บอร์ดพัฒนาระบบฝังตัวสำหรับการมอนิเตอร์คลื่นไฟฟ้าหัวใจผ่านเครือข่ายอินเตอร์เน็ต และโทรศัพท์มือถือโดยใช้ J2ME

Embedded Board Development for Heart Signal Monitoring via Internet and Mobile Phone Based on J2ME

โดย

นางสาวจิรภัทร์ รัตนะ

นายชินประพล บุตรโลหะกร

นายธนวัฒน์ สกุลหอม

ปริญญานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิศวกรรมศาสตรบัณฑิต

สาขาวิศวกรรมโทรคมนาคม

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

ปีการศึกษา 2548

บอร์ดพัฒนาระบบฝังตัวสำหรับการมอนิเตอร์คลื่นไฟฟ้าหัวใจผ่านเครือข่ายอินเตอร์เน็ต และโทรศัพท์มือถือโดยใช้ J2ME

Embedded Board Development for Heart Signal Monitoring via Internet and Mobile Phone Based on J2ME

โดย

นางสาวจิรภัทร์	รัตนะ	46015005
นายชินประพล	บุตรโลหะกร	46015008
นายธนวัฒน์	สกุลหอม	46015051

อาจารย์ที่ปรึกษา

รศ.ดร. กอบชัย เดชหาญ

อ. ศรวัฒน์ ชิวปรีชา

ปริญญานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิศวกรรมศาสตรบัณฑิต

สาขาวิศวกรรมโทรคมนาคม

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

ปีการศึกษา 2548

ผ่านการตรวจชิ้นงานแล้ว

(ลงชื่อ).....ผู้ตรวจ

ผ่านการตรวจรูปเล่มแล้ว

(ลงชื่อ).....ผู้ตรวจ

ปริญญานิพนธ์ปีการศึกษา 2548

ภาควิชาวิศวกรรมโทรคมนาคม

คณะวิศวกรรมศาสตร์ สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

เรื่อง **บอร์ดพัฒนาระบบฝังตัวสำหรับการมอนิเตอร์คลื่นไฟฟ้าหัวใจผ่านเครือข่ายอินเตอร์เน็ต และโทรศัพท์มือถือโดยใช้ J2ME**

Embedded Board Development for Heart Signal Monitoring via Internet and Mobile Phone Based on J2ME

ผู้จัดทำ

1. นางสาวจิรภัทร์ รัตนะ 46015005
2. นายชินประพล บุตรโลหะกร 46015008
3. นายธนวัฒน์ สกุลหอม 46015051

.....อาจารย์ที่ปรึกษา

(รศ.ดร. กอบชัย เดชหาญ)

.....อาจารย์ที่ปรึกษา

(อ. ศรวัฒน์ ชิวปรีชา)

บอร์ดพัฒนาระบบฝังตัวสำหรับการมอนิเตอร์คลื่นไฟฟ้าหัวใจผ่านเครือข่ายอินเตอร์เน็ตและโทรศัพท์มือถือโดยใช้ J2ME

Embedded Board Development for Heart Signal Monitoring Via Internet and Mobile Phone Based on J2ME

โดย นางสาวจิรภัทร์ รัตนะ 46015005
นายชินประพล บุตรโลหะกร 46015008
นายธนวัฒน์ สกุลหอม 46015051

อาจารย์ที่ปรึกษา รศ.ดร.กอบชัย เดชหาญ
อ.ศรวัฒน์ ชิวปรีชา

บทคัดย่อ

ในปัจจุบัน การติดต่อสื่อสารผ่านอินเตอร์เน็ตมีการใช้งานกันอย่างกว้างขวาง และในอนาคตการทำงานโดยการควบคุมด้วยการส่งข้อมูลต่าง ๆ มีแนวโน้มว่าจะสูงขึ้นเรื่อย ๆ จึงได้จัดทำโครงงานบอร์ดพัฒนาระบบฝังตัวผ่านเครือข่ายอินเตอร์เน็ต ซึ่งสัญญาณคลื่นไฟฟ้าหัวใจเป็นปัจจัยสำคัญที่แพทย์ใช้ในการวินิจฉัยของผู้ป่วย โดยบางกรณีแพทย์และผู้ป่วยอยู่ห่างไกลกันทำให้เกิดความไม่สะดวกในการวินิจฉัย ดังนั้นโครงงานนี้ จึงนำเสนอการออกแบบและสร้างบอร์ดพัฒนาระบบฝังตัวสำหรับให้เครื่องวัดคลื่นไฟฟ้าหัวใจทำงานเสมือนเป็น Web server ซึ่งคลื่นไฟฟ้าหัวใจที่วัดได้จากผู้ป่วยสามารถแสดงผลผ่านโปรแกรม Web browser และบนโทรศัพท์มือถือโดยใช้ J2ME การพัฒนานี้จะทำให้แพทย์สามารถวินิจฉัยอาการของผู้ป่วยได้สะดวกรวดเร็วขึ้น

ABSTRACT

The communication via Internet is widely used to control the signal transmission, this project presents embedded board development on Internet. The heart signal is important factor for doctor to examine the patience. Some case, the doctor is far away from the patience, it is inconvenient to examine. Thus, this project presents a design and implementation of an embedded board development for heart signal monitoring operated as web server. The measured heart waveforms from the patience are able to display via web browser program and on mobile phone by using J2ME. This development is useful for the doctor to examine the patience as quickly as possible.

กิตติกรรมประกาศ

ขอขอบพระคุณ รศ.ดร.กอบชัย เดชหาญ และ อ.ศรวัฒน์ ชิวปรีชา ซึ่งเป็นอาจารย์ที่ปรึกษา ผศ.สุรพันธ์ ยิ้มมั่น อาจารย์ภาควิชาฟิสิกส์อุตสาหกรรมและอุปกรณ์การแพทย์ สถาบันเทคโนโลยีพระจอมเกล้าพระนครเหนือ ผู้เอื้อเฟื้ออุปกรณ์ที่ใช้ในการทดลอง นายวสันต์ มงคลมาลี และ นายเอกสิทธิ์ บุรินทร์กุล ที่ให้คำปรึกษาและแนะนำแนวทางในการดำเนินงานในเรื่องต่าง ๆ จนโครงงานสำเร็จลุล่วงไปได้ด้วยดี

ขอขอบพระคุณพ่อและแม่ที่ให้กำลังใจมาโดยตลอด ได้อบรมสั่งสอน และส่งเสียให้ได้รับการศึกษาจนสำเร็จการศึกษาจนถึงทุกวันนี้

ผู้จัดทำ

สารบัญ

สารบัญ (ต่อ)

หน้า

สารบัญรูปภาพ

สารบัญรูปภาพ (ต่อ)

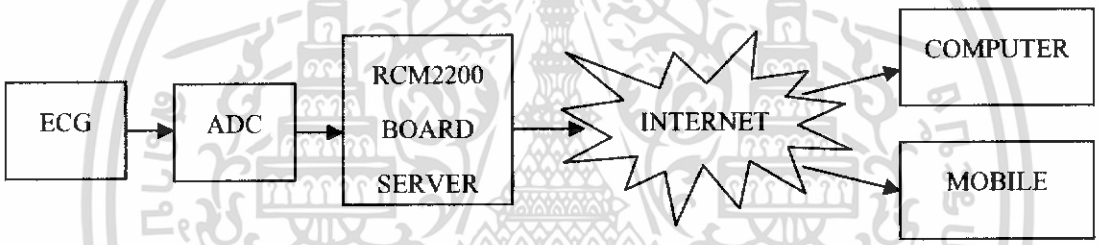
สารบัญรูปภาพ (ต่อ)

สารบัญตาราง

บทที่ 1
บทนำ

1.1 แนวคิดและที่มาของโครงงาน

ปัจจุบันไมโครคอนโทรลเลอร์ (Microcontroller) มีการพัฒนาไปมาก ทำให้มีความสามารถในด้านต่าง ๆ เพิ่มมากขึ้น ซึ่งในงานอุตสาหกรรม ไมโครคอนโทรลเลอร์ได้มีการถูกใช้ในระบบควบคุมอัตโนมัติและกึ่งอัตโนมัติ รวมไปถึงสิ่งอำนวยความสะดวกในชีวิตประจำวัน เนื่องจากคุณสมบัติของไมโครคอนโทรลเลอร์ที่มีขนาดเล็ก ทำการออกแบบและแก้ไขได้ง่ายในระบบของไมโครคอนโทรลเลอร์ที่ได้นำมาใช้ในโครงงานนี้คือ RCM2200 (Compact Rabbit Core with Ethernet) ได้มีการพัฒนาการรับส่งข้อมูลที่รวดเร็วผ่านโปรโตคอล TCP/IP จึงทำให้ RCM2200 มีความสามารถในการสื่อสารผ่านเน็ตเวิร์คได้ จึงเกิดแนวคิดในการที่จะนำเอาไมโครคอนโทรลเลอร์ RCM2200 ใช้ในการส่งข้อมูลผ่านทางเน็ตเวิร์คและในที่นี้ได้นำเอา RCM2200 BOARD SERVER ในการส่งสัญญาณคลื่นไฟฟ้า ECG ผ่านเครือข่าย ดังรูป



รูปที่ 1.1 การเชื่อมต่อใช้งาน RCM2200 ในระบบ Network

1.2 วัตถุประสงค์

เพื่อศึกษาและเรียนรู้การใช้งานไมโครคอนโทรลเลอร์ RCM2200 ในการรับส่งข้อมูลทางเน็ตเวิร์คและนำไปประยุกต์ใช้งาน โดยใช้ไมโครคอนโทรลเลอร์ RCM2200 เป็นตัวกลางระหว่างเครื่องคอมพิวเตอร์และโทรศัพท์เคลื่อนที่ที่ใช้อ่านข้อมูลกับเครื่องวัดคลื่นไฟฟ้าหัวใจ เพื่อเป็นประโยชน์กับผู้ป่วยที่อยู่ห่างไกลเพื่อที่แพทย์จะได้สามารถรับรู้สถานะการเต้นของหัวใจได้ตลอดเวลา

1.3 ขอบเขตของโครงงาน

เพื่อนำไมโครคอนโทรลเลอร์ RCM2200 เป็นตัวเชื่อมต่อและส่งข้อมูลระหว่างเครื่องวัดคลื่นไฟฟ้าหัวใจกับเครื่องคอมพิวเตอร์ที่ใช้อ่านค่าคลื่นไฟฟ้าหัวใจผ่านทางเน็ตเวิร์คโดยการรับการสั่งงานจากหน้าจอเว็บเพจที่ออกแบบไว้ นอกจากนี้ยังได้นำการประยุกต์การเชื่อมต่อกับโทรศัพท์เคลื่อนที่โดยใช้โปรแกรม J2ME เพื่อทำให้มีการแสดงผลบนหน้าจอของโทรศัพท์เคลื่อนที่

1.4 ผลที่คาดว่าจะได้รับ

1. สามารถส่งสัญญาณคลื่นไฟฟ้าหัวใจผ่านระบบอินเตอร์เน็ต (Internet) และโทรศัพท์เคลื่อนที่ได้
2. ทำให้เกิดความสะดวกและรวดเร็วในการส่งข้อมูล
3. สามารถจะพัฒนาการรับส่งข้อมูลให้ดีขึ้นในอนาคต

บทที่ 2
ทฤษฎีและหลักการ

2.1 ทฤษฎีเกี่ยวกับสัญญาณไฟฟ้าหัวใจ

หัวใจเป็นอวัยวะที่สำคัญมากที่สุดอย่างหนึ่งของร่างกาย ทำหน้าที่สูบฉีดโลหิตให้หมุนเวียนไปทั่วร่างกาย โดยที่การหดตัวและพองตัวอย่างสม่ำเสมอของหัวใจเพื่อส่งโลหิตไปทั่วร่างกายนั้นจะถูกควบคุมการทำงานด้วยกล้ามเนื้อพิเศษที่เรียกว่า กล้ามเนื้อหัวใจ (Myocardium) ที่ถูกกระตุ้นด้วยสัญญาณไฟฟ้าจาก Sinoatrial Node การหดตัวและพองตัวดังกล่าวนี้ เกิดไปพร้อมกับศักดาไฟฟ้า (Electric Potential)

2.1.1 สัญญาณไฟฟ้าจากหัวใจ

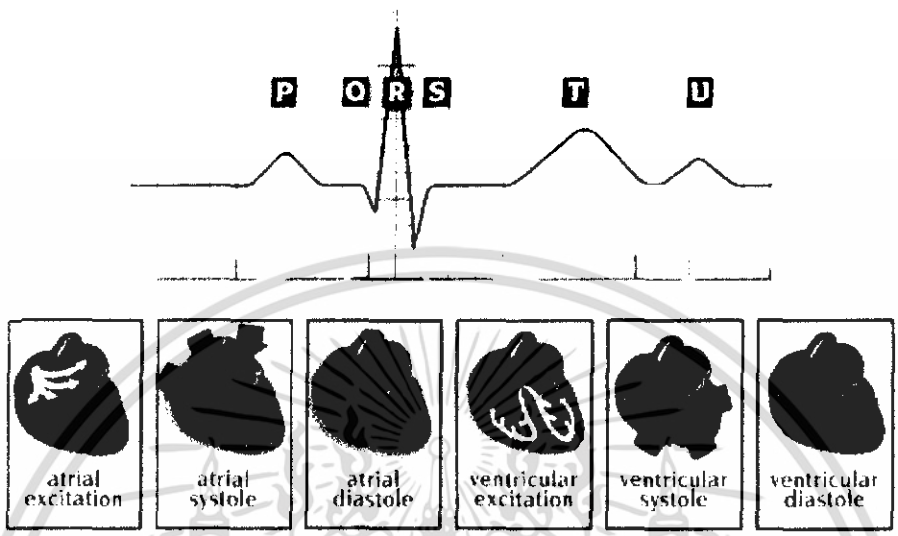
ตามปกติเซลล์กล้ามเนื้อหัวใจจะมีแรงดันไฟฟ้าภายในเซลล์ มีค่าเป็นลบมากกว่าแรงดันไฟฟ้าภายนอกเซลล์ประมาณ 90 มิลลิโวลต์ แสดงตามรูปที่ 2.1 แรงดันไฟฟ้าดังกล่าวเรียกว่า ศักย์ไฟฟ้าขณะอยู่นิ่ง (Resting Potential) ถ้าศักย์ไฟฟ้าขณะอยู่นิ่งมีค่าสูงกว่าขีดจำกัดค่าหนึ่ง จะมีการแตกตัวของอนุภาคและประจุไฟฟ้าเกิดขึ้นเมื่อมีการแตกตัวให้ประจุเกิดขึ้น ศักย์ไฟฟ้าภายในเซลล์มีค่าประมาณ +30 มิลลิโวลต์ และเซลล์กล้ามเนื้อจะมีการหดตัวทำให้เซลล์มีขนาดเล็กลงหลังจากนั้นประมาณ 20 มิลลิวินาที ศักย์ไฟฟ้าภายในเซลล์จะกลับไปมีค่าเท่ากับศักย์ไฟฟ้าขณะหยุดนิ่ง และเซลล์จะอยู่ในลักษณะคลายตัวจนกว่าวัฏจักรจะเริ่มซ้ำ (เมื่อศักย์ไฟฟ้าหยุดนิ่งเพิ่มขึ้นอีก)

หัวใจส่วนบนมีการเกี่ยวพันทางไฟฟ้ากับเซลล์ข้างเคียง ดังนั้นเมื่อเซลล์หนึ่งเกิดการแตกตัวให้ประจุเซลล์ใกล้เคียงจะได้รับการกระตุ้นให้ปลดปล่อยประจุด้วยคลื่นของการปลดปล่อยประจุ จะกระจายไปทั่วส่วนบน ในที่สุดเซลล์ทุกเซลล์ในหัวใจส่วนบนจะมีการแตกตัวให้ประจุ หัวใจส่วนบนจะหดตัว คลื่นของการปลดปล่อยประจุเกิดจากเซลล์จำนวนมากทำให้เกิดความต่างศักย์ไฟฟ้ามากพอที่จะวัดได้โดยการใช้ขั้วไฟฟ้าวางบนผิวหนัง ค่าแรงดันที่วัดได้ด้วยวิธีนี้เรียกว่า อิเล็กโทรคาร์ดิโอแกรม (Electrocardio gram)

ในทำนองเดียวกันกับเหตุการณ์ที่เกิดขึ้นในห้องหัวใจส่วนบน เซลล์ทั้งหมดที่เป็นองค์ประกอบของกล้ามเนื้อหัวใจส่วนล่างทั้ง 2 ห้องจะมีความสัมพันธ์ทางไฟฟ้ากับเซลล์ข้างเคียง ดังนั้น เซลล์ใดเซลล์หนึ่งในหัวใจส่วนล่างมีการแตกตัวและการหดตัวของอนุภาคและให้ประจุออกมาในทุก ๆ เซลล์ของหัวใจส่วนล่าง

อย่างไรก็ตาม หัวใจส่วนบนและหัวใจส่วนล่างไม่ได้เชื่อมต่อกันโดยตรง บริเวณแนวเชื่อมต่อของช่องทางไฟฟ้าของหัวใจส่วนบนและหัวใจส่วนล่างเรียกว่า โหนดเอวี (Atrio Ventricular Node) การส่งผ่านสัญญาณทางไฟฟ้าระหว่างหัวใจส่วนบนและหัวใจส่วนล่าง จะทำให้ทั้ง 2 ส่วนได้รับสัญญาณช้ากว่ากัน 0.04 วินาที การนี้ทำให้หัวใจส่วนบนที่เวลาฉีดเลือดให้หัวใจส่วนล่าง การล่าช้าดังกล่าวยังเป็นตัวจำกัดจำนวนครั้งต่อนาทีที่หัวใจบีบตัวตลอดจากหัวใจส่วนบนจนถึงหัวใจส่วนล่าง ในกรณีที่หัวใจส่วนบนมีการบีบรัดตัวเร็วเกินไป การจำกัดอัตราการบีบรัดตัวของหัวใจส่วนล่างเป็นการทำให้ชีวิต

ให้ชีวิตปลอดภัย ทั้งนี้เพราะว่าการสูบฉีดเลือดของหัวใจส่วนล่างนี้เองที่ทำให้เลือดไหลไปสู่สมองและอวัยวะต่าง ๆ ได้มากที่สุด ถ้าการบีบตัวดังกล่าวเกิดขึ้นเร็วเกินไป การไหลของเลือดจะลดลงเนื่องจากไม่มีเวลามากพอที่จะใช้สูบเลือดเข้าสู่หัวใจส่วนล่าง ซึ่งเวลาดังกล่าวก็คือช่วงเวลาระหว่างการหด



รูปที่ 2.1 แสดงความสัมพันธ์ของการทำงานของหัวใจกับการเกิดคลื่นไฟฟ้าหัวใจ

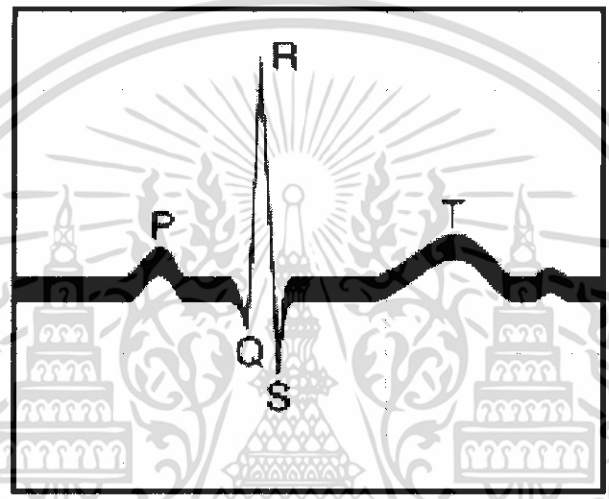
จากรูปที่ 2.1 ซึ่งเป็นแผนภาพวงจรควบคุมการปลดปล่อยประจุของเซลล์กล้ามเนื้อหัวใจ ถ้าไม่มีการกระตุ้นจากภายนอกทำให้เกิดการแตกตัวให้ประจุไฟฟ้า เซลล์หัวใจจะค่อย ๆ เปลี่ยนแปลงจากสภาพที่มีศักย์ไฟฟ้าเท่ากับศักย์ไฟฟ้าขณะพุดนิ่ง มีการแตกตัวของอนุภาพให้ประจุไฟฟ้าด้วยจำนวนคงที่ เซลล์ที่มีการปลดปล่อยประจุไฟฟ้าเร็วที่สุดจะเป็นเซลล์ที่นำ ซึ่งทำให้เกิดการปลดปล่อยประจุที่จุดโหนดเอวี สัญญาณการปลดปล่อยประจุจากเซลล์นำจะกระจายสู่เซลล์ต่าง ๆ ในหัวใจส่วนบนก่อน จากนั้นโหนดเอวีจะนำสัญญาณไฟฟ้าไปยังเส้นใยไฟฟ้า ซึ่งจะนำไฟฟ้าเข้าสู่หัวใจส่วนบนที่จะนำสัญญาณไฟฟ้าไปยังเส้นใยไฟฟ้าซึ่งจะนำไฟฟ้าสู่หัวใจส่วนล่างอย่างรวดเร็ว การกระตุ้นให้เกิดการปลดปล่อยประจุในหัวใจส่วนล่างทั้ง 2 ห้องจะเกิดขึ้นพร้อมกัน โดยเริ่มจากภายในสู่ผนังหัวใจภายนอก เซลล์ในหัวใจตอนบนจะมีแนวโน้มที่จะปลดปล่อยประจุประมาณ 60 100 ครั้งต่อวินาที เซลล์ในโหนดเอวีซึ่งเป็นรอยต่อระหว่างหัวใจส่วนบนกับหัวใจส่วนล่างมีแนวโน้มที่จะปลดปล่อยประจุประมาณ 30 ครั้งต่อวินาที

ดังนั้นการหดตัวของหัวใจส่วนบนจะเกิดขึ้นก่อน ตามด้วยระยะเวลาที่ทิ้งช่วงและการหดตัวของหัวใจส่วนล่าง จากนั้นจะมีระยะพุดพักก่อนที่จะมีการบีบตัวของหัวใจ หรือวัฏจักรการทำงานของหัวใจครั้งต่อไปจะเกิดขึ้น เซลล์กล้ามเนื้อหัวใจมีการปลดปล่อยประจุและหดตัวเป็นลำดับ เป็นจังหวะ เป็นเวลา สัญญาณจากโหนดเอสเอจะเข้ามาแล้วทำให้เกิดการปลดปล่อยประจุและหดตามอัตราที่เป็นลักษณะเฉพาะตัว ดังนั้นโหนดเอวีจึงเป็นผู้นำในการทำงานของหัวใจ ถ้าโหนดเอวีไม่ทำงานหรือสัญญาณที่ส่งมาถูกแนวโหนดเอวีที่เป็นโรคกักไว้ หัวใจห้องล่างจะยังคงบีบรัดตัวได้อย่างมีประสิทธิภาพ เพราะว่าเซลล์

บางเซลล์ในหัวใจส่วนล่างสามารถที่จะปลดปล่อยประจุได้เองและทำตัวเป็นผู้นำในหัวใจส่วนล่าง อัตราการเต้นของหัวใจในลักษณะนี้จะช้า (ประมาณ 30 ครั้งต่อนาที) แต่จังหวะการเต้นของหัวใจเพื่อความอยู่รอดเช่นนี้มักจะเพียงพอที่จำให้ชีวิตรอดได้

2.1.2 ความหมายและรูปร่างของสัญญาณคลื่นไฟฟ้าหัวใจ

ลักษณะของสัญญาณคลื่นไฟฟ้าหัวในที่ปกติ แสดงได้ในรูปที่ 2.2 ภาพคลื่นไฟฟ้าหัวใจที่บันทึกได้ จะเริ่มตั้งแต่ก่อนการบีบตัวของหัวใจจนกระทั่งมีการคลายตัวในแต่ละครั้ง ดังนั้นจึงเกิดสัญญาณขึ้นเป็นจังหวะโดยมีความถี่เท่ากับอัตราการเต้นของหัวใจ



รูปที่ 2.2 แสดงผลอิเล็กโทรคาร์ดิโอแกรมของคนปกติ

ภาพคลื่นไฟฟ้าหัวใจในแต่ละจังหวะประกอบด้วยคลื่นไฟฟ้าย่อย 3 คลื่น คือ

1. ช่วงคลื่น P เป็นผลรวมทางไฟฟ้าขบวนการดีโพลาร์ไรซ์ที่เกิดขึ้นที่หัวใจห้องบนทั้งซ้ายและขวา ซึ่งเกิดก่อนที่หัวใจทั้งสองห้องจะมีการบีบตัว

2. ช่วงคลื่น QRS เป็นผลรวมทางไฟฟ้าจากขบวนการดีโพลาร์ไรซ์ของหัวใจห้องล่างด้านซ้ายและขวาซึ่งเกิดขึ้นก่อนที่หัวใจทั้งสองข้างจะมีการบีบตัว โดยที่ขนาดของคลื่นสัญญาณ R สำหรับการทำงานปกติของหัวใจมีค่าประมาณ 1 มิลลิโวลต์

3. ช่วงคลื่น T เป็นผลรวมทางไฟฟ้าจากขบวนการรีโพลาร์ไรซ์ของหัวใจห้องล่างทั้งซ้ายและขวาและเกิดขึ้นก่อนที่หัวใจทั้งสองห้องจะมีการคลายตัว โดยขนาดของสัญญาณ T มีค่าประมาณ 1/3 ของขนาดของสัญญาณ R

สำหรับขบวนการรีโพลาร์ไรซ์ของหัวใจห้องบน อาจจะเกิดขึ้นในช่วงระหว่างที่หัวใจห้องล่างมีการบีบตัว แต่ค่าขนาดจะไม่ปรากฏเนื่องจากค่าของสัญญาณช่วงคลื่น QRS มีค่ามากกว่า

2.1.3 การตรวจวัดคลื่นไฟฟ้าของหัวใจ

การวัดสัญญาณคลื่นไฟฟ้าหัวใจตามมาตรฐาน มีวิธีวัดอยู่ 2 แบบ คือ

1. แบบไบโพลาร์ลิมบ์ลีด (Bipolar Limb Lead)

วิธีนี้จะวางขั้วไฟฟ้า (Electrode) ระหว่างแขนและขา ซึ่งเป็นการวัดการเปลี่ยนแปลงระหว่างจุด 2 จุดซึ่งมีมาตรฐานของตำแหน่งที่จะวางขั้วไฟฟ้า 3 แบบ ดังแสดงในรูปที่ 2.3

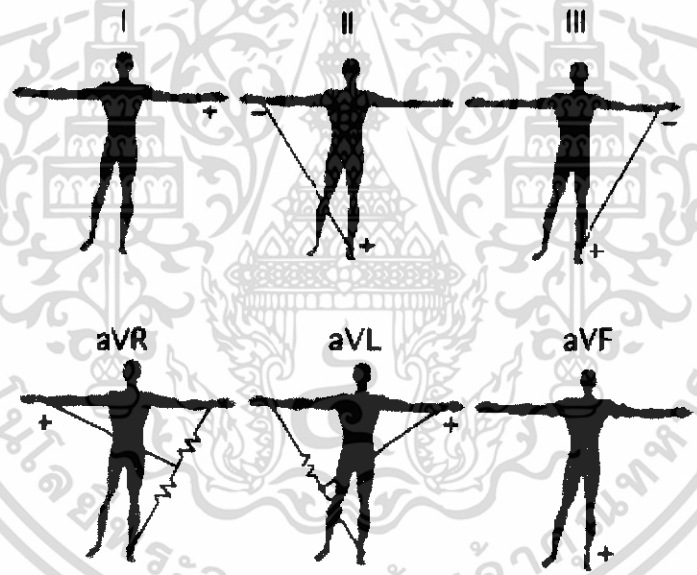
1.1 ลีด I (Lead I) จะใช้ขั้วไฟฟ้าวัดศักดาที่แขนขวาและแขนซ้ายเทียบกัน โดยใช้ศักดาจากขาขวาเป็นสัญญาณอ้างอิง

1.2 ลีด II (Lead II) จะใช้ขั้วไฟฟ้าวัดศักดาที่แขนขวาและแขนซ้ายเทียบกัน โดยใช้ศักดาจากขาขวาและแขนซ้ายเป็นสัญญาณอ้างอิง

1.3 ลีด III (Lead III) จะใช้ขั้วไฟฟ้าวัดศักดาที่แขนขวาและแขนซ้ายเทียบกัน โดยใช้ศักดาจากขาขวาและแขนขวาเป็นสัญญาณอ้างอิง

2. แบบยูนิโพลาร์ลิมบ์ลีด (Unipolar Limb Lead)

วิธีนี้จะสามารถตรวจสอบบริเวณหัวใจได้อย่างสะดวก ดังรูปที่ 2.3 ที่ขั้วไฟฟ้าของแขนและขาจะรวมความต้านทานเข้าด้วยกันเป็นจุด ๆ หนึ่ง และบันทึกการเปลี่ยนแปลงศักดาไฟฟ้าระหว่างขั้วไฟฟ้าทั้ง 6 อันที่วางไว้บนหน้าอก



รูปที่ 2.3 แสดงวิธีวัดคลื่นไฟฟ้าหัวใจทั้ง 2 แบบ

2.1.4 อิเล็กโทรด

การจะวัดศักย์ไฟฟ้าและกระแสไฟฟ้าบนร่างกายจะต้องมีตัวเชื่อม นั่นก็คือ อิเล็กโทรดที่ทำงานเสมือนเป็นทรานสดิวเซอร์ เพราะในร่างกายมีการนำกระแสด้วยไอออน แต่ในเครื่องวัดจะมีการนำกระแสด้วยอิเล็กตรอน ดังนั้นอิเล็กโทรดต้องทำหน้าที่เปลี่ยน Ionic Current ให้เป็น Electric Current

ไดอะแกรมพื้นหน้าระหว่างอิเล็กโทรดและอิเล็กโตรไลท์ แสดงไว้ในรูป กระแสไฟฟ้าจะข้ามจากอิเล็กโทรดไปยังอิเล็กโทรไลท์จะประกอบด้วย

1. อิเล็กตรอนที่เคลื่อนที่ทิศทางตรงข้ามกับกระแสอิเล็กโทรด
2. แคทไอออนเคลื่อนที่ทิศทางเดียวกับกระแสไฟฟ้า
3. แอนไอออนเคลื่อนที่ทิศทางตรงข้ามกับกระแสไฟฟ้าอิเล็กโทรไลท์

สำหรับประจุที่ข้ามพื้นหน้านั้น เนื่องจากไม่มีอิเล็กตรอนอิสระในอิเล็กโทรไลท์และไม่มีแคทไอออนและแอนไอออนในอิเล็กโทรดด้วย จึงต้องมีการเปลี่ยนแปลงเกิดขึ้น เพื่อถ่ายทอดประจุระหว่างพาหะทั้งสอง ดังนั้น อิเล็กโทรไลท์ที่ห่อหุ้มด้วยโลหะจะมีศักย์ไฟฟ้าต่างไป เรียกว่า ศักย์ไฟฟ้าครึ่งเซลล์ (Half – Cell Potential) แต่เราไม่สามารถวัดศักย์ไฟฟ้าครึ่งเซลล์ของอิเล็กโทรดได้ จึงต้องใช้อิเล็กโทรดอีกอันในการเปรียบเทียบศักย์ไฟฟ้า

ศักย์ไฟฟ้าครึ่งเซลล์ของอิเล็กโทรดที่กล่าวมาเป็นภาวะที่ไม่มีกระแสไหล ถ้ามีกระแสไหลศักย์ไฟฟ้าที่วัดจะมีค่าเปลี่ยนแปลงไป ความแตกต่างนี้เป็นผลมาจากโพลาไรเซชั่นของอิเล็กโทรด ความต่างศักย์ไฟฟ้าที่เกิดขึ้น เมื่อเปรียบเทียบกับภาวะสมดุลนั้นเรียกว่า Over Voltage มีกลไกที่เกี่ยวข้องกับกระบวนการ 3 ส่วนคือ

1. Ohmic Over Voltage เป็นผลเนื่องจากความต้านทานของอิเล็กโทรด เมื่อมีกระแสไฟฟ้าผ่านอิเล็กโทรดทั้งสองอัน เมื่อความต้านทานของอิเล็กโทรดเปลี่ยนแปลงตามกระแสไฟฟ้า ศักย์ไฟฟ้าที่เกินทางด้านโอห์ม (Ohm Over Voltage) ก็จะไม่มีความสัมพันธ์เป็นเส้นตรงกับกระแสไฟฟ้าตามกฎของโอห์ม

2. Concentration Over Voltage เกิดจากความเข้มข้นที่เป็นผลมาจากการเปลี่ยนแปลงการกระจายของไอออนในอิเล็กโทรไลท์

3. Activation Over Voltage เป็นผลของการถ่ายทอดประจุของปฏิกิริยาการเติมและการลดออกซิเจน ไม่สามารถเปลี่ยนกลับได้หมด

2.1.4.1 อิเล็กโทรดที่โพลาไรซ์และอิเล็กโทรดที่ไม่มีโพลาไรซ์

ตามทฤษฎีสามารถแบ่งอิเล็กโทรดออกได้เป็น 2 ชนิด คือ

1. อิเล็กโทรดที่โพลาไรซ์ อิเล็กโทรดนี้เมื่อผ่านกระแสไฟฟ้าเข้าไปจะไม่มีกระแสไฟฟ้าข้ามพื้นหน้าของอิเล็กโทรดและอิเล็กโทรไลท์จะทำงานเหมือนเป็นคาปาซิเตอร์

2. อิเล็กโทรดที่ไม่มีโพลาไรซ์ เมื่อมีกระแสผ่านจะสามารถผ่านพื้นหน้าได้อย่างเสรีโดยไม่สูญเสียพลังงาน ดังนั้นจึงไม่เกิด Over Voltage แต่เราไม่สามารถสร้างอิเล็กโทรดที่โพลาไรซ์และอิเล็กโทรดที่ไม่มีโพลาไรซ์ได้อย่างสมบูรณ์

2.1.4.2 คุณสมบัติอิเล็กโทรด

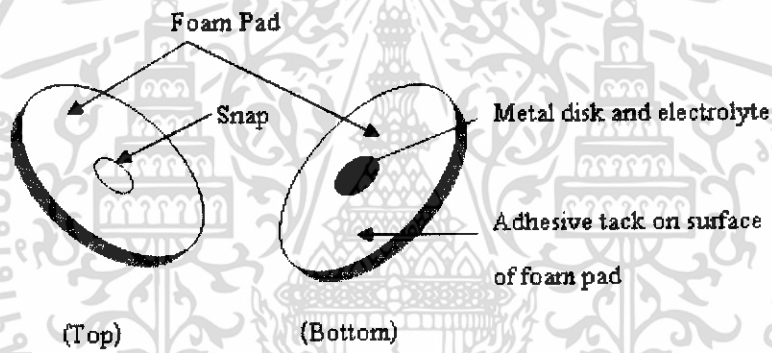
เราจะพบว่าคุณสมบัติกระแสและแรงดันอิเล็กโทรดนั้นไม่เป็นเส้นตรง เพราะอิเล็กโทรดมีคุณสมบัติเฉพาะตัว คือความไวต่อกระแสที่ผ่านอิเล็กโทรด ถ้ามีความเข้มของกระแสมาก คุณสมบัติจะต่างไปจากเดิม และคุณสมบัติยังขึ้นอยู่กับรูปคลื่นไฟฟ้า ถ้าเป็นสัญญาณชายน์ต้องขึ้นกับความถี่ด้วย เพราะอิเล็กโทรดเหมือนมีความต้านทานและตัวเก็บประจุอยู่ด้วย เมื่อนำอิเล็กโทรดมาติดที่ผิวหนัง เราต้องพิจารณาพื้นหน้าระหว่างอิเล็กโทรด อิเล็กโทรไลท์ และผิวหนังด้วย เราใช้ครีมอิเล็กโทรไลท์ที่ประกอบด้วยคลอไรด์ไอออนทาก่อนที่จะติดอิเล็กโทรดเป็นตัวประสาน ปัจจัยหนึ่งที่มีผลต่อคุณสมบัติ

ทางไฟฟ้าของผิวหนังคือ การวัด Grlvinic Skin Reflex (GSR) เพราะเกี่ยวกับเหงื่อและท่อของต่อมเหงื่อที่จะมีโซเดียมโพแทสเซียมและคลอไรด์ไอออนหลั่งจากต่อมเหงื่อ ทำให้เกิดความต่างศักย์ของรูของท่อนำเหงื่อกับผิวหนัง แต่ส่วนประกอบที่กล่าวมาสามารถตัดไปได้กับอิเล็กโทรดที่ใช้วัดทางชีววิทยาธรรมดา ไม่เกี่ยวกับการวัดทางผิวหนัง

2.1.4.3 อิเล็กโทรดแบบแผ่นที่ทำด้วยโลหะ

อิเล็กโทรดที่ใช้น้อยในการรับศักย์ไฟฟ้าทางชีววิทยา คือ อิเล็กโทรดที่ทำด้วยโลหะ โดยการนำแผ่นโลหะมาสัมผัสกับผิวหนัง และมักใช้ครีมอิเล็กโทรไลท์เชื่อมระหว่างกลางเพื่อทำให้มีการสัมผัสที่ดียิ่งขึ้น

รูปที่ 2.4 แสดงอิเล็กโทรดแผ่นโลหะแบบต่าง ๆ เป็นอิเล็กโทรดที่ใช้รวดเร็วและใช้แล้วทิ้ง ทั้งนี้เพื่อประหยัดเวลาและบุคลากรทางด้านนี้ อิเล็กโทรดนี้ประกอบด้วยพลาสติกที่ทำเป็นโฟมและมีแผ่นเงินติดอยู่ข้างหนึ่ง แผ่นเงินนี้อาจเคลือบด้วยซิลเวอร์คลอไรด์ ในการใช้งานนี้ผู้ใช้เพียงแต่ทำความสะอาดผิวหนัง เปิดห่ออิเล็กโทรด ดึงกระดาษที่ปิดอยู่แล้วกดอิเล็กโทรดลงบนผิวหนังทันที



รูปที่ 2.4 แสดงอิเล็กโทรดแผ่นที่ทำด้วยโฟม (ชนิดใช้แล้วทิ้งเลย) สำหรับเครื่อง ECG ใช้ปิดผิวหนังที่ติดแขน - ขา

2.1.4.4 ข้อแนะนำในการใช้อิเล็กโทรดในทางปฏิบัติ

ในการใช้อิเล็กโทรดโลหะสำหรับวัดศักดาทางไฟฟ้าหรือกระตุ้นก็ดี จะต้องคำนึงถึงข้อปฏิบัติ 5 ประการ ดังต่อไปนี้คือ

1. ในการสร้างอิเล็กโทรดรวมทั้งสายไฟที่นำมาต่อ โดยเฉพาะส่วนที่จะต้องสัมผัสกับเนื้อเยื่อของร่างกาย ควรเป็นวัสดุชนิดเดียวกันตลอด เมื่อใช้วัสดุอย่างที่สาม เช่น วัสดุที่ใช้ในการเชื่อมก็ควรจะใช้ฉนวนหุ้มไว้ ไม่ให้สัมผัสกับเนื้อเยื่อหรืออิเล็กโทรไลท์ของร่างกาย โลหะต่างชนิดกันไม่ควรนำมาสัมผัสกันเพราะจะมีศักย์ไฟฟ้าครึ่งเซลล์ต่างกัน นอกจากนี้เมื่อสัมผัสกับอิเล็กโทรไลท์ก็จะทำให้มีปฏิกิริยาเคมีไฟฟ้าเกิดขึ้น เป็นผลให้มีโพราไรเซชั่นเพิ่มเติม และมักจะทำให้อิเล็กโทรดอันหนึ่งถูกกัดกร่อนไป ปัจจัยนี้ทำให้ศักดาไฟฟ้าครึ่งเซลล์มีเสถียรภาพน้อย ทำให้เพิ่มการรบกวนทางไฟฟ้าของอิเล็กโทรดได้

2. เมื่อใช้อิเล็กโทรดคู่ใดคู่หนึ่งสำหรับวัดศักย์ไฟฟ้าในร่างกาย ควรใช้อิเล็กโทรดที่ทำด้วยวัสดุอย่างเดียวกัน เนื่องจากครึ่งเซลล์ศักย์ไฟฟ้าที่เกิดขึ้นมีค่าเท่ากัน ดังนั้นศักย์ไฟฟ้า DC ที่ป้อนเข้าไปยังอินพุทของแอมปลิไฟเออร์จะได้มีค่าน้อยมาก อันเป็นการทำให้การอิ่มตัวของแอมปลิไฟเออร์ไม่เกิดขึ้น โดยเฉพาะเมื่อแอมปลิไฟเออร์ที่ใช้เป็นชนิด DC และมีกำลังขยายสูง

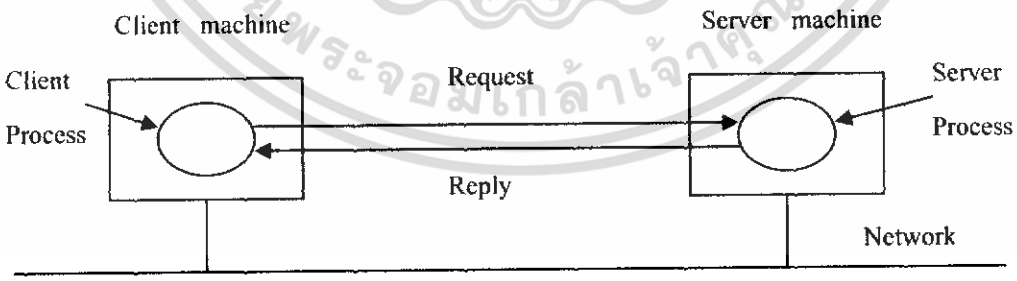
3. อิเล็กโทรดที่ติดบนผิวหนังมักจะหลุดง่าย อย่างไรก็ดี ปัญหาเหล่านี้จะไม่เกิดขึ้น ถ้าอิเล็กโทรดได้รับการออกแบบที่ดี เส้นลวดที่ต่อออกมาจากอิเล็กโทรดควรจะอ่อนตัวได้มาก แต่ต้องแข็งแรง จุดต่อของเส้นลวดที่เข้าไปยังแผ่นอิเล็กโทรดมักจะหลุดง่าย เนื่องมาจากการโค้งของเส้นลวดที่มีอยู่เรื่อย ๆ จะคำนึงถึงข้อนี้ด้วยเสมอในการออกแบบ

4. อิเล็กโทรดมักถูกใช้ในสภาวะแวดล้อมที่มีความชื้นสูง ฉนวนของอิเล็กโทรดเหล่านี้มักทำด้วยวัสดุจากพวกโพลิเมอร์ ซึ่งสามารถดูดน้ำได้ดีเมื่อใช้ไปนาน ๆ

5. การใช้แอมปลิไฟเออร์ที่มีอินพุทสูง ๆ ทำให้การบันทึกไฟได้ผลดี ถ้าอิมพีแดนซ์ของแอมปลิไฟเออร์มีค่าไม่สูงพอ นอกจากจะได้สัญญาณที่มีความถี่สูงลดลงแล้ว ยังมีรูปร่างผิดเพี้ยนไปด้วย

2.2 ทฤษฎีเกี่ยวกับระบบเครือข่ายคอมพิวเตอร์

ระบบเครือข่ายคอมพิวเตอร์ (Computer) หมายความถึงเครื่องคอมพิวเตอร์ตั้งแต่สองเครื่องขึ้นไปที่เป็นอิสระต่อกัน นำมาเชื่อมต่อถึงกันได้โดยไม่คำนึงถึงระยะทางระหว่างเครื่องทั้งสอง หมายถึงการเชื่อมต่อถึงกันนั้นก็ไม่ได้จำกัดเอาไว้ว่าจะต้องใช้แบบใด ขอเพียงให้แลกเปลี่ยนข่าวสารข้อมูลกันได้ ดังรูปที่ 2.5 เป็นระบบผู้ให้บริการและใช้บริการ การสื่อสารในระบบนี้ “ผู้ใช้บริการ” จะส่งความต้องการในลักษณะของการร้องขอ (Request) ไปยัง “ผู้ให้บริการ” ซึ่งเมื่อผู้ให้บริการทำงานตามความต้องการนั้น ๆ เสร็จแล้วจะส่งผลที่ได้กลับมายังผู้ใช้บริการในลักษณะของการตอบสนอง (Reply) สัดส่วนของผู้ให้บริการต่อผู้ใช้บริการนี้ขึ้นอยู่กับลักษณะของงานและประสิทธิภาพที่ต้องการเป็นหลัก



รูปที่ 2.5 ระบบผู้ให้บริการและผู้ใช้บริการ (Client / Server)

ระบบเครือข่ายที่ใช้อยู่ในปัจจุบันคือ ระบบอินเตอร์เน็ต ที่มีต้นกำเนิดมาจากระบบเครือข่ายชื่อ ARPANET โดยใช้สายโทรศัพท์เป็นหลัก ต่อมาเมื่อระบบการสื่อสารแบบคลื่นวิทยุความถี่สูงและการสื่อสารดาวเทียมเริ่มเข้ามามีบทบาทและนำมาใช้ในระบบมากขึ้น ทำให้โปรโตคอลที่เคยใช้ได้ผลดีนั้น

เกิดปัญหาไม่สามารถใช้งานได้อีกต่อไป โปรโตคอลรุ่นต่อมาจึงได้รับการออกแบบเพื่อนำมาใช้ทดแทนแบบเก่า โดยมีวัตถุประสงค์ในการเชื่อมการติดต่อระหว่างระบบที่มีความแตกต่างกันเป็นเรื่องหลัก ผลที่ได้รับคือ โปรโตคอลที่เรียกว่าโปรโตคอลมาตรฐานแบบ TCP/IP ซึ่งได้รับการปรับปรุงจนนำมาใช้งานจริงได้ในปี ค.ศ. 1974 การปรับปรุงรุ่นต่อมาสำเร็จในปี ค.ศ. 1988

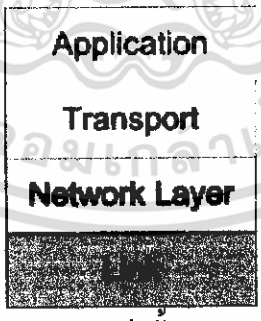
โปรโตคอลมาตรฐานแบบ TCP/IP ยังมีวัตถุประสงค์หลักอีกสองข้อสำคัญคือ ความสามารถในการแก้ไขปัญหาที่เกิดขึ้นในระบบเครือข่าย เช่น ในกรณีที่ผู้ส่งและผู้รับยังคงมีการติดต่อกันอยู่ แต่โหนดกลางที่ใช้เป็นผู้ช่วยรับ – ส่งข้อมูลเกิดเสียหายใช้การไม่ได้ หรือสายสื่อสารบางช่วงถูกตัดขาด โปรโตคอลนี้จะต้องสามารถจัดการหาทางเลือกอื่นเพื่อทำให้การสื่อสารดำเนินต่อไปได้โดยอัตโนมัติ ข้อที่สองคือ จะต้องมีความอ่อนตัวต่อการสื่อสารข้อมูลได้หลายชนิด ทั้งแบบที่ไม่มีความเร่งด่วน และแบบที่ต้องการรับประกันความเร่งด่วนของข้อมูล

2.2.1 รูปแบบระบบเครือข่าย TCP/IP

TCP/IP (Transmission Control Protocol / Internet Protocol) เป็นชุดของโปรโตคอลที่ถูกใช้ในการสื่อสารผ่านเครือข่ายอินเทอร์เน็ต ได้รับการพัฒนามาตั้งแต่ปี ค.ศ. 1960 ซึ่งถูกใช้เป็นครั้งแรกในเครือข่าย ARPANET ซึ่งต่อมาได้ขยายการเชื่อมต่อไปทั่วโลกเป็นเครือข่ายอินเตอร์เน็ต ทำให้ TCP/IP เป็นที่ยอมรับอย่างกว้างขวางจนถึงปัจจุบัน

2.2.2 การแบ่งชั้นของ TCP/IP

TCP/IP เป็นชุดโปรโตคอลที่ประกอบด้วยโปรโตคอลย่อยหลายตัว โดยแต่ละตัวก็จะทำหน้าที่ในแต่ละเลเยอร์ ซึ่งรับผิดชอบและแปลความหมายของข้อมูลในแต่ละระดับของการสื่อสารซึ่งในภาพร่วมแล้ว TCP/IP แบ่งออกเป็น 4 เลเยอร์ ดังรูปที่ 2.6



รูปที่ 2.6 การแบ่งชั้นของ TCP/IP

ในแต่ละเลเยอร์จะมีหน้าที่ดังนี้

2.2.1.1 ชั้นสื่อสารเชื่อมต่อข้อมูล (Link Layer)

เลเยอร์นี้มีหน้าที่ควบคุมการรับส่งข้อมูลในระดับฮาร์ดแวร์ของเครือข่าย รับผิดชอบการรับส่งข้อมูลในระดับกายภาพ จนถึงการแปลความจากสัญญาณไฟฟ้าเป็นข้อมูลทางคอมพิวเตอร์

2.2.1.2 ชั้นสื่อสารควบคุมเครือข่าย (Network Layer)

ทำหน้าที่รับข้อมูลจากชั้น Transport Layer และค้นหาและเลือกเส้นทาง ระหว่างผู้รับและผู้ส่ง เทียบได้กับ Network Layer ของ OSI Model โปรโตคอลในเลเยอร์นี้ได้แก่ IP, ICMP, IGMP

2.2.1.3 ชั้นจัดการนำส่งข้อมูล (Transport Layer)

รับผิดชอบการรับส่งข้อมูลระหว่างปลายด้านส่งและด้านรับข้อมูล และส่งข้อมูลขึ้นไปให้ Application Layer นำไปใช้งาน ต่อ เทียบได้กับ Session Layer และ Transport Layer ของ OSI Model

2.2.1.4 ชั้นสื่อสารการประยุกต์ (Application Layer)

