สำนักหอสมุดกลาง พระจอมเกล้าลาดกระบัง

ซีโร่ มัลติเพล็กซ์เซอร์

ZERO MULTIPLEXER

T042156

โดย

นาย ปัฐวิกร พลอยประเสริฐ

นางสาว ภิรดี พิทยาธิคุณ

นาย ฤชิษฐ์ หิตะวัฒนกุล

เลขหมู่.....
เลขทะเบียน 42156
วัน, เดือน, ปี 14 พ.ค. 2545

.b.....
.i.....

ปริญญานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิศวกรรมศาสตรบัณฑิต

สาขาวิศวกรรมโทรคมนาคม

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

ปีการศึกษา 2543

b41270850

ซีโร่ มัลติเพล็กซ์เซอร์

ZERO MULTIPLEXER

โดย

นาย ปัฐวิกร พลอยประเสริฐ 40010455

นางสาว ภิรดี พิทยาธิคุณ 40010565

นาย ฤชิษฐ์ หิตะวัฒนกุล 40010649

อาจารย์ที่ปรึกษา

อาจารย์ กฤษณ์ วงรุจิระ

ดร. สุทธิชัย นพนาคีพงษ์

รายงานนี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิศวกรรมศาสตรบัณฑิต

สาขาวิชาวิศวกรรมโทรคมนาคม

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

ปีการศึกษา 2543

ปริญญานิพนธ์ปีการศึกษา 2543

ภาควิชาวิศวกรรมโทรคมนาคม

คณะวิศวกรรมศาสตร์ สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

เรื่อง ซีโร่ มัลติเพล็กซ์เซอร์

ZERO MULTIPLEXER

ผู้จัดทำ

1. นายปัฐวิกร พลอยประเสริฐ 40010455
2. นางสาวภิรดี พิทยาธิคุณ 40010565
3. นายฤชิษฐ์ ทิตะวัฒนกุล 40010649

อาจารย์ที่ปรึกษา

(อ.กฤษณ์ วงรุจิระ)

อาจารย์ที่ปรึกษา

(ดร.สุทธิชัย นพนาคีพงษ์)

ซีโร่ มัลติเพล็กซ์เซอร์

ZERO MULTIPLEXER

โดย 1. นาย ปัฐวิกร พลอยประเสริฐ 40010455

2. นางสาว ภิรดี พิทยาธิคุณ 40010565

3. นาย ฤชิษฐ์ ทิตะวัฒนกุล 40010649

อาจารย์ที่ปรึกษา อ. กฤษณ์ วงรุจิระ

ดร. สุทธิชัย นพนาคีพงษ์

บทคัดย่อ

ปริญญานิพนธ์นี้เป็นการศึกษาและสร้างเครื่องรับส่ง ที่สามารถส่งข้อมูลได้พร้อมกัน จากนั้นนำสัญญาณที่ได้ไปมัลติเพล็กซ์ต่อโดยวิธีการของการมัลติเพล็กซ์แบบแบ่งเวลาเชิงสถิติ ข้อมูลที่ได้จะถูกส่งผ่านตัวกลางไปทางด้านรับ ส่วนทางด้านรับก็จะทำการดิมัลติเพล็กซ์เอาข้อมูลสัญญาณต่างๆออกมาได้

ABSTRACT

This project presents the transceiver which can simultaneously transmit the data after that, the digital signal is multiplexed by means of statistical time division multiplex. The data is transmitted though the media to receiver . Next the receiver will demultiplex the data from input channel.

สารบัญ

		หน้า
บทที่ 1	บทนำ	1
บทที่ 2	ทฤษฎีและหลักการ	2
	2.1 ระบบการสื่อสาร	2
	2.2 การพัฒนาการส่งข้อมูลแบบดิจิตอล	6
	2.3 โครงสร้างของวงจรส่งข้อมูลแบบดิจิตอล	7
	2.4 เทคนิคการส่งผ่านข้อมูล	9
	2.5 รูปแบบการส่งข้อมูลดิจิตอล	10
	2.6 วิธีการส่งข้อมูลแบบดิจิตอลมัลติเพล็กซ์	17
	2.7 การใช้สายสื่อสารร่วมกัน	21
	2.8 มัลติเพล็กซ์เซอร์	25
	2.9 การเชื่อมต่อกับอุปกรณ์ภายนอก	43
บทที่ 3	หลักการออกแบบ	46
	3.1 บล็อกไดอะแกรมแสดงการส่งผ่านข้อมูล	46
	3.2 บล็อกไดอะแกรมแสดงการมัลติเพล็กซ์สัญญาณ	46
	3.3 ชุดดิพสวิทซ์(Dip Switch)	47
	3.4 ส่วนการเชื่อมต่อกับอุปกรณ์ปลายทางข้อมูล (DTE)	47
	3.5 ส่วนของการรับส่งข้อมูลความเร็วต่ำแบบอะซิงโครนัส	48
	3.6 กระบวนการมัลติเพล็กซ์	51
	3.7 ส่วนของการรับส่งข้อมูลความเร็วสูงแบบซิงโครนัส	52
	3.8 ไมโครคอนโทรเลอร์	54
บทที่ 4	การทดลองและผลการทดลอง	60
	4.1 การทดสอบการลูปกลับ	60
	4.2 การทดสอบการเชื่อมโยง (Link Test)	60
	4.3 การทดสอบวงจรรับส่งข้อมูลผ่านพอร์ตอนุกรม	64
	4.4 การทดสอบวงจรรับส่งข้อมูลแบบซิงโครไนเซชั่น	64
บทที่ 5	บทวิจารณ์และสรุปผลการทดลอง	70

สารบัญรูป

รูปที่		หน้า
2.1	องค์ประกอบของระบบสื่อสาร	2
2.2	การสื่อสารแบบซิมเพล็กซ์	4
2.3	การสื่อสารแบบฮาล์ฟดูเพล็กซ์	5
2.4	การสื่อสารแบบฟูลดูเพล็กซ์	6
2.5	โครงสร้างพื้นฐานของวงจรส่งข้อมูลดิจิตอล	7
2.6	การส่งแบบอนุกรม	9
2.7	การส่งแบบขนาน	10
2.8	การส่งข้อมูลแบบอะซิงโครนัส	12
2.9	เฟรมข้อมูลคาร์แรคเตอร์โอเรียนต์	13
2.10	เฟรมข้อมูลบิตโอเรียนต์	14
2.11	การตรวจหาตัวอักษร SYN ของ USART	15
2.12	ป้องกันการตรวจหาตัว SYN ผิดพลาด	16
2.13	ดิจิตอลไฮเออราคีของแต่ละประเทศ	18
2.14	โครงสร้างแบบดิจิตอลไฮเออราคีที่รวมเป็นหน่วยเดียวในโลก	18
2.15	หลักการมัลติเพล็กซ์สัญญาณดิจิตอล	20
2.16	การเชื่อมโยงระหว่างเทอร์มินัลกับโฮสต์แบบจุดต่อจุด	21
2.17	การใช้สายร่วมกันระหว่างเทอร์มินัล	22
2.18	รูปแบบของมัลติดรอปไลน์	23
2.19	กระบวนการโพลลิ่ง	24
2.20	กระบวนการชีเล็กติง	24
2.21	การมัลติเพล็กซ์เมื่อส่งข้อมูลจากหลายเทอร์มินัล	26
2.22	ภาพ FDM และTDM	27
2.23	การมัลติเพล็กซ์ชนิดแบ่งตามความถี่	27
2.24	การทำงานของ TDM	29
2.25	บิต-อินเตอร์ลีฟวิง	30
2.26	เฟรมซึ่งมีเฮดเดอร์	31
2.27	การมัลติเพล็กซ์ข้อมูลจากซิงโครนัสเทอร์มินัล	31

รูปที่ หน้า

2.28	การใช้มัลติสตรีมโมเด็มในรูปแบบต่างๆ	32
2.29	การทำงานของ STDM	34
2.30	การมัลติเพล็กซ์ TDM เทียบกับ STDM	35
2.31	รูปแบบเฟรม TDM	36
2.32	ตัวอย่าง STDM ร่วมกับ TDM	37
2.33	STDM ในรูปแบบวงแหวน	38
2.34	STDM ในรูปแบบของมัลติดรอปไลน์	38
2.35	STDM เซอร์วิสเรโช	39
2.36	ปัญหาของบิตซิงโครไนเซชั่น	41
2.37	การส่งสัญญาณเวลาพร้อมกับสัญญาณข้อมูล	42
2.38	การมัลติเพล็กซ์สัญญาณข้อมูลจากเทอร์มินัลหลายชั้น	42
2.39	การมัลติเพล็กซ์สัญญาณหลายรูปแบบ	43
2.40	ระบบสายเคเบิลส่งข้อมูลฟูลดูเพล็กซ์ตามมาตรฐาน	44
2.41	เปรียบเทียบ TDM กับ STDM	45
3.1	การส่งผ่านข้อมูล	46
3.2	บล็อกไดอะแกรมของชุดรับส่งสัญญาณ	46
3.3	วงจรดิพสวิตซ์สำหรับ UART	47
3.4	วงจร RS232	48
3.5	โครงสร้างภายในของ 16550	49
3.6	รูปวงจร 16550	50
3.7	วงจรสร้างสัญญาณนาฬิกาและซิงโครนัส	51
3.8	โครงสร้างภายใน 8251	52
3.9	โฟลว์ชาร์ทแสดงการทำงานและลำดับการโปรแกรม	53
3.10	แผนผังระบบโปรแกรมทางด้านส่ง	55
3.11	แผนผังระบบโปรแกรมทางด้านรับ	56
3.12	วงจรไมโครคอนโทรลเลอร์	57
3.13	วงจรรวม	58

รูปที่ หน้า

4.1	บล็อกไดอะแกรมการทดสอบลูปกลับ	59
4.2	ผลการทดลองในลูปกลับ	60
4.3	การเชื่อมโยงระหว่างคู่สถานีของเครื่องมัลติเพล็กซ์ข้อมูล	61
4.4	การเลือกบิตเรทในการรับส่งข้อมูล	61
4.5	การเริ่มต้นส่งข้อมูลจากสถานีต้นทาง	62
4.6	การรับข้อมูลทางสถานีด้านรับปลายทาง	62
4.7(ก)	แสดงการรับส่งได้ระหว่างสถานีส่งและสถานีรับแบบเป็นตัวอักษร	63
4.7(ข)	แสดงการรับส่งได้ระหว่างสถานีส่งและสถานีรับแบบเป็นข้อความ	63
4.8	สัญญาณทีทีแอลเทียบกับระดับสัญญาณแรงดันมาตรฐาน RS-232	64
4.9	สัญญาณแซมปลิ้งของ UART	64
4.10	สัญญาณนาฬิกาเปรียบเทียบกับสัญญาณซิงโครไนเซชั่น	65
4.11	สัญญาณซิงโครไนเซชั่นและสัญญาณนาฬิกา	65
4.12	สัญญาณนาฬิกาเปรียบเทียบกับสัญญาณซิงโครไนเซชั่นและข้อมูล	66
4.13	สัญญาณซิงโครไนเซชั่นและสัญญาณซิงดีเทค	66
4.14	สัญญาณซิงโครไนเซชั่นและข้อมูลเทียบกับสัญญาณ TxRdyด้านส่ง	67
4.15	สัญญาณซิงโครไนเซชั่นและข้อมูลเทียบกับสัญญาณ TxRdyด้านรับ	68
4.16	สัญญาณซิงโครไนเซชั่นและข้อมูลเทียบกับสัญญาณ RxRdy	68

สารบัญตาราง

ตาราง	หัวเรื่อง	หน้า
ตารางที่ 2.1	ชนิดของอุปกรณ์ปลายทางวงจรข้อมูล	8
ตารางที่ 2.2	ดิจิตอลไฮเออราคี	17
ตารางที่ 2.3	ชนิดของอุปกรณ์ดิจิตอลมัลติเพล็กซ์	19
ตารางที่ 3.1	ระดับสัญญาณของสถานะลอจิกแบบทีทีแอล	48



บทที่ 1
บทนำ

ความเจริญก้าวหน้าทางเศรษฐกิจและการขยายตัวทางสังคมในปัจจุบันทำให้มีความต้องการในการแลกเปลี่ยนข่าวสารและข้อมูลเพิ่มขึ้นอย่างรวดเร็ว ความต้องการในการใช้เส้นทางในการสื่อสารข้อมูลมีมากขึ้น วิธีการมัลติเพล็กซ์เป็นวิธีหนึ่งในการที่จะนำข้อมูลจากหลายๆแหล่ง ส่งไปในช่องสัญญาณเดียวกัน และทางด้านรับก็จะทำการดีมัลติเพล็กซ์ เพื่อแยกข้อมูลออกไปตามช่องข้อมูลอย่างถูกต้องเพื่อให้สามารถตอบสนองความต้องการในการใช้สายส่งข้อมูลเพื่อการติดต่อสื่อสารได้มากขึ้น

แนวความคิดของปริญญานิพนธ์นี้คือ สร้างอุปกรณ์สื่อสาร ในส่วนของการรับส่งข้อมูลความเร็วต่ำที่มีการรับส่งข้อมูลกันแบบอะซิงโครนัส แล้วแล้วทำการรวบรวมข้อมูลความเร็วต่ำทั้ง 4 ช่องสัญญาณเข้าด้วยกัน โดยใช้วิธีการมัลติเพล็กซ์แบบ STDM (Statistical Time Division Multiplex)แล้วจึงทำการรับส่งข้อมูลกันระหว่าง 2 สถานีด้วยความเร็วตามมาตรฐานของ ซีโร่ มัลติเพล็กซ์(64 กิโลบิตต่อวินาที)โดยใช้วิธีการซิงโครนัสแบบ Character Oriented หลังจากนั้นจึงทำการแยกสัญญาณกลับไปยังช่องสัญญาณที่ถูกต้อง

บทที่ 2
ทฤษฎีและหลักการ

2.1 ระบบการสื่อสาร

การสื่อสารข้อมูล (Data Communication) เป็นส่วนประกอบสำคัญในระบบการประมวลผลโดยคอมพิวเตอร์ในแบบต่างๆ ไม่ว่าจะเป็นระบบออนไลน์ ระบบประมวลผลระยะไกล และระบบการประมวลผลแบบกระจาย แต่ก่อนจะอธิบายถึงการสื่อสารข้อมูล จะขอกล่าวถึงระบบการสื่อสารโดยทั่วๆไปก่อน เพื่อที่ต่อไปจะได้เข้าใจใน *ระบบการสื่อสารข้อมูล* ซึ่งเป็นแขนงหนึ่งของ *ระบบการสื่อสาร* ได้โดยง่าย

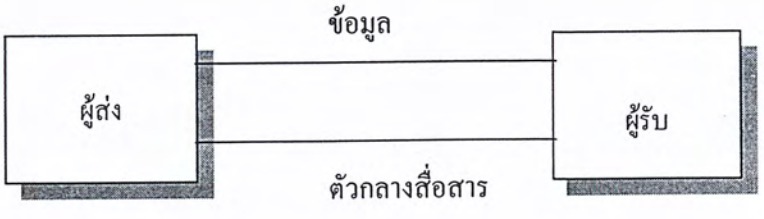
2.1.1 จุดประสงค์ของการสื่อสาร

จะเห็นได้ว่ารูปแบบของการสื่อสารมีลักษณะต่างๆกันไป แต่ไม่ว่าจะเป็นรูปแบบใดก็ตาม จุดประสงค์ของการสื่อสารจะเหมือนกันคือ การส่งข่าวสาร (Information) จากที่หนึ่งไปยังอีกที่หนึ่ง สำหรับข่าวสารที่ถูกส่งออกไปอาจจะเป็นเรื่องที่กำลังคุยอยู่ เป็นข้อความในจดหมาย หรือเป็นเสียงดนตรีก็ได้ สำหรับในการสื่อสารข้อมูล จุดประสงค์ก็คือ การส่งข้อมูล (Data or Encoded Information) เช่น ตัวอักษรซึ่งอยู่ในลักษณะของข้อมูลฐานสองที่ถูกเข้ารหัส เช่น รหัสแอสกี (ASCII Code) เป็นต้น

2.1.2 องค์ประกอบของระบบการสื่อสาร

โดยปกติ ในการส่งข่าวสารจากที่แห่งหนึ่งไปยังอีกแห่งหนึ่ง จะต้องมีองค์ประกอบของระบบอย่างน้อย 3 ประการ คือ

1. ผู้ส่ง (Source) : สร้างข้อมูลและส่งข้อมูลไป
2. ผู้รับ (Receiver) : ปลายทางของการส่งข่าวสาร
3. ตัวกลางในการส่ง (Transmission) : นำข้อมูลจากผู้ส่งไปให้ผู้รับ



รูปที่ 2.1 แสดงองค์ประกอบของระบบการสื่อสาร

ตัวอย่างเช่น ในการอ่านหนังสือ ผู้ส่งก็คือผู้เขียน ผู้รับก็คือผู้อ่าน และตัวกลางก็คือหนังสือ ในการสนทนาระหว่างบุคคล ผู้ส่งคือผู้ที่กำลังพูด ผู้รับก็คือผู้ที่กำลังฟัง และตัวกลางในการสื่อสารก็คืออากาศหรือคลื่นเสียงนั่นเอง

นอกจากจะทราบถึงองค์ประกอบของการสื่อสารแล้ว ควรจะทราบถึงสิ่งซึ่งสัมพันธ์กับการทำงานของระบบการสื่อสารด้วย ซึ่งจะทำให้ระบบการสื่อสารนั้นดีหรือเลวอย่างไร ได้แก่

1. การสื่อสารจะมีประสิทธิภาพก็ต่อเมื่อข่าวสารสามารถเข้าใจได้ดี เช่น ในการพูดคุยกัน จะประสบผลก็ต่อเมื่อใช้ภาษาที่ทั้ง 2 ฝ่ายเข้าใจกัน หรือในกรณีของการส่งข้อมูลระหว่างอุปกรณ์คอมพิวเตอร์ 2 เครื่องนั้น หากฝ่ายส่ง ได้ทำการส่งข้อมูลด้วยอัตราเร็ว 1,200 บิตต่อวินาที (Bit per second : Bps) แต่ฝ่ายรับถูกตั้งให้รับข้อมูลได้ด้วยอัตราเร็ว 9,600 บิตต่อวินาที การรับข้อมูลนั้นจะไม่สำเร็จ

2. คุณลักษณะเฉพาะตัว (Individual Characteristics) ขององค์ประกอบแต่ละอย่าง ไม่ว่าผู้รับ ผู้ส่ง หรือตัวกลาง จะเป็นตัวกำหนดคุณลักษณะทั้งหมด (Overall Characteristics) ของระบบสื่อสารได้ เช่น การพูดคุยกันทางโทรศัพท์นั้น จะไม่สามารถมองเห็นหน้าตาท่าทางของผู้พูดได้ หรือการส่งข้อมูลนั้น หากฝ่ายส่งสามารถส่งข้อมูลได้ถึง 9600 บิตต่อวินาที และฝ่ายรับสามารถรับข้อมูลได้ถึง 19200 บิตต่อวินาที แต่ตัวกลางยอมให้ข้อมูลผ่านได้แค่ 2400 บิตต่อวินาที จะทำให้ทั้งระบบถูกจำกัดอัตราการส่งข้อมูลได้แค่เพียง 2400 บิตต่อวินาที หรือในกรณีที่เป็นการส่งผ่านข้อมูลระหว่างเครือข่าย (Internetwork) อัตราการส่งข้อมูลของทั้งระบบนั้นจะถูกจำกัดด้วยการส่งของเครือข่ายที่ยอมให้ข้อมูลผ่านด้วยอัตราที่ต่ำที่สุด คล้ายกับการไหลของน้ำผ่านท่อขนาดต่างๆกัน อัตราการไหลของน้ำจะถูกจำกัดด้วยขนาดของท่อที่เล็กที่สุดเช่นเดียวกัน

3. การรบกวน เกิดขึ้นได้เสมอในระบบการสื่อสารใดๆ ซึ่งหากการรบกวนมีมาก ก็จะส่งผลให้เกิดความผิดพลาดของการส่งข้อมูลได้มาก และทำให้ข้อมูลที่ส่งผ่านระบบการสื่อสารนั้นถูกส่งด้วยอัตราที่ต่ำลง

2.1.3 นิยามการส่งข้อมูล

ในตอนต้นคงได้ทราบแล้วว่า องค์ประกอบของการสื่อสารนั้น อย่างน้อยจะต้องประกอบด้วยส่วนผู้รับ ส่วนผู้ส่ง และตัวกลาง ซึ่งในการติดต่อสื่อสารเพื่อส่งข้อมูลระหว่างผู้ส่งและผู้รับโดยผ่านตัวกลางนั้น ปกติจะทำได้ 3 ลักษณะคือ

1.การสื่อสารแบบซิมเพล็กซ์ (Simplex)

2. การสื่อสารแบบฮาล์ฟดูเพล็กซ์ (Half-Duplex)

3. การสื่อสารแบบฟูลดูเพล็กซ์ (Full-Duplex)

การสื่อสารแบบซิมเพล็กซ์ หรือการสื่อสารแบบทางเดียว

(One Way Communication)

ตัวกลางในการส่งข้อมูลนั้น อาจจะพูดได้ว่าประกอบด้วยช่องสัญญาณ (channel) ซึ่งอนุญาตให้ข้อมูลผ่านได้ช่องเดียวหรือหลายช่อง ช่องสัญญาณนี้คล้ายกับท่อน้ำ โดยจะไหลผ่านจากแหล่งกำเนิดน้ำไปยังแหล่งรับน้ำ ทำนองเดียวกันกับการส่งข้อมูลดังแสดงในรูป ซึ่งจะประกอบด้วยช่องสัญญาณเพียงช่องเดียว และปลายทางด้านหนึ่งเป็นผู้รับ การส่งข้อมูลจะเป็นลักษณะที่ข้อมูลถูกส่งไปทางเดียวกัน หรือ One Way Communication ตัวอย่างเช่น การแพร่ภาพทางโทรทัศน์ การกระจายเสียงของสถานีวิทยุต่างๆ การส่งน้ำตามท่อ หรือการจราจรระบบทางเดียว เป็นต้น



รูปที่ 2.2 แสดงการสื่อสารแบบซิมเพล็กซ์

การสื่อสารแบบฮาล์ฟดูเพล็กซ์ หรือการสื่อสารแบบทางใดทางหนึ่ง

(Either-way Communication)

จากรูปที่ 2.3 จะเห็นได้ว่า ถ้าอุปกรณ์ปลายทางทั้งสองด้านสามารถส่งหรือรับข้อมูลได้แล้ว การส่งข้อมูลผ่านช่องสัญญาณเดียวนั้นจะสามารถส่งไปได้ทั้งสองทาง แต่ต้องสลับกัน จะส่งในเวลาเดียวกันไม่ได้ การส่งข้อมูลแบบนี้เรียกว่า "การสื่อสารแบบฮาล์ฟ ดูเพล็กซ์" ตัวอย่างเช่น วิทยุสื่อสารในรถตำรวจ ซึ่งเมื่อเวลาผู้พูดพูดจบ มักจะต่อท้ายด้วยคำว่า "เปลี่ยน" ทั้งนี้เพื่อให้ผู้รับสามารถทราบได้อย่างรวดเร็วว่าข้อมูลที่ส่งมานั้นหมดแล้ว สามารถส่งข้อมูลตอบกลับไปได้ นั่นคือเมื่อผู้รับได้รับข้อมูลแล้ว ผู้รับจะใช้ระยะเวลาหนึ่งในการตีความและทราบว่าข้อมูลจากผู้ส่งหมดแล้ว และพร้อมที่จะตอบกลับไป ซึ่งช่วงเวลานี้เรียกว่ารีแอคชั่นไทม์ (reaction time) และเมื่อผู้รับต้องการส่งข้อมูลตอบกลับไป จะมีการกดสวิตซ์ ซึ่งต้องใช้เวลาในการที่จะเปลี่ยนสภาวะจากผู้รับเป็นผู้ส่ง ช่วงเวลาที่กดสวิตซ์นี้เรียกว่าไลน์เทอร์นอะราวน์ไทม์ (line turnaround time) ดังนั้นจะเห็นได้ว่า ระบบฮาล์ฟ ดูเพล็กซ์นั้น การเปลี่ยนแปลงการส่งข้อมูลการส่งด้านหนึ่งจะต้องเสียเวลาไป ซึ่งเวลานี้

ประกอบด้วย รีแอคชั่นไทม์ และ ไลน์เทอร์นอะราวน์ไทม์ รวมกันเรียกว่า ซิสเต็มเทอร์นอะราวน์ไทม์ (system turnaround time)



รูปที่ 2.3 แสดงระบบการสื่อสารแบบฮาล์ฟดูเพล็กซ์

การสื่อสารแบบฟูลดูเพล็กซ์ หรือการสื่อสารแบบสองทาง
(Both-Way Communication)

ในรูป 2.4 จะเห็นได้ว่า หากตัวกลางมีช่องสัญญาณ 2 ช่อง และอุปกรณ์ปลายทางสามารถส่งและรับข้อมูลได้พร้อมๆกันแล้ว การส่งข้อมูลจะสามารถส่งได้ทั้งสองทิศทางพร้อมกัน การติดต่อส่งข้อมูลแบบนี้เรียกว่า "ฟูลดูเพล็กซ์" หรือ "การสื่อสารแบบสองทาง" ตัวอย่างเช่น ระบบการจราจรสองทาง ระบบโทรศัพท์ทั่วๆไป ซึ่งการส่งข้อมูลทั้งสองทางนี้ ไม่ต้องเสียเวลาในการสับสวิตซ์ไปมา นั่นคือไม่ต้องเสียไลน์เทอร์นอะราวน์ไทม์ ซึ่งมีในระบบฮาล์ฟดูเพล็กซ์ แต่ยังต้องเสียรีแอคชั่นไทม์

ในการที่ผู้ใช้ปลายทางด้านหนึ่งสังเกตว่าอีกปลายด้านหนึ่งส่งข้อมูลมาหมดแล้ว และพร้อมที่จะตอบกลับข้อมูลที่ได้รับนั้น ดังนั้นในระบบฟูลดูเพล็กซ์นี้ ซิสเต็มเทอร์นอะราวน์ไทม์ จะสั้นกว่าระบบฮาล์ฟดูเพล็กซ์ นอกจากนั้นจะสังเกตได้ว่า ระบบไฟฟ้าของฟูลดูเพล็กซ์จะใช้สายไฟ 4 เส้น (Four-wire lines) โดยช่องสัญญาณได้มาจากแต่ละคู่สาย (แต่อย่างไรก็ตาม หลักการของฟูลดูเพล็กซ์นั้น ตัวกลางจะต้องมีช่องสัญญาณ 2 ช่อง ดังนั้นในบางกรณี สายไฟ 2 เส้นอาจจะถูกแยกเป็น 2 ช่องสัญญาณ ทำให้สามารถส่งสัญญาณข้อมูลแบบฟูลดูเพล็กซ์ได้)

ข้อสังเกตอีกอย่างหนึ่งของ ฟูลดูเพล็กซ์ก็คือ ถึงแม้ว่าตัวกลางตลอดจนอุปกรณ์ปลายทางทั้งสองข้างมีความสามารถในการรับส่งพร้อมกันได้ แต่ในการใช้งานจริงนั้น ผู้ใช้ปลายทางอาจส่งข้อมูลมาแบบฮาล์ฟดูเพล็กซ์ ซึ่งอาจเนื่องมาจากการทำงานของโอเปอเรเตอร์ (operator) ซึ่งปกติแล้วจะไม่สามารถแปลความหมายของข่าวสารที่เข้ามาและส่งข่าวออกไปในเวลาเดียวกันได้ แต่จะทำงานแค่อย่างใดอย่างหนึ่ง เช่นเดียวกับโปรแกรมประยุกต์ที่ทำงานบนคอมพิวเตอร์ ที่อาจจะเป็นตัวกำหนดให้การใช้งานของระบบการสื่อสารเป็นแบบฮาล์ฟดูเพล็กซ์ ซึ่งโปรแกรมประยุกต์หล่าวนี้จะรอข้อมูลที่ส่งมาจากปลายทางอีกด้านหนึ่ง แล้วจึงประมวลผลข้อมูลและตอบกลับต่อข้อมูลนั้น ซึ่งในการตอบกลับนี้จะสามารถทำได้เร็วกว่าการใช้ระบบสื่อสารแบบฮาล์ฟดูเพล็กซ์ เนื่องจากไม่ต้องเสีย ไลน์เทอร์นอะราวน์ไทม์



รูปที่ 2.4 แสดงระบบการสื่อสารแบบฟูลดูเพล็กซ์

นอกจากนี้ในบางระบบ ช่องสัญญาณหนึ่งของระบบฟูลดูเพล็กซ์จะถูกใช้ในการส่งตัวอักษรที่ถูกพิมพ์ (key) จากเทอร์มินัล (terminal) ไปสู่คอมพิวเตอร์ ส่วนอีกช่องสัญญาณหนึ่งถูกใช้ในการสะท้อน (echoback) ตัวอักษรจากคอมพิวเตอร์กลับไปแสดงบนหน้าจอของเทอร์มินัล ซึ่งระบบนี้จะช่วยตรวจสอบความถูกต้องของการส่งข้อมูลได้ โดยพิจารณาว่าข้อมูลที่ถูกพิมพ์นั้น ถูกต้องตรงกับข้อมูลที่ได้ปรากฏขึ้นบนหน้าจอนั้นหรือไม่อย่างไร

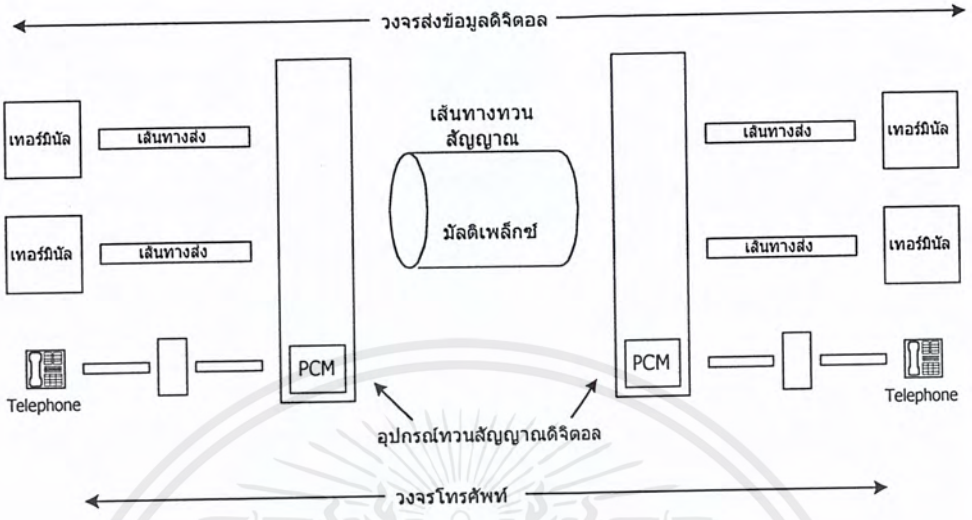
2.2 การพัฒนาการส่งข้อมูลแบบดิจิตอล

การส่งข้อมูลในอดีตได้มีการใช้ประโยชน์จากวงจรอนาลอกของเครื่อข่ายโทรศัพท์ที่ติดตั้งแล้วทั่วประเทศอย่างมีประสิทธิภาพ แต่อย่างไรก็ตาม วงจรส่งแบบอนาลอกที่มีอยู่ได้ถูกออกแบบเพื่อใช้สัญญาณอนาลอกที่เป็นสัญญาณเสียง ดังนั้น การส่งสัญญาณดิจิตอลที่มีค่า 2 ค่า คือ 0 กับ 1 จะรับเอาความเพี้ยนแต่ละชนิดเข้าไปด้วย ทำให้การส่งด้วยความเร็วสูงทำได้ยาก โดยทั่วไปการใช้วงจรส่งอนาลอกต่อโทรศัพท์ 1 เส้น ความเร็วในการส่งมีค่าประมาณ 4 กิโลบิตต่อวินาที และถึงแม้ว่าจะใช้เทคนิคพิเศษ ความเร็วในการส่งก็จะถูกจำกัดอยู่ที่ประมาณหลายสิบกิโลบิทต่อวินาทีเท่านั้น

นอกจากนี้ การส่งข้อมูลแบบดิจิตอลนี้ยังสามารถขจัดความเพี้ยนแต่ละชนิด ที่เกิดจากการส่งข้อมูลแบบอนาลอกได้ คุณภาพเกี่ยวกับการส่งข้อมูลจึงเพิ่มขึ้นมาก

สาเหตุที่ในอดีตไม่ค่อยมีการปฏิบัติการส่งข้อมูลแบบดิจิตอล เนื่องจากเทคนิคทางดิจิตอลยังมีไม่เพียงพอ จึงยังไม่มีการวางวงจรส่งดิจิตอลทั่วประเทศ แต่ในปัจจุบัน เทคโนโลยีแต่ละชนิดได้พัฒนาขึ้นอย่างรวดเร็ว เนื่องจากความต้องการในการส่งข้อมูลและการสื่อสารทางโทรสารมีสูงมาก และความก้าวหน้าในการนำวงจรส่งดิจิตอลเข้ามาใช้นั้น ได้เป็นไปอย่างรวดเร็ว ซึ่งส่งผลทำให้วิธีส่งข้อมูลในอนาคตจะมุ่งสู่ระบบการส่งข้อมูลแบบดิจิตอล

2.3 โครงสร้างของวงจรส่งข้อมูลแบบดิจิตอล



รูปที่ 2.5 โครงสร้างพื้นฐานของวงจรส่งข้อมูลดิจิตอล

วงจรข้อมูลดิจิตอลจะมีโครงสร้างที่เหมือนกันกับวงจรส่งข้อมูลอนาลอก ซึ่งจะประกอบด้วยเส้นทางส่ง ซึ่งอยู่ระหว่างเทอร์มินัลกับอุปกรณ์ทวนสัญญาณ และมีเส้นทางทวนสัญญาณซึ่งจะอยู่ระหว่างตัวอุปกรณ์ทวนสัญญาณ

ในกรณีส่งข้อมูลแบบดิจิตอล ไม่จำเป็นต้องมีอุปกรณ์โมเด็มเพื่อใช้แปลงสัญญาณดิจิตอลกับอนาลอก โดยสัญญาณดิจิตอลจากเทอร์มินัลจะถูกแปลงให้มีรูปคลื่นที่เหมาะสมสำหรับการส่งแบบดิจิตอล และปลายเทอร์มินัลทั้งสองเช่นเดียวกับโมเด็ม เรียกอุปกรณ์นี้ว่า ดิจิตอลเซอร์วิสยูนิต (DSU:Digital Service Unit) ซึ่งอยู่ติดกับปลายเทอร์มินัลทั้งสองเช่นเดียวกับโมเด็ม กล่าวคือ จะติดตั้งตามบ้านพักอาศัย โดยที่อุปกรณ์ DSU นี้มีอยู่หลายแบบ ขึ้นอยู่กับความเร็วในการส่งข้อมูล การเชื่อมต่อเทอร์มินัลและโครงสร้างวงจร

อุปกรณ์ DSU ของวงจรส่งดิจิตอลมีความเร็วสูงหลายขนาด ได้แก่ 64 กิโลบิทต่อวินาที ไปจนถึง 3.072 เมกกะบิทต่อวินาที เช่นเดียวกับการส่งข้อมูลอนาลอก เมื่อต้องการใช้วงจรส่งดิจิตอลให้สามารถส่งข้อมูลได้ในระยะทางไกลๆ จำเป็นต้องใช้อุปกรณ์ทวนสัญญาณดิจิตอล อุปกรณ์เหล่านี้ ได้แก่ อุปกรณ์ PCM มัลติเพล็กซ์และอุปกรณ์ดิจิตอลมัลติเพล็กซ์ โดยที่อุปกรณ์ PCM มัลติเพล็กซ์จะแปลงสัญญาณอนาลอก (สัญญาณเสียง)

จากเส้นทางส่งให้เป็นสัญญาณดิจิตอลเป็นรหัสด้วยวิธี PCM แล้วมัลติเพล็กซ์ อุปกรณ์นี้จะใช้สำหรับการส่งอุปกรณ์ทวนสัญญาณในวงจรส่งข้อมูลแบบอนาลอก ดังนั้นการส่งข้อมูลดิจิตอลโดยทั่วๆไปจะไม่ใช้วิธีนี้

อุปกรณ์ดิจิตอลมัลติเพล็กซ์ จะมีโครงสร้างของดิจิตอลไฮเออราคี (Digital Hierachy) เป็นอุปกรณ์ที่ใช้สำหรับมัลติเพล็กซ์สัญญาณดิจิตอล ส่วนหนึ่งของอุปกรณ์จะใช้ในการส่งข้อมูลดิจิตอลเฉพาะด้าน ดังจะกล่าวต่อไป นอกจากนี้ เส้นทางการส่งระหว่างเทอร์มินัลกับอุปกรณ์ทวนสัญญาณบาลานซ์แพร์เคเบิลเป็นแบบ 4 สาย ส่วนทางเส้นทางทวนสัญญาณจะใช้ตัวอย่างการส่ง ได้แก่ โคแอคเชียลเคเบิล .เส้นใยแสง.ไมโครเวฟ เป็นต้น

ในการส่งข้อมูลดิจิตอล มีหลักการส่งโดยอาศัยจังหวะไทม์มิ่ง หรือ สัญญาณนาฬิกา ร่วมกันในระบบการส่ง (เครือข่าย) โดยที่แต่ละสถานีทวนสัญญาณจะจัดเตรียมดิจิตอลสัญญาณนาฬิกาซัพพลาย (Digital Clock Supply) ซึ่งจะสร้างสัญญาณนาฬิกา ให้กับเครือข่าย เพื่อให้การทำงานของอุปกรณ์ DSU และอุปกรณ์มัลติเพล็กซ์ที่ได้กล่าวมาแล้วเป็นไปตามจังหวะ

ความเร็วในการส่งข้อมูล	เทอร์มินัลอินเตอร์เฟซ	โครงสร้างวงจร		วิธีการสื่อสาร	วิธีการซิงโครนัส
ต่ำกว่า 200 bit/s	V/X	เซอร์กิตสวิทชิ่ง	4W	ฟูลดูเพล็กซ์	อะซิงโครนัส (สตาร์ทสตอป)
ต่ำกว่า 300 bit/s	“	วงจรต่อตรง/เซอร์กิตสวิทชิ่ง	“	“	“
ต่ำกว่า 1,200 bit/s	“	“	“	“	“
2,400 bit/s	“	วงจรต่อตรง/เซอร์กิตสวิทชิ่ง /แพคเกตสวิทชิ่ง	“	“	ซิงโครนัส
4,800 bit/s	“	“	“	“	“
9,600 bit/s	“	“	“	“	“

ตารางที่ 2.1 ชนิดของอุปกรณ์ปลายทางวงจรข้อมูล

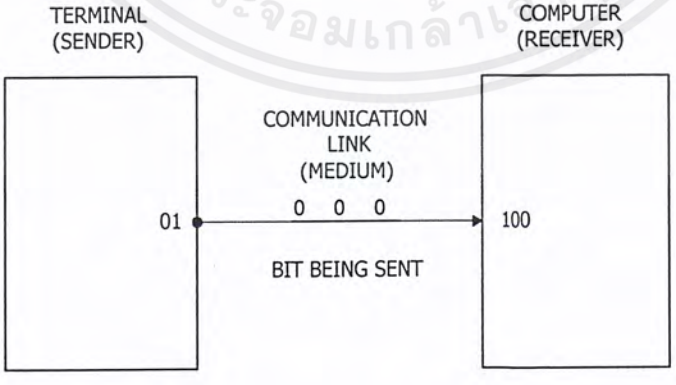
หลังจากเปลี่ยนสัญญาณดิจิตอลที่ส่งจากเทอร์มินัล A เป็นสัญญาณที่เหมาะสมสำหรับการส่งแบบดิจิตอลในอุปกรณ์ DSU แล้ว จะส่งต่อไปยังเส้นทางส่งด้วยวิธีส่งข้อมูลแบบดิจิตอล สำหรับสถานีทวนสัญญาณซึ่งรับสัญญาณนี้ หลังจากการมัลติเพล็กซ์สัญญาณข้อมูลจากหลายวงจรไปจนถึงหลายสิบวงจร โดยอุปกรณ์มัลติเพล็กซ์ของการส่งสัญญาณข้อมูลดิจิตอลครั้งหนึ่งแล้ว ยังต้องมัลติ-เพล็กซ์รวมกับสัญญาณดิจิตอลอื่นๆ ได้แก่ วงจรโทรศัพท์ สัญญาณดิจิตอลที่ถูกส่งโดยวิธีแบบดิจิตอลมัลติเพล็กซ์ในเส้นทางส่งทวนสัญญาณ โดยจะส่งผ่านสถานีทวนสัญญาณ จากนั้นจะถูกรับโดย DSU ด้านรับและส่งต่อไปยังเทอร์มินัล B

2.4 เทคนิคการส่งผ่านข้อมูล

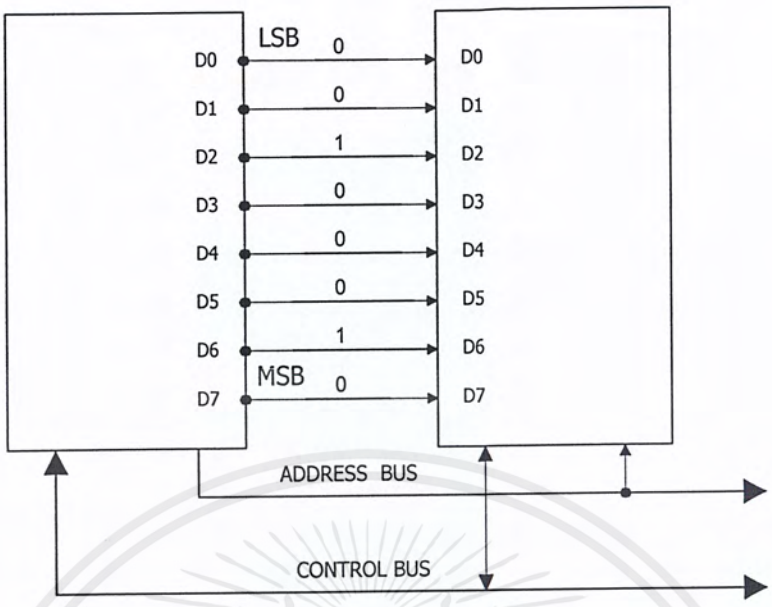
ในการส่งผ่านข้อมูล เราสามารถแบ่งลักษณะของการส่งผ่านข้อมูลได้เป็น 2 แบบ นั่นคือ การส่งผ่านข้อมูลแบบขนานและการส่งผ่านข้อมูลแบบอนุกรม โดยที่การส่งผ่านข้อมูลโดยใช้สายคู่เดียว คือการส่งแบบอนุกรม และการส่งข้อมูลโดยใช้สายส่งหลายคู่ จะเป็นการส่งข้อมูลแบบขนาน

โดยทั่วไปแล้วในกรณีส่งข่าวสารซึ่งแสดงโดยกลุ่มรหัส 0 กับ 1 นั้น ถ้าส่งโดยรวมรหัสนั้นทั้งหมดแล้วส่งในเวลาเดียวกัน เวลาที่ใช้ก็จะสั้นที่สุด คือ ถ้าต้องการส่ง A จะต้องแทนด้วยรหัส JIS 7 ที่ใช้ในการตรวจสอบข้อผิดพลาดเพิ่มเข้าไปจะได้กลุ่มรหัส 01000001 ซึ่งถ้ากลุ่มรหัสนี้ส่งพร้อมๆ กัน ในเส้นทางที่เท่ากัน 8 เส้น และส่งไปในเวลาเดียวกัน เรียกว่า การส่งแบบขนาน ถ้าหากระยะทางยาวขึ้นเท่าใด ก็จะทำให้ค่าใช้จ่ายของเส้นทางเพิ่มสูงขึ้นเท่านั้น ดังนั้นส่วนมากมักจะใช้วิธีการส่งเช่นนี้ในอุปกรณ์เดียวกัน หรือภายในอาคารเดียวกันที่มีระยะการส่งสั้นๆ

การส่งข้อมูลระยะไกลๆ ดังแสดงในรูปที่ 2.7 รหัสใช้แทนอักษร A จะถูกส่งเข้าไปในเส้นทางเดียวกันทีละบิท และทางรับก็จะรวบรวมแต่ละบิทที่ถูกส่งเข้ามาตามลำดับ เมื่อรวบรวมรหัสทั้งหมดแล้วก็เข้าใจว่าข่าวสารเป็น A เรียกว่า การส่งแบบอนุกรม



รูปที่ 2.6 หลักการส่งแบบอนุกรม



รูปที่ 2.7 หลักการส่งแบบขนาน

การส่งผ่านข้อมูลแบบขนานจะเร็วกว่าการส่งแบบอนุกรม เพราะทุกบิทถูกส่งออกไปพร้อมกัน ดังนั้น ในการส่งผ่านข้อมูลแบบขนานจะถูกนำมาใช้ในการทำงานภายในคอมพิวเตอร์ แต่การใช้สายหลายๆเส้นจะทำให้เกิดความสับสน ระบบมีราคาสูง และรับสัญญาณรบกวนได้ง่าย ถ้านำการส่งผ่านข้อมูลแบบขนานมาใช้กับการส่งข้อมูลที่มีระยะทางไกล ดังนั้นจึงนิยมใช้วิธีการส่งแบบอนุกรม แต่ว่าการส่งแบบนี้จะเป็นการส่งตัวอักษรเรียงต่อๆกันไป จึงจำเป็นต้องบอกด้านรับให้รู้ถึงช่วงแบ่งแยกระหว่างแต่ละตัวอักษร ซึ่งวิธีการดังกล่าวนี้จะได้กล่าวใน วิธีการซิงโครไนเซชั่น (Synchronization)

2.5 รูปแบบการส่งข้อมูลดิจิตอล

วิธีส่งข้อมูลแบบดิจิตอลสามารถแบ่งได้ใหญ่ๆเป็น 2 ชนิด ตามการกำหนดจังหวะของเทอร์มินัลกับอุปกรณ์ส่งดิจิตอล คือ

1. การซิงโครนัส (Synchronous)
2. การอะซิงโครนัส (Asynchronous)

ในการส่งข้อมูลแบบซิงโครนัส เทอร์มินัลจะทำงานตามจังหวะสัญญาณนาฬิกาของอุปกรณ์ส่งส่วนสัญญาณข้อมูล จะถูกส่งออกไปตามช่วงเวลาสัญญาณนาฬิกาของเครือข่าย ลักษณะเช่นนี้ การส่งข้อมูลที่ความเร็วขนาดปานกลางถึงความเร็วสูง จะมีประสิทธิภาพสูงสุด และเทอร์มินัลแบบซิงโครนัสก็จะใช้วิธีการส่งแบบนี้

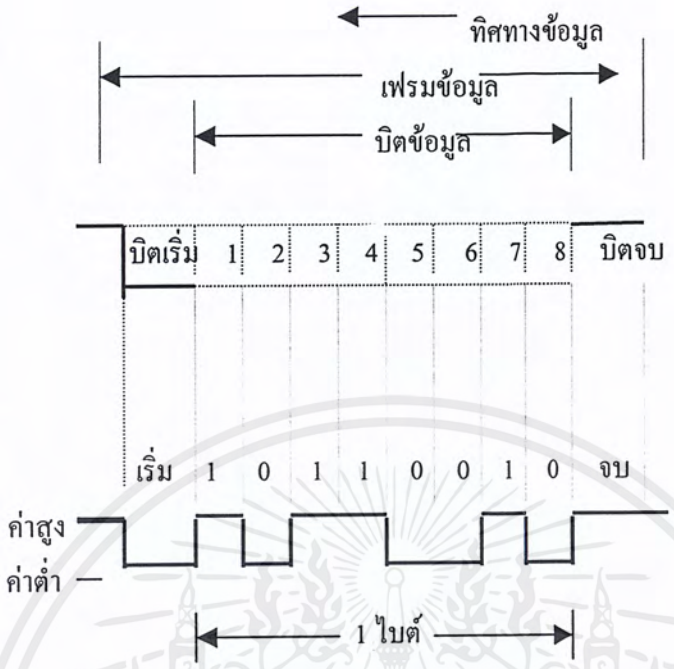
ถึงแม้ว่าเป็นการส่งข้อมูลแบบซิงโครนัส แต่เนื่องจากการทำงานรับส่งที่เทอร์มินัลไม่มีการเข้าจังหวะกับสัญญาณนาฬิกาในเครือข่าย จึงมีความจำเป็นที่จะต้องใช้วิธีที่เหมาะสมในการแปลงสัญญาณข้อมูลให้เป็นดิจิตอล หรือการเข้ารหัส ซึ่งมีจังหวะตามสัญญาณนาฬิกาในเครือข่าย เมื่อเปรียบเทียบประสิทธิภาพในการส่งกับวิธีซิงโครนัสแล้ว ค่อนข้างไม่ดี แต่การส่งข้อมูลความเร็วต่ำของเทอร์มินัลแบบสตาร์ท-สตอป (อะซิงโครนัส) จะเป็นวิธีที่มีประสิทธิภาพ

ต่อไปนี้จะกล่าวถึงการส่งข้อมูลแบบซิงโครนัสและอะซิงโครนัส

2.5.1 การส่งข้อมูลแบบอะซิงโครนัส หรือ การส่งข้อมูลแบบ Start-Stop

เทคนิคนี้ถูกนำมาใช้ในการแก้ปัญหาของการส่งข้อมูลโทรเลขและเทเล็กซ์ (telex) เป็นวิธีที่ถูกเสนอโดย Baudot ในปี พ.ศ. 2417 โดยที่ในการส่งข้อมูลแต่ะตัวอักษรนั้นจะมีการส่งบิตเริ่มต้น (start bit) ไป 1 บิต ก่อนส่งบิตของข้อมูลตัวอักษร และตามท้ายข้อมูลตัวอักษรด้วยบิตจบ (Stop Bit) 1, 1.5 หรือ 2 บิต ดังตัวอย่างแสดงในรูป กล่าวคือ ในขณะที่ไม่มีกาส่งตัวอักษร สภาวะของสายจะมีค่าตรรกะเป็น 1 เมื่อมีการส่งข้อมูล ชิป UART จะทำให้สายมีสภาวะเริ่มต้นเป็น 1 อีก 1.5 หรือ 2 บิตไทม์ ซึ่งเป็นการส่งบิตจบ หลังจากนั้นจึงจะส่งบิตเริ่มต้นของข้อมูลตัวอักษรต่อไป

ในกรณีที่ไม่มีข้อมูลส่งต่อไป สภาวะของสายจะเป็น 1 ไปจนกว่าจะมีการส่งข้อมูลอีก การส่งแบบนี้จะส่งได้ครั้งละ 1 ตัวอักษร โดยที่ช่วงเวลาระหว่างตัวอักษรจะนานเท่าใดก็ได้ สำหรับชิป UART ของฝั่งรับ เมื่อตรวจสอบได้ว่ามีการเปลี่ยนสภาวะของสายจาก 1 เป็น 0 ก็จะเริ่มสัญญาณเวลาเพื่อทำการอ่านสัญญาณบิตข้อมูล (โดยปกติแล้วจะใช้ 5 หรือ 7 ตัวอย่าง ในการตรวจสอบ) ดังนั้นหากอ่านข้อมูลแล้วตัดสินได้ว่าบิตที่มีการเปลี่ยนสภาวะนั้นมีค่าเป็น 1 ก็แสดงว่าสภาวะของสายเปลี่ยนแปลงนั้นเกิดจากสัญญาณรบกวน จึงต้องตรวจสอบหาบิตเริ่มต้นทีจะถูกส่งมาต่อไป แต่ในทางตรงข้าม หากตัดสินได้ว่าบิตนั้นเป็น 0 แสดงว่ามีการส่งบิตเริ่มต้นมา ชิป UART จะจะทำการอ่านบิตข้อมูลของตัวอักษรและบิตจบซึ่งตามท้ายมา ในกรณีที่บิตจบยังมีค่าเป็น 0 แสดงว่าเกิดความผิดพลาดในการรับข้อมูล



รูป 2.8 แสดงการส่งข้อมูลแบบอะซิงโครนัส

สำหรับวิธีนี้ ไลน์อะแคปเตอร์ฝั่งรับจะต้องถูกตั้งให้อ่านข้อมูลด้วยอัตราเดียวกับการส่งข้อมูลของฝั่งส่ง นั่นคือสัญญาณเวลาจะเท่ากัน นอกจานั้นจำนวนของบิตจบและค่าพาริตี้ (parity bit) ต้องถูกตั้งให้ตรงกับฝั่งตรงข้ามด้วย แต่อย่างไรก็ตาม เนื่องจากเป็นสัญญาณเวลาของเครื่องต่างกัน ดังนั้นอาจจะเป็นไปได้ที่สัญญาณเวลาของฝั่งรับไม่สอดคล้อง (synchronized) ตรงกับฝั่งส่งเลยทีเดียว แต่ปัญหานี้ไม่สำคัญสำหรับการส่งแบบอะซิงโครนัส เนื่องจากความแตกต่างเล็กน้อยของสัญญาณเวลาทั้งสองฝั่งจะไม่ทำให้เกิดความผิด-พลาดของการรับส่งข้อมูลสำหรับช่วงเวลาของข้อมูลมากที่สุดเพียง 8+3 บิตนี้ นอกจากนั้นเนื่องจากสัญญาณเวลาของฝั่งรับจะเริ่ม ณ จุดเริ่มต้นของบิตเริ่ม ดังนั้นเฟส (phase) ของสัญญาณเวลาทั้งสองจะตรงกันด้วย

การส่งข้อมูลแบบอะซิงโครนัสนี้เป็นวิธีการง่ายๆ เหมาะสำหรับเทอร์มินัลซึ่งส่งข้อมูลตัวอักษร (text) เทอร์มินัลแบบนี้ไม่จำเป็นต้องมีบัฟเฟอร์ขนาดใหญ่ และชิป UART ยังมีราคาถูก เช่น เทเลพรินเตอร์ (teleprinter) , VDU หรือไมโครคอมพิวเตอร์ เป็นต้น ในทางกลับกัน ประสิทธิภาพของการส่งข้อมูลแบบนี้ไม่สูงนัก เพราะในการส่งทุกๆตัวอักษร 8 บิตนั้นจะมีโอเวอร์เฮดอยู่ 2-3 บิต ซึ่งเป็นการสูญเสียความสามารถในการรับส่งข้อมูลของสายไปถึง 20-24% ดังนั้นประสิทธิภาพการส่งข้อมูลแบบนี้จังไม่เกิน 8/(8+2) = 80%

2.5.2 การส่งข้อมูลแบบซิงโครนัส

ดังที่ได้กล่าวมาแล้วว่าการส่งข้อมูลแบบอะซิงโครนัสนั้น มีประสิทธิภาพการส่งข้อมูลไม่เกิน 80% ดังนั้นในศตวรรษที่ 1960 IBM จึงได้เสนอเทคนิคการส่งข้อมูลซึ่งมีประสิทธิภาพการส่งสูงและมีความน่าเชื่อถือได้ วิธีนี้นำเอาข้อมูลอักษรมารวมกันเป็นบล็อค สามารถตรวจสอบได้ว่าบล็อคข้อมูลนั้นเริ่มต้นและสิ้นสุดที่ใด นอกจากนี้ยังมีเทรเลอร์ (trailer) ของบล็อค ซึ่งเป็นบิตเพิ่มเกิน (redundant bit) เพื่อใช้ในการตรวจสอบความผิดพลาดของการส่งข้อมูล จึงทำให้การส่งข้อมูลมีความน่าเชื่อถือสูง วิธีนี้เรียกทั่วไปว่า *การส่งข้อมูลแบบซิงโครนัส* (Synchronous Transmission) ซึ่งใช้ส่งข้อมูลได้อย่างมีประสิทธิภาพ ปัจจุบันในการส่งข้อมูลแบบซิงโครนัสนี้อาจแบ่งได้เป็น 2 ชนิด คือ

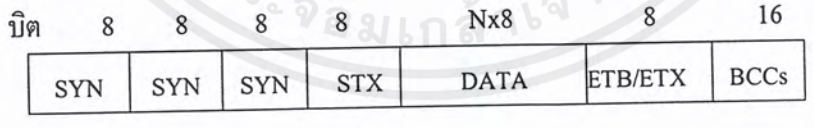
1.คาแรกเตอร์โอเรียนต์ (Character Oriented)

2. บิตโอเรียนต์ (Bit Oriented)

