การประยุกต์การใช้งานตัวแปลงสัญญาณแบบอะนาลอกไปสู่แบบดิจิตอล (PCF8591) เพื่อใช้วัดและควบคุมความชื้นในดิน

8-BIT A/D CONVERTER (PCF8591) APPLICATION FOR SOIL HUMIDITY MEASUREMENT AND CONTROL

นายมานพ รัตนะ รหัส 46380251

นางสาวศุลีพร ภักดีแก้ว รหัส 46380253

ปริญญานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาหลักสูตรปริญญาวิศวกรรมศาสตรบัณฑิต

สาขาวิชาวิศวกรรมคอมพิวเตอร์ ภาควิชาวิศวกรรมไฟฟ้าและคอมพิวเตอร์

คณะวิศวกรรมศาสตร์ มหาวิทยาลัยนเรศวร

ปีการศึกษา 2549

ใบรับรองโครงงานวิศวกรรม

หัวข้อโครงงาน การประยุกต์การใช้งานตัวแปลงสัญญาณแบบอะนาลอกไปสู่แบบดิจิตอล (PCF8591) เพื่อใช้วัดและควบคุมความชื้นในดิน

ผู้ดำเนินโครงงาน นายมานพ รัตนะ รหัส 46380251

นางสาวศุลีพร ภักดีแก้ว รหัส 46380253

อาจารย์ที่ปรึกษา ดร.อัครพันธ์ วงศ์กังแห

สาขาวิชา วิศวกรรมคอมพิวเตอร์

ภาควิชา วิศวกรรมไฟฟ้าและคอมพิวเตอร์

ปีการศึกษา 2549

.....

คณะวิศวกรรมศาสตร์ มหาวิทยาลัยนเรศวร อนุมัติให้โครงงานฉบับนี้เป็นส่วนหนึ่งของการศึกษาหลักสูตรวิศวกรรมศาสตรบัณฑิต สาขาวิชาวิศวกรรมคอมพิวเตอร์

คณะกรรมการสอบโครงงานวิศวกรรม

.....ประธานกรรมการ

(ดร.อัครพันธ์ วงศ์กังแห)

.....กรรมการ

(ดร.สุรเดช จิตประไพกุลศาล)

.....กรรมการ

(อาจารย์ศิริพร เดชะศิลารักษ์)

หัวข้อโครงงาน	การประยุกต์การใช้งานตัวแปลงสัญญาณแบบอะนาลอกไปสู่แบบดิจิตอล (PCF8591) เพื่อใช้วัดและควบคุมความชื้นในดิน
ผู้ดำเนินโครงงาน	นายมานพ รัตนะ รหัส 46380251
	นางสาวศุลีพร ภักดีแก้ว รหัส 46380253
อาจารย์ที่ปรึกษา	ดร.อัครพันธ์ วงศ์กังแห
สาขาวิชา	วิศวกรรมคอมพิวเตอร์
ภาควิชา	วิศวกรรมไฟฟ้าและคอมพิวเตอร์
ปีการศึกษา	2549

บทคัดย่อ

โครงงานนี้ได้จัดทำขึ้นเพื่อสร้างอุปกรณ์วัดและควบคุมความชื้นในดิน โดยใช้งานตัวแปลงสัญญาณแบบอะนาลอกไปสู่แบบดิจิตอล (PCF8591) โดยนำอุปกรณ์ที่สร้างขึ้นมาใช้ในการดูแลรักษาต้นไม้เมื่อคุณไม่มีเวลาในการรดน้ำหรือไม่อยู่บ้าน เมื่อความชื้นของดินบริเวณที่ปลูกต้นไม้ต่ำวาล์วน้ำจะทำการปล่อยน้ำออกมาเพื่อปรับให้ความชื้นในดินอยู่ในระดับที่กำหนดแล้วจึงหยุดทำงาน

โครงงานนี้ได้นำ PCF8591 ซึ่งภายในบอร์ดไมโครคอนโทรลเลอร์มีอยู่ 4 พอร์ต มาใช้รับค่าแรงดันไฟฟ้าและส่งเข้าไปประมวลผลภายในชิพ MCU เบอร์ P89C51RD2BN ขับเอาท์พุตออกทางพอร์ต 8255 เพื่อนำไปใช้ในการควบคุมวาล์วน้ำต่อไป โปรแกรมควบคุมใช้ภาษาซีในการเขียนควบคุมอุปกรณ์ทั้งหมด ได้วงจรวัดและควบคุมความชื้น ซึ่งสามารถนำมาใช้งานได้จริง

Project	8-bit A/D Converter (PCF8591) Application for Soil Humidity Measurement and Control.
Name	Mr.Manop Rattana ID. 46380251
	Miss Suleeporn Pakdeekaew ID. 45380079
Project Advisor	Dr.Akaraphunt Vongkunghae
Major	Computer Engineering.
Department	Electrical and Computer Engineering.
Academic Year	2006

ABSTRACT

An 8-bit A/D Converter (PCF8591) Application for Soil Humidity Measurement and Control project is created, a simple Soil Humidity Measurement and Control device.

This project used 8-bit A/D Converter (PCF8591) on a microcontroller board, 4 A/D ports are used to take the voltage and send to MCU processor P89C51RD2BN, to 8255 ports for driving indicators or other devices. A software in C programming language is used and embedded in the MCU.

กิตติกรรมประกาศ

การจัดทำโครงงานการประยุกต์การใช้งานตัวแปลงสัญญาณแบบอะนาลอกไปสู่แบบดิจิตอล (PCF8591) เพื่อใช้วัดและควบคุมความชื้นในดินสำเร็จลุล่วงไปได้ด้วยดี ก็ด้วยความเสียสละ ความอนุเคราะห์ และน้ำใจจากบุคคลหลายฝ่าย โดยทั้งนี้คณะผู้จัดทำขอขอบพระคุณคณะกรรมการควบคุมโครงงานทุกท่านอันได้แก่ ดร.สุรเดช จิตประไพกุลศาล และอาจารย์ศิริพร เดชะศิลารักษ์ รวมทั้งอาจารย์ที่ปรึกษาคือ ดร.อัครพันธ์ วงศ์กังแห ที่กรุณาสละเวลา แนวความคิด ประสบการณ์ และคำปรึกษาอันเป็นประโยชน์ในการทำโครงงานนี้เป็นอย่างยิ่ง

ขอบคุณเพื่อนๆ ทุกคนที่คอยถามไถ่ ช่วยเหลือและแนะนำ ทั้งในเรื่องการเรียนและการจัดทำโครงงานงานในครั้งนี้

ขอกราบขอบพระคุณบิดา มารดา ญาติพี่น้อง ที่ช่วยดูแล เป็นกำลังใจตลอดมา

มานพ รัตนะ

ศุลีพร ภักดีแก้ว

สารบัญ

สารบัญ(ต่อ)

สารบัญ(ต่อ)

สารบัญตาราง

สารบัญรูป

สารบัญรูป(ต่อ)

บทที่ 1
บทนำ

1.1 ความเป็นมาและความสำคัญของปัญหา

ประเทศไทยเป็นประเทศที่ประชากรส่วนใหญ่ประกอบอาชีพเกษตรกรรมและยังเคยเป็นอาชีพที่สร้างรายได้ให้กับประเทศมากที่สุด ซึ่งการเกษตรในปัจจุบันได้มีการนำเทคโนโลยีและเครื่องทุนแรงต่างๆ เข้ามาใช้ในการทำเกษตรกันอย่างแพร่หลาย การใช้เทคโนโลยีอันรวมไปถึงปัจจัยภายนอกในการปลูกพืชด้วย เช่น การควบคุมอุณหภูมิ การควบคุมปริมาณธาตุอาหารในดิน การควบคุมความเป็นกรด- เบสของดิน การควบคุมปริมาณและความเข้มของแสง และการควบคุมความชื้นในดิน เป็นต้น

การควบคุมความชื้นในดินเป็นปัจจัยสำคัญที่มีอิทธิพลต่อการเจริญเติบโตของพืชและการให้ผลผลิตทางการเกษตรเป็นอย่างมาก เพราะพืชต้องใช้น้ำในกระบวนการหายใจและแลกเปลี่ยนก๊าซอันเป็นกระบวนการที่ทำให้พืชเจริญเติบโต จะเห็นได้ว่าถ้าหากพืชขาดธาตุอาหารก็อาจจะยังมีชีวิตอยู่ได้ แต่ถ้าขาดน้ำพืชก็จะเหี่ยวเฉาและตายในที่สุด พืชแต่ละชนิดจะต้องการปริมาณของน้ำต่างกัน ปริมาณความชื้นในดินที่พืชแต่ละชนิดต้องการก็ย่อมแตกต่างกัน การประยุกต์ใช้ 8- bit A/D converter (PCF8591) ในบอร์ด MCS-51 เพื่อวัดและควบคุมปริมาณความชื้นภายในดินจะช่วยได้มากกับพืชที่ต้องการน้ำอย่างเพียงพอและสม่ำเสมอ การที่เกษตรกรรดน้ำนั้นอาจจะยังไม่เพียงพอกับที่พืชต้องการ สภาพภูมิประเทศและสภาพดินที่แตกต่างกันนั้นทำให้น้ำซึมผ่านผิวดินลงไปได้น้อยและกักเก็บน้ำได้แตกต่างกัน ส่งผลให้พืชได้รับน้ำอย่างไม่เพียงพอและสม่ำเสมอ

โครงงานนี้จึงน่าจะเป็นอีกทางเลือกหนึ่งในการช่วยและอำนวยความสะดวกในการปลูกพืช น้ำซึ่งเป็นตัวช่วยละลายแร่ธาตุต่างๆภายในดิน เมื่อพืชได้รับปัจจัยต่างๆในการการสังเคราะห์แสงเพื่อสร้างอาหารอย่างเหมาะสมและเพียงพอแล้ว ผลผลิตทางการเกษตรย่อมเพิ่มขึ้นและมีคุณภาพตามดีตามความต้องการของตลาด ถ้ามีอุปกรณ์วัดและควบคุมความชื้นนี้ เกษตรกรก็ไม่ต้องคอยรดน้ำพืช สามารถเอาเวลาว่างจากการรดน้ำไปดูแลและทำงานการเกษตรด้านอื่นๆได้ ก่อให้เกิดรายได้เพิ่มเติมอีกทางหนึ่งได้