เป็นเลเยอร์ที่แอพพลิเคชั่นเรียกโปรโตคอลระดับล่าง ๆ ลงไป เพื่อให้บริการต่าง ๆ เช่น FTP, SMTP , Telnet , HTTP , POP

2.2.3 โครงสร้างของโปรโตคอล TCP/IP

เนื่องจาก TCP/IP เป็นชุดของโปรโตคอลประกอบด้วยโปรโตคอลหลายตัว ทำงานร่วมกันในเลเยอร์ต่างๆ และมีหน้าที่แตกต่างกันออกไป ได้แก่

- **TCP : (Transmission Control Protocol)**

อยู่ใน Transport Layer ทำหน้าที่จัดการและควบคุมการรับส่งข้อมูล และมีกลไกควบคุมการรับส่งข้อมูลให้มีความถูกต้อง (reliable) และมีการสื่อสารอย่างเป็นกระบวนการ (connection - orient)

- **UDP : (User Datagram Protocol)**

อยู่ใน Transport Layer ทำหน้าที่จัดการและควบคุมการรับส่งข้อมูล แต่ไม่มีกลไกควบคุมการรับส่งข้อมูลให้มีเสถียรภาพและเชื่อถือได้ (unreliable, connectionless) โดยปล่อยให้เป็นหน้าที่ของแอพพลิเคชั่นเลเยอร์ แต่ UDP มีข้อได้เปรียบในการส่งข้อมูลได้ทั้งแบบ unicast, multicast และ broadcast อีกทั้งยังทำการติดต่อสื่อสารได้เร็วกว่า TCP เนื่องจาก TCP ต้องเสีย overhead ให้กับขั้นตอนการสื่อสารที่ทำให้ TCP มีความน่าเชื่อถือในการรับส่งข้อมูลนั่นเอง

- **IP : (Internet Protocol)**

อยู่ใน Internetwork Layer เป็นโปรโตคอลหลักในการสื่อสารข้อมูล มีหน้าที่ค้นหาเส้นทางระหว่างผู้รับและผู้ส่ง โดยใช้ IP Address ซึ่งมีลักษณะเป็นเลขสี่ชุด แต่ละชุดมีค่าตั้งแต่ 0 - 255 เช่น 172.17.3.12 ในการอ้างอิงโฮสต์ต่าง ๆ และกลไกการ Route เพื่อส่งต่อข้อมูลไปจนถึงจุดหมายปลายทาง

- **ICMP : (Internet Control Message Protocol)**

อยู่ใน Internetwork Layer มีหน้าที่ส่งข่าวสารและแจ้งข้อผิดพลาดให้แก่ IP

- **IGMP : (Internet Group Management Protocol)**

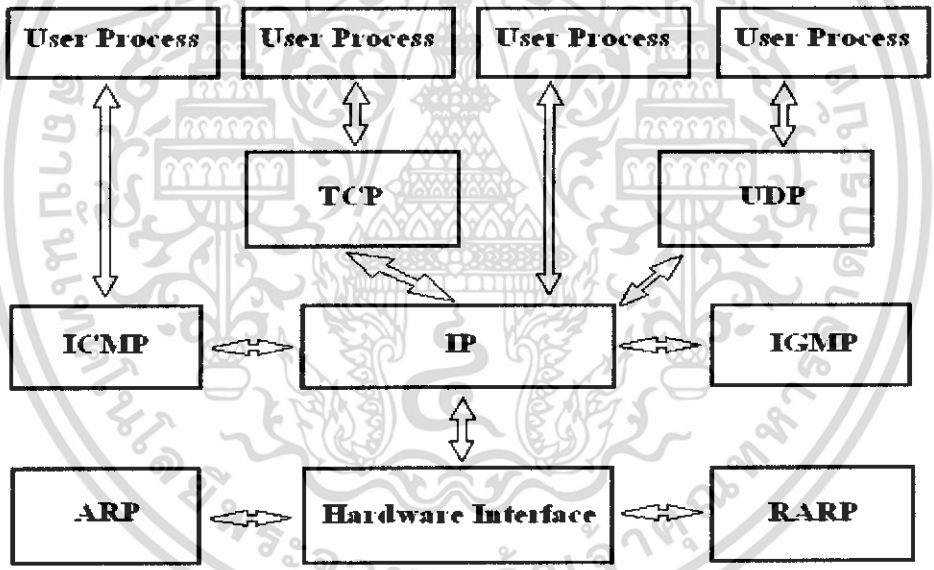
อยู่ใน Network Layer ทำหน้าที่ในการส่ง UDP ดาต้าแกรมไปยัง กลุ่มของโฮสต์ หรือ โฮสต์หลาย ๆ ตัวพร้อมกัน

- **ARP : (Address Resolution Protocol)**

อยู่ใน Link Layer ทำหน้าที่เปลี่ยนระหว่าง IP แอดเดรส ให้เป็นแอดเดรสของ Network Interface เรียกว่า MAC Address ในการติดต่อระหว่างกัน MAC Address คือหมายเลขประจำของ Hardware Interface ซึ่งในโลกนี้จะไม่มี MAC Address ที่ซ้ำกัน มีลักษณะเป็นเลขฐาน 16 ยาว 6 ไบต์ เช่น 23:43:45:AF:3D:78 โดย 3 ไบต์แรกจะเป็นรหัสของผู้ผลิต และ 3 ไบต์หลังจะเป็นรหัสของผลิตภัณฑ์

- **RARP : (Reverse Address Resolution Protocol)**

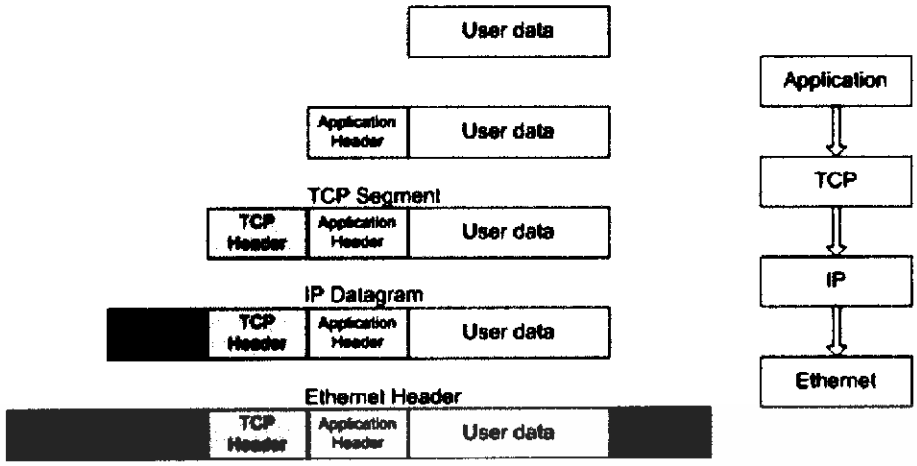
อยู่ในลิงค์เลเยอร์เช่นกัน แต่ทำหน้าที่กลับกันกับ ARP คือ เปลี่ยนระหว่างแอดเดรสของ Network Interface ให้ เป็นแอดเดรสที่ใช้โดย IP Address



รูปที่ 2.7 แสดงให้เห็นถึงความสัมพันธ์ระหว่างโปรโตคอลต่าง ๆ ใน TCP/IP

2.2.4 Encapsulation

การ Encapsulation คือ การนำข้อมูลที่ต้องการส่งมาประกอบรวมกับข้อมูลที่เป็นส่วนควบคุมของโปรโตคอล โดยข้อมูลส่วนที่เป็นส่วนควบคุมนั้นจะอยู่ในส่วนหัวของข้อมูลเรียกว่า " เฮดเดอร์ " (Header) ภายใน Header จะบรรจุข้อมูลที่สำคัญของโปรโตคอลที่ทำการ Encapsulate คือ แอดเดรสต้นทาง แอดเดรสปลายทาง ความยาวข้อมูล รหัสตรวจสอบความผิดพลาดข้อมูล

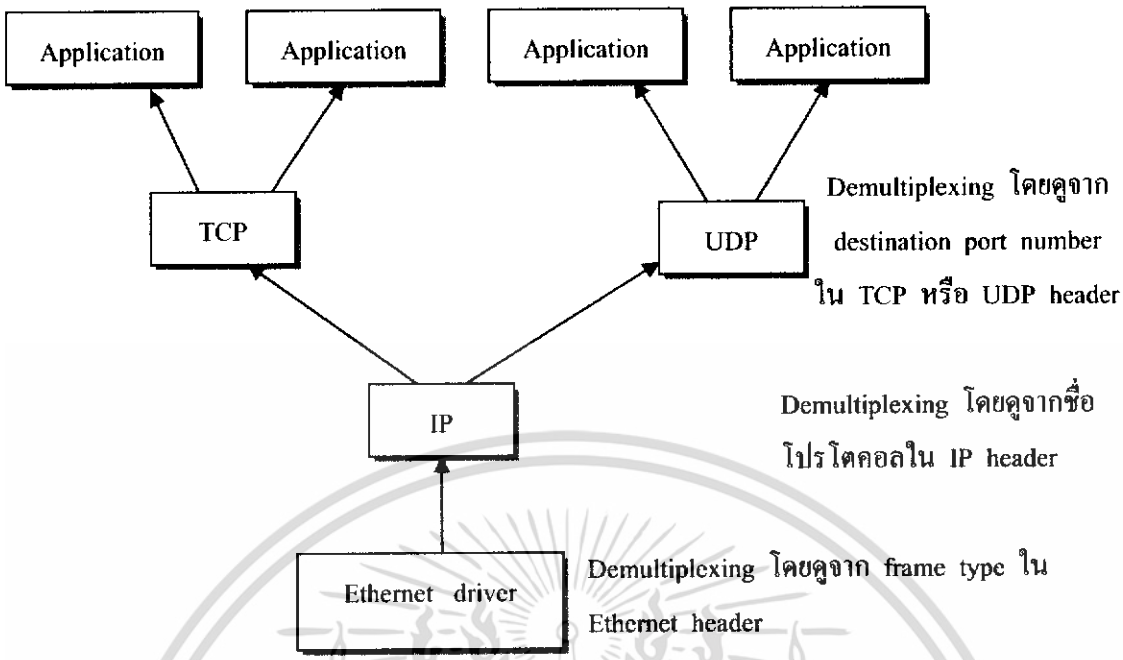


รูปที่ 2.8 ขั้นตอนการ encapsulation เมื่อข้อมูลถูกส่งผ่านโปรโตคอลต่างๆ

- ข้อมูลที่ผ่านการ Encapsulate ในแต่ละระดับมีชื่อเรียกแตกต่างกัน ข้อมูลที่มาจาก User หรือก็คือข้อมูลที่ User เป็นผู้ป้อนให้กับ Application เรียกว่า " User Data "
- เมื่อ Application ได้รับข้อมูลจาก User ก็จะนำมาประกอบกับส่วนหัวของ Application เรียกว่า " Application Data " และส่งต่อไปยังโปรโตคอล TCP
- เมื่อโปรโตคอล TCP ได้รับ Application Data ก็จะนำมารวมกับ Header ของ โปร-โตคอล TCP เรียกว่า " TCP Segment " และส่งต่อไปยังโปรโตคอล IP
- เมื่อโปรโตคอล IP ได้รับ TCP Segment ก็จะนำมารวมกับ Header ของ โปรโตคอล IP เรียกว่า " IP Datagram " และส่งต่อไปยังเลเยอร์ Datalink Layer
- ในระดับ Datalink จะนำ IP Datagram มาเพิ่มส่วน Error Correction และ flag เรียกว่า Ethernet Frame ก่อนจะแปลงข้อมูลเป็นสัญญาณไฟฟ้า ส่งผ่านสายสัญญาณที่เชื่อมโยงอยู่ต่อไป

2.2.5 Demultiplexing

เมื่อผู้รับ ได้รับข้อมูล ก็จะเกิดกระบวนการทำงานย้อนกลับคือ โปรโตคอลเดียวกัน ทางฝั่งผู้รับก็จะได้รับข้อมูลส่วนที่เป็น Header ก่อนและนำไปประมวลผล จึงจะทราบว่าข้อมูลที่ตามมามีลักษณะอย่างไร ซึ่งกระบวนการย้อนกลับนี้เรียกว่า Demultiplexing



รูปที่ 2.9 การ Demultiplexing ข้อมูล

การ Demultiplex และ Encapsulate เป็นสิ่งคู่กันและสอดคล้องกัน อุปกรณ์ที่จะสื่อสารกันบนเน็ตเวิร์กได้ จะต้องมีส่วนที่ทำหน้าที่ทั้งสอง โดยการ Demultiplexing ใช้ในตอนที่รับข้อมูลจากเน็ต-เวิร์ก และการ Encapsulate ใช้ตอนที่ทำการส่งข้อมูลในทุกเลเยอร์ของโปรโตคอล

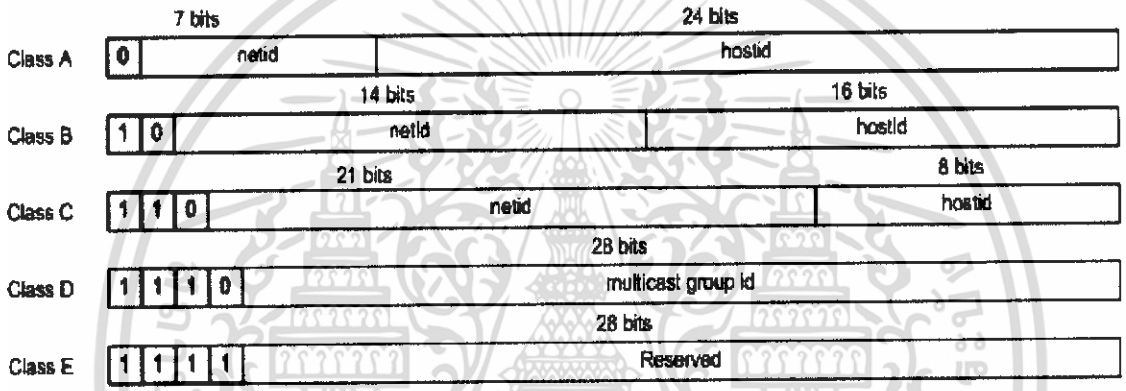
2.2.6 โปรโตคอล IP (Internet Protocol)

IP เป็นโปรโตคอลที่ทำหน้าที่รับภาระในการนำข้อมูลไปส่งยังผู้รับ ที่เชื่อมต่ออยู่ในระบบ network ซึ่งทั้งสองฝั่งอาจอยู่คนละเน็ตเวิร์คกันก็ได้ โปรโตคอลอื่น ๆ ในระดับ Network Layer ขึ้นไป ทั้ง TCP , UDP , ICMP ต่างก็ต้องอาศัยโปรโตคอล IP ในการรับส่งข้อมูลทั้งสิ้น

โปรโตคอล IP มีความสามารถในการค้นหาเส้นทางจากผู้รับไปยังผู้ส่ง มีกลไกที่ชาญฉลาดในการค้นหาเส้นทาง สามารถค้นหาเส้นทางได้ไปถึงผู้รับได้เอง หากมีเส้นทางที่สามารถไปได้ แต่ไม่ได้ติดต่อระหว่างผู้รับกับผู้ส่งโดยตรง และไม่มีการยืนยันว่า ข้อมูลถึงผู้รับจริงหรือไม่ ทั้งนี้อาจเกิดจากหลายสาเหตุ เช่น ที่อยู่ของผู้รับไม่มีการเชื่อมต่ออยู่ในระบบ Internet กล่าวได้ว่า โปรโตคอล IP มีหน้าที่ในการค้นหาเส้นทางเท่านั้น ไม่มีการยืนยันผลสำเร็จในการส่งข้อมูล หากเกิดข้อผิดพลาดในการส่งข้อมูล แม้ว่าจะมีการส่ง ICMP massage กลับมารายงานข้อผิดพลาด แต่ก็รับประกันไม่ได้อยู่ดีว่า ICMP message จะกลับมาถึงเรียบร้อยหรือไม่ ด้วยเหตุนี้ จึงถือว่า IP เป็นโปรโตคอลที่ไม่มีความน่าเชื่อถือ (reliable)

2.2.6.1 IP Addressing

ทุกอินเตอร์เฟสที่ต่ออยู่บนอินเตอร์เน็ต จะต้องมีหมายเลขประจำตัวเพื่อใช้ในการสื่อสารข้อมูล เรียกว่า Internet Address หรือเรียกย่อ ๆ ว่า IP Address โดยค่า IP Address นี้จะเป็นหมายเลขจำนวน 32 บิต แต่แทนที่จะกำหนดให้เลขทั้ง 32 บิตนั้นถูกนับต่อเนื่องกันไป ก็จะใช้วิธีการแบ่งหมายเลขดังกล่าวออกเป็นกลุ่มของเลขขนาด 8 บิตจำนวน 4 ชุด และคั่นแต่ละชุดด้วยจุด ตัวอย่างเช่น 172.17.3.12 นอกจากนี้ใน IP Address นั้นยังถูกแบ่งออกเป็น 2 ส่วนคือ ส่วนที่เป็นแอดเดรสของเน็ตเวิร์ก (Network ID) และส่วนที่เป็นแอดเดรสของโฮสต์ (Host ID) ซึ่งข้อมูลในส่วนนี้จะถูกใช้สำหรับค้นหาเส้นทางของ IP ในการที่จะขนส่งข้อมูลจากต้นทางให้ถึงปลายทางอย่างถูกต้อง เพื่อเป็นการกำหนดขนาดของเน็ตเวิร์กสำหรับ IP Address ต่าง ๆ ดังนั้นจึงมีการจัด IP Address ในแต่ละช่วงออกเป็นคลาส (class) ต่าง ๆ กัน จาก A ถึง E เพื่อจะได้ทำการจัดสรร IP Address ได้อย่างเหมาะสมกับขนาดของเน็ตเวิร์ก



รูปที่ 2.10 การกำหนด IP Address ในคลาสต่าง ๆ

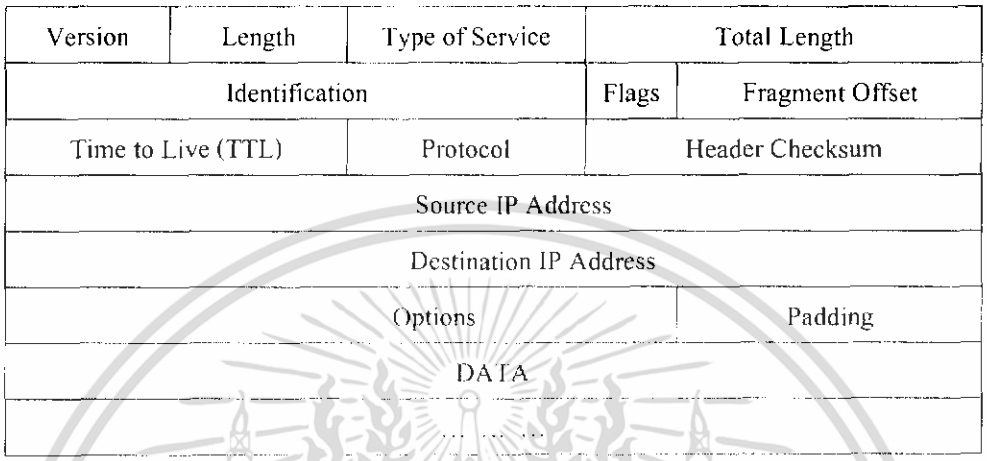
จากข้อกำหนดในการแบ่งคลาสของ IP Address หากลองนำบิตที่อยู่ในตอนต้นของ IP Address ในแต่ละคลาสมาแปลงเป็น IP Address ในเลขฐานสิบ จะเห็นว่าแต่ละคลาสครอบคลุม IP Address ช่วงต่าง ๆ ดังตารางที่ 2.1

Class	Range
A	0.0.0.0 - 127.255.255.255
B	128.0.0.0 - 191.255.255.255
C	192.0.0.0 - 223.255.255.255
D	224.0.0.0 - 239.255.255.255
E	240.0.0.0 - 255.255.255.255

ตารางที่ 2.1 แสดงช่วงของ IP Address ในแต่ละคลาส

2.2.6.2 IP Header

เมื่อข้อมูลถูกส่งลงมาจากชั้น Transport Layer สู่ชั้น Network Layer กระบวนการ Encapsulate ของ IP Protocol จะทำการเพิ่มส่วน Header ลงไป Header ของ IP datagram มีขนาด 20 - 32 ไบต์ มีส่วนประกอบต่าง ๆ ดังแสดงในรูปที่ 2.11



รูปที่ 2.11 IP Header

ตำแหน่ง	ชื่อ	อธิบาย
0 – 3	Version	มีขนาด 4 บิตเป็นเวอร์ชั่นของ IP ปัจจุบันค่านี้ถูกกำหนดให้เป็น 4
4 – 7	Length	มีขนาด 4 บิตเป็นค่าความยาวของ Header นี้ โดยปกติจะเป็น 5 หมายความว่า 5*32 บิต = 20 ไบต์
8 - 15	Type of Service	เป็นข้อมูลขนาด 8 บิต
16 – 31	Total Length	เป็นฟิลด์ที่บอกจำนวนไบต์ทั้งหมดของ IP Datagram ด้วยขนาด 16 บิตทำให้ Datagram มีขนาดสูงสุดไม่เกิน 65535 ไบต์ และมีขนาดเล็กสุดไม่ต่ำกว่า 512 ไบต์
32 – 47	Identification	ใช้ในกรณีที่มีการแบ่งดาต้าแกรมออกเป็นแฟรกเมนต์ เมื่อนำกลับมารวมกันใหม่จะได้รู้ว่ามาจากดาต้าแกรมเดียวกัน
48 – 50	Flags	ใช้ในกรณีที่มีการแบ่งข้อมูลออกเป็นแฟรกเมนต์ มีความหมายดังนี้ บิต 0 : reversed – เป็น 0 เสมอ บิต 1 (DF) – 0 = May Fragment, 1 = Don't Fragment บิต 2 (MF) – 0 =Last Fragment, 1 = More Fragments
51 – 63	Fragment offset	เป็นส่วนระบุข้อมูลที่ใช้แยกรวมข้อมูล เพื่อให้ข้อมูลที่ถูกแยกออกเป็นแฟรกเมนต์กลับมารวมกันได้อย่างถูกต้องตามลำดับ
64 – 71	Time to Live (TTL)	เป็นจำนวนครั้งสูงสุดที่ดาต้าแกรมนี้จะถูกส่งผ่านเครือข่ายไปยัง

<table>
<tr><td></td><td></td><td>ปลายทางได้ เพื่อ ป้องกันไม่ให้ดาต้าแกรมถูกเราต์ไปเรื่อยๆอย่างไม่สิ้นสุด ปกติค่านี้จะเริ่มต้นที่ 32 และจะถูกลดค่าลงทีละ 1 เมื่อมีการเราต์ จนค่านี้มีค่าเป็น 0 ก็จะไม่ถูกเราต์อีกต่อไป</td></tr>
<tr><td>72 – 79</td><td>Protocol</td><td>เป็นข้อมูลที่ระบุโปรโตคอลที่ส่งดาต้าแกรมนี้มา ตัวอย่างโปรโตคอลที่ใช้บ่อย ๆ ได้แก่
<table>
<tr><th>โปรโตคอล</th><th>ค่าในฟิลด์ Protocol</th><th>อธิบาย</th></tr>
<tr><td>ICMP</td><td>1</td><td>Internet Control Message Protocol</td></tr>
<tr><td>TCP</td><td>6</td><td>Transmission Control Protocol</td></tr>
<tr><td>UDP</td><td>17</td><td>User Datagram Protocol</td></tr>
</table></td></tr>
<tr><td>80 – 95</td><td>Header Checksum</td><td>เป็นส่วนตรวจสอบความถูกต้องของข้อมูลใน Header โดยไม่เกี่ยวกับส่วนข้อมูลที่อยู่ภายใน payload ค่านี้จะถูกคำนวณใหม่ทุกครั้งที่มีการเปลี่ยนแปลงข้อมูลใน Header (เช่น TTL ที่มีการเปลี่ยนแปลงทุกครั้งที่ IP data gram ถูกส่งผ่านเราเตอร์)</td></tr>
<tr><td>96 – 127</td><td>Source IP Address</td><td>คือ IP Address ของผู้ส่งดาต้าแกรม</td></tr>
<tr><td>128 – 163</td><td>Destination IP Address</td><td>คือ IP Address ของผู้รับดาต้าแกรม</td></tr>
<tr><td>ไม่แน่นอน</td><td>Options</td><td>มีขนาดข้อมูลไม่แน่นอน ใช้สำหรับกำหนดค่าพารามิเตอร์ปลีกย่อย ซึ่งส่วนใหญ่ไม่มีการนำไปใช้งาน</td></tr>
<tr><td>ขึ้นอยู่กับ Option</td><td>Padding</td><td>มีข้อมูลว่างเปล่า ใช้เป็นส่วนเติมเต็มของฟิลด์ Option ให้ครบ 32 ไบต์</td></tr>
</table>

ตารางที่ 2.2 อธิบายตำแหน่งต่าง ๆ ของ IP Header

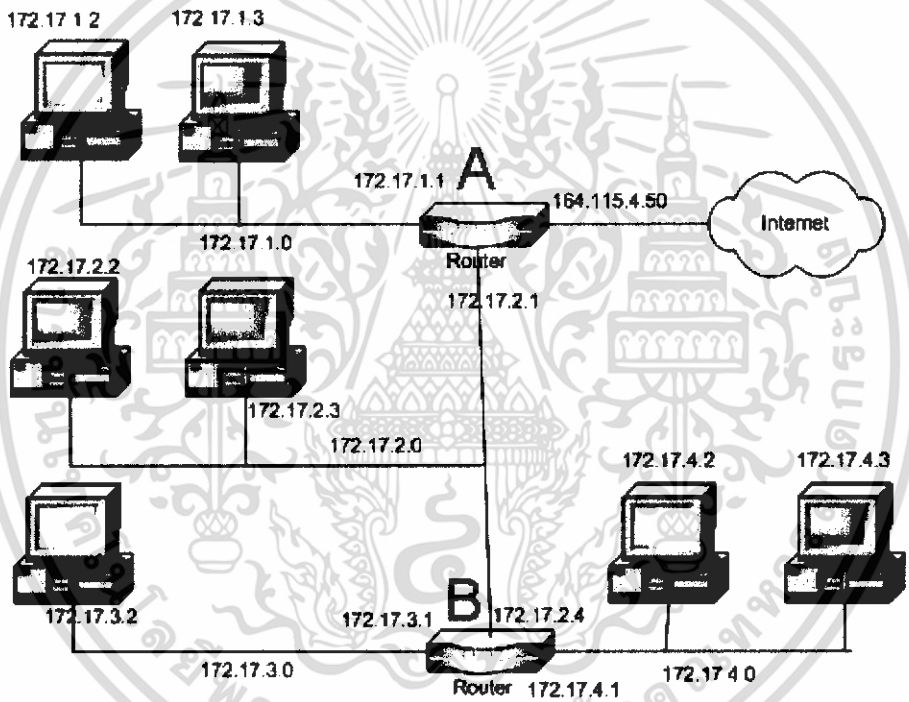
2.2.6.3 IP Routing

IP Routing เป็นกระบวนการค้นหาเส้นทางในการส่งผ่านข้อมูลจากต้นทางไปยังปลายทางโดยผ่านการส่งต่อข้อมูลไปจนกว่าจะถึงปลายทาง นับเป็นกลไกสำคัญที่ทำให้ IP เป็นโปรโตคอลที่สามารถส่งข้อมูลจากโฮสต์หนึ่งไปอีกโฮสต์หนึ่งได้แม้ว่าจะอยู่ไกลแสนไกล

- **Host** โฮสต์เป็นอุปกรณ์ที่ทำหน้าที่ให้กำเนิดข้อมูลในกรณีเป็นผู้ส่ง หรือทำหน้าที่รับข้อมูลไปใช้งานในกรณีเป็นผู้รับ การสื่อสารข้อมูลใด ๆ จะต้องเป็นการสื่อสารจากโฮสต์ไปยังโฮสต์เสมอ สำหรับ IP Packet แล้วข้อมูลในเฮดเดอร์ที่ปรากฏอยู่ในฟิลด์ Source Address และ Destination Address ซึ่งเรียกว่า IP Address จะเป็นหมายเลขระบุตำแหน่งของโฮสต์ต้นทางและโฮสต์ ปลายทางเท่านั้น

- **Network** เน็ตเวิร์คเป็นเครือข่ายที่มีการเชื่อมต่อกันของโฮสต์ 2 ตัวขึ้นไป โฮสต์แต่ละตัวในเน็ตเวิร์คเดียวกันสามารถเชื่อมต่อถึงกันได้โดยตรง
- **Router** เราเตอร์ทำหน้าที่ในการส่งผ่านข้อมูลจากเน็ตเวิร์กหนึ่งไปยังอีกเน็ตเวิร์กหนึ่ง ตำแหน่งของเราเตอร์จะอยู่ในจุดที่เชื่อมต่อระหว่างสองเน็ตเวิร์กเข้าด้วยกัน ด้วยข้อกำหนดของ IP ข้อมูลจะส่งไปถึงกันโดยตรงข้ามเน็ตเวิร์กไม่ได้ จะต้องอาศัยเราเตอร์เป็นผู้ทำหน้าที่ส่งผ่านข้อมูลไปให้ ใน Router จะมี Routing Table สำหรับเก็บข้อมูล เพื่อใช้ในการพิจารณาเลือกเส้นทางในการส่งดาต้าแกรม

ในการอธิบายกระบวนการ Routing ให้เป็นที่เข้าใจในเบื้องต้น ขออธิบายจากเน็ตเวิร์คตัวอย่างที่แสดงในรูปที่ 2.12



รูปที่ 2.12 ตัวอย่างโครงข่าย

การ Routing จะเป็นไปตามขั้นตอนดังนี้

1. ถ้าโฮสต์ต้นทางและปลายทางต่อเชื่อมร่วมอยู่ในเน็ตเวิร์กเดียวกัน มีการเชื่อมต่อถึงกันโดยตรง เช่น อีเธอร์เน็ตหรือโทเค็นริง ดังแสดงในภาพที่ 2.12 เป็นการติดต่อระหว่าง 172.17.2.2 และ 172.17.2.3 IP ดาต้าแกรมก็จะถูกส่งไปยังโฮสต์ปลายทางโดยตรง

2. หากโฮสต์ต้นทางและปลายทางไม่ได้อยู่ในเน็ตเวิร์กเดียวกัน IP ดาต้าแกรมจะถูกส่งไปยัง Default Router

3. เมื่อเราเตอร์ได้รับ IP Datagram จากข้อ 2 แล้วตรวจสอบดู หากพบว่าโฮสต์ปลายทางต่อร่วมอยู่บนเน็ตเวิร์กเดียวกันกับเราเตอร์ ให้ทำการส่งดาต้าแกรมไปที่โฮสต์นั้น เช่น หาก 172.17.3.2 ต้องการส่งดาต้าแกรมไปยัง 172.17.4.2 จะต้องส่งดาต้าแกรมไปที่ Router B, Router B จะส่งดาต้าแกรมต่อไปยังโฮสต์ปลายทาง

4. หากไม่ได้ต่อร่วมกันก็ส่งดาต้าแกรมไปที่เราเตอร์ตัวต่อไป โดย Router จะเป็นผู้เลือกเส้นทาง ซึ่งมีอยู่ 2 กรณีคือ

- ถ้ามีข้อมูลของโฮสต์ปลายทางอยู่ใน Routing Table Router จะส่งดาต้าแกรมไปยัง router ตัวที่ระบุไว้ใน routing table

- ถ้าไม่มีข้อมูลของโฮสต์ปลายทางอยู่ใน Routing Table Router จะส่งดาต้าแกรมไปยัง default router และกลับไปที่ขั้นตอนในข้อ 3 ใหม่ จนกว่า IP Datagram จะเดินทางถึงปลายทาง หรือหมดเวลาในการส่ง (TTL = 0)

สมมติว่าเครื่อง 172.17.1.3 ต้องการติดต่อกับ 172.17.4.3 จะต้องส่งIP datagram ไปยัง Router A หาก Router A มีข้อมูลเกี่ยวกับ 172.17.4.3 อยู่ ก็จะรู้ว่าต้องส่งดาต้าแกรมไปยัง Router B คือ 172.17.2.4 และ Router B ก็จะส่ง IP datagram ไปยังโฮสต์ปลายทางได้สำเร็จ

2.2.6.4 Subnet Addressing / Subnet Mask

ในการใช้งาน โปรโตคอล TCP/IP ใน Internet นั้นการแบ่ง IP Address ออกเป็นแอดเดรสของเน็ตเวิร์ก (Net ID) และแอดเดรสของ โฮสต์ ตามที่ระบุของแต่ละคลาสค่อนข้างจะขาดประสิทธิภาพ คือในเน็ตเวิร์กคลาส A และ B แต่ละเน็ตเวิร์กนั้น สามารถมีจำนวนโฮสต์ได้มาก ซึ่งการที่จะนำ IP Address มาใช้อย่างทั่วถึงนั้นมีโอกาสเป็นไปได้ยากมากทั้งคลาส A และคลาส B เพราะมีโอกาสน้อยมากที่จะมีเน็ตเวิร์ก ใดในโลกมีจำนวนโฮสต์มากมายขนาดนั้นอยู่ภายในเน็ตเวิร์กเดียว ดังนั้น IP Address ที่จัดสรรให้ไปในแต่ละเน็ตเวิร์กของ คลาสเหล่านี้จึงถูกใช้ไม่หมดและไม่สามารถนำไปใช้ประโยชน์ที่อื่นได้เลย ดังนั้นเพื่อให้การจัดสรร IP เป็นไปอย่างมีประสิทธิภาพ จึงมีการนำส่วนของ Host ID มาแบ่งย่อยเป็นสองส่วนคือ Subnet ID และ Host ID ทำให้ได้เน็ตเวิร์คย่อยหลาย ๆ เน็ตเวิร์ก โดยในแต่ละเน็ตเวิร์ก มีจำนวนโฮสต์ไม่มากเกินไปและเพียงพอต่อการใช้งาน

2.2.7 โปรโตคอล TCP (Transmission control Protocol)

TCP เป็นโปรโตคอลที่ใช้สื่อสารระหว่างโฮสต์ที่มีความน่าเชื่อถือ จะเห็นได้ว่าโปรโตคอลในระดับ IP หรือแม้กระทั่ง UDP จะสนใจข้อมูลเพียง 1 Datagram กลไกของโปรโตคอลจะมีหน้าที่ตรวจสอบความถูกต้องเพียงเฉพาะ Datagram นั้น ๆ เมื่อจะทำ การส่งดาต้าแกรมใหม่ก็จะถือว่าเป็นข้อมูลชุดใหม่ที่ไม่มีความสัมพันธ์ใด ๆ กับข้อมูล Datagram อื่น (การสื่อสาร 1 ครั้ง จึงใช้เพียง 1 Data gram) แต่

สำหรับ TCP แล้วจะเห็นว่าข้อมูลนั้นเป็น stream คือ มีความสัมพันธ์ต่อเนื่องกัน มีกลไกในการตรวจสอบทั้งด้านส่งและด้านรับ เพื่อให้แน่ใจว่าสามารถสื่อสารกันได้จริงจึงจะมีการส่งรับข้อมูลเกิดขึ้น ตลอดจนการยกเลิกการติดต่อก็มีกลไกสำหรับแจ้งให้อีกฝั่งทราบ ทำให้การสื่อสารด้วย TCP จึงเสมือนว่าทั้ง 2 ฝ่าย คือ ฝ่ายรับและฝ่ายส่งได้ทำการต่อสาย เน็ตเวิร์กถึงกัน (connected) ตลอดเวลาที่การรับส่งข้อมูลจนกระทั่งการสื่อสารทั้งหมดเสร็จสิ้นจึงจะทำการยกเลิกการเชื่อมต่อนั้นเสีย

จุดเด่นประการสำคัญของ TCP ที่กล่าวถึงอยู่เสมอคือ ความมีเสถียรภาพและความถูกต้องของการสื่อสารซึ่งมีความเชื่อถือได้สูง คุณสมบัติที่ทำให้ TCP มีข้อดีดังกล่าวคือ

1. ข้อมูลที่จะส่งผ่าน TCP จะถูกนำมาแตกย่อยออกเป็นส่วน ๆ ให้มีขนาดเหมาะสมสำหรับการส่งข้อมูล โดย TCP มีกลไกในการพิจารณาว่าขนาดเท่าใดจะทำให้การรับ - ส่งนั้นมีประสิทธิภาพและน่าเชื่อถือสูงสุด โดยข้อมูลแต่ละชุดที่แบ่งออกและทำการส่งโดย TCP แต่ละครั้งจะเรียกว่า TCP เซกเมนต์

2. ในการส่งข้อมูลแต่ละครั้ง TCP จะมีการจับเวลาไว้เสมอ เพื่อรอการตอบรับจากผู้รับว่าได้รับข้อมูลถูกต้อง หากหมดเวลาแล้วไม่มีการตอบรับ TCP จะถือว่าข้อมูลไปไม่ถึงและทำการแก้ปัญหาที่เกิดขึ้น เช่น ยกเลิกการติดต่อ, ส่งข้อมูลซ้ำ ทำให้ Application ทราบสถานะการส่งข้อมูลตลอดเวลา

3. TCP มี checksum ซึ่งจะครอบคลุมทั้ง TCP Header และ TCP Data เพื่อเป็นการป้องกันและตรวจสอบว่าข้อมูลที่ส่งมานั้นถูกต้อง และไม่ได้ถูกแก้ไขระหว่างทาง หาก TCP ได้รับข้อมูลที่ทำการตรวจสอบกับ checksum แล้วปรากฏว่า มีความผิดพลาดเกิดขึ้น TCP จะทิ้งข้อมูลที่ได้รับและจะไม่ทำการตอบรับข้อมูลนั้นกลับไปยังผู้ส่ง คือถือเสมือนว่าไม่ได้รับข้อมูลนั้น เพื่อให้ทางฝ่ายผู้ส่งทำการส่งใหม่หรือหาข้อบกพร่องและพยายามแก้ไขตามแต่แอพพลิเคชั่นทางฝ่ายผู้ส่งเห็น สมควร

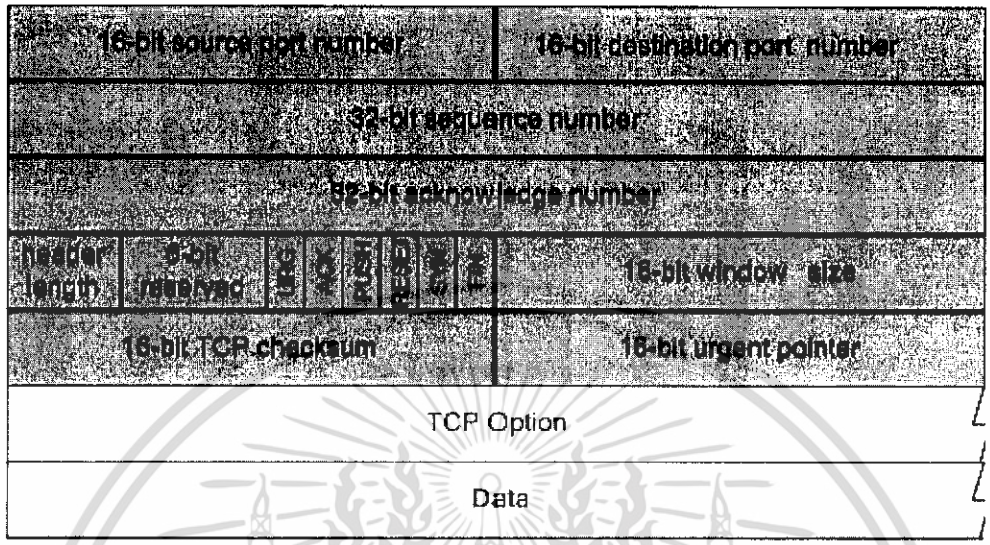
4. เนื่องจาก TCP อาศัย IP ในการส่งข้อมูล ซึ่ง IP เองอาจจะถูกแฟรกเมนต์ได้ และทำให้ข้อมูลที่ถูกแฟรกเมนต์นั้นส่งถึงปลายทางในลำดับที่ไม่ถูกต้องได้ หน้าที่ของ TCP เมื่อรับข้อมูลที่แฟรกเมนต์มานั้นจะต้องนำข้อมูลแต่ละส่วนมาประกอบ รวมกันให้ถูกต้องสมบูรณ์ก่อนจะส่งไปยัง Application Layer ต่อไป

5. การส่ง - รับข้อมูลด้วย IP อาจจะมีกรณีที่ IP Datagram นั้นถูกส่งซ้ำขึ้นได้ TCP ที่รับข้อมูลซ้ำดังกล่าวจะต้องทราบว่า เป็น IP Datagram ที่ซ้ำและไม่นำข้อมูลไปใช้งาน

6. TCP มีกลไกควบคุมการไหลของข้อมูล (Flow Control) โดยการควบคุมนี้จะต้องอาศัยลำดับของการรับส่งที่ถูกต้อง และสัมพันธ์กันทั้ง 2 ฝั่ง ในขณะเดียวกันข้อมูลที่ส่งนั้นจะต้องอาศัย IP หลาย Datagram จึงจะได้รับข้อมูลครบทั้งหมด ดังนั้น ในการรับข้อมูลทางฝ่ายรับจึงต้องเตรียมบัฟเฟอร์ไว้จำนวนหนึ่งเพื่อรอรับข้อมูลและรวบรวมข้อมูลทั้งชุดให้อยู่ในบัฟเฟอร์ ก่อนที่จะทำการจัดเรียงข้อมูลตรวจสอบความถูกต้องแล้วจึงส่งต่อไปยังแอพพลิเคชั่น ด้วยเหตุผลดังกล่าวจะเห็นได้ว่าขนาดของข้อมูลมิได้ถูกจำกัดที่ขนาดของ Datagram ใด ๆ ข้อมูลที่ส่งอาจจะมีขนาดใหญ่มากอยู่ในหลาย Datagram ก็เป็นได้ ดังนั้น เพื่อป้องกันการส่งข้อมูลขนาดใหญ่เร็วเกินไปจนทำให้ทางฝ่ายรับ ไม่มีหน่วยความจำ

เพียงพอที่จะเป็นบัฟเฟอร์ที่พักข้อมูล การส่งข้อมูลจึงถูกจำกัดโดยจะอนุญาตให้ทำการส่งข้อมูลได้เท่าที่ฝ่ายรับมีบัฟเฟอร์เพียงพอเท่านั้น

2.2.7.1 TCP Header



รูปที่ 2.13 TCP Header

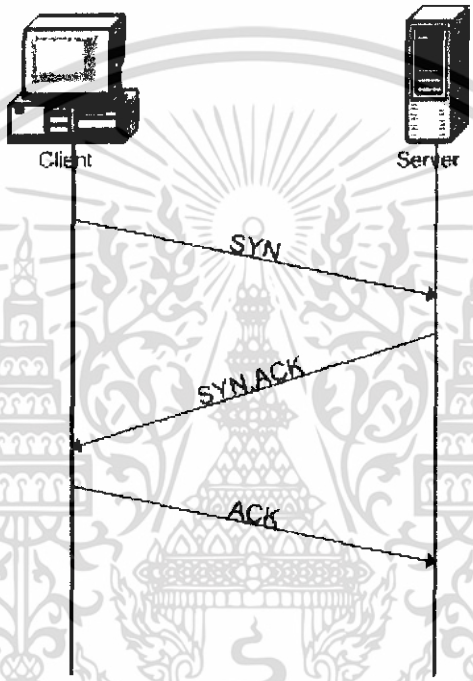
ชื่อ	อธิบาย
Source Port Number	หมายถึง พอร์ตที่โฮสต์ต้นทางใช้ในการสื่อสารกันของเซสชั่นนี้ และ TCP/IP จะใช้พอร์ตนั้นไป ตลอดตราบใดที่การสื่อสารในเซสชั่นนี้ยังไม่ยุติลง โดยทั่วไปพอร์ตนี้จะเรียกว่า "ไคลเอนต์พอร์ต" คือพอร์ตที่ไคลเอนต์เปิดขึ้นมาเพื่อรอการตอบรับจากเซิร์ฟเวอร์ (พิจารณาจากทิศทางของแพ็กเก็ตที่ส่งมาจากไคลเอนต์ไปยังเซิร์ฟเวอร์) ไคลเอนต์พอร์ตจะมีหมายเลขไม่แน่นอนและเปลี่ยนไปทุกครั้งที่มีการเริ่มการเชื่อมต่อใหม่ เป็นพอร์ตที่ถูกเปิดไว้ในระยะเวลาสั้น ๆ ค่าที่เป็นไปได้ของพอร์ตนี้ขึ้นอยู่กับการจัดสรรของระบบปฏิบัติการ ในการกำหนดขอบเขตของพอร์ตเหล่านี้ส่วนใหญ่จะมีค่า อยู่ในช่วง 1024 - 5000
Destination Port Number	หมายถึง หมายเลขพอร์ตบนโฮสต์ปลายทางที่โฮสต์ต้นทางต้องการติดต่อด้วย โดยนัยแล้วจะหมายถึงแอพพลิเคชั่นที่ให้บริการอยู่พอร์ตนั้นที่โฮสต์ปลายทางนั่นเอง พอร์ตนั้นจะเรียกอีกอย่างหนึ่งว่า "เซิร์ฟเวอร์พอร์ต" หมายเลขพอร์ตที่เปิดไว้จะขึ้นอยู่กับแอพพลิเคชั่นที่ให้บริการ โดยทั่วไปแอพพลิเคชั่นแต่ละประเภทจะมีหมายเลขพอร์ต เป็นมาตรฐานสำหรับให้ไคลเอนต์ได้เรียกใช้บริการ
Sequence Number	เป็นฟิลด์ที่ระบุถึงหมายเลขลำดับที่ใช้อ้างอิงในการสื่อสารข้อมูลแต่ละครั้ง เพื่อให้ทั้ง 2 ฝ่าย จะได้รับทราบตรงกันว่าเป็นข้อมูลของชุดใด การนำไปใช้งานจะได้ไม่ปะปนกัน และมีลำดับที่ถูกต้อง เนื่องจากการสื่อสารข้อมูลผ่าน TCP นั้น

<table>
<tr><td></td><td>จังหวะและลำดับเป็นส่วนสำคัญของโปรโตคอลไม่ยิ่งหย่อนไปกว่าข้อมูลใน TCP Header รวมไปถึง การที่ข้อมูลในแต่ละ TCP Segment อาจจะถูกทำการแฟรกเมนต์ในเลเยอร์ของ IP ถัดลงไป ทำให้ข้อมูลถูกแบ่งออกและส่งไปในลำดับที่ไม่เรียงกัน หากไม่มีจุดอ้างอิงของข้อมูลก็จะไม่สามารถอ่านข้อมูลกลับใหม่ได้อย่างสมบูรณ์และถูกต้อง การส่งข้อมูลและการตอบรับจะใช้ฟิลด์นี้เป็นตัวยืนยันระหว่างกันเสมอ</td></tr>
<tr><td>Acknowledge Number</td><td>ทำหน้าที่เช่นเดียวกับ Sequence Number ต่างกันตรงที่ Sequence Number ใช้ในการตอบรับกล่าวคือ Sequence Number ที่ใช้ในการอ้างอิงนั้นผู้ที่เริ่มส่งข้อมูลจะเป็นผู้กำหนดเลขขึ้นมาและส่งไปพร้อมกับการสร้างการเชื่อมต่อครั้งใหม่ แต่สำหรับฝ่ายที่ถูกติดต่อก็จำเป็นต้องกำหนดหมายเลขสำหรับใช้อ้างอิง ในการตอบรับเช่นกัน ค่าที่อยู่ใน Acknowledge Number ก็คือหมายเลขที่ใช้อ้างอิงในการตอบรับนี้</td></tr>
<tr><td>Header Length</td><td>โดยปกติความยาวของ TCP Header จะเท่ากับ 20 ไบต์ แต่ถ้าหากมีการใช้ Option อาจจะทำให้ ขนาดของเฮดเดอร์ยาวขึ้นตามข้อมูลที่ต้องเพิ่มมาจาก Option นั้น แต่ทั้งหมดแล้วจะไม่เกิน 60 ไบต์</td></tr>
<tr><td>TCP segment</td><td>เป็นข้อมูลในระดับบิตที่ใช้เป็นตัวบอกคุณสมบัติของ TCP Segment ที่กำลังส่งอยู่นั้น และใช้เป็นตัวควบคุมจังหวะ การรับส่งข้อมูลด้วย ซึ่ง Flag ทั้งหมดมีอยู่ 6 บิต แต่ละบิตมีชื่อและมีความหมายดังนี้
<table>
<tr><th>Flag</th><th>อธิบาย</th></tr>
<tr><td>URG</td><td>ใช้บอกความหมายว่าเป็นข้อมูลด่วน และมีข้อมูลพิเศษมาด้วย (อยู่ใน Urgent pointer)</td></tr>
<tr><td>ACK</td><td>แสดงว่าข้อมูลในฟิลด์ Acknowledge Number นำมาใช้งานได้</td></tr>
<tr><td>PUSH</td><td>เพื่อแจ้งให้ผู้รับข้อมูลทราบว่า ควรจะส่งข้อมูล Segment นี้ไปยังโพรเซสที่กำลังรออยู่ทันที</td></tr>
<tr><td>RESET</td><td>ใช้ในกรณีที่เกิดการสับสนขึ้นด้วยเหตุผลต่างๆ เช่น โฮสต์มีปัญหา ให้เริ่มต้นสื่อสารกันใหม่</td></tr>
<tr><td>SYN</td><td>ใช้ในการเริ่มต้นขอติดต่อกับปลายทาง</td></tr>
<tr><td>FIN</td><td>ใช้ส่งเพื่อแจ้งให้ปลายทางทราบว่ายุติการติดต่อ</td></tr>
</table></td></tr>
<tr><td>Window Size</td><td>เป็นขนาดของการรับ - ส่งข้อมูลในแต่ละครั้งที่ทางฝ่ายผู้รับจะสามารถรับได้ เนื่องจากในการรับข้อมูลนั้น ทางผู้รับจะต้องจัดเตรียมหน่วยความจำในการพักข้อมูลที่มากจาก TCP และทำการ Demultiplex ออกมา หากไม่มีการตกลง ถึงขนาดที่ทางฝ่ายรับสามารถรับได้ ก็จะทำให้การสื่อสารข้อมูลไม่สมดุล และฝ่ายรับจะประมวลผลไม่สมบูรณ์ ซึ่งจะส่งผลให้ต้องส่ง ข้อมูลซ้ำหลายครั้ง</td></tr>
</table>

