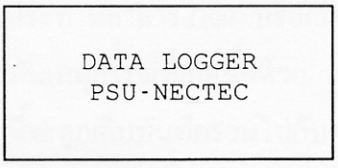
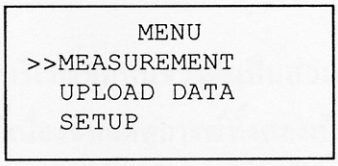
ภาคผนวก ก การใช้งานเครื่องวัดและบันทึกค่าพีเอช

การทำงานของเครื่องวัดและบันทึกค่าพีเอช เมื่อผู้ใช้กดสวิตช์เปิดเครื่อง หน้าจอแสดงผลจะแสดงข้อความ ดังภาพประกอบ ก-1 ประมาณ 1-2 วินาที หลังจากนั้นจะแสดงเมนูให้ผู้ใช้เลือกฟังก์ชันการทำงาน ดังภาพประกอบ ก-2 ซึ่งผู้ใช้สามารถเลือกฟังก์ชันการทำงานโดยกดคีย์ Up หรือ Down โดยให้ลูกศรตัวเลือกฟังก์ชันการทำงานชี้ไปที่ฟังก์ชันที่ต้องการ แล้วกดคีย์ Enter เครื่องจะเข้าสู่ฟังก์ชันการทำงานที่ผู้ใช้เลือก โดยรายละเอียดการทำงานในแต่ละฟังก์ชันของเครื่องมีดังต่อไปนี้

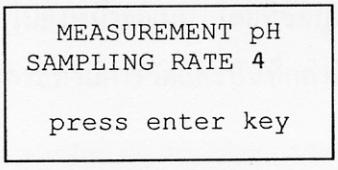


ภาพประกอบ ก-1

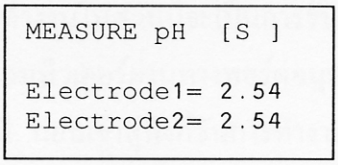


ภาพประกอบ ก-2

- MEASUREMENT ฟังก์ชันการทำงานในส่วนนี้จะเป็นส่วนของการวัดและบันทึกค่าพีเอช ซึ่งก่อนที่จะใช้งานฟังก์ชันนี้นั้น ผู้ใช้จะต้องทำการกำหนดค่าเริ่มต้นให้ตัวเครื่องก่อน (อยู่ในฟังก์ชันการทำงาน Setup) เมื่อเข้าสู่การทำงานในฟังก์ชันนี้ หน้าจอแสดงผลจะแสดงข้อความฟังก์ชันการทำงานและอัตราการสุ่มที่ผู้ใช้ได้กำหนดไว้ดังภาพประกอบ ก-3 ให้ผู้ใช้กดคีย์ Enter เครื่องจะเริ่มการวัดและบันทึกค่าพีเอชตามอัตราการสุ่มที่ได้กำหนดไว้ ซึ่งในระหว่างการวัดและบันทึกค่าพีเอช หน้าจอแสดงผลจะแสดงค่าพีเอชทั้งสองอิเลคโทรดที่วัดได้ในขณะนั้น ดังภาพประกอบ ก-4



ภาพประกอบ ก-3

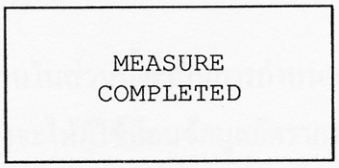


ภาพประกอบ ก-4

ในระหว่างการวัดและบันทึกค่าพีเอช หากผู้ใช้มีอาการเจ็บ (Painful), ไอ (Cough), อาเจียน (Vomit), นอน (Sleep), รับประทานอาหาร (Eat) หรือเกิดเหตุการณ์อื่นๆ (Misc.) ให้ผู้ใช้กดคีย์เหตุการณ์ที่ตัวเครื่องให้สัมพันธ์กับเหตุการณ์ที่เกิดขึ้นด้วย เช่นหากเกิดการไอขึ้นให้ผู้ใช้กดคีย์ Cough เป็นต้น ซึ่งเหตุการณ์ต่างๆนี้จะถูกเก็บบันทึกรวมไปกับค่าพีเอชที่วัดได้ลงในหน่วยความจำของเครื่องด้วย ในการกดคีย์เหตุการณ์แต่ละครั้งนั้น ตัวเครื่องจะเปิดไฟให้ความสว่างหน้าจอแสดงผลประมาณ 1 วินาที เพื่อบอกให้ผู้ใช้ทราบว่าคีย์เหตุการณ์ที่กดได้ถูกนำไปเก็บในหน่วยความจำแล้ว

ที่หน้าจอแสดงผลบริเวณมุมบนขวาจะเป็นส่วนที่แสดงสถานะของเหตุการณ์การนอนและรับประทานอาหาร ทั้งนี้เนื่องจากเหตุการณ์ทั้งสองเป็นเหตุการณ์ที่เกิดขึ้นเป็นระยะเวลานาน ดังนั้นเหตุการณ์สองจะมีตำแหน่งเริ่มต้นและตำแหน่งสิ้นสุดของการเกิดเหตุการณ์ ซึ่งสถานะที่แสดงบนหน้าจอบริเวณดังกล่าวจะบอกถึงการเริ่มต้นของเหตุการณ์นั้นๆแล้ว โดยที่สถานะ S จะหมายถึงเหตุการณ์การนอน และ E หมายถึงเหตุการณ์การรับประทานอาหาร เช่นเมื่อผู้ใช้เริ่มรับประทานอาหาร ให้ผู้ใช้กดคีย์ Eat ซึ่งเมื่อกดคีย์ดังกล่าว สถานะที่หน้าจอแสดงผลเป็น [E] ตัวอักษร E ที่แสดงจะเป็นตัวบอกว่าผู้ใช้อยู่ในระหว่างการรับประทานอาหาร และเมื่อผู้ใช้รับประทานอาหารเสร็จ ให้ผู้ใช้กดคีย์ Eat อีกครั้ง เพื่อบอกว่าได้รับประทานอาหารเสร็จแล้ว ตัวอักษร E ที่หน้าจอแสดงผลจะหายไป

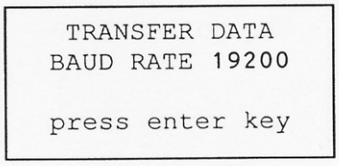
เมื่อตัวเครื่องเก็บบันทึกข้อมูลค่าพีเอชจนเต็มความจุหน่วยความจำแล้ว (ประมาณ 32 กิโลไบต์) หน้าจอแสดงผลจะแสดงข้อความเสร็จสิ้นการวัดและบันทึกค่าพีเอช ดังภาพประกอบ ก-5 ให้ผู้ใช้กดคีย์ Enter เพื่อกลับไปยังหน้าจอเมนู โดยที่ข้อมูลค่าพีเอชที่เก็บไว้ในหน่วยความจำของเครื่องนี้จะไม่มีการสูญหาย จนกระทั่งมีการวัดและบันทึกค่าพีเอชในครั้งถัดไป



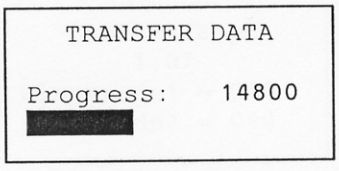
ภาพประกอบ ก-5

- Upload ฟังก์ชันการทำงานในส่วนนี้จะเป็นการรับส่งข้อมูลค่าพีเอชที่เก็บอยู่ในหน่วยความจำของตัวเครื่องไปยังเครื่องคอมพิวเตอร์ผ่านทางพอร์ตอนุกรมแบบ RS-232 ด้วยความเร็วในการรับส่งข้อมูล 19200 บิต/วินาที ก่อนเข้าสู่ฟังก์ชันการทำงานนี้ ผู้ใช้ต้องเชื่อมต่อตัวเครื่องและเครื่องคอมพิวเตอร์เข้าด้วยกันด้วยสาย RS-232 และเปิดโปรแกรมวิเคราะห์ค่าพีเอชบนเครื่องคอมพิวเตอร์ให้พร้อมสำหรับการรับส่งข้อมูลค่าพีเอช เมื่อเข้าสู่การทำงานในฟังก์ชันนี้ หน้าจอแสดงผลจะแสดงข้อความฟังก์ชันการทำงานและความเร็วในการรับส่งข้อมูล ดังภาพประกอบ ก-6) ให้ผู้ใช้กดคีย์ Enter เครื่องจะเริ่มส่งข้อมูลค่าพีเอชจากหน่วยความจำที่ได้เก็บบันทึกเอาไว้ไปยังเครื่องคอมพิงเตอร์ ซึ่งในระหว่างการรับส่งข้อมูลหน้าจอแสดงผลจะแสดงสถานะการรับส่งข้อมูล ดังภาพประกอบ ก-7

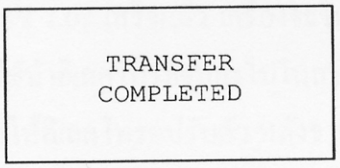
เมื่อเครื่องส่งข้อมูลจากหน่วยความจำของเครื่องหมดแล้ว หน้าจอแสดงผลจะแสดงข้อความเสร็จสิ้นการรับส่งข้อมูล ดังภาพประกอบ ก-8 ให้ผู้ใช้กดคีย์ Enter เพื่อกลับไปยังหน้าจอเมนู



ภาพประกอบ ก-6



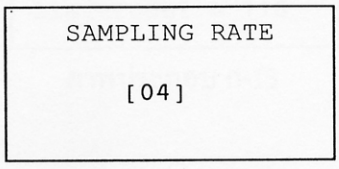
ภาพประกอบ ก-7



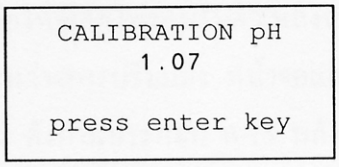
ภาพประกอบ ก-8

- Setup ฟังก์ชันการทำงานในส่วนนี้จะเป็นการกำหนดค่าเริ่มต้นต่างๆให้กับตัวเครื่อง เมื่อเข้าสู่การทำงานในฟังก์ชันนี้ เครื่องจะให้ผู้ใช้ป้อนข้อมูลอัตราการสุ่ม (หน่วยเป็นวินาที) ซึ่งหน้าจอแสดงผลจะแสดงดังภาพประกอบ ก-9 ผู้ใช้สามารถเปลี่ยนอัตราการสุ่มได้ด้วยการกดคีย์ Up หรือ

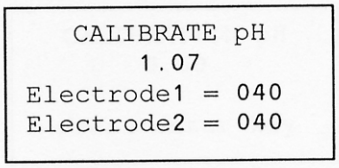
Down เมื่อเลือกค่าอัตราสุ่มที่ต้องการได้แล้วให้กดคีย์ Enter หลังจากนั้นเครื่องจะให้ผู้ใช้ทำการปรับแต่ง (Calibration) ค่าพีเอช 1.07 โดยหน้าจอแสดงผลจะแสดงดังภาพประกอบ ก-10 ซึ่งเมื่อถึงขั้นตอนนี้ให้ผู้ใช้นำอิเลคโทรดจุ่มลงไปในสารละลายพีเอชมาตรฐาน 1.07 เมื่อจุ่มแล้วให้รอประมาณ 3-5 นาที เพื่อให้อิเลคโทรดปรับตัว หลังจากนั้นให้กดคีย์ Enter เพื่อให้เครื่องเริ่มการปรับแต่งพีเอช 1.07 โดยในระหว่างการปรับแต่ง หน้าจอแสดงผลจะแสดงระดับค่าแรงดันของสารละลายพีเอชที่วัดได้ในขณะนั้น ดังภาพประกอบ ก-11



ภาพประกอบ ก-9

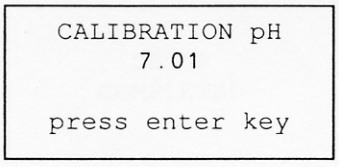


ภาพประกอบ ก-10

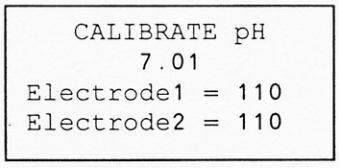


ภาพประกอบ ก-11

หลังจากปรับแต่งพีเอช 1.07 เสร็จแล้ว เครื่องจะให้ปรับแต่งพีเอช 7.01 ซึ่งจะแสดงหน้าจอดังภาพประกอบ ก-12 ให้ผู้ใช้นำอิเลคโทรดจุ่มลงไปในสารละลายพีเอชมาตรฐาน 7.01 เมื่อจุ่มแล้วให้รอประมาณ 3-5 นาที เพื่อให้อิเลคโทรดปรับตัว หลังจากนั้นให้กดคีย์ Enter เพื่อให้เครื่องเริ่มการปรับแต่งพีเอช 7.01 ในระหว่างการปรับแต่ง หน้าจอแสดงผลจะแสดงระดับค่าแรงดันของสารละลายพีเอชที่วัดได้ในขณะนั้น ดังภาพประกอบ ก-13

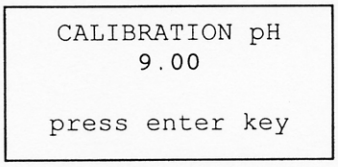


ภาพประกอบ ก-12

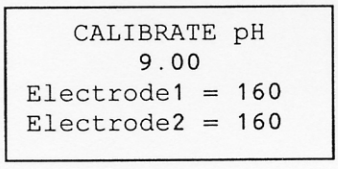


ภาพประกอบ ก-13

หลังจากปรับแต่งพีเอช 7.01 เสร็จแล้ว เครื่องจะให้ปรับแต่งพีเอช 9.00 ซึ่งจะแสดงหน้าจอดังภาพประกอบ ก-14 ให้ผู้ใช้นำอิเลคโทรดจุ่มลงไปในสารละลายพีเอชมาตรฐาน 9.00 เมื่อจุ่มแล้วให้รอประมาณ 3-5 นาที เพื่อให้อิเลคโทรดปรับตัว หลังจากนั้นให้กดคีย์ Enter เพื่อให้เครื่องเริ่มการปรับแต่งพีเอช 9.00 ในระหว่างการปรับแต่ง หน้าจอแสดงผลจะแสดงระดับค่าแรงดันของสารละลายพีเอชที่วัดได้ในขณะนั้น ดังภาพประกอบ ก-15 หลังจากปรับแต่งพีเอช 9.00 เสร็จแล้ว เครื่องจะแสดงข้อความเสร็จสิ้นการกำหนดค่าต่างๆ ดังภาพประกอบ ก-16 ให้ผู้ใช้กดคีย์ Enter เพื่อกลับไปยังหน้าจอเมนู



ภาพประกอบ ก-14

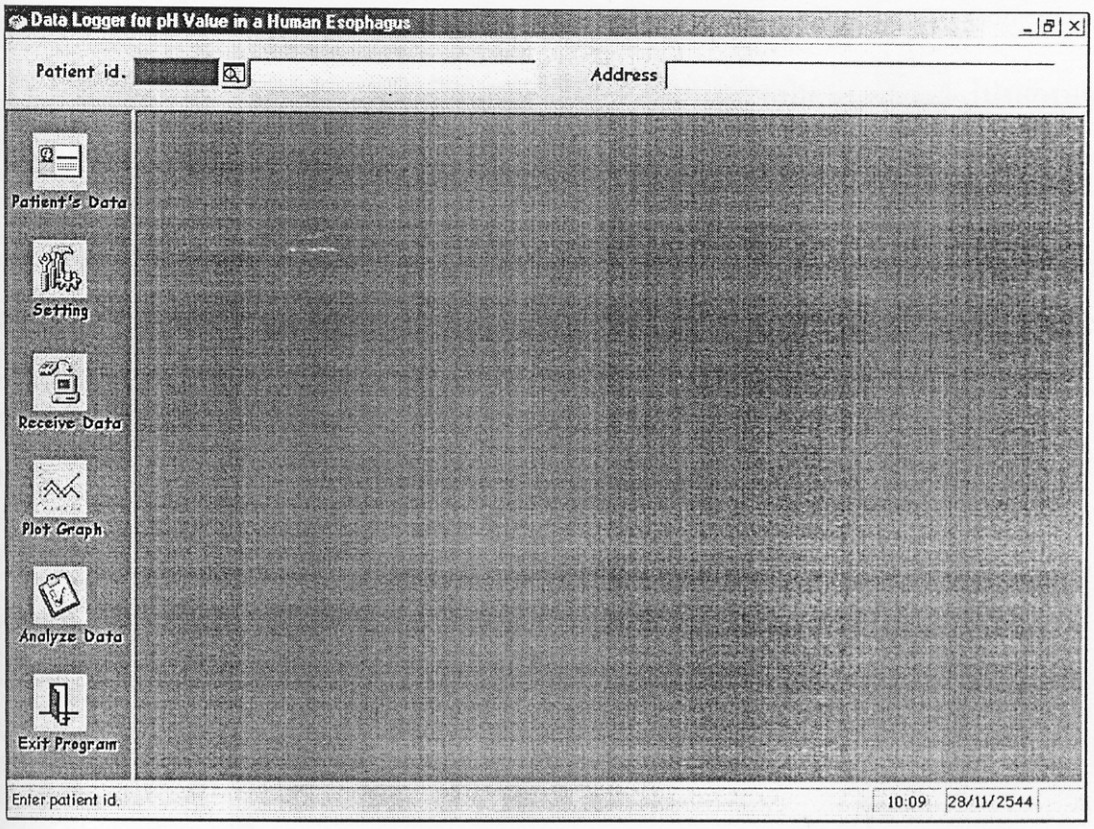


ภาพประกอบ ก-15

ภาพประกอบ ก-16

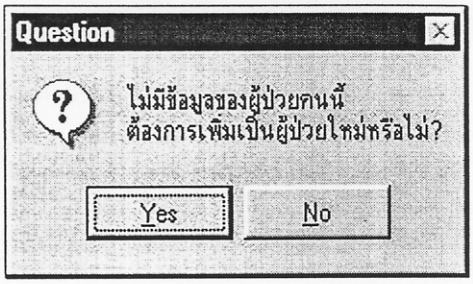
ภาคผนวก ข การใช้งานโปรแกรมวิเคราะห์ค่าพีเอช