1.2 วัตถุประสงค์

1. เพื่อศึกษาการเชื่อมต่อและการประยุกต์ใช้งานตัวแปลงสัญญาณ
2. เพื่อเข้าใจหลักการพื้นฐานของโปรแกรมและพัฒนาให้สามารถควบคุมไมโครคอนโทรลเลอร์ในงานวัดและงานควบคุมอัตโนมัติได้
3. เพื่อให้มีความรู้ความสามารถในการออกแบบวงจรและประยุกต์ใช้งานวัดและควบคุม
4. เพื่อสามารถติดตามความเคลื่อนไหวและแสวงหาความรู้ในเรื่องระบบควบคุมอัตโนมัติ
5. เพื่อพัฒนาและส่งเสริมทางด้านการเกษตรให้มีความก้าวหน้าและมีประสิทธิภาพ
6. เพื่อให้มีความรู้ความสามารถในการนำซอฟแวร์และฮาร์ดแวร์ ให้ทำงานร่วมกันได้อย่างสอดคล้อง

1.3 ขอบเขตของโครงงาน

1. ออกแบบโปรแกรมและวงจรวัดความชื้น เพื่อป้อนสัญญาณให้กับตัวแปลงสัญญาณอะนาลอกเป็นดิจิตอล เพื่อนำมาใช้ในการวัดและควบคุมความชื้นภายในดิน สำหรับใช้ในการดูแลเรื่องการให้น้ำของพืช
2. ศึกษาหลักการที่น่าจะนำมาใช้เป็นตัวระบุค่าความชื้นภายในดิน

1.4 ขั้นตอนการดำเนินงาน

ตารางที่ 1.1 ขั้นตอนการดำเนินงานโครงงานเครื่องวัดและควบคุมความชื้น

กิจกรรม	ปี 2548		ปี 2549									
	พ.ย.	ธ.ค.	ม.ค.	ก.พ.	มี.ค.	เม.ย.	พ.ค.	มิ.ย.	ก.ค.	ส.ค.	ก.ย.	ต.ค.
1. ศึกษาและรวบรวมข้อมูลที่เกี่ยวข้องกับการวัดและควบคุมความชื้น รวมทั้งปัญหาของการวัดและควบคุมความชื้นในปัจจุบัน												
2. ออกแบบสร้างวงจรวัดและควบคุมความชื้นในดินและฐานข้อมูลที่เกี่ยวข้อง												
3. สร้างวงจรวัดและควบคุมความชื้นในดิน และฐานข้อมูล												

กิจกรรม	ปี 2548		ปี 2549									
	พ.ย.	ธ.ค.	ม.ค.	ก.พ.	มี.ค.	เม.ย.	พ.ค.	มิ.ย.	ก.ค.	ส.ค.	ก.ย.	ต.ค.
4. ทดลองการวัดและควบคุมความชื้นในดิน และบันทึกผล												
5. ทำการปรับปรุงและแก้ไขปัญหา												
6. จัดทำรายงาน												

1.5 ผลที่คาดว่าจะได้รับ

1. มีความรู้ในการเชื่อมต่อและการประยุกต์ใช้งานตัวแปลงสัญญาณอะนาลอกเป็นดิจิตอล
2. สามารถสร้างโปรแกรมให้สามารถควบคุมไมโครคอนโทรลเลอร์ในงานวัดและควบคุมได้
3. ทราบความเคลื่อนไหวและความรู้ในเรื่องระบบควบคุมอัตโนมัติ
4. พัฒนาและส่งเสริมทางด้านการเกษตรให้มีความก้าวหน้าและมีประสิทธิภาพ
5. มีความรู้ความสามารถในการนำซอฟแวร์และฮาร์ดแวร์มาทำงานร่วมกันได้ และได้ชิ้นงานออกมาเป็นเครื่องวัดและควบคุมความชื้นในดิน

1.6 งบประมาณ

วงจรและอุปกรณ์	1, 000 บาท
หนังสือและเอกสาร	1, 000 บาท
ทำรูปเล่มรายงาน	700 บาท
อุปกรณ์อื่นๆ	500 บาท
รวมทั้งสิ้น	3, 200 บาท

บทที่ 2
หลักการและทฤษฎีที่เกี่ยวข้อง

2.1 การวัดและควบคุมความชื้นในดิน

2.1.1 ความชื้นของดิน (Soil Moisture)

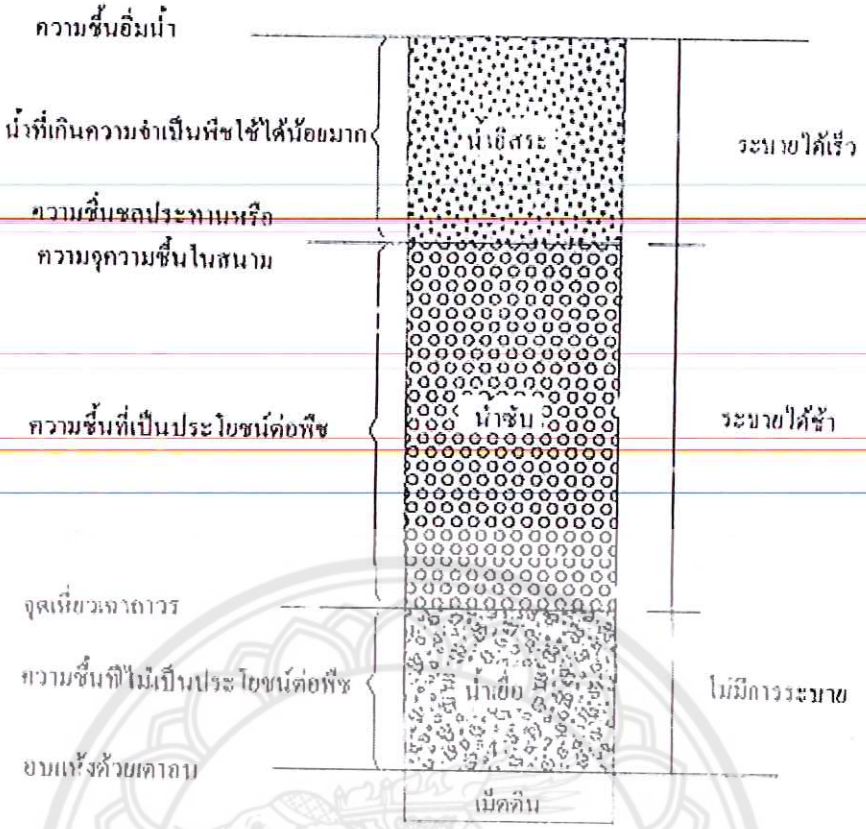
ความชื้นของดินในที่นี้หมายถึง น้ำในดินเท่านั้น ซึ่งแบ่งได้เป็น 4 ประเภท [1] ดังนี้

1. ความชื้นอิ่มน้ำ (Saturation) คือปริมาณน้ำในดินที่อยู่ในช่องว่างระหว่างเม็ดดินทั้งหมด ทั้งช่องว่างขนาดเล็กและช่องว่างขนาดใหญ่ ถ้าดินระบายน้ำได้ดี น้ำที่อยู่ในช่องว่างขนาดใหญ่ก็จะเคลื่อนที่สู่เบื้องล่างด้วยแรงดึงดูดของโลกภายในระยะเวลาสั้น

2. ความชื้นชลประทานหรือความจุความชื้นในสนาม (Field Capacity) คือปริมาณน้ำในดิน ที่เหลือหลังจากน้ำอิสระถูกระบายออกจากช่องว่างขนาดใหญ่หมดแล้ว หรือปริมาณน้ำสูงสุดที่ดินสามารถดูดยึดไว้ได้จากแรงดึงดูดของโลก (มีน้ำซับและน้ำเยื่อ) จึงทำให้ช่องว่างขนาดเล็กมีน้ำอยู่เต็ม และช่องว่างขนาดใหญ่มีอากาศอยู่เต็ม ระดับความชื้นชลประทานเป็นระดับความชื้นในดินที่เป็นประโยชน์ต่อพืชสูงสุด คือรากพืชสามารถดูดนำไปใช้และอยู่ในดินได้นานพอที่พืชจะดูดไปใช้ได้เพียงพอ

3. ความชื้นจุดเหี่ยวถาวร (Permanent Wilting Point) คือปริมาณน้ำในดิน ที่พืชไม่สามารถดูดใช้ได้เพียงพอกับการคายน้ำ ถ้าหากไม่ได้รับน้ำเพิ่ม พืชก็จะเริ่มเหี่ยวเฉาจนกระทั่งเหี่ยวเฉาถาวร เรียกว่าเป็นความชื้นที่จุดเหี่ยวเฉาถาวร ก่อนที่พืชจะเหี่ยวเฉาถาวร พืชจะเกิดอาการเหี่ยวเฉาได้หลายครั้ง โดยเฉพาะวันที่มีอากาศร้อนจัด มีความชื้นอากาศต่ำ มีลมแรง พืชใบบางและใบกว้าง ทำให้มีการคายน้ำออกทางใบมาก เมื่อมากกว่าอัตราการดูดน้ำจากดิน พืชก็จะเหี่ยวเฉา แต่เมื่ออากาศเย็นลงพืชจะสดชื่นเช่นเดิม

4. ความชื้นเมื่ออบแห้ง คือปริมาณน้ำในดินหลังจากถูกอบไว้ที่อุณหภูมิ 105-110°C นานกว่า 15 ชั่วโมง จนไม่มีน้ำระเหยออกมาจากดิน ถือว่าดินในสภาพนี้มีค่าแรงดึงความชื้นไม่น้อยกว่า 10,000 บาร์ และจะใช้น้ำหนักดินอบแห้งเป็นหลักสำหรับคำนวณหาค่าต่างๆ



รูปที่ 2.1 แสดงความชื้นที่ระดับต่างๆ

2.1.2 หลักการวัดความชื้น

ในโครงงานนี้จะใช้คุณสมบัติของน้ำและการนำทางไฟฟ้ามาใช้ในการวัดความชื้น กล่าวคือ ถ้าในดินมีความชื้นมาก ก็จะมีปริมาณน้ำในดินมาก ทำให้การนำไฟฟ้าเป็นไปได้ดี แต่ถ้าหากดินแห้ง การนำไฟฟ้าก็จะน้อย โดยจะนำขั้วไฟฟ้าที่เป็นทองแดงสองอันเสียบลงไปในดินและใช้สายไฟเป็นตัวเชื่อมต่อในการส่งกระแสไฟฟ้าเข้าวงจรวัดและควบคุมความชื้น สำหรับนำไปประมวลผลและส่งสัญญาณไปควบคุมหัวจ่ายน้ำต่อไป

