ปีการศึกษา 2533

การหาค่าเฉลี่ย RMS ด้วยไมโครโปรเซสเซอร์

โดย

นางสาวจินตนา ควรทรงธรรม 301038

อาจารย์ที่ปรึกษา

อาจารย์ ธำรงศักดิ์ สุกใส

ปริญญานิพนธ์ปีการศึกษา 2533

ภาควิชา วิศวกรรมระบบควบคุม

คณะวิศวกรรมศาสตร์ สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

เรื่อง การหาค่าเฉลี่ย RMS ด้วยไมโครโปรเซสเซอร์

ผู้จัดทำ

1. นางสาวจินตนา ควรทรงธรรม 301038

.....อาจารย์ที่ปรึกษา

(อาจารย์ ธำรงศักดิ์ สุกใส)

การหาค่าเฉลี่ย RMS ด้วย ไมโครโปรเซสเซอร์

จินตนา ควรทรงธรรม
อาจารย์ธำรงศักดิ์ สุกใส อาจารย์ที่ปรึกษา
ปีการศึกษา 2533

บทคัดย่อ

ค่า RMS เป็นค่าที่ยอมรับกันทั่วไปในการบอกระดับของสัญญาณ AC ปริญญานิพนธ์ฉบับนี้กล่าวถึงการหาค่า RMS ของสัญญาณ AC แบบดิจิตอลโดยใช้วิธีการสุ่มตัวอย่าง(sampling) สัญญาณอินพุทในช่วงเวลาที่เท่าๆกันของ 1 คาบของสัญญาณ แล้วนำไปเก็บ และประมวลผลในไมโครคอนโทรลเลอร์ 8031

ถึงแม้ว่าวิธีนี้จะมีความยุ่งยากและซับซ้อนกว่าวิธีทางอนาลอกก็ตาม แต่ก็มีข้อดีกว่าตรงที่ จะสามารถหาค่า RMS ของสัญญาณอินพุททุกรูปสัญญาณ โดยไม่ขึ้นกับลักษณะของสัญญาณอินพุท ไม่ว่าจะเป็นรูป sine หรือไม่ก็ตาม เพราะเราใช้เทคนิคการสุ่มตัวอย่าง(sampling) เข้าช่วย และยังมีความเที่ยงตรงสูงอีกด้วย

RMS VALUE BY MICROPROCESSOR

Jintana Kuansongtham

Thamrongsak Suksai Adviser

1990

ABSTRACT

This thesis describe the method for finding RMS value of AC signal in digital mode by sampling the input signal in 1 cycle and process in 8031 microcontroller

Eventhough this method is more difficult and more complex than analog method, it has an advantage in the aspect that it can be calculate the RMS of any input signal, not depending on the shape of the input signal which can be sine wave or not. Because we use the sampling technique and it has high accuracy as well.

สารบัญ

สารบัญรูป

สารบัญตาราง

บทที่ 1

บทนำ

ในปัจจุบัน เครื่องวัดสัญญาณไฟฟ้ากระแสสลับ ที่วัดค่า RMS จริงๆนั้น ยังมีน้อย หรือหากมีก็มีราคาแพง ส่วนใหญ่เครื่องวัดที่ใช้กันอยู่ มักจะวัด โดยขึ้นอยู่บนมาตรฐานของสัญญาณ sine หากสัญญาณที่ต้องการจะวัดมิใช่สัญญาณ sine แต่เป็นสัญญาณรูปสี่เหลี่ยม สามเหลี่ยม หรือ สัญญาณรูปอื่นๆ ค่าที่ได้ก็จะผิดพลาดไป โครงงานนี้ ก็จะเสนอเทคนิคในการวัดสัญญาณกระแสสลับ โดยที่สัญญาณที่วัดได้ก็จะเป็นสัญญาณ RMS จริงๆ เพราะได้มาจาก การสุ่มตัวอย่าง สัญญาณจนครบ 1 คาบ แล้วนำมาคำนวณหาค่า RMS ตามสูตร คือ หาผลรวมของค่าสัญญาณที่สุ่มได้ยกกำลังสอง แล้วถอดรากที่สอง จากนั้นก็นำไปหารด้วยคาบเวลา ซึ่งก็คือจำนวนครั้งของการสุ่มตัวอย่าง นั่นเอง

การประมวลผล ในที่นี้เราใช้ 8031 ไมโครคอนโทรลเลอร์ เพราะมันเหมาะสมกับงานทางด้านควบคุม วงจรที่ใช้ได้อธิบายไว้อย่างละเอียดในบทที่ 3 พร้อมรูป ทางซอฟท์แวร์ ใช้ภาษาเครื่องของ 8031 โดยตรง

บทที่ 2

ทฤษฎี หรือ หลักการ

2.1 หลักการหาค่า RMS

ค่า RMS ของสัญญาณรายคาบทุกชนิด หาได้จาก

$$V_{rms} = \frac{1}{T}\sqrt{\int V^2 \, dt} \quad \text{----------(1)}$$

ซึ่งเราสามารถแปลงสมการ (1) ให้อยู่ในรูปผลรวม (Summation) ตั้งแต่ i=0 ถึง N ได้ดังนี้

$$V_{rms} = \frac{1}{T}\sqrt{\sum_{i=0}^{N} (V_i)^2} \quad \text{----------(2)}$$

โดยที่ V_i = ศักดาที่ต้องการหาค่า RMS

N = จำนวนครั้งของการสุ่มตัวอย่าง ใน 1 คาบ

จากสมการที่ (2) เราพบว่า หากเราทำการสุ่มตัวอย่างสัญญาณที่เราต้องการหาค่า RMS มา N ค่า จากนั้นทำการยกกำลังสองทุกๆค่า นำมาบวกกัน แล้วถอดรากที่สอง และหารด้วยจำนวนครั้ง ของการสุ่มตัวอย่าง ผลลัพธ์ที่ได้ ก็คือค่า RMS ของสัญญาณนั้นๆ

2.2 <u>หลักการหาค่ารากที่สองโดยการทำซ้ำ</u> (Iteration Method)

วิธีหาค่ารากที่ n ใดๆ ที่ n เป็นค่าบวก จะเริ่มจากสมการ

$$X = Y^{(1/n)} \quad \text{----------(3)}$$

โดยที่ Y เป็นค่าที่ต้องการหารากที่ n

X เป็นค่ารากที่ n ของ Y

สมการที่ (3) สามารถเขียนใหม่ได้เป็น

$$Y = X^n \quad \text{----------(4)}$$

เรานิยาม ฟังก์ชั่น f(X) เป็น

$$f(X) = (Y - X^n) \quad \text{----------(5)}$$

ที่ค่า X ที่เราต้องการ(ค่าที่เป็นรากที่ n ของ Y)ซึ่งให้เป็น X_o ดังนั้น $f(X_o) = 0$ แต่ค่า X_o เราไม่สามารถรู้ได้ก่อน ดังนั้น เราจึงต้องทำการเดาค่า X ค่า X ที่เราเดากับค่า X ที่เราต้องการ จะมีความสัมพันธ์กันตาม อนุกรมเทเลอร์(Taylor Series Expansion) รอบจุด X

$$f(X_o) = 0 = f(X) + f(X_o - X)df/dx\big|_x + ... \quad \text{------(6)}$$

เนื่องจาก $df/dx = -nX^{(n-1)}$ เราจึงได้ค่าประมาณของสมการที่ (6) เป็น

$$0 = Y - X^n - (X_o - X).nX^{(n-1)}$$

เมื่อจัดพจน์ใหม่ จะได้

$$X_{o} = X_{o} = \frac{Y + (n-1)\,X^{n}}{n.X^{(n-1)}} \qquad \text{----------(7a)}$$

$$X_{o} = X_{o} = X + \frac{1}{n.X^{(n-1)}} \cdot (Y - X^{n}) \qquad \text{---------(7b)}$$

จากการเดาค่า X ในครั้งแรก นำไปแทนในสมการ (7) จะสามารถหาค่าประมาณที่ดีขึ้น เป็นค่า X_{o} ค่า X_{o} นี้ สามารถแทน X ได้ และหาค่า X_{o} ต่อไปเรื่อยๆ จนถึงระดับความเที่ยงตรงที่ยอมรับได้ ก็จะได้ค่า X ที่เป็น รากที่ n ของ Y ในทางปฏิบัติความเที่ยงตรงจะถูกจำกัด โดยความแม่นยำในการคำนวณ (Numerical Precision)

สำหรับกรณี n=2 , $X = \sqrt{Y}$ สมการทำซ้ำ (Iteration Equation) จะเป็น

$$X_{new} = \frac{1}{2}.(X_{old} + Y/X_{old}) \qquad \text{----------(8)}$$

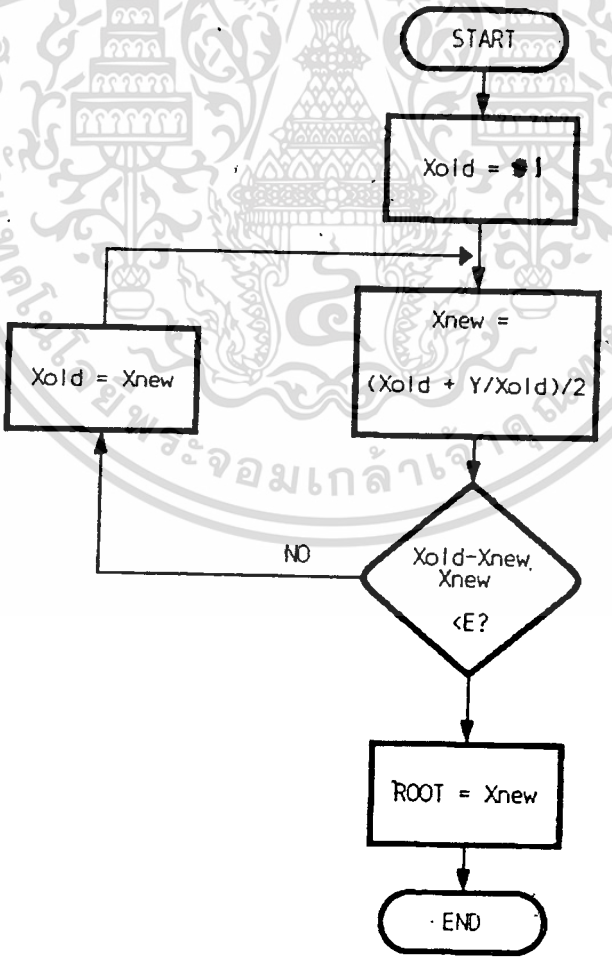
สมการกรณีพิเศษนี้ เรียกว่า สูตรของเฮรอน (Heron's Formula) ลำดับของการทำซ้ำ เพื่อหาค่ารากที่สอง ของ 2 ซึ่งได้จากการใช้สูตรของเฮรอน แสดงในตารางที่ 2.1 ซึ่งจะเห็นได้ว่า มีการลู่เข้าเร็วมาก

Iteration Step	Estimate	Error ($\sqrt{2}$ − estimate)
0	1.0	0.4
1	1.5	-0.1
2	1.416666666666667	-2×10^{-3}
3	1.41421568627451	-2×10^{-6}
4	1.41421356237469	-2×10^{-12}
5	1.414213562373095	10^{-15} (all digits shown are correct)

Table 3.2.1 *The sequence of approximations associated with finding the square root of 2 using Heron's Rule. The initial guess was X = 1.*

