ระบบแสดงคลื่นไฟฟ้าหัวใจบนไมโครคอมพิวเตอร์

(Micro Computer Based EKG Monitor)

โดย

นายพีรพงษ์	วัฒนเกษมธรรม	35103194
นายสมคิด	ฤทธิ์คง	35103206
นายสฤษฏ์	วงศ์ว่องการ	35103211

บริญญานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรบริญญาวิศวกรรมศาสตร์บัณฑิต

สาขา วิศวกรรมอิเล็คทรอนิคส์

สถาบันเทคโนโลยีพระจอมเกล้า เจ้าคุณทหารลาดกระบัง

ปีการศึกษา 2537

ปริญญานิพนธ์ปีการศึกษา 2537

ภาควิชา อิเล็คทรอนิคส์

คณะวิศวกรรมศาสตร์ สถาบันเทคโนโลยีพระจอมเกล้า เจ้าคุณทหารลาดกระบัง

เรื่อง ระบบแสดงคลื่นไฟฟ้าหัวใจบนไมโครคอมพิวเตอร์

(Micro Computer Based EKG Monitor)

คณะผู้จัดทำ

1.	นายพีรพงษ์	วัฒนเกษมธรรม	35103194
2.	นายสมคิด	ฤทธิ์คง	35103206
3.	นายสฤษฎี	วงศ์ว่องการ	35103211

อาจารย์ที่ปรึกษา

.....

(ดร.กิตติพล ชิตสกุล)

ระบบแสดงคลื่นไฟฟ้าหัวใจบนไมโครคอมพิวเตอร์

คณะผู้จัดทำ

นาย พีระพงษ์ วัฒนเกษมธรรม

นาย สมคิด ฤทธิ์คง

นาย สฤษฎ์ วงศ์ว่องการ

อาจารย์ที่ปรึกษา

ดร. กิติพล ชิตสกุล

บทคัดย่อ

ในปริญญานิพนธ์เล่มนี้ ได้นำเสนอการออกแบบระบบแสดงคลื่นไฟฟ้าหัวใจ บนไมโครคอมพิวเตอร์ โดยพิจารณาว่าสัญญาณคลื่นไฟฟ้าหัวใจผ่านการขยายโดยวงจร Conditioning Amp มีขนาดพอเหมาะกับการแสดงผล และแปลงสัญญาณคลื่นไฟฟ้าหัวใจ โดยวงจรเปลี่ยนสัญญาณอนาลอกเป็นดิจิตอล 8 บิท และ ส่งข้อมูลแบบอนุกรม โดยใช้ไมโครโปรเซสเซอร์ 8031 ส่งข้อมูลตามมาตราฐาน RS232 และศึกษาการรับ และแสดงผลกราฟพิคบนจอมอนิเตอร์ ซึ่งเป็นการศึกษาความเป็นไปได้ในการรับ-ส่งข้อมูลที่เป็นคลื่นไฟฟ้าหัวใจ และนำไปพัฒนาในการใช้งานจริงต่อไป

Micro Computer Based EKG Monitor

Student Name

1. Mr. Pheerapong Wattanagasemkit

2. Mr. Somkid Litkong

3. Mr. Sitsadee Wongwongkran

Advisor

Dr. Kittipol Chitsakul

Abstract

THE CONTENT OF THIS THESIS WE PRESENT THE ANALYSIS AND DESIGN OF THE MICROPROCESSOR BASED EKG SIGNAL MONITOR.

WE FIRST DESCRIBE THE CIRUIT OF THE CONDITIONING AMPLIFIER WHICH PROVIDE APPROPRIATE SIGNAL CHARACITERISTIC, THEN WE DEFINED THE COMPONENTS OF 8 BIT ANALOG TO DIGITAL CONVERTER AND ALSO COVER SOME DETAILS OF RS 232 SERIAL INTERFACE AND THE IMPORTANT PART IS THE CONCEPT OF (CPU 8031) MICROPROCESSOR LATER THE GRAPHIC OUTPUT OF THE EKG SIGNAL ARE DISPLAYED ON THE MONITOR BEING GENARATED BY THE PROGRAMS PROVIDED IN THE APPENDIX.

กิตติกรรมประกาศ

ในการสร้างโครงงานปริญญานิพนธ์เล่มนี้ ทางคณะผู้จัดทำได้รับความอนุเคราะห์ช่วยเหลือให้คำปรึกษา และแนะนำแนวทางในการดำเนินงานจากท่านอาจารย์ดังนี้ คือ

ดร. กิตติพล ชิตสกุล อาจารย์ที่ปรึกษาโครงงาน

ทางคณะผู้จัดทำโครงงานจึงขอขอบพระคุณท่านมา ณ.โอกาสนี้ ด้วย

สารบัญ

บทที่ 1

บทนำ (Introduction)

จุดประสงค์และแนวคิดของโครงงาน

แนวคิด

จากสัญญาณ ECG (Electro Cardiogram) ซึ่งเป็นสัญญาณที่อยู่ในรูปฟังก์ชั่นของแรงดัน time domain ดังนั้นก่อนที่เราจะประมวลข้อมูล เราจะต้องทำการแปลงแรงดันให้เป็นข้อมูลทาง digital ก่อนด้วย A/D หลังจากนั้นจะส่งเข้าประมวลผลใน Computer โดยผ่าน Serial port ของ Computer จึงต้องมีการคำนวณหาอัตราส่งข้อมูลสูงสุด - Baud rate เป็นอัตราความเร็วในการ interface รับ-ส่ง ข้อมูลกับ Computer โดยมีความเร็วเป็น bits/sec

เพราะฉะนั้นการเลือก Baud rate = 9,600 bits/sec

ซึ่งข้อมูล 1 ตัว จะต้องใช้จำนวนบิททั้งหมดต่ำสุด 11 บิท คือ

8 Bit data , 1 bit stop , 2 bit start , No parity

จะได้ความเร็ว = $\frac{9,600}{11}$ = 872.72 characters/sec

ซึ่งเป็นความเร็วสูงสุดแต่ในการใช้งานเราจะใช้ที่ความเร็วนี้ไม่ได้ เพราะความเร็วหมายถึงว่า เราไม่ได้ใช้เวลาในการ Intial port เตรียมข้อมูลและจัดเก็บข้อมูลเลย ดังนั้นจึงต้องลดลงมาซึ่งความเร็วสูงสุดจะเป็นเท่าไรขึ้นอยู่กับความเร็วของเครื่องคอมพิวเตอร์ที่ใช้ ซึ่งในโครงงานนี้เราเลือกใช้ที่ 400 char/sec ซึ่งหากคลื่นหัวใจที่วัดอยู่ในช่วง 10-100 ครั้ง/นาที จึงเท่ากับ

400 char/sec = 24,000 char/minute(Sampling rate = 300 ครั้ง/sec)

ที่ 10 ครั้ง/minute = 2,400 char/cycle (การ Sampling)

ที่ 100 ครั้ง/minute = 240 ครั้ง/cycle

ซึ่งที่ 400 ครั้ง/sec เราจะได้ T = 2.5 msec หมายความว่าทางทฤษฎีเราสามารถเลิกการเปลี่ยนแปลงของแรงดันที่มีความถี่ 2T ได้ ซึ่งเป็นช่วงที่สัญญาณ ECG มีการเปลี่ยนแปลงในช่วงสั้นๆ

ในการรับข้อมูล 1 ครั้ง เราจะใช้เวลา 6 sec ซึ่งทั้งนี้ขึ้นอยู่กับอัตราการเต้นของหัวใจต่ำสุดที่เครื่องสามารถวัดได้ โดยในที่นี้เราตั้งไว้ที่ 20 cycle/minute

ถ้าเต้น 20 cycle/minute หมายความว่า 1 cycle ในเวลา 3 sec คือ การที่เราจะรับข้อมูลของสัญญาณได้ครบ 2 cycle เราจะต้องใช้เวลา 6 sec

ส่วนอัตราการเต้นสูงสุด ซึ่งจะมี Error มากมายเราวัดที่

$$\frac{fs}{2} \text{ คือ } \frac{300 \text{ ครั้ง/sec}}{2} = 150 \text{ ครั้ง/sec}$$

หรือ 150 H_Z ซึ่งเท่ากับ 3,000 ครั้ง/minute ซึ่งไม่มีมนุษย์คนใดสามารถทำได้อยู่แล้ว แต่เราจะมาพิจารณาในช่วงแรงดัน ECG ที่มีการเปลี่ยนแปลงในเวลาสั้นๆ แทนจึงกลายเป็นค่า Band width ของเครื่องวัดได้ คือมี Band width = 150 H_Z

รูปที่ 1.1 แสดงค่าเวลาการเปลี่ยนแปลงของแรงดันในช่วงสั้นๆ (t)

การเปลี่ยนแปลงของแรงดันในช่วงสั้นๆ ซึ่งเวลาต่ำสุดที่สามารถวัดได้ 1/150 = 6.66 msec

ส่วน Bandd width ของสัญญาณคลื่นไฟฟ้าหัวใจ $f_{max} = \frac{1}{2t}$

โดยทั่วไปค่า t มีค่า 0.08 - 0.1 วินาที เพราะฉะนั้น f_{max} มีค่า 6.25 - 5H_Z

แสดงว่า Band width สัญญาณคลื่นไฟฟ้าหัวใจ น้อยกว่า Band width ของเครื่องวัด เพราะฉะนั้น ใช้งานในการวัดสัญญาณคลื่นไฟฟ้าหัวใจได้

บทที่ 2

คลื่นไฟฟ้าหัวใจและความผิดปกติของจังหวะการเต้นของหัวใจ

(Electrocardiogram and Arrhythmias)

ในบทนี้เป็นการศึกษาความเป็นมาของคลื่นไฟฟ้าหัวใจและ ความผิดปกติของจังหวะการเต้นของหัวใจ โดยในส่วนของคลื่นไฟฟ้าหัวใจจะกล่าวถึงการเกิดปรากฏการณ์ไฟฟ้าภายในเซล การทำงานของหัวใจ การนำไฟฟ้าภายในหัวใจและลักษณะของสัญญาณคลื่นไฟฟ้าหัวใจที่บันทึกได้ ตลอดจนแนวทางในการพิจารณาและวิเคราะห์คลื่นไฟฟ้าหัวใจ

2.1 ปรากฏการณ์ไฟฟ้าของเซล

การทำงานของอวัยวะต่างๆ ในร่างกายของมนุษย์อยู่ภายใต้การควบคุมของระบบประ-สาท โดยคำสั่งจากระบบประสาทจะถูกส่งไปตามเส้นประสาทต่างๆ เข้าสู่เซลของกล้ามเนื้อ หรือจากอวัยวะรับความรู้สึก ซึ่งถูกกระตุ้นจากภายนอกผ่านเส้นประสาทกลับเข้าสู่ระบบประสาท การส่งข้อมูลของคำสั่งจากระบบประสาทไปยังกล้ามเนื้อ หรือจากอวัยวะรับความรู้สึกกลับเข้าสู่ระบบประสาทนี้ เป็นการส่งในรูปของสัญญาณไฟฟ้าและเคมี โดยข้อมูลที่ส่งผ่านเซลจะเป็นการส่งผ่านด้วยไฟฟ้าและจุดต่อระหว่างเซลจะเป็นการส่งผ่านแบบเคมี สัญญาณไฟฟ้าในการส่งผ่านข้อมูลนี้ เกิดจากการเปลี่ยนแปลงศักดาไฟฟ้าภายในเซล ดังนั้นศักดาไฟฟ้าของสิ่งมีชีวิต (Bioelectric Potentials) จึงถือกำเนิดมาจากเซลหรืออาจกล่าวได้ว่า เซลเป็นเสมือนจุดกำเนิดของศักดาไฟฟ้าทั้งหมดภายในร่างกาย

การเกิดศักดาไฟฟ้าขึ้นภายในเซลเนื่องจากบริเวณรอบๆ เซลประกอบด้วยของเหลวซึ่งมีไอออนต่างๆ ปะปนอยู่ ไอออนที่มีบทบาทสำคัญต่อกลไกการเกิดศักดาไฟฟ้าภายในเซล ก็คือ โซเดียมไอออน (Na+) โปตัสเซียมไอออน (K+) และคลอไรด์ไอออน (CI-) โดยแต่ละเซลจะมีผนังเซลหรือเซลเมมเบรน (Cell Membrance) ซึ่งมีคุณสมบัติต่อไอออนเป็นแบบ Semipermeable คือ จะยอมให้อนุภาคหรือไอออนบางชนิดผ่านผนังเซลได้ ในขณะที่ไอออนอีกส่วนหนึ่งไม่สามารถผ่านไปได้ โดยปกติแล้วผนังเซลจะยอมให้เฉพาะโปตัสเซียมไอออนและคลอไรด์ไอออนผ่านไปได้ ส่วนโซเดียมไอออนจะไม่สามารถผ่านผนังเซลได้

.เมื่อความเข้มข้นของไอออนที่รอยต่อของผนังเซลเกิดความลาดเอียง ของความเข้มข้น (Concentration gradient) คือ ไอออนบริเวณด้านใดด้านหนึ่งของผนังเซลมีความเข้มข้นมากกว่าอีกด้านหนึ่ง ก็จะมีการแพร่ของไอออนจากด้านที่มีความเข้มข้นมากกว่า ไปยังด้านที่มี

ความเข้มข้นน้อยกว่า เมื่อไอออนผ่านผนังเซลไปแล้ว ก็จะทำให้เกิดความไม่สมดุลย์ของประจุไฟฟ้า เป็นผลให้เกิดสนามไฟฟ้าด้านการแพร่ของไอออน ทำให้ไอออนเคลื่อนผ่านผนังเซลลดลง จนกระทั่งถึงสภาวะสมดุล (Equillibrium) เมื่อแรงด้านการเคลื่อนที่ของไอออน เนื่องจากสนามไฟฟ้าและแรงที่เกิดจากความแตกต่างของปริมาณความเข้มข้นของไอออนมีค่าเท่ากัน ก็จะทำให้ ไอออนหยุดการแพร่ผ่านผนังเซล

ในเซลของสัตว์เลี้ยงลูกด้วยนมนั้น ปริมาณความเข้มข้นของโปตัสเซียมไอออนภายในเซลจะมีค่ามากกว่าความเข้มข้นของไอออนภายนอกเซลอยู่ประมาณ 30 เท่า ไอออนและความเข้มข้นของโซเดียมไอออนภายนอกเซล จะมีค่ามากกว่าความเข้มข้นภายในเซลอยู่ ประมาณ 10 เท่า เนื่องจากคุณสมบัติของผนังเซลที่ยอมให้โปตัสเซียมไอออนผ่านไปได้ จึงทำให้เกิดการแพร่ของโปตัสเซียมไอออนจากภายในออกสู่ภายนองเซล ส่วนคลอไรด์ไอออนจะมีอัตราการแพร่น้อยกว่าโปตัสเซียมไอออน เนื่องจากแรงดึงดูดของประจุไฟฟ้าภายในเซลทำให้เกิดการสูญเสียประจุไฟฟ้าบวกขึ้นภายในเซล เป็นผลให้ศักดาไฟฟ้าภายในเซลมีค่าเป็นลบเมื่อเทียบกับภายนอกเซล เมื่อสภาวะสมดุลมาถึงความต่างศักย์ระหว่างภายในเซลกับภายนอกเซลมีค่า ระหว่าง -50 มิลลิโวลท์ ถึง -100 มิลลิโวลท์ โดยขึ้นอยู่กับชนิดของเซล ค่าของความต่างศักย์นี้เรียกว่า ศักดาไฟฟ้าขณะอยู่นิ่ง (Resting Potential) ซึ่งศักดาไฟฟ้านี้จะมีค่าคงที่อยู่เสมอตราบที่เซลนั้นยังไม่ถูกกระตุ้น เซลที่อยู่ในสภาวะนี้ เรียกว่า เซลอยู่ในสภาวะโพลาไรซ์ (PPolarized) และการสูญเสียสภาวะของศักดาไฟฟ้าขณะอยู่นิ่ง ซึ่งเรียกปรากฏการณ์นี้ว่า ดีโพลาไรเซชั่น (Depolarization) ที่สภาวะสมดุลย์นี้ ค่าของศักดาไฟฟ้าขณะอยู่นิ่งนั้นสามารถประมาณค่าได้ตามสมการของเนิรสท์ (Nemst Equation) ซึ่งเป็นฟังก์ชั่นของความเข้มข้นของไอออน ในแต่ละด้านของเซลได้ดังนี้

$$E = \left(\frac{RT}{ZF}\right) \ln \left(\frac{C_0}{C_1}\right)$$

โดยที่ E คือ ค่าความต่างศักย์ของเซล (หน่วยเป็นโวลท์)

R คือ ค่าคงที่ของก๊าซ (เท่ากับ 8.314 จูล/โมล-เคลวิน)

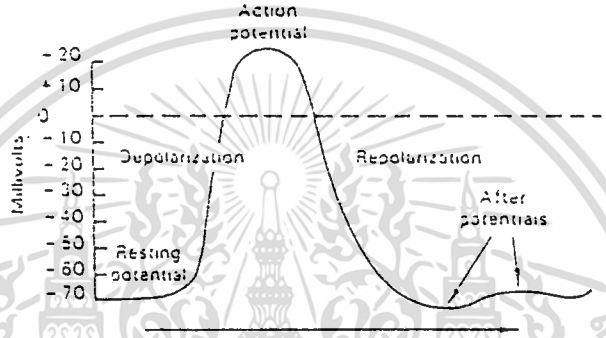
T คือ ค่าของอุณหภูมิสัมบูรณ์ (หน่วยเป็นองศาเคลวิน)

Z คือ ค่าวาเลนซีของไอออน

F คือ ค่าคงที่ของฟาราเดย์ (เท่ากับ 1 ฟาราเดย์ หรือ 96,500 คูลอมบ์/โมล)

C_0 , C_1 คือ ค่าความเข้มข้นของไอออนภายนอกและภายในเซลตามลำดับ (หน่วยเป็นโมล)

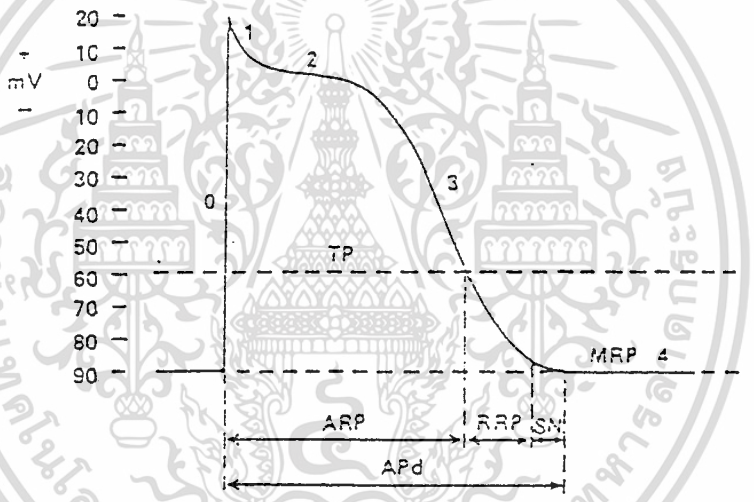
เซลที่อยู่ในสภาวะโพลาไรซ์ สามารถถูกกระตุ้นด้วยสิ่งเร้าหลายประเภทขึ้นอยู่กับชนิดของเซลนั้น เช่น ความร้อน แสง และรูปแบบอื่นๆ ซึ่งการกระตุ้นของสิ่งเร้าอย่างน้อยต้องมีแรงกระตุ้นเพียงพอ ที่จะทำให้เกิดการเปลี่ยนแปลงศักดาไฟฟ้าภายในของเซล โดยไม่คำนึงถึงระยะเวลาของการกระตุ้นว่านานเพียงใด คุณสมบัติของเซลในลักษณะนี้ เป็นไปตามกฎการเกิดขึ้นหรือไม่เกิดขึ้น (All-or-Nothing Law) ค่าของแรงกระตุ้นน้อยที่สุด ที่ทำให้เกิดการเปลี่ยนแปลงภายในเซล เรียกว่า ค่าวิกฤตของการกระตุ้น (Threshold Value)



รูป 2.1 ศักดาไฟฟ้าทำงาน

เมื่อเซลถูกกระตุ้น คุณสมบัติของผนังเซลจะเปลี่ยนแปลงไปชั่วขณะ โดยจะยอมให้โซเดียมไอออนผ่านเข้าภายในเซลอย่างรวดเร็ว ทำให้ความต่างศักย์ระหว่างภายในเซลกับภายนอกเซลมีค่าเพิ่มขึ้นจนถึงประมาณบวก 20 มิลลิโวลท์ ศักดาไฟฟ้าขณะนี้เรียกว่าศักดาไฟฟ้าทำงาน (Action Potential) และจะกลับคืนสู่ศักดาไฟฟ้าขณะอยู่นิ่ง ดังรูป 2.1 ขณะที่เกิดศักดาไฟฟ้าทำงานนี้ เซลจะอยู่ในสภาพดีโพลาไรซ์ (Depolarized) และการเปลี่ยนแปลงของศักดาไฟฟ้าจากศักดาไฟฟ้าขณะอยู่นิ่ง ไปเป็นศักดาไฟฟ้าทำงาน เรียกว่า ดีโพลาไรเซชั่น (Depolarization) หลังจากนี้คุณสมบัติของผนังเซลจะกลับคืนสู่สภาพเดิม คือยอมให้โปตัสเซียมไอออนผ่านผนังเซลไปได้ส่วนโซเดียมไอออนก็จะถูกขบวนการที่เรียกว่า โซเดียมปั๊ม (Sodium Pump) นำออกจากเซลอย่างช้าๆ ซึ่งขบวนการนี้จะใช้พลังงานจากการสันดาบของเซล Metabolisem) เพื่อนำโซเดียมไอออนนอกจากเซล ศักดาไฟฟ้าของเซลก็จะคืนสู่ศักดาไฟฟ้าขณะอยู่นิ่งตามเดิมซึ่งเรียกปรากฏการณ์นี้ว่า รีโพลาไรเซชั่น (Repolarization) เซลก็จะกลับสู่สภาวะปกติ จนกว่าจะมีการกระตุ้นใหม่ ช่วงเวลาที่ศักดาไฟฟ้าของเซลเปลี่ยนจากศัก-

ดาไฟฟ้าขณะอยู่นิ่ง ไปเป็นศักดาไฟฟ้าทำงาน แล้วกลับมาสู่ศักดาไฟฟ้าขณะอยู่นิ่งตามเดิมนั้นจะมีระยะเวลาแตกต่างกันขึ้นอยู่กับชนิดของเซล ดังรูป 2.2 แสดงช่วงเวลาต่างๆ ของการเปลี่ยนแปลงศักดาไฟฟ้าของเซลกล้ามเนื้อหัวใจเมื่อถูกกระตุ้น โดยที่แนวแกนตั้งเป็นขนาดของศักดาไฟฟ้าและแนวแกนนอนเป็นเวลาตัวอักษรย่อต่างๆ ใช้อธิบายการเปลี่ยนแปลงของศักดาไฟฟ้าที่เกิดขึ้นดังนี้ คือ 0 = Depolarization; 1,2,3 = Phases of Repolarization; 4 = Diastolic Phases = MRP;MRP = Membrane Resting Potential; DAP = Duration of Action Potential;TP = Threshold Potential;ARP = Absolute Refractory Period; RRP = Relatice Refractory Period; SN = Supernormal Period.



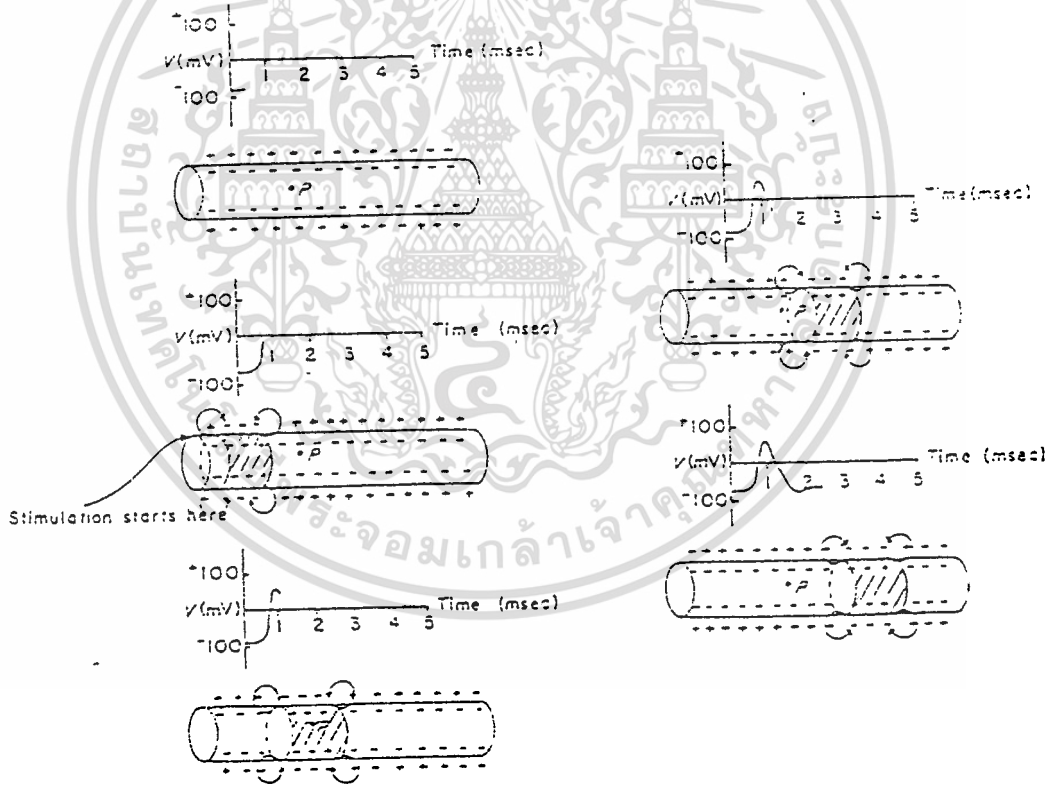
รูป 2.2 ช่วงเวลาต่างๆ ของการเปลี่ยนแปลงศักดาไฟฟ้าของเซลกล้ามเนื้อหัวใจเมื่อถูกกระตุ้น

ศักดาไฟฟ้าทำงานของเซลที่ถูกกระตุ้น ทำให้เกิดความลาดเอียงของศักย์ไฟฟ้า (Potential gradient) เป็นผลให้เกิดกระแสไหลไปกระตุ้นเซลอื่นๆ ที่อยู่ใกล้เคียง ซึ่งถ้าการกระตุ้นเป็นไปตามกฎการเกิดขึ้นและไม่เกิดขึ้นของเซล ก็จะทำให้เกิดศักดาไฟฟ้าทำงานต่อกันไปเรื่อยๆ ลักษณะเช่นนี้เป็นการนำไฟฟ้าของศักดาไฟฟ้าทำงาน ดังรูป 2.3 ซึ่งแสดงการนำไฟฟ้าภายในเซลประสาทและกราฟของศักดาไฟฟ้าที่จุด P ตามเวลาที่เกิดขึ้น ถ้าการนำไฟฟ้านี้เกิดขึ้นที่เซลประสาท ศักดาไฟฟ้าทำงานของเซลประสาทก็คือคำสั่งของระบบประสาทต่อเซลกล้ามเนื้อ ซึ่งศักดาไฟฟ้าทำงานนี้จะส่งจากเซลประสาทเซลหนึ่งไปยังเซลที่อยู่ถัดไปเรื่อยๆ

ตามเส้นประสาท จนถึงเซลกล้ามเนื้อ ซึ่งจะทำให้เกิดการหดตัวของกล้ามเนื้อและเกิดศักดาไฟฟ้าทำงานขึ้นด้วย ในทางตรงข้าม เมื่อเซลของอวัยวะรับความรู้สึกถูกกระตุ้นจากพลังงานภายนอกจะเกิดศักดาไฟฟ้าทำงานขึ้น แล้วส่งผ่านเส้นประสาทกลับไปยังระบบประสาท

2,2 การทำงานของหัวใจ

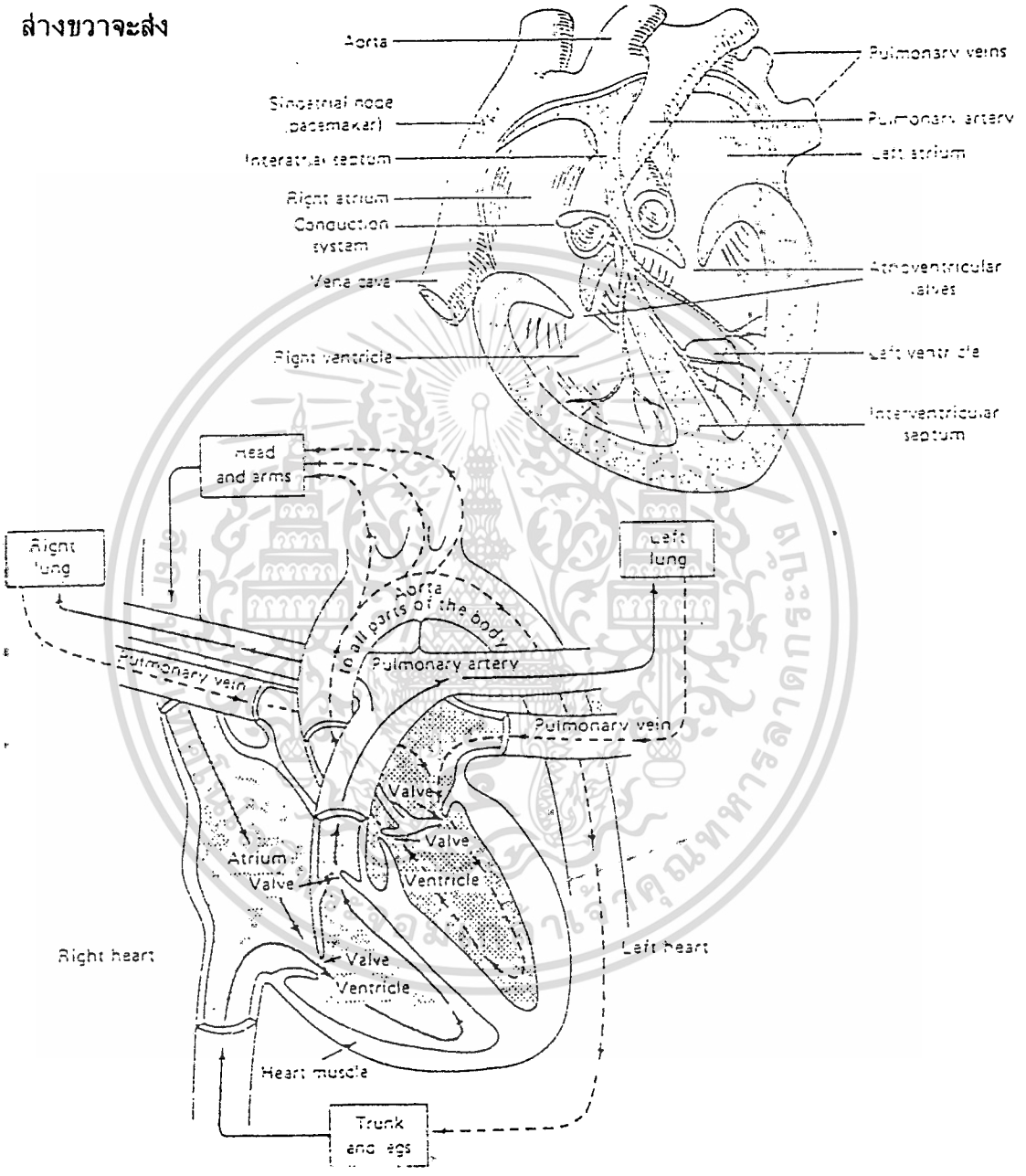
หัวใจทำหน้าที่เสมือนลูกสูบจ่ายโลหิตไปเลี้ยงเซลต่างๆ ในร่างกายรวมทั้งเซลของกล้ามเนื้อหัวใจเอง หัวใจตั้งอยู่ในทรวงอกเหนือกระบังลมค่อนมาทางด้านซ้าย ภายในหัวใจแบ่งออกเป็น 4 ห้องมีผนังกั้นระหว่างซีกซ้ายและซีกขวา เรียกว่า เซปตั้ม (Septum) โดยห้องหัวใจที่อยู่ทางด้านขวาทำหน้าที่รับโลหิตดำจากส่วนต่างๆ ของร่างกายการบีบตัวของกล้ามเนื้อหัวใจเพื่อส่งโลหิตออกไป จะกระทำพร้อมกันทั้งทางซีกขวาและซีกซ้าย ห้องหัวใจที่อยู่ด้านบน



รูป 2.3 การนำไฟฟ้าของศักดาไฟฟ้าทำงานภายในเซล

เรียกว่า หัวใจห้องบน (Atrium) และ ห้องหัวใจที่อยู่ด้านล่างเรียกว่า หัวใจห้องล่าง (Ventricle) โลหิตดำจากร่างกายจะไหลกลับเข้าสู่หัวใจทางหลอดโลหิตดำใหญ่ซุปพีเรียเวนาคาวา (Superior vena cava) และ อินฟีเรียเวนาคาวา (Inferior vana cava)

เข้าสู่ห้องหัวใจบนขวาโลหิตแดงที่ได้รับออกซิเจนจากปอดจะเข้าสู่หัวใจทางหลอดโลหิตแดงพัล-โมนารีเวน (Pulmonary veins) เข้าสู่ห้องหัวใจบนซ้ายหัวใจห้องบนทั้งขวาและซ้ายจะบีบตัวส่งโลหิตไปยังหัวใจห้องล่างซึ่งมีลิ้นหัวใจกั้นอยู่ หลังจากที่หัวใจห้องบนหดตัวส่งโลหิตมายังหัวใจห้องล่างชั่วครู่ หัวใจห้องล่างจะหดตัวส่งโลหิตออกไปพร้อมกันทั้งซ้ายและขวา หัวใจห้องล่างขวาจะส่ง

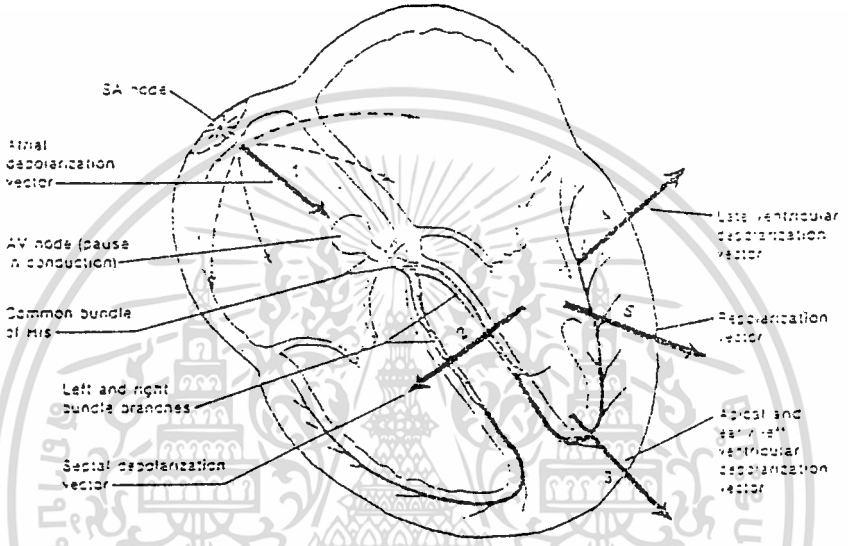


รูป 2.4 แผนผังของหัวใจและระบบการสูบฉีดโลหิตของหัวใจ

โลหิตดำไปยังปอดเพื่อรับออกซิเจน โดยผ่านหลอดโลหิตดำใหญ่พัลโมน่ารีอาเตอรี (Pulmonary Artery) และห้องล่างซ้ายจะส่งโลหิตไปเลี้ยงร่างกาย โดยผ่านหลอดโลหิตแดงใหญ่เออร์ตาร์ (Aurta) ซึ่งการทำงานทั้งหมดที่ได้กล่าวมานี้แสดงในรูป 2.4

2.3 ระบบนำไฟฟ้าของหัวใจ

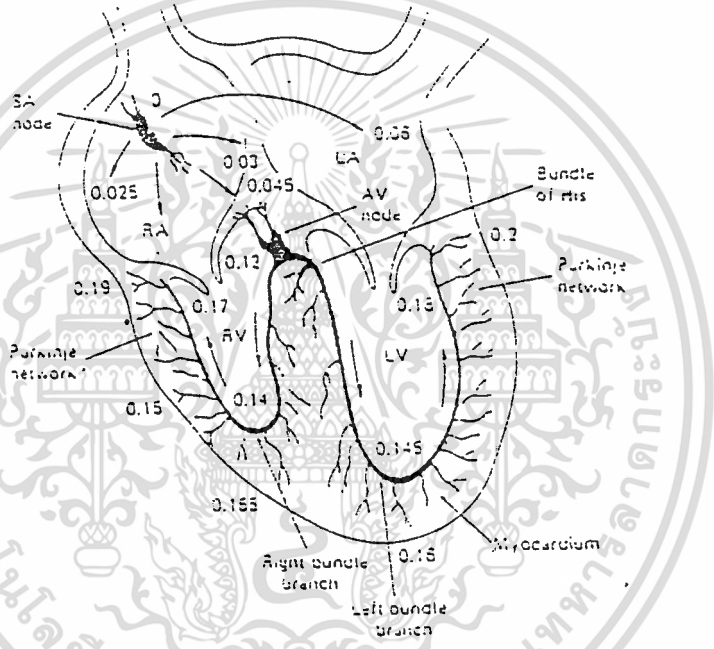
การทำงานของกล้ามเนื้อหัวใจก็จะเป็นไปในลักษณะเดียวกันกับกล้ามเนื้ออื่นๆ กล่าวคือจะถูกกระตุ้นด้วยศักดาไฟฟ้าทำงาน แต่การกระตุ้นนี้มิได้มาจากระบบประสาทส่วนกลางหรือสมอง เป็นการกระตุ้นต่อเซลกล้ามเนื้อหัวใจที่เกิดขึ้นจากกายในตัวหัวใจเอง โดยประกอบด้วยกลุ่มเซลกลุ่มหนึ่งที่ทำหน้าที่ผลิตพัลส์ทำหน้าที่คล้ายกับเส้นประสาท โดยระบบนำไฟฟ้านี้จะนำพัลส์ไฟฟ้าไปกระตุ้นต่อเซลกล้ามเนื้อหัวใจ ดังรูป 2.5



รูป 2.5 การนำไฟฟ้าภายในหัวใจ

บริเวณผนังด้านในของหัวใจระหว่างหลอดโลหิตใหญ่ซุปพีเรียเวนาคาวาและอินฟีเรียเวนาคาวาจะมีเซลอยู่กลุ่มหนึ่งที่มีคุณสมบัติพิเศษ เซลกลุ่มนี้จะสร้างพัลส์ไฟฟ้าเพื่อกระตุ้นเซลกล้ามเนื้อหัวใจซึ่งเซลกลุ่มนี้ เรียกว่า SA node (Sinoatrial node) หรือไซนัสโหนด (Sinus node) หรือเพสเมคเกอร์ (Pacemaker) โดยความถี่ของพัลส์ที่ SA node สร้างขึ้นจะมีอิทธิพลในการกำหนดอัตราการเต้นของหัวใจ ซึ่งโดยปกติแล้วค่าของอัตราการเต้นของหัวใจจะเท่ากับความถี่ของพัลส์ที่ SA node นี้ปล่อยออกมา พัลส์ไฟฟ้านี้จะแผ่กระจายออกจาก SA node ผ่านหัวใจห้องบนทั้งซ้ายและขวา ไปสู่ AV node (Atrioventricular node) โดยที่ AV node นี้จะอยู่ที่ผนังกั้นหัวใจทางด้านขวาระหว่างห้องบนขวาและห้องล่างขวา พัลส์ไฟฟ้าที่ผ่านหัวใจห้องบนจะทำให้หัวใจห้องบนหดตัวบีบโลหิตมายังห้องล่างเส้นทางนำไฟฟ้า จาก SA-node ไปสู่ AV node ดังรูป 2.5 ที่ AV node นี้ประกอบด้วยเซลประสาทที่ทำหน้าที่หน่วงเวลาประมาณ 70 msเพื่อให้การทำงานของหัวใจห้องบนและห้องล่างสัมพันธ์กันจาก AV node

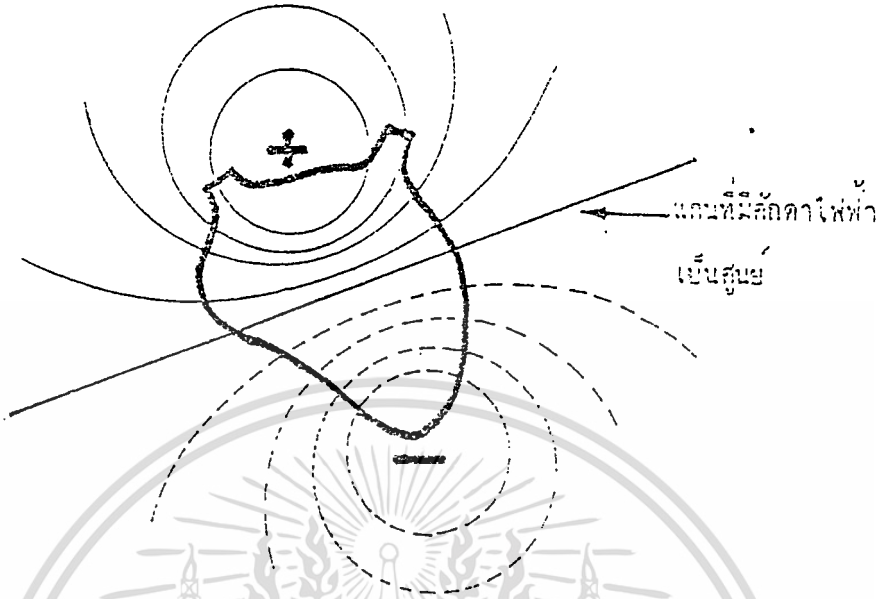
จะมีระบบนำไฟฟ้าในการนำพัลส์ไฟฟ้าไปสู่กล้ามเนื้อหัวใจส่วนของบันเดิ้งออฟฮิส (Bundle of His) ซึ่งแยกออกเป็น 3 เส้นทาง โดยจะไปสู่ห้องล่างซ้ายสองสาขาและห้องล่างขวาอีกหนึ่งสาขา แต่ละสาขาก็จะนำพัลส์ไฟฟ้าไปกระตุ้นเซลกล้ามเนื้อหัวใจห้องล่าง โดยผ่านกล้ามเนื้อหัวใจในส่วนของเพอรกิจเจไฟเบอร์ (Purkinje fibers) รูป 2.6 แสดงให้เห็นถึงช่วงเวลาใช้ในการส่งพัลส์ไฟฟ้าขึ้นมาเช่นเดียวกับ SA node ไปยังส่วนต่างๆของหัวใจ ซึ่งที่ AV node นี้มีการผลิตพัลส์ไฟฟ้าขึ้นมาเช่นเดียวกับที่ Sa node แต่สำหรับคนปกติแล้วความถี่ของพัลส์ไฟฟ้าที่ AV node ผลิตขึ้นจะมีอัตราต่ำกว่าของ SA node



รูป 2.6 ช่วงเวลาที่ใช้ในการส่งพัลส์ไฟฟ้าจาก SA node ไปยังส่วนต่างๆ ของหัวใจ

AV node จะถูกกระตุ้นด้วยพัลส์จาก SA node ทำให้อัตราการเต้นของหัวใจมีค่าเท่ากับความถี่ของ SA node แต่ถ้าเส้นทางนำไฟฟ้าจาก SA node ไปสู่ AV node ผิดปกติหรือถูกสกัดกั้น (AV Block) หัวใจก็จะเต้นด้วยพัลส์ที่ AV node สร้างขึ้น ซึ่งจะมีค่าประมาณ 40-45 ครั้งต่อนาที

2.4 คลื่นไฟฟ้าหัวใจ (Electrocardiogram : ECG)



รูป 2.7 การกระจายของศักดาไฟฟ้าบนผิวหนังมีลักษณะเสมือนเป็นอิเล็กตริคไดโพล

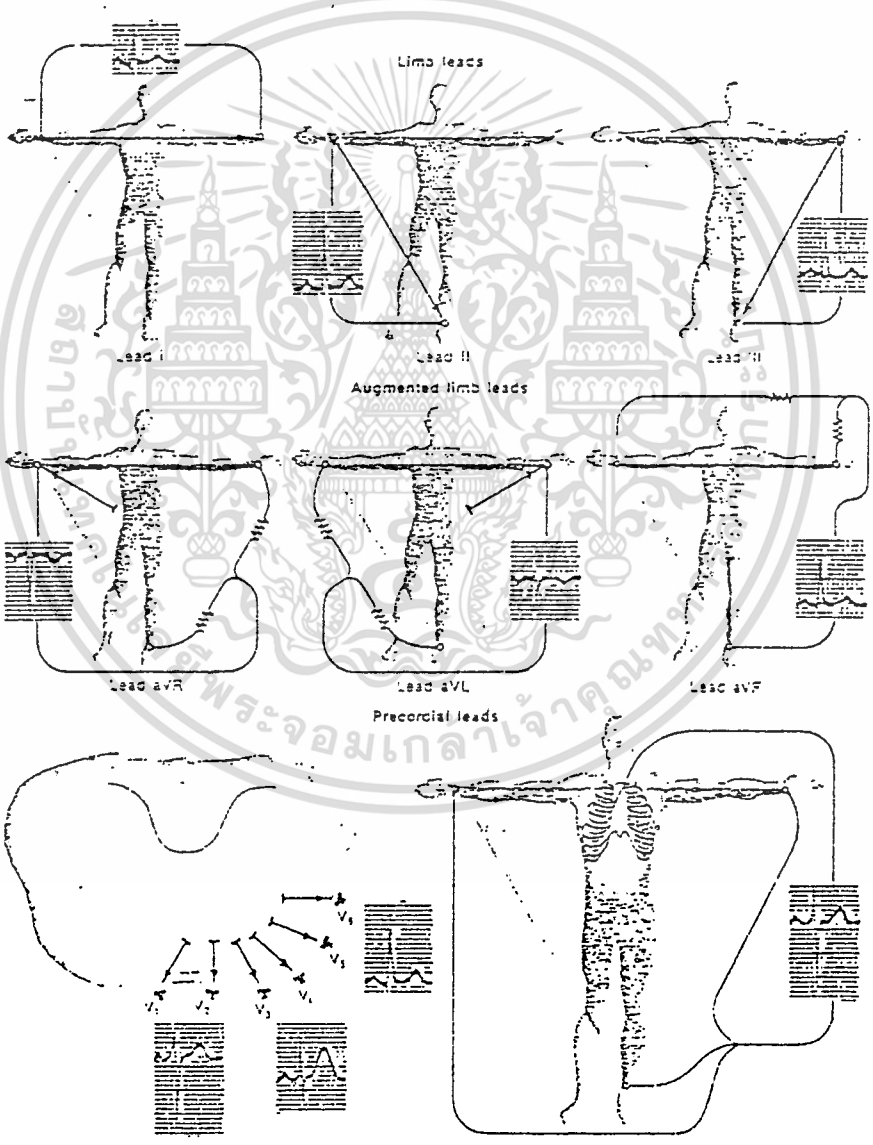
จากที่ได้กล่าวมาแล้วในหัวข้อที่ผ่านมา การทำงานของกล้ามเนื้อเกิดจากการที่พัลส์ไฟฟ้ามากระตุ้นทำให้กล้ามเนื้อเกิดการหดตัวและเกิดศักดาไฟฟ้าทำงานขึ้นด้วย ซึ่งกล้ามเนื้อหัวใจก็เป็นเช่นเดียวกันการเคลื่อนที่ของไอออนภายในเซลกล้ามเนื้อ ทำให้เกิดศักดาไฟฟ้าทำงานและทำให้หัวใจเต้น การเคลื่อนที่ของไอออนภายในเซลกล้ามเนื้อหัวใจ จะรวมตัวเป็นการไหลของกระแสไฟฟ้า และเป็นผลทำให้เกิดความต่างศักดาไฟฟ้าภายนอกเนื้อเยื่อและที่บริเวณผิวหนังของร่างกาย การไหลของกระแสจะเกิดขึ้นเฉพาะเวลาที่เกิดการกระจายของศักดาไฟฟ้าที่ทำงานเท่านั้น ดังนั้นเราอาจจะพิจารณาได้ว่าหัวใจเป็นเสมือนแหล่งกำเนิดไฟฟ้า ซึ่งบรรจุอยู่ภายในกล้ามตัวนำร่างกาย ศักดาไฟฟ้าที่เกิดขึ้นจะมีการกระจายออกจากขั้วบวกและขั้วลบไปตามส่วนต่างๆ ของร่างกาย เหมือนกับเป็นอิเล็กตริคไดโพล (Electric dipole) ดังแสดงในรูป 2.7 และสามารถวัดศักดาไฟฟ้าตกคร่อมระหว่างจุดใดๆ ที่อยู่บนผิวหนังของร่างกายได้ ซึ่งศักดาไฟฟ้าที่วัดได้นี้เรียกว่า สัญญาณคลื่นไฟฟ้าหัวใจ (Electrocardiogram) เรียกย่อๆ ว่า ECG โดยคลื่นไฟฟ้าหัวใจที่วัดได้ระหว่างจุดต่างๆ จะไม่เหมือนกันขึ้นอยู่กับมุมและระยะทางของตำแหน่งที่วัดกระทำต่อแกนหัวใจ (Heart axis) ดังรูป 2.8 สัญญาณคลื่นไฟฟ้าหัวใจที่วัดได้จากคนปกติจะเป็น ดังในรูป 2.9 แต่ละช่วงของสัญญาณจะมีชื่อ เรียกแทนตัวอักษร P,Q,R,S,T,U ซึ่งจะมีความสัมพันธ์กับการทำงานของหัวใจในช่วงต่างๆ ภายในหนึ่งรอบของ

การเต้นของหัวใจ แต่ละช่วงของสัญญาณจะมีความหมายดังต่อไปนี้

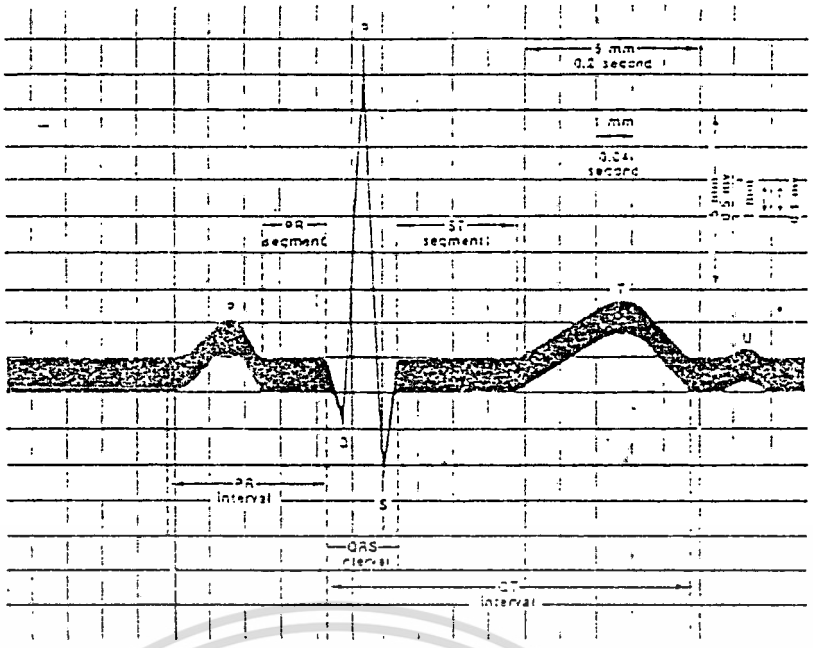
- สัญญาณ P เกิดจากการทำงานของหัวใจห้องบน จะมีคาบเวลาประมาณ 80-120 ms

- สัญญาณรวม QRS เกิดจากการทำงานของหัวใจห้องล่าง จะมีคาบเวลาประมาณ 80-100 ms และสัญญาณ R จะมีขนาดสูงที่สุด เนื่องจากหัวใจห้องล่างจะต้องบีบโลหิตส่งยังทุกส่วนของร่างกายผ่านผนังของหัวใจห้องล่างจึงมีความหนามากกว่าส่วนอื่นๆ

โดยการนำขั้วไฟฟ้าไปติดที่แขนขวาและแขนซ้ายหรือที่เรียกว่าลีด I (Lead I) การที่สัญญาณ R มีขนาดสูงเป็นเพราะผลรวมของศักดาไฟฟ้าทำงานของเซลเป็นจำนวนมาก



รูป 2.8 แสดงตำแหน่งการวัดคลื่นไฟฟ้าหัวใจทั้ง 12 ลีดมาตราฐาน



รูป 2.9 องค์ประกอบต่างๆ ของคลื่นไฟฟ้าหัวใจ (ลีด 1)

- สัญญาณ T เกิดจากการคลายตัวของกล้ามเนื้อห้องล่าง มีคาบเวลาประมาณ 200 ms และมีขนาดประมาณ 30% ของสัญญาณ R

- สัญญาณ U ยังไม่ทราบสาเหตุแน่นอน แต่สันนิษฐานกันว่าเกิดจากการกลับคืนสู่ระดับศักดาไฟฟ้าขณะอยู่นิ่งอย่างช้าๆ ของเซลกล้ามเนื้อหัวใจห้องล่างหรือที่เรียกว่า ศักดาไฟฟ้าตามหลัง (After potential)

ช่วงเวลต่างๆ ของ	ช่วงเวลาปกติ (วินาที)	
คลื่นไฟฟ้าหัวใจ	ค่าเฉลี่ย	ช่วงเวลา
ช่วงเวลาของ PR*	0.18	0.12 - 0.20
ช่วงเวลาของ QR	0.08	ถึง 0.10
ช่วงเวลาของ QT	0.40	ถึง 0.43
ช่วงเวลาของ ST (คือ QT - QRS)	0.32	---------

หมายเหตุ * ช่วงเวลาของ PR วัดได้จากเวลาของสัญญาณ P ถึงเวลาเริ่มของสัญญาณรวม QRS

ตาราง 2.1 ช่วงเวลาต่างๆ ของคลื่นไฟฟ้าหัวใจ

เวลาในแต่ละช่วงของคลื่นไฟฟ้าหัวใจ แสดงถึงการส่งสัญญาณไฟฟ้าไปยังเนื้อเยื่อของกล้ามเนื้อหัวใจที่จุดต่างๆ ซึ่งเวลาในแต่ละช่วงของสัญญาณปกติสรุปไว้ในตาราง 2.1

2.5 การวิเคราะห์คลื่นไฟฟ้าหัวใจ

คลื่นไฟฟ้าหัวใจเป็นการบันทึกสภาพการทำงานของหัวใจ ที่วัดได้บนผิวหนังของร่างกายด้วยเครื่องวัดและแสดงสัญญาณไฟฟ้าหัวใจ (Electrocardiograph) จึงถือได้ว่าคลื่นไฟฟ้าหัวใจเป็นพารามิเตอร์ของร่างกาย (Physiological parameter) ที่นำมาใช้ประโยชน์ในการวินิจฉัยอาการ ความผิดปกติและประเมินสภาวะของหัวใจได้เป็นอย่างดี การวิเคราะห์และแปรผลคลื่นไฟฟ้าหัวใจที่บันทึกได้จึงเป็นข้อมูลที่สำคัญในการประกอบการรักษาโรคต่างๆ ของผู้ป่วย โดยเฉพาะเกี่ยวกับหัวใจและหลอดเลือด การวิเคราะห์คลื่นไฟฟ้าหัวใจจะทำการบันทึกคลื่นไฟฟ้าหัวใจลงบนกระดาษกราฟ ซึ่งมีแกนนอนเป็นฐานเวลา (หน่วยเป็นวินาที) ส่วนแกนตั้งเป็นความสูงของคลื่น (หน่วยเป็นมิลลิโวลท์) แล้วอาศัยแพทย์หรือผู้เชี่ยวชาญในการแปลผลมาทำการอ่านและวัดพารามิเตอร์ต่างๆ เช่น รูปคลื่นความสูงของคลื่น ระยะเวลาในช่างต่างๆ ระดับของเส้นสันฐานเป็นต้น แล้วทำการสรุปรวบรวมวิเคราะห์ และคาดคะเนว่าผู้ป่วยมีสภาวะของหัวใจเป็นเช่นใด โดยอาจทำการวิเคราะห์ทั้ง 12 ลีด มาตราฐาน หรือเพียงลีดใดลีดหนึ่งขึ้นอยู่กับความสามารถและความชำนาญในการวิเคราะห์ผลของผู้เชี่ยวชาญนั้น ข้อจำกัดของวิธีการแบบนี้ อยู่ที่เวลาที่ใช้ในการวิเคราะห์ผลเนื่องจากภาระกิจประจำของแพทย์หรือผู้เชี่ยวชาญทำให้การทราบผลที่วิเคราะห์ไม่ทันต่อการดูแลรักษาผู้ป่วย ดังนั้นในปัจจุบัน ด้วยความเจริญก้าวหน้าทางเทคโนโลยี ทำให้สามารถนำคอมพิวเตอร์เข้ามาช่วยในการวิเคราะห์ คลื่นไฟฟ้าหัวใจแทนคน โดยการแปลงสัญญาณไฟฟ้าหัวใจให้อยู่ในรูปของข้อมูลดิจิตอล แล้วส่งไปยังคอมพิวเตอร์ซึ่งมีโปรแกรมที่ทำหน้าที่วิเคราะห์ข้อมูลดิจิตอลของสัญญาณไฟฟ้าหัวใจที่เข้ามา เพื่อทำการแปลผลและรายงานผลให้ทราบ ซึ่งวิธีนี้เป็นการแก้ปัญหาบางส่วนของวิธีการวิเคราะห์ในแบบเดิมได้ เช่น กรณีไม่มีแพทย์หรือผู้เชี่ยวชาญมีภาระกิจที่ไม่สามารถทำการวิเคราะห์ผลได้ ทั้งยังให้ผลลัพธ์ที่รวดเร็ว แม่นยำและถูกต้อง ทันต่อการรักษาอาการของผู้ป่วย ดังนั้นคอมพิวเตอร์จึงมีบทบาทสำคัญที่เข้ามาช่วยลดภาระของบุคลากร หรือทดแทนการขาดบุคลากร โดยที่ผลการ

