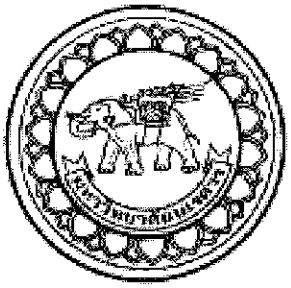


เครื่องวัดความชื้นในดิน

SOIL MOISTURE METER

นายจักรนะรินทร์ พรมชัย รหัส 43362417

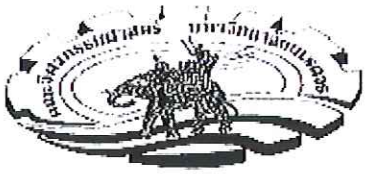
นายรังสฤษฎ์ สุทธิคุณ รหัส 43362631

ปริญญานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาหลักสูตรปริญญาวิศวกรรมศาสตรบัณฑิต

สาขาวิชาวิศวกรรมไฟฟ้า ภาควิชาวิศวกรรมไฟฟ้าและคอมพิวเตอร์

คณะวิศวกรรมศาสตร์ มหาวิทยาลัยนเรศวร

ปีการศึกษา 2548



ใบรับรองโครงงานวิศวกรรม

หัวข้อโครงงาน	เครื่องวัดความชื้นในดิน			
ผู้ดำเนินโครงงาน	นายจักรนะรินทร์	พรมชัย	รหัส	43362417
	นายรังสฤษฎ์	สุทธิคุณ	รหัส	43362631
อาจารย์ที่ปรึกษา	ผศ.ดร.ยงยุทธ	ชนบดีเฉลิมรุ่ง		
อาจารย์ที่ปรึกษาร่วม	ผศ.สมบัติ	ชื่นชูกลิ่น		
สาขาวิชา	วิศวกรรมไฟฟ้า			
ภาควิชา	วิศวกรรมไฟฟ้าและคอมพิวเตอร์			
ปีการศึกษา	2548			

.....

คณะวิศวกรรมศาสตร์ มหาวิทยาลัยนเรศวร อนุมัติให้โครงงานฉบับนี้เป็น**ส่วนหนึ่งของ**การศึกษาตามหลักสูตรวิศวกรรมศาสตรบัณฑิต สาขาวิชาวิศวกรรมไฟฟ้า

คณะกรรมการสอบโครงงานวิศวกรรม

..... ประธานกรรมการ

ผศ. ดร.ยงยุทธ ชนบดีเฉลิมรุ่ง

..... กรรมการ

ดร.สุรเชษฐ์ กานต์ประชา

..... กรรมการ

อาจารย์พนัส นัถฤทธิ์

หัวข้อโครงงาน	เครื่องวัดความชื้นในดิน			
ผู้ดำเนินโครงงาน	นายจักรนะรินทร์	พรมชัย	รหัส	43362417
	นายรังสฤษฎ์	สุทธิคุณ	รหัส	43362631
อาจารย์ที่ปรึกษา	ผศ.ดร.ยงยุทธ	ชนบดีเฉลิมรุ่ง		
อาจารย์ที่ปรึกษาร่วม	ผศ.สมบัติ	ชื่นชูกลิ่น		
สาขาวิชา	วิศวกรรมไฟฟ้า			
ภาควิชา	วิศวกรรมไฟฟ้าและคอมพิวเตอร์			
ปีการศึกษา	2548			

.....

บทคัดย่อ

ในปัจจุบันมีการศึกษาและวิจัยเกี่ยวกับความชื้นในดินกันมากขึ้น เนื่องมาจากการเปลี่ยนแปลงไปของสภาวะโลกที่เป็นอยู่ในปัจจุบัน ซึ่งไม่แน่นอนเมื่อความชื้นในดินเปลี่ยนทำให้เกิดผลกระทบต่อระบบการผลิตที่ต้องรักษาความชื้นในดินเอาไว้ทั้งในทางเกษตรกรรมและด้านอุตสาหกรรม โครงงานนี้จึงนำเสนอการสร้างเครื่องมือวัดความชื้นในดินโดยอาศัยหลักการของ CAPACITANCE มาประยุกต์สร้างเครื่องมือวัดความชื้นในดินที่มีราคาถูกและมีประสิทธิภาพใกล้เคียงกับเครื่องมือวัดความชื้นในดินที่มีจำหน่ายตามท้องตลาดที่มีราคาแพงมาใช้งานได้ ซึ่งหวังว่าโครงงานนี้จะสามารถทำประโยชน์แก่ผู้นำไปใช้ หรือศึกษาวิจัยปรับปรุงได้ในภายภาคหน้า

Project Title	**Soil Moisture Meter**	
Name	Mr. Jaknarin Promchai	ID. 43362417
	Mr. Rangsarit Sootticoon	ID. 43362631
Project Advisor	Assist. Prof. Dr. Yongyut Chonbodeechalermroong	
Co - project advisor	Assist. Prof. Sombat Shunchuklin	
Major	Electrical Engineering	
Department	Electrical and Computer Engineering	
Academic Year	2005	

.....

ABSTRACT

At present, the study and research in soil moisture has increased in a wide range of interest such as, in agricultural and industrial areas.

This project therefore proposes a soil moisture meter by means of capacitance measuring. As a result, we obtain a cheap and reasonably accurate instrument compared with the commercial one.

กิตติกรรมประกาศ

ผู้จัดทำขอขอบพระคุณเป็นอย่างสูง สำหรับอาจารย์ทุกท่านกับการถ่ายทอดความรู้ในการทำโครงงาน โดยเฉพาะอย่างยิ่ง ผศ.ดร.ยงยุทธ ชนบดีเฉลิมรุ่ง ที่ได้ให้คำปรึกษาและคำแนะนำตรวจแก้ไขข้อบกพร่องต่าง ๆ ด้วยความเอาใจใส่ตลอด ทำให้โครงงานจนสำเร็จลุล่วงไปด้วยดี

ขอขอบคุณการไฟฟ้าฝ่ายผลิตแห่งประเทศไทย ฝ่ายปฏิบัติการภาคตะวันออกเฉียงเหนือ ที่เอื้อเฟื้อในการใช้ห้องสอบเทียบ

ท้ายสุดนี้ ผู้จัดทำขอกราบขอบพระคุณ บิดามารดา ที่ให้กำลังใจตลอดมา ตลอดจนเพื่อน ๆ พี่ ๆ น้อง ๆ ทุกคนที่อยู่เบื้องหลังในความสำเร็จของโครงงานเล่มนี้

นายจักรนะรินทร์ พรมชัย

นายรังสฤษฎ์ สุทธิคุณ

สารบัญ

หน้า

สารบัญตาราง

ตารางที่ หน้า

สารบัญรูป

รูปที่ หน้า

บทที่ 1
บทนำ

1.1 หลักการและเหตุผล

ในปัจจุบันมีการให้ความสำคัญกับเรื่องความชื้นในดินกันมากขึ้น ทั้งในทางของการเกษตร ซึ่งการเกษตรแต่ละชนิดแต่ละประเภทต่าง มีความต้องการความชื้นที่แตกต่างกันตามแต่ละชนิดของพืชพรรณ ในด้านอุทกศาสตร์ความชื้นในดินก็สามมารถทำนายการเกิดอุทกภัยได้จากการวัดความชื้นในดินโดยดูได้จากดินสามารถรองรับปริมาณน้ำได้มากน้อยเพียงใด ด้านธรณีสิ่งแวดล้อมนั้นความชื้นในดินสามารถบอกได้ว่าธรณีและสิ่งแวดล้อมเหล่านั้นมีลักษณะเช่นไร

อุปกรณ์วัดความชื้นในดินที่วางขายในปัจจุบันยังมีราคาแพงอยู่มาก ทั้งยังค่าบำรุงรักษาที่แพง ซึ่งหากเป็นการใช้เพื่อการเกษตรทั่วไปเงินที่ลงทุนไปย่อมไม่คุ้มค่าอย่างแน่นอน ดังนั้นผู้จัดทำโครงงานจึงคิดว่าน่าจะมีเครื่องมือที่จะทำให้ทราบค่าความชื้นในดิน ซึ่งมีราคาไม่แพงนักแต่มีความสามารถใกล้เคียงกับเครื่องมือที่มีขายทั่วๆไป ซึ่งจะทำให้ประหยัดเงินได้มากขึ้น และด้วยประการนี้เองจึงเกิดโครงงานเครื่องวัดความชื้นในดิน (soil moisture meter)ขึ้น

1.2 วัตถุประสงค์

1. เพื่อศึกษาเรื่องความชื้นในดิน
2. เพื่อศึกษาการวัดค่าของตัวเก็บประจุ
3. สามารถนำความรู้เรื่องความชื้นในดิน และการวัดค่าตัวเก็บประจุมาประยุกต์สร้างเครื่องวัดความชื้นในดินได้

1.3 ขอบข่ายของงาน

1. ศึกษาเรื่องความชื้นในดิน
2. ศึกษาการวัดค่าตัวเก็บปะจุ
3. สร้างและทดลองการทำงานของเครื่องวัดความชื้น
4. วิเคราะห์และสรุปคุณสมบัติของเครื่อง

1.4 ขั้นตอนการดำเนินงาน

1. ศึกษาเรื่องลักษณะทั่วไปของดิน
2. ศึกษาเรื่องความชื้นในดิน
3. ศึกษาเรื่องหลักการ Capacitance
4. สร้างเครื่องวัดความชื้นในดิน
5. ทดลองเครื่องมือวัดความชื้นในดิน
6. วิเคราะห์และสรุปผลการการทดลอง

ตารางที่ 1.1 แสดงขั้นตอนการดำเนินงานของโครงงาน

กิจกรรม	พ ฤ. ย	ธ. ค	ม. ค	ก. พ	มี. ค	เม .ย	พ. ค	มิ. ย	ก. ค	ส. ค	ก. ย	ต. ค
1. ศึกษาเรื่องลักษณะ ทั่วไป ของ ดิน	←	—	—	→								
2. ศึกษาเรื่องความชื้นในดิน		←	—	—	→							
3. ศึกษาเรื่องหลักการ Capacitance			←	—	—	→						
4. สร้างเครื่องวัดความชื้นในดิน					←	—	—	—	→			
5. ทดลองเครื่องมือวัดความชื้นใน ดิน										←	→	
6. วิเคราะห์และสรุปผลการการทดลอง											←	→

1.5 ผลที่คาดว่าจะได้รับ

1. มีความรู้ความเข้าใจเกี่ยวกับความชื้นในดินและหลักการ Capacitance
2. สามารถสร้างเครื่องมือวัดความชื้นในดินได้
3. เครื่องมือที่สร้างขึ้นสามารถใช้งานได้จริง

1.6 งบประมาณ

อุปกรณ์สร้างเครื่องวัดความชื้นในดิน	1,500 บาท
เอกสาร	500 บาท
รวม	2,000 บาท

บทที่ 2
หลักการและทฤษฎี

2.1 ความสัมพันธ์ระหว่างดินและน้ำ [4]

เม็ดดินที่เรียงตัวกันอยู่มีช่องว่างระหว่างเม็ดดิน น้ำจะแทรกเข้าไปอยู่ในช่องว่างและเกาะติดกับเม็ดดินในลักษณะต่างๆกันด้วยแรงสองชนิด คือ adhesive force และ cohesive force น้ำที่อยู่ในช่องว่างนั้นทั้งหมดจะเป็นปริมาตรสูงสุดที่ดินจะกักเก็บเอาไว้ได้ หากไม่มีแรงภายนอกมากระทำแต่เนื่องจากมีแรงดึงดูดของโลกมากระทำอยู่ตลอดเวลา ซึ่งเป็นแรงที่มากกว่าแรง adhesive และ cohesive จึงทำให้น้ำไหลลงสู่ที่ต่ำกว่า น้ำที่ไหลด้วยสาเหตุดังกล่าวนี้ก็คือ gravity water หรือ free water ส่วนน้ำในช่องว่างเล็กๆที่ไม่สามารถไหลด้วยแรงดึงดูดของโลก แต่จะมีการเคลื่อนที่ด้วยแรงดูดซับ (capillary force) น้ำซึ่งอยู่ในสภาพนี้เรียกว่า capillary water ซึ่งการเคลื่อนที่จะช้ากว่ากรณี gravity water และจะมีทิศทางไปทางใดก็ได้ ส่วนน้ำที่ยึดติดกับเม็ดดินจะไม่เคลื่อนที่ เรียกว่า hygroscopic water

2.1.1 การเคลื่อนที่ของความชื้นในดิน

หลังจากที่น้ำซึมผ่านผิวดินลงมาแล้ว ก็จะไหลต่อไปด้วยแรงดึงดูดของโลกตามช่องว่างของดินและด้วยแรงดูดซับ(capillary) ตามช่องว่างขนาดเล็ก อัตราที่น้ำบนผิวดินไหลซึมเข้าไปในดินต่อหนึ่งหน่ายเวลาเรียกว่า infiltration rate หรือ intake rate อัตราดังกล่าวขึ้นอยู่กับองค์ประกอบหลายอย่างด้วยกัน เช่น ความลึกของน้ำที่ขังอยู่บนผิวดิน ลักษณะโครงสร้างของดิน เนื้อดิน อุณหภูมิของน้ำและดิน ตลอดจนจำนวนความชื้นที่มีอยู่ในดิน ในตอนแรกที่มีการให้น้ำแก่ดินอัตราการซึมผ่านผิวดินจะสูงเนื่องจากผิวดินยังแห้ง เมื่อดินชั้นบนเริ่มอิ่มตัว อัตราการซึมผ่านค่อยๆลดลง และในที่สุด ก็จะถึงจุดหนึ่งซึ่งอัตราการซึมผ่านผิวดินจะมีค่าคงที่ตลอดไปจนกว่าจะหยุดการให้น้ำ ค่าคงที่ดังกล่าวนี้จะประมาณเท่ากับความสามารถน้ำซึมผ่านได้ของดินนั่นเอง

2.1.2 น้ำในดิน (Soil Water)

น้ำเป็นปัจจัยที่สำคัญที่สุดปัจจัยหนึ่งสำหรับการดำรงชีวิตของสิ่งมีชีวิตทั้งหลาย สำหรับพืชแล้วน้ำมีความสำคัญเกี่ยวข้องอยู่กับระบบชีวิตของพืชหลายประการคือ

1. เป็นวัตถุดิบที่จำเป็นสำหรับขบวนการสังเคราะห์แสง (Photosynthesis) ในการสร้างอาหารของพืช

2. จำเป็นสำหรับการหล่อเลี้ยงเซลล์และช่วยให้เซลล์พืชเต่งตึง ซึ่งทำให้ต้นไม้ทรงตัวและยืนต้นอยู่ได้

3. เป็นตัวทำละลายธรรมชาติที่ดีที่สุดที่ละลายอาหารแร่ธาตุให้อยู่ในรูปที่พืชจะนำไปใช้ประโยชน์ได้

ดังนั้นจะเห็นได้ว่า หากพืชขาดน้ำไปเสียแล้วก็ไม่อาจมีชีวิตอยู่ได้ และน้ำซึ่งพืชจะได้มาเพื่อการดำรงชีวิตนั้นส่วนใหญ่ที่สุดจะได้มาจากน้ำในดินแทบทั้งสิ้น ดังนั้นน้ำในดินจึงมีความสำคัญและจำเป็นต่อพืชอย่างมากถ้าภายในของช่องว่างทั้งหมดของดินมีน้ำเข้าไปแทนที่อากาศจนเต็ม ดินนั้นก็จะเป็นดินที่อิ่มตัวด้วยน้ำ หรือดินอิ่มน้ำ (Saturated Soil) และเป็นปริมาตรน้ำสูงที่สุดที่ดินจะเก็บเอาไว้ได้

2.1.3 ความชื้นของดิน (Soil Moisture) [5]

ความชื้นในดินในที่นี้หมายถึง น้ำในดินเท่านั้น ซึ่งแบ่งได้เป็น

1. ความชื้นอิ่มน้ำ (Saturation) คือปริมาณน้ำในดินที่อยู่ในช่องว่างระหว่างเม็ดดินทั้งหมดทั้งช่องว่างขนาดเล็กและช่องว่างขนาดใหญ่ ถ้าดินระบายน้ำได้ดี น้ำที่อยู่ในช่องว่างขนาดใหญ่ก็เคลื่อนที่สู่เบื้องล่างด้วยแรงดึงดูดของโลกในระยะเวลาสั้น

2. ความชื้นชลประทานหรือความจุความชื้นในสนาม (Field Capacity) คือปริมาณน้ำในดินที่เหลือหลังจากน้ำอิสระถูกระบายออกจากช่องว่างขนาดใหญ่หมดแล้ว หรือปริมาณน้ำสูงสุดที่ดินสามารถ ดูดยึดไว้ได้จากแรงดึงดูดของโลก จึงทำให้ช่องว่างขนาดเล็กมีน้ำอยู่เต็ม และช่องว่างขนาดใหญ่มีอากาศอยู่เต็ม ระดับความชื้นชลประทานเป็นความชื้นที่มีประโยชน์สูงสุดต่อพืช คือรากพืชสามารถดูดน้ำไปใช้และอยู่ในดินได้นานพอที่พืชจะดูดไปใช้ได้อย่างเพียงพอ

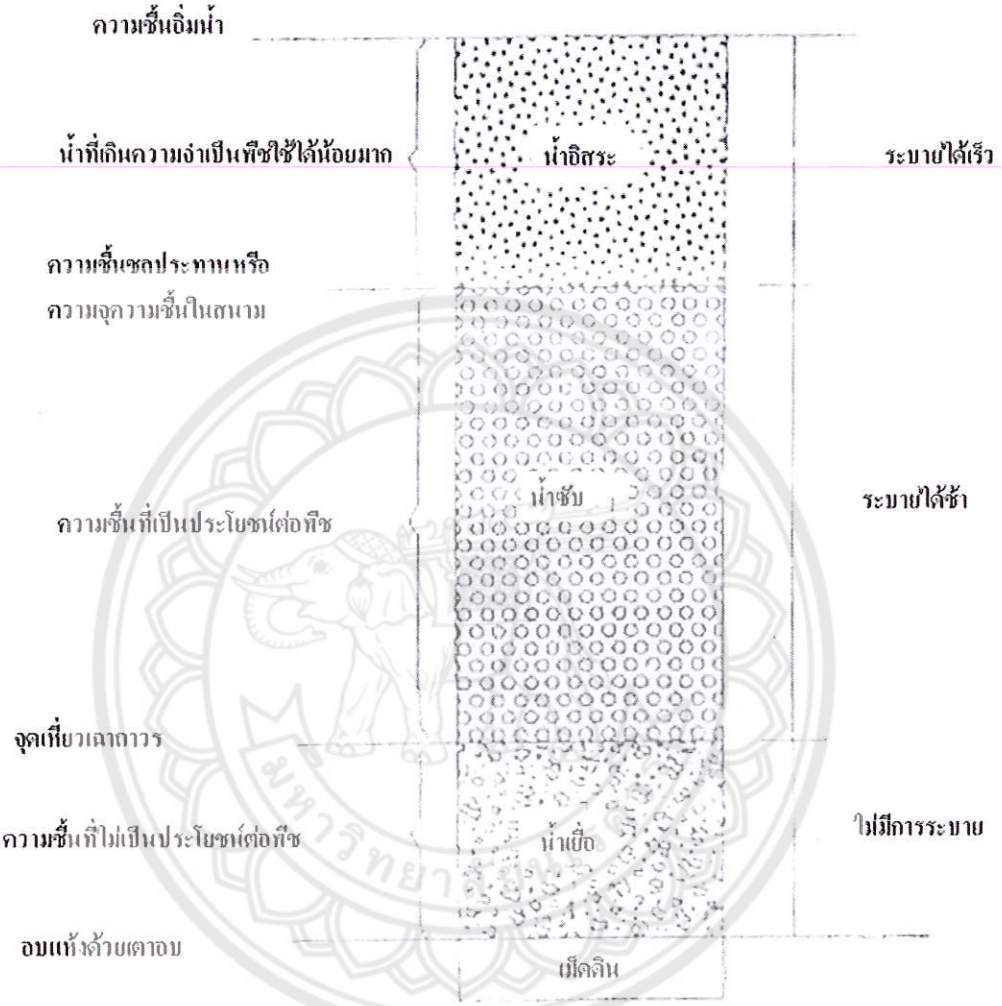
ตารางที่ 2.1 ค่าความชื้นชลประทานและความชื้นที่จุดเหี่ยวถาวรของดินชนิดต่างๆ

เนื้อดิน	ความชื้นชลประทาน (FC)		ความชื้นที่จุดเหี่ยวถาวร (PWP)	
	% โดยน้ำหนัก	มิลลิเมตร/เมตร	% โดยน้ำหนัก	มิลลิเมตร/เมตร
ดินทราย	9	149.5	4	66
ดินร่วนปนทราย	14	210	6	90
ดินร่วน	22	308	10	140
ดินร่วนปนเหนียว	27	364	13	175
ดินเหนียวปนทราย	31	403	15	195
ดินเหนียว	35	437	17	212.5

ที่มา : กลุ่มงานจักรกลการเกษตร. 2538: 10

3. ความชื้นจุดเหี่ยวถาวร (Permanent Wilting Point) คือปริมาณน้ำในดิน ที่พืชไม่สามรถดูดใช้ได้เพียงพอกับความต้องการคายน้ำ ถ้าหากไม่ได้รับน้ำเพิ่ม พืชก็จะเริ่มเหี่ยวเฉาจนกระทั่งเหี่ยวเฉาถาวร เรียกว่าเป็นความชื้นที่จุดเหี่ยวเฉาถาวร พืชจะเกิดการเหี่ยวเฉาได้หลายครั้ง โดยเฉพาะวันที่มีอากาศร้อนจัด มีความชื้นอากาศต่ำ มีลมแรง พืชใบบางและพืชใบกว้าง ทำให้มีการคายน้ำออกทางใบมาก เมื่อมากกว่าอัตราการดูดน้ำจากดิน พืชก็จะเริ่มเหี่ยวเฉา แต่เมื่ออากาศเย็นลงพืชก็จะสดชื่นเช่นเดิม

4. ความชื้นเมื่ออบแห้ง คือปริมาณน้ำในดินหลังจากถูกอบไว้ที่อุณภูมิ 105 – 110 องศาเซลเซียส นานกว่า 15 ชั่วโมง จนไม่มีน้ำระเหยออกมาจากดิน ถือว่าดินในสภาพนี้มีค่าแรงดึงความชื้นไม่น้อยกว่า 10,000 บาร์และจะใช้ดินอบแห้งเป็นหลักสำหรับการคำนวณหาค่าต่างๆ



รูปที่ 2.1 ระดับน้ำในดินและความชื้นของดินที่ระดับต่างๆ [5]

2.1.4 การวัดความชื้นของดิน [3]