การทำงานโปรแกรมวิเคราะห์ค่าพีเอช เมื่อผู้ใช้เปิดโปรแกรม โปรแกรมจะเข้าสู่หน้าจอหลัก ดังภาพประกอบ ข-1 ซึ่งในหน้าจอหลักนี้จะมีเมนูการทำงานต่างๆของโปรแกรม ได้แก่ ข้อมูลผู้ป่วย (Patient's Data), กำหนดค่า (Setting), รับข้อมูล (Receive Data), พล็อตกราฟ (Plot Graph), วิเคราะห์ข้อมูล (Analyze Data) และจบโปรแกรม (Exit Program) ซึ่งจะกล่าวถึงรายละเอียดในแต่ละเมนูในภายหลัง



ภาพประกอบ ข-1

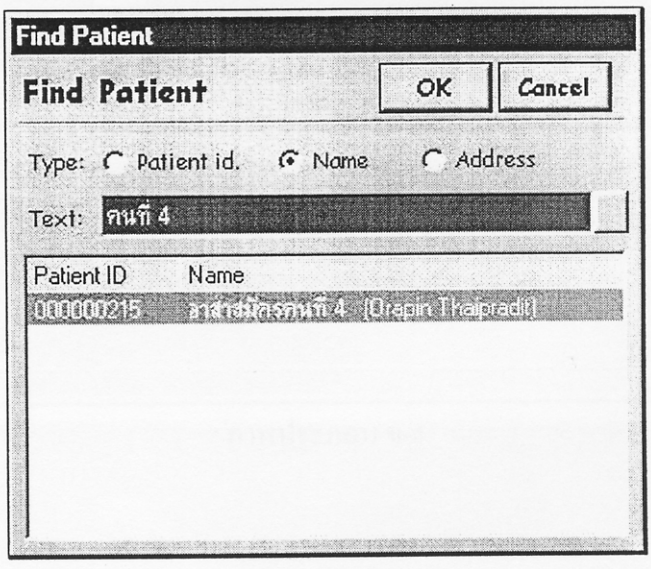
เมื่อเข้าสู่โปรแกรมแล้ว เมนูทุกเมนูยกเว้นเมนู Exit Program จะถูกล็อกไว้ไม่ให้ใช้งานได้ จนกระทั่งผู้ใช้ป้อนหมายเลขประจำตัวผู้ป่วยที่ถูกต้องก่อน ซึ่งสามารถป้อนได้ที่ช่องหมายเลขประจำตัวผู้ป่วย (Patient id) หากผู้ใช้ป้อนหมายเลขประจำตัวของผู้ป่วยไม่ถูกต้อง กล่าวคือไม่มีข้อมูลของผู้ป่วยหมายเลขดังกล่าวในฐานข้อมูล โปรแกรมจะแสดงหน้าจอแจ้งเตือนว่าไม่พบข้อมูล และจะถามว่าต้องการเพิ่มหมายเลขประจำตัวผู้ป่วยนี้เป็นผู้ป่วยใหม่หรือไม่ ดังภาพประกอบ ข-2



ภาพประกอบ ข-2

หากต้องการเพิ่มหมายเลขประจำตัวดังกล่าวเป็นผู้ป่วยใหม่ให้กดปุ่ม Yes โปรแกรมจะเข้าสู่หน้าจอ Patient's Data เพื่อรอให้ผู้ใช้ป้อนข้อมูลผู้ป่วยใหม่ ซึ่งจะกล่าวถึงในภายหลัง หรือกดปุ่ม No เพื่อกลับไปป้อนหมายเลขประจำตัวผู้ป่วยใหม่อีกครั้ง

ถ้าผู้ใช้ไม่ทราบหมายเลขประจำตัวผู้ป่วยแต่ทราบชื่อหรือที่อยู่ของผู้ป่วย ให้กดปุ่ม เพื่อเข้าสู่หน้าจอค้นหาตามชื่อหรือที่อยู่ของผู้ป่วย โปรแกรมจะแสดงหน้าจอสำหรับค้นหามาให้ ดังภาพประกอบ ข-3

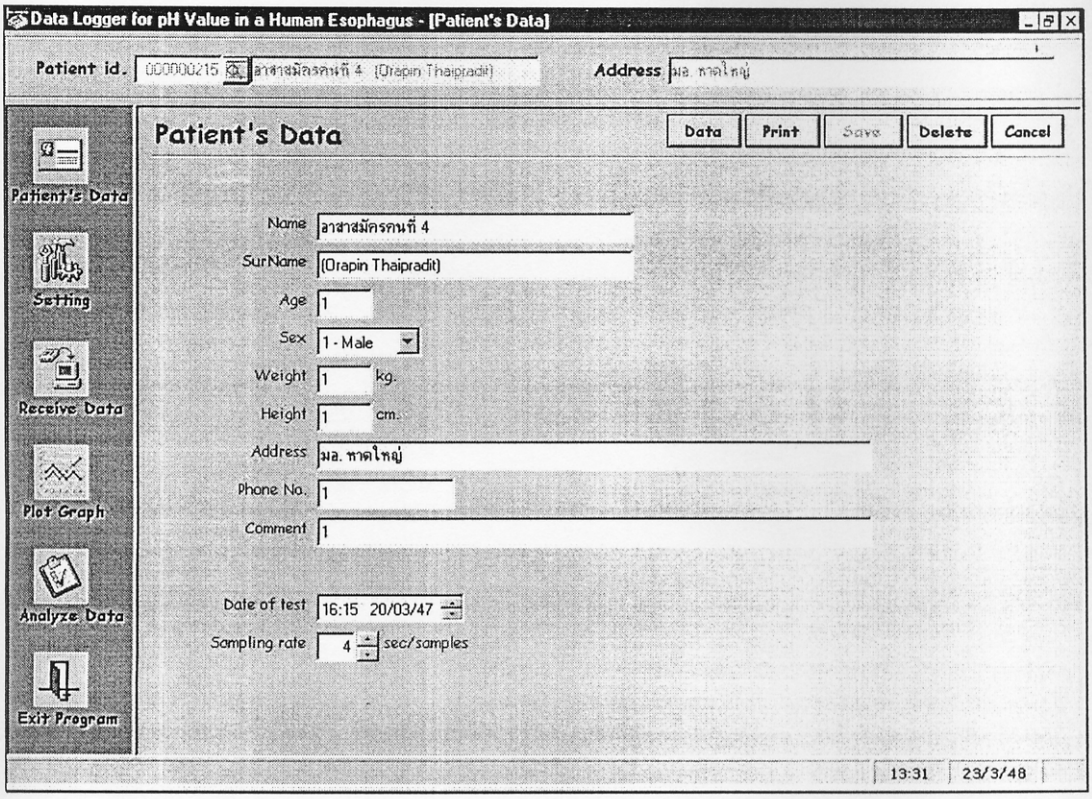


ภาพประกอบ ข-3

ในหน้าจอนี้ ผู้ใช้ต้องระบุว่าจะค้นหาโดยใช้หมายเลขประจำตัว หรือชื่อ หรือที่อยู่ จากตัวเลือกประเภทการค้น (Type) และป้อนข้อความที่จะใช้ค้นหาในช่องข้อความ (Text) แล้วกด Enter หรือกดปุ่ม โปรแกรมจะค้นหาข้อมูลในฐานข้อมูลที่ใกล้เคียงกับข้อความที่ผู้ใช้ป้อน และจะแสดงข้อมูลที่พบให้ผู้ใช้เลือกในส่วนด้านล่างของหน้าจอ ผู้ใช้สามารถเลือกรายการของผู้ป่วยที่ต้องการ เมื่อเลือกแล้วให้กดปุ่ม OK หรือกดปุ่ม Cancel เพื่อยกเลิกการค้นหา

หลังจากป้อนหมายเลขประจำตัวผู้ป่วยแล้วผู้ใช้สามารถเลือกใช้เมนูต่างๆได้ทุกเมนู โดยมีรายละเอียดการใช้งานในแต่ละเมนูดังนี้

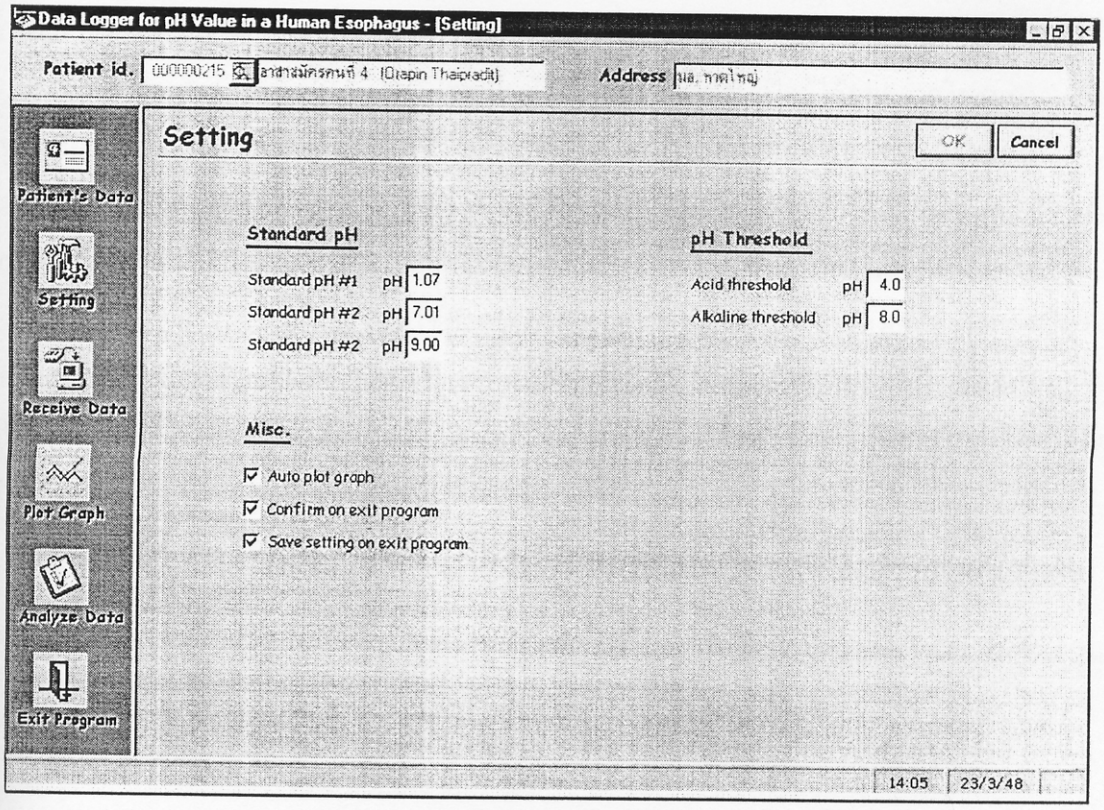
1. Patient's Data เป็นหน้าจอที่ใช้สำหรับเพิ่มหรือแก้ไขข้อมูลผู้ป่วย โดยจะเก็บข้อมูลประวัติของผู้ป่วย เช่นชื่อ, นามสกุล, น้ำหนัก และส่วนสูง เป็นต้น ตลอดจนเวลาที่เริ่มทำการบันทึกและความถี่ของการบันทึก เพื่อนำข้อมูลนี้ไปประกอบการวินิจฉัยของแพทย์ โดยหน้าจอในส่วนนี้แสดงดังภาพประกอบ ข-4



ภาพประกอบ ข-4

ในหน้าจอนี้ผู้ใช้สามารถเพิ่มหรือแก้ไขข้อมูลของผู้ป่วยได้โดยการป้อนข้อมูลในช่องตามหัวข้อต่างๆ หลังจากนั้นให้กดปุ่ม Save เพื่อบันทึกข้อมูล หรือกดปุ่ม Delete เพื่อลบข้อมูลของผู้ป่วยคนนี้ หรือสามารถพิมพ์ข้อมูลของผู้ป่วยออกทางเครื่องพิมพ์ได้โดยการกดปุ่ม Print หรือสามารถดูข้อมูลค่าพีเอชของผู้ป่วยที่รับมาจากเครื่องวัดและบันทึกค่าพีเอชได้โดยการกดปุ่ม Data ซึ่งข้อมูลที่ได้จะเป็นข้อมูลดิบที่ยังไม่แปลงเป็นระดับค่าพีเอช (เป็นค่าระดับของแรงดันที่วัดได้) ผู้ใช้สามารถปิดหน้าจอนี้ได้โดยการกดปุ่ม Cancel

2. Setting เป็นส่วนที่ใช้สำหรับกำหนดค่าเริ่มต้นต่างๆให้กับตัวโปรแกรม ซึ่งได้แก่ ระดับค่าพีเอชมาตรฐานที่ใช้ในการปรับแต่งทั้ง 3 ค่า, ระดับค่าพีเอชที่ใช้อ้างอิงเพื่อตรวจจับการเกิดรีฟลักซ์ทั้งในส่วนกรดและด่าง และการกำหนดค่าอื่นๆ ดังภาพประกอบ ข-5



ภาพประกอบ ข-5

ในหน้าจอนี้ผู้ใช้สามารถแก้ไขค่าเริ่มต้นต่างๆได้โดยการแก้ไขข้อมูลในช่องต่างๆ หลังจากนั้นให้กดปุ่ม [OK] เพื่อให้โปรแกรมนำค่าที่กำหนดใหม่ไปใช้ หรือสามารถปิดหน้าจอนี้ได้โดยการกดปุ่ม [Cancel] โดยค่าต่างๆที่สามารถกำหนดได้ผ่านหน้าจอนี้มีดังต่อไปนี้

- Standard pH #1, #2 และ #3 ทั้งสามค่าจะเป็นค่าพีเอชที่ใช้ในขั้นตอนการปรับแต่ง (Calibrate) เครื่องวัดและบันทึกค่าพีเอช

- Acid threshold จะเป็นระดับค่าพีเอชที่ใช้ในการตรวจจับรีฟลักซ์ในด้านกรด กล่าวคือหากค่าพีเอชที่วัดได้ในขณะทำการวัดมีค่าต่ำกว่าค่าๆนี้นานกว่า 15 วินาที จะถือว่าเป็นรีฟลักซ์

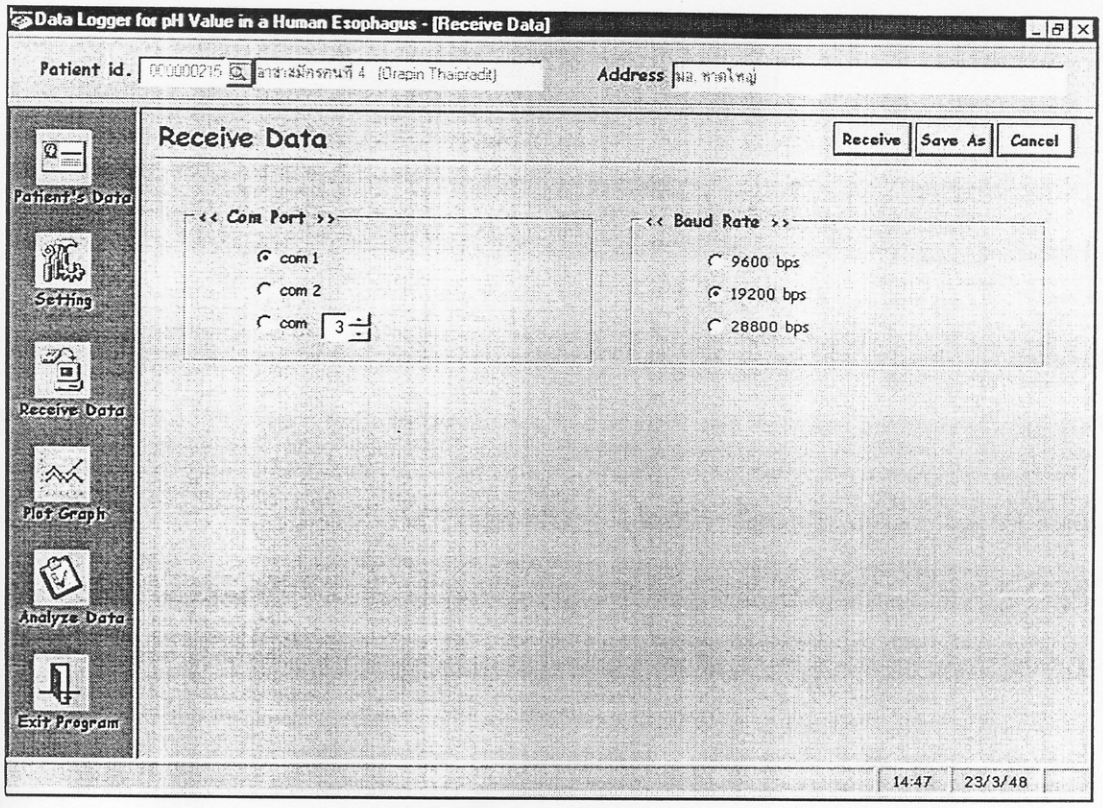
- Alkaline threshold จะเป็นระดับค่าพีเอชที่ใช้ในการตรวจจับรีฟลักซ์ในด้านด่าง กล่าวคือหากค่าพีเอชที่วัดได้ในขณะทำการวัดมีค่าสูงกว่าค่าๆนี้นานกว่า 15 วินาที จะถือว่าเป็นรีฟลักซ์

- Auto plot graph จะเป็นตัวควบคุมการพล๊อตกราฟโดยอัตโนมัติเมื่อเข้าสู่หน้าจอ Plot Graph โดยหากเลือกตัวเลือกนี้จะเป็นการพล๊อตโดยอัตโนมัติ

- Confirm on exit program จะเป็นตัวควบคุมการยืนยันเมื่อผู้ใช้ออกจากโปรแกรม โดยหากเลือกตัวเลือกนี้โปรแกรมจะขึ้นหน้าจอถามว่าผู้ใช้ต้องการออกจากโปรแกรมหรือไม่

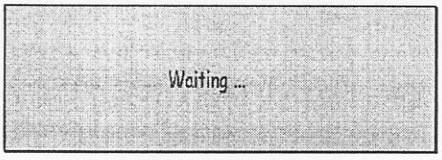
- Save setting on exit program จะเป็นตัวควบคุมบันทึกค่าต่างๆที่ได้กำหนดโดยอัตโนมัติเมื่อผู้ใช้ออกจากโปรแกรม โดยหากเลือกตัวเลือกนี้จะเป็นการให้บันทึกค่าก่อนออกจากโปรแกรม