วิเคราะห์ยังคงความเชื่อถือได้อยู่ การวิเคราะห์คลื่นไฟฟ้าหัวใจสามารถพิจารณาได้ 5 หัวข้อ [1][2] คือ

2.5.1 อัตราการเต้นของหัวใจ (Rate)

2.5.2 จังหวะการเต้นของหัวใจ (Rhythm)

2.5.3 แนวแกนของหัวใจ (Axis)

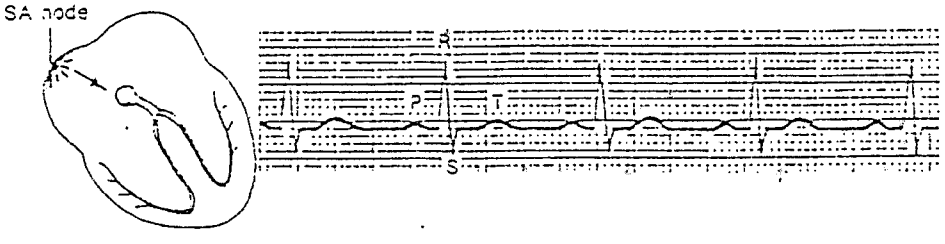
2.5.4 กล้ามเนื้อของหัวใจพองโต (Hypertrophy)

2.5.5 กล้ามเนื้อหัวใจตาย (Infarction)

การวิเคราะห์ทั้ง 5 หัวข้อนี้ เป็นการพิจารณาความผิดปกติที่เกิดขึ้นกับหัวใจ โดยสามารถสังเกตุได้จากคลื่นไฟฟ้าหัวใจ โดยแต่ละหัวข้อจะมีวิธีการสังเกตุที่แตกต่างกันไป ซึ่งสามารถศึกษาวิธีการต่างๆ นี้จากเอกสารอ้างอิง [1][3]

2.6 ความผิดปกติของจังหวะการเต้นของหัวใจ (Arrhythmia)

จังหวะการเต้น (Rhythm) เป็นคุณสมบัติสำคัญอย่างหนึ่งของหัวใจที่แสดงถึง ความสามารถในการทำงานของหัวใจอย่างสม่ำเสมอตลอดเวลา ซึ่งคุณสมบัตินี้เองจะเป็นสิ่งที่แสดงความเป็นปกติของหัวใจโดยจะสามารถทราบได้จากการบันทึกสัญญาณไฟฟ้าหัวใจ (หรือคลื่นไฟฟ้าหัวใจนั่นเอง) จังหวะการเต้นของหัวใจปกติเป็นไปอย่างสม่ำเสมอแน่นอน ตามความถี่ของ SA node ที่กระตุ้นเซลของหัวใจ เพราะว่าทุกๆ ส่วนของกล้ามเนื้อหัวใจ และระบบนำไฟฟ้าทำงานอย่างเป็นปกติ ดังรูป 2.10 แต่เมื่อบางส่วนของหัวใจเกิดการกระตุ้นที่ผิดปกติขึ้นในบางสถานะการณ์ ก็จะทำให้จังหวะการเต้นผิดแปลกไปจากจังหวะปกติ ซึ่งผลของความผิดปกตินี้ก็จะสะท้อนออกมาทางคลื่นไฟฟ้าหัวใจให้เห็นได้ ความผิดปกติของจังหวะการเต้น มีศัพท์ทางการแพทย์ เรียกกันว่า "Arrhythmia" โดยความหมายของศัพท์คำนี้ แปลว่าไม่มีจังหวะ แต่ที่ใช้กันอยู่นี้ หมายถึง ความผิดปกติของจังหวะ (abnormal rhythm) ซึ่งมีศัพท์ที่มีความหมายในทำนองเดียวกันนี้ที่พบบ่อยๆ เช่น rhythm disturbance และ dysrhythmia

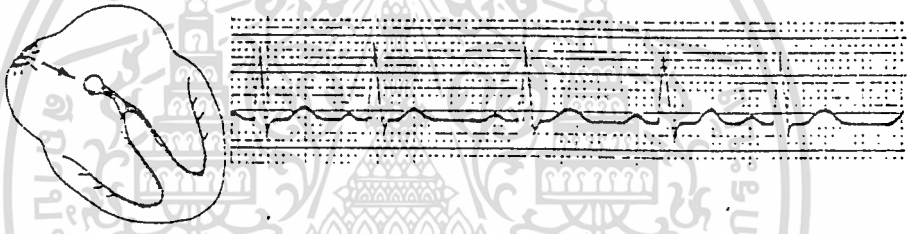


รูป 2.10 สัญญาณไฟฟ้าหัวใจที่มีลักษณะรูปร่างและจังหวะการเต้นเป็นปกติ

ความผิดปกติของจังหวะการเต้นของหัวใจอาจจะแบ่งเป็น 4 กลุ่มใหญ่ๆ ทำให้สามารถจำแนกลักษณะอาการและเข้าใจกลไกของความผิดปกติที่เกิดขึ้นได้ง่ายและรวดเร็วขึ้น. ดังต่อไปนี้

2.6.1 Varying Ryhthm เป็นลักษณะความผิดปกติของจังหวะการเต้นของหัวใจกลุ่มหนึ่ง ซึ่งมีลำดับของสัญญาณ คือ P-QRS-T เป็นปกติธรรมดา แต่จังหวะการเต้น (คือช่วงเวลาระหว่างสัญญาณ P-QRS-T ในแต่ละรอบ) นั้นเกิดการเปลี่ยนแปลงอย่างต่อเนื่องไม่สม่ำเสมอ ลักษณะอาการที่จัดอยู่ในกลุ่มดังกล่าวนี้ ได้แก่

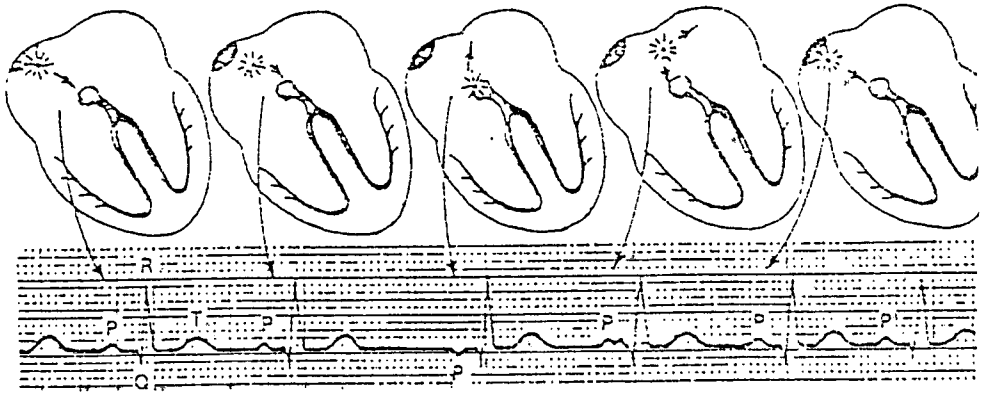
2.6.1.1 Sinur Arrhythmia เป็นลักษณะความผิดปกติของจังหวะการเต้นที่จัดอยู่ในกลุ่มของ Vary Rhythm ซึ่งบ่อยครั้งที่สาเหตุมาจากโรคหัวใจ Sick Sa Node disease ลักษณะความผิดปกติของคลื่นไฟฟ้าหัวใจในประเภทนี้ จะมีข้อมูลสังเกตุได้ คือสัญญาณ P-QRS-T



รูปที่ 2.11 Sinus Arrhythmia

มีลักษณะเป็นปกติทั้งขนาดและรูปร่าง แต่ช่วงเวลาระหว่างแต่ละรอบของสัญญาณเหล่านี้จะไม่สม่ำเสมอ ซึ่งเป็นลักษณะที่ผิดปกติไปจากธรรมดา ดังรูป 2.11 เนื่องจากการกระตุ้น (pacemaker) ทุกครั้งจะเกิดขึ้นที่บริเวณ SA node (สังเกตุได้จากสัญญาณ P) แต่การกระตุ้นที่เกิดขึ้นจะไม่สม่ำเสมอ ทำให้สัญญาณที่ส่งออกไปกระตุ้นเซลอื่นคลาดเคลื่อนจากช่วงเวลาปกติ เป็นผลให้จังหวะการเต้นของหัวใจไม่สม่ำเสมอเหมือนปกติ

2.6.2 Wandering Pacemaker เป็นลักษณะความผิดปกติของจังหวะการเต้นประเภทหนึ่ง ในกลุ่มของ Vary Rhythm ที่มีสาเหตุมาจากการเปลี่ยนตำแหน่งของการกระตุ้นภายในหัวใจ ซึ่งสามารถสังเกตุได้จากการเปลี่ยนแปลงรูปร่างของสัญญาณ P การเปลี่ยนตำแหน่งของการกระตุ้นนี้ส่งผลให้ผลรวมของจังหวะการเต้นของหัวใจผิดปกติไป ดังรูป 2.12 เนื่องจากตำแหน่งของการกระตุ้นเปลี่ยนจากจุดหนึ่งไปยังอีกจุดหนึ่งในบริเวณหัวใจห้องบน

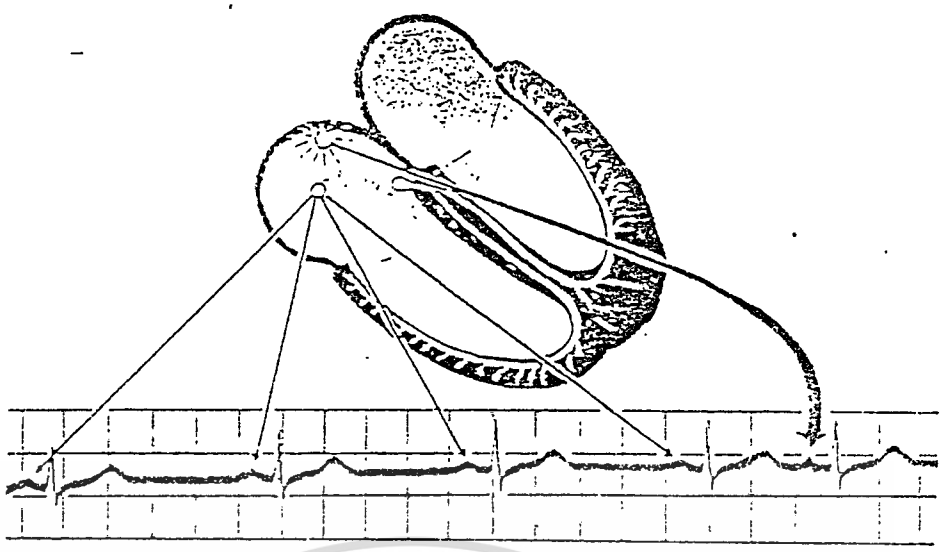


รูป 2.12 Wandering Pacemaker

2.6.2 Extra Beats and Skips เป็นลักษณะความผิดปกติของจังหวะการเต้นของหัวใจกลุ่มหนึ่งที่สามารถจำแนกลักษณะความผิดปกตินั้นด้วยสายตาได้โดยง่าย ศัพท์คำว่า "Extra Beats" หมายถึง สัญญาณไฟฟ้าหัวใจที่เกิดขึ้นก่อนกำหนดที่คาดไว้ ส่วนศัพท์คำว่า "Skips" หมายถึงสัญญาณไฟฟ้าหัวใจที่ขาดหายไปจากเวลาที่คาดว่าจะพบสัญญาณนี้ ทำให้เกิดพื้นที่ของ baseline ที่ว่างไว้ การจำแนกความผิดปกติในลักษณะนี้ สามารถสังเกตุความแตกต่างระหว่างสัญญาณไฟฟ้าหัวใจปกติได้จากสัญญาณรวม QRS และช่วงเวลาระหว่างสัญญาณไฟฟ้าหัวใจลูกหนึ่งไปอีกลูกหนึ่ง (pause)

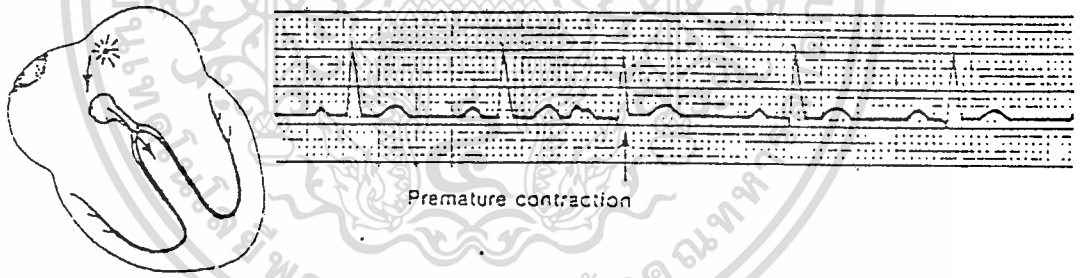
2.6.2.1 Premature Beats เป็นผลเนื่องจากการเกิดสัญญาณกระตุ้นก่อนที่จะถึงรอบการทำงานของหัวใจ ทำให้เกิดสัญญาณไฟฟ้าหัวใจปรากฏขึ้นก่อนเวลาปกติ ในแต่ละรอบการทำงานดังรูป 2.13 ซึ่งความผิดปกติในลักษณะนี้ สัญญาณไฟฟ้าหัวใจอาจดูเหมือนสัญญาณปกติหรือมีรูปแบบที่ผิดแปลกแตกต่างออกไป แต่สัญญาณที่ผิดปกตินี้จะเกิดก่อนเวลาในแต่ละรอบการทำงานของหัวใจ โดยความผิดปกติในลักษณะนี้สามารถจำแนกย่อยตามบริเวณที่เกิดการกระตุ้นได้ ดังนี้

- Atrial Premature เป็นลักษณะความผิดปกติของจังหวะการเต้นของหัวใจที่เกิดขึ้นบริเวณหัวใจห้องบน เนื่องจากเกิดการกระตุ้นก่อนกำหนดในบริเวณนี้ ทำให้เกิดสัญญาณ P ก่อนเวลา



รูป 2.13 สัญญาณไฟฟ้าหัวใจที่มี premature beat เกิดขึ้น

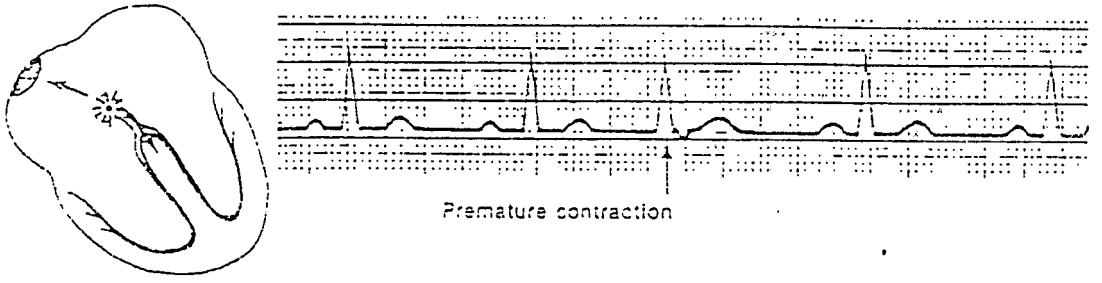
ปกติและสัญญาณ P นี้จะมีรูปร่างผิดแปลกไปไม่เหมือนสัญญาณ P ที่วัดได้ด้วยลีดเดียวกัน เนื่องจากสัญญาณกระตุ้นไม่ได้เกิดจากบริเวณ SA node แต่ก็จะเกิดการนำไฟฟ้าภายในหัวใจ เช่นเดียว กับการกระตุ้นที่บริเวณ SA node ปกติ ดังรูป 2.14



รูป 2.14 Atrual Premature Contractions

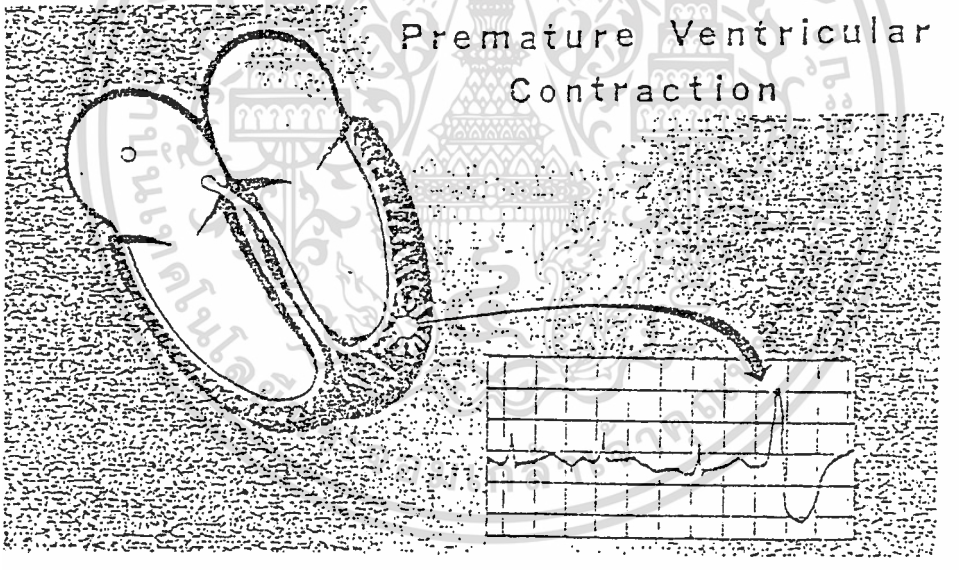
- AV NOdal Premature เป็นลักษณะความผิดปกติของจังหวะการเต้น เนื่องจากเกิดการกระตุ้นที่ผิดปกติบริเวณ AV node ก่อนที่จะเริ่มรอบการทำงานของหัวใจ ลักษณะเช่นนี้ทำให้เกิดสัญญาณรวม QRS ขึ้นก่อน โดยไม่มีสัญญาณ P ปรากฏขึ้นก่อน หรืออาจจะเกิดหลังจากสัญญาณรวม QRS เนื่องจากเกิดการกระตุ้นย้อนกลับไปยังบริเวณหัวใจห้องบนได้ดังรูป 2.15

- Premature Ventricular Contractions (ใช้ชื่อย่อว่า PVCs หรือ PVC) เป็นลักษณะความผิดปกติของจังหวะการเต้น เนื่องจาเกิดการกระตุ้นที่ผิดปกติบริเวณหนึ่งบริเวณใด



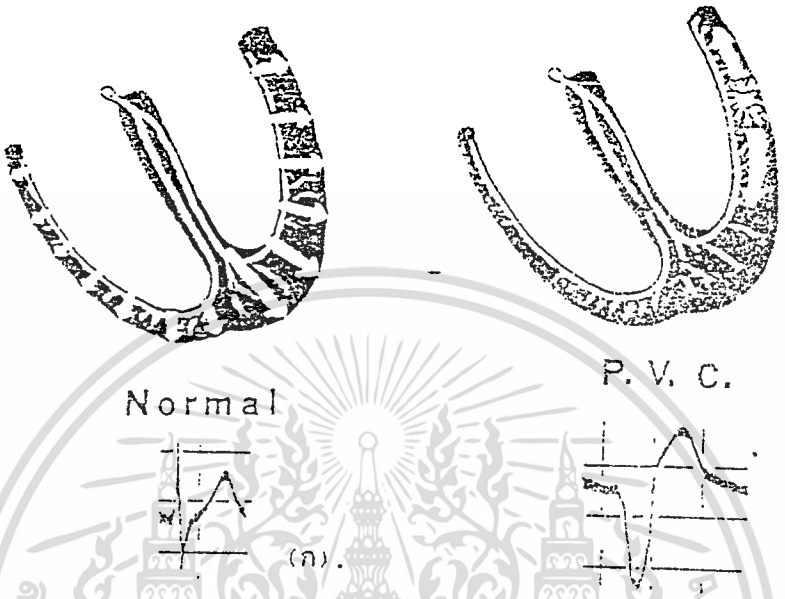
รูป 2.15 AV Nodal Premature Contractions

ของหัวใจห้องล่าง ซึ่งลักษณะการเกิดสัญญาณผิดปกติก็จะเกิดก่อนกำหนดของรอบการทำงานของหัวใจที่คาดไว้ โดยจะสังเกตุจากคลื่นไฟฟ้าหัวใจได้โดยง่าย ที่บริเวณสัญญาณรวม QRS ดังรูป 2.16



รูป 2.16 Premature Ventricular Contractions

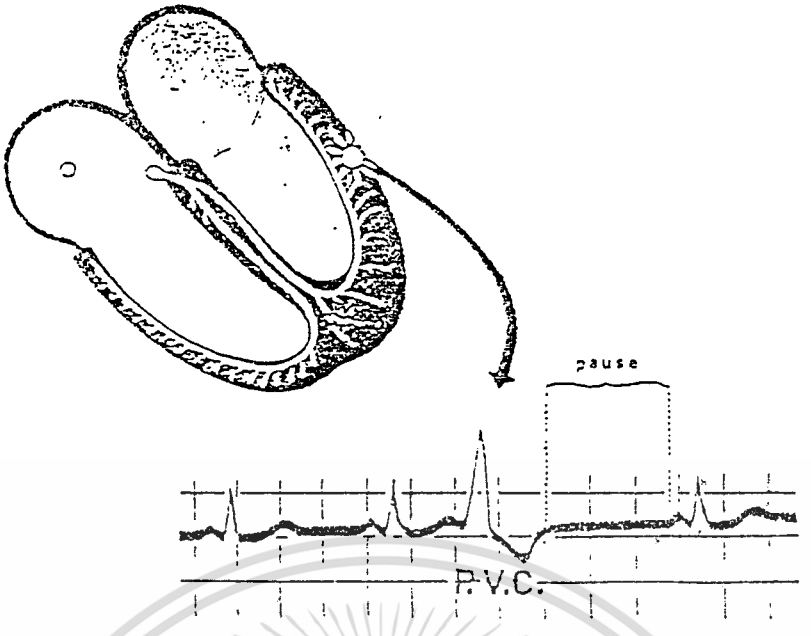
ซึ่งลักษณะแตกต่างจากสัญญาณไฟฟ้าหัวใจปกติอย่างเห็นได้ชัด เนื่องจากการกระตุ้นก่อนกำหนดของ PVC ทำให้การนำไฟฟ้าต่อไปยังระบบนำไฟฟ้าของ Bundle Branch ช้าลงเป็นผลให้ช่วงกว้างของสัญญาณรวม QRS เพิ่มมากขึ้น ดังรูป 2.17 ในระหว่างการนำไฟฟ้าปกติบริเวณหัวใจห้องล่างนั้น จะเกิดการ Depolarization ทันทีทั้งด้านซ้ายและด้านขวา โดยมีทิศทางพุ่งออกตามทิศทางของด้านนั้นหรือตามทิศทางของลูกศร ดังรูป 2.17 (ก). เป็นผลให้ช่วงกว้างของสัญญาณรวม QRS แคบมาก



รูป 2.17 เปรียบเทียบสัญญาณ QRS ของสัญญาณไฟฟ้าหัวใจกับสัญญาณที่เกิด PVC

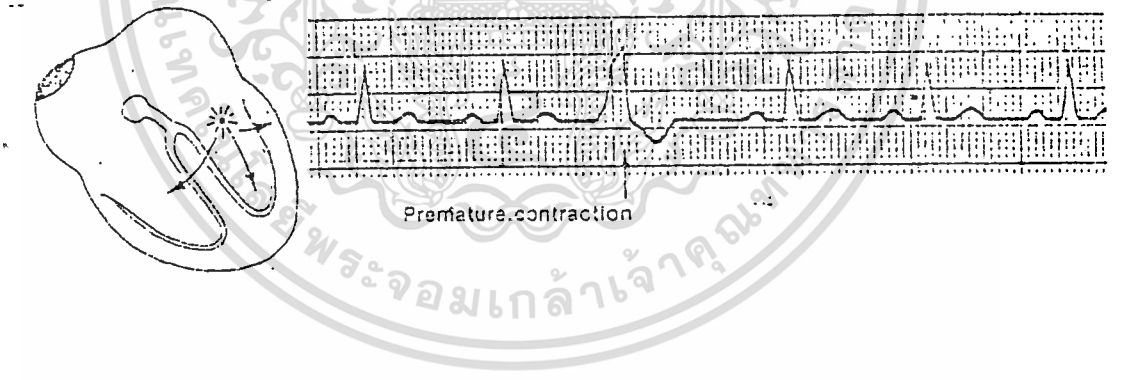
(ก) สัญญาณไฟฟ้าหัวใจปกติ (ข) สัญญาณที่เกิด PVC

แต่ถ้าเกิด PVC ขึ้นบริเวณด้านหนึ่งด้านใดของหัวใจห้องล่างจะทำให้เกิด Depolarization ก่อนอีกด้านหนึ่ง เป็นผลให้สัญญาณรวม QRS มีขนาดใหญ่และกว้างมากขึ้น ดังรูป 2.17 (ข). หลังจากเกิด PVC ขึ้นแล้วจะมีช่วงเวลาหนึ่งที่หัวใจจะหยุดการนำไฟฟ้าชั่วขณะหนึ่ง เรียกว่า Compensatory Pause ดังรูป 2.18



รูป 2.18 ลักษณะของ Compensatory pause หลังจากเกิด PVC

ลักษณะของ PVC ที่เกิดขึ้นนั้นมีหลายรูปแบบ ซึ่งแต่ละรูปแบบก็จะมีศัพท์ทางการแพทย์ที่ใช้เรียกแตกต่างกันไปตามรูปแบบนั้นๆ ที่เกิดขึ้นได้แก่

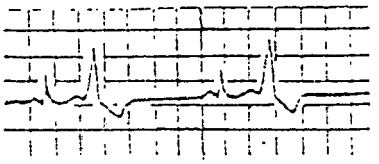


รูป 2.9 Interpolated PVCs

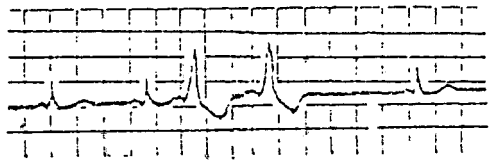
- Interpolated PVCs เป็น PVC ประเภทหนึ่งที่เกิดควบคู่กับสัญญาณไฟฟ้าหัวใจปกติ แต่ลักษณะของ PVC นี้จะไม่มี Compensatory pause เกิดขึ้น และไม่รบกวนจังหวะการเต้นของหัวใจปกติอีกด้วย ดังรูป 2.19

Bigemlny เป็น PVC ประเภทหนึ่งที่เกิดสัญญาณไฟฟ้าหัวใจปกติหนึ่งลูก ควบคู่กับสัญญาณ Premature Beat หนึ่งลูก และเกิดลักษณะเช่นนี้ซ้ำๆ กันขึ้น ดังรูป 2.20 (ก)

Trigeminy เป็น PVC ประเภทหนึ่งที่เกิดสัญญาณไฟฟ้าหัวใจปกติหนึ่งลูก ควบคู่กับสัญญาณ Premture Beats สองลูกติดกัน และเกิดลักษณะเช่นนี้ซ้ำๆ กันขึ้น ดังรูป 2.20 (ข)



รูป 2.20 (ก) Bigeminy

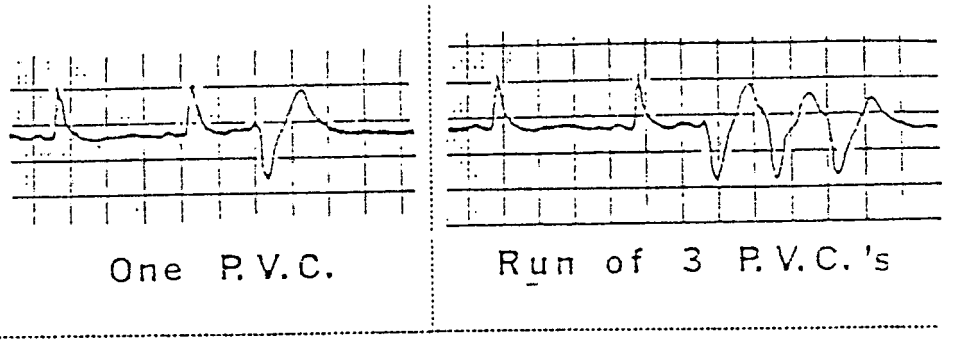


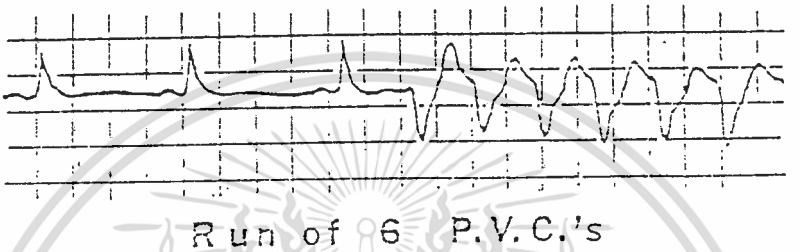
(ข) Trigeminy

Run of n. PVCs เป็น PVC ประเภทหนึ่งที่เกิดจากการกระตุ้นก่อนกำหนด หนึ่งครั้งเป็นผลให้การกระตุ้นตามมาอีกหลายครั้ง (n = จำนวนครั้งที่เกิดความผิดปกติ) ดังรูป 2.21

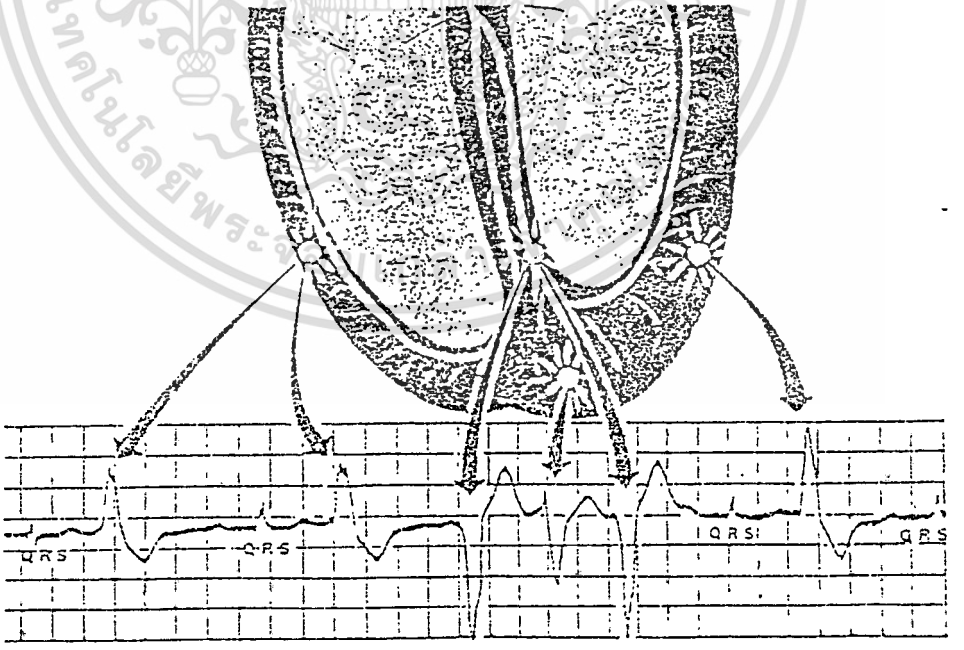
Multlfocal PVCs เป็น PVC ประเภทหนึ่งที่เกิดจากการกระตุ้นก่อนกำหนดที่บริเวณหัวใจห้องล่างหลายๆ จุด เป็นผลให้เกิดการกระตุ้นลุกลามต่อไปจนถึงขั้น Ventricular Tachycardia หรือ Ventricular Fibrillation ดังรูป 2.22

R-on-T Phenomenon เป็น PVC ประเภทหนึ่งที่เกิดขึ้นในช่วงเวลาที่เกิดสัญญาณ T ซึ่งเป็นช่วงเวลาที่หัวใจเกิด Rpolarization ดังรูป 2.23 ซึ่งถือได้ว่าเป็นลักษณะความผิดปกติที่รุนแรงอีกประเภทหนึ่ง

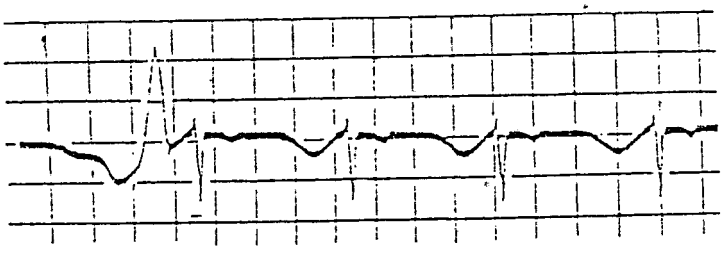




รูป 2.21 Run of n. PVCs

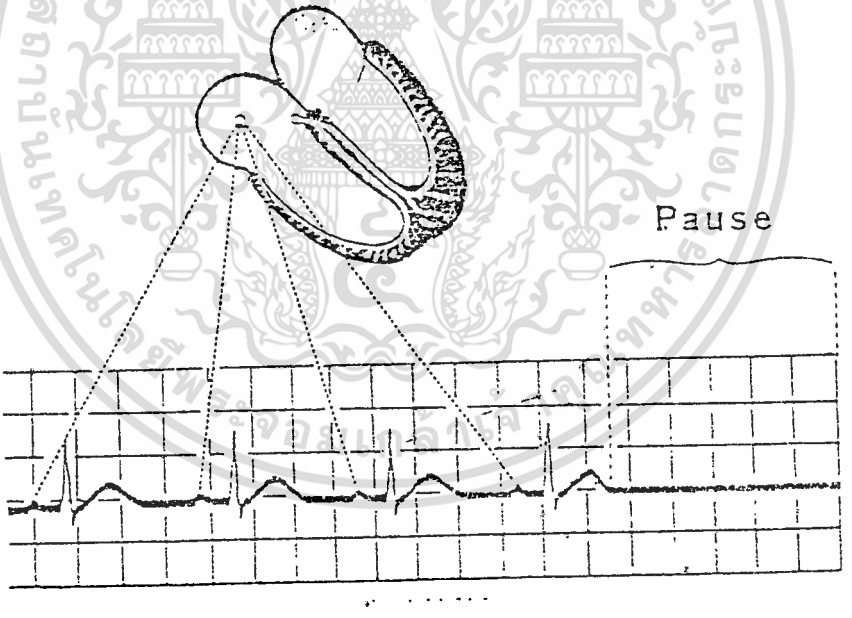


รูป 2.22 Multifocal PVCs



รูป 2.23 R-on-T Phenomenono

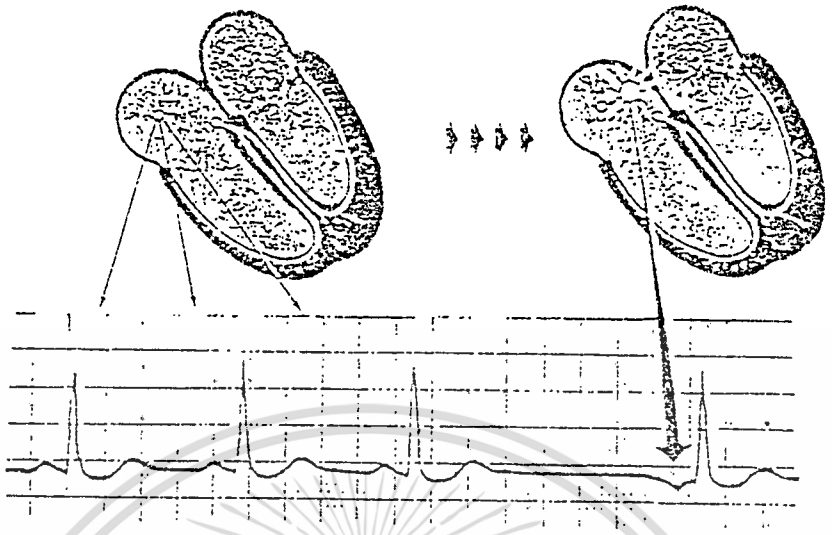
2.6.2.2 Escape Beats เกิดขึ้นเมื่อการกระตุ้นปกติล้มเหลว ทำให้การกระตุ้นหยุดไปหนึ่งหรือมากกว่าหนึ่งรอบการทำงาน เป็นผลให้เกิดการขาดช่วงของสัญญาณไป ดังรูป 2.24 ความผิดปกติในลักษณะนี้สามารถจำแนกตามบริเวณที่เกิดความผิดปกติได้ดังนี้



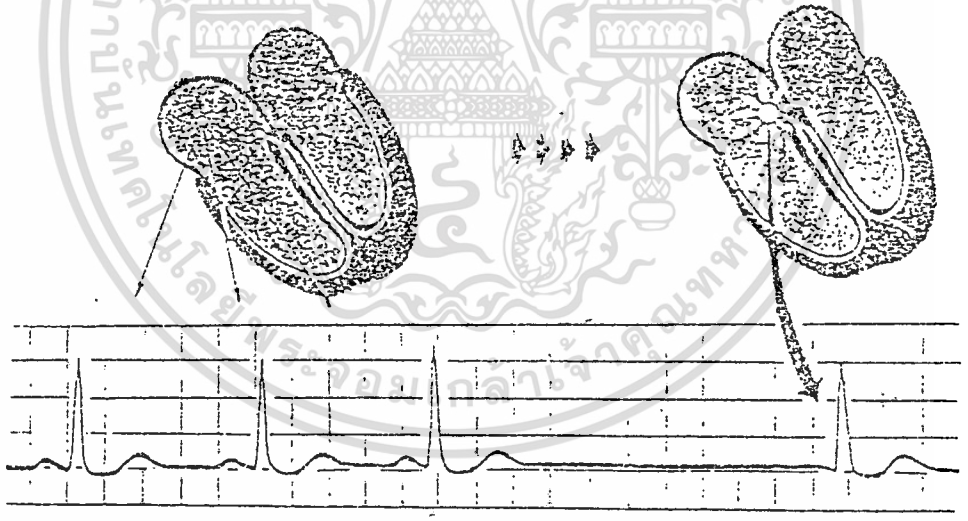
รูป 2.24 สัญญาณไฟฟ้าหัวใจที่เกิด Excape Beat ขึ้น

- Atrial Escape เป็นความผิดปกติที่มีลักษณะเหมือนกับ Atrial Premature แต่เกิดขึ้นหลังจากการขาดช่วงสัญญาณ (pause) มาแล้ว ดังรูป 2.25

- AV Nodal Escape เป็นความผิดปกติที่มีลักษณะเหมือนกับ AV Nodal Premature แต่เกิดขึ้นหลังจากการขาดช่วงสัญญาณ (pause) มาแล้วดังรูป 2.26



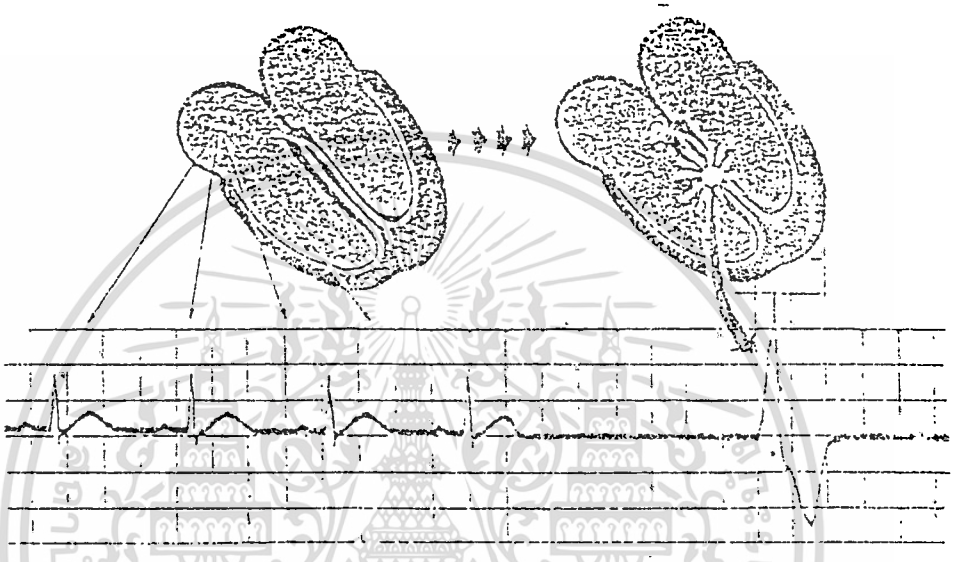
รูป 2.25 Atrial Escape



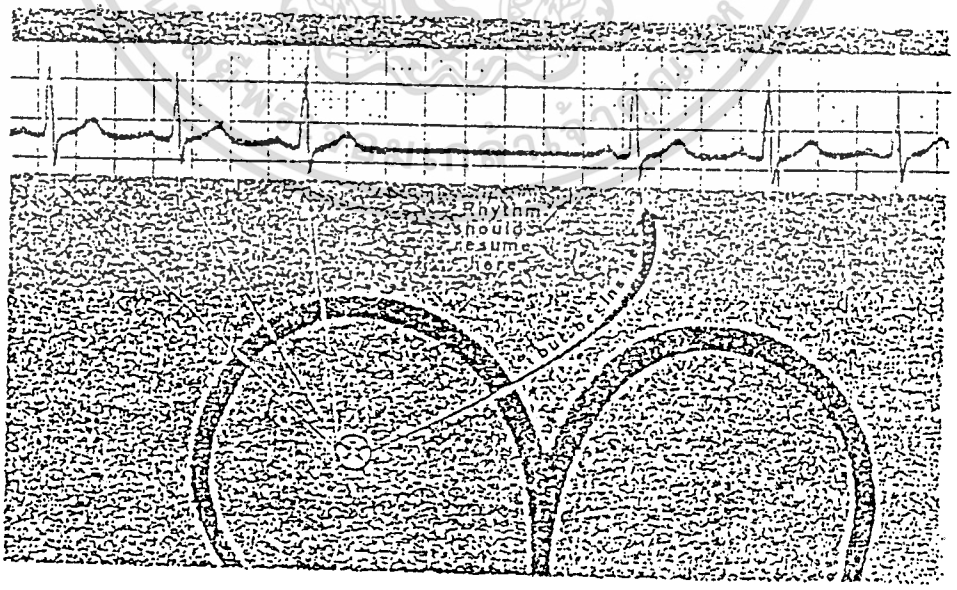
รูป 2.26 AV Nodal Escape

- Ventricular Escape เป็นความผิดปกติที่มีลักษณะเหมือนกับ Premature Ventricular Contractions แต่เกิดขึ้นหลังจากการขาดช่วงสัญญาณ (pause) มาแล้ว ดังรูป 2.27

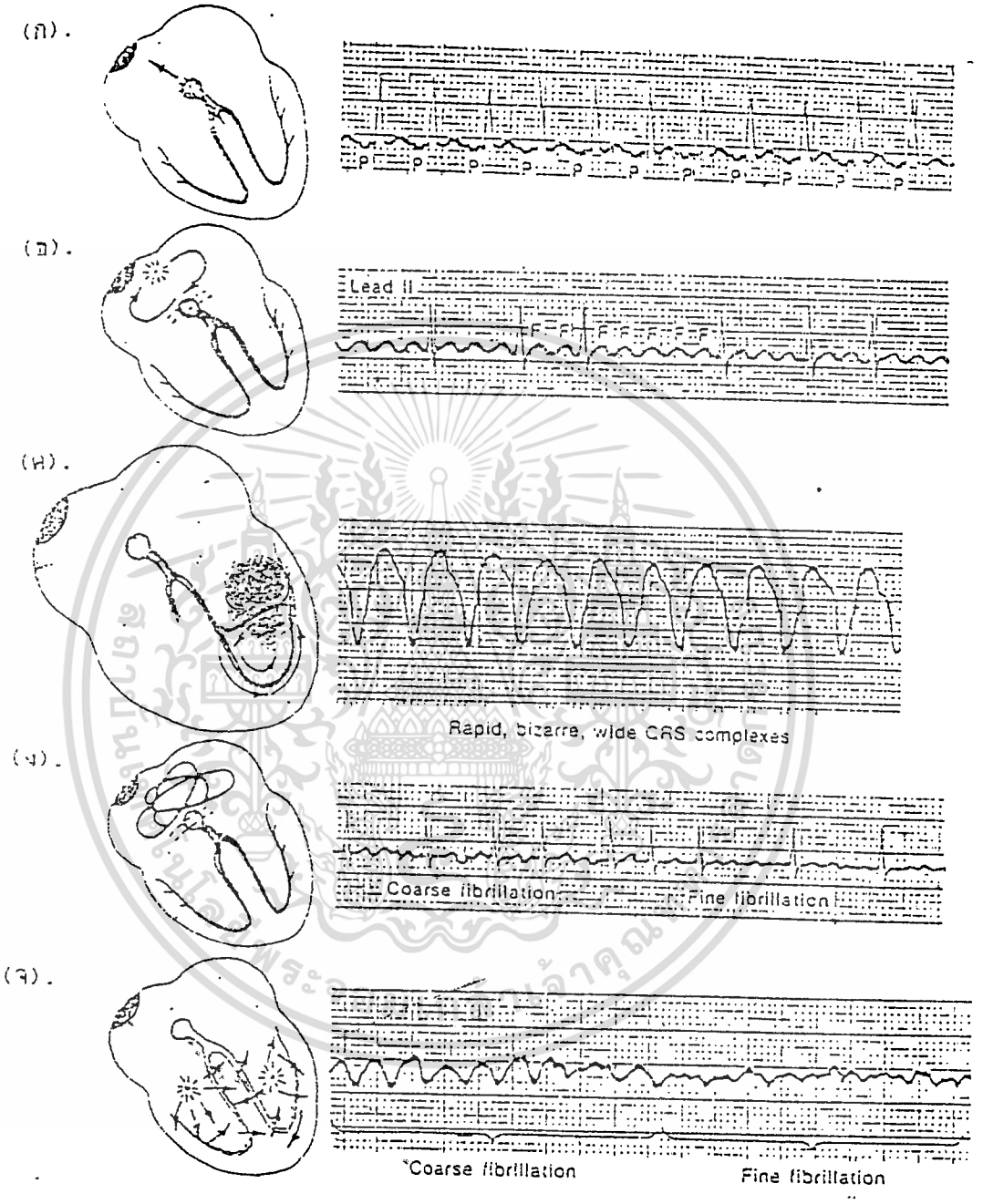
2.6.2.3 Sinus Arrest เกิดขึ้นเมื่อบริเวณกระตุ้นของ SA node หยุดการทำงานไปชั่วขณะหนึ่ง ทำให้ไม่สามารถส่งสัญญาณกระตุ้นออกมาได้ หลังจากการขาดช่วงของการกระตุ้นนี้ก็จะเกิดการกระตุ้นของบริเวณ SA node ใหม่ทำให้เกิดการเต้นของหัวใจด้วยอัตราการเต้นของหัวใจใหม่ ดังรูป 2.28



รูป 2.27 Ventricular Escape



รูป 2.28 Sinus Arrest



รูป 2.29 Rapic Rhythm

(ก). Paroxysmal Tachycardia (ข). Atrial Fultter

(ค). Ventricular Flutter (ง). Atrial Fibrillation

(จ). Ventricular Fibrillation

2.6.3.1 Paroxysmal (Sudden) Tachycardia

2.6.3.2 Atrial Flutter

2.6.3.3 Ventricular Flutter

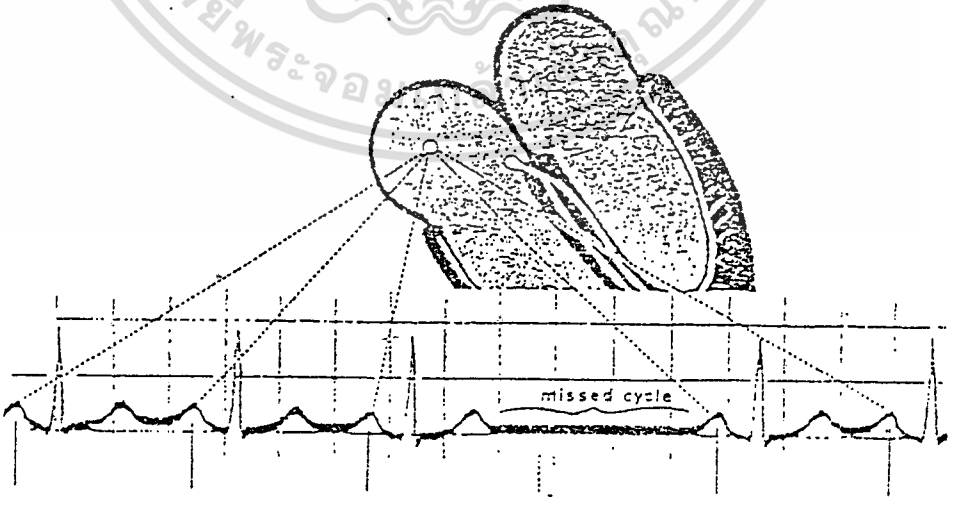
2.6.3.4 Atrial Fibrillation

2.6.3.5 Ventricular Fibrillation

2.6.3 Rapid Rhythm เป็นลักษณะความผิดปกติที่มีสาเหตุจากการมีอัตราการเต้นของหัวใจเร็วมาก โดยที่รูปร่างและลักษณะของสัญญาณไฟฟ้าหัวใจอาจจะผิดปกติหรือไม่นั้นขึ้นอยู่กับบริเวณที่เกิดความผิดปกตินั้น ซึ่งลักษณะความผิดปกติประเภทนี้มีความแตกต่างกันในเรื่องของรูปร่างของสัญญาณ เนื่องจากบริเวณที่เกิดความผิดปกติต่างกัน แต่ข้อสังเกตุหลักอยู่ที่การมีอัตราการเต้นของหัวใจที่เร็วมาก ดังนั้นจึงจะขอขยายความในบางหัวข้อเท่านั้น ซึ่งลักษณะความผิดปกติในแต่ละประเภทสามารถจำแนกได้จากรูป 2.29 โดยมีหัวข้อย่อยดังต่อไปนี้

2.6.4 Heart Blocks เป็นการปิดกั้นการนำไฟฟ้าภายในหัวใจซึ่งพื้นที่ภายในหัวใจที่จะเกิด การปิดกั้นของสัญญาณกระตุ้นได้มีด้วยกัน 3 บริเวณคือ SA node , AV node และ Bundle Branch

2.6.4.1 SA Block มีสาเหตุมากจากการที่ SA node หยุดการกระตุ้นชั่วคราว เป็นเวลาอย่างน้อยหนึ่งรอบการทำงานของหัวใจ แล้วกลับมาทำงานเป็นปกติเหมือนเดิม ดังรูป 2.30

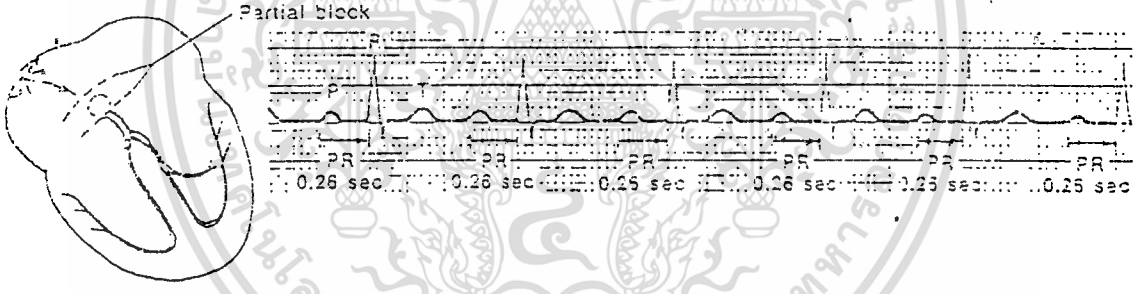


รูป 2.30 SA Block

ซึ่งจะเห็นได้ว่าสัญญาณไฟฟ้าหัวใจเกิดขาดช่วงขึ้น โดยจะปรากฏให้เห็นเพียง Baseline เท่านั้น

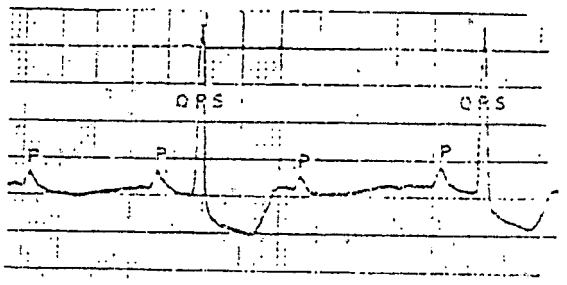
2.6.4.2 AV Block เป็นความผิดปกติของจังหวะการเต้น ที่จะทำให้เกิดการหน่วงเวลาของสัญญาณกระตุ้นจากหัวใจห้องบนที่บริเวณ AV node ให้นานยิ่งขึ้น ก่อนที่จะส่งต่อไปยังหัวใจห้องล่าง เวลาที่หน่วงนี้สามารถสังเกตุได้จากสัญญาไฟฟ้าหัวใจในช่วงเวลา PR ซึ่งเป็นเวลาตั้งแต่เริ่มต้นของสัญญาณ P (เริ่มเกิดจากการกระตุ้นบริเวณ SA node) จนกระทั่งถึงเวลาเริ่มต้นของสัญญาณรวม QRS (สัญญาณถึงหัวใจห้องล่าง) โดยมีค่าเฉลี่ยอยู่ระหว่าง 0.12-0.20 วินาที ถ้าช่วงเวลา PR มากกว่า 0.20 วินาทีแล้ว ก็สามารถสันนิษฐานได้ว่าเกิด AV Block ขึ้นอย่างแน่นอน ลักษณะของ AV Block สามารถจำแนกได้ 3 ลักษณะดังนี้ คือ

- First degree AV Block มีลักษณะความผิดปกติที่สังเกตุได้จากสัญญาณไฟฟ้าหัวใจ คือ ช่วงเวลา PR มากกว่า 0.20 วินาที ดังรูป 2.31

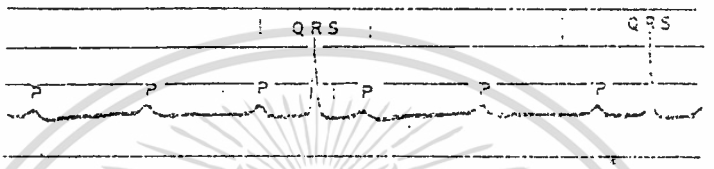


รูป 2.31 First degree AV Block

- Second degree AV Block มีลักษณะความผิดปกติที่สังเกตุได้จากสัญญาณไฟฟ้าหัวยใจ คือ จะเกิดสัญญาณ P หนึ่งหรือสองครั้งก่อนที่จะมีสัญญาณรวม QRS เพียงครั้งเดียว (2:1 หรือ 3:1 AV Block) ดังรูป 2.32 ซึ่งมีลักษณะความผิดปกติ 2 อาการที่สำคัญคือ Wenckebach Phenomenon (หรือ Mobitz 1)เกิดขึ้นเมื่อช่วงเวลา PR มีความกว้างเพิ่มมากขึ้นจนกระทั่ง AV node ไม่เกิดการกระตุ้นขึ้น ซึ่งทำให้ไม่เกิดสัญญาณรวม QRS ดังรูป 2.33 และ Mobitz ll เกิดขึ้นเมื่อสัญญาณไฟฟ้าหัวใจเกิดการขาดหาย (Dropped Beat) ไปของสัญญาณรวม QRS โดยที่ช่วงเวลา PR ยังคงมีความกว้างเป็นปกติอยู่ ดังรูป 2.34

