ศึกษาและพัฒนาไมโครโปรเซสเซอร์ รุ่น Rabbit 3000

Research & Development for Microprocessor Rabbit 3000

โดย

นางสาว สุจิตรา ท้วมจันทร์ รหัสนักศึกษา 43010475

นาย ต่อพงษ์ แสงโชติช่วงชัย รหัสนักศึกษา 43010852

ปริญญานิพนธ์ส่วนนี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิศวกรรมศาสตรบัณฑิต

ภาควิชาวิศวกรรมสารสนเทศ

คณะวิศวกรรมศาสตร์

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

ปีการศึกษา 2546

RESEARCH AND DEVELOPMENT FOR RABBIT 3000

BY

MISS SUCHITRA TUAMJUN NO 43010475

MR. TOPONG SANGCHOTCHAUNGCHAI NO 43010852

A THESIS SUBMITED IN PARTIAL FULFILLMENT OF

THE REQUIREMENT FOR THE DEGREE OF

BACHELOR IN DEPARTMENT OF INFORMATION ENGINEERING

FACULTY OF ENGINEERING

KING MONGKUT'S INSTITUTE OF TECHNOLOGY LADKRABANG 2003

หัวข้อปริญญานิพนธ์ REASEARCH AND DEVELOPMENT FOR RABBIT3000

ชื่อนักศึกษา นาวสาว สุจิตรา ท้วมจันทร์ รหัสนักศึกษา 43010475

นาย ต่อพงษ์ แสงโชติช่วงชัย รหัสนักศึกษา 43010852

อาจารย์ที่ปรึกษา รศ.ดร. ปิติเขต สู้รักษา

อาจารย์บุณย์ชนะ ภู่ระหงษ์

ระดับการศึกษา ปริญญาตรี วิศวกรรมศาสตรบัณฑิต

สาขา วิศวกรรมสารสนเทศ

ภาควิชา วิศวกรรมสารสนเทศ

ปีการศึกษา 2546

ปริญญานิพนธ์นี้ได้รับความเห็นชอบจากอาจารย์ที่ปรึกษาเรียบร้อยแล้ว

.....

(รศ.ดร. ปิติเขต สู้รักษา)

.....

(อาจารย์บุณย์ชนะ ภู่ระหงษ์)

หัวข้อปริญญานิพนธ์ REASEARCH AND DEVELOPMENT FOR RABBIT3000
ชื่อนักศึกษา นาวสาว สุจิตรา ท้วมจันทร์ รหัสนักศึกษา 43010475
นาย ต่อพงษ์ แสงโชติช่วงชัย รหัสนักศึกษา 43010852
อาจารย์ที่ปรึกษา รศ.ดร. ปิติเขต สู้รักษา
อาจารย์บุณย์ชนะ ภู่ระหงษ์
ระดับการศึกษา ปริญญาตรี วิศวกรรมศาสตรบัณฑิต
สาขา วิศวกรรมสารสนเทศ
ภาควิชา วิศวกรรมสารสนเทศ
ปีการศึกษา 2546

บทคัดย่อ

โครงงานนี้ นำเสนอการควบคุมอุปกรณ์ไฟฟ้าโดยใช้ Rabbit 3000 Rabbit 3000 มีความสามารถหลากหลายในระบบเครือข่าย ซึ่งเราได้มีการศึกษาทางด้าน ฮาร์ดแวร์และ ด้านซอฟแวร์มีความเข้าใจในการทำงานของ Rabbit 3000 เช่นเรื่องการmapping address การติดต่ออุปกรณ์ต่างๆและเข้าใจในการติดต่อทางอินเตอร์เน็ต ด้านซอฟแวร์ ซึ่งเราต้องใช้ Dynamic C , C , CGI และ Java โครงงานนี้เสร็จไปได้ด้วยดี ซึ่งเป็นก้าวแรกในการพัฒนาและการนำไปใช้ในชีวิตประจำวันต่อไป

Thesis Tile REASEARCH AND DEVELOPMENT FOR RABBIT3000

Student Miss Suchitra Taumjun No 43010475

Mr. Topong Sangchotchaungchai No 43010852

Advisor Assoc.Prof.Dr. Pitikhate Sooraksa

Mr. Boochana Poorahong

Graduate Level Bachelor Degree of information Engineering

Department Information Engineering

Academic year 2004

ABSTRACT

This project presents an application of Rabbit 3000. Rabbit 3000 is able to communicate in Network System . This study includes both hardware and software aspects. For hardware, we study about mapping address, interface other devices and internet connectors. For software, we study Dynamic C, C, CGI and Java .This project may be recognized as the first step of development for Rabbit 3000 application.

กิตติกรรมประกาศ

ปริญญานิพนธ์ฉบับนี้ ได้รับความช่วยเหลือทางด้านอุปกรณ์และข้อมูลเป็นอย่างดี จาก รศ.ดร. ปิติเขต สู้รักษา และ อาจารย์บุณย์ชนะ ภู่ระหงษ์ อาจารย์ที่ปรึกษาปริญญานิพนธ์ ซึ่งให้คำแนะนำและข้อคิดเห็นเป็นอย่างดี ของการวิจัยมาด้วยดีตลอด

สุดท้ายนี้ ผู้ทำปริญญานิพนธ์ใคร่ขอกราบขอบพระคุณท่านอาจารย์ที่ปรึกษา และขอขอบคุณทุกคนที่ให้ความช่วยเหลือให้ ปริญญานิพนธ์ฉบับนี้สำเร็จลุล่วงไปได้ด้วยดี

สารบัญ

สารบัญ (ต่อ)

สารบัญรูปภาพ

สารบัญรูปภาพ (ต่อ)

สารบัญตาราง

สารบัญตาราง (ต่อ)

สารบัญตาราง (ต่อ)

บทที่ 1

บทนำ

1.1แนวคิดและที่มาของปัญหา

ในปัจจุบันความต้องการเทคโนโลยีมากขึ้นทำให้ผู้ใช้คิดว่า ถ้าเราสามารถควบคุมอุปกรณ์ต่างๆเช่น ทีวี,ตู้เย็น,พัดลม เป็นต้น ผ่านระบบเครือข่ายและให้เครื่องใช้ไฟฟ้าติดต่อกันเองได้จึงเกิดแนวคิดของปริญญานิพนธ์นี้ขึ้นมา

1.2วัตถุประสงค์

1. เพื่อการศึกษา Rabbit 3000 และนำความรู้นี้ไปพัฒนาในด้านการควบคุมผ่านเครือข่าย
2. เพื่อศึกษาความเป็นไปได้ที่จะนำ Rabbit มาใช้ในอนาคต
3. เพื่อนำมาใช้ประยุกต์ในอุปกรณ์ทางด้านเครือข่าย

1.3 ขอบเขตของโครงงาน

ศึกษา Rabbit และนำความรู้มาประยุกต์ใช้ในระบบเครือข่ายโดยสมมุติให้ Rabbit ในการควบคุมการทำงานเครื่องใช้ไฟฟ้าของระบบโดยผ่านระบบอินเตอร์เน็ต (Internet)

1.4 ผลที่คาดว่าจะได้รับ

1. สามารถควบคุมระบบสิ่งต่างๆผ่านระบบเครือข่ายได้
2.ทำให้เกิดความสะดวกในเรื่องของข้อมูลและข่าวสาร
3.เพื่อการวางแผนของธุรกิจเกี่ยวกับ อินเตอร์เน็ตในอนาคต

บทที่ 2
ทฤษฎีที่เกี่ยวข้องกับ

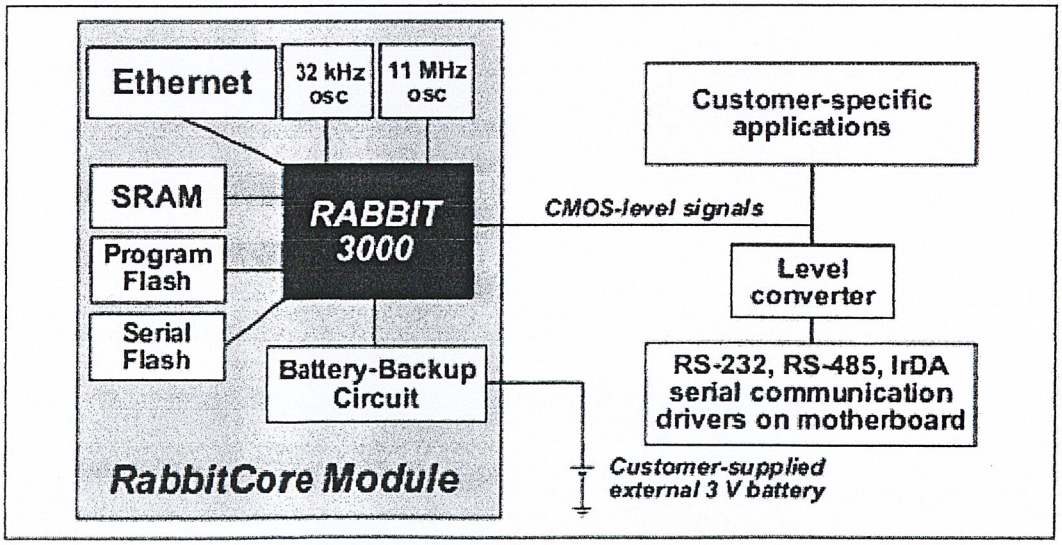
คุณสมบัติทางด้าน ฮาร์ดแวร์

2.1.1ชุดพัฒนา RCM3000

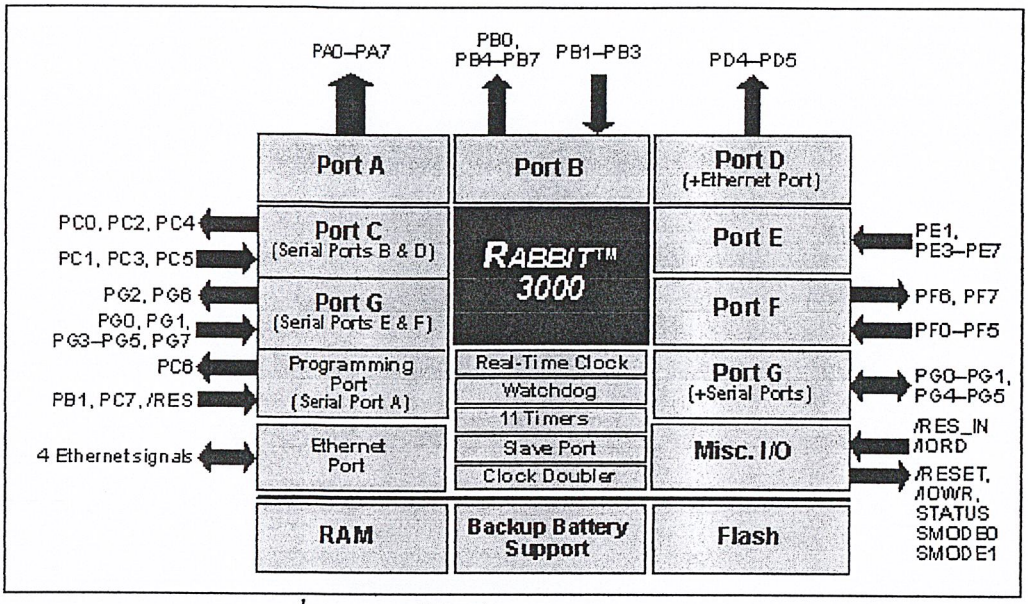
ชุดพัฒนาRabbit 3000 ประกอบไปด้วย Rabbit Core Module และ Prototyping Board ซึ่ง เราสามารถเขียนโปรแกรมทดสอบ Rabbit 3000 รวมไปถึงการออกแบบแอพพลิเคชั่น

ชุดพัฒนานี้เป็นชุดพัฒนาสำหรับ TCP/IP เป็นของบริษัท Rabbit Semiconductor ซึ่งชุดพัฒนา RCM3000(Rabbit Core Module 3000) ใช้โปรเซสเซอร์ของบริษัทตัวเองโดยรุ่น Rabbit 3000 มีคุณสมบัติคร่าวๆ ดังนี้

- ใช้โปรเซสเซอร์ Rabbit 3000 ทำงานที่ 29.4 MHz ใช้ไฟเลี้ยง 3.3 V
- ใช้ Port 10/100 base T
- หน่วยความจำแฟลช 512 KB/SDRAM 512 KB
- มี 52 ดิจิตอล I/O พอร์ท
- พอร์ทอนุกรม 6 พอร์ท
- มีโปรแกรมสำหรับประยุกต์สำหรับพัฒนา คือ Dynamic C ซึ่งเป็นภาษาพื้นฐาน



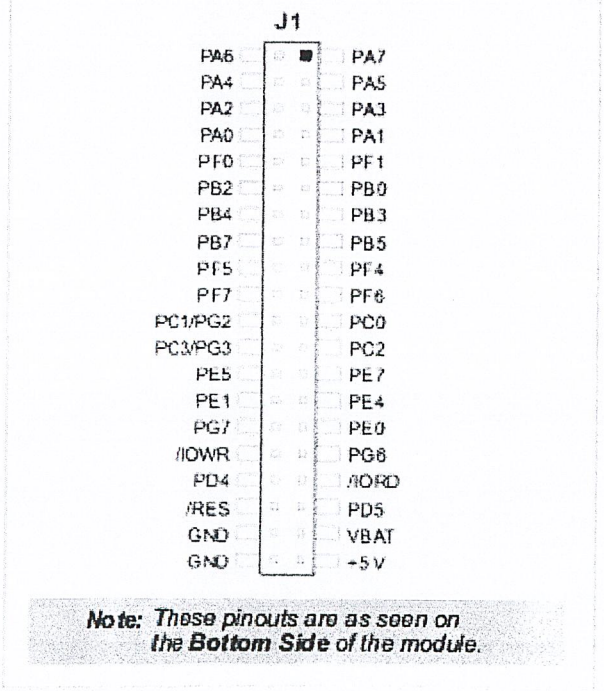
รูปที่ 2.1 RCM 3000



รูปที่ 2.2 การใช้ พอร์ทต่างของ Rabbit 3000

2.1.1.2 RCM 3000 ดิจิตอลอินพุตท์และเอาพุตท์

RCM3000 มี52อินพุทต์/เอาพุตท์ แบ่งได้เป็น 7 พอร์ท พอร์ทละ 8 บิต บน J1 J2 โดยจะมี 44 ขาสัญญาณเป็นได้ทั้ง เอาพุตท์และอินพุตท์ คือ PA0-PA7 , PB2-PB7, PD2-PD7, PE0-PE1, PF0-PF7, PG20-PG7



รูปที่ 2.3 ขาต่างๆ ของ RCM3000

Pin		Pin Name	Default Use	Alternate Use	Notes
Header J1	1-8	PA[7:0]	Parallel I/O	External data bus (ID0-ID7) Slave port data bus (SD0-SD7)	External Data Bus
	9	PF1	Input/Output	QD1A CLKC	
	10	PF0	Input/Output	QD1B CLKD	
	11	PB0	Input/Output	CLKB	
	12	PB2	Input/Output	IA0 /SWR	External Address 0 Slave port write
	13	PB3	Input/Output	IA1 /SRD	External Address 1 Slave port read
	14	PB4	Input/Output	IA2 SA0	External Address 2 Slave Port Address 0
	15	PB5	Input/Output	IA3 SA1	External Address 3 Slave Port Address 1
	16	PB7	Input/Output	IA5 /SLAVEATTN	External Address 5 Slave Port Attention
	17	PF4	Input/Output	AQD1B PWM0	
	18	PF5	Input/Output	AQD1A PWM1	
	19	PF6	Input/Output	AQD2B PWM2	
	20	PF7	Input/Output	AQD2A PWM3	
	21	PC0	Output	TXD	Serial Port D
	22	PC1/PG2	Input/Output	RXD/TXF	Serial Port D Serial Port F
	23	PC2	Output	TXC	Serial Port C

	24	PC3/PG3	Input/Output	RXC/RXF	Serial Port C Serial Port F
	25	PE7	Input/Output	I7 /SCS	External Address 7 Slave Port Chip Select
Header J1	26	PE5	Input/Output	I5 INT1B	
	27	PE4	Input/Output	I4 INT0B	
	28	PE1	Input/Output	I1 INT1A	I/O Strobe 1 Interrupt 1A
	29	PE0	Input/Output	I0 INT0A	I/O Strobe 0 Interrupt 0A
	30	PG7	Input/Output	RXE	Serial Port E
	31	PG6	Input/Output	TXE	
	32	/IOWR	Output		External write strobe
	33	/IORD	Input		External read strobe
	34	PD4	Input/Output	ATXB	Alternate Serial Port B
	35	PD5	Input/Output	ARXB	
	36	/RES	Reset output	Reset input	Reset output from Reset Generator
	37	VBAT			
	38	GND			
	39	+5 V			
	40	GND			

ตารางที่ 1 หน้าที่ของขาต่างๆใน RCM3000

2.1.1.3 การเชื่อมต่อหน่วยความจำของ I/O

Rabbit 3000 มีบัสแอดเดรส (A0-A19) และบัสดาต้า (D0-D7) ไปยังชิพหน่วยความจำแบบแฟลชและ SDRAM มีอินพุตท์/เอาพุตท์สำหรับเขียน (/IOWR) และอินพุตท์/เอาพุตท์สำหรับอ่าน (/IORD) เพื่อใช้ติดต่อกับอุปกรณ์ภายนอก

พอร์ทขนาน A สามารถถูกใช้เป็นดาต้าบัสI/O จากภายนอกได้โดยตรง(external I/O data bus) โดยแยกอินพุตท์และเอาพุตท์ออกจากดาต้าบัสหลัก พอร์ทขนาน B (PB3-PB7) ก็สามารถใช้เป็นดาต้าบัสข้อมูลภายนอก ได้เช่นเดียวกัน

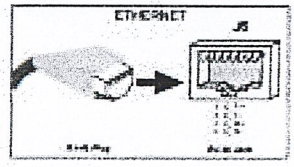
2.1.1.4 พอร์ทอนุกรม

มีพอร์ทอนุกรม 6 พอร์ท คือ A, B,C ,D ,E และ F พอร์ทอนุกรมทั้ง 6 สามารถทำงานอะซิงโคนัส เพื่อให้อัตราการส่งข้อมูลของระบบสัญญาณนาฬิกาแบ่งจาก 16 พอร์ทอะซิงโคนัสสามารถจัดการที่ 7-8 บิตข้อมูล บิตที่ 9 เป็นบิตแอดเดรสซึ่งถูกส่งเป็นไบต์แรกของข้อความ(message) พอร์ทอนุกรม A ,B ,C และ D จะทำงานในโหมดสัญญาณนาฬิกาแบบอนุกรม ในโหมดนี้จะรับสัญญาณนาฬิกาเข้ามาเพื่อซิงโคนัสข้อมูลเข้าและออกตามจังหวะของสัญญาณนาฬิกาในการสื่อสารของอุปกรณ์ 2 ตัว ที่สนับสนุนการใช้สัญญาณนาฬิกา เมื่อ Rabbit3000 ใช้สัญญาณนาฬิกาอัตราการส่งข้อมูล จะสูงถึง 80 % ของระบบสัญญาณนาฬิกาที่ถูกแบ่งจาก 128 หรือ 183,750 bps ที่ความเร็วของสัญญาณนาฬิกาที่ 29.4 MHz

พอร์ทอนุกรม E และ F สามารถตั้งค่าให้เป็น SDLC/HDLC ซึ่งเป็นรูปแบบของ SDLC จะสนับสนุนโปรโตคอล Irda ทั้ง 2พอร์ท

2.1.1.5 Ethernet Port

Programmable I/O Interface เป็นส่วนของการเชื่อมต่อกับอุปกรณ์ภายนอกเพื่อควบคุมหรืออ่านสถานะต่างๆของอุปกรณ์ โดยผู้ใช้สามารถโปรแกรมเองได้



รูปที่ 2.4 พอร์ท Ethernet RJ-45

2.1.2 Rabbit 3000 Design Feature

Rabbit 3000 ได้พัฒนาการออกแบบใหม่ขึ้นมา แต่ตัวประมวลผลและชุดคำสั่งยังคงมีคุณสมบัติใกล้เคียงกันเมื่อเปรียบเทียบกับตัวประมวลผลตัวอื่นๆ โดยที่มีรูปบบและโครงสร้างของบอร์ดและแผงผังของรีจิสเตอร์คล้ายกับ z80และ z180 ซึ่งมีความหล้าหลังและมีความฟุ่มเฟือยในการใช้ทรัพยากรจึงได้เพิ่มประสิทธิภาพโดยการเพิ่ม 1 ไบต์ค่อ 1 ออปโค๊ต สำหรับชุดคำสั่งใหม่ซึ่ง

เป็นข้อได้เปรียบของการพัฒนาโดยที่มีความคล้ายกับ Z80 และ Z180 ซึ่งง่ายต่อความเข้าใจ ในภาษาแอสแซมบลีของ Rabbit โดยที่ชุดคำสั่งของ Z80 และ Z180 นั้นสามารถนำมาใช้ในการเขียนแอสแซมบลีและประมวลผลบน Rabbit

มีการเปลี่ยนเทคโนโลยีที่ใช้บน Z80 และ Z180 และลักษณะพิเศษของ Rabbit จะถูกนำมาแทนที่ เช่น Rabbit ไม่สนับสนุนการใช้ Dynamic Ram แต่สนับสนุนการใช้ Static Ram เพราะมีราคาถูกลง Rabbit ไม่ใช้ DMA (Direct Memory Access) เพราะ DMA ไม่สามารถนำไปใช้กับระบบอื่นได้ เช่นเรื่องความเร็วในการอินเตอร์รัพบน Routine, External Sate Machine, Slave Process

ผู้ที่เคยขียนภาษา C บนชุดคำสั่งบน Z80 มักจะประสบปัญหาเรื่องการทำงานของภาษา C เพราะส่วนใหญ่ของปัญหาคือการขาดแคลนคำสั่งที่ใช้ ในการควบคุม 16-bit words และสำหรับการควบคุมการรับส่งข้อมูลไปยัง computed address โดยเฉพาะในขณะที่ สแตก เก็บข้อมูลอยู่

ปัญหาอย่างอื่น เช่นการประมวลผลแบบ 8 bit ซึ่งมีความเร็วในการประเมินผลที่ช้าและขาดแคลนเรื่อง number crunching เรื่อง ค่าในการคำนวณซึ่งเป็นสิ่งที่สำคัญในการพัฒนาในอนาคต เพื่อที่จะทำให้ระบบเล็กลง

Rabbit สนับสนุน 4 ระดับ สำหรับ การจัดอันดับ ค่าความสำคัญ (Priorities) ซึ่งเป็นลักษณะที่เพิ่มประสิทธิภาพในการจัดการในเรื่องความเร็วของ เส้นทางในการอินเตอร์รัพ

2.1.2.1 Rabbit 8-bit Processor Vs Other Processor

Rabbit 3000 โปรเซสเซอร์ได้ถูกออกแบบโดยใช้ระบบวัตถุประสงค์ของการฝึกหัด (Object of creating practical system) ซึ่งในการแก้ไขปัญหาที่เกิดขึ้นในปัจจุบันโดยใช้รูปแบบเศรษฐศาสตร์

- Rabbit ที่ถูกออกแบบให้มีการทำงานโดยที่ใช้ ระบบ EMI น้อย และมีสัญญาณนาฬิกามากกว่า 40MHz โดยที่เราจัดรูปแบบ The spilt power supply, The clock double, The clock spectrum spreader และการจัดวางอุปกรณ์บน PC board
- Rabbit มีการประมวลโดยมีการกำหนดรูปแบบ การจัดาวางและมีความกระทัดรัดในเรื่องชุดคำสั่ง ที่มีส่วนช่วยเหลือ คือ ตัวคอมไพเลอร์(Complier) และ ไลเบอรี่(Library) ที่มีกฎจำนวนมาก มีการเตรียมRabbit สำหรับ 186,386,8051 และ ez80

- หน่วยความจำของ Rabbit ได้การยอมรับเรื่องประสิทธิภาพและง่ายต่อการออกแบบ โดยไม่ต้องอาศัยอุปกรณ์ภายนอกที่ต้องการของ Static Ram , Battery-backed external Memory ได้ถูกสนับสนุนจาก Built-in functionality ในขณะที่มีการลดพลังงานมี นาฬิกาจัดการเกี่ยวกับหน่วยความจำที่
- Rabbit มีบัสภายนอกใช้ 2 ช่วงสัญญาณนาฬิกา สำหรับในการอ่าน และ 3 ช่วง สัญญาณนาฬิกา สำหรับการเขียน ซึ่งเป็นข้อได้เปรียบ เมื่อเปรียบเทียบกับการ ออกแบบสัญญาณนาฬิกาเดี่ยว โดยกล่าวถึงข้อได้เปรียบว่าง่ายต่อการออกแบบเพื่อ หลีกเลี่ยง การแย่งบัสเขียนด้วย Solid data และการถือครองของข้อมูล การเลี่ยนแปลง ของผลลัพธ์ของหน่วยความจำที่สามารถใช้เวลามากกว่า1/2ของรอบการทำงานบัส ความสามารถใช้ Asymmetric Clock ที่ถูกโดย Double clock เป็นข้อได้เปรียบถ้า เปรียบเทียบกับระบบสัญญาณนาฬิกาเดี่ยว มีการทำงานโดยใช้มีการเพิ่มความเร็วบัส เป็น 2 เท่า ไม่สามารถเป็นความจริงได้ ในความเป็นจริงของหน่วยความจำ นอกจาก ได้รับการสนับสนุนจาก Fast-cache แรม
- Rabbit สามารถทำงานได้ที่ 3.6 V หรือน้อยกว่านี้ แต่สามารถทำงานได้ที่ 5V ขาเข้า และมี เส้นทางที่ 2 ของบัส (Second complete Bus) สำหรับการทำงานของ I/O ซึ่ง ได้มากจาก memory bus second auxiliary bus สามารถทำงานได้โดยแอพพลิเคชั่น เหมือนกับการออกแบบคำสั่ง ลักษณะพิเศษของระบบอีกอย่างคือการที่สามารถใช้งาน ได้โดยการออกแบบการใช้ 3V และ 5Vผสมกัน และสามารถหลีกเลี่ยงปัญหาที่ เกี่ยวกับการโหลด ปัญหาระบบ EMI ซึ่งมีผลมาจากหน่วยความจำของบัส ไปเชื่อมต่อ อุปกรณ์ I/O ที่เป็นจำนวนมาก
- Rabbit อาจจะโปรแกรมจากระยะทางไกลรวมไปถึงการ cold-boot เส้นทางการ เชื่อมต่อแบบอนุกรม นอกเหนือจากเส้นทางนี้ มีการเชื่อมต่อกับระบบเครือข่าย มีการ ใช้งานผ่านอินเตอร์เน็ตสามารถสร้าง และ ติดต่อกับ Rabbit โดยผ่านอินเตอร์เน็ต
- Rabbit 3000 สามารถติดต่อกับอุปกรณ์ภายนอกได้จำนวนมาก เพื่อสำหรับการแข่งขัน กับตัวประมวลผลตัวอื่นๆ

Rabbit มีขนาด 8 บิตในการประมวลผล โดยมี 8 บิต ดาต้าบัสภายนอก และ 8 บิตดาต้าบัส ภายใน เพราะการทำงานส่วนใหญ่ จะทำงาน 8 บิตบัสภายนอก และเพื่อใช้ชุดคำสั่งให้มี ความกระทัดรัด การทำงานหลายอย่างมีประสิทธิภาพพอกับ 16 บิต

2.1.2.2 แนะนำส่วนต่างๆของชิพและความสามารถ

ส่วนประกอบภายนอกของ ชิพอุปกรณ์ที่มีประโยชน์และมีขนาดเล็กโดยเป็นระบบ embedded จะประกอบด้วย พอร์ทอนุกรม, ระบบสัญญาณนาฬิกา, เครื่องกำเนิดเวลา/วัน, ขนานI/O, พอร์ทลูก(slave port), การถอดรหัส monitor , ไทเมอร์ เป็นต้น

2.1.2.3 5 V Tolerant อินพุตท์

Rabbit 3000 มีการทำงานโดยมี โวลล์ อยู่ระหว่าง 1.8V-3.6V แต่ Rabbit3000 ส่วนใหญ่สามารถรับ 5 V อินพุตท์ ยกเว้น ขาตัวจ่ายไฟ กับขาบัฟเฟอร์ออสซิเลเตอร์ ในขณะที่สัญญาณมีขนาด 5 V ส่งไปยังขาTolerant ถ้ามีความต่างศักย์ มากกว่านั้น Rabbit จะปิดการทำงานทันที ข้อได้เปรียบของ 5V tolerant คือการให้อุปกรณ์ ที่มีขนาด 5V สามารถทำงานได้โดยผ่าน suitable switching ก่อนที่จะไปติดต่อโดยตรงกับ rabbit กล่าวโดยรวมว่า HCT มีการทำงานที่ 5V และมีอินพุตท์ที่สามารถเข้าได้ต้องมีความต่างศักย์อยู่ระหว่าง 0.8-2 V

2.1.2.4 พอร์ทอนุกรม

มี 6 พอร์ท คือ A, B, C, D, E และ F มีทั้งหมด 6 พอร์ทสามารถทำงานได้บนโหมดอะซิงโคนัสซึ่งมีขนาดอัตราในการส่งเท่ากับสัญญาณนาฬิกา หาร 8 โดยที่พอร์ทของอะซิงโครนัสใช้ 7 บิตหรือ 8 บิต โดยที่มีพาริตี้บิต หรือไม่มีก็ได้ โดยให้บิตที่ 9 เป็น แอดเดรส โดยมีการตั้งค่าหรือลบเพื่อแสดงที่ไบต์แรกของ ข้อความ เมื่อต้องการความช่วยเหลือ

โปรแกรมพอร์ทอนุกรม สามารถบอกข้อมูลจากเอาพุตท์ชิพรีจิสเตอร์ ขณะที่ ไบต์สุดท้ายได้มาถึง เป็นสิ่งที่สำคัญมากสำหรับ RS-489 เพราะว่าการสื่อสารแบบทางเดียว (Half- duplex) ไม่สามารถติดต่อได้โดยตรงจนกระทั่งบิตสุดท้ายได้ถูกส่งออกไป เราจะเรียกว่า แอดเดรสบิต(address bit) หลังจากดาต้าบิตสุดท้ายถ้าแอดเดรสบิต มักจะถูกติดตามด้วย high stop bit ปกติเราใช้สำหรับการส่ง 2 บิตหยุด หรือ พาริตี้บิต ถ้ามีความต้องการส่งโดยตรง

พอร์ทอนุกรม A, B, C, D สามารถทำงานที่สัญญาณนาฬิกาในโหมดอนุกรมได้ในการทำงานโหมดนี้ สัญญาณอะซิงโครนัสจะควบคุมข้อมูลที่เข้าออกโดยที่ พอร์ทอนุกรมของ Rabbit หรือ อุปกรณ์ ทางไกลที่ช่วยเหลือการหาสัญญาณนาฬิกา ในขณะที่ Rabbit จัดเตรียมสัญญาณนาฬิกานั้นมีอัตรในการส่งอาจจะสูงขึ้นเป็น ½ ของระบบความถี่สัญญาณนาฬิกาในขณะที่สัญญาณนาฬิกาจัดการโดยอุปกรณ์ ชนิดอื่นๆ มีค่ามากที่สุดของข้อมูล จะถูกระบบ แบ่งออกเป็น 6 ส่วน

จำเป็นต้องที่เวลา จัดหาสัญญาณนาฬิกาแบบภายนอกด้วยสัญญาณนาฬิกาแบบภายในบน โหมดอนุกรม บางทีอาจจะใช้ สนับสนุน " SPI bus device"

พอร์ทอนุกรม A มีความสามารถพิเศษ คือสามารถทำให้ระบบ cool-boot ได้หลังจากการตั้งค่า พอร์ทอนุกรม A เป็น พอร์ทปกติที่ใช้สำหรับพัฒนาสำหรับการพัฒนาโปรแกรมโดย Dynamic C

ทุกๆ พอร์ทอนุกรม มี ไทม์เมอร์โหมดพิเศษ ไว้เพื่อการติดต่อแบบมาตราฐานการสื่อสารข้อมูล

2.1.2.5 ระบบสัญญาณนาฬิกา

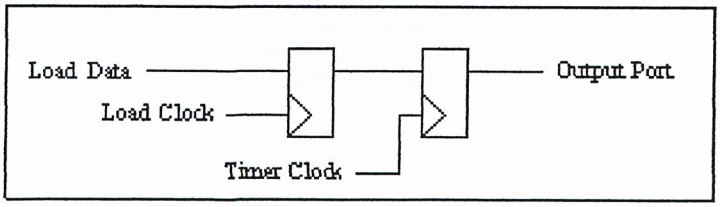
ออสซิเรเตอร์ ตามปกติใช้คริสตัลภายนอก ชนิดของความถี่ซึ่งอยู่ระหว่าง 1.8-26 MHz การประมวลผลสัญญาณนาฬิกา ถูกกำหนดจากเอาพุตท์ของ ออสซิเรเตอร์ โดยมีการ2เท่าของความถี่การใช้ความถี่โดยตรง หรือการหารด้วย 2, 4 , 6, 8 การประมวลผลสัญญาณจะมีค่าเท่ากับ 32.768MHz สัญญาณนาฬิกาแบบต่อเนื่อง(real-time) ออสซิเรเตอร์ สำหรับการทำงานที่ต้องการกำลังต่ำ โดยปกติ ออสซิเรเตอร์ สามารถปิดภายใต้การควบคุมของซอฟแวร์

2.1.2.6 32.768KHz ออสซิเรเตอร์ Output

32.768 KHz ออสซิเรเตอร์ อินพุตท์ ได้ถูกออกแบบเพื่อที่จะรองรับ 32.768 KHz โดยแนะนำให้ใช้สัญญาณกำลัง"Tinny Logic" ซึ่งเป็นส่วนที่มีการสนับสนุนทางด้านเอกสารและมีราคาที่ถูก 32.6768 KHz ถูกใช้สำหรับ แบตเตอรี่ -backable 48บิตเคาเตอร์ ซึ่งมีหน้าที่รักษาแบบ real – time clock (RTC) ซึ่งสามารถอ่านและตั้งค่าได้ด้วยโปรแกรม

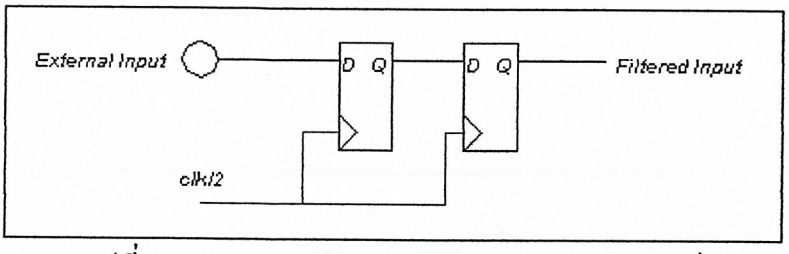
2.1.2.7 พอร์ทขนาน I/O

มีทั้งหมด 56 พอร์ทขนาน I/O โดยแบ่งออกเป็น 7 เส้น แต่ละเส้นมีขนาด 8 บิต โดยมีจุดมุ่งหมาย A-G ส่วนใหญ่ของพอร์ทมีหลายฟังก์ชั่น แต่ละ ดาต้าอนุกรม หรือ ชิพ เลือก strobe ส่วนขนาน D, E, F มีความสามารถในการทำไทเมอร์อะซิงโคนัส เอาพุตท์



รูปที่ 2.5 เอาพุตท์แคสเคส รีจิสเตอร์สำหรับ พอร์ทขนาน D และ E

โดยที่ strobe ไปยังพอร์ทซึ่งถูกโหลดเก็บไว้ในรีจิสเตอร์เลเวลแรก แล้วรีจิสเตอร์จะถูกส่งไปยังเอาพุตท์รีจิสเตอร์แล้วเลือกเวลาของสัญญาณนาฬิกา โดยสัญญาณนาฬิกาถูกเลือกโดยที่เอาพุตท์ของไทเมอร์ A1,B1,B2 หรือ สัญญาณนาฬิการอบข้าง(หารด้วย2) สัญญาณไทเมอร์เป็นข้อได้เปรียบในการสร้างฟิลส์ที่สามารถควบคุมได้ ซึ่งขอบเขตได้ถูกวางตำแหน่งเรียบร้อยแล้วและที่ความถูกต้องสูง ในเวลานั้น แอพฟลิเคชั่น(Application) นี้จะประกอบไปด้วยสัญญาณการติดต่อสื่อสาร , ขนาดความกว้างของฟิลส์มอลดูเลชั่นและการควบคุม stepping motor

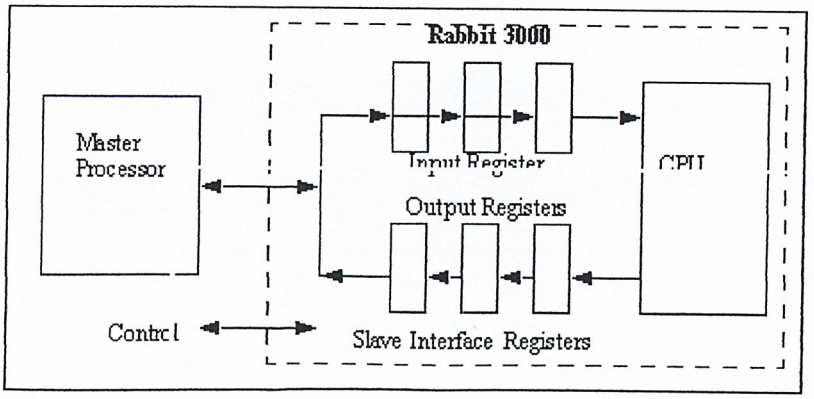


รูปที่ 2.6 การกรองสัญญาณดิจิตอลของขาอินพุตท์

ขาอินพุตท์ส่งไปยังพอร์ทขนานจะถูกกรองด้วยแคสเคส D ฟลิป-ฟล็อป ซึ่งจะป้องกันฟิลส์ที่มาขนาดเล็กแล้ว สัญญาณนาฬิกาที่อยู่รอบมาจาก รีค็อกไนส์, อะซิงโครนัส , ฟิลส์ภายนอก ไปยังฟิลส์ภายใน และหลีกเลี่ยงปัญหา Meta stability

2.1.2.8พอร์ทลูก (Slave Port)

พอร์ทลูกได้ถูกออกแบบให้มาใช้กับโปรเซสเซอร์ตัวอื่นๆ หรือไม่ก็ใช้กับ Rabbit ตัวอื่นๆ มีพอร์ทสามารถแบ่งได้ด้วย พอร์ทขนาน A และ พอร์ทข้อมูล bidirectional โดยที่ ตัวแม่ สามารถอ่านรีจิสเตอร์ทั้ง 3 รีจิสเตอร์ก่อนแล้วค่อยเลือกเส้นทางได้มาจาก รีจิสเตอร์แอดเดรสและอ่านได้มาจาก strobe เป็นสาเหตุ ที่ทำให้รีจิสเตอร์ สามารถมีเอาพุตท์ด้วยพอร์ทได้โดยที่รีจิสเตอร์ตัวเดิมสามารถเขียนเหมือนกับ รีจิสเตอร์ I/O โดย Rabbit ตัวลูก ที่เพิ่มอีก 3 เส้นทางจะเอามาทำเป็นเส้นทางย้อนกลับ



รูปที่ 2.7 เส้นทางข้อมูลของพอร์ทลูก

Rabbit ลูก สามารถอ่านรีจิสเตอร์ได้ด้วยรีจิสเตอร์ที่เหมือนกัน เหมือนกันกับ I/Oรีจิสเตอร์ ในขณะที่ข้อมูลเข้ามาเขียนลงรีจิสเตอร์ แล้วส่งไปยังรีจิสเตอร์อีกตัวหนึ่ง ในขณะที่ตัวแม่สามารถให้ตัวลูกอ่านข้อมูลที่เข้าไป ส่วนสายอื่นๆจะบอกให้ตัวแม่ จะบอกการไหลของข้อมูลและยังไม่ได้อ่านจาก ตัวแม่ พอร์ทลูกสามารถส่งย้อนสัญญาณไปยังตัวแม่เพื่อที่จะจัดรูปแบบ เพื่อใช้ในการติดต่อสื่อสารบนพอร์ทลูก

2.1.2.9 Auxiliary I/O Bus

Rabbit 3000 มีชุดคำสั่งที่ช่วยเหลือการเข้าถึงหน่วนความจำและการเข้าถึง I/O การเข้าถึงหน่วยความจำจะมีพื้นที่ประมาณ 1 MB ส่วน I/O จะมีพื้นที่ว่างประมาณ 64k I/O ทั่วไปการออกแบบตัวประมวลผลโดยใช้ แอดเดรส และ แถวข้อมูล ชุดเดียวกัน โดยใช้สำหรับหน่วยความจำและ ที่ว่างของ I/O การแชร์ข้อมูลและแอดเดรส บ่อยครั้งมีการแย่งชิงกันหรือออกแบบโดยมีเงื่อนไขที่ซับซ้อน บัสของหน่วยความจำส่วนใหญ่ จะมีเวลาวิกฤต (Critical Time) และความสามารถในการโหลดถูกกำหนดโดยแชร์ข้อมูลด้วยบัส I/O

