การใช้งาน FPGA ร่วมกับไมโครคอนโทรลเลอร์ (MCS-51)

USING FPGA WITH MICROCONTROLLER (MCS-51)

นายวีระเดช วาดอินทร์

นายศราวุธ ปรัชญาคุณ

ปริญญานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิศวกรรมศาสตรบัณฑิต

สาขาวิชาวิศวกรรมการวัดคุม

ภาควิชาวิศวกรรมการวัดคุม คณะวิศวกรรมศาสตร์

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

ปีการศึกษา 2547

USING FPGA WITH MICROCONTROLLER (MCS-51)

WEERADECH WADOIN

SARAVUT PRATCHAYAKUN

A THESIS SUBMITTED IN PARTIAL FULFILLMENT

OF THE REQUIREMENT FOR THE DEGREE OF

BACHELOR OF ENGINEERING IN INSTRUMENTATION ENGINEERING

DEPARTMAENT OF INSTRUMENTATION ENGINEERING

FACULTY OF ENGINEERING

KING MONGKUT'S INSTITUTE OF TECHNOLOGY LADKRABANG

2004

ภาควิชาวิศวกรรมการวัดคุม

คณะวิศวกรรมศาสตร์

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

ใบรับรองปริญญานิพนธ์

หัวข้อปริญญานิพนธ์	การใช้งาน FPGA ร่วมกับไมโครคอนโทรลเลอร์ (MCS-51)			
	USING FPGA WITH MICROCONTROLLER (MCS-51)			
นักศึกษาผู้จัดทำ	นายวีระเดช	วาดอินทร์	รหัสประจำตัว	45015614
	นายศราวุธ	ปรัชญาคุณ	รหัสประจำตัว	45015617
ปริญญา	วิศวกรรมศาสตรบัณฑิต			
สาขาวิชา	วิศวกรรมการวัดคุม			
ปีการศึกษา	2547			

อาจารย์ผู้ควบคุมปริญญานิพนธ์	ลายมือชื่อ
รศ.พิพัฒน์ เลาหสงคราม	

วัน/เดือน/ปี ที่สอบ วันพุธที่ 20 เมษายน พ.ศ. 2548

สถานที่สอบ ณ ห้องสอบปริญญานิพนธ์ ภาควิชาวิศวกรรมการวัดคุม

ภาควิชารับรองแล้ว

(รศ. ประสิทธิ์ จุลเสรีวงศ์)

หัวหน้าภาควิชาวิศวกรรมการวัดคุม

หัวข้อปริญญานิพนธ์	การใช้งาน FPGA ร่วมกับไมโครคอนโทรลเลอร์ (MCS-51)	
	USING FPGA WITH MICROCONTROLLER (MCS-51)	
นักศึกษาผู้จัดทำ	นายวีระเดช	วาดอินทร์
	นายศราวุธ	ปรัชญาคุณ
อาจารย์ที่ปรึกษา	รศ.พิพัฒน์	เลาหสงคราม
ปีการศึกษา	2547	

บทคัดย่อ

ด้วยเทคโนโลยีที่มีอยู่ในปัจจุบันทำให้การออกแบบ และการใช้งานวงจรดิจิตอลไม่ต้องมีการต่อสายเพื่อเชื่อมวงจร และสามารถป้องกันความผิดพลาดที่เกิดจากสายที่ใช้ในการต่อวงจร เพราะเราสามารถรวมวงจรดิจิตอลไว้ในชิพตัวเดียวได้ และในการออกแบบวงจรด้วย FPGA โดยการใช้ Schematic หรือใช้ภาษาอธิบายการทำงานของวงจรจะทำได้สะดวกกว่า และที่สำคัญการออกแบบโดยวิธีนี้สามารถแก้ไขโมเดล (Model) หรือเปลี่ยนแปลงเทคโนโลยีได้สะดวกกว่า เพราะไม่ต้องวาดวงจรใหม่ นั่นคือการออกแบบโดยใช้ภาษาอธิบายฮาร์ดแวร์ จะทำให้โมเดลที่ได้ไม่ขึ้นกับเทคโนโลยีโดยโครงงานนี้จะเป็นการนำ FPGA ไปประยุกต์ใช้ร่วมกับไมโครคอนโทรลเลอร์ (MCS-51) เพื่อให้ติดต่อกับหน่วยความจำภายนอกได้

Thesis Title	Using FPGA with Microcontroller (MCS-51)	
Authors	Mr. Weeradech	Wadoin
	Mr. Saravut	Pratchayakun
Thesis Advisor	Assoc.Prof. Phiphut	Laohasongkram
Year	2004	

ABSTRACT

Also present technology ideal design and use digital circuit. It want not wire connect circuit and cannot protect error at error on wire at connect on circuit, because it can mix digital circuit in a ship and design circuit. FPGA circuit by use schematic or explain work language of circuit so will convenient make and important design by this design. It can correct error in printing model or change technology convenient because it make not new design circuit. So it is language explain hardware so your model not high technology by this project use FPGA in application mix with microcontroller (MCS-51) for link external memory

กิตติกรรมประกาศ

ปริญญานิพนธ์ฉบับนี้สำเร็จลุล่วงได้ด้วยดีเพราะได้รับความเมตตาจาก รองศาสตราจารย์ พิพัฒน์ เลาหสงคราม ที่ได้ให้คำแนะนำแก่ผู้วิจัยตลอดมา อีกทั้งยังเอื้อเฟื้ออุปกรณ์และเครื่องมือต่างๆ ในการทำปริญญานิพนธ์นี้ผู้วิจัยรู้สึกซาบซึ้ง และขอกราบขอบพระคุณเป็นอย่างสูง

ขอขอบพระคุณ อาจารย์ภาควิชาวิศวกรรมการวัดคุมทุกท่าน ที่ให้คำแนะนำอันเป็นประโยชน์ต่อการทำปริญญานิพนธ์ฉบับนี้

ขอกราบขอบพระคุณ คุณพ่อ คุณแม่ อันเป็นที่รักยิ่ง ที่คอยห่วงใยและให้การสนับสนุนในการศึกษา รวมทั้งขอขอบคุณเพื่อนๆ ภาควิศวกรรมการวัดคุม และภาควิศวกรรมคอมพิวเตอร์ที่คอยเป็นกำลังใจ และคอยช่วยเหลือกันมาตลอด

คุณค่าและประโยชน์อันพึงมีจากปริญญานิพนธ์ฉบับนี้ ผู้วิจัยขอมอบแด่ผู้มีพระคุณทุกท่าน

คณะผู้จัดทำ

สารบัญ

สารบัญ(ต่อ)

สารบัญ(ต่อ)

สารบัญตาราง

สารบัญภาพ

สารบัญภาพ(ต่อ)

สารบัญภาพ(ต่อ)

บทที่ 1
บทนำ

1.1 ความเป็นมาและเหตุจูงใจของการวิจัย

เนื่องจากในปัจจุบันการใช้งานไมโครคอนโทรลเลอร์ (MCS-51) นั้นถูกใช้งานอย่างแพร่หลาย ซึ่งบางครั้งอาจจะต้องใช้งานร่วมกับวงจรดิจิตอล ซึ่งการออกแบบ และใช้งานวงจรดิจิตอลนั้นจะต้องมีการต่อสายซึ่งค่อนข้างยุ่งยาก อาจมีความผิดพลาดเกิดขึ้นเนื่องจากตัวอุปกรณ์และสายที่ใช้ในการต่อวงจร และในการแก้ไขข้อผิดพลาดจะทำได้ค่อนข้างยาก ในปัจจุบันการออกแบบวงจรดิจิตอลทำได้สะดวกขึ้น เนื่องจากมีอุปกรณ์ซึ่งเรียกว่า FPGA (Field Programmable Gate Array) ซึ่งสามารถออกแบบวงจรรวมดิจิตอลให้อยู่ภายในชิพตัวเดียวได้ด้วยการเขียนผังวงจร ภาพวงจร (Schematic) การเขียนไดอะแกรมการเปลี่ยนสถานะ (State Diagram) การเขียนภาษาบรรยายเชิงพฤติกรรม (HDL) และวิธีอื่นๆ ขึ้นอยู่กับความสามารถของโปรแกรมที่ใช้ออกแบบ ดังนั้นในโครงงานนี้จะเป็นการนำ FPGA มาแทนที่วงจรดิจิตอลต่างๆ ที่ออกแบบขึ้นเพื่อใช้งานร่วมกับไมโครคอนโทรลเลอร์ (MCS-51)

1.2 วัตถุประสงค์ของปริญญานิพนธ์

ปริญญานิพนธ์นี้จะเป็นการศึกษาและออกแบบวงจรรวมดิจิตอลภายใน FPGA เพื่อนำไปใช้งานร่วมกับไมโครคอนโทรลเลอร์ (MCS-51) โดยวงจรภายใน FPGA จะทำหน้าที่เป็นวงจรสลับบัสข้อมูลของไมโครคอนโทรลเลอร์ (MCS-51) ในการติดต่อกับหน่วยความจำข้อมูลภายนอก เพื่อให้สามารถทำงานในลักษณะที่เรียกว่า Emulator ได้

1.3 ขอบเขตของปริญญานิพนธ์

ปริญญานิพนธ์เล่มนี้จะกล่าวถึงการออกแบบวงจรรวมดิจิตอล รวมถึงการทดสอบวงจรรวมดิจิตอลที่ได้ออกแบบภายใน FPGA ในการเป็นตัวกลางซึ่งทำหน้าที่เป็นวงจรสลับบัสในการรับส่งข้อมูลระหว่างไมโครคอนโทรลเลอร์ (MCS-51) กับหน่วยความจำภายนอก

1.4 ขั้นตอนการศึกษา

การทำโครงการวิจัยในปริญญานิพนธ์ฉบับนี้ มีขั้นตอนการศึกษาเริ่มจากการศึกษาทฤษฎีที่เกี่ยวข้องเช่น การรับส่งข้อมูลอนุกรม การติดต่อหน่วยความจำโปรแกรม และหน่วยความจำข้อมูลภายนอกของไมโครคอนโทรลเลอร์ (MCS-51) ภาษา และวิธีการที่ใช้ในการออกแบบวงจรรวมดิจิตอลภายใน FPGA จากนั้นนำความรู้ที่ได้ศึกษาทั้งหมดมาออกแบบวงจรรวมภายใน FPGA

และออกแบบบอร์ดที่ใช้งานรวมกับ FPGA จากนั้นทำการทดสอบการทำงานร่วมกันของ FPGA กับบอร์ดที่ออกแบบขึ้นมา รวมถึงการสรุปผลที่ได้จากการทดลอง ปัญหาที่พบในการทำโครงการ และแนวทางในการนำไปพัฒนา

บทที่ 2
ทฤษฎีที่เกี่ยวข้อง

2.1 ทฤษฎีไมโครคอนโทรลเลอร์

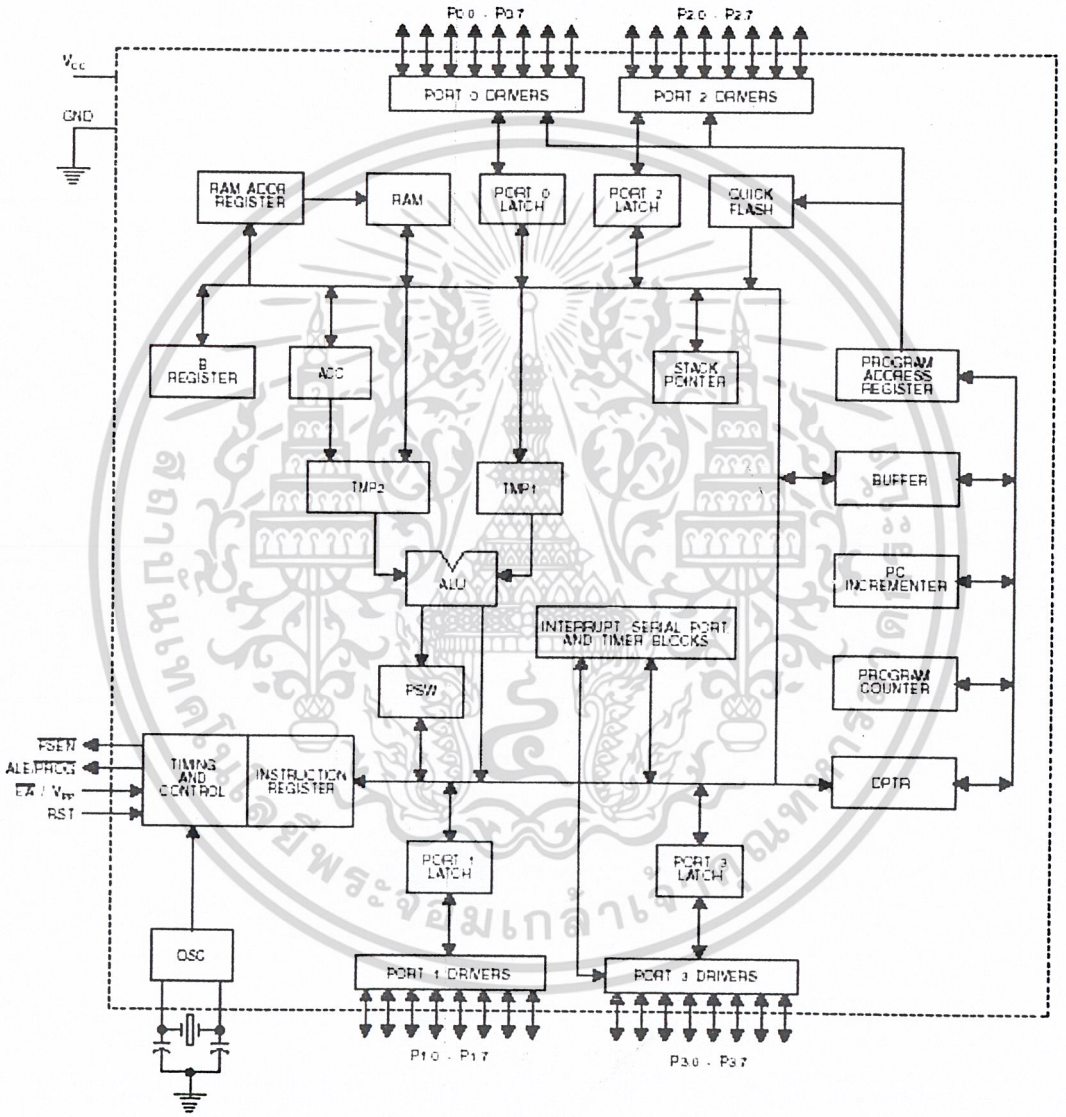
ไมโครคอนโทรลเลอร์(Microcontroller) เป็นอุปกรณ์อิเล็กทรอนิกส์แบบหนึ่งที่รวมเอาหน่วยประมวลผล หน่วยคำนวณทางคณิตศาสตร์และลอจิก วงจรรับสัญญาณอินพุต วงจรขับสัญญาณเอาท์พุต หน่วยความจำ วงจรกำเนิดสัญญาณนาฬิกาไว้ด้วยกัน ทำให้สามารถนำไปใช้งานแทนวงจรอิเล็กทรอนิกส์ที่ซับซ้อนได้เป็นอย่างดี ช่วยลดจำนวนอุปกรณ์และขนาดของระบบ ในขณะที่มีความสามารถสูงขึ้น ภายใต้งบประมาณที่เหมาะสม

ไมโครคอนโทรลเลอร์มาจากคำ 2 คำรวมกันคือ “ไมโคร” (micro) ซึ่งหมายถึงไมโครโปรเซสเซอร์ (microprocessor) ซึ่งเป็นอุปกรณ์ประมวลผลข้อมูลขนาดเล็ก ภายในประกอบด้วยหน่วยประมวลผลกลางหรือซีพียู (CPU: Central Processing Unit) หน่วยคำนวณทางคณิตศาสตร์และลอจิก (ALU: Arithmatic Logic Unitl) วงจรเชื่อมต่อหน่วยความจำ และวงจรสัญญาณนาฬิกา อีกคำหนึ่งคือคำว่า “คอนโทรลเลอร์” (controller) หมายถึง อุปกรณ์ควบคุม ดังนั้นไมโครคอนโทรลเลอร์จึงเป็นอุปกรณ์ที่ใช้ในการควบคุม โดยที่สามารถเขียนโปรแกรมเพื่อกำหนดภาพแบบการควบคุมได้อย่างอิสระ

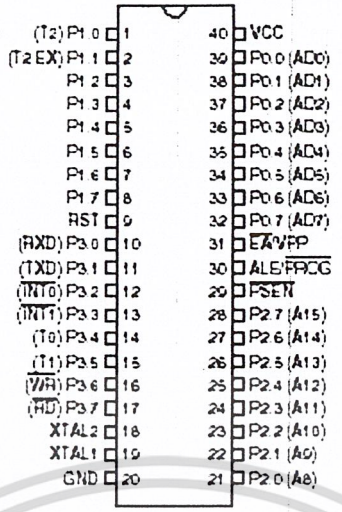
2.1.1 คุณสมบัติของไมโครคอนโทรลเลอร์ ตระกูล MCS-51 อนุกรม AT89Cxx

- เป็นไมโครคอนโทรลเลอร์ที่ใช้ซีพียูขนาด 8 บิต
- ภายในมีหน่วยความจำโปรแกรมเป็นแบบแฟลชสามารถลบและเขียนใหม่ได้พันครั้ง
- หน่วยความจำข้อมูลพื้นฐานเป็นหน่วยความจำแบบแรม ในบางเบอร์จะมีหน่วยความจำแบบ EEPROM เพิ่มเติม
- ขาพอร์ตเป็นแบบสองทิศทาง สามารถใช้งานเป็นได้ทั้งอินพุตและเอาท์พุต
- มีวงจรสื่อสารอนุกรมแบบ Full Duplex
- Timer/Counter ขนาด 16 บิตอย่างน้อย 2 ตัว
- สามารถรองรับแหล่งกำเนิด interrupt ได้ 6 ประเภท
- สามารถขยายหน่วยความจำภายนอกเพิ่มเติมได้สูงสุด 64 Kbytes
- มีวงจรกำเนิดสัญญาณนาฬิกาอยู่ภายในชิพ

ในภาพที่ 2-1 เป็นโครงสร้างพื้นฐานของไมโครคอนโทรลเลอร์ MCS-51 ในอนุกรม AT89Cxx จะเห็นได้ว่าโครงสร้างของ AT89Cxx จะเหมือนกับไมโครคอนโทรลเลอร์ตระกูล MCS-51 พื้นฐาน หากแต่แตกต่างกันเฉพาะหน่วยความจำโปรแกรมแบบแฟลชที่เพิ่มเติมเข้ามา หากเป็นไมโครคอนโทรลเลอร์ในอนุกรม 87xx หน่วยความจำโปรแกรมภายในจะเป็นแบบ EEPROM และบางเบอร์สามารถโปรแกรมได้เพียงครั้งเดียว



ภาพที่ 2.1 แสดงรายละเอียดโครงสร้างหลักของไมโครคอนโทรลเลอร์ MCS-51 แบบแฟลชของ Atmel



ภาพที่ 2.2 แสดงการจัดขามาตรฐานของไมโครคอนโทรลเลอร์ MCS-51 ในอนุกรม AT89C5x

2.1.2 การใช้งาน Timer/counter

การใช้งาน Timer/counter นั้นมีรีจิสเตอร์ที่ต้องใช้ 2 ตัวคือ TMOD และ TCON โดยที่ TMOD เป็นรีจิสเตอร์ขนาด 8 บิตไม่สามารถเข้าถึงระดับบิตได้ แบ่งการทำงานออกเป็น 2 ส่วนคือ 4 บิตล่างใช้เลือกโหมดการทำงาน Timer/Counter 0 และ 4 บิตบนใช้เลือกโหมด Timer/Counter 1 ในการทำงานของ Timer นั้นจะเริ่มการนับตั้งแต่ 0000H จนถึง FFFFH (ในกรณี 16 บิต) และเมื่อเปลี่ยนจาก FFFFH เป็น 0000H จะเกิดการโอเวอร์โฟลขึ้น

Gate	C/T	M1	M0	Gate	C/T	M1	M0

ภาพที่ 2.3 แสดงบิตต่างๆ ในรีจิสเตอร์ TMOD

โดยที่ Gate ใช้เลือกลักษณะการควบคุมการทำงานของ Timer/Counter

C/T ใช้เลือกการทำงานระหว่าง Timer/Counter

M1,M0 ใช้เลือกโหมดการทำงานของ Timer/Counter

"00" เลือกการทำงานโหมด Timer/Counter 13 บิต

"01" เลือกการทำงานโหมด Timer/Counter 16 บิต

"10" เลือกการทำงานโหมด Timer/Counter 8 บิตแบบตั้งค่าอัตโนมัติ

"11" สำหรับ Timer0 เลือกให้ทำงานในโหมด Timer/Counter แยกส่วน โดยแยกออกเป็น Timer/Counter 8 บิต 2 ตัว ริจิสเตอร์ TL0 จะได้รับการควบคุมการเปิดปิดจาก

ปิดจากบิต TR0 ในรีจิสเตอร์ TCON และรีจิสเตอร์ TH0 ซึ่งเป็น Timer/Counter 8 บิตอีกตัวหนึ่ง จะได้รับการควบคุมจากบิต TR1 ในรีจิสเตอร์ TCON ในกรณีของ Timer1 เป็นการสั่งให้ Timer/Counter1 หยุดทำงาน

ส่วนรีจิสเตอร์ TCON ใช้ควบคุมการทำงานของ Timer/Counter เป็นรีจิสเตอร์ขนาด 8 บิต สามารถเข้าถึงระดับบิตได้ มีโครงสร้างดังภาพ

TF1	TR1	TF0	TR0	IE1	IT1	IE0	IT0

ภาพที่ 2.4 แสดงบิตต่างๆ ในรีจิสเตอร์ TCON

โดยที่ TF1/0 แสดงการโอเวอร์โฟลว์ของ Timer 1, 0 จะเซตโดย hardware และเคลียร์โดย Software

TR1/0 ควบคุมการเปิดปิด Timer 1, 0 จะเซตและเคลียร์โดย software

IE1/0 แสดงการอินเตอร์รัพท์จาก INT1 และ INT0

IT1/0 บิตเลือกชนิดสัญญาณอินเตอร์รัพท์จากภายนอก INT1 และ INT0

การเริ่มและหยุด Timer นั้นสามารถควบคุมได้ที่บิต TRx ในรีจิสเตอร์ TCON โดยปกติแล้ว TRx จะเคลียร์หลังจากระบบถูกรีเซต ซึ่งเป็นการให้ Timer ไม่นับและ TRx นี้จะเซตให้เริ่มทำงานได้ด้วย SETB TRx และสามารถหยุดการทำงานได้ด้วยคำสั่ง CLR TRx

2.1.3 ทฤษฎีการสื่อสารผ่านพอร์ตอนุกรม

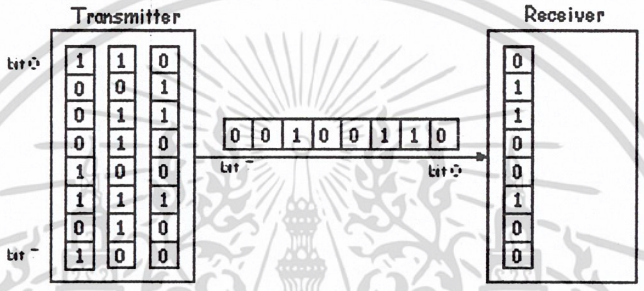
เป็นการรับส่งข้อมูลทีละบิต จนครบ 1 ไบต์ ถ้าต้องการส่งข้อมูล 1 ไบต์ คือ D_0 - D_7 อาจจะส่งบิต D_0 ออกไปก่อนแล้วตามด้วย D_1 ไปเรื่อยๆ จนถึง D_7 การส่งข้อมูลทั้งสองแบบมีข้อดีข้อเสียแตกต่างกันคือ การส่งข้อมูลแบบขนานสามารถส่งข้อมูลได้เร็ว คือ ส่งทีเดียวจะได้ข้อมูลครบ 1 ไบต์ แต่ถ้าต้องส่งเป็นระยะไกลๆ จะสิ้นเปลืองสายสัญญาณมาก ถ้าเป็นการส่งแบบอนุกรมเมื่อต้องการส่งข้อมูลเป็นระยะไกลๆ จะช่วยประหยัดสายสัญญาณเนื่องจากจะใช้สายอย่างน้อยเพียง 2 เส้น คือ สายสัญญาณกับสายกราวด์ แต่การรับส่งข้อมูลจะใช้เวลานานเนื่องจากเป็นการส่งทีละบิต ในบทนี้จะกล่าวถึงพื้นฐานการรับส่งข้อมูลแบบอนุกรมโดยเน้นที่ตัว MCS-51 เป็นสำคัญ

การสื่อสารข้อมูลแบบ Asynchronous

เป็นการรับและส่งข้อมูลโดยไม่จำเป็นต้องมีสัญญาณนาฬิการ่วมด้วยแต่จะใช้การกำหนดค่าอัตราเร็วในการรับส่งข้อมูลให้มีค่าเท่ากัน ซึ่งเรียกอัตราเร็วนี้ว่า Baud rate (bit/second) ภาพแบบของข้อมูลที่ใช้ในการรับส่งแบบ Asynchronous ประกอบด้วย 4 ส่วนด้วยกัน คือ

1. บิตเริ่มต้น (start bit) มีขนาด 1 บิต
2. บิตข้อมูลแบบอนุกรม มีขนาด 8 บิต
3. บิตตรวจสอบพาริตี้ (parity bit) มีขนาด 1 บิตหรือ ไม่มี
4. บิตปิดท้ายหรือบิตหยุด (stop bit) มีขนาด 1 บิต

เมื่อไมโครคอนโทรลเลอร์ต้องการจะรับส่งข้อมูลแบบอนุกรม ตัวไมโครคอนโทรลเลอร์จะส่งข้อมูลออกไปทางพอร์ตแบบขนานก่อน จากนั้นจะมีอุปกรณ์มาต่อที่พอร์ต เพื่อแปลงข้อมูลแบบขนานให้เป็นแบบอนุกรมอีกทีหนึ่ง (Parallel to Serial Conversion) ตัวแปลงข้อมูลนี้อาจจะพิจารณาง่ายๆว่าเป็น Shift Register ดังภาพ



ภาพที่ 2.5 แสดงการส่งข้อมูลแบบอนุกรม

สำหรับตัวรับข้อมูลแบบอนุกรมนั้นเมื่อตัวรับข้อมูลทำงานจะเป็นการรับเข้ามาใน shift register แล้วส่งข้อมูลให้ไมโครคอนโทรลเลอร์ แบบขนานอีกทีหนึ่ง (Serial to Parallel) ไมโครคอนโทรลเลอร์ในปัจจุบันจะมีตัวแปลง Parallel to Serial และ Serial to Parallel อยู่ในชิพไอซีเรียกว่า Universal Asynchronous Receiver Transmitter (UART) การส่งข้อมูลแบบอนุกรมนั้นจะต้องมีการเพิ่มเติมข้อมูลบางอย่างเข้าไป เพื่อให้การรับส่งข้อมูลสามารถทำงานได้ถูกต้องมากขึ้น โดยมีการเติมค่าบิตต่างๆ ลงไปตามภาพที่ 2.6

Start	D_0	D_1	D_2	D_3	D_4	D_5	D_6	D_7	Parity	Stop

ภาพที่ 2.6 แสดงบิตต่างๆ ของการรับส่งข้อมูลแบบอนุกรม

ถ้ามีการส่งข้อมูลแบบ 8 บิต จะต้องส่งบิตแรกออกไปก่อน เรียกว่า บิตเริ่มต้น (Start Bit) ถ้ามีการส่งข้อมูลหลายๆไบต์ออกมา บิตนี้จะเป็นตัวบอกว่ามีข้อมูลใหม่มาแล้ว โดยทั่วไปบิตเริ่มต้นมักมีระดับลอจิกเป็น "0" ต่อจากบิตเริ่มต้นจะเป็นข้อมูลบิต D_0 ถึง D_7 จากนั้นจะตามด้วยบิต

ตรวจสอบ และบิตหยุด เพื่อบอกการสิ้นสุดของข้อมูล บิตหยุดอาจมีจำนวนมากกว่า 1 บิตก็ได้ เช่น ½ บิต, 2 บิต

การสื่อสารแบบอนุกรมนี้ การกำหนดอัตราเร็วในการรับส่งข้อมูลจะบอกเป็นบิตต่อวินาที (bit per second : bps) ที่เรียกว่าอัตราบอดหรือบอดเรต (baud rate) โดยค่าที่ใช้กันทั่วไปมีหลายค่า ดังแสดงในตารางที่ 2.1

ตารางที่ 2.1 แสดงอัตราบอดเรต

อัตราบอด	ช่วงเวลาของแต่ละบิต
110	9.91 ms
150	6.67 ms
300	3.33 ms
600	1.67 ms
1200	0.833 ms
2400	0.417 ms
4800	0.208 ms
9600	0.104 ms
19200	0.052 ms

2.1.4 การรับส่งข้อมูลแบบอนุกรม

การรับส่งข้อมูลแบบอนุกรมกับไมโครคอนโทรลเลอร์ MCS-51 นั้นภายในชิพจะมี UART อยู่ในตัว ซึ่งเป็นข้อดีของไมโครคอนโทรลเลอร์ พอร์ตอนุกรมของ MCS-51 จะใช้ขา TXD และ RXD ในการรับส่งข้อมูลโดยขาทั้ง 2 จะอยู่ในพอร์ท 3 คือ P3.1 หรือขา 11 เป็น TXD และ P3.0 หรือขา 10 เป็น RXD พอร์ตอนุกรมของ MCS-51 สามารถทำงานแบบ full duplex ได้ คือสามารถรับส่งข้อมูลได้ในเวลาเดียวกัน โดยในการรับส่งข้อมูลจะมีบัฟเฟอร์สำหรับเก็บข้อมูลให้ใช้

Register ที่สำคัญในการรับส่งข้อมูล คือ SBUF และ SCON ซึ่งเป็นรีจิสเตอร์ที่อยู่ใน Special Function Register โดยที่ถ้าเขียนข้อมูลลงไปที่รีจิสเตอร์ Serial Port Buffer (SBUF) จะเป็นการส่งข้อมูลออกทางพอร์ตอนุกรม และถ้าอ่านข้อมูลจาก SBUF นี้จะเป็นการรับข้อมูลจากพอร์ตอนุกรม

SM0	SM1	SM2	REN	TB8	RB8	TI	RI

ภาพที่ 2.7 แสดงบิตต่างๆ ของรีจิสเตอร์ SCON

ตารางที่ 2.2 แสดงรายละเอียดการทำงานของ SCON Register

บิต	ชื่อบิต	การทำงาน
SCON.7	SM0	บิตเลือกโหมดการทำงาน
SCON.6	SM1	บิตเลือกโหมดการทำงาน
SCON.5	SM2	บิตเลือกการทำงานแบบ Single หรือ Multiprocessor 1 เลือก Multiprocessor ใช้ได้กับโหมด 2 และ 3 0 เลือก Single Processor ใช้ได้กับทุกโหมด
SCON.4	REN	บิตควบคุมให้รับหรือไม่รับข้อมูล 1 ให้รับข้อมูลได 0 ไม่รับข้อมูล
SCON.3	TB8	ข้อมูลบิตที่ 9 ที่ จะส่งออกไปในโหมด 2,3
SCON.2	RB8	ข้อมูลบิตที่ 9 จะรับเข้ามาในบิตนี้
SCON.1	TI	จะเป็น 1 เมื่อสิ้นสุดการส่งข้อมูล 1 ไบต์
SCON.0	RI	จะเป็น 1 เมื่อสิ้นสุดการรับข้อมูล 1 ไบต์

ในการจะรับส่งข้อมูลผ่านพอร์ทอนุกรมต้องทำการป้อนค่าให้กับรีจิสเตอร์ SCON ก่อน โดยการกำหนด บิต SM0 และ SM1 เพื่อเลือกโหมดการทำงานและลักษณะต่างๆของการรับและส่งข้อมูล

- โหมด 0 พอร์ตสื่อสารอนุกรม 8 บิต โดยการส่งจะเลื่อนออกทีละบิตโดยส่งบิต D0 ออกไปก่อนทางขา RxD และไม่มีการส่งบิตเริ่มต้นแต่จะส่ง ชิฟต์คล็อก (Shift clock) ทางขา TxD0(ความเร็ว 1/12 เท่าของสัญญาณนาฬิกาของ ซีพียู)
- โหมด 1 พอร์ตสื่อสารอนุกรม 10 บิต แบ่งเป็นข้อมูล 8 บิต บิตเริ่มต้น 1 บิต และบิตสิ้นสุด 1บิต และสามารถเปลี่ยนแปลงความเร็วในการส่งข้อมูลได้โดยขึ้นอยู่กับบิต SMOD ใน PCON และอัตราโอเวอร์โฟลว์ของไทม์เมอร์ 1
- โหมด 2 พอร์ตสื่อสารอนุกรม 11 บิต แบ่งเป็นข้อมูล 9 บิต บิตเริ่มต้น 1 บิต และบิตสิ้นสุด 1บิต (TB8 นิยมนำมาใช้ส่ง Parity Bit) ความเร็วในการรับส่งข้อมูลเท่ากับ 1/32และ 1/64

ของสัญญาณนาฬิกาของ ซีพียูโดยขึ้นกับ SMODใน PCONBaudRateโหมด 2=(1/32) (ความถี่สัญญาณนาฬิกา) เมื่อ SMOD=1BaudRateโหมด 2=(1/64)(ความถี่สัญญาณนาฬิกา) เมื่อ SMOD=0

- โหมด 3 พอร์ตสื่อสารอนุกรม 11 บิต แบ่งเป็นข้อมูล 9 บิต บิตเริ่มต้น 1 บิต และบิตสิ้นสุด 1 บิตเหมือนโหมด 2 ยกเว้นอัตราความเร็วจะขึ้นกับบิต SMOD ใน PCON และอัตราโอเวอร์โฟลว์ของไทม์เมอร์ 1,2

2.1.5 อัตราการส่งข้อมูลของพอร์ตอนุกรม

ในโหมด 0 และ 2 ไม่สามารถกำหนดอัตราการรับส่งข้อมูลได้ โดยในโหมด 0 จะมีอัตรา baud rate= ความถี่ oscillator หารด้วย 12 ในโหมด 1 จะมี 2 ค่าคือ ความถี่ oscillator หารด้วย 32 ที่ SMOD="0" และหารด้วย 64 ที่ SMOD="1" ซึ่งสามารถกำหนดได้ในรีจิสเตอร์ PCON บิตที่ 7

การคำนวณหาค่าอัตราบอดเรตที่กำหนดด้วย Timer 1 สามารถหาได้ด้วยสมการนี้

$$Buad\ Rate = 2^{SMOD}/32 \times ความถี่\ oscillator\ /(12 \times [256-TH1])$$

โดยที่ SMOD เป็นค่าของบิตภายในรีจิสเตอร์ PCON ซึ่งอาจมีค่าเป็น 0 หรือ 1

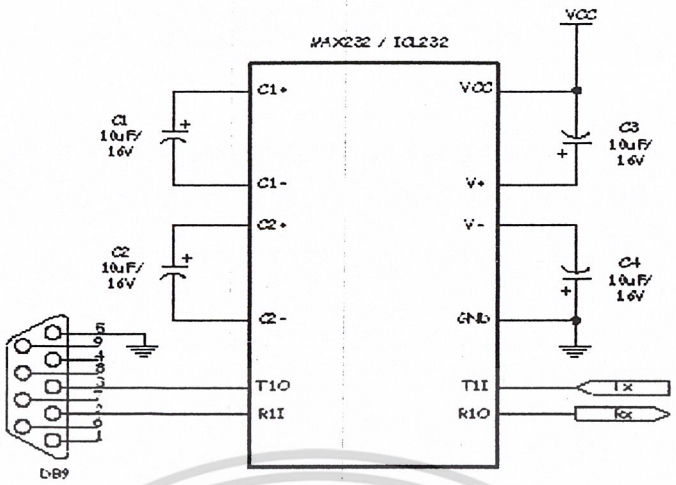
TH1 ค่าภายในรีจิสเตอร์ TH1 ใช้สำหรับ Reload ค่าของการนับเวลา