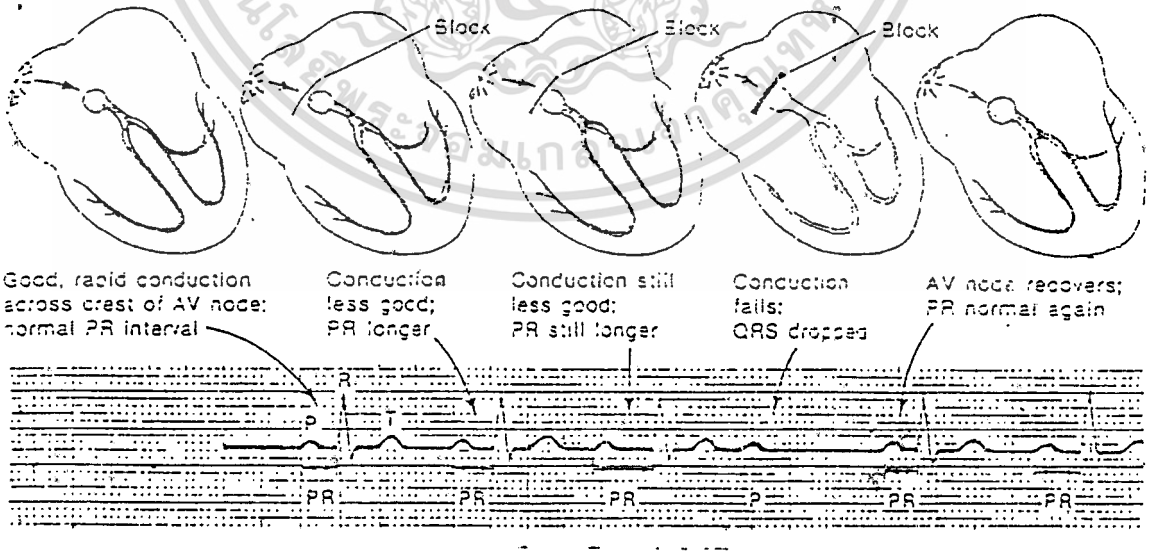


2:1 AV Block

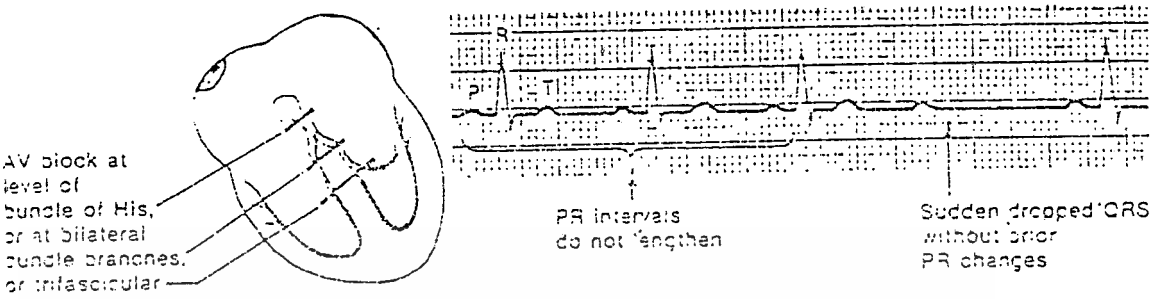


3:1 AV Block

รูป 2.32 2:1 และ 3:1 AV Block

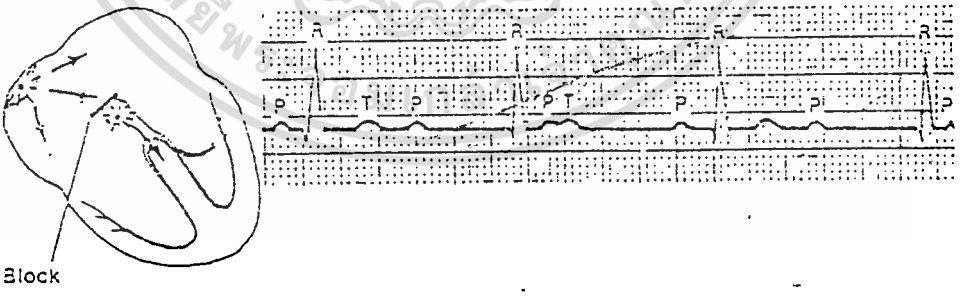


รูป 2.33 Wenckebach Phenomenon (Mobitz 1)



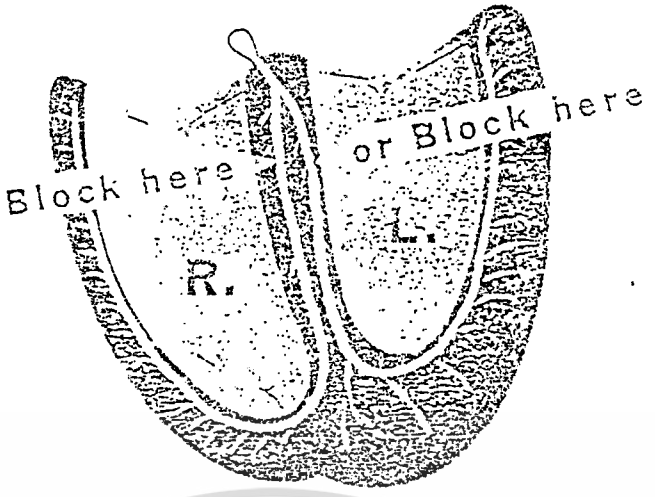
รูป 2.34 Mobitz II

- Third degree AV Block (Complete AC Block) เกิดขึ้นเมื่อไม่มีสัญญาณกระตุ้นจากหัวใจห้องบนมายัง AV node ทำให้ไม่เกิดการตอบสนองของหัวใจห้องล่างและบริเวณหัวใจห้องล่างจะเกิดการกระตุ้นขึ้นเองอย่างอิสระ ดังรูป 2.35 ซึ่งมีข้อสังเกตุได้ว่าความถี่ของหัวใจห้องบน (สัญญาณ P) มีค่าคงที่แน่นอน และความถี่ของหัวใจห้องล่าง (สัญญาณรวม QRS) เป็นไปอย่างอิสระและมีค่าช้าลงไปมาก



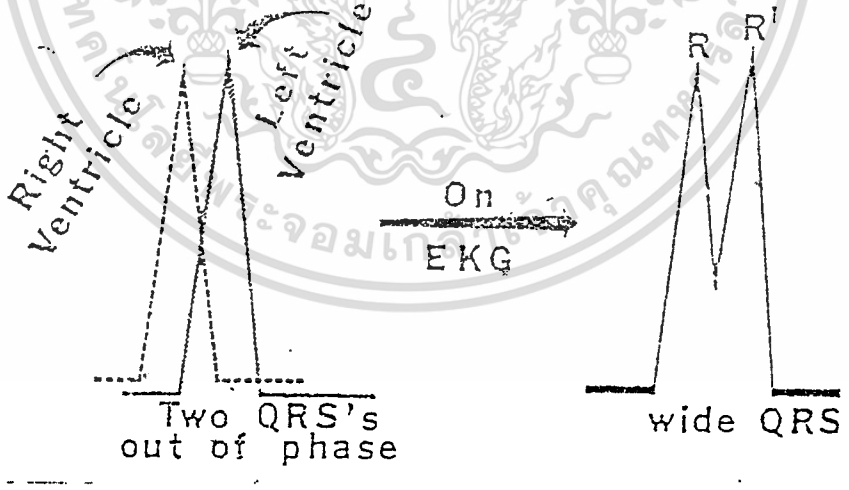
รูป 2.35 Third degree AV Block (Complete AV Block)

2.6.4.3 Bundle Branch Block มีสาเหตุมาจากการปิดกั้นการนำไฟฟ้าบริเวณด้านซ้ายและด้านขวาของ Bundle Branch ดังรูป 2.36 โดยปกติแล้ว การนำไฟฟ้าไปยังกล้ามเนื้อหัวใจห้องล่างทั้งด้านซ้ายและด้านขวา โดยผ่าน Bundle Branch จะใช้เวลาเท่าๆกัน

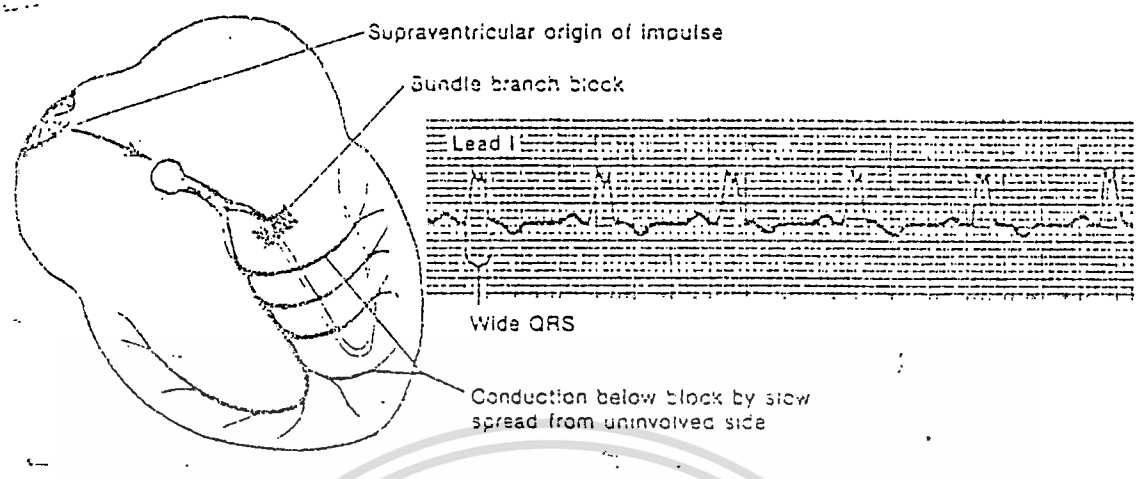


รูป 2.36 ลักษณะการปิดกั้นการนำไฟฟ้า (Block) บริเวณ Bundle Branch

ถ้าเกิดการปิดกั้นการนำไฟฟ้าขึ้นด้านใดด้านหนึ่ง ก็จะทำให้เกิดการหน่วงเวลาของสัญญาณด้านนั้นออกไป เป็นผลให้สัญญาณ QRS รวมแตกออกเป็น 2 ส่วนดังรูป 2.37 โดยให้สัญลักษณ์เป็น R และ R แทนจุดสูงสุดของสัญญาณ QRS ที่เกิดจากการนำไฟฟ้าของหัวใจห้องล่างแต่ละด้าน



รูป 2.37 ลักษณะของสัญญาณ QRS รวมที่เกิด Bundle Branch ที่แยกเป็น 2 ส่วน ซึ่งผลรวมของสัญญาณ QRS นี้จะทำให้ความกว้างของสัญญาณ QRS รวมกว้างมากขึ้น (widened QRS) รูป 2.38 แสดงลักษณะของสัญญาณไฟฟ้าหัวใจที่เกิด Bundle Branch Block



รูป 2.38 ลักษณะของสัญญาณไฟฟ้าหัวใจที่เกิด Bundle Branch Block

Asytole และ Ventricular Flbrillation

มีลักษณะของสัญญาณไฟฟ้าหัวใจที่เกิดการขาดหายของสัญญาณเป็นระยะเวลานาน กรณีของ Asystole ส่วน Ventricular Fibrillation มีลักษณะ สัญญาณที่มีรูปร่างไม่แน่นอน ซึ่งทั้งสองนี้ไม่สามารถที่จะตรวจสอบสัญญาณ R ได้

Bradycardia มีลักษณะของสัญญาณไฟฟ้าหัวใจ ที่มีอัตรการเต้นของหัวใจ ช้ากว่าค่าพิกัดมาตราฐาน

Techycardia มีลักษณะของสัญญาณไฟฟฟ้าหัวใจ ที่มีอัตราการเต้นของหัวใจเร็วกว่าค่าพิกัดมาตราฐาน

Dropped Beat มีลักษณะของสัญญาณไฟฟ้าหัวใจ ที่เกิดการขาดช่วงของสัญญาณไป ซึ่งช่วงเวลาไม่เกิดค่าพิกัดของ Asytole และไม่เกิด Premature Beat ตามมา

Premature Ventricular Contractions (PVCs) เป็นลักษณะของสัญญาณไฟฟ้าหัวใจที่เกิดก่อนกำหนดเวลาที่คาดไว้และตาม Compensatory Pause

R-on-T Phenomenon เป็นลักษณะของ PVCs ที่เกิดขึ้นในช่วงเวลาที่เกิดสัญญาณ T

- Bigeminy เป็นลักษณะของ PVCs ที่เกิดสัญญาณไฟฟ้าหัวใจปกติหนึ่งลูกควบคู่กับ Premature Beat หนึ่งลูก

- Trigeminy เป็นลักษณะของ PVC ที่เกิดสัญญาณไฟฟ้าหัวใจปกติหนึ่งลูกควบคู่กับ Premature Beat 2 ลูกติดกัน

Interpolated PVCs เป็นลักษณะของ PVCs ที่ไม่ได้ตามด้วย Compensatory Pause

Atrial Premature Beats (APBs) เป็นลักษณะของสัญญาณไฟฟ้าหัวใจที่มี Premature Beat เป็นสัญญาณ P ดังนั้นการตรวจสอบ ในลักษณะของช่วงเวลา R-R จึงเป็นการตรวจสอบช่วงเวลาของ Compensatory Pause ซึ่งจะมีค่าอยู่ระหว่าง Compensatory Pause ของ PVCs กับของ Interpolated PVCs

บทที่ 3

ทฤษฎีไมโครคอนโทรลเลอร์ 8031

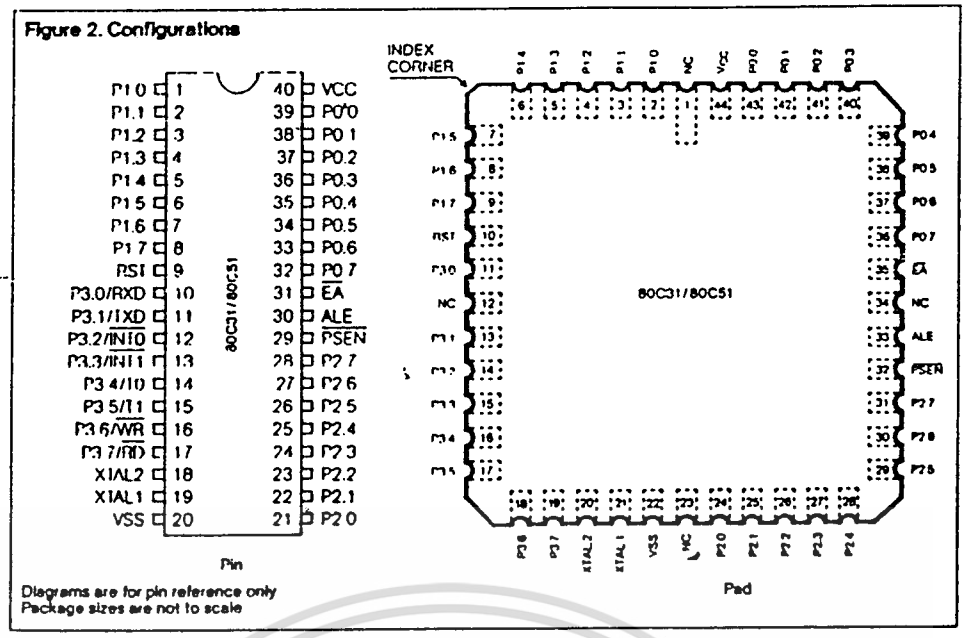
8031 เป็นไมโครคอนโทรลเลอร์แบบชิพเดี่ยว ซึ่งอยู่ในตระกูล MCS-51 ลักษณะโครงสร้างของ 8031 แสดงอยู่ในรูปที่ 3.1 ซึ่งสถาปัตยกรรมของ 8031 สร้างขึ้นด้วย HMOS

ลักษณะของ 8031

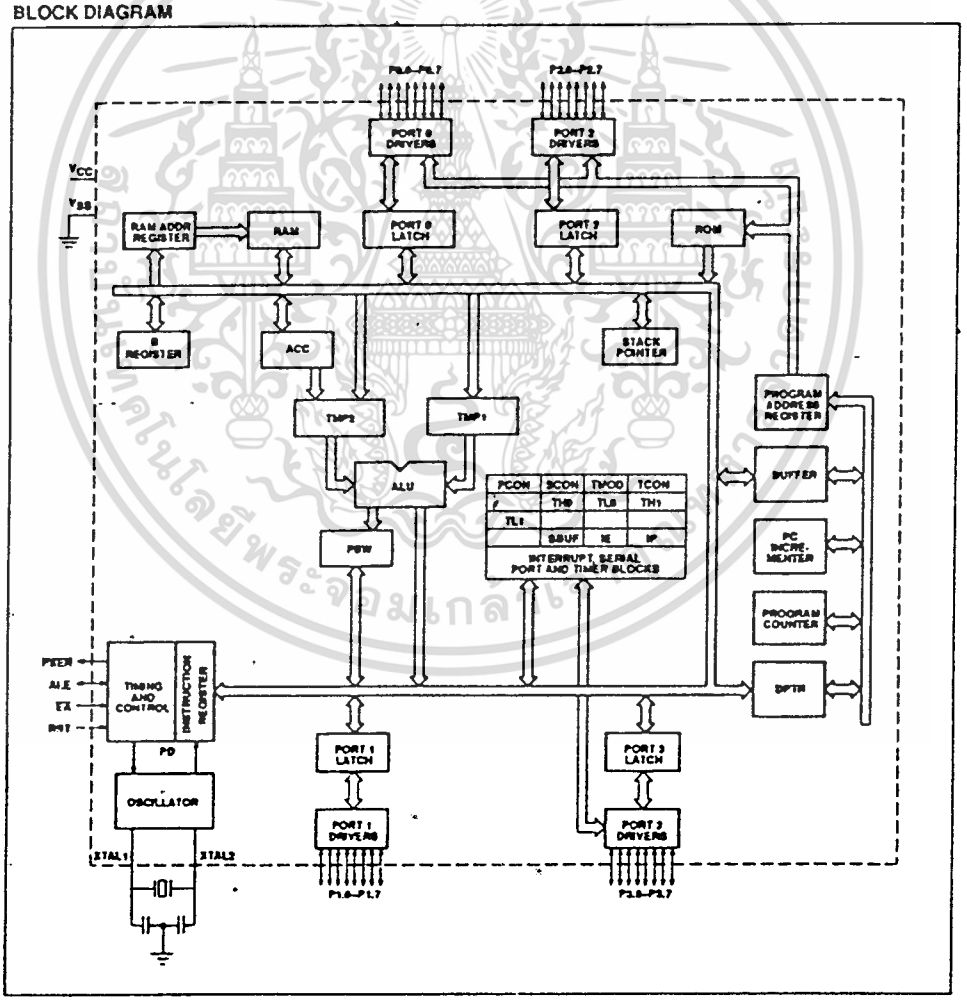
- เป็น CPU แบบ 8 บิท
- มีวงจรออสซิลเลเตอร์และ CLOCK อยู่ในตัว
- มีขาสัญญาณเข้าและออก (I/O) 32 ขา
- แยกหน่วยความจำของข้อมูลได้ 64K และหน่วยความจำของโปรแกรมอีก 64K
- มี TIMER และ COUNTER แบบ 16 บิท ถึง 2 ตัว
- สัญญาณอินเทอร์รัพท์ 6 แหล่ง 5 VECTOR ซึ่งแบ่งลำดับความสำคัญออกเป็น 2 ระดับ

การจัดหน่วยความจำของ 8031

8031 แบ่งหน่วยความจำสำหรับโปรแกรมและข้อมูลอย่างละ 64 K และยังมีหน่วยความจำประเภทแรมอีก 128 ไบท์ ซึ่งอยู่ในตัวของ 8031 และบน 8031 ยังประกอบด้วย รีจิสเตอร์ ที่ทำหน้าที่พิเศษ "SFRs" (SPECIAL FUNCTION RECISTOR) ซึ่งแสดงในตารางที่ 1



BLOCK DIAGRAM



รูปที่ 3.1 โครงสร้างของ 8031

* ตารางที่ 1.

SYMBOL	NAME	ADDRESS
* ACC	ACCUMULATOR	0E0H
* B	B REGISTOR	0F0H
* PSW	PROGRAM STATUS WORD	0D0H
SP	STACK POINTER	81H
DPTR	DATA POINTER (CONSIST OF DPH,PPL)	83H
DPL		82H
* P0	PORT 0	80H
* P1	PORT 1	90H
* P2	PORT 2	0A0H
* P3	PORT 3	0B0H
* IP	INTERUPT PIORITY CONTROL	0B8H
* IE	INTERUPT ENABLE CONTROL	0A8H
TMOD	TIMER/COUNTER MODE CONTROL	89H
+*T2CON	TIMER/COUNTER CONTROL	0C8H
TCON	TIMER/COUNTER 2 CONTROL	88H
TH0	TIMER0 (HIGH BYTE)	8CH
TL0	TIMER0 (LOW BYTE)	8AH
TH1	TIMER1 (HIGH BYTE)	8DH
TL1	TIMER1 (LOW BYTE)	8BH
+ TH2	TIMER2 (HIGH BYTE)	0CDH
+ TL2	TIMER2 (LOW BYTE)	0CCH
+ RCAP2H	TIMER/COUNTER2 CAPTURE (HI)	0CBH
+ RCAP2L	TIMER/COUNTER2 CAPTURE REG (LO)	0CAH
* SCON	SERIAL CONTROL	98H
SBUF	SERIAL DATA BUFER	99H
PCON	POWER CONTROL	87H

<u>หมายเหตุ</u> *SFRs ที่มีเครื่องหมายดอกจันทร์อยู่ข้างหน้า หมายความว่า สามารถเข้าถึงได้ โดยการอ่าน แอดเดรสแบบไบท์ และแบบบิท คือ สามารถจะเข้าถึงบิทใดบิทหนึ่งใน SFRs ได้โดยตรง

+ SFRs ที่มีเครื่องหมายบวกแสดงว่ามีอยู่ใน 8032/8052 เท่านั้น

รายละเอียดของรีจิสเตอร์ที่ทำหน้าพิเศษ

ACCUMULATOR

ACC เป็นแอดดิวมูเลเตอร์หรือรีจิสเตอร์ A

B REGISTER

รีจิสเตอร์ B ใช้ในคำสั่งคูณและหาร ส่วนในคำสั่งอื่นสามารถใช้เหมือนกับรีจิสเตอร์ทั่วๆ ไป ในการพักข้อมูล

PROGRAM STATUS WORD

PSW ประกอบด้วยรายละเอียดดังในรูปที่ 3.2

รูปที่ 3.2 PSE : PROGRAM STATUS WORD

(MSB) (LSB)

CY AC F0 RS1 RS0 0V - P

SYMBOL	POSITION	NAME AND SIGNIFICANCE
CY	PSW.7	CARRY FLAG
AC	PSW.6	AUXILLARY CARRY FLAG (FOR BCD OPERATION)
F0	PSW.5	FLAG 0 (FOR GENERAL PURPOSES)
RS1	PSW.4	REGISTOR BANK SELECT CONTROL BITS 1&0 SET/CLEARED BY SOLFWARE TO DETERMINE
RS0	PSW.3	WORKING REGISTOR BANK (SEE NOTE)
OV	PSW.2	OVERFLOW FLAG
-	PSW.1	(RESERVED)
P	PSW.0	PARITY FLAG. SET/CLEARED BY HARDWARE EACH INSTRUCTION TO INDICATE AN ODD/EVEN NUMBER OF "ONE" BITS INTHE ACCUMULATOR., ie. EVEN PARITY.

NOTE - THE CONSISTS OF (RS1, RS0) ENABLE THE WORKING REGISTOR BANKS AS FOLLOWS.

(0.0) - BANK 0	(00H-07H)
(0.1) - BANK 1	(08H-0FH)
(1.0) - BANK 2	(10H-17H)
(1.1) - BANK 3	(18H-1FH)

STACK POINTER

SP เป็นรีจิสเตอร์ขนาด 8 บิท เมื่อใช้คำสั่ง PUSH และ CALL SP จะเพิ่มขึ้นก่อนที่จะเก็บข้อมูลหรือแอดเดรส ซึ่ง STACK จะอยู่ที่ไหนก็ได้ในหน่วยความจำ (RAM) บน 8031 ภายหลังการ RESET แสตกจะมาอยู่ที่ 07H นั่นคือ แสตกจะเริ่มที่ 80H

DATA POINTER

DPTR ประกอบด้วย DPH และ DPL ซึ่งจะรวมกันเป็นรีจิสเตอร์ขนาด 16 บิท การเข้าถึง DPTR สามารถทำได้ทั้งแบบ 16 บิท และ แบบ 8 บิท หน้าที่ของ DPTR ถ้าใช้แบบ 16 บิท จะทำหน้าที่เป็นตัวชี้ข้อมูลที่อยู่ในหน่วยความจำภายนอก

PORT 0-3

พอร์ต 0, พอร์ต 1, พอร์ต 2, และพอร์ต 3 เป็น SFR ตัวหนึ่งที่สามารถแลตช์ข้อมูลได้สามารถใช้เป็นอินพุทพอร์ต และเอ้าท์พุทพอร์ตได้ทั้ง 3 พอร์ต

SERIAL DATA BUFFER

แบ่งเป็นรีจิสเตอ์บัฟเฟอร์ในทางส่งและบัฟเฟอร์ในทางรับ เมื่อมีการส่งข้อมูลภายในให้ SBUF ข้อมูลจะถูกส่งมาที่ บัฟเฟอร์ในทางส่ง ที่ซึ่งจะใช้ทำการส่งข้อมูลอนุกรม การส่งข้อมูลให้ SBUF จะเป็นการเริ่มต้นการส่งด้วย และถ้าอ่านข้อมูลจาก SBUF ข้อมูลจะถูกอ่านจาก บัฟเฟอร์ในทางรับ

TIMER REGISTERS

รีจิสเตอร์คู่ (THo , TLo) และ (TH1 , TL1) เป็นตัวจับเวบาและตัวนับขนาด 16 บิท จะกล่าวรายละเอียดในภายหลัง

CONTROL REGISTERS

รีจิสเตอร์หน้าที่พิเศษ IP, IE, TMOD, TCON, T2CON, SCON และ PCON ประกอบด้วยบิทควบคุมและบิทสถานะสำหรับระบบการอินเตอร์รัพท์

โครงสร้างและการทำงานของพอร์ต

8031 มี I/O พอร์ต อยู่ 4 พอร์ต โดยแต่ละพอร์ตจะเป็นพอร์ตแบบ 2 ทิศทาง มีการแลตช์ข้อมูลได้ (SFR.PO-P3) รวมทั้งมีวงจรขับทางด้านเอ้าท์พุท และบัฟเฟอร์ทางด้านอินพุท พอร์ต 0 และ พอร์ต 2 ใช้สำหรับติดต่อกับหน่วยความจำภายนอก ในการใช้ 8031 ติดต่อกับหน่วยความจำภายนอก

พอร์ต 0 จะให้เอ้าท์พุทเป็น LOW BYTE ของแอดเดรสของหน่วยความจำภายนอก และจะทำการ MULTIPLEX กับข้อมูลที่จะเขียนหรืออ่าน สรุปคือ PO จะเป็นทั้ง ADDRESS และ

DATA ส่วนพอร์ต 2 จะให้แอดเดรสไบท์สูง (MSB) ของหน่วยความจำภายนอก

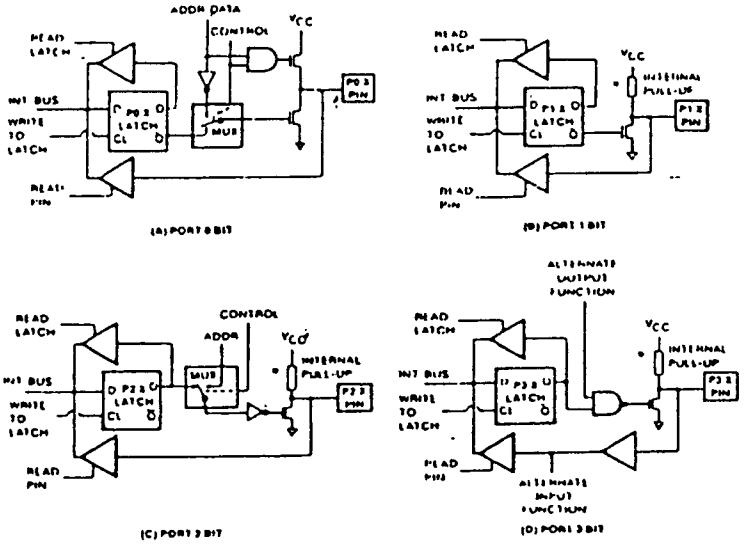
ขาของพอร์ต 3 ทั้งหมดกับอีก 2 บิท ของพอร์ต 1 (8052) จะทำงานหลายหน้าที่ ดังรายละเอียดในรูปที่ 3.3

* P1.0	T2 (TIMER/COUNTER 2 EXTERNAL INPUT)
* P1.1	T2EX (TIMER/COUNTER 2 CAPTURE/RELOAD TRIGGER)
P3.0	RXD (SERIAL INPUT PORT)
P3.1	TXD (SERIAL OUTPUT PORT)
P3.2	INT0 (EXTERNAL INTERUPT)
P3.3	INT1 (EXTERNAL INTERUPT)
P3.4	T0 (TIMER/COUNTER 0 EXTERNAL INPUT)
P3.5	T1 (TIMER/COUNTER 1 EXTERNAL INPUT)
P3.6	WR (EXTERNAL DATA MEMORY WRITE STROBE)
P3.7	RD (EXTERNAL DATA MEMORY READ STROBE)

หมายเหตุ* P1.0 และ P1.1 สงวนไว้สำหรับ 8052

โครงสร้างของพอร์ต

ในรูปที่ 3.3 แสดงโครงสร้างของ I/O ของแต่ละพอร์ต โดยแสดงเพียงพอร์ตละบิท ส่วนของพอร์ตแลตช์ใช้ D-FLIP FLOP ซึ่งจะรับข้อมูลมาจากพัสภายในโดยสัญญาณ "WRITE TO LATCH" ของ CPU ขา Q ของ D-FLIP FLOP จะถูกป้อนกลับไปที่บัสภายในเพื่อตอบส่นองต่อสัญญาณ "READ LATCH" จาก CPU เมื่อ CPU ต้องการอ่านพอร์ต ส่วนสัญญาณที่ภายนอกของพอร์ตจะถูกต่อเข้าขาข้อมูลภายในและขาพอร์ตภายนอกนี้จะถูกอ่านโดยตอบสนองต่อสัญญาณ "READ PIN" (ดูรูป 3.3 ประกอบ) ส่วนคำสั่งอื่นๆ จะอ่านใช้สัญญาณ "READ PIN" ดูรายละเอียดคำสั่งที่อ่านข้อมูลจาก READ LATCH ในรูปที่ 3.4



ANL	(LOGIC AND, e.g., ANL P1, A)
ORL	(LOGIC OR, e.g., ORL P2, A)
XRL	(LOGIC EX-OR e.g., XRL P3, A)
JBC	(JUMP IF BIT = 1 AND CLEAR BIT, e.g., JBC P1.1, LABEL
CPL	(COMPLEMENT BIT, e.g., CPL P3.0)
INC	(INCREMENT, e.g., INC P2)
DEC	(DECREMENT, e.g., DEC P2)
DJNZ	(DECREMENT AND JUMP IF NOT ZERO, e.g., DJNZ P3, LABEL
MOV PX.Y,C	(MOVE CARRY BIT TO BIT Y OF PORT X)
CLR PX.Y	(CLEAR BIT Y OF PORT X)
SET PX.Y	(SET BIT Y OF PORT X)

รูปที่ 3.4 คำสั่งอ่านจาก READ PIN

เหตุที่บางคำสั่งอ้องอ่านพอร์ตจาก "READ LATCH" เพราะในบางกรณี CPU อาจเข้าใจสัญญาณที่ขาของพอร์ตผิดพลาด ตัวอย่างเช่น PORT 1 ขาที่ 1 ถูกต่ออยู่กับขา BASE ของ TRANSISTOR และให้สัญญาณทางออกเป็น HIGH แก่ขา BASE ของทร่านซิสเตอร์ ในขณะที่ทรานซิสเตอร์นำกระแสจะทำให้แรงดันที่ขา BASE เหลือโดยประมาณ 0.6 โวลท์ ซึ่งขา BASE ถูกต่ออยู่กับพอร์ต เมื่อ CPU ทำการอ่านพอร์ จะทำให้เข้าใจสัญญาณที่ขาของพอร์ตผิดไปทั้งๆ ที่ขณะนั้นขาของพอร์ตเป็น HIGH อยู่

จากรูป 3.3 จะเป็นว่าพอร์ต 0 นอกจากจะเป็น I/O พอร์ตแล้วยังเป็นได้ทั้งแอดเดรสบัสและบัสข้อมูล ส่วนพอร์ต 2 เป็นทั้ง I/O พอร์ตและแอดเดรส ซึ่งควบคุมโดยสัญญาณภายในชิพ ในขณะทำการติดต่อกับหน่วยความจำภายนอก SFR ของพอร์ต 2 จะไม่เปลี่ยนค่า หลังจากให้แอดเดรสแล้ว แต่ SFR ของพอร์ต 0 จะเปลี่ยเป็น HIGH ทั้ง 8 บิท เพื่อที่จะรับข้อมูลที่จะส่งมาทางบัสข้อมูลของหน่วยความจำ (เนื่องจาก SFR สามารถแลตธ์ข้อมูลได้ถ้าไม่ทำให้เป็น HIGH อาจเกิดการผิดพลาดในการอ่านข้อมูลจากหน่วยความจำภายนอก)

พอร์ต 1, พอร์ต 2 และ พอร์ต 3 จะมีการพูลอัพภายใน ส่วนในพอร์ต 0 จะเป็นแบบ OPEN DRAIN OUTPUT แต่ในกรณีที่ใช้พอร์ต 1,2,3 เป็นอินพุท จะต้องทำให้แต่ละบิทเป็น HIGH ก่อนเพื่อทำให้ FET ที่อยู่ทางด้านทางออกหยุดนำกระแสและรักษาระดับสัญญาณเป็น HIGH จากความต้านทานพูลอัพภายใน (ดูรูป 3.3 ประกอบ)

ข้อแตกต่างของพอร์ต 0 ก็คือ ไม่มีการพูลอัพภายใน (ดูรูป (3A) ประกอบ) FET ตัวที่ทำหน้าที่พูลอัพ (ตัวบน) จะใช้เพียงให้เพ้าท์พุทเป็น 1 ในขณะที่ติดต่อกับหน่วยความจำภายนอก ในกรณีอื่นๆ FET ตัวนี้จะถูกทำให้ คัทออฟ ฉะนั้นพอร์ต 0 ที่ใช้เป็นเอ้าท์พุทพอร์ตจะเป็น OPENDRAIN

การเขียน 1 (FFH) ไปที่พอร์ต 0 จะทำให้ FET ทั้ง 2 ตัวหยุดทำงานผลคือทำให้ขาพอร์ต 0 ลอยและสามารถใช้เป็นขาอินพุทแบบความต้านหน่วงสูง (HI-Z INPUT) การรีเซ็ท 8031 จะทำให้ทุกพอร์ตมีระดับสัญญาณเป็น HIGH

<u>การติดต่อกับหน่วยความจำภายนอก</u>

การติดต่อกับหน่วยความจำภายนอกกระทำได้ 2 แบบ คือ ติดต่อกับโปรแกรมภายนอกกับติดต่อกับหน่วยความจำภายนอกที่เก็บข้อมูล การติดต่อกับโปรแกรมภายนอกจะใช้สัญญาณ PSEN (PROGRAM STORE ENABLE) เป็นสัญญาณอ่านโปรแกรมภายนอก ส่วนการติดต่อกับหน่วยความภายนอกที่เก็บข้อมูลจะใช้ RD(P3.7) และ WR(P3.6) เหมือนกับ PCU ทั่วๆ ไป

การติดต่อกับข้อมูลภายนอกจะใช้ได้ทั้ง 16 บิท (MOVX @DPTR) หรือ เป็นแบบ 8 บิท (MOVX @RI) เมื่อไรก็ตามที่ต้องใช้แอดเดรสขนาด 16 บิท แอดเดรสไบท์สูง (A8-A15) จะออกทางพอร์ต 2

เมื่อติดต่อกับหน่วยความจำของข้อมูลภายนอกแบบ 8 บิท (MOVX A,@RI) จะไม่มีผลกระทบกับ SFR ของพอร์ต 2 คือ SFR ของพอร์ต 2 จะยังคงแลตช์ข้อมูลเดิมเอาไว้ประโยชน์คือ สามารถใช้พอร์ต 2 กำหนดหน้า (PAGE) ของหน่วยความจำ

<u>การที่จะติดต่อกับหน่วยความจำภายนอกได้ขึ้นอยู่กับสภาวะอยู่ 2 ประการคือ</u>

1. เมื่อต่อขา EA ของ 8031 ลงกราวด์
2. เมื่อไรก็ตามที่โปรแกรมเคาท์เตอร์ (PC) มีค่ามากกว่า OFFFH

<u>สัญญาณ PSEN</u>

เมื่อมีการเฟชคำสั่งจากโปรแกรมภายนอก สัญญาณ PSEN จะแอคทีฟ 2 ครั้ง ทุกๆ 1 แมชีนไซเคิล (ยกเว้นคำสั่ง MOVX)

สัญญาณ ALE

จะเป็นขาสโตรบในการแลตช์แอเดรสไบท์ต่ำ เมื่อทำการติดต่อกับหน่วยความจำภายนอก ALE จะแอคทีฟ 2 ครั้งทุกๆ แมชีนไซเคิล ยกเว้นขณะทำคำสั่งที่ติดต่อกับข้อมูลในหน่วยความจำภายนอก (MOVX) ฉะนั้นในกรณีที่ระบบไม่ได้ใช้หน่วยความจำภายนอก (MOVX) การแอคทีฟของ ALE จะคงที่ในอัตรา 1/6 ของความถี่ออสซิลเลเตอร์ เพื่อใช้เป็น CLOCK ให้กับวงจรภายนอกได้ การใช้งานบางกรณีต้องการที่จะใช้ทั้งโปรแกรมและข้อมูลอยู่ในเพจเดียวกัน (64K) เราสามารถทำได้โดยการรวมสัญญาณ PSEN และ RD โดยใช้ AND GATE

TIMER/COUNTER

8031 มี TIMER/COUNTER อยู่ 2 ตัว คือ T0 และ T1 สัญญาณ INPUT ที่จะป้อนให้ COUNTER นั้นทำงานที่ขอบขาลง (1 TO 0) คือ ต้องเป็นพัลส์ HIGH 1 แมชีนไซเคิลและเป็น LOW1 แมชีนไซเคิล ฉะนั้นความถี่สูงสุด COUNTER จะนับได้นั้นประมาณ 1/24 ของความถี่ออสซิเลเตอร์ การทำงานของ TIMER/COUNTER แบ่งเป็น 3 โหมด ดังกล่าวต่อไปนี้

โหมด 0

การทำงานในโหมดนี้รีจิสเตอร์ถูกกำหนดให้เป็นแบบ 13 บิท โดยการนับจากค่าที่ทุกบิทเป็น HIGH ไปจนทุกๆ บิทเป็น 0 เกิด OVERFLOW และจะให้สัญญาณอินเตอร์รัพท์ โดยเซ็ทแฟลก TF0 หรือ TF1 การที่จะให้ TIMER/COUNTER ตัวใดอยู่ในโหมดใดนั้นกำหนดได้จากรีจิสเตอร์ TMOD (รูปที่ 3.5)

รูปที่ 3.5 TMOD : TIMER/COUNTER MODE CONTROL REGESTOR

(MSB) (LSB)

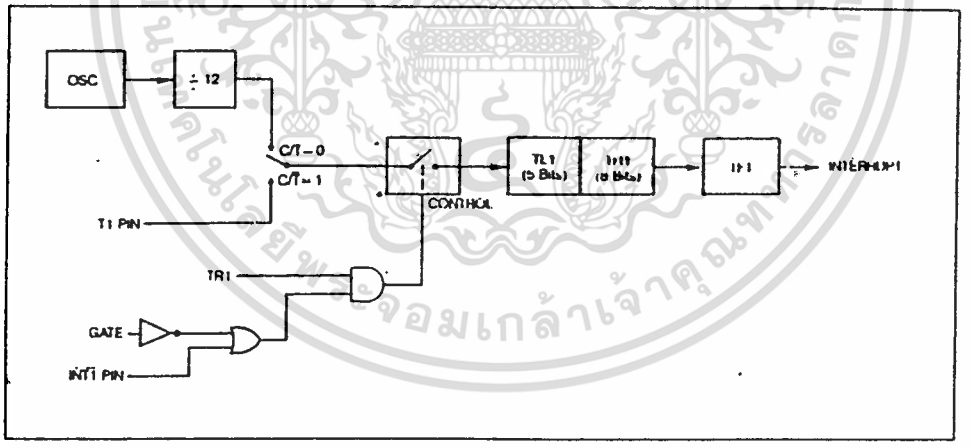
GATE	C/T	M1	M0	GATE	C/T	M1	M0
-------TIMER 1--------				--------TIMER 0-------			

GATE GATING CONTROL WHEN SET TIMER/COUNTER "X" IS ENABLE ONLY WHILE "INTx" PIN IS SET WHEN CLEARED TIMER "X" IS ENABLED WHENEVER"TRx"CONTROL BIT IS SET

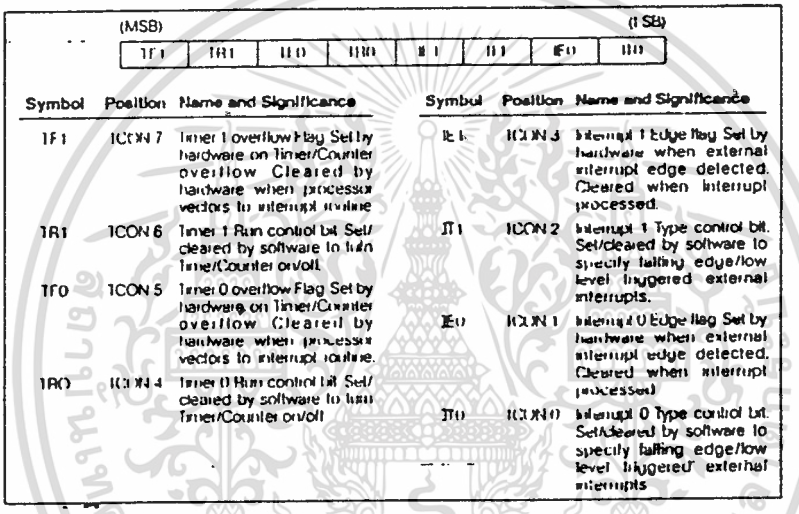
C/T TIMER OR COUNTER SELECTOR CLEAR FOR TIME OPERATION (INPUT FROM INTERNAL SYSTEM CLOCK) SET FOR COUNTER OPERATION (INPUT FROM "Tx" INPUT PIN)

M1	M0	OPERATION MODE
0	0	MCS-48 TIMER "TLX" SERVES AS FIVE BIT PRESCALER
0	1	16 BIT TIMER/COUNTER "THx" AND "TLx" ARE CADCADED THERE IS NO PRESCALE
1	0	8 BIT AUTO-RELOAD TIMER/COUNTER "THx" HOLDS A VALUE WHICH IS TO BE RELOADED INTO TLx EACH TIME IT OVERFLOWS
1	1	(TIMER 0) TL0 IS AN EIGHT BITTIMER C O U N T E R CONTROLED BY STANDARD TIMER 0 CONTROL BITS TH0 IS THE EIGHT BIT TIMER ONLY CONTROLED BY TIMER 1 CONTROL BIT
1	1	(TIMER 1) TIMER/COUNTER 1 STOPED

COUNTER จะทำงานได้ก็ต่อเมื่อ TR1 = 1 และ GATE=0 หรือ INT1 = 1 (ถ้าเซ็ท GATE = 1 TIMER/COUNTER) จะถูกควบคุมด้วยสัญญาณ INT1 จากภายนอก ประโยชน์ในการทำงานแบบนี้ คือ ใช้วัดความกว้างของพัลส์จากอินพุทภายนอก TR1 เป็นบิทควบคุมอยู่ใน TCON ดังในรูปที่ 3.7



รูปที่ 3.6 TIMER/COUNTER 1 MODE 0 : 13 BIT COUNTER



รูปที่ 3.7 TCON : TIMER/COUNTER CONTROL REGISTOR

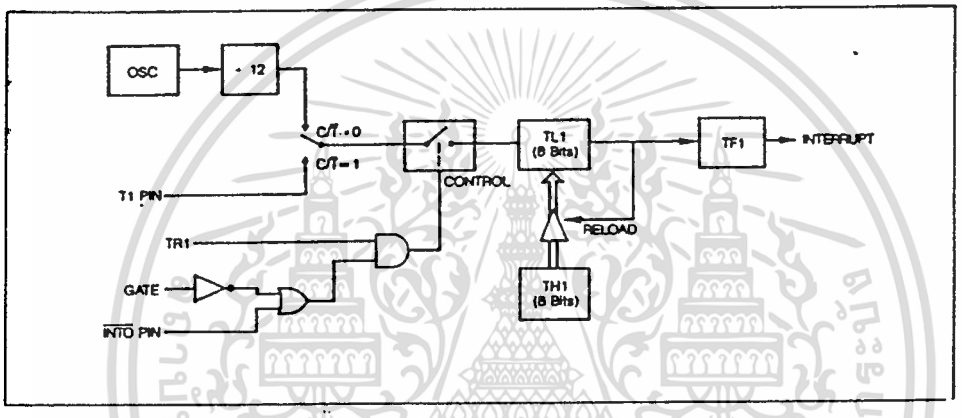
ในโหมด 0 ที่จะแบ่ง TH1 เป็น 8 บิท กับ TL1 อีก 5 บิทโดยที่เหลืออีก 3 บิท นั้นไม่ได้ใช้ แลการใช้งานจะเหมือนกันทั้ง TIMER1 และ TIMER0

โหมด 1

การใช้งานเหมือนกับโหมด 0 ยกเว้น รีจิสเตอร์ที่ใช้จะเป็นแบบ 16 บิท

โหมด 2

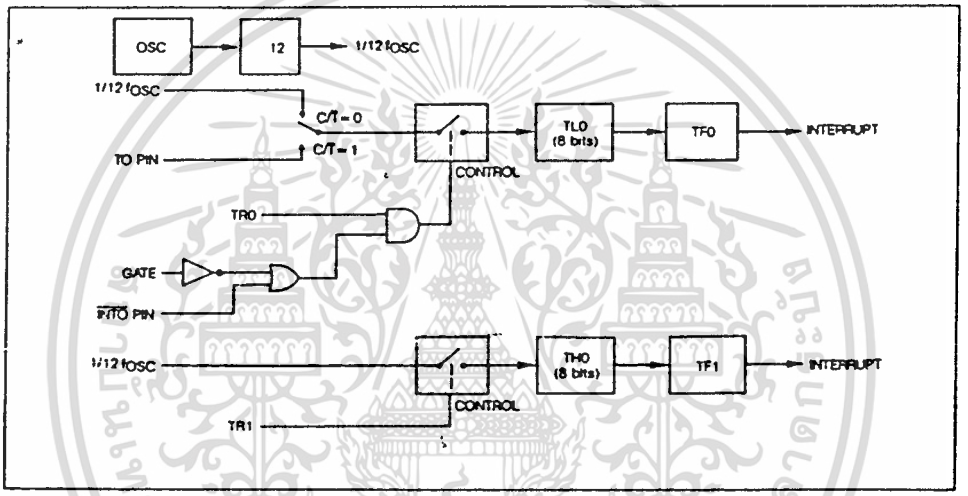
ในโหมด 2 รีจิสเตอร์จะเป็นแบบ 8 บิท โดยที่ TL1 จะสามารถโหลดข้อมูลจาก TH1 ได้ใหม่ (AUTO-RELOAD) เมื่อเกิดโอเวอร์โฟลวจาก TL1 (ดูรูปที่ 3.8) โดยที่ค่าใน TH1 จะไม่ถูกเปลี่ยการทำงานอื่นๆ จะเหมือนกับโหมด 0



รูปที่ 3.8 TIMER/COUNTER 1 MODE 2 : 8 BIT AUTO-RELOAD

โหมด 3

ในโหมดฮ 3 นี้จะแยก TLO และ THO ของ TIMER 0 ใช้โดยอิสระ (ดังรูปที่ 3.9) TLO จะใช้บิทควบคุมคือ GATE, TRO, INTO และ THO ถูกใช้เป็น TIMER (นับแมชีนไซเคิล และรับช่วงการใช้ TR1 และ TF1 ของ TIMER1 ฉะนั้นในโหมด 3 นี้ THO จะควบคุมการอินเตอร์รัพท์ของ TIMER1 (TF1) เมื่อใช้ TIMER 0 ในโหมด 3 แล้ว TIMER 1 สามารถจะสลับใช้ระหว่างโหมด 3 และโหมดอื่นได้หรือใช้เป็น BAUD RATE GENERATER



รูปที่ 3.9 TUNER/COUNTER 0 MODE 3 : TWO 8 BIT COUNTERS

<u>พอร์ตอนุกรม</u>

พอร์ตอนุกรมนี้เป็น FULL DUPLEX ที่สมารถรับข้อมูลใน BYTE ที่สองได้ โดย BYTE แรกยังไม่ถูกอ่านอกไปจาก BUFFER แต่อย่างไรก็ตามข้อมูล BYTE แรกจะต้องถูกอ่านไปก่อนที่การรับข้อมูลใน BYTE ที่สองจะเสร็จสมบูรณ์มิฉะนั้นข้อมูล BYTE แรกสูญเสียไป (ถูกทับด้วยข้อมูลที่ตามมา) ข้อมูลที่จะใช้ในการ ส่งและรับจะถูกพักไว้ ณ ที่เดียวกันคือ SBUF

การเขียนข้อมูลไปที่ SBUF จะเป็นการโหลดข้อมูลให้กับ TRANSMIT REGISTOR และการอ่าน SBUF จะเป็นการอ่านข้อมูลจาก RECEIVER REGISTOR พอร์ตอนุกรมแบ่งการทำงานออกเป็น 4 โหมด

โหมด 0 : ข้อมูลจะเข้ามาทาง RXD ส่วนข้อมูลทางออกจะออกทาง TXD ความเร็วในการส่ง (BAUD RATE) จะถูกกำหนดตายตัวเป็น 1/12 ของความถี่ออสซิลเลเตอร์ของระบบ ในโหมด 0 จะเป็นการส่งข้อมูลขนาด 8 บิท (โดย LSB ออกไปก่อน)

โหมด 1 : ส่งและรับข้อมูลขนาด 10 บิท ซึ่งประกอบด้วย START BIT (0), ข้อมูล 8 บิท (LSB ออกก่อน), STOP BIT ในขณะรับข้อมูล STOP BIT จะถูกส่งให้ RB8 ในรีจิสเตอร์หน้าที่พิเศษ SCON ความเร็วในการส่งไม่กำหนดตายตัว (ดูการหา BAUD RATE)

โหมด 2 : ส่งและรับข้อมูลขนาด 11 บิท ประกอบด้วย START BIT (0), ข้อมูล 8 บิท (LSB ก่อน), ข้อมูลบิทที่ 9 ที่สามารถโปแกรมได้ และอีก 1 STOP BIT บิทที่ 9 ของข้อมูลสามารถ SET เป็น 0 หรือ 1 ก็ได้ ประโยชน์อาจใช้เป็นตัวส่งพาริตี้บิท โดยนำค่าของแฟลก P ใน PSW มาไว้ใน TB8 และในขณะทำการรับข้อมูลบิทที่9 ของข้อมูลจะถูกโหลดเข้าไปที่ RB8 ของ SCON ความเร็วในการส่งจะถูกโปรแกรมเป็น 1/32 หรือ 1/64 ของออสซิลเลเตอร์

การทำงานทั้ง 4 โหมด ทางด้านส่งจะเริ่มการส่งขึ้นก็ต่อเมื่อ SBUF ถูกใช้เป็นปลายทางของคำสั่งต่างๆ เช่น MOV SBUF,A ในทางด้านรับการรับจะเริ่มก็ต่อเมื่อ RI=0 และ REN=1 ในโหมด 0 ส่วนโหมดอื่นๆ การรับข้อมูลจะเริ่มต้นเมื่อมี START BIT เข้ามา และ REN=1

MULTIPROCESSOR COMMUNICATIONS

ในโหมด 2 และ 3 มีเงื่อนไขพิเศษสำหรับ การติดต่อระหว่างหน่วยประมวลผลหลายตัว ในโหมดนี้ข้อมูลในบิทที่ 9 จะถูกโหลดเข้าไปใน RB8 ต่อจากนั้นจึงจะรับ STOP BIT อินเตอร์รัพท์แฟลกของพอร์ตอนุกรมแอตทีฟก็ต่อเมื่อ RB8=1 การจะใช้ลักษณะพิเศษนี้ต้องเซ็ท SM2 ใน SCON

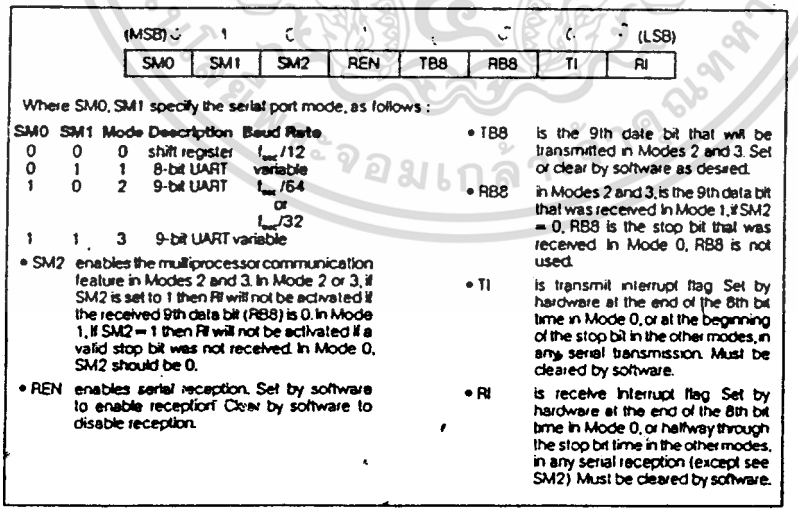
เมื่อตัวประมวลผลตัวแม่ต้องการจะส่งข้อมูลเป็นบล็อกให้กับตัวลูก ซึ่งมีอยู่หลายตัว ฉะนั้น BYTE แรกที่จะต้องส่งคือ ตัวกำหนดว่าจะให้ตัวใดเป็นตัวรับ (การกำหนด ADDRESS) โดยที่ BYTE ที่เป็น ADDRESS จะต้องแตกต่างจาก BYTE ข้อมูลซึ่งเราทำได้โดยให้ บิทที่ 9 ของ

ข้อมูลเป็น 1 ใน BYTE ที่กำหนดให้เป็น ADDRES ของตัวลูก (SLAVE) ส่วนในตัวลูกทุกตัวต้องเซ็ท SM2=1 เมื่อทำดังนี้หน่วยประมวลผลที่เป็นตัวลูกทุกตัวจะไม่ถูก อินเตอร์รัพท์ที่เกิดจากการส่งข้อมูลอนุกรมถ้าข้อมูลที่ส่งนั้นบิทที่ 9 ไม่ถูกเซ็ท ฉะนั้นหน่วยประมวลผลตัวลูกทุกตัวระรับการส่งที่กำหนดให้เป็น ADDRESS แต่จะมีเพียงตัวเดียวที่ตรวจสอบแล้วว่าถูกกำหนดให้เป็นปลายทางในการส่งข้อมูลจึงเตรียมการรับบล็อกของข้อมูลและเคลียร์ SM2 ด้วย ส่วนตัวอื่นๆ คงปล่อยให้ SM2 เซ็ทอยู่อย่างเดิมและทำงานตามปกติต่อไป

SM2 นี้จะไม่มีผลในการใช้งานในโหมด 0 ส่วนในโหมด 1 จะใช้สำหรับตรวจสอบความถูกต้องของ STOP BIT ในโหมด 1 เมื่อเซ็ท SM2=1 แล้ว RI(RECIEVE INTERRUPT) จะไม่ถูก SET ถ้าไม่ได้รับ STOP BIT ที่ถูกต้อง

SERIAL PORT CONTROL REGISTOR

ในรูปที่ 3.8 เป็นรายละเอียดของ SFR ทำหน้าที่ควบคุมพอร์ตอนุกรม (SCON) ภายใน SCON ไม่เพียงแต่มีการเลือกโหมดเท่านั้นแต่ยังรวมทั้งบิทที่ 9 ของข้อมูล (TB8, RB8) และ SERIAL PORT INTERUPT (TI และ RI)



(MSB) (LSB)

SM0	SM1	SM2	REN	TB8	RB8	TI	RI

Where SM0, SM1 specify the serial port mode, as follows :

SM0	SM1	Mode	Description	Baud Rate
0	0	0	shift register	f_{osc}/12
0	1	1	8-bit UART	variable
1	0	2	9-bit UART	f_{osc}/64 or f_{osc}/32
1	1	3	9-bit UART	variable

- SM2 enables the multiprocessor communication feature in Modes 2 and 3. In Mode 2 or 3, if SM2 is set to 1 then RI will not be activated if the received 9th data bit (RB8) is 0. In Mode 1, if SM2 = 1 then RI will not be activated if a valid stop bit was not received. In Mode 0, SM2 should be 0.
- REN enables serial reception. Set by software to enable reception. Clear by software to disable reception.
- TB8 is the 9th data bit that will be transmitted in Modes 2 and 3. Set or clear by software as desired.
- RB8 in Modes 2 and 3, is the 9th data bit that was received. In Mode 1, if SM2 = 0, RB8 is the stop bit that was received. In Mode 0, RB8 is not used.
- TI is transmit interrupt flag. Set by hardware at the end of the 8th bit time in Mode 0, or at the beginning of the stop bit in the other modes, in any serial transmission. Must be cleared by software.
- RI is receive interrupt flag. Set by hardware at the end of the 8th bit time in Mode 0, or halfway through the stop bit time in the other modes, in any serial reception (except see SM2). Must be cleared by software.

