทำการทดสอบไอซี 20 ขาหรือ 24 ขา ทำหน้าที่เลือกขาอินพุตและเอาท์พุทของไอซีที่ทำการทดสอบ ทำการสั่งให้แสดงสมการลอจิกที่ได้จากการวิเคราะห์นอกจากนี้ยังสั่งงานให้เริ่มต้นแสดง Timing Diagram อีกด้วย โฟลวชาร์จการการทำงานแสดงดังรูปที่ 4.1 โดยเริ่มแรกรอรับคำสั่งว่าจะทดสอบไอซีกี่ขาแล้วทำการเลือกขาอนพุทและเอาท์พุท หลังจากนั้นก็ทำการ Enable โดยการกดที่ปุ่ม Select IO เพื่อทำการสแกนค่าอินพุทและเก็บค่าเอาท์พุท แล้วนำข้อมูลที่ได้มาทำการวิเคราะห์และแสดงผล

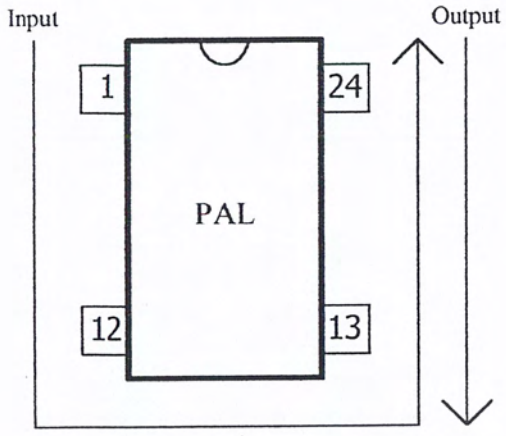
ในส่วนของโปรแกรมสำหรับเลือกไอซี 20 ขาหรือ 24 ขานั้นจะมีลักษณะการทำงานที่เหมือนกันเพียงแต่เปลี่ยนที่หน้าต่างของโปรแกรมสำหรับผู้ใช้งานให้สามารถมองดูง่ายขึ้นคือ ถ้าเป็นไอซี 20 ขาก็จะมี LED แสดงแค่ 20 ดวง กราฟ Timing Diagram แสดงแค่ 18 ขาเท่านั้นเพราะจะตัดขาที่เป็น GND และ VCC ออก ในทำนองเดียวกันไอซี 24 ขา มีLED 24 ดวงและกราฟ Timing Diagram แสดง 22 ขาเท่านั้น



รูปที่ 4.1 แสดงโฟลวชาร์จการทำงานของโปรแกรมหลัก

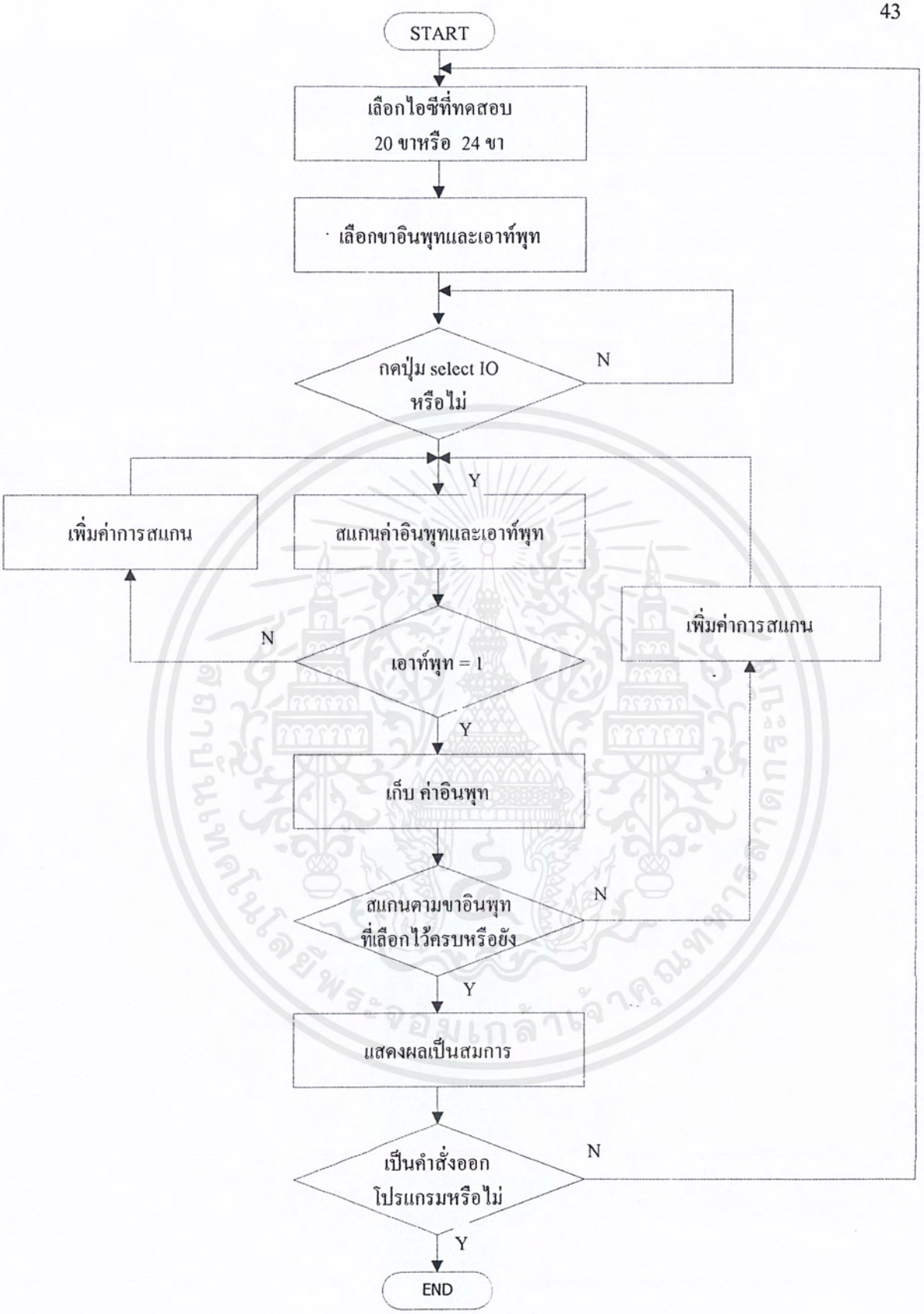


รูปที่ 4.2 แสดงโฟลวชาร์จแสดงการทำงานในส่วนของ Timing Diagram



รูปที่ 4.3 แสดงการสแกนค่าอินพุทและเอาท์พุท

ในส่วนของการอ่านค่าเอาท์พุทนั้นต้องตรวจสอบว่าอินพุทที่ป้อนให้กับไอซีนั้นได้เอาท์พุทเป็นอะไร ซึ่งเราจะสนใจเฉพาะอินพุทที่ป้อนให้กับไอซีแล้วได้เอาท์พุทที่เป็นลอจิก "1" เท่านั้นหรือในรูปของ minterm แล้วแล้วเก็บค่าอินพุทไว้ใน Arrayซึ่งค่าที่เก็บนี้จะนำมาทำการวิเคราะห์หาสมการอีกทีหนึ่ง โดยวิธีการสแกนเพื่อเก็บค่าอินพุทและเอาท์พุทจะใช้การสแกนแล้วเพิ่มค่าขึ้นทีละ 1 ค่าแล้วตรวจสอบว่าเอาท์พุทเป็น 1 หรือไม่ โดยจะเก็บค่าอินพุทที่ทำให้เอาท์พุทเป็น 1 แล้วจึงเพิ่มค่าการสแกนขึ้นอีก 1 ค่าเมื่อทำครบแล้วก็จะนำค่าอินพุทที่ทำให้เอาท์พุทเป็น 1 มาวิเคราะห์หาสมการรูปการสแกนแสดงดังรูปที่ 4.3 และรูปที่ 4.4 แสดงโฟลวชาร์จการสแกนค่าอินพุทและเอาท์พุท

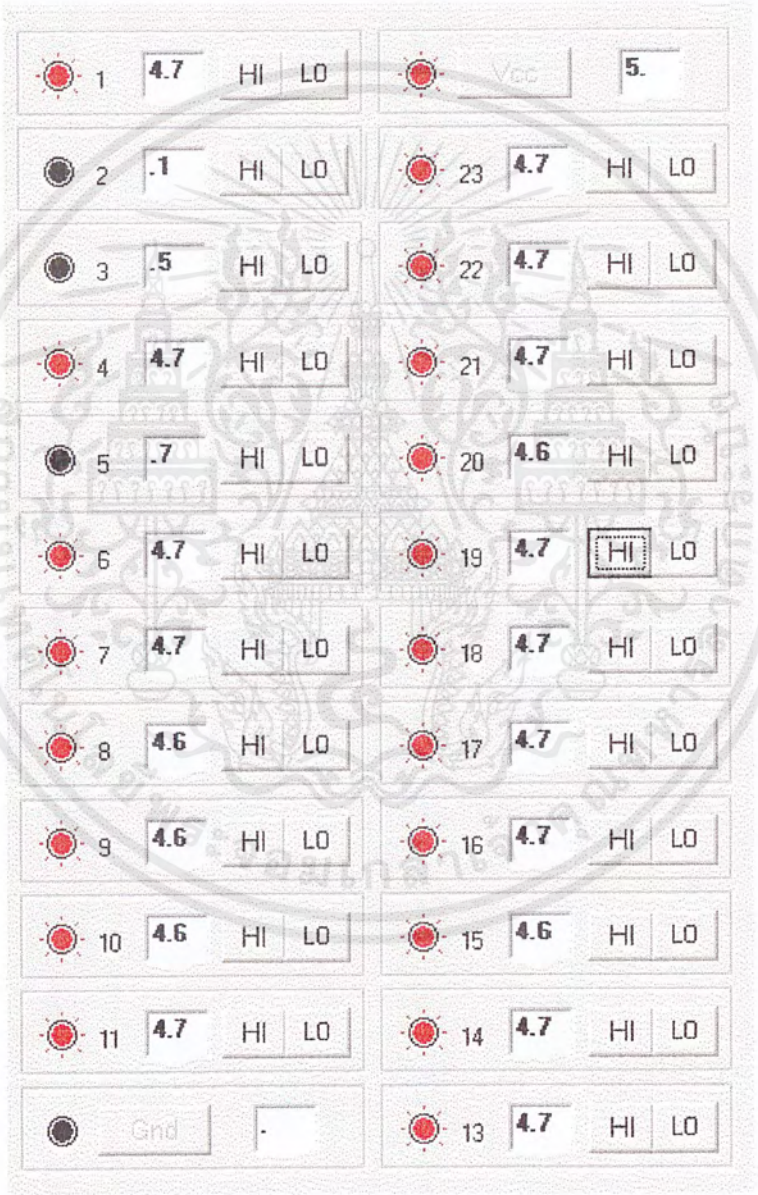


รูปที่ 4.4 แสดงโฟลวชาร์จการสแกนค่าอินพุทและเอาท์พุท

4.2 การใช้งานโปรแกรม

การทำงานของโปรแกรมจะอธิบายตามรูปภาพโดยในแต่ละภาพคือส่วนของการสั่งงานของผู้ใช้สั่งงานโปรแกรมดังนี้คือ

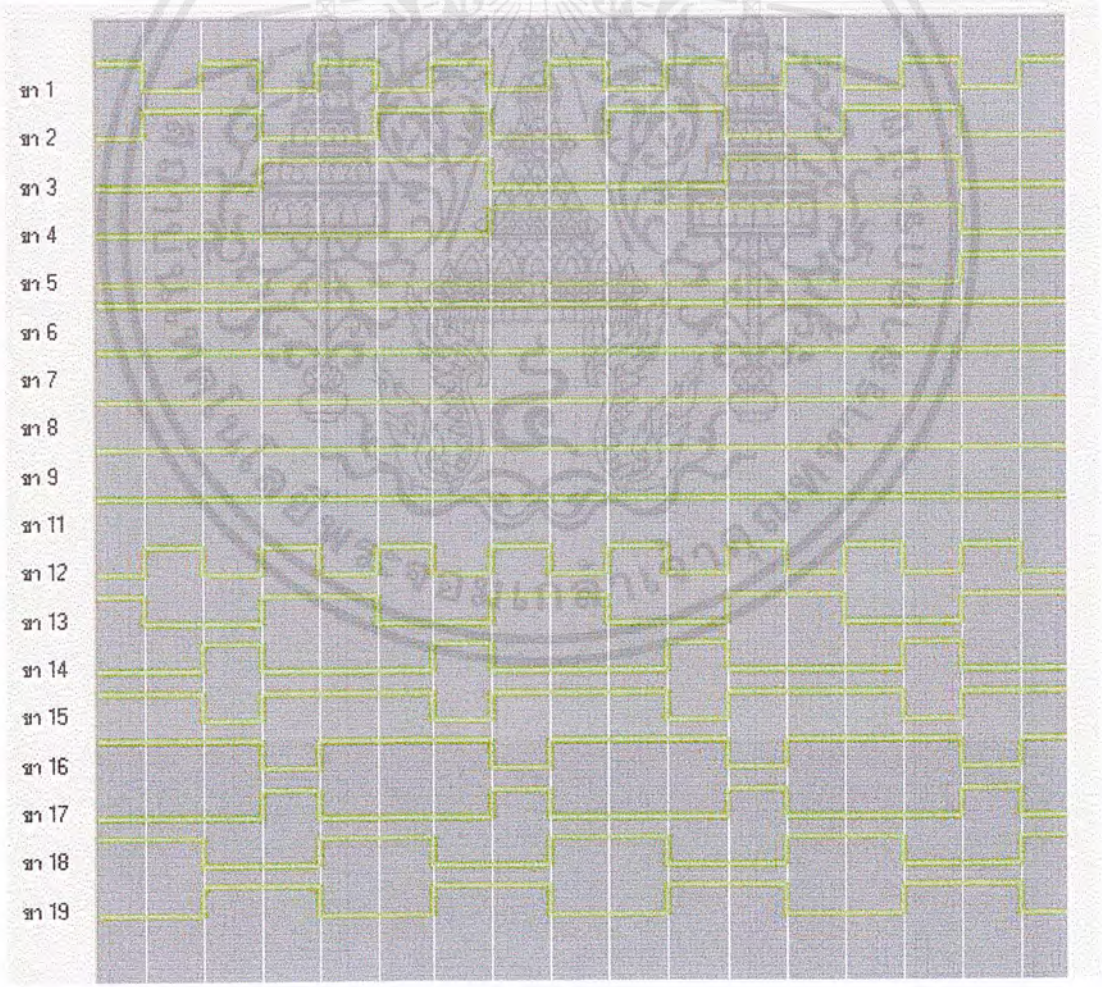
4.2.1 ส่วนของการสั่งงานด้วยปุ่ม HI และ LO



รูปที่ 4.5 แสดงโปรแกรมการสั่งงานด้วยปุ่ม HI และ LO

จากรูปที่เห็นเป็นการทดสอบไอซี PAL 24 ขาจะเห็นว่ามีปุ่มใช้งานอยู่ 2 ปุ่มคือ HI และ LO ใช้สำหรับสั่งให้ป้อนลอจิก "1" และ "0" กับไอซีที่ทดสอบ ในแต่ละขาจะมี LED และ TextBox สำหรับแสดงสถานะของลอจิกในขณะนั้นโดยที่ LED สีแดง
หมายถึงลอจิกเป็น "1" ตัวเลขใน TextBox จะใกล้เคียงกับ 5V และถ้า LED สีดำหมายถึงลอจิกเป็น "0" ตัวเลขใน TextBox จะเท่ากับ 0V ซึ่งตัวเลขที่อยู่ใน TextBox ได้มาจาก PCF 8591 ทำการแปลงสัญญาณอนาลอกเป็นดิจิตอลแล้วโปรแกรมจะทำการแปลงกลับมาเป็นอนาลอกแสดงใน TextBox ให้เห็นอีกครั้ง และในส่วนของการทดสอบไอซี 20ขาก็จะมีลักษณะเช่นเดียวกันเพียงแค่เปลี่ยนจาก 24ขาเหลือ 20ขาเท่านั้นเอง

4.2.2 ส่วนการแสดงผล Timing Diagram



รูปที่ 4.6 แสดงการแสดงผลของ Timing Diagram

จากรูปที่เห็นเป็นทดสอบไอซี PAL 20ขา จะเห็นว่ามีอยู่ 2 ขาที่ไม่แสดงให้เห็นคือขา VCC และขา GND โดยในรูปได้เลือกขา 1,2,3,4,5 เป็นขาอินพุทและขา 12,13,14,15,16,17,18,19 เป็นขาเอาท์พุทลักษณะของ Timing Diagram จะแสดงให้เห็นตามสมการลอจิกที่อยู่ภายในของไอซี PAL ส่วนขาที่ไม่เลือกก็จะแสดงเป็น "1" ตลอด

4.2.3 ส่วนของปุ่มการสั่งงาน

ในส่วนของปุ่มสั่งงานของโปรแกรมนี้จะมีอยู่ 2 ส่วนด้วยกันดังแสดงในรูปที่ 4.7และรูปที่ 4.8 ในรูปที่ 4.7 จะมีอยู่ทั้ง 5 ปุ่มและมีหน้าที่ดังนี้

รูปที่ 4.7 แสดงปุ่มการสั่งงาน

- ปุ่มStop Time ทำหน้าที่ในการสั่งงานให้มีการแสดงข้อมูลของระดับแรงดันใน TextBox

-ปุ่ม Pin 24 ทำหน้าที่เลือกว่าจะทดสอบไอซี 20 ขาหรือ 24 ขา

-ปุ่ม Reset ทำหน้าที่สั่งงานให้เริ่มต้นแสดง Timing Diagram ใหม่

-ปุ่ม Refresh ทำหน้าที่สั่งงานให้แสดง Timing Diagram ใหม่อีกครั้ง

-ปุ่ม EXIT ทำหน้าที่สั่งให้ออกจากโปรแกรม

รูปที่ 4.8 แสดงปุ่มการสั่งงาน

จากรูปที่ 4.8 การทำงานในแต่ละส่วนมีดังนี้

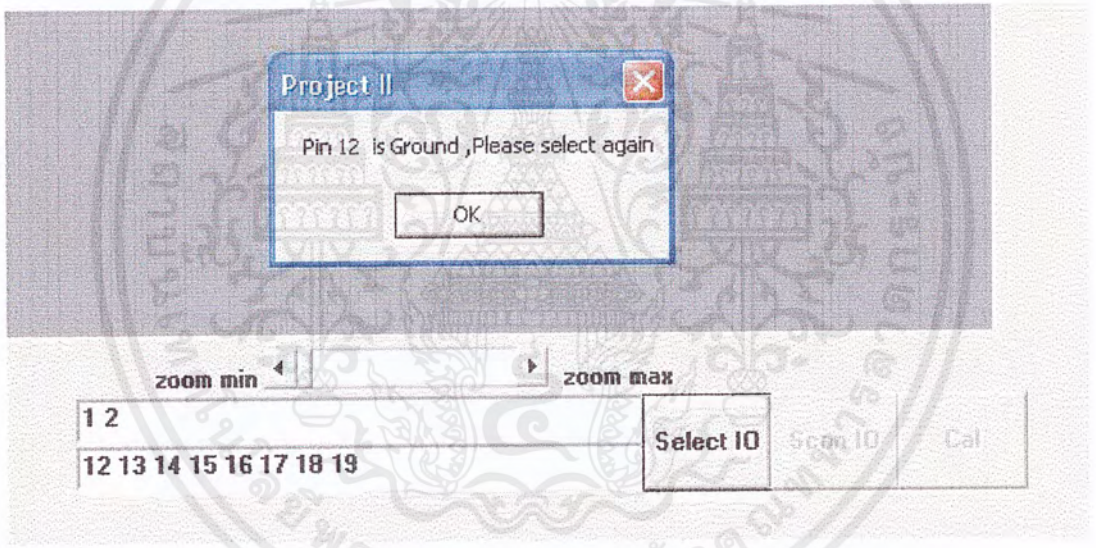
-ปุ่ม Select IO ทำหน้าที่สั่งงานให้ Enable Buffer หลังจากที่เลือกขาอินพุทและขาเอาท์พุทแล้ว

-ปุ่ม Scan IO ทำหน้าที่สั่งงานให้ป้อนอินพุทกับไอซีที่ทดสอบโดยจำนวนค่าในการป้อนเท่ากับ 2^X (X = จำนวนอินพุท)

-ปุ่ม Cal ทำหน้าที่สั่งงานให้แสดงสมการเอาท์พุทจากการทดสอบไอซี PAL ให้เห็นโดยสมการจะมีจำนวนเท่ากับจำนวนเอาท์พุทที่ได้เลือกไว้

-Text Box Input Pin และ Output Pin ทำหน้าที่สำหรับป้อนขาอินพุทและเอาท์พุทของไอซีที่ทดสอบการป้อนต้องมีการเว้นดังตัวอย่างเช่น เลือกอินพุทขา 1 ถึงขา 4 ต้องป้อนดังนี้คือ 1 2 3 4 เป็นต้น ในกรณีที่มีการป้อนผิดเช่นเลือกขา VCC และ GND เป็นขาอินพุทหรือเอาท์พุทจะข้อความเตือนเพื่อให้เลือกใหม่อีกครั้งดังแสดงในรูปที่ 4.9

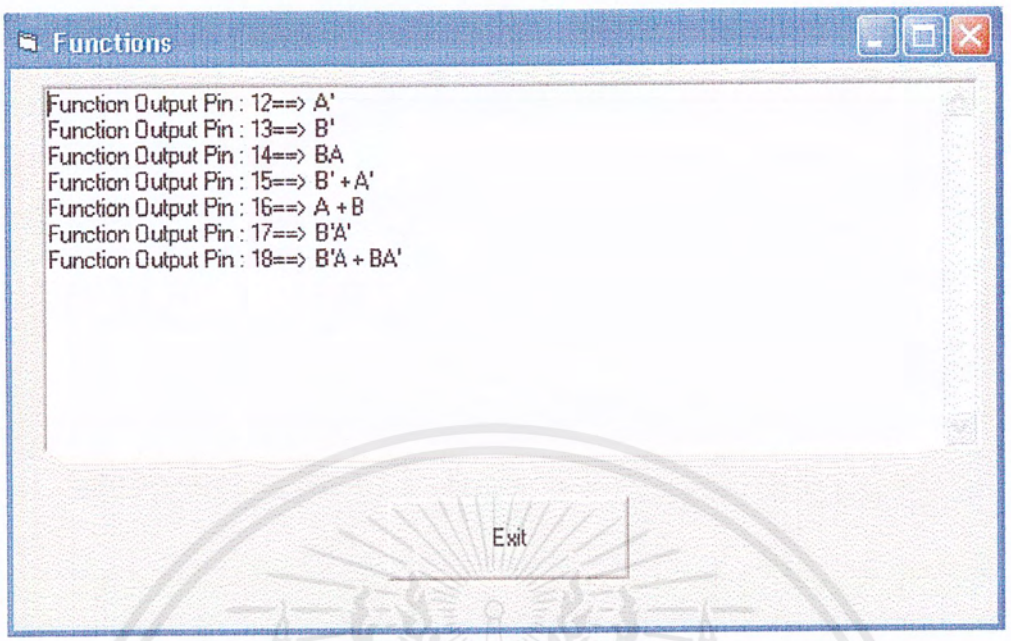
-ScrollBar ทำหน้าที่สั่งงานให้คาบเวลาของ Timing Diagram กว้างขึ้น



รูปที่ 4.9 แสดงข้อความเตือนเมื่อเลือกขาอินพุทขาเอาท์พุทผิด

4.2.4 ส่วนของการแสดงสมการลอจิก

การแสดงสมการลอจิกจะเห็นจากรูปที่ 4.10 ซึ่งสมการที่เห็นจะมีจำนวนเท่ากับขาเอาท์พุทที่เลือกไว้ใน Text Box ซึ่งในรูปจะเห็นปุ่ม Exit มีไว้สำหรับสั่งงานให้กลับไปยังหน้าจอของโปรแกรมหลัก



รูปที่ 4.10 แสดงสมการลอจิก

บทที่ 5
สรุปผลการทำงานและวิจารณ์

การทำงานและข้อจำกัดของการใช้งาน

ฮาร์ดแวร์จะทำการตรวจสอบสถานะลอจิก แล้วเก็บข้อมูลเพื่อนำไปวิเคราะห์หาสมการลอจิกและแสดงผลเป็นTimingDiagramโดยใช้เทคโนโลยี I^2C ในการส่งข้อมูลแล้วโปรแกรมก็เอาข้อมูลที่ได้มาประมวลผลซึ่งการทำงานของเครื่อง PLD TESTER ที่สร้างขึ้นมีข้อจำกัดการทำงานโดยสามารถสรุปได้ดังนี้

1. ไอซี PAL ที่สามารถอ่านได้ต้องเป็นวงจรดิจิตอลแบบ Combination เท่านั้นไม่สามารถอ่านไอซี PAL ที่เป็นวงจรดิจิตอลแบบ Sequential ได้

2. ถ้าไอซี PAL ที่ใช้ขาอินพุทมากจะใช้เวลานานในการสแกนค่าอินพุท

3. การวิเคราะห์หาสมการโปรแกรมจะรองรับตัวแปรอินพุทได้สูงสุด 4 ตัวแปรหรือสามารถเลือกขาอินพุทได้สูงสุด 4 ขาเท่านั้นถ้ามากกว่านี้โปรแกรมจะ Error

4. สมการเอาท์พุทแสดงให้เห็นได้สูงสุด 7 สมการถ้ามากกว่านี้โปรแกรมจะ Error

แนวทางในการพัฒนาต่อ

1.สามารถพัฒนาเพื่อที่อ่านไอซี PAL ที่เป็นวงจร Sequential ได้

2. พัฒนาโปรแกรมส่วนของการวิเคราะห์หาสมการลอจิก(โปรแกรม QM)ให้รองรับตัวแปรได้มากขึ้น

3. พัฒนาโปรแกรมส่วนของการสแกนอินพุทและเอาท์พุทให้ใช้เวลาน้อยลง

เอกสารอ้างอิง

1 กฤษดา ใจเย็น และคณะ , เรียนรู้และปฏิบัติการเชื่อมต่อคอมพิวเตอร์กับอุปกรณ์ภายนอกผ่านพอร์ตอนุกรม , บริษัทอินโนเวทีฟ เอ็กเพอริเมนต์ จำกัด

2 ธวัชชัย เลื่อนฉวี , ดิจิตอลเทคนิค เล่ม 1และ เล่ม 2 , พิมพ์ครั้งที่ 7 , 23 บุ๊คเซ็นเตอร์, มิตรนราการพิมพ์

3 ธีรวัฒน์ ประกอบผล , ดิจิตอลอิเล็กทรอนิกส์ , แมคกรอ-ฮิล อินเตอร์เนชั่นแนล อิน เตอไพร์ส , อิงค์

4 ประวีณา อมรรัตนกุล , คู่มือนักออกแบบ Microsoft Visual Basic Design Patterns, พิมพ์ครั้งที่ 1 บริษัทซีเอ็ดยูเคชั่น จำกัด (มหาชน) , สำนักพิมพ์สามย่าน.COM

5 สัจจะ จรัสรุ่งรวีวร , คู่มือการสร้างแอพพลิเคชันด้วย Visual Basics 6 , พิมพ์ครั้งที่ 1 , บริษัทดวงกมลสมัย จำกัด , สำนักพิมพ์อินโฟเพรส

ภาคผนวก ก.