ภาพแบบทั่วไปของการหาค่า Baud Rate ในโหมด 1 และ 3 สามารถหาได้ดังนี้

$$Baud\ Rate = Timer1\ Overflow\ Rate\ /\ 32$$

ตารางที่ 2.3 แสดงตัวอย่างความถี่สัญญาณนาฬิกาที่ใช้กำหนด Baud Rate ค่าต่างๆ

ค่า Baud Rate	Crystal	SMOD โหมด	ค่าใน TH1	ค่า Baud Rate ที่ได้	Error
9,600	12.00	1	-7(F9H)	8,923	7%
9,600	11.059	0	-3(FDH)	9,600	0
2,400	11.059	0	-12(F4H)	2,400	0
1,200	11.059	0	-24(E8H)	1,200	0

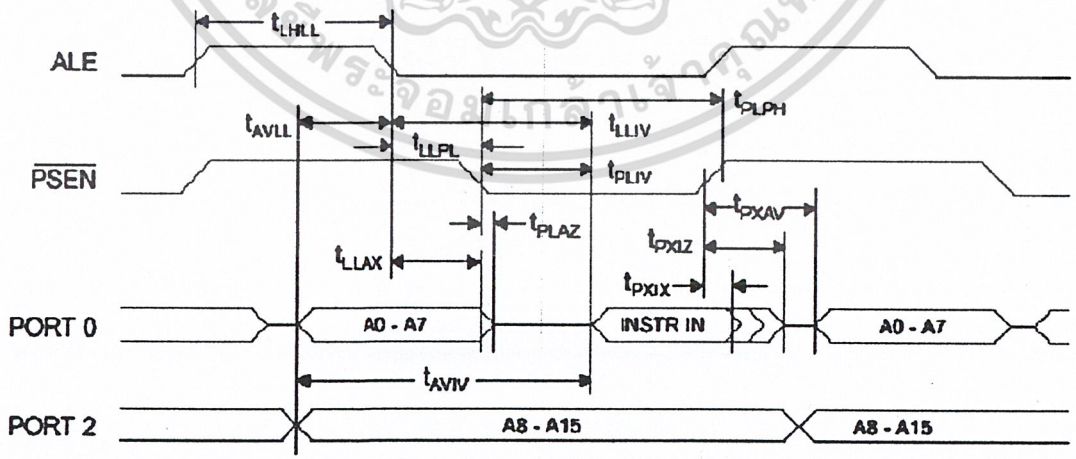


ภาพที่ 2.8 แสดงวงจรเชื่อมต่อ MAX232 เข้ากับพอร์ตอนุกรมของMCS-51

2.1.6 ไดอะแกรมเวลาและการต่อวงจรภายนอก

ปกติระบบคอมพิวเตอร์ทุกระบบจะต้องมี หน่วยอินพุต และหน่วยเอาท์พุตด้วย ดังนั้นจึงต้องมีภาพแบบการส่งข้อมูลระหว่างหน่วยต่างๆ เหล่านี้กับ CPU ใน CPU แต่ละตัวก็มีภาพแบบการส่งข้อมูลแตกต่างกันไป ดังนั้นเมื่อศึกษา CPU ให้เข้าใจมากขึ้น ก็จะต้องรู้ถึงภาพแบบการรับส่งข้อมูลนี้เราเรียกว่าไดอะแกรมเวลานั้นเอง

External Progam Memory Read Cycle



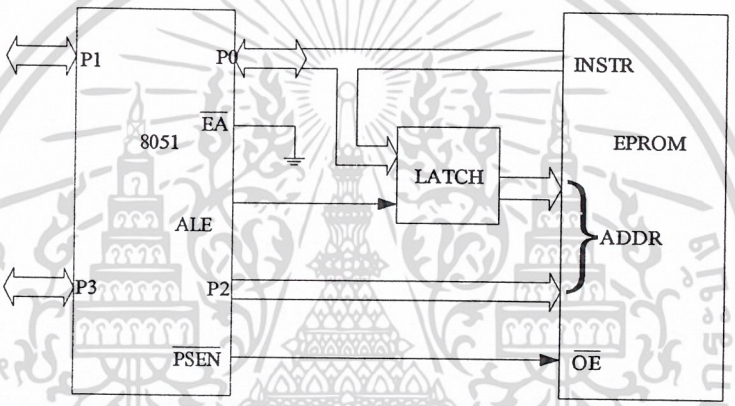
ภาพที่ 2.9 แสดงผังการทำงานของการอ่านโปรแกรมภายนอก

MCS -51 สามารถต่อกับอุปกรณ์ภายนอกได้ 2 กลุ่มดังนี้

-หน่วยความจำโปรแกรมภายนอก (ROM) ไดอะแกรมเวลามีเฉพาะการอ่านโปรแกรมจาก ROM เท่านั้น เพราะ ROM อ่านได้อย่างเดียวเขียนไม่ได้

-หน่วยความจำข้อมูลภายนอก(RAM) ไดอะแกรมเวลามีทั้งการอ่านและเขียนข้อมูล

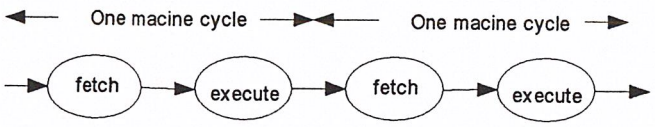
จากภาพที่ 2.9 ALE ทำงานที่ขอบขาลง จะเกิดขึ้นเมื่อต้องการใช้ PORT 0 เป็น A0-A7 ส่วน PSEN ทำงานที่ขอบขาขึ้น จะเกิดขึ้นเมื่อต้องการนำคำสั่งจากบัสข้อมูล (PORT 0) เข้าสู่ MCS -51 และสุดท้าย PORT 2 จะเปลี่ยนแปลงเมื่อใช้ PORT 0 เป็น A0-A7 เพราะว่า address นั้นมีทั้งหมด 16 บิต เมื่อใช้ PORT 2 เป็น A8-A15 ด้วย เพื่อให้เข้าใจมากขึ้นขอให้ดูวงจรในภาพที่ 2.10 ประกอบ



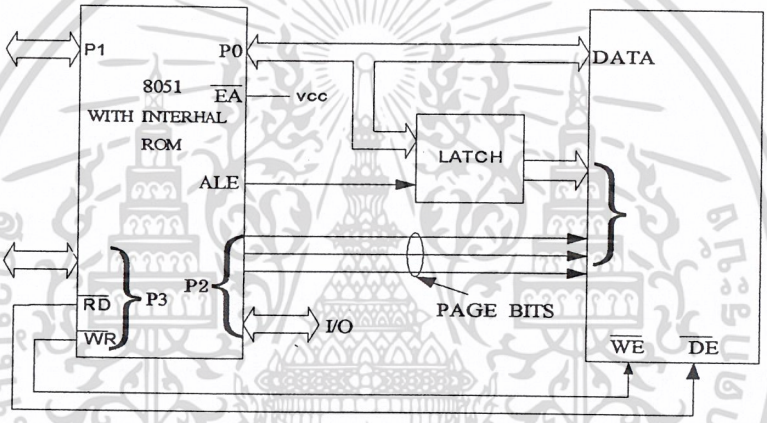
ภาพที่ 2.10 แสดงการต่อ MCS-51 กับหน่วยความจำโปรแกรมภายนอก

การต่อวงจรของ MCS-51 นั้น จะใช้ขา PORT 0 ต่อโดยตรงกับขา data bus (D0-D7) ของ RAM (Read-Only Memory) หรือ EPROM (Electrically Programmable Read-Only Memory) ซึ่งเป็นหน่วยความจำที่สามารถอ่านได้อย่างเดียว (ภายใน EPROM นี้จะเก็บโปรแกรมที่จะสั่งให้ MCS-51 ทำงาน) นอกจากนั้นขา PORT 0 ยังถูกต่อเข้ามาที่ LATCH อีกด้วยเพื่อใช้เป็นขา address (A0-A7) โดยมันจะใช้เป็นขา address เมื่อมีสัญญาณจากขา ALE เท่านั้น โดยมันจะนำ address จาก PORT 0 ไปวางให้ค้างไว้ที่เอาท์พุตของ LATCH จนกว่าจะมีการส่งสัญญาณจาก ALE เข้ามาอีกครั้งหนึ่ง PORT 2 จะมีการเปลี่ยนแปลง address (A8-A15) ในทันที เมื่อ ALE ส่งสัญญาณไปที่ LATCH และขา $\overline{\text{PSEN}}$ จะส่งสัญญาณไปที่ขา $\overline{\text{OE}}$ ของ EPROM เมื่อต้องการให้ EPROM ส่งข้อมูลออกมาสู่บัสข้อมูล (PORT 0) จากนั้นMCS-51 จะอ่านข้อมูลจาก PORT 0 เข้าไปเป็นคำสั่งนั่นเอง การทำงานตั้งแต่ MCS-51 ส่งสัญญาณ ALE ไปเลือก address และส่ง $\overline{\text{PSEN}}$ ไปอ่านข้อมูล จนกระทั่ง MCS-51 อ่านข้อมูลมาเก็บเป็นคำสั่ง กลุ่มการทำงานเหล่านี้เราเรียกว่าขบวนการเฟทซ์ (fetch) และเมื่อ MCS-51 นำคำสั่งนั้นไปถอดรหัสและทำคำสั่งนั้นจนสิ้นสุดการทำงานของคำสั่งเราเรียกกลุ่มการ

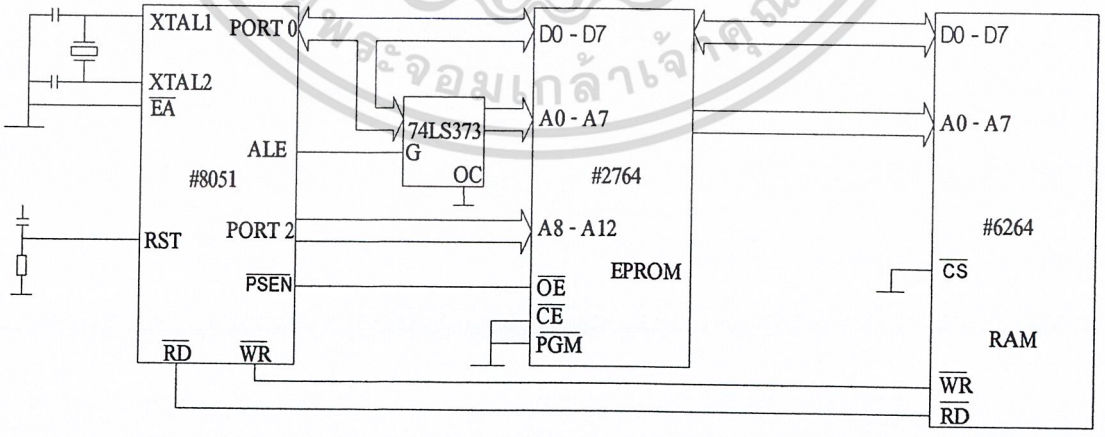
ทำงานนี้ว่า ขบวนการเอ็กซีคิ้ว (execute) เมื่อรวมขบวนการเฟทซ์และเอ็กซีคิ้วเข้าด้วยกันจะถูกเรียกว่า 1 แมซีนไซเคิล(one machine cycle) ขอให้ดูภาพที่ 2.11 ประกอบ



ภาพที่ 2.11 แสดงวงจรการทำงานของ 1 ไซเคิล



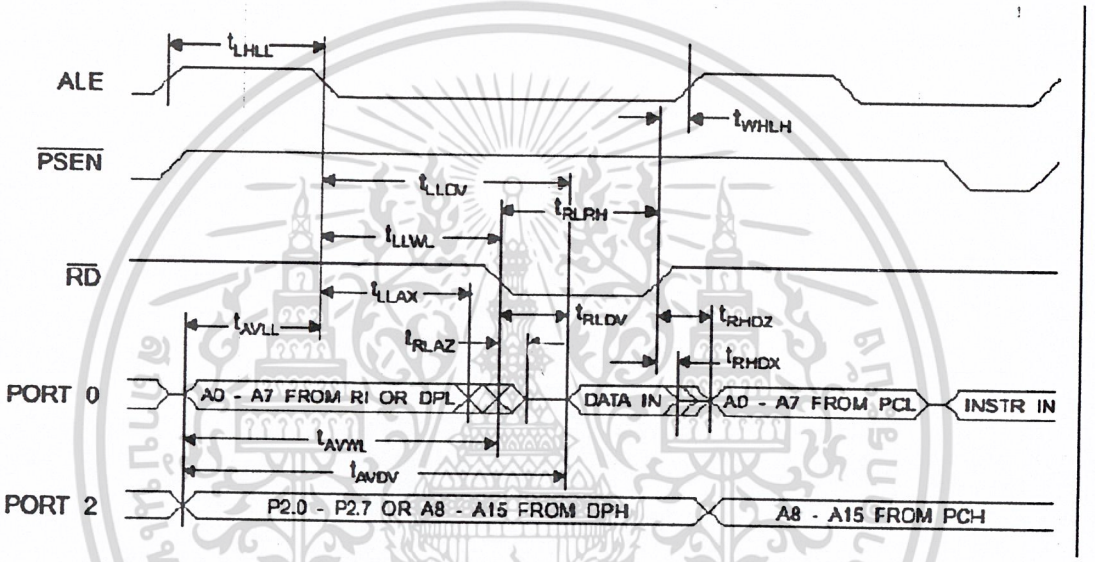
ภาพที่ 2.12 แสดงการต่อ MCS-51 กับหน่วยความจำข้อมูลภายนอก



ภาพที่ 2.13 แสดงการต่อ MCS-51 กับหน่วยความจำข้อมูลภายนอก (#6264)

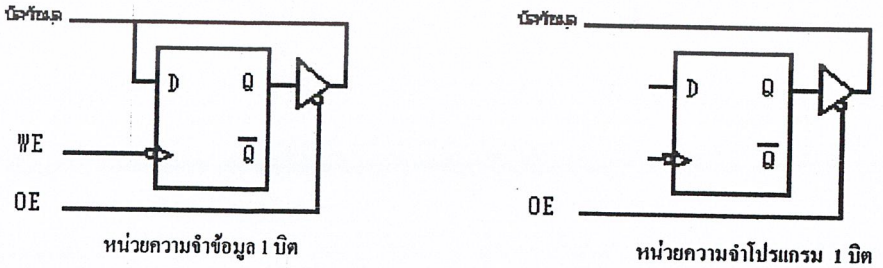
จากภาพที่ 2.13 จะแตกต่างกันระหว่างการต่อหน่วยความจำโปรแกรมภายนอก (#2764) และหน่วยความจำข้อมูลภายนอก (#6264) โดยถ้าเป็นหน่วยความจำโปรแกรมจะใช้ $\overline{\text{PSEN}}$ ต่อกับขา $\overline{\text{OE}}$ แต่ถ้าเป็นหน่วยความจำข้อมูลจะใช้ขา $\overline{\text{RD}}$ และ $\overline{\text{WR}}$ แทน เพราะหน่วยความจำข้อมูลต้องมีการอ่านและเขียน แต่หน่วยความจำโปรแกรมเขียนอย่างเดียว ส่วนผังการอ่านข้อมูลภายนอกแสดงดังภาพที่ 2.14

External Data Memory Read Cycle



ภาพที่ 2.14 แสดงผังการอ่านข้อมูลภายนอก

เพื่อให้เข้าใจความแตกต่างระหว่าง หน่วยความจำโปรแกรม (ROM) และหน่วยความจำข้อมูล (RAM) ขอให้ดูภาพที่ 2.15 ประกอบ

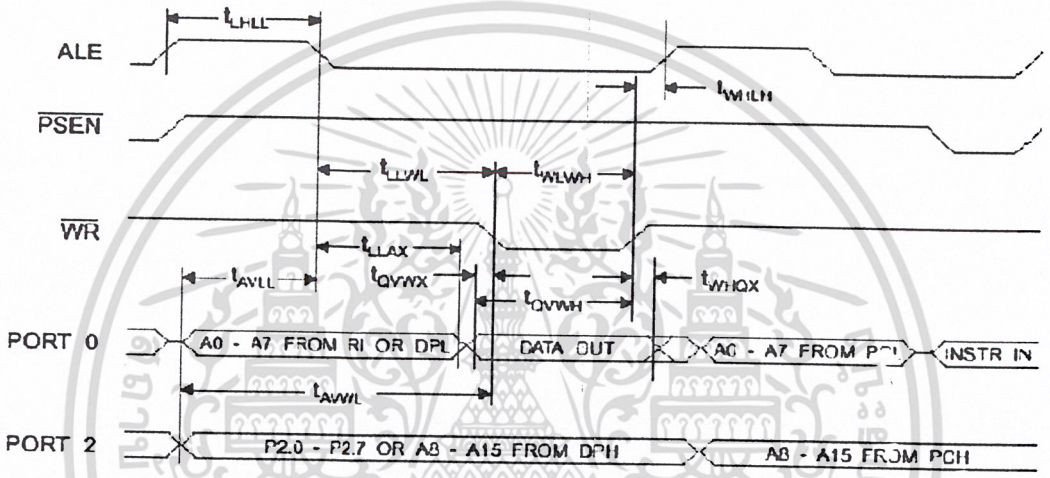


ก. หน่วยความจำข้อมูล (RAM) **ข.** หน่วยความจำโปรแกรม (ROM)

ภาพที่ 2.15 แสดงความแตกต่างของภายในหน่วยความจำข้อมูลและหน่วยความจำโปรแกรม

ภายในหน่วยความจำจะประกอบด้วยฟลิบฟลอบหลายๆ ตัวต่อเรียงกัน และแต่ละตัวนั้นจะถูกควบคุมจากขาสัญญาณภายนอก ซึ่งหน่วยความจำข้อมูลนั้น ภายในนั้นมีการทำงานคล้ายกับ D ฟลิบฟลอบ โดยใช้ขา WE ต่อกับขาสัญญาณนาฬิกา และขา OE เป็นขาควบคุมการเปิดให้เอาท์พุตออกสู่ภายนอก และขาสัญญาณ D จะต่อกับบัสข้อมูลส่วนหน่วยความจำโปรแกรมนั้นไม่มีข้อมูลเข้า เพราะไม่สามารถเขียนเข้าไปได้

External Data Memory Write Cycle



ภาพที่ 2.16 แสดงผังการเขียนข้อมูลภายนอก

จากผังการทำงานในภาพที่ 2.16 เป็นการเขียนข้อมูลภายนอก ผังการทำงานจะคล้ายกับภาพที่ 2.14 แต่ต่างกันที่เปลี่ยนจากขาสัญญาณ RD เป็นขาสัญญาณ WR เท่านั้น

2.1.7 กระบวนการอินเตอร์รัปต์

การอินเตอร์รัปต์ (interrupt) เป็นชื่อเรียกกระบวนการที่เข้ามาขัดจังหวะการทำงาน โดยปกติของไมโครคอนโทรลเลอร์ ในไมโครคอนโทรลเลอร์ MCS-51 สามารถตอบสนองการอินเตอร์รัปต์ที่เกิดขึ้นได้จาก 5 แหล่งกำเนิดสำหรับเบอร์ AT89C51 ประกอบด้วย การรับสัญญาณอินเตอร์รัปต์จากภายนอกผ่านทางขา INT0 และ INT1 สัญญาณอินเตอร์รัปต์จากไทเมอร์/เคาเตอร์ T0 และ T1 และสัญญาณอินเตอร์รัปต์จากพอร์ตอนุกรมภายในไมโครคอนโทรลเลอร์ ในขณะที่ไมโครคอนโทรลเลอร์ AT89C51 และอนุกรม AT89Sxx สามารถตอบสนองการอินเตอร์รัปต์ได้จาก 6 แหล่งกำเนิด โดยเพิ่มการรับสัญญาณอินเตอร์รัปต์จากไทเมอร์/เคาเตอร์ 2 อีกหนึ่งแหล่งกำเนิด

2.1.7.1 การจัดการอินเตอร์รัปต์ในไมโครคอนโทรลเลอร์ MCS-51

เมื่อมีการอินเตอร์รัปต์ในไมโครคอนโทรลเลอร์ MCS-51 เกิดขึ้น และมีการเอ็นเอเบิลการตอบสนองการอินเตอร์รัปต์ไว้ กระบวนการหลังจากนั้นซีพียูจะทำการกระโดดไปยังแอดเดรสในหน่วยความจำที่กำหนดไว้เรียกตำแหน่งแอดเดรสนี้ว่า แอดเดรสอินเตอร์รัปต์เวกเตอร์ (interrupt vector address) ดังนั้น จะต้องมีการเขียนโปรแกรมย่อยการบริการอินเตอร์รัปต์ไว้ที่แอดเดรสอินเตอร์รัปต์เวกเตอร์นี้ โดยค่าของแอดเดรสอินเตอร์รัปต์เวกเตอร์จะแตกต่างกันไปในการอินเตอร์รัปต์แบบต่างๆ ดังมีรายละเอียดต่อไปนี้

การอินเตอร์รัปต์ภายนอกที่ขา INT0 มีค่าแอดเดรสอินเตอร์รัปต์เวกเตอร์อยู่ที่ 0003H

การอินเตอร์รัปต์จากไทเมอร์ 0 มีค่าแอดเดรสอินเตอร์รัปต์เวกเตอร์อยู่ที่ 000BH

การอินเตอร์รัปต์ภายนอกที่ขา INT1 มีค่าแอดเดรสอินเตอร์รัปต์เวกเตอร์อยู่ที่ 0013H

การอินเตอร์รัปต์จากไทเมอร์ 1 มีค่าแอดเดรสอินเตอร์รัปต์เวกเตอร์อยู่ที่ 001BH

การอินเตอร์รัปต์จากพอร์ตอนุกรม มีค่าแอดเดรสอินเตอร์รัปต์เวกเตอร์อยู่ที่ 0023H

การอินเตอร์รัปต์จากไทเมอร์ 2 มีค่าแอดเดรสอินเตอร์รัปต์เวกเตอร์อยู่ที่ 002BH

2.1.7.2 การเขียนโปรแกรมย่อยบริการอินเตอร์รัปต์

มีหลักการโดยทั่วๆ ไปดังนี้

1. ต้องเริ่มต้นด้วยแอดเดรสอินเตอร์รัปต์เวกเตอร์เสมอ เพื่อให้การตรวจสอบการทำงานทำได้ง่าย และแยกส่วนของโปรแกรมย่อยนี้ออกจากโปรแกรมหลักหรือโปรแกรมย่อยอื่นๆ อย่างชัดเจน ด้วยคำสั่ง ORG xxxxH (ค่าแอดเดรสอินเตอร์รัปต์เวกเตอร์)

2. เมื่อเข้าสู่โปรแกรมย่อย ควรเก็บค่าของรีจิสเตอร์หรือแฟลกที่ใช้แสดงสถานะต่างๆ ซึ่งต้องมีการใช้งานในโปรแกรมย่อยบริการอินเตอร์รัปต์นี้ไว้ในสเต็กเสียก่อน เพื่อป้องกันความผิดพลาดที่อาจเกิดขึ้นต่อการทำงานของทั้งโปรแกรมบริการอินเตอร์ และโปรแกรมหลัก ด้วยคำสั่ง PUSH

3. เมื่อเขียนโปรแกรมบริการอินเตอร์รัปต์เรียบร้อยแล้ว ให้ทำการคืนค่าของรีจิสเตอร์ที่นำมาใช้งานโปรแกรมบริการอินเตอร์รัปต์ด้วยคำสั่ง POP ยกเว้นรีจิสเตอร์ที่ต้องการนำผลการกระทำในโปรแกรมบริการอินเตอร์นี้ไปใช้งาน ซึ่งในทางปฏิบัติจริง ไม่พบมากนัก และไม่แนะนำให้เขียนโปรแกรมในลักษณะนี้

4. ปิดท้ายโปรแกรมย่อยบริการอินเตอร์รัปต์ด้วยคำสั่ง RETI เสมอ

ตัวอย่างโปรแกรมบริการอินเตอร์รัปต์อันเนื่องจากสัญญาณนอกที่ขา INT0 มีดังนี้

```
ORG  003H
PUSH  A
PUSH  P
PUSH  PSW
*
*       ; ส่วนของโปรแกรมย่อยบริการอินเตอร์รัปต์
*
POP  PSW
POP  P1
POP  A
RETI
```

2.1.8 รีจิสเตอร์ที่เกี่ยวข้องกับการอินเตอร์รัปต์ในไมโครคอนโทรลเลอร์ MCS-51

การอินเตอร์รัปต์ในไมโครคอนโทรลเลอร์ MCS-51 แบบแฟลช มีรีจิสเตอร์ที่เกี่ยวข้องอยู่ 2 ตัว ดังมีรายละเอียดต่อไปนี้

รีจิสเตอร์เอ็นเอเบิลการอินเตอร์รัปต์หรือ IE (Interrupt Enable register)

มีแอดเดรสอยู่ที่ A8H ในพื้นที่ของรีจิสเตอร์ฟังก์ชั่นพิเศษหรือ SFR มีขนาด 8 บิต สามารถเข้าถึงได้ในระดับบิต ใช้ในการเอ็นเอเบิลการตอบสนองการอินเตอร์รัปต์ในแบบต่างๆ มีรายละเอียดการทำงานดังนี้

บิต 7	บิต 6	บิต 5	บิต 4	บิต 3	บิต 2	บิต 1	บิต 0
EA	-	ET2	ES	ET1	EX1	ET0	EX0

ภาพที่ 2.17 แสดงบิตต่างๆ ของรีจิสเตอร์ IE

EA (Global enable/disable interrupt): ใช้เอ็นเอเบิลและดิสเอเบิลการตอบสนองการอินเตอร์-รัปต์ทั้งหมด

"0" ดิสเอเบิลการอินเตอร์รัปต์ นั่นคือ กำหนดให้ไมโครคอนโทรลเลอร์ไม่ตอบสนองการอินเตอร์-รัปต์

"1" เอ็นเอเบิลการอินเตอร์รัปต์ นั่นคือ กำหนดให้ไมโครคอนโทรลเลอร์สามารถตอบสนองการอินเตอร์รัปต์จากแหล่งกำเนิดต่างๆ

นั่นคือ ถ้าต้องการให้ไมโครคอนโทรลเลอร์ตอบสนองการอินเตอร์รัปต์ไม่ว่าจะแหล่งกำเนิดใด จะต้องเซตบิตนี้ก่อนเสมอ

ET2 (Timer 2 interrupt enable): ใช้ในการเอ็นเอเบิลการอินเตอร์รัปต์อันเนื่องมาจากการ โอเวอร์โฟลว หรือการแคปเจอร์ในไทเมอร์/เคาน์เตอร์ 2 จะมีเฉพาะในเบอร์ AT89C52 และในอนุกรม AT89Sxx เท่านั้น บิตนี้สามารถเซตและเคลียร์ด้วยกระบวนการทางซอฟต์แวร์

ES (Serial port interrupt enable bit): ใช้ในการเอ็นเอเบิลการอินเตอร์รัปต์อันเนื่องมาจากการรับหรือส่งข้อมูลทางพอร์ตอนุกรมภายในไมโครคอนโทรลเลอร์ MCS-51 สามารถเซตและเคลียร์ด้วยกระบวนการทางซอฟต์แวร์

ET1 (Timer 1 interrupt enable): ใช้ในการเอ็นเอเบิลการอินเตอร์รัปต์อันเนื่องมาจากการโอเวอร์โฟลวในไทเมอร์/เคาน์เตอร์ 1 บิตนี้สามารถเซตและเคลียร์ด้วยกระบวนการทางซอฟต์แวร์

EX1 (External interrupt 1 enable): ใช้ในการเอ็นเอเบิลการอินเตอร์รัปต์อันเนื่องมาจากสัญญาณภายนอกที่ป้อนเข้ามายังขา INT1 บิตนี้สามารถเซตและเคลียร์ด้วยกระบวนการทางซอฟต์แวร์

ET0 (Timer 0 interrupt enable): ใช้ในการเอ็นเอเบิลการอินเตอร์รัปต์อันเนื่องมาจากการ โอเวอร์โฟลวในไทเมอร์/เคาน์เตอร์ 0 บิตนี้สามารถเซตและเคลียร์ด้วยกระบวนการทางซอฟต์แวร์

EX0 (External interrupt 1 enable bit): ใช้ในการเอ็นเอเบิลการอินเตอร์รัปต์อันเนื่องมาจากสัญญาณภายนอกที่ป้อนเข้ามายัง ขา INT0 บิตนี้สามารถเซต และเคลียร์ด้วยกระบวนการทางซอฟต์แวร์

สำหรับบิต 6 ของรีจิสเตอร์ IE ไม่มีการใช้งานต้องกำหนดให้เป็น "0" เสมอ

รีจิสเตอร์จัดลำดับความสำคัญการตอบสนองการอินเตอร์รัปต์หรือ IP (Interrupt Priority register)

มีแอดเดรสอยู่ที่ 0B8H ในพื้นที่ของรีจิสเตอร์ฟังก์ชั่นพิเศษ หรือ SFR มีขนาด 8 บิต สามารถเข้าถึงได้ในระบิต ใช้ในการเลือกลำดับความสำคัญของการตอบสนองการอินเตอร์รัปต์ว่าต้องการให้ตอบสนองสัญญาณอินเตอร์รัปต์จากแหล่งกำเนิดใดเป็นลำดับก่อนหลัง ถ้าต้องการให้การอินเตอร์รัปต์จากแหล่งกำเนิดใดมีความสำคัญสูงสุด ให้กำหนดที่บิตนั้นเป็น "1"มีรายละเอียดของรีจิสเตอร์ IP ดังนี้

บิต 7	บิต 6	บิต 5	บิต 4	บิต 3	บิต 2	บิต 1	บิต 0
-	-	PT2	PS	PT1	PX1	PT0	PX0

ภาพที่ 2.18 แสดงบิตต่างๆ ของรีจิสเตอร์ IP

PT2 (Timer 2 interrupt priority bit): ใช้ในการกำหนดความสำคัญของการอินเตอร์รัปต์อันเนื่องมาจากการโอเวอร์โฟลวหรือการแคปเจอร์ในไทเมอร์/ เคาน์เตอร์ 2 จะมีเฉพาะในเบอร์ AT89C52 และในอนุกรม AT89Sxx เท่านั้น บิตนี้สามารถเซตและเคลียร์ด้วยกระบวนการทางซอฟแวร์

PS (Serial port interrupt priority bit): ใช้ในการกำหนดความสำคัญของการอินเตอร์รัปต์อันเนื่องมาจากการรับหรือส่งข้อมูลทางพอร์ตอนุกรมภายในไมโครคอนโทรลเลอร์ MCS-51 สามารถเซตและเคลียร์ด้วยการะบวนการทางซอฟแวร์

PT1 (Timer 1 interrupt priority bit): ใช้ในการกำหนดความสำคัญของการอินเตอร์รัปต์อันเนื่องมาจากการโอเวอร์โฟลวในไทเมอร์/ เคาน์เตอร์ 1 บิตนี้สามารถเซตและเคลียร์ด้วยกระบวนการทางซอฟแวร์

PX1 (External interrupt 1 priority bit): ใช้ในการกำหนดความสำคัญของการอินเตอร์รัปต์อันเนื่องมาจากสัญญาณภายนอกที่ป้อนเข้ามายังขา INT1 บิตนี้สามารถเซตและเคลียร์ด้วยกระบวนการทางซอฟแวร์

PT0 (Timer 0 interrupt priority bit): ใช้ในการกำหนดความสำคัญของการอินเตอร์รัปต์อันเนื่องมาจากสัญญาณภายนอกที่ป้อนเข้ามายังขา INT0 บิตนี้สามารถเซตและเคลียร์ด้วยกระบวนการทางซอฟแวร์

สำหรับบิต 6 และ 7 ของรีจิสเตอร์ IP ไม่มีการใช้งาน ต้องกำหนดให้เป็น 0 เสมอ

2.1.9 การทำงานครั้งละหนึ่งคำสั่ง (Single Step)

โครงสร้างการอินเตอร์รัปต์ของ 8051 ช่วยให้สามารถทดสอบโปรแกรมสั้นๆ ได้ด้วยการทำงานทีละคำสั่งดังที่กล่าวไว้แล้ว การร้องขออินเตอร์รัปต์จะไม่ตอบสนองในขณะที่กำลังทำงานอินเตอร์รัปต์ในระดับในระดับความสำคัญเดียวกัน มันจะไม่ตอบสนองหลังคำสั่ง RETI จนกว่าอย่างน้อยที่สุดคำสั่งได้ทำงานไปแล้วอีกหนึ่งคำสั่งจากโปรแกรม ดังนั้นในการอินเตอร์รัปต์ครั้งหนึ่งๆ ที่เข้ามามันจะไม่สามารถรับเข้ามาใหม่ได้ จนกว่าอย่างน้อยที่สุดได้ทำงานไปแล้วหนึ่งคำสั่งของโปรแกรมหลัก ทางหนึ่งที่ใช้ประโยชน์ของลักษณะงานเช่นนี้ คือ การทำงานแบบครั้งละหนึ่งขั้นตอนด้วยการโปรแกรมการอินเตอร์รัปต์ ภายนอกในแต่ละครั้ง ให้แอกทีฟด้วยระดับต่ำ และโปรแกรมบริการอินเตอร์รัปต์จะให้สิ้นสุดด้วยรหัสต่อไปนี้ คือ

```
JNB P3.2, $     ; ให้รอที่นี่จนกว่า INT0 จะเข้าสู่ระดับสูง
JB P3.2, $      ; ให้รอที่นี่จนกว่า INT0 จะเข้าสู่ระดับต่ำ
RETI            ; กลับไปที่เก่าและทำงานหนึ่งคำสั่ง
```

ดังนั้นถ้าขา INT0 ที่เป็นขา P3.2 มีขาเป็นระดับต่ำ ซีพียูจะไปยังโปรแกรมการบริการอินเตอร์รัพต์ภายนอก 0 และจะยังคงอยู่ที่โปรแกรมการบริการอินเตอร์รัพต์ จนกว่า INT0 จะมีพัลส์ (คือต่ำ -> สูง -> ต่ำ) ดังนั้นมันจะทำงานคำสั่ง RETI และกลับไปยังโปรแกรมที่ทดสอบอยู่ และทำงานหนึ่งคำสั่งและกลับเข้าสู่โปรแกรมการบริการอินเตอร์รัพต์ใหม่และรอจนกว่าจะมีพัลส์ลูกใหม่ที่ขา P3.2 การทำงานหนึ่งคำสั่งของโปรแกรมทดสอบจะควบคุมด้วยพัลส์ที่ขา P3.2 ในแต่ละลูก

2.2 การออกแบบวงจรดิจิตอลด้วย FPGA

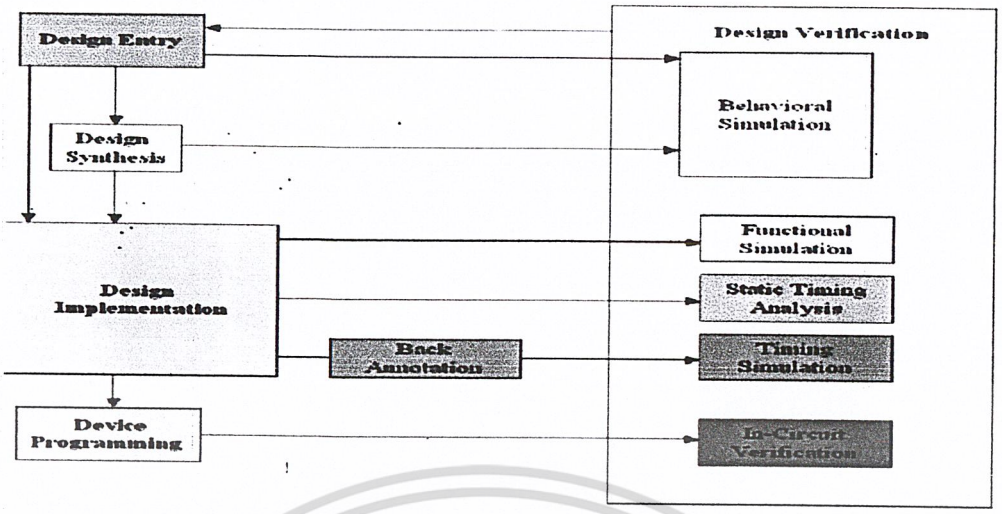
ในกระบวนการออกแบบวงจรโดยทั่วไปมักจะมีขั้นตอนหรือการทำงานเป็นขั้นตอนดังนี้