รูปที่ 3.8 SCON : SERIAL PORT CONTROL REGISTOR

อัตราความเร็วในการส่ง (BAUD RATES)

ในโหมด 0 ความเร็วในการส่งกำหนดไว้แน่นอนคือ

$$\text{MODE 0 BAUD RATE} = \frac{\text{OSCILLATOR FREQUENCY}}{12}$$

ในโหมด 2ความเร็วในการส่งขึ้นอยู่กับ SMOD ซึ่งอยู่ใน PCON ถ้า SMOD = 0 ความเร็วจะเท่ากับ 1/64 ของความถี่ออสซิลเลเตอร์ ถ้า SMOD = 1 ความเร็วจะเป็น 1/32 ของความถี่ออสซิลเลเตอร์

ความเร็วในการส่งในโหมด 2 คำนวณได้จากสูตรดังนี้

$$\text{MODE2 BAUD RATE} = \frac{2^{\text{SMOD}} \times (\text{OSCILLATOR FREQUENCY})}{64}$$

โหมด 1 และโหมด 3 ความเร็วในการส่งถูกกำหนดโดยอัตราของ OVERFLOW ของ TIMER 1

การใช้ TIMER1 ในการกำเนิด BAUD RATE

เมื่อใช้ TIMER1 เป็นตัวกำเนิด BAUD RATE ความเร็วจะขึ้นอยู่กับ OVERFLOW RATE และค่าที่อยู่ใน SMOD ความเร็วคำนวณได้จากสูตร

$$\text{MODE 1,3 BAUD RATE} = \frac{2^{\text{SMOD}} \times (\text{TIMER1 OVERFOLW RATE})}{32}$$

ในการใช้ TIMER1 เป็นตัวกำเนิดความเร็วในการส่งข้อมูลนี้ จะต้องไม่ยอมให้มีการอินเตอร์รัพท์ของ TIMER1 วิธีการใช้ TIMER1 เป็นตัวกำเนิดความเร็วนี้ โดยทั่วไปเราจะเซ็ทให้ TIMER1/COUNTER เป็น TIMER และอยู่ในโหมด 2 ซึ่ง TIMER ในโหมด 2 นี้ทำ AUTO-RELOAD ได้ (เซ็ทไบท์สูงของ TMOD=0010B) ในกรณีนี้ความเร็วจะคำนวณได้จากสูตร

$$\text{MODE 1,3 BAUD RATE} = \frac{2^{\text{SMOD}} \times \text{OSCILLATOR FREQUENCY}}{32 \quad 12 \times [256-(\text{th1})]}$$

โดยที่ค่า TH1 จะเป็นค่าในช่อง RELOAD ของตารางที่ 2

BAUD RATE	fOSC	SMOD	TIMER 1		
			C/T	MODE	RELOAD VALUE
MODE 0 MAX : 1MHZ	12MHz	X	X	X	X
MODE 2 MAX : 375K	12MHz	1	X	X	X
MODE 1,3 : 62.5K	12MHz	1	0	2	FFH
19.2K	11.059MHz	1	0	2	FDH
9.6K	11.059MHz	0	0	2	FDH
4.8K	11.059MHz	0	0	2	FAH
2.4K	11.059MHz	0	0	2	F4H
1.2K	11.059MHz	0	0	2	E8H
137.5	11.059MHz	0	0	2	1DH
110	6MHz	0	0	2	72H
110	12MHz	0	0	1	FEEBH

ตารางที่ 2 TIMER1 GENRATED COMMONLY USED BAUD RATES

รายละเอียดเพิ่มเติมเกี่ยวกับ MODE

ข้อมูลอนุกรมจะผ่านเข้าออกทางขา RXD ส่วนขา TXD นั้นจะกำเนิดสัญญาณนาฬิกา ข้อมูลที่ใช้ในการส่งและรับมีขนาด 8 บิท และ อัตราการส่งกำหนดตายตัวที่ 1/12 ของความถี่ออสซิลเลเตอร์

รูปที่ 3.9.1 แสดงแผนภาพเวลา (TIMING DIAGRAM) และการทำงานอย่างง่ายของพอร์ตอนุกรมในโหมด 0 การส่งจะเริ่มเมื่อมีการเขียนข้อมูลมาที่รีจิสเตอร์ SBUF สัญญาณ "WRITE TO SBUF" ที่ S6P2 จะให้ค่า1 แก่บิทที่9 ของรีจิสเตอร์เลื่อนข้อมูลส่ง (TRANSMIT SHIFT REGISTER) และบอกส่วนควบคุมทางการส่ง ให้เริ่มต้นการส่ง ระยะเวลาตั้งแต่เกิดสัญญาณ "WRITE TO SBUF" จนกระทั่งสัญญาณ SEND เริ่มทำงานจะใช้เวลาทั้งสิ้น 1 แมชีน-ไซเคิล

สัญญาณ SEND ทำให้ข้อมูล ของรีจิสเตอร์เลื่อนข้อมูล (SHIFT REGISTER) ถูกส่งมายังขา "ALTERNATE OUTPUT FUNCTION" ของ P3.0 (ดูรูปโครงสร้างพอร์ต 3) และยังทำให้ SHIFT CLOCK มายังขา "ALTERNATE OUTPUT FUNCTION" ของ P3.1(TXD) ซึ่งจะเป็น LOW ในระหว่างเวลาของ S3,S4 และ S5 ของทุกๆ แมชีนไซเคิล และเป็น HI ระหว่าง S6,S1 และ S2 ที่เวลา S6P2 ของทุกๆ แมชีนไซเคิลที่สัญญาณ SEND แอคทีฟข้อมูลใน SHIFT REGISTER จะถูกเลื่อนไปทางขวา 1 บิท

ในขณะข้อมูลถูกเลื่อนไปทางขวา 1 บิท ข้อมูล 0 จะถูกเลื่อนเข้ามา ทางซ้าย 1 บิท และเมื่อข้อมูลบิทสำคัญสูงสุด (MSB) ของ SHIFT REGISTER มาถึงยังตำแหน่ง OUTPUT ของ SHIFT REGISTER แล้ว สัญญาณ 1 ที่ถูกโหลดมายังบิทที่ 9 ของ SHIFT REGISTER ในตอนแรก จะเข้ามาทางซ้ายของบิท MSB และข้อมูลทางซ้ายทั้งหมดของบิทที่เข้ามาต่อท้ายบิท MSB จะเป็น 0 ทั้งหมด สภาวะเช่นนี้จำทำให้หน่วยควบคุมการส่ง (TX CONTROL BLOCK)จะทำการเลื่อนข้อมูลอีก 1 ครั้งเป็นครั้งสุดท้ายก่อนที่จะยกเลิก สัญญาณ SEND และ เซ็ทบิท TI การกระทำทั้ง 2 อย่างนี้ จะเกิดขึ้นที่ S1P1 ของแมชีนไซเคิลที่ 10 หลังจากเกิดสัญญาณ "WRITE TO SUBF"

ทางด้านรับจะเริ่มต้นเมื่อ REN=1 และ RI=0 ที่ S6P2 ของแมชีนไซเคิลต่อไป หน่วยควบยคุมการรับ (RX CONTROL UNIT) เขียนข้อมูล 1111110 ไปที่รีจิสเตอร์เลื่อนข้อมูลทางด้านรับ (RECIVE SHIFT REGISTER) และสัญญาณนาฬิกาเฟสถัดไปจะทำให้ขา RECIVE ทำงาน(ดูรูป 3.9.1) ข้อมูลที่อยู่ในตัวเลื่อนข้อมูลด้านรับจะถูกเลื่อนมาทางด้าน ซ้าย 1 ตำแหน่ง

ข้อมูลที่เข้ามาทางขวาคือข้อมูลบิทที่ถูกสุ่มตัวอย่างทางขา P3.0 ที่เวลา S5P2 ของแมชีนไซเคิลเดียวกัน (S5P2)

ในขณะที่ข้อมูลเข้ามาทางขวา บิท 1 จะถูกดันเข้ามาทางซ้าย และเมื่อบิทที่เป็น 0 ซึ่งถูกโหลดให้ในตอนเริ่มต้น (1111110) มาถึงทางด้านซ้ายมือสุดของตัวเลื่อนข้อมูล เป็นเหตุให้ RX CONTROL BLOCK จะทำการเลื่อนข้อมูลอีกครั้งซึ่งเป็นครั้งสุดท้าย แล้วทำการโหลดค่าข้อมูลที่รับเข้ามาให้กับ SBUF สัญญาณ RECEIVE จะถูกเคลียร์ RI จะถูกเซ็ทที่ S1P1 ของแมชีน-ไซเคิลที่ 10 หลังจากบิท RI ถูกเคลียร์ในตอนเริ่มต้น

<u>รายละเอียดเพิ่มเติมเกี่ยวกับ MODE1 (ดูรูป 3.10)</u>

โหมด 1 จะเป็นการส่งรับข้อมูลขนาด 10 บิท โดยมี 1 START BIT (0),8 บิทข้อมูลและ 1 STOP บิท (1) การส่งจะเริ่มต้นเมื่อมีการใช้คำสั่งที่ต้องใช้ SBUF เป็นปลายทาง เช่น MOV SBUF, A ในขณะที่เขียนข้อมูลไปที่ SBUF บิทที่ 9 ของ TX SHIFT REGISTOR จะถูกโหลดด้วย 1 ในทางส่งจะมีการทำงานเหมือนกับโหมด 0 ดังที่ได้อธิบายมาข้างต้น ส่วนทางด้านรับจะเริ่มตั้งแต่ตรวจพบขอบขาลง (1 TO 0) ของ RXD เมื่อตรวจพบขอบขาลงของสัญญาณทางขา RXD ข้อมูล 1FFH จะถูกโหลดให้ INPUT SHIFT REGISTOR ข้อมูลจะถูกสุ่มตัวอย่าง 3 ครั้ง และข้อมูลที่ได้คือ ข้อมูลจากตัวอย่าง 2 ใน 3 ที่สุ่มมาได้ เหตุที่ทำอย่างนี้เพื่อป้องกันสัญญาณรบกวนที่อาจเกิดขึ้นได้ ในระหว่างบิทแรก (START BIT) ถ้าตัวอย่างที่สุ่มมาได้ไม่ใช่ 0 วงจรส่วนรับข้อมูล จะถูกรีเซ็ทและจะกลับไปรอรับการเกิด START BIT อีกครั้ง ในกรณีที่ START BIT เกิดขึ้นถูกต้องข้อมูลจะถูก SHIFT เข้าไปใน INPUT SHIFT REGISTER และจะรับข้อมูลส่วนที่เหลือต่อไป ในขณะที่บิทของข้อมูลถูกเลื่อนเข้ามาทางขวา "1" จะถูกเลื่อนออกทางซ้าย เมื่อ START BIT มาถึงตำแหน่งซ้ายสุดของ SHIFT REGISTOR จะทำให้ RX CONTROL BLOCK ทำการเลื่อนอีกครั้งพร้อมทั้งโหลด SBUF และ RB8 และเซ็ทแฟลก RI สัญญาณที่โหลด SBUF,RB8 และเซ็ท RI จะเกิดขึ้นได้ก็ต่อเมื่อสภาวะเกิดขึ้นเมื่อในช่วงเวลาการเลื่อนข้อมูลครั้งสุดท้ายจากสภาวะต่อไปนี้

1. RI = 0
2. SM2 = 0 หรือ THE RECEIVED STOP BIT = 1

ถ้าไม่พบสภาวะทั้งสองอย่างนี้การรับข้อมูลนั้นถือว่าล้มเหลว ถ้าสภาวะทั้งสองเกิดขึ้นถูกต้อง STOP BIT จะเข้าไปที่ RB8 ข้อมูล 8 บิท จะเข้าไปที่ SBUF และ RI จะถูกกระตุ้น เมื่อถึงขั้นตอนนี้ ไม่ว่าสภาวะทั้งสองจะเกิดขึ้นหรือไม่ระบบจะกลับเข้าสู่การคอย START BIT ต่อไป

รายละเอียดเกี่ยวกับโหมด 2 และ 3

เป็นการรับส่งข้อมูล 11 บิท ผ่านทาง RXD และ TXD โดยมีรูปแบบการส่งดังนี้ START BIT(0),8 DATA BIT(LSB ออกก่อน) บิทที่ 9 ซึ่งโปรแกรมได้ และ STOP BIT (1) ในการส่งข้อมูลบิทที่ 9 (TD8 ใน SCON) สามารถที่จะกำหนดให้เป็น 0 หรือ 1 ได้ ส่วนทางด้านรับข้อมูลบิทที่ 9 จะเข้าไปที่ RB8 (ในรีจิสเตอร์ SCON)ในโหมด 2 ความเร็วในการส่งสามารถโปรแกรมให้เป็น 1/32 หรือ 1/64 ของความถี่ออสซิลเลเตอร์ได้ด้วยบิท SMOD ในรีจิสเตอร์ PCON ส่วนโหมด 3 ความเร็วในการส่งถูกกำหนด โดย TIMER0 หรือ TIMER1 ขึ้นอยู่กับ TCLK และ RCLK

รูปที่ 3.10.1 และ 3.10.2 แสดงแผนภาพเวลา และการทำงานอย่างง่ายของพอร์ตอนุกรม โหมด 2 และ 3 ตามลำดับการทำงานของภาครับจะเหมือนกับโหมด 1 ทุกประการยกเว้นภาคส่งที่มีส่วนแตกต่างกับโหมด 1 คือมีส่วนของบิทที่ 9 ของ TRANSMIT SHIFT REGISTER

เมื่อสัญญาณการส่งเริ่มขึ้น สัญญาณนาฬิกาที่ใช้ในการเลื่อนข้อมูล จะให้ข้อมูลบิท 1 กับบิทที่ 9 ของ SHIFT REGISTER(เพื่อเป็น STOP BIT)หลังจากนั้นข้อมูล 0 จะถูกเลื่อนเข้ามาจากทางซ้าย (บิทข้อมูลจะเลื่อนออกทางขวา) เมื่อบิท TB8 มาถึงตำแหน่ง OUTPUT ของ SHIFT REGISTER บิทหยุด (STOP BIT)จะเข้ามาต่อท้ายบิท TB8 และขณะนี้ข้อมูลทางซ้ายจะเป็น 0 ทั้งหมด ในสภาวะนี้ TX CONTROL UNIT จะทำการเลื่อนข้อมูลอีก 1 ครั้ง เป็นครั้งสุดท้าย

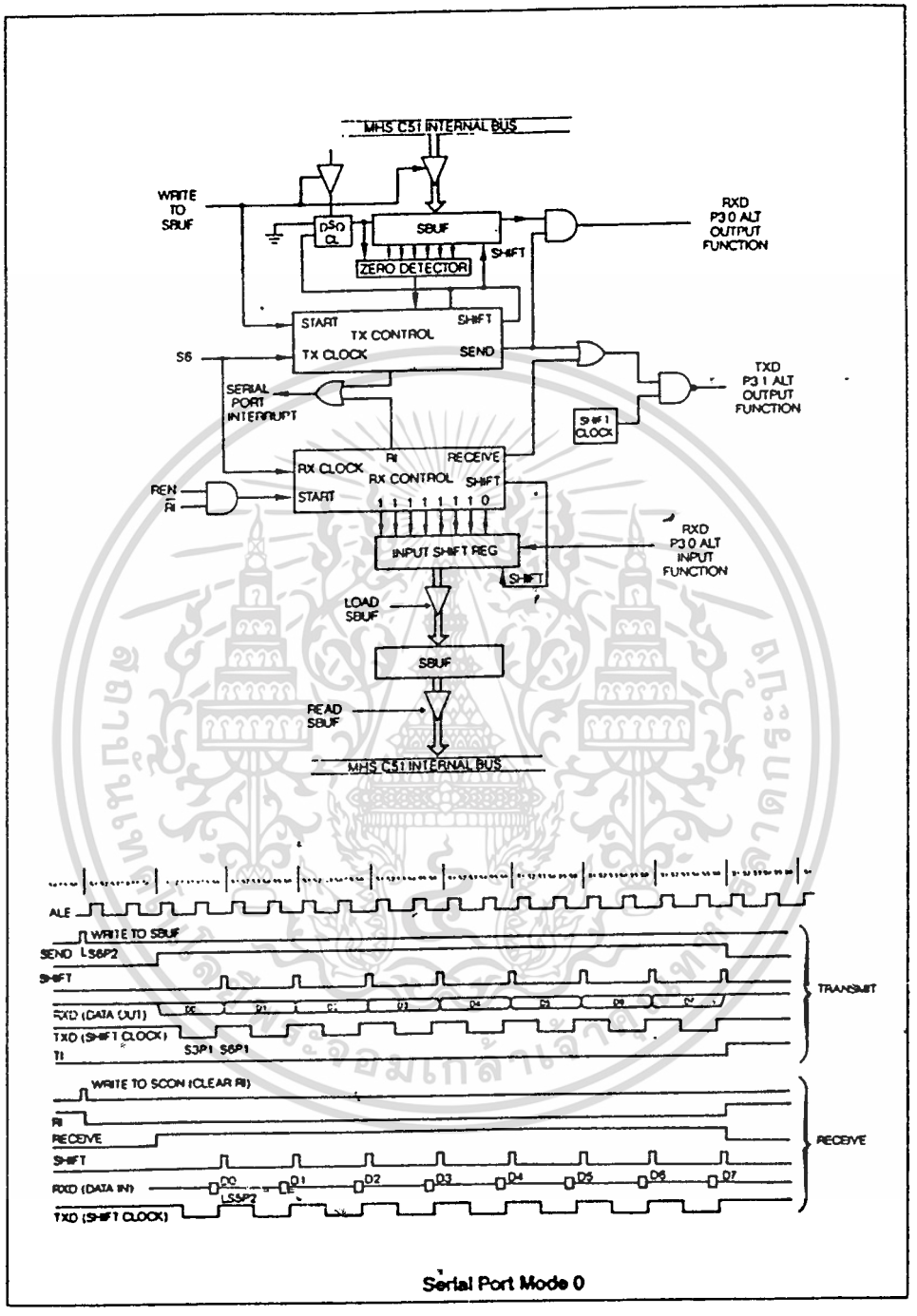
ทางด้านรับ เมื่อ START BIT ถูกเลื่อนเข้ามาถึง OUTPUT ของ SHIFT REGISTER (ในโหมด 2 และ 3) จะทำให้ ข้อมูลถูกเลื่อนอีก 1 ครั้ง ซึ่งเป็นครั้งสุดท้าย ต่อจากนั้น จะโหลดข้อมูลให้ SBUF และ RB8 ต่อจากนั้นจะทำการเซ็ทบิท RI ขบวนการเหล่านี้จะเกิด ก็ต่อเมื่อ

1. RI = 0
2. SM2 = 0 หรือ ข้อมูลบิทที่ 9=1

สรุป ข้อมูลจะเข้ามาที่ SBUF และ แฟลก RI จะเซ็ทก็ต่อเมื่อ RI สภาวะเดิม ต้องเป็น 0 อยู่ และอีกสภาวะคือ SM2 ต้องเป็น 0 หรือไม่เช่นนั้นข้อมูลบิทที่ 9 ต้องเป็น 1

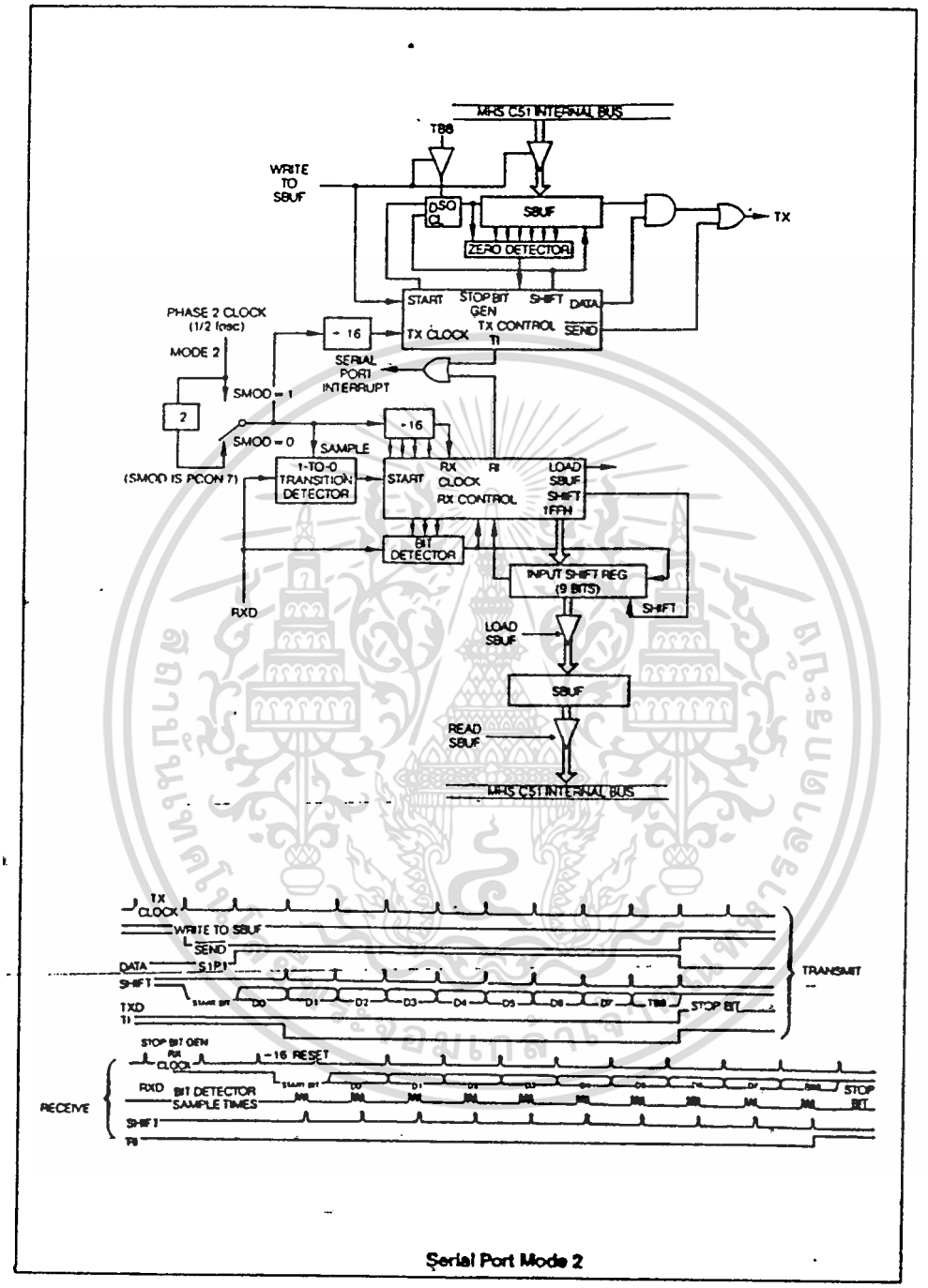
ข้อสังเกตุ คือ เมื่อ RI ให้สัญญาณอินเตอร์รัพท์แล้ว ในส่วนของโปรแกรมต้องเคลียร์ RI ให้ด้วยส่วนในโหมด 2 และ 3 ต้องรีเซ็ทข้อมูลในบิทที่ 9 ต้องเป็น 1 ยกเว้นการใช้งานแบบ MULTI PROCESSOR COMMUNI-CATION ดังที่ กล่าวมาข้างต้น

รูปที่ 3.9.1 SERIAL PORT MODE 0



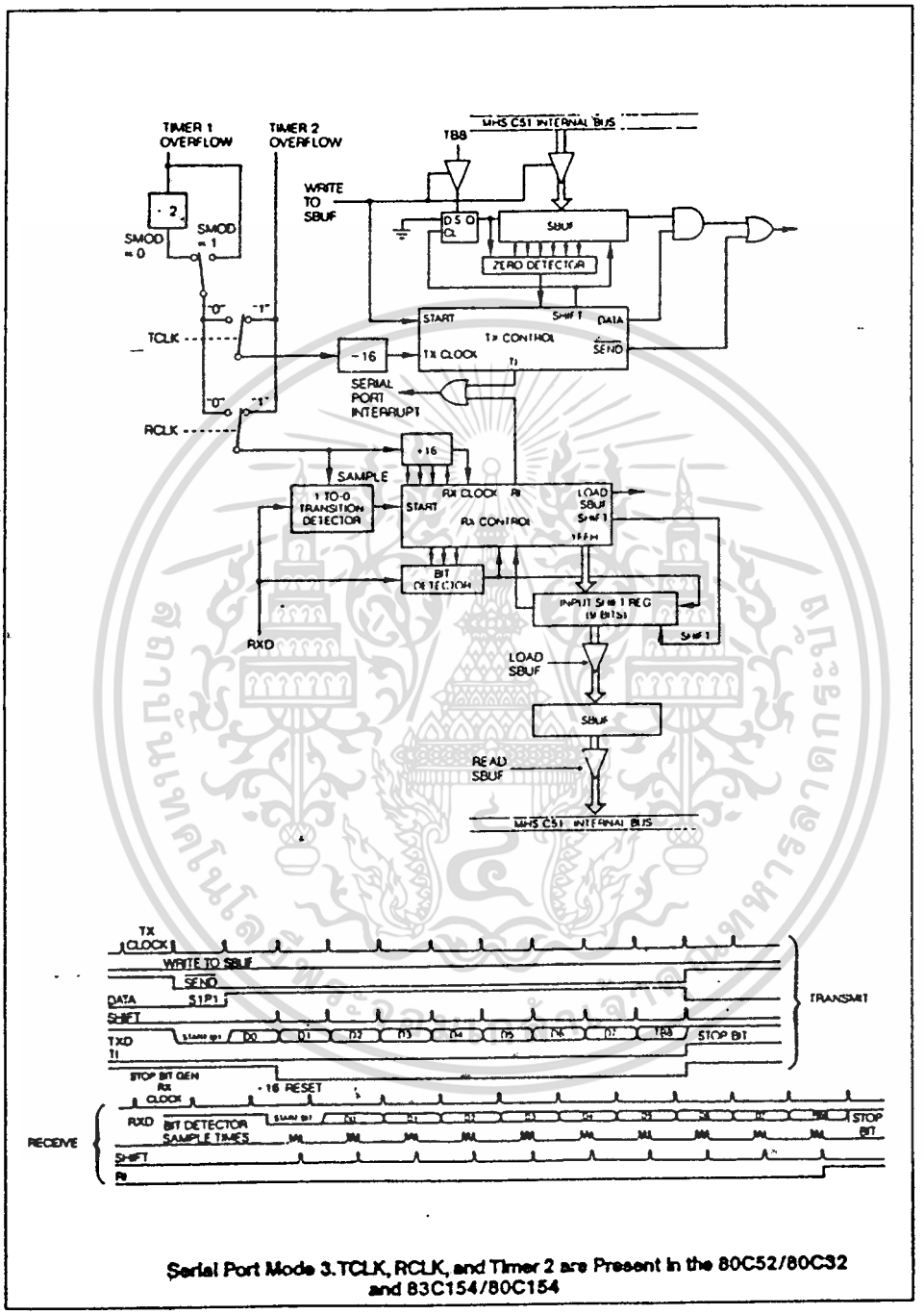
Serial Port Mode 0

รูปที่ 3.9.2 SERIAL PORT MODE 2



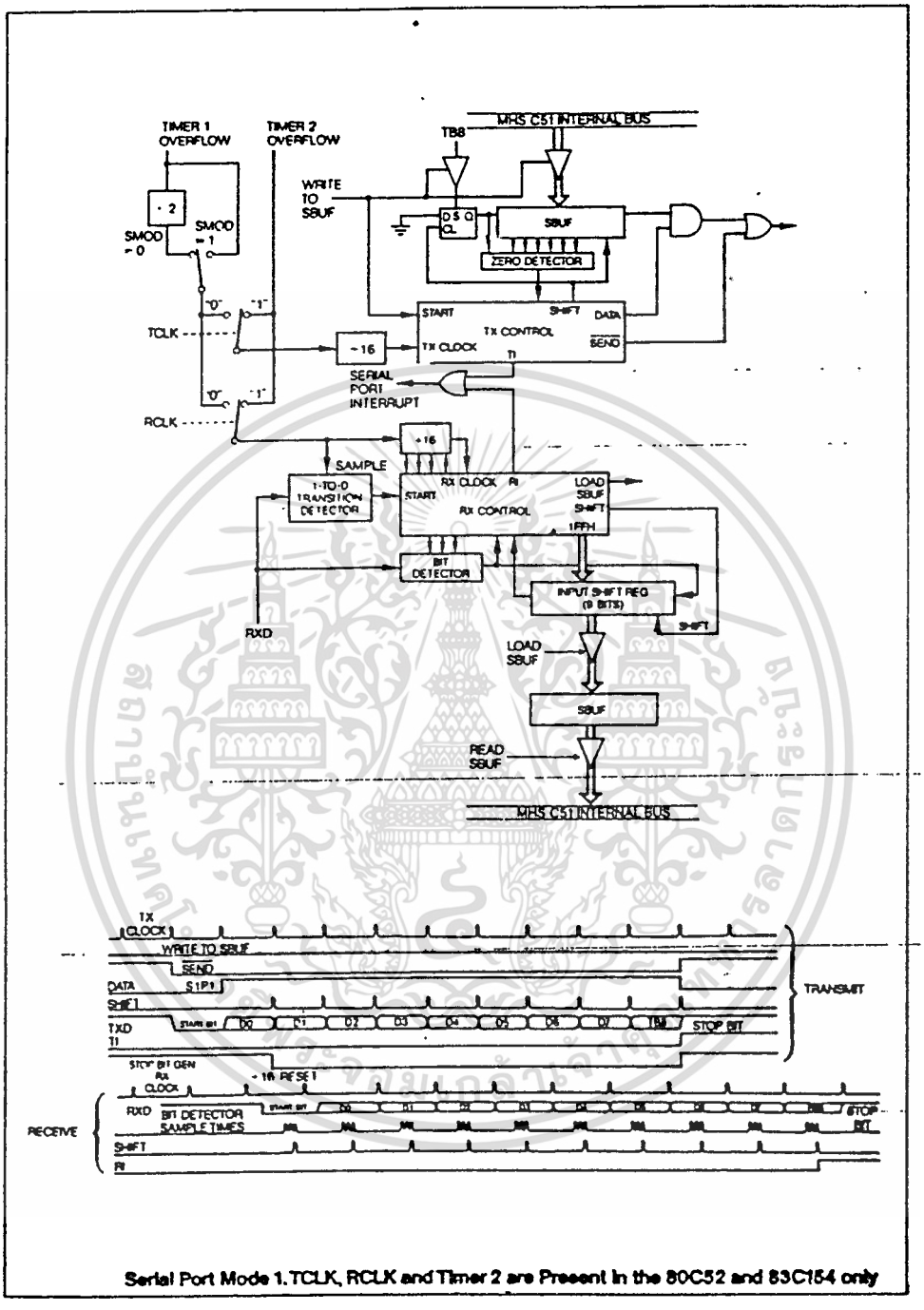
Serial Port Mode 2

รูปที่ 3.9.3 SERIAL PORT MODE 3



Serial Port Mode 3.TCLK, RCLK, and Timer 2 are Present in the 80C52/80C32 and 83C154/80C154

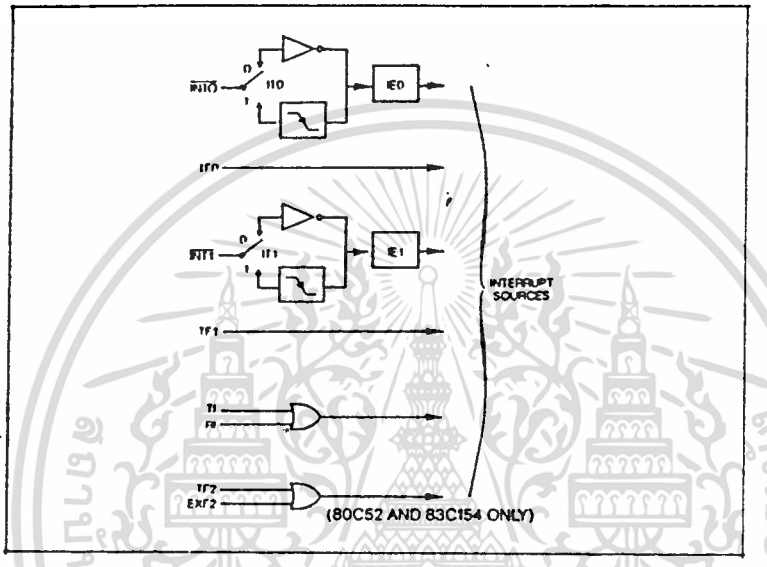
รูปที่ 3.10 SERIAL PORT MODE 1



Serial Port Mode 1. TCLK, RCLK and Timer 2 are Present in the 80C52 and 83C154 only

อินเตอร์รัพท์

8031 จะมีแหล่งกำเนิดสัญญาณอินเต่อร์รัพท์ 5 อย่าง ดังในรูปที่ 3.11



รูปที่ 3.11 INTERUPT SOURCE

ขาอินเตอร์รัพท์ภายนอกมีเพียง 2 ขา คือ INT0 และ INT1 ซึ่งสามารถโปรแกรมให้เป็นแบบกระตุ้นด้วยระดับสัญญาณ (LEVEL ACTIVATED) หรือการกระตุ้นด้วยขอบของสัญญาณ (TRANSITION ACTIVATED) ขึ้นอยู่กับบิท IT0 และ IT1 ในรีจิสเตอร์ TCON แฟลกที่กำเนิดสัญญาณอินเตอร์รัพท์ที่แท้จริงคือ IE0 และ IE1 ใน TCON

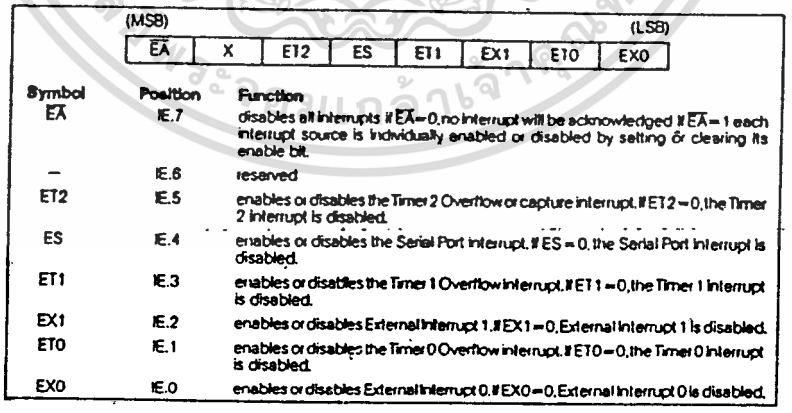
เมื่อมีสัญญาณอินเตอร์รัพท์จากภายนอกแฟลกที่กำเนิดสัญญาณอินเตอร์รัพท์จะถูกเซ็ทเคลียร์โดยฮาร์แวร์ (ภายใน 8031) ขณะที่ CPU กระโดดไปทำในอินเตอร์รัพท์รูทีน โดยที่ต้องโปรแกรมให้รับการอินเตอร์รัพท์แบบขอบของสัญญาณ ถ้าชนิดของอินเตอร์รัพท์ถูกเซ็ทให้เป็นการอินเตอร์รัพท์โดยระดับของสัญญาณ แฟลกอินเตอร์รัพท์ต้อง CLEAR ด้วยซอร์ฟแวร์ ภายในซัพรูทีนของอินเตอร์รัพท์

อินเตอร์รัพท์ของ TIMER 0 และ TIMER 1 เกิดขึ้นโดยการแฟลก TF0 และ TF1 เมื่อเกิดอินเตอร์รัพท์ขึ้นแฟลกจะถูกเคลียร์โดยฮาร์ดแวร์เมื่อกระโดดไปทำงานที่ SERVICE-ROUTINE

อินเตอร์รัพท์ของพอร์ตอนุกรม เกิดขึ้นจากทางรับ หรือทางส่งข้อมูลโดยที่โปรแกรมต้องตรวจเช็คว่าเป็นอินเตอร์รัพท์จากด้านรับ (RI) หรือทางด้านส่ง (TI) และจะต้องทำการเคลียร์แฟลก อินเตอร์รัพท์ด้วยซอร์ฟแวร์

บิทที่กำเนิดสัญญาณอินเตอร์รัพท์สามารถเซ็ทหรือเคลียร์ได้โดยซอร์ฟแวร์ โดยจะให้ผลเหมือนกับสัญญาณที่กระทำโดยฮาร์ดแวร์นั่น หมายความว่าสัญญาณอินเตอร์รัพท์สามารถจะเกิดขึ้นเหรืออินเตอร์รัพท์ที่ค้างอยู่สามารถยกเลิกได้โดยซอร์ฟแวร์

แหล่งกำเนิดของสัญญาณอินเตอร์รัพท์แต่ละตัว สามารถจะ ENABLE หรือ DISABLE โดยการเซ็ทหรือเคลียร์บิทที่อยู่ในรีจิสเตอร์ IE ข้อสังเกตุในรีจิสเตอร์ IE บิทที่ 7 คือ EA จะเป็นตัวควบคุมการ ENABLE หรือ DISABLE ของสัญญาณอินเตอร์รัพท์ทุกสัญญาณ ฉะนั้นเมื่อต้องการใช้อินเตอร์รัพท์ต้องไม่ลืมที่จะเซ็ทบิท EA ด้วย หลังจากนั้นจึงทำการ ENABLE สัญญาณอินเตอร์รัพท์ที่ต้องการ



(MSB) (LSB)

$\overline{EA}$	X	ET2	ES	ET1	EX1	ET0	EX0

Symbol	Position	Function
$\overline{EA}$	IE.7	disables all interrupts. If $\overline{EA}$ = 0, no interrupt will be acknowledged. If $\overline{EA}$ = 1 each interrupt source is individually enabled or disabled by setting or clearing its enable bit.
–	IE.6	reserved
ET2	IE.5	enables or disables the Timer 2 Overflow or capture interrupt. If ET2 = 0, the Timer 2 interrupt is disabled.
ES	IE.4	enables or disables the Serial Port interrupt. If ES = 0, the Serial Port interrupt is disabled.
ET1	IE.3	enables or disables the Timer 1 Overflow interrupt. If ET1 = 0, the Timer 1 interrupt is disabled.
EX1	IE.2	enables or disables External Interrupt 1. If EX1 = 0, External Interrupt 1 is disabled.
ET0	IE.1	enables or disables the Timer 0 Overflow interrupt. If ET0 = 0, the Timer 0 interrupt is disabled.
EX0	IE.0	enables or disables External Interrupt 0. If EX0 = 0, External Interrupt 0 is disabled.

รูปที่ 3.12 IE : INTERUPT ENABLE REGISTER

ลำดับความสำคัญของการอินเตอร์รัพท์

แหล่งสัญญาณอินเตอร์รัพท์แต่ละสัญญาณสามารถโปรแกรมได้ว่าให้เป็นลำดับความสำคัญสูงหรือลำดับความสำคัญต่ำ โดยที่ลำดับความสำคัญต่ำจะถูกอินเตอร์รัพท์ด้วยสัญญาณอินเตอร์-รัพท์ที่มีความสำคัญสูงกว่าและลำดับความสำคัญสูงจะไม่ถูกอินเตอร์รัพท์โดยสัญญาณอินเตอร์รัพท์ที่มีความสำคัญต่ำกว่า

ถ้ามีการอินเตอร์รัพท์ด้วยลำดับความสำคัญเท่ากันมากกว่า 1 สัญญาณ CPU จะทำการตรวจ (POLLING) และตัดสินใจว่าจะให้บริการกับสัญญาณอินเตอร์รัพท์ตัวใด ฉะนั้นในแต่ละลำดับความสำคัญยังมีการจัดลำดับความสำคัญไว้อีกดังรายละเอียดข้างล่าง

SOURCE	PIORITY WITHIN LEVEL
1. IE0	(HIGHEST)
2. TF0	
3. IE1	
4. TF1	
5. RI+TI	
6. TF2+EXF2	(LOWEST)

หมายเหตุ ลำดับความสำคัญนี้ใช้เฉพาะเมื่อมีสัญญาณอินเตอร์รัพท์ในลำดับความสำคัญเท่ากันมากกว่า 1 สัญญาณ

การทำงานของสัญญาณอินเตอร์รัพท์

แฟลกของสัญญาณอินเตอร์รัพท์จะถูกสุ่มตัวอย่างในเสตท 5 เฟสที่ 2 ของทุกๆ แมชีน-ไซเคิลและจะทำการตรวจ (POLLING) การอินเตอร์รัพท์จาก 5 แหล่งสัญญาณในแมชีนไซ-เคิล ต่อมาถ้าผลของการสุ่มตัวอย่างพบว่า แฟลกอินเตอร์รัพท์ถูกเซ็ทในเสตทที่ 5 เฟสที่ 2 ของแมชีนไซเคิลที่ผ่านมาแล้ว จะมีการเรียกไปยังส่วนของโปรแกรมบริการ อินเตอร์รัพท์ หากว่าไม่ถูกขัดขวางด้วยสภาวะใดสภาวะหนึ่งดังต่อไปนี้

1. กำลังทำคำสั่งอยู่ในโปรแกรมบริการอินเตอร์รัพท์ที่มีความสำคัญเท่ากันหรือสูงกว่า
2. ไม่ใช่ไซเคิลสุดท้ายของคำสั่งที่กำลังปฏิบัติอยู่
3. คำสั่งที่ปฏิบัติอยู่นั้นคือ RETI หรือ คำสั่งที่ติดต่อกับรีจิสเตอร์ IE หรือ IP

ในสภาวะตามข้อ 2 เพื่อเป็นการประกันว่าคำสั่งที่ปฏิบัติถึงไซเคิลสุดท้ายแล้ว จะไม่ถูกอินเตอร์รัพท์จนกว่าจะปฏิบัติคำสั่งนั้นจนจบเสียก่อน

ตามข้อ 3.นั้นในกรณีที่ CPU กำลังทำคำสั่ง RETI หรือกลังติดต่อกับ IE หรือ IP ตัวใดตัวหนึ่งอยู่แล้วเกิดอินเตอร์รัพท์ขึ้น CPU จะยอมให้มีการอินเตอร์รัพท์แต่ต้องปฏิบัติอย่างน้อยอีก 1 คำสั่ง หลังจากทำคำสั่ง IE,EP หรือ RETI ตัวอย่างเช่น ถูกอินเตอร์รัพท์ในขณะที่กำลังทำคำสั่ง RETI หน่วยประมวลผลจะสั่งแอดเดรสคืนให้ PC หลังจากคำสั่ง RETI และปฏิบัติอีก 1 คำสั่งในโปรแกรมหลักต่อจากนั้นจึงจะตอบสนองการอินเตอร์รัพท์ CPU รับรู้การอินเตอร์รัพท์โดยกระโดดไปทำโปรแกรมบริการอินเตอร์รัพท์ ในบางกรณีจะไม่เคลียร์แฟลกที่กำเนิดสัญญาณอินเตอร์รัพท์นั้น กรณีที่ไม่เคลียร์แฟลกส่วนของโปรแกรมของผู้ใช้จะต้องมีคำสั่งเคลียร์แฟลกเอง เช่น การอินเตอร์รัพท์ที่เกิดจากพอร์ตอนุกรม ส่วนสัญญาณอินเตอร์รัพท์จากภายนอก แฟลกจะถูกเคลียร์ให้ถ้าเป็นการโปรแกรมให้รับการอินเตอร์รัพท์แบบการเปลี่ยนแปลงขอบของสัญญาณ การตอบสนองสัญญาณอินเตอร์รัพท์ CPU จะกระโดดไปที่ตำแหน่งของโปรแกรมบริการอินเตอร์รัพท์ตามชนิดของอินเตอร์รัพท์ ดังนี้

SOURCE	VECTOR ADDRESS
IE0	0003H
TF0	000BH
IE1	0013H
TF1	001BH
R1+T1	0023H
TF2+EXF2	002BH (8032/52)

การอินเตอร์รัพท์จากภายนอก

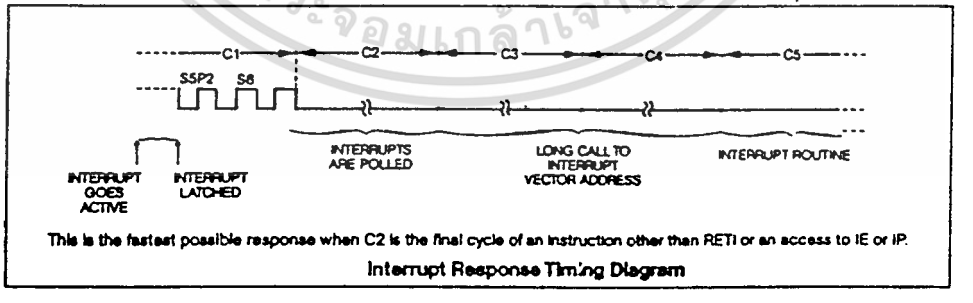
การอินเตอร์รัพท์จากภายนอกกระทำได้ 2 อินพุทคือ INT0 และ INT1 โดยสามารถโปรแกรมให้เป็นแบบการเปลี่ยนแปลงขอบสัญญาณหรือเป็นแบบระดับสัญญาณก็ได้ โดยเซ็ทหรือเคลียร์บิท IT1 หรือ IT0 ในรีจิสเตอร์ TCON ถ้า ITx = 0 จะเป็นการอินเตอร์รัพท์แบบระดับสัญญาณ (LOW) ถ้า ITx = 1 จะเป็นการรับอินเตอร์รัพท์แบบการเปลี่ยนแปลงขอบของสัญญาณและในกรณีนี้ที่ขา INTx จะต้องได้รับสัญญาณ HIGH 1 ไซเคิล และ LOW ในไซเคิลต่อไปส่วนของอินเตอร์รัพท์จะเซ็ท IEx ใน TCON ฉะนั้นสัญญาณที่เป็น HIGH และ LOW ที่กล่าวมาจะต้องมีค่าอย่างน้อย 12 คาบเวลาของความถี่ออสซิลเลเตอร์ และ IEx จะถูกเคลียร์เมื่อโปรแกรมกระโดดไปทำงานในส่วนของเซอร์วิสรูทีนโดยอัตโนมัติ ถ้าสัญญาณอินเตอร์รัพท์จากภายนอกเป็นการอินเตอร์รัพท์แบบระดับสัญญาณ (LOW) วงจรอินเตอร์รัพท์ภายนอกต้องรักษาระดับสัญญาณ 0 จนกว่าส่วนบริการอินเตอร์รัพท์จะทำงานและต้องถอนตัวจากการอินเตอร์รัพท์ก่อนที่ CPU จะเสร็จสิ้นโปรแกรมบริการอินเตอร์รัพท์

เวลาในการตอบสนองการอินเตอร์รัพท์

ถ้าการอินเตอร์รัพท์จากภายนอกเกิดขึ้นในภาวะปกติ CPU จะใช้เวลาตั้งแต่เซ็ทอินเตอร์รัพท์แฟลก ตรวจสอบ (POLLING) ไปจนถึงกระโดดไปทำคำสั่งของซับรูทีนอย่างน้อย

ที่สุด 3 แมชีนไซเคิล ในบางสภาวะจะใช้เวลามากกว่า 3 แมชีนไซเคิลถ้าผลของอินเตอร์รัพท์ถูกขัดขวางด้วยสภาวะใดสภาวะหนึ่งใน 3 ข้อ จากที่กล่าวมาแล้วข้างต้น เช่น ถ้าผลของการอินเตอร์รัพท์ในขณะที่ CPU ทำคำสั่งอยู่ซับรูทีนของอินเตอร์รัพท์ที่ลำดับความสำคัญเท่ากัน หรือสูงกว่าเวลาจะมากหรือน้อยขึ้นอยู่กับโปรแกรมในซับรูทีนของอินเตอร์รัพท์ที่กำลังทำอยู่ ถ้าคำสั่งที่กำลังดำเนินการนั้นไม่ใช่แมชีนไซเคิลสุดท้าย เวลาในการตอบสนองการอินเตอร์รัพท์จะมากขึ้นแต่จะไม่เกิน 3 แมชีนไซเคิล เพราะว่าคำสั่งที่ยาวที่สุด (คูณและหาร) จะยาวเพียง 4 แมชีนไซเคิล และถ้าคำสั่งที่กำลังดำเนินการอยู่เป็นคำสั่ง RETI หรือคำสั่งติดต่อกับรีจิสเตอร์ IE หรือ IP เวลาในการตอบสนองการอินเตอร์รัพท์จะเพิ่มขึ้นแต่ไม่มากกว่า 5 แมชีนไซเคิล (จะต้องทำอย่างมากที่สุดอีก 1 แมชีนไซเคิลสำหรับทำให้คำสั่ง (RETI) จบสมบูรณ์และกับอีก 4 แมชีนไซเคิลสำหรับคำสั่งที่ยาวที่สุด อีก 1 คำสั่ง)

<u>สรุป</u> สัญญาณอินเตอร์รัพท์เดี่ยว (โดยไม่ซ้อนกับอินเตอร์รัพท์อื่น) จะใช้เวลาในการตอบสนองมากกว่า 3 แมชีนไซเคิล และน้อยกว่า 8 แมชีนไซเคิล



รูปที่ 3.13 INTERUPT RESPONSE TIMING DIAGRAM

การรีเซ็ท

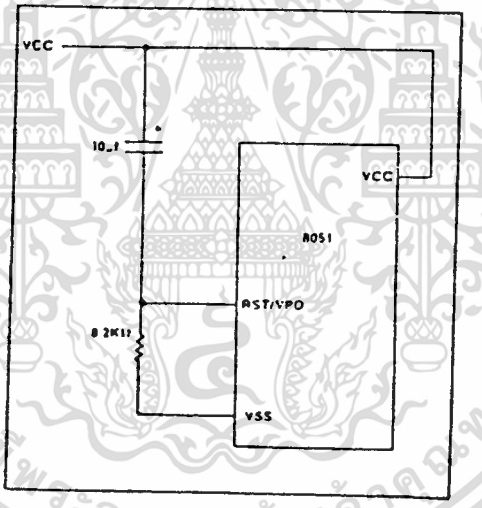
สัญญาณรีเซ็ทเป็นสัญญาณอินพุททางขา 9 การรีเซ็ทจะสมบูรณ์จะต้องรักษาระดับ HIGH อย่างน้อยที่สุด 2 แมชีนไซเคิล (24 คาบเวลาของออสซิลเลเตอร์) การรีเซ็ทภายในตัว CPU จะเริ่มในระหว่างไซเคิลที่ 2 นับตั้งแต่ขา RST เป็น HIGH ผลของการรีเซ็ทจะมีผลกับรีจิสเตอร์ ดังต่อไปนี้

REGISTOR	CONTENT
PC	0000H
ACC	00H
B	00H
PSW	00H
SP	07H
DPTR	0000H
P0-P3	FFH
IP	(XX000000)
IE	(0X000000)
TMOD	00H
TCON	00H
TH0	00H
TL0	00H
TH1	00H
TL1	00H
SCON	00H
SBUF	00H
PCON	00H

RAM ภายในจะไม่ถูกเคลียร์เมื่อ CPU ถูกรีเซ็ท

การรีเซ็ทเมื่อเปิดเครื่อง

เมื่อจ่ายไฟเข้าระบบควรจะมีการรีเซ็ท CPU ก่อนเพื่อรอให้ทั้งระบบอยู่ในสภาวะพร้อมที่จะทำงาน ซึ่งทำได้โดยต่อ C ขนาด 10uF จาก VCC มาที่ขา 9 ต่อ R ขนาด 8.2K ลงกราวด์



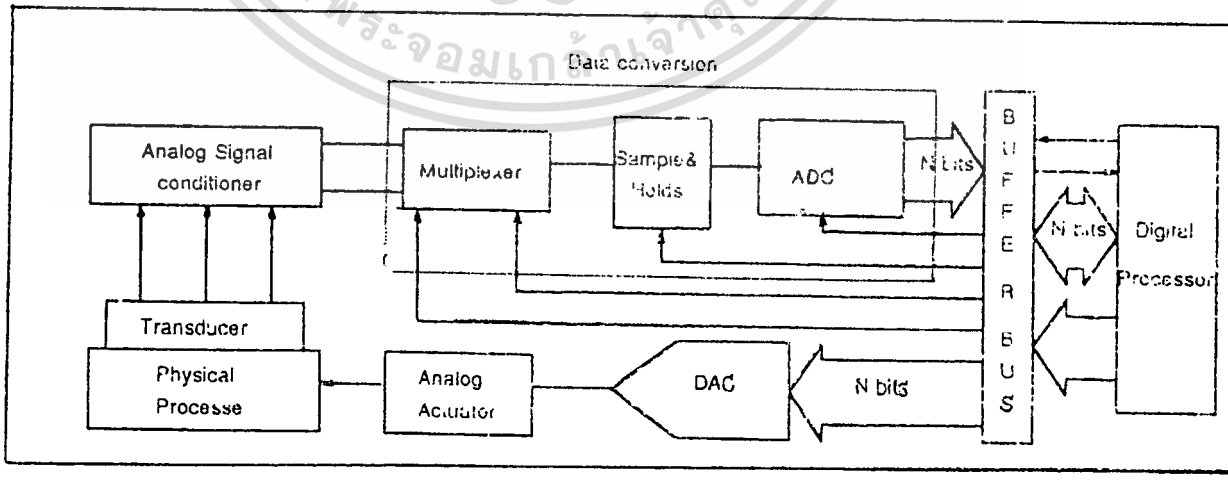
รูปที่ 3.14 POWER ON RESET CIRCUIT

บทที่ 4

ทฤษฎีการแปลงสัญญาณอนาลอกเป็นดิจิตอล

ทฤษฎีของ Data Acquisltion and Conversion

รูปแบบสัญญาณไฟฟ้าที่เราพบเห็นและคุ้นเคยในชีวิตประจำวันจะอยู่ในรูปของสัญญาณที่ต่อเนื่องเรียกว่าสัญญาณอนาลอก (Analog Signal)ซึ่งแต่เดิมการนำเอาสัญญาณไฟฟ้าดังกล่าวมาประมวล(Process) เพื่อให้มีรูปแบบที่เหมาะสมจะกระทำในแบบอนาลอกนั่นเอง แต่เมื่อเทคนิคและอุปกรณ์การประมวลสัญญาณทางดิจิตอลได้รับการพัฒนาขึ้นมา เนื่องจากพบว่าในรูปแบบของดิจิตอล การประมวล เก็บ สื่อสาร และการนำเสนอกระทำได้ง่ายและอย่างมีประสิทธิภาพมากกว่า ดังนั้นการเปลี่ยนรูปสัญญาณ (conversion) จึงได้มีความจำเป็นขึ้นมา ในรูปที่ 4.1 เป็นตัวอย่างแสดงระบบควบคุมที่ใช้การประมวลข้อมูลในระบบดิจิตอล ในระบบที่ยกมาเป็นตัวอย่างนี้การเปลี่ยนแปลงทางกายภาพในลักษณะใดๆ ก็ตาม (physical process) เช่นความดัน อุณหภูมิ ฯลฯ จะถูกเปลี่ยนให้เป็นสัญญาณไฟฟ้าที่มีความต่อเนื่อง(สัญญาณอนาลอก) โดยทรานสดิวเซอร์ที่คุณสมบัติที่เหมาะสมกับรูปแบบทางกายภาพนั้น สัญญาณไฟฟ้านั้นถูกปรับให้อยู่ในรูปและขนาดที่เหมาะสมก่อนโดย Analog signal conditioner เช่น วงจรขยาย หรือ ฟิลเตอร์ เป็นต้น ADC จะทำหน้าที่เปลี่ยนรูปแบบของสัญญาณจากอนาลอกเป็นดิจิตอล ตัวประมวลทางดิจิตอล (Digital processors) เช่นคอมพิวเตอร์ จะจัดการกับข้อมูลเพื่อนำเสนอหรือถูกเปลี่ยนกลับมาอยู่ในรูปแบบอนาลอกโดย DAC เพื่อป้อนกลับไปควบคุม Physical Process.



