ระบบแสดงเวลาการเดินทางของรถประจำทาง

BUS'S ARRIVING SYSTEM

โดย

นายนัฐพล สมบูรณ์พงษ์ 45015013

นายปรีชา แจ่มแจ้ง 45015016

อาจารย์ที่ปรึกษา

อ.ธเนศ พัฒนธาดาพงษ์

ปริญญานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิศวกรรมศาสตรบัณฑิต

สาขาวิชาวิศวกรรมโทรคมนาคม

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

ปีการศึกษา 2547

ระบบแสดงเวลาการเดินทางของรถประจำทาง

BUS'S ARRIVING SYSTEM

โดย

นายนัฐพล สมบูรณ์พงษ์

นายปรีชา แจ่มแจ้ง

ปริญญานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิศวกรรมศาสตรบัณฑิต

สาขาวิชาวิศวกรรมโทรคมนาคม

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

ปีการศึกษา 2547

ปริญญานิพนธ์ปีการศึกษา 2547

ภาควิชาวิศวกรรมโทรคมนาคม

คณะวิศวกรรมศาสตร์ สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

เรื่อง ระบบแสดงเวลาการเดินทางของรถประจำ

BUS'S ARRIVING SYSTEM

ผู้จัดทำ

1. นายนัฐพล สมบูรณ์พงษ์ 45015013

2. นายปรีชา แจ่มแจ้ง 45015016

..... อาจารย์ที่ปรึกษา

(อ.ธเนศ พัฒนธาดาพงษ์)

ระบบแสดงเวลาการเดินทางของรถประจำทาง

BUS'S ARRIVING SYSTEM

โดย นายนัฐพล สมบูรณ์พงษ์ 45015013

นายปริชา แจ่มแจ้ง 45015016

อาจารย์ที่ปรึกษา อ.ธเนศ พัฒนธาดาพงษ์

บทคัดย่อ

โครงงานนี้เป็นการนำเสนอระบบแสดงเวลาเดินทางของรถประจำทาง โดยอาศัย หลักการบันทึกค่าเวลาที่รถประจำทางใช้ในการเดินทางจากป้ายหนึ่งไปยังป้ายที่อยู่ถัดไปตลอดสายการเดินรถ โดยข้อมูลจะถูกจัดเก็บในฐานข้อมูล และนำค่าเวลาที่ได้มาทำการหาค่าเวลาเฉลี่ยในละวันที่ใช้ในการเดินทางในแต่ละป้าย และแสดงเวลาที่รถประจำทางจะมาถึงผ่านบอร์ดแสดงผลที่ติดตั้งอยู่ในแต่ละป้าย และสามารถตรวจสอบข้อมูลผ่านทางระบบโทรศัพท์มือถือโดยใช้เทคนิค WAP ได้

ABSTRACT

This project is presented the bus's arriving system. We recorded time which bus use to move from bus stop to next bus stop around trip in each day. The recorded time will be use to calculate average time which bus used to move hop by hop around trip in each day. The result of calculation show in term of time which bus will come by using display board at the bus stop and possible to check via mobile phone by using WAP technique.

สารบัญ

หน้าที่

สารบัญรูปภาพ

บทที่ 1 บทนำ

บทที่ 2 ทฤษฎีหรือหลักการ

2.1 ข้อมูลที่อยู่ในรูปแบบของสัญญาณอนาลอกและดิจิตอล 5

2.1.1 ข้อมูล (Data) 5

2.1.2 สัญญาณ (Signal) 6

2.1.3 ข้อมูลและสัญญาณ (Data and Signal) 6

2.1.4 การส่งผ่าน (Transmission) 6

2.1.5 การส่งผ่านข้อมูลดิจิตอล (Digital Transmission) 7

2.2 รูปแบบของการสื่อสารข้อมูล 7

2.2.1การสื่อสารแบบอนุกรม 7

2.2.2 การสื่อสารแบบอะซิงโครนัส (Asynchronous Transmission) 8

2.2.3 การสื่อสารแบบซิงโครนัส (Synchronous Transmission) 9

2.2.4 การใช้บิตตรวจข้อผิดพลาด 10

2.2.5 ทิศทางของการสื่อสารข้อมูล 11

2.3 เอฟเอสเค(Frequency Shift Keying:FSK) 11

2.3.1การมอดูเลตแบบเอฟเอสเค 11

2.3.1.1 BFSK 11

2.3.1.2 M-ary FSK 13

2.3.2วงจรภาครับของ BFSK 14

2.3.3 การมอดูเลตสัญญาณดิจิตอลทางความถี่ 16

2.3.4 การดีมอดูเลตสัญญาณดิจิตอลทางความถี่ 17

2.4 เอฟเอ็ม (FM) 18

2.4.1ระบบการรับและส่งเอฟเอ็ม 18

2.4.1.1ภาคส่ง 18

2.4.1.2 ภาครับ 18

2.4.2 การมอดูเลตทางความถี่ (Frequency Modulation: FM) 19

2.4.3 การสร้างสัญญาณ FM 20

2.4.4 แบนด์วิดธ์ของสัญญาณเอฟเอ็ม 22

2.4.5การดีมอดูเลตทางความถี่ (Frequency Demodulation) 22

สารบัญ(ต่อ)

	หน้าที่
2.4.5.1 ดีแทกเตอร์แบบหาความชัน	22
2.4.5.2 ซีโรครอสซิงดีเทกเตอร์	23
2.4.5.3 เฟสล็อกลูป	24
2.5 WAP (Wireless Application Protocol)	30
2.5.1 ระบบของ WAP เมื่อเปรียบเทียบกับ WEB เป็นอย่างไร	31
2.5.2 WAP มีประโยชน์อย่างไร	31
2.6 Wireless Markup Language (WML)	31
2.7 การทำงานของ MySQL	34
2.8 การทำงานของ PHP	34
2.9 การทำงานของMCS-51	35
2.10 การเชื่อมต่อ HARDWARE ผ่านทางพอร์ต RS-232	36
2.11 รูปแบบของการสื่อสารข้อมูลอนุกรม	37
2.12 การทำงานร่วมกันระหว่างโปรแกรมต่างๆ	39
บทที่ 3 การคำนวณและการสร้าง	
3.1 การออกแบบวงจรเข้ารหัส	41
3.1.1 วงจรมัลติเพล็กซ์	41
3.1.2 วงจร ไบนารีเคาน์เตอร์	42
3.1.3 วงจรออสซิลเลเตอร์	42
3.2 วงจรเครื่องรับและเครื่องส่งเอฟเอ็ม	44
3.3 การทำงานของเอฟเอสเคมอดูเลเตอร์และดีมอดูเลเตอร์	46
3.3.1 เอฟเอสเคมอดูเลเตอร์	46
3.3.2 เอฟเอสเคดีมอดูเลเตอร์	49
3.4 การออกแบบบอร์ดแสดงผล	53
3.5 การออกแบบวงจรในส่วนของไมโครคอนโทรลเลอร์	54
3.6 การออกแบบโปรแกรม	56
3.6.1 การทำงานในส่วนของไคลเอนต์	56
3.6.2 การทำงานในส่วนของเซิร์ฟเวอร์คอมพิวเตอร์	58
3.6.3 การทำงานในส่วนของการประมาณหาค่าเวลาเฉลี่ย	59
3.6.4 การทำงานในส่วนของการค้นหารถประจำทางที่ต้องการรอใกล้สุดอยู่อีกกี่ป้าย	60
3.6.5 การทำงานในส่วนของการแสดงผล	61

สารบัญ(ต่อ)

	หน้าที่
บทที่ 4 การทดลองและผลการทดลอง	
4.1 การทดลอง	62
4.2 ผลการทดลอง	67
บทที่ 5 บทวิจารณ์และบทสรุป	
5.1 ปัญหาที่เกิดขึ้นจากการทดลอง	76
5.2 แนวทางในการแก้ปัญหา	76
ภาคผนวก	
หนังสืออ้างอิง	

สารบัญรูปภาพ

	หน้าที่
รูปที่ 1.1 แสดงบล็อกไดอะแกรมของเครื่องส่งจากรถประจำทาง	1
รูปที่ 1.2 แสดงบล็อกไดอะแกรมของเครื่องรับภายในป้ายรถประจำทาง	2
รูปที่ 1.3 แสดงบิตข้อมูลที่ส่ง	3
รูปที่ 2.1 แสดงการส่งผ่านข้อมูลแบบอนุกรม	7
รูปที่ 2.2 แสดงการทำงานในรูปแบบอะซิงโครนัส	8
รูปที่ 2.3 แสดงบล็อกข้อมูลการสื่อสารแบบซิงโครนัส	9
รูปที่ 2.4 แสดงการสื่อสารแบบทางเดียว	11
รูปที่ 2.5 แสดงการสื่อสารแบบฮาฟดูเพล็กซ์	11
รูปที่ 2.6 แสดงการสื่อสารแบบฟูลดูเพล็กซ์	11
รูปที่2.7 แผนภาพแสดงซิกนัลสเปซ ของระบบ BFSK	13
รูปที่ 2.8 แผนภาพวงจรรับของระบบ BFSK 466 ไฟฟ้าสื่อสาร	15
รูปที่ 2.9 แผนภาพซิกนัลสเปซ สำหรับการตัดสินใจบิตของระบบ BFSK	15
รูปที่ 2.10 แสดงการทำงานพื้นฐานของเอฟเอสเค	16
รูปที่ 2.11 แสดงสัญญาณเอฟเอสเค	17
รูปที่ 2.12 แสดงบล็อกไดอะแกรมภาคส่งเอฟเอ็ม	18
รูปที่ 2.13 แสดงบล็อกไดอะแกรมของเครื่องรับเอฟเอ็ม	19
รูปที่ 2.14 แสดงสัญญาณคลื่นพาห์สัญญาณข้อมูลและสัญญาณมอดูเลตแบบFM	21
รูปที่ 2.15 วงจรดีมอดูเลตสัญญาณเอฟเอ็มแบบหาความชัน	23
รูปที่ 2.16 ขั้นตอนการดีมอดูเลตสัญญาณเอฟเอ็มโดยวิธีซีโรครอสซิงดีเทกเตอร์	23
รูปที่ 2.17 แสดงแผนผังของเฟสล็อคลูป	24
รูปที่ 2.18 (a) บล็อกไดอะแกรมของเฟสล็อคลูป (b) ทรานสเฟอร์ฟังก์ชั่นของ VCO (c) ทรานสเฟอร์ฟังก์ชั่นของเฟสดีเทคเตอร์	26
รูปที่ 2.19 ทรานเซี้ยนของเฟสล็อคลูปอันเนื่องมาจากการเปลี่ยนแปลงความถี่ของสัญญาณอ้างอิง	27
รูปที่ 2.20 แสดงตัวอย่างลูปฟิลเตอร์	28
รูปที่ 2.21 แสดงผัง Programmable divider โดยใช้ IC ตระกูล TTL	29
รูปที่ 2.22 แสดงระบบการทำงานของ WAP	30
รูปที่ 2.23 เปรียบเทียบระบบ WAP และWEB	31
รูปที่ 2.24 แสดงการส่งข้อมูลแบบอนุกรมด้วยความเร็ว 9600 บิตต่อวินาที	36
รูปที่ 2.25 แสดงการส่งข้อมูลขนาด 8 บิตแบบอนุกรมพร้อมด้วย บิตเริ่มต้น, บิตพาริตี้, บิตหยุด	37
รูปที่ 2.26 แสดงระดับแรงดันสัญญาณของพอร์ตอนุกรม RS-232 กับ TTL	38

สารบัญรูปภาพ(ต่อ)

หน้าที่

รูปที่ 2.27 แสดงวงจรภายในของไอซี MAX232 38
รูปที่ 2.28 แสดงการทำงานร่วมกันระหว่างโปรแกรมต่างๆ 39
รูปที่ 2.29 แสดงหน้าจอการทำงานของโปรแกรมอีมูเลเตอร์ WINWAP 3.1 PRO 40
รูปที่ 3.1 วงจรเข้ารหัสดิจิตอล 41
รูปที่ 3.2 แสดงสัญญาณของวงจร Multiplex 41
รูปที่ 3.3 แสดงสัญญาณของวงจร Binary counter 42
รูปที่ 3.4 แสดงสัญญาณของวงจรออสซิลเลเตอร์ 42
รูปที่ 3.5 แสดงวงจรของไอซีเบอร์NE555 43
รูปที่ 3.6 แสดงบิตข้อมูลที่จะส่งทั้งหมด 44
รูปที่ 3.7 บล็อกไดอะแกรมของเครื่องส่งเอฟเอ็ม 44
รูปที่ 3.8 บล็อกไดอะแกรมของเครื่องรับเอฟเอ็ม 45
รูปที่ 3.9 แสดงวงจรเครื่องส่งเอฟเอ็ม 45
รูปที่ 3.10 แสดงวงจรเครื่องรับเอฟเอ็ม 46
รูปที่ 3.11 แสดงวงจรภายในของ XR2206 48
รูปที่ 3.12 แสดงวงจรเอฟเอสเคมอดูเลเตอร์โดยใช้ ไอซีเบอร์ XR 2206 49
รูปที่ 3.13 แสดงวงจรภายในของ XR 2211 50
รูปที่ 3.14 แสดงวงจรเอฟเอสเคดีมอดูเลเตอร์โดยใช้ ไอซีเบอร์ XR 2211 52
รูปที่ 3.15 แสดงวงจรแสดงผล 53
รูปที่ 3.16 แสดงวงจรไมโครคอนโทรลเลอร์ 54
รูปที่ 3.17 แสดงโฟลว์ชาร์ตการทำงานในส่วนของไคลเอนต์ 56
รูปที่ 3.18 แสดงโฟลว์ชาร์ตการทำงานในส่วนของคอมพิวเตอร์ (เซิร์ฟเวอร์) 58
รูปที่ 3.19 แสดงโฟลว์ชาร์ตการทำงานในส่วนของการประมาณหาค่าเวลาเฉลี่ย 59
รูปที่ 3.20 แสดงการทำงานในส่วนของการค้นหารถประจำทางที่ต้องการรอใกล้สุดอยู่อีกกี่ป้าย 60
รูปที่ 3.21 แสดงการทำงานในส่วนของการแสดงผล 61
รูปที่ 4.1 แสดงวงจรเข้ารหัส 62
รูปที่ 4.2 วงจรเอฟเอสเคมอดูเลเตอร์ 63
รูปที่ 4.3 วงจรเอฟเอสเคดีมอดูเลเตอร์ 64
รูปที่ 4.4 วงจรไมโครคอนโทรลเลอร์ 65
รูปที่ 4.5 วงจรแสดงผล 66
รูปที่ 4.6 แสดงสัญญาณวงจรเข้ารหัส 67
รูปที่ 4.7 แสดงสัญญาณเอฟเอสเคมอดูเลเตอร์ 67
รูปที่ 4.8 แสดงสัญญาณเอฟเอสเคดีมอดูเลเตอร์ 68

สารบัญรูปภาพ(ต่อ)

หน้าที่

รูปที่ 4.9 แสดงสัญญาณวงจรไมโครคอนโทรลเลอร์ที่ขา 10 ของไอซี MCS-5 68

รูปที่ 4.10 แสดงสัญญาณวงจรไมโครคอนโทรลเลอร์ที่ขา 11 ของไอซี MCS-51 69

รูปที่ 4.11 แสดงสัญญาณวงจรไมโครคอนโทรลเลอร์ที่ขา 1 ของไอซี 74HC164 69

รูปที่ 4.12 แสดงสเปกตรัมเอาต์พุทของเครื่องส่งเอฟเอ็ม 27 MHz 70

รูปที่ 4.13 เปรียบเทียบสัญญาณที่ใช้มอดูเลตกับสัญญาณที่ดีเทคได้ของเครื่องรับ เอฟเอ็ม 27 เมกะเฮิรตซ์ 71

รูปที่ 4.14 แสดงสเปกตรัมที่วัดได้ที่อินพุตของเครื่องรับเอฟเอ็ม 27 MHz 72

รูปที่ 4.15 แสดงการเก็บข้อมูลเข้าไปยังฐานข้อมูลของMySQL 73

รูปที่ 4.16 แสดงหน้าจอการทำงานของ VISUAL C++ 73

รูปที่ 4.17แสดงหน้าจอการทำงานของ VISUAL C++ กรณีเริ่มต้นการทำงาน 74

รูปที่ 4.18 แสดงหน้าจอการทำงานของโปรแกรม WinWAP 3.1 PRO ในการรับค่า 74

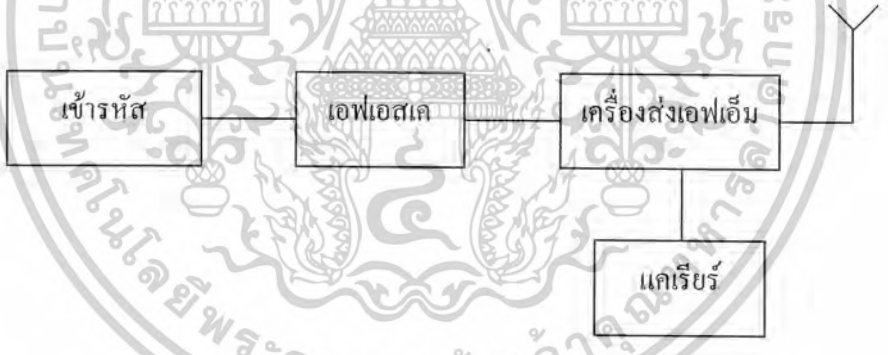
รูปที่ 4.19 แสดงหน้าจอการทำงานของโปรแกรม WinWAP 3.1 PRO แสดงผลการค้นหา 75

รูปที่ 4.20 แสดงหน้าจอการทำงานของโปรแกรม WinWAP 3.1 PRO กรณีไม่มีข้อมูล 75

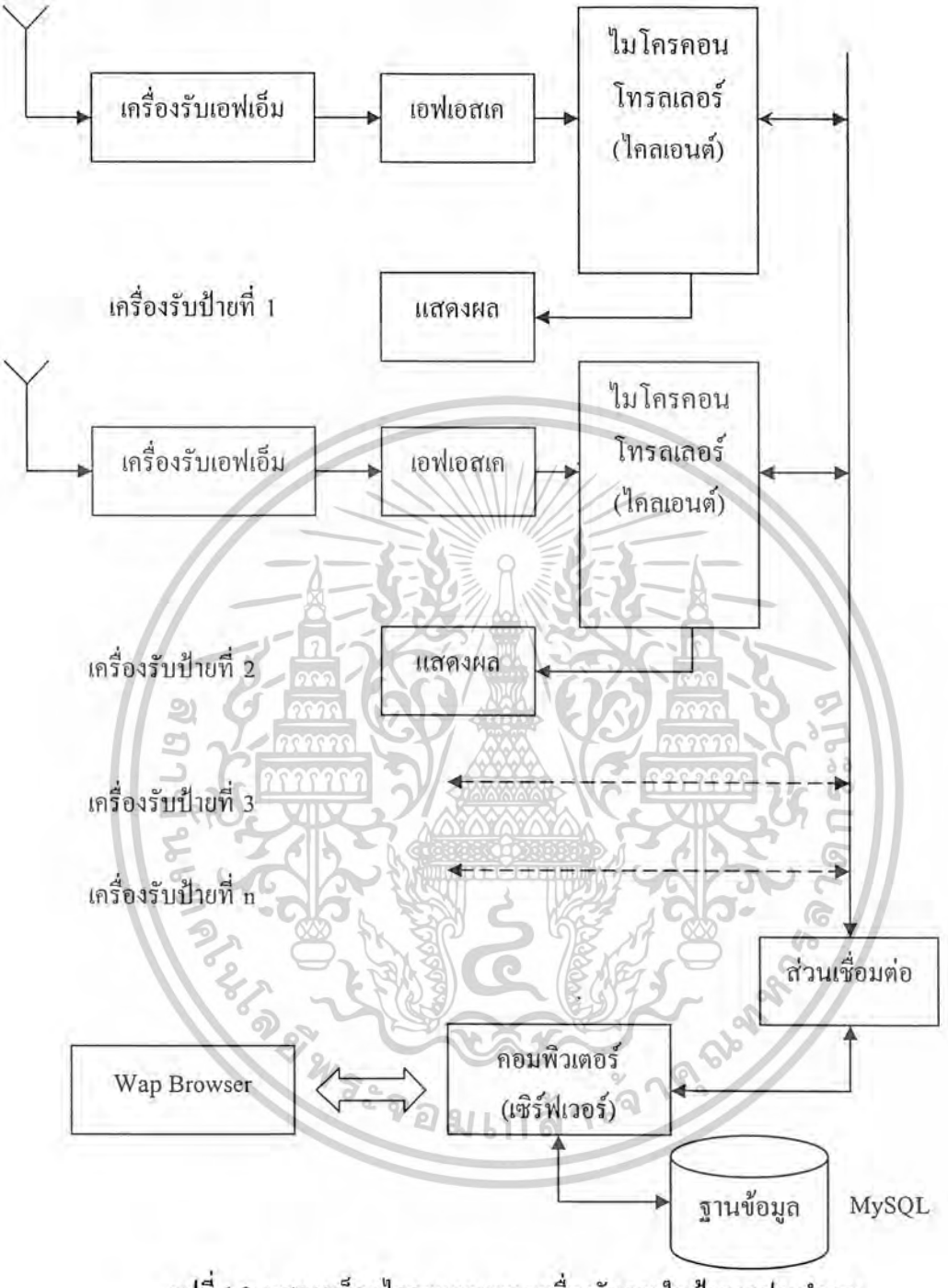
บทที่ 1
บทนำ

ปัจจุบันนี้การเดินทางโดยใช้รถประจำทางมีบทบาทเข้ามาในชีวิตประจำวันมากขึ้น ทำให้ความต้องการในการใช้ระบบขนส่งมวลชนได้เพิ่มขึ้นทุกปี การรอรถประจำทางในแต่ละครั้งต้องใช้ระยะเวลาในการรอนานจากปัญหาดังกล่าวถ้าสามารถประมาณการเวลาของรถประจำทางแต่ละคันที่เข้ามายังป้ายรถประจำทางที่รออยู่นั้นทำให้ไม่เสียเวลาในการรอรถประจำทางสามารถแก้ไขได้โดยการบันทึกค่าสถิติโดยอาศัยระยะเวลาการเดินทางปรกติตลอดทั้งสายการเดินรถ เพื่อนำมาใช้ในการหาค่าเฉลี่ยในแต่ละช่วงเวลาก็จะสามารถคำนวณระยะเวลาในการเดินทางไปยังสถานที่ต่างๆได้ระบบนี้จะแสดงค่าต่างๆผ่านทางบอร์ดแสดงผล(Display Board) ที่ติดภายในป้ายรถประจำทาง และยังสามารถรับรู้ข้อมูลได้อีกทางหนึ่งโดยการใช้โทรศัพท์มือถือผ่านทาง Wap Browserโดยการป้อนข้อมูลทางโทรศัพท์มือถือซึ่งโทรศัพท์มือถือจะส่งข้องมูลไปยัง Wap Gateway เพื่อให้ Wap Server ส่งข้อมูลกลับมายังโทรศัพท์มือถือ สามารถทราบเวลาเดินทางของรถประจำทางในทุกสถานที่ได้โดยผ่านทางโทรศัพท์มือถือ

บล็อกไดอะแกรมการทำงานโดยรวมของระบบแสดงเวลาการเดินทางของรถประจำทาง ดังรูปที่ 1.1



รูปที่ 1.1 แสดงบล็อกไดอะแกรมของเครื่องส่งจากรถประจำทาง



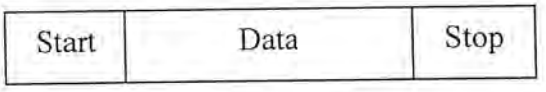
รูปที่ 1.2 แสดงบล็อกไดอะแกรมของเครื่องรับภายในป้ายรถประจำทาง

การทำงานของระบบแสดงเวลาการเดินทางของรถประจำทางนั้นมีอยู่ด้วยกันหลายส่วน

ส่วนเครื่องส่ง

- ตัวเข้ารหัส(ENCODER) เป็นวงจรเข้ารหัสสัญญาณรหัสรถประจำทาง (ID bus) แต่ละสาย ซึ่งรหัสของรถแต่ละคันจะแตกต่างกันออกไป โดยรหัสนี้สามารถเปลี่ยนแปลงได้โดยการเปลี่ยนที่ดิปสวิตช์ ซึ่งจะเป็นรูปแบบรหัสดิจิตอลที่มีขนาด 8 บิตซึ่งรหัสทั้ง 8 บิตนี้จะรวมเอาทั้งสตาร์ทบิต (start bit) และ

สตอปบิต(stop bit)โดยสตอปบิตถูกสร้างขึ้นมาจากสตาร์ทบิตถัดไปไว้เพื่อตัวรับสัญญาณสามารถถอดรหัสสัญญาณออกมาได้ไม่มีความผิดพลาดเกิดขึ้น โดยมีไอซี 74HC151และไอซี 74HC93 คอยทำหน้าที่ในการเข้ารหัสข้อมูลไอซีทั้งสองตัวจะทำหน้าที่ประสานจังหวะการมัลติเพล็กซ์เพื่อได้สัญญาณที่เป็นรูปแบบอนุกรม



รูปที่ 1.3 แสดงบิตข้อมูลที่ส่ง

- เอฟเอสเค (FSK) เป็นบล็อกที่ทำหน้าที่ในการแปลงสัญญาณจากอนาลอกเป็นดิจิตอล เพื่อลดสัญญาณรบกวนเนื่องจากนอยส์ต่างๆจากภายนอก และยังทำให้สัญญาณที่รับได้มีคุณภาพมากกว่าที่จะส่งเป็นสัญญาณอนาลอกโดยไอซี XR2206 เป็นอุปกรณ์ที่ทำหน้าที่แปลงสัญญาณเป็นรูปแบบเอฟเอสเคเพื่อนำสัญญาณออกไปสู่วงจรเอฟเอ็มต่อไป

- เครื่องส่งเอฟเอ็ม (FM-TRANSMITTER) ทำหน้าที่ในการส่งสัญญาณเอฟเอสเคเพื่อให้ส่งผ่านไปยังตัวรับที่อยู่ตามแต่ละป้ายรถประจำทางได้โดยสามารถกำหนดความถี่ที่จะส่งออกไปได้ ซึ่งความถี่ที่ส่งไปจะต้องไม่ไปรบกวนการทำงานของอุปกรณ์ต่างๆ

- สายอากาศ (Antenna) ใช้สำหรับส่งคลื่นออกไปยังเครื่องรับ โดยกำหนดการกระจายของคลื่นให้อยู่ในขอบเขตที่จำกัด

ส่วนเครื่องรับ

- เครื่องรับเอฟเอ็ม (FM Receiver) เป็นตัวรับสัญญาณความถี่ที่ส่งมาจากเครื่องส่งโดยนำความถี่ที่รับได้มาทำการถอดรหัส

- เอฟเอสเคดีมอดูเลเตอร์ (FSK Demodulator) ทำหน้าที่ในการแปลงสัญญาณความถี่ในรูปแบบสัญญาณอนาลอกแปลงกลับไปเป็นสัญญาณรูปแบบดิจิตอล

- บอร์ดแสดงผล (Display Board) เป็นส่วนที่ทำหน้าที่ในการแสดงผลว่าตอนนี้รถประจำทางสายที่เท่าไหร่จะเข้ามาในเวลาประมาณเท่าใดและในช่วงเวลานั้นสายรถประจำทางสายนั้นถึงป้ายไหนแล้ว ซึ่งแบ่งการทำงานของการแสดงผลออกเป็น 2 ส่วนคือส่วนแรกเป็นส่วนที่ติดตั้งระหว่างป้ายเป็นส่วนที่รับข้อมูลมาจากไมโครคอนโทรลเลอร์ส่วนที่ 2 เป็นส่วนที่แสดงผลผ่านทางโทรศัพท์มือถือ

-ไมโครคอนโทรลเลอร์ (ไคลเอนต์) โดยส่วนนี้ทำหน้าที่หลายอย่างหน้าที่แรกคือทำการรับสัญญาณที่เป็นรูปแบบดิจิตอลจากวงจร FSK ซึ่งสัญญาณดิจิตอลจะเป็นข้อมูลแบบ อะซิงโครนัส และส่งสัญญาณเป็นแบบอนุกรมโดยผ่านวงจร SIPO (Serial Input Parallel Output) ซึ่งเราสามารถนำข้อมูลออกมาจากสัญญาณนี้ได้โดยการเขียนโปรแกรมในการรับข้อมูลแล้วเปรียบเทียบหาข้อมูลแต่ละช่วงเวลาโดยข้อมูลที่ได้มาแล้วก็จะทำการเก็บข้อมูลไว้เพื่อให้เซิร์ฟเวอร์นำข้อมูลไปประมวลผล

-เซิร์ฟเวอร์(Server)เป็นส่วนประมวลผลข้อมูลหลักที่ใช้ในการรับส่งเฟรมข้อมูลต่างๆที่ได้มาจากไคลเอนต์เพื่อเก็บข้อมูลเข้าไปยังฐานข้อมูลแล้วทำการคำนวณค่าต่างๆจึงส่งไปยังไมโครคอนโทรลเลอร์

ตัวต่างๆ โดยการกวาดสัญญาณจากแอดเดรสแรกส่งข้อมูลไปที่ไมโครคอนโทรลเลอร์MCS-51แต่ละตัวถ้ามีตัวใดมีรหัสแอดเดรสตรงกันกับที่ได้ส่งข้อมูลไปทำการรับข้อมูลต่างๆที่ได้มาเพื่อนำไปแสดงผลและถ้ามีข้อมูลที่ต้องการส่ง(กรณีที่มีรถประจำทางผ่านป้ายนั้น) ก็จะส่งไปยังเซิร์ฟเวอร์และตัวเซิร์ฟเวอร์เองก็จะทำหน้าที่เป็นตัวรับข้อมูล เพื่อนำข้อมูลจากตัวส่ง(ไมโครคอนโทรลเลอร์MCS-51)เข้ามาประมวลผลโดยไมโครคอนโทรลเลอร์ตัวอื่นก็จะยังไม่ส่งข้อมูลในกรณีที่ตัวแรกยังส่งข้อมูลไม่เสร็จ เมื่อตัวแรกส่งข้อมูลเสร็จแล้วเซิร์ฟเวอร์ก็จะทำการเรียกแอดเดรสต่อไป ถ้าไคลเอนต์มีข้อมูลที่ต้องการจะส่ง ต้องให้เซิร์ฟเวอร์ทำการกวาดสัญญาณมาถึงแอดเดรสของตัวเองก่อนก่อนจึงจะส่งข้อมูลได้ซึ่งความสำคัญของส่วนนี้ยังไม่หมดเพราะจะต้องมีการเก็บฐานข้อมูลไว้เพื่อใช้ในการหาค่าเฉลี่ยเวลาของรถประจำทางเพื่อให้เวลาที่ได้นั้นมีความใกล้เคียงความเป็นจริงมากที่สุด

- WAP (Wireless Application Protocol) เป็นเทคโนโลยีที่อนุญาตผู้ใช้ทั่วไปสามารถเข้าถึงอินเตอร์เน็ตจากโทรศัพท์มือถือ สามารถรับและแสดงผลบนจอภาพโดยจะใช้งานบนเว็บไคเอ็นและเว็บเซิร์ฟเวอร์โดยมี WAP Gateway เป็นสื่อกลางระหว่างอินเตอร์เน็ตและเครือข่ายไร้สายของอุปกรณ์

บทที่ 2
ทฤษฎีหรือหลักการ

2.1 ข้อมูลที่อยู่ในรูปแบบของสัญญาณอนาลอกและดิจิตอล

ในการส่งข้อมูลจากต้นทางไปยังปลายทางมีคุณสมบัติของสัญญาณข้อมูลอย่างหนึ่งที่น่าสนใจคือ การเคลื่อนที่ของข้อมูลและขบวนการจัดการกับข้อมูล เพื่อให้มั่นใจได้ว่าข้อมูลที่ส่งไปและรับมา สามารถที่จะเข้าใจกันได้สำหรับการศึกษาทั้งหมดนี้ มีจุดสำคัญที่เราจะเข้าไปเกี่ยวข้องด้วยกันคือปริมาณชนิดอนาลอกและชนิดดิจิตอล

ปริมาณชนิดอนาลอกและชนิดดิจิตอลเป็นปริมาณที่สอดคล้องกับปริมาณที่มีค่าแบบต่อเนื่องและไม่ต่อเนื่อง ตามลำดับ ศัพท์ 2 คำนี้จะถูกนำมาใช้กันเสมอในการสื่อสารข้อมูลอย่างน้อยที่สุดก็นำมาใช้ในเรื่อง 3 เรื่อง ดังต่อไปนี้คือ

- ข้อมูล (Data)
- สัญญาณ (Signal)
- การส่งผ่าน (Transmission)

2.1.1 ข้อมูล (Data)

ข้อมูลแบบอนาลอกนั้นเป็นข้อมูลที่เกิดขึ้นแล้วมีค่าต่อเนื่องในช่วงเวลาที่พิจารณา เป็นต้นว่าเสียงหรือภาพจะเป็นข้อมูลที่มีการแปรเปลี่ยนรูปแบบของความเข้มอย่างต่อเนื่อง อุณหภูมิและความดันก็เป็นปริมาณที่ต่อเนื่องที่จัดได้ว่าให้ข้อมูลแบบอนาลอกแต่สำหรับข้อมูลแบบดิจิตอลนั้นเป็นข้อมูลที่มีค่าเฉพาะที่กระโดดไปเป็นค่าที่ไม่ต่อเนื่อง เช่น ค่าเลขจำนวนเต็มเป็นต้น

ตัวอย่างที่เป็นที่รู้จักกันอย่างแพร่หลายมากที่สุด สำหรับข้อมูลอนาลอกก็คือข้อมูลเสียงซึ่งข้อมูลของระบบในการสื่อสารบางระบบ จะถูกออกแบบไว้สำหรับข้อมูลเลขฐานสอง ซึ่งจำนวนของรหัสที่แทนตัวอักษรจะใช้ลำดับของบิต ตัวอย่างของรหัสที่เก่าแก่มากที่สุดอันหนึ่งคือ รหัสของมอร์ส ปัจจุบันรหัสที่ใช้กันอย่างแพร่หลายที่สุดชนิดหนึ่งคือรหัส ASCII (American Standard Code for Information Interchanger) ASCII นอกจากจะได้รับความนิยมในอเมริกาแล้วยังได้รับความนิยมในที่ต่างๆ ด้วย ตัวอักขระแต่ละตัวของรหัสนี้แทนได้ด้วยเลขฐานสอง 7 หลัก ซึ่งเลขฐาน สอง 7 หลัก สามารถแทนอักขระที่ไม่ซ้ำกันได้ทั้งหมดถึง 128 ตัว ซึ่งมีจำนวนเกินตัวอักขระที่ใช้กันอยู่ ดังนั้นรหัสเลขฐานสองบางตัวจึงถูกนำมาแทนตัวอักขระสำหรับการควบคุม และตัวอักขระสำหรับการควบคุมส่วนหนึ่งก็ใช้ในการควบคุมสำหรับการพิมพ์และรหัสเลขฐานสองที่แทนตัวอักขระส่วนที่เหลือบางตัวก็จะใช้ในระบบการสื่อสาร โดยทั่วไปแล้ว การเข้ารหัสในการเก็บและการส่งผ่านข้อมูลจะใช้จำนวนเลขไบนารี่ 8 บิตต่อ 1 ตัวอักษร โดยบิตที่ 8 ก็คือพาริตี้บิต ที่ใช้สำหรับตรวจสอบความผิดพลาดบิตที่ 8 นี้ จะถูกกำหนดขึ้นเพื่อที่จะให้จำนวนของบิต 1 มีค่าเป็นจำนวนคู่เมื่อใช้การตรวจพาริตี้แบบคู่ และให้จำนวนของบิต 1 มีค่าเป็นจำนวนคี่เมื่อใช้การตรวจพาริตี้แบบคี่

2.1.2 สัญญาณ (Signal)

ในระบบของการสื่อสารข้อมูลจะเคลื่อนที่จากจุดหนึ่ง ไปยังอีกจุดหนึ่งในรูปแบบของสัญญาณไฟฟ้าซึ่งสัญญาณอนาลอกก็คือ คลื่นแม่เหล็กไฟฟ้าที่มีการเปลี่ยนแปลงอย่างต่อเนื่องและสัญญาณดังกล่าวนี้เองที่จะถูกส่งผ่านเข้าไปในตัวกลางชนิดต่างๆ ทั้งนี้ขึ้นอยู่กับสเปคตรัมของสัญญาณ ตัวกลางอาจเป็นสาย เช่น สายคู่บิดเกลียว สายโคแอคเชียล สายใยนำแสง หรืออาจจะเป็นตัวกลางแบบไร้สายเช่น ชั้นบรรยากาศหรือสุญญากาศ ส่วนสัญญาณดิจิตอลนั้นอาจเป็นขบวนของพัลส์โวลท์เตจ ที่ใช้ระดับของพัลส์ที่ส่งไปเป็นตัวแทนข้อมูล เช่น ใช้ระดับโวลท์เตจคงที่ค่าบวก แทนค่าไบนารี่ 1 และระดับโวลท์เตจลบแทนค่าไบนารี่ 0

ซึ่งจะเห็นได้ว่า แบนด์วิดธ์ที่กว้างนั้นก็จะทำให้เกิดสัญญาณที่มีความผิดพลาดน้อย สำหรับข้อมูลที่มีอัตรา x บิต ต่อวินาทีสามารถที่จะส่งได้โดยใช้สายส่งที่มีแบนด์วิดธ์ x Hz ด้วยคุณภาพดีพอสมควร แต่จะดีมากถ้าแบนด์วิดธ์เพิ่มขึ้นเป็น $2x$ Hz

2.1.3 ข้อมูลและสัญญาณ (Data and Signal)

จากที่ได้กล่าวมาแล้ว เรากำลังสนใจสัญญาณที่ใช้แทนข้อมูลอนาลอกและข้อมูลดิจิตอลในสถานการณ์โดยทั่วไปแล้วข้อมูลอนาลอกจะเป็นฟังก์ชั่นเกี่ยวกับเวลาและมีสเปคตรัมที่จำกัดอยู่ค่าหนึ่ง ซึ่งเราสามารถแทนข้อมูลอนาลอกดังกล่าว ได้ด้วยสัญญาณคลื่นแม่เหล็กไฟฟ้าที่มีค่าสเปคตรัมเดียวกัน และข้อมูลดิจิตอลนั้นสามารถแทนได้ด้วยสัญญาณดิจิตอล ที่มีค่าระดับโวลท์เตจที่แตกต่างกันสองระดับเพื่อแทนค่าเลขฐานสอง 0 และ 1

2.1.4 การส่งผ่าน (Transmission)

ความแตกต่างระหว่างสัญญาณดิจิตอลและอนาลอกนั้นควรพิจารณาให้ชัดเจนขึ้นอีกทั้งสัญญาณอนาลอกและสัญญาณดิจิตอลนั้น จะถูกนำส่งตัวกลางที่เหมาะสมและขบวนการในการดัดแปลงสัญญาณเพื่อให้เหมาะสมกับตัวกลางการส่งผ่านก็เป็นหน้าที่ของระบบส่งผ่าน (Transmission System) การส่งผ่านในรูปแบบของอนาลอกเป็นวิธีการส่งผ่านสัญญาณอนาลอกไป โดยไม่มีการเปลี่ยนแปลงสาระข้อมูล ไม่ว่าจะเป็นกรณีใดๆสัญญาณอนาลอกจะเกิดการลดทอนหลังจากที่เดินทางผ่านเข้าไปในตัวกลาง ดังนั้นเพื่อให้สัญญาณสามารถเดินทางไปถึงปลายทางได้ระยะทางไกลๆ ระบบของการส่งผ่านแบบอนาลอกก็จะมีตัวเพิ่มความแรงของสัญญาณ และยังเป็นการเพิ่มพลังงานให้กับสัญญาณรบกวนอีกด้วย และยิ่งมีการใช้ตัวขยายหลายๆตัวแบบอนุกรม เพื่อให้ได้ระบบในการส่งสัญญาณได้ไกลๆยิ่งทำให้สัญญาณผิดเพี้ยนไปยิ่งขึ้น

สำหรับข้อมูลแบบอนาลอก เช่น เสียง ความผิดพลาดเพียงส่วนเล็กน้อยสามารถยอมให้เกิด ขึ้นได้เพราะว่าข้อมูลยังสามารถเข้าใจได้ แต่สำหรับข้อมูลดิจิตอลการต่อตัวขยายแบบอนุกรมจะทำให้เกิดการผิดพลาด

2.1.5 การส่งผ่านข้อมูลดิจิตอล (Digital Transmission)

การส่งผ่านข้อมูลด้วยวิธีการนี้จะเกี่ยวข้องกับเนื้อหาของสัญญาณ สัญญาณดิจิตอลจะถูกส่งไปได้ในระยะทางที่จำกัดก่อนที่การลดทอนจะทำอันตรายต่อองค์ประกอบของข้อมูล ดังนั้นเพื่อ ให้การส่งสามารถที่ส่งไปได้ในระยะทางไกลๆ เราจึงใช้ตัวทวนสัญญาณ(Repeater) เพื่อกู้สัญญาณดั้งเดิมกลับคืนมา โดยที่ตัวทวนสัญญาณเมื่อได้รับสัญญาณดิจิตอลมาแล้ว จะทำการกู้รูปแบบของ 1 และ 0 กลับคืนมาอีกครั้ง และส่งต่อออกไปใหม่ ซึ่งทำให้สามารถเอาชนะการลดทอนลงไปได้

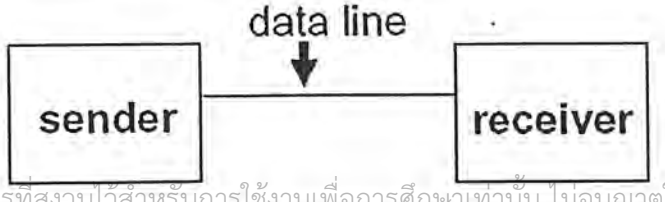
ด้วยเทคนิคอย่างเดียวกันกับที่ได้กล่าวมาแล้ว อาจนำมาใช้ได้กับสัญญาณอนาลอก ที่ใช้เป็นตัวส่งถ่ายข้อมูลดิจิตอล ณ ตำแหน่งในพื้นที่ๆเหมาะสม ระบบของการส่งผ่านก็จะใช้เครื่องทวนสัญญาณแทนที่จะเป็นตัวขยายสัญญาณ ตัวทวนสัญญาณจะกู้สัญญาณดิจิตอลกลับคืนมาจากสัญญาณอนาลอก และก็สร้างสัญญาณอนาลอกใหม่ขึ้นมา ทำให้ไม่เกิดการสะสมของสัญญาณรบกวนต่อๆไป

ในปัจจุบันมีแนวโน้มที่จะหันมาใช้การสื่อสารระบบดิจิตอลแทนระบบอนาลอก แม้ว่าจะมีการลงทุนใช้ระบบอนาลอกมาก่อนอย่างมากก็ตาม เหตุผลที่สำคัญคือ

- ดิจิตอลเทคโนโลยี การพัฒนาเทคโนโลยีของวงจรดิจิตอล LSI และ VLSI ทำให้ราคาและขนาดของวงจรดิจิตอลลดลง ในขณะที่เครื่องมือทางอนาลอกไม่ได้ลดลงเลย
- คุณภาพของข้อมูลสำหรับขบวนการทางดิจิตอล การใช้ตัวทวนสัญญาณแทนที่จะใช้ตัวขยายสัญญาณทำให้อิทธิพลของสัญญาณรบกวนไม่ถูกสะสม ทำให้เราสามารถส่งข้อมูลไปได้ระยะทางไกลๆ แม้ว่าคุณภาพของสายจะไม่ดีก็ตาม
- ความจุของการใช้งานมีมาก มันเป็นเรื่องที่สิ้นเปลืองมากในการที่จะต้องการสร้างทาง เดินการส่งผ่านข้อมูลที่มีแบนด์วิดธ์กว้างมากๆเช่น ช่องสัญญาณดาวเทียมและเส้น ใยนำแสง ดังนั้นการนำเอาขบวนการในการมัลติเพล็กซ์เข้ามาใช้งานดูจะเป็นประโยชน์อย่างมากในเรื่องของความจุ ซึ่งขบวนการมัลติเพล็กซ์ทางด้านเวลาดูจะเป็นวิธีการที่ง่ายและราคาถูกกว่าขบวนการมัลติเพล็กซ์ทางด้านความถี่
- ความปลอดภัยและความเป็นส่วนตัว เทคนิคการย่อข้อมูลพร้อมที่จะนำเข้ามาใช้กับข้อมูลดิจิตอลและพร้อมที่จะนำมาใช้กับข้อมูลอนาลอกที่ถูกดิจิตอลไว้แล้ว
- การรวมกันเข้าเป็นหนึ่งเดียวกัน ด้วยขบวนการทางข้อมูลพร้อมที่จะนำเข้ามาใช้กับข้อมูลดิจิตอล สัญญาณจะมีรูปแบบที่เหมือนกัน และสามารถดำเนินการได้ในลักษณะเดียวกันซึ่งมันก็จะทำให้ประหยัดและสะดวกในการรวมเสียง ภาพและข้อมูลดิจิตอลเข้าไว้ด้วยกัน

2.2 รูปแบบของการสื่อสารข้อมูล

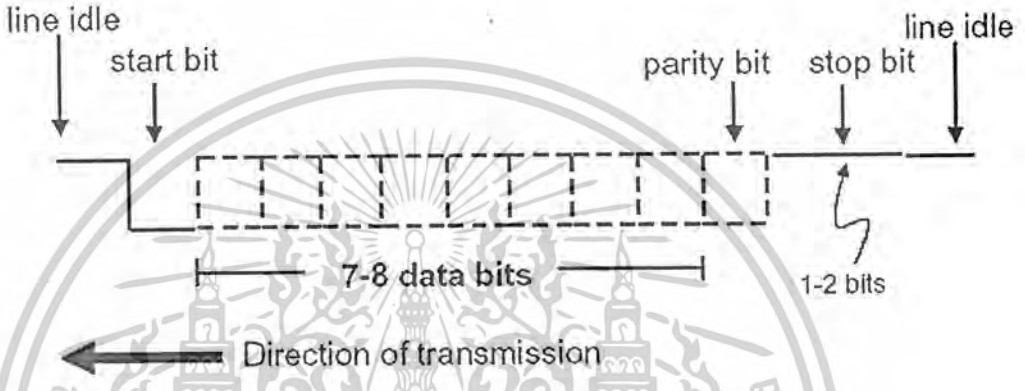
2.2.1การสื่อสารแบบอนุกรม



รูปที่ 2.1 แสดงการส่งผ่านข้อมูลแบบอนุกรม

การส่งผ่านข้อมูลแบบอนุกรมเป็นรูปแบบการส่งผ่านข้อมูลทีละบิตเรียงกันไป โดยผ่านสายนำสัญญาณเพียงเส้นเดียว การที่เราจะแยกข้อมูลออกมาได้จากสัญญาณที่ส่งมาจะต้องทำการหาจุดเริ่มต้นของข้อมูลและจุดสิ้นสุดของข้อมูล โดยมีการกำหนดโปรโตคอล (Protocols) ของการสื่อสารขึ้นมีรูปแบบการส่งอยู่ 2 รูปแบบ การสื่อสารแบบอะซิงโครนัส (Asynchronous Transmission) และการสื่อสารแบบซิงโครนัส (Synchronous Transmission)

2.2.2 การสื่อสารแบบอะซิงโครนัส (Asynchronous Transmission)