2.5.3 คาแรกเตอร์โอเรียนต์

จากรูป 2.9 แสดงการส่งข้อมูลแบบซิงโครนัสชนิดคาแรคเตอร์โอเรียนต์ ซึ่งเป็นการส่งข้อมูลที่ข้อมูลในบล็อกเป็นไบต์ของตัวอักษร กล่าวคือจำนวนบิตในบล็อกเป็นจำนวนคูณ (multiple) ของจำนวนบิตของตัวอักษร เช่น หาก 1 ตัวอักษรประกอบด้วย 8 บิตแล้ว จำนวนบิตในบล็อกจะเป็น 1024 นอกจากนี้ช่วงเวลาระหว่างปลายของบิตตัวสุดท้ายของอักษรตัวหน้ากับอักษรที่ตามมาจะมีค่าเท่ากับ 0 (กล่าวคือตัวอักษรถูกส่งติดต่อกันมา) หรือเท่ากับผลคูณของเวลาในการส่งอักษรตัวหนึ่ง (กล่าวคือมีช่วงเว้นของข้อมูลตัวอักษรหนึ่งตัวหรือหลายตัวอักษร แล้วจึงมีข้อมูลตัวอักษรส่งตามมา)

การส่งข้อมูลแบบซิงโครนัสชนิดคาแรคเตอร์โอเรียนต์นี้ เหมาะสำหรับการส่งข้อมูลที่เป็นตัวอักษร ตัวอย่างของคาแรคเตอร์โอเรียนต์ เช่น การส่งข้อมูลแบบ Bisync (Binary Synchronous Communication) เป็นต้น



SYN = Synchronous

STX = Start of Text

ETC = End of Transmission Block

ETX = End of Text

BCCs = Block Check Characters

รูป 2.9 แสดงตัวอย่างเฟรมข้อมูลแบบคาแรคเตอร์โอเรียนต์

2.5.4 บิตโอเรียนต์

เนื่องจากการส่งข้อมูลแบบซิงโครนัสชนิดคาแรคเตอร์โอเรียนต์ ไม่ค่อยจะเหมาะสมนักสำหรับการส่งข้อมูลชนิดอื่นๆที่ไม่เป็นตัวอักษร เช่น ข้อมูลเสียง.ข้อมูลภาพ ตลอดจนไบนารีดาต้า (Binary Data) ดังนั้นจึงได้มีการคิดค้นวิธีการส่งข้อมูลแบบซิงโครนัสชนิดบิตโอเรียนต์ขึ้น ซึ่งเป็นการส่งข้อมูลโดยที่จำนวนบิตในบล็อกจะมีค่าใดค่าหนึ่งก็ได้ และไม่จำเป็นต้องเป็นจำนวนคูณของจำนวนบิตของตัวอักษรดังแสดงในรูป การส่งแบบนี้ข้อมูลอาจจะเป็นตัวอักษร รูปภาพ และข้อมูลเสียงได้ ตัวอย่างของบิตเรียนต์ เช่น การส่งโดยวิธี Synchronous Data Link Control (SDLC) ของเครื่อง IBM หรือ High-level Data Link Control (HDLC) ตามมาตรฐานของ International Standards Organization (ISO) เป็นต้น

แฟล็กซ์	แอดเดรส	คอนโทรล	ข้อมูล	FCS	แฟล็กซ์

แฟล็ก (Flag) : ใช้บ่งบอกจุดเริ่มต้นและจุดสิ้นสุดของเฟรม
คอนโทรล (Control) : ใช้ควบคุมการส่งข้อมูล
แอดเดรส (Address) : ของฝั่งส่งหรือฝั่งรับ
FCS (Frame Check Sequence) : ใช้ตรวจสอบความถูกต้องของเฟรม

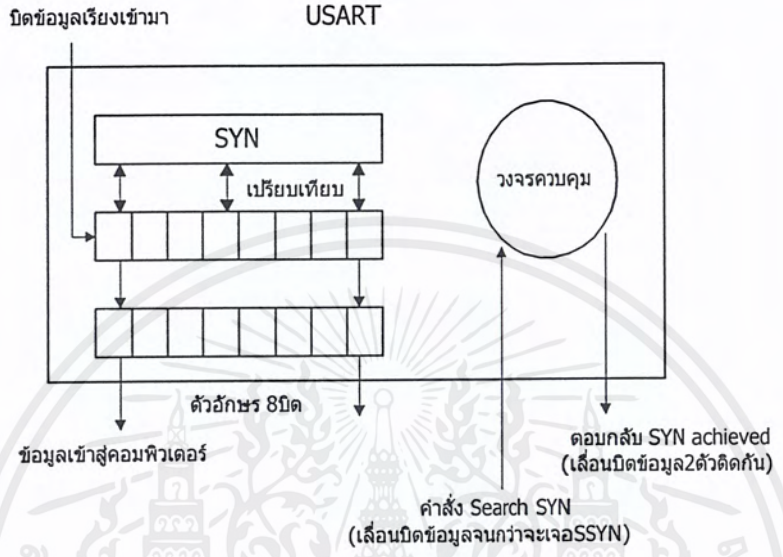
รูป 2.10 แสดงตัวอย่างเฟรมข้อมูลแบบบิตโอเรียนต์

2.5.5 บิตซิงโครนัสและคาแรคเตอร์ซิงโครไนเซชั่นสำหรับการส่ง

การส่งข้อมูลแบบซิงโครนัสก็เช่นเดียวกับแบบอะซิงโครนัส คือฝั่งรับจะสามารถระบุค่า 1 และ 0 ของบิตตามฝั่งส่งได้ถูกต้องอย่างไร และจะระบุว่าเป็นตัวอักษรใดได้ถูกต้องอย่างไร สำหรับปัญหาของบิตซิงโครไนเซชั่นนั้นทำได้ง่าย โดยจะมีการส่งสัญญาณเวลาไปให้ฝั่งรับพร้อมกับข้อมูล ซึ่งหากการส่งเป็นระยะทางสั้นๆ เช่น เทอร์มินัลส่งข้อมูลให้กับโมเด็มนั้น สัญญาณเวลาจะถูกส่งไปตามสายวงจร (circuit) ซึ่งแยกกับสายส่งข้อมูล แต่ถ้าเป็นการส่งระยะไกลแล้ว ตัวแปลงสัญญาณ เช่น โมเด็ม จะใช้สัญญาณเวลาเดียวกันในการส่งบิตข้อมูลออกจากพอร์ตของฝั่งส่ง และการแปลงสัญญาณข้อมูลจากดิจิตอลเป็นอนาลอก ทำให้สัญญาณเวลาที่ใช้ส่งข้อมูล ถูกผสมผสานไปกับสัญญาณเวลาจากฝั่งส่งให้กับฝั่งรับ ทำให้ทั้ง 2 ฝั่งทำงานได้สอดคล้องกัน (synchronism) และเป็นที่มาของคำว่า “synchronous transmission” นั่นเอง

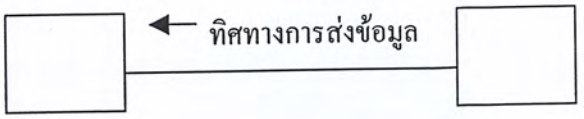
สำหรับปัญหาของคาแรกเตอร์ซิงโครไนเซชั่นนั้น สามารถแก้ไขได้โดยในแต่ละบล็อกข้อมูล จะมีการใส่กลุ่มของบิตที่มีรูปแบบพิเศษ เพื่อใช้ในการซิงโครไนเซชั่น เช่น ตัวอักษร SYN หรือ

แฟล็ก (flag : กลุ่มของบิตค่า 01111110) ปะไปเป็นเฮดเดอร์ของเฟรมข้อมูลที่ส่งไป ชิป USART ของฝั่งรับจะเลื่อนแถวอนุกรมของบิตที่เข้ามาทีละบิตสู่บัฟเฟอร์ และจะเปรียบเทียบชุดของบิต 8 บิตในบัฟเฟอร์กับตัวอักษร SYN ดังแสดงในรูปที่ 2.11



รูปที่ 2.11 แสดงการตรวจหาตัวอักษร SYN ของ USART

ถ้าหากไม่ตรงกัน ก็จะเลื่อนบิตต่อไปเข้ามาอีก 1 บิต แล้วเปรียบเทียบ 8 บิตในบัฟเฟอร์กับตัวอักษร SYN อีก ทำเช่นนี้ไปจนกว่าจะพบชุดของ 8 บิต ที่ตรงกับตัวอักษร SYN ซึ่งแสดงว่าไบต์ของบิตเหล่านั้นอาจจะเป็นตัวอักษร SYN หรือไม่ใช่ก็ได้ ซึ่ง USART จะตรวจสอบต่อไปโดยเลื่อนอีก 8 บิตต่อไปเข้าสู่บัฟเฟอร์เพื่อเปรียบเทียบกับตัวอักษร SYN หากตรงกัน แสดงว่าได้พบจุดเริ่มต้นของบล็อกแล้ว จึงสามารถนำไบต์ถัดไปมาใช้เป็นข้อมูลได้ ส่วนสาเหตุที่ต้องมีการตรวจสอบหาตัวอักษร SYN ถึง 2 ตัว เพื่อหาจุดเริ่มต้นของบล็อกก็เพื่อป้องกันความผิดพลาดของการตรวจสอบ (false synchronization) ซึ่งเกิดเนื่องจากกลุ่มของบิตระหว่างตัวอักษร 2 ตัว มีรูปแบบตรงกับตัวอักษร SYN ดังแสดงในรูปที่ 2.12 อนึ่งการตรวจสอบหาตัวอักษร SYN นี้อาจทำด้วยซอฟต์แวร์ก็ได้ แต่จะทำงานได้ช้ากว่าชิป USART



C	b	a	z	x	x
	10000110	01000110			

ก.) ตรวจหา SYN ผิดพลาด

SYN	SYN	c	b	a	y	y

ข.) ตรวจเจอ SYN 2 ตัวติดกันแสดงว่าถูกต้อง

รูปที่ 2.12 แสดงการป้องกันการตรวจหาตัวอักษร SYN ผิดพลาด

2.5.6 ประสิทธิภาพในการส่งข้อมูล

การส่งข้อมูลแบบซิงโครนัสมีประสิทธิภาพสูงกว่าแบบอะซิงโครนัสมาก และทำให้มีการใช้ความสามารถของสายส่งเกือบทั้งหมด ตัวอย่างเช่น หากมีการส่งบล็อกของข้อมูลต่อเนื่องกันตลอด โดยที่บล็อกประกอบด้วย 3 ตัวอักษร SYN , 4 ตัวอักษรสำหรับ STX , ETX และ BCC แล้วหากข้อมูลในบล็อกมีจำนวน 240 ตัว จะมีประสิทธิภาพการส่งถึง (240*8) / (247*8) = 97%

2.5.7 ซิงโครนัสและอะซิงโครนัส

เทอร์มินัลที่ส่งข้อมูลแบบซิงโครนัส เป็นเทอร์มินัลที่ทำงานซับซ้อนกว่าแบบอะซิงโครนัสมาก เพราะจะต้องสามารถสร้างและถอดเฟรมของข้อมูลได้ และต้องมีบัฟเฟอร์โตพอที่จะเก็บบล็อกของข้อมูลไว้ก่อนที่จะส่งออกพร้อมๆกันทีเดียว ต่างกับเทอร์มินับแบบอะซิงโครนัสซึ่งส่งข้อมูลได้ทีละตัวอักษรตามอัตราการพิมพ์ของโอเปอเรเตอร์ ดังนั้นเทอร์มินัลแบบซิงโครนัสจึงมีราคาแพงกว่าเทอร์มินัลแบบอะซิงโครนัส แต่อย่างไรก็ตามเมื่อเทคโนโลยีพัฒนาขึ้นเรื่อยๆ ราคาที่ใช้ในการสร้างเทอร์มินัลแบบซิงโครนัสก็ลดต่ำลงมาใกล้เคียงกับเทอร์มินัลแบบอะซิงโครนัส

นอกจากนี้ ข้อดีอีกอย่างหนึ่งของเทอร์มินัลแบบซิงโครนัสคือ ข้อมูลในบล็อกจะต่อเนื่องกัน ดังนั้นหากฝั่งรับตรวจพบว่ามีการขาดสัญญาณข้อมูลในบล็อก ก็แสดงว่าเกิดปัญหาในการส่งข้อมูล

สำนักหอสมุดกลาง พระจอมเกล้าลาดกระบัง

แต่สำหรับอะซิงโครนัสนั้น เนื่องจากข้อมูลอักษรอาจไม่ต่อเนื่องกัน ผู้รับจึงไม่ทราบว่าเกิดปัญหาในการส่งข้อมูลมาหรือไม่อย่างไร

ดังนั้น แนวโน้มในการส่งข้อมูลจะเป็นแบบซิงโครนัสมากกว่า เพราะเป็นการใช้สายอย่างมีประสิทธิภาพกว่า แต่อย่างไรก็ตามยังมีการส่งข้อมูลแบบอะซิงโครนัสอยู่มาก เพราะว่าเทอร์มินัลในปัจจุบันนี้ส่วนใหญ่เป็นแบบอะซิงโครนัสซึ่งมีราคาถูก นอกจากนั้นเทอร์มินัลแบบอะซิงโครนัสยังจะมีความนิยมเพิ่มมากขึ้น เมื่อมีการใช้ Statistical Multiplexers (STDM) เพื่อรวบรวมข้อมูลของเทอร์มินัลแบบอะซิงโครนัสหลายๆตัว และส่งข้อมูลผ่านสายส่งแบบซิงโครนัสด้วยอัตราการส่งสูง เป็นการเพิ่มประสิทธิภาพการใช้สายได้มาก และมีความถูกต้องของข้อมูลสูง

2.6 วิธีการส่งแบบดิจิตอลมัลติเพล็กซ์

2.6.1 บทสังเขปของการส่งแบบดิจิตอลมัลติเพล็กซ์

การส่งแบบดิจิตอลมีลักษณะเช่นเดียวกับแบบอนาลอก คือเมื่อมีการส่งในระบบทางไกล จะใช้การมัลติเพล็กซ์สัญญาณ และมีการกำหนดโครงสร้างดิจิตอลไฮเออราคี ซึ่งมีเป้าหมายที่จะทำการมัลติเพล็กซ์สัญญาณดิจิตอลอย่างมีประสิทธิภาพ และมีดีกรีมัลติเพล็กซ์สูงสุดเท่าที่จะเป็นไปได้

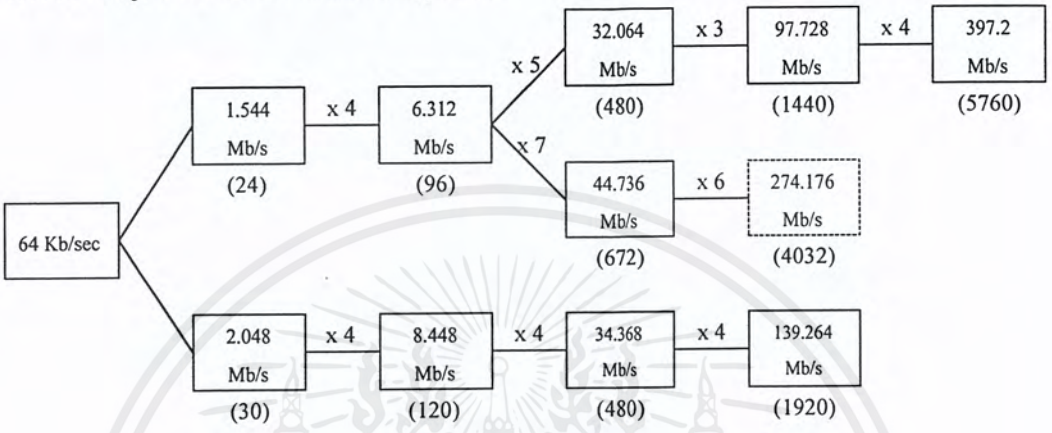
สำหรับดิจิตอลไฮเออราคี จะตัดสินจากความเร็วของสัญญาณดิจิตอลในแต่ละชนิดของการสื่อสารกับความเร็วที่ส่งได้ในตัวกลางแต่ละชนิด สำหรับระบบเครือข่ายไฟฟ้าสื่อสารในประเทศญี่ปุ่น ได้กำหนดไว้ดังในตารางที่ 2.2

ลำดับชั้น	ความเร็วในการส่ง	วงจรโทรศัพท์ (จำนวนแชนแนล)
แบเรอกรุ๊ป	3.2 kbit/s 6.4 kbit/s 12.8 kbit/s	-
กรุ๊ปที่ 0	64 kbit/s	1 วงจร
กรุ๊ปที่ 1	1.544 Mbit/s	24 วงจร
กรุ๊ปที่ 2	6.312 Mbit/s	96 วงจร
กรุ๊ปที่ 3	32.064 Mbit/s	480 วงจร
กรุ๊ปที่ 4	97.728 Mbit/s	1,440 วงจร
กรุ๊ปที่ 5	397.200 Mbit/s	5,760 วงจร

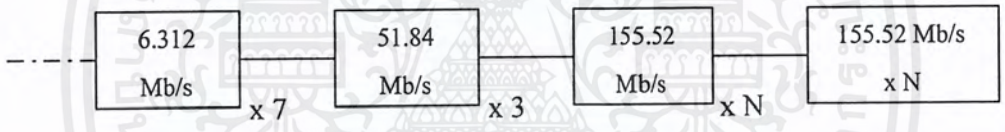
ตาราง 2.2 ดิจิตอลไฮเออราคี

42156

นอกจากมาตรฐานสากลที่เกี่ยวข้องกับดิจิตอลไฮเออราคีแล้ว ยังมีระบบญี่ปุ่นซึ่งใช้วิธี 1.544 เมกะบิทต่อวินาที ซึ่งจะมี 24 ช่องสัญญาณ ส่วนในระบบยุโรปใช้วิธี 2.048 เมกะบิทต่อวินาที มี 30 ช่องสัญญาณ ดังนั้นการรวมเข้าเป็นหนึ่งเดียวกัน จึงทำได้ค่อนข้างยาก ในปัจจุบันจึงมีระบบดิจิตอลไฮเออราคีอยู่เพียง 3 ระดับ ดังแสดงตามรูป2.13



รูปที่ 2.13 โครงสร้างแบบดิจิตอลไฮเออราคีของแต่ละประเทศ



รูปที่ 2.14 โครงสร้างแบบดิจิตอลไฮเออราคีที่รวมเป็นหน่วยเดียวในโลก

ในอนาคต การพัฒนาการให้บริการแบบบรอดแบนด์ได้ขยายกว้างไปทั่วโลก การยุบ 3 ระบบให้เหลือเพียงระบบเดียวจึงเป็นเรื่องจำเป็น และเมื่อปี 1988 ระบบซิงโครนัสดิจิตอลไฮเออราคี (SDH : Synchronous Digital Hierarchy) ดังแสดงในรูป 2.10 จึงได้ถูกกำหนดให้เป็นมาตรฐานโดย CCITT และจากนี้ไป จะอธิบายระบบไฮเออราคีในอดีต

ระบบการมัลติแพล็กซ์ของสัญญาณดิจิตอลจะทำตามวิธีการดิจิตอลไฮเออราคี แต่วิธีมัลติ-เพล็กซ์จะใช้การไทม์ดิวิชั่นมัลติเพล็กซ์ (TDM) ซึ่งเหมาะสำหรับสัญญาณดิจิตอล โดยที่วิธีการไทม์ดิวิชั่นมัลติเพล็กซ์มีอยู่ 2 แบบ คือ แบบซิงโครนัส และ สตัฟพัลส์มัลติเพล็กซ์ ซึ่งวิธีการสตัฟมัลติเพล็กซ์มีการใช้บ้างแล้ว โดยมีจุดมุ่งหมายเพื่อส่งข้อมูลดิจิตอลระหว่างอุปกรณ์ทวนสัญญาณในเส้นทางส่งแบบอนาลอก การมัลติเพล็กซ์สัญญาณที่สูงกว่ากรุ๊ปดีกรีก็ใช้วิธีการนี้

อย่างไรก็ตาม วิธีการนี้มีประสิทธิภาพเมื่อใช้ในกรณีที่มีการส่งระหว่างจุดต่อจุด เช่น ระหว่างเส้นทางทวนสัญญาณ แต่ถ้าคิดถึงการพัฒนาการส่งดิจิตอลซึ่งจำเป็นต้องมีชุมสายการสอดใส่และแบ่งแยกสัญญาณแล้ว วิธีการแบบซิงโครนัสจะมีประสิทธิภาพที่ดีกว่า

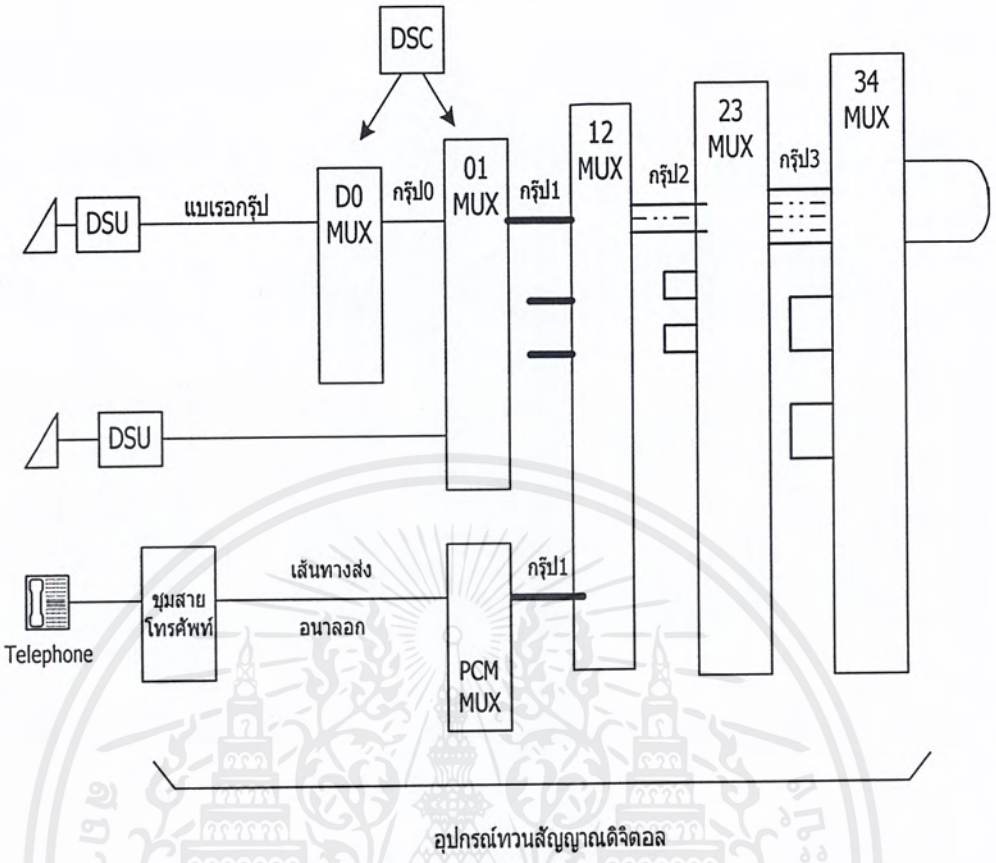
ดังนั้น การมัลติเพล็กซ์สัญญาณดิจิตอลทั้งหมดด้วยวิธีการซิงโครนัส สามารถทำได้ตามอุดมคติ แต่มีปัญหาทางเทคนิคและความประหยัดที่ตามมาในปัจจุบัน วิธีการส่งข้อมูลดิจิตอล ถ้ามีการมัลติเพล็กซ์ต่ำกว่า กรุ๊ปที่ 1 แล้วจะใช้วิธีการส่งแบบซิงโครนัส นอกจากนี้จะใช้วิธีการส่งแบบสตัฟมัลติเพล็กซ์ และวิธีการมัลติเพล็กซ์จะปฏิบัติการโดยขึ้นอยู่กับอุปกรณ์ดิจิตอลมัลติเพล็กซ์ ดังแสดงในตารางที่ 2.3 และหลักการแสดงในรูป 2.15

2.6.2 การมัลติเพล็กซ์สัญญาณแบเรอ/สัญญาณกรุ๊ปที่ 0

สัญญาณแบเรอในรูปร่างของเอนเวลอปที่ส่งมาจากอุปกรณ์ DSU ดังแสดงในรูป จะส่งต่อไปบนเส้นทางส่งและรับ โดยอุปกรณ์แปลงข้อมูลมัลติเพล็กซ์ (DO-MUX:DO-Multiplex) ที่ติดตั้งที่สถานีทวนสัญญาณและสัญญาณแบเรอที่รับเข้ามา โดยที่จังหวะเอนเวลอปจะถูกแปลงให้เป็นสัญญาณยูนิเวอร์เซล 64 กิโลบิทต่อวินาที และจะถูกไทม์มัลติเพล็กซ์ในหน่วยออคเตจที่สัญญาณกรุ๊ป 0 ขนาด 64 กิโลบิทต่อวินาที ด้วยวิธีการซิงโครนัสมัลติเพล็กซ์

อุปกรณ์มัลติเพล็กซ์	ชื่อย่อ	ฟังก์ชั่น	หมายเหตุ
อุปกรณ์มัลติเพล็กซ์ข้อมูล	D 0-MUX	3.2 kbit/s แบเรอ*20 6.4 kbit/s แบเรอ*10 กรุ๊ปที่ 0 12.8 kbit/s แบเรอ*5	สัญญาณแบเรอ 64 k (ตรงกับสัญญาณกรุ๊ปที่ 0) นั้นจะถูกรับโดย 01-MUX
อุปกรณ์มัลติเพล็กซ์กรุ๊ปที่ 0	01-MUX	กรุ๊ปที่ 0*23 กรุ๊ปที่ 1	-
อุปกรณ์มัลติเพล็กซ์กรุ๊ปที่ 1	12-MUX	กรุ๊ปที่ 1*4 กรุ๊ปที่ 2	-
อุปกรณ์มัลติเพล็กซ์กรุ๊ปที่ 2	23-MUX	กรุ๊ปที่ 2*5 กรุ๊ปที่ 3	-
อุปกรณ์มัลติเพล็กซ์กรุ๊ปที่ 3	34-MUX	กรุ๊ปที่ 3*3 กรุ๊ปที่ 4	-
อุปกรณ์มัลติเพล็กซ์กรุ๊ปที่ 4	45-MUX	กรุ๊ปที่ 4*4 กรุ๊ปที่ 5	-

ตารางที่ 2.3 ชนิดของอุปกรณ์ดิจิตอลมัลติเพล็กซ์



รูปที่ 2.15 หลักการมัลติเพล็กซ์สัญญาณดิจิตอล

สัญญาณยูนิเวอร์เซล 64 กิโลบิทต่อวินาที จะเป็นสัญญาณซึ่งมีขนาดความเร็วพัลส์ 64 กิโลบิตต่อวินาที หรือความเร็วข่าวสาร เท่ากับความเร็วแบเรอ ดังนั้น ถ้าความเร็วแบเรอมีค่า L กิโลบิตต่อวินาที (L = 3.2, 6.4, 12.8, 64) แล้วโครงสร้างสัญญาณจะประกอบด้วยเฟรมในหน่วยของออคเตจ ซึ่งมีขนาด 64/L ในกรณีที่บิท F มีค่าเท่ากับ 0 คงที่

อนึ่งสัญญาณกรุ๊ป 0 มีพื้นฐานคาบแต่ละคาบเท่ากับ 20 ออคเตจ (160 บิท) ความถี่ 40 เฮิรตซ์ โดยมีโครงสร้างจากเฟรมเรียงด้วยสัญญาณที่ถูกมัลติเพล็กซ์ในแต่ละออคเตจ ส่งผลให้บิท F ในแต่ละออคเตจถูกปรับค่าตามแพทเทิร์นเฉพาะเพื่อใช้สำหรับนำเอาจังหวะดึงเฟรม นอกจานั้นยังสามารถมัลติเพล็กซ์สัญญาณที่มีความเร็วแบเรอที่ต่างกันในขั้นกรุ๊ป 0 เดียวกัน การมัลติเพล็กซ์สัญญาณแบเรอที่ขนาด L กิโลบิตต่อวินาที ทำได้สูงสุดที่ 64/L วงจร

สัญญาณกรุ๊ป 0 ซึ่งถูกสร้างโดยการมัลติเพล็กซ์ หลังจากไทม์ดิวชั่นมัลติเพล็กซ์ในหน่วยออคเตจเหมือนกับสัญญาณกรุ๊ป 0 ขนาด 1.544 เมกะบิตต่อวินาที ด้วยอุปกรณ์แลกเปลี่ยนมัลติเพล็กซ์กรุ๊ป 0 แล้วจะถูกส่งไปยังอุปกรณ์มัลติเพล็กซ์ขั้นต่อไป หรือถูกส่งไปยังเส้นทางทวนสัญญาณ

สำหรับสัญญาณกรุ๊ป 1 มีคาบพื้นฐาน 24 ออคเตจ + 1 บิท (193 บิท) ความถี่ 8 กิโลเฮิรตซ์ และโครงสร้างประกอบด้วยเฟรม ซึ่งถูกจัดเรียงด้วยบิทแพทเทิร์นที่ใช้สำหรับคาบของเฟรมที่ตำแหน่ง 193 บิทหลังสุด และออคเตจที่ 24 รวมกับสัญญาณกรุ๊ปที่ 0 ที่ถูกมัลติเพล็กซ์แล้ว

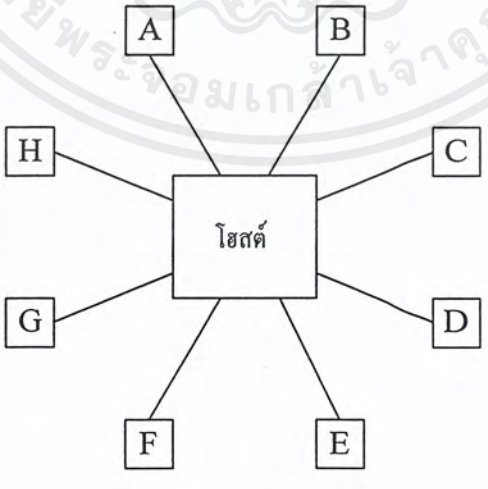
สำหรับสัญญาณกรุ๊ปที่ 1 ถ้าเป็นสัญญาณแบบเรอ 3.2 กิโลบิตต่อวินาที สามารถมัลติเพล็กซ์กรุ๊ป 0 ได้สูงสุด 23 วงจร

2.7 การใช้สายสื่อสารร่วมกัน

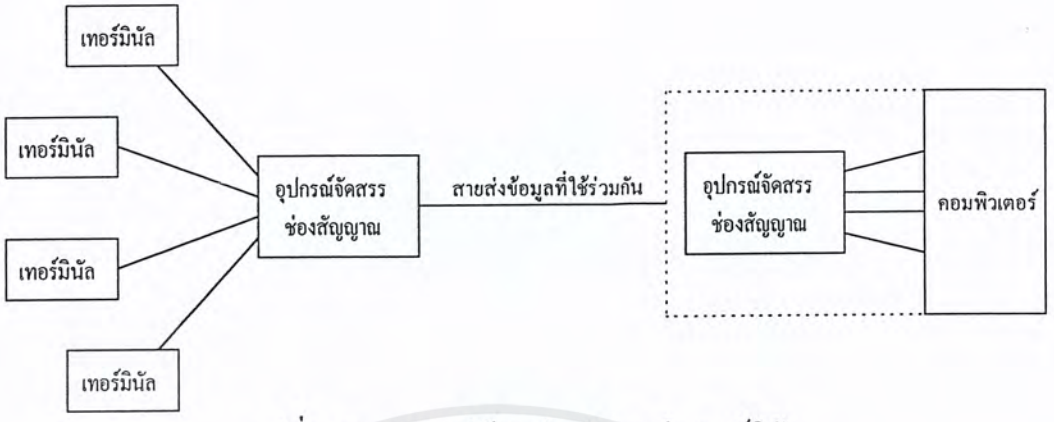
2.7.1 การจัดสรรความสามารถในการส่งข้อมูลของสายแบบสถิตและแบบพลวัต

ปกติแล้วสายเช่า (leased link) จะถูกกว่าสายแบบสวิตซ์ (switched link) หากมีการส่งข้อมูลมากกว่า 2-3 ชั่วโมงในแต่ละวัน นอกจากนั้นการส่งข้อมูลผ่านสายเช่า สามารถส่งด้วยอัตราส่งสูงและอัตราความผิดพลาดต่ำ

แต่อย่างไรก็ตาม การใช้สายเช่าเพื่อส่งข้อมูลจากเทอร์มินัลต่างๆเข้าสู่โฮสต์ในลักษณะจุดต่อจุดดังในแสดงในรูปที่ 2.16 นั้น เป็นการใช้สายส่งอย่างไม่มีประสิทธิภาพ เนื่องจากเวลาใช้งานของเทอร์มินัลเป็นเวลาของการป้อนข้อมูลเป็นส่วนใหญ่ ตัวอย่างเช่น สายเช่าซึ่งส่งข้อมูลด้วยอัตรา 9,600 บิตต่อวินาที นั้น สามารถส่งข้อมูลได้ถึง 17,280,000 บิต ภายในเวลา 30 นาที (1,800 วินาที * 9,600 บิต = 17,280,000 บิต) แต่สำหรับเทอร์มินัลที่รับข้อมูลจากคีย์บอร์ดนั้นภายในเวลา 30 นาทีนี้จะรับข้อมูลและส่งออกไปตามสายได้ประมาณ 4,000 ตัวอักษร (สำหรับนักพิมพ์ที่มีความชำนาญ) ดังนั้นกล่าวได้ว่าประสิทธิภาพของการใช้สายจะเป็น 4,000 * 8 / 17,280,000 = 0.0018 เท่านั้น จะเห็นว่าเป็นอัตราการใช้สายที่ต่ำมาก ดังนั้นเพื่อให้มีการใช้สายอย่างมีประสิทธิภาพ จึงอาจให้มีการใช้สายส่งข้อมูลร่วมกันระหว่างหลายเทอร์มินัลหรือหลายๆอุปกรณ์ ดังแสดงในรูป 2.17



รูปที่ 2.16 แสดงการเชื่อมโยงระหว่างเทอร์มินัลกับโฮสต์ ในลักษณะของจุดต่อจุด



รูปที่ 2.17 แสดงการใช้สายร่วมกันระหว่างเทอร์มินัล

ดังกล่าวมาแล้วว่าสายส่งแต่ละชนิดมีความสามารถในการส่งข้อมูลแตกต่างกัน เช่น สายคู่ตีเกลียวมีแบนด์วิธในระดับ 10^6 เฮิรตซ์ หรือแบนด์วิดธ์สำหรับการส่งข้อมูลด้วยดาวเทียมอยู่ในระดับถึง 500 เมกะเฮิรตซ์ (3.7 — 4.2 กิกะเฮิรตซ์) เป็นต้น ดังนั้นในการใช้สายร่วมกันระหว่างเทอร์มินัลหรืออุปกรณ์ต่างๆนั้น เพื่อป้องกันไม่ให้เทอร์มินัลต่างๆส่งข้อมูลออกมาชนกัน จะทำได้โดยการแบ่ง *ความสามารถในการส่งข้อมูล (capacity) ของสาย* ซึ่งอาจจะเป็นแบนด์วิดธ์หรือเวลาในการใช้สายให้แก่เทอร์มินัลต่างๆ

การแบ่งความสามารถในการส่งข้อมูลของสายนั้น อาจทำได้ 2 รูปแบบคือ

1. การแบ่งแบบสถิต (static allocation) เป็นการแบ่งความสามารถในการส่งข้อมูลให้แก่แต่ละเทอร์มินัลด้วยสัดส่วนที่คงที่แน่นอน และแต่ละเทอร์มินัลจะส่งข้อมูลได้ไม่มากกว่าความสามารถในการส่งข้อมูลที่ถูกแบ่งให้ นอกจากนั้นเมื่อเทอร์มินัลใดไม่ส่งข้อมูลในส่วนของตัวมันเอง ความสามารถในการส่งข้อมูลนั้นก็จะไม่สามารถเปลี่ยนไปให้เทอร์มินัลอื่นใช้ได้ ตัวอย่างของการใช้สายร่วมกันในลักษณะของการแบ่งแบบสถิต เช่น มัลติเพล็กซิงชนิดแบ่งตามความถี่ (frequency division multiplexing) และมัลติเพล็กซิงชนิดแบ่งตามเวลา (time division multiplexing) เป็นต้น

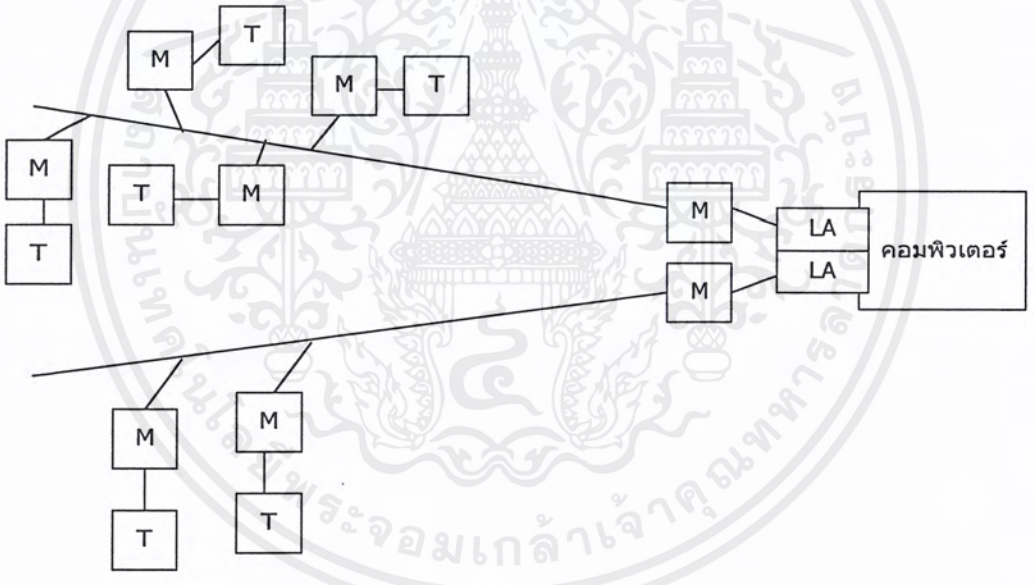
2. การแบ่งแบบพลวัต (dynamic allocation) เป็นการแบ่งความสามารถในการส่งข้อมูลให้แก่แต่ละเทอร์มินัลในลักษณะของพลวัต ทั้งนี้ขึ้นอยู่กับความต้องการในการส่งข้อมูลของเทอร์มินัลต่างๆ กล่าวคือจะแบ่งความสามารถในการส่งข้อมูลให้แก่เทอร์มินัลเมื่อเทอร์มินัลต้องการส่งข้อมูลหรือร้องขอ (request) มาเท่านั้น สำหรับการใช้สายร่วมกันชนิดที่แบ่งแบบพลวัตนี้ ได้แก่ มัลติดรอปไลน์ (multidrop line) มัลติเพล็กซิงชนิดแบ่งตามเวลาที่ใช้สถิติ (statistical TDM) ตลอดจนการส่งข้อมูลแบบแพ็กเกตสวิตชิง เป็นต้น

จะเห็นได้ว่าสำหรับการแบ่งความสามารถในการส่งข้อมูลของสายแบบสถิตนั้นอาจทำได้โดยเทคนิคง่ายๆ แต่ในขณะหนึ่ง หากเทอร์มินัลใดไม่ส่งข้อมูลก็จะมีการสูญเสียความสามารถในการ

ส่งข้อมูลของสายไป ส่วนการแบ่งตามความสามารถในการส่งข้อมูลแบบพลวัตนั้น ถึงแม้เทคนิคจะยุ่งยากและต้องเสียเฮดเดอร์บ้าง แต่ก็เป็นการใช้สายอย่างมีประสิทธิภาพ และสำหรับเทคนิคการใช้สายร่วมกันระหว่างเทอร์มินัลนั้นอาจจะแบ่งตามชนิดของการส่งข้อมูลในสายออกได้เป็น 2 วิธีใหญ่ๆ คือ มัลติดรอปไลน์ และมัลติเพล็กซิง

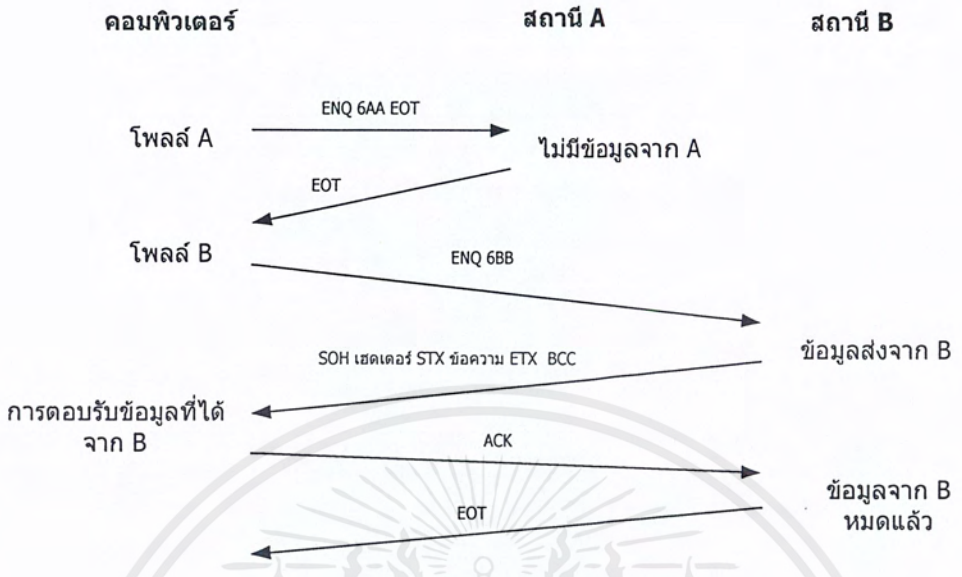
2.7.2 มัลติดรอปไลน์ (Multidrop Line)

จากรูปที่ 2.18 เป็นการแสดงรูปแบบ (configuration) ของการใช้สายร่วมกันแบบมัลติดรอปไลน์ ซึ่งจะเห็นได้ว่าถ้าหากเทอร์มินัลหลายตัวส่งข้อมูลลงในสายส่งพร้อมกัน ข้อมูลจะชนกันและเสียหายได้ ซึ่งปรากฏการณ์ที่เทอร์มินัลหลายตัวพยายามส่งข้อมูลลงในสายพร้อมกันนี้ (กล่าวคือใช้ทรัพยากร (resource) พร้อมกัน) เรียกว่า การแย่งชิง (contention) ดังนั้นจึงต้องมีวิธีการควบคุมไม่ให้มีการส่งข้อมูลลงไปชนกันในสาย ซึ่งการควบคุมนี้ปกติแล้วอาศัย 2 วิธีคือ โพลลิง (polling) และซีเล็กติง (selecting)

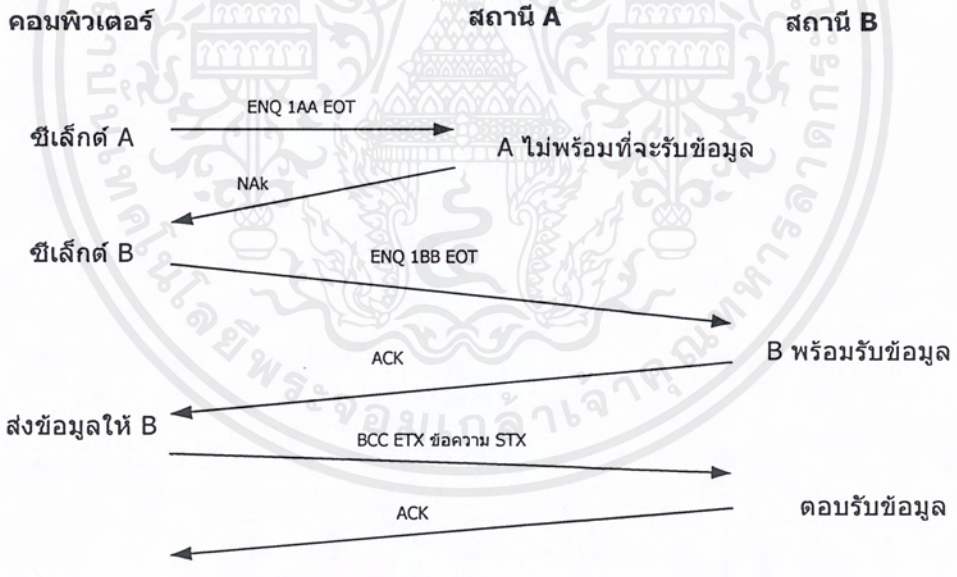


รูปที่ 2.18 แสดงรูปแบบของมัลติดรอปไลน์

โพลลิง เป็นวิธีการที่คอมพิวเตอร์ส่งข้อมูลหรือโพลลิงเมสเสจ (polling message) ไปถามเทอร์มินัลต่างๆ เวียนสลับกันไปว่ามีข้อมูลต้องการส่งหรือไม่ ถ้ามีข้อมูลที่จะส่ง เทอร์มินัลก็สามารถส่งข้อมูลออกไป แต่ถ้าไม่มี เทอร์มินัลจะตอบปฏิเสธ (no-traffic response) ต่อโพลลิงเมสเสจที่ได้รับมานั้น สำหรับตัวอย่างของวิธีการโพลลิงได้แสดงในรูปที่ 2.19



รูปที่ 2.19 แสดงกระบวนการโพลลิง



รูปที่ 2.20 แสดงกระบวนการซีเล็กติง

ในรูปที่ 2.19 นี้จะเห็นได้ว่าค่า 6 ในโพลลิงเมสเสจบ่งบอกถึงการโพลล์ ส่วน EOT ที่ส่งก่อนหน้าโพลลิงเมสเสจนั้นไม่จำเป็นเท่าใดนัก เพียงแต่เป็นสัญญาณบอกให้เทอร์มินัลทุกตัวที่อยู่ในสายให้อยู่ในสภาวะที่พร้อมรับคำสั่ง กล่าวคือไม่ให้ส่ง 1RTS (Request To Send) ออกมา

สำหรับการ *ซีเล็กติง* นั้นเป็นวิธีการที่คอมพิวเตอร์ส่งซีเล็กติงเมสเสจ (selecting message) ไปถามเทอร์มินัลต่างๆว่าจะพร้อมรับข้อมูลหรือไม่ หากเทอร์มินัลพร้อมก็จะตอบรับ (acknowledge) กลับไป แต่หากไม่พร้อม เทอร์มินัลก็สามารถปฏิเสธการรับข้อมูลได้ สำหรับตัวอย่างของวิธีการซีเล็กติงแสดงดังรูปที่ 2.20

2.8 มัลติเพล็กซ์เซอร์

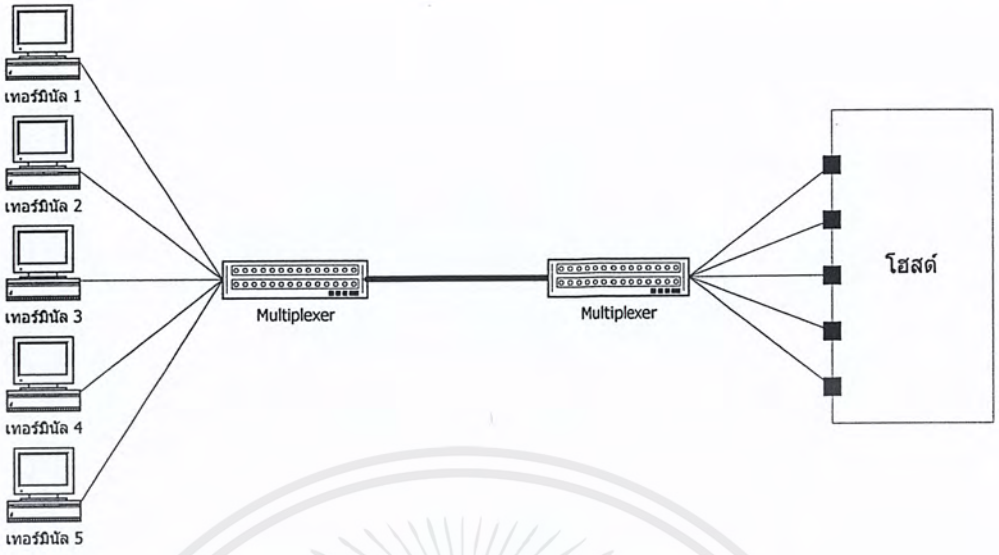
อุปกรณ์มัลติเพล็กซ์เซอร์ เป็นอุปกรณ์ที่จะช่วยให้เทอร์มินัลสามารถใช้สายร่วมกันได้ โดยมัลติเพล็กเซอร์จะรวบรวมข้อมูลอัตราส่งต่ำจากหลายเทอร์มินัล และส่งออกพร้อมๆกันด้วยอัตราการส่งสูงไปยังปลายอีกด้านหนึ่ง ซึ่งจะมีมัลติเพล็กเซอร์อีกตัวหนึ่งทำหน้าที่ *แยกสัญญาณ (demultiplex)* ข้อมูลที่ได้รับมา เพื่อที่จะได้ข้อมูลเดิมซึ่งแยกตามเทอร์มินัล โดยปกติแล้วการใช้มัลติเพล็กเซอร์นั้นผู้ใช้จะรู้สึกเหมือนกับว่าใช้เทอร์มินัลเชื่อมต่อแบบจุดต่อจุดกับอุปกรณ์ที่อยู่อีกฝั่งหนึ่ง (ซึ่งอาจจะเป็นพอร์ตของโฮสต์หรือคอมพิวเตอร์) ดังแสดงในรูปที่ 2.21

สำหรับการทำงานของมัลติเพล็กเซอร์เป็นการแบ่งความสามารถในการส่งข้อมูลของสายส่งให้แก่แต่ละเทอร์มินัล โดยนำข้อมูลจากเทอร์มินัลหลายๆเทอร์มินัลส่งไปในสายส่งสัญญาณเพียงเส้นเดียว ซึ่งที่สายส่งสัญญาณนี้ถูกแบ่งออกเป็นเซคชั่น และข้อมูลจจากแต่ละเทอร์มินัลจะถูกส่งมาโดยเซคชั่นที่เฉพาะเจาะจง

การส่งข้อมูลแบบมัลติเพล็กซ์นี้ จะประหยัดกว่าการที่ในแต่ละเทอร์มินัลแยกส่งข้อมูลกัน จากรูปที่ 2.1 จะแสดงรูปแบบของระบบการสื่อสาร โดยใช้มัลติเพล็กซ์เทอร์มินัล (Multiplex Terminal) โดยที่มีตัวมัลติเพล็กซ์เซอร์ทั้ง 2 ตัว ทำการเชื่อมต่อระหว่างเทอร์มินัลกับคอมพิวเตอร์ และข้อมูลที่ส่งผ่าน มักจะเป็นเบริส์ท (burst) ไม่ใช่ข้อมูลที่ส่งมาอย่างต่อเนื่อง

การมัลติเพล็กซ์ ที่นิยมใช้กันมากมี 3 วิธี คือ

1. การมัลติเพล็กซ์แบบแบ่งตามเวลา (Frequency Division Multiplexing :FDM) ซึ่งเป็นแบบที่นิยมกันมากที่สุด โดยเฉพาะด้านการวิทยุและโทรทัศน์
2. การมัลติเพล็กซ์แบบแบ่งตามเวลา (Time Division Multiplex : TDM) ซึ่งรู้จักดีในชื่อของซิงโครนัส TDM (Synchronous TDM) ส่วนใหญ่ใช้ในการการมัลติเพล็กซ์สัญญาณเสียงดิจิตอล
3. การมัลติเพล็กซ์แบบแบ่งตามเวลาด้วยสถิติ (Statistical Division Multiplexing :STDM) ซึ่งเรียกอีกอย่างว่า อะซิงโครนัส TDM (Asynchronous TDM) หรือ อินเทลลิเจนท์ TDM (Intelligent TDM) ในที่นี้ เราจะใช้ชื่อเรียกสั้นๆว่า STDM ซึ่งเป็นวิธีการมัลติเพล็กซ์ที่ปรับปรุงการทำงานมาจากวิธีซิงโครนัส TDM ให้มีประสิทธิภาพสูงยิ่งขึ้น เพื่อรองรับจำนวนช่องสัญญาณให้ได้มากขึ้น



รูปที่ 2.21 แสดงการใช้มัลติเพล็กซ์เมื่อส่งข้อมูลจากหลายเทอร์มินัล

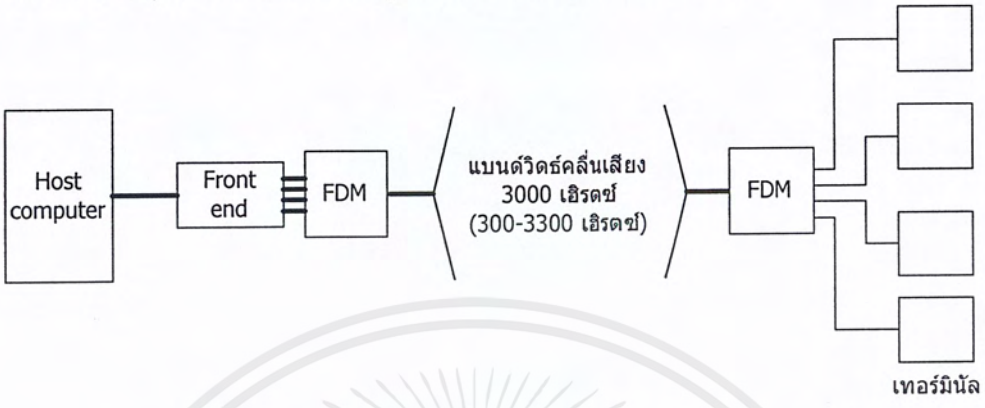
2.8.1 มัลติเพล็กซิงชนิดแบ่งตามความถี่ (FDM)

การมัลติเพล็กซิงชนิดแบ่งตามความถี่ เป็นเทคนิคคล้ายกับการแพร่สัญญาณของวิทยุและโทรทัศน์ที่มีการแบ่งคลื่นส่งให้แก่แต่ละสถานีส่ง ซึ่งในรูปที่ 2.23 แสดงวิธีการของการมัลติเพล็กซิงชนิดแบ่งตามความถี่ จากรูปจะเห็นว่าเป็นการแบ่งความถี่ของช่องสัญญาณของสายส่งออกเป็นช่องสัญญาณย่อยๆ และใช้ช่องสัญญาณ (หรือช่องความถี่) ย่อยแต่ละช่องนั้นสำหรับการนำข้อมูลของแต่ละเทอร์มินัล ซึ่งแต่ละเทอร์มินัลจะส่งข้อมูลภายในช่องความถี่ที่ได้รับเท่านั้น นอกจากนั้นระหว่างช่องความถี่ย่อยจะมีการ์ดแบนด์ (guard band) ซึ่งช่วยไม่ให้เกิดการรบกวนจากคลื่นของช่องความถี่ข้างเคียง เทคนิคการมัลติเพล็กซิงชนิดแบ่งตามความถี่นี้ จะใช้หลักการของ FSK ในการส่งข้อมูล 1 และ 0 ดังแสดงในรูปที่ 2.24 ซึ่งเห็นได้ว่าเมื่อข้อมูลของเทอร์มินัลมีค่าเป็น 1 ความถี่ของสัญญาณจะมีค่าแตกต่างกับเมื่อข้อมูลมีค่าเป็น 0 และเนื่องจากต้องมีการ์ดแบนด์ จึงทำให้อัตราการส่งข้อมูลโดยรวมทุกช่องสัญญาณย่อยของ FDM จะน้อยกว่าอัตราการส่งข้อมูลของช่องสัญญาณที่ไม่ได้ถูกแยกออกเป็นช่องสัญญาณย่อย

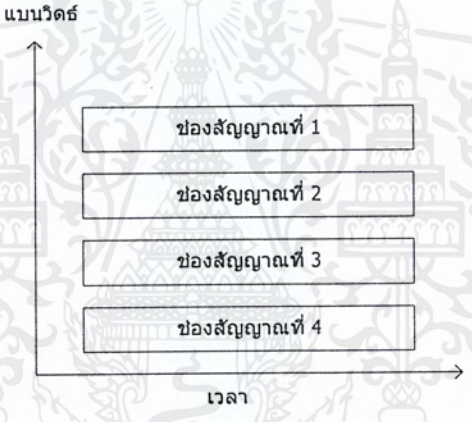
ดังนั้น อาจจะกล่าวได้ว่าเทคนิคการมัลติเพล็กซิงชนิดแบ่งตามความถี่ เป็นวิธีการที่ค่อนข้างจะง่าย และมัลติเพล็กเซอร์เองจะมีโมเด็มอยู่ในตัว ดังนั้นมัลติเพล็กเซอร์แบบ FDM จะมีราคาถูก แต่ในทางตรงข้าม FDM มีประสิทธิภาพในการใช้สายต่ำ เนื่องจาก