Checksum	ฟิลด์ที่ใช้ในการตรวจสอบความถูกต้องของข้อมูลใน TCP เซกเมนต์ ใช้ระบุหมายเลข Sequence Number ของ TCP เซกเมนต์ล่าสุดที่อยู่ในโหมด Urgent
Urgentn Pointer	ข้อมูลเพิ่มเติมซึ่งจะอยู่ใน TCP Header เมื่อมีการตั้งค่า option บางอย่างที่ต้องการข้อมูลเพิ่มเติมซึ่งไม่มีใน TCP Header เช่น MSS, Strict Route

ตารางที่ 2.3 รายละเอียด TCP Header

2.2.7.2 Connection Establishment



รูปที่ 2.14 3 - way handshake

ก่อนที่จะเริ่มต้นการสื่อสาร จะต้องมีการส่งสัญญาณเพื่อบอกโฮสต์อีกฝั่งหนึ่งให้เตรียมตัวติดต่อ ซึ่งกระบวนการที่ใช้มีชื่อเรียกว่า 3 Way Handshake มีขั้นตอนดังนี้

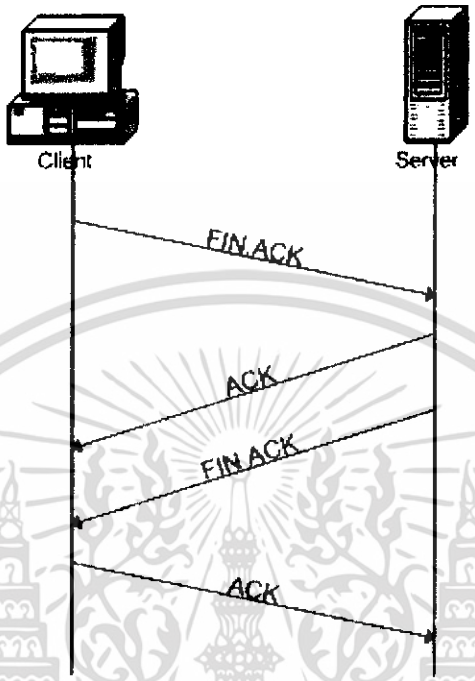
1. เครื่องไคลเอนต์จะทำการส่งเซกเมนต์ โดยเปิด SYN Flag ระบุหมายเลขพอร์ตที่ต้องการติดต่อบนเซิร์ฟเวอร์และระบุหมายเลข ลำดับของข้อมูล (ISN - Initial Sequence Number)

2. เครื่องเซิร์ฟเวอร์เมื่อได้รับข้อมูลเซกเมนต์จากข้อ 1 ก็จะตอบกลับด้วยการเพิ่มค่า ISN ที่ได้รับขึ้นอีก 1 พร้อมทั้งระบุหมายเลขลำดับ (ISN) ของตนเอง และเปิด SYN กับ ACK Flag

3. ไคล์เอนต์เมื่อได้รับการตอบกลับจากเซิร์ฟเวอร์ตามข้อ 2 ก็จะทำการตอบรับกลับไป โดยการเพิ่มค่า ISN ของเซิร์ฟเวอร์ขึ้นอีก 1 และเปิด ACK Flag เมื่อผ่านการสร้าง connection ทั้ง 3 ขั้นตอนแล้ว ตอนนี้

ทั้งไคล์เอนต์ และเซิร์ฟเวอร์เปรียบเสมือนมีการเชื่อมต่อถึงกันแล้ว สถานะของการเชื่อมต่อในขณะนี้เรียกว่า "Established "

2.2.7.3 Connection Termination



รูปที่ 2.15 Connection Termination

เมื่อการสื่อสารของทั้งสองฝั่งจบลง และไม่ต้องการรับส่งข้อมูลอีกต่อไป จะต้องทำตามขั้นตอนการยุติการสื่อสารเพื่อให้การสื่อสารจบลงอย่างสมบูรณ์ ซึ่งมีอยู่ 4 ขั้นตอนคือ

1. ไคลเอนต์ทำการส่ง ISN พร้อมกับ FIN ACK Flag ไปยังเซิร์ฟเวอร์
2. เซิร์ฟเวอร์ทำการตอบรับ ISN และบวกค่า ISN อีก 1 พร้อม ACK Flag
3. เซิร์ฟเวอร์ทำการส่ง ISN พร้อมกับ FIN ACK Flag ไปยังไคลเอนต์
4. ไคลเอนต์ทำการตอบรับ ISN และบวกค่า ISN อีก 1 พร้อม ACK Flag

การยุติการเชื่อมต่อ โดยส่ง FIN ACK ออกไปมีความหมายคือ โฮสต์ที่ส่งไม่มีข้อมูลจะส่งไปอีก มิใช่ต้องการปิดการสื่อสารทั้งหมดในทันที ดังนั้นจึงต้องทำทั้งสองทางการสื่อสารจึงจะยุติลงอย่างสมบูรณ์ ในการใช้งานจริง อาจมีการยุติการสื่อสารเพียงด้านเดียว คือหยุดส่งข้อมูล แต่ยังคงเปิดพอร์ตไว้รอรับข้อมูลจากอีกด้านหนึ่ง ทั้งนี้ขึ้นอยู่กับลักษณะการใช้งาน การปิดพอร์ตสื่อสารเพียงด้านเดียวเช่นนี้ เรียกว่า " Half - Close "

2.2.8 โปรโตคอล HTTP (Hyper Text Transfer Protocol)

โปรโตคอล HTTP เป็นโปรโตคอลหลักที่ใช้แลกเปลี่ยนข้อมูลกันระหว่างเซิร์ฟเวอร์และไคลเอนต์ของ WWW โดยถูกออกแบบมาให้มีความกะทัดรัด สามารถทำงานได้รวดเร็ว มีกระบวนการที่ไม่ซับซ้อนและมีคำสั่งที่ใช้งานไม่มากนัก แต่สามารถรองรับข้อมูลได้ทุกแบบ ไม่ว่าจะเป็นข้อมูลทั่วไปที่เข้ารหัสแบบ MIME หรือเป็นข้อมูลกราฟิก เช่น ไฟล์ที่เป็น GIF หรือ JPEG เป็นต้น

HTTP จะแบ่งการทำงานออกเป็น 2 ด้านคือ ด้านเว็บเซิร์ฟเวอร์และด้านไคลเอนต์ โดยไคลเอนต์จะติดต่อเข้ามายังเซิร์ฟเวอร์โดยใช้โปรแกรมบราวเซอร์และอ้างถึงแอดแดรสของเซิร์ฟเวอร์โดยใช้รูปแบบของ URL ส่วนด้านเซิร์ฟเวอร์จะส่งข้อมูลกลับมาในรูปแบบภาษา HTML (Hyper Text Markup Language) โดยโปรโตคอล HTTP ใช้วิธีการเข้ารหัสแบบ MIME เป็นมาตรฐานการทำงาน

2.2.8.1 โครงสร้างข้อมูลของ HTTP

โครงสร้างข้อมูลของ HTTP จะแบ่งออกเป็น 2 ส่วนใหญ่ ๆ คือ ส่วนเฮดเดอร์หรือเรียกว่า Metadata จะเป็นส่วนเก็บข้อมูลที่จำเป็นต้องใช้ภายในโปรโตคอล ส่วนที่สองเป็นส่วนข้อมูลจริงที่ต้องการรับส่ง ทั้งนี้ HTTP ถูกออกแบบมาให้สามารถรับส่งข้อมูลผ่าน Proxy หรือ Firewall ต่าง ๆ ได้ โดยการทำงาน HTTP จะอาศัยโปรโตคอลพื้นฐาน TCP/IP ซึ่งทั่วไปจะใช้หมายเลขพอร์ตที่ 80

โปรโตคอล HTTP ในปัจจุบันได้พัฒนาขึ้นมาเป็นเวอร์ชั่น 1.1 (จากเดิมคือ เวอร์ชั่น 1.0) ซึ่งโปรแกรมบราวเซอร์ที่แพร่หลายทั่วไปนั้นจะสามารถรองรับโปรโตคอลในเวอร์ชั่นใหม่นี้ได้ โดยใน HTTP เวอร์ชั่น 1.1 นี้ได้เพิ่มประสิทธิภาพทำงานให้สูงขึ้น และปรับปรุงในด้านต่าง ๆ ที่ทำให้มีความสามารถมากขึ้น ดังนี้

- ลดภาระของการเชื่อมต่อผ่านโปรโตคอล TCP และสามารถใช้ประสิทธิภาพของ TCP ได้อย่างเต็มที่
- สามารถทำการบีบอัดข้อมูลที่รับส่งระหว่างเซิร์ฟเวอร์และไคลเอนต์ได้
- รองรับการทำงานแบบ Virtual Host หมายถึง เว็บเซิร์ฟเวอร์เครื่องหนึ่ง ๆ มีชื่อโดเมนมากกว่าหนึ่งชื่อได้
- สามารถรองรังการทำงานได้หลายภาษา
- โอนไฟล์ข้อมูลเฉพาะบางส่วนได้จะมีประโยชน์มาก ซึ่งคุณสมบัตินี้จะมีประโยชน์มากในกรณีที่มีการโอนไฟล์ข้อมูลขนาดใหญ่ และเกิดปัญหาขึ้นระหว่างการทำงาน ซึ่งโปรโตคอล HTTP 1.1 มีจุดเด่นที่สามารถตรวจสอบได้ และโอนไฟล์ข้อมูลต่อจากส่วนที่เคยโอนมาแล้วได้

2.2.8.2 คำสั่งของโปรโตคอล HTTP

HTTP มีคำสั่งต่าง ๆ ไม่มากนัก โดยมีคำสั่งที่ใช้งานแพร่หลายอยู่เพียง 3 คำสั่ง คือ GET, HEAD และ POST ส่วนคำสั่งอื่น ๆ อีก 4 คำสั่งคือ PUT, DELETE, LINK และ UNLINK มีให้ใช้งานเช่นกัน แต่ไม่เป็นที่นิยมมากนัก รายละเอียดคำสั่งของ HTTP มีดังนี้

คำสั่ง	รายละเอียด
GET	ใช้อ่านข้อมูลจากเว็บเซิร์ฟเวอร์และส่งไปยังไคลเอนต์โดยมีรูปแบบดังนี้ GET <URL> HTTP/1.0 ตัวอย่างเช่น ต้องการให้เว็บเซิร์ฟเวอร์ส่งไฟล์ sale.html จากโดเมน www.netcorp.com ไปยังไคลเอนต์จะใช้รูปแบบของคำสั่ง GET ดังนี้ GET www.netcorp.com/sale.html /1.0 นอกจากนี้ คำสั่ง GET ยังสามารถกำหนดเงื่อนไขให้อ่านข้อมูลจากเว็บเซิร์ฟเวอร์เฉพาะที่มีการเปลี่ยนแปลงแก้ไขได้ด้วย
HEAD	คำสั่งนี้จะทำงานคล้ายกับคำสั่ง GET แต่เว็บเซิร์ฟเวอร์จะส่งข้อมูลกลับมาให้เฉพาะในรายละเอียดของ Metadata หรือข้อมูลในเฮดเดอร์เท่านั้น ส่วนข้อมูลที่เป็น HTML จะไม่ถูกส่งมาด้วย ซึ่งคำสั่ง HEAD นี้จะใช้เพื่อทดสอบว่าข้อมูลตาม URL นั้น ๆ มีการเปลี่ยนแปลงหรือไม่เท่านั้น
POST	เป็นคำสั่งที่ตรงข้ามกับคำสั่ง GET และ HEAD โดยทำหน้าที่ส่งข้อมูลจากไคลเอนต์ไปยังเว็บเซิร์ฟเวอร์นั้นจะไม่ค่อยมีใช้งาน นอกจากในกรณีที่ HTML ทำงานในลักษณะที่ให้ผู้ใช้กรอกข้อมูลตามแบบฟอร์ม (เช่น รายละเอียดส่วนตัวของผู้ใช้งาน) และส่งข้อมูลนี้กลับมาเก็บที่เว็บเซิร์ฟเวอร์
PUT	เป็นคำสั่งที่ทำงานเหมือนกับคำสั่ง POST แต่ไม่เป็นที่นิยม
DELETE	เพื่อให้ไคลเอนต์สั่งเว็บเซิร์ฟเวอร์ลบ URL ที่กำหนดไว้ออกจากเว็บเซิร์ฟเวอร์ แต่เป็นคำสั่งที่ไม่นิยมใช้มากนัก เนื่องจากเว็บเซิร์ฟเวอร์ทั่วไปมักจะทำงานในแบบอ่านข้อมูลได้เท่านั้น (Read Only)
LINK	เป็นคำสั่งที่เชื่อม URL ที่ต้องการไปยังเว็บเซิร์ฟเวอร์อื่น
UNLINK	ยกเลิกคำสั่ง LINK ให้กลับมาใช้เว็บเซิร์ฟเวอร์เดิมตามที่กำหนดไว้ใน URL

ตารางที่ 2.4 รายละเอียดคำสั่งของ HTTP

2.2.8.3 สถานะการทำงานของ HTTP

โปรโตคอล HTTP ได้กำหนดรหัสแสดงสถานะการทำงานของโปรโตคอลไว้ โดยแบ่งกลุ่มของรหัสสถานะออกไว้เป็น 5 กลุ่ม คือ

รหัสสถานะ	ประเภท	รายละเอียด
100 – 199	Information	เป็นรหัสสถานะกลุ่มที่เปิดให้โปรแกรมประยุกต์ต่าง ๆ กำหนดใช้งานได้เอง
200 – 299	Successful	กลุ่มรหัสที่แสดงว่าการทำงานสำเร็จ

รหัสสถานะ	ประเภท	รายละเอียด
300 – 399	Redirection	กลุ่มรหัสนี้จะใช้ภายในโปรโตคอล HTTP เอง โดยเป็นการทำงานที่ต่อเนื่องมาจากโปรเซสก่อนหน้านี้ ซึ่งไคลเอนต์เป็นผู้สั่งงาน
400 - 499	Client Error	ใช้แสดงปัญหาที่เกิดขึ้นกับไคลเอนต์
500 - 599	Server Error	ใช้แสดงปัญหาที่เกิดขึ้นกับเซิร์ฟเวอร์

ตารางที่ 2.5 รหัสแสดงสถานะการทำงานของโปรโตคอล HTTP

2.2.8.4 HTML (Hypertext Markup Language)

HTML เป็นสิ่งที่ผู้พัฒนาโฮมเพจคุ้นเคยกันดี และเป็นภาษาที่ออกแบบมาเพื่อให้โปรแกรมบราวเซอร์สามารถเข้าใจและทำงานได้ในแบบของไฮเปอร์เท็กซ์ ซึ่งผู้สร้างเว็บเพจจะใช้ภาษา HTML นี้ในการสร้างเว็บเพจและเก็บไว้ในเว็บเซิร์ฟเวอร์ เมื่อมีผู้ใช้งานติดต่อผ่านโปรแกรมบราวเซอร์ที่เครื่องไคลเอนต์ โดยระบุ URL ของเว็บเซิร์ฟเวอร์นั้น ๆ ไฟล์ HTML ที่เก็บไว้ในเว็บเซิร์ฟเวอร์ก็จะถูกส่งไปยังไคลเอนต์โดยใช้โปรโตคอล HTTP และแสดงผลให้ผู้ใช้งานเห็นโดยผ่านโปรแกรมบราวเซอร์ HTML มีพื้นฐานมาจากภาษาที่เรียกว่า SGML (Standard Generalized Markup Language) และ HTML ได้รับการออกแบบมาให้ใช้งานกับเว็บเพจที่ไม่มีการเปลี่ยนแปลงหรือเคลื่อนไหวในเพจนั้น ๆ ซึ่งในกรณีที่ต้องการพัฒนาให้เว็บเพจสามารถเปลี่ยนแปลงหรือเคลื่อนไหว (Dynamic html) ได้นั้น จะต้องใช้ภาษาอื่นเข้ามาร่วมด้วย เช่น Visual Basic หรือภาษาที่เป็นสคริปต์ต่าง ๆ เช่น Vbscript เป็นต้น

- แท็ก (Tag)

ความเข้าใจในเรื่องของ Tag เป็นส่วนสำคัญของการเรียนรู้ HTML โดย Tag จะทำหน้าที่กำหนดขอบเขตหรือแบ่งแยกการสั่งงานต่าง ๆ ในไฟล์ HTML ให้ตัวแปลภาษาสามารถเข้าใจได้ หน้าที่ Tag คือ ใช้ในการสร้างเฮดดิ้ง, กำหนดย่อหน้า, สร้างลิสต์, กำหนดรูปแบบ (Formatting) และการเชื่อมไปยังเว็บไซต์อื่น ๆ โดยการเริ่มต้น Tag จะใช้เป็นตัวอักษรอยู่ภายในสัญลักษณ์ "<" หรือ ">" และปิดท้ายด้วยตัวอักษรที่อยู่ระหว่างสัญลักษณ์ "</" และ ">" ตัวอย่างเช่น ถ้าต้องการให้แสดงคำว่า "Welcome" เป็นอักษรตัวหนา (boldface) ก็จะต้องกำหนด Tag เป็น <B>Welcome</B> เป็นต้น ซึ่ง Tag บางประเภทก็ไม่จำเป็นต้องปิดท้าย เช่น
 ซึ่งหมายถึง Line Break ตัวอย่างเช่น ให้แสดงว่า Good Morning และต่อท้ายด้วย Line Break จะทำได้โดย Good Morning
 เป็นต้น ส่วน Tag ที่เป็นคำอธิบายโปรแกรม (Comments) จะเริ่มต้นด้วย " <!- - " และปิดท้ายด้วย " - -> "

โครงสร้างโดยทั่วไปของไฟล์ HTML

```
<HTML>
<HEAD>
<TITLE>
```

<----------- รายละเอียดของ title ------------>
</TITLE>
<----------- Header อื่น ๆ ------------>
</HEAD>
<BODY>
<----------- ส่วนที่เป็นเนื้อหาของเอกสาร ------------>
</BODY>
</HTML>

ในบางครั้งคำสั่งใน HTML ก็ไม่สามารถครอบคลุมการทำงานได้เพียงพอ จึงจำเป็นต้องใช้ภาษาอื่น ๆ เข้าทำงานร่วมด้วย ซึ่งเรียกว่า Script Language ตัวอย่างเช่น JavaScript, VBScript หรือ Jscript เป็นต้น Tag ที่ใช้กำหนดสคริปต์นี้ก็คือ <SCRIPT LANGUAGE> ตัวอย่างเช่น <SCRIPT LANGUAGE = JAVASCRIPT> หมายถึง กำหนดเนื้อที่ในส่วนที่จะเป็นสคริปต์ของ JavaScript เป็นต้น

- Image Map

เป็นเว็บเพจต่าง ๆ บางครั้งจะสังเกตได้ว่าไม่ได้มีแต่ข้อความเท่านั้นที่สามารถคลิกต่อไปได้ รูปภาพบางรูปก็สามารถคลิกเมาส์เพื่อแสดงข้อความอื่น ๆ หรือลิงค์ต่อไปยังเพจอื่น ๆ ได้เช่นกัน นอกจากนั้นรูปภาพรูปเดียวกันยังสามารถแบ่งออกเป็นส่วนย่อย ๆ ที่สามารถคลิกเมาส์แล้วลิงค์ไปยังที่ต่างกันได้อีกด้วย ตัวอย่างเช่น ในเว็บเพจที่มีรูปแผนที่แสดงที่ตั้งของจังหวัดต่าง ๆ ในประเทศ เมื่อคลิกที่จังหวัดใดก็จะแสดงรายละเอียดของจังหวัดนั้น ๆ เป็นต้น ซึ่งความสามารถลักษณะนี้เรียกว่า "Image Map"

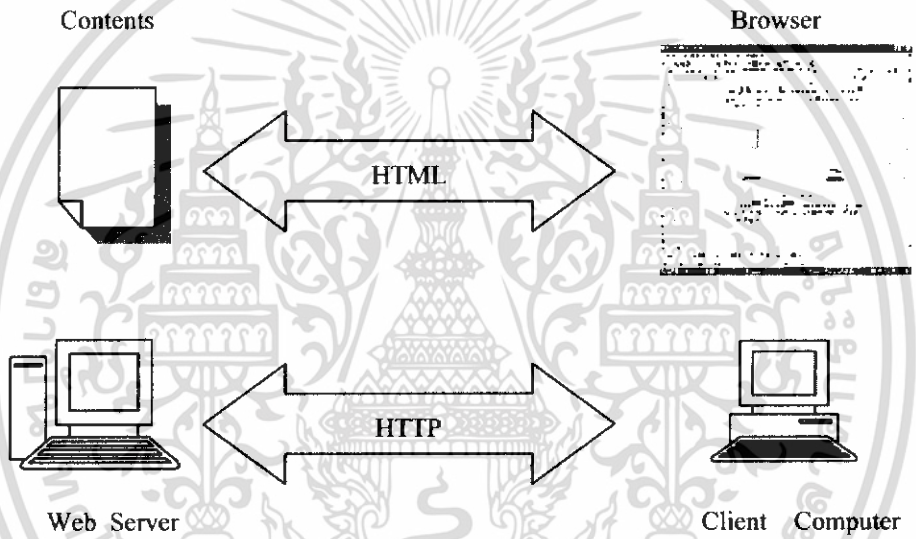
- CGI (Common Gateway Interface)

ในการใช้งาน HTTP ที่ต้องการติดต่อกับเซิร์ฟเวอร์อื่นที่ไม่ใช่ HTTP ด้วยกันเช่น การกรอกข้อมูลผ่านฟอร์ม และต้องการให้ไปค้นหาข้อมูลเซิร์ฟเวอร์อื่นที่เก็บฐานข้อมูลไว้ การทำงานในลักษณะนี้สามารถทำได้หลายวิธี แต่กลไกพื้นฐานที่ง่ายและสะดวกที่สุดคือ การทำงานโดยใช้ CGI หรือ Common Gateway Interface ซึ่งหลักการทำงานของ CGI นี้ ไคลเอนต์จะกำหนดไฟล์ CGI ที่ต้องการเรียกใช้งานโดยระบุในลักษณะของ URL เมื่อ HTTP หรือเซิร์ฟเวอร์ได้รับคำสั่งก็จะเรียกไฟล์ CGI นั้น ๆ ขึ้นมาทำงาน จากนั้น HTTP หรือเว็บเซิร์ฟเวอร์ก็จะส่งผลลัพธ์การทำงานกลับไปให้ไคลเอนต์ ตัวอย่างการเรียกใช้งาน CGI เช่น ต้องการเรียกใช้งาน CGI ชื่อ getcust.cgi อยู่ในไดเร็คทอรี่ cgi-bin ที่เว็บไซต์ www.netcorp.com จะอ้างถึงได้ดังนี้ http://www.netcorp.com/cgi-bin/getcust.cgi ในบางครั้งต้องมีที่ส่งค่าที่เป็นพารามิเตอร์เข้าไปใน CGI จะกำหนดได้ด้วยสัญลักษณ์ ? ถ้าหากจะเปรียบเทียบกับการเขียนด้วยภาษาซีก็คือ argc/argv ของฟังก์ชั่น main() ตัวอย่างเช่น http://www.metcorp.com/cgi-bin/getcust.cgi?a=1 ซึ่งภาษาที่จะใช้ในการเขียน CGI นั้นก็เป็นประเภทภาษาสคริปต์ต่าง ๆ เช่น Perl หรืออาจจะเขียนเป็นโปรแกรมด้วยภาษาระดับสูงต่าง ๆ ก็ได้ เช่น ภาษา C เป็นต้น โดยให้โปรแกรมไป

ทำงานตามที่กำหนดไว้ และสร้างไฟล์ HTML ขึ้นมา ส่วนในการติดต่อกับ HTTP หรือเว็บเซิร์ฟเวอร์ ของ CGI นั้นจะรับข้อมูลจากบราวเซอร์ที่ผ่านมาที่เว็บเซิร์ฟเวอร์ โดย CGI ต้องอ่านข้อมูลจากอินพุต มาตรฐาน เช่นในภาษาซีจะอ้างถึง stdin และในทำนองเดียวกัน การแสดงเอาต์พุตนั้น CGI จะเขียน ข้อมูลลงเอาต์พุตมาตรฐาน หรือ stdout ในภาษา C จากนั้นเอาต์พุตก็จะถูกส่งมาแสดงผลที่จอภาพของ บราวเซอร์โดยอัตโนมัติ

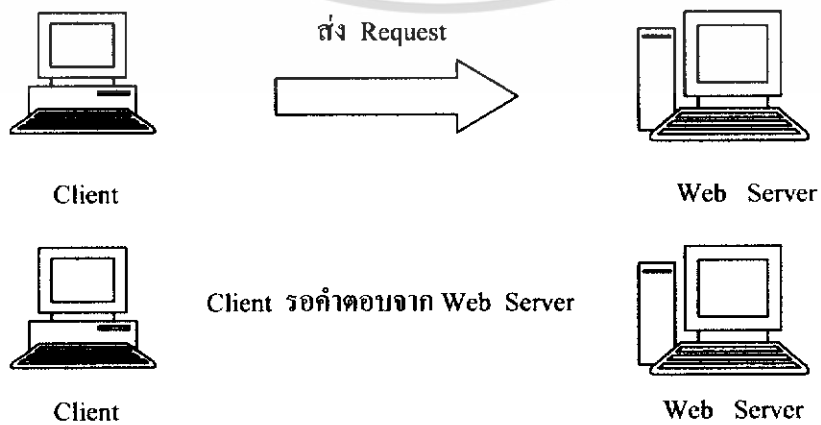
2.2.8.5 ความสัมพันธ์ระหว่าง HTTP กับ HTML.

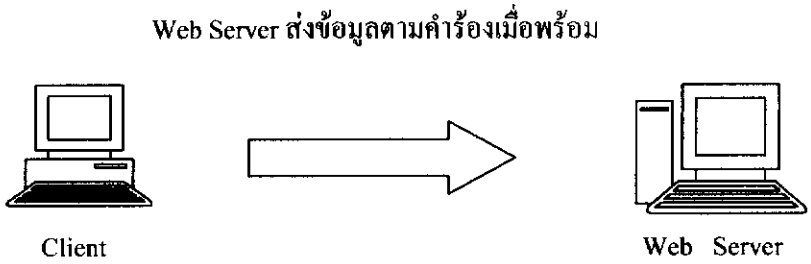
HTTP คือโปรโตคอลที่ใช้สื่อสารระหว่าง Client Computer กับ Server Computer ทำให้ทั้ง สองเครื่องรู้ว่าจะจัดการส่งข้อมูลไปอย่างไร ส่วน HTML คือสื่อภาษาที่ทำให้เอกสารหรือ Contents ที่ อยู่บนเครื่อง Server Computer เมื่อถูกส่งมาที่ Client Computer แล้วจะนำไปแสดงได้อย่างไร เรา เรียนซอฟท์แวร์ที่ใช้แสดงนี้ว่า Browser



รูปที่ 2.16 HTTP กับ HTML

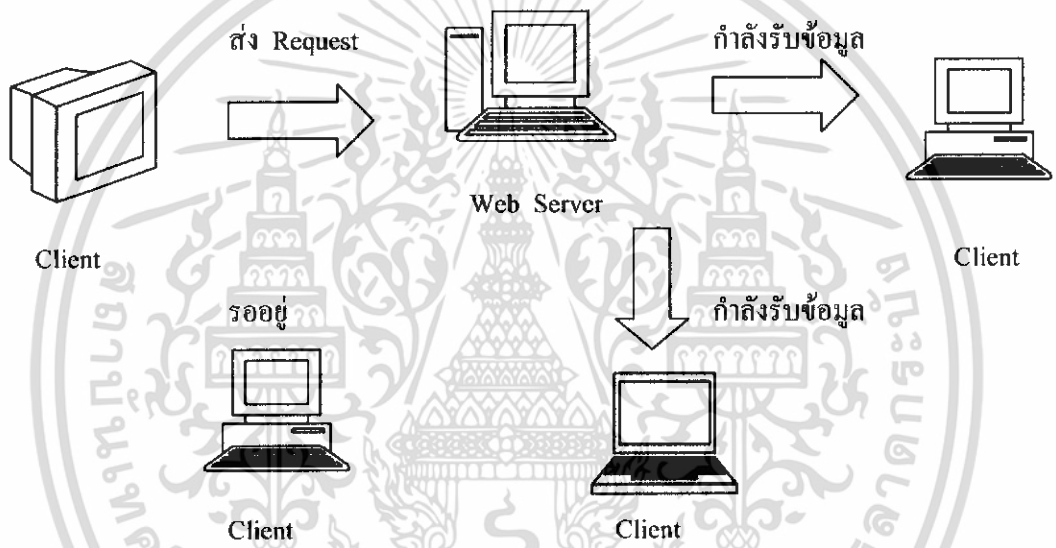
โปรโตคอล HTTP นี้จะวิ่งอยู่บน TCP/IP อีกชั้นหนึ่ง รูปแบบการทำงานจะไม่มีการจองสาย โดย client จะเรียกข้อมูลจาก server โดยการส่ง request ไปแล้วจะตัดการติดต่อทันที จากนั้นจะรอ จนกระทั่ง server ส่งข้อมูลมาได้





รูปที่ 2.17 รูปแบบการทำงานของ HTTP

ประโยชน์ของการทำงานแบบไม่จองสายของ HTTP ทำให้ WWW server สามารถให้บริการ Client ได้หลาย ๆ คนพร้อม ๆ กัน การสื่อสารของ WWW จึงมีประสิทธิภาพมากขึ้น

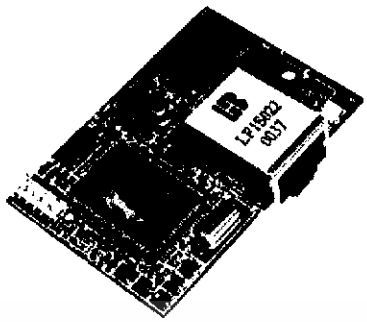


รูปที่ 2.18 การทำงานแบบไม่จองสายของ HTTP

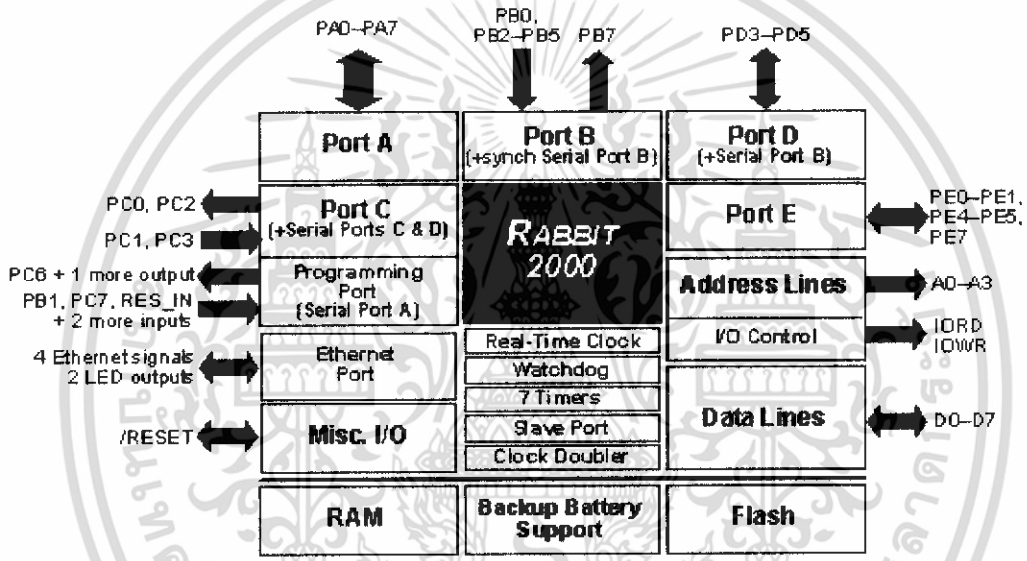
2.3 ทฤษฎีเกี่ยวกับ Rabbit

ลักษณะโครงสร้างของ Rabbit Core RCM2200 เป็นชิ้นส่วนไมโครโปรเซสเซอร์ที่ถูกออกแบบให้มีความสามารถที่หลากหลาย มี Port Ethernet สำหรับ LAN และ Internet ระบบวงจรที่ใช้ใน Rabbit Core RCM2200 ถูกออกแบบมาให้งานต่อการใช้งาน มีความเร็วอยู่ที่ 22.1 MHz หน่วยความจำที่ใช้เป็นแบบ Static Ram และ Flash Memory สัญญาณนาฬิกาที่ใช้งานในวงจรมี 2 แหล่ง (Oscillator และ Time-Keeping) ใช้แหล่งจ่ายไฟ +5V ในการทำงานของบอร์ดนอกจากนี้ยังสามารถติดต่อกับอุปกรณ์ภายนอก โดยผ่านทาง Port ที่จัดสรรไว้ผ่านคอนเน็คเตอร์ J4 กับ J5

2.3.1 ลักษณะที่สำคัญและความสามารถของ RCM2200



รูปที่ 2.19 บอร์ดวงจร RCM2200



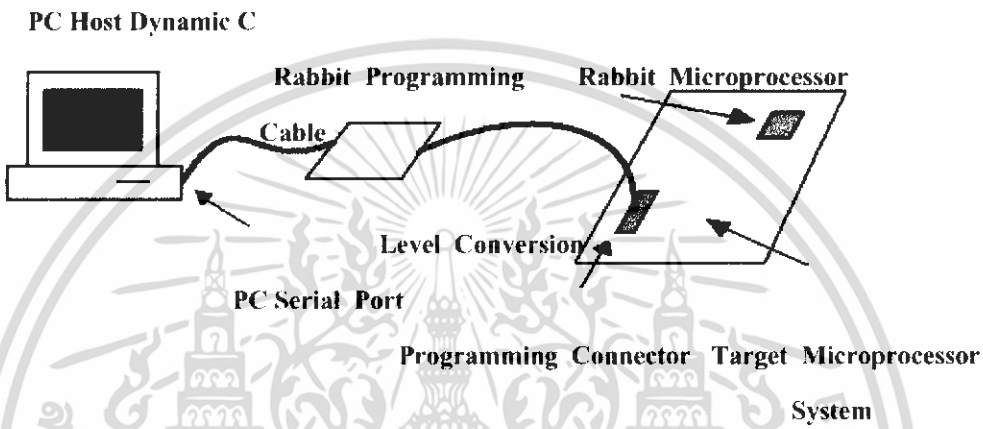
รูปที่ 2.20 ระบบภายใน RCM2200

- ขนาดเล็ก 1.60" x 2.30" x 0.86" หรือ (41 mm x 58 mm x 22 mm)
- ไมโครโปรเซสเซอร์ Rabbit 2000 ใช้ความถี่ในการทำงานที่ 22.1 MHz
- มีขาที่เป็น Input / Output ทั้งหมด 26 ขา เป็นแบบ Parallel มีโหมดการทำงาน หลากหลาย
- มี Data Bus 8 เส้น (D0 - D7)
- มี Address Bus 4 เส้น (A0 - A3)
- มี Flash Memory ขนาด 256 K และ Static Ram ขนาด 128K
- มีสัญญาณนาฬิกาให้จังหวะการทำงาน
- ใช้หัวต่อสัญญาณแบบ RJ – 45 เป็น Ethernet Port
- Port Serial เป็นชนิด CMOS ซึ่งเป็นอัตราบอร์ดแบบ Asynchronous ได้ความถี่สูงสุดถึง 345, 600 bps อัตราบอร์ดที่ใช้แบบ Synchronous ได้ความถี่สูงสุดถึง 138,240 bps

- Port Serial เป็นชนิด CMOS ซึ่งเป็นอัตราบอร์ดแบบ Asynchronous ได้ความถี่สูงสุดถึง 345, 600 bps อัตราบอร์ดที่ใช้แบบ Synchronous ได้ความถี่สูงสุดถึง 138,240 bps

2.3.2 การเชื่อมต่อ RCM2200

RCM2200 ออกแบบให้เชื่อมต่อกับ Motherboard เพื่อให้มีประสิทธิภาพในการใช้งาน โดยมีจุดเชื่อมต่อภายนอก คือ Headers 26 pin 2 Header ในบอร์ด RCM2200 จะประกอบด้วย 10Base-T Ethernet Port, 256k Flash Memory และ 128k Static RAM



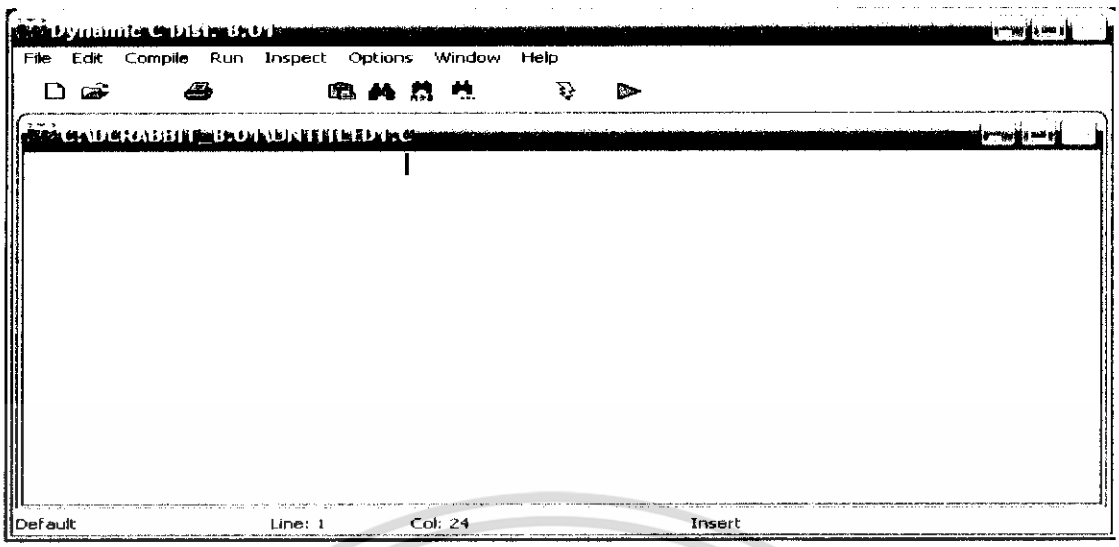
รูปที่ 2.21 การเชื่อมต่อ Board RCM2200 กับระบบคอมพิวเตอร์

2.3.3 พอร์ตอนุกรม (serial ports)

พอร์ตในการเชื่อมต่อ Rabbit Core RCM2200 บอร์ดไม่มีสาย RS - 232 หรือสาย RS - 485 โดยตรงบนบอร์ด การจะติดต่อด้วยสาย RS - 232 หรือ RS - 485 สามารถทำได้โดยการติดต่อผ่านบอร์ด Prototyping ซึ่งมีการสนับสนุนมาตรฐาน RS - 232 ไว้ในบอร์ดเดียวกัน

2.3.4 โปรแกรม Dynamic C

Dynamic C คือระบบการพัฒนารวมของการเขียนซอฟท์แวร์ สร้างมาจากระบบคอมพิวเตอร์ของ IBM ซึ่งออกแบบให้เข้ากับคอมพิวเตอร์ทั่วไปให้ใช้งานได้



รูปที่ 2.22 โปรกรม Dynamic C

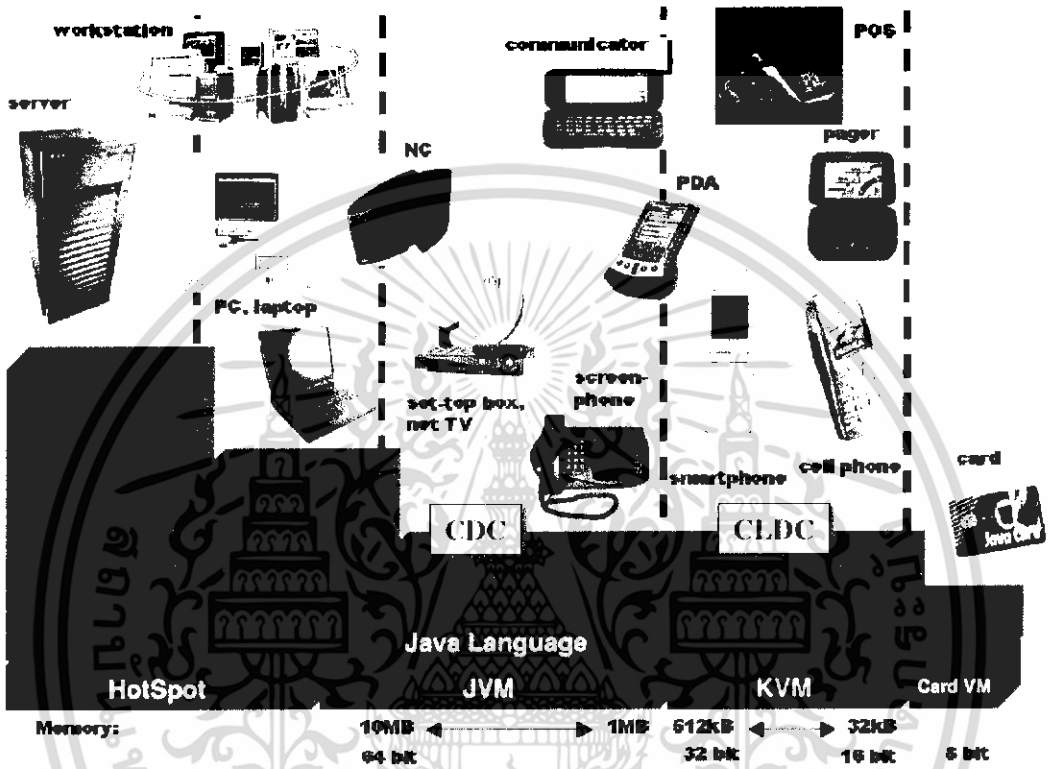
โปรแกรม Dynamic C นั้นมีการใช้ในเครือข่ายตั้งแต่ปี ค.ศ. 1989 ถูกออกแบบสำหรับการเขียนโปรแกรมที่ฝังระบบ และมีความสามารถตรวจสอบในตัวของมันเองอย่างรวดเร็ว สำหรับไมโครโปรเซสเซอร์ Rabbit 2000 ที่ใช้งานกันทั่วไป สามารถติดต่อกันโดยสายแพ 10 สาย ที่พอร์ต C โปรแกรมพื้นฐานของระบบ มีข้อมูลประมาณ 1,000 ไบท์ที่ใช้ในการจัดเตรียม Debugging และการติดต่อข้อมูลต่าง ๆ Dynamic C ต้องการ BIOS เพื่อใช้ในการตรวจสอบโปรแกรมเพื่อที่จะใช้งานได้สะดวกถ้าผู้ใช้หยุดการ Run โปรแกรมและใช้โปรแกรมใหม่ BIOS ก็จะเช็คการทำงานใหม่ตลอด Dynamic C ออกแบบให้เข้ากับภาษา Assembly หรือใช้ได้กับโปรแกรมภาษาซี การ Interrupt อาจจะเกิดขึ้นได้กับการเขียนในภาษา Dynamic C หรือภาษา Assembly ซึ่งซอฟต์แวร์ Dynamic C เป็นซอฟต์แวร์ที่มีประสิทธิภาพในการเขียนโปรแกรมเพื่อควบคุมการทำงานของไมโครโปรเซสเซอร์ Rabbit 2000 โดยซอฟต์แวร์มีโครงสร้างการใช้งานคล้ายคลึงกับภาษา C ลักษณะของตัวซอฟต์แวร์โปรแกรม Dynamic C จะสนับสนุน TCP/IP โดยจะมี Libraries หลักที่สนับเสนุนคือ DCRTCP.LIB, DNS.LIB, IP.LIB, TCP.LIB, UDP.LIB, NET.LIB ส่วนในการติดต่อชั้นเครือข่ายของโปรโตคอล TCP/IP จะมี Libraries ARP.LIB กับ ICMP.LIB

ผู้ใช้โปรแกรม Dynamic C มีตัวเลือกในการที่จะพัฒนาซอฟต์แวร์ภาษาเขียนใน flash memory ขนาด 256 k ไบต์ หรือใน Static Ram ขนาด 128 k ไบต์ ผลการทำงานในหน่วยความจำ คือการบันทึกข้อมูลสามารถบันทึกได้ถึง 100,000 ครั้งของการเขียน

ข้อเสียของการใช้ flash memory เมื่อมีการดีบักโปรแกรมเพื่อขัดจังหวะการทำงาน จะทำให้ Interrupt เกิดข้อผิดพลาด การทำงานของโปรแกรมก็จะหยุดตามไปด้วย

2.4 ทฤษฎีเกี่ยวกับโปรแกรม J2ME

J2ME (Java 2 Platform, Micro Edition) คือสิ่งที่ทำให้โทรศัพท์มือถือมีความสามารถมากขึ้น ใช้สำหรับการเขียนโปรแกรมบนอุปกรณ์ขนาดเล็กซึ่งมีทรัพยากร เช่น การแสดงผล ขนาดของหน่วยความจำ และความสามารถในการประมวลผลจำกัด โดยตัวอย่างของอุปกรณ์เหล่านี้ก็ได้แก่ โทรศัพท์มือถือและ PDA เป็นต้น โดยโครงสร้างของเทคโนโลยี Java เหล่านี้สามารถแสดงได้ดังรูป



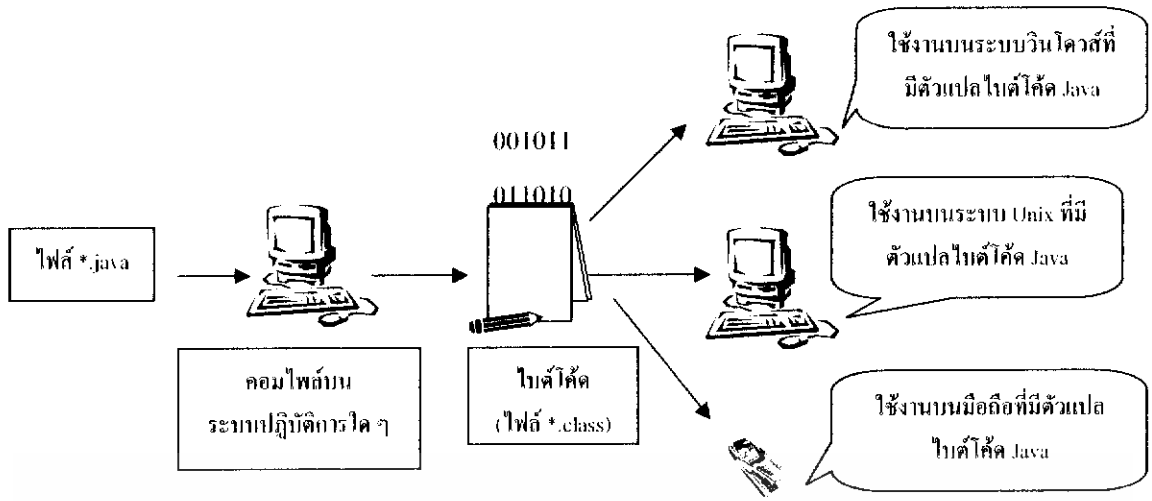
รูปที่ 2.23 แสดงแพล็ตฟอร์มของ Java 2

จากรูปจะเห็นได้ว่า Java 2 มีส่วนสำคัญที่เกี่ยวข้องกันอยู่ 3 ส่วนคือ

1. Virtual Machine
2. Configuration
3. Profile

2.4.1 Virtual Machine

โปรแกรม Java ทุกตัวจะต้องทำงานภายใต้ Java Virtual Machine (JVM) เสมอ เมื่อเรา คอมไพล์โปรแกรมเป็นไบต์โค้ด (ไฟล์ *.class) แล้ว Java Virtual Machine จะทำหน้าที่แปลงไบต์โค้ดเหล่านี้ไปเป็นภาษาเครื่องและทำงานตามคำสั่งนั้น ๆ ต่อไป ด้วยวิธีการนี้เอง โปรแกรม Java จึงสามารถทำงานได้บนทุกระบบปฏิบัติการ



รูปที่ 2.24 ลักษณะการทำงานของโปรแกรมภาษา Java

JVM จะเปลี่ยนไปตามระบบปฏิบัติการ (Operating System) ของอุปกรณ์แต่ละชนิด ซึ่งใน J2ME ได้ใช้ Configuration เป็นตัวกำหนด JVM ดังนี้

Configuration	JVM
CDC	CVM (Compact Virtual Machine)
CLDC	KVM (Kilobyte Virtual Machine)

ตาราง 2.6 แสดง Configuration ใน J2ME ที่เป็นตัวกำหนด JVM

2.4.2 Configuration

Configuration เป็นตัวระบุ Virtual Machine และคลาสไลบรารีพื้นฐานซึ่งจะมีเหมือนกันในอุปกรณ์ทุกตัวที่ถูกจัดอยู่ในกลุ่มเดียวกัน

โดย Configuration ใน J2ME ได้แบ่งกลุ่มของอุปกรณ์ออกเป็น 2 กลุ่ม โดยใช้คุณสมบัติของหน่วยความจำ การแสดงผล และความสามารถในการประมวลผลเป็นตัวกำหนด ดังนี้

- CDC (Connected Device Configuration)

คุณสมบัติของอุปกรณ์ในกลุ่ม ได้แก่

- มีหน่วยความจำตั้งแต่ 2 - 16 MB
- มีหน่วยประมวลผลขนาด 32 บิต เป็นอย่างน้อย
- ความเร็วในการเชื่อมต่อเครือข่ายค่อนข้างสูง