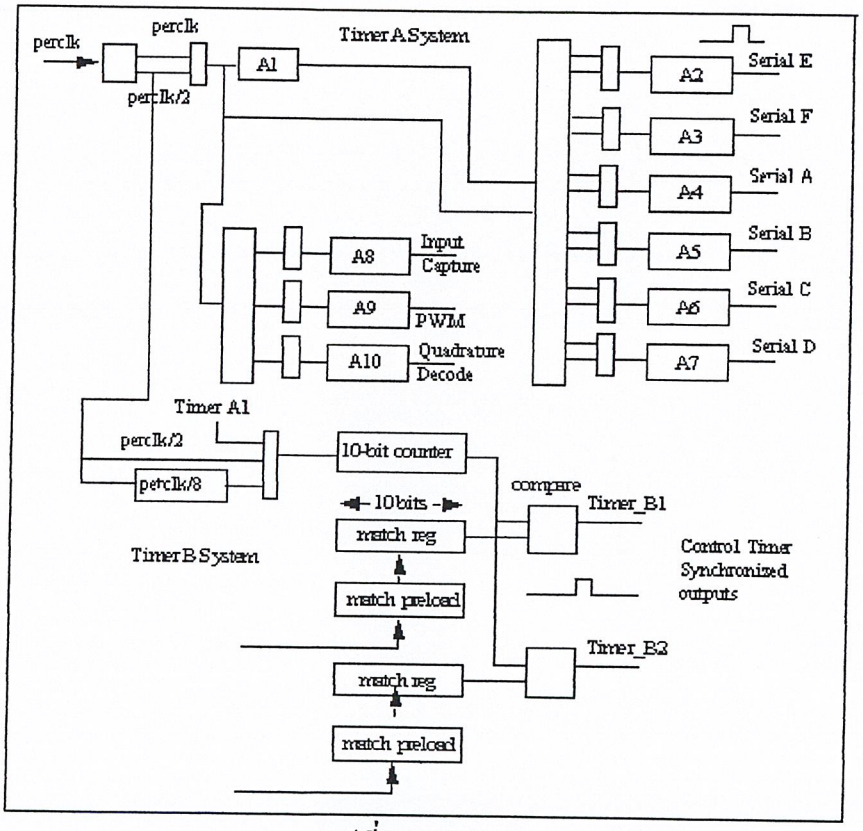
Rabbit 3000 มีคำสั่งที่ใช้ในการแบ่งบัส สำหรับ I/O และหน่วยความจำของ Auxiliary I/O บัส ถูกใช้ขาเดียวกันโดยพอร์ทลูก พอร์ทขนาน A ถูกเตรียมไว้สำหรับ 8 bidirectional แถวข้อมูลขนาน B ขนาด 2:7 บิต เตรียมไว้ 6 แถวแอดเดรส ต้องใช้ในการส่งสัญญาณอย่างน้อย 6 เส้นจาก 16 เส้น เป็นการใช้พื้นที่ I/O เต็มที่ Auxiliary บัส โดยปกติจะทำงานบนวนรอบการทำงาน I/O ชิพเลือก เช่นเดียวกับ การอ่านและเขียน strobe ซึ่งมีหลายค่าที่ขาต่างๆดังนั้นการรวมแถวแอดเดรสสามารถเป็นบัฟเฟอร์ได้

การติดต่ออุปกรณ์ I/O กับ auxiliary bus จะช่วยเพิ่มความสามารถในการโหลดสำหรับ RCM 3000 มีประมาณ 2-3 ขา ที่ต้องการการติดต่อจาก core module ตั้งแต่พอร์ทลูกและI/O บัสสามารถแชร์ ขาได้ และบัสหน่วยความจำ ไม่มีความต้องการมากเพื่อเตรียมพร้อมไว้สำหรับความสามารถ I/O เพราะ I/O มีความสามารถน้อยและมีความเร็วน้อยกว่า บัสหน่วยความจำ ในอนาคตสามารถทำงานโดยปราศจาก EMI และปัญหาการสะท้อนของสัญญาณ 5 V ซึ่งสามารถไปบน I/O บัส ตั้งแต่ อินพุตท์Rabbit 3000 สามารถรับ 5 V ซึ่งสัญญาณนี้ทำได้ง่าย

2.1.2.10 Timers

Rabbit มีระบบ timer หลายระบบโดยมีสัญญาณอินเตอร์รัพคาบ ซึ่งมาจาก 32.768 KHz ออสซิเรเตอร์ หารด้วย 6 จะมีอินเตอร์รัพทุก 448 ไมโครวินาที ถ้าเป็นไปได้ มีจุดมุ่งหมายตามความ

ต้องการของสัญญาณอินเตอร์รัพ Timer A ประกอบด้วย 10 countdown ขนาด 8 บิตและสามารถรีโหลดรีจิสเตอร์ซึ่งสามารถแคสเคส ขึ้น 2 ขั้น แต่ละครั้ง รีจิสเตอร์ countdown สามารถตั้งค่าโดยแบ่งตามจำนวนเลขซึ่งอยู่ระหว่าง 1-256 เอาพุตท์จาก 6 ค่า ไทเมอร์ซึ่งเตรียมค่าไว้ อัตราการส่งสัญญาณนาฬิกาสำหรับรีจิสเตอร์พอร์ทอนุกรมจำนวนหนึ่งจาก รีจิสเตอร์ทั้งหมด เป็นสาเหตุที่ทำให้เกิดอินเตอร์รัพและสัญญาณนาฬิกาไทเมอร์ซิงโครไนส์ พอร์ทเอาพุตท์ขนาน Timer B ประกอบด้วย 10 บิตเคาเตอร์ ซึ่งไม่สามารถเขียนได้แต่อ่าน มี 2 รีจิสเตอร์โดยที ขนาด 10 บิตและเปรียบเทียบ ถ้า match รีจิสเตอร์สามารถเข้าได้กันเคาเตอร์และฟัลส์ เป็นค่าเอาพุตท์ที่ออกมา ดังนั้นไทเมอร์สามารถถูกโปรแกรมไปยังเอาพุตท์ที่เป็นฟัลส์เพื่อจะตัดสินใจนับในอนาคต ฟัลส์ถูกสร้างมาจากสัญญาณนาฬิกาของไทเมอร์ซิงโครไนส์ รีจิสเตอร์เอาพุตท์พอร์ทขนาน เหมือนกับสาเหตุที่มาจากอินเตอร์รัพซึ่ง Time B สะดวกในการสร้างและมีความเที่ยงตรงในเรื่องของเวลา เพราะว่าได้ถูกโปรแกรมส่วนควบคุมเอาไว้



รูปที่ 2-8 Timer

2.1.2.11 Input Capture Channels

การเข้าช่องสัญญาณซึ่งใช้ในการตัดสินใจนั้นขึ้นอยู่กับเวลาและการไปยังสถานที่เราต้องการ ในกรณีที่ถูกส่งสัญญาณ โดยสัญญาณที่ขอบขาขึ้น หรือขอบขาลง ขาใดขาหนึ่งบนขาทั้งหมด 16 ขา สามารถเลือกขา อินพุตท์ สำหรับช่องสัญญาณใด ช่องสัญญาณหนึ่ง ของ A มีขนาด

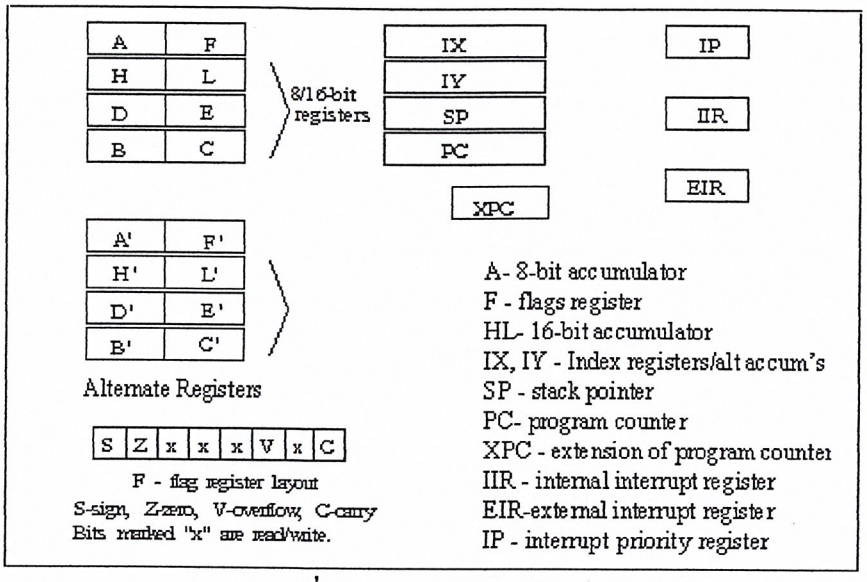
16 บิต เคาเตอร์ ใช้สำหรับเก็บข้อมูลของไทเมอร์ซึ่งจะเก็บไว้ที่เคาเตอร์จะถูกควบคุมโดยเอาพุตท์ไทเมอร์ A8 และสามารถตั้งค่า เคาเตอร์ที่อัตราเร็วจากสัญญาณนาฬิกาเต็มรูปแบบ ซึ่งมีความเร็วเต็มที่ของสัญญาณนาฬิกาคือ 1/256 ความเร็ว สัญญาณนาฬิกา

ซึ่งมี 2 สถานะคือ เปิดและปิด สถานะเปิดจะใช้ในการเริ่มการนับและหยุดการนับ แต่อย่างไร เคาเตอร์อาจจะทำงานต่อไปเรื่อยๆ จนกระทั้งอยู่ในสถานะหยุดถูก encounted ทั้งเริ่มและหยุดอาจจะใช้ latch กับเวลาในขณะที่ทำงานอยู่ ซึ่งสถานะที่เกิดขึ้นอย่างรวดเร็วค่อนข้างที่จะแน่นอนมากกว่า เริ่มหรือหยุดเคาเตอร์ในขาเดียวกัน อาจจะเกิดการตรวจสอบสถานะเริ่มและหยุด ตัวอย่าง สถานะเริ่มและหยุดอาจจะเป็นอินพุตท์ที่มาจากการแจกไปยังขาต่างๆ

อินพุทต์จะเลือกช่องสัญญาณสามารถวัดมาตรฐานได้จากความกว้างของความเร็วฟัลส์ ถูกทำโดยการเริ่มการทำงานของเคาเตอร์บนลูกแรกของฟัลส์และสามารถเก็บค่าได้จาก ลูกที่ 2 ของฟัลส์ ในกรณีนี้ค่าผิดพลาดมากที่สุดในการวัดมาตราฐานมีค่าใกล้เคียงกัน คาบที่ 2 ของสัญญาณนาฬิกาซึ่งถูกใช้ไปนับ เคาเตอร์ ถ้ามีเวลาที่เหมาะสมซึ่งอยู่ระหว่างช่วงเวลาสำหรับการอินเตอร์รัพในช่วงเวลานั้น เคาเตอร์จะถูกเก็บค่าเอาไว้ ในกรณีนี้ค่า หยุดและเริ่มจะถูกยกเลิกการติดต่อกับการเริ่มต้นหรือการหยุดของเคาเตอร์ และปกติกลายเป็นการเก็บสถานะซึ่งอาจจะเจาะจงที่เคาเตอร์สามารถเคลียร์ และเริ่มได้โดยที่อยู่ภายใต้การทำงานของซอฟแวร์ที่ควบคุมและค่าที่เก็บเอาไว้มีการตอบสนองในรูปอิพุตท์

2.1.3 Processor register

รีจิสเตอร์ของ Rabbit มีลักษณะใกล้เคียงกับ Z18 หรือ Z80 ซึ่งมีรีจิสเตอร์แบบใหม่เพิ่มขึ้นมาคือ XPC และ IP โดยที่มีรีจิสเตอร์ EIR ซึ่งยังคงเหมือนกับ Z180 ซึ่งถูกใช้เป็นบอกข้อมูลเกี่ยวกับเวกเตอร์ของอินเตอร์รัพสำหรับในการอินเตอร์รัพ ส่วนรีจิสเตอร์ IIR เกิดขึ้นที่ตำแหน่งเดียวกับ logical ในชุดคำสั่ง เหมือนกับ Z80 รีจิสเตอร์แต่มีบางฟังก์ชั่นชี้ไปยังตารางของอินเตอร์รัพเวกเตอร์ สำหรับในการสร้างอินเตอร์รัพ



รูปที่ 2-9 Rabbit รีจิสเตอร์

ตัวประมวลผลของ Rabbit มีตัวประมวลผล 2ตัว คือ A รีจีสเอตร์จะเก็บไว้ในการคำนวณ 8 บิตในการจัดการ 8 บิต เช่น and หรือ or รีจิสเตอร์ HL ซึ่งมีขนาด 16 บิตจะถูกเก็บไว้สำหรับการประมวลผล 16 บิตคำสั่งการทำงาน ADD HL,DE ซึ่งเป็นการรวมกันของรีจิสเตอร์ 16 บิต DE ไปยัง รีจิสเตอร์ HL สำหรับการทำงานหลายอย่าง IXและ IY สามารถทำหน้าที่แทนสำหรับ HL

รีจิสเตอร์สามารถทำให้ F ซึ่งเป็นค่ารีจิสเตอร์หรือ สถานะของรีจิสเตอร์จะมีค่าซึ่งเตรียมข้อมูลเกี่ยวกับการทำงานครั้งสุดท้ายค่ารีจิสเตอร์ไม่สามารถเข้าถึงได้ หน่วยความจำได้โดยตรง แต่งตั้ง ใช้คำสั่ง POP AF และ PUSH AF โดยปกติแล้วค่าจะถูกตรวจสอบโดยคำสั่ง Jump Flag ซึ่งแสดงการทำงานด้านคณิตศาสตร์ และการทำงานด้ายลอจิก ซึ่งรวมไปถึงกฎ ของแต่ละคำสั่ง มี 4 บิตที่ไม่ได้ใช้ สำหรับอ่าน/เขียนในค่ารีจิสเตอร์ซึ่งจะเก็บค่าต่างๆและเส้นทางของคำสั่ง PUSH AF และ POP AF แต่บิต ควรใช้อย่างระมัดระวังตั้งแต่ Rabbit สามารถใช้บิต พวกนี้ในเรื่องอื่นด้วย

รีจิสเตอร์ IX , IY และ HL สามารถเก็บไว้ในindex ซึ่งก็บค่าพอยเตอร์ที่อยู่ของหน่วยความจำสำหรับข้อมูลสามารถส่งข้อมูลและเก็บข้อมูล จนกระทั่ง Rabbit มีแอดเดรสจำนวนหลาย MB หรือมากพอสำหรับการเก็บข้อมูลอินเด็กรีจิสเตอร์สามารถเข้าถึงแอดเดรสได้โดยตรงประมาณ 64 k ของหน่วยความจำ ขอบเขตตำแหน่งจะมีผลมาจากอุปกรณ์การแมพผิ่งของหน่วยความจำและคำสั่งพิเศษบางคำสั่งสำหรับบางโปรแกรมใช้หน่วยความจำประมาณ 64 K ซึ่งเพียงพอต่อการใช้งาน

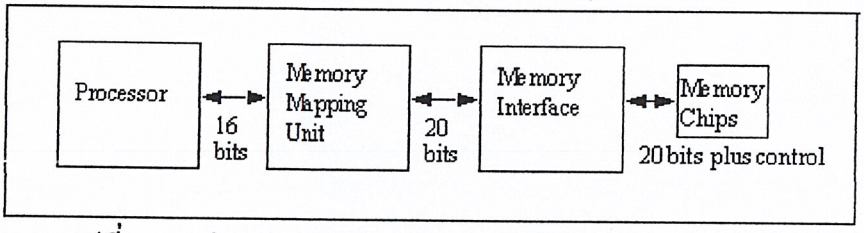
โดยให้ รีจิสเตอร์ Sp จะเป็นตัวชี้สแตก (stack) ซึ่งใช้สำหรับการหาเส้นทางและการเชื่อมต่อการอินเตอร์รัพ เช่นเดียวกับการเก็บข้อมูลทั่วไป

ข้อได้เปรียบของ Rabbit คือการมีรีจิสเตอร์ให้เลือกเป็นชุดคำสั่งพิเศษ 2 ชุดที่ทำหน้าที่พิเศษในการสลับของ รีจิสเตอร์ที่ถูกเลือก กับรีจิสเตอร์ทั่วไป การติดต่อระหว่างที่รีจิสเตอร์ถูกเลือกในสมัย Z80 นั้นเป็นไปได้ยากเพราะว่า คำสั่งการแลกเปลี่ยนมีไว้สำหรับค่ากลางที่เกี่ยวกับการติดต่อสื่อสารระหว่างรีจิสเตอร์ทั่วไป กับรีจิสเตอร์ที่ถูกเลือกเอาไว้โดยมีประสิทธิภาพมากขึ้นโดยการกำหนดตัวเลขให้แก่รีจิสเตอร์แต่ละตัวทำให้ง่ายต่อการเขียนโปรแกรม

2.1.3.1 Memory Mapping

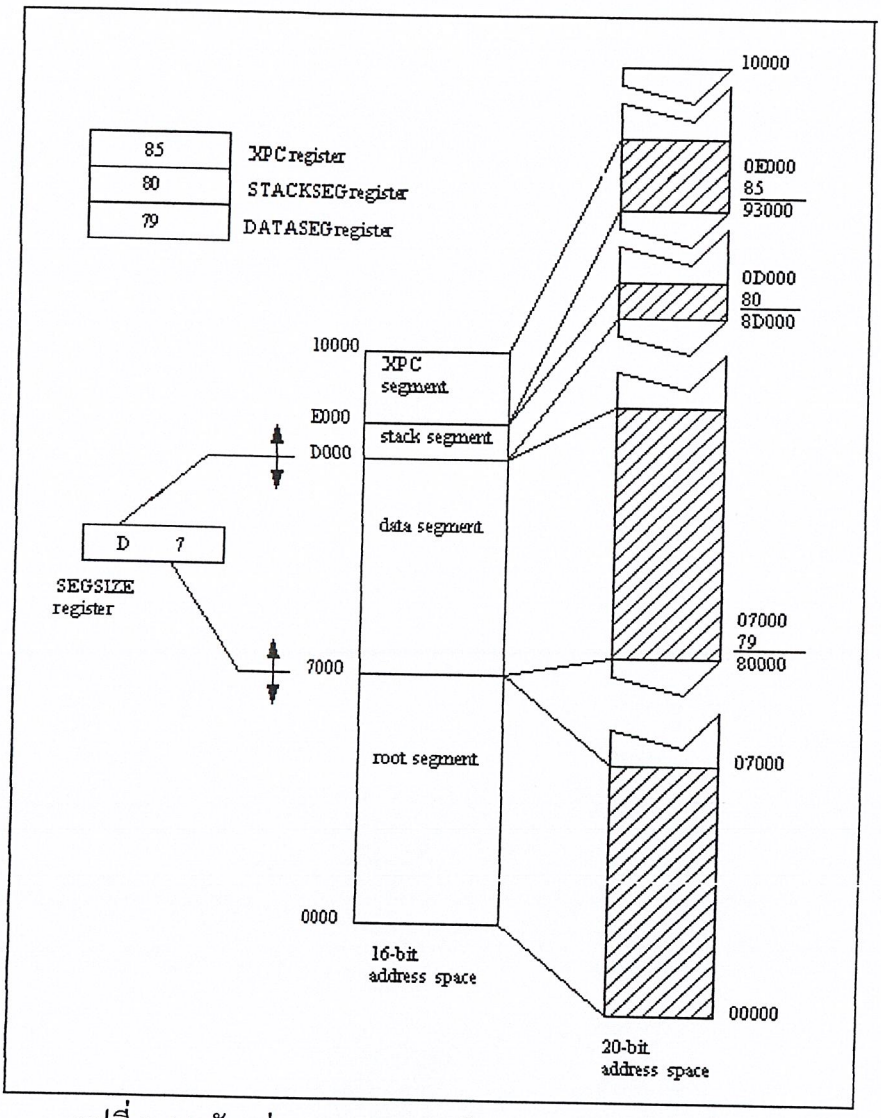
ชุดคำสั่งของ Rabbit มีการติดต่อกับที่ว่างของการเก็บข้อมูลโดยตรงซึ่งหมายความว่าที่อยู่ของคำสั่งจะมีความยาว 16 บิตและรีจิสเตอร์อาจจะใช้ในการชี้ตำแหน่งของข้อมูล,โปรแกรมเคาเตอร์และ สแตกพอยต์(SP) ซึ่งมีความยาว 16 บิต

เพราะชุดคำสั่งของ Rabbit ใช้แอดเดรสที่มีความยาว16 บิต ชุดคำสั่งที่สั้นกว่าและสามารถประมวลผลได้เร็วกว่า โดยที่ Rabbit มีการวางตำแหน่งของหน่วยความจำที่คล้าย Z180 แต่ไประสิทธิภาพมากกว่า โดยดูรูปที่ 3.2 จะแสดงถึงความสัมพันธ์ระหว่างส่วนต่างๆ



รูปที่ 2-10 ส่วนประกอบการจัดตำแหน่งของหน่วยความจำ

เราสามารถแบ่งหน่วยความจำออกเป็นส่วนๆ ได้ดังรูปที่ 2-11



รูปที่ 2-11 ตัวอย่างการของการทำงาน Memory Mapping

Root Segment เป็นการกำหนดตำแหน่งพื้นฐานของหน่วยความจำแบบแฟลชและบรรจุโค๊ตเริ่มต้น เช่นเดียวกับโค๊ตที่เกิดการเก็บข้อมูล

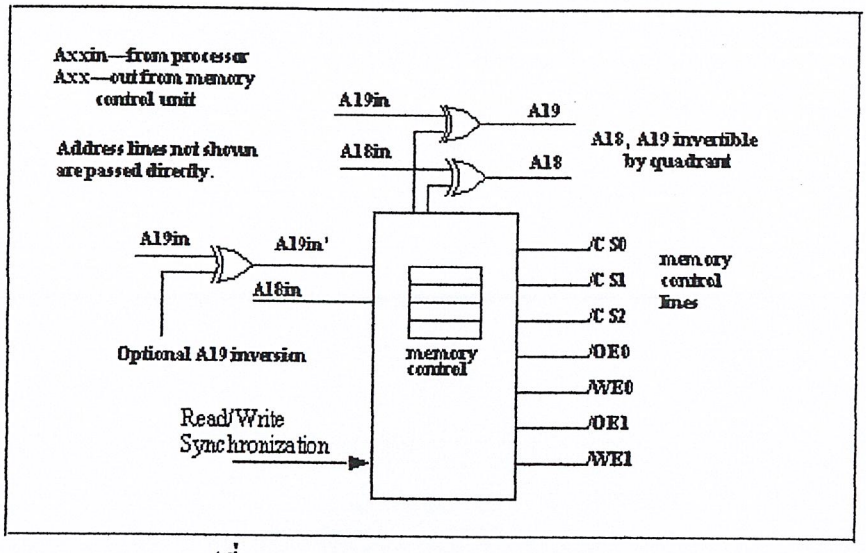
Data segment ใช้ในการเก็บหลายรูปแบบ โดยใช้ผลรวมของวิธีการต่างๆสำหรับการตั้งค่าบนหน่วยความจำ

Stack memory เป็นส่วนที่เก็บข้อมูลของระบบ โดยแต่ละสแตกจะมีขนาด 4 K

XPC segment โดยปกติใช้เป็นในการประมวลผลของโค๊ต ซึ่งไม่ได้เก็บข้อมูลใน root segment หรือ data segment ชุดคำสั่งพิเศษ จะได้รับการสนับสนุนจากการประมงลผลซึ่งได้เห็นใน XPC segment

The memory interface unit จะได้รับข้อมูลขนาด 20 บิตซึ่งถูกสร้างจาก MMU มีเงื่อนไขในการเปลี่ยน ที่แถว A16,A18,A19 แถวตำแหน่งอื่นๆสามารถเข้าได้เลย memory interface

unit จะควบคุมสัญญาณสำหรับข้อมูลภายนอก การเชื่อมต่อกับข้อมูลภายนอกโดยเลือก (/CS0,/CS1,/CS2), ข้อมูลที่ถูกส่งออก (/OE0,/OE1) และสามารถเขียนที่(/WE0,/WE1) แต่ละสัญญาณจะตอบสนองไปยัง แถวที่ควบคุมสามารถพบในหน่วยความจำแบบstaticซึ่งสร้างมาจากหน่วยความจำที่ควบคุมสัญญาณ The 20 บิต ตำแหน่งจะถูกแบ่งเป็น หนึ่งในสี่ของ 256k *bank control register* สำหรับแต่ละส่วนเพื่อที่จะใช้ในการตัดสินใจว่าจะเลือกใช้ตัวไหน และจะยอมให้เอาพุตท์ออกตัวไหนและจะให้ตัวไหนเขียนไปบนหน่วยความจำ



รูปที่ 2-12 แสดง memory interface unit

2.1.3.2 Extended Code Space

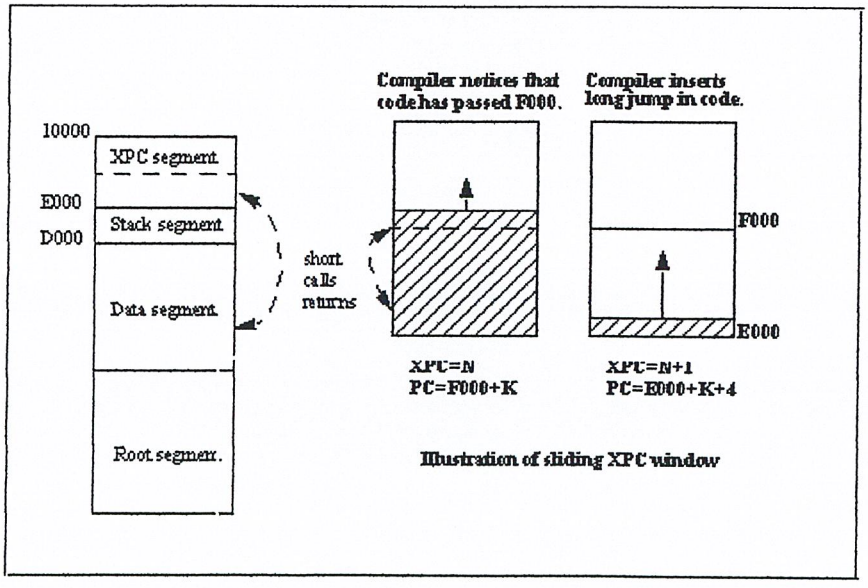
Rabbit จะใช้วิธีการ paging โดยใช้ความสามารถในการประมวลผลของโปรแกรมซึ่งมีโค๊ดอยู่ โดยความสามารถนี้จะไม่มีในการประมวลผลแบบ 16 บิตแอดเดรส และไม่สามารถสนับสนุนโดยZ180

Rabbit ใช้ในการ paging โดยใช้วิธีการ sliding page โดยแต่ละหน้า(page) จะมีขนาด 8K มันคือส่วนXPC 8 บิต XPC รีจิสเตอร์ จะเก็บหน้าต่างๆรีจิสเตอร์เฉพาะ ส่วนของหน่วยคามจำ ซึ่งเป็นที่หน้าต่าง(window)ชี้ ในขณะที่โปรแกรมจะถูกประมวลผลในส่วนของXPC , การกระโดดของ 16 บิต และการส่งค่ากลับโดยใช้ การเข้าถึงโค๊ด ในส่วนอื่นๆใน 3 segment ซึ่งอยู่ในที่ว่าง 16 บิต แอดเดรส ถ้ามีการควบคุมการถ่ายโอนออกจากหน้าต่างที่ต้องการ แล้วการ long jump, long call หรือ long return ได้ถูกใช้ ชุดคำสั่งได้ปรับเปลี่ยนโปรแกรมcounter(PC) และ XPC register สาเหตุหนึ่งที่ XPC window ชี้ได้ถึงความแตกต่างบนส่วนต่างๆของหน่วยความจำ ที่ตำแหน่งของ long jump, call หรือ return ที่ได้ถูกการติดตั้ง XPC segment จะมีขนาด 8k long โดยที่ส่วนของ XPC ถูก

ระบุตำแหน่งบนหน่วยความจำ 4K เพราะว่าหน้าต่างสามารถทำการ slid โดยขนาดครึ่งหนึ่งของหน้าต่าง สามารถเป็นไปได้ในการประมวลผลโดยไม่มีการใช้ ช่องว่างบนหน่วยความจำ

โดยตัวประเมินทำให้เกิดโค๊ดบน XPC window โดยที่หน้าต่าง จะถูกเลื่อนลงโดย 4k ในขณะที่โค๊ด ไปหลังจาก F000 การทำสำเร็จโดย long jump ซึ่งวางตำแหน่งอีกรอบหนึ่งโดยมีหน้าต่างที่มีขนาดน้อยกว่า 4k ตัวอย่างรูปที่ 2-13 ตัวประมวลผลไม่สามารถแสดงโดยไม่มีขอบเขตที่จบของหน้าได้ โค๊ดทั้งหมดจะถูกประมวลผลสำหรับ XCP window ที่มี 24 บิตตำแหน่งซึ่งมาจากการประกอบ 8 บิต XPC และ 16 บิตแอดเดรส Short jump และ การเรียกถูกใช้ ได้จัดเตรียมทรัพยากรและเป้าหมายคำสั่งทั้งสองมีเหมือนกับ XCP address โดยปกติในแต่ละคำสั่งจะใช้หน้าต่างยาวประมาณ 4K มีขนาด 16บิตแอดเดรส ระหว่าง E000+n และ F000+m ซึ่ง m และ n จะอยู่เหนือคำสั่งของ 24-28 ไบต์ แต่มีความสามารถเหมือน4096 ไบต์ ในแนวยาว ตั้งแต่หน้าต่างได้ถูกกำหนดให้ไม่เกิน 8k ตัวประมวลผลไม่สามารถประมวลผลได้ในข้อความเดียว หรือที่มีความต้องการมากกว่า 8k

โปรแกรมโค๊ดซึ่งยังอยู่ใน ส่วนของroot หรือส่วนของ XPC โปรแกรมโค๊ด อาจจะอยู่ในส่วนของข้อมูล โค๊ดสามารถประมวลผลในส่วนอขง สแตกแต่โดยปกติแต่จะถูกจำกัดในสถานะพิเศษ โค๊ดบนรูท หมายถึง ส่วนอื่นๆที่นอกเหนือส่วนของ XPC ซึ่งสามารถเรียกใช้โค๊ดต่างๆที่อยู่บน root โดยการใช้short jump และ call แต่อย่างไรก็ตาม การ long call อาจจะถูกใช้ในขณะที่โค๊ดใน XPC segment ได้ถูกเรียกใช้ ฟังก์ชั่นที่อยู่บนรูทจะมีประสิทธิภาพมากขึ้น เพราะการ long call และการ long return ต้องการ 32 clock สำหรับการประมวลผล และการ short return ต้องการ 20 สัญญาณนาฬิกาความแตกต่างกันที่ขนาด แต่สัญลักษณ์สำหรับ subroutines ที่สั้นๆ



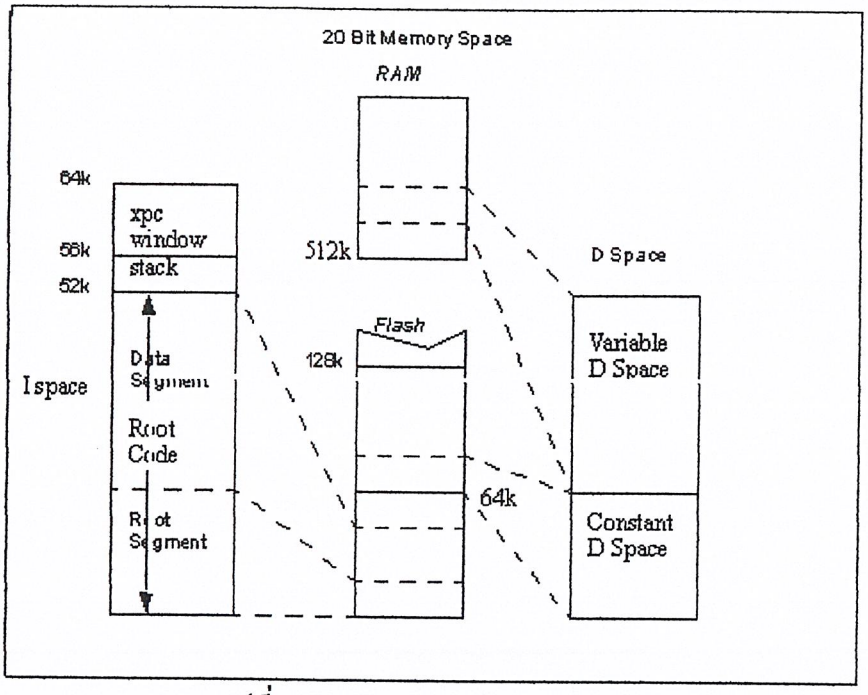
รูปที่ 2-13 Use of XPC Segment

2.1.3.3 Separate I and D space – Extending Data Memory

โดยปกติหน่วยความจำ จะมีพื้นที่ว่างประมาณ 64 K ไว้สำหรับรูทโค๊ต, the static และ XPC หน้าต่างแต่สามารถอัดได้โดยใช้พื้นที่ว่างประมาณ 40K หรือน้อยกว่านั้น โดยที่ XPC ต้องการ 8K สแตกมีความต้องการ 4K และระบบส่วนใหญ่ต้องการอย่าง 12k สำหรับ root code ส่วนที่เหลือจะเป็นจะเก็บไว้สำหรับ โปรแกรมย่อยๆ

การที่ใช้พื้นที่ว่างในแรมหรือหน่วยความจำแบบแฟลชนี้จะไม่มีการแมพลงไปในพื้นที่ว่าง 64K และแล้วการส่งผ่านข้อมูลที่ใช้ฟังก์ชั่นcall หรือการใช้ภาษาAssembly นั้นจะใช้คำสั่ง LDP ซึ่งเป็นคำสั่งทีเข้าถึงตัวหน่วยความจำขนาด 20บิตแอดเดรส เป็นการเข้าถึงพื้นที่ส่วนที่ว่างของข้อมูลแต่คำสั่งนี้จะมีประสิทธิภาพต่ำกว่าคำสั่งส่งผ่าน 64K โดยการใช้ 16 บิตแอดเดรส

Rabbit 3000 สนับสนุนการแยกของ I และ D หรือคำสั่งและพื้นที่ว่าง ในขณะที่มีการกระจายของ I และ D ก็จะสามารถทำการปรับปรุงสำหรับตำแหน่งบน ส่วนของroot หรือส่วนของ data แยก I และ D space มีความหมายว่าการทำงานเกิดความแตกต่างระหว่าง การส่งคำสั่งที่จากหน่วยความจำ และการส่งหรือเก็บข้อมูลในหน่วยความจำ ในขณะที่แยก I และ D space ทำให้เกิดการรวมกันระหว่างรูทและ ข้อมูลโดยปกติแล้ว 52 k ไบต์สำหรับ root code บน I space ในที่ว่าง D ส่วนของ root code นั้นจะเป็นส่วนหนึ่งของD space โดยปกติจะใช้สำหรับ ค่าคงที่ของข้อมูลที่จะทำการแมพลงไปในหน่วยความจำแบบแฟลช ในขณะที่ส่วนของข้อมูลนั้นจะเป็นส่วนหนึ่งของ D space มีค่าของข้อมูลที่จะทำการแมพลงไปในแรม การแยก I และ D space นี้จะต้องเพิ่มค่าของรูทโค๊ต และ รูทดาต้า เพราะว่า ค่าทั้งสองยาวไม่เพียงพอกับการแบ่งข้อมูลบนหน่วยความจำเดียวกันในขณะที่มีการแบ่งตำแหน่งเดียวกัน



รูปที่ 2-14 Separate I and D Space

โดยปกติการแยกระหว่าง I และ D space นี้จะมีการทำงานดังรูปที่ 2-14 ใน I space ส่วนของ root และส่วนของข้อมูลจะถูกรวมกันเป็น single root code ซึ่งใน D space ส่วนนี้ จะแบ่งไว้สำหรับแฟลชและแรมไว้เตรียมพร้อมกับ การเก็บสำหรับค่าของข้อมูลและข้อมูลที่สำคัญ อุปกรณ์สำเร็จในการแยก 20 บิตสำหรับ D space ตรงกันข้ามอย่างใดอย่างหนึ่งระหว่าง A16 กับ A19 สำหรับการเข้าของข้อมูล โดยที่version อาจจะถูกเจาะจงสำหรับส่วนของรูท และส่วนของข้อมูล โดยที่ปกติแล้ว A16 มีการย้อนกลับของข้อมูลจากคาต้าแอดเดรสในส่วนของรูท เป็นสาเหตุทำให้การเข้าถึงข้อมูลไปยังรูทต้องย้ายไปมากกว่า 64k ส่งไปยังส่วนของแฟลชโดยเริ่มต้นที่ 20 บิตจะเก็บ 64k สำหรับค่าคงที่ของข้อมูล A19 โดยปกติแล้วจะส่งย้อนกลับสำหรับการเข้าถึงของข้อมูล ไปยังข้อมูล เป็นสาเหตุให้ การเข้าถึงของข้อมูลในส่วนของข้อมูลได้ย้ายไปยังตำแหน่ง 512k หรือมากกว่านั้น ใน 20 บิตที่ว่างในตำแหน่งโดยปกติจะถูกแมพส่งไปยังแรมโดยที่ส่วนของสแตก และ XPC ไม่ได้ถูกแบ่ง I และ D space และการเข้าถึงหน่วยความจำแบ่งระหว่าง I และ D space

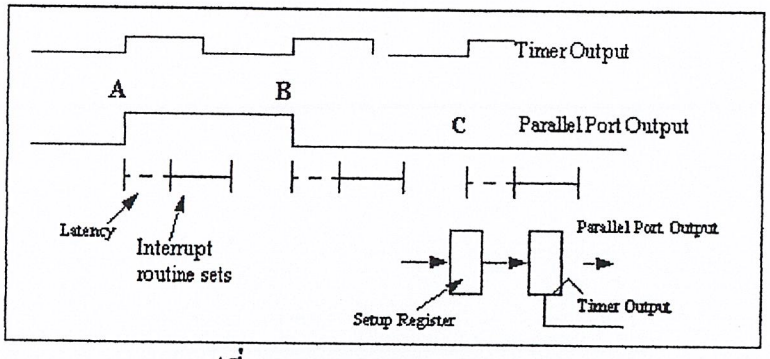
เป็นข้อได้เปรียบของการมี root code space ในการทำงานที่เร็วขึ้น เพราะ short call จะใช้ 16 บิตแอเดรส ในการcall โดยรวมการเปรียบเทียบสำหรับ long call ซึ่งมีขนาด 20 บิตแอดเดรส สำหรับโค๊ตภายนอก ข้อมูลที่อยู่ในรูทมีการเข้าออกได้ง่ายขึ้น เพราะการเปรียบเทียบระดับของคำสั่งว่ามีค่าแค่ไหนสำหรับการเข้าถึงข้อมูลใน 20 บิตที่ว่าง ส่วนใหญ่ของ overhead นำไปใช้ในถ่ายเทไปยัง 20 บิตในการประมวลผล ซึ่งจะใช้ 8 และ 16 บิตรีจิสเตอร์

2.1.4 Rabbit Capabilities

ในเรื่องจะกล่าวเกี่ยวกับความสามารถของต่างๆที่น่าสนใจของ Rabbit ซึ่งอาจจะไม่ชัดเจนจาก technician description

2.1.4.1 Precisely Timed Output Pulse

The Rabbit สามารถสร้างเอาพุตท์อย่างแม่นยำ ภายใต้การควบคุมของโปรแกรมผลกระทบของ interrupt เราจะไม่สนใจเพราะ อินเตอร์รัพ(Interrupt) ส่วนใหญ่ได้เตรียมพร้อมกับช่วงของ pulse ในอนาคตเป็น clock รีจิสเตอร์บน clock ถัดไป



รูปที่ 2-15 Timed Output Pulse

Timer ของ output ในรูปนี้ นะเป็นสัญญาณแบบ Periodic จนกระทั่ง เส้นทางของอินเตอร์รัพเสร็จ ในขณะนั้นคือ 1 ช่วงเวลาการได้ใช้รูปแบบของ synchronous pulse เป็น Output แบบพอร์ทขนาน

Interrupt latency อาศัยลำดับความสำคัญของ อินเตอร์รัพและจำนวนเวลา ซึ่งเส้นทางอื่นๆของอินเตอร์รัพซึ่งมีค่าเหมือนกัน หรือ ลำดับก่อนหลัง(Priority) สูงกว่าจะคอยขัดขวางการอินเตอร์รัพ ชุดคำสั่งแรกของการอินเตอร์รัพของเส้นทาง จะเริ่มการทำงานได้ภายใน 30สัญญาณนาฬิกาของการเรียกของอินเตอร์รัพหรับอินเตอร์รัพต่อไป ที่มีค่าความสำคัญสูงที่สุด สรุปว่าจะใช้19 สัญญาณนาฬิกา สำหรับชุดคำสั่งที่ยาวที่สุด เมื่อการประมวลผลเสร็จ และอีก 10สัญญาณนาฬิกาสำหรับการอินเตอร์รัพต่อไป กับการประมวลผล Pushing รีจิสเตอร์ต้องการ 10-12 นาฬิกาต่อ 16 บิต รีจิสเตอร์ การPOP รีจิสเตอร์ต้องการ 7-9 สัญญาณนาฬิกา การคืนค่าจากอินเตอร์รัพต้องการ 7 สัญญาณนาฬิกา ถ้ารีจิสเตอร์ทั้งสามสามารถเก็บข้อมูลและคำสั่ง20คำสั่งนี้ มีค่าประมาณ 5 นาฬิกาสำหรับการทำงาน การเข้าถึง อินเตอร์รัพของเส้นทาง จำเป็นต้องใช้ ประมาณ 200 สัญญาณนาฬิกาหรือ 10ไมโครวินาที มีความกว้าง 20MHz สัญญาณนาฬิกา

ความกว้างของPulse modulate output ซึ่งจะมีความกว้างของฟัลส์ที่น้อยที่สุด คือ 10 ไมโครวินาที ถ้ามีการขยายสัญญาณทุกๆ 10ms แล้ว pulse ที่เกิดขึ้นมาใหม่มีความกว้างแตกต่างกัน จำนวนมากสามารถสร้างได้ด้วยอัตราความเร็ว 100 ครั้งต่อวินาที

การติดต่อแบบอะซิงโครนัสแบบ พอร์ทอนุกรม เป็นรับสัญญาณ อะซิงโครนัส บน อินพุตท์แบบอนุกรม อินพุตท์มีค่าในการตรวจสอบมากกว่าค่าส่งข้อมูล

การสร้างฟัลส์ด้วยเวลาที่แน่นอนนั้นจะมีความเกี่ยวข้องอยู่กับ 2 อย่างซึ่งสามารถควบคุมได้ คือช่วงเวลาตั้งแต่ 10ไมโครวินาที ถึง 20ไมโครวินาที

ในการใช้เวลาในการสร้างสัญญาณคาบถูกสร้างขึ้นมาทุกๆ 10ไมโครวินาที ถ้า timer B ใช้ในการควบคุมเอาพุตท์รีจิสเตอร์จะมีค่าประมาณ 100ครั้ง ซึ่งดีกว่าที่มีในปัจจุบัน เป็นเพราะ time B มี match รีจิสเตอร์ ซึ่งสามารถเขียนโปรแกรมในการสร้างฟัลส์ ที่เวลาที่เราต้องการ The match รีจิสเตอร์ มี 2 cascades รีจิสเอตร์ The match รีจิสเตอร์ และ match รีจิสเตอร์ ตัวถัดไป The match รีจิสเตอร์ถูกโหลดพร้อมกับข้อมูลที่อยู่ในรีจิสเอตร์ที่ถัดไป ในขณะที่มีการสร้างรีจิสเอตร์ Timer B สามารถถูกสัญญาณนาฬิกา ได้โดย Cycle/2 สามารถหารด้วยตัวเลยที่อยู่ระหว่าง 1-256 Timer B มีความเร็วในการนับ 10 MHz ด้วยความเร็ว 20MHz ของสัญญาณนาฬิกา เหตุการณ์ที่เกิดขึ้นควรมีค่าประมาณ 10 ns Timer B และ match register มีขนาด 10 bit