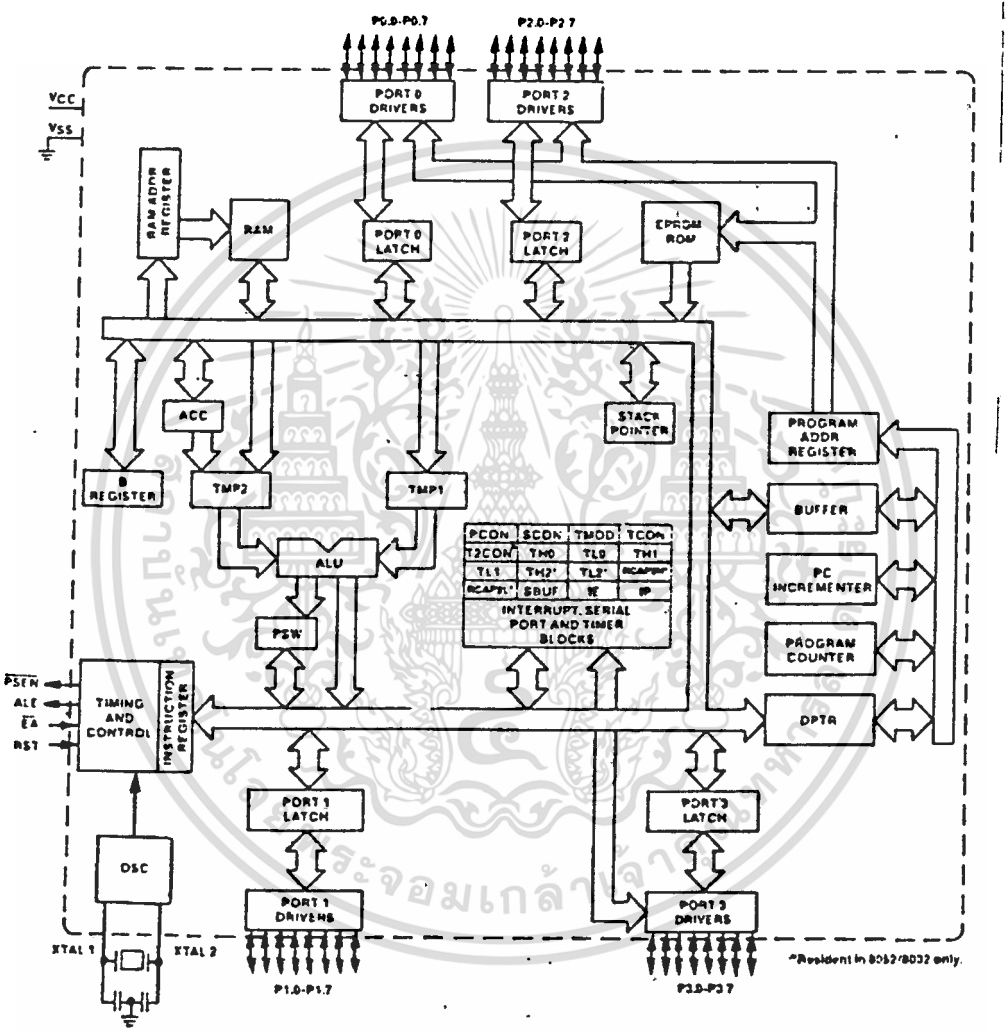
ตารางที่ 2.2.1 แสดงการหาค่ารากที่สอง ของ 2

โดยใช้สูตรของ เฮรอน

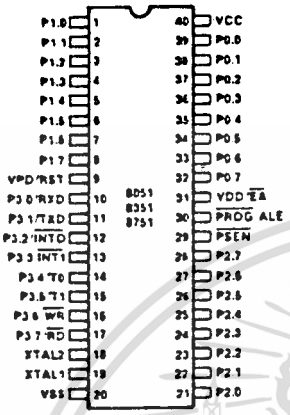


รูปที่ 2.2.1 แสดงโฟลว์ชาร์ต การหาค่ารากที่สองของ เฮรอน

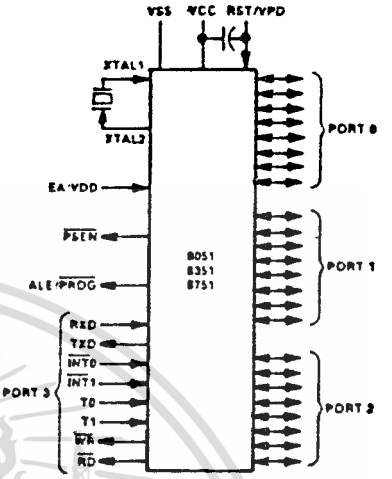
2.3 8031 ไมโครคอนโทรเลอร์



รูปที่ 2.3.1 แสดงสถาปัตยกรรมของ 8031



รูปที่ 2.3.2 แสดงการจัดขาของ 8031



รูปที่ 2.3.3 แสดงสัญลักษณ์ทางตรรก

2.3.1 คุณสมบัติ

- ต้องการไฟเลี้ยง 5 โวลต์
- มีหน่วยความจำภายในสำหรับข้อมูล 128 ไบต์
- 4 รีจีสเตอร์ แบงค์
- ต่อกับหน่วยความจำภายนอกได้ 64 ไบต์
- คำสั่ง 1 ไซเคิล กินเวลา 1 s ถ้าใช้คริสตอล 12 MHz
- บัส อินพุท/เอาท์พุท 2 ทิศทาง 16 เส้น
- ไทม์เมอร์/เคาท์เตอร์ 16 บิต 2 ชุด
- พอร์ทอนุกรม ซึ่งสามารถโปรแกรมได้
- อ้างแอดเดรสแบบบิต และแบบไบต์ได้
- กระทำการแบบบิตได้
- กำหนดความสำคัญของอินเทอร์รัพท์ได้ 2 ระดับ

2.3.2 สถาปัตยกรรม และ การจัดการ

ไมโครคอมพิวเตอร์ ประกอบด้วย หน่วยประมวลผลกลาง หน่วยความจำ 2 ชนิด (หน่วยความจำชั่วคราว และ หน่วยความจำถาวร) อินพุท/เอาท์พุทพอร์ท รีจีสเตอร์เก็บข้อมูลและสถานะต่างๆ ส่วนประกอบเหล่านี้ ติดต่อกันผ่านทางบัสข้อมูล 8 บิท บัสที่กล่าวถึงจะมีบัฟเฟอร์กับภายนอก ผ่านทางI/O พอร์ท

หน่วยประมวลผลกลาง (Central Processing Unit : CPU)

CPU ถือเป็นหัวสมองของไมโครคอมพิวเตอร์ ทำหน้าที่อ่านโปรแกรมของผู้ใช้ และปฏิบัติตามคำสั่งนั้น ส่วนประกอบเบื้องต้นของมันคือ ALU (Arithmatic Logic Unit) 8 บิท ซึ่งติดต่อกับ รีจีสเตอร์ A,B,PSW,SP และ โปรแกรมเคาท์เตอร์ 16 บิต และ รีจีสเตอร์ DPTR

หน่วยคำนวณ และกระทำกระบวนการทางตรรก (Arithmatic Logic Unit:ALU)

ALU สามารถกระทำการคำนวณ และกระบวนการทางตรรก กับตัวแปร 8 บิท การคำนวณ หมายถึง การบวก การลบ การคูณ และการหาร ส่วนกระบวนการทางตรรก หมายถึง การ AND,OR,EXCLUSIVE-OR รวมทั้งการหมุนบิท เคลียร์บิท ทำคอมพลีเมนต์ ฯลฯ นอกจากนี้ ALU ยังกระทำการตัดสินใจตามเงื่อนไขที่กำหนดให้อีกด้วย

สิ่งที่พิเศษสำหรับ สถาปัตยกรรมของ 8031 คือ ALU สามารถกระทำกระบวนการกับข้อมูลแบบบิตเดียวได้ เช่นเดียวกับข้อมูลแบบ 8 บิต แต่ละบิตสามารถจะถูก เซ็ต เคลียร์ คอมพลีเมนต์ เคลื่อนย้าย ทดสอบ หรือใช้ในการคำนวณทางตรรกได้ คุณสมบัติข้อนี้ ทำให้ 8031 เหมาะสมกับงานทางระบบควบคุม

ชุดคำสั่ง (Instruction Set)

ชุดคำสั่ง สามารถแบ่งได้เป็น 5 กลุ่ม ดังนี้

1.ชุดคำสั่งทางด้านการคำนวณ

2.ชุดคำสั่งการคำนวณทางตรรก

3.ชุดคำสั่งการโอนย้ายข้อมูล

4.ชุดคำสั่งการกระทำการแบบบิต

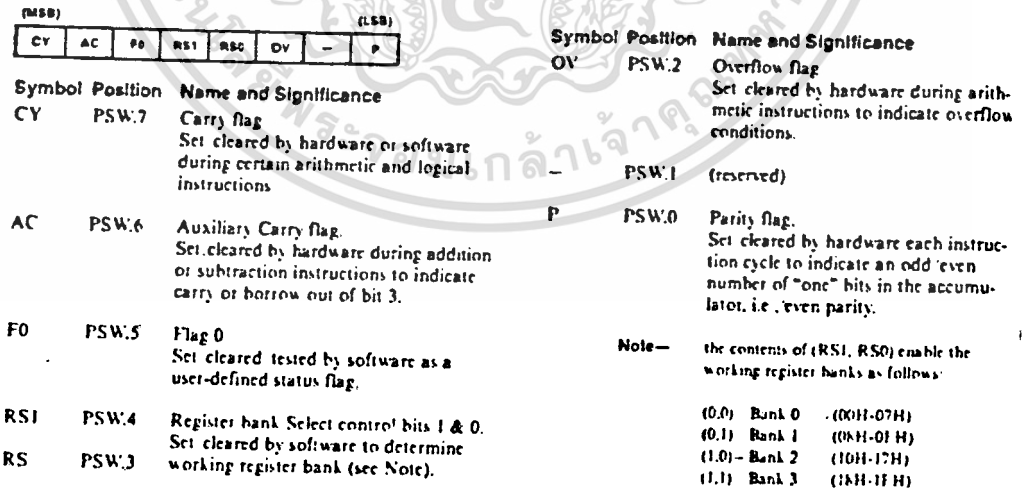
5.ชุดคำสั่งเกี่ยวกับเงื่อนไข และการกระโดด

ชุดคำสั่งของ 8031 ถูกออกแบบมาเพื่อให้โปรแกรมมีประสิทธิภาพที่สุด ทั้งในเรื่องขนาดของคำสั่ง และความเร็วในการคำนวณ ขนาดของคำสั่ง 1 คำสั่งจะไม่เกิน 3 ไบต์ แต่โดยส่วนใหญ่คำสั่งจะมีขนาดไม่เกิน 1-2 ไบต์ ในทางปฏิบัติ คำสั่งแต่ละคำสั่ง จะถูกปฏิบัติภายใน 1 หรือ 2 ไซเคิล นั่นคือ ภายใน 1-2 ไมโครวินาที เมื่อใช้คริสตอล 12 MHz ยกเว้นเพียงคำสั่งการคูณและการหารเท่านั้น ที่กินเวลาถึง 4 ไซเคิล

Accumulator และ PSW

8031 มีสถาปัตยกรรมซึ่งมีรากฐานอยู่บน แอกคูมูเลเตอร์ 8 บิท เรียกว่า "A" ซึ่งจะทำหน้าที่ เก็บข้อมูลชั่วคราวก่อนการคำนวณ (ก็คือตัวตั้ง) และเก็บผลที่ได้จากการคำนวณด้วย เช่น การบวก ลบ คูณ หาร การกระทำการทางตรรก และนอกจากนี้ มันยังช่วยในการโอนย้ายข้อมูลแบบพิเศษอีกด้วย มีคำสั่งมากมายที่กระทำกับ "A" เท่านั้น

คำสั่งหลายชุด จะมีผลต่อแฟลกสถานะ ซึ่งถูกจัดกลุ่มให้อยู่รวมกัน ในรูปของ Program Status Word : PSW ดังแสดงในรูป



รูปที่ 2.3.4 แสดง PSW

รีจีสเตอร์อื่นๆ

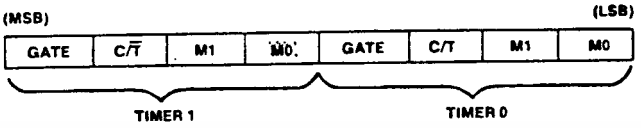
"B" เป็นรีจิสเตอร์ 8 บิท ทำหน้าที่ในการคูณและหารร่วมกับ "A" ในการเก็บอินพุทตัวที่สอง และเก็บผลลัพธ์ 8 บิทของการคูณและหารด้วย

"SP" (Stack Pointer) เป็นรีจีสเตอร์ขนาด 8 บิทที่ทำหน้าที่ชี้ตำแหน่งของแสต็ก ซึ่งจะบอกถึง แอดเดรสสุดท้ายที่ได้ถูกทำการ PUSH มาเก็บไว้ในแสต็ก SP จะเพิ่มค่าและลดค่าโดยอัตโนมัติด้วยคำสั่ง PUSH หรือ POP และการเรียกโปรแกรมย่อยกับการรีเทอร์น เมื่อทำการรีเซ็ต SP จะชี้อยู่ที่ 07H เสมอ

"DPTR"(Data Pointer) เป็นรีจีสเตอร์ 16 บิท ใช้ในการเปิดตาราง การกระโดดแบบไม่โดยตรง และการโอนย้ายข้อมูลภายนอก ไบต์บน และ ไบต์ล่างของ DPTR สามารถแยกกระทำการได้โดยอิสระ เรียก DPH และ DPL ตามลำดับ

2.3.3 Timer/Counter

8031 มี Timer/Counter ภายใน 2 ตัว ขนาด 16 บิท และสามารถเลือกโหมดได้หลายโหมดขึ้นกับการประยุกต์ใช้งาน Timer/Counter แต่ละตัวประกอบด้วยไบต์สูงและไบต์ต่ำ เรียกว่า TH0,TL0 และ TH1,TL1 แต่ละคู่สามารถเลือกโหมดได้อย่างอิสระ ภายใต้รีจิสเตอร์ที่ทำการกำหนดโหมด เรียกว่า TMOD และจะสามารถควบคุมได้ด้วยรีจิสเตอร์ TCON

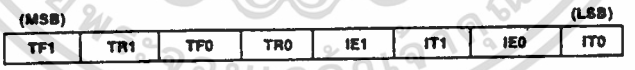


GATE — Gating control When set, Timer/counter "x" is enabled only while "INTx" pin is high and "TRx" control pin is set. When cleared Timer "x" is enabled whenever "TRx" control bit is set

C/$\overline{T}$ — Timer or Counter Selector Cleared for Timer operation (input from internal system clock). Set for Counter operation (input from "Tx" input pin).