ตัวอย่างของอุปกรณ์ในกลุ่มนี้ ได้แก่ Pocket PC และ Set – Top Box ของเคเบิลทีวี

รูปที่ 2.25 ตัวอย่างอุปกรณ์ในกลุ่ม CDC

- CLDC (Connected Limited Device Configuration)

คุณสมบัติของอุปกรณ์ในกลุ่ม ได้แก่

- มีหน่วยความจำ 160 512 MB โดยควรมีหน่วยความจำแบบ Non – Volatile Memory อย่างน้อย 128 MB สำหรับไลบรารีของ CLDC และ Virtual Machine (VM) และควรมีหน่วยความจำแบบ Volatile Memory อย่างน้อย 32 KB สำหรับ Volatile Memory ในการรันโปรแกรม
- มีหน่วยประมวลผลขนาด 16 32 บิต ซึ่งมีความเร็วอย่างน้อย 25 MHz
- มีข้อจำกัดในการแสดงผล
- ความเร็วในการเชื่อมต่อเครือข่ายค่อนข้างต่ำ

ตัวอย่างอุปกรณ์ที่จัดอยู่ในกลุ่มนี้ ได้แก่ โทรศัพท์มือถือ เพจเจอร์ เป็นต้น

รูปที่ 2.26 ตัวอย่างอุปกรณ์ในกลุ่ม CLDC

<u>หมายเหตุ</u> : Non – Volatile Memory เป็นหน่วยความจำที่สามารถเก็บข้อมูลถาวรเอาไว้ได้ แม้ว่าจะปิดเครื่องหรือแบตเตอรี่หมดก็ตาม เราจึงนำหน่วยความจำประเภทนี้มาเก็บไลบรารีและ Volatile Memory ซึ่งจำเป็นจะต้องใช้ทุกครั้งได้ ในขณะที่ข้อมูลใด ๆ ก็ตามที่เก็บไว้ใน Volatile Memory จะหายไปถ้ามีการเปิดเครื่องใหม่ Volatile Memory จึงถูกนำมาใช้เก็บข้อมูลชั่วคราวขณะรันโปรแกรม

กลุ่มคลาสพื้นฐานที่มีให้เรียกใช้ซึ่งกำหนดเอาไว้ใน Configuration ของอุปกรณ์มือถือหรือ CLDC มีดังนี้

Package ใน CLDC	รายละเอียด
ที่สืบทอดมาจาก J2SE	
java.io	กลุ่มคลาสสำหรับรับส่งข้อมูล
java.lang	กลุ่มคลาสสำหรับภาษา Java
java.util	กลุ่มคลาสต่าง ๆ
ที่เพิ่มเติมใน CLDC	
javax.microedition.io	กลุ่มคลาสสำหรับรับส่งข้อมูลผ่านระบบเครือข่าย

ตาราง 2.7 แสดงกลุ่มคลาสพื้นฐานที่กำหนดไว้ใน CLDC

2.4.3 Profile

Profile เป็นตัวกำหนดกลุ่มของไลบรารีที่เพิ่มเติมมาจาก Configuration เพื่อรองรับข้อแตกต่างของอุปกรณ์แต่ละชนิด ดังนั้น Profile จึงเกี่ยวข้องกับคุณลักษณะทางด้านฮาร์ดแวร์ของอุปกรณ์แต่ละตัว เช่น อุปกรณ์มีช่องทาง (Interface) ติดต่อกับผู้ใช้อย่างไร หรืออุปกรณ์ติดต่อกับเครือข่ายได้อย่างไร เป็นต้น

Profile	Configuration	ตัวอย่างอุปกรณ์
MIDP	CLDC	โทรศัพท์มือถือ, เพจเจอร์ 2 ทาง
PDAP	CLDC	PDA
Personal	CDC	Pocket PC
RMI	CDC	อุปกรณ์ใด ๆ ก็ได้

ตาราง 2.8 แสดงตัวอย่าง Profile สำหรับอุปกรณ์ประเภทต่าง ๆ

2.4.3.1 MIDP โปรไฟล์สำหรับมือถือ

MIDP (Mobile Information Device Profile) เป็นกลุ่มคลาสไลบรารีที่รองรับการเขียนโปรแกรมบนโทรศัพท์มือถือ ใน MIDP 1.0 ได้กำหนด API พื้นฐานสำหรับการพัฒนาโปรแกรมดังนี้

- วงจรการทำงานของโปรแกรม
- การเชื่อมต่อเครือข่ายแบบ HTTP
- การติดต่อกับผู้ใช้ (User Interface)
- การเก็บข้อมูล (Persistent Storage)

2.4.3.2 การเขียนโปรแกรม MIDlet

การเขียนโปรแกรมเพื่อทำงานบนโทรศัพท์มือถือ จะใช้ MIDP และ CLDC ในการพัฒนา ซึ่งโปรแกรมหรือแอพพลิเคชั่นที่พัฒนาขึ้นมาสำหรับมือถือนั้นมีชื่อเรียกว่า MIDlet โดย Package ใน MIDP ที่เราจะนำมาใช้เขียนโปรแกรมมีดังนี้

Package ใน MIDP	รายละเอียด
javax.microedition.lcdui	กลุ่มคลาสสำหรับ User Interface ที่ใช้ในการติดต่อกับผู้ใช้
javax.microedition.midlet	กลุ่มคลาสสำหรับแอพพลิเคชั่นของ J2ME
javax.microedition.rms	กลุ่มคลาสสำหรับจัดการข้อมูลหน่วยความจำถาวรของอุปกรณ์

ตาราง 2.9 แสดงกลุ่มคลาสพื้นฐานใน MIDP

2.4.3.3 โครงสร้างของ MIDlet

ในการสร้าง MIDlet ทุกตัวจะต้องสืบทอดคลาส javax.microedition.midlet. MIDlet โดยคลาสนี้ประกอบไปด้วยเมธอด startApp(), pauseApp() และ destroyApp() ซึ่งใช้ในการควบคุมสถานการณ์ทำงานของ MIDlet โดยมีโครงสร้างดังนี้

```
import javax.microedition.lcdui.*;
import javax.microedition.midlet.*;

public class HelloWorld extends MIDlet
        //Constructor
        HelloWorld(){
        ....
        }
public void startApp(){
....
}
```

```
public void pauseApp(){
....
}
public void destroyApp(){
....
}
}
```

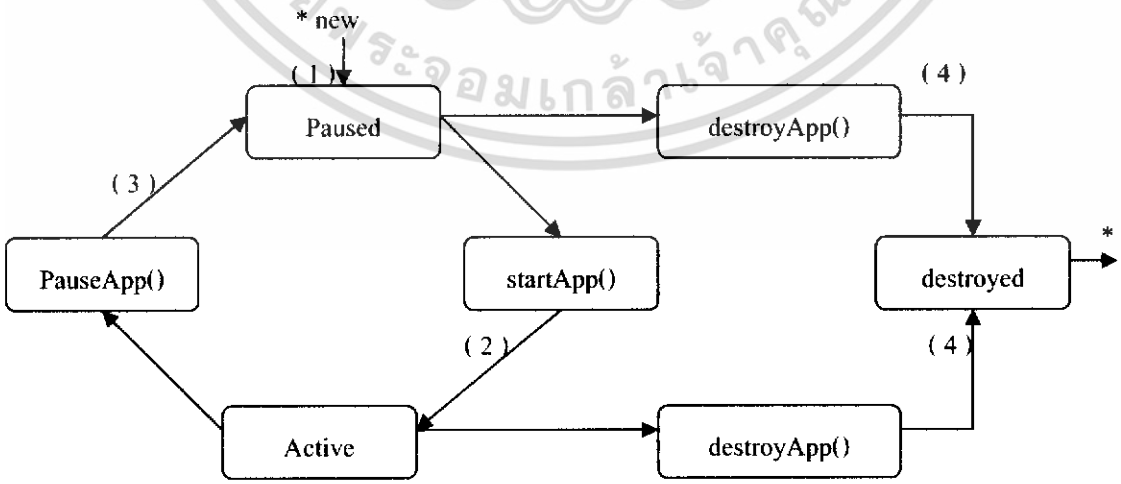
startApp(), pauseApp() และ destroyApp() เป็นเมธอดที่แสดงถึงสถานะต่าง ๆ ของ MIDlet เช่น กำลังทำงาน หยุดการทำงานชั่วคราว หรือยกเลิกการทำงาน โดยเราสามารถสรุปการใช้งานของแต่ละเมธอดได้ดังนี้

เมธอด	การใช้งาน
startApp	จัดหาทรัพยากรในระบบเพื่อกำหนดการเริ่มต้นทำงาน
pauseApp	หยุดการทำงานชั่วคราวเพื่อย้ายการทำงานไปยังส่วนอื่น ๆ
destroyApp	ยกเลิกทรัพพยากรที่จัดหามาและหยุดการทำงานของ MIDlet

ตาราง 2.10 แสดงการทำงานในแต่ละเมธอด

2.4.3.4 สถานะการทำงานของ MIDlet

การทำงานของ MIDlet และการเปลี่ยนสถานะของ MIDlet จากสถานะหนึ่งไปยังอีกสถานะหนึ่ง สามารถอธิบายได้ดังนี้



รูปที่ 2.27 แสดงสถานการณ์ทำงานของ MIDlet

1. Constructor สร้างตัวจำลอง MIDlet โดยในตอนนี้ MIDlet มีสถานะเป็นหยุดการทำงานชั่วคราว

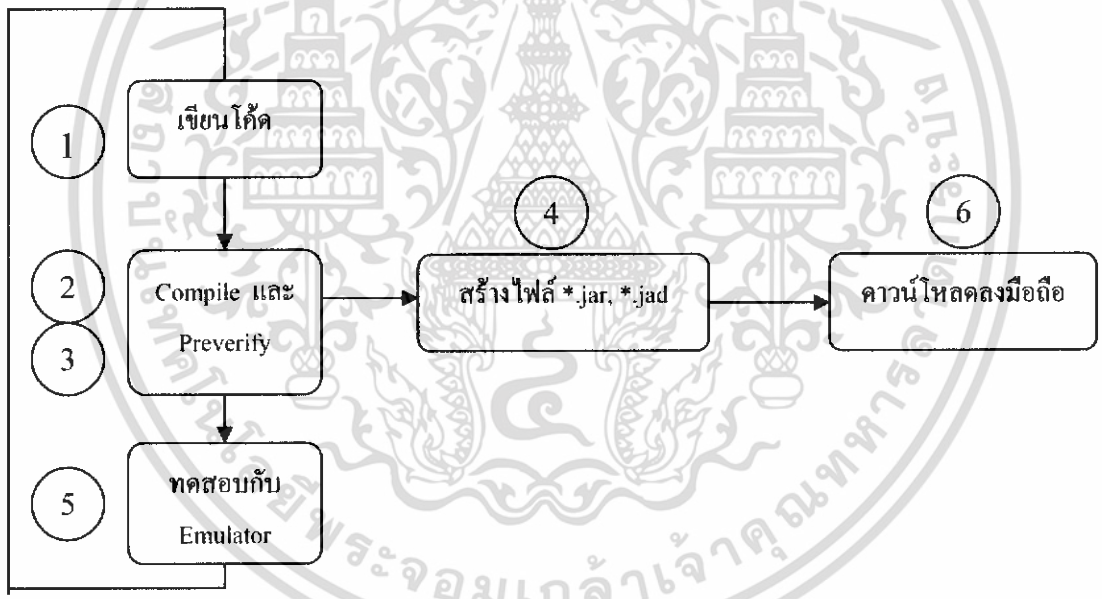
2. เมธอด startApp() ถูกเรียกเพื่อจัดหาทรัพยากรที่ต้องการในการทำงาน ในตอนนี้ MIDlet จะเปลี่ยนสถานะมาเป็น กำลังทำงาน

3. เมธอด pauseApp() ถูกเรียกเมื่อไม่ต้องการให้ MIDlet ทำงานต่อไป ซึ่ง MIDlet จะคืนทรัพยากรที่ดึงมาใช้งานและเปลี่ยนสถานะมาเป็นหยุดการทำงานชั่วคราว และวนกลับไปสถานะที่ 2 ใหม่เมื่อต้องการให้ MIDlet เริ่มต้นทำงานอีกครั้ง

4. เมธอด destroyApp() ถูกเรียกเมื่อไม่ต้องการเรียกใช้ MIDlet อีกต่อไป หรือเพื่อคืนหน่วยความจำให้แอพพลิเคชั่นอื่นได้ใช้งาน ในตอนนี้ MIDlet จะคืนทรัพยากรทั้งหมดและเปลี่ยนสถานะมาเป็น ยกเลิกการทำงาน

2.4.3.5 ขั้นตอนในการพัฒนา MIDlet

การพัฒนา MIDlet มีขั้นตอนและรายละเอียดดังนี้



รูปที่ 2.28 แสดงขั้นตอนการพัฒนาจาวาบนมือถือ

1. **เตรียมโค้ด** โดยอาจเขียนใน Notepad, Editplus หรือ Text Editor ตัวอื่นที่ถนัด แล้วบันทึกไฟล์ให้มีนามสกุล *.java

2. **คอมไพล์โค้ด** ในขั้นตอนนี้จะได้ผลลัพธ์เป็นไฟล์ *.class

3. **ตรวจสอบ (Preverify)** นำไฟล์ *.class ที่ได้จากขั้นตอนที่ 2 มาตรวจสอบว่าไฟล์ *.class ทำงานได้ถูกต้องหรือไม่

4. **สร้างไฟล์ *.jar** นำไฟล์ *.class ที่ผ่านการตรวจสอบและไฟล์ Resource ทั้งหมดลงในไฟล์ *.jar

5. **ทดสอบโปรแกรม** โดยใช้ Emulator

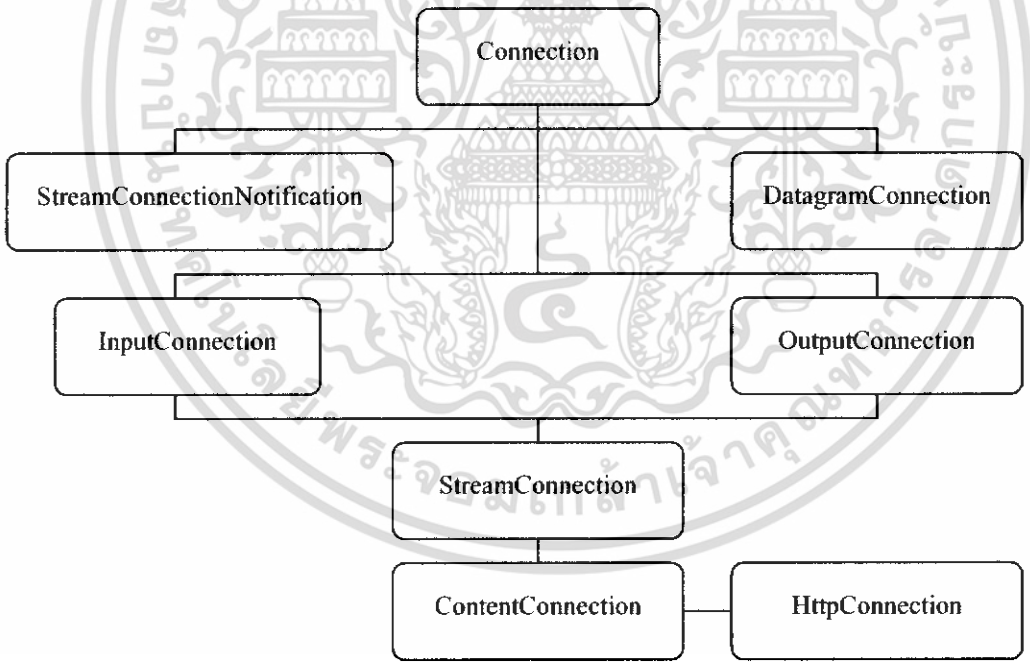
6. **ใช้งานจริง** ดาวน์โหลดแอพพลิเคชั่นลงในโทรศัพท์มือถือ

2.4.4 การเชื่อมต่อกับเครือข่ายไร้สาย

จุดเด่นของ MIDlet คือ ความสามารถในการต่อเข้าระบบเครือข่ายไร้สาย (Wireless Connections) การเชื่อมต่อเครือข่ายของ J2ME จะต้องมีความยืดหยุ่นสูงเพื่อรองรับอุปกรณ์หลากหลายประเภท ขณะเดียวกันก็จะต้องสนับสนุนเฉพาะอุปกรณ์ด้วย โดย J2ME ได้ดึงเอาบางส่วนของแพ็คเกจ java.io และแพ็คเกจ java.network มาใช้และได้กำหนดขอบเขตการเชื่อมต่อกับระบบเครือข่ายขึ้นมาใหม่ โดยเรียกว่า Generic Connection Framework (GCF)

- Generic Connection Framework (GCF)

Generic Connection Framework (GCF) คือ โครงสร้างทั่วไปของการเชื่อมต่อเครือข่ายที่มีความยืดหยุ่นต่อความหลากหลายในรูปแบบการเชื่อมต่อ Generic Connection Framework ประกอบด้วย 1 คลาส คือ คลาส Connector และ 7 อินเตอร์เฟส นอกจากนี้ MIDP ยังได้กำหนดอินเตอร์เฟสเพิ่มอีก 1 อินเตอร์เฟส คือ HttpConnection ซึ่งทั้งหมดนี้ถูกบรรจุอยู่ในแพ็คเกจ javax.microedition.io



รูปที่ 2.29 แสดงอินเตอร์เฟสใน Generic Connection Framework

อินเตอร์เฟสแต่ละตัวมีหน้าที่ต่างกัน ดังนี้

อินเตอร์เฟส	หน้าที่
Connection	การเชื่อมต่อแบบพื้นฐาน เช่น เปิดหรือปิดการเชื่อมต่อ
InputConnection	การเชื่อมต่อเพื่ออ่านข้อมูล
OutputConnection	การเชื่อมต่อเพื่อเขียนข้อมูล
StreamConnection	รวมการเชื่อมต่อทั้งเพื่ออ่านและเขียนข้อมูล ใช้ในการเชื่อมต่อแบบ Socket
ContentConnection	สืบทอดจากอินเตอร์เฟส StreamConnection ใช้จัดการกับข้อมูลผ่านการเชื่อมต่อแบบ HTTP
StreamConnection Notification	รอให้การเชื่อมต่อกับเครือข่ายสำเร็จและคืนออบเจ็กต์ของอินเตอร์เฟส StreamConnection
DatagramConnection	การเชื่อมต่อแบบ Datagram

ตาราง 2.11 แสดงหน้าที่ของอินเตอร์เฟสแต่ละตัว

2.4.5 ขั้นตอนการทำงานของ ECG MIDlet

2.4.5.1 Program Specification (ที่ใช้ในการพัฒนา)

- Java version: 1.5.0_016
- J2ME Wireless Toolkit version: 2.3 Beta
- Emulator: Series_60_MIDP_Concept_SDK_Beta_0_3_1_Nokia_edition
- MIDP version: 2.0
- CLDC version: 1.0
- Display Resolution : 176 x 208 pxl

โปรแกรมนี้สามารถแสดงผลได้อย่างถูกต้องบนโทรศัพท์มือถือ Nokia Series 60

2.4.5.2 ขั้นตอนการทำงานของโปรแกรม

โปรแกรมนี้จะแบ่งการทำงานออกเป็น 2 ส่วน คือ

- ส่วนของการควบคุม MIDlet และติดต่อกับ HTTP Server โดยไฟล์ HeartECG.java จะเป็นคลาสหลักที่ใช้ในการควบคุม MIDlet และติดต่อกับ HTTP Server
- ส่วนคือการนำข้อมูลดิบมาวาดเป็นกราฟ และไฟล์ ECGCanvas.java เป็นคลาสที่ใช้ในการวาดกราฟ ซึ่งลำดับขั้นการทำงานจะเป็นดังนี้

1. เริ่มต้น J2ME จะเรียก method startApp() เสมอเพื่อเริ่มการทำงาน startApp() จะทำการแสดงผลหน้าฟอร์มที่ให้ผู้ใช้กรอก URL ลงไป ส่วน Constructor ของคลาส HeartECG นั้นจะถูกเรียกมาโดยอัตโนมัติเนื่องจากคลาสนี้สืบทอดมาจากคลาส MIDlet ทำให้ระบบจะจัดการเรียก Constructor ให้เอง เมื่อดูใน Constructor แล้วจะเห็นว่ามีการสร้างฟอร์มอยู่ ซึ่งฟอร์มนี้จะประกอบด้วย Text Field และปุ่มต่างๆ เมื่อผู้ใช้กดปุ่มใดๆ method command action() ก็จะถูกทำงาน และการทำงานก็จะเป็นไปตามที่เรากำหนดไว้สำหรับแต่ละปุ่มเท่านั้น

รูปที่ 2.30 การแสดงผลหน้าฟอร์มที่ให้ผู้ใช้กรอก URL ลงไป

เมื่อผู้ใช้กรอก URL และกดปุ่ม Connect แล้ว method commandAction() ก็จะทำงานทันที

2. Method commandAction() จะตอบสนองตามปุ่มที่เรากด ในที่นี้ เมื่อเรากดปุ่ม Connect ก็จะมีการทำงาน 2 อย่างคือ 1. สร้าง Object ของ Canvas เอาไว้เพื่อที่จะทำการแสดงผล 2. สร้าง Thread เพื่อใช้ในการติดต่อกับ HTTP Server โดยเอาค่าจาก TextField ที่เรากรอกมาเป็น URL เก็บไว้ในตัวแปร inputURL ก่อน จากนั้นจึงทำการเริ่มการทำงานของ Thread t โดยสั่ง t.start() ซึ่ง t.start() จะเป็นการเรียก method run() โดยอัตโนมัติ ภายใน method run() (ทำงานคนละ Thread กับ Thread หลัก) ก็จะทำการเริ่มการติดต่อกับ Server โดยเรียกใช้ method getViaHttpConnection()
3. getViaHttpConneciton() เมื่อไล่ดูโค้ดแล้ว แต่ละบรรทัด มีการทำงานดังนี้
   3.1. สร้าง Buffer เพื่อรอเก็บข้อมูลจาก InputStream ที่มาจาก Server
   3.2. เปิดการเชื่อมต่อกับ Server ตาม URL ที่ผู้ใช้กำหนด (Connector.open(url))
   3.3. สั่งให้ Canvas เขียนคำว่า Loading... เพื่อรอรับข้อมูลเพื่อจะนำไปวาด
   3.4. เก็บค่า InputStream จาก Server ไว้ในตัวแปร
   3.5. ในลูป while จะเป็นการอ่านค่าจาก InputStream มาทีละตัวอักษรเก็บไว้ใน Buffer ซึ่งค่าที่ได้นี้ก็คือค่าตัวอักษรแต่ละตัวใน Tag HTML ที่โหลดมาจาก Server
   3.6. ใน Finally หมายความว่าถ้าไม่มีข้อมูลเหลืออยู่ใน InputStream แล้วก็ให้ปิดไป

3.7. นำค่า Buffer ที่ได้มาเก็บเป็น String และนำไปหาค่า Amplitude ที่ฝังอยู่ใน Tag HTML ออกมา และเก็บไว้ใน Array ชื่อว่า ampVal โดยใช้ method paramToArray() จากนั้นก็เซ็ตค่า Array นี้ ให้กับ Canvas เพื่อนำไปวาดออกมาทางหน้าจอ

4. method paramToArray() ภายในจะมีการวนลูปเพื่อที่จะเซ็ตค่าให้กับ Array แต่ละตัวโดยค่าของตัวเลขแต่ละตัวที่จะนำมาเก็บใน Array นั้นจะได้มาจากการส่งค่า index ของ parameter ไปทำการค้นหาใน Tag HTML ที่ได้มา โดยใช้ method parse(int index)

```
.....
<param name=a0 value=5>
<param name=a1 value=5>
<param name=a2 value=5>
<param name=a3 value=5>
<param name=a4 value=5>
<param name=a5 value=0>
.....
```

โดยสมมุติเมื่อส่งค่า index = 0 มันก็จะไปหาบรรทัดที่มีคำว่า "...name = a0" แล้วก็หาค่าที่อยู่หลังจากคำว่า "value =" และอยู่ก่อนเครื่องหมาย ">" ซึ่งเราก็จะได้ค่า 5 มา จากนั้นนำ 5 ซึ่งเป็น String อยู่นั้นมาแปลงเป็น Integer อีกโดยใช้ Integer.parseInt(); ซึ่งรายละเอียดของ parse() จะอธิบายถัดจากนี้

5. method parse() ขั้นแรกทำการหาค่า offset ก่อนซึ่งค่า offset นี้คือค่าที่ใช้ในการเลื่อนตำแหน่งของตัวอักษรที่ต้องการจากจุดเริ่มที่มีคำว่า "name=axxx value=" ไปยังค่าตัวเลขที่ต้องการ
เช่น ถ้าค่า index = 1 เริ่มแรกเราจะได้ค่า offset = 14
จากนั้น ก็หาตำแหน่งของคำว่า "name=a1 value=" เมื่อได้ตำแหน่งแล้ว ก็เลื่อนตำแหน่งไป 14 ตัวอักษร ก็จะเจอ "5> ...." แล้วก็ทำการตัด String ด้านหน้าทิ้งไป แล้วก็ ตัดอีกครั้งโดยเริ่มที่ตัวอักษรตัวแรกถึงตัวอักษร ">" เราก็จะได้ "5" ออกมา แล้วนำค่านี้มาแปลงให้เป็น Integer ส่งกลับไปเก็บไว้ใน Array

อธิบายเรียงแต่ละขั้นตอนเป็นดังนี้

5.1. ในส่วนที่ดักเงื่อนไข if นั้น เพื่อหาค่า offset โดยคำนวณจากค่า index ที่ได้รับมา

5.2. หาตำแหน่งเริ่มต้นของ คำที่เราใช้อ้างอิง ซึ่งเราได้ใช้คำว่า "name=axxx value="

5.3. หาตำแหน่งของตัวอักษรที่เราจะนำมาเป็นตัวเลข Amplitude โดยนำตำแหน่งที่เราหาได้จากข้อที่แล้วมาบวกกับค่า offset (ซึ่งถ้าลองนับดูจะเป็นว่าเมื่อเราเริ่มนับจาก ตัว "n" ในคำว่า "name=a1 value" ไป 14 ตัวอักษร เราก็จะเจอเลข 5)

5.4. ตัด String ส่วนที่อยู่หน้าเลข 5 ออก ซึ่งเราก็จะเหลือ Tag Html ตั้งแต่เลข 5 นี้ไปจนจบ
(String body = valBuffer.substring(keyIndex);)

5.5. ตัดเฉพาะตัวเลข 5 ออกมาโดยหลักการคือ ตัดตัวอักษรโดยนับตั้งแต่ตัวแรก ไปจนกว่าจะเจอเครื่องหมาย " > "

5.6. นำค่าที่แปลงเป็น integer แล้ว return ไปเก็บไว้ใน Array

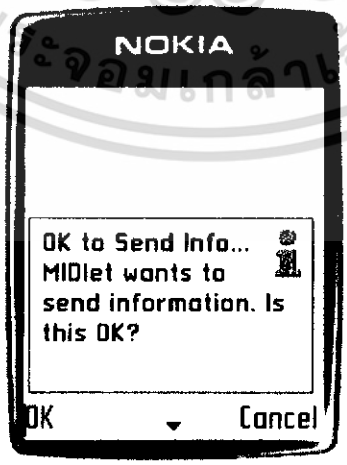
6. เมื่อได้ Array ของ ค่า Amplitude แล้ว ก็นำไปเซ็ตค่าใน Canvas เพื่อที่จะนำค่าที่ได้มาวาดต่อไป ส่วนที่เหลือนี้เป็นการทำงานใน ECGCanvas ที่ใช้ในการแสดงผลกราฟฟิก

7. Method ที่สำคัญใน ECGCanvas ก็คือ method setValue() และ paint()
SetValue() จะใช้ในการเซ็ตค่า Array ที่ได้มาจากคลาสหลัก โดยที่ method paint() จะนำค่าที่ได้จาก Array นี้ไปวาดกราฟ โดยรายละเอียดของแต่ละส่วนที่นำมาวาดนั้นได้อธิบายไว้ใน Source Code แล้ว

รูปที่ 2.31 แสดงรูปกราฟคลื่นไฟฟ้าหัวใจจำลอง

2.4.5.3 ข้อจำกัดของโปรแกรม

1. การทำงานจะต้องมีการติดต่อกับ Server ทุกๆ 15 วินาที ซึ่งในการติดต่อแต่ละครั้ง ผู้ใช้ต้องทำการตอบ Accept การเชื่อมต่อทุกครั้ง อันเนื่องมาจาก method Connector.open(url) นั้นในการทำงานต้องถามการเปิดการติดต่อก่อน ถ้าจะให้สั่ง Connector.open(url) เพียงครั้งเดียวแล้วไป Get inputStream มาทุก 15 วินาทีก็ไม่ได้เนื่องจากข้อมูลนั้นจะไม่ตามมาได้ต้องเปิดการ connect ก่อนทุกครั้ง



รูปที่ 2.32 การถามการเปิดการติดต่อก่อน

2. การวาดภาพ โปรแกรมนี้จะไม่ได้แสดงการวาดทีละเส้นเป็น Animation แต่จะวาดภาพออกมาทำเดียวทั้งชุด ซึ่งผู้ใช้ต้องกดปุ่มซ้ายหรือขวาเพื่อดูกราฟเอาเอง
3. เวลาที่ใช้ในการ Connect ใหม่คือ 15 วินาที ซึ่งผู้พัฒนาอาจจะเห็นว่าไม่เหมาะสม ถ้าจะแก้ไขต้องไปในไฟล์ HeartECG.java โดยแก้ค่า INTERVAL เป็นค่าช่วงเวลาที่เราต้องการ หน่วยเป็น มิลลิวินาที

บทที่ 3
การออกแบบวงจร

3.1 การออกแบบสามารถแบ่งออกเป็น 3 ส่วนใหญ่ ๆ คือ

1. เครื่องวัดคลื่นไฟฟ้าหัวใจ (ECG : Electrocardiograph)

ECG เป็นเครื่องมือที่ใช้ในการวัดสัญญาณคลื่นไฟฟ้าหัวใจ ซึ่งสัญญาณคลื่นไฟฟ้าหัวใจจะมีขนาดสัญญาณน้อยมาก ซึ่งประมาณ 1 มิลลิโวลท์ โดยรับสัญญาณจากอิเล็กโทรดที่ติดที่ผิวหนังซึ่งจะต้องมีส่วนขยายสัญญาณ (Amplifier) เพื่อแสดงผลของสัญญาณคลื่นไฟฟ้าหัวใจออกทางเอาท์พุต นอกจากนี้ จะต้องมีส่วนขยายสัญญาณเสียง (Audio Amplifier)

2. RABBIT

Rabbit 2000 นี้ได้พัฒนามาจากไมโครคอนโทรลเลอร์ Z180 ซึ่ง Rabbit สามารถใช้ในการอินเตอร์เฟสเข้ากับเครือข่ายอินเตอร์เน็ต ซึ่งการเขียนโปรแกรมเพื่อใช้งานบน Rabbit โดยใช้ไดนามิก C ในการเขียน

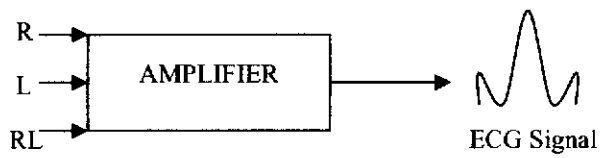
3. J2ME

ส่วนประกอบที่สำคัญของ J2ME ได้แก่ JVM (Java Virtual Machine), Configuration ที่เป็นตัวระบุ Virtual Machine และคลาสไลบรารีพื้นฐานและ Profile ที่เป็นตัวกำหนดกลุ่มของไลบรารีที่เพิ่มเติมมาจาก Configuration เพื่อรองรับข้อแตกต่างของอุปกรณ์แต่ละชนิด สำหรับ Profile ของโทรศัพท์มือถือจะต้องมี MIDP (Mobile Information Device Profile) ที่รองรับการเขียนโปรแกรมบนโทรศัพท์มือถือ ซึ่งโปรแกรมนั้นมีชื่อว่า MIDlet โดยขั้นตอนในการพัฒนามีดังนี้

- เตรียมโค้ด (นามสกุล*.java)
- คอมไพล์โค้ด (ได้ไฟล์ *.class)
- ตรวจสอบว่าไฟล์ *.class ทำงานได้ถูกต้องหรือไม่
- สร้างไฟล์ *.jar, *.jad
- ทดสอบโปรแกรม โดยใช้ Emulator
- ใช้งานจริง คือ ดาวน์โหลดแอพพลิเคชั่นลงในโทรศัพท์มือถือ

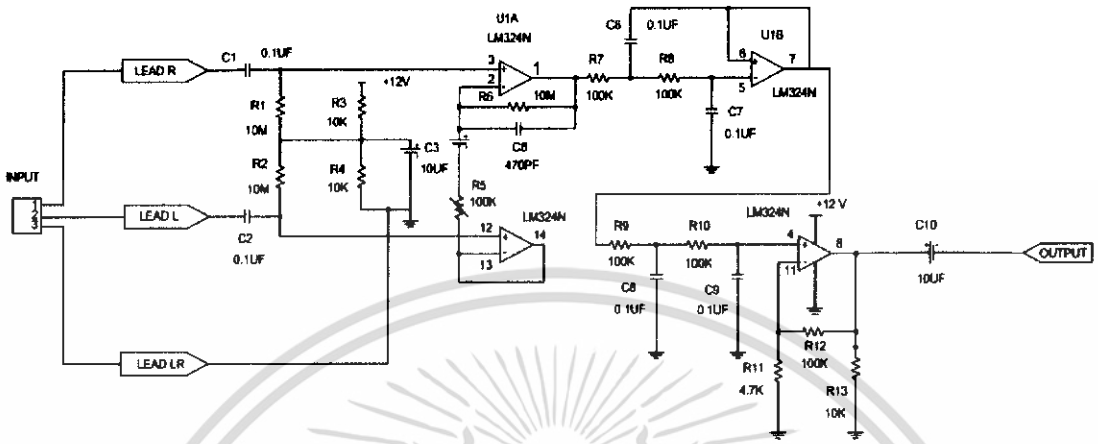
3.2 การออกแบบวงจรและการสร้างภาคขยายคลื่นไฟฟ้าหัวใจ

การออกแบบภาคขยายคลื่นหัวใจไฟฟ้า เราสามารถออกแบบเป็นบล็อกไดอะแกรมได้ดังนี้



รูปที่ 3.1 บล็อกไดอะแกรมของภาคขยายคลื่นไฟฟ้าหัวใจ

3.2.1 ส่วนที่ทำหน้าที่ขยายคลื่นไฟฟ้าหัวใจ



รูปที่ 3.2 แสดงวงจร Amplifier

วงจรขยายความแตกต่าง (Differential Amplifier) ดังแสดงในรูป 3.2 เป็นวงจรแรกที่ขยายคลื่นไฟฟ้าหัวใจ ที่มีขนาดของสัญญาณน้อยมากเพียงประมาณ 1 มิลลิโวลต์ โดยการรับสัญญาณจากอิเล็กโทรดที่ติดบนผิวหนัง ซึ่งมีค่าความต้านทานสูง ดังนั้น วงจรที่จะนำมาใช้ต้องมีคุณสมบัติพิเศษดังต่อไปนี้

ก. อินพุทอิมพีแดนซ์สูงมาก เมื่อเทียบกับความต้านทานของผิวหนัง เพื่อป้องกันการเสียสมดุลของวงจรและการบั่นทอนสัญญาณที่ป้อนเข้าสู่อินพุท การเสียสมดุลของวงจรจะมีผลเสียต่อวงจรขยาย คือ สัญญาณรบกวนที่เข้ามาในลักษณะคอมมอนโหมด (Common Mode Signal) ไม่สามารถกำจัดออกไปได้และยังทำให้เกิดศักดาไฟฟ้าออฟเซ็ท (Offset Voltage) ซึ่งจะถูกขยายให้มีขนาดมากขึ้นที่เอาท์พุท ถ้าศักดาไฟฟ้าออฟเซ็ทมีค่ามากจะทำให้วงจรขยายอิ่มตัว มีศักดาไฟฟ้าเอาท์พุทอยู่ที่ค่าเกือบเท่ากับศักดาไฟฟ้าของแหล่งจ่ายไฟด้านใดด้านหนึ่ง (บวกหรือลบ) และวงจรไม่สามารถทำงานได้

ข. ค่า CMRR (Common Mode Rejection Ratio) สูง ค่า CMRR เป็นคุณสมบัติอย่างหนึ่งของวงจรขยายความแตกต่างที่สามารถกำจัดสัญญาณรบกวนได้ คุณสมบัติอันนี้ก็คือ การมีอัตราขยายของสัญญาณดิฟเฟอร์เรนเชียลโหมด (Differential Mode Signal) สูงและมีอัตราขยายของสัญญาณคอมมอนโหมดต่ำ ทั้งนี้เนื่องจากสัญญาณที่ต้องการขยาย (ECG) จะเข้าไปที่อินพุทในลักษณะสัญญาณคอมมอนโหมด

3.2.2 วงจรแปลงสัญญาณจากอนาลอกเป็นดิจิตอล (A/D Converter : ADC)

โดยทั่วไปแล้ว ADC จะแบ่งออกเป็นชนิดที่ให้เอาท์พุทออกมาเป็นเลขฐานสิบ และเป็นเลขฐานสอง ADC ที่ทำหน้าที่เปลี่ยนสัญญาณอนาลอกเป็นสัญญาณดิจิตอลที่มีเอาท์พุทเป็นเลขฐานสิบ

มักจะใช้เป็นดิจิตอลโวลต์มิเตอร์และถูกใช้ใน digital panel meter และ DMM คอนเวอร์เตอร์ ผลที่ได้จากการทำงานของ ADC เป็นเลขฐานสอง เอาท์พุทที่ออกมาจะเป็นเลขแบบหลายชนิด สำหรับแบบที่ให้เอาท์พุทออกมาเป็นเลขฐานสิบ (เช่น 3 ½ หรือ 4 ½) โดยทั่วไปแล้ว ADC ที่มีเอาท์พุทเป็นเลขฐานสองจะมีจำนวนบิตเป็น 4, 6, 8, 10, 12, 14 และ 16 บิต ซึ่งอาจมีอาการ error ขึ้นบ้างเล็กน้อย เนื่องจากการใช้ discrete binary step เพื่อแทนสัญญาณอนาลอกที่มีความต่อเนื่องกันเรียกว่า quantizing error

ADC ขนาด 16 บิต มีความถูกต้องละเอียดแม่นยำมากกว่าแบบ 4 บิต เพราะว่ามันแบ่งอินพุทหรืออ้างอิงโวลต์เตจเป็น discrete step ที่เล็ก ๆ ตัวอย่างเช่น แต่ละ step ใน ADC แบบ 4 บิต จะต้องเป็นหนึ่งในสิบห้า ($2^4 - 1 = 15$) ของอินพุทโวลต์เตจ ผลที่ออกมาคือ 6.7% (1/15*100 = 6.7 เปอร์เซ็นต์)

อย่างไรก็ตาม ในกรณีของ ADC ขนาด 8 บิต ควรจะมี discrete step เป็นจำนวน 255 ($2^8 - 1 = 255$) ซึ่งจะเท่ากับ 0.39 เปอร์เซ็นต์ (1/255 *100 = 0.39 เปอร์เซนต์) ซึ่ง ADC แบบ 8 บิตมีความละเอียดแม่นยำมากกว่าแบบ 4 บิต

นอกจากนี้ความผิดพลาดที่เกิดขึ้นใน ADC อีกอย่างหนึ่งก็คือ analog component เช่น วงจรเปรียบเทียบและความผิดพลาดอื่น ๆ อันเนื่องมาจากโครงข่ายวงจรของตัวต้านทาน ความละเอียดแม่นยำของ ADC เรียกว่า accuracy ของไอซี ADC โดยค่า accuracy ของไอซี ADC ที่มีเอาท์พุทเป็นเลขฐานสองมีช่องกว้างจาก ± ½ LSB ถึง +2 LSB ส่วน accuracy ของไอซี ADC ที่มีเอาท์พุทเป็นเลขฐานสิบจะมีช่องกว้างตั้งแต่ 0.01 ถึง 0.05 เปอร์เซ็นต์

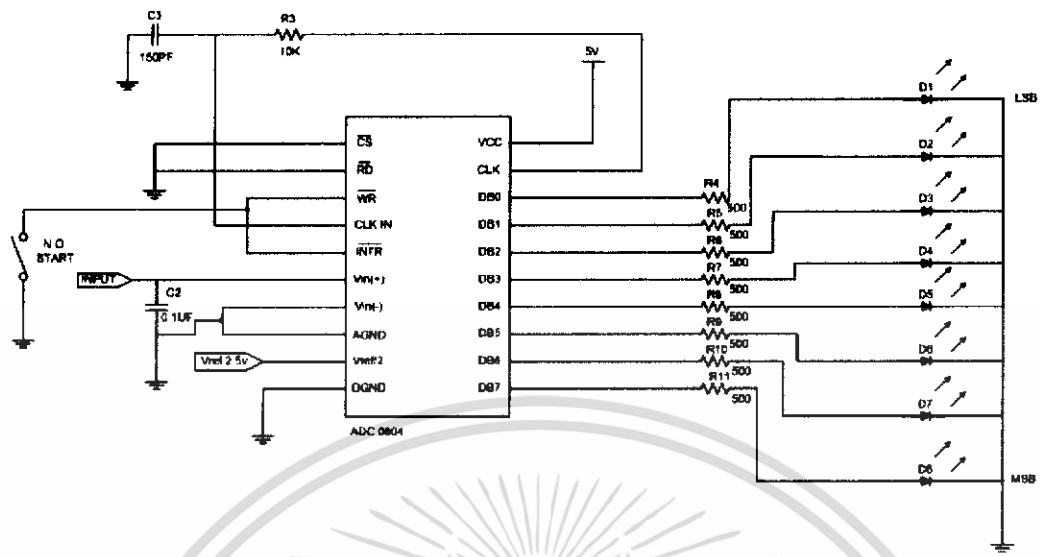
Conversion time เป็นรายละเอียดเฉพาะของ ADC อีกอันหนึ่งที่สำคัญ ซึ่งเป็นเวลาที่ ADC ใช้ในการให้ไอซีเปลี่ยนค่าอนาล็อกทางอินพุท ให้เป็นเอาท์พุทเลขฐานสอง (หรือเลขฐานสิบ) โดยทั่วไปแล้ว conversion time ของ ADC ที่มีเอาท์พุทเป็นเลขฐานสิบมักอยู่ในช่วง 200 ถึง 400 ms ส่วน ADC ที่มีเอาท์พุทเป็นเลขฐานสองมักมีค่า conversion time อยู่ระหว่าง 0.05 ถึง 100,000 μs

โดยทั่วไปนอกเหนือจากที่กล่าวมาของ ADC คือ ค่าแรงดันจากแหล่งจ่ายไฟมักมีค่าประมาณ +5 V ระดับแรงดันเอาท์พุทเป็นทั้งแบบ TTL, CMOS หรือ tristate (3 สถานะ) ช่วงกว้างของอินพุทโวลต์เตจ มักจะเป็น 5 V ค่าการสูญเสียกำลังสูงสุดของ ADC มักมีค่าอยู่ระหว่าง 15 ถึง 3000 mW

บางขาของไอซี ADC 0804 อาจเหมือนกับขาของไมโครโปรเซสเซอร์ที่เราใช้งานกันทั่วไป เช่น ADC 0804 ใช้ชื่อขา INTR, WR, RD ซึ่งคล้ายกับขา INTR, WR, RD ในไมโครโปรเซสเซอร์ 8085 ไอซี ADC 0804 สามารถเชื่อมต่อกับไมโครโปรเซสเซอร์แบบ 8 บิต

ขา CS Control input ใช้สำหรับปรับสัญญาณ (chip select) จากวงจรถอดรหัสค่าแอดเดรสในไมโครโปรเซสเซอร์

ADC 0804 มีเอาท์พุทเป็นเลขฐานสองและมี conversion time เพียง 100 μs เท่านั้น อินพุทและเอาท์พุทของมันเข้ากันได้ทั้ง MOS และ TTL มีตัวกำเนิด clock รวมอยู่ในชิฟสำเร็จรูปอยู่แล้ว โดยจะต้องต่ออุปกรณ์ภายนอก (ตัวต้านทาน, ตัวเก็บประจุ) เพิ่มเติมเพื่อให้ทำงานได้ ไอซี ADC 0804 ทำงานด้วยไฟฟ้ากระแสตรง 5 V จากเพาเวอร์ซัพพลาย และสามารถใส่แรงดันทางอินพุทได้ตั้งแต่ 0 ถึง 5 V ไอซี ADC 0804 converter สามารถนำมาทดสอบได้โดยใช้วงจรดังรูปที่ 3.3



รูปที่ 3.3 วงจรแปลงสัญญาณอนาลอกเป็นดิจิตอล

หน้าที่ของวงจรคือ ใส่รหัสความแตกต่างของแรงดันไฟฟ้า $V_{in}(+)$ และ $V_{in}(-)$ เปรียบเทียบกับแรงดันอ้างอิง ในที่นี้คือ 2.5 V เพื่อให้สัมพันธ์กับค่าตัวเลขฐานสอง

บทที่ 4
การทดลองและผลการทดลอง

4.1 ทดสอบการทำงานวงจรแอมปลิไฟเออร์ (Amplifier)

การวัดค่าการตอบสนองความถี่ของวงจรแอมปลิไฟเออร์ เป็นหัวใจหลักของวงจรขยายคลื่นไฟฟ้าหัวใจ เราต้องการวัด CMRR (Common Mode Rejection Ratio) ของวงจร ซึ่ง CMRR นี้คือค่าอัตราส่วนของอัตราการขยายแบบโหมดความแตกต่าง (Differential Mode) กับอัตราการขยายแบบโหมดร่วม (Common Mode) ซึ่งค่านี้ควรมีค่าสูง ๆ เพื่อเพิ่มขีดความสามารถในการกำจัดสัญญาณรบกวนที่เข้ามา และขยายสัญญาณความถี่คลื่นไฟฟ้าหัวใจให้แรงขึ้น ซึ่งการวัดในโหมดความแตกต่างทำได้โดยการนำสัญญาณคลื่นซายน์เวฟค่าแรงดันต่ำ ป้อนเข้าสู่อินพุท ส่วนอีกอินพุทหนึ่งก็ทำการต่อลงกราวน์ แล้วทำการเปลี่ยนค่าความถี่ ส่วนในแบบโหมดร่วมก็ทำโดยการนำอินพุททั้งสองมาช็อตกัน แล้วก็ป้อนสัญญาณเหมือนกับโหมดความแตกต่าง แล้วนำมาคำนวณหาค่า CMRR ซึ่งสามารถหาได้จากสูตร

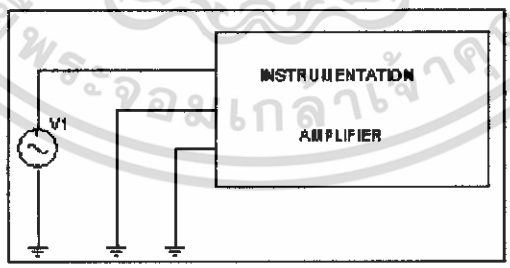
$$\text{CMRR} = 20 \log \frac{Adm}{Acm}$$

โดย Adm คือ อัตราขยายของวงจรแบบดิฟเฟอเรนเชียลโหมด

Acm คือ อัตราขยายของวงจรแบบคอมมอนโหมด

4.1.1 การทดลองหาอัตราขยายของวงจรขยายแบบดิฟเฟอเรนเชียลโหมด

1. ต่อวงจรตามรูปที่ 4.1
2. ป้อนคลื่นสัญญาณซายน์เวฟ (Sine Wave) ค่าแรงดันต่ำ ๆ ที่ความถี่ต่าง ๆ
3. วัดแรงดันเอาท์พุทของวงจรแอมปลิไฟเออร์

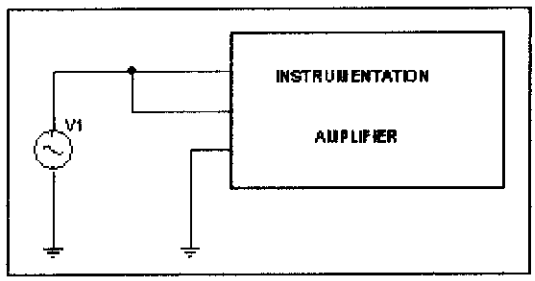


รูปที่ 4.1 วงจรที่ใช้หาอัตราขยายของวงจรขยายแบบดิฟเฟอเรนเชียลโหมด

4.1.2 การทดลองหาอัตราขยายของวงจรขยายแบบคอมมอนโหมด

1. ต่อวงจรดังรูปที่ 4.2
2. ป้อนคลื่นสัญญาณซายน์เวฟ (Sine Wave) ค่าแรงดันต่ำ ๆ ที่ความถี่ต่าง ๆ

3. วัดแรงดันเอาท์พุทของวงจรแอมปลิไฟเออร์



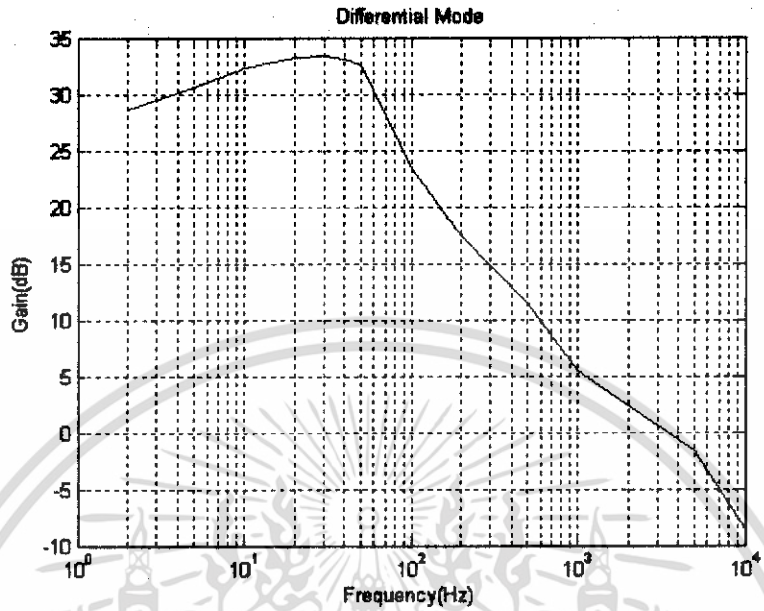
รูปที่ 4.2 วงจรที่ใช้หาอัตราขยายของวงจรขยายแบบคอมมอนโหมด