1. ทำงานโดยแบ่งความสามารถในการส่งข้อมูลของสายเป็นแบบสถิต (static allocation)

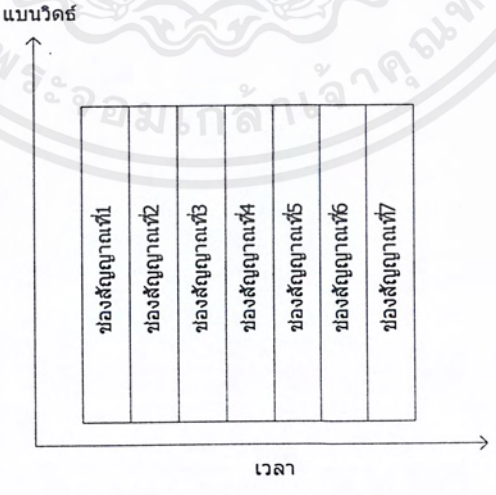
2. อัตราการส่งข้อมูลโดยรวมต่ำ ซึ่งปกติมักจะต่ำกว่า 2,000 บิตต่อวินาที ในขณะที่โมเด็มความเร็วสูงอาจใช้ส่งข้อมูลได้อัตราถึง 9,600 บิตต่อวินาที ดังนั้นจึงมักจะใช้ FDM กับระบบซึ่งมีอัตราการส่งข้อมูลน้อย เช่น การส่งข้อมูลเทเล็กซ์ผ่านวงจรโทรศัพท์ เป็นต้น



รูปที่ 2.22 แสดงหลักการของการมัลติเพล็กซิงชนิดแบ่งตามความถี่



(ก) มัลติเพล็กซ์แบบแบ่งตามความถี่



(ข) มัลติเพล็กซ์แบบแบ่งตามเวลา

รูปที่ 2.23 แสดงภาพของ FDM และ TDM

2.8.2 มัลติเพล็กซิงชนิดแบ่งตามเวลา (TDM)

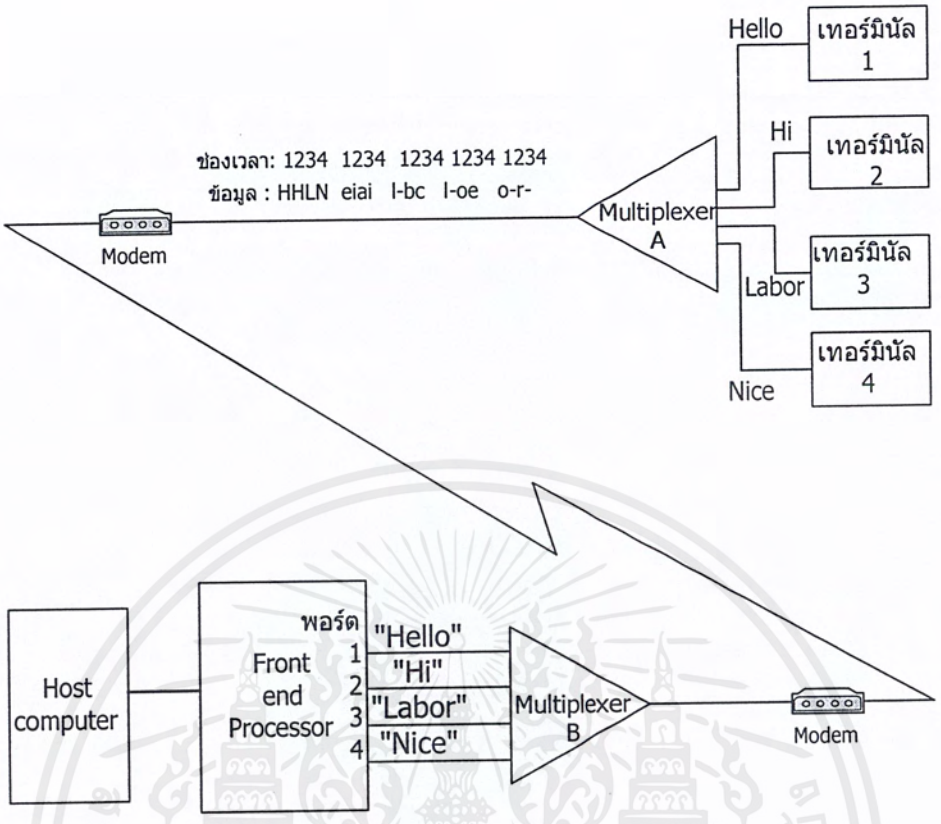
มัลติเพล็กซิงชนิดแบ่งตามเวลา เป็นการแบ่งเวลาในการใช้สายส่งเพื่อใช้ส่งข้อมูลแก่แต่ละเทอร์มินัล โดยที่ข้อมูลของแต่ละเทอร์มินัลจะถูกสลับเวียนส่งลงไปในสาย ซึ่งตัวอย่างการทำงานของ TDM ได้แสดงในรูปที่ 2.24 จะเห็นได้ว่ามัลติเพล็กเซอร์นำเอาข้อมูลตัวอักษรจากแต่ละเทอร์มินัลซึ่งมีอัตราการส่งข้อมูลต่ำมาทำเป็นเฟรม เพื่อส่งออกสู่สายส่งด้วยอัตราการส่งข้อมูลสูงไปยังฝั่งรับ ซึ่งจะแยกข้อมูลตัวอักษรในเฟรมให้แก่อุปกรณ์ฝั่งรับ เช่น พอร์ตของคอมพิวเตอร์ได้อย่างถูกต้อง

จากรูปที่ 2.24 (ก) จะเห็นได้ว่ามัลติเพล็กเซอร์ใช้เวลาในการส่งข้อมูลตัวอักษรของแต่ละเทอร์มินัลสลับกันไป ซึ่งช่วงเวลาที่แต่ละเทอร์มินัลได้รับเพื่อใช้ส่งข้อมูล เรียกว่า ช่องเวลา (time slot) เช่น ข้อมูลจากเทอร์มินัลที่ 1 ถูกส่งไปในช่องเวลาแรก ข้อมูลจากเทอร์มินัลที่ 2 ส่งไปในช่องเวลาที่ 2 เป็นต้น และมัลติเพล็กเซอร์ฝั่งตรงข้ามจะทำงานสอดคล้องกัน (ซิงโครไนเซชัน) โดยแยกข้อมูลในแต่ละช่องเวลาให้แก่อุปกรณ์ฝั่งรับได้อย่างถูกต้องสอดคล้อง

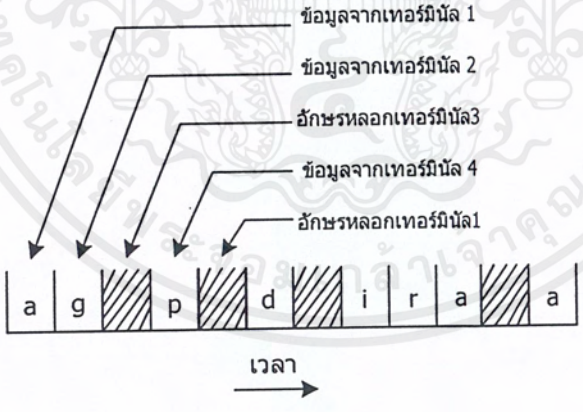
ดังนั้นถ้าแต่ละเทอร์มินัลส่งข้อมูลด้วยอัตราส่ง 1,200 บิตต่อวินาที แล้ว ข้อมูลที่ถูกส่งโดย TDM จะมีอัตราส่ง 4,800 บิตต่อวินาที และนอกจากนั้นในช่องเวลาของตัวเอง ถ้าเทอร์มินัลใดไม่มีข้อมูลจะส่งมัลติเพล็กเซอร์จะใส่อักษรหลอก (dummy character) เข้าไปแทนเพื่อให้มัลติเพล็กเซอร์ฝั่งตรงข้ามสามารถทำงานได้สอดคล้องกัน ซึ่งตัวอย่างของ TDM ที่กล่าวข้างต้น เป็น คาแรคเตอร์-อินเตอร์ลีฟวิง (character interleaving) หรือ ไบต์-อินเตอร์ลีฟวิง (byte interleaving) กล่าวคือเป็นการมัลติเพล็กซ์ข้อมูลตัวอักษรเพื่อส่งเป็นเฟรมไปพร้อมกัน

บิตซึ่งมีค่าเป็น 1 หรือ 0 แล้ว จะเป็นการยากสำหรับมัลติเพล็กเซอร์ฝั่งรับที่จะทราบว่าบิตไหนเป็นบิตนอกจากแบบ คาแรคเตอร์-อินเตอร์ลีฟวิงแล้ว การมัลติเพล็กซ์ยังอาจเป็นแบบ บิต-อินเตอร์ลีฟวิง ซึ่งแบบนี้ มัลติเพล็กเซอร์จะนำเอาบิตข้อมูลของแต่ละเทอร์มินัลมามัลติเพล็กซ์กัน เพื่อส่งออกพร้อมกันเป็นเฟรม ดังแสดงในรูปที่ 2.24(ข) สำหรับบิต-อินเตอร์ลีฟวิงนี้จะแตกต่างกับคาแรคเตอร์-อินเตอร์ลีฟวิง ตรงที่ ถ้าเทอร์มินัลใดไม่มีข้อมูลส่ง มัลติเพล็กเซอร์จะใส่อักษรหลอกลงไปในช่องเวลาของเทอร์มินัลนั้น แต่ถ้าเป็นข้อมูลหลอก ดังนั้นสำหรับ บิต-อินเตอร์ลีฟวิง นี้เฟรมจะมีเฮดเดอร์บ่งบอกว่าบิตในช่องเวลาใดไม่ใช่ข้อมูลจริง

ปกติแล้วอัตราการส่งข้อมูลลงไปในสายของมัลติเพล็กเซอร์ จะมากกว่าหรือเท่ากับอัตรการส่งข้อมูลโดยรวมของทุกเทอร์มินัลที่ต่อกับมัลติเพล็กเซอร์ นอกจากนั้นการที่มัลติเพล็กเซอร์นำข้อมูลตัวอักษรมาทำเป็นเฟรมนั้น จะมีการใช้เฮดเดอร์บางส่วนเข้าไปด้วยเพื่อให้มัลติเพล็กเซอร์ทั้งสองฝั่งสามารถทำงานสอดคล้องกันอย่างถูกต้อง กล่าวคือการมัลติเพล็กซิงต้องสอดคล้องกับการดีมัลติเพล็กซิง ดังในรูปที่ 2.25 ซึ่งมัลติเพล็กเซอร์ย่อมต้องใช้เวลาในการส่งโอเวอร์เฮดเหล่านี้

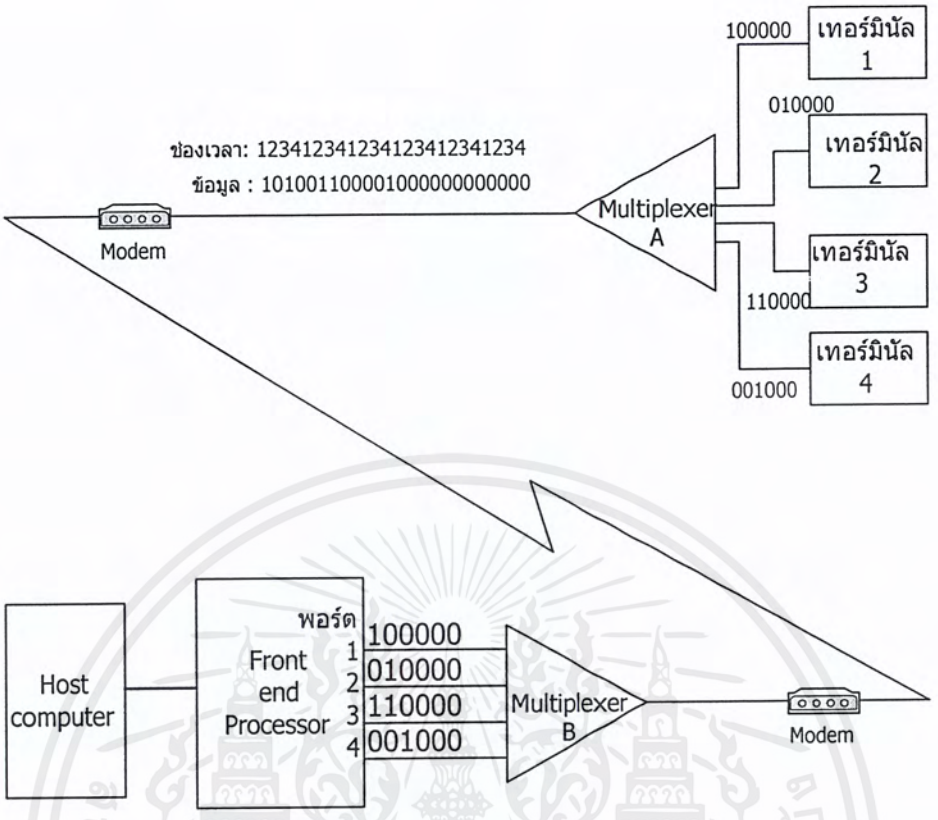


(ก) แสดงการทำงานของ TDM



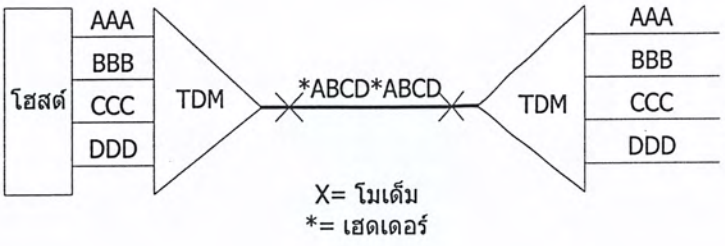
(ข) เมื่อใส่อักษรหลอกลงในช่องเวลาของเทอร์มินัลที่ไม่มีข้อมูลส่ง

รูปที่ 2.24 แสดงการทำงานของ TDM

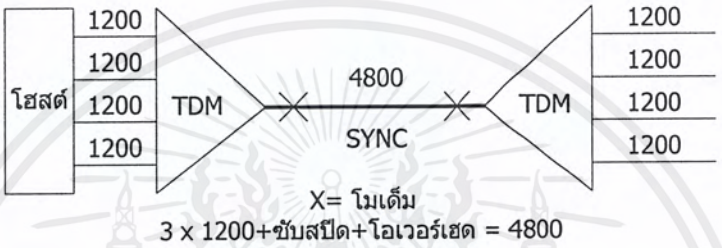


รูปที่ 2.25 แสดงตัวอย่างของ บิต-อินเตอร์ลีฟวิง

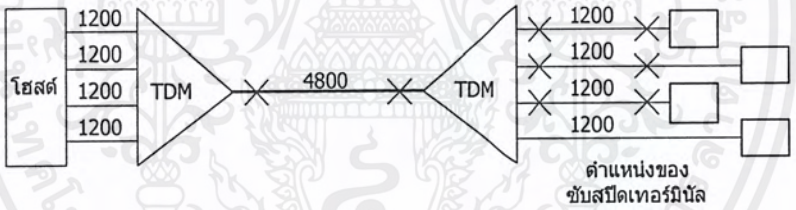
ถ้าเทอร์มินัลที่ใช้เป็นซิงโครนัสเทอร์มินัล การที่ต้องส่งเฮดเดอร์เหล่านี้ทำให้อัตราการส่งข้อมูลลงสายของมัลติเพล็กเซอร์ไม่สามารถเท่ากับอัตราการส่งข้อมูลโดยรวมของเทอร์มินัลได้ ดังนั้นโดยทั่วไปจะมีเทอร์มินัลตัวหนึ่งส่งข้อมูลด้วยอัตราการส่งที่ต่ำเรียกว่า *ซับสปีด (supspeed)* ดังในแสดงในรูปที่ 2.26 (ก) , (ข) และ จากรูปที่ 2.26 (ข) จะเห็นว่าในกรณีที่เทอร์มินัลอยู่ไกลจากมัลติเพล็กเซอร์ และต้องใช้โมเด็มในการส่งข้อมูลจากเทอร์มินัลสู่มัลติเพล็กเซอร์แล้ว พอร์ตของมัลติเพล็กเซอร์สำหรับเทอร์มินัลนั้นไม่ควรจะเป็นซับสปีดไลน์ (supspeed line) เพราะปกติโมเด็มมีอัตราการส่งข้อมูลที่แน่นอน และการส่งข้อมูลแบบซิงโครนัสนั้นต้องมีการสอดคล้องของจังหวะเวลาของการรับส่งข้อมูลเสมอ ดังนั้นหากอัตราการส่งข้อมูลของเทอร์มินัลซึ่งเป็นซับสปีดต่างไปจากอัตราการส่งข้อมูลของโมเด็มเกินกว่า 0.01% แล้ว จะทำให้การทำงานของโมเด็มผิดพลาดได้ เช่น โมเด็มมีอัตราส่ง 1,200 บิตต่อวินาที แล้วถ้าซับสปีดมีอัตราการส่ง 1,100 บิตต่อวินาที จะทำให้การรับส่งข้อมูลไม่ถูกต้อง



รูปที่ 2.26 แสดงตัวอย่างของเฟรมซึ่งมีเฮดเดอร์



(ก) ตัวอย่างของซับสปีดพอร์ต



(ข) ซับสปีดพอร์ตต้องติดอยู่กับมัลติเพล็กเซอร์

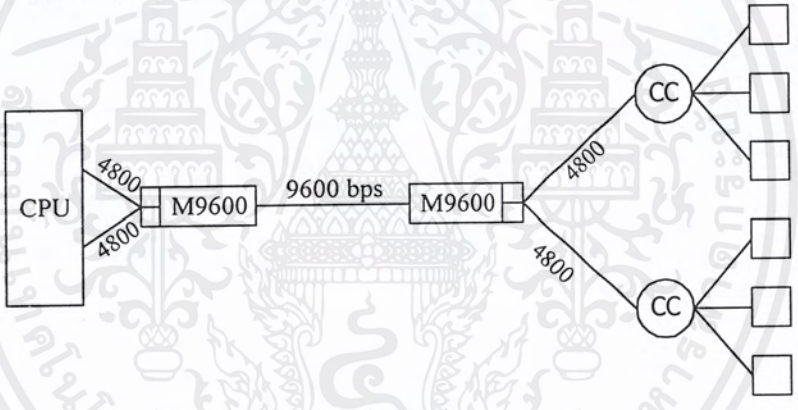
รูปที่ 2.27 แสดงการมัลติเพล็กซ์ข้อมูลจากซิงโครนัสเทอร์มินัล

สำหรับกรณีของอะซิงโครนัสนั้น การส่งข้อมูลของแต่ละตัวอักษรจะมีการใส่บิตเริ่มและบิตจบเข้าไปด้วย ซึ่งมัลติเพล็กเซอร์จะถอดบิตเริ่มและบิตจบนี้ออกและนำข้อมูลตัวอักษรจริงๆเท่านั้นไปรวมทำเป็นเฟรมและส่งแบบซิงโครนัสลงไปในสาย และจะเป็นหน้าที่ของมัลติเพล็กเซอร์ฝั่งรับในการที่จะแยกซิงโครนัสเฟรม แล้วใส่บิตเริ่มและบิตจบให้แก่แต่ละตัวอักษรเพื่อส่งให้แก่อุปกรณ์ฝั่งรับ ดังนั้นจะเห็นว่าถึงแม้เทอร์มินัลจะส่งข้อมูลด้วยอัตรา 1,200 บิตต่อวินาที ก็ตาม (ข้อมูลซึ่งรวมบิตเริ่มและบิตจบ) อัตราการส่งข้อมูลตัวอักษรจริงจะไม่ถึง 1,200 บิตต่อวินาที และเมื่อมัลติเพล็กเซอร์นำข้อมูลเฉพาะตัวอักษรไปทำเป็นเฟรมซึ่งมีโอเวอร์เฮดรวมไปด้วยนั้น ถึงแม้มัลติเพล็กเซอร์จะใช้เวลาในการส่งโอเวอร์เฮดด้วยก็ตาม อัตราการส่งเฟรมข้อมูลลงในสายของมัลติเพล็กเซอร์จะสามารถ

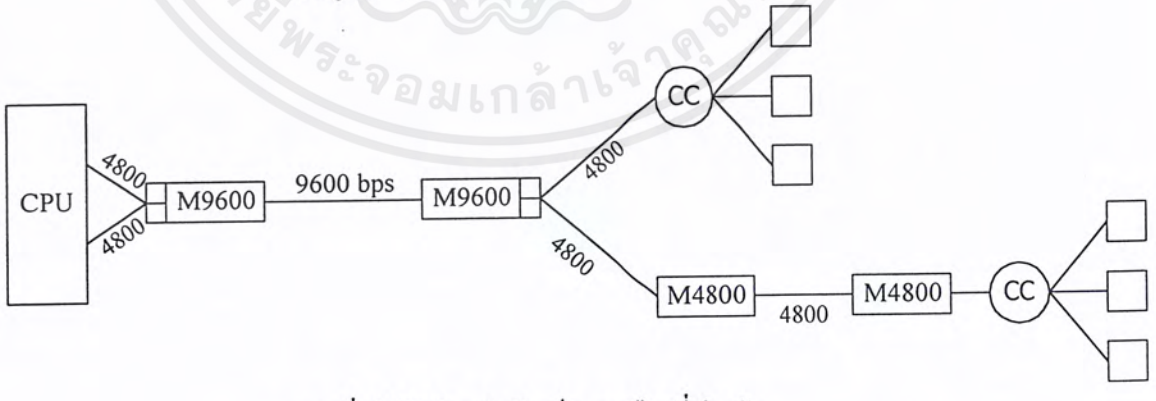
เท่ากับอัตราการส่งข้อมูลโดยรวมของทุกอะซิงโครนัสเทอร์มินัล เช่น 4 * 1,200 = 4,800 บิตต่อวินาที ได้นั่นเอง

2.8.3 มัลติสตรีมโมเด็ม (Multistream Modem)

ดังได้กล่าวมาแล้วข้างต้นว่า เมื่อใช้มัลติเพล็กซิงชนิดแบ่งตามเวลา กับซิงโครนัสเทอร์มินัล จะทำให้เทอร์มินัลบางตัวต้องส่งข้อมูลซับสปีด เพื่อทดแทนกับการส่งโอเวอร์เฮดของมัลติเพล็กเซอร์ ในการทำให้มัลติเพล็กเซอร์ 2 ตัวทำงานสอดคล้องกัน ดังนั้นจึงมีการนำเอามัลติสตรีมโมเด็มซึ่งเป็นโมเด็มที่สามารถทำการมัลติเพล็กซ์ข้อมูลได้ด้วยมาใช้งาน เพื่อทำการกล้ำข้อมูลและแยกข้อมูล จึงสามารถใช้จังหวะเวลานี้ในการมัลติเพล็กซ์และดีมัลติเพล็กซ์ ดังนั้นจึงไม่จำเป็นต้องมีโอเวอร์เฮดของเฟรมอีก และจะทำให้อัตราการส่งข้อมูลของมัลติสตรีมโมเด็มเท่ากับอัตราการส่งข้อมูลโดยรวมของซิงโครนัสเทอร์มินัลได้ สำหรับรูปที่ 2.28 แสดงการใช้งานของมัลติสตรีมโมเด็มในรูปแบบต่างๆกันไป ในรูปนี้ อุปกรณ์ควบคุมกลุ่ม (cluster controller) ช่วยให้โฮสต์สามารถโพลล์เทอร์มินัลทั้งกลุ่ม (group poll) เพียงครั้งเดียวแทนที่จะต้องโพลล์แต่ละเทอร์มินัล



(ก) รูปแบบมัลติสตรีมโมเด็ม



(ข) เมื่อใช้มัลติสตรีมโมเด็มจากเมืองที่ห่างกัน

รูปที่ 2.28 แสดงตัวอย่างของการใช้มัลติสตรีมโมเด็มในรูปแบบต่างๆ

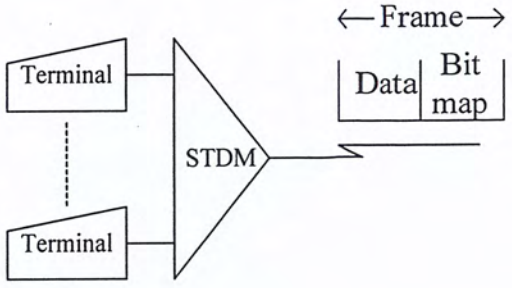
2.8.4 การมัลติเพล็กซ์แบบแบ่งตามเวลาด้วยสถิติ (Statistical Division Multiplexing :STDM)

การมัลติเพล็กซ์แบบแบ่งตามเวลาด้วยสถิติ เรียกอีกอย่างว่า อะซิงโครนัส TDM (Asynchronous TDM) หรือ อินเทลลิเจนท์ TDM (Intelligent TDM) ในที่นี้ เราจะใช้ชื่อเรียกสั้นๆว่า STDM ซึ่งเป็นวิธีการมัลติเพล็กซ์ที่ปรับปรุงการทำงานมาจากวิธีซิงโครนัส TDM ให้มีประสิทธิภาพสูงยิ่งขึ้นเพื่อรองรับจำนวนช่องสัญญาณให้ได้มากขึ้น

ทั้ง TDM และ FDM ทำงานโดยแบ่งความสามารถในการส่งข้อมูลของสายส่ง (ไม่ว่าเป็นแบนด์วิดธ์ หรือช่องเวลา) ให้แก่แต่ละเทอร์มินัลในลักษณะของการแบ่งแบบสถิต ซึ่งหากเทอร์มินัลใดไม่ส่งข้อมูลก็จะเกิดการสูญเสียความสามารถของการส่งข้อมูล ดังนั้นจึงมีการนำเอา STAT MUX (STDM) มาใช้งาน ซึ่ง STAT MUX จะให้ความสามารถในการส่งข้อมูลแก่เทอร์มินัลเมื่อมีการร้องขอมาเท่านั้น กล่าวคือ STDM จะทำงานโดยเฉลี่ยเวลาการใช้สายเพื่อส่งข้อมูลให้แก่เทอร์มินัลต่างๆ เมื่อเทอร์มินัลตัวใดไม่มีข้อมูลส่ง STDM ก็จะเฉลี่ยเวลานั้นเพื่อใช้ในการส่งข้อมูลของเทอร์มินัลตัวอื่น

ดังนั้น ตามปกติแล้ว STDM จะเหมาะสำหรับเทอร์มินัลซึ่งทำงานแบบอะซิงโครนัส เพราะในบางขณะอาจมีเทอร์มินัลบางตัวไม่ทำงาน และสำหรับเทอร์มินัลที่ ทำงานนั้นก็มักส่งข้อมูลไม่ถึง 20% ของอัตราการส่งข้อมูลของมันเมื่อเทอร์มินัลถูกใช้งานแบบโต้ตอบ (interactive) ดังนั้น STDM ซึ่งต่อเข้ากับเทอร์มินัล 16 ตัว อาจจะถูกโปรแกรมให้รับข้อมูลจาก 12 เทอร์มินัลเท่านั้นในขณะหนึ่งๆ โดยการส่งสัญญาณ CTS ให้แก่เทอร์มินัลแค่ 12 ตัวในเวลาหนึ่ง และเฉลี่ยเวลาการใช้สายแก่ 12 เทอร์มินัลนี้ตามการร้องขอ และเพื่อให้มัลติเพล็กเซอร์ฝั่งรับสามารถแยกข้อมูลออกจากเฟรมแล้วส่งให้แก่อุปกรณ์ฝั่งรับได้อย่างถูกต้อง ในแต่ละเฟรมจะมี (mapping) (หรือเฮดเดอร์ เช่น แอดเดรส) ซึ่งบ่งบอกว่าข้อมูลส่วนใดเป็นของอุปกรณ์ไหนและมีจำนวนเท่าไร

ในการเปรียบเทียบระหว่าง TDM กับ STDM นี้ ใช้การกำหนดเฟรมกับตำแหน่งของข้อมูลในแต่ละเฟรมซึ่ง STDM สามารถใช้กับเฟรมที่สามารถปรับเปลี่ยนความยาวได้ โดยที่การจะเข้าใจการจัดการของ STDM ลองสมมติว่ามีเทอร์มินัล 8 เทอร์มินัลต่ออยู่กับตัวมัลติเพล็กซ์เซอร์ที่แสดงดังรูปที่ 2.29 ถ้าในระหว่างระบบการสแกน 1 ครั้ง เทอร์มินัล 1,2,3 และ 6 มีข้อมูลอินพุทและส่งข้อมูลที่เป็นตัวอักษร ถ้าSTDM จัดการกับบิทแต่ละตัวของอินพุท ถ้าพอร์ทนั้นแอคทีฟ จะเซทบิทเป็น 1 แต่ถ้าพอร์ทนั้นไม่แอคทีฟจะเซทค่าให้เป็น 0



ก เฟรมข้อมูล

Terminal	Terminal Active
1	X
2	Y
3	Z
4	No Active
5	No Active
6	Q
7	No Active
8	No Active

frame

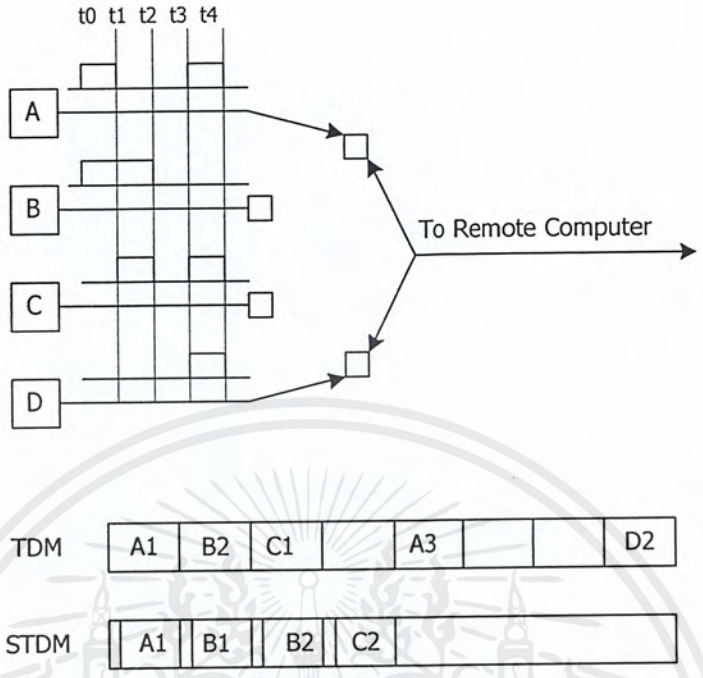
100111
X
Y
Z
Q

ข โครงสร้างของเฟรม

รูปที่ 2.29 การทำงานของ STDM

ตัวอย่างเฟรม แสดงในรูป 2.29 (ข) เป็นการแสดงวิธีใช้บิทแมป (Bit Map) ที่สามารถลดขนาดของเฟรมได้เมื่อเฟรมนั้นไม่แอคทีฟ หรือเฟรมนั้นไม่มีข้อมูล ซึ่งถ้าเทอร์มินัล 1,2,3 และ 6 เป็นแอคทีฟแล้ว ค่าสถานะบิทแมปจะถูกเซทให้เป็น1 อักขระจากแต่ละเทอร์มินัลจะเพิ่มขึ้นหลังจากมีการบิทแมปเป็นรูปแบบของเฟรมที่มีความยาวแตกต่างกัน และการบิทแมปจะแสดงตำแหน่งที่แจริงของอักขระแต่ละตัวในเฟรม

เนื่องจากว่า TDM ได้ใช้ประโยชน์จากหลักความเป็นจริงที่ว่า ตัวอุปกรณ์ที่อยู่กับตัวมัลติเพล็กซ์ไม่ได้มีความต้องการในการส่งข้อมูลกันทุกตัวตลอดเวลา อัตราการส่งข้อมูลในสายการมัลติเพล็กซ์จึงน้อยกว่าผลรวมของอัตราข้อมูลของอุปกรณ์ที่มาต่อด้วย ซึ่งตัวมัลติเพล็กซ์ทางสถิติสามารถใช้อัตราข้อมูลที่ให้บริการแก่จำนวนอุปกรณ์ได้เท่าๆกับตัวมัลติเพล็กซ์แบบซิงโครนัส ในทางกลับกัน ถ้าตัวมัลติเพล็กซ์ทางสถิติกับตัวมัลติเพล็กซ์แบบซิงโครนัสใช้กับสายการเชื่อมต่อข้อมูลด้วยอัตราข้อมูลที่เท่ากันแล้ว ตัวมัลติเพล็กซ์เชิงสถิติก็จะสามารถให้บริการแก่อุปกรณ์ได้มากกว่า



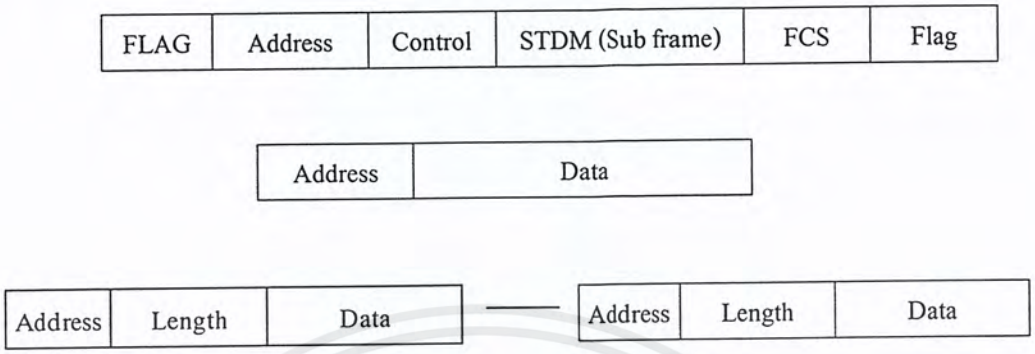
รูปที่ 2.30 แสดงการมัลติเพล็กซ์ TDM เทียบกับ STDM

จากรูป 2.30 ได้เปรียบเทียบให้เห็นความแตกต่างของ TDM แบบสถิติกับแบบซิงโครนัส ในรูปจะแสดงแหล่งข้อมูลสี่แหล่ง, แสดงข้อมูลที่แหล่ง และแสดงข้อมูลที่เกิดขึ้นที่เวลาทั้งสี่ ในกรณีของ TDM แบบซิงโครนัส ในบางเฟรมจะมีการส่งช่องสัญญาณที่ว่างไปด้วย แต่สำหรับ TDM แบบสถิติจะไม่ปล่อยให้มีช่องเวลาที่ว่างในการส่ง เฟรมแต่ละครั้งถ้ามีข้อมูลที่จะส่งอยู่

โครงสร้างของเฟรมที่ใช้กับการมัลติเพล็กซ์เชิงสถิติจะมีลักษณะเด่นกล่าวคือมีบิทควบคุมอยู่ส่วนหัว (ที่ต้องการให้มีจำนวนน้อยที่สุดเพื่อให้การส่งผ่านข้อมูลได้ข้อมูลจำนวนมากๆ) โดยทั่วไประบบ TDM เชิงสถิติจะใช้โปรโตคอลการส่งแบบซิงโครนัส เช่น HDLC ในการทำงาน โดยที่ HDLC จะประกอบไปด้วยเฟรมข้อมูลมีบิทการควบคุมสำหรับการทำงานในขบวนการมัลติเพล็กซ์ รูป 2.17 แสดงรูปแบบสองรูป แบบที่เป็นไปได้ในแบบแรก มีเพียงแหล่งกำเนิดของข้อมูลแห่งเดียวที่มีข้อมูลส่งไปใน 1 เฟรม โดยแหล่งกำเนิดตัวนั้นถูกกำหนดด้วยแอดเดรส ความยาวของฟิลด์ข้อมูลเปลี่ยนแปลงได้ จุดสิ้นสุดของเฟรมคือจุดสิ้นสุดข้อมูลและที่ท้ายเฟรมก็จะมีการทำเครื่องหมายไว้ให้รู้ว่าสุดเฟรม โครงสร้างแบบนี้ทำงานได้ดีในสภาพการส่งจำนวนไม่มาก แต่จะไม่มีประสิทธิภาพในสภาวะภาระการส่งข้อมูลจำนวนมาก

กรรมวิธีในการปรับปรุงประสิทธิภาพดังกล่าวให้ดีขึ้นก็คือ การยอมให้มีการส่งข้อมูลจากหลายแหล่งเป็นแพคเกจลงไปในเฟรมเดียวกัน ซึ่งก็จะต้องให้มีการกำหนดความยาวข้อมูลของแต่ละ

แหล่งลงไป ดังนั้นส่วนย่อยของเฟรมของ TDM เชิงสถิติก็จะประกอบไปด้วยลำดับของฟิลด์ข้อมูล โดยที่แต่ละตัวจะมีการติดป้ายแอดเดรสและความยาวข้อมูลแจ้งเอาไว้



รูปที่ 2.31 รูปแบบเฟรม TDM

ดังนั้น ตามปกติแล้ว STDM จะเหมาะสำหรับเทอร์มินัลซึ่งทำงานแบบอะซิงโครนัส เพราะในบางขณะอาจมีเทอร์มินัลบางตัวไม่ทำงาน และสำหรับเทอร์มินัลที่ ทำงานนั้นก็มักส่งข้อมูลไม่ถึง 20% ของอัตราการส่งข้อมูลของมันเมื่อเทอร์มินัลถูกใช้งานแบบโต้ตอบ (interactive) ดังนั้น STDM ซึ่งต่อเข้ากับเทอร์มินัล 16 ตัว อาจจะถูกโปรแกรมให้รับข้อมูลจาก 12 เทอร์มินัลเท่านั้นในขณะหนึ่งๆ โดยการส่งสัญญาณ CTS ให้แก่เทอร์มินัลแค่ 12 ตัวในเวลาหนึ่ง และเฉลี่ยเวลาการใช้สายแก่ 12 เทอร์มินัลนี้ตามการร้องขอ และเพื่อให้มัลติเพล็กเซอร์ฝั่งรับสามารถแยกข้อมูลออกจากเฟรมแล้วส่งให้แก่อุปกรณ์ฝั่งรับได้อย่างถูกต้อง ในแต่ละเฟรมจะมี (mapping) (หรือเฮดเดอร์ เช่น แอดเดรส) ซึ่งบ่งบอกว่าข้อมูลส่วนใดเป็นของอุปกรณ์ไหนและมีจำนวนเท่าไร

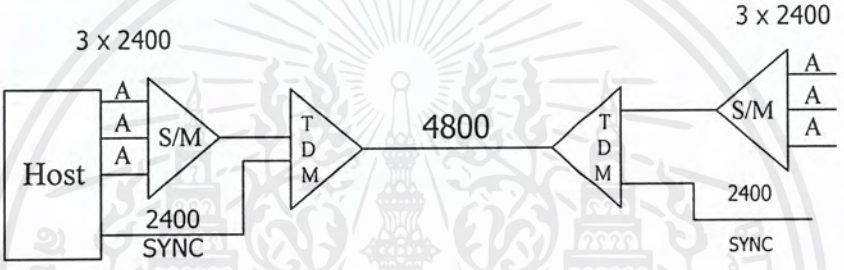
ปกติแล้ว STDM จะมีบัฟเฟอร์สำหรับเทอร์มินัล เพื่อจะเก็บข้อมูลจนกว่าช่องเวลาของเทอร์มินัลนั้นส่งข้อมูลลงไปในสายอย่างมีประสิทธิภาพมากขึ้นถึงแม้เทอร์มินัลที่ใช้จะเป็นแบบอะซิงโครนัสก็ตาม และอาจจะกล่าวได้ว่าประสิทธิภาพในการส่งข้อมูลของ STDM เกินกว่า 100% ดังตัวอย่างต่อไปจะถูกสร้างขึ้น และถ้าหลายเทอร์มินัลมีข้อมูลส่งมามากๆ พร้อมกันจนบัฟเฟอร์รับไม่ไหว STDM อาจจะส่งสัญญาณ X-OFF ไปให้เทอร์มินัลหยุดส่งข้อมูลจนกว่าจะได้รับสัญญาณ X-ON

สำหรับบัฟเฟอร์ซึ่งใช้ในการเก็บข้อมูลในขณะที่อัตราการส่งข้อมูลจากเทอร์มินัลเข้ามายัง STDM มากกว่าอัตราการส่งข้อมูลออกสู่สายนั้น ควรจะโตพอที่จะรับการโหลดสูงสุดได้ในช่วงเวลาหนึ่ง ดังนั้นจากที่ได้กล่าวมาแล้วข้างต้นว่า STDM จะเหมาะสำหรับอะซิงโครนัสเทอร์มินัล ซึ่ง STDM จะช่วยให้มีการนี้

สมมติว่าอัตราการส่งข้อมูลลงในสายส่งความเร็วสูงมีค่าเป็น 9,600 บิตต่อวินาที นั่นคือ 1,200 ตัวอักษร ต่อวินาที และเนื่องจากเฟรมของ STDM มีโอเวอร์เฮดต่างๆปะปนไปด้วย ดังนั้นข้อมูลจริงอาจจะมีเพียง 1,100 ตัวอักษร ซึ่งถูกส่งภายใน 1 วินาที ดังนั้นข้อมูลที่ส่งนั้นรับมาจากอะซิง

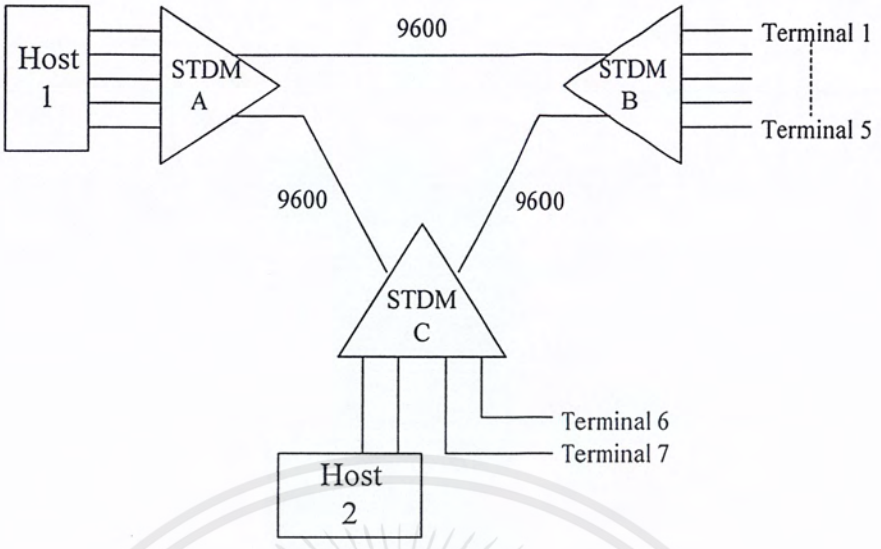
โครนัสเทอร์มินัลที่การส่ง 1 ตัวอักษร จะประกอบด้วย 10 บิต ทำให้อัตราการส่งข้อมูลโดยรวมจากเทอร์มินัลทั้งหมดเท่ากับ 10 × 1,100 =11,000 บิตต่อวินาที ในขณะที่ STDM ส่ง 9,600 บิตต่อวินาที ดังนั้นประสิทธิภาพของการส่งข้อมูลของ STDM เป็น 11,000/9,600 หรือ 114.6% นั่นเอง และนอกจากนั้นยังอาจเพิ่มประสิทธิภาพของ STDM มากขึ้นได้อีก โดยการอัดข้อมูล (data compression) ที่อยู่ในเฟรม

ส่วนซิงโครนัสเทอร์มินัลนั้น ปกติจะมีการส่งข้อมูลอยู่เกือบตลอดเวลา ช่วงเวลาที่สายว่างจะน้อย อีกทั้งการส่งข้อมูลเป็นลักษณะบล็อกซึ่งข้อมูลในบล็อกจะต่อเนื่องกัน ดังนั้น โดยทั่วไปแล้ว STDM ที่ใช้กับซิงโครนัสเทอร์มินัลจะมีราคาแพงกว่า STDM ที่ใช้กับอะซิงโครนัสเทอร์มินัลมาก ดังนั้นหากต้องการจะมัลติเพล็กซ์ข้อมูลจากซิงโครนัสเทอร์มินัลแล้ว อาจจะใช้ TDM ร่วมกับ STDM แบบอะซิงโครนัส ดังแสดงในรูปที่ 2.32



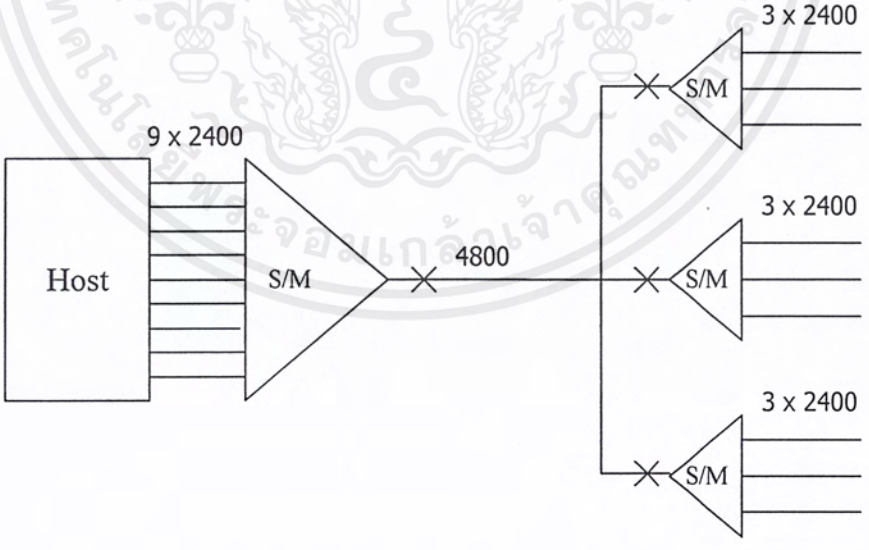
รูปที่ 2.32 ตัวอย่าง STDM ร่วมกับ TDM

สำหรับการใช้งานของ STDM นั้นอาจจะมีหลายรูปแบบ เช่น รูปแบบวงแหวน ดังแสดง ในรูปที่ 2.33 ซึ่งเทอร์มินัลบางตัว เช่น เทอร์มินัล 1,2,3,6,7 อาจจะส่งข้อมูลติดต่อกับโฮสต์ 1 และเทอร์มินัล 4,5 อาจจะติดต่อกับโฮสต์ 2 นอกจากนั้นโฮสต์ 1 และโฮสต์ 2 เองก็อาจจะส่ง ข้อมูลติดต่อกันซึ่งสำหรับรูปแบบนี้มีข้อดีก็คือ ถึงแม้สายที่เชื่อมระหว่าง STDM A และ B จะขาดก็ตาม ข้อมูลจากเทอร์มินัล 1,2,3 ก็อาจจะถูกส่งติดต้อกับโฮสต์ 1 ได้โดยผ่าน STDM C นอกจากการงานในรูปแบบข้างต้นแล้ว STDM อาจจะทำหน้าที่สวิตชิงได้ด้วย เช่น ผู้ที่ใช้เทอร์มินัล 1 อาจจะขอติดต่อเพื่อส่งข้อมูลกับโฮสต์ 1 หรือ โฮสต์ 2 ก็ได้ซึ่งข้อมูลของเทอร์มินัล 1 ก็จะถูกสวิตช์ไปยังปลายทางได้โดย STDM B



รูปที่ 2.33 แสดง STDM ในรูปแบบของวงแหวน

สำหรับรูปที่ 2.34 แสดงการใช้ STDM ในลักษณะของมัลติดรอปไลน์ กล่าวคือ ข้อมูลของเทอร์มินัลจะถูกเก็บเป็นเฟรมใน STDM ที่ติดกับเทอร์มินัล จนกว่าจะได้รับโพลล์จาก STDM ซึ่งติดต่อกับโฮสต์ เฟรมข้อมูลจึงถูกส่งออกไป การต่อแบบนี้ทำให้มีการใช้สายร่วมกันระหว่าง STDM หลายตัว และการใช้งานในรูปแบบนี้จะเห็นได้ว่า จำนวนพอร์ตของโฮสต์จะเท่ากับจำนวนเทอร์มินัลที่ส่งข้อมูลติดต่อกับโฮสต์

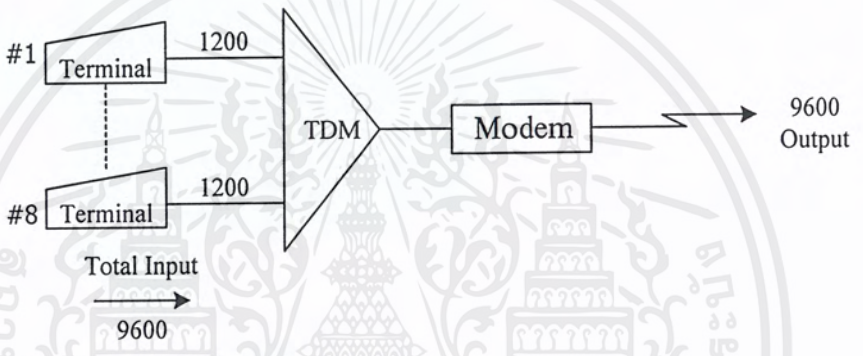


รูป 2.34 STDMในรูปแบบของมัลติดรอปไลน์

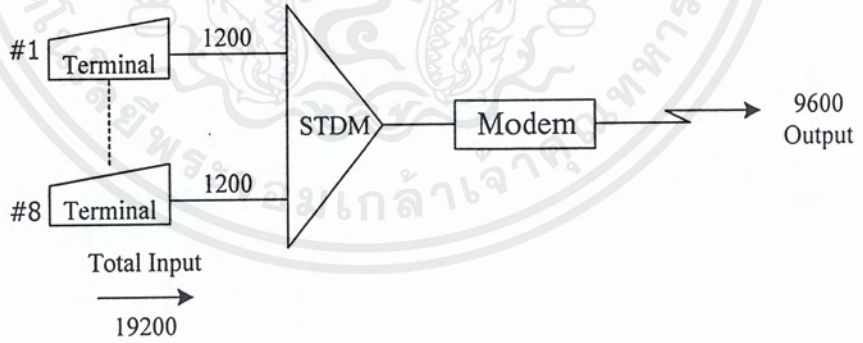
STDM จะมีจำนวนข้อมูลของอะซิงโครนัสมากกว่า TDM ถึง 4 เท่า โดยที่ความจุที่เพิ่มขึ้นนี้เรียกว่า มัลติเพล็กซ์เซอร์ เซอร์วิสเรโช (Multiplexer's Service Ratio) เป็นการเพิ่มความได้เปรียบของเวลาที่สูญเสียไป โดยที่ STDM จะเลื่อนตัวสตาร์ท,สตอป และ พาริตี้บิทจากตัวรับแบบอะซิงโครนัสแต่ละตัวก่อนที่จะส่งและสร้างรูปแบบอักขระขึ้นมาใหม่เมื่อมีการมัลติเพล็กซ์ บางครั้ง STDM จะมีการบีบอัดข้อมูลลงในฟังค์ชั่นที่มีการใช้งาน เพื่อที่จะเพิ่มเซอร์วิส เรโช

2.8.5 เซอร์วิสเรโช (Service Ratio)

เพื่อที่จะเข้าใจคำนิยามของ STDM เซอร์วิส เรโช จะอธิบายรูปที่ 2.18 ซึ่งแสดงระบบ TDM ที่ต่อกับโมเด็มซึ่งมีการทำงานด้วยความเร็ว 9600 บิตต่อวินาที



ก. รูปแบบการ TDM



ข. STDM แบบ 2:1 service ratio

รูปที่ 2.35 STDM service ratio

เมื่อใช้ TDM ในการมัลติเพล็กซ์ข้อมูล อัตราความเร็วรวมของอินพุทจะไม่สามารถมากกว่าอัตราเร็วของเอาท์พุท ดังนั้นอินพุท 8 ช่องสัญญาณ ทำงานที่อัตราเร็ว 1200 บิตต่อวินาที จะให้ผลลัพธ์ออกมา 8*1200 หรือ 9600 บิตต่อวินาที

ถ้าค่าเรโชของเซอร์วิส STDM เท่ากับ 2 โดยเฉลี่ยแล้วจะสามารถใช้ในการรวมอินพุทโดยมีอัตราเร็วรวมมากกว่าเป็น 2 เท่า หรือ 19200 บิตต่อวินาที การคิดเป็น "โดยค่าเฉลี่ย" เนื่องจากในช่วงเวลานานๆ จะมีบางเทอร์มินัลที่ใช้งานและบางเทอร์มินัลจะไม่ได้ใช้งาน ดังนั้นตัวมัลติเพล็กซ์เซอร์จะสามารถรับอัตราเร็วข้อมูลได้เป็น 2 เท่าของอัตราเร็วเอาท์พุทเดิม

ถ้าพิจารณาในกรณีที่แย่ที่สุดในการใช้งาน จะพบว่าถ้าแต่ละเทอร์มินัลเป็นคอมพิวเตอร์ และแต่ละเทอร์มินัลพยายามส่งข้อมูลขนาดใหญ่ออกไปพร้อมๆกัน จะพบว่า STDM ที่มีอัตราความเร็วอินพุทรวมเท่ากับ 19200 บิตต่อวินาที และตัวมัลติเพล็กซ์เซอร์ไม่สามารถจะใช้การเฉลี่ยค่าเวลาได้ เพราะว่าแต่ละเทอร์มินัลจะส่งข้อมูลออกมากอย่างต่อเนื่องพร้อมๆกัน หลังจากมีการกวาด (strip) ของสตาร์ทบิท, สตอปบิท และพาริตี้บิท อัตราเร็วของผลรวมด้านอินพุทจะเกินอัตราเร็วในสายส่ง

ถึงแม้ว่า STDM จะมีพื้นที่ของบัฟเฟอร์ ในขณะที่แต่ละเทอร์มินัลส่งข้อมูล พื้นที่ของบัฟเฟอร์จะเต็มและเกิดการโอเวอร์โฟล ซึ่งทำให้เกิดการสูญเสียข้อมูล ในการป้องกันการสูญเสียข้อมูล STDM จะมีการควบคุมการโอเวอร์โฟล โดยการปิดอุปกรณ์ตัวเดียวหรือหลายตัวก็ได้เมื่อพื้นที่บัฟเฟอร์ใกล้จะเต็ม จากนั้นในขณะที่ข้อมูลจากบัฟเฟอร์ถูกส่งออกไปในสายส่งและสภาวะของการใช้บัฟเฟอร์ลดลงอยู่ในระดับที่แน่นอน STDM จะหยุดการทำงานของโฟลว์คอนโทรล์ และจะให้ทำการต่ออุปกรณ์เข้ากับตัวมัลติเพล็กซ์เซอร์อีก

2.8.6 คอนเซ็นเทรเตอร์ (Concentrator)

แต่เดิมนั้น อุปกรณ์การสื่อสารที่สามารถทำงานแบบ STDM มักจะถูกเรียกว่า *คอนเซ็นเทรเตอร์* กล่าวคือเป็นอุปกรณ์ในการรวบรวมข้อมูลจากเทอร์มินัลซึ่งมีอัตราส่งข้อมูลต่ำ เพื่อส่งลงไปในสายส่งที่มีอัตราส่งสูง แต่ในปัจจุบัน STDM ได้มีการพัฒนาไปมาก และมีอุปกรณ์การสื่อสารหลายชนิดในท้องตลาดซึ่งทำหน้าที่รวบรวมข้อมูล บางอย่างเรียกว่า คอนเซ็นเทรเตอร์ บางอย่างถูกเรียกว่า *อุปกรณ์ควบคุมเครือข่าย (Network Controller)* หรือ *อุปกรณ์ควบคุมการสื่อสาร (communication Controller)* ซึ่งนอกเหนือจากการรวบรวมข้อมูลจากเทอร์มินัลแล้ว ยังมีความสามารถในการควบคุมดูแล (monitor and control) อุปกรณ์อื่นๆของเครือข่ายได้อีกด้วย และอุปกรณ์บางอย่าง เช่น ตัวแปลงโปรโตคอล

2.8.7 บิตซิงโครไนเซชันสำหรับมัลติเพล็กเซอร์

ดังได้กล่าวมาแล้วว่า การส่งข้อมูลแบบอนุกรมนั้น จะต้องมีการทำงานที่สอดคล้องกันระหว่างผู้ส่งและผู้รับ ซึ่งการซิงโครไนซ์กันนี้แบ่งออกได้เป็น บิตซิงโครไนเซชันและคาแรกเตอร์ซิงโครไนเซชัน สำหรับคาแรกเตอร์ซิงโครไนเซชันนั้นใช้เทคนิคของการส่งข้อมูลแบบอะซิงโครนัส