M1	M0	Operating Mode
0	0	MCS-48 Timer "TLx" serves as five-bit prescaler.
0	1	16 bit Timer/Counter "THx" and "TLx" are cascaded; there is no prescaler
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-bit timer counter-controlled by the standard Timer 0 control bits TH0 is an eight-bit timer only controlled by Timer 1 control bits.
1	1	(Timer 1) Timer-counter 1 stopped.

รูปที่ 2.3.5 แสดง TMOD

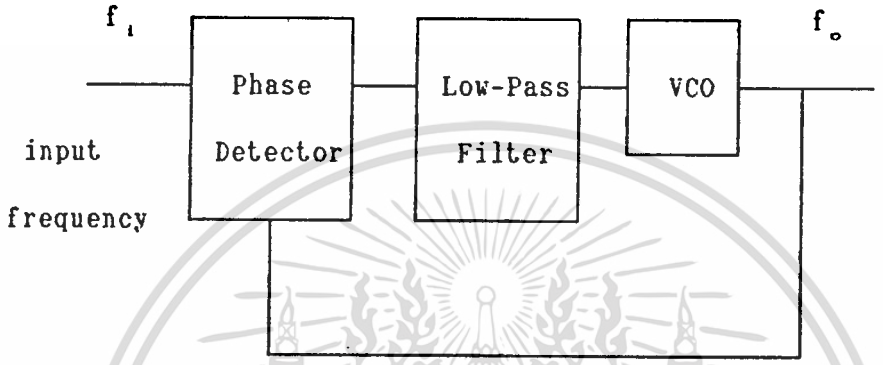


Symbol	Position	Name and Significance	Symbol	Position	Name and Significance
TF1	TCON.7	Timer 1 overflow Flag. Set by hardware on timer/counter overflow. Cleared by hardware when processor vectors to interrupt routine.	IE1	TCON.3	Interrupt 1 Edge flag. Set by hardware when external interrupt edge detected. Cleared when interrupt processed.
TR1	TCON.6	Timer 1 Run control bit. Set/cleared by software to turn timer/counter on/off.	IT1	TCON.2	Interrupt 1 Type control bit. Set/cleared by software to specify falling edge/low level triggered external interrupts.
TF0	TCON.5	Timer 0 overflow Flag. Set by hardware on timer/counter overflow. Cleared by hardware when processor vectors to interrupt routine.	IE0	TCON.1	Interrupt 0 Edge flag. Set by hardware when external interrupt edge detected. Cleared when interrupt processed.
TR0	TCON.4	Timer 0 Run control bit. Set/cleared by software to turn timer/counter on/off.	IT0	TCON.0	Interrupt 0 Type control bit. Set/cleared by software to specify falling edge/low level triggered external interrupts.

รูปที่ 2.3.6 แสดง TCON

2.4 <u>เฟสล็อคลูป</u> (Phase Lock Loop)

เฟสล็อกลูป ประกอบด้วย ตัวเปรียบเทียบเฟส ตัวกรองผ่านความถี่ต่ำ และ VCO ซึ่งประกอบกันแสดงดังบล็อกไดอาแกรม รูป 2.4.1



รูปที่ 2.4.1 แสดงเฟสล็อกลูป

จากรูป จะเห็นว่า ส่วนประกอบทั้งสามอยู่ในส่วน ฟีดฟอร์เวอร์ด ขณะที่ในส่วนป้อนกลับเป็นเพียงการเชื่อมต่อกันระหว่าง VCO และ ตัวเปรียบเทียบเฟส

VCO เป็นออสซิลเลเตอร์ โดยที่ความถี่จะถูกกำหนดด้วย ความต้านทานหรือตัวเก็บประจุภายนอกความถี่ VCO (f_o) จะถูกป้อนกลับไปยังตัวเปรียบเทียบเฟส ซึ่งมันจะถูกเปรียบเทียบ กับความถี่อินพุท (f_i) เอาท์พุทของตัวเปรียบเทียบเฟส คือ ค่าศักดาผิดพลาด (Error Voltage) ซึ่งก็คือ ศักดาไฟฟ้ากระแสตรงเฉลี่ย ที่เป็นสัดส่วนโดยตรงกับความต่างของความถี่ ($f_i - f_o$) และ เฟส($\Delta\phi$) ของอินพุทและ VCO

จากนั้น ศักดาผิดพลาดจะถูกกรองเพื่อกำจัดสัญญาณรบกวน และป้อนสู่ VCO ศักดาผิดพลาดจะทำการบังคับความถี่ของ VCO ให้เป็นไปในทางที่จะลดความแตกต่างของความถี่ระหว่าง f_i และ f_o เมื่อ VCO เริ่มเปลี่ยนความถี่ ลูปก็จะอยู่ในสภาวะติดตาม(Capture State) กระบวนการนี้จะดำเนินต่อไปเรื่อยๆ จนกระทั่งความถี่ VCO และ ความถี่อินพุทมีค่าเท่ากันพอดี ที่จุดนี้ ลูปจะอยู่ในภาวะล็อค(Phas-Locked) ระหว่างภาวะล็อค ความถี่ VCO จะเท่ากับความถี่อินพุทของลูป แต่จะต้องมีความต่างเฟสอยู่บ้าง ซึ่งจำเป็นสำหรับการสร้างศักดาผิดพลาด เพื่อทำให้ลูปยังคงล็อคอยู่

จะเห็นว่า ระบบเฟสล็อคลูป มี 3 สภาวะ ที่แตกต่างกัน ดังนี้

1. สภาวะวิ่งโดยอิสระ (Free - running)

2. สภาวะติดตาม (Capture)

3. สภาวะล็อค (Phase - lock)

คุณลักษณะพลวัตร (Dynamic Characteristic) ของระบบ ถูกควบคุมด้วยตัวกรอง ถ้าความแตกต่างระหว่างความถี่ VCO กับ ความถี่อินพุท มีค่ามากเกินไป ผลของสัญญาณผิดพลาดอาจจะสูงเกินไปที่จะผ่านตัวกรองไปได้ สิ่งที่ตามมาก็คือ สัญญาณจะไม่อยู่ในช่วงพิสัยการล็อค

2.4.1 <u>ตัวเปรียบเทียบเฟส</u> (Phase Detector)

ตัวเปรียบเทียบเฟส สร้างศักดาไฟฟ้ากระแสตรง ที่เป็นสัดส่วนโดยตรงกับ ความต่างเฟสระหว่าง สัญญาณอินพุทของระบบ PLL กับสัญญาณจาก VCO ค่าศักดานี้ มักจะเรียกกันในนามของศักดาผิดพลาด

$$V_o = K_d . \Delta\phi \quad \text{---------(9)}$$

V_o = ค่าเฉลี่ยศักดาเอาท์พุทของตัวเปรียบเทียบเฟส (โวลต์)

K_d = เกนของตัวเปรียบเทียบเฟส (โวลต์/เรเดียน)

$\Delta\phi$ = ความต่างเฟส (เรเดียน)

2.4.2 VCO (Voltage Control Oscillator)

VCO ทำหน้าที่สร้างความถี่เอาท์พุท ซึ่งมีสัดส่วนโดยตรงกับศักดาอินพุท VCO เรียกอีกอย่างได้ว่า ตัวเปลี่ยนศักดาเป็นความถี่ (Voltage to Frequency Convertor)

$$W_o = K_o.V_f \quad \text{----------(10)}$$

W_o = ความถี่เอาท์พุทของ VCO (เรเดียน/วินาที)

V_f = ศักดาอินพุทจากตัวกรองผ่านความถี่ต่ำ (โวลต์)

K_o = เกนของ VCO (เรเดียน/วินาที/โวลต์)

2.4.3 ตัวกรองผ่านความถี่ต่ำ (Low Pass Filter)

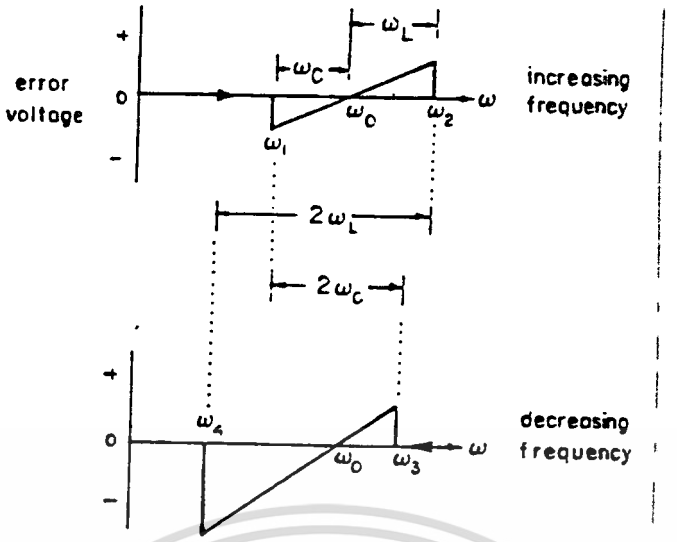
ตัวกรองผ่านความถี่ต่ำ ทำหน้าที่กำจัดสัญญาณรบกวน และส่วนประกอบความถี่สูง (High Frequency Component) จากศักดาเอาท์พุทของตัวเปรียบเทียบเฟส และให้ศักดาไฟฟ้ากระแสตรงเฉลี่ย (dc) นอกจากนี้ ตัวกรองผ่านความถี่ต่ำยังเป็นตัวกำหนดสมรรถนะพลวัตร (Dynamic Performance) ของระบบ ซึ่งรวมทั้งแฟคเตอร์ต่อไปนี้

- พิสัยเข้าสู่ภาวะล็อค และพิสัยล็อค (Capture & Lock Range)
- แบนวิทธ์ (Bandwith)
- ผลตอบสนองชั่วคราว (Transient Response)

2.4.4 พิสัยเข้าสู่ภาวะล็อก และพิสัยล็อค (Capture & Lock Range)

พิสัยล็อค ($2W_L$) ของระบบเฟสล็อคลูป คือช่วงความถี่ของระบบ ที่จะตามการเปลี่ยนแปลงของความถี่อินพุท

พิสัยเข้าสู่ภาวะล็อค ($2W_C$) คือช่วงที่เฟสล็อคลูปเข้าสู่ภาวะล็อค



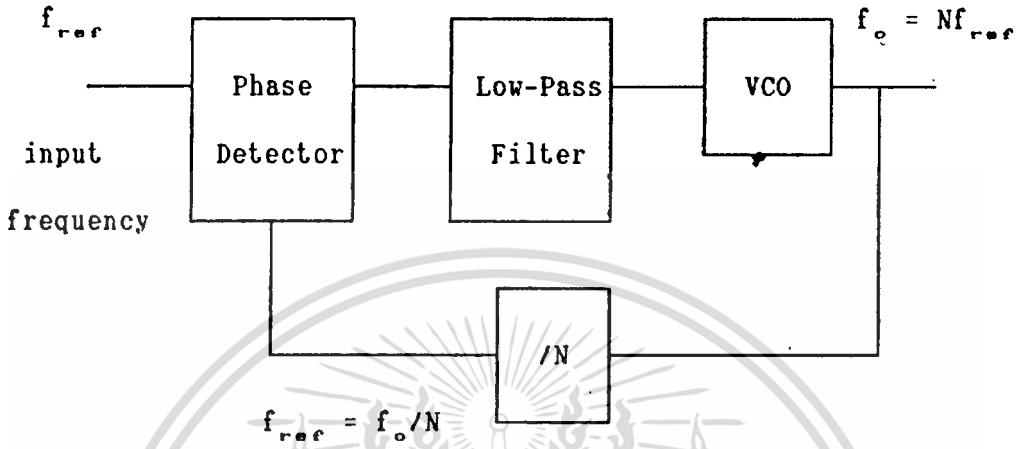
รูปที่ 2.4.2 พิสัยเข้าสู่ภาวะล็อค และพิสัยล็อค

lock range : $2W_L = W_2 - W_4$

capture range : $2W_c = W_3 - W_4$

จากรูปบน ความถี่อินพุท (W_1) จะค่อยๆเพิ่มขึ้นจนกระทั่ง W_1 มีค่าเท่ากับ W_3 ซึ่งเป็นค่าต่ำสุดของพิสัยเข้าสู่ภาวะล็อค ลูปจะอยู่ในภาวะล็อค กับความถี่อินพุท ทำให้ศักดาผิดพลาดของลูปเป็นค่าลบ เมื่อความถี่อินพุทมีค่าเพิ่มขึ้น ศักดาผิดพลาดก็จะค่อยๆเพิ่มขึ้นอย่างเป็นเชิงเส้นกับส่วนกลับของเกนของ VCO หรือ $1/K_o$ เมื่อความถี่อินพุทเท่ากับความถี่ VCO ศูนย์กลาง (VCO free - running frequency) ค่าศักดาผิดพลาดจะเป็นศูนย์ ลูปก็จะยังคงแทร็คตามความถี่อินพุท จนกระทั่งถึงความถี่ W_2 ซึ่งเป็นค่าสูงสุดของพิสัยการล็อค เมื่อความถี่อินพุทมีค่ามากกว่า W_2 ลูปจะหลุดภาวะการล็อค ศักดาผิดพลาดจะกลับเป็นศูนย์ VCO ก็จะกลับไปอยู่ที่ความถี่ศูนย์กลาง เมื่อความถี่อินพุทลดลง กระบวนการเดิมก็จะเกิดขึ้นอีก ต่างกันเพียงแต่ว่า ค่าศักดาผิดพลาดจะเป็นบวกที่ W_3 ซึ่งก็คือค่าสูงสุดของพิสัยเข้าสู่ภาวะล็อค