3. Receive Data เป็นส่วนที่ใช้สำหรับรับข้อมูลจากหน่วยความจำของเครื่องวัดและบันทึกค่าพีเอชมาเก็บยังหน่วยความจำของเครื่องคอมพิวเตอร์ เพื่อนำข้อมูลนี้มาวิเคราะห์ในขั้นตอนต่อไป โดยในส่วนนี้ผู้ใช้ต้องกำหนดหมายเลขพอร์ตที่ใช้เชื่อมต่อ และความเร็วที่ใช้รับส่งข้อมูลให้ตรงกับการกำหนดในส่วนของเครื่องวัดและบันทึกค่าพีเอชด้วย ดังภาพประกอบ ข-6

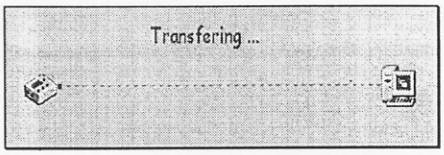


ภาพประกอบ ข-6

ในการใช้งานหน้าจอนี้ผู้ใช้ต้องเชื่อมต่อเครื่องคอมพิวเตอร์และเครื่องวัดและบันทึกค่าพีเอชเข้าด้วยกันด้วยสาย RS-232 และกำหนดหมายเลขพอร์ตอนุกรมและความเร็วในการรับส่งข้อมูลที่ใช้ในการเชื่อมต่อในหัวข้อ Com Port และ Baud Rate ตามลำดับ หลังจากนั้นโดยการกดปุ่ม Receive เพื่อรอรับข้อมูล โปรแกรมจะแสดงหน้าการรอรับข้อมูลจากเครื่องวัดและบันทึกค่าพีเอช ดังภาพประกอบ ข-7 เมื่อข้อมูลเริ่มถูกส่งเข้ามายังโปรแกรม โปรแกรมจะแสดงหน้าจอสถานะการรับข้อมูล ดังภาพประกอบ ข-8 และเมื่อเสร็จสิ้นการรับข้อมูล โปรแกรมจะบันทึกข้อมูลที่รับมาให้เป็นข้อมูลของผู้ป่วยคนปัจจุบันโดยอัตโนมัติ

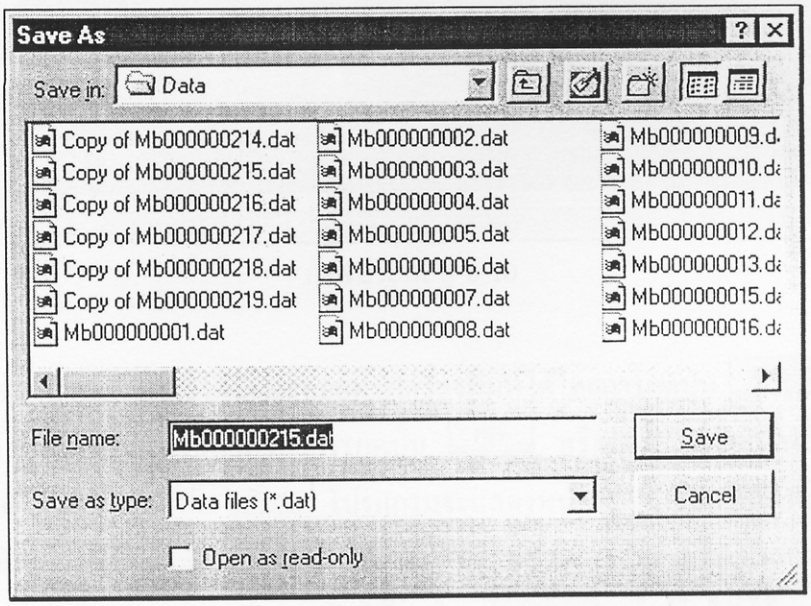


ภาพประกอบ ข-7



ภาพประกอบ ข-8

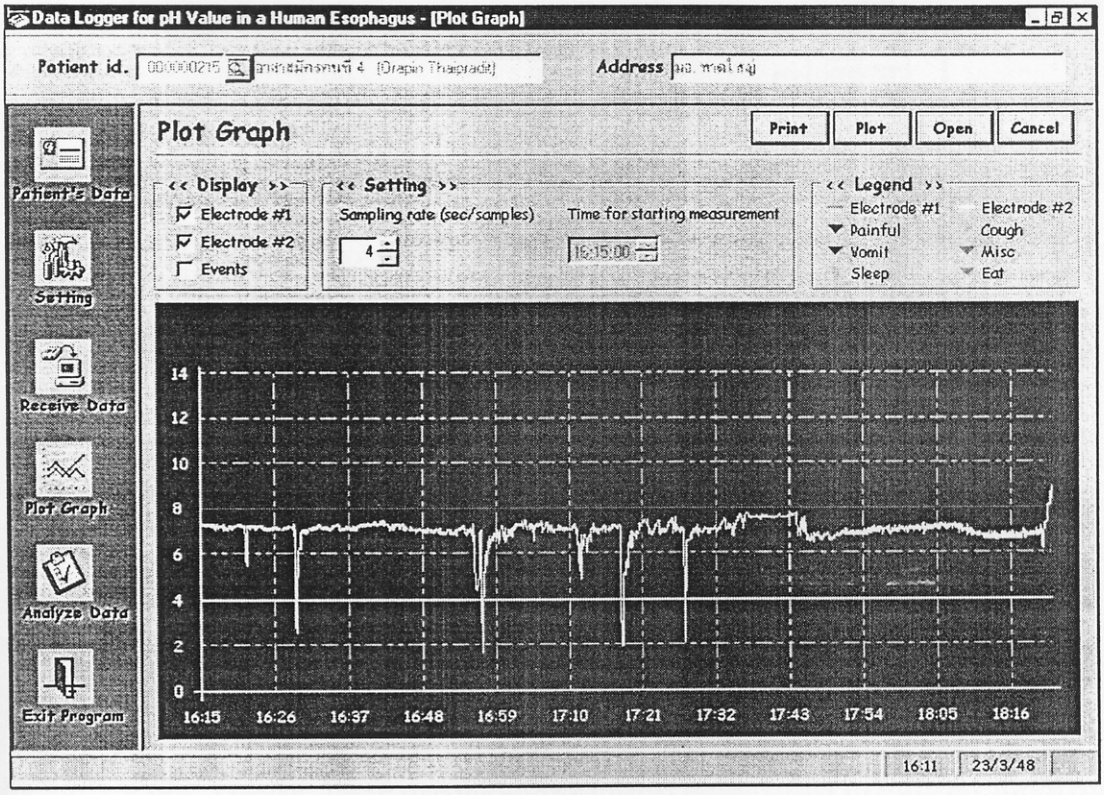
นอกจากนี้ผู้ใช้สามารถบันทึกข้อมูลที่รับมาเป็นไฟล์อื่นได้โดยการกดปุ่ม Save As โปรแกรมจะแสดงหน้าจอให้ผู้ใช้บันทึกไฟล์ ดังภาพประกอบ ข-9 ซึ่งผู้ใช้สามารถเลือกโฟลเดอร์ที่ต้องการเก็บไฟล์ได้จากช่อง Save in และป้อนชื่อไฟล์ที่ต้องการลงในช่อง File name หลังจากนั้นให้กดปุ่ม Save เพื่อบันทึกข้อมูล หรือกดปุ่ม Cancel เพื่อปิดหน้าต่างการบันทึกไฟล์



ภาพประกอบ ข-9

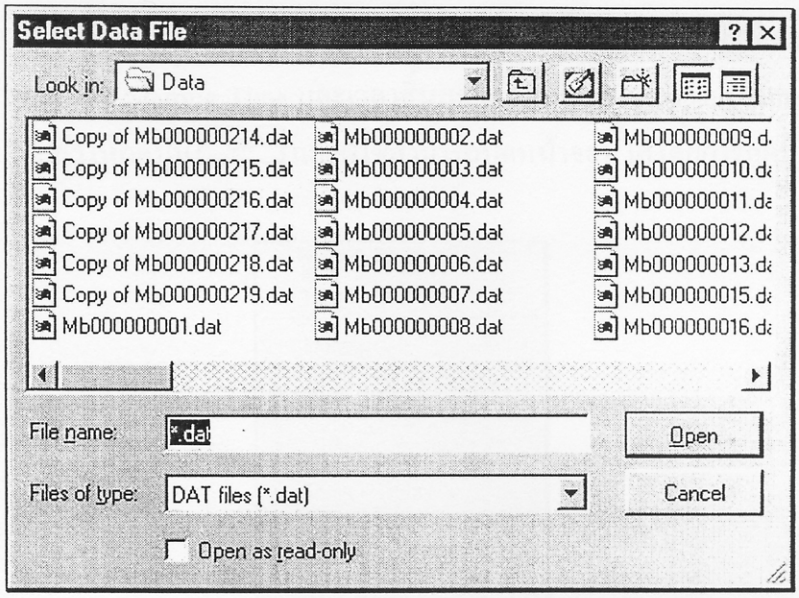
ผู้ใช้สามารถปิดหน้าจอการรับข้อมูลนี้ได้โดยการกดปุ่ม Cancel

4. Plot Graph เป็นส่วนใช้สำหรับแสดงข้อมูลที่รับมาจากเครื่องวัดและบันทึกค่าพีเอชในรูปแบบของกราฟ ซึ่งผู้ใช้สามารถเลือกแสดงข้อมูลของแต่ละอิเลคโทรดหรือเหตุการณ์ที่เกิดขึ้นระหว่างการวัดได้ ดังภาพประกอบ ข-10



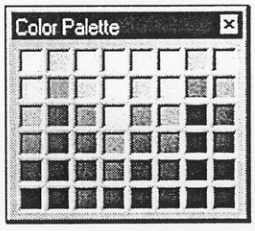
ภาพประกอบ ข-10

ผู้ใช้สามารถพิมพ์กราฟข้อมูลออกทางเครื่องพิมพ์ได้โดยการกดปุ่ม Print หรือสามารถสั่งให้โปรแกรมพล๊อตกราฟใหม่ได้โดยการกดปุ่ม Plot หรือสามารถเลือกไฟล์ข้อมูลที่จะนำมาพล๊อตกราฟได้โดยการกดปุ่ม Open โปรแกรมจะแสดงหน้าจอการเลือกไฟล์ ดังภาพประกอบ ข-11 ซึ่งผู้ใช้สามารถเลือกโฟลเดอร์ที่ต้องการเลือกไฟล์ได้จากช่อง Look in และป้อนชื่อไฟล์ที่ต้องการลงในช่อง File name หลังจากนั้นให้กดปุ่ม Open เพื่อเปิดไฟล์ข้อมูล หรือกดปุ่ม Cancel เพื่อปิดหน้าต่างการเลือกไฟล์



ภาพประกอบ ข-11

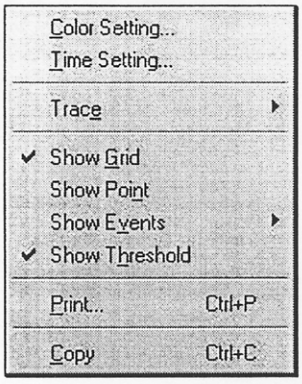
ผู้ใช้สามารถเลือกดูกราฟแต่ละอิเลคโทรดหรือเหตุการณ์ที่เกิดขึ้นได้โดยการเลือกตัวเลือกในหัวข้อ Display และสามารถเลือกสีของเส้นกราฟแต่ละเส้นรวมไปถึงสีของเหตุการณ์ต่างๆได้โดยเลือกสีในหัวข้อ Legend โดยการกดปุ่มเมาส์ซ้ายที่บริเวณสัญลักษณ์ของอิเลคโทรดหรือเหตุการณ์ โปรแกรมจะแสดงหน้าจอตารางสีขึ้นมาให้ผู้ใช้เลือก ดังภาพประกอบ ข-12 ผู้ใช้สามารถเปลี่ยนสีได้โดยการเลือกสีจากตารางสีดังกล่าว



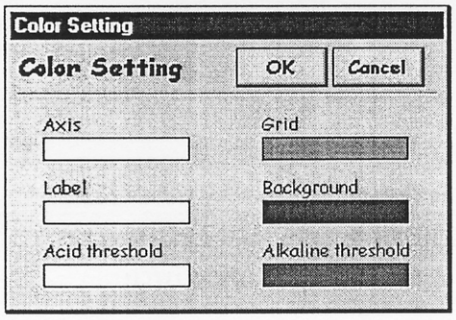
ภาพประกอบ ข-12

ผู้ใช้สามารถเปลี่ยนสีส่วนต่างในบริเวณพื้นที่แสดงกราฟ เช่นสีพื้นหลัง สีเส้นระดับค่าอ้างอิงการเกิดรีฟลักซ์ทั้งด้านกรดและด่าง สีแกนกราฟ เป็นต้น โดยการกดปุ่มเมาส์ด้านขวาบนพื้นที่ว่างบริเวณพื้นที่แสดงกราฟ โปรแกรมจะแสดงเมนูให้ผู้ใช้เลือก ดังภาพประกอบ ข-13 ให้เลือกเมนู Color Setting จะปรากฏหน้าจอกำหนดสี ดังภาพประกอบ ข-14 เมื่อเลือกสีที่ต้องการได้แล้วให้กดปุ่ม OK และผู้ใช้สามารถปิดหน้าจอนี้ได้โดยการกดปุ่ม Cancel นอกจากนี้ผู้ใช้ยังสามารถเลือกดูกราฟในแต่ละช่วงเวลาได้ได้โดยการเลือกเมนู Time Setting จากเมนูในภาพ

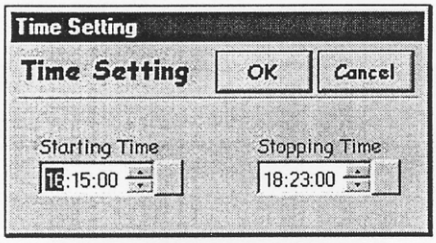
ประกอบ ข-13 จะปรากฏหน้าจอการกำหนดช่วงเวลา ดังภาพประกอบ ข-15 ผู้ใช้สามารถเลือกช่วงเวลาเริ่มต้นได้จากหัวข้อ Starting Time และเวลาสิ้นสุดได้จากหัวข้อ Stopping Time เมื่อเลือกช่วงเวลาที่ต้องการได้แล้วให้กดปุ่ม OK และผู้ใช้สามารถปิดหน้าจอนี้ได้โดยการกดปุ่ม Cancel



ภาพประกอบ ข-13



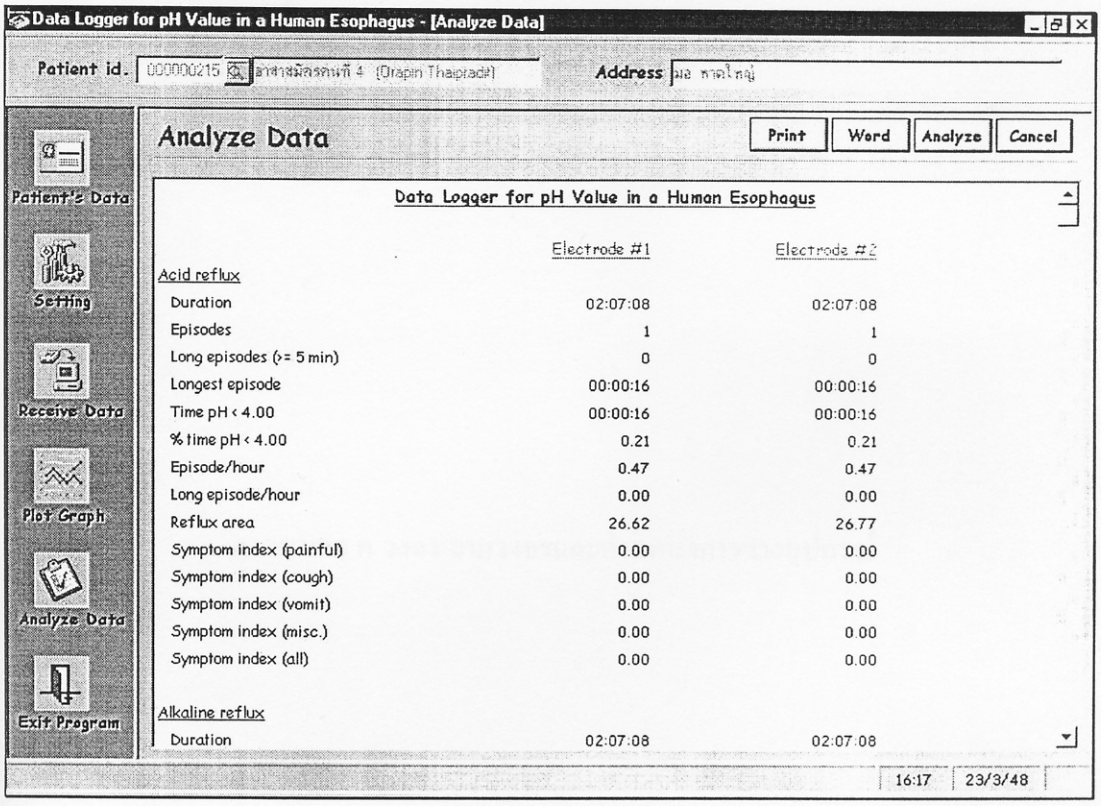
ภาพประกอบ ข-14



ภาพประกอบ ข-15

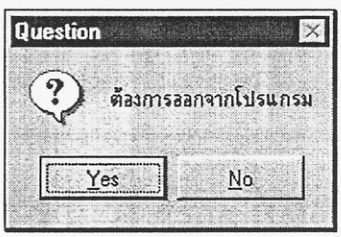
5. Analyze Data เป็นส่วนที่ทำหน้าที่วิเคราะห์ข้อมูล และแสดงผลการวิเคราะห์ ดังภาพประกอบ ข-16 ซึ่งโปรแกรมจะแสดงพารามิเตอร์ต่างๆที่ได้วิเคราะห์ได้ให้ผู้ใช้ดู ผู้ใช้สามารถพิมพ์ผลการวิเคราะห์ออกทางเครื่องพิมพ์ได้โดยการกดปุ่ม Print หรือสามารถส่งผลการวิเคราะห์ไป

ให้กับโปรแกรมไมโครซอฟต์เวิร์ดโดยการกดปุ่ม Word หรือให้โปรแกรมทำการวิเคราะห์ข้อมูลใหม่ได้โดยการกดปุ่ม Analyze หรือสามารถปิดหน้าจอนี้ได้โดยการกดปุ่ม Cancel