2.1.3 ประสิทธิภาพในการวัดและควบคุมความชื้น

1. สามารถใช้ตรวจสภาพของดินได้ โดยจะทราบได้ทันทีว่า ในขณะนี้ดินมีสภาพเป็นอย่างไร เช่น มีสภาพชื้นแฉะหรือว่าแห้งจนเกินไป เป็นต้น

2. สามารถแสดงผลออกมา เพื่อให้ทราบว่าตอนนี้ดินมีสภาพเป็นอย่างไร และทำการปรับปรุงความชื้นในดินก่อนที่จะทำการเพาะปลูกต้นไม้

3. สามารถเป็นเครื่องมือสำหรับใช้ในการบำรุงรักษาต้นไม้ โดยการแสดงผลในรูปของ 7-Segment ว่าขณะนี้ดินมีสภาพเป็นอย่างไร ควรจะรดน้ำต้นไม้ได้หรือยัง หรือว่าชื้นไปจนอาจทำให้รากเน่า [2]

2.2 หลักการของตัวแปลงสัญญาณแบบอะนาลอกไปสู่แบบดิจิตอล

ในโครงงานนี้เราเลือกใช้ตัวแปลงสัญญาณแบบอะนาลอกไปสู่ดิจิตอลสัญญาณที่ใช้ในอุปกรณ์อิเล็คทรอนิคส์ มี 2 ชนิด [3] คือ สัญญาณอะนาลอกและสัญญาณดิจิตอล สัญญาณอะนาลอกจะใช้ในอุปกรณ์ทั่วๆ ไปและใช้ในการควบคุมแบบเก่า ปัจจุบันไมโครคอนโทรลเลอร์เข้ามามีบทบาทในการควบคุมเป็นอย่างมาก เนื่องจากทำได้ง่ายและรวดเร็วยิ่งขึ้น ในการควบคุมนั้นเราจำเป็นต้องใช้สัญญาณดิจิตอลในการติดต่อกับไมโครโปรเซสเซอร์หรือไมโครคอนโทรลเลอร์ แต่ในความเป็นจริงเราต้องใช้สัญญาณอะนาลอกในการควบคุม ดังนั้นเราจึงจำเป็น ต้องมีการเปลี่ยนสัญญาณอะนาลอกเป็นสัญญาณดิจิตอล แล้วจึงนำสัญญาณนั้นเข้ามาสู่ไมโครโปรเซสเซอร์หรือไมโครคอนโทรลเลอร์ เพื่อใช้ควบคุมระบบต่อไป แม้ว่าสัญญาณอะนาลอกนั้นมีความแน่นอนและแม่นยำสูง แต่สัญญาณอะนาลอกนั้นก็ควบคุมได้ยาก เนื่องจากในสภาพแวดล้อม มีสัญญาณรบกวนอยู่มาก (noise) และการที่จะทำให้การควบคุมแบบอะนาลอก มีความสามารถควบคุม เท่ากับการควบคุมแบบดิจิตอลนั้นทำได้ยาก เนื่องจากวงจรควบคุมแบบอะนาลอกจะต้องมีความซับซ้อนสูง อย่างไรก็ตาม สัญญาณดิจิตอลก็ไม่สามารถทดแทนความละเอียดของสัญญาณอะนาลอกได้อย่างสมบูรณ์ แต่ทำให้การควบคุมนั้นทำให้ง่ายและสะดวกยิ่งขึ้น

ในส่วนนี้เป็นการแปลงสัญญาณอะนาลอกที่รับมาจากหัววัดความชื้นให้เป็นสัญญาณทางดิจิตอล เพื่อส่งให้ซีพียูรับไปประมวลผลอีกทีหนึ่ง ในการทำวงจร ADC (Analog to Digital Converter) จะมีไอซีสำหรับวงจรนี้ให้เลือกใช้หลายเบอร์ ในส่วน ADC ของระบบควบคุมความชื้นด้วย ไมโครคอนโทรลเลอร์ ได้เลือกใช้ไอซีเบอร์ PCF8591 มาทำเป็นดิจิตอลโวลต์มิเตอร์ (digital voltmeter) จากหัววัดความชื้น

การแปลงสัญญาณอะนาลอกเป็นสัญญาณดิจิตอลมีประโยชน์มากในการควบคุมอุปกรณ์สวิตชิ่ง ซึ่งมีลักษณะการแปลงสัญญาณได้หลายวิธี แต่ละวิธีจะมีอัลกอริทึม ความรวดเร็วในการทำงาน และการใช้อุปกรณ์ฮาร์ดแวร์ต่างกันด้วย ทำให้ขนาดและราคาต่างกัน ขึ้นกับความต้องการของผู้ใช้ที่จะต้องเลือกให้เหมาะสมกับงานที่ใช้ และงบประมาณที่มีอยู่ ชิพเบอร์ PCF8591 เป็น A/D ขนาด 8 บิต 4 ช่อง และ D/A 8 บิต 1 ช่อง สามารถต่อใช้งานเพื่อการทดลองโดยใช้พอร์ท 4 A/D, 1 D/A ได้ทันที

2.2.1 Analog to Digital Conversion (ADC)

เหตุการณ์ต่างๆ ที่เป็นอยู่ในปัจจุบันมักจะเป็นเหตุการณ์อะนาลอกเป็นส่วนใหญ่ดังนั้นสัญญาณต่าง ๆ ที่ส่งออกมาจึงเป็นสัญญาณอะนาลอกด้วย แต่ในตัวเครื่องคอมพิวเตอร์จะมีลักษณะการทำงานของมันเป็นแบบดิจิตอล คือ มีแรงดันไฟฟ้าในสื่อ ข้อมูลที่แน่นอน คือ '0' และ '1' ดังนั้นเมื่อเหตุการณ์หรือสัญญาณที่ส่งเข้ามาที่คอมพิวเตอร์มีลักษณะเป็นสัญญาณอะนาลอก จึงต้องมีตัวที่เป็นตัวแปลงสัญญาณช่วยอีกขั้นหนึ่ง

2.2.2 ตัวแปลงสัญญาณ

การเชื่อมต่อเหตุการณ์แบบอะนาลอกที่อยู่แวดล้อมเข้าสู่ระบบคอมพิวเตอร์ จำเป็นจะต้องมีตัวกลางในการแปลงเหตุการณ์อะนาลอกให้เป็นสัญญาณทางอิเล็กทรอนิกส์ ตัวแปลงดังกล่าวเรียกว่า"ทรานส์ดิวเซอร์" (Transducer) [4] การที่จะทำให้คอมพิวเตอร์เชื่อมต่อเข้ากับเหตุการณ์อะนาลอกโดยตรงนั้นไม่ได้ เราไม่สามารถเชื่อมต่อค่าความชื้นเข้าคอมพิวเตอร์ได้ จึงจำเป็นต้องใช้ตัวแปลงสัญญาณค่าความชื้นนั้นเป็นสัญญาณทางอิเล็กทรอนิกส์อาจได้เป็นสัญญาณอะนาลอกที่มีค่าสัมพันธ์กับค่าความชื้นนั้น และทำการแปลงสัญญาณอะนาลอกให้เป็นดิจิตอลอีกครั้ง โดยกระบวนการทางอิเล็กทรอนิกส์ จึงสามารถนำค่าความร้อนดังกล่าวเข้าสู่ระบบคอมพิวเตอร์ได้ ดังรูปที่ 2.2

สัญญาณอิเล็กทรอนิกส์ที่ใช้กันอยู่ทั่วไป แบ่งออกเป็น 2 ประเภท คือ

1. สัญญาณแบบอะนาลอก : ซึ่งแอมพลิจูดมีค่าอยู่ระหว่างระดับสัญญาณสูงสุดกับระดับสัญญาณอ้างอิง

2.สัญญาณดิจิตอล : ซึ่งมีระดับสัญญาณหลายระดับที่ถูกแทนด้วยเลขฐานสอง (binary)

รูปที่ 2.2 แสดงองค์ประกอบการทำงานของ ADC

วงจร ADC แบบพื้นฐานที่ใช้กันอยู่มีหลายประเภทด้วยกัน PCF8591 เป็นตัวแปลงสัญญาณแบบประมาณค่าใกล้เคียง (Successive Approximation ADC) [5] วงจร ADC แบบนี้จะใช้รีจิสเตอร์ฐานสองหรือไบนารีรีจิสเตอร์ในการส่งข้อมูลดิจิตอลของวงจร DAC (Digital to Analog Converter) ภายใน แต่ละบิตของรีจิสเตอร์จะเซตและรีเซตโดยการควบคุมจากวงจรควบคุม ต่อไปจะอธิบายการทำงานของ ADC แบบนี้ไปทีละขั้นตอนจากรูปที่ 2.3 ดังนี้

กำหนดแรงดันอะนาลอกอินพุต (Vin) มีค่า 4V (Vref = 5V)

1. ส่งสัญญาณเริ่มต้นการทำงาน (start converter) มายังชักเซสซีฟแอปพร็อกซิเมชั่นรีจิสเตอร์

2. ขณะนี้สถานะของรีจิสเตอร์จะไม่ว่าง (busy) สัญญาณนาฬิกาลูกแรกถูกส่งเข้ามาเพื่อกำหนดให้ค่าของรีจิสเตอร์เท่ากับ 00000000

3. เอาต์พุตของ DAC จะเป็น 0V ส่งไปในวงจรเปรียบเทียบ เพื่อเปรียบเทียบกับ Vin ในขณะนี้จะได้เอาต์พุตเท่ากับ 0V กำหนดเป็นลอจิก "0"

4. เมื่อสัญญาณนาฬิกาลูกต่อไปเข้ามา จะทำการเซตบิต MSB (Most Significant Bit) ของรีจิสเตอร์เป็น “1”