มีอยู่หลายวิธี แต่ก็อาจแบ่งเป็นวิธีใหญ่ๆ ได้ 2 วิธีคือ

1. วัดโดยทางตรง

1.1 โดยการวิเคราะห์หาความชื้นโดยตรงจากดินตัวอย่างที่เก็บมาจากบริเวณที่เราต้องการทราบความชื้น วิธีนี้เป็นวิธีดั้งเดิม แต่ก็เป็นวิธีที่ดีที่สุดซึ่งทำได้โดยการเก็บดินบรรจุลงไปในกล่องโลหะแล้วปิดฝาให้มิดชิดแล้วชั่ง ต่อจากนั้นก็นำดินเข้าเตาอบที่ 105 – 110 องศาเซลเซียส เป็นเวลา 24 ชั่วโมง เพื่อไล่น้ำออก จากนั้นก็ชั่งอีกครั้งหนึ่งเพื่อหาปริมาณความชื้นที่หายไป ปริมาณความชื้นในดินก็อาจคำนวณออกมาเป็นเปอร์เซ็นต์โดยน้ำหนัก (percent by weight) Pw ได้คือ

Pw = [(น้ำหนักของดินชื้น – น้ำหนักดินที่เตาอบ) / น้ำหนักดินที่เตาอบ] * 100 (2.1)

1.2โดยการวิเคราะห์ปริมาณความชื้นของวัสดุพรุน (porous media) ที่ฝังอยู่ในดินที่ต้องการทราบปริมาณความชื้น วัสดุพรุนนี้ปกติใช้พวก gypsum block ซึ่งสามารถดูดความชื้นได้ โดยการฝัง gypsum block นี้ลงไปในดิน แล้วปล่อยทิ้งไว้ให้ความชื้นในดินแทรกซึมเข้าไปใน block นี้ จนกระทั่งความชื้นใน block อยู่ในสภาพสมดุลกับความชื้นในดินแล้วนำ gypsum block นี้มาชั่งคำนวณหาน้ำหนักของน้ำที่อยู่ใน gypsum block นั้น ซึ่งจะเป็นปริมาณของน้ำที่อยู่ในดินที่ต้องการจะทราบความชื้นนั่นเอง

2. การวัดโดยทางอ้อม

สามารถทำได้หลายวิธีคือ

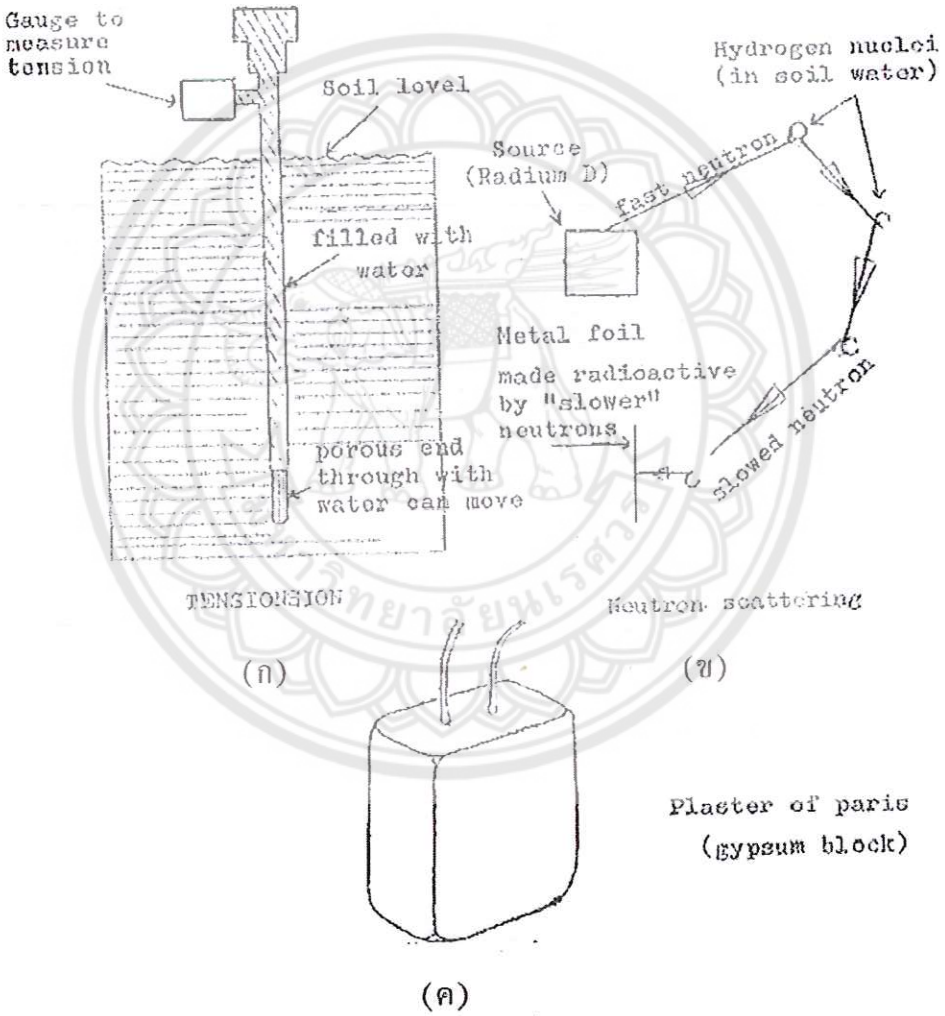
2.1วัด Conductance ของดิน สำหรับวิธีนี้เป็นการวัดความชื้นโดยอาศัยหลักที่ว่า น้ำในดินไม่ใช่น้ำบริสุทธิ์ และจะมีไอออนต่างๆละลายอยู่ ดังนั้นจึงเป็นสื่อไฟฟ้าได้ดี ถ้าน้ำในดินมากไอออนที่ละลายอยู่ก็จะมากด้วย conductivity ของดินนั้นก็จะสูง ในทางตรงข้ามถ้าในดินมีน้ำน้อย conductivity ของดินก็จะต่ำ จากความสัมพันธ์ระหว่าง conductivity ของดิน กับปริมาณความชื้นในดินจึงทำให้สามารถ วัดความชื้นในดินได้

2.2 Heat conductivity เป็นการวัดโดยอาศัยสมบัติบางประการของดินหรือ porous media ที่เกี่ยวข้องอยู่กับความร้อนก็ทำได้โดยอาศัยหลักการที่ว่า ถ้าดินที่มีความชื้นน้อย จะนำความร้อนได้ยากกว่าดินที่มีความชื้นมาก ความร้อนที่เกิดขึ้นก็จะแพร่กระจายออกไปจาก heating element ลงไปในดินแล้วผ่านกระแสไฟลงไป ถ้าดินที่มีความชื้นมาก ความร้อนที่เกิดขึ้นก็จะกระจายออกไปจาก heating element อย่างรวดเร็ว ดังนั้นความร้อนที่เกิดขึ้นก็จะไม่สะสมอยู่ในบริเวณนั้นมากนัก แต่ถ้าดินนั้นมีความชื้นน้อย หรือเป็นดินที่ค่อนข้างจะแห้ง ความร้อนที่เกิดขึ้นก็จะสะสมอยู่มาก จากความสัมพันธ์ระหว่างความชื้นในดินกับความสามารถในการนำความร้อนของดินสามารถทำให้วัดความชื้นในดินได้

2.3 การวัด Tension ของ porous cup ซึ่งอยู่ในสภาพที่สมดุลกับความชื้นในดินนั้น อาศัยกระบอกกลวงตอนปลายด้านหนึ่งประกอบด้วย porous cup ส่วนปลายด้านหนึ่งติดอยู่กับ monometer หรือ vacuum gage ก่อนวัดก็รินน้ำลงในกระบอกนั้นให้เต็มแล้วฝังปลายของกระบอกที่เป็นส่วนของ porous cup ลงไปในดิน ถ้าน้ำในดินมีน้อยและถูกยึดด้วยแรงที่สูงกว่าน้ำในกระบอก น้ำในกระบอกก็จะไหลออกมา เพื่อที่จะรักษาระดับ tension ให้เท่ากัน ซึ่งก็มีผลทำให้เข็มใน vacuum gage สูงขึ้น และจะสูงมากน้อยเพียงใดขึ้นอยู่กับระดับความชื้นในดินที่มีอยู่ในขณะนั้นทำให้สามารถรู้ปริมาณความชื้นในดินได้

2.4 Neutron Scattering โดยอาศัยหลักที่ว่าเมื่อส่ง neutron ออกจากเครื่องไปกระทบน้ำ ก็จะสะท้อนกลับเข้ามาในเครื่องอีก ถ้าน้ำในดินมีมาก ปริมาณ neutron ก็จะสะท้อนกลับมามาก ถ้าในดินมีน้ำน้อย neutron ที่สะท้อนกลับมาก็จะน้อย ซึ่งปริมาณของ neutron ที่สะท้อนกลับสามารถทำให้รู้ปริมาณความชื้นในดินได้

เนื่องจากเครื่องวัดชนิดนี้ใช้สารกัมมันตภาพรังสี จึงราคาแพงมากและอาจเป็นอันตรายต่อสุขภาพของผู้ใช้หากไม่ระมัดระวัง จึงไม่เป็นที่นิยม แต่ข้อดีของเครื่องวัดความชื้นชนิดนี้คือสามารถวัดความชื้นได้เร็ว



รูปที่ 2.2 เครื่องวัดความชื้นในดินแบบต่างๆ [3]

(ก) การวัด Tension ของ porous cup

(ข) การวัดโดยวิธี Neutron Scattering

(ค) การวัดConductivity โดย gypsum block

2.1.5 ความสัมพันธ์ของค่าตัวเก็บประจุ (Capacitance) และค่าคงตัวไดอิเล็คตริก (Dielectric) [3]

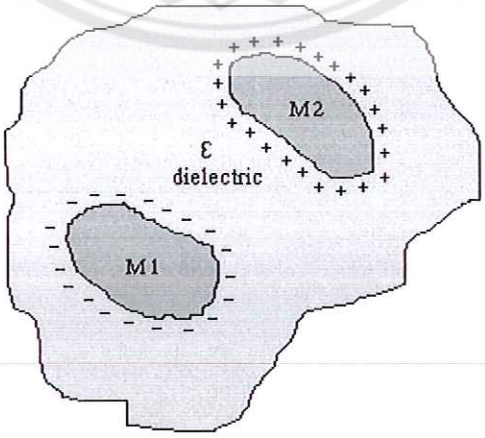
เนื่องจากน้ำมีค่าคงตัวของไดอิเล็กตริกสูง และดินมีค่าคงตัวของไดอิเล็กตริกต่ำ เมื่อปริมาณน้ำในดินเปลี่ยนแปลง ผลรวมของค่าไดอิเล็กตริกในดินจึงเปลี่ยนแปลงซึ่งก็มีผลต่อค่า capacitance ด้วย ซึ่งค่าคงที่ไดเล็กตริกของวัสดุต่างๆแสดงได้ดังนี้

ตารางที่ 2.2 ค่าคงที่ไดอิเล็กตริกของวัสดุต่างๆ

วัสดุ	ค่าคงที่ไดอิเล็กตริก
อากาศ	1.0006
แอลกอฮอล์	25
คาร์บอนไดออกไซด์	1.001
น้ำ(กลั่น)	80
น้ำ(ทะเล)	81
ดิน(แห้ง)	2.8
แก้ว	4 - 7
กระดาษ	3
ยาง	2.5-3

2.2 CAPACITANCE [10]

เราสามารถหาค่า capacitance ของตัวนำ 2 ตัว ซึ่งมีค่าประจุรวมเป็น บวก และ ลบ วางห่างกันในสภาวะที่มีค่า dielectric เดียวกัน ดังรูปที่ 2.3



รูปที่2.3 ตัวนำ 2 ตัวอยู่ที่มีค่า dielectric เดียวกัน [10]

จากรูปที่ 2.3 ได้สมการดังนี้

$$C = \frac{Q}{V_0} \tag{2.2}$$

โดยที่ C คือค่า capacitance

Q คือผลรวมของประจุที่เป็นบวก

V_0 คือความต่างศักย์ระหว่างตัวนำ 2 ตัว

หรืออาจเขียนใหม่ดังนี้

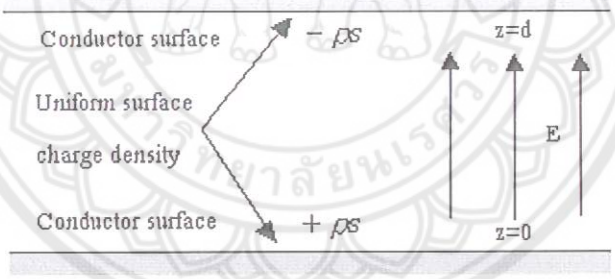
$$C = \frac{\oint \varepsilon E.dS}{-\int E.dL} \tag{2.3}$$

$$\varepsilon = \varepsilon_0 \varepsilon_r \tag{2.4}$$

$\varepsilon_0 = 8.854 \times 10^{-12}\ F/m$

ε_r = relative dielectric constant

S คือพื้นที่ผิวของตัวเก็บประจุ



รูปที่ 2.4 แผ่นตัวนำวางขนานกัน [10]

เราสามารถให้นิยามใหม่ของ capacitance จากตัวเก็บประจุ 2 ตัวมาเป็นแผ่นตัวนำ 2 แผ่น วางขนานห่างกันเป็นระยะทาง d ให้เริ่มจาก z=0 ถึง z=d มีสภาวะเป็น $\pm \rho$ s ดังรูปที่ 2.4

ความต่างศักย์ระหว่างแผ่นตัวนำ 2 แผ่นคือ

$$V_0 = -\int_{upper}^{lower} E.dL = -\int_0^d \frac{\rho s}{\varepsilon} dz = \frac{\rho s}{\varepsilon} d \tag{2.5}$$

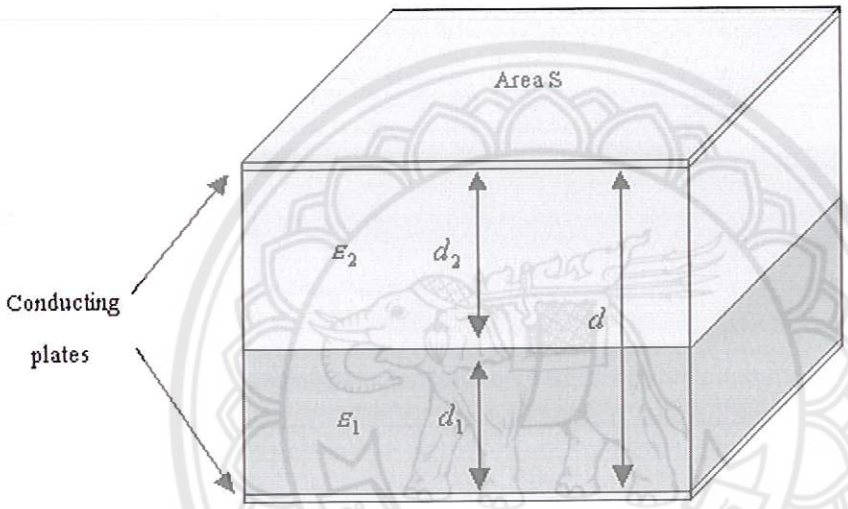
และ

$$Q = \rho s S \tag{2.6}$$

จากสมการ V_0, Q ได้สมการ capacitance ใหม่

$$C = \frac{Q}{V_0} = \frac{\varepsilon S}{d} \tag{2.7}$$

อีกกรณีหนึ่งคือถ้าแผ่นตัวนำมีสารหรือวัสดุหุ้มอยู่มาประกบกันจะทำให้มีค่า dielectric 2 ค่าแยกกันดังรูปที่ 2.5



รูปที่ 2.5 แผ่นตัวนำที่มีวัตถุหุ้มวางขนานกัน [10]

จากรูปที่ 2.5 จะเห็นว่าเหมือนกับมีตัวเก็บประจุมาต่ออนุกรมกันอยู่จึงเขียนสมการได้ดังนี้

$$C = \frac{1}{\frac{1}{C_1} + \frac{1}{C_2}} = \frac{1}{\frac{d_1}{\varepsilon_1 S} + \frac{d_2}{\varepsilon_2 S}} \tag{2.8}$$

2.3 พื้นฐานไมโครคอนโทรเลอร์ [2]

ไมโครคอนโทรเลอร์เป็นอุปกรณ์ทางอิเล็กทรอนิกชนิดหนึ่ง ซึ่งมีความสามารถในการใช้เป็นอุปกรณ์ควบคุมที่สามารถโปรแกรมการทำงานได้ ซึ่งปัจจุบันนี้มีการใช้งานไมโครคอนโทรเลอร์กันอย่างแพร่หลาย เนื่องจากใช้งานง่ายและสามารถประยุกต์ใช้งานได้หลายอย่าง

2.3.1ส่วนประกอบของไมโครคอนโทรเลอร์

ในปัจจุบันนี้ไมโครคอนโทรเลอร์มีการผลิตขึ้น ซึ่งมีหลายแบบหลายรุ่นขึ้นกับบริษัทผู้ผลิตแต่โดยทั่วไปแล้วไมโครคอนโทรเลอร์จะประกอบด้วยส่วนประกอบที่คล้ายกันดังนี้

1.1 หน่วยประมวลผลกลาง (Central Processor Unit : CPU) จะเป็นส่วนที่จะเป็นตัวตัดสินใจต่างๆ ซึ่งจะทำงานตามโปรแกรมที่เราป้อนเข้าไปในตัวไมโครคอนโทรเลอร์

1.2 หน่วยความจำ (Memory) เป็นตัวที่จะเก็บข้อมูลต่างๆที่ต้องใช้ในไมโครคอนโทรเลอร์ ไม่ว่าจะเป็นหน่วยความจำโปรแกรมหรือหน่วยความจำข้อมูล ซึ่งมีหลายชนิดได้แก่ ROM, EPROM, EEPROM, RAM, FLASH

1.3 พอร์ตสัญญาณเข้าและออก (Input & Output Port) เป็นส่วนที่จะใช้ในการติดต่อกับอุปกรณ์ภายนอก

1.4 คุณสมบัติอื่นๆ ไมโครคอนโทรเลอร์สมัยใหม่จะมีฟังก์ชั่นพิเศษเพิ่มเติมเช่น Timer/Counter, Analog to Digital Converter, AnalogComparator, UART/USART

2.3.2 PIC microcontroller

PIC คือ microcontroller อีกตระกูลหนึ่ง ย่อมาจากคำว่า Peripheral Interface Controller ซึ่ง concept ของ microcontroller ตระกูลนี้ก็คือ พยายามรวมเอาทุกอย่างเอาไว้ในตัวของมันไม่ว่าจะเป็น PROGRAM MEMROY, RAM, EEPROM, SERIAL, I2C, PWM, A/D ฯลฯ โดยไม่จำเป็นต้องต่ออุปกรณ์เสริมจากภายนอก ในตัวของ PIC จะมีฟังก์ชันที่ใช้ในการประมวลผล รวมทั้งหน่วยความจำ ซึ่งทำให้มันเหมือนกัน CPU

2.3.3 หน่วยความจำของ PIC

ในอดีตหน่วยความจำของ PIC จะค่อนข้างน้อย คืออยู่ระหว่าง 512 words ถึง 4K words แต่ในปัจจุบัน บริษัท microchip ซึ่งเป็นเจ้าของ PIC ได้พัฒนาจนทำให้ memory ของ PIC มีขนาดเป็นหลายสิบกิโลไบต์ และมีทีท่าว่าจะขยายได้ใหญ่ขึ้นเรื่อยๆ ในเรื่องของการนับขนาดของหน่วยความจำของ PIC จะนับไม่เหมือนปกติ โดยที่ หนึ่งคำสั่งของ PIC จะมีขนาด 14 bits ดังนั้นเราจะเรียกว่า 1 word ของ PIC จะมีขนาด 14 bits เช่น PIC16F84A ระบุว่ามีหน่วยความจำ 1 K (ซึ่งหมายถึง 1 Kword ถ้าคำนวณให้เป็นแบบ 1 byte = 8 bit จะได้ว่า 1 x 1,024 x 14 = 14,336 bits ดังนั้นก็คือ 14,336/(8 x 1,024) = 1.75K bytes นั่นเอง

2.3.4 สถาปัตยกรรมของ PIC

PIC จะยึดถือการออกแบบที่ว่ารวมทุกอย่างไว้ใน chip ตัวเดียวโดยไม่ต้องต่ออุปกรณ์ใดๆ เพิ่มเติม ผลที่ตามมาก็คือแผ่นวงจรจะมีขนาดเล็ก และอุปกรณ์ที่ใช้จะไม่มาก บางงานอาจจะใช้แค่ PIC เพียงตัวเดียวโดยไม่ต้องใช้ chip อื่นมาเพิ่มเติมเลย นี่คือคุณสมบัติพิเศษของ PIC ซึ่งปัจจุบันหลายบริษัทที่ผลิต microcontroller ก็เริ่มจะหัดมาเลียนแบบแนวทางนี้ครับ แต่ทุกอย่างย่อมมีข้อเสีย เนื่องจาก concept ที่จะรวมทุกอย่างไว้ใน chip เดียว ทำให้ program memory และ data memory ไม่สามารถขยายโดยใช้กับ memory ภายนอกได้

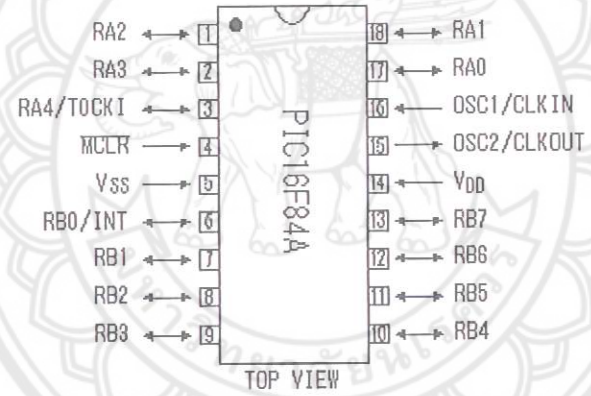
2.3.5 การเขียนโปรแกรม Microcontroller

ภาษาที่ใช้สำหรับการเขียนโปรแกรมบนMicrocontoller แบ่งได้เช่นเดียวกับการเขียนโปรแกรมบนคอมพิวเตอร์คือ ภาษาระดับสูง และภาษาระดับต่ำ

ภาษาระดับสูงเช่น C, Basic ข้อดีคือเขียนง่าย, แก้ไขเปลี่ยนแปลง หรือเพิ่มเติมได้ง่าย ส่วนข้อเสียก็คือการทำงานจะช้า ขนาดโปรแกรมที่เขียนมีขนาดใหญ่

ภาษาระดับต่ำ ซึ่งก็คือ ภาษา Assembly ข้อดีคือ ตัว compiler แจกฟรี ขนาดโปรแกรมหลังจาก compiled แล้วมีขนาดเล็ก โปรแกรมมีความเร็ว แต่ข้อเสียก็คือเขียนยาก เพราะลักษณะภาษาไม่ค่อยสื่อความหมาย แก้ไขเปลี่ยนแปลงยาก

2.4 ไมโครคอนโทรเลอร์ PIC16F84A [7]



รูปที่ 2.6 โครงสร้างขาของ PIC16F84A [7]