เมื่อ Vin = คลื่นซายน์เวฟขนาด 540 mVp-p

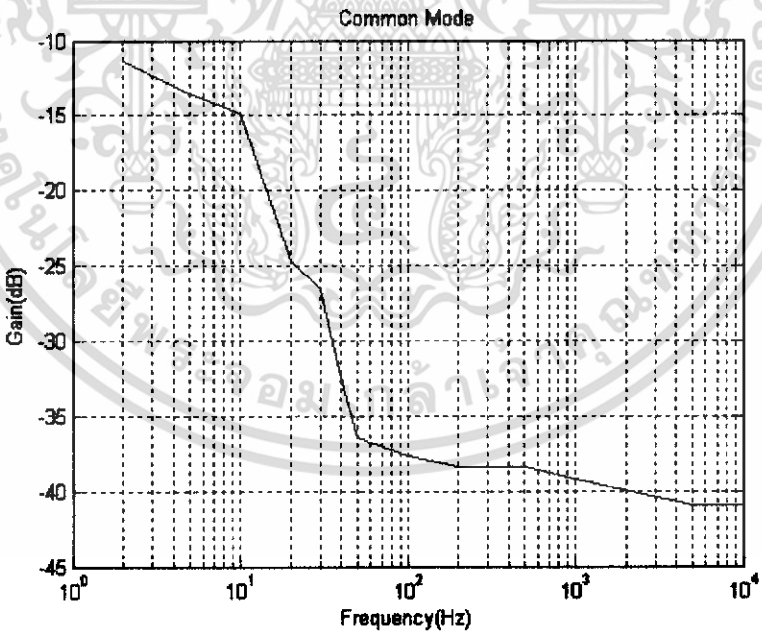
ความถี่ (Hz)	Differential Mode Gain			Common Mode Gain			CMRR	
	ขนาด (Vp-p)	อัตราขยาย (V/V)	อัตราขยาย (dB)	ขนาด (mVp-p)	อัตราขยาย (V/V)	อัตราขยาย (dB)	V/V	dB
2	14.6	27.03	28.63	150	0.27	-11.12	97.33	39.86
5	18.4	34.07	30.65	111	0.21	-13.55	201.8	46.1
10	22.1	40.92	32.23	99.4	0.18	-14.7	222.33	46.90
20	24.6	45.55	33.17	31.7	0.058	-24.73	776.02	57.80
30	25.1	46.48	33.35	21.6	0.047	-26.55	1162.03	61.30
40	24.5	45.37	33.14	13	0.024	-32.37	1869.23	65.43
50	23.8	42.22	32.51	8	0.015	-36.47	2975	69.47
100	10.9	14.63	23.30	7.3	0.013	-37.38	1493.15	63.48
200	4.0	7.41	17.39	7	0.012	-38.41	571.43	55.14
500	2.03	3.75	11.50	6.6	0.012	-38.41	307.58	49.76
1000	1.02	1.88	5.52	6	0.011	-39.08	170	44.61
5000	0.45	0.83	-1.58	5	0.009	-40.66	90	39.08
10000	0.2	0.37	-8.62	5	0.009	-40.66	40	32.04

ตารางที่ 4.1 ผลการทดลองการวัดแรงดันทางเอาท์พุทของวงจรแอมปลิไฟเออร์

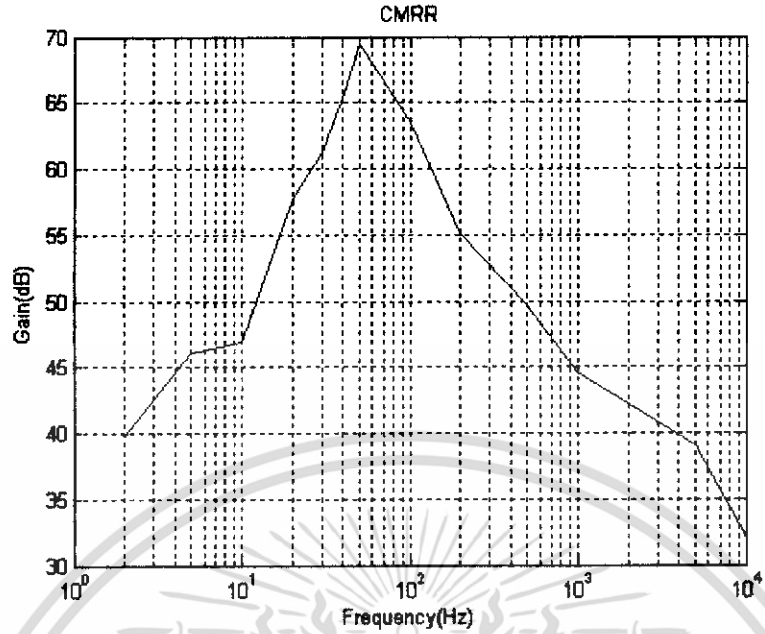
ตารางที่ 4.1 จะพบว่าค่า CMRR จะแปรผกผันกับความถี่คือ ที่ความถี่ต่ำมีค่า CMRR สูง ส่วนที่ความถี่สูงจะมีค่า CMRR ต่ำ ซึ่งเหมาะสำหรับวงจรคลื่นไฟฟ้าหัวใจที่ใช้ในโครงงานนี้



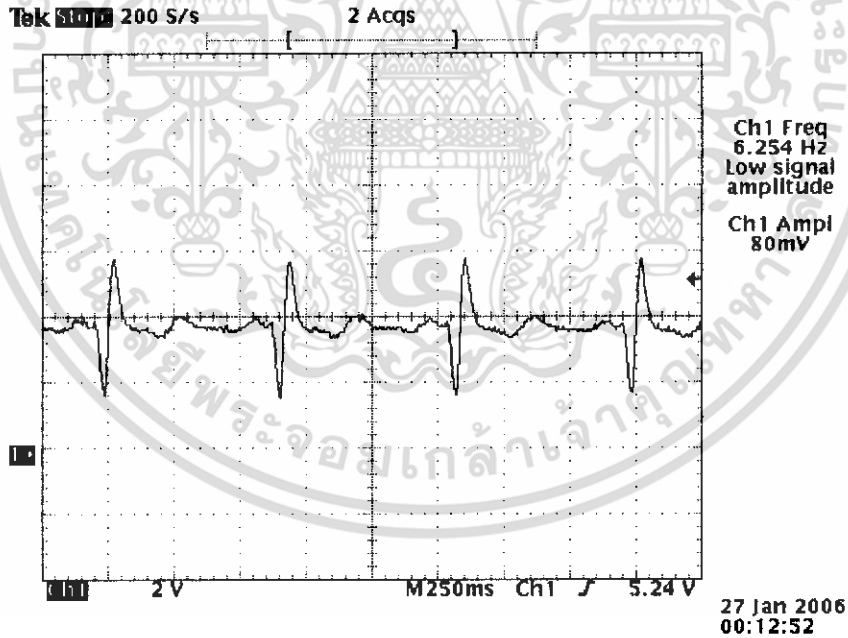
รูปที่ 4.3 กราฟแสดงอัตราขยายของวงจรดิฟเฟอเรนเชียลโหมด



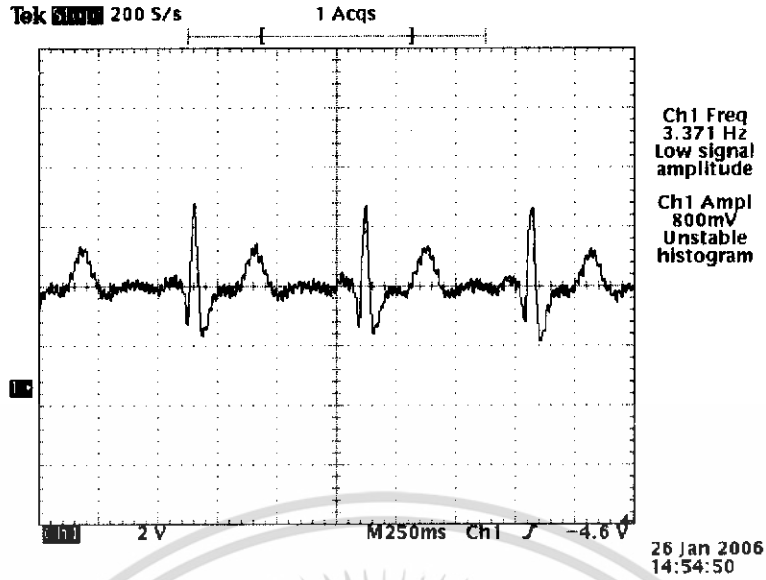
รูปที่ 4.4 กราฟแสดงอัตราขยายของวงจรขยายแบบคอมมอนโหมด



รูปที่ 4.5 กราฟแสดงค่า CMRR

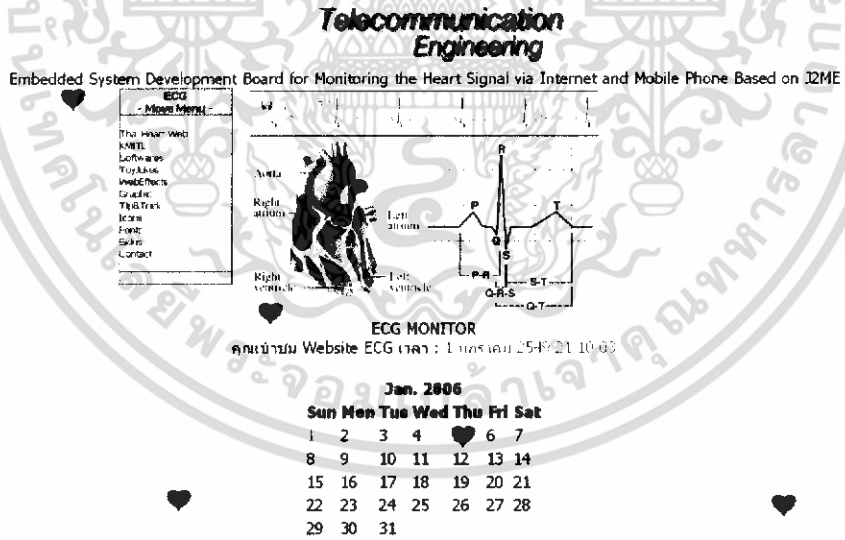


รูปที่ 4.6 รูปสัญญาณโดยใช้อีซีจี ซิมมูเลเตอร์

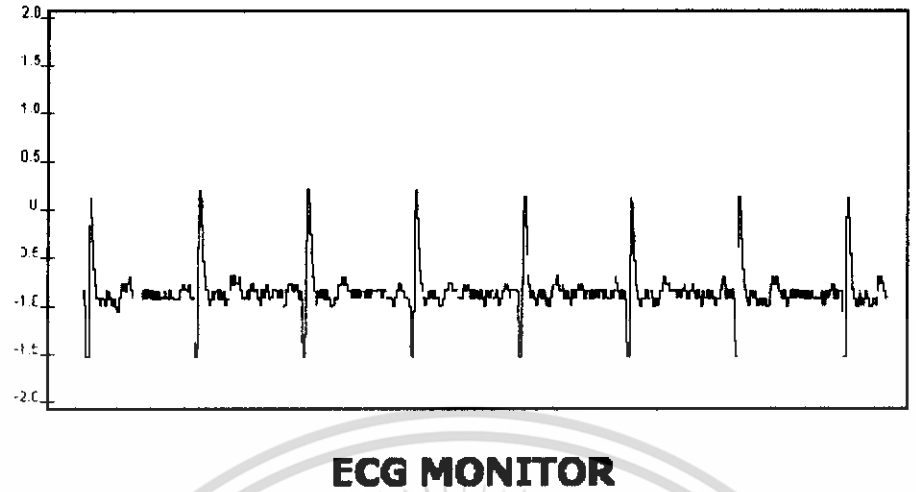


รูปที่ 4.7 รูปสัญญาณที่ได้จากการวัดจริง

4.2 ขั้นตอนการเข้าสู่เว็บเพจ



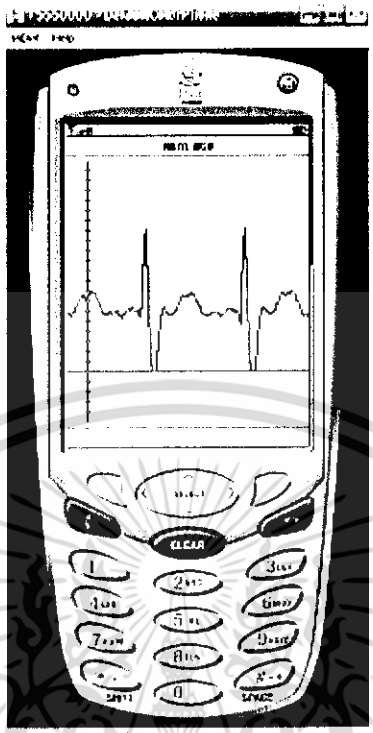
รูปที่ 4.8 หน้าหลักของเว็บ



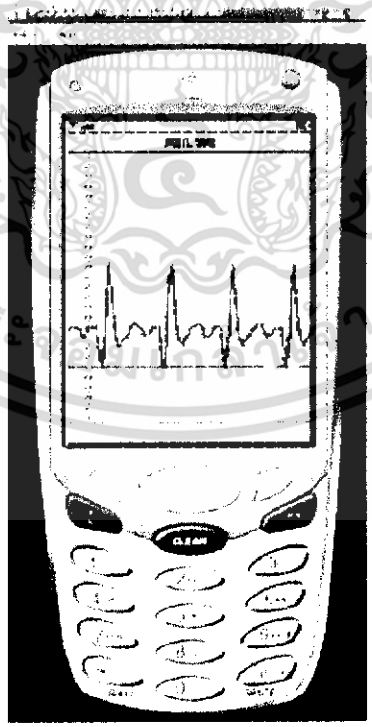
รูปที่ 4.9 สัญญาณที่ได้โดยใช้อีซีจี ซิมมูเลเตอร์

รูปที่ 4.10 สัญญาณที่ได้จากการวัดจริง

4.3 ขั้นตอนการเข้าสู่โทรศัพท์มือถือ



รูปที่ 4.11 การแสดงผลจากการใช้ซีจี ซิมมูเลเตอร์



รูปที่ 4.12 การแสดงผลจากการวัดจริง

บทที่ 5
บทสรุปและบทวิจารณ์

การทดลองวงจรขยายสัญญาณคลื่นไฟฟ้าหัวใจสามารถสรุป ออกเป็นส่วนต่าง ๆ ได้ดังนี้

- ส่วนที่ขยายสัญญาณคลื่นไฟฟ้าหัวใจ อินพุตจากอิเล็กโทรดเข้ามา 2 อินพุทผ่าน C คัปปลิ้งกระแสไฟตรงเข้า op - amp เบอร์ LM 324N (U1A, U1D) โดยมี Voltage divider เป็นตัวกำหนดค่าแรงดันอินพุตของ op - amp และมี R5 100K เป็นตัวปรับ Gain control ซึ่งมีอัตราขยายตั้งแต่ 100 - 10,000 (ในอุดมคติ) เอาท์พุต จาก op - amp U1B ผ่าน Low pass filter เพื่อกรองเฉพาะความถี่ต่ำผ่านเข้า op - amp U1C ซึ่งเป็น op - amp ที่ทำหน้าที่เป็น amplifier ขยายแรงดันออกทางเอาท์พุต
- วงจรแปลงอนาล็อกเป็นดิจิตอล ทำการแปลงสัญญาณคลื่นไฟฟ้าหัวใจ ซึ่งเป็นสัญญาณอนาลอกเป็นสัญญาณดิจิตอล ซึ่งเป็นเลขฐานสอง จำนวน 8 บิต แล้วนำไปประมวลผลต่อในไมโครคอนโทรลเลอร์ RCM2200

ปัญหาในการทดลอง

การปรับแต่งรูปคลื่นไฟฟ้าหัวใจทำได้ยากมาก เนื่องจาก noise ที่เข้ามารบกวนมากเกินไป จึงต้องเปลี่ยนอิเล็กโทรดใหม่ เนื่องจากอาจเป็นเพราะสารที่เคลือบในอิเล็กโทรดเสื่อม

อุปกรณ์ทางฮาร์ดแวร์มักจะมีปัญหาบ่อยๆ ทำให้การทดลองมีความล่าช้าทั้งทางด้านวงจรและโปรแกรม

ขาดทักษะและความชำนาญในเรื่องของการออกแบบและปฏิบัติงานรวมถึงความรู้ด้านอุปกรณ์การแพทย์ จึงทำให้เกิดความล่าช้าในการทำโครงการ

เอกสารอ้างอิง

[1] Z-World, " Dynamic C " [CD-ROM], Z-World, 1999.

[2] เรืองไกร รังสีพล, " เจาะระบบ TCP/IP ", บริษัท โปรวิชั่น จำกัด, 2001.

[3] นิรุธ อำนวยศิลป์, " สร้างเว็บเพจอย่างไร้ขีดจำกัด CGI&Perl ", บริษัท ซัคเซส มีเดีย จำกัด

[4] สัลยุทธ์ สว่างสุวรรณ, ศิวพงษ์ ตั้งสุจริต, " เครือข่ายคอมพิวเตอร์ " บริษัท เพียร์สัน เอ็ดดูเคชั่น อินโดไชน่า จำกัด, 2542.

[5] ทรงเกียรติ ภาวดี, " เก่ง J2ME ให้ครบสูตร ", บริษัท ซีเอ็ดยูเคชั่น จำกัด (มหาชน), 2546.

[6] กาญจนา ตันวิสุทธิ์, " เขียนเกมและโปรแกรมบนมือถือ J2ME ", บริษัท ไอดีซี อินโฟ ดิสทริบิวเตอร์ เซ็นเตอร์ จำกัด, 2547.

ภาคผนวก

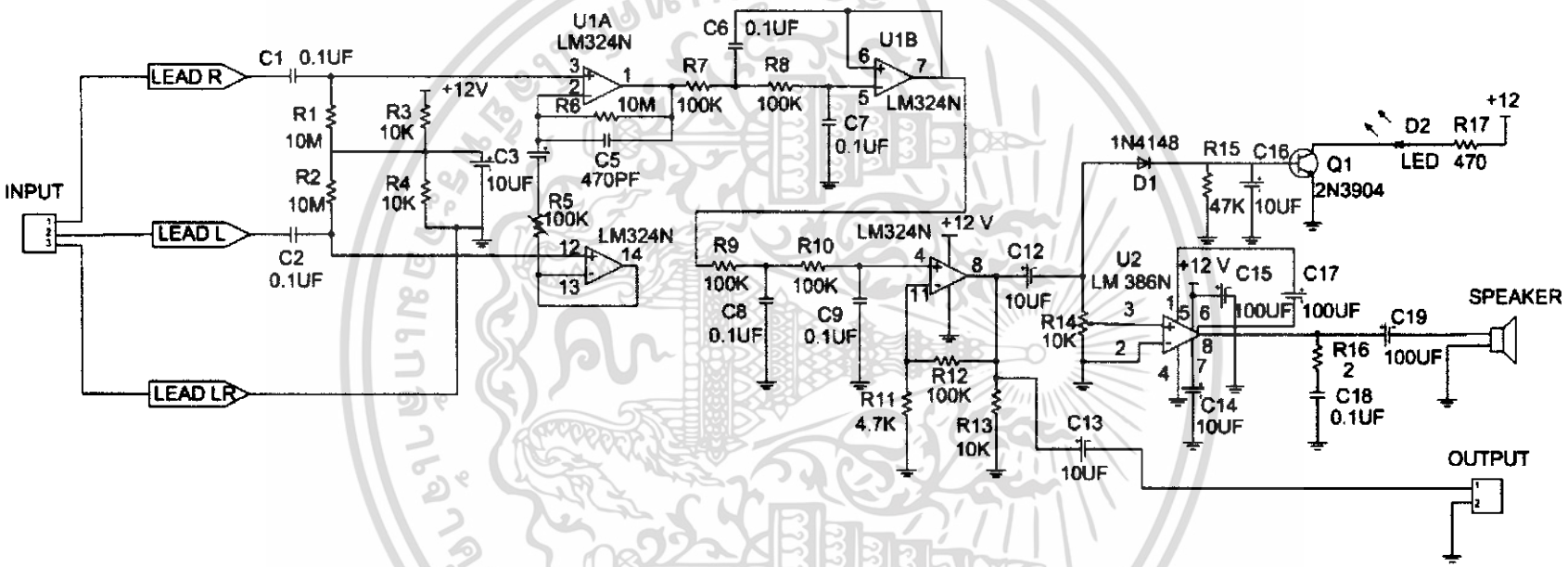
รูปแสดงอุปกรณ์ที่ใช้ในการทดลองเครื่องวัดคลื่นไฟฟ้าหัวใจ

ก) กล่องเครื่องวัดคลื่นไฟฟ้าหัวใจ

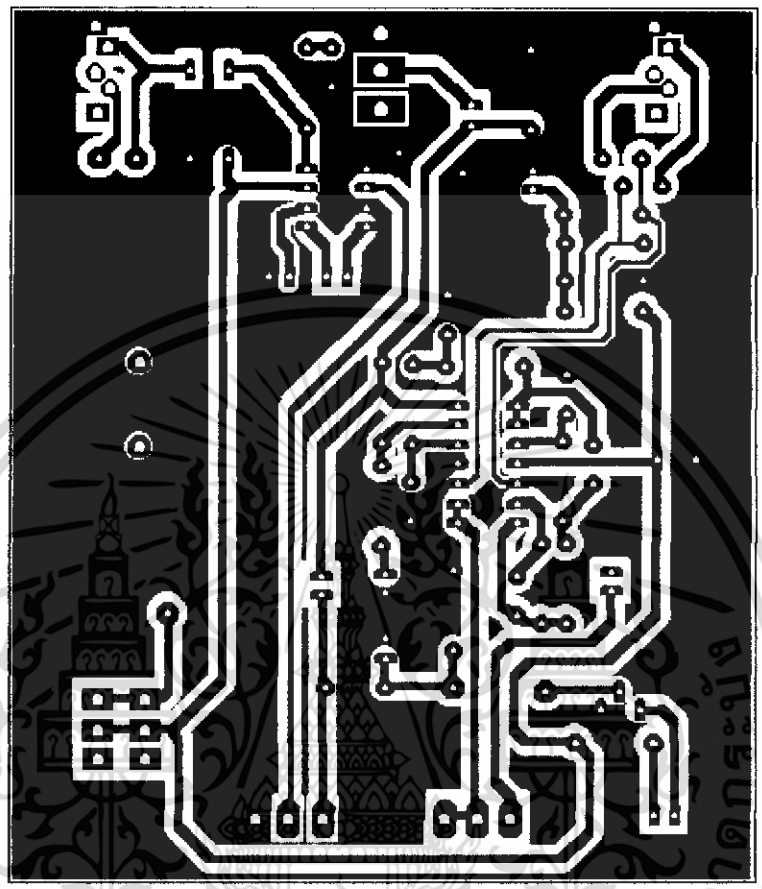
ข) เครื่องซิมมูเลเตอร์

ค) หัวเล็กโพรบและสายโพรบที่ใช้ในการทดลอง

ง) แสดงการวัดคลื่นไฟฟ้าหัวใจ



รูปวงจรเครื่องวัดคลื่นไฟฟ้าหัวใจ



รูปลายวงจรเครื่องวัดคลื่นไฟฟ้าหัวใจ

โปรแกรมไดนามิกซี

```
#define MY_IP_ADDRESS "161.246.18.9"
#define MY_NETMASK    "255.255.255.0"
#define MY_GATEWAY    "161.246.18.1"
#define HTTP_MAXSERVERS 1
#define MAX_TCP_SOCKET_BUFFERS 1
#define REDIRECTHOST    MY_IP_ADDRESS
#memmap xmem
#use "dcrtcp.lib"
#use "http.lib"
#define REDIRECTTO   "http://" REDIRECTHOST "/index.shtml"

#ximport "G:\Project ECG/firstpag.html"  index_html
#ximport "G:\Project ECG/kmitlECG.class" kmitlECG_class
#ximport "G:\Project ECG/logo.gif" logo_gif
#ximport "G:\Project ECG/heart.gif" heart_gif
#ximport "G:\Project ECG/ecga.jpeg" ecga_jpeg
#ximport "G:\Project ECG/pagECG.html"  pagECG_html

const HttpType http_types[] =
{
   { ".html", "text/html", shtml_handler},
   { ".html", "text/html", NULL},
   { ".cgi",  "  ",        NULL},
   { ".gif",  "image/gif", NULL},
   { ".jpeg", "image/jpeg", NULL}
};

int  A;
int  B;

const HttpSpec http_flashspec[] =
{
{HTTPSPEC_FILE,  "/", index_html, NULL, 0, NULL, NULL},
{HTTPSPEC_FILE,  "/index.html", index_html, NULL, 0, NULL,NULL},
{HTTPSPEC_FILE,   "/kmitlECG.class",   kmitlECG_class,   NULL,   0,   NULL,
NULL},
{HTTPSPEC_FILE,  "/logo.gif",  logo_gif,       NULL, 0, NULL, NULL},
{HTTPSPEC_FILE,  "/heart.gif", heart_gif,      NULL, 0, NULL, NULL},
{HTTPSPEC_FILE,  "/ecga.jpeg", ecga_jpeg ,     NULL, 0, NULL, NULL},
{HTTPSPEC_FILE,  "/pagECG.html",  pagECG_html,  NULL, 0, NULL,NULL},
{HTTPSPEC_VARIABLE, "A", 0, &A,  INT8,  "%d", NULL},
{HTTPSPEC_VARIABLE, "B", 0, &B,  INT8,  "%d", NULL}
};

/*
  void delay()
{
   int   i;
   i=0;
   while (i!=1){
      costate {
         waitfor(DelayMs(1));
```

```
 //until 200 ms have passed since
       i++;
       }
    }
}


*/
 void UpdateADC();
main()
{
    A = 0;
    B = 0;
    sock_init();
    http_init();
    tcp_reserveport(80);

    WrPortI (SPCR,&SPCRShadow,0x80);

    while (1)
       {
       UpdateADC();

        http_handler();
        }
       }

void UpdateADC()
{
    A = RdPortI(PADR);
    B =A;
 // B =(-A*3)/2;
 //delay();
}
```