Soflware Program

โปรแกรมหลัก

```
Private Declare Sub Sleep Lib "kernel32" (ByVal dwMilliseconds As Long)
Dim Minterms() As String
Dim DontCares() As String
Dim Variables() As String
Dim AllTerms() As String
Dim InitFormHeight As Integer
Dim WidthTime As Single
Dim HightTime As Single
Dim x As Single
Dim y As Single
Dim Data As Byte
Dim Data1 As Byte
Dim Data2 As Byte
Dim KeepOldScroll As Integer
Dim IndexDataRow As Integer
Dim IndexDataColum As Integer
Dim CountDataColumIndex As Integer
Dim A() As Boolean
Dim KeepPinOutputIs1(1 To 23) As String
Dim PinInput() As String
Dim PinOutput() As String
Dim ShowEqu() As String
Dim ShowEquNum As Integer

Private Sub CmdCal_Click()
Dim i As Integer
Dim j As Integer
Dim Str As String
ShowEquNum = 0
ReDim ShowEqu(UBound(PinOutput))
For i = 0 To UBound(PinOutput)
Str = KeepPinOutputIs1(Val(PinOutput(i)))
   Call QM(Str, Val(UBound(PinInput)) + 1)
Next
For i = 0 To UBound(PinOutput)
Str = "Function Output Pin : " & PinOutput(i) & "==> " & ShowEqu(i) & vbNewLine
ShowEquation.txtEqu.Text = ShowEquation.txtEqu.Text & Str
Next
ShowEquation.Show
```

```
For i = 0 To UBound(PinOutput)
   KeepPinOutputIs1(Val(PinOutput(i))) = ""
Next
CmdCal.Enabled = False
End Sub

Function CheckInput(ByRef StoreInArray() As String) As Boolean
Dim i  As Integer
Dim InArray As Integer
For i = 0 To UBound(StoreInArray)
InArray = Val(StoreInArray(i))
If CmdSelectPin.Caption = "Pin 24" Then
   If InArray = 24 Then
      MsgBox " Pin 24 is Vcc,Please select again"
      CheckInput = True
   ElseIf InArray = 12 Then
      MsgBox "Pin 12  is Ground ,Please select again"
      CheckInput = True
   ElseIf InArray > 24 Then
      MsgBox "Data out of range"
      CheckInput = True
   End If
Else
If InArray = 20 Then
   MsgBox "Pin 20 is Vcc ,Please select again"
   CheckInput = True
   ElseIf InArray = 10 Then
      MsgBox "Pin 10  is Ground ,Please select again"
      CheckInput = True
   ElseIf InArray > 20 Then
      MsgBox "Data out of range"
      CheckInput = True
   End If
End If
Next
End Function

Private Sub CmdGetPinIO_Click()
Dim i As Integer
If txtInputPin = "input Pin " Or txtOutputPin = "output Pin " Then
```

```
    MsgBox "Please enter pin Input and pin Output"
    txtInputPin.SetFocus
    Exit Sub
End If
GetInputOutput PinInput, TrimS(txtInputPin)
GetInputOutput PinOutput, TrimS(txtOutputPin)
If InputSameAsOutput = True Then
    MsgBox " Please enter pin Input and pin Output, Example 1 2 3....."
    Exit Sub
ElseIf RepeatedTerm(PinInput) Then
    MsgBox "At least one input is entered twice. Remove the duplicate and try again."
    Exit Sub
ElseIf RepeatedTerm(PinOutput) Then
    MsgBox "At least one output is entered twice. Remove the duplicate and try again."
    Exit Sub
ElseIf CheckInput(PinInput) = True Then
    Exit Sub
ElseIf CheckInput(PinOutput) = True Then
    Exit Sub
End If

  Call I2CStart              'Start
  Call Send8BIT(&H71)        'Control Word Read
  Call Ack                   'Acknowledge
  Data = dat                 'Read Data
  Call Ack                   'Acknowledge
  Call I2CStop               'Stop
  Call I2CStart              'Start
  Call Send8BIT(&H73)        'Control Word Read
  Call Ack                   'Acknowledge
  Data1 = dat                'Read Data
  Call Ack                   'Acknowledge
  Call I2CStop               'Stop
  Call I2CStart              'Start
  Call Send8BIT(&H75)        'Control Word Read
  Call Ack                   'Acknowledge
  Data2 = dat                'Read Data
  Call Ack                   'Acknowledge
  Call I2CStop               'Stop
  For i = 0 To UBound(PinInput)
```

```
If CmdSelectPin.Caption = "Pin 24" Then
    If Val(PinInput(i)) > 16 Then
        Data2 = Data2 Or 2 ^ (Val(PinInput(i)) - 17)
    ElseIf Val(PinInput(i)) > 8 Then
        Data1 = Data1 Or 2 ^ (Val(PinInput(i)) - 9)
    Else
        Data = Data Or 2 ^ (Val(PinInput(i)) - 1)
    End If
Else ' "Pin20"
  If Val(PinInput(i)) >= 15 Then
      Data2 = Data2 Or 2 ^ (Val(PinInput(i)) - 15)
    ElseIf Val(PinInput(i)) >= 7 Then
        Data1 = Data1 Or 2 ^ (Val(PinInput(i)) - 7)
    Else
      Data = Data Or 2 ^ (Val(PinInput(i)) + 1)
    End If
  End If
Next
For i = 0 To UBound(PinOutput)
If CmdSelectPin.Caption = "Pin 24" Then
    If Val(PinOutput(i)) > 16 Then
      Data2 = Data2 And (Not (CByte(2 ^ (Val(PinOutput(i)) - 17))))
    ElseIf Val(PinOutput(i)) > 8 Then
      Data1 = Data1 And (Not (CByte(2 ^ (Val(PinOutput(i)) - 9))))
    Else
      Data = Data And (Not (CByte(2 ^ (Val(PinOutput(i)) - 1))))
    End If
Else ' "Pin20"
  If Val(PinOutput(i)) >= 15 Then
      Data2 = Data2 And (Not (CByte(2 ^ (Val(PinOutput(i)) - 15))))
    ElseIf Val(PinOutput(i)) >= 7 Then
        Data1 = Data1 And (Not (CByte(2 ^ (Val(PinOutput(i)) - 7))))
    Else
      Data = Data And (Not (CByte(2 ^ (Val(PinOutput(i)) + 1))))
    End If
End If
Next
If CmdSelectPin.Caption = "Pin 24" Then
    Data2 = Data2 And (Not (CByte(2 ^ 7)))
Else ' "Pin20"
```

```
Data2 = Data2 Or 2 ^ 7
    End If
  Call Sendout(Val(&H70), Val(Data))
    Data1 = Data1 Or 8
  Call Sendout(Val(&H72), Val(Data1))
  Call Sendout(Val(&H74), Val(Data2))
    DoEvents
  Sleep 300
    Data1 = Data1 And 247
  Call Sendout(Val(&H72), Val(Data1))
  DoEvents
  Sleep 300
  CmdScanFn.Enabled = True
End Sub

Private Function RepeatedTerm(ByRef ArgArray() As String) As Boolean
Dim i As Integer, j As Integer
For i = 0 To UBound(ArgArray)
    For j = i + 1 To UBound(ArgArray)
        If ArgArray(i) = ArgArray(j) Then
            RepeatedTerm = True
            Exit Function
        End If
    Next j
Next i
End Function

Private Function InputSameAsOutput() As Boolean
Dim i As Integer, j As Integer
For i = 0 To UBound(PinInput)
    For j = 0 To UBound(PinOutput)
        If PinInput(i) = PinOutput(j) Then
            InputSameAsOutput = True
            Exit Function
        End If
    Next j
Next i
End Function

Private Sub CmdIO_Click(Index As Integer)
```

```
Dim Address As Integer
Dim KeepIndex As Integer
CmdReset.Enabled = True
KeepIndex = Index
If CmdSelectPin.Caption = "Pin 24" Then
   If Index >= 16 Then
      Address = &H74        ' Address For #3
   ElseIf Index >= 8 Then
      Address = &H72        ' Address For #2
   Else
      Address = &H70        ' Address For #1
   End If
   If Index >= 16 Then
      Index = Index - 16   ' Address For #3
   ElseIf Index >= 8 Then
      Index = Index - 8   ' Address For #2       ' Address For #1 default
   End If
Else
   If Index >= 18 Then
      Address = &H74        ' Address For #3
      Index = Index - 18 ' Address For #3
   ElseIf Index >= 6 Then
      Address = &H72        ' Address For #2
      If Index >= 14 Then
         Index = Index - 10   ' Address For #2
      Else
         Index = Index - 6   ' Address For #2
      End If
   Else
   Address = &H70        ' Address For #1
   Index = Index + 2   ' Address For #1
   End If
End If
 Call I2CStart                  'Start
 Call Send8BIT(Val(Address Or 1))    'Control Word Read
 Call Ack                       'Acknowledge
 Data = dat                     'Read Data
 Call Ack                       'Acknowledge
 Call I2CStop                   'Stop
 Data = Data Or (2 ^ Index)
```

```
    If Address = &H72 Then
        Data = Data And 247
    End If
Call Sendout(Val(Address), Val(Data))
Call ReadDataToArray(False, 0)
Call DrawGraph
End Sub

Private Function AtLeastOneTermUsed(ByRef TermsInGroup As TermsGroups) As Boolean
Dim grp As Group, trm As Term
For Each grp In TermsInGroup
    For Each trm In grp.Terms
        If trm.IsUsed = True Then AtLeastOneTermUsed = True
        Exit For
    Next
Next
Set grp = Nothing: Set trm = Nothing
End Function

Private Function CompareTerms(ByRef Term1 As Term, ByRef Term2 As Term) As Term
'Pre:  Two objects of type Term are passed that contain 1 difference
Dim newTerm As New Term
Dim strTmpRep As String
Dim i As Integer
If Not Term1.Has1DifferenceWith(Term2.Representation) Then Exit Function
newTerm.DefineBasedOnObject Term1
For i = 1 To Len(Term1.Representation)
    If Mid(Term1.Representation, i, 1) <> Mid(Term2.Representation, i, 1) Then
        strTmpRep = newTerm.Representation
        Mid$(strTmpRep, i, 1) = "-"
        Term1.IsUsed = True: Term2.IsUsed = True
        newTerm.Representation = strTmpRep
        newTerm.Ones = CountOnes(newTerm.Representation)
        Set CompareTerms = newTerm
        Exit For
    End If
Next i
Set newTerm = Nothing
End Function
```

```
Function Resort(ByRef StoreInArray() As String)
Dim i As Integer
Dim j As Integer
Dim KeepData As String
For i = 0 To UBound(StoreInArray)
   For j = i + 1 To UBound(StoreInArray)
   If Val(StoreInArray(i)) > Val(StoreInArray(j)) Then
   KeepData = StoreInArray(i)
   StoreInArray(i) = StoreInArray(j)
   StoreInArray(j) = KeepData
   End If
   Next
Next
End Function

Function QM(Minterm As String, iNum As Integer)
Dim i As Integer
Dim Str As String
Dim Arr() As String
GetInputOutput Minterms, TrimS(Minterm)
GetInputOutput DontCares, TrimS("")
ReDim Arr(iNum + 1) As String
Str = ""
For i = 1 To iNum
Arr(i) = Chr(64 + i)
Next
For i = iNum To 1 Step -1
Str = Str & Arr(i) & " "   '64
Next
i = 0
GetInputOutput Variables, TrimS(Str)
Call MergeIntoAllTerms
MousePointer = vbHourglass
DoEvents
Dim Groups() As TermsGroups, CurrentGroup As Integer 'array index
Dim xGroup As Group
Dim xTerm As Term
ReDim Groups(1) As TermsGroups  'Initialize one group to store the initial terms
Set Groups(CurrentGroup) = New TermsGroups
Dim strRep As String
```

```
Dim NumOnes As Integer
'in the terms' binary representation
Dim CreatedGroupsNumOf1 As String
For i = 0 To UBound(AllTerms)
    strRep = DecToBin(Val(AllTerms(i)), UBound(Variables) + 1)
  AllTerms(i) = strRep
  NumOnes = CountOnes(strRep)
  If InStrRev(stringcheck:=CreatedGroupsNumOf1, stringmatch:=NumOnes, compare:=vbTextCompare) Then
    Groups(CurrentGroup).Group(CStr(NumOnes)).Terms.AddTerm Representation:=strRep, Ones:=NumOnes, sKey:=strRep
  Else 'No group exists for this number of 1's, create a group
    Set xGroup = Groups(CurrentGroup).AddGroup(sKey:=CStr(NumOnes))
    xGroup.NumberOfOnes = NumOnes
    Set xTerm = xGroup.Terms.AddTerm(Representation:=strRep, Ones:=NumOnes, sKey:=strRep)
    CreatedGroupsNumOf1 = CreatedGroupsNumOf1 & NumOnes
  End If
Next i
Dim tmpGroup As Group
Do
  CurrentGroup = CurrentGroup + 1
  ReDim Preserve Groups(CurrentGroup + 1) As TermsGroups
  Set Groups(CurrentGroup) = New TermsGroups
  'Compare each group in the current set with the one next to it
  i = 0
  Do Until i + 1 > Groups(CurrentGroup - 1).GroupsCount - 1
    Do Until Groups(CurrentGroup - 1).DoesGrpExists(CStr(i)) = True _
      And Groups(CurrentGroup - 1).DoesGrpExists(CStr(i + 1)) = True
      i = i + 1
    Loop
    Set tmpGroup = CompareGroups(Group1:=Groups(CurrentGroup - 1).Group(CStr(i)), _
        Group2:=Groups(CurrentGroup - 1).Group(CStr(i + 1)))
    If tmpGroup.Terms.TermsCount <> 0 Then
      Groups(CurrentGroup).AddGroupFromGroup Original:=tmpGroup
    End If
    i = i + 1
  Loop
Loop While AtLeastOneTermUsed(TermsInGroup:=Groups(CurrentGroup - 1))
Set xGroup = Nothing
Dim PrimeImp() As String, PrimeImpIndex As Integer
For i = UBound(Groups) - 1 To 0 Step -1
  For Each tmpGroup In Groups(i)
```

```
        For Each xTerm In tmpGroup.Terms
            If Not xTerm.IsUsed Then
                ReDim Preserve PrimeImp(PrimeImpIndex) As String
                PrimeImp(PrimeImpIndex) = CStr(xTerm.Representation)
                PrimeImpIndex = PrimeImpIndex + 1
            End If
        Next
    Next
Next i
Dim PICoverCount() As Integer   'Stores how many minterm each prime implicant covers
ReDim PICoverCount(UBound(PrimeImp)) As Integer
Dim chart() As Boolean
ReDim chart(UBound(PrimeImp), UBound(Minterms)) As Boolean
Dim CoveredTerms() As Boolean
ReDim CoveredTerms(UBound(Minterms)) As Boolean
Dim j As Integer
For i = 0 To UBound(PrimeImp)
    For j = 0 To UBound(Minterms)
        If Covers(PrimeImp(i), AllTerms(j)) Then
            PICoverCount(i) = PICoverCount(i) + 1
            chart(i, j) = True
        End If
    Next j
Next i
Dim essential As Boolean, AtRow As Integer
Dim row As Integer, col As Integer
Dim Expression As String
For i = 0 To UBound(Minterms)
    For j = 0 To UBound(PrimeImp)
        If chart(j, i) = True And essential = False Then
            essential = True: AtRow = j
        ElseIf chart(j, i) = True And essential = True Then
            essential = False
            Exit For
        End If
    Next j
    If essential = True Then
        Expression = Expression & ConvertToLiteral(PrimeImp(AtRow), Variables) & " + "
        For col = 0 To UBound(Minterms)
            If chart(AtRow, col) = True Then
```

```
            CoveredTerms(col) = True
            For row = 0 To UBound(PrimeImp)
              If chart(row, col) = True Then
                chart(row, col) = False
                PICoverCount(row) = PICoverCount(row) - 1
              End If
            Next row
          End If
        Next col
        essential = False 'prepare for next pass
      End If
    Next i
    Dim CoverCount As Integer, CurPrimImp As Integer
    Dim PIWithMaxTerms As Integer, MaxTerms As Integer
    For i = 0 To UBound(CoveredTerms)
      If Not CoveredTerms(i) Then
        Do
          If Covers(PrimeImp(CurPrimImp), AllTerms(i)) Then
            If PICoverCount(CurPrimImp) > MaxTerms Then
              PIWithMaxTerms = CurPrimImp
              MaxTerms = PICoverCount(CurPrimImp)
            End If
          End If
          CurPrimImp = CurPrimImp + 1
        Loop Until CurPrimImp = UBound(PrimeImp)
        Expression = Expression & ConvertToLiteral(PrimeImp(PIWithMaxTerms), Variables) & " + "
        CoveredTerms(i) = True
        For col = 0 To UBound(Minterms)
          If chart(PIWithMaxTerms, col) = True Then
            CoveredTerms(col) = True
            For row = 0 To UBound(PrimeImp)
              If chart(row, col) = True Then
                chart(row, col) = False
                PICoverCount(row) = PICoverCount(row) - 1
              End If
            Next row
          End If
        Next col
        MaxTerms = 0
      End If
```

```
    CurPrimImp = 0
  Next i
  ShowEqu(ShowEquNum) = Mid$(Expression, 1, Len(Expression) - 3)
  ShowEquNum = ShowEquNum + 1
  MousePointer = vbNormal
  End Function

  Private Sub MergeIntoAllTerms()
  Dim i As Integer, j As Integer
  ReDim AllTerms(UBound(Minterms)) As String
  For i = 0 To UBound(Minterms)
    AllTerms(i) = Minterms(i)
  Next i
  On Error GoTo DontCareNotDimErr
  ReDim Preserve AllTerms(UBound(AllTerms) + UBound(DontCares) + 1) As String
  For j = 0 To UBound(DontCares)
    AllTerms(i) = DontCares(j)
    i = i + 1
  Next j
  DontCareNotDimErr:
  End Sub

  Private Function CompareGroups(ByRef Group1 As Group, ByRef Group2 As Group) As Group
  Dim Grp1Term As Term, Grp2Term As Term
  Dim newGroup As Group, newTerm As Term
  Set newGroup = New Group
  Dim i As Integer
  On Error GoTo KeyAlreadyExistErr
  For Each Grp1Term In Group1.Terms
    For Each Grp2Term In Group2.Terms
      Set newTerm = CompareTerms(Grp1Term, Grp2Term)
        If (Not newTerm Is Nothing) Then    'And (DatasKey <> CStr(newTerm.Representation))
        newGroup.Terms.AddTerm Representation:=newTerm.Representation, _
        Ones:=newTerm.Ones, sKey:=CStr(newTerm.Representation)
        i = i + 1
      End If
    Next
  Next
  newGroup.NumberOfOnes = newGroup.Terms.AccessFirstTerm.Ones
  Set Grp1Term = Nothing
```

```
Set Grp2Term = Nothing
Set newTerm = Nothing
'Return object reference
Set CompareGroups = newGroup
Set newGroup = Nothing
Exit Function
KeyAlreadyExistErr:
Resume Next
End Function

Private Sub CmdLo_Click(Index As Integer)
Dim Address As Integer
Dim KeepIndex As Integer
CmdReset.Enabled = True
KeepIndex = Index
If CmdSelectPin.Caption = "Pin 24" Then
   If Index >= 16 Then
      Address = &H74       ' Address For #3
   ElseIf Index >= 8 Then
      Address = &H72       ' Address For #2
   Else
      Address = &H70       ' Address For #1
   End If
   If Index >= 16 Then
      Index = Index - 16   ' Address For #3
   ElseIf Index >= 8 Then
      Index = Index - 8    ' Address For #2     ' Address For #1 default
   End If
Else
   If Index >= 18 Then
      Address = &H74       ' Address For #3
      Index = Index - 18 ' Address For #3
   ElseIf Index >= 6 Then
      Address = &H72       ' Address For #2
      If Index >= 14 Then
         Index = Index - 10   ' Address For #2
      Else
         Index = Index - 6   ' Address For #2
      End If
   Else
```

```
    Address = &H70        ' Address For #1
    Index = Index + 2   ' Address For #1
    End If
  End If
  Call I2CStart                    'Start
  Call Send8BIT(Val(Address Or 1))    'Control Word Read
  Call Ack                         'Acknowledge
  Data = dat                       'Read Data
  Call Ack                         'Acknowledge
  Call I2CStop                     'Stop
  Data = Data And (Not (2 ^ Index))
  If Address = &H72 Then  Data = Data And 247
  End If
  Call Sendout(Val(Address), Val(Data))
  Call ReadDataToArray(False, 0)
  Call DrawGraph
  End Sub

  Private Sub CmdRef_Click()
  DrawGraph
  End Sub
  Private Sub CmdScanFn_Click()
  Dim i As Long
  Dim j As Integer
  CmdReset.Enabled = True
  If CmdScanFn.Caption = "Stop Scan" Then
        CmdScanFn.Caption = "Scan IO"
        CmdScanFn.Enabled = False
        Exit Sub
  Else
     CmdScanFn.Caption = "Stop Scan"
  End If
  For i = 0 To (2 ^ ((UBound(PinInput) + 1))) - 1
  If CmdScanFn.Caption = "Scan IO" Then
     Exit Sub
  End If
    Call I2CStart                  'Start
    Call Send8BIT(&H71)              'Control Word Read
    Call Ack                       'Acknowledge
    Data = dat                     'Read Data
```

```
Call Ack                    'Acknowledge
Call I2CStop                'Stop
Call I2CStart               'Start
Call Send8BIT(&H73)             'Control Word Read
Call Ack                    'Acknowledge
Data1 = dat                 'Read Data
Call Ack                    'Acknowledge
Call I2CStop                'Stop
Call I2CStart               'Start
Call Send8BIT(&H75)         'Control Word Read
Call Ack                    'Acknowledge
Data2 = dat                 'Read Data
Call Ack                    'Acknowledge
Call I2CStop                'Stop
If CmdSelectPin.Caption = "Pin 24" Then
  For j = 0 To UBound(PinInput)
  If Val(PinInput(j)) <= 8 Then ' Ad 1
    If (i And 2 ^ j) = (2 ^ j) Then   '1
      Data = Data Or 2 ^ (Val(PinInput(j)) - 1)
    Else      ' 0
      Data = Data And (Not (CByte(2 ^ (Val(PinInput(j)) - 1))))
    End If
  ElseIf Val(PinInput(j)) <= 16 Then  ' Ad 2
    If (i And 2 ^ j) = (2 ^ j) Then   '1
      Data1 = Data1 Or 2 ^ (Val(PinInput(j)) - 9)
    Else      ' 0
      Data1 = Data1 And (Not (CByte(2 ^ (Val(PinInput(j)) - 9))))
    End If
  Else   ' Ad 3
    If (i And 2 ^ j) = (2 ^ j) Then   '1
      Data2 = Data2 Or 2 ^ (Val(PinInput(j)) - 17)
    Else      ' 0
      Data2 = Data2 And (Not (CByte(2 ^ (Val(PinInput(j)) - 17))))
    End If
  End If
  Next
Else   ' 20 Pin
  For j = 0 To UBound(PinInput)
  If Val(PinInput(j)) <= 6 Then ' Ad 1
    If (i And 2 ^ j) = (2 ^ j) Then   '1
```

```
        Data = Data Or 2 ^ (Val(PinInput(j)) + 1)
      Else      ' 0
        Data = Data And (Not (CByte(2 ^ (Val(PinInput(j)) + 1))))
      End If
    ElseIf Val(PinInput(j)) <= 14 Then   ' Ad 2
      If (i And 2 ^ j) = (2 ^ j) Then    '1
        Data1 = Data1 Or 2 ^ (Val(PinInput(j)) - 7)
      Else      ' 0
        Data1 = Data1 And (Not (CByte(2 ^ (Val(PinInput(j)) - 7))))
      End If
    Else   ' Ad 3
      If (i And 2 ^ j) = (2 ^ j) Then    '1
        Data2 = Data2 Or 2 ^ (Val(PinInput(j)) - 15)
      Else      ' 0
        Data2 = Data2 And (Not (CByte(2 ^ (Val(PinInput(j)) - 15))))
      End If
    End If
    Next
  End If
    If CmdSelectPin.Caption = "Pin 24" Then
      Data2 = Data2 And (Not (CByte(2 ^ 7)))
    Else ' "Pin20"
      Data2 = Data2 Or 2 ^ 7
    End If
      Data1 = Data1 And 247
    Call Sendout(Val(&H70), Val(Data))
    Call Sendout(Val(&H72), Val(Data1))
    Call Sendout(Val(&H74), Val(Data2))
    DoEvents
    Sleep 200
    Call ReadDataToArray(True, i)
    Call DrawGraph
  Next
  CmdScanFn.Caption = "Scan IO"
  CmdScanFn.Enabled = False
  CmdCal.Enabled = True
  End Sub

  Private Sub CmdSelectPin_Click()
  Dim i As Integer
```

```
Dim j As Integer
    Call I2CStart                   'Start
    Call Send8BIT(Val(&H75))    'Control Word Read
    Call Ack                        'Acknowledge
    Data = dat                      'Read Data
    Call Ack                        'Acknowledge
    Call I2CStop                    'Stop
If CmdSelectPin.Caption = "Pin 24" Then
    CmdSelectPin.Caption = "Pin 20"
    Frame1(17).Visible = False
    Frame1(7).Visible = False
    Frame1(0).Visible = False
    Frame1(21).Visible = False
    CmdIO(11).Enabled = True
    j = 1
    For i = 0 To 23
      If i = 10 Or i = 11 Or i = 12 Or i = 13 Then
          j = j - 1
          GoTo NextPin200
      Else
        If i = 9 Then
          CmdIO(i).Caption = "Gnd"
          CmdIO(i).Enabled = False
        ElseIf i = 23 Then
          CmdIO(i).Caption = "Vcc"
          CmdIO(i).Enabled = False
        Else
          CmdIO(i).Caption = j
        End If
      End If
  NextPin200:
      If i > 21 Then
        GoTo NextPin20
      End If
      If i > 8 Then
        LbPin(i).Caption = "ขา " & i + 2
      Else
        LbPin(i).Caption = "ขา " & i + 1
      End If
      If i > 17 Then
```

```
            LbPin(i).Visible = False
        End If
NextPin20:
        j = j + 1
    Next
    Data = Data Or 128
Else
    CmdSelectPin.Caption = "Pin 24"
    Frame1(17).Visible = True
    Frame1(7).Visible = True
    Frame1(0).Visible = True
    Frame1(21).Visible = True
    CmdIO(9).Enabled = True
    For i = 0 To 23
        If i = 11 Then
            CmdIO(i).Caption = "Gnd"
            CmdIO(i).Enabled = False
        ElseIf i = 23 Then
            CmdIO(i).Caption = "Vcc"
            CmdIO(i).Enabled = False
        Else
            CmdIO(i).Caption = i + 1
        End If

        If i > 21 Then
            GoTo NextPin24
        End If
        If i > 10 Then
            LbPin(i).Caption = "ขา " & i + 2
        Else
            LbPin(i).Caption = "ขา " & i + 1
        End If
        If i > 17 Then
            LbPin(i).Visible = True
        End If
NextPin24:
    Next
    Data = Data And 127
End If
Call Sendout(Val(&H74), Val(Data))
```

```
End Sub

Private Sub CmdTimer_Click()
If CmdTimer.Caption = "Stop Timer" Then
   CmdTimer.Caption = "Timer"
   Timer2.Enabled = True
ElseIf CmdTimer.Caption = "Timer" Then
   CmdTimer.Caption = "Stop Timer"
   Timer2.Enabled = False
End If
End Sub

Private Sub CmdReset_Click()
Dim i As Integer
Graph.Refresh
CountDataColumIndex = 0
For i = 1 To 24
   A(i, 0) = False
Next
WidthTime = 0.6
HightTime = 0.4
End Sub
Private Sub ScrollGraph_Change()
  Graph.Left = -ScrollGraph.value
  Call DrawGraph
End Sub