2.4.1 คุณสมบัติทางเทคนิคของ PIC 16F84A

1. ซีพียูเป็นแบบ RISC (Reduce Instruction-Set Computer) มีคำสั่งใช้งานเพียง 35 คำสั่ง
2. ความถี่สัญญาณนาฬิกา ตั้งแต่ไฟตรงถึง 20 MHz
3. ขนาดหน่วยความจำโปรแกรม 1024 เวิร์ด
4. หน่วยความจำแรมข้อมูล 68 ไบต์
5. หน่วยความจำข้อมูลอีอีพรอม 64 ไบต์
6. ตอบสนองแหล่งกำเนิดอินเตอร์รัปต์ได้ 10 แหล่ง
7. มีสเต็ก 8 ระดับ

8. มีวงจรเพาเวอร์ออนรีเซต (POR) , เพาเวอร์อัปไทเมอร์ (PWRT) และออสซิลเลเตอร์สตาร์ตอัปไทเมอร์ (OST)

9. มีวอตช์ด็อกไทเมอร์ (WDT) ที่มีวงจรออสซิลเลเตอร์ในตัว ทำให้มีความน่าเชื่อถือในการทำงานสูง

10. เลือกป้องกันข้อมูลทั้งในหน่วยความจำโปรแกรมและหน่วยความจำข้อมูลและเลือกระดับป้องกันได้

11. เลือกใช้วงจรกำเนิดสัญญาณนาฬิกาได้ 6 โหมดหลัก

1. โหมด EC ใช้สัญญาณนาฬิกาจากภายนอก

2. โหมด ER ใช้ตัวต้านทางภายนอก

3. โหมด INTRC ใช้วงจร RC ภายในไมโครคอนโทรเลอร์มี 2 ความถี่ให้เลือก

4. โหมด LP ใช้คริสตอลพลังงานต่ำ ความถี่สูงไม่เกิน 200 kHz

5. โหมด XL ใช้คริสตอลความถี่ตั้งแต่ 2 MHz สูงสุดไม่เกิน 4 MHz

6. โหมด HS ใช้คริสตอลความถี่สูง สูงสุดไม่เกิน 20 MHz

12. สามารถโปรแกรมในวงจรได้

13. ไฟเลี้ยง +2 ถึง +5.5 V

14. กระแสซิงค์และซอร์สของพอร์ต 25mA

15. ไทเมอร์ 1 ตัว คือ TMR0

16. การใช้พลังงานไฟฟ้าในกรณีไม่ขับโหลด

น้อยกว่า 2 mA ที่ +5V และสัญญาณนาฬิกา 4MHz, 15 μ A ที่ +2V และสัญญาณนาฬิกา 32 kHz น้อยกว่า 0.5 μ A ในโหมดประหยัดพลังงานที่ไฟเลี้ยง +3V

2.4.2 รายละเอียดขาของไมโครคอนโทรเลอร์ PIC 16F84A [7]

ตารางที่ 2.3 ขาของไมโครคอนโทรเลอร์ PIC 16F84A

ชื่อขา	ตำแหน่งขา	ชนิดของขา	ชนิดของวงจรบัฟเฟอร์	รายละเอียดการทำงาน
V_{DD}	14	อินพุต	-	- ขาต่อไฟเลี้ยงบวก ตั้งแต่ 2-5V
Vss	5	อินพุต	-	- ขาต่อกราวน์
ขาพอร์ต A เป็นขาพอร์ต 2 ทิศทาง				
RA0	17	อินพุต/เอาต์พุต	ทีทีแอล	- ขาพอร์ต RA0

RA1	18	อินพุต/เอาต์พุต	ทีทีแอล	- ขาพอร์ต RA1
RA2/VREF	1	อินพุต/เอาต์พุต	ทีทีแอล	- ขาพอร์ต RA2
RA3	2	อินพุต/เอาต์พุต	ทีทีแอล	- ขาพอร์ต RA3
RA4/TOCKI	3	อินพุต/เอาต์พุต	ชมิตต์ทริกเกอร์	- ขาพอร์ต RA4 - อินพุตสัญญาณนาฬิกาของไทเมอร์ 0
MCLR	4	อินพุต/เอาต์พุต	ชมิตต์ทริกเกอร์	- ขารีเซตหลัก - อินพุตรับแรงดันสูง สำหรับการโปรแกรม
Osc2/CLKout	15	เอาต์พุต	-	- เอาต์พุตของสัญญาณนาฬิกาหลัก เมื่อทำงานในโหมด ER มีความถี่เท่ากับ1/4 ของความถี่ที่ขา OSC1
Osc1/CLKin	16	อินพุต	ชมิตต์ทริกเกอร์	- ขาต่อคริสตอลอินพุตจากภายนอก
ขาพอร์ต B เป็นขาพอร์ต 2				
RB0/INT	6	อินพุต/เอาต์พุต	ทีทีแอล/ชมิตต์ทริกเกอร์	- ขาพอร์ต RB0 - อินพุตรับสัญญาณอินเทอร์รัปต์จากภายนอก
RB1	7	อินพุต/เอาต์พุต	ทีทีแอล	- ขาพอร์ต RB1
RB2	8	อินพุต/เอาต์พุต	ทีทีแอล	- ขาพอร์ต RB2
RB3	9	อินพุต/เอาต์พุต	ทีทีแอล	- ขาพอร์ต RB3

RB4	10	อินพุต/ เอาต์พุต	ทีทีแอล	- ขาพอร์ต RB4 - สามารถกำเนิดสัญญาณ อินเทอร์รัปต์อันเนื่องมาจาก มีการเปลี่ยนแปลงลอจิกที่ขานี้
RB5	11	อินพุต/ เอาต์พุต	ทีทีแอล	- ขาพอร์ต RB5 - สามารถกำเนิดสัญญาณ อินเทอร์รัปต์อันเนื่องมาจาก มีการเปลี่ยนแปลงลอจิกที่ขานี้
RB6	12	อินพุต/ เอาต์พุต	ทีทีแอล/ ชมิตต์ทริก เกอร์	- ขาพอร์ต RB6 - สามารถกำเนิดสัญญาณ อินเทอร์รัปต์อันเนื่องมาจาก มีการเปลี่ยนแปลงลอจิกที่ขานี้
RB7	13	อินพุต/ เอาต์พุต	ทีทีแอล/ ชมิตต์ทริก เกอร์	- ขาพอร์ต RB7 - สามารถกำเนิดสัญญาณ อินเทอร์รัปต์อันเนื่องมาจาก มีการเปลี่ยนแปลงลอจิกที่ขานี้

จากตารางสามารถสรุปการจัดตำแหน่งขาและหน้าที่ได้ดังนี้

OSC1/CLKIN : Oscillator crystal input /External clock source input.

OSC2/CLKOUT : Oscillator crystal output.

MCLR(inv) : Master clear(reset)input /Programming voltage input.

RA0 - RA3 : Bi-directional I/O port.

RA4/T0CKI : Bi-directional I/O port.

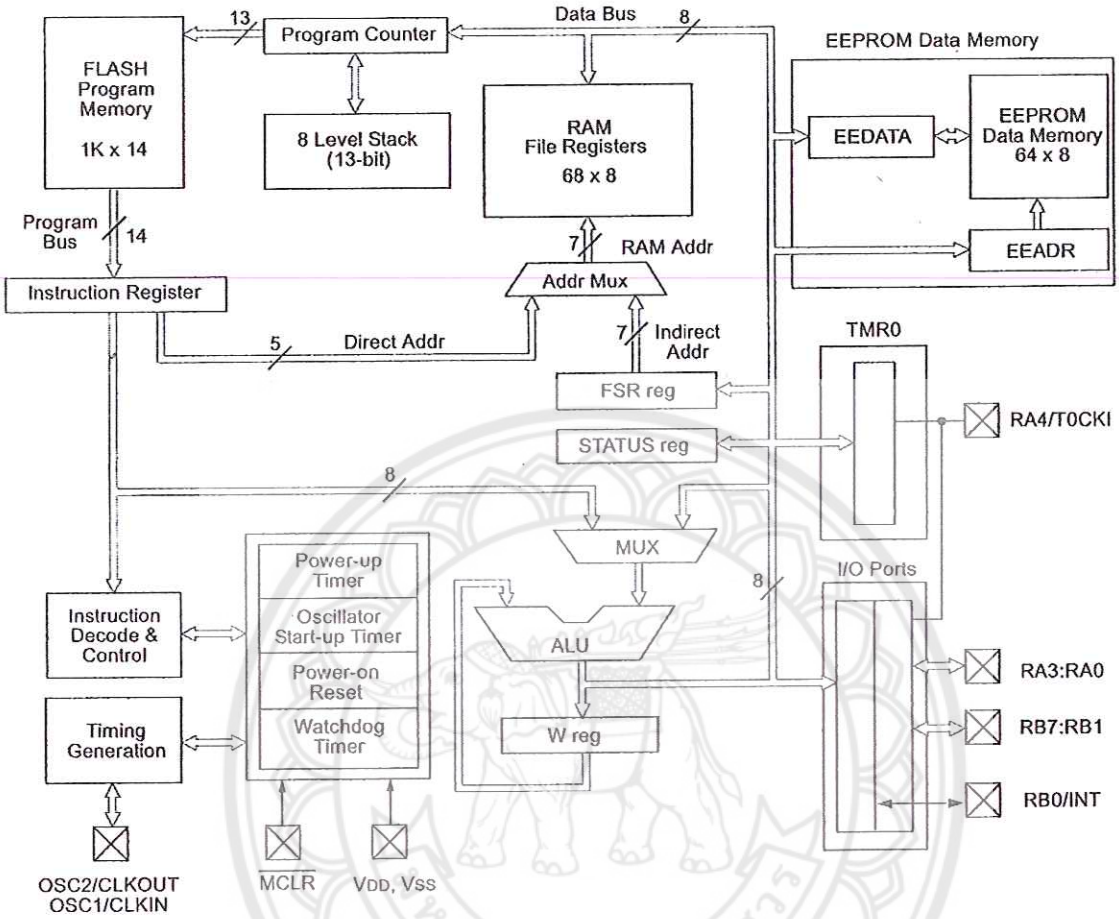
RB0/INT : Bi-directional I/O port.

RB1 - RB7 : Bi-directional I/O port.

V_{ss} : Ground

V_{DD} : Positive supply(+2.0V to +5.5V)

2.4.3โครงสร้างภายในของ PIC 16F84A



รูปที่ 2.7 โครงสร้างภายในของ PIC 16F84A [6]

FlashProgramMemory

Flash memory เป็นพื้นที่หน่วยความจำสำหรับเก็บ program ที่เราเขียนขึ้น ซึ่งมีขนาด1,024 words ถึงแม้ว่าจะไม่มีไฟฟ้าจ่ายให้กับ MCU ข้อมูลที่เก็บอยู่ใน flash memory ก็จะไม่หายไป จุดเด่นของ Flash memory ก็คือสามารถเขียนทับเข้าไปใหม่ได้หลายๆ ครั้ง ซึ่งจำนวนครั้งจะอยู่ที่ประมาณ 1000 ครั้ง

ภายใน program memory อาจสามารถแบ่งเป็นส่วนๆ ได้ดังนี้

Reset Vector (0000h) เมื่อการ reset เกิดขึ้นเนื่องจากมีการป้อนไฟเข้า MCU เป็นครั้งแรก หรือเกิด WDT(Watchdog Timer) time-out หรือในกรณีอื่นๆ โปรแกรมจะเริ่มต้นหลังจาก reset ณ ตำแหน่งนี้

Peripheral Interrupt Vector (0004h) เมื่อเกิดการ interrupt ขึ้น program memory pointer จะชี้มายัง ณ ตำแหน่งนี้

Configuration word (2007h) การทำงานเบื้องต้นของ PIC จะถูกกำหนดที่หน่วยความจำตรง

นี้ ไม่ว่าจะเป็น Enable/Disable Power-up timer, Enable/Disable Watchdog timer, Oscillator Selection bits(กำหนดที่มาของสัญญาณนาฬิกา) หน่วยความจำที่ตำแหน่งนี้ไม่สามารถเขียนได้ด้วยการเขียนโปรแกรม จะต้องกำหนดในขณะทำการเขียนโปรแกรมลงสู่ Flash memory ของ MCU

RAM(Random Access Memory) File Registers

หน่วยความจำของ RAM ภายใน PIC16F84(A) จะมีโครงสร้างเป็นแบ่งเป็น bank โดยจะมีขนาด 80 bytes(00h-4Fh) ต่อ bank ในกรณีของ PIC16F84A จะมี 2 banks Memory ในส่วนนี้จะถูกแบ่งออกเป็น2ส่วนคือ

ส่วนแรก 12 bytes(00h-0Bh) ของแต่ละ bank ซึ่งจะถูกเรียกว่า SFR(Special Function Registers) จะใช้สำหรับบันทึกสถานการณ์ทำงานของ PIC, เงื่อนไขของการกำหนดสถานะของ port ว่าเป็น input/output ports และเงื่อนไขอื่นๆ

ส่วนที่สองจะมีขนาด 68 bytes(0Ch-4Fh) ซึ่งเริ่มตั้งแต่ ไบต์ที่ 13 จะถูกเรียกว่า GPR(General Purpose Registers) ซึ่งสามารถใช้เป็นหน่วยความจำที่จะเก็บผลลัพธ์ และเงื่อนไขต่างๆ เพื่อใช้ในการประมวลผลโปรแกรมขณะโปรแกรมทำงาน ถึงแม้ว่า PIC16F84A จะมี 2 bank แต่ SFR จะมีทั้งหมดเพียง 16 ชนิดไม่ใช่ 24 ชนิด โดย SFR บางตัวจะมีอยู่ทั้ง 2 bank ส่วน GPR ถึงแม้จะเปลี่ยน bank ก็ยังคงชี้ไปยังตำแหน่งเดิมเพราะ 16F84A มี GPR อยู่เพียง bank เดียวเท่านั้น และเมื่อไม่มีไฟฟ้าจ่ายให้กับ PIC ค่าใน memory ส่วนนี้จะหายไปหมด อย่างไรก็ตามพื้นที่ใน GPR สามารถเขียนใหม่กี่ครั้งก็ได้ ไม่มีข้อจำกัดจำนวนครั้ง

EEPROM (Electrically Erasable ProgrammableRead Only Memory)

หน่วยความจำในส่วนนี้เมื่อไม่มีไฟฟ้าจ่ายให้กับ MCU แล้วข้อมูลที่อยู่ภายในยังคงอยู่จะไม่หายไป และสามารถเขียนด้วยคำสั่งของ program จะมีขนาด 64 bytes. อย่างไรก็ตามการเขียนซ้ำก็มีข้อจำกัดโดยสามารถเขียนทับใหม่ได้ประมาณ 1 ล้านครั้ง ดังนั้นหน่วยความในส่วนนี้จะใช้เก็บข้อมูลที่ไม่ค่อยจะเปลี่ยนแปลงบ่อยนัก หน่วยความจำในส่วนนี้สามารถเก็บข้อมูลได้นาน 40 ปี

SFR Registers

SFR(Special Function Registers) มีอยู่ 16 ชนิดด้วยกัน ซึ่งสามารถอ้างถึงด้วยการ เปลี่ยนตำแหน่งของ bank ที่จะอ้าง ทั้งหมดจะมีขนาด 160 bytes ตัวแปรใน SFR มีดังนี้

INDF : จะเก็บค่าของ Data memory ที่ถูกชี้แบบ indirect addressing

TMR0 : เป็นTimer counter ของ Timer 0

PCL : เก็บค่า 8 bits ล่างของ program counter

STATUS : จะเก็บค่า Flag ของผลลัพธ์ที่เกิดจากการคำนวณ

FSR : เป็น pointer ใช้สำหรับอ้างอิง data memory แบบ indirect

PORTA : เก็บค่าสถานะของ PORTA

PORTB : เก็บค่าสถานะของ PORTB

EEDATA : เก็บค่าของ Data ที่ EEPADR ชี้อยู่

EEADR : ตำแหน่งของ EEPROM ที่ต้องการอ้างถึง

PCLATH : เป็น 5 bits บนของ program counter

INTCON : ใช้คำคุมการเกิด Interruption

OPTION_REG : ใช้สำหรับกำหนด Mode การทำงานของ MCU

TRISA : ใช้กำหนด Mode ของ PORTA ว่าเป็น INPUT หรือ OUTPUT

TRISB : ใช้กำหนด Mode ของ PORTB ว่าเป็น INPUT หรือ OUTPUT

EECON1 : เป็น register ที่ใช้ควบคุม EEPROM

EECON2 : เป็น register ที่ใช้ป้องกันการเขียน EEPROM

Program Counter

เป็น counter ที่แสดงถึงตำแหน่ง address ของ program ที่เขียนเข้าไปไว้ใน flash memory ที่กำลังทำการประมวลผล ซึ่งจะเป็น counter ขนาด 13 bits โดยทั่วไปแล้ว counter ตัวนี้จะเพิ่มขึ้น 1 ทุกๆ ครั้งเมื่อมีการประมวลผลคำสั่งเกิดขึ้น 1 ครั้ง ซึ่งค่าที่แสดงก็คือตำแหน่งของคำสั่งต่อไปที่จะทำการประมวลผล แต่เมื่อประมวลคำสั่ JUMP ตัว counter จะมีค่าเท่ากับตำแหน่งที่คำสั่ง JUMP นั้นอ้างถึง

8 Level Stack

stack คือ memory ซึ่งจะเก็บค่า return address ของ program ตัวอย่างเช่น เมื่อต้องทำการประมวลผลอย่างหนึ่งหลายๆ ครั้ง ซึ่ง program ในส่วนนี้ถูกสร้างเป็น subroutine ไว้ ในตอนสุดท้ายของ subroutine ก็จะมีคำสั่ง RUTURN ทุกครั้ง ในการเรียกใช้เราจะใช้คำสั่ง CALL ในการเรียก subroutine ตำแหน่ง program address ที่ถัดจากคำสั่ง CALL ก็จะถูกเก็บลงสู่ stack (กระบวนการนี้บางครั้งจะเรียกว่า PUSH) หลังจากได้ประมวลผลคำสั่งใน subroutine แล้ว ในตอนสุดท้ายเมื่อมาเจอคำสั่ง RETURN มันก็จะทำการกระโดดไปยังตำแหน่งที่เก็บไว้ใน stack (กระบวนการนี้บางครั้งจะเรียกว่าPOP) แต่เนื่องจากว่า stack มีขนาดเพียง 8 เท่านั้น นั่นก็หมายความว่าเราสามารถเรียก คำสั่ง CALL ได้ เก้าครั้งติดต่อกันเท่านั้น ซึ่งถ้าใช้คำสั่ง CALL ไปมากกว่านั้นโดยไม่ RETURN กลับ ค่า stack จะถูกทับเป็นผลทำให้เมื่อใช้คำสั่ง RETURN ก็จะไม่สามารถกลับไปยังตำแหน่งเดิมได้

Instruction Register

คำสั่งต่างๆ ของโปรแกรม ที่ถูกชี้โดย program counter จะถูกอ่านข้าไปยัง register ตัวนี้ โดยกระบวนการนี้จะถูกเรียกว่า FETCH.

Instruction Decode & Control

คำสั่งที่ถูก FETCH ไว้ใน instruction registers จะถูกแปลความหมายและทำงานตามคำสั่งนั้น

Multiplexer and Arithmetic Logic Unit

โดยการแปลความหมายและทำงานตามคำสั่งจะถูกกระทำโดย Multiplexer และ the Arithmetic Logic Unit(ALU)

W Register

ย่อมาจาก work register มันจะมีหน้าที่สำหรับเก็บผลของการคำนวณที่เกิดจาก ALU เอาไว้ชั่วคราว เพื่อที่จะนำมาคำนวณต่อไป ตัวของมันจะทำหน้าที่เป็นตัวกลางในการคำนวณต่างๆ และมันยังทำหน้าที่ส่งผ่านสถานะoutput ของ input-output port อีกด้วย

STATUS Register

เป็น register ซึ่งจะเก็บค่าผลของ ALU (เช่น ผลลัพธ์ของการ บวก ลบ ของ register เป็น 0, บวก, ลบ), เงื่อนไขการเกิด timeout , เป็นตัวกำหนดว่าขณะนี้ PIC อ้าง register ที่ bank ไหน

FSR Register

FSR(File Select Register) ใช้สำหรับอ้างตำแหน่งของ RAM ในรูปแบบ indirect address การอ้างแบบ direct address ก็คือรูปแบบที่อ้างถึง Address นั้นโดยเฉพาะเจาะจงเลย เช่น movfw h'20' ซึ่งหมายความว่าทำการอ่านค่า ที่ address 20 มาเก็บไว้ที่ w register ในกรณีนี้สามารถอ้างตำแหน่งได้ตั้งแต่ 0 ถึง 127 หรืออ้างได้เพียง 7 bit ซึ่งจะอยู่ในขอบเขต 1 bank ในการที่จะเปลี่ยน bank จำเป็นที่จะต้องเกี่ยวข้องกับ RP0 bits ของ STATUS register การอ้างแบบ indirect address โดยใช้ FSR register จะนิยมใช้ในการอ้าง address ที่อยู่ติดๆ กันด้วยการอาศัยคำสั่ง inc FSR เพื่อเลื่อนไปยังตำแหน่ง memory ถัดไป

Address Multiplexer

ใช้เป็นตัวแบ่งแยกว่าเป็น indirect addressing หรือ the direct address. ซึ่งหมายความว่าทำการอ่านค่า ที่ address 20 มาเก็บไว้ที่ w register ในกรณีนี้สามารถอ้างตำแหน่งได้ตั้งแต่ 0 ถึง 127 หรืออ้างได้เพียง 7 bit ซึ่งจะอยู่ในขอบเขต 1 bank ในการที่จะเปลี่ยน bank จำเป็นที่จะต้องเกี่ยวข้องกับ RP0 bits ของ STATUS register การอ้างแบบ indirect address โดยใช้ FSR register จะนิยมใช้ในการอ้าง address ที่อยู่ติดๆ กันด้วยการอาศัยคำสั่ง inc FSR เพื่อเลื่อนไปยังตำแหน่ง memory ถัดไป

EEDATA

เป็น register ที่จะใช้เมื่อมีการอ่านหรือเขียนข้อมูล EEPROM

EEADR

เป็น register ที่ใช้สำหรับอ้าง address ของ EEPROM. ซึ่งใน PIC16F84A จะมีหน่วยความจำ EEPROM อยู่ทั้งหมด 64 bytes เมื่อจะทำการเขียน EEPROM จะต้อง เขียนข้อมูล 55h และ AAh ไปยัง EECON2 เสียก่อนจึงจะเริ่มต้นการเขียน EEPROM ได้

Timer

PIC16F84A จะมี timer เพียงแค่ตัวเดียว (TMR0) มีขนาด 8 bits การทำงานของมันก็คือมันจะทำการนับไปเรื่อยๆ และจะเกิดการ time-out เมื่อการนับมาถึง 256 ซึ่งจะทำให้ T0IF bit ของ INTCON register ซึ่งเป็น SFR กลายเป็น "1" ซึ่งมีผลทำให้เกิดการ interrupt เกิดขึ้นเมื่อเกิดการ time-out. สำหรับการที่จะกำหนดว่าจะให้มีการ interrupt ของ TMR0 เกิดขึ้นได้หรือไม่นั้นกำหนดได้จาก the GIF bit