โปรแกรมในส่วนจาวา

```
import java.awt.*;
import java.applet.*;

public class kmitlECG extends Applet implements Runnable
{
    int i = 0, x = 60, j;
    int y[] = new int[670];
      int y670;
    public void init()
      {
            y[0] = 320-Integer.parseInt(getParameter("a0"));
          y[1] = 320-Integer.parseInt(getParameter("a1"));
            y[2] = 320-Integer.parseInt(getParameter("a2"));
            y[3] = 320-Integer.parseInt(getParameter("a3"));
            y[4] = 320-Integer.parseInt(getParameter("a4"));
            y[5] = 320-Integer.parseInt(getParameter("a5"));
            y[6] = 320-Integer.parseInt(getParameter("a6"));
          y[7] = 320-Integer.parseInt(getParameter("a7"));
            y[8] = 320-Integer.parseInt(getParameter("a8"));
            y[9] = 320-Integer.parseInt(getParameter("a9"));
            y[10] = 320-Integer.parseInt(getParameter("a10"));
            y[11] = 320-Integer.parseInt(getParameter("a11"));
            y[12] = 320-Integer.parseInt(getParameter("a12"));
            y[13] = 320-Integer.parseInt(getParameter("a13"));
            y[14] = 320-Integer.parseInt(getParameter("a14"));
            y[15] = 320-Integer.parseInt(getParameter("a15"));
            y[16] = 320-Integer.parseInt(getParameter("a16"));
          y[17] = 320-Integer.parseInt(getParameter("a17"));
            y[18] = 320-Integer.parseInt(getParameter("a18"));
            y[19] = 320-Integer.parseInt(getParameter("a19"));
            y[20] = 320-Integer.parseInt(getParameter("a20"));
            y[21] = 320-Integer.parseInt(getParameter("a21"));
            y[22] = 320-Integer.parseInt(getParameter("a22"));
            y[23] = 320-Integer.parseInt(getParameter("a23"));
            y[24] = 320-Integer.parseInt(getParameter("a24"));
            y[25] = 320-Integer.parseInt(getParameter("a25"));
            y[26] = 320-Integer.parseInt(getParameter("a26"));
          y[27] = 320-Integer.parseInt(getParameter("a27"));
            y[28] = 320-Integer.parseInt(getParameter("a28"));
            y[29] = 320-Integer.parseInt(getParameter("a29"));
            y[30] = 320-Integer.parseInt(getParameter("a30"));
            y[31] = 320-Integer.parseInt(getParameter("a31"));
            y[32] = 320-Integer.parseInt(getParameter("a32"));
            y[33] = 320-Integer.parseInt(getParameter("a33"));
            y[34] = 320-Integer.parseInt(getParameter("a34"));
            y[35] = 320-Integer.parseInt(getParameter("a35"));
            y[36] = 320-Integer.parseInt(getParameter("a36"));
          y[37] = 320-Integer.parseInt(getParameter("a37"));
            y[38] = 320-Integer.parseInt(getParameter("a38"));
            y[39] = 320-Integer.parseInt(getParameter("a39"));
            y[40] = 320-Integer.parseInt(getParameter("a40"));
            y[41] = 320-Integer.parseInt(getParameter("a41"));
            y[42] = 320-Integer.parseInt(getParameter("a42"));
```

```
  y[43] = 320-Integer.parseInt(getParameter("a43"));
  y[44] = 320-Integer.parseInt(getParameter("a44"));
  y[45] = 320-Integer.parseInt(getParameter("a45"));
  y[46] = 320-Integer.parseInt(getParameter("a46"));
y[47] = 320-Integer.parseInt(getParameter("a47"));
  y[48] = 320-Integer.parseInt(getParameter("a48"));
  y[49] = 320-Integer.parseInt(getParameter("a49"));
  y[50] = 320-Integer.parseInt(getParameter("a50"));
  y[51] = 320-Integer.parseInt(getParameter("a51"));
  y[52] = 320-Integer.parseInt(getParameter("a52"));
  y[53] = 320-Integer.parseInt(getParameter("a53"));
  y[54] = 320-Integer.parseInt(getParameter("a54"));
  y[55] = 320-Integer.parseInt(getParameter("a55"));
  y[56] = 320-Integer.parseInt(getParameter("a56"));
y[57] = 320-Integer.parseInt(getParameter("a57"));
  y[58] = 320-Integer.parseInt(getParameter("a58"));
  y[59] = 320-Integer.parseInt(getParameter("a59"));
  y[60] = 320-Integer.parseInt(getParameter("a60"));
  y[61] = 320-Integer.parseInt(getParameter("a61"));
  y[62] = 320-Integer.parseInt(getParameter("a62"));
  y[63] = 320-Integer.parseInt(getParameter("a63"));
  y[64] = 320-Integer.parseInt(getParameter("a64"));
  y[65] = 320-Integer.parseInt(getParameter("a65"));
  y[66] = 320-Integer.parseInt(getParameter("a66"));
y[67] = 320-Integer.parseInt(getParameter("a67"));
  y[68] = 320-Integer.parseInt(getParameter("a68"));
  y[69] = 320-Integer.parseInt(getParameter("a69"));
  y[70] = 320-Integer.parseInt(getParameter("a70"));
  y[71] = 320-Integer.parseInt(getParameter("a71"));
  y[72] = 320-Integer.parseInt(getParameter("a72"));
  y[73] = 320-Integer.parseInt(getParameter("a73"));
  y[74] = 320-Integer.parseInt(getParameter("a74"));
  y[75] = 320-Integer.parseInt(getParameter("a75"));
  y[76] = 320-Integer.parseInt(getParameter("a76"));
y[77] = 320-Integer.parseInt(getParameter("a77"));
  y[78] = 320-Integer.parseInt(getParameter("a78"));
  y[79] = 320-Integer.parseInt(getParameter("a79"));
  y[80] = 320-Integer.parseInt(getParameter("a80"));
  y[81] = 320-Integer.parseInt(getParameter("a81"));
  y[82] = 320-Integer.parseInt(getParameter("a82"));
  y[83] = 320-Integer.parseInt(getParameter("a83"));
  y[84] = 320-Integer.parseInt(getParameter("a84"));
  y[85] = 320-Integer.parseInt(getParameter("a85"));
  y[86] = 320-Integer.parseInt(getParameter("a86"));
y[87] = 320-Integer.parseInt(getParameter("a87"));
  y[88] = 320-Integer.parseInt(getParameter("a88"));
  y[89] = 320-Integer.parseInt(getParameter("a89"));
  y[90] = 320-Integer.parseInt(getParameter("a90"));
  y[91] = 320-Integer.parseInt(getParameter("a91"));
  y[92] = 320-Integer.parseInt(getParameter("a92"));
  y[93] = 320-Integer.parseInt(getParameter("a93"));
  y[94] = 320-Integer.parseInt(getParameter("a94"));
  y[95] = 320-Integer.parseInt(getParameter("a95"));
  y[96] = 320-Integer.parseInt(getParameter("a96"));
y[97] = 320-Integer.parseInt(getParameter("a97"));
  y[98] = 320-Integer.parseInt(getParameter("a98"));
  y[99] = 320-Integer.parseInt(getParameter("a99"));
```

```
  y[100] = 320-Integer.parseInt(getParameter("a100"));
  y[101] = 320-Integer.parseInt(getParameter("a101"));
  y[102] = 320-Integer.parseInt(getParameter("a102"));
  y[103] = 320-Integer.parseInt(getParameter("a103"));
  y[104] = 320-Integer.parseInt(getParameter("a104"));
  y[105] = 320-Integer.parseInt(getParameter("a105"));
  y[106] = 320-Integer.parseInt(getParameter("a106"));
y[107] = 320-Integer.parseInt(getParameter("a107"));
  y[108] = 320-Integer.parseInt(getParameter("a108"));
  y[109] = 320-Integer.parseInt(getParameter("a109"));
  y[110] = 320-Integer.parseInt(getParameter("a110"));
  y[111] = 320-Integer.parseInt(getParameter("a111"));
  y[112] = 320-Integer.parseInt(getParameter("a112"));
  y[113] = 320-Integer.parseInt(getParameter("a113"));
  y[114] = 320-Integer.parseInt(getParameter("a114"));
  y[115] = 320-Integer.parseInt(getParameter("a115"));
  y[116] = 320-Integer.parseInt(getParameter("a116"));
y[117] = 320-Integer.parseInt(getParameter("a117"));
  y[118] = 320-Integer.parseInt(getParameter("a118"));
  y[119] = 320-Integer.parseInt(getParameter("a119"));
  y[120] = 320-Integer.parseInt(getParameter("a120"));
  y[121] = 320-Integer.parseInt(getParameter("a121"));
  y[122] = 320-Integer.parseInt(getParameter("a122"));
  y[123] = 320-Integer.parseInt(getParameter("a123"));
  y[124] = 320-Integer.parseInt(getParameter("a124"));
  y[125] = 320-Integer.parseInt(getParameter("a125"));
  y[126] = 320-Integer.parseInt(getParameter("a126"));
y[127] = 320-Integer.parseInt(getParameter("a127"));
  y[128] = 320-Integer.parseInt(getParameter("a128"));
  y[129] = 320-Integer.parseInt(getParameter("a129"));
  y[130] = 320-Integer.parseInt(getParameter("a130"));
  y[131] = 320-Integer.parseInt(getParameter("a131"));
  y[132] = 320-Integer.parseInt(getParameter("a132"));
  y[133] = 320-Integer.parseInt(getParameter("a133"));
  y[134] = 320-Integer.parseInt(getParameter("a134"));
  y[135] = 320-Integer.parseInt(getParameter("a135"));
  y[136] = 320-Integer.parseInt(getParameter("a136"));
y[137] = 320-Integer.parseInt(getParameter("a137"));
  y[138] = 320-Integer.parseInt(getParameter("a138"));
  y[139] = 320-Integer.parseInt(getParameter("a139"));
  y[140] = 320-Integer.parseInt(getParameter("a140"));
  y[141] = 320-Integer.parseInt(getParameter("a141"));
  y[142] = 320-Integer.parseInt(getParameter("a142"));
  y[143] = 320-Integer.parseInt(getParameter("a143"));
  y[144] = 320-Integer.parseInt(getParameter("a144"));
  y[145] = 320-Integer.parseInt(getParameter("a145"));
  y[146] = 320-Integer.parseInt(getParameter("a146"));
y[147] = 320-Integer.parseInt(getParameter("a147"));
  y[148] = 320-Integer.parseInt(getParameter("a148"));
  y[149] = 320-Integer.parseInt(getParameter("a149"));
  y[150] = 320-Integer.parseInt(getParameter("a150"));
  y[151] = 320-Integer.parseInt(getParameter("a151"));
  y[152] = 320-Integer.parseInt(getParameter("a152"));
  y[153] = 320-Integer.parseInt(getParameter("a153"));
  y[154] = 320-Integer.parseInt(getParameter("a154"));
  y[155] = 320-Integer.parseInt(getParameter("a155"));
  y[156] = 320-Integer.parseInt(getParameter("a156"));
```

```
y[157] = 320-Integer.parseInt(getParameter("a157"));
  y[158] = 320-Integer.parseInt(getParameter("a158"));
  y[159] = 320-Integer.parseInt(getParameter("a159"));
  y[160] = 320-Integer.parseInt(getParameter("a160"));
  y[161] = 320-Integer.parseInt(getParameter("a161"));
  y[162] = 320-Integer.parseInt(getParameter("a162"));
  y[163] = 320-Integer.parseInt(getParameter("a163"));
  y[164] = 320-Integer.parseInt(getParameter("a164"));
  y[165] = 320-Integer.parseInt(getParameter("a165"));
  y[166] = 320-Integer.parseInt(getParameter("a166"));
y[167] = 320-Integer.parseInt(getParameter("a167"));
  y[168] = 320-Integer.parseInt(getParameter("a168"));
  y[169] = 320-Integer.parseInt(getParameter("a169"));
  y[170] = 320-Integer.parseInt(getParameter("a170"));
  y[171] = 320-Integer.parseInt(getParameter("a171"));
  y[172] = 320-Integer.parseInt(getParameter("a172"));
  y[173] = 320-Integer.parseInt(getParameter("a173"));
  y[174] = 320-Integer.parseInt(getParameter("a174"));
  y[175] = 320-Integer.parseInt(getParameter("a175"));
  y[176] = 320-Integer.parseInt(getParameter("a176"));
y[177] = 320-Integer.parseInt(getParameter("a177"));
  y[178] = 320-Integer.parseInt(getParameter("a178"));
  y[179] = 320-Integer.parseInt(getParameter("a179"));
  y[180] = 320-Integer.parseInt(getParameter("a180"));
  y[181] = 320-Integer.parseInt(getParameter("a181"));
  y[182] = 320-Integer.parseInt(getParameter("a182"));
  y[183] = 320-Integer.parseInt(getParameter("a183"));
  y[184] = 320-Integer.parseInt(getParameter("a184"));
  y[185] = 320-Integer.parseInt(getParameter("a185"));
  y[186] = 320-Integer.parseInt(getParameter("a186"));
y[187] = 320-Integer.parseInt(getParameter("a187"));
  y[188] = 320-Integer.parseInt(getParameter("a188"));
  y[189] = 320-Integer.parseInt(getParameter("a189"));
  y[190] = 320-Integer.parseInt(getParameter("a190"));
  y[191] = 320-Integer.parseInt(getParameter("a191"));
  y[192] = 320-Integer.parseInt(getParameter("a192"));
  y[193] = 320-Integer.parseInt(getParameter("a193"));
  y[194] = 320-Integer.parseInt(getParameter("a194"));
  y[195] = 320-Integer.parseInt(getParameter("a195"));
  y[196] = 320-Integer.parseInt(getParameter("a196"));
y[197] = 320-Integer.parseInt(getParameter("a197"));
  y[198] = 320-Integer.parseInt(getParameter("a198"));
  y[199] = 320-Integer.parseInt(getParameter("a199"));
  y[200] = 320-Integer.parseInt(getParameter("a200"));
  y[201] = 320-Integer.parseInt(getParameter("a201"));
  y[202] = 320-Integer.parseInt(getParameter("a202"));
  y[203] = 320-Integer.parseInt(getParameter("a203"));
  y[204] = 320-Integer.parseInt(getParameter("a204"));
  y[205] = 320-Integer.parseInt(getParameter("a205"));
  y[206] = 320-Integer.parseInt(getParameter("a206"));
y[207] = 320-Integer.parseInt(getParameter("a207"));
  y[208] = 320-Integer.parseInt(getParameter("a208"));
  y[209] = 320-Integer.parseInt(getParameter("a209"));
  y[210] = 320-Integer.parseInt(getParameter("a210"));
  y[211] = 320-Integer.parseInt(getParameter("a211"));
  y[212] = 320-Integer.parseInt(getParameter("a212"));
  y[213] = 320-Integer.parseInt(getParameter("a213"));
```

```
  y[214] = 320-Integer.parseInt(getParameter("a214"));
  y[215] = 320-Integer.parseInt(getParameter("a215"));
  y[216] = 320-Integer.parseInt(getParameter("a216"));
y[217] = 320-Integer.parseInt(getParameter("a217"));
  y[218] = 320-Integer.parseInt(getParameter("a218"));
  y[219] = 320-Integer.parseInt(getParameter("a219"));
  y[220] = 320-Integer.parseInt(getParameter("a220"));
  y[221] = 320-Integer.parseInt(getParameter("a221"));
  y[222] = 320-Integer.parseInt(getParameter("a222"));
  y[223] = 320-Integer.parseInt(getParameter("a223"));
  y[224] = 320-Integer.parseInt(getParameter("a224"));
  y[225] = 320-Integer.parseInt(getParameter("a225"));
  y[226] = 320-Integer.parseInt(getParameter("a226"));
y[227] = 320-Integer.parseInt(getParameter("a227"));
  y[228] = 320-Integer.parseInt(getParameter("a228"));
  y[229] = 320-Integer.parseInt(getParameter("a229"));
  y[230] = 320-Integer.parseInt(getParameter("a230"));
  y[231] = 320-Integer.parseInt(getParameter("a231"));
  y[232] = 320-Integer.parseInt(getParameter("a232"));
  y[233] = 320-Integer.parseInt(getParameter("a233"));
  y[234] = 320-Integer.parseInt(getParameter("a234"));
  y[235] = 320-Integer.parseInt(getParameter("a235"));
  y[236] = 320-Integer.parseInt(getParameter("a236"));
y[237] = 320-Integer.parseInt(getParameter("a237"));
  y[238] = 320-Integer.parseInt(getParameter("a238"));
  y[239] = 320-Integer.parseInt(getParameter("a239"));
  y[240] = 320-Integer.parseInt(getParameter("a240"));
  y[241] = 320-Integer.parseInt(getParameter("a241"));
  y[242] = 320-Integer.parseInt(getParameter("a242"));
  y[243] = 320-Integer.parseInt(getParameter("a243"));
  y[244] = 320-Integer.parseInt(getParameter("a244"));
  y[245] = 320-Integer.parseInt(getParameter("a245"));
  y[246] = 320-Integer.parseInt(getParameter("a246"));
y[247] = 320-Integer.parseInt(getParameter("a247"));
  y[248] = 320-Integer.parseInt(getParameter("a248"));
  y[249] = 320-Integer.parseInt(getParameter("a249"));
  y[250] = 320-Integer.parseInt(getParameter("a250"));
  y[251] = 320-Integer.parseInt(getParameter("a251"));
  y[252] = 320-Integer.parseInt(getParameter("a252"));
  y[253] = 320-Integer.parseInt(getParameter("a253"));
  y[254] = 320-Integer.parseInt(getParameter("a254"));
  y[255] = 320-Integer.parseInt(getParameter("a255"));
  y[256] = 320-Integer.parseInt(getParameter("a256"));
y[257] = 320-Integer.parseInt(getParameter("a257"));
  y[258] = 320-Integer.parseInt(getParameter("a258"));
  y[259] = 320-Integer.parseInt(getParameter("a259"));
  y[260] = 320-Integer.parseInt(getParameter("a260"));
  y[261] = 320-Integer.parseInt(getParameter("a261"));
  y[262] = 320-Integer.parseInt(getParameter("a262"));
  y[263] = 320-Integer.parseInt(getParameter("a263"));
  y[264] = 320-Integer.parseInt(getParameter("a264"));
  y[265] = 320-Integer.parseInt(getParameter("a265"));
  y[266] = 320-Integer.parseInt(getParameter("a266"));
y[267] = 320-Integer.parseInt(getParameter("a267"));
  y[268] = 320-Integer.parseInt(getParameter("a268"));
  y[269] = 320-Integer.parseInt(getParameter("a269"));
  y[270] = 320-Integer.parseInt(getParameter("a270"));
```

```
  y[271] = 320-Integer.parseInt(getParameter("a271"));
  y[272] = 320-Integer.parseInt(getParameter("a272"));
  y[273] = 320-Integer.parseInt(getParameter("a273"));
  y[274] = 320-Integer.parseInt(getParameter("a274"));
  y[275] = 320-Integer.parseInt(getParameter("a275"));
  y[276] = 320-Integer.parseInt(getParameter("a276"));
y[277] = 320-Integer.parseInt(getParameter("a277"));
  y[278] = 320-Integer.parseInt(getParameter("a278"));
  y[279] = 320-Integer.parseInt(getParameter("a279"));
  y[280] = 320-Integer.parseInt(getParameter("a280"));
  y[281] = 320-Integer.parseInt(getParameter("a281"));
  y[282] = 320-Integer.parseInt(getParameter("a282"));
  y[283] = 320-Integer.parseInt(getParameter("a283"));
  y[284] = 320-Integer.parseInt(getParameter("a284"));
  y[285] = 320-Integer.parseInt(getParameter("a285"));
  y[286] = 320-Integer.parseInt(getParameter("a286"));
y[287] = 320-Integer.parseInt(getParameter("a287"));
  y[288] = 320-Integer.parseInt(getParameter("a288"));
  y[289] = 320-Integer.parseInt(getParameter("a289"));
  y[290] = 320-Integer.parseInt(getParameter("a290"));
  y[291] = 320-Integer.parseInt(getParameter("a291"));
  y[292] = 320-Integer.parseInt(getParameter("a292"));
  y[293] = 320-Integer.parseInt(getParameter("a293"));
  y[294] = 320-Integer.parseInt(getParameter("a294"));
  y[295] = 320-Integer.parseInt(getParameter("a295"));
  y[296] = 320-Integer.parseInt(getParameter("a296"));
y[297] = 320-Integer.parseInt(getParameter("a297"));
  y[298] = 320-Integer.parseInt(getParameter("a298"));
  y[299] = 320-Integer.parseInt(getParameter("a299"));
  y[300] = 320-Integer.parseInt(getParameter("a300"));
  y[301] = 320-Integer.parseInt(getParameter("a301"));
  y[302] = 320-Integer.parseInt(getParameter("a302"));
  y[303] = 320-Integer.parseInt(getParameter("a303"));
  y[304] = 320-Integer.parseInt(getParameter("a304"));
  y[305] = 320-Integer.parseInt(getParameter("a305"));
  y[306] = 320-Integer.parseInt(getParameter("a306"));
y[307] = 320-Integer.parseInt(getParameter("a307"));
  y[308] = 320-Integer.parseInt(getParameter("a308"));
  y[309] = 320-Integer.parseInt(getParameter("a309"));
  y[310] = 320-Integer.parseInt(getParameter("a310"));
  y[311] = 320-Integer.parseInt(getParameter("a311"));
  y[312] = 320-Integer.parseInt(getParameter("a312"));
  y[313] = 320-Integer.parseInt(getParameter("a313"));
  y[314] = 320-Integer.parseInt(getParameter("a314"));
  y[315] = 320-Integer.parseInt(getParameter("a315"));
  y[316] = 320-Integer.parseInt(getParameter("a316"));
y[317] = 320-Integer.parseInt(getParameter("a317"));
  y[318] = 320-Integer.parseInt(getParameter("a318"));
  y[319] = 320-Integer.parseInt(getParameter("a319"));
  y[320] = 320-Integer.parseInt(getParameter("a320"));
  y[321] = 320-Integer.parseInt(getParameter("a321"));
  y[322] = 320-Integer.parseInt(getParameter("a322"));
  y[323] = 320-Integer.parseInt(getParameter("a323"));
  y[324] = 320-Integer.parseInt(getParameter("a324"));
  y[325] = 320-Integer.parseInt(getParameter("a325"));
  y[326] = 320-Integer.parseInt(getParameter("a326"));
y[327] = 320-Integer.parseInt(getParameter("a327"));
```

```
  y[328] = 320-Integer.parseInt(getParameter("a328"));
  y[329] = 320-Integer.parseInt(getParameter("a329"));
  y[330] = 320-Integer.parseInt(getParameter("a330"));
  y[331] = 320-Integer.parseInt(getParameter("a331"));
  y[332] = 320-Integer.parseInt(getParameter("a332"));
  y[333] = 320-Integer.parseInt(getParameter("a333"));
  y[334] = 320-Integer.parseInt(getParameter("a334"));
  y[335] = 320-Integer.parseInt(getParameter("a335"));
  y[336] = 320-Integer.parseInt(getParameter("a336"));
y[337] = 320-Integer.parseInt(getParameter("a337"));
  y[338] = 320-Integer.parseInt(getParameter("a338"));
  y[339] = 320-Integer.parseInt(getParameter("a339"));
  y[340] = 320-Integer.parseInt(getParameter("a340"));
  y[341] = 320-Integer.parseInt(getParameter("a341"));
  y[342] = 320-Integer.parseInt(getParameter("a342"));
  y[343] = 320-Integer.parseInt(getParameter("a343"));
  y[344] = 320-Integer.parseInt(getParameter("a344"));
  y[345] = 320-Integer.parseInt(getParameter("a345"));
  y[346] = 320-Integer.parseInt(getParameter("a346"));
y[347] = 320-Integer.parseInt(getParameter("a347"));
  y[348] = 320-Integer.parseInt(getParameter("a348"));
  y[349] = 320-Integer.parseInt(getParameter("a349"));
  y[350] = 320-Integer.parseInt(getParameter("a350"));
  y[351] = 320-Integer.parseInt(getParameter("a351"));
  y[352] = 320-Integer.parseInt(getParameter("a352"));
  y[353] = 320-Integer.parseInt(getParameter("a353"));
  y[354] = 320-Integer.parseInt(getParameter("a354"));
  y[355] = 320-Integer.parseInt(getParameter("a355"));
  y[356] = 320-Integer.parseInt(getParameter("a356"));
y[357] = 320-Integer.parseInt(getParameter("a357"));
  y[358] = 320-Integer.parseInt(getParameter("a358"));
  y[359] = 320-Integer.parseInt(getParameter("a359"));
  y[360] = 320-Integer.parseInt(getParameter("a360"));
  y[361] = 320-Integer.parseInt(getParameter("a361"));
  y[362] = 320-Integer.parseInt(getParameter("a362"));
  y[363] = 320-Integer.parseInt(getParameter("a363"));
  y[364] = 320-Integer.parseInt(getParameter("a364"));
  y[365] = 320-Integer.parseInt(getParameter("a365"));
  y[366] = 320-Integer.parseInt(getParameter("a366"));
y[367] = 320-Integer.parseInt(getParameter("a367"));
  y[368] = 320-Integer.parseInt(getParameter("a368"));
  y[369] = 320-Integer.parseInt(getParameter("a369"));
  y[370] = 320-Integer.parseInt(getParameter("a370"));
  y[371] = 320-Integer.parseInt(getParameter("a371"));
  y[372] = 320-Integer.parseInt(getParameter("a372"));
  y[373] = 320-Integer.parseInt(getParameter("a373"));
  y[374] = 320-Integer.parseInt(getParameter("a374"));
  y[375] = 320-Integer.parseInt(getParameter("a375"));
  y[376] = 320-Integer.parseInt(getParameter("a376"));
y[377] = 320-Integer.parseInt(getParameter("a377"));
  y[378] = 320-Integer.parseInt(getParameter("a378"));
  y[379] = 320-Integer.parseInt(getParameter("a379"));
  y[380] = 320-Integer.parseInt(getParameter("a380"));
  y[381] = 320-Integer.parseInt(getParameter("a381"));
  y[382] = 320-Integer.parseInt(getParameter("a382"));
  y[383] = 320-Integer.parseInt(getParameter("a383"));
  y[384] = 320-Integer.parseInt(getParameter("a384"));
```

```
  y[385] = 320-Integer.parseInt(getParameter("a385"));
  y[386] = 320-Integer.parseInt(getParameter("a386"));
y[387] = 320-Integer.parseInt(getParameter("a387"));
  y[388] = 320-Integer.parseInt(getParameter("a388"));
  y[389] = 320-Integer.parseInt(getParameter("a389"));
  y[390] = 320-Integer.parseInt(getParameter("a390"));
  y[391] = 320-Integer.parseInt(getParameter("a391"));
  y[392] = 320-Integer.parseInt(getParameter("a392"));
  y[393] = 320-Integer.parseInt(getParameter("a393"));
  y[394] = 320-Integer.parseInt(getParameter("a394"));
  y[395] = 320-Integer.parseInt(getParameter("a395"));
  y[396] = 320-Integer.parseInt(getParameter("a396"));
y[397] = 320-Integer.parseInt(getParameter("a397"));
  y[398] = 320-Integer.parseInt(getParameter("a398"));
  y[399] = 320-Integer.parseInt(getParameter("a399"));
  y[400] = 320-Integer.parseInt(getParameter("a400"));
  y[401] = 320-Integer.parseInt(getParameter("a401"));
  y[402] = 320-Integer.parseInt(getParameter("a402"));
  y[403] = 320-Integer.parseInt(getParameter("a403"));
  y[404] = 320-Integer.parseInt(getParameter("a404"));
  y[405] = 320-Integer.parseInt(getParameter("a405"));
  y[406] = 320-Integer.parseInt(getParameter("a406"));
y[407] = 320-Integer.parseInt(getParameter("a407"));
  y[408] = 320-Integer.parseInt(getParameter("a408"));
  y[409] = 320-Integer.parseInt(getParameter("a409"));
  y[410] = 320-Integer.parseInt(getParameter("a410"));
  y[411] = 320-Integer.parseInt(getParameter("a411"));
  y[412] = 320-Integer.parseInt(getParameter("a412"));
  y[413] = 320-Integer.parseInt(getParameter("a413"));
  y[414] = 320-Integer.parseInt(getParameter("a414"));
  y[415] = 320-Integer.parseInt(getParameter("a415"));
  y[416] = 320-Integer.parseInt(getParameter("a416"));
y[417] = 320-Integer.parseInt(getParameter("a417"));
  y[418] = 320-Integer.parseInt(getParameter("a418"));
  y[419] = 320-Integer.parseInt(getParameter("a419"));
  y[420] = 320-Integer.parseInt(getParameter("a420"));
  y[421] = 320-Integer.parseInt(getParameter("a421"));
  y[422] = 320-Integer.parseInt(getParameter("a422"));
  y[423] = 320-Integer.parseInt(getParameter("a423"));
  y[424] = 320-Integer.parseInt(getParameter("a424"));
  y[425] = 320-Integer.parseInt(getParameter("a425"));
  y[426] = 320-Integer.parseInt(getParameter("a426"));
y[427] = 320-Integer.parseInt(getParameter("a427"));
  y[428] = 320-Integer.parseInt(getParameter("a428"));
  y[429] = 320-Integer.parseInt(getParameter("a429"));
  y[430] = 320-Integer.parseInt(getParameter("a430"));
  y[431] = 320-Integer.parseInt(getParameter("a431"));
  y[432] = 320-Integer.parseInt(getParameter("a432"));
  y[433] = 320-Integer.parseInt(getParameter("a433"));
  y[434] = 320-Integer.parseInt(getParameter("a434"));
  y[435] = 320-Integer.parseInt(getParameter("a435"));
  y[436] = 320-Integer.parseInt(getParameter("a436"));
y[437] = 320-Integer.parseInt(getParameter("a437"));
  y[438] = 320-Integer.parseInt(getParameter("a438"));
  y[439] = 320-Integer.parseInt(getParameter("a439"));
  y[440] = 320-Integer.parseInt(getParameter("a440"));
  y[441] = 320-Integer.parseInt(getParameter("a441"));
```

```
  y[442] = 320-Integer.parseInt(getParameter("a442"));
  y[443] = 320-Integer.parseInt(getParameter("a443"));
  y[444] = 320-Integer.parseInt(getParameter("a444"));
  y[445] = 320-Integer.parseInt(getParameter("a445"));
  y[446] = 320-Integer.parseInt(getParameter("a446"));
y[447] = 320-Integer.parseInt(getParameter("a447"));
  y[448] = 320-Integer.parseInt(getParameter("a448"));
  y[449] = 320-Integer.parseInt(getParameter("a449"));
  y[450] = 320-Integer.parseInt(getParameter("a450"));
  y[451] = 320-Integer.parseInt(getParameter("a451"));
  y[452] = 320-Integer.parseInt(getParameter("a452"));
  y[453] = 320-Integer.parseInt(getParameter("a453"));
  y[454] = 320-Integer.parseInt(getParameter("a454"));
  y[455] = 320-Integer.parseInt(getParameter("a455"));
  y[456] = 320-Integer.parseInt(getParameter("a456"));
y[457] = 320-Integer.parseInt(getParameter("a457"));
  y[458] = 320-Integer.parseInt(getParameter("a458"));
  y[459] = 320-Integer.parseInt(getParameter("a459"));
  y[460] = 320-Integer.parseInt(getParameter("a460"));
  y[461] = 320-Integer.parseInt(getParameter("a461"));
  y[462] = 320-Integer.parseInt(getParameter("a462"));
  y[463] = 320-Integer.parseInt(getParameter("a463"));
  y[464] = 320-Integer.parseInt(getParameter("a464"));
  y[465] = 320-Integer.parseInt(getParameter("a465"));
  y[466] = 320-Integer.parseInt(getParameter("a466"));
y[467] = 320-Integer.parseInt(getParameter("a467"));
  y[468] = 320-Integer.parseInt(getParameter("a468"));
  y[469] = 320-Integer.parseInt(getParameter("a469"));
  y[470] = 320-Integer.parseInt(getParameter("a470"));
  y[471] = 320-Integer.parseInt(getParameter("a471"));
  y[472] = 320-Integer.parseInt(getParameter("a472"));
  y[473] = 320-Integer.parseInt(getParameter("a473"));
  y[474] = 320-Integer.parseInt(getParameter("a474"));
  y[475] = 320-Integer.parseInt(getParameter("a475"));
  y[476] = 320-Integer.parseInt(getParameter("a476"));
y[477] = 320-Integer.parseInt(getParameter("a477"));
  y[478] = 320-Integer.parseInt(getParameter("a478"));
  y[479] = 320-Integer.parseInt(getParameter("a479"));
  y[480] = 320-Integer.parseInt(getParameter("a480"));
  y[481] = 320-Integer.parseInt(getParameter("a481"));
  y[482] = 320-Integer.parseInt(getParameter("a482"));
  y[483] = 320-Integer.parseInt(getParameter("a483"));
  y[484] = 320-Integer.parseInt(getParameter("a484"));
  y[485] = 320-Integer.parseInt(getParameter("a485"));
  y[486] = 320-Integer.parseInt(getParameter("a486"));
y[487] = 320-Integer.parseInt(getParameter("a487"));
  y[488] = 320-Integer.parseInt(getParameter("a488"));
  y[489] = 320-Integer.parseInt(getParameter("a489"));
  y[490] = 320-Integer.parseInt(getParameter("a490"));
  y[491] = 320-Integer.parseInt(getParameter("a491"));
  y[492] = 320-Integer.parseInt(getParameter("a492"));
  y[493] = 320-Integer.parseInt(getParameter("a493"));
  y[494] = 320-Integer.parseInt(getParameter("a494"));
  y[495] = 320-Integer.parseInt(getParameter("a495"));
  y[496] = 320-Integer.parseInt(getParameter("a496"));
y[497] = 320-Integer.parseInt(getParameter("a497"));
  y[498] = 320-Integer.parseInt(getParameter("a498"));
```

```
  y[499] = 320-Integer.parseInt(getParameter("a499"));
  y[500] = 320-Integer.parseInt(getParameter("a500"));
  y[501] = 320-Integer.parseInt(getParameter("a501"));
  y[502] = 320-Integer.parseInt(getParameter("a502"));
  y[503] = 320-Integer.parseInt(getParameter("a503"));
  y[504] = 320-Integer.parseInt(getParameter("a504"));
  y[505] = 320-Integer.parseInt(getParameter("a505"));
  y[506] = 320-Integer.parseInt(getParameter("a506"));
y[507] = 320-Integer.parseInt(getParameter("a507"));
  y[508] = 320-Integer.parseInt(getParameter("a508"));
  y[509] = 320-Integer.parseInt(getParameter("a509"));
  y[510] = 320-Integer.parseInt(getParameter("a510"));
  y[511] = 320-Integer.parseInt(getParameter("a511"));
  y[512] = 320-Integer.parseInt(getParameter("a512"));
  y[513] = 320-Integer.parseInt(getParameter("a513"));
  y[514] = 320-Integer.parseInt(getParameter("a514"));
  y[515] = 320-Integer.parseInt(getParameter("a515"));
  y[516] = 320-Integer.parseInt(getParameter("a516"));
y[517] = 320-Integer.parseInt(getParameter("a517"));
  y[518] = 320-Integer.parseInt(getParameter("a518"));
  y[519] = 320-Integer.parseInt(getParameter("a519"));
  y[520] = 320-Integer.parseInt(getParameter("a520"));
  y[521] = 320-Integer.parseInt(getParameter("a521"));
  y[522] = 320-Integer.parseInt(getParameter("a522"));
  y[523] = 320-Integer.parseInt(getParameter("a523"));
  y[524] = 320-Integer.parseInt(getParameter("a524"));
  y[525] = 320-Integer.parseInt(getParameter("a525"));
  y[526] = 320-Integer.parseInt(getParameter("a526"));
y[527] = 320-Integer.parseInt(getParameter("a527"));
  y[528] = 320-Integer.parseInt(getParameter("a528"));
  y[529] = 320-Integer.parseInt(getParameter("a529"));
  y[530] = 320-Integer.parseInt(getParameter("a530"));
  y[531] = 320-Integer.parseInt(getParameter("a531"));
  y[532] = 320-Integer.parseInt(getParameter("a532"));
  y[533] = 320-Integer.parseInt(getParameter("a533"));
  y[534] = 320-Integer.parseInt(getParameter("a534"));
  y[535] = 320-Integer.parseInt(getParameter("a535"));
  y[536] = 320-Integer.parseInt(getParameter("a536"));
y[537] = 320-Integer.parseInt(getParameter("a537"));
  y[538] = 320-Integer.parseInt(getParameter("a538"));
  y[539] = 320-Integer.parseInt(getParameter("a539"));
  y[540] = 320-Integer.parseInt(getParameter("a540"));
  y[541] = 320-Integer.parseInt(getParameter("a541"));
  y[542] = 320-Integer.parseInt(getParameter("a542"));
  y[543] = 320-Integer.parseInt(getParameter("a543"));
  y[544] = 320-Integer.parseInt(getParameter("a544"));
  y[545] = 320-Integer.parseInt(getParameter("a545"));
  y[546] = 320-Integer.parseInt(getParameter("a546"));
y[547] = 320-Integer.parseInt(getParameter("a547"));
  y[548] = 320-Integer.parseInt(getParameter("a548"));
  y[549] = 320-Integer.parseInt(getParameter("a549"));
  y[550] = 320-Integer.parseInt(getParameter("a550"));
  y[551] = 320-Integer.parseInt(getParameter("a551"));
  y[552] = 320-Integer.parseInt(getParameter("a552"));
  y[553] = 320-Integer.parseInt(getParameter("a553"));
  y[554] = 320-Integer.parseInt(getParameter("a554"));
  y[555] = 320-Integer.parseInt(getParameter("a555"));
```

```
  y[556] = 320-Integer.parseInt(getParameter("a556"));
y[557] = 320-Integer.parseInt(getParameter("a557"));
  y[558] = 320-Integer.parseInt(getParameter("a558"));
  y[559] = 320-Integer.parseInt(getParameter("a559"));
  y[560] = 320-Integer.parseInt(getParameter("a560"));
  y[561] = 320-Integer.parseInt(getParameter("a561"));
  y[562] = 320-Integer.parseInt(getParameter("a562"));
  y[563] = 320-Integer.parseInt(getParameter("a563"));
  y[564] = 320-Integer.parseInt(getParameter("a564"));
  y[565] = 320-Integer.parseInt(getParameter("a565"));
  y[566] = 320-Integer.parseInt(getParameter("a566"));
y[567] = 320-Integer.parseInt(getParameter("a567"));
  y[568] = 320-Integer.parseInt(getParameter("a568"));
  y[569] = 320-Integer.parseInt(getParameter("a569"));
  y[570] = 320-Integer.parseInt(getParameter("a570"));
  y[571] = 320-Integer.parseInt(getParameter("a571"));
  y[572] = 320-Integer.parseInt(getParameter("a572"));
  y[573] = 320-Integer.parseInt(getParameter("a573"));
  y[574] = 320-Integer.parseInt(getParameter("a574"));
  y[575] = 320-Integer.parseInt(getParameter("a575"));
  y[576] = 320-Integer.parseInt(getParameter("a576"));
y[577] = 320-Integer.parseInt(getParameter("a577"));
  y[578] = 320-Integer.parseInt(getParameter("a578"));
  y[579] = 320-Integer.parseInt(getParameter("a579"));
  y[580] = 320-Integer.parseInt(getParameter("a580"));
  y[581] = 320-Integer.parseInt(getParameter("a581"));
  y[582] = 320-Integer.parseInt(getParameter("a582"));
  y[583] = 320-Integer.parseInt(getParameter("a583"));
  y[584] = 320-Integer.parseInt(getParameter("a584"));
  y[585] = 320-Integer.parseInt(getParameter("a585"));
  y[586] = 320-Integer.parseInt(getParameter("a586"));
y[587] = 320-Integer.parseInt(getParameter("a587"));
  y[588] = 320-Integer.parseInt(getParameter("a588"));
  y[589] = 320-Integer.parseInt(getParameter("a589"));
  y[590] = 320-Integer.parseInt(getParameter("a590"));
  y[591] = 320-Integer.parseInt(getParameter("a591"));
  y[592] = 320-Integer.parseInt(getParameter("a592"));
  y[593] = 320-Integer.parseInt(getParameter("a593"));
  y[594] = 320-Integer.parseInt(getParameter("a594"));
  y[595] = 320-Integer.parseInt(getParameter("a595"));
  y[596] = 320-Integer.parseInt(getParameter("a596"));
y[597] = 320-Integer.parseInt(getParameter("a597"));
  y[598] = 320-Integer.parseInt(getParameter("a598"));
  y[599] = 320-Integer.parseInt(getParameter("a599"));
  y[600] = 320-Integer.parseInt(getParameter("a600"));
  y[601] = 320-Integer.parseInt(getParameter("a601"));
  y[602] = 320-Integer.parseInt(getParameter("a602"));
  y[603] = 320-Integer.parseInt(getParameter("a603"));
  y[604] = 320-Integer.parseInt(getParameter("a604"));
  y[605] = 320-Integer.parseInt(getParameter("a605"));
  y[606] = 320-Integer.parseInt(getParameter("a606"));
y[607] = 320-Integer.parseInt(getParameter("a607"));
  y[608] = 320-Integer.parseInt(getParameter("a608"));
  y[609] = 320-Integer.parseInt(getParameter("a609"));
  y[610] = 320-Integer.parseInt(getParameter("a610"));
  y[611] = 320-Integer.parseInt(getParameter("a611"));
  y[612] = 320-Integer.parseInt(getParameter("a612"));
```

```
  y[613] = 320-Integer.parseInt(getParameter("a613"));
  y[614] = 320-Integer.parseInt(getParameter("a614"));
  y[615] = 320-Integer.parseInt(getParameter("a615"));
  y[616] = 320-Integer.parseInt(getParameter("a616"));
y[617] = 320-Integer.parseInt(getParameter("a617"));
  y[618] = 320-Integer.parseInt(getParameter("a618"));
  y[619] = 320-Integer.parseInt(getParameter("a619"));
  y[620] = 320-Integer.parseInt(getParameter("a620"));
  y[621] = 320-Integer.parseInt(getParameter("a621"));
  y[622] = 320-Integer.parseInt(getParameter("a622"));
  y[623] = 320-Integer.parseInt(getParameter("a623"));
  y[624] = 320-Integer.parseInt(getParameter("a624"));
  y[625] = 320-Integer.parseInt(getParameter("a625"));
  y[626] = 320-Integer.parseInt(getParameter("a626"));
y[627] = 320-Integer.parseInt(getParameter("a627"));
  y[628] = 320-Integer.parseInt(getParameter("a628"));
  y[629] = 320-Integer.parseInt(getParameter("a629"));
  y[630] = 320-Integer.parseInt(getParameter("a630"));
  y[631] = 320-Integer.parseInt(getParameter("a631"));
  y[632] = 320-Integer.parseInt(getParameter("a632"));
  y[633] = 320-Integer.parseInt(getParameter("a633"));
  y[634] = 320-Integer.parseInt(getParameter("a634"));
  y[635] = 320-Integer.parseInt(getParameter("a635"));
  y[636] = 320-Integer.parseInt(getParameter("a636"));
y[637] = 320-Integer.parseInt(getParameter("a637"));
  y[638] = 320-Integer.parseInt(getParameter("a638"));
  y[639] = 320-Integer.parseInt(getParameter("a639"));
  y[640] = 320-Integer.parseInt(getParameter("a640"));
  y[641] = 320-Integer.parseInt(getParameter("a641"));
  y[642] = 320-Integer.parseInt(getParameter("a642"));
  y[643] = 320-Integer.parseInt(getParameter("a643"));
  y[644] = 320-Integer.parseInt(getParameter("a644"));
  y[645] = 320-Integer.parseInt(getParameter("a645"));
  y[646] = 320-Integer.parseInt(getParameter("a646"));
y[647] = 320-Integer.parseInt(getParameter("a647"));
  y[648] = 320-Integer.parseInt(getParameter("a648"));
  y[649] = 320-Integer.parseInt(getParameter("a649"));
  y[650] = 320-Integer.parseInt(getParameter("a650"));
  y[651] = 320-Integer.parseInt(getParameter("a651"));
  y[652] = 320-Integer.parseInt(getParameter("a652"));
  y[653] = 320-Integer.parseInt(getParameter("a653"));
  y[654] = 320-Integer.parseInt(getParameter("a654"));
  y[655] = 320-Integer.parseInt(getParameter("a655"));
  y[656] = 320-Integer.parseInt(getParameter("a656"));
y[657] = 320-Integer.parseInt(getParameter("a657"));
  y[658] = 320-Integer.parseInt(getParameter("a658"));
  y[659] = 320-Integer.parseInt(getParameter("a659"));
  y[660] = 320-Integer.parseInt(getParameter("a660"));
  y[661] = 320-Integer.parseInt(getParameter("a661"));
  y[662] = 320-Integer.parseInt(getParameter("a662"));
  y[663] = 320-Integer.parseInt(getParameter("a663"));
  y[664] = 320-Integer.parseInt(getParameter("a664"));
  y[665] = 320-Integer.parseInt(getParameter("a665"));
  y[666] = 320-Integer.parseInt(getParameter("a666"));
y[667] = 320-Integer.parseInt(getParameter("a667"));
  y[668] = 320-Integer.parseInt(getParameter("a668"));
  y[669] = 320-Integer.parseInt(getParameter("a669"));
```

```
        y670 = 320-Integer.parseInt(getParameter("a670"));


        setBackground(Color.black);
        new Thread(this).start();
}

public void update (Graphics g) {paint(g);}
public void paint(Graphics g)
{
        g.setColor(Color.yellow);
        g.drawLine(31, 30, 31, 360);
        g.drawLine(31, 30, 745, 30);
        g.drawLine(31, 360, 745, 360);
        g.drawLine(745, 30, 745, 360);

        g.setColor(Color.yellow);
        g.drawLine(30, 29, 30, 361);
        g.drawLine(30, 29, 746, 29);
        g.drawLine(30, 361, 746, 361);
        g.drawLine(746, 29, 746, 361);

        g.drawLine(25, 35,  35, 35);
        g.drawLine(25, 75,  35, 75);
        g.drawLine(25, 115,  35, 115);
        g.drawLine(25, 155, 35, 155);
        g.drawLine(25, 195, 35, 195);
        g.drawLine(25, 235, 35, 235);
        g.drawLine(25, 275, 35, 275);
        g.drawLine(25, 315, 35, 315);
        g.drawLine(25, 355, 35, 355);

        g.setColor(Color.blue) ;   g.drawString("2.0",8,35);
        g.setColor(Color.blue) ;   g.drawString("1.5",8,75);
        g.setColor(Color.blue) ;   g.drawString("1.0",8,115);
        g.setColor(Color.blue) ;   g.drawString("0.5",8,155);
        g.setColor(Color.blue) ;  g.drawString("0",15,195);
        g.setColor(Color.blue) ;   g.drawString("-0.5",3,235);
        g.setColor(Color.blue) ;   g.drawString("-1.0",3,275);
        g.setColor(Color.blue) ;   g.drawString("-1.5",3,315);
        g.setColor(Color.blue) ;   g.drawString("-2.0",3,355);

        for(j = 0; j < 669; j++)
        {
              g.setColor(Color.black);
              g.fillRect(x, 32, 8, 255);
              g.setColor(Color.red);
              g.drawLine(x, y[j],  x++, y[j+1]);
              try { Thread.sleep(15);
        }
              catch (Exception e) { }
        }
              g.setColor(Color.black);
              g.fillRect(x, 32, 8, 255);
              g.setColor(Color.red);
              g.drawLine(x, y670, x++, y[0]);
              try { Thread.sleep(15);
```

```
            }
                  catch (Exception e) { }
                  x = 60;
      }

      public void run()
      {
            while ( i < 255 )
            {
                  try{ Thread.sleep(15); }
                  catch (Exception e) { }
                  repaint();
            }
      }
}
```

โปรแกรมแสดงหน้าแรกของ ECG

```
<html>
<head>
<title>ECG Graph Display's  BLUESKY</title>
<style>
<!--
.styling{
background-color:black;
color:lime;
font: bold 16px MS Sans Serif;
padding: 3px;
}
-->
</style>
</head>
<meta http-equiv="Refresh" content="10";url=kmitl.html>
<center>
<!--  -->
<!--body onLoad=window.setTimeout("location.href='kmitl.html'",1000)-->
<table width="600" >
  <tr>
    <td width="400">
      <applet  code = "kmitlECG.class" width    = "770"     height
      = "390"     >
            <param name=a0 value="<!--#echo var="B"-->">
            <param name=a1 value="<!--#echo var="B"-->">
            <param name=a2 value="<!--#echo var="B"-->">
            <param name=a3 value="<!--#echo var="B"-->">
            <param name=a4 value="<!--#echo var="B"-->">
            <param name=a5 value="<!--#echo var="B"-->">
                  <param name=a6 value="<!--#echo var="B"-->">
            <param name=a7 value="<!--#echo var="B"-->">
            <param name=a8 value="<!--#echo var="B"-->">
            <param name=a9 value="<!--#echo var="B"-->">
            <param name=a10 value="<!--#echo var="B"-->">
            <param name=a11 value="<!--#echo var="B"-->">
            <param name=a12 value="<!--#echo var="B"-->">
            <param name=a13 value="<!--#echo var="B"-->">
            <param name=a14 value="<!--#echo var="B"-->">
            <param name=a15 value="<!--#echo var="B"-->">
                  <param name=a16 value="<!--#echo var="B"-->">
            <param name=a17 value="<!--#echo var="B"-->">
            <param name=a18 value="<!--#echo var="B"-->">
            <param name=a19 value="<!--#echo var="B"-->">
            <param name=a20 value="<!--#echo var="B"-->">
            <param name=a21 value="<!--#echo var="B"-->">
            <param name=a22 value="<!--#echo var="B"-->">
            <param name=a23 value="<!--#echo var="B"-->">
            <param name=a24 value="<!--#echo var="B"-->">
            <param name=a25 value="<!--#echo var="B"-->">
                  <param name=a26 value="<!--#echo var="B"-->">
            <param name=a27 value="<!--#echo var="B"-->">
            <param name=a28 value="<!--#echo var="B"-->">
            <param name=a29 value="<!--#echo var="B"-->">
```

```
<param name=a30 value="<!--#echo var="B"-->">
<param name=a31 value="<!--#echo var="B"-->">
<param name=a32 value="<!--#echo var="B"-->">
<param name=a33 value="<!--#echo var="B"-->">
<param name=a34 value="<!--#echo var="B"-->">
<param name=a35 value="<!--#echo var="B"-->">
       <param name=a36 value="<!--#echo var="B"-->">
<param name=a37 value="<!--#echo var="B"-->">
<param name=a38 value="<!--#echo var="B"-->">
<param name=a39 value="<!--#echo var="B"-->">
<param name=a40 value="<!--#echo var="B"-->">
<param name=a41 value="<!--#echo var="B"-->">
<param name=a42 value="<!--#echo var="B"-->">
<param name=a43 value="<!--#echo var="B"-->">
<param name=a44 value="<!--#echo var="B"-->">
<param name=a45 value="<!--#echo var="B"-->">
       <param name=a46 value="<!--#echo var="B"-->">
<param name=a47 value="<!--#echo var="B"-->">
<param name=a48 value="<!--#echo var="B"-->">
<param name=a49 value="<!--#echo var="B"-->">
<param name=a50 value="<!--#echo var="B"-->">
<param name=a51 value="<!--#echo var="B"-->">
<param name=a52 value="<!--#echo var="B"-->">
<param name=a53 value="<!--#echo var="B"-->">
<param name=a54 value="<!--#echo var="B"-->">
<param name=a55 value="<!--#echo var="B"-->">
       <param name=a56 value="<!--#echo var="B"-->">
<param name=a57 value="<!--#echo var="B"-->">
<param name=a58 value="<!--#echo var="B"-->">
<param name=a59 value="<!--#echo var="B"-->">
<param name=a60 value="<!--#echo var="B"-->">
<param name=a61 value="<!--#echo var="B"-->">
<param name=a62 value="<!--#echo var="B"-->">
<param name=a63 value="<!--#echo var="B"-->">
<param name=a64 value="<!--#echo var="B"-->">
<param name=a65 value="<!--#echo var="B"-->">
       <param name=a66 value="<!--#echo var="B"-->">
<param name=a67 value="<!--#echo var="B"-->">
<param name=a68 value="<!--#echo var="B"-->">
<param name=a69 value="<!--#echo var="B"-->">
<param name=a70 value="<!--#echo var="B"-->">
<param name=a71 value="<!--#echo var="B"-->">
<param name=a72 value="<!--#echo var="B"-->">
<param name=a73 value="<!--#echo var="B"-->">
<param name=a74 value="<!--#echo var="B"-->">
<param name=a75 value="<!--#echo var="B"-->">
       <param name=a76 value="<!--#echo var="B"-->">
<param name=a77 value="<!--#echo var="B"-->">
<param name=a78 value="<!--#echo var="B"-->">
<param name=a79 value="<!--#echo var="B"-->">
<param name=a80 value="<!--#echo var="B"-->">
<param name=a81 value="<!--#echo var="B"-->">
<param name=a82 value="<!--#echo var="B"-->">
<param name=a83 value="<!--#echo var="B"-->">
<param name=a84 value="<!--#echo var="B"-->">
<param name=a85 value="<!--#echo var="B"-->">
```

```
      <param name=a86 value="<!--#echo var="B"-->">
<param name=a87 value="<!--#echo var="B"-->">
<param name=a88 value="<!--#echo var="B"-->">
<param name=a89 value="<!--#echo var="B"-->">
<param name=a90 value="<!--#echo var="B"-->">
<param name=a91 value="<!--#echo var="B"-->">
<param name=a92 value="<!--#echo var="B"-->">
<param name=a93 value="<!--#echo var="B"-->">
<param name=a94 value="<!--#echo var="B"-->">
<param name=a95 value="<!--#echo var="B"-->">
       <param name=a96 value="<!--#echo var="B"-->">
<param name=a97 value="<!--#echo var="B"-->">
<param name=a98 value="<!--#echo var="B"-->">
<param name=a99 value="<!--#echo var="B"-->">
<param name=a100 value="<!--#echo var="B"-->">
<param name=a101 value="<!--#echo var="B"-->">
<param name=a102 value="<!--#echo var="B"-->">
<param name=a103 value="<!--#echo var="B"-->">
<param name=a104 value="<!--#echo var="B"-->">
<param name=a105 value="<!--#echo var="B"-->">
       <param name=a106 value="<!--#echo var="B"-->">
<param name=a107 value="<!--#echo var="B"-->">
<param name=a108 value="<!--#echo var="B"-->">
<param name=a109 value="<!--#echo var="B"-->">
<param name=a110 value="<!--#echo var="B"-->">
<param name=a111 value="<!--#echo var="B"-->">
<param name=a112 value="<!--#echo var="B"-->">
<param name=a113 value="<!--#echo var="B"-->">
<param name=a114 value="<!--#echo var="B"-->">
<param name=a115 value="<!--#echo var="B"-->">
       <param name=a116 value="<!--#echo var="B"-->">
<param name=a117 value="<!--#echo var="B"-->">
<param name=a118 value="<!--#echo var="B"-->">
<param name=a119 value="<!--#echo var="B"-->">
<param name=a120 value="<!--#echo var="B"-->">
<param name=a121 value="<!--#echo var="B"-->">
<param name=a122 value="<!--#echo var="B"-->">
<param name=a123 value="<!--#echo var="B"-->">
<param name=a124 value="<!--#echo var="B"-->">
<param name=a125 value="<!--#echo var="B"-->">
       <param name=a126 value="<!--#echo var="B"-->">
<param name=a127 value="<!--#echo var="B"-->">
<param name=a128 value="<!--#echo var="B"-->">
<param name=a129 value="<!--#echo var="B"-->">
<param name=a130 value="<!--#echo var="B"-->">
<param name=a131 value="<!--#echo var="B"-->">
<param name=a132 value="<!--#echo var="B"-->">
<param name=a133 value="<!--#echo var="B"-->">
<param name=a134 value="<!--#echo var="B"-->">
<param name=a135 value="<!--#echo var="B"-->">
       <param name=a136 value="<!--#echo var="B"-->">
<param name=a137 value="<!--#echo var="B"-->">
<param name=a138 value="<!--#echo var="B"-->">
<param name=a139 value="<!--#echo var="B"-->">
<param name=a140 value="<!--#echo var="B"-->">
<param name=a141 value="<!--#echo var="B"-->">
<param name=a142 value="<!--#echo var="B"-->">
```

```
<param name=a143 value="<!--#echo var="B"-->">
<param name=a144 value="<!--#echo var="B"-->">
<param name=a145 value="<!--#echo var="B"-->">
      <param name=a146 value="<!--#echo var="B"-->">
<param name=a147 value="<!--#echo var="B"-->">
<param name=a148 value="<!--#echo var="B"-->">
<param name=a149 value="<!--#echo var="B"-->">
<param name=a150 value="<!--#echo var="B"-->">
<param name=a151 value="<!--#echo var="B"-->">
<param name=a152 value="<!--#echo var="B"-->">
<param name=a153 value="<!--#echo var="B"-->">
<param name=a154 value="<!--#echo var="B"-->">
<param name=a155 value="<!--#echo var="B"-->">
      <param name=a156 value="<!--#echo var="B"-->">
<param name=a157 value="<!--#echo var="B"-->">
<param name=a158 value="<!--#echo var="B"-->">
<param name=a159 value="<!--#echo var="B"-->">
<param name=a160 value="<!--#echo var="B"-->">
<param name=a161 value="<!--#echo var="B"-->">
<param name=a162 value="<!--#echo var="B"-->">
<param name=a163 value="<!--#echo var="B"-->">
<param name=a164 value="<!--#echo var="B"-->">
<param name=a165 value="<!--#echo var="B"-->">
      <param name=a166 value="<!--#echo var="B"-->">
<param name=a167 value="<!--#echo var="B"-->">
<param name=a168 value="<!--#echo var="B"-->">
<param name=a169 value="<!--#echo var="B"-->">
<param name=a170 value="<!--#echo var="B"-->">
<param name=a171 value="<!--#echo var="B"-->">
<param name=a172 value="<!--#echo var="B"-->">
<param name=a173 value="<!--#echo var="B"-->">
<param name=a174 value="<!--#echo var="B"-->">
<param name=a175 value="<!--#echo var="B"-->">
      <param name=a176 value="<!--#echo var="B"-->">
<param name=a177 value="<!--#echo var="B"-->">
<param name=a178 value="<!--#echo var="B"-->">
<param name=a179 value="<!--#echo var="B"-->">
<param name=a180 value="<!--#echo var="B"-->">
<param name=a181 value="<!--#echo var="B"-->">
<param name=a182 value="<!--#echo var="B"-->">
<param name=a183 value="<!--#echo var="B"-->">
<param name=a184 value="<!--#echo var="B"-->">
<param name=a185 value="<!--#echo var="B"-->">
      <param name=a186 value="<!--#echo var="B"-->">
<param name=a187 value="<!--#echo var="B"-->">
<param name=a188 value="<!--#echo var="B"-->">
<param name=a189 value="<!--#echo var="B"-->">
<param name=a190 value="<!--#echo var="B"-->">
<param name=a191 value="<!--#echo var="B"-->">
<param name=a192 value="<!--#echo var="B"-->">
<param name=a193 value="<!--#echo var="B"-->">
<param name=a194 value="<!--#echo var="B"-->">
<param name=a195 value="<!--#echo var="B"-->">
      <param name=a196 value="<!--#echo var="B"-->">
<param name=a197 value="<!--#echo var="B"-->">
<param name=a198 value="<!--#echo var="B"-->">
<param name=a199 value="<!--#echo var="B"-->">
```

```
<param name=a200 value="<!--#echo var="B"-->">
<param name=a201 value="<!--#echo var="B"-->">
<param name=a202 value="<!--#echo var="B"-->">
<param name=a203 value="<!--#echo var="B"-->">
<param name=a204 value="<!--#echo var="B"-->">
<param name=a205 value="<!--#echo var="B"-->">
      <param name=a206 value="<!--#echo var="B"-->">
<param name=a207 value="<!--#echo var="B"-->">
<param name=a208 value="<!--#echo var="B"-->">
<param name=a209 value="<!--#echo var="B"-->">
<param name=a210 value="<!--#echo var="B"-->">
<param name=a211 value="<!--#echo var="B"-->">
<param name=a212 value="<!--#echo var="B"-->">
<param name=a213 value="<!--#echo var="B"-->">
<param name=a214 value="<!--#echo var="B"-->">
<param name=a215 value="<!--#echo var="B"-->">
      <param name=a216 value="<!--#echo var="B"-->">
<param name=a217 value="<!--#echo var="B"-->">
<param name=a218 value="<!--#echo var="B"-->">
<param name=a219 value="<!--#echo var="B"-->">
<param name=a220 value="<!--#echo var="B"-->">
<param name=a221 value="<!--#echo var="B"-->">
<param name=a222 value="<!--#echo var="B"-->">
<param name=a223 value="<!--#echo var="B"-->">
<param name=a224 value="<!--#echo var="B"-->">
<param name=a225 value="<!--#echo var="B"-->">
      <param name=a226 value="<!--#echo var="B"-->">
<param name=a227 value="<!--#echo var="B"-->">
<param name=a228 value="<!--#echo var="B"-->">
<param name=a229 value="<!--#echo var="B"-->">
<param name=a230 value="<!--#echo var="B"-->">
<param name=a231 value="<!--#echo var="B"-->">
<param name=a232 value="<!--#echo var="B"-->">
<param name=a233 value="<!--#echo var="B"-->">
<param name=a234 value="<!--#echo var="B"-->">
<param name=a235 value="<!--#echo var="B"-->">
      <param name=a236 value="<!--#echo var="B"-->">
<param name=a237 value="<!--#echo var="B"-->">
<param name=a238 value="<!--#echo var="B"-->">
<param name=a239 value="<!--#echo var="B"-->">
<param name=a240 value="<!--#echo var="B"-->">
<param name=a241 value="<!--#echo var="B"-->">
<param name=a242 value="<!--#echo var="B"-->">
<param name=a243 value="<!--#echo var="B"-->">
<param name=a244 value="<!--#echo var="B"-->">
<param name=a245 value="<!--#echo var="B"-->">
      <param name=a246 value="<!--#echo var="B"-->">
<param name=a247 value="<!--#echo var="B"-->">
<param name=a248 value="<!--#echo var="B"-->">
<param name=a249 value="<!--#echo var="B"-->">
<param name=a250 value="<!--#echo var="B"-->">
<param name=a251 value="<!--#echo var="B"-->">
<param name=a252 value="<!--#echo var="B"-->">
<param name=a253 value="<!--#echo var="B"-->">
<param name=a254 value="<!--#echo var="B"-->">
<param name=a255 value="<!--#echo var="B"-->">
```

```
      <param name=a256 value="<!--#echo var="B"-->">
<param name=a257 value="<!--#echo var="B"-->">
<param name=a258 value="<!--#echo var="B"-->">
<param name=a259 value="<!--#echo var="B"-->">
<param name=a260 value="<!--#echo var="B"-->">
<param name=a261 value="<!--#echo var="B"-->">
<param name=a262 value="<!--#echo var="B"-->">
<param name=a263 value="<!--#echo var="B"-->">
<param name=a264 value="<!--#echo var="B"-->">
<param name=a265 value="<!--#echo var="B"-->">
       <param name=a266 value="<!--#echo var="B"-->">
<param name=a267 value="<!--#echo var="B"-->">
<param name=a268 value="<!--#echo var="B"-->">
<param name=a269 value="<!--#echo var="B"-->">
<param name=a270 value="<!--#echo var="B"-->">
<param name=a271 value="<!--#echo var="B"-->">
<param name=a272 value="<!--#echo var="B"-->">
<param name=a273 value="<!--#echo var="B"-->">
<param name=a274 value="<!--#echo var="B"-->">
<param name=a275 value="<!--#echo var="B"-->">
       <param name=a276 value="<!--#echo var="B"-->">
<param name=a277 value="<!--#echo var="B"-->">
<param name=a278 value="<!--#echo var="B"-->">
<param name=a279 value="<!--#echo var="B"-->">
<param name=a280 value="<!--#echo var="B"-->">
<param name=a281 value="<!--#echo var="B"-->">
<param name=a282 value="<!--#echo var="B"-->">
<param name=a283 value="<!--#echo var="B"-->">
<param name=a284 value="<!--#echo var="B"-->">
<param name=a285 value="<!--#echo var="B"-->">
       <param name=a286 value="<!--#echo var="B"-->">
<param name=a287 value="<!--#echo var="B"-->">
<param name=a288 value="<!--#echo var="B"-->">
<param name=a289 value="<!--#echo var="B"-->">
<param name=a290 value="<!--#echo var="B"-->">
<param name=a291 value="<!--#echo var="B"-->">
<param name=a292 value="<!--#echo var="B"-->">
<param name=a293 value="<!--#echo var="B"-->">
<param name=a294 value="<!--#echo var="B"-->">
<param name=a295 value="<!--#echo var="B"-->">
       <param name=a296 value="<!--#echo var="B"-->">
<param name=a297 value="<!--#echo var="B"-->">
<param name=a298 value="<!--#echo var="B"-->">
<param name=a299 value="<!--#echo var="B"-->">
<param name=a300 value="<!--#echo var="B"-->">
<param name=a301 value="<!--#echo var="B"-->">
<param name=a302 value="<!--#echo var="B"-->">
<param name=a303 value="<!--#echo var="B"-->">
<param name=a304 value="<!--#echo var="B"-->">
<param name=a305 value="<!--#echo var="B"-->">
       <param name=a306 value="<!--#echo var="B"-->">
<param name=a307 value="<!--#echo var="B"-->">
<param name=a308 value="<!--#echo var="B"-->">
<param name=a309 value="<!--#echo var="B"-->">
<param name=a310 value="<!--#echo var="B"-->">
<param name=a311 value="<!--#echo var="B"-->">
<param name=a312 value="<!--#echo var="B"-->">
```

```
<param name=a313 value="<!--#echo var="B"-->">
<param name=a314 value="<!--#echo var="B"-->">
<param name=a315 value="<!--#echo var="B"-->">
      <param name=a316 value="<!--#echo var="B"-->">
<param name=a317 value="<!--#echo var="B"-->">
<param name=a318 value="<!--#echo var="B"-->">
<param name=a319 value="<!--#echo var="B"-->">
<param name=a320 value="<!--#echo var="B"-->">
<param name=a321 value="<!--#echo var="B"-->">
<param name=a322 value="<!--#echo var="B"-->">
<param name=a323 value="<!--#echo var="B"-->">
<param name=a324 value="<!--#echo var="B"-->">
<param name=a325 value="<!--#echo var="B"-->">
      <param name=a326 value="<!--#echo var="B"-->">
<param name=a327 value="<!--#echo var="B"-->">
<param name=a328 value="<!--#echo var="B"-->">
<param name=a329 value="<!--#echo var="B"-->">
<param name=a330 value="<!--#echo var="B"-->">
<param name=a331 value="<!--#echo var="B"-->">
<param name=a332 value="<!--#echo var="B"-->">
<param name=a333 value="<!--#echo var="B"-->">
<param name=a334 value="<!--#echo var="B"-->">
<param name=a335 value="<!--#echo var="B"-->">
      <param name=a336 value="<!--#echo var="B"-->">
<param name=a337 value="<!--#echo var="B"-->">
<param name=a338 value="<!--#echo var="B"-->">
<param name=a339 value="<!--#echo var="B"-->">
<param name=a340 value="<!--#echo var="B"-->">
<param name=a341 value="<!--#echo var="B"-->">
<param name=a342 value="<!--#echo var="B"-->">
<param name=a343 value="<!--#echo var="B"-->">
<param name=a344 value="<!--#echo var="B"-->">
<param name=a345 value="<!--#echo var="B"-->">
      <param name=a346 value="<!--#echo var="B"-->">
<param name=a347 value="<!--#echo var="B"-->">
<param name=a348 value="<!--#echo var="B"-->">
<param name=a349 value="<!--#echo var="B"-->">
<param name=a350 value="<!--#echo var="B"-->">
<param name=a351 value="<!--#echo var="B"-->">
<param name=a352 value="<!--#echo var="B"-->">
<param name=a353 value="<!--#echo var="B"-->">
<param name=a354 value="<!--#echo var="B"-->">
<param name=a355 value="<!--#echo var="B"-->">
      <param name=a356 value="<!--#echo var="B"-->">
<param name=a357 value="<!--#echo var="B"-->">
<param name=a358 value="<!--#echo var="B"-->">
<param name=a359 value="<!--#echo var="B"-->">
<param name=a360 value="<!--#echo var="B"-->">
<param name=a361 value="<!--#echo var="B"-->">
<param name=a362 value="<!--#echo var="B"-->">
<param name=a363 value="<!--#echo var="B"-->">
<param name=a364 value="<!--#echo var="B"-->">
<param name=a365 value="<!--#echo var="B"-->">
      <param name=a366 value="<!--#echo var="B"-->">
<param name=a367 value="<!--#echo var="B"-->">
<param name=a368 value="<!--#echo var="B"-->">
<param name=a369 value="<!--#echo var="B"-->">
```

```
<param name=a370 value="<!--#echo var="B"-->">
<param name=a371 value="<!--#echo var="B"-->">
<param name=a372 value="<!--#echo var="B"-->">
<param name=a373 value="<!--#echo var="B"-->">
<param name=a374 value="<!--#echo var="B"-->">
<param name=a375 value="<!--#echo var="B"-->">
       <param name=a376 value="<!--#echo var="B"-->">
<param name=a377 value="<!--#echo var="B"-->">
<param name=a378 value="<!--#echo var="B"-->">
<param name=a379 value="<!--#echo var="B"-->">
<param name=a380 value="<!--#echo var="B"-->">
<param name=a381 value="<!--#echo var="B"-->">
<param name=a382 value="<!--#echo var="B"-->">
<param name=a383 value="<!--#echo var="B"-->">
<param name=a384 value="<!--#echo var="B"-->">
<param name=a385 value="<!--#echo var="B"-->">
       <param name=a386 value="<!--#echo var="B"-->">
<param name=a387 value="<!--#echo var="B"-->">
<param name=a388 value="<!--#echo var="B"-->">
<param name=a389 value="<!--#echo var="B"-->">
<param name=a390 value="<!--#echo var="B"-->">
<param name=a391 value="<!--#echo var="B"-->">
<param name=a392 value="<!--#echo var="B"-->">
<param name=a393 value="<!--#echo var="B"-->">
<param name=a394 value="<!--#echo var="B"-->">
<param name=a395 value="<!--#echo var="B"-->">
       <param name=a396 value="<!--#echo var="B"-->">
<param name=a397 value="<!--#echo var="B"-->">
<param name=a398 value="<!--#echo var="B"-->">
<param name=a399 value="<!--#echo var="B"-->">
<param name=a400 value="<!--#echo var="B"-->">
<param name=a401 value="<!--#echo var="B"-->">
<param name=a402 value="<!--#echo var="B"-->">
<param name=a403 value="<!--#echo var="B"-->">
<param name=a404 value="<!--#echo var="B"-->">
<param name=a405 value="<!--#echo var="B"-->">
       <param name=a406 value="<!--#echo var="B"-->">
<param name=a407 value="<!--#echo var="B"-->">
<param name=a408 value="<!--#echo var="B"-->">
<param name=a409 value="<!--#echo var="B"-->">
<param name=a410 value="<!--#echo var="B"-->">
<param name=a411 value="<!--#echo var="B"-->">
<param name=a412 value="<!--#echo var="B"-->">
<param name=a413 value="<!--#echo var="B"-->">
<param name=a414 value="<!--#echo var="B"-->">
<param name=a415 value="<!--#echo var="B"-->">
       <param name=a416 value="<!--#echo var="B"-->">
<param name=a417 value="<!--#echo var="B"-->">
<param name=a418 value="<!--#echo var="B"-->">
<param name=a419 value="<!--#echo var="B"-->">
<param name=a420 value="<!--#echo var="B"-->">
<param name=a421 value="<!--#echo var="B"-->">
<param name=a422 value="<!--#echo var="B"-->">
<param name=a423 value="<!--#echo var="B"-->">
<param name=a424 value="<!--#echo var="B"-->">
<param name=a425 value="<!--#echo var="B"-->">
```

```
      <param name=a426 value="<!--#echo var="B"-->">
<param name=a427 value="<!--#echo var="B"-->">
<param name=a428 value="<!--#echo var="B"-->">
<param name=a429 value="<!--#echo var="B"-->">
<param name=a430 value="<!--#echo var="B"-->">
<param name=a431 value="<!--#echo var="B"-->">
<param name=a432 value="<!--#echo var="B"-->">
<param name=a433 value="<!--#echo var="B"-->">
<param name=a434 value="<!--#echo var="B"-->">
<param name=a435 value="<!--#echo var="B"-->">
       <param name=a436 value="<!--#echo var="B"-->">
<param name=a437 value="<!--#echo var="B"-->">
<param name=a438 value="<!--#echo var="B"-->">
<param name=a439 value="<!--#echo var="B"-->">
<param name=a440 value="<!--#echo var="B"-->">
<param name=a441 value="<!--#echo var="B"-->">
<param name=a442 value="<!--#echo var="B"-->">
<param name=a443 value="<!--#echo var="B"-->">
<param name=a444 value="<!--#echo var="B"-->">
<param name=a445 value="<!--#echo var="B"-->">
       <param name=a446 value="<!--#echo var="B"-->">
<param name=a447 value="<!--#echo var="B"-->">
<param name=a448 value="<!--#echo var="B"-->">
<param name=a449 value="<!--#echo var="B"-->">
<param name=a450 value="<!--#echo var="B"-->">
<param name=a451 value="<!--#echo var="B"-->">
<param name=a452 value="<!--#echo var="B"-->">
<param name=a453 value="<!--#echo var="B"-->">
<param name=a454 value="<!--#echo var="B"-->">
<param name=a455 value="<!--#echo var="B"-->">
       <param name=a456 value="<!--#echo var="B"-->">
<param name=a457 value="<!--#echo var="B"-->">
<param name=a458 value="<!--#echo var="B"-->">
<param name=a459 value="<!--#echo var="B"-->">
<param name=a460 value="<!--#echo var="B"-->">
<param name=a461 value="<!--#echo var="B"-->">
<param name=a462 value="<!--#echo var="B"-->">
<param name=a463 value="<!--#echo var="B"-->">
<param name=a464 value="<!--#echo var="B"-->">
<param name=a465 value="<!--#echo var="B"-->">
       <param name=a466 value="<!--#echo var="B"-->">
<param name=a467 value="<!--#echo var="B"-->">
<param name=a468 value="<!--#echo var="B"-->">
<param name=a469 value="<!--#echo var="B"-->">
<param name=a470 value="<!--#echo var="B"-->">
<param name=a471 value="<!--#echo var="B"-->">
<param name=a472 value="<!--#echo var="B"-->">
<param name=a473 value="<!--#echo var="B"-->">
<param name=a474 value="<!--#echo var="B"-->">
<param name=a475 value="<!--#echo var="B"-->">
       <param name=a476 value="<!--#echo var="B"-->">
<param name=a477 value="<!--#echo var="B"-->">
<param name=a478 value="<!--#echo var="B"-->">
<param name=a479 value="<!--#echo var="B"-->">
<param name=a480 value="<!--#echo var="B"-->">
<param name=a481 value="<!--#echo var="B"-->">
<param name=a482 value="<!--#echo var="B"-->">
```

```
<param name=a483 value="<!--#echo var="B"-->">
<param name=a484 value="<!--#echo var="B"-->">
<param name=a485 value="<!--#echo var="B"-->">
      <param name=a486 value="<!--#echo var="B"-->">
<param name=a487 value="<!--#echo var="B"-->">
<param name=a488 value="<!--#echo var="B"-->">
<param name=a489 value="<!--#echo var="B"-->">
<param name=a490 value="<!--#echo var="B"-->">
<param name=a491 value="<!--#echo var="B"-->">
<param name=a492 value="<!--#echo var="B"-->">
<param name=a493 value="<!--#echo var="B"-->">
<param name=a494 value="<!--#echo var="B"-->">
<param name=a495 value="<!--#echo var="B"-->">
      <param name=a496 value="<!--#echo var="B"-->">
<param name=a497 value="<!--#echo var="B"-->">
<param name=a498 value="<!--#echo var="B"-->">
<param name=a499 value="<!--#echo var="B"-->">
<param name=a500 value="<!--#echo var="B"-->">
<param name=a501 value="<!--#echo var="B"-->">
<param name=a502 value="<!--#echo var="B"-->">
<param name=a503 value="<!--#echo var="B"-->">
<param name=a504 value="<!--#echo var="B"-->">
<param name=a505 value="<!--#echo var="B"-->">
      <param name=a506 value="<!--#echo var="B"-->">
<param name=a507 value="<!--#echo var="B"-->">
<param name=a508 value="<!--#echo var="B"-->">
<param name=a509 value="<!--#echo var="B"-->">
<param name=a510 value="<!--#echo var="B"-->">
<param name=a511 value="<!--#echo var="B"-->">
<param name=a512 value="<!--#echo var="B"-->">
<param name=a513 value="<!--#echo var="B"-->">
<param name=a514 value="<!--#echo var="B"-->">
<param name=a515 value="<!--#echo var="B"-->">
      <param name=a516 value="<!--#echo var="B"-->">
<param name=a517 value="<!--#echo var="B"-->">
<param name=a518 value="<!--#echo var="B"-->">
<param name=a519 value="<!--#echo var="B"-->">
<param name=a520 value="<!--#echo var="B"-->">
<param name=a521 value="<!--#echo var="B"-->">
<param name=a522 value="<!--#echo var="B"-->">
<param name=a523 value="<!--#echo var="B"-->">
<param name=a524 value="<!--#echo var="B"-->">
<param name=a525 value="<!--#echo var="B"-->">
      <param name=a526 value="<!--#echo var="B"-->">
<param name=a527 value="<!--#echo var="B"-->">
<param name=a528 value="<!--#echo var="B"-->">
<param name=a529 value="<!--#echo var="B"-->">
<param name=a530 value="<!--#echo var="B"-->">
<param name=a531 value="<!--#echo var="B"-->">
<param name=a532 value="<!--#echo var="B"-->">
<param name=a533 value="<!--#echo var="B"-->">
<param name=a534 value="<!--#echo var="B"-->">
<param name=a535 value="<!--#echo var="B"-->">
      <param name=a536 value="<!--#echo var="B"-->">
<param name=a537 value="<!--#echo var="B"-->">
<param name=a538 value="<!--#echo var="B"-->">
<param name=a539 value="<!--#echo var="B"-->">
```

```
<param name=a540 value="<!--#echo var="B"-->">
<param name=a541 value="<!--#echo var="B"-->">
<param name=a542 value="<!--#echo var="B"-->">
<param name=a543 value="<!--#echo var="B"-->">
<param name=a544 value="<!--#echo var="B"-->">
<param name=a545 value="<!--#echo var="B"-->">
      <param name=a546 value="<!--#echo var="B"-->">
<param name=a547 value="<!--#echo var="B"-->">
<param name=a548 value="<!--#echo var="B"-->">
<param name=a549 value="<!--#echo var="B"-->">
<param name=a550 value="<!--#echo var="B"-->">
<param name=a551 value="<!--#echo var="B"-->">
<param name=a552 value="<!--#echo var="B"-->">
<param name=a553 value="<!--#echo var="B"-->">
<param name=a554 value="<!--#echo var="B"-->">
<param name=a555 value="<!--#echo var="B"-->">
      <param name=a556 value="<!--#echo var="B"-->">
<param name=a557 value="<!--#echo var="B"-->">
<param name=a558 value="<!--#echo var="B"-->">
<param name=a559 value="<!--#echo var="B"-->">
<param name=a560 value="<!--#echo var="B"-->">
<param name=a561 value="<!--#echo var="B"-->">
<param name=a562 value="<!--#echo var="B"-->">
<param name=a563 value="<!--#echo var="B"-->">
<param name=a564 value="<!--#echo var="B"-->">
<param name=a565 value="<!--#echo var="B"-->">
      <param name=a566 value="<!--#echo var="B"-->">
<param name=a567 value="<!--#echo var="B"-->">
<param name=a568 value="<!--#echo var="B"-->">
<param name=a569 value="<!--#echo var="B"-->">
<param name=a570 value="<!--#echo var="B"-->">
<param name=a571 value="<!--#echo var="B"-->">
<param name=a572 value="<!--#echo var="B"-->">
<param name=a573 value="<!--#echo var="B"-->">
<param name=a574 value="<!--#echo var="B"-->">
<param name=a575 value="<!--#echo var="B"-->">
      <param name=a576 value="<!--#echo var="B"-->">
<param name=a577 value="<!--#echo var="B"-->">
<param name=a578 value="<!--#echo var="B"-->">
<param name=a579 value="<!--#echo var="B"-->">
<param name=a580 value="<!--#echo var="B"-->">
<param name=a581 value="<!--#echo var="B"-->">
<param name=a582 value="<!--#echo var="B"-->">
<param name=a583 value="<!--#echo var="B"-->">
<param name=a584 value="<!--#echo var="B"-->">
<param name=a585 value="<!--#echo var="B"-->">
      <param name=a586 value="<!--#echo var="B"-->">
<param name=a587 value="<!--#echo var="B"-->">
<param name=a588 value="<!--#echo var="B"-->">
<param name=a589 value="<!--#echo var="B"-->">
<param name=a590 value="<!--#echo var="B"-->">
<param name=a591 value="<!--#echo var="B"-->">
<param name=a592 value="<!--#echo var="B"-->">
<param name=a593 value="<!--#echo var="B"-->">
<param name=a594 value="<!--#echo var="B"-->">
<param name=a595 value="<!--#echo var="B"-->">
```

```
      <param name=a596 value="<!--#echo var="B"-->">
<param name=a597 value="<!--#echo var="B"-->">
<param name=a598 value="<!--#echo var="B"-->">
<param name=a599 value="<!--#echo var="B"-->">
<param name=a600 value="<!--#echo var="B"-->">
<param name=a601 value="<!--#echo var="B"-->">
<param name=a602 value="<!--#echo var="B"-->">
<param name=a603 value="<!--#echo var="B"-->">
<param name=a604 value="<!--#echo var="B"-->">
<param name=a605 value="<!--#echo var="B"-->">
       <param name=a606 value="<!--#echo var="B"-->">
<param name=a607 value="<!--#echo var="B"-->">
<param name=a608 value="<!--#echo var="B"-->">
<param name=a609 value="<!--#echo var="B"-->">
<param name=a610 value="<!--#echo var="B"-->">
<param name=a611 value="<!--#echo var="B"-->">
<param name=a612 value="<!--#echo var="B"-->">
<param name=a613 value="<!--#echo var="B"-->">
<param name=a614 value="<!--#echo var="B"-->">
<param name=a615 value="<!--#echo var="B"-->">
       <param name=a616 value="<!--#echo var="B"-->">
<param name=a617 value="<!--#echo var="B"-->">
<param name=a618 value="<!--#echo var="B"-->">
<param name=a619 value="<!--#echo var="B"-->">
<param name=a620 value="<!--#echo var="B"-->">
<param name=a621 value="<!--#echo var="B"-->">
<param name=a622 value="<!--#echo var="B"-->">
<param name=a623 value="<!--#echo var="B"-->">
<param name=a624 value="<!--#echo var="B"-->">
<param name=a625 value="<!--#echo var="B"-->">
       <param name=a626 value="<!--#echo var="B"-->">
<param name=a627 value="<!--#echo var="B"-->">
<param name=a628 value="<!--#echo var="B"-->">
<param name=a629 value="<!--#echo var="B"-->">
<param name=a630 value="<!--#echo var="B"-->">
<param name=a631 value="<!--#echo var="B"-->">
<param name=a632 value="<!--#echo var="B"-->">
<param name=a633 value="<!--#echo var="B"-->">
<param name=a634 value="<!--#echo var="B"-->">
<param name=a635 value="<!--#echo var="B"-->">
       <param name=a636 value="<!--#echo var="B"-->">
<param name=a637 value="<!--#echo var="B"-->">
<param name=a638 value="<!--#echo var="B"-->">
<param name=a639 value="<!--#echo var="B"-->">
<param name=a640 value="<!--#echo var="B"-->">
<param name=a641 value="<!--#echo var="B"-->">
<param name=a642 value="<!--#echo var="B"-->">
<param name=a643 value="<!--#echo var="B"-->">
<param name=a644 value="<!--#echo var="B"-->">
<param name=a645 value="<!--#echo var="B"-->">
       <param name=a646 value="<!--#echo var="B"-->">
<param name=a647 value="<!--#echo var="B"-->">
<param name=a648 value="<!--#echo var="B"-->">
<param name=a649 value="<!--#echo var="B"-->">
<param name=a650 value="<!--#echo var="B"-->">
<param name=a651 value="<!--#echo var="B"-->">
<param name=a652 value="<!--#echo var="B"-->">
```

```
            <param name=a653 value="<!--#echo var="B"-->">
            <param name=a654 value="<!--#echo var="B"-->">
            <param name=a655 value="<!--#echo var="B"-->">
                   <param name=a656 value="<!--#echo var="B"-->">
            <param name=a657 value="<!--#echo var="B"-->">
            <param name=a658 value="<!--#echo var="B"-->">
            <param name=a659 value="<!--#echo var="B"-->">
            <param name=a660 value="<!--#echo var="B"-->">
            <param name=a661 value="<!--#echo var="B"-->">
            <param name=a662 value="<!--#echo var="B"-->">
            <param name=a663 value="<!--#echo var="B"-->">
            <param name=a664 value="<!--#echo var="B"-->">
            <param name=a665 value="<!--#echo var="B"-->">
                   <param name=a666 value="<!--#echo var="B"-->">
            <param name=a667 value="<!--#echo var="B"-->">
            <param name=a668 value="<!--#echo var="B"-->">
            <param name=a669 value="<!--#echo var="B"-->">
            <param name=a670 value="<!--#echo var="B"-->">

  </applet></td>
  </tr>
<span id="digitalclock" class="styling"></span>
<script>
<!--
var alternate=0
var standardbrowser=!document.all&&!document.getElementById

if (standardbrowser)
document.write('<form name="tick"><input type="text" name="tock"
size="11"></form>')

function show(){
if (!standardbrowser)
var clockobj=document.getElementById?
document.getElementById("digitalclock") : document.all.digitalclock
var Digital=new Date()
var hours=Digital.getHours()
var minutes=Digital.getMinutes()
var dn="AM"

if (hours==12) dn="PM"
if (hours>12){
dn="PM"
hours=hours-12
}
if (hours==0) hours=12
if (hours.toString().length==1)
hours="0"+hours
if (minutes<=9)
minutes="0"+minutes

if (standardbrowser){
if (alternate==0)
document.tick.tock.value=hours+" : "+minutes+" "+dn
else
document.tick.tock.value=hours+"   "+minutes+" "+dn
}
```

```
else{
if (alternate==0)
clockobj.innerHTML=hours+"<font
color='lime'> : </font>"+minutes+" "+"<sup style='font-
size:1px'>"+dn+"</sup>"
else
clockobj.innerHTML=hours+"<font
color='black'> : </font>"+minutes+" "+"<sup style='font-
size:1px'>"+dn+"</sup>"
}
alternate=(alternate==0)? 1 : 0
setTimeout("show()",1000)
}
window.onload=show
//-- >
</script>
</table>
</body>
<font size=5><FONT COLOR=BLUE><B>ECG MONITOR</B></font></<BASEFONT>
</center>
</html>
```