Private Sub ScrollZoom_Change()
Dim ScrollChangeValue As Integer
ScrollChangeValue = Abs(KeepOldScroll - ScrollZoom.value)
If KeepOldScroll < ScrollZoom.value Then
      WidthTime = WidthTime + ScrollChangeValue / 20
Else
      WidthTime = WidthTime - ScrollChangeValue / 20
End If
KeepOldScroll = ScrollZoom.value
Call DrawGraph
End Sub
Function DrawGraph()
Dim Xaxis As Single
```

```
Dim Range As Single
Dim Yaxis As Single
Dim i As Integer
Dim j As Integer
Graph.Width = WidthTime * CountDataColumIndex.
  ScrollGraph.Max = Graph.Width - Graph1.Width + 2
 ScrollGraph.Visible = (Graph.Width > Graph1.Width)
Range = 0.3
Graph.Refresh
Graph1.DrawWidth = 1
Yaxis = HightTime + 0.2
If CmdSelectPin.Caption = "Pin 24" Then
   j = 22
Else
   j = 18
End If
 For i = 1 To j
   Graph1.Line (0, Yaxis)-(0.1, Yaxis), RGB(255, 255, 255)
   Graph1.Line (0, Yaxis + HightTime)-(0.1, Yaxis + HightTime), RGB(255, 255, 255)
  Yaxis = Yaxis + HightTime + 0.3
 Next
Graph.DrawWidth = 1
Xaxis = WidthTime
For i = 1 To CountDataColumIndex
   Graph.Width = Graph.Width + WidthTime
   Graph.Line (Xaxis, 0)-(Xaxis, Graph.Height), RGB(250, 250, 250)
   Xaxis = Xaxis + WidthTime
Next
For IndexDataRow = 1 To 24
   If CmdSelectPin.Caption = "Pin 20" Then
      If IndexDataRow = 1 Or IndexDataRow = 2 Or IndexDataRow = 12 Or IndexDataRow = 22 Or IndexDataRow = 23 Or
IndexDataRow = 24 Then
         GoTo NextFor
      End If
   Else
      If IndexDataRow = 12 Then
      GoTo NextFor
      End If
   End If
   If IndexDataRow = 24 Then
```

```
    GoTo NextFor
  End If
  x = 0
 Range = HightTime + Range + 0.3
  y = Range
  If A(IndexDataRow, CountDataColumIndex) = True Then
    End If
  Graph.DrawWidth = 2
  For IndexDataColum = 1 To CountDataColumIndex Step 1
    If A(IndexDataRow, IndexDataColum) = False Then
      If A(IndexDataRow, IndexDataColum - 1) = False Then
        Call HorLine(x, y)
      Else
   Call DownToUp(x, y)
      End If
      Else
      If A(IndexDataRow, IndexDataColum - 1) = True Then
        Call HorLine(x, y)
      Else
    Call UpToDown(x, y)
      End If
     End If
  Next
NextFor:
Next
End Function

Function UpToDown(x1 As Single, y1 As Single)
Graph.Line (x1, y1)-(x1, y1 - HightTime), RGB(120, 255, 0)
Graph.Line (x1, y1 - HightTime)-(x1 + WidthTime, y1 - HightTime), RGB(120, 255, 0)
x = x1 + WidthTime
y = y1 - HightTime
End Function
Function DownToUp(x1 As Single, y1 As Single)
Graph.Line (x1, y1)-(x1, y1 + HightTime), RGB(120, 255, 0)
   Graph.Line (x1, y1 + HightTime)-(x1 + WidthTime, y1 + HightTime), RGB(120, 255, 0)
x = x1 + WidthTime
y = y1 + HightTime
End Function
```

```
Function HorLine(x1 As Single, y1 As Single)
Graph.Line (x1, y1)-(x1 + WidthTime, y1), RGB(120, 255, 0)
x = x1 + WidthTime
End Function

Function ReadDataToArray(ReadOutputIs1 As Boolean, ValueScan As Long)
Dim i As Integer
CountDataColumIndex = CountDataColumIndex + 1
ReDim Preserve A(1 To 24, CountDataColumIndex) As Boolean
Call I2CStart
Call Send8BIT(&H90)
Call Ack
Call Send8BIT(&H45)
Call Ack
Call I2CStop
Call I2CStart
Call Send8BIT(&H91)
Call Ack
Data = dat
   Call MAck
   If ((dat * 5) / 255) >= 2.2 Then
      A(2, CountDataColumIndex) = True
   Else
      A(2, CountDataColumIndex) = False
   End If
   Call MAck
      If ((dat * 5) / 255) >= 2.2 Then
      A(3, CountDataColumIndex) = True
   Else
      A(3, CountDataColumIndex) = False
   End If
   Call MAck
      If ((dat * 5) / 255) >= 2.2 Then
      A(4, CountDataColumIndex) = True
   Else
      A(4, CountDataColumIndex) = False
   End If
Call Ack
Call I2CStop
Call I2CStart
```

```
Call Send8BIT(&H91)
Call Ack
  If ((dat * 5) / 255) >= 2.2 Then
    A(1, CountDataColumIndex) = True
  Else
    A(1, CountDataColumIndex) = False
  End If
Call Ack
Call I2CStop
Call I2CStart
Call Send8BIT(&H92)
Call Ack
Call Send8BIT(&H45)
Call Ack
Call I2CStop
Call I2CStart
Call Send8BIT(&H93)
Call Ack
Data = dat
  Call MAck
    If ((dat * 5) / 255) >= 2.2 Then
    A(6, CountDataColumIndex) = True
  Else
    A(6, CountDataColumIndex) = False
  End If
  Call MAck
    If ((dat * 5) / 255) >= 2.2 Then
    A(7, CountDataColumIndex) = True
  Else
    A(7, CountDataColumIndex) = False
  End If
  Call MAck
    If ((dat * 5) / 255) >= 2.2 Then
    A(8, CountDataColumIndex) = True
  Else
    A(8, CountDataColumIndex) = False
  End If
Call Ack
Call I2CStop
Call I2CStart
```

```
Call Send8BIT(&H93)
Call Ack
  If ((dat * 5) / 255) >= 2.2 Then
    A(5, CountDataColumIndex) = True
  Else
    A(5, CountDataColumIndex) = False
  End If
Call Ack
Call I2CStop
Call I2CStart
Call Send8BIT(&H94)
Call Ack
Call Send8BIT(&H45)
Call Ack
Call I2CStop
Call I2CStart
Call Send8BIT(&H95)
Call Ack
Data = dat
Call MAck
  If ((dat * 5) / 255) >= 2.2 Then
    A(10, CountDataColumIndex) = True
  Else
    A(10, CountDataColumIndex) = False
  End If
  Call MAck
  If ((dat * 5) / 255) >= 2.2 Then
    A(11, CountDataColumIndex) = True
  Else
    A(11, CountDataColumIndex) = False
  End If
  Call MAck
  If ((dat * 5) / 255) >= 2.2 Then
    A(12, CountDataColumIndex) = True
  Else
    A(12, CountDataColumIndex) = False
  End If
Call Ack
Call I2CStop
Call I2CStart
```

```
Call Send8BIT(&H95)
Call Ack
  If ((dat * 5) / 255) >= 2.2 Then
    A(9, CountDataColumIndex) = True
  Else
    A(9, CountDataColumIndex) = False
  End If
 Call Ack
 Call I2CStop
 Call I2CStart
 Call Send8BIT(&H96)
 Call Ack
 Call Send8BIT(&H45)
 Call Ack
 Call I2CStop
 Call I2CStart
 Call Send8BIT(&H97)
 Call Ack
 Data = dat
 Call MAck
   If ((dat * 5) / 255) >= 2.2 Then
     A(14, CountDataColumIndex) = True
   Else
     A(14, CountDataColumIndex) = False
   End If
   Call MAck
     If ((dat * 5) / 255) >= 2.2 Then
     A(15, CountDataColumIndex) = True
    Else
     A(15, CountDataColumIndex) = False
    End If
    Call MAck
      If ((dat * 5) / 255) >= 2.2 Then
      A(16, CountDataColumIndex) = True
     Else
      A(16, CountDataColumIndex) = False
     End If
  Call Ack
  Call I2CStop
  Call I2CStart
```

```
Call Send8BIT(&H97)
Call Ack
  If ((dat * 5) / 255) >= 2.2 Then
    A(13, CountDataColumIndex) = True
  Else
    A(13, CountDataColumIndex) = False
  End If
Call Ack
Call I2CStop
Call I2CStart
Call Send8BIT(&H98)
Call Ack
Call Send8BIT(&H45)
Call Ack
Call I2CStop
Call I2CStart
Call Send8BIT(&H99)
Call Ack
Data = dat
Call MAck
  If ((dat * 5) / 255) >= 2.2 Then
    A(18, CountDataColumIndex) = True
  Else
    A(18, CountDataColumIndex) = False
  End If
  Call MAck
    If ((dat * 5) / 255) >= 2.2 Then
    A(19, CountDataColumIndex) = True
  Else
    A(19, CountDataColumIndex) = False
  End If
  Call MAck
    If ((dat * 5) / 255) >= 2.2 Then
    A(20, CountDataColumIndex) = True
  Else
    A(20, CountDataColumIndex) = False
  End If
Call Ack
Call I2CStop
Call I2CStart
```

```
Call Send8BIT(&H99)
Call Ack
  If ((dat * 5) / 255) >= 2.2 Then
    A(17, CountDataColumIndex) = True
  Else
    A(17, CountDataColumIndex) = False
  End If
Call Ack
Call I2CStop
Call I2CStart
Call Send8BIT(&H9A)
Call Ack
Call Send8BIT(&H45)
Call Ack
Call I2CStop
Call I2CStart
Call Send8BIT(&H9B)
Call Ack
Data = dat
Call MAck
  If ((dat * 5) / 255) >= 2.2 Then
    A(22, CountDataColumIndex) = True
  Else
    A(22, CountDataColumIndex) = False
  End If
  Call MAck
    If ((dat * 5) / 255) >= 2.2 Then
    A(23, CountDataColumIndex) = True
  Else
    A(23, CountDataColumIndex) = False
  End If
  Call MAck
    If ((dat * 5) / 255) >= 2.2 Then
    A(24, CountDataColumIndex) = True
  Else
    A(24, CountDataColumIndex) = False
  End If
Call Ack
Call I2CStop
Call I2CStart
```

```
Call Send8BIT(&H9B)
Call Ack
  If ((dat * 5) / 255) >= 2.2 Then
    A(21, CountDataColumIndex) = True
  Else
    A(21, CountDataColumIndex) = False
  End If
Call Ack
Call I2CStop
If ReadOutputIs1 = True Then
  If CmdSelectPin.Caption = "Pin 24" Then
    For i = 0 To UBound(PinOutput)
      If A(Val(PinOutput(i)), CountDataColumIndex) = True Then
        KeepPinOutputIs1(Val(PinOutput(i))) = KeepPinOutputIs1(Val(PinOutput(i))) & CStr(ValueScan) & " "
      End If
    Next
  Else   'Pin 20
    For i = 0 To UBound(PinOutput)
      If A((Val(PinOutput(i)) + 2), CountDataColumIndex) = True Then
        KeepPinOutputIs1(Val(PinOutput(i))) = KeepPinOutputIs1(Val(PinOutput(i))) & CStr(ValueScan) & " "
      End If
    Next
  End If
End If
End Function

Private Sub CmdExit_Click()
  Unload Me
  Set frmMc = Nothing
  End
End Sub

Private Sub Timer2_Timer()
Timer2.Enabled = False
If CmdSelectPin.Caption = "Pin 24" Then
Call I2CStart
Call Send8BIT(&H90)
Call Ack
Call Send8BIT(&H45)
Call Ack
```

```
Call I2CStop
Call I2CStart
Call Send8BIT(&H91)
Call Ack
Output(0).Text = Format((dat * 5) / 255, "#.#")
If Val(Output(0).Text) >= 2.2 Then
Red(0).Visible = True
Black(0).Visible = False
Else
Red(0).Visible = False
Black(0).Visible = True
End If
Call MAck
Output(1).Text = Format((dat * 5) / 255, "#.#")
If Val(Output(1).Text) >= 2.2 Then
Red(1).Visible = True
Black(1).Visible = False
Else
Red(1).Visible = False
Black(1).Visible = True
End If
Call MAck
Output(2).Text = Format((dat * 5) / 255, "#.#")
If Val(Output(2).Text) >= 2.2 Then
Red(2).Visible = True
Black(2).Visible = False
Else
Red(2).Visible = False
Black(2).Visible = True
End If
Call MAck
Output(3).Text = Format((dat * 5) / 255, "#.#")
If Val(Output(3).Text) >= 2.2 Then
Red(3).Visible = True
Black(3).Visible = False
Else
Red(3).Visible = False
Black(3).Visible = True
End If
Call Ack
```

```
Call I2CStop
Call I2CStart
Call Send8BIT(&H92)
Call Ack
Call Send8BIT(&H45)
Call Ack
Call I2CStop
Call I2CStart
Call Send8BIT(&H93)
Call Ack
Output(4).Text = Format((dat * 5) / 255, "#.#")
If Val(Output(4).Text) >= 2.2 Then
Red(4).Visible = True
Black(4).Visible = False
Else
Red(4).Visible = False
Black(4).Visible = True
End If
Call MAck
Output(5).Text = Format((dat * 5) / 255, "#.#")
If Val(Output(5).Text) >= 2.2 Then
Red(5).Visible = True
Black(5).Visible = False
Else
Red(5).Visible = False
Black(5).Visible = True
End If
Call MAck
Output(6).Text = Format((dat * 5) / 255, "#.#")
If Val(Output(6).Text) >= 2.2 Then
Red(6).Visible = True
Black(6).Visible = False
Else
Red(6).Visible = False
Black(6).Visible = True
End If
Call MAck
Output(7).Text = Format((dat * 5) / 255, "#.#")
If Val(Output(7).Text) >= 2.2 Then
Red(7).Visible = True
```

```
Black(7).Visible = False
Else
Red(7).Visible = False
Black(7).Visible = True
End If
Call Ack
Call I2CStop
Call I2CStart
Call Send8BIT(&H94)
Call Ack
Call Send8BIT(&H45)
Call Ack
Call I2CStop
Call I2CStart
Call Send8BIT(&H95)
Call Ack
Output(8).Text = Format((dat * 5) / 255, "#.#")
If Val(Output(8).Text) >= 2.2 Then
Red(8).Visible = True
Black(8).Visible = False
Else
Red(8).Visible = False
Black(8).Visible = True
End If
Call MAck
Output(9).Text = Format((dat * 5) / 255, "#.#")
If Val(Output(9).Text) >= 2.2 Then
Red(9).Visible = True
Black(9).Visible = False
Else
Red(9).Visible = False
Black(9).Visible = True
End If
Call MAck
Output(10).Text = Format((dat * 5) / 255, "#.#")
If Val(Output(10).Text) >= 2.2 Then
Red(10).Visible = True
Black(10).Visible = False
Else
Red(10).Visible = False
```

```
Black(10).Visible = True
End If
Call MAck
Output(11).Text = Format((dat * 5) / 255, "#.#")
Red(11).Visible = False
Black(11).Visible = True
Call Ack
Call I2CStop
Call I2CStart
Call Send8BIT(&H96)
Call Ack
Call Send8BIT(&H45)
Call Ack
Call I2CStop
Call I2CStart
Call Send8BIT(&H97)
Call Ack
Output(12).Text = Format((dat * 5) / 255, "#.#")
If Val(Output(12).Text) >= 2.2 Then
Red(12).Visible = True
Black(12).Visible = False
Else
Red(12).Visible = False
Black(12).Visible = True
End If
Call MAck
Output(13).Text = Format((dat * 5) / 255, "#.#")
If Val(Output(13).Text) >= 2.2 Then
Red(13).Visible = True
Black(13).Visible = False
Else
Red(13).Visible = False
Black(13).Visible = True
End If
Call MAck
Output(14).Text = Format((dat * 5) / 255, "#.#")
If Val(Output(14).Text) >= 2.2 Then
Red(14).Visible = True
Black(14).Visible = False
Else
```

```
Red(14).Visible = False
Black(14).Visible = True
End If
Call MAck
Output(15).Text = Format((dat * 5) / 255, "#.#")
If Val(Output(15).Text) >= 2.2 Then
Red(15).Visible = True
Black(15).Visible = False
Else
Red(15).Visible = False
Black(15).Visible = True
End If
Call Ack
Call I2CStop
Call I2CStart
Call Send8BIT(&H98)
Call Ack
Call Send8BIT(&H45)
Call Ack
Call I2CStop
Cail I2CStart
Call Send8BIT(&H99)
Call Ack
Output(16).Text = Format((dat * 5) / 255, "#.#")
If Val(Output(16).Text) >= 2.2 Then
Red(16).Visible = True
Black(16).Visible = False
Else
Red(16).Visible = False
Black(16).Visible = True
End If
Call MAck
Output(17).Text = Format((dat * 5) / 255, "#.#")
If Val(Output(17).Text) >= 2.2 Then
Red(17).Visible = True
Black(17).Visible = False
Else
Red(17).Visible = False
Black(17).Visible = True
End If
```

```
Call MAck
Output(18).Text = Format((dat * 5) / 255, "#.#")
If Val(Output(18).Text) >= 2.2 Then
Red(18).Visible = True
Black(18).Visible = False
Else
Red(18).Visible = False
Black(18).Visible = True
End If
Call MAck
Output(19).Text = Format((dat * 5) / 255, "#.#")
If Val(Output(19).Text) >= 2.2 Then
Red(19).Visible = True
Black(19).Visible = False
Else
Red(19).Visible = False
Black(19).Visible = True
End If
Call Ack
Call I2CStop
Call I2CStart
Call Send8BIT(&H9A)
Call Ack
Call Send8BIT(&H45)
Call Ack
Call I2CStop
Call I2CStart
Call Send8BIT(&H9B)
Call Ack
Output(20).Text = Format((dat * 5) / 255, "#.#")
If Val(Output(20).Text) >= 2.2 Then
Red(20).Visible = True
Black(20).Visible = False
Else
Red(20).Visible = False
Black(20).Visible = True
End If
Call MAck
Output(21).Text = Format((dat * 5) / 255, "#.#")
If Val(Output(21).Text) >= 2.2 Then
```

```
Red(21).Visible = True
Black(21).Visible = False
Else
Red(21).Visible = False
Black(21).Visible = True
End If
Call MAck
Output(22).Text = Format((dat * 5) / 255, "#.#")
If Val(Output(22).Text) >= 2.2 Then
Red(22).Visible = True
Black(22).Visible = False
Else
Red(22).Visible = False
Black(22).Visible = True
End If
Call MAck
Output(23).Text = Format((dat * 5) / 255, "#.#")
If Val(Output(23).Text) >= 2.2 Then
Red(23).Visible = True
Black(23).Visible = False
Else
Red(23).Visible = False
Black(23).Visible = True
End If
Call Ack
Call I2CStop
Call I2CStart                'Start
Call Send8BIT(&H71)          'Control Word Read
Call Ack                     'Acknowledge
Call Ack                     'Acknowledge
Call I2CStop                 'Stop
Call I2CStart                'Start
Call Send8BIT(&H73)          'Control Word Read
Call Ack                     'Acknowledge
Call Ack                     'Acknowledge
Call I2CStop                 'Stop
Call I2CStart                'Start
Call Send8BIT(&H75)          'Control Word Read
Call Ack                     'Acknowledge
Call Ack                     'Acknowledge
```

```
Call I2CStop
  Else
Call I2CStart
Call Send8BIT(&H90)
Call Ack
Call Send8BIT(&H45)
Call Ack
Call I2CStop
Call I2CStart
Call Send8BIT(&H91)
Call Ack
Data = dat
Call MAck
Data = dat
Call MAck
Output(0).Text = Format((dat * 5) / 255, "#.#")
If Val(Output(0).Text) >= 2.2 Then
Red(0).Visible = True
Black(0).Visible = False
Else
Red(0).Visible = False
Black(0).Visible = True
End If
Call MAck
Output(1).Text = Format((dat * 5) / 255, "#.#")
If Val(Output(1).Text) >= 2.2 Then
Red(1).Visible = True
Black(1).Visible = False
Else
Red(1).Visible = False
Black(1).Visible = True
End If
Call Ack
Call I2CStop
Call I2CStart
Call Send8BIT(&H92)
Call Ack
Call Send8BIT(&H45)
Call Ack
Call I2CStop
```

```
Call I2CStart
Call Send8BIT(&H93)
Call Ack
Output(2).Text = Format((dat * 5) / 255, "#.#")
If Val(Output(2).Text) >= 2.2 Then
Red(2).Visible = True
Black(2).Visible = False
Else
Red(2).Visible = False
Black(2).Visible = True
End If
Call MAck
Output(3).Text = Format((dat * 5) / 255, "#.#")
If Val(Output(3).Text) >= 2.2 Then
Red(3).Visible = True
Black(3).Visible = False
Else
Red(3).Visible = False
Black(3).Visible = True
End If
Call MAck
Output(4).Text = Format((dat * 5) / 255, "#.#")
If Val(Output(4).Text) >= 2.2 Then
Red(4).Visible = True
Black(4).Visible = False
Else
Red(4).Visible = False
Black(4).Visible = True
End If
Call MAck
Output(5).Text = Format((dat * 5) / 255, "#.#")
If Val(Output(5).Text) >= 2.2 Then
Red(5).Visible = True
Black(5).Visible = False
Else
Red(5).Visible = False
Black(5).Visible = True
End If
Call Ack
Call I2CStop
```

```
Call I2CStart
Call Send8BIT(&H94)
Call Ack
Call Send8BIT(&H45)
Call Ack
Call I2CStop
Call I2CStart
Call Send8BIT(&H95)
Call Ack
Output(6).Text = Format((dat * 5) / 255, "#.#")
If Val(Output(6).Text) >= 2.2 Then
Red(6).Visible = True
Black(6).Visible = False
Else
Red(6).Visible = False
Black(6).Visible = True
End If
Call MAck
Output(7).Text = Format((dat * 5) / 255, "#.#")
If Val(Output(7).Text) >= 2.2 Then
Red(7).Visible = True
Black(7).Visible = False
Else
Red(7).Visible = False
Black(7).Visible = True
End If
Call MAck
Output(8).Text = Format((dat * 5) / 255, "#.#")
If Val(Output(8).Text) >= 2.2 Then
Red(8).Visible = True
Black(8).Visible = False
Else
Red(8).Visible = False
Black(8).Visible = True
End If
Call MAck
Output(9).Text = Format((dat * 5) / 255, "#.#")
If Val(Output(9).Text) >= 2.2 Then
Red(9).Visible = True
Black(9).Visible = False
```

```
Else
Red(9).Visible = False
Black(9).Visible = True
End If
Call Ack
Call I2CStop
Call I2CStart
Call Send8BIT(&H96)
Call Ack
Call Send8BIT(&H45)
Call Ack
Call I2CStop
Call I2CStart
Call Send8BIT(&H97)
Call Ack
Output(14).Text = Format((dat * 5) / 255, "#.#")
If Val(Output(14).Text) >= 2.2 Then
Red(14).Visible = True
Black(14).Visible = False
Else
Red(14).Visible = False
Black(14).Visible = True
End If
Call MAck
Output(15).Text = Format((dat * 5) / 255, "#.#")
If Val(Output(15).Text) >= 2.2 Then
Red(15).Visible = True
Black(15).Visible = False
Else
Red(15).Visible = False
Black(15).Visible = True
End If
Call MAck
Output(16).Text = Format((dat * 5) / 255, "#.#")
If Val(Output(16).Text) >= 2.2 Then
Red(16).Visible = True
Black(16).Visible = False
Else
Red(16).Visible = False
Black(16).Visible = True
```

```
End If
Call MAck
Output(17).Text = Format((dat * 5) / 255, "#.#")
If Val(Output(17).Text) >= 2.2 Then
Red(17).Visible = True
Black(17).Visible = False
Else
Red(17).Visible = False
Black(17).Visible = True
End If
Call Ack
Call I2CStop
Call I2CStart
Call Send8BIT(&H98)
Call Ack
Call Send8BIT(&H45)
Call Ack
Call I2CStop
Call I2CStart
Call Send8BIT(&H99)
Call Ack
Output(18).Text = Format((dat * 5) / 255, "#.#")
If Val(Output(18).Text) >= 2.2 Then
Red(18).Visible = True
Black(18).Visible = False
Else
Red(18).Visible = False
Black(18).Visible = True
End If
Call MAck
Output(19).Text = Format((dat * 5) / 255, "#.#")
If Val(Output(19).Text) >= 2.2 Then
Red(19).Visible = True
Black(19).Visible = False
Else
Red(19).Visible = False
Black(19).Visible = True
End If
Call MAck
Output(20).Text = Format((dat * 5) / 255, "#.#")
```

```
If Val(Output(20).Text) >= 2.2 Then
Red(20).Visible = True
Black(20).Visible = False
Else
Red(20).Visible = False
Black(20).Visible = True
End If
Call MAck
Output(21).Text = Format((dat * 5) / 255, "#.#")
If Val(Output(21).Text) >= 2.2 Then
Red(21).Visible = True
Black(21).Visible = False
Else
Red(21).Visible = False
Black(21).Visible = True
End If
Call Ack
Call I2CStop
Call I2CStart
Call Send8BIT(&H9A)
Call Ack
Call Send8BIT(&H45)
Call Ack
Call I2CStop
Call I2CStart
Call Send8BIT(&H9B)
Call Ack
Output(22).Text = Format((dat * 5) / 255, "#.#")
If Val(Output(22).Text) >= 2.2 Then
Red(22).Visible = True
Black(22).Visible = False
Else
Red(22).Visible = False
Black(22).Visible = True
End If
Call MAck
Output(23).Text = Format((dat * 5) / 255, "#.#")
If Val(Output(23).Text) >= 2.2 Then
Red(23).Visible = True
Black(23).Visible = False
```

```
Else
Red(23).Visible = False
Black(23).Visible = True
End If
Call MAck
Data = dat
Call MAck
Data = dat
Call Ack
Call I2CStop
Call I2CStart                'Start
Call Send8BIT(&H71)          'Control Word Read
Call Ack                     'Acknowledge
Call Ack                     'Acknowledge
Call I2CStop                 'Stop
Call I2CStart                'Start
Call Send8BIT(&H73)          'Control Word Read
Call Ack                     'Acknowledge
Call Ack                     'Acknowledge
Call I2CStop                 'Stop
Call I2CStart                'Start
Call Send8BIT(&H75)          'Control Word Read
Call Ack                     'Acknowledge
Call Ack                     'Acknowledge
Call I2CStop
End If
Timer2.Enabled = True
End Sub

Private Sub Sendout(ADDR, B As Byte)
Call I2CStart          'Start
Call Send8BIT(Val(ADDR))    'Send Control Word
Call Ack               'Acknowledge
Call Send8BIT(Val(B))       'Send Data
Call Ack               'Acknowledge
Call I2CStop           'Stop
End Sub
Private Sub Send8BIT(Add As Byte)
 For i = 7 To 0 Step -1              ' Loop 8 Cycle
   If (Add And 2 ^ i) = 2 ^ i Then       'Test Bit 0 OR 1
```

```
      Call Send1
    Else
      Call send0
    End If
  Next i
  End Sub
 Private Function dat()
Dim dat1 As Byte
For i = 7 To 0 Step -1
   MSComm1.RTSEnable = True   'SDA=1
   MSComm1.DTREnable = True   'SCL=1
   If (MSComm1.CDHolding And &H80) <> &H80 Then
     dat1 = 2 ^ i Or dat1
   End If
   MSComm1.DTREnable = False  'SCL=0
Next i
dat = dat1
End Function

Private Sub Send1()
MSComm1.RTSEnable = True   'SDA=1
MSComm1.DTREnable = True   'SCL=1
MSComm1.DTREnable = False  'SCL=0
End Sub

Private Sub send0()
MSComm1.RTSEnable = False  'SDA=0
MSComm1.DTREnable = True   'SCL=1
MSComm1.DTREnable = False  'SCL=0
End Sub

Private Sub Ack()
MSComm1.RTSEnable = True   'SDA=1
MSComm1.DTREnable = True   'SCL=1
MSComm1.DTREnable = False  'SCL=0
End Sub

Private Sub MAck()
MSComm1.RTSEnable = False  'SDA=0
MSComm1.DTREnable = True   'SCL=1
```

```
MSComm1.DTREnable = False   'SCL=0
MSComm1.RTSEnable = True    'SDA=1
End Sub

Private Sub I2CStop()
MSComm1.RTSEnable = False   'SDA=0
MSComm1.DTREnable = True    'SCL=1
MSComm1.RTSEnable = True    'SDA=1
End Sub

Private Sub I2CStart()
MSComm1.RTSEnable = True    'SDA=1
MSComm1.DTREnable = True    'SCL=1
MSComm1.RTSEnable = False   'SDA=0
MSComm1.DTREnable = False   'SCL=0
End Sub

Private Sub Form_Load()
Dim i As Integer
Dim j As Single
ReDim Preserve A(1 To 24, 0) As Boolean
WidthTime = 0.6
HightTime = 0.4
Graph.ScaleMode = 7
Graph1.ScaleMode = 7
Graph.Move 0.2, 0
Graph.Height = Graph1.Height
Graph.BorderStyle = 0
  ScrollGraph.Top = Pic.Height 'Graph1.Height
  ScrollGraph.Left = Pic.Left
  ScrollGraph.Width = Pic.Width
  ScrollGraph.Visible = (Graph.Width > Graph1.Width)
  j = LbPin(0).Top
For i = 1 To 21
j = j + 395
LbPin(i).Top = j
Next
CmdCal.Enabled = False
CmdScanFn.Enabled = False
CmdReset.Enabled = False
```

```
Timer2.Enabled = False
MSComm1.PortOpen = True
CountDataColumIndex = 0
For i = 0 To 23
   Green(i).Visible = False
Next
   Call I2CStart                  'Start
   Call Send8BIT(Val(&H75))    'Control Word Read
   Call Ack                       'Acknowledge
   Data = dat                     'Read Data
   Call Ack                       'Acknowledge
   Call I2CStop                   'Stop
   Data = Data And 127
Call Sendout(Val(&H74), Val(Data))
End Sub
```