1. System Requirement
2. System Design
3. System Implementation
4. Testing and Debuging
5. Documentation

กระบวนการทำ System Requirement คือ การหาความต้องการว่าเราต้องการสร้างวงจรอะไร วงจรทำงานอย่างไร และมีเอาท์พุทอย่างไรบ้าง กระบวนการทำ System Design คือการออกแบบและหาวิธีแก้ไขปัญหาจากโจทย์ที่ได้มาในขั้นตอน System Requirement จากนั้นจึงเป็นการทำ System Implementation เพื่อสร้างวงจรตามที่ได้ออกแบบไว้ เมื่อได้วงจรที่ต้องการแล้วจึงทำ Testing and Debuging เพื่อทำการทดสอบ และแก้ไขวงจรให้ทำงานถูกต้องในกรณีที่มีปัญหา

ขั้นตอนสุดท้ายเป็นการทำ Documentation เพื่อสร้างเอกสารอธิบายการทำงานของวงจรที่ออกแบบทั้งหมด

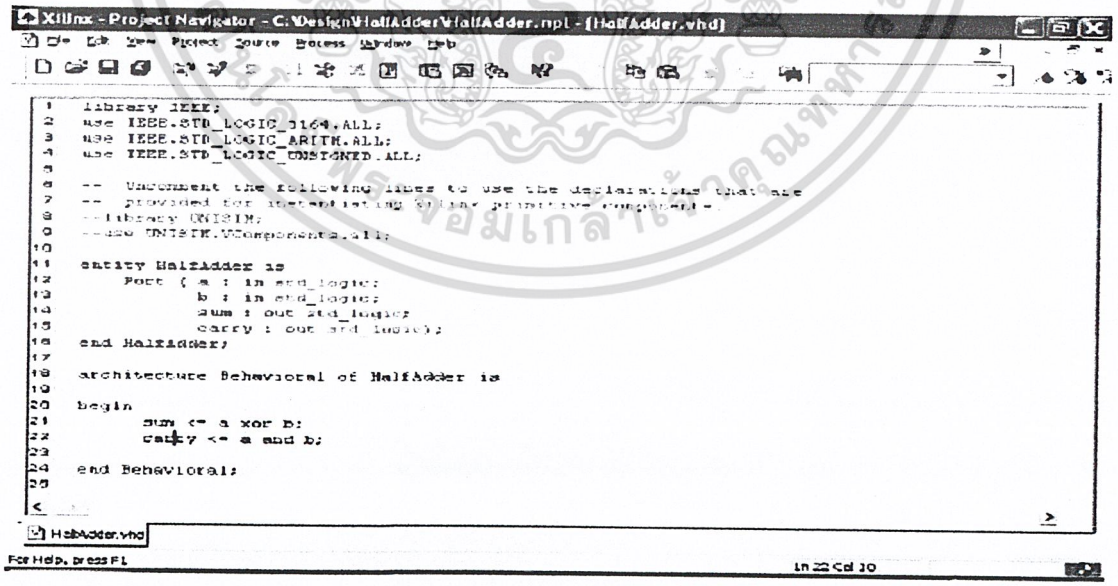
1. ในกระบวนการออกแบบวงจรดิจิตอลด้วย FPGA เริ่มจากขั้นตอนการทำ System Implementation จนถึง Testing and Debuging โดยทั่วๆ ไปแล้วมีขั้นตอนการทำงานย่อยๆ หลายขั้นตอน ดังภาพที่ 2.19 มีรายละเอียดดังนี้



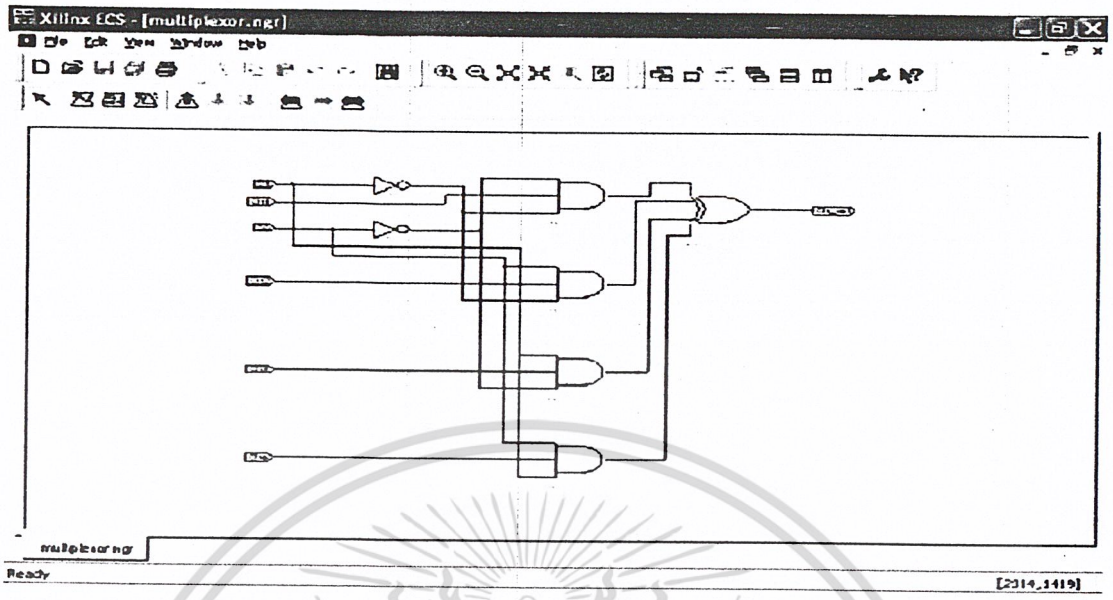
ภาพที่ 2.19 แสดงขั้นตอนการออกแบบวงจรดิจิตอลด้วย FPGA

2.2.1 การออกแบบวงจร (Design Entry)

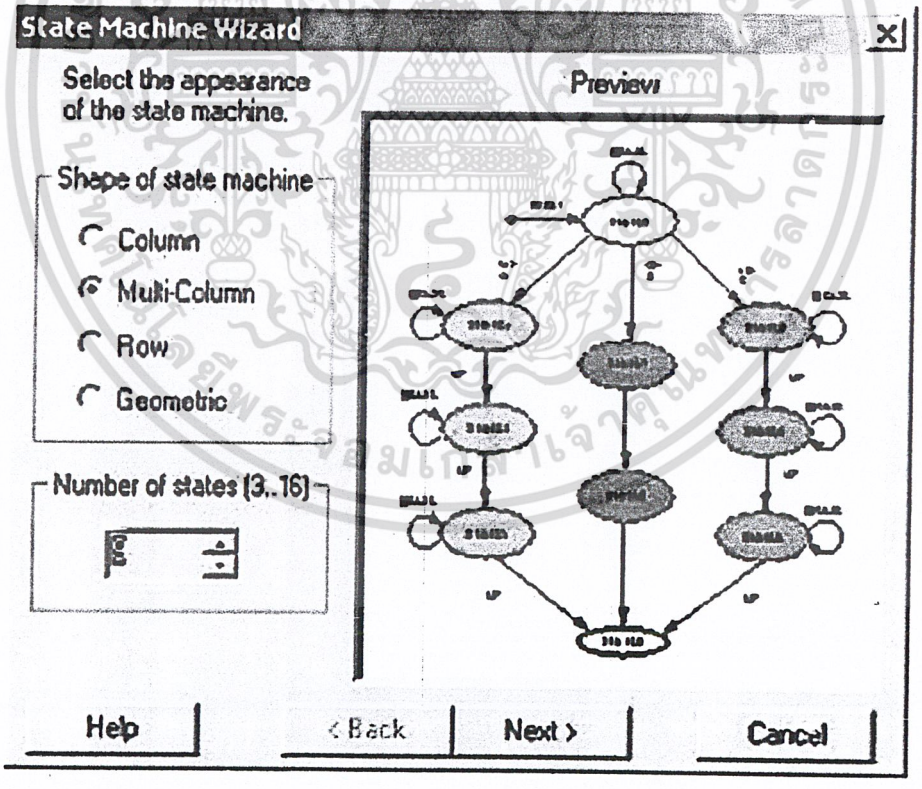
เป็นการนำเอาคุณสมบัติของวงจรดิจิตอลที่ต้องการสร้างมาออกแบบด้วยวิธีการต่างๆ เช่น การเขียนด้วยภาษา HDL (Hardware Description Language) ซึ่งอาจจะเป็นภาษา VHDL (Verilog) ดังภาพที่ 2.20 หรือสร้างวงจรด้วยวิธีการวาดผังวงจร (Schematic) ดังภาพที่ 2.21 หรือเขียนเป็นแผนภูมิสถานะ (State Diagram) ดังภาพที่ 2.22 เพื่อให้ได้วงจรตามต้องการ



ภาพที่ 2.20 แสดงการออกแบบวงจรโดยการเขียนด้วยภาษา VHDL



ภาพที่ 2.21 แสดงการออกแบบวงจรโดยการวาดผังวงจร (Schematic)



ภาพที่ 2.22 แสดงการออกแบบวงจรโดยการเขียน State Diagram

2.2.2 การตรวจสอบวงจรที่ออกแบบ (Design Varification)

เป็นการนำโค้ด (Code) VHDL (หรือ Varilog) ของวงจรที่ออกแบบ หรือวงจรที่ออกแบบโดยวิธีวาดผังวงจรไปทำการตรวจสอบความถูกต้องด้วยโปรแกรมจำลองการทำงาน โดยทั่วไปจะมีการจำลองการทำงานใน 3 ระดับคือ

2.2.2.1 Behavioral simulation

เป็นการจำลองเฉพาะพฤติกรรมของวงจร โดยยังไม่คิดถึงโครงสร้างของวงจรภายในเพื่อให้ได้แบบจำลองการทำงานเบื้องต้น

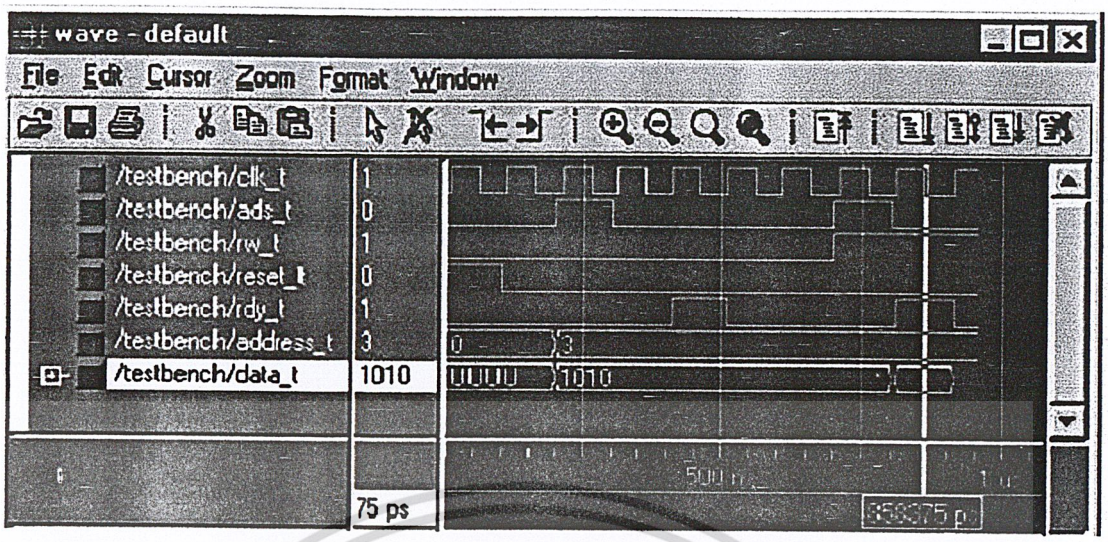
2.2.2.2 Functional simulation

เป็นการนำโค้ดในระดับ RTL (Register-transfer-level) มาจำลองการทำงานเพื่อตรวจสอบการทำงานวงจรก่อนจะนำไปสังเคราะห์วงจร (Systhesis)

2.2.2.3 Timing Simulation

เป็นการจำลองการทำงานที่ใกล้เคียงกับ Hardware จริงมากที่สุดเนื่องจากเป็นการนำข้อมูล Timing ที่เกิดขึ้นใน FPGA มาใช้ ขั้นตอนนี้มักจะใช้ก่อนทำการทดสอบกับ FPGA จริง หรือใช้หาข้อผิดผลาดในกรณีที่การทำงานจริงบน FPGA ไม่ถูกต้องเหมือนผลที่ได้จาก Functional simulation โดยทำการสร้างสัญญาณอินพุทจำลอง เพื่อป้อนให้กับวงจรที่ออกแบบแล้วทำการตรวจสอบสัญญาณเอาท์พุทว่าถูกต้อง หรือไม่โดยใช้ซอฟท์แวร์จำลองการทำงาน เช่น ModelSim ดังภาพที่ 2.23

อย่างไรก็ตามในการสร้างวงจรดิจิตอลต่างๆ โดยทั่วๆ ไปผู้ออกแบบมักจะจำลองการทำงานของวงจรก่อนเพื่อให้แน่ใจว่าวงจรที่ออกแบบสามารถทำงานได้อย่างถูกต้อง แต่ถ้าเป็นวงจรง่ายๆ ผู้ออกแบบอาจไม่มีความจำเป็นที่จะต้องจำลองการทำงานของวงจรแต่อย่างใด อาจข้ามไปทำการสังเคราะห์วงจร (Synthesis) เลยก็ได้



ภาพที่ 2.23 แสดงการจำลองการทำงานของวงจร

2.2.3 การสังเคราะห์วงจรที่ออกแบบ (Design Synthesis)

การสังเคราะห์วงจร คือ การแปลงโค้ดให้เป็นวงจรในระดับเกต (Gate-level) ซึ่งจะเป็น Netlist ของเกตต่างๆ ในวงจร การสังเคราะห์วงจรนั้นจะต้องมีการระบุว่าเป็น FPGA หรือ CPLD ชื่อผู้ผลิตและหมายเลขรุ่นที่ใช้ด้วยการป้อนค่าพารามิเตอร์ต่างๆ ให้กับโปรแกรม Synthesis Tools บางตัวจะสามารถให้ผู้ใช้เลือกวิธีการสังเคราะห์วงจรได้ว่าต้องการ optimize แบบใดระหว่างความเร็ว และพื้นที่

2.2.4 Design Implementation

ขั้นตอนนี้ประกอบด้วย การแบ่งวงจร การวางอุปกรณ์และการเชื่อมต่อสัญญาณ (Partitioning, Placement and Routing) ตามลำดับโดยจะใช้ Software Tool การแบ่งวงจรเป็นการนำวงจรที่ได้จากการสังเคราะห์แล้วมาแบ่งเป็นวงจรย่อยๆ เพื่อนำไปวางลงในโครงสร้างของอุปกรณ์ชนิดต่างๆ ที่อยู่ภายในชิพ จากนั้นทำการวิเคราะห์ว่าควรวางวงจรย่อยๆ นี้ที่ตำแหน่งใดบ้างจึงจะเหมาะสมที่สุด เมื่อเสร็จแล้วให้ทำการเชื่อมต่อสัญญาณต่างๆ ภายในชิพเข้าด้วยกัน ข้อมูลที่ใช้ในขั้นตอนนี้อาจได้มาจาก Design Entry โดยตรงในกรณีที่เป็น Schematic หรือจาก Design Synthesis ในกรณีที่เป็น HDL

2.2.5 การโปรแกรมข้อมูลวงจรลงชิพ (Device Programming)

การโปรแกรมข้อมูลนั้นสามารถทำได้โดยการนำไฟล์บิทสตรีม เช่นไฟล์ที่มีนามสกุล .bit (ไฟล์ข้อมูลวงจรของ FPGA) ที่ได้จากการ Implementation เพื่อดาวน์โหลดลงชิพ โดยใช้เครื่องโปรแกรมซึ่งทำมาโดยเฉพาะ หรืออาจใช้ดาวน์โหลดเคเบิลก็ได้

2.3 VHDL

ความซับซ้อนและขนาดของระบบดิจิตอลในปัจจุบันได้เพิ่มมากขึ้นทุกขณะ ส่งผลให้มีการนำคอมพิวเตอร์เพื่อช่วยในการออกแบบหรือ CAD มาใช้ในขบวนการออกแบบฮาร์ดแวร์เพิ่มขึ้นเช่นกัน อีกทั้งอุปกรณ์และวิธีการออกแบบใหม่ๆ ก็ถูกพัฒนาขึ้นมาเพื่อช่วยอำนวยความสะดวกให้กับนักออกแบบมากขึ้นด้วย สำหรับภาษาบรรยายอุปกรณ์ฮาร์ดแวร์ (HDL: Hardware Description Language) ก็เป็นเครื่องมืออย่างหนึ่งที่ได้รับการพัฒนามาอย่างต่อเนื่อง เพื่อช่วยให้การปรับปรุงขบวนการออกแบบระบบดิจิตอลเป็นไปอย่างมีประสิทธิภาพ

2.3.1 องค์ประกอบพื้นฐานของ VHDL

ภาพแบบพื้นฐานที่ใช้ในการบรรยายถึงองค์ประกอบของ VHDL จะประกอบไปด้วยส่วนกำหนดการเชื่อมต่อ (Interface) และส่วนกำหนดลักษณะเชิงสถาปัตยกรรม (Architecture) โดยในการบรรยายการเชื่อมต่อจะขึ้นต้นด้วยคำว่า ENTITY แล้วตามด้วยชื่อขององค์ประกอบจากนั้นตามด้วยคำว่า IS และถัดมาจะเป็นการบรรยายถึงพอร์ตการติดต่ออินพุท - เอาท์พุท ขององค์ประกอบ ส่วนลักษณะภายนอกอื่นๆ เช่น เวลา อุณหภูมิก็สามารถรวมเข้าไปในส่วนนี้ได้เช่นกัน ในส่วนของการกำหนดลักษณะเชิงสถาปัตยกรรมจะขึ้นต้นด้วยคำว่า ARCHITECTURE ซึ่งเป็นส่วนที่ใช้บรรยายหน้าที่การทำงานขององค์ประกอบ โดยหน้าที่การทำงานนี้จะขึ้นอยู่กับสัญญาณอินพุท เอาท์พุท และพารามิเตอร์อื่นๆ ที่ได้กำหนดไว้ในส่วนของการเชื่อมต่อดังภาพที่ 2.24 และสำหรับการบรรยายหน้าที่ขององค์ประกอบจะเริ่มต้นหลังจากคำว่า BEGIN เป็นต้นไป

```
ENTITY component_name IS
    Input and output ports
    Physical and other parameters
END [component_name] ;

ARCHITECTURE identifier OF component_name IS
    [declartion]
BEGIN
    specification of the functionality of the component
    in terms of its input lines and as influenced
    by physical and other parameters
END [identifier];
```

ภาพที่ 2.24 แสดงการกำหนดการเชื่อมต่อและสถาปัตยกรรม

2.3.2 ภาพแบบการเขียน Architecture ของ VHDL

VHDL มีภาพแบบการเขียนอยู่ 3 ภาพแบบ

1 Structural Model การอธิบายวงจรโดยใช้โมดูลต่างๆ มาเชื่อมต่อกัน ให้เห็นโครงสร้างภายใน

2 Behavioral Model การอธิบายการทำงานของวงจรในระดับลอจิกเกท

3 Sequential Model การอธิบายขั้นตอนการทำงานโดยใช้ Sequential Statement ทำงานทีละขั้นตอน

ต่อไปนี้จะเป็นตัวอย่างการเขียนโปรแกรมเพื่ออธิบายวงจร RS F/F โดยใช้ Architecture ที่แตกต่างกัน

Structural Model

```
u1: nand2 port map (set, q, qb);
u2: nand2 port map (reset, qb, q);
```

Behavioral Model

```
qb <= not (q and set) after 2 ns;
q  <= not (qb and reset) after 2 ns;
```

Sequential Model

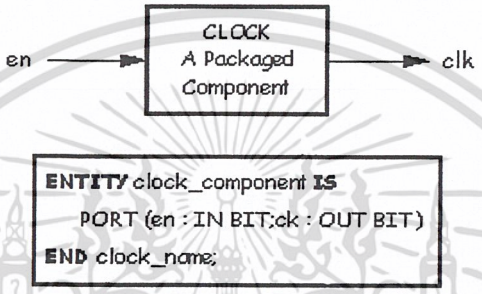
```
process (set, reset)
Begin
   If set = '0' and reset = '1' then
      q <= '0' after 2 ns; qb <= '1' after 4 ns;
   If ...
   ...
   End process;
```

สำหรับภาพแบบการเขียนที่เราจะเลือกใช้นั้น ขึ้นอยู่กับความเหมาะสมของฮาร์ดแวร์ เราสามารถที่จะใช้ภาพแบบต่างๆ มาร่วมกันได้อยู่ใน Architecture ตัวเดียวกันก็ขึ้นอยู่กับความเหมาะสมเช่นกัน จะเห็นว่าในแต่ละแบบก็มีจุดเด่นของตัวมันเองแต่ทั้ง 3 แบบก็ให้ผลลัพธ์เหมือนกัน

Sequential มีการอธิบาย Architecture เป็นลำดับขั้นตอนที่ชัดเจน เข้าใจง่าย โดยปกติแล้วโปรแกรมที่อยู่ภายใน Architecture นั้นจะมีการ process พร้อมๆกันทุกบรรทัด (เหมือนวงจรจริงที่มีการเชื่อมต่อสัญญาณกันทั้งวงจร เมื่อมีสัญญาณ input ตัวใดเปลี่ยน output ก็จะมีการเปลี่ยนตามทันที จะมี delay ก็เพียงเล็กน้อย) แต่แบบ Sequential สามารถจะมองเป็นขั้นตอนได้โดยใส่ Process Statement คร่อมเข้าไปในโปรแกรม ส่วน Structural ก็มีการดึง Component NAND มาจะมองเห็นการเชื่อมต่อที่ชัดเจน ส่วน Behavioral นั้นมีการอธิบายคุณสมบัติระดับลอจิกเกท

2.3.3 การกำหนดการเชื่อมต่อ

การกำหนดการเชื่อมต่อเป็นระดับบนสุดของการออกแบบ โดยในระดับนี้ต้องกำหนดพอร์ตสำหรับการติดต่อกับองค์ประกอบภายนอกอื่นๆ ดังตัวอย่างในภาพที่ 2.25 ซึ่งเป็นบล็อคไดอะแกรม และการบรรยายการเชื่อมต่อขององค์ประกอบสำหรับตัวจ่ายสัญญาณนาฬิกา ในบรรทัดแรกของการบรรยายการเชื่อมต่อ เป็นการกำหนดชื่อขององค์ประกอบ ซึ่งกำหนดเป็น clock_ component ตามด้วยคำว่า PORT และชื่อของพอร์ตอยู่ภายในวงเล็บส่วน IN และ OUT เป็นการกำหนดโหนดของสัญญาณให้เป็นอินพุทหรือเอาท์พุท และ BIT เป็นการแสดงชนิดของข้อมูล



ภาพที่ 2.25 แสดงบล็อคไดอะแกรมและการบรรยายการเชื่อมต่อของ clock_ component

2.3.4 การกำหนดภาพแบบการบรรยาย

หน้าที่การทำงานขององค์ประกอบจะถูกบรรยายภายในส่วนนี้ ซึ่งในการบรรยายสามารถกำหนดค่าของสัญญาณเอาท์พุทในเทอมของอินพุทหรือในภาพขององค์ประกอบอื่นๆ หรือทั้งสองอย่างรวมกันก็ได้ ดังตัวอย่างการบรรยายของ clock_component ในภาพที่ 2.26 ซึ่งเป็นการบรรยายในเชิงพฤติกรรมโดยมี en เป็นอินพุทและ ck เป็นเอาท์พุท PROCESS เป็นคำที่ใช้ในการเริ่มต้นสำหรับการบรรยายในเชิงพฤติกรรม และภายในโปรเซสกำหนดให้ periodic เป็นตัวแปรที่มีค่าเริ่มต้นเป็น "0" ถ้าสัญญาณ en มีค่าเป็น "1" จะทำให้ตัวแปร periodic ถูกคอมพลีเมนท์ (complement) และส่งค่าให้กับ ck ซึ่งเป็นสัญญาณเอาท์พุท และสำหรับคำสั่ง WAIT กำหนดให้สัญญาณมีคาบเวลาเท่ากับ 1 ไมโครวินาที

```
ARCHITECTURE behavioral OF clock_component IS

BEGIN
   PEOCESS
      VARIABLE periodic : BIT := '0';
   BEGIN
      IF en='1' THEN
       periodic :=  Not periodic;
      END IF;
       ck <= periodic;
       WAIT FOR 1 US;
  END PROCESS;
END behavioral;
```

ภาพที่ 2.26 แสดงการบรรยายเชิงพฤติกรรมของ clock _ component

2.3.5 หน่วยการออกแบบแพ็กเก็ต

ข้อมูลต่างๆ ตลอดจนโปรแกรมย่อยที่เป็นประโยชน์ต่อการเขียนภาพแบบการบรรยายระบบดิจิตอล สามารถเก็บไว้ในส่วนของแพ็กเก็ต ซึ่งหน่วยการออกแบบต่างๆ เช่น หน่วยการออกแบบ Entity หน่วยการออกแบบสถาปัตยกรรมหรือ หน่วยการออกแบบแพ็กเกจอื่นๆ สามารถเรียกข้อมูลเหล่านี้ไปใช้ได้ นอกจากนั้นสิ่งที่นิยมทำกันมากคือการนำภาพแบบมาตรฐานต่างๆ เช่น อุปกรณ์มาตรฐาน (เช่น ไอซีตระกูล 74XX เป็นต้น) มาเก็บไว้ในภาพของแพ็กเกจที่ทุกคนสามารถเข้าถึงได้ ตามปกติแล้วแพ็กเกจจะแบ่งออกเป็น 2 ส่วนคือ การประกาศแพ็กเกจ (Package declaration) และ ส่วนของบอดี้แพ็กเกจ (Package body) เนื่องจากแพ็กเกจถูกสร้างขึ้นเป็นส่วนแยกต่างหากออกจากภาพแบบที่ กำลังเขียนอยู่ ฉะนั้นการที่นำแพ็กเกจไปใช้นั้นจะต้องมีการเชื่อมโยงหรืออ้างอิงเสียก่อน ซึ่งในภาษา VHDL สามารถ กระทำได้ด้วยชุดคำสั่ง USE

2.3.5.1 PACKAGE DECLARATION

ส่วนที่มีความสำคัญที่สุดของแพ็กเกจ (ถ้ามองในแง่ของการนำไปใช้จากภายนอก) ได้แก่ส่วนการประกาศแพ็กเกจ เนื่องจากเป็นส่วนที่ใช้กำหนดชื่อของสิ่งที่ประกาศอยู่ภายในแพ็กเกจ สำหรับนำไปใช้ภายนอกตัวของแพ็กเกจเอง ถ้ามีการประกาศสิ่งใดๆ ในส่วนบอดี้แพ็กเกจแต่ไม่ถูกประกาศในส่วนการประกาศแพ็กเกจจะทำให้ค่า และพฤติกรรมไม่สามารถนำไปใช้งานในส่วนนอกได้ซึ่งเปรียบเทียบได้กับสิ่งที่ประกาศไว้ในส่วนของการประกาศ Entity คือ จุดเชื่อมต่อหรือพอร์ตที่มีหน้าที่ติดต่อกับโลกภายนอก ฉะนั้นโดยทั่วไปแล้วแพ็กเกจสามารถสร้างขึ้นได้โดยไม่จำเป็นต้องมีส่วนบอดี้ และยังสามารถนำไปใช้งานจากภาพแบบภายนอกได้เช่น ใช้สำหรับประกาศชนิด (Type) หรือสัญญาณ เช่นเดียวกับส่วนบอดี้แพ็กเกจที่ไม่จำเป็นต้องมีส่วนของการประกาศแพ็กเกจ แต่แพ็กเกจนั้นจะไม่สามารถนำไปใช้จากภาพแบบอื่นได้

```
PACKAGE package_name IS
    Package_declarative_part
END package_name;
```

ภาพที่ 2.27 แสดงโครงสร้างทั่วไปของส่วนการประกาศแพ็กเกจ

2.3.5.2 PACKAGE BODY

โครงสร้างซึ่งประกอบด้วยลำดับคำสั่งที่ใช้บรรยายฟังก์ชันการทำงาน ของโปรแกรมย่อยทั้งหลาย ซึ่งชื่อของโปรแกรมย่อยนั้นๆ ได้ถูกประกาศไปแล้วในส่วนของการประกาศแพ็กเกจ จะถูกเก็บไว้ในส่วนของบอดี้แพ็กเกจ ทั้งนี้รวมถึงการกำหนดค่าคงที่ต่างๆ อันได้แก่ค่าคงที่ที่ถูกประกาศชื่อไว้ก่อนในส่วนของการประกาศแพ็กเกจ และถูกกำหนดค่าในส่วนของบอดี้แพ็กเกจ ฉะนั้นในส่วนของบอดี้แพ็กเกจจึงไม่จำเป็นต้องมี ถ้าในส่วนของการประกาศแพ็กเกจไม่มีการ ประกาศชื่อที่เป็นโปรแกรมย่อย หรือค่าคงที่ การเขียนบอดี้แพ็กเกจนั้นจะเป็นไปตามกฎเกณฑ์ดังแสดงในภาพที่ 2.28

```
PACKAGE BODY package_name IS
    declarative part
END package_name;
```

ภาพที่ 2.28 แสดงโครงสร้างของบอดี้แพ็กเกจ

2.3.6 หน่วยการออกแบบ Configuration

ดังที่ทราบกันแล้วว่าระบบดิจิตอลภาพแบบหนึ่งไม่ว่าจะเป็นอะไรก็ตามจะสามารถมีหน่วยการออกแบบ Entity ได้เพียงหนึ่งเดียวเท่านั้น ซึ่งในหน่วยการออกแบบ Entity หนึ่งหน่วยนี้อาจจะมีสถาปัตยกรรมที่เป็นหน่วยรองได้หลายหน่วย ดังนั้นจะต้องมีหน่วยการออกแบบ Configuration มาเพื่อกำหนดการใช้ Configuration ของการประกอบ Entity กับหน่วยการออกแบบสถาปัตยกรรมหน่วยใดๆ เข้าด้วยกัน

```
COMFIGURTRION identifier OF entiry_name IS
    Configuretion_declarative_part
END ;
```

ภาพที่ 2.29 แสดงโครงสร้างโดยทั่วไปของหน่วยการออกแบบโครงแบบ

2.3.7 โปรแกรมย่อย

การใช้ฟังก์ชันและโพรซีเจอร์ใน VHDL เปรียบได้กับการใช้โปรแกรมย่อยในการเขียนโปรแกรมภาษาชั้นสูงทั่วๆ ไปค่าที่ถูกส่งกลับหรือถูกเปลี่ยนแปลงโดยโปรแกรมย่อยอาจจะมีหรือไม่มีผลต่อฮาร์ดแวร์โดยตรงก็ได้ เช่น ถ้าใช้ฟังก์ชันแทนการกระทำในสมการบูลีนก็จะมีผลต่อวงจรลอจิกจริงๆ ในขณะที่ถ้าใช้โปรแกรมย่อยในการเปลี่ยนชนิดของข้อมูล หรือในการคำนวณค่าการหน่วงเวลาแล้วก็จะไม่มีผลต่อโครงสร้างของฮาร์ดแวร์ ภาพที่ 2.30 แสดงการใช้โพรซีเจอร์เพื่อเปลี่ยนข้อมูลชนิด 8 บิตเป็นค่าจำนวนเต็ม และภาพที่ 2.31 แสดงการใช้ฟังก์ชันโดยกำหนดให้ X เป็นตัวแปรชนิด บิตแทนการกระทำในสมการบูลีน

```
TYPE byte IS ARRAY (7 DOWNTO 0) OF BIT;
 ...
PROCEDURE byte_to_integer (ib : IN byte; oi : OUT INTEGER) IS
        VARIABLE result: INTEGER := 0;
BEGIN
        FOR i IN 0 TO 7 LOOP
                IF ib(i) = '1' THEN
                result  := result + 2**i;
                END IF;
        END LOOP;
        oi := result;
END byte_to_integer
```

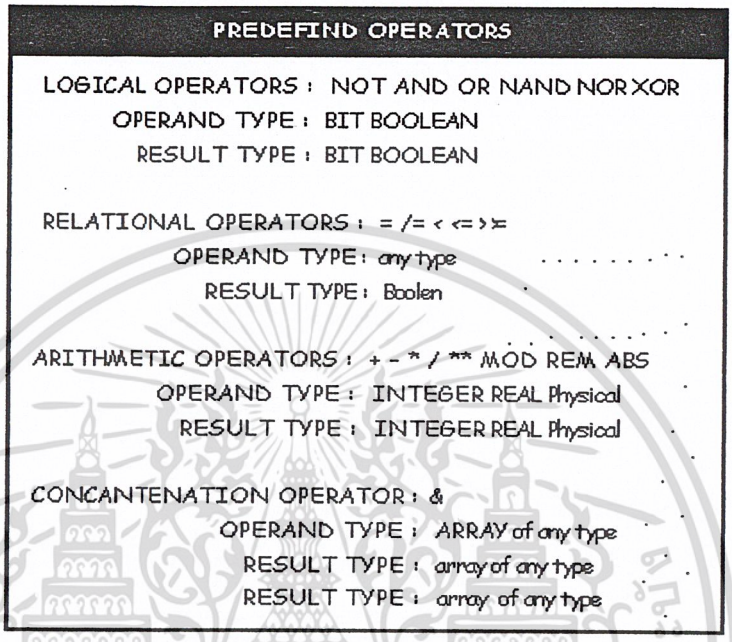
ภาพที่ 2.30 แสดงการใช้โพรซีเจอร์

```
FUNCTION f (a, b, c:BIT) RETURN BIT  IS
        VARIABLE x:BIT;
BEGIN
        x := ((NOT a) AND (NOT b) AND c);
        RETURN x;
END f;
```

ภาพที่ 2.31 แสดงการใช้ฟังก์ชัน

2.3.8 โอเปอร์เรเตอร์

การบรรยายเชิงพฤติกรรมในภาษา VHDL มีตัวดำเนินการหรือโอเปอร์เรเตอร์ทางลอจิกและคณิตศาสตร์เช่นเดียวกับภาษาซอฟต์แวร์ทั่วไปดังภาพที่ 2.32



ภาพที่ 2.32 แสดงตัวดำเนินการใน VHDL

2.3.9 เวลาและความพร้อมเพรียง

ในวงจรอิเล็กทรอนิกส์อุปกรณ์ทุกๆ ตัวจะอยู่ในสภาพเตรียมพร้อมเสมอ (Always Active) และจะมีเรื่องของเวลาเข้ามาเกี่ยวข้องในทุกๆเหตุการณ์ที่เกิดขึ้นเสมอ VHDL เป็นภาษาที่ได้รับการออกแบบมาเพื่อให้สามารถบรรยายภาพแบบ และการพ้องกันของเวลาสำหรับการทำงานของอุปกรณ์ได้อย่างถูกต้อง การบรรยายการทำงานที่อยู่ภายในส่วนของการบรรยายสถาปัตยกรรม จะมีการทำงานที่พร้อมเพรียงกันเสมอ หรือแม้แต่โปรเซสซึ่งมีการทำงานภายในเป็นแบบลำดับคำสั่งก็ตาม ซึ่งหากมีหลายโปรเซสอยู่ภายในโครงสร้างเดียวกัน ทุกๆ โปรเซสก็จะทำงานไปพร้อมๆ กัน

2.3.10 สัญญาณและตัวแปร

สัญญาณมีลักษณะเป็นเสมือนตัวกลางฮาร์ดแวร์ที่ใช้ในการส่งผ่านข้อมูล และมีเรื่องของเวลาเข้ามาเกี่ยวข้องด้วย การกำหนดค่าให้กับสัญญาณจะใช้สัญลักษณ์ <= ในการส่งค่า และสามารถใช้คำสั่ง AFTER เพื่อกำหนดช่วงเวลาในการส่งผ่านค่าของสัญญาณ เช่น w <= a AFTER 12 NS