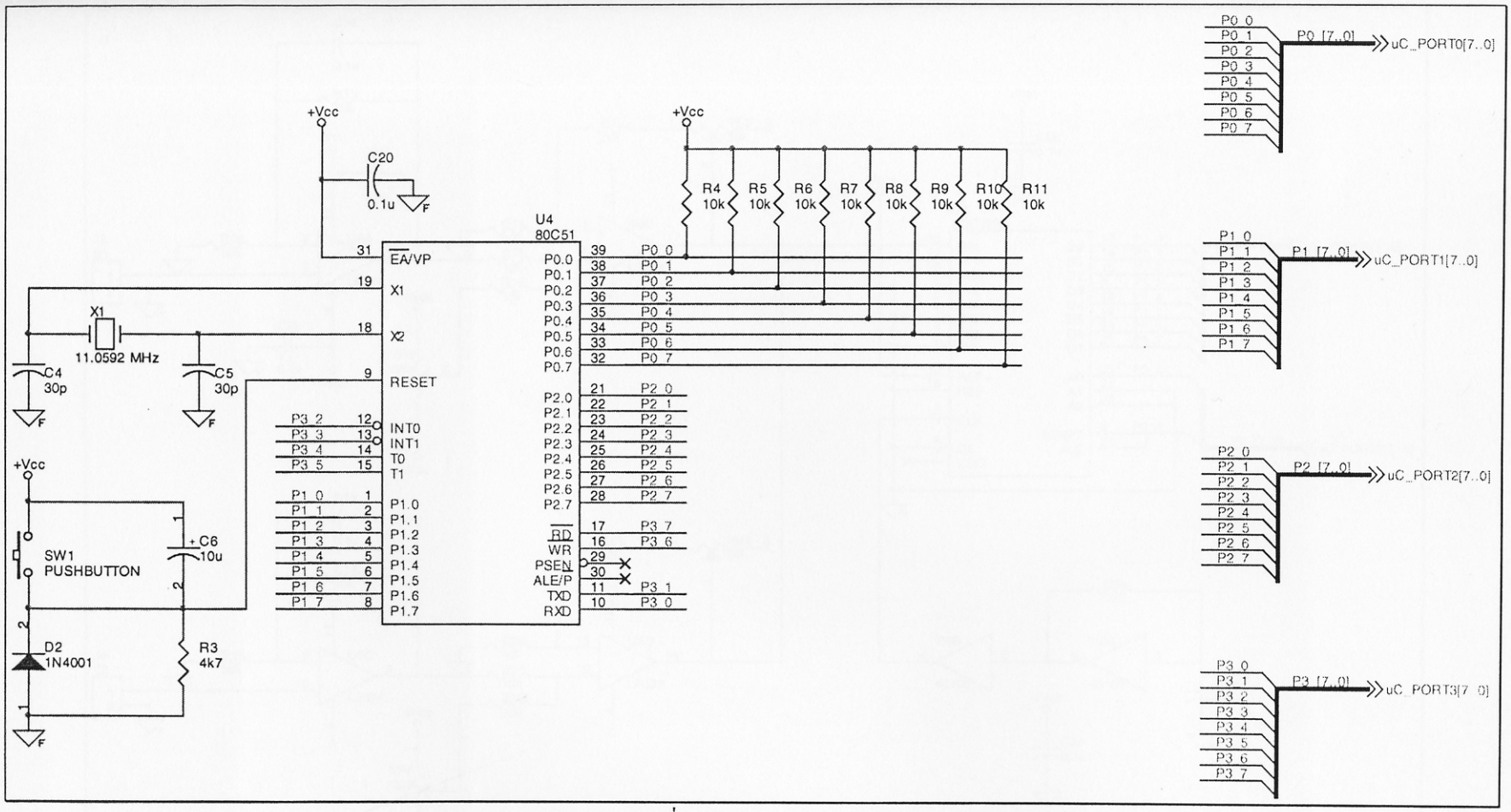
ภาพประกอบ ข-16

6. Exit Program เป็นเมนูที่ใช้ออกจากโปรแกรม ซึ่งเมื่อเลือกเมนูนี้ โปรแกรมจะจบการทำงานลง แต่หากผู้ใช้ได้เซตเลือกตัวเลือก Confirm on exit program ไว้ในหน้าจอ Setting โปรแกรมจะแสดงหน้าจอถามยืนยันการออกจากโปรแกรมอีกครั้ง ดังภาพประกอบ ข-17 หากต้องการออกโปรแกรมให้กดปุ่ม Yes หรือกดปุ่ม No เพื่อยกเลิกการออกจากโปรแกรม

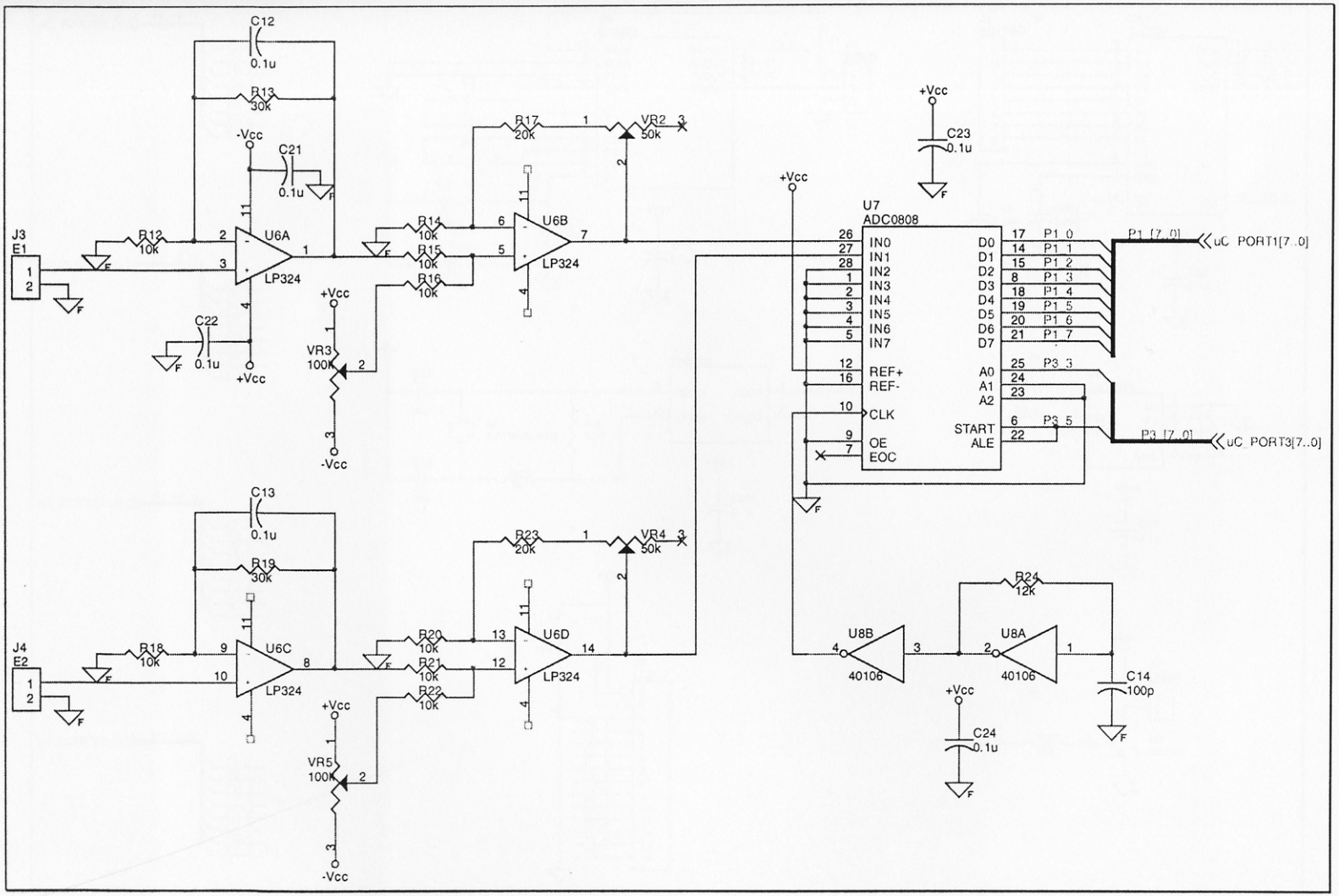


ภาพประกอบ ข-17

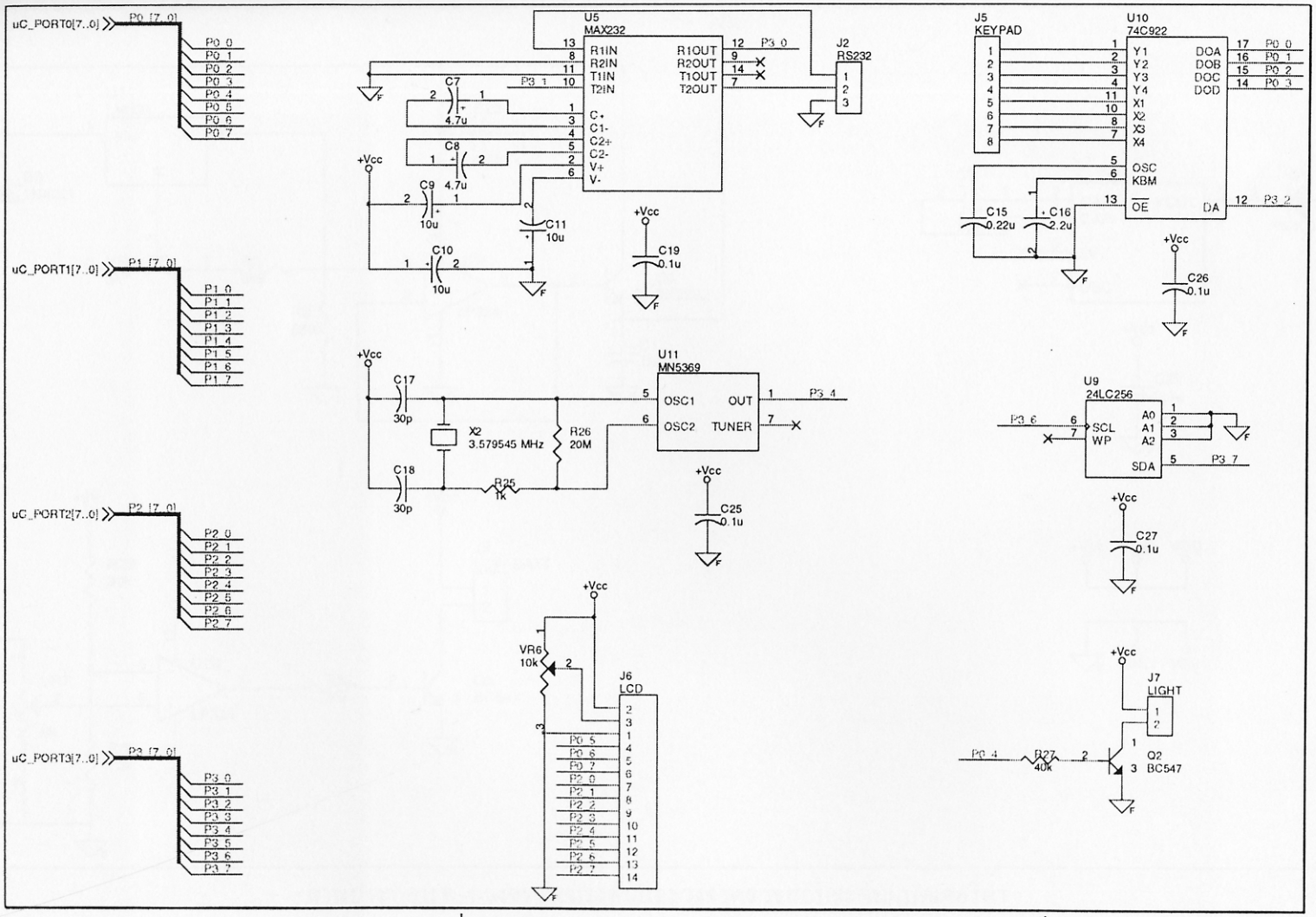
ภาคผนวก ค วงจร ลายวงจรและตำแหน่งการวางอุปกรณ์



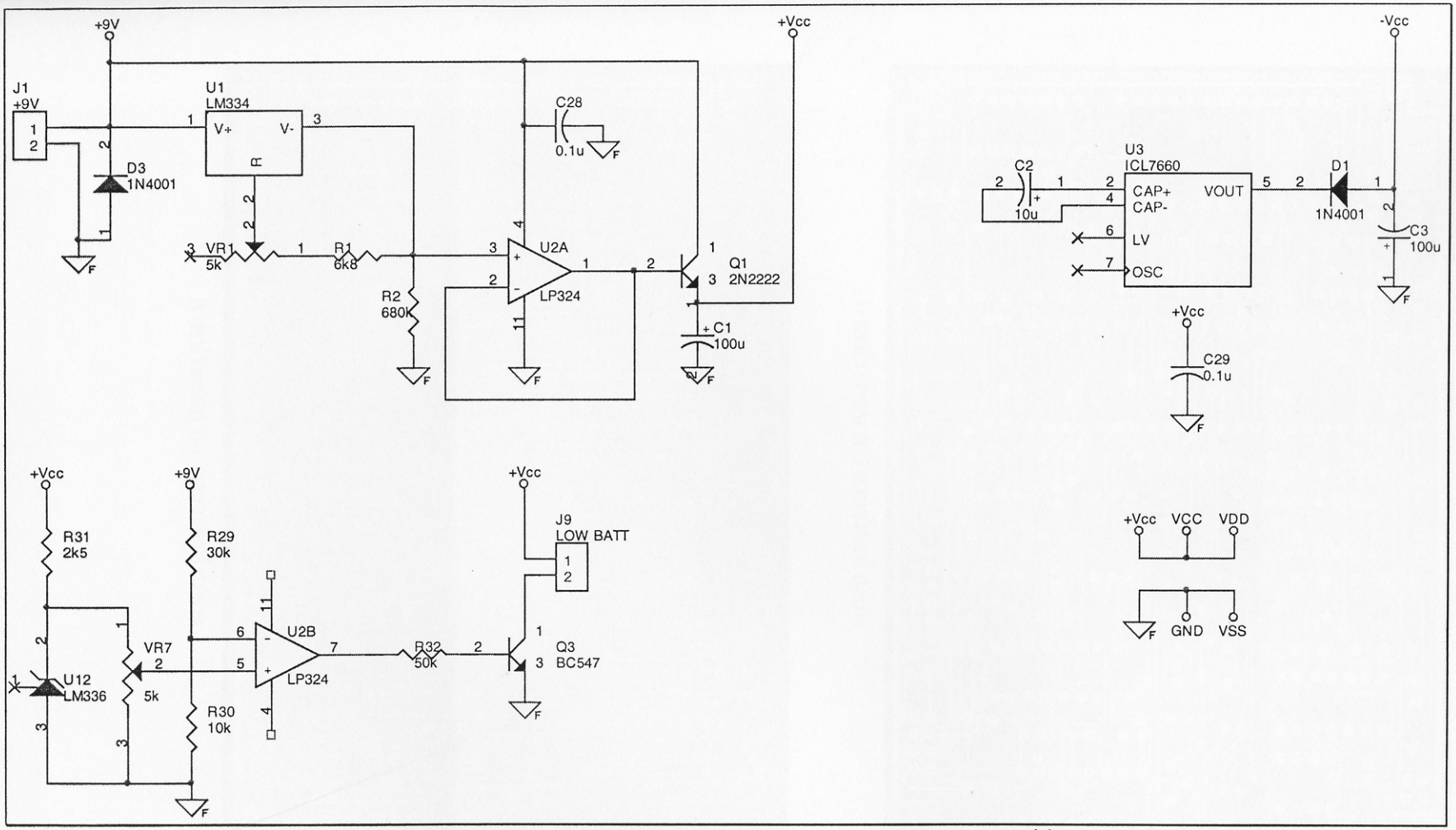
ภาพประกอบ ค-1 การเชื่อมต่อในส่วนของไมโครคอนโทรลเลอร์



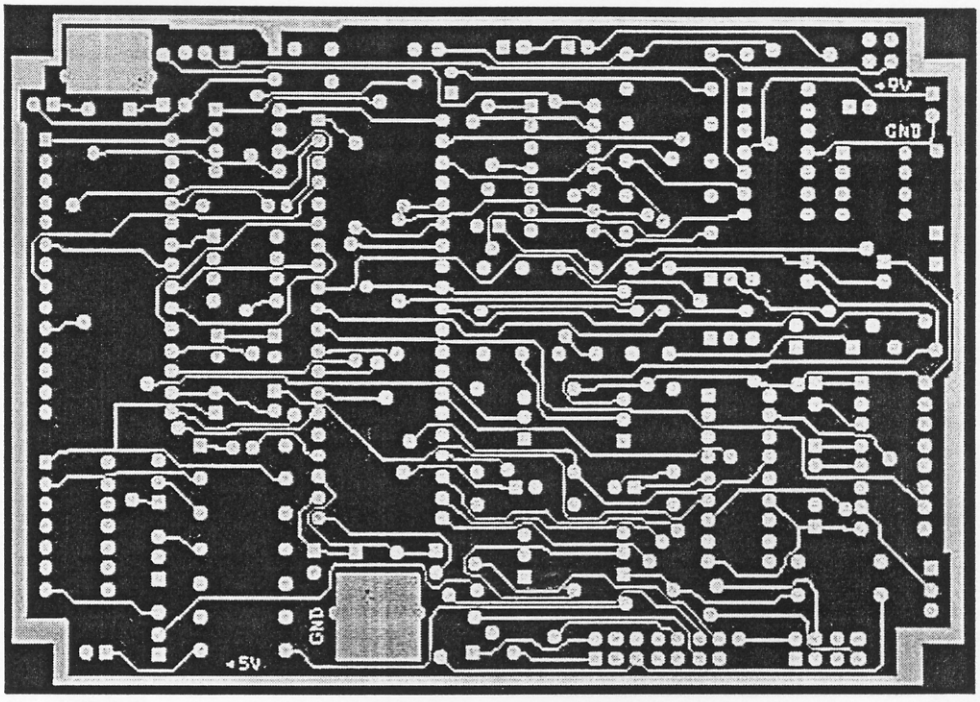
ภาพประกอบ ค-2 การเชื่อมต่อในส่วนของวงจรขยายและวงจรแปลงสัญญาณอนาลอกเป็นดิจิตอล