MODULES

```
Option Explicit
Public Function ConvertToLiteral(ByVal Rep As String, ByRef ArrVariables() As String) As String
If Len(Rep) = 0 Then
   ConvertToLiteral = ""
   Exit Function
End If
Dim i As Integer, ArrIndex As Integer
Dim Output As String, CurChar As String
For i = 1 To Len(Rep)
   CurChar = Mid$(Rep, i, 1)
   If CurChar <> "-" Then
      If CurChar = "0" Then
         Output = Output & ArrVariables(ArrIndex) & "'"
      ElseIf CurChar = "1" Then
         Output = Output & ArrVariables(ArrIndex)
      End If
   End If
   ArrIndex = ArrIndex + 1
Next i
ConvertToLiteral = RTrim$(Output)
End Function

Public Function Max(ByRef Arr() As String, Optional ByRef AtElement As Integer) As Integer
   Dim i As Integer, max_value As Integer
   max_value = 0
   For i = 0 To UBound(Arr)
      If Val(Arr(i)) > max_value Then
         max_value = Val(Arr(i))
         AtElement = i
      End If
   Next i
   Max = max_value
End Function

Public Function CountOnes(ByVal value As Variant) As Integer
   Dim i As Integer
   Dim strValue As String
   Dim TotalOnes As Integer
   strValue = CStr(value)
```

```
  For i = 1 To Len(strValue)
    If Mid$(strValue, i, 1) = "1" Then
      TotalOnes = TotalOnes + 1
    End If
  Next i
  CountOnes = TotalOnes
End Function

Public Function DecToBin(DecNum As Integer, Optional NumOfBits As Integer) As String
Dim TempValue As Integer
Dim BinValue As String
Dim BinLength As Integer
If DecNum < 0 Then DecNum = 0
Do
  TempValue = DecNum Mod 2
  BinValue = CStr(TempValue) + BinValue
  DecNum = DecNum \ 2
Loop Until DecNum = 0
Do While Len(BinValue) < NumOfBits
  BinValue = "0" & BinValue
Loop
DecToBin = BinValue
End Function

Public Function Covers(ByVal PrimeImp As String, ByVal Minterm As String) As Boolean
Covers = True
If Len(PrimeImp) = Len(Minterm) Then      'Obviously they should be of the same length to start with
  Dim i As Integer
  Dim CurPrimeChr As String
  For i = 1 To Len(PrimeImp)
    CurPrimeChr = Mid$(PrimeImp, i, 1)
    If CurPrimeChr <> "-" And CurPrimeChr <> Mid$(Minterm, i, 1) Then
      Covers = False
      Exit Function
    End If
  Next i
Else
  Covers = False
End If
End Function
```

```
Public Sub GetInputOutput(ByRef StoreInArray() As String, ByVal strText As String, Optional Delimiter As String = " ")
Dim i As Integer, j As Integer, ArrIndex As Integer
Dim KeepData As String
Dim CurChar As String, strBuffer As String
For i = 1 To Len(strText)
  CurChar = Mid$(strText, i, 1)
  If CurChar = Delimiter And strBuffer <> "" Then
    ReDim Preserve StoreInArray(ArrIndex) As String
    StoreInArray(ArrIndex) = strBuffer
    strBuffer = ""
    ArrIndex = ArrIndex + 1
  Else
    If CurChar <> Delimiter Then strBuffer = strBuffer & CurChar
  End If
Next i
If strBuffer <> "" Then    'If we finished but still have something in the buffer
  ReDim Preserve StoreInArray(ArrIndex) As String
  StoreInArray(ArrIndex) = strBuffer
End If
End Sub
```

CLASS MODULES

```
Private mvarTerms As Terms
Private mvarNumOnes As Integer
Public Property Let NumberOfOnes(ByVal Ones As Integer)
   mvarNumOnes = Ones
End Property

Public Property Get NumberOfOnes() As Integer
   NumberOfOnes = mvarNumOnes
End Property

Public Property Get Terms() As Terms
   If mvarTerms Is Nothing Then
      Set mvarTerms = New Terms
   End If
   Set Terms = mvarTerms
End Property

Public Property Set Terms(vData As Terms)
   Set mvarTerms = vData
End Property
Private Sub Class_Terminate()
   Set mvarTerms = Nothing
End Sub

Option Explicit
Private varRepresentation As Variant
Private intOnes As Integer
Private bUsed As Boolean
Public Function Has1DifferenceWith(ByVal varValue As Variant) As Boolean
   Dim strThisValue As String, strOtherValue As String
   strThisValue = CStr(varRepresentation)
   strOtherValue = CStr(varValue)
   If Abs(Len(strThisValue) - Len(strOtherValue)) > 1 Then
      Has1DifferenceWith = False
      Exit Function
   End If
      Dim i As Integer, Diff As Integer
   For i = 1 To Len(strThisValue)
      If Mid$(strThisValue, i, 1) <> Mid$(strOtherValue, i, 1) Then
```

```
Private Sub Class_Terminate()
  Set mCol = Nothing
End Sub
```

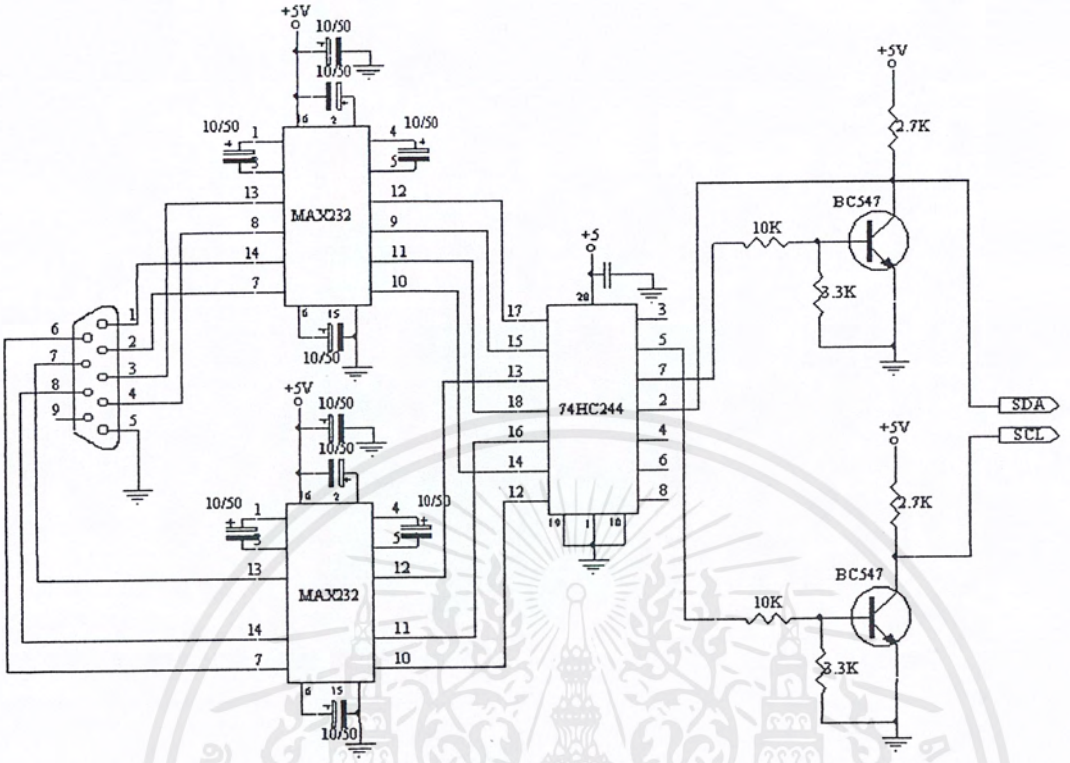
```
            Diff = Diff + 1
        End If
    Next i

    If Diff = 1 Then
        Has1DifferenceWith = True
    Else
        Has1DifferenceWith = False
    End If
End Function

Public Sub DefineBasedOnObject(ByRef Object As Term)
    varRepresentation = Object.Representation
    intOnes = CountOnes(varRepresentation)
End Sub

Public Property Let IsUsed(ByVal vData As Boolean)
    bUsed = vData
End Property

Public Property Get IsUsed() As Boolean
    IsUsed = bUsed
End Property

Public Property Let Ones(ByVal vData As Integer)
    intOnes = vData
End Property

Public Property Get Ones() As Integer
    Ones = intOnes
End Property

Public Property Let Representation(ByVal vData As Variant)
    varRepresentation = vData
End Property

Public Property Get Representation() As Variant
    Representation = varRepresentation
End Property
```

```
Private Sub Class_Initialize()
  intOnes = 0
  bUsed = False
End Sub

Option Explicit
Private mCol As Collection
Dim DatasKey As String
Public Function AddTerm(Representation As Variant, Ones As Integer, Optional sKey As String) As Term
  Dim objNewMember As Term
  Set objNewMember = New Term
  objNewMember.Representation = Representation
  objNewMember.Ones = Ones
  If Len(sKey) = 0 Then
    mCol.Add objNewMember
  Else
    If DatasKey <> sKey Then
      DatasKey = sKey
      mCol.Add objNewMember, sKey
    End If
  End If
    Set AddTerm = objNewMember
  Set objNewMember = Nothing
End Function

Public Property Get AccessTerm(vntIndexKey As Variant) As Term
  Set AccessTerm = mCol(vntIndexKey)
End Property

Public Property Get AccessFirstTerm() As Term
  Dim x As Term
  For Each x In mCol
    Set AccessFirstTerm = x
    Exit Property
  Next x
End Property

Public Property Get TermsCount() As Long
  TermsCount = mCol.Count
End Property
```

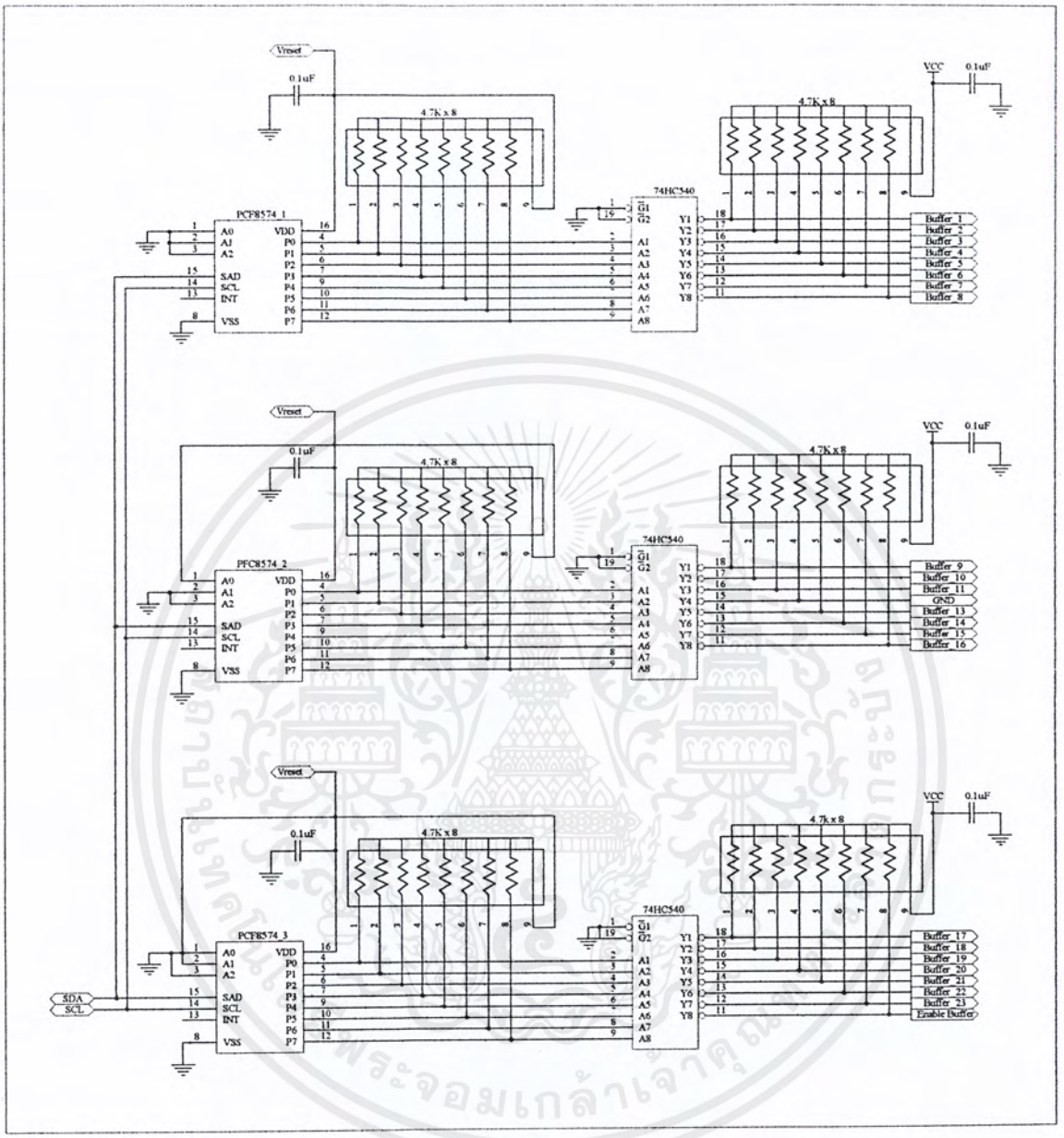
```
Public Function AddGroup(Optional sKey As String) As Group
On Error GoTo KeyAlreadyExistsError
    Dim objNewMember As Group
    Set objNewMember = New Group
    If Len(sKey) = 0 Then
        mCol.Add objNewMember
    Else
        mCol.Add objNewMember, sKey
    End If
    Set AddGroup = objNewMember
    Set objNewMember = Nothing
Exit Function
KeyAlreadyExistsError:
If Err.Number = 457 Then        'This is the error number
    Set AddGroup = mCol(sKey)
    MsgBox "error occured in add group try to add group with key = " & sKey
End If
End Function

Public Property Get Group(vntIndexKey As Variant) As Group
  Set Group = mCol(vntIndexKey)
End Property

Public Property Get GroupsCount() As Long
    GroupsCount = mCol.Count
End Property

Public Sub RemoveGroup(vntIndexKey As Variant)
    mCol.Remove vntIndexKey
End Sub

Public Property Get NewEnum() As IUnknown
    Set NewEnum = mCol.[_NewEnum]
End Property

Private Sub Class_Initialize()
    Set mCol = New Collection
End Sub
```

ภาคผนวก ข.

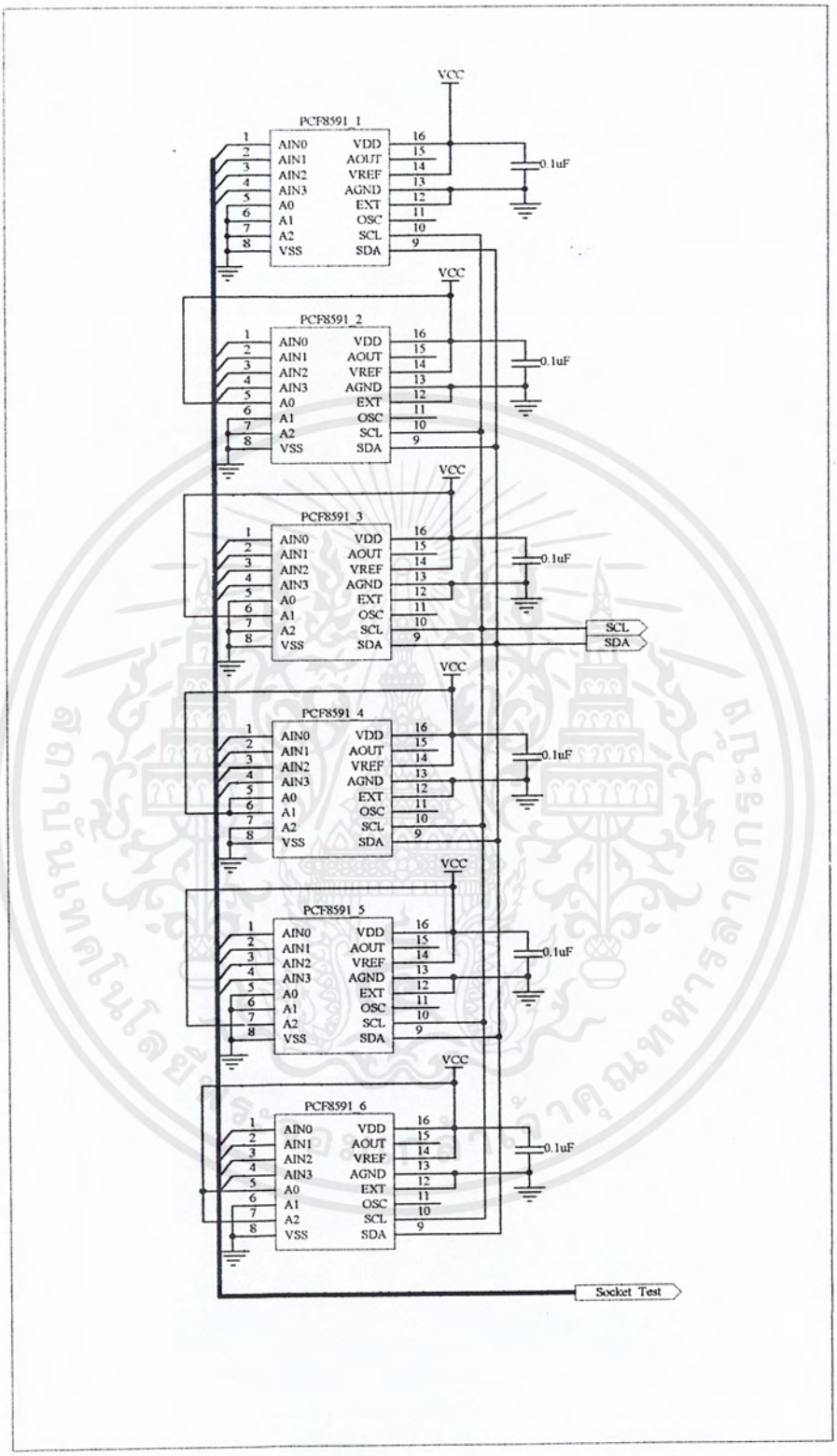
วงจรและลายทองแดงของฮาร์ดแวร์



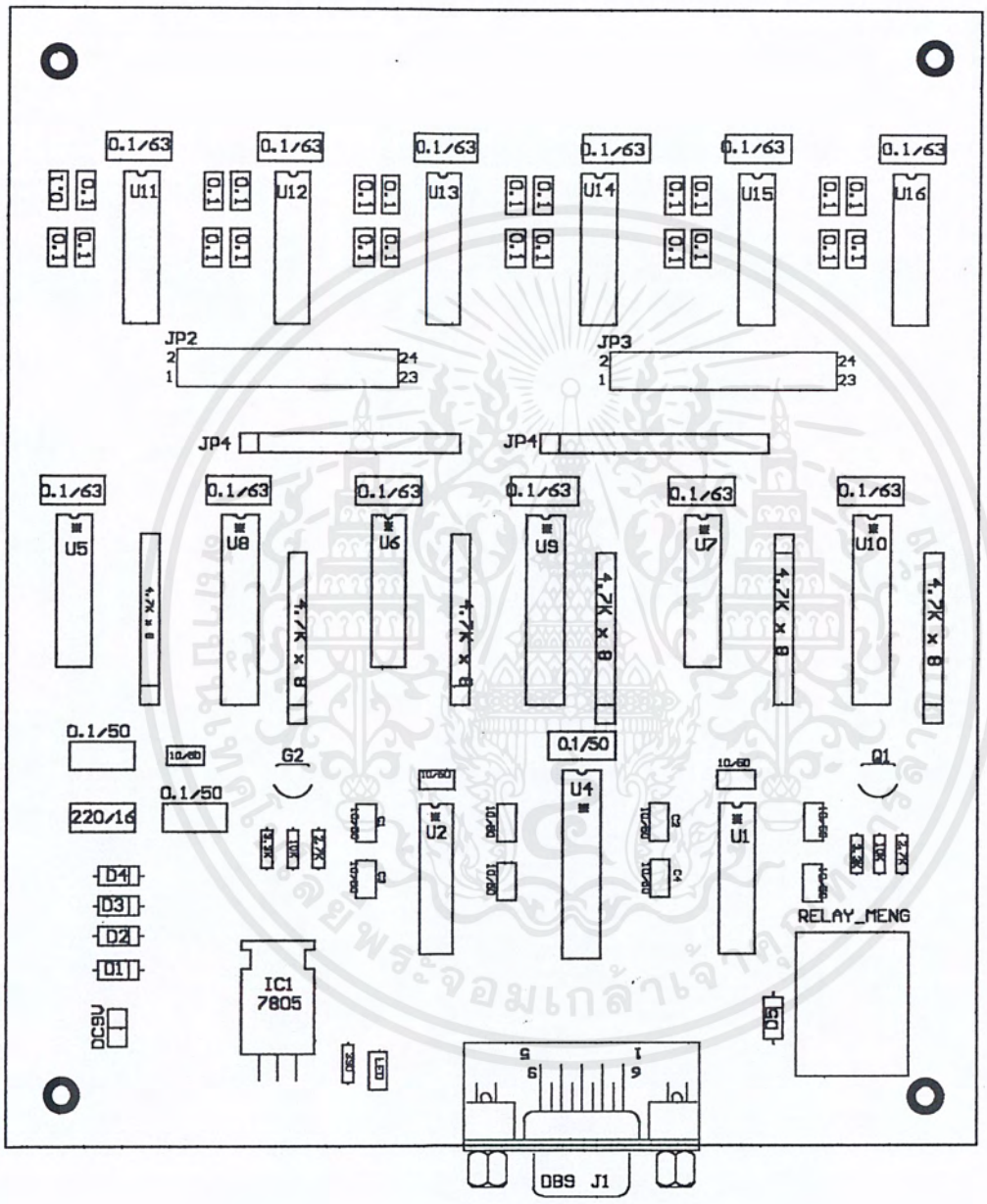
วงจรการเชื่อมต่อพอร์ตอนุกรม



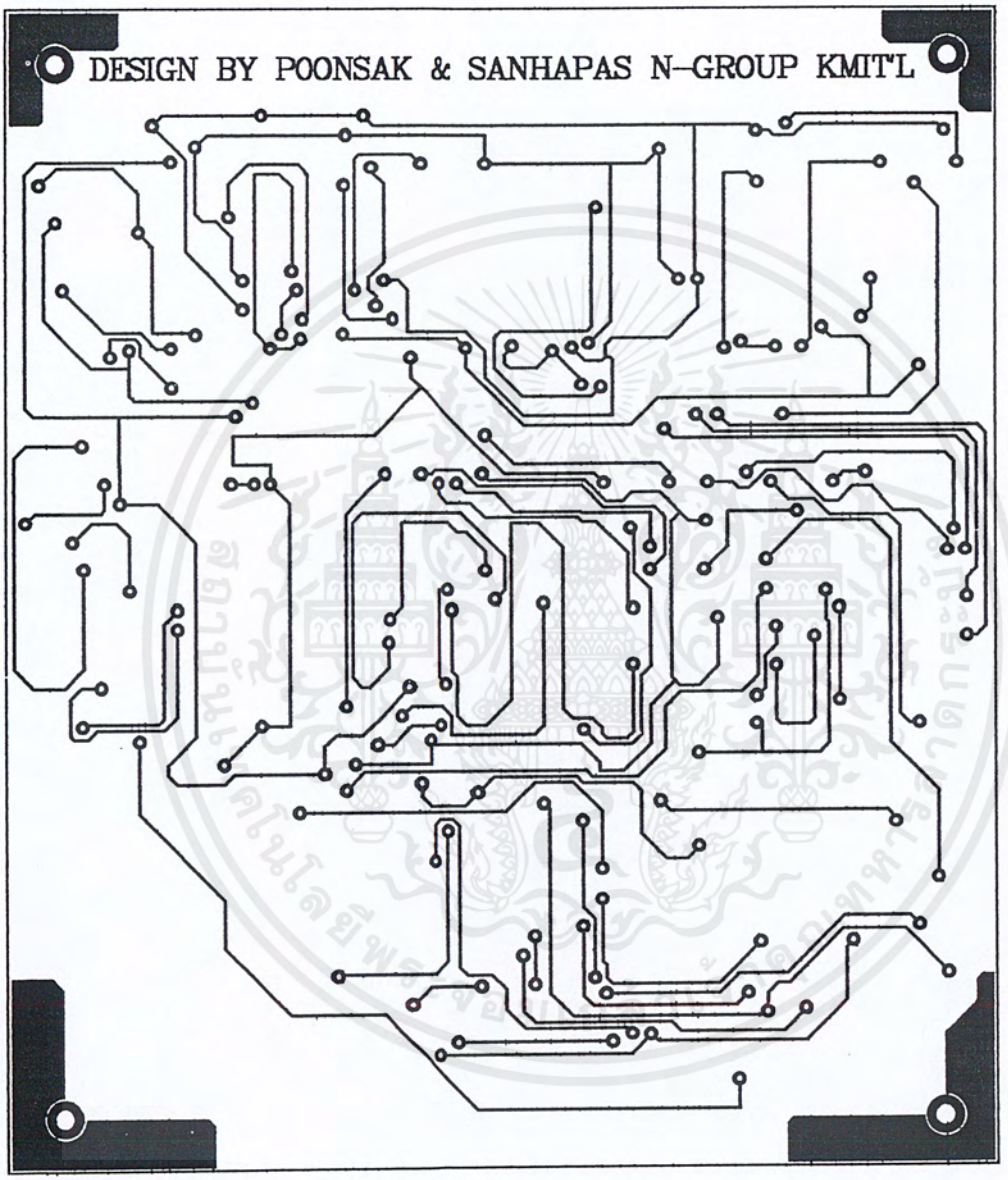
วงจรการขยายพอร์ตให้กับระบบบัส I^2C ด้วย PCF8574