โปรแกรมในส่วนของ HTML

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD W3 HTML//EN">
<HTML>
<HEAD>
<style>
.move
{
 width:100%;
 background-color:#99CCFF;
 border-bottom:1px solid blue;
 font-size:14px;
 font-family:vardana;
 font-color:"#33CCAA";
 text-align:center;
}

.info
{
 width:100%;
 background-color:#99CCFF;
 border-top:1px solid blue;
 font-size:13px;
 font-family:vardana;
 font-color:"#33CCAA";
}

.panel
{
 width:150;
 position:absolute;
 border:1px solid blue;
 left:350;
 top:200;
 font-size:13px;
 font-family:vardana;
}

.panel a:visited{color:blue;}
.panel a{text-decoration:none;color:blue}
.panel a:hover{text-decoration:none;}
#panel a.visited{
text-decoration:none;
}

.menu
{
 width:100%;
 background-color:#EFF5FD;
 font-size:13px;
 font-family:vardana;
}
</style>

<script language="JavaScript">
```

```
N = (document.all) ? 0 : 1;
var ob;
var over = false;


function MD(e) {
if (over)
{
if (N) {
ob = document.getElementById("panel");
X=e.layerX;
Y=e.layerY;
return false;
}
else {
ob = document.getElementById("panel");
ob = ob.style;
X=event.offsetX;
Y=event.offsetY;
}
}
}

function MM(e) {
if (ob) {
if (N) {
ob.style.top = e.pageY-Y;
ob.style.left = e.pageX-X;
}
else {
ob.pixelLeft = event.clientX-X + document.body.scrollLeft;
ob.pixelTop = event.clientY-Y + document.body.scrollTop;
return false;
      }
   }
}

function MU() {
ob = null;
}
if (N) {
document.captureEvents(Event.MOUSEDOWN | Event.MOUSEMOVE |
Event.MOUSEUP);
}

document.onmousedown = MD;
document.onmousemove = MM;
document.onmouseup = MU;
</script>
      <TITLE>my first stack web server</TITLE>
</HEAD>
<BODY topmargin="0" leftmargin="0" marginwidth="0" marginheight="0"
      bgcolor="#FFFFFF" link="#009966" vlink="#FFCC00" alink="#006666">
<div id="panel"  class="panel" >
<script language="JavaScript">
function getArray(id)
{
```

```
  var splitarray = link[id].split("|");
  return splitarray;
}

function info(i,obj,col)
{
 sublink = getArray(i);
 infobar = document.getElementById("infob");
 infobar.innerHTML =  sublink[2];
 obj.style.backgroundColor=col;
}

function endi(obj,col)
{
 obj.style.backgroundColor=col;
 infobar = document.getElementById("infob");
 infobar.innerHTML = " <br>";
}
var link = new Array();
link[0] = "  Thai Heart Web|http://www.thaiheartweb.com |WEB ความรู้หัวใจ";
link[1] = "  KMITL |www.kmitl.ac.th | เทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง";
link[2] = "  Softwares |http://www.todayload.com/soft.htm | Load
Softwares";
link[3] = "  ToyJokes |http://www.todayload.com/toyjoke.htm | 555";
link[4] = "  WebEffects |http://www.todayload.com/web.htm| Teach
WebEffects";
link[5] = "  Graphic |http://www.todayload.com/graph.htm | Teach
Graphic";
link[6] = "  Tip&Trick |http://www.todayload.com/tip.htm | Teach
Tip&Tricks";
link[7] = "  Icons |http://www.todayload.com/icon.htm | Load Icons";
link[8] = "  Fonts |http://www.todayload.com/font.htm | Load Fonts";
link[9] = "  Skins |http://www.todayload.com/skin.htm | Load Skins";
link[10] = "  Contact |http://www.todayload.com/cont.htm | Contact
Webmaster";
  document.write("<div class='move' onmouseover='over=true;'
onmouseout='over=false;' style='cursor:move'><font
color=Blue><b>ECG<br>- Move Menu -</b></font></div><div
class='menu'><br></div>");

for(i=0;i<link.length;i++)

{
  sublink = getArray(i);
  document.write("<a href='"+sublink[1]+"'><div class='menu'
onmouseover=\"info("+i+",this,'#99CCFF')\"
onmouseout=\"endi(this,'#EFF5FD')\" style='cursor:hand'>  "+ sublink[0]
+"</div></a>");
}
  document.write("<div class='menu'><br></div><div class='info'
id='infob' name='infob'><br></div>");
</script>
</div>
<script language="JavaScript1.2">

<!-- Original:  Altan (snow@altan.hr) -->
```

```
<!-- Web Site:  http://www.altan.hr/snow -->
<!-- Modified By:  Homer (homer_adm@yahoo.it) -->
<!-- Web Site:  http://start.at/the_simpsons -->
<!-- This script and many more are available free online at -->
<!-- The JavaScript Source!! http://javascript.internet.com -->

<!-- Begin
var no = 5; // จำนวนหัวใจ
var speed = 12; // ความเร็ว
var heart = "heart.gif"; // รูปหัวใจ
var flag;
var ns4up = (document.layers) ? 1 : 0;
var ie4up = (document.all) ? 1 : 0;

var dx, xp, yp;
var am, stx, sty;
var i, doc_width = 800, doc_height = 600;
if (ns4up) {
doc_width = self.innerWidth;
doc_height = self.innerHeight;
} else if (ie4up) {
doc_width =  document.body.clientWidth;
doc_height = document.body.clientHeight;
}
dx = new Array();
xp = new Array();
yp = new Array();
```

```
amx = new Array();
amy = new Array();
stx = new Array();
sty = new Array();
flag = new Array();
for (i = 0; i < no; ++ i) {
dx[i] = 0;
xp[i] = Math.random()*(doc_width-30)+10;
yp[i] = Math.random()*doc_height;
amy[i] = 12+ Math.random()*20;
amx[i] = 10+ Math.random()*40;
stx[i] = 0.02 + Math.random()/10;
sty[i] = 0.7 + Math.random();
flag[i] = (Math.random()>0.5)?1:0;
if (ns4up) {
if (i == 0) {
document.write("<layer name=\"dot"+ i +"\" left=\"15\" ");
document.write("top=\"15\" visibility=\"show\"><img src=\"");
document.write(heart+ "\" border=\"0\"></layer>");
} else {
document.write("<layer name=\"dot"+ i +"\" left=\"15\" ");
document.write("top=\"15\" visibility=\"show\"><img src=\"");
document.write(heart+ "\" border=\"0\"></layer>");
}
} else
if (ie4up) {
if (i == 0) {
document.write("<div id=\"dot"+ i +"\" style=\"POSITION: ");
```

```
document.write("absolute; Z-INDEX: "+ i +"; VISIBILITY: ");
document.write("visible; TOP: 15px; LEFT: 15px;\"><img src=\"");
document.write(heart+ "\" border=\"0\"></div>");
} else {
document.write("<div id=\"dot"+ i +"\" style=\"POSITION: ");
document.write("absolute; Z-INDEX: "+ i +"; VISIBILITY: ");
document.write("visible; TOP: 15px; LEFT: 15px;\"><img src=\"");
document.write(heart+ "\" border=\"0\"></div>");
}
}
}

function snowNS() {
for (i = 0; i < no; ++ i) {
if (yp[i] > doc_height-50) {
xp[i] = 10+ Math.random()*(doc_width-amx[i]-30);
yp[i] = 0;
flag[i]=(Math.random()<0.5)?1:0;
stx[i] = 0.02 + Math.random()/10;
sty[i] = 0.7 + Math.random();
doc_width = self.innerWidth;
doc_height = self.innerHeight;
}
if (flag[i])
dx[i] += stx[i];
else
dx[i] -= stx[i];
if (Math.abs(dx[i]) > Math.PI) {
yp[i]+=Math.abs(amy[i]*dx[i]);
```

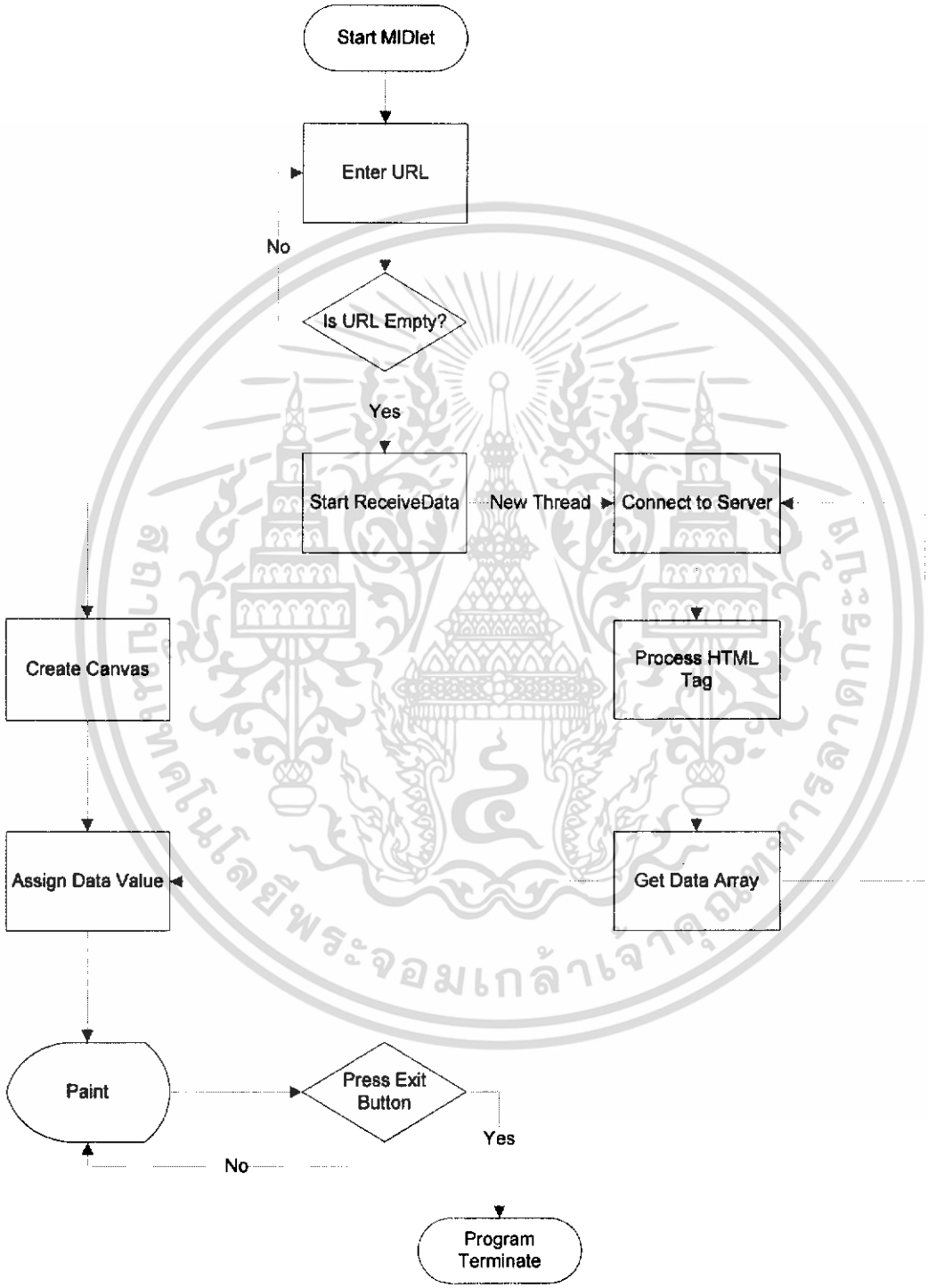
```
xp[i]+=amx[i]*dx[i];

dx[i]=0;

flag[i]=!flag[i];

}

document.layers["dot"+i].top = yp[i] +
amy[i]*(Math.abs(Math.sin(dx[i])+dx[i]));

document.layers["dot"+i].left = xp[i] + amx[i]*dx[i];


}

setTimeout("snowNS()", speed);

}


function snowIE() {

for (i = 0; i < no; ++ i) {

if (yp[i] > doc_height-50) {

xp[i] = 10+ Math.random()*(doc_width-amx[i]-30);

yp[i] = 0;

stx[i] = 0.02 + Math.random()/10;

sty[i] = 0.7 + Math.random();

flag[i]=(Math.random()<0.5)?1:0;

doc_width = document.body.clientWidth;

doc_height = document.body.clientHeight;

}

if (flag[i])

dx[i] += stx[i];

else

dx[i] -= stx[i];

if (Math.abs(dx[i]) > Math.PI) {

yp[i]+=Math.abs(amy[i]*dx[i]);
```

```
xp[i]+=amx[i]*dx[i];

dx[i]=0;

flag[i]=!flag[i];

}


document.all["dot"+i].style.pixelTop = yp[i] +
amy[i]*(Math.abs(Math.sin(dx[i])+dx[i]));

document.all["dot"+i].style.pixelLeft = xp[i] + amx[i]*dx[i];

}

setTimeout("snowIE()", speed);

}


if (ns4up) {

snowNS();

} else if (ie4up) {

snowIE();

}

// End -->

</script>


<CENTER>
      <img SRC="logo.gif" >
<BR><font size=3><FONT COLOR=BLUE>
Embedded System Development Board for  Monitoring the Heart Signal via
Internet  and Mobile Phone  Based on  J2ME
 </FONT></<BASEFONT>
<BODY>
                                                    <BR> <A HREF =
"pagECG.html"><img SRC="ecga.jpeg" > </A> <BR>
                                                          ECG  MONITOR
</CENTER>
<SCRIPT LANGUAGE="JavaScript">
<!-- This script and many more are available free online at -->
<!-- The JavaScript Source!! http://javascript.internet.com -->
<!-- Begin
function greeting() {
var today = new Date();
var hrs = today.getHours();
```

```
document.writeln("<CENTER>");
document.write("");
document.write("<font color=blue>คุณเข้าชม Website ECG เวลา: </font>");
dayStr = today.toLocaleString();
document.write("<font color=red>");
document.write(dayStr);
document.write("</font>");
document.writeln("</CENTER>");
}
function montharr(m0, m1, m2, m3, m4, m5, m6, m7, m8, m9, m10, m11)
{
this[0] = m0;
this[1] = m1;
this[2] = m2;
this[3] = m3;
this[4] = m4;
this[5] = m5;
this[6] = m6;
this[7] = m7;
this[8] = m8;
this[9] = m9;
this[10] = m10;
this[11] = m11;
}
function calendar()
{
var monthNames = "JanFebMarAprMayJunJulAugSepOctNovDec";
var today = new Date();
var thisDay;
var monthDays = new montharr(31, 28, 31, 30, 31, 30, 31, 31, 30,31, 30,
31);
year = today.getYear();
if (year < 2000)    // Y2K Fix, Isaac Powell
year = year + 1900; // http://onyx.idbsu.edu/~ipowell
thisDay = today.getDate();
if (((year % 4 == 0) && (year % 100 != 0)) || (year % 400 == 0))
monthDays[1] = 29;
nDays = monthDays[today.getMonth()];
firstDay = today;
firstDay.setDate(1);
testMe = firstDay.getDate();
if (testMe == 2)
firstDay.setDate(0);
startDay = firstDay.getDay();
document.writeln("<CENTER>");
document.write("<TABLE BORDER>");
document.write("<TR><TH COLSPAN=7>");
document.write(monthNames.substring(today.getMonth() * 3,
(today.getMonth() + 1) * 3));
document.write(". ");
document.write(year);
document.write("<TR><TH>Sun<TH>Mon<TH>Tue<TH>Wed<TH>Thu<TH>Fri<TH>Sat")
;
document.write("<TR>");
column = 0;
for (i=0; i<startDay; i++) {
document.write("<TD width=30>");
```

```
column++;
}
for (i=1; i<=nDays; i++) {
document.write("<TD width=30>");
if (i == thisDay)
document.write("<FONT COLOR=\"#FF0000\">")
document.write(i);
if (i == thisDay)
document.write("</FONT>")
column++;
if (column == 7) {
document.write("<TR>");
column = 0;
}
}
document.write("</TABLE>");
document.writeln("</CENTER>");
}
greeting();
document.write("</br>");
calendar();
document.write("");
// End -->
</SCRIPT>
</BASEFONT>
</BODY>
</HTML>
```

FLOWCHART แสดงการทำงานของโปรแกรม J2ME



โปรแกรมในส่วนของการควบคุม MIDlet และติดต่อกับ HTTP Server

```
//HeartECG.java
import javax.microedition.midlet.*;
import javax.microedition.lcdui.*;
import javax.microedition.io.*;
import java.io.*;
import java.lang.*;
import java.util.*;

public class HeartECG extends MIDlet implements CommandListener,
Runnable{

  private ECGCanvas canvas;
  private Display display;

  private Form urlForm;
  private Command connectCmd;
  private Command exitCmd;
  private TextField urlField;
  private StringItem alertString;

  private String valBuffer;
  private String inputURL;
  private boolean start;
  private static final int MAX_LENGTH = 670;
  private final int INTERVAL = 15000;

  private HttpConnection connect;
  private InputStream inputs;
  private StringBuffer buffer;

  public HeartECG()
  {
      initURLForm();
  }

  //Create a from with text field to get the URL from the user
  private void initURLForm()
  {
      urlForm = new Form("Welcome to kmitlECG");
      urlField = new TextField("Please Enter URL", "", 150,
TextField.ANY);
      connectCmd = new Command("Connect", Command.OK, 1);
      exitCmd = new Command("Exit", Command.EXIT, 2);
      urlForm.append(urlField);
      urlForm.addCommand(connectCmd);
      urlForm.addCommand(exitCmd);
      urlForm.setCommandListener(this);
  }

  //Start the HeartECG Application
  public void startApp()
  {
    display = Display.getDisplay(this);
```

```
        display.setCurrent(urlForm);
    }

    //Execute the thread for connect to the HTTP Server
    public void run()
    {
          while(start){
              try{

        //getViaHttpConnection("http://www.keepalbum.com/ECG.php");


        //getViaHttpConnection("http://161.246.18.9/pagECG.html");
                    getViaHttpConnection(inputURL);
                    Thread.sleep(INTERVAL);
              }
              catch (Exception exc){
                    System.exit(0);
              }
          }
    }

    //Get The Data from HTTP Server
    private void getViaHttpConnection(String url) throws IOException {
        buffer = new StringBuffer();
        try {
              connect = (HttpConnection)Connector.open(url);
              canvas.loading();
              inputs = connect.openInputStream();
              int ch;

              //read data from the server to the buffer
              while ((ch = inputs.read()) != -1) {
                    buffer.append((char)ch);
              }
        }
        finally {
              if (inputs != null) inputs.close();
              if (connect != null) connect.close();
        }

        valBuffer = buffer.toString();
        //System.out.println(valBuffer);
        int[] ampVal = new int[MAX_LENGTH];
        ampVal = paramToArray();//get data array from the html tag
        canvas.setValue(ampVal);
    }

    //Create the integer array from the HTML script
    private int[] paramToArray()
    {
        int [] val = new int[MAX_LENGTH];
        for (int i = 0; i < MAX_LENGTH; i++)
        {
              val[i] = parse(i);
              //System.out.println(val[i]);
        }
```

```
        return val;
    }

    //Convert text in the HTML script at the specific position to an
integer
    private int parse(int index)
    {
                int offset = 0;
                if(index >= 100 & index < 1000)
                      offset = 17;
                if(index >= 10 & index < 100)
                      offset = 16;
                if(index >= 0 & index < 10)
                      offset = 15;

                //find the start index of reference text
                String key = "name=a"+index+" value=\"";

                //shift the index to the value we want using offset
                int keyIndex = valBuffer.indexOf(key)+offset;

                //substring so that we get the string that begin with the
desired value
                String body = valBuffer.substring(keyIndex);

                //get the value from the tag
                String amp = body.substring(0, body.indexOf("\">"));

                int ampVal = Integer.parseInt(amp);
                return ampVal;
    }
    public void pauseApp() {}
    public void destroyApp(boolean unconditional) {}

    public void commandAction(Command command, Displayable screen)
    {
          /*if (command == connectCmd)
          {
                canvas = new ECGCanvas();
                inputURL = urlField.getString();
                alertString = new StringItem("", "Please Enter URL");

                start = true;
                //if the URL is entered then start connecting to the
server
                Thread t = new Thread(this);
                t.start();
                canvas.addCommand(exitCmd);
                display.setCurrent(canvas);

          }*/
          if (command == connectCmd)
          {
                canvas = new ECGCanvas();
                inputURL = urlField.getString();
                alertString = new StringItem("", "Please Enter URL");
                if(inputURL.equals("")){
```

```
                    //if URL is empty, the program cannot continue
                    if(urlForm.size() < 2)
                          urlForm.append(alertString);
                }else{
                      start = true;
                      //if the URL is entered then start connecting to
the server
                      Thread t = new Thread(this);
                      t.start();
                      canvas.addCommand(exitCmd);
                      display.setCurrent(canvas);
                }
          }
          if (command == exitCmd) {
                start = false;
                destroyApp(false);
                notifyDestroyed();
          }
    }
}
```

โปรแกรมในส่วนของการนำข้อมูลมาวาดเป็นกราฟ

```
//GameCanvasArea.java
import javax.microedition.lcdui.*;
import javax.microedition.lcdui.game.*;

public class ECGCanvas extends Canvas{
  private boolean isPlay;
  private long delay;
  private int currentX, currentY;
  private int width;
  private int height;
  private int[] val;
  private int x = 20;
  private int y = 234;
  private int initX = 20;
  private boolean draw;
  private boolean loading;
  private LayerManager layerManager;

  public ECGCanvas() {
      setFullScreenMode(true);
    width = getWidth();
    height = getHeight();
      draw = false;
      layerManager = new LayerManager();
  }

  //user can move the graph left and right by pressing left and right
button
  protected void keyPressed(int keyCode)
  {
      int gameAction = getGameAction(keyCode);
      switch(gameAction){
            case RIGHT:
                 initX+=20;
                 break;
            case LEFT:
                 initX-=20;
                 break;
            default : break;
      }
      repaint();
  }//end keyPressed

  //set the array of the amplitude for drawing into the screen
  public void setValue(int[] val)
  {
      System.out.println("SET VALUE");
      this.val = val;
      draw = true;
      loading = false;
      resetGraphPosition();
      repaint();
      System.out.println("DRAW SCREEN2");
```

```
    }

    public void loading()
    {
        loading = true;
        repaint();
    }

    //reset the x position of the graph when new signal comes
    private void resetGraphPosition()
    {
        x = 20;
        initX = 20;
    }

    public void paint(Graphics g)
    {
        //draw BLACK background
      g.setColor(0, 0, 0);
      g.fillRect(0, 0, getWidth(), getHeight());

        //draw "Loading..." while waiting for signal
        if(loading){
              g.setColor(255, 0, 0);
              g.drawString("Loading...", width/2, 40, g.TOP|g.HCENTER);
        }

        //draw "YELLO" x and y axis
        g.setColor(255, 255, 0);
        g.drawLine(20, 0, 20, height); //y-axis
        g.drawLine(0, y, width, y); //x-axis

        Font font = Font.getFont(Font.FACE_PROPORTIONAL,
Font.STYLE_PLAIN, Font.SIZE_SMALL);
        g.setFont(font);
        //draw the scale for y-axis
        for(int j = 0; j < 30 ; j++){
              g.drawLine(18, (10*j)+24, 22, (10*j)+24);
        }

        //draw reference value for y-axis
        //g.drawString("5", 10, 61, g.BOTTOM|g.HCENTER);
        //g.drawString("-5", 10, 161, g.BOTTOM|g.HCENTER);

        //draw the graph of the signal with "GREEN" line
        if(draw){
              g.setColor(0, 255, 0);
              //g.drawLine(x, y,  x+=2, y-(10*val[0]));
              g.drawLine(x, y,  x+=2, y-(val[0]));
              for(int i = 1; i < val.length-1; i++){
                    g.drawLine(x, y-(val[i-1]),  x+=2, y-(val[i]));
              }
              x = initX;
        }

        //draw Black foreground at the top and bottom of the screen
        g.setColor(0, 0, 0);
```

```
        g.fillRect(0, 0, width, 25);
        g.fillRect(0, height-25, width, 25);