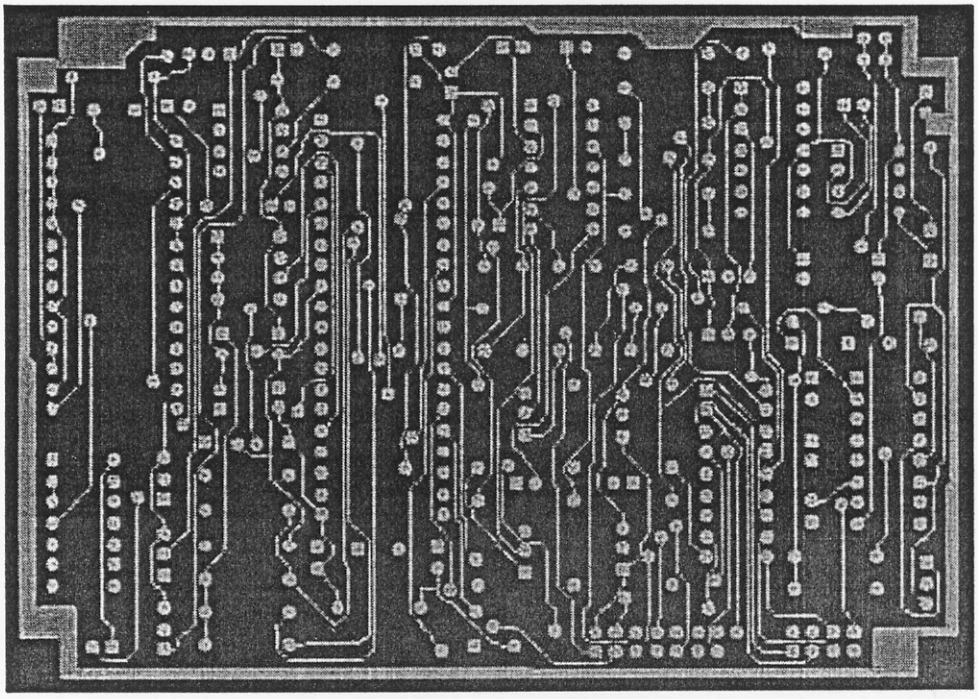
ภาพประกอบ ค-3 การเชื่อมต่อในส่วนของคีย์บอร์ด หน่วยความจำ จอแสดงผล และอื่นๆ



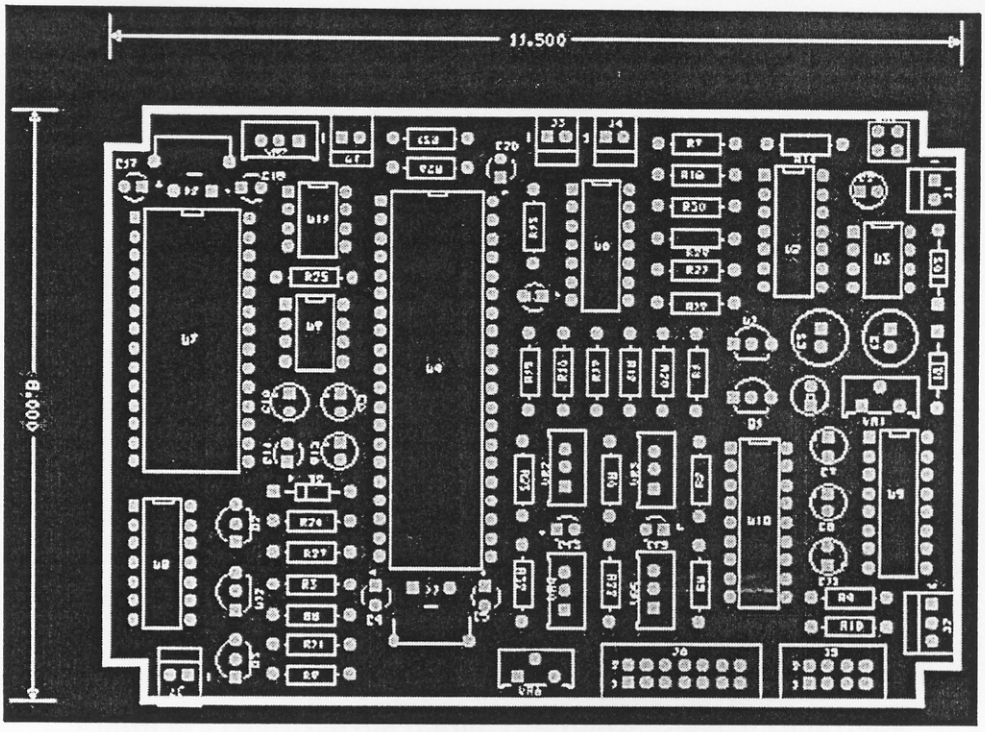
ภาพประกอบ ค-4 วงจรจ่ายแรงดันและวงจรตรวจสอบระดับแบตเตอรี่ต่ำ