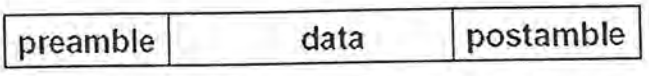


รูปที่ 2.2 แสดงการทำงานในรูปแบบอะซิงโครนัส

การสื่อสารแบบอะซิงโครนัส หรือเรียกอีกอย่างหนึ่งว่าเป็น การสื่อสารแบบระบุจุดเริ่มต้น และจุดสิ้นสุด (Start-Stop Transmission) ลักษณะของสัญญาณที่ใช้ในการติดต่อสื่อสารกันจะประกอบไปด้วย บิตเริ่มต้น (สตาร์ทบิต) บิตของข้อมูลที่สื่อสาร (transmission data) จำนวน 8 บิต บิตตรวจข้อผิดพลาด (parity bit) และบิตสิ้นสุด (stop bit) สำหรับบิตตรวจสอบข้อผิดพลาดจะใช้หรือไม่ใช้ก็ได้ ดังนั้นสัญญาณจึงต้องประกอบด้วยส่วนประกอบอย่างน้อย 3 ส่วน

จะเห็นว่าขณะที่ไม่มีข้อมูลส่งออกมาสถานะของการส่งจะเป็นแบบว่าง (Idle) ซึ่งจะมีระดับของสัญญาณเป็น 1 ตลอดเวลา เพื่อความแน่ใจว่าปลายทาง หรือฝ่ายรับยังคงติดต่อกับต้นทาง หรือฝ่ายส่งอยู่ เมื่อเริ่มจะส่งข้อมูลสัญญาณของอะซิงโครนัสจะเป็น 0 หนึ่งช่วงสัญญาณนาฬิกา ซึ่งบิตนี้เราเรียกว่าบิตเริ่มต้น ตามหลังของบิตเริ่มต้นจะเป็นบิตข้อมูลสำหรับ 1 ตัวอักษร ตามหลังบิตข้อมูลก็จะเป็นบิตตรวจข้อผิดพลาด แล้วจะตามด้วยบิตสิ้นสุด ถ้าไม่ใช่บิตตรวจข้อผิดพลาด ตามหลังบิตข้อมูลก็จะเป็นบิตสิ้นสุดเลย หลังจากนั้นถ้าไม่มีข้อมูลส่งออกมาสัญญาณจะกลับไปอยู่ที่สถานะแบบว่างอีก เพื่อรอการส่งข้อมูลต่อไปจะเห็นว่าการสื่อสารแบบอะซิงโครนัสนี้ มีลักษณะเป็นไปทีละตัวอักษร และสัญญาณที่ส่งออกมา มีบางส่วนเป็นบิตเริ่มต้น บิตสิ้นสุด และบิตตรวจข้อผิดพลาด ทำให้ความเร็วในการส่งข้อมูลต่อวินาทีน้อยลงไป เนื่องจากต้อง สูญเสียช่องทางการสื่อสารให้กับ บิตเริ่มต้น บิตสิ้นสุด และบิตตรวจข้อผิดพลาด (ถ้ามีใช้) ตลอดเวลา การสื่อสาร แบบอะซิงโครนัสนี้มักใช้ในการติดต่อระหว่างคอมพิวเตอร์กับอุปกรณ์รอบข้าง

2.2.3 การสื่อสารแบบซิงโครนัส (Synchronous Transmission)



รูปที่ 2.3 แสดงบล็อกข้อมูลการสื่อสารแบบซิงโครนัส

การสื่อสารแบบซิงโครนัส จะทำการจัดกลุ่มของข้อมูลเป็นกลุ่มๆ และทำการส่งข้อมูลทั้งกลุ่มไปพร้อมกันในทีเดียว เราเรียกกลุ่มของข้อมูลนี้ว่า บล็อกของข้อมูล (Block of Data) ซึ่งตัวอักษรตัวแรก และตัวถัดไปที่อยู่ในบล็อกเดียวกันจะไม่มีอะไรมาคั่นเหมือนอย่างแบบอะซิงโครนัส ที่ต้องใช้บิตเริ่มต้น และบิตสิ้นสุดคั่นทุกๆ ตัวอักษร แต่จะมีข้อมูลเริ่มต้นซึ่งเป็นลักษณะของบิตพิเศษที่ส่งมาเพื่อให้รู้ว่านั้นคือจุดเริ่มต้นของกลุ่มตัวอักษรที่กำลังส่งเรียงกันเข้ามา เช่น อักขระซิงก์ (SYN character) โดยที่อักขระซิงก์มีรูปแบบบิต คือ 00010110 ตัวอย่างของการส่งแสดงได้ จากรูปที่ 2.3 เมื่อลายทางตรวจพบอักขระซิงก์ หรือ 00010110 แล้วจะทราบได้ทันทีว่าบิตที่ตามมาคือบิตตัวอักษรแต่ละตัว แต่การใช้อักขระซิงก์เพียงตัวเดียวอาจเกิดข้อผิดพลาดได้ เช่น ถ้าเราส่งตัวอักษร b และตัวอักษร a ติดต่อกันไป ซึ่งตัวอักษร b มีรูปแบบบิตคือ 01100010 และตัวอักษร a มีรูปแบบบิตคือ 01100001 การส่งจะแสดงได้

จะเห็นว่าเครื่องปลายทางจะตรวจพบอักขระซิงก์ระหว่างบิตของตัวอักษร b และตัวอักษร a ทำให้เข้าใจว่าบิตต่อไปจะเป็นบิตของกลุ่มข้อมูล ซึ่งจะทำให้การรับข้อมูลนั้นเกิดผิดพลาดขึ้นได้ ดังนั้นจึงแก้ปัญหาด้วยการใช้อักขระซิงก์ 2 ตัวต่อกันเป็นลักษณะของบิตพิเศษที่บอกให้ทราบว่าเป็นจุดเริ่มต้นบิตของกลุ่มข้อมูล ตัวอย่างของการใช้อักขระซิงก์ 2 ตัวในการสื่อสารแบบซิงโครนัส และการตัดแถวของบิตข้อมูลออกเป็นกลุ่มทีละ 8 บิตเพื่อแทนข้อมูลแสดงได้ การสื่อสารแบบซิงโครนัสนี้มักใช้ในการติดต่อระหว่างคอมพิวเตอร์

ประสิทธิภาพของการส่งผ่านข้อมูลแบบอะซิงโครนัส และแบบซิงโครนัส

การส่งผ่านข้อมูลแบบซิงโครนัสนั้นส่วนมากแล้ว ตลอดทางของสายส่งจะใช้ส่งผ่านข้อมูลเต็มตลอดทั้งสาย การส่งผ่านข้อมูลแบบอะซิงโครนัสนั้นสายส่งจะขาดความต่อเนื่องของสัญญาณข้อมูลที่ส่งผ่าน หรือถ้ามีสัญญาณข้อมูลที่ส่งผ่านต่อเนื่องกันเต็มตลอดทั้งสายแล้ว ก็จะสูญเสียช่องทางในการส่งไปกับการส่งบิตเริ่มต้นและบิตสิ้นสุดของแต่ละตัวอักษร

ตัวอย่างเช่น กรณีที่ส่งผ่านข้อมูลที่อยู่ในรูปของรหัส ASCII ซึ่งตัวอักษรหนึ่งตัวถูกแทนด้วย 8 บิต ถ้ามีการส่งกลุ่มของข้อมูล 240 ตัวอักษร ในกรณีการส่งผ่านข้อมูลแบบซิงโครนัสมีการใช้ตัวอักขระซิง 3 ตัว และการส่งผ่านข้อมูลแบบอะซิงโครนัสไม่มีการใช้บิตตรวจข้อผิดพลาด ดังนั้นเราสามารถคำนวณหาอัตราส่วนระหว่างการส่งข้อมูลได้ดังนี้ บิตทั้งหมดของตัวอักษรที่ส่งจะได้240 ตัวอักษร x 8 บิต/ตัวอักษร = 1920 บิต

แบบซิงโครนัส

บิตของตัวอักขระซิงที่ใช้จะได้SYN 3 ตัวเท่ากับ 3x8 บิต=24บิต ผลรวมของบิตที่ต้องส่งทั้งหมด = 1920 + 24 = 1944 บิตอัตราส่วนระหว่างการส่งข้อมูลที่ต้องส่งจริง กับจำนวนบิตทั้งหมดที่จำเป็นต้องส่งคือ1920 หารด้วย 1944 จะได้ประมาณ 99 %

แบบอะซิงโครนัส

บิตเริ่มต้น และบิตสิ้นสุดที่ใช้จะได้ 2 x 240 = 480 บิต ผลรวมของบิตที่ต้องส่งทั้งหมด = 1920 + 480 = 2400 บิต อัตราส่วนระหว่างการส่งข้อมูลที่ต้องส่งจริง กับจำนวนบิตทั้งหมดที่จำเป็นต้องส่งคือ1920 หารด้วย 2400 จะได้ประมาณ 80 %

2.2.4 การใช้บิตตรวจข้อผิดพลาด

บิตตรวจข้อผิดพลาด หรือพาริตี้บิต จะเป็นบิตที่ใช้เพื่อทำหน้าที่ตรวจสอบความถูกต้องของข้อมูลที่ส่ง ซึ่งมีอยู่ 2 แบบด้วยกันคือ การตรวจสอบจำนวนคี่ (odd parity) และการตรวจสอบจำนวนคู่ (even parity)

1. การตรวจสอบจำนวนคี่ (Odd parity) หมายถึง บิตตรวจสอบจะต้องนับบิตที่มีค่าของ 1 สำหรับกลุ่มของบิตที่จะส่งและต้องการตรวจสอบอยู่เป็นจำนวนคี่ เช่น ถ้านับบิตที่มีค่าของ 1 ในกลุ่มของบิตที่จะส่ง และต้องการ ตรวจสอบได้เป็นจำนวนคู่ บิตตรวจสอบนี้จะต้องมีค่าเป็น 1 เพื่อที่จะรวมเป็นจำนวนคี่ แต่ถ้าจำนวนนับได้เป็นจำนวนคี่ บิตตรวจสอบก็จะมีค่าเป็น 0

ตัวอย่างถ้าข้อมูลที่ต้องการส่งมี 7 บิต คือ 0110011 บิตตรวจสอบจำนวนคี่จะต้องมีค่าเป็น 1 เพราะนับบิตที่มีค่าของ 1 ได้เท่ากับ 4 ตัว ซึ่งเป็นเลขคู่ เมื่อรวมกับบิตตรวจสอบจำนวนคี่ที่มีค่าเป็น 1 ก็จะนับได้เป็น 5 ตัวซึ่งเป็นเลขคี่และการส่งข้อมูลพร้อมบิตตรวจสอบไปจะได้เป็น 10110011

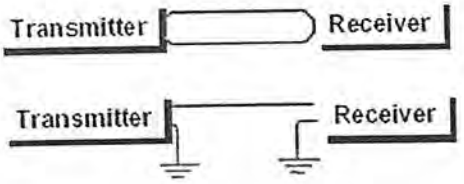
2. การตรวจสอบจำนวนคู่ (Even parity) หมายถึง บิตตรวจสอบจะต้องนับบิตที่มีค่าของ 1 สำหรับกลุ่มของบิตที่จะส่งและต้องการตรวจสอบอยู่เป็นจำนวนคู่ เช่น ถ้านับบิตที่มีค่าของ 1 ในกลุ่มของบิตที่จะส่งและต้องการ ตรวจสอบได้เป็นจำนวนคู่ บิตตรวจสอบนี้จะต้องมีค่าเป็น 0 เพื่อที่จะรวมเป็นจำนวนคู่ แต่ถ้าจำนวนนับได้เป็นจำนวนคี่บิตตรวจสอบก็จะมีค่าเป็น1

ตัวอย่างสมมุติว่าถ้าข้อมูลที่ต้องการส่งมี 7 บิต คือ 0110011 บิตตรวจสอบจำนวนคู่จะต้องมีค่าเป็น 0 เพราะนับบิตที่มีค่าของ 1 ได้เท่ากับ 4 ตัว ซึ่งเป็นเลขคู่ การส่งข้อมูลพร้อมบิตตรวจสอบไปจะได้เป็น 00110011 การตรวจสอบความถูกต้องทำได้โดย ระหว่างต้นทางและปลายทางจะต้องตกลงกันว่าจะใช้ตัวตรวจข้อผิดพลาดชนิดใด ถ้าใช้ตัวตรวจข้อผิดพลาดแบบจำนวนคี่แล้วเมื่อปลายทางรับข้อมูลจะตรวจสอบจำนวนบิตที่มีค่าเป็น 1 ว่าเป็นจำนวนคี่หรือไม่ ถ้าไม่เป็นจำนวนคี่แสดงว่าข้อมูลเกิดความผิดพลาดขึ้น ปลายทางจะต้องแจ้งให้ต้นทางทราบ อาจจะให้ต้นทางส่งข้อมูลมาใหม่อีกครั้ง ส่วนการใช้ตัวตรวจข้อผิดพลาดแบบจำนวนคู่ก็จะใช้หลักการคล้ายๆ กัน

2.2.5 ทิศทางของการสื่อสารข้อมูล

สามารถแบ่งทิศทางการสื่อสารของข้อมูลได้เป็น 3 แบบ คือ

1. แบบทิศทางเดียว (Simplex) เป็นทิศทางการสื่อสารข้อมูลแบบที่ข้อมูลจะถูกส่งจากทิศทางหนึ่งไปยังอีกทิศทางโดยไม่สามารถส่งข้อมูลย้อนกลับมาได้ เช่นระบบวิทยุ หรือโทรทัศน์

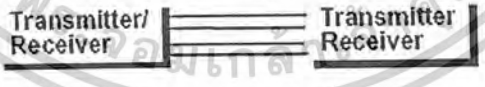


รูปที่ 2.4 แสดงการสื่อสารแบบทางเดียว

2. แบบกึ่งสองทิศทาง (Half Duplex) เป็นทิศทางการสื่อสารข้อมูลแบบที่ข้อมูลสามารถส่งกลับกันได้ 2 ทิศทาง แต่จะไม่สามารถส่งพร้อมกันได้ โดยต้องผลัดกันส่งครั้งละทิศทางเท่านั้น เช่น วิทยุสื่อสารแบบผลัดกันพูด

รูปที่ 2.5 แสดงการสื่อสารแบบฮาฟดูเพล็กซ์

3. แบบสองทิศทาง (Full Duplex) เป็นทิศทางการสื่อสารข้อมูลแบบที่ข้อมูลสามารถส่งพร้อม ๆ กันได้ทั้ง 2 ทิศทาง ในเวลาเดียวกัน เช่น ระบบโทรศัพท์



รูปที่ 2.6 แสดงการสื่อสารแบบฟูลดูเพล็กซ์

2.3 เอฟเอสเค(Frequency Shift Keying:FSK)

2.3.1การมอดูเลตแบบเอฟเอสเค

2.3.1.1 BFSK

ในระบบการมอดูเลตแบบ BFSK (Binary FSK) จะใช้สัญญาณ ไซนูซอยด์ 2 ชุดที่มีแอมพลิจูดเท่ากันแต่มีความถี่ต่างกันสำหรับแทนข้อมูลดิจิตอล 0 และ 1 กล่าวคือ

$$s_1(t) = \sqrt{\frac{2E_b}{T_b}}\cos(2\pi f_1 t) \qquad 0 \le t \le T_b \tag{2.1}$$

$$s_2(t) = \sqrt{\frac{2E_b}{T_b}}\cos(2\pi f_2 t) \qquad 0 \le t \le T_b \tag{2.2}$$

เมื่อพิจารณาสัญญาณทั้ง 2 ตัว เราสามารถเลือกฟังก์ชั่นเบซิส 2 ตัวที่เหมาะสมได้โดยกำหนดให้มีค่าดังนี้

$$\phi_1(t) = \sqrt{\frac{2}{T_b}}\cos(2\pi f_1 t) \qquad 0 \le t \le T_b \tag{2.3}$$

$$\phi_2(t) = \sqrt{\frac{2}{T_b}}\cos(2\pi f_2 t) \qquad 0 \le t \le T_b \tag{2.4}$$

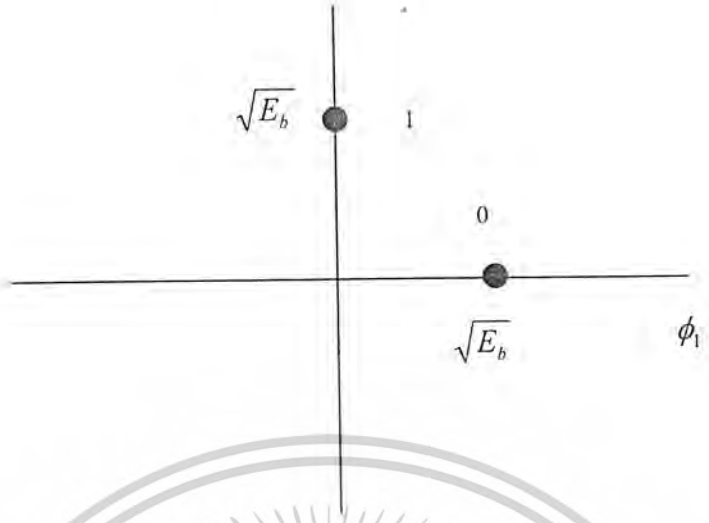
โดยสัมประสิทธิ์การคูณ s_{11}, s_{12}, s_{21}, และ s_{22} หาได้จากความสัมพันธ์ตามสมการดังนี้

$$s_{11} = \sqrt{E_b}\ ,\ s_{12} = 0, s_{21} = 0,\ s_{22} = \sqrt{E_b} \tag{2.5}$$

และถ้านำสัญญาณ $s_1(t), s_2(t), s_3(t), s_4(t)$ มาเขียนในรูปแบบของเวกเตอร์จะได้ผลดังนี้

$$s_1 = \begin{bmatrix} \sqrt{E_b} \\ 0 \end{bmatrix}, \quad s_2 = \begin{bmatrix} 0 \\ \sqrt{E_b} \end{bmatrix} \tag{2.6}$$

นำเวกเตอร์เหล่านี้ไปเขียนในระบบแกน ϕ_1 และ ϕ_2 จะได้จุดสัญญาณ 2 จุดบนซิกนัลสเปซ ดังแสดงในรูปที่ 2.7



รูปที่2.7 แผนภาพแสดงซิกนัลสเปซ ของระบบ BFSK

2.3.1.2 M-ary FSK

สำหรับกรณีทั่งไปการมอดูเลตแบบ FSK หรือที่เรียกว่า M-ary FSK จะใช้สัญญาณ ไซนูซอยด์ที่มีความถี่แตกต่างกัน ฉะนั้นจึงต้องมีฟังก์ชั่นเบซิสมากถึง M ตัว โดยแต่ละตัวมีความถี่ต่างกัน เพื่อใช้แทนข้อมูลดิจิตอลที่ต่างกัน M รูปแบบ ดังนี้

$$s_i(t) = \sqrt{\frac{2E_s}{T_s}}\cos(2\pi f_i t) \qquad 0 \le t \le T_s \tag{2.7}$$

โดย i = 1, 2, … , M และ E_s คือพลังงานของสัญญาณต่อหนึ่งสัญลักษณ์ที่ส่งออกในช่วงเวลา T_s วินาที

ฟังก์ชั่นเบซิสที่ใช้จะต้องมีจำนวนมากเท่ากับจำนวนสัญญาณ โดยกำหนดให้มีค่าดังนี้

$$\phi_i(t) = \sqrt{\frac{2}{T_s}}\cos(2\pi f_i t) \qquad 0 \le t \le T_s \tag{2.8}$$

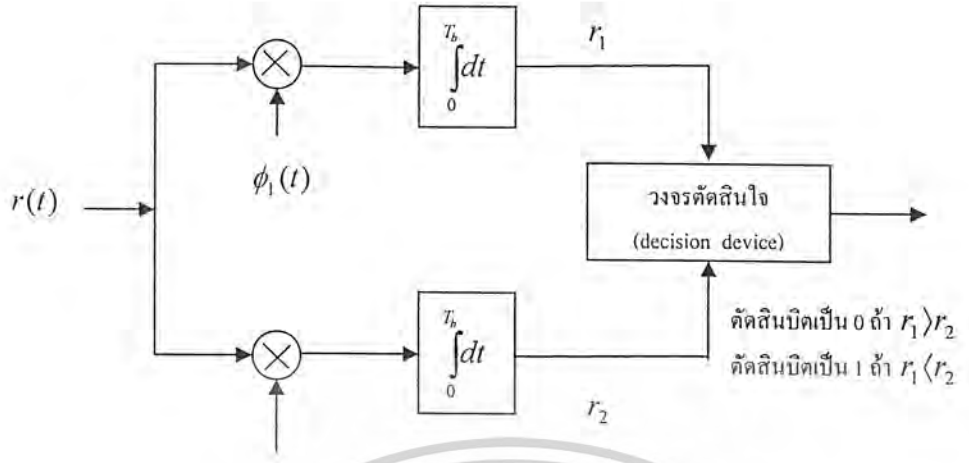
โดย i = 1, 2, … , M และถ้านำสัญญาณ $s_1(t), s_2(t), s_3(t), s_4(t)$ มาเขียนในรูปแบบของเวกเตอร์จะได้ผลดังนี้

$$s_1 = \begin{bmatrix} \sqrt{E_s} \\ 0 \\ 0 \\ 0 \\ . \\ . \\ . \\ 0 \end{bmatrix} \quad , \quad s_2 = \begin{bmatrix} 0 \\ \sqrt{E_s} \\ 0 \\ 0 \\ . \\ . \\ . \\ 0 \end{bmatrix} \quad , \quad s_3 = \begin{bmatrix} 0 \\ 0 \\ \sqrt{E_s} \\ 0 \\ . \\ . \\ . \\ 0 \end{bmatrix} \quad , \quad s_4 = \begin{bmatrix} 0 \\ 0 \\ . \\ . \\ . \\ 0 \\ 0 \\ \sqrt{E_s} \end{bmatrix}$$

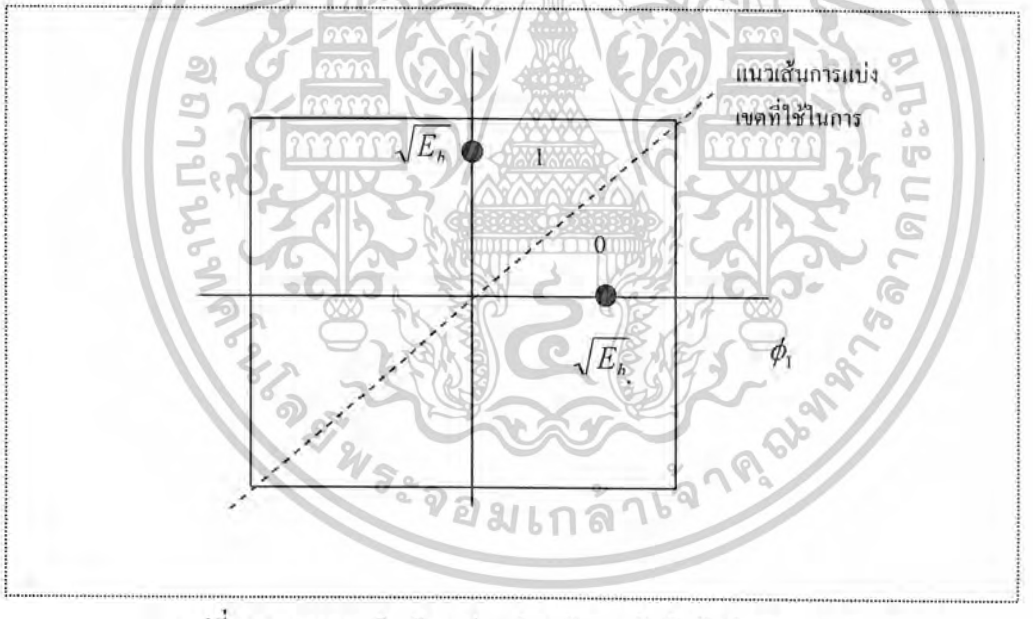
เนื่องจากการมอดูเลต M-ary FSK ต้องมีฟังก์ชั่นเบซิสจำนวนมาก ดังนั้นการแสดงในรูปของเวกเตอร์จึงทำได้ยากเพราะมีจำนวนแกนที่ต้องแสดงมากถึง M แกน อย่างไรก็ตามมีข้อสังเกตที่น่าสนใจคือจะมีจุดสัญญาณเพียง 1 จุดบนแกนแต่ละแกนเสมอ

2.3.2วงจรภาครับของ BFSK

มีลักษณะดังรูปที่ 2.8 จากรูปสัญญาณที่รับได้ r(t) ถูกนำไปคูณกับฟังก์ชั่นเบซิส 2 ชุดคือ $\phi_1(t)$ และ $\phi_2(t)$ โดยแยกจากกันเป็นสองส่วน ในแต่ละส่วนจะถูกนำไปผ่านวงจรอินทิเกรตจนครบคาบของเวลาการส่งหนึ่งบิตคือตั้งแต่เวลา 0 ถึง T_b ผลลัพธ์ที่ได้ประกอบด้วย r_1 และ r_2 จากนั้นนำค่าทั้งสองไปป้อนสู่วงจรตัดสินใจ เพื่อตัดสินใจว่าวงจรที่รับได้นั้นน่าจะเกิดจากการส่งข้อมูลดิจิตอล 0 หรือ 1 ในกระบวนการตัดสินใจบิตสามารถทำได้โดยการเปรียบเทียบขนาดของ r_1 และ r_2 หากพบว่า $r_1 \rangle r_2$ แล้วตัดสินใจว่าต้นทางส่งบิตที่มีค่าเป็น 0 แต่ถ้าพบว่า $r_1 \langle r_2$ แล้วให้ตัดสินใจว่าต้นทางส่งบิตที่มีค่าเป็น 1 สำหรับการกำหนดเกณฑ์ที่ใช้ในการตัดสินใจว่าบิตน่าจะเป็น 0 หรือ 1 นั้นสามารถพิจารณาและทำความเข้าใจได้จากแผนภาพซิกนัลสเปซ ของ BFSK ในรูปที่ 2.9 ในรูปได้มีการกำหนดเส้นทางแบ่งเขตพื้นที่ออกเป็น 2 ส่วนในแนวทแยง ถ้าจุดสัญญาณที่ได้รับได้ตกอยู่ในพื้นที่ซีกล่างก็จะเป็นจุดที่อยู่ใกล้จุดที่ $\phi_1 = \sqrt{E_b}$ มากกว่าจุดที่ $\phi_2 = \sqrt{E_b}$ ฉะนั้นให้ตัดสินใจว่าต้นทางได้ส่งสัญญาณ $s_1(t)$ มา(ข้อมูลดิจิตอลเป็น 0) หากจุดสัญญาณที่รับได้ตกอยู่ในพื้นที่ซีกบน ก็จะเป็นจุดที่อยู่ใกล้จุดที่ $\phi_2 = \sqrt{E_b}$ มากกว่าจุดที่ $\phi_1 = \sqrt{E_b}$ ฉะนั้นให้ตัดสินใจว่าต้นทางได้ส่งสัญญาณ $s_1(t)$ มา(ข้อมูลดิจิตอลเป็น 1)



รูปที่ 2.8 แผนภาพวงจรรับของระบบ BFSK



รูปที่ 2.9 แผนภาพซิกนัลสเปซ สำหรับการตัดสินใจบิตของระบบ BFSK

การตัดสินใจโดยอาศัยกฎเกณฑ์ดังกล่าวนี้สามารถก่อให้เกิดความผิดพลาดได้และมีความผิดพลาดเกิดขึ้นได้และมีความผิดพลาดเกิดขึ้นได้ 2 รูปแบบ สำหรับความผิดพลาดแบบที่หนึ่ง ต้นทางตั้งใจส่งข้อมูลทางดิจิตอลเท่ากับ 0 ออก จึงได้กำเนิดสัญญาณ $s_1(t)$ และส่งออก หากแต่ระดับสัญญาณรบกวนเกาส์มีขนาดสูงมากจนทำให้สัญญาณที่รับได้กลับไปตกสู่พื้นที่ซีกบน กรณีเช่นนี้ทำให้วงจรภาครับมีการตัดสินใจที่ผิดพลาด สำหรับความผิดพลาดแบบที่สอง ต้นทางตั้งใจส่งข้อมูลดิจิตอลเท่ากับ 1 ออกจึงได้ให้กำเนิดสัญญาณ $s_2(t)$ และส่งออก หากแต่ระดับสัญญาณรบกวนเกาส์มีขนาดสูงมากจนทำให้สัญญาณที่รับได้กลับไปตกอยู่ในพื้นที่ซีกล่าง ผลที่เกิดขึ้นคือวงจรภาครับมีการตัดสินใจที่ผิดพลาดเช่นเดียวกัน

2.3.3 การมอดูเลตสัญญาณดิจิตอลทางความถี่ (Frequency Shift Keying Modulation)

ตัวกำเนิดสัญญาณ FSK มีหลักการทำงานคือ เมื่อข้อมูลสัญญาณดิจิตอลที่เข้ามาเป็นแบบไบนารี่ ซึ่งมีการเปลี่ยนแปลงระดับลอจิกระหว่าง 1 กับ 0 จะทำให้ความถี่ของคลื่นพาห์มีการเปลี่ยนแปลงระหว่างสองความถี่ คือความถี่ที่ลอจิก "1" หรือความถี่มาร์ค (Mark Frequency, f_1) และความถี่ที่ลอจิก "0" หรือความถี่สเปซ (Space Frequency, f_0)

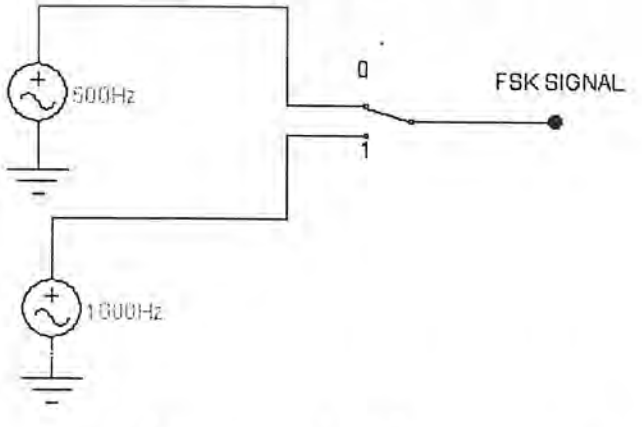
$e = A\sin 2\pi f_1 t$; เมื่อสถานะบิทเป็น "1"

$e = A\sin 2\pi f_0 t$; เมื่อสถานะบิทเป็น "0"

การเปลี่ยนแปลงความถี่แต่ละครั้งจะเกิดขึ้น เมื่อมีการเปลี่ยนแปลงของสถานะของลอจิกของสัญญาณข้อมูลทางด้านอินพุทจากลอจิก 1 เป็นลอจิก 0 (หรือในทางกลับกันคือการเปลี่ยนสถานะจากลอจิก0เป็นลอจิก1)นั่นคือการเปลี่ยนแปลงของสัญญาณทางด้านเอาท์พุทจะเท่ากับ อัตราการเปลี่ยนแปลงทาง ด้านอินพุท ในดิจิตอลมอดูเลชั่นนั้นอัตราการเปลี่ยนแปลงของสัญญาณด้านอินพุทจะเรียกว่า "อัตราบิต" (bit rate) มีหน่วยเป็นบิตต่อวินาที (bps) ส่วนอัตราการเปลี่ยนแปลงทางด้านเอาท์พุทจะเรียกว่า "อัตราบอด" (Baud rate) ดังนั้นการส่งการส่งข้อมูลด้วยวิธีการแบบ FSK อัตราบิตจะเท่ากับอัตราบอดเสมอ บิต rate จำนวนบิตข้อมูลดิจิตอลที่เป็น 0 หรือ 1ที่ส่งใน 1 วินาที ในแต่ละช่องสัญญาณ Baud rate เป็นหน่วยวัดจำนวนชุดข้อมูลที่ถูกส่งเข้าไปในสาย ซึ่งใน 1 ชุดอาจจะมีจำนวนกี่บิตก็ได้ ในบางครั้ง บิต rate อาจจะเท่ากับ Baud rate ก็ได้ เช่น ถ้าเป็นการมอดูเลตโดยที่ข้อมูลเป็นแบบไบนารี่ คือ 1 แทนข้อมูล 1 และ 0 แทนข้อมูล 0

การทำงานของตัวกำเนิดสัญญาณดิจิตอลเปลี่ยนแปลงความถี่มีหลักการทำงานดังนี้ โดยมีแหล่งกำเนิดสัญญาณหรือออสซิลเลเตอร์ (Oscillator) 2ชุด (f_1 และ f_0) และมีสัญญาณข้อมูลดิจิตอลแบบไบนารี่เข้ามาทางด้านอินพุทเพื่อควบคุมตัวกำเนิดสัญญาณเอฟเอสเค ซึ่งการทำงานของตัวกำเนิดสัญญาณดิจิตอลเปลี่ยนแปลงความถี่จะมีลักษณะคล้ายสวิตช์เลื่อนแบบ 2 ตำแหน่ง จะเลื่อนไปตามสัญญาณควบคุม ดังนั้นสัญญาณดิจิตอลที่เข้ามาเป็นสัญญาณที่มาควบคุมการเลื่อนตำแหน่งของสวิตช์ที่จะให้สัญญาณ f_1 หรือ f_0 ออกมาทางด้านเอาท์พุท

หลักการทำงานพื้นฐานของสัญญาณเอฟเอสเค



รูปที่ 2.10 แสดงการทำงานพื้นฐานของเอฟเอสเค

สัญญาณดิจิตอลเปลี่ยนแปลงความถี่ที่ได้จะมีสมการดังนี้คือ

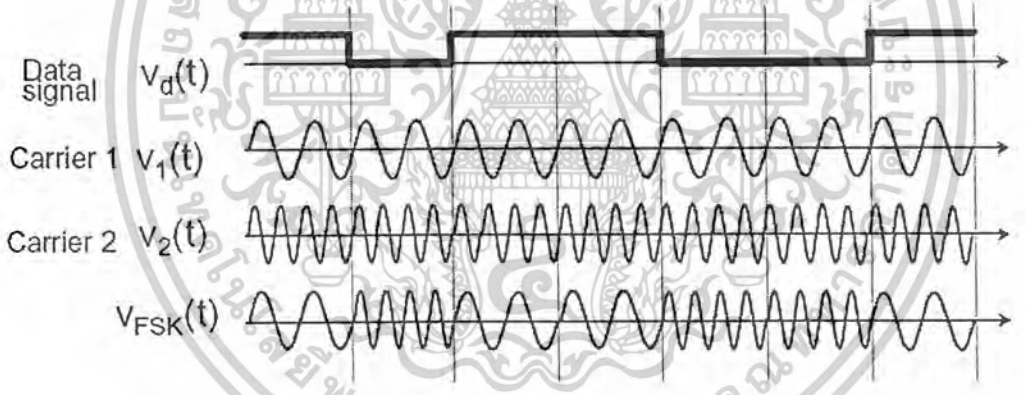
$$v(t) = A\cos(\omega_c \pm \Delta\omega)t \qquad (2.9)$$

โดย A = ขนาดของสัญญาณเอฟเอสเค

ω_c = ความถี่คลื่นพาห์

$\Delta\omega$ = ความถี่เบี่ยงเบน

เครื่องหมาย + และ – จะเปลี่ยนแปลงอย่างใดอย่างหนึ่งตามสัญญาณดิจิตอล และจะเห็นว่าสัญญาณดิจิตอลเปลี่ยนแปลงความถี่ จะมีขนาดของสัญญาณคงที่แต่มีความถี่ที่เปลี่ยนแปลงไปตาม $\omega_c+\Delta\omega$ หรือ $\omega_c-\Delta\omega$ สัญญาณดิจิตอลเปลี่ยนแปลงความถี่จะมีลักษณะเฟสที่ไม่ต่อเนื่อง (Phase Discontinuous) ตรงบริเวณนี้ตรงกับตำแหน่งการเปลี่ยนแปลงสถานะลอจิกของสัญญาณดิจิตอล เนื่องจากการใช้แหล่งกำเนิดของสัญญาณแยกกัน 2 ชุด (f_1 และ f_0) และสัญญาณดิจิตอลเปลี่ยนแปลงความถี่ที่ได้นี้จะมีรูปคลื่นมากกว่า 1 คาบเวลาต่อสัญญาณดิจิตอล 1 บิต



รูปที่ 2.11 แสดงสัญญาณเอฟเอสเค

สัญญาณรูปคลื่นดังแสดงดังรูปข้างบนแสดงถึงการเปลี่ยนแปลงความถี่ในกรณีที่มี Data input เป็น 1 สัญญาณ FSK output ที่ออกมาจะได้สัญญาณ Carrier 1 ในกรณีที่มี Data input เป็น 0 สัญญาณ FSK output ที่ออกมาจะได้สัญญาณ Carrier 2

2.3.4 การดีมอดูเลตสัญญาณดิจิตอลทางความถี่ (Frequency Shift Keying Demodulation)

วงจรที่ใช้โดยทั่วไปของการดีมอดูเลตไบนารี่เอฟเอสเค คือ วงจรเฟสล็อกลูป (Phase Lock Loop) ซึ่งระบบเฟสล็อกลูปเอฟเอสเคดีมอดูเลเตอร์ (PLL – FSK demodulator) สัญญาณอินพุทของเฟสล็อกลูปจะมีการเปลี่ยนแปลงระหว่างมาร์ค (mark) และสเปซ (space) ซึ่งค่าโวลท์เตจกระแส ตรงผิดพลาดที่เอาต์พุตซึ่งเกิดจากการเปรียบเทียบเฟสจากความถี่ที่ชิพไป เพราะที่อินพุทมีเพียงสองความถี่เท่านั้น จึงทำ

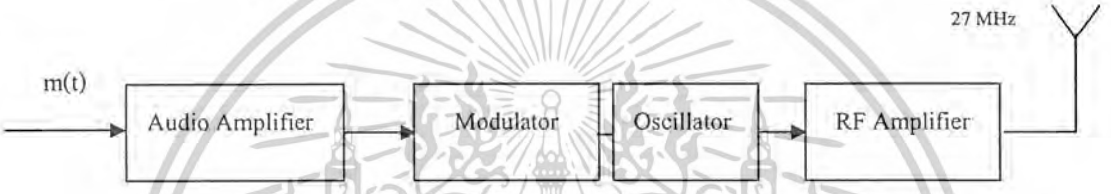
ให้ค่าโวลท์เตจผิดพลาดสองสถานะที่เอาต์พุตซึ่ง ก็คือ ลอจิก 1 และลอจิก 0 ดังนั้น เอาต์พุตของระบบเอฟเอสเคจะมีระดับตามเอฟเอสเคอินพุท ซึ่งโดยทั่วไปความถี่ของเฟส ล็อกลูปจะมีค่าเท่ากับความถี่กลางของเอฟเอสเคมอดูเลเตอร์

2.4 เอฟเอ็ม (FM)

2.4.1ระบบการรับและส่งเอฟเอ็ม

2.4.1.1 ภาคส่ง

ภาคส่งของเอฟเอ็มจะมีบล็อกไดอะแกรมดังรูปที่ 2.12 ซึ่งเป็นเครื่องส่งพื้นฐาน โดยจะมีอุปกรณ์วาแร็กเตอร์(Varactor) ซึ่งสามารถเปลี่ยนค่าความจุตามแรงดันย้อนกลับ ทำให้ความถี่ของออสซิลเลเตอร์เปลี่ยนแปลงได้เป็นสัญญาณเอฟเอ็มแล้วผ่านขยายอาร์เอฟ (RF) แล้วส่งออกไป



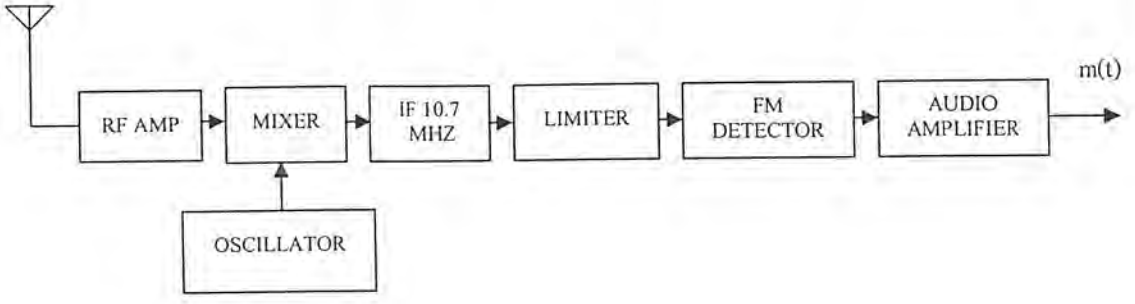
รูปที่ 2.12 แสดงบล็อกไดอะแกรมภาคส่งเอฟเอ็ม

2.4.1.2 ภาครับ

จากรูปที่ 2.13 เป็นภาครับที่ใช้วิธีซุปเปอร์เฮตเทอโรดายน์ ซึ่งประกอบด้วยวงจรขยายอาร์เอฟ(RF AMP), มิกเซอร์ (MIXER), วงจรขยายไอเอฟ (IF AMP), วงจรโลคอลออสซิลเลเตอร์ (Local Oscillator), ส่วนดีมอดูเลต และภาคขยายเสียง

จากรูปที่ 2.13 สัญญาณที่รับได้จากสายอากาศป้อนเข้าสู่วงจรขยายอาร์เอฟเพื่อทำการขยายสัญญาณให้มีขนาดแรงขึ้น แล้วส่งต่อมายังวงจรมิกเซอร์ จะทำหน้าที่ผสมสัญญาณจากวงจรขยายอาร์เอฟกับสัญญาณจากวงจรโลคอลออสซิลเลเตอร์ วงจรมิกเซอร์จะทำงานในช่วงที่ไม่ลิเนียร์ ทำให้ผลลัพธ์เป็นสัญญาณความถี่ผลรวมซึ่งตัดทิ้งไป ความถี่ผลต่างจะมีค่าเท่ากับความถี่ไอเอฟซึ่งมีค่า 10.7 MHz ป้อนเข้าสู่วงจรขยาย IF วงจรขยาย IF นี้เป็นวงจรขยายเลือกความถี่ที่มีค่าความถี่ศูนย์กลางคงที่ ไม่ว่าเราจะจูนวงจรขยาย RF เพื่อรับสัญญาณความถี่ใดก็ตาม สัญญาณจากวงจรโลคอลออสซิลเลเตอร์ที่เข้าไปผสมที่วงจรมิกเซอร์ จะต้องมีค่าพอดีและให้ผลลัพธ์ออกมามีค่าคงที่ซึ่งเท่ากับความถี่ ไอเอฟ = 10.7 MHz เสมอ

สัญญาณเอาท์พุทที่ออกมาจากวงจรขยายไอเอฟจะเหมือนกับสัญญาณที่รับได้ที่วงจรขยายอาร์เอฟต่างกันที่แต่ละความถี่จะลดลงจากความถี่อาร์เอฟเป็นความถี่ไอเอฟและหลังจากวงจรขยายไอเอฟก็จะเป็นวงจรลิมิตเตอร์ เพื่อจำกัดขนาดสัญญาณโดยที่ขนาดความถี่ยังเท่าเดิม จากนั้นก็จะผ่านไปยังวงจรดีมอดูเลตก็จะได้สัญญาณข่าวสารกลับคืนมา



รูปที่ 2.13 แสดงบล็อกไดอะแกรมของเครื่องรับเอฟเอ็ม

2.4.2 การมอดูเลตทางความถี่ (Frequency Modulation: FM)

การมอดูเลตทางความถี่เป็นกรรมวิธีการหนึ่งที่ฝากสัญญาณเสียงหรือสัญญาณเบสแบนด์ ซึ่งเป็นสัญญาณข้อมูลข่าวสาร(Information) ไปกับคลื่นพาห์ (Carrier wave) ซึ่งการมอดูเลตแบบนี้ความถี่ของคลื่นพาห์จะเปลี่ยนแปลงไปตามขนาดของสัญญาณเบสแบนด์ โดยจะเพิ่มขึ้นหรือลดลงตามขนาดของสัญญาณเบสแบนด์ที่นำมามอดูเลต แต่ขนาดของคลื่นพาห์ยังคงที่ การเปลี่ยนความถี่ของคลื่นพาห์จะมีความสัมพันธ์กับค่าแอมปลิจูดของสัญญาณข้อมูล โดยสัญญาณข่าวสารจะแสดงอยู่ในสมการ

$$m(t) = A_m \cos(2\pi f_m t) \tag{2.10}$$

โดยที่ $m(t)$ คือ สัญญาณข่าวสาร

A_m คือ ขนาดของสัญญาณข่าวสาร

สมการจะเป็นสมการแบบไม่เชิงเส้น(Nonlinear Modulation Process) และสเปกตรัมของสัญญาณเอฟเอ็มก็จะไม่มีความสัมพันธ์กับสัญญาณข่าวสารแต่อย่างใดในการศึกษาระบบเอฟเอ็มจะต้องศึกษาในกรณีที่ข่าวสารเป็นสัญญาณที่มีความถี่ค่าเดียว (Single-Tone Modulation, Single-Tone Frequency)

พิจารณาสัญญาณข่าวสารที่กำหนดอยู่ในรูปสมการ

$$m(t) = A_m \cos(2\pi f_m t)$$

ความถี่ที่เวลา t ใดๆของสัญญาณเอฟเอ็มจะอยู่ในรูปสมการ

$$f_i(t) = f_c + k_f A_m \cos(2\pi f_m t) \tag{2.11}$$

$$= f_c + \Delta f \cos(2\pi f_m t) \tag{2.12}$$

โดยที่ $f_i(t)$ คือ ความถี่ที่เวลา t ใดๆ

f_c คือ ความถี่กลาง

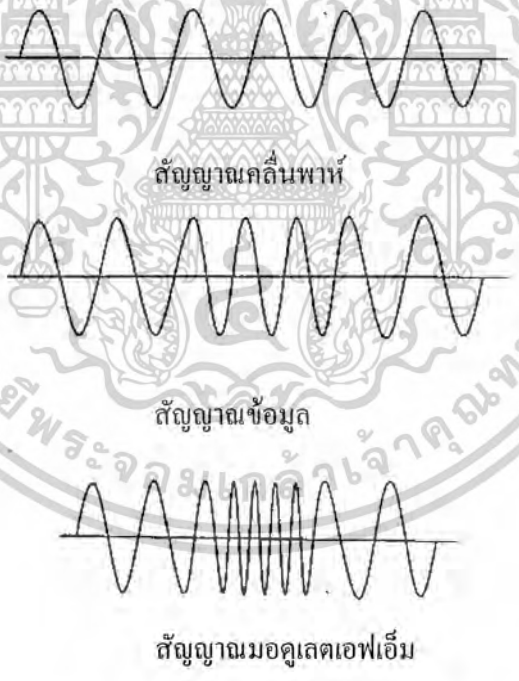
$\Delta f = k_f A_m$ เทอม Δf คือความถี่เบี่ยงเบน(Frequency Deviation) เป็นค่าที่บ่งบอกถึงความถี่ที่มากที่สุดที่เบี่ยงเบนไปจากคลื่นพาห์ ลักษณะพื้นฐานของสัญญาณเอฟเอ็มคือ Δf จะแปรตามขนาดของสัญญาณข่าวสารแต่จะไม่แปรตามความถี่ของสัญญาณข่าวสารและจากสมการ $f_i(t) = f_c + k_f A_m \cos(2\pi f_m t)$ อาร์กิวเมนต์ θ (t) หาได้จาก

$$\theta(t) = 2\pi \int_0^t f_i(t)dt \tag{2.13}$$

$$= 2\pi f_c t + \frac{\Delta f}{f_m}\sin(2\pi f_m t) \tag{2.14}$$

อัตราส่วนของ $\frac{\Delta f}{f_m}$ เรียกว่า ดัชนีการมอดูเลต (Modulation Index, β)

ดังนั้นสัญญาณที่ได้นี้จะเป็นสัญญาณเอฟเอ็มเพราะความถี่ที่เปลี่ยนไปจากความถี่คลื่นพาห์แปลตรงกับสัญญาณ m (t) ที่เข้ามามอดูเลต