5. ในกรณีนี้เป็น ADC ขนาด 8 บิตดังนั้นการที่บิต MSB เซตจะทำให้วงจร DAC แปลงค่าเป็นแรงดัน 128 ×0.0195 = 2.496V เมื่อนำไปเปรียบเทียบที่วงจรเปรียบเทียบแรงดัน แต่ก็ยังน้อยกว่า Vin ดังนั้นเอาต์พุตของวงจรเปรียบเทียบยังคงเป็น “0” ทำให้รีจิสเตอร์ยังคงค่าบิต MSB ให้เป็น “1” ต่อไป

6. ต่อมาบิต B6 (ถัดจากบิต MSB 1 บิต เนื่องจากมี 8 บิต กำหนดบิต MSB = B7) จะเซตซึ่งจะมีค่าเท่ากับ 64×0.0195 = 1.248V นำไปรวมกับค่าของบิต MSB ที่มีอยู่ 2.496V จะได้ 3.744V นำไปเปรียบเทียบกับ Vin ก็ยังน้อยกว่า รีจิสเตอร์จึงยังคงค่า B6 ไว้ที่ “1” เช่นกัน

7. ต่อมาบิต B5 จะเซตทำให้แรงดันเอาต์พุตมา DAC กลายเป็น (128+64+32)×0.0195 = 4.368V ซึ่งมากกว่า Vin ทำให้วงจรเปรียบเทียบเกิดการเปลี่ยนสถานะเป็น “1” ซึ่งจะส่งสัญญาณมาควบคุมให้ B5 กลายเป็น “0”

8. ต่อมาบิต B4 จะเซตทำให้แรงดันเอาต์พุตมา DAC กลายเป็น (128+64+0+16)×0.0195 = 4.056V ซึ่งมากกว่า Vin ทำให้วงจรเปรียบเทียบเกิดการเปลี่ยนสถานะเป็น “1” ซึ่งจะส่งสัญญาณมาควบคุมให้ B4 กลายเป็น “0”

9. ต่อมาบิต B3 จะเซตทำให้แรงดันเอาต์พุตมา DAC กลายเป็น (128+64+0+0+8)×0.0195 = 3.9V ซึ่งน้อยกว่า Vin รีจิสเตอร์จึงยังคงค่า B3 ไว้ที่ “1”

10. ต่อมาบิต B2 จะเซตทำให้แรงดันเอาต์พุตมา DAC กลายเป็น (128+64+0+0+8+4)×0.0195 = 3.978V ซึ่งน้อยกว่า Vin รีจิสเตอร์จึงยังคงค่า B2 ไว้ที่ “1”

11.ต่อมาบิต B1 จะเซตทำให้แรงดันเอาต์พุตมา DAC กลายเป็น (128+64+0+0+8+4+2)×0.0195 = 4.017V ซึ่งมากกว่า Vin ทำให้วงจรเปรียบเทียบเกิดการเปลี่ยนสถานะเป็น “1” ซึ่งจะส่งสัญญาณมาควบคุมให้ B1 กลายเป็น “0”

12. เมื่อบิต LSB (Least Significant Bit) ถูกเซต ทำให้แรงดันเอาต์พุตมา DAC กลายเป็น (128+64+0+0+8+4+0+1)×0.0195 = 3.9975V นำไปเปรียบเทียบกับ Vin ปรากฏว่าน้อยกว่า Vin ทำให้ที่บิต B0 หรือ LSB มีค่าเป็น “1”

13. ขณะนี้ทุกบิตในรีจิสเตอร์ถูกนำมาแปลงค่าเรียบร้อยแล้ว ทำให้สถานะของรีจิสเตอร์กลับมาเป็น “พร้อมทำงาน (ready)”

14. ข้อมูลดิจิตอลที่ได้จากการ ADC แบบนี้ จะมีค่า 11001101_2 หรือ 3.9975V ซึ่งใกล้เคียงกับ Vin 4V มากที่สุด ถ้าหากรีจิสเตอร์มีจำนวนบิตมากกว่านี้ ความละเอียดของข้อมูลที่แปลงได้จะมีความใกล้เคียงมากขึ้น ช่วงเวลาของการแปลงสัญญาณจะเริ่มต้นขึ้นตั้งแต่สัญญาณนาฬิกาลูกแรกถูกส่งเข้าไปเตรียมระบบไปจนถึงเมื่อสภาวะของ รีจิสเตอร์กลับมาเป็น “พร้อมทำงาน” อีกครั้งหนึ่ง ซึ่งจะต้องใช้จำนวนสัญญาณนาฬิกาเท่ากับ n+1 พัลส์ โดย n เท่ากับจำนวนบิตของรีจิสเตอร์

ดังนั้นถ้าหาก ADC แบบซักเซสซีฟแอปพร็อกซิเมชั่นขนาด 8 บิต ตามตัวอย่างที่อธิบายมานี้ใช้สัญญาณนาฬิกาความถี่ 1.25 MHz เวลาที่ใช้ทั้งหมดในการแปลงสัญญาณจะคำนวณได้ดังนี้

1. คำนวณคาบเวลาของสัญญาณนาฬิกา

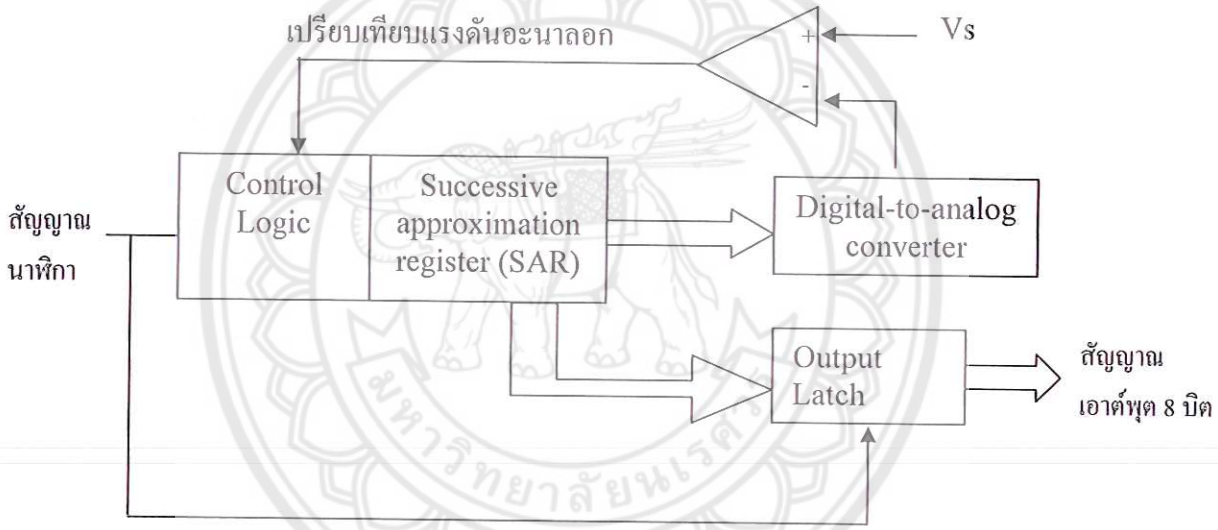
$fclk = 1.25\ MHz = 1.25 \times 10^6$

$T = 1/(1.25 \times 10^6) = 0.8$ ไมโครวินาที

2. จำนวนสัญญาณนาฬิกาทั้งหมดที่ใช้ในการแปลงเท่ากับ n+1, n มีค่าเท่ากับ 8 เนื่องจากมีจำนวนสัญญาณนาฬิกาที่ใช้ทั้งหมดจึงเท่ากับ 8+1 = 9

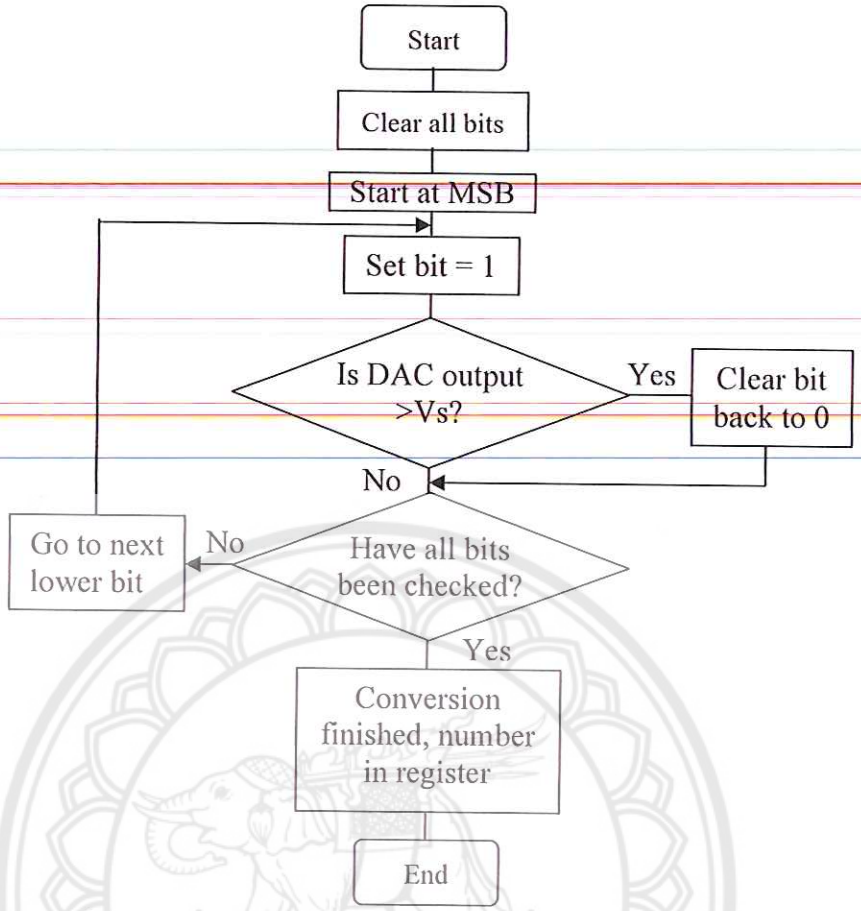
3. เวลาทั้งหมดที่ใช้เท่ากับ $9 \times 0.8 = 7.2$ ไมโครวินาที

จะเห็นว่าวงจร ADC แบบซักเซสซีฟแอปพร็อกซิเมชั่นมีความเร็วในการทำงานสูงพอสมควรเหมาะอย่างยิ่งในการนำไปใช้กับไมโครคอนโทรลเลอร์ขนาดกลางอย่าง MCS-51