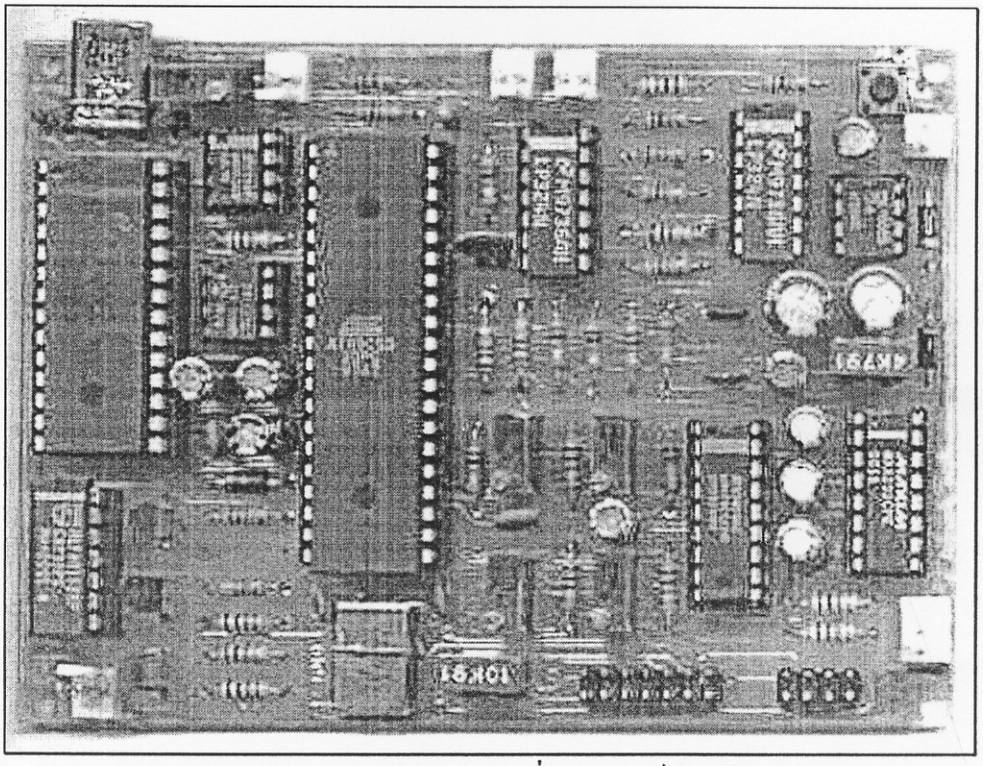
ภาพประกอบ ค-5 ลายวงจรด้านบน



ภาพประกอบ ค-6 ลายวงจรด้านล่าง



ภาพประกอบ ค-7 ตำแหน่งการวางอุปกรณ์



ภาพประกอบ ค-8 แผ่นลายวงจรที่ลงอุปกรณ์เรียบร้อยแล้ว

ภาคผนวก ง ไฟล์โปรแกรมควบคุมการทำงานเครื่องวัดและบันทึกค่าพีเอช

```
/*
        Filename:               pH.c
        FileDesc:               Data Logger for pH Value in a Human
Esophagus
        Compiler:               AVC51.EXE v1.216
        Model:                  SMALL
        CommandLine:            AVC51.EXE pH.c          ; 12 clock per
machine cycle (8051, 89C52)
                                AVC51.EXE -d:CLK6 pH.c  ;  6 clock per
machine cycle (89c51rd2)
*/

#include <8051.h>
#include <math.h>
#include <stdio.h>
#include <conio.h>
#include <intrpt.h>

typedef unsigned char   byte;
typedef unsigned int    word;
typedef unsigned long   dword;

/* No operation define */
#ifdef CLK6
#define nop             asm(" nop "); \
                        asm(" nop ")
#else
#define nop             asm(" nop ")
#endif

/* LCD define */
#define LCD_DATA        (P2)
#define LCD_RS          (P0_BITS.B5)
#define LCD_RW          (P0_BITS.B6)
#define LCD_E           (P0_BITS.B7)

/* EEPROM define */
#define EEPROM_TOTCAP   (32768)
#define EEPROM_CFGCAP   (1)
#define EEPROM_STDCAP   (60)
#define EEPROM_DATCAP   (32707)
#define EEPROM_ADRSTD   (0)
#define EEPROM_ADRDATA  (60)
#define EEPROM_ADRRATE  (32767)
#define EEPROM_SCL      (P3_BITS.B6)
#define EEPROM_SDA      (P3_BITS.B7)

/* Keypad define */
#define KP_DATA         (P0 & 0x0f)
#define KP_DA           (P3_BITS.B2)
#define KP_KEYNONE      (0xff)
#define KP_KEYUP        (0x0a)
#define KP_KEYDOWN      (0x0b)
#define KP_KEYENTER     (0x08)
#define KP_KEYLIGHT     (0x09)
#define KP_KEYFOOD      (0x06)
#define KP_KEYSLEEP     (0x07)
#define KP_KEYETC       (0x00)
```

```
#define KP_KEYVOMIT     (0x01)
#define KP_KEYCOUGH     (0x02)
#define KP_KEYHURT      (0x03)
#define KPDisInt()      EX0 = 0
#define KPEnInt()       EX0 = 1

/* A/D define */
#define ADC_DATA        (P1)
#define ADC_START       (P3_BITS.B5)
#define ADC_SEL         (P3_BITS.B3)
#define ADC_CH1         (0)
#define ADC_CH2         (1)
#define ADC_CH3         (2)

/* Light define */
#define LIGHT           (P0_BITS.B4)
#define LightOn()       LIGHT = 1
#define LightOff()      LIGHT = 0

/* CPU low power mode */
#define CPUIdle()       PCON |= 0x01
#define CPUDown()       PCON |= 0x02

#define STD_PH1         (1.07)
#define STD_PH2         (7.01)
#define STD_PH3         (9)
#define DEF_RATE        (5)
#define MIN_RATE        (1)
#define MAX_RATE        (99)
#define MAX_RAW_PH      (191)

union {
        byte value;
        struct {
                unsigned b7:1;  /* reserve */
                unsigned b6:1;  /* reserve */
                unsigned b5:1;  /* hurt */
                unsigned b4:1;  /* cough */
                unsigned b3:1;  /* vomit */
                unsigned b2:1;  /* etc. */
                unsigned b1:1;  /* sleep */
                unsigned b0:1;  /* food */
        } bits;
} cur_status, old_status; /* user's status */

byte key_buf;                   /* keypad buffer */
byte samp_rate;                 /* sampling rate */
byte light_time;                /* light's time */
byte measure_time;              /* measure's time */
word eeprom_addr;               /* eeprom's address pointer */
bit unsigned measure_flag;      /* measure's flag */
extern bit unsigned ustatus;    /* uart's status */
static byte lcd_addr[4]         = {0x80, 0xc0, 0x90, 0xd0};

/* Millisecond delay time (11.0592 MHz) */
void mdelay(register word d) {
        register byte i;
```

```
        while(d--)
#ifdef CLK6
                for(i = 0; i < 226; i++);
#else
                for(i = 0; i < 113; i++);
#endif
}

/* Delay time about 54 * d usec */
void delay(register word d) {
#ifdef CLK6
        register word i = d;

        while(i--);
#endif
        while(d--);
}

/* Get digit of number */
byte GetDigit(register word num) {
        register byte dig = 1;

        while((num = (int)(num / 10)))
                dig++;

        return dig;
}

/* Put CPU to power down mode */
void CPUSleep(void) {
        if((!measure_time) && (!light_time))
                CPUDown();
}

/* Write a number to df function */
void WriteNum(register word num, byte digit, void (*df)(char)) {
#define MAXDIG  5
        register byte i;
        char buf[MAXDIG];
        register char *bp = &buf[MAXDIG - 1];

        /* check digit */
        if(!digit)
                digit = GetDigit(num);
        else if(digit > MAXDIG)
                digit = MAXDIG;

        /* convert number to string */
        for(i = 0; i < digit; i++) {
                *bp-- = (num % 10) + '0';
                num  /= 10;
        }

        /* write number */
        for(i = 0; i < digit; i++)
                df(*++bp);
}
```

```
/* Write a instrution to lcd */
void LCDWriteInst(byte inst) {
        LCD_DATA = inst;
        LCD_RS   = 0;
        LCD_RW   = 0;
        LCD_E    = 0;
        LCD_E    = 1; delay(25);
        LCD_E    = 0; delay(100);
}

/* Write a data to lcd */
void LCDWriteData(byte data) {
        LCD_DATA = data;
        LCD_RS   = 1;
        LCD_RW   = 0;
        LCD_E    = 0;
        LCD_E    = 1; delay(25);
        LCD_E    = 0; delay(100);
}

/* Move a cursor to (x,y) position */
void LCDGotoXY(byte col, byte row) {
        LCDWriteInst(lcd_addr[row & 0x03] + (col & 0x0f));
}

/* Write a string to lcd at (x,y) position */
void LCDWriteStr(byte col, byte row, register char *s) {
        LCDGotoXY(col, row);

        /* write a string */
        while(*s)
                LCDWriteData(*s++);
}

/* Write a number to lcd at (x,y)  position */
void LCDWriteNum(byte col, byte row, word num, byte dig) {
        LCDGotoXY(col, row);
        WriteNum(num, dig, LCDWriteData);
}

/* Write a string to lcd at center of a line */
void LCDCenterLine(byte line, register char *str) {
        char buf[17];
        register byte len = 0;

        /* copy and check a string */
        while((*str) && (len < 16))
                buf[len++] = *str++;
        buf[len] = '\0';

        /* make even */
        len = (len + 1) & 0xfe;

        /* write string */
        LCDWriteStr(8 - (len >> 1), line, buf);
}

/* Initialize lcd module */
```

```
void LCDInit(void) {
        LCDWriteInst(0x38);     /* function set */
        LCDWriteInst(0x0c);     /* display on/off */
        LCDWriteInst(0x01);     /* clear lcd */
}

/* Timer interrupt function */
void interrupt TimerInt(void) {
        if(measure_time)
                measure_time--;

        if(light_time) {
                if(!(--light_time))
                        LightOff();
        }
}

/* Initialize timer */
void TimerInit(void) {
        set_vector(TIMER0, TimerInt);   /* set timer0 interrupt
vector */

        TR0   = 0;      /* disable timer0 */
        TMOD &= 0xf0;   /* clear mode of timer0 */
        TMOD |= 0x06;   /* timer0 auto-reload mode */
        TH0   = 0xc4;   /* about 1 Hz counter */
        TL0   = TH0;
        ET0   = 1;      /* enable timer0 interrupt */
        TR0   = 1;      /* enable timer0 */
}

/* Initialize uart */
void UartInit(void) {
        TR1   = 0;      /* disable timer1 */
        TMOD &= 0x0f;   /* clear mode of timer1 */
        TMOD |= 0x20;   /* timer1 auto-reload mode */
#ifdef CLK6
        TH1   = 0xfa;   /* about 19200 baud at 11.0592MHz */
#else
        TH1   = 0xfd;   /* about 19200 baud at 11.0592MHz */
#endif
        TL1   = TH1;
        PCON |= 0x80;   /* enable double baud rate */
        SCON  = 0x52;   /* mode 1, receiver enable */
        TR1   = 1;      /* enable timer1 */

        ustatus = 1;
}

/* Keypad check keypress */
byte KPKeypress(void) {
        byte key = key_buf;

        delay(3000);
        key_buf = KP_KEYNONE; /* clear keypad buffer */

        return key;
}
```

```
/* Read key was pressed from keypad */
byte KPReadKey(void) {
        register byte key;

        /* wait until keypress */
        while((key = KPKeypress()) == KP_KEYNONE)
                CPUSleep();

        return key;
}

/* Keypad interrupt function */
void interrupt KeypadInt(void) {
        key_buf = KP_DATA; /* save a key */

        /* light and status key */
        if(measure_flag) {
                switch(KPKeypress()) {
                        case KP_KEYLIGHT:
                        LightOn();
                        light_time = 3;
                        break;

                        case KP_KEYHURT:
                        LightOn();
                        light_time = 1;

                        cur_status.bits.b5 = 1;
                        break;

                        case KP_KEYCOUGH:
                        LightOn();
                        light_time = 1;

                        cur_status.bits.b4 = 1;
                        break;

                        case KP_KEYVOMIT:
                        LightOn();
                        light_time = 1;

                        cur_status.bits.b3 = 1;
                        break;

                        case KP_KEYETC:
                        LightOn();
                        light_time = 1;

                        cur_status.bits.b2 = 1;
                        break;

                        case KP_KEYSLEEP:
                        LightOn();
                        light_time = 1;

                        cur_status.bits.b1 = !cur_status.bits.b1;
                        LCDGotoXY(13, 0);
```

```
                        if(cur_status.bits.b1)
                                LCDWriteData('S');
                        else
                                LCDWriteData(' ');
                        break;

                        case KP_KEYFOOD:
                        LightOn();
                        light_time = 1;

                        cur_status.bits.b0 = !cur_status.bits.b0;
                        LCDGotoXY(14, 0);
                        if(cur_status.bits.b0)
                                LCDWriteData('E');
                        else
                                LCDWriteData(' ');
                        break;
                }
        }
}

/* Initialize interrupt for keypad */
void KeypadInit(void) {
        set_vector(EXTI0, KeypadInt);

        EX0 = 0;        /* disable external interrupt0 */
        IE0 = 0;        /* clear interrupt0 flag */
        IT0 = 1;        /* interrupt0 edge detect */
        EX0 = 1;        /* enable external interrupt0 */
}

/* eeprom start condition */
void EEPROMStart(void) {
        EEPROM_SDA = 1; nop;
        EEPROM_SCL = 1; nop;
        EEPROM_SDA = 0; nop;
        EEPROM_SCL = 0; nop;
}

/* eeprom stop condition */
void EEPROMStop(void) {
        EEPROM_SDA = 0; nop;
        EEPROM_SCL = 1; nop;
        EEPROM_SDA = 1; nop;
}

/* Send data to eeprom */
void EEPROMSendByte(byte data) {
        register byte i, ack;

        do {
                /* send each bit */
                for(i = 0; i < 8; i++) {
                        if((data << i) & 0x80)
                                EEPROM_SDA = 1;
                        else
                                EEPROM_SDA = 0;
                        nop;
```

```
                EEPROM_SCL = 1; nop;
                EEPROM_SCL = 0; nop;
            }

            /* read acknowledge */
            EEPROM_SCL = 1; nop;
            ack = EEPROM_SDA;
            EEPROM_SCL = 0; nop;
        } while(ack);
}

/* Write data to eeprom */
void EEPROMWriteData(word addr, byte data) {
        EEPROMStart();                          /* start condition */
        EEPROMSendByte(0xa0);                   /* wait until EEPROM
not busy */
        EEPROMSendByte((addr >> 8) & 0xff);     /* send address high
*/
        EEPROMSendByte(addr & 0xff);            /* send address low
*/
        EEPROMSendByte(data);                   /* send data */
        EEPROMStop();                           /* stop condition */
        delay(300);                             /* delay for write
circle time */
}

/* Read data from eeprom */
byte EEPROMReadData(word addr) {
        register byte i, data = 0;

        EEPROMStart();                          /* start condition */
        EEPROMSendByte(0xa0);                   /* wait until EEPROM
not busy */
        EEPROMSendByte((addr >> 8) & 0xff);     /* send address high
*/
        EEPROMSendByte(addr & 0xff);            /* send address low
*/

        EEPROMStart();                          /* start condition */
        EEPROMSendByte(0xa1);                   /* device address */

        /* recive each bit */
        for(i = 0; i < 8; i++) {
                EEPROM_SCL = 1; nop;
                data = (data << 1) | EEPROM_SDA;
                EEPROM_SCL = 0; nop;
        }

        /* no acknowledge respond */
        EEPROM_SCL = 1; nop;
        EEPROM_SCL = 0; nop;
        EEPROMStop();

        delay(300); /* delay for write circle time */

        return data;
}
```

```
/* Read data from a/d */
byte ADCReadData(byte channel) {
        if(channel == ADC_CH1)
                ADC_SEL = 0;
        else
                ADC_SEL = 1;
        nop;

        ADC_START = 0; nop;
        ADC_START = 1; nop;
        ADC_START = 0; delay(10);

        return ADC_DATA;
}

/* Show pH value  */
void ShowPH(byte col, byte row, double ph) {
        byte num1, num2;

        if(ph < 0.0)
                ph = 0.0;
        else if(ph > 14.0)
                ph = 14.0;

        num1 = (byte)floor(ph);
        num2 = (byte)floor((double)(ph - (double)num1) * 10.0);
        if(num2 >= 10) {
                num1++;
                num2 -= 10;
        }

        LCDWriteNum(col, row, num1, 2);
        LCDWriteNum(col + 3, row, num2, 1);
}

/* Calibrate standard pH */
void Calibrate(double std_ph) {
        byte i, data;
        byte num = (byte)floor(std_ph);
        byte den = (byte)((byte)((std_ph - num) * 100.0) % 100);

        LCDWriteInst(0x01);
        LCDCenterLine(0, "CALIBRATION pH");
        LCDWriteNum(6, 1, num, 1);
        LCDWriteStr(7, 1, ".");
        LCDWriteNum(8, 1, den, 2);
        LCDCenterLine(3, "press enter key");
        if(KPReadKey() != KP_KEYENTER)
                return;

        LCDWriteInst(0x01);
        LCDCenterLine(0, "CALIBRATE pH");
        LCDWriteNum(6, 1, num, 1);
        LCDWriteStr(7, 1, ".");
        LCDWriteNum(8, 1, den, 2);
        LCDWriteStr(0, 2, "Electrode1 =");
        LCDWriteStr(0, 3, "Electrode2 =");
```

```
        for(i = 0; i < 10; i++) {
                /* electrode #1 */
                data = ADCReadData(ADC_CH1);
                LCDWriteNum(13, 2, data, 3);
                EEPROMWriteData(eeprom_addr++, data);

                /* electrode #2 */
                data = ADCReadData(ADC_CH2);
                LCDWriteNum(13, 3, data, 3);
                EEPROMWriteData(eeprom_addr++, data);

                /* wait for next calibration */
                measure_time = samp_rate;
                while(measure_time)
                        CPUIdle(); /* put CPU to idle mode */
        }
}

/* Menu measurement */
void Measure(void) {
        byte i, j, data;
        double ma[2], ba[2], mb[2], bb[2], avr[2][3];
        word addr_end = EEPROM_TOTCAP - EEPROM_CFGCAP - 3;

        samp_rate        = EEPROMReadData(EEPROM_ADRRATE);      /*
read sampling rate from eeprom */
        cur_status.value = 0;   /* clear current status */
        old_status.value = 0;   /* clear old status */

        /* show title */
        LCDWriteInst(0x01);
        LCDCenterLine(0, "MEASUREMENT pH");
        LCDWriteStr(0, 1, "SAMPLING RATE");
        LCDWriteNum(14, 1, samp_rate, 2);
        LCDCenterLine(3, "press enter key");
        while(KPReadKey() != KP_KEYENTER);

        /* average calibrate pH value */
        eeprom_addr = EEPROM_ADRSTD;
        for(i = 0; i < 3; i++) {
                avr[0][i] = 0.0;
                avr[1][i] = 0.0;
                for(j = 0; j < 10; j++) {
                        avr[0][i] += EEPROMReadData(eeprom_addr++);
                        avr[1][i] += EEPROMReadData(eeprom_addr++);
                }
                avr[0][i] /= 10.0;
                avr[1][i] /= 10.0;
        }

        /* calculate pH equation: pH = (m * raw_data) + b */
        for(i = 0; i < 2; i++) {
                ma[i] = (double)(STD_PH2 - STD_PH1) / (avr[i][1] -
avr[i][0]);
                ba[i] = (double)STD_PH2 - (avr[i][1] * ma[i]);
                mb[i] = (double)(STD_PH3 - STD_PH2) / (avr[i][2] -
avr[i][1]);
                bb[i] = (double)STD_PH2 - (avr[i][1] * mb[i]);
```

```
        }

        /* show measure message */
        LCDWriteInst(0x01);
        LCDWriteStr(0, 0, "MEASURE pH  [  ]");
        LCDWriteStr(0, 2, "Electrode1= 00.0");
        LCDWriteStr(0, 3, "Electrode2= 00.0");

        /* measurement loop */
        measure_flag = 1;
        eeprom_addr  = EEPROM_ADRDATA;
        while(eeprom_addr <= addr_end) {
                /* electrode */
                for(i = 0; i < 2; i++) {
                        data = ADCReadData(i);
                        if(data > MAX_RAW_PH)
                                data = MAX_RAW_PH;

                        if(floor((double)data * 100.0) <= floor(avr
[i][1] * 100.0))
                                ShowPH(12, 2 + i, ((double)data * ma
[i]) + ba[i]);
                        else
                                ShowPH(12, 2 + i, ((double)data * mb
[i]) + bb[i]);

                        EEPROMWriteData(eeprom_addr++, data);
                }

                /* status */
                if(cur_status.value != old_status.value) {
                        EEPROMWriteData(eeprom_addr++,
cur_status.value | (MAX_RAW_PH + 1));
                        cur_status.value &= 0x03;
                        old_status.value  = cur_status.value;
                }

                /* wait for next measure */
                measure_time = samp_rate;
                while(measure_time)
                        CPUIdle(); /* put CPU to idle mode */
        }
        measure_flag = 0;

        /* measurement complete */
        LCDWriteInst(0x01);
        LCDCenterLine(1, "MEASURE");
        LCDCenterLine(2, "COMPLETED");
        while(KPReadKey() != KP_KEYENTER);
}

/* Menu upload data */
void Upload(void) {
        word count = 0;
        byte col = 0, data = 0xff, status = 0;
        word addr_end = EEPROM_TOTCAP - EEPROM_CFGCAP - 3;
        double div = (double)EEPROM_ADRRATE / 100.0 * 6.25;
```

```
        /* show title */
        LCDWriteInst(0x01);
        LCDCenterLine(0, "TRANSFER DATA");
        LCDCenterLine(1, "BAUD RATE 19200");
        LCDCenterLine(3, "press enter key");
        while(KPReadKey() != KP_KEYENTER);

        /* show transfer message */
        LCDWriteInst(0x01);
        LCDCenterLine(0, "TRANSFER DATA");
        LCDWriteStr(0, 2, "Progress:");

        putch(0x0c); /* clear terminal screen */

        /* !!! Do not modify this messages. The software will detect
this messages to start receive data !!! */
        puts("\nData Logger for pH Value in a Human Esophagus");
        puts("\n E1\t E2\t Flag");
        /* !!! ------------------------------------------------------
---------------------------------- !!! */

        /* upload loop */
        eeprom_addr = EEPROM_ADRSTD;
        while(eeprom_addr <= addr_end) {
                /* electrode 1 */
                if(data > MAX_RAW_PH)
                        data = EEPROMReadData(eeprom_addr++);
                WriteNum(data, 3, putch);
                putch('\t');

                /* electrode 2 */
                data = EEPROMReadData(eeprom_addr++);
                WriteNum(data, 3, putch);
                putch('\t');

                /* status */
                data = 0xff;
                if(eeprom_addr > EEPROM_ADRDATA) {
                        data = EEPROMReadData(eeprom_addr++);
                        if(data > MAX_RAW_PH) {
                                status = data & 0x03;
                                WriteNum(data & 0x3f, 3, putch);
                        }
                        else
                                WriteNum(status, 3, putch);
                }
                putch('\n');

                LCDWriteNum(11, 2, ++count, 5); /* show current line
of data */
                if((byte)(eeprom_addr / div) > col) {
                        LCDGotoXY(col++, 3);
                        LCDWriteData(0xff); /* show current progress
*/
                }
        }
        putch(26); /* end of data */
```

```
        /* transfer complete */
        LCDWriteInst(0x01);
        LCDCenterLine(1, "TRANSFER");
        LCDCenterLine(2, "COMPLETED");
        while(KPReadKey() != KP_KEYENTER);
}

/* Menu setup */
void Setup(void) {
        byte key;

        /* read sampling rate from eeprom */
        samp_rate = EEPROMReadData(EEPROM_ADRRATE);
        if(samp_rate > MAX_RATE)
                samp_rate = DEF_RATE;

        /* get sampling rate */
        LCDWriteInst(0x01);
        LCDCenterLine(0, "SAMPLING RATE");
        LCDWriteStr(6, 2, "[   ]");
        LCDWriteInst(0x0e);
        do {
                LCDWriteNum(7, 2, samp_rate, 2);
                LCDGotoXY(8, 2);

                key = KPReadKey();
                if(key == KP_KEYUP) {
                        if(++samp_rate > MAX_RATE)
                                samp_rate = MIN_RATE;
                }
                else if(key == KP_KEYDOWN) {
                        if(--samp_rate < MIN_RATE)
                                samp_rate = MAX_RATE;
                }
        } while(key != KP_KEYENTER);
        LCDWriteInst(0x0c);

        EEPROMWriteData(EEPROM_ADRRATE, samp_rate); /* write sampling
to eeprom */

        /* calibration standard pH */
        eeprom_addr = EEPROM_ADRSTD;
        Calibrate(STD_PH1); /* calibrate standard pH #1 */
        eeprom_addr = EEPROM_ADRSTD + 20;
        Calibrate(STD_PH2); /* calibrate standard pH #2 */
        eeprom_addr = EEPROM_ADRSTD + 40;
        Calibrate(STD_PH3); /* calibrate standard pH #3 */

        /* setup complete */
        LCDWriteInst(0x01);
        LCDCenterLine(1, "SETUP");
        LCDCenterLine(2, "COMPLETED");
        while(KPReadKey() != KP_KEYENTER);
}

void main(void) {
        byte key, choice = 0;
```

```
        /* initial variables */
        ustatus      = 0;
        measure_flag = 0;
        key_buf      = KP_KEYNONE;

        /* initial peripherals */
        di();
        LightOff();
        LCDInit();
        UartInit();
        TimerInit();
        KeypadInit();
        ei();

        /* show title */
        LCDWriteInst(0x01);
        LCDCenterLine(1, "DATA LOGGER");
        LCDCenterLine(2, "PSU-NECTEC");
        delay(60000); delay(60000); delay(60000);

        /* main loop */
        for(;;) {
                /* show menu */
                LCDWriteInst(0x01);
                LCDCenterLine(0, "MENU");

                /* select menu */
                do {
                        LCDWriteStr(0, 1, "  MEASUREMENT");
                        LCDWriteStr(0, 2, "  UPLOAD DATA");
                        LCDWriteStr(0, 3, "  SETUP");
                        LCDGotoXY(0, choice + 1);
                        LCDWriteData(0x7e);

                        key = KPReadKey();
                        if(key == KP_KEYUP)
                                choice = (choice + 2) % 3;
                        else if(key == KP_KEYDOWN)
                                choice = (choice + 1) % 3;
                } while(key != KP_KEYENTER);

                /* check menu */
                if(choice == 0)
                        Measure();      /* measurement */
                else if(choice == 1)
                        Upload();       /* upload data */
                else if(choice == 2)
                        Setup();        /* setup */
        }
}
```

ผลงานตีพิมพ์เผยแพร่จากวิทยานิพนธ์

การประชุมวิชาการทางวิศวกรรมศาสตร์มหาวิทยาลัยสงขลานครินทร์ ครั้งที่ 3
8-9 ธันวาคม 2547

การพัฒนาระบบช่วยวินิจฉัยโรค Gastroesoghageal Reflux

Development of Gastroesophageal Reflux Diagnosis System

สามารถ เครือทอง[1] ชูศักดิ์ ลิ่มสกุล[2] เสกสิต โอสถากุล[3] บุญเจริญ วงศ์กิตติศึกษา[4]
[1,2,4] ภาควิชาวิศวกรรมไฟฟ้า คณะวิศวกรรมศาสตร์ มหาวิทยาลัยสงขลานครินทร์ อ.หาดใหญ่ จ.สงขลา 90112
[3] ภาควิชากุมารเวชศาสตร์ คณะแพทยศาสตร์ มหาวิทยาลัยสงขลานครินทร์ อ.หาดใหญ่ จ.สงขลา 90112
Email: chusak.l@psu.ac.th

Samart Kruthong[1] Chusak Limsakul[2] Seksit Osthakul[3] Booncharoen Wongkittisusa[4]
[1,2,4] Department of Electrical Engineering, Faculty of Engineering, Prince of Songkla University, Hat Yai, Songkhla 90112
[3] Department of , Faculty of Medicine, Prince of Songkla University, Hat Yai, Songkhla 90112
Email: chusak.l@psu.ac.th

บทคัดย่อ

งานวิจัยนี้เสนอการพัฒนาระบบตรวจวัดภาวะกรดด่างในหลอดอาหารเพื่อช่วยให้แพทย์สามารถวินิจฉัยโรค Gastroesopha-geal reflux (GER) ระบบที่พัฒนาขึ้นจะประกอบด้วย 2 ส่วน ส่วนที่หนึ่งเป็นเครื่องวัดและบันทึกค่า pH ในหลอดอาหารที่ผู้ป่วยสามารถพกพาติดตัวไปได้ เครื่องจะสามารถบันทึกค่า pH ในหลอดอาหารและอาการที่เกิดขึ้นในขณะที่กำลังวัดเช่น การนอน การรับประทานอาหาร เป็นต้น และบันทึกค่าพีเอช ได้ 24 ชั่วโมง โดยสามารถถ่ายข้อมูลไปยังไมโครคอมพิวเตอร์ผ่านทางพอร์ต RS232 ส่วนที่สองเป็นโปรแกรมซึ่งทำงานบนเครื่องไมโครคอมพิวเตอร์เพื่อช่วยให้แพทย์สามารถวิเคราะห์ข้อมูลค่า pH เพื่อหาค่าพารามิเตอร์ที่ช่วยในการวินิจฉัยโรค จากการทดลองจำลองข้อมูลพบว่าเครื่อง วัดค่า pH และโปรแกรมวิเคราะห์ผลสามารถทำงานได้ถูกต้องตามที่ออกแบบไว้โดยสามารถหาพารามิเตอร์ได้ถูกต้องตามที่จำลองไว้ และสามารถวัดค่า pH ของสารละลายบัฟเฟอร์ได้ถูกต้องโดยมีความผิดพลาด ±0.3 pH นอกจากนี้ได้มีการศึกษาในอาสาสมัครโดยทำการวัดค่าพีเอชเปรียบเทียบกับเครื่องของบริษัท Metronics (ที่มีขายในท้องตลาด) ผลจากการวัดแสดงให้เห็นว่าแนวโน้มของค่าพีเอชและ reflux ที่วัดได้จากเครื่องต้นแบบและเครื่องของบริษัท Metronics มีลักษณะคล้ายคลึงกัน และพบว่า reflux ยังเกิดขึ้นในช่วงเวลาเดียวกันด้วย

คำสำคัญ: ภาวะกรดด่าง หลอดอาหาร Gastroesophageal reflux ไมโครคอนโทรลเลอร์

ABSTRACT

This research proposes the development of Ambulatory esophageal pH monitoring. This equipment assists the doctor to determine the pH values in the esophageal for diagnosis the Gastroesophageal reflux. The system consists of 2 parts. The first one is the hardware for measuring and logging the pH value in the esophageal and record the events which occur in the period of measuring such as sleeping and eating etc. The second part is the analysis software which runs on the PC. This software is used for receiving the data from the first part via RS232 port and analysis the data to determine the parameters that the doctor can use it for diagnosis. The results showed that with the simulation data the hardware part and software part could record and determine correctly the ph values and parameters. The measurement in the volunteers was investigated by comparing with the Medtronic Ambulatory Esophageal pH Monitoring (a commercial equipment). The measurement results from the volunteers showed that the pattern of the pH values from our prototype and a commercial were similar and the reflux were found at the same period.