หมายถึงการกำหนดค่าสัญญาณ a ให้กับ w หลังจากเวลา ผ่านไป 12 นาโนวินาที ในทางตรงข้ามตัวแปรมีลักษณะเป็นเสมือนตัวกลางที่ใช้ในการส่งผ่านข้อมูล และไม่มีเรื่องของเวลาเข้ามาเกี่ยวข้อง ซึ่งตัวแปรจะถูกใช้ในส่วนที่มีการทำงานเป็นแบบลำดับคำสั่งเช่นใน ฟังก์ชัน โพรซีเจอร์ และ โปรเซส สำหรับการกำหนดค่าให้กับตัวแปรจะใช้สัญลักษณ์ : =

2.3.11 ข้อดี ของ VHDL

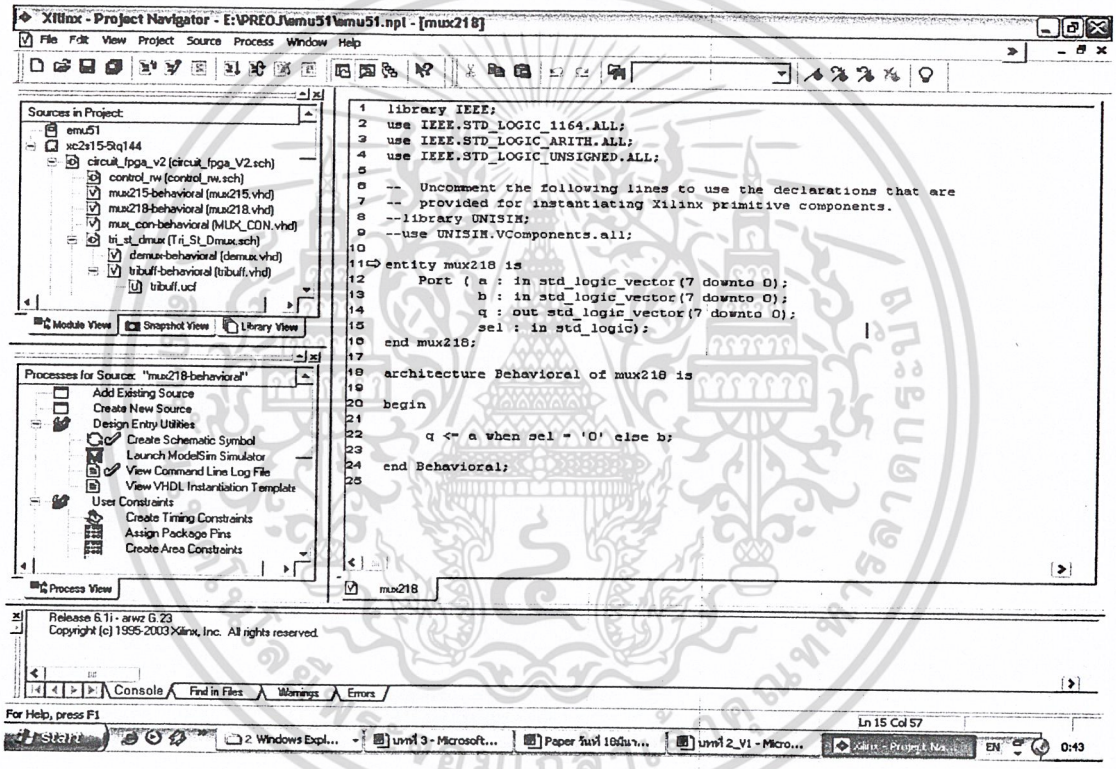
1. ภาพแบบของภาษาที่เข้าใจได้ง่าย
2. มีโครงสร้างของภาษาที่สามารถปรับเปลี่ยนให้เข้ากับ Hardware ได้ง่าย
3. มี Statement ให้ใช้งานอยู่หลายตัว
4. สามารถออกแบบในภาพแบบ Top-down design ได้
5. มีLibraryต่างๆให้เลือกใช้งาน ได้ทำให้ออกแบบได้ง่ายขึ้นเป็นประโยชน์ต่อ programmer
6. การออกแบบวงจรสามารถทำได้โดยง่าย
7. มือมีการแก้ไขหรือเปลี่ยนแปลงวงจรสามารถทำได้โดยง่าย เพียงโปรแกรมวงจรใหม่ลงไปไม่ต้องมีการเปลี่ยนแปลงตัว Hardware
8. การทดสอบไม่จำเป็นต้องทดลองกับวงจรจริงสามารถใช้การ Simulate ทดลองดูผลลัพธ์ของวงจรที่เราออกแบบได้ว่าถูกต้องหรือไม่

บทที่ 3
การออกแบบและวิธีการดำเนินงานวิจัย

3.1 การออกแบบวงจรดิจิตอลภายใน FPGA

3.1.1 การออกแบบวงจรดิจิตอลโดยใช้ VHDL

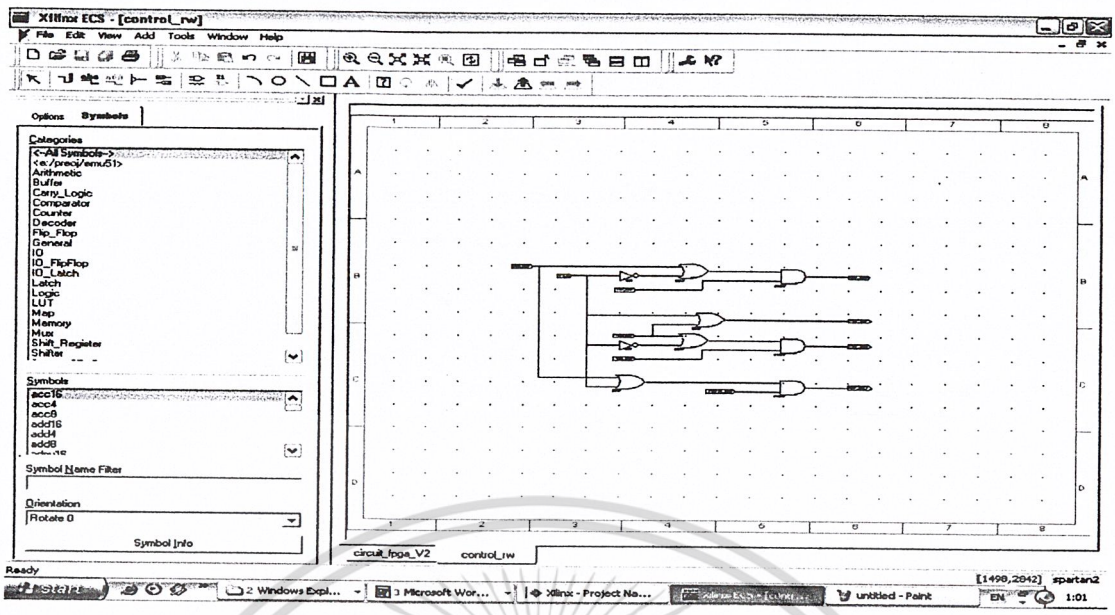
ในส่วนนี้จะเป็นการสร้างวงจรดิจิตอลที่ต้องการขึ้นมาโดยใช้ภาษา VHDL เช่น Latch, Multiplexers และ Tri-State Buffer เป็นต้น



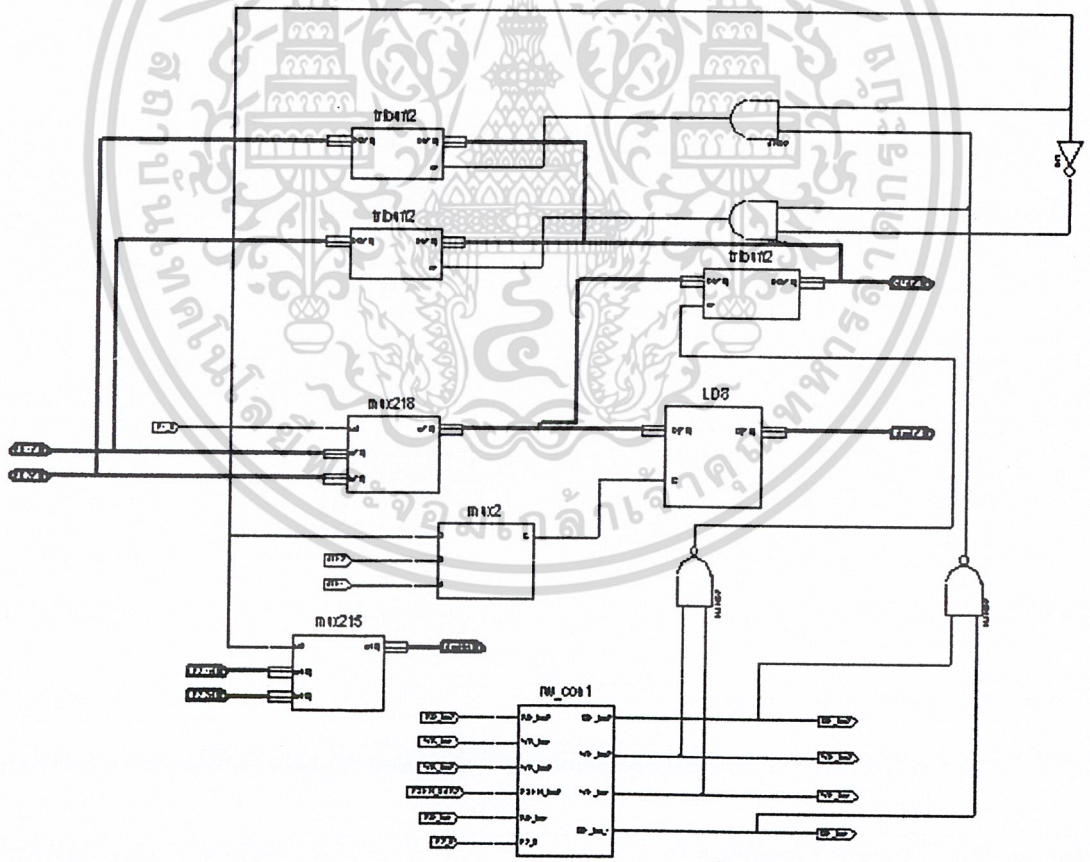
ภาพที่ 3.1 แสดงตัวอย่างวงจร Multiplexers โดยใช้ VHD

3.1.2 การออกแบบวงจรดิจิตอลโดยการวาดผังวงจร (Schematic)

ในส่วนนี้จะเป็นการนำวงจรดิจิตอลที่ได้จากการเขียน VHDL และจาก Library นำมาเชื่อมต่อกันเป็นวงจรดิจิตอลที่ต้องการ

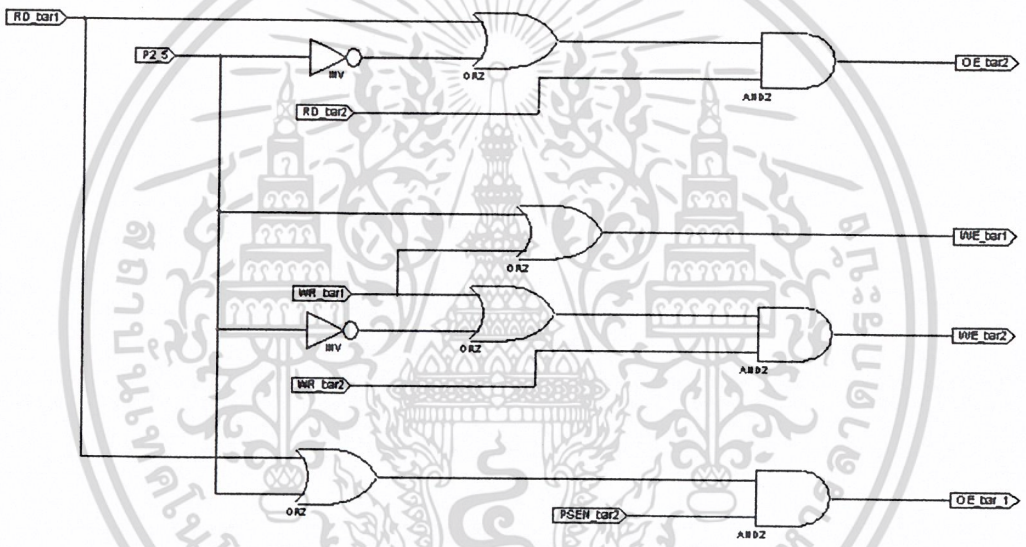


ภาพที่ 3.2 แสดงตัวอย่างวงจร โดยการวาดผัง (Schematic)



ภาพที่ 3.3 แสดงวงจรภายใน FPGA

จากภาพที่ 3.3 เป็นการนำเอาวงจร Multiplex และ Latch มาต่อร่วมกัน โดยวงจร Multiplex จะถูกควบคุมด้วย P1.0 ของไมโครคอนโทรลเลอร์ตัวที่ 1 ถ้า P1.0 เป็น '0' จะเลือกเอาสัญญาณ ALE และบัสข้อมูลของไมโครคอนโทรลเลอร์ตัวที่ 1 โดยสัญญาณ ALE ที่รับเข้ามาจะนำไปกระตุ้นวงจร Latch ให้เก็บค่า Address เอาไว้ใช้ในการติดต่อกับหน่วยความจำ ซึ่งจะมีส่วนที่ทำหน้าที่ในการเลือกว่าติดต่อกับหน่วยความจำภายนอกตัวที่ 1 หรือ ตัวที่ 2 ซึ่งส่วนควบคุมการติดต่อกับหน่วยความจำข้อมูลภายนอกเป็นการนำเอาสัญญาณจากขา $\overline{\text{RD}}$, $\overline{\text{WR}}$, $\overline{\text{PSEN}}$ ของไมโครคอนโทรลเลอร์ตัวที่ 2 และ $\overline{\text{RD}}$, $\overline{\text{WR}}$, P2.5 ของไมโครคอนโทรลเลอร์ตัวที่ 1 มาใช้ในการเลือกว่าจะติดต่อกับหน่วยความจำภายนอกตัวที่ 1 หรือ ตัวที่ 2 ซึ่งจะประกอบด้วย Logic Gate หลายชนิดมาต่อรวมกันดังภาพที่ 3.4 โดยมี Truth table ดังตารางที่ 3.1



ภาพที่ 3.4 แสดงวงจรควบคุมการติดต่อกับหน่วยความจำภายนอก

จะเห็นว่าที่เอาท์พุทของ Multiplex ซึ่งเป็นบัสข้อมูลของหน่วยความจำภายนอกนั้นมีลักษณะเป็นบัสข้อมูล 2 ทิศทางมีโอกาสเกิดการชนกันของข้อมูลได้ ผู้ออกแบบจึงได้เพิ่ม Tristate Buffer เข้าไปเพื่อป้องกันการชนกันของข้อมูลโดยมีสัญญาณที่ใช้กระตุ้นการทำงานของหน่วยความจำภายนอก และ P1.0 ของไมโครคอนโทรลเลอร์ตัวที่ 1 เป็นตัวควบคุมการทำงานของ Tristate Buffer

การทำงานของ Tristate Buffer ที่เอาท์พุทของ Multiplex นั้นมีโอกาสชนกันของข้อมูลที่เป็นเอาท์พุทของ Multiplex กับข้อมูลของหน่วยความจำภายนอกในสภาวะการอ่านข้อมูล Tristate Buffer ที่เพิ่มเข้าไปจะทำงานเมื่อมีสัญญาณ $\overline{\text{WE}}$ เป็น '0' ซึ่งเป็นสัญญาณการเขียนหน่วยความจำภายนอก โดยในสภาวะที่ไม่ทำงานเอาท์พุทของ Tristate Buffer จะเป็นค่าความต้านทานสูง ส่วน

อีกด้านซึ่งข้อมูลมีโอกาสชนกับข้อมูล P0 ของไมโครคอนโทรลเลอร์ โดย Tristate Buffer ที่เพิ่มเข้าไปจะทำงานเมื่อมีสัญญาณ $\overline{OE}$ เป็น '0' ซึ่งเป็นสัญญาณการอ่านหน่วยความจำภายนอก โดยมีสัญญาณ P1.0 ทำหน้าที่เลือกว่าให้เอาท์พุทออก P0 ของไมโครคอนโทรลเลอร์ตัวที่ 1 หรือตัวที่ 2 โดยมี Truth table ดังตารางที่ 3.2 และ 3.3

ตารางที่ 3.1 แสดง Truth table ของวงจรควบคุมการติดต่อกับหน่วยความจำภายนอก

INPUT						OUTPUT			
$\overline{PSEN\,2}$	$\overline{RD\,2}$	$\overline{WR\,2}$	P2.5	$\overline{RD\,1}$	$\overline{WR\,1}$	$\overline{OE\,1}$	$\overline{WE\,1}$	$\overline{OE\,2}$	$\overline{WE\,2}$
1	1	1	0	1	1	1	1	1	1
1	1	1	0	1	0	1	0	1	1
1	1	1	1	1	0	1	1	1	0
0	1	1	0	1	1	0	1	1	1
0	1	1	1	1	1	0	1	1	1
1	1	1	0	0	1	0	1	1	1
1	1	1	1	0	1	1	1	0	1
1	0	1	0	1	1	1	1	0	1
1	0	1	1	1	1	1	1	0	1
1	1	0	0	1	1	1	1	1	0
1	1	0	1	1	1	1	1	1	0

ตารางที่ 3.2 แสดง Truth table ของสัญญาณควบคุม Tristate Buffer สภาวะการเขียนข้อมูล

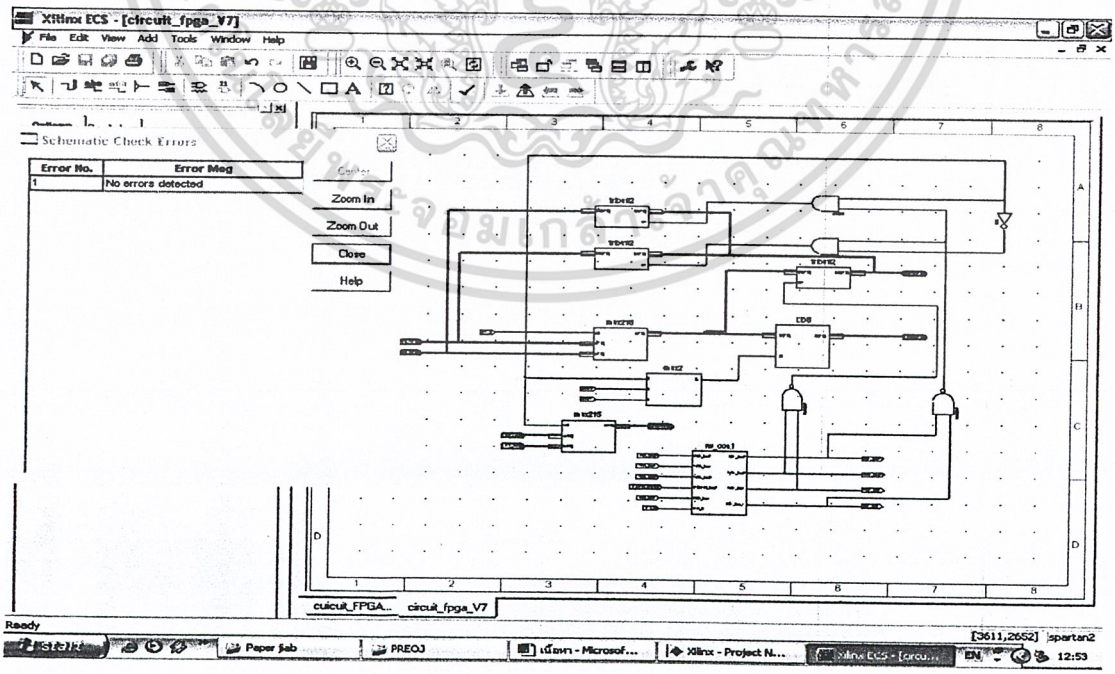
INPUT			OUTPUT
$\overline{WE\,1}$	$\overline{WE\,2}$	Data	Data
1	1	Di	Z
0	1	Di	Di
1	0	Di	Di

ตารางที่ 3.3 แสดง Truth table ของสัญญาณควบคุม Tristate Buffer สภาวะการอ่านข้อมูล

INPUT					OUTPUT	
$\overline{OE\ 1}$	$\overline{OE\ 2}$	P1.0	DataA	DataB	DataA	DataB
1	1	0	Da	Db	Z	Z
1	1	1	Da	Db	Z	Z
0	1	0	Da	Db	Da	Z
0	1	1	Da	Db	Z	Db
1	0	0	Da	Db	Da	Z
1	0	1	Da	Db	Z	Db

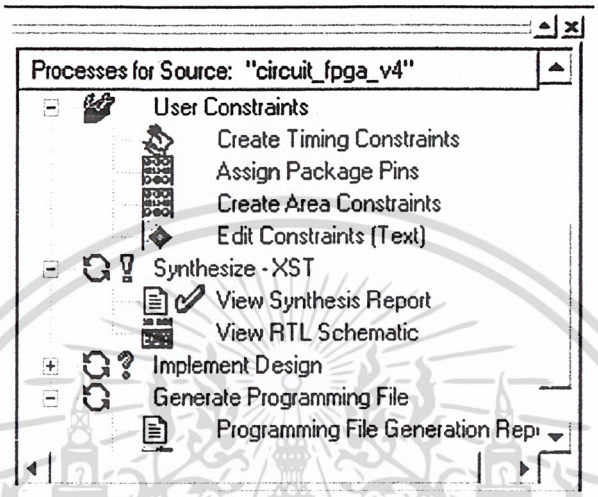
3.1.3 ตรวจสอบความถูกต้องของวงจร และการสังเคราะห์วงจร (Synthesis)

ทำการตรวจสอบวงจรที่ออกแบบว่าถูกต้องตามหลักการออกแบบของโปรแกรม หรือไม่ โดยการคลิกที่ไอคอน ✓ หรือเลือกที่เมนู Tools -> Check Schematic จะปรากฏวินโดวส์ Schematic Check Errors ถ้าออกแบบถูกต้องจะปรากฏข้อความ No errors detected ในช่อง Error Msg แต่ถ้าผลในการตรวจสอบไม่ถูกต้องจะปรากฏข้อความแสดงจุดที่เกิดการผิดพลาดในช่อง Error Msg และภาพวงจรก็จะเกิดแถบสีตรงที่เกิดความผิดพลาด



ภาพที่ 3.5 แสดงการตรวจสอบความถูกต้อง

เมื่อตรวจสอบความถูกต้องเสร็จเรียบร้อยแล้ว ให้กลับสู่วินโดวส์ Xilinx-Project Navigator ดับเบิ้ลคลิ๊กที่ไอคอน Synthesis-XST เมื่อทำการสังเคราะห์วงจรเสร็จเรียบร้อยแล้วจะปรากฏเครื่องหมาย ! แสดงว่าการสังเคราะห์เสร็จเรียบร้อย แต่กระบวนการย่อยอื่นๆ ยังไม่เสร็จเรียบร้อย ดังภาพที่ 3.6



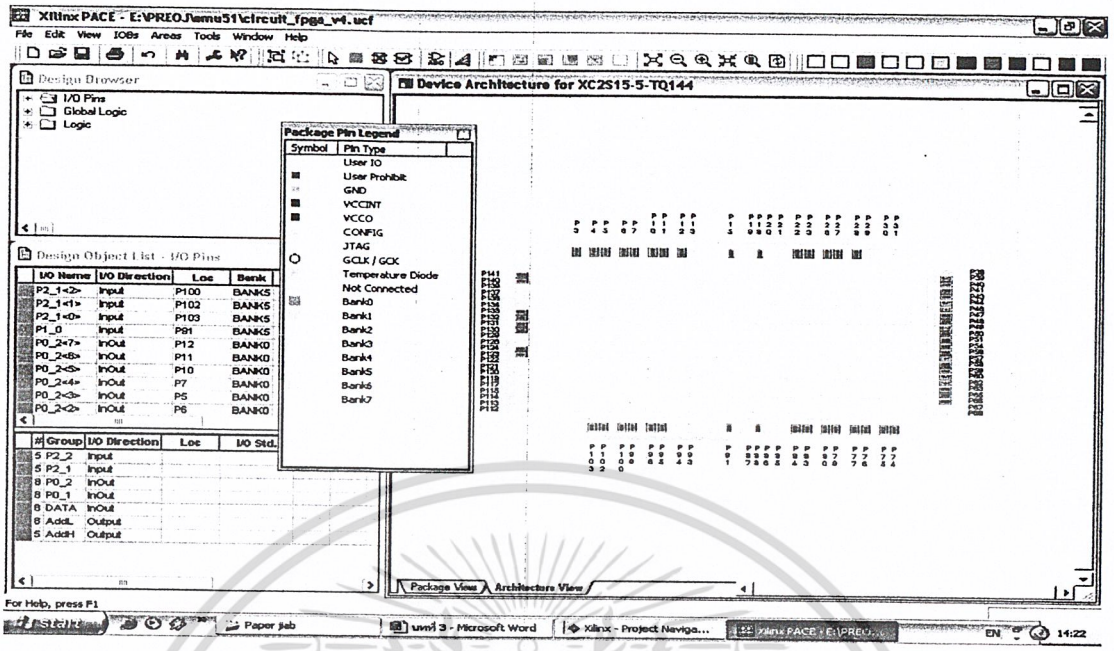
ภาพที่ 3.6 แสดงการสังเคราะห์วงจร

3.1.4 กำหนดขาให้กับ FPGA เพื่อเชื่อมต่อกับอุปกรณ์ภายนอก

เมื่อสังเคราะห์วงจรเรียบร้อย ในส่วนนี้จะเป็นการกำหนดขาอินพุท และเอาท์พุทให้กับ FPGA เพื่อเชื่อมต่อกับอุปกรณ์ภายนอก ซึ่งในที่นี้คือ ไมโครคอนโทรลเลอร์ (MCS-51) 2 ตัว และหน่วยความจำ RAM เบอร์ 6264 ซึ่งมีขนาด 8 Kbyte 2 ตัว ซึ่งขา Address และขา Data ของ RAM ทั้ง 2 ตัวเชื่อมถึงกัน โดยการดับเบิ้ลคลิ๊กที่ไอคอน Assign Package Pins ในวินโดวส์ Xilinx-Project Navigator จะปรากฏวินโดวส์ ดังภาพที่ 3.7 กำหนดขาต่างๆ ตามตารางที่ 3.4 ลงในช่อง Loc ซึ่งอยู่ภายในช่อง Design Object List –I/O Pins

ตารางที่ 3.4 แสดงการกำหนดขาเพื่อเชื่อมต่อกับอุปกรณ์ภายนอก

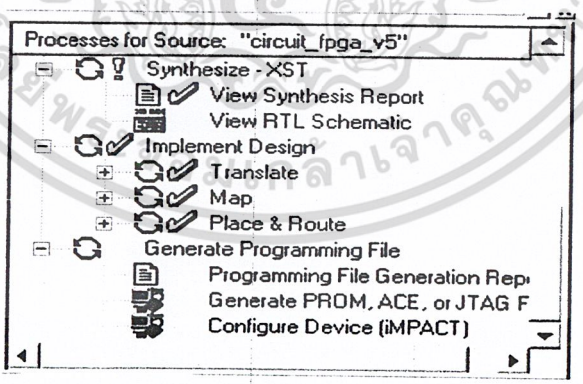
MCS-51 (Master)		MCS-51 (Slave)		RAM	
ขา MCS-51	ขา FPGA	ขา MCS-51	ขา FPGA	ขา RAM	ขา FPGA
P0.0	74	P0.0	4	A0	49
P0.1	76	P0.1	3	A1	50
P0.2	75	P0.2	6	A2	54
P0.3	77	P0.3	5	A3	56
P0.4	79	P0.4	7	A4	57
P0.5	80	P0.5	10	A5	59
P0.6	83	P0.6	11	A6	63
P0.7	84	P0.7	12	A7	65
P1.0	112	P2.0	28	A8	62
P1.1	114	P2.1	27	A9	60
P1.2	113	P2.2	26	A10	51
P1.3	115	P2.3	23	A11	58
P1.4	118	P2.4	22	A12	66
P1.5	117	P2.5	21	D0	47
P2.0	103	P2.6	20	D1	43
P2.1	102	P2.7	19	D2	41
P2.2	100	P3.0	134	D3	40
P2.3	99	P3.1	136	D4	39
P2.4	96	P3.3	137	D5	44
P2.5	95	P3.4	139	D6	46
P2.6	94	P3.6	140	D7	48
P2.7	93	P3.7	141	OE_BAR1	132
P3.3	120	ALE	13	WE_BAR1	130
P3.4	121	$\overline{\text{PSEN}}$	30	OE_BAR2	133
P3.5	122			WE_BAR2	131
P3.6	123				
P3.7	124				
ALE	87				



ภาพที่ 3.7 แสดงการกำหนดขาเพื่อเชื่อมต่อกับอุปกรณ์ภายนอก

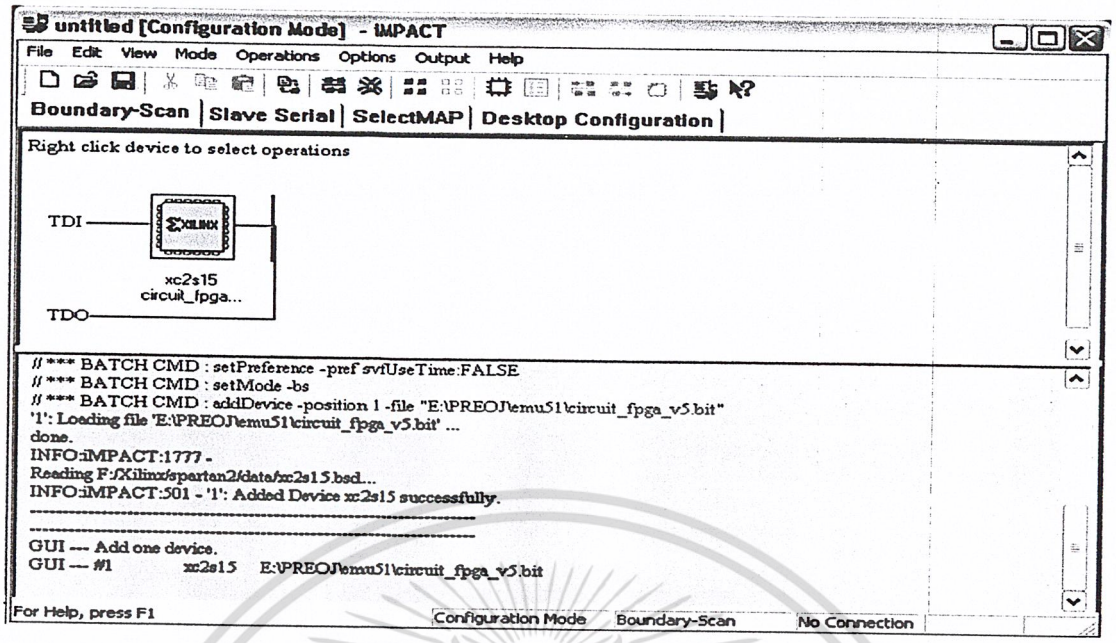
3.1.5 การโปรแกรมข้อมูลวงจรลงชิพ

เมื่อกำหนดขาเรียบร้อยแล้วให้กลับสู่วินโดวส์ Xilinx-Project Navigator เพื่อทำการ Implement โดยการดับเบิ้ลคลิ๊กที่ไอคอน Implement Design เมื่อทำการ Implement เสร็จเรียบร้อย จะมีเครื่องหมายถูก ✔ เกิดขึ้นดังภาพที่ 3.8



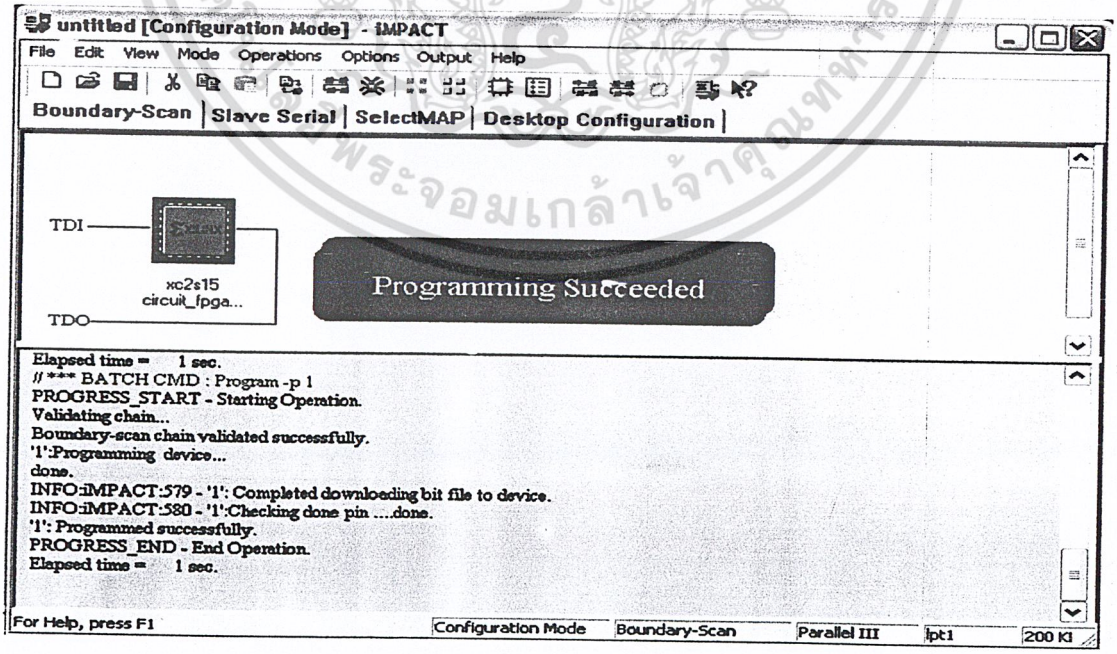
ภาพที่ 3.8 แสดงการ Implement

การดาวน์โหลดข้อมูลลง FPGA เริ่มด้วยการดับเบิ้ลคลิ๊กที่ไอคอน Configure Device (iMPACT) หลังจากนั้นจะปรากฏวินโดวส์ Untitled-iMPACT ดังภาพที่ 3.9



ภาพที่ 3.9 แสดงวินโดวส์ Untitled-iMPACT

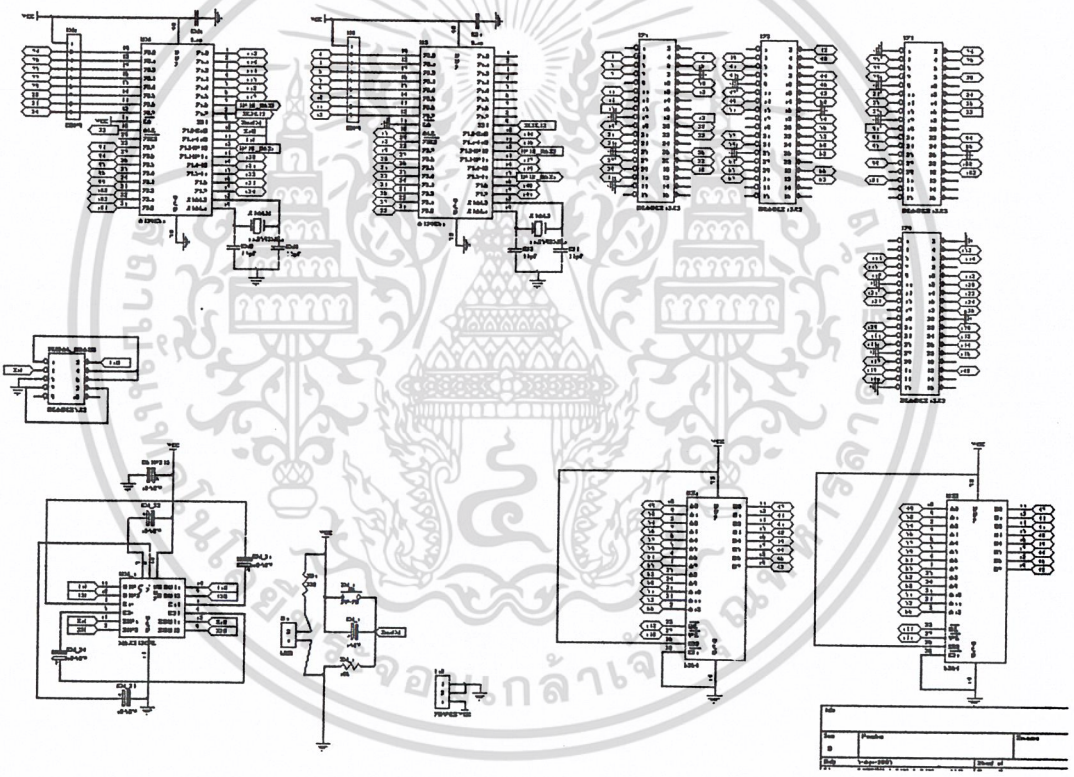
ดาวน์โหลดข้อมูลลง FPGA โดยการคลิ๊กขวาที่ภาพไอซี เลือก Program จะเกิดวินโดวส์ Program Option คลิ๊ก OK หลังจากนั้นจะปรากฏวินโดวส์ Operating Status เพื่อแสดงสภาวะของการดาวน์โหลด เมื่อกระบวนการดาวน์โหลดข้อมูลเสร็จสมบูรณ์จะปรากฏข้อความ Programming Succeeded ดังภาพที่ 3.10