รูปที่ 2.3 แสดงตัวแปลงสัญญาณแบบประมาณค่าใกล้เคียง [6]

จากคำอธิบายวิธีการแปลงสัญญาณที่แบบประมาณค่าใกล้เคียง เราสามารถนำมาเขียนเป็นแผนภูมิ (Flow chart) ได้ดังรูปที่ 2.4



รูปที่ 2.4 แสดงแผนภูมิวิธีการแปลงสัญญาณแบบประมาณค่าใกล้เคียง [7]

2.2.3 ความเที่ยงตรงของวงจร ADC

เป็นการเปรียบเทียบแรงดันอะนาลอกของวงจร ADC กับแรงดันที่ควรจะเกิดขึ้นจริง ยกตัวอย่างที่ข้อมูลดิจิตอลสูงสุดของวงจร ADC ขนาด 8 บิต เมื่อเทียบเป็นแรงดันอะนาลอกควรมีค่าเท่ากับ 5.0000 V แต่จากการคำนวณในตัวอย่างก่อนหน้านี้ถ้าทั้ง 8 บิตเป็นลอจิก '1'แล้วจะได้ค่าแรงดัน 4.9804 V นั่นคือเกิดความผิดพลาดไป 0.0195 V หรือ 1.95 mV แต่การบวกค่าความเที่ยงตรงของวงจร ADC มักระบุเป็นจำนวนที่เทียบกับ VLSB (Volt Least Significant Bit) ดังนั้นในวงจร ADC ขนาด 8 บิต ที่ยกเป็นตัวอย่างนี้จึงมีค่าความเที่ยงตรง (หรือบางทีเรียกเป็นค่าความผิดพลาด) เป็น ±1/2LSB

2.2.4 ค่าเวลาในการแปลงสัญญาณ (conversion time)

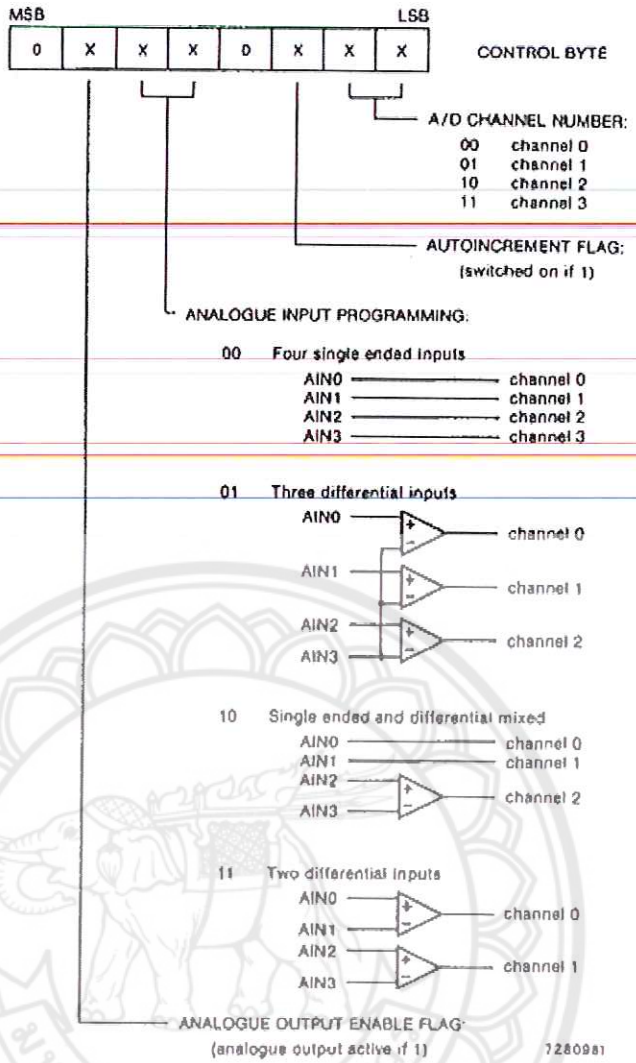
เป็นค่าของเวลาทั้งหมดที่วงจร ADC แบบวงจรนับแรมป์และแบบปรับค่าอย่างต่อเนื่องใช้ในการแปลงสัญญาณอะนาลอกเป็นดิจิตอลจนเสร็จสิ้น พารามิเตอร์ตัวนี้มักจะปรากฏในคุณสมบัติของไอซีที่ทำงานเป็นวงจร ADC เมื่อไอซีแปลงสัญญาณเสร็จสิ้นลง จะส่งสัญญาณที่เรียกว่า EOC (End of conversion) ออกมา ค่าเวลาในการแปลงสัญญาณของวงจร ADC จะขึ้นอยู่กับจำนวนบิต

ของวงจร, ค่าความถี่ของสัญญาณนาฬิกาที่ใช้ในการแปลงสัญญาณและขนาดของสัญญาณอะนาลอกอินพุต

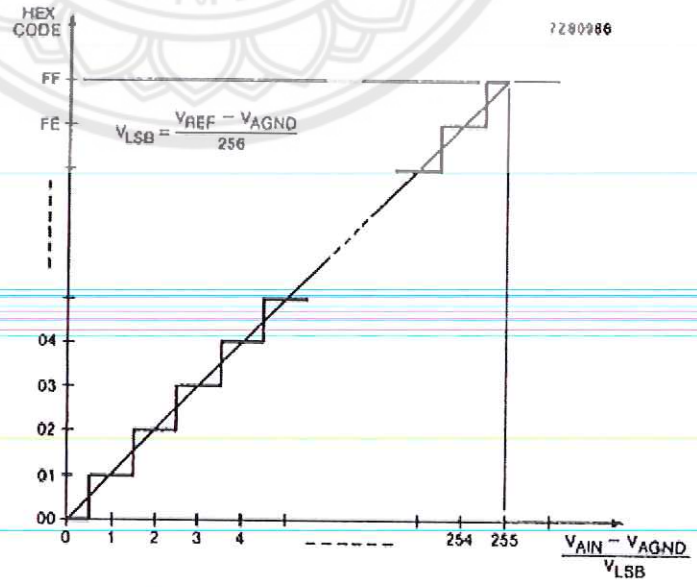
2.3 ข้อมูลเบื้องต้นของ PCF8591

ในการทดลองนี้จะใช้ไอซี ADC ที่มีความสามารถสูงเบอร์ PCF8591 เนื่องจากในตัวมันมีวงจร ADC แบบปรับค่าอย่างต่อเนื่องขนาด 8 บิต สูงถึง 4 ช่อง ทั้งยังมีวงจร DAC อีก 1 ช่องด้วย ระบบการเชื่อมต่อเป็นแบบบัส I^2C ทำให้ใช้สายสัญญาณเพียง 2 เส้น ทั้งยังสามารถต่อพ่วงกันได้สูงสุด 8 ตัว ทำให้ได้วงจร ADC รวมกันสูงสุดถึง 32 ช่อง และวงจร DAC สามารถนำไปประยุกต์ใช้งานได้อย่างกว้างขวาง มีรายละเอียดคุณสมบัติทางเทคนิคดังนี้ [5]

1. ทำงานโดยใช้แหล่งจ่ายไฟชุดเดียว
2. ทำงานที่แรงดัน 2.5 V ถึง 6 V
3. กินกระแสขณะอยู่ในสภาวะแสตนด์บายต่ำ
4. ติดต่อกับไมโครคอมพิวเตอร์หรือไมโครคอนโทรลเลอร์ผ่านระบบบัส I^2C
5. เลือกตำแหน่งแอดเดรสทางฮาร์ดแวร์จากขา A0, A1, A2 สามารถพ่วงกันได้สูงสุดถึง 8 ตัว
6. อัตราการสุ่มข้อมูล (Sampling Rate) ขึ้นอยู่กับความเร็วของสัญญาณนาฬิกาบนบัส I^2C
7. วงจรแปลงสัญญาณอะนาลอกเป็นดิจิตอล (ADC) สามารถรับสัญญาณอะนาลอกได้ 4 ช่อง ทั้งยังเลือกได้ว่าให้ทำงานแบบแยกช่องหรือทำงานเป็นวงจรดิฟเฟอเรนเชียล ดังแสดงในรูปที่ 2.6 ในที่นี้จะเลือกการทำงานแบบ single end inputs หรือแบบแยกช่อง ซึ่งมีกราฟคุณลักษณะของแรงดันเป็นดังรูปที่ 2.7
8. การอ่านค่าสามารถกำหนดให้เลื่อนช่องอินพุตโดยอัตโนมัติได้
9. สัญญาณอะนาลอกมีระดับแรงดันตั้งแต่ V_{SS} ไปจนถึง V_{DD}
10. วงจรแปลงสัญญาณอะนาลอกเป็นดิจิตอลเป็นแบบปรับค่าอย่างต่อเนื่องขนาด 8 บิต
11. มีวงจรแปลงสัญญาณดิจิตอลเป็นอะนาลอกขนาด 8 บิต 1 ช่อง

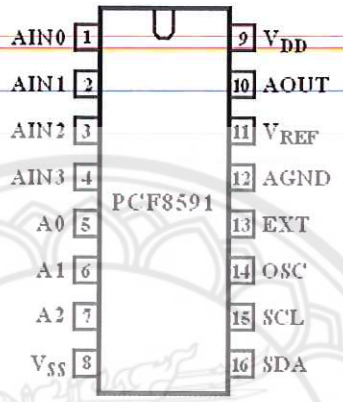


รูปที่ 2.5 แสดงรายละเอียดข้อมูลควบคุมที่เขียนลงในรีจิสเตอร์ควบคุมภายในไอซี PCF8591 [8]



รูปที่ 2.6 แสดงคุณลักษณะของ ADC แบบ Single end inputs

PCF8591 ทำหน้าที่เป็นไอซีแปลงสัญญาณอะนาลอกเป็นดิจิตอลขนาด 8 บิต 4 ช่องและทำหน้าที่เป็นไอซีแปลงสัญญาณดิจิตอลเป็นอะนาลอกได้เช่นกัน ด้วยการควบคุมผ่านระบบบัส I^2C ทำให้สามารถต่อพ่วงไอซี PCF8591 ได้สูงสุดถึง 8 ตัว รองรับการอ่านค่าสัญญาณอะนาลอกอินพุตได้สูงสุดถึง 32 ช่องและสามารถส่งสัญญาณอะนาลอกเอาต์พุตสูงสุดได้ถึง 8 ช่องด้วยการกำหนดแอดเดรสจากขา A0, A1 และ A2 การจัดขาของ PCF8591 แสดงในรูปที่ 2.8 ส่วนรายละเอียดตำแหน่งขาต่างๆ ดังนี้