และ T0IE bit ของ INTCON register ซึ่งเป็น SFR โดยถ้าเป็น "1" ก็หมายความว่ากำหนดให้มีการ interrupt เกิดขึ้นได้

I/OPorts

PIC16F84 (A) จะมี I/O Ports ทั้งหมด 13 ขา ซึ่งการกำหนดว่าจะให้ขานั้นเป็น INPUT หรือ OUTPUT นั้นสามารถกำหนดได้ด้วยโปรแกรม ทั้ง 13 ขานั้นสามารถแบ่งออกได้เป็น 2 กลุ่มด้วยกันก็คือ 5 ขาเป็น A port และอีก 8 ขาที่เหลือเรียกว่า B port

TimingGeneration

PIC จะมีวงจรภายในที่จะสร้างสัญญาณนาฬิกาในการกำหนดความจังหวะของการทำงานของตัวมัน โดยสัญญาณนาฬิกานี้จะมีแหล่งกำเนิดมาจาก crystal หรือ ceramic oscillator จากภายนอก เมื่อสัญญาณนาฬิกาต้องการความแม่นยำสูงเราจะต้องเลือกใช้ crystal แต่โดยปกติทั่วๆไปแล้วจะใช้ ceramic resonator ต่อเข้ากับ capacitors เป็น module อยู่ภายนอก PIC16F84A จะ execute 1 คำสั่ง(1 cycle) จะใช้สัญญาณนาฬิกา 4 pulses โดยจะใช้ pipeline architecture แต่ในกรณีของคำสั่ง JUMP จะใช้ 2 cycle สำหรับเวลาที่ใช้ในการ execute นั้นโดยปกติแล้วจะใช้เวลา 200 nanoseconds ถ้าใช้ crystal ที่ความถี่ 20MHz, 1/(20MHz) = 50 nanoseconds. หมายความว่าสามารถ execute คำสั่ง 5,000,000 instructions ภายในเวลา 1 วินาที

Initialization circuits

ภายใน PIC16F84A จะต้องมีการกำหนดคุณสมบัติการทำงานหลายอย่างดังนี้

POWER ON Timer : เป็น Timer ที่ใช้สำหรับสร้างช่วงระยะเวลาก่อนที่จะใช้ MCU ทำงานเพื่อรอจนกระทั่งแรงดันไฟฟ้าที่ป้อนให้กับ MCU จะนิ่งในกรณีเมื่อมีการป้อนไฟเข้าไปใหม่

OSC StartTimer : เป็น Timer ที่ใช้สำหรับสร้างช่วงระยะเวลาก่อนที่จะใช้ MCU ทำงานเพื่อรอจนกระทั่งสัญญาณนาฬิกาที่ป้อนให้กับ MCU จะนิ่งในกรณีเมื่อมีการป้อนไฟเข้าไปใหม่

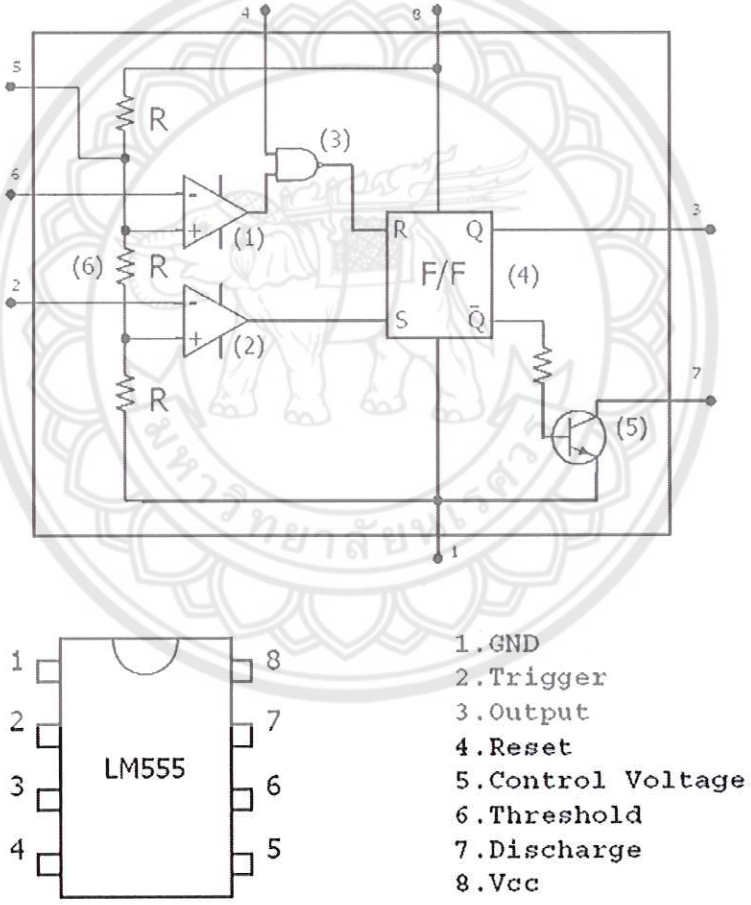
POW ON Reset : จะทำการ RESET วงจรภายใน PIC ใหม่เมื่อมีการป้อนไฟเข้าไปใหม่

Watchdog Timer : เป็น Timer สำหรับจับเวลาว่าโปรแกรมบางช่วงใน PIC ทำงานนานเกินไปหรือไม่ เพื่อป้องกันอาการที่เรียกว่า Dead Lock ซึ่ง Timer ตัวนี้จะต้องทำการ clear ด้วยคำสั่งทาง software เมื่อ timer ตัวนี้นับจนกระทั่ง time-out เกิดขึ้น PIC จะกลับไปอยู่ในสถานะเหมือนกับสภาพที่มีการป้อนไฟเข้าไปใหม่

2.5 IC เบอร์ 555 [8]

IC เบอร์ 555 เป็นไอซี ที่นิยมใช้กันมากในการนำไปสร้างสัญญาณรูปคลื่นแบบต่างๆ เช่น สัญญาณ SquareWave , สัญญาณพัลส์ สัญญาณ ramp และวงจรตั้งเวลา ไอซีเบอร์ 555 เป็นอุปกรณ์วงจรรวมที่มีอุปกรณ์อิเล็คทรอนิคส์อื่นๆ อยู่ภายใน และมีส่วนที่ต้องต่อภายนอกเพื่อควบคุมการทำงานและใช้งานเป็นลักษณะต่างๆ ซึ่งง่ายต่อการออกแบบ และง่ายในการสร้างสัญญาณพัลซ์ความถี่ต่างๆ อีกทั้งสามารถเข้าใจการทำงานได้ง่าย นอกจากไอซีเบอร์ 555 แล้วยังมีไอซีเบอร์ 556 ที่เป็นแบบ Dual Timer ประกอบด้วย ไอซีเบอร์ 555 จำนวน 2 ตัว อยู่ภายในตัวเดียวกัน เพื่อใช้เป็นวงจรตั้งเวลา และสะดวกในการออกแบบวงจรที่ต้องใช้ไอซีเบอร์ 555 หลายๆตัว

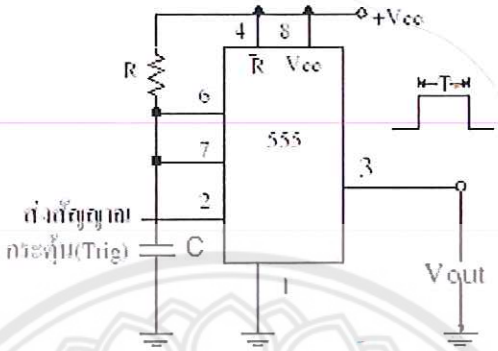
ส่วนประกอบของไอซีเบอร์ 555



รูปที่ 2.8 ส่วนประกอบของไอซีเบอร์555 [8]

2.5.1 การใช้ 555 เป็นโมโนสเตเบิล มัลติไวเบรเตอร์

โมโนสเตเบิลเป็นวงจรที่เอาต์พุตเสถียรที่ระดับลอจิก "0" หรือ "1" เพียงสถานะเดียวเมื่อได้รับสัญญาณ กระตุ้นจะเปลี่ยนสถานะของเอาต์พุตไปเป็นสถานะที่ไม่เสถียรชั่วขณะนานเท่ากับ ช่วงเวลาที่กำหนดโดย RC วงจรพื้นฐานเป็น



รูปที่2.9 โมโนสเตเบิลมัลติไวเบรเตอร์ [8]

2.5.2 การสร้างสัญญาณ Single Shot

โดยหลักการของการสร้างสัญญาณแบบ Single Shot คือวงจรโมโนสเตเบิลมัลติไวเบรเตอร์นั่นเอง เพียงแต่สร้างสัญญาณ ครั้งเดียวแล้วหยุด การหาค่าคาบเวลาทำได้เช่นเดียวกับวงจรโมโนสเตเบิลมัลติไวเบรเตอร์ ซึ่งหาคาบเวลาได้จากสมการ

$$T = 0.693\ RC \qquad (2.9)$$

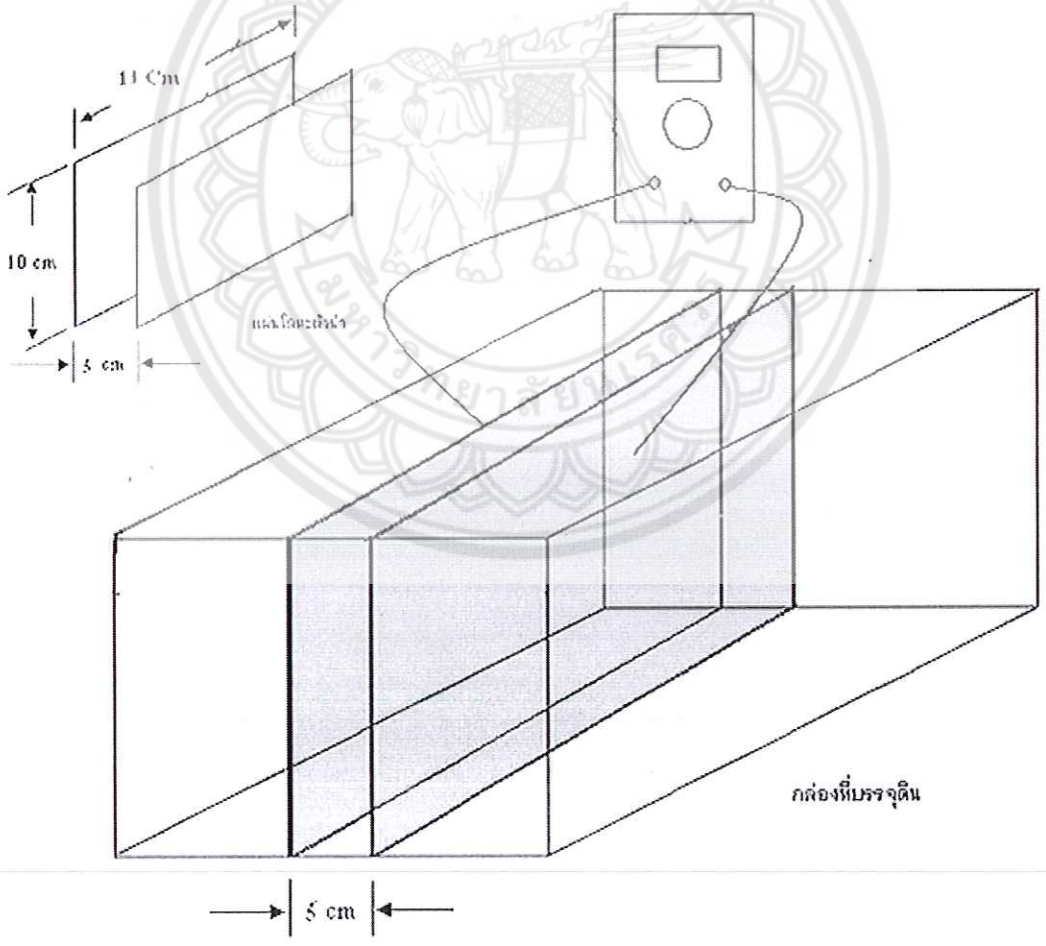
หลังจากได้ทราบทฤษฎีต่างที่เกี่ยวข้องแล้ว ทำให้สามารถทำการออกแบบเพื่อสร้างเครื่องวัดความชื้นในดินขึ้นได้ โดยได้นำเสนอในบทที่3 ซึ่งเป็นการออกแบบและการสร้าง

บทที่ 3
การออกแบบและการสร้าง

ดังที่ได้กล่าวมาแล้วว่าความชื้นในดินมีผลต่อค่าความจุไฟฟ้า ซึ่งจำเป็นอย่างยิ่งที่ต้องมีอุปกรณ์ที่จะมาทำการวัดความจุไฟฟ้า ดังนั้นในที่นี้จึงจะกล่าวถึงการทดลองหาความสัมพันธ์ระหว่างดินและค่าความจุไฟฟ้าและการออกแบบเครื่องมือวัดความจุไฟฟ้า เพื่อนำข้อมูลมาใช้ในการทำเครื่องวัดความชื้นในดิน

3.1 การสร้างชุดทดลองวัดความชื้น

ในการสร้างชุดทดลองนั้น ผู้สร้างได้ทำการสร้างกล่องสี่เหลี่ยมขึ้นมาเพื่อทำการใส่ดินและโลหะตัวนำสองแผ่นมาเพื่อใช้ในการวัดและทดลองดังรูป



รูปที่ 3.1 การออกแบบชุดทดลองวัดความชื้นในดิน

3.1.1 การทดลองวัดความชื้นในดินช่วงแรก

สำหรับการทดลองวัดความชื้นในช่วงแรกนี้ เป็นการทดลองค่าความสัมพันธ์ระหว่างความชื้นในดินกับค่า ความจุไฟฟ้าของดิน โดยใช้ LCR มิเตอร์

เพื่อให้เห็นว่าน้ำมีผลต่อการเปลี่ยนแปลงของค่าความจุไฟฟ้าของดิน จึงได้ออกแบบและทำการทดลอง โดยการนำหลักการ Capacitance มาใช้ ซึ่งนำแผ่นตัวนำ 2 แผ่น วางขนานกัน ระห่างช่องว่างนั้นนำดินที่มีความชื้นต่างๆกันใส่ แล้วทำการวัดค่าประจุ

วิธีการทดลอง

1.นำแผ่นโลหะตัวนำที่หุ้มด้วยฉนวนบาง ขนาด 13x10cm 2 แผ่นวางขนานกัน ให้ระยะห่างประมาณ 5 cm ทำการวัดค่าประจุโดยใช้ มัลติมิเตอร์ แล้วบันทึกผล

2.นำแผ่นตัวนำใส่ลงในกล่องที่บรรจุดินทรายอยู่ โดยรักษาระยะห่างเอาไว้ จากนั้นทำการวัดค่าประจุ

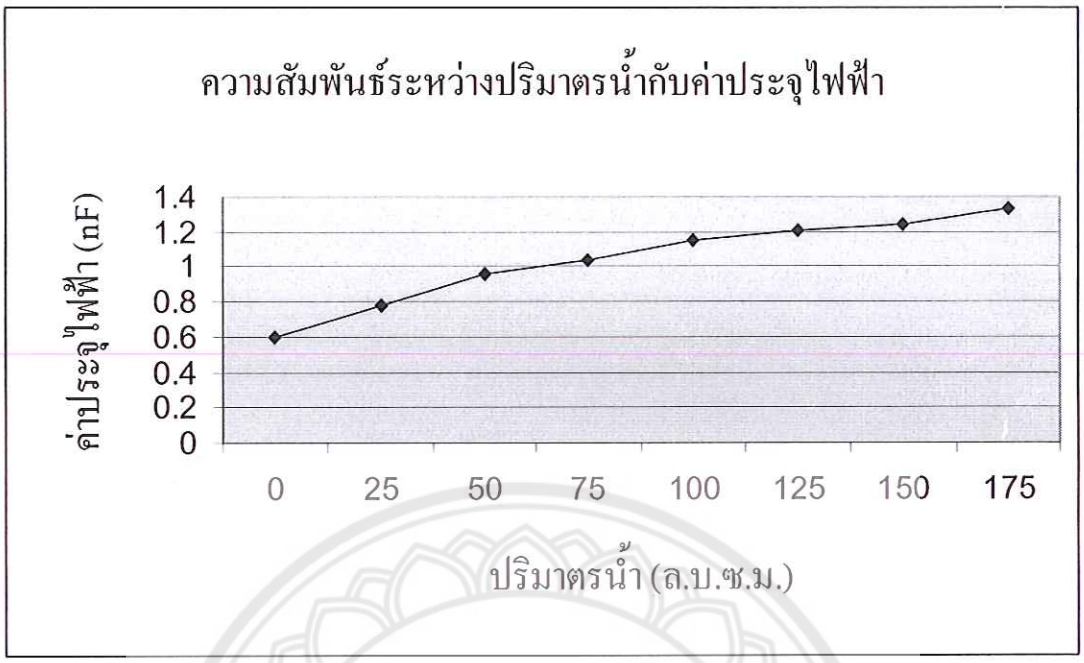
3.เริ่มใส่น้ำลงในกล่องทรายครั้งละ25 cm^3 แล้วบันทึกความแตกต่างของประจุที่ได้จากการใส่น้ำแต่ละครั้ง แล้วนำข้อมูลมาวาดกราฟ

(หมายเหตุ: การทดลองนี้ใช้กล่องพลาสติกทำกล่องใส่ทรายและใช้อะลูมิเนียมในการทำแผ่นโลหะตัวนำ และใช้เครื่อง Multi Meter ทำการวัดค่า)

ผลการทดลอง

ตารางที่ 3.1 ค่า Capacitance และค่าปริมาตรน้ำ

ปริมาตรน้ำ(cm^3)	*Capacitance*
0	0.6 nF
25	0.78 nF
50	0.96 nF
75	1.04 nF
100	1.15 nF
125	1.10 nF
150	1.24 nF
175	1.33 nF



รูปที่ 3.2 กราฟความสัมพันธ์ระหว่างปริมาตรน้ำกับค่าประจุ

จะเห็นได้ว่ากราฟระหว่างค่าความจุไฟฟ้าและปริมาตรน้ำแปรผันตรงกัน ซึ่งทำให้ทราบว่าปริ--มาตร น้ำที่เปลี่ยนไปมีผลอย่างมากต่อค่าความจุไฟฟ้าในดิน

3.1.2 การทดลองวัดความชื้นในดินระยะที่สอง

ทำการเปลี่ยนกล่องภาชนะที่ใส่ดินเป็นกล่องอะคริลิค โลหะตัวนำจากแผ่นPCBและระยะห่างระหว่างแผ่นโลหะทั้งสองเป็น 5 เซนติเมตร และใช้ดินร่วน ดินทรายและดินเหนียวในการทดลองทำการวัดค่าความจุไฟฟ้าโดยใช้ LCR Meter

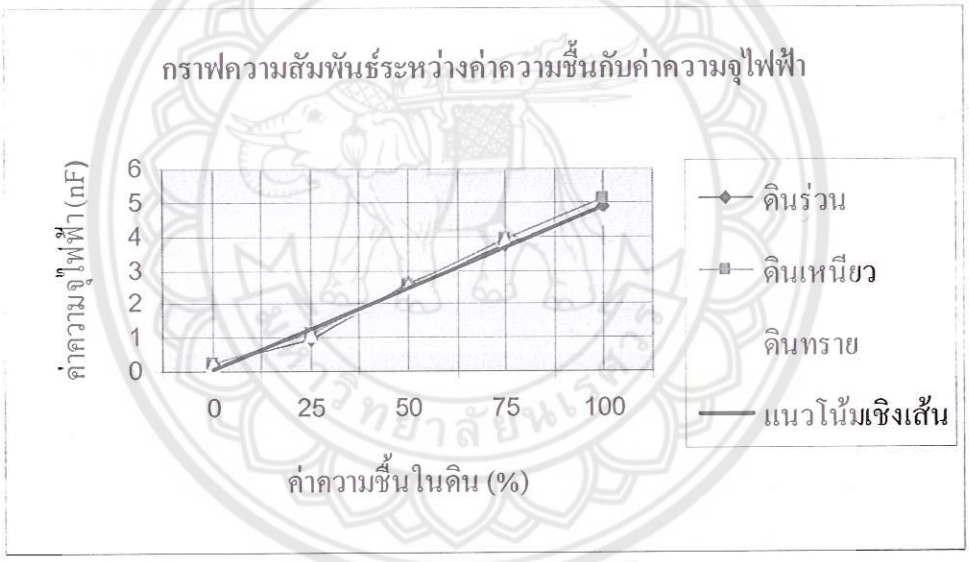
ขั้นตอนการทดลอง

เตรียมดินชนิดต่างใส่กล่อง กิโลกรัมแล้วเ ติมน้ำครั้งละ 0.2 กิโลกรัม การทำการเติมน้ำแต่ละครั้งให้ทำการวัดค่าความจุไฟฟ้าในดินและก่อนการเติมน้ำครั้งต่อไปให้เว้นระยะประมาณ 5 นาที เพื่อให้น้ำได้ซึมสู่ดินอย่างทั่วถึง

ผลการทดลอง

ตารางที่ 3.2 แสดงค่าความจุไฟฟ้ากับปริมาตรน้ำในดินชนิดต่างๆ

ปริมาตรน้ำ		ค่าความจุไฟฟ้า		
		ดินทราย	ดินร่วน	ดินเหนียว
0 กิโลกรัม	0 %	0.23 nF	0.28 nF	0.29 nF
0.2กิโลกรัม	25%	1.08 nF	0.96 nF	1.12 nF
0.4 กิโลกรัม	50%	2.47 nF	2.61 nF	2.57 nF
0.6กิโลกรัม	75%	3.84 nF	3.74 nF	3.81 nF
0.8 กิโลกรัม	100%	4.86 nF	4.94 nF	5.01 nF