ในการใช้ Timer B เอาพุตท์สามารถถูกกำหนดตำแหน่งมีค่าตรงกับ clk/2 Timer B ทำได้ ดังนั้นจึงถูกใช้ในการจับเวลาซึ่ง เหตุการณ์ที่เกิดขึ้น ข้างนอกสามารถย้ายไป จุดเชื่อมต่อโดยแถวอินเตอร์รัพภายนอก แถวอินเตอร์รัพสามารถจัดการสำหรับ อินเตอร์รัพเพื่อที่จะเอาค่า ขอบขาขึ้น, ขอบขาลง หรือทั้งสองอย่าง เลือกเอาในเวลาที่เกิดขอบขึ้น อินเตอร์รัพ routine สามารถอ่านค่าได้ โดย timer B counter ในเวลาการทำงานของอินเตอร์รัพ routine ขึ้นอยู่กับตอนที่ Timer ถูกอ่าน สามารถถูกลบออกจากค่าเวลา ถ้าไม่มี interrupt ตัวอื่นๆ ที่เหมือนกัน หรือมีค่าลำดับก่อนหลังที่สูงกว่าแล้ว ไม่แน่ว่าตำแหน่งของขอบขาลงจนใกล้เวลาของอินเตอร์รัพ latency หรือเกี่ยวกับ เวลาของการทำงานของคำสั่งที่ยาวที่สุด โดยมีค่าประมาณ 10 สัญญาณนาฬิกาหรือ 0.5 ไมโครวินาที สำหรับ 20MHz สัญญาณของสัญญาณนาฬิกา มาตรฐานของความกว้างของฟัลส์สำหรับความยาว

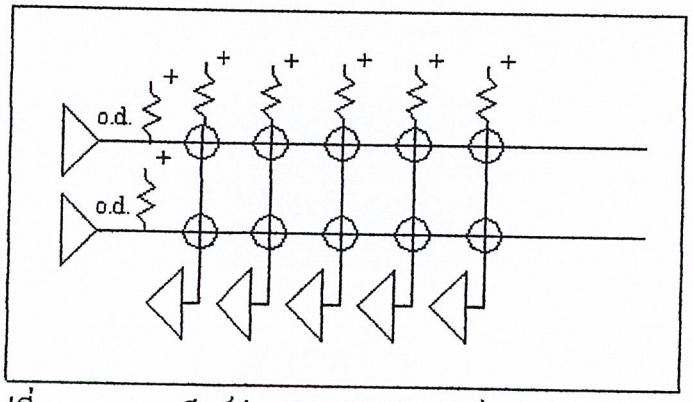
ของฟัลส์ประมาณ 1ไมโครวินาที ถ้าหลายฟัลส์มีความต้องการมาตรฐานในการทำงานแล้ว ความแม่นยำจะลดลงแต่ การลดลงนี้ จะมีค่าน้อยที่สุด โดยความระมัดระวังของการโปรแกรม

2.1.4.2 Pulse Width Modulation to Reduce Reply Power

โดยปกติ relays ต้องการกระแสให้อยู่ในสภาวะปิดนั้นน้อยกว่าค่าที่ต้องการเพื่อที่จะปิดมัน เช่น ถ้ามีdriver ได้ถูกswitch ที่75% ของ Duty cycle ซึ่งใช้ความกว้างของ pulse modulation หลังจากช่วงเวลาแรก ในขณะที่ relay armature ได้ถูกหยิบขึ้นมา โดยการถือค่ากระแสประมาณ 75% ของ duty-cycle และกำลังในการใช้ประมาณ 56%ของส่วนใหญ่

2.1.4.3 Open-Drain Output Used for Key Scan

The parallel Port D output ถูกโปรแกรมเฉพาะ เพื่อที่จะเปิดออก เป็นสิ่งที่น่าสนใจสำหรับการตรวจสอล switch matrix แถวนอนนั้นทำงานอยู่ในแถวข้างล่าง แล้วคอลัมภ์ตรวจสอบสำหรับสายอินพุทต์ ข้างล่าง เป็นการทวนสำหรับในแต่ละแถว ซึ่งเป็นข้อได้เปรียบของการใช้เปิดทางออกของเอาพุตท์ ถ้ามี 2 อย่างบน คอลัมภ์เดียวกัน ซึ่งอยู่ข้างล่าง ไม่มีการแย่งระหว่างระหว่างไดร์ว สายสูง และสายอื่นๆที่เป็นสายต่ำ



รูปที่ 2-16 การเคลียร์ช่องสำหรับเอาพุตท์สำหรับ Key Scan

2.1.4.4 Cold Boot

ส่วนใหญ่ microprocessor เริ่มการทำงานมีการกำหนดตำแหน่ง บ่อยครั้ง ตำแหน่งเท่ากับ0 หลังจากการ reset หรืออยู่ในสภาวะเปิด Rabbit มี mode การทำงานอยู่ 2 โหมด ขา (SMODE0,SMODE1) ในสถานะของโลจิกของทั้ง2 ขา เป็นตัวตัดสินในใจการเริ่มการทำงาน หลังจากรีเซ็ท ถ้าทั้ง2 ขาต่อลงกราว์แล้ว สถานะของ Rabbit จะเริ่มการทำงานที่ตำแหน่ง 0 ในการรี

เซ็ท ตำแหน่ง0 มีความหมายถึงเป็นจุดเริ่มต้นของ หน่วยความจำที่มีการติดต่อกับส่วนควบคุม หน่วยความจำ /CS0 และ /OE0 แต่อย่างไรก็ตาม ก็มีวิธีการเริ่มต้น มีอยู่3วิธีที่เป็นไปได้ ซึ่งเป็นทางเลือกจากทั้งหมดในการยอมรับ การเคลื่อนที่บนพอร์ทสื่อสาร ซึ่งใช้ในการเก็บโปรแกรมที่จำเป็นต่อการบูท ในแรมซึ่งสามารถใช้ในการสนับสนุนขบวนการ บูทครั้งที่2เช่น การดาว์โหลดโปรแกรมบนพอร์ทติดต่อสื่อสารเดียวกัน

ช่องการติดต่อสื่อสารทั้ง 3 ช่องอาจจะใช้สำหรับ bootstrap, พอร์ทอนุกรม A ในโหมดอะซิงโครนัสที่ 2400 bps พอร์มอนุกรม Aบน โหมดซิงโครนัส ร่วมกับสัญญาณนาฬิกาภายนอก หรือพอร์ทลูก

วิธีการcold-boot ยอมรับจาก กลุ่ม3ไบต์ กำหนดค่าแอดเดรส และ ดาต้าไบต์ แต่ละTripet นี้เป็นสาเหตุ ที่ทำให้เกิดการเขียนข้อมูลบนหน่วยความจำ หรือ ที่ว่างภายใน I/O ค่าของบิตที่สูงของตำแหน่งได้ถูกตั้งค่าไว้สำหรับ พื้นที่เฉพาะเจาะจงของที่ว่างของ I/O และด้วยเหตุนี้การเขียนจึงมีขอบเขตไว้สำหรับ 32 kแรกของว่างซึ่ง cold boot เป็นตัวตัดสินใจในการเก็บข้อมูลสำหรับตำแหน่งข้อมูลในที่ว่าง I/O ซึ่งเป็นสาเหตุในการทำงานต้องเริ่มการทำงานที่ address0 ตั้งแต่ชิพหน่วยความจำสามารถประมาณเอาไว้เข้าถึง address0 โดยการเก็บข้อมูลใน I/O space แรมสามารถทำได้ชั่วคราวเมื่อถูกแมพจาก 0 จนถึงไม่ยอมรับค่าอีก เพื่อที่จะทำงานร่วมกับการเขียนโปรโตคอล ของหน่วยความจำแบบแฟลชซึ่งมีความซับซ้อนมาก ซึ่งโดยปกติจะไม่มีหน่วยความจำเก็บไว้ใน address 0

ลักษณะพิเศษของความสามารถของ cool-boot

-หน่วยความจำแบบแฟลชสามารถเป็นตัวเชื่อมระหว่าง CPU และ โปรแกรม เส้นทางของพอร์ทอนุกรม หรือ บูทโปรแกรมพอร์ทขนานก่อนการเชื่อมต่อ

-ในการแก้ไขโปรแกรมของหน่วยความจำแบบแฟลช เราสามารถทำได้ในตอนทำงานอยู่ ในขณะที่อยู่ในระหว่างการพัฒนาซอฟแวร์ ขณะนั้น เราสามารถพัฒนาได้โดยการรีโหลด จากซอฟแวร์ regardless ของสถานะนั้น อยู่ในขณะที่กำลังทำงานในกระบวรการเป็นมาตรฐานทั่วไปของ program cable สำหรับ Dynamic C ที่เราสามารถใช้ในการพัฒนารูปแบบเพื่อที่จะรีเซ็ทและ cold-boot เป้าหมาย

-ถ้าrabbit ได้ถูกใช้เป็นตัวประมวลผลตัวลูก ตัวประมวลผลตัวแม่สามารถทำ cold boot บนพอร์ทลูก หมายความว่า ตัวลูกสามารถทำงานแทนได้โดยไม่มีการเปลี่ยนแปลงของหน่วยความจำต้องการแรมเท่านั้น

2.1.4.5 The Slave Port

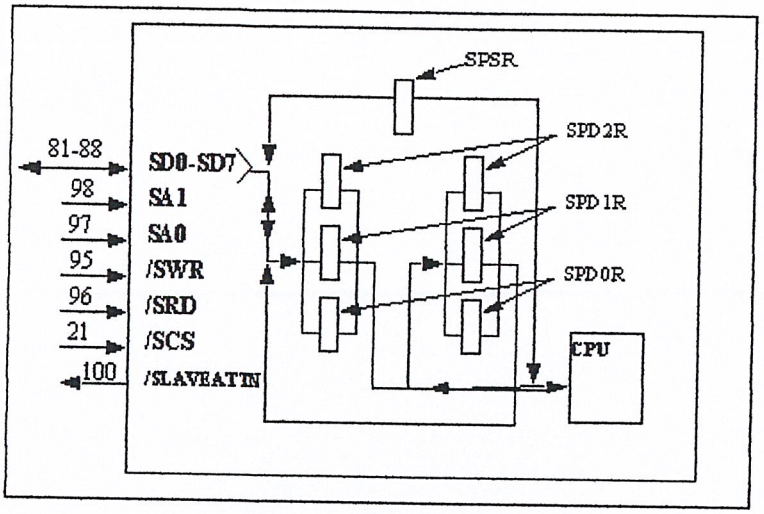


Figure 2-17 Slave port

The slave processor, พอร์ทลูกมีการเชื่อมต่อกับ ดาต้าบัสของไมโครโปรเซสเซอร์ตัวแม่ การติดต่อสื่อสารระหว่างตัวแม่ กับ ตัวลูก สามารถอาศัยรีจิสเตอร์3 ตัวบนอุปกรณ์บน rabbit ในแต่ละเส้นทางของการติดต่อสื่อสาร สำหรับผลลัพท์ทั้งหมด 6 ดาต้ารีจิสเตอร์ดังภาพรวมพอร์ทลูกที่ซึ่งถูกอ่านโดยslave หรือ master โดย2เส้นทางของ slave address ได้ถูกใช้โดย master เป็นคนเลือกรีจิสเตอร์ ให้อ่านหรือเขียน รีจิสเตอร์นำเอาข้อมูลจาก master ไปยัง slave The register นั้น ทำงานสวนกลับกันทางเดิม ปรากฏว่ามี read-write register ทั้ง 3 ตัวอยู่ภายใต้การควบคุม ดังนั้น master สามารถจัดการเขียนส่งไปยัง status register ซึ่งถูกเอาไปใช้เช่นเดียวกับ อุปกรณ์ สัญญาณต่างๆ และเป็นไปไม่ได้ที่สามารถเขียนส่งไปยัง status register โดยที่รีจิสเตอร์ทั้ง3ตัวนี้ สามารถเขียนเช่นเดียวกับอ่าน register ส่งไปยังslave rabbit master เป็นตัวจัดการความสามารถของ strobe จนถึงอ่าน three read data register และ status register ซึ่งรีจิสเตอร์เป็นรีจิสเตอร์ที่เขียนส่งไปยัง Rabbit

รีจิสเตอร์ตัวแรก หรือ รีจิสเตอร์ทั้ง 3 คู่มีความพิเศษในการเขียน สามารถอินเตอร์รัพโปรเซสเซอร์ตัวอื่นๆได้ในจุดเชื่อมต่อของการติดต่อสื่อสารของ master-slave สายเอาท์พุต ที่มา

จากตัวลูกจะรักษาสถานะไว้ ในขณะที่ตัวลูกเขียนส่งๆไปยัง รีจิสเตอร์ลูก 0 เส้นทางนี้อาจจะถูกอินเตอร์รัพโดย ตัวแม่ได้ ระบบวงจรภายในซึ่งอยู่ในตัวลูกสามารถถูกกำหนดค่า อินเตอร์รัพของตัวลูกในขณะที่ ตัวแม่ เขียนส่งไปยังรีจิสเตอร์ลูกที่ 0

รีจิสเตอร์แสดงสถานะซึ่งมีค่าในทั้ง 2 ฝั่ง ได้ถูกเก็บไว้ในรีจิสเตอร์ทุกๆตัว และมีการรายงาน ถ้าความสามารถของอินเตอร์รัพ ได้ถูกร้องขอจาก ข้างใดข้างหนึ่ง ซึ่งรีจิสเตอร์แสดงสถานะเก็บค่า track ว่า "full-empty" เป็นสถานะของแต่ละรีจิสเตอร์ รีจิสเตอร์ลูกพิจารณาในขณะที่ข้างหนึ่งของการเชื่อมต่อ มันกลายสถานะว่าง ถ้าฝั่งอื่นอ่านมัน เส้นทางนี้สามารถที่ทดสอบได้ ถ้าเส้นทางอื่นมีการปรับเปลี่ยนรีจิสเตอร์ หรือฝั่งอื่นมีพื้นที่ในการเก็บข้อมูลลงรีจิสเอตร์

การเชื่อมการติดต่อของ The master –slave สามารถทำให้เป็นความจริง "Set and forget" อาศัยข้อตกลงที่ใช้ในการติดต่อสื่อสาร โดยแต่ละฝั่งสามารถทำให้เกิดการสั่งการ หรือการร้องขอโดยการเก็บข้อมูลในรีจิสเตอร์ และแล้วไปเกี่ยวกับการทำงาน ในขณะที่ทางด้านอื่นสามารถดูแลเกี่ยวกับการร้องขอ ตามตาราง ส่วนด้านอื่นสามารถเตือนโดยอินเตอร์รัพสามารถเก็บไว้ได้ ในขณะที่ข้อมูลถูกเก็บไว้ใน รีจิสเตอร์ 0

รีจิสเตอร์ทั้งสามตัว ถูกตรวจสอบโดยแต่ละด้าน สำหรับแต่ละเส้นทางการติดต่อรีจิสเตอร์ตัวแรก, รีจิสเตอร์ 0, มีฟังก์ชั่นพิเศษเพราะว่า อินเตอร์รัพสามารถถูกสร้างขึ้นโดยการเขียนของรีจิสเตอร์โดยมีสาเหตุมาจาก อินเตอร์รัพย้ายไปฝั่งอื่น ถ้าอินเตอร์รัพสามารถทำได้ มีโปรโตคอลชนิดหนึ่งที่สามารถเก็บข้อมูลใน รีจิสเตอร์ 1,2 และขั้นตอนสุดท้ายของการเก็บ จะเก็บไว้ที่รีจิสเตอร์ 0 แล้ว 24 บิตของข้อมูลมีค่าส่งไปยังเส้นทางของอินเตอร์รัพบนฝั่งตรงข้ามการการติดต่อ

ข้อมูลที่มีขนาดใหญ่จะถูกส่งข้ามการติดต่อนั้นต้องอาศัยอินเตอร์รัพ สำหรับการส่งทีละไบต์ เหมือนกับส่งไปยังพอร์ทอนุกรม หรือ UART ในกรณีการส่งแบบ Full-Duplex สามารถส่งไปต่างๆเหมือนกับ สามารถทำอะไรก็ได้ด้วย UART overhead สำหรับการส่งของอินเตอร์รัพอาจจะมีการส่งประมาณ 100 สัญญาณนาฬิกาต่อ ไบต์ สมมุติคำสั่ง 20 คำสั่งบนเส้นทางของอินเตอร์รัพหลายวิธีที่สร้างความพอใจโดยส่งข้อมูล 2 ไบต์ ในแต่ละอินเตอร์รัพ ซึ่งจะมีค่าใกล้เคียงกับครึ่งหนึ่งของ overhead ถ้าพบอยู่ระหว่างกระบวนการ, ข้อมูลสามารถถูกส่งด้วยความเร็วประมาณ 25 สัญญาณนาฬิกาต่อ ไบต์ โดยแต่จุด รีจิสเตอร์แสดงสถานะกำลังรอสำหรับฝั่งอื่นๆที่อ่าน/เขียน ข้อมูลรีจิสเตอร์แล้วเขียน/อ่าน อีกรอบโดยฝั่งอื่น

2.1.4.5 Slave Rabbit As a Protocol UART

Application ที่ดี สำหรับ Rabbit ใช้ ตัวลูกเพื่อสร้าง 4 พอร์ท UART ซึ่งเราสามารถใช้ตาม รายละเอียดของมาตรฐานของการติดต่อตัวแม่ส่งและรับข้อความก่อน พอร์ทลูก ความผิดพลาด, ส่ง ใหม่อีกรอบอื่นๆสามารถจัดการได้ด้วยตัวลูก

2.1.5. Pin Assignments and Functions

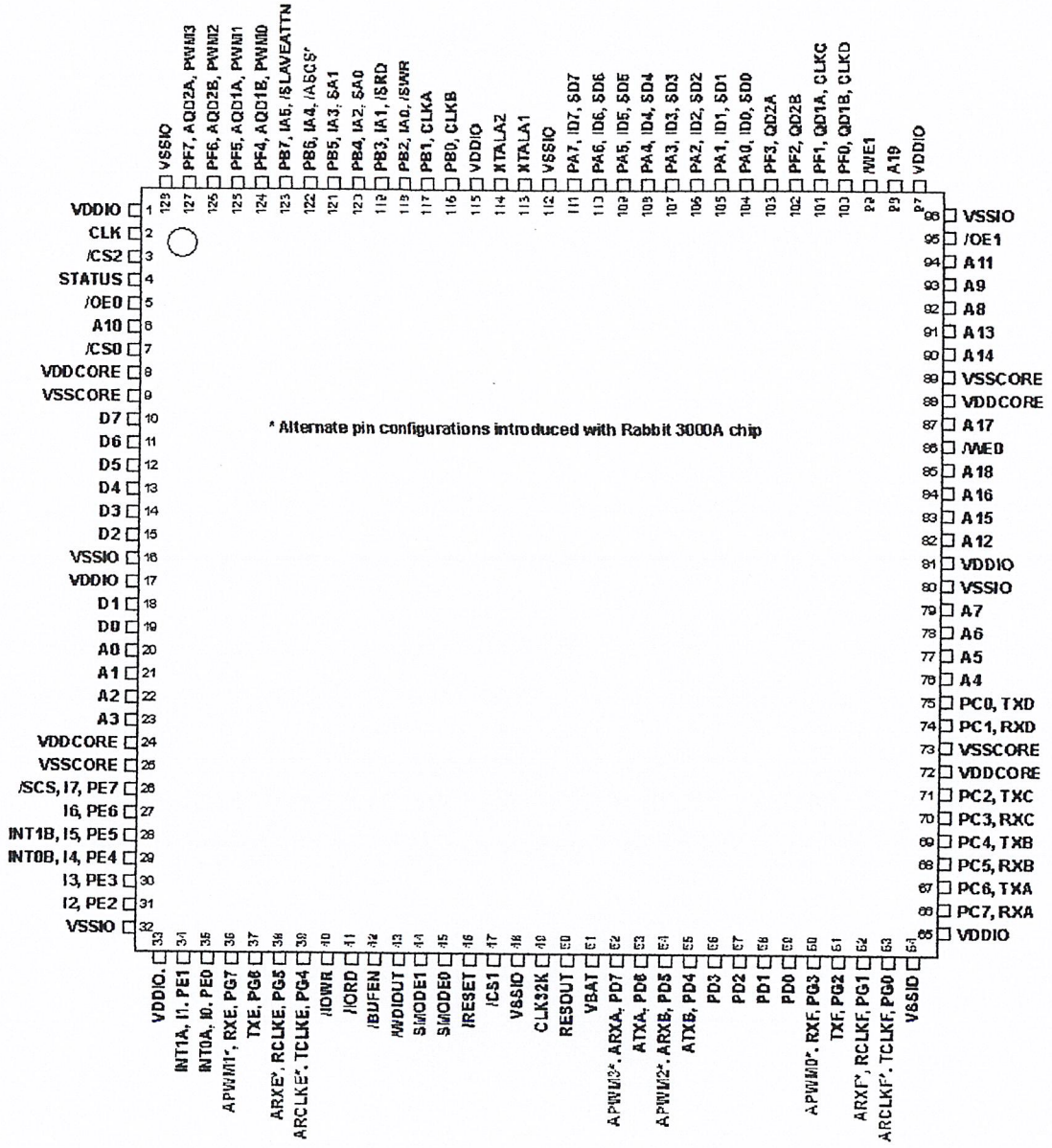
2.1.5.1 LQFP Package

2.1.5.1.1 Pin out

Rabbit 3000 (AT56C55-IL1T, IL2T)

128-pin Low-Profile Quad Flat Pack (LQFP)

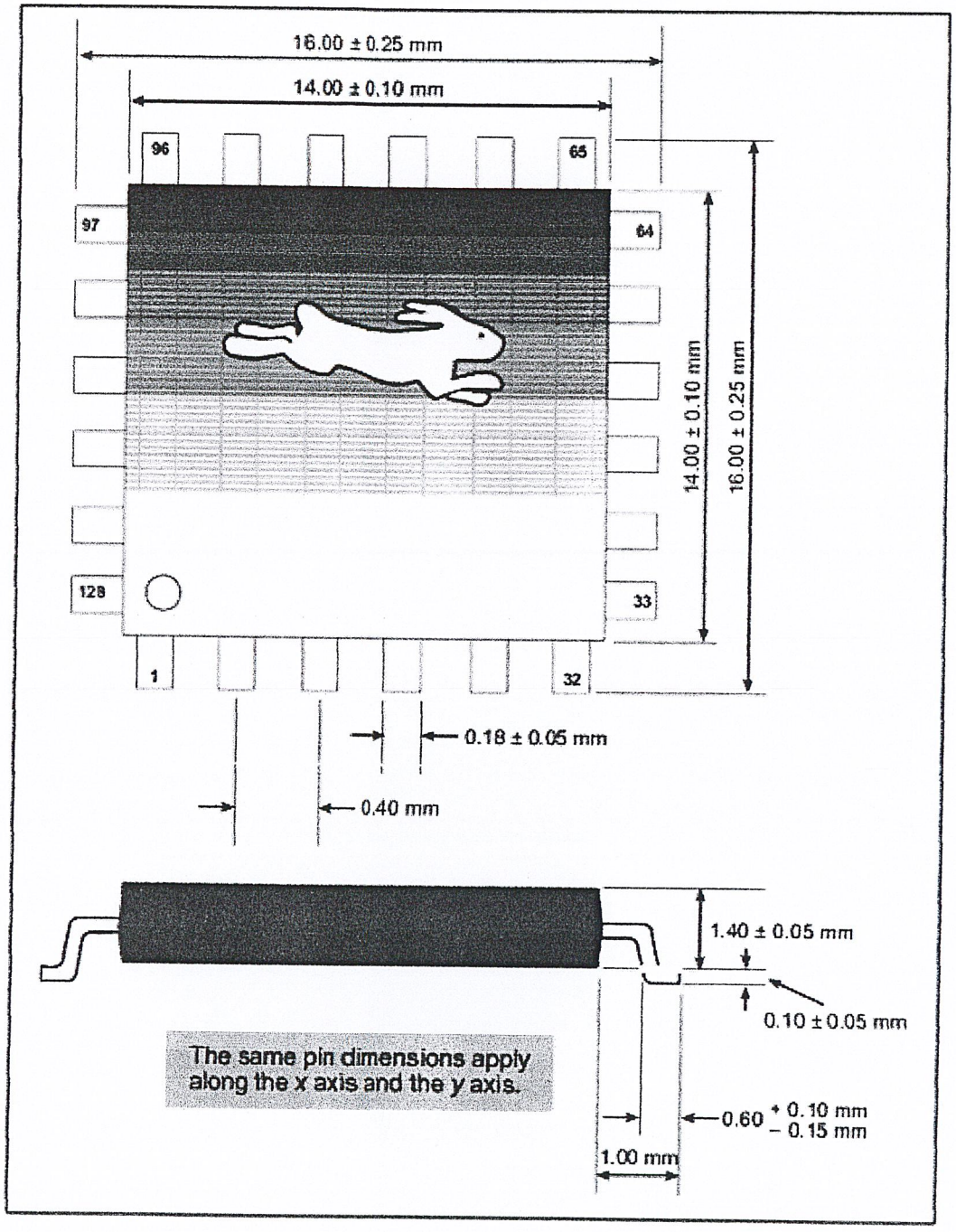
14 × 14 Body, 0.4 mm pitch



รูปที่ 2.18 Package Outline and Pin Assignments

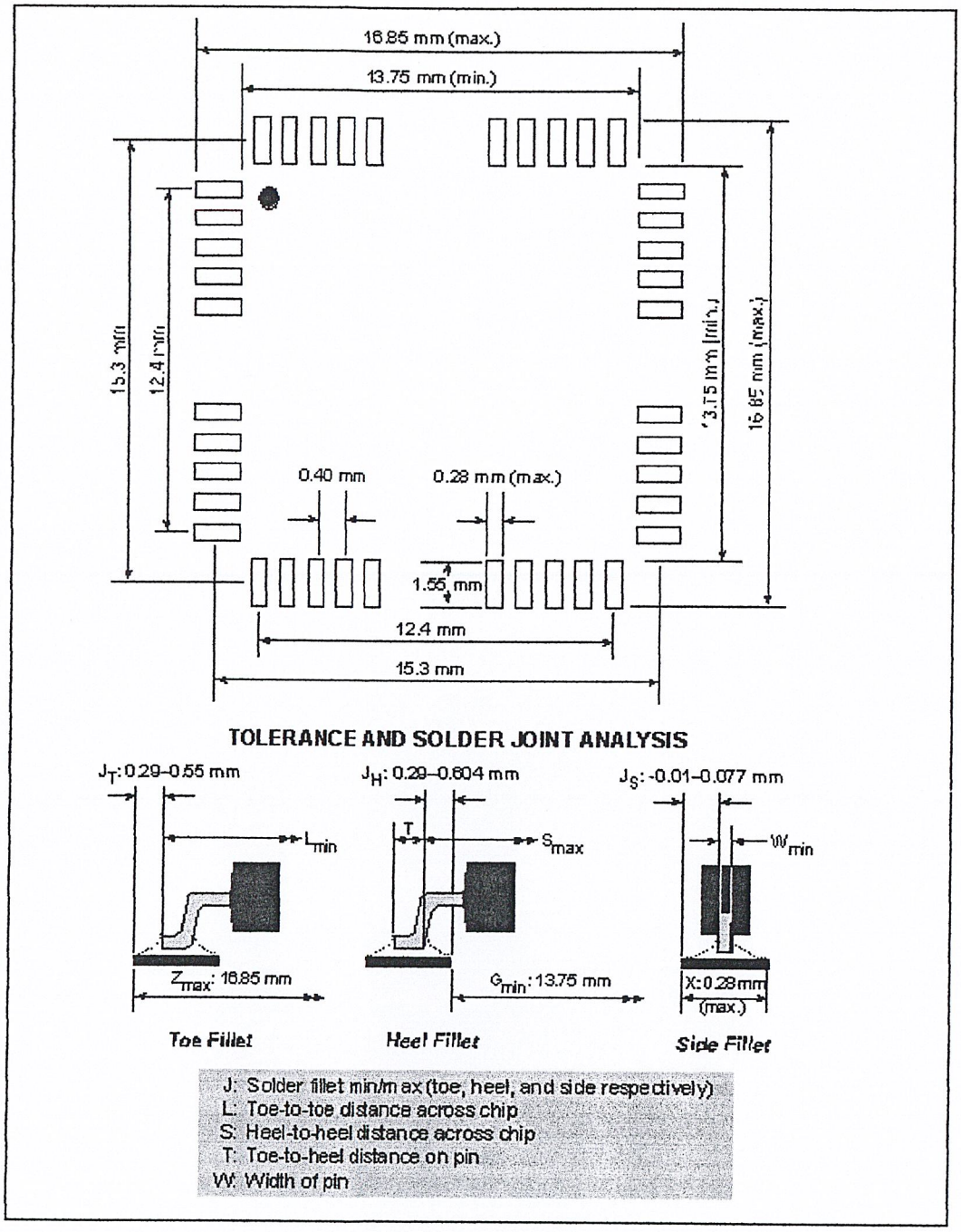
2.1.5.1.2 Mechanical Dimensions and Land Pattern

รูปที่ 2-19 shows แสดงถึงทางกายภาพ ดูจากมิติของ of the Rabbit 3000 LQFP package.



รูปที่ 2-19 Mechanical Dimensions Rabbit LQFP Package

รูปที่ 2-20 แสดง บอร์ด PC สำหรับกระต่าย 3000 LQFP 128 ขา โดยมีพื้นฐาน จากIPC-SM-782 มาตรฐานพัฒนาโดย Surface Mount Land Pattern และSurface Mount Dsign and Land Pattern, IPC, Northbrook, IL, 1999.



รูปที่ 2-20 PC Board Land Pattern for Rabbit 3000 128-pin LQFP

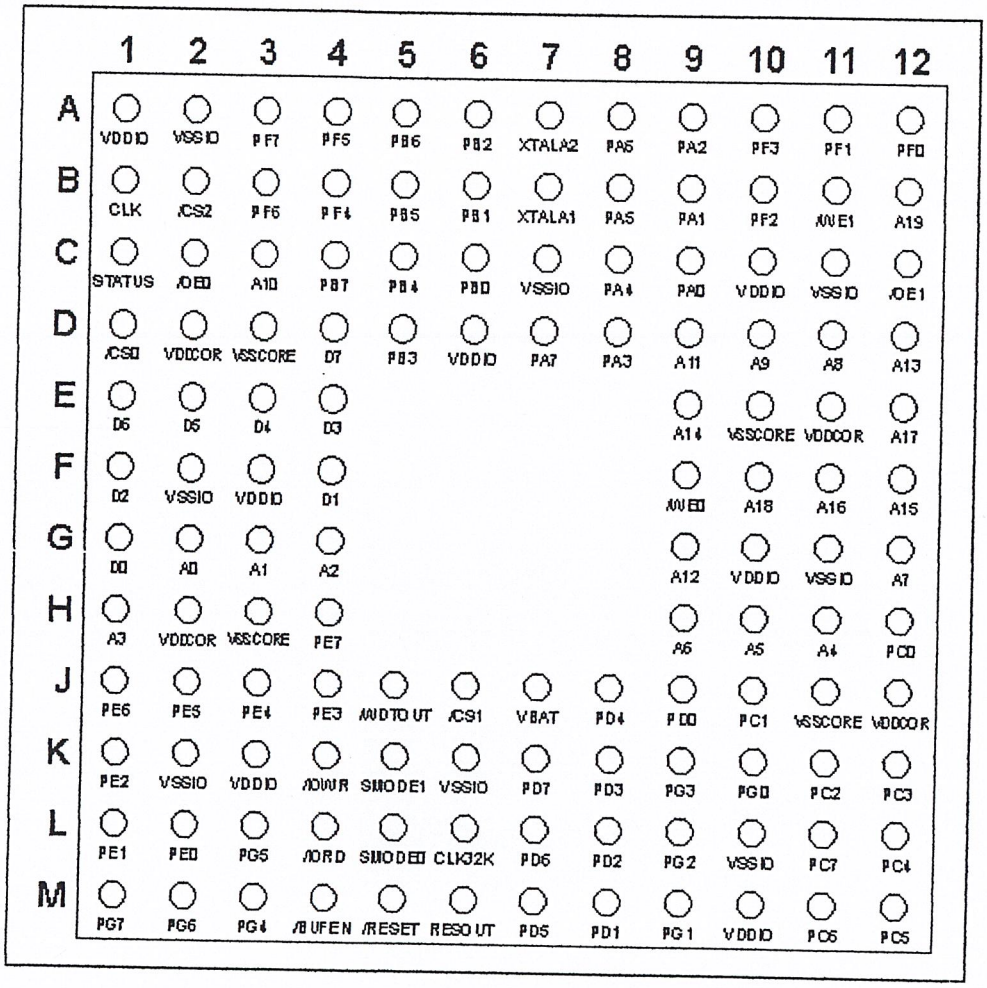
2.1.5.2 Ball Grid Array Package

2.1.5.2.1 Pinout

Rabbit 3000 (AT56C55-IZ1T, IZ2T)

128-pin Thin Map Ball Grid Array (TFBGA)

ขนาด 10 * 10 , 0.8 mm ขา



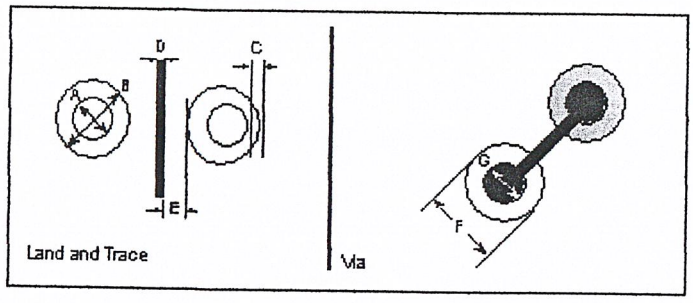
รูปที่ 2-21 Ball Grid Array Pinout Looking Through the Top of Package

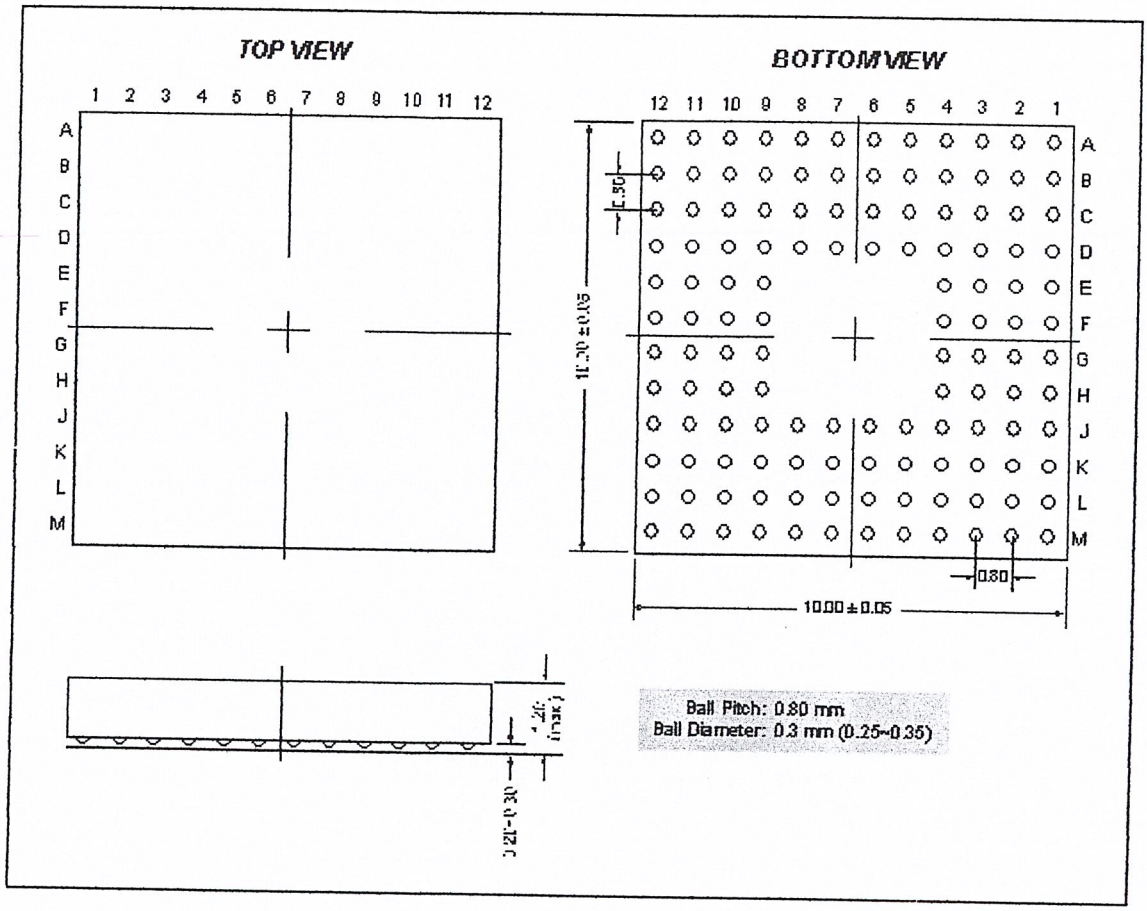
2.1.5.2.2 Mechanical Dimensions and Land Pattern

ตาราง 2-2. Ball and Land Size Dimensions				
Nominal Ball Diameter (mm)	**Tolerance Variation (mm)**	**Ball Pitch (mm)**	**Nominal Land Diameter (mm)**	**Land Variation (mm)**
0.3	0.35-0.25	0.8	0.25	0.25-0.20

พิจารณาจากตารางที่ 2-3 มีพื้นฐานจากการออกแบบขนาด 5 mm และ แสดงระหว่าง single conductor กับ solder lands.

ตาราง 2-3. Design Considerations (all dimensions in mm)		
Key	**Feature**	**Recommendation**
A	Solder Land Diameter	0.254 (0.010)
B	NSMD Defined Land Diameter	0.406 (0.016)
C	Land to Mask Clearance (min.)	0.050 (0.002)
D	Conductor Width (max.)	0.127 (0.005)
E	Conductor Spacing (typ.)	0.127 (0.005)
F	Via Capture Pad (max.)	0.406 (0.016)
G	Via Drill Size (max.)	0.254 (0.010)





รูปที่ 2-22 BGA Package Outline

2.1.5.3 Rabbit Pin Descriptions

ตารางที่ 2-2 แสดงสิ่งทั้งหมดบนอุปกรณ์ , ตามด้วยทิศทาง,ฟังก์ชั่น,และหมายเลขขาที่อยู่บนสิ่งนั้น.