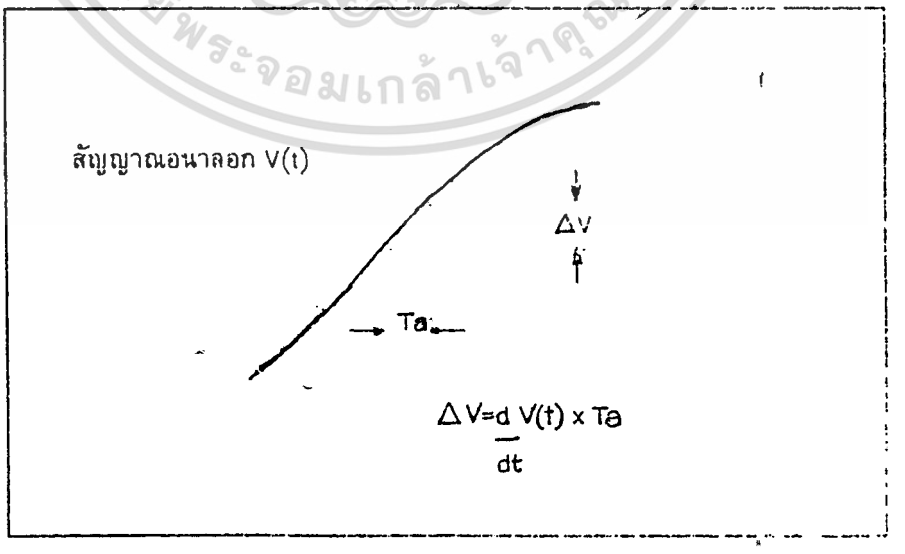
รูปที่ 4.1 ระบบควบคุมที่มีการประมวลข้อมูลแบบดิจิตอล

ในระบบที่มีข้อมูลที่ต้องประมวลในเวลาเดียวกันหลายๆ ข้อมูล หาก ADC ทำงานได้เร็วพอจะไม่จำเป็นต้องใช้ ADC หลายๆ ตัวทำงานแยกกันสำหรับข้อมูลแต่ละชุด แต่จะใช้วิธีแบ่งเวลา (timesharing)โดยวิธี multiplexing (รูปที่ 4.1) วงจร sampling and hold (S/H) จะสุ่ม (sample) ขนาดของสัญญาณอนาลอกมาและเก็บ (hold) ไว้ชั่วขณะเพื่อรอให้ ADC รับไปเปลี่ยนให้เป็นสัญญาณดิจิตอลจนเรียบร้อยแล้วค่อยสุ่มสัญญาณใหม่ ทั้งนี้เพื่อที่ไม่จำเป็นต้องใช้ ADC ที่มีความเร็วสูงราคาแพงข้อมูลดิจิตอลจะถูกส่งต่อไปยัง System bus และถูกประมวลโดย Processor ผลของการประมวลจะถูกส่งกลับออกมา เพื่อเปลี่ยนสัญญาณอนาลอกโดย DAC เพื่อไปควบคุมกิจกรรามทางกายภาพของระบบผ่าน Analog actuator

<u>ทฤษฎีการ Sampling</u>

ในการแปลงสัญญาณอนาลอกเป็นดิจิตอลเป็นรหัสดิจิตอลนั้น ADC ต้องใช้เวลาช่วงหนึ่งในการจัดการ ซึ่งช่วงเวลาดังกล่าวนั้นขึ้นอยู่กับหลายๆ แฟคเตอร์เช่น ความละเอียดของการเปลี่ยนสัญญาณ (จำนวนดิจิตอลบิท) เทคนิคของการเปลี่ยนแปลงสัญญาณ และความเร็วในการทำงานของอุปกรณ์ร่วมอื่นๆ การกำหนดความเร็วของการแปลงสัญญาณนี้ขึ้นอยู่กับการประยุกต์ใช้งานเฉพาะอย่าง และความแม่นยำที่ต้องการ

ช่วงเวลาในการแปลงสัญญาณบางครั้งอาจเรียกว่า Aperture time ซึ่งความหมายโดยทั่วไปหมายถึงช่วงเวลาที่เกิดความไม่แน่นอนขึ้นในการวัด และผลก็คือเกิดความผิดพลาด (error) ต่อค่าที่วัดได้



รูปที่ 4.2 แสดง error จากการวัดใน Aperture time

ในรูปที่ 4.2 สัญญาณอนาลอก V(t) มีอัตราการเปลี่ยน dv/dt ในช่วง Aperture-time t_a ดังนั้นช่วงการเปลี่ยนแปลงอนาลอกจะเท่ากับ V โดย

$$V = T_a \frac{d\,V(t)}{dt}$$

ดังนั้นหากเวลาที่ ADC ใช้ในการเปลี่ยนสัญญาณในช่วงเวลา T_a นี้รหัสดิจิตอลที่ได้อาจจะตรงกับขนาดของสัญญาณอนาลอกค่าใดค่าหนึ่งในช่วงเวลานี้และสันทำได้ง่ายและราคาถูกกว่าที่เกิดขึ้น ซึ่งแน่นอนว่าในบางครั้งเป็นไปได้ที่รหัสดิจิตอลจะตรงกับขนาดของสัญญาณอนาลอกที่เกิดขึ้น เรียก error ที่เกิดขึ้นนี้ว่า Aperture time error.

ตัวอย่างในกรณีสัญญาณอินพุทเป็นรูปซายน์ อัตราการเปลี่ยนแปลงบนรูปคลื่นจะเกิดสูงที่สุดตรงบริเวณจุดตัดแกนเวลารอบๆ จุดศูนย์โวลท์ (Zero Crossing) และ Aperture-time error คือ

$$V = T_a \frac{d}{dt}(A\sin\omega t)_{t=0} = T_a A\omega$$

และ error รวม (E)คิดจากอัตราส่วนของขนาดเต็มสเกล คือ

$$E = \frac{\Delta V}{2A} = \pi f T_a$$

ดังนั้นหากต้องการเปลี่ยนสัญญาณเป็นรูปซายน์ความถี่ 1 กิโลเฮิร์ท ให้เป็นสัญญาณดิจิตอล 10 บิท ซึ่งยอมให้ error ไม่เกินกว่า resolution (จะกล่าวถึงภายหลัง) คือ $1/2^{10}$ LSB หรือ 0.001 ดังนั้นเวลา Aperture time จะต้องอยู่ในช่วง

$$T_a = \frac{E}{\pi f} = \frac{0.001}{3.1444\times10^3} = 320\times10^{-9}$$

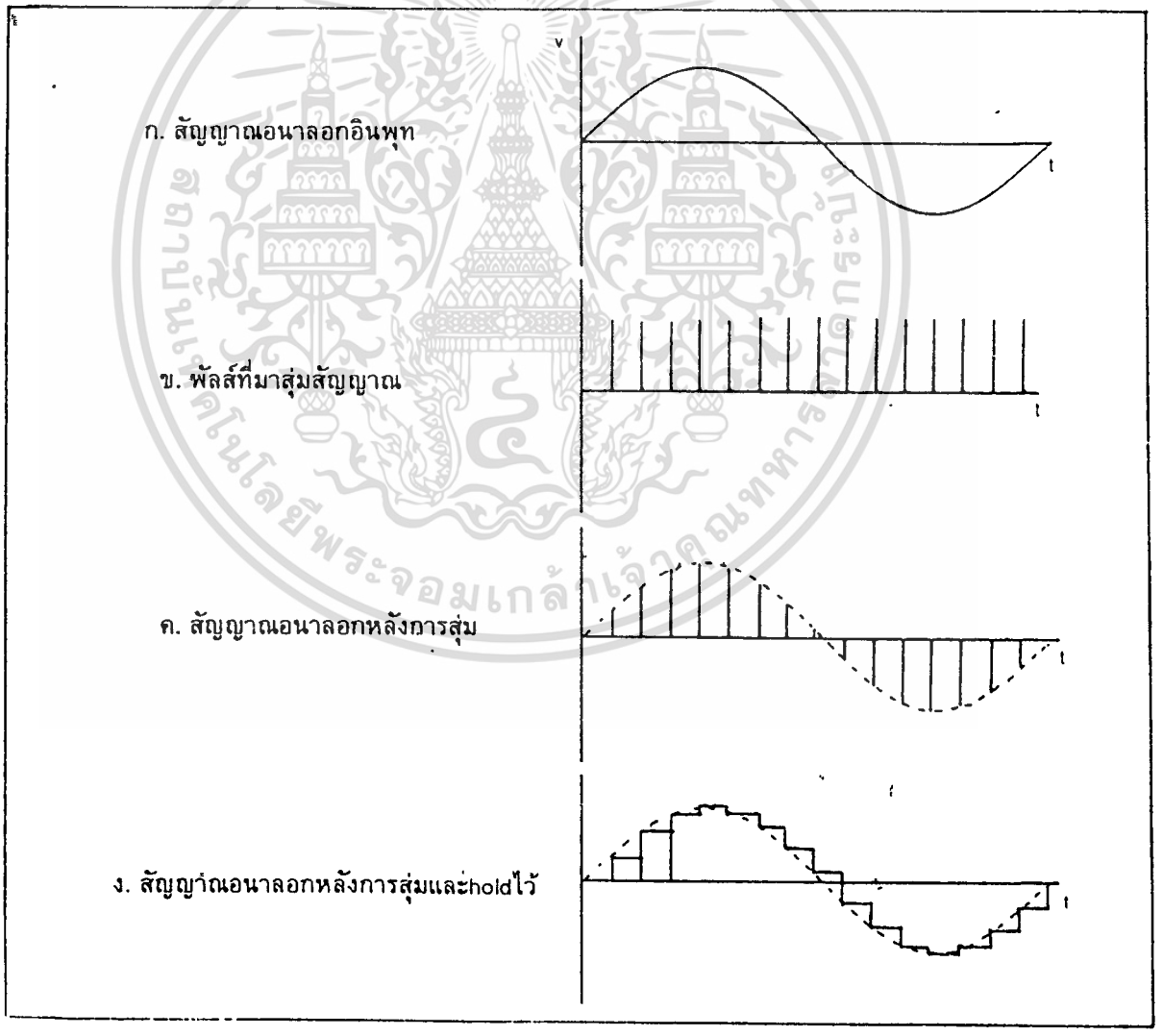
จะเห็นว่าแม้สัญญาณ 1 กิโลเฮริ์ท จะไม่ใช่ความถี่สูงก็จริง แต่ ADC ที่ใช้ต้องการเวลาในการเปลี่ยนในเวลา 320 นาโนวินาที ให้เป็นรหัส 10 บิท วิธีอื่นที่ไม่จำเป็นต้องใช้ ADC ความเร็วสูงคือการใช้ Sample and Hold ซึ่ง Sample and Hold ที่มี Aperture time น้อยๆ นั้นทำได้ง่ายและราคาถูกกว่า

Sample and Hold และ Aperture error

วงจร Sample and Hold จะทำการสุ่ม (Sampling) สัญญาณอินพุท และนำสัญญาณที่สุ่มนั้นมาเก็บ (hold) ไว้ในช่วงเวลาหนึ่งได้ ซึ่งส่วนใหญ่จะใช้การประจุแรงดันนั้นไว้ในตัว

เก็บประจุที่รั่วไหลต่ำ Aperture time ของ Sample & Hold คือเวลาตั้งแต่เริ่มสุ่มสัญญาณจนเก็บประจุค่าแรงดันจนถึงค่าสุ่มซึ่งสำหรับ Sample and Hold แล้ว Aperture time ขึ้นอยู่กับแบนด์วิดธ์ และ Switching time ของอุปกรณ์เอคทีฟ (จะกล่าวภายหลัง) ที่ใช้ในวงจร ซึ่งหาและสร้างได้ง่ายและราคาถูกกว่าการสร้าง ADC ความเร็วสูง

ในการสุ่มสัญญาณอนาลอกจะถูกสุ่มเป็นระยะๆ คงที่ตามรูป 4.3 การสุ่มจะเป็นการตัดต่อสัญญาณอนาลอกในช่วงเวลาอันสั้นด้วยสวิทช์ที่ทำงานด้วยความเร็วสูง ผลของการสุ่มสัญญาณด้วยความเร็วจะเสมือนกับการคูณขบวนสัญญาณพัลส์แคบๆ กับสัญญาณอนาลอกซึ่งจะได้เป็นสัญญาณที่มองดูแล้วระหว่างขบวนการพัลส์กับสัญญาณอนาลอก โดยเสมือนว่าสัญญาณอนาลอกจะขี่มาบนขบวนพัลส์ ดังแสดงในรูปถ้าหากสัญญาณอนาลอกที่ถูกสุ่มถูก Hold จนกว่าสัญญาณค่าใหม่ถูกสุ่มเข้ามาซึ่งจะได้ลักษณะของเอ้าท์พุทที่ได้แสดงไว้ในรูป ง.

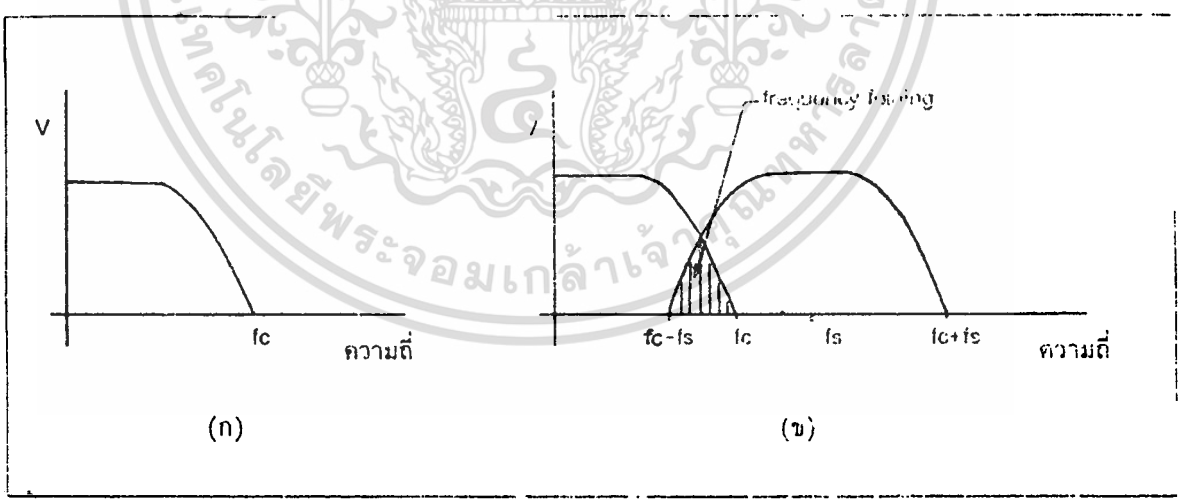


รูปที่ 4.3 การสุ่มสัญญาณ

มีปัญหาที่ว่าอัตราการสุ่มสัญญาณนั้นควรมีขนาดเท่าใด จึงจะไม่ทำให้ข้อมูลสูญเสียไปเมื่อสัญญาณนั้นถูกเปลี่ยนกลับมาเป็นเช่นเดิม คำตอบก็คือ ขึ้นอยู่กับความถี่ของสัญญาณอนาลอก ทฤษฎีของการสุ่มกล่าวไว้ว่า "ถ้าสัญญาณต่อเนื่องซึ่งมีความถี่และฮาร์โมนิคส์ไม่เกิน f_c ถูกสุ่มด้วยอัตราการสุ่มไม่น้อยกว่า $2f_c$ แล้วสัญญาณดังกล่าวจะสามารถเปลี่ยนกลับมาได้อย่างเดิมโดยไม่สูญเสีย รายละเอียดหรือผิดเพี้ยนไป"

Frequency folding and Allasing

จากทฤษฎีของการสุ่มสามารถอธิบายด้วยลักษณะรูปสเปคตรัมของสัญญาณในรูปที่ 4.4 รูป (ก) แสดงให้เห็นสเปคตรัมของสัญญาณที่ถูกสุ่มซึ่งแบนด์วิดธ์ไม่เกิน f_c ในขณะที่สัญญาณนี้จะถูกสุ่มด้วยความถี่ f_s ขบวนการมอดูเลชั่นจะทำให้แถบสเปคตรัมของสัญญาณสุ่มขยายกว้างออกจาก f_s เป็น $2f_s$ $3f_s$.... ได้เป็นดังรูป 4.4 (ข) ถ้าความถี่ของสัญญาณสุ่ม f_s ไม่สูงพอหลังจากการสุ่มสเปคตรัมบางส่วนของ f_s จะหาซ้อนกลับสเปคตัมของสัญญาณที่เรียกว่า frequency Folding หากเป็นเช่นนี้ก็จะทำให้เกิดความเพี้ยนแก่สัญญาณอนาลอกจากการซ้อนกันของสเปคตัมเมื่อสัญญาณถูกเปลี่ยนกลับให้อยู่ในรูปเดิม

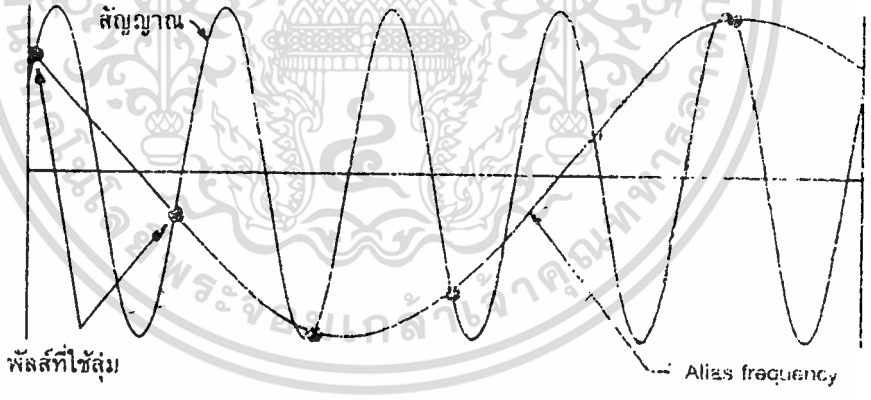


รูป 4.4 (ก) แสดงสเปคตัมของสัญญาณอนาลอกที่จะถูกสุ่ม

รูป 4.4 (ข) สเปคตัมหลังจากการสุ่ม เกิด Frequency folding

และถ้าเลื่อนความถี่ของการสุ่มให้สูงขึ้นจนโอกาสการซ้อนของสเปคตัมหมดไป ($f_s - f_c = f_c$) และการเปลี่ยนกลับของสัญญาณหลังจากถูกสุ่มก็ยังคงเหมือนเดิมได้

จากที่กล่าวมาแสดงถึงการสนับสนุนทฤษฎีการสุ่มที่ว่าให้ $f_s > 2f_c$ นั้นก็คือการกำจัดการซ้อนกันของสเปคตัมซึ่งทำได้สองวิธี วิธีหนึ่งด้วยการใช้อัตราการสุ่มที่สูงพอและ อีกวิธีหนึ่งคือการทำการฟิลเตอร์ความถี่ของสัญญาณอนาลอกก่อนการสุ่ม(Anti alias filters) เพื่อให้แบนด์วิดธ์ไม่เกินไปกว่า fs/2 ในทางปฏิบัติแล้วจะยังคงเกิด frequency folding ได้เสมอจากส่วน ฮาร์โมนิคส์ของสัญญาณรวมทั้งสเปคตัมของสัญญาณรบกวนที่ยังคงอยู่แม้ว่าทำการฟิลเตอร์ก่อนหน้ามาแล้วก็ตามการกำจัดการซ้อนกันของสเปคตัม ดังวิธีนี้ได้ผลอีกก็คือ พยายามให้การสุ่มสัญญาณเป็นไปอย่างรวดเร็วมากที่สุด ซึ่งปกติจะสูงกว่าความถี่ต่ำสุดตามทฤษฎี Sampling คือ $2f_c$ เสมอ



รูปที่ 4.5 การเกิด Alias frequency จากการสุ่มด้วยความถี่ต่ำกว่า 2 เท่าของความถี่สัญญาณอินพุทรูปซายน์

ผลของการใช้อัตรากสุ่มที่ไม่เหมาะสมจะเกิดเป็นสัญญาณความถี่ต่ำ เรียกว่า Alas Frequency เมื่อสัญญาณถูกเปลี่ยนกลับมาเช่นเดิมหลังจากถูกสุ่มแล้วแสดงใน รูปที่ 4.5 จะเห็นว่าความถี่ alias อาจจะแตกต่างจากความถี่เดิมไปมาก

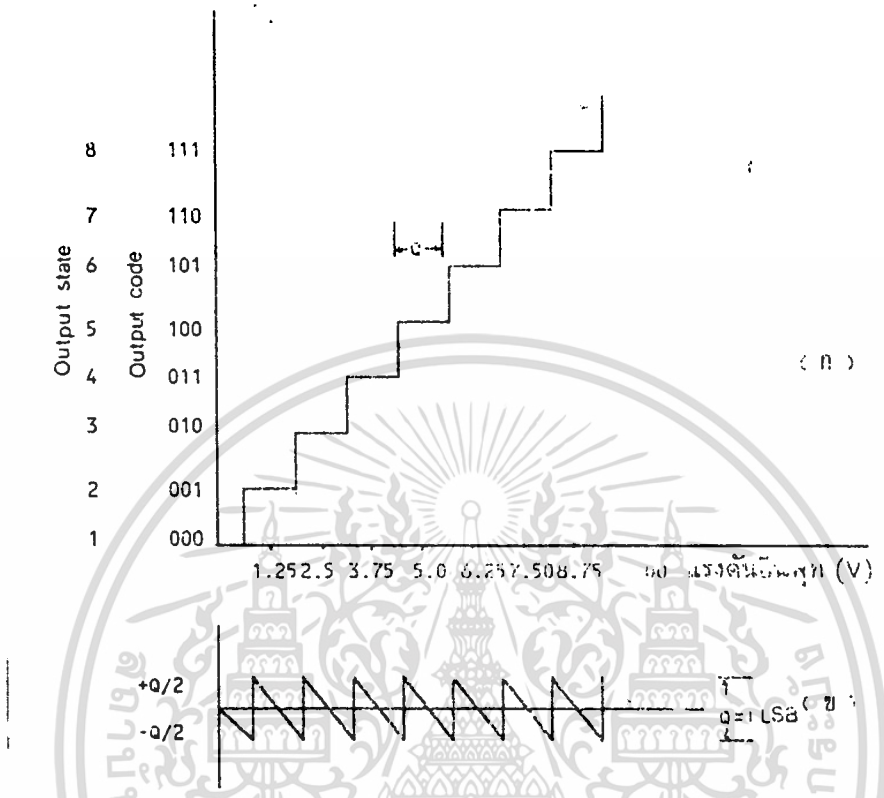
Anti alais filter จะช่วยลดสัญญาณในแถบความถี่ที่ทำให้เกิด alias frequency ในขณะที่ต้องไม่ทำให้เกิดความผิดเพี้ยนของสัญญาณในแบนด์ที่ใช้งานและไม่ลดความแม่นในการวัดโดยรวมอีกด้วย ในการใช้ Anti alais filter ปริมาณการขจัดความถี่สูงนั้นขึ้นอยู่กับ :

- ความถี่สูงสุดที่สนใจ
- อัตราการสุ่ม และ
- ความละเอียดของภาวะแปลงสัญญาณ

ฟิลเตอร์ที่ใช้จึงอาจเป็น พาสซีฟฟิลเตอร์ แอคทีฟฟิลเตอร์ หรือ switched capacitor ฟิลเตอร์

Quantlzing theory

Quantizing เป็นขบวนการที่เปลี่ยนสัญญาณอนาลอกเป็นสัญญาณ ที่ไม่ต่อเนื่อง (discrete signal) หลังการสุ่ม โดยผ่านขบวนการ เข้ารหัส (Coding) จัดให้สัญญาณที่ไม่ต่อเนื่องนั้นอยู่ในรูปที่ง่ายต่อการประมวลและเป็นสัดส่วนสัมพันธ์กับสัญญาณอนาลอก เช่น ในรูปของรหัสไบนารี(Binary)เป็นต้น หากนำเอาขนาดของสัญญาณอนาลอกและรหัสดิจิตอลที่ได้จากการ Quantize มาเขียนกราฟก็จะได้กราฟแสดง Quantize transter function ดังรูปที่ 4.6



รูปที่ 4.6 Transfer function ของ Quantize 3 บิท ตามทฤษฎี

ในรูปกราฟแสดงให้เห็นถึงความสัมพันธ์กัน ระหว่างสัญญาณอนาลอกที่ขนาดอยู่ระหว่าง 0 ถึง +10 โวลท์ ถูก Quantize และ encode เป็นรหัสไบนารี่(Binary) 3 บิทได้ 8 ระดับ จาก 0000 ถึง 111 เนื่องจากในระบบไบนารี่รหัสดิจิตอลแต่ละค่าจะแทนขนาดของสัญญาณอนาลอกแต่ละค่าที่เป็นสัดส่วนกับค่าเต็มสเกลโดยค่าสูงสุดของรหัสดิจิตอลคือทุกบิทเป็น 1 จะเท่ากับสัญญาณอนาลอกเต็มสเกลคูณด้วย $(1-2^{-n})$ โดย n เป็นจำนวนบิทของรหัสดิจิตอล และรหัสดิจิตอลแต่บิทที่เป็น 1 จะเท่ากับขนาดเต็มสเกลของอนาลอกคูณกับค่า weighting ของรหัสชนิดนั้นหารด้วย 2^n ตัวอย่างเช่น ค่าเต็มสเกลของสัญญาณอนาลอกเป็น 10 โวลท์ รหัส 1011 จะแทนขนาดสัญญาณอนาลอกอินพุท

$$V_{input} = \frac{R_s}{2^n}\{(1x2^3)+(0x2^2)+(1x2^1)+(1x2^0)\}$$

$$= \frac{10}{2^4}\{(1x2^3)+(0x2^2)+(1x2^1)+(1x2^0)\}$$

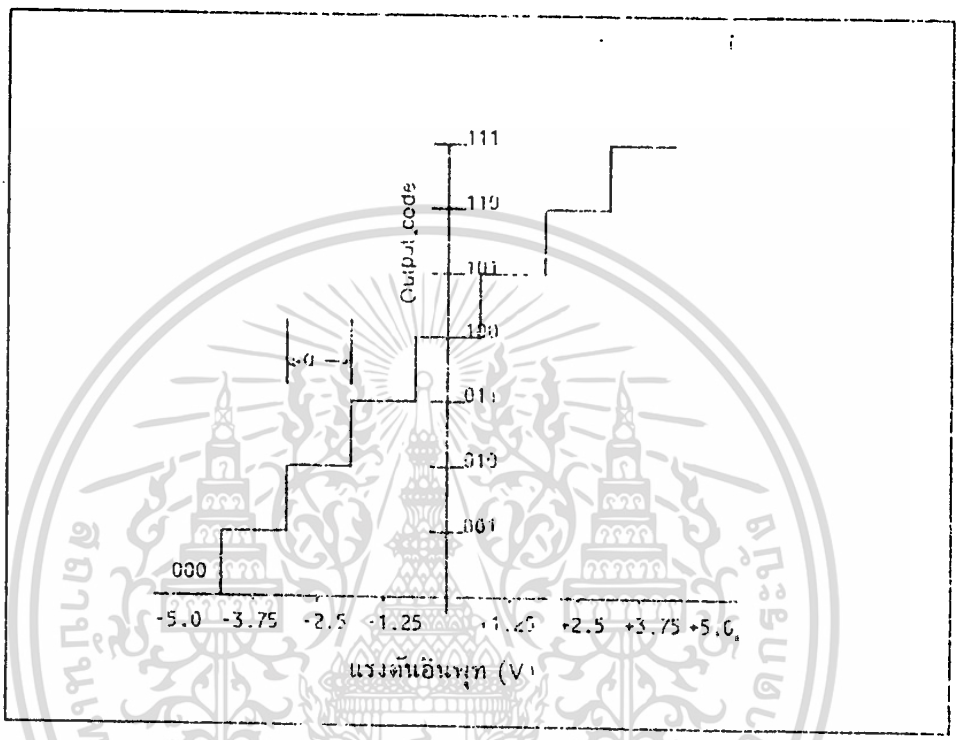
จุดสำคัญที่เกี่ยวกราฟ transfer fuction ในรูปที่ 4.6 อันแรกได้แก่ ความละเอียด (resolution) ของ quantizer ซึ่งกำหนดได้จากจำนวนบิทของรหัสดิจิตอล หรือจากกราฟคือขนาดกว้างของ ขั้นระดับ(step) ทางแกนอนาลอกอินพุทว่าเป็นสัดส่วนเท่าใดระหว่าง ค่าเต็มสเกลอนาลอกกับค่า 2^n

จำนวนสถานะเอ้าท์พุทกำหนดได้จากจำนวนบิทคือเท่ากับ 2^n สถานะ ตัวอย่างกรณี ADC 8 บิท Quantizer จะให้เอ้าท์พุท 256 สถานะและ 12 บิทให้ 4096 สถานะต่อค่าเต็มสเกลของอนาลอก ในไดอะแกรมแสดงทรานสเฟอร์ฟังชั่น จะเห็นจุดแบ่ง ระดับ(Decision point หรือ Theshold level) สัญญาณอนาลอกจะมีจำนวน 2^n -1 จุดที่อยู่ที่ 0.625, 1.875, 3.125, 4.375, 5.625, 6.875, และ 8.125 โวลท์ ระหว่างจุดดังกล่าวเป็นสัญญาณอนาลอกซึ่งแปลงเป็นรหัสดิจิตอล 1 สถานะ ดังนั้นค่าเหล่านี้จะต้องปรับให้ถูกต้องมากที่สุดเพื่อแปลงขนาดของอนาลอกให้ตรงกับค่าที่ทำการ Quantized แรงดันที่ 1.25, 2.50, 3.75, 5.0, 6.25, 7.2 และ8.75 โวลท์ เป็นจุดกึ่งกลางในช่วงของสัญญาณอนาลอกที่แสดงสถานะเอ้าท์พุทดิจิตอลฟังชั่นที่มีลักษณะเป็นขั้นบันไดนี้สามารถประมาณเป็นเส้นตรงได้ โดยการโยงเส้นตรงระหว่างจุดเริ่มและจุดปลาย ณ จุดกึ่งกลางของรหัสดิจิตอลสถานะสุดท้าย สังเกตุว่าในทางทฤษฎีแล้วเส้นตรงนี้จะต้องผ่านจุดกึ่งกลางของทุกระดับดิจิตอล

รหัสตัวเลขสำหรับการเปลี่ยนข้อมูล

รหัสตัวเลขที่นิยมนำมาใช้ในระบบเปลี่ยนข้อมูลได้แก่รหัสไบนารีหรือที่เรียกว่า Straight binary โดยที่รหัสไบนารีสถานะสูงสุดจะแทนสัญญาณอนาลอก FRS x $(1-2^n)$ โวลท์ ตัวอย่างเช่น หากสัญญาณอนาลอกเต็มสเกล (FRS) เท่ากับ 20 โวลท์ สำหรับ ADC ขนาด 12 บิท รหัส 1111 1111 1111 จะแทนสัญญาณอนาลอกขนาด $20x(1-2^{12})$หรือ19.9951171 โวลท์ นอกจากรหัสไบนารีธรรมดาดังกล่าวยังมีการใช้ระบบไบนารีแบบอื่นๆ ในระบบการแปลงสัญญาณ แก่ ออฟเซ็ทไบนารี,Two's complement, BCD ซึ่งแต่ละชนิดมีข้อดีและความเหมาะสมต่างกัน ตัวอย่างเช่นระบบ BCD เหมาะสำหรับการแสดงตัวเป็นตัวเลขหน้าปัทม์ หรือต่อเข้ากับดิจิตอลมิเตอร์รหัส Two's complement เหมาะสำหรับการคำนวณทางคณิตศาสตร์ลอจิก และสำหรับระบบออฟเซ็ทไบนารีนั้นเหมาะสำหรับการแปลงสัญญาณอินพุทที่ทั้งช่วงบวกและลบ ในรูปที่ 4.7 แสดง transfer function ของ ADC 3 บิท ที่ใช้รหัสออฟเซ็ทไบนารี

นอกจากมาตรฐานของการใช้รหัสตัวเลขแล้วยังมีมาตรฐานของการเลือก ช่วงของขนาดแรงดันอินพุทสำหรับ ADC คือ หากเป็นสัญญาณช่วงบวกหรือลบอย่างเดียวจะใช้ 0-5 โวลท์หรือ 0-10 โวลท์ แต่ถ้าเป็นช่วงลบจะใช้ 2.5 โวลท์,-5 โวลท์และ -10 โวลท์ เป็นมาตรฐาน



รูปที่ 4.7 Transfer function ของ ADC 3 บิทที่ใช้รหัสออฟเซ็ทไบนารี

Analog to Digital Converter (ADC)

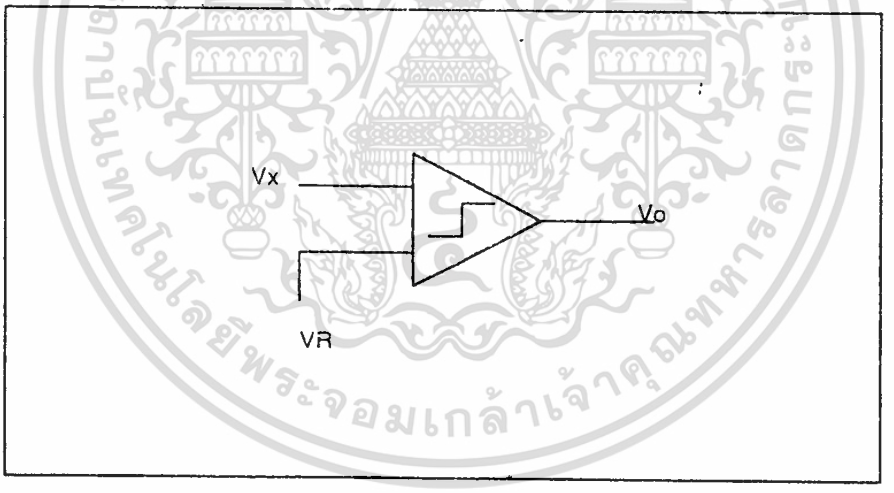
ลักษณะการจัดวงจร ADC มีหลายแบบ แต่ที่นิยมใช้มีเพียงไม่กี่แบบและส่วนใหญ่จะอยู่ในรูปของวงจรรวม

<u>Basic conversion method</u>

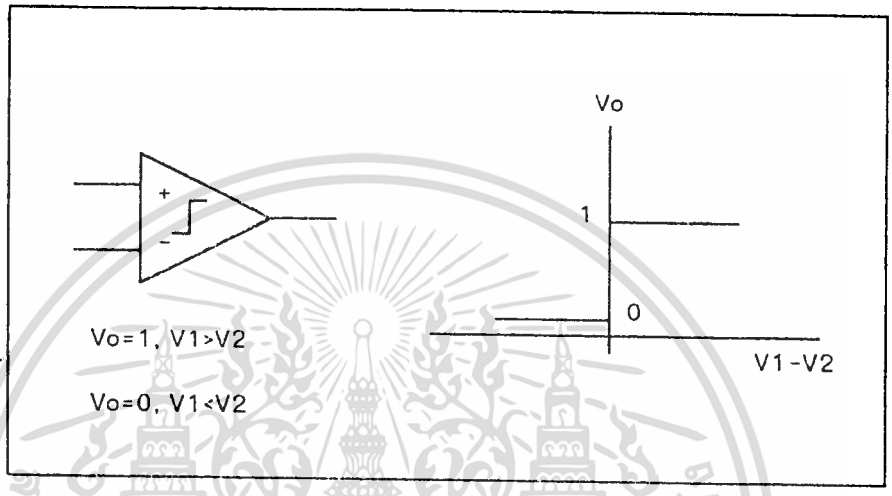
วิธีการแปลงสัญญาณอนาลอกเป็นดิจิตอลแบบง่ายๆ แสดงในรูปที่ 4.8 แรงดันอินพุทที่ไม่ทราบค่า V_x จะต่อเข้ากับขาอินพุทขาหนึ่งของอนาลอกคอมพาเรเตอร์ และแรงดันอ้างอิงที่ขนาดแปรตามเวลา V_R ต่อเข้ากับอินพุทอีกขาหนึ่งของคอมพาเรเตอร์ ลักษณะของทรานเฟอร์ฟังชั่น ของคอมพาเรเตอร์แสดงในรูปที่ 4.9 ถ้าแรงดันอินพุท V_1 มากกว่า อินพุท V_2 แล้วแรงดันเอ้าท์พุทจะเป็นลอจิก 1 ถ้าอินพุท V_1 มากกว่า V_2 แล้วเอ้าท์พุทจะเป็นศูนย์

วิธีในการแปลงข้อมูลคือ แรงดันอ้างอิงจะถูกแปรค่าจนกระทั่งรู้ค่าแรงดันอินพุทที่ผิดพลาดไม่เกิน Quantization error ของคอนเวอร์เตอร์ ในแนวความคิดแล้วตรรกะของ ADC คือพยายามเลือกกลุ่มของส.ป.ส.ไบนารี a_i เพื่อให้ผลต่างระหว่างแรงดันอินพุท V_x และค่าที่ Quantize ได้ครั้งสุดท้าย น้อยกว่า 0.5 LSB ซึ่งเขียนเป็นสมการได้

$$\left| V_x - V_{FSR} \sum_{i=1}^{n} a_i 2^i \right| < 0.5LSB$$



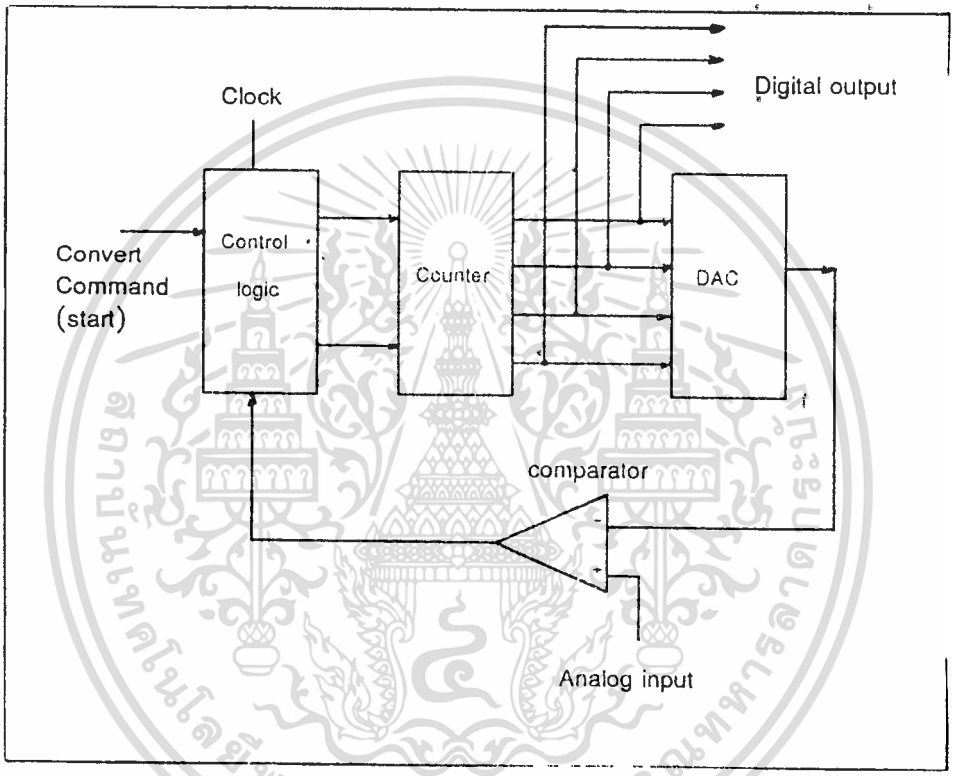
รูปที่ 4.8 แสดงวิธีการพื้นฐานของ ADC



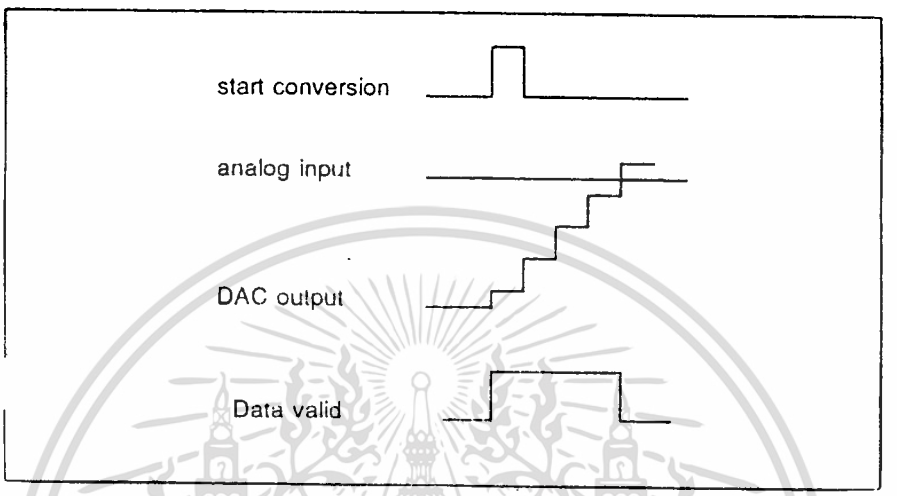
รูปที่ 4.9 แสดงทรานสเฟอร์ฟังชั่นของคอมพาเรเตอร์

Counter type ADC

การจัดวงจร ADC ลักษณนี้เป็นแบบที่ง่ายที่สุด หลักการทำงานของวงจรคือ การเปรียบเทียบขนาดของแรงดันที่เอาท์พุทของ DAC กับสัญญาณอนาลอกที่ไม่ทราบค่า V_{in} การทำงานจะเริ่มโดยสัญญาณ start conversion ลอจิกคอนโทรลจะรีเซ็ทเคาท์เตอร์ให้เป็นศูนย์ แล้วเริ่มนับขึ้นจากศูนย์เอาท์พุทของเคาท์เตอร์จะป้อนให้ DAC เพื่อแปลงเป็นสัญญาณอนาลอกลักษณะเป็นขั้นบันได นำมาเปรียบเทียบกับสัญญาณอนาลอกอินพุทที่คอมพาเรเตอร์ โดยเคาท์-เตอร์จะยังนับจนกระทั่งเอาท์พุทเท่ากับสัญญาณอนาลอกอินพุทหรือต่างกันไม่เกิน 1 LSB แล้วคอมพาเรเตอร์จะเปลี่ยนสถานะไปหยุดการนับของเคาท์เตอร์และ latch ค่าจากเคาท์เตอร์เพื่อรอการประมวลต่อไป และรอรับสัญญาณ start ใหม่



รูปที่ 4.10 บล็อกไดอะแกรมของ counter type DAC



รูปที่ 4.11 timing diagram counter type DAC

วงจรนี้มีข้อเสียที่ ทำงานได้ช้าเพราะการ Conversion แต่ละครั้งเคาท์เตอร์จะต้องถูกรีเซ็ทและเริ่มนับจากศูนย์ทุกครั้งดังนั้นในการ conversin เป็นดิจิตอล n บิทจะใช้จำนวน clock ถึง 2^n เพื่อเปลี่ยนให้ได้ค่าสูงสุดเต็มสเกล ส่วนข้อดีคือสร้างได้ง่ายรวดเร็วราคาถูก แต่ความแม่นยำขึ้นอยู่กับ DAC ที่ใช้

Tracking ADC

Tracking ADC จะปรับปรุงวงจรแบบ counter type ทางด้านความเร็ว โดยใช้เคาท์เตอร์แบบนับขึ้นลงได้ไม่จำเป็นต้องเริ่มจากการนับจากศูนย์ทุกครั้ง แต่จะเริ่มนับจากค่าที่ได้ Latch ไว้จากการเปลี่ยนสัญญาณครั้งหลังสุด ดังนั้นส่วนควบคุมทางลอจิกจึงซับซ้อนมากกว่า โดยการทำงานจะเป็นดังนี้ เอ้าท์พุท DAC จะถูกเปรียบเทียบกับสัญญาณอินพุท (Vin) หาก Vin มากกว่า ลักษณะลอจิกของคอมพาเรเตอร์จะควบคุมให้เคาท์เตอร์นับขึ้น แต่ถ้า Vin น้อยกว่าเคาน์เตอร์จะนับลงจนกว่าค่าหลังสุดของเคาน์เตอร์จะต่างจากสัญญาณอนาลอก อินพุทไม่

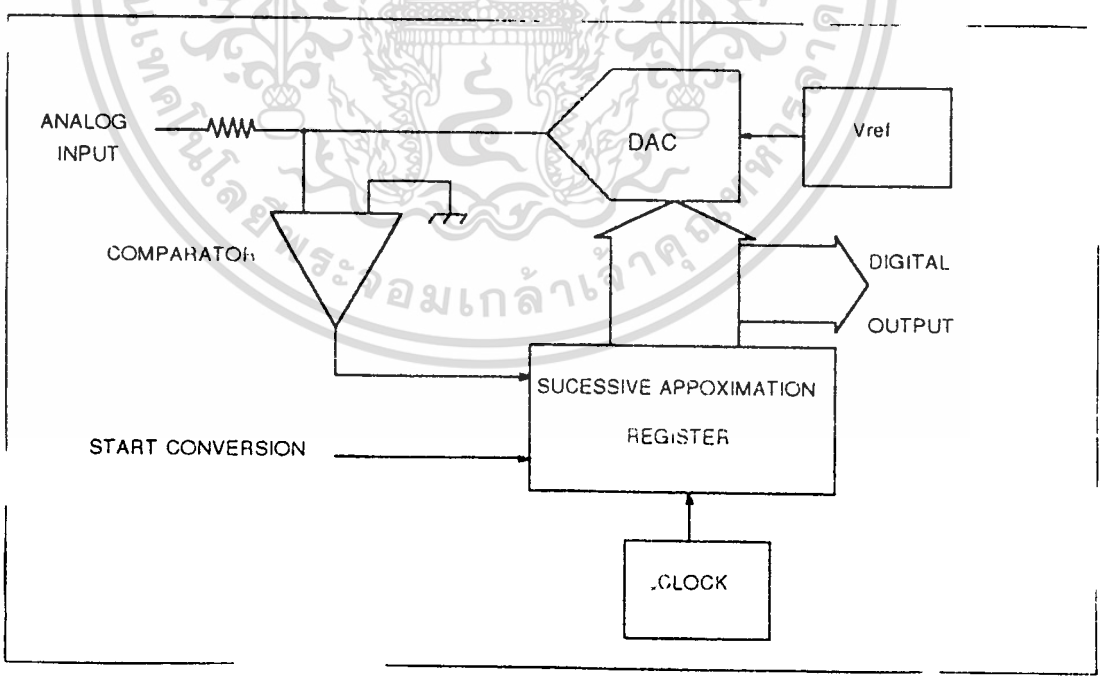
เกิน 1 LSB และค่าของเคาน์เตอร์จะถูก latch ไว้จากนั้นเคาน์เตอร์จะทำงานแบบติดตาม (track) สัญญาณอินพุทจนได้ค่าเท่ากันอีกก็จะ latch ค่าใหม่ไว้

จากลักษณะการทำงานดังกล่าว V_{in} จะต้องไม่เปลี่ยนแปลงเร็วมากกว่าการทำงานของเคาน์เตอร์ มิฉะนั้นค่าเอ๊าท์พุทที่ได้จะไม่สอดคล้องกับสัญญาณอินพุท ตัวอย่าง ในกรณีสัญญาณรูปซายน์ซึ่งเปลี่ยนแปลงขนาดได้มากที่สุดเท่ากับค่าเต็มสเกล อัตราการเปลี่ยนแปลงจะเท่ากับอัตราการเปลี่ยนแปลงเอ๊าท์พุทของเคาน์เตอร์ คือ 1 LSB/clock period ดังนั้นถ้าต้องการให้ ADC ตามอินพุทได้จะต้องให้

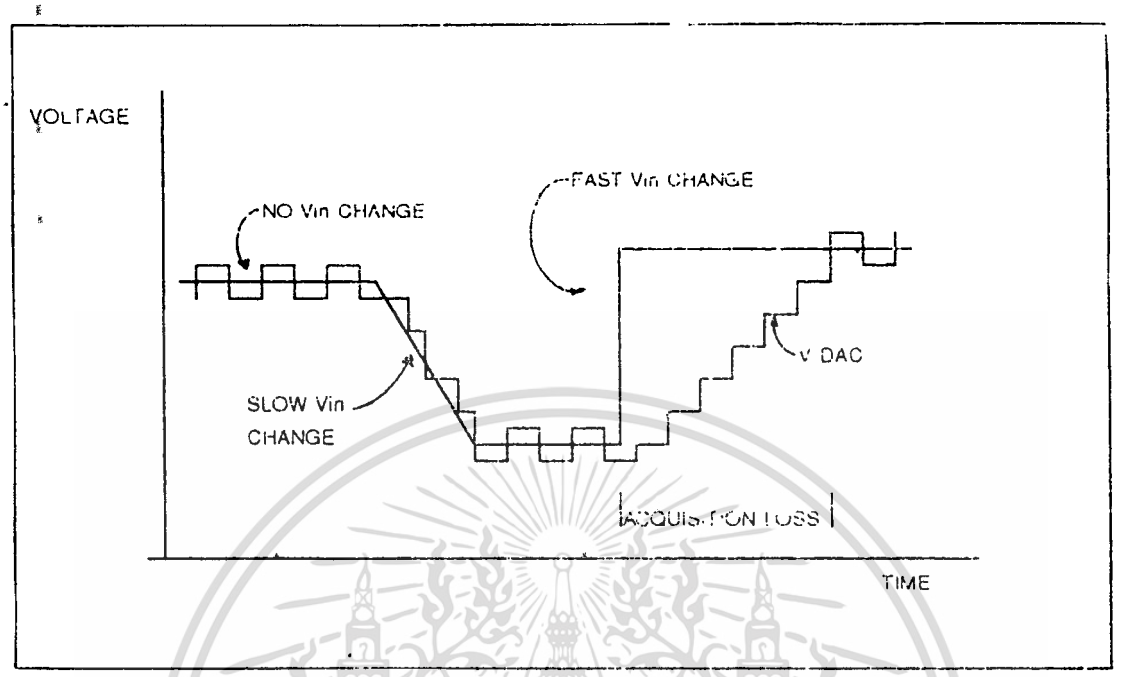
$$\frac{V_{FS} f_0}{2} < \frac{V_{FS} f_c}{2^n}$$

และ $$f_0 < \frac{f_c}{2^n}$$

f_c คือความถี่ของ clock



รูปที่ 4.12 บล็อกไดอะแกรมของวงจร tracking converter

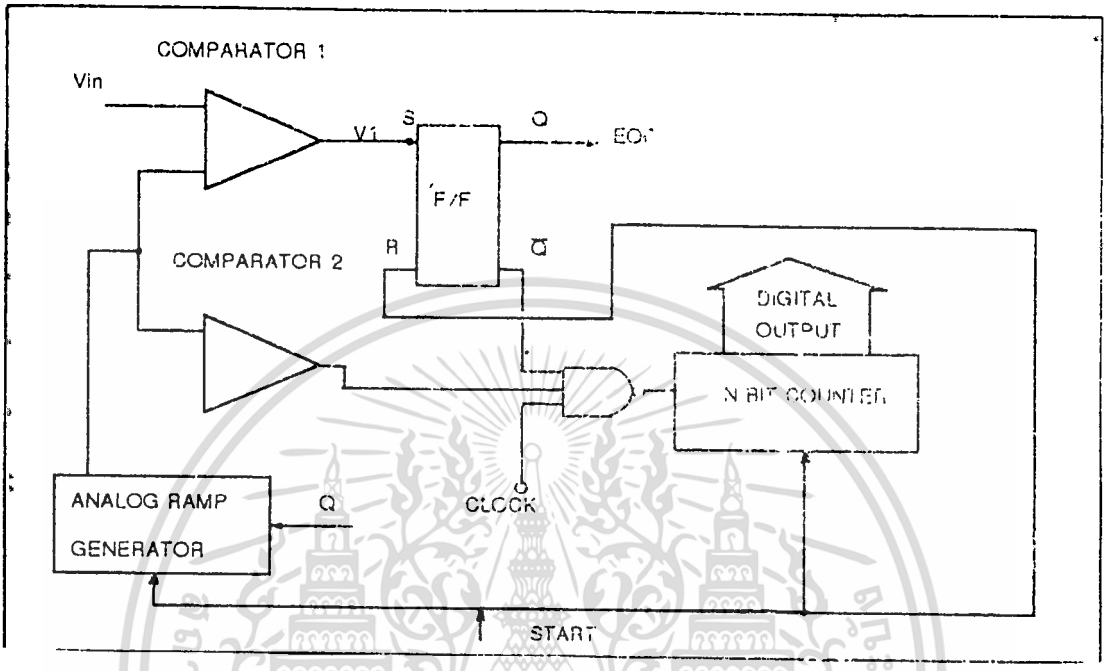


รูปที่ 4.13 Timming Diagram

Integrating ADC

หัวใจสำคัญของวงจร ADC ชนิดนี้คือวงจร integrator เทคนิคของ ADC แบบ Integration คือจะใช้สัญญาณ ramp ต่อเนื่องแทนสัญญาณขั้นบันไดจาก DAC ซึ่งแบ่งตามลักษณะการทำงานได้สองแบบ คือ Single Slope Converter และ Dual Slope Converter

1). Single Slope Converter



รูปที่ 4.14 Single slope converter

สัญญาณอนาลอกแบบ ramp จะใช้เป็นแรงดันอ้างอิงที่เพิ่มขึ้นอย่างคงที่จากค่าต่ำกว่าศูนย์เล็กน้อยจนถึงค่าที่สูงสุดค่าเต็มสเกลเล็กน้อย ซึ่งเวลาที่จะใช้จากการสแกนของสัญญาณ ramp จากศูนย์ถึงค่าแรงดันอินพุทจะเป็นสัดส่วนกับแรงดันอินพุท

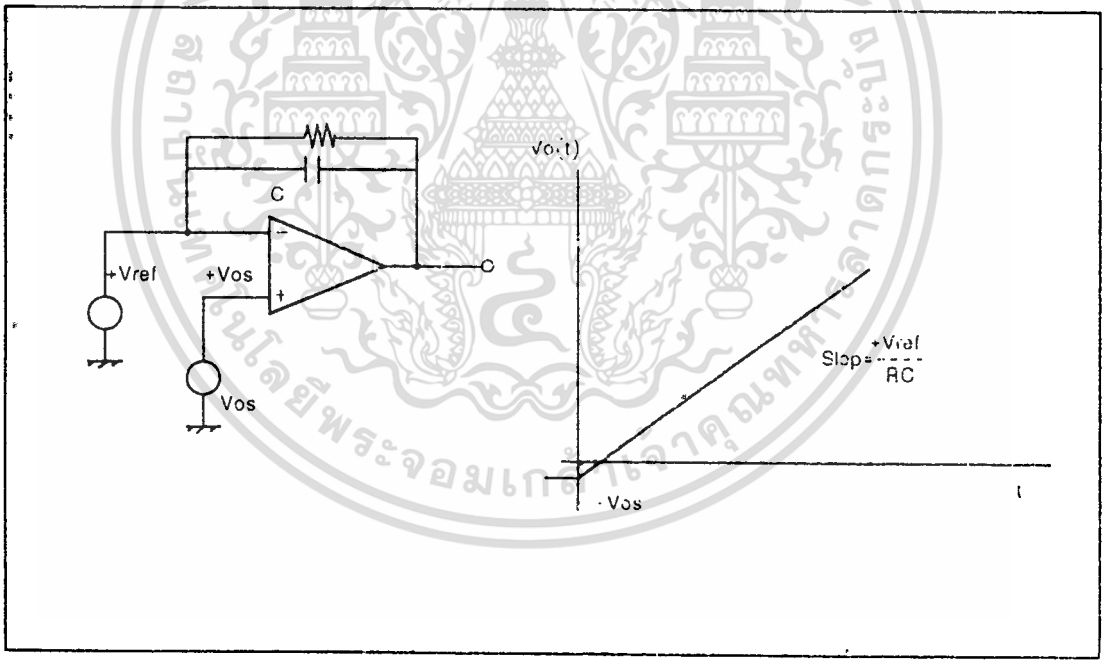
การ convesin จะเริ่มด้วยสัญญาณ start conversion ทำการรีเซ็ทไบนารีเคาน์-เตอร์ และเริ่มสร้างสัญญาณ ramp จากแรงดันที่ต่ำกว่าศูนย์โวลท์ เมื่อสัญญาณ ramp ผ่านศูนย์โวลท์ เอ้าท์พุทจากคอมพาเรเตอร์ 2 จะ high และเปิดเกทปล่อยพัลส์เข้าสู่เคาน์เตอร์

เคาน์เตอร์จะเริ่มนับจนกระทั่งสัญญาณ ramp มีขนาดเท่าแรงดันอนาลอก อินพุท V_{in} ในเวลานี้เอ้าท์พุทจากคอมพาเรเตอร์ 1 จะ high และปิดเกทไม่ให้ clock เข้าสู่เคาน์เตอร์ จะเป็นสัดส่วนแรงดันกับอินพุท เนื่องจาก VR = KT โดย R เป็นสโลปของ ramp (ซึ่งคงที่) ในหน่วยโวลท์/วินาที และ T เป็นจำนวนในการเคาน์เตอร์หารด้วย f_c ซึ่งเป็นความถี่สัญญาณ

clock ถ้าเลือกให้สโลปของ ramp เป็น $V_{FSR}\ f_c/2^n$ จำนวนที่เคาท์เตอร์นับได้จะเท่ากับอัตราส่วนทางไบนารีหรือ V_{in}/V_{FSR}

เวลาในการเปลี่ยน T_c ของ ADC แบบนี้จะแปรเป็นสัดส่วนกับแรงดันที่อินพุท V_{in} เวลาที่ใช้ในการเปลี่ยนมากที่สุดเมื่อ $V_{in} = V_{FSR}$ คือ $T_{MAX} = 2^n/f_c$ และเช่นเดียวกับใน ADC แบบเคาน์เตอร์ ramp ค่าของรหัสเอ้าท์พุทสุดท้ายจะต่างจากค่าของ V_{in} ไม่เกิน 0.5 LSB