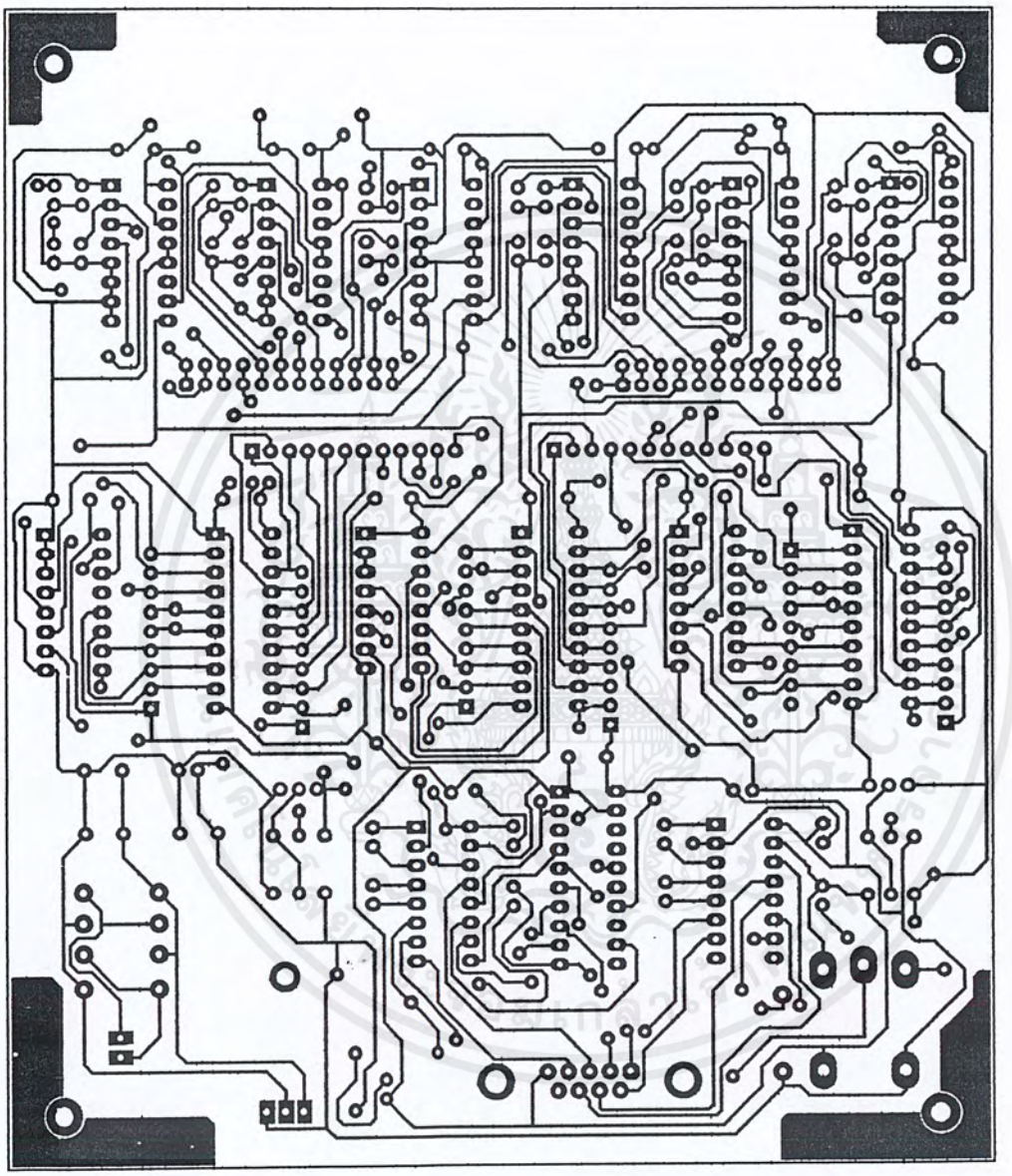
วงจรการเชื่อมต่อสัญญาณอะนาลอกกับระบบบัส I^2C



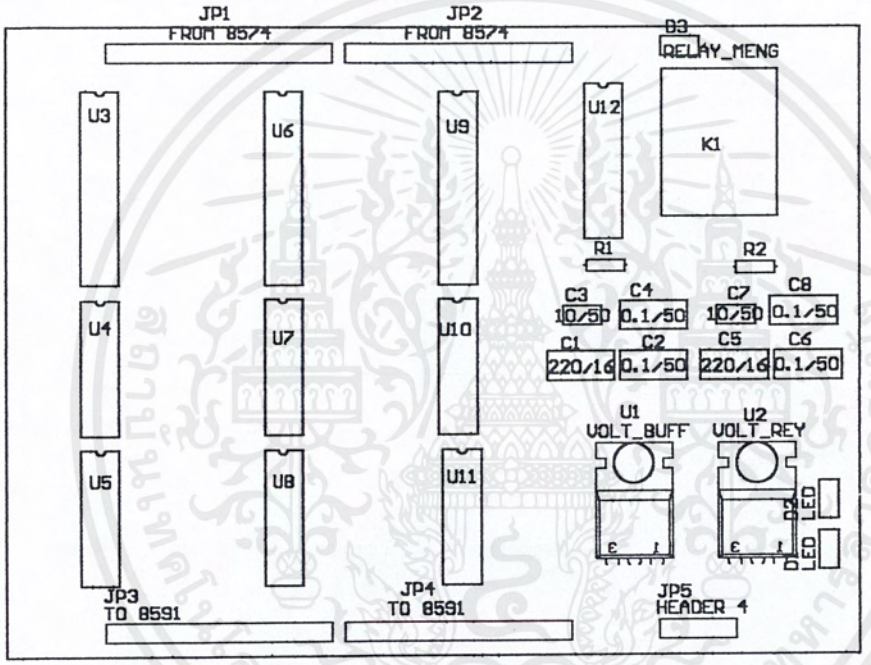
การวางอุปกรณ์ด้านบนของวงจรหลัก



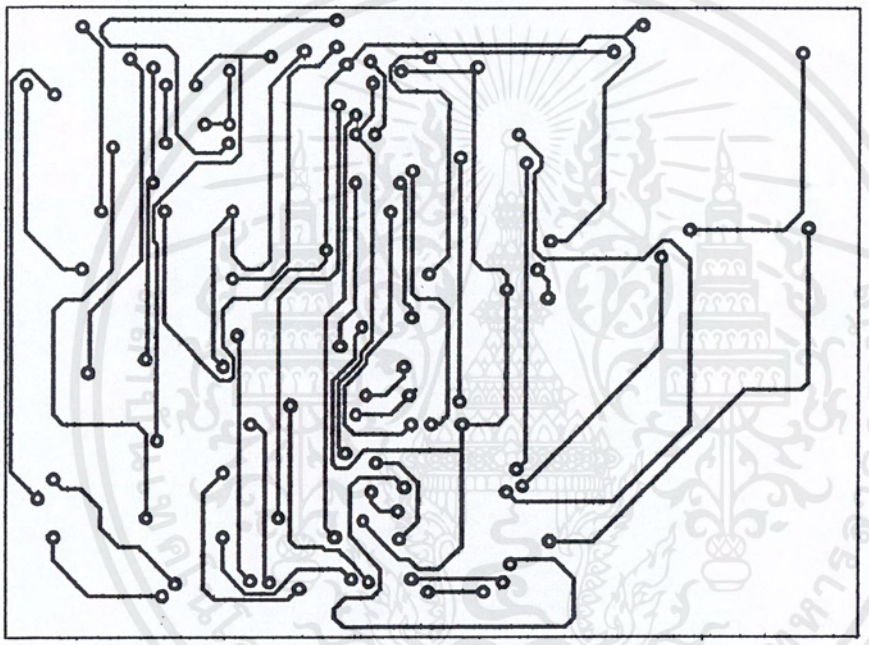
ลายทองแดงด้านบนของวงจรหลัก



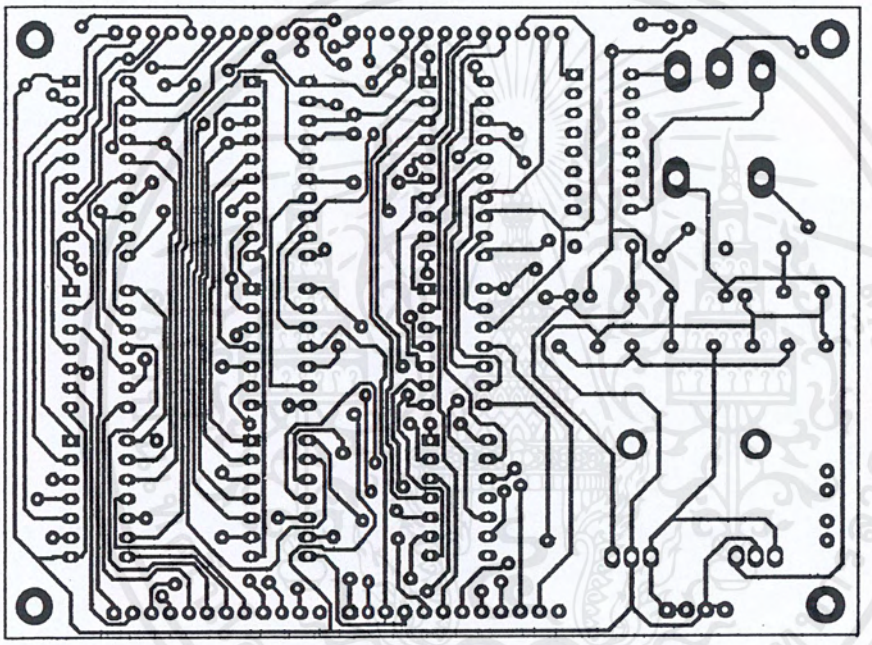
ลายทองแดงด้านล่างของวงจรหลัก



การวางอุปกรณ์ด้านบนของวงจรบัฟเฟอร์



ลายทองแดงด้านบนของวงจรบัฟเฟอร์



ลายทองแดงด้านล่างของวงจรบัฟเฟอร์

ภาคผนวก ค.

Data Sheet

HARRIS SEMICONDUCTOR

ICL232

+5V Powered Dual RS-232 Transmitter/Receiver

December 1993

Features

- Meets All RS-232C Specifications
- Requires Only Single +5V Power Supply
- Onboard Voltage Doubler/Inverter
- Low Power Consumption
- 2 Drivers
  - ±9V Output Swing for +5V Input
  - 300Ω Power-off Source Impedance
  - Output Current Limiting
  - TTL/CMOS Compatible
  - 30V/μs Maximum Slew Rate
- 2 Receivers
  - ±30V Input Voltage Range
  - 3kΩ to 7kΩ Input Impedance
  - 0.5V Hysteresis to Improve Noise Rejection
- All Critical Parameters are Guaranteed Over the Entire Commercial, Industrial and Military Temperature Ranges

Applications

- Any System Requiring RS-232 Communications Port
  - Computer - Portable and Mainframe
  - Peripheral - Printers and Terminals
  - Portable Instrumentation
  - Modems
  - Dataloggers

Description

The ICL232 is a dual RS-232 transmitter/receiver interface circuit that meets all EIA RS-232C specifications. It requires a single +5V power supply, and features two onboard charge pump voltage converters which generate +10V and -10V supplies from the 5V supply.

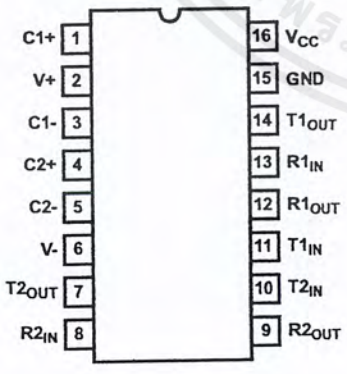
The drivers feature true TTL/CMOS input compatibility, slew-rate-limited output, and 300Ω power-off source impedance. The receivers can handle up to +30V, and have a 3kΩ to 7kΩ input impedance. The receivers also have hysteresis to improve noise rejection.

Ordering Information

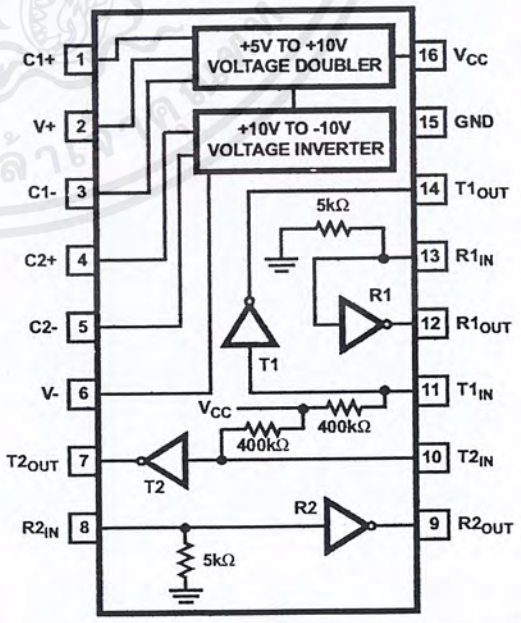
PART NUMBER	TEMPERATURE RANGE	PACKAGE
ICL232CPE	0°C to +70°C	16 Lead Plastic DIP
ICL232CJE	0°C to +70°C	16 Lead Ceramic DIP
ICL232CBE	0°C to +70°C	16 Lead SOIC (W)
ICL232IPE	-40°C to +85°C	16 Lead Plastic DIP
ICL232IJE	-40°C to +85°C	16 Lead Ceramic DIP
ICL232IBE	-40°C to +85°C	16 Lead SOIC (W)
ICL232MJE	-55°C to + 125°C	16 Lead Ceramic DIP

Pinouts

ICL232
(PDIP, CDIP, SOIC)
TOP VIEW



Functional Diagram



CAUTION: These devices are sensitive to electrostatic discharge. Users should follow proper I.C. Handling Procedures.

File Number **3020.2**

Absolute Maximum Ratings

V_{CC} to Ground ..... (GND -0.3V) <V_{CC} < 6V
V+ to Ground ..... (V_{CC} -0.3V) <V+ < 12V
V- to Ground ..... -12V < V- < (GND +0.3V)
Input Voltages
$T1_{IN}$, $T2_{IN}$ ..... (V- -0.3V) < V_{IN} < (V+ +0.3V)
$R1_{IN}$, $R2_{IN}$ ..... ±30V
Output Voltages
$T1_{OUT}$, $T2_{OUT}$ ..... (V- -0.3V) < V_{TXOUT} < (V+ +0.3V)
$R1_{OUT}$, $R2_{OUT}$ ..... (GND -0.3V) < V_{RXOUT} < (V_{CC} +0.3V)
Short Circuit Duration
$T1_{OUT}$, $T2_{OUT}$ ..... Continuous
$R1_{OUT}$, $R2_{OUT}$ ..... Continuous
Storage Temperature Range ..... -65°C to +150°C
Lead Temperature (Soldering 10s) ..... +300°C

Thermal Information

Thermal Resistance	θ_{JA}	θ_{JC}
Ceramic DIP Package	80°C/W	24°C/W
Plastic DIP Package	100°C/W	-
SOIC Package	100°C/W	-

Maximum Power Dissipation 250mW
Operating Temperature Range
ICL232C ..... 0°C to +70°C
ICL232I ..... -40°C to +85°C
ICL232M ..... -55°C to +125°C

CAUTION: Stresses above those listed in "Absolute Maximum Ratings" may cause permanent damage to the device. This is a stress only rating and operation of the device at these or any other conditions above those indicated in the operational sections of this specification is not implied.

Electrical Specifications

Test Conditions: V_{CC} = +5V ±10%, T_A = Operating Temperature Range. Test Circuit as in Figure 8 Unless Otherwise Specified

PARAMETER	TEST CONDITIONS	LIMITS			UNITS
		MIN	TYP	MAX	
Transmitter Output Voltage Swing, T_{OUT}	$T1_{OUT}$ and $T2_{OUT}$ loaded with 3kΩ to Ground	±5	±9	±10	V
Power Supply Current, I_{CC}	Outputs Unloaded, T_A = +25°C	-	5	10	mA
T_{IN}, Input Logic Low, V_{IL}		-	-	0.8	V
T_{IN}, Input Logic High, V_{IH}		2.0	-	-	V
Logic Pullup Current, I_P	$T1_{IN}$, $T2_{IN}$ = 0V	-	15	200	µA
RS-232 Input Voltage Range, V_{IN}		-30	-	+30	V
Receiver Input Impedance, R_{IN}	V_{IN} = ±3V	3.0	5.0	7.0	kΩ
Receiver Input Low Threshold, V_{IN} (H-L)	V_{CC} = 5.0V, T_A = +25°C	0.8	1.2	-	V
Receiver Input High Threshold, V_{IN} (L-H)	V_{CC} = 5.0V, T_A = +25°C	-	1.7	2.4	V
Receiver Input Hysteresis, V_{HYST}		0.2	0.5	1.0	V
TTL/CMOS Receiver Output Voltage Low, V_{OL}	I_{OUT} = 3.2mA	-	0.1	0.4	V
TTL/CMOS Receiver Output Voltage High, V_{OH}	I_{OUT} = -1.0mA	3.5	4.6	-	V
Propagation Delay, t_{PD}	RS-232 to TTL	-	0.5	-	µs
Instantaneous Slew Rate, SR	C_L = 10pF, R_L = 3kΩ, T_A = +25°C (Notes 1, 2)	-	-	30	V/µs
Transition Region Slew Rate, SR_T	R_L = 3kΩ, C_L = 2500pF Measured from +3V to -3V or -3V to +3V	-	3	-	V/µs
Output Resistance, R_{OUT}	V_{CC} = V+ = V- = 0V, V_{OUT} = ±2V	300	-	-	Ω
RS-232 Output Short Circuit Current, I_{SC}	$T1_{OUT}$ or $T2_{OUT}$ shorted to GND	-	±10	-	mA

NOTES:

1. Guaranteed by design.
2. See Figure 4 for definition.

Typical Performance Curves

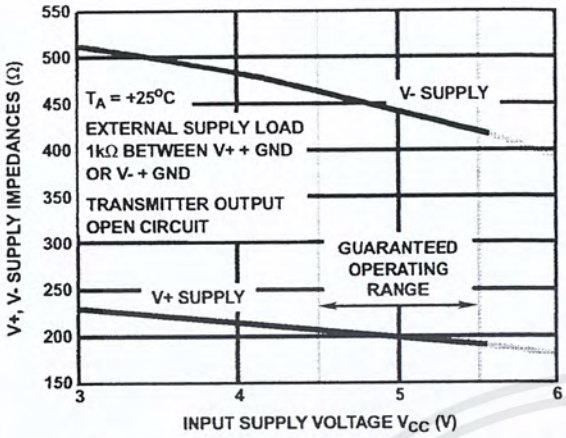


FIGURE 1. V+, V- OUTPUT IMPEDANCES vs V_{CC}

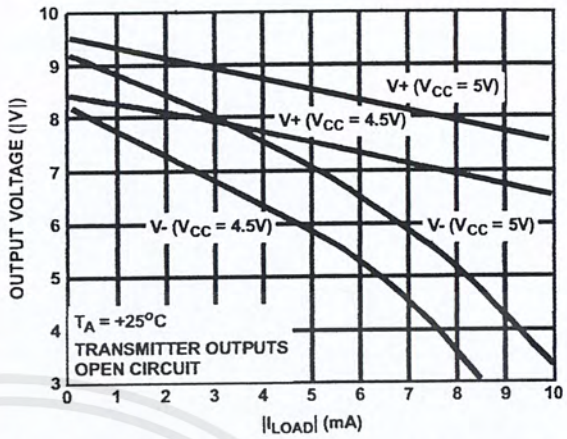


FIGURE 2. V+, V- OUTPUT VOLTAGES vs LOAD CURRENT

Pin Descriptions

PLASTIC DIP, CERAMIC DIP	SOIC	PIN NAME	DESCRIPTION
1	1	C1+	External capacitor "+" for internal voltage doubler.
2	2	V+	Internally generated +10V (typical) supply.
3	3	C1-	External capacitor "-" for internal voltage doubler.
4	4	C2+	External capacitor "+" internal voltage inverter.
5	5	C2-	External capacitor "-" internal voltage inverter.
6	6	V-	Internally generated -10V (typical) supply.
7	7	$T2_{OUT}$	RS-232 Transmitter 2 output ±10V (typical).
8	8	$R2_{IN}$	RS-232 Receiver 2 input, with internal 5K pulldown resistor to GND.
9	9	R2out	Receiver 2 TTL/CMOS output.
10	10	$T2_{IN}$	Transmitter 2 TTL/CMOS input, with internal 400K pullup resistor to V_{CC}.
11	11	$T1_{IN}$	Transmitter 1 TTL/CMOS input, with internal 400K pullup resistor to V_{CC}.
12	12	$R1_{OUT}$	Receiver 1 TTL/CMOS output.
13	13	$R1_{IN}$	RS-232 Receiver 1 input, with internal 5K pulldown resistor to GND.
14	14	$T1_{OUT}$	RS-232 Transmitter 1 output ±10V (typical).
15	15	GND	Supply Ground.
16	16	VCC	Positive Power Supply +5V ±10%

Detailed Description

The ICL232 is a dual RS-232 transmitter/receiver powered by a single +5V power supply which meets all EIA RS232C specifications and features low power consumption. The functional diagram illustrates the major elements of the ICL232. The circuit is divided into three sections: a voltage doubler/inverter, dual transmitters, and dual receivers.

Voltage Converter

An equivalent circuit of the dual charge pump is illustrated in Figure 3.

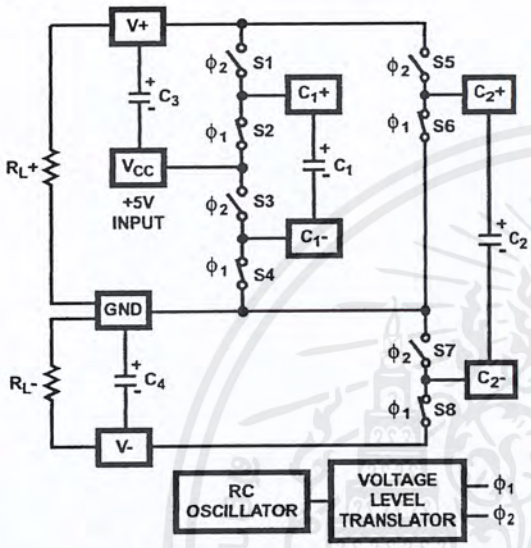
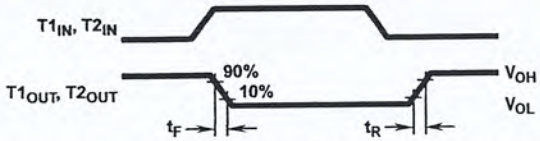


FIGURE 3. DUAL CHARGE PUMP

The voltage quadrupler contains two charge pumps which use two phases of an internally generated clock to generate +10V and -10V. The nominal clock frequency is 16kHz. During phase one of the clock, capacitor C1 is charged to V_{CC}. During phase two, the voltage on C1 is added to V_{CC}, producing a signal across C2 equal to twice V_{CC}. At the same time, C3 is also charged to $2V_{CC}$, and then during phase one, it is inverted with respect to ground to produce a signal across C4 equal to $-2V_{CC}$. The voltage converter accepts input voltages up to 5.5V. The output impedance of the doubler (V+) is approximately 200Ω, and the output impedance of the inverter (V-) is approximately 450Ω. Typical graphs are presented which show the voltage converters output vs input voltage and output voltages vs load characteristics. The test circuit (Figure 8) uses 1µF capacitors for C1-C4, however, the value is not critical. Increasing the values of C1 and C2 will lower the output impedance of the voltage doubler and inverter, and increasing the values of the reservoir capacitors, C3 and C4, lowers the ripple on the V+ and V- supplies.



$$\text{Instantaneous Slew Rate (SR)} = \frac{(0.8)(V_{OH} - V_{OL})}{t_R} \text{ or } \frac{(0.8)(V_{OL} - V_{OH})}{t_F}$$

FIGURE 4. SLEW RATE DEFINITION

Transmitters

The transmitters are TTL/CMOS compatible inverters which translate the inputs to RS-232 outputs. The input logic threshold is about 26% of V_{CC}, or 1.3V for V_{CC} = 5V. A logic 1 at the input results in a voltage of between -5V and V- at the output, and a logic 0 results in a voltage between +5V and (V+ -0.6V). Each transmitter input has an internal 400kΩ pullup resistor so any unused input can be left unconnected and its output remains in its low state. The output voltage swing meets the RS-232C specification of ±5V minimum with the worst case conditions of: both transmitters driving 3kΩ minimum load impedance, V_{CC} = 4.5V, and maximum allowable operating temperature. The transmitters have an internally limited output slew rate which is less than 30V/µs. The outputs are short circuit protected and can be shorted to ground indefinitely. The powered down output impedance is a minimum of 300Ω with ±2V applied to the outputs and V_{CC} = 0V.

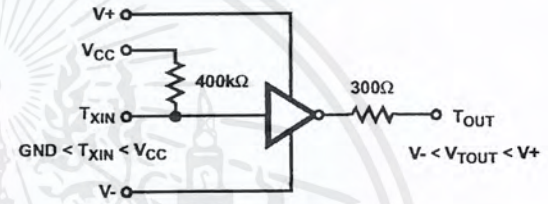


FIGURE 5. TRANSMITTER

Receivers

The receiver inputs accept up to ±30V while presenting the required 3kΩ to 7kΩ input impedance even it the power is off (V_{CC} = 0V). The receivers have a typical input threshold of 1.3V which is within the ±3V limits, known as the transition region, of the RS-232 specification. The receiver output is 0V to V_{CC}. The output will be low whenever the input is greater than 2.4V and high whenever the input is floating or driven between +0.8V and -30V. The receivers feature 0.5V hysteresis to improve noise rejection.

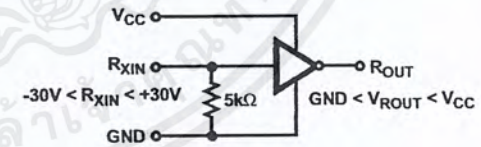
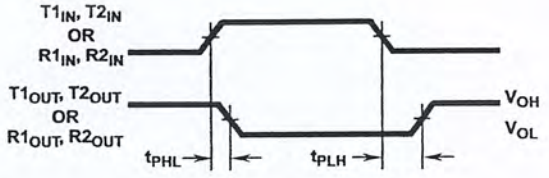


FIGURE 6. RECEIVER



$$\text{Average Propagation Delay} = \frac{t_{PHL} + t_{PLH}}{2}$$

FIGURE 7. PROPAGATION DELAY DEFINITION

Test Circuits

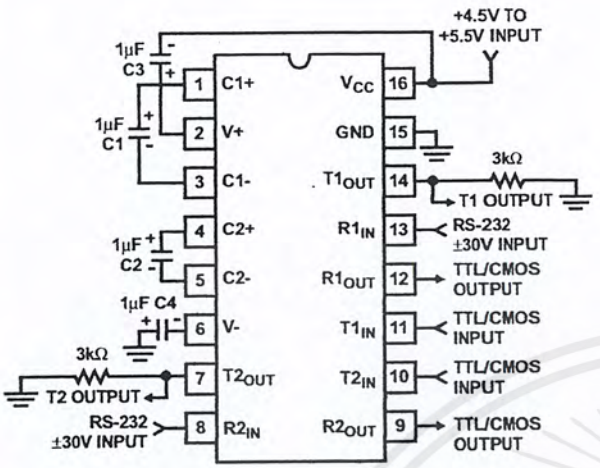


FIGURE 8. GENERAL TEST CIRCUIT

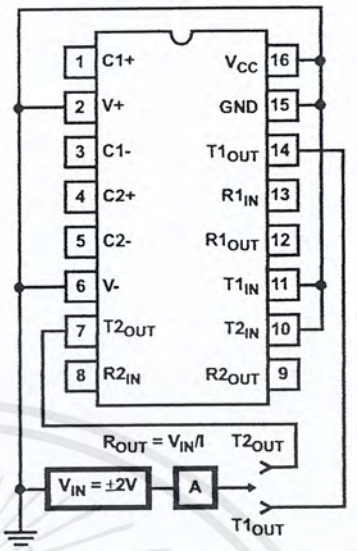


FIGURE 9. POWER-OFF SOURCE RESISTANCE CONFIGURATION

Applications

The ICL232 may be used for all RS-232 data terminal and communication links. It is particularly useful in applications where ±12V power supplies are not available for conventional RS-232 interface circuits. The applications presented represent typical interface configurations.

A simple duplex RS-232 port with CTS/RTS handshaking is illustrated in Figure 10. Fixed output signals such as DTR (data terminal ready) and DSRS (data signaling rate select) is generated by driving them through a 5kΩ resistor connected to V+.