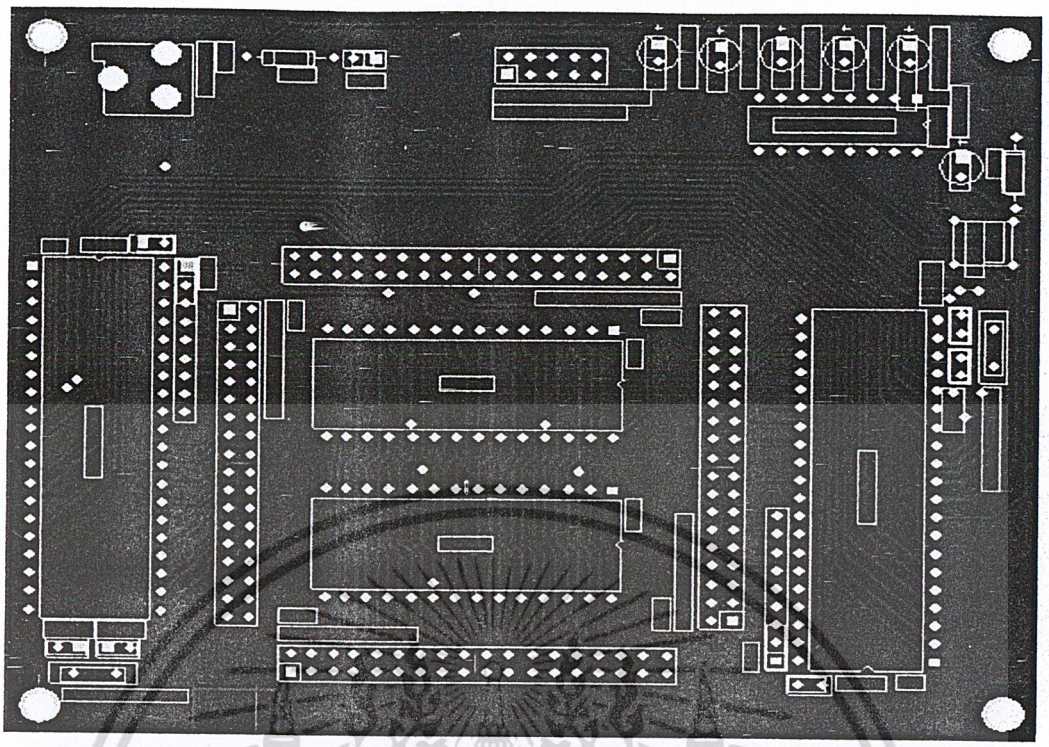
ภาพที่ 3.10 แสดงการดาวน์โหลดข้อมูลลง FPGA ที่สมบูรณ์แล้ว

3.2 การออกแบบสร้างบอร์ดไมโครคอนโทรลเลอร์เพื่อใช้งานร่วมกับ FPGA

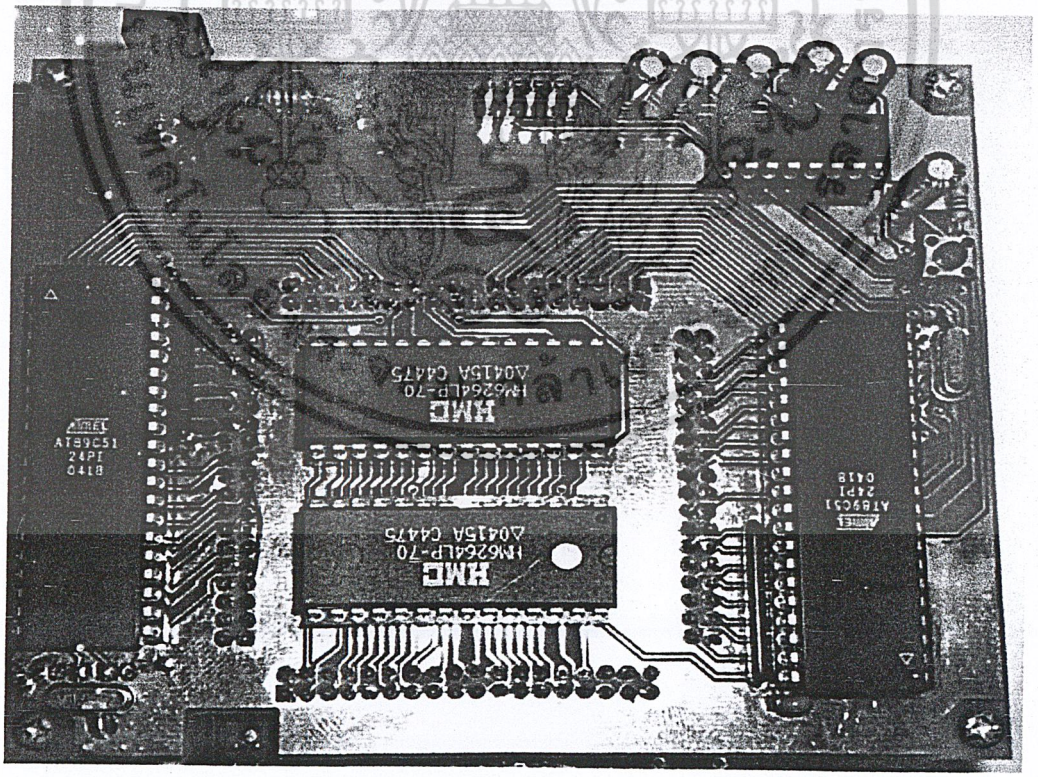
การออกแบบสร้างบอร์ดนี้จะประกอบด้วย ไมโครคอนโทรลเลอร์ (MCS-51) 2 ตัว หน่วยความจำข้อมูล (RAM เบอร์ 6264) 2 ตัว และส่วนของวงจรรับส่งข้อมูลผ่านพอร์ตอนุกรม ซึ่งขาของไมโครคอนโทรลเลอร์ และหน่วยความจำข้อมูลจะต่ออยู่กับคอนเนคเตอร์ขนาด 36 ขา จำนวน 4 ตัว ดังตารางที่ 3.4 เพื่อใช้ในการเชื่อมต่อกับบอร์ด FPGA โดยรายละเอียดสามารถดูได้จากภาพที่ 3.10 ซึ่งในการทำแผ่น PCB เพื่อใช้งานร่วมกับ FPGA นั้นจะต้องทำแผ่น PCB 2 ด้าน โดยที่ด้านหลัง และที่ว่างด้านบนของแผ่น PCB จะทำเป็นกราวน์ทั้งหมดเพื่อช่วยในการลดสัญญาณรบกวน ซึ่งสัญญาณรบกวนนั้นจะมีผลต่อการส่งข้อมูล อาจทำให้ข้อมูลที่ทำการรับส่งผิดพลาดได้



ภาพที่ 3.11 แสดงวงจรของบอร์ดไมโครคอนโทรลเลอร์



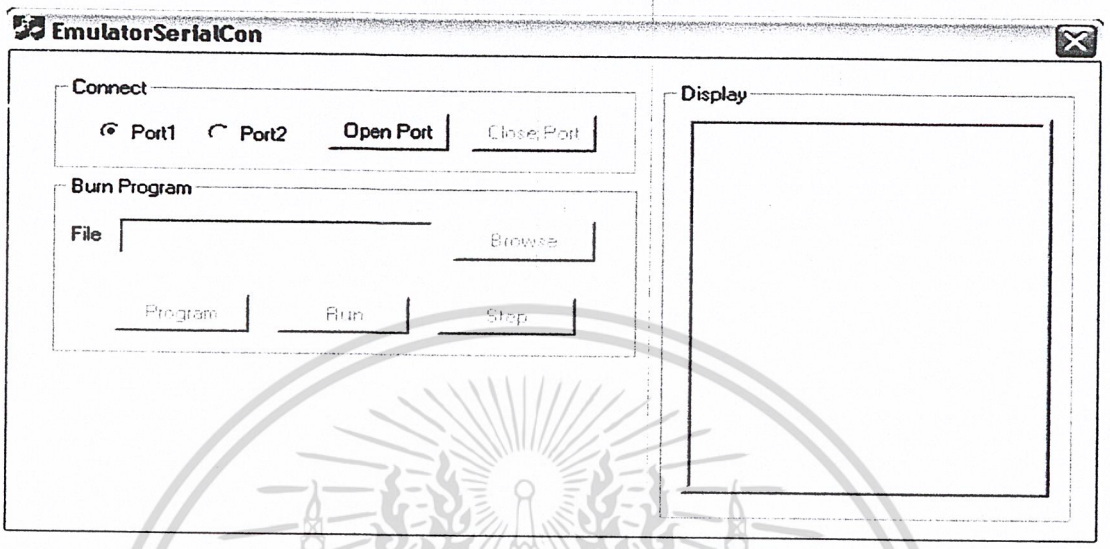
ภาพที่ 3.12 แสดงลายทองแดงของบอร์ดไมโครคอนโทรลเลอร์



ภาพที่ 3.13 แสดงบอร์ดไมโครคอนโทรลเลอร์

3.3 Control Display

จะประกอบด้วย 3 ส่วน คือ Connect,Burn Program,Display โดยส่วนต่างๆ จะมีหน้าที่ดังนี้



ภาพที่ 3.14 แสดง Control Display

3.3.1 ส่วน Connect

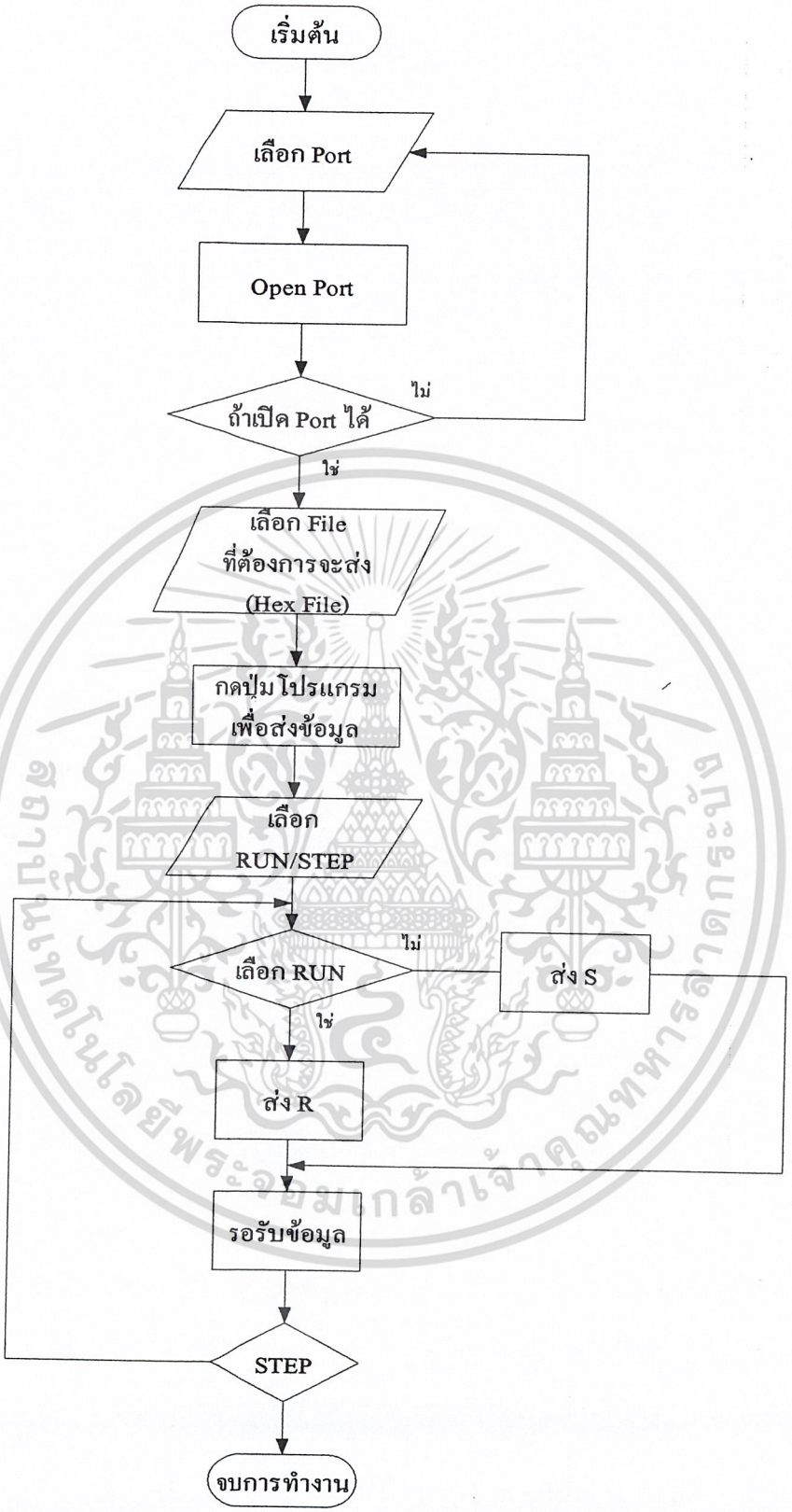
มีหน้าที่ในการเชื่อมต่อกับพอร์ตอนุกรมของไมโครคอนโทรลเลอร์ โดยเลือกว่าจะติดต่อผ่านพอร์ต1 หรือ พอร์ต 2 ของคอมพิวเตอร์ การติดต่อทำได้โดยการเลือกพอร์ตที่จะใช้ในการติดต่อ จากนั้นคลิ๊กที่ Open Port เมื่อต้องการหยุดการติดต่อให้คลิ๊กที่ Close Port

3.3.2 ส่วน Burn Program

มีหน้าที่ในควบคุมการทำงานของ Hardware ปุ่ม Browse ใช้ในการเลือกไฟล์ที่ต้องการส่งไปให้ Hardware โดยที่ไฟล์ที่ส่งจะเป็น .hex ช่อง File ใช้แสดงชื่อไฟล์ที่เลือก ปุ่ม Program ใช้ในการส่งข้อมูลซึ่งเป็น Programไปยัง Hardware ปุ่ม Run ใช้ในการสั่งให้ Hardware ทำงานตาม Program แบบครั้งเดียวจบ ปุ่ม Step ใช้ในการสั่งให้ Hardware ทำงานตาม Program ทีละคำสั่ง

3.3.3 ส่วน Display

ใช้แสดงค่าหน่วยความจำข้อมูล และค่ารีจิสเตอร์ภายในไมโครคอนโทรลเลอร์ตัวที่ 2 ซึ่งถูกใช้ให้ทำงานตามโปรแกรมที่ใส่ลงไป โดยจะแสดงเป็นค่าของตำแหน่ง แล้วตามด้วยข้อมูลที่อยู่ภายใน



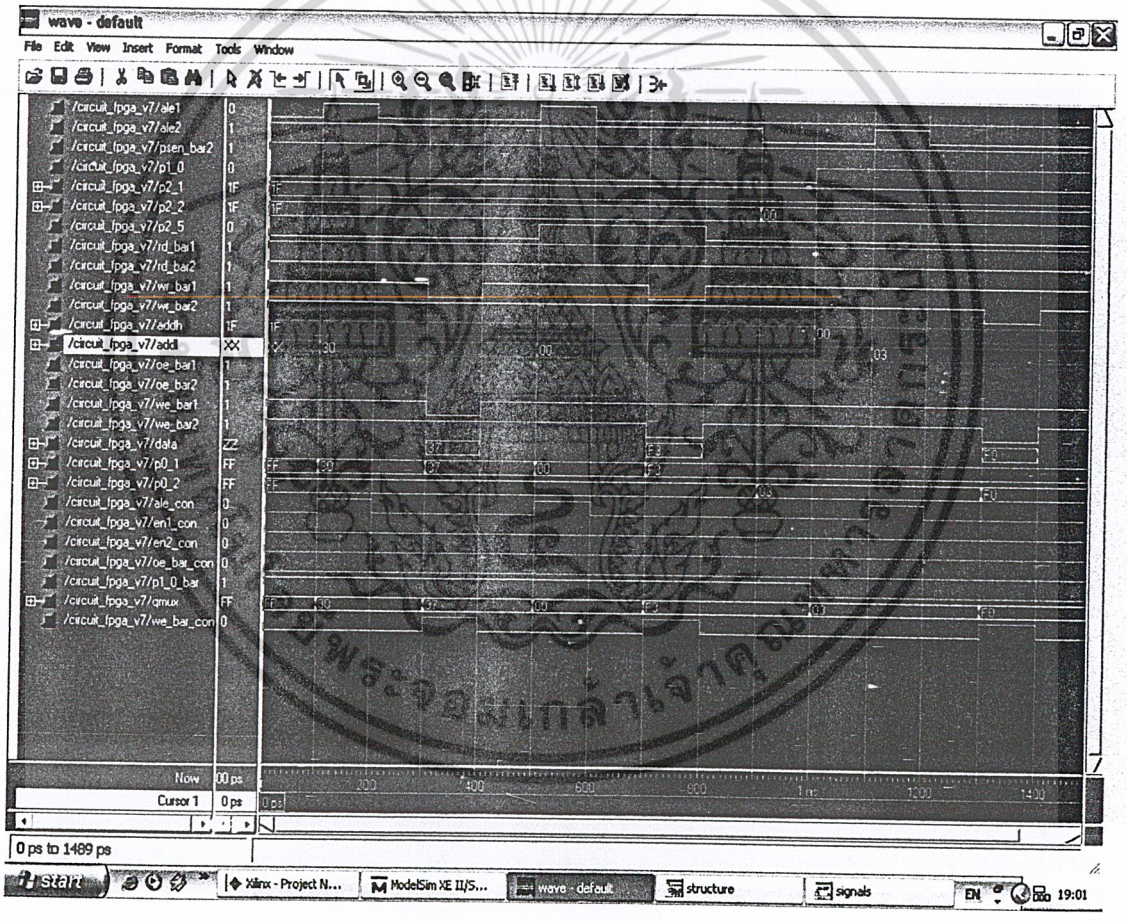
ภาพที่ 3.15 แสดง Flow Chart การใช้งานของ Control Display

บทที่ 4

ผลการทดลอง

4.1 ผลการจำลองการทำงานของวงจรภายใน FPGA

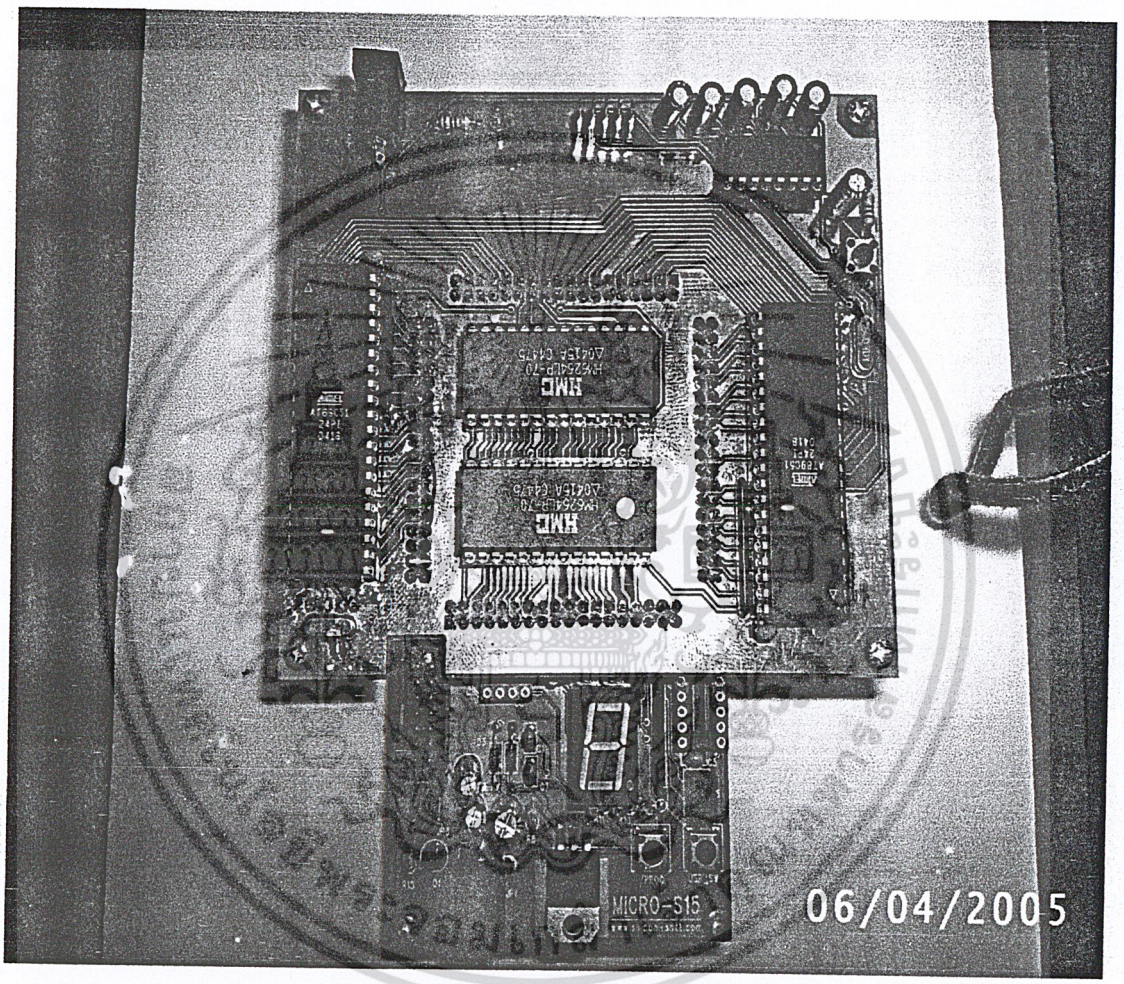
วงจรภายใน FPGA ที่ออกแบบขึ้นนั้นประกอบด้วยวงจรดิจิตอลที่สร้างขึ้นด้วยการเขียน VHDL และการวาดผังวงจร (Schematic) ซึ่งข้อดีของการออกแบบวงจรดิจิตอลภายใน FPGA คือ สามารถใช้โปรแกรม ModelSim XE ในการจำลองการทำงานก่อนเพื่อให้แน่ใจว่าวงจรที่ออกแบบสามารถทำงานได้อย่างถูกต้อง



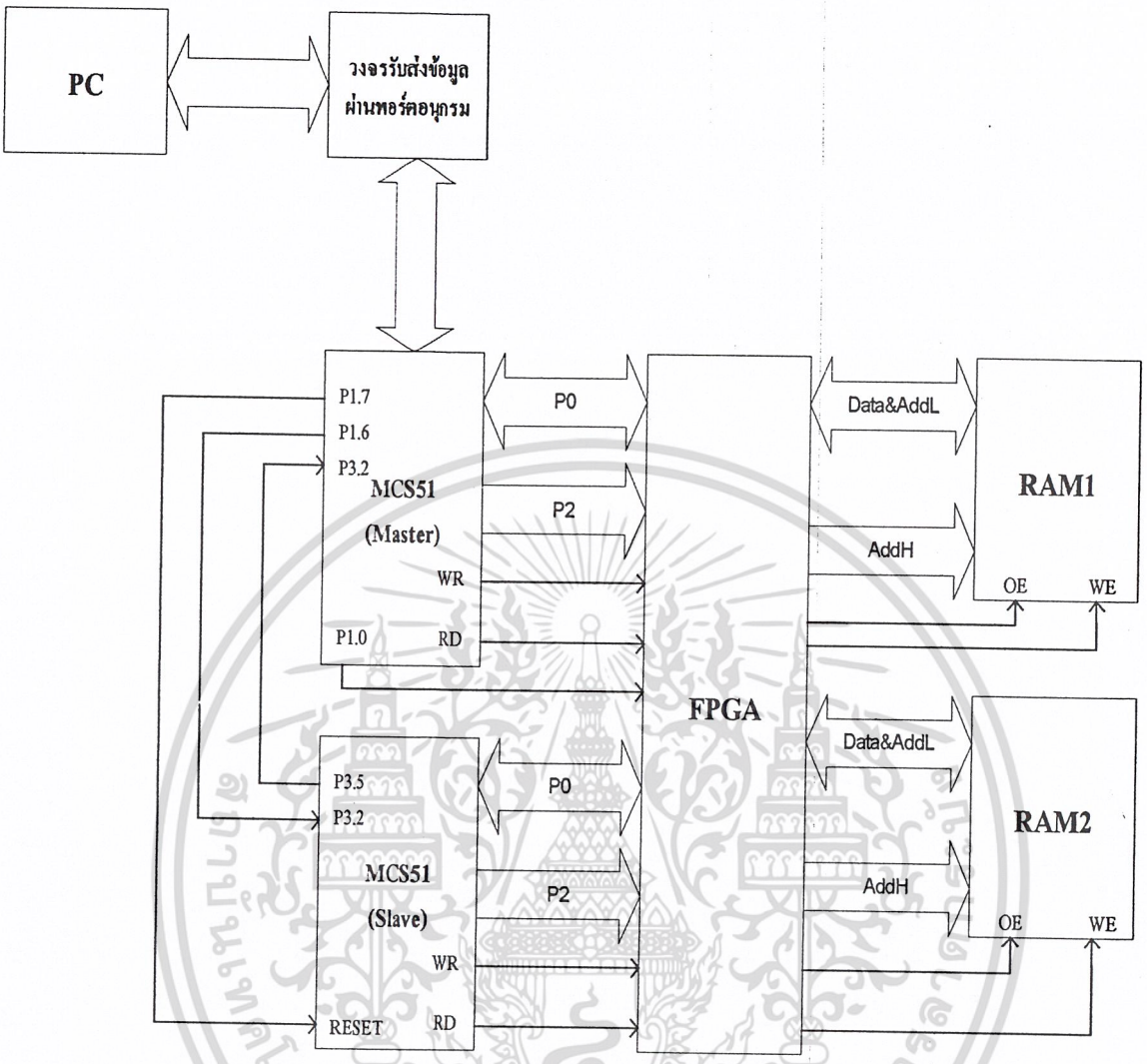
ภาพที่ 4.1 แสดงสัญญาณที่ได้จากการจำลองการทำงาน

4.2 การทดสอบการทำงานร่วมกับบอร์ดไมโครคอนโทรลเลอร์ที่ออกแบบ

บอร์ดไมโครคอนโทรลเลอร์ที่ออกแบบขึ้นนี้จะประกอบด้วยอุปกรณ์หลักๆ คือไมโครคอนโทรลเลอร์เบอร์(MCS-51) AT89C51 จำนวน 2 ตัว, หน่วยความจำ (RAM) เบอร์ HM6264LP-70 จำนวน 2 ตัว ดังภาพที่ 4.2 เป็นการต่อบอร์ดไมโครคอนโทรลเลอร์ทีออกแบบขึ้นมากับบอร์ด FPGA เข้าด้วยกัน

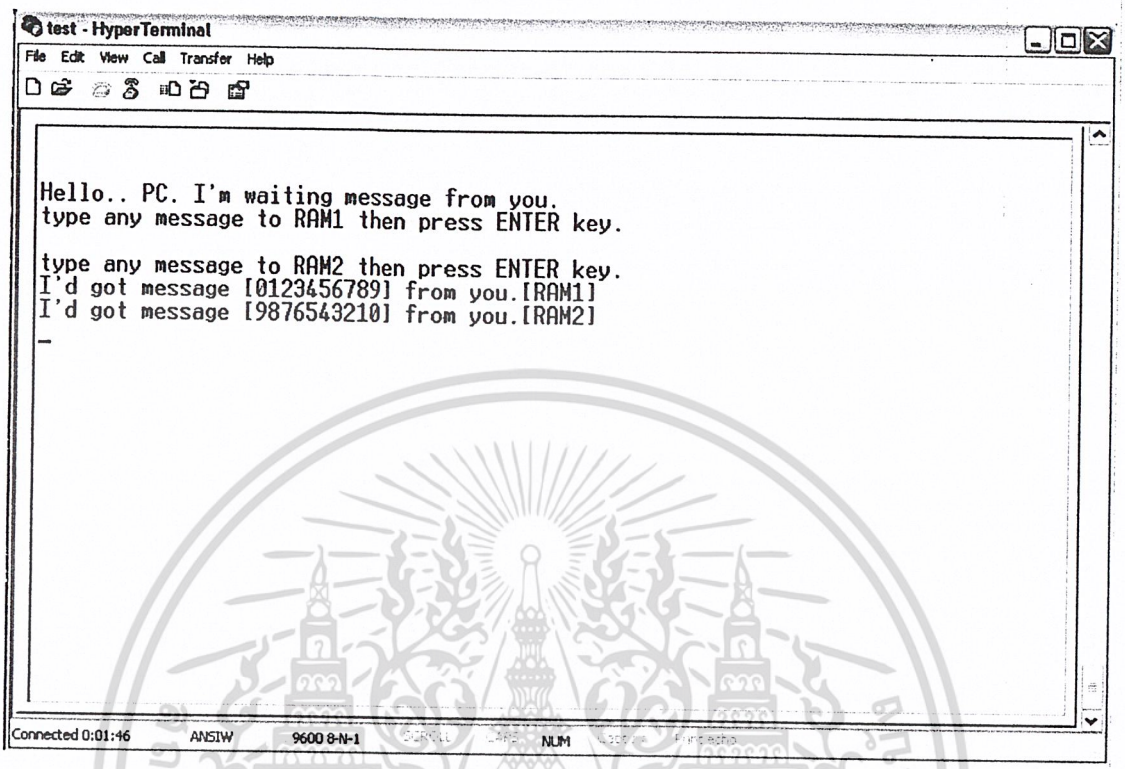


ภาพที่ 4.2 แสดงการต่อบอร์ดไมโครคอนโทรลเลอร์เข้ากับบอร์ด FPGA



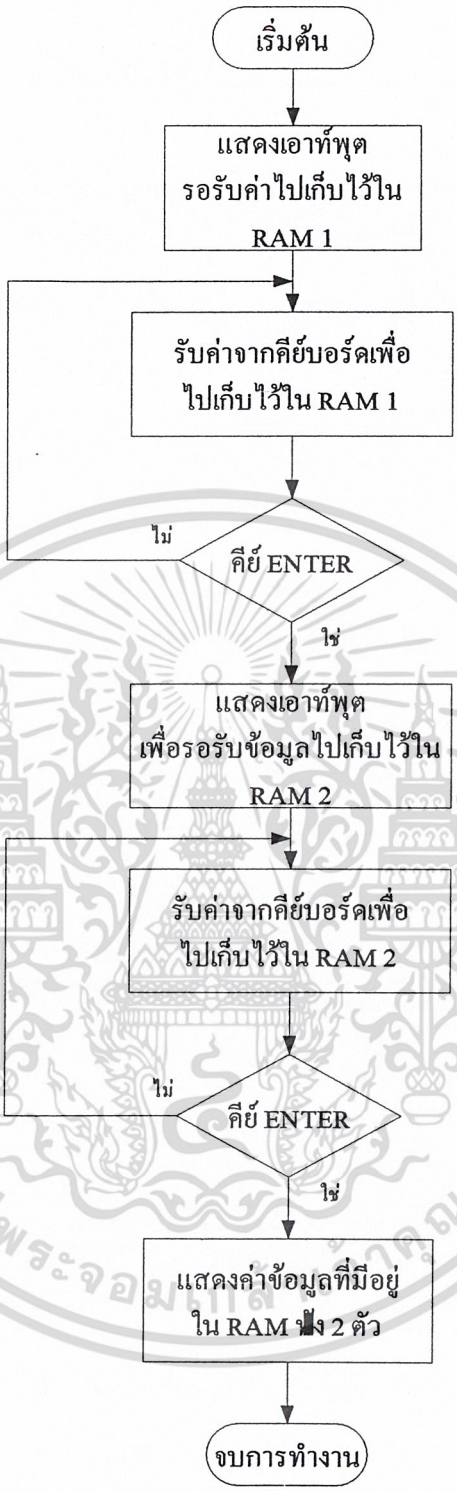
ภาพที่ 4.3 Block Diagram แสดงส่วนต่างๆ ของวงจร

4.2.1 ทดสอบการติดต่อระหว่างไมโครคอนโทรลเลอร์ กับหน่วยความจำข้อมูลภายนอก



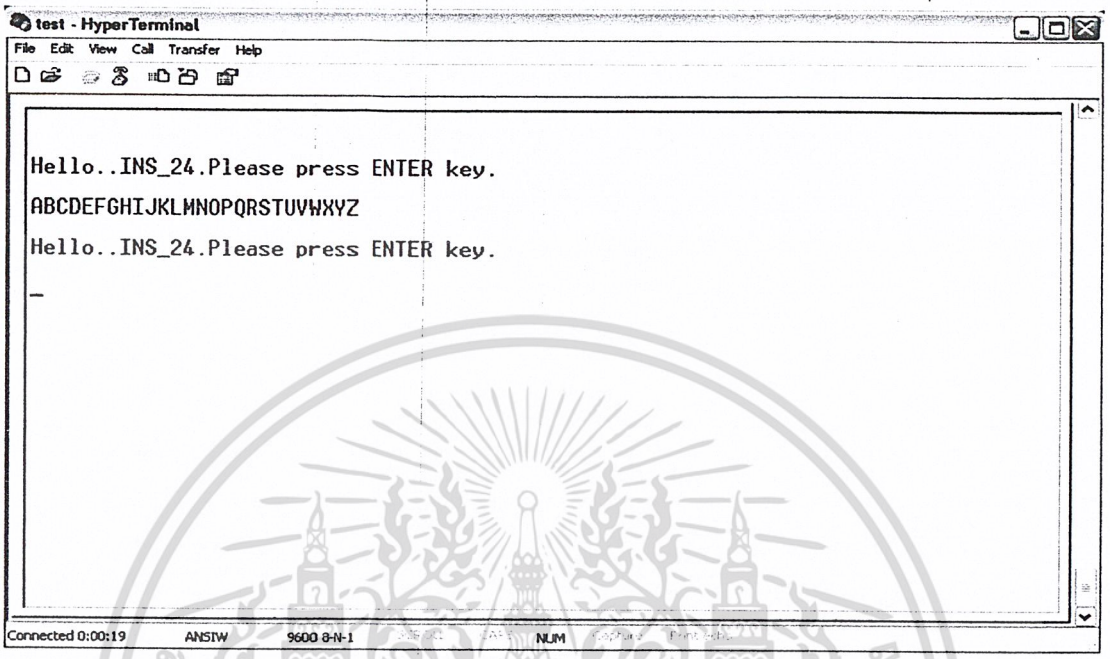
ภาพที่ 4.4 แสดงผลการทดสอบการติดต่อระหว่างไมโครคอนโทรลเลอร์ กับหน่วยความจำข้อมูล ภายนอก แสดงออก Hyper Terminal

เป็นการติดต่อระหว่างไมโครคอนโทรลเลอร์กับหน่วยความจำข้อมูลภายนอก 2 ตัว โดยในการทดสอบจะป้อนข้อมูล 0123456789 ไปเก็บไว้ในหน่วยความจำข้อมูลตัวที่ 1 และข้อมูล 9876543210 ไปเก็บไว้ใน RAM ตัวที่ 2 โดยคีย์บอร์ด หลังจากนั้นโปรแกรมจะแสดงค่าภายในหน่วยความจำข้อมูลทั้ง 2 ตัวออก Hyper Terminal เพื่อให้เข้าใจมากขึ้นดู Flowchart ภาพที่ 4.5 ประกอบ



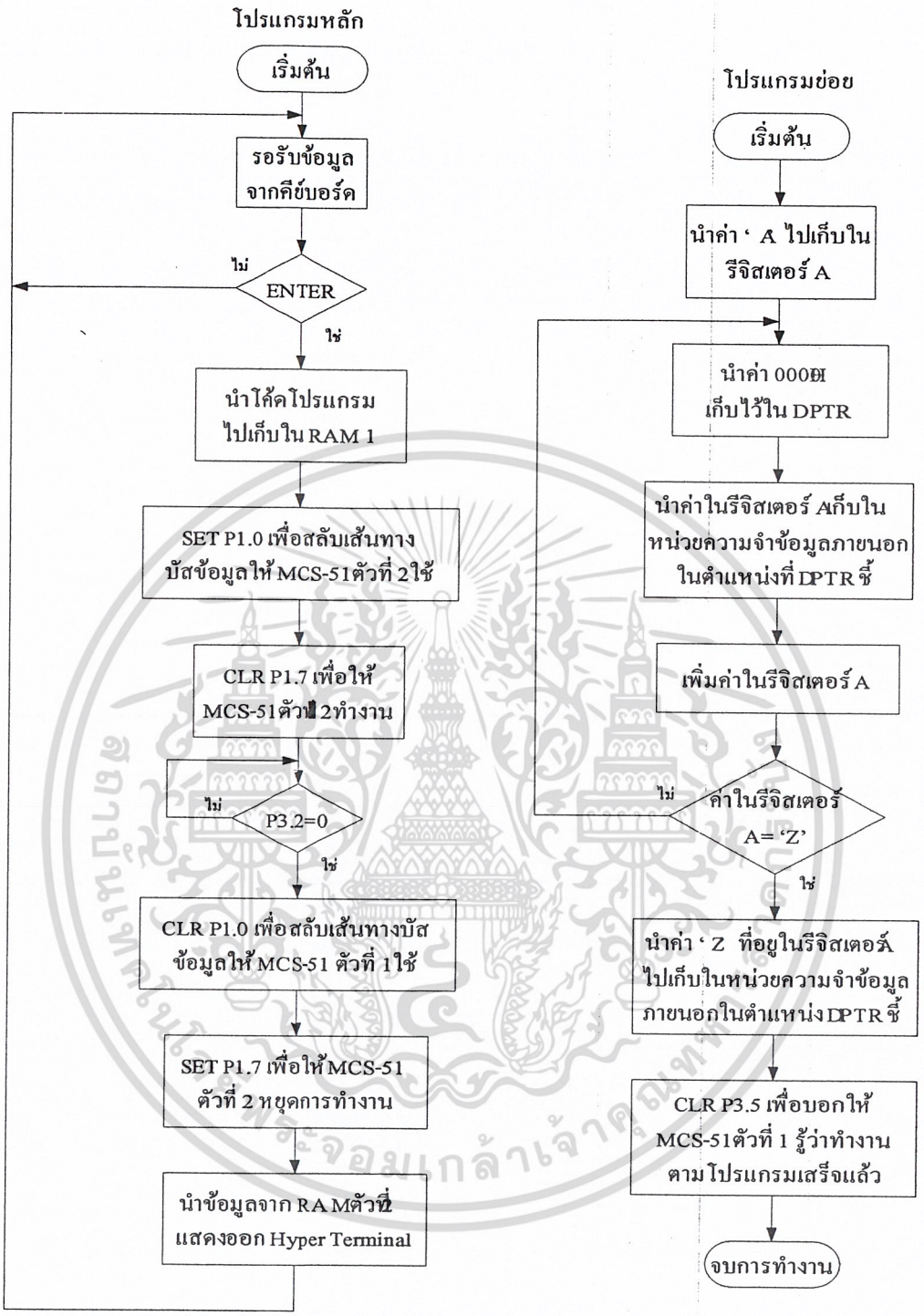
ภาพที่ 4.5 แสดง Flowchart ทดสอบการติดต่อระหว่างไมโครคอนโทรลเลอร์กับหน่วยความจำข้อมูลภายนอก

4.2.2 การทดสอบทำงานของไมโครคอนโทรลเลอร์ตัวที่ 2 ตามโค้ดภายในหน่วยความจำภายนอกตัวที่ 1 แบบที่ 1



ภาพที่ 4.6 แสดงผลการทดสอบการทำงานในแบบที่ 1 โดยแสดงออก Hyper Termina!