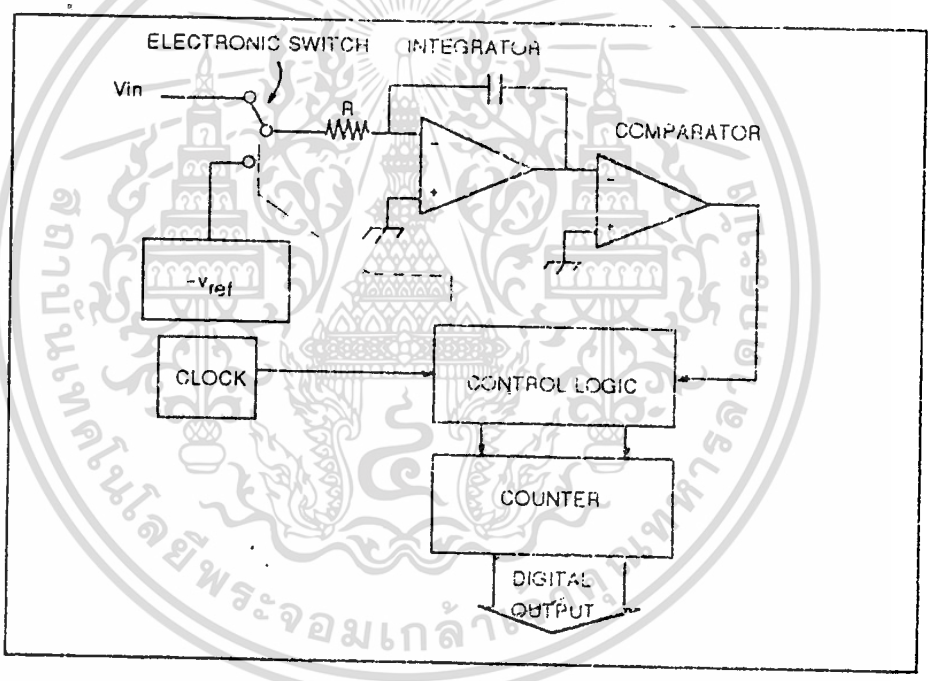
ในรูปที่ 4.15 แสดงวงจรกำเนิดแรงดัน ramp อย่างง่าย โดยการต่อแรงดันอ้างอิงกับอินทิเกรเตอร์เมื่อสวิทช์เปิด C จะทำการประจุและเพิ่มขนาดแรงดันเอ้าท์พุท ข้อเสียประการหนึ่งคือหาใช้งานไปนานๆ การเปลี่ยนแปลงค่า RC ตามอุณหภูมิจะทำให้สโลปคลาดเคลื่อนด้วย ADC ชนิดนี้ จึงไม่เป็นที่นิยมใช้ในปัจจุบัน



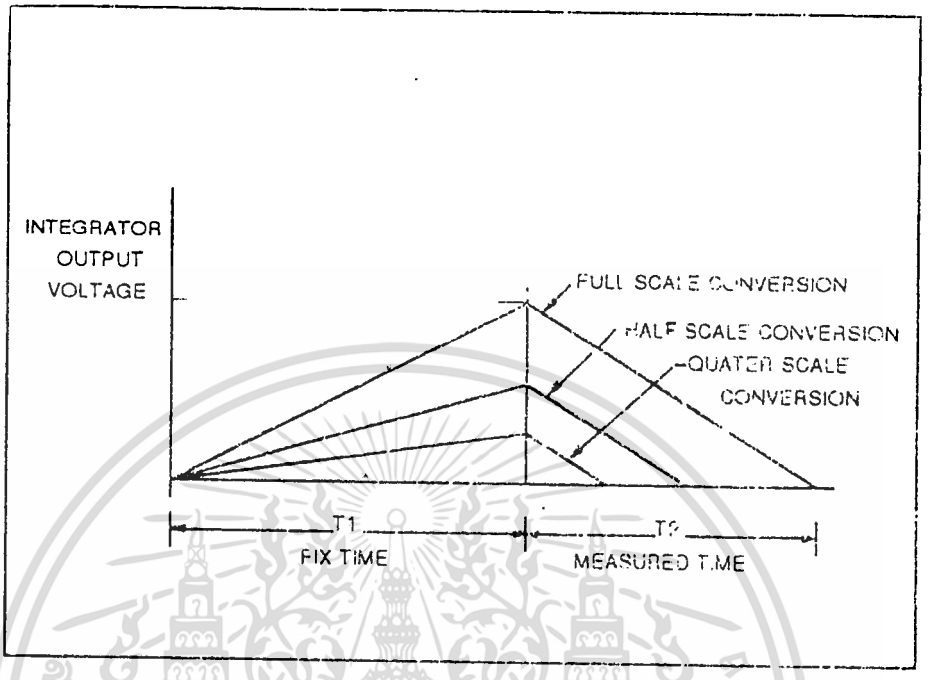
รูปที่ 4.15 วงจร Ramp voltage generator อย่างง่าย และลักษณะของเอ้าท์พุท

2). Dual Slope Converter

ADC แบบ dual slope ได้รับการพัฒนาขึ้นมาเพื่อแก้ไขจุดบกพร่องของ single slope ADC การจัดวงจรแสดงในรูปที่ 4.16 ในแต่ละวัฏจักรของการทำงานของวงจร จะมีสองช่วง คือ T_1 และ T_2 ในเวลา T_1 จะเป็นช่วงเวลาที่ได้รับการออกแบบให้มีค่าแน่นอนคงที่ ในช่วงเวลานี้สัญญาณอินพุทจะต่อเข้ากับอินทิเกรเตอร์ผ่านสวิทช์ S ซึ่งทำให้เอ้าท์พุทที่ถูกอินทิเกรท V_{int} เป็นรูปสัญญาณ ramp ที่ขนาดเพิ่มขึ้นทางบวกและสโลปขึ้นอยู่กับขนาดของ V_{in} จนกระทั่ง V_{int} ถึงค่าค่าหนึ่งเมื่อสิ้นสุด T_1



รูปที่ 4.16 บล็อกไดอะแกรมของ Dual Slope ADC



รูปที่ 4.17 การทำงานของ Dual slope ADC

ในช่วงเวลา T_2 อินพุทจะถูกตัดออกจากอินทิเกรเตอร์และต่อกับแรงดันอ้างอิงซึ่งมีค่าเป็นลบเข้ากับอินพุทของอินทิเกรเตอร์ โดยการควบคุมทางลอจิกในลักษณะเช่นนี้จะทำให้ V_{int} ลดลงด้วยสโลปคงที่จากการคายประจุผ่านลง V_{ref} เมื่อเริ่มต้นเวลา T_2 เคาท์เตอร์จะรีเซ็ทและเริ่มนับ จนเมื่อ V_{int} มีค่าลดลงถึงศูนย์ คอมพาเรเตอร์จะเปลี่ยนสถานะไปบอกส่วนควบคุมลอจิกให้หยุดนับ และเอ้าท์พุทของเคาน์เตอร์จะถูกแปลงเป็นรหัสดิจิตอลความสัมพันธ์ระหว่างช่วงเวลากับแรงดันอินพุทจะเป็นไปตามสมการ

$$T2 = T1 \frac{V_{in}}{V_{ref}}$$

ดังนั้นรหัสดิจิตอลที่แสดงค่า T_2 จะแสดงค่าอัตราส่วนของแรงดันอินพุทต่อแรงดันอ้างอิงด้วย

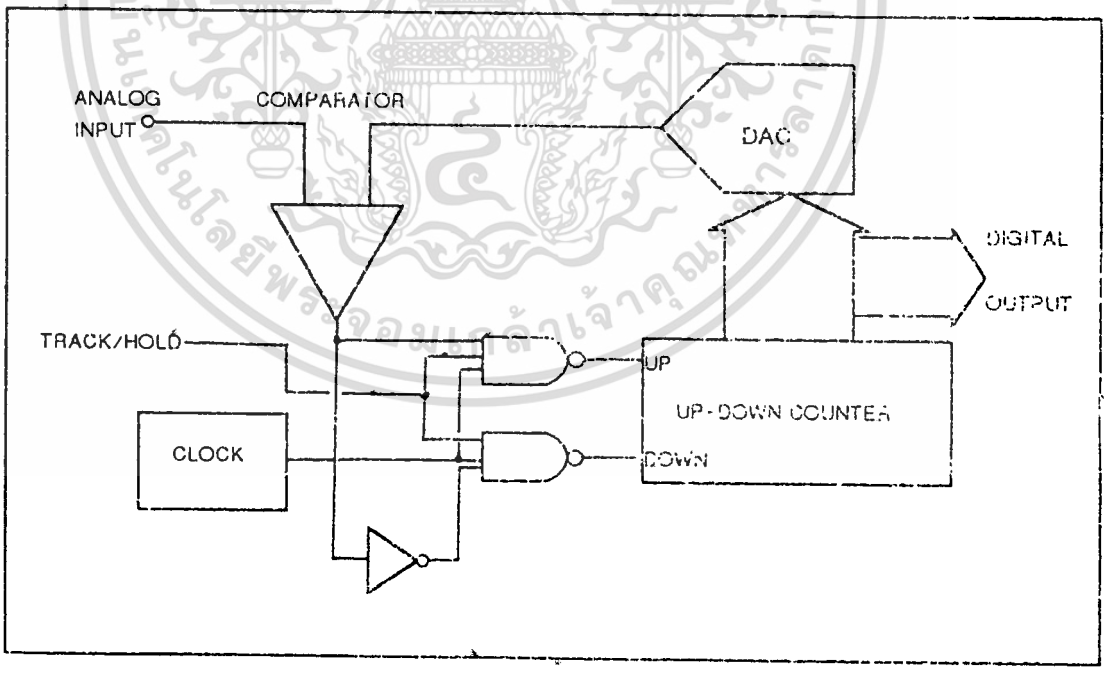
คุณลักษณะสำคัญของ dual slope มีหลายประการคือประการแรกความแม่นยำของมันไม่ขึ้นอยู่กับเสถียรภาพของสัญญาณ clock และตัวเก็บประจุ แต่จะขึ้นอยู่กับค่าความเที่ยงตรง

ของแรงดันอ้างอิงและความเป็นเชิงเส้นของอินทิเกรเตอร์ ประการที่สองการจำกัดสัญญาณรบกวนด้วยตัวเองของวงจรสามารถกระทำได้ ถ้าเซ็ทให้ T_1 มีขนาดเท่ากับคาบเวลาของสัญญาณรบกวนเช่น ในการจำกัดสัญญาณ 50 เฮิร์ท T_1 จะให้มีค่า 20ms

ส่วนข้อเสียที่สำคัญของ ADC นี้คือความเร็วในการ conversion ค่อนข้างต่ำจึงมักนิยมใช้กับเครื่องมือวัดที่ไม่ต้องการความเร็วเช่น ดิจิตอลมิเตอร์เป็นต้น

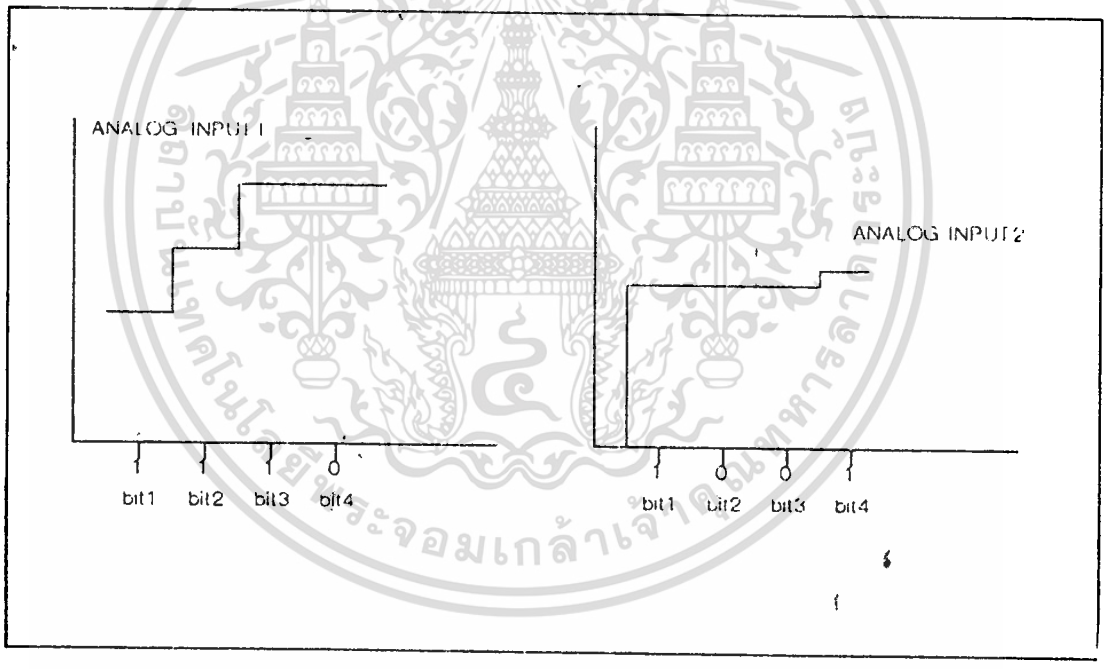
Sucessive Appoximation ADC

วงจร ADC ชนิดนี้ได้รับความนิยมในงานประยุกต์ที่ต้องการความเร็วปานกลางและค่อนข้างสูง การจัดวงจรจะคล้ายกันกับแบบเคาน์เตอร์ ที่ทำงานในลักษณะการป้อนกลับ ซึ่งบล็อกไดอะแกรมในรูปที่ 4.18 แสดงฟังชั่นต่างๆ ใน ADC ชนิดนี้ คอมพาเรเตอร์จะคอยเปรียบเทียบเอ้าท์พุทจาก DAC กับอนาลอกอินพุท V_{in} เอ้าท์พุทจะไปควบคุม Sucessive Appoximation register (SAR) ซึ่งเป็น ไอซี MSI(Midium Scale Integrated - circuit) ที่ได้รับการออกแบบเป็นพิเศษเพื่อทำหน้าที่นี้เฉพาะ



รูปที่ 4.18 บล็อกไดอะแกรมของ Sucessive Appoximation ADC

ในรูปที่ 4.19 แสดงไทมิ่งไดอะแกรมของ ADC ที่มีระดับอนาลอก 1 และ 2 ที่ระดับ 1 เมื่อ clock เข้าไป 1 ลูก จะทำให้ MSB(most significant bit) (บิท 4)เป็น 1 ทุกบิทอื่นยังคงเป็นศูนย์ DAC จะเปลี่ยนเอ้าท์พุทของ SAR เป็นอนาลอกเปรียบเทียบสัญญาณอนาลอกอินพุท ถ้าผลการเปรียบเทียบที่คอมพาเรเตอร์บอกว่าน้อยกว่าอินพุทก็ให้คงบิทนั้นเป็น 1 ไว้ แต่ถ้ามากกว่าจะให้บิทนั้นเป็น 0 จากนั้นทำการทดสอบบิทถัดไปโดยทำให้เป็น 1 หากผลรวมของสองบิทหรือบิทหลังมากกว่าก็ทำให้บิทนั้นเป็น 0 แต่ถ้าน้อยกว่าให้คง 1 ไว้ แล้วทดสอบบิทถัดไปตามกรรมวิธีดังกล่าวจนครบทกบิทหรือจนกว่าเอ้าท์พุทจะต่างจาก V_{in} ไม่เกิน 1 LSB ในตัวอย่างแสดงการทำงานเมื่อ V_{in} ลดต่ำลงมาอีกระดับหนึ่งด้วยเช่นกัน



รูปที่ 4.19 timing Diagram ของ SAR

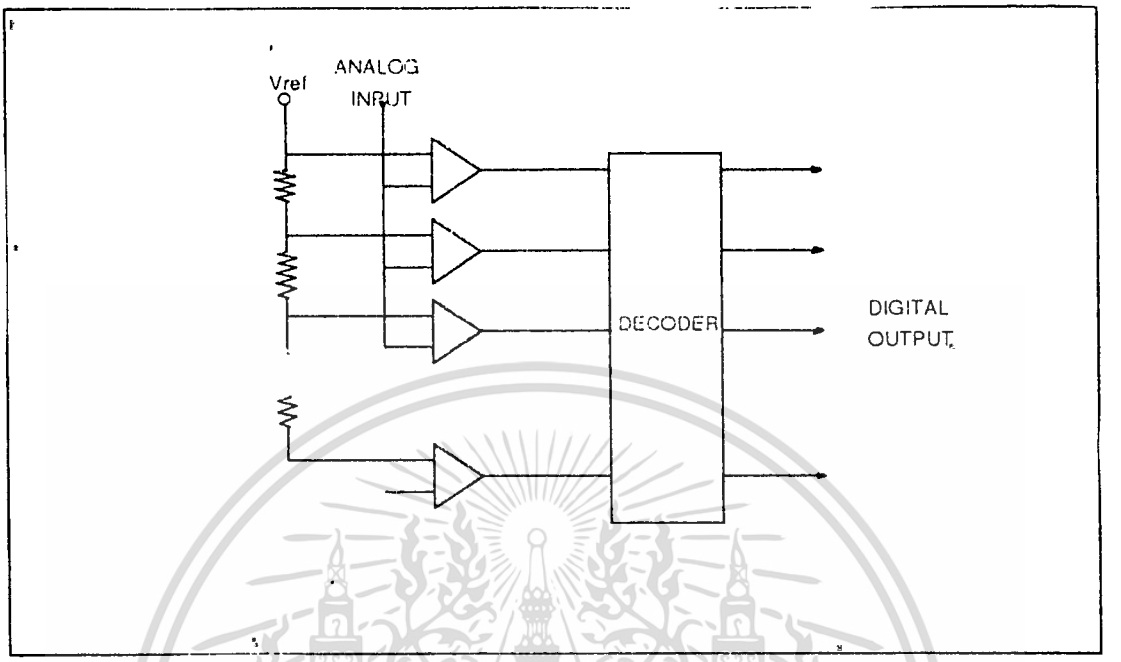
มีข้อจำกัดประการหนึ่งสำหรับการ conversion คือสัญญาณอนาลอกอินพุท จะต้องคงที่ในช่วงเวลาที่ทำการเปลี่ยนแปลงสัญญาณโดยเปลี่ยนได้ไม่เกิน 1/2 LBS ในช่วงสุดท้ายของการเปลี่ยนสัญญาณดิจิตอลเอ้าท์พุทจะออกมาขนานกันทุกบิท แต่บางแบบจะให้เอ้าท์พุทออกมาในลักษณะอนุกรม

วงจร ADC แบบนี้สามารถทำงานได้สองโหมด คือโหมดที่ทำงานโดยอิสระ(Free run) และโหมดที่รอคำสั่ง start conversion จากภายนอกเวลาที่ใช้ในการเปลี่ยนสัญญาณ(n+1) ลูกของพัลส์ clock ลูกแรกจะใช้ในการรีเซ็ทรีจิสเตอร์ภายใน

สุดท้าย คุณภาพของระบบจะขึ้นอยู่กับคุณภาพของ DAC ในระบบเป็นอย่างยิ่ง

Parallel (Flash) ADC

สำหรับการแปลงสัญญาณที่ต้องการความเร็วสูงมากๆ เช่นการแปลงสัญญาณภาพโทรทัศน์ เรดาห์ จำเป็นต้องใช้ ADC แบบพิเศษ ที่เรียกว่า Parallel ADC ซึ่งแสดงบล็อกไดอะแกรมดังรูปที่ 4.20 หลักการทำงานคือ จะใช้คอมพาเรเตอร์ ทำการเปรียบเทียบสัญญาณอนาลอกอินพุทกับแรงดันอ้างอิงที่แบ่งแรงดันให้สอดคล้องกับรหัสดิจิตอล โดยใช้ตัวต้านทานแล้วแปลงเอ้าพุทจากคอมพาเรเตอร์ให้ตรงกับรหัสดิจิตอล ซึ่งจะเห็นว่าอุปสรรคทางด้านความเร็วจะถูกจำกัดเพียง Propagation time ของ คอมพาเรเตอร์เท่านั้น แต่อุปสรรคที่สำคัญต่อการพัฒนาวงจรชนิดนี้บนชิปไอซี คือ วงจรนี้ต้องการคอมพาเรเตอร์ถึง 2^n-1 ตัว สำหรับ ADC 1 ตัว แต่ก็ได้ ADC ชนิดที่ทำงานได้รวดเร็วที่สุดเช่นเดียวกัน



รูปที่ 4.20 บล็อกไดอะแกรมของ Flash ADC

ADC Chip

จากความก้าวหน้าทางเทคโนโลยีสารกึ่งตัวนำและความต้องการระบบ Data acquistion ที่มีขนาดเล็กและกินกำลังงานต่ำจึงมีการนำเอาองค์ประกอบของระบบหลายๆ ส่วนมาลงไว้บนชิปเดียว การเลือกใช้ชิป ADC สำหรับงานประยุกต์ทั่วๆ ไปจึงมักแค่เพียงพิจารณาถึง resolution ว่าต้องการกี่บิท ความเร็วของการแปลง conversion time ขนาดแรงงานอินพุทและไฟเลี้ยง ในการออแบบระบบที่ใช้ ชิป ADC จะขึ้นอยู่การเลือกโหมดและใช้ Timming diagram ที่ผู้ผลิตเสนอให้มาในคู่มือการใช้งาน

ตัวอย่างชิป ADC ที่นิยมนำมาใช้งาน ได้แก่ ADC0804 [2] ซึ่งเป็น CMOS 8 บิท Sucessive appoximation ADC ราคาประหยัดที่มีวงจรกำเนิดสัญญาณนาฬิกาภายใน (internal clock)ปรับความถี่ได้ด้วยการเปลี่ยนค่า RC ภายนอกมีวงจร logic control ที่ทำให้สามารถต่อเชื่อมโยง (interface) กับระบบไมโครโปรเซสเซอร์ 8 บิท ได้โดยง่าย ในรูปที่ 4.20 แสดงรายละเอียดของ ADC 0804 timming diagrams และบล็อกไดอะแกรม ส่วนรายละเอียดอื่นๆ และการประยุกต์อยู่ในภาคผนวกของโครงงานนี้

บทที่ 5

การรับ-ส่ง ข้อมูลแบบอนุกรม แบบ RS 232C

ในการศึกษาโครงงานระบบแสดงคลื่นไฟฟ้าหัวใจบนไมโครคอมพิวเตอร์(Micro Computer Based EKG Monitor) ให้ใช้การสื่อสารรับ-ส่ง ข้อมูลแบบอนุกรมซึ่งมีรูปแบบดังต่อไปนี้

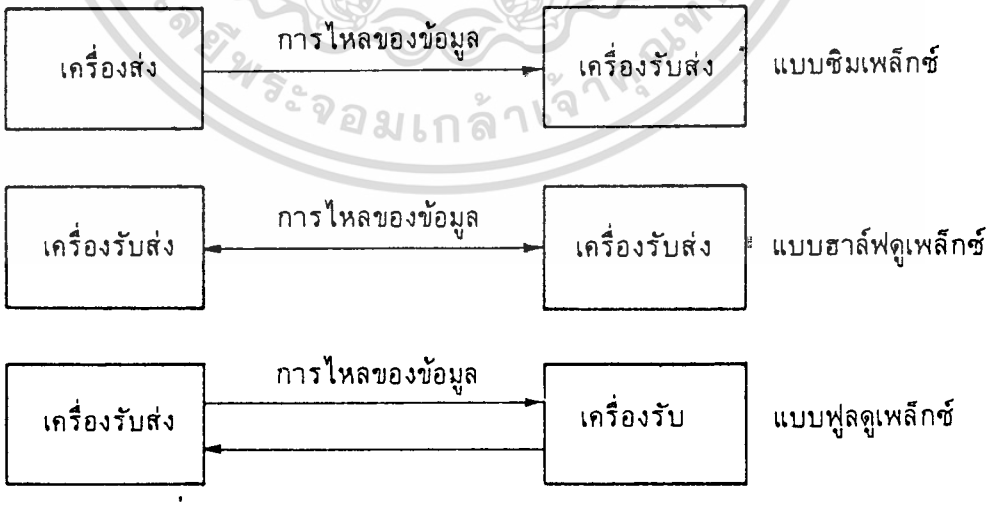
รูปแบบของการติดต่อสื่อสารระบบอนุกรม

การติดต่อแบบอนุกรมอาจจะแบ่งตามรูปลักษณะได้ 3 แบบ ตามรูปที่ 5.1

1. **แบบซิมเพล็กซ์** (simplex) ข้อมูลส่งได้ในทางเดียวเท่านั้น บางครั้งก็เรียกว่าการส่งทิศทางเดียว (Unidirectional data bus)

2. **แบบฮาล์ฟดูเพล็กซ์** (half duplex) ข้อมูลสามารถส่งได้ทั้งสองสถานี แต่จะต้องผลัดกันส่งและผลัดกันรับ จะส่งและรับพร้อมกันไม่ได้

3. **แบบฟูลดูเพล็กซ์** (full duplex) ทั้งสองสถานีสามารถรับและส่งได้ในเวลาเดียวกัน



รูปที่ 5.1 รูปแบบของการติดต่อสื่อสารข้อมูลแบบอนุกรม

การส่งแบบฟูลดูเพล็กซ์และฮาล์ฟดูเพล็กซ์ ไม่ขึ้นอยู่กับจำนวนของสายในการติดต่อ บางครั้งคำว่า ทูไวร์ (two wire) หรือสองเส้น และโฟร์ไวร์ (four wire) หรือ 4 เส้น ใช้ในการบรรยายถึงลักษณะการสื่อสารข้อมูลซึ่งอาจจะทำให้เข้าใจและฮาล์ฟดูเพล็กซ์ สายโทรศัพท์ทั่วไปเป็นแบบ 2 เส้น ส่วนในสายที่เป็นแบบเช่า (lease line) นั้นส่วนมากจะเป็น 4 เส้น

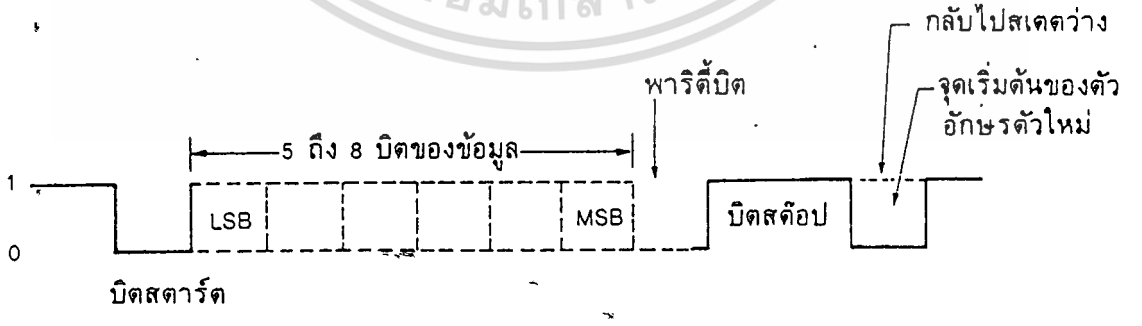
ความเร็วในการถ่ายโอนข้อมูลแบบอนุกรม

ความเร็วของการถ่ายโอนข้อมูลแบบอนุกรม หน่วยวัดเป็นบิทต่อวินาที (bps) หน่วยที่บรรยายถึงการเปลี่ยนแปลงของสัญญาณใน 1 วินาที เรียกว่าบอดเรต (baud rate) หรืออัตราบอด หลายคนยังเข้าใจสับสนระหว่างอัตราบอดและอัตราบิท (bit rate) การเปลี่ยนแปลงของสัญญาณ 1 ครั้ง อาจจะแสดงถึงการส่งข้อมูลแบบอนุกรมมากกว่า 1 บิท ถ้าเขียนในรูปของสมการทางคณิตศาสตร์เราก็จะได้

อัตราบิท (bit rate) = อัตราบอด (baud rate) x (บิทใน 1 บอด)

การสื่อสารแบบอะซิงโครนัส

การส่งแบบอะซิงโครนัสนี้ พัฒนามาจากการส่งโทรพิมพ์ในสมัยก่อน ลักษณะของสัญญาณแสดงไว้ในรูปที่ 5.2 เพื่อเพิ่มกลไกในการรับส่งอย่างถูกต้อง สัญญาณอะซิงโครนัส จะประกอบด้วยบิทเริ่มต้นหรือบิทสตาร์ท (start) และบิทสิ้นสุดหรือบิทสต๊อป (stop bit)



รูปที่ 5.2 ฟอร์แมตการสื่อสารแบบอะซิงโครนัส

ขณะที่สถานะของการส่งเป็นแบบว่าง (Idle) คือ ยังไม่มีสัญญาณส่งออกมา จะมีสัญญาณหรือมีแรงดัน (หรือกระแส) ตลอดเวลา เพื่อความแน่ใจว่าฝ่ายรับยังติดต่ออยู่กับฝ่ายส่ง เมื่อเริ่มจะส่งข้อมูล สัญญาณของอะซิงโครนัสจะเป็น 0 หนึ่งช่วงสัญญาณนาฬิกา บิทนี้เรียกว่า สาร์ทบิท ตามหลังของสตาร์ทบิทก็จะเป็นข้อมูลสำหรับ 1 ตัวอักษร ซึ่งอาจจะมีขนาดตั้งแต่ 5 บิท จนถึง 8 บิท โดยบิทที่มีค่าน้อยที่สุด (LSB) จะถูกส่งออกมาก่อนไล่ไปจนถึงบิตที่มีค่ามากที่สุด (MSR) การเข้ารหัสอักขระนี้ส่วนมากจะนิยมใช้รหัส ASCII แรกเริ่มที่เดียวในงานของโทรพิมพ์เขาใช้รหัส Baudot ซึ่งใช้ 5 บิท ในการแทนอักขระ 1 ตัว ตามหลังข้อมูลก็จะเป็นพาริตี้บิท ซึ่งอาจจะใช้หรือไม่ใช้ก็ได้ พาริตี้บิตทำหน้าที่เป็น ตัวตรวจสอบความถูกต้องของสัญญาณที่ได้รับ พาริตี้บิทอาจจะเป็นแบบคู่ (Even) หรือแบบคี่ (Odd) หมายความว่าถ้าหากเป็นพาริตี้คู่ จำนวนบิทที่เป็น 1 ในช่วงบิทข้อมูลกับบิทพาริตี้รวมแล้วจะต้องเป็นจำนวนคู่ ผู้ส่งจะต้องทำหน้าที่ตรวจสอบข้อมูลแล้วใส่พาริตี้บิทเอง ฝ่ายรับเมื่อรับแล้วก็จะต้องตรวจสอบดูว่าเป็นจริงดังสถานการณ์ที่ตั้งเอาไว้หรือไม่ หากผิดพลาดก็หมายความว่าสัญญาณที่รับผิดพลาดไปจากสถานีส่งส่งออกมา ทั้งนี้ทั้งนั้นจะต้องผิดเป็นจำนวนคี่เท่านั้นคือผิดไป 1 บิท 3 บิท หรือ 5 บิท พร้อมกันจึงจะตรวจสอบได้ว่าผิด มองเห็นง่ายๆ ว่าถ้าผิดเป็นจำนวนคู่ ผลรวมของจำนวนหนึ่งก็ยังเป็นคู่อยู่ดีทั้งนี้ทั้งนั้นไม่ได้หมายความว่าพาริตี้คี่ (Odd Parity) จะตรวจสอบการผิดพลาดเป็นจำนวนคี่ความจริงแล้วตรวจสอบความผิดพลาดได้เหมือนกับพาริตี้คู่ (Even Parity) แต่แทนที่จะตรวจสอบดูว่าสัญญาณที่รับเข้ามีจำนวนคู่ ก็ตรวจสอบดูว่ามีจำนวนคี่หรือเปล่าอย่างไรก็ตามโอกาสที่จะผิดพลาด 2 บิทพร้อมกันมีน้อยมาก

ย้อนกลับมากดูสัญญาณอะซิงโครนัสใหม่ หลังจากบิทพาริตี้แล้วก็จะต้องมีสต๊อปบิท ซึ่งเป็น 1 ความกว้างของสต๊อปบิทอาจจะเป็น 1,1.5 หรือ 2 พัลส์ของสัญญาณนาฬิกา แล้วแต่ผู้รับและผู้ส่งจะตกลงใช้กันเอง การเริ่มใช้พอร์ตอนุกรม (ทางออกอนุกรม) จึงจำเป็นจะต้องตั้งค่าต่าง ๆ สำหรับเป็นการส่งแบบอนุกรมอันได้แก่

1. ความเร็วในการส่ง
2. ความยาวรหัส 1 อักขระ
3. บิทตรวจสอบ
4. จำนวนสต๊อปบิท

ในการส่งโทรพิมพ์ หรือโทรเลขเมื่อก่อนนี้ใช้ความเร็ว แค่ 70 บอด และ 110 บอด สำหรับคอมพิวเตอร์ความเร็วในการส่งมีให้เลือกตั้งแต่ 110,200,300,1200,2400,4800, 9600 บอด และสูงไปกว่านั้น เนื่องจากมี IC หลายเบอร์ทำหน้าที่รับส่งแบบอะซิงโครนัสให้ใช้ การส่งแบบอนุกรมจึงสะดวกสบายสำหรับคนออกแบบพอร์ตอนุกรม สำหรับมาตรฐานพอร์ตอนุกรมจะกล่าวถึงในบทที่ 3

จะเห็นว่ากลไกในการซิงโครนัสของการสื่อสารอะซิงโครนัส มีลักษณะเป็นไปทีละตัวอักขระ จำนวนพัลส์ของสัญญาณที่ส่งออกยังมีบางส่วนใช้ในการควบคุมการส่งอยู่อันได้แก่ บิทสตาร์ท บิทสต๊อบ และบิทพาริตี้ ทำให้ความเร็วการส่งอักขระต่อวินาทีน้อยลงไป การส่งสัญญาณด้วยความเร็ว 300 บอด สำหรับการเข้ารหัส 7 บิท ไม่ได้หมายความว่าส่งได้ 300 หารด้วย 7 อักขระต่อวินาที

ฮาร์แวร์ที่เกี่ยวข้องกับการสื่อสารข้อมูล

พอร์ต RS 232C

โดยปกติไมโครคอมพิวเตอร์จะมีพอร์ตที่เป็นแบบอนุกรม เรียกชื่อกันว่า RS 232C อยู่ในตัวเองอยู่แล้ว หลายเครื่องไม่มีมากับเครื่อง อย่างเช่น IBM PC จำเป็นจะต้องมีการ์ดที่เรียกว่าอะซิงโครนัสอะแดปเตอร์ (Asyncchronous Communication Adapter) มาเสียบใส่

พอร์ต RS 232 นี้ทำหน้าที่รับและส่งข้อมูลในแบบอนุกรมเรียกว่า Universal Asynchronous Adapter เหตุที่มีชื่อเรียกว่า RS 232C ก็เนื่องจากสมาคมผู้ผลิตอุปกรณ์อิเล็กทรอนิกส์ของอเมริกาหรือ EIA ได้กำหนดมาตรฐานของอุปกรณ์การสื่อสารแบบอนุกรมเอาไว้ภายใต้ชื่อว่า RS 232C ความจริงมาตรฐานของการส่งข้อมูลแบบอนุกรมมีหลายมาตรฐาน แต่ที่นิยมกันมากที่สุดสำหรับไมโครคอมพิวเตอร์ก็คือ RS 232C

หน้าที่สำคัญของการสื่อสารแบบอะซิงโครนัสก็คือ

รับสัญญาณ

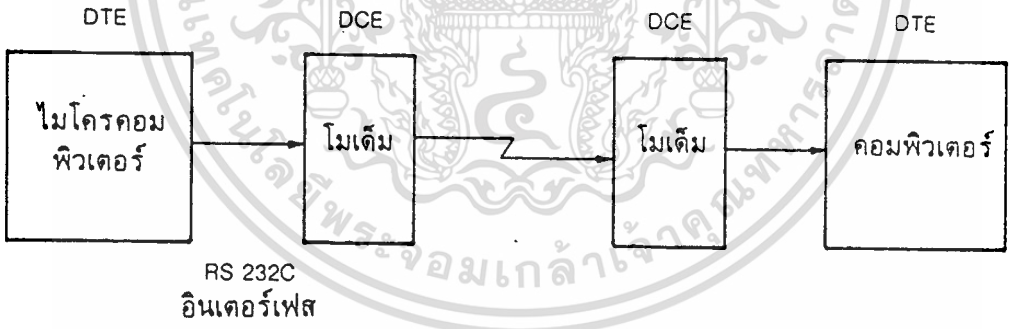
1. เปลี่ยนสัญญาณเข้ามาแบบอนุกรมให้เป็นแบบขนาน
2. ตรวจสอบความผิดพลาดของสัญญาณที่รับ
3. ตัดสต๊อบบิทและพาริตี้บิทออก
4. ส่งสัญญาณให้ซีพียูรู้ว่ารับสัญญาณไว้แล้ว

ส่งสัญญาณ

1. เปลี่ยนสัญญาณแบบขนานจากซีพียูค่อยทยอยส่งออกเป็นแบบอนุกรม
2. เพิ่มสต๊อบบิทและพาริตี้
3. เพิ่มสัญญาณควบคุมโมเดมที่ต่อเชื่อม (ถ้ามี)

มาตรฐาน RS 232C

มาตรฐาน RS 232C ได้จัดพิมพ์ขึ้นเมื่อปี ค.ศ. 1969 โดยสมาคมผู้ผลิตอุปกรณ์อิเล็กทรอนิกส์แห่งสหรัฐอเมริกา RS ย่อมาจาก Recommened Standard ส่วน 232 เป็นหมายเลขบ่งบอกของมาตรฐานตัวนี้ C เป็นหมายเลขของฉบับท้ายสุดของมาตรฐานตัวนี้ จุดประสงค์ของมาตรฐานตัวนี้เพื่อบรรยายคุณลักษณะของการเชื่อมต่ออุปกรณ์รับส่งข้อมูลปลายทาง (Data Terminal Equipment DTE) กับอุปกรณ์สื่อสารข้อมูล (Data Communication Equipment DCE) สำหรับผู้ใช้ไมโครคอมพิวเตอร์ DTE ก็หมายถึงตัวไมโครคอมพิวเตอร์ และ DCE ก็หมายถึงโมเดม อุปกรณ์อื่นๆ เช่น เครื่องพิมพ์ที่รับสัญญาณแบบอนุกรม อาจจะเป็นได้ทั้ง DTE



รูปที่ 5.3 การใช้ RS 232C เชื่อมต่ออุปกรณ์

และ DCE ขึ้นอยู่กับผู้ผลิต ข้อแตกต่างของ DTE และ DCE จะเห็นได้จาก รูหที่ 5.3 จากรูปนี้เราจะเห็นได้ว่า RS 232C มีส่วนสำคัญอย่างใหญ่หลวงสำหรับ การสื่อสารข้อมูลระหว่างไมโครคอมพิวเตอร์

ความจริงอีกประการหนึ่งของ RS 232C ก็คือ ความเร็วและระยะทางของการเชื่อมต่อ RS 232C สามารถเชื่อมต่อการถ่ายโอนข้อมูลได้จาก 0-20,000 บิท ต่อวินาที ซึ่ง

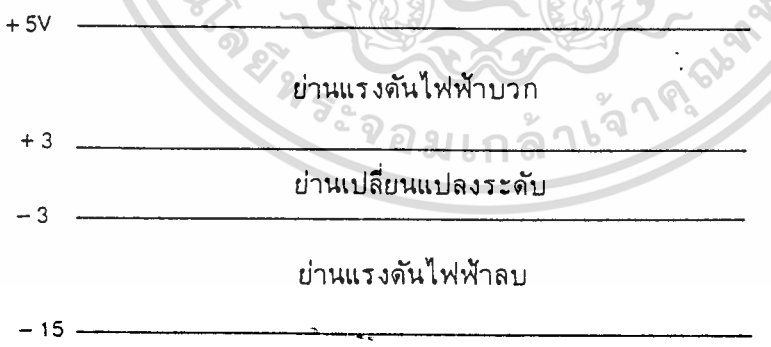
เพียงพอสำหรับไมโครคอมพิวเตอร์ที่มีขนาดอัตราบอด 110 ถึง 9600 บอด ความยาวของสายเชื่อมต่อสัญญาณตามมาตรฐานของ RS 232C จำกัดอยู่แค่ 50 ฟุต ซึ่งเพียงพอสำหรับการสื่อสารไมโครคอมพิวเตอร์กับอุปกรณ์รอบนอก

ลักษณะของสัญญาณ RS 232C

เพื่อเป็นหลักประกันว่าข้อมูลถูกส่งออกไปอย่างถูกต้อง และอุปกรณ์ถูกควบคุมอย่างถูกต้อง จำเป็นจะต้องมีข้อตกลงกันในเรื่องของสัญญาณที่ใช้ มาตรฐาน RS 232C กำหนดย่านของแรงดันไฟฟ้าในสัญญาณเพื่อสนองจุดประสงค์ข้างบน ดังแสดงในตารางที่ 5.1 และรูป 5.4

ตารางที่ 5.1

มาตรฐานของการใช้แรงดันไฟฟ้า			
แรงดันไฟฟ้า	สถานภาพลอจิก	สถานภาพของสัญญาณ	ฟังก์ชันในการควบคุม
บวก	0	สเปซ	ออน
ลบ	2	มาร์ค	ออฟ

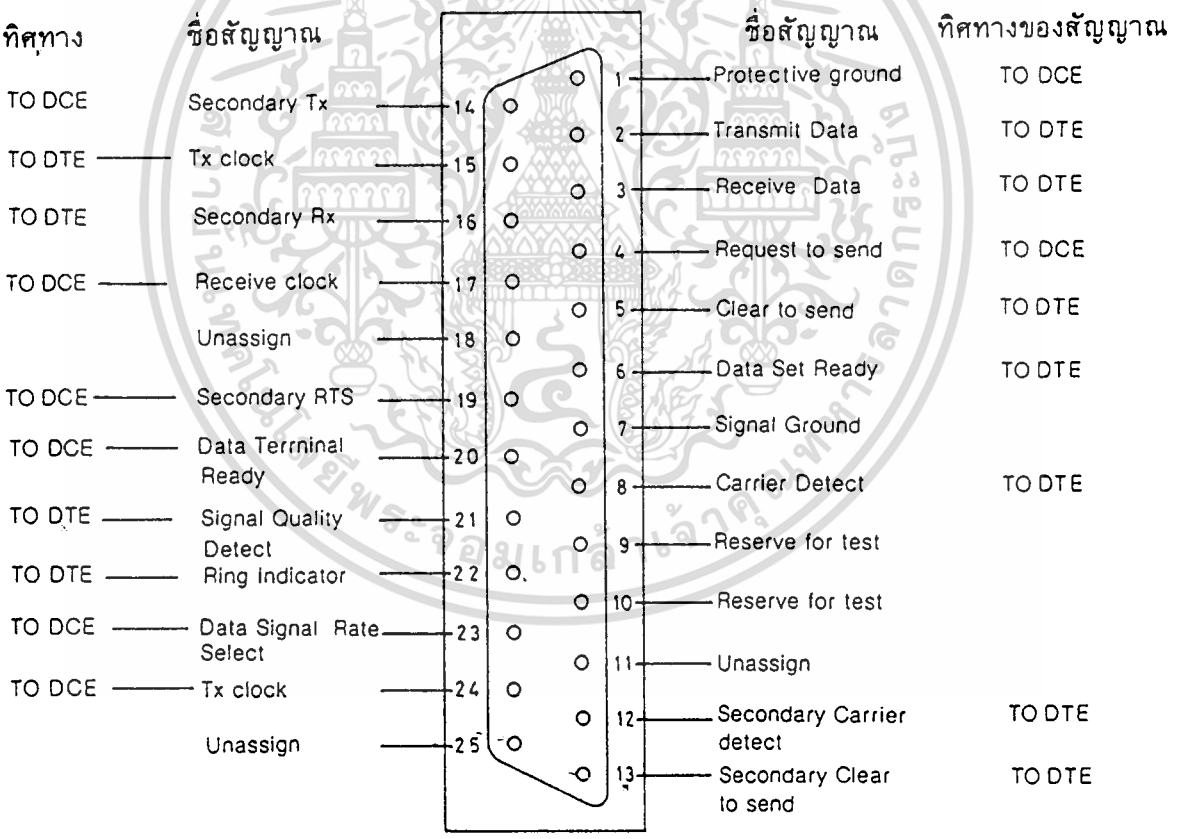


รูปที่ 5.4 ย่านของแรงดันไฟฟ้าที่ใช้ในสัญญาณ RS 232C

สำหรับไมโครคอมพิวเตอร์บางเครื่อง ใช้แต่สัญญาณลอจิกออกมาเป็นสัญญาณของ RS - 232C เลย อย่างเช่น อะซิงโครนัสอะแดปเตอร์ของ IBM PC ในกรณีเช่นนี้ระยะทางของสายที่เชื่อมต่ออาจจะไปได้สั้นกว่า 50 ฟุต ดังที่กล่าวเอาไว้เนื่องจากระดับของกราวด์เปลี่ยนแปลงไป อันเนื่องจากการสูญเสียไปในความต้านทานของสาย ผู้ที่เคยใช้ IBM PC อาจจะเคยประสบกับปัญหานี้มาแล้วว่า เอ๊ะทำไมต่อสัญญาณ RS 232C เกินกว่า 10 ฟุต แล้วยังใช้งานไม่ได้ แต่อย่างไรก็ตาม RS 232C ของ IBM PC ยังมีโอกาสให้เลือกใช้ 20 มิลลิแอมแปร์ กระแสวนกลับแทนแรงดันไฟฟ้า (20 mA loop จะกล่าวในหัวข้อถัดไป)

การกำหนดจุดข้อต่อของ RS 232C

ในทางฟิสิกส์แล้ว มาตรฐานของ RS 232C กำหนดข้อต่อแบบ DB-25 แต่ละขาของข้อต่อกำหนดไว้ดังรูปที่ 5.5 อย่างไรก็ตามผู้ผลิตไมโครคอมพิวเตอร์



รูปที่ 5.5 การกำหนดของข้อต่อ RS 232

อาจจะใช้ข้อต่อชนิดอื่น ที่นอกเหนือไปจาก DB-25 ยกตัวอย่าง เช่น Fujitsu F-8 IBM AT, IBM Jr เป็นต้น ตัวเมียของข้อต่อควรจะอยู่ที่ตัวโมเด็ม ขณะที่ตัวผู้ควรจะอยู่ที่ asynchronous communication adapter หรือที่ตัวไมโครคอมพิวเตอร์เอง อย่างไรก็ตามผู้ผลิตหลายรายไม่ได้ทำตามกฎเกณฑ์ที่ว่านี้

สัญญาณต่างๆ ถูกมอบหมายให้ทำหน้าที่ดังนี้

Transmit Data (TD ขาที่ 2)

เป็นสัญญาณที่ส่งออกจาก DTE (หรือตัวไมโครคอมพิวเตอร์) ไปยังโมเด็ม หรือต่อเข้าโดยตรงกับไมโครคอมพิวเตอร์ตัวอื่น หรือเครื่องพิมพ์ เมื่อไม่มีสัญญาณส่งออกสถานภาพของลอจิกที่ขานี้จะมีค่าเท่ากับ "1" หรือเทียบเท่ากับสต๊อปบิท

Receive Data (RD ขาที่ 3)

เป็นทางของสัญญาณเข้าไปยัง DTE หรือไมโครคอมพิวเตอร์ เมื่อไม่มีสัญญาณรับเข้ามาขานี้จะมีสถานภาพทางลอจิก เป็น "1"

Request To Send (RTS ขาที่ 4)

ใช้สำหรับส่งสัญญาณไปยังโมเด็ม หรือเครื่องพิมพ์เป็นการเรียกร้องที่จะส่งสัญญาณมาทางขา 2 สัญญาณนี้ใช้คู่กับ CTS หรือ Clear to send อุปกรณ์รับหากได้รับสัญญาณ RTS จะตรวจสอบตัวเองว่าพร้อมจะรับสัญญาณได้หรือยัง หากพร้อมที่จะรับก็ส่งสัญญาณออกไปที่สาย CTS

Clear To Send (CTS ขาที่ 5)

ดังอธิบายไว้ใน RTS เมื่อสัญญาณนี้อยู่ในสถานะออฟ(negative voltage หรือลอจิก "1") หมายความว่า อุปกรณ์รับกำลังบอกว่าพร้อมที่จะรับข้อมูลแล้ว

Data Set Ready (DSR ขาที่ 6)

เมื่อมีสัญญาณสายนี้อยู่ในสถานะออน (หรือลอจิก 0) เป็นการบอกไมโครคอมพิวเตอร์หรือฝ่ายส่งว่า โมเด็มต่อเข้ากับสายโทรศัพท์เรียบร้อยแล้วและพร้อมที่จะส่งได้แล้ว โมเด็มที่มีการหมุนหมายเลขอัตโนมัติจะส่งสัญญาณสายนี้ไปบอกให้ คอมพิวเตอร์รู้ว่าต่อโทรศัพท์ได้สำเร็จแล้ว

Signal Ground (SG ขาที่ 7)

SG ทำหน้าที่เป็นระดับแรงดันอ้างอิงสำหรับทุกๆ สายของสัญญาณ จะมีแรงดันเป็น "0" เมื่อเทียบกับสัญญาณตัวอื่น

Carrier Detect (CD ขาที่ 8)

โมเด็มจะส่งสัญญาณที่อยู่ในสถานะออน (ลอจิก "0") ไปบอกไมโครคอมพิวเตอร์ เมื่อได้รับสัญญาณจากโมเด็มของอีกฝ่ายหนึ่ง สัญญาณนี้จะนำไปจุด LED บอกว่าได้รับสัญญาณจากโมเด็มอีกฝ่ายหนึ่งแล้ว ไป LED จะอยู่บนหน้าปัดของโมเด็มเอง

Data Terminal Ready (DTR ขาที่ 20)

คอมพิวเตอร์เปิดสัญญาณสายนี้ให้ออน (ลอจิก "0") เมื่อพร้อมที่จะติดต่อกับโมเด็ม โมเด็มส่วนมากจะไม่รายงานสถานภาพของตัวเอง (CD,USR และ CTS) ให้คอมพิวเตอร์รู้ หากคอมพิวเตอร์ไม่เปิดสัญญาณ DTR

Ring Indicator (RI ขาที่ 22)

สัญญาณนี้ใช้ในโมเด็มที่เป็นระบบตอบได้อัตโนมัติ (Auto-answer) สัญญาณนี้จะออนเมื่อมีสัญญาณกระดิ่งมา และออฟระหว่างเสียงดังของกระดิ่ง

ท่านผู้อ่าน อาจสับสนระหว่างสถานภาพของลอจิกกับสถานภาพของสัญญาณโดยปกติเราจะคุ้นเคยอยู่กับความรู้สึกที่ว่า เมื่อแรงดันเป็นบวก หรือสัญญาณออนลอจิกน่าจะเป็น "1" สำหรับสัญญาณต่างๆ ที่กล่าวมานี้จะมีลักษณะตรงกันข้าม ทำไมเขากำหนดกฎเกณฑ์ออกมาอย่างนี้ ก็เพราะว่าแต่เดิมนั้น การติดต่อกันทางโทรเลขการทำงานของสัญญาณจะต้องครบวงจรทั้งฝ่ายส่งและฝ่ายรับ เมื่อลอจิกเป็น "0" หรือขณะที่ไม่มีอะไรส่งควรจะมีสัญญาณทางไฟฟ้าครบวงจรอยู่ตลอดเวลาก็โดยการให้ค่าแรงดันที่ฝ่ายส่ง ดังนั้นจึงถือกันว่าสัญญาณไฟบวกใช้เป็นลอจิก "0"

ตารางที่ 5.2 คุณลักษณะโดยย่อของสัญญาณ RS 232C

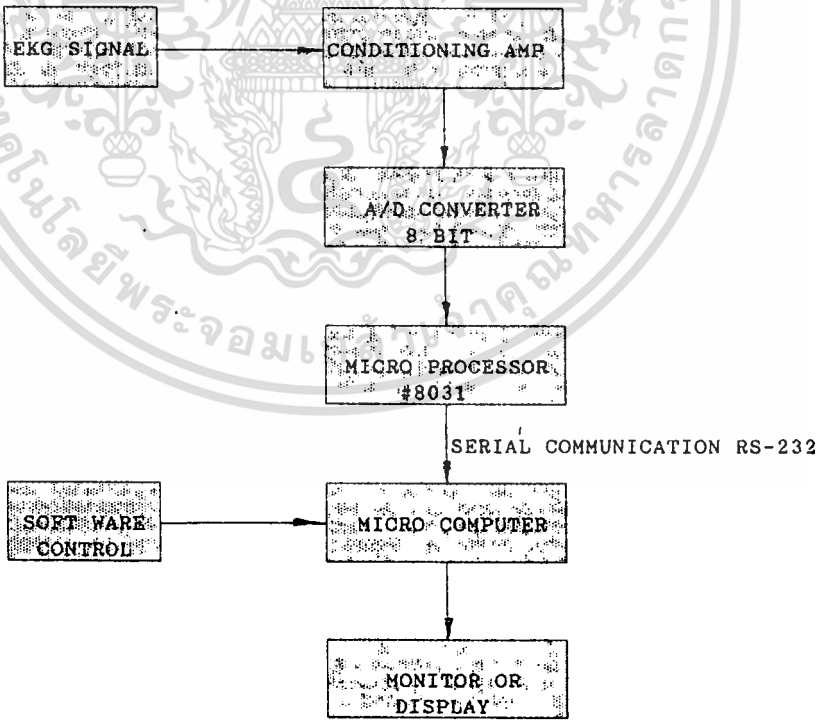
Driver output logic levels with	15V>O_h>5V
3k to 7k load	-5>O_l>-15 โวลต์
Driver output voltage when open	Vo <25 โวลต์
circuit	
Driver output impedance with Power off	R_o>300 Ohms
Output short circuit current	I_o < 0.5 A
Driver slew rate	dv/dt < 30 V/s
Receiver input impedance	7k >R_{in}>3k
Receiver input voltage	+15 compatible with
	driver
Receiver output with open	MARK
circuit input	
Receiver output with +3V input	SPACE
Receiver output with -3V input	MARK
+15	LOGIC 0 = SPACE =
+5	CONTROL ON
+5	Noise Margin
+3	
+3	Transition Region
-3	
-3	Noise Margin
-5	
-5	LOGIC 1 = MARK =
-15	CONTROL OFF

บทที่ 6

โครงงานระบบแสดงคลื่นไฟฟ้าหัวใจบนไมโครคอมพิวเตอร์

(MICRO COMPUTER BASED EKG MONITOR)