Keywords: pH values, Esophageal, Gastroesophageal reflux, Microcontroller

1. บทนำ

มีโรคชนิดหนึ่งในระบบทางเดินอาหารอันมีสาเหตุจากการทำงานของกล้ามเนื้อหูรูดหลอดอาหารส่วนปลายไม่สามารถทำหน้าที่กั้นระหว่างหลอดอาหารและกระเพาะอาหารได้ตามปกติ ทำให้เกิดการไหลย้อนของกรดจากกระเพาะรวมถึงอาหารขึ้นสู่หลอดอาหาร ความผิดปกตินี้มีชื่อเป็นทางการแพทย์ว่าโรค Gastroesophageal reflux (GER) โรคชนิดนี้ทำให้ให้เกิดผลข้าง

เดียงตามมา เช่น หลอดอาหารอักเสบเพราะถูกทำลายโดยกรดจากกระเพาะ อาการปอดอักเสบสูดสำลัก ปวดเจ็บหน้าอกเรื้อรัง เป็นต้น

การวินิจฉัยภาวะความผิดปกติดังกล่าวทำได้หลายวิธี เช่น การใช้เครื่องบันทึกค่าพีเอช การใช้สารไอโซโทป การใช้กล้องตรวจระบบทางเดินอาหาร(Gastroscope) และการกลืนแป้งเอกซเรย์ จากวิธีการทั้งหมด การใช้เครื่องวัดค่าพีเอชมีข้อได้เปรียบกว่าวิธีอื่นเพราะสามารถตรวจจับความผิดปกติได้ตลอด 24 ชั่วโมง ในขณะที่การตรวจวิธีอื่นๆเป็นการตรวจในช่วงเวลาสั้นๆ จึงทำให้มีโอกาสตรวจไม่พบความผิดปกติได้ เพราะการไหลย้อนจากกระเพาะอาหารสู่หลอดอาหารไม่ได้เกิดตลอดเวลาทั้งวัน ดังนั้นวิธีการตรวจบันทึกค่าพีเอชในหลอดอาหารจึงมีความแม่นยำสูงกว่าวิธีอื่นๆ และถือว่าเป็น Gold standard ในการวินิจฉัยโรค GER

ปัจจุบันเครื่องบันทึกค่าพีเอชในหลอดอาหารรวมถึงซอฟท์แวร์ที่ใช้แปลผลเป็นอุปกรณ์ที่ผลิตจากต่างประเทศที่มีราคาแพง ประมาณ 4 แสนบาทต่อ 1 ชุด จึงทำให้มีการใช้จำกัดเพียงในโรงพยาบาลขนาดใหญ่ระดับโรงเรียนแพทย์และโรงพยาบาลเอกชนบางแห่งเท่านั้น แต่ยังไม่มีการใช้ในโรงพยาบาลทั่วไปในระดับจังหวัด เนื่องจากอุปกรณ์ดังกล่าวมีราคาแพง ดังนั้นการที่สามารถผลิตเครื่องตรวจดังกล่าวได้เองในประเทศ นอกจากจะเป็นการลดการพึ่งพาเทคโนโลยีต่างประเทศ ยังช่วยให้แพทย์ตามโรงพยาบาลต่างๆ ในประเทศไทยมีโอกาสใช้เครื่องบันทึกค่าพีเอชในหลอดอาหารเพื่อการวินิจฉัยโรค GER ได้อย่างแพร่หลายขึ้น

2. การออกแบบ

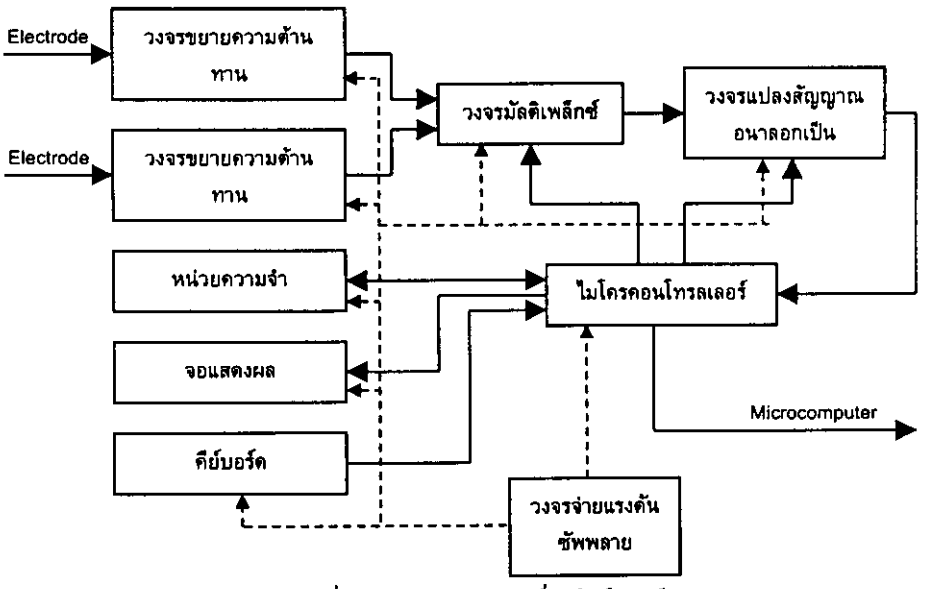
ได้ทำการศึกษาและออกแบบเครื่องวัดและบันทึกค่าพีเอช โดยแบ่งการทำงานออกเป็น 2 ส่วนใหญ่ๆ คือ ส่วนฮาร์ดแวร์ และส่วนซอฟต์แวร์[1]

1. ส่วนฮาร์ดแวร์ ในที่นี้คือเครื่องบันทึกค่าพีเอช ซึ่งทำหน้าที่เป็นเครื่องวัดและบันทึกค่าพีเอช โดยมีไมโครคอนโทรลเลอร์เป็นตัวควบคุมการทำงานโดยสามารถวัดและบันทึกค่าพีเอชได้พร้อมกัน 2 ช่อง นอกจากนี้สามารถถ่ายข้อมูลจากหน่วยความจำไปยังเครื่องไมโครคอมพิวเตอร์ผ่านทางพอร์ท อนุกรมแบบ RS232

2. ส่วนซอฟต์แวร์ คือส่วนที่รับข้อมูลจากเครื่องบันทึกค่าพีเอชผ่าทางพอร์ตอนุกรมแบบ RS232 และทำการวิเคราะห์ข้อมูลและแสดงผลการวิเคราะห์ทางจอคอมพิวเตอร์

2.1 การออกแบบในส่วนฮาร์ดแวร์

เครื่องบันทึกค่าพีเอชที่ได้ออกแบบไว้สามารถรับค่าพีเอชจากอิเล็กโตรดได้พร้อมๆกัน 2 ช่องโดยมีส่วนประกอบดังแสดงในรูปที่ 1 และมีรายละเอียดดังต่อไปนี้



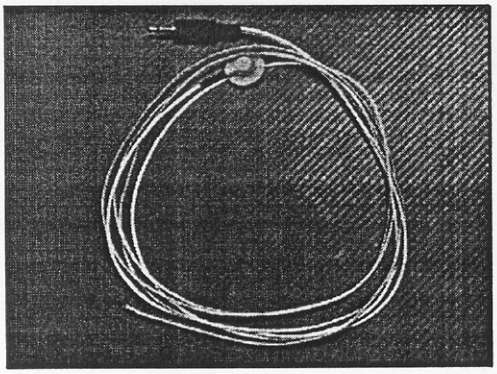
รูปที่ 1 ส่วนประกอบของเครื่องบันทึกค่าพีเอช

2.1.1 อิเล็กโตรด

อิเล็กโตรดที่ใช้วัดค่าพีเอชจะเป็นอิเล็กโตรดขนาดเล็กมีเส้นผ่านศูนย์กลางประมาณ 3 ม.ม ดังรูปที่ 2 ซึ่งสามารถสอดเข้าไปในหลอดอาหารโดยผ่านทางรูจมูกและขั้วอ้างอิง(สีเหลือง) ต้องติดที่ผิดหนังตรงบริเวณแขน อิเล็กโตรดที่ใช้เป็นของบริษัท Metronic

2.1.2 วงจรขยายความต้านทานอินพุทสูง (High input impedance amplifier)

เนื่องจากการวัดค่าพีเอชโดยใช้อิเล็กโตรดนั้น ความต้านทานภายในของอิเล็กโตรดมีค่าสูงมาก และแรงดันที่วัดได้มีค่าแรงดันที่ต่ำและอยู่ในช่วงลบ ทำให้ไม่สามารถแปลงเป็นสัญญาณดิจิตอลได้โดยตรงจำเป็นต้องมีวงจรขยายที่มีความต้านทานขาเข้าสูงและปรับระดับสัญญาณให้แรงดันที่ได้มีค่าอยู่ในช่วง 0-5 โวลท์



รูปที่ 2 ลักษณะอิเล็กโตรดที่ใช้

2.1.3 วงจรมัลติเพล็กซ์ (Multiplex circuit) และวงจรแปลงสัญญาณอนาลอกเป็นดิจิตอล (A/D converter)

เนื่องจากเครื่องบันทึกค่าพีเอชสามารถวัดค่าพีเอชได้พร้อมๆกัน 2 อิเล็กโตรด ดังนั้นสัญญาณอินพุทที่เข้ามาจึงมี 2 สัญญาณ จึงต้องใช้วงจรมัลติเพล็กทำการเลือกสัญญาณใดสัญญาณหนึ่งก่อน แล้วจึงส่งต่อให้กับวงจรแปลงสัญญาณอนาลอกเป็นดิจิตอล เพื่อทำการแปลงสัญญาณ วงจรในส่วนนี้ได้ใช้ IC ADC0808 ซึ่งเป็น A/D converter ที่มีความ ละเอียดขนาด 8 บิท และมีวงจรมัลติเพล็ก 8 ช่อง

2.1.4 ไมโครคอนโทรลเลอร์ (Microcontroller)

ไมโครคอนโทรลเลอร์ที่ใช้เป็นไมโครคอนโทรลเลอร์เบอร์ AT89C52 ในตระกูล 80C51 เพื่อทำหน้าที่ควบคุมการทำงานของวงจรต่างๆดังต่อไปนี้

- ควบคุมการทำงานของวงจรมัลติเพล็กและวงจรแปลงสัญญาณอนาลอกเป็นดิจิตอล
- เก็บบันทึกข้อมูลค่าพีเอชที่วัดได้ไว้ในหน่วยความจำ
- ควบคุมการทำงานของส่วนที่ติดต่อกับผู้ใช้ เช่น จอแสดงผล คีย์บอร์ด เป็นต้น
- ควบคุมการส่งข้อมูลให้ไมโครคอมพิวเตอร์ผ่านทางพอร์ตอนุกรม

2.1.5 หน่วยความจำ (Memory)

ใช้สำหรับเก็บข้อมูลค่าพีเอชที่ได้จากการวัด ซึ่งหน่วยความจำที่ใช้จะเป็นแบบ 2 Wire Serial EEPROM เบอร์ 24C256 โดยมีคุณสมบัติที่สำคัญดังนี้

- แรงดันไฟเลี้ยงต่ำ (4.5 โวลท์ ถึง 5.5 โวลท์)
- มีความจุขนาด 32 Kbytes
- มีการเชื่อมต่อแบบอนุกรม ซึ่งใช้สายเพียง 2 เส้น
- มี Schmitt Triggerและวงจรกรองอินพุทเพื่อกำจัดสัญญาณรบกวน
- การขนถ่ายข้อมูล 2 ทิศทาง
- มีขาป้องกันการเขียนข้อมูล
- เขียนข้อมูลได้ 100000 ครั้ง

2.1.6 คีย์บอร์ด (Keyboard)

ทำหน้าที่ติดต่อกับผู้ใช้งานโดยให้ผู้ใช้งานสามารถป้อนข้อมูลต่างๆให้กับเครื่องบันทึกค่าพีเอช ซึ่งคีย์บอร์ดที่เลือกใช้เป็นคีย์บอร์ดขนาด 16 คีย์ ต่อในลักษณะเมตริก 4x4 และมีไอซีเบอร์ MM74C922 เป็นตัว Encoder คีย์จำนวน 16 คีย์นำมาใช้งานเพียง 10 คีย์โดยใช้สำหรับเลือกฟังก์ชันการทำงาน 3 คีย์ คีย์สำหรับเลือกอาการต่างๆของผู้ป่วยในขณะที่ทำการวัดจำนวน 6 คีย์ และคีย์สำหรับแสงสว่างเพื่อใช้ดูค่าพีเอชในที่มืด

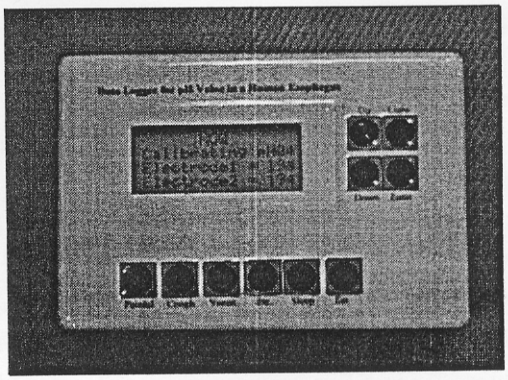
2.1.7 จอแสดงผล (Display)

ทำหน้าที่แสดงค่าพีเอชที่วัดได้ และติดต่อกับผู้ใช้งานโดยการแสดงข้อความต่างๆ โดยจอแสดงผลที่ใช้จะเป็น LCD Module 14 ขา รุ่น DMC164 สามารถแสดงตัวอักษรได้ 4 บรรทัด บรรทัดละ 16 ตัวอักษร

2.1.8 วงจรจ่ายแรงดันซัพพลาย (Power)

ใช้แบตเตอรี่ชนิดประจุใหม่ได้ 12 V 700 mAh แล้วแปลงเป็นแรงดัน $\pm$ 5 V เพื่อจ่ายแรงดันให้กับวงจรต่างๆ

2.2 การทำงานของเครื่องวัดและบันทึกค่าพีเอช



รูปที่ 3 ต้นแบบเครื่องวัดและบันทึกค่าพีเอช

รูปที่ 3 แสดงต้นแบบของเครื่องวัดและบันทึกค่าพีเอชที่ได้พัฒนาขึ้น เมื่อผู้ใช้กดสวิทช์เปิด เครื่องจะปรากฏเมนูบนหน้าจอ LCD ให้ผู้ใช้เลือกฟังก์ชันการทำงาน ผู้ใช้สามารถเลือกฟังก์ชันการทำงานโดยกดคีย์ Up/Down และ Enter

ก่อนทำการวัดต้องทำการ Setup เครื่องใหม่ทุกครั้งโดยเลือกฟังก์ชันการทำงานในโหมด Setup เมื่อเข้าสู่โหมด Setup เครื่องจะให้ผู้ใช้ป้อนข้อมูลอัตราการสุ่ม ซึ่งผู้ใช้สามารถเลือกอัตราการสุ่มโดยใช้คีย์ Up/Down และ Enter หลังจากป้อนข้อมูลอัตราการสุ่มแล้ว เครื่องจะให้ผู้ใช้ทำการ Calibrate ค่าพีเอช 1.07 7.01 และ 9.00 ตามลำดับ

เมื่อเครื่องวัดถูก Setup เรียบร้อยแล้วถ้าต้องการทำการวัดให้เลือกฟังก์ชันการทำงานในโหมดการวัด ซึ่งเครื่องจะทำการวัดและบันทึกค่าพีเอชตามอัตราการสุ่มที่ตั้งไว้และถ้าผู้ป่วยมีอาการเจ็บ ไอ อาเจียน นอน รับประทานอาหาร หรือมีเหตุการณ์อื่นๆใน

ระหว่างที่ทำการวัดและบันทึก ผู้ป่วยต้องกดคีย์ที่สัมพันธ์กับอาการที่เกิดขึ้น ลักษณะอาการจะถูกบันทึกลงในหน่วยความจำด้วย

ในกรณีที่เลือกฟังก์ชันการทำงานในโหมดส่งข้อมูล เครื่องจะทำการส่งข้อมูลค่าพีเอชและลักษณะอาการที่บันทึกไว้ไปยังเครื่องไมโครคอมพิวเตอร์ผ่านทางพอร์ต RS232 ด้วยความเร็ว 19200 บิตต่อวินาที

2.3 รายละเอียดของโปรแกรมวิเคราะห์ผล

เมื่อทำการวัดและบันทึกค่าพีเอชเรียบร้อยแล้ว ข้อมูลที่ได้จากการวัดจะส่งไปยังเครื่องไมโครคอมพิวเตอร์เพื่อทำการวิเคราะห์และแปลผลด้วยโปรแกรมที่พัฒนาขึ้นด้วยภาษา Visual Basic โปรแกรมนี้จะช่วยวิเคราะห์พารามิเตอร์ที่จะทำให้แพทย์วินิจฉัยโรคได้ง่ายขึ้น โดยการทำงานของโครงสร้างหลักของส่วนโปรแกรมแบ่งได้ดังนี้

2.3.1 ส่วนเมนูหลัก เมื่อผู้ใช้รันโปรแกรมจะปรากฏหน้าต่างของเมนูหลัก เพื่อให้ผู้ใช้เลือกการทำงานย่อยของโปรแกรมซึ่งได้แก่ ส่วนรับข้อมูลประวัติผู้ป่วย ส่วนกำหนดค่าต่าง ๆของโปรแกรม ส่วนรับข้อมูลจากเครื่องบันทึกค่าพีเอช ส่วนแสดงข้อมูลด้วยกราฟ และส่วนวิเคราะห์ข้อมูล

2.3.2 ส่วนรับข้อมูลผู้ป่วย เป็นส่วนของโปรแกรมที่ใช้เก็บบันทึกข้อมูลประวัติของผู้ป่วย เช่น ชื่อ นามสกุล น้ำหนัก และส่วนสูง เป็นต้น ตลอดจนเวลาที่เริ่มบันทึกและความถี่ของการบันทึก

2.3.3 ส่วนที่ใช้สำหรับกำหนดค่าเริ่มต้น เป็นส่วนที่ใช้สำหรับกำหนดค่าเริ่มต้นต่าง ๆให้กับโปรแกรม ได้แก่ พีเอชมาตรฐานที่ใช้ Calibrate และระดับพีเอชที่ใช้อ้างอิงเพื่อตรวจจับการเกิด Reflux

2.3.4 รับข้อมูลจากเครื่องบันทึกค่าพีเอช เป็นส่วนที่ใช้สำหรับรับข้อมูลจากเครื่องบันทึกค่าพีเอชมาเก็บที่เครื่องคอมพิวเตอร์ โดยผู้ใช้สามารถเลือกพอร์ตและความเร็วที่ใช้รับข้อมูล