เป็นการทำงานของไมโครคอนโทรลเลอร์ตัวที่ 2 ตามโค้ดภายในหน่วยความจำภายนอกตัวที่ 1 เมื่อไมโครคอนโทรลเลอร์ตัวที่ 1 จะทำการส่งข้อมูลซึ่งเป็นโค้ดของโปรแกรมไปเก็บไว้ที่หน่วยความจำข้อมูลตัวที่ 1 เมื่อกระบวนการส่งข้อมูลระหว่าง ไมโครคอนโทรลเลอร์ที่ 1 กับหน่วยความจำข้อมูลตัวที่ 1 เสร็จสิ้น ไมโครคอนโทรลเลอร์ตัวที่ 1 ก็จะทำการเซ็ต P1.0 เพื่อสลับเส้นทางการส่งข้อมูลเพื่อให้ ไมโครคอนโทรลเลอร์ตัวที่ 2 สามารถทำการอ่านโค้ดจาก หน่วยความจำข้อมูลตัวที่ 1 ได้ หลังจากนั้น ไมโครคอนโทรลเลอร์ตัวที่ 1 จะส่งสัญญาณไปเคลียร์ขารีเซตของ ไมโครคอนโทรลเลอร์ตัวที่ 2 โดยการเคลียร์ P1.7 เพื่อให้สามารถทำงานตามโค้ดที่อ่านมาจาก หน่วยความจำข้อมูลตัวที่ 1 โดยจะเป็นนำค่า A – Z ไปเก็บในหน่วยความจำข้อมูลตัวที่ 2 หลังจากนั้น ไมโครคอนโทรลเลอร์ตัวที่ 2 ก็จะสร้างสัญญาณเพื่อส่งไปบอกไมโครคอนโทรลเลอร์ตัวที่ 1 ว่าการทำงานตามโปรแกรมเสร็จเรียบร้อย โดยการเคลียร์ P3.5 ไมโครคอนโทรลเลอร์ตัวที่ 1 จะทำการสลับเส้นทางการส่งข้อมูล เพื่อไปดึงค่าข้อมูลในหน่วยความจำข้อมูลตัวที่ 2 ไปแสดงค่าบน Hyper Terminal เพื่อให้เข้าใจมากขึ้นดู Flowchart ภาพที่ 4.7 ประกอบ

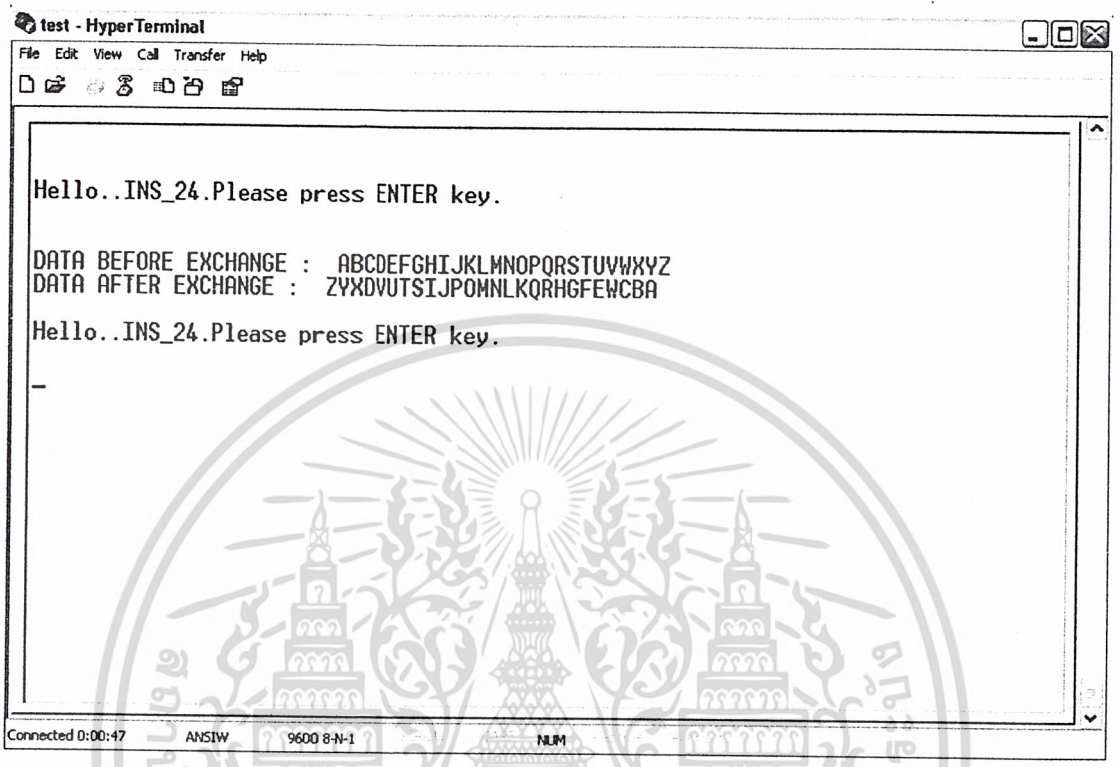


ก. โปรแกรมที่ให้ไมโครคอนโทรลเลอร์ตัวที่ 1ทำงาน

ข. โปรแกรมที่ให้ไมโครคอนโทรลเลอร์ตัวที่ 2 ทำงาน

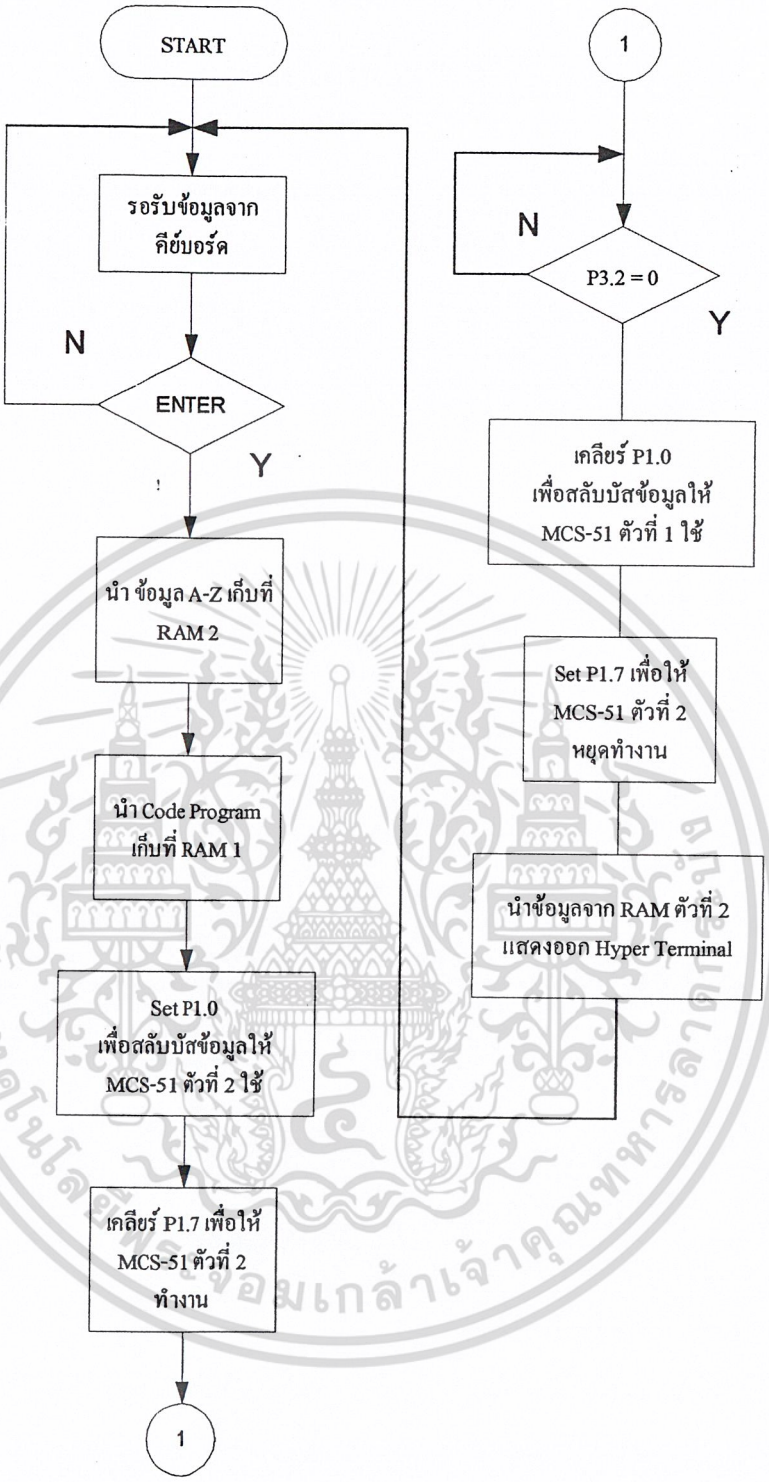
ภาพที่ 4.7 Flowchart แสดงการทำงานของไมโครคอนโทรลเลอร์ ในการทดสอบแบบที่ 1

4.2.3 การทดสอบทำงานของไมโครคอนโทรลเลอร์ตัวที่ 2 ตามโค้ดภายในหน่วยความจำภายนอกตัวที่ 1 แบบที่ 2



ภาพที่ 4.8 แสดงผลการทดสอบการทำงานในแบบที่ 2 โดยแสดงออก Hyper Terminal

การทดสอบในแบบที่ 2 นั้นเมื่อได้รับการคีย์ ENTER ไมโครคอนโทรลเลอร์ตัวที่ 1 จะนำค่า A-Z ไปเก็บที่ RAM ตัวที่ 2 จากนั้นจะนำโค้ดโปรแกรมไปเก็บที่ RAM ตัวที่ 1 แล้วสลับเส้นทางบัสข้อมูลด้วยการเซต P1.0 จากนั้นจึงเคลียร์ขารีเซตของไมโครคอนโทรลเลอร์ตัวที่ 2 เพื่อให้ไมโครคอนโทรลเลอร์ตัวที่ 2 ทำงานตามโค้ดโปรแกรมใน RAM ตัวที่ 1 ซึ่งเป็นโปรแกรมที่ใช้สลับตำแหน่งข้อมูลที่อยู่ใน RAM ตัวที่ 2 จากนั้นไมโครคอนโทรลเลอร์ตัวที่ 1 ก็จะนำค่าที่อยู่ใน RAM ตัวที่ 2 ซึ่งถูกไมโครคอนโทรลเลอร์ตัวที่ 2 สลับตำแหน่งข้อมูลแสดงออก Hyper Terminal โดยผลการทดสอบเป็นดังภาพที่ 4.8



ภาพที่ 4.9 Flowchart แสดงการทำงานของไมโครคอนโทรลเลอร์ตัวที่ 1 ในการทดสอบแบบที่ 2

ภาพที่ 4.10 Flowchart แสดงการทำงานของไมโครคอนโทรลเลอร์ตัวที่ 2 ในการทดสอบแบบที่ 2

บทที่ 5
บทสรุปและข้อเสนอแนะ

จากผลการทดลองที่ผ่านมา สรุปได้ว่าการออกแบบวงจรดิจิตอลภายในตัว FPGA นั้น สามารถใช้แทนวงจรดิจิตอล และลอจิกเกทชนิดต่างๆ ได้ ทั้งยังสามารถนำวงจร และลอจิกเกทเหล่านั้นมาเชื่อมต่อกันให้เป็นวงจรขนาดที่ใหญ่ขึ้นได้ ทำให้สามารถลดความยุ่งยากในการต่อสายเพื่อเชื่อมแต่ละวงจรเข้าด้วยกัน ซึ่งในการทดลองจะใช้ FPGA เป็นตัวกลางทำหน้าที่เป็นวงจรสลับเส้นทางการใช้บัสข้อมูลในการติดต่อกันระหว่าง ไมโครคอนโทรลเลอร์ กับหน่วยความจำข้อมูลภายนอก โดยที่ไมโครคอนโทรลเลอร์ตัวที่ 1 สามารถอ่าน และเขียนหน่วยความจำข้อมูลภายนอก ทั้ง 2 ตัวได้ ส่วนไมโครคอนโทรลเลอร์ตัวที่ 2 นั้น สามารถอ่าน และเขียนหน่วยความจำข้อมูลภายนอกตัวที่ 2 ได้ โดยหน่วยความจำข้อมูลภายนอกตัวที่ 1 นั้นจะสามารถอ่านได้อย่างเดียวซึ่งเปรียบเสมือนกับเป็นหน่วยความจำโปรแกรมภายนอก ในส่วนของ Control display ที่ออกแบบขึ้นมาเพื่อใช้ในการแสดงผลนั้นยังไม่สามารถทำงานได้

5.1 ปัญหาที่พบในการทำวิจัย

1. ในการออกแบบวงจรรวมดิจิตอลภายใน FPGA เมื่อทำการจำลองการทำงานของวงจร (Simulate) โดยใช้ซอฟท์แวร์นั้นผลของการจำลองการทำงานนั้นสามารถทำงานได้ แต่เมื่อนำมาทดลองกับวงจรที่ใช้งานไม่สามารถทำงานได้

2. ในการออกแบบบอร์ดใช้งานร่วมกับ FPGA ซึ่งมีลักษณะการส่งข้อมูลแบบขนานนั้นจะพบสัญญาณรบกวน ซึ่งทำให้การรับส่งข้อมูลนั้นเกิดความผิดพลาด

3. ส่วน Control Display ไม่สามารถแสดงผลได้ตามที่ได้ออกแบบไว้

5.2 ข้อเสนอแนะและแนวทางในการพัฒนา

1. พัฒนาบอร์ดใช้งานร่วมกับ FPGA ในส่วนของไมโครคอนโทรลเลอร์ตัวที่ 2 ให้มีอุปกรณ์เสริมเพื่อใช้ในการ Interface กับอุปกรณ์ภายนอก

2. พัฒนาโปรแกรมให้สามารถทำงานแบบ Break Point ได้

3. พัฒนาส่วน Control Display ให้สามารถแสดงผลได้

บรรณานุกรม

- ขรรค์ชัย ตุลละสกุล, 2546, การใช้งานโปรแกรมออกแบบวงจรรวมดิจิตอล, กรุงเทพฯ : ซีเอ็ดยูเคชั่น, 352 หน้า
- ชัยวัฒน์ ลิ้มพรจิตรวิไล และวรพจน์ กรแก้ววัฒนกุล, เรียนรู้การปฏิบัติการไมโครคอนโทรลเลอร์ MCS-51 แบบแฟลช, กรุงเทพฯ : อินโนเวตีฟ เอ็กเพอริเมนต์ จำกัด, 400 หน้า
- ชำนาญ ปัญญาใส และวัชรากร หนูทอง, 2547, ภาษา VHDL สำหรับการออกแบบวงจรดิจิตอล, กรุงเทพฯ : ซีเอ็ดยูเคชั่น, 432 หน้า
- http://www.xilinx.com
- http://www.astronlogic.com

ภาคผนวก

โปรแกรมทดสอบการติดต่อระหว่างไมโครคอนโทรลเลอร์ กับหน่วยความจำข้อมูลภายนอก

```
;-------------------------------------------------------------------
; Define User Register
;-------------------------------------------------------------------
COUNT1      EQU         020H
COUNT2      EQU         022H

;-------------------------------------------------------------------
; Main Program.
;-------------------------------------------------------------------
            ORG         0000H
MAIN:       MOV         TMOD,#021H
            MOV         TH1,#0FDH
            MOV         TL1,#0FDH
            SETB        TR1
            CLR         P1.0
            MOV         SCON,#040H
            SETB        REN

START:      MOV         DPTR,#HELLO_TEXT1
            ACALL       TX_TEXT
            MOV         DPTR,#1000H
            MOV         COUNT1,#00H

LOOP1:      ACALL       R_DATA
            MOVX        @DPTR,A
            CJNE        A,#00DH,RX_NEXT1
            MOV         DPTR,#HELLO_TEXT2
            ACALL       TX_TEXT
            MOV         DPTR,#2000H
            MOV         COUNT2,#00H
            SETB        REN
```

```
LOOP2:          ACALL       R_DATA
                MOVX        @DPTR,A
                CJNE        A,#00DH,RX_NEXT2
                MOV         DPTR,#GOT_TEXT1
                ACALL       TX_TEXT

                MOV         DPTR,#1000H
REPLY_LOOP1:    MOVX        A,@DPTR
                ACALL       TEXT_CMD
                INC         DPTR
                DJNZ        COUNT1,REPLY_LOOP1
                MOV         DPTR,#GOT_TEXT2
                ACALL       TX_TEXT
                MOV         DPTR,#GOT_TEXT1
                ACALL       TX_TEXT

                MOV         DPTR,#2000H
REPLY_LOOP2:    MOVX        A,@DPTR
                ACALL       TEXT_CMD
                INC         DPTR
                DJNZ        COUNT2,REPLY_LOOP2
                MOV         DPTR,#GOT_TEXT3
                ACALL       TX_TEXT
                SJMP        START

RX_NEXT1:       INC         DPTR
                INC         COUNT1
                AJMP        LOOP1

RX_NEXT2:       INC         DPTR
                INC         COUNT2
                AJMP   LOOP2
```

```
;-------------------------------------------------------------------
;RECIEVE DATA
;-------------------------------------------------------------------
R_DATA:     JNB     RI,$
            CLR     RI
            MOV     A,SBUF
            RET

;-------------------------------------------------------------------
;TRANMIT DATA
;-------------------------------------------------------------------
TEXT_CMD:   CLR     TI
            MOV     SBUF,A
            JNB     TI,$
            CLR     TI
            RET

;-------------------------------------------------------------------
; TX Serial Text from ROM Pointer
;-------------------------------------------------------------------
TX_TEXT:    CLR     TI
TX_LOOP:    CLR     A
            MOVC    A,@A+DPTR
            INC     DPTR
            CJNE    A,#0FFH,TX_CHAR
            RET
TX_CHAR:    MOV     SBUF,A
            JNB     TI,$
            CLR     TI
            AJMP    TX_LOOP
```

```
;-------------------------------------------------------------------
;Define Constant < Store in Flash EEPROM Program Memory >
;-------------------------------------------------------------------
HELLO_TEXT1:    DB    00AH,00DH
                DB    00AH,00DH
                DB    'Hello.. PC. I',027H,'m waiting message from
                      you.',00AH,00DH
                DB    'type any message to RAM1 then press ENTER
                      key.',00AH,00DH,0FFH
HELLO_TEXT2:    DB    00AH,00DH
                DB    'type any message to RAM2 then press ENTER
                      key.',00AH,00DH,0FFH
GOT_TEXT1:      DB    'I',027H,'d got message [',0FFH
GOT_TEXT2:      DB    '] from you.[RAM1]',00AH,00DH,0FFH
GOT_TEXT3:      DB    '] from you.[RAM2]',00AH,00DH,0FFH
```

<u>โปรแกรมทดสอบทำงานของไมโครคอนโทรลเลอร์ตัวที่ 2 ตามโค้ดภายในหน่วยความจำภายนอกตัวที่ 1 แบบที่ 1</u>

โปรแกรมภายในไมโครคอนโทรลเลอร์ตัวที่ 1

```
;-------------------------------------------------------------------
;PORT CONTROL
;-------------------------------------------------------------------
RESET2      BIT       P1.7
SWITCH      BIT       P1.0
CHECK       BIT       P3.2

;-------------------------------------------------------------------
; Define User Register
;-------------------------------------------------------------------
ADDHI       EQU       030H
```

```
ADDLI       EQU     031H
ADDHX       EQU     032H
ADDLX       EQU     033H

;-------------------------------------------------------------------
; Main Program.
;-------------------------------------------------------------------
ORG         0000H
MAIN:       MOV     TMOD,#021H
            MOV     TH1,#0FDH
            MOV     TL1,#0FDH
            SETB    TR1
            MOV     SCON,#040H
            SETB    REN
START:      CLR     SWITCH
            SETB    RESET2
            MOV     DPTR,#SERIAL_TEXT
            ACALL   TX_TEXT

LOOP:       JNB     RI,$
            CLR     RI
            MOV     A,SBUF
            CJNE    A,#00DH,LOOP
            MOV     ADDHX,#00H
            MOV     ADDLX,#00H
            MOV     DPTR,#PRO
            ACALL   LOAD_PRO
            SETB    SWITCH
            CLR     RESET2
            JB      CHECK,$
            CLR     SWITCH
            SETB    RESET2
```

```
                MOV     DPTR,#2000H
REPLY_LOOP:     CLR     TI
                MOVX    A,@DPTR
                MOV     SBUF,A
                JNB     TI,$
                CLR     TI
                INC     DPTR
                CJNE    A,#'Z',REPLY_LOOP
                SJMP    START

;---------------------------------------------------------------
; TX Serial Text from ROM Pointer
;---------------------------------------------------------------
TX_TEXT:    CLR     TI
TX_LOOP:    CLR     A
            MOVC    A,@A+DPTR
            INC     DPTR
            CJNE    A,#0FFH,TX_CHAR.
            RET

TX_CHAR:    MOV     SBUF,A
            JNB     TI,$
            CLR     TI
            AJMP    TX_LOOP

;---------------------------------------------------------------
; LOAD PROGRAM TO RAM1
;---------------------------------------------------------------
LOAD_PRO:   CLR     A
            MOVC    A,@A+DPTR
            INC     DPTR
```

```
                MOV         ADDHI,DPH
                MOV         ADDLI,DPL
                CJNE        A,#0AAH,LOAD_PRO1
                RET

LOAD_PRO1:      MOV         DPH,ADDHX
                MOV         DPL,ADDLX
                MOVX        @DPTR,A
                INC         DPTR
                MOV         ADDHX,DPH
                MOV         ADDLX,DPL
                MOV         DPH,ADDHI
                MOV         DPL,ADDLI
                SJMP        LOAD_PRO

;-------------------------------------------------------------
;Define Constant < Store in Flash EEPROM Program Memory >
;-------------------------------------------------------------
SERIAL_TEXT:    DB          00AH,00DH
                DB          00AH,00DH
                DB          'Hello..INS_24.Please press ENTER key.',00AH,00DH
                DB          00AH,00DH,0FFH
PRO:            DB          74H,41H,90H,00H,00H,0F0H,0A3H,04H,0B4H,05AH
                DB          0FAH,0F0H,0C2H,0B5H,0AAH
```

โปรแกรมที่ให้ไมโครคอนโทรลเลอร์ตัวที่ 2 ทำงาน

```
        ORG         0000H           ; Reset Vector
        MOV         A,#'A'          ; 74  41
        MOV         DPTR,#0000H     ; 90  00  00
LOOP:   MOVX        @DPTR,A         ; F0
        INC         DPTR            ; A3
        INC         A               ; 04
```

```
        CJNE    A,#'Z',LOOP     ; B4  5A  FA
        MOVX    @DPTR,A         ; F0
        CLR     P3.5            ; C2  B5
```

<u>โปรแกรมทดสอบทำงานของไมโครคอนโทรลเลอร์ตัวที่ 2 ตามโค้ดภายในหน่วยความจำภายนอกตัวที่ 1 แบบที่ 2</u>

โปรแกรมภายในไมโครคอนโทรลเลอร์ตัวที่ 1

```
;-------------------------------------------------------------
;PORT CONTROL
;-------------------------------------------------------------
RESET2      BIT     P1.7
SWITCH      BIT     P1.0
CHECK       BIT     P3.2

;-------------------------------------------------------------
; Define User Register
;-------------------------------------------------------------
ADDHI       EQU     030H
ADDLI       EQU     031H
ADDHX       EQU     032H
ADDLX       EQU     033H

;-------------------------------------------------------------
; Main Program.
;-------------------------------------------------------------
            ORG     0000H
MAIN:       MOV     TMOD,#021H
            MOV     TH1,#0FDH
            MOV     TL1,#0FDH
            SETB    TR1
```

```
        MOV     SCON,#040H
        SETB    REN

START:  CLR     SWITCH
        SETB    RESET2
        MOV     DPTR,#SERIAL_TEXT
        ACALL   TX_TEXT

LOOP:   JNB     RI,$
        CLR     RI
        MOV     A,SBUF
        CJNE    A,#00DH,LOOP
        MOV     DPTR,#BEFORE
        ACALL   TX_TEXT

        MOV     A,#'A'
        MOV     DPTR,#2000H
LOOP1:  CLR     TI
        MOVX    @DPTR,A
        MOV     SBUF,A
        JNB     TI,$
        CLR     TI
        INC     DPTR
        INC     A
        CJNE    A,#'Z',LOOP1
        CLR     TI
        MOVX    @DPTR,A
        MOV     SBUF,A
        JNB     TI,$
        CLR     TI
```

```
        MOV     ADDHX,#00H
        MOV     ADDLX,#00H
        MOV     DPTR,#PRO
        ACALL   LOAD_PRO
        SETB    SWITCH
        CLR     RESET2
        JB      CHECK,$

        CLR     SWITCH
        SETB    RESET2
        MOV     DPTR,#AFTER ; Set Pointer Serial TX
        ACALL   TX_TEXT

        MOV     DPTR,#2000H
REPLY_LOOP:
        CLR     TI
        MOVX    A,@DPTR
        MOV     SBUF,A
        JNB     TI,$
        CLR     TI
        INC     DPTR
        CJNE    A,#'A',REPLY_LOOP
        SJMP    START

;---------------------------------------------------------------------
; TX Serial Text from ROM Pointer
;---------------------------------------------------------------------
TX_TEXT:    CLR     TI
TX_LOOP:    CLR     A
            MOVC    A,@A+DPTR
            INC     DPTR
            CJNE    A,#0FFH,TX_CHAR.
            RET
```

```
TX_CHAR:    MOV     SBUF,A
            JNB     TI,$
            CLR     TI
            AJMP    TX_LOOP

;---------------------------------------------------------------------
; LOAD PROGRAM TO RAM1
;---------------------------------------------------------------------
LOAD_PRO:   CLR     A
            MOVC    A,@A+DPTR
            INC     DPTR
            MOV     ADDHI,DPH
            MOV     ADDLI,DPL
            CJNE    A,#0AAH,LOAD_PRO1
            RET

LOAD_PRO1:  MOV     DPH,ADDHX
            MOV     DPL,ADDLX
            MOVX    @DPTR,A
            INC     DPTR
            MOV     ADDHX,DPH
            MOV     ADDLX,DPL
            MOV     DPH,ADDHI
            MOV     DPL,ADDLI
            SJMP    LOAD_PRO

;---------------------------------------------------------------------
;Define Constant < Store in Flash EEPROM Program Memory >
;---------------------------------------------------------------------
SERIAL_TEXT:    DB  00AH,00DH
                DB  00AH,00DH
                DB  'Hello..INS_24.Please press ENTER key.',00AH,00DH
```

```
                DB      00AH,00DH,0FFH
BEFORE:         DB      00AH,00DH
                DB      'DATA BEFORE EXCHANGE :  ',0FFH
AFTER:          DB      00AH,00DH
                DB      'DATA AFTER EXCHANGE :  ',0FFH
PRO:            DB      90H,20H,00H,78H,30H,0E0H,0F6H,0A3H,08H,0B4H
                DB      05AH,0F9H,18H,79H,30H,07EH,05H,07FH,03H,0E7H
                DB      0C6H,0F7H,09H,18H,0DFH,0F9H,09H,18H,0DEH,0F3H
                DB      90H,20H,00H,78H,30H,0E6H,0F0H,08H,0A3H,0B4H
                DB      41H,0F9H,0C2H,0B5H,0AAH
```

โปรแกรมที่ให้ไมโครคอนโทรลเลอร์ตัวที่ 2 ทำงาน

```
                org     0000h       ; Reset Vector
                mov     dptr,#2000h ; 90  20  00
                mov     r0,#30h     ; 78  30
loop:           movx    a,@dptr     ; E0
                mov     @r0,a       ; F6
                inc     dptr        ; A3
                inc     r0          ; 08
                cjne    a,#'Z',loop ; B4  5A  F9
                dec     r0          ; 18

                mov     r1,#30h     ; 79  30
                mov     r6,#05h     ; 7E  05
loop2:          mov     r7,#03h     ; 7F  03
loop1:          mov     a,@r1       ; E7
                xch     a,@r0       ; C6
                mov     @r1,a       ; F7
                inc     r1          ; 09
                dec     r0          ; 18
                djnz    r7,loop1    ; DF  F9
                inc     r1          ; 09
```

```
                dec         r0          ; 18
                djnz        r6,loop2    ; DE F3

                mov         dptr,#2000h ; 90 20 00
                mov         r0,#30h     ; 78 30
loop3:          mov         a,@r0       ; E6
                movx        @dptr,a     ; F0
                inc         r0          ; 08
                inc         dptr        ; A3
                cjne        a,#'A',loop3 ; B4 41 F9
                clr         p3.5        ; C2 B5
```

<u>โปรแกรมทดสอบการทำงาน PORT 1 ของไมโครคอนโทรลเลอร์ตัวที่ 2</u>

โปรแกรมภายในไมโครคอนโทรลเลอร์ตัวที่ 1

```
;-------------------------------------------------------------------
;PORT CONTROL
;-------------------------------------------------------------------
RESET2          BIT         P1.7
SWITCH          BIT         P1.0
CHECK           BIT         P3.2

;-------------------------------------------------------------------
; Define User Register
;-------------------------------------------------------------------
ADDHI           EQU         030H
ADDLI           EQU         031H
ADDHX           EQU         032H
ADDLX           EQU         033H
```

```
;-------------------------------------------------------------------------
; Main Program.
;-------------------------------------------------------------------------

        ORG     0000H
MAIN:   MOV     TMOD,#021H
        MOV     TH1,#0FDH
        MOV     TL1,#0FDH
        SETB    TR1
        MOV     SCON,#040H
        SETB    REN
START:  CLR     SWITCH
        SETB    RESET2
        MOV     DPTR,#SERIAL_TEXT
        ACALL   TX_TEXT

LOOP:   JNB     RI,$
        CLR     RI
        MOV     A,SBUF
        CJNE    A,#00DH,LOOP

        MOV     ADDHX,#00H
        MOV     ADDLX,#00H
        MOV     DPTR,#PRO
        ACALL   LOAD_PRO
        SETB    SWITCH
        CLR     RESET2
        JB      CHECK,$

        CLR     SWITCH
        SETB    RESET2
```

```
;-------------------------------------------------------------------
; TX Serial Text from ROM Pointer
;-------------------------------------------------------------------
TX_TEXT:    CLR         TI
TX_LOOP:    CLR         A
            MOVC        A,@A+DPTR
            INC         DPTR
            CJNE        A,#0FFH,TX_CHAR.
            RET
TX_CHAR:    MOV         SBUF,A
            JNB         TI,$
            CLR         TI
            AJMP        TX_LOOP

;-------------------------------------------------------------------
; LOAD PROGRAM TO RAM1
;-------------------------------------------------------------------
LOAD_PRO:   CLR         A
            MOVC        A,@A+DPTR
            INC         DPTR
            MOV         ADDHI,DPH
            MOV         ADDLI,DPL
            CJNE        A,#0AAH,LOAD_PRO1
            RET

LOAD_PRO1:  MOV         DPH,ADDHX
            MOV         DPL,ADDLX
            MOVX        @DPTR,A
            INC         DPTR
            MOV         ADDHX,DPH
            MOV         ADDLX,DPL
            MOV         DPH,ADDHI
```

```
                MOV         DPL,ADDLI
                SJMP        LOAD_PRO

;-------------------------------------------------------------------
;Define Constant < Store in Flash EEPROM Program Memory >
;-------------------------------------------------------------------
SERIAL_TEXT:    DB      00AH,00DH
                DB      00AH,00DH
                DB      'Hello..INS_24.Please press ENTER key.',00AH,00DH
                DB      00AH,00DH,0FFH
PRO:            DB      74H,01H,0F5H,90H,11H,09H,23H,80H,0F9H,7DH
                DB      0AH,7FH,64H,7EH,0E6H,00H,00H,0DEH,0FCH,0DFH
                DB      0F8H,0DDH,0F4H,22H,0AAH
```