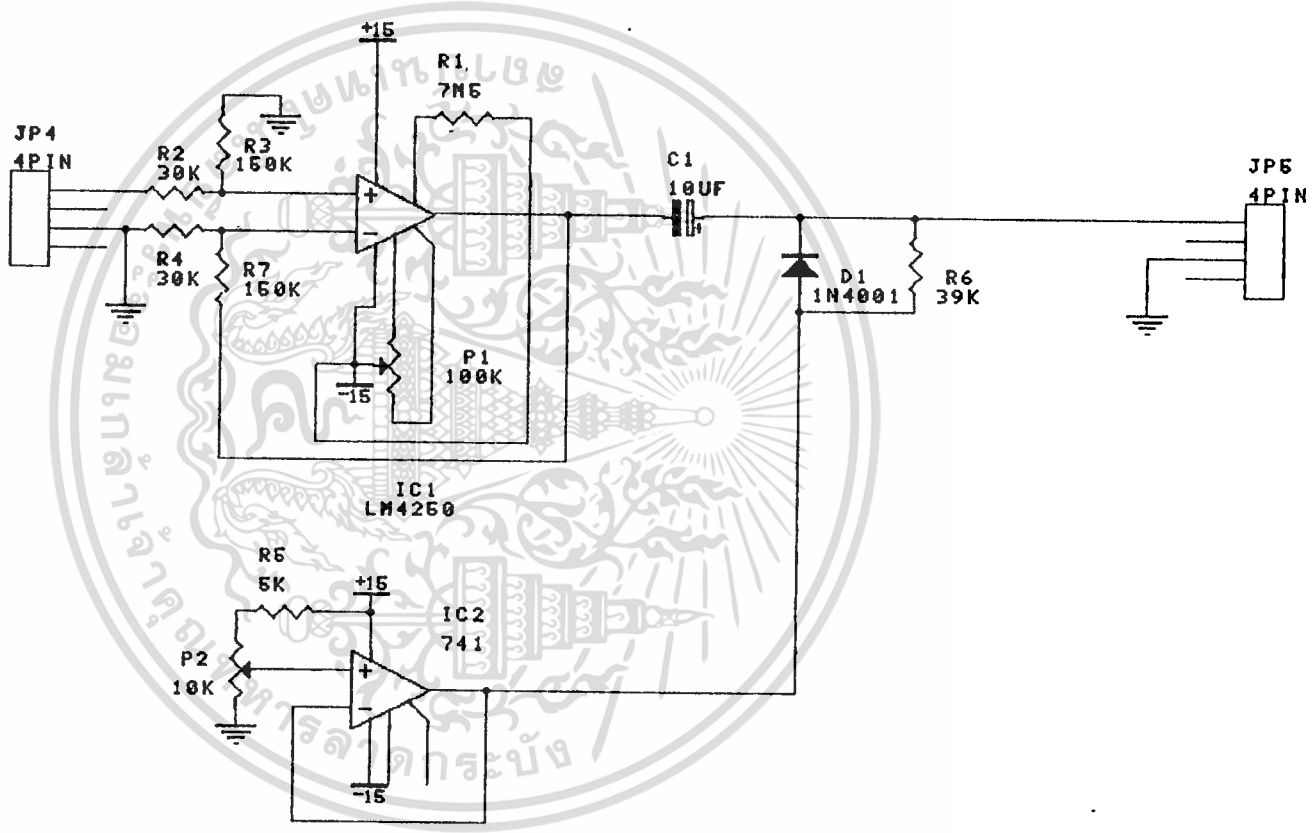
ในการศึกษาโครงงานระบบแสดงคลื่นไฟฟ้าหัวใจบนไมโครคอมพิวเตอร์นี้ มีหลักการทำงานของระบบตามบล็อกไดอะแกรม ดังรูปที่ 6.1 โดยสัญญาณคลื่นไฟฟ้าหัวใจก็ต้องการแสดงผลถูกป้อนเข้าที่ ภาคอินพุทของวงจร Condittioning Amplifier ซึ่งจะทำหน้าที่ขยายระดับสัญญาณคลื่นไฟฟ้าหัวใจ ให้มีขนาดใหญ่ที่เหมาะสมเพื่อที่จะป้อนให้กับภาคอินพุทของวงจรแปลงสัญญาณอะนาลอกเป็นดิจิตอล วงจรแปลงสัญญาณอนาลอกเป็นดิจิตอลจะให้เอ้าท์พุทซึ่งเป็นสัญญาณดิจิตอล 8 บิท ป้อนให้กับไมโครโปรเซสเซอร์ ซึ่งไมโครโปรเซสเซอร์ทำหน้าที่ส่งค่าสัญญาณดิจิตอลออกไปในรูปการสื่อสารแบบอนุกรม RS-232 ซึ่งไมโครโปรเซสเซอร์ใช้เบอร์ 8031 อยู่ในตระกูล MCS51 ข้อดีของไมโครโปรเซสเซอร์เบอร์นี้คือ มีพอร์ตอนุกรมภายในสามารถใช้งานได้ทันที ซึ่งต่างจาก Z-80 ซึ่งจะต้องใช้ IC CHIP เพิ่มเติมในส่วนของการสื่อสารข้อมูล เมื่อสัญญาณคลื่นไฟฟ้าถูกส่งไปยัง Micro Computer ซึ่งมีการเขียนโปรแกรมการรับค่าสัญญาณดิจิตอล นำไปแสดงผลบนจอของ Micro Computer ต่อไป



รูปที่ 6.1 แสดงระบบคลื่นไฟฟ้าหัวใจบนไมโครคอมพิวเตอร์

CONDITTIONING AMP

วงจร Conditioning AMP ทำหน้าที่ ขยายสัญญาณคลื่นหัวใจไฟฟ้าที่เข้ามาทาง Probe in put โดยให้มีอัตราการขยายที่ป้อนให้กับภาคอินพุทของวงจรแปลงสัญญาณอนาลอกเป็นดิจิตอล ADC ซึ่งต้องมีอัตราการขยายที่เหมาะสมกับข้อจำกัดของแรงดันอินพุทของ ADC ด้วย ในวงจรที่ใช้ IC1 LM4250 ซึ่งทำหน้าที่เป็น Conditioning AMP โดยต่อในลักษณะเป็นดิฟเฟอร์เรนเชียลแอมป์โดยขา inverting ถูกต่อลงกราวด์ และขา non inverting เป็นอินพุทของวงจร ซึ่ง Gain ที่ใช้วงจรนี้คือ 5 เท่า โดย Gaind ถูกกำหนดโดย R_2, R_3 R_4, R_7 ส่วน $R_1$7.5 m.เป็นตัวเซ็ทค่ากระแสภายในของออฟแอมป์ ส่วน P_1 100K มีไว้สำหรับปรับออฟเซ็ท ส่วน IC_2 741 หรือ OP-07 ทำหน้าที่เป็นวงจรปรับระดับแรงดันอ้างอิงโดยตั้งไว้ที่ 1-1.5 โวลท์ ส่วน C_1, D_1, R_6 เป็นวงจรยกระดับของสัญญาณ (Clamper) ดังนั้นสัญญาณคลื่นไฟฟ้าหัวใจจากเครื่องกำเนิดจะต่อเข้ามายัง JP4 และ ถูกขยายโดย IC_2 LM4250 จะได้เอาท์พุทที่ JP5 เพื่อไปเข้าวงจร A/D Converter 8 bit ต่อไป



Title: CONDITIONING AMP		
Size: A4	Number:	Revision:
Date: 12-MAR 1995		Sheet of
File: CONAMP/1		Drawn By:

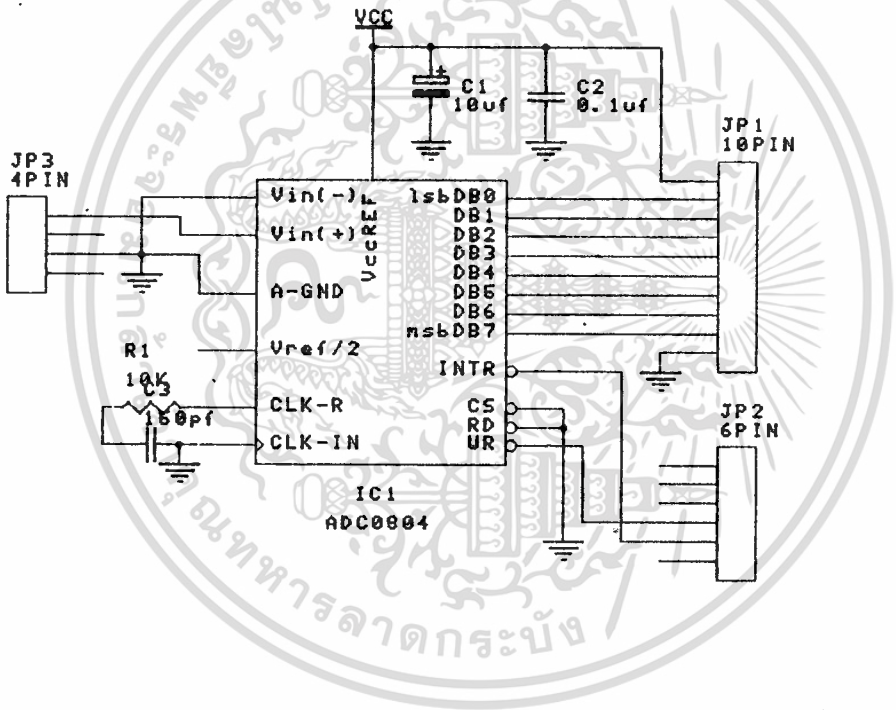
ANALOG TO DIGITAL CONVERTOR (ADC)

สัญญาณคลื่นไฟฟ้าหัวใจได้ถูกขยายโดยวงจร Conditioning AMP จากนั้นจะเข้าวงจรแปลงสัญญาณอนาลอกเป็นสัญญาณดิจิตอล (ADC) เพื่อที่จะส่งค่าข้อมูลที่แปลงเป็นค่าเลขดิจิตอลให้กับ ECG CONTROLLER BROAD เพื่อที่จะสั่งค่าข้อมูลให้กับไมโครคอมพิวเตอร์

ในโครงงานี้ใช้ ADC0804 ซึ่งเป็น Successive Approximation A/D ซึ่งภายในมีเอ้าท์พุทเป็นลักษณะ three state output สามารถที่ต่อกับ CPU ได้โดยตรง ค่าเวลาในการแปลงสัญญาณ (Conversion time) มีค่า 100 uS มีวงจรกำเนิดสัญญาณ clock ภายในโดยการต่อ โดยมีความถี่เท่ากับ

$$f = \frac{1}{(1.1\ RC)}$$

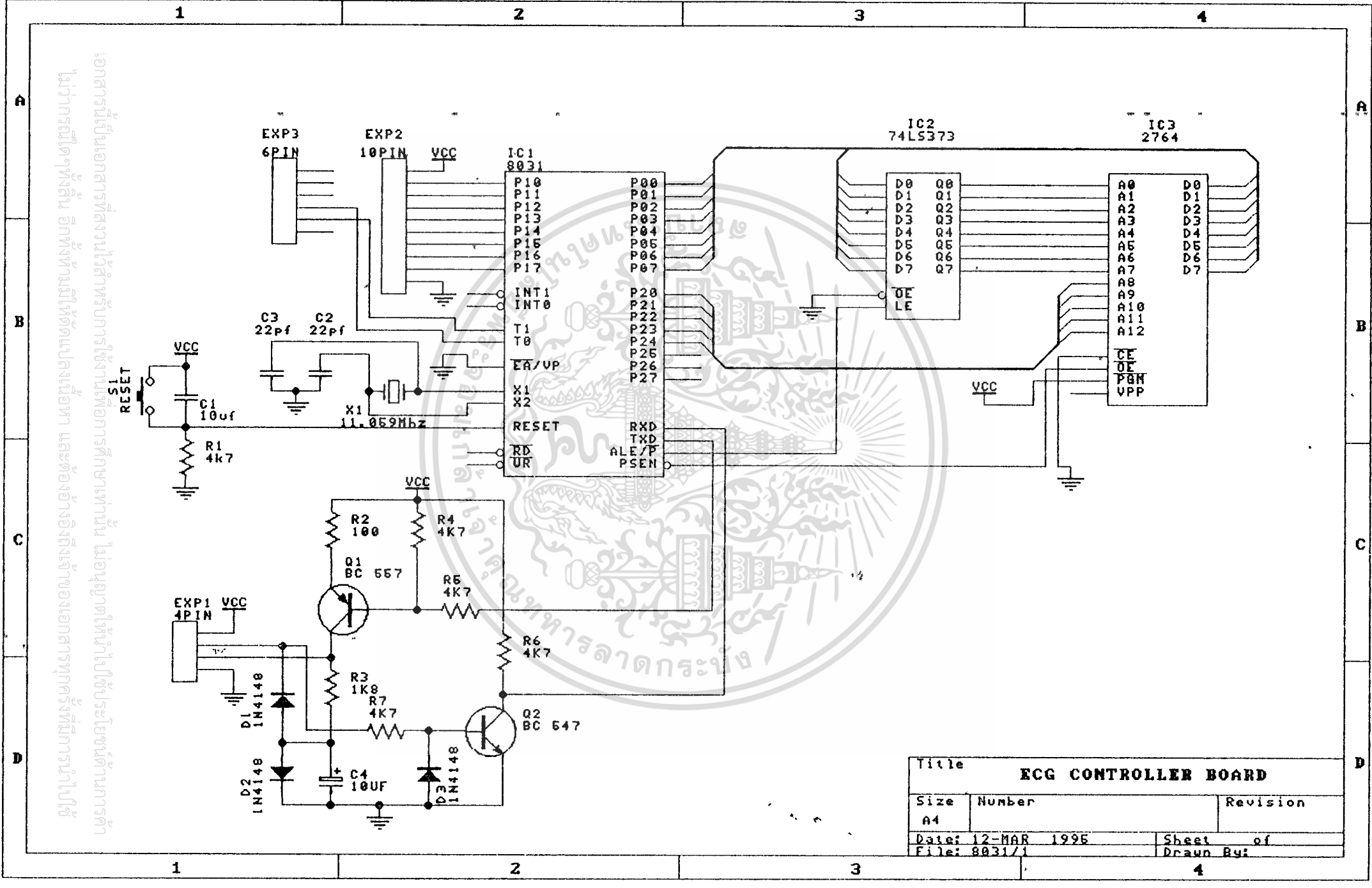
ซึ่งในวงจรที่ใช้งานใช้ค่า R=10K , C=150Pf เพราะฉะนั้น f=606 KH_z แรงดันอินพุทที่ป้อนให้ ADC มีค่า 0-5 โวลท์ โดยเข้าที่ JP3 การทำงานของ ADC ถูกควบคุมโดย CPU 8031 จาก ECG CONTROLLER โดยจะส่งสัญญาณ Start (WR) มาให้ ADC 0804 และ CPU 8031 รอสัญญาณ INT จาก ADC เมื่อมีการแปลงสัญญาณเสร็จแล้ว ส่วนขา CS , RD จะต่อลงกราวด์โดยตรง เพื่อลดความยุ่งยากในการใช้ CPU 8031 สัญญาณดิจิตอลที่ถูกแปลงแล้วจะถูกส่งโดย DB0 - DB7 ที่ JP1 ซึ่งจะส่งให้กับ CPU 8031 ต่อไป



Title: A/D CONVERTOR		
Size: A4	Number	Revision
Date: 12-MAR 1995		Sheet of
File: A/1		Drawn By:

ECG CONTROLLER BROAD

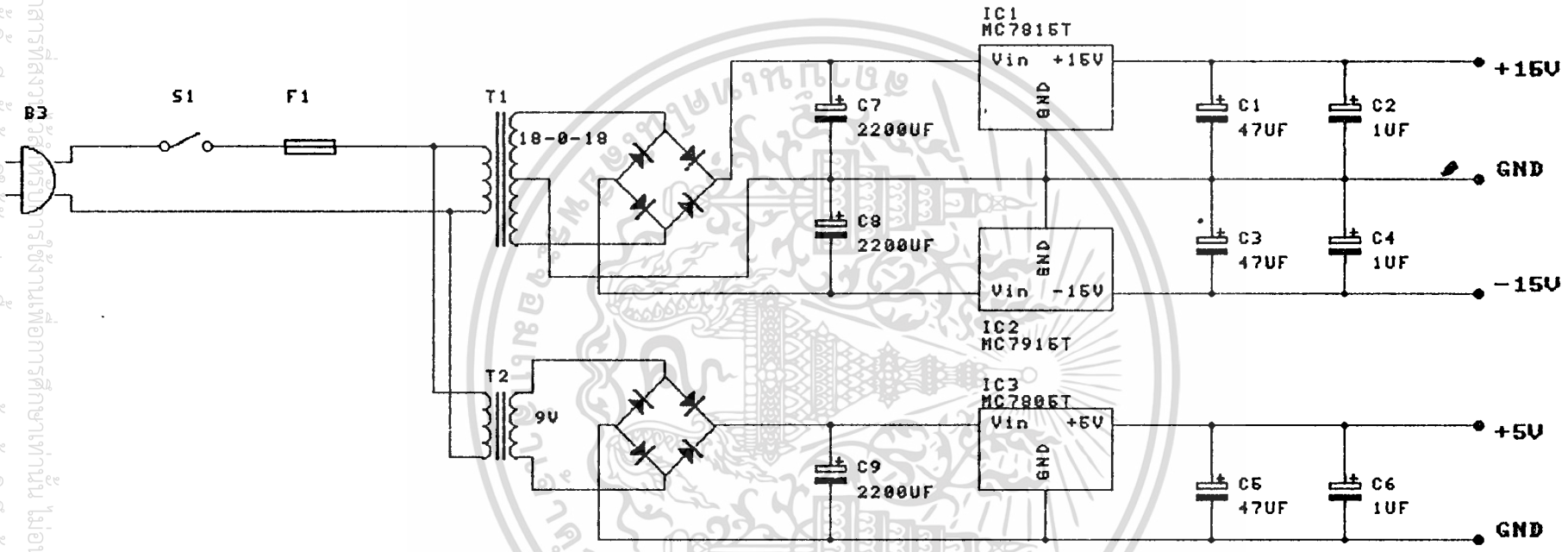
CPU 8031 ทำหน้าที่เป็นตัวประมวลผลข้อมูลที่ได้จากวงจร ADC โดยเข้ามาที่ P1.0-P1.7 หรือ EXP2 และเป็นตัวส่งข้อมูล 8 บิท สัญญาณคลื่นไฟฟ้าหัวใจ ซึ่งเป็นค่าดิจิตอลโดยส่งข้อมูลแบบอนุกรม แบบ RS232 ผ่านทางขา TXD ของ 8031 โดยเริ่มแรก CPU 8031 จะส่งสัญญาณ Srart (WR) ซึ่งเป็น active low ไปให้กับ ADC 0804 ผ่านทาง EXP3 เมื่อ ADC0804 ได้รับสัญญาณ start ก็ทำงานโดยแปลงสัญญาณ ECG เป็นดิจิตอล 8 บิท เมื่อแปลงเสร็จจะส่งสัญญาณ INT ผ่านทาง EXP.3 ซึ่ง CPU 8031 รอสัญญาณ INT เมื่อได้รับสัญญาณก็จะรับสัญญาณดิจิตอล 8 บิทเข้ามาที่พอร์ต 1 (P1.0 - P1.7) จากนั้นก็นำข้อมูลที่รับเข้ามาได้แล้วส่งไปยัง SBUF ภายใน 8031 และส่งค่าข้อมูล 8 บิทโดยผ่านทาง TXD โดยต่อวงจรแปลงระดับสัญญาณให้ติดต่อกับ PC โดยใช้วงจร Q1 BC557 ส่งข้อมูลแบบอนุกรม ผ่านทาง EXP1ไปยัง Micro Computer เพื่อที่จะแสดงผลต่อไป ขา TXD สามารถต่อโดยใช้ MAX232 ก็ได้ ส่วน CPU 8031 ใช้ OSC 11.059 MH_Z สร้างฐานเวลาให้กับ CPU S_1, C_1, R_4 เป็นวงจร Reset โดย Reset Active High ส่วน PORT0, PORT2 เป็น Addressing โดยติดต่อกับ EPROM เบอร์ 2764 ซึ่งเก็บข้อมูลแบบ code โดยการโปรแกรมให้ CPU ทำงานตามที่กล่าวมาข้างต้นขา EA ถูกต่อลงกราวด์เพื่อติดต่อกับหน่วยความจำภายนอกสัญญาณ PSEN ไว้ติดต่อกับ หน่วยความจำแบบ ROM หรือ #2764 นั่นเอง


EXP3
6PIN
EXP2
10PIN
VCC
IC1
8031
P10
P11
P12
P13
P14
P15
P16
P17
P00
P01
P02
P03
P04
P05
P06
P07
INT1
INT0
T1
T0
EA/VP
X1
X2
RESET
RD
WR
P20
P21
P22
P23
P24
P25
P26
P27
RXD
TXD
ALE/P
PSEN
IC2
74LS373
D0
D1
D2
D3
D4
D5
D6
D7
Q0
Q1
Q2
Q3
Q4
Q5
Q6
Q7
OE
LE
IC3
2764
A0
A1
A2
A3
A4
A5
A6
A7
A8
A9
A10
A11
A12
CE
OE
PGM
VPP
C3
22pf
C2
22pf
X1
11.059Mhz
S1
RESET
C1
10uf
R1
4k7
R2
100
R4
4K7
Q1
BC 557
R5
4K7
R6
4K7
EXP1
4PIN
R3
1K8
R7
4K7
Q2
BC 547
D1
1N4148
D2
1N4148
D3
1N4148
C4
10UF
Title
ECG CONTROLLER BOARD
Size
A4
Number
Revision
Date: 12-MAR 1995
Sheet of
File: 8031/1
Drawn By:

DC SUPPLY

เป็นภาคจ่ายไฟให้กับระบบของโครงงาน ซึ่งใช้ DC VOLTAGE 5V, $\pm$ 15 V. โดย Transfromer T1 ป้อนค่า AC Voltage 18-0-18 V. ให้กับวงจร Regtifier โดยมี C_7, C_8 เป็นต้น Filter Capsacitor ลดค่าแรงดันริบเปิ้ล 7815, 7915 เป็น IC แปลงแรงดันเป็น DC Voltage คือ +15 V, -15 V. โดยมี C_1, C_2, C_3, C_4 เป็น Filter ทางด้านเอ๊าท์พุท

Transformer T2 ป้อนค่า AC voltage 0-9 V. ให้กับวงจร Regtifier โดยมี C_9 เป็น Fillter Capacitor ลดค่าแรงดันริบเปิ้ล 7805 เป็น IC แปลงแรงดัน PC Voltage คือ 5V. โดยมี C_5, C_6 เป็น Filter ทางด้านเอ๊าท์พุท



Title: DC SUPPLY		
Size: A4	Number	Revision
Date: 12-MAR 1995	Sheet of	
File: SUPPLY/1	Drawn By:	

โปรแกรมที่ใช้งานจริงของ CPU 8031

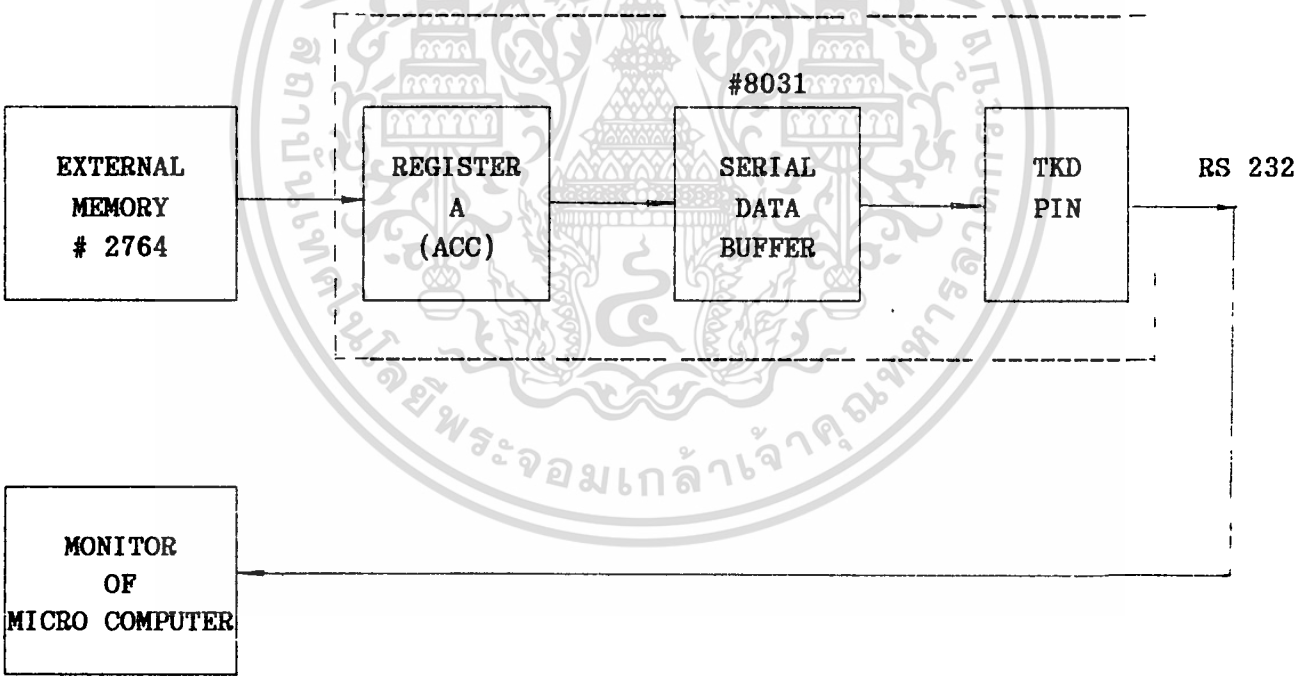
และ ผลการทดสอบ วงจรที่ใช้งานจริงใน

โครงงาน

ป้อนสัญญาณความถี่ต่ำของสัญญาณ Sine wave และ Square wave

การทดสอบการทำงานของโครงงาน

1. ในการทดสอบการทำงานของโครงงานในด้าน Hard Ware ในส่วนของคอนโทรลเลอร์ CPU 8031 ได้ทำการทดสอบโดยการเขียนโปรแกรมใน Emulator EE232 โดยทำการเขียนโปรแกรมให้ CPU 8031 ส่งค่าข้อมูล 8 บิท ที่อยู่ภายใน Memory จำนวน 400 ค่า ส่งออกไปยังขา TXD ซึ่งไปต่อกับพอร์ต 1 หรือ พอร์ต 2 ของ Micro Computer ให้แสดงผลตามค่าข้อมูลซึ่งไม่เขียนเป็นรูปของคลื่นไฟฟ้าหัวใจ ซึ่งได้ผลเป็นที่น่าพอใจ โดยโปรแกรมจะแสดงไว้ในหน้าถัดไป



รูปแสดงขั้นตอนการทดสอบการทำงานของ Hard Ware CPU 8031

การแสดงผลหน้าจอ Monitor สามารถปรับ Range ได้ 4 ช่วง

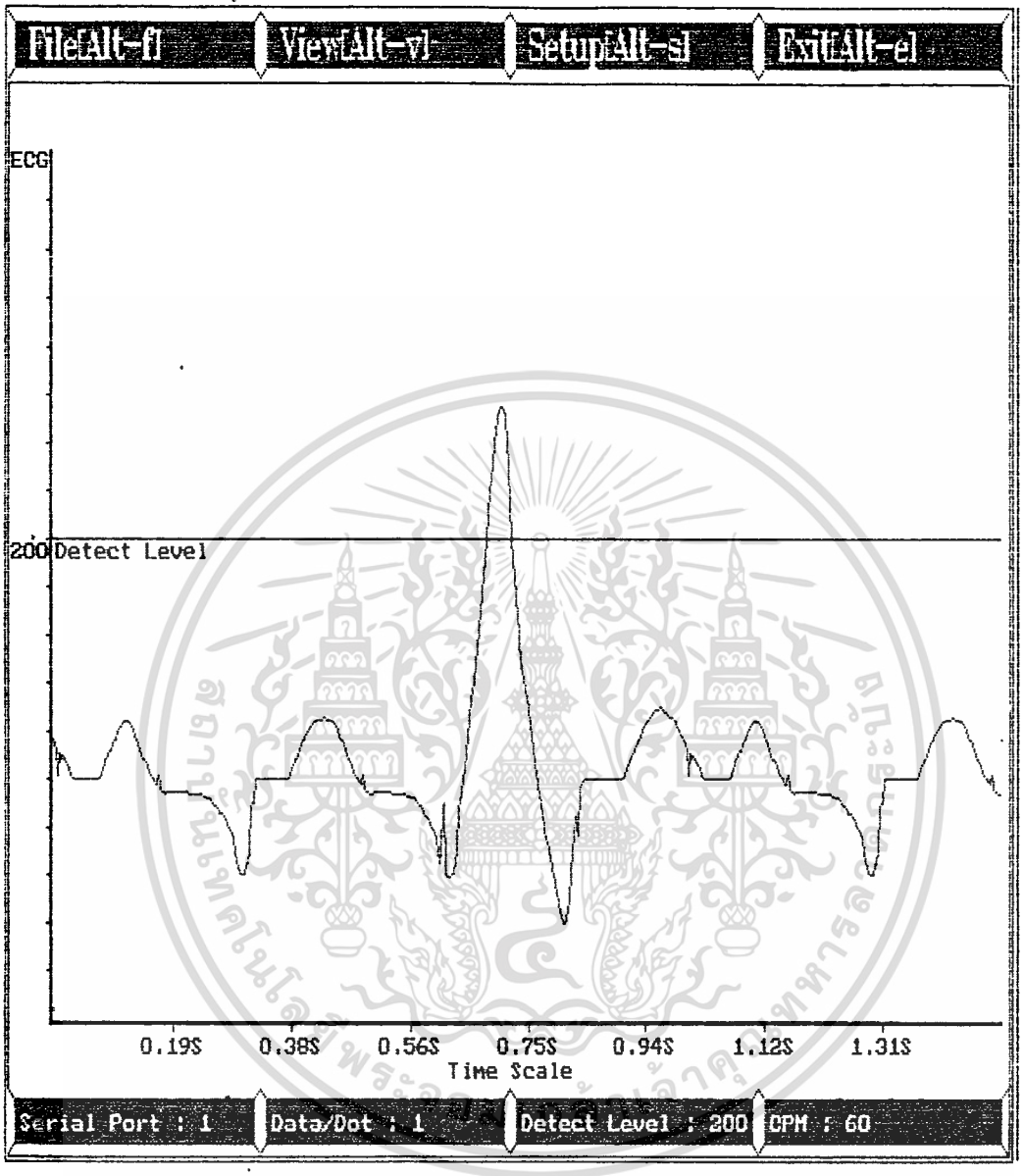
คือ

1). Range 1 (0-1.31S) ดังแสดงในรูปที่ 6.2

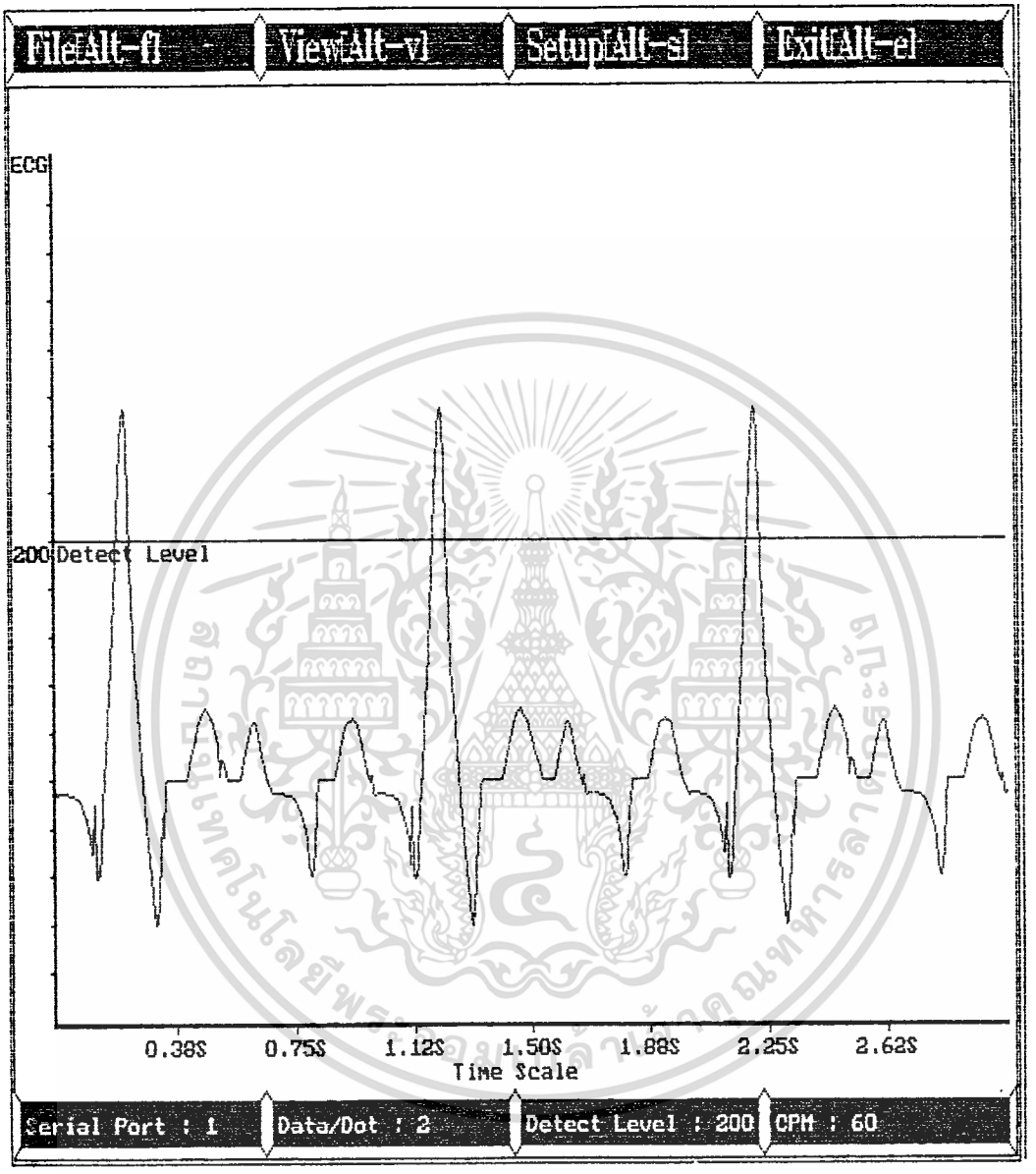
2). Range 2 (0-2.62S) ดังแสดงในรูปที่ 6.3

3). Range 3 (0-3.94S) ดังแสดงในรูปที่ 6.4

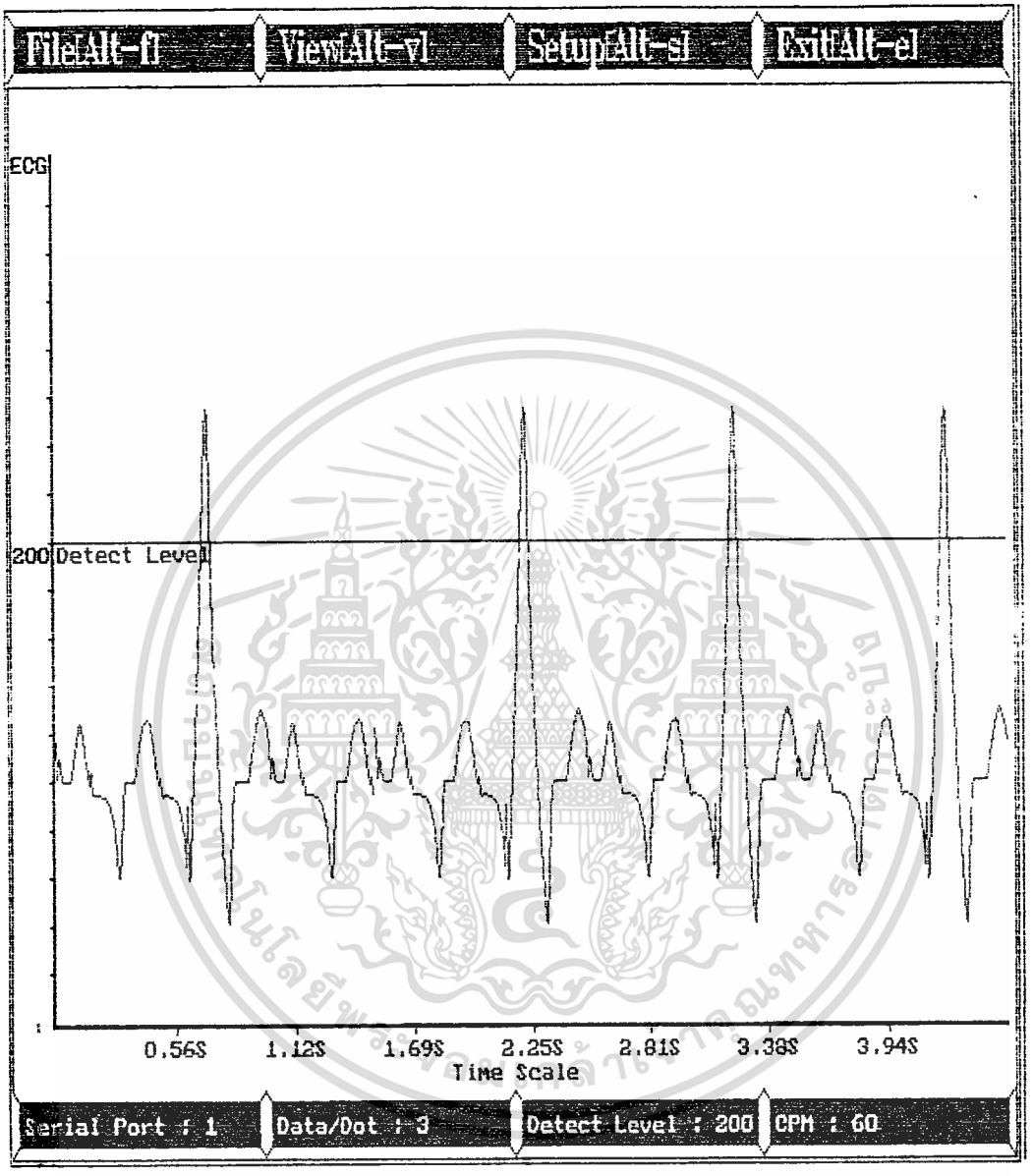
4). Range 4 (0-5.25S) ดังแสดงในรูปที่ 6.5



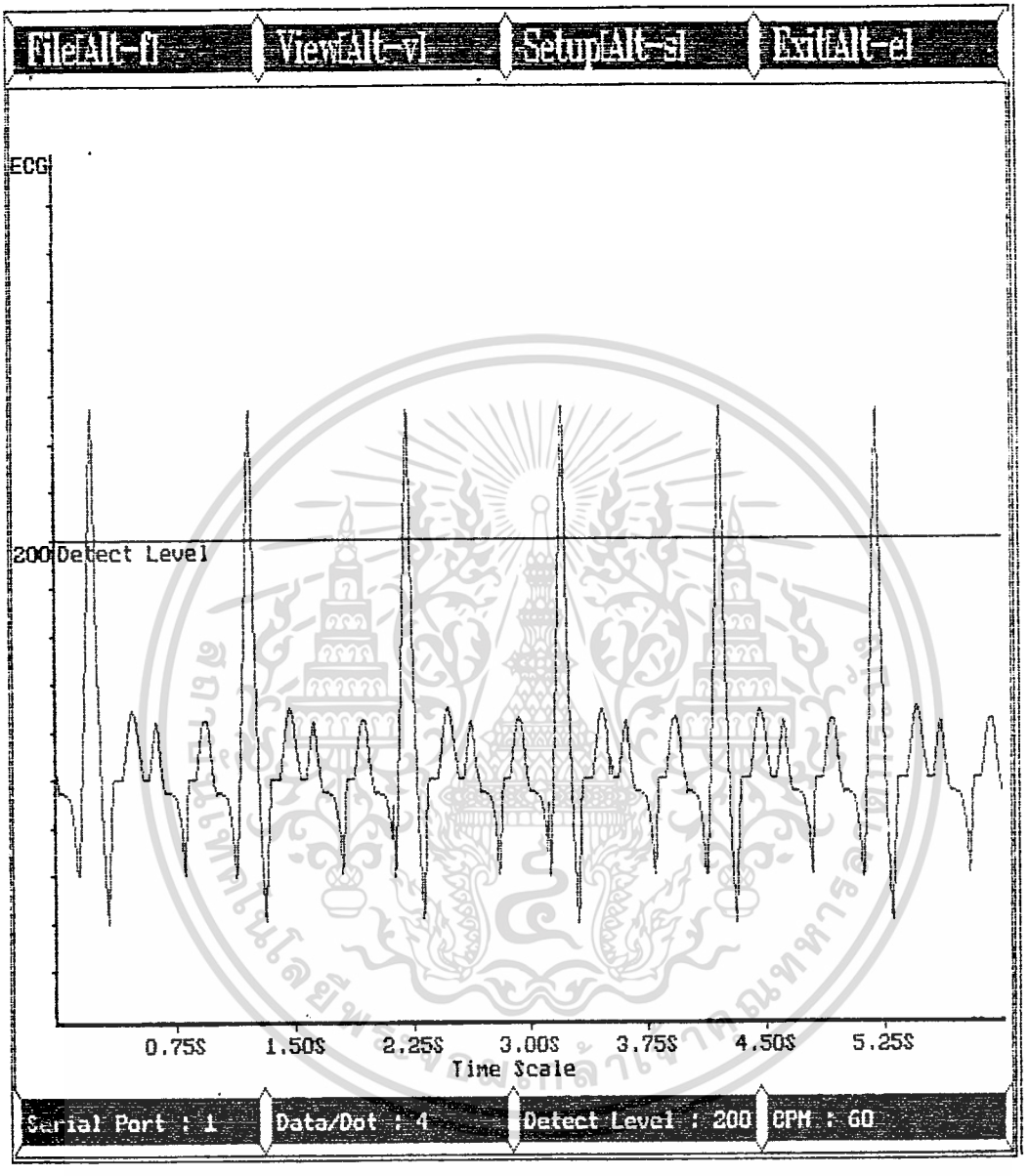
รูปที่ 6.2 แสดงผลหน้าจอ Monitor ปรับ Range ที่ 1 (0-1.35S)
ของสัญญาณ ECG จำลอง วัด CPM ได้ 60 CPM



รูปที่ 6.3 แสดงผลหน้าจอ Monitor ปรับ Range ที่ 2 (0-2.62S) ของสัญญาณ ECG จำลองวัด CPM ได้ 60 CPM



รูปที่ 6.4 แสดงผลหน้าจอ Monitor ปรับ Range ที่ 3 (0-3.94S)
ของสัญญาณ ECG จำลองวัด CPM ได้ 60 CPM



รูปที่ 6.5 แสดงผลหน้าจอ Monitor ปรับ Range ที่ 4 (0-5.25S)
ของสัญญาณ ECG จำลองวัด CPM ได้ 60 CPM

โปรแกรมที่ใช้ทดสอบในการส่งค่าสัญญาณ ECG

เทียม ให้กับ SOFT WARE บนไมโครคอมพิวเตอร์

```
0000                                    CPU     "8051.TBL"
0000                                    HOF     "INT8"
0000                                    ORG     0000H
                        ;
                        ;
0039 =                  ECG:            EQU     0039H
0099 =                  TI:             EQU     099H
008E =                  TR1:            EQU     08EH
0098 =                  RI:             EQU     098H
0080 =                  P0:             EQU     80H
0081 =                  SP:             EQU     81H
0082 =                  DPL:            EQU     82H
0083 =                  DPH:            EQU     83H
008A =                  TL0:            EQU     8AH
008B =                  TL1:            EQU     8BH
008C =                  TH0:            EQU     8CH
008D =                  TH1:            EQU     8DH
0090 =                  P1:             EQU     90H
00A0 =                  P2:             EQU     0A0H
00B0 =                  P3:             EQU     0B0H
0089 =                  TMOD:           EQU     89H
00C0 =                  TCON:           EQU     0C0H
0098 =                  SCON:           EQU     98H
0099 =                  SBUF:           EQU     99H
00A8 =                  IE:             EQU     0A8H
00B8 =                  IP:             EQU     0B8H
00D0 =                  PSW:            EQU     0D0H
00E0 =                  ACC:            EQU     0E0H
00F0 =                  B:              EQU     0F0H
0087 =                  PCON:           EQU     87H
                        ;
                        ;
                        ;
                        ;
0000 7F00               DELAY:          MOV     R7,#00H
0002 7E00               RES2:           MOV     R6,#00H
0004 00                 RES1:           NOP
0005 00                                 NOP
0006 DEFC                               DJNZ    R6,RES1
0008 DFF8                               DJNZ    R7,RES2
                        ;
                        ;
000A 759852                             MOV     SCON,#52H   ;SET MODE 8 BIT UART
                        ;
000D 758920                             MOV     TMOD,#20H   ;SET MODE TIMER 8 BIT AUTORE
LOAD
0010 758DFD                             MOV     TH1,#0FDH   ;9600 BAUD
0013 D28E                               SETB    TR1         ;START TIMER 1
                        ;
                        ;
                        ;
                        ;
0015 900039             START:          MOV     DPTR,#ECG
0018 E500               ADJ:            MOV     A,00H
001A 93                                 MOVC    A,@A+DPTR
001B F599                               MOV     SBUF,A
001D 3099FD                             JNB     TI,$
0020 C299                               CLR     TI
                        ;
```

```
                        ;
                        ;
                        ;
0022 7F00               DELAY1:  MOV     R7,#00H
0024 00                 RES3:    NOP
0025 00                          NOP
0026 00                          NOP
0027 00                          NOP
0028 00                          NOP
0029 00                          NOP
002A DFF8                        DJNZ    R7,RES3
                        ;
                        ;
                        ;
002C A3                          INC     DPTR
002D E583                        MOV     A,DPH
002F B401E6                      CJNE    A,#01H,ADJ
0032 E582                        MOV     A,DPL
0034 B4C8E1                      CJNE    A,#0C8H,ADJ
0037 80DC                        SJMP    START
                        ;
                        ;
                        ;
0039 6464646464         ECG:     DFB     64H,64H,64H,64H,64H,64H,64H,64H,64H,64H
0043 6464646464                  DFB     64H,64H,64H,64H,64H,64H,64H,64H,64H,64H
004D 6465666686A                 DFB     64H,65H,66H,68H,6AH,6BH,6DH,6EH,70H,73H
0057 7475777879                  DFB     74H,75H,77H,78H,79H,7AH,7BH,7CH,7CH,7DH
0061 7D7D7D7E7E                  DFB     7DH,7DH,7DH,7EH,7EH,7DH,7DH,7DH,7CH,7CH
006B 7B7A787675                  DFB     7BH,7AH,78H,76H,75H,74H,72H,70H,6EH,6CH
0075 6B69676665                  DFB     6BH,69H,67H,66H,65H,63H,63H,62H,66H,60H
007F 5F5F5E5E5E                  DFB     5FH,5FH,5EH,5EH,5EH,5FH,5FH,5FH,5FH,5FH
0089 5F5F5F5F5F                  DFB     5FH,5FH,5FH,5FH,5FH,5FH,5EH,5EH,5EH,5EH
0093 5E5E5E5E5E                  DFB     5EH,5EH,5EH,5EH,5EH,5DH,5DH,5DH,5CH,5CH
009D 5C5B5A5959                  DFB     5CH,5BH,5AH,59H,59H,58H,56H,55H,54H,53H
00A7 51504F4D4B                  DFB     51H,50H,4FH,4DH,4BH,46H,41H,46H,59H,5AH
00B1 3C3B3B3B3C                  DFB     3CH,3BH,3BH,3BH,3CH,3DH,41H,46H,59H,5AH
00BB 5F64696E77                  DFB     5FH,64H,69H,6EH,77H,7CH,87H,88H,8CH,94H
00C5 9BA2AAB0BA                  DFB     9BH,0A2H,0AAH,0B0H,0BAH,0BEH,0C8H,0D2H,C
DCH,0E1H
00CF E7F0F5FAFE                  DFB     0E7H,0F0H,0F5H,0FAH,0FEH,0FFH,0FFH,0FEH,
0FBH,0F0H
00D9 E6D7C8BEB9                  DFB     0E6H,0D7H,0C8H,0BEH,0B9H,0AFH,0A8H,0A0H,
92H,93H
00E3 8C8A85827A                  DFB     8CH,8AH,85H,82H,7AH,76H,71H,6BH,66H,62H
00ED 5C5A55504B                  DFB     5CH,5AH,55H,50H,4BH,4AH,46H,43H,3FH,3CH
00F7 3835302C2A                  DFB     38H,35H,30H,2CH,2AH,28H,28H,29H,2DH,32H
0101 3C464D5550                  DFB     3CH,46H,4DH,55H,50H,4DH,61H,62H,63H,64H
010B 6464646464                  DFB     64H,64H,64H,64H,64H,64H,64H,64H,64H,64H
0115 6464646464                  DFB     64H,64H,64H,64H,64H,64H,64H,64H,64H,64H
011F 6464646465                  DFB     64H,64H,64H,64H,65H,66H,68H,6AH,6BH,6EH
0129 6F70727375                  DFB     6FH,70H,72H,73H,75H,76H,78H,7AH,7BH,7CH
0133 7D7E7F8081                  DFB     7DH,7EH,7FH,80H,81H,81H,81H,82H,82H,81H
013D 80807F7E7E                  DFB     80H,80H,7FH,7EH,7EH,7EH,7DH,7BH,7AH,78H
0147 7776757271                  DFB     77H,76H,75H,72H,71H,66H,6FH,6DH,6EH,6CH
0151 6B6A696665                  DFB     6BH,6AH,69H,66H,65H,64H,64H,64H,64H,64H
015B 6464646464                  DFB     64H,64H,64H,64H,64H,64H,64H,64H,64H,64H
0165 6465666686B                 DFB     64H,65H,66H,68H,6BH,6DH,6EH,70H,70H,73H
016F 7475777797A                 DFB     74H,75H,77H,79H,7AH,7BH,7CH,7CH,7CH,7CH
0179 7B7A787675                  DFB     7BH,7AH,78H,76H,75H,72H,70H,6EH,6CH,6CH
```

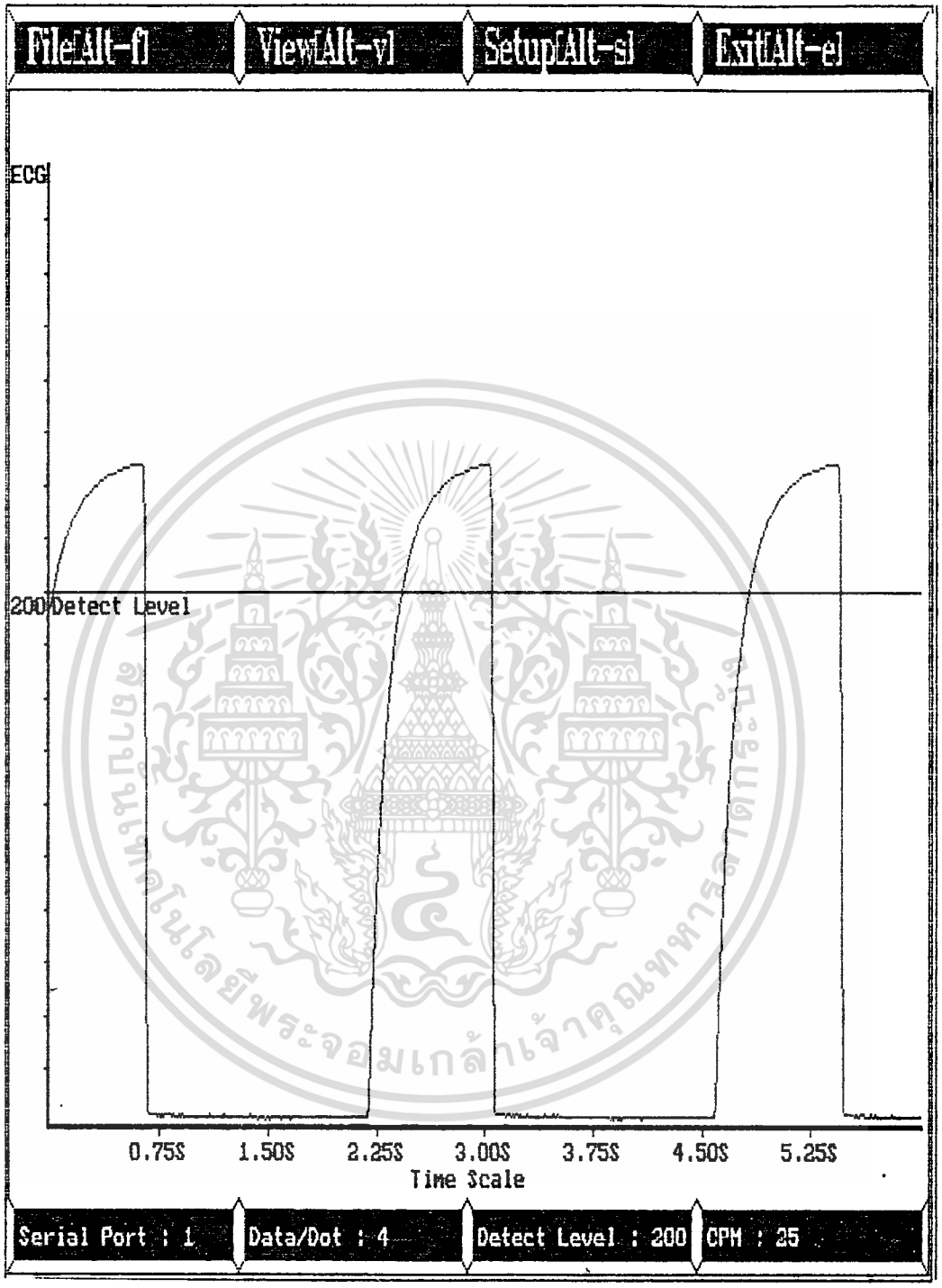
```
0183 6B69676665                 DFB   6BH,69H,67H,66H,65H,63H,62H,62H,66H,60H
018D 5F5F5E5F5F                 DFB   5FH,5FH,5EH,5FH,5FH,5FH,5FH,5FH,5FH,5FH
0197 5F5F5F5F5F                 DFB   5FH,5FH,5FH,5FH,5FH,5FH,5FH,5EH,5EH,5EH
01A1 5E5E5E5E5E                 DFB   5EH,5EH,5EH,5EH,5EH,5EH,5DH,5DH,5CH,5CH
01AB 5C5B5A5959                 DFB   5CH,5BH,5AH,59H,59H,58H,56H,56H,56H,54H
01B5 5351504F4D                 DFB   53H,51H,50H,4FH,4DH,4BH,46H,41H,3FH,3DH
01BF 3C3C3C4141                 DFB   3CH,3CH,3CH,41H,41H,46H,59H,5AH,5AH,5AH
                        ;
                        ;
0000                            END
```

```
00E0  ACC       0018  ADJ       00F0  B
0000  DELAY     0022  DELAY1    0083  DPH
0082  DPL       0039  ECG       00A8  IE
00B8  IP        0080  P0        0090  P1
00A0  P2        00B0  P3        0087  PCON
00D0  PSW       0004  RES1      0002  RES2
0024  RES3      0098  RI        0099  SBUF
0098  SCON      0081  SP        0015  START
00C0  TCON      008C  TH0       008D  TH1
0099  TI        008A  TL0       008B  TL1
0089  TMOD      008E  TR1
```

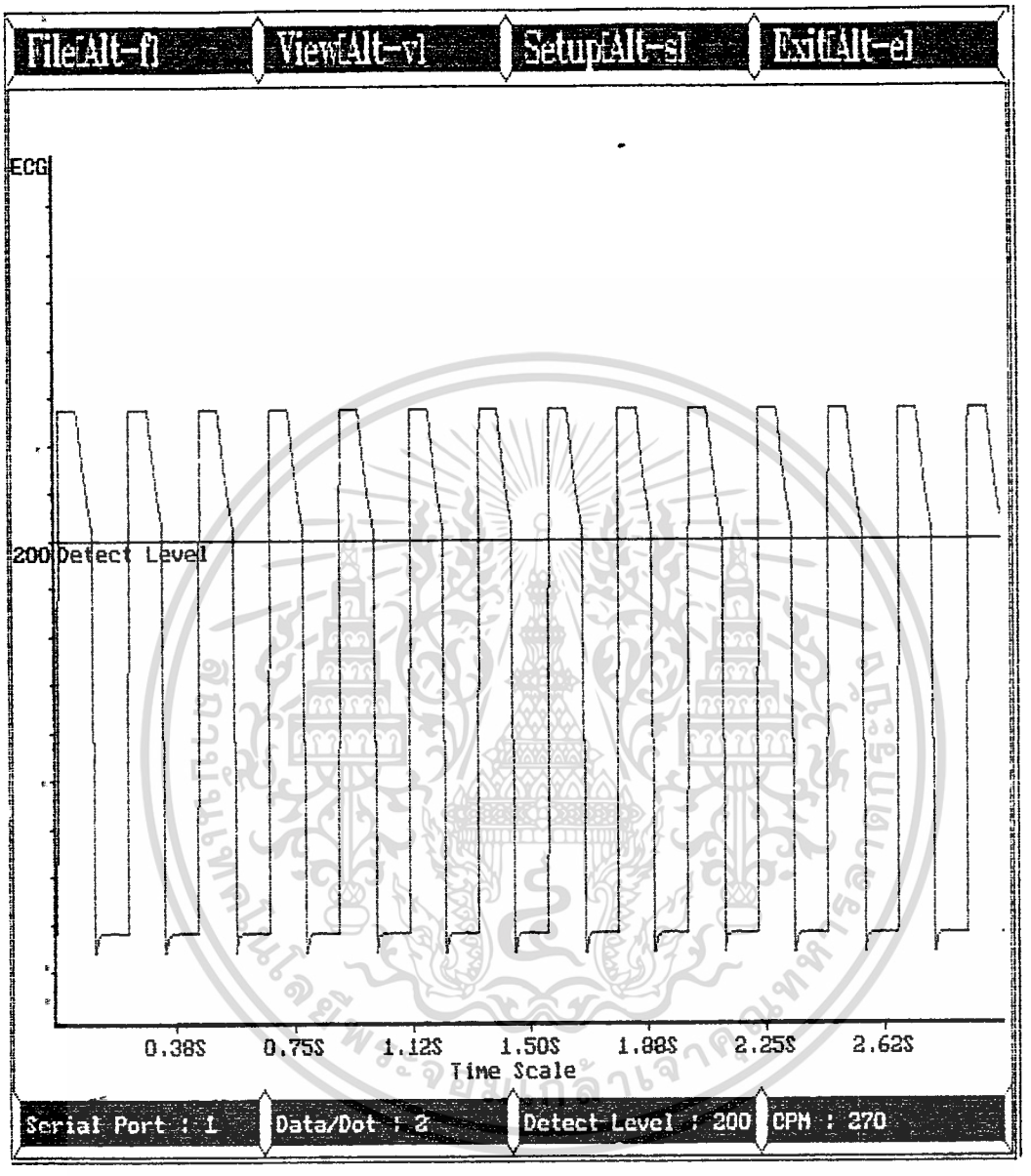
2. การทดสอบวงจรใช้งานจริง

โดยการป้อนสัญญาณที่มีความถี่ต่ำ เช่นวงจร Astable Multivibator ป้อนให้กับภาคอินพุทของ Hard Ware และเขียนโปรแกรมที่ใช้งานจริง ให้กับ 8031 โดยผ่านทาง Emulator EE232 และดูภาพที่แสดงผลทางหน้าจอของ Micro Computer