รูปที่ 2.14 แสดงสัญญาณคลื่นพาห์สัญญาณข้อมูลและสัญญาณมอดูเลตแบบFM

2.4.3 การสร้างสัญญาณ FM

การสร้างสัญญาณ FM นั้นสามารถแบ่งได้กว้างๆ เป็น 2 วิธี คือวิธีสร้างโดยตรง (direct method) และวิธีสร้างโดยอ้อม (indirect method) สำหรับวิธีแรกนั้นจะทำการแปรเปลี่ยนความถี่ของสัญญาณคลื่นพาห์ไปตามขนาดของสัญญาณเบสแบนด์โดยตรงเลย ส่วนวิธีที่สองจะต้องสร้างสัญญาณ FM แถบ

ความถี่แคบ (narrowband FM) ขึ้นมาก่อน แล้วจากนั้นก็อาศัยวงจรคูณความถี่เพื่อเพิ่มขนาดค่าความถี่เบี่ยงเบน (frequency deviation) และความถี่ของคลื่นพาห์ให้ได้ตามที่ต้องการ

วิธีสร้างโดยตรง

สำหรับวิธีการสร้างโดยตรงนั้นอาศัยอุปกรณ์ที่เรียกว่า Voltage-controlled oscillators (VCO) ในการสร้างสัญญาณคลื่นพาห์ที่มีความถี่แปรเปลี่ยนไปตามขนาดของสัญญาณเบสแบนด์ วงจรดังกล่าวสร้างจากวงจรออสซิลเลเตอร์ที่มีองค์ประกอบส่วนหนึ่งเป็นอุปกรณ์ที่เรียกว่า วาแรกเตอร์ โดยอุปกรณ์ชิ้นนี้มีคุณสมบัติที่สำคัญคือค่าความจุไฟฟ้าเปลี่ยนแปลงได้ตามขนาดของแรงดันไฟฟ้าที่คร่อมอยู่ ดังนั้นหากต่อสัญญาณเบสแบนด์ที่จะส่งตรงที่ขั้วทั้งสองของวาแรกเตอร์ ค่าความจุไฟฟ้าก็จะแปรเปลี่ยนไปตามขนาดของสัญญาณเบสแบนด์ ซึ่งส่งผลให้วงจรออสซิลเลเตอร์กำเนิดความถี่ที่เปลี่ยนแปลงไปตาม m(t) ด้วย การสร้างสัญญาณ FM โดยตรงในลักษณะนี้เป็นวิธีการที่ค่อนข้างง่าย และค่าเบี่ยงเบนความถี่ที่ได้ก็จะมีค่าค่อนข้างสูงพอควร หากแต่ปัญหาหลักของวิธีแบบนี้คือ ความถี่กลางของคลื่นพาห์มีโอกาสคลาดเคลื่อนจากค่าจริงได้ง่าย ดังนั้นเพื่อขจัดปัญหาดังกล่าวจึงต้องมีการนำเอาวงจรเฟสล็อกลูป (phase locked loop) มาใช้ช่วยในการล็อกความถี่ของคลื่นพาห์ให้มีค่าคงที่

วิธีสร้างโดยอ้อม

ขั้นตอนในการสร้างสัญญาณ FM โดยอ้อมนั้นจะเริ่มต้นด้วยการอินทิเกรตสัญญาณเบสแบนด์ m(t) จากนั้นนำสัญญาณดังกล่าวผ่านเข้าวงจรเฟสมอดูเลเตอร์แถบความถี่แคบ เพื่อให้ได้สัญญาณ FM แถบความถี่แคบ เนื่องจากค่าเบี่ยงเบนความถี่สูงสุดของสัญญาณ FM ที่ได้จากขั้นตอนนี้มีค่าที่ต่ำมาก ด้วยเหตุนี้สัญญาณ FM ที่สร้างได้จึงเป็นสัญญาณ FM แถบความถี่แคบเท่านั้น

2.4.4 แบนด์วิดธ์ของสัญญาณเอฟเอ็ม

ในระบบเอฟเอ็มจำนวนไซด์แบนด์และแอมปลิจูดของไซด์แบนด์ขึ้นอยู่กับ ค่าดัชนีการมอดูเลต โดยความถี่ของไซด์แบนด์มีค่าความสัมพันธ์กับความถี่ของสัญญาณที่มอดูเลต กล่าวคือไซด์แบนด์คู่แรกมีความสัมพันธ์เท่ากับ $f_c \pm f_m$ ไซด์แบนด์คู่ที่สองมีความสัมพันธ์เท่ากับ $f_c \pm 2f_m$ ...ฯลฯ ฉะนั้นแบนด์วิดธ์ของคลื่นเอฟเอ็มต้องครอบคลุมจำนวนไซด์แบนด์ที่สำคัญทุกตัว นั่นคือแบนด์วิดธ์ขึ้นอยู่กับค่าดัชนีการมอดูเลตและความถี่ของสัญญาณที่มอดูเลตแต่ดัชนีการมอดูเลตเท่ากับอัตราส่วน ระหว่างความเบี่ยงเบน (Δf) ต่อความถี่ของสัญญาณที่เข้ามามอดูเลต (f_m) ดังนั้น ถ้าทราบความถี่เบี่ยงเบนและความถี่ของสัญญาณมอดูเลตก็สามารถคำนวณหาแบนด์วิดธ์ได้จากสูตร

$$BW = f_m \times \text{จำนวนไซด์แบนด์} \times 2 \tag{2.15}$$

$$BW = 2(\beta + 1) f_{m\max} \tag{2.16}$$

การมอดูเลตทางความถี่จะใช้วงจรที่เครื่องส่งที่ประกอบด้วยวงจร LC ออสซิลเลเตอร์ทำหน้าที่กำเนิดคลื่นพาห์ วงจรขยายสัญญาณให้มีกำลังส่งเพิ่มขึ้น และวงจรรีแอคแตนซ์มอดูเลเตอร์ที่ทำหน้าที่มอดูเลตสัญญาณเอฟเอ็ม ซึ่งวงจรมอดูเลตนี้จะอาศัยการเปลี่ยนแปลงค่ารีแอคแตนซ์ (Variable Reactance) ทำให้ความถี่คลื่นพาห์เปลี่ยนแปลงตามแอมปลิจูดของสัญญาณข้อมูล โดยอาศัยพื้นฐานของวงจรขยาย

สัญญาณให้ทำงานในช่วงที่ไม่เป็นเชิงเส้น เมื่อวงจรถูกขับให้ทำงาน ในช่วงที่ไม่เป็นเชิงเส้น ทำให้เกิดการเปลี่ยนแปลงจุดทำงาน (Operating point) ของทรานซิสเตอร์ ส่งผลให้เกิดการมอดูเลตสัญญาณระหว่างความถี่คลื่นพาห์กับสัญญาณข้อมูล

2.4.5 การดีมอดูเลตทางความถี่ (Frequency Demodulation)

ในการดีมอดูเลตสัญญาณ FM เพื่อดึงสัญญาณเบสแบนด์กลับมานั้นสามารถกระทำได้หลายวิธี เช่น ดีเทกเตอร์แบบหาความชัน (Slope detector) ซีโรครอสซิงดีเทกเตอร์ (zero-crossing detector) เฟสล็อกลูป (phase locked loop) และ quadrature detection

2.4.5.1 ดีแทกเตอร์แบบหาความชัน

การดีมอดูเลตสัญญาณเอฟเอ็มของวิธีนี้อาศัยการหาอนุพันธ์ของสัญญาณเอฟเอ็ม หรือก็คือการหาความชันของสัญญาณนั่นเอง และจากนั้นก็นำสัญญาณที่ได้ไปผ่านวงจรเอนเวโลปดีเทกเตอร์เพื่อดึงสัญญาณเบสแบนด์กลับคืนมา ดูขั้นตอนการดีมอดูเลตในรูปที่ 2.15 เริ่มแรกสัญญาณเอฟเอ็ม ถูกส่งเข้าวงจรลิมิตเตอร์เพื่อขจัดปัญหาความไม่แน่นอนในการลดทอนของสัญญาณที่ไม่เท่ากันในขณะที่ส่งผ่านช่องสัญญาณที่อาจมีปัญหามาจากการเฟดดิงในช่องสัญญาณสื่อสาร ดังนั้นสัญญาณที่ได้ก็จะมีเอนเวโลปที่คงที่ตลอด สังเกตว่าสัญญาณที่ได้ประกอบด้วยสัญญาณกระแสตรงที่มีขนาดแปรเปลี่ยนตามความถี่ของสัญญาณคลื่นพาห์ และให้สัญญาณ FM ที่ได้เป็น

$$s_1(t) = A_c \cos[2\pi f_c t + 2\pi k_f \int_{-\infty}^{t} m(\eta)d\eta\} = A_c \cos[2\pi f_c t + \theta(t)] \quad (2.17)$$

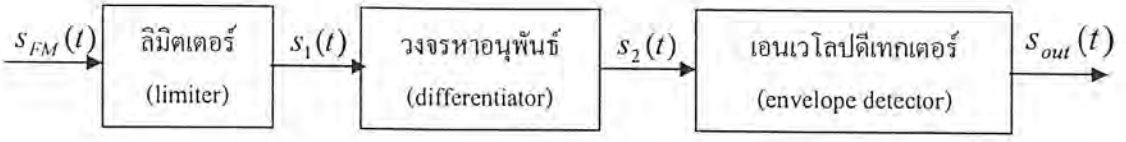
เมื่อสัญญาณนี้ผ่านเข้าวงจรหาอนุพันธ์เพื่อคำนวณความชันของสัญญาณ ผลที่ได้คือสัญญาณ

$$s_2(t) = -A_c[2\pi f_c + \frac{d\theta(t)}{d(t)}]\sin[2\pi f_c t + \theta(t) \quad (2.18)$$

และเมื่อนำ $s_2(t)$ ผ่านเข้าวงจรเอนเวโลปทีเทกเตอร์ก็จะได้สัญญาณ

$$s_{out}(t) = A_c[2\pi f_c + \frac{d\theta(t)}{dt}] = A_c 2\pi f_c + A_c 2\pi k_f m(t) \quad (2.19)$$

สังเกตว่าสัญญาณที่ได้ประกอบด้วยสัญญาณกระแสตรงที่มีขนาดแปรตามค่าความถี่ของสัญญาณคลื่นพาห์ รวมอยู่กับสัญญาณแบสแบนด์ $m(t)$ ที่ต้องการ



รูปที่ 2.15 วงจรดีมอดูเลตสัญญาณเอฟเอ็มแบบหาความชัน

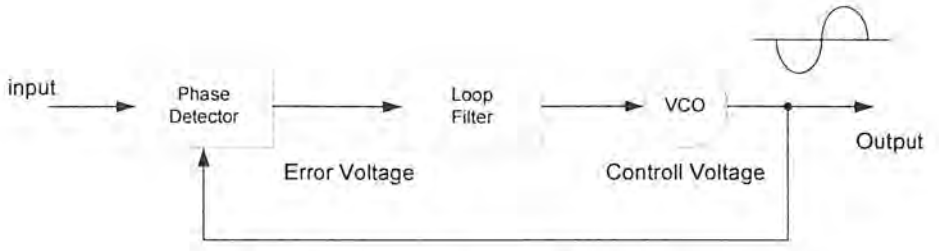
2.4.5.2 ซีโรกรอสซิงดีเทกเตอร์

วิธีการมอดูเลตแบบต่อไปนี้มีชื่อเรียกว่า ซีโรกรอสซิงดีเทกเตอร์ โดยอาศัยการแปลงความถี่ของสัญญาณให้กลายเป็นพัลส์ที่มีจำนวนเท่าๆกับจำนวนครั้งที่สัญญาณตัดผ่านศูนย์ จากนั้นนำสัญญาณกลายพัลส์นี้ไปหาค่าเฉลี่ยเพื่อแปลงให้กลายเป็นแอมพลิจูดอีกทอดหนึ่ง ดูขั้นตอนการทำงานในรูปที่ 2.16 และตัวอย่างรูปที่ได้ในแต่ละขั้นในรูปที่ 2.16 ขั้นตอนแรกสัญญาณเอฟเอ็ม จะถูกส่งผ่านเข้าลิมิตเตอร์เพื่อให้ได้สัญญาณพัลส์ $s_1(t)$ ที่มีความกว้างที่ขึ้นอยู่กับความถี่ของสัญญาณขณะนั้นๆ จากนั้นสัญญาณที่ได้จะถูกป้อนเข้าวงจรอนุพันธ์เพื่อให้ได้สัญญาณอิมพัลส์ $s_2(t)$ สำหรับนำไปใช้ในการควบคุมการสร้างสัญญาณพัลส์ที่มีคาบเวลาคงที่ $s_3(t)$ โดยอาศัยวงจรมัลติไวเบรเตอร์แบบเอกเสถียร (monostable multivibrator) สัญญาณนี้เมื่อนำเมื่อนำไปผ่านวงจรกรองผ่านต่ำซึ่งทำหน้าที่เหมือนการหาค่าเฉลี่ยของสัญญาณ ผลที่ได้ก็คือสัญญาณเบสแบนด์ที่ต้องการ สังเกตว่าเมื่อใดที่สัญญาณเอฟเอ็มมีความถี่สูงคือมีการตัดผ่านศูนย์บ่อยครั้ง จำนวนพัลส์รูปสี่เหลี่ยมก็จะมีจำนวนความถี่มากค่าเฉลี่ยของสัญญาณก็จะสูงตาม ในทางกลับกัน เมื่อใดที่สัญญาณเอฟเอ็มมีความถี่ต่ำ มีการตัดผ่านศูนย์น้อยจำนวนพัลส์รูปสี่เหลี่ยมก็จะมีจำนวนน้อยตามไปด้วย ซึ่งเมื่อหาค่าเฉลี่ยของสัญญาณแล้วก็จะได้ค่าที่ต่ำ จากที่ได้กล่าวมาทั้งหมดจะเห็นว่าเราสามารถแปลงความถี่ให้กลายเป็นแอมพลิจูดได้ซึ่งก็คือการดีมอดูเลตสัญญาณเอฟเอ็ม

รูปที่ 2.16 ขั้นตอนการดีมอดูเลตสัญญาณเอฟเอ็มโดยวิธีซีโรกรอสซิงดีเทกเตอร์

2.4.5.3 เฟสล็อกลูป

เฟสล็อคลูป (Phase Locked Loop) เป็นระบบป้อนกลับที่บังคับให้วงจรออสซิลเลเตอร์มีความถี่หรือเฟสเปลี่ยนแปลงไปตามความถี่หรือเฟสของสัญญาณอ้างอิงภายนอก เฟสล็อคลูปประกอบด้วยภาคสำคัญ 3 ภาค คือ ภาคเทียบเฟสหรือเฟสดีเทคเตอร์ (Phase Detector) ภาคลูปฟิลเตอร์ (Loop filter) และภาค VCO ดังแสดงในรูปที่ 2.17 ในที่นี้สมมุติว่าเราต่อเอาต์พุตจากวงจร VCO



รูปที่ 2.17 แสดงแผนผังของเฟสล็อคลูป

สมมุติว่ามีสัญญาณความถี่อ้างอิงภายนอกเป็นสัญญาณรายคาบ (Periodic) เข้ามาที่อินพุต ภาคเทียบเฟสทำหน้าที่เปรียบเทียบเฟสระหว่างสัญญาณอ้างอิงกับสัญญาณจาก VCO เอาต์พุตที่ได้จากภาคเฟสดีเทคเตอร์จะเป็นแรงดันที่มีแอมปลิจูดเป็นสัดส่วนกับผลต่างในเฟสของสัญญาณทั้งสองที่ทำการเปรียบเทียบ แรงดันผลต่างนี้จะป้อนไปยังวงจรลูปฟิลเตอร์ซึ่งเป็นฟิลเตอร์ชนิดโลว์พาสกรองเอาแต่เฉพาะความถี่ต่าง ๆ ตามที่ต้องการ เพื่อส่งไปควบคุมการออสซิลเลทของ VCO ต่อไป

เมื่อลูปอยู่ในสภาวะล็อค (Lock)ความถี่ของ VCO จะเท่ากับความถี่ของสัญญาณที่อินพุตพอดี อาจจะมีเฟสที่แตกต่างกันออกไป แต่เฟสที่แตกต่างกันออกไปนั้นจะมีค่าคงที่ (Constant phase difference) ในกรณีที่มีเฟสไม่ตรงกับภาคเฟสดีเทคเตอร์จะจ่ายแรงดันคลาดเคลื่อน (Error voltage) ไปควบคุมการทำงานของ VCO เพื่อมิให้เฟสคลาดเคลื่อนจนกว่าจะเข้าสู่สภาวะล็อคเอาต์พุตของ VCO จะมีแอมปลิจูดคงที่เสมอ แต่ความถี่จะเปลี่ยนแปลงไปตามสัญญาณที่อินพุต เฟสล็อคลูปยังมีประโยชน์ เช่น ในการดีมอดสัญญาณ FM (หรือ PM) เนื่องจากเอาต์พุตของเฟสดีเทคเตอร์มีค่าความสัมพันธ์กับการเปลี่ยนเฟสของคลื่นพาหะ

พื้นฐานของเฟสล็อคลูป

เฟสล็อคลูป ได้ถูกคิดค้นขึ้นมาครั้งแรกตั้งแต่ก่อนปี ค.ศ. 1932 โดยวิศวกรชาวฝรั่งเศส De Bellescize ซึ่งเขาได้รับการยกย่องว่าผู้คิดค้น "Coherent Communication" เฟสล็อคลูปถูกนำมาใช้อย่างแพร่หลายในทางอุตสาหกรรมเมื่อมันถูกผลิตมาในรูปแบบของวงจรรวมแล้ว ไอซีเฟสล็อคลูปตัวแรกเกิดขึ้นราวๆ ปี ค.ศ. 1965 เป็นอุปกรณ์อนาลอกล้วน ๆ โดยใช้ Analog multiplier (Four-quadrant multiplier) เป็นเฟสดีเทคเตอร์ และลูปฟิลเตอร์ใช้วงจรกรองความถี่ RC แบบพาสซีพหรือแอกทีฟและใช้ออสซิลเลเตอร์ที่ควบคุมด้วยแรงดัน (VCO) ในการผลิตความถี่เอาต์พุตของเฟสล็อคลูป ทุกวันนี้เฟสล็อคลูปชนิดนี้เรียกว่า ลิเนียร์เฟสล็อคลูป (LPLL) หลายปีผ่านไปเฟสล็อคลูปไม่มีการเปลี่ยนแปลงอะไรมากนัก แต่ก็เริ่มเข้าสู่ส่วนที่เป็นดิจิตอล ดิจิตอลเฟสล็อคลูปตัวแรกเกิดขึ้นราวปี ค.ศ. 1970 ซึ่งเป็นอุปกรณ์ผสม (hybrid device) นั่นคือ เฉพาะส่วนที่เป็นเฟสดีเทคเตอร์เท่านั้นที่สร้างจากวงจรดิจิตอล ซึ่งสร้างมาจากเอ็กซ์คลูซีฟออร์เกท หรือ เจเคฟลิปฟลอป ส่วนที่เหลือยังคงเป็นวงจรอนาลอกเฟสล็อคลูปชนิดนี้เรียกว่า คลาสสิคดอลดิจิตอลเฟสล็อคลูป (DPLL) ไม่กี่ปีต่อมาเฟสล็อคลูป ที่เป็นดิจิตอลทั้งหมดได้ถูกสร้างขึ้น (ADPLL) จะไม่มีตัวต้านทานหรือตัวเก็บประจุอยู่ภายในเลย เฟสล็อคลูปจะไม่เกี่ยวข้องกับฮาร์ดแวร์ แต่

จะเป็นโปรแกรมคอมพิวเตอร์ เฟสล็อคลูปชนิดนี้เรียกว่า ซอฟต์แวร์เฟสล็อคลูป (SPLL) ในปัจจุบัน DPLL นิยมใช้กันมากเนื่องจากนำมาประยุกต์ใช้งานได้โดยทั่วไปมีราคาถูก

เฟสล็อคลูป คือวงจรที่จะทำการซิงโครไนซ์ (synchronize) ความถี่หรือเฟสของสัญญาณเอาต์พุต (ที่ผลิตโดยออสซิลเลเตอร์) กับสัญญาณอ้างอิงหรือสัญญาณอินพุต เมื่อซิงโครไนซ์กันแล้วซึ่งมักจะเรียกกันว่าอยู่ในสภาวะล็อค (Lock State) ความแตกต่างระหว่างเฟสของสัญญาณเอาต์พุตของออสซิลเลเตอร์กับสัญญาณอ้างอิงจะมีค่าเท่ากับศูนย์หรือน้อยมาก

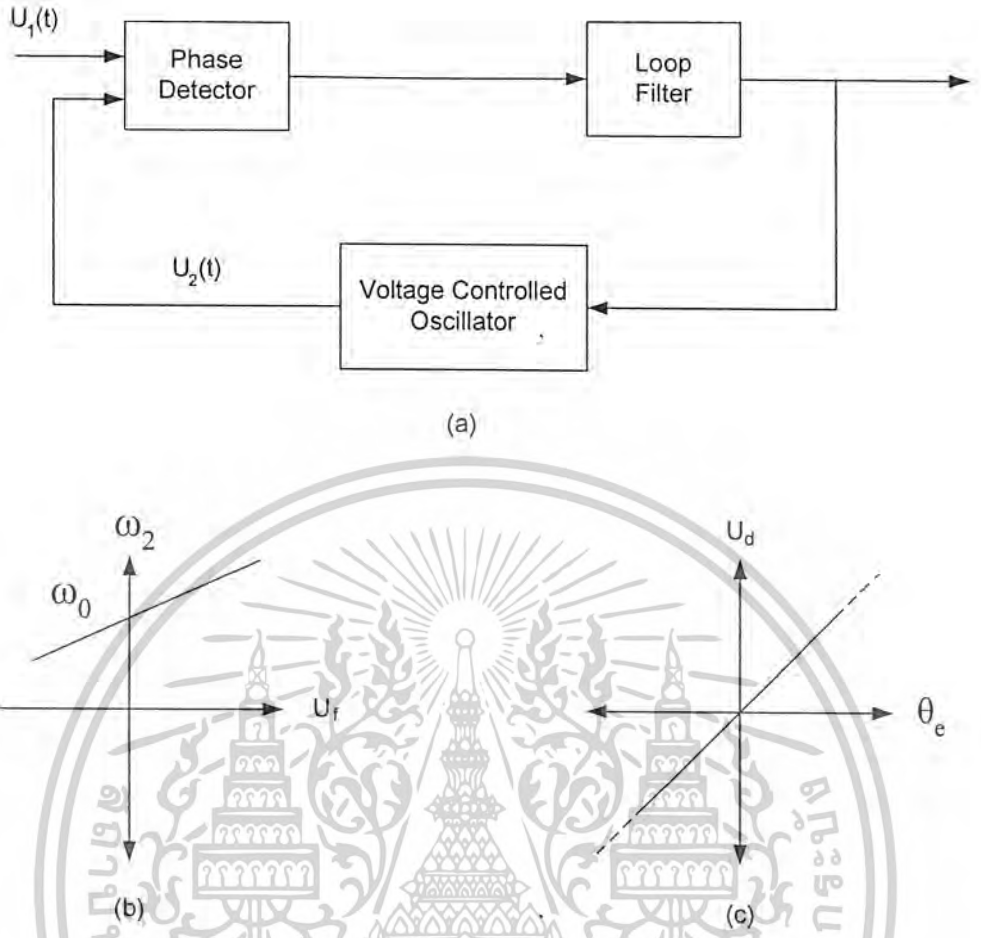
บล็อกไดอะแกรมของเฟสล็อคลูป แสดงได้ดังรูปที่ 2.18(a) ซึ่งประกอบด้วยบล็อกการทำงานพื้นฐานดังนี้

1. ออกซิลเลเตอร์ที่ควบคุมด้วยแรงดัน (Voltage Control Oscillator :VCO)
2. เฟสดีเทคเตอร์ (Phase Detector :PD)
3. ลูปฟิลเตอร์ (Loop Filter :LP)

วงจรเฟสล็อคลูปบางวงจรจะใช้ออสซิลเลเตอร์ที่ควบคุมด้วยกระแสแทนที่ VCO ในกรณีนี้ เฟสดีเทคเตอร์จะประพฤติตัวเป็นแหล่งจ่ายกระแสแทนที่จะเป็นแหล่งจ่ายแรงดันแต่อย่างไรก็ตามหลักการทำงานก็ยังคงเหมือนกัน

ก่อนที่จะกล่าวถึงหลักการทำงานของ PLL จะขอกำหนดสัญญาณต่าง ๆ ในเฟสล็อคลูปดังต่อไปนี้

- $U_1(t)$ คือสัญญาณอ้างอิง (หรือสัญญาณอินพุท)
- ω_1 คือความถี่เชิงมุมของสัญญาณอ้างอิง
- $U_2(t)$ คือสัญญาณเอาท์พุทของ VCO
- ω_2 คือความถี่ของสัญญาณเอาท์พุท
- $U_d(t)$ คือสัญญาณเอาท์พุทของเฟสดีเทคเตอร์
- $U_f(t)$ คือสัญาณเอาท์พุทของ ลูปฟิลเตอร์
- θ_e คือความต่างเฟส (phase error) ระหว่างสัญญาณ $U_1(t)$ กับสัญญาณ $U_2(t)$



รูปที่ 2.18 (a) บล็อกไดอะแกรมของเฟสล็อคลูป
(b) ทรานสเฟอร์ฟังก์ชั่นของ VCO
(c) ทรานสเฟอร์ฟังก์ชั่นของเฟสดีเทคเตอร์

จากบล็อกไดอะแกรมของเฟสล็อคลูป VCO จะผลิตความถี่เชิงมุม ω_2 ซึ่งกำหนดโดยสัญญาณเอาต์พุตของลูปฟิลเตอร์ หรือเขียนเป็นสมการได้เป็น

$$\omega_2(t) = \omega_0 + K_0 U_f(t) \tag{2.20}$$

เมื่อ ω_2 คือความถี่เชิงมุมกลางของ VCO (VCO Gain) มีหน่วยเป็น $S^{-1} V^{-1}$ สามารถพล็อตเป็นกราฟได้ดังรูปที่ 2.18 (b)

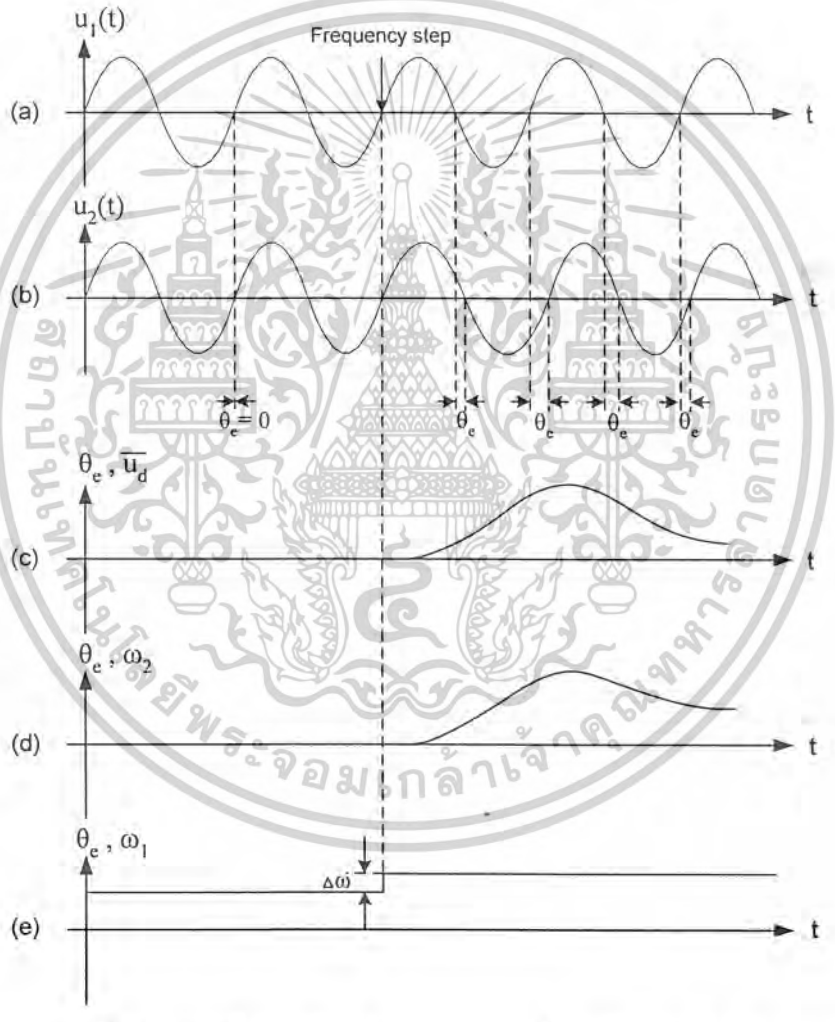
เฟสดีเทคเตอร์ (หรืออาจเรียกว่าเฟสคอมพาราเตอร์) จะทำหน้าที่เปรียบเทียบเฟสของสัญญาณเอาต์พุตกับเฟสของสัญญาณอินพุตและทำให้เกิดสัญญาณเอาต์พุต $U_d(t)$ ซึ่งเป็นสัดส่วนโดยประมาณกับความต่างเฟส θ_e เขียนสมการได้เป็น

$$U_d(t) = K_d \theta_e \tag{2.21}$$

เมื่อ K_d คือเกนของเฟสดีเทคเตอร์ มีหน่วยเป็น $V.rad^{-1}$

สัญญาณเอาต์พุตของเฟสดีเทคเตอร์จะเป็นสัญญาณ ACที่ขี่ (Superimposed) อยู่บนส่วนประกอบทาง DC แต่สัญญาณ AC เป็นสิ่งที่ไม่ต้องการ ดังนั้นลูปฟิลเตอร์จะทำหน้าที่กำจัดสัญญาณ AC ออกไป เพื่อเหลือแต่แรงดัน DC ไปควบคุมความถี่ของ VCO

พิจารณาการทำงานโดยรวมของเฟสล็อกลูป อันดับแรกสมมติให้ความถี่เชิงมุม ω_1 ของสัญญาณอินพุต U_1 (t) เท่ากับความถี่กลาง ω_0 ในสภาวะนี้ VCO จะทำงานโดยผลิตความถี่ที่ความถี่กลางของมัน และจะเห็นว่าความต่างเฟส θ_e เท่ากับศูนย์ สัญญาณเอาต์พุต U_d(t) ของเฟสดีเทคเตอร์จะเป็นศูนย์ด้วย ดังนั้นสัญญาณเอาต์พุตของลูปฟิลเตอร์ U_f จะเท่ากับศูนย์ ซึ่งสภาวะที่ความถี่อินพุท ω_1 เท่ากับความถี่กลางของมัน VCO นี้จะเป็นเงื่อนไขที่ทำให้ VCO ผลิตความถี่ที่ความถี่กลางของ VCO



รูปที่ 2.19 ทรานเซี้ยนของเฟสล็อคลูปอันเนื่องมาจากการเปลี่ยนแปลงความถี่ของสัญญาณอ้างอิง

(a) สัญญาณอ้างอิง

(b) สัญญาณเอาท์พุท U_1(t) ของ VCO

(c) สัญญาณ U_d(t) ซึ่งเป็นฟังก์ชั่นของเวลา

(d) ความถี่เชิงมุม ω_2ของ VCO ซึ่งเป็นฟังก์ชั่นของเวลา

(e) ความถี่เชิงมุม ω_2 ของสัญญาณอ้างอิง U_1 (t)

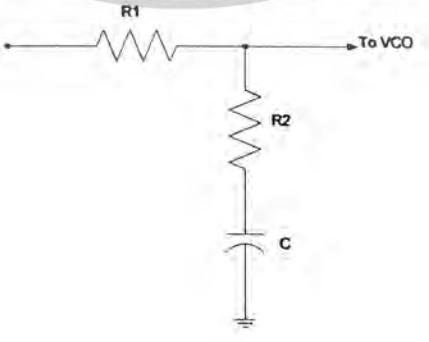
สมมติให้ในสภาวะเริ่มต้นความต่างเฟส θ_e ไม่เท่ากับศูนย์ สัญญาณเอาต์พุตของเฟสดีเทคเตอร์ $U_d(t)$ ก็จะไม่เท่ากับศูนย์ หลังจากช่วงเวลาหนึ่ง สัญญาณเอาต์พุตของลูปฟิลเตอร์ก็จะมีค่าหนึ่ง ๆ (ที่ไม่เท่ากับศูนย์) ทำให้ VCO ผลิตความถี่ขึ้นมาค่าหนึ่ง ซึ่งในที่สุดจะทำให้ความเฟส θ_e เท่ากับศูนย์

ต่อไปสมมติให้ความถี่ของสัญญาณอินพุตเปลี่ยนแปลงอย่างทันทีทันใดที่เวลา t_0 เป็นจำนวน $\Delta\omega$ ดังแสดงในรูปที่ 2.19 ซึ่งจะทำให้เฟสของสัญญาณอินพุตเริ่มจะนำ (leading) เฟสของสัญญาณเอาต์พุต และทำให้เกิดความต่างเฟสขึ้นตามเวลาทำให้สัญญาณเอาต์พุต $U_d(t)$ ของเฟสดีเทคเตอร์เพิ่มขึ้นตามเวลาด้วย หลังจากช่วงเวลาหนึ่งที่กำหนดโดยลูปฟิลเตอร์สัญญาณเอาต์พุต $U_f(t)$ ของลูปฟิลเตอร์ก็จะเพิ่มขึ้นและมีผลทำให้ VCO ก็จะผลิตความถี่ที่สูงขึ้นซึ่งจะทำให้ความต่างเฟสมีค่าน้อยลง หลังจากช่วงเวลาหนึ่ง VCO ก็จะผลิตความถี่ที่เท่ากับความถี่ของสัญญาณอินพุตค่าความต่างเฟสในสภาวะนี้จะมีค่าเท่ากับศูนย์หรือค่าใดค่าหนึ่ง ขึ้นอยู่กับชนิดของลูปฟิลเตอร์ที่ใช้

จะเห็นว่าในขณะนี้ VCO จะผลิตความถี่มากกว่าความถี่กลาง ω_0 ของ VCO เป็นจำนวน $\Delta\omega$ ซึ่งจะทำให้สุดท้ายสัญญาณ $U_d(t)$ มีค่า $U_f = \Delta\omega/K_0$ ถ้าความถี่กลางของสัญญาณอินพุตถูกมอดูเลททางความถี่ (FM) โดยสัญญาณความถี่ต่ำใดๆแล้ว สัญญาณเอาต์พุตของลูปฟิลเตอร์จะเป็นสัญญาณที่ถูกมอดดูเลทแล้ว (demodulated signal) ดังนั้นเราอาจใช้เฟสล็อคลูปเป็น FM ดีเทคเตอร์ได้ การพิจารณาอย่างง่าย ๆ ที่ผ่านมานี้แสดงให้เห็นว่าเฟสล็อคลูปไม่มีอะไรซับซ้อนเป็นเพียงแต่ระบบเซอร์โว (servo system) ซึ่งควบคุมเฟสของสัญญาณเอาต์พุต $U_f(t)$

ดังได้แสดงมาแล้วในรูปที่ 2.19 เฟสล็อคลูปของสัญญาณเอาต์พุตเท่ากับเฟสของสัญญาณอินพุตเสมอ ระบบนี้จะล็อคอยู่ตลอดเวลา อย่างไรก็ตามในบางกรณีที่ความถี่ของสัญญาณอินพุตเปลี่ยนแปลงไปมากอย่างทันทีทันใด อาจจะทำให้ระบบไม่ล็อคก็ได้ ซึ่งกลไกภายในของเฟสล็อคลูปจะพยายามกลับมาอยู่ที่สภาวะล็อคอีกครั้ง แต่จะล็อคได้หรือไม่นั้นก็ขึ้นกับองค์ประกอบอย่างอื่นอีกด้วย

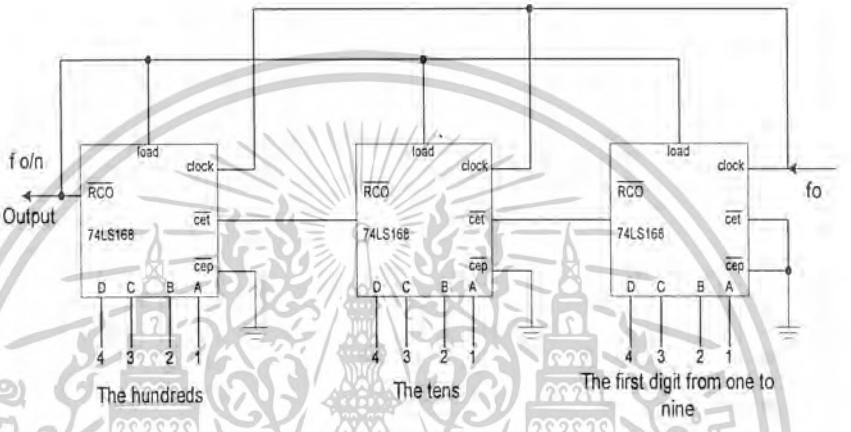
ลูปฟิลเตอร์ เป็นวงจรชนิดโลว์พาสธรรมดา ซึ่งทำหน้าที่กรองเอาเฉพาะสัญญาณความถี่ต่ำมาควบคุมความถี่ของ VCO โดยทั่วไปมักใช้ลูปฟิลเตอร์ชนิดพาสซีฟ (มีแต่ R กับ C หรืออาจใช้ฟิลเตอร์ชนิดแอคทีฟก็ได้) ดังแสดงในรูปที่ 2.20 ลูปฟิลเตอร์นี้เป็นตัวกำหนดคุณสมบัติการเปลี่ยนแปลงความถี่ก่อนเข้าสู่สภาวะล็อคที่เรียกว่าคุณสมบัติชั่วครู่ (Transient) ถ้าเลือกอัตราขยายลูป (Loop gain) และค่าคงตัวเวลาของลูป (Loop time constant) ไม่เหมาะสมความถี่ของเฟสล็อคลูปจะไม่ล็อคและจะเปลี่ยนไปเปลี่ยนมา



รูปที่ 2.20 แสดงตัวอย่างลูปฟิลเตอร์

ยังมีอีกภาคหนึ่งที่มีผลต่อช่วงเวลาที่ใช้ในการล็อคความถี่ นั่นก็คือ ภาคหาร N (หรือ Programmable divider) เวลาที่ใช้ในการล็อคความถี่เมื่อ N มีค่าน้อยสุดจะไม่เท่ากับเมื่อ N มีค่ามากที่สุด วงจรหาร N เกิดจากวงจรนับฐานสิบ (Decade counter) หลาย ๆ ชุดมาต่อร่วมกับเกทต่างๆเพื่อให้สามารถเลือกสั่งให้วงจรนับทำหน้าที่หารความถี่ได้ตามตัวเลขที่ตั้งไว้

วงจรหาร N นี้เป็นตัวที่รับคำสั่งเกี่ยวกับความถี่ไปควบคุม VCO เพื่อให้กำเนิดสัญญาณตามที่ต้องการ ตัว N จะเป็นตัวที่กำหนดย่านความถี่และจำนวนช่วงของความถี่ในวงจรรูปที่2.21 แสดงวงจรหารชนิดที่ใช้ไอซีตระกูล TTL



รูปที่ 2.21 แสดงผัง Programmable divider โดยใช้ IC ตระกูล TTL

วงจรหาร N บางชนิดใช้วิธีป้อนข้อมูล N เป็นแบบอนุกรม (Serial) วงจรหารประเภทนี้จะมีความซับซ้อน เพราะต้องมีสัญญาณนาฬิกา (Clock) มีวงจรแลทช์ (Latch) ฯลฯ ในการป้อนข้อมูล วงจรหาร N ประเภทนี้จะถูกควบคุมการทำงานด้วยไมโครคอมพิวเตอร์

ปัญหาสำคัญของวงจรสังเคราะห์ความถี่อีกอย่างหนึ่งก็คือ วงจรหาร N (หรือวงจรหารที่ตั้งโปรแกรมได้) ไม่สามารถทำงานที่ความถี่สูงกว่า 25 MHz ได้ ฉะนั้นเราจึงต้องหาทางลดทอนความถี่ที่ป้อนแก่วงจรหาร N ลง เพื่อให้วงจรโลจิกของวงจรหาร N ทำงานได้วิธีต่าง ๆ ที่นิยมได้แก่ ใช้ความถี่จากออสซิลเลเตอร์พิเศษ (PLL) มามิกซ์กับ VCO ให้ความถี่ลดลงก่อนที่จะป้อนให้แก่วงจรหาร อีกวิธีหนึ่งก็คือ ใช้วิธีปรีสเกลแบบสองโมดูลัสหารล่วงหน้าโดยใช้ตัวหาร 2 ค่า

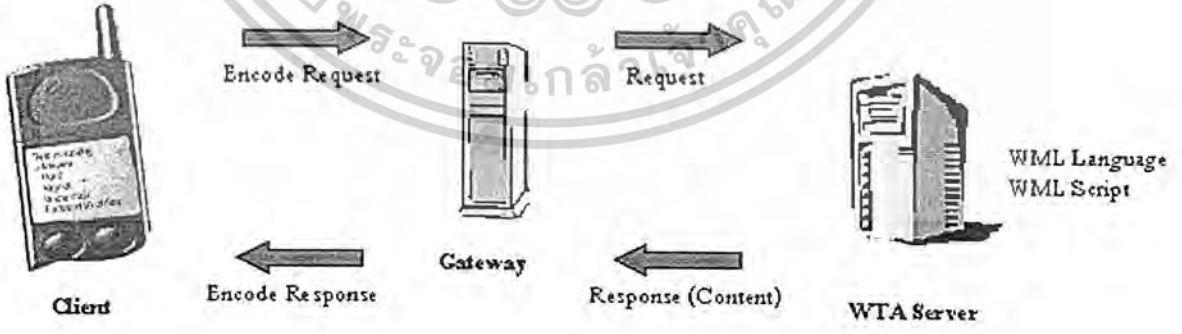
2. 5 WAP (Wireless Application Protocol)

WAP ย่อมาจากคำว่า Wireless Application Protocol WAP พัฒนาโดยความริเริ่มของ อิริสัน Motorola Nokia Unwired Planet จัดตั้งเป็น WAP Forum WAP Forum มีหน้าที่กำหนดมาตรฐานของ WAP พัฒนาโดยความริเริ่มของ อิริสัน Motorola Nokia Unwired Planet จัดตั้งเป็น WAP Forum WAP Forum มีหน้าที่กำหนดมาตรฐานของ WAP

WAP เป็น Application หนึ่งที่เป็นบริการเสริมสำหรับโทรศัพท์เคลื่อนที่ โดยใช้บริการหรือเปิดดูได้ทาง Screen ของโทรศัพท์เคลื่อนที่ สามารถเลือกดูข้อมูลต่างๆ ได้มากมายเช่นเดียวกับ Web เช่น ซื้อสิ้นค้า ส่ง Email ดูข่าวสาร ฯลฯ มีรูปแบบเป็นโปรโตคอลอีกรูปแบบหนึ่งที่มีขนาดลดทอนมาจากโพรโตคอลของอินเตอร์เน็ต ใช้กับโทรศัพท์เคลื่อนที่ ทั้งนี้เพราะว่าโทรศัพท์เคลื่อนที่มีข้อจำกัดหลายอย่างเมื่อเปรียบเทียบกับ PC Computer เช่น แบนด์วิดธ์ในการรับส่งแคบ จอแสดงผลมีขนาดเล็ก มีหน่วยความจำกัด CPU มีความสามารถน้อย กำลังไฟฟ้ามีจำกัดดังนั้นการพัฒนา WAP จึงต้องจำกัดอยู่ภายใต้เหตุผลข้างต้นWAP มีโปรโตคอลของตนเองที่ชื่อว่าWML(Wireless Markup Language) ถ้าเปรียบเทียบกับ WEB ก็คือHTML(Hypertext Markup Language)WML ได้พัฒนามาจาก HTML ให้สามารถทำงานได้ตามสภาพเทคโนโลยีของโทรศัพท์เคลื่อนที่ไร้สาย การใช้ WAP จะต้องมีโทรศัพท์เคลื่อนที่ที่สามารถใช้บริการ WAP ได้ บริเวณส่วนที่ใช้ WAP เราเรียกว่า WAP Micro browser จะลดรูปมาจาก Internet browser เช่น Netscape Internet Explorer

ระบบของ WAP โดยหลักประกอบไปด้วย

1. Client ทำหน้าที่ Request ขอใช้บริการโดยใช้ Micro browser เป็นตัวอ่านและส่งข้อมูล
2. Gateway ทำหน้าที่ Encoder เข้ารหัสให้ Client สามารถ อ่านข้อมูลได้
3. WTA หรือ Wireless Telephony Application ทำหน้าที่เป็น Server เก็บข้อมูล WML และ Response เมื่อมีการ Request จาก Client



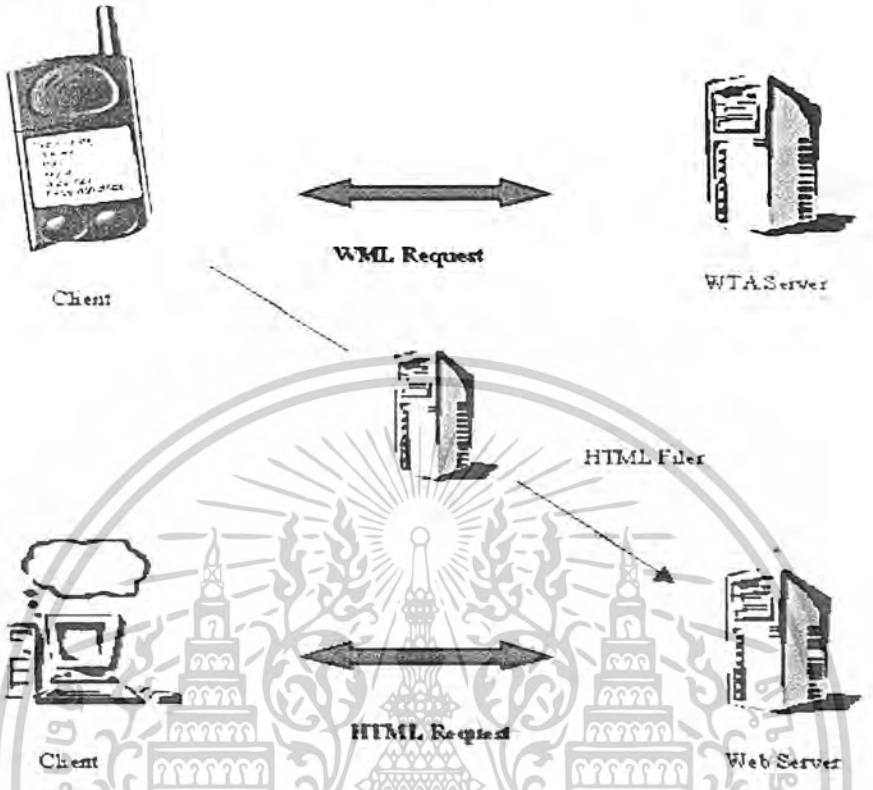
รูปที่ 2.22 แสดงระบบการทำงานของWAP

ระบบ WAPเริ่มจาก

1. Client Request ข้อมูลโดยผ่านทาง Gateway จะเป็นเข้ารหัสและส่ง Request ไปให้ WTA
2. Serverเมื่อ WTA Server ได้รับ Request ก็จะส่งข้อมูล WML ให้กับ Client

3. Gate Way จะเป็นตัวถอดรหัสและส่งให้ Client

2.5.1 ระบบของ WAP เมื่อเปรียบเทียบกับ WEB เป็นอย่างไร



รูปที่ 2.23 เปรียบเทียบระบบ WAP และWEB

WAP และ WEB หลักการจะไม่แตกต่างกัน WAP จะติดต่อเข้าสู่อินเตอร์เน็ตหรือ WEB ได้โดยผ่าน HTML Filterโดย HTML Filter จะทำหน้าที่แปลงข้อมูลจาก HTML เป็น WML