ตารางที่ 2-4 Rabbit Pin Descriptions					
Pin Group	**Pin Name**	**Direction**	**Function**	**Pin Numbers LQFP**	**Pin Numbers TFBGA**
Hardware	CLK	Output	Internal Clock	2	B1
	CLK32K	Input	32 kHz ออสซิเรเตอร์ In	49	L6
	/RESET	Input	Master Reset	46	M5
	RESOUT	Output	Reset Output	50	M6
	XTALA1	Input	Main ออสซิเรเตอร์ In	113	B7
	XTALA2	Output	Main ออสซิเรเตอร์ Out	114	A7
CPU Buses	ADDR[19:0]	Output	Address Bus	various	
	DATA[7:0]	Bidirectional	Data Bus	10-15, 18-19	D4, E1-E4, F1, F4, G0
Status/Control	/WDTOUT	Output	WDT Time-Out	43	J5

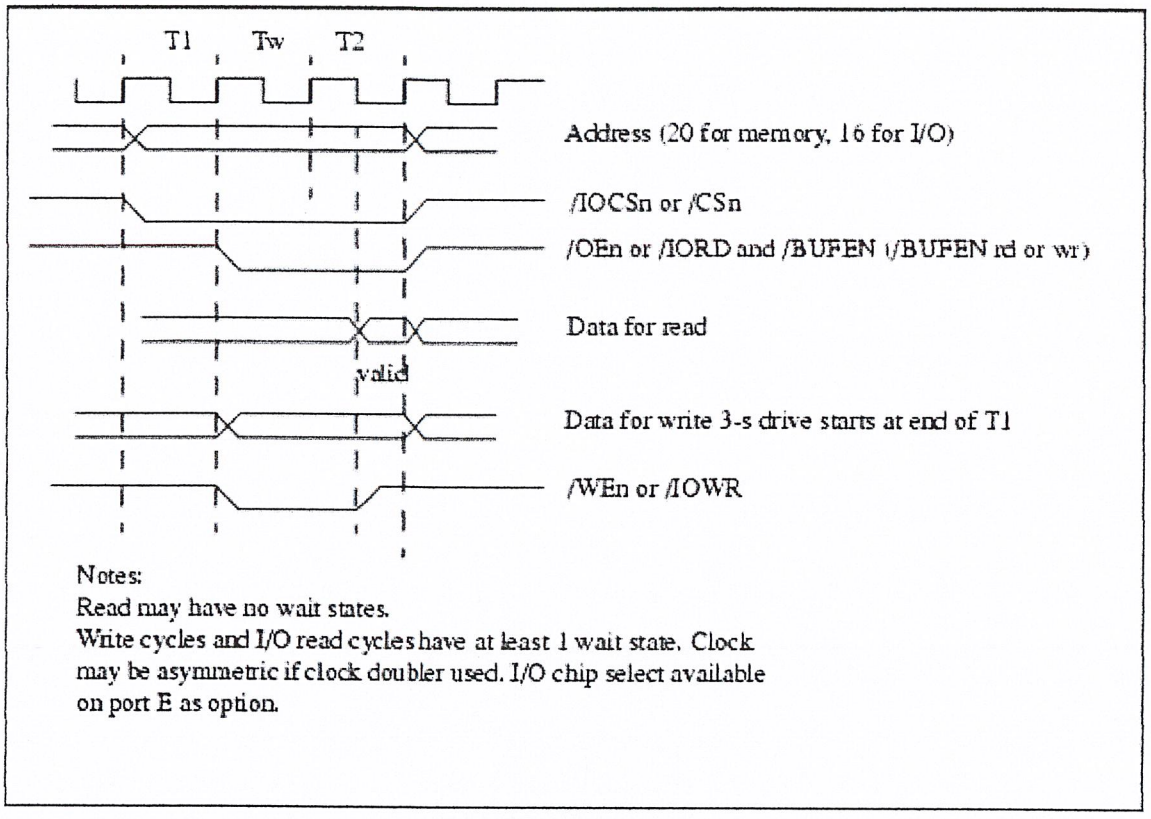
	STATUS	Output	Instruction Fetch First ไบต์	4	C1
	SMODE[1:0]	Input	Bootstrap Mode Select	44, 45	K5, L5
Memory Chip Selects	/CS0	Output	Memory Chip Select 0	7	D1
	/CS1	Output	Memory Chip Select 1	47	J6
	/CS2	Output	Memory Chip Select 2	3	B2
Memory Output Enables	/OE0	Output	Memory Output Enable 0	5	C2
	/OE1	Output	Memory Output Enable 1	95	C12
Memory Write Enables	/WE0	Output	Memory Write Enable 0	86	F9
	/WE1	Output	Memory Write	99	B11

			Enable 1		
I/O Control	/BUFEN	Output	I/O Buffer Enable	42	M4
	/IORD	Output	I/O Read Enable	41	L4
	/IOWR	Output	I/O Write Enable	40	K4
I/O ports	PA[7:0]	Input / Output	I/O Port A	111-104	D7, A8, B8, C8, D8, A9, B9, C9
I/O ports (continued)	PB[7:0]	Input / Output	I/O Port B	123-116	C4, A5, B5, C5, D5, A6, B6, C6
	PC[7:0]	4 In / 4 Out	I/O Port C	66-71, 74, 75	L11, M11, M12, L12, K12, K11, J10, H12
	PD[7:0]	Input / Output	I/O Port D	52-59	K7, L7, M7, J8, K8, L8, M8, J9
	PE[7:0]	Input / Output	I/O Port E	26-31, 34, 35	H4, J1-J4, K1, L1-L2
	PF[7:0]	Input / Output	I/O Port F	127-124, 103-100	A3, B3, A4, B4, A10, B10, A11, A12

	PG[7:0]	Input / Output	I/O Port G	36-38, 60-63	M1, M2, L3, M3, K9, L9, M9, K10
Power, processor core	VDDCORE		+3.3 V	8, 24, 72, 88	D2, E11, H2, J12
Power Processor I/O Ring	VDDIO		+3.3 V	1, 17, 33, 65, 81, 97, 115	A1, C10, D6, F3, G10, K3, M10
Power Battery Backup	VBAT		+3.3 V or battery	51	J7
Ground Processor Core	VSSCORE		Ground	9, 25, 73, 89	D3, E10, H3, J11
Ground Processor I/O Ring	VSSIO		Ground	16, 32, 48, 64, 80, 96, 112, 128	A2, C7, C11, F2, G11, K2, K6, L10

2.1.5.4 Bus Timing

บัสภายนอกมีความจำเป็นในเวลาที่เหมือนกันสำหรับ วงรอบการทำงานของหน่วยความจำ หรือวงรอบการทำงานของ อินพุตท์และเอาต์พุต วงรอบการทำงานของหน่วยความจำเริ่มต้นที่การเลือกชิพ และแถวแอดเดรส สัญญาณนาฬิกาต่อมา เอาพุตท์ ทำงานโดยอ้างเพื่อที่จะอ่าน ดาต้าเอาพุตท์และการเขียน ทำงานเพื่อที่จะสงวนสำหรับการเขียน



รูปที่ 2-23. Bus Timing Read and Write

ในบางกรณี Timing จะแสดงดังรูปที่ 5-6 มีการเพิ่มส่วนหน้าโดยการเข้าถึงแบบหน่วยความจำแบบแฟลช ในขณะที่ สัญญาณครั้งแรก ซึ่งอันดับที่เข้าถึงจะแสดงในรูปที่ 5-6 ในกรณีนี้ แอดเดรสและชิพเลือกเปลี่ยนค่าหลังจาก 1ช่วงสัญญาณและรับค่าสุดท้ายสำหรับหน่วยความจำที่มีการเข้า เอาพุตท์ทำงานและเขียนทำงาน จะถูกดีเลย์โดย 1 ช่วงสัญญาณจากเวลาช่วงสุดท้าย แอดเดรสที่เสถียรและชิพเลือกทำงานอยู่ โดยปกติการที่เข้าถึงหน่วยความจำแบบแฟลชเพื่อเริ่มวนรอบการทำงานของคำสั่งที่เข้ามา โดยยกเลิกหลังจาก 1 ช่วงสัญญาณ ในขณะที่โปรเซสเซอร์เข้าใจการอ่านข้อมูลหรือเขียนวนรอบการทำงานของดาต้าบัส ผู้ใช้ไม่พยายามที่จะออกแบบซึ่งการใช้ชิพเลือกหรือ แอดเดรสหน่วยความจำ สถานะเปลี่ยนส่งสัญญาณโดยปราศจากการใช้สิ่งนี้เข้าไปในการพิจารณา.

2.1.5.5 Description of Pins with Alternate Functions

ตารางที่ 2-5. Pins With Alternate Functions			
Pin Name	**Output Function**	**Input Function**	**Input Capture Option**
PA[7:0]	SLAVE D[7:0], ID[7:0]	SLAVE D[7:0], ID[7:0]	
PB7	SLAVEATTN, IA5		
PB6	IA4	/ASCS[1]	
PB5	IA3	SD1	
PB4	IA2	SD0	
PB3	IA1	/SRD	
PB2	IA0	/SWR	
PB1	CLKA	CLKA	
PB0	CLKB	CLKB	
PC7	n/a	RXA	yes
PC6	TXA	n/a	
PC5	n/a	RXB	yes
PC4	TXB	n/a	
PC3	n/a	RXC	yes
PC2	TXC	n/a	
PC1	n/a	RXD	yes
PC0	TXD	n/a	

PD7	APWM3*	ARXA	yes
PD6	ATXA		
PD5	APWM2*	ARXB	yes
PD4	ATXB		
PD3			yes
PD2			
PD1			yes
PD0			
PE7	I7	/SCS (slave chip select)	
PE6	I6		
PE5	I5	INT1B	
PE4	I4	INT0B	
PE3	I3		
PE2	I2		
PE1	I1	INT1A	
PE0	I0	INT0A	
PF7	PWM3	AQD2A	yes
PF6	PWM2	AQD2B	
PF5	PWM1	AQD1A	yes
PF4	PWM0	AQD1B	
PF3		QD2A	yes

PF2		QD2B	
PF1	CLKC	QD1A, CLKC	yes
PF0	CLKD	QD1B, CLKD	
PG7	APWM1*	RXE	yes
PG6	TXE		
PG5	RCLKE	RCLKE, ARXE*	yes
PG4	TCLKE	TCLKE, ARCLKE*	
PG3	APWM0*	RXF	
PG2	TXF		
PG1	RCLKF	RCLKF, ARXF*	
PG0	TCLKF	TCLKF, ARCLKF*	

[1] Introduced with Rabbit 3000A chip

ตารางที่ 2-6. Parallel Port x Alternate Functions

Parallel Port x Function Register (PCFR) (Address = 0x0055h)
(PDFR) (Address = 0x0065h)
(PEFR) (Address = 0x0075h)
(PFFR) (Address = 0x003Dh)
(PGFR) (Address = 0x004Dh)

Bit(s)	Value	Description
7:0	0	บิตพอร์ทมีผลตอบสนองฟังก์ชั่นปกติ
	1	บิต พอร์ท corresponding พาสัญญาณต่างๆ ไปยังเอาท์พุต บิตเท่านั้นที่มี่การเลือกฟังก์ชั่น มีบิตควบคุมในรีจิสเตอร์ มี 4 พอร์ท พอร์ท C, 4 พอร์ทในพอร์ท D, 8 พอร์ทในพอร์ท E, 4 พอร์มในพอร์ท F, และ

		8 พอร์ทในพอร์ท G.

ตารางที่ 2-7. Parallel Port x Alternate Functions Control Bits

	Alternate Output Function					
Bit	**Port B**	**Port C**	**Port D**	**Port E**	**Port F**	**Port G**
7	/SLAVEATTN, IA5		APWM3	I7	PWM3	APWM1
6	IA4	TXA	ATXA	I6	PWM2	TXE
5	IA3		APWM2	I5	PWM1	RCLKE
4	IA2	TXB	ATXB	I4	PWM0	TCLKE
3	IA1			I3		APWM0
2	IA0	TXC		I2		TXF
1	CLKA			I1	CLKC	RCLKF
0	CLKB	TXD		I0	CLKD	TCLKF

2.1.5.6 DC Characteristics

ตารางที่ 2-8. Rabbit 3000 Absolute Maximum Ratings

Symbol	Parameter	Maximum Rating
TA	อุณหภูมิในการทำงาน	-55° to +85°C
TS	อุณหภูมิในเก็บ	-65° to +150°C

	ค่าโวลล์ที่มากที่สุด ออสซิเรเตอร์ บัฟเฟอร์ของอินพุตท์ 5-V-tolerant I/O	 VDD + 0.5 V 5.5 V
VDD	ค่าโวลล์มากที่สุดในการทำงาน	3.6 V

การกระตุ้นดังรูปที่ 2-7 เป็นสาเหตุหนึ่งที่ทำให้เครื่องชำรุด มีอัตรานี้เป็นการกระตุ้นเท่านั้น และฟังก์ชั่นการจัดการของ Rabbit 3000 ใช้ได้ในสถานการณ์นี้หรือสถานการณ์อื่นใดเลยเหล่านั้นแสดงในส่วนนี้ไม่ถูกส่อให้เห็น. การนำออกแสดงเพื่อเงื่อนไขอัตราสูงที่สุดสมบูรณ์สำหรับระยะเวลาที่ขยายอาจจะมีผลต่อความเชื่อถือได้ของ Rabbit 3000

ตารางที่ 2-8 โครงสร้าง DC ลักษณะของ Rabbit 3000 ที่ 3.3 V ข้ามระยะอุณหภูมิที่กระทำที่แนะนำจาก TA = -55°C to +85°C, VDD = 3.0 V to 3.6 V.

ตารางที่ 2-9 3.3 Volt DC Characteristics

Symbol	Parameter	Test Conditions	Min	Typ	Max	Units
VDD	โวลล์ในการส่ง		3.0	3.3	3.6	V
VIH	โวลล์ที่เข้ามาในระดับสูง		2.0			V
VIL	โวลล์ที่เข้ามาในระดับต่ำ				0.8	V
VOH	โวลล์ที่ออกมาในระดับสูง	IOH = 6.8 mA, VDD = VDD (min)	0.7 × VDD			V
VOL	โวลล์ที่ออกมาในระดับต่ำ	IOL = 6.8 mA, VDD = VDD (min)			0.4	V
IIH	กระแสที่รับเข้ามาในระดับสูง(กรณีเลวร้าย	VIN = VDD, VDD = VDD (max)			10	µA

	ที่สุด,ในทุกบัฟเฟอร์)					
IIL	กระแสที่รับเข้ามาในระดับต่ำ(กรณีเลวร้ายที่สุด,ในทุกบัฟเฟอร์)	VIN = VSS, VDD = VDD (max)	-10			µA
IOZ	กระแสที่ออกมาในขณะที่มีค่าอิมพลีแดนท์สูง	VIN = VDD or VSS,VDD = VDD (max), no pull-up	-10		10	µA

2.1.5.7 I/O Buffer Sourcing and Sinking Limit

ถ้ามิใช่มิฉะนั้นเจาะจง, บริเวณบัฟเฟอร์ I/O ของ Rabbit มีความสามารถจัดการของข้อมูลและกินไฟขนาด 6.8 ma ต่อขา ที่ความเร็วที่เปลี่ยน AC จำกัดเกี่ยวกับค่าที่สูงที่สุดทนรับกระแสอนุญาติ

6. Rabbit Internal I/O Registers

ตารางที่ 2-10 Rabbit 3000 Peripherals and Interrupt Service Vectors

On-Chip Peripheral	ISR Starting Address
System Management	{IIR[7:1], 0, 0x00}
Memory Management	No interrupts
Slave Port	{IIR[7:1], 0, 0x80}
Parallel Port A	No interrupts
Parallel Port F	No interrupts
Parallel Port B	No interrupts
Parallel Port G	No interrupts
Parallel Port C	No interrupts

Input Capture	{IIR[7:1], 1, 0xA0}
Parallel Port D	No interrupts
Parallel Port E	No interrupts
External I/O Control	No interrupts
Pulse Width Modulator	No interrupts
Quadrature Decoder	{IIR[7:1], 1, 0x90}
External Interrupts	INT0 {EIR, 0x00} INT1 {EIR, 0x10}
Timer A	{IIR[7:1], 0, 0xA0}
Timer B	{IIR[7:1], 0, 0xB0}
Serial Port A (async/cks)	{IIR[7:1], 0, 0xC0}
Serial Port E (async/hdlc)	{IIR[7:1], 1, 0xC0}
Serial Port B (async/cks)	{IIR[7:1], 0, 0xD0}
Serial Port F (async/hdlc)	{IIR[7:1], 1, 0xD0}
Serial Port C (async/cks)	{IIR[7:1], 0, 0xE0}
Serial Port D (async/cks)	{IIR[7:1], 0, 0xF0}
RST 10 instruction	{IIR[7:1], 0, 0x20}
RST 18 instruction	{IIR[7:1], 0, 0x30}
RST 20 instruction	{IIR[7:1], 0, 0x40}
RST 28 instruction	{IIR[7:1], 0, 0x50}
RST 38 instruction	{IIR[7:1], 0, 0x70}

2.1.6.1 Default Values for all the Peripheral Control Registers

ค่าพื้นฐานสำหรับรีจิสเตอร์คอนโทรลอุปกรณ์เสริมทั้งหมดถูกแสดงในตาราง 2-11 รีจิสเตอร์ภายใน CPU มีผลกระทบ โดยการรีเซ็ทของ Stack Pointer (SP), Program Counter (PC), the IIR รีจิสเตอร์, the EIR รีจิสเตอร์ และ IP รีจิสเตอร์ IP รีจิสเตอร์ ถูกตั้งเหมือนกัน (หยุดการทำงานของอินเตอร์รัพ) ขณะที่รีจิสเตอร์ CPU ที่แสดงรายการอื่นๆทั้งหมดถูกตั้งคืนค่าเป็น ศูนย์ทั้งหมด.

ตารางที่ 2-11. Rabbit Internal I/O Registers

Register Name	Mnemonic	I/O Address	R/W	Reset
Global Control/Status Register	GCSR	0x00	R/W	11000000
Global Clock Modulator 0 Register	GCM0R	0x0A	W	00000000
Global Clock Modulator 1 Register	GCM1R	0x0B	W	00000000
Breakpoint/Debug Control Register	BDCR	0x0C	W	0xxxxxxx
Global Power Save Control Register	GPSCR	0x0D	W	0000x000
Global Output Control Register	GOCR	0x0E	W	00000000
Global Clock Double Register	GCDR	0x0F	W	00000000
MMU Instruction/Data Register	MMIDR	0x10	R/W	00000000
MMU Common Base Register	STACKSEG	0x11	R/W	00000000
MMU Bank Base Register	DATASEG	0x12	R/W	00000000
MMU Common Bank Area Register	SEGSIZE	0x13	R/W	11111111
Memory Bank 0 Control Register	MB0CR	0x14	W	00001000
Memory Bank 1 Control Register	MB1CR	0x15	W	xxxxxxxx
Memory Bank 2 Control Register	MB2CR	0x16	W	xxxxxxxx

Memory Bank 3 Control Register	MB3CR	0x17	W	xxxxxxxx
MMU Expanded Code Register	MECR	0x18	R/W	xxxxx000
Memory Timing Control Register	MTCR	0x19	W	xxxx0000
Slave Port Data 0 Register	SPD0R	0x20	R/W	xxxxxxxx
Slave Port Data 1 Register	SPD1R	0x21	R/W	xxxxxxxx
Slave Port Data 2 Register	SPD2R	0x22	R/W	xxxxxxxx
Slave Port Status Register	SPSR	0x23	R	00000000
Slave Port Control Register	SPCR	0x24	R/W	0xx00000
Global ROM Configuration Register	GROM	0x2C	R	0xx00000
Global RAM Configuration Register	GRAM	0x2D	R	0xx00000
Global CPU Configuration Register	GCPU	0x2E	R	0xx00001
Global Revision Register	GREV	0x2F	R	0xx00000
Port A Data Register	PADR	0x30	R/W	xxxxxxxx
Port B Data Register	PBDR	0x40	R/W	00xxxxxx
Port B Data Direction Register	PBDDR	0x47	W	11000000
Port C Data Register	PCDR	0x50	R/W	x0x1x1x1
Port C Function Register	PCFR	0x55	W	x0x0x0x0
Port D Data Register	PDDR	0x60	R/W	xxxxxxxx
Port D Control Register	PDCR	0x64	W	xx00xx00
Port D Function Register	PDFR	0x65	W	xxxxxxxx
Port D Drive Control Register	PDDCR	0x66	W	xxxxxxxx

Port D Data Direction Register	PDDDR	0x67	W	00000000
Port D Bit 0 Register	PDB0R	0x68	W	xxxxxxxx
Port D Bit 1 Register	PDB1R	0x69	W	xxxxxxxx
Port D Bit 2 Register	PDB2R	0x6A	W	xxxxxxxx
Port D Bit 3 Register	PDB3R	0x6B	W	xxxxxxxx
Port D Bit 4 Register	PDB4R	0x6C	W	xxxxxxxx
Port D Bit 5 Register				

2.1.7 Memory Interface and Mapping

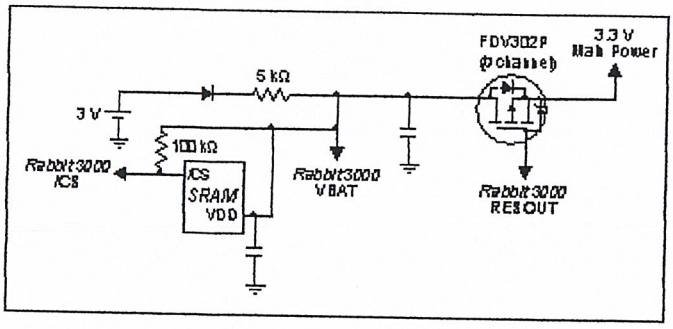
2.1.7.1 Interface for Static Memory Chips

โดยทั่วไปหน่วยความจำแบบสแตติกจะมีแถวแอดเดรส(address),แถวข้อมูล,แถวที่ชิพเลือกแถวที่เป็นเอาท์พุตและเขียนข้อมูลได้ Rabbit 3000 จะมีแถวที่มีลักษณะคล้ายกันซึ่งสามารถเชื่อมต่อไปยังหน่วยความจำแบบสแตติก(Static Memory) ชิพไม่สามารถเลือกได้เพราะ ชิพมีฟังก์ชั่นพิเศษ ในขณะที่การเริ่มทำงาน ไม่สามารถอยู่ในโหมด Cold boot ได้,เริ่มการทำงานที่แอดเดรสเท่ากับ 0 ซึ่งในหน่วยความจำจะติดต่อที่ ขา/CS0 โดยที่แรม(RAM)แบบสแตติกจะเลือกติดต่อที่ขา /CS1 เพราะว่า Dynamic C ยอมให้ สแตติกแรมเชื่อมต่อกับขา /CS1

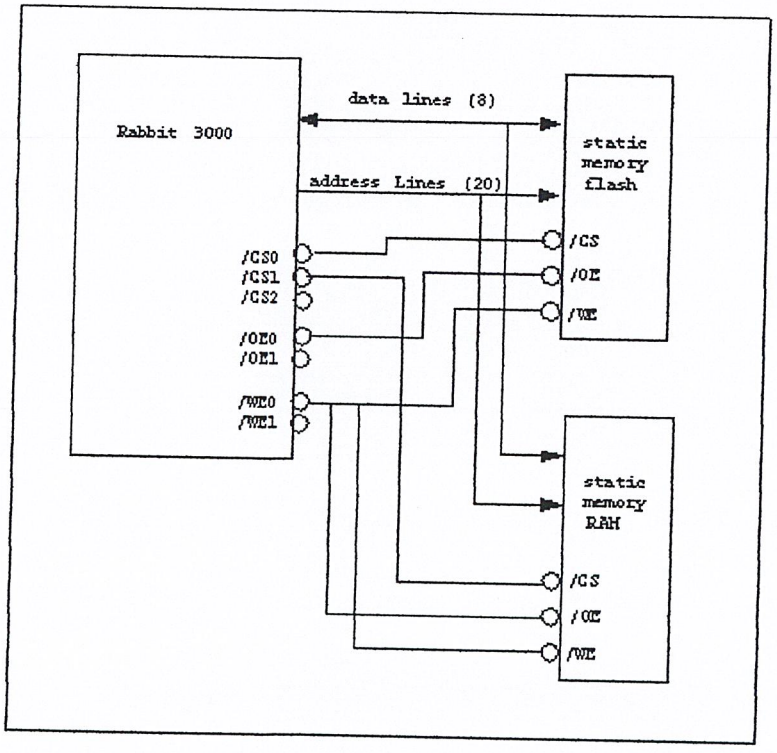
โดยสิ่งที่เพิ่มเข้ามานั้นคือ /cs1 มีความามารถพิเศษช่วยในการสนับสนุนแบตเตอรี่ของสแตติกแรม โดยในขณะที่จบการทำงาน แต่พลังงานของแบตเตอรี่ยังส่งพลังงานไปเลี้ยงที่ขาเพาเวอร์ (VBAT) /CS1 จะทำให้มีค่าอิมพิแดนท์ที่สูงและมีการให้ดึงกระแสกลับไปยัง /CS1 ด้วยเหตุนี้หน่วยความจำแบบสแตติกจะอยู่ในสถานะพร้อมใช้งาน โดยที่ขา RESOUT ยังคงมีค่าที่สูงอยู่

ในขณะที่ กระบวนการนี้จะมีกำลังที่ลดลงและแบตเตอรี่จะส่งพลังงานไปยัง VBAT ซึ่งจะทำให้ RESOUT สามารถควบคุมไฟฟ้าสำหรับโปรเซสเซอร์ และชิพแบบสแตติก

เป็นไปได้ที่จะให้ /CS1สามารถทำงานได้ตลอดเวลา ถ้ามีแบตเตอรี่ภายนอกคอยช่วยเหลือ จะถูกใช้ให้ส่งผ่าน /CS1 อย่างช้าๆ โดยอยู่ในกระบวนของหน่วยความจำ



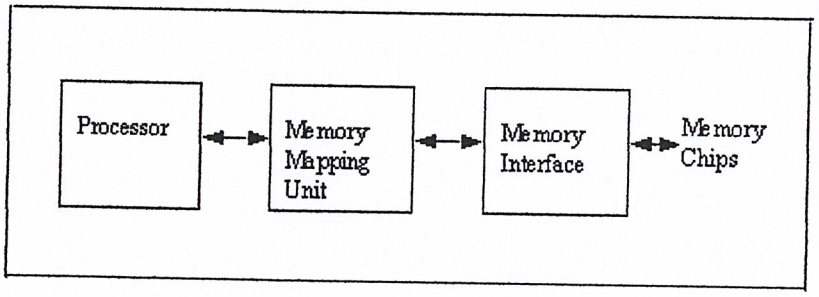
รูปที่ 2.24 Battery-Backup Circuit



รูปที่ 2.25 Typical Memory Chip Connection

2.1.7.2 Memory Mapping Overview

ในรูป8.3 เป็นการแสดงให้เห็นถึงการทำงานของหน่วยความจำ Rabbit ซึ่ง Memory Mapping Unit (MMU) ได้รับข้อมูลที่มีขนาด 16 บิต และแปลงข้อมูลเหล่านั้นเป็นข้อมูลขนาด 20 บิต โดยที่ Memory Interface จะรับข้อมูลขนาด 20 บิตและสร้างสัญญาณควบคุมโดยตรงไปยังชิพของหน่วยความจำ



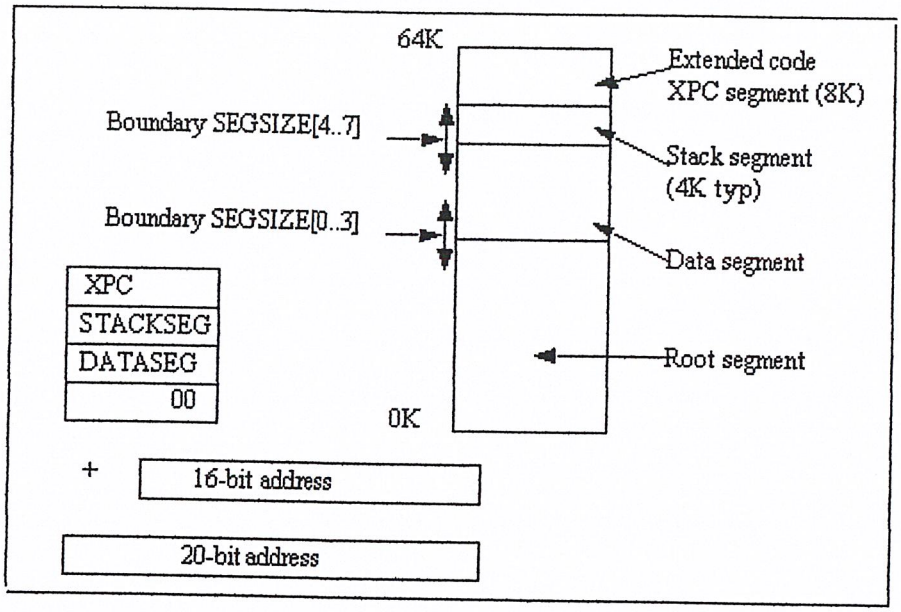
รูปที่2-27Overview of Rabbit Memory Mapping

2.1.7.3 Memory Mapping Unit

โดยที่ข้อมูลขาด 64 k มีขนาด 16บิตเข้ามาโดย คำสังของโปรเซสเซอร์ซึ่งจะถูกแบ่งออกเป็นส่วน โดยแต่ส่วนจะมีขนาด 4K ยกเว้นส่วนที่เป็นส่วนที่เป็นโค๊ต(code) ส่วนนี้จะมีการปรับขนาดและบางส่วนมีการเพิ่มขึ้นจากขนาดที่เป็นศูนย์ และบางส่วนได้ลบออกจากแผนที่แอดเดรส

ข้อมูลแบ่งออกเป็น 4 ส่วนนี้จะแสดงในรูปที่ 8.4 โดย Segment size register (SEGSIZE) กำหนดขนาดในไดอะแกรม ส่วนของการขยายรหัสมักจะเกิดที่แอสเดรส 0E000h-0FFFFh โดยส่วนของสแตกมีการขยายมาจาก แอสเดรสเฉพาะ โดยมาจากค่า 4บิตบนของ SEGSIZE ไปยัง 0DFFFh เช่น ถ้าค่า 4 บิตบนของ SEGSIZE มีค่า 0Dh แล้วส่วนของ สแตก จะเกิดขึ้นที่ 0D000h-0DFFFh, 4K ถ้า4บิตบนของ SEGSIZE ใหญ่กว่าหรือมีค่าเท่ากับ 0Eh ส่วนของสแตกจะหายไป ถ้าบิตเหล่านั้นถูกกำหนดให้มีค่าเท่ากับ 0 , ข้อมูล2 ส่วนที่อยู่ข้างล่าง ส่วนของ สแตกก็จะหายไปด้วย

4บิตล่างของ SEGSIZE ได้กำหนด ขอบเขตล่างที่แสดงอยู่ในรูป ถ้ามีขนาดเท่ากับขอบหรือมากกหว่าหรือมากกว่า 0Eh ส่วนของข้อมูลจะหายไป ถ้าส่วนของข้อมูลเริ่มต้นที่0 ส่วนของโค๊ดก็จะหายไป



รูปที่ 2-27 Memory Segments

MMU จะรับข้อมูลที่มีขนาด 16 บิตจากโปรเซสเซอร์และเปลี่ยนเป็น 20 บิต โดยมีการทำงานดังนี้

1. นำข้อมูลที่มีขนาด 16บิตโดยตรวจสอบ 4บิตบนของ แอดเดรส โดยทุกข้อมูลมีความเป็นไปได้ที่จะต้องเลือก 1 ใน 4 ส่วนนี้คือ 1.XPC 2. STACKSEG 3.DATA SEGMENT 4.00
2. โดยแต่ละส่วนจะเป็นรีจิสเตอร์มีขนาด 8 บิตซึ่งจะนำมารวมเป็น 4 บิตบนของข้อมูง 16 บิต เพื่อที่จะทำให้ข้อมูลมีขนาด 20 บิต

ตารางที่ 2-12 Segment Registers

Segment Register	Function
XPC	ส่วนของตำแหน่งโค๊ตภายนอกในหน่วยความจำ.อ่านและเขียนได้จากกระบวนของชุดคำสั่ง: ld a,xpc, ld xpc,a, lcall, lret, ljp
STACKSEG = 11h	ส่วนของตำแหน่งสแตกในหน่วยความจำทางกายภาพ
DATASEG = 12h	ส่วนของตำแน่งข้อมูลในหน่วยความจำทางกายภาพ

ตารางที่ 2-13 Segment Size Register

	Bits 7..4	Bits 3..0
SEGSIZE = 13h	ขอบเขตสส่วนของตำแหน่งสแตก	ขอบเขตสส่วนของตำแหน่งข้อมูล

2.1.7.4 Memory Interface Unit

ข้อมูลขนาด20บิตจะถูกสร้างโดยMMUส่งไปยัง Memory Interface Unit ซึ่งจะแยกเขียนควบคุมรีจิสเตอร์สำหรับ 256k ซึ่งเป็น ควอแรนท์แรกของ 1M ของหน่วยความจำทางกายภาพ การควบคุมรีจิสเตอร์จะมีการระบุการร้องขอเข้าไปในหน่วยความจำที่ติดต่อกับ Rabbit โดยมีชิพ 3แบบที่เลือกเส้นทางของเอาพุท (/CS0,/CS1,/CS2) สามารถเลือกใช้ชิพโดยเลือกจาก3ชนิดนี้ มีการ

กำหนดเลือกชิพจะถูกเลือกสำหรับให้ข้อมูลผ่านไปยังควอแรนท์ ซึ่งชิพตัวเดียวสามารถส่งได้มากกว่า 1 ควอแรนท์ เช่น ถ้ามีแรมขนาด 512 k ได้ถูกติดตั้งและเลือกขา /CS1 มันจะได้ถูกจัดสรรให้ใช้ /CS1 สำหรับการเข้าไปยัง ควอแรนท์ที่3และ4 ได้ ด้วยเหตุนี้การกำหนด ชิพแรมจะอยู่ระหว่าง 8000h ถึง 0FFFFFh

2.1.7.5 Bank Control Register

ตารางที่ 8-3 จะอธิบายการทำงานของ รีจิสเตอร์ที่คอยควบคุม Bank โดยเป็นรีจิสเตอร์ที่สามารถเขียนได้อย่างเดียว โดยแต่ละรีจิสเตอร์จะควบคุม 1ควอแรนท์ใน 1M ของหน่วยความจำ

ตารางที่ 2-14 Memory Bank Control Register x (MBxCR=14h+x)

Memory Bank x Control Register (MB0CR) (Address = 0x14) (MB1CR) (Address = 0x15) (MB2CR) (Address = 0x16) (MB3CR) (Address = 0x17)		
Bit(s)	**Value**	**Description**
7:6	00	4 สถานะการรอเพื่อเข้าถึง Bank
	01	2 สถานะการรอเพื่อเข้าถึง Bank
	10	1 สถานะการรอเพื่อเข้าถึง Bank
	11	0 สถานะการรอเพื่อเข้าถึง Bank
5	0	ผ่าน A[19] เพื่อเข้าถึง Bank
	1	ไม่ให้ผ่าน A[19] เพื่อเข้าถึง Bank
4	0	ผ่าน A[18] เพื่อเข้าถึง Bank
	1	ไม่ให้ผ่าน A[18] เพื่อเข้าถึง Bank
3:2	00	/OE0 and /WE0 ทำงานเพื่อเข้าถึง Bank
	01	/OE1 and /WE1 ทำงานเพื่อเข้าถึง Bank
	10	/OE0 only ทำงานเพื่อเข้าถึง Bank (i.e. อ่านอย่างเดียว).

		จัดการแบบปกติโดยใช้วิธีการอื่นๆ
	11	/OE1 only ทำงานเพื่อเข้าถึง Bank (i.e. อ่านอย่างเดียว) การจัดการแบบปกติโดยใช้วิธีอื่นๆ
1:0	00	/CS0 is activeทำงานเพื่อเข้าถึง Bank
	01	/CS1 is activeทำงานเพื่อเข้าถึง Bank
	1x	/CS2 is activeทำงานเพื่อเข้าถึง Bank

บิตที่ 7,6 เป็นเลขที่บอกสถานะซึ่งจะใช้ในการเข้าถึง ควอแรนท์ โดยไม่มีสานะการรอ ,การอ่านต้อง 2 ช่วงเวลาและ การเขียนต้องการ 3 ช่วงเวลา โดยสถานะการรอนี้ จะถูกรวมลงไปในเลขจำนวนนี้ด้วย สถานะการรอถูกใช้ในการเข้าถึงข้อมูล,ไม่สำหรับหน่วยความจำที่มาจากชุดคำสั่งที่ได้ถูกวิเคราะห์มา

บิตที่ 5,4 เป็นบิตที่ยอมให้ แอสเดรสบนมีการสลับเปลี่ยนแปลง ซึ่งจะเกิดหลังจากมีการทำลอจิกโดยเลือก bank รีจิสเตอร์ ดังนั้นการตั้งค่าที่บรรทัดนั้นจะไม่มีผลต่อการใช้ bank รีจิสเตอร์ ในทางกลับกันถ้าการติดตั้วหน่วยความจำ 1M มีการจัดทำเพื่อชิพ ที่มีขนาด 256k ถ้าหน่วยความจำมีขนาดใหญ่กว่า 256 k สามารถผ่านได้ทั้ง 4 ควอแรนท์ ไม่มีผลกระทบภายนอกกับควอแรนท์ซึ่ง รีจิสเตอร์ควบคุมหน่วยความจำ bank ควบคุมอยู่

บิตที่ 3 มีการเขียน ฟัลส์ เพื่อให้หน่วยความจำเข้าถึงควอแรนท์ใช้ประโยชน์ในการป้องกันข้อมูลแบบแฟรส จากฟัลส์ที่เขียนผิด

บิตที่ 2 เลือกชุดOEX และ WEX เพื่อการเข้าถึงควอแรนท์

บิตที่1 เป็นการตัดสินใจในการเลือก ชิพ จาก 3 ชิพ ซึ่งจะนำไปใช้เพื่อการเข้าถึง ควอแรนท์โดย ทุกๆบิตของ รีจิสเตอร์ จะถูกทำให้เป็น 0 เมื่อมีการรีเส็ท

2.1.7.5.1 Optional A16,A19 Inversion by Segment (/CS1 Enable)

ในทางกลับกันของ A16 หรือ A19 ได้ถูกควบคุมให้มีการเขียนหรืออ่าน รีจิสเตอร์ MMIDR ซึ่งถูกใช้ในการส่งค่ากลับของส่วนที่เป็น root และ ส่วนที่เป็นข้อมูล โดยการสลับค่าที่ในขณะที่ ส่วนนั่นได้ถูกเข้า

ข้อกำหนดต่างๆของ /CS1 มีค่าที่ใช้ได้สำหรับระบบที่ถูกดึงผ่านเวลาการเข้าของแบตเตอรี่สำรองของแรม โดยที่ทำให้ /CS1 ใช้ได้, การหน่วงเวลาของสวิสช์มีผลทำให้ /CS1 จะมีค่าสูงขึ้น

เมื่อกำลังจะเลิกการใช้งานโดยใช้วิธีการ บายพาส เป็นการเพิ่มพลังงานในการใช้ ตั้งแต่แรมซึ่ง สามารถใช้และควบคุมการเข้าควบคุมโดย /OE1

ตารางที่ 2-15 MMU Instruction/Data Register (MMIDR =010h)		
MMU Instruction/Data Register (MMIDR) (Address = 0x10)		
Bit(s)	**Value**	**Description**
7:6	00	ไม่สนใจบิตและจะคืนค่าเป็น 0 ขณะที่มีการอ่าน
5	0	Enable A16 and A19 inversion independent of instruction/data.
	1	Enable A16 and A19 inversion (controlled by bits 0-3) สำหรับให้เข้าเท่านั้น. สิ่งนี้ทำให้คำสั่งใช้ได้/ข้อมูลที่แบ่ง. ที่แยกออกมา I และ D space
4	0	ทำงานปกติ /CS1
	1	Force /CS1 always active. สิ่งนี้จะไม่เป็นสาเหตุให้ที่ขัดกันใดๆตราบเท่าที่หน่วยความจำที่ใช้ /CS1 ไม่ยังแชร์กับเอาพุตท์หรือใช้เขียนกับหน่วยความจำอื่นๆ
3	0	ทำงานปกติ
	1	สำหรับเข้า DATASEG , ไม่ผ่าน A19 ก่อน MBxCR (bank select) ตัดสินใจ
2	0	ทำงานปกติ
	1	สำหรับเข้า DATASEG ไม่ผ่าน A16
1	0	ทำงานปกติ
	1	สำหรับเข้า root , ไม่ผ่าน A19 ก่อน MBxCR (bank select) ตัดสินใจ
0	0	ทำงานปกติ

	1	สำหรับเข้าroot, ไม่ผ่าน A16

ตารางที่ 2-16 MMU Expanded Code Register (MECR = 18h)

MMU Expanded Code Register (MECR) (Address = 0x18)		
Bit(s)	**Value**	**Description**
7:3		บิตถูกไม่สนใจสำหรับการเขียน และค่าเป็น 0 เมื่อมีการอ่าน
2:0	0xx	ทำงานปกติ
	100	สำหรับ XPC เข้า, .ใช้ MB0CR อย่างอิสระของ f A19-A18.
	101	สำหรับ XPC เข้า, use MB1CR อย่างอิสระของ A19-A18.
	110	สำหรับ XPC เข้า, use MB2CR อย่างอิสระของ A19-A18.
	111	สำหรับ XPC เข้า, use MB3CR อย่างอิสระของ A19-A18.

หน่วยความจำที่ควบคุมเวลา (MTCR) สามารถขยายเวลาสำหรับหน่วยความจำเอาพุตท์ทำงาน และการเขียนทำงาน ดูจากตารางที่ 7-2 สำหรับรายละเอียดไว้อย่างไรของเวลาสำหรับหน่วยความจำของstrobe มีผลอย่างไร ในขณะที่ใช้งาน

ตารางที่ 2-17 Memory Timing Control Register (MTCR, adr = 019h)

Memory Timing Control Register (MTCR) (Address = 0x19)		
Bit(s)	**Value**	**Description**
7:4	xxxx	บิตเหล่านี่ควรเก็บไว้ไม่ตวรใช้
3	0	เวลาปกติสำหรับ /OE1B (ขอบขาขึ้นถึงขอบขาขึ้น , 1 clock minimum).
	1	ขยายเวลาสำหรับ /OE1B (เพิ่ม 1/2 clock มากกว่าแบบปกติ).
2	0	เวลาปกติสำหรับ /OE0B (ขอบขาขึ้นถึงขอบขาขึ้น , 1 clock minimum).
	1	ขยายเวลาสำหรับ /OE0B (เพิ่ม 1/2 clock มากกว่าแบบปกติ).

	1	สำหรับเข้าroot, ไม่ผ่าน A16

ตารางที่ 2-16 MMU Expanded Code Register (MECR = 18h)

MMU Expanded Code Register (MECR) (Address = 0x18)		
Bit(s)	**Value**	**Description**
7:3		บิตถูกไม่สนใจสำหรับการเขียน และค่าเป็น 0 เมื่อมีการอ่าน
2:0	0xx	ทำงานปกติ
	100	สำหรับ XPC เข้า, .ใช้ MB0CR อย่างอิสระของ f A19-A18.
	101	สำหรับ XPC เข้า, use MB1CR อย่างอิสระของ A19-A18.
	110	สำหรับ XPC เข้า, use MB2CR อย่างอิสระของ A19-A18.
	111	สำหรับ XPC เข้า, use MB3CR อย่างอิสระของ A19-A18.

หน่วยความจำที่ควบคุมเวลา (MTCR) สามารถขยายเวลาสำหรับหน่วยความจำเอาพุตท์ทำงาน และการเขียนทำงาน ดูจากตารางที่ 7-2 สำหรับรายละเอียดไว้อย่างไรของเวลาสำหรับหน่วยความจำของstrobe มีผลอย่างไร ในขณะที่ใช้งาน

ตารางที่ 2-17 Memory Timing Control Register (MTCR, adr = 019h)

Memory Timing Control Register (MTCR) (Address = 0x19)		
Bit(s)	**Value**	**Description**
7:4	xxxx	บิตเหล่านี้ควรเก็บไว้ไม่ตวรใช้
3	0	เวลาปกติสำหรับ /OE1B (ขอบขาขึ้นถึงขอบขาขึ้น , 1 clock minimum).
	1	ขยายเวลาสำหรับ /OE1B (เพิ่ม 1/2 clock มากกว่าแบบปกติ).
2	0	เวลาปกติสำหรับ /OE0B (ขอบขาขึ้นถึงขอบขาขึ้น , 1 clock minimum).
	1	ขยายเวลาสำหรับ /OE0B (เพิ่ม 1/2 clock มากกว่าแบบปกติ).

1	0	เวลาปกติสำหรับ /WE1B (ขอบขาลงถึงขอบขาลง, 1 clock minimum).
	1	ขยายเวลาสำหรับ /WE1B (ขอบขาลงถึงขอบขาลง , 2 clock minimum).
0	0	เวลาปกติสำหรับ /WE0B (ขอบขาลงถึงขอบขาลง, 1+1/2 clocks minimum).
	1	ขยายเวลาสำหรับ /WE0B (ขอบขาลงถึงขอบขาลง, 2 clocks minimum).