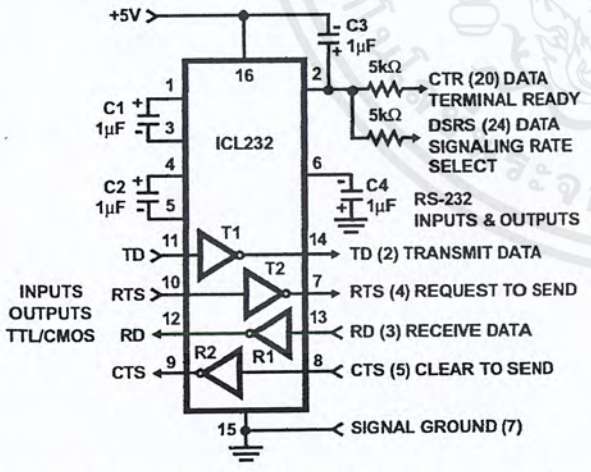


FIGURE 10. SIMPLE DUPLEX RS-232 PORT WITH CTS/RTS HANDSHAKING

In applications requiring four RS-232 inputs and outputs (Figure 11), note that each circuit requires two charge pump capacitors (C1 and C2) but can share common reservoir capacitors (C3 and C4). The benefit of sharing common reservoir capacitors is the elimination of two capacitors and the reduction of the charge pump source impedance which effectively increases the output swing of the transmitters.

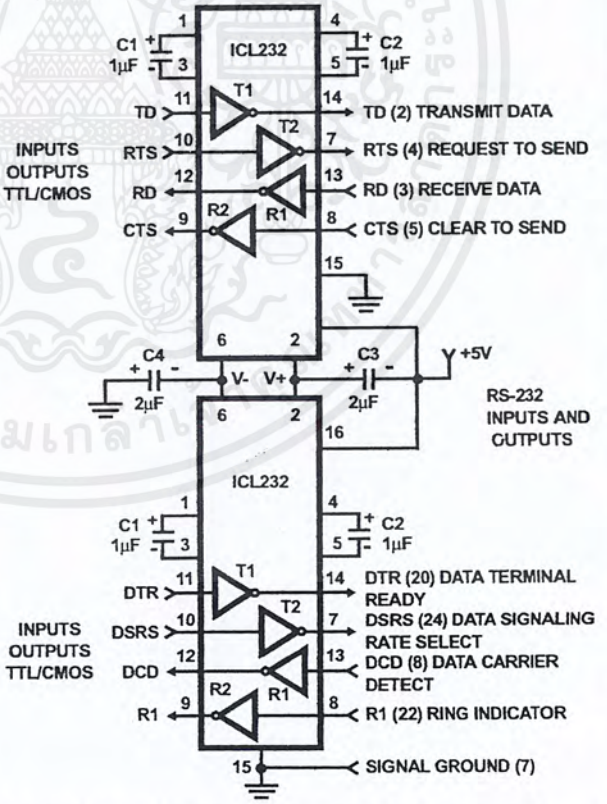


FIGURE 11. COMBINING TWO ICL232s FOR 4 PAIRS OF RS-232 INPUTS AND OUTPUTS

5 BLOCK DIAGRAM

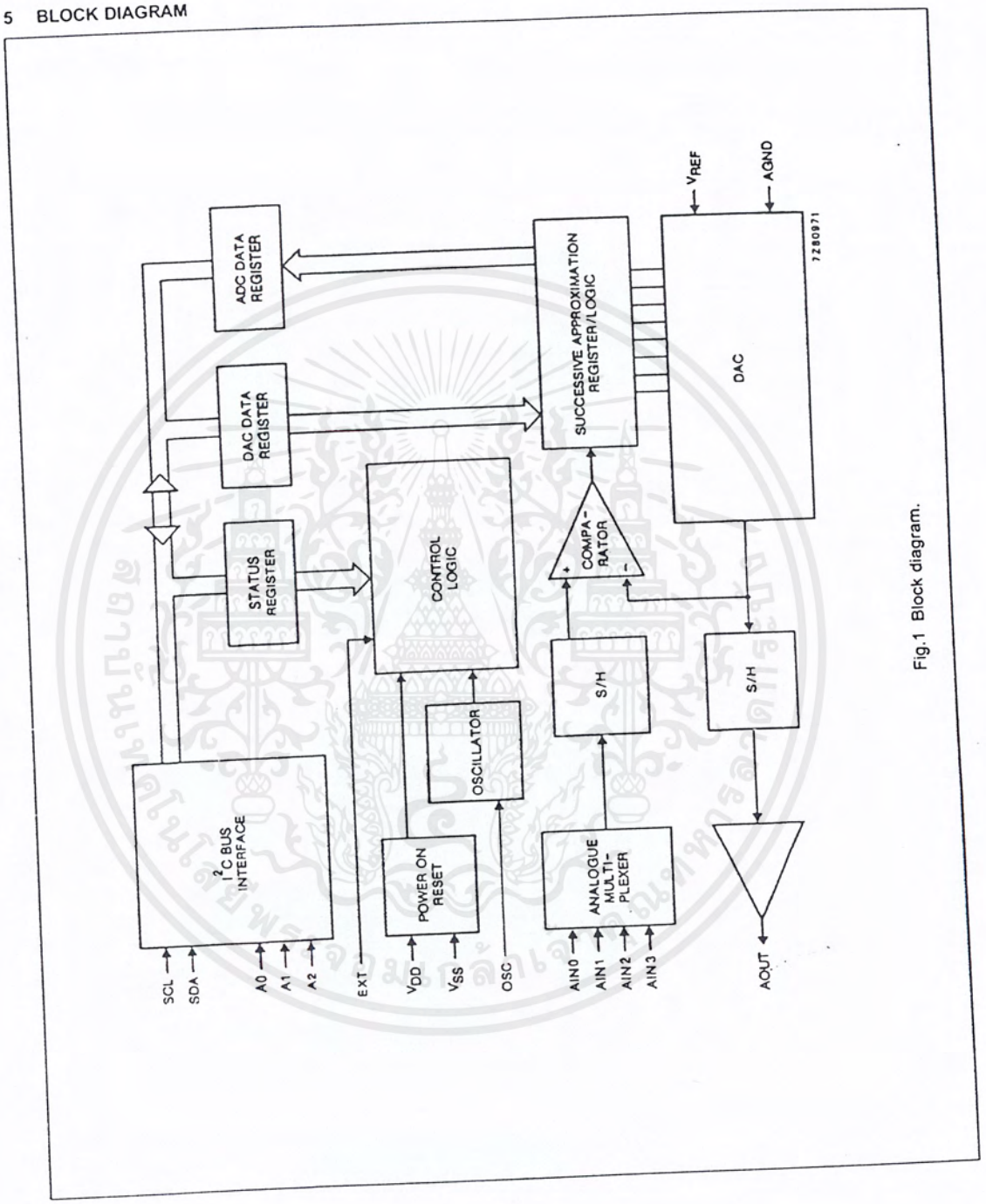


Fig.1 Block diagram.

7 FUNCTIONAL DESCRIPTION

7.1 Addressing

Each PCF8591 device in an I²C-bus system is activated by sending a valid address to the device. The address consists of a fixed part and a programmable part. The programmable part must be set according to the address pins A0, A1 and A2. The address always has to be sent as the first byte after the start condition in the I²C-bus protocol. The last bit of the address byte is the read/write-bit which sets the direction of the following data transfer (see Figs 3, 15 and 16).

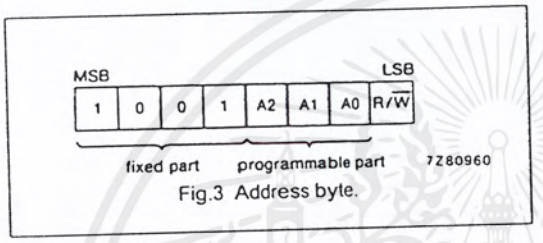


Fig.3 Address byte.

7.2 Control byte

The second byte sent to a PCF8591 device will be stored in its control register and is required to control the device function.

The upper nibble of the control register is used for enabling the analog output, and for programming the analog inputs as single-ended or differential inputs. The lower nibble selects one of the analog input channels defined by the upper nibble (see Fig.4). If the auto-increment flag is set the channel number is incremented automatically after each A/D conversion.

If the auto-increment mode is desired in applications where the internal oscillator is used, the analog output enable flag in the control byte (bit 6) should be set. This allows the internal oscillator to run continuously, thereby preventing conversion errors resulting from oscillator start-up delay. The analog output enable flag may be reset at other times to reduce quiescent power consumption.

The selection of a non-existing input channel results in the highest available channel number being allocated. Therefore, if the auto-increment flag is set, the next selected channel will be always channel 0. The most significant bits of both nibbles are reserved for future functions and have to be set to 0. After a Power-on reset condition all bits of the control register are reset to 0. The D/A converter and the oscillator are disabled for power saving. The analog output is switched to a high-impedance state.

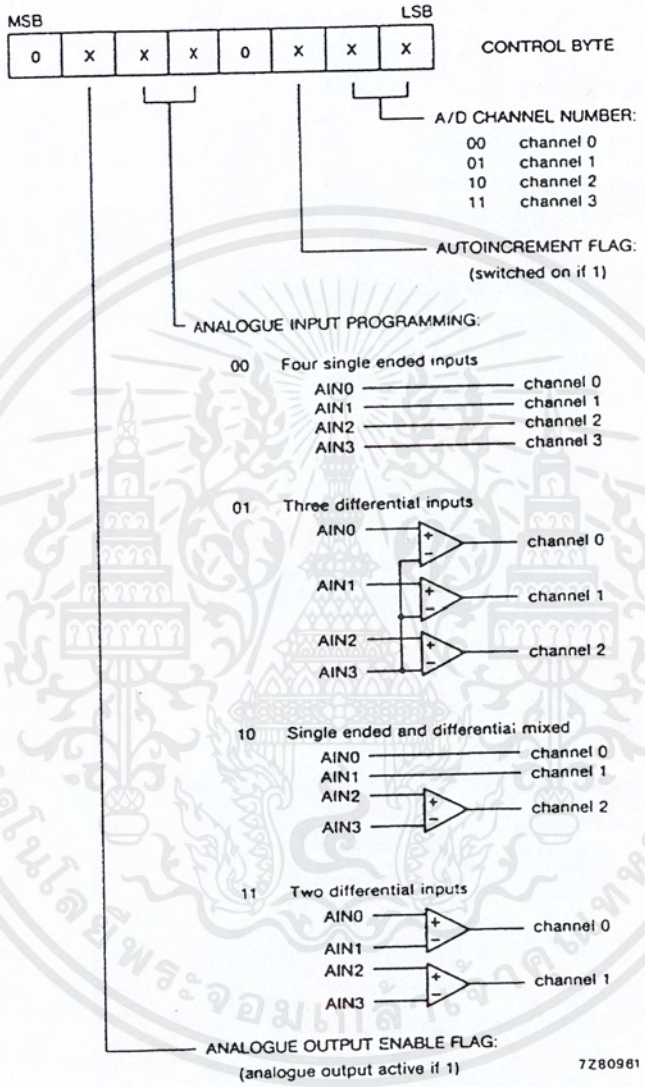


Fig.4 Control byte.

7.3 D/A conversion

The third byte sent to a PCF8591 device is stored in the DAC data register and is converted to the corresponding analog voltage using the on-chip D/A converter. This D/A converter consists of a resistor divider chain connected to the external reference voltage with 256 taps and selection switches. The tap-decoder switches one of these taps to the DAC output line (see Fig.5).

The analog output voltage is buffered by an auto-zeroed unity gain amplifier. This buffer amplifier may be switched on or off by setting the analog output enable flag of the control register. In the active state the output voltage is held until a further data byte is sent.

The on-chip D/A converter is also used for successive approximation A/D conversion. In order to release the DAC for an A/D conversion cycle the unity gain amplifier is equipped with a track and hold circuit. This circuit holds the output voltage while executing the A/D conversion.

The output voltage supplied to the analog output AOUT is given by the formula shown in Fig.6. The waveforms of a D/A conversion sequence are shown in Fig.7

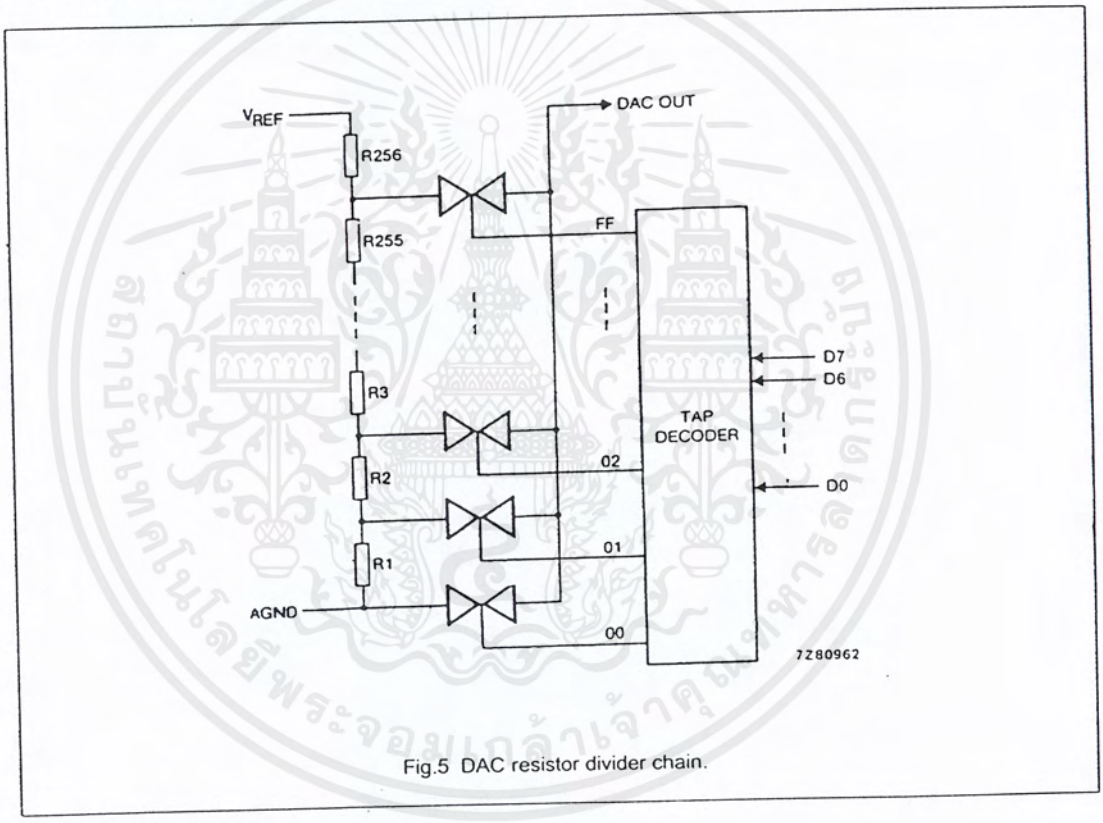


Fig.5 DAC resistor divider chain.

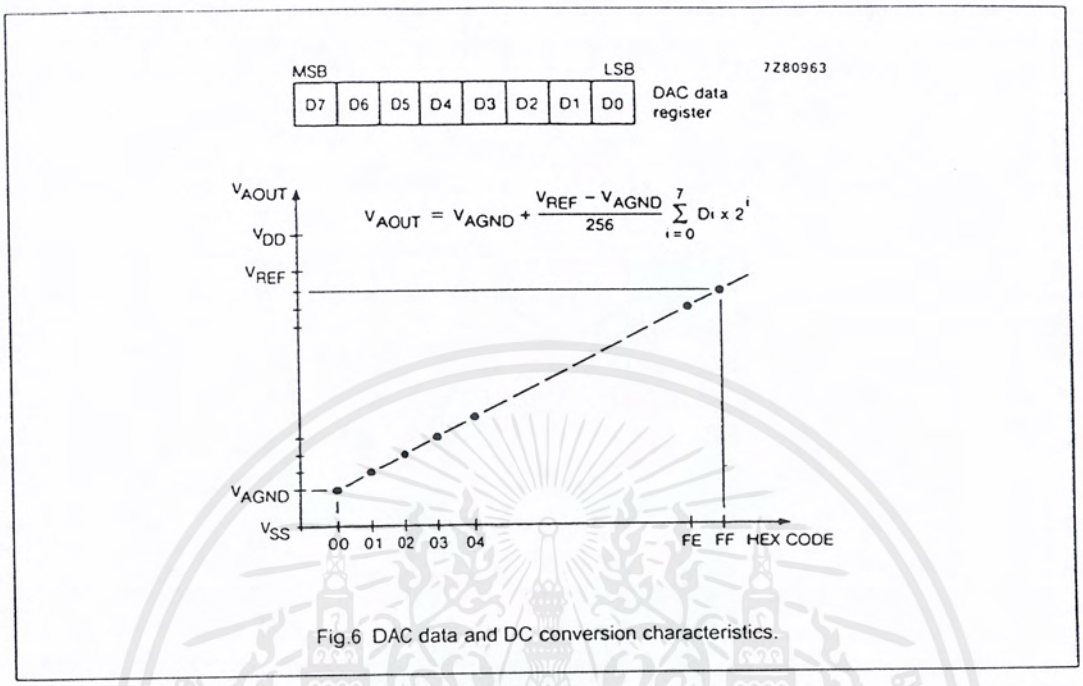


Fig.6 DAC data and DC conversion characteristics.

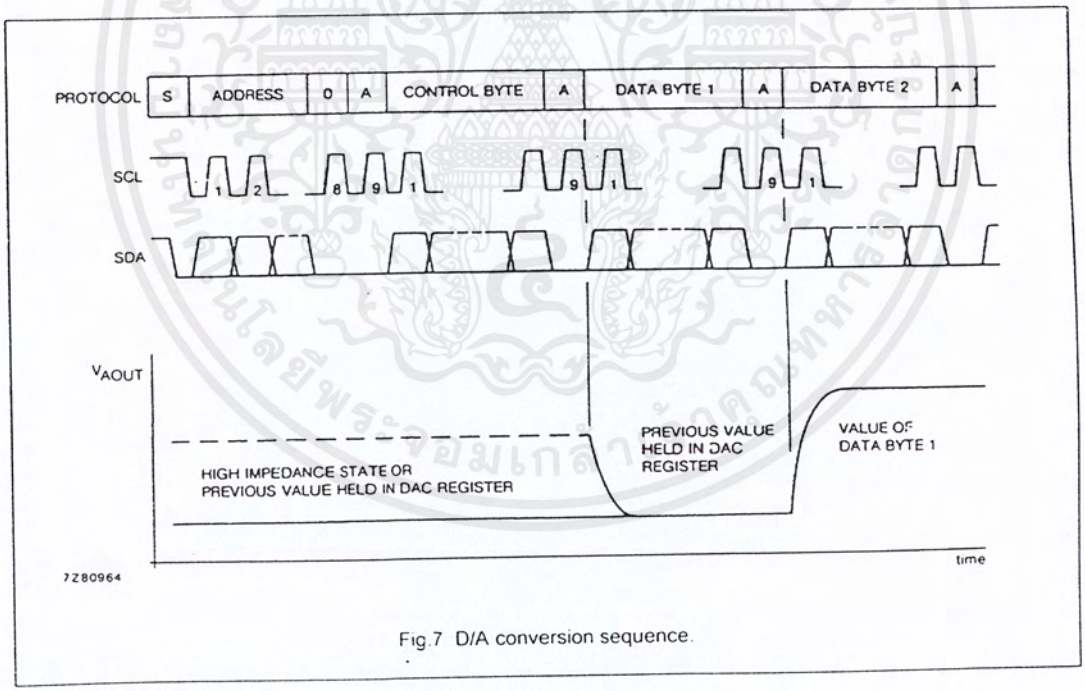


Fig.7 D/A conversion sequence.

7.4 A/D conversion

The A/D converter makes use of the successive approximation conversion technique. The on-chip D/A converter and a high-gain comparator are used temporarily during an A/D conversion cycle.

An A/D conversion cycle is always started after sending a valid read mode address to a PCF8591 device. The A/D conversion cycle is triggered at the trailing edge of the acknowledge clock pulse and is executed while transmitting the result of the previous conversion (see Fig.8).

Once a conversion cycle is triggered an input voltage sample of the selected channel is stored on the chip and is converted to the corresponding 8-bit binary code. Samples picked up from differential inputs are converted to an 8-bit two's complement code (see Figs 9 and 10)

The conversion result is stored in the ADC data register and awaits transmission. If the auto-increment flag is set the next channel is selected.

The first byte transmitted in a read cycle contains the conversion result code of the previous read cycle. After a Power-on reset condition the first byte read is a hexadecimal 80. The protocol of an I^2C-bus read cycle is shown in Chapter 8, Figs 15 and 16.

The maximum A/D conversion rate is given by the actual speed of the I^2C-bus.

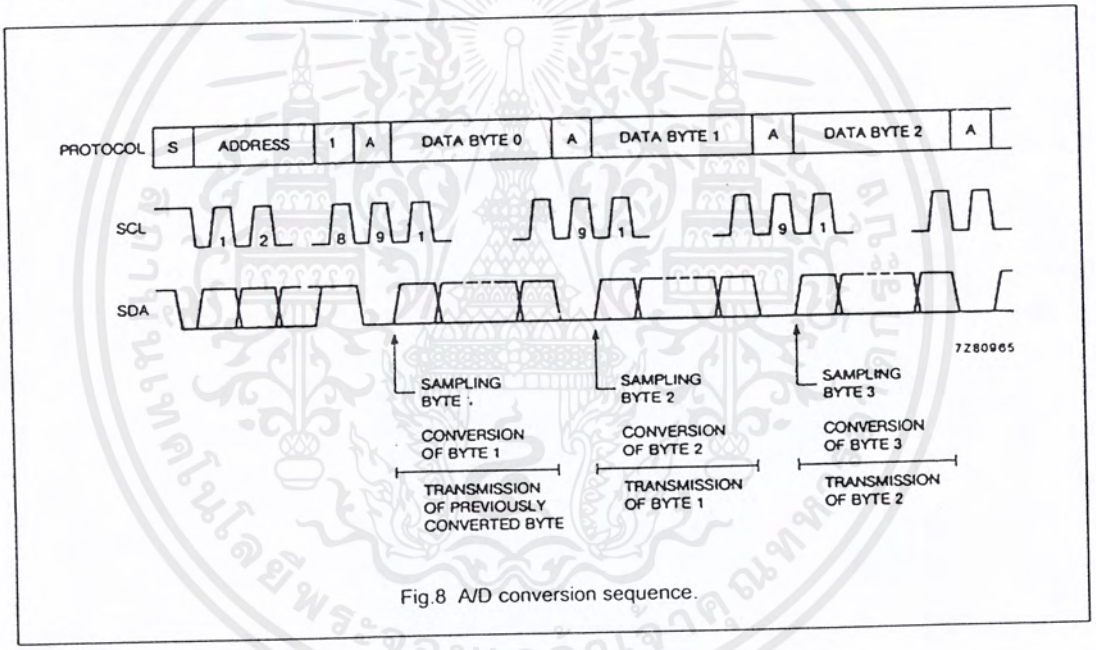


Fig.8 A/D conversion sequence.

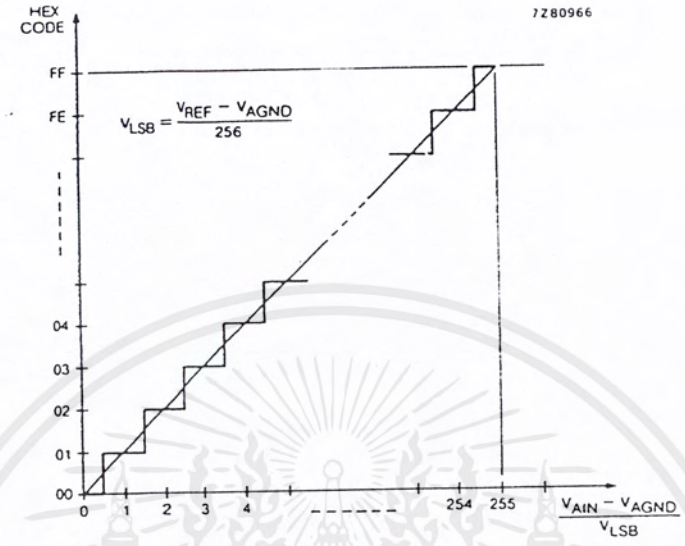


Fig.9 A/D conversion characteristics of single-ended inputs.

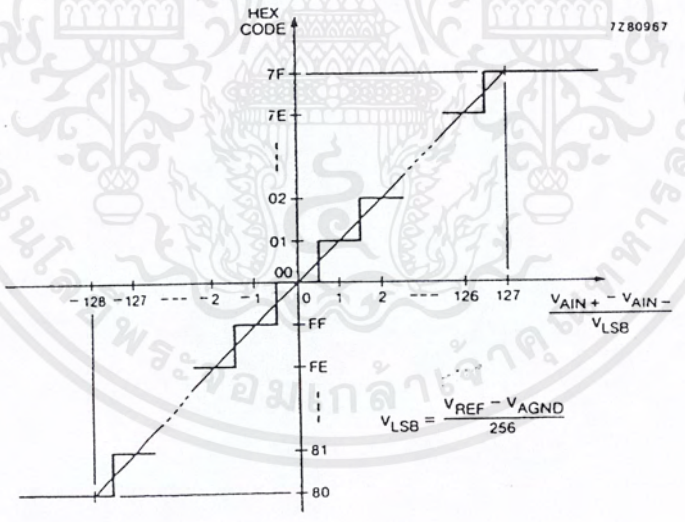


Fig.10 A/D conversion characteristics of differential inputs.

7.5 Reference voltage

For the D/A and A/D conversion either a stable external voltage reference or the supply voltage has to be applied to the resistor divider chain (pins V_{REF} and AGND). The AGND pin has to be connected to the system analog ground and may have a DC off-set with reference to V_{SS}.

A low frequency may be applied to the V_{REF} and AGND pins. This allows the use of the D/A converter as a one-quadrant multiplier; see Chapter 15 and Fig.6.

The A/D converter may also be used as a one or two quadrant analog divider. The analog input voltage is divided by the reference voltage. The result is converted to a binary code. In this application the user has to keep the reference voltage stable during the conversion cycle.

7.6 Oscillator

An on-chip oscillator generates the clock signal required for the A/D conversion cycle and for refreshing the auto-zeroed buffer amplifier. When using this oscillator the EXT pin has to be connected to V_{SS}. At the OSC pin the oscillator frequency is available.

If the EXT pin is connected to V_{DD} the oscillator output OSC is switched to a high-impedance state allowing the user to feed an external clock signal to OSC.

8 CHARACTERISTICS OF THE I²C-BUS

The I²C-bus is for bidirectional, two-line communication between different ICs or modules. The two lines are a serial data line (SDA) and a serial clock line (SCL). Both lines must be connected to a positive supply via a pull-up resistor. Data transfer may be initiated only when the bus is not busy.

8.1 Bit transfer

One data bit is transferred during each clock pulse. The data on the SDA line must remain stable during the HIGH period of the clock pulse as changes in the data line at this time will be interpreted as a control signal

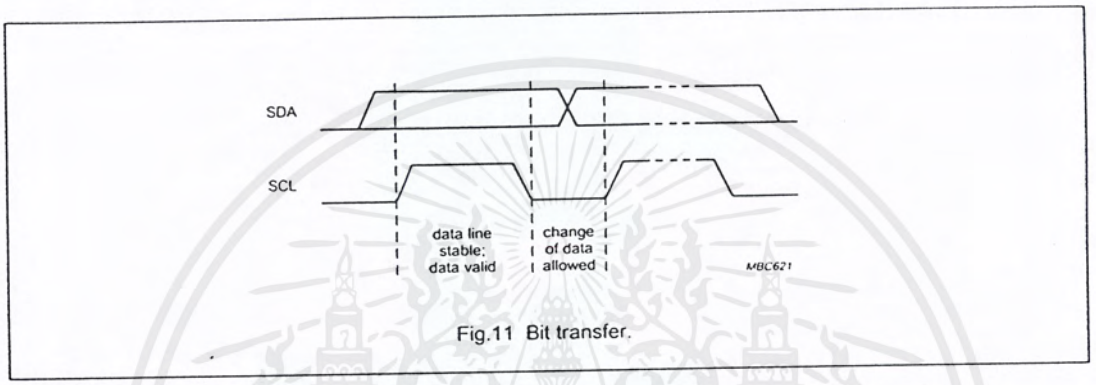


Fig.11 Bit transfer.

8.2 Start and stop conditions

Both data and clock lines remain HIGH when the bus is not busy. A HIGH-to-LOW transition of the data line, while the clock is HIGH, is defined as the start condition (S). A LOW-to-HIGH transition of the data line while the clock is HIGH, is defined as the stop condition (P).