2.5.2 WAP มีประโยชน์อย่างไร

เข้าถึงสู่ระบบอินเตอร์เน็ตสามารถติดต่อสื่อสารได้ไร้พรมแดนอย่างแท้จริงคือโทรศัพท์เคลื่อนที่ด้วยกันและโทรศัพท์เคลื่อนที่กับอินเตอร์เน็ต เนื่องจากโทรศัพท์เคลื่อนที่ไร้สายและมีขนาดเล็กจึงมีอำนาจในการติดต่อสื่อสารได้ดีกว่าInternet PC Application ต่างๆบนWAPนี้จะถูกพัฒนาโดยใช้ภาษา WMLซึ่งย่อมาจาก Wireless Markup Language เป็นภาษาที่ตัดบางส่วนออกมาจากภาษา XML (Extensible Markup Language) ซึ่ง WMLนั้นถูกออกแบบมาเพื่อรองรับการพัฒนา application บนอินเตอร์เน็ตผ่านโทรศัพท์มือถือ หรือ เครื่องมือสื่อสารไร้สายต่างๆซึ่งมีข้อจำกัดในเรื่องของ พื้นที่หน้าจอแสดงผลที่จำกัด รวมถึงข้อจำกัดในเรื่องของจำนวน หน่วยความจำ ที่มีอยู่จำนวนจำกัด ทำให้ WML ถูกออกแบบมาเพื่อต้องทำการบีบอัดข้อมูลให้เล็กลงได้ด้วย

2.6 Wireless Markup Language (WML)

เป็นภาษาที่ใช้ในการสร้างเอกสาร ซึ่งสนับสนุนการแสดงผลบนหน้าจออุปกรณ์สื่อสารไร้สาย ที่สนับสนุนเทคโนโลยี WAP (Wireless Application Protocol) โดยภาษา WML มีการพัฒนามาจากภาษา

HTML ซึ่งเป็นมาตรฐานบนเครือข่ายอินเทอร์เน็ต ทำให้ WML มีความคล้ายคลึงกับ HTML ในแง่การสร้างและพัฒนาระบบ แต่เหมาะสำหรับการแสดงผลบนจอภาพขนาดเล็กผ่านเครือข่ายสื่อสารที่มีแบนด์วิดธ์ต่ำ เช่น เครือข่ายโทรศัพท์เคลื่อนที่ หรือวิทยุติดตามตัวแบบสองทาง รูปแบบการจัดการโครงสร้างของเอกสารตามมาตรฐาน WML จะอยู่ในลักษณะของการ์ด (Card) ซึ่งเว็บไซต์แต่ละแห่งจะมีการจัดแบ่งหน้าจอแต่ละหน้าออกจากกัน การเลือกรายการจะเป็นเงื่อนไขในการเลือกการ์ด โดยผู้ใช้มีอิสระในการเข้าชมข้อมูลในการ์ดต่างๆ ตามต้องการ โดยความสามารถของมาตรฐาน WML มีดังนี้

- สนับสนุนการแสดงข้อความและรูปภาพ WML สนับสนุนการประยุกต์ใช้งานเพื่อแสดง ข้อมูลในรูปแบบของข้อความ (Text) และ รูปภาพ (Image) ไปสู่ผู้ใช้บริการผ่านอุปกรณ์ปลายทาง ทั้งนี้จะครอบคลุมถึงการกำหนดโครงร่างของการแสดงผลบนหน้าจอและ แสดงข้อความแนะนำสู่ผู้ใช้บริการ ผู้พัฒนาโปรแกรมควรกำหนดรูปแบบมาตรมาตรฐาน ในการนำเสนอข้อมูลไปสู่อุปกรณ์ปลายทาง และเปิดโอกาสให้อุปกรณ์บราวเซอร์ปลายทาง ได้มีอิสระในการตัดสินใจกำหนดแนวทาง ในการแสดงข้อมูล วิธีการดังกล่าวเป็นไปในลักษณะเดียวกันกับการแสดงผลแบบ HTML
- การรับข้อมูลจากผู้ใช้บริการ WML สนับสนุนการรับข้อมูลป้อนเข้าจาก ผู้ใช้บริการรูปแบบ ในการรับข้อมูลอาจเป็นไปได้ทั้งในลักษณะของแบบฟอร์ม เมนูแสดงรายการ หรือ เป็นเพียงการป้อนตัวเลขง่าย ๆ ซึ่งทั้งหมดนี้จะปรากฏขึ้นบนหน้าจอของอุปกรณ์ปลายทาง ข้อกำหนดมาตรฐาน WAP ระบุเพียงความต้องการมาตรฐานทางเทคนิค ในการสนับสนุน การรับข้อมูลมิได้ชี้ชัดลงไปในรายละเอียด จึงถือเป็นการเปิดรับมาตรฐานอันหลากหลายของผู้ผลิตอุปกรณ์เครื่องลูกข่ายแต่ละราย ไม่เป็นการปิดกั้นทางเทคโนโลยี
- การเข้าถึงแหล่งข้อมูล WML สนับสนุนการสืบค้นข้อมูลและเข้าถึงแหล่งข้อมูลต่างๆ โดยใช้กลไกของ URL ตามมาตรฐานเดียวกับเครือข่ายอินเทอร์เน็ต ตัวอย่างเช่นการใช้คำสั่ง HTTP เป็นต้น วิธีในการเข้าถึงแหล่งข้อมูลเป็นได้ทั้งแบบการสร้างลิงค์เชื่อมโยง (Hyperlink) ตามมาตรฐาน HTML หรือการเชื่อมโยงระหว่างการ์ดตามมาตรฐาน WML เอง
- สนับสนุนมาตรฐานข้อมูลนานาชาติ เอกสารตามมาตรฐาน WML สนับสนุนรูปแบบตัวอักษรตามมาตรฐาน Unicode ซึ่งเป็นมาตรฐานที่รองรับตัวการแสดงผลอักษรทุก ภาษาทั่วโลก ทำให้สามารถรองรับการประยุกต์ใช้งานระดับนานาชาติ โดยไม่ต้องคอยตามสร้างโปรแกรม ประยุกต์แยกย่อยให้กับแต่ละชาติเป็นกรณีพิเศษ
- การใช้แบนด์วิดธ์อย่างมีประสิทธิภาพ มาตรฐาน WML มีการใช้เทคนิคหลายๆ ประการในการควบคุมประสิทธิภาพการนำเสนอข้อมูลบนอุปกรณ์สื่อสารปลายทาง ผ่านเครือข่ายสื่อสารที่มีแบนด์วิดธ์ต่ำ สิ่งที่สำคัญก็คือความสามารถ ในการส่งข้อมูล 1 การ์ดจากเครือข่ายไปสู่ผู้ใช้บริการหลายๆ ได้ในเวลาเดียวกัน รวมถึงความสามารถในการจัดสรรกระบวนการควบคุมการเชื่อมต่อต่างๆ สู่ผู้ใช้บริการ

WML คือภาษาหนึ่งของ XML ที่มีลักษณะเกี่ยวโยงจากชุดตัวอักษร WMLคือ เซตของข้อมูลอักษรโดยเป็นอักษรที่เป็นจริงทั้งหมดที่มีชนิดของข้อมูลที่มีอยู่ ชุดตัวอักษรในข้อมูล สำหรับ XML และ WMLคือชุดตัวอักษรสากลของ ISO/IEC -10646 ในปัจจุบันชุดตัวอักษรเหล่านี้จะเป็นอย่างเดียวกันของ Unicode 2.0

ชุดข้อมูล WMLอาจเป็นรหัสอักษรใดๆก็ได้ที่จะมีการกำหนดโดยการกำหนดของ HTML 4.0 Specification ชุดอักษรของข้อมูล WMLอาจเปลี่ยนเป็นรหัสอย่างอื่นก็ได้ อย่างไรก็ตามการเปลี่ยนเป็นรหัสอย่างอื่นสามารถนำไปสู่การสูญเสียข้อมูล และต้องหลีกเลี่ยงเมื่อผู้ใช้ข้อมูลพยายามใช้ที่เป็นรหัสเข้าช่วยการเปลี่ยนแปลงรหัสโดยไม่จำเป็น ต้องหลีกเลี่ยงได้เมื่อจะเกิดการสูญเสียข้อมูล ถ้าต้องการทำเช่นนั้น การแปลงรหัสควรจะทำก่อนที่จะมีการส่งข้อมูลไปหาผู้ให้ข้อมูลกับผู้ใช้อีกทีหนึ่ง

WML Syntax

ภาษา WMLโดยส่วนใหญ่จะเป็นโครงสร้างรูปแบบของลำดับคำจากภาษา XML เป็นส่วนใหญ่ Entities ข้อมูล WMLสามารถบรรจุตัวเลขหรือชื่อของตัวอักษรที่มีอยู่จริง ซึ่งจะกำหนดตัวอักษรอย่างเจาะจงลงไปในชุดของข้อมูลอักษร Entities ถูกใช้ในการระบุตัวอักษรในชุดอักษรข้อมูล ซึ่งจะต้องเป็นการออกในชนิดข้อมูลของ WML ampersand และจบลงด้วย semicolon

Tag เป็นตัวบรรยายคำสั่งของภาษาในลักษณะต่างๆโดยจะมีรูปแบบหลักๆดังนี้

<tag> ใช้เป็นตัวเริ่มคำสั่งต่างๆ

</tag> ใช้เป็นตัวจบคำสั่งต่างๆ

<tag/> เป็นตัวประกอบว่าง เช่น
 เป็นตัวกำหนดการจบบรรทัด

Elements เป็นตัวระบุคำสั่งต่างๆ ที่สมบูรณ์สามารถทำงานได้ โดยประกอบด้วย tag เปิด และ tag ปิด หรือ tag เดี่ยวๆ ดังตัวอย่างต่อไปนี้

<tag> content </tag>

- or –

<tag/>

Attributes การระบุข้อมูล WML จะมีข้อมูลเพิ่มเติมสำหรับตัวประกอบข้อมูลต่างๆ จะถูกกำหนดมาแล้วเสมอใน start tag ของตัวพื้นฐาน ตัวอย่างเช่น < tag attr = " value " / > บันทึกชื่อของข้อมูลจะต้องอยู่ด้านล่างกรณี WML กำหนดไว้ว่า ค่าของข้อมูลทั้งหมดจะอยู่ในวงเล็บโดยใช้ double question marks (*) หรือ single quotation marks (') อย่างใดอย่างหนึ่งก็ได้ เครื่องหมาย single quotation สามารถใช้ร่วมกับค่าของข้อมูลเมื่อค่าถูกกำหนดให้จำกัดโดย double quotation marks และ Vice versa คุณสมบัติของเอกลักษณ์อาจรวมถึงค่าของข้อมูลได้เช่นกัน

Comments คำแนะนำหรือหมายเหตุภายใน Source Code ที่มีไว้เพื่ออธิบายส่วนต่างๆของโปรแกรมมีรูปแบบการใช้งานดังนี้ <!- คำแนะนำ --> โดยที่คำภายในสัญลักษณ์นี้จะไม่แสดงออกให้กับผู้ใช้เห็น

Variables ปัจจัยกำหนด สามารถกำหนดสำหรับ card WML และมีความหมายหลากหลายในการใช้ของ deck สำหรับการใช้รูปแบบต่างๆของ card สลับกับ deck นั้น รูปประโยคโครงสร้างจะถูกใช้ตามข้างล่างนี้

$identifier

$(identifier)

$(identifier: conversion)

ในกรณีที่ตัวแปรมีอักขระว่างเป็นส่วนประกอบด้วยตัวแปรตัวนั้นต้องใช้วงเล็บในการอ้างถึง ยกตัวอย่างเช่น (A B)

Card and Deck ข้อมูลทั้งหมดของ WML จะถูกจัดให้อยู่ในรูปของ cards และ decks โดยที่ decks 1decks จะมีcard บรรจุอยู่ได้หลาย cards และ cards1 cardsจะเปรียบเสมือนหน้าจอที่จะปรากฏต่อผู้ใช้งาน

องค์ประกอบของ WML ทั้งหมดมีข้อมูลหลักอยู่ 2 อย่างคือ id และ class ที่สามารถใช้เป็นตัวส่งข้อมูล server-side ตัว id มีไว้เพื่อเป็นตัวประกอบให้ชื่อเฉพาะภายใน single deck. class มีไว้เพื่อเป็นตัวช่วยเป็นหน่วยพื้นฐานให้หนึ่งหรือชุดต่างๆที่มากกว่านั้น

2.7 การทำงานของ MySQL

เป็นDatabase Server ที่เหมาะกับองค์กรขนาดกลางที่มีข้อมูลไม่มากนัก และเป็นระบบจัดการฐานข้อมูลเชิงสัมพันธ์ (Relation Database Management System) ซึ่งเป็นฟรีแวร์ทางด้านฐานข้อมูลจึงได้รับความนิยมอย่างมากในปัจจุบัน สามารถดาวน์โหลดซอร์สโค้ดได้จากอินเตอร์เน็ตโดยไม่เสียค่าใช้จ่าย และสามารถแก้ไขได้ตามความต้องการ พร้อมทั้งยังสนับสนุนการใช้งานบนระบบปฏิบัติการตัวอย่างเช่น Unix, Mac และWindows นอกจากนี้ยังทำงานร่วมกับ Java,C,C++,PHP,ASP

การทำงานของ MySQL นั้นอาจจะยุ่งยากในการคีย์ข้อมูล เพราะทำงานบน DOS จึงได้ใช้โปรแกรมที่สามารถจัดการฐานข้อมูลเป็นไปได้โดยง่าย คือ phpMyAdmin ซึ่งเป็นโปรแกรม mysql client แบบ opensource ที่ใช้จัดการ MySQL ได้ผ่านหน้าเว็บบราวเซอร์ได้เลย ซึ่งโปรแกรมนี้เขียนด้วย PHP

2.8 การทำงานของ PHP

ในช่วงแรกภาษาที่นิยมใช้ในการทำงานของระบบเครือข่ายคือ HTML (Hypertext Markup Language) แต่ภาษา HTML เป็น static Language (คือภาษาที่ใช้สร้างข้อมูลประเภท ตัวอักษร ภาพ หรือออบเจ็กต์อื่นๆที่ไม่สามารถเปลี่ยนแปลงได้ด้วยตัวเองหรือข้อมูลคงที่นั่นเอง) ต่อมาได้มีการพัฒนาภาษาที่เป็น Dynamic Language (คือภาษาที่ข้อมูลจะเปลี่ยนแปลงโดยอัตโนมัติตามเงื่อนไขต่างๆที่ผู้เขียนกำหนดไว้) ขึ้นมามากมาย โดยเฉพาะภาษาประเภทสคริปต์ (Script) ที่สามารถติดต่อกับผู้ใช้ได้ และหนึ่งในภาษานั้นก็คือภาษา PHP ซึ่งเป็นภาษาที่ได้รับความนิยมเป็นอย่างมากในปัจจุบัน

ภาษา PHP ถูกสร้างขึ้นในปี ค.ศ. 1994 มีชื่อว่า Personal Homepage เป็นภาษา Open source Product คือสามารถนำมาใช้งานโดยไม่เสียค่าใช้จ่าย ต่อมาได้เปลี่ยนชื่อเป็น PHP Hypertext Preprocessor คุณสมบัติของตัวแปรบน PHP ต้องขึ้นต้นด้วย $ เช่น

$variable = value;

ชนิดของตัวแปร สามารถกำหนดได้โดยไม่ต้องประกาศชนิดของตัวแปรก็ได้ และเมื่อกำหนดค่าให้กับตัวแปรที่มีชนิดที่แตกต่างจากชนิดเดิม ชนิดของตัวแปรจะเปลี่ยนแปลงตามชนิดของตัวแปรที่กำหนดให้ใหม่ ค่าคงที่ เป็นตัวแปรอีกประเภทหนึ่งที่ใช้เก็บค่าใดค่าหนึ่งตลอดการใช้งานโปรแกรม โดยไม่มีการเปลี่ยนแปลงค่า

ลักษณะของ PHP ที่แตกต่างจากภาษาสคริปต์แบบอื่นๆ คือ PHP ได้รับการพัฒนาและออกแบบมา เพื่อใช้งานในการสร้างเอกสารแบบ HTML โดยสามารถสอดแทรกหรือแก้ไขเนื้อหาได้โดยอัตโนมัติ ดังนั้นจึงกล่าวว่า PHP เป็นภาษาที่เรียกว่า server-side หรือ HTML-embedded scripting language เป็นเครื่องมือที่สำคัญชนิดหนึ่งที่ช่วยให้เราสามารถสร้างเอกสารแบบ Dynamic HTML ได้อย่างมีประสิทธิภาพและมีลูกเล่นมากขึ้น

เนื่องจากว่า PHP ไม่ได้เป็นส่วนหนึ่งของตัว Web Server ดังนั้นถ้าจะใช้ PHP ก็จะต้องดูก่อนว่า Web Server นั้นสามารถใช้สคริปต์ PHP ได้หรือไม่ ยกตัวอย่างเช่น PHP สามารถใช้ได้กับ Apache Web Server และ Personal Web Server (PWS) สำหรับระบบปฏิบัติการ Windows 95/98/NT

ในกรณีของ Apache เราสามารถใช้ PHP ได้สองรูปแบบคือ ในลักษณะของ CGI และ Apache Module ความแตกต่างอยู่ตรงที่ว่า ถ้าใช้ PHP เป็นแบบโมดูล PHP จะเป็นส่วนหนึ่งของ Apache หรือเป็นส่วนขยายในการทำงานนั่นเอง ซึ่งจะทำงานได้เร็วกว่าแบบที่เป็น CGI เพราะว่า ถ้าเป็น CGI แล้ว ตัวแปรชุดคำสั่งของ PHP ถือว่าเป็นแค่โปรแกรมภายนอก ซึ่ง Apache จะต้องเรียกขึ้น มาทำงานทุกครั้งที่ต้องการใช้ PHP ดังนั้น ถ้ามองในเรื่องของประสิทธิภาพในการทำงาน การใช้ PHP แบบที่เป็นโมดูลหนึ่งของ Apache จะทำงานได้มีประสิทธิภาพมากกว่า

2.9 การทำงานของMCS-51

ในงานควบคุมระบบต่างๆ ได้นำไมโครโปรเซสเซอร์หรือไมโครคอมพิวเตอร์เข้ามาใช้ในงานควบคุมโดยในการใช้งานนั้นจะต้องมีการต่อร่วมกับอุปกรณ์ภายนอกจำพวกพอร์ต I/O หน่วยความจำข้อมูล(RAM) หน่วยความจำโปรแกรม(ROM)ทั้งยังมีชิปจำพวก UART(Universal Asynchronous Receiver Transmitter) เพื่อใช้ในการรับส่งข้อมูลแบบอนุกรม โดยมีหน้าที่เปลี่ยนข้อมูลจากรูปแบบขนาน (Parallel)ไปเป็นข้อมูลในรูปแบบอนุกรม (Series) สำหรับทำการส่งชุดข้อมูลและแปลงข้อมูลจากรูปแบบอนุกรมกลับเป็นรูปแบบขนานในขั้นตอนการรับข้อมูล ซึ่งจะเห็นว่าการจะนำไมโครโปรเซสเซอร์หรือไมโครคอมพิวเตอร์ ดังกล่าวมาใช้ในระบบควบคุมโดยเฉพาะนั้น จะค่อนข้างยุ่งยากสิ้นเปลืองอุปกรณ์ร่วมต่างๆ เพื่อที่จะให้ครอบคุมการใช้งานควบคุมที่มีประสิทธิภาพ ฉะนั้นจากปัญหาต่างๆที่ผ่านมา จึงทำให้มีบริษัทผู้ผลิตชิปไมโครโปรเซสเซอร์หรือไมโครคอมพิวเตอร์ดังกล่าวได้ทำการพัฒนาและผลิตชิปที่มีการรวมคุณสมบัติต่าง ๆ เอาไว้ครบเพื่องานควบคุมโดยเฉพาะ

คุณสมบัติของ MCS-51

1. ใช้เทคโนโลยีชั้นสูงในการสร้างโดยมีทั้งประเภทHMOS, CMOSและ CHMOS ทำงานด้วยแหล่งจ่ายไฟ+5 Vdc เพียงแหล่งเดียว

2. มีหน่วยประมวลผลขนาด 8 บิต

3. สามารถติดต่อกับหน่วยความจำภายนอกทั้งหน่วยความจำโปรแกรม และหน่วยความจำข้อมูลได้สูงสุด 64 Kbytes

4. มีพอร์ต I/O แบบขนานสองทิศทางจำนวน 4 พอร์ต พอร์ตละ 8 บิต รวมทั้งหมดเป็น 32 บิต จะใช้ในการเข้าถึงแอดเดรสและข้อมูลสำหรับติดต่อกับหน่วยความจำภายนอก

5. พอร์ตใช้งานทุกพอร์ตและมีลักษณะเป็นพอร์ตแลทช์ (Latch) คงสภาวะ

6. มีขาพอร์ตที่ใช้สำหรับการรับส่งข้อมูลแบบอนุกรม

7. หนึ่ง Machine Cycle จะใช้เวลา 1 ไมโครวินาที โดยใช้ X-TAL 12 MHZ

8. สามารถกำหนดการใช้งานพอร์ต อนุกรม I/O ได้ ในระดับไบต์หรือบิตได้ โดยตรง

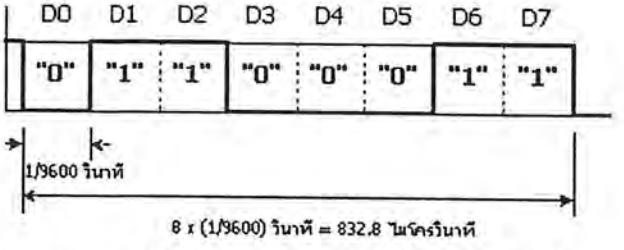
9. ตัวเลขทางคณิตศาสตร์ใช้ได้ทั้งระบบฐานสอง และฐานสิบหก

2.10 การเชื่อมต่อ HARDWARE ผ่านทางพอร์ต RS-232

เนื่องจากว่าการติดต่อระหว่างไมโครคอนโทรลเลอร์กับคอมพิวเตอร์ ไม่สามารถติดต่อกันได้โดยตรง เนื่องมาจากว่าสัญญาณไฟฟ้ามีความแตกต่างกัน การที่จะติดต่อระหว่างกันต้องมีอุปกรณ์ที่ทำหน้าที่ในการเชื่อมต่อสัญญาณ เพื่อให้ข้อมูลสามารถรับส่งกันได้

ในการสื่อสารข้อมูลแบบอนุกรม เพื่อรับหรือส่งข้อมูลจะเป็นลักษณะของกลุ่มข้อมูล ดังนั้นอัตราความเร็วจะต้องมีค่าเท่ากันระหว่างการรับและการส่งโดยทั่วไปเราจะระบุความเร็วของจำนวนบิตในการรับและส่งข้อมูล เป็นจำนวนของบิตที่จะส่งใน 1 วินาที โดยเรียกความเร็วในการส่งข้อมูลว่า อัตราบอด (Baud Rate) ซึ่งมีหน่วยเป็นบิตต่อวินาที เช่น 300, 1,200, 2,400, 4,800 และ 9,600 บิตต่อวินาที ในรูป 2.24 มีการส่งข้อมูลด้วยความเร็ว 9600 บิตต่อวินาที จะใช้เวลาในการรับส่งข้อมูลหนึ่งบิตมีค่าเท่ากับ 1/9600 หรือ 104.1 ไมโครวินาที และเวลาในการรับส่งข้อมูลทั้ง 8 บิตจะมีค่าเท่ากับ 8 x 104.1 หรือ832.8 ไมโครวินาที

D7	D6	D5	D4	D3	D2	D1	D0
1	1	0	0	0	1	1	0



รูปที่ 2.24 แสดงการส่งข้อมูลแบบอนุกรมด้วยความเร็ว 9600 บิตต่อวินาที

2.11 รูปแบบของการสื่อสารข้อมูลอนุกรม

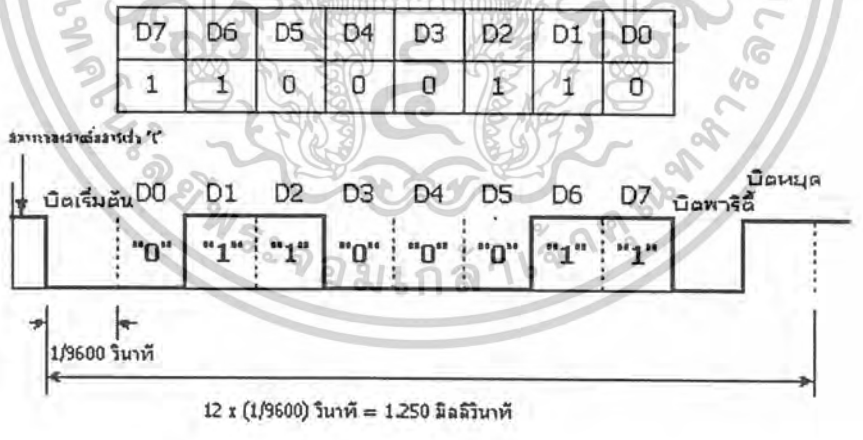
การสื่อสารข้อมูลอนุกรมแบบอะซิงโคนัส เป็นวิธีการรับและส่งข้อมูลโดยไม่ต้องอาศัยสัญญาณนาฬิกาส่งร่วมไปด้วย แต่จะใช้อัตราความเร็วของจำนวนข้อมูลต่อวินาที และจะทำการเพิ่มบิตข้อมูลบางอย่างร่วมไปกับการส่งข้อมูลจริง เพื่อจะได้ทำการตรวจสอบข้อมูลได้อย่างถูกต้องมากยิ่งขึ้นแสดงดังรูปที่ 2.25 ประกอบด้วยกัน 4 ส่วนคือ

1.บิตเริ่มต้น (Start bit) จะมีขนาด 1 บิต จะเป็นระดับลอจิกตรงกันข้ามกับระดับลอจิกของสภาวะสายสื่อสาร ขณะที่ยังไม่มีการส่งข้อมูล

2. บิตข้อมูล (Data bit) จะเริ่มจากบิตที่มีนัยสำคัญต่ำสุดก่อนหรือ บิต LSB ก่อน โดยข้อมูลที่จะส่งอาจจะมีขนาด 5, 6, 7 หรือ 8 บิตก็ได้

3. บิตแสดงสภาวะเลขคู่หรือเลขคี่ (Parity bit) มีขนาด 1 บิต โดยบิตนี้จะนำไปต่อท้ายกับบิตข้อมูล ค่าของบิตนี้ขึ้นอยู่กับจำนวนค่าของข้อมูลที่เป็น "1" โดยเลือกการส่งข้อมูลเป็นแบบ พาริตี้คู่ หรือ พาริตี้คี่ ตัวอย่าง ถ้ากำหนดให้มีการส่งข้อมูลแบบพาริตี้คู่ แต่ข้อมูลมีเลข 1 เป็นจำนวนคี่ ก็จะให้บิตพาริตี้นี้เป็น "1"เพื่อจะได้จำนวนเลข "1" เป็นคู่นั่นเอง ทำนองเดียวกันทางด้านรับเองก็ต้องมีการตรวจสอบจำนวนข้อมูลที่รับเข้ามาเป็น "1" รวมทั้งบิตพาริตี้ 1 บิต ถ้ามีค่า "1"เป็นจำนวนคู่ แสดงว่าข้อมูลที่รับเข้ามาถูกต้อง สามารถกำหนดการรับและส่งข้อมูลเป็นแบบ NONE โดยไม่ต้องมีการตรวจสอบพาริตี้บิตก็ได้

4. บิตสุดท้ายหรือบิตหยุด (Stop bit) เป็นการระบุถึงขอบเขตของการสิ้นสุดข้อมูล โดยจะทำให้ขาข้อมูลมีสถานะ ลอจิกเป็น "1" ซึ่งอาจมีจำนวนมากกว่า หนึ่งบิตก็ได้ เช่น 1 บิต 1.5 บิต หรือ 2 บิต

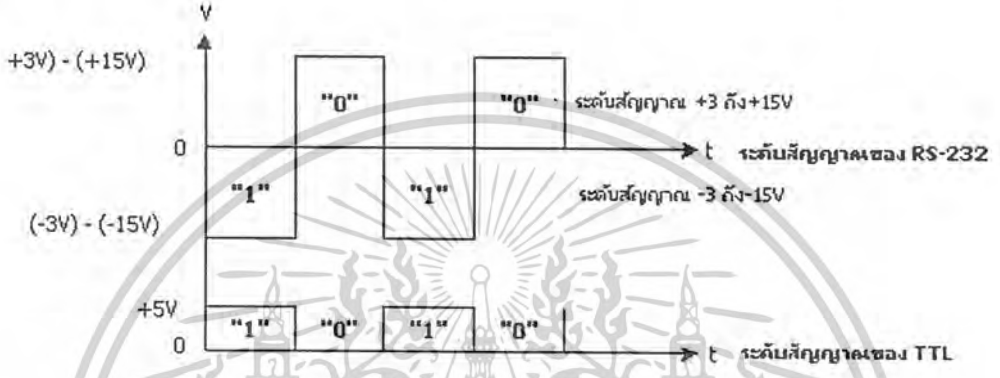


รูปที่ 2.25 แสดงการส่งข้อมูลขนาด 8 บิตแบบอนุกรมพร้อมด้วย บิตเริ่มต้น, บิตพาริตี้, บิตหยุด

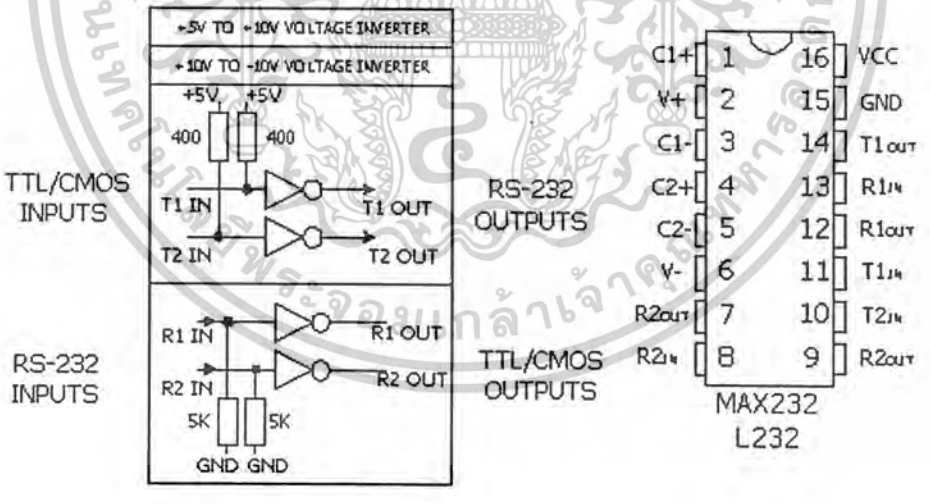
การกำหนดมาตรฐานการเชื่อมต่อแบบอนุกรม EIA RS-232 (x) เป็นมาตรฐานอุตสาหกรรม โดยคณะกรรมการสมาคมอุตสาหกรรมอิเล็กทรอนิกส์ (Electronic Industries Association) ออกแบบมาเพื่อใช้ในการส่งข้อมูลอนุกรมแบบอะซิงโครนัส 2 ทิศทางเพื่อให้มีการใช้งานในการเชื่อมต่อที่สอดคล้องกันระหว่างอุปกรณ์คอมพิวเตอร์ต่างๆ การรับส่งสัญญาณจะกำหนดความยาวสูงสุดไว้ที่ไม่เกิน 50 ฟุตโดยมี

ระดับ สัญญาณตั้งแต่ 3 โวลต์ จนถึง 15 โวลต์ สำหรับลอจิก "0" และมีระดับแรงดันที่ -3 โวลต์ จนถึง -15 โวลต์สำหรับลอจิก"1"

ดังนั้นสังเกตได้ว่าจะมีระดับแรงดันที่ใช้ในสถานะลอจิก "0" และ ลอจิก "1" แตกต่างออกไปจากระบบไอซีดิจิตอลทั่วๆไปการต่อใช้งานจึงต้องมีอุปกรณ์ที่ทำหน้าที่เปลี่ยนระดับแรงดันจาก 0 - 5 โวลต์ จากไมโครคอนโทรลเลอร์ ให้เป็นระดับแรงดันที่สูงกว่า +3 หรือต่ำกว่า - 3 โดยจะมีไอซีสำเร็จรูปพร้อมใช้งาน หรืออาจจะต่อวงจรจากทรานซิสเตอร์ได้



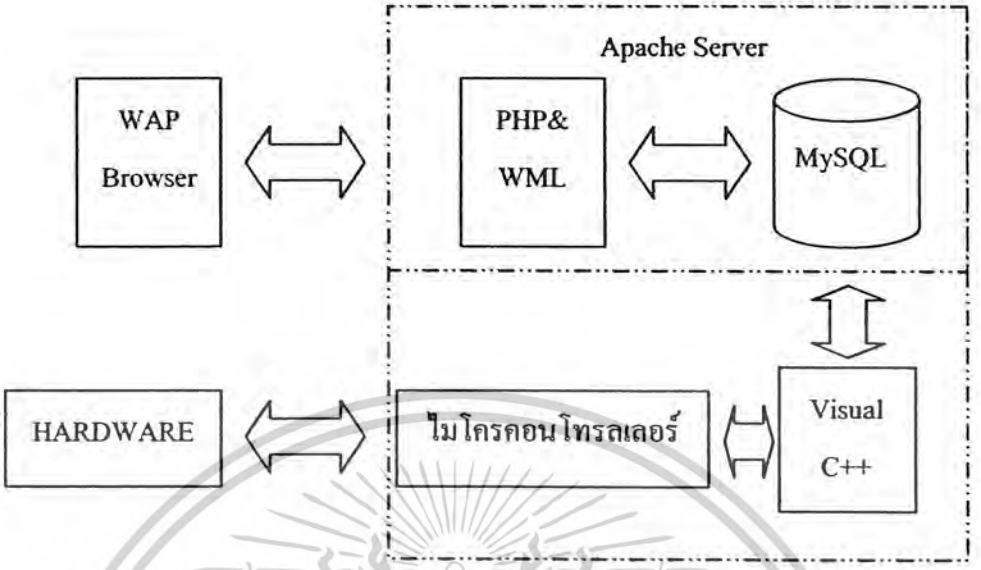
รูปที่ 2.26 แสดงระดับแรงดันสัญญาณของพอร์ตอนุกรม RS-232 กับ TTL



รูปที่ 2.27 แสดงวงจรภายในของไอซี MAX232

เป็นไอซีที่แปลงระดับสัญญาณจากระดับ TTL ไปเป็นระดับของ RS-232 และในทำนองเดียวกันก็รับระดับสัญญาณจาก RS-232เพื่อแปลงเป็นระดับสัญญาณจากระดับ TTL ให้กับไมโครคอนโทรลเลอร์ได้ ส่วนเชื่อมต่อนี้จะใช้ไอซีเบอร์ MAX232 ซึ่งเป็นไอซีที่ทำหน้าที่ในการแปลงแรงดันในระดับของสัญญาณทีทีแอลหรือซีมอส ไปเป็นสัญญาณไฟฟ้าขนาด ±15โวลต์

2.12 การทำงานร่วมกันระหว่างโปรแกรมต่างๆ



รูปที่ 2.28 แสดงการทำงานร่วมกันระหว่างโปรแกรมต่างๆ

แยกหน่วยแสดงผลทำงานออกเป็น 2 ส่วน

1. ส่วนแสดงผลภายในป้ายรถประจำทาง

ในแต่ละป้ายแสดงผลจะรับสัญญาณมาจากเซิร์ฟเวอร์โดย Visual C++ ทำหน้าที่ในการรับส่งข้อมูลต่างๆไปยัง MySQL เพื่อเก็บข้อมูลเข้าไปยังฐานข้อมูลแล้วทำหน้าที่ในการคำนวณต่างๆเพื่อส่งกลับไปยังบอร์ดแสดงผลในแต่ละป้าย โดยค่าที่ส่งออกไป เช่น เวลาโดยเฉลี่ยที่รถประจำทางจะมาถึง และรถประจำทางสายที่ต้องการรออยู่ถัดจากป้ายที่รอกี่ป้าย โดยส่งค่าเป็นรหัสแอสกี เมื่อข้อมูลส่งไปยังไคลเอนต์ในแต่ละป้าย ก็จะทำการแปลงข้อมูลที่เป็นรหัสแอสกีไปเป็นรหัส BCD เช่น

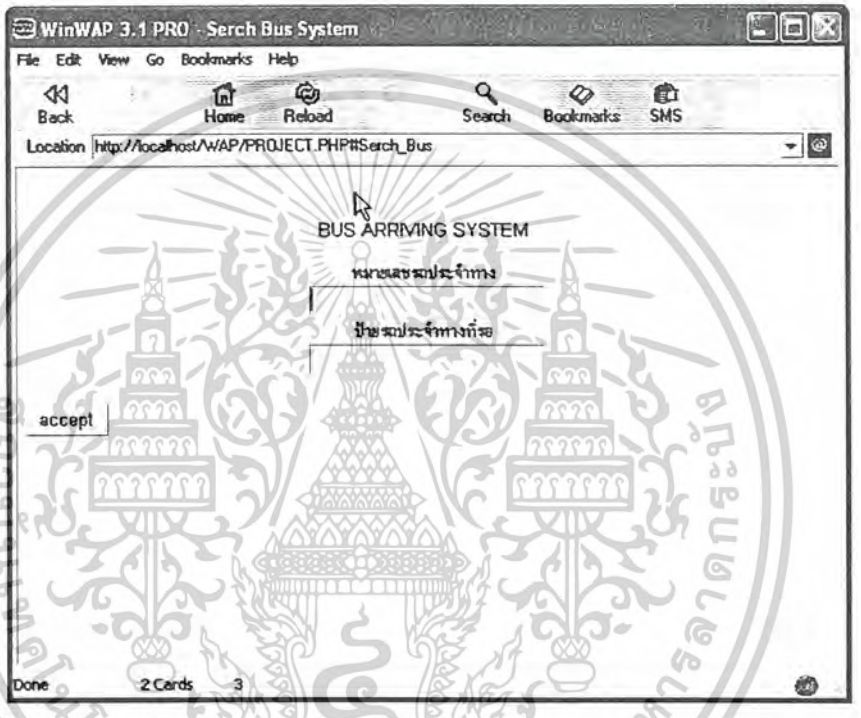
เซิร์ฟเวอร์ส่งข้อมูลมาเป็น 10522153 จะแบ่งออกเป็น2ชุด ชุดแรกคือ 1052 หมายถึง รถประจำทางสายที่1 จะมาถึงในเวลาเฉลี่ย 05 นาที โดยตอนนี้รถประจำทางที่ต้องการรออยู่ถัดจากป้ายนี้ 2 ป้าย ชุดที่สองคือ 2153 หมายถึง รถประจำทางสายที่ 2 จะมาถึงป้ายในเวลาประมาณ 15 นาที โดยตอนนี้รถประจำทางที่ต้องการรออยู่ถัดจากป้ายนี้ 3 ป้าย การแปลงข้อมูลไปเป็นรหัส BCD ซึ่งรูปแบบลักษณะเป็นดังนี้

05 รหัสแอสกี '0' คือ 30H ในฐาน 16 รหัสแอสกี '5' คือ 35H ในฐาน 16

15 รหัสแอสกี '1' คือ 31H ในฐาน 16 รหัสแอสกี '5' คือ 35H ในฐาน 16

2. ส่วนแสดงผลผ่านทางโทรศัพท์เคลื่อนที่

ใช้โปรแกรม WMLและPHP ซึ่งสามารถทำงานร่วมกันได้เพื่อแสดงข้อมูลต่างๆแสดงผลออกทางโทรศัพท์เคลื่อนที่ โดย PHP จะใช้ในการเขียนเกี่ยวกับฟังก์ชันการคำนวณค่าต่างๆ เพื่อให้ WMLซึ่งจะติดต่อระหว่างโทรศัพท์มือถือกับ PHP โดยดึงข้อมูลต่างๆในการคำนวณมาจากฐานข้อมูลใน MySQL โดยการแสดงผลจะใช้อีมูเลเตอร์ (Emulator) ของโทรศัพท์เคลื่อนที่ คือ WinWap 3.1 Pro ซึ่งสามารถแสดงผลข้อมูลต่างๆเสมือนเป็นโทรศัพท์เคลื่อนที่

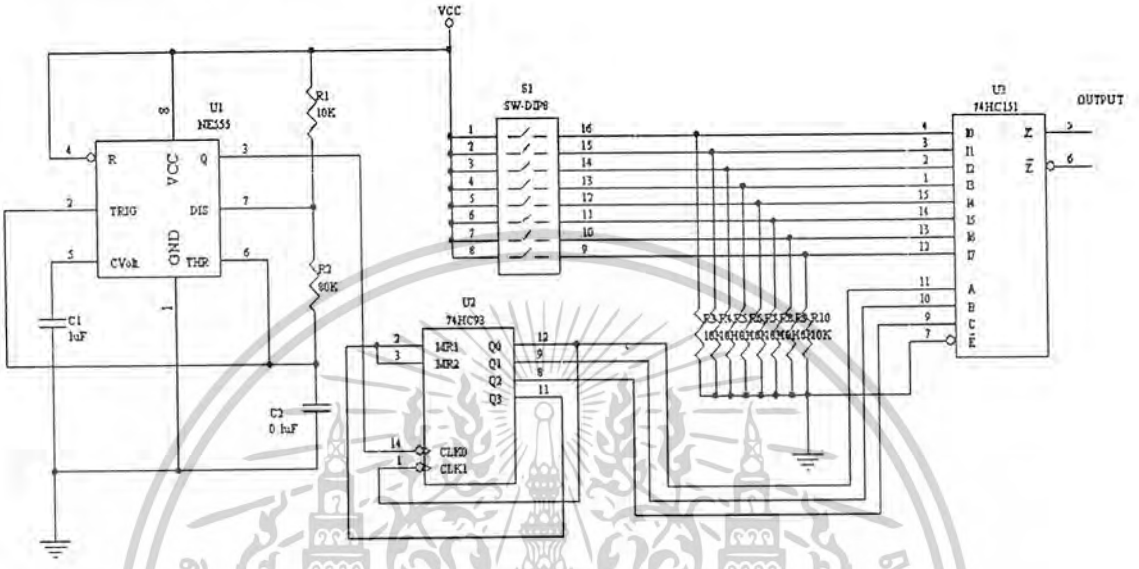


รูปที่ 2.29 แสดงหน้าจอการทำงานของโปรแกรมอีมูเลเตอร์ WINWAP 3.1 PRO

จากรูปที่ 2.29 แสดงหน้าการทำงานของโปรแกรมอีมูเลเตอร์ซึ่งรันอยู่บนเซิร์ฟเวอร์จำลองโดยใช้โปรแกรมเซิร์ฟเวอร์ของ Apache ทำงานร่วมกับ PHP, WML และMySQL โดยสามารถป้อนข้อมูลผ่านทางบราวเซอร์ของWAPได้

บทที่ 3
การคำนวณและการสร้าง

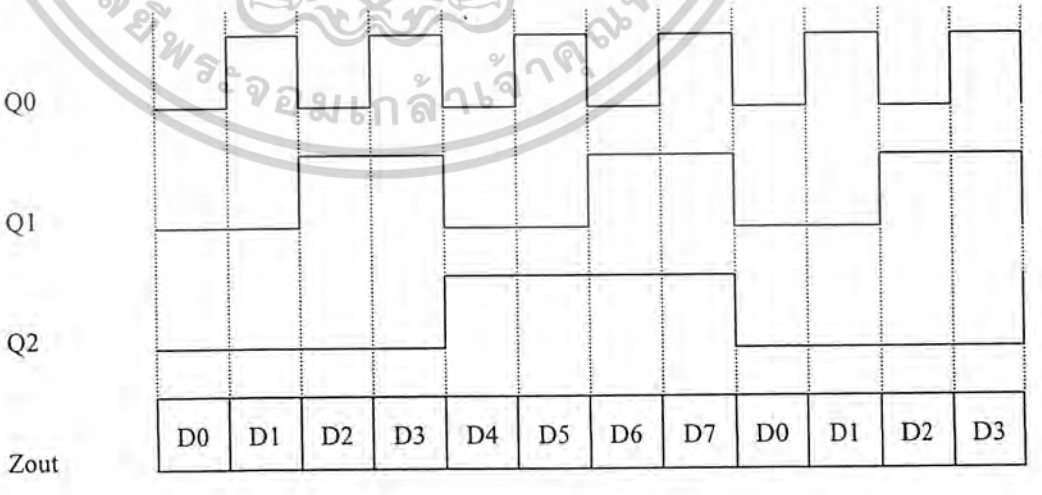
3.1 การออกแบบวงจรเข้ารหัส



รูปที่ 3.1 วงจรเข้ารหัสดิจิตอล

การเข้ารหัสดิจิตอลโดยใช้การมัลติเพล็กซ์แบบแบ่งเวลาโดยใช้วงจร Binary counter เป็นตัวทำหน้าที่ในการเลือกสวิทซ์ในแต่ละช่อง ซึ่งวงจร binary counter ก็จะมีส่วนที่รับสัญญาณ oscillatorโดยใช้วงจร อะสเตเบิล (Astable) ที่ถูกคำเนิดมาจาก ไอซีเบอร์ NE555 ทำหน้าที่ในการผลิตความถี่ขึ้นมา

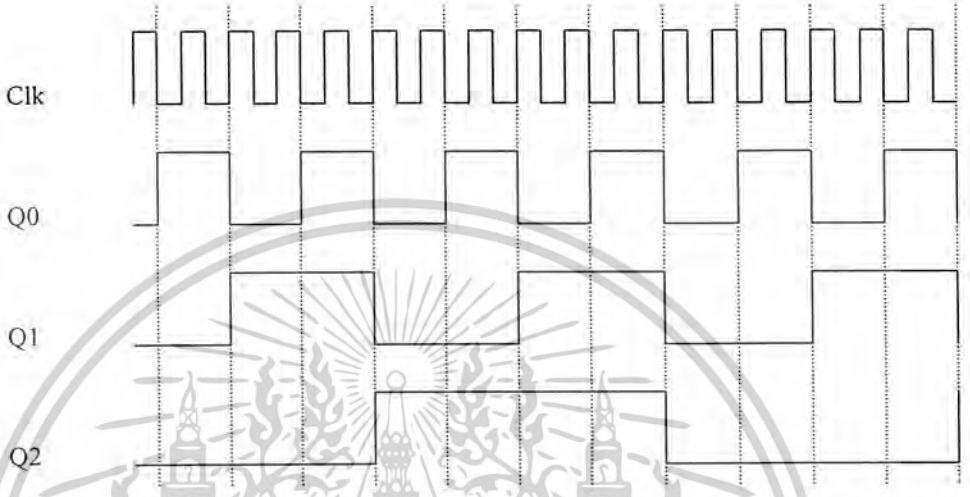
3.1.1 วงจรมัลติเพล็กซ์



รูปที่ 3.2 แสดงสัญญาณของวงจร Multiplex

ใช้ไอซีเบอร์ 74HC151 เป็นไอซีที่สามารถรับสัญญาณเข้ามาได้ 8 ช่องสัญญาณและเอาท์พุตออก 1 ช่องสัญญาณ โดยมีขาสัญญาณในการเลือกช่องสัญญาณขนาด 3 บิต

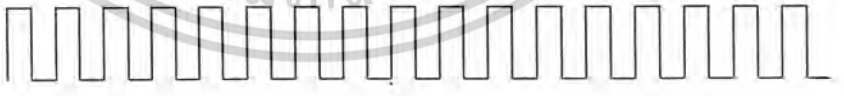
3.1.2 วงจรไบนารีเคาน์เตอร์(Binary Counter)