Breakpoint/Debug ยอมให้ ชุดสั่ง RST28 สามารถใช้ซอฟแวร์ breakpoint โดยปกติ RST28 เป็นสาเหตุทำให้การเรียกที่อยู่ของหน่วยความจำ แต่ในการคำนวณของชุดคำสั่งได้ถูกแก้ไขใน ขณะที่ Breakpoint/debug ได้ทำงาน RST28 เป็นคนดูแล NOP ในโหมด Breakpoint/Debug

ตารางที่ 2-18 Breakpoint/Debug Control Register (BDCR, adr = 01ch)

Breakpoint/Debug Control Register (BDCR) (Address = 0x1C)		
Bit(s)	**Value**	**Description**
7	0	Normal RST 28 operation.
	1	RST 28 is NOP.
6:0		บิตเล่านี้ไม่ควรใช้

2.1.7.6 Allocation of Extended Code and Data

Dynamic C จะประมวลผลโค๊ต ส่งไปยังที่ว่างของrootโค๊ต หรือ ที่ว่างของขยายโค๊ต โดยที่ root โค๊ต จะเริ่มทำงานที่ทำงานตำแหน่งที่ต่ำและ ประมวลต่อมา

การกำหนดตำแหน่งของส่วนขยายโค๊ตจะเริ่มเหนือroot โค๊ตและข้อมูล การเลือกตำแหน่ง โดยปกติจะเริ่มและจบที่หน่วยความจำแบบ แฟลช

ข้อมูลต่าง ถูกกำหนดจากการทำงานของแรมส่งกลับไปยังหน่วยความจำ โดยการติดตั้งเริ่ม จากตำแหน่งที่ 52kใน64k แล้วทำต่อไปเรื่อย 52kจะเริ่มการแชร์กับ root โค๊ตและข้อมูล,และได้ถูก กำหนดขึ้นมาจาก 0

Dynamic C มีการรวมกันระหว่าง extended และ ข้อมูลที่คงที่ ซึ่งจะอยู่ในส่วนของ แฟลช

2.1.7.7 Instruction and Data Support

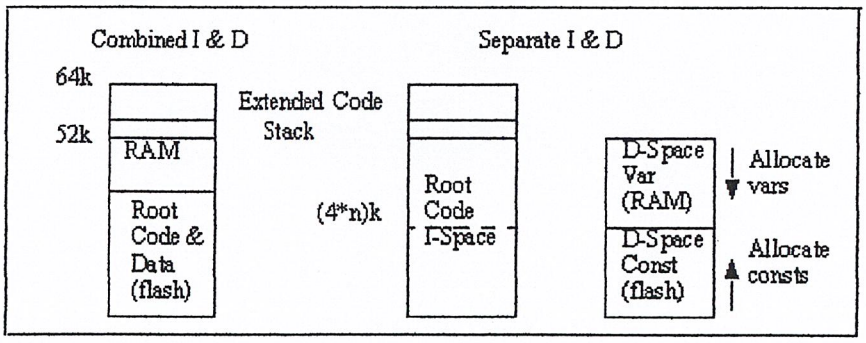
Instruction and Data Support (I and D space) จะทำงานโดยมีการกำหนดการส่งกลับของแอดเดรสที่ A16/A19 ในขณะที่โปรเซสเซอร์ เข้า D space แต่ไม่มีการส่งแถวนั้นกลับ ถ้าโปรเซสเซอร์เข้า I space รีจิสเตอร์ MMIDR ใช้ในการควบคุม ซึ่งเป็นสิ่งที่สำคัญในการเข้าใจ ในบิตตรงข้ามของ A16,A19 ที่การกำหนดโดย IและD space เกิดขึ้นก่อนโดย บิตบน 2 บิตของ 20 บิต จะถูกใช้ในการตัดสินใจกับ ควอแรนท์ และ รีจิสเตอร์ bank ควบคุมการเข้าออกของหน่วยความจำ ความแตกต่างกับบิตี้เป็นส่วนกลับของ Optional แอดเดรส บิตของ A19,A16 จะถูกควบคุมโดย 4 การควบคุมหน่วยความจำ bank หลังจากควอแรนท์ได้ถูกคำนวณเรียบร้อยแล้ว

ตารางที่ 2-19 MMU Instruction/Data Register (MMIDR=010h)

Bits 7:5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
000	1-force /CS1 สามารถทำงานได้	1-Invert A19 สำหรับข้อมูลที่เข้าในส่วนของข้อมูลก่อน เข้าส่วนของควอแรนด์	1-Invert A16 สำหรับข้อมูลที่เข้าส่วนของดาต้า	1-Invert A19 สำหรับข้อมูลที่เข้าในส่วนของ root ก่อนเข้าส่วนควอแรนด์	1-Invert A16 สำหรับข้อมูลที่เข้าส่วนของ root

ยกตัวอย่าง 1Mหน่วยความจำแบบแฟรส ได้ถูกควบคุมโดย /CS0, /WE0, /OE0 เพื่อที่จะทำให้ข้อมูลผ่านไปยัง ควอแรนท์แรก และ MB0CR ทำให้ ไป/CS0และ /WE0หรือ /OE0 เข้าสู่bank แล้วถ้า A18,A19 เท่ากับ 0- 256k แรกของหน่วยความจำแบบแฟรส สามารถเห็นใน 256k ไบต์ แรกของหน่วยความจำแบบกายภาพ ถ้าการเข้าในควอแรนท์ที่ 2 หน่วยความจำไม่สามารเลือก ยกเว้น MB1CR ได้ถูกกำหนดลงไปในหน่วยความจำ แต่อย่างไรก็ตามถ้า A18 ได้ถูกสลับโดยตั้งค่าที่บิตที่ 4ใน MB0CR ไปยัง a1, แล้ว 256kส่วนที่ 2ของ แฟรชยังคงถูกกำหนดลงในควอแรนท์แรก แต่ควอแรนท์แรกไม่สามารถเปลี่ยนได้เพราะการสลับที่จะเกิดได้ก็ต่อเมื่อ ควอแรนท์นั้นได้ถูกเลือกแล้ว การสลับของ A19 หรือ A16 ได้ถูกกำหนดโดย รีจิสเตอร์MMIDR บน D space การเข้าได้ถูกใช้ในการส่งไป I และ D space มีความแตกต่างของที่หน่วยความจำ การกระจายของ I และ D space จะเกิดขึ้นเมื่อ หน่วยความจำ 2ส่วนแรกใน 64 k สำหรับแต่ละส่วน ส่วนรูทโค๊ต และส่วนของข้อมูลอย่างใดอย่างหนึ่ง หรือทั้ง A19 และ A16 สามารถสลับกันได้ เป็นเหตุผลหนึ่งที่เป็นสิ่งที่ทำให้เกิดผลที่ตามมาการกำหนดตำแหน่งของหน่วยความจำทั่วไปจะอยู่ใน หน่วยความจำแบบแฟลช ที่

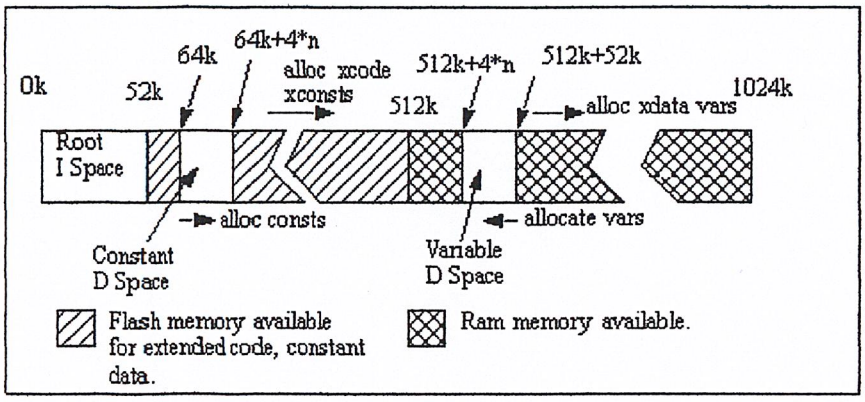
อยู่ต่ำกว่า 512k ทางด้านทางกายภาพของหน่วยความจำ โดยทางกลับกันของ A19 บน D space สามารถเข้าถึงหน่วยความจำที่ต่ำกว่า 512k และสามารถถือครองบน แฟรชและมีการเปลี่ยนแปลงไปยังแรมสำหรับ การเข้าถึงของ D โดยโค๊ตของ A16, D เข้าไปใกล้ 64 k แต่โดยปกติจะไม่ต่ำกว่า 512 k ของหน่วยความจำแบบแฟลช แต่เมื่อดูถึงความแตกต่างของข้อมูลทั้ง 2 ชนิดค่าคงที่จะเก็บไว้ในแฟลช และค่าที่มีการเปลี่ยนแปลงได้เก็บไว้ในแรม เพราะข้อมูลทั้ง2 ชนิดนี้ได้ถูกจองเพื่อแบ่ง D space โดยแบ่งเป็น 2 ส่วน ซึ่งในการรวมของIและ D space โดยมีส่วนของส่วนของ Root ได้ถือครองโค๊ตทั้งคู่และค่าคงที่ในแฟลช ส่วนข้อมูลที่อยู่ในแรม ไม่ต่อเนื่องของI และ D space ส่วนของ root โค๊ตและส่วนของข้อมูลจะถูกกำหนดในบริเวณเดียวกันและสร้างไปเรื่อยๆๆ เริ่มจากล่างสุดของหน่วยความจำแบบ แฟลช ใน I space จะมีการแบ่งระหว่างส่วนของroot และส่วนของข้อมูลไม่การเกี่ยวข้องกัน เพราะว่า รีจิสเตอร์ DATASEG จะมีค่า 0และไม่สามารถแบ่งในทางกายภาพได้ แต่อย่างไรก็ดี ถ้า D space เข้าไปใน A16 ได้เมื่อถูกสลับแล้วสำหรับส่วนของ root ได้ถูกกำหนดใน 64k ต่อไปของ แฟลชและส่วนของข้อมูลจะถูกกำหนดลงไปในหน่วยความจำที่มากกว่า 512k ในแรม โดยการแบ่งข้อมูลเป็น 2 ส่วนที่ไม่ติดกันสำหรับค่าคงที่และค่าที่สามารถเปลี่ยนแปลงได้ ถ้าส่วนของสแตก และส่วนของ extended โค๊ต จะเกิดที่ 12k ที่อยู่บนที่ว่าง 64k แล้วยังอยู่ใน 52k จะมีการำกลับใน 52k ของแฟลชและ 52 ส่วนข้อมูลที่ว่าง ในบางทีการไม่ต่อเนื่องของ 2 ส่วนนี้ มีความสัมพันธ์ของขนาดใน 2 ส่วนนี้ขึ้นอยู่กับ บิตต่ำ 4บิต ของ SEGSIZE โดยหมายความว่า ส่วนละ4kซึ่งอยู่ระหว่าง ส่วนของ root และส่วนของข้อมูล



รูปที่ 2-28 Combined versus Separate I & D Space

การใช้หน่วยความจำทางกายภาพซึ่งจะสดงให้ดูในรูปที่ 2-28 โดยที่ n เป็นจำนวนของ 4k เมื่อสนใจท่ค่าคงที่ D space ในรูปจะยึดเอาหน่วยความจำที่ต่ำกว่า 512k ซึ่งเนส่วนประกอบทั้งหมดของแฟลช และ ที่อยู่บน512k คือ แรม เช่น ถ้า Low-cost 32k*8 แรมซึ่งใช้ในการกำหนดใน ควอแรนท์

ที่3 /CS1 แรมจะเริ่มทำซ้ำไปมาจำนวน 8 ครั้งในควอแรนท์ที่ 3 จากตำแหน่ง ที่ 512k ไปยัง 768k โดยข้อมูลอาจจะมีขนาด 32k แต่จะมีข้อมูลได้มากที่สุดคือ 28k อีก 4k จะเป็นส่วนของสแตกถ้าสแตกถูกต้องการ จะทำให้ผลรวมของค่าต่างๆ ต้องเปลี่ยนไปด้วย



รูปที่ 2-29 Use of Physical Memory Separate I & D Space Model

รูปนี้จะแสดงถึงทิศทางที่ตัวแปรและค่าคงที่ได้ถูกแบ่งเป็น ตัวประมวลผล และ assembled โดยแต่ละลูกศรจะแสดงถึงค่าเริ่มต้นในตำแหน่งของหน่วยความจำ เป็นสิ่งที่สำคัญเพราะ Dynamic C จะแก้ไขปัญหาข้อผิดพลาดของโปรแกรมที่ต้องการเก็บค่าคงที่จำนวนหนึ่งและค่าต่างๆในที่ว่างของข้อมูลและต้องการเข้าถึงข้อมูลโดยไม่คำนึงถึง สถานะของผู้ใช้ โดยการแก้ไขปัญหาของ Dynamic จะมีค่าที่ถูกเก็บไว้เหนือส่วนของข้อมูล โดยจะเริ่มที่ 52 k และทำงานต่อไปเรื่อยๆ ในหน่วยความจำผู้ใช้จะถูกแบ่งโดยตัวคอมไพลเลอร์ เริ่มต้น ซึ่งเริ่มแก้ปัญหาจากด้านล่างของค่าคงที่ของ dynamic C

2.1.8 Parallel Ports

Rabbit มีพอร์ทขนานอยู่ 7พอร์ทโดยแต่ละพอร์ทมีขนาด 8 บิต เช่น A, B, C, D, E, F และ G โดยแต่ละขาจะใช้สำหรับพอร์ทขนานซึ่งใช้กับฟังก์ชั่นต่าง โดยมีคุณสมบัติของแต่ละพอร์ทดังนี้

- พอร์ท A แชร์กับเชื่อมต่อพอร์ทข้อมูลของพอร์ทลูก(slave port)และบัสข้อมูลของ I/O
- พอร์ท B แชร์กับเส้นทางควบคุมสำหรับพอร์ทลูก(slave port), จะเป็นส่วนประกอบของI/O,และเป็น clock I/O สำหรับclockของพอร์ทอนุกรมA และ B
- พอร์ท C แชร์กับพอร์ทข้อมูล I/O แบบอนุกรม
- พอร์ท D มีขนาด 4บิต แชร์กับสลับขา I/O สำหรับพอร์ท อนุกรมA และ B 4 บิต ไม่สามารถแชร์ได้ โดยที่พอร์ท D สามารถแก้ไขให้มีการออกของข้อมูล พอร์ท D

ยังมี Output preload registers ซึ่งสามารถสร้าง clock ไปยังเอาท์พุต รีจีสเตอร์ อยู่ใน timer control สามารถสร้างpulse ได้

- พอร์ท E ทุกบิตของพอร์ท E สามารถปรับแก้เป็น I/O Strobe 4 บิตของพอร์ท E สามารถใช้กับอินเตอร์รัพข้อมูลภายนอก 1บิตของพอร์ทEสามารถแชร์กับชิพพอร์ทลูกที่เลือกเอาไว้ พอร์ท E มี output preload register อยู่ภายใต้การควบคุมของ timer control สำหรับสร้างฟัลส์
- พอร์ท F เป็นเอาต์พุท สามารถหรับค่าให้เปิดทางของเอาพุตท์ พอร์ทขนาน F เอาพุตท์สามารถนำ 4 Pulse-Width Modulation เอาพุตท์ ในด้านอินพุตท์ พอร์ทขนาน F อินพุต?สามารถนำอินพุตท์ไปยัง 2ช่องของตัวถอดรหัสควอแรนด์ พอร์ท F สามารถใช้ขาสัญญาณนาฬิกาสำหรับ พอร์ทอนุกรม C และ D
- พอร์ท G เป็นพอร์ทที่มีแต่เอาท์พุต พอร์ท G สามารถแก้ให้เอาท์พุตออกได้ พอร์ทอินพุตท์และเอาท์พุตของG ที่ถูกใช้สำหรับการเข้าถึงจุดอื่นแบบอนุกรามรอบข้างบนชิพแต่ละอย่างใช้อะซิงโคนัสหรือ การติดต่อแบบSDLC/HDLC
- พอร์ท D-G โดยมีวิธีการเข้าถึงแบบเดียวกันโดยใช้ดิจิตอล I/O

2.1.8.1 Parallel Port A

ตาราง 2-20. Parallel Port A Registers

Register Name	Mnemonic	I/O address	R/W	Reset
Port A Data Register	PADR	0x30	R/W	xxxxxxxx
Slave Port Control Register	SPCR	0x24	R/W	0xx00000

ตารางที่ 2-21 Parallel Port A Data Register Bit Functions

	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
PADR (R/W) adr = 030h	PA7	PA6	PA5	PA4	PA3	PA2	PA1	PA0

รีจิสเตอร์นี้ไม่ควรใช้ในขณะที่ พอร์ทลูกหรือ ส่วนของI/O บัส มีการทำงาน เพราะว่า พอร์ทลูกควบคุมรีจิสเตอร์ซึ่งจะใช้โดยไม่คำนึงถึงถึงพอร์ทขนาน A เมื่อพอร์ททำงานแบบอินพุทต์ จะเก็บข้อมูลไว้ 080h ใน SPCR(slave port control register) มีพอร์ททำงานเป็น output จะเก็บที่ 084h ใน SPCR พอร์ทขนาน A จะเป็นอินพุตท์หลังจากการรีเส็ท

ในขณะที่พอร์ทอ่าน ค่าที่อ่านได้บนขามีค่าเท่ากับ 1 โดยที่1 มีค่าเท่ากับ high และ 0 มีค่าเท่ากับ low

2.1.8.2 Parallel Port B

พอร์ทขนาน B โดยมี 8 ขา สามารถเขียนโปรแกรมได้และสามารถทำเป็นขาอินพุตท์และเอาท์พุตได้พร้อมๆกัน

หลังจากการเซ็ทอัพแล้ว พอร์ทขนาน B จะมีขาที่ 0-5จะเป็นInput แล้วขาที่ PB6 และ PB7 จะเป็น output ซึ่งจะมีค่าเป็น0

ตารางที่ 2-22 Parallel Port B Registers

Register Name	Mnemonic	I/O address	R/W	Reset
Port B Data Register	PBDR	0x40	R/W	00xxxxxx
Port B Data Direction Register	PBDDR	0x47	W	11000000

ตารางที่ 2-23 Parallel Port B Register Bit Functions

	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
PBDR (R/W) adr = 040h	PB7	PB6	PB5	PB4	PB3	PB2	PB1	PB0
PBDDR (W) adr = 047h	dir = out	dir = out	dir = out	dir = out	dir = out	dir = out	dir = out	dir = out

ในขณะที่ส่วนประกอบของดาต้าบัสมีการทำงานอยู่นี้น พอร์ทที่2-7 จัดการ 6แถวข้อมูล โดยที่มีผลคือ 6แถวจาก16แถวหมายความว่า เต็มไปด้วยที่ว่างของI/O

ในขณะที่พอร์ทลูก(slave port) ทำงาน โดยที่พอร์ทขนานPB2-PB7 ได้กำหนดค่าของฟังก์ชั้นต่างๆของพอร์ทลูก แต่อย่างไรก็ตามเป็นสิ่งที่เป็นไดเมนการอ่าน PB0-5 ใช้พอร์ท B คาต้า รีจิสเตอร์ในขณะที่ PB2-PB7 จะถูกใช้สำหรับ พอร์ทลูก พอรีทที่ 6-7 สามารถอ่านสัญญาณที่ขับเคลื่อนมา

PB0 จะทำหน้าที่เป็นจะเป็น Clock ของพอร์ทอนุกรมแบบ A ,PB1 จะทำหน้าที่เป็น Clock ของพอร์ทอนุกรมแบบ B

PBDR- Parallel Port B Data Register Read/Write

PBDDR- Parallel Port B Data Direction Register . A"1" จะทำให้ขานั้นเป็นเอาท์พุต ซึ่งจะเป็นรีจิสเตอร์ที่ไว้สำหรับเขียนเท่านั้น

2.1.8.3 Parallel Port C

Parallel Port C ซึ่งแสดงดังตารางที่ 9-6 โดยที่ 4 พอร์ทอินพุตท์ และ 4 พอร์ทเอาท์พุต พอร์ทที่เป็นเลขคู่คือ PC0, PC2, PC4, PC6 จะเป็นขาเอาท์พุต และพอร์ทที่เป็นเลขคี่ PC1, PC3, PC5, PC7 เป็นขาอินท์พุต ในขณะที่ดาต้ารีจิสเตอร์อ่าน 1,3,5,7 จะส่งข้อมูลไปยังที่ขา โดยที่บิต 0,2,4,6 จะส่งค่าของสัญญาณเพื่อที่จะทำให้เป็นเอาท์พุตปัฟเฟอร์ โดยแต่ละดาต้ารีจิสเตอร์ของพอร์ท C จะถูกขับไปยังขาหรือพอร์ทใดพอร์ทหนึ่งของ พอร์ทในการส่งแบบอนุกรม ค่าบิตใน PCFR (Port C Function Register) ถ้าดาต้ารีจิสเตอร์หรือพอร์ทที่มีการส่งจะส่งไปยังขานั้น

ตารางที่ 2-24 Parallel Port C Registers

Register Name	Mnemonic	I/O address	R/W	Reset
Port C Data Register	PCDR	0x50	R/W	x0x1x1x1
Port C Function Register	PCFR	0x55	W	x0x0x0x0

ตารางที่ 2-25 Parallel Port C Register Bit Functions

	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0

PCDR (r) adr = 050h	PC7 in	Echo drive	PC5 in	Echo drive	PC3 in	Echo drive	PC1 in	Echo drive
PCDR (w) adr = 050h	x	PC6	x	PC4	x	PC2	x	PC0
PCFR (w) adr = 055h	x	Drive TXA	x	Drive TXB	x	Drive TXC	x	Drive TXD

พอร์ทขนาน C จะแชร์โดยใช้ขาเดียวกับอนุกรม พอร์ท A-D โดยที่อินท์พุตบนพอร์ทขนาน สามารถปรับเป็นพอร์ทอินพุตท์แบบอนุกรม ในขณะที่ให้ เอาพุตท์เป็นต่อแบบอนุกรม

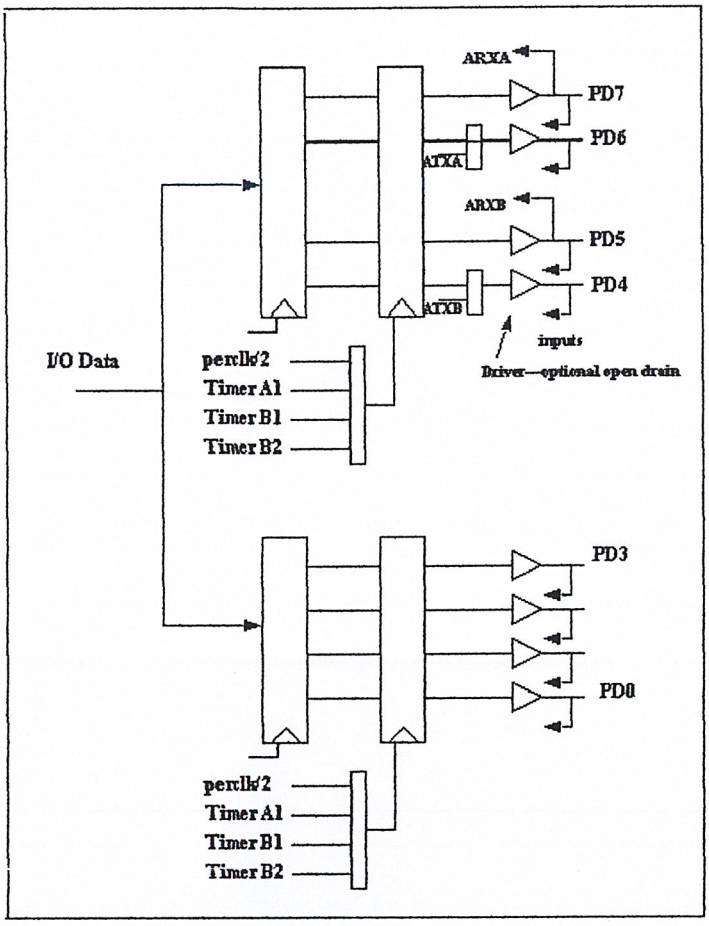
ดาต้าไลน์ยังสามารถอ่านขา PCDR(Parallel C Data Register) พอร์ทเอาพุตท์แบบขนาน สามาระถเลือกให้ออกไปยังพอร์ทแบบอนุกรมได้โดยปรับค่าที่ PCFR ในขณะที่พอร์ทเอาท์พุต แบบขนานจะถูกเลือกให้ออกพอร์ทอนุกรมโดยไม่สนใจค่าที่เก็บไว้ใน ดาต้ารีจิสเตอร์

2.1.8.4 Parallel Port D

พอร์ท D แบบขนาน จะมี 8 ขาซึ่งสามารถโปรแกรมได้ โดยสามารถแบ่งเป็น อินต์พุตและเอาท์พุต ในขณะที่โปรแกรมอยู่นั้น ขาสามารถแบ่งได้โดยการปิดพอร์ท เอาท์พุตหรือเอาพุตท์แบบปกติ ขาของพอร์ท เอาท์พุตสามารถกำหนดตำแหน่งโดยบิต ถ้ามีการออกแบบ พอร์ท D บิตที่ 4และ5 สามารถเลือกใช้สำหรับพอร์ทB แบบอนุกรม และ บิตที่ 6 และ7 สามารถเลือกใช้สำหรับพอร์ท A แบบอนุกรม

การรีเส็ท ดาต้าไดเร็กรีจิสเตอร์ จะเป็น0 รวมทั้งขาอินพุตท์ด้วย บิตที่อยู่ในรีจิสเตอร์จะ หรับค่าเป็น 0 เพื่อทำให้แน่ใจว่าข้อมูลถูกเข้าไปในรีจิสเตอร์คือสิ่งที่นำออกเมื่อโหลด. รีจิสเตอร์ อื่นๆทั้งหมดที่เกี่ยวข้องกับพอร์ทจะไม่ถูกทำการรีเส็ท.

ตารางที่ 2-26 Parallel Port D Registers				
Register Name	**Mnemonic**	**I/O address**	**R/W**	**Reset**
Port D Data Register	PDDR	0x60	R/W	xxxxxxxx
Port D Control Register	PDCR	0x64	W	xx00xx00
Port D Function Register	PDFR	0x65	W	xxxxxxxx
Port D Drive Control Register	PDDCR	0x66	W	xxxxxxxx
Port D Data Direction Register	PDDDR	0x67	W	00000000
Port D Bit 0 Register	PDB0R	0x68	W	xxxxxxxx
Port D Bit 1 Register	PDB1R	0x69	W	xxxxxxxx
Port D Bit 2 Register	PDB2R	0x6A	W	xxxxxxxx
Port D Bit 3 Register	PDB3R	0x6B	W	xxxxxxxx
Port D Bit 4 Register	PDB4R	0x6C	W	xxxxxxxx
Port D Bit 5 Register	PDB5R	0x6D	W	xxxxxxxx
Port D Bit 6 Register	PDB6R	0x6E	W	xxxxxxxx
Port D Bit 7 Register	PDB7R	0x6F	W	xxxxxxxx



รูปที่ 2-30 Parallel Port D Block Diagram

ตารางที่ 2-27 Parallel Port D Register functions

	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
PDDR (R/W) adr = 060h	PD7	PD6	PD5	PD4	PD3	PD2	PD1	PD0
PDDCR (W) adr =	out = open drain	out = open drain	out = open drain	out = open drain	out = open drain	out = open drain	out = open drain	out = open drain

066h								
PDFR (W) adr = 065h	x	alt TXA	x	alt TXB	x	x	x	x
PDDDR (W) adr = 067h	dir = out	dir = out	dir = out	dir = out	dir = out	dir = out	dir = out	dir = out
PDB0R (W) adr = 068h	x	x	x	x	x	x	x	PD0
PDB1R (W) adr = 069h	x	x	x	x	x	x	PD1	x
PDB2R (W) adr = 06Ah	x	x	x	x	x	PD2	x	x
PDB3R (W) adr = 06Bh	x	x	x	x	PD3	x	x	x
PDB4R (W)	x	x	x	PD4	x	x	x	x

adr = 06Ch								
PDB5R (W) adr = 06Dh	x	x	PD5	x	x	x	x	x
PDB6R (W) adr = 06Eh	x	PD6	x	x	x	x	x	x
PDB7R (W) adr = 06Fh	PD7	x	x	x	x	x	x	x

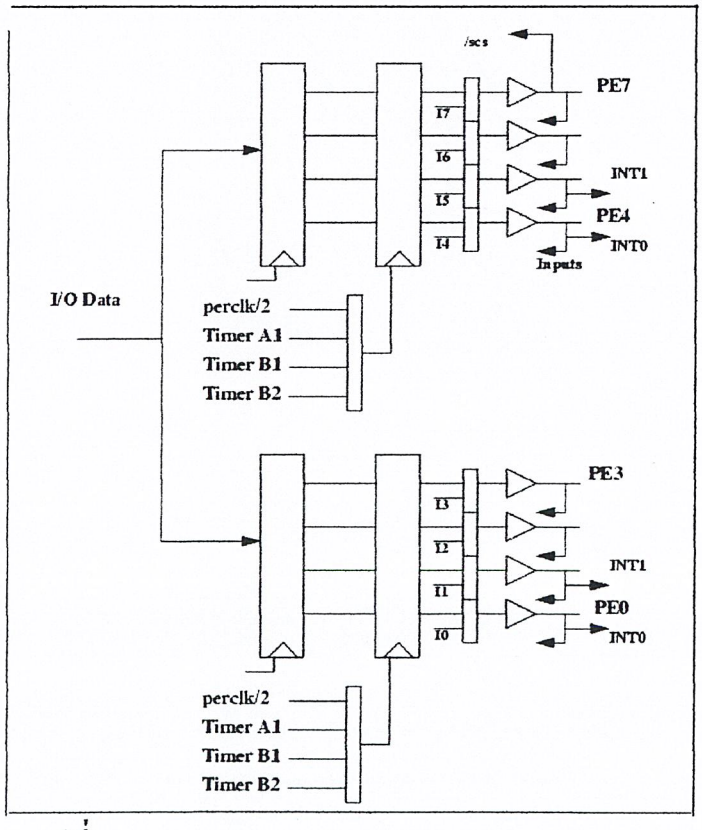
ตารางที่ 2-28 Parallel Port D Control Register (adr = 064h)			
Bits 7, 6	**Bits 5, 4**	**Bits 3, 2**	**Bits 1, 0**
x,x	00--clock upper nibble on pclk/2 01--clock on timer A1 10--clock on timer B1 11--clock on timer B2	x,x	00--clock lower nibble on pclk/2 01--clock on timer A1 10--clock on timer B1 11--clock on timer B2

- PDDR--Parallel Port D data register. Read/Write.
- PDDDR--Parallel Port D data direction register. A "1"ทำให้ขาเป็นเอาพุตท์ และสามารถเขียนได้เท่านั้น
- PDDCR--Parallel Port D drive control register. A "0" ทำให้ขาเป็นเอาพุตท์และt. A "1" ทำให้ขานั้นสามารถส่งข้อมูลออกไปด้วย แต่สามารถเขียนได้เท่านั้น

- PDFR--Parallel Port D function control register. พอร์ทนี้สามารถเป็นพอร์ทเอาพุตท์โดยมีตำแหน่งที่4และ6 เป็นพอร์ทแบบอนุกรม เขียนได้เท่านั้น
- PDBxR—เป็นพอร์ทขนาด 8 บิตใช้ในการตั้งเป็นเอาพุตท์เป็นพอร์ทเฉพาะ
- PDCR--Parallel Port D control register. เป็นรีจิสเตอร์ที่ถูกใช้สำหรับการควบคุมสัญญาณนาฬิกาของส่วนบนและส่วนล่างแบ่งออกเป็นส่วนๆของเอาพุตท์รีจิสเตอร์ตัวสุดท้าย มีการเซ็ท ค่าที่บิต 0, 1 , 4 และ 5 จะมีค่าเป็น 0

2.1.8.5 Parallel Port E

Parallel Port E จะมีขา I/O ทั้งหมด 8ขาและสามารถเขียนโปรแกรมและแบ่งเป็นขา 0t อินพุตท์และเอาพุตท์ โดยที่ PE7 ถูกใช้เมื่อ port ลูกเลือกไว้ในขณะที่ พอร์ทลูกกำลังทำงานอยู่โดยแต่ละพอร์ทเอาพุตท์ของ E สามารถปรับเปลี่ยนเป็น I/O strobes 4 ใน พอร์ท E จะถูกใช้เป็นอินเตอร์รัพ รีเวส อินท์พุต (Interrupt request Input) เอาท์พุตรีจิสเตอร์ ถูกส่งออกไปและใช้เวลาในการควบคุม เป็นสิ่งที่ทำได้เพื่อการสร้าง precise timing pulse



รูปที่ 2-31 Parallel Port E Block Diagram

ตารางที่ 2-29 Parallel Port E Registers				
Register Name	**Mnemonic**	**I/O address**	**R/W**	**Reset**
Port E Data Register	PEDR	0x70	R/W	xxxxxxxx
Port E Control Register	PECR	0x74	W	xx00xx00
Port E Function Register	PEFR	0x75	W	00000000
Port E Data Direction Register	PEDDR	0x77	W	00000000
Port E Bit 0 Register	PEB0R	0x78	W	xxxxxxxx
Port E Bit 1 Register	PEB1R	0x79	W	xxxxxxxx
Port E Bit 2 Register	PEB2R	0x7A	W	xxxxxxxx
Port E Bit 3 Register	PEB3R	0x7B	W	xxxxxxxx
Port E Bit 4 Register	PEB4R	0x7C	W	xxxxxxxx
Port E Bit 5 Register	PEB5R	0x7D	W	xxxxxxxx
Port E Bit 6 Register	PEB6R	0x7E	W	xxxxxxxx
Port E Bit 7 Register	PEB7R	0x7F	W	xxxxxxxx

- PEDR--Port E data register. อ่านค่าต่างๆที่ขา เขียนที่พอร์ท E preload register.
- PEDDR--Port E data direction register. มีค่าเท่ากับ "1" ทำให้มีการติดต่อที่ขา คล้ายกับเอาพุทต์ รีจีสเตอร์มีค่าเท่ากับ 0 หลังจากการเซ็ท
- PEFR--Port E function register. มีค่าเท่ากับ "1" ทำให้เอาพุตม์ออกมามีลักษณะคล้ายกับI/O strobe. โดยธรรมชาติของ I/O strobe ที่ถูกควบคุมโดย the I/O bank control registers (IBxCR). เส้นทางของข้อมูลจะถูกตั้งค่าเพื่อเป็นเอาพุตท์ของของ I/O strobe
- PEBxR—เป็นรีจิสเตอร์เฉพาะเมื่อตั้งค่าเฉพาะบิตเอาพุตท์

- PECR--Parallel Port E control register. เป็นรีจิสเตอร์ที่ใช้ควบคุมการทำงานของสัญญาณนาฬิกาของส่วนบนและส่วนล่างแบ่งออกเป็นส่วนๆของเอาพุตท์รีจิสเตอร์ตัวสุดท้าย มีการเซ็ท ค่าที่บิต 0, 1 , 4 และ 5 จะมีค่าเป็น 0

โดยหลังจากการรีเซ็ท รีจิสเตอร์ดาต้าได้เร็กชั่นและฟังก์ชั่นรีจิสเตอร์จะปรับค่าเท่ากับ 0 โดยจะทำให้ทุกขาและจะทำการผิดเอาท์พุตฟังก์ชั้นอื่นๆ ด้วย นอกจากนี้บิตที่อยู่ในรีจิสเตอร์ควบคุมจะมีค่าเท่ากับ "0" (บิต 0, 1, 4, 5) เป็นที่แน่นอน ข้อมูลจะถูกclock ส่งไปยังเอาท์พุต รีจิสเตอร์ในขณะที่มีการ โหลด รีจิสเตอร์อื่นๆที่เกี่ยวข้องกับ พอร์ท E จะไม่มีผลกระทบหลังจากการรีเซ็ท

ตารางที่ 2-30 Parallel Port E Register functions

	Bit 7	**Bit 6**	**Bit 5**	**Bit 4**	**Bit 3**	**Bit 2**	**Bit 1**	**Bit 0**
PEDR (R/W) adr = 070h	PE7	PE6	PE5	PE4	PE3	PE2	PE1	PE0
PEFR (W) adr = 075h	alt /I7	alt /I6	alt /I5	alt /I4	alt /I3	alt /I2	alt /I1	alt /I0
PEDDR (W) adr = 077h	dir = out	dir = out	dir = out	dir = out	dir = out	dir = out	dir = out	dir = out
PEB0R (W) adr = 078h	x	x	x	x	x	x	x	PE0
PEB1R (W) adr = 079h	x	x	x	x	x	x	PE1	x
PEB2R (W) adr = 07Ah	x	x	x	x	x	PE2	x	x
PEB3R (W) adr = 07Bh	x	x	x	x	PE3	x	x	x

PEB4R (W) adr = 07Ch	x	x	x	PE4	x	x	x	x
PEB5R (W) adr = 07Dh	x	x	PE5	x	x	x	x	x
PEB6R (W) adr = 07Eh	x	PE6	x	x	x	x	x	x
PEB7R (W) adr = 07Fh	PE7	x	x	x	x	x	x	x

ตารางที่ 2-31 Parallel Port E Control Register (adr = 074h)

Bits 7, 6	Bits 5, 4	Bits 3, 2	Bits 1, 0
x,x	00--clock upper nibble on pclk/2 01--clock on timer A1 10--clock on timer B1 11--clock on timer B2	x,x	00--clock lower nibble on pclk/2 01--clock on timer A1 10--clock on timer B1 11--clock on timer B2

2.1.8.6 Parallel Port F (PPF)

PPF เป็นพอร์ทที่สามารถกำหนดแต่ละบิตสำหรับกำหนดเส้นทางและไดร์ว เป็นเรื่องปกติของอินพุตท์และควบคุมเอาพุตท์และบันทึกลงใน PFDR ,เป็นเอาท์พุต, บิตของพอร์ทที่เก็บในบัฟเฟอร์ ด้วยข้อมูลที่ถูกเขียนสำหรับ PFDR (Port C Data Register) ส่งไปยังขาเอาท์พุตที่ขอบขา, g เอาพุตท์ของ A1 , Timer B1 และ Timer B2 สามารถใช้สำหรับฟังก์ชั่น โดยแต่ละช่วงของพอร์ทจะมีการเลือกบริเวณในการควบคุมไทเมอร์

ซึ่งอินพุตท์และเอาท์พุตจะใช้สำหรับการเข้า-ออกส่งไปยังส่วนต่างบริเวณชิพ เช่น เอาท์พุต , PPF(Parallel Port F)ของเอาพุตท์ สามารถนำเอาพุตท์ซึ่งมีค่าเท่ากับ 4 Pulse-Wide Modulation เอาท์พุต อินพุตท์ จะมีข้อมูลส่งไปยังควอแรนท์ ถอดรหัส ในขณะที่ SPC(Serial Port C) และ SPD(Serial Port D) ถูกใช้ในโหมดสัญญาณนาฬิกาแบบอนุกรม ,2ขา ของ PDF ซึ่งใช้ในการส่ง

สัญญาณserial clock ในขณะที่ clock ภายใน จะถูกเลือกใน พอร์ทอนุกรม เป็นผลตอบสนองของ PPF จะถูกกำหนดเป็นเอาท์พุท

ตารางที่ 2-32 Parallel Port F Registers

Register Name	Mnemonic	I/O address	R/W	Reset
Port F Data Register	PFDR	0x38	R/W	xxxxxxxx
Port F Control Register	PFCR	0x3C	W	xx00xx00
Port F Function Register	PFFR	0x3D	W	xxxxxxxx
Port F Drive Control Register	PFDCR	0x3E	W	xxxxxxxx
Port F Data Direction Register	PFDDR	0x3F	W	00000000

ตารางที่ 2-33 Parallel Port F Register Functions

	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
PFDR (R/W) adr = 038h	PF7	PF6	PF5	PF4	PF3	PF2	PF1	PF0
PFFR (W) adr = 03Dh	pwm[3]	pwm[2]	pwm[1]	pwm[0]	x	x	sclk_c	sclk_d
PFDCR	out =	out =	out =	out =	out =	out =	out =	out =

(W) adr = 03Eh	open drain	open drain	open drain	open drain	open drain	open drain	open drain	open drain
PFDDR (W) adr = 03Fh	dir = out	dir = out	dir = out	dir = out	dir = out	dir = out	dir = out	dir = out

ตารางที่ 2-34 Parallel Port F Control Register (adr = 03Ch)

Bits 7, 6	Bits 5, 4	Bits 3, 2	Bits 1, 0
x,x	00--clock upper nibble on pclk/2 01--clock on timer A1 10--clock on timer B1 11--clock on timer B2	x,x	00--clock lower nibble on pclk/2 01--clock on timer A1 10--clock on timer B1 11--clock on timer B2

- PFDR--Port F data register. อ่านค่าต่างๆที่ขา เขียนไปยังพอร์ท F preload register.
- PFCR--Parallel Port F control register. เป็นรีจิสเตอร์ที่ถูกใช้สำหรับการควบคุมสัญญาณนาฬิกาของส่วนบนและส่วนล่างแบ่งออกเป็นส่วนๆของเอาพุตท์รีจิสเตอร์ตัวสุดท้าย มีการเซ็ท ค่าที่บิต 0, 1 , 4 และ 5 จะมีค่าเป็น 0
- PFFR--Port F function register. ตั้งค่าเป็น1 สำหรับการทำงานหลายฟังก์ชั่นเอาพุตท์ บิตที่ 7-4 ทำงานเอาพุตท์ PWM และบิตที่ 1-0 ทำงานเป็นอะซิงโครนัสของพอร์ท C and D สัญญาณเอาพุทต์สำหรับพอร์ทอนุกรมที่ต้องเปลี่ยนค่าสำหรับการสร้างสัญญาณนาฬิกาภายใน
- PFDCR--Parallel Port F drive control register. ค่า A "0" ทำให้ขาที่คล้ายคลึงกับเอาพุตท์ A “1” ทำขาให้เปิดพอร์ทให้เอาพุตท์ออกไปได้ เขียนเท่านั้น
- PFDDR--Port F data direction register. มีค่าเท่ากับ "1" ทำให้มีการติดต่อที่ขาคล้ายกับเอาพุทต์ รีจีสเตอร์มีค่าเท่ากับ 0 หลังจากการเซ็ท

2.1.8.6.1 Using Parallel Port A and Parallel Port F

มีข้อผิดพลาดเกิดขึ้นในRabbit 3000 มีผลมาจากการขัดแย้งระหว่าง Parallel Port A (PPA) กับ Parallel Port F (PPF) โดยที่ ข้อผิดพลาดนี้เกิดจากตำแหน่งของข้อมูล ของ รีจิสเตอร์ข้อมูลขาออกของ PPA จะมีขนาด 16 แอดเดรสตั้งแต่ 30 ถึง 3F (Hex) โดยที่ PPF จะใช้ข้อมูลที่ 38 ถึง3F และการถอดรหัสของ PPAจะไม่มีการปรับค่าด้วย

Parallel Port A	Parallel Port F
Parallel Inputs	Full Functionality
Parallel Outputs	Parallel Inputs, PWM, Serial Port Clocks
Slave Port	Parallel Inputs, PWM, Serial Port Clocks
Auxiliary I/O Bus	Full Functionality