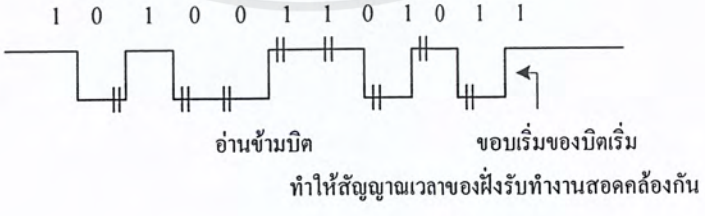
ซึ่งมีการใช้บิตเริ่มและบิตจบช่วยในการส่งข้อมูลแต่ละตัวอักษรและเทคนิคการส่งข้อมูลแบบซิงโครนัสซึ่งจะปะตัวควบคุม เช่น SYN หรือแฟล็กไปทุกๆเฟรมของข้อมูล

สำหรับบิตซิงโครไนเซชันนั้น ถ้าเป็นการส่งระยะใกล้อาจจะมีการส่งสัญญาณเวลาจากอุปกรณ์ส่งไปยังอุปกรณ์รับ ซึ่งเป็นวงจรที่แยกจากสายส่งข้อมูล เพื่อฝั่งรับจะใช้สัญญาณเวลาที่ได้มาในการอ่านข้อมูลที่เข้ามา แต่หากเป็นการส่งข้อมูลระยะไกลแล้ว การส่งสัญญาณเวลาที่แยกไปต่างหากจะเป็นการสิ้นเปลืองสาย ดังนั้นสำหรับการส่งข้อมูลแบบอะซิงโครนัสนั้น ฝั่งรับเมื่อตรวจพบบิตที่เริ่มส่งมา ก็จะทำให้สัญญาณเวลาของตนเองทำงานสอดคล้องกับสัญญาณเวลาของฝั่งส่ง และใช้สัญญาณเวลานั้นในการอ่านสัญญาณข้อมูลจำนวนไม่เกิน 11 บิตเท่านั้น ดังนั้นหากสัญญาณเวลาของฝั่งส่งและฝั่งรับจะแตกต่างกันไปบ้างเล็กน้อย ก็จะไม่กระทบต่อการซิงโครไนซ์กันระหว่างสองฝั่ง การส่งข้อมูลแบบอะซิงโครนัสจึงไม่จำเป็นต้องมีการส่งสัญญาณเวลาจากฝั่งส่งไปยังฝั่งรับ

แต่สำหรับข้อมูลแบบซิงโครนัสนั้น เนื่องจากบล็อกข้อมูลยาวมากถึง 4,000 บิต ดังนั้นความคลาดเคลื่อนของสัญญาณเวลาทั้งสองฝั่งจะทำให้การรับข้อมูลผิดเพี้ยนไปได้ง่าย ดังแสดงในรูปที่ 2.36 (ก) ซึ่งสัญญาณเวลาของฝั่งรับเร็วกว่าฝั่งส่ง ทำให้มีการอ่านบิตเดิมซ้ำ 2 ครั้ง และในรูปที่ 2.36 (ข) ซึ่งสัญญาณเวลาของฝั่งรับช้ากว่าฝั่งส่ง จะทำให้มีการอ่านข้ามบางบิตไป ดังนั้นในการส่งข้อมูลระยะไกลสำหรับซิงโครนัส จะมีการส่งสัญญาณเวลาจากฝั่งส่งไปยังฝั่งรับ เพื่อใช้ในการอ่านข้อมูล ซึ่งสัญญาณเวลานี้จะถูกส่งไปพร้อมๆกับสัญญาณข้อมูล ดังแสดงในรูป 2.37

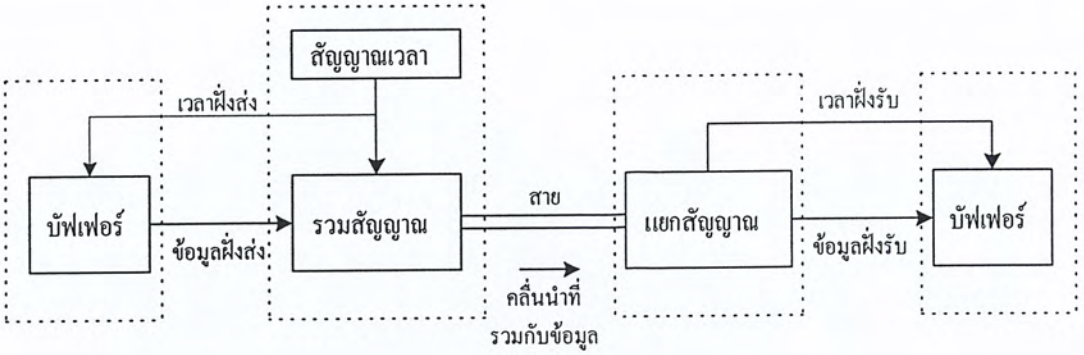


(ก) เมื่อสัญญาณเวลาฝั่งรับเร็วกว่าฝั่งส่ง



(ข) เมื่อสัญญาณเวลาฝั่งรับช้ากว่าฝั่งส่ง

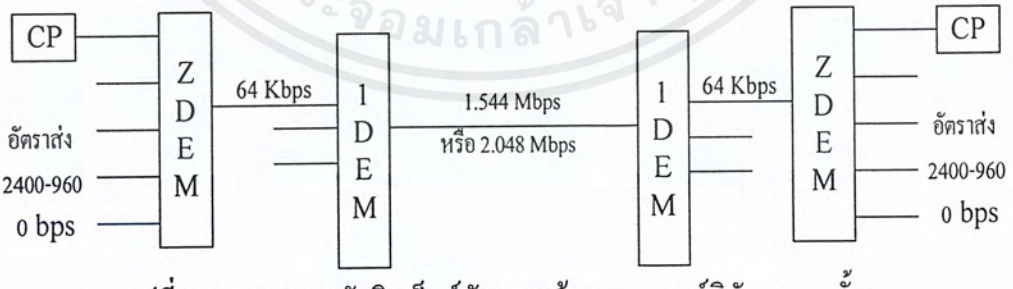
รูปที่ 2.36 แสดงปัญหาของบิตซิงโครไนเซชั่น



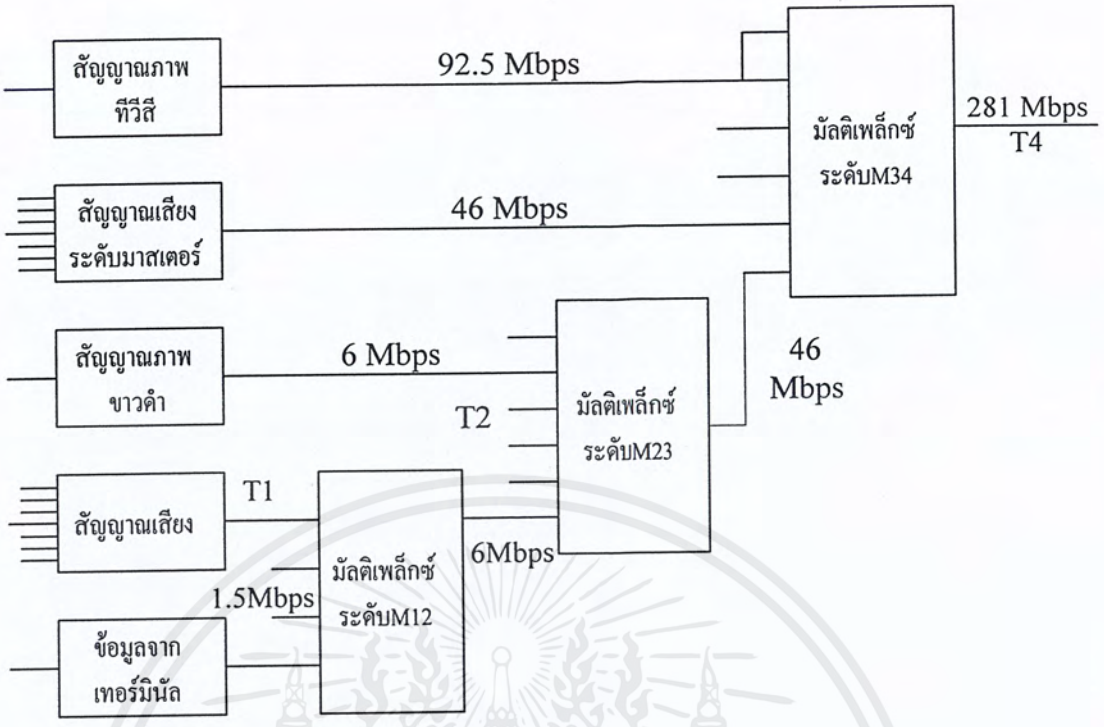
รูปที่ 2.37 แสดงการส่งสัญญาณเวลาพร้อมกับสัญญาณข้อมูล

2.8.8 การผสมสัญญาณหลายรูปแบบ (Multiplexing of Different Signals)

จากรูปที่ 2.38 จะแสดงถึงการมัลติเพล็กซ์แบบ TDM ของสัญญาณข้อมูลจากเทอร์มินัลซึ่งมีอัตราการส่งข้อมูลต่ำ เพื่อให้ส่งได้พร้อมๆกันด้วยอัตราการส่งสูงจนถึงมาตรฐานของ T1 Carrier หรือ CCITT และในรูปที่ 2.39 แสดงถึงการมัลติเพล็กซ์แบบแบ่งตามเวลาของสัญญาณหลายรูปแบบเพื่อให้ส่งออกไปพร้อมๆกันในระบบ AT&T จากรูปจะเห็นได้ว่า ข้อมูลสัญญาณเสียง 24 ช่องสัญญาณจะถูกมัลติเพล็กซ์ไปพร้อมๆกับข้อมูลจากเทอร์มินัลต่างๆ ซึ่งมีอัตราส่งรวมกัน 1.544 เมกะบิตต่อวินาที และนอกจากนี้ สัญญาณที่ถูกมัลติเพล็กซ์แล้ว ยังอาจถูกมัลติเพล็กซ์ในระดับที่สูงขึ้นไปอีกกับสัญญาณดิจิตอลของภาพขาวดำ (picture phone) นอกจากนั้นสัญญาณดิจิตอลของทีวีสีซึ่งมีอัตราการส่งถึง 92.5 เมกะบิตต่อวินาที ก็อาจถูกดีมัลติเพล็กซ์ออกเป็นสัญญาณซึ่งมีอัตราส่ง 46 เมกะบิตต่อวินาที เพื่อที่จะมัลติเพล็กซ์ไปพร้อมกับสัญญาณอื่นๆ ทำให้สัญญาณข้อมูลจากคอมพิวเตอร์ สัญญาณภาพขาวดำและสี สามารถส่งผ่านช่องสัญญาณซึ่งมีแบนด์วิดธ์สูงขึ้นไปพร้อมๆกันได้



รูปที่ 2.38 แสดงการมัลติเพล็กซ์สัญญาณข้อมูลจากเทอร์มินัลหลายๆชั้น



รูปที่ 2.39 แสดงการมัลติเพล็กซ์สัญญาณหลายรูปแบบ

2.9 การเชื่อมต่อกับอุปกรณ์ภายนอกด้วย RS-232

ในรูปที่ 2.40 แสดงไดอะแกรมรูปภาพของการต่อ RS-232 แบบต่างๆที่ไม่ใช่แบบมาตรฐาน จากรูปที่ 2.40 (ก) แสดงการต่อสายเคเบิลแบบฟูลดูเพล็กซ์ ซึ่งถูกต้องตามมาตรฐานทุกอย่าง สำหรับรูปที่ 2.40 (ข) จะแสดงการต่อที่ต่างออกไปจากมาตรฐาน โดยไดอะแกรมแต่ละรูปจะแทนคอนเนคเตอร์ด้วยกล่องสี่เหลี่ยมผืนผ้าขอบมน และได้กำหนดไว้ด้วยว่าคอนเนคเตอร์ตัวใดเป็น DTE หรือ DCE หมายเลขที่เขียนกำกับไว้ในคอนเนคเตอร์แต่ละตัว แทนขาของคอนเนคเตอร์ที่ใช้กำกับเซอร์กิตต่างๆ เซอร์กิตแต่ละตัวแทนด้วยเส้นตรงในแนวนอน พร้อมด้วยตัวอักษรของสัญญาณต่างๆกำกับไว้ ลูกศรที่ปลายเส้นตรงแต่ละเส้นจะแสดงทิศทางการส่ง

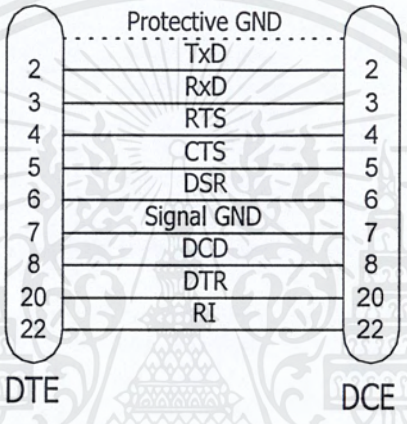
2.9.1 รูปแบบ Three-wire economy

จากไดอะแกรมรูปภาพในรูปที่ 2.40 (ข) สายเคเบิลที่ใช้นั้นใช้เพียง 3 เส้น คือใช้ที่ขา 2,3 และ 7 (ถ้าไม่ต่อขา 1 : กราวน์ โพรเทคทีฟ เซอร์กิต ซึ่งเป็นกรณีใช้งาน) การต่อสายเคเบิลแบบนี้ส่วนมากมักใช้ในการอินเตอร์เฟสกับระบบไมโครคอมพิวเตอร์ แต่มีบางระบบที่เราต้องต่อสายเคเบิลเพิ่มเข้าไปอีก การต่อสายเคเบิลแบบนี้ ใช้ในการส่งข้อมูลแบบฟูลดูเพล็กซ์ ซึ่งเป็นแบบที่ใช้เซอร์กิต

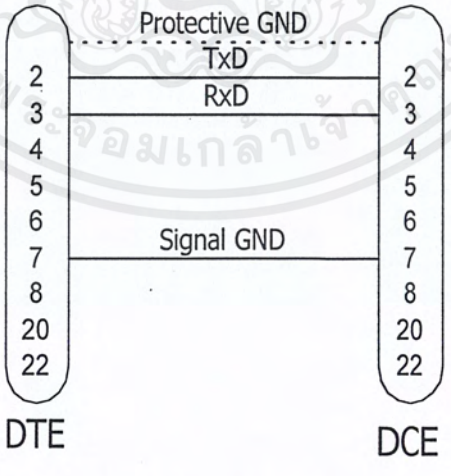
น้อยที่สุด คือใช้ขาเพียง ขา2 :ส่งข้อมูล , ขา3 :รับข้อมูล และ ขา7 : กราวน์ของสัญญาณ (เป็นตัวอ้างอิงของระดับแรงดันที่ขา 2 และ ขา3)

ข้อผิดพลาดที่เกิดขึ้นบ่อยที่สุดของการใช้งานของสายเคเบิลแบบนี้ คือ อุปกรณ์บางตัวของระบบไมโครคอมพิวเตอร์จะใช้เซอร์กิตแบบ ร้องขอการส่ง (Request to send) และ เซอร์กิตแบบเคลียร์เพื่อส่ง (Clear to send) ด้วย ดังนั้นอุปกรณ์เหล่านี้จะส่งข้อมูลได้ก็ต่อเมื่อมันได้รับสัญญาณ "เคลียร์เพื่อส่ง" ก่อน

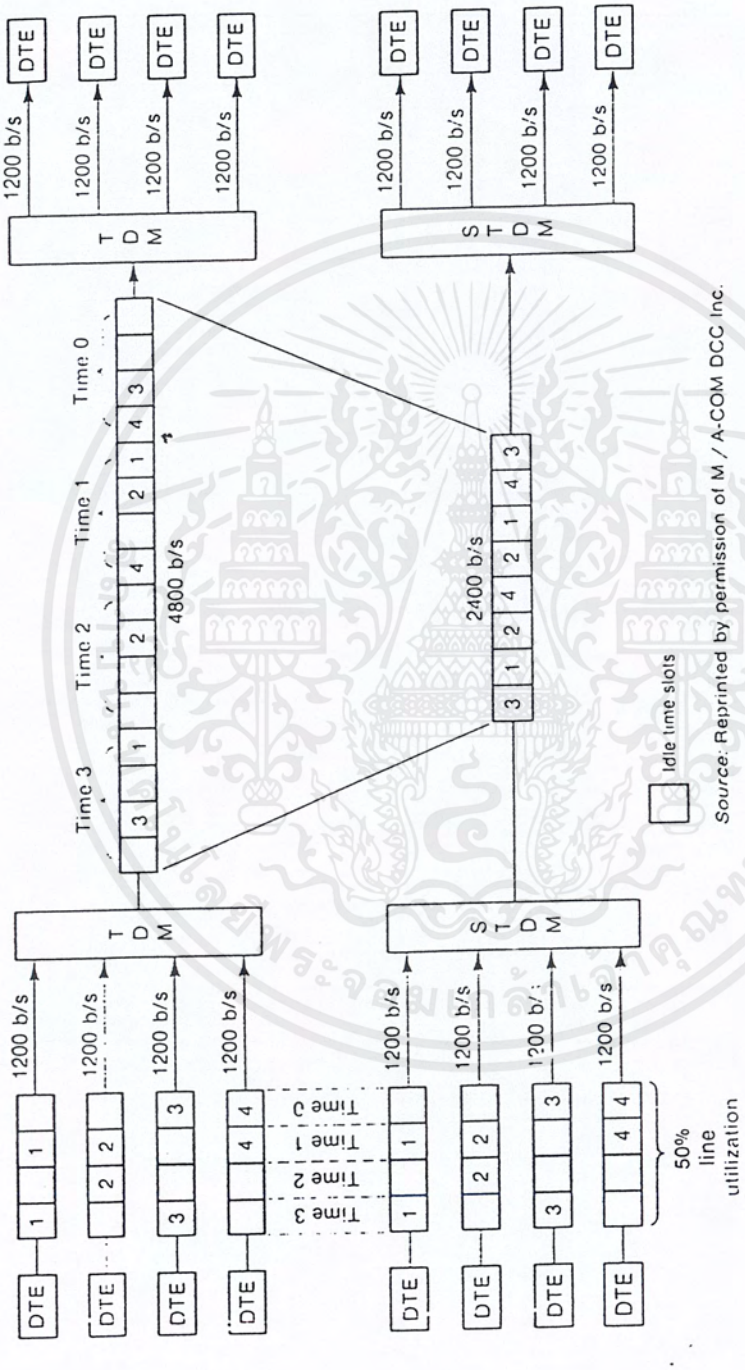
ตัวอย่างของอุปกรณ์ที่ใช้งานในลักษณะนี้ ได้แก่ IC พวก USART (Universal - Synchronous/Asynchronous Receiver/Transmitter) ซึ่งใช้เป็นพอร์ตอินพุตและเอาท์พุตแบบอนุกรม



รูปที่ 2.40 (ก) ระบบสายเคเบิลส่งข้อมูลฟูลดูเพล็กซ์ตามมาตรฐาน



รูปที่ 2.40 (ข) สายเคเบิลแบบ three-wire economy

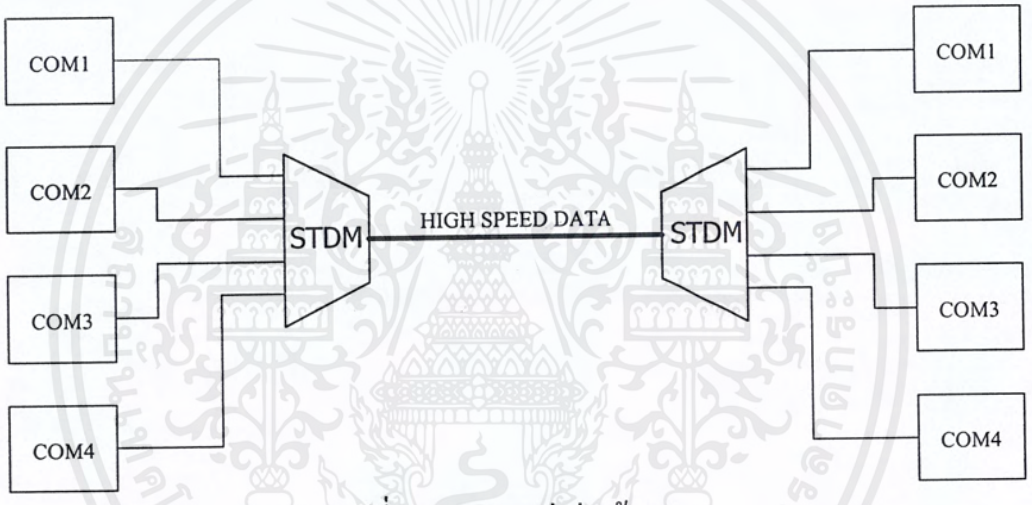


รูปที่ 2.41 แสดงการเปรียบเทียบระหว่าง TDM กับ STDM

บทที่3
การคำนวณและการสร้าง

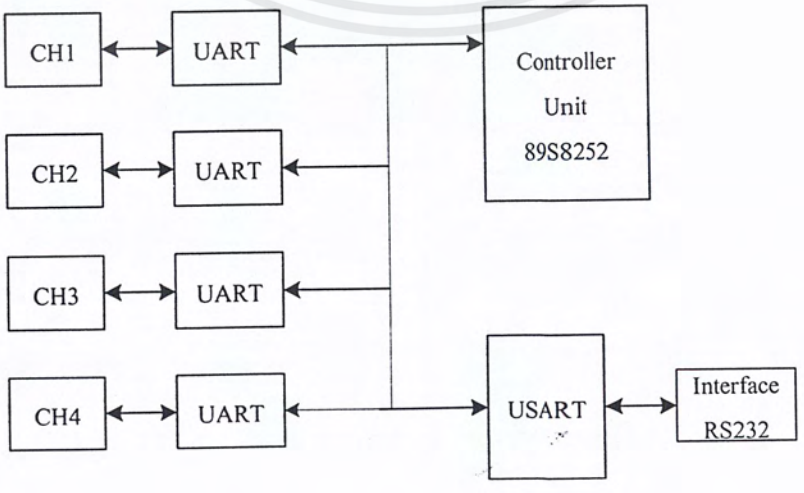
3.1 บล็อกไดอะแกรมแสดงการส่งผ่านข้อมูล

การออกแบบ ซีโร่ มัลติเพล็กซ์เซอร์ จะใช้เทคนิคการมัลติเพล็กซ์แบบแบ่งเวลาเชิงสถิติ (Statistical Time Division Multiplex:STDM) โดยภาคส่งจะนำ 4 ช่องสัญญาณมามัลติเพล็กซ์รวมกันแล้วส่งออกไปในช่องสัญญาณ 1 ช่อง ส่วนทางภาครับจะทำ ดีมัลติเพล็กซ์ (Demultiplex) สัญญาณกลับเป็น 4 ช่องสัญญาณตามเดิมโดยในส่วนของไฮสปีดนั้น เป็นการจัดการกับการเชื่อมต่อข้อมูลแบบซิงโครนัสซึ่งจัดการแบบ คาแรกเตอร์ โอเรียนต์ (character oriented) หรือ ไบต์



รูปที่ 3.1 แสดงการส่งผ่านข้อมูล

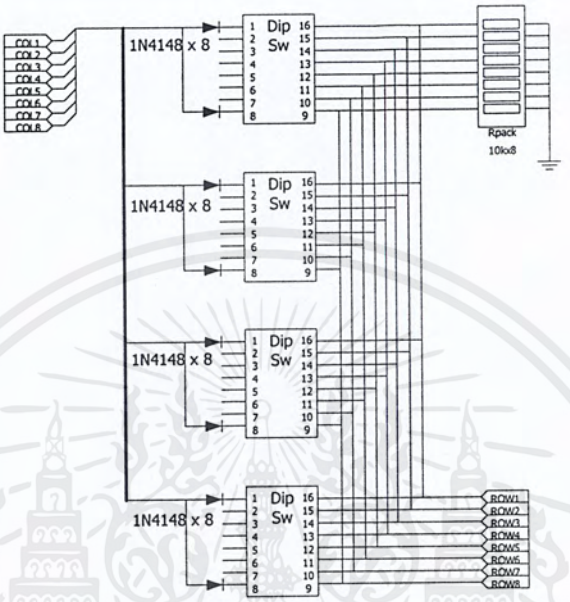
3.2 บล็อกไดอะแกรมแสดงการมัลติเพล็กซ์สัญญาณ



รูปที่ 3.2 บล็อกไดอะแกรมของชุดรับส่งสัญญาณ

3.3 ชุด ดิพสวิตซ์ (Dip Switch)

ทำงานโดยจะต้องเซตค่าดิพสวิตซ์ ให้ตรงกับบิตเรต (Bit Rate) ที่ส่งออกมาจากคอมพิวเตอร์แต่ละช่องสัญญาณก่อนจะใช้การมัลติเพล็กซ์แบบแบ่งเวลาเชิงสถิติ



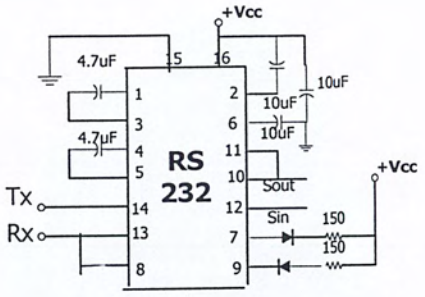
รูปที่ 3.3 แสดงวงจรดิพสวิตช์สำหรับ UART

3.4 ส่วนการเชื่อมต่อกับอุปกรณ์ปลายทางข้อมูล (DTE)

การเชื่อมต่อข้อมูลกับอุปกรณ์ปลายทางจะกระทำโดยผ่านพอร์ตอนุกรม RS232 โดยที่ขาสัญญาณเอาท์พุทจะมีขาที่ทำหน้าที่เป็นเอาท์พุต ได้แก่ DTR,RTS และ TxD โดยที่รีจิสเตอร์ที่ทำหน้าที่ควบคุมขาเหล่านี้ คือรีจิสเตอร์ควบคุมโมเด็ม (MCR) โดยมีแอดเดรสอยู่ถัดจากรีจิสเตอร์หลักของพอร์ตอนุกรม 4 ตำแหน่ง รีจิสเตอร์ควบคุมการทำงานของขา DTR จะอยู่ที่ตำแหน่งบิต 0 ส่วนขา RTS จะอยู่ที่ตำแหน่งบิต 1 ส่วนขา TxD เป็นขาปกติในการส่งข้อมูล ดังนั้นจึงมีแอดเดรสอยู่ที่แอดเดรสอยู่ที่แอดเดรสของรีจิสเตอร์หลัก

ระดับแรงดันที่ใช้งานสำหรับพอร์ตอนุกรม มีมาตรฐานระบุช่วงระดับแรงดันสำหรับการทำงานของพอร์ตอนุกรมไว้ว่า ที่ลอจิก 0 จะมีระดับสัญญาณ +3 ถึง +15 ส่วนลอจิก 1 จะมีระดับสัญญาณ –3 ถึง –15 ระดับสัญญาณนี้ไม่สามารถนำขาเอาท์พุตใดๆต่อเข้ากับลอจิกเพื่อใช้งานได้โดยตรง จะต้องผ่านวงจรเพื่อแปลงระดับแรงดันเสียก่อน

RS232 จะทำหน้าที่แปลงระดับแรงดันให้อยู่ในระดับทีทีแอล โดยลอจิก 0 ซึ่งเดิมมีระดับสัญญาณ +3 ถึง +15 จะแปลงเป็น 0 โวลท์ และส่วนลอจิก 1 ซึ่งมีสัญญาณ –3 ถึง –15 จะแปลงเป็น +5 โวล์ท ทั้งนี้เพื่อให้สามารถเชื่อมต่อกับอุปกรณ์ดิจิตอลอื่นที่ใช้ระดับแรงดันทีทีแอลได้



รูปที่ 3.4 รูปวงจร RS232

สาเหตุที่เราไม่ใช้การแทนลอจิกแบบทีทีแอลในการส่งข้อมูลที่ระยะไกลๆ ก็เป็นเพราะว่า สถานะลอจิกแบบทีทีแอลนั้น จะถูกรบกวนจากสัญญาณรบกวนได้ง่าย และนอกจากนี้ยังมีปัญหาในเรื่องของส่งในระยะทางไกลๆอีกด้วย

Stat Mux	Signal Voltage	
	-25 V < V1 < -3V	3V < V1 <25V
BSC	1	0
Signal Condition	MARK	SPACE
Function	OFF	ON

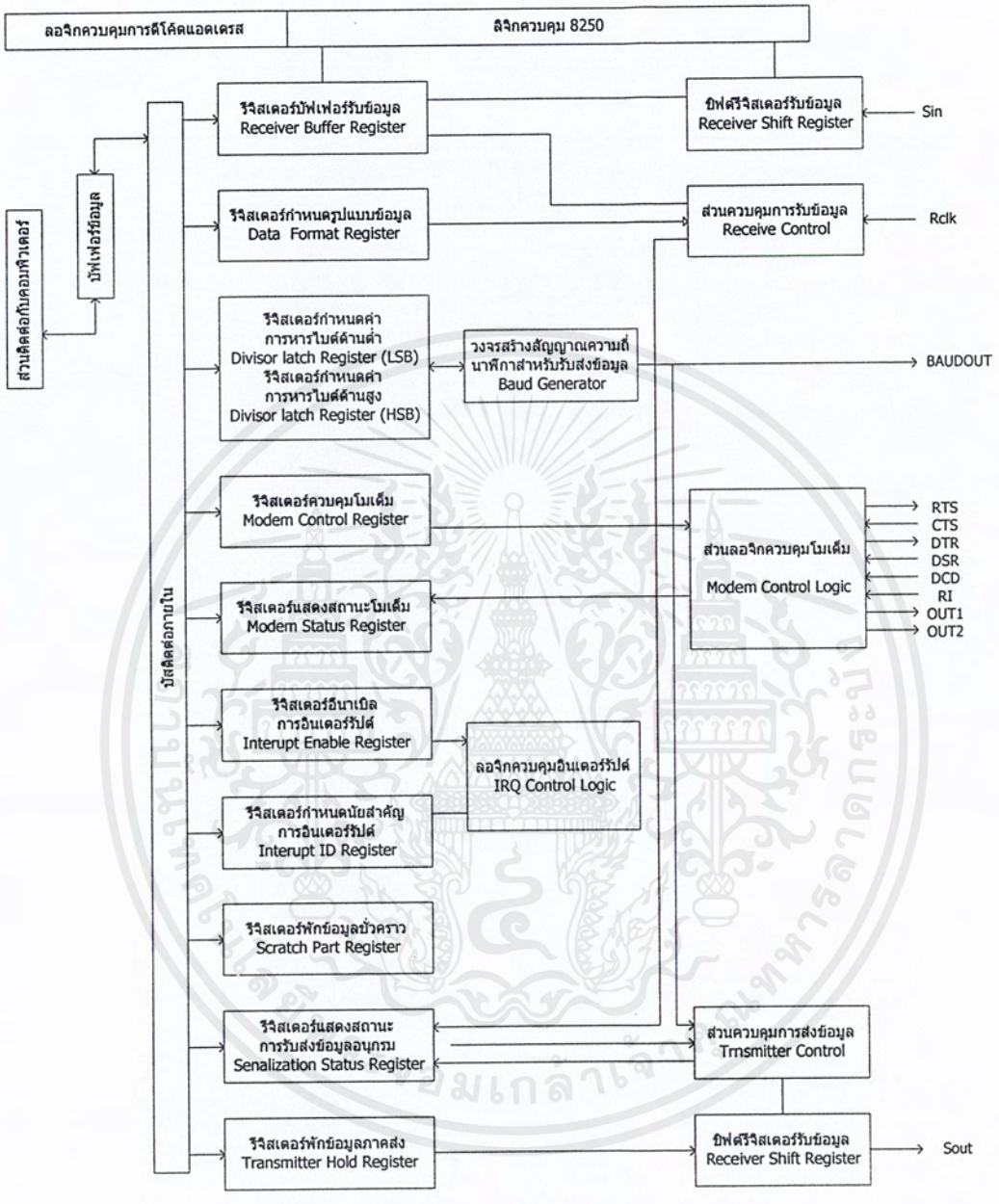
ตารางที่ 3.1 แสดงระดับสัญญาณของสถานะลอจิกแบบ ทีทีแอล

จะเห็นได้ว่า RS232 ยอมให้มีมาร์จินของสัญญาณรบกวนได้ไม่เกิน 2 โวลท์ ซึ่งตามข้อกำหนดระยะทางที่สามารถใช้ทำการสื่อสารข้อมูลจะได้ไม่เกิน 50 ฟุต ตามค่ามาตรฐานที่กำหนด

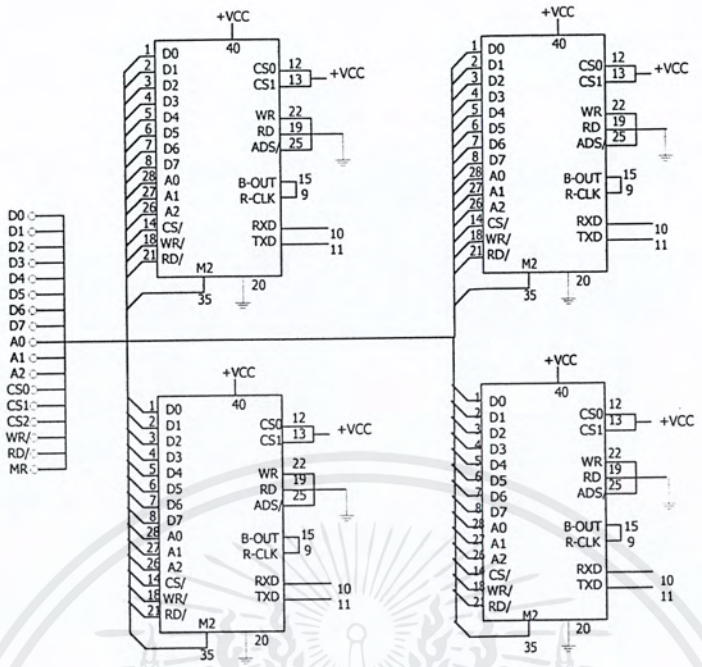
3.5 ส่วนของการรับส่งข้อมูลความเร็วต่ำแบบอะซิงโครนัส (Asynchronous)

ในโครงงานนี้จะใช้ไอซี GM16C550 ทำหน้าที่ในการเปลี่ยนการส่งข้อมูลอนุกรมเป็นการส่งข้อมูลแบบขนานเพื่อส่งให้ชุดไมโครคอนโทรลเลอร์ และเปลี่ยนข้อมูลขนานให้เป็นข้อมูลอนุกรม เพื่อส่งให้เทอร์มินัลที่อยู่ปลายทาง

รายละเอียดของ UART GM16C550 มีดังนี้



รูปที่ 3.5 โครงสร้างภายในของ 16550



รูปที่ 3.6 รูปวงจร 16550

ขั้นตอนในการจัดเตรียม GM16C550 ในการแลกเปลี่ยนข้อมูล

1. ทำการกำหนดแอดเดรสเพื่อเลือกไปยังรีจิสเตอร์ LCR แล้วทำการโค้ดข้อมูลให้บิท DLAB มีค่าเป็น 1
2. ทำการแอดเดรสเพื่อเลือกไปยังรีจิสเตอร์ DLL และ DLM เพื่อทำการกำหนดความเร็วในการส่งข้อมูล ซึ่งอัตราข้อมูลมีความสัมพันธ์ระหว่างข้อมูลใน DLL และ DLM

ตัวอย่างเช่น ถ้าต้องการติดต่อด้วยอัตราข้อมูล 1200 บอด และใช้ออสซิลเลเตอร์นี้ให้ความถี่สัญญาณนาฬิกาคือ 1.8432 MHz

ค่าเลขฐาน 10 ใน DLL กับ DLM = 1843200/(1200x16)

= 96

= 60 H

ซึ่งแปลงเป็นข้อมูลไบนารีแล้วเก็บในรีจิสเตอร์ DLL และ DLM ได้ดังนี้

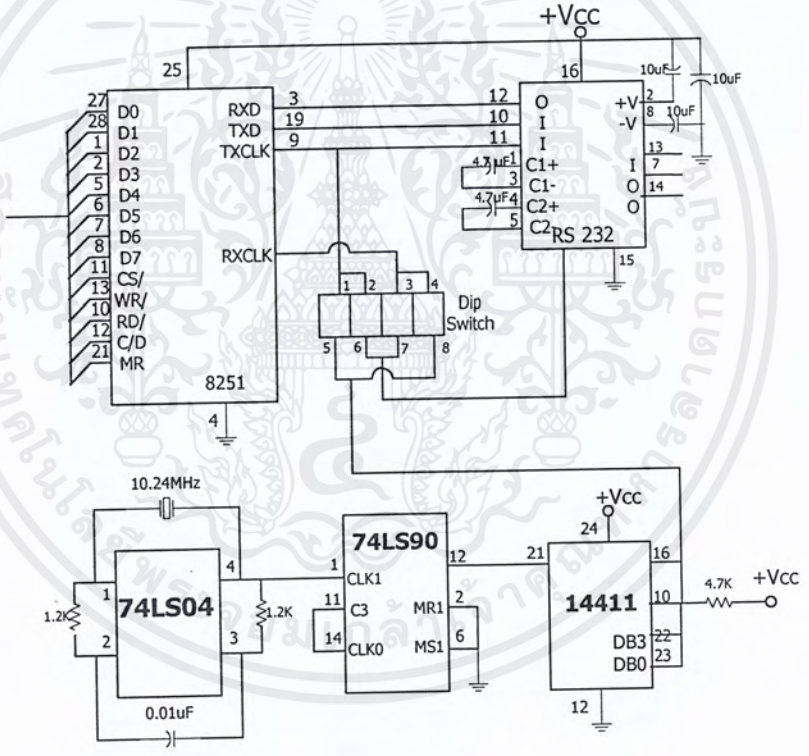
เมื่อขาสัญญาณเป็นแบบ แอกทีฟ โลวว์ จะเขียนข้อมูลลงใน DLL และ DLM รีจิสเตอร์

3. ต้องการส่งข้อมูลก็จะโค้ดข้อมูลลงไปที่ D0 – D7 เมื่อขาสัญญาณ $\overline{WR}$ แอกทีฟ โลวข้อมูลก็จะถูกทำการกำหนดแอดเดรสเพื่อทำการเลือกรีจิสเตอร์ LCR, ทำการกำหนดข้อมูลโดยให้ DLAB มีค่าเป็น 0 และทำการกำหนดการรับส่งข้อมูล
4. ทางด้านรับ เมื่อส่งไปยังฝ่ายรับ
5. ด้านฝ่ายรับก็จะโค๊ดแอดเดรสเพื่อเลือกรีจิสเตอร์ LSR และตรวจสอบดูว่าบิท D0 ใน LSR เป็น 1 หรือ 0 (เป็น 1 เมื่อตัวรับสามารถรับข้อมูลได้ และเป็น 0 เมื่อข้อมูลถูกอ่านไปจากบัฟ-เฟอร์) เมื่อขาสัญญาณ $\overline{RD}$ แอกทีฟโลว์

3.6 กระบวนการมัลติเพล็กซ์

สัญญาณดิจิตอลทั้ง 4 ช่องสัญญาณ จะถูกส่งผ่านสายส่งเพียงเส้นเดียว โดย STAT MUX จะให้ความสามารถในการส่งข้อมูลแก่เทอร์มินอลที่มีข้อมูลเท่านั้น โดยจะสามารถรับความเร็วของแต่ละช่องสัญญาณได้แตกต่างกัน ตั้งแต่ 300 , 600 , 1200 , 2400 , 4800 , 7200 , 9600 ,19200 บิตต่อวินาที เมื่อทำการมัลติเพล็กซ์ทั้ง 4 ช่องสัญญาณ เพื่อส่งรวมกันไปด้วยความเร็ว 64 กิโลบิตต่อวินาที โดยช่วงเวลาในการส่งข้อมูลของแต่ละช่องสัญญาณจะถูกจัดการแบบคาแรกเตอร์ โอเรียนต์ ในรูปแบบของการใช้ตัวอักษรบ่งบอกต้นเฟรมและท้ายเฟรม โดยสัญญาณซิงโครนัสถูกสร้างขึ้นจากวงจรสร้างสัญญาณนาฬิกา โดยมีสัญญาณคาแรกเตอร์ ซิงโครนัส (character synchronous) เป็นตัวกำหนดขอบเขตของเฟรมข้อมูล

เราจะสร้างสัญญาณนาฬิกา 64 กิโลบิตต่อวินาที และ คาแรกเตอร์ ซิงโครนัสป้อนให้ชุด USART โดยวงจรสร้างสัญญาณนาฬิกาและซิงโครนัส แสดงดังในรูปที่ 3.7



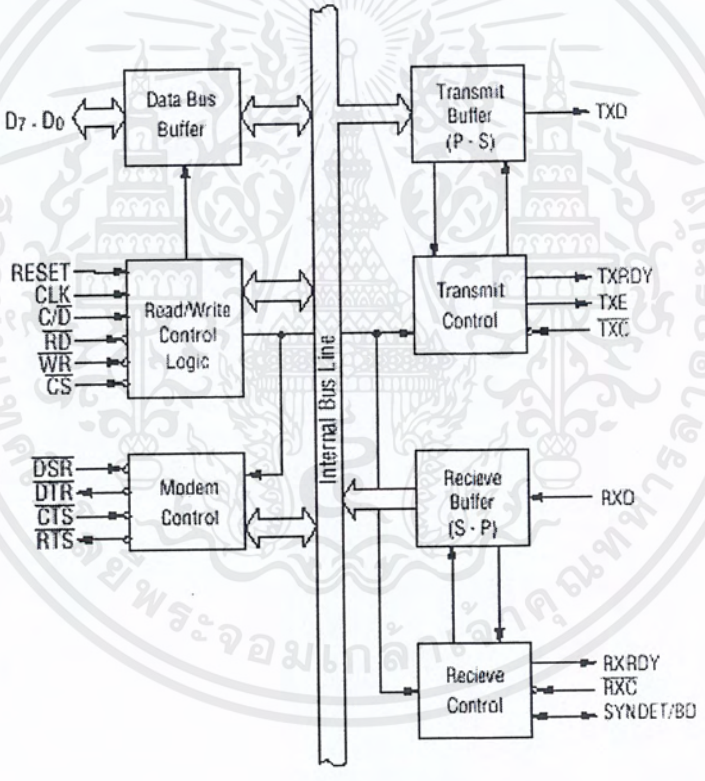
รูปที่ 3.7 แสดงวงจรสร้างสัญญาณนาฬิกาและซิงโครนัส

การออกแบบในที่นี้ใช้ MC14411 เป็นบิตเรต เจนเนอร์เรเตอร์ (Bit Rate Generator) สำหรับเบอร์ 8251 เนื่องจาก MC14411 นี้สามารถให้กำเนิดสัญญาณนาฬิกาอัตราบอดที่แตกต่างกันได้ 14 ตัว ซึ่งในโครงงานนี้ต้องการจะสร้างความถี่ขึ้นมาเท่ากับ 64 กิโลบิตต่อวินาที เพื่อที่จะใช้สำหรับเป็นความถี่อ้างอิงในการส่งผ่านข้อมูลแบบซิงโครนัส ดังนั้นจึงต้อนำความถี่ที่จะหารแล้วลงตัวที่ 64 กิโลบิตต่อวินาที ความถี่ที่จะต้องสร้างขึ้นมาคือ ความถี่ 1.024 เมกะเฮิรตซ์ แล้วมาเข้าวงจรหารสิบด้วย IC เบอร์ 7490 จากนั้นจึงนำความถี่ 1.024 เมกะเฮิรตซ์ต่อเข้ากับ Xin (ขา21) และ Xout (ขา20) และทำการหารความถี่โดย

วงจรภายใน และ MC14411 ยังมีขาสัญญาณสำหรับการเลือกอัตราบิต หรือความถี่ในการส่งผ่านข้อมูล ด้วย คือสัญญาณ RSA (ขา23) และขาสัญญาณ RSB (ขา 22) เพื่อใช้เลือกสัญญาณนาฬิกาอ้างอิง สำหรับการออกแบบและเลือกอัตราความถี่เป็น 64 เท่าและเลือกเอาท์พุตที่ขา F6 ก็จะได้ความถี่ 64 กิโลเฮิรตซ์ ตามต้องการ โดยการต่อขา RSA และ RSB เข้าให้มีสถานะเป็น "1"

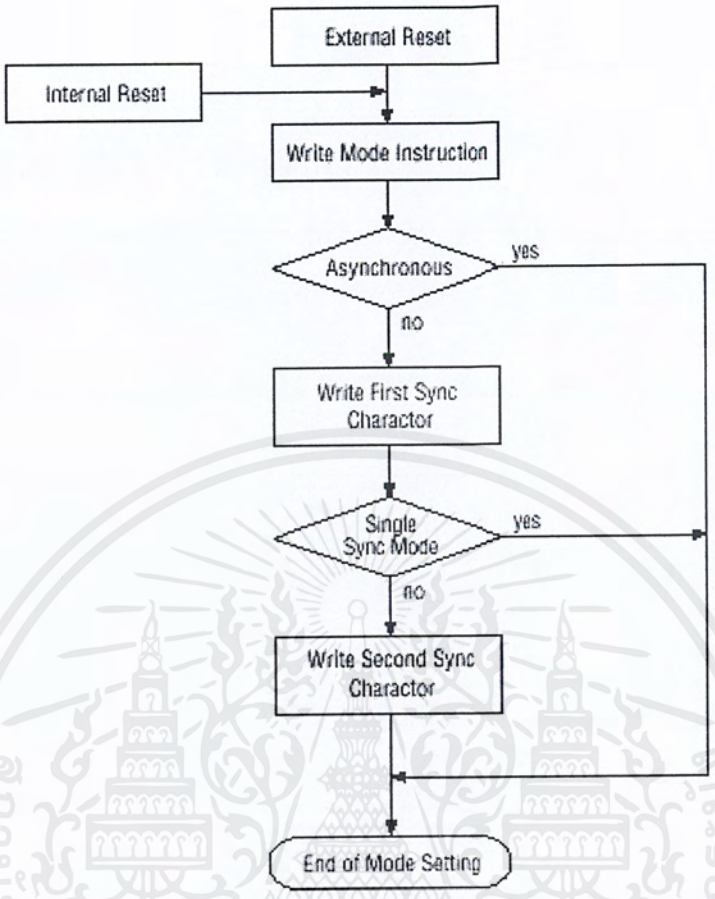
3.7 ส่วนของการรับส่งข้อมูลความเร็วสูงแบบซิงโครนัส

ชิพ 8251 A ซึ่งเป็น USART (Universal Synchronous/Asynchronous Receiver/Transmitter) เพื่อทำการส่งสัญญาณจากช่องสัญญาณความเร็วต่ำ (UART) ไปยังช่องสัญญาณความเร็วสูง (64 kbps) ภายใต้การควบคุมของไมโครคอนโทรลเลอร์ โครงสร้างภายใน 8251 A แสดงได้ดังรูปที่ 3.8



รูปที่ 3.8 แสดงโครงสร้างภายใน 8251 A

8251 A จะรับสัญญาณคาแรคเตอร์ ที่ถูกมัลติเพล็กซ์แล้วเข้ามาทาง D0 – D7 ข้อมูลที่เป็นขนานเหล่านี้จะถูกทำให้เป็นอนุกรม 8251 A จะถูกจัดการโดยซอฟต์แวร์ที่มีลำดับการเซตค่าที่แน่นอน ดังรูปที่ 3.9 จะแสดงลำดับการโปรแกรม 8251 A



รูปที่ 3.9 โฟลว์ชาร์ทแสดงการทำงานและลำดับการโปรแกรม

ลำดับขั้นตอนในการโปรแกรม 8251 มีดังนี้

1. รีเซต
2. โหมด อินสตรัคชั่น (Mode Instruction)
3. คอมมานด์ อินสตรัคชั่น (Command Instruction)

การนำข้อมูลออกหรือการอ่านจะถูกควบคุมโดยชุดไมโครคอนโทรลเลอร์ทำการรีเซต เมื่อเริ่มต้นลำดับการกระตุ้น คำสั่งต่อมาจะถูกแปลความให้เป็นโหมด อินสตรัคชั่น และหลังจากเขียนโหมด อินสตรัคชั่น แล้วก็จะเป็นการเขียนคอมมานด์ อินสตรัคชั่น

ซีเรียลไมโครพอร์ต (Serial Microport) จะประกอบด้วย

- TxD ใช้สำหรับส่งข้อมูล
- RxD ใช้สำหรับรับข้อมูล
- CS ถ้าได้รับสัญญาณ low จะหมายถึงอนุญาตให้ซีเรียลพอร์ตทำงานได้ โดยสัญญาณ RD, WR, CS, DTR, CTS, RTS, TxC, RxC จะแอกทีฟที่สัญญาณ 0 และสำหรับ $C/\overline{D}$ ถ้าเป็น "0" จะหมายถึงสัญญาณข้อมูล ถ้าเป็น "1" จะหมายถึงสัญญาณควบคุมสัญญาณที่ต่อกับ CPU
- คำสั่งข้อมูลจะเป็น 8 บิต D7-D0 ซึ่งแต่ละบิตจะมีความหมาย

- TxRDY พร้อมจะรับข้อมูลไปส่ง O/P
- RxRDY ข้อมูลพร้อมจะส่งไปยังบัสข้อมูล O/P
- TxEMPTY บัฟเฟอร์ข้อมูลว่า O/P
- TxC สัญญาณนาฬิกากำหนดการส่งข้อมูล
- RxC สัญญาณนาฬิการกำหนดการรับข้อมูล
- SYNDET/BD จะมีค่าเป็นไฮเมื่อตัวอักษร syn ถูกตรวจพบขณะที่ทำงานอยู่ในโหมดซิงโครนัส

สัญญาณควบคุมโมเดมมี 4 ตัว ประกอบไปด้วย $\overline{DTR}, \overline{RTS}$ ที่เป็นเอาท์พุต และ $\overline{DSR}$ กับ $\overline{CTS}$ ที่เป็นอินพุต

3.8 ไมโครคอนโทรลเลอร์

ในโครงงานนี้ใช้ไอซีเบอร์ AT89S8252 เป็นไมโครคอนโทรลเลอร์ขนาด 8 บิต และเป็นไอซีประเภท CMOS ซึ่งเป็นไอซีที่มีประสิทธิภาพสูง กินไฟต่ำ มีหน่วยความจำแบบแฟลชบรรจุอยู่ภายในตัวขนาด 8 กิโลไบต์ ทั้งยังทำงานร่วมและทดแทนไมโครคอนโทรลเลอร์ตระกูล MCS-51 ได้ทั้งคำสั่งและการจัดเรียงขา

โปรแกรมการทำงานของทั้งด้านรับและด้านส่ง มีแผนผังการทำงานดังแสดงข้างล่างและมีคำอธิบายขั้นตอนการทำงานได้ดังนี้

<u>ภาคส่ง</u>

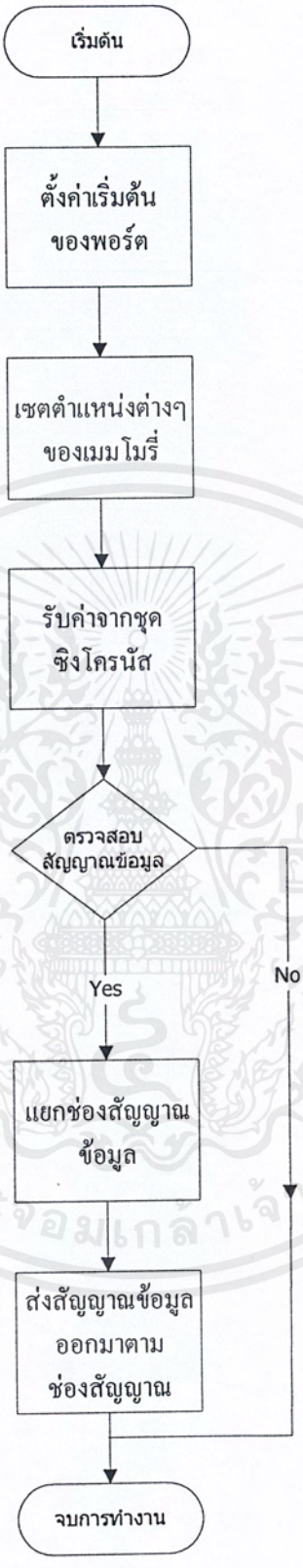
สัญญาณอินพุตจากคอมพิวเตอร์จะส่งออกมาทางพอร์ตอนุกรม เข้ามาที่ชุดฮาร์ดแวร์อินเตอร์เฟส MAX 232 โดยทำการเซตค่าบิตเรตให้ตรงกับค่า ดิพสวิตซ์ ที่ชุด STAT MUX สัญญาณเอาท์พุตของ MAX 232 จะถูกนำเข้ามาที่ขา Sin (ขา 10) ของ UART (GM16C550) ซึ่ง GM16C550 จะทำการแปลงสัญญาณอนุกรมเป็นสัญญาณขนานออกที่ขา D7 – D0 โดยชุดไมโครคอนโทรลเลอร์จะทำการวนรับสัญญาณ (Polling) ทีละตัว จนครบทุกช่องสัญญาณ โดยจะรับเฉพาะ UART ที่มีสัญญาณอินพุตส่งมาจากคอมพิวเตอร์เท่านั้น แล้วนำข้อมูลทั้ง 4 ช่องสัญญาณ ทำการมัลติเพล็กซ์โดยชุดไมโครคอนโทรลเลอร์และจัดเฟรมให้กับข้อมูลเหล่านั้น แล้วส่งผ่านออกทาง 8251 A แบบซิงโครนัสด้วยความเร็ว 64 กิโลบิตต่อวินาที ผ่านทาง MAX 232