2.3.5 ส่วนแสดงข้อมูล เป็นส่วนใช้สำหรับแสดงข้อมูลในรูปแบบของกราฟซึ่งผู้ใช้สามารถเลือกอิเล็กโตรดที่ต้องการแสดงกราฟและตั้งเวลาเริ่มต้นของการวัดค่าพีเอช

2.3.6 ส่วนวิเคราะห์ข้อมูล เป็นส่วนที่ทำหน้าที่วิเคราะห์ข้อมูลและแสดงผลการวิเคราะห์ โดยค่าพารามิเตอร์ที่วิเคราะห์ได้มีดังนี้

ก. Reflux คือ การเกิดค่าพีเอชต่ำกว่าระดับค่าพีเอชอ้างอิงนานเกิน 15 วินาที (ส่วนใหญ่ใช้ค่าพีเอช 4) ในกรณีที่เป็นกรดหรือการเกิดค่าพีเอชสูงกว่าระดับพีเอชอ้างอิงนานเกิน 15 วินาที (ส่วนใหญ่ใช้ค่าพีเอช 8)ในกรณีของด่าง

ข. Episode คือจำนวนครั้งที่เกิด Reflux

ค. Long Episode คือจำนวนครั้งที่เกิด Reflux ที่นานกว่า 5 นาที

ง. % Time pH < 4 คือร้อยละของเวลาที่ค่าพีเอชที่วัดได้ต่ำกว่า pH 4 เทียบกับเวลาทั้งหมดที่ใช้ในการวัด

จ. Episode/hour คือ Episodeทั้งหมดต่อเวลาที่ใช้ในการวัดทั้งหมด

ฉ. Reflux area คือ พื้นที่ทั้งหมดของการเกิด Reflux

ช. Symptom index คือ จำนวนครั้งของการเกิดเหตุการณ์ในขณะที่เกิด Refluxหารด้วย จำนวนครั้งที่เกิดเหตุการณ์นั้นทั้งหมด

3. การทดสอบ

3.1 การทดสอบโปรแกรมวิเคราะห์ผล

ได้ทำการทดสอบโดยจำลองค่าพีเอชพร้อมกับเหตุการณ์ที่เกิดขึ้นในขณะที่ทำการวัดด้วยโปรแกรม Excel แล้วบันทึกไว้เป็นแฟ้มข้อมูล หลังจากนั้นรันโปรแกรมให้วิเคราะห์ข้อมูลดังกล่าว จะได้กราฟของค่าพีเอชพร้อมเหตุการณ์แสดงในรูปที่ 6 และผลการวิเคราะห์ค่าพารามิเตอร์แสดงในรูปที่ 7 และ 8 สำหรับการเกิด Acid Reflux และ Alkaline Reflux ตามลำดับ

จากรูปที่ 4 จะเห็นว่าค่าพีเอชที่วัดได้มีค่าอยู่ระหว่าง pH4 และ pH10 และจะเกิดอาการเจ็บและไอขึ้นในช่วงเวลา 14.08 น.ถึง 15.46 น. อย่างละ 1 ครั้งและหลังเวลา 6.28 น. อย่างละ 1 ครั้ง ซึ่งตรงกับที่จำลองไว้

จากรูปที่ 5 จะเห็นว่าไม่เกิด Acid Reflux แต่เกิด Alkaline Reflux โดยมี Episode ที่เท่ากับหรือมากกว่า 5 นาทีอยู่ 2 ครั้ง และมี Symptom index สำหรับอาการเจ็บและไอ 50 % ซึ่งตรงกับที่จำลองไว้

3.2 ทดสอบความถูกต้องในการวัดและความน่าเชื่อถือของเครื่อง

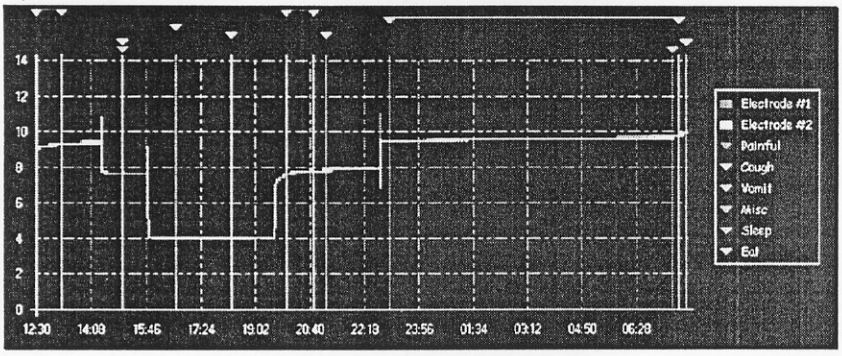
ในการทดสอบความถูกต้องในการวัดค่าพีเอชและน่าเชื่อถือได้ทำการทดลองวัดค่าพีเอช 1.07 และ 7.01 สองครั้ง ครั้งแรก ทำการวัดเป็นเวลา 7 ชั่วโมงที่อุณหภูมิห้อง และครั้งที่ 2 ทำการวัดเป็นเวลา 4 ชั่วโมงที่อุณหภูมิ 37 องศาเซลเซียส ในขณะที่ทำการบันทึกข้อมูลได้กดคีย์กิจกรรมต่าง ๆ ดังแสดงในตารางที่ 1 หลังจากนั้นได้นำข้อมูลที่บันทึกไว้มาแสดงผลดังรูปที่ 6 และ 7 จากผลการทดลองพบเครื่องสามารถวัดค่าพีเอชได้ตรงจากผลการทดลองจะเห็นว่าเครื่องสามารถวัดค่าพีเอชได้ตรงตามที่กำหนดในตารางที่ 1 โดยมีความผิดพลาดไม่เกิน $\pm$ 0,3 pH และสามารถบันทึกกิจกรรมได้ตรงกับการกดคีย์ที่แสดงในตารางที่ 1

3.3 การทดสอบในอาสาสมัคร

ในการทดสอบกับอาสาสมัครนั้น จะทำการวัดค่าพีเอชในหลอดอาหารของอาสาสมัครปกติในช่วงบ่ายหลังอาหารเที่ยงโดยทำการวัดไปพร้อม ๆกับเครื่องของต่างประเทศ ยี่ห้อ Medtronic รุ่น digi-crapper pH วิธีการวัดทำโดยสอดสายอีเล็กโตรด 2 เส้นลงในหลอดอาหาร เส้นหนึ่งต่อเข้ากับเครื่องต้นแบบ และอีกเส้นต่อเข้ากับเครื่อง Medtronic แล้วทำการบันทึกค่าพีเอชไม่น้อยกว่า 2 ชั่วโมง โดยเครื่องต้นแบบจะเก็บข้อมูลทุก ๆ 4 วินาที และเครื่อง Medtronic จะเก็บข้อมูลทุก ๆ 1 วินาที โดยได้ทดสอบกับอาสาสมัคร 5 คน ปรากฏว่าผลการวัดในอาสาสมัครในคนที่ 1 และคนที่ 2 ได้ผลไม่ดีเนื่องจากคาดว่าน่ามีปัญหาของตำแหน่งอิเล็กโตรดทั้งสองเครื่องไม่ได้อยู่ในระดับเดียวกัน

ผลการทดสอบกับอาสาสมัคร พบว่าค่าพีเอชที่วัดได้จากเครื่องต้นแบบและเครื่อง Medtronic มีค่าพีเอชที่ใกล้เคียงกัน และมีแนวโน้มในทิศทางเดียวกัน นอกจากนี้สามารถตรวจวัดการเกิดค่าพีเอชต่ำกว่า 4 pH ได้เหมือนกันทั้งสองเครื่อง ยกเว้นในกรณีที่ค่าพีเอชต่ำกว่า 4 pH ในช่วงเวลาสั้นมากๆ จะเห็นไม่เหมือนกัน เนื่องจากจุดการเก็บข้อมูลไม่ตรงกันทั้งสองเครื่อง ดังตัวอย่างแสดงในรูปที่ 8

เมื่อนำผลการวัดไปวิเคราะห์ด้วยโปรแกรมวิเคราะห์ผลด้วยโปรแกรมที่พัฒนาขึ้นเทียบกับโปรแกรมวิเคราะห์ผลของบริษัท Medtronic พบว่าได้ผลการวิเคราะห์เหมือนกัน โดยพบว่ามี reflux หนึ่งครั้งในอาสาสมัครคนที่ 3 (รูปที่ 8) ที่เวลาประมาณ 17.01 น.



รูปที่ 4 กราฟของค่าพีเอชและการเกิดเหตุการณ์ในขณะที่ทำการวัด
(รูปสามเหลี่ยมแสดงตำแหน่งที่เกิดเหตุการณ์)

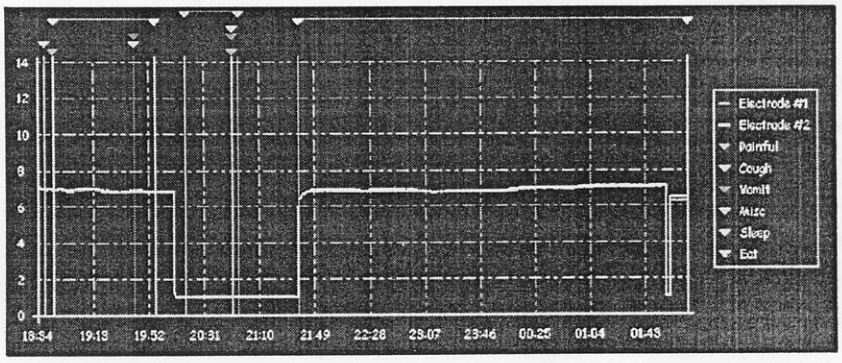
Data Logger for pH Value in a Human Esophagus

	Electrode #1	Electrode #2
Acid reflux		
Duration	19:30:25	19:30:25
Episodes	0	0
Long episodes (>= 5 min)	0	0
Longest episode	00:00:00	00:00:00
Time pH < 4.00	00:00:00	00:00:00
% time pH < 4.00	0.00	0.00
Episode/hour	0.00	0.00
Long episode/hour	0.00	0.00
Reflux area	0.00	0.00
Symptom index (painful)	0.00	0.00
Symptom index (cough)	0.00	0.00
Symptom index (vomit)	0.00	0.00
Symptom index (misc.)	0.00	0.00
Symptom index (all)	0.00	0.00
Alkaline reflux		
Duration	19:30:25	19:30:25
Episodes	2	2
Long episodes (>= 5 min)	2	2
Longest episode	09:14:45	09:14:45
Time pH > 8.00	11:11:35	11:11:35
% time pH > 8.00	57.38	57.38
Episode/hour	0.10	0.10
Long episode/hour	0.10	0.10
Reflux area	62619.62	62305.77
Symptom index (painful)	50.00	50.00
Symptom index (cough)	50.00	50.00
Symptom index (vomit)	0.00	0.00
Symptom index (misc.)	0.00	0.00
Symptom index (all)	28.57	28.57

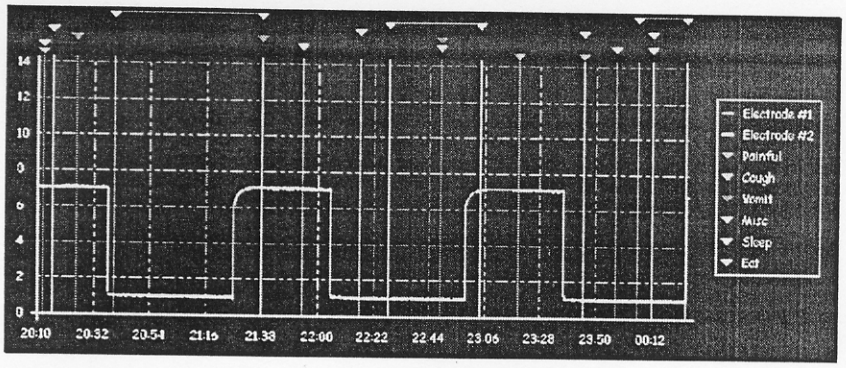
รูปที่ 5 ผลการวิเคราะห์

ตารางที่ 1 ค่าพีเอช และกิจกรรมที่กดคีย์ในเวลาต่างๆ

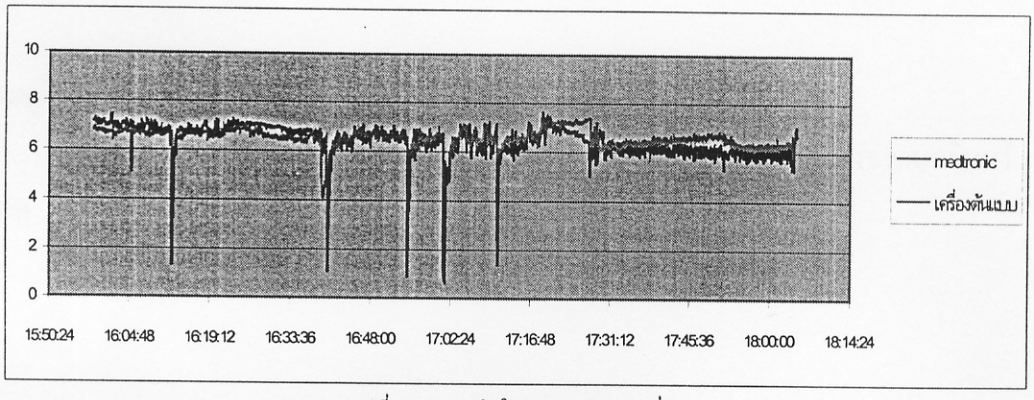
บันทึกค่าพีเอชที่อุณหภูมิห้อง		บันทึกค่าพีเอชที่ 37 °C	
12/12/45 เวลา	ค่าพีเอชที่วัด/คีย์กิจกรรมที่กด	15/12/45 เวลา	ค่าพีเอชที่วัด/คีย์กิจกรรมที่กด
18.34	พีเอช 7.01	20.10	พีเอช 7.01
18.38	ไอ	20.12	เจ็บ/ไอ
18.44	เจ็บ	20.15	อื่นๆ
18.45	นอน	20.25	อาเจียน
19.41	ไอ/อาเจียน	20.37	พีเอช 1.07
19.56	ตื่นนอน	20.40	รับประทานอาหาร
20.10	พีเอช 1.07	21.27	พีเอช 7.01
20.17	รับประทานอาหาร	21.37	อาเจียน/รับประทานอาหารเสร็จ
20.50	เจ็บ/อาเจียน/อื่นๆ	21.54	ไอ
20.55	รับประทานอาหารเสร็จ	22.05	พีเอช 1.07
21.37	พีเอช 7.01	22.16	อื่นๆ
21.38	นอน	22.27	นอน
1.57	พีเอช 1.07	22.48	ไอ/อาเจียน
2.00	ตื่นนอน	22.58	พีเอช 7.01
		23.04	ตื่นนอน
		23.19	เจ็บ
		23.37	พีเอช 1.07
		23.45	เจ็บ/อื่นๆ
		23.57	ไอ
		0.06	รับประทานอาหาร
		0.12	ไอ/อื่นๆ
		0.25	รับประทานอาหารเสร็จ



รูปที่ 6 ผลทดสอบความน่าเชื่อถือของเครื่องครั้งที่ 1 วันที่ 12/12/45



รูปที่ 7 ผลทดสอบความน่าเชื่อถือของเครื่องครั้งที่ 2 วันที่ 15/12/45



รูปที่ 8 ผลการวัดในอาสาสมัครคนที่ 3

4. สรุปผลงานวิจัย

ได้พัฒนาระบบช่วยวินิจฉัยโรค GER ซึ่งแบ่งออกเป็นสอง ส่วน โดยส่วนแรกเป็นเครื่องวัดและบันทึกค่าพีเอชที่สามารถบันทึกค่าพีเอชได้ 24 ชม. ถ้าเก็บข้อมูลทุกๆ 15 วินาที มีคีย์บันทึกอาการต่างๆของผู้ป่วยที่เกิดขึ้นในระหว่างการวัด 6 คีย์ ส่วนที่สองเป็นโปรแกรมวิเคราะห์ผลซึ่งสามารถวิเคราะห์พารามิเตอร์ต่างๆที่ช่วยให้แพทย์วินิจฉัยโรค GER ได้ จากการทดลองพบว่าทั้งเครื่องวัดและโปรแกรมสามารถทำงานได้อย่างถูกต้อง และเมื่อนำเครื่องต้นแบบไปทดสอบกับอาสาสมัครคนปกติ พบว่าเครื่องต้นแบบสามารถวัดค่าพีเอชได้ใกล้เคียงกับเครื่องที่สั่งซื้อมาจากต่างประเทศ รวมทั้งผลการวิเคราะห์ก็ให้ผลเหมือนกัน จึงสรุปได้ว่าระบบที่พัฒนาขึ้นสามารถทำงานได้ดี และให้ผลเหมือนกับระบบที่สั่งซื้อมาจากต่างประเทศ

กิตติกรรมประกาศ

งานวิจัยนี้ได้รับการนับสนุนจากศูนย์เทคโนโลยีอิเล็กทรอนิกส์และคอมพิวเตอร์แห่งชาติ

เอกสารอ้างอิง

[1] ชูศักดิ์ ลิ่มสกุล, บุญเจริญ วงศ์กิตติศึกษา, เสกสิตโอสถากุล, และ สามารถ เครือทอง. 2547. โครงการพัฒนาระบบตรวจวัดภาวะกรดด่างในหลอดอาหาร. รายงานวิจัยฉบับสมบูรณ์. คณะวิศวกรรมศาสตร์ มหาวิทยาลัยสงขลานครินทร์.

[2] Wobschall D. 1987. Circuit Design for Electronic Instrumentation. Second Edition. RR. Donnelley &Sons Company.

[3[Vandenplas Y., *et al.* 1992. A Standardized Protocol for the Methodology of Esophageal pH Monitoring and Interpretation of the Data for the Diagnosis of Gastroesophageal Reflux. Journal of Pediatric Gastroenterology and Nutrition. 14:467-471.

[4[McLauchlan G., *et al.* 1987. Electrodes for 24 Hour pH Monitoring - A Comparative Study. GUT. 28,935-939.

[5] Emde C., Garner A. and Blum AL. 1987. Technical Aspects of Intraluminal pH-Metry in Man : Current Status and Recommendation. Progree report. GUT. 28,1177-1188.