ตารางที่ 2-35 สาเหตุที่ทำให้เกิดข้อผิดพลาด

- ถ้ามีการใช้งานของ auxiliary I/O bus ซึ่งใช้ PPA แล้ว ข้อผิดพลาดนี้จะไม่เกิดขึ้น และสามารถใช้ฟังก์ชั่นบน PPF ได้
- ถ้าให้ PPA รับอินพุทต์ แล้วข้อผิดพลาดนี้ไม่แสดงออกมาและสามารถใช้ฟังก์ชั่นของPPFที่มีค่า
- ถ้าใช้ PPA เป็นเอาท์พุต แล้วจะไม่สามารถใช้ PPF เป็นเอาท์พุต ยกเว้นใช้ PWM และสัญญาณนาฬิกาเอาท์พุตซึ่งเตรียมทราบถึงการเขียนสำหรับรีจิสเตอร์ควบคุมของ PPF โดยเขียนข้อมูลไปยังรีจิสเตอร์ข้อมูลที่ส่งออกของ PPA โดยปกติจะแก้ได้โดยการออกPPA จนกระทั่ง การเซ็ทค่าของ PPF และสวิตซ์ของ PPA เป็นเอาท์พุต จะสามารถใช้ขาบน PPF เป็นอินพุตท์
- ถ้าใช้พอต์ลูก แล้วไม่สามารถ PPF เป็น ออกเป็นขนานแต่เรายังสามารถใช้ฟังก์ชั่นอื่นของ PPF ได้

เป็นการง่ายที่จะหลีกเลี่ยงปัญหาในขณะที่มีการขัดแย้งเราต้องกำหนดค่าของอินพุตท์ และเอาท์พุตจะหลีกเลี่ยงข้อบกพร่องได้ พอร์ทขนานA สามารถถูกใช้เป็นอินพุตท์, ซึ่ง PPF กรณีมีฟังก์ชันเต็ม, หรือถ้าพอร์ทขนานA ไม่สามารถถูกใช้เป็นอินพุตท์, ใช้เขาใดๆบน PPF ไม่ใช้สำหรับ

PWM หรือสิ่งที่นำออกนาฬิกาเป็นชุดเป็นสิ่งที่นำเข้าและใช้ precaution ของการตั้งค่าขึ้น PPF การขัดแย้งกันเกี่ยวกับฟังก์ชั่นของ PPF ถูกทำให้ใช้ได้

2.1.8.7 Parallel Port G

PPG เป็นพอร์ทที่สามารถกำหนดแต่ละบิตสำหรับกำหนดเส้นทางและไดร์วเป็นเรื่องปกติของอินพุตท์และควบคุมเอาพุตท์และบันทึกลงใน PGDR, เป็นเอาท์พุต, บิตของพอร์ทที่เก็บในบัฟเฟอร์ ด้วยข้อมูลที่ถูกเขียนสำหรับ PGDR (Port G Data Register) ส่งไปยังขาเอาท์พุตที่ขอบขา, เอาพุตท์ของ A1 , ไทเมอร์ B1 และไทเมอร์ B2 สามารถใช้สำหรับฟังก์ชั่น โดยแต่ละช่วงของพอร์ทจะมีการเลือกบริเวณในการควบคุมไทเมอร์

ซึ่งอินพุตท์และเอาท์พุตจะใช้สำหรับการเข้า-ออกส่งไปยังส่วนต่างบริเวณชิพ เหมือนกับเอาพุตท์ ,โดยที่ พอร์ท G ได้ส่งข้อมูลและสัญญาณนาฬิกาเอาพุตท์จาก Serial Port E (SPE) และ F(SPF) อินพุตท์ ,โดยที่พอร์ท G ได้ส่งข้อมูลและสัญญาณนาฬิกา อินพุตท์สำหรับ 2 พอร์ทขนาน

ตารางที่ 2-36 Parallel Port G Registers

Register Name	Mnemonic	I/O address	R/W	Reset
Port G Data Register	PGDR	0x48	R/W	xxxxxxxx
Port G Control Register	PGCR	0x4C	W	xx00xx00
Port G Function Register	PGFR	0x4D	W	xxxxxxxx
Port G Drive Control Register	PGDCR	0x4E	W	xxxxxxxx
Port G Data Direction Register	PGDDR	0x4F	W	00000000

ตารางที่ 2-37 Parallel Port G Data Register Functions

	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
PGDR (R/W)	PG7	PG6	PG5	PG4	PG3	PG2	PG1	PG0

adr = 048h								
PGFR (W) adr = 04Dh	x	SOUT_E	RCLK_E	TCLK_E	x	SOUT_F	RCLK_F	TCLK_F
PGDCR (W) adr = 04Eh	out = open drain	out = open drain	out = open drain	out = open drain	out = open drain	out = open drain	out = open drain	out = open drain
PGDDR (W) adr = 04Fh	dir = out	dir = out	dir = out	dir = out	dir = out	dir = out	dir = out	dir = out

ตารางที่ 2-38 Parallel Port G Control Register (adr= 04Ch)			
Bits 7, 6	**Bits 5, 4**	**Bits 3, 2**	**Bits 1, 0**
x,x	00--clock upper nibble on pclk/2 01--clock on timer A1 10--clock on timer B1 11--clock on timer B2	X,x	00--clock lower nibble on pclk/2 01--clock on timer A1 10--clock on timer B1 11--clock on timer B2

- PGDR--Port G data register. อ่านค่าต่างที่ขา เขียนไปยังพอร์ท G preload register.
  - PGCR--Parallel Port G control register. เป็นรีจิสเตอร์ที่ใช้ควบคุมการทำงานของสัญญาณนาฬิกาของส่วนบนและส่วนล่างแบ่งออกเป็นส่วนๆของเอาพุตท์รีจิสเตอร์ตัวสุดท้าย มีการเซ็ท ค่าที่บิต 0, 1 , 4 และ 5 จะมีค่าเป็น 0
- PGFR--Port G function register. ตั้งบิทให้ "1" เพื่อทำให้ฟังก์ชันเอาพุตท์สลับกันใช้ได้. บิต 6 และ2 ทำให้ พอร์ทอนุกรมอะซิงโครนัส หรือ SDLC/HDLCใช้ได้ E

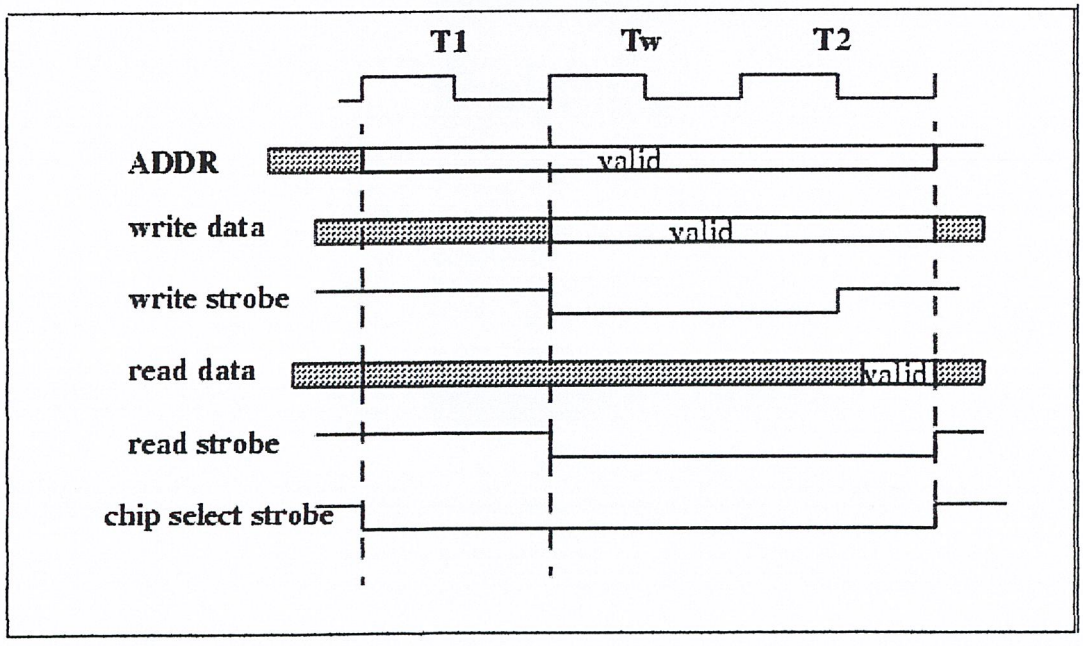
และเอาท์พุต F. และบิท 5-4 และ1-0 ทำให้ SDLC/HDLCใช้ได้ ส่งและรับเอาพุตท์ของสัญญาณนาฬิกาสำหรับพอร์ทเป็นชุดE และ F

- PGDCR--Parallel Port G drive control register. "0" ทำขาที่คล้ายคลึงอินพุตท์ที่ออกประจำ "1" ทำขาที่คล้ายขาเอาพุตท์เปิด. เขียนเท่านั้น
- PGDDR--Port G ดาต้าไดเร็กรีจิสเตอร์มีค่าเท่ากับ "1" ทำให้มีการติดต่อที่ขาคล้ายกับเอาพุทต์ รีจีสเตอร์มีค่าเท่ากับ 0 หลังจากการเซ็ท

ดาต้าไดเร็กรีจิสเตอร์เท่ากับ 0, ทำทุกขาที่เป็นขาอินพุตท์ บิตที่มีรู้แน่นอนจะอยู่ในรีจิสเตอร์ควบคุมเท่ากับ 0 (bits 0, 1, 4, 5) ข้อมูลจะเป็นสัญญาณนาฬิกาที่ถูกส่งไปยังเอาพุตท์รีจิสเตอร์ในขณะที่มีการโหลด รีจิสเตอร์ที่เกี่ยวข้องกับพอร์ท G จะไม่ได้ทำการรีเส็ท

2.1.9 I/O Bank Control Registers

ขาของพอร์ท E สามารถแบ่งเป็น I/O strobe โดยแต่ละ 8 Strobe จะมีรีจิสเตอร์ควบคุมโดยที่ควบคุม strobe และระยะเวลาคอยโดยใส่ค่าที่ขาลงไปในบัส I/O Cycle การเขียนจะถูกปิดบังสำหรับ strobe อื่นชนิดของ strobe สามารถดูที่รูป 10-1 แต่ละ 8 I/O strobe จะทำงานสำหรับตำปน่งที่เกิดขึ้น 1/8 th ของ 64k ของ พื้นที่ว่างของแอดเดรส I/O ที่อยู่ภายนอก



รูปที่ 2-32. External I/O Bus Cycles

ตารางที่ 2-39 I/O Bank Control Reg (adr IBxCR = 08xh)			
Bits 7,6	**Bits 5,4**	**Bit 3**	**Bits 2-0**
Wait state code 11-1 10-3 01-7 00-15	/IX strobe type 00—ชิพเลือก 01—อ่าน strobe 10—เขียน strobe 11—อ่าน หรือ เขียน strobe	1--permit write 0--inhibit write	Ignored

8 I/O bankรีจิสเตอร์ควบคุมได้กำหนดจำนวนของ สถานะการรอI/Oที่ประยุกต์การเข้าถึง I/O ภายนอกในพื้นที่ที่ถูกควบคุมจากแต่ละรีจีสเตอร์แม้ว่าเกี่ยวกับ strobe จะไม่สามารถใช้ได้

คอนโทรลเหนือการสร้างของสถานะการรออิสระของไม่ว่าหรือไม่เกี่ยวข้อง strobe ในพอร์ท E ทำงานอยู่ 2 บิตบนของแต่ละรีจิสเตอร์ได้กำหนดจำนวนของสถานะรอ มี 4 อย่างให้เลือก คือ 1,2,7และ15 ในการรีเส็ท บิตจะถูกเคลียร์ทำให้ได้ 15 สถานะการรอ มีอย่างน้อย 1 เป็นสถานะการรอI/O และค่าที่ต่ำที่สุดของ I/O ภายนอกวัฏจักรอ่านใช้เวลา 3 สัญญาณนาฬิกา ห้ามการเขียนฟังก์ชั่นมีผลต่อ Port E ที่เขียน strobe และ /IOWRสัญญาณ

การควบคุมบิตจะมีผลกระทบกับช่องว่างI/Oภายใน ซึ่งไม่มีรีจิสเอตร์ที่เกี่ยวข้องกับการอ่านหรือเขียน การเข้าถึงI/O ภายในหรือเขียนจะมีระยะเวลาเท่ากับ 2 สัญญาณนาฬิกา

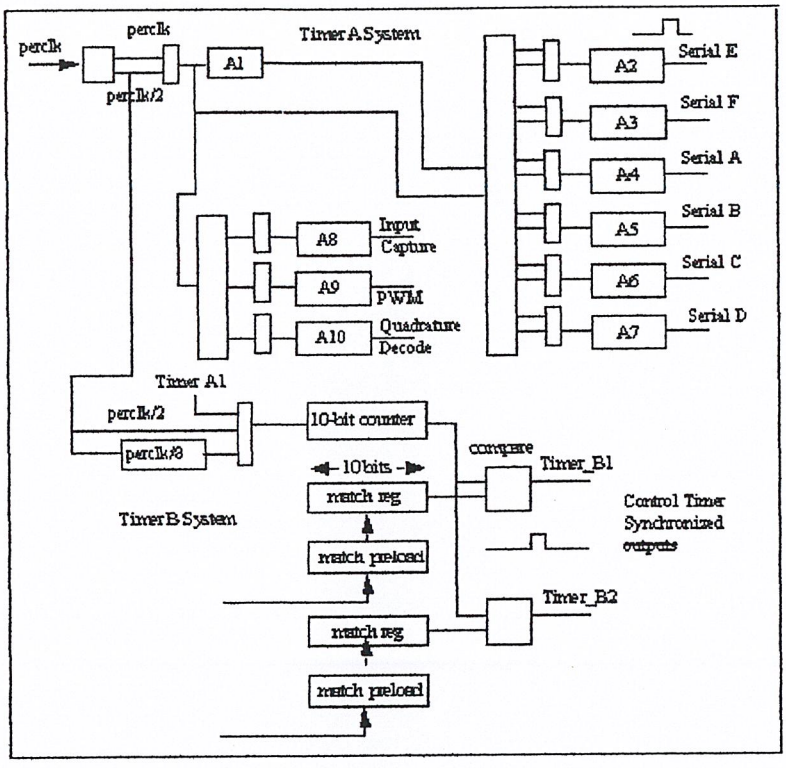
I/O strobe ส่วนใหญ่ทำการเชื่อมต่อกับอุปกรณ์ภายนอกได้ง่าย มีการคืนค่า5 บิตบนในแต่ละรีจิสเตอร์ ถูกเคลียร์ไป PPE (Parallel Port E) จะไม่เป็นเอาท์พุตส่งสัญญาณ ถ้ามิใช่ บิตของรีจิสเตอร์ data-direction ถูกกำหนดสำหรับออกแบบตำแหน่งเอาพุตท์ นอกจากนี้รีจีสเตอร์ฟังก์ชั่นพอร์ท E ต้องตั้งค่าที่"1" สำหรับแต่ละตำแหน่ง

แต่ละรีจิสเตอร์I/O bankได้ถูกเลือกโดย 3บิทสำคัญมากที่สุดอยู่ I/O ขนาด 16 บิต ตาราง 2-40 การแสดงความสัมพันธ์ระหว่างรีจิสเตอร์ควบคุมI/O และที่ว่างที่คล้ายคลึงกัน ซึ่งมีขนาด 64 K

ตารางที่ 2-40 External I/O Register Address Range and Pin Mapping			
Control Register	**Port E Pin**	**I/O Address A[15:13]**	**I/O Address Range**
IB0CR	PE0	000	0x0000-0x1FFF
IB1CR	PE1	001	0x2000-0x3FFF
IB2CR	PE2	010	0x4000-0x5FFF
IB3CR	PE3	011	0x6000-0x7FFF
IB4CR	PE4	100	0x8000-0x9FFF
IB5CR	PE5	101	0xA000-0xBFFF
IB6CR	PE6	110	0xC000-0xDFFF
IB7CR	PE7	111	0xE000-0xFFFF

2.1.10 Timers

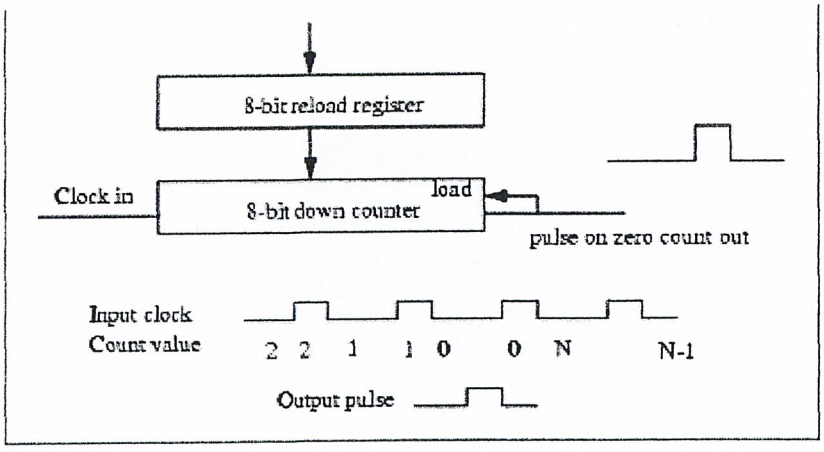
จะมีอยู่ 2 ตัว คือ Timers A และ Timers B โดยที่ Timers A ส่วนใหญ่จะใช้สำหรับการสร้างสัญญาณนาฬิกาสำหรับอุปกรณ์ต่างๆ อัตราการส่งสัญญาณ สำหรับพอร์ทอนุกรม สัญญาณนาฬิกาแบบคาบสำหรับสัญญาณ ของพอร์ทขนาน D และ E หรือสำหรับการสร้างคาบของอินเตอร์รัพ Timers A1-A7 ซึ่งเป็นจุดประสงค์ทั่วไปของTimers และ Timers A8-A10 ใช้ไปสำหรับอุปกรณ์รอบข้างที่เฉพาะเจาะจงลงไป Timer B ใช้สำหรับฟังก์ชั่นเดียวกัน แต่ไม่สามารถสร้างอัตราความเร็วของสัญญาณได้ Timer B มีความบอบบาง ในขณะที่ มันใช้งานอยู่ เพราะว่าโปรแกรมสามารถอ่าน Timer จากตัวนับที่ทำงานอยู่ตลอดเวลา และเหตุการณ์สามารถถูกโปรแกรมได้เผื่ออนาคต



รูป2-33 Block Diagram of Timers A and B

Timer A ประกอบไปด้วย countdown Timers A1-A10 แสดงในรูปที่ 11-1

Timer A1 และ A2-A10 มีขนาด 8 บิต รีจิสเตอร์นับถอยหลัง(Countdown รีจิสเตอร์) แสดงในรูปที่ 11-2 มีรีโหลดรีจิสเตอร์(Reload รีจิสเตอร์) ซึ่งบรรจุเลขในระยะ 0-255 ตัวนับจะถูกหารโดย (N+1) ตัวอย่าง ถ้า รีโหลด รีจิสเตอร์ มี 127 แล้ว 128 ฟัลส์(Pulse)เข้ามาทางด้านซ้ายแล้วออกไปด้านขวา ถ้ารีโหลดรีจิสเตอร์มีค่าเท่ากับ 0 แล้วแต่ละฟัลส์ออกมาจากด้านซ้ายผลลัพธ์ในฟัลส์ที่ด้านขวา มีการหารด้วย 1



รูปที่ 2-34 . Reload รีจิสเตอร์ Operation

ระบบการทำงานของ Timers สามารถถูกควบคุมโดย สัญญาณนาฬิกาของอุปกรณ์เสริม หรือ สัญญาณอุปกรณ์เสริมหาร2 สัญญาณนาฬกาโดยปกติจะเหมือนกับสัญญาณนาฬิกาของสัญญาณนาฬิกาของหน่วยประมวลโดยหารด้วย8 เอาพุตท์ฟัลส์จะเป็น 1 สัญญาณนาฬิกา สัญญาณนาฬิกาของตัวนับ(Counter) สามารถแทนที่บนขอบขาลงของฟัลส์ ในขณะที่ตัวนับจะมีค่าเท่ากับ0 รีโหลดรีจิสเตอร์จะถูกโหลดฟัลส์อินพุตท์ต่อไปเพื่อแทนที่ของตัวนับที่จะดำเนินต่อไป รีโหลดรีจิสเตอร์อาจจะถูกโหลดที่ระยะเวลาหนึ่งตั้งแต่สัญญาณนาฬิกาของอุปกรณ์เสริมถูกโหลดใหม่ synchronous กับสัญญาณนาฬิกาของตัวประมวลผล.

ไทเมอร์ A2 ,A3,A4,A5,A6, และA7 จะเตรียมความเร็วในการส่งสัญญาณนาฬิกาสำหรับพอร์ทอนุกรม พอร์ท E ,F.A.B.C และ D ยกเว้นสำหรับถ้ามีค่าความเร็วต่ำ , สัญญาณนาฬิกา A1 ไม่ต้องการที่ถูกใช้เป็นที่วัดอินพุตท์ของสัญญาณนาฬิกาสำหรัย A2-A7 เช่น ถ้าสัญญาณนาฬิกาของระบบเท่ากับ 11.0592 MHz และ A4 หารด้วย 144 อะซิงโครนัสความเร็วในการส่งของ 2400bps (โดยสมมุติ Timerถูกสัญญาณนาฬิกาโดยนาฬิกาอุปกรณ์เสริมที่หารโดย 2) สัญญาณนาฬิกาอินพุตท์เพื่อพอร์ทอนุกรมมีค่า 8 หรือ 16 ครั้ง อัตราการส่งสำหรับอะซิงโครนัสโหมดและ 8 ครั้ง อัตราความเร็วในการส่งสำหรับซิงโครนัสโหมด ค่ามากที่สุดของซิงโครนัส อัตราความเร็วในการส่งเท่ากับ 11.0592 MHz สัญญาณนาฬิกาเท่ากับ 11,059,200/(1/8) =1,382,400

สำหรับ 7 ของตัวนับ(A1-A7) เงื่อนไขการนับสถานีปลายทางถูกรายงานในรีจิสเตอร์สถานะและถูกโปรแกรมในการสร้างอินเตอร์รัพ มีหนึ่งเป็นอินเตอร์รัพเวกเตอร์ สำหรับ Times A และลำดับก่อนหลังอินเตอร์รัพทั่วไป รีจิสเตอร์สถานะทั่วไป (TACSR) มีบิตสำหรับแต่ละไทเมอร์ จะแสดงถ้าเอาท์พุตฟัลส์ สำหรับไทเมอร์มีนำออกมาตั้งแต่ การอ่านครั้งสุดท้ายของรีจิสเตอร์สถานะ ในขณะที่รีจิสเตอร์สถานะทำการอ่าน บิตเหล่านั้นจะถูกลบ ไม่มีบิตที่เสียไป สิ่งใดสิ่งหนึ่งจะถูกอ่านโดยรีจิสเตอร์สถานะอ่านหรือ ถูกติดตั้งหลังจากที่ รีจิสเตอร์สถานะอ่านเสร็จสมบูรณ์ ถ้าบิตบนและอินเตอร์รัพที่คล้ายกันทำงาน แต่อย่างไรก็ตามอินเตอร์รัพที่กระจายออกมาไม่ถูกต้อง สำหรับแต่ละบิตกับอินเตอร์รัพที่เกิดขึ้น ถ้าบิตอ่านในสถานะรีจิสเตอร์ มันจะลบและไม่มีอินเตอร์รัพเพื่อบิตมาร้องขอ มันเป็นไปได้ที่บิตจะเป็นสาหตุที่ทำให้เกิดอินเตอร์รัพ บิตที่เพิ่มเติมหนึ่งหรือมากกว่าจะถูกตั้งก่อนรีจิสเตอร์สถานะถูกอ่าน. หลังบิทเหล่านี้ถูกลบ พวกมันไม่สามารถเป็นสาเหตุให้อินเตอร์รัพ บิทถ้ามีบนและอินเตอร์รัพที่คล้ายคลึงถูกทำให้ แล้วใช้ได้อินเตอร์รัพจะใช้สถานที่ทันทีที่ลำดับก่อนหลังยอม อย่างไรก็ตามถ้าบิทบนและอินเตอร์รัพที่คล้ายคลึงถูกรีเซ็ท

ถึงแม้ว่าไทเมอร์ A8-A10 คือส่วนหนึ่ง ของไทเมอร์ A, แต่มักจะถูกทำงานด้วยอินพุตท์ ฟัลส์, PWM, และตัวถอดรหัสควอแรนด์ บริเวณที่อยู่รอบนอกทำสัญญาณนาฬิกา โดยที่ไทเมอร์สามารถสร้างอินเตอร์รัพ แต่ไทเมอร์ไม่สามารถสร้าง ยิ่งกว่านั้นไทเมอร์ ไม่สามารถทำการแคสเคสกับ A1

2.1.10.1 Timer A I/O รีจิสเตอร์

รีจิสเตอร์ของ ไทเมอร์ A แสดงในตารางที่ 2-41

ตารางที่ 2-41Timer A I/O รีจิสเตอร์

รีจิสเตอร์ Name	Mnemonic	I/O address	R/W	Reset
Timer A Control/Status รีจิสเตอร์	TACSR	0xA0	R/W	00000000
Timer A Prescale รีจิสเตอร์	TAPR	0xA1	W	xxxxxxx1
Timer A Time Constant 1 รีจิสเตอร์	TAT1R	0xA3	W	xxxxxxxx
Timer A Control รีจิสเตอร์	TACR	0xA4	W	00000000
Timer A Time Constant 2 รีจิสเตอร์	TAT2R	0xA5	W	xxxxxxxx
Timer A Time Constant 8 รีจิสเตอร์	TAT8R	0xA6	W	xxxxxxxx
Timer A Time Constant 3 รีจิสเตอร์	TAT3R	0xA7	W	xxxxxxxx
Timer A Time Constant 9 รีจิสเตอร์	TAT9R	0xA8	W	xxxxxxxx
Timer A Time Constant 4 รีจิสเตอร์	TAT4R	0xA9	W	xxxxxxxx
Timer A Time Constant 10 รีจิสเตอร์	TAT10R	0xAA	W	xxxxxxxx
Timer A Time Constant 5 รีจิสเตอร์	TAT5R	0xAB	W	xxxxxxxx
Timer A Time Constant 6 รีจิสเตอร์	TAT6R	0xAD	W	xxxxxxxx
Timer A Time Constant 7 รีจิสเตอร์	TAT7R	0xAF	W	xxxxxxxx

สรุปความสามารถของไทเมอร์A

ตารางที่ 2-42 ความสามารถของTimer A

Timer	Cascade	Interrupt	Dedicated connection
A1	none	yes	Parallel Ports D-G, Timer B
A2	from A1	yes	Serial Port E
A3	from A1	yes	Serial Port F
A4	from A1	yes	Serial Port A
A5	from A1	yes	Serial Port B
A6	from A1	yes	Serial Port C
A7	from A1	yes	Serial Port D
A8	none	no	ดักจับอินพุตท์
A9	none	no	Pulse Width Modulator
A10	none	no	Quadrature Decoder

รีจิสเตอร์ควบคุม/สถานะของไทเมอร์ A(The control/status รีจิสเตอร์ สำหรับ ไทเมอร์ A (TACSR))แสดงในตารางที่ 2-43

ตารางที่ 2-43 Timer A Control and Status รีจิสเตอร์ (adr = 0A0h)

	Bit 7	**Bit 6**	**Bit 5**	**Bit 4**	**Bit 3**	**Bit 2**	**Bit 1**	**Bit 0**
Read	A7 นับค่า	A6 นับค่า	A5 นับค่า	A4 นับค่า	A3 นับค่า	A2 นับค่า	A1 นับค่า	เขียนเท่านั้น
Write	A7	A6	A5	A4	A3	A2	A1	1--

	อินเตอร์รัพทำงาน	อินเตอร์รัพทำงาน	อินเตอร์รัพทำงาน	อินเตอร์รัพทำงาน	อินเตอร์รัพทำงาน	อินเตอร์รัพทำงาน	อินเตอร์รัพทำงาน	Timer A ทำงาน (pclk/2)

บิตที่ 1-7 อ่าน/เขียน ตัวนับปลายทางจะถึง Timers A1-A7 การอ่านที่รีจิสเตอร์สถานะจะทำการลบบิตเหล่านี้ (บิต1-7) การเขียนถึงบิตเหล่านั้นเป็นอินเตอร์รัพสำหรับ Timers ที่คล้ายกัน

บิต 0 เขียน ตั้งค่า"1" เพื่อทำให้สัญญาณนาฬิกา (perclk/2) ใช้ได้ สำหรับ Timer A ตั้งค่า "0" เพื่อยกเลิกสัญญาณนาฬิกา (perclk/2) บิตที่ 1-7 ถูกเขียน (เขียนเท่านั้น) เพื่อทำให้อินเตอร์รัพ ใช้ได้เป็นเวลาคล้ายคลึงกัน

รีจิสเตอร์ ควบคุม (TACR) แผนงานแสดงในตารางที่ 2-44

ตารางที่ 2-44 รีจิสเอตร์ควบคุม Timer A (adr = 0A4h)

Bit 7 A7	Bit 6 A6	Bit 5 A5	Bit 4 A4	Bit 3 A3	Bit 2 A2	Bits 1, 0
Source A7 0-pclk/2 1-A1	Source A6 0-pclk/2 1-A1	Source A5 0-pclk/2 1-A1	Source A4 0-pclk/2 1-A1	Source A3 0-pclk/2 1-A1	Source A2 0-pclk/2 1-A1	00—อินเตอร์รัพ ไม่ทำงาน 01—เป็นสัญญาณนาฬิการอบข้าง 1 10--เป็นสัญญาณนาฬิการอบข้าง 2 11--เป็นสัญญาณนาฬิการอบข้าง 3

The Timer A Prescale รีจิสเตอร์ (TAPR) เป็นสัญญาณนาฬิกาหลักสำหรับTimer A โดยที่จะถูกทำสัญญาณนาฬิกาโดยสัญญาณนาฬิการอบข้างหารด้วย2

The prescale รีจิสเตอร์ (TAPR) จะแสดงในตารางที่ 11-5

ตารางที่ 2-45 รีจิสเตอร์Prescale Timer A (adr = 0A1h)

Bits 7:1	Bit 0

These bits are ignored.	0-The main clock สำหรับ Timer Aเป็นสัญญาณนาฬิการอบข้าง 1-The main clock สำหรับTimer Aเป็นสัญญาณนาฬิการอบข้าง/2

TATXR มีขนาด 8 บิตซึ่งจะมีค่าระหว่าง 0-255 โดยค่าคงที่ของเวลาจะมีผลต่อช่วงเวลาถุดไป เช่น Timer A นับถอยหลังไปถึง 0 Timer นับมอลดูเลชั่น (n+1) โดยที่ค่า nที่กำนนดเป็นค่า คงที่จองเวลา ค่าคงที่ของเวลารีจิสเตอร์ เขียนเท่านั้น ค่าคงที่ของเวลารีจิสเตอร์ดูได้ที่ ตาราง11-1

2.1.10.2 Timer B

แสดงตารางของ Timer B สามารถไดร์วโดยตรงโดย perclk/2 หารด้วย8 หรือเอาพุตท์ของไทเมอร์ A1 ไทเมอร์ B มีการทำงาน 10 บิตลง ตัวนับจะเปรียบเทียบระหว่าง 2 รีจิสเตอร์ B1 Match รีจิสเตอร์และ B2 รีจิสเตอร์ ในขณะเปลี่ยนแปลงตัวนับเพื่อค่าเท่ากับรีจิสเตอร์ ฟัลส์ภายในกับความยาว 1สัญญาณนาฬิกาอุปกรณ์เสริมเกิดขึ้น ฟัลส์สามารถถูกใช้เพื่อสาเหตุอินเตอร์รัพและ/หรือสัญญาณนาฬิกา เอาพุทต์รีจิสเตอร์ของพอร์ทขนาน Dและ E

Match รีจิสเตอร์ถูกโหลดจาก match preload รีจิสเตอร์ถูกเขียนโดย คำสั่ง I/O ดาต้าไบต์ใน match preload รีจิสเตอร์ ในขณะที่ฟัลส์มีการรีจิสเตอร์ในการสร้าง

ถ้า match รีจิสเตอร์ต้องการเปลี่ยนความต้องการไบต์สำคัญมากที่สุดเพื่อถูกเปลี่ยนแรก รีจิสเตอร์ I/O เป็น ไทเมอร์ B แสดงในตาราง

ตารางที่ 2-46 Timer B รีจิสเตอร์

รีจิสเตอร์ Name	Mnemonic	I/O address	R/W	Reset
Timer B Control/Status รีจิสเตอร์	TBCSR	0xB0	R/W	xxxxx000
Timer B Control รีจิสเตอร์	TBCR	0xB1	W	xxxx0000
Timer B MSB 1 รีจิสเตอร์	TBM1R	0xB2	W	xxxxxxxx
Timer B LSB 1 รีจิสเตอร์	TBL1R	0xB3	W	xxxxxxxx
Timer B MSB 2 รีจิสเตอร์	TBM2R	0xB4	W	xxxxxxxx
Timer B LSB 2 รีจิสเตอร์	TBL2R	0xB5	W	xxxxxxxx

Timer B Count MSB รีจิสเตอร์	TBCMR	0xBE	R	xxxxxxxx
Timer B Count LSB รีจิสเตอร์	TBCLR	0xBF	R	xxxxxxxx

The control/status รีจิสเตอร์ Timer B (TBCSR) is laid out as shown in Table 11-7.

ตารางที่ 2-47 Timer B Control and Status รีจิสเตอร์ (TBCSR) (adr = 0B0h)			
Bits 7:3	**Bit 2**	**Bit 1**	**Bit 0**
Not used	1 แมชต์กับแมชต์รีจิสเตอร์ 2 ถูกตรวจสอบที่บิตนั้น บิตจะถูกลบเมื่อรีจิสเตอร์อ่าน ตั้งค่าเป็น 1ในขณะที่อินเตอร์รัพทำงาน	1-1 แมชต์กับแมชต์รีจิสเตอร์ ถูกตรวจสอบที่บิตนั้น บิตจะถูกลบเมื่อรีจิสเตอร์อ่าน ตั้งค่าเป็น 1ในขณะที่อินเตอร์รัพทำงาน	1—มีการทำงานสัญญาณนาฬิกาหลักสำหรับไทเมอร์

รีจิสเตอร์ควบคุม สำหรับ Timer B (TBCR) is laid out as shown in Table 2-48.

ตารางที่ 2-48 Timer B Control รีจิสเตอร์ (TBCR)		
Bits 7:4	**Bits 3:2**	**Bits 1:0**
Not used	00--Counter clocked โดย perclk/2 01--Counter clocked โดยเอาพุตท์ของ timer A1 1x--Timer clocked โดยperclk/2 หารด้วย 8	00--Interrupt disabled xx--Interrupt priority xx enabled.

The MSB *x* รีจิสเตอร์สำหรับ Timer B (TBM1R/TBM2R) แสดงในตารางที่ 2-49

ตารางที่ 2-49. Timer B Count MSB x รีจิสเตอร์s		
Timer B Count MSB x รีจิสเตอร์ (TBM1R) (Address = 0xB2) (TBM2R) (Address = 0xB4)		
Bit(s)	**Value**	**Description**
7:6	Write	2 MSBs ของค่าคอมแพรสำหรับ the Timer B comparator ถูกเก็บ ค่าที่เตรียมไว้จะถูกโหลดไปยังคแมแพเรเตอร์ในขณะที่กระแสของคอมแพเรเตอร์ตรวจสอบแมตช์
5:0		บิตอ่านจะเป็น 0

The LSB *x* รีจิสเตอร์s สำหรับ Timer B (TBL1R/TBL2R) แสดงในตารางที่ 2-50

ตารางที่ 2-50 Timer B Count LSB x รีจิสเตอร์		
Timer B Count LSB x รีจิสเตอร์ (TBL1R) (Address = 0xB3) (TBL2R) (Address = 0xB5)		
Bit(s)	**Value**	**Description**
7:0	Write	8 LSBs ของค่าที่เปรียบเทียบ สำหรับ the Timer B ถูกเก็บ ค่าทีเปรียบเทียบจะโหลดคก็บไว้ในตัวเปรียบเทียบในขณะที่กระแสตัวเปรียบเทียบได้ตรวจการแมตช์

ตารางที่ 2-51. Timer B Count MSB รีจิสเตอร์		
Timer B Count MSB รีจิสเตอร์ (TBCMR) (Address = 0xBE)		
Bit(s)	**Value**	**Description**
7:6	Read	ค่าของกระแสของ 2 MSBs ของ Timer B counter ได้ถูกราบงาน
5:0		บิตที่อ่านจะมีค่าเป็น 0.

ตารางที่ 2-52 Timer B Count LSB รีจิสเตอร์		
Timer B Count LSB รีจิสเตอร์ (TBCLR) (Address = 0xBF)		
Bit(s)	**Value**	**Description**
7:0	Read	ค่าของกระแสของ 8 LSBs ของ Timer B counter จะถูกรายงาน

2.1.11. Rabbit Serial Ports

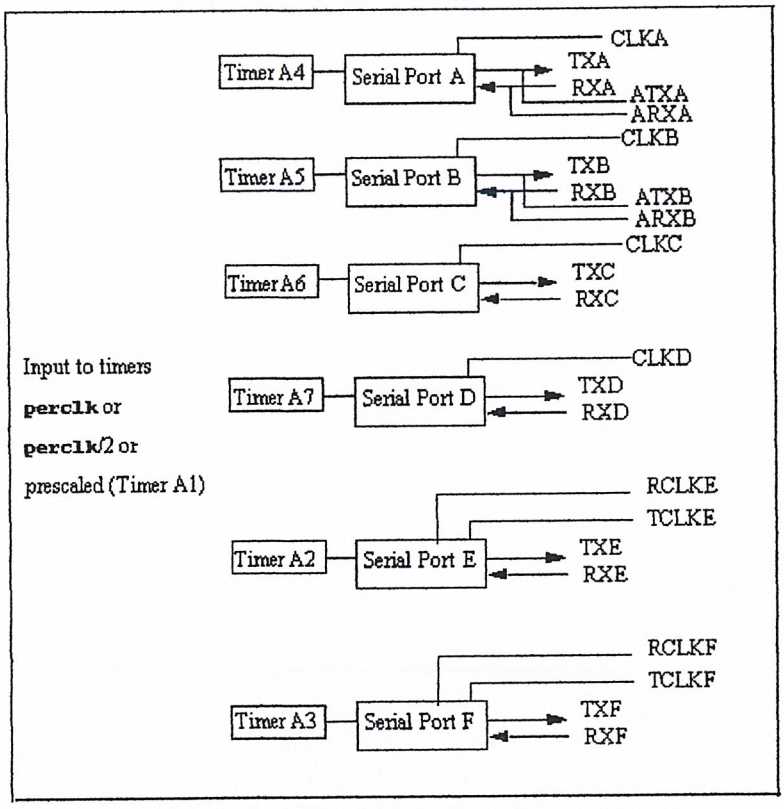
Rabbit3000 มี 6 พอร์ทเป็นชุดชิปมี A, B, C, E, และ F พอร์ททั้งหมดสามารถกระทำอะซิงโครนัส ติดต่อโดยมีอัตรการส่งที่สูง. พอร์ท A และ พอร์ท B สามารถถูกเปลี่ยนไปยังขาต่างๆ สลับกัน. พอร์ท E และ พอร์ท F การติดต่อแบบซิงโครนัส SDLC/HDLC การสนับสนุนอะซิงโคนัสเป็นมาตรฐาน พอร์ท A มีความสามารถพิเศษของการนำไปใช้เพื่อการบูทระยะไกลของไมโครโปรเซสเซอร์ผ่าน อะซิงโครนัส, ซิงโครนัส, หรือ IrDA (อะซิงโครนัส แบบอนุกรม)

ตาราง 2-53 รายละเอียดของสัญญาณอะซิงโครนัสแบบพอร์ทขนาน

ตาราง 2-53 . Serial Port Signals		
Serial Port	**Signal Name**	**Function**
Serial Port A	TXA	พอร์ทอนุกรมส่งออก
	RXA	พอร์ทอนุกรมรับเข้า
	CLKA	สัญญาณในโหมดสัญญานาฬิกา (bidirectional)
	ATXA	เลือกพอร์ทอนุกรมส่งออก
	ARXA	เลือกพอร์ทอนุกรมรับเข้า
Serial Port B	TXB	พอร์ทอนุกรมส่งออก
	RXB	พอร์ทอนุกรมรับเข้า
	CLKB	สัญญาณในโหมดสัญญานาฬิกา (bidirectional)

	ATXB	เลือกพอร์ทอนุกรมส่งออก
	ARXB	เลือกพอร์ทอนุกรมรับเข้า
Serial Port C	TXC	พอร์ทอนุกรมส่งออก
	RXC	พอร์ทอนุกรมรับเข้า
	CLKC	สัญญาณในโหมดสัญญานาฬิกา (bidirectional)
Serial Port D	TXD	พอร์ทอนุกรมส่งออก
	RXD	พอร์ทอนุกรมรับเข้า
	CLKD	สัญญาณในโหมดสัญญานาฬิกา (bidirectional)
Serial Port E	TXE	พอร์ทอนุกรมส่งออก
	RXE	พอร์ทอนุกรมรับเข้า
	TCLKE	ข้อกำหนดในการส่งสัญญาณภายนอก
	RCLKE	ข้อกำหนดในการรับสัญญาณภายนอก
Serial Port F	TXF	พอร์ทอนุกรมส่งออก
	RXF	พอร์ทอนุกรมรับเข้า
	TCLKF	ข้อกำหนดในการส่งสัญญาณภายนอก
	RCLKF	ข้อกำหนดในการรับสัญญาณภายนอก

Figure 2-34 shows a block diagram of the serial ports.