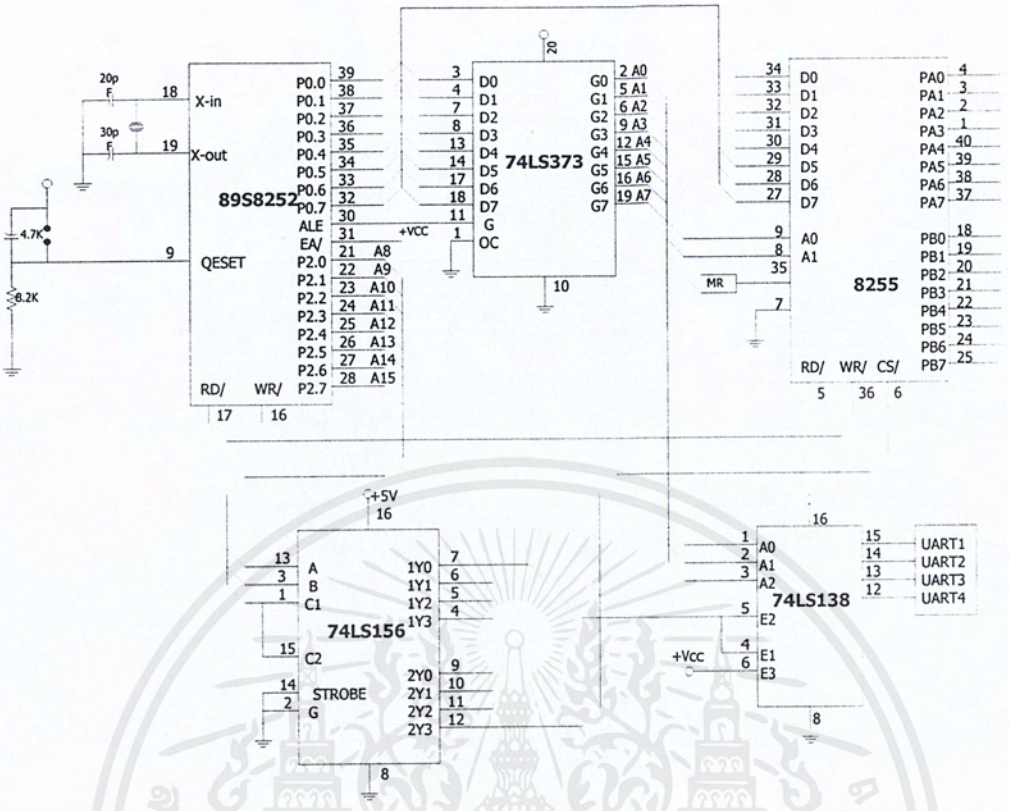
รูปที่ 3.10 แผนผังระบบโปรแกรมทางด้านส่ง

ภาครับ

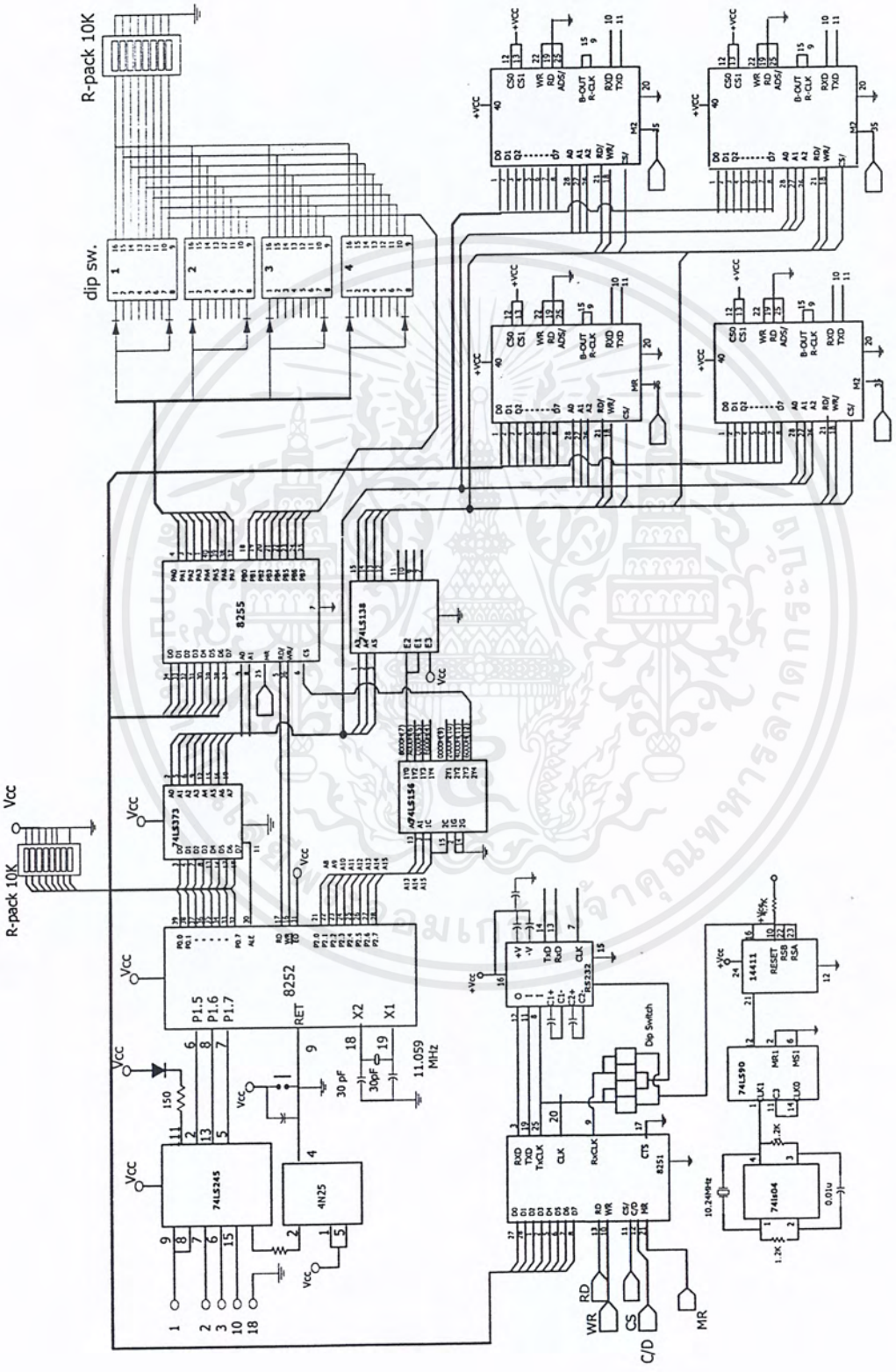
สัญญาณจากภาคส่งจะถูกส่งมายังภาครับ ผ่าน MAX 232 เพื่อแปลงสัญญาณจากระดับ RS-232 เป็นระดับสัญญาณ TTL ผ่านทาง 8251 A เข้ามาที่ชุดไมโครคอนโทรลเลอร์ทำการดีมัลติเพล็กซ์สัญญาณส่งผ่าน UART 16C550 ไปยังคอมพิวเตอร์ภาครับ



รูปที่ 3.11 แผนผังระบบโปรแกรมด้านรับ



รูปที่ 3.12 วงจรไมโครคอนโทรลเลอร์



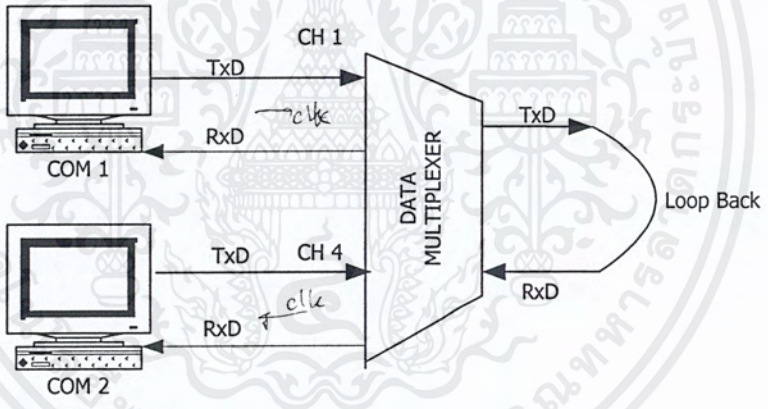
รูปที่ 3.13 รูปวงจรรวม

บทที่ 4
การทดลองและผลการทดลอง

จะกล่าวถึงการทดลองการทำงานของเครื่องมัลติเพล็กซ์ข้อมูล ว่าสามารถที่จะทำงานได้ตามคุณลักษณะเฉพาะที่ได้กำหนดไว้หรือไม่

4.1 การทดสอบลูปกลับ (Loop Back Test)

การทดสอบลูปกลับนี้ เป็นการทดลองเพื่อต้องการที่จะทราบว่าข้อมูลจากเครื่องคอมพิวเตอร์ที่ส่งไปยังเครื่องมัลติเพล็กซ์ข้อมูลนั้น สามารถที่จะรับข้อมูลกลับมายังคอมพิวเตอร์ตัวเดิมแล้วมีข้อผิดพลาดอย่างไรบ้าง โดยการทดลองนี้จะทำการลูปข้อมูลกลับที่ช่องสัญญาณความเร็วสูง (64 กิโลบิตต่อวินาที) ซึ่งจะเป็นการตัดปัญหาการเกิดข้อผิดพลาดอันเนื่องมาจากตัวกลางในการสื่อสารข้อมูลระหว่างคู่สถานี ดังแสดงในรูปที่ 4.1



รูปที่ 4.1 บล็อกไดอะแกรมการทดสอบลูปกลับ

ขั้นตอนการทดลอง

1. นำเครื่องคอมพิวเตอร์ 2 เครื่องมาเชื่อมต่อกันเข้ากับเครื่องมัลติเพล็กซ์ข้อมูลที่ช่อง 1 และช่องที่ 8 ดังแสดงในรูปที่ 4.1
2. ทำการลูปข้อมูลกลับที่ช่องสัญญาณข้อมูลความเร็วสูง
3. เลือกอัตราบิตเรทที่ใช้ในการส่งข้อมูล โดยเลือกที่ดิพสวิตซ์ทางด้านหลังเครื่อง โดยเลือกให้ตรงกับความเร็วที่คอมพิวเตอร์จะรับหรือส่งข้อมูล
4. ทำการส่งข้อมูลจากเครื่องคอมพิวเตอร์ทั้ง 2 เครื่อง โดยจะต้องเลือกอัตราบิตเรทในการส่งและรับข้อมูลให้ตรงกับที่เลือกดิพสวิตซ์ของช่องสัญญาณที่ต่อไว้

5. ทำการกดคีย์ตัวอักษรใดๆเพื่อส่งข้อมูล ซึ่งจะปรากฏให้เห็นบนจอภาพในส่วน LOCAL และให้สังเกตว่าตัวอักษรที่รับกลับเข้ามาจากการทำลูปกลับ ซึ่งจะปรากฏให้เห็นในหน้าต่างส่วนของ REMOTE มีข้อผิดพลาดหรือไม่ เช่น ตัวอักษรที่รับกลับเข้ามามีจำนวนไม่ครบตามที่ส่งออกไป หรือตัวอักษรที่รับกลับเข้ามาไม่เหมือนที่ส่งออกไป เป็นต้น

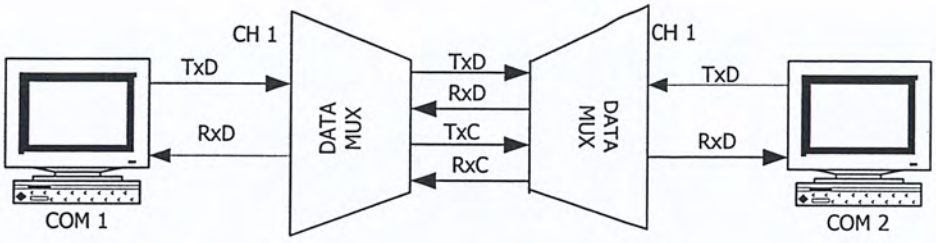
จากการทดลองส่งข้อมูลโดยส่งข้อความต่างๆตามรูปที่ 4.2 จะเห็นว่าสามรถรับข้อความเหล่านั้นกลับมาได้อย่างถูกต้อง ไม่ว่าจะทำการเลือกอัตราบิตเรทที่ใช้อัตราใดก็ตาม ซึ่งแสดงว่าเครื่องมัลติเพล็กซ์ข้อมูลสามารถทำงานได้ตามที่ออกแบบไว้ในระดับหนึ่งแล้ว



รูปที่ 4.2 แสดงผลการทดลองในลูปกลับ

4.2 การทดสอบการเชื่อมโยง (LINK TEST)

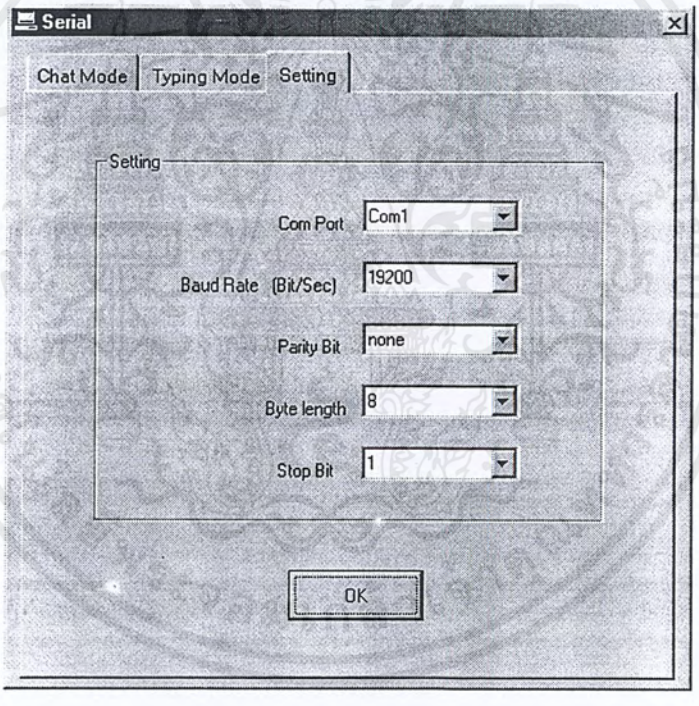
การทดสอบการเชื่อมโยง จะเป็นการทดลองเพื่อต้องการทราบว่า คู่สถานีของเครื่องมัลติเพล็กซ์ข้อมูลสามารถที่จะผ่านข้อมูลระหว่างกันได้หรือไม่ และมีข้อผิดพลาดประการใดบ้าง โดยในการทดลองส่วนนี้ เราจะทำการส่งผ่านข้อมูลจากคอมพิวเตอร์หลายๆเครื่อง เพื่อให้เครื่องมัลติเพล็กซ์ข้อมูลทำการมัลติเพล็กซ์และดีมัลติเพล็กซ์ข้อมูล โดยมีตัวกลางในการส่งผ่านข้อมูลเชื่อมต่ออยู่ระหว่างคู่สถานี ดังแสดงในรูปที่ 4.3



รูปที่ 4.3 การทดสอบการเชื่อมโยงระหว่างคู่สถานีของเครื่องมัลติเพล็กซ์ข้อมูล

ขั้นตอนการทดลอง

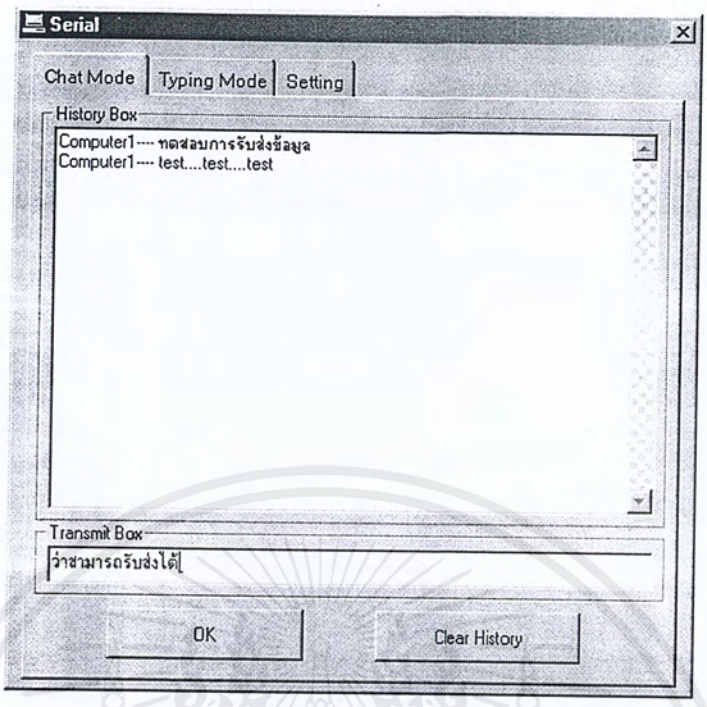
1. เลือกอัตราบิตเรทของการรับส่งข้อมูลให้ตรงกับความเร็วที่ได้เลือกไว้ในแต่ละช่องการรับส่ง



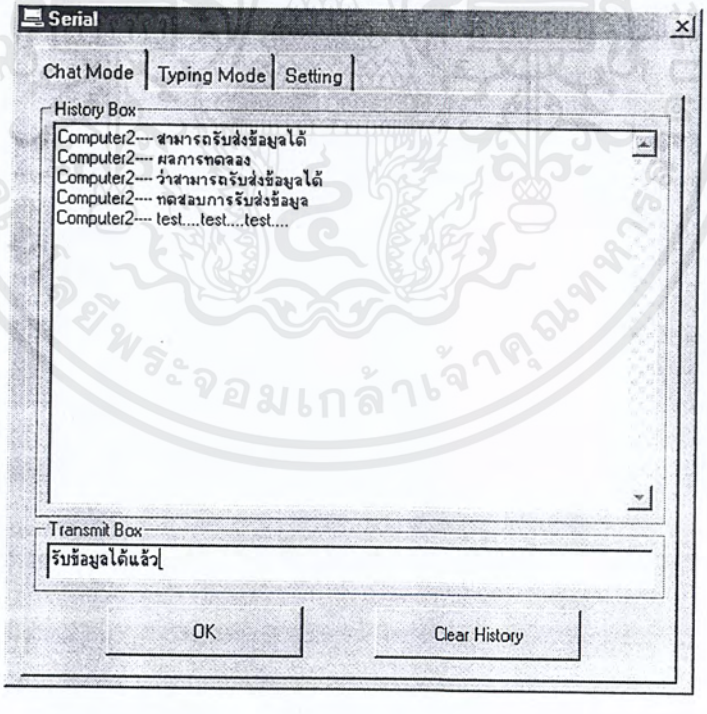
รูปที่4.4 แสดงการเลือกบิตเรทในการรับส่งข้อมูล

2. ปลายทางทำการส่งลูปกลับทางด้านสถานีต้นทางเพื่อแสดงให้สถานีต้นทางทราบว่าได้ส่งข้อมูลใดไปที่สถานีโดยการทดสอบจะมี 2ขั้นตอน คือ

2.1 ทดสอบโดยให้สถานีทางด้านส่ง ทำการส่งข้อมูลเพียงอย่างเดียว และให้สถานีทางด้านรับรับข้อมูลเพียงอย่างเดียว (ซิมเพล็กซ์)

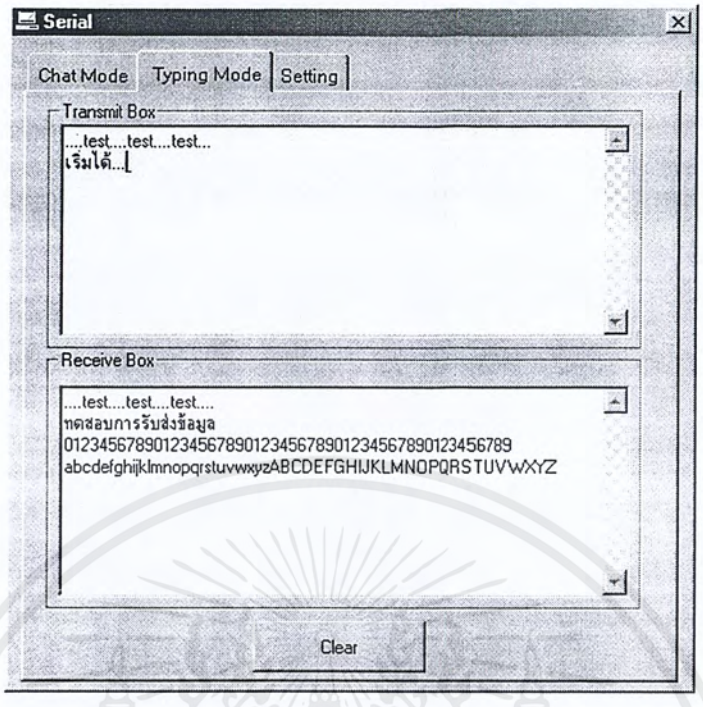


รูปที่ 4.5 แสดงการเริ่มต้นส่งข้อมูลจากสถานีต้นทาง

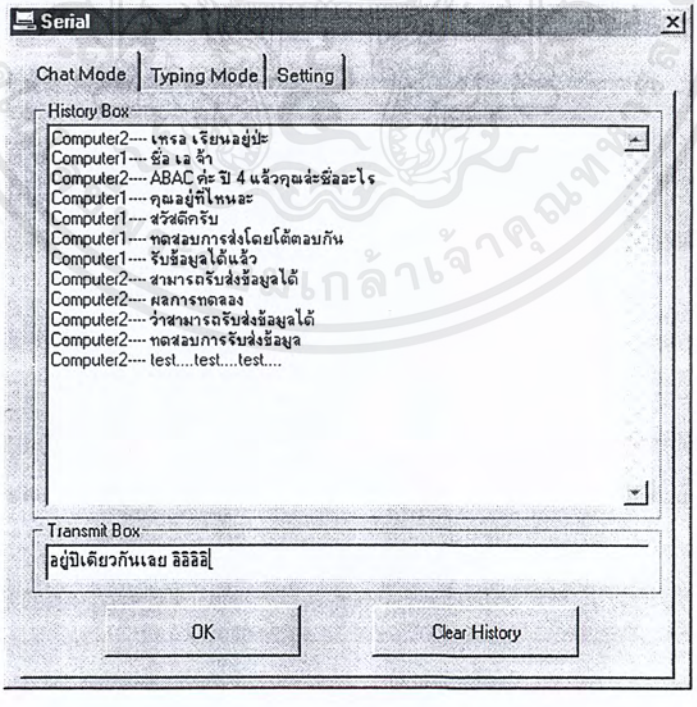


รูปที่ 4.6 แสดงการรับข้อมูลทางสถานีทางด้านรับปลายทาง

2.2 ทดสอบโดยให้ทั้ง 2 สถานีทำการโต้ตอบกัน(ฟูล ดูเพล็กซ์)ทำการกดคีย์ตัวอักษรใดๆ เพื่อส่งข้อมูลตัวอักษรออกมา และสังเกตว่าตัวอักษรทางด้านสถานีปลายทางที่ปรากฏนั้นถูกต้องหรือไม่



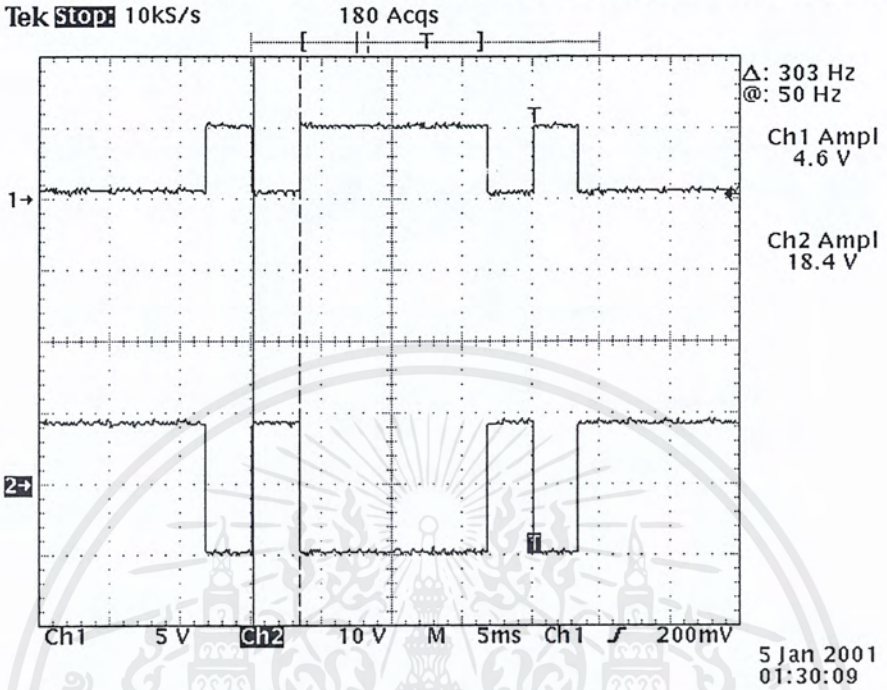
รูปที่ 4.7(ก) แสดงการรับส่งได้ระหว่างสถานีส่งและสถานีรับแบบเป็นตัวอักษร



รูปที่ 4.7(ข) แสดงการรับส่งได้ระหว่างสถานีส่งและสถานีรับแบบเป็นข้อความ

4.3 การทดสอบวงจรรับส่งข้อมูลผ่านพอร์ตอนุกรม

วัดระดับสัญญาณข้อมูลแบบทีทีแอลเปรียบเทียบกับระดับสัญญาณแรงดันมาตรฐาน RS-232

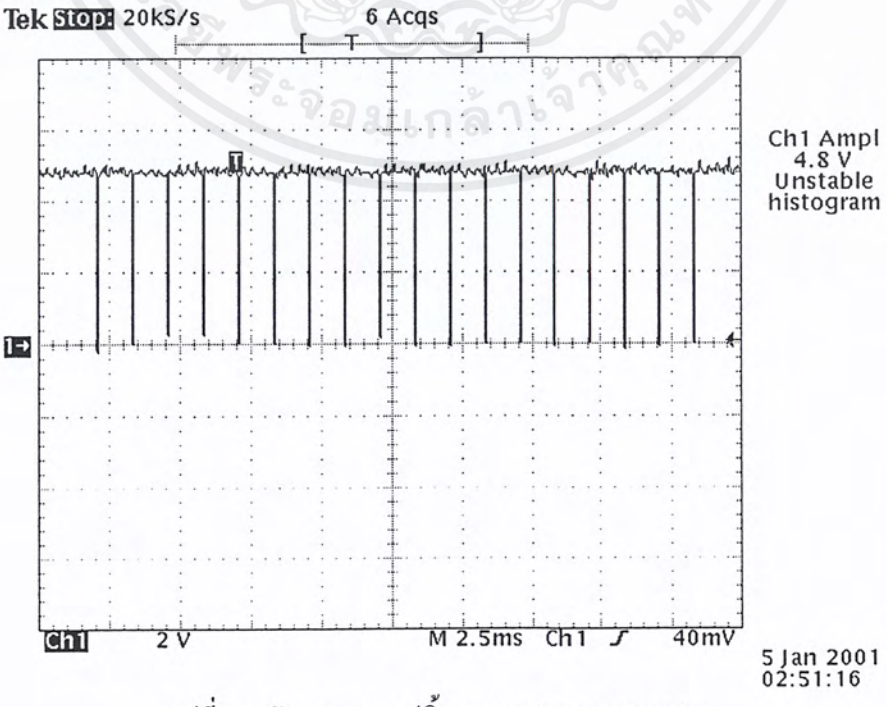


รูปที่ 4.8 ระดับสัญญาณทีทีแอลเปรียบเทียบกับระดับสัญญาณแรงดันมาตรฐาน RS-232

จากรูปที่ 4.8 เปรียบเทียบ ด้านบนคือสัญญาณทีทีแอล และด้านล่างคือ สัญญาณแรงดัน RS-232

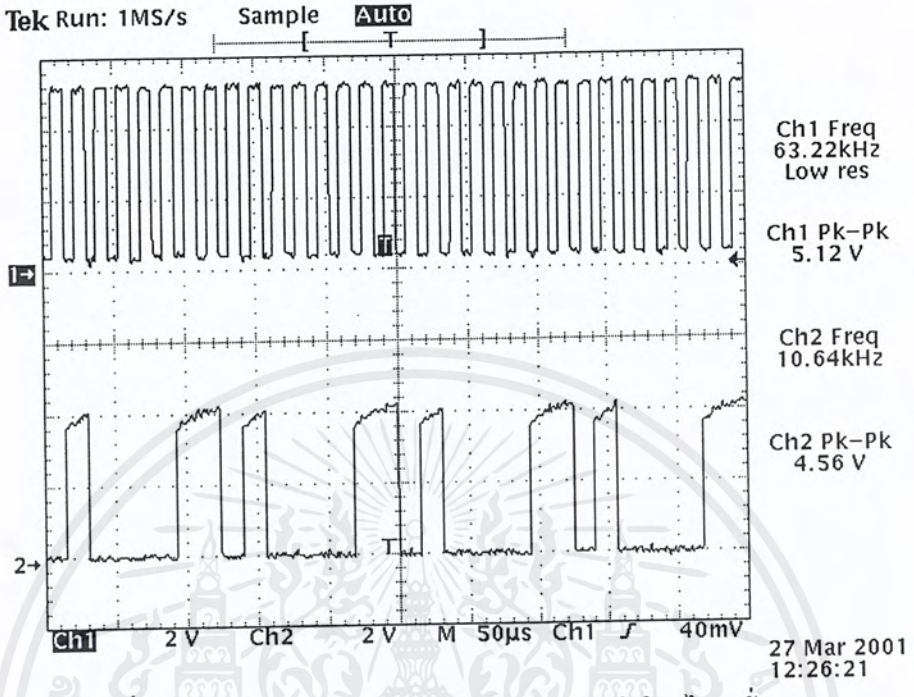
4.4 การทดสอบวงจรรับส่งข้อมูลแบบซิงโครไนเซชั่น

4.4.1 ทำการวัดสัญญาณซิงโครไนเซชั่น เพื่อนำมาทำการเปรียบเทียบกับสัญญาณนาฬิกา 64 กิโลเฮิรตซ์



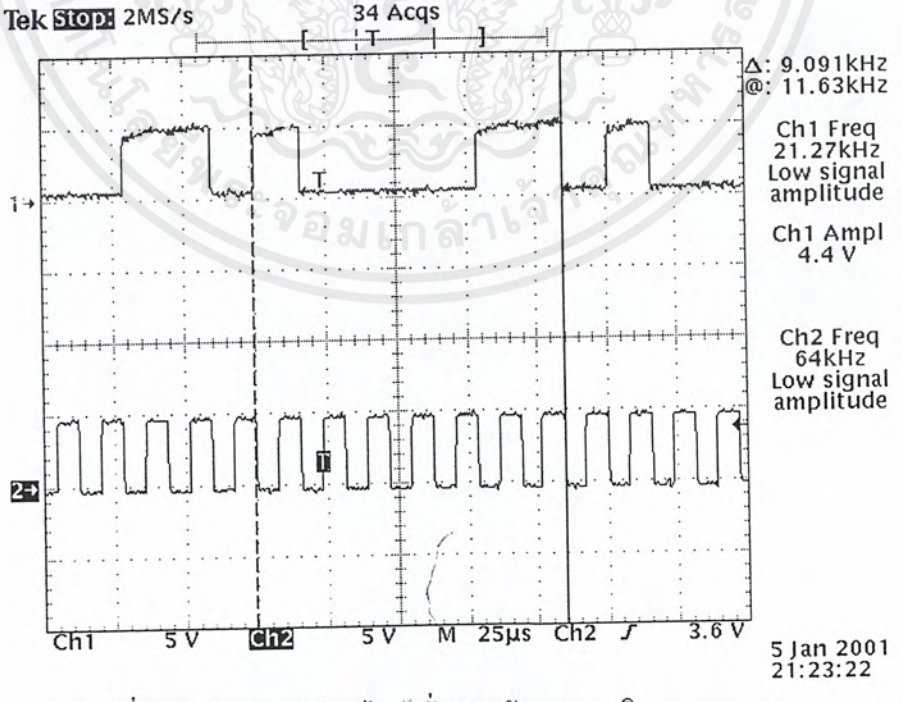
รูปที่ 4.9 สัญญาณแซมปลิ้งของ UART GM16C550

4.4.2 เปรียบเทียบสัญญาณชิงโครไนเซชั่นกับสัญญาณนาฬิกา



รูปที่ 4.10 สัญญาณนาฬิกาเปรียบเทียบกับสัญญาณชิงโครไนเซชั่น

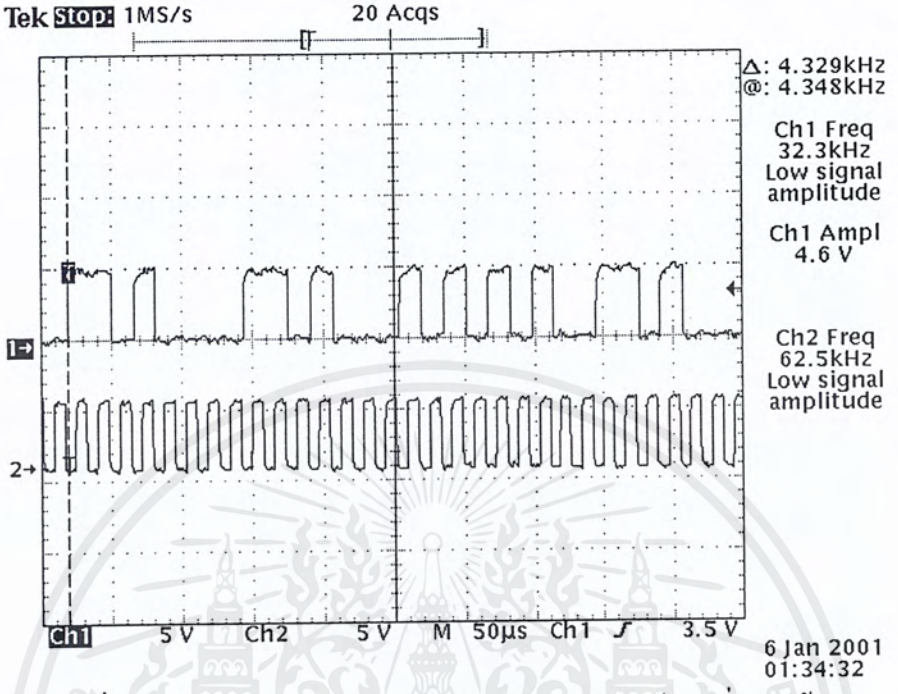
จากรูปที่ 4.10 จะเป็นการเปรียบเทียบ โดยด้านบน (Channel 1) คือสัญญาณนาฬิกาที่มาจากขา TxC และด้านล่าง (Channel 2) คือ สัญญาณชิงโครไนเซชั่น



รูปที่ 4.11 สัญญาณชิงโครไนเซชั่นและสัญญาณนาฬิกา

รูปที่ 4.11 เป็นการเปรียบเทียบ ด้านบนคือสัญญาณชิงโครไนเซชั่นและด้านล่างคือสัญญาณนาฬิกา

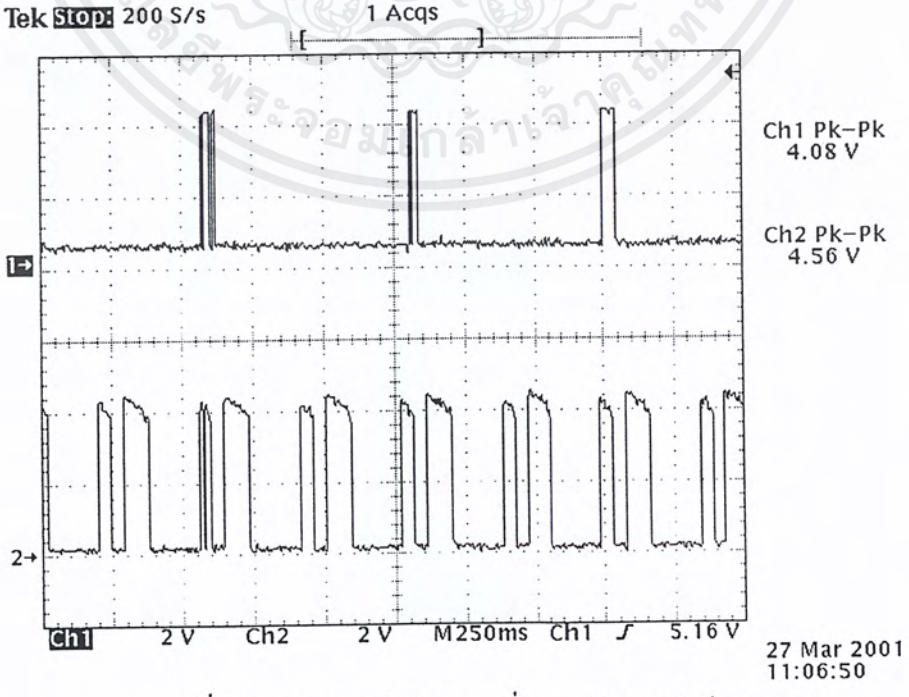
4.4.3 เปรียบเทียบสัญญาณชิงโครไนเซชั่นและข้อมูลกับสัญญาณนาฬิกา



รูปที่ 4.12 สัญญาณนาฬิกาเปรียบเทียบกับสัญญาณชิงโครไนเซชั่นและข้อมูล

จากรูปที่ 4.12 จะเป็นการเปรียบเทียบ โดยด้านบน (Channel 1) คือสัญญาณ ชิงโครไนเซชั่นและข้อมูล และด้านล่าง (Channel 2) คือ สัญญาณนาฬิกา

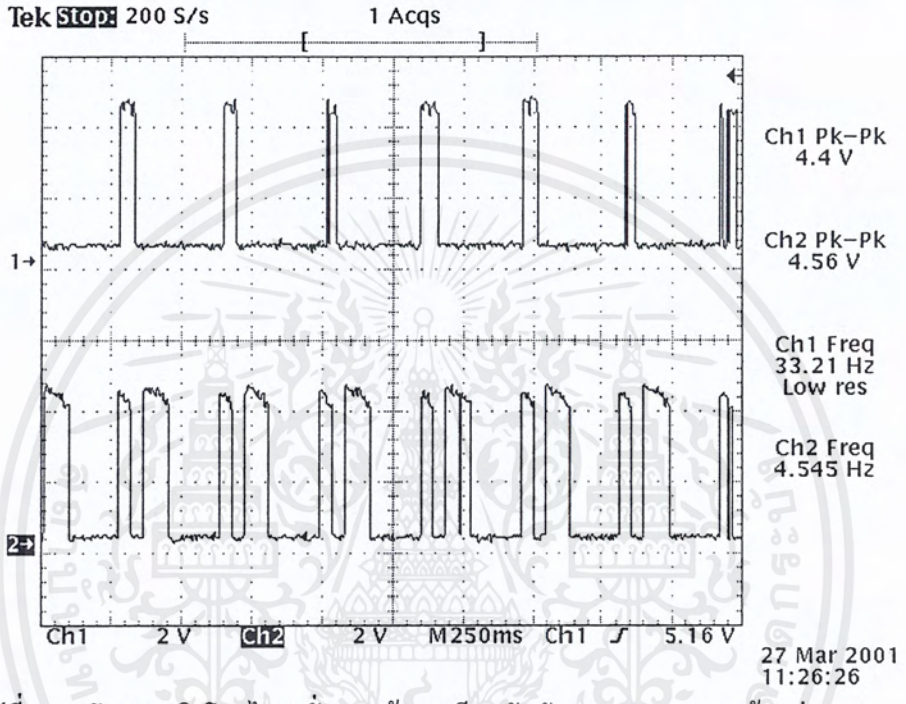
4.4.4 เปรียบเทียบสัญญาณชิงโครไนเซชั่นและสัญญาณชิงดีเทค



รูปที่ 4.13 สัญญาณชิงโครไนเซชั่นและสัญญาณชิงดีเทค

จากรูปที่ 4.13 คือ สัญญาณซิงโครไนเซชั่น และข้อมูลที่ส่งออกไปเมื่อเปรียบเทียบกับสัญญาณซิงดีเทค โดยด้านบน (Channel 1) คือสัญญาณซิงดีเทค และด้านล่าง (Channel 2) คือ สัญญาณซิงโครไนเซชั่นและข้อมูล

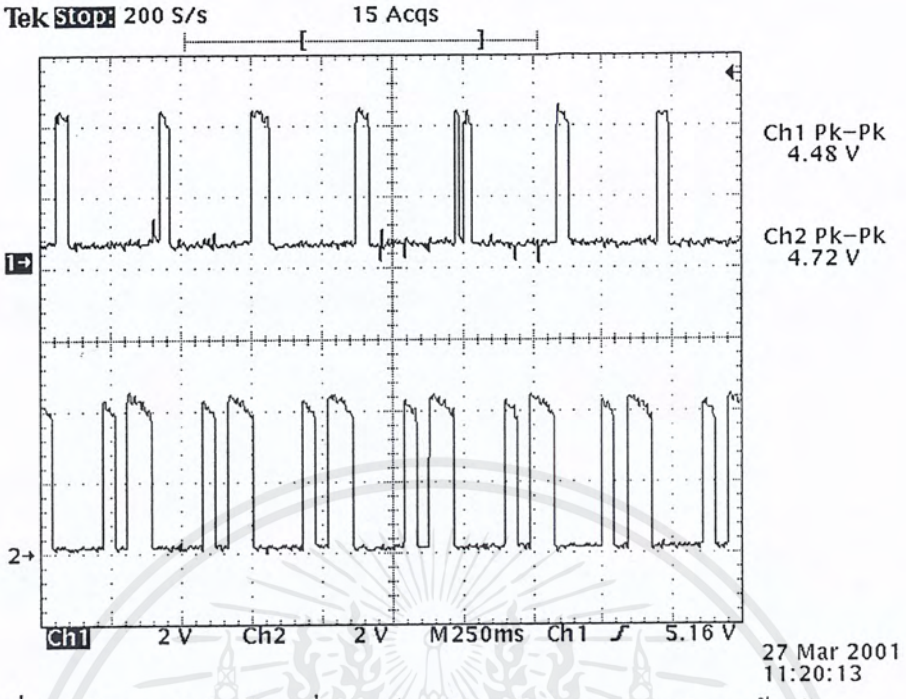
4.4.5 เปรียบเทียบสัญญาณซิงโครไนเซชั่นและข้อมูลเทียบกับสัญญาณ TxRdy ทางด้านส่ง



รูปที่ 4.14 สัญญาณซิงโครไนเซชั่นและข้อมูลเทียบกับสัญญาณ TxRdyทางด้านส่ง

จากรูปที่ 4.14 คือ สัญญาณซิงโครไนเซชั่น และข้อมูลที่ส่งออกไปเมื่อเปรียบเทียบกับสัญญาณ TxRdy โดยด้านบน (Channel 1) คือสัญญาณสัญญาณซิงโครไนเซชั่นและข้อมูลและด้านล่าง (Channel 2) คือ สัญญาณพร้อมส่งข้อมูลจากขา TxRdy ของทางด้านส่ง

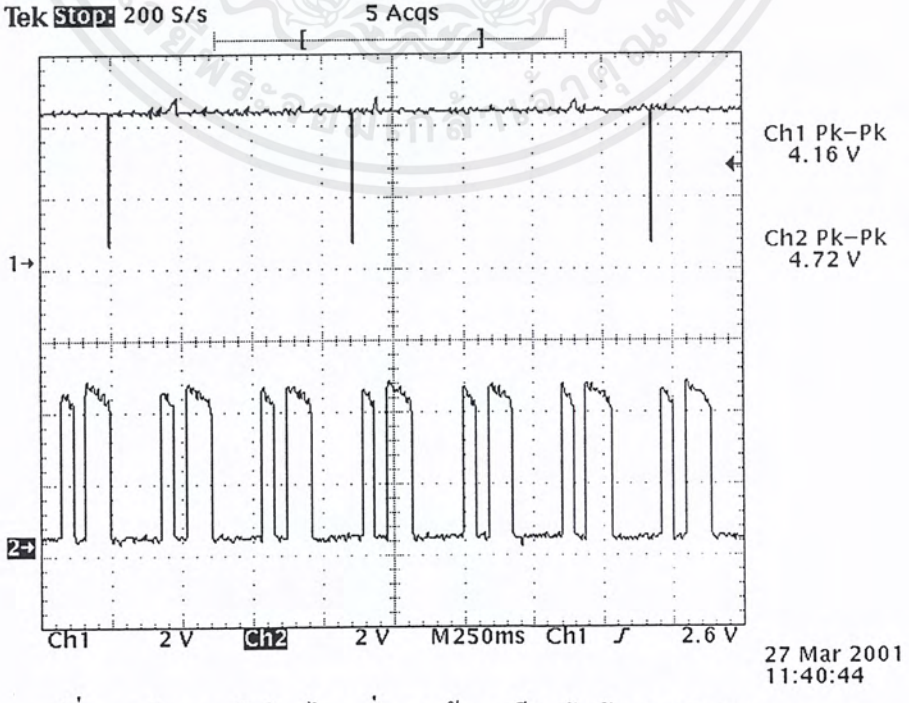
4.4.6 เปรียบเทียบสัญญาณซิงโครไนเซชั่นและข้อมูลเทียบกับสัญญาณ TxRdy ทางด้านรับ



รูปที่ 4.15 สัญญาณซิงโครไนเซชั่นและข้อมูลเทียบกับสัญญาณ TxRdy ทางด้านรับ

จากรูปที่ 4.15 คือ สัญญาณซิงโครไนเซชั่น และข้อมูลที่ส่งออกไปเมื่อเปรียบเทียบกับสัญญาณ TxRdy โดยด้านบน (Channel 1) คือสัญญาณซิงโครไนเซชั่นและข้อมูล และด้านล่าง (Channel 2) คือสัญญาณพร้อมส่งข้อมูลจากขา TxRdy ของทางด้านรับ

4.4.7 เปรียบเทียบสัญญาณซิงโครไนเซชั่นและข้อมูลเทียบกับสัญญาณ RxRdy



รูปที่ 4.16 สัญญาณซิงโครไนเซชั่นและข้อมูลเทียบกับสัญญาณ RxRdy

จากรูปที่ 4.16 คือ สัญญาณซิงโครไนเซชั่น และข้อมูลที่ส่งออกไปเมื่อเปรียบเทียบกับสัญญาณ RxRdy โดยด้านบน (Channel 1) คือสัญญาณพร้อมรับข้อมูลจากขา RxRdy และด้านล่าง (Channel 2) คือสัญญาณซิงโครไนเซชั่นและข้อมูล

บทที่ 5
บทวิจารณ์และสรุปผลการทดลอง

ปริญญานิพนธ์นี้ได้จัดทำขึ้นครอบคลุมเนื้อหาการมัลติเพล็กซ์ และดีมัลติเพล็กซ์สัญญาณข้อมูลแบบฟูลดูเพล็กซ์ รวมทั้งมีการเขียนโปรแกรมสั่งให้รับส่งข้อมูลแบบอะซิงโครนัสและแบบซิงโครนัสเพื่อทำการมัลติเพล็กซ์และดีมัลติเพล็กซ์สัญญาณให้ไปออกยังช่องสัญญาณเอาท์พุทที่ต้องการ โดยจะใช้ไบท์แรกที่ต่อจากสัญญาณซิงโครนัสเป็นตัวบอกว่ามีข้อมูลของช่องสัญญาณข้อมูลไหนส่งมาบ้าง เพื่อที่ทางด้านรับจะสามารถแยกสัญญาณข้อมูลได้อย่างถูกต้อง

ปัญหาที่พบ คือ ไม่สามารถที่จะจัดหาอุปกรณ์จำนวนมากเช่นคอมพิวเตอร์ได้และสัญญาณนาฬิกาที่ต้องการความละเอียดสูงมาก จึงแก้ปัญหาด้วยการส่งสัญญาณนาฬิกาไปพร้อมกับสัญญาณข้อมูลเพื่อนำไปใช้ดีมัลติเพล็กซ์สัญญาณที่ด้านรับ และการรับ-ส่งยังต้องใช้การเซตทางด้านฮาร์ดแวร์ที่มีค่าตรงกันของทางด้านส่งและทางด้านรับ รวมทั้งช่องสัญญาณที่รับส่งต้องตรงกันด้วยจึงจะสามารถรับส่งกันได้ และการส่งข้อมูลที่มีลักษณะต่อเนื่องกันเป็นไฟล์จะมีความผิดพลาดเกิดขึ้นเมื่อส่งด้วยความเร็วที่สูงขึ้น

แนวทางการพัฒนา ในการออกแบบอุปกรณ์ในอนาคต สามารถนำไปเชื่อมต่อกับอุปกรณ์อื่นเช่น โมเด็ม เพื่อให้สามารถใช้งานได้อย่างมีประสิทธิภาพมากยิ่งขึ้น เนื่องจากทางฮาร์ดแวร์ยังสามารถรองรับได้ อีกทั้งเมื่อมีการพัฒนาทางด้านเทคโนโลยีในการผลิตชิป จะทำให้สามารถรองรับความเร็วที่สูงขึ้นเมื่อต้องการส่งข้อมูลที่มีลักษณะต่อเนื่องกันเป็นไฟล์ได้ถูกต้องทั้งหมด และไม่มีความผิดพลาด

ภาคผนวก ก.
การใช้งานเครื่องมัลติเพล็กซ์ข้อมูล

SW7	SW6	SW5	อัตราเร็ว (bps)
0	0	0	19200
0	0	1	9600
0	1	0	7200
0	1	1	4800
1	0	0	2400
1	0	1	1200
1	1	0	600
1	1	1	300

SW3	SW4	พาริตี้
0	0	None
0	1	None
1	0	Odd
1	1	Even

SW2	จำนวน Stop bit
0	1
1	2

SW1	SW0	จำนวนบิต
0	0	5
0	1	6
1	0	7
1	1	8

ตารางที่ ก-1 การตั้งค่าคุณสมบัติของช่องสัญญาณ



ภาคผนวก ข.

โปรแกรมซอร์สโค้ดสำหรับเครื่องมัลติเพล็กซ์ข้อมูล

```
;=============================================================

;=============================================================

        Counter         equ     30h

        FlagSend        equ     31h

        FlagRev         equ     32h

        Temp            equ     33h

        DLL             equ     34h

        DLM             equ     35h

        ContW           equ     36h

        DplUart         equ     37h

        DphUart         equ     38h

        DataUart1       equ     39h

        DataUart2       equ     3ah

        DataUart3       equ     3bh

        DataUart4       equ     3ch

        DataUsart1      equ     3dh

        DataUsart2      equ     3eh

        DataUsart3      equ     3fh

        DataUsart4      equ     40h

        ChSendFlag      equ     08h

        port1           equ     8000h

        port2           equ     8008h

        port3           equ     8010h

        port4           equ     8018h

;=============================================================

;=============================================================
```

```
                org     0000h
                call    delay
                call    set8255

                call    Set8251

SynDet:         mov     dptr,#4001h
                movx    a,@dptr
                jnb     acc.6,SynDet
                mov     p1,#0aah
                call    ReadDipSw
;*************************************************************
                ;Start Program
;*************************************************************

                mov     counter,#00h
                mov     FlagSend,#00h
                mov     FlagRev,#00h
                setb    ChSendFlag

;*************************************************************
                ;Main  Program
;*************************************************************
;*************************************************************

LoopMain:       call    ReadUart
                call    Read8251
                call    WriteUart
                call    Write8251
```

```
                jmp     LoopMain

;*************************************************************
;*************************************************************
                ;Read Uart
;*************************************************************

ReadUart:
ReadUart1:      mov     dptr,#8005h
                movx    a,@dptr
                jnb     acc.0,ReadUart2
                ;---------------------
                mov     a,FlagSend              ;Keep Flag1
                orl     a,#00000001b
                mov     FlagSend,a
                ;---------------------
                mov     dptr,#8000h             ;Keep Data1
                movx    a,@dptr
                mov     DataUart1,a
ReadUart2:      mov     dptr,#800dh
                movx    a,@dptr
                jnb     acc.0,ReadUart3
                ;---------------------
                mov     a,FlagSend              ;Keep Flag2
                orl     a,#00000010b
                mov     FlagSend,a
                ;---------------------
                mov     dptr,#8008h             ;Keep Data2
                movx    a,@dptr
                mov     DataUart2,a
```

```
ReadUart3:      mov     dptr,#8015h
                movx    a,@dptr
                jnb     acc.0,ReadUart4
                ;---------------------
                mov     a,FlagSend              ;Keep Flag3
                orl     a,#00000100b
                mov     FlagSend,a
                ;---------------------
                mov     dptr,#8010h             ;Keep Data3
                movx    a,@dptr
                mov     DataUart3,a
ReadUart4:      mov     dptr,#801dh
                movx    a,@dptr
                jnb     acc.0,RetReadUart
                ;---------------------
                mov     a,FlagSend              ;Keep Flag4
                orl     a,#00001000b
                mov     FlagSend,a
                ;---------------------
                mov     dptr,#8018h             ;Keep Data4
                movx    a,@dptr
                mov     DataUart4,a
RetReadUart:    ret

;************************************************************
                ;Read 8251
;************************************************************

Read8251:
ChStatus8251:   mov     dptr,#4001h
```

```
                movx    a,@dptr
                jnb     acc.1,RetRead8251
                mov     dptr,#4000h             ;Read Data
                movx    a,@dptr
                mov     temp,a
check16:        cjne    a,#16H,CheckCounter     ;Check Data16
                jmp     ClrCounter
CheckCounter:   mov     a,counter
chCount0:       cjne    a,#00h,chCount1         ;Check Count0
                mov     FlagRev,temp
                jmp     IncCount
chCount1:       cjne    a,#01h,chCount2         ;Check Count1
                mov     DataUsart1,temp
                jmp     IncCount
chCount2:       cjne    a,#02h,chCount3         ;Check Count2
                mov     DataUsart2,temp
                jmp     IncCount
chCount3:       cjne    a,#03h,chCount4         ;Check Count3
                mov     DataUsart3,temp
                jmp     IncCount
chCount4:       mov     DataUsart4,temp         ;Check Count4
IncCount:       inc     counter
                mov     a,counter
```

```
                cjne    a,#05h,RetRead8251
ClrCounter:     mov     counter,#00h
RetRead8251:    ret

;*************************************************************
                ;Write 8251
;*************************************************************

RETWRITE82511:  JMP     RETWRITE8251

Write8251:
ChWrite8251:    mov     dptr,#4001h
                movx    a,@dptr
                jnb     acc.0,RetWrite82511

CheckFlag:      mov     a,FlagSend
                cjne    a,#00h,SendFlag
                jmp     RetWrite8251

SendFlag:       jnb     ChSendFlag,chFlagBit1
                mov     dptr,#4000h
                mov     a,FlagSend
                movx    @dptr,a
                clr     ChSendFlag

                jmp     RetWrite8251
chFlagBit1:     mov     a,FlagSend
                anl     a,#00000001b
                jnb     acc.0,chFlagBit2
                mov     dptr,#4000h
                mov     a,DataUart1
```

```
                movx    @dptr,a
                mov     a,FlagSend
                clr     acc.0
                mov     FlagSend,a
                cjne    a,#00h,RetWrite8251
                setb    ChSendFlag
                jmp     RetWrite8251

chFlagBit2:     mov     a,FlagSend
                anl     a,#00000010b
                jnb     acc.1,chFlagBit3
                mov     dptr,#4000h
                mov     a,DataUart2
                movx    @dptr,a
                mov     a,FlagSend
                clr     acc.1
                mov     FlagSend,a
                cjne    a,#00h,RetWrite8251
                setb    ChSendFlag
                jmp     RetWrite8251

chFlagBit3:     mov     a,FlagSend
                anl     a,#00000100b
                jnb     acc.2,chFlagBit4
                mov     dptr,#4000h
                mov     a,DataUart3
                movx    @dptr,a
                mov     a,FlagSend
                clr     acc.2
                mov     FlagSend,a
                cjne    a,#00h,RetWrite8251
```

```
                setb    ChSendFlag
                jmp     RetWrite8251

chFlagBit4:     mov     a,FlagSend
                anl     a,#00001000b
                jnb     acc.3,RetWrite8251
                mov     dptr,#4000h
                mov     a,DataUart4
                movx    @dptr,a
                mov     a,FlagSend
                clr     acc.3
                mov     FlagSend,a
                cjne    a,#00h,RetWrite8251
                setb    ChSendFlag

RetWrite8251:   ret

;*************************************************************
                ;Write Uart
;*************************************************************

WriteUart:
chFlagRav1:     mov     a,FlagRev
                jnb     acc.0,chFlagRav2
                clr     acc.0
                mov     FlagRev,a
                mov     dptr,#8000h
                mov     a,DataUsart1
                movx    @dptr,a
                jmp     ShiftData

chFlagRav2:     mov     a,FlagRev
```

```
                jnb     acc.1,chFlagRav3
                clr     acc.1
                mov     FlagRev,a
                mov     dptr,#8008h
                mov     a,DataUsart1
                movx    @dptr,a
                jmp     ShiftData

chFlagRav3:     mov     a,FlagRev
                jnb     acc.2,chFlagRav4
                clr     acc.2
                mov     FlagRev,a
                mov     dptr,#8010h
                mov     a,DataUsart1
                movx    @dptr,a
                jmp     ShiftData

chFlagRav4:     mov     a,FlagRev
                jnb     acc.3,RetWriteUart
                clr     acc.3
                mov     FlagRev,a
                mov     dptr,#8018h
                mov     a,DataUsart1
                movx    @dptr,a

ShiftData:      mov     DataUsart1,DataUsart2
                mov     DataUsart2,DataUsart3
                mov     DataUsart3,DataUsart4

RetWriteUart:   ret

;*************************************************************
```

```
                    ;Set Uart
;*************************************************************

SetUart:
 setu:
                    MOV         A,DPLUART       ;SET UART
                    ADD         A,#03H
                    MOV         DPL,A
                    MOV         DPH,DPHUART
                    mov         a,#80h
                    MOVX        @DPTR,A         ;SET DLAB

                    PUSH        DPL
                    MOV         DPL,DPLUART
                    MOV         DPH,DPHUART
                    MOV         A,DLL           ;DLL 300
                    MOVX        @DPTR,A

                    INC         DPTR
                    MOV         A,DLM           ;SET BAUD RATE
                    MOVX        @DPTR,A          ;DLM 300
                    POP         DPL

                    MOV         A,ContW         ;00011010B
                    MOVX        @DPTR,A          ;SET
300,E,7,1
                    CALL        DELAY1
                    RET
```

```
;*************************************************************
                        ;Set Usart
;*************************************************************


Set8251:                MOV DPTR,#4001H
                        MOV A,#0H
                        MOVX @DPTR,A
                        MOVX @DPTR,A
                        MOVX @DPTR,A

                        MOV A,#40H          ;RESET
                        MOVX @DPTR,A

                        MOV A,#0CH          ;MODE WORD
0000 1100 8 BIT
                        MOVX @DPTR,A

                        MOV A,#16H          ;SYNC WORD
                        MOVX @DPTR,A        ;0001 0110
                        MOVX @DPTR,A

                        MOV A,#85H
                        MOVX @DPTR,A          ;1000 0101

    START_BYTE:         MOV A,#0CCH         ;1010 1010
                        MOV DPTR,#4000H
                        MOVX @DPTR,A
                        LCALL DELAY
```

```
                    ret

;*************************************************************
                ;Read Dip
;*************************************************************


ReadDipSW:      mov     r4,#04h
                mov     r5,#00001000b

LoopRd:         mov     dptr,#6000h
                mov     a,r5
                movx    @dptr,a
                call    delay1
                mov     dptr,#6001h
                movx    a,@dptr

;                cjne    a,#0fah,errr
;                mov     p1,#55h
;                jmp     $

;       errr:    mov     p1,#00h
;                jmp     $

                call    FindUart
                mov     a,r4
                dec     a
                mov     b,#08h
                mul     ab
                mov     DplUart,a
                mov     DphUart,#80h
```

```
                call    SetUart
                mov     a,r5
                rr      a
                mov     r5,a
                djnz    r4,LoopRd
                ret

;*************************************************************
                ;Find ControlWord Uart
;*************************************************************

FindUart:
FindDLL:        mov     temp,a
                anl     a,#11100000b
                rl      a
                rl      a
                rl      a
                mov     dptr,#tableDLL
                movc    a,@a+dptr
                mov     DLL,a

FindDLM:        mov     a,temp
                anl     a,#11100000b
                rl      a
                rl      a
                rl      a
                mov     dptr,#tableDLM
                movc    a,@a+dptr
                mov     DLM,a
```

```
FinContW:       mov     a,temp
                anl     a,#00011111b
                mov     ContW,a
                ret
;*************************************************************
                ;set8255
;*************************************************************

Set8255:        mov     dptr,#6003h
                mov     a,#82h
                movx    @dptr,a
                ret

;*************************************************************


        DELAY1:     MOV R0,#10
                    DJNZ R0,$
                    RET
        DELAY:      MOV R2,#10
        DEL:        MOV R3,#179
        DE:         MOV R4,#0
                    DJNZ R4,$
                    DJNZ R3,DE
                    DJNZ R2,DEL
                    RET

tableDLL:       db      06h,0ch,10h,18h,30h,60h,0c0h,80h
tableDLM:       db      00h,00h,00h,00h,00h,00h,00h,01h
                END
```



ภาคผนวก ค.