2.4.5 <u>ตัวคูณความถี่</u> (Frequency Multiplier)



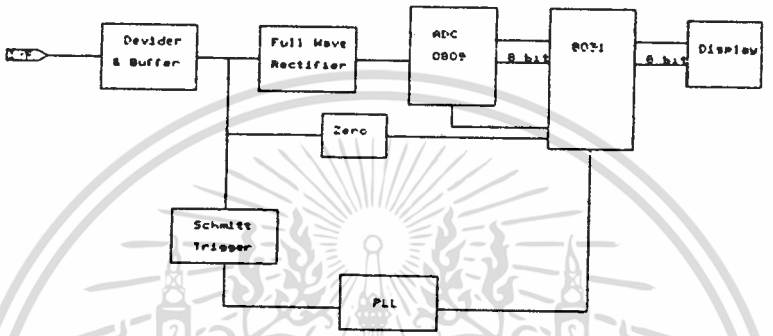
รูปที่ 2.4.3 แสดงบล็อคไดอาแกรมของตัวคูณความถี่

โดยพื้นฐานแล้ว ตัวคูณความถี่ เป็นแหล่งจ่ายความถี่ซึ่งเอาท์พุทจะเป็นจำนวนเต็มเท่าของความถี่อินพุทอ้างอิง

จากรูป ตัวคูณความถี่สร้างมาจากวงจรเฟสล็อคลูป โดยทำการแทรกวงจรหาร N เข้าไประหว่างส่วน VCO กับส่วนเปรียบเทียบเฟส เมื่อเราทำการเปรียบเทียบกับรูป 2.4.1 จะพบว่า ส่วนเปรียบเทียบเฟสของตัวคูณความถี่ จะสร้างศักดาเฉลี่ยที่เป็นสัดส่วนโดยตรงกับความต่างเฟสระหว่างความถี่อินพุทอ้างอิง (f_{ref}) และความถี่เอาท์พุทจากส่วนหาร N (f_o/N) ส่วนหาร N จะทำการสร้างเอาท์พุทพัลส์ 1 ลูก สำหรับ ทุกๆ N อินพุทพัลส์ เอาท์พุทของตัวเปรียบเทียบเฟส หลังจากถูกทำการกรองแล้ว ก็จะถูกส่งไปควบคุมความถี่เอาท์พุทของ VCO (f_o) ซึ่งเท่ากับ N เท่าของความถี่อินพุทอ้างอิง

บทที่ 3

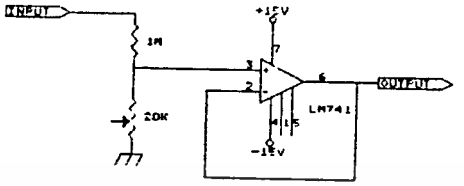
การออกแบบ และ การสร้าง



การทำงานของวงจรทั้งหมดโดยย่อเป็นดังนี้

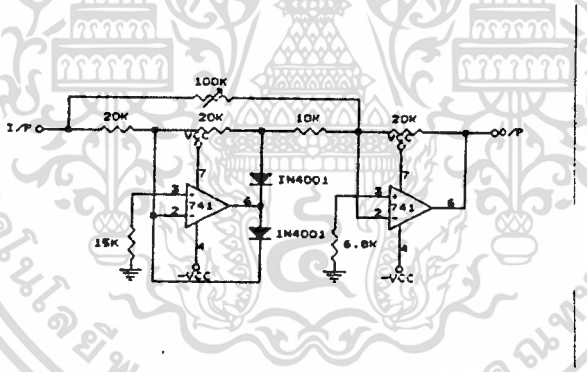
สัญญาณอินพุท (V_i) หลังจากทำการหารศักดาไฟฟ้าแล้ว ก็ถูกส่งเข้าวงจรดัดกระแสแบบเต็มคลื่น เพื่อทำการแปลงสัญญาณนี้ให้มีแต่ชีกบวก แล้วนำเข้าวงจรแปลงให้เป็นดิจิตอล ซึ่งรับอินพุทได้ในระดับ 0 ถึง 5 โวลต์ นอกจากนี้ สัญญาณอินพุทก็ยังถูกป้อนเข้าวงจรเช็คระดับศูนย์โวลต์ เพื่อทำการตรวจจับสัญญาณอินพุทว่าเป็นบวกหรือลบ แล้วบอก 8031 และสัญญาณอินพุทก็ถูกป้อนเข้าวงจร ชมิทท์ ทริกเกอร์ เพื่อทำการแปลงสัญญาณอินพุท แต่ละไซเคิลให้เป็นสัญญาณรูปสี่เหลี่ยมที่มีดิวตี้ไซเคิล 50% ป้อนให้กับ PLL ส่วน PLL นั้น ทำหน้าที่เป็นตัวทวีคูณความถี่ (Frequency Multiplier) ซึ่งจะทำการทวีคูณความถี่ จากความถี่อินพุทไปเป็นจำนวน N เท่า เพื่อใช้เป็นสัญญาณบอก 8031 ให้ทำการสุ่มตัวอย่าง เมื่อ 8031ได้รับสัญญาณให้สุ่มตัวอย่าง ก็จะส่งสัญญาณไปบอก ADC อีกต่อหนึ่ง ให้ ADC ทำการแปลงสัญญาณอินพุท ที่เป็นอนาลอกซึ่งมีแต่ชีกบวกจากวงจรดัดกระแสแบบเต็มคลื่น เป็นดิจิตอล 8 บิท ส่งให้ 8031 8031 ก็จะทำการเช็คว่าข้อมูลที่ได้รับนั้น เป็นบวกหรือเป็นลบจากวงจรเช็คระดับศูนย์โวลต์ เช่นนี้เรื่อยไปจนครบ 1 ไซเคิล เมื่อครบ 1 ไซเคิลแล้ว 8031 ก็จะเก็บค่าไว้ได้ N จำนวน เพื่อใช้ในการประมวลผลหาค่า RMS แล้วส่งออกแสดงผลที่ 7 เซ็กเมนต์

วงจรหารศักดาไฟฟ้า (Voltage Devider & Buffer)



วงจรทำการหารศักดาไฟฟ้าจาก 400 โวลต์ ให้เหลือ 5 โวลต์ และมีส่วนของบัฟเฟอร์ด้วย ซึ่งใช้ IC 741 ต่อให้มีกำลังขยายเป็นหนึ่ง

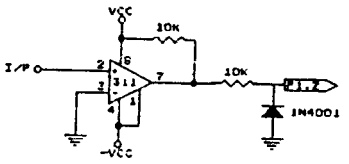
วงจรเรียงกระแสแบบเต็มคลื่น (Full Wave Rectifier)



วงจรนี้ประกอบด้วย ออปแอมป์ 2 ตัว ทำหน้าที่เป็น Inverting Amp กับ Summing Amp เหตุที่ต้องใช้ออปแอมป์ในการสร้างวงจร Rectifier ในครั้งนี้แทนที่จะใช้ Bridge Diode ก็เพราะเราต้องการวงจร Rectifier ที่มีกราวน์ร่วมกันระหว่างอินพุทกับ 8031

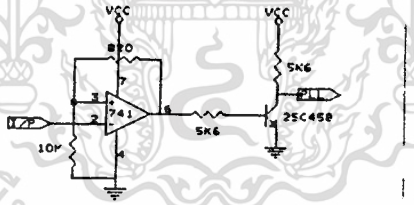
หลักการของวงจรนี้คือ ใช้การรวมเอาท์พุทของ half wave rectifier กับอินพุทของมัน จากรูป A1 เป็น Inverting Rectifier เอาท์พุทจาก A1 จะถูกรวมกันสัญญาณอินพุทใน A2 ซึ่งเป็น Summing Mixer ได้เอาท์พุทที่เป็น Full Wave ป้อนให้ ADC ต่อไป

วงจรเช็คระดับศูนย์โวลต์ (Zero-Crossing Detector)



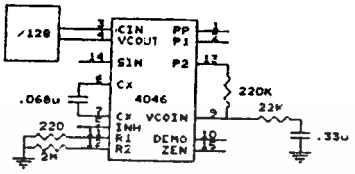
เป็นวงจรที่ทำหน้าที่ตรวจเช็คว่าสัญญาณอินพุทว่ามีค่าเป็นบวกหรือเป็นลบ แล้วบอก 8031 ประกอบด้วยออปแอมป์ 311 ทำหน้าที่เป็นคอมพาราเตอร์ แล้วใช้ไดโอดบล็อกช่วงลบทิ้งไป เพื่อให้เหลือระดับโวลต์เดจ 0 ถึง +5 โวลต์ เข้า 8031 ได้

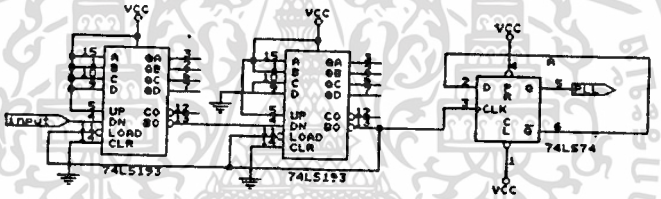
วงจรชมิทท์ ทริกเกอร์ (Schimtt Trigger)



วงจรนี้ทำหน้าที่เปลี่ยนสัญญาณอินพุท ซึ่งเป็นสัญญาณรายคาบรูปต่างๆ ให้เป็นสัญญาณรูปสี่เหลี่ยม มีขนาด 5 V และมี duty cycle 50% โดยใช้ 741 ทำหน้าที่เป็น Comparator ที่มี Hysteresis แปลงสัญญาณอินพุทให้เป็นสัญญาณรูปสี่เหลี่ยมที่มีขนาด + 5 V นำไปขับทรานซิสเตอร์ 2SC458 ที่ทำหน้าที่เป็นสวิตช์ ได้สัญญาณรูปสี่เหลี่ยมที่มีขนาด + 5 V ออกมาที่ขาคอลเลคเตอร์ ป้อนเข้าอินพุทของ PLL

วงจรคูณความถี่ (Frequency Multiplier)

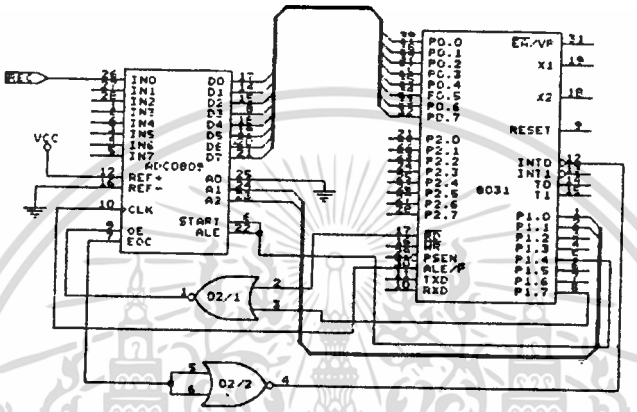




จากรูป ใช้ 4046 เป็น IC เฟสล็อคลูป ซึ่งกรณีนี้ใช้ทำหน้าที่เป็นตัวทวีคูณความถี่ (Frequency Multiplier) โดยจะทวีคูณความถี่ไปเป็นจำนวน 128 เท่าของความถี่อินพุท วงจรหาร 128 นั้น สร้างมาจาก วงจร counter ภายนอก โดยใช้ IC 74LS193 2 ตัว และ D/FF เบอร์ 74LS74 1 ตัว