โปรแกรมที่ให้ไมโครคอนโทรลเลอร์ตัวที่ 2 ทำงาน

```
                ORG         0000H               ; Reset Vector
                MOV         A,#01H              ; 74  01
LOOP:           MOV         P1,A                ; F5  90
                ACALL       DELAY_1s            ; 11  09
                RL          A                   ; 23
                SJMP        LOOP                ; 80  F9

;-------------------------------------------------------------------
; Dummy Delay time 1S
;-------------------------------------------------------------------
DELAY_1s:       MOV         R5,#10              ; 7D  0A
DELAY_100ms:    MOV         R7,#100             ; 7F  64
DELAY_100ms_1:  MOV         R6,#0E6H            ; 7E  E6
DELAY_100ms_2:  NOP                             ; 00
                NOP                             ; 00
                DJNZ        R6,DELAY_100ms_2    ; DE  FC
                DJNZ        R7,DELAY_100ms_1    ; DF  F8
```

```
DJNZ        R5,DELAY_100ms    ; DD F4
RET                           ; 22
```

<u>โปรแกรมทดสอบการทำงาน PORT 1 ของไมโครคอนโทรลเลอร์ตัวที่ 2 แบบ STEP</u>

โปรแกรมภายในไมโครคอนโทรลเลอร์ตัวที่ 1

```
;-------------------------------------------------------------
;PORT CONTROL
;-------------------------------------------------------------
RESET2      BIT       P1.7
SWITCH      BIT       P1.0
CHECK       BIT       P3.2

;-------------------------------------------------------------
; Define User Register
;-------------------------------------------------------------
ADDHI       EQU       030H
ADDLI       EQU       031H
ADDHX       EQU       032H
ADDLX       EQU       033H

;-------------------------------------------------------------
; Main Program.
;-------------------------------------------------------------
            ORG       0000H
MAIN:       MOV       TMOD,#021H
            MOV       TH1,#0FDH
            MOV       TL1,#0FDH
            SETB      TR1
            MOV       SCON,#040H
            SETB      REN
```

```
        CLR     SWITCH
        SETB    RESET2
        CLR     INT0
        MOV     DPTR,#SERIAL_TEXT
        ACALL   TX_TEXT
LOOP:   JNB     RI,$
        CLR     RI
        MOV     A,SBUF
        CJNE    A,#00DH,LOOP
        MOV     ADDHX,#00H
        MOV     ADDLX,#00H
        MOV     DPTR,#PRO1
        ACALL   LOAD_PRO

        MOV     ADDHX,#00H
        MOV     ADDLX,#30H
        MOV     DPTR,#PRO2
        ACALL   LOAD_PRO
        SETB    SWITCH
        CLR     RESET2
START:  CLR     INT0
        JB      CHECK,$
        SETB    INT0
        MOV     DPTR,#SERIAL_TEXT
        ACALL   TX_TEXT
LOOP1:  JNB     RI,$
        CLR     RI
        MOV     A,SBUF
        CJNE    A,#00DH,LOOP1
        SJMP    START
```

```
;-------------------------------------------------------------------
; TX Serial Text from ROM Pointer
;-------------------------------------------------------------------
TX_TEXT:    CLR         TI
TX_LOOP:    CLR         A
            MOVC        A,@A+DPTR
            INC         DPTR
            CJNE        A,#0FFH,TX_CHAR.
            RET
TX_CHAR:    MOV         SBUF,A
            JNB         TI,$
            CLR         TI
            AJMP        TX_LOOP

;-------------------------------------------------------------------
; LOAD PROGRAM TO RAM1
;-------------------------------------------------------------------
LOAD_PRO:   CLR         A
            MOVC        A,@A+DPTR
            INC         DPTR
            MOV         ADDHI,DPH
            MOV         ADDLI,DPL
            CJNE        A,#0AAH,LOAD_PRO1
            RET

LOAD_PRO1:  MOV         DPH,ADDHX
            MOV         DPL,ADDLX
            MOVX        @DPTR,A
            INC         DPTR
            MOV         ADDHX,DPH
            MOV         ADDLX,DPL
            MOV         DPH,ADDHI
```

```
                MOV         DPL,ADDLI
                SJMP        LOAD_PRO

;---------------------------------------------------------------------
;Define Constant < Store in Flash EEPROM Program Memory >
;---------------------------------------------------------------------
SERIAL_TEXT:    DB      00AH,00DH
                DB      00AH,00DH
                DB      'Hello..INS_24.Please press ENTER key.',00AH,00DH
                DB      00AH,00DH,0FFH
PRO1:           DB      11H,30H,00H,0C2H,0B5H,30H,0B2H,0FDH,0D2H,0B5H
                DB      20H,0B2H,0FDH,32H,0AAH
PRO2:           DB      75H,0A8H,81H,0D2H,88H,74H,01H,0F5H,90H,23H
                DB      80H,0FBH,0AAH
```

โปรแกรมที่ให้ไมโครคอนโทรลเลอร์ตัวที่ 2 ทำงาน

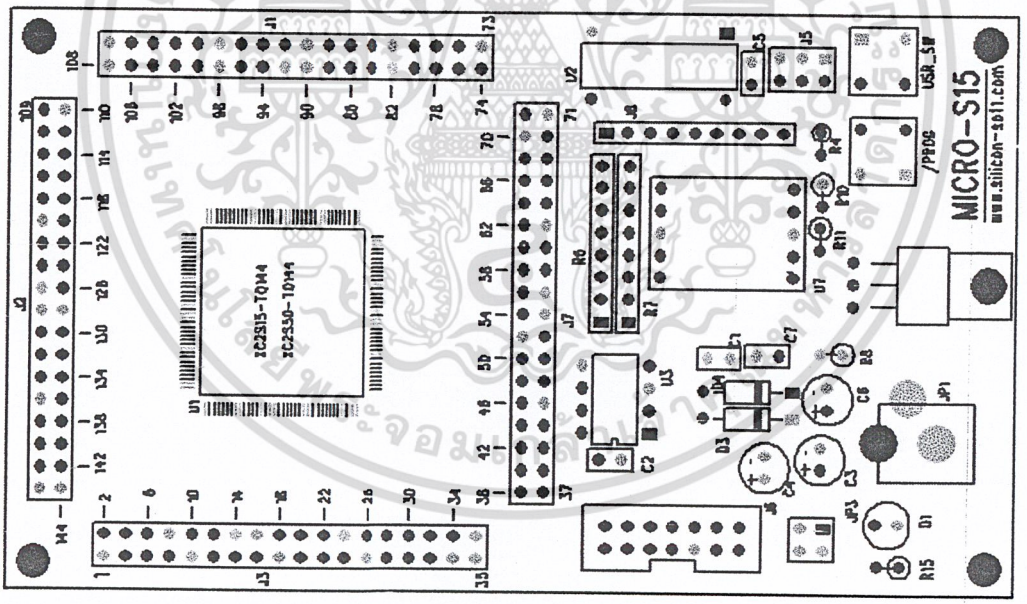
```
                ORG     0000H       ; Reset Vector
                ACALL   MAIN        ; 11  30

                ORG     0003H       ; Interrupt Vector
                CLR     P3.5        ; C2  B5
                JNB     P3.2,$      ; 30  B2  FD
                SETB    P3.5        ; D2  B5
                JB      P3.2,$      ; 20  B2  FD
                RETI                ; 32

                ORG     0030H       ; Start Vector
MAIN:           MOV     IE,#81H     ; 75  A8  81
                SETB    IT0         ; D2  88

                MOV     A,#01H      ; 74  01
LOOP:           MOV     P1,A        ; F5  90
```

```
RL      A       ; 23
SJMP    LOOP    ; 80 FB
```

MICRO-S15
FPGA Development Kit
User's Manual/คู่มือการใช้งาน
Spartan-II
XC2S15

บทนำ

บอร์ด MICRO-S15 เป็นบอร์ดพัฒนาสำหรับใช้ในการออกแบบวงจรดิจิตอล แบบ High Level Design ที่ใช้ภาษา ABEL หรือ HDL (VHDL, Verilog HDL) ในการออกแบบ ซึ่งใช้งานร่วมกับชุดโปรแกรม Xilinx ISE *Web*PACK ซึ่งเป็นฟรีโปรแกรม (Free software) ของบริษัท Xilinx ที่ท่านสามารถดาวน์โหลดได้ที่ http://www.xilinx.com/webpack/index.htm (หรือจากแผ่นซีดีรอม Silicon-Soft FPGA Tools & Literature Release 1.0) ชุดโปรแกรมนี้มีโปรแกรมย่อยมากมายสำหรับให้ผู้ใช้ได้ใช้ในการออกแบบวงจร เช่น โปรแกรม Xilinx ECS ซึ่งเป็นโปรแกรมช่วยในการวาดแผนภาพเค้าร่างของ โปรแกรม Xilinx XST ซึ่งช่วยในการสังเคราะห์ (Synthesis) โมเดลวงจรจากภาษา ABEL หรือ HDL ให้อยู่ในรูปของแผนภาพวงจร (Schematic) รวมทั้งโปรแกรมที่ช่วยในการคอนฟิกกูเรชั่น (Configuration) ชิปได้ทั้งตระกูล CPLD (Complex Programmable Logic Device) และ FPGA (Field Programmable Gate Array) โดยผ่านสายดาวน์โหลด SP-JP-CA01 (ซึ่งเป็นสายดาวน์โหลดโปรแกรมที่ออกแบบลงสู่ชิปเอฟพีจีเอที่ทางบริษัทได้พัฒนาขึ้น และแถมมากับชุดพัฒนาดังกล่าว) โดยโปรแกรมสามารถรันบนวินโดวส์ 9X/2000/ME/XP ซึ่งทั้งหมดนี้อยู่ภายในชุดโปรแกรม Xilinx ISE *Web*PACK (รายละเอียดและวิธีการใช้งานสามารถดูได้จาก หัวข้อ "การใช้งานชุดโปรแกรม Xilinx ISE *Web*PACK")

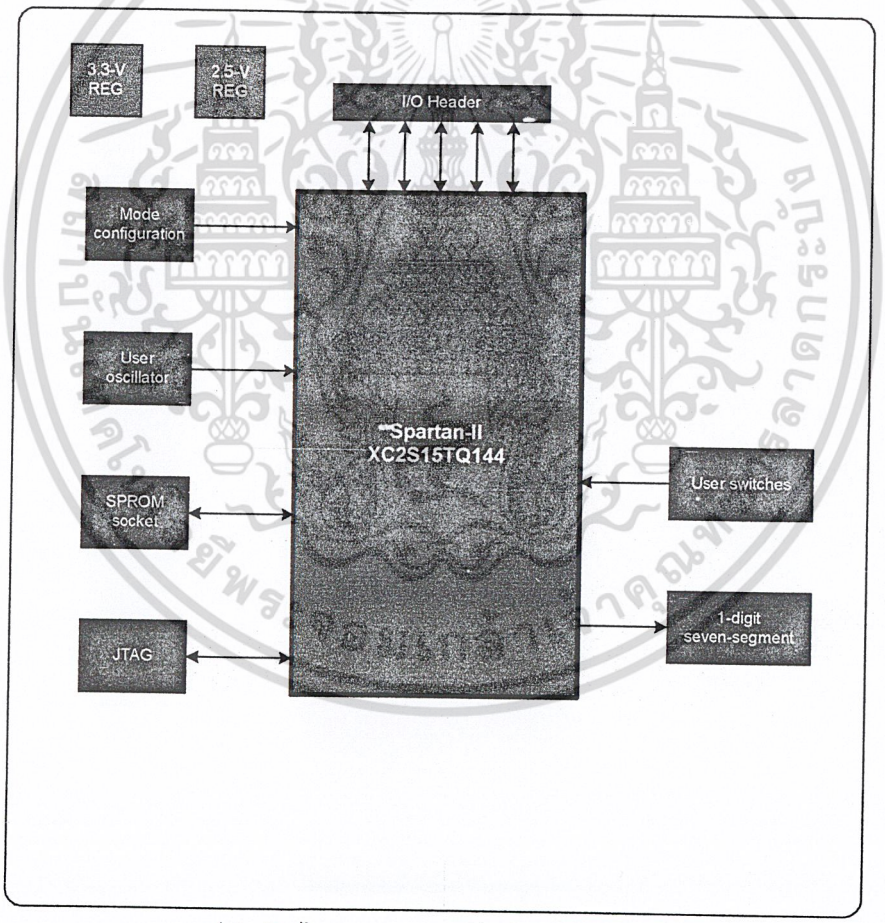


รูป 1 บอร์ด **MICRO-S15**

Silicon-soft Co., Ltd., Software Park Building, 7th Floor Unit I,
92/28 Chaeng Wattana Road, Klong Gluea, Pakkred, Nonthaburi 11120, Thailand.
www.silicon-soft.net /www.silicon-soft.com E-mail: **sale@silicon-soft.net**

ทำความรู้จักกับบอร์ด MICRO-S15

บอร์ด MICRO-S15 ประกอบไปด้วยชิปเอฟพีจีเอ (FPGA: Field Programmable Gate Array) ของบริษัท Xilinx ตระกูล Spartan II เบอร์ XC2S15-5TQ144 ขนาด 15,000 เกท พร้อมกับชุดไฟเลี้ยงใช้ได้ตั้งแต่ +6 ถึง +9 โวลต์ ต่อผ่านทางแจ็คอะแดปเตอร์ มีวงจรเรกูเลเตอร์เพื่อสร้างไฟเลี้ยงคงที่ ที่ 2.5 โวลต์ เพื่อนำไปเป็นไฟเลี้ยงให้กับคอร์แกนกลางภายในชิปเอฟพีจีเอ, 3.3 โวลต์นำไปเป็นไฟเลี้ยงในกับ SPROM (Serial PROM) และพอร์ต JTAG (Joint Test Action Group) บนบอร์ดประกอบด้วย 7-SEGMENTS, LED, SWITCHS โดยความถี่สัญญาณนาฬิกาหลัก 3.57945 MHz (สามารถเปลี่ยนได้สูงสุดถึง 200 MHz ขึ้นอยู่กับวงจรที่ออกแบบ) และยังสามารถเลือกโปรแกรมชิปจากโหมดคอนฟิกกูเรชั่นได้ทั้งโปรแกรมผ่านพอร์ต JTAG ในกรณีทดสอบวงจรหรือโปรแกรมผ่านทาง SPROM เมื่อไม่ต้องการดาวน์โหลดทุกครั้งที่ใช้งาน



รูป 2 บล็อกไดอะแกรมของบอร์ด MICRO-S15

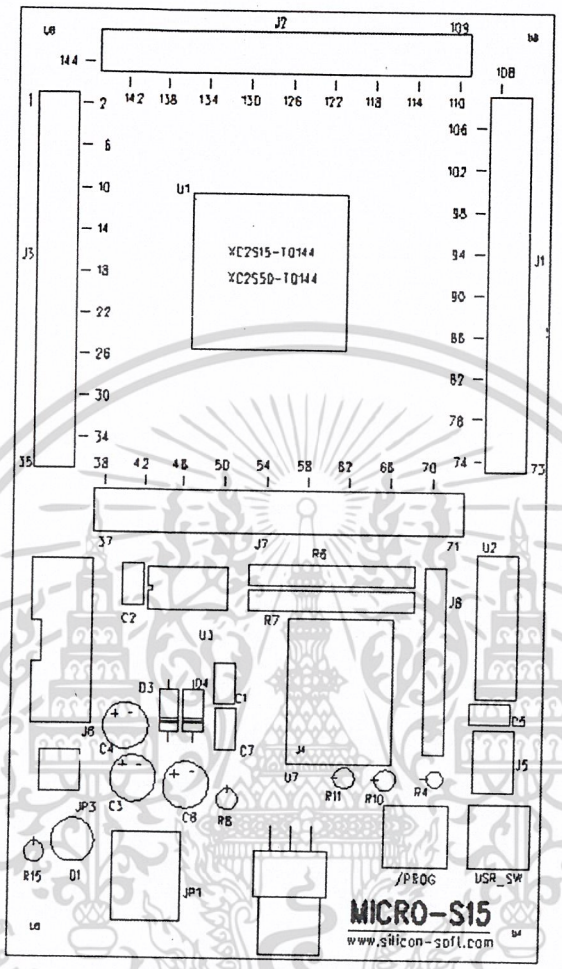
Silicon-soft Co., Ltd., Software Park Building, 7th Floor Unit I,
92/28 Chaeng Wattana Road, Klong Gluea, Pakkred, Nonthaburi 11120, Thailand.
www.silicon-soft.net /www.silicon-soft.com E-mail: **sale@silicon-soft.net**

คุณสมบัติทางเทคนิค

- บอร์ดบรรจุชิปเอฟพีจีเอตระกูล Xilinx Spartan เบอร์ XC2S15-5TQ144 มีขนาดความจุ 15,000 เกท
- มี I/O ให้เลือกใช้ถึง 86 pin กำหนดเป็น in, out หรือ Bi-directional ได้
- RAM ภายใน Chip ขนาด 16 Kbit กำหนดให้เป็น Single-port หรือ Dual-port ได้
- ภายใน FPGA มีวงจร DLL (Delay Lock Loop) จำนวน 4 วงจร ทำหน้าที่ลด clock skew สำหรับวงจรความถี่สูง สามารถคูณความถี่ได้มากกว่า 2 เท่า, สามารถหารความถี่ด้วย 1.5, 2, 2.5, 3, 4, 5, 8, 16 และสามารถสร้างสัญญาณที่มีเฟสชิปจากสัญญาณอ้างอิง 90, 180, 270 องศา ได้
- สนับสนุนการเชื่อมต่อแบบ LVTTL(3.3V), LVCMOS2(2.5V) , PCI(3V/5V, 33MHz/66MHz)
- ต่อใช้งานกับ TTL IC ได้โดยตรง
- บนบอร์ดสามารถเลือกระดับสัญญาณของ I/O ได้ 2 ระดับ 2.5V และ 3.3V หรือจะต่อกับแรงดันอ้างอิงภายนอกได้
- ซอกเกตสำหรับ Oscillator chip ทำหน้าที่เป็นสัญญาณนาฬิกาหลักสำหรับวงจร
- 7 segment 1 digit สำหรับใช้งานแสดงผล
- 1 Push Switch สำหรับใช้งานเป็น input switch
- พอร์ต JTAG สำหรับเชื่อมต่อกับสายดาวน์โหลด (JTAG Cable) เพื่อโปรแกรมชิปเอฟพีจีเอในโหมด Boundary –Scan ซึ่งเป็นมาตรฐาน IEEE 1149.1 Test Access Port (TAP)

Silicon-soft Co., Ltd., Software Park Building, 7th Floor Unit I,
92/28 Chaeng Wattana Road, Klong Gluea, Pakkred, Nonthaburi 11120, Thailand.
www.silicon-soft.net /www.silicon-soft.com E-mail: **sale@silicon-soft.net**

รายละเอียดการใช้งาน



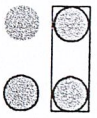
- DC JACK (JP1) configuration

- **Vcco Setting (JP3)**

จั๊มเปอร์ JP3 ใช้สำหรับการเลือกไฟเลี้ยงขาเอฟพีจีเอและอุปกรณ์เอาต์พุตต่างๆ โดยสามารถเลือกไฟเลี้ยงได้ 2 แบบคือ 2.5V (LVCMOS2) และ 3.3V (LVTTL 2-24 mA)

2.5 Volt Setting

Silicon-soft Co., Ltd., Software Park Building, 7th Floor Unit I,
92/28 Chaeng Wattana Road, Klong Gluea, Pakkred, Nonthaburi 11120, Thailand.
www.silicon-soft.net /www.silicon-soft.com E-mail: **sale@silicon-soft.net**

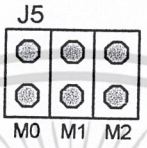


3.3 Volt Setting

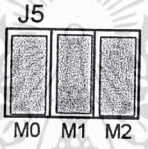
- **Mode Configuration**

โหมดการโปรแกรมชิป โดยที่เลือกโหมดจากขา M0, M1, M2 โดยที่ M0, M1, M2 ถูกพูลอัพ (Pull-Up) ทำให้ค่าที่อ่านได้จะมีค่าเป็น '1' เสมอ

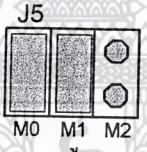
- ปลดจั้มเปอร์ (Open Jumpers) ออก จะได้ค่าเป็น M0, M1, M2 = 1, 1, 1 (Defaults)



- สวมจั้มเปอร์ (Close Jumpers) ลงไป จะได้ค่าเป็น M0, M1, M2 = 0, 0, 0



- ถ้าต้องการโปรแ กรมโดยใช้ SPROM ให้เลือกโหมด Master Serial mode โดยให้เซ็ตจั้มเปอร์ ดังนี้ M0=0, M1=0, M2=1



- ถ้าต้องการโปรแกรมผ่านสาย JTAG จะไม่ขึ้นอยู่กับโหมด ซึ่งไม่ว่าจะ เซ็ตจั้มเปอร์โหมดใดๆ ก็สามารถโปรแกรมได้ (Independent of the mode selection)

- รายละเอียดของ mode ต่าง ๆ ในการคอนฟิกกูเรชั่น สามารถศึกษาได้จาก Spartan data Sheet ของ Xilinx หรือที่เว็ป http://www.Xilinx.com

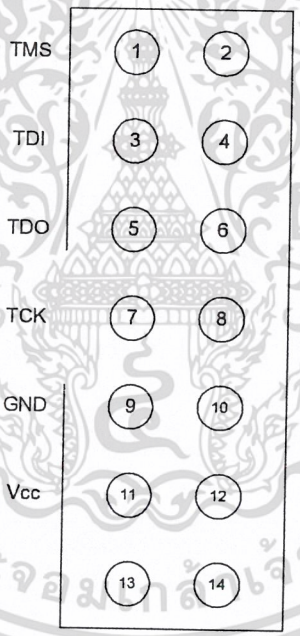
- **User Oscillator (U2)**

บนบอร์ด MICRO-S15 มีออสซิเลเตอร์ขนาด 6 MHz ออกแบบเป็น socket สามารถถอดเปลี่ยนได้ โดยออสซิเลเตอร์ถูกต่อเข้ากับขา P88 (GCK0) ของชิปเอฟพีจีเอบนบอร์ด

- **JTAG Connector (JTAG)**

บนบอร์ด MICRO-S15 มีพอร์ต JTAG สำหรับใช้เชื่อมต่อกับ สายดาวน์โหลด SP-JP-CA01 ซึ่งเป็นสาย JTAG เพื่อคอนฟิกกูเรชั่นวงจรที่ออกแบบลงสู่ชิปเอฟพีจีเอ อ่านรายละเอียดได้จาก หัวข้อ "*การดาวน์โหลดวงจรโดยใช้โปรแกรม iMPACT*"

Pin on JTAG Connector	JTAG Pin	Fuction
1	TMS	Test Mode Select – this signal is decoded by the JTAG state machine to control test operations.
3	TDI	Test Data Input – this signal is used to transmit serial test instructions and data.
5	TDO	Test Data Output – data from the target system is read at this pin.
7	TCK	Test Clock – Drives the test logic for all devices on a JTAG chain.
9, 13	GND	Ground – Supplies ground reference to the cable.
11	Vcc	Power – Supplies VCC (5 V, 10 mA, typically) to the cable.
2, 4, 6, 8,10, 12, 14	NC	Not connected



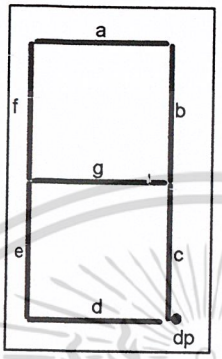
- **SPROM Socket (U3)**

  บนบอร์ด MICRO-S15 มีซ็อกเก็ตแบบ DIP ขนาด 8 พินเตรียมไว้สำหรับให้ผู้ใช้สำหรับโปรแกรมชิปโดยใช้ SPROM เมื่อไม่ต้องการดาวน์โหลดโปรแกรมทุกครั้งที่ใช้งานบอร์ด โดยเลือกโหมดการโปรแกรมเป็นแบบ Master Serial mode (M0=0, M1=0, M2=1) โดยเมื่อเสียบอะแด็ปเตอร์จะทำการดาวน์โหลดอัตโนมัติทันที โดยพิน OE/Reset ของ SPROM จะถูกเชื่อมต่อกับ Ground

Silicon-soft Co., Ltd., Software Park Building, 7th Floor Unit I,
92/28 Chaeng Wattana Road, Klong Gluea, Pakkred, Nonthaburi 11120, Thailand.
www.silicon-soft.net /www.silicon-soft.com E-mail: **sale@silicon-soft.net**

- **Seven-segment (U7)**

  บนบอร์ด MICRO-S15 มีส่วนแสดงผลเอาท์พุตบน 7-Segment หนึ่งหลัก โดยแต่ละ Segment จะติดสว่างจะต้องป้อนลอจิก '1' โดยรายละเอียดของตำแหน่งต่างๆ ของ 7-Segment และการเชื่อมต่อกับขาของเอฟพีจีเอ ดังแสดงในรูปและตารางข้างล่าง



ตารางแสดงตำแหน่ง Segments ที่ถูกเชื่อมต่อกับขาของเอฟพีจีเอบนบอร์ด

7-segment pin	FPGA pin
A	P63
B	P66
C	P62
D	P59
E	P57
F	P60
G	P58
Dp	P65

- **USER-Switch (USR_SW)**

  บนบอร์ด MICRO-S15 มี Switch ถูกพูลอัป (Pull-up) ถ้ากดจะได้ลอจิก '0' ปล่อยจะได้ลอจิก '1' ให้ผู้ใช้สามารถเลือกใช้งานได้ โดยมีรายละเอียดตำแหน่งของสวิตช์ที่เชื่อมต่อกับขาเอฟพีจีเอดังนี้

USER Switch	FPGA pin
USR_SW	P67

Silicon-soft Co., Ltd., Software Park Building, 7th Floor Unit I,
92/28 Chaeng Wattana Road, Klong Gluea, Pakkred, Nonthaburi 11120, Thailand.
www.silicon-soft.net /www.silicon-soft.com E-mail: **sale@silicon-soft.net**

❑ I/O Header (HD1–HD4)

ทุกๆ ขาของเอฟพีจีเอจะถูกเชื่อมต่อกับคอนเน็คเตอร์ J1 ถึง J4 ตามลำดับ โดยมีรายละเอียดดังตารางต่อไปนี้

FPGA pin	Function	J3
P1	Vcco	1
P2	TCK	2
P3	I/O	3
P4	I/O	4
P5	I/O, Vref	5
P6	I/O	6
P7	I/O	7
P8	GND	8
P9	Vccint	9
P10	I/O	10
P11	I/O	11
P12	I/O, Vref	12
P13	I/O	13
P14	Vccint	14
P15	I, GCK3	15
P16	Vcco	16
P17	GND	17
P18	I, GCK2	18
P19	I/O	19
P20	I/O	20
P21	I/O, Vref	21
P22	I/O	22
P23	I/O	23
P24	Vccint	24
P25	GND	25
P26	I/O	26
P27	I/O	27
P28	I/O, Vref	28
P29	I/O	29
P30	I/O ($\overline{\text{WRITE}}$)	30
P31	I/O ($\overline{\text{CS}}$)	31
P32	TDI	32
P33	GND	33
P34	TDO	34
P35	Vcco	35
P36	Vcco	36

Silicon-soft Co., Ltd., Software Park Building, 7th Floor Unit I,
92/28 Chaeng Wattana Road, Klong Gluea, Pakkred, Nonthaburi 11120, Thailand.
www.silicon-soft.net /www.silicon-soft.com E-mail: **sale@silicon-soft.net**

FPGA pin	Function	J7 pin
P37	CCLK	1
P38	I/O (DOUT, BUSY)	2
P39	I/O (DIN, D0)	3
P40	I/O	4
P41	I/O, Vref	5
P42	-	6
P43	I/O	7
P44	I/O (D1)	8
P45	GND	9
P46	I/O (D2)	10
P47	I/O	11
P48	I/O, Vref	12
P49	I/O (D3)	13
P50	I/O	14
P51	I/O, IRDY	15
P52	GND	16
P53	Vcco	17
P54	I/O, TRDY	18
P55	Vccint	19
P56	I/O	20
P57	I/O (D4)	21
P58	I/O, Vref	22
P59	I/O	23
P60	I/O (D5)	24
P61	GND	25
P62	I/O (D6)	26
P63	I/O	27
P64	-	28
P65	I/O, Vref	29
P66	I/O	30
P67	I/O (D7)	31
P68	I/O ($\overline{\text{INIT}}$)	32
P69	$\overline{\text{PROG}}$	33
P70	Vcco	34
P71	Vcco	35
P72	DONE	36

Silicon-soft Co., Ltd., Software Park Building, 7th Floor Unit I,
92/28 Chaeng Wattana Road, Klong Gluea, Pakkred, Nonthaburi 11120, Thailand.
www.silicon-soft.net /www.silicon-soft.com E-mail: **sale@silicon-soft.net**

FPGA pin	Function	J1 pin
P73	GND	1
P74	I/O	2
P75	I/O	3
P76	I/O	4
P77	I/O, Vref	5
P78	-	6
P79	I/O	7
P80	I/O	8
P81	GND	9
P82	Vccint	10
P83	I/O	11
P84	I/O	12
P85	I/O, Vref	13
P86	I/O	14
P87	I/O	15
P88	I, GCK0	16
P89	GND	17
P90	Vcco	18
P91	I, GCK1	19
P92	Vccint	20
P93	I/O	21
P94	I/O, Vref	22
P95	I/O	23
P96	I/O	24
P97	Vccint	25
P98	GND	26
P99	I/O	27
P100	I/O	28
P101	-	29
P102	I/O, Vref	30
P103	I/O	31
P104	-	32
P105	-	33
P106	M2	34
P107	Vcco	35
P108	Vcco	36

Silicon-soft Co., Ltd., Software Park Building, 7th Floor Unit I,
92/28 Chaeng Wattana Road, Klong Gluea, Pakkred, Nonthaburi 11120, Thailand.
www.silicon-soft.net /www.silicon-soft.com E-mail: **sale@silicon-soft.net**

FPGA pin	Function	J2 pin
P109	M0	1
P110	GND	2
P111	M1	3
P112	I/O	4
P113	I/O	5
P114	I/O	6
P115	I/O, Vref	7
P116	-	8
P117	I/O	9
P118	I/O	10
P119	GND	11
P120	I/O	12
P121	I/O	13
P122	I/O, Vref	14
P123	I/O	15
P124	I/O	16
P125	Vccint	17
P126	I/O, TRDY	18
P127	Vcco	19
P128	GND	20
P129	I/O, IRDY	21
P130	I/O	22
P131	I/O	23
P132	I/O, Vref	24
P133	I/O	25
P134	I/O	26
P135	GND	27
P136	I/O	28
P137	I/O	29
P138	-	30
P139	I/O, Vref	31
P140	I/O	32
P141	I/O	33
P142	TMS	34
P143	GND	35
P144	Vcco	36

สรุป

ทางทีมงาน SiliconSoft หวังเป็นอย่างยิ่งว่าผู้ใช้จะสามารถออกแบบวงจรดิจิตอลโดยใช้เอฟพีจีเอได้ง่ายอย่างที่ตั้งใจไว้

ถ้าหากมีปัญหาใดๆ ในการใช้งานบอร์ด MICRO-S15 สามารถสอบถามได้ที่ support@silicon-soft.net

Silicon-soft Co., Ltd., Software Park Building, 7th Floor Unit I,
92/28 Chaeng Wattana Road, Klong Gluea, Pakkred, Nonthaburi 11120, Thailand.
www.silicon-soft.net /www.silicon-soft.com E-mail: **sale@silicon-soft.net**

HT6264-70
CMOS 8K×8-Bit SRAM

Features

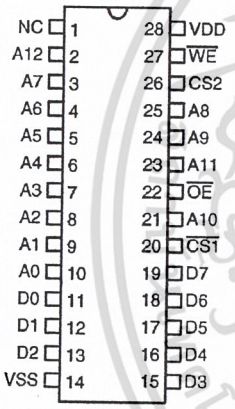
- Single 5V power supply
- Low power consumption
  - Operating: 200mW (Typ.)
  - Standby: 5μW (Typ.)
- 70ns (Max.) high speed access time
- Memory expansion by pin $\overline{OE}$
- Common I/O using tri-state outputs
- TTL compatible interface levels
- Fully static operation
- Pin-compatible with standard 8K×8 bits of EPROM/MASK ROM
- 28-pin DIP/SDIP/SOP package

General Description

The HT6264-70 is a 65,536-bit static random access memory organized as 8192 words by 8 bits and operates from a single 5-volt power supply. It is built with HOLTEK's high performance CMOS 0.8μm SPDM process.

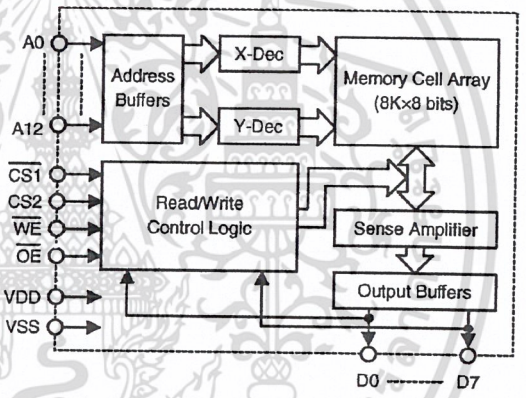
Six-transistor full COMS memory cell provides low standby current and high-reliability. Inputs and tri-state outputs are TTL compatible and allow for direct interfacing with common system bus structures.

Pin Assignment



HT6264-70
– 28 DIP/SDIP/SOP

Block Diagram



Pin Description

Pin No.	Pin Name	I/O	Description
10~3, 25, 24, 21, 23, 2	A0~A12	I	Address input pins
11~13, 15~19	D0~D2, D3~D7	I/O	Data input and output signal pins
26, 20	CS2, $\overline{CS1}$	I	Chip select signal pin
22	$\overline{OE}$	I	Output enable signal pin
27	$\overline{WE}$	I	Write enable signal pin
28	VDD	I	Positive power supply pin
14	GND	I	Negative power supply pin
1	NC		No connection

Absolute Maximum Ratings*

VDD to GND ..... −0.3V to +7.0V

IN, IN/OUT Voltage to GND, V_T ..... −0.3V to V_{CC}+0.5V

Operating Temperature, T_{opr} ..... −40°C to +85°C

Storage Temperature (Plastic), T_{stg} ..... −55°C to +125°C

Temperature Under Bias, T_{bias} ..... −10°C to +85°C

Power Dissipation, P_T 1.0W

* Note: Stresses above those listed under "Absolute Maximum Ratings" may cause permanent damage to the device. These are stress ratings only. Functional operation of this device at these or any other conditions above those indicated in the operational sections of this specification is not implied and exposure to absolute maximum rating conditions for extended periods may affect device reliability.