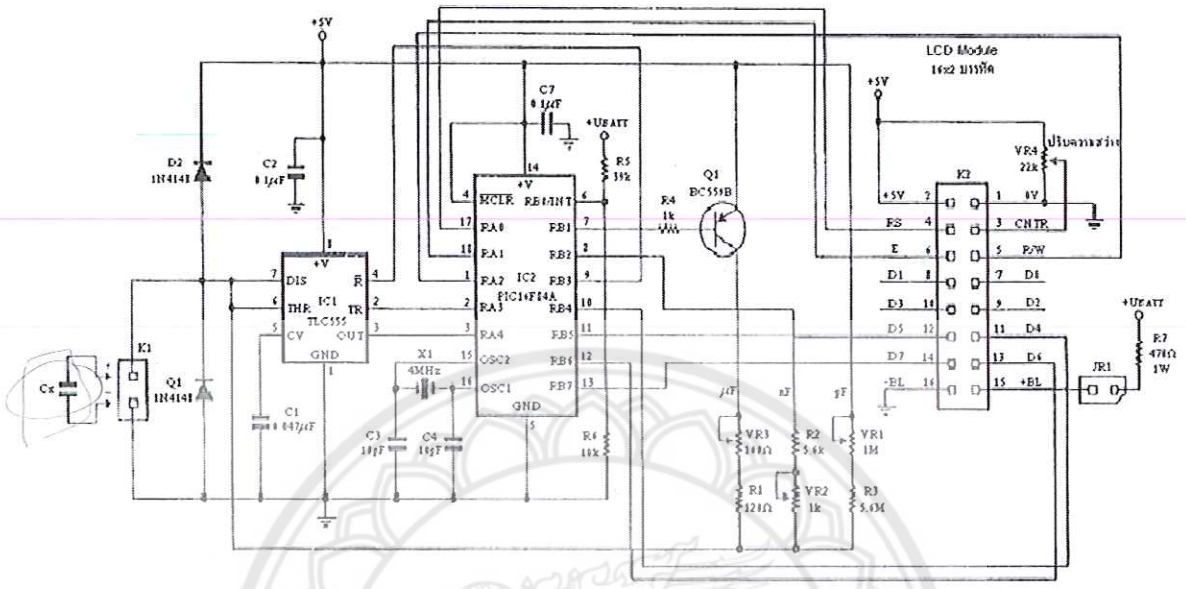


รูปที่ 3.3 กราฟความสัมพันธ์ระหว่างค่าความชื้นกับค่าความจุไฟฟ้าในดินชนิดต่างๆ

เมื่อนำผลการทดลองมาทำการ plot graph พบว่ากราฟที่ที่ได้นั้น ถึงแม้ค่าที่วัดได้จะแตกต่างกันตามสภาพดิน แต่ค่าที่ได้ก็ใกล้เคียงกัน เมื่อเพิ่มเส้นแนวโน้มเชิงเส้นให้กราฟแล้วค่าที่ออกมาแทบจะเป็นค่าเดียวกัน ดังนั้นจึงสามารถนำค่าที่ได้จากเส้นแนวโน้มเชิงเส้นมาเป็นข้อมูลในการทำเครื่องมือวัดความชื้น ในดินได้

3.2 การออกแบบและการสร้างเครื่องวัดความจุไฟฟ้า

3.2.1วงจรและการออกแบบ



รูปที่ 3.4 วงจรเครื่องวัดค่าความจุไฟฟ้า [9]

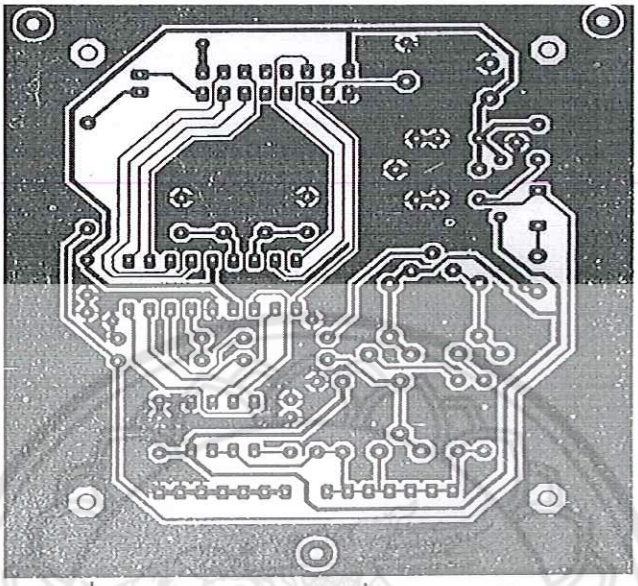
3.2.2 หลักการทำงานของวงจร

จากรูปที่ 3.4 จะเห็นได้ว่าตัวเก็บประจุที่ต้องการทำการวัดหรือ C_x จะถูกนำมาต่อเข้ากับคอนเน็กเตอร์ K1 ของวงจร ซึ่งจะเชื่อมต่อเข้ากับวงจรไทเมอร์ด้วยไอซี 555 โดยวงจรทำงานเป็นวงจรโมโนสเตเบิ้ลมัลติไวเบรเตอร์ (monostable multivibrator : MMV) และทำงานร่วมกับไมโครคอนโทรเลอร์ PIC16F84

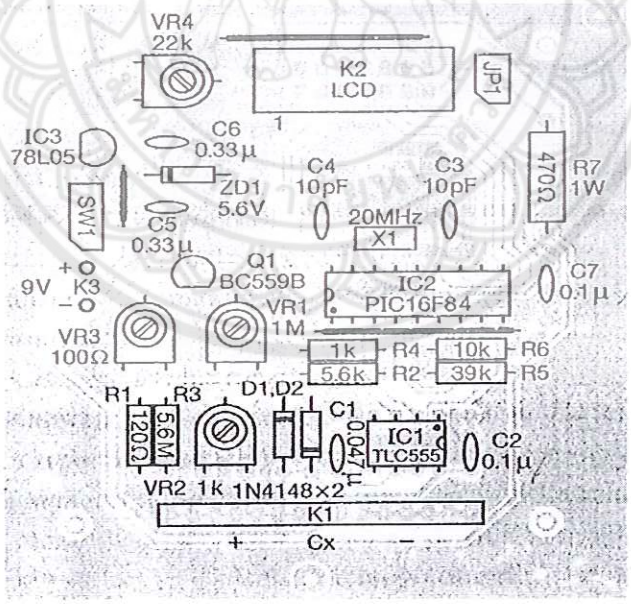
ไมโครคอนโทรเลอร์ PIC16F84 ในวงจรนี้ถูกกำหนดให้ทำงานที่สัญญาณนาฬิกา 20 เมกะเฮิตซ์ และทำหน้าที่ควบคุมการทำงานของไอซี 555 โดยใช้ขาเอาต์พุตพอร์ต 2 ขา ต่อควบคุมไปที่ไอซี 555 ที่ขารีเซต (reset) หรือขา R และขาอินพุตทริกเกอร์ (trigger) หรือขา TR ในขณะเดียวกันไมโครคอนโทรเลอร์ก็จะคอยตรวจสอบจับสัญญาณที่ขา OUT ของไอซี 555 ด้วย

ตัวเก็บประจุที่นำมาต่อวัด (C_x) จะมีผลต่อการเปลี่ยนแปลงสัญญาณพัลซ์ที่ขา OUT ของไอซีไทเมอร์ 555 โดยถ้าหากนำตัวเก็บประจุมาต่อวัด จะทำให้ช่วงเวลาที่เป็นลอจิกไฮของพัลซ์เอาต์พุตยาวนานมากขึ้น ช่วงเวลาตลอดที่พัลซ์มีค่าลอจิกไฮนี้เอง ไมโครคอนโทรเลอร์จะทำการนับโดยเพิ่มค่าขึ้นเรื่อยๆ และเมื่อสัญญาณเอาต์พุตของไอซี 555 ตกกลับมาเป็นลอจิกล์อีกครั้ง ผลที่ได้จากการนับจะถูกนำไปประมวลผลและแสดงผลเป็นค่าความจุที่วัดได้ต่อไป และแสดงผลทางจอ LCD

ทางด้านของจอแสดงผล LCD นั้นจะใช้แบบจอแสดงผล LCD โมดูล 2 บรรทัด 16 ตัวอักษร ใช้วิธีการเชื่อมต่อเพื่อควบคุมในโหมด 4 บิด และต่อเข้ากับ PIC



รูปที่ 3.5 ลายทองแดงของเครื่องวัดความจุไฟฟ้า [9]



รูปที่ 3.6 แสดงการลงอุปกรณ์ด้านบนของเครื่องวัดความจุไฟฟ้า [9]

3.2.3 การสร้าง

หลังจากได้จัดเตรียมแผ่นวงจรพิมพ์ที่มีลายทองแดงตามแบบและเตรียมอุปกรณ์ต่างๆ ตามรายการอุปกรณ์จนครบแล้ว ก็เริ่มทำการลงอุปกรณ์ต่างๆ ที่ด้านบนของแผ่นวงจรพิมพ์ตามลำดับ

การทดสอบการทำงานของวงจรเบื้องต้น โดยนำแบตเตอรี่ 9 โวลต์มาเชื่อมต่อและเปิดสวิตช์เครื่อง นำโวลต์มิเตอร์มาตรวจเช็คว่ามีแรงดัน +5 โวลต์จ่ายให้กับ IC1 และ IC2 ที่ขาไฟเลี้ยงตามวงจรถูกต้องหรือไม่ ซึ่งผู้สร้างพบว่าแรงดันที่ได้จากถ่านนั้นไม่มีความเสถียรเพียงพอเมื่อใช้ไปไม่นานแรงดันก็ลดอย่างรวดเร็วเนื่องจากวงจรดึงโหลดมาก ทำให้แรงดันที่ขา IC ทั้งสองแรงดันไม่ถึงจึงปรับมาใช้หม้อแปลงทดแทนซึ่งพบว่า แรงดันที่ขา IC ทั้งสองคงที่และเสถียรเพียงพอ

3.2.4 การปรับแต่งเครื่องโดยใช้การวัดเทียบกับตัวเก็บประจุอ้างอิง

การปรับแต่งเครื่องครั้งสุดท้ายจะเป็นการปรับแต่งเครื่องอีกครั้ง โดยใช้วิธีการวัดค่าของตัวเก็บประจุที่ใช้อ้างอิง ซึ่งเป็นตัวเก็บประจุชนิดความเที่ยงตรงสูง คือมีค่าความผิดพลาดประมาณ 1 เปอร์เซ็นต์ โดยควรจะหาตัวเก็บประจุอ้างอิงนี้มาใช้ปรับแต่งในย่านพิโกฟารัด และนาโนฟารัด

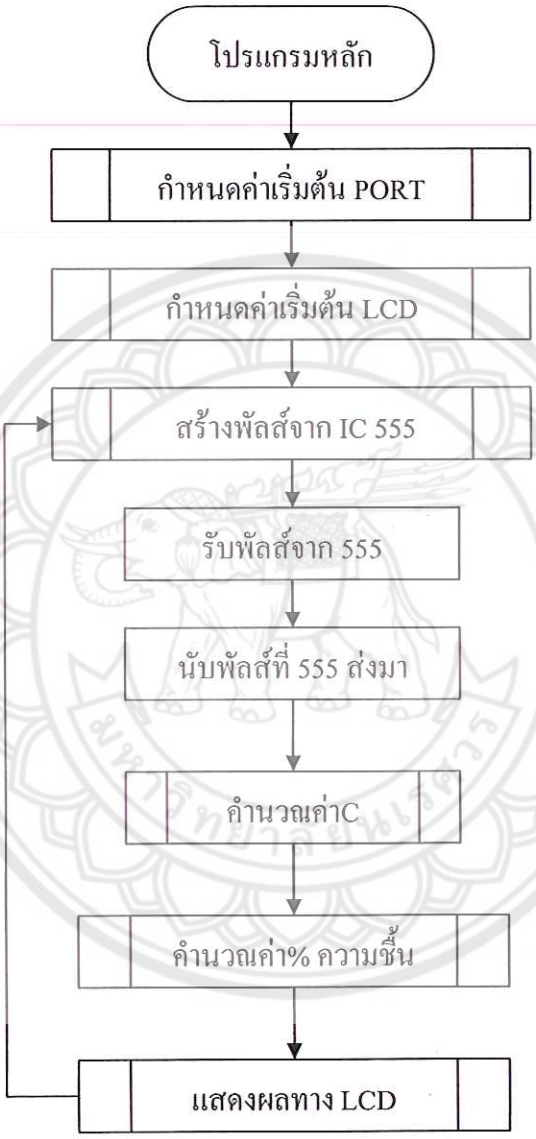
จากนั้นจึงนำตัวเก็บประจุอ้างอิงค่า 470 พิโกฟารัด ต่อเข้ากับจุดวัดและทำการปรับ VR1 จนอ่านค่าได้ 470 พิโกฟารัด ทำซ้ำอีกครั้งโดยใช้ตัวเก็บประจุค่า 220 นาโนฟารัด และทำการปรับที่ VR2 จนอ่านค่าได้ 220 นาโนฟารัด และเพื่อความแน่ใจในเครื่องวัดที่สร้างขึ้น ผู้สร้างจึงได้นำเครื่องที่สร้างขึ้นไปทำการสอบเทียบที่ห้องปฏิบัติการสอบเทียบฝ่ายปฏิบัติการภาคตะวันออกเฉียงเหนือ การไฟฟ้าฝ่ายผลิตแห่งประเทศไทยซึ่งผลการทดสอบดังต่อไปนี้

ตารางที่ 3.3 ผลการสอบเทียบเครื่องวัดค่าความจุไฟฟ้าที่สร้างขึ้น

UUT Range	Nominal Value	Tolerance (±)	Resolution	Unit	UUT Reading	| Error |	Uncertainty
200	190	5.8	0.1	pF	188.9	1.1	-
2	1.9	0.039	0.001	ηF	1.920	0.020	-
20	19	0.39	0.01	ηF	18.96	0.04	-
200	190	3.9	0.1	ηF	189.3	0.7	-
2	1.9	0.039	0.001	μF	1.909	0.009	-
20	19	0.60	0.01	μF	18.81	0.19	-
200	190	6.0	0.1	μF	189.8	0.2	-
2000	1900	98	1	μF	-	-	-

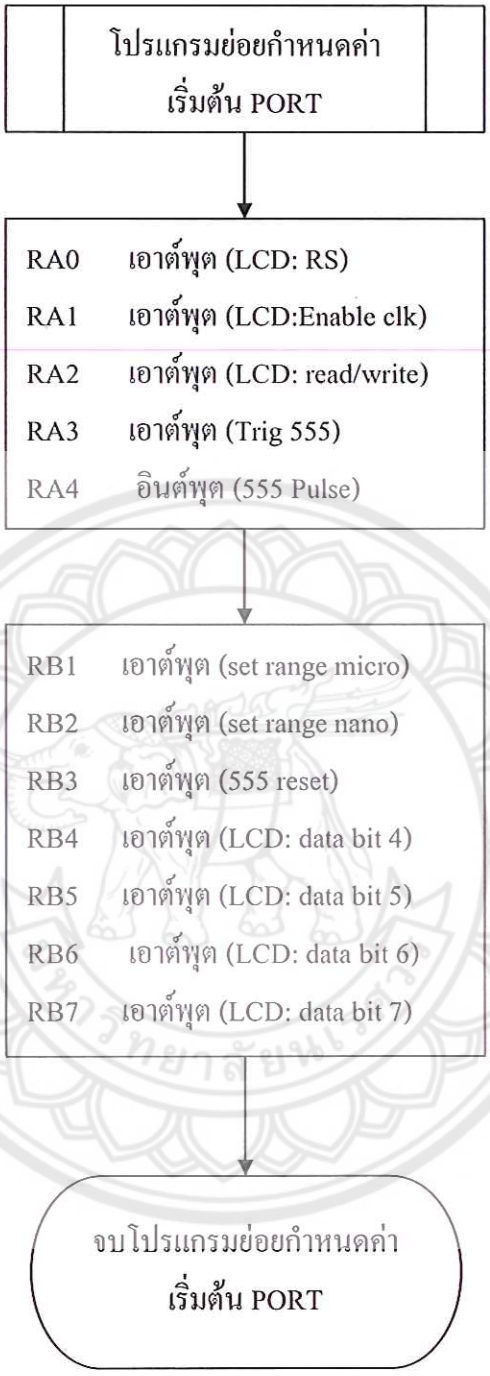
3.3 การปรับปรุงเครื่องวัดค่าประจุไฟฟ้าให้เป็นเครื่องวัดความชื้นในดิน

ในการปรับปรุงนี้ เราเน้นปรับปรุงที่ตัวโปรแกรมควบคุมการทำงานของไมโครคอนโทรเลอร์ PIC16F84A โดยการเพิ่มและแก้ไขโปรแกรมให้คำนวณค่าความจุไฟฟ้าให้แสดงค่าออกมาเป็นเปอร์เซ็นต์ ความชื้น โดยการทำงานของโปรแกรมสามารถแสดงเป็น Flow Chart ได้ดังนี้



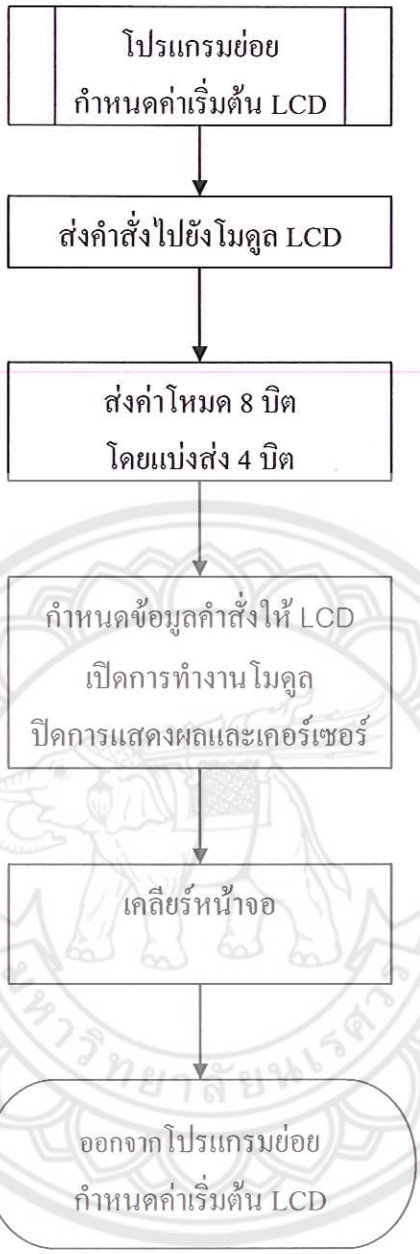
รูปที่ 3.7 แผนผังการทำงานของโปรแกรมหลัก

จากผังการทำงานของโปรแกรมหลัก เครื่องวัดความชื้นในดินทำงานโดยเริ่มต้นที่ การกำหนดค่า port ให้ไมโครคอนโทรเลอร์และ LCD ให้ทำงานในลักษณะใด IC 555 จะสร้างพัลส์ส่งไปยังไมโครคอนโทรเลอร์ให้ประมวลผลคำนวณค่า Capacitance และค่าความชื้นในดิน แล้วจึงส่งไปแสดงผลไปยัง LCD ซึ่งการทำงานของโปรแกรมจะวนไปเช่นนี้ต่อเนื่องไป เมื่อมีการวัดในครั้งต่อๆไป



รูปที่ 3.8 แผนผังโปรแกรมย่อยกำหนดค่าเริ่มต้น PORT

โปรแกรมย่อยกำหนดค่าเริ่มต้น PORT เป็นการกำหนดให้ขาของไมโครคอนโทรเลอร์ทำงานในลักษณะเป็นอินพุต หรือ เอาต์พุต ในการรับเข้าหรือส่งออกข้อมูล

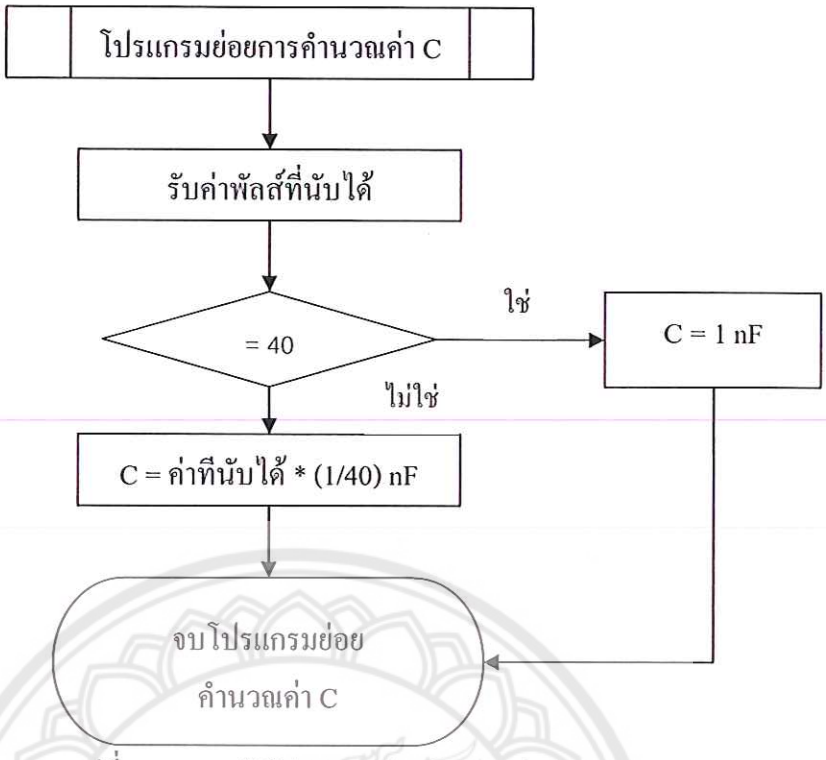


รูปที่ 3.9 แผนผังโปรแกรมย่อยกำหนดค่าเริ่มต้น LCD

โปรแกรมย่อยกำหนดค่าเริ่มต้น LCD เป็นการกำหนดค่าเริ่มต้นให้ LCD เตรียมทำงานเพื่อสื่อสารข้อมูลโดยทำงานโหมด 8 บิต แต่การเชื่อมต่อกับไมโครคอนโทรเลอร์เป็นแบบ 4 บิต จึงต้องทำการแบ่งส่งข้อมูลทีละ 4 บิต จากนั้นจึงเปิดการทำงานของ LCD และทำการเคลียร์หน้าจอเพื่อรอรับข้อมูลที่จะนำมาแสดงผลต่อไป

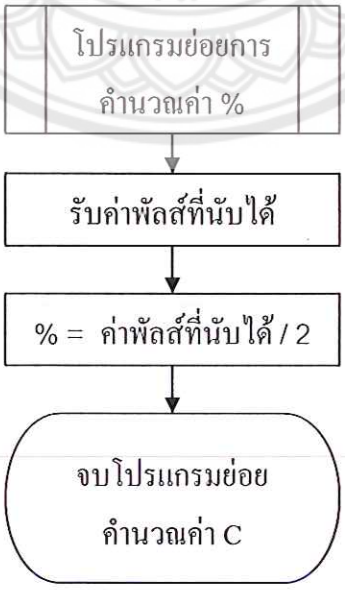
รูปที่ 3.10 แผนผังโปรแกรมย่อยการสร้างพัลส์จาก IC 555

การสร้างพัลส์จาก IC 555 ต้องทำการเซตค่าของไมโครคอนโทรเลอร์ให้อยู่ในช่วงทำการวัดนาโนฟารัด สร้างสัญญาณพัลส์จาก RC ในวงจรเมื่อได้รับสัญญาณทริกเกอร์จากไมโครคอนโทรเลอร์ และส่งพัลส์ที่สร้างขึ้นไปยังไมโครคอนโทรเลอร์เพื่อนำไปประมวลผลต่อไป