โปรแกรมซอร์สโค้ดสำหรับการรับส่งข้อมูล

```
Option Explicit

Public Baud, Pari, Length, Port, Sto, Sett, AllInp, Inp, outp, outp2

As String

Public Comm, x, i As Integer

Private Sub Command1_Click()
             Call chat
End Sub

Private Sub Command2_Click()

            Text2.Text = ""
            Text1.Text = ""

End Sub

Private Sub Command3_Click()

            Call setcomm

End Sub

Private Sub setcomm()

            Baud = Combo1.Text

            Pari = Combo2.Text

            Sto = Combo3.Text

            Length = Combo4.Text

            Port = Combo5.Text

            Select Case Pari

            Case "none"
                        Pari = "n"
            Case "Even"
                        Pari = "e"
            Case "Odd"
                        Pari = "o"
            End Select

            If Port = "Com1" Then

                    If MSComm1.CommPort = 2 Then
                            MSComm1.CommPort = 1

                    End If
             Else
                    If MSComm1.CommPort = 1 Then
                            MSComm1.CommPort = 2

                     End If

             End If
```

```
            Sett = Baud + "," + Pari + "," + Length + "," + Sto
            MSComm1.Settings = Sett

End Sub


Private Sub Command4_Click()
            Text4.Text = ""
            Text3.Text = ""
End Sub

Private Sub Form_Load()

            MSComm1.Settings = "9600,n,8,1"
            MSComm1.CommPort = 1
            MSComm1.PortOpen = True
            MSComm1.RThreshold = 60
            MSComm1.InputLen = 60

            Combo5.Text = "Com1"
            Combo1.Text = "9600"
            Combo2.Text = "none"
            Combo3.Text = "1"
            Combo4.Text = "8"

End Sub

Private Sub MSComm1_OnComm()

    If SSTab1.Tab = 0 Then

                AllInp = MSComm1.Input
                Text1.Text = "Computer2---- " & AllInp & Chr(13) &

Chr(10) & Text1.Text

                AllInp = ""
     Else: Text4.Text = Text4.Text & MSComm1.Input
     End If

End Sub


Private Sub chat()

            outp = Text2.Text
            If Len(outp) <= 60 Then

                    x = 60 - Len(outp)
                    If x > 0 Then
                            For i = 1 To x
                                outp = outp + " "
                            Next I

                    End If

             Else
                            outp2 = Mid(outp, 1, 60)
                            MSComm1.Output = outp2
                            outp = Mid(outp, 61, Len(outp))
```

```
                                        x = 60 - Len(outp)
                                        For i = 1 To x
                                         outp = outp + " "
                                        Next i

            End If

            Text1.Text = "Computer1---- " & Text2.Text & Chr(13) &
Chr(10) & Text1.Text

            MSComm1.Output = outp
            Text2.Text = ""
End Sub

Private Sub SSTab1_Click(PreviousTab As Integer)

            If SSTab1.Tab = 0 Then
                MSComm1.InputLen = 60
                MSComm1.RThreshold = 60

            Else

                MSComm1.InputLen = 1
                MSComm1.RThreshold = 1

            End If
End Sub

Private Sub Text2_KeyPress(KeyAscii As Integer)
            If KeyAscii = 13 Then

                    KeyAscii = 0
                    Call chat

            End If

End Sub

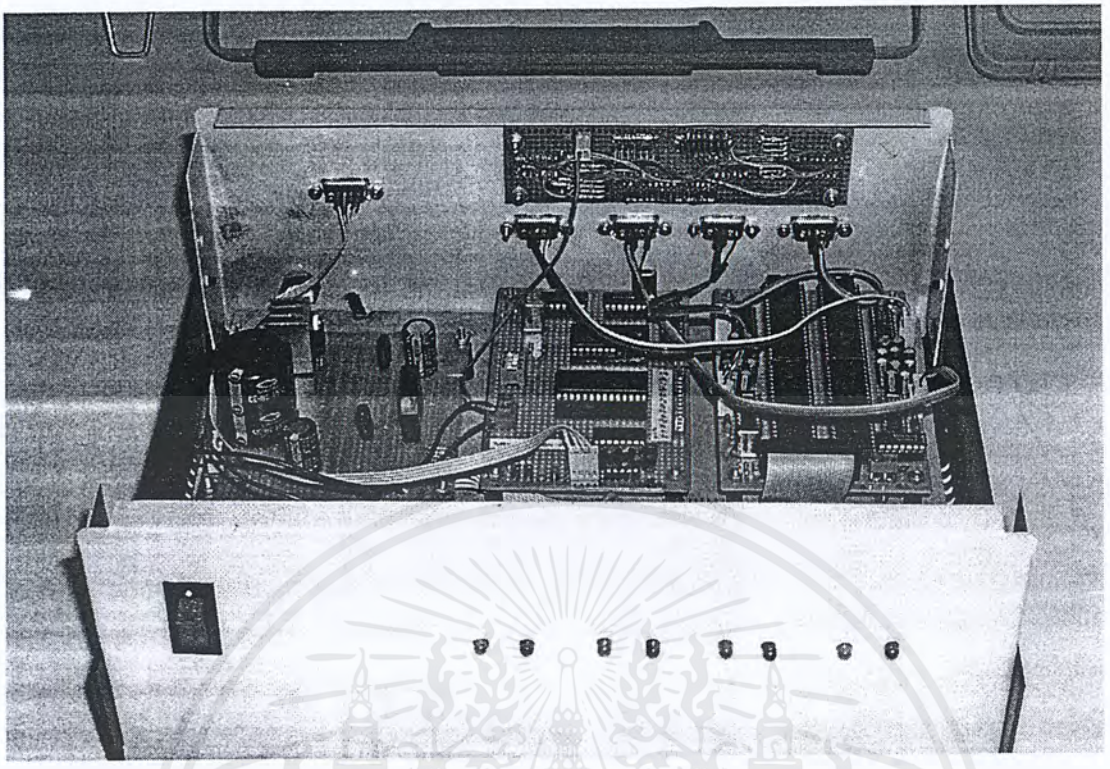
Private Sub Text3_KeyPress(KeyAscii As Integer)
        If KeyAscii = 13 Then

            MSComm1.Output = vbNewLine
        Else: MSComm1.Output = Chr(KeyAscii)

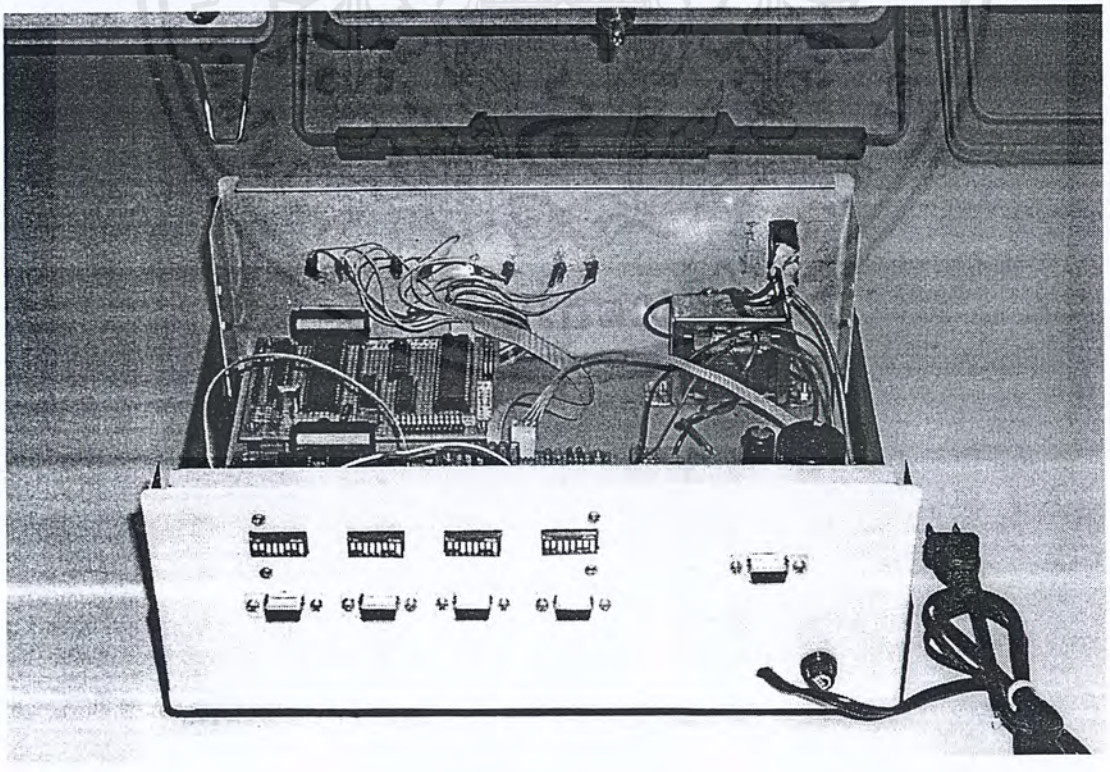
        End If

End Sub
```

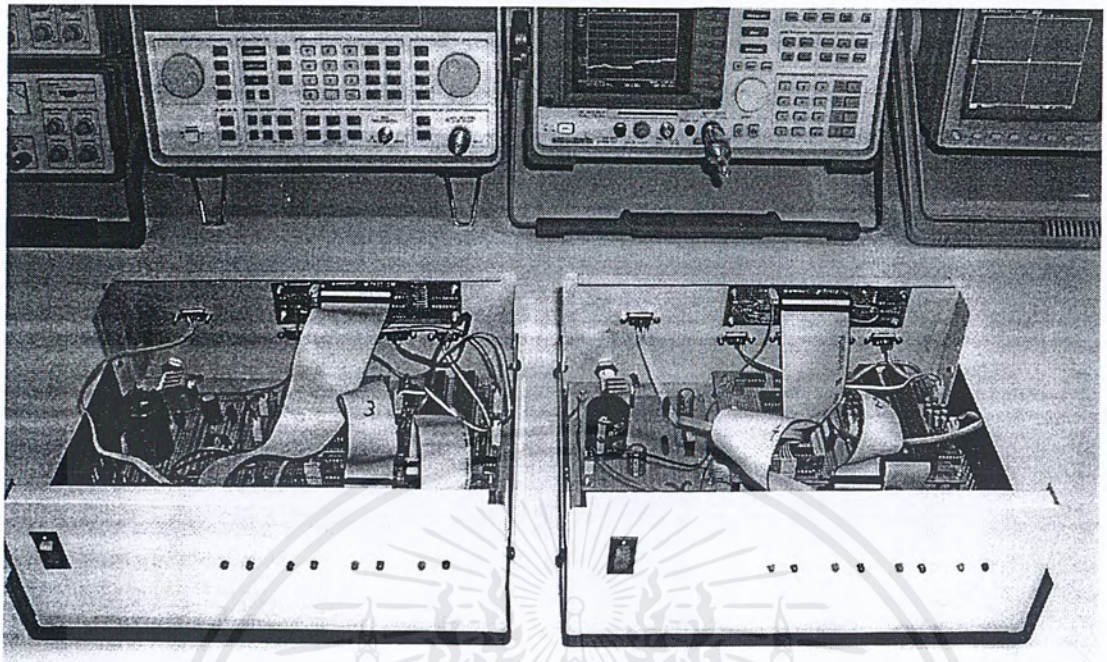
ภาคผนวก ง.
ภาพเครื่องมัลติเพล็กซ์ข้อมูล



ภาพแสดงเครื่องมัลติเพล็กซ์รับ-ส่งข้อมูล (ด้านหน้า)



ภาพแสดงเครื่องมัลติเพล็กซ์รับ-ส่งข้อมูล (ด้านหลัง)



ภาพชิ้นงานแสดงเครื่องมัลติเพล็กซ์ทั้งเครื่องรับและเครื่องส่ง



ภาคผนวก จ.
DATA SHEET

National Semiconductor

June 1995

PC16550D Universal Asynchronous Receiver/Transmitter with FIFOs†

General Description

The PC16550D is an improved version of the original 16450 Universal Asynchronous Receiver/Transmitter (UART). Functionally identical to the 16450 on powerup (CHARACTER mode)* the PC16550D can be put into an alternate mode (FIFO mode) to relieve the CPU of excessive software overhead.

In this mode internal FIFOs are activated allowing 16 bytes (plus 3 bits of error data per byte in the RCVR FIFO) to be stored in both receive and transmit modes. All the logic is on chip to minimize system overhead and maximize system efficiency. Two pin functions have been changed to allow signalling of DMA transfers.

The UART performs serial-to-parallel conversion on data characters received from a peripheral device or a MODEM, and parallel-to-serial conversion on data characters received from the CPU. The CPU can read the complete status of the UART at any time during the functional operation. Status information reported includes the type and condition of the transfer operations being performed by the UART, as well as any error conditions (parity, overrun, framing, or break interrupt).

The UART includes a programmable baud rate generator that is capable of dividing the timing reference clock input by divisors of 1 to ($2^{16}-1$), and producing a 16 × clock for driving the internal transmitter logic. Provisions are also included to use this 16 × clock to drive the receiver logic. The UART has complete MODEM-control capability, and a processor-interrupt system. Interrupts can be programmed to the user's requirements, minimizing the computing required to handle the communications link.

The UART is fabricated using National Semiconductor's advanced M²CMOS process.

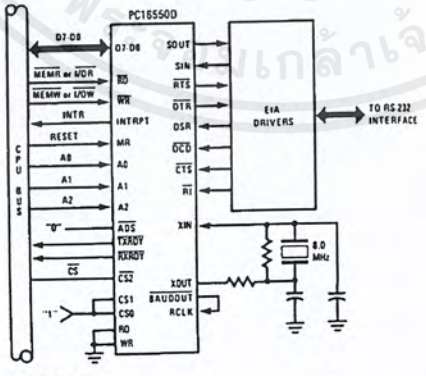
*Can also be reset to 16450 Mode under software control.

†Note: This part is patented.

Features

- Capable of running all existing 16450 software.
- Pin for pin compatible with the existing 16450 except for CSOUT (24) and NC (29). The former CSOUT and NC pins are $\overline{\text{TXRDY}}$ and $\overline{\text{RXRDY}}$, respectively.
- After reset, all registers are identical to the 16450 register set.
- In the FIFO mode transmitter and receiver are each buffered with 16 byte FIFO's to reduce the number of interrrupts presented to the CPU.
- Adds or deletes standard asynchronous communication bits (start, stop, and parity) to or from the serial data.
- Holding and shift registers in the 16450 Mode eliminate the need for precise synchronization between the CPU and serial data.
- Independently controlled transmit, receive, line status, and data set interrupts.
- Programmable baud generator divides any input clock by 1 to ($2^{16}-1$) and generates the 16 × clock.
- Independent receiver clock input.
- MODEM control functions (CTS, RTS, DSR, DTR, RI, and DCD).
- Fully programmable serial-interface characteristics:
  - 5-, 6-, 7-, or 8-bit characters
  - Even, odd, or no-parity bit generation and detection
  - 1-, 1½-, or 2-stop bit generation
  - Baud generation (DC to 1.5M baud).
- False start bit detection.
- Complete status reporting capabilities.
- TRI-STATE® TTL drive for the data and control buses.
- Line break generation and detection.
- Internal diagnostic capabilities:
  - Loopback controls for communications link fault isolation
  - Break, parity, overrun, framing error simulation.
- Full prioritized interrupt system controls.

Basic Configuration



TL/C/8652–1

TRI-STATE® is a registered trademark of National Semiconductor Corp.

©1995 National Semiconductor Corporation TL/C/8652

RRD-B30M75/Printed in U. S. A.

PC16550D Universal Asynchronous Receiver/Transmitter with FIFOs

Table of Contents

1.0 ABSOLUTE MAXIMUM RATINGS

2.0 DC ELECTRICAL CHARACTERISTICS

3.0 AC ELECTRICAL CHARACTERISTICS

4.0 TIMING WAVEFORMS

5.0 BLOCK DIAGRAM

6.0 PIN DESCRIPTIONS

7.0 CONNECTION DIAGRAMS

8.0 REGISTERS

8.1 Line Control Register
8.2 Typical Clock Circuits

8.0 REGISTERS (Continued)

8.3 Programmable Baud Generator
8.4 Line Status Register
8.5 FIFO Control Register
8.6 Interrupt Identification Register
8.7 Interrupt Enable Register
8.8 Modem Control Register
8.9 Modem Status Register
8.10 Scratchpad Register
8.11 FIFO Interrupt Mode Operation
8.12 FIFO Polled Mode Operation

9.0 TYPICAL APPLICATIONS

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

1.0 Absolute Maximum Ratings

Temperature Under Bias	0°C to +70°C
Storage Temperature	−65°C to +150°C
All Input or Output Voltages with Respect to V_{SS}	−0.5V to +7.0V
Power Dissipation	1W

Note: *Maximum ratings indicate limits beyond which permanent damage may occur. Continuous operation at these limits is not intended and should be limited to those conditions specified under DC electrical characteristics.*

2.0 DC Electrical Characteristics

T_A = 0°C to +70°C, V_{DD} = +5V ±10%, V_{SS} = 0V, unless otherwise specified.

Symbol	Parameter	Conditions	Min	Max	Units
V_{ILX}	Clock Input Low Voltage		−0.5	0.8	V
V_{IHX}	Clock Input High Voltage		2.0	V_{DD}	V
V_{IL}	Input Low Voltage		−0.5	0.8	V
V_{IH}	Input High Voltage		2.0	V_{DD}	V
V_{OL}	Output Low Voltage	I_{OL} = 1.6 mA on all (Note 1)		0.4	V
V_{OH}	Output High Voltage	I_{OH} = −1.0 mA (Note 1)	2.4		V
I_{CC}(AV)	Average Power Supply Current	V_{DD} = 5.5V, T_A = 25°C No Loads on output SIN, DSR, DCD, CTS, RI = 2.0V All other inputs = 0.8V		15	mA
I_{IL}	Input Leakage	V_{DD} = 5.5V, V_{SS} = 0V All other pins floating. V_{IN} = 0V, 5.5V		±10	μA
I_{CL}	Clock Leakage			±10	μA
I_{OZ}	TRI-STATE Leakage	V_{DD} = 5.5V, V_{SS} = 0V V_{OUT} = 0V, 5.25V 1) Chip deselected 2) WRITE mode, chip selected		±20	μA
V_{ILMR}	MR Schmitt V_{IL}			0.8	V
V_{IHMR}	MR Schmitt V_{IH}		2.0		V

Note 1: Does not apply to XOUT

Capacitance T_A = 25°C, V_{DD} = V_{SS} = 0V

Symbol	Parameter	Conditions	Min	Typ	Max	Units
C_{XIN}	Clock Input Capacitance	f_c = 1 MHz Unmeasured pins returned to V_{SS}		7	9	pF
C_{XOUT}	Clock Output Capacitance			7	9	pF
C_{IN}	Input Capacitance			5	7	pF
C_{OUT}	Output Capacitance			6	8	pF
$C_{I/O}$	Input/Output Capacitance			10	12	pF

3.0 AC Electrical Characteristics $T_A = 0°C$ to $+70°C$, $V_{DD} = +5V \pm 10\%$

Symbol	Parameter	Conditions	Min	Max	Units
t_{ADS}	Address Strobe Width		60		ns
t_{AH}	Address Hold Time		0		ns
t_{AR}	$\overline{RD}$, RD Delay from Address	(Note 1)	30		ns
t_{AS}	Address Setup Time		60		ns
t_{AW}	$\overline{WR}$, WR Delay from Address	(Note 1)	30		ns
t_{CH}	Chip Select Hold Time		0		ns
t_{CS}	Chip Select Setup Time		60		ns
t_{CSR}	$\overline{RD}$, RD Delay from Chip Select	(Note 1)	30		ns
t_{CSW}	$\overline{WR}$, WR Delay from Select	(Note 1)	30		ns
t_{DH}	Data Hold Time		30		ns
t_{DS}	Data Setup Time		30		ns
t_{HZ}	$\overline{RD}$, RD to Floating Data Delay	@100 pF loading (Note 3)	0	100	ns
t_{MR}	Master Reset Pulse Width		5000		ns
t_{RA}	Address Hold Time from $\overline{RD}$, RD	(Note 1)	20		ns
t_{RC}	Read Cycle Delay		125		ns
t_{RCS}	Chip Select Hold Time from $\overline{RD}$, RD	(Note 1)	20		ns
t_{RD}	$\overline{RD}$, RD Strobe Width		125		ns
t_{RDD}	$\overline{RD}$, RD to Driver Enable/Disable	@100 pF loading (Note 3)		60	ns
t_{RVD}	Delay from $\overline{RD}$, RD to Data	@100 pF loading		60	ns
t_{WA}	Address Hold Time from $\overline{WR}$, WR	(Note 1)	20		ns
t_{WC}	Write Cycle Delay		150		ns
t_{WCS}	Chip Select Hold Time from $\overline{WR}$, WR	(Note 1)	20		ns
t_{WR}	$\overline{WR}$, WR Strobe Width		100		ns
t_{XH}	Duration of Clock High Pulse	External Clock (8, Max.)	55		ns
t_{XL}	Duration of Clock Low Pulse	External Clock (8, Max.)	55		ns
RC	Read Cycle = $t_{AR} + t_{RD} + t_{RC}$		280		ns
WC	Write Cycle = $t_{AW} + t_{WR} + t_{WC}$		280		ns
Baud Generator					
N	Baud Divisor		1	$2^{16}-1$	
t_{BHD}	Baud Output Positive Edge Delay	100 pF Load		175	ns
t_{BLD}	Baud Output Negative Edge Delay	100 pF Load		175	ns
t_{HW}	Baud Output Up Time	$f_X = 8, \div 2$, 100 pF Load	75		ns
t_{LW}	Baud Output Down Time	$f_X = 8, \div 2$, 100 pF Load	100		ns
Receiver					
t_{RAI}	Delay from Active Edge of $\overline{RD}$ to Reset Interrupt			—	ns
t_{RINT}	Delay from $\overline{RD}$, RD (RD RBR/or RD LSR) to Reset Interrupt	100 pF Load		1000	ns
t_{RXI}	Delay from $\overline{RD}$ RBR to $\overline{RXRDY}$ Inactive			290	ns
t_{SCD}	Delay from RCLK to Sample Time			2000	ns
t_{SINT}	Delay from Stop to Set Interrupt	(Note 2)		1	RCLK Cycles

Note 1: Applicable only when $\overline{ADS}$ is tied low.

Note 2: In the FIFO mode (FCR0 = 1) the trigger level interrupts, the receiver data available indication, the active RXRDY indication and the overrun error indication will be delayed 3 RCLKs. Status indicators (PE, FE, BI) will be delayed 3 RCLKs after the first byte has been received. For subsequently received bytes these indicators will be updated immediately after RDRBR goes inactive. Timeout interrupt is delayed 8 RCLKs.

Note 3: Charge and discharge time is determined by V_{OL}, V_{OH} and the external loading.

Note 4: These specifications are preliminary.

3.0 AC Electrical Characteristics (Continued)

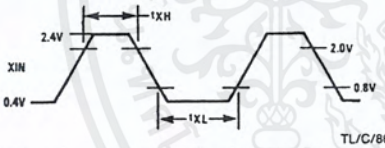
Symbol	Parameter	Conditions	Min	Max	Units
Transmitter					
t_{HR}	Delay from $\overline{WR}$, WR (WR THR) to Reset Interrupt	100 pF Load		175	ns
t_{IR}	Delay from $\overline{RD}$, RD (RD IIR) to Reset Interrupt (THRE)	100 pF Load		250	ns
t_{IRS}	Delay from Initial INTR Reset to Transmit Start		8	24	BAUDOUT Cycles
t_{SI}	Delay from Initial Write to Interrupt	(Note 1)	16	24	BAUDOUT Cycles
t_{STI}	Delay from Stop to Interrupt (THRE)	(Note 1)	8	8	BAUDOUT Cycles
t_{SXA}	Delay from Start to TXRDY active	100 pF Load		8	BAUDOUT Cycles
t_{WXI}	Delay from Write to TXRDY inactive	100 pF Load		195	ns
Modem Control					
t_{MDO}	Delay from $\overline{WR}$, WR (WR MCR) to Output	100 pF Load		200	ns
t_{RIM}	Delay from $\overline{RD}$, RD to Reset Interrupt (RD MSR)	100 pF Load		250	ns
t_{SIM}	Delay from MODEM Input to Set Interrupt	100 pF Load		250	ns

Note 1: This delay will be lengthened by 1 character time, minus the last stop bit time if the transmitter interrupt delay circuit is active. (See FIFO Interrupt Mode Operation).

Note 2: These specifications are preliminary.

4.0 Timing Waveforms (All timings are referenced to valid 0 and valid 1)

External Clock Input (24.0 MHz Max.)



TL/C/8652-2

AC Test Points

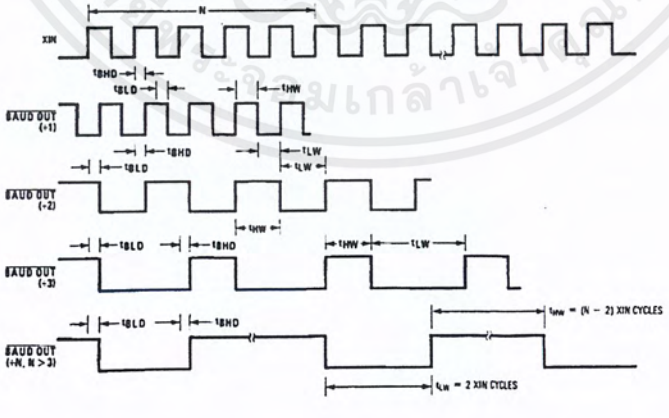


TL/C/8652-3

Note 1: The 2.4V and 0.4V levels are the voltages that the inputs are driven to during AC testing.

Note 2: The 2.0V and 0.8V levels are the voltages at which the timing tests are made.

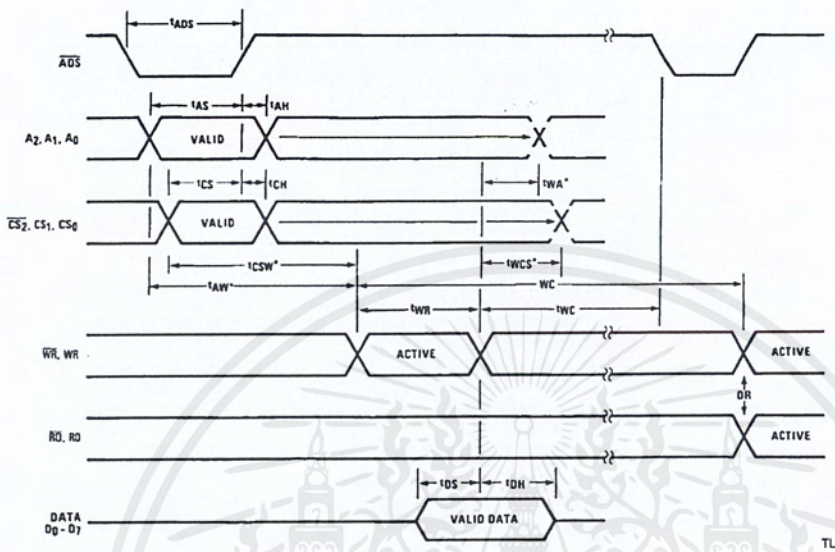
$\overline{BAUDOUT}$ Timing



TL/C/8652-4

4.0 Timing Waveforms (Continued)

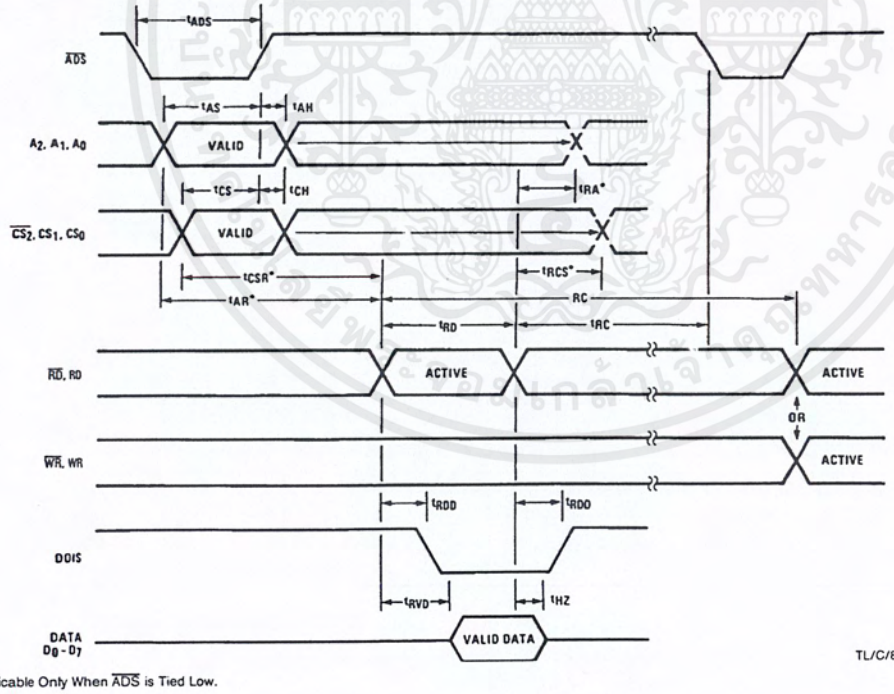
Write Cycle



TL/C/8652–5

*Applicable Only When ADS is Tied Low.

Read Cycle

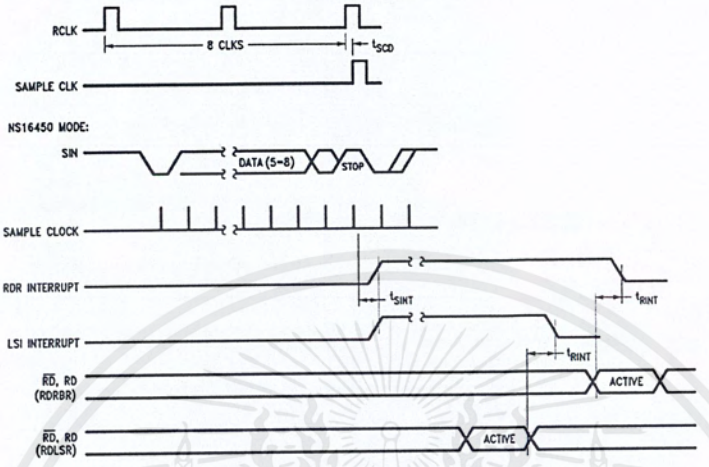


TL/C/8652–6

*Applicable Only When ADS is Tied Low.

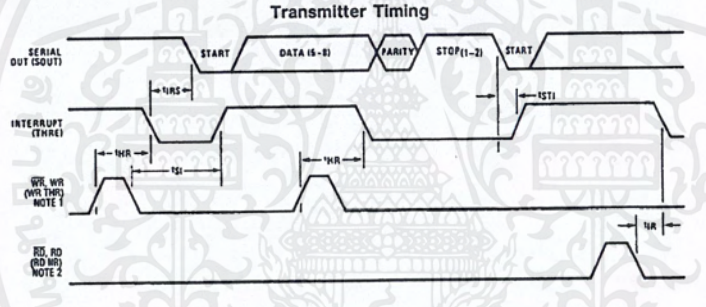
4.0 Timing Waveforms (Continued)

Receiver Timing



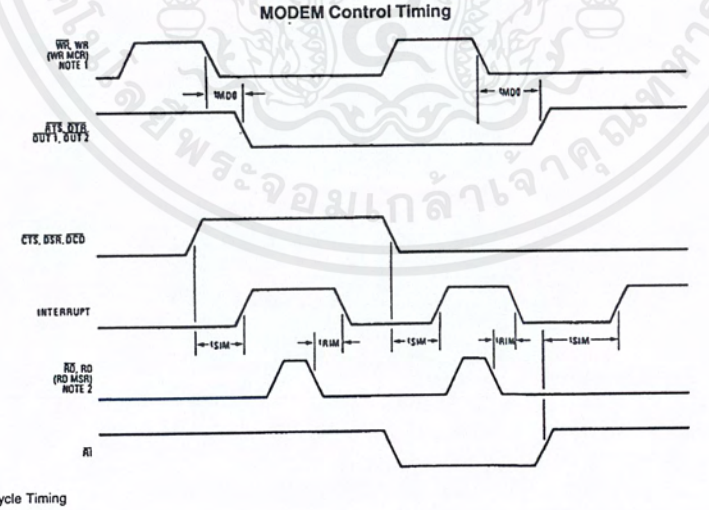
TL/C/8652–7

Transmitter Timing



TL/C/8652–8

MODEM Control Timing



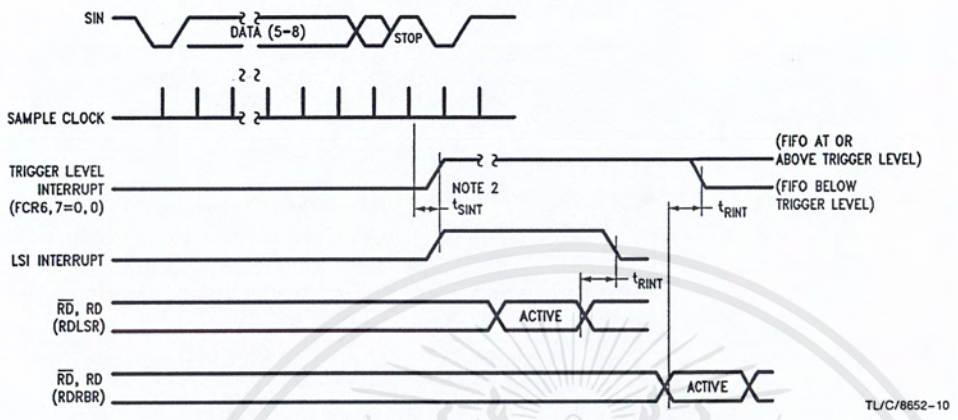
TL/C/8652–9

Note 1: See Write Cycle Timing

Note 2: See Read Cycle Timing

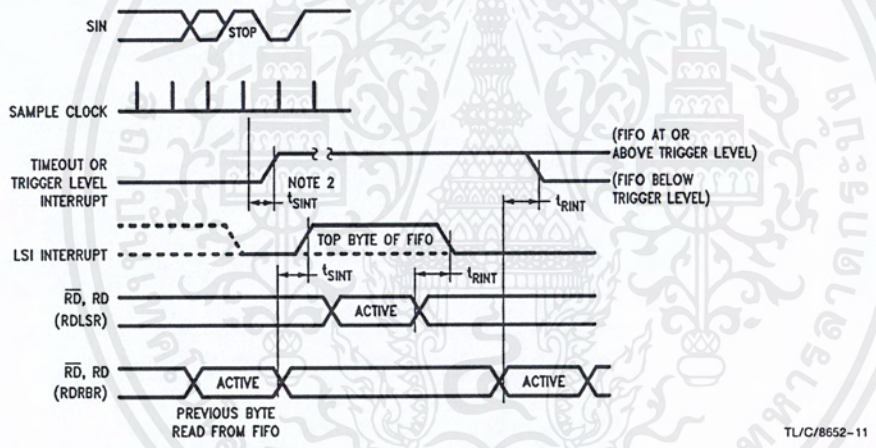
4.0 Timing Waveforms (Continued)

RCVR FIFO First Byte (This Sets RDR)



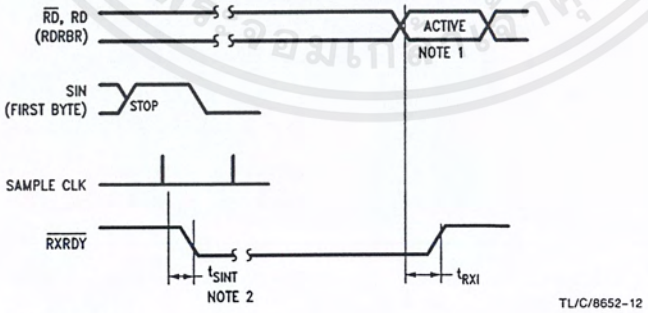
TL/C/8652–10

RCVR FIFO Bytes Other Than the First Byte (RDR Is Already Set)



TL/C/8652–11

Receiver Ready (Pin 29) FCR0 = 0 or FCR0 = 1 and FCR3 = 0 (Mode 0)



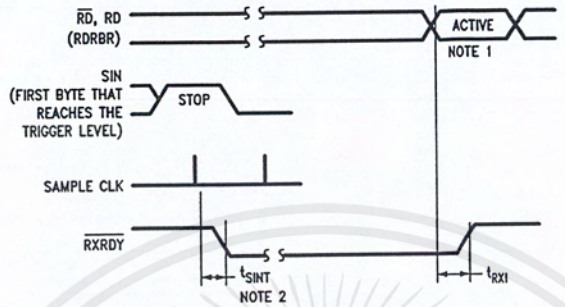
TL/C/8652–12

Note 1: This is the reading of the last byte in the FIFO.

Note 2: If FCR0 = 1, then t_{SINT} = 3 RCLKs. For a timeout interrupt, t_{SINT} = 8 RCLKs.

4.0 Timing Waveforms (Continued)

Receiver Ready (Pin 29) FCR0 = 1 and FCR3 = 1 (Mode 1)

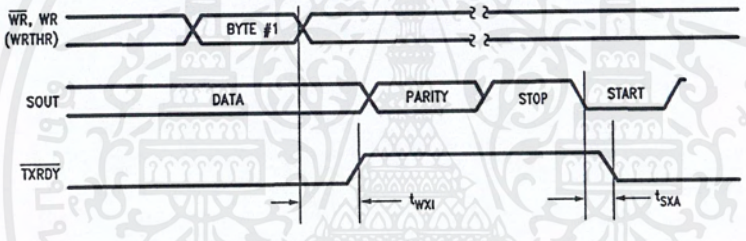


TL/C/8652–13

Note 1: This is the reading of the last byte in the FIFO.

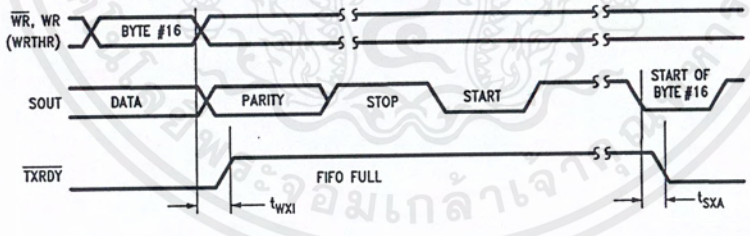
Note 2: If FCR0 = 1, t_{SINT} = 3 RCLKs.

Transmitter Ready (Pin 24) FCR0 = 0 or FCR0 = 1 and FCR3 = 0 (Mode 0)



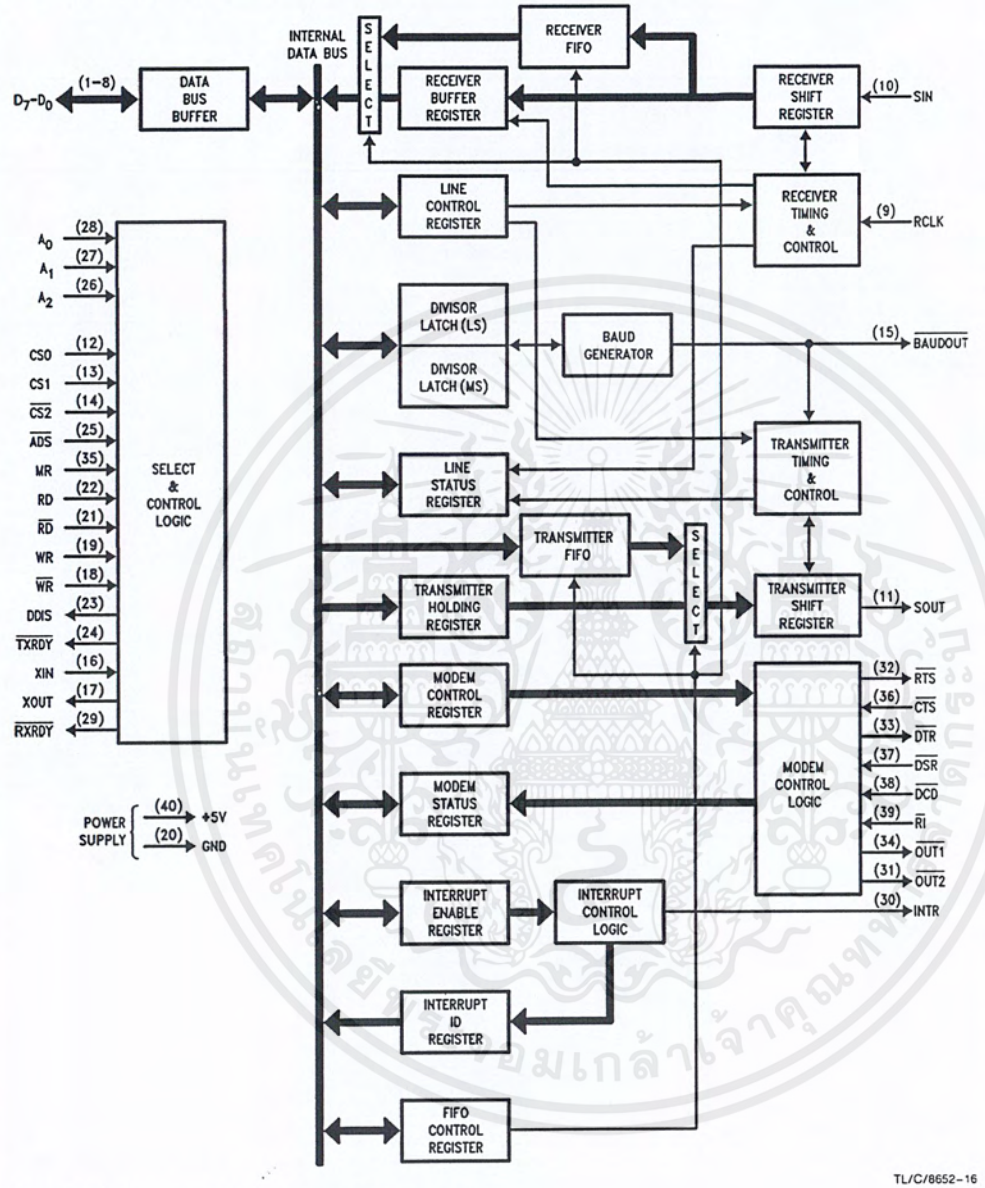
TL/C/8652–14

Transmitter Ready (Pin 24) FCR0 = 1 and FCR3 = 1 (Mode 1)



TL/C/8652–15

5.0 Block Diagram



TL/C/8652-16

Note: Applicable pinout numbers are included within parenthesis.

6.0 Pin Descriptions

The following describes the function of all UART pins. Some of these descriptions reference internal circuits.

In the following descriptions, a low represents a logic 0 (0V nominal) and a high represents a logic 1 (+2.4V nominal).

A0, A1, A2, Register Select, Pins 26–28: Address signals connected to these 3 inputs select a UART register for the CPU to read from or write to during data transfer. A table of registers and their addresses is shown below. Note that the state of the Divisor Latch Access Bit (DLAB), which is the most significant bit of the Line Control Register, affects the selection of certain UART registers. The DLAB must be set high by the system software to access the Baud Generator Divisor Latches.

Register Addresses

DLAB	A_2	A_1	A_0	Register
0	0	0	0	Receiver Buffer (read), Transmitter Holding Register (write)
0	0	0	1	Interrupt Enable
X	0	1	0	Interrupt Identification (read)
X	0	1	0	FIFO Control (write)
X	0	1	1	Line Control
X	1	0	0	MODEM Control
X	1	0	1	Line Status
X	1	1	0	MODEM Status
X	1	1	1	Scratch
1	0	0	0	Divisor Latch (least significant byte)
1	0	0	1	Divisor Latch (most significant byte)

$\overline{ADS}$, Address Strobe, Pin 25: The positive edge of an active Address Strobe ($\overline{ADS}$) signal latches the Register Select (A0, A1, A2) and Chip Select (CS0, CS1, CS2) signals.

Note: An active $\overline{ADS}$ input is required when the Register Select (A0, A1, A2) and Chip Select (CS0, CS1, CS2) signals are not stable for the duration of a read or write operation. If not required, tie the $\overline{ADS}$ input permanently low.

$\overline{BAUDOUT}$, Baud Out, Pin 15: This is the 16 × clock signal from the transmitter section of the UART. The clock rate is equal to the main reference oscillator frequency divided by the specified divisor in the Baud Generator Divisor Latches. The $\overline{BAUDOUT}$ may also be used for the receiver section by tying this output to the RCLK input of the chip.

CS0, CS1, $\overline{CS2}$, Chip Select, Pins 12–14: When CS0 and CS1 are high and $\overline{CS2}$ is low, the chip is selected. This enables communication between the UART and the CPU. The positive edge of an active Address Strobe signal latches the decoded chip select signals, completing chip selection. If $\overline{ADS}$ is always low, valid chip selects should stabilize according to the t_{CSW} parameter.

$\overline{CTS}$, Clear to Send, Pin 36: When low, this indicates that the MODEM or data set is ready to exchange data. The $\overline{CTS}$ signal is a MODEM status input whose conditions can be tested by the CPU reading bit 4 (CTS) of the MODEM Status Register. Bit 4 is the complement of the $\overline{CTS}$ signal. Bit 0 (DCTS) of the MODEM Status Register indicates whether the $\overline{CTS}$ input has changed state since the previous reading of the MODEM Status Register. $\overline{CTS}$ has no effect on the Transmitter.

Note: Whenever the CTS bit of the MODEM Status Register changes state, an interrupt is generated if the MODEM Status Interrupt is enabled.

D_7–D_0, Data Bus, Pins 1–8: This bus comprises eight TRI-STATE input/output lines. The bus provides bidirectional communications between the UART and the CPU. Data, control words, and status information are transferred via the D_7–D_0 Data Bus.

$\overline{DCD}$, Data Carrier Detect, Pin 38: When low, indicates that the data carrier has been detected by the MODEM or data set. The $\overline{DCD}$ signal is a MODEM status input whose condition can be tested by the CPU reading bit 7 (DCD) of the MODEM Status Register. Bit 7 is the complement of the $\overline{DCD}$ signal. Bit 3 (DDCD) of the MODEM Status Register indicates whether the $\overline{DCD}$ input has changed state since the previous reading of the MODEM Status Register. $\overline{DCD}$ has no effect on the receiver.

Note: Whenever the DCD bit of the MODEM Status Register changes state, an interrupt is generated if the MODEM Status Interrupt is enabled.

DDIS, Driver Disable, Pin 23: This goes low whenever the CPU is reading data from the UART. It can disable or control the direction of a data bus transceiver between the CPU and the UART.

$\overline{DSR}$, Data Set Ready, Pin 37: When low, this indicates that the MODEM or data set is ready to establish the communications link with the UART. The $\overline{DSR}$ signal is a MODEM status input whose condition can be tested by the CPU reading bit 5 (DSR) of the MODEM Status Register. Bit 5 is the complement of the $\overline{DSR}$ signal. Bit 1 (DDSR) of the MODEM Status Register indicates whether the $\overline{DSR}$ input has changed state since the previous reading of the MODEM Status Register.

Note: Whenever the DDSR bit of the MODEM Status Register changes state, an interrupt is generated if the MODEM Status Interrupt is enabled.

$\overline{DTR}$, Data Terminal Ready, Pin 33: When low, this informs the MODEM or data set that the UART is ready to establish a communications link. The $\overline{DTR}$ output signal can be set to an active low by programming bit 0 (DTR) of the MODEM Control Register to a high level. A Master Reset operation sets this signal to its inactive (high) state. Loop mode operation holds this signal in its inactive state.

INTR, Interrupt, Pin 30: This pin goes high whenever any one of the following interrupt types has an active high condition and is enabled via the IER: Receiver Error Flag; Received Data Available: timeout (FIFO Mode only); Transmitter Holding Register Empty; and MODEM Status. The INTR signal is reset low upon the appropriate interrupt service or a Master Reset operation.

MR, Master Reset, Pin 35: When this input is high, it clears all the registers (except the Receiver Buffer, Transmitter Holding, and Divisor Latches), and the control logic of the UART. The states of various output signals (SOUT, INTR, $\overline{OUT\ 1}$, $\overline{OUT\ 2}$, $\overline{RTS}$, $\overline{DTR}$) are affected by an active MR input (Refer to Table I.) This input is buffered with a TTL-compatible Schmitt Trigger with 0.5V typical hysteresis.

$\overline{OUT\ 1}$, Output 1, Pin 34: This user-designated output can be set to an active low by programming bit 2 (OUT 1) of the MODEM Control Register to a high level. A Master Reset operation sets this signal to its inactive (high) state. Loop mode operation holds this signal in its inactive state. In the XMOS parts this will achieve TTL levels.

6.0 Pin Descriptions (Continued)

$\overline{OUT\ 2}$, Output 2, Pin 31: This user-designated output that can be set to an active low by programming bit 3 (OUT 2) of the MODEM Control Register to a high level. A Master Reset operation sets this signal to its inactive (high) state. Loop mode operation holds this signal in its inactive state. In the XMOS parts this will achieve TTL levels.

RCLK, Receiver Clock, Pin 9: This input is the 16 × baud rate clock for the receiver section of the chip.

RD, $\overline{RD}$, Read, Pins 22 and 21: When RD is high or $\overline{RD}$ is low while the chip is selected, the CPU can read status information or data from the selected UART register.

Note: Only an active RD or $\overline{RD}$ input is required to transfer data from the UART during a read operation. Therefore, tie either the RD input permanently low or the $\overline{RD}$ input permanently high, when it is not used.

$\overline{RI}$, Ring Indicator, Pin 39: When low, this indicates that a telephone ringing signal has been received by the MODEM or data set. The $\overline{RI}$ signal is a MODEM status input whose condition can be tested by the CPU reading bit 6 (RI) of the MODEM Status Register. Bit 6 is the complement of the $\overline{RI}$ signal. Bit 2 (TERI) of the MODEM Status Register indicates whether the $\overline{RI}$ input signal has changed from a low to a high state since the previous reading of the MODEM Status Register.

Note: Whenever the RI bit of the MODEM Status Register changes from a high to a low state, an interrupt is generated if the MODEM Status Interrupt is enabled.

$\overline{RTS}$, Request to Send, Pin 32: When low, this informs the MODEM or data set that the UART is ready to exchange data. The $\overline{RTS}$ output signal can be set to an active low by programming bit 1 (RTS) of the MODEM Control Register. A Master Reset operation sets this signal to its inactive (high) state. Loop mode operation holds this signal in its inactive state.

SIN, Serial Input, Pin 10: Serial data input from the communications link (peripheral device, MODEM, or data set).

SOUT, Serial Output, Pin 11: Composite serial data output to the communications link (peripheral, MODEM or data set). The SOUT signal is set to the Marking (logic 1) state upon a Master Reset operation.

$\overline{TXRDY}$, $\overline{RXRDY}$, Pins 24, 29: Transmitter and Receiver DMA signalling is available through two pins (24 and 29). When operating in the FIFO mode, one of two types of DMA signalling per pin can be selected via FCR3. When operating as in the 16450 Mode, only DMA mode 0 is allowed. Mode 0 supports single transfer DMA where a transfer is made between CPU bus cycles. Mode 1 supports multi-transfer DMA where multiple transfers are made continuously until the RCVR FIFO has been emptied or the XMIT FIFO has been filled.

RXRDY, Mode 0: When in the 16450 Mode (FCR0 = 0) or in the FIFO Mode (FCR0 = 1, FCR3 = 0) and there is at least 1 character in the RCVR FIFO or RCVR holding register, the RXRDY pin (29) will be low active. Once it is activated the RXRDY pin will go inactive when there are no more characters in the FIFO or holding register.

RXRDY, Mode 1: In the FIFO Mode (FCR0 = 1) when the FCR3 = 1 and the trigger level or the timeout has been reached, the RXRDY pin will go low active. Once it is activated it will go inactive when there are no more characters in the FIFO or holding register.

TXRDY, Mode 0: In the 16450 Mode (FCR0 = 0) or in the FIFO Mode (FCR0 = 1, FCR3 = 0) and there are no characters in the XMIT FIFO or XMIT holding register, the TXRDY pin (24) will be low active. Once it is activated the TXRDY pin will go inactive after the first character is loaded into the XMIT FIFO or holding register.

TXRDY, Mode 1: In the FIFO Mode (FCR0 = 1) when FCR3 = 1 and there are no characters in the XMIT FIFO, the $\overline{TXRDY}$ pin will go low active. This pin will become inactive when the XMIT FIFO is completely full.

V_{DD}, Pin 40: +5V supply.

V_{SS}, Pin 20: Ground (0V) reference.

WR, $\overline{WR}$, Write, Pins 19 and 18: When WR is high or $\overline{WR}$ is low while the chip is selected, the CPU can write control words or data into the selected UART register.

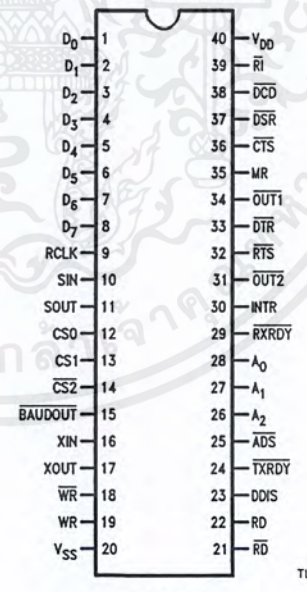
Note: Only an active WR or $\overline{WR}$ input is required to transfer data to the UART during a write operation. Therefore, tie either the WR input permanently low or the $\overline{WR}$ input permanently high, when it is not used.

XIN (External Crystal Input), Pin 16: This signal input is used in conjunction with XOUT to form a feedback circuit for the baud rate generator's oscillator. If a clock signal will be generated off-chip, then it should drive the baud rate generator through this pin.

XOUT (External Crystal Output), Pin 17: This signal output is used in conjunction with XIN to form a feedback circuit for the baud rate generator's oscillator. If the clock signal will be generated off-chip, then this pin is unused.