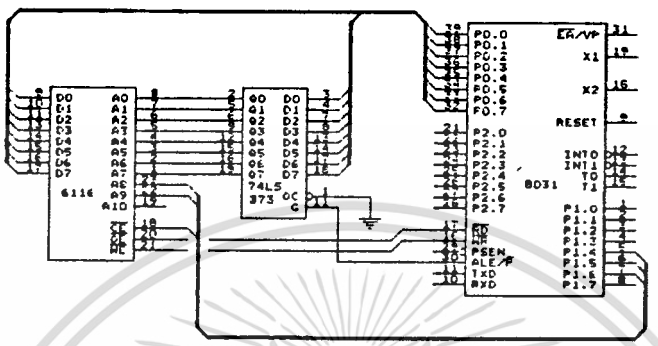
ผลการทดลองได้ช่วง Lock Range 2-120 Hz

วงจรแปลงอนาลอกเป็นดิจิตอล (Analog to Digital Converter)



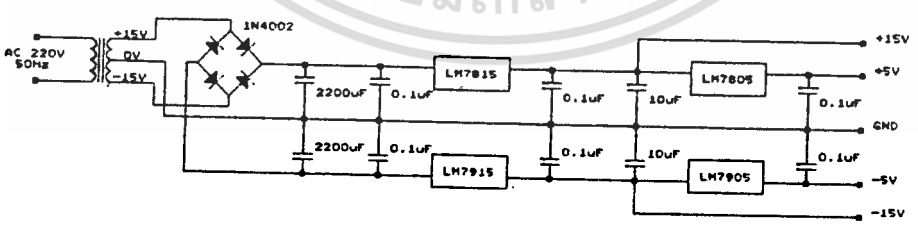
วงจรทำหน้าที่แปลงสัญญาณอินพุทที่เป็นอนาลอกให้เป็นดิจิตอล คือ เมื่อได้รับสัญญาณจาก 8031 เข้ามาที่ขา START ของ ADC ADC ก็จะเริ่มแปลงสัญญาณอินพุทจากอนาลอก เป็นดิจิตอล เมื่อทำการแปลงเรียบร้อยแล้ว ก็จะทำการส่งสัญญาณ EOC (End of Conversion) ไปบอก 8031 ให้ทำการรับข้อมูลที่เป็นดิจิตอลจาก ADC ได้

วงจรหน่วยความจำชั่วคราว (6116)



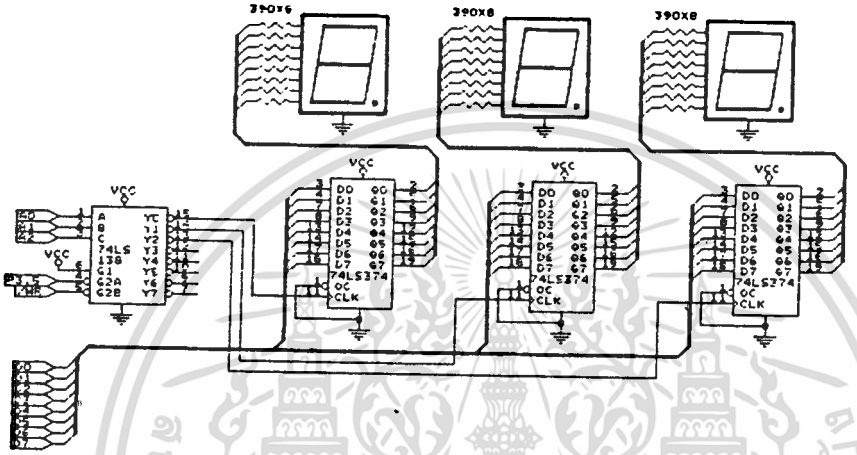
ใช้ 6116 ทำหน้าที่เป็น External Data Memory สำหรับเก็บข้อมูล 128 ไบท์ ที่ได้จาก ADC ในที่นี้ 6116 มีการอ้างแอดแดรสแบบ 8 บิท เพราะสะดวกและรวดเร็วกว่าการอ้างแบบ 16 บิท ดังนั้นจึงต้องมีการเลือก Page Address เพื่อให้อ้างแอดเดรสได้ครบ 2 KBYTE

แหล่งจ่ายไฟ

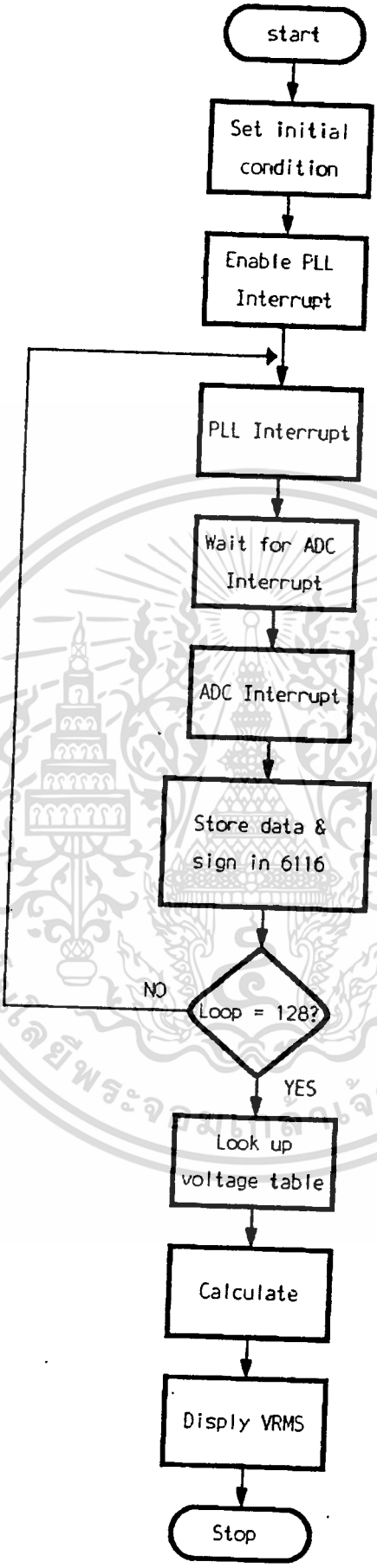


ในที่นี้ใช้ IC เร็กกูเลเตอร์เบอร์ 7805,7905 สำหรับศักดาบวกและลบ 5 โวลต์ และ IC เบอร์ 7815,7915 สำหรับศักดาบวกและลบ 15 โวลต์

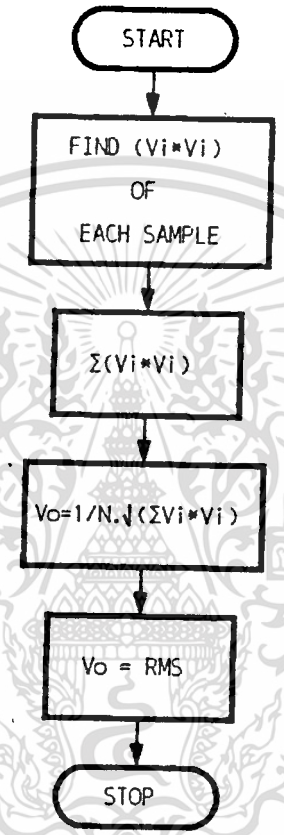
ส่วนแสดงผล (Display)



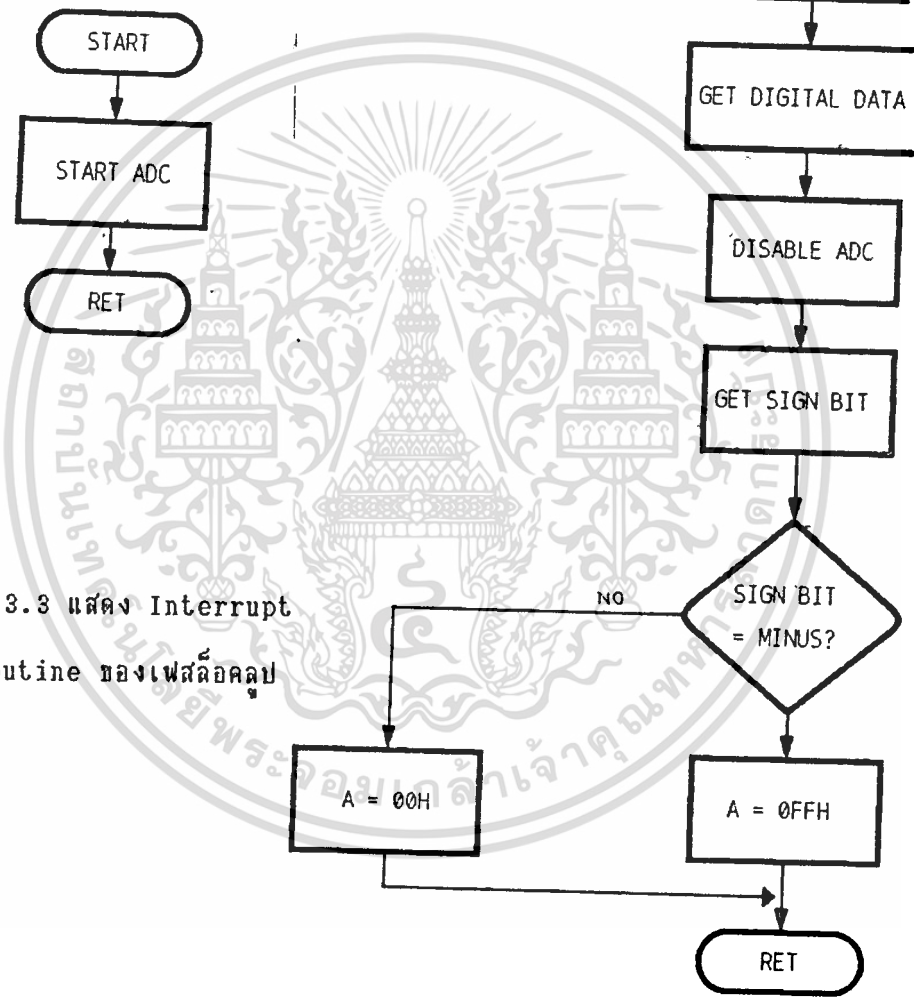
เราใช้ 7 Segment 3 ตัว ในการแสดงผล โดยมี 74LS138 เป็นตัวเลือกแต่ละหลัก ของ 7 Segment 74LS374 เป็นตัวคงค่าข้อมูลผ่านความต้านทาน 330 โอห์ม เข้า 7 Segment แต่ละหลัก