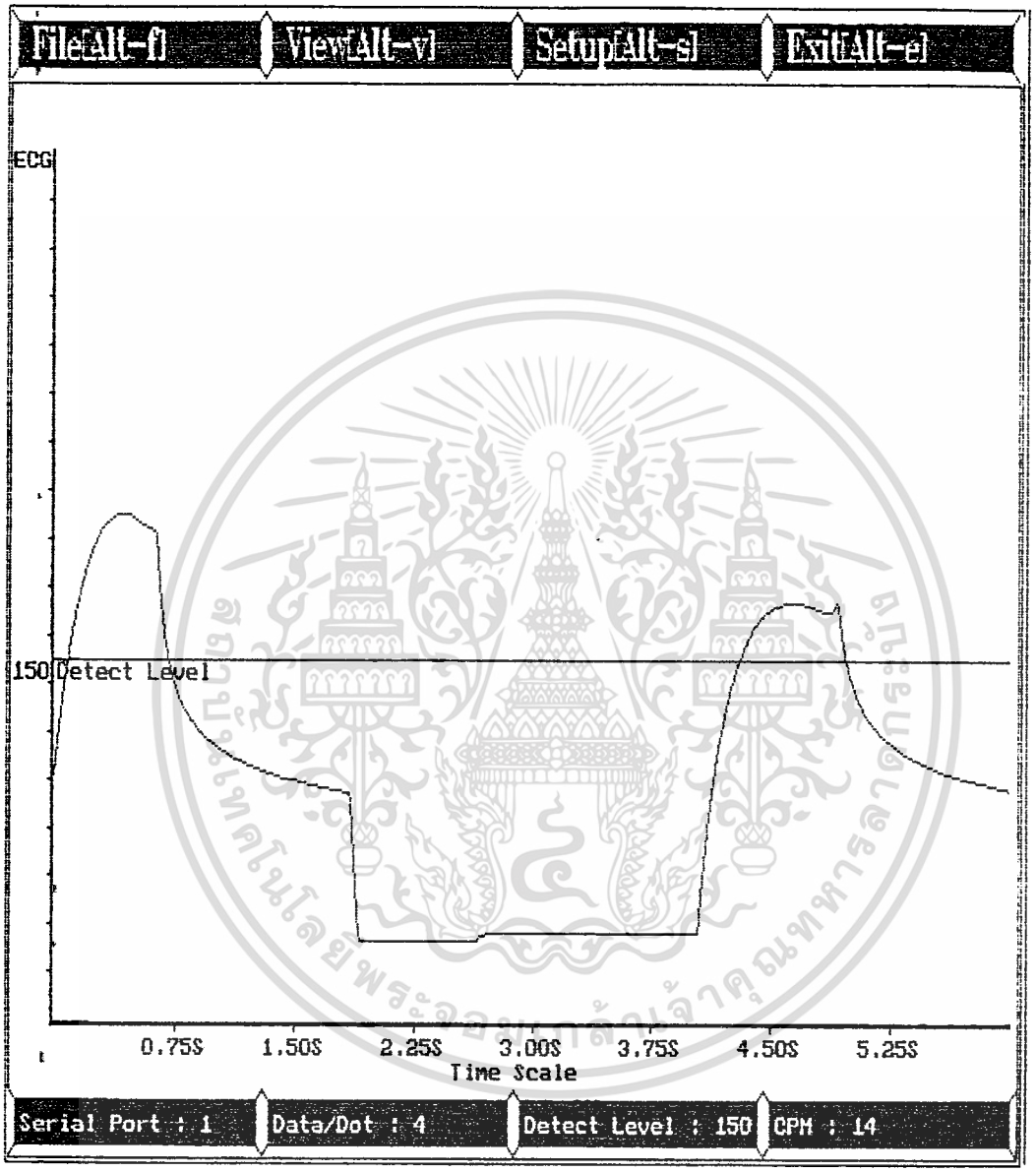
รูปแสดงขั้นตอนการทดสอบวงจที่ใช้งานจริง



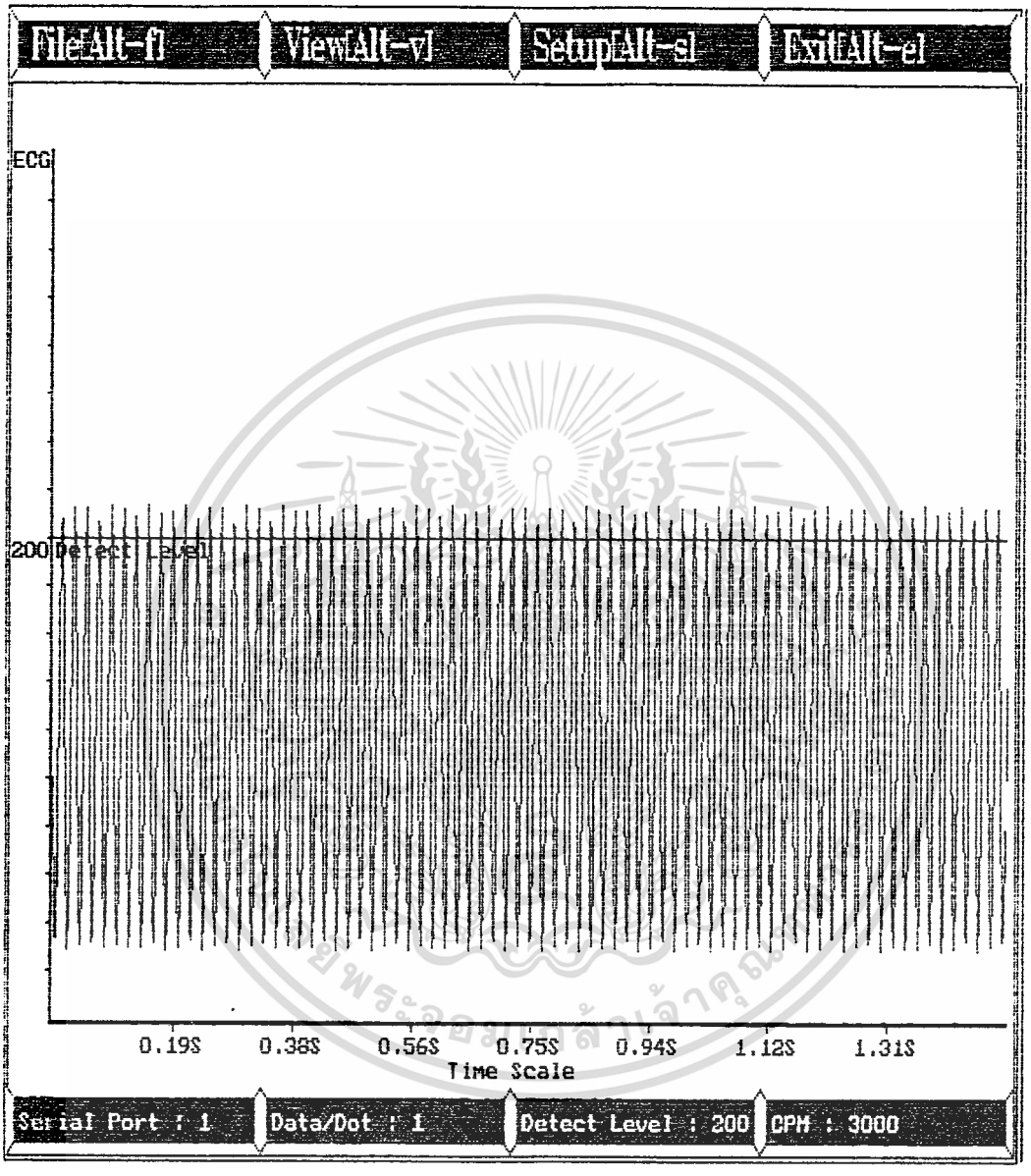
รูปที่ 6.6 แสดงผลหน้าจอของ Monitor โดยการป้อนสัญญาณจากวงจร Astable Multiribrator ความถี่ 0.5 H_Z วัด CPM ได้ 25 CPM



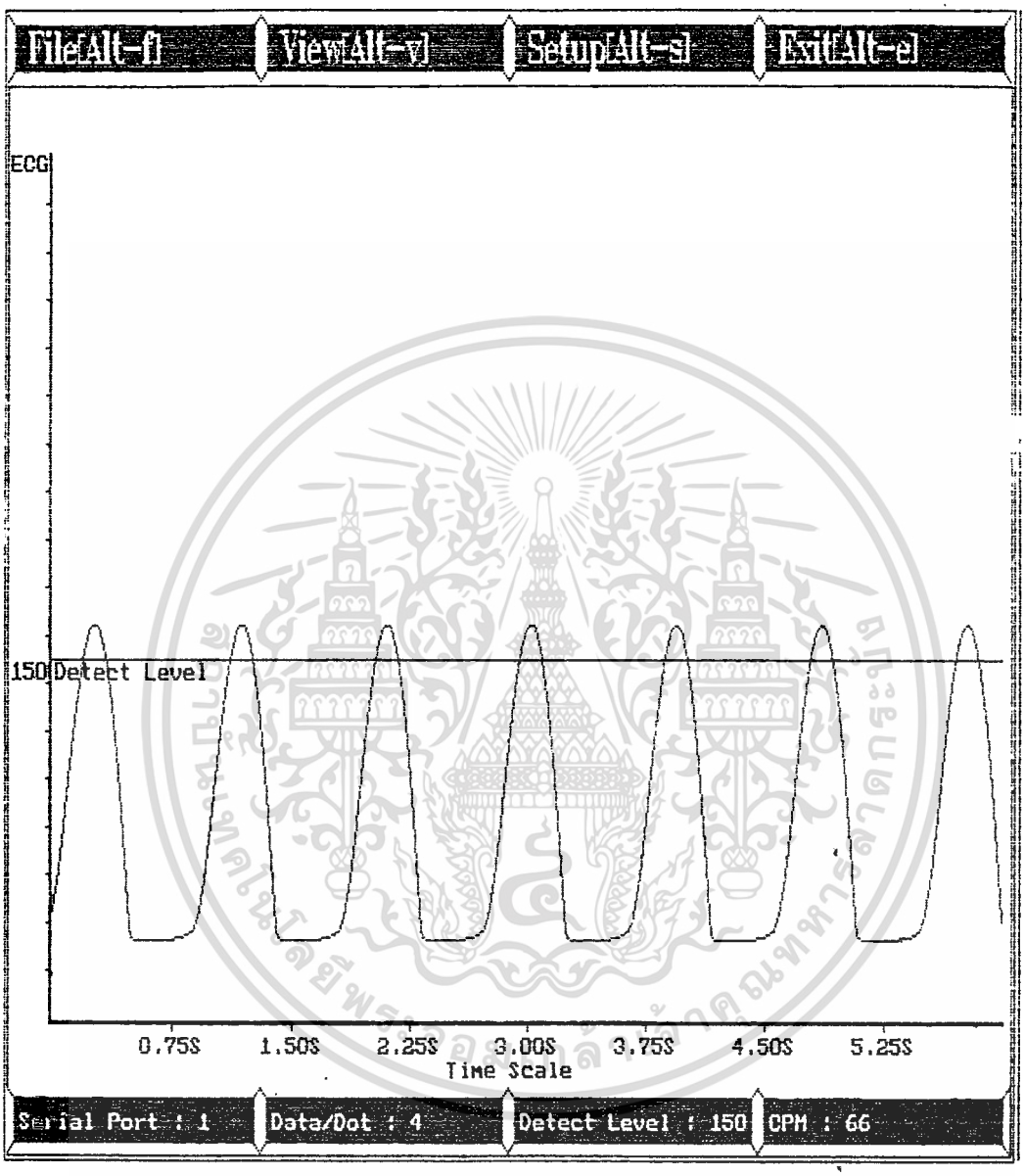
รูปที่ 6.7 แสดงผลหน้าจอของ Monitor โดยการป้อนสัญญาณจากวงจร Astable Multivibrator ความถี่ 4.5 H_Z วัด CPM ได้ 270 CPM



รูปที่ 6.8 แสดงผลการทดสอบหาค่าความถี่ต่ำสุดที่สามารถแสดงผลบน Monitor ได้ โดยการปรับค่าความถี่จาก Function Generator วัดได้ 0.25 H_Z วัด CPM ได้ 14 CPM



รูปที่ 6.9 แสดงผลการทดสอบหาค่าความถี่สูงสุดที่สามารถแสดงผลบน Monitor ได้ ซึ่งวัดได้โดยการปรับค่าความถี่จาก Function Generator วัดได้ $50H_Z$ วัด CPM ได้ 3000 CPM



รูปที่ 6.10 แสดงผลหน้าจอของ Monitor โดยการป้อนสัญญาณ Sine wave ความถี่ 1.1 H_Z วัด CPM ได้ 66 CPM

โปรแกรมที่ใช้งานจริงของ CPU 8031

ในการส่งข้อมูล 8 บิท แบบอนุกรม

```
0000                                    CPU     "8051.TBL"
0000                                    HOF     "INT8"
0000                                    ORG     0000H
                        ;
                        ;
0080 =                  P0:             EQU     80H
0099 =                  TI:             EQU     99H
008E =                  TR1:            EQU     8EH
0098 =                  RI:             EQU     98H
0081 =                  SP:             EQU     81H
0082 =                  DPL:            EQU     82H
0083 =                  DPH:            EQU     83H
008A =                  TL0:            EQU     8AH
008B =                  TL1:            EQU     8BH
008C =                  TH0:            EQU     8CH
008D =                  TH1:            EQU     8DH
0090 =                  P1:             EQU     90H
00A0 =                  P2:             EQU     0A0H
00B0 =                  P3:             EQU     0B0H
0089 =                  TMOD:           EQU     89H
00C0 =                  TCON:           EQU     0C0H
0098 =                  SCON:           EQU     98H
0099 =                  SBUF:           EQU     99H
00A8 =                  IE:             EQU     0A8H
00B8 =                  IP:             EQU     0B8H
00D0 =                  PSW:            EQU     0D0H
00E0 =                  ACC:            EQU     0E0H
00F0 =                  B:              EQU     0F0H
0087 =                  PCON:           EQU     87H
                        ;
                        ;
                        ;
                        ;
                        ;
0000 7F00               START:          MOV     R7,#00H
0002 7E00               RES2:           MOV     R6,#00H
0004 00                 RES1:           NOP
0005 00                                 NOP
0006 DEFC                               DJNZ    R6,RES1
0008 DFF8                               DJNZ    R7,RES2
                        ;
                        ;
000A 759852                             MOV     SCON,#52H
000D 758920                             MOV     TMOD,#20H
0010 758DFD                             MOV     TH1,#0FDH
0013 D28E                               SETB    TR1
                        ;
                        ;
0015 85FF90                             MOV     P1,0FFH
0018 85FFB0                             MOV     P3,0FFH
001B C2B4               RUN:            CLR     0B4H            ;WRITE SIGNAL
001D D2B4                               SETB    0B4H
001F 20B5FD             WAIT:           JB      0B5H,WAIT
0022 E590                               MOV     A,P1
0024 F599                               MOV     SBUF,A
0026 3099FD                             JNB     TI,$
0029 C299                               CLR     TI
                        ;
```

```
                        ;
                        ;
002B AF00               DELAY:  MOV     R7,00H
002D 00                 RES3:   NOP
002E 00                         NOP
002F 00                         NOP
0030 00                         NOP
0031 00                         NOP
0032 00                         NOP
0033 DFF8                       DJNZ    R7,RES3
0035 80E4                       SJMP    RUN
                        ;
0000                            END
```

```
00E0  ACC               00F0  B               002B  DELAY
0083  DPH               0082  DPL             00A8  IE
00B8  IP                0080  P0              0090  P1
00A0  P2                00B0  P3              0087  PCON
00D0  PSW               0004  RES1            0002  RES2
002D  RES3              0098  RI              001B  RUN
0099  SBUF              0098  SCON            0081  SP
0000  START             00C0  TCON            008C  TH0
008D  TH1               0099  TI              008A  TL0
008B  TL1               0089  TMOD            008E  TR1
001F  WAIT
```

บทที่ 7

บทสรุปและวิจารณ์ผลการทำงาน

โครงงานระบบแสดงคลื่นไฟฟ้าหัวใจบนไมโครคอมพิวเตอร์ ซึ่งเป็นการศึกษาความเป็นไปได้ของการรับ-ส่ง ข้อมูลแบบอนุกรม และการแสดงผลให้มีความผิดพลาดน้อยหรือความเชื่อถือได้ของการแสดงผล ซึ่งจากการได้ทำโครงงานนี้ ได้ทำการศึกษาการเขียนโปรแกรม การแสดงผลบนหน้าจอ MICRO COMPUTER การศึกษาเกี่ยวกับวงจรที่ใช้งานในโครงงานไม่ว่าเป็นวงจร Conditioning AMP, Analog to Digital Converter, การใช้ Controller CPU 8031 ซึ่งต้องศึกษาทั้งด้าน Hard Ware, Soft Ware, การรับ-ส่งข้อมูลแบบอนุกรม

การทำงานด้าน Hard Ware สามารถวัดความถี่ที่ใช้งานได้ ซึ่งได้ผลเป็นที่น่าพอใจในทางแนวคิดความถี่ของเครื่องวัดควรจะวัดได้ คือ $150H_Z$ ในการวัดสัญญาณที่ความถี่เกิน $50H_Z$ การแสดงผลที่หน้าจอ Monitor จะเป็นลักษณะขีดๆ เป็นเส้นๆ ดังในการทดลอง แต่ค่าที่วัดได้จากหน้าจอคือ CPM ให้ผลใกล้เคียงกับความถี่ที่ป้อนเข้า Input ของ Hard Ware

การทำงานด้าน Soft Ware แสดงผลหน้าจอบนไมโครคอมพิวเตอร์ ซึ่งค่าเวลาที่ใช้แสดงประมาณ 5 วินาที และสามารถพิมพ์ผลการแสดงออกที่ Printer โดยการเลือกที่หน้าจอเมนู (Menu) โดยตรง

บรรณานุกรม

สุนทร วิทูสุรพจน์ ; "การโปรแกรมภาษาแอสเซมบลีของไมโครคอนโทรลเลอร์ตระกูล 8051" บริษัท ซีเอ็ดยูเคชั่น จำกัด , 2537

สุนทร วิทูสุรพจน์ ; "การใช้งานไมโครคอนโทรลเลอร์ตระกูล 8051", บริษัท ซีเอ็ดยูเคชั่น จำกัด , 2537

National Semiconductor Corpowation ; "General Purpose Linear Devices Data book", California, 1989, Edition

Intel Corporation ; "MCS-51 Microcontrollers", บริษัท อีทีที จำกัด, 2535

นต.ดร.ไพศาล สงวนหมู่, รศ.ยืน ภู่วรรณ ; "การสื่อสารข้อมูล และ ไมโครคอมพิวเตอร์-เนตเวอร์ค", บริษัท ซีเอ็ดยูเคชั่น จำกัด, 2528.

สนั่น สุขวัฒน์ ; "การวิภาคศาสตร์และสรีรวิทยา" คณะแพทย์ศาสตร์มหาวิทยาลัย มหิดล

ภาคผนวก

```
/*=====================================================================*/
/*                      Main Programs For Project                      */
/*            PRO22.C is 400 Samping Rate & Print Screen               */
/*          Programs Design : MR. Perapong Wattanagasamtum             */
/*=====================================================================*/

/* Quality Print Screen */
#define SINGLE 0
#define DOUBLE 1
#define QUAD   2

#include<dos.h>
#include<bios.h>
#include<stdio.h>
#include<stdlib.h>
#include<graphics.h>
#include<time.h>
#include"pro22men.c"
#include"sound.c"

/* Default Setup */
int fs_data=4;                  /* Samping data plot                    */
int PORT   =0;                  /* No. Serial Port (0=com1)             */
int Peak   =200;                /* Peak Detect Value                    */
#define xplot           1       /* Set Distant between data in X axis */

/* Define Display Area Plot Signal */
#define Size            2400    /* No. Array of data                    */
#define D_Left          30      /* Set point Display Signal Left        */
#define D_Right         629     /* Set point Display Signal Right       */
#define Top             60      /* Set point Display Signal Top         */
#define Bottom          420     /* Set point Display Signal Bottom      */

/*================================================
                COLOR IN TURBO C
  ================================================
DARK COLORS
    BLACK       BLUE            GREEN
    CYAN        RED             MAGENTA
    BROWN       LIGHTGRAY       DARKGRAY

LIGHT COLORS
    LIGHTBLUE   LIGHTGREEN      LIGHTCYAN
    LIGHTRED    LIGHTMAGENTA    YELLOW
    WHITE
  ================================================ */


unsigned short int Data[Size],Buffer[Size];
char name[15],Oldv[25];
FILE *fp;
int v,Rate=1;
time_t Start,End;
float Diff;
int C_Scale =   Yellow   ; /* Scale axis  Color                */
int C_Scales=   Lightcyan; /* Scale  Color                     */
int C_Plot  =   White    ; /* Signal plot  Color               */
int C_Detect=   Lightcyan; /* Line Detect Level  Color         */
int C_Text  =   Yellow   ; /* Text in Display  Color           */
int C_Block =   Green    ; /* Data Entry Block  Color          */
int C_Etext =   Yellow   ; /* Error Text Color                 */

/*================================================
                  MAIN PROGRAM
```

```
  ============================================== */
main()
{ Chkspeed();
  Open_graph();
  Set_text();
  Menu(); /* main_menu have 1 set menu */
  Scale();
  Display();
  Read_key();
  Logo();
  closegraph();
  printf("\n Good BYE.");
  exit(0);
}

VIEW()
{   port_init(PORT,0xe3);
    Views:if(!kbhit())
            {
              setcolor(C_Text);
              outtextxy(42,60,"Now Rx Data from serial ");
              outtextxy(42,80,"Press any key Cancel View ");
              setcolor(C_Etext);
              Score(230,60,PORT+1);
              Rport();
              setcolor(C_Text);
              outtextxy(42,42,"Press any key Stop View mode");
              view_int();
              setcolor(C_Back);
              outtextxy(42,42,"Press any key Stop View mode");
              goto Views;
            }
  return;
}

SAVE()
{  Sname:
   Get_name();
    if( (fp=fopen(name,"wb") )==NULL)
       { setcolor(C_Etext);
         outtextxy(250,160,"Cannot Write file");
         Sound();
         goto Sname;
       }
    fwrite(&Data,sizeof(int),Size,fp);
    fclose(fp);
   return;
}

OPEN()
{   Rname:
    Get_name();
    if( (fp=fopen(name,"rb") )==NULL)
       { setcolor(C_Etext);
         outtextxy(250,160,"Cannot Open file");
         Sound();
         goto Rname;
       }
    fread(Data,sizeof(int),Size,fp);
    fclose(fp);
    view_int();
}

/* Get file name process */
Get_name()
{  Blocks(100,100,500,140);
```

```
    outtextxy(110,117,"Enter file name :");
    gotoxy(32,8);scanf("%s",name);
    Blockss(100,100,500,140);
}


/* Draw Data Entry Window Frame */
Blocks(int x1,int y1,int x2,int y2)
{  setcolor(C_Line);
   rectangle(x1,y1,x2,y2);
   rectangle(x1-3,y1-3,x2+3,y2+3);
   setfillstyle(1,C_Block);
   floodfill(x1+1,y1+1,C_Line);
}


/* DEL. Data Entry Window Frame */
Blockss(int x1,int y1,int x2,int y2)
{  setcolor(C_Back);
   rectangle(x1,y1,x2,y2);
   rectangle(x1-3,y1-3,x2+3,y2+3);
   setfillstyle(1,C_Back);
   floodfill(x1+1,y1+1,C_Back);
}

/* Read data from RS-232C */
Rport()
{ int n;
  Start=time('\0');
  for(n=0;n<Size;n++) Buffer[n] = rport(PORT);
  End=time('\0');
  Diff = difftime(End,Start);
  outtextxy(50,42," Rx Time: ");
  Score(Diff);
  for(n=0;n<Size;n++) Data[n]=Buffer[n];
  return;
}


/* Read Char from RS-232 */
rport(int Port)
{ union REGS r;
  while(!(Status(PORT) & 256 ))  /* Check New Data Recieve */
  if (kbhit())
   { Rate=0;
     main();
   }
  r.x.dx = Port;
  r.h.ah = 2;       /* No. interupt */
  int86(0x14,&r,&r);
  if(r.h.ah & 128)
  {setcolor(C_Etext);
   outtextxy(60,120,"read error detected in serial port");
   getch();
   setcolor(C_Back);
   outtextxy(60,120,"read error detected in serial port");
   }
  return r.h.al;
}

/* Send Char */
sport(int Port,int Sdata)
{ union REGS r;
  r.x.dx = Port;
  r.h.al = Sdata;       /* Data */
```

```
  r.h.ah = 1;       /* No. INT. */

  int86(0x14,&r,&r);
  if(r.h.ah & 128)
       { setcolor(C_Etext);
         Sound();
         outtextxy(60,120,"Send error detected in serial port");
         getch();
         setcolor(C_Back);
         outtextxy(60,120,"Send error detected in serial port");
         exit(1);
                  }
}


/* Check Status port when read data */
/* value in r.x.ax
=========================================
   Meaning ("1")                        bits
   Data Ready                            0
   Overrun Error                         1
   Parity Error                          2
   Framing Error                         3
   Break-Detect Error                    4
   Transfer hold-register empty          5
   Transfer shift-register empty         6
   Time-out Error                        7 */

Status(int Port)
{ union REGS r;
  r.x.dx = Port;
  r.h.ah = 3;
  int86(0x14,&r,&r);
  return r.x.ax;
}


/*================================================
    Vaule in  r.h.al(Initial Port) --> Line Status
================================================
    bit            meaning
   7,6,5         000 = 110  baud
                 001 = 150  baud
                 010 = 300  baud
                 011 = 600  baud
                 100 = 1200 baud
                 101 = 2400 baud
                 110 = 4800 baud
                 111 = 9600 baud
   4,3           00,10 = No   parity
                 01    = Odd  parity
                 11    = Even parity
   2             0   = 1 Stop bits
                 1   = 2 Stop bits
   1,0           10  = 7 Bits data
                 11  = 8 Bits data
   ================================
   bit    7  6  5  4  3  2  1  0
   code   1  1  1  0  0  0  1  1    = 0xe3 = 227
================================================*/

port_init(int Port,int code)
{ union REGS r;
  r.x.dx = Port;  /* No. Port */
  r.h.ah = 0;
  r.h.al = code;
```

```
  int86(0x14,&r,&r);
  return;
}


/* Draw Graph & Calculate rate beat */
view_int()
{
  int x,y1,n,m,max,Slope,k[2];
  time_t Start,End;
  float Diff1,Diff2,Time,Rates;

  n = 0;
  Slope = 0;
  k[0] = 0 ;
  k[1] = 0 ;

  /* ============================================
       PRIORITY SYMBOL
     ============================================
     1. ()
     2. !,++,-- [From right to left]
     3. *,/,%
     4. +,-
     5. <,<=,>,>=
     6. ==,!=
     7. &&
     8. ||
     ============================================ */


  for (max=0;max<2;)

  { if( Data[n]<Data[n+1] && Data[n+1]>=Data[n+2] && Data[n+1]>=Peak
  && Slope==0 && n<Size-3) /* Used n,n+1,n+2 will be protect transient */
      { k[max]=n+1;
        max++;
        Slope=1;     /* Negative Slope */
      }

    if(Data[n]>Data[n+1] && Data[n+1]>Data[n+2] && Data[n]>=Peak
       && Slope==1 && n<Size-3) Slope=0;
    n++;
    if (n>Size-1) max=3;
  }
  /* 400 cycle/second = Time 2.5 msec */
  Time = 0.0025*(k[1]-k[0]);
  if(k[1]==0) Time = 0;
  /* Rate = Cycle per minute */
  if(Time!=0)  /* If Time != 0 Calculate Approximation int value */
    { Rates = 60/Time;
      Rate = Rates;
      if((Rates-Rate)>0.5) Rate=Rate+1;
     }
  else Rate = 0;
  k[1] = k[0] = 0;

  x=D_Left;
  for (n=0;n<Size-fs_data;n=n+fs_data)
  { if(Data[n]>Bottom-Top) Data[n]=Bottom-Top;
    if(Data[n+fs_data]>Bottom-Top) Data[n+fs_data]=Bottom-Top;
    if(x>D_Right) x=D_Left;
    if(x==D_Left)
      {setcolor(C_Back);
       line (x,40,x,Bottom);
      }
```

```
    setcolor(C_Back);
    for(m=0;m<xplot;m++)
    line(x+1+m,40,x+1+m,Bottom);
    setcolor(C_Plot);
    line(x,Bottom-Data[n],x+xplot,Bottom-Data[n+fs_data]);
/*  line(x,Bottom-Data[n]-1,x+xplot,Bottom-Data[n+1]-1); Double Line*/
    x=x+xplot;
  }
  setcolor(C_Detect);
  line(D_Left,Bottom-Peak,D_Right,Bottom-Peak);
  Score(4,Bottom-Peak+2,Peak);
  outtextxy(D_Left+2,Bottom-Peak+2,"Detect Level");
  Disp_CPM();
/*  for (n=75;n<=(D_Right-D_Left);n=n+75)
  line(D_Left+n,Bottom+1,D_Left+n,Top+1);  /* Time Scale Devider      */
  return;
}


Scale()
{ int n;
  char S1[7][7];
  setcolor(C_Scale);
  line(D_Left-1,Bottom+1,D_Right+1,Bottom+1); /* Time Scale Line1     */
  line(D_Left-1,Bottom+2,D_Right+1,Bottom+2); /* Time Scale Line2     */
  line(D_Left-1,Bottom+1,D_Left-1,Top-1);     /* Vertical Scale Line1 */
  line(D_Left-2,Bottom+2,D_Left-2,Top-1);     /* Vertical Scale Line2 */
  setcolor(C_Scales);
  for (n=20;n<(Bottom-Top);n=n+20)
  line(D_Left-4,Bottom-n,D_Left-1,Bottom-n);  /* Vertical Scale Devider */
  outtextxy(D_Left-26,Top,"ECG");
  /* 1 Display = 1.5 Sec = 600 data = 600 dot */
  /* 75 = 600dot/8cell(10 scale point)     */
  for (n=75;n<=(D_Right-D_Left);n=n+75)
  line(D_Left+n,Bottom+5,D_Left+n,Bottom+1);  /* Time Scale Devider     */
  for (n=0;n<7;n++)                          /* Write Time Scale value */
  {  sprintf(S1[n],"%.2fS",(n+1)*0.1875*fs_data); /* Save float in text */
     outtextxy(D_Left+75*(n+1)-20,Bottom+8,S1[n]);/* Show float in text */
  }
  outtextxy(D_Right-350,Bottom+18,"Time Scale");
  return;
 }


/*=======================================
EXAMPLE CHECK TIME LOOP
Time()
{ time_t Start,End;
  float Diff;

  Start=time('\0');
  End=time('\0');
  Diff = difftime(End,Start);
  printf("\nTime: %.2f",Diff);
  }
======================================= */

/* Show int in graphic mode  */
int Score(int sx,int sy,int number)
{
   char str[10];
   sprintf(str,"%d",number);
   outtextxy(sx,sy,str);
   return;
}
```

```
DATA_DOT()
{ int n,in;
  char name[12];
  n = fs_data;

  sprintf(name,"DATA/DOT : %d",fs_data);
  do { Bar3d(2,32);
       Text_bar(2,32,COLORS[C_Mtext],name);
       in=bioskey(0);  /* Read Key */
       switch(in)      /* Detect Key */
                  { case Down:   if(n==1) n=4;
                                 else n--;
                                 fs_data=n;
                                 sprintf(name,"DATA/DOT : %d",fs_data);
                                 break;
                    case Up:     if(n==4) n=1;
                                 else n++;
                                 fs_data=n;
                                 sprintf(name,"DATA/DOT : %d",fs_data);
                                 break;
                    case Enter:  Bbar3d(2,32);
                                 break;
                  }
     } while(in!=Enter); /* Return command code when key = 'ENTER' */
   main();
}

DETC_LEV()
{ int n,in;
  char name[19];
  n = Peak;

  sprintf(name,"Detect Level : %d",Peak);
  do { Bar3d(2,32);
       Text_bar(2,32,COLORS[C_Mtext],name);
       in=bioskey(0);  /* Read Key */
       switch(in)      /* Detect Key */
                  { case Down:   if(n==1) n=255;
                                 else n--;
                                 Peak=n;
                                 sprintf(name,"Detect Level : %d",Peak);
                                 break;
                    case Up:     if(n==255) n=1;
                                 else n++;
                                 Peak=n;
                                 sprintf(name,"Detect Level : %d",Peak);
                                 break;
                    case Enter:  Bbar3d(2,32);
                                 break;
                  }
     } while(in!=Enter); /* Return command code when key = 'ENTER' */
 main();
}


SERIAL_X()
{ int n,in;
  char name[12];
  n = PORT+1;

  sprintf(name,"Serial Port: %d",PORT+1);
  do { Bar3d(2,32);
       Text_bar(2,32,COLORS[C_Mtext],name);
       in=bioskey(0);  /* Read Key */
       switch(in)      /* Detect Key */
                  { case Down:   if(n==1) n=2;
```

```
                                else n--;
                                PORT=n-1;
                                sprintf(name,"Serial Port: %d",PORT+1);
                                break;
                    case Up:    if(n==2) n=1;
                                else n++;
                                PORT=n-1;
                                sprintf(name,"Serial Port: %d",PORT+1);
                                break;
                    case Enter: Bbar3d(2,32);
                                break;
                  }
      } while(in!=Enter); /* Return command code when key = 'ENTER' */
 main();
}

Display()
{ Disp_Serial();
  Disp_Dot();
  Disp_Detc();
  Disp_CPM();
}

Disp_Serial()
{ Bar3d(0,449);
  setcolor(C_Menu+8);
  sprintf(Oldv,"Serial Port : ");
  outtextxy(10,460,Oldv);
  sprintf(Oldv,"Serial Port : %d",PORT+1);
  setcolor(C_Back);
  outtextxy(10,460,Oldv);
}

Disp_Dot()
{ Bar3d(1,449);
  sprintf(Oldv,"Data/Dot : %d",fs_data);
  setcolor(C_Back);
  outtextxy(168,460,Oldv);
}

Disp_Detc()
{ Bar3d(2,449);
  sprintf(Oldv,"Detect Level : ");
  setcolor(C_Menu+8);
  outtextxy(326,460,Oldv);
  sprintf(Oldv,"Detect Level : %d",Peak);
  setcolor(C_Back);
  outtextxy(326,460,Oldv);
}

Disp_CPM()
{ Bar3d(3,449);
  sprintf(Oldv,"CPM : ");
  setcolor(C_Menu+8);
  outtextxy(484,460,Oldv);
  if(Rate==0)
   { sprintf(Oldv,"CPM : Below 20");
     setcolor(LIGHTRED);
     Sound();
   }
  else if(Rate==1) sprintf(Oldv," ");
       else { sprintf(Oldv,"CPM : %d",Rate);
              setcolor(C_Back);
              }
  outtextxy(484,460,Oldv);
```

```
}


REPLOT()
{ view_int();
  return;
}


SET_COLOR1()
{ int n,color,in;
  char name[12];
  n =0;

  sprintf(name,"C_Scale");
  color=C_Scale;
  do { Bar3d(2,32);
       Text_bar(2,32,COLORS[C_Mtext],name);
       Bar3d(3,32);
       Text_bar(3,32,COLORS[color],TCOLORS[color]);
       in=bioskey(0);  /* Read Key */
       switch(in)      /* Detect Key */
                 { case Left:   if(color==0) color=15;
                                else color--;
                                Case_color(n,color);
                                break;
                   case Right:  if(color==15) color=0;
                                else color++;
                                Case_color(n,color);
                                break;
                   case Up:     if(n==0) n=8;
                                else n--;
                                 switch(n)
                                { case 0: color=C_Scale;
                                          sprintf(name,"C_Scale");
                                          break;
                                  case 1: color=C_Scales;
                                          sprintf(name,"C_Scales");
                                          break;
                                  case 2: color=C_Plot;
                                          sprintf(name,"C_Plot");
                                          break;
                                  case 3: color=C_Detect;
                                          sprintf(name,"C_Detect");
                                          break;
                                  case 4: color=C_Text;
                                          sprintf(name,"C_Text");
                                          break;
                                  case 5: color=C_Line;
                                          sprintf(name,"C_Line");
                                          break;
                                  case 6: color=C_Block;
                                          sprintf(name,"C_Block");
                                          break;
                                  case 7: color=C_Etext;
                                          sprintf(name,"C_Etext");
                                          break;
                                  case 8: SET_COLOR();
                                          break;
                                 }
                                break;
                   case Down:   if (n==8) n=0;
                                else n++;
                                 switch(n)
                                { case 0: color=C_Scale;
                                          sprintf(name,"C_Scale");
```

```
                                        break;
                                case 1: color=C_Scales;
                                        sprintf(name,"C_Scales");
                                        break;
                                case 2: color=C_Plot;
                                        sprintf(name,"C_Plot");
                                        break;
                                case 3: color=C_Detect;
                                        sprintf(name,"C_Detect");
                                        break;
                                case 4: color=C_Text;
                                        sprintf(name,"C_Text");
                                        break;
                                case 5: color=C_Line;
                                        sprintf(name,"C_Line");
                                        break;
                                case 6: color=C_Block;
                                        sprintf(name,"C_Block");
                                        break;
                                case 7: color=C_Etext;
                                        sprintf(name,"C_Etext");
                                        break;
                                case 8: SET_COLOR();
                                        break;
                                }
                              break;
                    case Enter: Bbar3d(2,32);
                                Bbar3d(3,32);
                                break;
                  }
      } while(in!=Enter); /* Return command code when key = 'ENTER' */
   main() ;
}

Case_color(int n,int color)
{ switch(n)
 { case 0: C_Scale=color;
           break;
   case 1: C_Scales=color;
           break;
   case 2: C_Plot=color;
           break;
   case 3: C_Detect=color;
           break;
   case 4: C_Text=color;
           break;
   case 5: C_Line=color;
           break;
   case 6: C_Block=color;
           break;
   case 7: C_Etext=color;
           break;
  }
  return;
}


print_scr(int startx,int starty,int endx,int endy,int pagex,int pagey,int density)
{ register int i,x,y,px;
  int cols,color,sum;

  endx++; endy++;
  cols = endx - startx;
  for(; pagey>=0; pagey--) print('\n');
  for(y = starty; y<endy; y+=8 )
   { for( px=0; px<pagex; px++) print(' ');
```

```
    set_graphics(cols,density);
    for(x=startx; x<endx; x++)
     { sum=0;
       for(i=0; i<8; i++)
         { if(y+i<endy) { color = getpoint(x,y+i);
                          if(color) sum += 1<<(7-i);
                        }
          }
        print(sum);
       }
    print('\n');
   }
}

set_graphics(int cols,int density)
{ union { unsigned char c[2];
          unsigned int i;
        } u;
  char den_code;

  u.i=cols;
  switch(density)
  { case SINGLE: den_code = 75;
                 break;
    case DOUBLE: den_code = 76;
                 break;
    case QUAD:   den_code = 90;
                 break;
  }
  print(27);print(51);print(22); /* 22/256 line spacing */
  print(27);print(den_code);print(u.c[0]);print(u.c[1]);
}

print(char ch)
{
  biosprint(0,ch,0);
}

getpoint(int x,int y)
{ union REGS r;

  r.h.ah = 13;
  r.x.dx = y;
  r.x.cx = x;
  r.h.bh = 0;
  int86(16,&r,&r);
  return r.h.al;
}
```

```
/* ================= Main_menu + 4 Sub menu  ====================*/
/*           Select by High Light & Set Display Color            */


#define main_menu       0
#define sub_menu        1

/* Main_mane  */
#define File            0
#define View            1
#define Setup           2
#define Exit            3

/* Sub_menu File */
#define Open            0
#define Save            1
#define Print           2
#define Replot          3

/* Sub_menu Setup */
#define Data_dot        0
#define Detc_lev        1
#define Serial_x        2
#define Set_color       3

/* Define Key for bioskey() */
#define Left    0x4b00
#define Right   0x4d00
#define Down    0x5000
#define Up      0x4800
#define Enter   0x1c0d
#define ESC     0x011b
#define Alt_f   0x2100
#define Alt_s   0x1f00
#define Alt_v   0x2f00
#define Alt_e   0x1200
#define F1      0x3b00
#define F2      0x3c00
#define F3      0x3d00
#define F4      0x3e00
#define F5      0x3f00
#define F6      0x4000
#define F7      0x4100
#define F8      0x4200
#define F9      0x4300

#define Black           0
#define Blue            1
#define Green           2
#define Cyan            3
#define Red             4
#define Magenta         5
#define Brown           6
#define Linhtgray       7
#define Darkgray        8
#define Lightblue       9
#define Lightgreen      10
#define Lightcyan       11
#define Lightred        12
#define Lightmagenta    13
#define Yellow          14
#define White           15

/*================================================
                COLOR IN TURBO C
```

```
  ============================================
<<DARK COLORS>>
    BLACK(0)     BLUE(1)          GREEN(2)
    CYAN(3)      RED(4)           MAGENTA(5)
    BROWN(6)     LIGHTGRAY(7)     DARKGRAY(8)

<<LIGHT COLORS>>
    LIGHTBLUE(9)LIGHTGREEN(10)  LIGHTCYAN(11)
    LIGHTRED(12)LIGHTMAGENTA(13)YELLOW(14)
    WHITE(15)    BLINK(128)
  ============================================== */

#include<dos.h>
#include<stdio.h>
#include<graphics.h>

int  key         = 0 ;                    /* Set Hot Key  */
int  key_arrow[2]={0,0};                    /* Set new menu */
int  key_old[2]  ={0,0};                    /* Set old menu */
char main_text[4][20];
char sub_text[2][4][20];
char *TCOLORS[]={"BLACK(0)"    ,"BLUE(1)"         ,"GREEN(2)",
                 "CYAN(3)"     ,"RED(4)"          ,"MAGENTA(5)",
                 "BROWN(6)"    ,"LIGHTGRAY(7)"    ,"DARKGRAY(8)",
                 "LIGHTBLUE(9)","LIGHTGREEN(10)"  ,"LIGHTCYAN(11)",
                 "LIGHTRED(12)","LIGHTMAGENTA(13)","YELLOW(14)",
                 "WHITE(15)"   ,"BLINK(128)"};
int COLORS[]= {0,1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,128};

/* Define Color  */
int     C_Back  =       Blue ;/* Back Ground  Color                    */
int     C_Menu  =       Green ;/* Title Bar Menu [have Light Color]  */
int     C_Bar   =       Yellow;/* High Light Bar Color                */
int     C_Mtext =       Black ;/* Text in Menu  Color                 */
int     C_Line  =       Yellow;/* Frame Window  Color                 */
#define C_Logo          LIGHTGRAY/* Back ground Logo Color           */

/*================================================
                    MAIN PROGRAM
================================================*/


Main()
{

  Open_graph();
  Set_text();
  Menu(); /* main_menu have 1 set menu */
  Bar3d(0,449);
  Bar3d(1,449);
  Bar3d(2,449);
  Bar3d(3,449);
  settextstyle(0,1,0);
  setcolor(WHITE);
  outtextxy(100,200,"Test TEXT");
  settextstyle(0,0,0);
  Read_key();
  Logo();
  closegraph();
  printf("\n Good BYE.");
  exit(0);
}

Read_key()
{
```

```
Key:  /* Start Menu */
key = bioskey(0);                /* Receive Hot key  */
switch(key)
{
  case Alt_f: key_arrow[main_menu] = File;
            /* Show_menu(main or sub,sub_name,Lim_menu);  */
              key_arrow[sub_menu] = Show_menu(sub_menu,File,3);
              break;
  case Alt_v: key_arrow[main_menu] = View;
              key_arrow[sub_menu]  = 0;      /* Don't have sub */
              break;
  case Alt_s: key_arrow[main_menu] = Setup;
              key_arrow[sub_menu] = Show_menu(sub_menu,Setup-1,3);
              break;
  case Alt_e: key_arrow[main_menu] = Exit;
              key_arrow[sub_menu]  = 0;      /* Don't have sub */
              break;
  case F3:    key_arrow[main_menu] = File;
              key_arrow[sub_menu]  = Open;
              break;
  case F2:    key_arrow[main_menu] = File;      /* Key = F2 Save */
              key_arrow[sub_menu]  = Save;
              break;
  case F4:    key_arrow[main_menu] = File;      /* Key = F4 Print*/
              key_arrow[sub_menu]  = Print;
              break;
  case F9:    key_arrow[main_menu] = File;      /* Key = F4 Print*/
              key_arrow[sub_menu]  = Replot;
              break;
  case F5:    key_arrow[main_menu] = Setup;     /* Key = F5 Data/dot*/
              key_arrow[sub_menu]  = Data_dot;
              break;
  case F6:    key_arrow[main_menu] = Setup;     /* Key = F6 Detect level*/
              key_arrow[sub_menu]  = Detc_lev;
              break;
  case F7:    key_arrow[main_menu] = Setup;     /* Key = F7 Serial */
              key_arrow[sub_menu]  = Serial_x;
              break;
  case F8:    key_arrow[main_menu] = Setup;     /* Key = F5 Data/dot*/
              key_arrow[sub_menu]  = Set_color;
              break;
  case ESC:   Esc:
              key_arrow[main_menu]=Show_menu(main_menu,main_menu,3);
              switch(key_arrow[main_menu])
               { case 0:key_arrow[sub_menu]=Show_menu(sub_menu,File,3);
                        break;
                 case 2:key_arrow[sub_menu]=Show_menu(sub_menu,Setup-1,3);
                        break;
                }
              break;
  default:    goto Key;
}
if(key_arrow[main_menu]==9)
      { Menu();
        goto Key;
       }
if(key_arrow[sub_menu] ==9)
      { key_arrow[sub_menu] =0;
        goto Esc;
      }
switch(key_arrow[main_menu])
{ case File: key_old[main_menu] = File;
             switch(key_arrow[sub_menu])
              { case Open: key_old[sub_menu] = Open;
```

```
                                OPEN();
                                break;
                        case Save: key_old[sub_menu] = Save;
                                SAVE();
                                break;
                        case Print: key_old[sub_menu]= Print;
                                print_scr( 0,0,640,480,1,1,1);
                                break;
                        case Replot: key_old[sub_menu] = Replot;
                                REPLOT();
                                break;
                        default:    break;
                        }
                    goto Key;
        case View: key_old[main_menu] = View;
                    key_old[sub_menu]  = 0;
                    VIEW();
                    goto Key;
        case Setup: key_old[main_menu] = Setup;
                    switch(key_arrow[sub_menu])
                    { case Data_dot: key_old[sub_menu] = Data_dot;
                                DATA_DOT();
                                break;
                      case Detc_lev: key_old[sub_menu] = Detc_lev;
                                DETC_LEV();
                                break;
                      case Serial_x: key_old[sub_menu]= Serial_x;
                                SERIAL_X();
                                break;
                      case Set_color:SET_COLOR();
                                break;
                      default:  break;
                      }
                    goto Key;
      /* Exit Programs */
      case Exit:  key_old[main_menu] = Exit;
                  key_old[sub_menu] = 0;
                  break;
      /* Exit Menu by 'ESC' */
      case 9: goto Key;
     default: goto Key;
    }
  return ;
  }

  /* ================================================
                  END MAIN PROGRAM
     ================================================ */


  /* ================================================
                  FUNCTION PROGRAM
     ================================================ */


  /* Creat window menu */
  Menu()
  { Frame(2,31,635,477,C_Back);     /* Create window  */
    Mbar3d();
    setcolor(C_Menu);
    Text_menu(main_menu,main_menu);       /* Creat Text Munu */
    return ;
  }

  Del_menu()
  { int y;
```

```
  setcolor(C_Menu);
  for(y=3;y<30;y++)line(3,y,637,y);
  Mbar3d();
}


Logo()
{ Open_graph();
  Frame(2,2,635,477,C_Logo);
  setcolor(Magenta);
  settextstyle(1,0,2);
  outtextxy(250,50,"Base Analysis");
  setcolor(Lightgreen);
  settextstyle(1,0,1);
  outtextxy(95,180,"Advisor : Dr. Kittiphon");
  setcolor(C_Mtext);
  outtextxy(10,250,"Programs Design : Mr. Perapong Wattanagasamtum  No. 35.103194");
  outtextxy(10,300,"Circuits Design : Mr. Somkid   Lidkong          No. 35.103206");
  outtextxy(10,350,"                : Mr. Sidsadee Wongwangwit      No. 35.103211");
  setcolor(Magenta);
  outtextxy(50,400,"Electronics Engineering Department ");
  outtextxy(10,430,"Fucuty Of Engineering");
  outtextxy(30,450,"King Mongkut's Institute of Technology Ladkrabang.");
/*  ปโฎ ภาส่ภณพีะ สแดป.กพีรบลลร
  ปโฎ อรมโถ วนบโสฉ 35103206
  ปโฎ อวพัฑา สฉหะสะภฉกโล 35103211
  ฦโษโลฦะ กโฑฑโภศ */
  getch();
  return ;
}

Open_graph()
{ int driver,mode;
  driver = DETECT;
  mode = 0;                          /* 640X480 = 0 -> 639,0 -> 479 */
  initgraph(&driver,&mode," ");
 return ;
 }


/* Draw Frame Window */
Frame(int x1,int y1,int x2,int y2,char c_back)
{  /* Out size frame */
   setcolor(C_Line);
   rectangle(0,0,637,479);
   /* In size frame  */
   rectangle(x1,y1,x2,y2);
   setfillstyle(1,c_back);
   floodfill(3,33,C_Line);
   return;
}


/* Write Menu Text */
#define CELL    158
#define ORGT    15
#define ONE     ORGT+(0*CELL)
#define TWO     ORGT+(1*CELL)
#define TREE    ORGT+(2*CELL)
#define FOUR    ORGT+(3*CELL)
Text_menu(menu,sub_name)
{ int x;
  /*================================
            TEXT MENU
     1 Character Length 7 Pixcell
  ================================*/
```

```
  setcolor(C_Mtext);
  settextstyle(1,0,1);
  if(menu==main_menu)
  { outtextxy(ONE ,5,main_text[0]);
    outtextxy(TWO ,5,main_text[1]);
    outtextxy(TREE,5,main_text[2]);
    outtextxy(FOUR,5,main_text[3]);
   }
  else
  { outtextxy(ONE ,5,sub_text[sub_name][0]);
    outtextxy(TWO ,5,sub_text[sub_name][1]);
    outtextxy(TREE,5,sub_text[sub_name][2]);
    outtextxy(FOUR,5,sub_text[sub_name][3]);
   }
  settextstyle(0,0,1);
  return;

}

Set_text()
{
  sprintf(main_text[0],"File[Alt-f]");
  sprintf(main_text[1],"View[Alt-v]");
  sprintf(main_text[2],"Setup[Alt-s]");
  sprintf(main_text[3],"Exit[Alt-e]");

  sprintf(sub_text[0][0],"Open[F3]");
  sprintf(sub_text[0][1],"Save[F2]");
  sprintf(sub_text[0][2],"Print[F4]");
  sprintf(sub_text[0][3],"Replot[F9]");

  sprintf(sub_text[1][0],"Data/Dot[F5]");
  sprintf(sub_text[1][1],"Detc lev[F6]");
  sprintf(sub_text[1][2],"Serial_x[F7]");
  sprintf(sub_text[1][3],"Set Color[F8]");

}


/* Draw Menu Hiht Light */
Bar(int x)
{
  Mbar(x,C_Bar);
  return;
}

/* DEL. Menu High Light */
Bbar(int x)
{
  Mbar(x,C_Menu+8);
  return;
}

Mbar(int x,char b_color)
{ int X;
  X = 1+(x*CELL);
  setcolor(b_color);
  rectangle(X+6,7,X+CELL-4,24);
  setfillstyle(1,b_color);
  floodfill(X+10,8,b_color);
  return;
}

Mbar3d()
```

```
{ Bar3d(0,2);
  Bar3d(1,2);
  Bar3d(2,2);
  Bar3d(3,2);
}

Text_bar(int x,int y,int color,char str[18])
{ int X,C_Lmenu;
  X= 2+(x*CELL);
  C_Lmenu = 8+C_Menu;
  setcolor(C_Lmenu);
  outtextxy(X+6,y+11,"                  ");
  setcolor(color);
  outtextxy(X+6,y+11,str);
  }


Bar3d(int x,int y)
{ int X,C_Lmenu;
  X= 2+(x*CELL);
  setcolor(C_Menu);
  rectangle(X,y,X+CELL,y+27);
  setfillstyle(1,C_Menu);
  floodfill(X+1,y+1,C_Menu);
  C_Lmenu = 8+C_Menu;
  setcolor(C_Lmenu);
  rectangle(X+4,y+4,X+CELL-4,y+23);
  setfillstyle(1,C_Lmenu);
  floodfill(X+5,y+5,C_Lmenu);
  line(X,y,X+4,y+4);
  line(X+CELL,y,X+CELL-4,y+4);
  line(X,y+27,X+4,y+23);
  line(X+CELL,y+27,X+CELL-4,y+23);
}

Bbar3d(int x,int y)
{ int X,n;
  X= 3+(x*CELL);
  setcolor(C_Back);
  for(n=0;n<CELL;n++,X++)line(X,y,X,y+27);
}

/* Keep Arrow Key Code */
Show_menu(menu,sub_name,lim_menu)
{ int n,in;
  if(menu==main_menu) n=key_old[menu];
  else n=key_old[menu];
  Del_menu();
  Bar(n);
  Text_menu(menu,sub_name);


  do { in=bioskey(0);  /* Read Key */
       switch(in)      /* Detect Key */
                  { case Left:   if(n==0) n=lim_menu; else n--;
                                 if(n==lim_menu) Bbar(0); else Bbar(n+1);
                                 Bar(n);
                                 Text_menu(menu,sub_name);
                                 break;
                    case Right:  if(n==lim_menu) n=0; else n++;
                                 if(n==0) Bbar(lim_menu); else Bbar(n-1);
                                 Bar(n);
                                 Text_menu(menu,sub_name);
                                 break;
                    case Enter:  Bbar(n);
```

```
                                if(menu!=main_menu)
                                { Del_menu();
                                  Text_menu(main_menu,main_menu);
                                 }
                                else Text_menu(main_menu,main_menu);
                                break;
                    case ESC:   Bbar(n);
                                n=9;   /* Don't active command when return */
                                in = Enter; /* Set key = 'ENTER' */
                                break;
                  }
      } while(in!=Enter); /* Return command code when key = 'ENTER' */
   return n;
}


SET_COLOR()
{ int n,color,in;
  char name[12];
  color = C_Back;
  n=4;

  sprintf(name,"C_Back");
  do { Bar3d(2,32);
       Text_bar(2,32,COLORS[C_Mtext],name);
       Bar3d(3,32);
       Text_bar(3,32,COLORS[color],TCOLORS[color]);
       in=bioskey(0);  /* Read Key */
       switch(in)      /* Detect Key */
                  { case Left:   if(color==0) color=15;
                                 else color--;
                                 switch(n)
                                 { case 4: C_Back=color;
                                           break;
                                   case 1: C_Menu=color;
                                           break;
                                   case 2: C_Bar=color;
                                           break;
                                   case 3: C_Mtext=color;
                                           break;
                                 }
                                 break;
                    case Right:  if(co
```

```
int timer;
}

Chkspeed()
{ timer=Time();
}

Sound()
{ tone(349,timer);
  tone(745,timer);
}

#define TIMERMODE   182             /* Code to put time in right mode */
#define FREQSCALE   1190000L        /* basic time freq. in hertz      */
#define TIMESCALE   1230L           /* number of counts in 0.1 second */
#define T_MODEPORT  67              /* port control timer mode        */
#define FREQPORT    66              /* port control tone  freq.       */
#define BEEPPORT    97              /* port control speaker           */
#define ON          79              /* turn on speaker                */

tone(int freq,int time)
{ int hibyt,lobyt,port;
  long i,count,divisor;

  if (freq==1)
   { port = inportb(BEEPPORT);
     count = TIMESCALE*time;
     outportb(BEEPPORT,port);    /* Turn OFF Speaker & Restore Status */
     for (i=0;i<count;i++);      /* Delay time */
     goto OFF;
   }
  divisor = FREQSCALE/freq;
  lobyt = divisor % 256;      /* = divisor & 255 */
  hibyt = divisor / 256;      /* = divisor >> 8  */
  count = TIMESCALE*time;
  outportb(T_MODEPORT,TIMERMODE);
  outportb(FREQPORT,lobyt);
  outportb(FREQPORT,hibyt);
  port = inportb(BEEPPORT); /* Save Status port Before Used */
  outportb(BEEPPORT,ON);      /* Turn ON Speaker */
  for (i=0;i<count;i++);      /* Delay time */
  OFF:
  outportb(BEEPPORT,port);   /* Turn OFF Speaker & Restore Status */
}

/* Check Speed CPU */
Time()
{ time_t Start,End;
  float Diff;
  int diff,num = 0;
  Start=time('\0');
  do { num++;
       End  = time('\0');
       Diff = difftime(End,Start);
       diff = Diff;
     } while(diff<1);
  num = num/4;
  printf("\n Time %d : %d ",diff,num);
  return num;
```