7.0 Connection Diagrams

Dual-In-Line Package



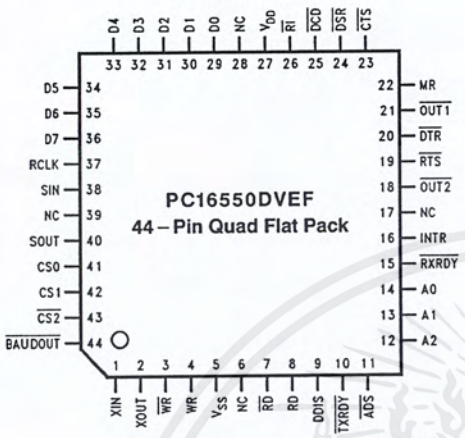
TL/C/8652–17

Top View

Order Number PC16550DN
See NS Package Number N40A

7.0 Connection Diagrams (Continued)

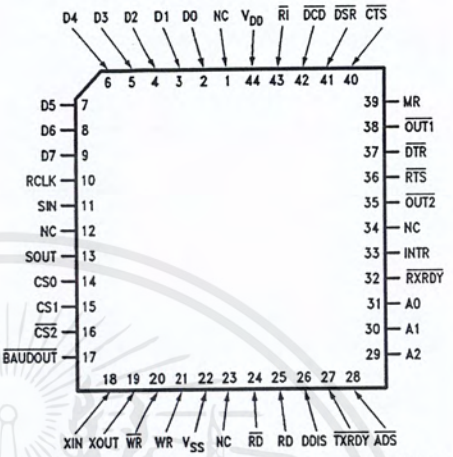
TQFP Package



TL/C/8652–26

Order Number PC16550DVEF
See NS Package Number VEF44A

Chip Carrier Package



TL/C/8652–18

Top View

Order Number PC16550DV
See NS Package Number V44A

TABLE I. UART Reset Configuration

Register/Signal	Reset Control	Reset State
Interrupt Enable Register	Master Reset	**0000** 0000 (Note 1)
Interrupt Identification Register	Master Reset	00**00** 0001
FIFO Control	Master Reset	00**00** 0000
Line Control Register	Master Reset	0000 0000
MODEM Control Register	Master Reset	**0000** 0000
Line Status Register	Master Reset	0110 0000
MODEM Status Register	Master Reset	XXXX 0000 (Note 2)
SOUT	Master Reset	High
INTR (RCVR Errs)	Read LSR/MR	Low
INTR (RCVR Data Ready)	Read RBR/MR	Low
INTR (THRE)	Read IIR/Write THR/MR	Low
INTR (Modem Status Changes)	Read MSR/MR	Low
$\overline{OUT\,2}$	Master Reset	High
$\overline{RTS}$	Master Reset	High
$\overline{DTR}$	Master Reset	High
$\overline{OUT\,1}$	Master Reset	High
RCVR FIFO	MR/FCR1•FCR0/ΔFCR0	All Bits Low
XMIT FIFO	MR/FCR1•FCR0/ΔFCR0	All Bits Low

Note 1: Boldface bits are permanently low.

Note 2: Bits 7–4 are driven by the input signals.

TABLE II. Summary of Registers

Bit No.	Register Address											
	0 DLAB = 0	0 DLAB = 0	1 DLAB = 0	2	2	3	4	5	6	7	0 DLAB = 1	1 DLAB = 1
	Receiver Buffer Register (Read Only)	Transmitter Holding Register (Write Only)	Interrupt Enable Register	Interrupt Ident. Register (Read Only)	FIFO Control Register (Write Only)	Line Control Register	MODEM Control Register	Line Status Register	MODEM Status Register	Scratch Register	Divisor Latch (LS)	Divisor Latch (MS)
	RBR	THR	IER	IIR	FCR	LCR	MCR	LSR	MSR	SCR	DLL	DLM
0	Data Bit 0 (Note 1)	Data Bit 0	Enable Received Data Available Interrupt (ERBFI)	"0" if Interrupt Pending	FIFO Enable	Word Length Select Bit 0 (WLS0)	Data Terminal Ready (DTR)	Data Ready (DR)	Delta Clear to Send (DCTS)	Bit 0	Bit 0	Bit 8
1	Data Bit 1	Data Bit 1	Enable Transmitter Holding Register Empty Interrupt (ETBEI)	Interrupt ID Bit (0)	RCVR FIFO Reset	Word Length Select Bit 1 (WLS1)	Request to Send (RTS)	Overrun Error (OE)	Delta Data Set Ready (DDSR)	Bit 1	Bit 1	Bit 9
2	Data Bit 2	Data Bit 2	Enable Receiver Line Status Interrupt (ELSI)	Interrupt ID Bit (1)	XMIT FIFO Reset	Number of Stop Bits (STB)	Out 1	Parity Error (PE)	Trailing Edge Ring Indicator (TERI)	Bit 2	Bit 2	Bit 10
3	Data Bit 3	Data Bit 3	Enable MODEM Status Interrupt (EDSSI)	Interrupt ID Bit (2) (Note 2)	DMA Mode Select	Parity Enable (PEN)	Out 2	Framing Error (FE)	Delta Data Carrier Detect (DDCD)	Bit 3	Bit 3	Bit 11
4	Data Bit 4	Data Bit 4	0	0	Reserved	Even Parity Select (EPS)	Loop	Break Interrupt (BI)	Clear to Send (CTS)	Bit 4	Bit 4	Bit 12
5	Data Bit 5	Data Bit 5	0	0	Reserved	Stick Parity	0	Transmitter Holding Register (THRE)	Data Set Ready (DSR)	Bit 5	Bit 5	Bit 13
6	Data Bit 6	Data Bit 6	0	FIFOs Enabled (Note 2)	RCVR Trigger (LSB)	Set Break	0	Transmitter Empty (TEMT)	Ring Indicator (RI)	Bit 6	Bit 6	Bit 14
7	Data Bit 7	Data Bit 7	0	FIFOs Enabled (Note 2)	RCVR Trigger (MSB)	Divisor Latch Access Bit (DLAB)	0	Error in RCVR FIFO (Note 2)	Data Carrier Detect (DCD)	Bit 7	Bit 7	Bit 15

Note 1: Bit 0 is the least significant bit. It is the first bit serially transmitted or received.

Note 2: These bits are always 0 in the 16450 Mode.

8.0 Registers

The system programmer may access any of the UART registers summarized in Table II via the CPU. These registers control UART operations including transmission and reception of data. Each register bit in Table II has its name and reset state shown.

8.1 LINE CONTROL REGISTER

The system programmer specifies the format of the asynchronous data communications exchange and set the Divisor Latch Access bit via the Line Control Register (LCR). The programmer can also read the contents of the Line Control Register. The read capability simplifies system programming and eliminates the need for separate storage in system memory of the line characteristics. Table II shows the contents of the LCR. Details on each bit follow:

Bits 0 and 1: These two bits specify the number of bits in each transmitted or received serial character. The encoding of bits 0 and 1 is as follows:

Bit 1	Bit 0	Character Length
0	0	5 Bits
0	1	6 Bits
1	0	7 Bits
1	1	8 Bits

Bit 2: This bit specifies the number of Stop bits transmitted and received in each serial character. If bit 2 is a logic 0, one Stop bit is generated in the transmitted data. If bit 2 is a logic 1 when a 5-bit word length is selected via bits 0 and 1, one and a half Stop bits are generated. If bit 2 is a logic 1 when either a 6-, 7-, or 8-bit word length is selected, two Stop bits are generated. The Receiver checks the first Stop-bit only, regardless of the number of Stop bits selected.

Bit 3: This bit is the Parity Enable bit. When bit 3 is a logic 1, a Parity bit is generated (transmit data) or checked (receive data) between the last data word bit and Stop bit of the serial data. (The Parity bit is used to produce an even or odd number of 1s when the data word bits and the Parity bit are summed.)

Bit 4: This bit is the Even Parity Select bit. When bit 3 is a logic 1 and bit 4 is a logic 0, an odd number of logic 1s is transmitted or checked in the data word bits and Parity bit. When bit 3 is a logic 1 and bit 4 is a logic 1, an even number of logic 1s is transmitted or checked.

Bit 5: This bit is the Stick Parity bit. When bits 3, 4 and 5 are logic 1 the Parity bit is transmitted and checked as a logic 0. If bits 3 and 5 are 1 and bit 4 is a logic 0 then the Parity bit is transmitted and checked as a logic 1. If bit 5 is a logic 0 Stick Parity is disabled.

Bit 6: This bit is the Break Control bit. It causes a break condition to be transmitted to the receiving UART. When it is set to a logic 1, the serial output (SOUT) is forced to the Spacing (logic 0) state. The break is disabled by setting bit 6 to a logic 0. The Break Control bit acts only on SOUT and has no effect on the transmitter logic.

Note: This feature enables the CPU to alert a terminal in a computer communications system. If the following sequence is followed, no erroneous or extraneous characters will be transmitted because of the break.

1. Load an all 0s, pad character, in response to THRE.
2. Set break after the next THRE.
3. Wait for the transmitter to be idle, (TEMT = 1), and clear break when normal transmission has to be restored.

During the break, the Transmitter can be used as a character timer to accurately establish the break duration.

TABLE III. Baud Rates, Divisors and Crystals

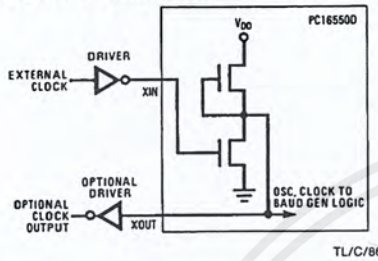
Baud Rate	1.8432 MHz Cystal		3.072 MHz Crystal		18.432 MHz Crystal	
	Decimal Divisor for 16 × Clock	Percent Error	Decimal Divisor for 16 × Clock	Percent Error	Decimal Divisor for 16 × Clock	Percent Error
50	2304	—	3840	—	23040	—
75	1536	—	2560	—	15360	—
110	1047	0.026	1745	0.026	10473	—
134.5	857	0.058	1428	0.034	8565	—
150	768	—	1280	—	7680	—
300	384	—	640	—	3840	—
600	192	—	320	—	1920	—
1200	96	—	160	—	920	—
1800	64	—	107	0.312	640	—
2000	58	0.69	96	—	576	—
2400	48	—	80	—	480	—
3600	32	—	53	0.628	320	—
4800	24	—	40	—	240	—
7200	16	—	27	1.23	160	—
9600	12	—	20	—	120	—
19200	6	—	10	—	60	—
38400	3	—	5	—	30	—
56000	2	2.86	—	—	21	2.04
128000	—	—	—	—	9	—

Note: For baud rates of 250k, 300k, 375k, 500k, 750k and 1.5M using a 24 MHz crystal causes minimal error.

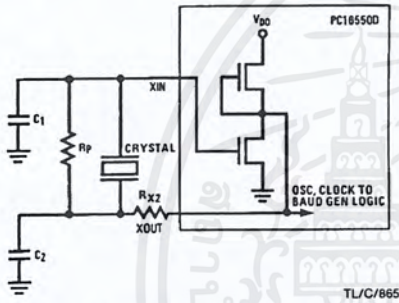
8.0 Registers (Continued)

Bit 7: This bit is the Divisor Latch Access Bit (DLAB). It must be set high (logic 1) to access the Divisor Latches of the Baud Generator during a Read or Write operation. It must be set low (logic 0) to access the Receiver Buffer, the Transmitter Holding Register, or the Interrupt Enable Register.

8.2 TYPICAL CLOCK CIRCUITS



TL/C/8652–19



TL/C/8652–20

Typical Crystal Oscillator Network (Note)

CRYSTAL	R_P	R_{X2}	C_1	C_2
3.1 MHz	1 MΩ	1.5k	10-30 pF	40-60 pF
1.8 MHz	1 MΩ	1.5k	10-30 pF	40-60 pF

Note: These R and C values are approximate and may vary 2x depending on the crystal characteristics. All crystal circuits should be designed specifically for the system.

8.3 PROGRAMMABLE BAUD GENERATOR

The UART contains a programmable Baud Generator that is capable of taking any clock input from DC to 24 MHz and dividing it by any divisor from 2 to $2^{16}-1$. The output frequency of the Baud Generator is 16 × the Baud [divisor # = (frequency input) ÷ (baud rate × 16)]. Two 8-bit latches store the divisor in a 16-bit binary format. These Divisor Latches must be loaded during initialization to ensure proper operation of the Baud Generator. Upon loading either of the Divisor Latches, a 16-bit Baud counter is immediately loaded.

Table III provides decimal divisors to use with crystal frequencies of 1.8432 MHz, 3.072 MHz and 18.432 MHz, respectively. For baud rates of 38400 and below, the error obtained is minimal. The accuracy of the desired baud rate is dependent on the crystal frequency chosen. Using a divisor of zero is **not** recommended.

8.4 LINE STATUS REGISTER

This register provides status information to the CPU concerning the data transfer. Table II shows the contents of the Line Status Register. Details on each bit follow.

Bit 0: This bit is the receiver Data Ready (DR) indicator. Bit 0 is set to a logic 1 whenever a complete incoming character has been received and transferred into the Receiver Buffer Register or the FIFO. Bit 0 is reset to a logic 0 by reading all of the data in the Receiver Buffer Register or the FIFO.

Bit 1: This bit is the Overrun Error (OE) indicator. Bit 1 indicates that data in the Receiver Buffer Register was not read by the CPU before the next character was transferred into the Receiver Buffer Register, thereby destroying the previous character. The OE indicator is set to a logic 1 upon detection of an overrun condition and reset whenever the CPU reads the contents of the Line Status Register. If the FIFO mode data continues to fill the FIFO beyond the trigger level, an overrun error will occur only after the FIFO is full and the next character has been completely received in the shift register. OE is indicated to the CPU as soon as it happens. The character in the shift register is overwritten, but it is not transferred to the FIFO.

Bit 2: This bit is the Parity Error (PE) indicator. Bit 2 indicates that the received data character does not have the correct even or odd parity, as selected by the even-parity-select bit. The PE bit is set to a logic 1 upon detection of a parity error and is reset to a logic 0 whenever the CPU reads the contents of the Line Status Register. In the FIFO mode this error is associated with the particular character in the FIFO it applies to. This error is revealed to the CPU when its associated character is at the top of the FIFO.

Bit 3: This bit is the Framing Error (FE) indicator. Bit 3 indicates that the received character did not have a valid Stop bit. Bit 3 is set to a logic 1 whenever the Stop bit following the last data bit or parity bit is detected as a logic 0 bit (Spacing level). The FE indicator is reset whenever the CPU reads the contents of the Line Status Register. In the FIFO mode this error is associated with the particular character in the FIFO it applies to. This error is revealed to the CPU when its associated character is at the top of the FIFO. The UART will try to resynchronize after a framing error. To do this it assumes that the framing error was due to the next start bit, so it samples this "start" bit twice and then takes in the "data".

Bit 4: This bit is the Break Interrupt (BI) indicator. Bit 4 is set to a logic 1 whenever the received data input is held in the Spacing (logic 0) state for longer than a full word transmission time (that is, the total time of Start bit + data bits + Parity + Stop bits). The BI indicator is reset whenever the CPU reads the contents of the Line Status Register. In the FIFO mode this error is associated with the particular character in the FIFO it applies to. This error is revealed to the CPU when its associated character is at the top of the FIFO. When break occurs only one zero character is loaded into the FIFO. The next character transfer is enabled after SIN goes to the marking state and receives the next valid start bit.

Note: Bits 1 through 4 are the error conditions that produce a Receiver Line Status interrupt whenever any of the corresponding conditions are detected and the interrupt is enabled.

8.0 Registers (Continued)

TABLE IV. Interrupt Control Functions

FIFO Mode Only	Interrupt Identification Register			Interrupt Set and Reset Functions			
Bit 3	Bit 2	Bit 1	Bit 0	Priority Level	Interrupt Type	Interrupt Source	Interrupt Reset Control
0	0	0	1	—	None	None	—
0	1	1	0	Highest	Receiver Line Status	Overrun Error or Parity Error or Framing Error or Break Interrupt	Reading the Line Status Register
0	1	0	0	Second	Received Data Available	Receiver Data Available or Trigger Level Reached	Reading the Receiver Buffer Register or the FIFO Drops Below the Trigger Level
1	1	0	0	Second	Character Timeout Indication	No Characters Have Been Removed From or Input to the RCVR FIFO During the Last 4 Char. Times and There Is at Least 1 Char. in It During This Time	Reading the Receiver Buffer Register
0	0	1	0	Third	Transmitter Holding Register Empty	Transmitter Holding Register Empty	Reading the IIR Register (if source of interrupt) or Writing into the Transmitter Holding Register
0	0	0	0	Fourth	MODEM Status	Clear to Send or Data Set Ready or Ring Indicator or Data Carrier Detect	Reading the MODEM Status Register

Bit 5: This bit is the Transmitter Holding Register Empty (THRE) indicator. Bit 5 indicates that the UART is ready to accept a new character for transmission. In addition, this bit causes the UART to issue an interrupt to the CPU when the Transmit Holding Register Empty Interrupt enable is set high. The THRE bit is set to a logic 1 when a character is transferred from the Transmitter Holding Register into the Transmitter Shift Register. The bit is reset to logic 0 concurrently with the loading of the Transmitter Holding Register by the CPU. In the FIFO mode this bit is set when the XMIT FIFO is empty; it is cleared when at least 1 byte is written to the XMIT FIFO.

Bit 6: This bit is the Transmitter Empty (TEMT) indicator. Bit 6 is set to a logic 1 whenever the Transmitter Holding Register (THR) and the Transmitter Shift Register (TSR) are both empty. It is reset to a logic 0 whenever either the THR or TSR contains a data character. In the FIFO mode this bit is set to one whenever the transmitter FIFO and shift register are both empty.

Bit 7: In the 16450 Mode this is a 0. In the FIFO mode LSR7 is set when there is at least one parity error, framing error or break indication in the FIFO. LSR7 is cleared when the CPU reads the LSR, if there are no subsequent errors in the FIFO.

Note: The Line Status Register is intended for read operations only. Writing to this register is not recommended as this operation is only used for factory testing. In the FIFO mode the software must load a data byte in the Rx FIFO via Loopback Mode in order to write to LSR2–LSR4. LSR0 and LSR7 can't be written to in FIFO mode.

8.5 FIFO CONTROL REGISTER

This is a write only register at the same location as the IIR (the IIR is a read only register). This register is used to enable the FIFOs, clear the FIFOs, set the RCVR FIFO trigger level, and select the type of DMA signalling.

Bit 0: Writing a 1 to FCR0 enables both the XMIT and RCVR FIFOs. Resetting FCR0 will clear all bytes in both FIFOs. When changing from the FIFO Mode to the 16450 Mode and vice versa, data is automatically cleared from the FIFOs. This bit must be a 1 when other FCR bits are written to or they will not be programmed.

Bit 1: Writing a 1 to FCR1 clears all bytes in the RCVR FIFO and resets its counter logic to 0. The shift register is not cleared. The 1 that is written to this bit position is self-clearing.

Bit 2: Writing a 1 to FCR2 clears all bytes in the XMIT FIFO and resets its counter logic to 0. The shift register is not cleared. The 1 that is written to this bit position is self-clearing.

Bit 3: Setting FCR3 to a 1 will cause the RXRDY and TXRDY pins to change from mode 0 to mode 1 if FCR0 = 1 (see description of RXRDY and TXRDY pins).

Bit 4, 5: FCR4 to FCR5 are reserved for future use.

Bit 6, 7: FCR6 and FCR7 are used to set the trigger level for the RCVR FIFO interrupt.

7	6	RCVR FIFO Trigger Level (Bytes)
0	0	01
0	1	04
1	0	08
1	1	14

8.6 INTERRUPT IDENTIFICATION REGISTER

In order to provide minimum software overhead during data character transfers, the UART prioritizes interrupts into four levels and records these in the interrupt Identification Register. The four levels of interrupt conditions in order of priority are Receiver Line Status; Received Data Ready; Transmitter Holding Register Empty; and MODEM Status.

8.0 Registers (Continued)

When the CPU accesses the IIR, the UART freezes all interrupts and indicates the highest priority pending interrupt to the CPU. While this CPU access is occurring, the UART records new interrupts, but does not change its current indication until the access is complete. Table II shows the contents of the IIR. Details on each bit follow:

Bit 0: This bit can be used in a prioritized interrupt environment to indicate whether an interrupt is pending. When bit 0 is a logic 0, an interrupt is pending and the IIR contents may be used as a pointer to the appropriate interrupt service routine. When bit 0 is a logic 1, no interrupt is pending.

Bits 1 and 2: These two bits of the IIR are used to identify the highest priority interrupt pending as indicated in Table IV.

Bit 3: In the 16450 Mode this bit is 0. In the FIFO mode this bit is set along with bit 2 when a timeout interrupt is pending.

Bits 4 and 5: These two bits of the IIR are always logic 0.

Bits 6 and 7: These two bits are set when FCR0 = 1.

8.7 INTERRUPT ENABLE REGISTER

This register enables the five types of UART interrupts. Each interrupt can individually activate the interrupt (INTR) output signal. It is possible to totally disable the interrupt system by resetting bits 0 through 3 of the Interrupt Enable Register (IER). Similarly, setting bits of the IER register to a logic 1, enables the selected interrupt(s). Disabling an interrupt prevents it from being indicated as active in the IIR and from activating the INTR output signal. All other system functions operate in their normal manner, including the setting of the Line Status and MODEM Status Registers. Table II shows the contents of the IER. Details on each bit follow.

Bit 0: This bit enables the Received Data Available Interrupt (and timeout interrupts in the FIFO mode) when set to logic 1.

Bit 1: This bit enables the Transmitter Holding Register Empty Interrupt when set to logic 1.

Bit 2: This bit enables the Receiver Line Status Interrupt when set to logic 1.

Bit 3: This bit enables the MODEM Status Interrupt when set to logic 1.

Bits 4 through 7: These four bits are always logic 0.

8.8 MODEM CONTROL REGISTER

This register controls the interface with the MODEM or data set (or a peripheral device emulating a MODEM). The contents of the MODEM Control Register are indicated in Table II and are described below.

Bit 0: This bit controls the Data Terminal Ready ($\overline{DTR}$) output. When bit 0 is set to a logic 1, the $\overline{DTR}$ output is forced to a logic 0. When bit 0 is reset to a logic 0, the $\overline{DTR}$ output is forced to a logic 1.

Note: The $\overline{DTR}$ output of the UART may be applied to an EIA inverting line driver (such as the DS1488) to obtain the proper polarity input at the succeeding MODEM or data set.

Bit 1: This bit controls the Request to Send ($\overline{RTS}$) output. Bit 1 affects the $\overline{RTS}$ output in a manner identical to that described above for bit 0.

Bit 2: This bit controls the Output 1 ($\overline{OUT\ 1}$) signal, which is an auxiliary user-designated output. Bit 2 affects the $\overline{OUT\ 1}$ output in a manner identical to that described above for bit 0.

Bit 3: This bit controls the Output 2 ($\overline{OUT\ 2}$) signal, which is an auxiliary user-designated output. Bit 3 affects the $\overline{OUT\ 2}$ output in a manner identical to that described above for bit 0.

Bit 4: This bit provides a local loopback feature for diagnostic testing of the UART. When bit 4 is set to logic 1, the following occur: the transmitter Serial Output (SOUT) is set to the Marking (logic 1) state; the receiver Serial Input (SIN) is disconnected; the output of the Transmitter Shift Register is "looped back" into the Receiver Shift Register input; the four MODEM Control inputs ($\overline{DSR}$, $\overline{CTS}$, $\overline{RI}$, and $\overline{DCD}$) are disconnected; and the four MODEM Control outputs ($\overline{DTR}$, $\overline{RTS}$, $\overline{OUT\ 1}$, and $\overline{OUT\ 2}$) are internally connected to the four MODEM Control inputs, and the MODEM Control output pins are forced to their inactive state (high). In the loopback mode, data that is transmitted is immediately received. This feature allows the processor to verify the transmit-and received-data paths of the UART.

In the loopback mode, the receiver and transmitter interrupts are fully operational. Their sources are external to the part. The MODEM Control Interrupts are also operational, but the interrupts' sources are now the lower four bits of the MODEM Control Register instead of the four MODEM Control inputs. The interrupts are still controlled by the Interrupt Enable Register.

Bits 5 through 7: These bits are permanently set to logic 0.

8.9 MODEM STATUS REGISTER

This register provides the current state of the control lines from the MODEM (or peripheral device) to the CPU. In addition to this current-state information, four bits of the MODEM Status Register provide change information. These bits are set to a logic 1 whenever a control input from the MODEM changes state. They are reset to logic 0 whenever the CPU reads the MODEM Status Register.

The contents of the MODEM Status Register are indicated in Table II and described below.

Bit 0: This bit is the Delta Clear to Send (DCTS) indicator. Bit 0 indicates that the $\overline{CTS}$ input to the chip has changed state since the last time it was read by the CPU.

Bit 1: This bit is the Delta Data Set Ready (DDSR) indicator. Bit 1 indicates that the $\overline{DSR}$ input to the chip has changed state since the last time it was read by the CPU.

Bit 2: This bit is the Trailing Edge of Ring Indicator (TERI) detector. Bit 2 indicates that the $\overline{RI}$ input to the chip has changed from a low to a high state.

Bit 3: This bit is the Delta Data Carrier Detect (DDCD) indicator. Bit 3 indicates that the $\overline{DCD}$ input to the chip has changed state.

Note: Whenever bit 0, 1, 2, or 3 is set to logic 1, a MODEM Status Interrupt is generated.

Bit 4: This bit is the complement of the Clear to Send ($\overline{CTS}$) input. If bit 4 (loop) of the MCR is set to a 1, this bit is equivalent to RTS in the MCR.

Bit 5: This bit is the complement of the Data Set Ready ($\overline{DSR}$) input. If bit 4 of the MCR is set to a 1, this bit is equivalent to DTR in the MCR.

Bit 6: This bit is the complement of the Ring Indicator ($\overline{RI}$) input. If bit 4 of the MCR is set to a 1, this bit is equivalent to OUT 1 in the MCR.

8.0 Registers (Continued)

Bit 7: This bit is the complement of the Data Carrier Detect ($\overline{DCD}$) input. If bit 4 of the MCR is set to a 1, this bit is equivalent to OUT 2 in the MCR.

8.10 SCRATCHPAD REGISTER

This 8-bit Read/Write Register does not control the UART in anyway. It is intended as a scratchpad register to be used by the programmer to hold data temporarily.

8.11 FIFO INTERRUPT MODE OPERATION

When the RCVR FIFO and receiver interrupts are enabled (FCR0 = 1, IER0 = 1) RCVR interrupts will occur as follows:

A. The receive data available interrupt will be issued to the CPU when the FIFO has reached its programmed trigger level; it will be cleared as soon as the FIFO drops below its programmed trigger level.

B. The IIR receive data available indication also occurs when the FIFO trigger level is reached, and like the interrupt it is cleared when the FIFO drops below the trigger level.

C. The receiver line status interrupt (IIR = 06), as before, has higher priority than the received data available (IIR = 04) interrupt.

D. The data ready bit (LSR0) is set as soon as a character is transferred from the shift register to the RCVR FIFO. It is reset when the FIFO is empty.

When RCVR FIFO and receiver interrupts are enabled, RCVR FIFO timeout interrupts will occur as follows:

A. A FIFO timeout interrupt will occur, if the following conditions exist:

- — at least one character is in the FIFO
- — the most recent serial character received was longer than 4 continuous character times ago (if 2 stop bits are programmed the second one is included in this time delay).
- — the most recent CPU read of the FIFO was longer than 4 continuous character times ago.

The maximum time between a received character and a timeout interrupt will be 160 ms at 300 baud with a 12-bit receive character (i.e., 1 Start, 8 Data, 1 Parity and 2 Stop Bits).

B. Character times are calculated by using the RCLK input for a clock signal (this makes the delay proportional to the baudrate).

C. When a timeout interrupt has occurred it is cleared and the timer reset when the CPU reads one character from the RCVR FIFO.

D. When a timeout interrupt has not occurred the timeout timer is reset after a new character is received or after the CPU reads the RCVR FIFO.

When the XMIT FIFO and transmitter interrupts are enabled (FCR0 = 1, IER1 = 1), XMIT interrupts will occur as follows:

A. The transmitter holding register interrupt (02) occurs when the XMIT FIFO is empty; it is cleared as soon as the transmitter holding register is written to (1 to 16 characters may be written to the XMIT FIFO while servicing this interrupt) or the IIR is read.

B. The transmitter FIFO empty indications will be delayed 1 character time minus the last stop bit time whenever the following occurs: THRE = 1 and there have not been at least two bytes at the same time in the transmit FIFO, since the last THRE = 1. The first transmitter interrupt after changing FCR0 will be immediate, if it is enabled.

Character timeout and RCVR FIFO trigger level interrupts have the same priority as the current received data available interrupt; XMIT FIFO empty has the same priority as the current transmitter holding register empty interrupt.

8.12 FIFO POLLED MODE OPERATION

With FCR0 = 1 resetting IER0, IER1, IER2, IER3 or all to zero puts the UART in the FIFO Polled Mode of operation. Since the RCVR and XMITTER are controlled separately either one or both can be in the polled mode of operation.

In this mode the user's program will check RCVR and XMITTER status via the LSR. As stated previously:

LSR0 will be set as long as there is one byte in the RCVR FIFO.

LSR1 to LSR4 will specify which error(s) has occurred. Character error status is handled the same way as when in the interrupt mode, the IIR is not affected since IER2 = 0.

LSR5 will indicate when the XMIT FIFO is empty.

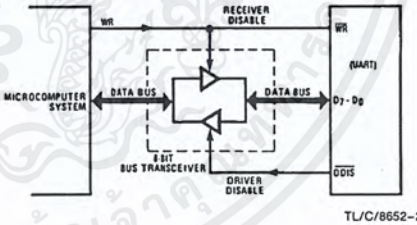
LSR6 will indicate that both the XMIT FIFO and shift register are empty.

LSR7 will indicate whether there are any errors in the RCVR FIFO.

There is no trigger level reached or timeout condition indicated in the FIFO Polled Mode, however, the RCVR and XMIT FIFOs are still fully capable of holding characters.

9.0 Typical Applications

Typical Interface for a High-Capacity Data Bus



TL/C/8652–23

9.0 Typical Applications (Continued)

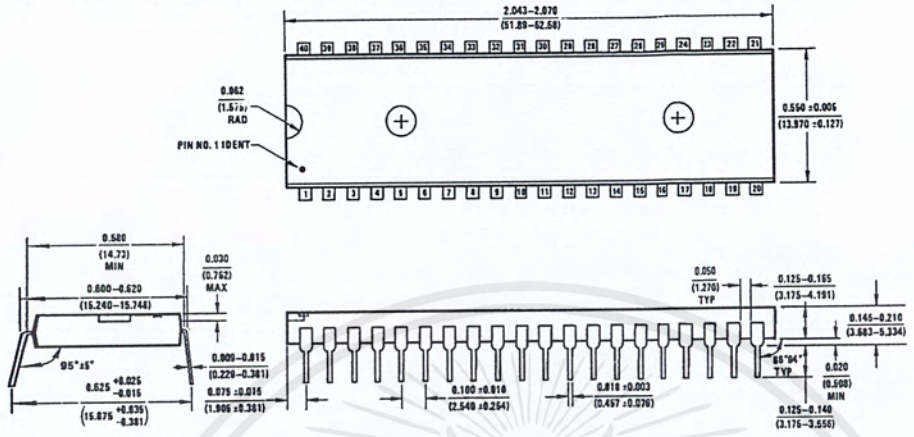


TL/C/8652–22

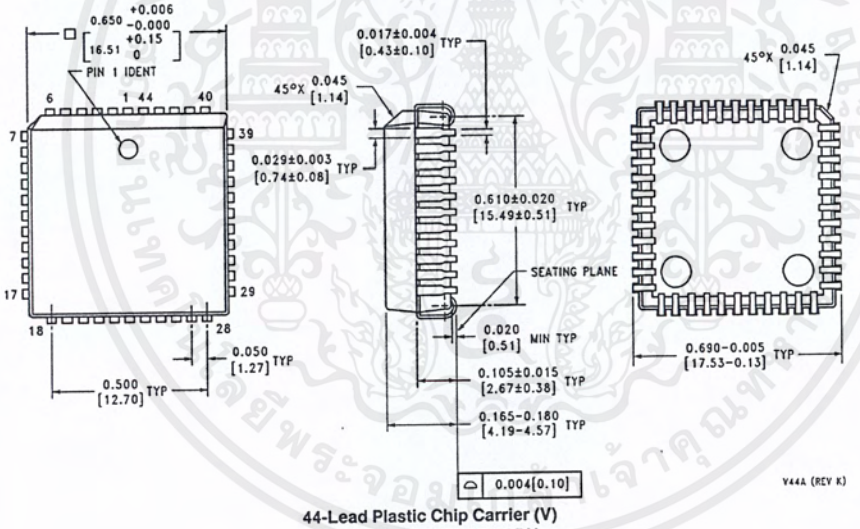
This shows the basic connections of an PC16550D to an 8088 CPU

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

10.0 Physical Dimensions inches (millimeters)

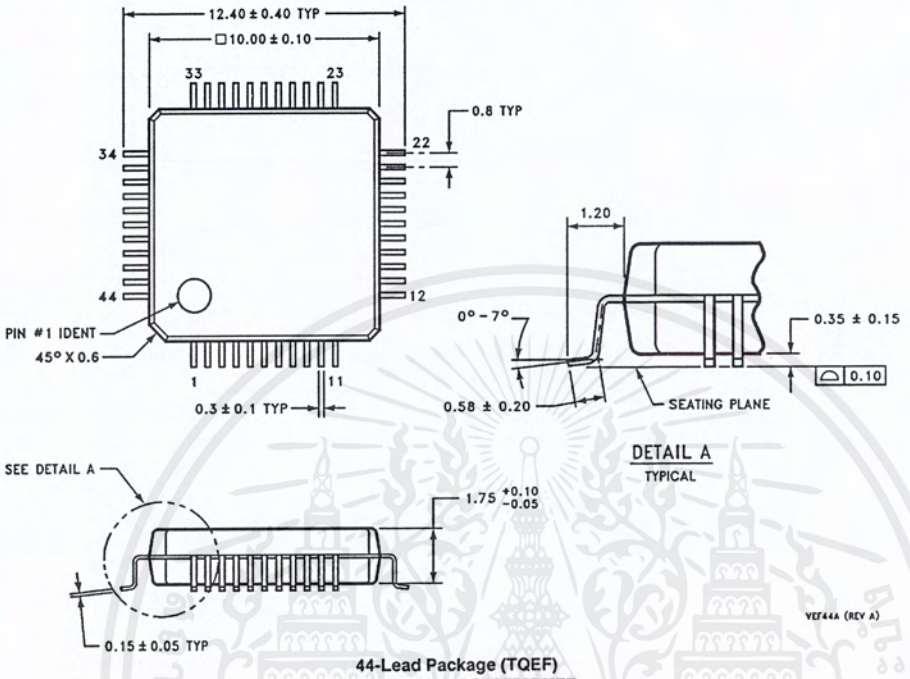


Plastic Dual-In-Line Package (N)
Order Number PC16550DN
NS Package Number N40A



44-Lead Plastic Chip Carrier (V)
Order Number PC16550DV
NS Package Number V44A

10.0 Physical Dimensions inches (millimeters) (Continued)



44-Lead Package (TQEF)
Order Number PC16550DVEF
NS Package Number VEF44A

LIFE SUPPORT POLICY

NATIONAL'S PRODUCTS ARE NOT AUTHORIZED FOR USE AS CRITICAL COMPONENTS IN LIFE SUPPORT DEVICES OR SYSTEMS WITHOUT THE EXPRESS WRITTEN APPROVAL OF THE PRESIDENT OF NATIONAL SEMICONDUCTOR CORPORATION. As used herein:

1. Life support devices or systems are devices or systems which, (a) are intended for surgical implant into the body, or (b) support or sustain life, and whose failure to perform, when properly used in accordance with instructions for use provided in the labeling, can be reasonably expected to result in a significant injury to the user.
2. A critical component is any component of a life support device or system whose failure to perform can be reasonably expected to cause the failure of the life support device or system, or to affect its safety or effectiveness.

National Semiconductor Corporation
2900 Semiconductor Drive
P.O. Box 58090
Santa Clara, CA 95052-8090
Tel: 1(800) 272-9959
TWX: (910) 339-9240

National Semiconductor GmbH
Livry-Gargan-Str. 10
D-82256 Fürstenfeldbruck
Germany
Tel: (81-41) 35-0
Telex: 527649
Fax: (81-41) 35-1

National Semiconductor Japan Ltd.
Sumitomo Chemical
Engineering Center
Bldg. 7F
1-7-1, Nakase, Mihama-Ku
Chiba-City,
Ciba Prefecture 261
Tel: (043) 299-2300
Fax: (043) 299-2500

National Semiconductor Hong Kong Ltd.
13th Floor, Straight Block,
Ocean Centre, 5 Canton Rd.
Tsimshatsui, Kowloon
Hong Kong
Tel: (852) 2737-1600
Fax: (852) 2736-9960

National Semiconductores Do Brazil Ltda.
Rue Deputado Lacorda Franco
120-3A
Sao Paulo-SP
Brazil 05418-000
Tel: (55-11) 212-5066
Telex: 391-1131931 NSBR BR
Fax: (55-11) 212-1181

National Semiconductor (Australia) Pty, Ltd.
Building 16
Business Park Drive
Monash Business Park
Nottinghill, Melbourne
Victoria 3168 Australia
Tel: (3) 558-9999
Fax: (3) 558-9998

National does not assume any responsibility for use of any circuitry described, no circuit patent licenses are implied and National reserves the right at any time without notice to change said circuitry and specifications.

E2O0017-27-X2

OKI Semiconductor

This version: Jan. 1998
Previous version: Aug. 1996

MSM82C51A-2RS/GS/JS

UNIVERSAL SYNCHRONOUS ASYNCHRONOUS RECEIVER TRANSMITTER

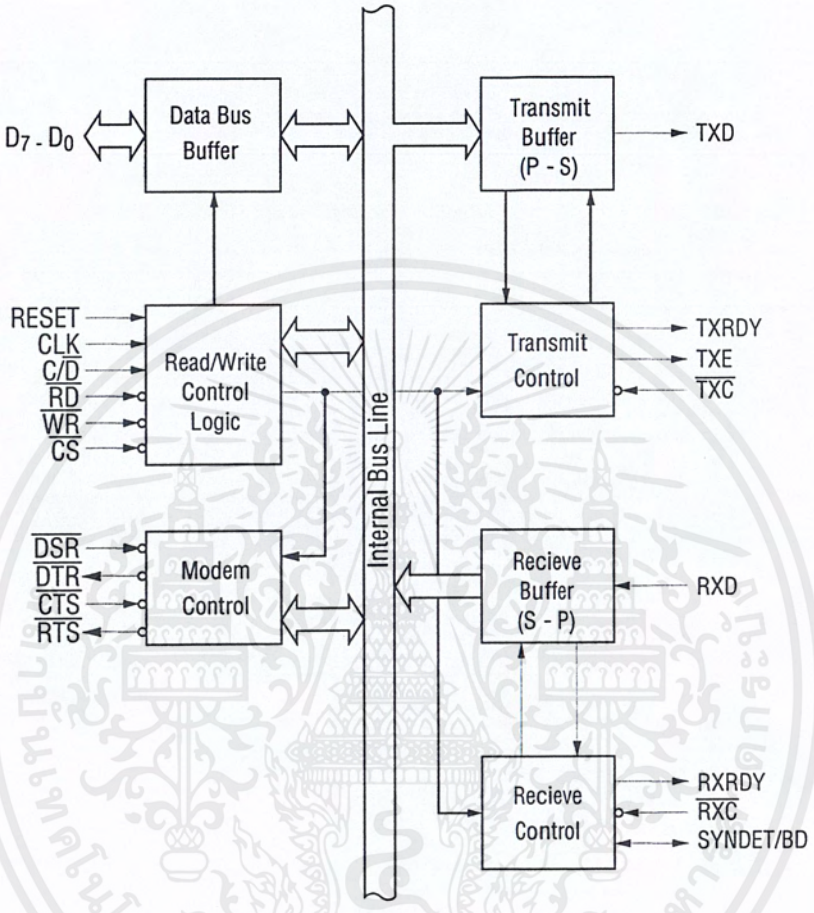
GENERAL DESCRIPTION

The MSM82C51A-2 is a USART (Universal Synchronous Asynchronous Receiver Transmitter) for serial data communication.
As a peripheral device of a microcomputer system, the MSM82C51A-2 receives parallel data from the CPU and transmits serial data after conversion. This device also receives serial data from the outside and transmits parallel data to the CPU after conversion.
The MSM82C51A-2 configures a fully static circuit using silicon gate CMOS technology. Therefore, it operates on extremely low power at 100 µA (max) of standby current by suspending all operations.

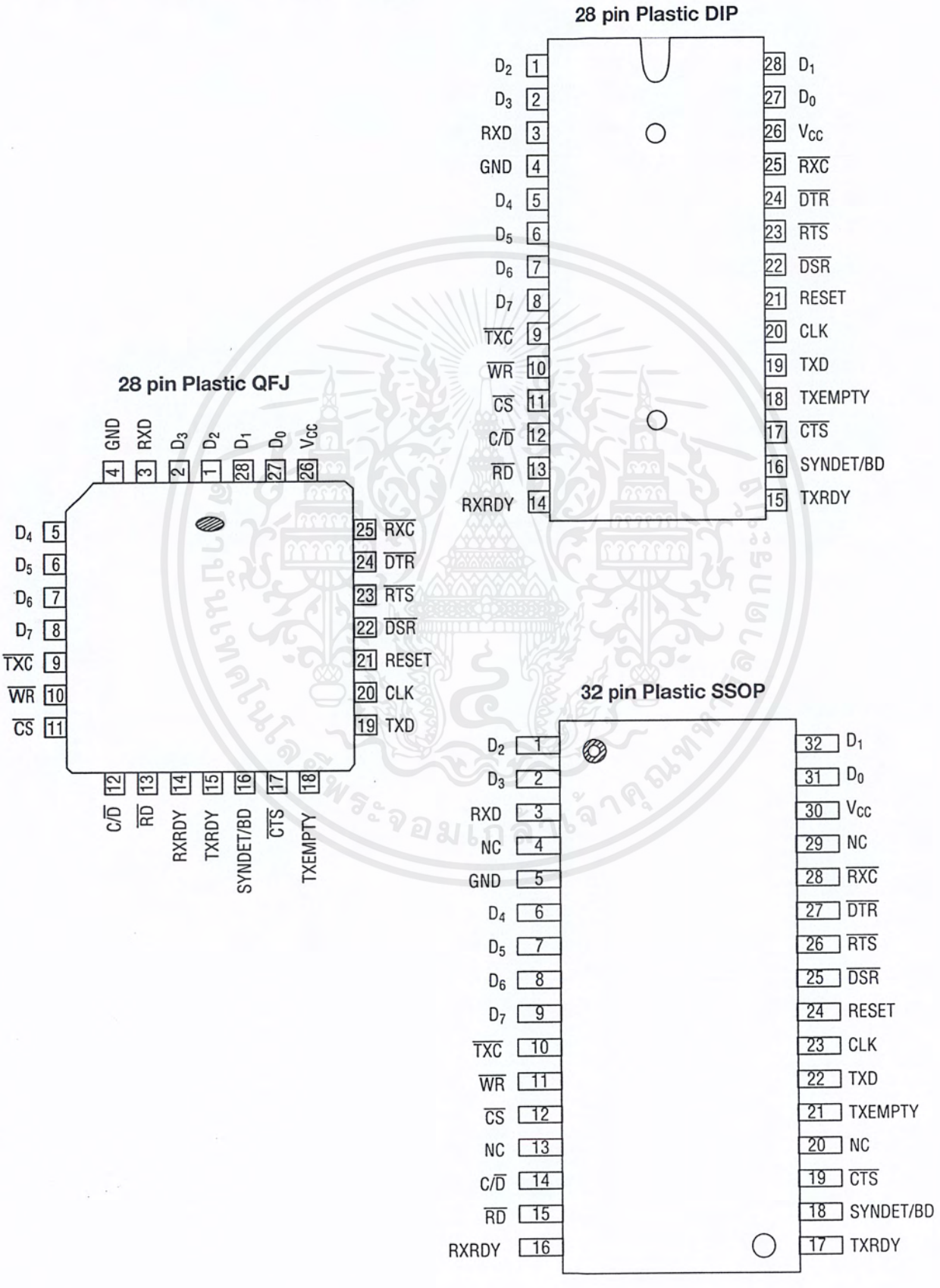
FEATURES

- Wide power supply voltage range from 3 V to 6 V
- Wide temperature range from –40°C to 85°C
- Synchronous communication upto 64 Kbaud
- Asynchronous communication upto 38.4 Kbaud
- Transmitting/receiving operations under double buffered configuration.
- Error detection (parity, overrun and framing)
- 28-pin Plastic DIP (DIP28-P-600-2.54): (Product name: MSM82C51A-2RS)
- 28-pin Plastic QFJ (QFJ28-P-S450-1.27): (Product name: MSM82C51A-2JS)
- 32-pin Plastic SSOP(SSOP32-P-430-1.00-K): (Product name: MSM82C51A-2GS-K)

FUNCTIONAL BLOCK DIAGRAM



PIN CONFIGURATION (TOP VIEW)



FUNCTION

Outline

The MSM82C51A-2's functional configuration is programed by software.
Operation between the MSM82C51A-2 and a CPU is executed by program control. Table 1 shows the operation between a CPU and the device.

Table 1 Operation between MSM82C51A and CPU

$\overline{CS}$	C/$\overline{D}$	$\overline{RD}$	$\overline{WR}$	
1	×	×	×	Data Bus 3-State
0	×	1	1	Data Bus 3-State
0	1	0	1	Status → CPU
0	1	1	0	Control Word ← CPU
0	0	0	1	Data → CPU
0	0	1	0	Data ← CPU

It is necessary to execute a function-setting sequence after resetting the MSM82C51A-2. Fig. 1 shows the function-setting sequence.
If the function was set, the device is ready to receive a command, thus enabling the transfer of data by setting a necessary command, reading a status and reading/writing data.

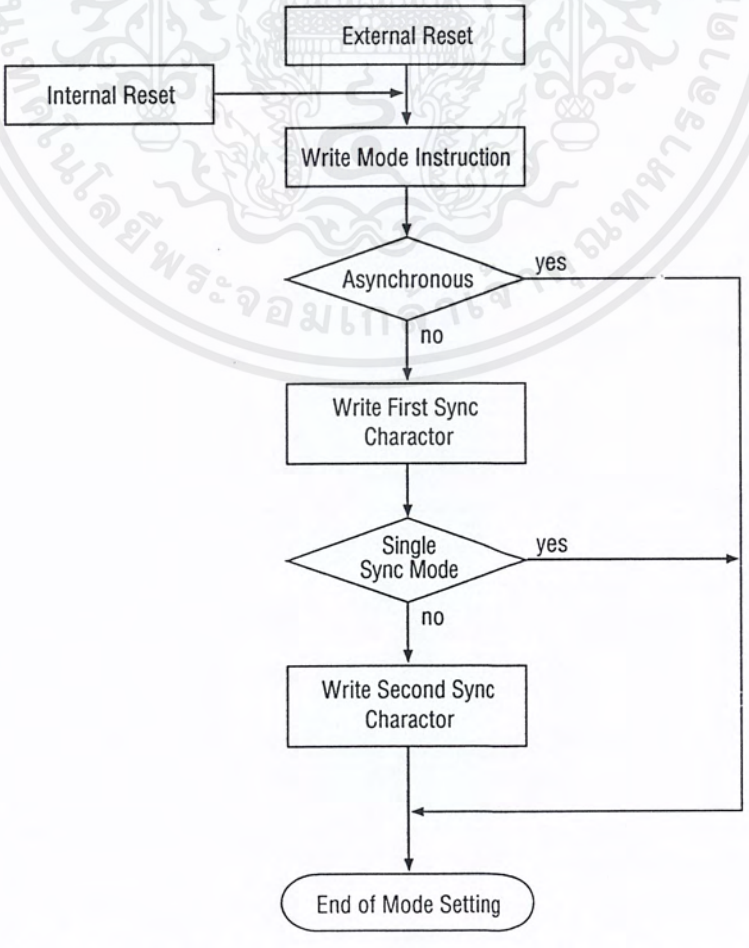


Fig. 1 Function-setting Sequence (Mode Instruction Sequence)

Control Words

There are two types of control word.
1. Mode instruction (setting of function)
2. Command (setting of operation)

1) Mode Instruction

Mode instruction is used for setting the function of the MSM82C51A-2. Mode instruction will be in "wait for write" at either internal reset or external reset. That is, the writing of a control word after resetting will be recognized as a "mode instruction."
Items set by mode instruction are as follows:

- Synchronous/asynchronous mode
- Stop bit length (asynchronous mode)
- Character length
- Parity bit
- Baud rate factor (asynchronous mode)
- Internal/external synchronization (synchronous mode)
- Number of synchronous characters (Synchronous mode)

The bit configuration of mode instruction is shown in Figures 2 and 3. In the case of synchronous mode, it is necessary to write one-or two byte sync characters.
If sync characters were written, a function will be set because the writing of sync characters constitutes part of mode instruction.

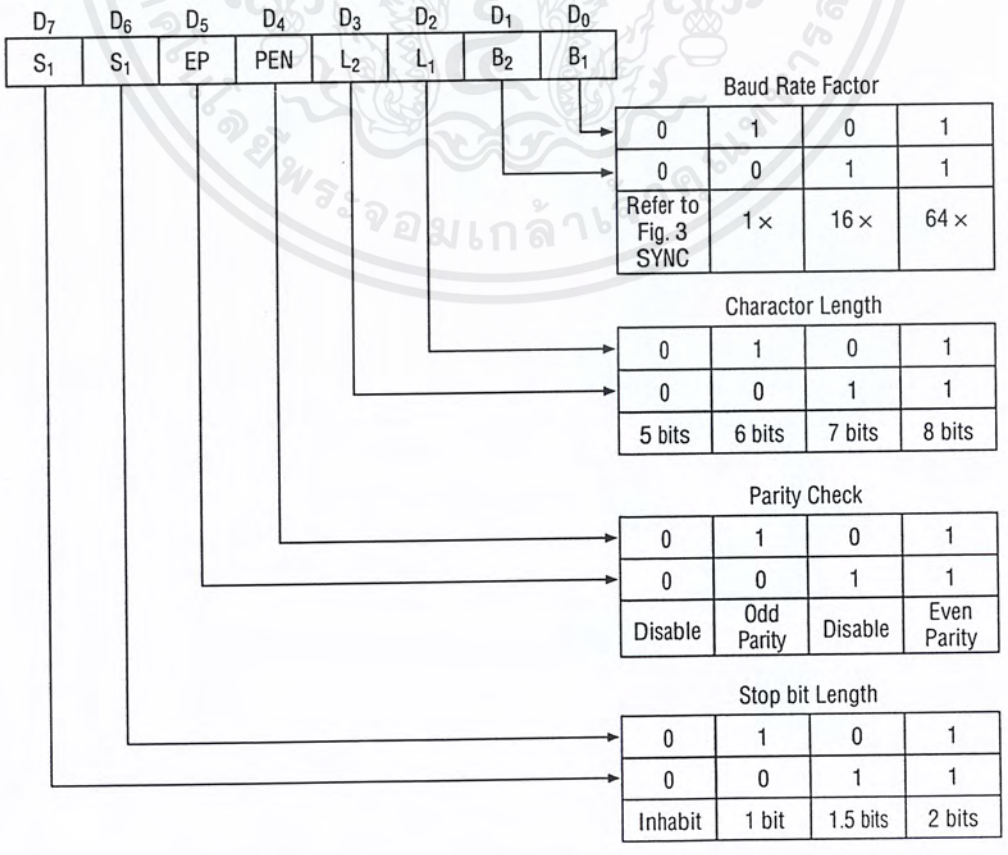


Fig. 2 Bit Configuration of Mode Instruction (Asynchronous)

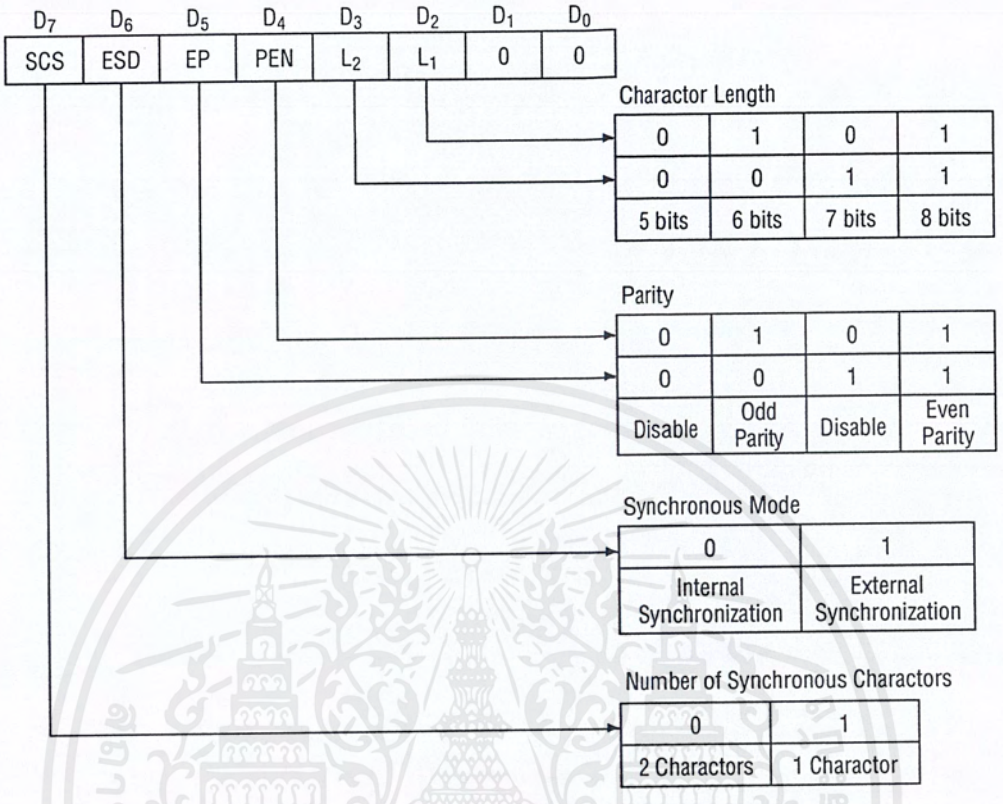


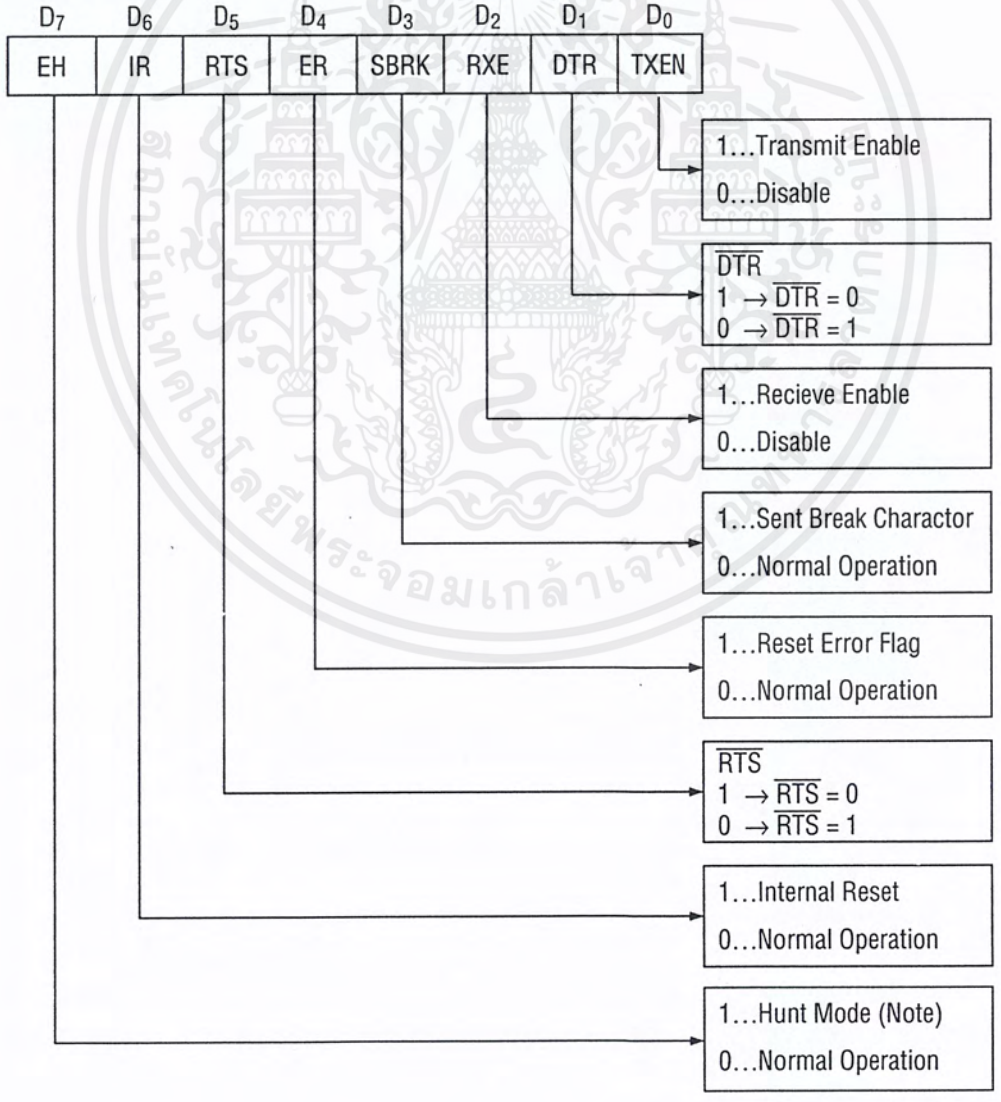
Fig. 3 Bit Configuration of Mode Instruction (Synchronous)

2) Command

Command is used for setting the operation of the MSM82C51A-2.
It is possible to write a command whenever necessary after writing a mode instruction and sync characters.
Items to be set by command are as follows:

- Transmit Enable/Disable
- Receive Enable/Disable
- $\overline{DTR}$, $\overline{RTS}$ Output of data.
- Resetting of error flag.
- Sending to break characters
- Internal resetting
- Hunt mode (synchronous mode)

The bit configuration of a command is shown in Fig. 4.