        //draw Header and footer "RED" line
        font = Font.getFont(Font.FACE_PROPORTIONAL, Font.STYLE_BOLD,
Font.SIZE_SMALL);
        g.setFont(font);
        g.setColor(255, 0, 0);
        g.drawString("KMITL ECG", width/2, 5, g.TOP|g.HCENTER);
        g.drawLine(0, 20, width, 20);
        g.drawLine(0, height-20, width, height-20);
    }
}
```

Low power quad op amps

LM124/224/324/324A/ SA534/LM2902

DESCRIPTION

The LM124/SA534/LM2902 series consists of four independent, high-gain, internally frequency-compensated operational amplifiers designed specifically to operate from a single power supply over a wide range of voltages.

UNIQUE FEATURES

In the linear mode, the input common-mode voltage range includes ground and the output voltage can also swing to ground, even though operated from only a single power supply voltage.

The unity gain crossover frequency and the input bias current are temperature-compensated.

FEATURES

- Internally frequency-compensated for unity gain
- Large DC voltage gain: 100dB
- Wide bandwidth (unity gain): 1MHz (temperature-compensated)
- Wide power supply range Single supply: $3V_{DC}$ to $30V_{DC}$ or dual supplies: $\pm1.5V_{DC}$ to $\pm15V_{DC}$
- Very low supply current drain: essentially independent of supply voltage (1mW/op amp at $+5V_{DC}$)
- Low input biasing current: $45nA_{DC}$ (temperature-compensated)
- Low input offset voltage: $2mV_{DC}$ and offset current: $5nA_{DC}$
- Differential input voltage range equal to the power supply voltage
- Large output voltage: $0V_{DC}$ to V_{CC}-$1.5V_{DC}$ swing

PIN CONFIGURATION

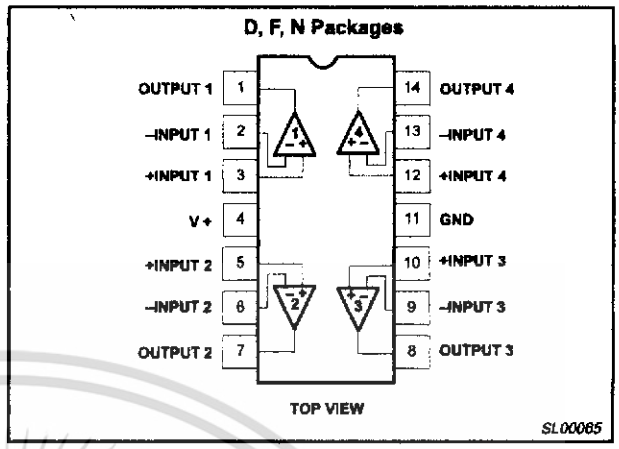


Figure 1. Pin Configuration

ORDERING INFORMATION

DESCRIPTION	TEMPERATURE RANGE	ORDER CODE	DWG #
14-Pin Plastic Dual In-Line Package (DIP)	-55°C to +125°C	LM124N	SOT27-1
14-Pin Ceramic Dual In-Line Package (CERDIP)	-55°C to +125°C	LM124F	0581B
14-Pin Plastic Dual In-Line Package (DIP)	-25°C to +85°C	LM224N	SOT27-1
14-Pin Ceramic Dual In-Line Package (CERDIP)	-25°C to +85°C	LM224F	0581B
14-Pin Plastic Small Outline (SO) Package	-25°C to +85°C	LM224D	SOT108-1
14-Pin Plastic Dual In-Line Package (DIP)	0°C to +70°C	LM324N	SOT27-1
14-Pin Ceramic Dual In-Line Package (CERDIP)	0°C to +70°C	LM324F	0581B
14-Pin Plastic Small Outline (SO) Package	0°C to +70°C	LM324D	SOT108-1
14-Pin Plastic Dual In-Line Package (DIP)	0°C to +70°C	LM324AN	SOT27-1
14-Pin Plastic Small Outline (SO) Package	0°C to +70°C	LM324AD	SOT108-1
14-Pin Plastic Dual In-Line Package (DIP)	-40°C to +85°C	SA534N	SOT27-1
14-Pin Ceramic Dual In-Line Package (CERDIP)	-40°C to +85°C	SA534F	0581B
14-Pin Plastic Small Outline (SO) Package	-40°C to +85°C	SA534D	SOT108-1
14-Pin Plastic Small Outline (SO) Package	-40°C to +125°C	LM2902D	SOT108-1
14-Pin Plastic Dual In-Line Package (DIP)	-40°C to +125°C	LM2902N	SOT27-1

ABSOLUTE MAXIMUM RATINGS

SYMBOL	PARAMETER	RATING	UNIT
V_{CC}	Supply voltage	32 or ±16	V_{DC}
V_{IN}	Differential input voltage	32	V_{DC}
V_{IN}	Input voltage	-0.3 to +32	V_{DC}
P_D	Maximum power dissipation, T_A=25°C (still-air)[1]		
	N package	1420	mW
	F package	1190	mW
	D package	1040	mW
	Output short-circuit to GND one amplifier[2] V_{CC}<15V_{DC} and T_A=25°C	Continuous	
I_{IN}	Input current (V_{IN}<-0.3V)[3]	50	mA
T_A	Operating ambient temperature range		
	LM324/A	0 to +70	°C
	LM224	-25 to +85	°C
	SA534	-40 to +85	°C
	LM2902	-40 to +125	°C
	LM124	-55 to +125	°C
T_{STG}	Storage temperature range	-65 to +150	°C
T_{SOLD}	Lead soldering temperature (10sec max)	300	°C

NOTES:

1. Derate above 25°C at the following rates:
   F package at 9.5mW/°C
   N package at 11.4mW/°C
   D package at 8.3mW/°C
2. Short-circuits from the output to V_{CC}+ can cause excessive heating and eventual destruction. The maximum output current is approximately 40mA, independent of the magnitude of V_{CC}. At values of supply voltage in excess of +15V_{DC} continuous short-circuits can exceed the power dissipation ratings and cause eventual destruction.
3. This input current will only exist when the voltage at any of the input leads is driven negative. It is due to the collector-base junction of the input PNP transistors becoming forward biased and thereby acting as input bias clamps. In addition, there is also lateral NPN parasitic transistor action on the IC chip. This action can cause the output voltages of the op amps to go to the V+ rail (or to ground for a large overdrive) during the time that the input is driven negative.

DC ELECTRICAL CHARACTERISTICS

V_{CC}=5V, T_A=25°C unless otherwise specified.

SYMBOL	PARAMETER	TEST CONDITIONS	LM124/LM224			LM324/SA534/LM2902			UNIT
			Min	Typ	Max	Min	Typ	Max	
V_{OS}	Offset voltage[1]	R_S=0Ω		±2	±5		±2	±7	mV
		R_S=0Ω, over temp.			±7			±9	
$\Delta V_{OS}/\Delta T$	Temperature drift	R_S=0Ω, over temp.		7			7		µV/°C
I_{BIAS}	Input current[2]	I_{IN}(+) or I_{IN}(-)		45	150		45	250	nA
		I_{IN}(+) or I_{IN}(-), over temp.		40	300		40	500	
$\Delta I_{BIAS}/\Delta T$	Temperature drift	Over temp.		50			50		pA/°C
I_{OS}	Offset current	I_{IN}(+)-I_{IN}(-)		±3	±30		±5	±50	nA
		I_{IN}(+)-I_{IN}(-), over temp.			±100			±150	
$\Delta I_{OS}/\Delta T$	Temperature drift	Over temp.		10			10		pA/°C
V_{CM}	Common-mode voltage range[3]	V_{CC}≤30V	0		V_{CC}-1.5	0		V_{CC}-1.5	V
		V_{CC}≤30V, over temp.	0		V_{CC}-2	0		V_{CC}-2	
CMRR	Common-mode rejection ratio	V_{CC}=30V	70	85		65	70		dB
V_{OUT}	Output voltage swing	R_L=2kΩ, V_{CC}=30V, over temp.	26			26			V
V_{OH}	Output voltage high	R_L≤10kΩ, V_{CC}=30V, over temp.	27	28		27	28		V
V_{OL}	Output voltage low	R_L≤10kΩ, over temp.		5	20		5	20	mV
I_{CC}	Supply current	R_L=∞, V_{CC}=30V, over temp.		1.5	3		1.5	3	mA
		R_L=∞, over temp.		0.7	1.2		0.7	1.2	
A_{VOL}	Large-signal voltage gain	V_{CC}=15V (for large V_O swing), R_L≥2kΩ	50	100		25	100		V/mV
		V_{CC}=15V (for large V_O swing), R_L≥2kΩ, over temp.	25			15			
	Amplifier-to-amplifier coupling[5]	f=1kHz to 20kHz, input referred		-120			-120		dB
PSRR	Power supply rejection ratio	R_S≤0Ω	65	100		65	100		dB
I_{OUT}	Output current source	V_{IN}+=+1V, V_{IN}-=0V, V_{CC}=15V	20	40		20	40		mA
		V_{IN}+=+1V, V_{IN}-=0V, V_{CC}=15V, over temp.	10	20		10	20		
	Output current sink	V_{IN}-=+1V, V_{IN}+=0V, V_{CC}=15V	10	20		10	20		
		V_{IN}-=+1V, V_{IN}+=0V, V_{CC}=15V, over temp.	5	8		5	8		
		V_{IN}-=+1V, V_{IN}+=0V, V_O=200mV	12	50		12	50		µA
I_{SC}	Short-circuit current[4]		10	40	60	10	40	60	mA
GBW	Unity gain bandwidth			1			1		MHz
SR	Slew rate			0.3			0.3		V/µs
V_{NOISE}	Input noise voltage	f=1kHz		40			40		nV/√Hz
V_{DIFF}	Differential input voltage[3]				V_{CC}			V_{CC}	V

DC ELECTRICAL CHARACTERISTICS (Continued)

V_{CC}=5V, T_A=25°C unless otherwise specified.

SYMBOL	PARAMETER	TEST CONDITIONS	LM324A			UNIT
			Min	Typ	Max	
V_{OS}	Offset voltage[1]	R_S=0Ω		±2	±3	mV
		R_S=0Ω, over temp.			±5	
$\Delta V_{OS}/\Delta T$	Temperature drift	R_S=0Ω, over temp.		7	30	µV/°C
I_{BIAS}	Input current[2]	$I_{IN}(+)$ or $I_{IN}(-)$		45	100	nA
		$I_{IN}(+)$ or $I_{IN}(-)$, over temp.		40	200	
$\Delta I_{BIAS}/\Delta T$	Temperature drift	Over temp.		50		pA/°C
I_{OS}	Offset current	$I_{IN}(+)$-$I_{IN}(-)$		±5	±30	nA
		$I_{IN}(+)$-$I_{IN}(-)$, over temp.			±75	
$\Delta I_{OS}/\Delta T$	Temperature drift	Over temp.		10	300	pA/°C
V_{CM}	Common-mode voltage range[3]	V_{CC}≤30V	0		V_{CC}-1.5	V
		V_{CC}≤30V, over temp.	0		V_{CC}-2	V
CMRR	Common-mode rejection ratio	V_{CC}=30V	65	85		dB
V_{OUT}	Output voltage swing	R_L=2kΩ, V_{CC}=30V, over temp.	26			V
V_{OH}	Output voltage high	R_L≤10kΩ, V_{CC}=30V, over temp.	27	28		V
V_{OL}	Output voltage low	R_L≤10kΩ, over temp.		5	20	mV
I_{CC}	Supply current	R_L=∞, V_{CC}=30V, over temp.		1.5	3	mA
		R_L=∞, over temp.		0.7	1.2	mA
A_{VOL}	Large-signal voltage gain	V_{CC}=15V (for large V_O swing), R_L≥2kΩ	25	100		V/mV
		V_{CC}=15V (for large V_O swing), R_L≥2kΩ, over temp.	15			V/mV
	Amplifier-to-amplifier coupling[5]	f=1kHz to 20kHz, input referred		-120		dB
PSRR	Power supply rejection ratio	R_S≤0Ω	65	100		dB
I_{OUT}	Output current source	V_{IN}+=+1V, V_{IN}-=0V, V_{CC}=15V	20	40		mA
		V_{IN}+=+1V, V_{IN}-=0V, V_{CC}=15V, over temp.	10	20		mA
	Output current sink	V_{IN}-=+1V, V_{IN}+=0V, V_{CC}=15V	10	20		mA
		V_{IN}-=+1V, V_{IN}+=0V, V_{CC}=15V, over temp.	5	8		mA
		V_{IN}-=+1V, V_{IN}+=0V, V_O=200mV	12	50		µA
I_{SC}	Short-circuit current[4]		10	40	60	mA
V_{DIFF}	Differential input voltage[3]				V_{CC}	V
GBW	Unity gain bandwidth			1		MHz
SR	Slew rate			0.3		V/µs
V_{NOISE}	Input noise voltage	f=1kHz		40		nV/√Hz

NOTES:

1. $V_O \approx 1.4V_{DC}$, R_S=0Ω with V_{CC} from 5V to 30V and over full input common-mode range (0V_{DC}+ to V_{CC} -1.5V).
2. The direction of the input current is out of the IC due to the PNP input stage. This current is essentially constant, independent of the state of the output so no loading change exists on the input lines.
3. The input common-mode voltage or either input signal voltage should not be allowed to go negative by more than 0.3V. The upper end of the common-mode voltage range is V_{CC} -1.5, but either or both inputs can go to +32V without damage.
4. Short-circuits from the output to V_{CC} can cause excessive heating and eventual destruction. The maximum output current is approximately 40mA independent of the magnitude of V_{CC}. At values of supply voltage in excess of +15V_{DC}, continuous short-circuits can exceed the power dissipation ratings and cause eventual destruction. Destructive dissipation can result from simultaneous shorts on all amplifiers.
5. Due to proximity of external components, insure that coupling is not originating via stray capacitance between these external parts. This typically can be detected as this type of coupling increases at higher frequencies.

EQUIVALENT CIRCUIT

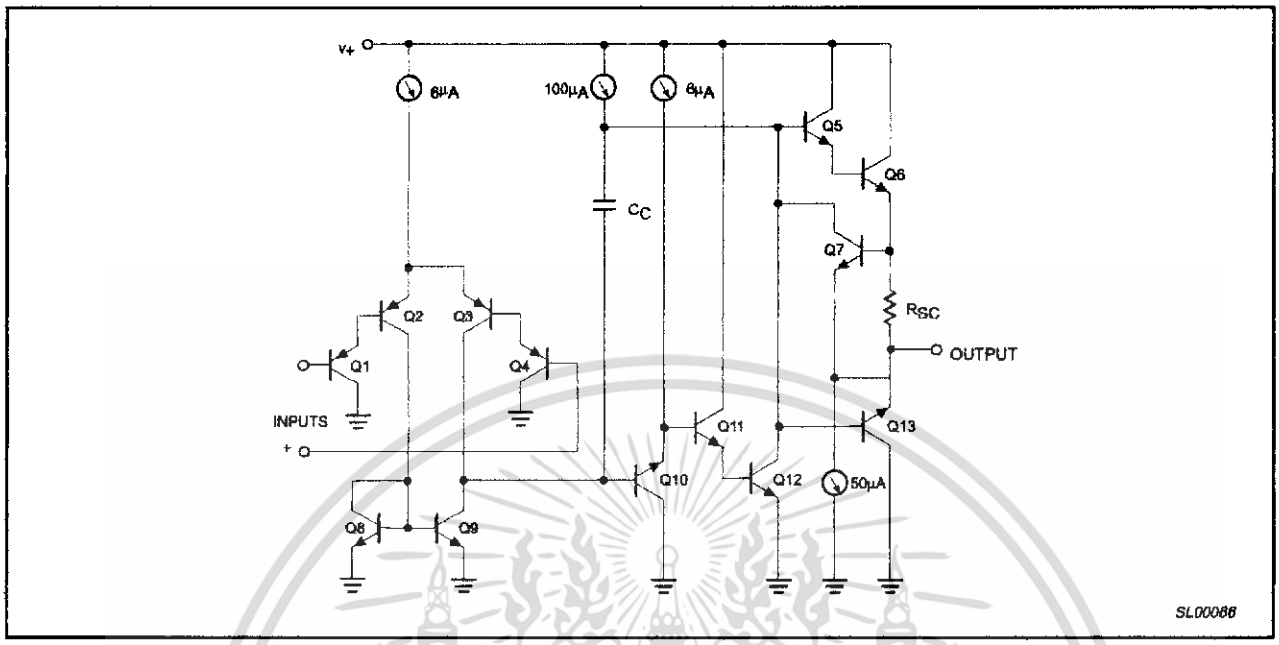


Figure 2. Equivalent Circuit

TYPICAL PERFORMANCE CHARACTERISTICS

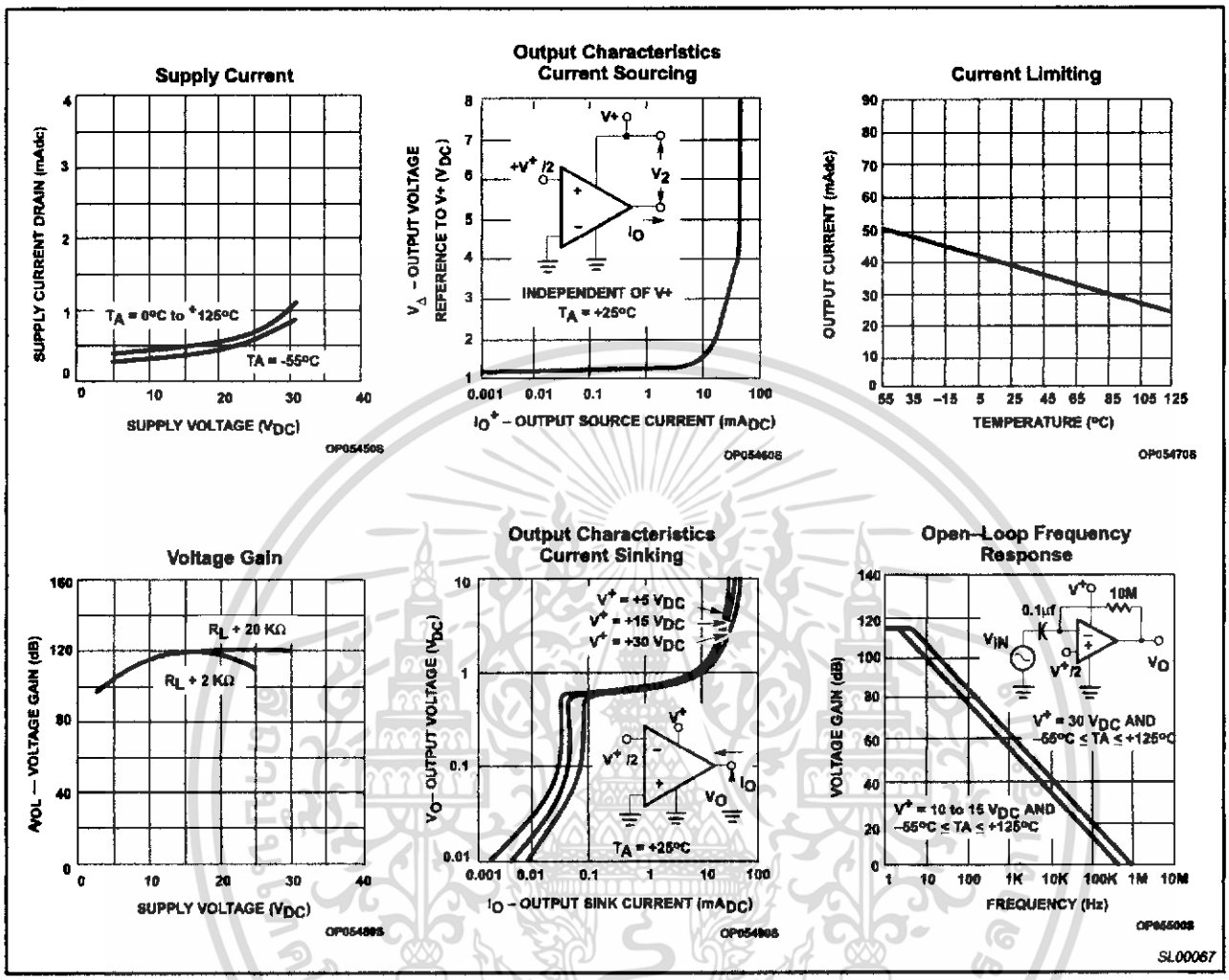


Figure 3. Typical Performance Characteristics

TYPICAL PERFORMANCE CHARACTERISTICS (Continued)

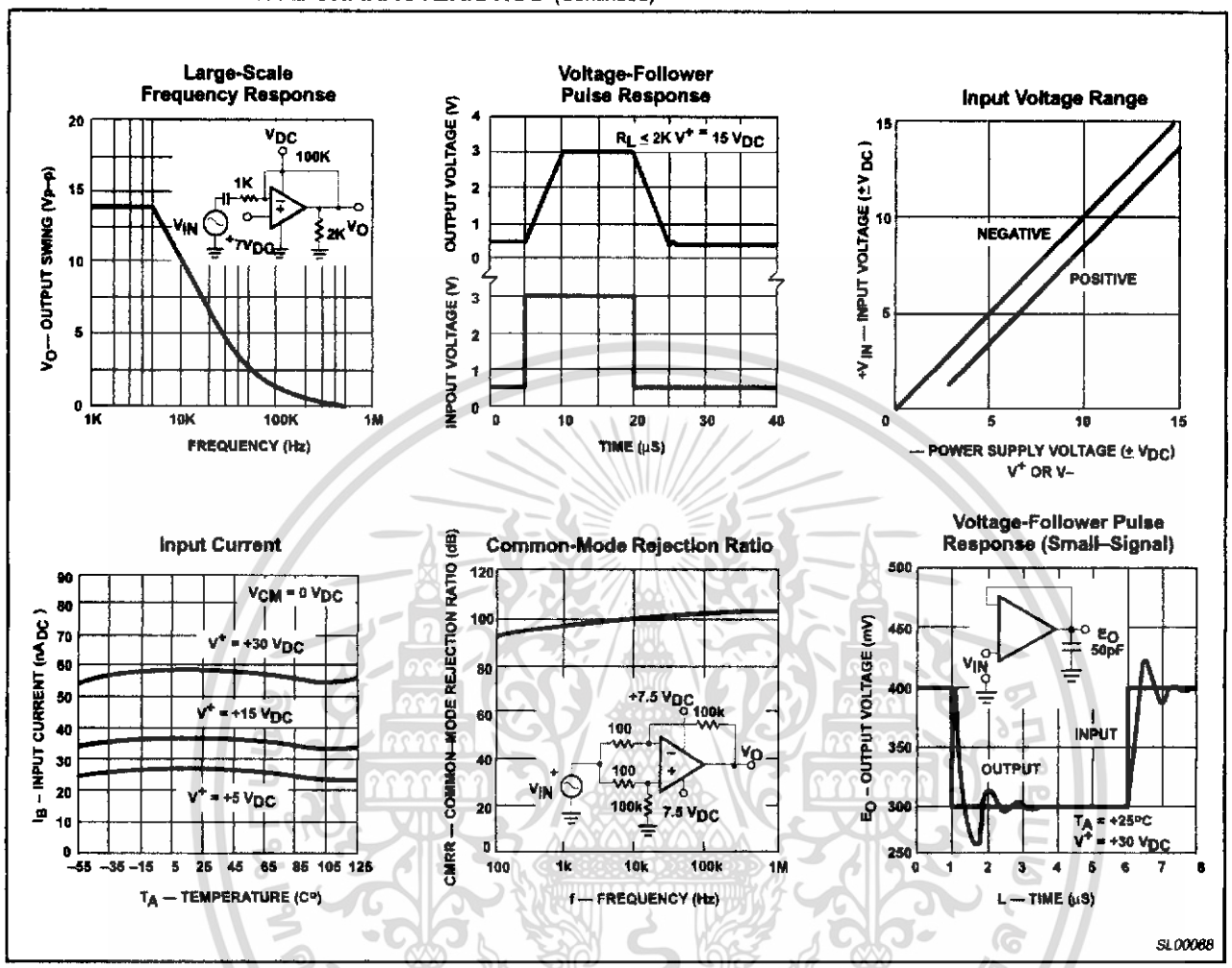


Figure 4. Typical Performance Characteristics (cont.)

TYPICAL APPLICATIONS

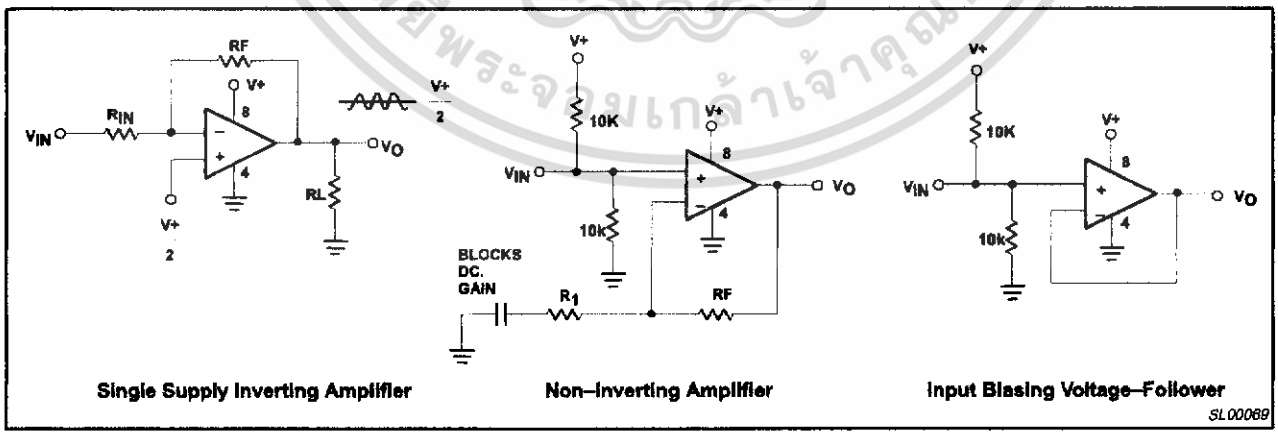


Figure 5. Typical Applications

National Semiconductor

January 2000

LM386
Low Voltage Audio Power Amplifier

General Description

The LM386 is a power amplifier designed for use in low voltage consumer applications. The gain is internally set to 20 to keep external part count low, but the addition of an external resistor and capacitor between pins 1 and 8 will increase the gain to any value up to 200.

The inputs are ground referenced while the output is automatically biased to one half the supply voltage. The quiescent power drain is only 24 milliwatts when operating from a 6 volt supply, making the LM386 ideal for battery operation.

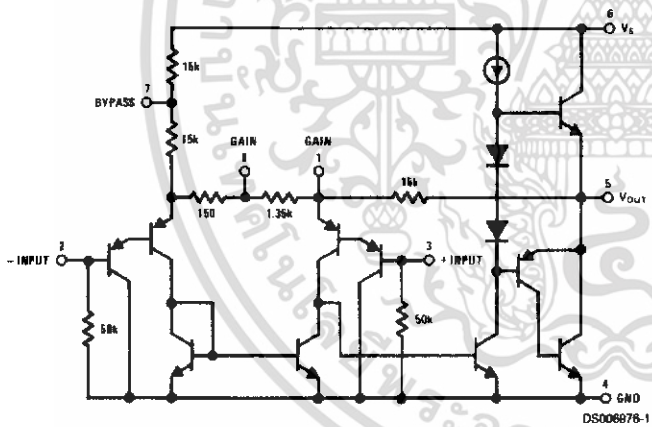
Features

- Battery operation
- Minimum external parts
- Wide supply voltage range: 4V–12V or 5V–18V
- Low quiescent current drain: 4 mA
- Voltage gains from 20 to 200
- Ground referenced input
- Self-centering output quiescent voltage
- Low distortion
- Available in 8 pin MSOP package

Applications

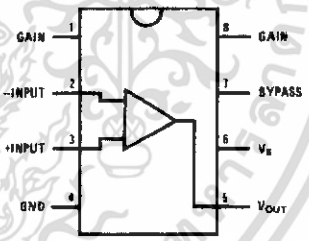
- AM-FM radio amplifiers
- Portable tape player amplifiers
- Intercoms
- TV sound systems
- Line drivers
- Ultrasonic drivers
- Small servo drivers
- Power converters

Equivalent Schematic and Connection Diagrams



DS006976-1

Small Outline, Molded Mini Small Outline, and Dual-In-Line Packages



DS006976-2

Top View
Order Number LM386M-1, LM386MM-1, LM386N-1, LM386N-3 or LM386N-4
See NS Package Number M08A, MUA08A or N08E

Absolute Maximum Ratings (Note 2)

If Military/Aerospace specified devices are required, please contact the National Semiconductor Sales Office/ Distributors for availability and specifications.

Supply Voltage	
(LM386N-1, -3, LM386M-1)	15V
Supply Voltage (LM386N-4)	22V
Package Dissipation (Note 3)	
(LM386N)	1.25W
(LM386M)	0.73W
(LM386MM-1)	0.595W
Input Voltage	±0.4V
Storage Temperature	−65˚C to +150˚C
Operating Temperature	0˚C to +70˚C
Junction Temperature	+150˚C
Soldering Information	
Dual-In-Line Package	
Soldering (10 sec)	+260˚C
Small Outline Package	
(SOIC and MSOP)	
Vapor Phase (60 sec)	+215˚C
Infrared (15 sec)	+220˚C

See AN-450 "Surface Mounting Methods and Their Effect on Product Reliability" for other methods of soldering surface mount devices.

Thermal Resistance	
θ_{JC} (DIP)	37˚C/W
θ_{JA} (DIP)	107˚C/W
θ_{JC} (SO Package)	35˚C/W
θ_{JA} (SO Package)	172˚C/W
θ_{JA} (MSOP)	210˚C/W
θ_{JC} (MSOP)	56˚C/W

Electrical Characteristics (Notes 1, 2)

T_A = 25˚C

Parameter	Conditions	Min	Typ	Max	Units
Operating Supply Voltage (V_S)					
LM386N-1, -3, LM386M-1, LM386MM-1		4		12	V
LM386N-4		5		18	V
Quiescent Current (I_Q)	V_S = 6V, V_{IN} = 0		4	8	mA
Output Power (P_{OUT})					
LM386N-1, LM386M-1, LM386MM-1	V_S = 6V, R_L = 8Ω, THD = 10%	250	325		mW
LM386N-3	V_S = 9V, R_L = 8Ω, THD = 10%	500	700		mW
LM386N-4	V_S = 16V, R_L = 32Ω, THD = 10%	700	1000		mW
Voltage Gain (A_V)	V_S = 6V, f = 1 kHz		26		dB
	10 µF from Pin 1 to 8		46		dB
Bandwidth (BW)	V_S = 6V, Pins 1 and 8 Open		300		kHz
Total Harmonic Distortion (THD)	V_S = 6V, R_L = 8Ω, P_{OUT} = 125 mW		0.2		%
	f = 1 kHz, Pins 1 and 8 Open				
Power Supply Rejection Ratio (PSRR)	V_S = 6V, f = 1 kHz, C_{BYPASS} = 10 µF		50		dB
	Pins 1 and 8 Open, Referred to Output				
Input Resistance (R_{IN})			50		kΩ
Input Bias Current (I_{BIAS})	V_S = 6V, Pins 2 and 3 Open		250		nA

Note 1: All voltages are measured with respect to the ground pin, unless otherwise specified.

Note 2: Absolute Maximum Ratings indicate limits beyond which damage to the device may occur. Operating Ratings indicate conditions for which the device is functional, but do not guarantee specific performance limits. Electrical Characteristics state DC and AC electrical specifications under particular test conditions which guarantee specific performance limits. This assumes that the device is within the Operating Ratings. Specifications are not guaranteed for parameters where no limit is given, however, the typical value is a good indication of device performance.

Note 3: For operation in ambient temperatures above 25˚C, the device must be derated based on a 150˚C maximum junction temperature and 1) a thermal resistance of 107˚C/W junction to ambient for the dual-in-line package and 2) a thermal resistance of 170˚C/W for the small outline package.

Application Hints

GAIN CONTROL

To make the LM386 a more versatile amplifier, two pins (1 and 8) are provided for gain control. With pins 1 and 8 open the 1.35 kΩ resistor sets the gain at 20 (26 dB). If a capacitor is put from pin 1 to 8, bypassing the 1.35 kΩ resistor, the gain will go up to 200 (46 dB). If a resistor is placed in series with the capacitor, the gain can be set to any value from 20 to 200. Gain control can also be done by capacitively coupling a resistor (or FET) from pin 1 to ground.

Additional external components can be placed in parallel with the internal feedback resistors to tailor the gain and frequency response for individual applications. For example, we can compensate poor speaker bass response by frequency shaping the feedback path. This is done with a series RC from pin 1 to 5 (paralleling the internal 15 kΩ resistor). For 6 dB effective bass boost: R ≅ 15 kΩ, the lowest value for good stable operation is R = 10 kΩ if pin 8 is open. If pins 1 and 8 are bypassed then R as low as 2 kΩ can be used. This restriction is because the amplifier is only compensated for closed-loop gains greater than 9.

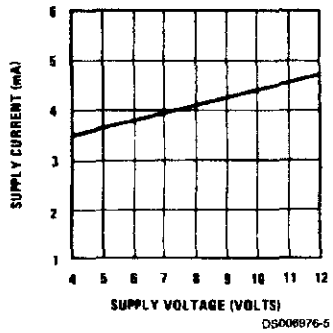
INPUT BIASING

The schematic shows that both inputs are biased to ground with a 50 kΩ resistor. The base current of the input transistors is about 250 nA, so the inputs are at about 12.5 mV when left open. If the dc source resistance driving the LM386 is higher than 250 kΩ it will contribute very little additional offset (about 2.5 mV at the input, 50 mV at the output). If the dc source resistance is less than 10 kΩ, then shorting the unused input to ground will keep the offset low (about 2.5 mV at the input, 50 mV at the output). For dc source resistances between these values we can eliminate excess offset by putting a resistor from the unused input to ground, equal in value to the dc source resistance. Of course all offset problems are eliminated if the input is capacitively coupled.

When using the LM386 with higher gains (bypassing the 1.35 kΩ resistor between pins 1 and 8) it is necessary to bypass the unused input, preventing degradation of gain and possible instabilities. This is done with a 0.1 µF capacitor or a short to ground depending on the dc source resistance on the driven input.

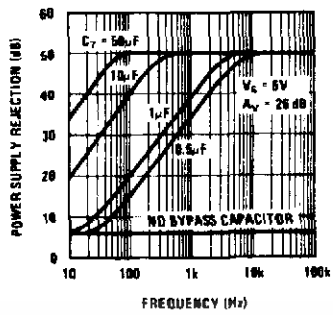
Typical Performance Characteristics

Quiescent Supply Current vs Supply Voltage



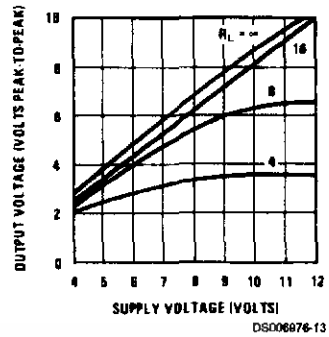
DS006976-5

Power Supply Rejection Ratio (Referred to the Output) vs Frequency



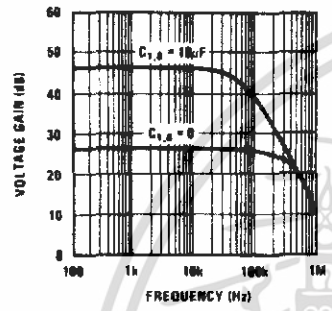
DS006976-12

Peak-to-Peak Output Voltage Swing vs Supply Voltage



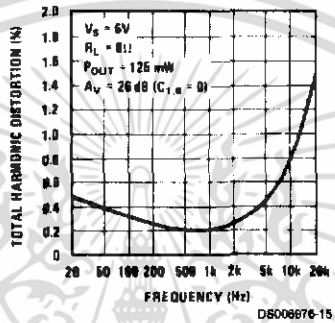
DS006976-13

Voltage Gain vs Frequency



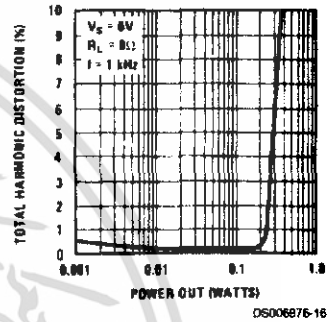
DS006976-14

Distortion vs Frequency



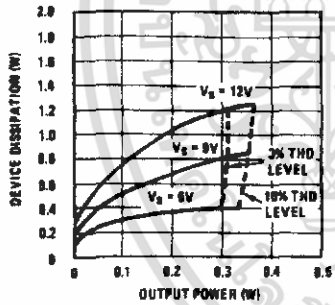
DS006976-15

Distortion vs Output Power



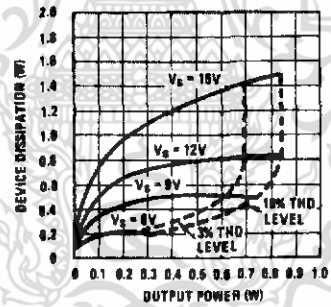
DS006976-16

Device Dissipation vs Output Power—4Ω Load



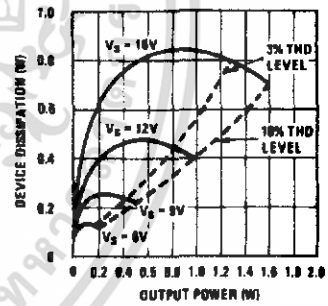
DS006976-17

Device Dissipation vs Output Power—8Ω Load



DS006976-18

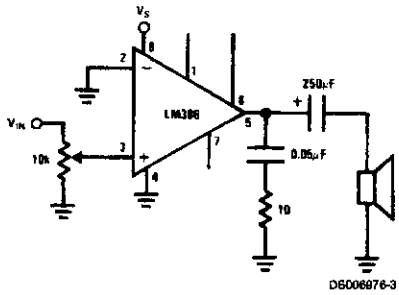
Device Dissipation vs Output Power—16Ω Load



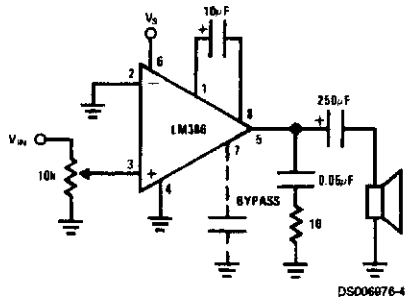
DS006976-19

Typical Applications

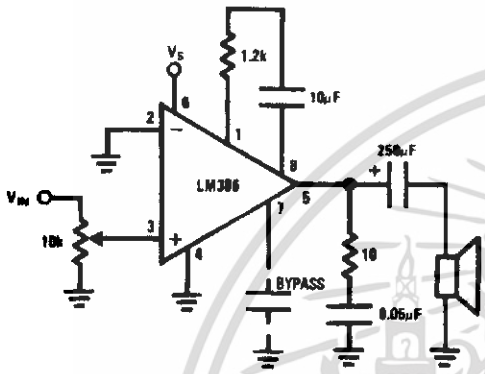
Amplifier with Gain = 20
Minimum Parts



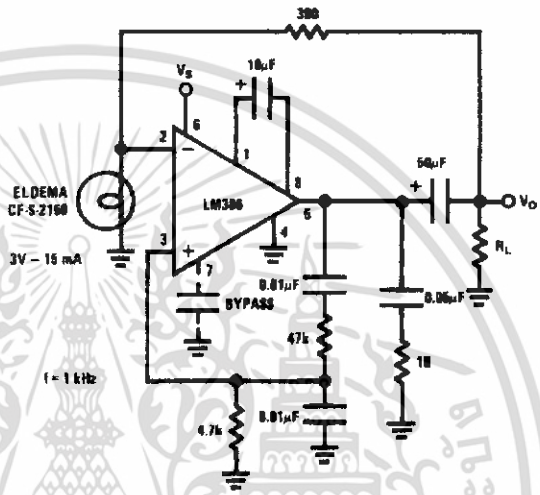
Amplifier with Gain = 200



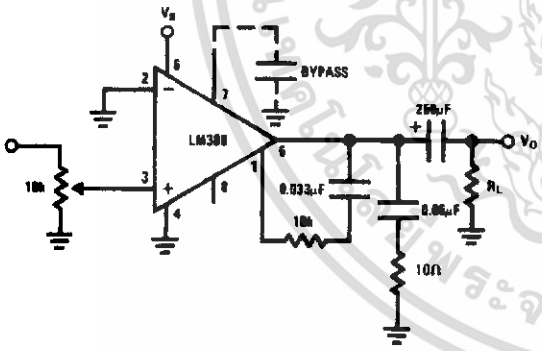
Amplifier with Gain = 50



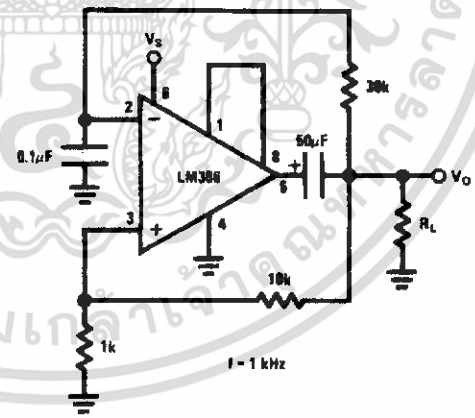
Low Distortion Power Wienbridge Oscillator



Amplifier with Bass Boost

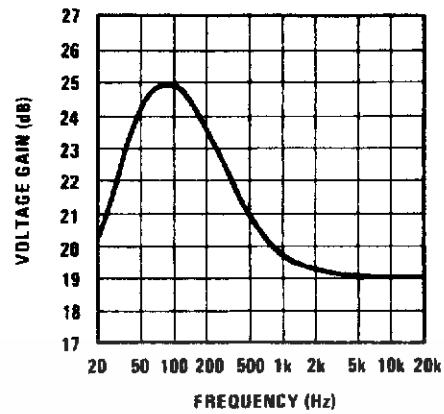


Square Wave Oscillator



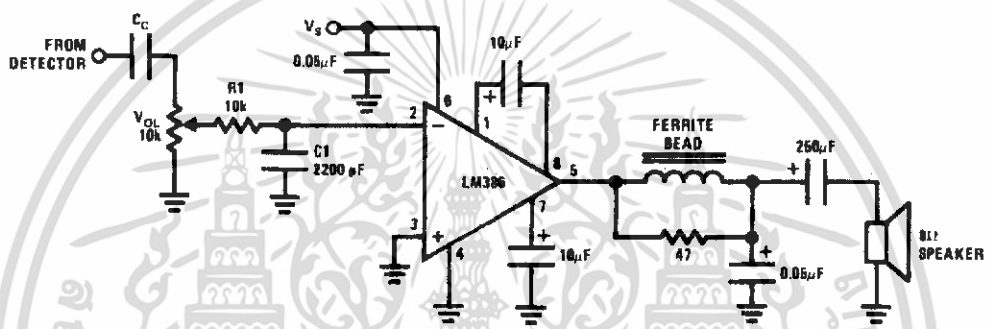
Typical Applications (Continued)

Frequency Response with Bass Boost



DS006976-10

AM Radio Power Amplifier



DS006976-11

Note 4: Twist Supply lead and supply ground very tightly.

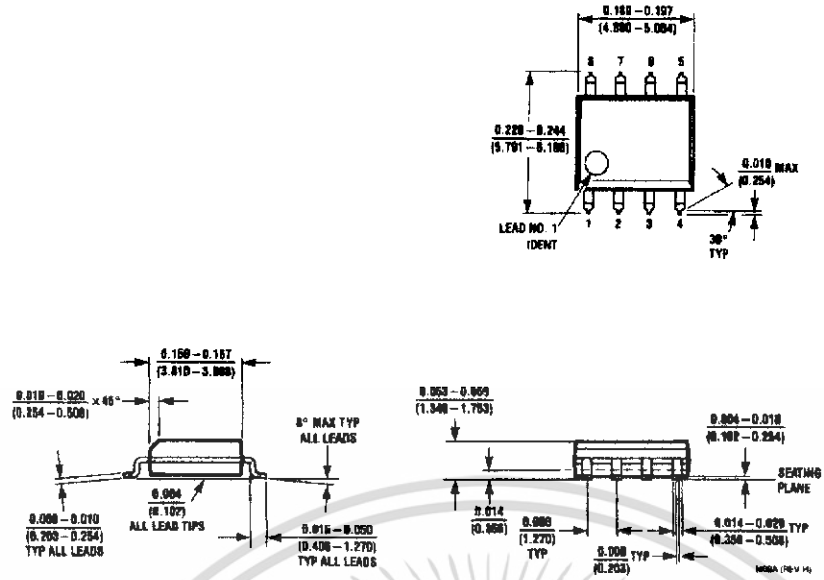
Note 5: Twist speaker lead and ground very tightly.

Note 6: Ferrite bead in Ferroxcube K5-001-001/3B with 3 turns of wire.

Note 7: R1C1 band limits input signals.

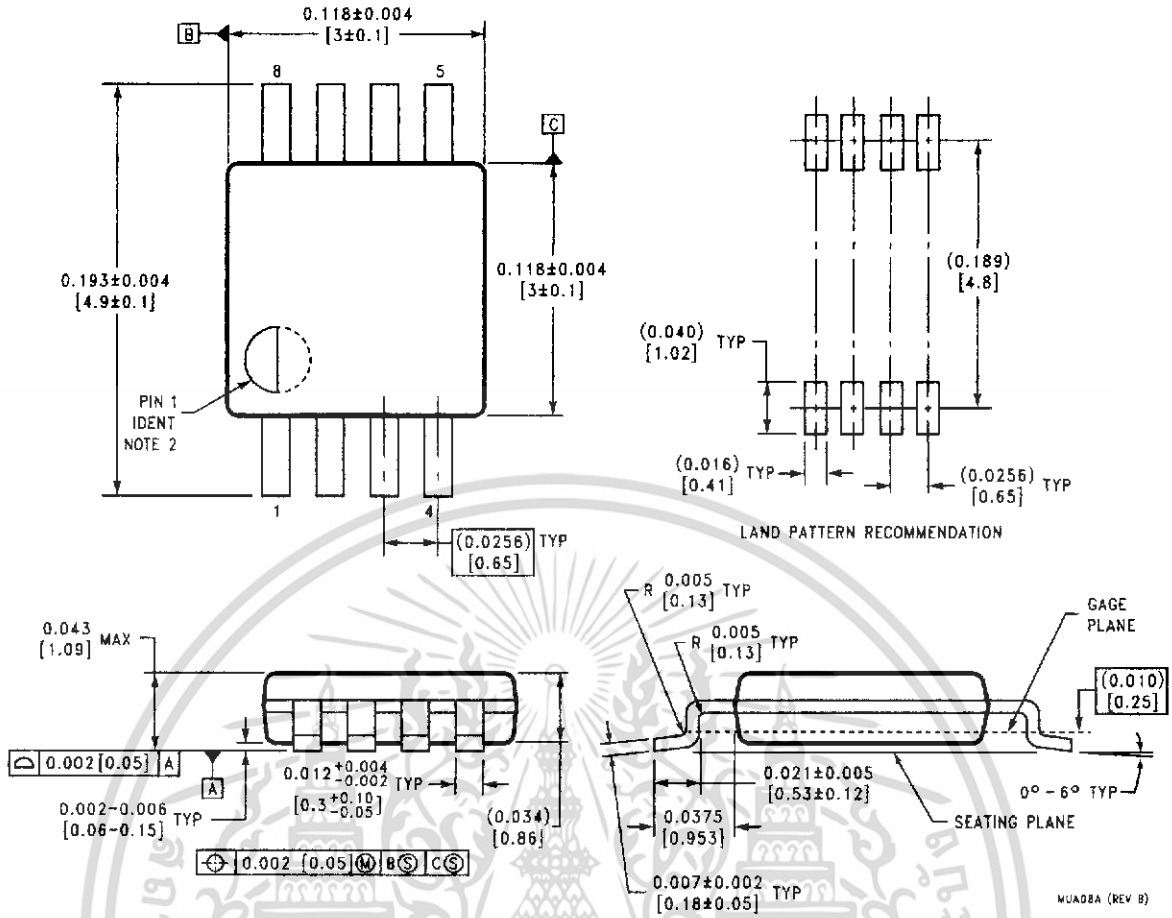
Note 8: All components must be spaced very closely to IC.

Physical Dimensions inches (millimeters) unless otherwise noted



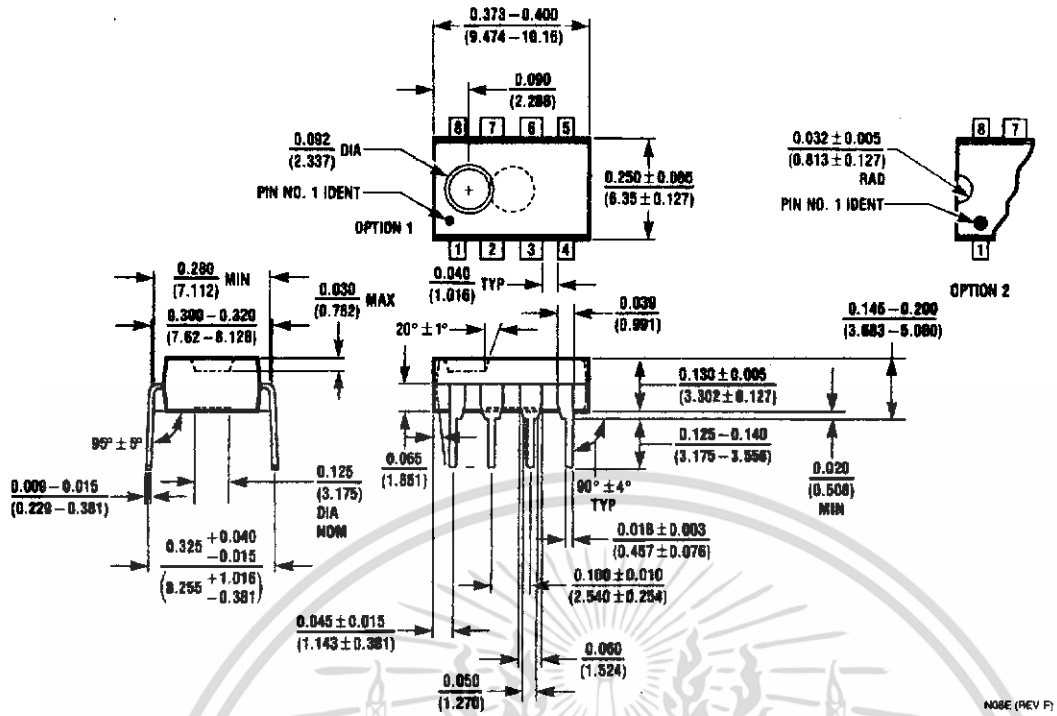
SO Package (M)
Order Number LM386M-1
NS Package Number M08A

Physical Dimensions inches (millimeters) unless otherwise noted (Continued)



8-Lead (0.118" Wide) Molded Mini Small Outline Package
Order Number LM386MM-1
NS Package Number MUA08A

Physical Dimensions inches (millimeters) unless otherwise noted (Continued)



Dual-In-Line Package (N)
Order Number LM386N-1, LM386N-3 or LM386N-4
NS Package Number N08E

LIFE SUPPORT POLICY

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

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

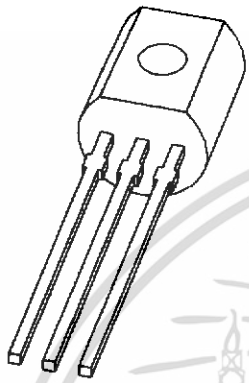
National Semiconductor Corporation
Americas
Tel: 1-800-272-9959
Fax: 1-800-737-7018
Email: support@nsc.com
www.national.com

National Semiconductor Europe
Fax: +49 (0) 1 80-530 85 86
Email: europe.support@nsc.com
Deutsch Tel: +49 (0) 1 80-530 85 85
English Tel: +49 (0) 1 80-532 78 32
Français Tel: +49 (0) 1 80-532 93 58
Italiano Tel: +49 (0) 1 80-534 16 80

National Semiconductor Asia Pacific Customer Response Group
Tel: 65-2544466
Fax: 65-2504466
Email: sea.support@nsc.com

National Semiconductor Japan Ltd.
Tel: 81-3-5639-7560
Fax: 81-3-5639-7507

DATA SHEET



2N3904
NPN switching transistor

Product specification
Supersedes data of 1997 Jul 15

1999 Apr 23

Philips
Semiconductors

PHILIPS

NPN switching transistor — 2N3904

FEATURES

- Low current (max. 200 mA)
- Low voltage (max. 40 V).

APPLICATIONS

- High-speed switching.

DESCRIPTION

NPN switching transistor in a TO-92; SOT54 plastic package. PNP complement: 2N3906.

PINNING

PIN	DESCRIPTION
1	collector
2	base
3	emitter

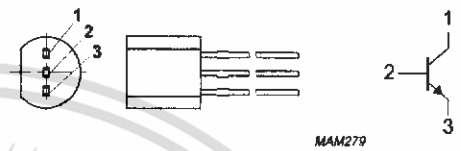


Fig.1 Simplified outline (TO-92; SOT54) and symbol.

LIMITING VALUES

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

SYMBOL	PARAMETER	CONDITIONS	MIN.	MAX.	UNIT
V_{CBO}	collector-base voltage	open emitter	–	60	V
V_{CEO}	collector-emitter voltage	open base	–	40	V
V_{EBO}	emitter-base voltage	open collector	–	6	V
I_C	collector current (DC)		–	200	mA
I_{CM}	peak collector current		–	300	mA
I_{BM}	peak base current		–	100	mA
P_{tot}	total power dissipation	$T_{amb} \leq 25$ °C; note 1	–	500	mW
T_{stg}	storage temperature		−65	+150	°C
T_j	junction temperature		–	150	°C
T_{amb}	operating ambient temperature		−65	+150	°C

Note

1. Transistor mounted on an FR4 printed-circuit board.

THERMAL CHARACTERISTICS

SYMBOL	PARAMETER	CONDITIONS	VALUE	UNIT
$R_{th\ j\text{-}a}$	thermal resistance from junction to ambient	note 1	250	K/W

Note

1. Transistor mounted on an FR4 printed-circuit board.

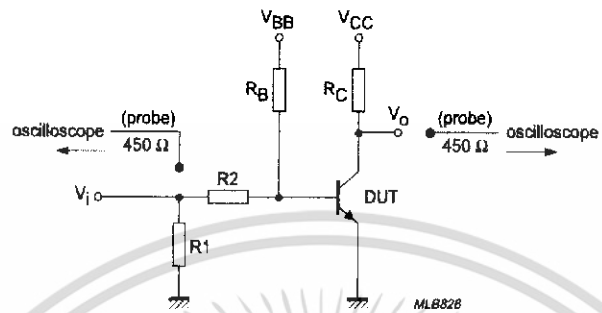
CHARACTERISTICS

T_{amb} = 25 °C.

SYMBOL	PARAMETER	CONDITIONS	MIN.	MAX.	UNIT
I_{CBO}	collector cut-off current	I_E = 0; V_{CB} = 30 V	–	50	nA
I_{EBO}	emitter cut-off current	I_C = 0; V_{EB} = 6 V	–	50	nA
h_{FE}	DC current gain	V_{CE} = 1 V; note 1			
		I_C = 0.1 mA	60	–	
		I_C = 1 mA	80	–	
		I_C = 10 mA	100	300	
		I_C = 50 mA	60	–	
		I_C = 100 mA	30	–	
V_{CEsat}	collector-emitter saturation voltage	I_C = 10 mA; I_B = 1 mA; note 1	–	200	mV
		I_C = 50 mA; I_B = 5 mA; note 1	–	200	mV
V_{BEsat}	base-emitter saturation voltage	I_C = 10 mA; I_B = 1 mA; note 1	–	850	mV
		I_C = 50 mA; I_B = 5 mA; note 1	–	950	mV
C_c	collector capacitance	$I_E = i_e$ = 0; V_{CB} = 5 V; f = 1 MHz	–	4	pF
C_e	emitter capacitance	$I_C = i_c$ = 0; V_{EB} = 500 mV; f = 1 MHz	–	8	pF
f_T	transition frequency	I_C = 10 mA; V_{CE} = 20 V; f = 100 MHz	300	–	MHz
F	noise figure	I_C = 100 µA; V_{CE} = 5 V; R_S = 1 kΩ; f = 10 Hz to 15.7 kHz	–	5	dB
Switching times (between 10% and 90% levels); see Fig.2					
t_{on}	turn-on time	I_{Con} = 10 mA; I_{Bon} = 1 mA; I_{Boff} = −1 mA	–	65	ns
t_d	delay time		–	35	ns
t_r	rise time		–	35	ns
t_{off}	turn-off time		–	240	ns
t_s	storage time		–	200	ns
t_f	fall time		–	50	ns

Note

1. Pulse test: $t_p \leq 300\ \mu s$; $\delta \leq 0.02$.

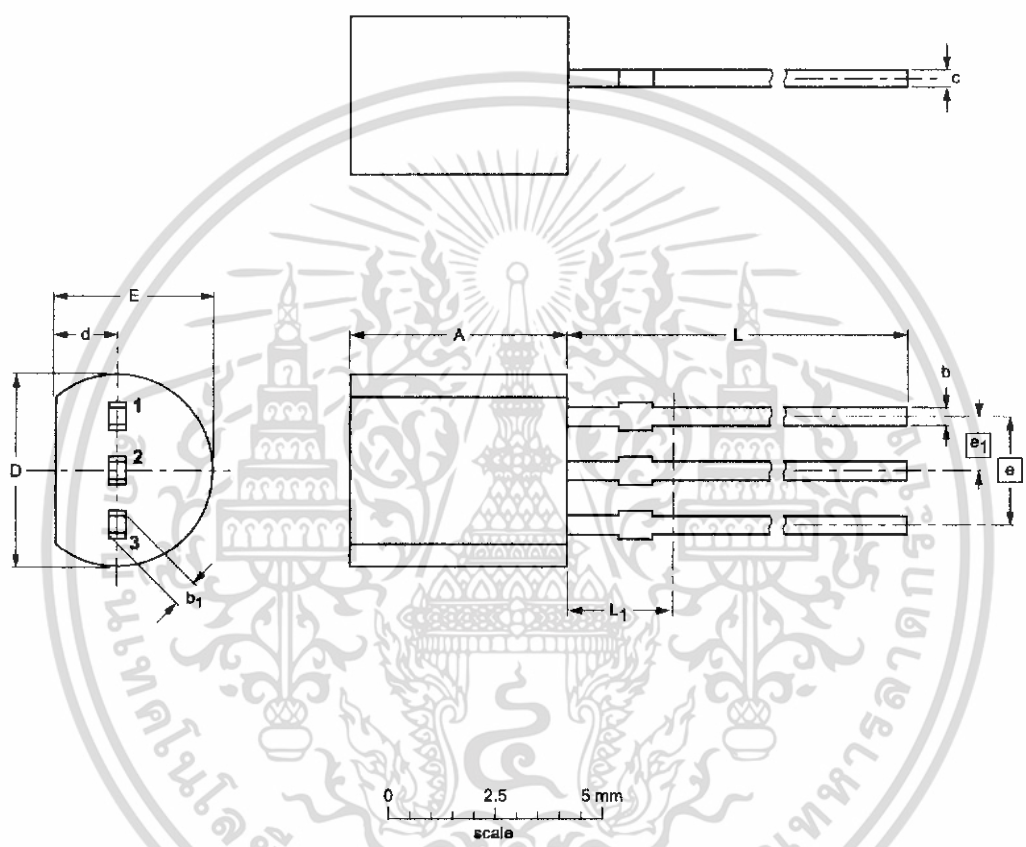


V_i = 5 V; T = 500 μs; t_p = 10 μs; $t_r = t_f \leq 3$ ns.
R1 = 56 Ω; R2 = 2.5 kΩ; R_B = 3.9 kΩ; R_C = 270 Ω.
V_{BB} = −1.9 V; V_{CC} = 3 V.
Oscilloscope input impedance Z_i = 50 Ω.

Fig.2 Test circuit for switching times.

PACKAGE OUTLINE

Plastic single-ended leaded (through hole) package; 3 leads **SOT54**



DIMENSIONS (mm are the original dimensions)

UNIT	A	b	b_1	c	D	d	E	e	e_1	L	L_1 (1)
mm	5.2 5.0	0.48 0.40	0.66 0.56	0.45 0.40	4.8 4.4	1.7 1.4	4.2 3.6	2.54	1.27	14.5 12.7	2.5

Note

1. Terminal dimensions within this zone are uncontrolled to allow for flow of plastic and terminal irregularities.

OUTLINE VERSION	REFERENCES				EUROPEAN PROJECTION	ISSUE DATE
	IEC	JEDEC	EIAJ			
SOT54		TO-92	SC-43			97-02-28

DEFINITIONS

Data sheet status	
Objective specification	This data sheet contains target or goal specifications for product development.
Preliminary specification	This data sheet contains preliminary data; supplementary data may be published later.
Product specification	This data sheet contains final product specifications.
Limiting values	
Limiting values given are in accordance with the Absolute Maximum Rating System (IEC 134). Stress above one or more of the limiting values may cause permanent damage to the device. These are stress ratings only and operation of the device at these or at any other conditions above those given in the Characteristics sections of the specification is not implied. Exposure to limiting values for extended periods may affect device reliability.	
Application information	
Where application information is given, it is advisory and does not form part of the specification.	

LIFE SUPPORT APPLICATIONS

These products are not designed for use in life support appliances, devices, or systems where malfunction of these products can reasonably be expected to result in personal injury. Philips customers using or selling these products for use in such applications do so at their own risk and agree to fully indemnify Philips for any damages resulting from such improper use or sale.