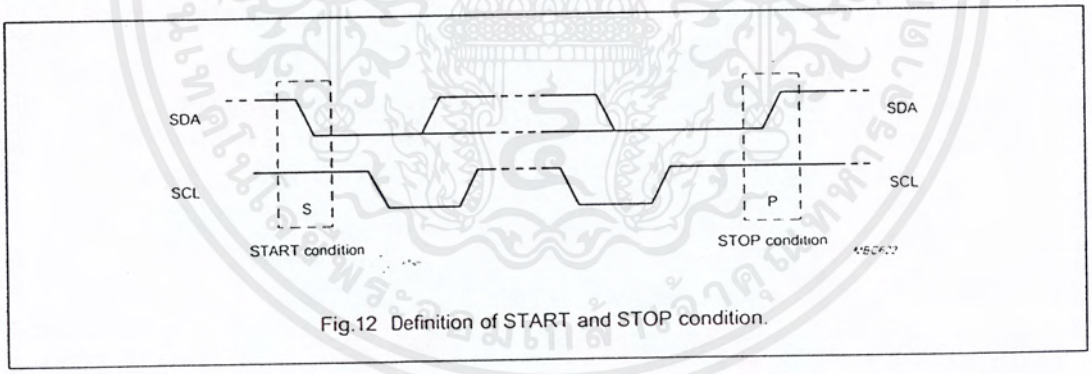


Fig.12 Definition of START and STOP condition.

8.3 System configuration

A device generating a message is a 'transmitter', a device receiving a message is the 'receiver'. The device that controls the message is the 'master' and the devices which are controlled by the master are the 'slaves'.

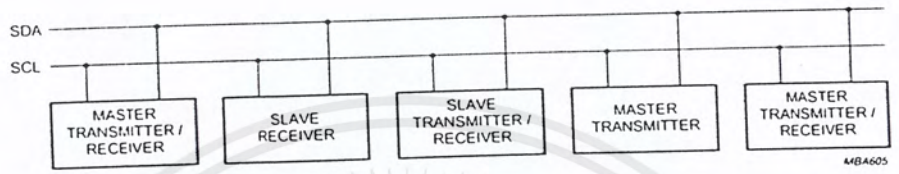


Fig.13 System configuration.

8.4 Acknowledge

The number of data bytes transferred between the start and stop conditions from transmitter to receiver is not limited. Each data byte of eight bits is followed by one acknowledge bit. The acknowledge bit is a HIGH level put on the bus by the transmitter whereas the master also generates an extra acknowledge related clock pulse. A slave receiver which is addressed must generate an acknowledge after the reception of each byte. Also a master must generate an acknowledge after the reception of each byte that has been clocked out of the slave transmitter. The device that acknowledges has to pull down the SDA line during the acknowledge clock pulse, so that the SDA line is stable LOW during the HIGH period of the acknowledge related clock pulse. A master receiver must signal an end of data to the transmitter by **not** generating an acknowledge on the last byte that has been clocked out of the slave. In this event the transmitter must leave the data line HIGH to enable the master to generate a stop condition.

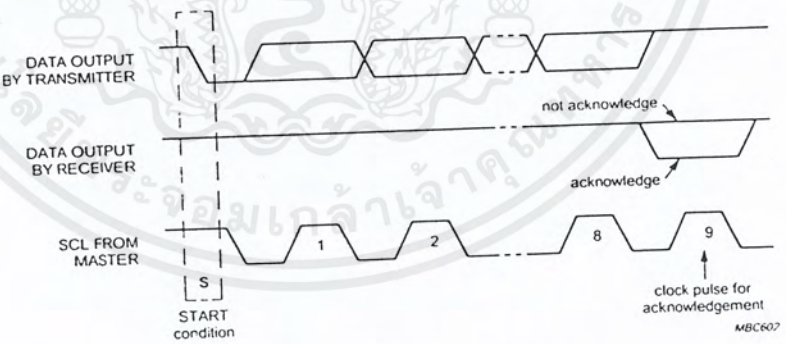


Fig.14 Acknowledgement on the I^2C-bus.

8.5 I²C-bus protocol

After a start condition a valid hardware address has to be sent to a PCF8591 device. The read/write bit defines the direction of the following single or multiple byte data transfer. For the format and the timing of the start condition (S), the stop condition (P) and the acknowledge bit (A) refer to the I²C-bus characteristics. In the write mode a data transfer is terminated by sending either a stop condition or the start condition of the next data transfer.

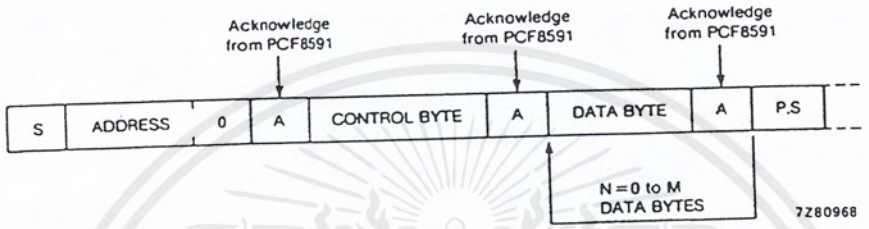


Fig.15 Bus protocol for write mode, D/A conversion.

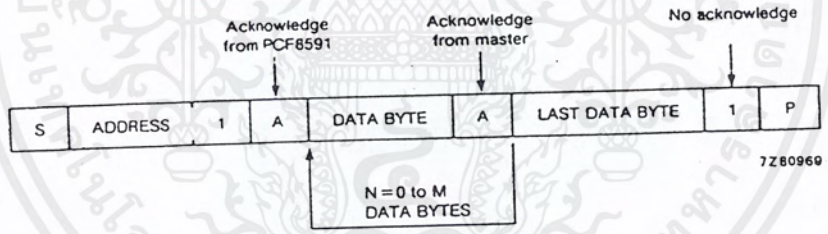


Fig.16 Bus protocol for read mode, A/D conversion.

9 LIMITING VALUES

In accordance with the Absolute Maximum Rating System (IEC 134).

SYMBOL	PARAMETER	MIN.	MAX.	UNIT
V_{DD}	supply voltage (pin 16)	−0.5	+8.0	V
V_I	input voltage (any input)	−0.5	V_{DD} + 0.5	V
I_I	DC input current	−	±10	mA
I_O	DC output current	−	±20	mA
I_{DD}, I_{SS}	V_{DD} or V_{SS} current	−	±50	mA
P_{tot}	total power dissipation per package	−	300	mW
P_O	power dissipation per output	−	100	mW
T_{amb}	operating ambient temperature	−40	+85	°C
T_{stg}	storage temperature	−65	+150	°C

10 HANDLING

Inputs and outputs are protected against electrostatic discharge in normal handling. However, to be totally safe, it is desirable to take precautions appropriate to handling MOS devices. Advice can be found in Data Handbook IC12 under *"Handling MOS Devices"*.

11 DC CHARACTERISTICS

V_{DD} = 2.5 V to 6 V; V_{SS} = 0 V; T_{amb} = −40 °C to +85 °C unless otherwise specified.

SYMBOL	PARAMETER	CONDITIONS	MIN.	TYP.	MAX.	UNIT
Supply						
V_{DD}	supply voltage (operating)		2.5	–	6.0	V
I_{DD}	supply current					
	standby	$V_I = V_{SS}$ or V_{DD}; no load	–	1	15	µA
	operating, AOUT off	f_{SCL} = 100 kHz	–	125	250	µA
	operating, AOUT active	f_{SCL} = 100 kHz	–	0.45	1.0	mA
V_{POR}	Power-on reset level	note 1	0.8	–	2.0	V
Digital inputs/output: SCL, SDA, A0, A1, A2						
V_{IL}	LOW level input voltage		0	–	$0.3 \times V_{DD}$	V
V_{IH}	HIGH level input voltage		$0.7 \times V_{DD}$	–	V_{DD}	V
I_L	leakage current					
	A0, A1, A2	$V_I = V_{SS}$ to V_{DD}	−250	–	+250	nA
	SCL, SDA	$V_I = V_{SS}$ to V_{DD}	−1	–	+1	µA
C_i	input capacitance		–	–	5	pF
I_{OL}	LOW level SDA output current	V_{OL} = 0.4 V	3.0	–	–	mA
Reference voltage inputs						
V_{REF}	reference voltage	$V_{REF} > V_{AGND}$; note 2	V_{SS} + 1.6	–	V_{DD}	V
V_{AGND}	analog ground voltage	$V_{REF} > V_{AGND}$; note 2	V_{SS}	–	V_{DD} − 0.8	V
I_{LI}	input leakage current		−250	–	+250	nA
R_{REF}	input resistance	pins V_{REF} and AGND	–	100	–	kΩ
Oscillator: OSC, EXT						
I_{LI}	input leakage current		–	–	250	nA
f_{OSC}	oscillator frequency		0.75	–	1.25	MHz

Notes

1. The power on reset circuit resets the I²C-bus logic when V_{DD} is less than V_{POR}.
2. A further extension of the range is possible, if the following conditions are fulfilled:

$$\frac{V_{REF} - V_{AGND}}{2} \geq 0.8\,V, V_{DD} \quad \frac{V_{REF} + V_{AGND}}{2} \geq 0.4\,V$$

12 D/A CHARACTERISTICS

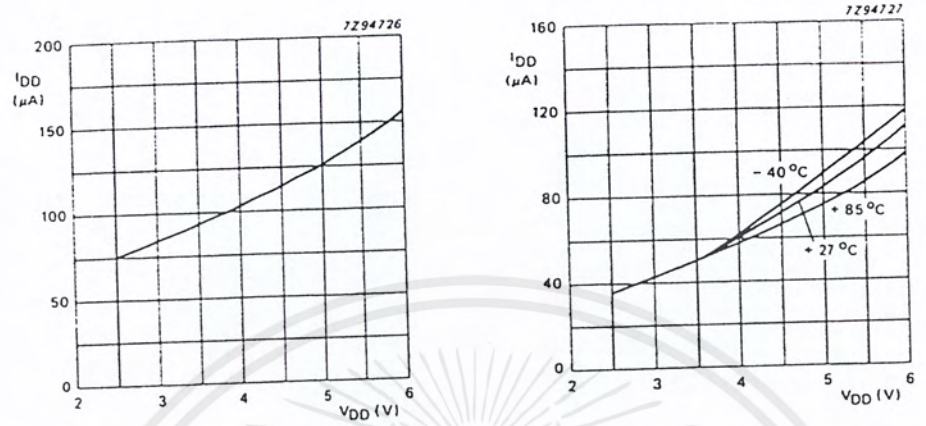
V_{DD} = 5.0 V; V_{SS} = 0 V; V_{REF} = 5.0 V; V_{AGND} = 0 V; R_L = 10 kΩ; C_L = 100 pF; T_{amb} = −40 °C to +85 °C unless otherwise specified.

SYMBOL	PARAMETER	CONDITIONS	MIN.	TYP.	MAX.	UNIT
Analog output						
V_{OA}	output voltage	no resistive load	V_{SS}	–	V_{DD}	V
		R_L = 10 kΩ	V_{SS}	–	0.9 × V_{DD}	V
I_{LO}	output leakage current	AOUT disabled	–	–	250	nA
Accuracy						
OS_e	offset error	T_{amb} = 25 °C	–	–	50	mV
L_e	linearity error		–	–	±1.5	LSB
G_e	gain error	no resistive load	–	–	1	%
t_{DAC}	settling time	to ½ LSB full scale step	–	–	90	µs
f_{DAC}	conversion rate		–	–	11.1	kHz
SNRR	supply noise rejection ratio	f = 100 Hz; V_{DDN} = 0.1 × V_{PP}	–	40	–	dB

13 A/D CHARACTERISTICS

V_{DD} = 5.0 V; V_{SS} = 0 V; V_{REF} = 5.0 V; V_{AGND} = 0 V; R_S = 10 kΩ; T_{amb} = −40 °C to +85 °C unless otherwise specified.

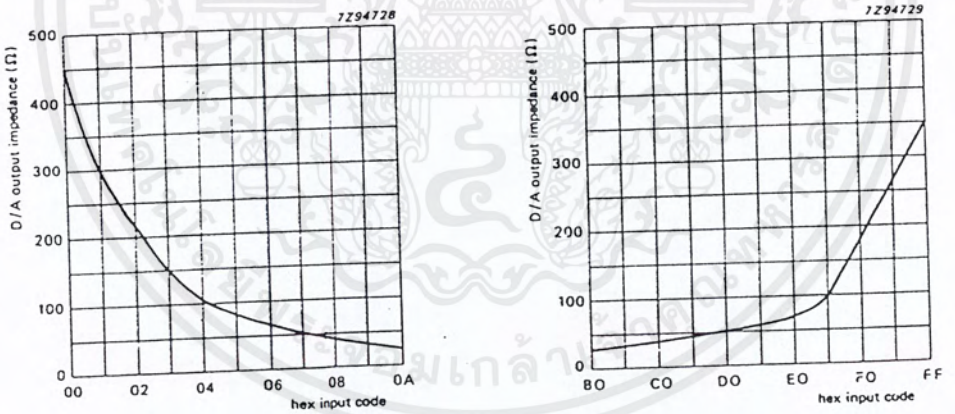
SYMBOL	PARAMETER	CONDITIONS	MIN.	TYP.	MAX.	UNIT
Analog inputs						
V_{IA}	analog input voltage		V_{SS}	–	V_{DD}	V
I_{LIA}	analog input leakage current		–	–	100	nA
C_{IA}	analog input capacitance		–	10	–	pF
C_{ID}	differential input capacitance		–	10	–	pF
V_{IS}	single-ended voltage	measuring range	V_{AGND}	–	V_{REF}	V
V_{ID}	differential voltage	measuring range; V_{FS} = V_{REF} − V_{AGND}	$-\frac{V_{FS}}{2}$		$+\frac{V_{FS}}{2}$	V
Accuracy						
OS_e	offset error	T_{amb} = 25 °C	–	–	20	mV
L_e	linearity error		–	–	±1.5	LSB
G_e	gain error		–	–	1	%
GS_e	small-signal gain error	ΔV_i = 16 LSB	–	–	5	%
CMRR	common-mode rejection ratio		–	60	–	dB
SNRR	supply noise rejection ratio	f = 100 Hz; V_{DDN} = 0.1 × V_{PP}	–	40	–	dB
t_{ADC}	conversion time		–	–	90	µs
f_{ADC}	sampling/conversion rate		–	–	11.1	kHz



(a) Internal oscillator; T_{amb} = +27 °C.

(b) External oscillator.

Fig.17 Operating supply current as a function of supply voltage (analog output disabled).



(a) Output impedance near negative power rail; T_{amb} = +27 °C.

(b) Output impedance near positive power rail; T_{amb} = +27 °C.

The x-axis represents the hex input-code equivalent of the output voltage.

Fig.18 Output impedance of analog output buffer (near power rails).

14 AC CHARACTERISTICS

All timing values are valid within the operating supply voltage and ambient temperature range and reference to V_{IL} and V_{IH} with an input voltage swing of V_{SS} to V_{DD}.

SYMBOL	PARAMETER	MIN.	TYP.	MAX.	UNIT
I²C-bus timing (see Fig.19; note 1)					
f_{SCL}	SCL clock frequency	–	–	100	kHz
t_{SP}	tolerable spike width on bus	–	–	100	ns
t_{BUF}	bus free time	4.7	–	–	µs
$t_{SU;STA}$	START condition set-up time	4.7	–	–	µs
$t_{HD;STA}$	START condition hold time	4.0	–	–	µs
t_{LOW}	SCL LOW time	4.7	–	–	µs
t_{HIGH}	SCL HIGH time	4.0	–	–	µs
t_r	SCL and SDA rise time	–	–	1.0	µs
t_f	SCL and SDA fall time	–	–	0.3	µs
$t_{SU;DAT}$	data set-up time	250	–	–	ns
$t_{HD;DAT}$	data hold time	0	–	–	ns
$t_{VD;DAT}$	SCL LOW-to-data out valid	–	–	3.4	µs
$t_{SU;STO}$	STOP condition set-up time	4.0	–	–	µs

Note

1. A detailed description of the I²C-bus specification, with applications, is given in brochure *"The I²C-bus and how to use it"*. This brochure may be ordered using the code 9398 393 40011.

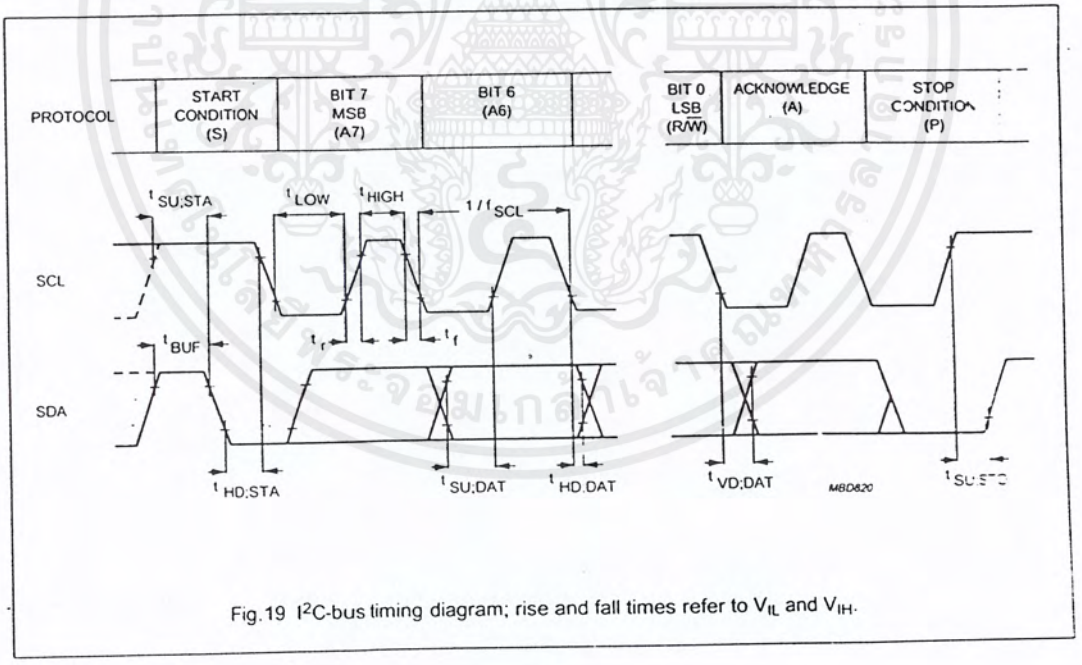


Fig.19 I²C-bus timing diagram; rise and fall times refer to V_{IL} and V_{IH}.

15 APPLICATION INFORMATION

Inputs must be connected to V_{SS} or V_{DD} when not in use. Analog inputs may also be connected to AGND or V_{REF}.

In order to prevent excessive ground and supply noise and to minimize cross-talk of the digital to analog signal paths the user has to design the printed-circuit board layout very carefully. Supply lines common to a PCF8591 device and noisy digital circuits and ground loops should be avoided. Decoupling capacitors (>10 μF) are recommended for power supply and reference voltage inputs.

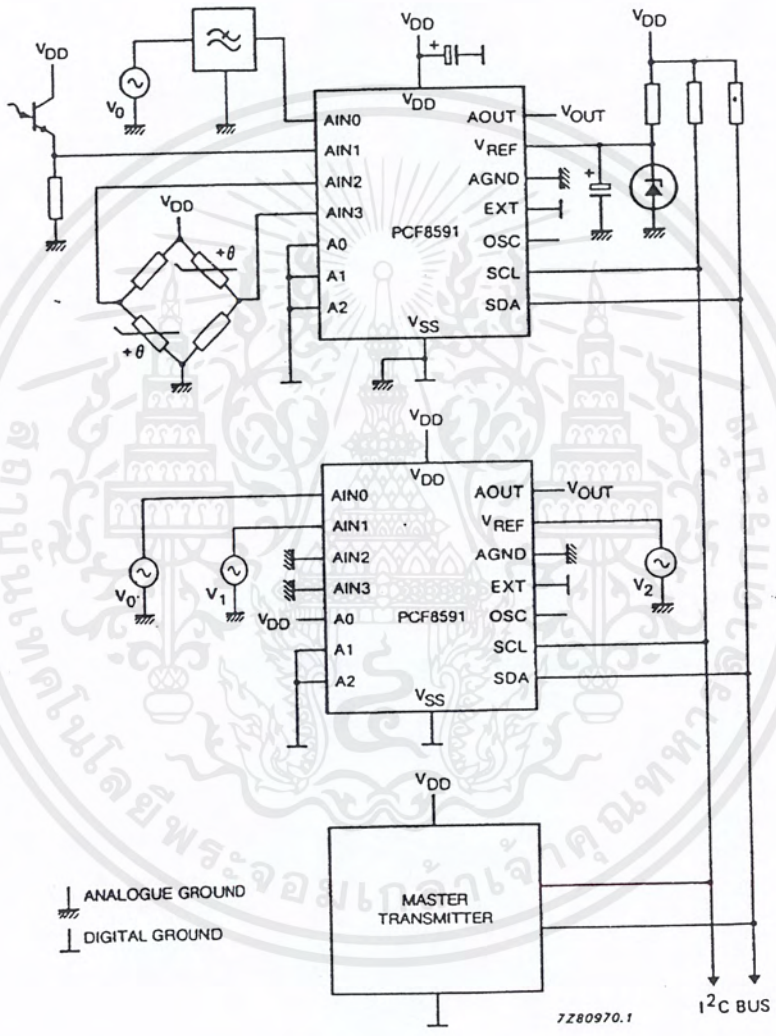


Fig.20 Application diagram.

Remote 8-bit I/O expander for I²C-bus PCF8574

1 FEATURES

- Operating supply voltage 2.5 to 6 V
- Low standby current consumption of 10 μA maximum
- I²C to parallel port expander
- Open-drain interrupt output
- 8-bit remote I/O port for the I²C-bus
- Compatible with most microcontrollers
- Latched outputs with high current drive capability for directly driving LEDs
- Address by 3 hardware address pins for use of up to 8 devices (up to 16 with PCF8574A)
- DIP16, or space-saving SO16 or SSOP20 packages.

2 GENERAL DESCRIPTION

The PCF8574 is a silicon CMOS circuit. It provides general purpose remote I/O expansion for most microcontroller families via the two-line bidirectional bus (I²C).

The device consists of an 8-bit quasi-bidirectional port and an I²C-bus interface. The PCF8574 has a low current consumption and includes latched outputs with high current drive capability for directly driving LEDs. It also possesses an interrupt line ($\overline{INT}$) which can be connected to the interrupt logic of the microcontroller. By sending an interrupt signal on this line, the remote I/O can inform the microcontroller if there is incoming data on its ports without having to communicate via the I²C-bus. This means that the PCF8574 can remain a simple slave device.

The PCF8574 and PCF8574A versions differ only in their slave address as shown in Fig.9.

3 ORDERING INFORMATION

TYPE NUMBER	PACKAGE		
	NAME	DESCRIPTION	VERSION
PCF8574P; PCF8574AP	DIP16	plastic dual in-line package; 16 leads (300 mil)	SOT38-1
PCF8574T; PCF8574AT	SO16	plastic small outline package; 16 leads; body width 7.5 mm	SOT162-1
PCF8574TS	SSOP20	plastic shrink small outline package; 20 leads; body width 4.4 mm	SOT266-1

4 BLOCK DIAGRAM

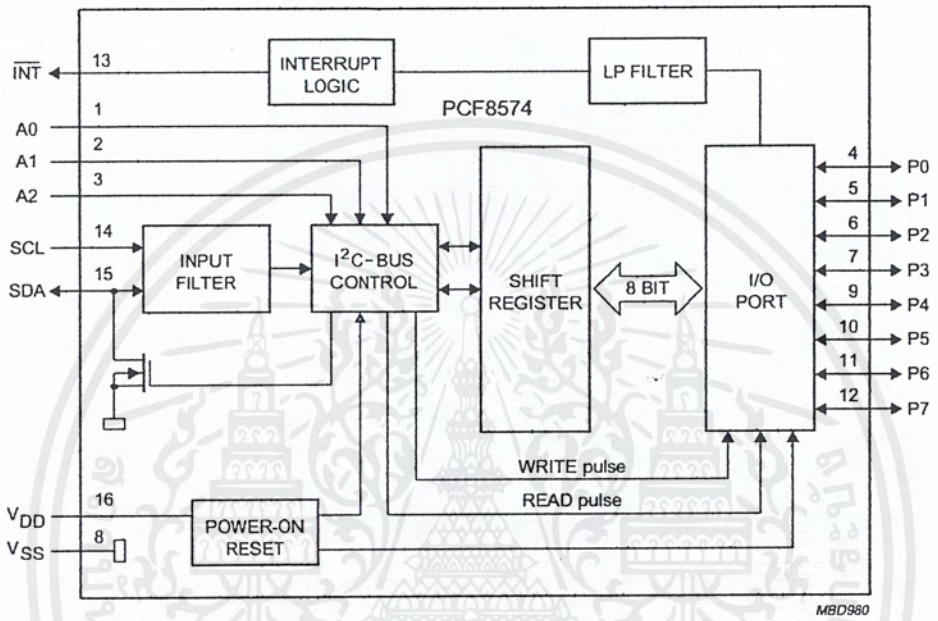


Fig.1 Block diagram (SOT38-1 and SOT162-1).

5 PINNING

SYMBOL	PIN		DESCRIPTION
	DIP16; SO16	SSOP20	
A0	1	6	address input 0
A1	2	7	address input 1
A2	3	9	address input 2
P0	4	10	quasi-bidirectional I/O 0
P1	5	11	quasi-bidirectional I/O 1
P2	6	12	quasi-bidirectional I/O 2
P3	7	14	quasi-bidirectional I/O 3
V_{SS}	8	15	supply ground
P4	9	16	quasi-bidirectional I/O 4
P5	10	17	quasi-bidirectional I/O 5
P6	11	19	quasi-bidirectional I/O 6
P7	12	20	quasi-bidirectional I/O 7
$\overline{INT}$	13	1	interrupt output (active LOW)
SCL	14	2	serial clock line
SDA	15	4	serial data line
V_{DD}	16	5	supply voltage
n.c.	–	3	not connected
n.c.	–	8	not connected
n.c.	–	13	not connected
n.c.	–	18	not connected

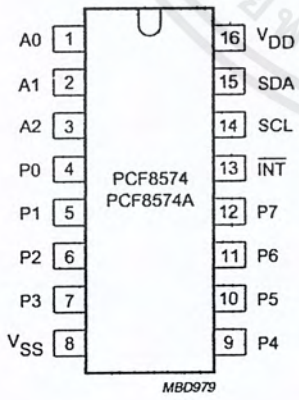


Fig.2 Pin configuration (DIP16; SO16).

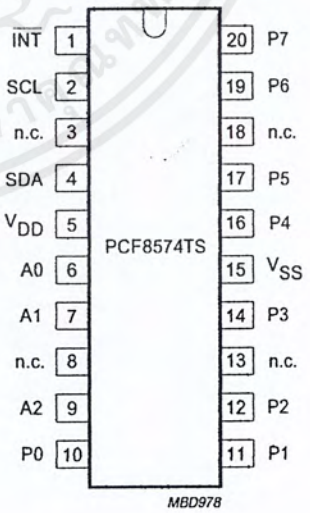


Fig.3 Pin configuration (SSOP20).

6 CHARACTERISTICS OF THE I²C-BUS

The I²C-bus is for 2-way, 2-line communication between different ICs or modules. The two lines are a serial data line (SDA) and a serial clock line (SCL). Both lines must be connected to a positive supply via a pull-up resistor when connected to the output stages of a device. Data transfer may be initiated only when the bus is not busy.

6.1 Bit transfer

One data bit is transferred during each clock pulse. The data on the SDA line must remain stable during the HIGH period of the clock pulse as changes in the data line at this time will be interpreted as control signals (see Fig.4).

6.2 Start and stop conditions

Both data and clock lines remain HIGH when the bus is not busy. A HIGH-to-LOW transition of the data line, while the clock is HIGH is defined as the start condition (S).
A LOW-to-HIGH transition of the data line while the clock is HIGH is defined as the stop condition (P) (see Fig.5).