รูปที่ 3.3 แสดงสัญญาณของวงจร Binary counter

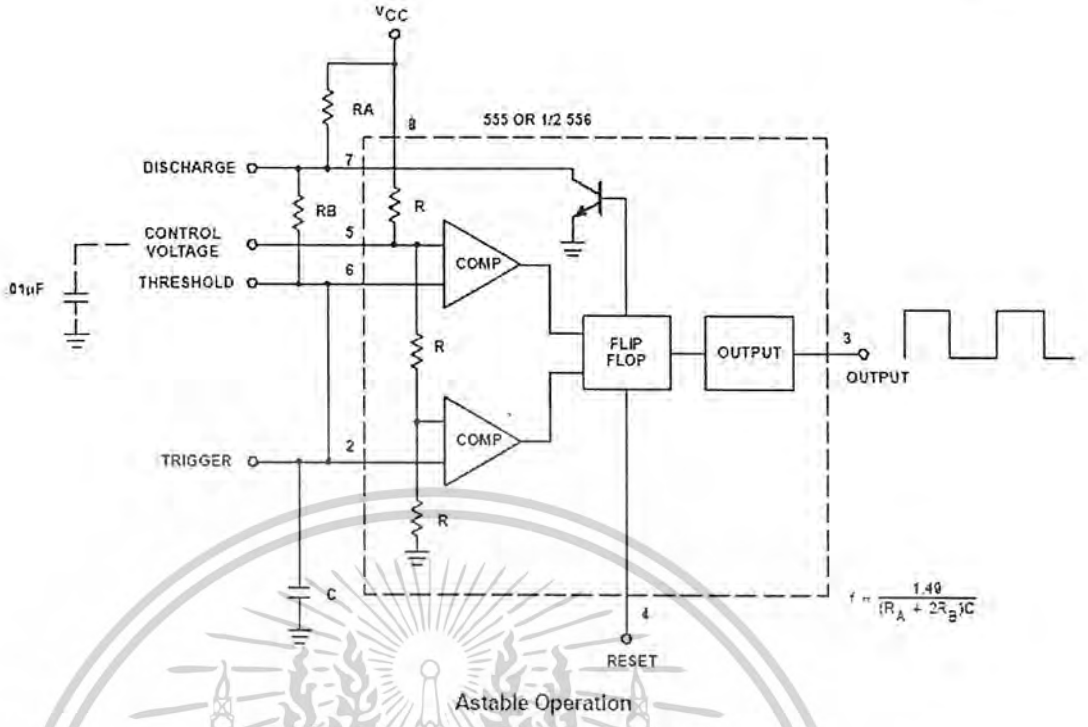
ใช้ ไอซีเบอร์ 74HC151 ทำการนับสัญญาณดิจิตอลขนาด4 บิต โดยวงจรมัลติเพล็กซ์จะต้องการการนับเพียง 3 บิต เท่านั้น จึงต้องใช้ขาสัญญาณ MSB ทำหน้าที่รีเซตวงจรนับเพื่อที่จะได้วงจรนับขนาด 3 บิต วงจรไบนารีเคาน์เตอร์ต้องการสัญญาณนาฬิกาจึงต้องมีวงจรกำเนิดสัญญาณนาฬิกา

3.1.3 วงจรออสซิลเลเตอร์



รูปที่ 3.4 แสดงสัญญาณของวงจรออสซิลเลเตอร์

โดยมีไอซีเบอร์ NE555 ทำหน้าที่ในการกำเนิดสัญญาณนาฬิกาเพื่อป้อนเข้าไปยังวงจรไบนารีเคาน์เตอร์



รูปที่ 3.5 แสดงวงจรของไอซีเบอร์NE555

การคำนวณค่าความถี่ของวงจร

กำหนดค่า $R_A = 2K\Omega, C = 0.1\mu F, f = 1000Hz$

$$f = \frac{1}{T} = \frac{1.443}{(R_A + 2R_B)C}$$

$$1000Hz = \frac{1.443}{(2K\Omega + 2R_B) \times 0.1\mu F}$$

$$R_B = 6.2K\Omega$$

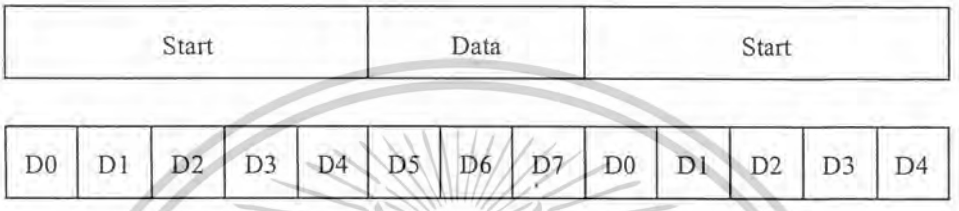
$$\text{Duty Cycle} = \frac{R_A + R_B}{R_A + 2R_B}$$

$$= \frac{2K\Omega + 6.2K\Omega}{2K\Omega + (2 \times 6.2K\Omega)}$$

$$= 0.57$$

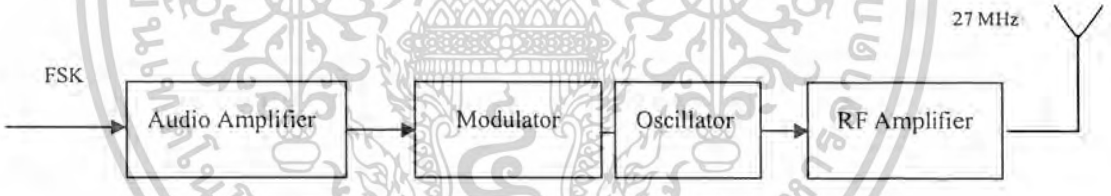
การทำงานโดยรวมของวงจรเข้ารหัส

มีดิปสวิตช์เป็นตัวกำหนดรหัสข้อมูลที่ต้องการซึ่งในรหัสข้อมูลที่ทำการกำเนิดขึ้นจะเป็นแบบอะซิงโครนัส ก็จะแบ่งข้อมูลขนาด 8 บิตออกเป็น 2 ส่วนคือ ส่วนที่เป็นสตาร์ทบิต และส่วนที่เป็นข้อมูล โดยส่วนสตาร์ทบิต จะมีบิตความยาว 5 บิต ส่วนข้อมูลจะมีความยาว 3 บิต เนื่องจากข้อมูลที่จะส่งไปมีข้อมูลน้อยกว่าสตาร์ทบิต แต่ถ้าเราเพิ่มบิตข้อมูลให้ยาวขึ้นหรือจำนวนบิตข้อมูลมากขึ้นจะทำให้เครื่องรับไม่สามารถตรวจสอบได้เพราะจะเกิดความผิดพลาดเกิดขึ้นในการรับข้อมูล แต่ละบิตจะสามารถกำหนดได้โดยดิปสวิตช์ทั้ง8ตำแหน่ง



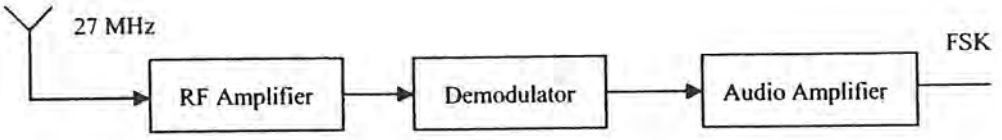
รูปที่ 3.6 แสดงบิตข้อมูลที่จะส่งทั้งหมด

3.2 วงจรเครื่องรับและเครื่องส่งเอฟเอ็ม



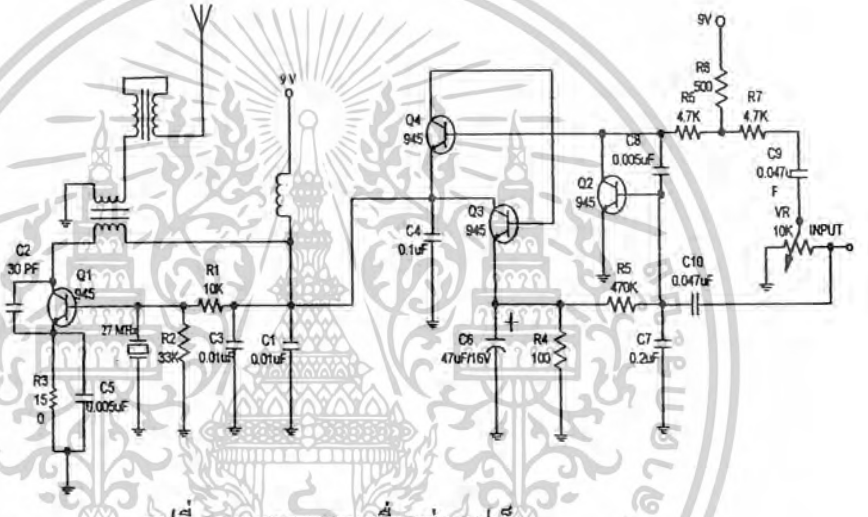
รูปที่ 3.7 บล็อกไดอะแกรมของเครื่องส่งเอฟเอ็ม

จากบล็อกไดอะแกรมแสดงหลักการทำงานในรูปที่ 3.7 คือสัญญาณข้อมูลเอฟเอสเคจะทำการเข้าไปขยายสัญญาณที่ภาคขยายสัญญาณ (Audio Amplifier) สัญญาณข้อมูลที่ออกจากภาคขยายจะถูกส่งเข้าไปมอดูเลตกับสัญญาณความถี่วิทยุ 27 MHz ซึ่งผลิตขึ้นจากภาคออสซิลเลเตอร์ที่ควบคุมความถี่ด้วยแร่คริสตอล 27 MHz หลังจากมอดดูเลตกับข้อมูลแล้วความถี่ 27 MHz จะเปลี่ยนไปเล็กน้อยตามการเปลี่ยนแปลงของสัญญาณข้อมูลที่เข้ามามอดูเลตด้วยซึ่งเป็นไปตามหลักของการมอดูเลตคลื่นสัญญาณในระบบเอฟเอ็ม ต่อจากนั้นจะถูกส่งต่อไปขยายให้กำลังแรงขึ้นที่ภาคขยายกำลังความถี่วิทยุ (RF Amplifier) ก่อนส่งสัญญาณไปออกสายอากาศ



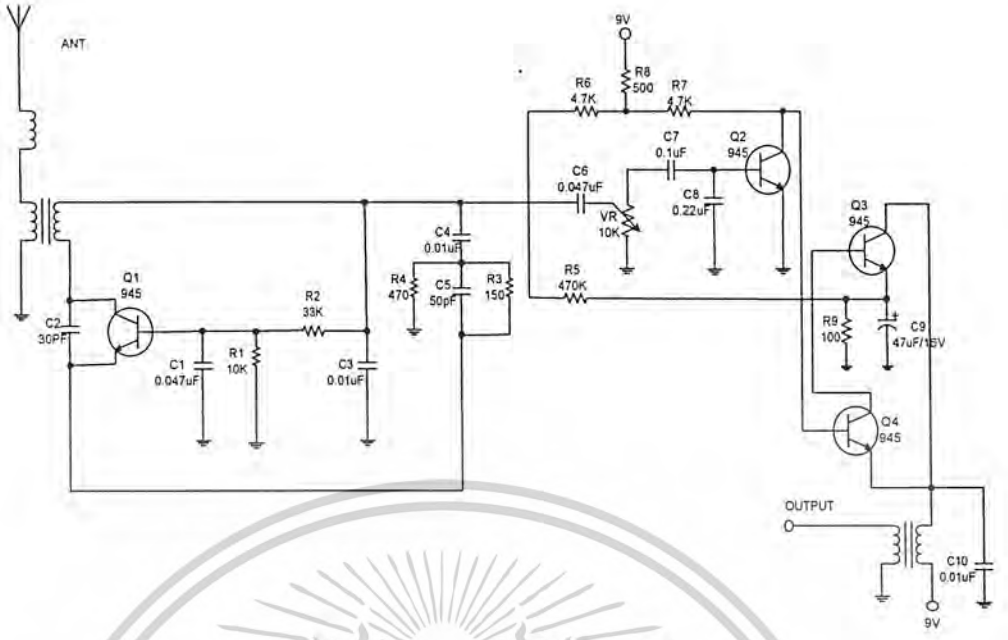
รูปที่ 3.8 บล็อกไดอะแกรมของเครื่องรับเอฟเอ็ม

สัญญาณที่รับได้จากสายอากาศจะป้อนเข้าสู่วงจรขยายกำลังความถี่วิทยุ (RF Amplifier) เพื่อขยายสัญญาณให้มีขนาดสูงขึ้นแล้วส่งต่อไปยังวงจรดีมอดูเลเตอร์เพื่อทำการจูนหาความถี่กลางของสัญญาณคลื่นพาห์เพื่อที่จะได้สัญญาณเอฟเอสเคออกมา แล้วก็จะนำไปขยายที่ส่วนของภาคขยายสัญญาณเพื่อให้ได้สัญญาณเอฟเอสเคกลับคืนมา



รูปที่ 3.9 แสดงวงจรเครื่องส่งเอฟเอ็ม

วงจรเครื่องส่งระบบเอฟเอ็มย่านความถี่ประมาณ 27MHz ทำหน้าที่ในการส่งข้อมูลจากวงจรเอฟเอสเคโดยส่งเข้ามายังอินพุตของวงจรเครื่องส่ง ซึ่งจะทำหน้าที่ในการกระจายสัญญาณออกอากาศไปยังเครื่องรับที่อยู่ที่ป้ายรถประจำทาง โดยมีทรานซิสเตอร์Q2 ทำหน้าที่ขยายสัญญาณให้มีความแรงป้อนไปยัง Q3และQ4 ทำหน้าที่ขยายอีกต่อหนึ่ง ส่งไปยังภาคมอดูเลเตอร์โดยมีคริสตอลเป็นออสซิลเลเตอร์ขนาด 27MHz สัญญาณข้อมูลจะถูกผสมกับสัญญาณวิทยุเพื่อส่งออกอากาศผ่านทางสายอากาศ



รูปที่ 3.10 แสดงวงจรเครื่องรับเอฟเอ็ม

จากรูปที่3.10 แสดงวงจรรับเอฟเอ็ม โดยจะรับสัญญาณมาจากสายอากาศ โดยมีQ1ทำงานร่วมกับขดลวดL1ขยายสัญญาณให้แรงขึ้นแล้วทำการดีมอดูเลเตอร์ได้สัญญาณข้อมูลที่ต้องการ โดยผ่านวงจรขยายของทรานซิสเตอร์ Q2-Q4 มีหม้อแปลง T1 ทำหน้าที่ส่งผ่านสัญญาณข้อมูลออกส่งไปยังวงจรเอฟเอสเคต่อไป

3.3 การทำงานของเอฟเอสเคมอดูเลเตอร์และดีมอดูเลเตอร์

ในการใช้งานเอฟเอสเคมอดูเลเตอร์จะใช้ไอซีเบอร์ XR 2206 ส่วนเอฟเอสเคมอดูเลเตอร์จะใช้ไอซีเบอร์ XR 2206 การทำงานและการออกแบบสามรถอธิบายได้ดังนี้

3.3.1 เอฟเอสเคมอดูเลเตอร์

เราใช้ไอซีเบอร์ XR 2206 ซึ่งเป็น Monolithic Function Generator ทำหน้าที่ในการมอดูเลตสัญญาณดิจิตอล ซึ่งสามารถใช้ได้กับอินพุทที่เป็นทั้ง TTL หรือ CMOS ได้ สามารถที่จะให้ค่าคลื่นรูปซายน์เอาท์พุทที่ได้ประมาณ 3 V_{p-p} และค่าความผิดเพี้ยนของสัญญาณที่เกิดขึ้นจะอยู่ระหว่าง 0.5%ถึง 2.5%

วงจรภายในของ XR 2206 จะประกอบด้วย 4 วงจรสามารถอธิบายเป็นบล็อกไดอะแกรมได้ดังรูปที่ 3.11 จะประกอบด้วย วงจรโวลท์เตจคอนโทรลออสซิลเลเตอร์ (VCO), วงจรคูณอนาลอก, วงจรปรับสัญญาณซายน์, บัฟเฟอร์และสวิทช์กระแส VCO จะผลิตเอาท์พุทความถี่ที่เป็นสัดส่วนกับกระแสอินพุทซึ่งสมารถกำหนดได้โดยตัวต้านทานที่ต่อเข้าที่ขาที่เกี่ยวข้องเวลาลงกราวด์ เนื่องจากมี 2 ขาที่เกี่ยวข้องกับเวลา ดังนั้นจึงสามารถสร้างเวลาได้ 2 ความถี่สำหรับการกำเนิดสัญญาณเอฟเอสเค โดยใช้ใช้ขา FSK Input ไปควบคุมค่าภายในส่วนของ Current Switches เพื่อให้เกิดการเลือกขาที่ตัวต่อต้านทานสำหรับวงจรโวลท์เตจคอนโทรลออสซิลเลเตอร์

วงจรที่ใช้งานแสดงดังรูปที่ 3.11 โดยนำสัญญาณจากวงจรเข้ารหัสมาเข้าที่ขา 9 ของ XR 2206 ระดับศักดาไฟฟ้าตรงของเอฟเอสเคเอาท์พุทที่ขา 2 ประมาณเท่ากับศักดาไฟฟ้าตรงที่ไบอัสเข้าที่ขา 3 ของ XR 2206 ซึ่งจากรูปที่ 3.11 จะพบว่า ขา 3 ถูกไบอัสด้วยครึ่งหนึ่งของค่า Vcc ดังนั้นจะได้ค่าศักดาไฟฟ้าตรงที่เอาท์พุทจะเท่ากับ $\frac{Vcc}{2}$ เราสามารถปรับค่าศักดาไฟฟ้าตรงที่เอาท์พุทได้โดยใช้ VR 2

ความถี่ในกรณีที่สัญญาณดิจิตอลเป็นรหัส "1" (Mark) หาได้จาก

กำหนดค่า $f_1 = 1000Hz, C = 0.01\mu F$

$$R_1 = \frac{1}{f_1 C} = \frac{1}{1000Hz \times 0.01\mu F}$$

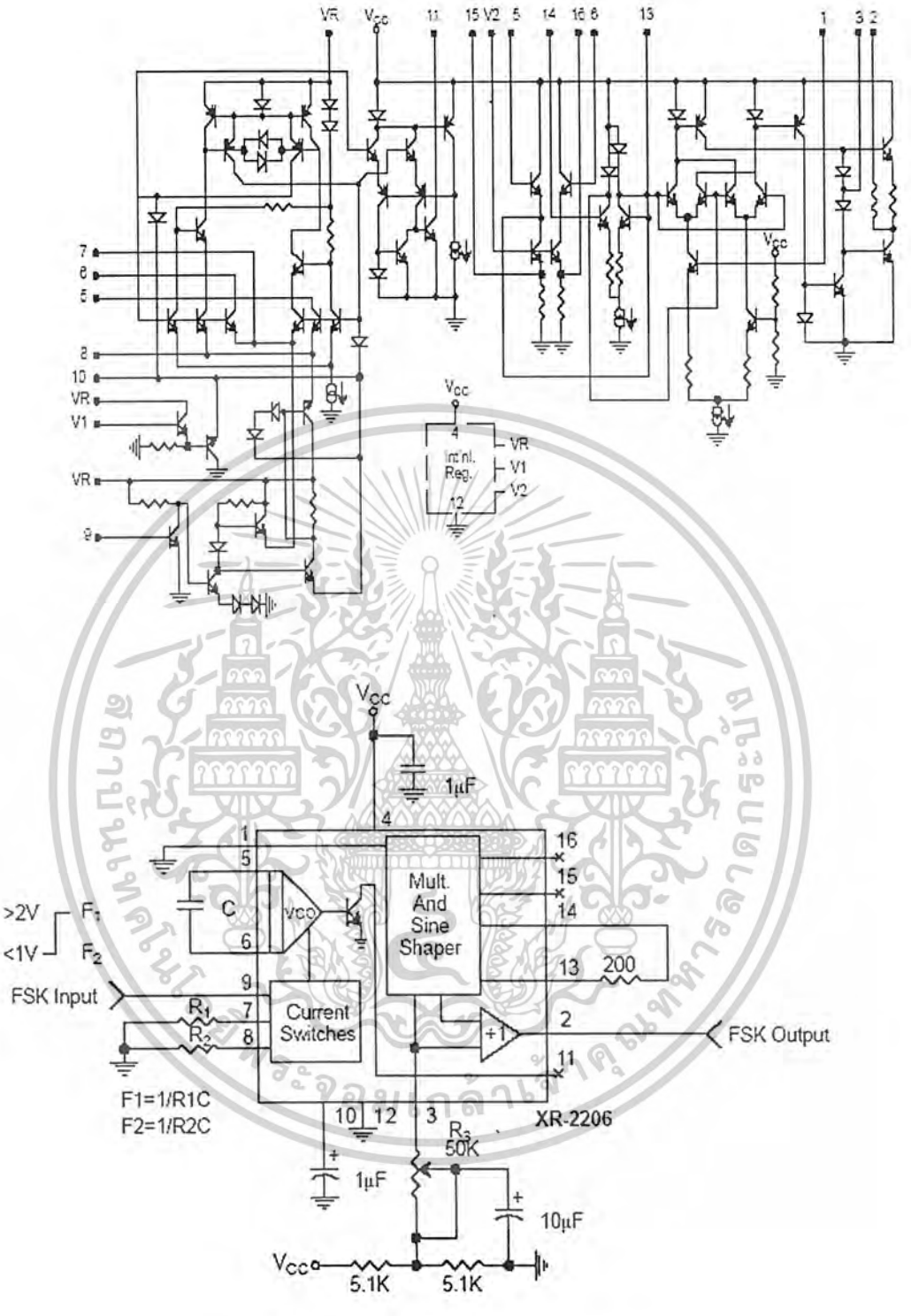
$$R_1 = 100K\Omega$$

ความถี่ในกรณีที่สัญญาณดิจิตอลเป็นรหัส "0" (Space) หาได้จาก

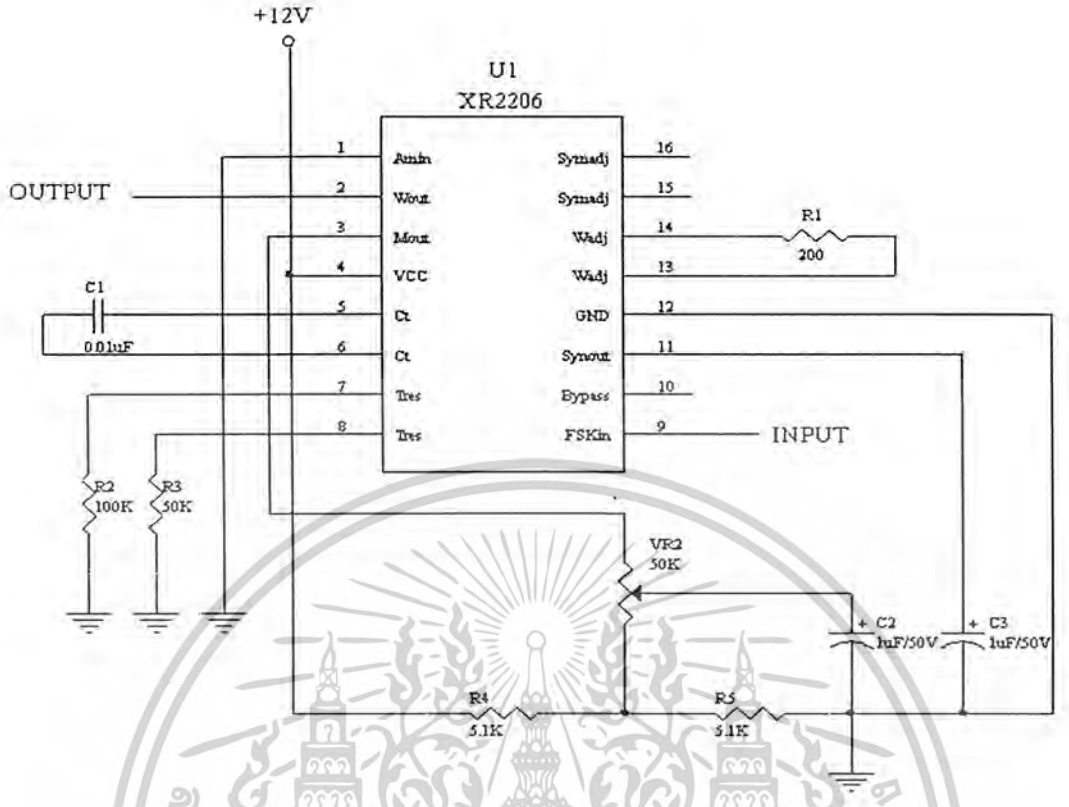
กำหนดค่า $f_2 = 2000Hz, C = 0.01\mu F$

$$R_2 = \frac{1}{f_2 C} = \frac{1}{2000Hz \times 0.01\mu F}$$

$$R_2 = 50K\Omega$$



รูปที่ 3.11 แสดงวงจรภายในของ XR2206

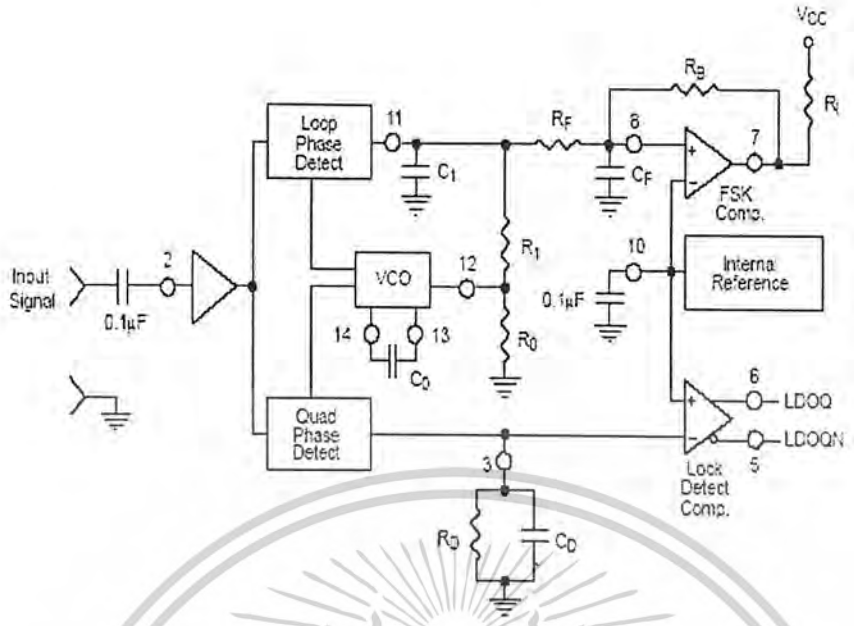


รูปที่ 3.12 วงจรเอฟเอสเคมอดูเลเตอร์โดยใช้ ไอซีเบอร์ XR 2206

3.3.2 เอฟเอสเคดีมอดูเลเตอร์

เอฟเอสเคดีมอดูเลเตอร์จะใช้ไอซีเบอร์ XR 2211 เป็นตัวดีมอดูเลตสัญญาณ ไอซีเบอร์ XR 2211 มีลักษณะทั่วไปก็คือเป็นวงจรเฟสล็อกลูปแบบโมโนลิทิก ทำงานในช่วงกว้างของไฟเลี้ยง 4.5 V ถึง 20 V และมีความกว้างโดยอยู่ในช่วง 0.01 Hz – 300 KHz สามารถใช้สัญญาณอนาลอกได้ในช่วง 2 mV – 3V

วงจรภายในของ ไอซีเบอร์ XR 2211 แสดงดังรูปที่ 3.13 โดยมีการทำงานดังต่อไปนี้ เอาท์พุทของเฟสดีเทกเตอร์ให้สัญญาณผลบวกและผลต่างความถี่ของสัญญาณอินพุทกับสัญญาณจาก โวลท์เตจคอนโทรลออสซิลเลเตอร์ นั่นก็คือความถี่เอาท์พุทจะเป็น $f_{in} + f_{vco}$ และ $f_{in} - f_{vco}$ ดังนั้นในขณะที่ล็อกความถี่สัญญาณเอาท์พุทจะเป็น $2f_{in}$ และ 0 Hz โดยการต่อตัวเก็บประจุคร่อมสัญญาณเอาท์พุทของเฟสดีเทกเตอร์ก็คือใส่วงจรกรองความถี่ต่ำนั่นเอง จะทำให้สัญญาณความถี่สูง ($f_{in} + f_{vco}$) ถูกลดทอนหายไปทำให้เหลือเพียงศักดากระแสตรงที่เกิดจากความต่างเฟสของความถี่ทั้งสอง จะเกิดเช่นนี้ไปเรื่อยๆ ทำให้เกิดโวลท์เตจคอนโทรลออสซิลเลเตอร์ติดตามความถี่ของสัญญาณอินพุทได้ ในส่วนที่เหลือของ XR 2211 ทำงานดังนี้คือ หากโวลท์เตจคอนโทรลออสซิลเลเตอร์ ถูกขับเคลื่อนด้วยความถี่ที่เหนือกว่าหรือต่ำกว่าความถี่ศูนย์กลางแล้ว วงจรเปรียบเทียบแรงดันจะสร้างสัญญาณเอาท์พุทลอจิกสูง และสัญญาณเอาท์พุทลอจิกต่ำเมื่อเฟสล็อกลูปอยู่ในช่วงล็อก



รูปที่ 3.13 แสดงวงจรภายในของ XR 2211

วงจรการใช้งานจริงแสดงดังรูปที่ 3.14 โดย R3 และ C3 จะกำหนดศูนย์กลางของของเฟสล็อกลูป R3 เป็นตัวกำหนดแถบความถี่ของระบบ C5และC6 จะกำหนดค่าเวลาคงตัวในวงจรกรองของวงรอบ C4 และ R6 กำหนด one- pole post-detection filter สำหรับข้อมูลเอาท์พุทของสัญญาณ FSK ตัวต้านทาน R4 จากขา 7 ไปขา 8 มีไว้เพื่อเป็นตัวป้อนกลับแบบบวกสำหรับตัวเปรียบเทียบแรงดันของสัญญาณ FSK เพื่อทำให้เกิดความรวดเร็วในการเปรียบเทียบสถานะของลอจิก

ความถี่มาร์คและสเปซกำหนดไว้คือ $f_1 = 1000Hz$ และ $f_2 = 2000Hz$

ขั้นที่ 1 คำนวณหาค่า f_0

$$f_0 = \sqrt{f_1 \times f_2} = \sqrt{1000 \times 2000}$$

$$= 1414.21Hz$$

ขั้นที่ 2 คำนวณหาค่า R_T

กำหนด $R_o = 70K\Omega$

$$R_T = R_0 + \frac{R_x}{2} = 70K\Omega + \frac{10K\Omega}{2}$$

$$= 75K\Omega$$

ขั้นที่ 3 คำนวณหาค่า C_0

$$C_0 = \frac{1}{R_0 \times f_0} = \frac{1}{70K\Omega \times 1414.21Hz}$$

$$= 10.1nF$$

ขั้นที่ 4 คำนวณหาค่า R_1

$$R_1 = \frac{R_0 \times f_0}{f_1 - f_2} \times 2 = \frac{70K\Omega \times 1414.21Hz}{2000Hz - 1000Hz}$$

$$= 98.99K\Omega$$

ขั้นที่ 5 คำนวณหาค่า C_1

$$C_1 = \frac{1250 \times C_0}{R_1 \times \zeta^2} = \frac{1250 \times 10.1nF}{98.99K\Omega \times 0.5^2}$$

$$= 0.51nF$$

ขั้นที่ 6 คำนวณหาค่า R_F โดย R_F ต้องมีค่าอย่างน้อย 5 เท่าของ R_1

$$R_F = R_1 \times 5 = 98.99K\Omega \times 5$$

$$= 494.95K\Omega$$

ขั้นที่ 7 คำนวณหาค่า R_B โดย R_B ต้องมีค่าอย่างน้อย 5 เท่าของ R_F

$$R_B = R_F \times 5 = 494.95K\Omega \times 5$$

$$= 2.47M\Omega$$

ขั้นที่ 8 คำนวณหาค่า R_{sum}

$$R_{sum} = \frac{(R_F + R_1) \times R_B}{R_1 + R_F + R_B}$$

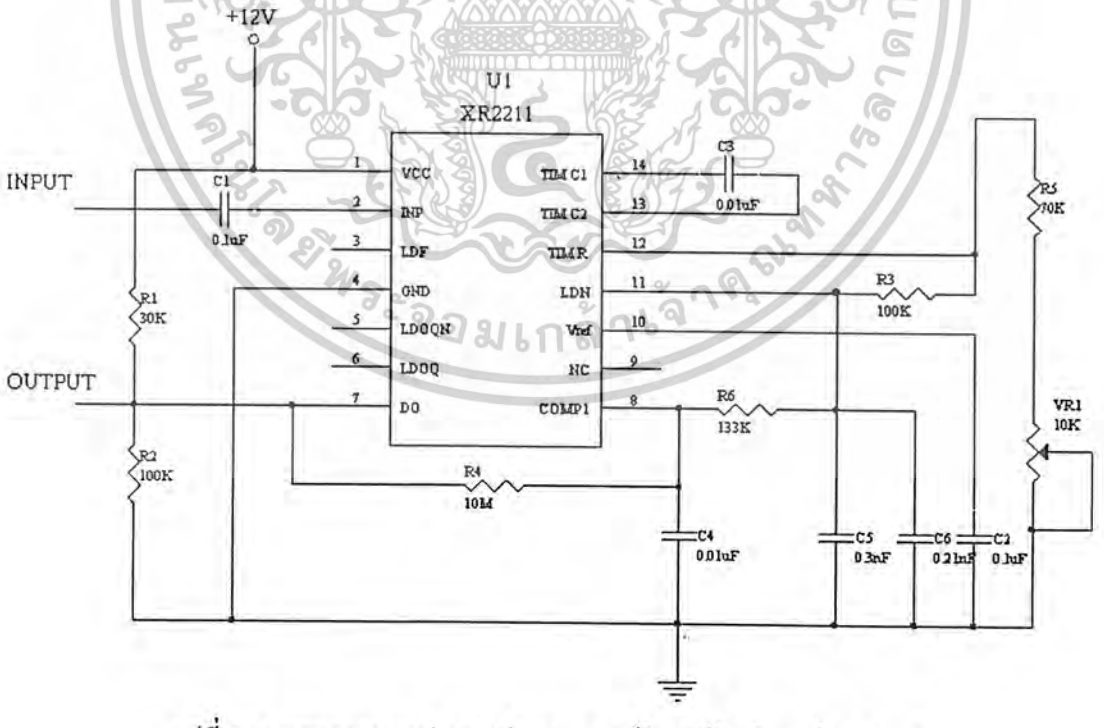
$$= \frac{(494.95K\Omega + 98.99K\Omega) \times 2.47M\Omega}{98.99K\Omega + 494.95K\Omega + 2.47M\Omega}$$

$$= 478.80K\Omega$$

ขั้นที่ 9 คำนวณหาค่า C_F

$$C_F = \frac{0.25}{R_{sum} \times Baud \quad Rate} = \frac{0.25}{478.80K\Omega \times 1000}$$

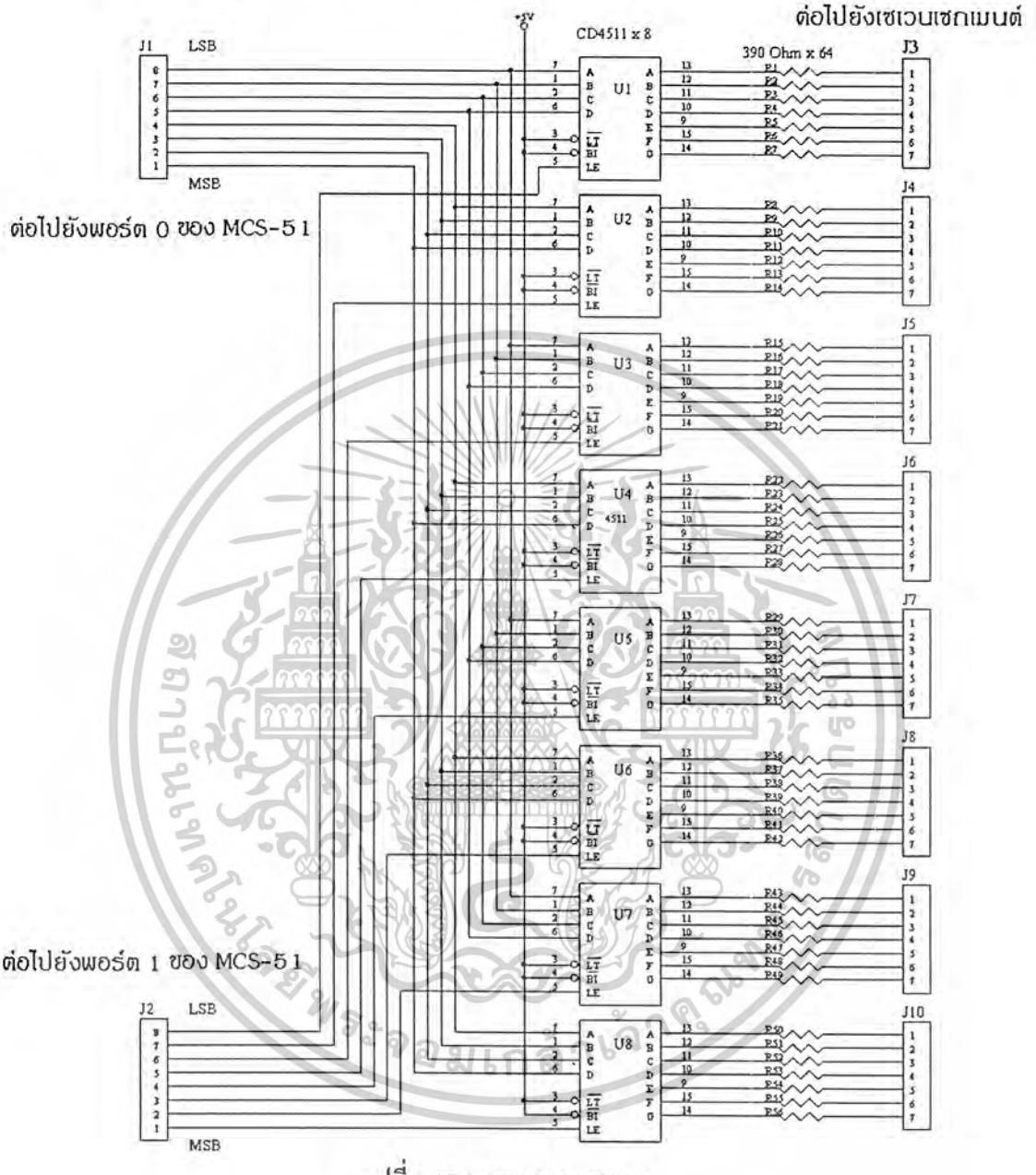
$$= 522.14pF$$



รูปที่ 3.14 แสดงวงจรเอฟเอสเคดีมอดูเลเตอร์โดยใช้ ไอซีเบอร์ XR 2211

3.4 การออกแบบบอร์ดแสดงผล

ใช้แสดงผลข้อมูลต่างๆที่ส่งมาจากพอร์ตของไมโครคอนโทรลเลอร์

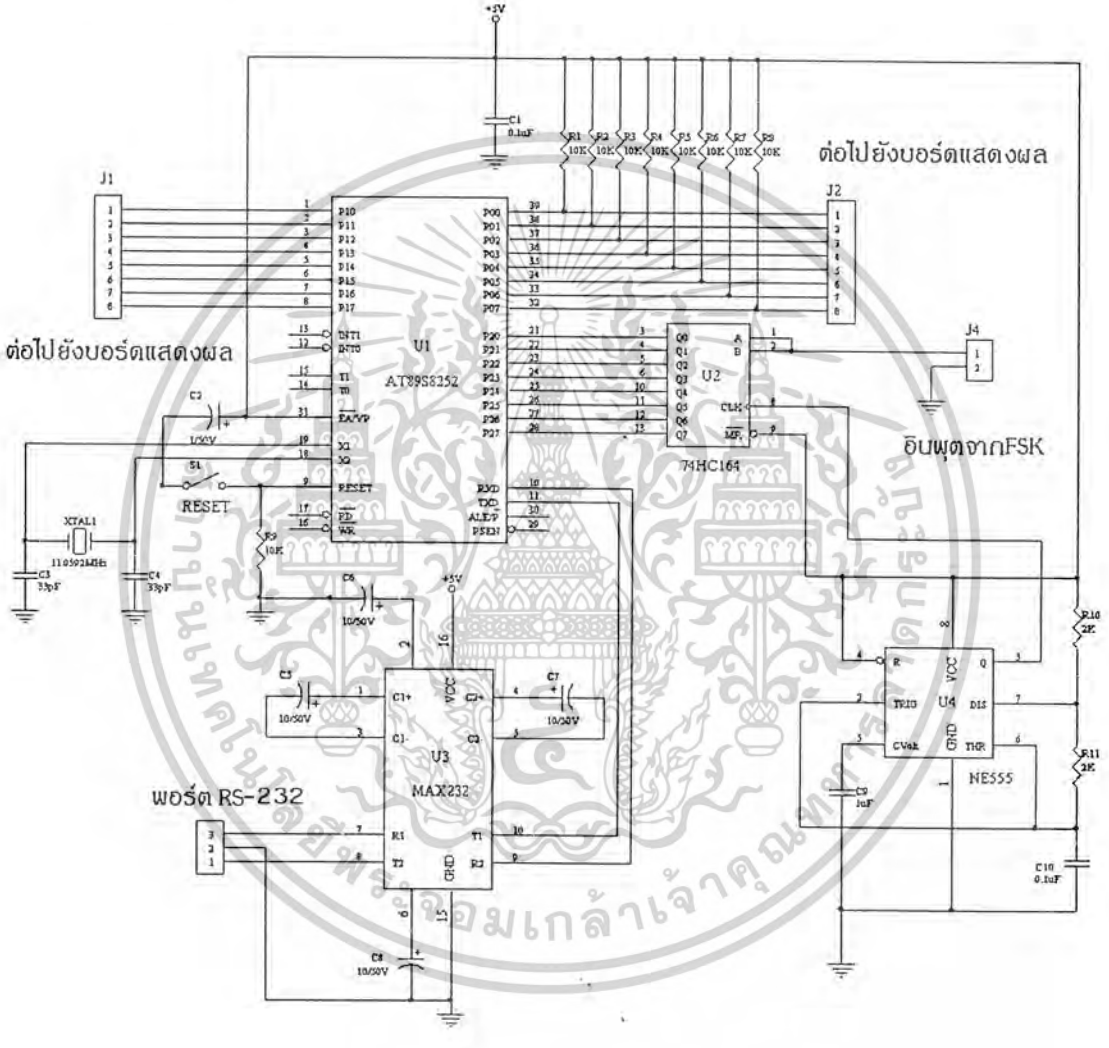


รูปที่ 3.15 แสดงวงจรแสดงผล

ในส่วนของวงจรนี้จะทำหน้าที่ในการแปลงรหัส BCD ไปแสดงผลยังเซเวนเซกเมนต์ โดยมีไอซีเบอร์ CD4511 ซึ่งทำหน้าที่ในการแปลงรหัสโดยมีวงจรไดรเวอร์และวงจรแลทช์อยู่ภายในไอซีเบอร์นี้ซึ่งในบอร์ดแสดงผล 1 ชิ้น จะมีเซเวนเซกเมนต์อยู่ 2 แถว แถวละ 4 ตัว ซึ่งแต่ละแถวจะแสดงหมายเลขรถประจำทาง ระยะเวลาเฉลี่ยที่รถประจำทางจะมาถึง และอีกกี่ป้ายรถประจำทางจะมาถึงป้ายที่ต้องการรอ

3.5 การออกแบบวงจรในส่วนของไมโครคอนโทรลเลอร์

การทำงานในส่วนนี้ได้ใช้ไมโครคอนโทรลเลอร์ตระกูล MCS-51 ซึ่งมีพอร์ตใช้งานอยู่4พอร์ต พอร์ต 0 และพอร์ต 1 ทำหน้าที่ในการแสดงผล พอร์ต 2 ทำหน้าที่ในการรับสัญญาณรหัสรถประจำทางจากวงจรเอฟเอสเคดีมอดูเลเตอร์ ส่วนพอร์ต 3 ทำหน้าที่ในการติดต่อระหว่างไมโครคอนโทรลเลอร์และเซิร์ฟเวอร์ของคอมพิวเตอร์



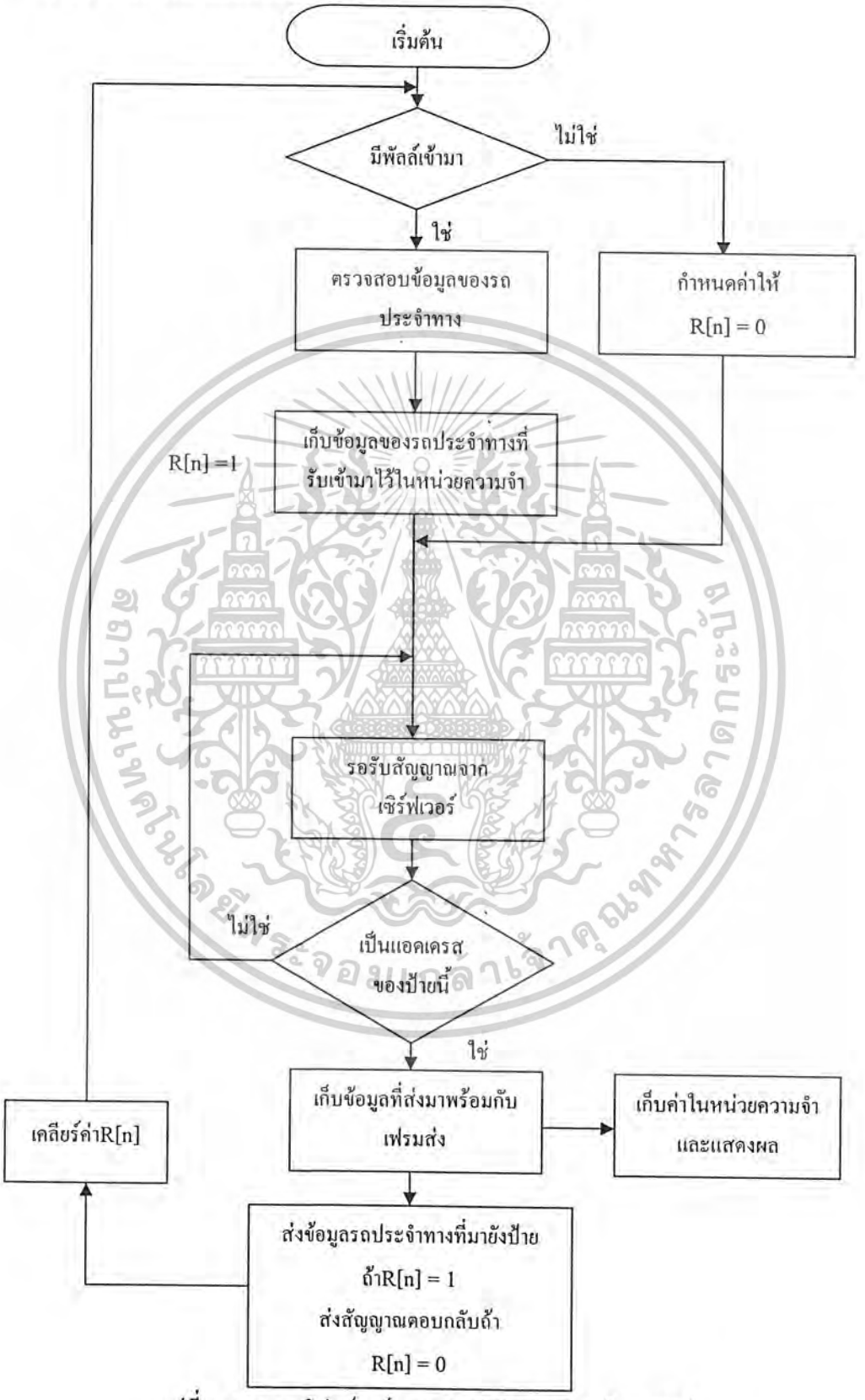
รูปที่ 3.16 แสดงวงจรไมโครคอนโทรลเลอร์

ในวงจรนี้ประกอบไปด้วยวงจรกำเนิดสัญญาณนาฬิกานำสัญญาณที่ได้ไปควบคุมการทำงานของวงจร SIPO (Serial Input Parallel Output) โดยใช้ไอซีเบอร์ 74HC164 ซึ่งอินพุตของวงจรได้มาจากวงจรเอฟเอสเคดีมอดูเลเตอร์ซึ่งเป็นข้อมูลแบบอนุกรมซึ่งมีความถี่ในการส่งคือประมาณ 1 กิโลเฮิรตซ์ ซึ่งเอาต์พุตของวงจรเอสไอพีโอจะเปลี่ยนแปลงตามอินพุต ทุกๆ 1 มิลลิวินาที ซึ่งการตรวจจับของ