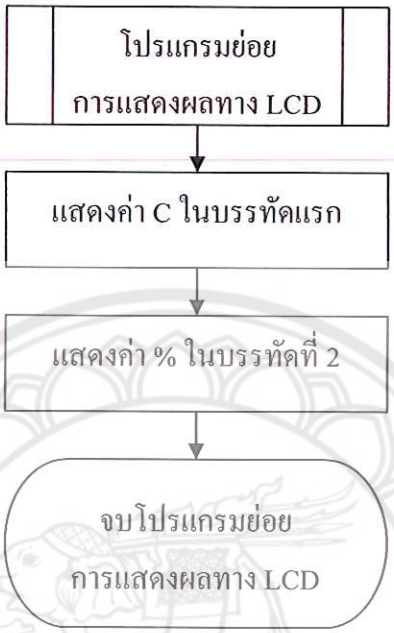
รูปที่ 3.11 แผนผังโปรแกรมย่อยคำนวณค่า C

คำนวณค่า C เนื่องจากการทดสอบพบว่าเมื่อนำค่า C=1nF มาต่อในวงจรพบว่าพัลส์ที่นับได้เท่ากับ 40 จึงกำหนดลงในโปรแกรมว่าหากไมโครคอนโทรเลอร์นับพัลส์ที่สร้างจาก IC 555 ได้ 40 จะให้ไมโครคอนโทรเลอร์แดงค่าเป็น 1 nF หากนับได้ค่าอื่นนอกเหนือจากนี้ ให้แสดงค่า C= ค่าทีนับได้คูณกับ(1/40) nF แล้วจึงนำไปแสดงผล



รูปที่ 3.12 แผนผังโปรแกรมย่อยคำนวณค่า %

ในการทดลองวัดค่าความจุไฟฟ้ากับเปอร์เซ็นต์ความชื้นพบว่า ที่ค่า 5 nF เป็นความชื้นที่ 100% ดังนั้นที่ 1nF จึงเป็นความชื้นที่20% และค่า 1 nF เป็นค่าที่ไมโครคอนโทรเลอร์นับสัญญาณพัลส์ได้ 40 ครั้งการที่จะทำให้การนับ 40 ครั้งที่ได้เป็น 20 นั่นก็คือนำค่าที่นับได้มาหารกับ 2 นั่นเอง



รูปที่ 3.13 แผนผังโปรแกรมย่อยการแสดงผลทาง LCD

การแสดงผลทางจอ LCDโดยให้แสดงค่า C ในบรรทัดแรกและแสดงค่า เปอร์เซ็นต์ความชื้นในบรรทัดที่ 2 ของจอ LCD

การดำเนินการในบทที่ 3 นี้เป็นการดำเนินการสร้างอุปกรณ์ที่ใช้ในการในการวัดความชื้นในดินทั้งหมด ตั้งแต่การสร้างกล่องภาชนะใส่ดินเพื่อการวัดความชื้น การทดลองเบื้องต้นของความสัมพันธ์ระหว่างค่าความจุไฟฟ้าของดินกับปริมาณความชื้นต่างๆ เพื่อนำมาเป็นข้อมูลในการสร้างเครื่องมือที่ทำการวัดและการปรับปรุงเครื่องมือวัดความชื้น รวมถึงการอธิบายการทำงานของเครื่องมือวัดความชื้นในดินที่สร้างขึ้น ซึ่งการทดสอบประสิทธิภาพของเครื่องมือที่ได้สร้างขึ้นนั้นจะได้นำเสนอในบทที่ 4 ซึ่งเป็นบทที่บรรยายถึงการทดสอบและการวิเคราะห์เครื่องมือวัดความชื้นในดินที่สร้างขึ้น

บทที่ 4
การทดสอบและการวิเคราะห์

การทดในบทนี้เป็นการทดสอบประสิทธิภาพของเครื่องวัดความชื้นในดินที่สร้างขึ้น โดยทำการวัดเทียบกับอุปกรณ์หลายๆชนิดดังต่อไปนี้

4.1 การทดสอบเครื่องวัดความชื้นในดินที่สร้างขึ้น

4.4.1 การทดสอบวัดค่าความจุไฟฟ้ากับตัวอย่างตัวเก็บประจุ

ทดสอบโดยนำค่า C มาตรฐานมาทำการวัดโดยเครื่องมือที่สร้างขึ้น เปรียบเทียบกับเครื่องมือวัดมาตรฐานคือ LCR Meter ซึ่งผลการทดสอบเป็นดังนี้

ตารางที่ 4.1 เปรียบเทียบค่าความจุไฟฟ้าที่วัดได้จาก Meter ที่สร้างเทียบกับ ตัวอย่างตัวเก็บประจุ

ตัวอย่างตัวเก็บประจุ	ค่าความจุไฟฟ้า (nF)	
	LCR Meter	Meter ที่สร้าง
0.5 nF	0.42	0.41
1 nF	0.918	0.87
2 nF	2.05	1.94
3.3 nF	3.37	3.35
4.7 nF	4.41	4.44

จากการทดลองการทดสอบวัดค่าความจุไฟฟ้ากับ Capacitor ตัวอย่างโดยเปรียบเทียบกับ LCR Meter พบว่าค่าที่อ่านได้จาก Meter ทั้งสอง มีค่าใกล้เคียงกัน ทำให้ทราบว่าเครื่องวัดความชื้นในดินที่สร้างขึ้นหากนำไปวัดค่าความจุไฟฟ้าแล้วมีประสิทธิภาพใกล้เคียงกับ LCR Meter

4.4.2 การทดสอบวัดค่าความชื้นในดินโดยเครื่องมือที่สร้างขึ้น

ทดสอบโดยใช้เครื่องมือที่สร้างขึ้น มาทำการวัดดินโดยเติมน้ำที่ปริมาตรต่างๆกันซึ่งคิด 100 % ความชื้นคือน้ำ 0.8 กิโลกรัม เพราะเป็นปริมาณน้ำที่เติมเข้าไปแล้วซึมออกมาจากผิวดินด้วยเหตุที่ว่า ดินที่ความชื้น 100% จะไม่สามารถรับน้ำหรือกักเก็บน้ำได้อีก ซึ่งผลการทดสอบสามารถแสดงได้ดังนี้

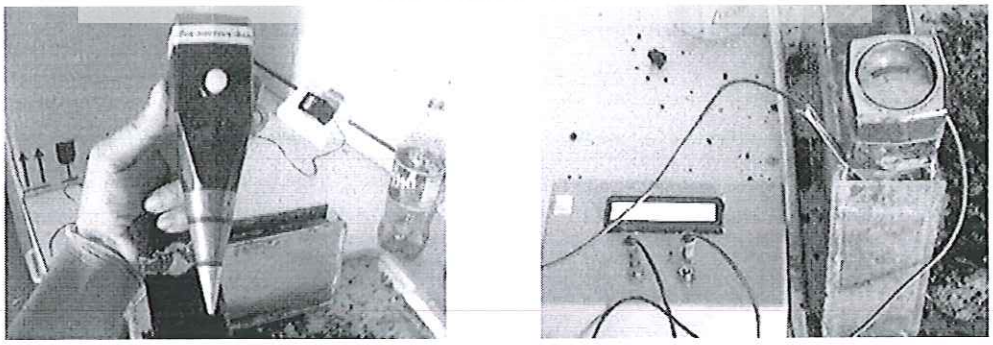
ตารางที่ 4.2 ปริมาณน้ำกับค่าความจุไฟฟ้าและเปอร์เซ็นต์ความชื้น ที่อ่านได้จากเครื่องมือที่สร้าง

ปริมาณน้ำ Kg.	ค่าความจุไฟฟ้า nF				เปอร์เซ็นต์ความชื้น %			
	1	2	3	เฉลี่ย	1	2	3	เฉลี่ย
0	0.13	0.15	0.15	0.14	2.6	3.1	3.1	2.93
0.2	0.91	0.95	0.94	0.93	18.2	19.0	18.9	18.7
0.4	2.72	2.66	2.70	2.69	54.4	53.2	54.1	53.9
0.6	3.82	3.80	3.78	3.80	76.4	76.0	75.6	76.0
0.8	4.87	4.82	4.91	4.87	97.4	96.4	98.2	97.3

จากการทดลองการทดสอบวัดค่าความชื้นในดินโดยเครื่องมือที่สร้างขึ้น พบว่าค่าเปอร์เซ็นต์ความชื้นที่ได้จากการวัด มีผลตามค่าความจุไฟฟ้า ค่าที่ได้ใกล้เคียงตามลักษณะเส้นแนวโน้มที่ได้จากการทดลองความสัมพันธ์ระหว่างค่าความชื้นกับค่าความจุไฟฟ้าในดินชนิดต่างๆ ทำให้ทราบว่าเครื่องมือวัดความชื้นในดินที่สร้างขึ้นสามารถอ้างอิงจากกราฟได้อย่างถูกต้อง

4.4.3 การทดสอบเครื่องวัดความชื้นในดินที่สร้างขึ้นเปรียบเทียบกับเครื่องวัดความชื้นแบบวัด PH

ทดสอบโดยนำเครื่องวัดความชื้นในดินที่สร้างขึ้นวัดความชื้นเปรียบเทียบกับเครื่องวัดความชื้นในดินแบบวัด PH ของภาควิชาทรัพยากรธรรมชาติและสิ่งแวดล้อม คณะเกษตรศาสตร์ ทรัพยากรธรรมชาติและสิ่งแวดล้อม มหาวิทยาลัยนเรศวร ซึ่งเป็นเครื่องวัดความชื้นภาคสนามแบบเข็มกระดิก ที่ระดับความชื้นต่างๆกัน



รูปที่ 4.1 การทดสอบเครื่องวัดความชื้นในดินที่สร้างขึ้นเปรียบเทียบกับเครื่องวัดความชื้นในดินแบบวัด PH

ตารางที่ 4.3 ทดสอบเครื่องวัดความชื้นในดินที่สร้างขึ้นเปรียบเทียบกับเครื่องวัดความชื้นแบบวัด PH

ปริมาณน้ำ Kg.	ค่าเปอร์เซ็นต์ความชื้น							
	เครื่องวัดความชื้นในดินแบบ PH				เครื่องวัดความชื้นในดินที่สร้าง			
	1	2	3	เฉลี่ย	1	2	3	เฉลี่ย
0	0	1	0	0	2	2	2	2
0.2	20.0	20.0	20.0	20.0	23.6	24.3	23.8	23.9
0.4	48.0	47.0	48.0	47.7	53.8	54.1	53.4	53.7
0.6	90	90	90	90	74.8	76.4	76.1	75.77
0.8	100.0	100.0	100.0	100.0	99.3	98.7	102.1	100.0

จากการทดลองจะเห็นว่าปริมาณน้ำที่ 0.8 กิโลกรัมเป็นปริมาณความชื้นที่ 100% เมื่อคิดเป็นสัดส่วนเปรียบเทียบปริมาณน้ำกับความชื้นแล้วค่าที่ได้จะแสดงได้ดังนี้

ปริมาณน้ำ 0 กิโลกรัม คิดเป็นความชื้น 0%

ปริมาณน้ำ 0.2 กิโลกรัม คิดเป็นความชื้น 25%

ปริมาณน้ำ 0.4 กิโลกรัม คิดเป็นความชื้น 50%

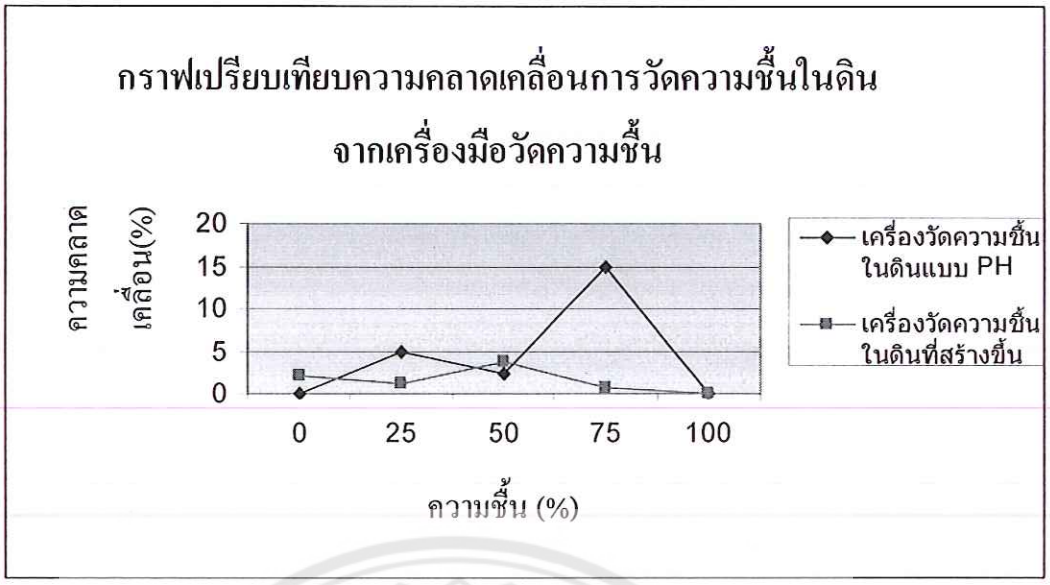
ปริมาณน้ำ 0.6 กิโลกรัม คิดเป็นความชื้น 75%

ปริมาณน้ำ 0.8 กิโลกรัม คิดเป็นความชื้น 100%

เมื่อได้สัดส่วนเปรียบเทียบปริมาณน้ำกับความชื้นแล้ว สามารถเปรียบเทียบค่าความคลาดเคลื่อนจากการทดลองทดสอบเครื่องวัดความชื้นในดินที่สร้างขึ้นเปรียบเทียบกับเครื่องวัดความชื้นแบบวัด PH ดังนี้

ตารางที่ 4.4 เปรียบเทียบค่าความคลาดเคลื่อนจากการทดลองทดสอบเครื่องวัดความชื้นในดินที่สร้างขึ้นเปรียบเทียบกับเครื่องวัดความชื้นแบบวัด PH

ปริมาณน้ำ (kg)	ความชื้น (%)	ค่าความชื้นเฉลี่ยที่อ่านได้จากเครื่องวัดความชื้นในดินแบบ PH	ค่าชื้นความคลาดเคลื่อน	ค่าความชื้นเฉลี่ยที่อ่านได้จากเครื่องวัดความชื้นในดินที่สร้าง	ค่าชื้นความคลาดเคลื่อน
0	0	0	0	2	2
0.2	25	20.0	5	23.9	1.1
0.4	50	47.7	2.3	53.7	3.7
0.6	75	90	15	75.7	0.7
0.8	100	100.0	0	100.0	0



รูปที่ 4.2 กราฟเปรียบเทียบความคลาดเคลื่อนการวัดความชื้นในดินจากเครื่องมือวัดความชื้น

จากการทดลองทดสอบเครื่องวัดความชื้นในดินที่สร้างขึ้น เปรียบเทียบกับเครื่องวัดความชื้นแบบวัด PH พบว่าของเครื่องวัดความชื้นในดินแบบ PH เป็นเครื่องวัดแบบเข็มกระดิก สเกลค่อนข้างหยาบ วัดจาก 0 – 80 % โดยนับทีละ 10% ซึ่งหากค่าเกิน 80% เข็มจะเกินสเกลไปชี้ที่อักษร wet การอ่านต้องใช้การประมาณ แต่ค่าที่อ่านได้มีค่าแตกต่างจากเครื่องวัดความชื้นที่สร้างขึ้นไม่มากนัก เมื่อพิจารณาถึงกราฟความคลาดเคลื่อนของเครื่องวัดความชื้นที่สร้างขึ้นและเครื่องวัดความชื้นแบบวัด PH พบว่าความผิดพลาดของเครื่องวัดความชื้นที่สร้างขึ้น จะผิดพลาดน้อยกว่าคิดเป็นคลาดเคลื่อนไม่เกิน ± 5% ความชื้น ส่วนเครื่องวัดความชื้นแบบ PH ค่าความคลาดเคลื่อนจะโดยเฉพาะช่วงตั้งแต่ 50% ความชื้นจะผิดพลาดมาก สาเหตุส่วนหนึ่งน่าจะมาจากการอ่านสเกลที่เข็มชี้เกิน 80% ซึ่งต้องใช้การประมาณค่ามากกว่าการอ่านที่สเกลอื่น

จากการทดสอบประสิทธิภาพของเครื่องมือวัดที่สร้างขึ้น ด้วยวิธีการทดสอบเทียบการวัดค่าตัวเก็บประจุตัวอย่าง การนำไปวัดความชื้นในดินโดยนำค่าที่ได้เปรียบเทียบกับข้อมูลในบทที่ 3 และการทดสอบวัดความชื้นในดินเทียบกับเครื่องวัดความชื้นในดินแบบวัดPH ทำให้พบว่าประสิทธิภาพของเครื่องมือที่สร้างขึ้นนั้นค่อนข้างดี แต่ก็มีความผิดพลาดและข้อเสียบางประการซึ่งจะได้นำเสนอในบทสรุปและข้อเสนอแนะ

บทที่ 5
สรุปผลและข้อเสนอแนะ

เครื่องวัดความชื้นในดินที่สร้างขึ้น เป็นการนำค่าความจุไฟฟ้ามาคิดเป็นค่าความชื้นในดิน โดยอาศัยการทดลองหาความสัมพันธ์ระหว่างค่าความจุไฟฟ้าที่เปลี่ยนไป เมื่อค่าความชื้นในดินเปลี่ยนไปแล้วนำค่าที่ได้มาทำการสร้างกราฟ เพื่อหาลักษณะเส้นแนวโน้มที่เกิดขึ้น ซึ่งจากเส้นแนวโน้มความสัมพันธ์ระหว่างค่าความจุไฟฟ้าและปริมาณน้ำที่ได้เติมเข้าจะถูกนำไปคิดเป็นเปอร์เซ็นต์ความชื้นของดิน และจากข้อมูลที่ได้ก็ถูกนำไปเป็นข้อมูลในการอ้างอิงในการสร้างเครื่องมือวัดความชื้นในดิน

สำหรับการทดสอบเครื่องมือวัดความชื้นในดินที่สร้างขึ้น ได้นำไปเปรียบเทียบกับเครื่องวัดความชื้นในดินแบบ PH ผลการทดสอบที่ได้ค่าความชื้นจะไม่แตกต่างกันมาก โดยลักษณะความชื้นที่ได้จากเครื่องวัดความชื้นที่สร้างขึ้นจะเป็นสัดส่วนที่แน่นอนกว่าเครื่องวัดความชื้นแบบ PH ความแตกต่างของความชื้นที่เกิดขึ้นอาจเป็นเพราะเป็นการวัดความชื้นคนละรูปแบบ สเกลการอ่านค่าก็แตกต่างกันด้วย เมื่อพิจารณาที่ความผิดพลาดพบว่าเครื่องมือวัดความชื้นในดินที่สร้างขึ้นมีความผิดพลาดน้อยกว่าเครื่องวัดความชื้นแบบ PH

การวัดความชื้นในดินเครื่องวัดความชื้นในดินที่สร้างขึ้นต้องขุดดินที่ต้องการวัดมาใส่ในภาชนะ ซึ่งเป็นข้อเสียของเครื่องวัดความชื้นในดินที่สร้างขึ้น หากสามารถปรับปรุงเครื่องวัดความชื้นในดินให้สามารถวัดความชื้นโดยวัดลงไปที่ดินโดยไม่ต้องนำมาใส่ในภาชนะได้ จะทำให้เครื่องมือวัดความชื้นในดินมีประสิทธิภาพมากขึ้น และการวัดแต่ละครั้งต้องทำการรีเซตเครื่องทุกครั้งเพื่อ โดยการถอดสายที่ทำการวัดออกจากเครื่องก่อนทำให้ไม่สะดวก หากทำการเพิ่ม รีเซตสวิตส์ จะทำให้ทำการวัดได้สะดวกมากขึ้น

เอกสารอ้างอิง

[1] ดอนสัน ปงผาบ. **การเขียนโปรแกรมภาษา ซี ในงานควบคุม.** กรุงเทพมหานคร : สมาคมส่งเสริมเทคโนโลยี (ไทย-ญี่ปุ่น). 2546.

[2] กฤษฎา ใจเย็น. **สนุกกับไมโครคอนโทรเลอร์ ฉบับ PIC.** กรุงเทพมหานคร. 2545.

[3] สราวุธ จริตงาม. **กลศาสตร์ของดิน (Soil Mechanic).** สงขลา : ภาควิชาวิศวกรรมโยธา มหาวิทยาลัยสงขลานครินทร์ วิทยาเขตหาดใหญ่. 2542.

[4] ธงชัย พึ่งรัศมี. **ธรณีวิทยาทั่วไป (General Geology).** กรุงเทพมหานคร. 2531.

[5] เกษมศรี ซับซ้อน. **ปฐพีวิทยา (Soil Science).** กรุงเทพมหานคร. 2541.

[6] Microchip Technology Inc. "Pic16F84A Data Sheet." [Online]. Available : Http://www.microchip.com. 2001.

[7] Thai MCU. "ไมโครคอนโทรเลอร์ Pic16F84." [Online]. Available : Http://www.thaimcu.com/article/getstart/pic16f84_hw.htm.

[8] RTC. "ICเบอร์ 555 monostable." [Online]. Available : http://www.rtc.ac.th/itc/tpanya/unit10/data/informate.html#7

[9] มานพ เกตุชีพ. **"เครื่องวัดค่าความจุไฟฟ้ารุ่นเปลี่ยนย่านวัดความจุไฟฟ้าอัตโนมัติ".** นิตยสารเซมิคอนดักเตอร์. ฉบับที่257. ธันวาคม 2546. หน้า 153.

[10] WILLIAM H.HAYT,JR. **Engineering Electromagnetic.** International Edition. Times Roman : McGraw-Hill, Inc. 1981.