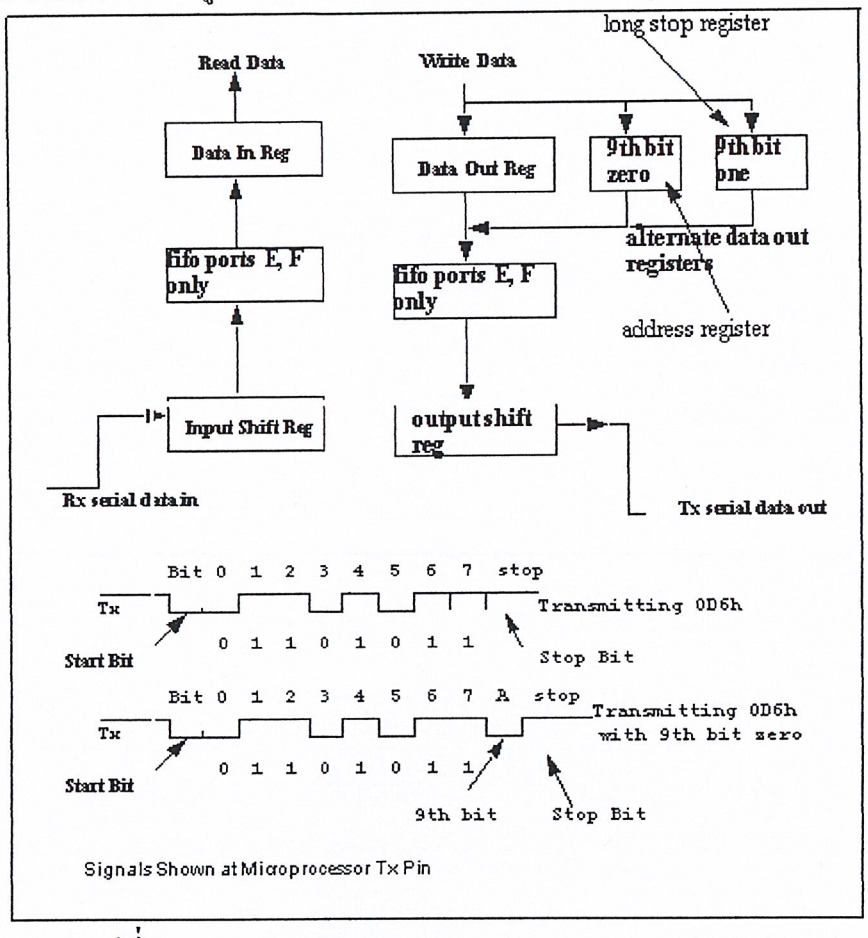
รูปที่ 2-35 Block Diagram of Rabbit Serial Ports

เป็นพอร์ทอนุกรมที่มีความสามารถในการจัดการเรื่องอัตราในการส่ง 500,000 bps ในโหมดอะซิงโคนัส และ ทำได้ 8 ครั้งจะเร็วกว่าในโหมดซิงโครนัส 7หรือ8 บิตข้อมูลจะถูกส่งและรับในโหมดอะซิงโคนัส เราจะเรียกว่า "9 th" บิต หรือ โหมดบิตแอดเดรสซึ่งจะจัดการโดยการสนับสนุน "9 th" จะสามารถปรับค่าสูงหรือต่ำได้โดยการเข้าถึงชุดรีจิสเตอร์แบบอนุกรม จนกระทั่ง Parity และ Multiple stop บิตจะไม่ช่วยเหลือโดยตรงจากอุปกรณ์โดยตรง บิท "ที่ 9" สามารถถูกใช้เพื่อออกบิตพิเศษหยุด(9-บิทสูง) หรือเปลี่ยนเพื่อแสดงพาริตี้

2.1.11.1 Serial Port Register Layout

รูปที่ 12-2 จะแสดงให้เห็นถึงฟังก์ชั่นบล็อกไดอะแกรมของพอร์ทอนุกรม โดยแต่ละพอร์ทอนุกรมจะมีดาต้ารีจิสเตอร์อยู่สถานะรีจิสเตอร์ การเขียนลงไปในดาต้ารีจิสเตอร์จะเริ่มการส่ง Least significant bit (LSB) โดยปกติจะเป็นการเริ่มส่งครั้งแรก เป็นเรื่องจริงสำหรับการติดต่อทั้ง 2 อย่างคือ อะซิงโคนัสและซิงโคนัส ถ้าเขียนถูกแสดงให้เห็นว่าจะทำให้แอดเดรสของดาต้ารีจิสเตอร์มีการสลับ , บิทแอดเดรสพิเศษ หรือ บิทที่ 9 (บิทที่ 8ถ้า 7 บิทข้อมูล) ถูกส่ง. เมื่อบิทข้อมูลได้ถูกรับ

, บิตแอดเดรสถูกอ่านจากดาต้ารีจิสเตอร์ (LSB แรก) รีจิสเตอร์คอนโทรลถูกใช้เพื่อตั้งส่งและรับตัวแปร Status รีจิสเตอร์อาจจะถูกเพื่อตรวจบนการคำนวณของพอร์ทอนุกรม



รูปที่ 2-35. Functional Block Diagram of a Serial Port

อินพุตท์ของสัญญาณที่เข้ามายังพอร์ทอนุกรมแต่ละตัวจะมี 8หรือ6 (selectable)ครั้ง อัตรเร็วในการส่งในโหมดอะซิงโดนัส และ 2 ครั้ง อัตราการส่งสำหรับโหมดclock อนุกรม ในขณะที่สัญญาณนาฬิกาภายในถูกใช้ Timer A2-A7 ให้สัญญาณนาฬิกาสำหรับ Serial Port A-F เวลาสามารถแบ่งความถี่โดยแต่ละเลขมาจาก 1-256 มีอินพุทต์ความถี่เพื่อTimers สามารถถูกเลือกในทางต่างๆ 1ในนั้นคือสัญญาณนาฬิการอบๆ โดยตัวเลือกและเลือกความถี่ผลึกสำหรับศูนย์กลาง ออสซิเรเตอร์, โดยส่วนใหญ่จะใช้กระบวนการส่งข้อมูลซึ่งจะ มากที่สุดใช้ปรกติอัตราสามารถถูกรับลงสู่เป็นจำนวนเกือบ 2400 bps หรือทำให้ตกต่ำโดย prescaling เวลากว่าA0 ที่ความถี่นาฬิกาของ Rabbit สูงที่สุด

2.1.11.2 Serial Port Register

โดยแต่ละพอร์ทจะมี 6 จิสเตอร์ซึ่งจะแสดงในตาราง สถานะ,การควบคุมและ รีจิสเตอร์ภายนอก อาจจะรูปแบบที่ต่างกันสำหรับพอร์ทอนุกรม

ตาราง 2-54 Serial Port A Registers

Register Name	Mnemonic	I/O Address	R/W	Reset
Serial Port A Data Register	SADR	0xC0	R/W	xxxxxxxx
Serial Port A Address Register	SAAR	0xC1	W	xxxxxxxx
Serial Port A Long Stop Register	SALR	0xC2	W	xxxxxxxx
Serial Port A Status Register	SASR	0xC3	R	0xx00000
Serial Port A Control Register	SACR	0xC4	W	xx000000
Serial Port A Extended Register	SAER	0xC5	W	00000000

ตาราง 2-55 Serial Port B Registers

Register Name	Mnemonic	I/O Address	R/W	Reset
Serial Port B Data Register	SBDR	0xD0	R/W	xxxxxxxx
Serial Port B Address Register	SBAR	0xD1	W	xxxxxxxx
Serial Port B Long Stop Register	SBLR	0xD2	W	xxxxxxxx
Serial Port B Status Register	SBSR	0xD3	R	0xx00000
Serial Port B Control Register	SBCR	0xD4	W	xx000000
Serial Port B Extended Register	SBER	0xD5	W	00000000

ตารางที่ 2-56 Serial Port C Registers

Register Name	Mnemonic	I/O Address	R/W	Reset

Serial Port C Data Register	SCDR	0xE0	R/W	xxxxxxxx
Serial Port C Address Register	SCAR	0xE1	W	xxxxxxxx
Serial Port C Long Stop Register	SCLR	0xE2	W	xxxxxxxx
Serial Port C Status Register	SCSR	0xE3	R	0xx00000
Serial Port C Control Register	SCCR	0xE4	W	xx000000
Serial Port C Extended Register	SCER	0xE5	W	00000000

ตารางที่ 2-57 Serial Port D Registers

Register Name	Mnemonic	I/O Address	R/W	Reset
Serial Port D Data Register	SDDR	0xF0	R/W	xxxxxxxx
Serial Port D Address Register	SDAR	0xF1	W	xxxxxxxx
Serial Port D Long Stop Register	SDLR	0xF2	W	xxxxxxxx
Serial Port D Status Register	SDSR	0xF3	R	0xx00000
Serial Port D Control Register	SDCR	0xF4	W	xx000000
Serial Port D Extended Register	SDER	0xF5	W	00000000

ตารางที่ 2-58 Serial Port E Registers

Register Name	Mnemonic	I/O Address	R/W	Reset
Serial Port E Data Register	SEDR	0xC8	R/W	xxxxxxxx
Serial Port E Address Register	SEAR	0xC9	W	xxxxxxxx
Serial Port E Long Stop Register	SELR	0xCA	W	xxxxxxxx

Serial Port E Status Register	SESR	0xCB	R	0xx00000
Serial Port E Control Register	SECR	0xCC	W	xx000000
Serial Port E Extended Register	SEER	0xCD	W	000x000x

ตารางที่ 2-59 Serial Port F Registers

Register Name	Mnemonic	I/O Address	R/W	Reset
Serial Port F Data Register	SFDR	0xD8	R/W	xxxxxxxx
Serial Port F Address Register	SFAR	0xD9	W	xxxxxxxx
Serial Port F Long Stop Register	SFLR	0xDA	W	xxxxxxxx
Serial Port F Status Register	SFSR	0xDB	R	0xx00000
Serial Port F Control Register	SFCR	0xDC	W	xx000000
Serial Port F Extended Register	SFER	0xDD	W	000x000x

ตารางที่ 2-60 Data Register All Ports

Serial Port x Data Register (SADR) (Address = 0xC0)
(SBDR) (Address = 0xD0)
(SCDR) (Address = 0xE0)
(SDDR) (Address = 0xF0)
(SEDR) (Address = 0xC8)
(SFDR) (Address = 0xD8)

Bit(s)	Value	Description
7:0	Read	Returns the contents of the receive buffer.
	Write	Loads the transmit buffer with a data ไบต์ for transmission.

ตารางที่ 2-61 Address Register All Ports		
Serial Port x Address Register (SAAR) (Address = 0xC1) (SBAR) (Address = 0xD1) (SCAR) (Address = 0xE1) (SDAR) (Address = 0xF1) (SEAR) (Address = 0xC9) (SFAR) (Address = 0xD9)		
Bit(s)	**Value**	**Description**
7:0	Read	คืนค่าของบริเวณเก็บข้อมูลชั่วคราว ในโหมดนสัญญาณนาฬิกาแบบอนุกรม การอ่านข้อมูลจากรีจิสเตอร์ทำให้เครื่องรับโดยอัตโนมัติเริ่มต้นที่ 1 ไบท์รับการคำนวณ (สิ่งที่บรรจุปัจจุบันของบริเวณบัฟเฟอร์ถูกอ่านแรก), การเอาออกthe ต้องการสำหรับซอฟแวร์ที่จะออกคำสั่งรับเริ่มต้น.
	Write	โหลดบริเวณบัฟเฟอร์เครื่องส่งกับไบท์ที่อยู่, บิทที่อยู่จะมีค่าเป็น "ศูนย์", สำหรับเครื่องส่งกำลัง. ใน โหมด HDLC, ไบท์สุดท้ายของเฟรมต้องถูกเขียนเพื่อให้รีจิสเตอร์เพื่อทำให้ subsequentใช้ได้ CRC และการปิดค่ากำลังของเครื่องส่ง ในโหมดสัญญาณนาฬิกาแบบอนุกรมเขียนข้อมูลถึงรีจิสเตอร์เป็นสาเหตุให้ผู้ส่งเริ่มต้น ไบท์ส่งการคำนวณ, การลบออกต้องการสำหรับซอฟแวร์ที่จะออกคำสั่งส่งเริ่มต้น.

ตารางที่ 2-62 Long Stop Register All Ports
Serial Port x Long Stop Register (SALR) (Address = 0xC2) (SBLR) (Address = 0xD2) (SCLR) (Address = 0xE2)

(SDLR) (Address = 0xF2) (SELR) (Address = 0xCA) (SFLR) (Address = 0xDA)		
Bit(s)	**Value**	**Description**
7:0	Read	ส่งข้อมูลของบัฟเฟอร์ของเครื่องรับกลับ
	Write	โหลดบริเวณบัฟเฟอร์ของเครื่องส่งกับไบท์ที่อยู่, บิทที่อยู่มีค่าเท่ากับ 1, สำหรับเครื่องส่งกำลัง. ใน โหมด HDLC ไบท์สุดท้ายถูกเขียนเพื่อรีจิสเตอร์เพื่อทำให้ subsequent ใช้ได้ การปิดเครื่องส่งกำลังค่า

ตารางที่ 2-63 Status Register Asynchronous Mode Only (All Ports)		
Serial Port x Status Register (SASR) (Address = 0xC3) (SBSR) (Address = 0xD3) (SCSR) (Address = 0xE3) (SDSR) (Address = 0xF3) (SESR) (Address = 0xCB) (SFSR) (Address = 0xDB)		
Bit(s)	**Value**	**Description (Async mode only)**
7	0	The receive data register ว่างอยู่
	1	มี 1 ไบท์ในบัฟเฟอร์ของเครื่องรับ. เปลี่ยนแปลงจาก "0" เป็น "1"เพื่อตั้งความต้องการอินเตอร์รัพเครื่องรับ flip-flop. FF อินเตอร์รัพลบ เมื่อตัวอักษรถูกอ่านจากบัฟเฟอร์ของข้อมูล. FF อินเตอร์รัพจะถูกตั้งทันทีถ้ามีตัวอักษรมากขึ้นมีให้ในFIFO หรือรีจิสเอตณีตัวถัดไปเพื่อถูกย้ายเข้าไปในบัฟเฟอร์ของข้อมูล.
6	0	ไบท์ในบัฟเฟอร์ของตัวรับคือข้อมูล, รับโดยบิทหยุดสมบูรณ์.

	1	บิทแอดเดรส หรือ บิตที่ 9 (บิตที่ 8) บิทที่รับ. บิทนี้ถูกตั้งถ้าตัวอักษรในรีจิสเตอร์ข้อมูลเครื่องรับมีบิตที่ 9 (บิตที่ 8) บิทนี้ถูกลบและถูกตรวจก่อนอ่านรีจิสเตอร์ข้อมูลตั้งแต่ค่าข้อมูลใหม่กับบิทที่อยู่ใหม่อาจจะถูกโหลดโดยทันทีเมื่อรีจิสเตอร์ข้อมูลถูกอ่าน ไบท์ในบัฟเฟอร์เครื่องรับคือที่อยู่, หรือ ไบท์กับข้อผิดพลาดที่เฟรม. ถ้าบิทแอดเดรสไม่ได้ยอมรับ. ถ้าข้อมูลในบัฟเฟอร์เป็นศูนย์ สิ่งนี้อาจจะหยุด
5	0	บัฟเฟอร์ของเครื่องรับไม่มีควรใช้งาน
	1	บิทนี้ถูกตั้งถ้าเครื่องรับมีการใช้งาน สิ่งที่เกิดขึ้นถ้ารีจิสเตอร์เปลี่ยนแปลงและรีจิสเตอร์ข้อมูลเต็ม และบิทจะเริ่มตวจสอบ บิทนี้จะลบเมื่อรีจิสเตอร์ข้อมูลของเครื่องรับถูกอ่าน
4	0	บิตนี้มีค่าเท่ากับ 0 จะอยู่ในซิงโครนัสโหมด
3	0	ตัวบัฟเฟอร์ในการส่งจะว่าง
	1	บัฟเฟอร์ข้อมูลเครื่องส่งเต็ม. บิทนี้ถูกตั้งเมื่อรีจิสเตอร์ข้อมูลส่งเต็ม, ซึ่งเต็ม, 1ไบท์ถูกเขียนเพื่อรีจิสเตอร์ข้อมูลพอร์ทอนุกรม มันถูกลบ เมื่อไบท์ถูกย้ายไปรีจิสเตอร์ตัวต่อไปของเครื่องส่งหรือ FIFO, หรือ เขียนการคำนวณถูกกระทำถึงบัญชีชื่อสถานะพอร์ทเป็นชุด. บิทนี้จะต้องการอินเตอร์รัพบนการเปลี่ยนแปลงจาก 1 ถึง 0 ถ้าอินเตอร์รัพทำงาน. เครื่องส่งอินเตอร์รัพถูกลบเมื่อบัปเฟอร์เครื่องส่งถูกเขียน, หรือค่าใดๆ (ซึ่งจะไม่สนใจเขียนเพื่อรีจิสเตอร์นี้
2	0	เครื่องส่งไม่ทำงาน
	1	เครื่องส่งกำลังทำงานอยู่ บิตนี้จะถูกตั้งถ้ารีจิสเตอร์ตัวต่อไปของเครื่องส่งกำลังส่งข้อมูล. มันจถูกตั้งค่าในขอบขาลงของบิตเริ่ม ในขณะที่ขอบสัญญาณในการส่งข้อมูลจากรีจิสเตอร์ข้อมูลเครื่องส่งไปยังรีจิสเตอร์ตัวต่อไปของเครื่องส่ง เครื่องส่งกำลังทำงาน บิตจะถูกลบที่สิ้นสุดของ บิตหยุดของตัวอักษรที่ส่ง บิตจะเป็นเหตุให้อินเตอร์รัพถูกล็อก ในขณะที่มันกำลังเปลี่ยนสถานะจากทำงานเป็นไม่ทำงาน หลังจากอักษรตัวสุดท้ายถูกส่งออกไป

1:0	00	บิตมีค่าเท่ากับ 0 จะอยู่ใน โหมด อะซิงโครนัส

ตารางที่ 2-64 Status Register Clocked Serial (Ports A-D only)		
Serial Port x Status Register (SASR) (Address = 0xC3) (SBSR) (Address = 0xD3) (SCSR) (Address = 0xE3) (SDSR) (Address = 0xF3)		
Bit(s)	**Value**	**Description (Clocked serial mode only)**
7	0	ค่ารีจิสเตอร์ข้อมูลเครื่องรับว่าง
	1	มีไบต์ในบัฟเฟอร์เครื่องรับ โดยพอร์อนุกรมได้ขออินเตอร์รัพในขณะที่บิตมีการตั้งค่า อินเตอร์รัพจะถูกลบในขณะที่บัฟเฟอร์เครื่องรับว่าง
6	0	บิตมีค่าเท่ากับ 0 เมื่อสัญญาณนาฬิกาอยู่ในโหมดอนุกรม.
5	0	บัฟเฟอร์เครื่องรับไม่ควรทำงาน
	1	บัฟเฟอร์เครื่องรับทำงาน บิตนี้จะถูกลบโดยอ่านบัฟเฟอร์จากเครื่องรับ
4	0	บิต มีค่าเป็น0 ในโหมดสัญญาณนาฬิกาแบบอนุกรม
3	0	บัฟเฟอร์ของเครื่องส่งจะว่าง.
	1	บัฟเฟอร์เครื่องส่งไม่ว่าง พอร์ทอนุกรมจะขออินเตอร์รัพ ในขณะที่เครื่องส่งเอาไบต์จากบัฟเฟอร์เครื่องส่ง อินเตอร์รัพเครื่องส่งถูกลบในขณะที่ บัฟเฟอร์เครื่องส่งถูกเขียน หรือค่าต่างๆ(ไม่ได้สนใจ)ถูกเขียนจากรีจิสเตอร์
2	0	เครื่องส่งจะไม่ทำงาน
	1	เครื่องส่งจะส่งไบต์ อินเตอร์รัพถูกสร้าง ในขณะที่เครื่องส่งจะลบบิต ซึ่งปรากฏว่า ถ้า เครื่องส่งกำลังส่งข้อมูลบิตอื่นๆ แต่ บัฟเฟอร์เครื่องจะว่าง
1:0	00	บิตจะเป็นค่า 0 ในโหมดสัญญาณนาฬิกาแบบอนุกรม

ตารางที่ 2-65 Status Register HDLC Mode (Ports E and F only)		
Serial Port x Status Register (SESR) (Address = 0xCB) (SFSR) (Address = 0xD3)		
Bit(s)	**Value**	**Description (HDLC mode only)**
7	0	รีจิสเตอร์ข้อมูลเครื่องรับจะว่าง
	1	มีไบต์ในบัฟเฟอร์เครื่องรับ พอร์ทอนุกรมขออินเตอร์รัพในขณะที่บิตตั้งค่า อินเตอร์รัพ จะถูกลบในขณะที่บัฟเฟอร์เครื่องรับว่าง
6,4	00	ไบต์ในบัฟเฟอร์เครื่องรับเป็นข้อมูล
	01	ไบต์ในบัฟเฟอร์เครื่องรับถูกเจอโดยไม่สนใจ
	10	ไบต์ในบัฟเฟอร์เครื่องรับเป็นเฟรมสุดท้าย กับValid CRC
	11	ไบต์ในบัฟเฟอร์เครื่องรับจะเป็นเหรมสุดท้าย กับ ความผิดปกติของCRC
5	0	บัฟเฟอร์เครื่องรับไม่ควรใช้ทำงาน
	1	บัฟเฟอร์เครื่องรับทำงาน บิตจะถูกลบโดยอ่านบัฟเฟอร์เครื่องรับ
3	0	บัฟเฟอร์เครื่องส่งจะว่าง
	1	บัฟเฟอร์เครื่องส่งไม่ว่าง พอร์ทอนุกรมจะขออินเตอร์รัพ ในขณะที่เครื่องส่งเอาไบต์จากบัฟเฟอร์เครื่องส่ง อินเตอร์รัพเครื่องส่งถูกลบในขณะที่ บัฟเฟอร์เครื่องส่งถูกเขียน หรือค่าต่างๆ(ไม่ได้สนใจ)ถูกเขียนจากรีจิสเตอร์
2:1	00	อินเตอร์รัพเครื่องส่งกำหนดให้บัฟเฟอร์สถานะว่าง
	01	เครื่องส่งหยุดส่งCRC อินเตอร์รัพจะถูกสร้างจนจบการส่งCRC ข้อมูลตอบอินเตอร์รัพเป็นสาเหตุมีค่าถูกส่งไปยังเฟรม และไม่มีอินเตอร์รัพในหารสร้างจาก ค่านั้น

	10	เครื่องส่งจะหยุดการส่ง Abort อินเตอร์รัพไม่ถูกสร้างที่หยุดการส่ง Abort
	11	เครื่องส่งหยุดการส่ง closing Flag ข้อมูลเขียนเป็นคำตอบของอินเตอร์รัพ เพราะอย่างน้อย2 ค่าที่ถูกส่งระหว่างเฟรม
0	0	ไบต์ในบัฟเฟอร์เครื่องรับมี 8 บิต
	1	ไบต์ในบัฟเฟอร์เครื่องรับมีขนาดน้อยกว่า 8 บิต

ตารางที่ 2-66 Serial Port Control Register Po0rts A and B		
Serial Port x Control Register (SACR) (Address = 0xC4) (SBCR) (Address = 0xD4)		
Bit(s)	**Value**	**Description**
7:6	00	ไม่มีการทำงาน ไม่บิตในอะซิงโคนัสโหมด
	01	ในสัญญาณนาฬิกาแบบอนุกรม เริ่มวิธีการรับไบต์
	10	ในสัญญาณนาฬิกาแบบอนุกรม เริ่มวิธีการส่งไบต์
	11	ในสัญญาณนาฬิกาแบบอนุกรม เริ่มวิธีการส่งไบต์และเริ่มวิธีการรับไบต์อ พร้อมกัน
5:4	00	พอร์ทขนาน C เป็นอินพุตท์
	01	พอร์ทขนาน D เป็นเอาพุตท์
	1x	ยกเลิกการรับอินพุตท์
3:2	00	โหมดอะซิงโครนัส กับ 8 บิตต่อตัวอักษร
	01	โหมดอะซิงโครนัส กับ 7บิตต่อตัวอักษร ในโหมดนี้ส่วนใหญ่บิตที่สำคัญ จะไม่สนใจในการส่งและข้อมูลที่ได้รับมักจะเป็น0
	10	สัญญาณนาฬิกาแบบอนุกรมกับสัญญาณนาฬิกาภายนอก พอร์ทอนุกราม A เป็นบนพอร์ทขนาน B1

		พอร์ทอนุกราม B เป็นบนพอร์ทขนาน B10
	11	สัญญาณนาฬิกาแบบอนุกรมกับสัญญาณนาฬิกาภายใน Serial Port A clock is on Parallel Port PB1 Serial Port B clock is on Parallel Port PB0
1:0	00	อินเตอร์รัพของพอร์ทอนุกรมไม่ทำงาน
	01	พอร์ทอนุกรมใช้อินเตอร์รัพลำดับก่อนหลัง 1.
	10	พอร์ทอนุกรมใช้อินเตอร์รัพลำดับก่อนหลัง 2

ตารางที่ 2-67 Serial Port Control Register Ports C and D

Serial Port x Control Register (SCCR) (Address = 0xE4)
(SDCR) (Address = 0xF4)

Bit(s)	Value	Description
7:6	00	ไม่มีการทำงาน ไม่มีบิตที่สนใจในโหมด อะซิงโคนัส
	01	ในสัญญาณนาฬิกาแบบอนุกรมเริ่มการรับไบต์
	10	ในสัญญาณนาฬิกาแบบอนุกรมเริ่มการส่งไบต์
	11	ในสัญญาณนาฬิกาแบบอนุกรมเริ่มการส่งและรับไบต์พร้อมๆกัน
5	0	เปิดการทำงานเครื่องรับอินพุทต์
	1	ปิดการทำงานเครื่องรับอินพุตท์
4	x	บิตนี้ไม่สนใจ
3:2	00	8 บิตต่อตัวอักษร
	01	7บิตต่อตัวอักษร ในโหมดนี้ส่วนใหญ่บิตที่สำคัญจะไม่สนใจในการส่งและข้อมูลที่ได้รับมักจะเป็น0

	10	สัญญาณนาฬิกาแบบอนุกรมกับสัญญาณนาฬิกาภายนอก พอร์ทอนุกรม C สัญญาณนาฬิกาบน Parallel Port PF1 พอร์ทอนุกรม C สัญญาณนาฬิกาบน Parallel Port PF0
	11	สัญญาณนาฬิกาแบบอนุกรมกับสัญญาณนาฬิกาภายใน พอร์ทอนุกรม C สัญญาณนาฬิกาบน Parallel Port PF1 พอร์ทอนุกรม C สัญญาณนาฬิกาบน Parallel Port PF0
1:0	00	อินเตอร์รัพของพอร์ทอนุกรมไม่ทำงาน
	01	พอร์ทอนุกรมใช้อินเตอร์รัพลำดับก่อนหลัง 1
	10	พอร์ทอนุกรมใช้อินเตอร์รัพลำดับก่อนหลัง 2
	11	พอร์ทอนุกรมใช้อินเตอร์รัพลำดับก่อนหลัง 3

ตารางที่ 2-68 Serial Port Control Register Ports E and F

Serial Port x Control Register (SECR) (Address = 0xCC) (SFCR) (Address = 0xDC)		
Bit(s)	**Value**	**Description**
7:6	00	ไม่มีการทำงาน ไม่มีบิตที่สนใจในโหมด อะซิงโครนัส
	01	ในโหมด HDLC, เครื่องรับกำลังในโหมดค้นหาค่า.
	10	ไม่มีการทำงาน
	11	ในโหมดHDLC , ส่งแบบ Abort pattern.
5	0	เครื่องรับอินพุทต์ทำงาน
	1	เครื่องรับอินพุทต์ไม่ทำงาน
4	x	ไม่สนใจบิต
3:2	00	โหมดอะซิงโครนัส 8 บิตต่อตัวอักษร

	01	โหมดอะซิงโครนัส 7 บิตต่อตัวอักษร ในโหมดนี้ส่วนใหญ่บิตที่สำคัญจะไม่สนใจในการส่งและข้อมูลที่ได้รับมักจะเป็น0
	10	โหมด HDLC กับสัญญาณนาฬิกาภายนอก สัญญาณนาฬิกาภายนอกถูกต้องการ ดูได้ดังนี้ Transmit clock (Serial Port F)--pins PG0 and PG1on Parallel Port G. Receive clock (Serial Port E)--pins PG4 and PG5 on Parallel Port G.
	11	โหมด HDLC กับสัญญาณนาฬิกาภายใน สัญญาณนาฬิกาภายในเท่ากับ 16* อัตราข้อมูล และ DPLL ใช้ในการย้อนกลับของเครื่องรับสัญญาณนาฬิกา ถ้ามีความจำเป็น สัญญาณนาฬิกาถูกต้องการดังนี้ Transmit clock (Serial Port F)--pins PG0 and PG1on Parallel Port G. Receive clock (Serial Port E)--pins PG4 and PG5 on Parallel Port G.
1:0	00	อินเตอร์รัพพอร์ทอนุกรมไม่ทำงาน
	01	พอร์ทอนุกรมใช้อินเตอร์รัพลำดับก่อนหลัง 1
	10	พอร์ทอนุกรมใช้อินเตอร์รัพลำดับก่อนหลัง 2
	11	พอร์ทอนุกรมใช้อินเตอร์รัพลำดับก่อนหลัง 3

ตารางที่ 2-69 Extended Register Asynchronous Mode All Ports
Serial Port x Extended Register (SAER) (Address = 0xC5) **(SBER) (Address = 0xD5)** **(SCER) (Address = 0xE5)** **(SDER) (Address = 0xF5)** **(SEER) (Address = 0xCD)**

(SFER) (Address = 0xDD)		
Bit(s)	**Value**	**Description (Async mode only)**
7:5	xxx	ไม่มีบิตที่สนใจในโหมด อะซิงโคนัส
4	0	ปกติ มีการถอรหัสข้อมูล อะซิงโคนัส
	1	ทำงานถอดรหัส RZI (3/16ths bit cell IrDA-compliant).
3	0	การหยุดแบบปกติ การเลือกควรถูกเลือกในขณะที่ บิตที่อยู่ ได้รับการยกเว้น
	1	การหยุดแบบรวดเร็ว การจบของหยุดตัวอักษรที่เลียนแบบถูกเขียนโดยบัฟเฟอร์และเครื่องรับสามารถเริ่มรวมตัวอักษร
2	0	สัญญาณนาฬิกาแบบอะซิงโคนัส 16X อัตราข้อมูล
	1	สัญญาณนาฬิกาแบบอะซิงโคนัส 8X อัตราข้อมูล
1:0	xx	ไม่สนใจบิตในอะซิงโคนัสโหมด

ตารางที่ 2-70 Extended Register Clocked Serial Mode (Ports A-D only)

Serial Port x Extended Register (SAER) (Address = 0xC5) (SBER) (Address = 0xD5) (SCER) (Address = 0xE5) (SDER) (Address = 0xF5)		
Bit(s)	**Value**	**Description (Clocked serial mode only)**
7	0	การทำงานสัญญาณนาฬิกาอนุกรมแบบปกติ
	1	การทำงานสัญญาณนาฬิกาอนุกรมแบบTimer synchronized
6	0	สัญญาณนาฬิกาอนุกรมแบบ Timer synchronize ใช้ Timer B1.
	1	สัญญาณนาฬิกาอนุกรมแบบ Timer synchronize ใช้ Timer B2.

5:4	00	Normal clocked serial clock polarity, inactive High. ภายในหรือภายนอกสัญญาณนาฬิกา
	01	Normal clocked serial clock polarity, inactive Low. Internal clock only.
	10	Inverted clocked serial clock polarity, inactive Low. Internal or external clock.
	11	Inverted clocked serial clock polarity, inactive High. Internal clock only.
3:2	xx	ไม่สนใจบิตในโหมดสัญญาณนาฬิกาแบบอนุกรม
1	0	ไม่มีผลต่อการส่ง
	1	ยกเลิกเครื่องส่งกระแสสัญญาณนาฬิกาอนุกรม. ไม่มีผลต่อบัฟเฟอร์
0	0	ไม่มีผลต่อการรับ
	1	ยกเลิกเครื่องรับกระแสสัญญาณนาฬิกาอนุกรม.

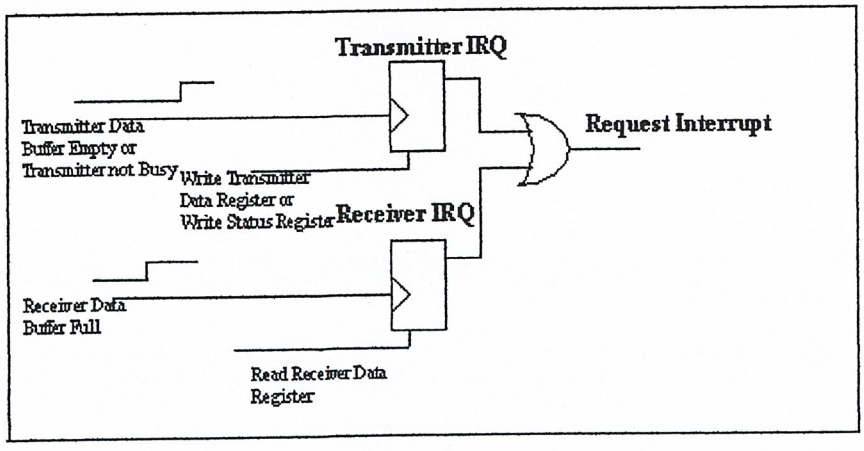
ตารางที่ 2-71 Extended Register HDLC Mode (Ports E and F only)

Serial Port x Extended Register (SEER) (Address = 0xCD) (SFER) (Address = 0xDD)		
Bit(s)	**Value**	**Description (HDLC mode only)**
7:5	000	ถอดรหัสข้อมูล NRZ สำหรับ HDLC เครื่องรับและเครื่องส่ง
	010	ถอดรหัสข้อมูล NRZIสำหรับ HDLC เครื่องรับและเครื่องส่ง
	100	ถอดรหัสข้อมูล Biphase-Level (Manchester) สำหรับ HDLC เครื่องรับและเครื่องส่ง
	110	ถอดรหัสข้อมูล Biphase-Space สำหรับ HDLC เครื่องรับและเครื่องส่ง
	111	ถอดรหัสข้อมูล Biphase-Mark สำหรับ HDLC เครื่องรับและเครื่องส่ง
4	0	ถอดรหัสข้อมูลHDLC

	1	ทำการเข้ารหัส RZI (1/4th bit cell IRDA-compliant). ใช้กับโหมดสัญญาณนาฬิกาภายใน และถอดรหัส ข้อมูล NRZ
3	0	เงื่อนไขไม่ทำงานเป็นค่าต่างๆ
	1	เงื่อนไขไม่ทำงานเป็นเหมือนกัน
2	0	ส่งค่าที่ต่ำกว่าปกติ
	1	ส่งค่าที่มากกว่ากว่าปกติ
1:0	xx	ไม่สนใจในบิตในโหมด HDLC

2.1.11.3 Serial Port Interrupt

โดยทั่วไปอินเตอร์รัพเวกเตอร์จะถูกใช้สำหรับการรับและการส่งอินเตอร์รัพ มีการต้องการให้อินเตอร์รัพแยกออกมา flip-flopสำหรับเครื่องรับและเครื่องส่ง ถ้าflip-flopเหล่านี้ถูกตั้งขึ้น อินเตอร์รัพถูกต้องการflip-flopถูกตั้งโดยขอบขาขึ้นเท่านั้น flip-flop ถูกลบโดย pulse ที่ถูกสร้างขึ้นโดย I/O อ่านหรือเขียนซึ่งจะแสดงที่รูป 12-3 ในขณะที่อินเตอร์รัพได้ถูกการร้องขอ อินเตอร์รัพจะย้ายทันทีเมื่อมีลำดับก่อนหลังและชุดคำสั่งประมวลผลสำเร็จ อินเตอร์รัพจะหายไป ถ้ามีความต้องการของ flip-flop ได้ถูกลบไปก่อนที่อินเตอร์รัพเข้ามาแทนที่ ถ้า flip-flop ไม่ถูกลบไปในอินเตอร์รัพ อินเตอร์รัพอื่นๆจะเข้ามาเมื่อมีการเรียงลำดับที่ต่ำลงมา



รูปที่ 2-36. Generation of Serial Port Interrupts

ความต้องการอินเตอร์รัพรับ flip-flop ถูกตั้งหลังจากบิตหยุดมีการทดลอง nominally 1/2 ของทาง ผ่านบิตหยุด บิตข้อมูลได้ถูกส่งบนสัญญาณนาฬิกาเดียวกันมาจากตัวรับ shift รีจิสเตอร์ ไปยังตัวรับของดาต้ารีจิสเตอร์

ความต้องการอินเตอร์รัพรับ flip-flop ถูกตั้งบนขอบขึ้นของบิทเริ่มต้นสำหรับดาดต้ารีจิสเตอร์ว่างและที่ขอบที่ตกลงของบิทหยุดสำหรับShiftรีจิสเตอร์ที่ว่าง ถ้ามิใช่ดาต้ารีจิสเตอร์บนขอบที่เราตกลงไว้นี้ของบิตหยุด ผู้ส่งต้องทำงานด้วย ผู้ส่งไม่สามารถทำงานได้เท่านั้นถ้าดาต้ารีจิสเตอร์ว่างตำแหน่งของบิตหยุด

2.1.11.4 Transmit Serial Data Timing

ในการส่งข้อมูลถ้าอินเตอร์รัพมีการทำงาน อินเตอร์รัพจะร้องขอในขณะที่ Transmit รีจิสเตอร์ในการส่งจะว่างและในขณะเดียวกัน อินเตอร์รัพเกิดขึ้นในขณะที่ shift รีจิสเตอร์และ Transmit รีจิสเตอร์ จะว่าง เมื่อบิตสุดท้ายจะออกไป เมื่อมีการขนส่งข้อมูลรีจิสเตอร์จะมีข้อมูลและ shift รีจิสเตอร์หยุดการส่งข้อมูล ข้อมูลหามี clock จาก transmit รีจิสเตอร์ไปยังShift รีจิสเตอร์ และ Shift รีจิสเตอร์มาสามารถหยุดการทำงานได้ อินเตอร์รัพจะมีการร้องขอได้ถูกลบโดยการเขียน ไปยังดาต้ารีจิสเตอร์หรือเขียนไปยัง status รีจิสเตอร์ ดาต้ารีจิสเตอร์โดยปกติเป็นclock ไปยังShift รีจิสเตอร์แต่ละช่วงเวลา Shift รีจิสเตอร์จะจบการส่งข้อมูล ในขณะที่ดาต้ารีจิสเตอร์จะว่าง เป็นสาเหตุที่ทำให้ มีการร้องขอจากอินเตอร์รัพ ปกติอินเตอร์รัพจะสามารถตอบกลับอินเตอร์รัพก่อน Shift รีจิสเตอร์ทำงาน โดยปกติอินเตอร์รัพเก็บข้อมูลต่อไปที่ดาต้ารีจิสเตอร์ ลบการร้องขอของอินเตอร์รัพและให้มีการส่งข้อมูลต่อไป ในขณะที่ข้อมูลได้ถูกส่งนั้น Interrupt Service Routine จะตอบกลับมา อินเตอร์รัพ จะทำให้ ดาต้ารีจิสเตอร์ว่าง ตั้งแต่รีจิสเตอร์ไม่มีข้อมูลมันจะลบ การร้องขออินเตอร์รัพโดยเก็บสถานะของรีจิสเตอร์ ที่เส้นทางควรจะมีการตรวจสอบ ถ้าShift รีจิสเตอร์ว่างถ้าเกิดขึ้น จะแสดงว่าอินเตอร์รัพจะตอบช้า Interrupt Routine ควรจะทำการลบครั้งสุดท้ายและเก็บค่า Status รีจิสเตอร์ ในกรณี Shift รีจิสเตอร์กลายเป็น 0 หลังจาก ค่าที่อยู่ในอินเตอร์รัพจะถูกลบออก Service Interrupt Routine จะส่งค่าคืนและจะทำ อินเตอร์รัพสุดท้าย ส่งไปยังroutine ให้มีการหยุดการทำงานของ เอาพุตท์ บัฟเฟอร์ในกรณี RS-485

2.1.11.5 Receive Serial Data Timing

ในขณะที่ตัวรับกำลังรับข้อมูล ถ้ามีการลงของขอบสัญญาณแสดงว่าจะเริ่มต้นบิตจะต้องมีการตวจสอบ ขอบขาลงจะสามารถตวรจสอบได้ ความแตกต่างของ Rx อินพุตท์ ระหว่าง ความแตกต่างของ 2 clock เริ่มที่ 8x และ 16x ในการส่งสัญญาณ แต่ละครั้งของการเริ่มบิตจะทำการ

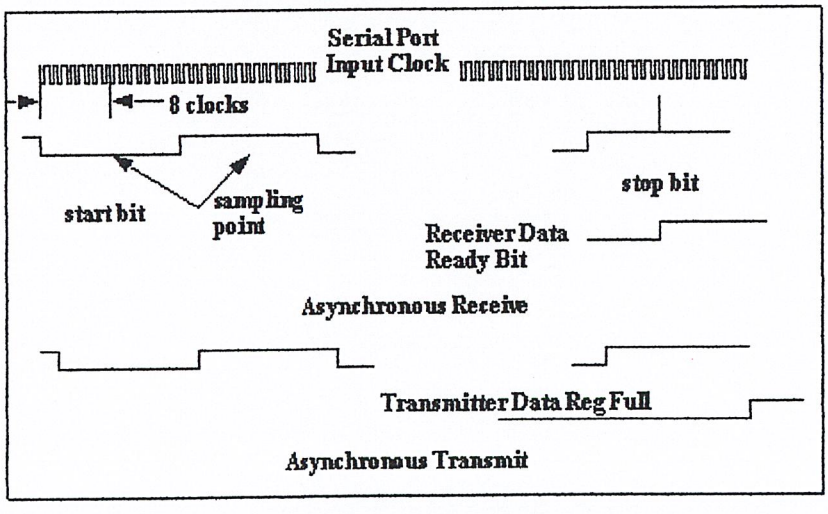
ตรวจสอบ บิตข้อมูลจะถูกการsampled บริเวณตรงกลางของแต่ละข้อมูลและถูกส่งไปยังตัวรับ รีจิสเตอร์ตัวต่อไป หลังจากที่ 7 หรือ8 บิตข้อมูลจะถูกรับ บิตต่อไปไม่เป็นบิตที่8 หรือบิตที่ 9 หรือจะหยุดการsampled ถ้า RX line ต่ำ Rx line คือค่าแอดเดรสบิตและ แอดเดรสบิตจะรับบิตในstatus รีจิสเตอร์ในขณะที่ทำงานอยู่ ถ้าแอดเดรสบิตจะทำการตรวจสอบ ตัวรับจะพยายามที่จะทำการ sample หยุดบิต ถ้าเส้นนั้นมีค่าสูงในขณะที่ sample ไปยังรีจิสเตอร์ข้อมูลตัวรับและอินเตอร์รัพ ถ้าทำงานต้องมีการร้องขอ

ในการรับอินเตอร์รัพจะต้องทำการร้องขอในขณะที่รีจิสเตอร์ข้อมูลตัวรับซึ่งจะมีข้อมูล ซึ่งจะเกิดขึ้นในขณะที่ข้อมูลจะถูกส่งมาจากรีจิสเตอร์ตัวรับตัวต่อไป ไปยัง รีจิสเตอร์ข้อมูล เป็นการเซ็ทค่าบิตที่ 7 ของรีจิสเตอร์สถานะ การร้องขอของอินเตอร์รัพและบิตที่7 จะถูกลบในขณะที่ข้อมูลนั้นถูกอ่านอยู่