รูปที่ 2.7 Pinning Diagram ของ PCF8591 [9]

ขา AIN0-AIN3 (ขา 1-4) เป็นขาอินพุตสำหรับป้อนสัญญาณอะนาลอกที่ต้องการแปลงค่า

ขา A0-A2 (ขา 5-7) เป็นขาสำหรับกำหนดข้อมูลแอดเดรสทางฮาร์ดแวร์ ปกติต่อลงกราวด์ แต่ถ้ามีการใช้งาน PCF8591 มากกว่า 1 ตัว ต้องกำหนดการต่อขา A0-A2 ของ PCF8591 ให้ไม่ตรงกัน จึงทำให้สามารถต่อใช้งานได้สูงสุด 8 ตัว

ขา V_{SS} (ขา 8) เป็นขาต่อกราวด์

ขา SDA, SCL (ขา 9 และ 10) เป็นขาเชื่อมต่อระบบบัส I^2C

ขา OSC (ขา 11) เป็นขาสำหรับต่อกับสัญญาณนาฬิกาภายนอกเมื่อขา EXT ต่อกับไฟ +5V และจะทำงานเป็นขาเอาต์พุตสัญญาณนาฬิกาถ้าขา EXT ต่อลงกราวด์

ขา EXT (ขา 12) เป็นขาสำหรับเลือกแหล่งกำเนิดสัญญาณนาฬิกา ถ้าต่อไฟ +5V จะเป็นการเลือกใช้สัญญาณนาฬิกาจากภายนอก โดยต่อสัญญาณนาฬิกาเข้าที่ขา OSC ถ้าต่อขานี้ลงกราวด์จะเป็นการเลือกใช้สัญญาณนาฬิกาจากภายใน

ขา AGND (ขา 13) เป็นขากราวด์ของแรงดันอ้างอิง ปกติต่อลงกราวด์

ขา V_{REF} (ขา 14) เป็นขาสำหรับป้อนแรงดัน ปกติต่อเข้าไฟเลี้ยง +5V

ขา AOUT (ขา 15) เป็นขาเอาต์พุตของวงจรแปลงสัญญาณดิจิตอลเป็นอะนาลอก

ขา V_{DD} (ขา 16) เป็นขาต่อไฟเลี้ยง จ่ายได้ตั้งแต่ +2V ถึง +6V ปกติใช้ +5V

2.4 รายละเอียดฟังก์ชั่นต่างๆ ของ PCF8591

2.4.1 ตำแหน่งแอดเดรส

ในระบบบัส I^2C การติดต่อกับอุปกรณ์แต่ละตัวต้องระบุแอดเดรสของอุปกรณ์เหล่านั้นอย่างชัดเจน ถ้าเป็นอ้างถึงแบบ 7 บิต ข้อมูลกำหนดแอดเดรส 4 บิตบนจะเป็นค่าแอดเดรสเฉพาะของอุปกรณ์ตัวนั้นๆ ที่กำหนดมาจากผู้ผลิต ผู้ใช้งานไม่สามารถเปลี่ยนแปลงได้ สำหรับไอซี PCF8591 จะมีค่าเท่ากับ 1001(ฐานสอง) ข้อมูล 3 บิตถัดมาจะเป็นค่าของแอดเดรสที่ผู้ใช้งานสามารถกำหนดได้ทางฮาร์ดแวร์เพื่อเลือกไอซี PCF8591 ที่ต้องการติดต่อด้วยในกรณีที่มีการต่อใช้งาน PCF8591 มากกว่า 1 ตัว โดยสามารถต่อพ่วงได้สูงสุดเท่ากับ 111 (ฐานสอง) = 8 ตัว ยกตัวอย่างการกำหนดแอดเดรส เช่น ตัวแรกเรากำหนดให้มีแอดเดรสเป็น 000 (ฐานสอง) ตัวที่สองกำหนดให้เป็น 001 (ฐานสอง) เป็นต้น ในการกำหนดแอดเดรสของไอซี PCF8591 จะต้องมีแอดเดรสไม่ซ้ำกัน ในที่นี้เราต่อใช้งานเพียงตัวเดียวแอดเดรสส่วนนี้จะถูกกำหนดเป็น 000 (ฐานสอง ส่วนบิต LSB จะใช้ในการกำหนดว่าต้องการอ่านหรือเขียนข้อมูลกับไอซีตัวนั้นๆ

2.4.2 ข้อมูลควบคุมของ PCF8591

หลังจากส่งข้อมูลกำหนดแอดเดรสให้แก่ PCF8591 แล้ว ต้องส่งข้อมูลควบคุมตามไปด้วย เพื่อกำหนดคุณสมบัติของวงจรแปลงสัญญาณอะนาลอกเป็นดิจิตอล และวงจรแปลงสัญญาณดิจิตอลเป็น อะนาลอกภายใน PCF8591 โดยมีรายละเอียดของข้อมูลในแต่ละบิตดังนี้

บิต 6 ของข้อมูลควบคุมใช้สำหรับการเอ็นเอเบิลขาอะนาลอกเอาต์พุต เมื่อต้องการเอ็นเอเบิลต้องกำหนดให้ขานี้เป็น “1”

บิต 4 และบิต 5 ของข้อมูลควบคุมใช้สำหรับการกำหนดรูปแบบของสัญญาณอะนาลอกอินพุตที่ป้อนให้แก่ PCF8591

บิต 2 ใช้สำหรับเลือกรูปแบบการอ่านข้อมูลจากขาอินพุตอะนาลอกว่าจะเป็นการอ่านจากเพียงอินพุตเดียวหรืออ่านแบบเรียงลำดับทุกอินพุต ถ้าต้องการเลือกให้อ่านแบบเรียงลำดับต้องกำหนดให้บิตนี้เป็น “1”

บิต 0 และบิต 1 ใช้สำหรับกำหนดช่องของอินพุตอะนาลอกที่ต้องการอ่าน ถ้ากำหนดให้บิต 2 เป็น “1” เพื่อกำหนดการอ่านให้เป็นแบบเรียงลำดับอินพุต หลังจากอ่านค่าของบิต 0 และบิต 1 แล้ว ในการอ่านค่าครั้งต่อไปจะเป็นการอ่านค่าอินพุตจากช่องถัดไป

ข้อมูลควบคุมทั้งหมดจะถูกเก็บไว้ในรีจิสเตอร์ควบคุมภายใน PCF8591

เมื่อจ่ายไฟให้แก่ PCF8591 ครั้งแรก บิตต่างๆ ของข้อมูลภายในรีจิสเตอร์ควบคุมจะเป็น “0”

2.4.3 ออสซิลเลเตอร์ของ PCF8591

วงจรออสซิลเลเตอร์ภายใน PCF8591 จะสร้างสัญญาณนาฬิกาสำหรับการแปลงสัญญาณอะนา-ลอกเป็นดิจิตอล เมื่อต้องการใช้วงจรออสซิลเลเตอร์ภายใน ขา EXT ต้องต่อลงกราวด์ (ความถี่ในการทำงานของ ADC เป็น 11.1 kHz) ถ้าต้องการใช้ออสซิลเลเตอร์จากภายนอกขา EXT ต้องต่อเข้ากับไฟบวก และป้อนสัญญาณนาฬิกาเข้าที่ขา OSC ของ PCF8591 โดยความถี่ของสัญญาณนาฬิกาสูงสุดที่ป้อนให้กับออสซิลเลเตอร์เท่ากับ 1.25 MHz

2.4.4 การอ่านค่าข้อมูลอินพุตอะนาลอกของ PCF8591

มีลำดับขั้นตอนดังนี้

1. เตรียมข้อมูลกำหนดแอดเดรส โดยในที่นี้กำหนดแอดเดรสของ PCF8591 ไว้ที่ 000 (ขา A0, A1, A2 ต่อลงกราวด์ทั้งหมด) และให้ทำงานในโหมดเขียนข้อมูล (ป้อนข้อมูลลอจิก "0" ให้แก่บิต R/W) เรียกโปรแกรมย่อยการติดต่อกับอุปกรณ์สเลฟ
2. ส่งข้อมูลควบคุมไปยัง PCF8591
3. ส่งสัญญาณ STOP
4. เรียกโปรแกรมย่อยการติดต่อกับอุปกรณ์สเลฟ
5. ส่งข้อมูลกำหนดแอดเดรสอีกครั้งโดยครั้งนี้กำหนดให้เป็นโหมดอ่านข้อมูล (ส่งลอจิก "1" ให้แก่บิต R/W) เพื่อเริ่มต้นอ่านข้อมูลจากช่องสัญญาณอะนาลอกอินพุต
6. อ่านค่าจากขาอินพุตของวงจรแปลงสัญญาณอะนาลอกเป็นดิจิตอลช่องที่ 1
7. หาต้องการอ่านค่าในช่องต่อไปก็ให้เริ่มการติดต่อใหม่ ดังนั้นในการเขียนโปรแกรมเพื่ออ่านค่าต่อเนื่องทั้ง 4 ช่องหรือมากกว่าจึงต้องเขียนโปรแกรมลูปเพื่อกำหนดรอบการทำงาน 4 รอบหรือมากกว่า ก็จะสามารถอ่านค่าได้ครบทุกช่อง

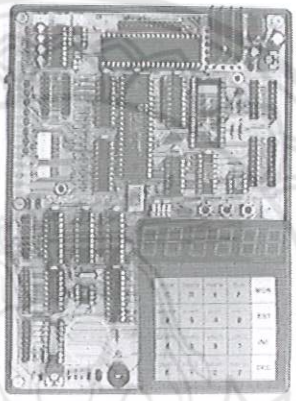
2.4.5 การเขียนข้อมูลไปยังวงจรแปลงสัญญาณดิจิตอลเป็นอะนาลอกของ PCF8591