6.3 System configuration

A device generating a message is a 'transmitter', a device receiving is the 'receiver'. The device that controls the message is the 'master' and the devices which are controlled by the master are the 'slaves' (see Fig.6).

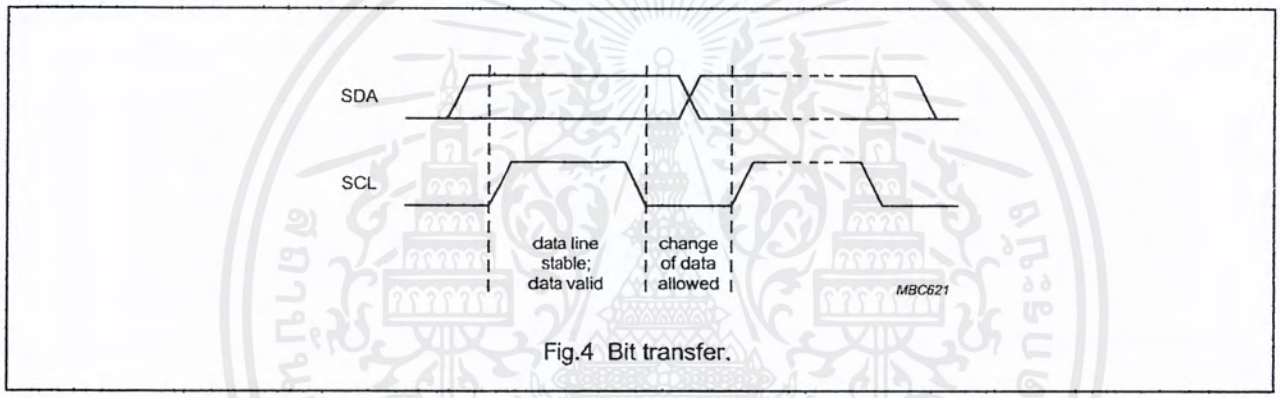


Fig.4 Bit transfer.

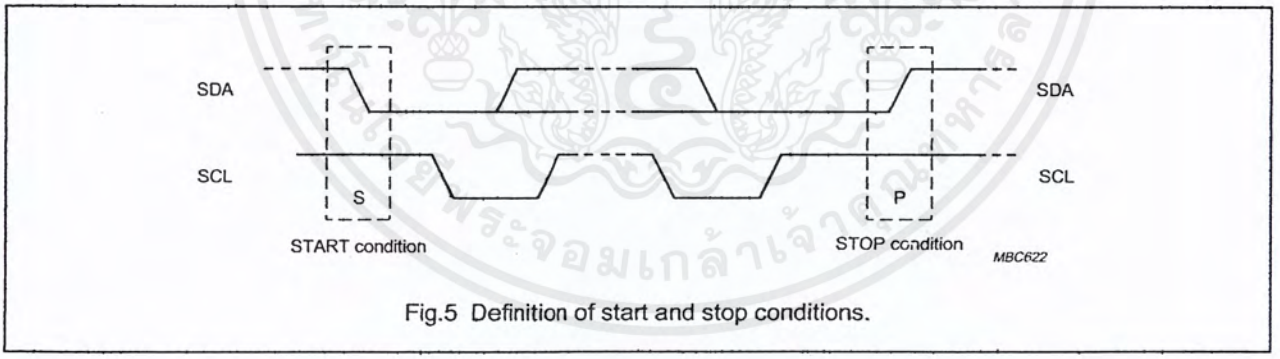


Fig.5 Definition of start and stop conditions.

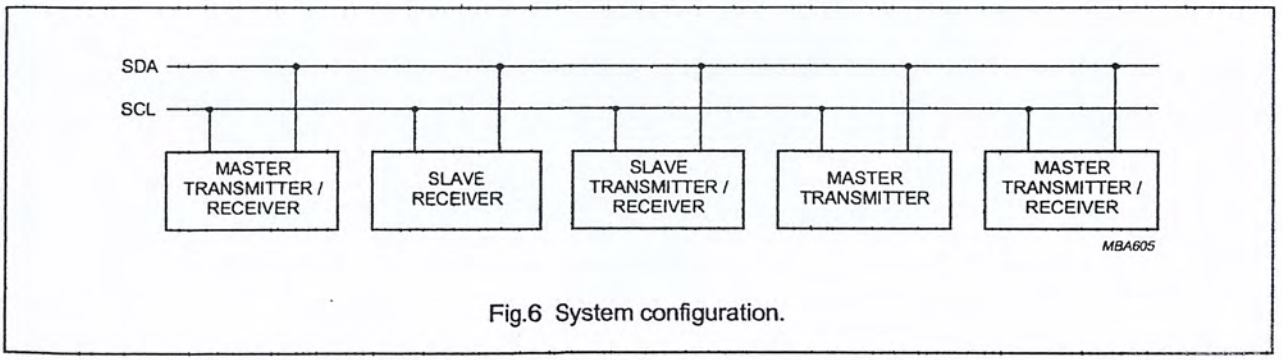


Fig.6 System configuration.

6.4 Acknowledge

The number of data bytes transferred between the start and the stop conditions from transmitter to receiver is not limited. Each byte of eight bits is followed by one acknowledge bit. The acknowledge bit is a HIGH level put on the bus by the transmitter whereas the master generates an extra acknowledge related clock pulse.

A slave receiver which is addressed must generate an acknowledge after the reception of each byte. Also a master must generate an acknowledge after the reception of each byte that has been clocked out of the slave transmitter. The device that acknowledges has to pull down the SDA line during the acknowledge clock pulse, so that the SDA line is stable LOW during the HIGH period of the acknowledge related clock pulse, set-up and hold times must be taken into account.

A master receiver must signal an end of data to the transmitter by **not** generating an acknowledge on the last byte that has been clocked out of the slave. In this event the transmitter must leave the data line HIGH to enable the master to generate a stop condition.

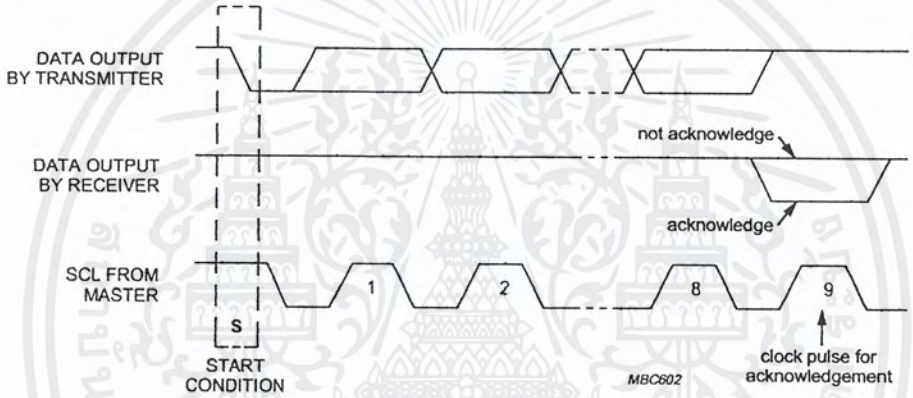


Fig.7 Acknowledgment on the I²C-bus.

7 FUNCTIONAL DESCRIPTION

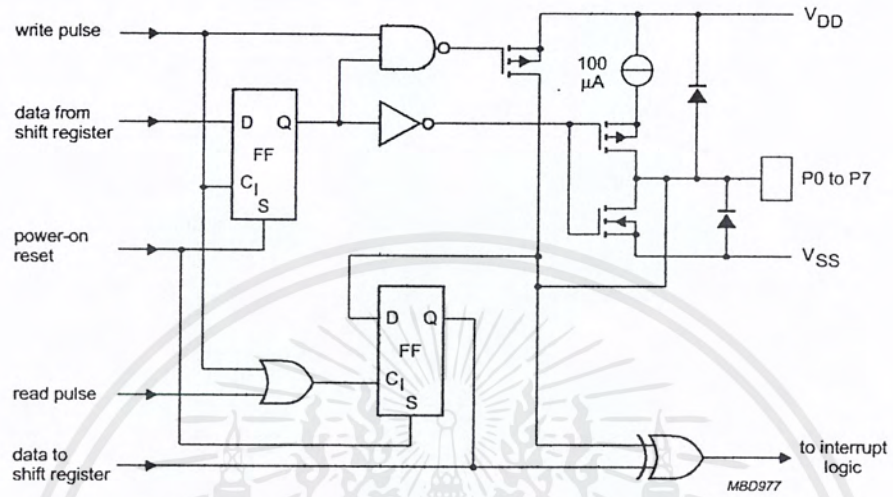
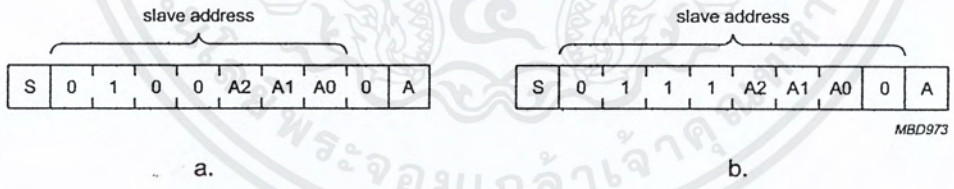


Fig.8 Simplified schematic diagram of each I/O.

7.1 Addressing

For addressing see Figs 9, 10 and 11.



(a) PCF8574.
(b) PCF8574A.

Fig.9 PCF8574 and PCF8574A slave addresses.

Each of the PCF8574's eight I/Os can be independently used as an input or output. Input data is transferred from the port to the microcontroller by the READ mode (see Fig.11). Output data is transmitted to the port by the WRITE mode (see Fig.10).

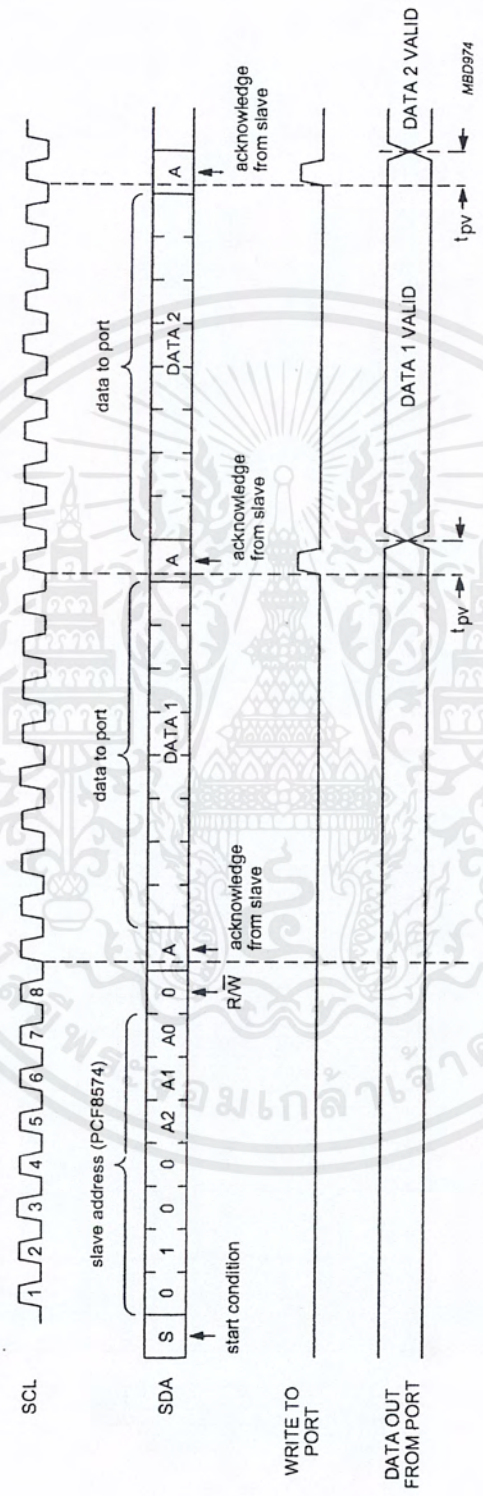
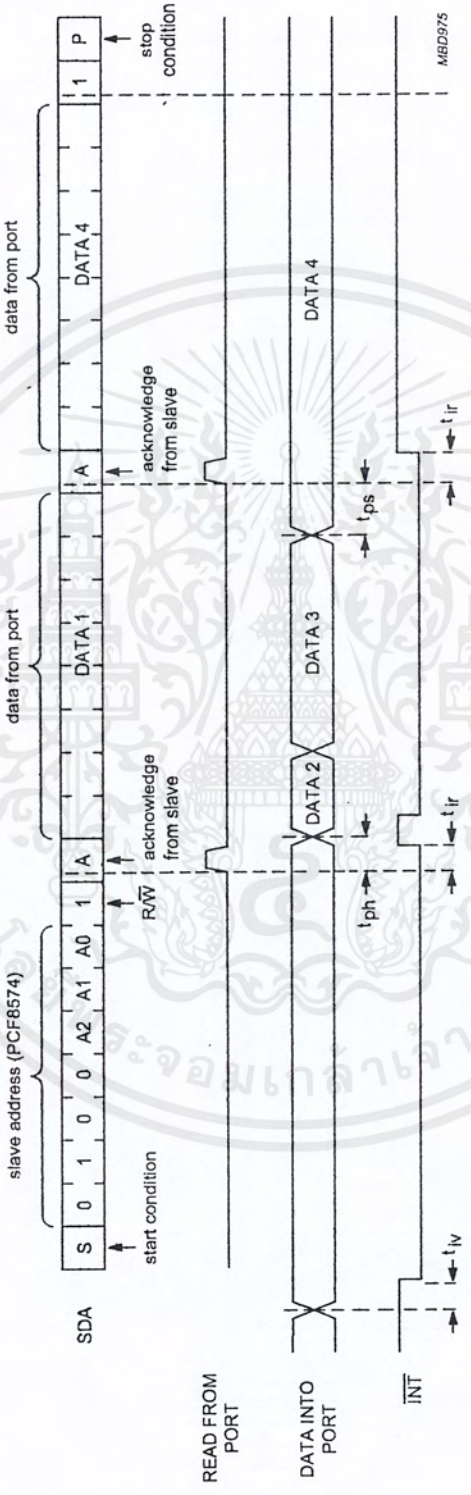


Fig.10 WRITE mode (output).



A LOW-to-HIGH transition of SDA, while SCL is HIGH is defined as the stop condition (P). Transfer of data can be stopped at any moment by a stop condition. When this occurs, data present at the last acknowledge phase is valid (output mode). Input data is lost.

Fig.11 READ mode (input).

7.2 Interrupt (see Figs 12 and 13)

The PCF8574 provides an open drain output ($\overline{INT}$) which can be fed to a corresponding input of the microcontroller. This gives these chips a type of master function which can initiate an action elsewhere in the system.

An interrupt is generated by any rising or falling edge of the port inputs in the input mode. After time t_{iv} the signal $\overline{INT}$ is valid.

Resetting and reactivating the interrupt circuit is achieved when data on the port is changed to the original setting or data is read from or written to the port which has generated the interrupt.

Resetting occurs as follows:

- In the READ mode at the acknowledge bit after the rising edge of the SCL signal
- In the WRITE mode at the acknowledge bit after the HIGH-to-LOW transition of the SCL signal
- Interrupts which occur during the acknowledge clock pulse may be lost (or very short) due to the resetting of the interrupt during this pulse.

Each change of the I/Os after resetting will be detected and, after the next rising clock edge, will be transmitted as $\overline{INT}$. Reading from or writing to another device does not affect the interrupt circuit.

7.3 Quasi-bidirectional I/Os (see Fig.14)

A quasi-bidirectional I/O can be used as an input or output without the use of a control signal for data direction. At power-on the I/Os are HIGH. In this mode only a current source to V_{DD} is active. An additional strong pull-up to V_{DD} allows fast rising edges into heavily loaded outputs. These devices turn on when an output is written HIGH, and are switched off by the negative edge of SCL. The I/Os should be HIGH before being used as inputs.

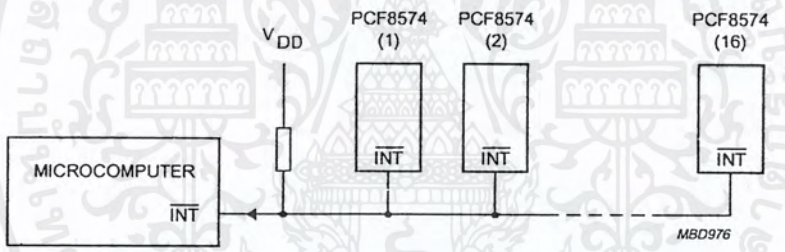


Fig.12 Application of multiple PCF8574s with interrupt.

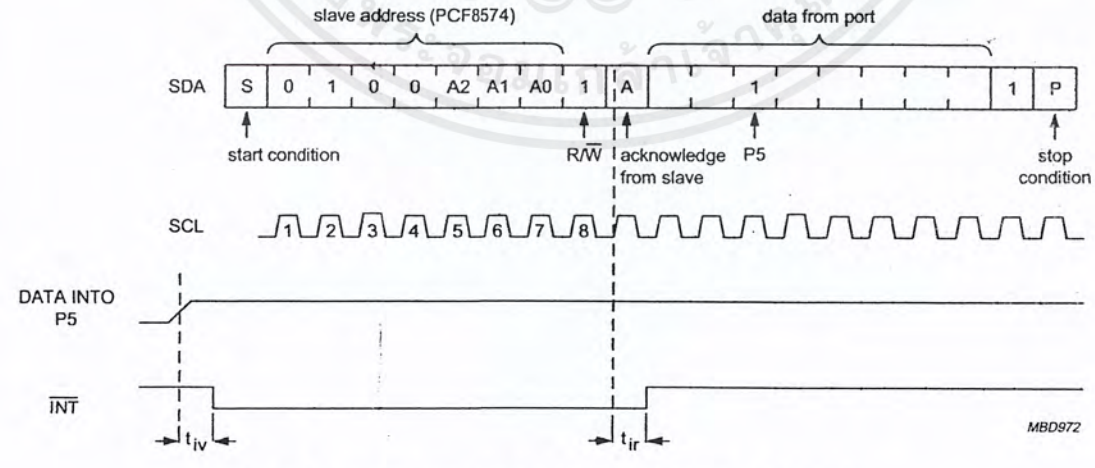


Fig.13 Interrupt generated by a change of input to I/O P5.

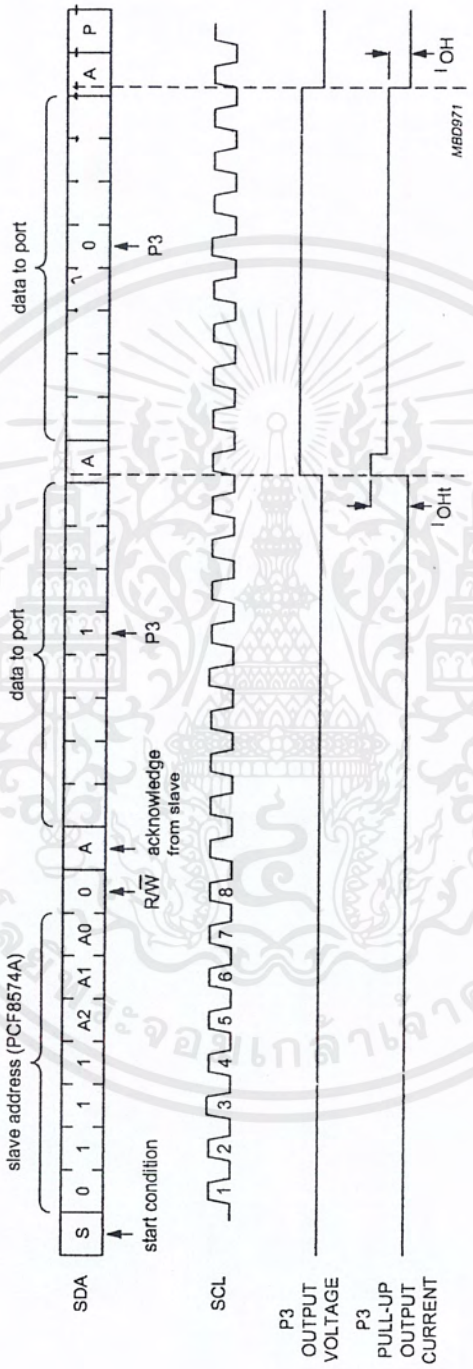


Fig.14 Transient pull-up current I_{OHt} while P3 changes from LOW-to-HIGH and back to LOW.

8 LIMITING VALUES

In accordance with the Absolute Maximum Rating System (IEC 134).

SYMBOL	PARAMETER	MIN.	MAX.	UNIT
V_{DD}	supply voltage	−0.5	+7.0	V
V_I	input voltage	V_{SS} − 0.5	V_{DD} + 0.5	V
I_I	DC input current	−	±20	mA
I_O	DC output current	−	±25	mA
I_{DD}	supply current	−	±100	mA
I_{SS}	supply current	−	±100	mA
P_{tot}	total power dissipation	−	400	mW
P_O	power dissipation per output	−	100	mW
T_{stg}	storage temperature	−65	+150	°C
T_{amb}	operating ambient temperature	−40	+85	°C

9 HANDLING

Inputs and outputs are protected against electrostatic discharge in normal handling. However, to be totally safe, it is desirable to take precautions appropriate to handling MOS devices. Advice can be found in Data Handbook IC12 under *"Handling MOS Devices"*.

10 DC CHARACTERISTICS

V_{DD} = 2.5 to 6 V; V_{SS} = 0 V; T_{amb} = −40 to +85 °C; unless otherwise specified.

SYMBOL	PARAMETER	CONDITIONS	MIN.	TYP.	MAX.	UNIT
Supply						
V_{DD}	supply voltage		2.5	−	6.0	V
I_{DD}	supply current	operating mode; V_{DD} = 6 V; no load; V_I = V_{DD} or V_{SS}; f_{SCL} = 100 kHz	−	40	100	µA
I_{stb}	standby current	standby mode; V_{DD} = 6 V; no load; V_I = V_{DD} or V_{SS}	−	2.5	10	µA
V_{POR}	Power-on reset voltage	V_{DD} = 6 V; no load; V_I = V_{DD} or V_{SS}; note 1	−	1.3	2.4	V
Input SCL; input/output SDA						
V_{IL}	LOW level input voltage		−0.5	−	+0.3V_{DD}	V
V_{IH}	HIGH level input voltage		0.7V_{DD}	−	V_{DD} + 0.5	V
I_{OL}	LOW level output current	V_{OL} = 0.4 V	3	−	−	mA
I_L	leakage current	V_I = V_{DD} or V_{SS}	−1	−	+1	µA
C_i	input capacitance	V_I = V_{SS}	−	−	7	pF

SYMBOL	PARAMETER	CONDITIONS	MIN.	TYP.	MAX.	UNIT
I/Os						
V_{IL}	LOW level input voltage		–0.5	–	$+0.3V_{DD}$	V
V_{IH}	HIGH level input voltage		$0.7V_{DD}$	–	$V_{DD} + 0.5$	V
$I_{IHL(max)}$	maximum allowed input current through protection diode	$V_I \geq V_{DD}$ or $V_I \leq V_{SS}$	–	–	±400	μA
I_{OL}	LOW level output current	V_{OL} = 1 V; V_{DD} = 5 V	10	25	–	mA
I_{OH}	HIGH level output current	$V_{OH} = V_{SS}$	30	–	300	μA
I_{OHt}	transient pull-up current	HIGH during acknowledge (see Fig.14); $V_{OH} = V_{SS}$; V_{DD} = 2.5 V	–	–1	–	mA
C_i	input capacitance		–	–	10	pF
C_o	output capacitance		–	–	10	pF
Port timing; $C_L \leq 100$ pF (see Figs 10 and 11)						
t_{pv}	output data valid		–	–	4	μs
t_{su}	input data set-up time		0	–	–	μs
t_h	input data hold time		4	–	–	μs
Interrupt $\overline{INT}$ (see Fig.13)						
I_{OL}	LOW level output current	V_{OL} = 0.4 V	1.6	–	–	mA
I_L	leakage current	$V_I = V_{DD}$ or V_{SS}	–1	–	+1	μA
TIMING; $C_L \leq 100$ pF						
t_{iv}	input data valid time		–	–	4	μs
t_{ir}	reset delay time		–	–	4	μs
Select inputs A0 to A2						
V_{IL}	LOW level input voltage		–0.5	–	$+0.3V_{DD}$	V
V_{IH}	HIGH level input voltage		$0.7V_{DD}$	–	$V_{DD} + 0.5$	V
I_{LI}	input leakage current	pin at V_{DD} or V_{SS}	–250	–	+250	nA

Note

1. The Power-on reset circuit resets the I²C-bus logic with $V_{DD} < V_{POR}$ and sets all I/Os to logic 1 (with current source to V_{DD}).

11 I²C-BUS TIMING CHARACTERISTICS

SYMBOL	PARAMETER	MIN.	TYP.	MAX.	UNIT
I²C-BUS TIMING (see Fig.15; note 1)					
f_{SCL}	SCL clock frequency	–	–	100	kHz
t_{SW}	tolerable spike width on bus	–	–	100	ns
t_{BUF}	bus free time	4.7	–	–	µs
$t_{SU;STA}$	START condition set-up time	4.7	–	–	µs
$t_{HD;STA}$	START condition hold time	4.0	–	–	µs
t_{LOW}	SCL LOW time	4.7	–	–	µs
t_{HIGH}	SCL HIGH time	4.0	–	–	µs
t_r	SCL and SDA rise time	–	–	1.0	µs
t_f	SCL and SDA fall time	–	–	0.3	µs
$t_{SU;DAT}$	data set-up time	250	–	–	ns
$t_{HD;DAT}$	data hold time	0	–	–	ns
$t_{VD;DAT}$	SCL LOW to data out valid	–	–	3.4	µs
$t_{SU;STO}$	STOP condition set-up time	4.0	–	–	µs

Note

1. All the timing values are valid within the operating supply voltage and ambient temperature range and refer to V_{IL} and V_{IH} with an input voltage swing of V_{SS} to V_{DD}.

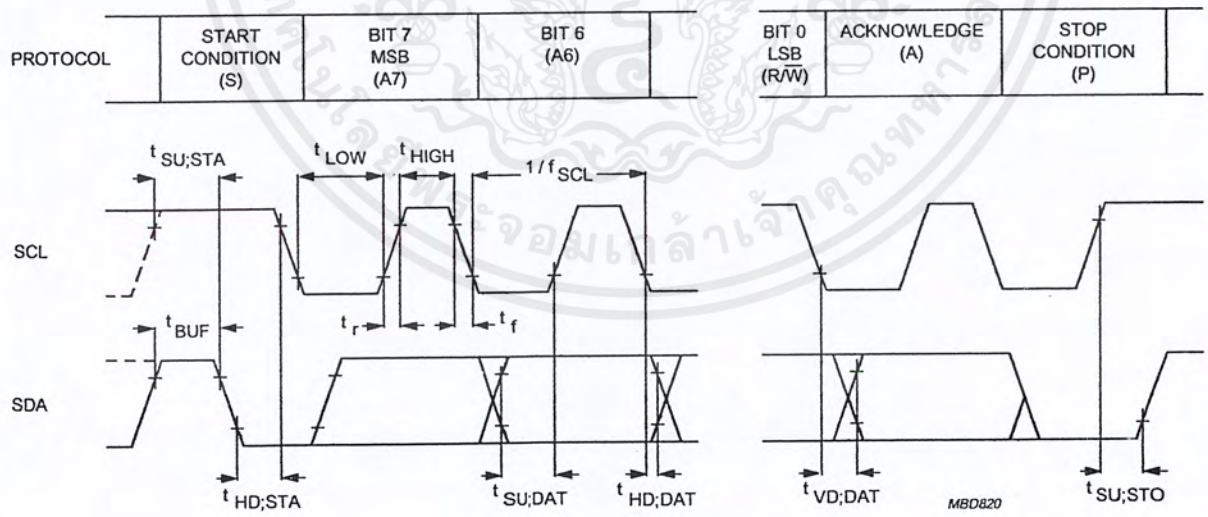


Fig.15 I²C-bus timing diagram.