รูป 3.1 แสดงโฟล์วชาร์ตโปรแกรมหลักของการหาค่า RMS



รูปที่ 3.2 แสดงโฟล์วชาร์ตของบล็อค Calculate



รูป 3.3 แสดง Interrupt Routine ของเฟสล็อคลูป

รูป 3.4 แสดง Interrupt Routine ของ ADC

```
0000                    SIGN    .equ    3AH     ;SIGN BIT
0000                    ROB     .equ    38H     ;BUF FOR R0 IN MUL
0000                    DOM     .equ    39H     ;DOMINANCE
0000                    MULT    .equ    3AH     ;MULTIPLIER
0000                    BUFF0   .equ    3BH     ;MULTIPLY BUFFER0
0000                    BUFF1   .equ    3CH     ;MULTIPLY BUFFER1
0000                    RESULT  .equ    3DH     ;REAL RESULT
0000                    U       .equ    44H     ;DEVIDER
0000                    V       .equ    45H     ;BUF FOR DIVISION
0000                    RES     .equ    46H     ;RES OF DIVISION
0000                    X0      .equ    47H     ;X_OLD
0000                    X1      .equ    48H     ;X_NEW
0000                    Y       .equ    49H     ;I/P TO BE SQR ROOT
0000                    ERR     .equ    4AH     ;ERROR
0000                    ERR1    .equ    4BH
0000                    B0      .equ    4CH     ;BUFF_0
0000                    B1      .equ    4DH     ;BUFF_1
0000                    B2      .equ    4EH     ;BUFF_2
0000                    B3      .equ    4FH     ;BUFF_3
0000                    B4      .equ    5FH     ;BUFF_4
0000                    ;*******************************
0000                            .ORG 0000H
0000 02 00 50                   LJMP INIT
0003                            .ORG 0003H
0003 02 00 20                   LJMP ADC
0006                    ;*******************************
0006                    ;*****   TIMER 0 INT     *****
0006                    ;*******************************
000B                            .ORG 000BH
000B 0A                         INC R2
000C BA 3F 02                   CJNE R2,#3FH,OK
000F 7B 99                      MOV R3,#99H
0011 32                 OK      RETI
0012                    ;*******************************
0012                    ;****   PLL (ext1) INT    ******
0012                    ;*******************************
0013                            .ORG 0013H
0013 7A 0F                      MOV R2,#0FH
0015 D2 93                      SETB P1.3       ;START ADC
0017 C2 93                      CLR P1.3
0019 32                         RETI
001A                    ;*******************************
001A                    ;*****   ADC (ext0) INT     ****
001A                    ;*******************************
0020                            .ORG 0020H
0020 C2 97              ADC     CLR P1.7        ;ENABLE ADC
0022 E3                         MOVX A,@R1      ;GET DIGITAL DATA
0023 D2 97                      SETB P1.7       ;DISABLE ADC
0025 FD                         MOV R5,A        ;R5 = DIGITAL DATA
0026 A2 92                      MOV C,P1.2      ;CARRY = SIGN
0028 40 04                      JC PLUS
002A 74 FF                      MOV A,#0FFH     ;IF MINUS, A=#FFH
002C 80 02                      SJMP GO
```

```
002E 74 00        PLUS   MOV A,#00H       ;IF PLUS, A=#00H
0030 7B FF        GO     MOV R3,#0FFH
0032 32                  RETI
0033              ;*****************************
0033              ;***   INITIALIZATION     ****
0033              ;*****************************
0050                     .ORG 0050H
0050 D2 88        INIT   SETB IT0          ;TRANSITION-ACT. EXT INT
0052 D2 8A               SETB IT1
0054 D2 AF               SETB EA           ;ENABLE GLOBAL INT
0056              ;*****************************
0056              ;********     BLINK     *******
0056              ;*****************************
0056 D2 A9               SETB ET0          ;ENABLE TIMER 0
0058 75 89 01            MOV TMOD,#00000001B  ;TIMER MODE 1
005B 7C 0A               MOV R4,#0AH       ;BLINK COUNTER
005D 75 70 FF     BLINK  MOV 70H,#0FFH
0060 75 71 FF            MOV 71H,#0FFH
0063 12 10 00            LCALL TRANS       ;CALL SUB DISP
0066 7A 3C               MOV R2,#3CH
0068 7B 00               MOV R3,#00H
006A D2 8C               SETB TR0          ;START TIMER0
006C BB 99 FD            CJNE R3,#99H,$
006F C2 8C               CLR TR0           ;STOP TIMER
0071 75 70 EE            MOV 70H,#0EEH
0074 75 71 EE            MOV 71H,#0EEH
0077 12 10 00            LCALL TRANS
007A 7A 3C               MOV R2,#3CH
007C 7B 00               MOV R3,#00H
007E D2 8C               SETB TR0
0080 BB 99 FD            CJNE R3,#99H,$
0083 C2 8C               CLR TR0
0085 DC D6               DJNZ R4,BLINK
0087              ;*****************************
0087 D2 A8        WWW    SETB EX0          ;ENABLE ADC INT
0089 53 90 C4            ANL P1,#11000100B  ;SEL ADC IN0 I/P & 6116 PAC
008C 78 00               MOV R0,#00H       ;R0 = 6116 INDEX ADD
008E 7C 80               MOV R4,#128D      ;R4 = SAMPLING LOOP
0090 C2 93               CLR P 1.3
0092              ;*********************************
0092              ;****  GET AND CONVERSE DATA ****
0092              ;*********************************
0092 D2 AA               SETB EX1          ;ENABLE PLL INT (EXT1)
0094 BA 0F FD     SAM    CJNE R2,#0FH,$    ;WAIT FOR PLL INT
0097 7A 00               MOV R2,#00H       ;CLR R2
0099 BB FF FD            CJNE R3,#0FFH,$   ;WAIT FOR ADC INT
009C 7B 00               MOV R3,#00H       ;CLR R3
009E C2 96               CLR P1.6          ;ENABLE 6116
00A0 F2                  MOVX @R0,A        ;MOVE SIGN IN 6116
00A1 08                  INC R0
00A2 ED                  MOV A,R5
00A3 F2                  MOVX @R0,A        ;MOVE DIG DTA IN 6116
00A4 08                  INC R0
```

```
00A5 D2 96              SETB P1.6        ;DISABLE 6116
00A7 DC EB              DJNZ R4,SAM      ;NUM OF SAMPLING = 128
00A9 C2 AA              CLR EX1          ;DISABLE PLL INT
00AB 78 00              MOV R0,#00H
00AD 79 00              MOV R1,#00H
00AF C2 95       LOOPP  CLR P1.5         ;PAGE ADD 000
00B1 C2 96              CLR P1.6
00B3 E2                 MOVX A,@R0       ;MOV SIGN DTA FROM 6116
00B4 D2 96              SETB P1.6
00B6 B4 00 05           CJNE A,#00H,PLUS1 ;SIGN DTA = PLUS?
00B9 90 14 00    MINUS  MOV DPTR,#1400H ;DPTR = MINUS DTA
00BC 80 03              SJMP GETDTA
00BE 90 12 00    PLUS1  MOV DPTR,#1200H ;DPTR = PLUS DTA
00C1 08          GETDTA INC R0
00C2 C2 96              CLR P1.6
00C4 E2                 MOVX A,@R0       ;RETRIEVE DIGITAL DTA
00C5 D2 96              SETB P1.6
00C7 08                 INC R0
00C8 C3                 CLR C
00C9 33                 RLC A            ;MUL A BY 2
00CA FB                 MOV R3,A
00CB 50 02              JNC PPP
00CD 05 83              INC DPH
00CF 93          PPP    MOVC A,@A+DPTR   ;LOOK UP TABLE
00D0 FC                 MOV R4,A
00D1 0B                 INC R3
00D2 EB                 MOV A,R3
00D3 93                 MOVC A,@A+DPTR   ;LOOK UP TABLE
00D4 D2 95              SETB P1.5        ;PAGE ADD 001
00D6 C2 96              CLR P1.6
00D8 F3                 MOVX @R1,A       ;MOV TABLE DTA TO 6116
00D9 09                 INC R1
00DA EC                 MOV A,R4
00DB F3                 MOVX @R1,A       ;MOV TABLE DTA TO 6116
00DC 09                 INC R1
00DD D2 96              SETB P1.6
00DF B8 00 CD           CJNE R0,#256D,LOOPP
00E2             ;*****************************************
00E2 75 3D 40           MOV RESULT,#40H ;RESULT = 40H
00E5 7A 04              MOV R2,#04H
00E7 A8 3D              MOV R0,RESULT
00E9 76 00       CLEAR  MOV @R0,#00H     ;CLR RESULT
00EB 08                 INC R0
00EC DA FB              DJNZ R2,CLEAR
00EE D2 95              SETB P1.5
00F0 78 00               MOV R0,#00H
00F2 C2 96       SAMAG  CLR P1.6
00F4 E2                 MOVX A,@R0       ;GET HI BYTE DTA
00F5 D2 96              SETB P1.6
00F7 08                 INC R0
00F8 F5 31              MOV 31H,A        ;MOV HI BYTE DTA TO 31H
00FA C2 96              CLR P1.6
00FC E2                 MOVX A,@R0       ;GET LO BYTE DTA
```

```
00FD D2 96                SETB P1.6
00FF 08                   INC R0
0100 F5 30                MOV 30H,A       ;MOV LO BYTE DTA TO 30H
0102 E5 31                MOV A,31H       ;CHECK SIGN BIT
0104 30 E7 0F             JNB A.7,NOCOM
0107 C3                    CLR C
0108 E5 30                MOV A,30H
010A F4                   CPL A           ;COMPLEMENT A
010B 24 01                ADD A,#01H      ;TWO COMPLEMENT
010D F5 30                MOV 30H,A       ;MOV BACK TO INT RAM
010F E5 31                MOV A,31H
0111 F4                   CPL A
0112 34 00                ADDC A,#00H
0114 F5 31                MOV 31H,A
0116 85 30 32     NOCOM   MOV 32H,30H     ;DTA IN 30H = 32H
0119 85 31 33             MOV 33H,31H     ;DTA IN 31H = 33H
011C              ;*********************************
011C              ;*      MULTIPLY SUBROUTINE      *
011C              ;*********************************
011C 88 38                MOV ROB,R0      ;KEEP OLD R0 IN ROB
011E 75 3A 32             MOV MULT,#32H
0121 75 3B 34             MOV BUFF0,#34H
0124 75 3C 34             MOV BUFF1,#34H
0127 7A 04                MOV R2,#04H
0129 A8 3B                MOV R0,BUFF0
012B 76 00        CLR     MOV @R0,#00H    ;CLEAR BUFF0
012D 08                   INC R0
012E DA FB                DJNZ R2,CLR
0130 7B 02                MOV R3,#02H     ;R3 = OUTTER LOOP
0132 75 39 30     REPEAT  MOV DOM,#30H
0135 85 3C 3B             MOV BUFF0,BUFF1
0138 7A 02                MOV R2,#02H     ;R2 = INNER LOOP
013A A8 39        AGAIN   MOV R0,DOM
013C E6                   MOV A,@R0       ;MOV DOM IN A
013D A8 3A                MOV R0,MULT
013F 86 F0                MOV B,@R0       ;MOV MULT IN B
0141 A4                   MUL AB          ;MULTIPLY AB
0142 A9 3B                MOV R1,BUFF0
0144 27                   ADD A,@R1
0145 F7                   MOV @R1,A
0146 05 3B                INC BUFF0
0148 A9 3B                MOV R1,BUFF0
014A E5 F0                MOV A,B
014C 37                   ADDC A,@R1
014D F7                   MOV @R1,A
014E 50 02                JNC SEA
0150 09                   INC R1
0151 07                   INC @R1
0152 05 39        SEA     INC DOM
0154 DA E4                DJNZ R2,AGAIN
0156 05 3C                INC BUFF1
0158 05 3A                INC MULT
015A DB D6                DJNZ R3,REPEAT
```

```
015C A8 38                MOV R0,ROB
015E                ;*****************************
015E 88 38          SS    MOV ROB,R0
0160 7B 04                MOV R3,#04H
0162 A8 3D                MOV R0,RESULT   ;R0 = RESULT
0164 75 3B 34             MOV BUFF0,#34H
0167 A9 3B                MOV R1,BUFF0    ;R1 = #34H
0169 C3                   CLR C
016A E7             CC    MOV A,@R1
016B 36                   ADDC A,@R0      ;ADD 34H WITH RESULT
016C F6                   MOV @R0,A
016D 08                   INC R0
016E 09                   INC R1
016F DB F9                DJNZ R3,CC
0171 A8 38                MOV R0,ROB
0173                ;*****************************
0173 B8 00 02             CJNE R0,#00H,SEANG     ;128 LOOP?
0176                ;*****************************
0176 80 03                SJMP ROTA
0178 02 00 F2       SEANG LJMP SAMAG
017B 7B 04          ROTA  MOV R3,#04H