ไมโครคอนโทลเลอร์ก็จะทำการดีเลย์เป็นเวลาประมาณ 1 มิลลิวินาทีเช่นกัน โดยจะตรวจสอบแบบวนไปเรื่อยๆ จนเจอรหัสที่เรากำหนดไว้ตรงกันก็จะทำการเก็บข้อมูลไว้ว่ารถประจำทางสายใดเข้ามาบ้างถ้าไม่มีรถประจำทางสายใดเข้ามา ก็จะเก็บข้อมูลไว้ว่าไม่มีรถประจำทางสายใดๆเข้ามายังป้ายรถประจำทาง ข้อมูลที่รับและส่งไปยังคอมพิวเตอร์ ในส่วนของเซิร์ฟเวอร์จะทำการส่งผ่านวงจรแปลงแรงดันโดยมีไอซีเบอร์ MAX-232 ทำหน้าที่ในการเปลี่ยนแปลงระดับแรงดันที่สามารถติดต่อกันได้

3.6 การออกแบบโปรแกรม

3.6.1 การทำงานในส่วนของไคลเอนต์

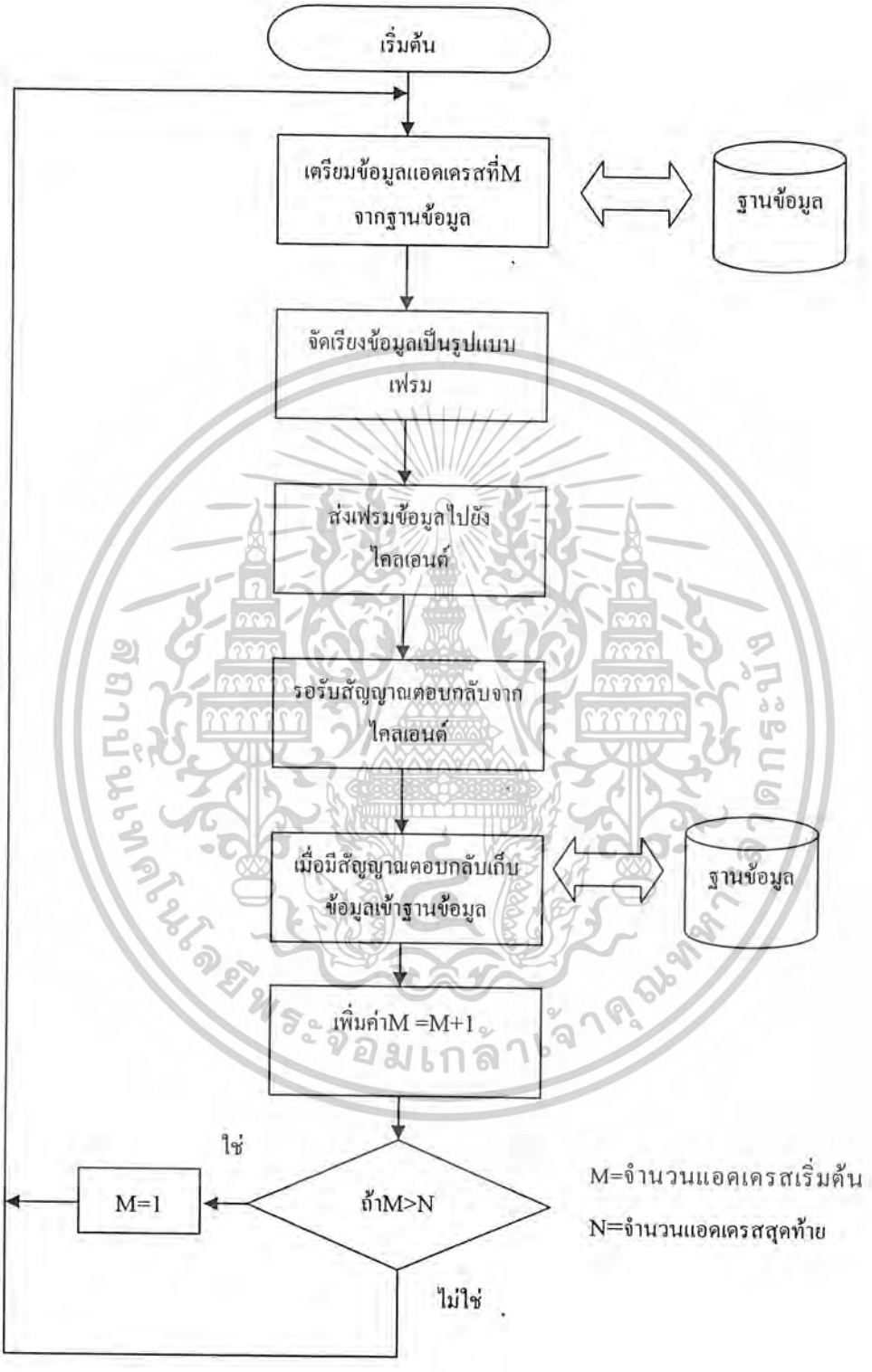


รูปที่ 3.17 แสดงโฟลว์ชาร์ตการทำงานในส่วนของไคลเอนต์

จากรูปที่ 3.17 แสดงการทำงานของโปรแกรมทางด้านไคลเอนต์ ซึ่งทำหน้าที่ตรวจสอบพัลส์ที่เข้ามาจากวงจรเพื่อตรวจสอบว่าสายรถประจำทางทางสายใดเข้ามายังป้ายแล้วเก็บข้อมูล ถ้ามีสัญญาณตรวจสอบจากเซิร์ฟเวอร์เข้ามาและทำการตรวจสอบว่าเป็นแอดเดรสของป้ายนี้หรือไม่ถ้าใช่ก็จะนำข้อมูลที่ส่งมาเก็บค่าไว้แล้วแสดงผลออกทางบอร์ดแสดงผล เมื่อรับข้อมูลเสร็จก็จะส่งสัญญาณตอบกลับโดยส่งข้อมูลของรถประจำทาง (B[n]) ที่มายังป้ายนี้ไปพร้อมกันด้วย

ทำการเขียนโปรแกรมเพื่อป้อนค่ารหัสHEXเข้าไมโครคอนโทรลเลอร์โดยใช้โปรแกรมRAD51 ในการคอมไพล์ภาษาแอสเซมบลีให้เป็นรหัสHEX Codeซึ่งอินพุตของพอร์ต 0 ของไมโครคอนโทรลเลอร์ทำหน้าที่ในการรับสัญญาณรหัสรถประจำทาง เพื่อตรวจสอบว่าเป็นรหัสของรถประจำทางคันใดที่เข้ามายังป้ายรถประจำทางซึ่งในโปรเจคนี้กำหนดรถไว้เพียง 3 คันเพื่อใช้ในการทดสอบแต่สามารถเพิ่มเติมได้เรื่อยๆในกรณีที่ต้องการตรวจสอบรถประจำทางจำนวนมากๆได้ แล้วเก็บค่านั้นไว้เพื่อส่งต่อไปยังเครื่องเซิร์ฟเวอร์ การแสดงผลจะใช้พอร์ต 1 (ข้อมูล) และพอร์ต 2 (แถวของหน่วยแสดงผล) ในการส่งข้อมูลต่างๆเพื่อส่งให้บอร์ดแสดงผล

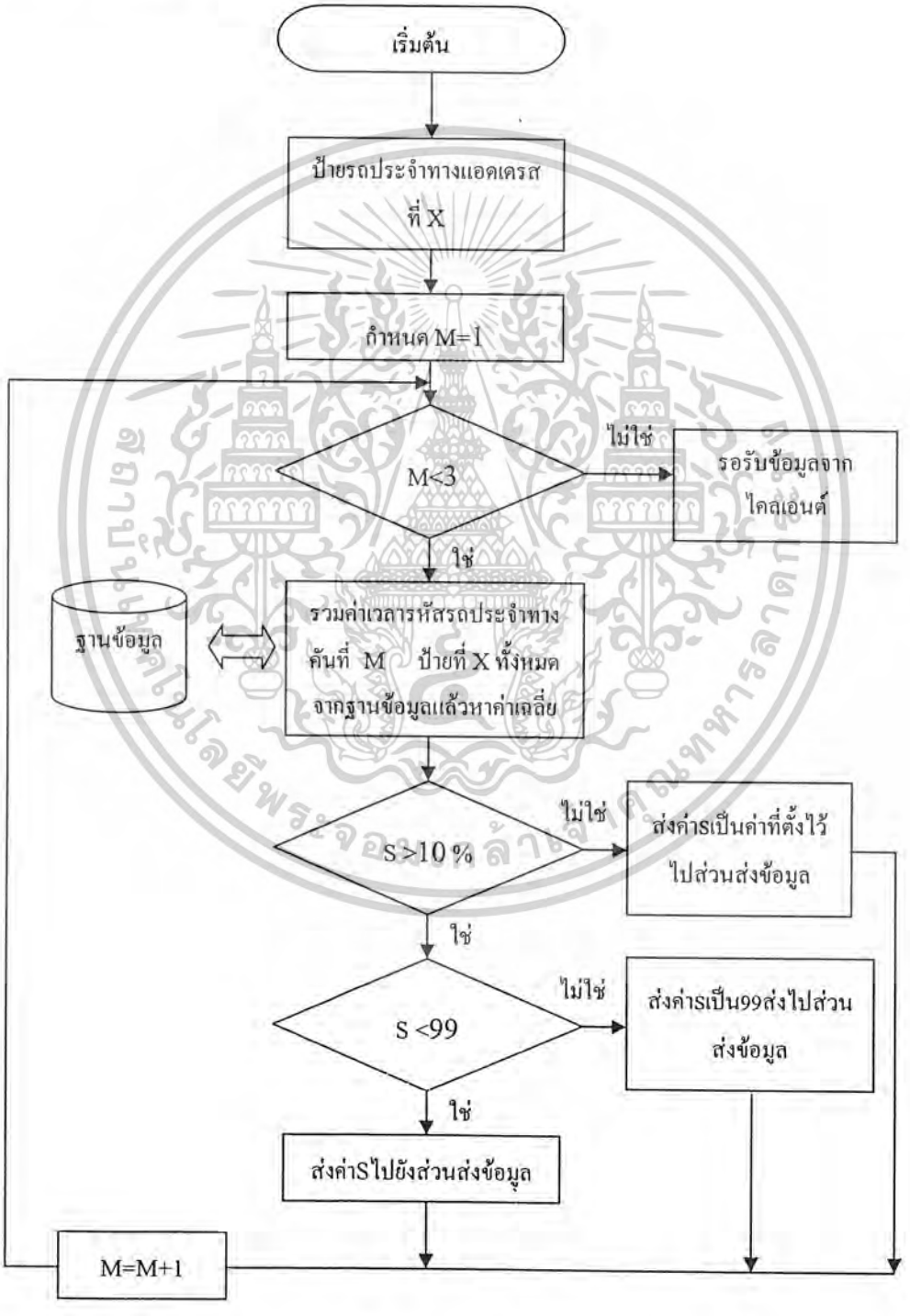
3.6.2 การทำงานในส่วนของเซิร์ฟเวอร์คอมพิวเตอร์



รูปที่ 3.18 แสดงโฟลว์ชาร์ตการทำงานในส่วนของคอมพิวเตอร์ (เซิร์ฟเวอร์)

จากรูปที่ 3.18 แสดงการทำงานในส่วนของคอมพิวเตอร์ขั้นตอนแรกคอมพิวเตอร์จะคำนวณค่าต่างๆที่เก็บอยู่ในฐานข้อมูลเพื่อทำการส่งไปยังเซิร์ฟเวอร์จะเริ่มต้นที่แอดเดรสแรกก่อนแล้ว จึงทำการส่งข้อมูลต่อไปจนครบจึงวนกลับมาส่งแอดเดรสแรกเมื่อข้อมูลพร้อมแล้ว ก็ทำการส่งออกไปแล้วก็จะรอสัญญาณตอบกลับจากเซิร์ฟเวอร์ เมื่อมีการตอบกลับก็จะทำการเก็บข้อมูลที่ได้จากเซิร์ฟเวอร์เข้าไปยังฐานข้อมูลแล้วนำไปประมวลผลเพื่อส่งค่าแอดเดรสต่อไป

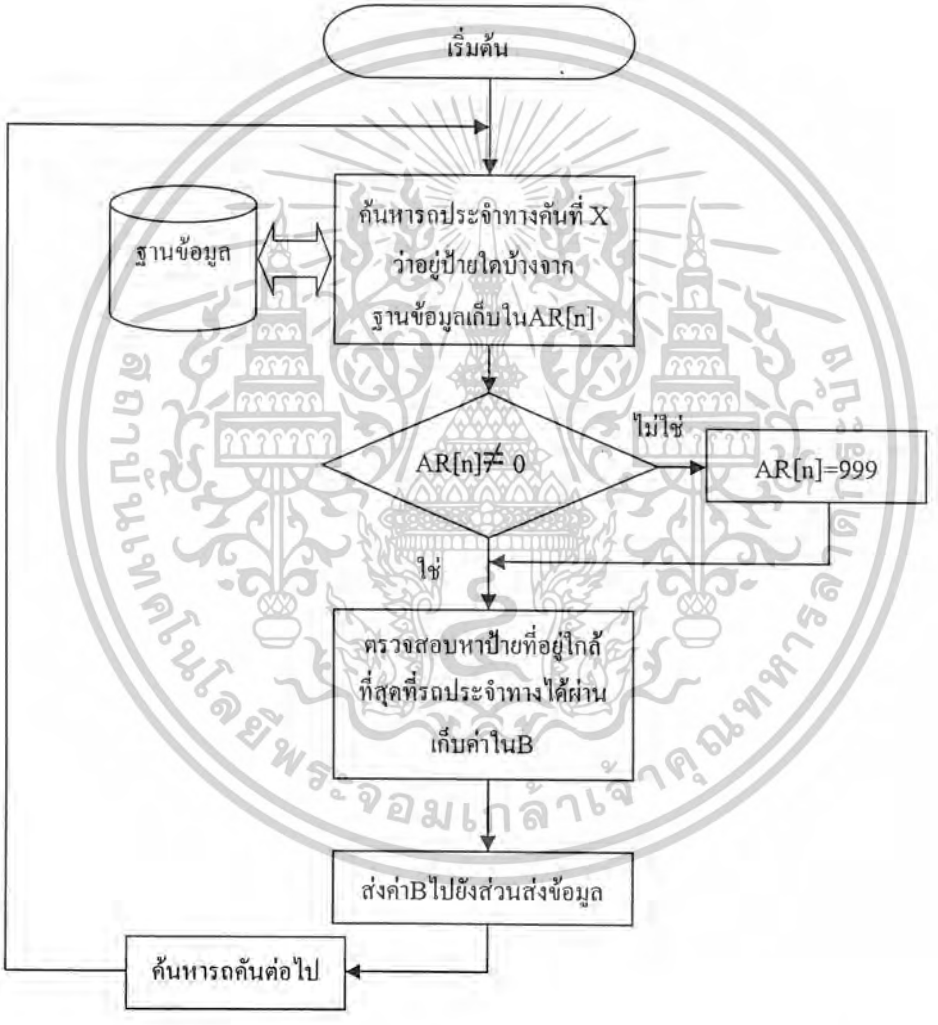
3.6.3 การทำงานในส่วนของการประมาณหาค่าเวลาเฉลี่ย



รูปที่ 3.19 แสดงโฟลว์ชาร์ตการทำงานในส่วนของการประมาณหาค่าเวลาเฉลี่ย

โดยจะทำหน้าที่ในการหาค่าเวลาเฉลี่ยทั้งหมดของรถประจำทางทุกสายที่ผ่านมายังป้ายนั้นๆ ถ้าหากระยะเวลาเฉลี่ยที่ได้คำนวณไว้มีค่าเกินกว่าเวลาปรกติที่เคยผ่านในช่วง10เปอร์เซ็นต์ของเวลาปรกติก็จะทำการนำค่าเวลาปรกตินั้นออกไปแสดงผลหรือถ้าหากค่าเวลาเฉลี่ยเกินกว่าที่บอร์ดแสดงผลคือเลขสองหลักโปรแกรมจะทำการแสดงผลสูงสุดที่99นาที การทำงานในส่วนนี้ข้อมูลทั้งหมดที่ได้มา ส่งไปยังหน่วยส่งข้อมูลซึ่งผ่านพอร์ต RS232 โดยข้อมูลที่ส่งไปอยู่ในรูปแบบแอสกีส่งด้วยความเร็วบอดเรต 9600 บิตต่อวินาที

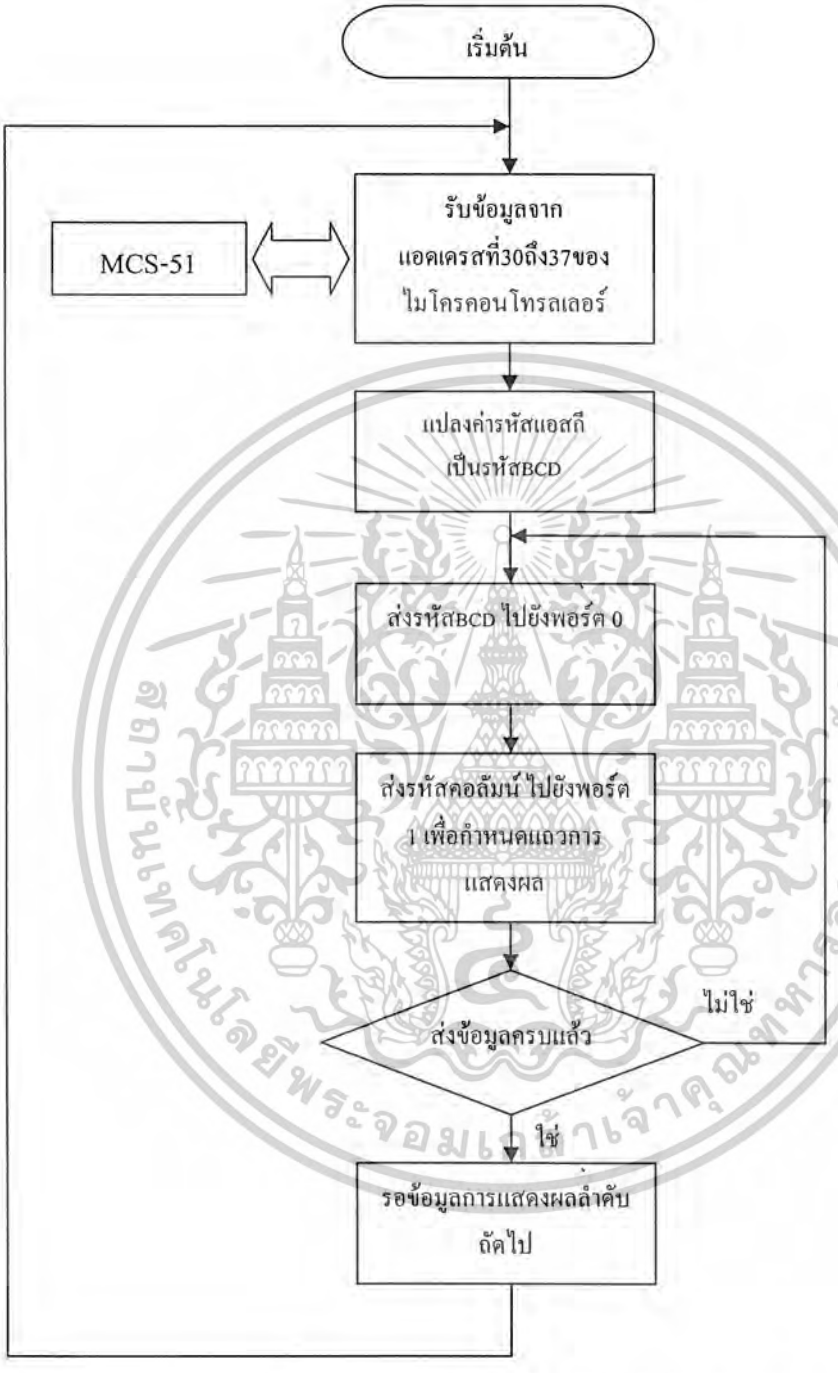
3.6.4 การทำงานในส่วนของการค้นหารถประจำทางที่ต้องการรอใกล้สุดอยู่อีกกี่ป้าย



รูปที่ 3.20 แสดงการทำงานในส่วนของการค้นหารถประจำทางที่ต้องการรอใกล้สุดอยู่อีกกี่ป้าย

โปรแกรมจะทำการสืบค้นข้อมูลต่างๆจากฐานข้อมูล MySQL ทำการค้นหาข้อมูลรถประจำทางสายนั้นๆว่าในช่วงเวลานี้สายรถประจำทางได้อยู่ห่างจากป้ายที่รออยู่ถัดไปอีกกี่ป้ายซึ่งจะทำการตรวจสอบหาป้ายที่ใกล้ที่สุดที่รถประจำทางได้ผ่านล่าสุด

3.6.5 การทำงานในส่วนของการแสดงผล



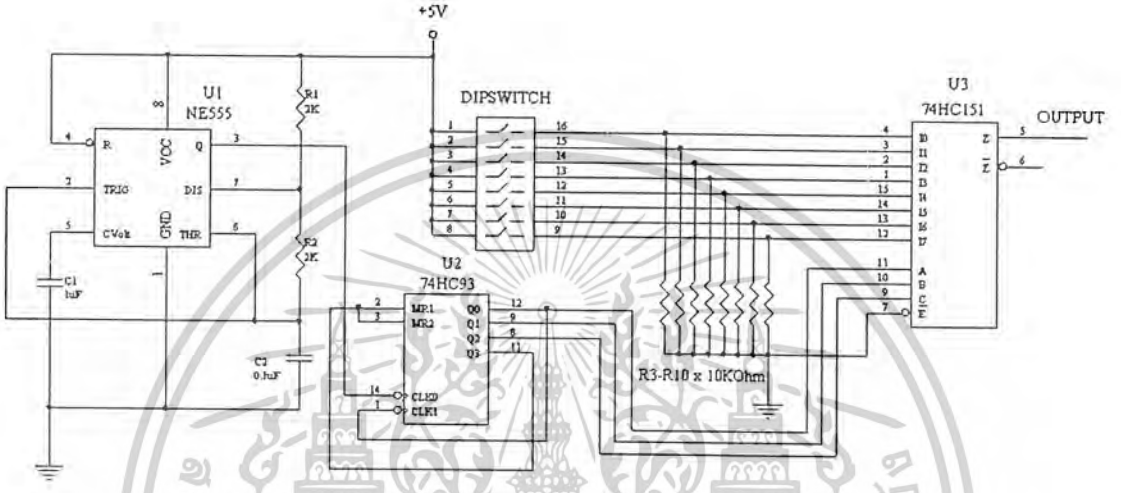
รูปที่ 3.21 แสดงการทำงานในส่วนของการแสดงผล

การทำงานงานในส่วนนี้จะรับข้อมูลมาจากไมโครคอนโทรลเลอร์แปลงค่ารหัสแอสกีเป็นรหัส BCD เพื่อนำข้อมูลมาจากไมโครคอนโทรลเลอร์นำไปแสดงผลโดยนำข้อมูลออกที่พอร์ต 0 และพอร์ต 1

บทที่ 4
การทดลองและผลการทดลอง

4.1 การทดลอง

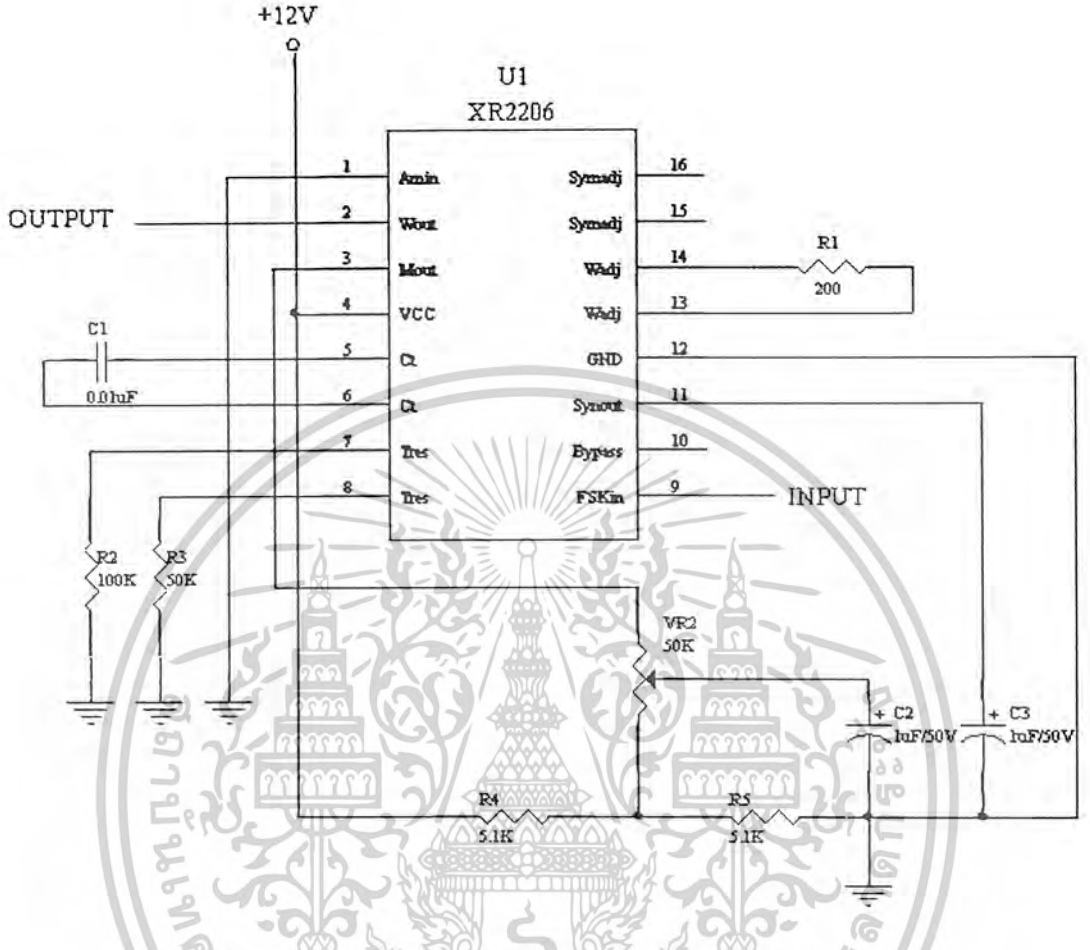
4.1.1 การทดลองวงจรเข้ารหัส



รูปที่ 4.1 แสดงวงจรเข้ารหัส

- ต่อวงจรตามรูปที่ 4.1 วัดสัญญาณที่เอาต์พุต
- ทำการเปลี่ยนค่าที่ดิปสวิตซ์โดยสามารถกำหนดรหัสต่างๆได้
- วัดสัญญาณที่ขา 5 ของไอซีเบอร์ 74HC151

4.1.2 การทดลองวงจรเอฟเอสเคมอดูเลเตอร์

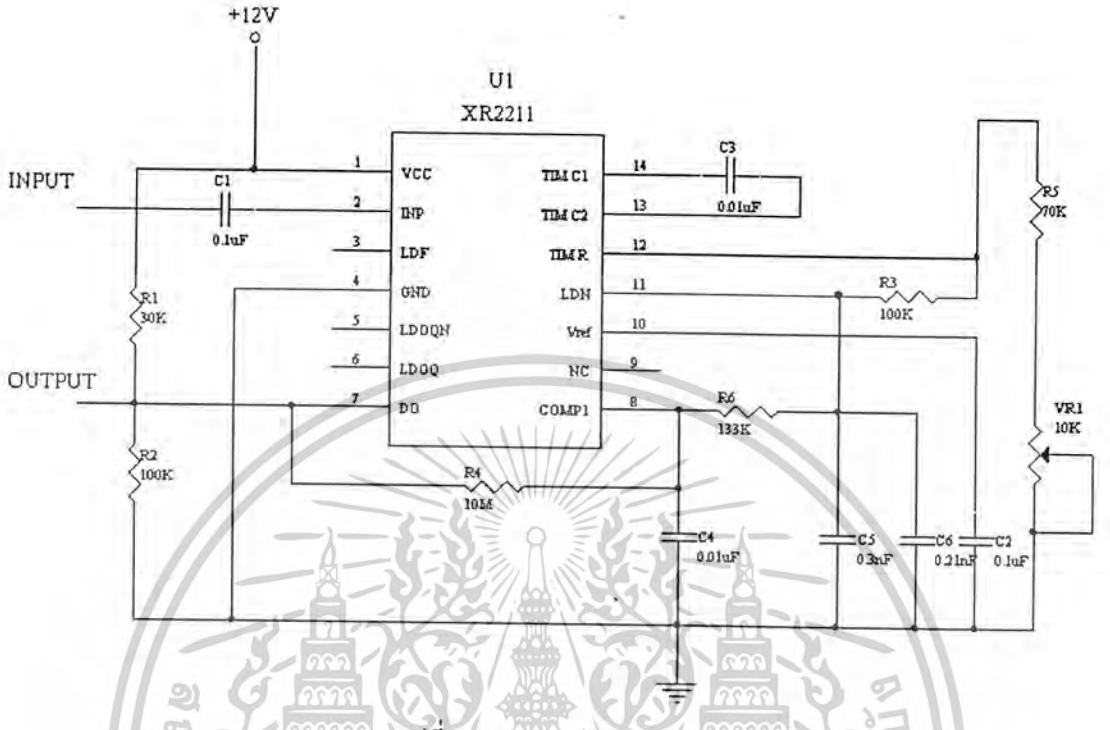


รูปที่ 4.2 วงจรเอฟเอสเคมอดูเลเตอร์

- ต่อวงจรตามรูปที่ 4.2 แล้วป้อนสัญญาณทางด้านอินพุตที่ขา9ของไอซี XR2206โดยอินพุตที่เข้ามาจากวงจรเข้ารหัส

- ทำการวัดสัญญาณเอาต์พุตที่ขา2ของไอซีสามารถปรับขนาดของสัญญาณได้ที่VR2

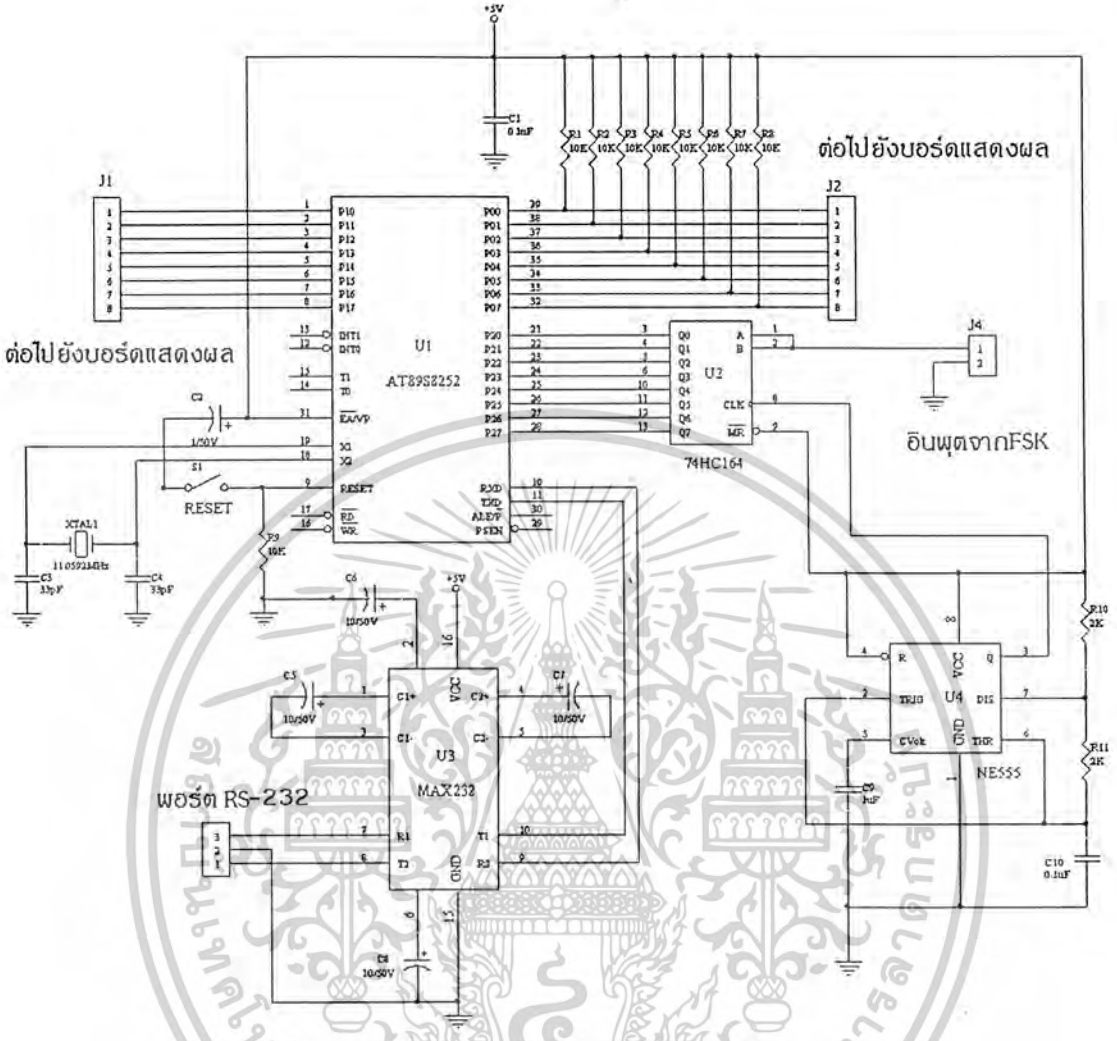
4.1.3 การทดลองวงจรเอฟเอสเคดีมอดูเลเตอร์



รูปที่ 4.3 วงจรเอฟเอสเคดีมอดูเลเตอร์

- ต่อวงจรตามรูปที่ 4.3 ป้อนสัญญาณอินพุตเข้าที่ขา2ของไอซี XR2211
- วัดสัญญาณเอาต์พุตที่ขา 7 ของไอซีโดยสามารถปรับเปลี่ยนค่าความถี่เบี่ยงเบนได้ที่VR1

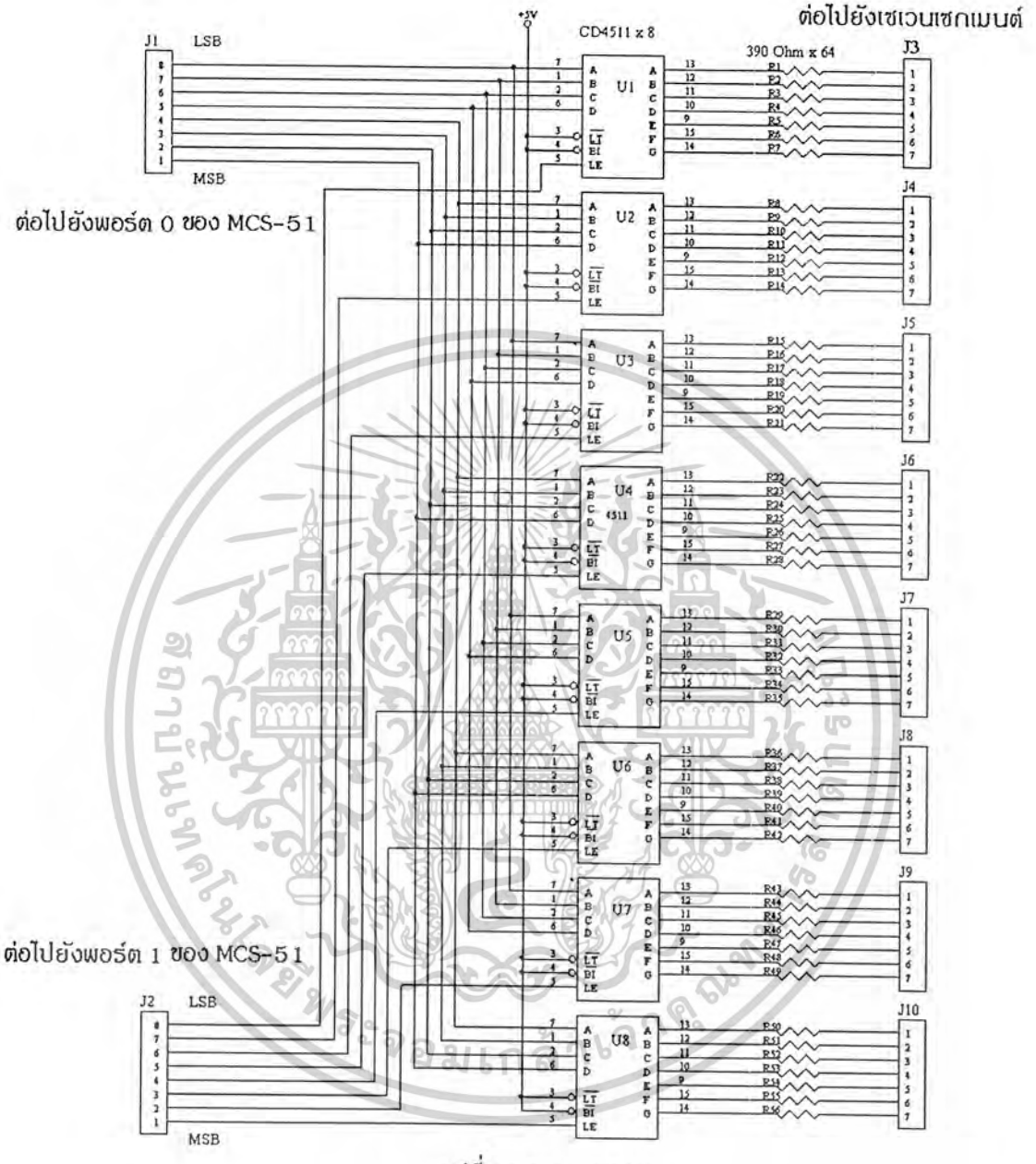
4.1.4 การทดลองวงจรไมโครคอนโทรลเลอร์ (ไคลเอนต์)



รูปที่ 4.4 วงจรไมโครคอนโทรลเลอร์

- ต่อวงจรตามรูปที่ 4.4 ป้อนสัญญาณอินพุตจากวงจรเอฟเอสเคดีมอดูเลเตอร์ที่จุด J4
- วัดสัญญาณที่ขา 1 และขา 8 ของไอซีเบอร์ 74HC164
- วัดสัญญาณเอาต์พุตของไอซีเบอร์ 74HC164 ทั้งหมดเพื่อเปรียบเทียบกับขาอินพุต
- วัดสัญญาณที่ขา 10 และขา 11 ของไอซีเบอร์ AT89S8252 เป็นสัญญาณข้อมูลรับและส่งสำหรับไมโครคอนโทรลเลอร์

4.1.5 การทดลองวงจรแสดงผล

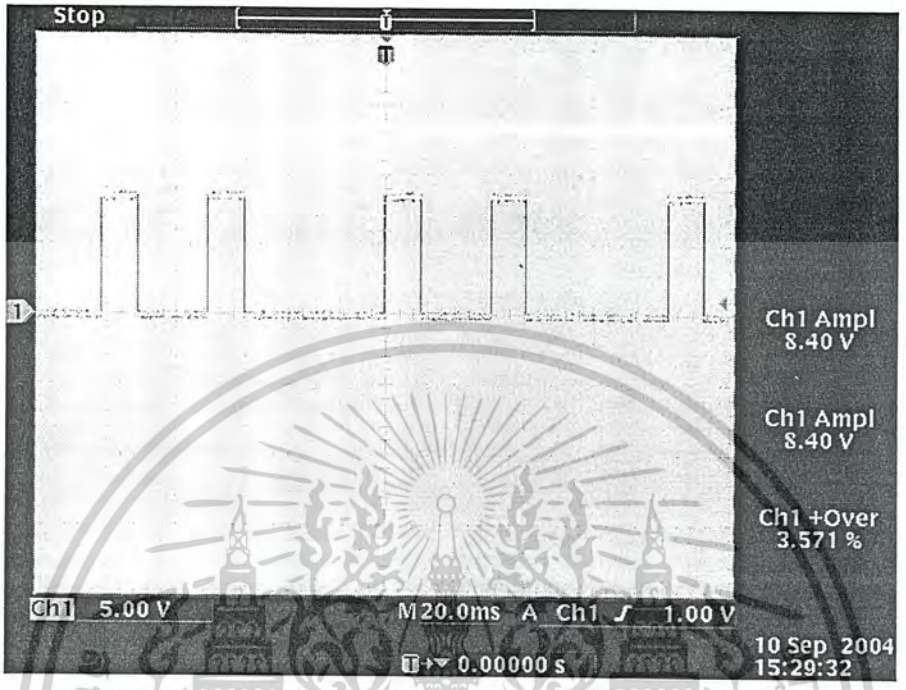


รูปที่ 4.5 วงจรแสดงผล

- ต่อวงจรทดลองตามรูปที่ 4.5ป้อนสัญญาณจากพอร์ต0และพอร์ต1จากไมโครคอนโทรลเลอร์ที่จุดต่อ J1 และ J2
- ดูผลของการแสดงผลจากบอร์ดแสดงผล
- เปรียบเทียบสัญญาณที่ขาอินพุตของไอซีเบอร์ CD4511 กับการแสดงผลจากบอร์ดแสดงผล

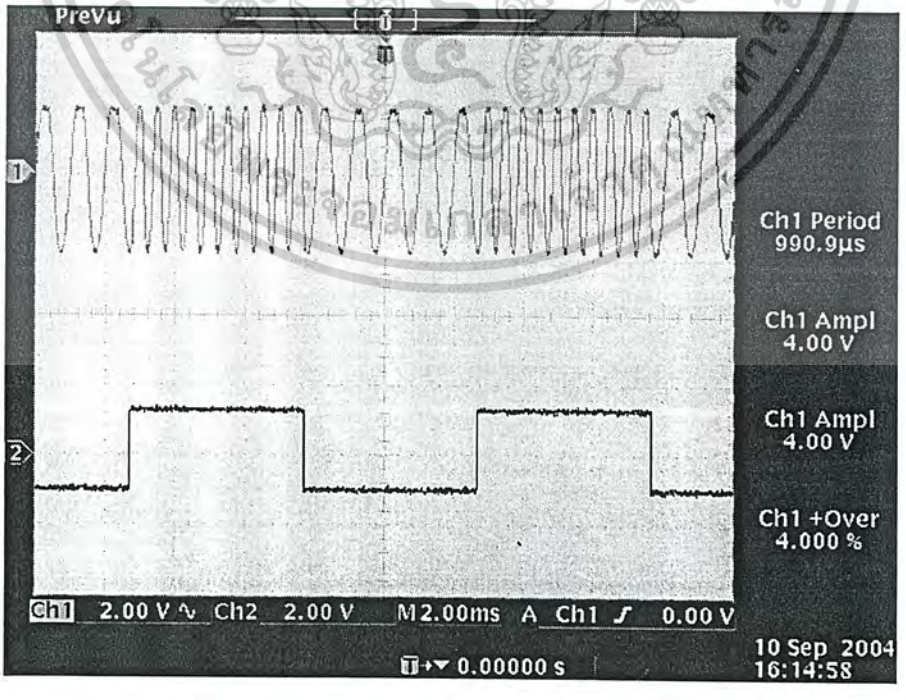
4.2 ผลการทดลอง

4.2.1 รูปสัญญาณวงจรเข้ารหัส



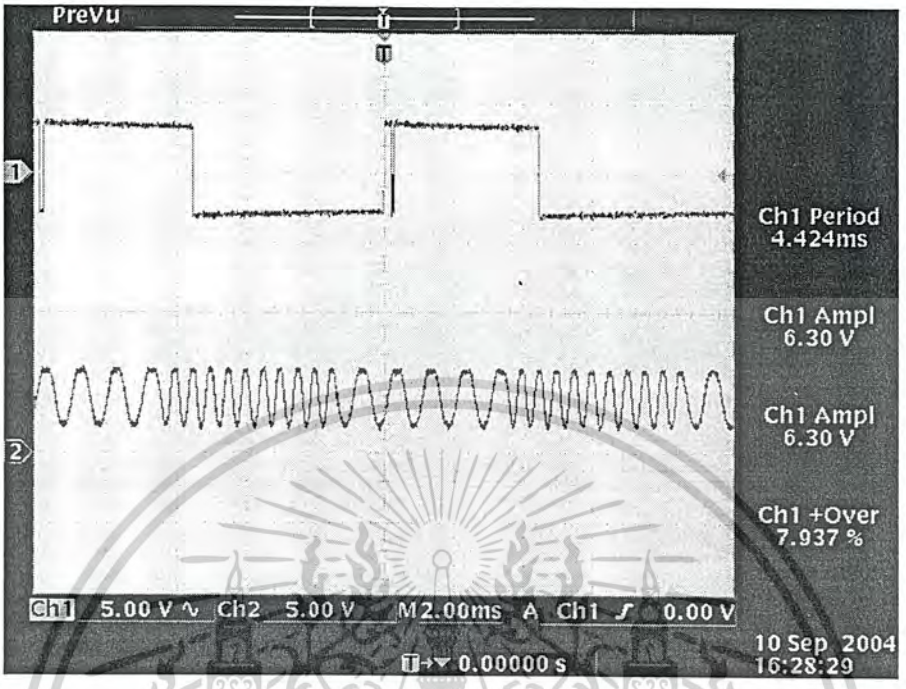
รูปที่ 4.6 แสดงสัญญาณวงจรเข้ารหัส

4.2.2 รูปสัญญาณวงจรเอฟเอสเคมอดูเลเตอร์



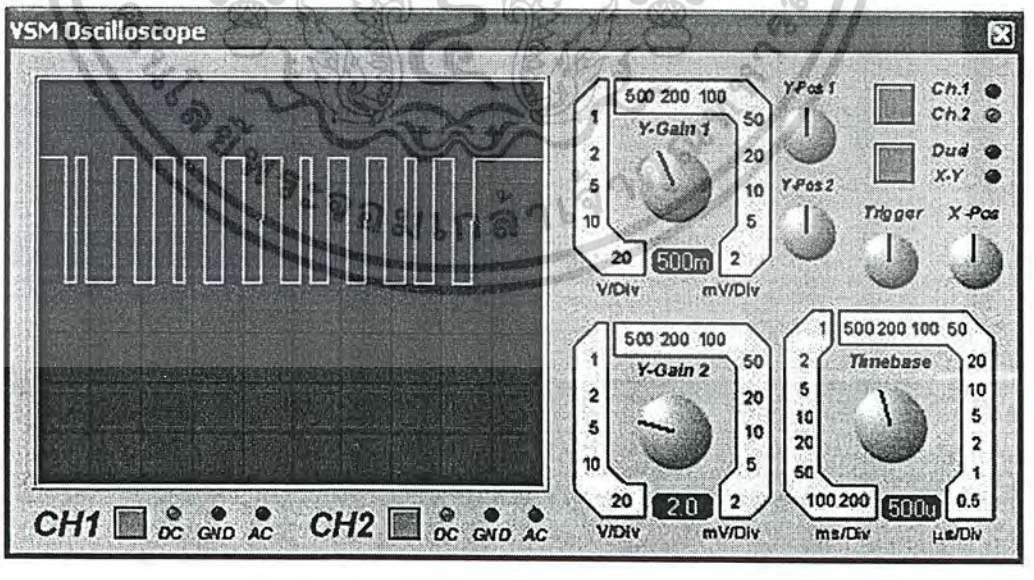
รูปที่ 4.7 แสดงสัญญาณเอฟเอสเคมอดูเลเตอร์