D.C. Characteristics

(VDD=5V±10%, GND=0V, Ta=−40°C to 85°C)

Symbol	Parameter		Min.	Typ.*	Max.	Unit
V_{DD}	Operating Voltage	—	4.5	5.0	5.5	V
I_{DD1}	Operating Current	T_{CYC}=Min. Cycle, I_{OUT}=0mA	—	—	45	mA
I_{DD2}		T_{CYC}=1μs, I_{OUT}=0mA	—	—	15	mA
I_{SB1}	Standby Current	$\overline{CS1}$=V_{IH} or CS2=V_{IL}	—	—	10	mA
I_{SB2}		$\overline{CS1}$=CS2>=V_{DD}−0.2V, V_{IN}>=V_{DD}−0.2V or V_{IN}<=0.2V	—	—	50	μA
V_{OH}	Output High Voltage	I_{OH}=−1.0mA	2.4	—	—	V
V_{OL}	Output Low Voltage	I_{OL}=4mA	—	—	0.4	V

Symbol	Parameter		Min.	Typ.*	Max.	Unit
V_{IH}	Input High Voltage	—	2.2	—	V_{DD}+0.3	V
V_{IL}	Input Low Voltage	—	–0.3	—	0.8	V
I_{LI}	Input Leakage Current	V_{DD}=5.5V, V_{IN}=GND to V_{DD}	—	—	1	μA
I_{LO}	Output Leakage Current	$\overline{CS1}$=V_{IH} or CS2=V_{IL} or $\overline{OE}$=V_{IH}, V_{OUT}=GND to V_{DD}	—	—	1	μA

*V_{DD}=5V, Ta=25°C

A.C. Characteristics

Read cycle

(V_{DD}=5V±10%, Ta=–40°C to 85°C)

Symbol	Parameter	Min.	Typ.	Max.	Unit
t_{RC}	Read Cycle Time	70	—	—	ns
t_{AA}	Address Access Time	—	—	70	ns
t_{ACS}	Chip Selection Access Time	—	—	70	ns
t_{OE}	Output Enabled to Outputs Valid	—	—	40	ns
t_{OH}	Outputs Hold from Address Change	5	—	—	ns
t_{CLZ}	Chip select to Outputs in Low-Z	10	—	—	ns
t_{OLZ}	Output Enabled to Outputs in Low-Z	10	—	—	ns
t_{CHZ}	Chip Disabled to Outputs in High-Z	0	—	30	ns
t_{OHZ}	Output Disabled to Outputs in High-Z	0	—	30	ns

Notes: 1. A read occurs during the overlap of a low $\overline{CS1}$, a high CS2, a low $\overline{OE}$, and a high $\overline{WE}$.
2. t_{CLZ} is specified from $\overline{CS1}$ or CS2 whichever occurs last.
3. t_{CHZ} is specified from $\overline{CS1}$ or CS2 whichever occurs first.
4. t_{CHZ} and t_{OHZ} are specified by the time when DATA OUT is floating

Write cycle

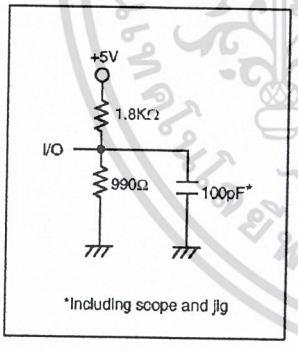
(V_{DD}=5V±10%, Ta=–40°C to 85°C)

Symbol	Parameter	Min.	Typ.	Max.	Unit
t_{WC}	Write Cycle Time	70	—	—	ns
t_{CW}	Chip Select to End of Write	35	—	—	ns
t_{AW}	Address Valid to End of Write	50	—	—	ns
t_{AS}	Address Setup Time	0	—	—	ns
t_{WP}	Write Pulse Width	25	—	—	ns
t_{WR}	Write Recovery Time	0	—	—	ns
t_{DW}	Data to Write Time Overlap	20	—	—	ns
t_{DH}	Data Hold from Write Time	0	—	—	ns
t_{OW}	Outputs Active from End of Write	5	—	—	ns
t_{OHZ}	Outputs Disable to Outputs in High-Z	0	—	40	ns
t_{WHZ}	Write to Outputs in High-Z	0	—	50	ns

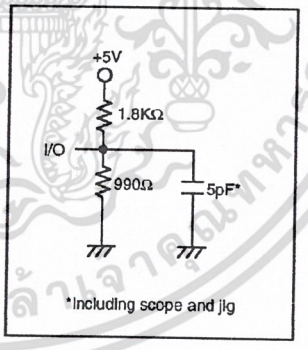
Notes: 1. A write cycle occurs during the overlap of a low $\overline{CS1}$, a high CS2, and a low $\overline{WE}$.
2. $\overline{OE}$ may be both high and low in a write cycle.
3. t_{AS} is specified from $\overline{CS1}$, CS2, or $\overline{WE}$, whichever occurs last.
4. t_{WP} is an overlap time of a low $\overline{CS1}$, a high CS2, and a low $\overline{WE}$.
5. t_{WR}, t_{DW} and t_{DH} is specified from $\overline{CS1}$, CS2, or $\overline{WE}$, whichever occurs first.
6. t_{WHZ} is specified by the time when DATA OUT is floating, not defined by output level.
7. When I/O pins are data output mode, don't force inverse signals to those pins.

A.C. Test Conditions

Item	Condition
Input Pluse Levels	0V to 3.0V
Input Rise and Fall Time	5ns
Input and Output Timing Reference Levels	1.5V
Output Load	See Figures below



Output Load



Output Load for
t_{CLZ}, t_{OLZ}, t_{CHZ}, t_{OHZ}, t_{WHZ} and t_{OW}

Operation Truth Table

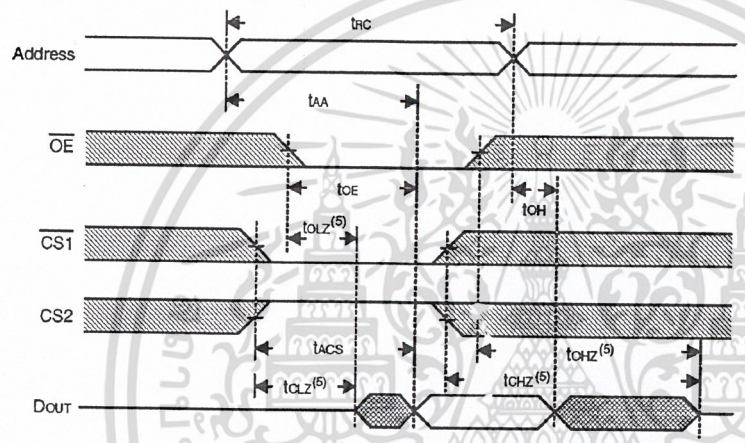
All relations between $\overline{WE}$, $\overline{OE}$, $\overline{CS1}$ and CS2 can be described as the following truth table

$\overline{CS1}$	CS2	$\overline{OE}$	$\overline{WE}$	Mode	D0~D7
H X	X* L	X X	X X	Standby	High-Z
L	H	L	H	Read	Dout
L	H	H	H	Read	High-Z
L	H	X	L	Write	Din

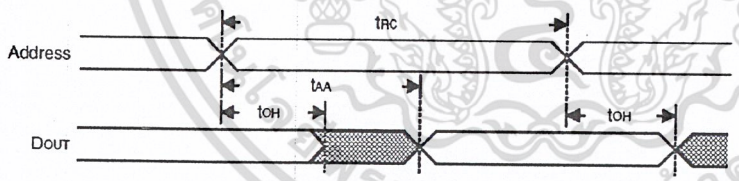
*X: Don't Care, Logical High or Low

Timing Diagrams

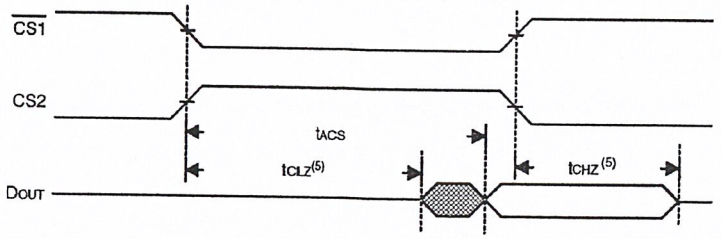
Read cycle 1(1)



Read cycle 2(1, 2, 4)

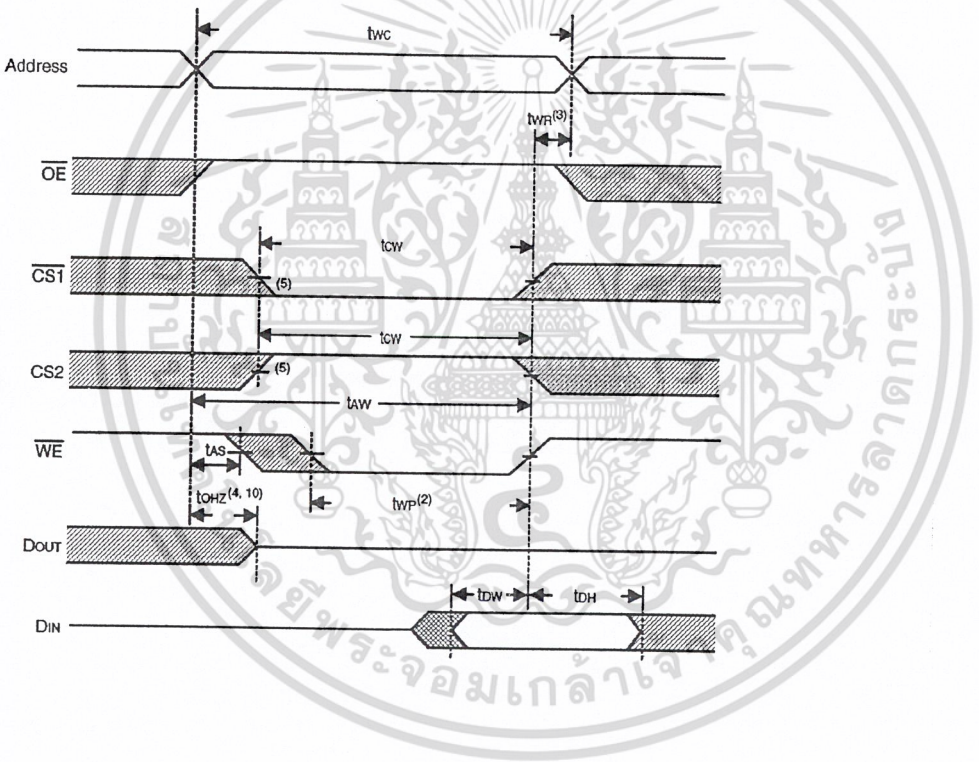


Read cycle 3(1, 3, 4)

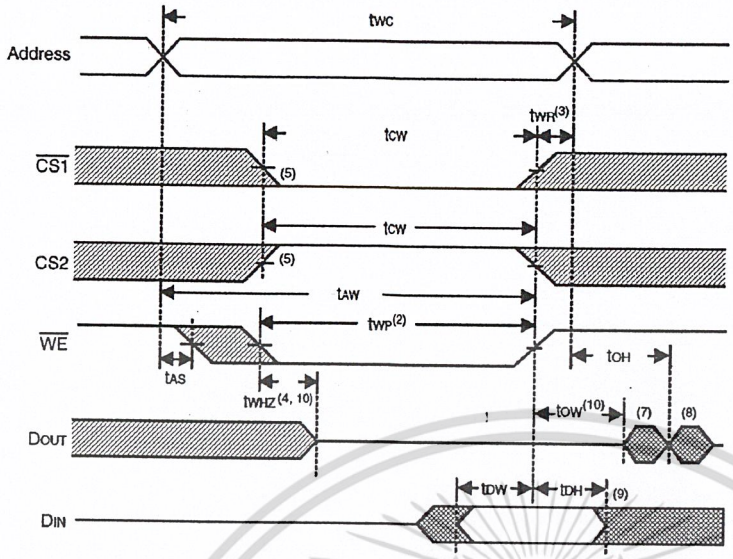


Notes:
1. $\overline{WE}$ is High for Read Cycle
2. Device is continuously enabled, $\overline{CS1}$=V_{IL} & CS2=V_{IH}
3. Address valid prior to or coincident with $\overline{CS1}$ transition low & CS2 transition high
4. $\overline{OE}$=V_{IL}
5. Transition is measured ± 500mV from steady state

Write cycle 1(1)



Write cycle 2[1, 6]



Notes:
1. $\overline{WE}$ must be high during all address transitions
2. A write occurs during the overlap (t_{WP}) of a low $\overline{CS1}$, a high CS2 and a low $\overline{WE}$
3. t_{WR} is measured from the earlier of $\overline{CS1}$ or $\overline{WE}$ going high and CS2 going low to the end of write cycle
4. During this period, I/O pins are in the output state so the input signals of oppsite phase to the outputs must not be applied
5. If the $\overline{CS1}$ low transition (or CS2 high transition) occurs simultaneously with the $\overline{WE}$ low transitions or after the $\overline{WE}$ transition, outputs remain in a high impedance state
6. $\overline{OE}$ is continuously low ($\overline{OE}=V_{IL}$)
7. D_{OUT} is the same phase of write data of this write cycle
8. D_{OUT} is the read data of next address
9. If $\overline{CS1}$ is low (or CS2 is high) during this period, I/O pins are in the output state. Then the data input signals of opposite phase to the outputs must not be applied to them
10. Transition is measured ±500mV from steady state

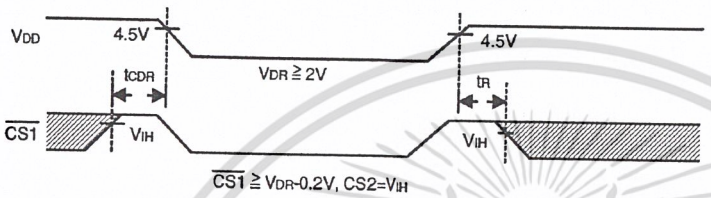
Data Rentention Characteristics

(Ta=–40°C to 85°C)

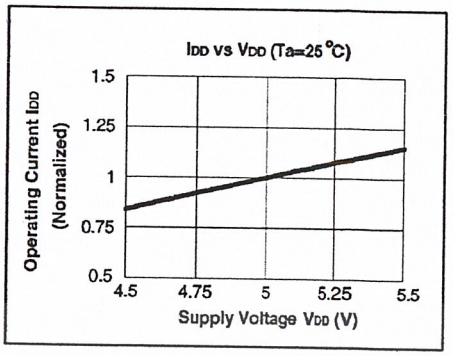
Symbol	Parameter	Conditions	Min.	Max.	Unit
V_{DR}	V_{DD} for Data Retention	$\overline{CS1}$=CS2≥V_{DD}–0.2V	2	5.5	V
I_{CCDR}	Data Retention Current	V_{DD}=3V, $\overline{CS1}$=CS2≥V_{DD}–0.2V V_{IN}≥V_{DD}–0.2V or V_{IN}≤0.2V	—	50	μA
t_{CDR}	Chip Disable Data Retention Time	See Retention Timing	0	—	ns
t_R	Operation Recovery Time	See Retention Timing	t_{RC}*	—	ns

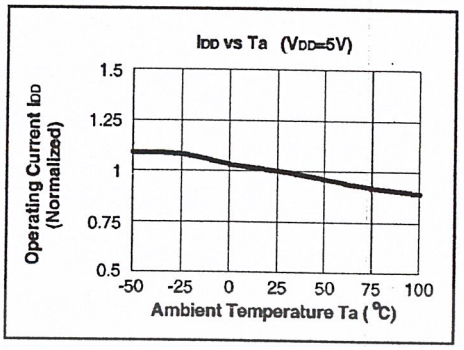
*t_{RC}=Read Cycle Time

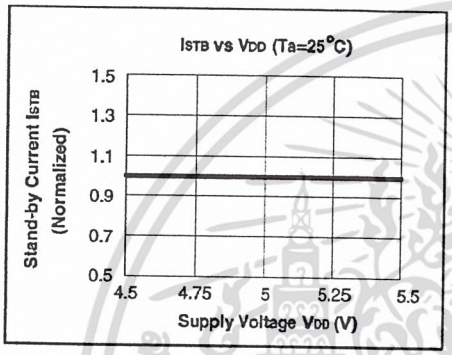
Low V_{DD} Data Retention Timing

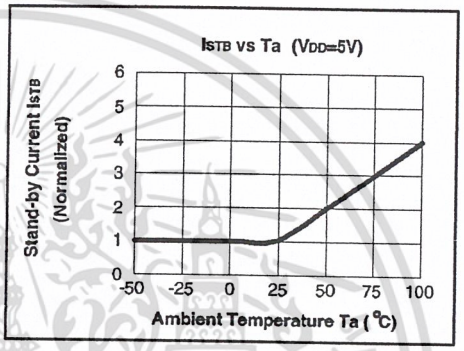


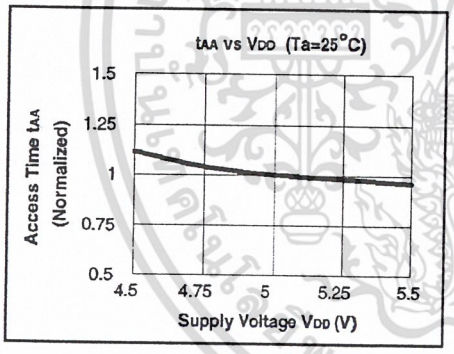
Characteristic Curves

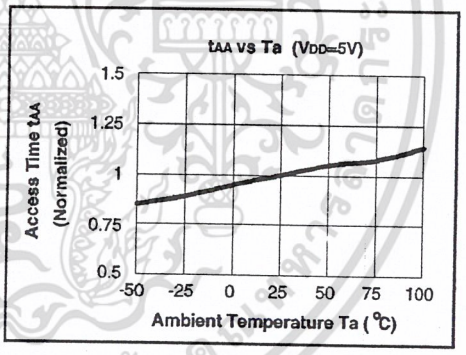


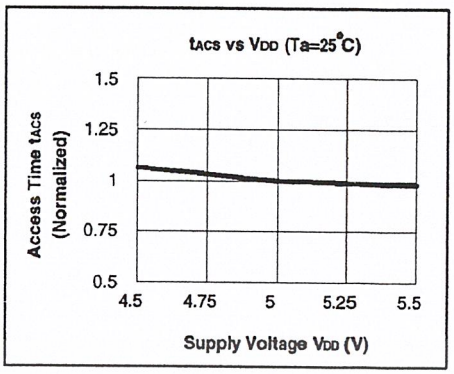


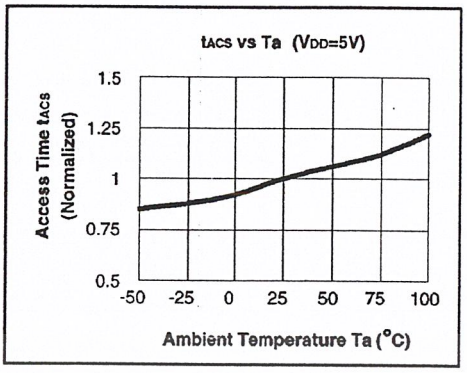


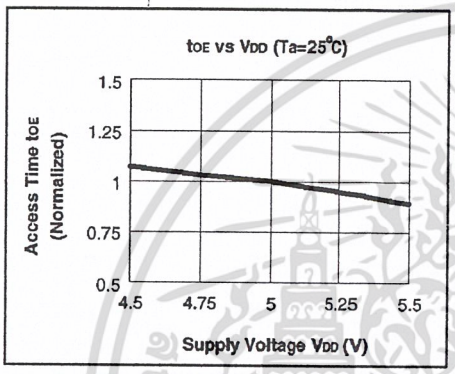


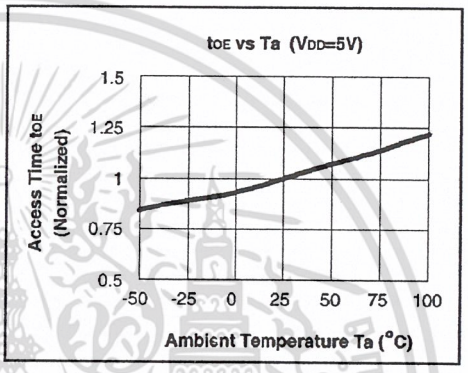








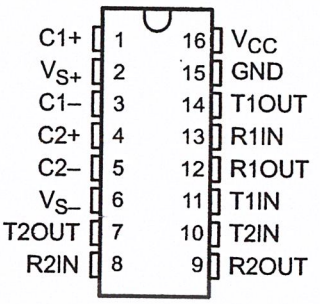




SLLS047H – FEBRUARY 1989 – REVISED FEBRUARY 2002

- Operates With Single 5-V Power Supply
- LinBiCMOS™ Process Technology
- Two Drivers and Two Receivers
- ±30-V Input Levels
- Low Supply Current . . . 8 mA Typical
- Meets or Exceeds TIA/EIA-232-F and ITU Recommendation V.28
- Designed to be Interchangeable With Maxim MAX232
- ESD Protection Exceeds JESD 22
  – 2000-V Human-Body Model (A114-A)
- Applications
  - TIA/EIA-232-F
  - Battery-Powered Systems
  - Terminals
  - Modems
  - Computers
- Package Options Include Plastic Small-Outline (D, DW, NS) Packages and Standard Plastic (N) DIPs

D, DW, N, OR NS PACKAGE
(TOP VIEW)



description

The MAX232 device is a dual driver/receiver that includes a capacitive voltage generator to supply EIA-232 voltage levels from a single 5-V supply. Each receiver converts EIA-232 inputs to 5-V TTL/CMOS levels. These receivers have a typical threshold of 1.3 V and a typical hysteresis of 0.5 V, and can accept ±30-V inputs. Each driver converts TTL/CMOS input levels into EIA-232 levels. The driver, receiver, and voltage-generator functions are available as cells in the Texas Instruments LinASIC™ library.

The MAX232 is characterized for operation from 0°C to 70°C. The MAX232I is characterized for operation from –40°C to 85°C.

AVAILABLE OPTIONS

T_A	PACKAGED DEVICES		
	SMALL OUTLINE (D, NS)	SMALL OUTLINE (DW)	PLASTIC DIP (N)
0°C to 70°C	MAX232D MAX232NS	MAX232DW	MAX232N
–40°C to 85°C	MAX232ID	MAX232IDW	MAX232IN

The D and DW packages are available taped and reeled by adding an R to the part number (i.e., MAX232DR). The NS package is only available taped and reeled.

absolute maximum ratings over operating free-air temperature range (unless otherwise noted)†

Input supply voltage range, V_{CC} (see Note 1) ..... –0.3 V to 6 V
Positive output supply voltage range, V_{S+} ..... V_{CC} – 0.3 V to 15 V
Negative output supply voltage range, V_{S-} ..... –0.3 V to –15 V
Input voltage range, V_I: Driver ..... –0.3 V to V_{CC} + 0.3 V
Receiver ..... ±30 V
Output voltage range, V_O: T1OUT, T2OUT ..... V_{S-} –0.3 V to V_{S+} + 0.3 V
R1OUT, R2OUT ..... –0.3 V to V_{CC} + 0.3 V
Short-circuit duration: T1OUT, T2OUT ..... Unlimited
Package thermal impedance, θ_{JA} (see Note 2): D package ..... 73°C/W
DW package ..... 57°C/W
N package ..... 67°C/W
NS package ..... 64°C/W
Lead temperature 1,6 mm (1/16 inch) from case for 10 seconds ..... 260°C
Storage temperature range, T_{stg} ..... –65°C to 150°C

† Stresses beyond those listed under "absolute maximum ratings" may cause permanent damage to the device. These are stress ratings only, and functional operation of the device at these or any other conditions beyond those indicated under "recommended operating conditions" is not implied. Exposure to absolute-maximum-rated conditions for extended periods may affect device reliability.

NOTE 1: All voltage values are with respect to network ground terminal.
2. The package thermal impedance is calculated in accordance with JESD 51-7.

recommended operating conditions

			MIN	NOM	MAX	UNIT
V_{CC}	Supply voltage		4.5	5	5.5	V
V_{IH}	High-level input voltage (T1IN,T2IN)		2			V
V_{IL}	Low-level input voltage (T1IN, T2IN)				0.8	V
R1IN, R2IN	Receiver input voltage				±30	V
T_A	Operating free-air temperature	MAX232	0		70	°C
		MAX232I	–40		85	

electrical characteristics over recommended ranges of supply voltage and operating free-air temperature range (unless otherwise noted)

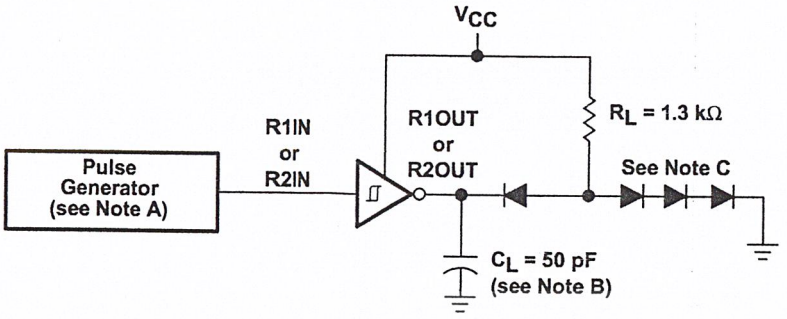
PARAMETER			TEST CONDITIONS		MIN	TYP†	MAX	UNIT
V_{OH}	High-level output voltage	T1OUT, T2OUT	R_L = 3 kΩ to GND		5	7		V
		R1OUT, R2OUT	I_{OH} = –1 mA		3.5			
V_{OL}	Low-level output voltage‡	T1OUT, T2OUT	R_L = 3 kΩ to GND			–7	–5	V
		R1OUT, R2OUT	I_{OL} = 3.2 mA				0.4	
VIT+	Receiver positive-going input threshold voltage	R1IN, R2IN	V_{CC} = 5 V,	T_A = 25°C		1.7	2.4	V
VIT–	Receiver negative-going input threshold voltage	R1IN, R2IN	V_{CC} = 5 V,	T_A = 25°C	0.8	1.2		V
Vhys	Input hysteresis voltage	R1IN, R2IN	V_{CC} = 5 V		0.2	0.5	1	V
ri	Receiver input resistance	R1IN, R2IN	V_{CC} = 5,	T_A = 25°C	3	5	7	kΩ
ro	Output resistance	T1OUT, T2OUT	$V_{S+} = V_{S-}$ = 0,	V_O = ± 2 V	300			Ω
IOS§	Short-circuit output current	T1OUT, T2OUT	V_{CC} = 5.5 V,	V_O = 0		±10		mA
IIS	Short-circuit input current	T1IN, T2IN	V_I = 0				200	µA
ICC	Supply current		V_{CC} = 5.5 V, T_A = 25°C	All outputs open,		8	10	mA

† All typical values are at V_{CC} = 5 V, T_A = 25°C.
‡ The algebraic convention, in which the least positive (most negative) value is designated minimum, is used in this data sheet for logic voltage levels only.
§ Not more than one output should be shorted at a time.

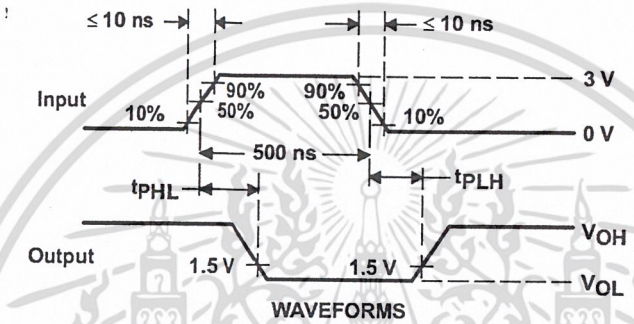
switching characteristics, V_{CC} = 5 V, T_A = 25°C

PARAMETER		TEST CONDITIONS	MIN	TYP	MAX	UNIT
$t_{PLH(R)}$	Receiver propagation delay time, low- to high-level output	See Figure 1		500		ns
$t_{PHL(R)}$	Receiver propagation delay time, high- to low-level output	See Figure 1		500		ns
SR	Driver slew rate	R_L = 3 kΩ to 7 kΩ, See Figure 2			30	V/µs
SR(tr)	Driver transition region slew rate	See Figure 3		3		V/µs

PARAMETER MEASUREMENT INFORMATION



TEST CIRCUIT

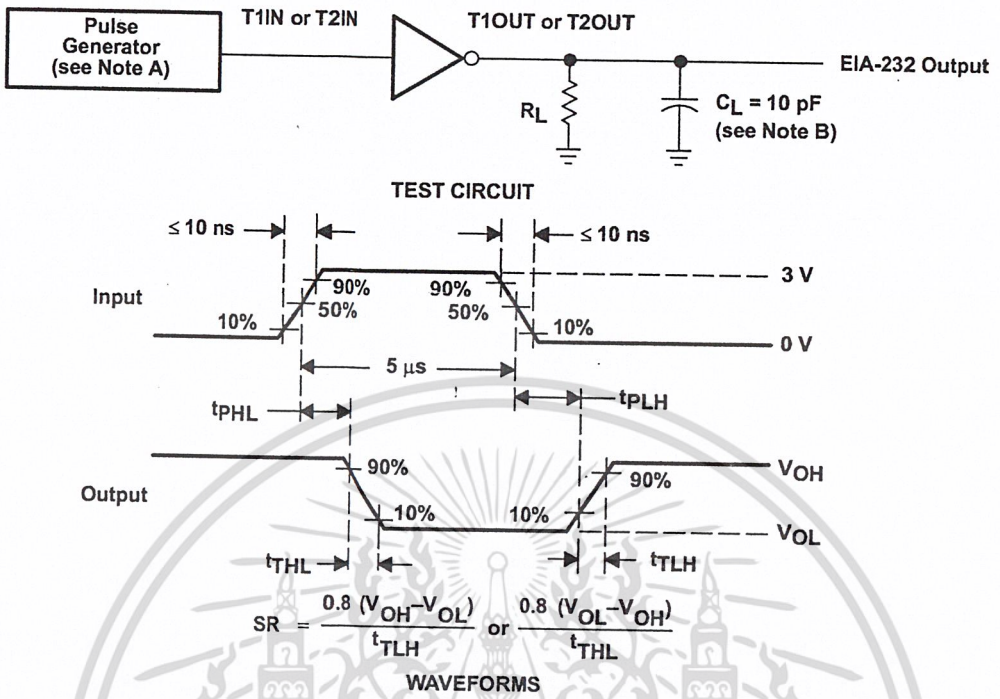


WAVEFORMS

NOTES: A. The pulse generator has the following characteristics: Z_O = 50 Ω, duty cycle ≤ 50%.
B. C_L includes probe and jig capacitance.
C. All diodes are 1N3064 or equivalent.

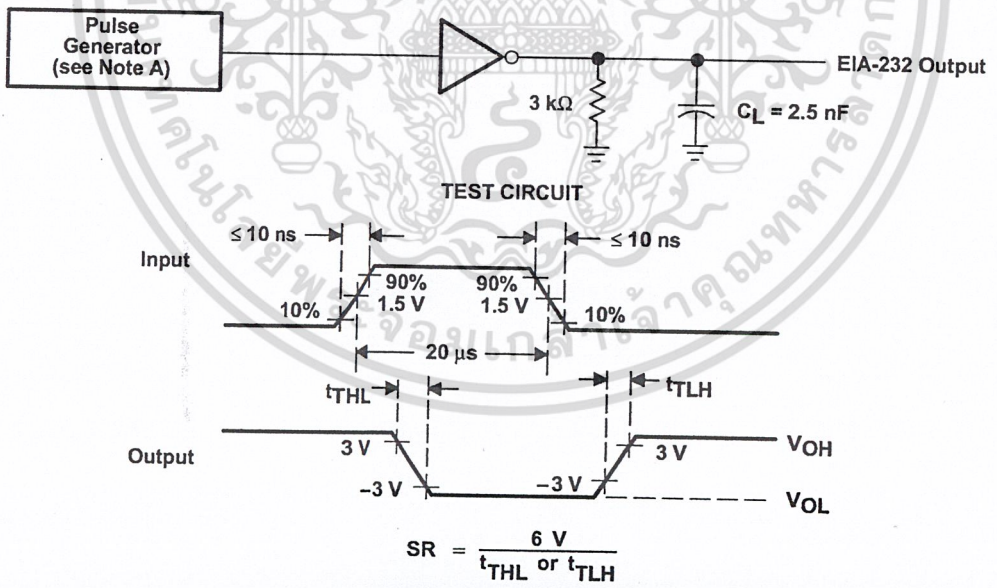
Figure 1. Receiver Test Circuit and Waveforms for t_{PHL} and t_{PLH} Measurements

PARAMETER MEASUREMENT INFORMATION



NOTES: A. The pulse generator has the following characteristics: Z_O = 50 Ω, duty cycle ≤ 50%.
B. C_L includes probe and jig capacitance.

Figure 2. Driver Test Circuit and Waveforms for t_{PHL} and t_{PLH} Measurements (5-µs input)



NOTE A: The pulse generator has the following characteristics: Z_O = 50 Ω, duty cycle ≤ 50%.

Figure 3. Test Circuit and Waveforms for t_{THL} and t_{TLH} Measurements (20-µs input)

APPLICATION INFORMATION

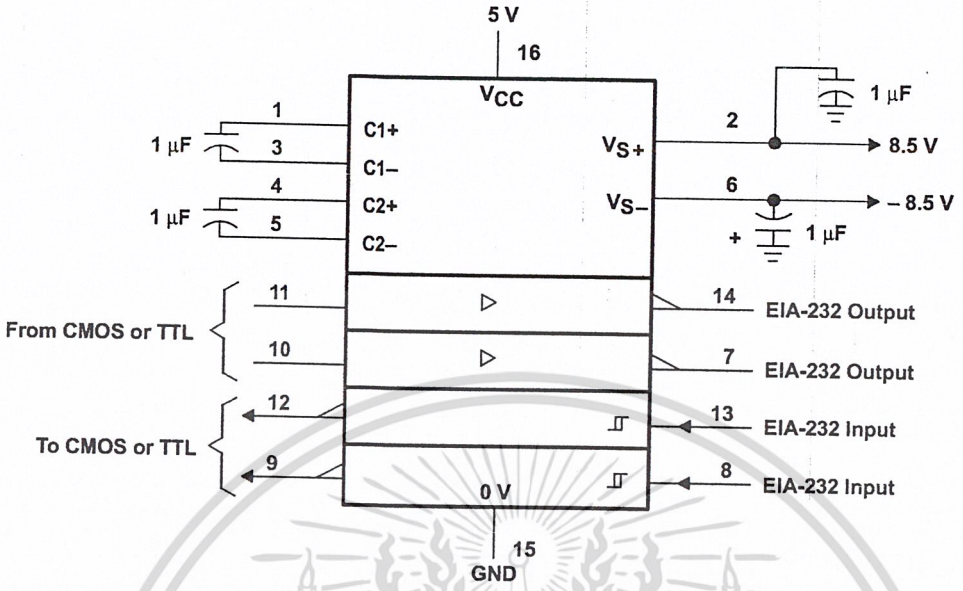


Figure 4. Typical Operating Circuit

IMPORTANT NOTICE

Texas Instruments Incorporated and its subsidiaries (TI) reserve the right to make corrections, modifications, enhancements, improvements, and other changes to its products and services at any time and to discontinue any product or service without notice. Customers should obtain the latest relevant information before placing orders and should verify that such information is current and complete. All products are sold subject to TI's terms and conditions of sale supplied at the time of order acknowledgment.

TI warrants performance of its hardware products to the specifications applicable at the time of sale in accordance with TI's standard warranty. Testing and other quality control techniques are used to the extent TI deems necessary to support this warranty. Except where mandated by government requirements, testing of all parameters of each product is not necessarily performed.

TI assumes no liability for applications assistance or customer product design. Customers are responsible for their products and applications using TI components. To minimize the risks associated with customer products and applications, customers should provide adequate design and operating safeguards.

TI does not warrant or represent that any license, either express or implied, is granted under any TI patent right, copyright, mask work right, or other TI intellectual property right relating to any combination, machine, or process in which TI products or services are used. Information published by TI regarding third–party products or services does not constitute a license from TI to use such products or services or a warranty or endorsement thereof. Use of such information may require a license from a third party under the patents or other intellectual property of the third party, or a license from TI under the patents or other intellectual property of TI.

Reproduction of information in TI data books or data sheets is permissible only if reproduction is without alteration and is accompanied by all associated warranties, conditions, limitations, and notices. Reproduction of this information with alteration is an unfair and deceptive business practice. TI is not responsible or liable for such altered documentation.

Resale of TI products or services with statements different from or beyond the parameters stated by TI for that product or service voids all express and any implied warranties for the associated TI product or service and is an unfair and deceptive business practice. TI is not responsible or liable for any such statements.

Mailing Address:

Texas Instruments
Post Office Box 655303
Dallas, Texas 75265