การเขียนข้อมูลไปยังขาอะนาลอกเอาต์พุตมีข้อแตกต่างจากการอ่านข้อมูลดังนี้

1. เรียกโปรแกรมย่อยการติดต่อกับอุปกรณ์สเลฟ
2. ส่งข้อมูลกำหนดแอดเดรสโดยให้ทำงานในโหมดเขียนข้อมูล (บิต R/W เป็นลอจิก "0")
3. ส่งข้อมูลควบคุม 40H ไปยัง PCF8591 เพื่อเอ็นเอเบิลอะนาลอกเอาต์พุต
4. ส่งข้อมูลไปยังเอาต์พุตอะนาลอก โดยค่าที่ส่งออกไปจะต้องมีค่าอยู่ระหว่าง 0 ถึง255
5. ส่งสภาวะหยุด

2.5 START-C51 v3.0 Microcontroller Single Board

ในโครงงานนี้เราเลือกใช้ START-C51 ของบริษัท ศิลารีเสิร์ช ซึ่งเป็น Single Board เพื่อการเริ่มต้นเรียนรู้เกี่ยวกับไมโครคอนโทรลเลอร์ในตระกูล MCS-51 ที่ง่ายต่อการเรียนรู้และใช้งาน โดยเน้นการเรียนรู้ด้วยภาษาซี [10] START-C51 เป็น Single Board ที่ใช้ไมโครคอนโทรลเลอร์เบอร์ 89C51RD2 ของ Philips บนบอร์ดจะมีทรัพยากรพื้นฐานที่จำเป็นต่อการเรียนรู้ สามารถพัฒนาได้บนบอร์ดหรือพัฒนาผ่านเครื่อง PC ทาง Remote Monitor ได้ มี Function ต่าง ๆ ถึง 16 Functions เพื่อช่วยให้อ่าน, เขียนข้อมูลในหน่วยความจำได้รวมทั้งยังมีพอร์ทสำหรับขยายการใช้งานเพื่อต่อกับอุปกรณ์ต่างๆ ภายนอกอีกด้วย การติดต่อระหว่างอุปกรณ์ต่างๆ ในบอร์ดจะติดต่อผ่าน I^2C รวมทั้ง PCF8591 ด้วย



รูปที่ 2.8 แสดงบอร์ด START-C51

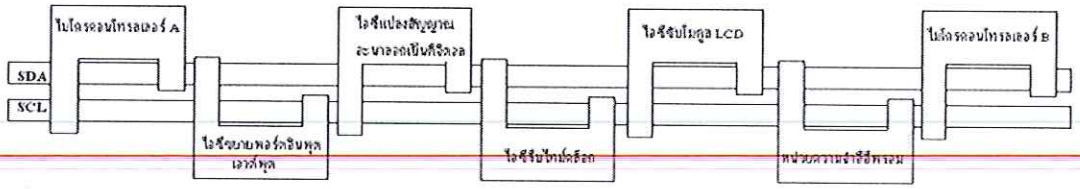
2.6 การติดต่อสื่อสารข้อมูลผ่านระบบบัส I^2C

2.6.1 ความรู้เบื้องต้นเกี่ยวกับ I^2C

I^2C ย่อมาจาก Inter-IC Communication หมายถึง การติดต่อสื่อสารระหว่างไอซี โดยบัส I^2Cได้รับการพัฒนาขึ้นโดยฟิลิปส์ (Philips) ด้วยจุดมุ่งหมายหลักคือ ต้องการให้ไอซีหรือโมดูลสามารถติดต่อ สั่งงาน และควบคุมภายใต้สายสัญญาณเพียง 2 เส้น เส้นหนึ่งคือ สายข้อมูล อีกเส้นหนึ่งคือ สายสัญญาณนาฬิกาที่ใช้ในการกำหนดจังหวะการทำงาน การต่อร่วมกันของอุปกรณ์บนบัส I^2C

ทำได้ง่ายมาก เพียงต่อสายข้อมูลสายสัญญาณนาฬิกาของอุปกรณ์แต่ละตัวขนานหรือพ่วงกันไป ส่วนการกำหนดแอดเดรสหรือตำแหน่งสำหรับติดต่ออุปกรณ์แต่ละตัว จะใช้รหัสป้อนข้อมูลและการกำหนดสภาวะลอจิกที่ขาแอดเดรสของอุปกรณ์แต่ละตัว

สายข้อมูลบนบัส I^2C มีชื่อเรียกอย่างเป็นทางการว่า สายข้อมูลอนุกรมหรือ SDA (Serial Data line) ส่วนสายสัญญาณนาฬิกามีชื่อเรียกว่า สายสัญญาณนาฬิกาอนุกรมหรือ SCL (Serial Clock line) ในการอธิบายต่อไปนี้จะเรียกสายสัญญาณทั้งสองว่า สาย SDA และ SCL

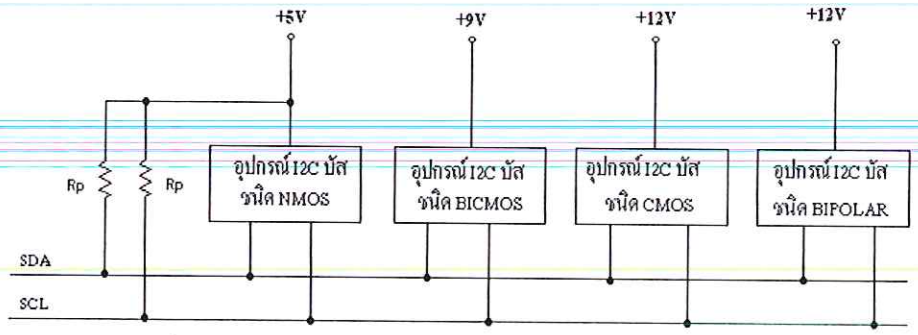


รูปที่ 2.9 ผังแสดงการเชื่อมต่อของอุปกรณ์ต่างๆ บนระบบบัส I^2C

2.6.2 คุณสมบัติโดยทั่วไปของบัส I^2C

สาย SDA และ SCL เป็นสัญญาณ 2 ทิศทาง (bi-directional line) ต้องมีการต่อตัวต้านทานพูลอัปกับแรงดัน +5V ไว้ตลอดเวลา เพื่อให้สายมีสถานะลอจิกสูงในขณะที่ไม่มีการติดต่อใช้งาน ทั้งยังช่วยในการป้องกันสัญญาณรบกวนที่อาจมีเข้ามาในสายสัญญาณทั้งสอง วงจรเอาต์พุต ของอุปกรณ์ที่ต่ออยู่บนบัส I^2C ต้องมีลักษณะเป็นวงจรเดรนเปิด (open-drain) หรือคอลเล็คเตอร์เปิด (open-collector) อัตราการถ่ายทอดข้อมูลบนบัส I^2C สูงถึง 100 กิโลบิตต่อวินาทีในโหมดปกติ (standard mode) และสูงถึง 400 กิโลบิตต่อวินาทีในโหมดความเร็วสูง (fast mode) อุปกรณ์ที่ต่อร่วมอยู่บนบัส I^2C จะต้องมีค่าความจุไฟฟ้ารวมที่เกิดขึ้นระหว่างสาย SDA และ SCL ไม่เกิน 400pF การเข้าถึงอุปกรณ์บนบัส I^2C ใช้ข้อมูลสำหรับการเข้าถึง 2 ค่า คือ 7 บิต (7-bit addressing) หรือ 10 บิต (10-bit addressing)

ข้อเด่นอีกประการหนึ่งของบัส I^2C คือ สามารถเชื่อมต่ออุปกรณ์ที่ใช้ไฟเลี้ยงไม่เท่ากันให้สามารถติดต่อสื่อสารกันได้ โดยอุปกรณ์บนบัส I^2C ตัวหนึ่งอาจใช้ไฟเลี้ยง +5V ในขณะที่อีกตัวหนึ่งใช้ไฟเลี้ยง +12V การต่อร่วมกันบนบัส I^2C สามารถกระทำได้ในลักษณะเดียวกันกับกรณีที่อุปกรณ์ทั้งสองใช้ไฟเลี้ยงเท่ากัน กล่าวคือ ให้ต่อสาย SDA และ SCL ของอุปกรณ์แต่ละตัวเข้าด้วยกัน และต้องต่อตัวต้านทานพูลอัป (R_p) เข้ากับแรงดัน +5V ไว้ด้วยเสมอ ดังแสดงในรูปที่ 2.11 ซึ่งเงื่อนไขนี้จะต้องเป็น Open drain สำหรับ FET และ Open collector สำหรับ BJT เท่านั้น



รูปที่ 2.10 การต่อตัวต้านทานพูลอัปบนสายสัญญาณในระบบบัส I^2C

ในกรณีที่อาจมีแรงดันไฟกระชากขนาดใหญ่ปะปนเข้ามาในบัส I^2C ที่ขา SDA และ SCL ของอุปกรณ์แต่ละตัวต้องต่อตัวต้านทานอนุกรมกับขา SDA และ SCL เรียกว่า R_S ก่อนเข้าสู่บัส I^2C

2.6.3 หลักการทำงานของ I^2C

บัส I^2C ประกอบด้วยสายสัญญาณ 2 เส้น ดังที่ได้กล่าวมาแล้วคือ SDA และ SCL อุปกรณ์ที่ต่อพ่วงบนบัสสามารถมีได้มากมาย ดังนั้นจึงต้องมีการกำหนดรูปแบบของการติดต่อบนบัส หรือเรียกว่า โปรโตคอล (protocol) เพื่อให้ผู้ใช้งานทราบว่า ขณะนี้อุปกรณ์ใดติดต่อกันอยู่ และอุปกรณ์ตัวใดเป็นตัวรับหรือตัวส่ง ต่อไปนี้จะอธิบายลักษณะ หน้าที่ และนิยามของอุปกรณ์ที่ต่ออยู่บนบัส I^2C เพื่อเป็นข้อตกลงพื้นฐานก่อนที่จะอธิบายการทำงานของบัส I^2C ต่อไป

อุปกรณ์ที่เป็นผู้สร้างข้อมูลหรือส่งข้อมูล เรียกว่า ตัวส่ง (transmitter)

อุปกรณ์ที่เป็นผู้รับข้อมูล เรียกว่า ตัวรับ (receiver) อุปกรณ์บนบัส I^2C สามารถเป็นได้ทั้งตัวรับและตัวส่ง บางอุปกรณ์ทำหน้าที่เป็นตัวรับอย่างเดียว จะไม่มีอุปกรณ์ใดบนบัส I^2C ที่ทำหน้าที่เป็นตัวส่งเพียงอย่างเดียว