4.2.3 รูปสัญญาณวงจรเอฟเอสเคดีมอดูเลเตอร์

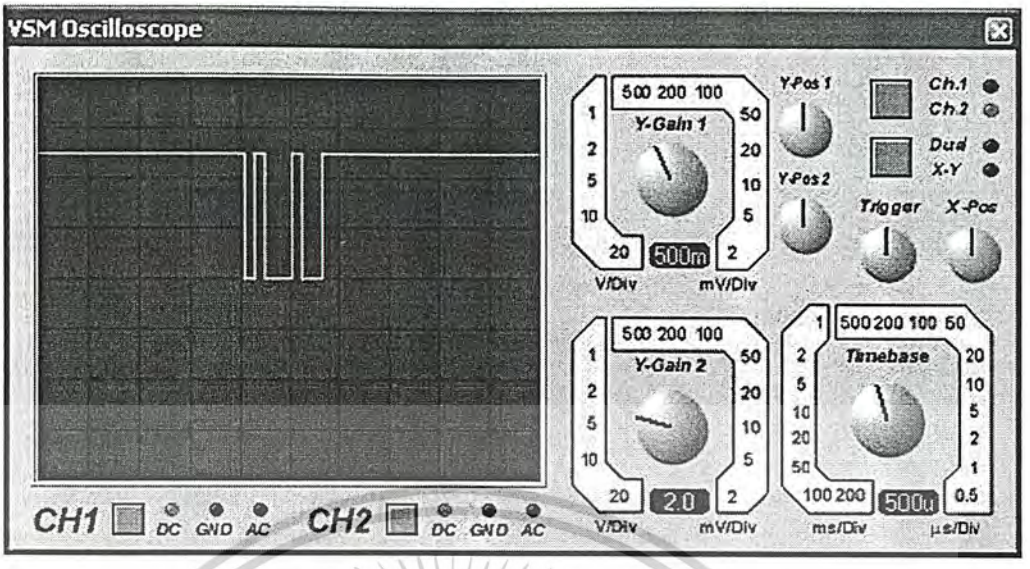


รูปที่ 4.8 แสดงสัญญาณเอฟเอสเคดีมอดูเลเตอร์

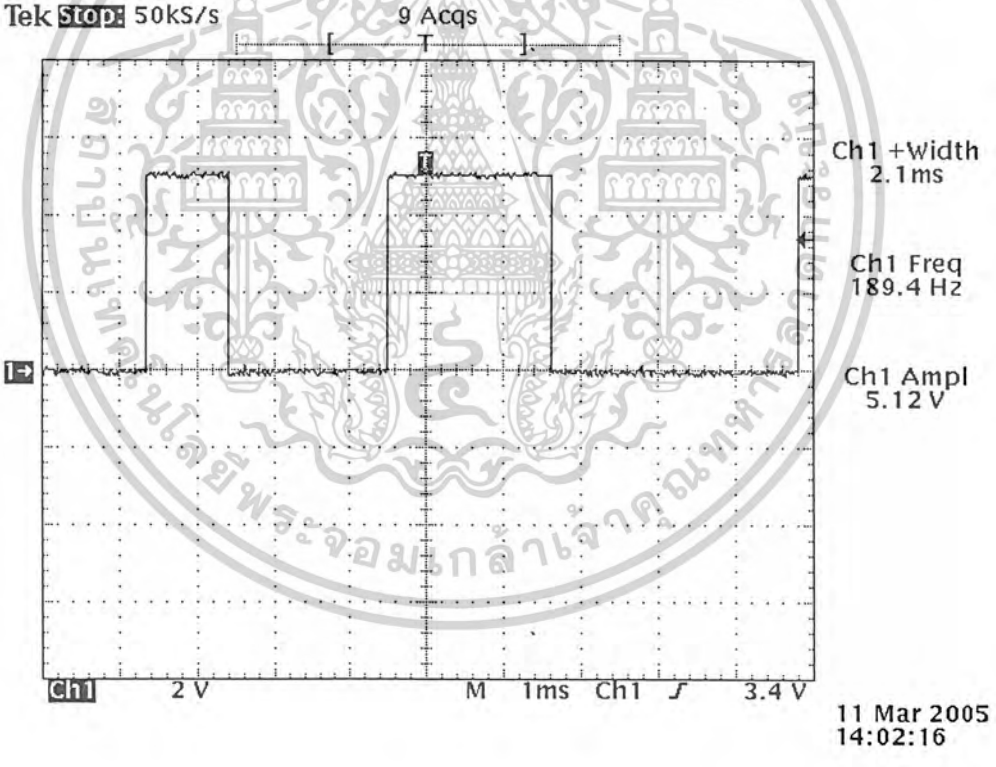
4.2.4 รูปสัญญาณวงจรไมโครคอนโทรลเลอร์ (ไคลเอนต์)



รูปที่ 4.9 แสดงสัญญาณวงจรไมโครคอนโทรลเลอร์ที่ขา 10 ของไอซี MCS-51ส่งค่ารหัสแอสกี 1322



รูปที่ 4.10 แสดงสัญญาณวงจรไมโครคอนโทรลเลอร์ที่ขา 11 ของไอซี MCS-51ส่งค่ารหัสแอสกี 91

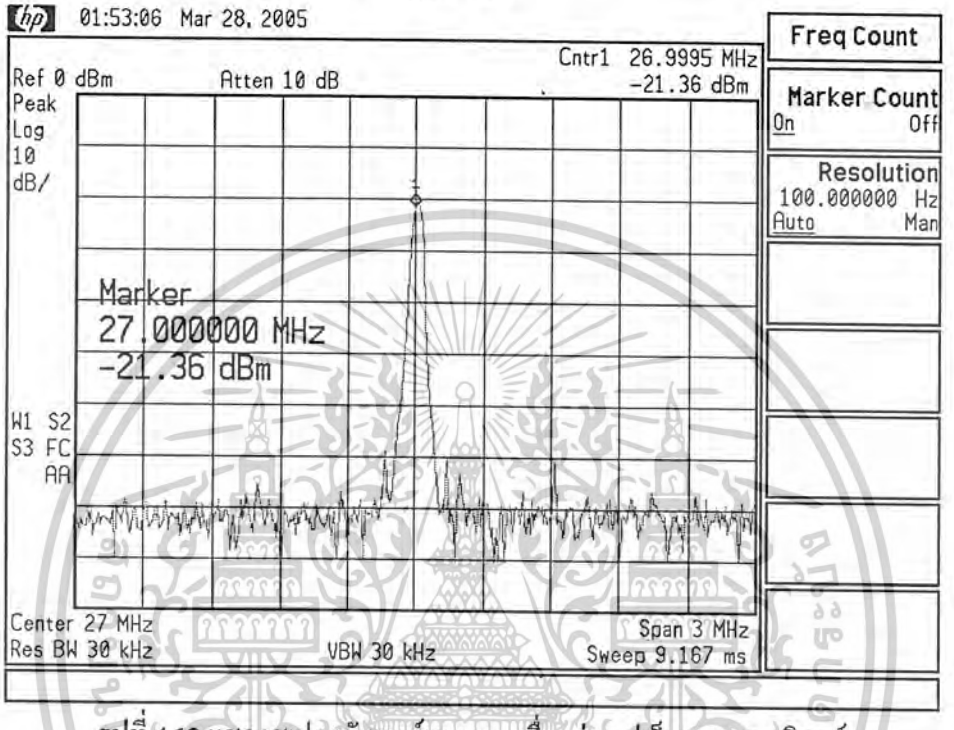


รูปที่ 4.11 แสดงสัญญาณวงจรไมโครคอนโทรลเลอร์ที่ขา 1 ของไอซี 74HC164

4.2.6 การทดลองหากำลังของเครื่องส่งเอฟเอ็ม

จุดประสงค์เพื่อทดลองหากำลังที่ส่งออกจากเครื่องส่ง

4.2.6.1 การทดลองหากำลังและความถี่ของเครื่องส่งเอฟเอ็ม 27 เมกะเฮิรตซ์โดยใช้สเปกตรัมอนาไรเซอร์วัดที่เอาท์พุตของเครื่องส่งโดยแสดงดังรูปที่ 4.12



รูปที่ 4.12 แสดงสเปกตรัมเอาต์พุทของเครื่องส่งเอฟเอ็ม 27 เมกะเฮิรตซ์

จากรูปที่ 4.12 เครื่องส่งเอฟเอ็ม27 เมกะเฮิรตซ์วัดความถี่ที่ใช้ในการส่งได้ 27.000 เมกะเฮิรตซ์

การหากำลังส่ง

จากรูปที่ 4.12 วัดขนาดสัญญาณได้ -21. 36dBm ดังนั้นหากำลังส่งได้

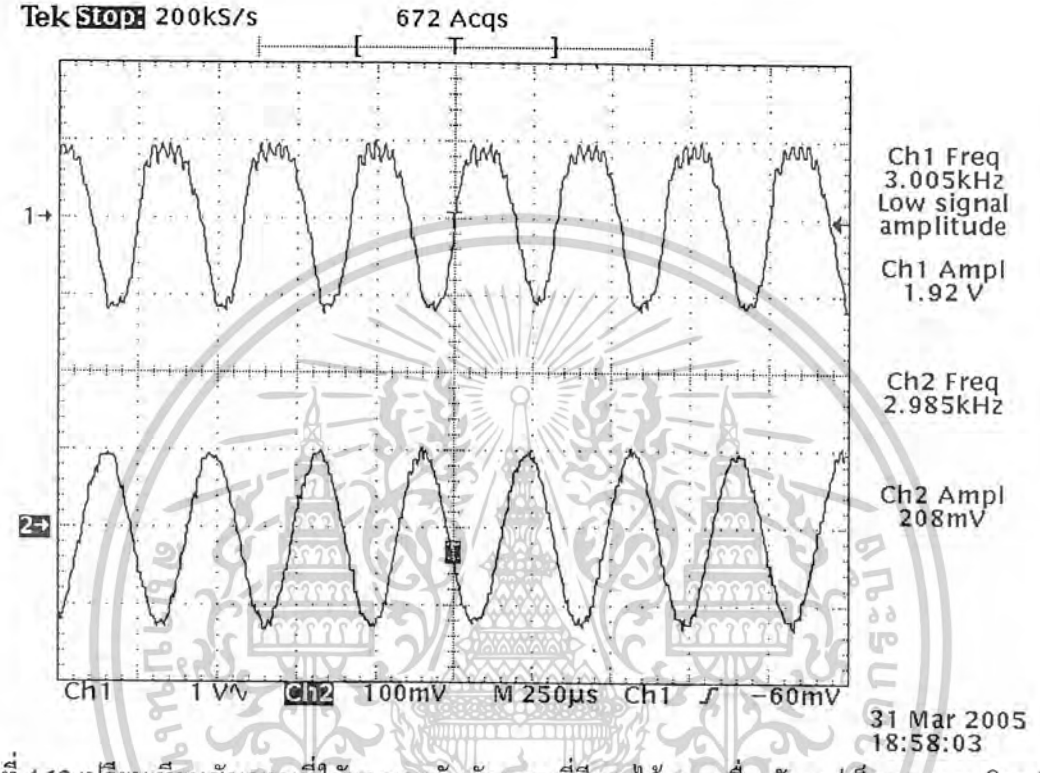
$$-21.36dBm = 10\log\frac{P}{1mW}$$

$$P = 7.31\times10^{-6}W$$

4.2.7 การทดลองหาเซนซิติวิตี้ (Sensitivity) ของเครื่องรับ

4.2.7.1 การหาเซนซิติวิตี้ของเครื่องรับเอฟเอ็ม 27 เมกะเฮิรตซ์

ใช้ซิกเนลเจนเนเรเตอร์ป้อนความถี่ 27 เมกะเฮิรตซ์ มอดูเลตกับสัญญาณไซน์ 3 กิโลเฮิรตซ์โดยเริ่มปรับแอมปลิจูดตั้งแต่ -100 dBm จนกว่าเครื่องรับจะดีเทคสัญญาณไซน์ 3 กิโลเฮิรตซ์ได้



รูปที่ 4.13 เปรียบเทียบสัญญาณที่ใช้มอดูเลตกับสัญญาณที่ดีเทคได้ของเครื่องรับเอฟเอ็ม 27 เมกะเฮิรตซ์

Ch 1 สัญญาณที่ใช้มอดูเลตด้านซิกแนลเจนเนเรเตอร์

Ch 2 สัญญาณที่ดีเทคได้ด้านเครื่องรับ

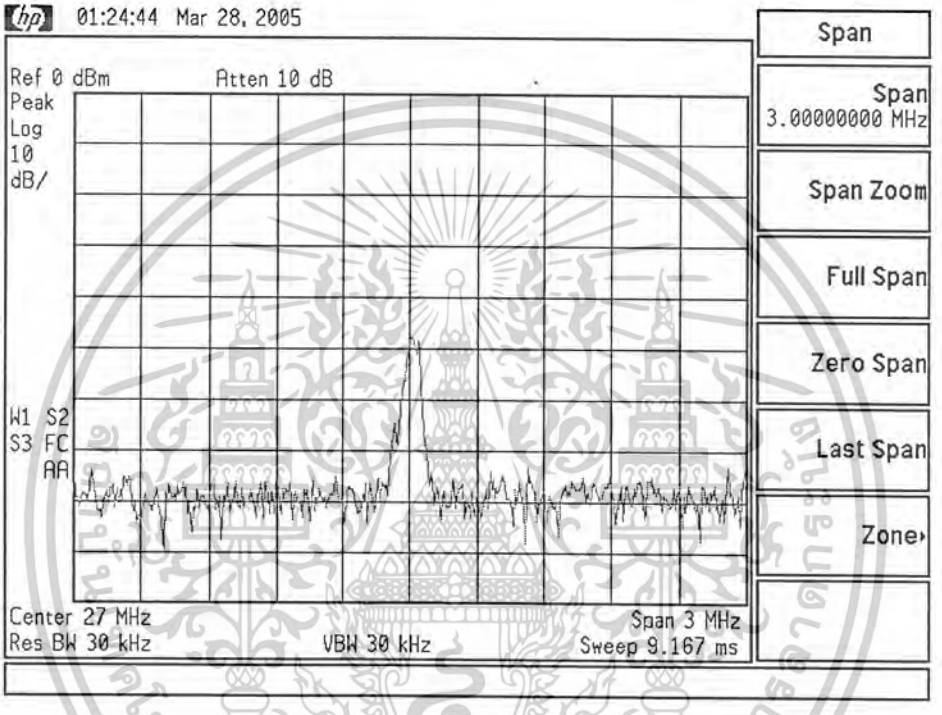
จากการทดลองโดยเริ่มปรับแอมพลิจูดตั้งแต่ -100 dBm ที่ซิกแนลเจนเนเรเตอร์ได้ผลคือ ที่ -17.05 dBm เครื่องรับจะเริ่มดีเทคสัญญาณได้โดยไม่มีความผิดเพี้ยนดังรูปที่ 4.13 ดังนั้นเซนซิติวิตี้ของเครื่องรับเอฟเอ็ม 27 เมกะเฮิรตซ์คือ -17.05 dBm

4.2.8 การทดลองหาค่าความเข้มของสัญญาณวิทยุ (RF Field Strength)

จุดประสงค์เพื่อทดลองหาความเข้มสัญญาณวิทยุของเครื่องส่งเอฟเอ็ม 27เมกะเฮิรตซ์ ที่ระยะทาง 5 เมตร

เนื่องจากการทดลองไม่มีเครื่องวัดความเข้มสัญญาณความเข้มวิทยุ 27 เมกะเฮิรตซ์ จึงทำการทดลองโดยใช้สเปกตรัมวัดความถี่ที่อินพุตของภาครับของเครื่องรับที่ระยะทาง 5 เมตร

หมายเหตุ ในการทดลองใช้สายอากาศแบบเส้นลวดทั้งเครื่องรับและเครื่องส่งโดยทำการทดลองภายในห้องทดลอง



รูปที่ 4.14 แสดงสเปกตรัมที่วัดได้ที่อินพุตของเครื่องรับเอฟเอ็ม 27 เมกะเฮิรตซ์

จากรูปที่ 4.14 เป็นการวัดความเข้มของสัญญาณวิทยุที่ภาครับเอฟเอ็ม 27เมกะเฮิรตซ์ได้ความแรงของสัญญาณ -47 dBm

และจากการทดสอบหาว่าเครื่องรับและเครื่องส่งสามารถรับส่งกันได้ไกลเท่าไหร่โดยการป้อนค่าสัญญาณไซน์ 1 กิโลเฮิรตซ์อินพุทที่เครื่องส่งและทดสอบโดยการใช้หูฟังติดตั้งไว้ที่เอาท์พุทของเครื่องรับ โดยทำการทดสอบที่ระยะทางต่างคือ

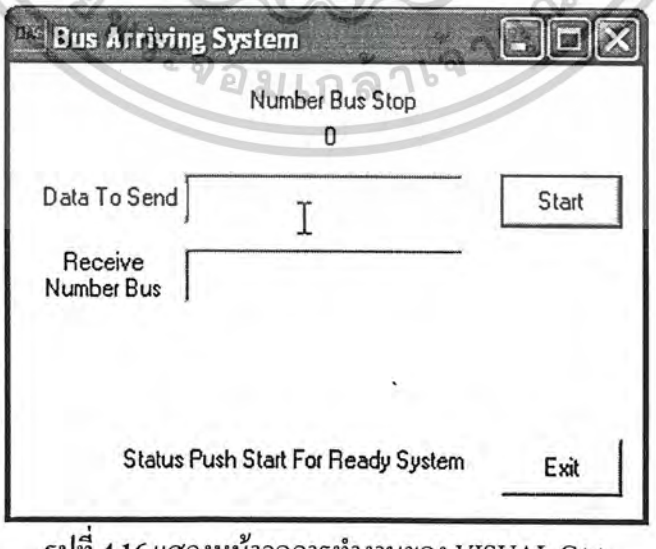
ระยะทาง(เมตร)	ความสามารถในการรับฟัง
10	ชัดเจนดี
15	ชัดเจนดี
20	ชัดเจนดี
22	ชัดเจนดี
23	ไม่ชัดเจน

4.2.9 รูปแสดงการเก็บข้อมูลเข้าไปยังฐานข้อมูลของMySQL

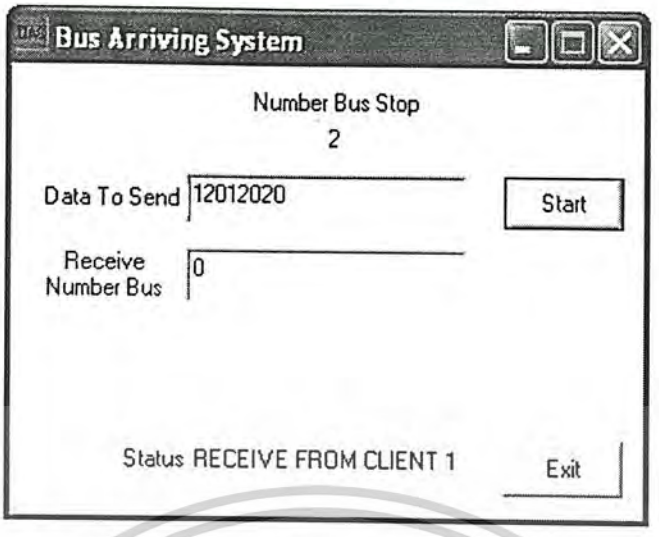
←T→	id_bus	id_bus_stop	time1	os_date	os_time
?	101	2	1108578163	02/17/05	01:22:43
?	201	2	1108578223	02/17/05	01:23:43
?	102	1	1108578345	02/17/05	01:25:45
?	101	3	1108578348	02/17/05	01:26:17
?	201	1	1108578441	02/17/05	01:27:21
?	101	1	1108578485	02/17/05	01:28:05
?	201	3	1108578487	02/17/05	01:28:07
?	101	1	1108578488	02/17/05	01:28:08
?	201	2	1108578490	02/17/05	01:28:10
?	101	3	1108578491	02/17/05	01:28:11
?	201	1	1108578492	02/17/05	01:28:12
?	201	3	1108578493	02/17/05	01:28:13
?	101	1	1108578494	02/17/05	01:28:14
?	201	2	1108578495	02/17/05	01:28:15
?	101	1	1108578498	02/17/05	01:28:18
?	101	2	1108578704	02/17/05	01:31:44

รูปที่ 4.15 แสดงการเก็บข้อมูลเข้าไปยังฐานข้อมูลของMySQL

4.2.10 รูปแสดงหน้าจอการทำงานของ VISUAL C++

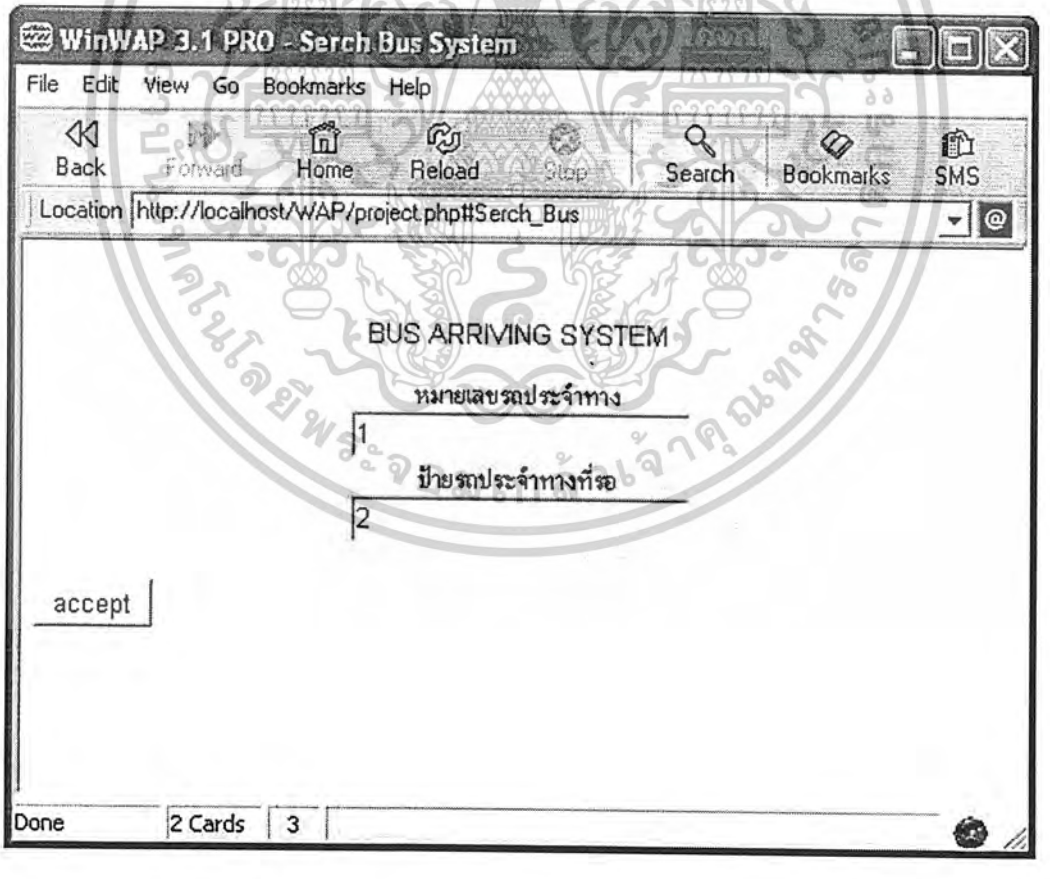


รูปที่ 4.16 แสดงหน้าจอการทำงานของ VISUAL C++

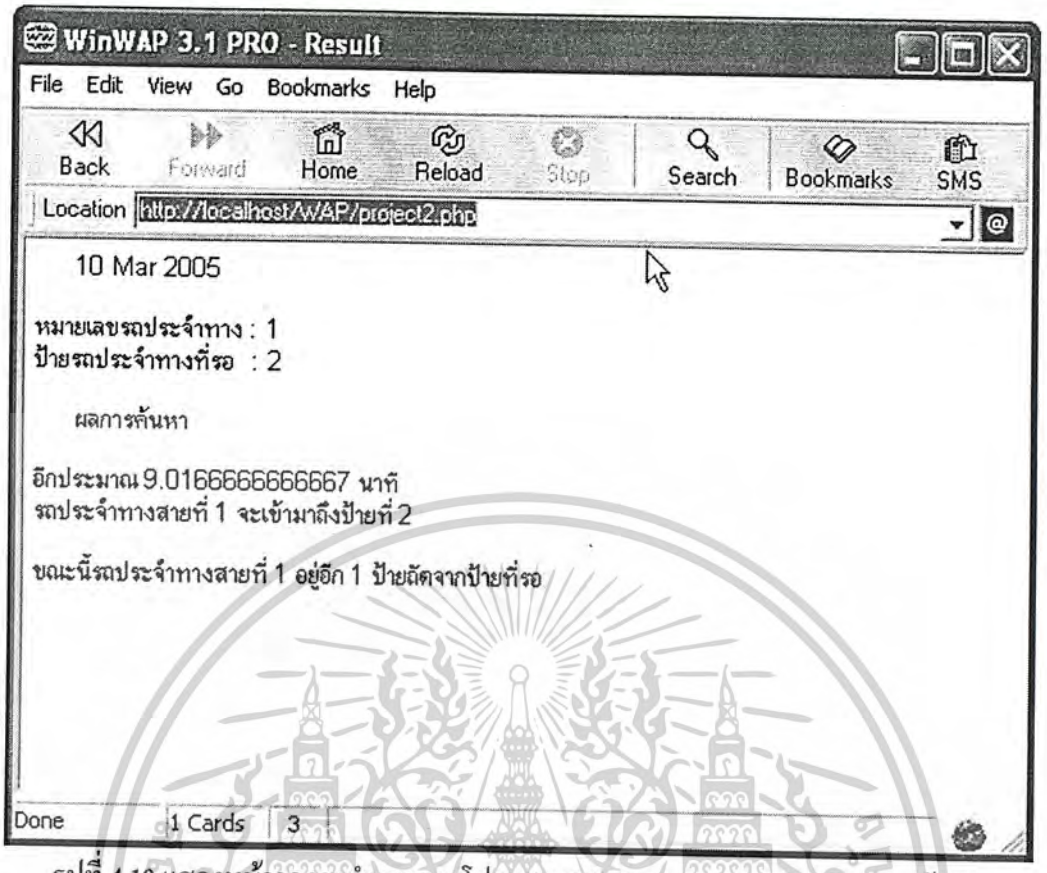


รูปที่ 4.17 แสดงหน้าจอการทำงานของ VISUAL C++ กรณีเริ่มต้นการทำงาน

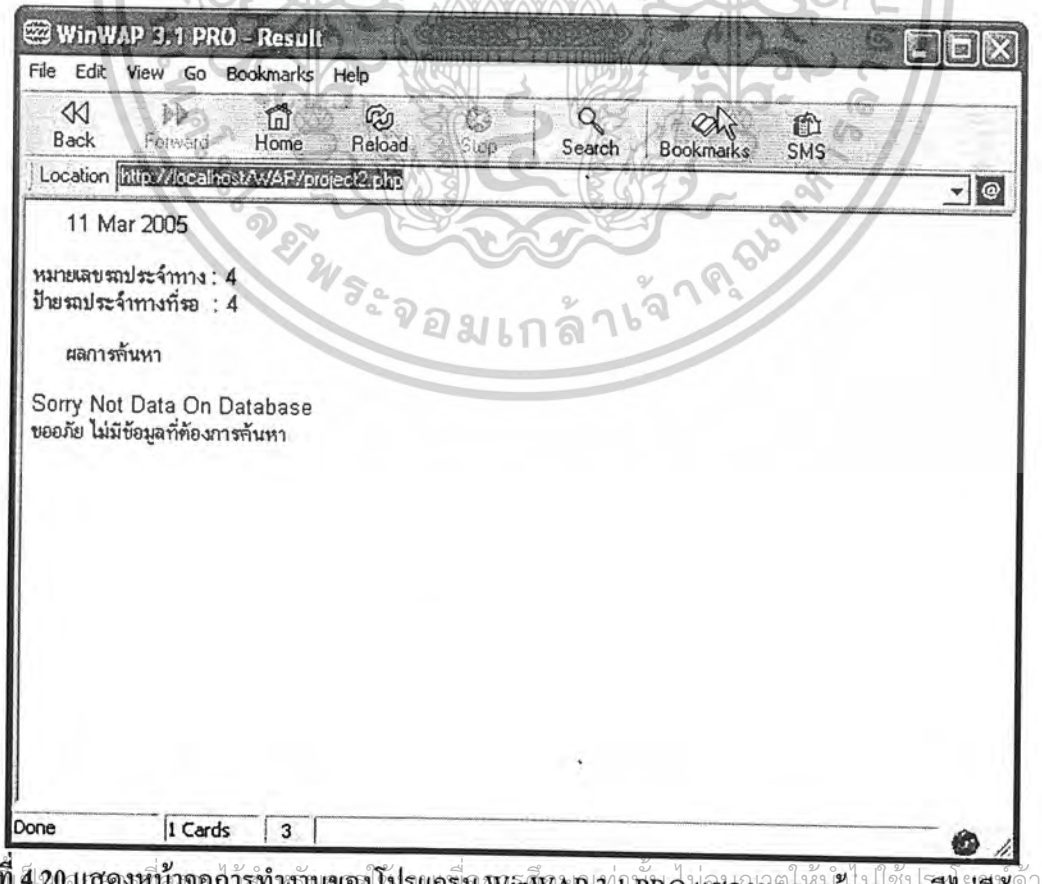
4.2.11 รูปแสดงหน้าจอการทำงานของโปรแกรม WinWAP 3.1 PRO



รูปที่ 4.18 แสดงหน้าจอการทำงานของโปรแกรม WinWAP 3.1 PRO ในการรับค่า



รูปที่ 4.19 แสดงหน้าจอการทำงานของโปรแกรม WinWAP 3.1 PRO แสดงผลการค้นหา



รูปที่ 4.20 แสดงหน้าจอการทำงานของโปรแกรม WinWAP 3.1 PRO แสดงผลการค้นหากรณีไม่มีข้อมูล

บทที่ 5
บทวิจารณ์และบทสรุป

5.1 ปัญหาที่เกิดขึ้นจากการทดลอง

จากการทดลองจะมีสัญญาณรบกวนเข้ามายังระบบแต่ยังไม่มีผลมากนัก เพราะสามารถตรวจจับสัญญาณที่เข้ามาได้ และค่าความถี่ที่ได้จะไม่ตรงกับความถี่ที่ได้คำนวณไว้เนื่องมาจากความผิดพลาดของค่าความต้านทานและตัวเก็บประจุเพราะจากที่คำนวณไว้ไม่สามารถหาได้

การปรับค่าต่างๆทำได้ลำบากเพราะเมื่อปรับไปเล็กน้อยก็ทำให้ความถี่เปลี่ยนไปอย่างมากเช่น การปรับค่าความถี่มาร์คและสเปซของวงจรเอฟเอสเคดีมอดูเลเตอร์ และความถี่ที่ใช้ในการถอดรหัสสัญญาณที่เข้ามายังพอร์ตของไมโครคอนโทรลเลอร์ จะต้องปรับให้เท่ากับค่าความถี่ที่เข้ารหัส ถ้าเกิดความถี่ไม่ตรงกันสัญญาณที่รับก็จะผิดพลาด

ไอซีชนิดซีมอสนั้นจะมีการเปลี่ยนแปลงไปถ้าหากเอามือไปถูกขาต่างๆหรือเส้นเชื่อมต่อระหว่างขาทำให้มีความผิดพลาดเกิดขึ้นในการเก็บข้อมูลไปยังฐานข้อมูล

การต่อสัญญาณไปยังพอร์ตอนุกรม RS232 ทำได้ที่ระยะทางที่จำกัดเพียง50ฟุตถ้าเกินกว่านี้สัญญาณจะเกิดการเปลี่ยนแปลงมีระดับแรงดันลดลงอย่างมากและไม่สามารถตรวจจับสัญญาณได้

5.2 แนวทางในการแก้ปัญหา

ปัญหาที่เกิดจากอุปกรณ์ สามารถแก้ไขได้โดยใช้ค่าความต้านทานและตัวเก็บประจุที่ใกล้เคียงที่สุดแต่ข้อมูลที่ได้ก็ยังคงเป็นค่าที่ถูกต้อง การปรับค่าเพื่อให้มีความแม่นยำจะใช้ตัวต้านทานปรับค่าที่มีความละเอียดในการปรับเพื่อให้ได้ค่าสัญญาณหรือความถี่ที่ต้องการ สายสัญญาณของพอร์ตคอมพิวเตอร์ซึ่งใช้สายได้ที่ระยะทางไม่ไกลนักและเกิดการรบกวนของสัญญาณ ซึ่งสามารถพัฒนาให้สายสัญญาณมีความยาวมากๆโดยอาจใช้การเชื่อมต่อทางแสงเข้ามาช่วยแล้วมีตัวรีพีทเตอร์ในการทวนสัญญาณถ้ากรณีป้ายต่างๆอยู่ห่างกันมากๆได้