017D 78 40                MOV R0,#40H     ;4 BYTE ROTETE
017F C3                   CLR C
0180 E6             ROTATE1       MOV A,@R0       ;A = RESULT
0181 33                   RLC A           ;DEVIDE BY 128
0182 F6                   MOV @R0,A
0183 08                   INC R0
0184 DB FA                DJNZ R3,ROTATE1
0186                ;*****************************
0186 75 3F 00             MOV 3FH,#00H    ;CLR 3FH
0189 75 40 00             MOV 40H,#00H    ;CLR 40H
018C 75 49 3F             MOV Y,#3FH
018F 75 47 50             MOV X0,#50H
0192 75 48 55             MOV X1,#55H
0195 75 4A 5E             MOV ERR,#5EH
0198 75 4B 5A             MOV ERR1,#5AH
019B A8 47                MOV R0,X0
019D 76 00                MOV @R0,#00H
019F 08                   INC R0
01A0 76 01                MOV @R0,#01H    ;X0 = 1
01A2 7C 03                MOV R4,#03H
01A4 08             CLR1  INC R0
01A5 76 00                MOV @R0,#00H
01A7 DC FB                DJNZ R4,CLR1
01A9 A8 49          AGAIN1 MOV R0,Y
01AB 79 65                MOV R1,#65H     ;MOV Y TO 65H
01AD 12 02 65             LCALL LOAD
01B0 A8 47                MOV R0,X0
01B2 79 60                MOV R1,#60H     ;MOV X0 TO 60H
01B4 12 02 65             LCALL LOAD
01B7 12 08 00             LCALL DIV       ;DEVIDE Y BY X0
01BA 78 76                MOV R0,#76H
01BC A9 48                MOV R1,X1       ;MOV RESULT TO X1
```

```
01BE 12 02 65              LCALL LOAD
01C1 A8 47                 MOV R0,X0
01C3 A9 48                 MOV R1,X1
01C5 C3                    CLR C
01C6 7C 05                 MOV R4,#05H
01C8 E6            SUM     MOV A,@R0
01C9 37                    ADDC A,@R1      ;ADD X0 WITH X1
01CA F7                    MOV @R1,A
01CB 08                    INC R0
01CC 09                    INC R1
01CD DC F9                 DJNZ R4,SUM
01CF C3                    CLR C
01D0 7C 05                 MOV R4,#05H
01D2 19            ROT     DEC R1
01D3 E7                    MOV A,@R1
01D4 13                    RRC A           ;DEVIDE X1 BY 2
01D5 F7                    MOV @R1,A
01D6 DC FA                 DJNZ R4,ROT
01D8 78 53                 MOV R0,#53H
01DA 79 54                 MOV R1,#54H
01DC 7C 04                 MOV R4,#04H
01DE E6            SHIFT   MOV A,@R0       ;SHIFT X0 TO THE -
01DF F7                    MOV @R1,A       ;LEFT 8 BIT
01E0 18                    DEC R0
01E1 19                    DEC R1
01E2 DC FA                 DJNZ R4,SHIFT
01E4 77 00                 MOV @R1,#00H
01E6 A8 47                  MOV R0,X0
01E8 79 65                 MOV R1,#65H     ;MOV X0 TO 65H
01EA 12 02 65              LCALL LOAD
01ED A8 48                 MOV R0,X1
01EF 79 60                 MOV R1,#60H     ;MOV X1 TO 60H
01F1 12 02 65              LCALL LOAD
01F4 12 08 00              LCALL DIV       ;DEVIDE X0 BY X1
01F7 78 76                 MOV R0,#76H
01F9 A9 4B                 MOV R1,ERR1     ;MOV 76H TO ERR
01FB 12 02 65              LCALL LOAD
01FE 7C 04                 MOV R4,#04H
0200 A8 4A                 MOV R0,ERR
0202 B6 00 0A      CHK     CJNE @R0,#00H,MORE      ;@R0 = 00H?
0205 18                    DEC R0
0206 DC FA                 DJNZ R4,CHK
0208 B6 FD 02              CJNE @R0,#0FDH,CHK1     ;@R0 = FDH?
020B 80 1A                 SJMP LOOP
020D 40 18         CHK1            JC LOOP         ;IF @R0<FDH, JMP TO LC
020F 7C 03         MORE    MOV R4,#03H
0211 A8 4A                 MOV R0,ERR
0213 B6 00 11      CHK2    CJNE @R0,#00H,LOOP
0216 18                    DEC R0
0217 DC FA                 DJNZ R4,CHK2
0219 B6 00 02              CJNE @R0,#00H,CHK3
021C 80 12                 SJMP RESULT1            ;ERR IS ACCEPT
021E B6 01 06      CHK3    CJNE @R0,#01H,LOOP
```

```
0221 18                  DEC R0
0222 B6 02 00            CJNE @R0,#02H,CHK4
0225 40 09        CHK4   JC RESULT1      ;IF @R0<02H, JMP TO RESULT
0227 A8 48        LOOP   MOV R0,X1
0229 A9 47               MOV R1,X0
022B 12 02 65            LCALL LOAD
022E 21 A9               AJMP AGAIN1              ;ERR TOO MUCH
0230 85 57 7C     RESULT1        MOV 7CH,57H              ;LO BYTE VRM
0233 85 56 7D            MOV 7DH,56H              ;HI BYTE VRMS
0236 74 00        BIN_BCD  MOV A,#00H
0238 C3                    CLR C
0239 79 7B                 MOV R1,#7BH
023B 77 00                 MOV @R1,#00H             ;R1 = BCD DTA
023D 19                    DEC R1
023E 77 00                 MOV @R1,#00H
0240 7A 10                 MOV R2,#10H
0242 78 7D        BCD1     MOV R0,#7DH              ;R0 = BIN DTA
0244 7B 02                 MOV R3,#02H
0246 E6           BCD2     MOV A,@R0
0247 33                    RLC A
0248 F6                    MOV @R0,A
0249 18                    DEC R0
024A DB FA                 DJNZ R3,BCD2
024C 78 7B                 MOV R0,#7BH
024E 7B 02                 MOV R3,#02H
0250 E6           BCD3     MOV A,@R0
0251 36                    ADDC A,@R0
0252 D4                    DA A                     ;DECIMAL ADJUST ACC
0253 F6                    MOV @R0,A
0254 18                    DEC R0
0255 DB F9                 DJNZ R3,BCD3
0257 DA E9                 DJNZ R2,BCD1
0259 85 7B 70              MOV 70H,7BH      ;LO BYTE BCD VRMS
025C 85 7A 71              MOV 71H,7AH      ;HI BYTE BCD VRMS
025F 12 10 00              LCALL TRANS
0262 02 00 87              LJMP WWW
0265                ;*****************************
0265 7C 05          LOAD   MOV R4,#05H
0267 E6             LOAD1  MOV A,@R0
0268 F7                    MOV @R1,A
0269 08                    INC R0
026A 09                    INC R1
026B DC FA                 DJNZ R4,LOAD1
026D 22                    RET
026E                ;*****************************
0800                       .ORG 0800H       ;DEVIDE SUBROUTINE
0800 88 4C          DIV    MOV B0,R0
0802 89 4D                 MOV B1,R1
0804 8A 4E                 MOV B2,R2
0806 8B 4F                 MOV B3,R3
0808 8C 5F                 MOV B4,R4
080A 7B 05                 MOV R3,#05H
080C 78 69                 MOV R0,#69H
```

```
080E 75 45 6A              MOV V,#6AH
0811 75 44 60              MOV U,#60H
0814 75 46 7A              MOV RES,#7AH
0817 A9 45                 MOV R1,V
0819 7C 05                 MOV R4,#05H
081B 77 00        CLR2     MOV @R1,#00H    ;CLEAR V
081D 09                    INC R1
081E DC FB                 DJNZ R4,CLR2
0820 7A 08        BYTE     MOV R2,#08H
0822 C3           BIT      CLR C
0823 E6                    MOV A,@R0
0824 33                    RLC A
0825 F6                    MOV @R0,A
0826 A9 45                 MOV R1,V
0828 7C 05                 MOV R4,#05H
082A E7           ROTATE   MOV A,@R1
082B 33                    RLC A
082C F7                    MOV @R1,A
082D 09                    INC R1
082E DC FA                 DJNZ R4,ROTATE
0830 C3                    CLR C
0831 7C 05                 MOV R4,#05H
0833 A9 45        SUBTRACT MOV R1,V
0835 E7                    MOV A,@R1
0836 A9 44                 MOV R1,U
0838 97                    SUBB A,@R1
0839 A9 45                 MOV R1,V
083B F7                    MOV @R1,A
083C 05 44                 INC U
083E 05 45                 INC V
0840 DC F1                 DJNZ R4,SUBTRACT
0842 40 1E                 JC BORROW
0844 A9 46                 MOV R1,RES
0846 E7                    MOV A,@R1
0847 B3                    CPL C
0848 33                    RLC A
0849 F7                    MOV @R1,A
084A 75 45 6A     CHECK    MOV V,#6AH
084D 75 44 60              MOV U,#60H
0850 DA D0                 DJNZ R2,BIT
0852 18                    DEC R0
0853 15 46                 DEC RES
0855 DB C9                 DJNZ R3,BYTE
0857 A8 4C                 MOV R0,B0
0859 A9 4D                 MOV R1,B1
085B AA 4E                 MOV R2,B2
085D AB 4F                 MOV R3,B3
085F AC 5F                 MOV R4,B4
0861 22                    RET
0862 A9 46        BORROW   MOV R1,RES
0864 E7                    MOV A,@R1
0865 B3                    CPL C
0866 33                    RLC A
```

```
0867 F7                     MOV @R1,A
0868 C3                     CLR C
0869 75 45 6A               MOV V,#6AH
086C 75 44 60               MOV U,#60H
086F C3                     CLR C
0870 7C 05                  MOV R4,#05H
0872 A9 45          ADD     MOV R1,V
0874 E7                     MOV A,@R1
0875 A9 44                  MOV R1,U
0877 37                     ADDC A,@R1
0878 A9 45                  MOV R1,V
087A F7                     MOV @R1,A
087B 05 44                  INC U
087D 05 45                  INC V
087F DC F1                  DJNZ R4,ADD
0881 80 C7                  SJMP CHECK
0883                ;*****************************
0883                ;***** SUBRUTINE  DISPLY *****
0883                ;*****************************
1000                        .ORG 1000H      ;TRNSFR 3 NIB TO 3 BYTE
1000 88 4C          TRANS   MOV B0,R0
1002 89 4D                  MOV B1,R1
1004 8A 4E                  MOV B2,R2
1006 C2 B5                  CLR P3.5        ;ENABLE DISP
1008 79 70                  MOV R1,#70H
100A E7                     MOV A,@R1
100B 54 0F                  ANL A,#0FH
100D F5 72                  MOV 72H,A
100F E7                     MOV A,@R1
1010 C4                     SWAP A
1011 54 0F                  ANL A,#0FH
1013 F5 73                  MOV 73H,A
1015 09                     INC R1
1016 E7                     MOV A,@R1
1017 54 0F                  ANL A,#0FH
1019 F5 74                  MOV 74H,A
101B                ;*********************************
101B                ;**** LOOK UP TABLE & DISPLAY ****
101B                ;*********************************
101B 7A 03                  MOV R2,#03H     ;R2 = LOOP COUNTER
101D 79 72                  MOV R1,#72H     ;R1 = DATA ADD
101F 90 11 00               MOV DPTR,#1100H ;DPTR AT TOP TABLE
1022 78 00                  MOV R0,#00H     ;R0 ENABLE 7 SEG DIGIT
1024 E7             LOOPD   MOV A,@R1       ;MOV DATA TO A
1025 93                     MOVC A,@A+DPTR  ;LOOK UP TABLE
1026 F2                     MOVX @R0,A      ;RESULT TO 7 SEG
1027 08                     INC R0          ;NEXT DIGIT
1028 09                     INC R1          ;NEXT DATA
1029 DA F9                  DJNZ R2,LOOPD
102B D2 B5                  SETB P3.5       ;DISABLE DISP
102D A8 4C                  MOV R0,B0
102F A9 4D                  MOV R1,B1
1031 AA 4E                  MOV R2,B2
```

```
1033 22                 RET
1034                ;*********************************
1034                ;***** TABLE FOR 7 SEGMENT ******
1034                ;*********************************
1100                    .ORG 1100H
1100 C0             TABLE   .BYTE 0C0H
1101 F9                     .BYTE 0F9H
1102 A4                     .BYTE 0A4H
1103 B0                     .BYTE 0B0H
1104 99                     .BYTE 99H
1105 92                     .BYTE 92H
1106 82                     .BYTE 82H
1107 F8                     .BYTE 0F8H
1108 80                     .BYTE 80H
1109 90                     .BYTE 90H
110A 88                     .BYTE 88H
110B 83                     .BYTE 83H
110C C6                     .BYTE 0C6H
110D A1                     .BYTE 0A1H
110E 00                     .BYTE 00H
110F FF                     .BYTE 0FFH
1110                        .END
```