ภาคผนวก ก

Source Code ของโปรแกรม

```
/*************************************************/
/*** C metering                                ***/
/*** Auther: R&D02                             ***/
/*************************************************/
#include        <pic.h>

        __CONFIG(XT&WDTDIS);

#define PORTBIT(adr, bit)       ((unsigned)(&adr)*8+(bit))

//*** LCD Pin definition
static bit      LCD_RW      @       PORTBIT(PORTA, 2);          // LCD : Read/Write
        low forever
static bit      LCD_EN      @       PORTBIT(PORTA, 1);          // LCD : Enable
        clk
static bit      LCD_RS      @       PORTBIT(PORTA, 0);          // LCD : RS
        instruction/data
static bit      D4      @       PORTBIT(PORTB, 4);          // LCD : Data bit 4
static bit      D5      @       PORTBIT(PORTB, 5);          // LCD : Data bit 5
static bit      D6      @       PORTBIT(PORTB, 6);          // LCD : Data bit 6
static bit      D7      @       PORTBIT(PORTB, 7);          // LCD : Data bit 7

//*** IC 555 Pin definition
static bit      TRIG    @       PORTBIT(PORTA, 3);          // Trig
static bit      PULSE   @       PORTBIT(PORTA, 4);          // Pulse
static bit      RESET555 @      PORTBIT(PORTB, 3);          // Reset
static bit      RANGE_u     @       PORTBIT(PORTB, 1);          // Select range u (micro)
static bit      RANGE_n     @       PORTBIT(PORTB, 2);          // Select range n (nano)
```

```
/***************** LCD E stobe *****************/
void LCD_STROBE()
{
//*** Create LCD : Enable High to Low
//***   _____
//***       |_____
//***
//*** For LCD activate
        unsigned char i;
        LCD_EN=1;                                   // LCD : Enable Set to High
        for(i = 5 ; --i ;)                          // Delay for High
        continue;
        LCD_EN=0;                                   // LCD : Enable Clear to Low
        for(i = 5 ; --i ;)                          // Delay for Low
        continue;
}

/*********************** LCD Delay *****************/
void Delay_LCD(void)
{
//*** LCD delay. When LCD received the instruction or data it will use the time to execute this
        instruction
//*** You will wait for execution time, Delay_LCD
        unsigned char i,j;
        for(i = 30 ; --i ;)                         // Outer loop delay
        for(j = 12 ; --j ;)                         // Inner loop delay
                continue;
}

/***************** LCD write *****************/
void lcd_write(unsigned char c)
```

```
{
//*** Sent data bus 8 bit to LCD 4 bit
//*** LCD is in 4 bits mode but data is 8 bit,we will seperate 8 bit to 4 bits upper and 4 bit lower
//*** data = b' xxxx xxxx '
//*** The data will be sent 4 bit(nibble) upper frist                    data='xxxx  0000'
//*** and then sent 4 bit lower                                                  data='0000  xxxx  '
//*** Example
//*** lcd_write(0xAF)
//*** data = b' 1010 1111'
        PORTB = (PORTB & 0x0F) | (c & 0xF0);                    // Port B = [Port B  and 0x0F] or
        [0xAF and 0xF0]
                                                //                ^                ^ Mask 4 bit upper
                                                //                                 ^ Clear Port B bit 4,5,6,7


                                                // Port B = [b'0000 xxxx'] or [b'1010 0000']
                                                // Port B =  b'1010 xxxx'
                                                // Sent data 4 bit upper to Portb B bit 4,5,6,7
        LCD_STROBE();                                           // Sent LCD Stobe for the 4 bit
        upper data execution.

        PORTB = (PORTB & 0x0F) | (c << 4);    // Port B = [Port B  and 0x0F] or [0xAF rotate left
        4 time]
                                                // 0xAF rotate left 4 time
                                                // 0xAF =       b'1010 1111'
                                                // rotate 1st       b'0101 1110'
                                                // rotate 2nd       b'1011 1100'
                                                // rotate 3rd       b'0111 1000'
                                                // rotate 4th       b'1111 0000'
                                                // Port B = [b'0000 xxxx'] or [b'1111 0000']
                                                // Port B =  b'1111 xxxx'
                                                // Sent data 4 bit lower to Portb B bit 4,5,6,7
```

```
        LCD_STROBE();                                   // Sent LCD Stobe for the 4 bit
        lower data execution.
        Delay_LCD();                                    // Wait for data execution
}

/********** LCD Write Charcter at position ***********/
void lcd_write_at(unsigned char pos,unsigned char ch)
{
//*** For write character to LCD address
//*** LCD address (16*2)
//*** First Line :
//*** [0x80] [0x81] [0x82] [0x83] [0x84] [0x85] [0x86] [0x87] [0x88] [0x89] [0x8A] [0x8B] [0x8C]
        [0x8D] [0x8E] [0x8F]
//*** Second Line :
//*** [0xC0] [0xC1] [0xC2] [0xC3] [0xC4] [0xC5] [0xC6] [0xC7] [0xC8] [0xC9] [0xCA] [0xCB]
        [0xCC] [0xCD] [0xCE] [0xCF]

        LCD_RS = 0;                                     // Set LCD mode to Instruction mode
        lcd_write(pos);                                 // Sent address to LCD
        LCD_RS = 1;                                     // Set LCD mode to Data mode
        lcd_write(ch);                                  // Sent character to LCD
//*** Example
//*** lcd_write_at(0x87,'A')                            // Write character 'A' to LCD address 0x87
//*** lcd_write_at(0xC3,'F')                            // Write character 'F' to LCD address 0xC3
//*** [ ] [ ] [ ] [ ] [ ] [ ] [ ] [A] [ ] [ ] [ ] [ ] [ ] [ ] [ ] [ ]
//*** [ ] [ ] [ ] [F] [ ] [ ] [ ] [ ] [ ] [ ] [ ] [ ] [ ] [ ] [ ] [ ]
}

/***************** Initialize LCD *****************/
void init_lcd()
{
//*** For initialize LCD to 4 bit mode, 1/16 duty, not blink cursor and clear LCD memory
```

```
        LCD_RS = 0;                         // Set LCD mode to Instruction mode
        lcd_write(0x28);                    // 4 bit mode, 1/16 duty, 5x8 font
        lcd_write(0x08);                    // display off
        lcd_write(0x0C);                    // display on, blink cursor off
        lcd_write(0x06);                    // entry mode
        lcd_write(0x01);                    // clear
}

void LCD_cls(void)
{
//*** For clear LCD display
        LCD_RS = 0;                         // Set LCD mode to Data mode
        lcd_write(0x01);                    // Sent command clear LCD screen
}
/****** Display Percent in second line ******/
void disp_percent(unsigned char num)
{
//*** For display percent number
//*** The number cannot direct sent to LCD because LCD will display data that be only character
        value,
//*** see the ASCII table
        unsigned char t;
        LCD_RS = 0;                         // Set LCD mode to Instruction mode
        lcd_write(0xC0);                    // LCD address 0xC0
        LCD_RS = 1;                         // Set LCD mode to Data mode
        for(t=15;--t;)
        lcd_write(0x20);            // Sent character ' ' (blank) to LCD second line
        lcd_write_at(0xc0,'H');             // write character 'C'to address 0x80
        lcd_write_at(0xc1,'M');             // write character 'x'to address 0x81
        LCD_RS = 0;                         // Set LCD mode to Instruction mode
        lcd_write(0xC7);                    // LCD address 0xC7
```

```
LCD_RS = 1;                                  // Set LCD mode to Data mode
if (num<10)                                  // If number < 10
lcd_write((num+0x30)*2);                     // Write character '0','1','2',...,'9'
                                             // number = 1 convert to character '1' = number +
0x30
                                             // character '0' = 0x30
                                             // character '1' = 0x31
                                             // character '2' = 0x32
                                             // character '3' = 0x33
                                             // character '4' = 0x34
                                             // character '5' = 0x35
                                             // character '6' = 0x36
                                             // character '7' = 0x37
                                             // character '8' = 0x38
                                             // character '9' = 0x39
else if (num<100)                            // if number <100
{                                            // number : 10,11,12,13, ... ,98,99
t=num/10;                                    // t = number / 10
lcd_write(((t)+0x30)*2);                     // Display first digit
lcd_write(((num-(t*10))+0x30)*2);            // Display second digit
}
else                                         // number > 100
{
t=num/100;
lcd_write((t+0x30)*2);
tt=(dec-(t*100))/10;
lcd_write((tt+0x30)*2);
lcd_write(((num-((t*100)+(tt*10)))+0x30)*2);
}
lcd_write(0x25);                             // Write character '%'
}
```

```
/***************** Display C Value *****************/
/*** format xx.xx  nF                          ***/
void disp_value(unsigned char dec,unsigned char dig,unsigned char unit)
{
//*** For Display C value
        unsigned char t,tt;
        LCD_RS = 0;                                     // Set LCD mode to Instruction
        mode
        lcd_write(0x80);                                // LCD address 0x80
        LCD_RS = 1;                                     // Set LCD mode to Data mode
        for(t=15;--t;)
        lcd_write(0x20);                        // Sent character ' ' (blank) to LCD first line
        lcd_write_at(0x80,'C');                         // write character 'C'to address 0x80
        lcd_write_at(0x81,'x');                         // write character 'x'to address 0x81

        LCD_RS = 0;                                     // Set LCD mode to Instruction
        mode
        lcd_write(0x86);                                // LCD address 0x86
        LCD_RS = 1;                                     // Set LCD mode to Data mode
        if (dec<10)                                     // If C value < 10
                                                        // Value : 0, 1, 2, ...,8 ,9
        lcd_write(dec+0x30);                            // Write first digit
        else if (dec<100)                               // If C value < 100
        {
        t=dec/10;
        lcd_write((t)+0x30);                            // Write first digit
        lcd_write((dec-(t*10))+0x30);                   // Write second digit
        }
        else                                            // If C valu > 100
        {
        t=dec/100;
        lcd_write(t+0x30);
```

```
        tt=(dec-(t*100))/10;
        lcd_write(tt+0x30);
        lcd_write((dec-((t*100)+(tt*10)))+0x30);

        }
        lcd_write(0x2E);                    // Write character '.' (point)
        lcd_write(dig+0x30);
        lcd_write(0x20);                    // Write character ' ' (blank)
        lcd_write('n');                     // Write unit  'n'
        lcd_write('F');                     // Write character 'F'
}

/********** Initialize Hardware ***************/
void init()
{
//*** For initialize microcontroller
        TRISA=0x10;                         // Port A Direction
        TRISB=0x00;                         // Port B Direction
        PORTB=0x00;                         // Clear Port B
        PORTA=0x00;                         // Clear Port A
        LCD_RW=0;                           // LCD Write mode
}
void Delay1(void)
{
        unsigned char i, j, k;
        for (i=80; --i;)
        for(j=50; --j;)
                for(k=15; --k;)
                        continue;
}

void trigger(void)
```

```
{
//*** For trig IC555 for one short
        unsigned char i;
        TRIG=1;
        TRIG=0;                                                 // TRIG
        for(i=10; --i;)                                 //  ____    ____
        continue;                                               //  |_____|
        TRIG=1;
}

void delay_count1(void)
{
        unsigned char c1;
        for(c1=78; --c1;)
        continue;
}

void delay_count2(void)
{
        unsigned char c1;
        for(c1=25; --c1;)
        continue;
}

void delay_count_nano(void)
{
        unsigned char c1;
        for(c1=78; --c1;)               //78
        continue;
}

void delay_count_pico(void)
```

```
{
        unsigned char c1;
        for(c1=5; --c1;)
        continue;
}
/***************** Main Program *****************/
main(void)
{
        unsigned char count1, temp, perc;
        unsigned char range_pico, range_nano;
        init();                                         // Init. Hardware
        for(temp=10;--temp;)
        {
        init_lcd();                                     // Init. LCD
        LCD_EN=0;
        LCD_RS = 0;                                     // goto position
        lcd_write(0x80);                        // Clear line
        LCD_RS = 1;                                     // write character
        lcd_write(0x20);
        }

        range_nano=0;                                   // goto range nano
        for(;;)
        {
/******************* nano range ******************/
nano_range:
        RESET555=0;                             // Reset TLC555
        TRIG=1;                                         // Trig for discharge
        TRISB2=1;
        RANGE_n=1;                              // Enable nano range
        RANGE_u=1;                              // Disable micro range
        disp_value(0,0,'n');                    // Display C value
```

```
        disp_percent(00);                       // Display percent
        while(range_nano==1)
        {
                RESET555=0;                     // Reset TLC555
                Delay1();
                Delay1();
                count1=0;                       // 0 -255
                RESET555=1;                     // Enable TLC555
                trigger();                      // trig
                delay_count2();                 // Delay for a single pulse out
                while(PULSE==1)                         // Count single pulse width
                {
                        ++count1;               // count = count+1
                        delay_count_nano();
                }
                        temp=count1;
                        disp_value(temp,0,'n');         // Display C value
                        disp_percent(0);        // Display percent
                }
        }
/**************** End of pico range **************/

        }

}
```

ภาคผนวก ข

ข้อมูล PIC16F84A

High Performance RISC CPU Features:

- Only 35 single word instructions to learn
- All instructions single-cycle except for program branches which are two-cycle
- Operating speed: DC - 20 MHz clock input
  DC - 200 ns instruction cycle
- 1024 words of program memory
- 68 bytes of Data RAM
- 64 bytes of Data EEPROM
- 14-bit wide instruction words
- 8-bit wide data bytes
- 15 Special Function Hardware registers
- Eight-level deep hardware stack
- Direct, indirect and relative addressing modes
- Four interrupt sources:
  - External RB0/INT pin
  - TMR0 timer overflow
  - PORTB<7:4> interrupt-on-change
  - Data EEPROM write complete

Peripheral Features:

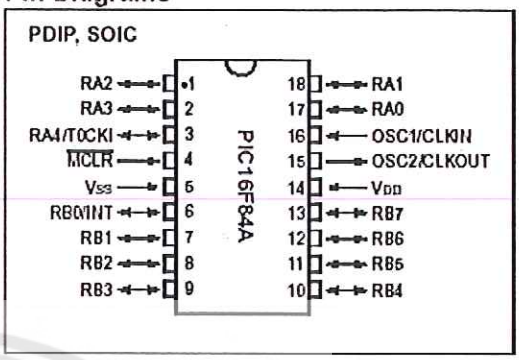
- 13 I/O pins with individual direction control
- High current sink/source for direct LED drive
  - 25 mA sink max. per pin
  - 25 mA source max. per pin
- TMR0: 8-bit timer/counter with 8-bit programmable prescaler

Special Microcontroller Features:

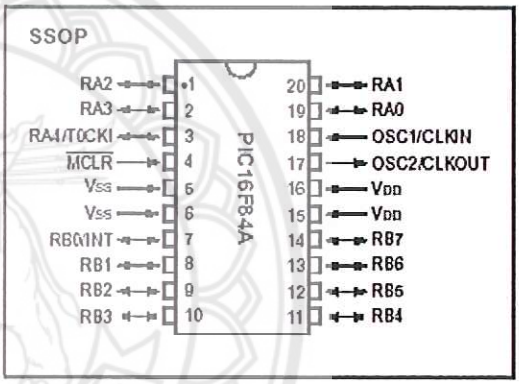
- 10,000 erase/write cycles *Enhanced* FLASH Program memory typical
- 10,000,000 typical erase/write cycles EEPROM Data memory typical
- EEPROM Data Retention > 40 years
- In-Circuit Serial Programming™ (ICSP™) - via two pins
- Power-on Reset (POR), Power-up Timer (PWRT), Oscillator Start-up Timer (OST)
- Watchdog Timer (WDT) with its own On-Chip RC Oscillator for reliable operation
- Code protection
- Power saving SLEEP mode
- Selectable oscillator options

Pin Diagrams

PDIP, SOIC



SSOP



CMOS Enhanced FLASH/EEPROM Technology:

- Low power, high speed technology
- Fully static design
- Wide operating voltage range:
  - Commercial: 2.0V to 5.5V
  - Industrial: 2.0V to 5.5V
- Low power consumption:
  - < 2 mA typical @ 5V, 4 MHz
  - 15 µA typical @ 2V, 32 kHz
  - < 0.5 µA typical standby current @ 2V

1.0 DEVICE OVERVIEW

This document contains device specific information for the operation of the PIC16F84A device. Additional information may be found in the PICmicro™ Mid-Range Reference Manual, (DS33023), which may be downloaded from the Microchip website. The Reference Manual should be considered a complementary document to this data sheet, and is highly recommended reading for a better understanding of the device architecture and operation of the peripheral modules.

The PIC16F84A belongs to the mid-range family of the PICmicro® microcontroller devices. A block diagram of the device is shown in Figure 1-1.

The program memory contains 1K words, which translates to 1024 instructions, since each 14-bit program memory word is the same width as each device instruction. The data memory (RAM) contains 68 bytes. Data EEPROM is 64 bytes.

There are also 13 I/O pins that are user-configured on a pin-to-pin basis. Some pins are multiplexed with other device functions. These functions include:

- External interrupt
- Change on PORTB interrupt
- Timer0 clock input

Table 1-1 details the pinout of the device with descriptions and details for each pin.

FIGURE 1-1: PIC16F84A BLOCK DIAGRAM

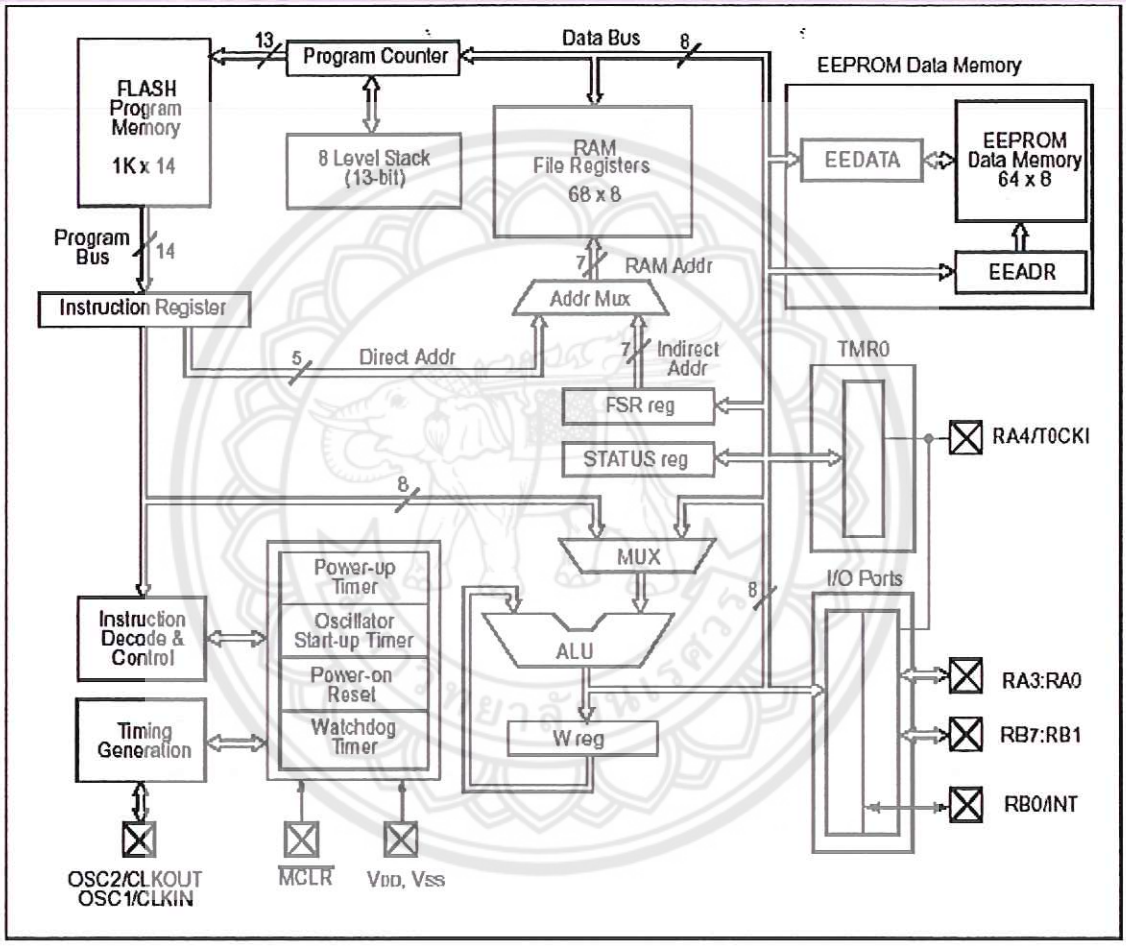


TABLE 1-1: PIC16F84A PINOUT DESCRIPTION

Pin Name	PDIP No.	SOIC No.	SSOP No.	I/O/P Type	Buffer Type	Description
OSC1/CLKIN	16	16	18	I	ST/CMOS(3)	Oscillator crystal input/external clock source input.
OSC2/CLKOUT	15	15	19	O	—	Oscillator crystal output. Connects to crystal or resonator in Crystal Oscillator mode. In RC mode, OSC2 pin outputs CLKOUT, which has 1/4 the frequency of OSC1 and denotes the instruction cycle rate.
$\overline{MCLR}$	4	4	4	I/P	ST	Master Clear (Reset) input/programming voltage input. This pin is an active low RESET to the device.
						PORTA is a bi-directional I/O port.
RA0	17	17	19	I/O	TTL	
RA1	18	18	20	I/O	TTL	
RA2	1	1	1	I/O	TTL	
RA3	2	2	2	I/O	TTL	
RA4/T0CKI	3	3	3	I/O	ST	Can also be selected to be the clock input to the TMR0 timer/counter. Output is open drain type.
						PORTB is a bi-directional I/O port. PORTB can be software programmed for internal weak pull-up on all inputs.
RB0/INT	6	6	7	I/O	TTL/ST(1)	RB0/INT can also be selected as an external interrupt pin.
RB1	7	7	8	I/O	TTL	
RB2	8	8	9	I/O	TTL	
RB3	9	9	10	I/O	TTL	
RB4	10	10	11	I/O	TTL	Interrupt-on-change pin.
RB5	11	11	12	I/O	TTL	Interrupt-on-change pin.
RB6	12	12	13	I/O	TTL/ST(2)	Interrupt-on-change pin. Serial programming clock.
RB7	13	13	14	I/O	TTL/ST(2)	Interrupt-on-change pin. Serial programming data.
VSS	5	5	5,6	P	—	Ground reference for logic and I/O pins.
VDD	14	14	15,16	P	—	Positive supply for logic and I/O pins.

Legend: I= input O = Output I/O = Input/Output P = Power
— = Not used TTL = TTL input ST = Schmitt Trigger input

Note 1: This buffer is a Schmitt Trigger input when configured as the external interrupt.
2: This buffer is a Schmitt Trigger input when used in Serial Programming mode.
3: This buffer is a Schmitt Trigger input when configured in RC oscillator mode and a CMOS input otherwise.

2.0 MEMORY ORGANIZATION

There are two memory blocks in the PIC16F84A. These are the program memory and the data memory. Each block has its own bus, so that access to each block can occur during the same oscillator cycle.

The data memory can further be broken down into the general purpose RAM and the Special Function Registers (SFRs). The operation of the SFRs that control the "core" are described here. The SFRs used to control the peripheral modules are described in the section discussing each individual peripheral module.

The data memory area also contains the data EEPROM memory. This memory is not directly mapped into the data memory, but is indirectly mapped. That is, an indirect address pointer specifies the address of the data EEPROM memory to read/write. The 64 bytes of data EEPROM memory have the address range 0h-3Fh. More details on the EEPROM memory can be found in Section 3.0.

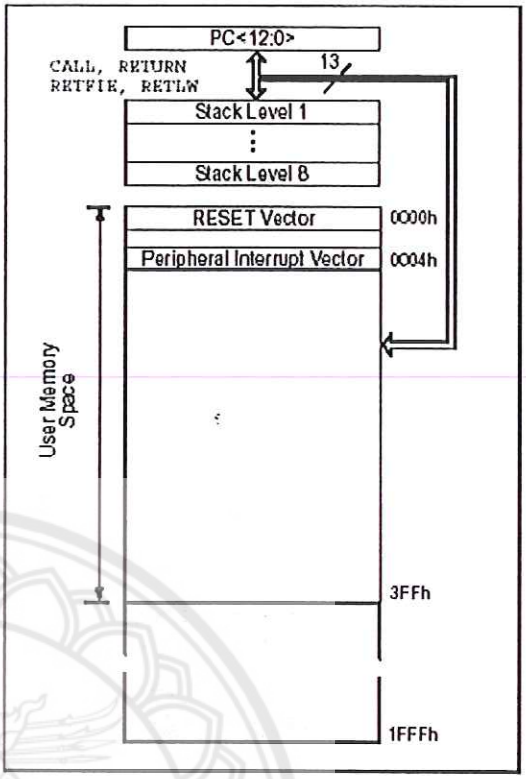
Additional information on device memory may be found in the PICmicro™ Mid-Range Reference Manual, (DS33023).