ภาคผนวก

โปรแกรม

```
// serial_projectDlg.cpp : implementation file

#include "stdafx.h"
#include "serial_project.h"
#include "serial_projectDlg.h"
#include <stdio.h>
#include <mysql.h>
#include <time.h>
#include <sys/types.h>
#include <sys/timeb.h>
#include <string.h>
#ifdef _DEBUG
#define new DEBUG_NEW
#undef THIS_FILE
static char THIS_FILE[] = __FILE__;
#endif

/////////////////////////////////////////////////////////////////////////////
//////
// CSerial_projectDlg dialog

CSerial_projectDlg::CSerial_projectDlg(CWnd* pParent /*=NULL*/)
	: CDialog(CSerial_projectDlg::IDD, pParent)
{
	//{{AFX_DATA_INIT(CSerial_projectDlg)
	m_output = _T("");
	m_input = _T("");
	m_number = 0;
	m_Show = " Push Start For Ready System ";
	//}}AFX_DATA_INIT
	//DEFINE FIXED BUS STOP
	id_bus_spc=0;
    FIXED_id_bus_stop0_1=30;

    FIXED_id_bus_stop1_2=20;
    FIXED_id_bus_stop2_3=15;
	m_hIcon = AfxGetApp()->LoadIcon(IDR_MAINFRAME);
}

void CSerial_projectDlg::DoDataExchange(CDataExchange* pDX)
{
	CDialog::DoDataExchange(pDX);
	//{{AFX_DATA_MAP(CSerial_projectDlg)
	DDX_Control(pDX, IDC_MSCOMM1, m_comp);
	DDX_Text(pDX, IDC_EDIT_output, m_output);
	DDX_Text(pDX, IDC_EDIT_input, m_input);
	DDX_Text(pDX, IDC_EDIT_number, m_number);
	DDX_Text(pDX, IDC_EDIT, m_Show);
	//}}AFX_DATA_MAP
}

BEGIN_MESSAGE_MAP(CSerial_projectDlg, CDialog)
	//{{AFX_MSG_MAP(CSerial_projectDlg)
	ON_WM_PAINT()
	ON_WM_QUERYDRAGICON()
	ON_BN_CLICKED(IDOK, OnSend)
	ON_BN_CLICKED(IDC_BUTTON1, OnExit)
```

```
	//}}AFX_MSG_MAP
END_MESSAGE_MAP()

////////////////////////////////////////////////////////////////////////////
//////
// CSerial_projectDlg message handlers

BOOL CSerial_projectDlg::OnInitDialog()
{
	CDialog::OnInitDialog();
	SetIcon(m_hIcon, TRUE);			// Set big icon
	SetIcon(m_hIcon, FALSE);		// Set small icon
	// TODO: Add extra initialization here
	// COM Port Initialisation
	m_comp.SetCommPort(1);
	m_comp.SetSettings("9600,N,8,1");
	m_comp.SetInputLen(1);
	m_comp.SetInputMode(0);
	m_comp.SetRTSEnable(TRUE);
	m_comp.SetRThreshold(1);
	//m_comp.SetSThreshold (1);
	m_comp.SetPortOpen(true);
	return TRUE;  // return TRUE  unless you set the focus to a
control
}
// If you add a minimize button to your dialog, you will need the code
below
//  to draw the icon.  For MFC applications using the document/view
model,
//  this is automatically done for you by the framework.

void CSerial_projectDlg::OnPaint()
{
	if (IsIconic())
	{
		CPaintDC dc(this); // device context for painting

		SendMessage(WM_ICONERASEBKGND, (WPARAM) dc.GetSafeHdc(), 0);

		// Center icon in client rectangle
		int cxIcon = GetSystemMetrics(SM_CXICON);
		int cyIcon = GetSystemMetrics(SM_CYICON);
		CRect rect;
		GetClientRect(&rect);
		int x = (rect.Width() - cxIcon + 1) / 2;
		int y = (rect.Height() - cyIcon + 1) / 2;

		// Draw the icon
		dc.DrawIcon(x, y, m_hIcon);
	}
	else
	{
		CDialog::OnPaint();
	}
}

HCURSOR CSerial_projectDlg::OnQueryDragIcon()
```

```
{
      return (HCURSOR) m_hIcon;
}

BEGIN_EVENTSINK_MAP(CSerial_projectDlg, CDialog)
    //{{AFX_EVENTSINK_MAP(CSerial_projectDlg)
      ON_EVENT(CSerial_projectDlg, IDC_MSCOMM1, 1 /* OnComm */,
OnOnCommMscomm1, VTS_NONE)
      //}}AFX_EVENTSINK_MAP
END_EVENTSINK_MAP()
void CSerial_projectDlg::OnOnCommMscomm1()
{
      // TODO: Add your control notification handler code here
      if (m_comp.GetCommEvent()==2 )
            {
                  VARIANT in_dat;
                  char buf_m_show[1];
                  in_dat = m_comp.GetInput();
                  CString strInput(in_dat.bstrVal);
                  m_input =strInput;
//receive id_bus 1_1
                  if (m_input==0x91)

                  {
                        id_bus=1;
                        m_input="1";
                        MySql_Add();
                        m_Show="RECEIVE FROM CLIENT ";
                        swap=id_bus_stop;
                        _itoa(id_bus_stop,buf_m_show,10);
                        m_Show=m_Show+buf_m_show[0];
                        OnSend();
                  }
//receive id_bus 2
                  else if (m_input==0x92)
                  {
                        id_bus=2;
                        m_input="2";
                        MySql_Add();
                        m_Show="RECEIVE FROM CLIENT ";
                        _itoa(id_bus_stop,buf_m_show,10);
                        m_Show=m_Show+buf_m_show[0];
                        OnSend();
                  }
//receive id_bus 1_2
                  else if (m_input==0x93)
                  {
                        id_bus=3;
                        m_input="3";
                        //MySql_Add();
                        m_Show="RECEIVE FROM CLIENT ";
                        _itoa(id_bus_stop,buf_m_show,10);
                        m_Show=m_Show+buf_m_show[0];
                        OnSend();
                  }
                  else if (m_input==0x94)
                  {
```

```
                        id_bus=0;
                        m_input="0";
                        m_Show="RECEIVE FROM CLIENT ";
                        _itoa(id_bus_stop,buf_m_show,10);
                        m_Show=m_Show+buf_m_show[0];
                        OnSend();
                  }
                  else
                        Sleep(1);
                        UpdateData(FALSE);
            }
}

void CSerial_projectDlg::OnSend()
{
            // send data
            id_bus_spc=0;
            m_number=m_number+1;
            char buf[3];
//send data bus stop number
            if (m_number==1)
            {
            int i_test,i_test3[1000],i_test4,i_result_min,i_result_sec;
            int i_test2;
            float i_result_percent,f_result;
            char test[20],buf_send[5];
            id_bus_stop=1;
            id_bus=101;
            MySql_Calc();
            i_test4=0;
            i_test2=numofrows;
//fine difference among times in bus stop
            for (i_test=0;i_test < (i_test2-1);i_test++)
            {
            i_test3[i_test] = mysql_time_row[i_test+1] -
mysql_time_row[i_test];
            i_test4 = i_test4 + i_test3[i_test];
            }
            if (i_test2==0)

// in case no data or initial data
            {
                  buf[0]='1';
                  buf[1]='0';
                  buf[2]='0';
                  buf[3]='0';
            }
            else
            {
                  i_result_sec = i_test4/(i_test2-1);
// average result in second
                  i_result_min = i_result_sec/60;
// average result in minute
      i_result_percent=abs(i_result_min - FIXED_id_bus_stop0_1) ;
      f_result=100 - ((i_result_percent / FIXED_id_bus_stop0_1) * 100);
// calculate percentage
                              if (f_result > 10)
```

```
// check fault of time not over than 10 percent
                                    i_result_min=FIXED_id_bus_stop0_1;
                                else
                                    Sleep(1);
                                if (i_result_min < 10)
// if result more than 10 minute
                                {
                                    buf[0]='1';
//send data 1
                                memset(&buf_send, 0, sizeof(buf_send));
                                    _itoa(i_result_min,buf_send,10);
                                    buf[1]='0';
//send data 2
                                    buf[2]=buf_send[0];
//send data 3
                                }
                                else
                                {
                                    if (i_result_min<99)
// if result over 99 minute
                                    {
                                        buf[0]='1';
//send data 1
                                memset(&buf_send, 0, sizeof(buf_send));
                                _itoa(i_result_min,buf_send,10);
                                        buf[1]=buf_send[0];
                                            //send data 2
                                        buf[2]=buf_send[1];
                                        //send data 3
                                    }
                                    else
                                    {
                                        buf[0]='1';
                                        buf[1]='9';
                                        buf[2]='9';
                                    }
                                }
                                    buf[3]='0';
//send data 4
            }
            m_comp.SetOutput(COleVariant("A"));
//SEND TO MCS-51 BUS STOP NUMBER 1 ( id bus 1)
            Sleep(1);
            m_comp.SetOutput(COleVariant(buf));
            m_output="";
            for (int i=0;i < 4;i++)
            m_output=m_output+buf[i];
            memset(&buf, 0, sizeof(buf));
            id_bus_stop=1;
            id_bus=201;
            MySql_Calc();
            i_test4=0;
            i_test2=numofrows;
            for (i_test=0;i_test < (i_test2-1);i_test++)
//fine difference among times in bus stop
```

```
            {
            i_test3[i_test] = mysql_time_row[i_test+1] -
mysql_time_row[i_test];
            i_test4 = i_test4 + i_test3[i_test];
            }
            if (i_test2==0)
// in case no data or initial data
            {
                  buf[0]='1';
                  buf[1]='0';
                  buf[2]='0';
                  buf[3]='0';
            }
            else
            {
                  i_result_sec = i_test4/(i_test2 - 1);
// average result in second
                i_result_min = i_result_sec/60;
// average result in minute
      i_result_percent=abs(i_result_min - FIXED_id_bus_stop0_1) ;
      f_result=100 - ((i_result_percent / FIXED_id_bus_stop0_1) * 100);
// calculate percentage
                        if (f_result > 10)

// check fault of time not over than 10 percent
                              i_result_min=FIXED_id_bus_stop0_1;
                        else
                              Sleep(1);
                        if (i_result_min < 10)
// if result more than 10 minute
                        {
                              buf[0]='2';
//send data 1
                        memset(&buf_send, 0, sizeof(buf_send));
                              _itoa(i_result_min,buf_send,10);
                              buf[1]='0';
//send data 2
                              buf[2]=buf_send[0];
//send data 3
                        }
                        else
                        {
                              if (i_result_min<99)
// if result over 99 minute
                              {
                                    buf[0]='2';
//send data 1
                        memset(&buf_send, 0, sizeof(buf_send));

                        _itoa(i_result_min,buf_send,10);
                                    buf[1]=buf_send[0];
                                          //send data 2
                                    buf[2]=buf_send[1];
//send data 3
                              }
                              else
                              {
```

```
                                        buf[0]='2';
                                        buf[1]='9';
                                        buf[2]='9';
                                    }
                                }
                                        buf[3]='0';
//send data 4
            }
            m_comp.SetOutput(COleVariant("a"));
//SEND TO MCS-51 BUS STOP NUMBER 1 ( id bus 2)
            Sleep(1);
            m_comp.SetOutput(COleVariant(buf));
            for ( i=0;i < 4;i++)
            m_output=m_output+buf[i];
            }
//send data  bus stop number 2
            else if (m_number==2)
            {
            int i_test,i_test3[1000],i_test4,i_result_min,i_result_sec;
            int i_test2;
            float i_result_percent,f_result;
            char test[20],buf_send[5];
            id_bus_stop=2;
            id_bus=101;
            MySql_Calc();
            i_test4=0;
            i_test2=numofrows;
            for (i_test=0;i_test < (i_test2-1);i_test++)
//fine difference among times in bus stop
            {
i_test3[i_test] = mysql_time_row[i_test+1] - mysql_time_row[i_test];
            i_test4 = i_test4 + i_test3[i_test];
            }
            if (i_test2==0)
// in case no data or initial data
            {
                  buf[0]='1';
                  buf[1]='0';
                  buf[2]='0';
                  buf[3]='0';
            }
            else
            {
                  i_result_sec = i_test4/(i_test2-1);

// average result in second
                i_result_min = i_result_sec/60;
// average result in minute
            i_result_percent=abs(i_result_min - FIXED_id_bus_stop1_2) ;
      f_result=100 - ((i_result_percent / FIXED_id_bus_stop1_2) * 100);
// calculate percentage
                              if (f_result > 10)

// check fault of time not over than 10 percent
                                    i_result_min=FIXED_id_bus_stop1_2;
                              else
                                    Sleep(1);
```

```
                                if (i_result_min < 10)
// if result more than 10 minute
                                {
                                      buf[0]='1';
//send data 1
                                memset(&buf_send, 0, sizeof(buf_send));
                                      _itoa(i_result_min,buf_send,10);
                                      buf[1]='0';
//send data 2
                                      buf[2]=buf_send[0];
//send data 3
                                }
                                else
                                {
                                      if (i_result_min<99)
// if result over 99 minute
                                      {
                                            buf[0]='1';
//send data 1
                                memset(&buf_send, 0, sizeof(buf_send));

                                _itoa(i_result_min,buf_send,10);
                                            buf[1]=buf_send[0];
//send data 2
                                            buf[2]=buf_send[1];
//send data 3

                                      }
                                      else
                                      {
                                            buf[0]='1';
                                            buf[1]='9';
                                            buf[2]='9';
                                      }
                                }
//id bus stop 2 number bus 1
                                      MySql_Search_Bus();
                                memset(&test, 0, sizeof(test));
                                sprintf(test,"%d",id_bus_stop_wait);

                                buf[3]=test[0];

            }
            m_comp.SetOutput(COleVariant("B"));
//SEND TO MCS-51 BUS STOP NUMBER 2 ( id bus 1)
            Sleep(1);
            m_comp.SetOutput(COleVariant(buf));
            m_output="";
            for (int i=0;i < 4;i++)
            m_output=m_output+buf[i];
            memset(&buf, 0, sizeof(buf));
            id_bus_stop=2;
            id_bus=201;
            MySql_Calc();
            i_test4=0;
            i_test2=numofrows;
            //sprintf(test,"%d",i_test2);
```

```
            //MessageBox(test);
            for (i_test=0;i_test < (i_test2-1);i_test++)
//fine difference among times in bus stop
            {
            i_test3[i_test] = mysql_time_row[i_test+1] -
mysql_time_row[i_test];
            i_test4 = i_test4 + i_test3[i_test];
            }


            if (i_test2==0)

// in case no data or initial data
            {
                  buf[0]='1';
                  buf[1]='0';
                  buf[2]='0';
                  buf[3]='0';
            }
            else
            {
                  i_result_sec = i_test4/(i_test2-1);

// average result in second
                  i_result_min = i_result_sec/60;
// average result in minute
            i_result_percent=abs(i_result_min - FIXED_id_bus_stop1_2) ;
      f_result=100 - ((i_result_percent / FIXED_id_bus_stop1_2) * 100);
// calculate percentage
                        if (f_result > 10)

// check fault of time not over than 10 percent
                              i_result_min=FIXED_id_bus_stop1_2;
                        else
                              Sleep(1);
                        if (i_result_min < 10)
// if result more than 10 minute
                        {
                              buf[0]='2';
//send data 1
                        memset(&buf_send, 0, sizeof(buf_send));
                              _itoa(i_result_min,buf_send,10);
                              buf[1]='0';
//send data 2
                              buf[2]=buf_send[0];
//send data 3
                        }
                        else
                        {
                              if (i_result_min<99)
// if result over 99 minute
                              {
                                    buf[0]='2';
//send data 1
                        memset(&buf_send, 0, sizeof(buf_send));

                        _itoa(i_result_min,buf_send,10);
```

```
                                        buf[1]=buf_send[0];
                                              //send data 2
                                        buf[2]=buf_send[1];
                                        //send data 3
                                  }
                                  else
                                  {
                                        buf[0]='2';
                                        buf[1]='9';
                                        buf[2]='9';
                                  }
                            }
//id bus stop 2 number bus 2
                                        MySql_Search_Bus();
                                  memset(&test, 0, sizeof(test));
                                  sprintf(test,"%d",id_bus_stop_wait);
                                  buf[3]=test[0];

            }
            m_comp.SetOutput(COleVariant("b"));
//SEND TO MCS-51 BUS STOP NUMBER 2 ( id bus 2)
            Sleep(1);
            m_comp.SetOutput(COleVariant(buf));
            for ( i=0;i < 4;i++)
            m_output=m_output+buf[i];
            }
//send data  bus stop number 3
            else if (m_number==3)

            {
            int i_test,i_test3[1000],i_test4,i_result_min,i_result_sec;
            int i_test2;
            float i_result_percent,f_result;
            char test[20],buf_send[5];
            id_bus_stop=3;
            id_bus=101;
            MySql_Calc();
            i_test4=0;
            i_test2=numofrows;
            for (i_test=0;i_test < (i_test2-1);i_test++)
//fine difference among times in bus stop
            {
i_test3[i_test] = mysql_time_row[i_test+1] - mysql_time_row[i_test];
            i_test4 = i_test4 + i_test3[i_test];
            }
            if (i_test2==0)

// in case no data or initial data
            {
                  buf[0]='1';
                  buf[1]='0';
                  buf[2]='0';
                  buf[3]='0';
            }
            else
            {
```

```
                i_result_sec = i_test4/(i_test2-1);

// average result in second
              i_result_min = i_result_sec/60;
// average result in minute
           i_result_percent=abs(i_result_min - FIXED_id_bus_stop1_2) ;
      f_result=100 - ((i_result_percent / FIXED_id_bus_stop1_2) * 100);
// calculate percentage
                              if (f_result > 10)

// check fault of time not over than 10 percent
                                    i_result_min=FIXED_id_bus_stop1_2;
                              else
                                    Sleep(1);
                              if (i_result_min < 10)
// if result more than 10 minute
                              {
                                    buf[0]='1';
//send data 1
                                    memset(&buf_send, 0,
sizeof(buf_send));
                                    _itoa(i_result_min,buf_send,10);
                                    buf[1]='0';
//send data 2
                                    buf[2]=buf_send[0];
//send data 3
                              }
                              else
                              {
                                    if (i_result_min<99)
                                    // if result over 99 minute
                                    {
                                          buf[0]='1';
//send data 1
                                    memset(&buf_send, 0, sizeof(buf_send));

                                    _itoa(i_result_min,buf_send,10);
                                          buf[1]=buf_send[0];
//send data 2
                                          buf[2]=buf_send[1];
//send data 3

                                    }
                                    else
                                    {
                                          buf[0]='1';
                                          buf[1]='9';
                                          buf[2]='9';
                                    }
                              }
//id bus stop 3 number bus 1
                                    MySql_Search_Bus();
                                    memset(&test, 0, sizeof(test));
                                    sprintf(test,"%d",id_bus_stop_wait);
                                    buf[3]=test[0];
            }
            m_comp.SetOutput(COleVariant("C"));
```

```
//SEND TO MCS-51 BUS STOP NUMBER 3 ( id bus 1)
            Sleep(1);
            m_comp.SetOutput(COleVariant(buf));
            m_output="";
            for (int i=0;i < 4;i++)
            m_output=m_output+buf[i];
            memset(&buf, 0, sizeof(buf));
            id_bus_stop=3;
            id_bus=201;
            MySql_Calc();
            i_test4=0;
            i_test2=numofrows;
            for (i_test=0;i_test < (i_test2-1);i_test++)
//fine difference among times in bus stop
            {
            i_test3[i_test] = mysql_time_row[i_test+1] -
mysql_time_row[i_test];
            i_test4 = i_test4 + i_test3[i_test];
            }
            if (i_test2==0)

// in case no data or initial data
            {
                  buf[0]='1';
                  buf[1]='0';
                  buf[2]='0';
                  buf[3]='0';
            }
            else
            {
                  i_result_sec = i_test4/(i_test2-1);

// average result in second
                  i_result_min = i_result_sec/60;
// average result in minute
            i_result_percent=abs(i_result_min - FIXED_id_bus_stop1_2) ;
      f_result=100 - ((i_result_percent / FIXED_id_bus_stop1_2) * 100);
// calculate percentage
                              if (f_result > 10)

// check fault of time not over than 10 percent
                                    i_result_min=FIXED_id_bus_stop1_2;
                              else
                                    Sleep(1);
                              if (i_result_min < 10)
// if result more than 10 minute
                              {
                                    buf[0]='2';
//send data 1
                              memset(&buf_send, 0, sizeof(buf_send));
                                    _itoa(i_result_min,buf_send,10);
                                    buf[1]='0';
//send data 2
                                    buf[2]=buf_send[0];
//send data 3
                              }
                              else
```

```
                                {
                                        if (i_result_min<99)
// if result over 99 minute
                                        {
                                                buf[0]='2';
//send data 1
                                memset(&buf_send, 0, sizeof(buf_send));

                                _itoa(i_result_min,buf_send,10);
                                                buf[1]=buf_send[0];
//send data 2
                                                buf[2]=buf_send[1];
//send data 3
                                        }
                                        else
                                        {
                                                buf[0]='2';
                                                buf[1]='9';
                                                buf[2]='9';
                                        }
                                }
//id bus stop 3 number bus 2
                                        MySql_Search_Bus();
                                        memset(&test, 0, sizeof(test));
                                        sprintf(test,"%d",id_bus_stop_wait);
                                        //AfxMessageBox(test);
                                        buf[3]=test[0];

                        }
                        m_comp.SetOutput(COleVariant("c"));
//SEND TO MCS-51 BUS STOP NUMBER 3 ( id bus 2)
                        Sleep(1);
                        m_comp.SetOutput(COleVariant(buf));
                        for ( i=0;i < 4;i++)
                        m_output=m_output+buf[i];
                        }
//send data  bus stop number 1 new loop
                        else

                        {
                        int i_test,i_test3[1000],i_test4,i_result_min,i_result_sec;
                        int i_test2;
                        float i_result_percent,f_result;
                        char test[20],buf_send[5];
                        id_bus_stop=1;
                        id_bus=101;
                        MySql_Calc();
                        i_test4=0;
                        i_test2=numofrows;
                        for (i_test=0;i_test < (i_test2-1);i_test++)
//fine difference among times in bus stop
                        {
i_test3[i_test] = mysql_time_row[i_test+1] - mysql_time_row[i_test];
                        i_test4 = i_test4 + i_test3[i_test];
                        }
```

```
            if (i_test2==0)

// in case no data or initial data
            {
                  buf[0]='1';
                  buf[1]='0';
                  buf[2]='0';
                  buf[3]='0';
            }
            else
            {
                  i_result_sec = i_test4/(i_test2-1);
// average result in second
                i_result_min = i_result_sec/60;
// average result in minute
            i_result_percent=abs(i_result_min - FIXED_id_bus_stop0_1) ;
      f_result=100 - ((i_result_percent / FIXED_id_bus_stop0_1) * 100);
// calculate percentage
                              if (f_result > 10)
// check fault of time not over than 10 percent
                                    i_result_min=FIXED_id_bus_stop0_1;
                              else
                                    Sleep(1);
                              if (i_result_min < 10)
// if result more than 10 minute
                              {
                                    buf[0]='1';
//send data 1
                              memset(&buf_send, 0, sizeof(buf_send));
                               _itoa(i_result_min,buf_send,10);
                                    buf[1]='0';
//send data 2
                                    buf[2]=buf_send[0];
//send data 3
                              }
                              else
                              {
                                    if (i_result_min<99)
// if result over 99 minute
                                    {
                                          buf[0]='1';
                                                      //send data 1
                              memset(&buf_send, 0, sizeof(buf_send));

                               _itoa(i_result_min,buf_send,10);
                                          buf[1]=buf_send[0];
//send data 2
                                          buf[2]=buf_send[1];
//send data 3

                                    }
                                    else
                                    {
                                          buf[0]='1';
                                          buf[1]='9';
                                          buf[2]='9';
                                    }
```

```
                                    )
                                                buf[3]='0';
//send data 4
            }
            m_comp.SetOutput(COleVariant("A"));
//SEND TO MCS-51 BUS STOP NUMBER 1 ( id bus 1)
            Sleep(1);
            m_comp.SetOutput(COleVariant(buf));
            m_output="";
            for (int i=0;i < 4;i++)
            m_output=m_output+buf[i];
            memset(&buf, 0, sizeof(buf));
            id_bus_stop=1;
            id_bus=201;
            MySql_Calc();
            i_test4=0;
            i_test2=numofrows;
            for (i_test=0;i_test < (i_test2-1);i_test++)
//fine difference among times in bus stop
            {
            i_test3[i_test] = mysql_time_row[i_test+1] -
mysql_time_row[i_test];
            i_test4 = i_test4 + i_test3[i_test];
            }
            if (i_test2==0)
// in case no data or initial data
            {
                  buf[0]='1';
                  buf[1]='0';
                  buf[2]='0';
                  buf[3]='0';
            }
            else
            {
                  i_result_sec = i_test4/(i_test2 - 1);
// average result in second
                i_result_min = i_result_sec/60;
// average result in minute
            i_result_percent=abs(i_result_min - FIXED_id_bus_stop0_1) ;
      f_result=100 - ((i_result_percent / FIXED_id_bus_stop0_1) * 100);
// calculate percentage
                              if (f_result > 10)
// check fault of time not over than 10 percent
                                    i_result_min=FIXED_id_bus_stop0_1;
                              else
                                    Sleep(1);
                              if (i_result_min < 10)
// if result more than 10 minute
                              {
                                    buf[0]='2';
//send data 1
                                    memset(&buf_send, 0,
sizeof(buf_send));
                                    _itoa(i_result_min,buf_send,10);
                                    buf[1]='0';
//send data 2
```

```
                                        buf[2]=buf_send[0];
//send data 3
                                }
                                else
                                {
                                        if (i_result_min<99)
// if result over 99 minute
                                        {
                                                buf[0]='2';
//send data 1
                                memset(&buf_send, 0, sizeof(buf_send));

                                _itoa(i_result_min,buf_send,10);
                                                buf[1]=buf_send[0];
//send data 2
                                                buf[2]=buf_send[1];
//send data 3
                                        }
                                        else
                                        {
                                                buf[0]='2';
                                                buf[1]='9';
                                                buf[2]='9';
                                        }
                                }
                                                buf[3]='0';
//send data 4
                        }
                        m_comp.SetOutput(COleVariant("a"));
//SEND TO MCS-51 BUS STOP NUMBER 1 ( id bus 2)
                        Sleep(1);
                        m_comp.SetOutput(COleVariant(buf));
                        for ( i=0;i < 4;i++)
                        m_output=m_output+buf[i];
                        m_number=1;
                        }
                        UpdateData(FALSE);
}
void CSerial_projectDlg::OnExit()
{
                        AfxMessageBox("Stop Bus Arriving System");
                        DestroyWindow();
}
void CSerial_projectDlg::MySql_Add()
{
//   start mysql
                        time_t start;
//timer & date
                        time( &start );
                        _tzset();
                        os_date=_strdate( dpbuf );
                        os_time=_strtime( tpbuf );
                        UpdateData(FALSE);
                        time1=start;
//receive to sql
                        link = mysql_init(NULL);
```

```
      mysql_real_connect(link,"localhost","","","bus_system",0,NULL,0);
      sprintf(sql,"INSERT INTO `table1` ( `id_bus` , `id_bus_stop` ,
`time1` , `os_date` , `os_time` ) VALUES
('%d','%d','%d','%s','%s')",id_bus,id_bus_stop,time1,os_date,os_time);
            mysql_query(link,sql);
            result = mysql_store_result(link);
            mysql_close(link);
}

void CSerial_projectDlg::MySql_Calc()
{
//query from mysql
            link = mysql_init(NULL);
            int i,convert_i;

      mysql_real_connect(link,"localhost","","","bus_system",0,NULL,0);
      sprintf(sql,"SELECT id_bus , id_bus_stop , time1 FROM table1
WHERE id_bus='%d' AND id_bus_stop='%d' ORDER BY `os_time` ASC
",id_bus,id_bus_stop);
            mysql_query(link,sql);
            result = mysql_store_result(link);
            numofrows = mysql_num_rows(result);
            i=0;
            while( row = mysql_fetch_row(result) )
            {
                  sprintf(buffer,"%s %s %s %s \n",row[0],row[1],row[2]);
                  convert_i=atoi(row[2]);
                  mysql_time_row[i]=convert_i;
                  i=i+1;
            }
             mysql_close(link);
}

void CSerial_projectDlg::MySql_Search_Bus()
{
//query from mysql
            int i,convert_i;
            int clone_bus=2;
            memset(&mysql_bus_stop_row, 0, sizeof(mysql_bus_stop_row));

            swap_search = id_bus;
            for (int bus_loop=0;bus_loop < clone_bus;bus_loop++)
            {
                  link = mysql_init(NULL);
      mysql_real_connect(link,"localhost","","","bus_system",0,NULL,0);
                  sprintf(sql,"SELECT id_bus , id_bus_stop , time1 FROM
table1 WHERE id_bus='%d'  ORDER BY `os_time` ASC  ",id_bus);
                  mysql_query(link,sql);
                  result = mysql_store_result(link);
                  numofrows = mysql_num_rows(result);
                  id_bus=id_bus+1;
                  i=0;
                  while( row = mysql_fetch_row(result) )
                  {
                 sprintf(buffer,"%s %s %s %s \n",row[0],row[1],row[2]);
                        convert_i=atoi(row[1]);
```

```
// get id_bus_stop
                        mysql_bus_stop_row[bus_loop] = convert_i;
                        i=i+1;
                  }
                        if (mysql_bus_stop_row[bus_loop] == 0)
                              mysql_bus_stop_row[bus_loop] = 999;
             mysql_close(link);
            }
                  bus_loop=0;
                  id_bus_stop_wait=0;
                  int id_bus_stop_wait1=0,id_bus_stop_wait2=0;
                  if (mysql_bus_stop_row[bus_loop] > id_bus_stop)
                  {
                        if  (mysql_bus_stop_row[bus_loop+1]>  id_bus_stop)
                              id_bus_stop_wait=0;
                         else
                              id_bus_stop_wait=id_bus_stop -
      mysql_bus_stop_row[bus_loop+1];
                  }
                  else
                  {
                              id_bus_stop_wait1=id_bus_stop -
mysql_bus_stop_row[bus_loop];
                  if (mysql_bus_stop_row[bus_loop+1] > id_bus_stop)
                        id_bus_stop_wait=id_bus_stop_wait1;
                     else
                        {
      id_bus_stop_wait2=id_bus_stop - mysql_bus_stop_row[bus_loop+1];

      if ((id_bus_stop_wait1 == 0) || (id_bus_stop_wait2==0))
                              {
                        if (id_bus_stop_wait1 > id_bus_stop_wait2)
                              id_bus_stop_wait = id_bus_stop_wait1;
                              else
                              id_bus_stop_wait = id_bus_stop_wait2;
                        }
                        else
                              {
                             if  (id_bus_stop_wait1 > id_bus_stop_wait2)
                              id_bus_stop_wait = id_bus_stop_wait2;
                              else
                              id_bus_stop_wait = id_bus_stop_wait1;
                                    }
                              }
                  )
                              id_bus = swap_search ;

}
```

```
                // serial_projectDlg.h : header file
//
//{{AFX_INCLUDES()
#include "mscomm.h"
//}}AFX_INCLUDES
#include "mysql.h"
#if !defined(AFX_SERIAL_PROJECTDLG_H__228D2A7E_33E7_4A24_B3C1_5412F85B7
5DA__INCLUDED_)
#define
AFX_SERIAL_PROJECTDLG_H__228D2A7E_33E7_4A24_B3C1_5412F85B75DA__INCLUDED
_

#if _MSC_VER > 1000
#pragma once
#endif // _MSC_VER > 1000

/////////////////////////////////////////////////////////////////////////
//////
// CSerial_projectDlg dialog

class CSerial_projectDlg : public CDialog
{
// Construction
public:
      void MySql_Search_Bus();
      void MySql_Calc();
      void MySql_Add();
      CString    m_data;
      MYSQL *link;
      MYSQL_RES *result;
      MYSQL_ROW row;
      int id_bus,id_bus_stop,id_bus_spc,id_bus_stop_wait;
      __int64 numofrows;
      char sql[256],buffer[256];
      char tpbuf[128],dpbuf[128],* os_date,* os_time;
      int
time1,swap,swap_search,time_calc,mysql_time_row[1000],mysql_bus_stop_ro
w[1000];
      int
FIXED_id_bus_stop0_1,FIXED_id_bus_stop1_2,FIXED_id_bus_stop2_3;
      CSerial_projectDlg(CWnd* pParent = NULL); // standard constructor

// Dialog Data
      //{{AFX_DATA(CSerial_projectDlg)
      enum { IDD = IDD_SERIAL_PROJECT_DIALOG };
      CMSComm     m_comp;
      CString     m_output;
      CString     m_input;
      int         m_number;
      CString     m_Show;
      //}}AFX_DATA

      // ClassWizard generated virtual function overrides
      //{{AFX_VIRTUAL(CSerial_projectDlg)
      protected:
      virtual void DoDataExchange(CDataExchange* pDX);      // DDX/DDV
support
```

```
//}}AFX_VIRTUAL

// Implementation
protected:
	HICON m_hIcon;

	// Generated message map functions
	//{{AFX_MSG(CSerial_projectDlg)
	virtual BOOL OnInitDialog();
	afx_msg void OnPaint();
	afx_msg HCURSOR OnQueryDragIcon();
	afx_msg void OnOnCommMscomm1();
	afx_msg void OnSend();
	afx_msg void OnExit();
	DECLARE_EVENTSINK_MAP()
	//}}AFX_MSG
	DECLARE_MESSAGE_MAP()
};

//{{AFX_INSERT_LOCATION}}
// Microsoft Visual C++ will insert additional declarations immediately
before the previous line.

#endif
// !defined(AFX_SERIAL_PROJECTDLG_H__228D2A7E_33E7_4A24_B3C1_5412F85B75
DA__INCLUDED_)
```

```
// Machine generated IDispatch wrapper class(es) created by Microsoft
Visual C++

// NOTE: Do not modify the contents of this file.  If this class is
regenerated by
//  Microsoft Visual C++, your modifications will be overwritten.


#include "stdafx.h"
#include "mscomm.h"

/////////////////////////////////////////////////////////////////////////////
//////
// CMSComm

IMPLEMENT_DYNCREATE(CMSComm, CWnd)

/////////////////////////////////////////////////////////////////////////////
//////
// CMSComm properties

/////////////////////////////////////////////////////////////////////////////
//////
// CMSComm operations

void CMSComm::SetCDHolding(BOOL bNewValue)
{
      static BYTE parms[] =
            VTS_BOOL;
      InvokeHelper(0x1, DISPATCH_PROPERTYPUT, VT_EMPTY, NULL, parms,
            bNewValue);
}

BOOL CMSComm::GetCDHolding()
{
      BOOL result;
      InvokeHelper(0x1, DISPATCH_PROPERTYGET, VT_BOOL, (void*)&result,
NULL);
      return result;
}

void CMSComm::SetCommID(long nNewValue)
{
      static BYTE parms[] =
            VTS_I4;
      InvokeHelper(0x3, DISPATCH_PROPERTYPUT, VT_EMPTY, NULL, parms,
            nNewValue);
}

long CMSComm::GetCommID()
{
      long result;
      InvokeHelper(0x3, DISPATCH_PROPERTYGET, VT_I4, (void*)&result,
NULL);
      return result;
}
```

```
void CMSComm::SetCommPort(short nNewValue)
{
	static BYTE parms[] =
		VTS_I2;
	InvokeHelper(0x4, DISPATCH_PROPERTYPUT, VT_EMPTY, NULL, parms,
		nNewValue);
}

short CMSComm::GetCommPort()
{
	short result;
	InvokeHelper(0x4, DISPATCH_PROPERTYGET, VT_I2, (void*)&result,
NULL);
	return result;
}

void CMSComm::SetCTSHolding(BOOL bNewValue)
{
	static BYTE parms[] =
		VTS_BOOL;
	InvokeHelper(0x5, DISPATCH_PROPERTYPUT, VT_EMPTY, NULL, parms,
		bNewValue);
}

BOOL CMSComm::GetCTSHolding()
{
	BOOL result;
	InvokeHelper(0x5, DISPATCH_PROPERTYGET, VT_BOOL, (void*)&result,
NULL);
	return result;
}

void CMSComm::SetDSRHolding(BOOL bNewValue)
{
	static BYTE parms[] =
		VTS_BOOL;
	InvokeHelper(0x7, DISPATCH_PROPERTYPUT, VT_EMPTY, NULL, parms,
		bNewValue);
}

BOOL CMSComm::GetDSRHolding()
{
	BOOL result;
	InvokeHelper(0x7, DISPATCH_PROPERTYGET, VT_BOOL, (void*)&result,
NULL);
	return result;
}

void CMSComm::SetDTREnable(BOOL bNewValue)
{
	static BYTE parms[] =
		VTS_BOOL;
	InvokeHelper(0x9, DISPATCH_PROPERTYPUT, VT_EMPTY, NULL, parms,
		bNewValue);
}

BOOL CMSComm::GetDTREnable()
```

```
{
      BOOL result;
      InvokeHelper(0x9, DISPATCH_PROPERTYGET, VT_BOOL, (void*)&result,
NULL);
      return result;
}

void CMSComm::SetHandshaking(long nNewValue)
{
      static BYTE parms[] =
            VTS_I4;
      InvokeHelper(0xa, DISPATCH_PROPERTYPUT, VT_EMPTY, NULL, parms,
            nNewValue);
}

long CMSComm::GetHandshaking()
{
      long result;
      InvokeHelper(0xa, DISPATCH_PROPERTYGET, VT_I4, (void*)&result,
NULL);
      return result;
}

void CMSComm::SetInBufferSize(short nNewValue)
{
      static BYTE parms[] =
            VTS_I2;
      InvokeHelper(0xb, DISPATCH_PROPERTYPUT, VT_EMPTY, NULL, parms,
            nNewValue);
}

short CMSComm::GetInBufferSize()
{
      short result;
      InvokeHelper(0xb, DISPATCH_PROPERTYGET, VT_I2, (void*)&result,
NULL);
      return result;
}

void CMSComm::SetInBufferCount(short nNewValue)
{
      static BYTE parms[] =
            VTS_I2;
      InvokeHelper(0xc, DISPATCH_PROPERTYPUT, VT_EMPTY, NULL, parms,
            nNewValue);
}

short CMSComm::GetInBufferCount()
{
      short result;
      InvokeHelper(0xc, DISPATCH_PROPERTYGET, VT_I2, (void*)&result,
NULL);
      return result;
}

void CMSComm::SetBreak(BOOL bNewValue)
{
```

```
        static BYTE parms[] =
              VTS_BOOL;
        InvokeHelper(0xd, DISPATCH_PROPERTYPUT, VT_EMPTY, NULL, parms,
              bNewValue);
}

BOOL CMSComm::GetBreak()
{
        BOOL result;
        InvokeHelper(0xd, DISPATCH_PROPERTYGET, VT_BOOL, (void*)&result,
NULL);
        return result;
}

void CMSComm::SetInputLen(short nNewValue)
{
        static BYTE parms[] =
              VTS_I2;
        InvokeHelper(0xe, DISPATCH_PROPERTYPUT, VT_EMPTY, NULL, parms,
              nNewValue);
}

short CMSComm::GetInputLen()
{
        short result;
        InvokeHelper(0xe, DISPATCH_PROPERTYGET, VT_I2, (void*)&result,
NULL);
        return result;
}

void CMSComm::SetNullDiscard(BOOL bNewValue)
{
        static BYTE parms[] =
              VTS_BOOL;
        InvokeHelper(0x10, DISPATCH_PROPERTYPUT, VT_EMPTY, NULL, parms,
              bNewValue);
}

BOOL CMSComm::GetNullDiscard()
{
        BOOL result;
        InvokeHelper(0x10, DISPATCH_PROPERTYGET, VT_BOOL, (void*)&result,
NULL);
        return result;
}

void CMSComm::SetOutBufferSize(short nNewValue)
{
        static BYTE parms[] =
              VTS_I2;
        InvokeHelper(0x11, DISPATCH_PROPERTYPUT, VT_EMPTY, NULL, parms,
              nNewValue);
}

short CMSComm::GetOutBufferSize()
{
        short result;
```

```
      InvokeHelper(0x11, DISPATCH_PROPERTYGET, VT_I2, (void*)&result,
NULL);
      return result;
}

void CMSComm::SetOutBufferCount(short nNewValue)
{
      static BYTE parms[] =
            VTS_I2;
      InvokeHelper(0x12, DISPATCH_PROPERTYPUT, VT_EMPTY, NULL, parms,
            nNewValue);
}

short CMSComm::GetOutBufferCount()
{
      short result;
      InvokeHelper(0x12, DISPATCH_PROPERTYGET, VT_I2, (void*)&result,
NULL);
      return result;
}

void CMSComm::SetParityReplace(LPCTSTR lpszNewValue)
{
      static BYTE parms[] =
            VTS_BSTR;
      InvokeHelper(0x13, DISPATCH_PROPERTYPUT, VT_EMPTY, NULL, parms,
            lpszNewValue);
}

CString CMSComm::GetParityReplace()
{
      CString result;
      InvokeHelper(0x13, DISPATCH_PROPERTYGET, VT_BSTR, (void*)&result,
NULL);
      return result;
}

void CMSComm::SetPortOpen(BOOL bNewValue)
{
      static BYTE parms[] =
            VTS_BOOL;
      InvokeHelper(0x14, DISPATCH_PROPERTYPUT, VT_EMPTY, NULL, parms,
            bNewValue);
}

BOOL CMSComm::GetPortOpen()
{
      BOOL result;
      InvokeHelper(0x14, DISPATCH_PROPERTYGET, VT_BOOL, (void*)&result,
NULL);
      return result;
}

void CMSComm::SetRThreshold(short nNewValue)
{
      static BYTE parms[] =
            VTS_I2;
```

```
        InvokeHelper(0x15, DISPATCH_PROPERTYPUT, VT_EMPTY, NULL, parms,
                nNewValue);
}

short CMSComm::GetRThreshold()
{
        short result;
        InvokeHelper(0x15, DISPATCH_PROPERTYGET, VT_I2, (void*)&result,
NULL);
        return result;
}

void CMSComm::SetRTSEnable(BOOL bNewValue)
{
        static BYTE parms[] =
                VTS_BOOL;
        InvokeHelper(0x16, DISPATCH_PROPERTYPUT, VT_EMPTY, NULL, parms,
                bNewValue);
}

BOOL CMSComm::GetRTSEnable()
{
        BOOL result;
        InvokeHelper(0x16, DISPATCH_PROPERTYGET, VT_BOOL, (void*)&result,
NULL);
        return result;
}

void CMSComm::SetSettings(LPCTSTR lpszNewValue)
{
        static BYTE parms[] =
                VTS_BSTR;
        InvokeHelper(0x17, DISPATCH_PROPERTYPUT, VT_EMPTY, NULL, parms,
                lpszNewValue);
}

CString CMSComm::GetSettings()
{
        CString result;
        InvokeHelper(0x17, DISPATCH_PROPERTYGET, VT_BSTR, (void*)&result,
NULL);
        return result;
}

void CMSComm::SetSThreshold(short nNewValue)
{
        static BYTE parms[] =
                VTS_I2;
        InvokeHelper(0x18, DISPATCH_PROPERTYPUT, VT_EMPTY, NULL, parms,
                nNewValue);
}

short CMSComm::GetSThreshold()
{
        short result;
        InvokeHelper(0x18, DISPATCH_PROPERTYGET, VT_I2, (void*)&result,
NULL);
```

```
        return result;
}

void CMSComm::SetOutput(const VARIANT& newValue)
{
        static BYTE parms[] =
                VTS_VARIANT;
        InvokeHelper(0x19, DISPATCH_PROPERTYPUT, VT_EMPTY, NULL, parms,
                &newValue);
}

VARIANT CMSComm::GetOutput()
{
        VARIANT result;
        InvokeHelper(0x19, DISPATCH_PROPERTYGET, VT_VARIANT,
(void*)&result, NULL);
        return result;
}

void CMSComm::SetInput(const VARIANT& newValue)
{
        static BYTE parms[] =
                VTS_VARIANT;
        InvokeHelper(0x1a, DISPATCH_PROPERTYPUT, VT_EMPTY, NULL, parms,
                &newValue);
}

VARIANT CMSComm::GetInput()
{
        VARIANT result;
        InvokeHelper(0x1a, DISPATCH_PROPERTYGET, VT_VARIANT,
(void*)&result, NULL);
        return result;
}

void CMSComm::SetCommEvent(short nNewValue)
{
        static BYTE parms[] =
                VTS_I2;
        InvokeHelper(0x1b, DISPATCH_PROPERTYPUT, VT_EMPTY, NULL, parms,
                nNewValue);
}

short CMSComm::GetCommEvent()
{
        short result;
        InvokeHelper(0x1b, DISPATCH_PROPERTYGET, VT_I2, (void*)&result,
NULL);
        return result;
}

void CMSComm::SetEOFEnable(BOOL bNewValue)
{
        static BYTE parms[] =
                VTS_BOOL;
        InvokeHelper(0x1c, DISPATCH_PROPERTYPUT, VT_EMPTY, NULL, parms,
                bNewValue);
```

```
}

BOOL CMSComm::GetEOFEnable()
{
      BOOL result;
      InvokeHelper(0x1c, DISPATCH_PROPERTYGET, VT_BOOL, (void*)&result,
NULL);
      return result;
}

void CMSComm::SetInputMode(long nNewValue)
{
      static BYTE parms[] =
            VTS_I4;
      InvokeHelper(0x1d, DISPATCH_PROPERTYPUT, VT_EMPTY, NULL, parms,
            nNewValue);
}

long CMSComm::GetInputMode()
{
      long result;
      InvokeHelper(0x1d, DISPATCH_PROPERTYGET, VT_I4, (void*)&result,
NULL);
      return result;
}
```

```
      ;*******************************************************
      ;*              Program  : Bus Arriving System         *
      ;*              Hardware : CPU 89S8252                 *
      ;*              Complier : SXA51                       *
      ;*******************************************************
            ORG     0000H
            LJMP    MAIN                          ;reset vecter
            ;
MAIN:             MOV     SP,#256-32              ;define stack = 32 byte
CHECK_ID_BUS:     MOV   P2,#0FFH                  ;SET PORT P2 TO INPUT
                  MOV   R3,#08H                   ;CHECK BIT 8 TIME BY R3
                  MOV   A,P2
LOOP_NEXT:        AJMP  CHECK_PULSE               ;CHECK PULSE FROM  P2
LOOP_NEXT_BIT:    ACALL DELAY_833                 ;DELAY NEXT BIT 1200 Hz
                  MOV   A,P2
                  DJNZ  R3,LOOP_NEXT
CHECK_NEXT:       CJNE  A,#00H,GET_BUS
                  MOV   40H,#01H
;MISSING DATA FROM BUS DEFINE ADDRESS 40H=01 (FOR CHECK)
                  SJMP  WAIT_SERVER
GET_BUS:          MOV   40H,#02H
;GET DATA FROM BUS DEFINE ADDRESS 40H=02     (FOR CHECK)
                  MOV   43H,A
;KEEP DATA BUS TO ADDRESS 43H                (FOR SEND)

      ;*******************************************************
      ;*               Set Value For Serial PORT             *
      ;*               Baud Rate 9600 Bps                    *
      ;*******************************************************
WAIT_SERVER:      MOV   TMOD,#00100000B           ;time1 mode2
                  MOV   SCON,#01010000B           ;mode1 serial port
                  MOV   TH1,#0FBH
                  MOV   A,#00H
                  MOV   PCON,A                    ;SMOD = 0
                  CLR   ET1                       ;clear timer1 interrupt
                  SETB  TR1                       ;start timer1
                  CLR   ES
                  CLR   EA
TX2:              LCALL RX_BYTE
                  LCALL DATA_MAP
                  LCALL DISPLAY
                  MOV   R4,40H
                  CJNE  R4,#02H,NOT_BUS
                  MOV   A,43H
;AT ADDRESS 43H WILL GOTTEN NUMBER BUS
                  CJNE  A,#91H,GET_BUS1
;COMPARE IF NUMBER BUS IS NUMBER  1        WILL BE SEND
                  MOV   A,#91H
                  LCALL TX_BYTE
                  LJMP  CHECK_ID_BUS
GET_BUS1:         CJNE  A,#92H,GET_BUS2
;COMPARE IF NUMBER BUS IS NUMBER  2        WILL BE SEND
                  MOV   A,#92H
                  LCALL TX_BYTE
                  LJMP  CHECK_ID_BUS
GET_BUS2:         CJNE  A,#93H,NOT_BUS
```

```
;COMPARE IF NUMBER BUS IS NUMBER  3     WILL BE SEND
                MOV   A,#93H
                LCALL TX_BYTE
                LJMP  CHECK_ID_BUS
NOT_BUS:        MOV   A,#94H
                LCALL TX_BYTE
                LJMP  CHECK_ID_BUS

      ;*********************************************************
      ;*                Send 1-Byte to SERIAL                  *
      ;*                Input   : ACC                          *
      ;*                Output  : Serial port                  *
      ;*********************************************************
TX_BYTE:        PUSH  IE
                CLR   TI
                MOV   SBUF,A
                JNB   TI,$
                CLR   TI
                POP   IE
                RET

      ;*********************************************************
      ;*              Receive Data From SERIAL                 *
      ;*              Input   :  Serial Port                   *
      ;*              Output  :  ACC                           *
      ;*********************************************************
RX_BYTE:        PUSH   IE
                JNB    RI,$                  ; Wait data
                CLR    RI
                MOV    A,SBUF
                POP    IE
                CJNE   A,#41H,RX_BYTE
;number bus 1   @ bus stop 1
                PUSH   IE
                JNB    RI,$                  ; Wait data
                CLR    RI
                MOV    A,SBUF
                MOV     30H,A
                POP    IE
                PUSH   IE
                JNB    RI,$                  ; Wait data
                CLR    RI
                MOV    A,SBUF
                MOV    31H,A
                POP    IE
                PUSH   IE
                JNB    RI,$                  ; Wait data
                CLR    RI
                MOV    A,SBUF
                MOV    32H,A
                POP    IE
                PUSH   IE
                JNB    RI,$                  ; Wait data
                CLR    RI
                MOV    A,SBUF
                MOV    33H,A
                POP    IE
```

```
RX_BYTE2:          PUSH    IE
                   JNB     RI,$              ; Wait data
                   CLR     RI
                   MOV     A,SBUF
                   POP     IE
                   CJNE    A,#61H,MISSING_DATA
; number bus 2     @ bus stop 1
                   PUSH    IE
                   JNB     RI,$              ; Wait data
                   CLR     RI
                   MOV     A,SBUF
                   MOV      34H,A
                   POP     IE
                   PUSH    IE
                   JNB     RI,$              ; Wait data
                   CLR     RI
                   MOV     A,SBUF
                   MOV     35H,A
                   POP     IE
                   PUSH    IE
                   JNB     RI,$              ; Wait data
                   CLR     RI
                   MOV     A,SBUF
                   MOV     36H,A
                   POP     IE
                   PUSH    IE
                   JNB     RI,$              ; Wait data
                   CLR     RI
                   MOV     A,SBUF
                   MOV     37H,A
                   POP     IE
                   MOV     38H,#00H
                   RET
MISSING_DATA:      MOV      34H,#00H                ; CASE LOSS DATA
                   MOV     35H,#00H
                   MOV     36H,#00H
                   MOV     37H,#00H
                   RET

      ;*********************************************************
      ;*              CHECK DATA FROM P2                       *
      ;*********************************************************

CHECK_PULSE:       MOV 70H,#090H
                   MOV R1,#10H
;LOOP CHECK NUMBER BUS FROM 00-16
CHK_N0:            DJNZ R1,CHK_N1
                   LJMP LOOP_NEXT_BIT
CHK_N1:            CJNE A,70H,CHK_N2
;COMPARE DATA FROM 00-16
                   MOV 40H,A
;DATA OF BUS AT ADDRESS 40H
                   LJMP CHECK_NEXT
;CHECK ID BUS
CHK_N2:            INC 70H
                   SJMP CHK_N0
```

```
      ;*********************************************************
      ;*              DELAY 833 FOR WAIT NEXT BIT AT P2        *
      ;*********************************************************

DELAY_833:          MOV R2,#08H                    ;DELAY 833 MICROSECOUND
LOOP3:              MOV R1,#0AH
LOOP2:              MOV R0,#05H
LOOP1:              DJNZ R0,LOOP1
                    DJNZ R1,LOOP2
                    DJNZ R2,LOOP3
                    RET                            ;END DELAY

      ;*********************************************************
      ;*             DATA MAP FOR DISPLAY                      *
      ;*********************************************************

DATA_MAP:           MOV R0,#30H                    ;DISPLAY DATA
CHECK_MAP_1:        MOV 42H,#30H
;ADDRESS DATA TO RECEIVE
                    MOV 41H,#00H
;DATA MAP (31H ----> 00H) TO (37H ---> 09H)
                    MOV A,@R0
                    CJNE A,#00H,CHECK_DATA_MAP
                    RET
CHECK_DATA_MAP:     CJNE A,42H,CHECK_MAP_LOOP
                    MOV  @R0,41H
                    LJMP CHECK_MAP_NEXT
CHECK_MAP_LOOP:     INC  42H
                    INC  41H
                    LJMP CHECK_DATA_MAP
CHECK_MAP_NEXT:     INC  R0
                    LJMP CHECK_MAP_1

      ;*********************************************************
      ;*              DISPLAY DATA ON DISPLAY BOARD            *
      ;*********************************************************

DISPLAY:            MOV  P0,#00H
                    MOV P1,#00H
                    MOV A,31H
                    SWAP A
                    ADD A,30H
                    MOV P0,A
                    MOV P1,03H                     ;COLUMN 1
                    MOV A,33H
                    SWAP A
                    ADD A,32H
                    MOV P0,A
                    MOV P1,0FH                     ;COLUMN 2
                    MOV A,35H
                    SWAP A
                    ADD A,34H
                    MOV P0,A
                    MOV P1,2FH                      ;COLUMN 3
                    MOV A,37H
                    SWAP A
                    ADD A,36H
```

```
MOV P0,A
MOV P1,0FFH                    ;COLUMN 4
RET
SJMP $
END
```

Project.php

```
<?php
header("Content-type: text/vnd.wap.wml");
echo "<?xml version=\"1.0\"?>";
echo "<!DOCTYPE wml PUBLIC \"-//WAPFORUM//DTD WML 1.1//EN\""
   . " \"http://www.wapforum.org/DTD/wml_1.1.xml\">";
?>

<wml>
<card id="splash" title="Welcome To BUS ARIVING SYSTEM">
<onevent type="ontimer">
      <go href="#Serch_Bus" />
</onevent>
<timer name="delay" value="20"/>
<p mode="wrap" align="center">
<b>ยินดีต้อนรับเข้าสู่<br/>BUS ARIVING SYSTEM<b/>
</p>
</card>
    <card id="Serch_Bus" title="Serch Bus System">
        <p>
            <do type="accept">
            <go href="project2.php" method="post">
            <postfield name="id_bus" value="$id_bus"/>
            <postfield name="id_bus_stop" value="$id_bus_stop"/>
            </go>
        </do>
  <p align="center">หมายเลขรถประจำทาง<br/>
<input name="id_bus" format="NNN" maxlength="3" />
</p>
<p align="center">ป้ายรถประจำทางที่รอ<br/>
<input name="id_bus_stop" format="NNN" maxlength="3" />
</p>
        </p>
    </card>
</wml>
```

Project2.php

```
<?php
header("Content-type: text/vnd.wap.wml");
print "<?xml version=\"1.0\"?>";
echo "<!DOCTYPE wml PUBLIC \"-//WAPFORUM//DTD WML 1.1//EN\""
   . " \"http://www.wapforum.org/DTD/wml_1.1.xml\">";
?>
<wml>
<card id="result" title="Result">
        <p>
        <?php
//start time
    $the_date = date("d M Y");
//วันที่
      print "    $the_date";
      print "<br/>";
// determine whether email was properly sent
            print "<br/>หมายเลขรถประจำทาง:$id_bus";
            print "<br/>ป้ายรถประจำทางที่รอ  : $id_bus_stop";
//แปลงค่าจากรถประจำทางสาย 1 เป็น 101  (สาย 1 คันที่ 1) ***เพื่อความสะดวก
            $swap_bus[0]=$id_bus;
            $swap_bus[1]=1;
            $id_bus=implode("0",$swap_bus);
        $link = mysql_connect("localhost", "", "")
//Data Base From MySql
  or die("Could not connect to database!");
  mysql_select_db("bus_system")
//Select From MySql
  or die("Could not select database!");
  $query = "select * from table1 where id_bus=$id_bus and
id_bus_stop=$id_bus_stop  order by os_time ASC ";
  $result = mysql_query($query,$link)
  or die("Query failed:$query");
  $i_row=0;
  while ($line = mysql_fetch_array($result, MYSQL_ASSOC)) {
 $mysql_time_row[$i_row]=$line[time1];
 $i_row=$i_row + 1;
  }
//*****************Calculate ****************
$i_sum=0;
$i_result=0;
  for ($i_loop=0;$i_loop<($i_row-1);$i_loop++)
  {
      $i_sum=$mysql_time_row[$i_loop+1] - $mysql_time_row[$i_loop];
      $i_result=$i_result + $i_sum;
  }
print "<br/><br/>      ผลการค้นหา";
  if ($i_result < 1)
  {
print "<br/><br/>Sorry Not Data On Database";
print "<br/>ขออภัย ไม่มีข้อมูลที่ต้องการค้นหา";
  }
  else      //else main
  {
```

```
    $FIXED_id_bus_stop0_1=30;

    $FIXED_id_bus_stop1_2=20;
    $FIXED_id_bus_stop2_3=15;

         $i_min = ($i_result / ($i_row-1))/60;
            print "<br/><br/>อีกประมาณ $i_min นาที <br/>รถประจำทางสายที่ $swap_bus[0] จะ
เข้ามาถึงป้ายที่ $id_bus_stop";

//*************************หาค่าป้ายรถประจำทางที่จะมาถึง************************//

//จำนวนรถประจำทางแต่ละเบอร์ที่ซ้ำกันกำหนดใช้ 2 คัน(101,102,10X) เพิ่มได้
                    $clone_bus=2;
               for ( $bus_loop=0;$bus_loop < $clone_bus;$bus_loop++)

                {
                    $query = "select * from table1 where id_bus=$id_bus
order by os_time ASC ";
                    $id_bus=$id_bus+1;
                    $result = mysql_query($query,$link)
                    or die("Query failed:$query");
                    $i_row=0;
                     while ($line = mysql_fetch_array($result,
MYSQL_ASSOC))
                    {
                        $mysql_bus_stop_row[$i_row]=$line[id_bus_stop];
                        $i_row=$i_row + 1;
                        $id_bus_array[$bus_loop]=$line[id_bus_stop];
//เก็บค่าป้ายสุดท้ายที่รถประจำทางถึง
                    }
                        if ($id_bus_array[$bus_loop] == 0)
//ถ้าไม่มีค่าให้ใส่ 999
                        $id_bus_array[$bus_loop] = 999;
                }                   //end for
//กรณีมีมากกว่า2คันต้องเพิ่ม loop for
                    $bus_loop=0;
                    if ($id_bus_array[$bus_loop] > $id_bus_stop)
//คันที่ 1 อยู่ป้ายก่อนหน้ารึไม่
                    {
                        if ($id_bus_array[$bus_loop+1] > $id_bus_stop)
//คันที่ 2 อยู่ป้ายก่อนหน้ารึไม่
                            $id_bus_stop_wait=0;
                        else
              $id_bus_stop_wait=$id_bus_stop  -  $id_bus_array[$bus_loop+1];
                     }
                     else
                     {
               $id_bus_stop_wait1=$id_bus_stop - $id_bus_array[$bus_loop];
        if ($id_bus_array[$bus_loop+1] > $id_bus_stop)   //คันที่ 2 อยู่ป้ายก่อนหน้ารึไม่

        $id_bus_stop_wait=$id_bus_stop_wait1;
                              else
                              {
        $id_bus_stop_wait2=$id_bus_stop - $id_bus_array[$bus_loop+1];
//หาว่า wait1 รึ wait2 ใกล้กว่ากัน
```

```
            if (($id_bus_stop_wait1 == 0) || ($id_bus_stop_wait2==0))
//กรณีมี 0
                                    {
                                if ($id_bus_stop_wait1 > $id_bus_stop_wait2)
                                    $id_bus_stop_wait = $id_bus_stop_wait1;
                                            else
                                    $id_bus_stop_wait = $id_bus_stop_wait2;
                                            }
                                           else
                                            {
                                if ($id_bus_stop_wait1 > $id_bus_stop_wait2)
                                    $id_bus_stop_wait = $id_bus_stop_wait2;
                                            else
                                    $id_bus_stop_wait = $id_bus_stop_wait1;
                                            }
                                        }
                        }
print "<br/><br/>ขณะนี้รถประจำทางสายที่ $swap_bus[0] อยู่อีก $id_bus_stop_wait ป้ายถัดจากป้ายที่รอ";
//*****************************สิ้นสุดการหาค่าป้ายรถประจำทางที่จะมาถึง**************************

  }          //end else main
             ?>
        </p>
    </card>
</wml>
```

หนังสืออ้างอิง

[1] พ.อ.เจนวิทย์ เหลืองอร่าม , ปียวิทย์ เหลืองอร่าม, "การเขียนโปรแกรมสำหรับ Wireless Applications ด้วย J2ME", ซีเอ็ดยูเคชั่น จำกัด, กรุงเทพฯ,2546.

[2] พ.ต.ท. สุชาติ กังวารจิตต์, "เครื่องรับส่งและระบบวิทยุสื่อสาร", ซีเอ็ดยูเคชั่น จำกัด, กรุงเทพฯ, 2541.

[3] ปราโมทย์ วาดเขียน, "พื้นฐานการสื่อสารข้อมูล", คณะวิศวกรรมศาสตร์ สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง, กรุงเทพฯ, พิมพ์ครั้งที่3, 2536.

[4] กิตติ ภักดีวัฒนะกุล, "คัมภีร์ PHP", เคทีพี คอมพ์ แอนด์ คอนซัลท์ จำกัด, กรุงเทพฯ, พิมพ์ครั้งที่6, 2547.

[5] ยุทธนา ลีลาศวัฒนกุล, "คู่มือการเขียนโปรแกรม และการใช้งาน Visual C++ 6.0 ฉบับโปรแกรมเมอร์", อินโฟเพรส, กรุงเทพฯ, 2544.