ROT	01D2	SHIFT	01DE	CHK	0202
CHK1	020D	MORE	020F	CHK2	0213
CHK3	021E	CHK4	0225	LOOP	0227
RESULT1	0230	BIN_BCD	0236	BCD1	0242
BCD2	0246	BCD3	0250	LOAD	0265
LOAD1	0267	DIV	0800	CLR2	081B
BYTE	0820	BIT	0822	ROTATE	082A
SUBTRACT	0833	CHECK	084A	BORROW	0862
ADD	0872	TRANS	1000	LOOPD	1024
TABLE	1100				

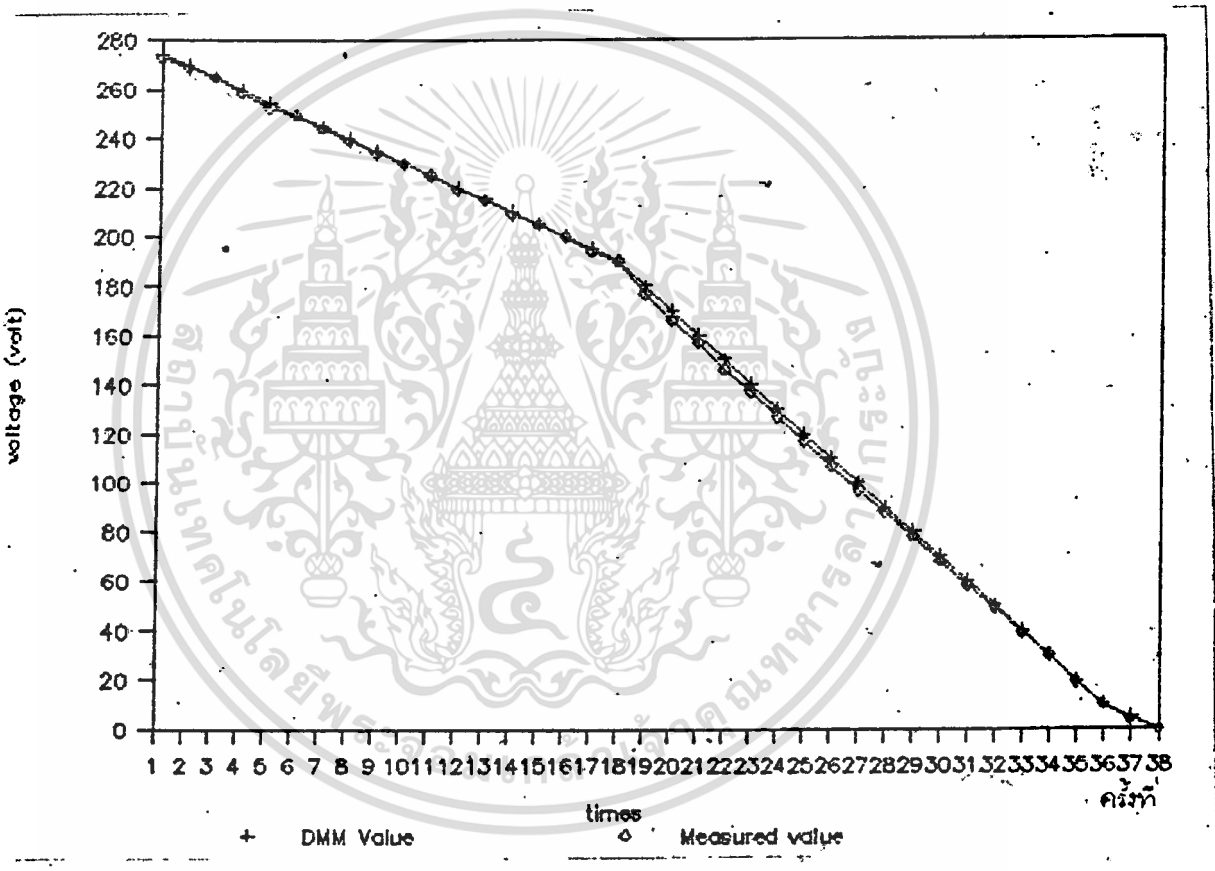
ADDR	00	01	02	03	04	05	06	07	08	09	0A	0B	0C	0D	0E	0F
0000	02	00	50	02	00	20	00	00	00	00	00	0A	BA	3F	02	7B
0010	99	32	00	7A	0F	D2	93	C2	93	32	00	00	00	00	00	00
0020	C2	97	E3	D2	97	FD	A2	92	40	04	74	FF	80	02	74	00
0030	7B	FF	32	00	00	00	00	00	00	00	00	00	00	00	00	00
0040	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
0050	D2	88	D2	8A	D2	AF	D2	A9	75	89	01	7C	0A	75	70	FF
0060	75	71	FF	12	10	00	7A	3C	7B	00	D2	8C	BB	99	FD	C2
0070	8C	75	70	EE	75	71	EE	12	10	00	7A	3C	7B	00	D2	8C
0080	BB	99	FD	C2	8C	DC	D6	D2	A8	53	90	C4	78	00	7C	80
0090	C2	93	D2	AA	BA	0F	FD	7A	00	BB	FF	FD	7B	00	C2	96
00A0	F2	08	ED	F2	08	D2	96	DC	EB	C2	AA	78	00	79	00	C2
00B0	95	C2	96	E2	D2	96	B4	00	05	90	14	00	80	03	90	12
00C0	00	08	C2	96	E2	D2	96	08	C3	33	FB	50	02	05	83	93
00D0	FC	0B	EB	93	D2	95	C2	96	F3	09	EC	F3	09	D2	96	B8
00E0	00	CD	75	3D	40	7A	04	A8	3D	76	00	08	DA	FB	D2	95

```
00F0   78 00 C2 96 E2 D2 96 08 F5 31 C2 96 E2 D2 96 08
0100   F5 30 E5 31 30 E7 0F C3 E5 30 F4 24 01 F5 30 E5
0110   31 F4 34 00 F5 31 85 30 32 85 31 33 88 38 75 3A
0120   32 75 3B 34 75 3C 34 7A 04 A8 3B 76 00 08 DA FB
0130   7B 02 75 39 30 85 3C 3B 7A 02 A8 39 E6 A8 3A 86
0140   F0 A4 A9 3B 27 F7 05 3B A9 3B E5 F0 37 F7 50 02
0150   09 07 05 39 DA E4 05 3C 05 3A DB D6 A8 38 88 38
0160   7B 04 A8 3D 75 3B 34 A9 3B C3 E7 36 F6 08 09 DB
0170   F9 A8 38 B8 00 02 80 03 02 00 F2 7B 04 78 40 C3
0180   E6 33 F6 08 DB FA 75 3F 00 75 40 00 75 49 3F 75
0190   47 50 75 48 55 75 4A 5E 75 4B 5A A8 47 76 00 08
01A0   76 01 7C 03 08 76 00 DC FB A8 49 79 65 12 02 65
01B0   A8 47 79 60 12 02 65 12 08 00 78 76 A9 48 12 02
01C0   65 A8 47 A9 48 C3 7C 05 E6 37 F7 08 09 DC F9 C3
01D0   7C 05 19 E7 13 F7 DC FA 78 53 79 54 7C 04 E6 F7
01E0   18 19 DC FA 77 00 A8 47 79 65 12 02 65 A8 48 79
01F0   60 12 02 65 12 08 00 78 76 A9 4B 12 02 65 7C 04
0200   A8 4A B6 00 0A 18 DC FA B6 FD 02 80 1A 40 18 7C
0210   03 A8 4A B6 0Q 11 18 DC FA B6 00 02 80 12 B6 01
0220   06 18 B6 02 00 40 09 A8 48 A9 47 12 02 65 21 A9
0230   85 57 7C 85 56 7D 74 00 C3 79 7B 77 00 19 77 00
0240   7A 10 78 7D 7B 02 E6 33 F6 18 DB FA 78 7B 7B 02
0250   E6 36 D4 F6 18 DB F9 DA E9 85 7B 70 85 7A 71 12
0260   10 00 02 00 87 7C 05 E6 F7 08 09 DC FA 22 00 00
0270   00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00

0800   88 4C 89 4D 8A 4E 8B 4F 8C 5F 7B 05 78 69 75 45
0810   6A 75 44 60 75 46 7A A9 45 7C 05 77 00 09 DC FB
0820   7A 08 C3 E6 33 F6 A9 45 7C 05 E7 33 F7 09 DC FA
0830   C3 7C 05 A9 45 E7 A9 44 97 A9 45 F7 05 44 05 45
0840   DC F1 40 1E A9 46 E7 B3 33 F7 75 45 6A 75 44 60
0850   DA D0 18 15 46 DB C9 A8 4C A9 4D AA 4E AB 4F AC
0860   5F 22 A9 46 E7 B3 33 F7 C3 75 45 6A 75 44 60 C3
0870   7C 05 A9 45 E7 A9 44 37 A9 45 F7 05 44 05 45 DC
0880   F1 80 C7 00 00 00 00 00 00 00 00 00 00 00 00 00
0890   00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00

1000   88 4C 89 4D 8A 4E C2 B5 79 70 E7 54 0F F5 72 E7
1010   C4 54 0F F5 73 09 E7 54 0F F5 74 7A 03 79 72 90
1020   11 00 78 00 E7 93 F2 08 09 DA F9 D2 B5 A8 4C A9
1030   4D AA 4E 22 00 00 00 00 00 00 00 00 00 00 00 00
1040   00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00

1100   C0 F9 A4 B0 99 92 82 F8 80 90 88 83 C6 A1 00 FF

tasm: Number of errors = 0
11:15am Feb 26,1991
```

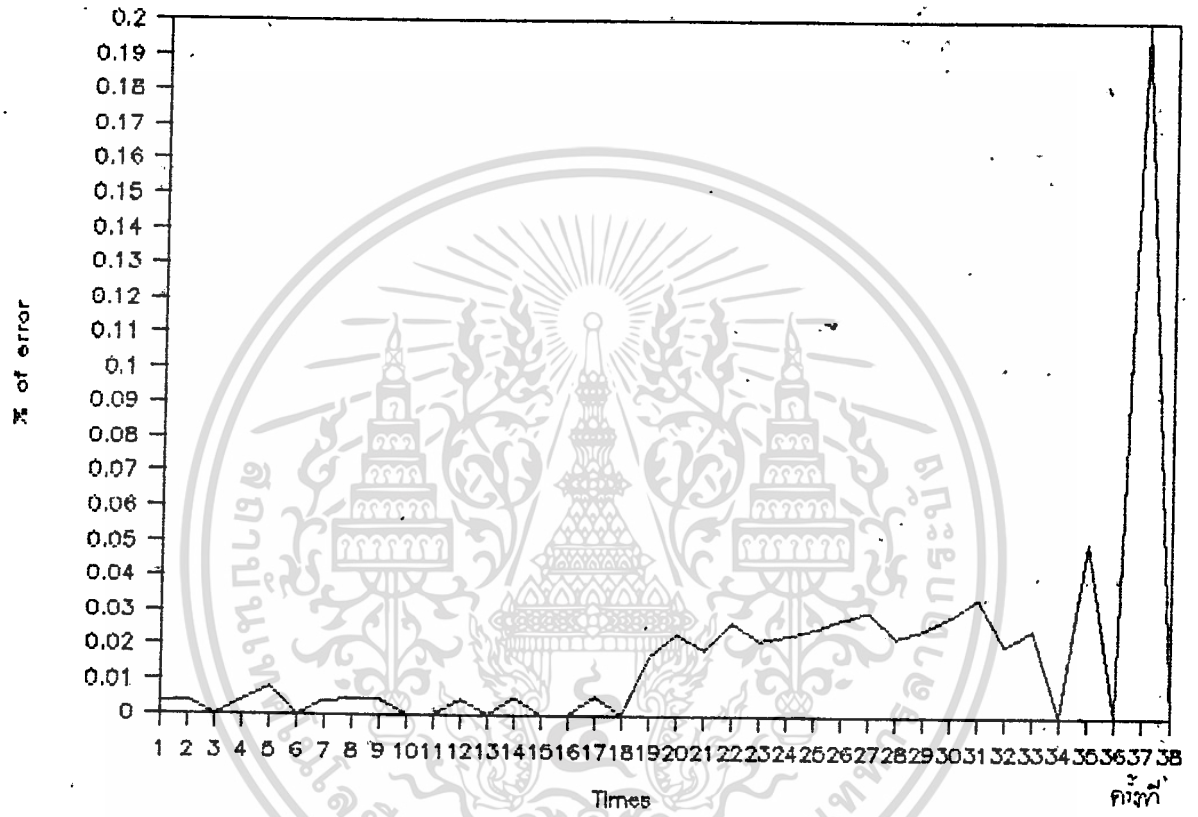
บทที่ 4

ผลการทดลอง

เมื่อทำการทดลอง วัดค่า RMS จากเครื่องวัดที่สร้างขึ้น ตั้งแต่ 0 - 275 โวลต์ พบว่า ค่าที่วัดได้ใกล้เคียงกับค่าที่วัดจาก ดิจิตอล มัลติมิเตอร์ มาก มีค่าผิดพลาดมากที่สุดเพียง 0.2 % เท่านั้น ซึ่งแสดง ได้ดังกราฟ



รูป 4.1 แสดงค่าที่ได้จากการวัด กับ ค่าจาก DMM



รูป 4.2 แสดงค่าผิดพลาดเป็นเปอร์เซนต์ของการวัด

บทที่ 5

สรุปและวิจารณ์

จากการทดลอง และการทดสอบสมมุติฐานที่ได้ตั้งไว้ในบทที่ 2 พบว่า เป็นไปตามที่คาดหวังไว้ คือเมื่อทำการวัดค่า RMS โดยใช้วงจรที่สร้างขึ้นตามแนวทางบทที่ 3 ปรากฏว่า มีค่าคลาดเคลื่อนจากที่ควรจะเป็น เพียง 0.2 เปอร์เซ็นต์ ซึ่งเป็นค่าที่สามารถยอมรับได้

จากการทดลอง เมื่อศักดาอินพุทลดลง พบว่า วงจรที่สร้างขึ้นจะแสดงค่า RMS ได้ช้าลงกว่าปกติ

กิตติกรรมประกาศ

โครงการนี้สำเร็จลงด้วยดี โดยต้องขอขอบคุณ อาจารย์ ธำรงศักดิ์ สุกใส และอาจารย์ วันชัย ริ้วรุจา ที่คอยให้คำปรึกษา ตลอดจนให้ความช่วยเหลือทางด้านต่างๆ และ ขอขอบใจ เพื่อนๆ ที่คอยให้กำลังใจในด้านต่างๆ จนปริญญานิพนธ์ฉบับนี้ สำเร็จลงได้ด้วยดี

จินตนา ควรทรงธรรม

2534

หนังสืออ้างอิง

1. M.A.H. Abdul-Karim, "Microprocessor - based implicit RMS meter" INT.J.ELECTRONICS,1987,vol 62,no.6,953-959
2. INTEL CORPORATION (UK) Ltd, "Microcontroller Handbook" 1985
3. Howard M. Berlin, "Design of Phase - Locked Loop Circuit, with Experiment"