2.1 Program Memory Organization

The PIC16FXX has a 13-bit program counter capable of addressing an 8K x 14 program memory space. For the PIC16F84A, the first 1K x 14 (0000h-03FFh) are physically implemented (Figure 2-1). Accessing a location above the physically implemented address will cause a wraparound. For example, for locations 20h, 420h, 820h, C20h, 1020h, 1420h, 1820h, and 1C20h, the instruction will be the same.

The RESET vector is at 0000h and the interrupt vector is at 0004h.

FIGURE 2-1: PROGRAM MEMORY MAP AND STACK - PIC16F84A



2.2 Data Memory Organization

The data memory is partitioned into two areas. The first is the Special Function Registers (SFR) area, while the second is the General Purpose Registers (GPR) area. The SFRs control the operation of the device.

Portions of data memory are banked. This is for both the SFR area and the GPR area. The GPR area is banked to allow greater than 116 bytes of general purpose RAM. The banked areas of the SFR are for the registers that control the peripheral functions. Banking requires the use of control bits for bank selection. These control bits are located in the STATUS Register. Figure 2-2 shows the data memory map organization.

Instructions MOVWF and MOVF can move values from the W register to any location in the register file ("F"), and vice-versa.

The entire data memory can be accessed either directly using the absolute address of each register file or indirectly through the File Select Register (FSR) (Section 2.5). Indirect addressing uses the present value of the RP0 bit for access into the banked areas of data memory.

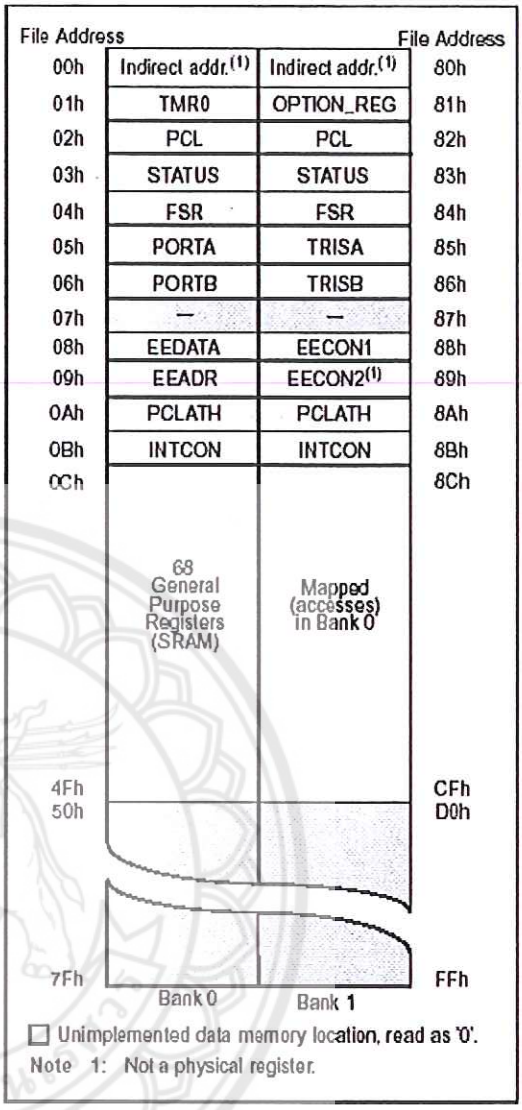
Data memory is partitioned into two banks which contain the general purpose registers and the special function registers. Bank 0 is selected by clearing the RP0 bit (STATUS<5>). Setting the RP0 bit selects Bank 1. Each Bank extends up to 7Fh (128 bytes). The first twelve locations of each Bank are reserved for the Special Function Registers. The remainder are General Purpose Registers, implemented as static RAM.

2.2.1 GENERAL PURPOSE REGISTER FILE

Each General Purpose Register (GPR) is 8-bits wide and is accessed either directly or indirectly through the FSR (Section 2.5).

The GPR addresses in Bank 1 are mapped to addresses in Bank 0. As an example, addressing location 0Ch or 8Ch will access the same GPR.

FIGURE 2-2: REGISTER FILE MAP - PIC16F84A



2.3 Special Function Registers

The Special Function Registers (Figure 2-2 and Table 2-1) are used by the CPU and Peripheral functions to control the device operation. These registers are static RAM.

The special function registers can be classified into two sets, core and peripheral. Those associated with the core functions are described in this section. Those related to the operation of the peripheral features are described in the section for that specific feature.

TABLE 2-1: SPECIAL FUNCTION REGISTER FILE SUMMARY

Addr	Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Value on Power-on RESET	Details on page
Bank 0											
00h	INDF	Uses contents of FSR to address Data Memory (not a physical register)								---- ----	11
01h	TMR0	8-bit Real-Time Clock/Counter								xxxx xxxx	20
02h	PCL	Low Order 8 bits of the Program Counter (PC)								0000 0000	11
03h	STATUS(2)	IRP	RP1	RP0	$\overline{TO}$	$\overline{PD}$	Z	DC	C	0001 1xxx	8
04h	FSR	Indirect Data Memory Address Pointer 0								xxxx xxxx	11
05h	PORTA(4)	—	—	—	RA4/T0CKI	RA3	RA2	RA1	RA0	---x xxxx	16
06h	PORTB(5)	RB7	RB6	RB5	RB4	RB3	RB2	RB1	RB0/INT	xxxx xxxx	18
07h	—	Unimplemented location, read as '0'								—	—
08h	EEDATA	EEPROM Data Register								xxxx xxxx	13,14
09h	EEADR	EEPROM Address Register								xxxx xxxx	13,14
0Ah	PCLATH	—	—	—	Write Buffer for upper 5 bits of the PC(1)					---0 0000	11
0Bh	INTCON	GIE	EEIE	T0IE	INTE	RBIE	T0IF	INTF	RBIF	0000 000x	10
Bank 1											
80h	INDF	Uses Contents of FSR to address Data Memory (not a physical register)								---- ----	11
81h	OPTION_REG	RBPU	INTEDG	T0CS	T0SE	PSA	PS2	PS1	PS0	1111 1111	9
82h	PCL	Low order 8 bits of Program Counter (PC)								0000 0000	11
83h	STATUS (2)	IRP	RP1	RP0	$\overline{TO}$	$\overline{PD}$	Z	DC	C	0001 1xxx	8
84h	FSR	Indirect data memory address pointer 0								xxxx xxxx	11
85h	TRISA	—	—	—	PORTA Data Direction Register					---1 1111	16
86h	TRISB	PORTB Data Direction Register								1111 1111	18
87h	—	Unimplemented location, read as '0'								—	—
88h	EECON1	—	—	—	EEIF	WRERR	WREN	WR	RD	---0 x000	13
89h	EECON2	EEPROM Control Register 2 (not a physical register)								---- ----	14
0Ah	PCLATH	—	—	—	Write buffer for upper 5 bits of the PC(1)					---0 0000	11
0Bh	INTCON	GIE	EEIE	T0IE	INTE	RBIE	T0IF	INTF	RBIF	0000 000x	10

Legend: x = unknown, u = unchanged, - = unimplemented, read as '0', q = value depends on condition

Note 1: The upper byte of the program counter is not directly accessible. PCLATH is a slave register for PC<12:8>. The contents of PCLATH can be transferred to the upper byte of the program counter, but the contents of PC<12:8> are never transferred to PCLATH.

2: The $\overline{TO}$ and $\overline{PD}$ status bits in the STATUS register are not affected by a $\overline{MCLR}$ Reset.

3: Other (non power-up) RESETS include: external RESET through $\overline{MCLR}$ and the Watchdog Timer Reset.

4: On any device RESET, these pins are configured as inputs.

5: This is the value that will be in the port output latch.

2.3.1 STATUS REGISTER

The STATUS register contains the arithmetic status of the ALU, the RESET status and the bank select bit for data memory.

As with any register, the STATUS register can be the destination for any instruction. If the STATUS register is the destination for an instruction that affects the Z, DC or C bits, then the write to these three bits is disabled. These bits are set or cleared according to device logic. Furthermore, the $\overline{TO}$ and $\overline{PD}$ bits are not writable. Therefore, the result of an instruction with the STATUS register as destination may be different than intended.

For example, `CLRF STATUS` will clear the upper three bits and set the Z bit. This leaves the STATUS register as `000u uluu` (where u = unchanged).

Only the `BCF`, `BSF`, `SWAPF` and `MOVWF` instructions should be used to alter the STATUS register (Table 7-2), because these instructions do not affect any status bit.

Note 1: The IRP and RP1 bits (STATUS<7:6>) are not used by the PIC16F84A and should be programmed as cleared. Use of these bits as general purpose R/W bits is NOT recommended, since this may affect upward compatibility with future products.

2: The C and DC bits operate as a $\overline{borrow}$ and digit $\overline{borrow}$ out bit, respectively, in subtraction. See the `SUBLW` and `SUBWF` instructions for examples.

3: When the STATUS register is the destination for an instruction that affects the Z, DC or C bits, then the write to these three bits is disabled. The specified bit(s) will be updated according to device logic

REGISTER 2-1: STATUS REGISTER (ADDRESS 03h, 83h)

R/W-0	R/W-0	R/W-0	R-1	R-1	R/W-x	R/W-x	R/W-x
IRP	RP1	RP0	$\overline{TO}$	$\overline{PD}$	Z	DC	C
bit 7							bit 0

bit 7-6 Unimplemented: Maintain as '0'

bit 5 RP0: Register Bank Select bits (used for direct addressing)
01 = Bank 1 (80h - FFh)
00 = Bank 0 (00h - 7Fh)

bit 4 $\overline{TO}$: Time-out bit
1 = After power-up, `CLRWDT` instruction, or `SLEEP` instruction
0 = A WDT time-out occurred

bit 3 $\overline{PD}$: Power-down bit
1 = After power-up or by the `CLRWDT` instruction
0 = By execution of the `SLEEP` instruction

bit 2 Z: Zero bit
1 = The result of an arithmetic or logic operation is zero
0 = The result of an arithmetic or logic operation is not zero

bit 1 DC: Digit carry/$\overline{borrow}$ bit (`ADDWF`, `ADDLW`, `SUBLW`, `SUBWF` instructions) (for $\overline{borrow}$, the polarity is reversed)
1 = A carry-out from the 4th low order bit of the result occurred
0 = No carry-out from the 4th low order bit of the result

bit 0 C: Carry/$\overline{borrow}$ bit (`ADDWF`, `ADDLW`, `SUBLW`, `SUBWF` instructions) (for $\overline{borrow}$, the polarity is reversed)
1 = A carry-out from the Most Significant bit of the result occurred
0 = No carry-out from the Most Significant bit of the result occurred

Note: A subtraction is executed by adding the two's complement of the second operand. For rotate (`RRF`, `RLF`) instructions, this bit is loaded with either the high or low order bit of the source register.

Legend:			
R = Readable bit	W = Writable bit	U = Unimplemented bit, read as '0'	
- n = Value at POR	'1' = Bit is set	'0' = Bit is cleared	x = Bit is unknown

2.3.2 OPTION REGISTER

The OPTION register is a readable and writable register which contains various control bits to configure the TMR0/WDT prescaler, the external INT interrupt, TMR0, and the weak pull-ups on PORTB.

Note: When the prescaler is assigned to the WDT (PSA = '1'), TMR0 has a 1:1 prescaler assignment.

REGISTER 2-2: OPTION REGISTER (ADDRESS 81h)

R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1
RBPU	INTEDG	T0CS	T0SE	PSA	PS2	PS1	PS0
bit 7							bit 0

bit 7 **$\overline{\text{RBPU}}$:** PORTB Pull-up Enable bit
1 = PORTB pull-ups are disabled
0 = PORTB pull-ups are enabled by individual port latch values

bit 6 **INTEDG:** Interrupt Edge Select bit
1 = Interrupt on rising edge of RB0/INT pin
0 = Interrupt on falling edge of RB0/INT pin

bit 5 **T0CS:** TMR0 Clock Source Select bit
1 = Transition on RA4/T0CKI pin
0 = Internal instruction cycle clock (CLKOUT)

bit 4 **T0SE:** TMR0 Source Edge Select bit
1 = Increment on high-to-low transition on RA4/T0CKI pin
0 = Increment on low-to-high transition on RA4/T0CKI pin

bit 3 **PSA:** Prescaler Assignment bit
1 = Prescaler is assigned to the WDT
0 = Prescaler is assigned to the Timer0 module

bit 2-0 **PS2:PS0:** Prescaler Rate Select bits

Bit Value	TMR0 Rate	WDT Rate
000	1 : 2	1 : 1
001	1 : 4	1 : 2
010	1 : 8	1 : 4
011	1 : 16	1 : 8
100	1 : 32	1 : 16
101	1 : 64	1 : 32
110	1 : 128	1 : 64
111	1 : 256	1 : 128

Legend:			
R = Readable bit	W = Writable bit	U = Unimplemented bit, read as '0'	
- n = Value at POR	'1' = Bit is set	'0' = Bit is cleared	x = Bit is unknown

2.3.3 INTCON REGISTER

The INTCON register is a readable and writable register that contains the various enable bits for all interrupt sources.

Note:	Interrupt flag bits are set when an interrupt condition occurs, regardless of the state of its corresponding enable bit or the global enable bit, GIE (INTCON<7>).

REGISTER 2-3: INTCON REGISTER (ADDRESS 0Bh, 8Bh)

R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-x
GIE	EEIE	T0IE	INTE	RBIE	T0IF	INTF	RBIF
bit 7							bit 0

bit 7 **GIE: Global Interrupt Enable bit**
1 = Enables all unmasked interrupts
0 = Disables all interrupts

bit 6 **EEIE: EE Write Complete Interrupt Enable bit**
1 = Enables the EE Write Complete interrupts
0 = Disables the EE Write Complete interrupt

bit 5 T0IE: TMR0 Overflow Interrupt Enable bit
1 = Enables the TMR0 interrupt
0 = Disables the TMR0 interrupt

bit 4 INTE: RB0/INT External Interrupt Enable bit
1 = Enables the RB0/INT external interrupt
0 = Disables the RB0/INT external interrupt

bit 3 RBIE: RB Port Change Interrupt Enable bit
1 = Enables the RB port change interrupt
0 = Disables the RB port change interrupt

bit 2 T0IF: TMR0 Overflow Interrupt Flag bit
1 = TMR0 register has overflowed (must be cleared in software)
0 = TMR0 register did not overflow

bit 1 INTF: RB0/INT External Interrupt Flag bit
1 = The RB0/INT external interrupt occurred (must be cleared in software)
0 = The RB0/INT external interrupt did not occur

bit 0 RBIF: RB Port Change Interrupt Flag bit
1 = At least one of the RB7:RB4 pins changed state (must be cleared in software)
0 = None of the RB7:RB4 pins have changed state

Legend:			
R = Readable bit	W = Writable bit	U = Unimplemented bit, read as '0'	
- n = Value at POR	'1' = Bit is set	'0' = Bit is cleared	x = Bit is unknown

2.4 PCL and PCLATH

The program counter (PC) specifies the address of the instruction to fetch for execution. The PC is 13 bits wide. The low byte is called the PCL register. This register is readable and writable. The high byte is called the PCH register. This register contains the PC<12:8> bits and is not directly readable or writable. If the program counter (PC) is modified or a conditional test is true, the instruction requires two cycles. The second cycle is executed as a NOP. All updates to the PCH register go through the PCLATH register.

2.4.1 STACK

The stack allows a combination of up to 8 program calls and interrupts to occur. The stack contains the return address from this branch in program execution.

Mid-range devices have an 8 level deep x 13-bit wide hardware stack. The stack space is not part of either program or data space and the stack pointer is not readable or writable. The PC is PUSHed onto the stack when a CALL instruction is executed or an interrupt causes a branch. The stack is POPed in the event of a RETURN, RETLW or a RETFIE instruction execution. PCLATH is not modified when the stack is PUSHed or POPed.

After the stack has been PUSHed eight times, the ninth push overwrites the value that was stored from the first push. The tenth push overwrites the second push (and so on).

2.5 Indirect Addressing; INDF and FSR Registers

The INDF register is not a physical register. Addressing INDF actually addresses the register whose address is contained in the FSR register (FSR is a *pointer*). This is indirect addressing.

EXAMPLE 2-1: INDIRECT ADDRESSING

- Register file 05 contains the value 10h
- Register file 06 contains the value 0Ah
- Load the value 05 into the FSR register
- A read of the INDF register will return the value of 10h
- Increment the value of the FSR register by one (FSR = 06)
- A read of the INDF register now will return the value of 0Ah.

Reading INDF itself indirectly (FSR = 0) will produce 00h. Writing to the INDF register indirectly results in a no-operation (although STATUS bits may be affected).

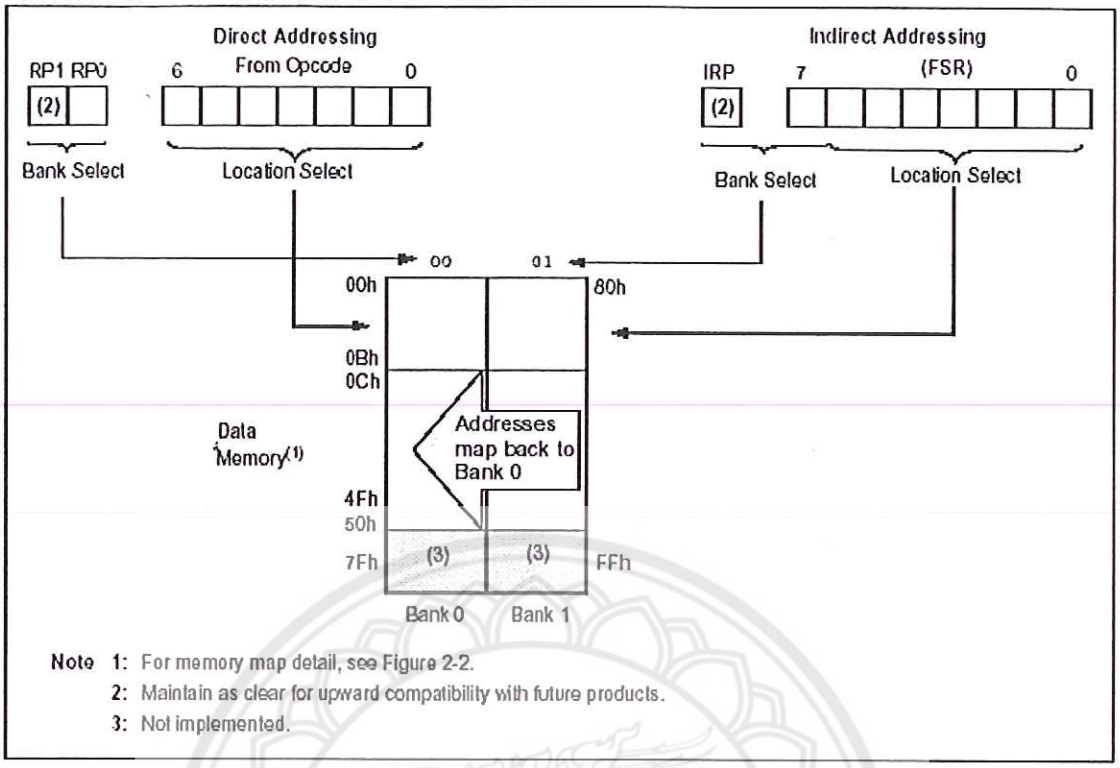
A simple program to clear RAM locations 20h-2Fh using indirect addressing is shown in Example 2-2.

EXAMPLE 2-2: HOW TO CLEAR RAM USING INDIRECT ADDRESSING

```
        movlw   0x20    ;initialize pointer
        movwf   FSR     ;to RAM
NEXT    clrf    INDF    ;clear INDF register
        incf    FSR     ;inc pointer
        btfss   FSR,4   ;all done?
        goto    NEXT    ;NO, clear next
CONTINUE
        :               ;YES, continue
```

An effective 9-bit address is obtained by concatenating the 8-bit FSR register and the IRP bit (STATUS<7>), as shown in Figure 2-3. However, IRP is not used in the PIC16F84A.

FIGURE 2-3: DIRECT/INDIRECT ADDRESSING



Note 1: For memory map detail, see Figure 2-2.
2: Maintain as clear for upward compatibility with future products.
3: Not implemented.

ภาคผนวก ค

วิธีการใช้เครื่องมือวัดความชื้นในดิน

ขั้นแรกทำการเตรียมดินที่จะใช้วัดความชื้น โดยการนำดินมาบิดละเอียดแล้วนำไปอบให้แห้งที่อุณหภูมิ 120 ^{0}C เสร็จแล้วนำดินมาใส่ภาชนะที่เตรียมไว้จนเต็ม

การวัดจะเปิดเครื่องที่สร้างขึ้นก่อน เพื่อให้เครื่องได้กำหนดค่าเริ่มต้นเป็น 0% ก่อนแล้วทำการต่อเครื่องกับภาชนะที่ใส่ดินแล้วจากนั้นก็เติมน้ำครั้งละ 0.2 kg ให้ทั่ว แล้วรอให้น้ำซึมเข้าไปในดินเป็นเวลา 5 นาทีก่อนแล้วถึงจะเติมน้ำครั้งต่อไป โดยที่ไม่ต้องถอดสายต่อออกจากเครื่องหรือปิดเครื่อง

ประวัติผู้เขียนโครงงาน

ชื่อ นายจักรนะรินทร์ พรมชัย

ภูมิลำเนา 198 หมู่ 6 ต.ดอนศิลา อ.เวียงชัย จ.เชียงราย 57210

ประวัติการศึกษา

- จบการศึกษาระดับมัธยมศึกษาจาก ร.ร. ดำรงราษฎร์สงเคราะห์ จ.เชียงราย
- ปัจจุบันศึกษาในระดับปริญญาตรีชั้นปีที่ 4

สาขาวิศวกรรมไฟฟ้า คณะวิศวกรรมศาสตร์

มหาวิทยาลัยนเรศวร

E-mail : jaknarin555@chaiyo.com

ชื่อ นายรังสฤษฎ์ สุทธิคุณ

ภูมิลำเนา 637 หมู่ 1 ต.หนองกุง อ.น้ำพอง จ.ขอนแก่น

ประวัติการศึกษา

- จบการศึกษาระดับมัธยมศึกษาจาก ร.ร. แก่นนครวิทยาลัย
- ปัจจุบันศึกษาในระดับปริญญาตรีชั้นปีที่ 4

สาขาวิศวกรรมไฟฟ้า คณะวิศวกรรมศาสตร์

มหาวิทยาลัยนเรศวร

E-mail : soottricoon@yahoo.com