Note: Seach mode for synchronous charactors in synchronous mode.

Fig. 4 Bit Configuration of Command

Status Word

It is possible to see the internal status of MSM82C51A-2 by reading a status word.
The bit configuration of status word is shown in Fig. 5.

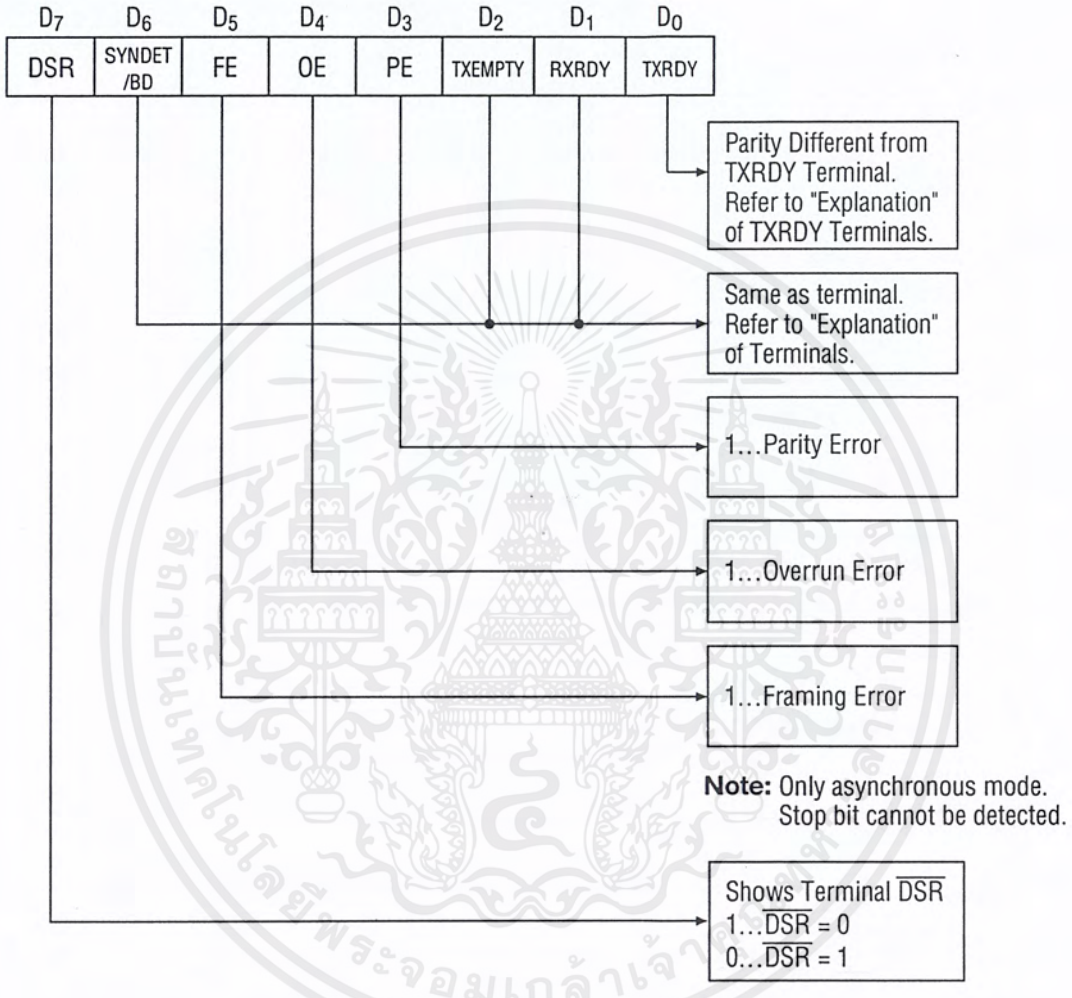


Fig. 5 Bit Configuration of Status Word

Standby Status

It is possible to put the MSM82C51A-2 in "standby status"
When the following conditions have been satisfied the MSM82C51A-2 is in "standby status."

(1) $\overline{CS}$ terminal is fixed at Vcc level.
(2) Input pins other $\overline{CS}$, D_0 to D_7, $\overline{RD}$, $\overline{WR}$ and C/$\overline{D}$ are fixed at Vcc or GND level (including SYNDET in external synchronous mode).

Note: When all output currents are 0, ICCS specification is applied.

Pin Description

D_0 to D_7 (I/O terminal)
This is bidirectional data bus which receive control words and transmits data from the CPU and sends status words and received data to CPU.

RESET (Input terminal)
A "High" on this input forces the MSM82C51A-2 into "reset status."
The device waits for the writing of "mode instruction."
The min. reset width is six clock inputs during the operating status of CLK.

CLK (Input terminal)
CLK signal is used to generate internal device timing.
CLK signal is independent of $\overline{RXC}$ or $\overline{TXC}$.
However, the frequency of CLK must be greater than 30 times the $\overline{RXC}$ and $\overline{TXC}$ at Synchronous mode and Asynchronous "x1" mode, and must be greater than 5 times at Asynchronous "x16" and "x64" mode.

$\overline{WR}$ (Input terminal)
This is the "active low" input terminal which receives a signal for writing transmit data and control words from the CPU into the MSM82C51A-2.

$\overline{RD}$ (Input terminal)
This is the "active low" input terminal which receives a signal for reading receive data and status words from the MSM82C51A-2.

C/$\overline{D}$ (Input terminal)
This is an input terminal which receives a signal for selecting data or command words and status words when the MSM82C51A-2 is accessed by the CPU.
If C/$\overline{D}$ = low, data will be accessed.
If C/$\overline{D}$ = high, command word or status word will be accessed.

$\overline{CS}$ (Input terminal)
This is the "active low" input terminal which selects the MSM82C51A-2 at low level when the CPU accesses.

Note: The device won't be in "standby status"; only setting $\overline{CS}$ = High.
Refer to "Explanation of Standby Status."

TXD (output terminal)
This is an output terminal for transmitting data from which serial-converted data is sent out. The device is in "mark status" (high level) after resetting or during a status when transmit is disabled. It is also possible to set the device in "break status" (low level) by a command.

TXRDY (output terminal)
This is an output terminal which indicates that the MSM82C51A-2 is ready to accept a transmitted data character. But the terminal is always at low level if $\overline{CTS}$ = high or the device was set in "TX disable status" by a command.

Note: TXRDY status word indicates that transmit data character is receivable, regardless of $\overline{CTS}$ or command.

If the CPU writes a data character, TXRDY will be reset by the leading edge or $\overline{WR}$ signal.

TXEMPTY (Output terminal)
This is an output terminal which indicates that the MSM82C51A-2 has transmitted all the characters and had no data character.
In "synchronous mode," the terminal is at high level, if transmit data characters are no longer remaining and sync characters are automatically transmitted. If the CPU writes a data character, TXEMPTY will be reset by the leading edge of $\overline{WR}$ signal.

Note : As the transmitter is disabled by setting $\overline{CTS}$ "High" or command, data written before disable will be sent out. Then TXD and TXEMPTY will be "High".

Even if a data is written after disable, that data is not sent out and TXE will be "High".After the transmitter is enabled, it sent out. (Refer to Timing Chart of Transmitter Control and Flag Timing)

$\overline{TXC}$ (Input terminal)
This is a clock input signal which determines the transfer speed of transmitted data.
In "synchronous mode," the baud rate will be the same as the frequency of $\overline{TXC}$.
In "asynchronous mode", it is possible to select the baud rate factor by mode instruction.
It can be 1, 1/16 or 1/64 the $\overline{TXC}$.
The falling edge of $\overline{TXC}$ sifts the serial data out of the MSM82C51A-2.

RXD (input terminal)
This is a terminal which receives serial data.

RXRDY (Output terminal)
This is a terminal which indicates that the MSM82C51A-2 contains a character that is ready to READ.
If the CPU reads a data character, RXRDY will be reset by the leading edge of $\overline{RD}$ signal.
Unless the CPU reads a data character before the next one is received completely, the preceding data will be lost. In such a case, an overrun error flag status word will be set.

$\overline{RXC}$ (Input terminal)
This is a clock input signal which determines the transfer speed of received data.
In "synchronous mode," the baud rate is the same as the frequency of $\overline{RXC}$.
In "asynchronous mode," it is possible to select the baud rate factor by mode instruction.
It can be 1, 1/16, 1/64 the $\overline{RXC}$.

SYNDET/BD (Input or output terminal)
This is a terminal whose function changes according to mode.
In "internal synchronous mode." this terminal is at high level, if sync characters are received and synchronized. If a status word is read, the terminal will be reset.
In "external synchronous mode, "this is an input terminal.
A "High" on this input forces the MSM82C51A-2 to start receiving data characters.
In "asynchronous mode," this is an output terminal which generates "high level" output upon the detection of a "break" character if receiver data contains a "low-level" space between the stop bits of two continuous characters. The terminal will be reset, if RXD is at high level.
After Reset is active, the terminal will be output at low level.

$\overline{\text{DSR}}$ (Input terminal)
This is an input port for MODEM interface. The input status of the terminal can be recognized by the CPU reading status words.

$\overline{\text{DTR}}$ (Output terminal)
This is an output port for MODEM interface. It is possible to set the status of $\overline{\text{DTR}}$ by a command.

$\overline{\text{CTS}}$ (Input terminal)
This is an input terminal for MODEM interface which is used for controlling a transmit circuit. The terminal controls data transmission if the device is set in "TX Enable" status by a command. Data is transmitable if the terminal is at low level.

$\overline{\text{RTS}}$ (Output terminal)
This is an output port for MODEM interface. It is possible to set the status $\overline{\text{RTS}}$ by a command.

ABSOLUTE MAXIMUM RATING

Parameter	Symbol	Rating			Unit	Conditions
		MSM82C51A-2RS	MSM82C51A-2GS	MSM82C51A-2JS		
Power Supply Voltage	V_{CC}	−0.5 to +7			V	With respect to GND
Input Voltage	V_{IN}	−0.5 to V_{CC} +0.5			V	
Output Voltage	V_{OUT}	−0.5 to V_{CC} +0.5			V	
Storage Temperature	T_{STG}	−55 to +150			°C	—
Power Dissipation	P_D	0.9	0.7	0.9	W	Ta = 25°C

OPERATING RANGE

Parameter	Symbol	Range	Unit
Power Supply Voltage	V_{CC}	3 - 6	V
Operating Temperature	T_{op}	−40 to 85	°C

RECOMMENDED OPERATING CONDITIONS

Parameter	Symbol	Min.	Typ.	Max.	Unit
Power Supply Voltage	V_{CC}	4.5	5	5.5	V
Operating Temperature	T_{op}	−40	+25	+85	°C
"L" Input Voltage	V_{IL}	−0.3	—	+0.8	V
"H" Input Voltage	V_{IH}	2.2	—	V_{CC} +0.3	V

DC CHARACTERISTICS

(V_{CC} = 4.5 to 5.5 V Ta = −40°C to +85°C)

Parameter	Symbol	Min.	Typ.	Max.	Unit	Measurement Conditions
"L" Output Voltage	V_{OL}	—	—	0.45	V	I_{OL} = 2.5 mA
"H" Output Voltage	V_{OH}	3.7	—	—	V	I_{OH} = −2.5 mA
Input Leak Current	I_{LI}	−10	—	10	µA	$0 \le V_{IN} \le V_{CC}$
Output Leak Current	I_{LO}	−10	—	10	µA	$0 \le V_{OUT} \le V_{CC}$
Operating Supply Current	I_{CCO}	—	—	5	mA	Asynchronous X64 during Transmitting/ Receiving
Standby Supply Current	I_{CCS}	—	—	100	µA	All Input voltage shall be fixed at V_{CC} or GND level.

AC CHARACTERISTICS

CPU Bus Interface Part

(V_{CC} = 4.5 to 5.5 V, Ta = –40 to 85°C)

Parameter	Symbol	Min.	Max.	Unit	Remarks
Address Stable before $\overline{RD}$	t_{AR}	20	—	ns	Note 2
Address Hold Time for $\overline{RD}$	t_{RA}	20	—	ns	Note 2
$\overline{RD}$ Pulse Width	t_{RR}	130	—	ns	—
Data Delay from $\overline{RD}$	t_{RD}	—	100	ns	—
$\overline{RD}$ to Data Float	t_{DF}	10	75	ns	—
Recovery Time between $\overline{RD}$	t_{RVR}	6	—	t_{CY}	Note 5
Address Stable before $\overline{WR}$	t_{AW}	20	—	ns	Note 2
Address Hold Time for $\overline{WR}$	t_{WA}	20	—	ns	Note 2
$\overline{WR}$ Pulse Width	t_{WW}	100	—	ns	—
Data Set-up Time for $\overline{WR}$	t_{DW}	100	—	ns	—
Data Hold Time for $\overline{WR}$	t_{WD}	0	—	ns	—
Recovery Time between $\overline{WR}$	t_{RVW}	6	—	t_{CY}	Note 4
RESET Pulse Width	t_{RESW}	6	—	t_{CY}	—

Serial Interface Part

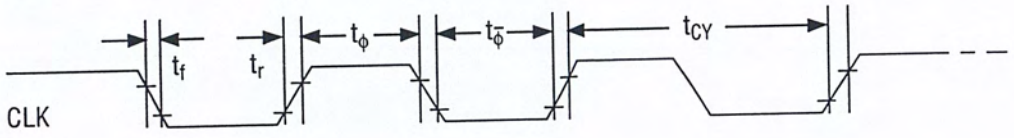
(V_{CC} = 4.5 to 5.5 V, Ta = –40 to 85°C)

Parameter		Symbol	Min.	Max.	Unit	Remarks
Main Clock Period		t_{CY}	160	—	ns	Note 3
Clock Low Tme		t_{ϕ}	50	—	ns	—
Clock High Time		t_{ϕ}	70	t_{CY} –50	ns	—
Clock Rise/Fall Time		t_r, t_f	—	20	ns	—
TXD Delay from Falling Edge of $\overline{TXC}$		t_{DTX}	—	1	µS	—
Transmitter Clock Frequency	1 × Baud	f_{TX}	DC	64	kHz	Note 3
	16 × Baud	f_{TX}	DC	615	kHz	
	64 × Baud	f_{TX}	DC	615	kHz	
Transmitter Clock Low Time	1 × Baud	t_{TPW}	13	—	t_{CY}	—
	16 ×, 64 × Baud	t_{TPW}	2	—	t_{CY}	—
Transmitter Clock High Time	1 × Baud	t_{TPD}	15	—	t_{CY}	—
	16 ×, 64 × Baud	t_{TPD}	3	—	t_{CY}	—
Receiver Clock Frequency	1 × Baud	f_{RX}	DC	64	kHz	Note 3
	16 × Baud	f_{RX}	DC	615	kHz	
	64 × Baud	f_{RX}	DC	615	kHz	
Receiver Clock Low Time	1 × Baud	t_{RPW}	13	—	t_{CY}	—
	16 ×, 64 × Baud	t_{RPW}	2	—	t_{CY}	—
Receiver Clock High Time	1 × Baud	t_{RPD}	15	—	t_{CY}	—
	16 ×, 64 × Baud	t_{RPD}	3	—	t_{CY}	—
Time from the Center of Last Bit to the Rise of TXRDY		t_{TXRDY}	—	8	t_{CY}	—
Time from the Leading Edge of $\overline{WR}$ to the Fall of TXRDY		$t_{TXRDY\ CLEAR}$	—	400	ns	—
Time From the Center of Last Bit to the Rise of RXRDY		t_{RXRDY}	—	26	t_{CY}	—
Time from the Leading Edge of $\overline{RD}$ to the Fall of RXRDY		$t_{RXRDY\ CLEAR}$	—	400	ns	—
Internal SYNDET Delay Time from Rising Edge of $\overline{RXC}$		t_{IS}	—	26	t_{CY}	—
SYNDET Setup Time for $\overline{RXC}$		t_{ES}	18	—	t_{CY}	—
TXE Delay Time from the Center of Last Bit		$t_{TXEMPTY}$	20	—	t_{CY}	—
MODEM Control Signal Delay Time from Rising Edge of $\overline{WR}$		t_{WC}	8	—	t_{CY}	—
MODEM Control Signal Setup Time for Falling Edge of $\overline{RD}$		t_{CR}	20	—	t_{CY}	—
RXD Setup Time for Rising Edge of $\overline{RXC}$ (1X Baud)		t_{RXDS}	11	—	t_{CY}	—
RXD Hold Time for Falling Edge of $\overline{RXC}$ (1X Baud)		t_{RXDH}	17	—	t_{CY}	—

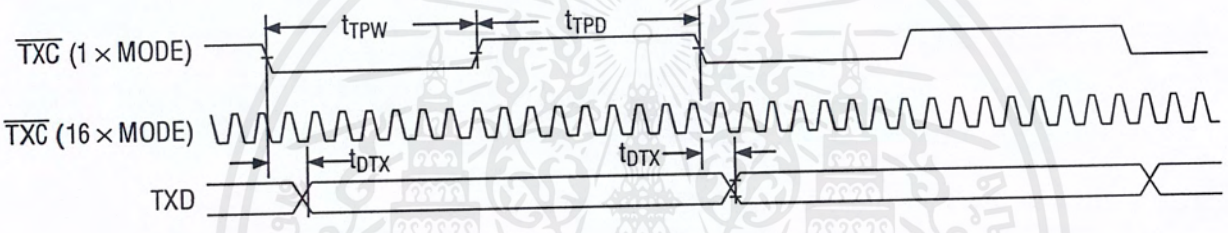
Notes:
1. AC characteristics are measured at 150 pF capacity load as an output load based on 0.8 V at low level and 2.2 V at high level for output and 1.5 V for input.
2. Addresses are $\overline{CS}$ and C/$\overline{D}$.
3. f_{TX} or $f_{RX} \leq 1/(30\ Tcy)$ 1× Baud
   f_{TX} or $f_{RX} \leq 1/(5\ Tcy)$ 16×, 64× Baud
4. This recovery time is mode Initialization only. Recovery time between command writes for Asynchronous Mode is 8 t_{CY} and for Synchronous Mode is 18 t_{CY}.
   Write Data is allowed only when TXRDY = 1.
5. This recovery time is Status read only.
   Read Data is allowed only when RXRDY = 1.
6. Status update can have a maximum delay of 28 clock periods from event affecting the status.

TIMING CHART

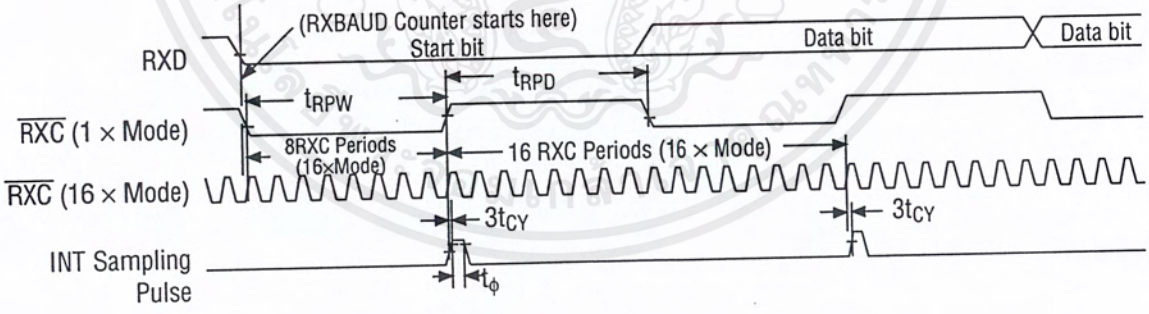
Sytem Clock Input



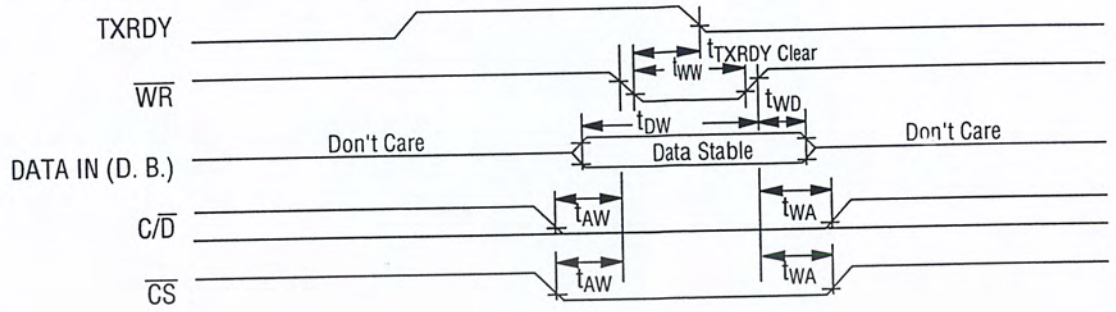
Transmitter Clock and Data



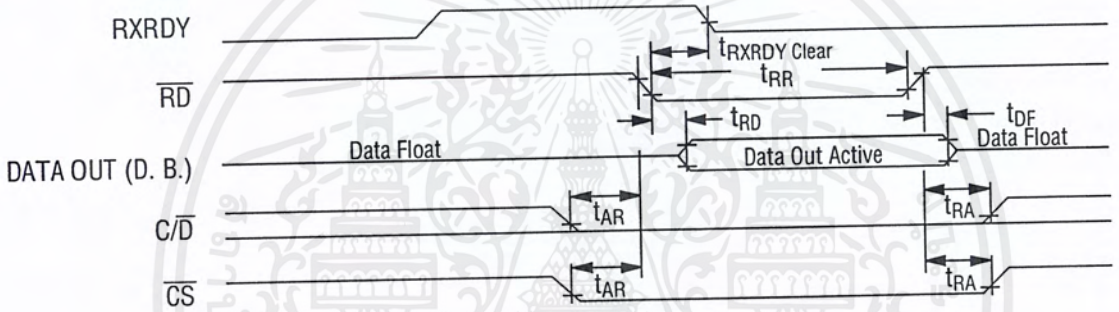
Receiver Clock and Data



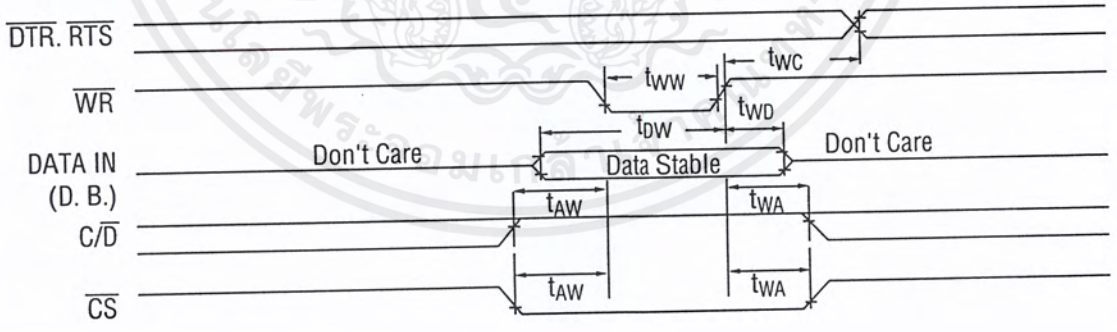
Write Data Cycle (CPU → USART)



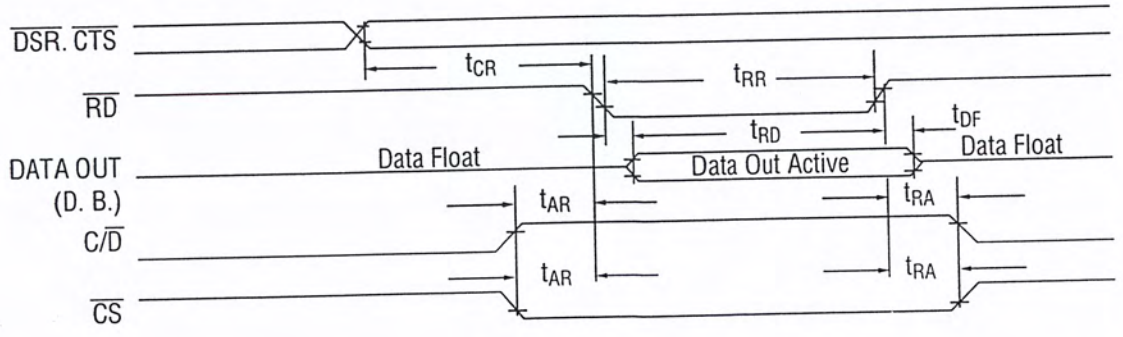
Read Data Cycle (CPU ← USART)



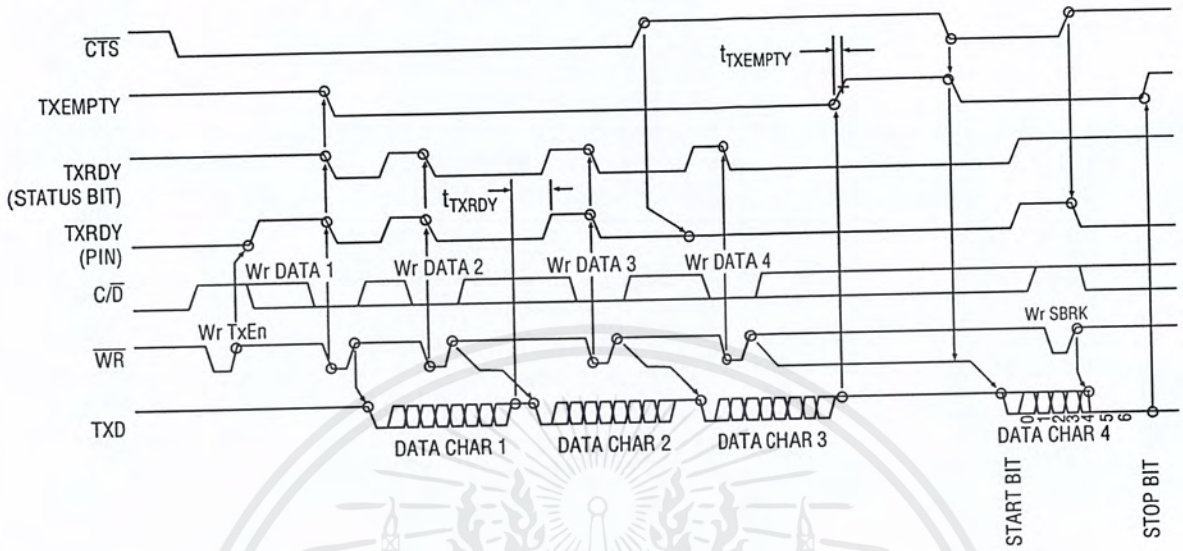
Write Control or Output Port Cycle (CPU → USART)



Read Control or Input Port Cycle (CPU ← USART)

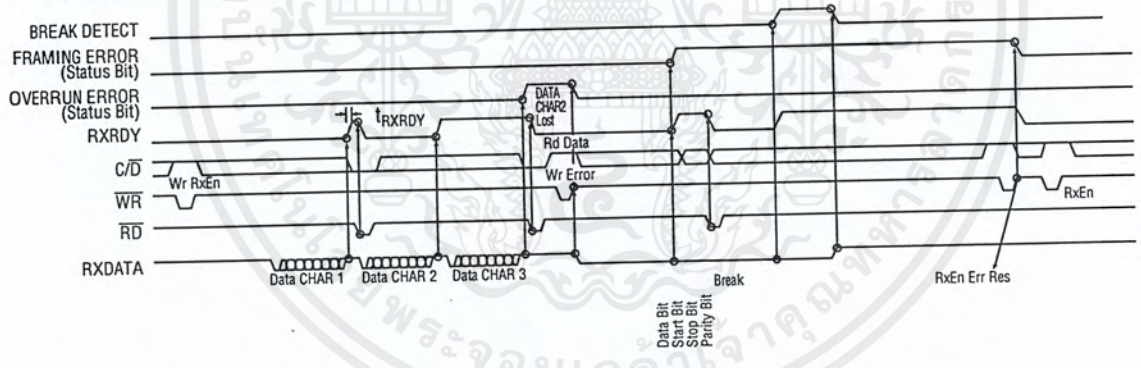


Transmitter Control and Flag Timing (ASYNC Mode)



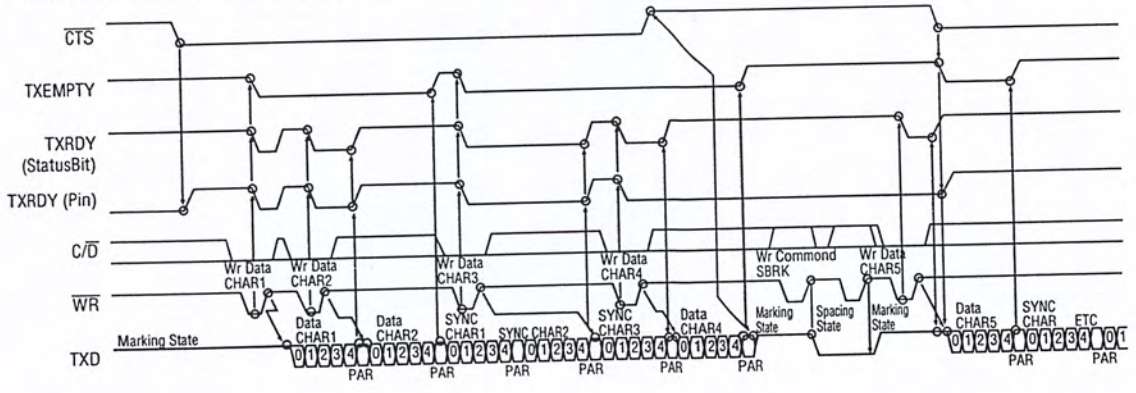
Note: The wave-form chart is based on the case of 7-bit data length + parity bit + 2 stop bit.

Receiver Control and Flag Timing (ASYNC Mode)



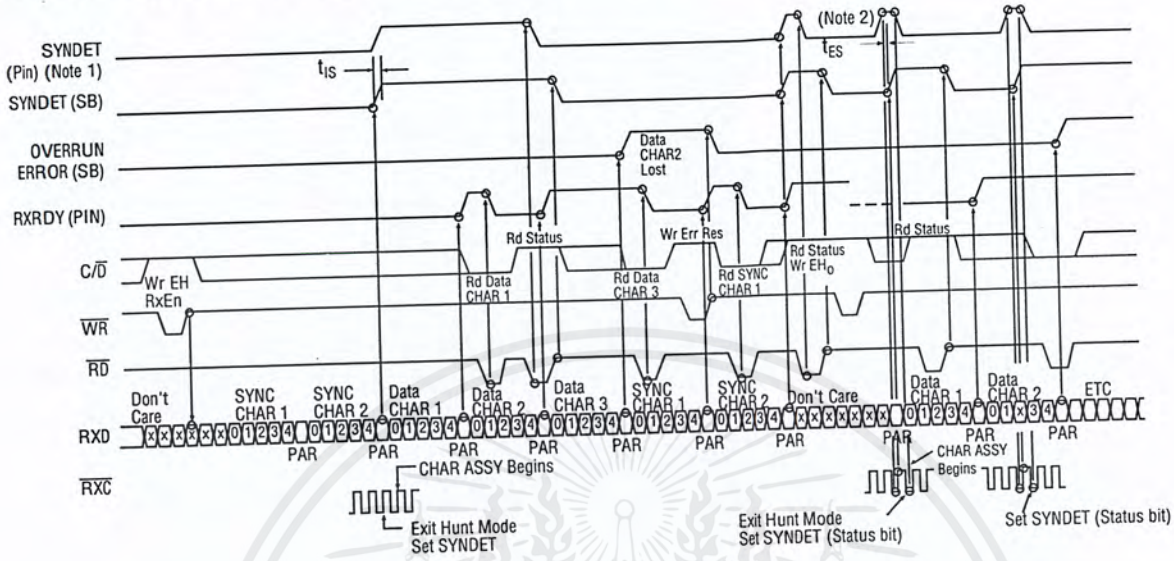
Note: The wave-form chart is based on the case of 7 data bit length + parity bit + 2 stop bit.

Transmitter Control and Flag Timing (SYNC Mode)



Note: The wave-form chart is based on the case of 5 data bit length + parity bit and 2 synchronous charactors.

Receiver Control and Flag Timing (SYNC Mode)



Note: 1. Internal Synchronization is based on the case of 5 data bit length + parity bit and 2 synchronous charactor.
2. External Synchronization is based on the case of 5 data bit length + parity bit.

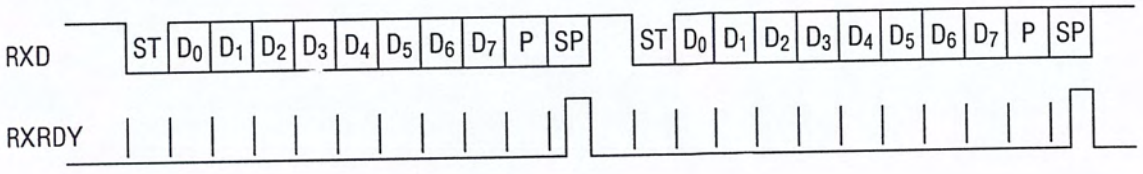
Note: 1. Half-bit processing for the start bit
When the MSM82C51A-2 is used in the asynchronous mode, some problems are caused in the processing for the start bit whose length is smaller than the 1-data bit length. (See Fig. 1.)

Start bit Length	Mode	Operation
Smaller than 7-Receiver Clock Length	×16	The short start bit is ignored. (Normal)
Smaller than 31-Receiver Clock Length	×64	The short start bit is ignored. (Normal)
8-Receiver Clock Length	×16	Data cannot be received correctly due to a malfunction.
32-Receiver Clock Length	×64	Data cannot be received correctly due to a malfunction.
9 to 16-Receiver Clock Length	×16	The bit is regarded as a start bit. (normal)
33 to 64-Receiver Clock Length	×64	The bit is regarded as a start bit. (normal)

2. Parity flag after a break signal is received (See Fig. 2.)
When the MSM82C51A-2 is used in the asynchrous mode, a parity flag may be set when the next normal data is read after a break signal is received.
A parity flag is set when the rising edge of the break signal (end of the break signal) is changed between the final data bit and the parity bit, through a RXRDY signal may not be outputted.
If this occurs, the parity flag is left set when the next normal dats is received, and the received data seems to be a parity error.

Half-bit Processing Timing Chart for the Start bit (Fig. 1)

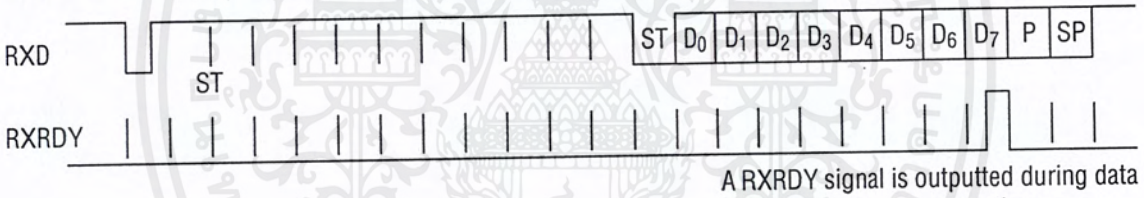
Normal Operation



The Start bit Is Shorter Than a 1/2 Data bit

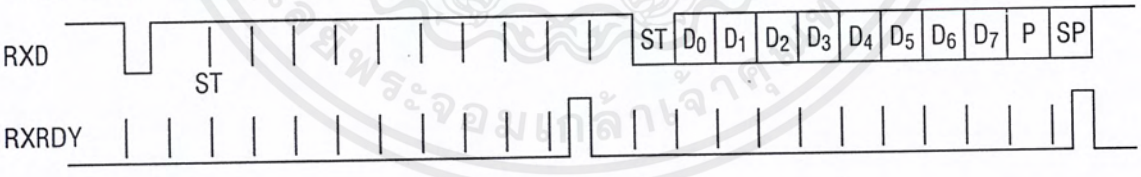


The Start bit Is a 1/2 Data bit (A problem of MSM82C51A-2)



A RXRDY signal is outputted during data reception due to a malfunction.

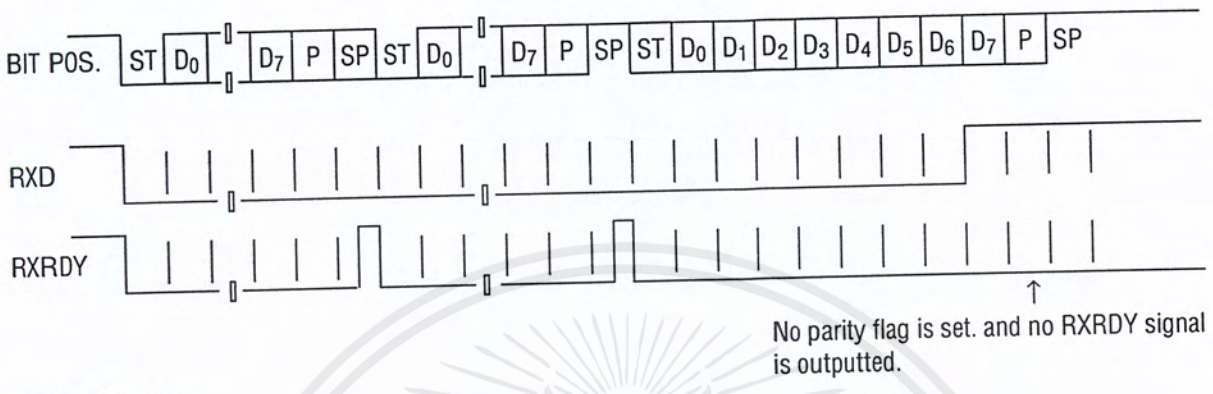
The Start bit Is Longer Than a 1/2 Data bit



ST: Start bit
SP: Stop bit
P: Parity bit
D_0 - D_7: Data bits

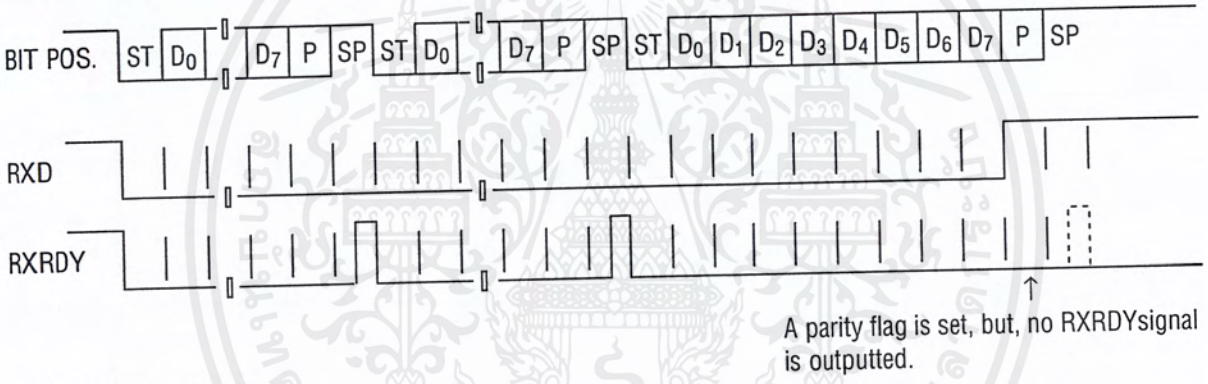
Break Signal Reception Timing and Parity Flag (Fig. 2)

Normal Operation



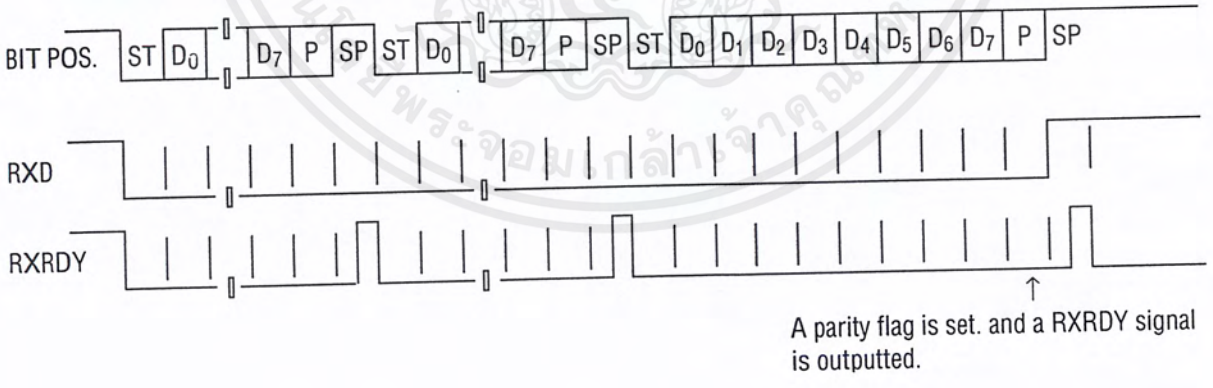
No parity flag is set. and no RXRDY signal is outputted.

Bug Timing



A parity flag is set, but, no RXRDYsignal is outputted.

Normal Operation



A parity flag is set. and a RXRDY signal is outputted.

NOTICE ON REPLACING LOW-SPEED DEVICES WITH HIGH-SPEED DEVICES

The conventional low speed devices are replaced by high-speed devices as shown below. When you want to replace your low speed devices with high-speed devices, read the replacement notice given on the next pages.

High-speed device (New)	Low-speed device (Old)	Remarks
M80C85AH	M80C85A/M80C85A-2	8bit MPU
M80C86A-10	M80C86A/M80C86A-2	16bit MPU
M80C88A-10	M80C88A/M80C88A-2	8bit MPU
M82C84A-2	M82C84A/M82C84A-5	Clock generator
M81C55-5	M81C55	RAM.I/O, timer
M82C37B-5	M82C37A/M82C37A-5	DMA controller
M82C51A-2	M82C51A	USART
M82C53-2	M82C53-5	Timer
M82C55A-2	M82C55A-5	PPI

Differences between MSM82C51A and MSM82C51A-2

1) Manufacturing Process

These devices use a 3 μ Si-Gate CMOS process technology and have the same chip size.

2) Function

These devices have the same logics except for changes in AC characteristics listed in (3-2).

3) Electrical Characteristics

3-1) DC Characteristics

Parameter	Symbol	MSM82C51A	MSM82C51A-2
V_{OL} measurement conditions	I_{OL}	+2.0 mA	+2.5 mA
V_{OH} measurement conditions	I_{OH}	-400 μA	-2.5 mA

Although the output voltage characteristics of these devices are identical, but the measurement conditions of the MSM82C51A-2 are more restricted than the MSM82C51A.

3-2) AC Characteristics

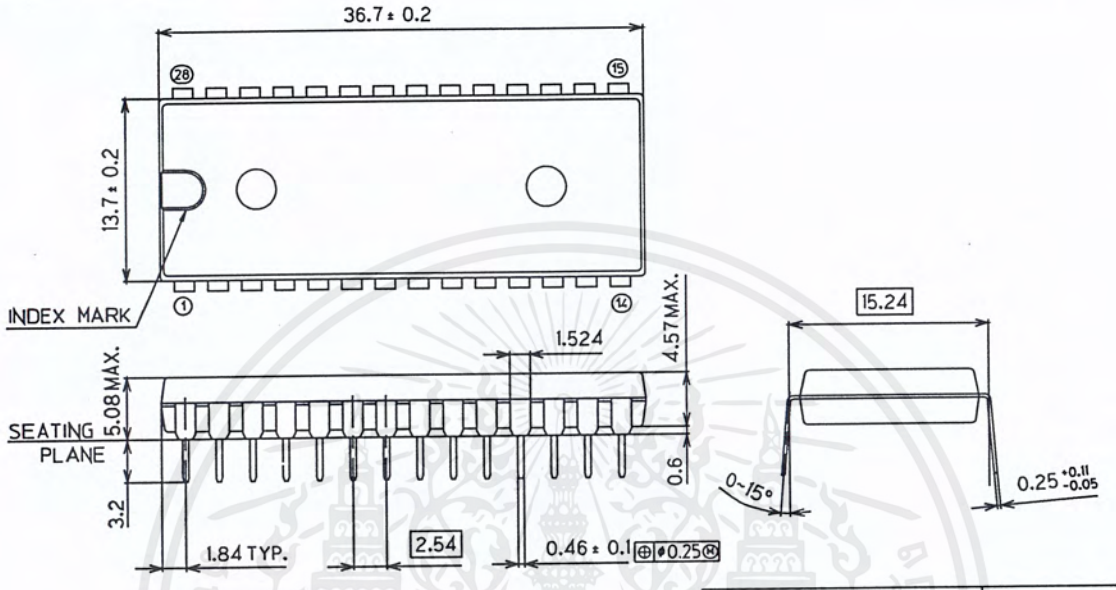
Parameter	Symbol	MSM82C51A	MSM82C51A-2
$\overline{RD}$ Pulse Width	t_{RR}	250 ns minimum	130 ns minimum
$\overline{RD}$ Rising to Data Difinition	t_{RD}	200 ns maximum	100 ns maximum
$\overline{RD}$ Rising to Data Float	t_{RF}	100 ns maximum	75 ns minimum
$\overline{WR}$ Pulse Width	t_{WW}	250 ns minimum	100 ns minimum
Data Setup Time for $\overline{WR}$ Rising	t_{DW}	150 ns minimum	100 ns minimum
Data Hold Time for $\overline{WR}$ Rising	t_{WD}	20 ns minimum	0 ns minimum
Master Clock Period	t_{CY}	250 ns minimum	160 ns minimum
Clock Low Time	$t_{\bar{\phi}}$	90 ns minimum	50 ns minimum
Clock High Time	t_{ϕ}	120 ns minimum t_{CY}-90 ns maximum	70 ns minimum t_{CY}-50 ns maximum

As shown above, the MSM82C51A-2 satisfies the characteristics of the MSM82C51A.

PACKAGE DIMENSIONS

(Unit : mm)

DIP28-P-600-2.54



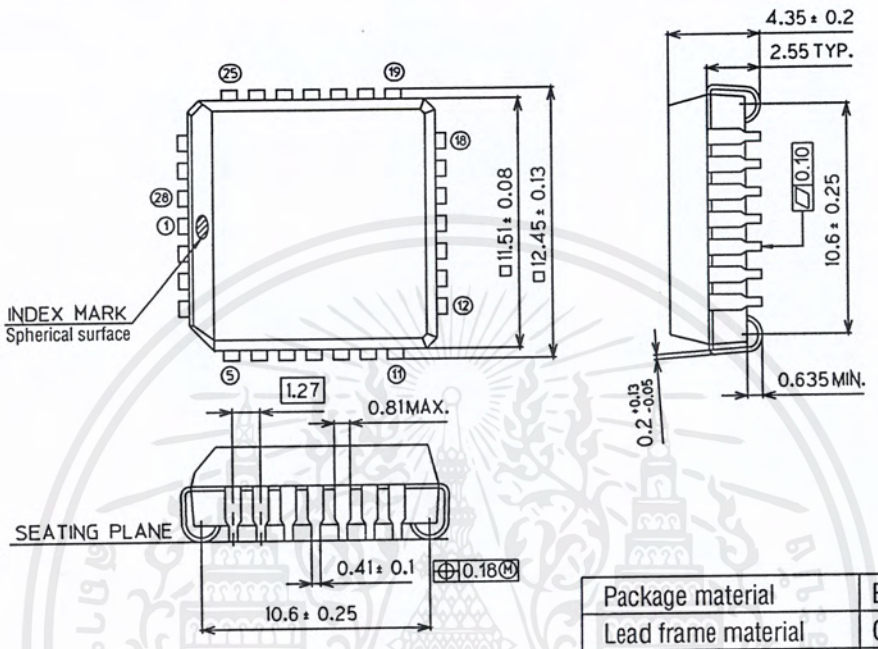
Package material	Epoxy resin
Lead frame material	42 alloy
Pin treatment	Solder plating
Solder plate thickness	5 μm or more
Package weight (g)	4.30 TYP.

Notes for Mounting the Surface Mount Type Package

The SOP, QFP, TSOP, SOJ, QFJ (PLCC), SHP and BGA are surface mount type packages, which are very susceptible to heat in reflow mounting and humidity absorbed in storage.
Therefore, before you perform reflow mounting, contact Oki's responsible sales person for the product name, package name, pin number, package code and desired mounting conditions (reflow method, temperature and times).

(Unit : mm)

QFJ28-P-S450-1.27



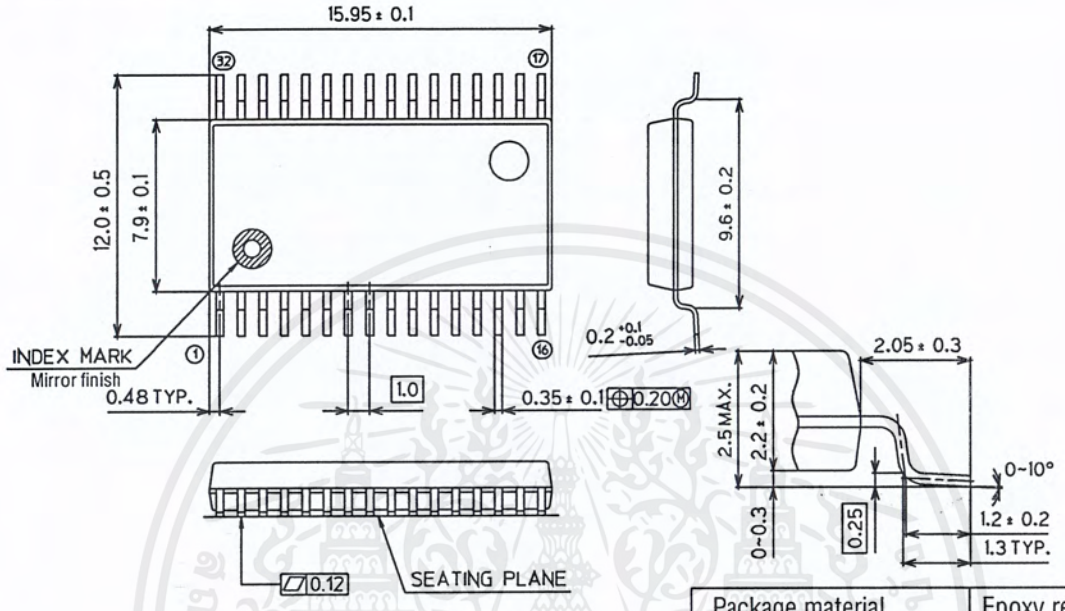
Package material	Epoxy resin
Lead frame material	Cu alloy
Pin treatment	Solder plating
Solder plate thickness	5 μm or more
Package weight (g)	1.00 TYP.

Notes for Mounting the Surface Mount Type Package

The SOP, QFP, TSOP, SOJ, QFJ (PLCC), SHP and BGA are surface mount type packages, which are very susceptible to heat in reflow mounting and humidity absorbed in storage.
Therefore, before you perform reflow mounting, contact Oki's responsible sales person for the product name, package name, pin number, package code and desired mounting conditions (reflow method, temperature and times).

(Unit : mm)

SSOP32-P-430-1.00-K



Package material	Epoxy resin
Lead frame material	42 alloy
Pin treatment	Solder plating
Solder plate thickness	5 μm or more
Package weight (g)	0.60 TYP.

Notes for Mounting the Surface Mount Type Package

The SOP, QFP, TSOP, SOJ, QFJ (PLCC), SHP and BGA are surface mount type packages, which are very susceptible to heat in reflow mounting and humidity absorbed in storage.
Therefore, before you perform reflow mounting, contact Oki's responsible sales person for the product name, package name, pin number, package code and desired mounting conditions (reflow method, temperature and times).

4) Notices on use

Note the following when replacing devices as the ASYNC pin is differently treated between the MSM82C84A and the MSM82C84A-5/MSM82C84A-2:

Case 1: When only a pullup resistor is externally connected to.
The MSM82C84A can be replaced by the MSM82C84A-2.

Case 2: When only pulldown resistor is externally connected to.
When the pulldown resistor is 8 kiloohms or less, the MSM82C84A can be replaced by the MSM82C84A-2.
When the pulldown resistor is greater than 8 kiloohms, use a pulldown resistor of 8 kiloohms or less.

Case 3: When an output of the other IC device is connected to the device.
The MSM82C84A can be replaced by the MSM82C84A-2 when the I_{OL} pin of the device to drive the $\overline{\text{ASYNC}}$ pin of the MSM82C84A-2 has an allowance of 100 μA or more.

หนังสืออ้างอิง

1. รศ.ดร. ปราโมทย์ วาดเขียน และ ศจ.ดร.วิวัฒน์ กิรานนท์ , 2530 , **พื้นฐานการสื่อสารข้อมูล** , อักษรสยามการพิมพ์
2. ศจ.ดร. วิวัฒน์ กิรานนท์ , **การสื่อสารข้อมูลดิจิตอล** , 2535 , โรงพิมพ์คุรุสภาลาดพร้าว
3. ดร.สุวิพล สิทธิชีวภาพ, **พื้นฐานแห่งการสื่อสารข้อมูล** , 2521, ด่านสุทธาการพิมพ์
4. ผศ.สมยศ จุณณะปิยะ, **การใช้งานไมโครคอนโทรเลอร์ตระกูล MCS51** , 2537, อักษรสยามการพิมพ์
5. John Bellamy , **Digital Telephony** ,1991,New York
6. Green DC. , **Data Communication** , 1995,Essex,Longman Scientific and Technical
7. David A. Stamper, **Business Data Communication** , 1991,Redwood City, CA.
8. ดร.พิพัฒน์ หิรัณย์วณิชชากร ,**ระบบการสื่อสารข้อมูลและเครือข่ายคอมพิวเตอร์** , 2543 , เอช. เอ็น กรุ๊ป จำกัด

กิตติกรรมประกาศ

ปริญญานิพนธ์ชิ้นนี้ สำเร็จลุล่วงได้ด้วยดี ด้วยความช่วยเหลือจากบุคคลหลายๆท่าน ทั้งในด้านความรู้ ความช่วยเหลือ เครื่องและอุปกรณ์ต่างๆ ตลอดจนถึงการสั่งสอน ให้คำแนะนำ อีกทั้งยังคอยเป็นกำลังใจในปริญญานิพนธ์นี้ซึ่งต้องกล่าวถึง

บิดา มารดา ที่คอยเป็นกำลังใจและสนับสนุนในเรื่องต่างๆ ในการทำปริญญานิพนธ์ชิ้นนี้

อาจารย์กฤษณ์ วงรุจิระ และดร.สุทธิชัย นพนาคีพงษ์ อาจารย์ที่ปรึกษาในการทำปริญญานิพนธ์ที่คอยให้คำแนะนำ ความช่วยเหลือ ให้คำปรึกษาในเรื่องต่างๆ และยังให้ความสนับสนุนในทุกๆด้าน

คุณสัมฤทธิ์ จินดาศรี ที่ช่วยเหลือและให้คำแนะนำในเรื่องของฮาร์ดแวร์ และคอยให้ความห่วงใยแก่พวกเราตลอดมาตั้งแต่เริ่มต้น จนสำเร็จลุล่วงไปด้วยดี

คุณเศรษฐกร กาเมือง ที่คอยให้คำปรึกษาในเรื่องข้อมูลและเทคนิคต่างๆในการทำฮาร์ดแวร์ที่เป็นประโยชน์อย่างมาก

และขอขอบคุณ คุณรัฐศาสตร์ อมฤทธิ์ เป็นอย่างมากที่คอยช่วยเหลือและให้คำแนะนำที่เป็นประโยชน์ และคอยให้กำลังใจเสมอ

คุณวริทธิ สันธิเดช , คุณอรนุช เลิศสุวรรณกิจ , คุณวรุฒม์ บุญธรรม , คุณอังคณา โชติเวชกุล, คุณสงคราม สุนทราญ , คุณธีรดา จันทรัศมี , คุณเจ จันทร์ศุภฤกษ์ , คุณปัทมา กรีสุธา ที่คอยให้กำลังใจให้เสมอมานับตั้งแต่เริ่มปริญญานิพนธ์ชิ้นนี้

คณะผู้จัดทำ

22 มีนาคม 2544