ในการรับอินเตอร์รัพมีการร้องขอถ้าบิต7 มีค่าสูง อินเตอร์รัพจะถูกร้องขอเมื่อรีจิสเตอร์ข้อมูลในการส่ง กลายเป็น0 หรือ รีจิสเตอร์ส่งตัวถัดไปกลายเป็น 0

ในการรับตรวจสอบสำหรับเริ่มบิตต่อไปนี้ทันทีหลังจากหยุดบิตเพื่อตรวจสอบ เหยุดบิตเป็นเรื่องปกติในการตรวจสอบที่ sample clock nominally จะเกิดที่บิตกลางของบิตหยุด ถ้ามี 8,9, แอดเดรส บิต

ในserial clock สามารถหรับ เป็นอย่างใดอย่างหนึ่ง 16*อัตราของข้อมูล หรือ *8 อัตราของข้อมูล



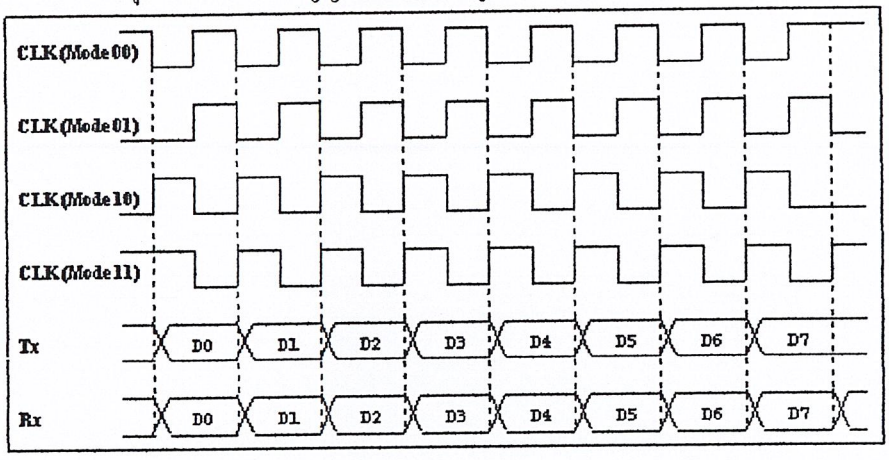
รูปที่ 2-37 Serial Port Synchronization

2.1.11.6 Clocked Serial Ports

พอร์ท A-D สามารถอยู่ใน clock โหมดได้โดยที่ data line และ clock line จะถูกไดร์วแสดง

ในรูปที่ 12-4 ข้อมูลและสัญญาณนาฬิกาจะถูกเตรียม 8 บิต burst สำหรับ LSB Shift out และ/หรือ รับครั้งแรกโดยปกติ Transmit Shift Register Advance บนการตกขอบของสัญญาณนาฬิกาและรับ sample ข้อมูลบนขอบขาขึ้นของสัญญาณนาฬิกา พอร์ทอนุกรมสามารถสร้างสัญญาณนาฬิกาได้ หรือจัดสัญญาณนาฬิกาเตรียมให้ภายนอก

Clock Polarity คือโปรแกรมใน clock serial อาจจะอยู่ในบิต (2,3) ของรีจิสเตอร์ควบคุม การทำงานของโหมดสัญญาณนาฬิกาแบบอนุกรม ด้วยอย่างได้อย่างหนึ่งโดย สัญญาณนาฬิกาจากภายนอกหรือสัญญาณนาฬิกา จากภายใน การส่งระหว่างภายในกับภายนอกควรจะทำอย่างระมัดระวัง โดยปกติที่เพิ่มขึ้นมาคือ ตัวความต้านทาน ที่จำเป็นบน สัญญาณ เพื่อป้องกัน spurious clock ในขณะที่ ในกลุ่มกำลังไดร์ว สัญญาณนาฬิกาอยู่



รูปที่2-38. Clock Polarities Supported in Clocked Serial Mode

ในโหมดสัญญาณนาฬิกาแบบอนุกรม , ชิพรีจิสเตอร์และ ดาต้ารีจิสเตอร์ ทำงานโดยใช้วิธีเดียวกันสำหรับการติดต่อแบบ ซิงโครนัส แต่อย่างไร วิธีการเริ่มแรกพื้นฐานในการส่งและรับคำสั่งจำเป็นต้องถูกสั่ง โดยการเขียนเพื่อให้บิต(7,6)ของรีจิสเตอร์ควบคุมสำหรับแต่ไบต์ส่งและรับ เป็นคำสั่งสำหรับส่งไบต์ ,มีความแตกต่างของคำสั่ งสำหรับแต่ละไบต์ที่ส่งหรือได้รับ เป็นคำสั่งสำหรับการส่งในแต่ไบต์และคำสั่งอื่นสามารถเริ่มการส่งและรับได้ในเวลาเดียวกันสำหนับการสื่อสารแบบการสื่อสารแบบ2ช่องทาง ทางอื่นๆ อ่านหรือ เขียนไปยัง Serial Port(SP) A-D มีแอดเดรสรีจิสเตอร์ (SXAR) ตัดออก ความต้องการในออกคำสั่งไม่ต่อเนื่องใรการรับและคำสั่งในการส่ง ในโหมดสัญญาณนาฬิกาแบบอนุกรม การอ่านข้อมูลจากผลตอบกลับของSxAR รีจิสเตอร์จะป็นแบบอัตโนมัติ เพราะผู้รับให้เริ่มขบวนการรับไบต์ ตัดความต้องการของซอฟแวร์ออก ให้ออกคำสั่งที่ใช้ในการเริ่มการทำงานของตัวรับ ข้อมูลที่ส่งไปจะอยู่ในบัฟเฟอร์ของตัวรับเริ่มการอ่านครั้งแรก ก่อนแทนที่ด้วยข้อมูลที่เข้ามาใหม่ การเขียนข้อมูลไปถึง รีจิสเตอร์SxAR เป็นสาเหตุให้การส่งเริ่มต้น

ขบวนการส่งไบต์ ไม่สนใจความต้องการของซอฟแวร์ในคำสั่งการเริ่มการส่ง มีผลกระทบของโค๊ต เล่านี้จะมีความแตกต่างกัน ขึ้นอยู่โดยไม่คำนึงโหมดจะเป็นสัญญาณนาฬิกาแบบภายนอก หรือ ภายใน

ในการส่งโดยใช้โหมดสัญญาณนาฬิกาแบบภายใน ผู้ใช้จะต้องโหลดจาก ดาต้ารีจิสเตอร์ และเก็บโค๊ตในการส่ง ในขณะที่ ชิพรีจิสเตอร์จบการส่งสัญลักษณ์ ถ้าอื่นๆ ดาต้ารีจิสเตอร์จะถูก โหลดเข้าไปชิพรีจิสเตอร์และถูกส่งโดย 8-clock Burst หนึ่งตัวอักษรทำงานอยู่ในการะบวนกการ ประมวลผลของการส่ง ในขณะที่ตัวอื่นๆกำลังรอในดาต้ารีจิสเตอร์เก็บค่าแบบ tagged กับคีตในการ ส่ง โค๊ตในการส่งจะมีประสิทธิภาพในการเก็บข้อมูลชั่วคราวเป็น 2เท่า

ในการรับตัวอักษรอยู่ในโหมดสัญญาณนาฬิกาภายใน รีจิสเตอร์รับตัวต่อไปควระจะหยุด การทำงาน ผู้ใช้เก็บโค๊ตตัวรับ ในรีจิสเตอร์ควบคุม การเกิดของ 8 สัญญาณนาฬิกาจะมีการสร้าง และผู้ส่งจำเป้นต้องตรวจสอบสัญญาณนาฬิกาและเอาพุตท์ตัวต่อไปไปยังสายข้อมูลบนขอบขาลง ของแต่ละสัญญาณนาฬิกา ตัวรับจะเป็นตัวอย่างของข้อมูลที่เป็นขอบขาขึ้นของแต่สัญญาณนาฬิกา สำหรับโหมดสัญญาณนาฬิกา 00 และ01 หรือขอบขาลงสำหรับโหมดสัญญาณนาฬิกา 10 และ 11 โหมดในการรับไม่สามารถทำการเพิ่มที่เก็บข้อมูลเป็น2เท่าได้ ในขณะที่ใช้สัญญาณนาฬิกาภายใน รีจิสเตอร์ตัวต่อไปจะไม่สามารถทำงานได้ก่อนที่จะรับตัวอักษรอื่นสามารถทำการเริ่มต้นได้ แต่ อย่างไรก็ตาม ร้องขออินเตอร์รัพและตัวอักษรอื่นมีการเข้ามาเมื่ออยู่ขอบขาขึ้นของฟัลส์ของ สัญญาณนาฬิกาตัวต่อไป ถ้าโค๊ตตัวรับจะถูกเก็บก่อนที่ตำแหน่งของขอบขาลงตัวต่อไป ตัวตัวรับตัว อื่นๆจะเริ่มโดยไม่มีการหยุดสัญญาณนาฬิกา อินเตอร์รัพจะยอมใน ½ สัญญาณนาฬิกา

ในการส่งแต่ละไบต์ในโหมดสัญญาณนาฬิกา โดยที่ผู้ใช้สามารถดหลดดาต้ารีจิสเตอร์และ เก็บโค๊ตการส่ง ในขณะที่รีจิสเตอร์ตัวต่อไปจะไม่ทำงานและตัวรับจะเตรียมสัญญาณนาฬิกา ดาต้า บิตจะถูกส่งไปยังรีจิสเตอร์ตัวต่อไปและเลื่อนออก โดยการส่งจะส่งไปยังรีจิสเตอร์ตัวต่อไป ไบต์ ใหม่สามารถโหลดเข้าไป รีจิสเตอร์ในการส่ง และโค๊ตการส่งใหม่จะถูกเก็บ

การรับไบต์ในโหมดสัญญาณนาฬิกาภายนอก โดยผู้ใช้จะตั้งค่ารับค็ตสำหรับค่าแรกของ ไบต์และเก็บโค๊ตที่ได้รับมา สำหรับไบต์ต่อไป หลังจากแต่ละไบต์จะถูกลบออกจากดาต้ารีจิสเตอร์ ตั้งแต่โค๊ตตัวรับจะถูกเก็บหลังจากตัวส่งส่งไปเก็บในไบต์ต่อไป เครื่องรับต้องบริการอินเตอร์รัพ ภายใน ½ นาฬิกาโดยอัตราในการส่ง เพื่อรักษาเครื่องส่งกำลังความเร็วเต็มที่ นี่ไม่ได้ฝึกถ้ามิใช่การ ควบคุมการไหลถูกทำหรือผู้ส่งใส่ช่องว่างระหว่างสัญญาณนาฬิกานั้น

โดยปกติเพื่อที่จะดำเนินการคมนาคมความเร็วสูงต่อไป การจัดการที่ดีที่สุดจะสำหรับ เครื่องรับเพื่อเตรียมสัญญาณนาฬิกา เมื่อเครื่องรับเตรียมนาฬิกา ผู้ส่งสามารถเพื่อเก็บขึ้นเพราะว่า มันสามารถบีบขนาดและมีเวลาตัวอักษรเต็มเพื่อตอบอินเตอร์รัพว่างเปล่าดาต้ารีจิสเตอร์ ผู้ส่งเสมอ.

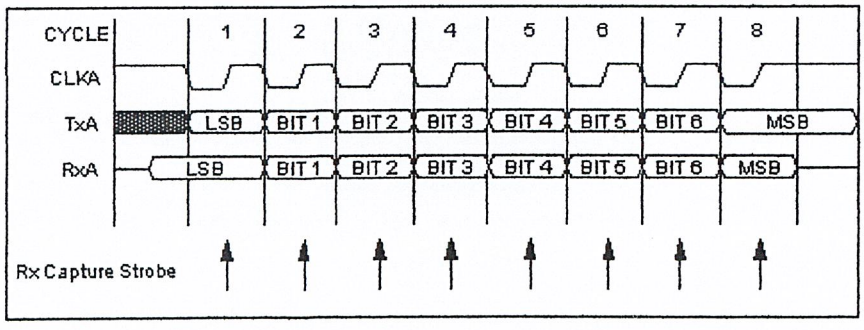
เครื่องรับจะตอบอินเตอร์รัพว่าถูกกำเนิดบนขอบที่ขึ้นของสัญญาณนาฬิกาสุดท้าย. ถ้าอินเตอร์รัพสามารถถูกบริการภายใน 1/2 สัญญาณนาฬิกา ที่นั่นจะถูกบริการไม่มีค้างในอัตราข้อมูล ถ้ามันใช้เครื่องรับนานกว่าเพื่อตอบ แล้วที่นั่นจะคือ ช่องว่างระหว่างไบท์ ความยาวซึ่งยึดหลักบนอินเตอร์รัพ ยกตัวอย่างเช่น, ถ้า อัตราความเร็วในการส่งคือ 400,000 bps, แล้วสูงถึง50,000 ไบท์ต่อที่สามารถถูกส่ง, หรือ 1 ไบท์ทุก 20 ไมโครวินาที ไม่มีข้อมูลจะถูกทำถ้าหายผู้ส่งสามารถตอบอินเตอร์รัพของมันภายใน 20 ไมโครวินาที ที่นั่นจะถูกทำไม่มีช้าลงถ้าเครื่องรับสามารถตอบอินเตอร์รัพของมันภายใน 1/2 นาฬิกาหรือ1.25 ไมโครวินาที ถ้ามันสามารถตอบภายใน 1.5 นาฬิกา, หรือ 2.75 ไมโครวินาที, อัตราข้อมูลจะช้าถึง 44,444 ไบท์ต่อวนาที ถ้ามันสามารถตอบใน 2.5 นาฬิกาหรือ6.25 ไมโครวินาที อัตราข้อมูลช้าถึง 40,000 ไบท์ต่อวินาที ถ้ามันสามารถตอบใน 3.5 นาฬิกาหรือ8.75 ไมโครวินาทีอัตราข้อมูลจะช้าถึง 36,363 ไบท์ต่อที่ และดังนี้ต่อไป.

ถ้ามีการสื่อสารแบบ2ทางเป็นสิ่งต้องการ สัญญาณนาฬิกาสามารถถูกย้อนไปมาเพื่อให้เครื่องรับเตรียมสัญญาณนาฬิกา นี่มากขึ้นทำให้ยุ่งยาก ตั้งแต่เครื่องรับไม่สามารถเริ่มข้ออความ. ถ้าความพยายามเครื่องรับเพื่อรับตัวอักษรและผู้ส่งไม่กำลังส่ง, บิทสุดท้ายที่ส่งจะถูกรับสำหรับ 8 บิททั้งหมด

2.1.11.7 Clocked Serial Timing

2.1.11.7.1 Clocked Serial Timing With Internal Clock

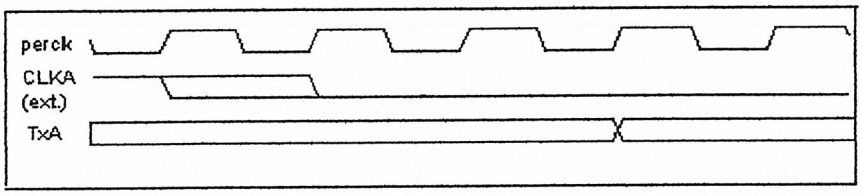
สำหรับการติดต่อสื่อสาร อะซิงโครนัสแบบอนุกรม สัญญาณนาฬิกาแบบอนุกรมสามารถสร้างได้จากRabbit หรือ อุปกรณ์ภายนอก แผงผังเวลารูปที่ 12-6 จะแสดงให้เป็นการสื่อสารแบบ2ทางละทางเดียว สัญญาณนาฬิกาแบบอนุกรมถูกสร้างภายในโดย Rabbit โหมดนาฬิกาที่สามารถเข้ากันได้ SPIอื่นๆ ที่สนับสนุนโดยกระต่าย 3000 ถูกแสดงในตัวเลข 12-5 กับนาฬิกาภายใน, อัตรานาฬิกาเป็นชุดสูงที่สุดถูกแสดง perclk/2.



รูปที่ 2-39. Full-Duplex Clocked Serial Timing Diagram with Internal Clock (Mode 00)

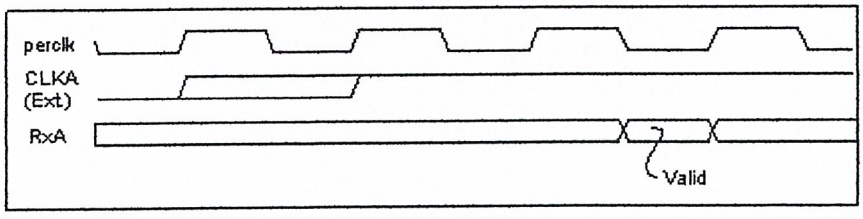
2.1.11.7.2 Clocked Serial Timing with External Clock

ในระบบที่ซึ่งสัญญาณนาฬิกาแบบอนุกรม Rabbit ถูกสร้างขึ้นโดยอุปกรณ์ภายนอก, สัญญาณนาฬิกาจะส่งสัญญาณต้องถูกเกิดขึ้นพร้อมกันกับนาฬิกาอุปกรณ์เสริมภายใน (perclk) ก่อนข้อมูลสามารถถูกส่งหรือรับโดย Rabbit การถือหลักเมื่อสัญญาณนาฬิกาแบบอนุกรมภายนอกถูกสร้างขึ้น ในความสัมพันธ์เพื่อ perclk มันอาจจะใช้ที่ใดๆจาก 2 ถึง 3 รอบสัญญาณนาฬิกาสำหรับสัญญาณนาฬิกาภายนอกที่จะถูกเกิดขึ้นพร้อมกันกับสัญญาณนาฬิกาภายในก่อนข้อมูลใดๆสามารถถูกโยกย้าย รูปที่ 12-7 จะแสดงให้เห็นถึงความสัมพันธ์ระหว่าง perclk สัญญาณนาฬิกาแบบอนุกรมแบบภายนอก และ การส่งข้อมูล



รูปที่ 2-40 Synchronous Serial Data Transmit Timing with External Clock (Mode 00)

รูปที่ 12-8 จะแสดงถึงเวลา ความสัมพันธ์ระหว่าง perclk สัญญาณนาฬิกาแบบอนุกรมแบบภายนอก และ การรับข้อมูล



รูปที่ 2-41 Synchronous Serial Data Receive Timing with External Clock (Mode 00)

2.1.11.8Serial Port Software Suggestions

เครื่องรับและผู้ส่งแชร์ใช้อินเตอร์รัพเดียวกัน แต่มันเป็นไปได้เพื่อทำเครื่องรับและส่งบริการอินเตอร์รัพทำงานประจำ (ISRs) แยกโดยการส่งไปอินเตอร์รัพเพื่อ 2 เส้นทางที่แตกต่าง นี่พึงปรารถนาเพื่อทำ ISR เชิงซ้อนน้อยลงและเพื่อรีดิวซ์อินเตอร์รัพปิดเวลา ไม่มีอินเตอร์รัพจะถูกทำตั้งแต่หายอินเตอร์รัพที่แตกต่าง flip-flops มีอยู่สำหรับรับและส่ง ผู้ส่งไปสามารถทดสอบบิทเต็มดาค้ารีจิสเตอร์เครื่องรับเพื่อส่งไป. ถ้าบิทนี้บน อินเตอร์รัพถูกส่งไปสำหรับรับ มิฉะนั้นสำหรับส่ง เครื่องรับรับการพิจารณาแรกเพราะว่ามันต้องบน อินเตอร์รัพถูกส่งไป ตั้งใจฟังที่บริการหรือข้อมูลสามารถถูกทำหาย.

```
interrupt:
push af          ; 10
ioi ld a,(SCSR)  ; 7 get status register serial port C
jp m,receive     ; 7 go service the receive interrupt
                 ;    else service transmit interrupt
```

อินเตอร์รัพจะรับรีจิสเตอร์AF ได้ถูกรักษาและสถานะของรีจิสเตอร์ได้ถูกโหลดเข้าไปในรีจิสเตอร์ A

งานประจำบริการอินเตอร์รัพสามารถเป็นทำงานของแบบฝึกหัดที่ดีและการรับการกระทำดีที่สุด ลบออกเป็นสาเหตุของอินเตอร์รัพออกและทำให้อินเตอร์รัพใช้ได้อีกครั้งทันทีที่เป็นไปได้. สิ่งนี้เก็บอินเตอร์รัพแฝงลงและยอมความเร็วเครื่องส่งกำลังเร็วที่สุดบนพอร์ทอนุกรมทั้งหมด

พอร์ทอนุกรมทั้งหมดปกติจะเกิดระดับลำดับก่อนหลัง 1 อินเตอร์รัพ ในสถานการณ์ผิดปกติ พอร์ทเป็นชุดหนึ่งหรือมากกว่าสามารถถูกแก้ไขเพื่อใช้อินเตอร์รัพลำดับก่อนหลังสูงกว่ามีข้อยกเว้นเพื่อทราบเมื่อพอร์ทอนุกรมต้องกระทำที่ความเร็วสูงอย่างที่สุด ที่ 115,200 bps ความเร็วสูงที่สุดของพอร์ทเป็นชุด PC, อินเตอร์รัพต้องถูกบริการใน 10 ครั้ง อัตราการส่งข้อมูล, หรือ 86 ไมโครวินาที ในคำสั่งไม่สูญเสียตัวอักษรที่รับมา ถ้า 6 พอร์ทอนุกรมทั้งหมดกำลังทำงานที่สิ่งนี้รับความเร็ว, มันจะจำเป็นเพื่อบริการอินเตอร์รัพใน น้อยกว่า 21.5 ไมโครวินาที เพื่อรับรองไม่มีตัวอักษรที่สูญเสีย. นอกจากนี้เวลาที่ใช้โดยอินเตอร์รัพอื่นๆของที่เท่ากันหรือลำดับก่อนหลังสูงกว่าคงต้องถูกพิจารณา. ตามปกติบริการเครื่องรับอาจจะปรากฏดังต่อไปนี้ข้างล่าง. ไบท์ที่ bufptr ถูกใช้เพื่อจ่าหน้าบริเวณเก็บข้อมูลชั่วคราวที่ซึ่งบิทข้อมูลถูกตั้งอยู่. มันจำเป็นเพื่อรักษาและ เพิ่มไบท์นี้เพราะว่าตัวอักษรสามารถถูกจัดการออกจากคำสั่งถ้า 2 อินเตอร์รัพเครื่องรับใช้สถานที่ในการสืบต่อที่รวดเร็ว

```
   receive:
push hl          ; 10 save hl
push de          ; 10 save de
ld hl,struct     ; 6
ld a,(hl)        ; 5 get in-pointer
ld e,a           ; 2 save in pointer in e
```

```
inc hl          ; 2 point to out-pointer
cmp a,(hl)      ; 5 see if in-pointer=out-pointer (buffer
full)
jr z,roverrun   ; 5 go fix up receiver over run
inc a           ; 2 incement the in pointer
and a,mask      ; 4 mask such as 11110000 if 16 buffer
locs
dec hl          ; 2
ld (hl),a       ; 6 update the in pointer
ioi ld a,(SCDR) ; 11 get data register port C, clears
interrupt request
ipres                   ; 4 restore the interrupt priority
; 68 clocks to here
; to level before interrupt took place
; more interrupts could now take place,
; but receiver data is in registers
; now handle the rest of the receiver interrupt routine
ld hl,bufbase   ; 6
ld d,0          ; 6
add hl,de       ; 2 location to store data
ld (hl),a       ;  6 put away the data ไบต์
pop de          ;7
pop hl          ; 7
pop af          ; 7
ret             ; 8 from interrupt
; 117 clocks to here
```

โดยปกติรับอินเตอร์รัพที่เปิดเกี่ยวกับ 68 นาฬิกาหรือ 3.5 ไมโครวินาทีที่ความเร็วนาฬิกา ของ20 เมกะเฮิร์ซ ถึงแม้ว่า 2 ตัวอักษรอาจจะถูกจัดการออกจากคำสั่ง สิ่งนี้จะไม่ถูกมองเห็นเมื่ออยู่ในระดับสูงกว่าที่ตรวจสถานะของบริเวณเก็บข้อมูลชั่วคราวสิ่งที่นำเข้าเพราะว่าอินเตอร์รัพทั้งหมดจะถูกเสร็จสิ้นก่อน ปกติระดับสูงกว่าสามารถกระทำเช็คบนสถานะบริเวณเก็บข้อมูลชั่วคราว วิธีทางการพิมพ์เพื่อจัดระเบียบบริเวณเก็บข้อมูลชั่วคราวต้องมีหนึ่งในตัวชี้และออกตัวชี้เพิ่มขึ้นนั้นผ่านที่อยู่ในบริเวณเก็บข้อมูลชั่วคราวข้อมูลให้วิธีกระจาย ปกติอินเตอร์รัพจับในตัวชี้และสูงกว่าจับต้องออกตัวชี้. ถ้า ในตัวชี้เท่ากับ เอาท์ตัวชี้, บริเวณเก็บข้อมูลชั่วคราวถูกพิจารณาเต็ม ถ้าเอาท์ตัวชี้บวก 1 เท่ากับอินท์ตัวชี้, บริเวณเก็บข้อมูลชั่วคราวว่างเปล่า การเพิ่มทั้งหมด ถูกทำใน แฟชั่น circular, มากที่สุดทำให้เสร็จโดยง่ายโดยการทำบริเวณเก็บข้อมูลชั่วคราวของ 2 ในความยาว, แล้วการและหน้ากากหลังการเพิ่ม ที่อยู่หน่วยความจำตามความเป็นจริงคือตัวชี้บวกที่อยู่เบสบริเวณเก็บข้อมูลชั่วคราว.

บทที่ 3

การออกแบบระบบ

ในโครงงานนี้ จะกล่าวถึงการทำงานของ Rabbit ในการนำมาประยุกต์ในชีวิตประจำวัน โดยมีการทำงานคร่าวๆดังต่อไปนี้

การทำงาน Rabbit ในการควบคุมอุปกรณ์เครื่องใช้ไฟฟ้าภายในบ้านจะแบ่งการทำงานออกเป็น 2 ส่วนใหญ่

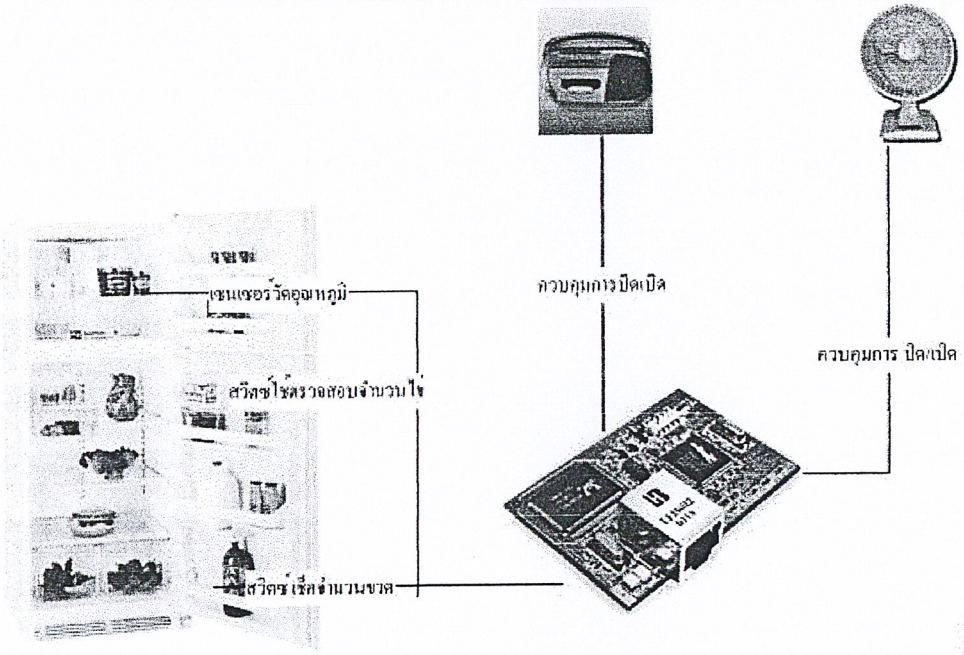
1.ในการเชื่อมต่อกับอุปกรณ์ไฟฟ้าภายในบ้านต่างๆ

1.1 ควบคุมการปิด/เปิดเครื่องใช้ไฟฟ้า

1.2 ควบคุมอุณหภูมิในตู้เย็น

1.3 รับข้อมูลจากสวิตซ์ที่อยู่ส่วนต่างๆของตู้เย็น

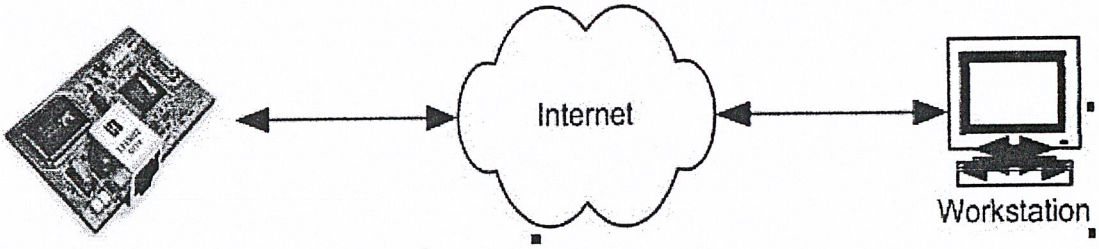
1.4 สวิตซ์ปิด/เปิดตู้เย็น



รูปที่ 3.1 การทำงานระหว่าง Rabbit กับอุปกรณ์ต่าง

2. ส่วนที่ให้ Rabbit เป็น Server ในการเชื่อมต่อกับ Internet

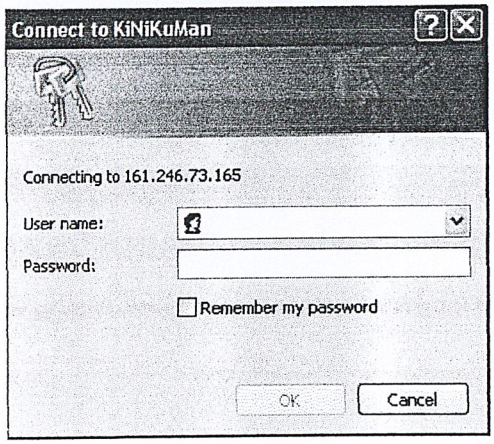
โดยส่งแบบ ดาต้าแพกเกจเพราะเราไม่มีความจำเป็นในการติอต่อแบบ REAL TIME โดยให้ Rabbit เป็น Server ขนาดเล็กซึ่ง Rabbit มีความหน่วยความจำและความสามารถเพียงพอในการทำงานประเภทนี้



รูปที่ 3.2 เป็นการเชื่อต่อระหว่าง Rabbit กับ work Station

ซึ่ง Rabbit จะทำหน้าที่ได้ 2 อย่างในเวลาเดียวกัน โดยเป็น

1. **ตัวประมวลผล** ซึ่ง Rabbit จะคอยรับค่า จากอุปกรณ์ต่างเพื่อนำมาประมวลผล ซึ่งเราจะสามารถเขียนโปรแกรมไปกำหนดการทำงานได้
2. **เซฟเวอร์** ซึ่งจะคอยเก็บข้อมูลและแสดงข้อมูลโดยผ่านทางอินเตอร์เน็ต และก่อนที่มีการเข้าต้องถึงตัวโปแกรมต้องผ่านใส่password ก่อน ดังรูป

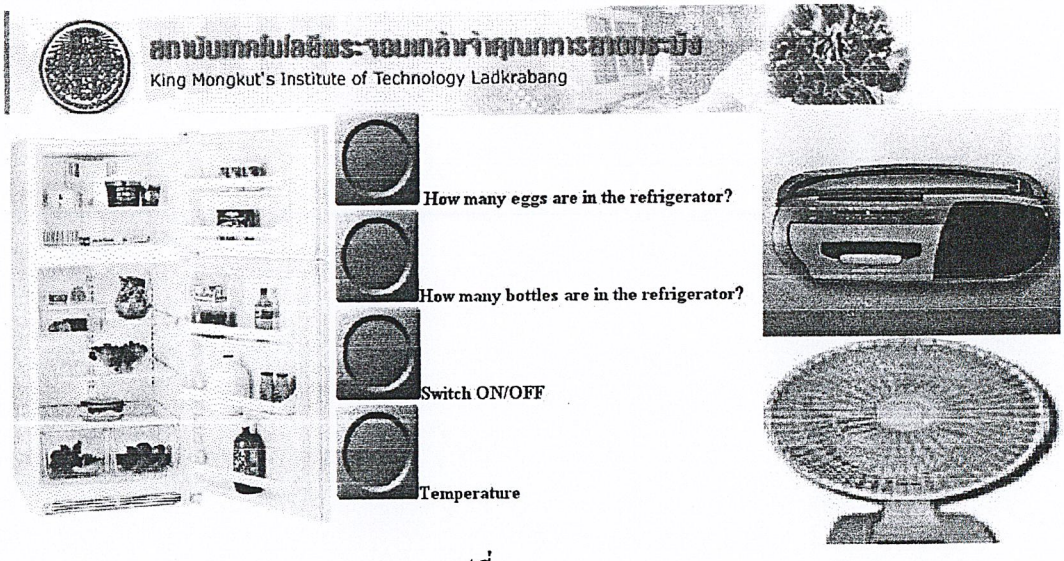


รูปที่ 3.3 ระบบการรักษาความปลอดภัยกาอยเข้าถึงตัวโปรแกรมใน Rabbit

บทที่ 4

ผลการทดลอง

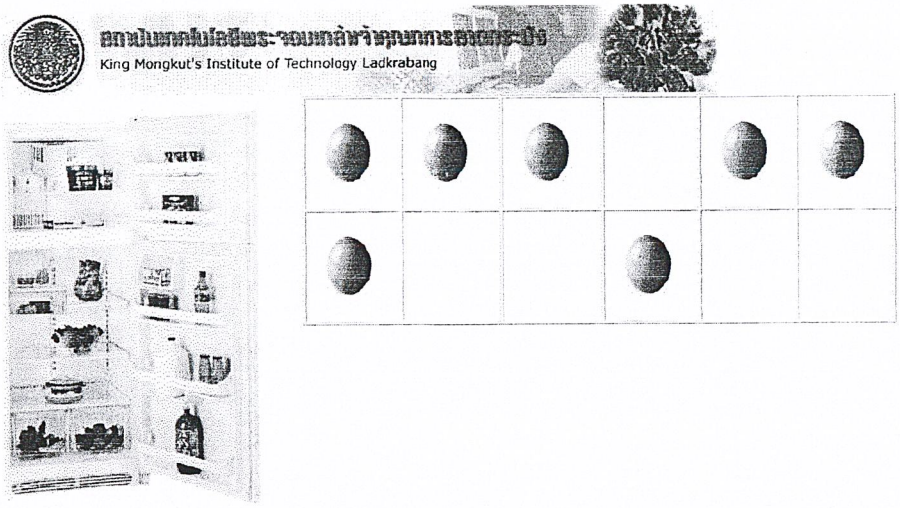
Interface ที่ผู้ใช้เรียกดูสถานะและควบคุมอุปกรณ์ต่างๆในบ้านโดยผ่านทางอินเตอร์เน็ต โดยใช้ Rabbit ซึ่ง Rabbit สามารถทำตัวเป็น Server คอยเก็บข้อมูลและแสดงสถานะของเครื่องใช้ไฟฟ้า ดังรูป



รูปที่ 4-1

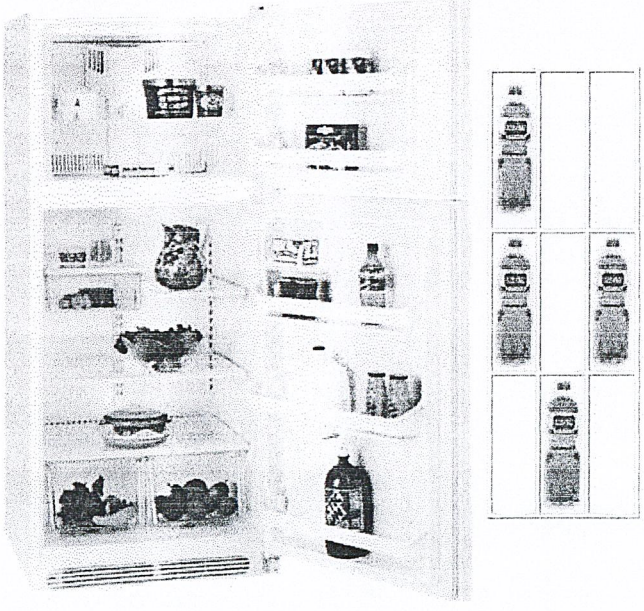
เป็นรูปแสดงหน้าแรกของเว็บ ซึ่ง Rabbitเป็น web server อยู่ โดยที่เราจะสามารถเข้าไปดูข้อมูลต่างๆได้ดังตัวเลือก

1.จะแสดงถึงสถานะของไข่ในตู้เย็นโดยได้รับข้อมูลจากรางไข่ส่งมายัง Rabbit แล้วไปประมวลผล เมื่อประมวลผลเสร็จก็ขึ้นบน Server ดังรูป



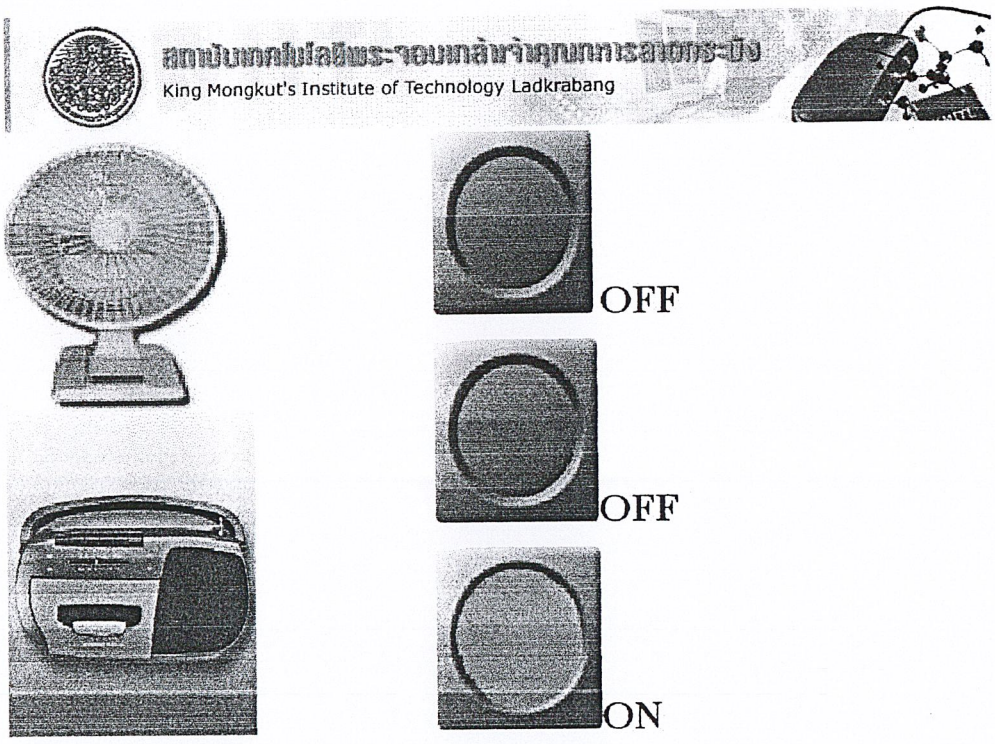
รูปที่ 4-2 แสดงสถานะของหน้าจอของไข่

2 .จะแสดงถึงสถานะของน้ำขวดในตู้เย็นโดยได้รับข้อมูลจากส่งมายัง Rabbit แล้วไปประมวลผล เมื่อประมวลผลเสร็จก็ขึ้นบน Server เพื่อแสดงผลดังรูป



รูปที่ 4-3 แสดงสถานะของขวดบนหน้าจอ

3. ควบคุมการทำงานของเครื่องใช้ไฟฟ้า



รูปที่ 4.5 แสดงถึงการควบคุมเครื่องใช้ไฟฟ้าผ่าน อินเตอร์เน็ต

บทที่ 5

สรุปและวิจารณ์ผลการทดลอง

ในการทดลองของโครงงานนี้สามารถควบคุมเครื่องใช้ไฟฟ้าผ่านอินเตอร์เน็ตโดยใช้ Rabbit 3000 ในการเชื่อโยงซึ่งสามารถเป็น Web Server ไปในตัวได้ด้วยซึ่งเป็นก้าวแรกในการนำ Rabbit มาใช้ประโยชน์ได้อีกในอนาคตและยังสามารถนำไปพัฒนาได้อีกมากในอนาคต

5.1 ปัญหาในการทดลอง

5.1.1 ในปัจจุบันยังไม่เป็นที่แพร่หลายในประเทศ ทำให้มีหนังสือหรือบทความน้อยจึงเป็นการยากมากที่จะหาข้อมูลจากแหล่งต่าง

5.1.2 Rabbit ค่อนข้างจะมีราคาที่สูงและหาได้ยาก

5.1.3 ปัญหาในเรื่อง อุปกรณ์การใช้งานเพราะ ว่าไม่มีคู่มือภาษาไทย

5.1.4 กำหนดขอบเขตของงานกว่างเกินไป ทำให้เกิดปัญหาการเริ่มต้นและหาจุดมุ่งหมายของงาน

5.2.5 ไม่มีประสบการณ์ทางด้านฮาร์ดแวร์

5.2 แนวทางการแก้ไข

5.2.1 ดาว์โหลดข้อมูลต่างๆจากเว็บที่เราได้ซื้อ Rabbit มา

5.2.2 ปรึกษากับอาจารย์เพื่อไขปัญหาข้อข้องใจได้

5.3 แนวทางการพัฒนา

5.3.1 ทำการศึกษาอุปกรณ์ไฟฟ้า ชนิดต่างๆ เพื่อนำมาพัฒนาใร โครงงานเพิ่มขึ้น

5.3.2 ทำการศึกษาให้ Rabbit ไปประยุกต์ไปใช้กับไมโครโปรเซสเซอร์ตัวอื่นๆ

5.3.3 ทำการศึกษานำไปใช้ในบนระบบเครือข่ายได้เพิ่มขึ้นอีก

5.3.4 สามารถดัดแปลงไปเป็นโครงงานไปเป็นธุรกิจได้

บรรณานุกรม

1. http://www.sonoma.edu/users/s/serceki/Lecture/Week_1/rabbit3000.pdf
2. http://www.rabbitsemiconductor.com/.../ UsersManual/2hard.htm
3. www.beyondlogic.org/ etherip/ip.htm
4. www.vtec.ch/rabbit.htm