อุปกรณ์ที่ทำหน้าที่ควบคุมจังหวะการติดต่อบนบัส I^2C เรียกว่า มาสเตอร์ (master)

อุปกรณ์ที่ถูกควบคุมหรืออุปกรณ์ที่ต่อพ่วงเข้าไปบนบัส I^2C เรียกว่า สเลฟ (slave)

ข้อกำหนด 2 ประการสำคัญของการติดต่อบนบัส I^2C คือ

1. การถ่ายทอดข้อมูลขณะเกิดขึ้นได้เมื่อบัสว่างเท่านั้น

2. ในระหว่างการถ่ายทอดข้อมูลเมื่อใดก็ตามที่สาย SCL มีสถานะเป็นลอจิกสูง สายข้อมูลต้องรักษาข้อมูลไว้ อย่าให้เกิดการเปลี่ยนแปลงขึ้นเด็ดขาด มิฉะนั้น สัญญาณที่เกิดขึ้นจะได้รับการแปลความหมายเป็นสัญญาณควบคุมแทน

2.6.4 สภาวะที่เกิดขึ้นบนบัส I^2C

มีด้วยกัน 5 สภาวะดังนี้

1. บัสว่าง (Bus not busy) สภาวะนี้เกิดขึ้นเมื่อสถานะลอจิกบนสาย SDA และ SCL เป็นลอจิกสูงทั้งคู่ นั่นหมายความว่า การถ่ายทอดข้อมูลสามารถเริ่มต้นขึ้นได้

2. เริ่มต้นการถ่ายทอดข้อมูล (start data transfer) เกิดขึ้นเมื่อสาย SDA มีการเปลี่ยนแปลงระดับลอจิกจากสูงไปต่ำ ในขณะที่สาย SCL มีสถานะลอจิกสูง เรียกสภาวะที่เกิดขึ้นนี้ว่า สภาวะเริ่มต้น (START)

3. การหยุดการถ่ายทอดข้อมูล (stop data transfer) เกิดขึ้นเมื่อสาย SDA มีการเปลี่ยนแปลงระดับลอจิกจากต่ำไปสูง ในขณะที่สาย SCL มีสถานะลอจิกสูง เรียกสภาวะที่เกิดขึ้นนี้ว่า สภาวะหยุด (STOP)

4. ข้อมูลดำรงอยู่บนบัส (data valid) สภาวะนี้เกิดขึ้นถัดจากสภาวะเริ่มต้น โดยสถานะที่เกิดขึ้นบนสาย SDA ก็คือข้อมูลที่ทำการถ่ายทอด เมื่อสาย SCL เป็นลอจิกสูง สถานะที่สาย SDA ต้องคงที่ เพื่อให้อุปกรณ์รับรู้ข้อมูลในจังหวะนั้นว่า เป็น “0” หรือ “1” ข้อมูลอาจเกิดการ

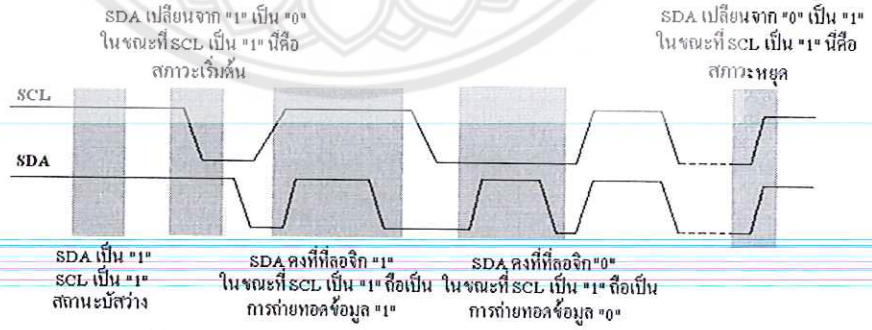
เปลี่ยนแปลงได้ในขณะที่สาย SCL เป็นลอจิกต่ำ แต่เมื่อใดก็ตามที่ต้องการให้เกิดการถ่ายทอดข้อมูลอย่างสมบูรณ์ สถานะลอจิกที่ขา SDA ต้องคงที่ตลอดช่วงเวลาที่สาย SCL มีสถานะลอจิกสูง หากเกิดการเปลี่ยนแปลงสถานะลอจิกในขณะที่สาย SCL มีลอจิกสูงอยู่นั้น อุปกรณ์มาสเตอร์ที่ทำการควบคุมการถ่ายทอดข้อมูลจะแปลความหมายเป็นสภาวะหยุดหรือสภาวะเริ่มต้นก็ได้ ทำให้ข้อมูลที่ทำการถ่ายทอดนั้นเกิดความผิดพลาดขึ้น

5. รับรู้ข้อมูล (acknowledge) เกิดขึ้นหลังจากที่การถ่ายทอดข้อมูลจากตัวส่งมายังตัวรับเกิดขึ้นอย่างสมบูรณ์ โดยตัวส่งจะทำการส่งข้อมูลมา 1 บิต เรียกว่า บิตรับรู้ (acknowledge bit) มีสถานะเป็นลอจิกสูง หลังจากส่งข้อมูลมาครบถ้วน ส่วนอุปกรณ์มาสเตอร์จะทำการส่งสัญญาณรับรู้พิเศษซึ่งสัมพันธ์กับสัญญาณนาฬิกา เพื่อตอบสนองบิตรับรู้ที่ส่งมาจากตัวส่ง ทางด้านตัวรับจะส่งบิตรับรู้ที่มีสถานะลอจิกต่ำลงบนบัส อุปกรณ์สเลฟถูกอ้างถึงในการติดต่อหรือกำลังติดต่ออยู่ในขณะนั้นก็จะกำเนิดบิตรับรู้เพื่อตอบสนองให้ทราบว่าได้รับข้อมูลในแต่ละไบต์เรียบร้อยแล้ว

ในรูปที่ 2.12 เป็นไดอะแกรมเวลาที่แสดงถึงการเกิดสภาวะต่างๆ บนบัส I^2C ไม่ว่าจะเป็นสภาวะบัสว่าง, เริ่มต้น, ถ่ายทอดข้อมูล, รับรู้ และหยุดการถ่ายทอดข้อมูล

2.6.5 การทำงานบนบัส I^2C

ก่อนที่จะเริ่มต้นการถ่ายทอดข้อมูลระหว่างอุปกรณ์ต่างๆ ที่ต่ออยู่บนบัส ต้องมีการอ้างถึงเสียก่อน โดยการอ้างถึงอุปกรณ์บนบัส I^2C นั้นจะใช้การอ้างถึงแบบ 7 บิต หรือ 10 บิต ในกรณีที่มีอุปกรณ์ต่ออยู่บนบัสไม่มาก ใช้การอ้างถึงแบบ 7 บิตก็เพียงพอ แต่ถ้ามีอุปกรณ์ต่ออยู่บนบัสมากกว่า 127 แอดเดรส จำเป็นต้องใช้การอ้างถึงแบบ 10 บิต หลังจากที่ติดต่ออุปกรณ์แต่ละตัวได้เรียบร้อยแล้ว ก็จะเริ่มต้นการถ่ายทอดข้อมูลต่อกันไป ดังนั้นหัวใจสำคัญในอันดับแรกของการทำงานบนบัส I^2C คือการอ้างถึงอุปกรณ์แต่ละตัว ในที่นี้จะอธิบายการอ้างถึงทั้ง 2 รูปแบบ



รูปที่ 2.11 ไดอะแกรมเวลาแสดงสถานะต่างๆ ในบัส I^2C

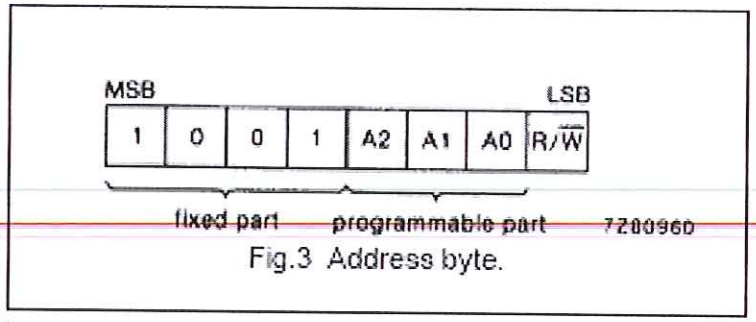


Fig.3 Address byte.

รูปที่ 2.12 รูปแบบของข้อมูลกำหนดแอดเดรสที่ใช้ในการอ้างถึงแบบ 7 บิต

2.6.6 การอ้างถึงแบบ 7 บิต (7-bit addressing)

ข้อมูลไบต์แรกที่เกิดขึ้นหลังจากสภาวะเริ่มต้นคือ ข้อมูลที่ใช้ในการอ้างถึงอุปกรณ์ที่ต้องการติดต่อ หรือ ข้อมูลกำหนดแอดเดรส โดยมีรูปแบบแสดงในรูปที่ 2.13 ใน 7 บิตบนรวมทั้งบิต MSB ด้วยจะเป็นข้อมูลแอดเดรสของอุปกรณ์สเลฟที่ต้องการติดต่อ โดยแบ่งเป็น บิตกำหนดแอดเดรสคงที่ (fixed address bit) จำนวน 4 บิต ซึ่งข้อมูลนี้อุปกรณ์แต่ละตัวจะถูกกำหนดมาจากผู้ผลิต ไม่สามารถเปลี่ยนแปลงแก้ไขได้ ถัดมาอีก 3 บิตเป็นบิตกำหนดแอดเดรสที่สามารถโปรแกรมได้ (programmable address bit) โดยผู้ใช้งานต้องกำหนดสถานะลอจิกให้แก่ขา A0-A2 ของอุปกรณ์ที่มีการเชื่อมต่อแบบบัส I^2C ส่วนในบิต LSB เป็นบิตที่ใช้กำหนดการอ่านหนังหรือเขียนข้อมูลกับอุปกรณ์สเลฟตัวนั้นๆ หากบิต LSB เป็น “0” หมายถึงต้องการเขียนข้อมูลไปยังอุปกรณ์นั้น ถ้าเป็น “1” จะเป็นการอ่านข้อมูลจากอุปกรณ์สเลฟ

ข้อมูลในไบต์ต่อมาคือ ข้อมูลควบคุม (Control byte) ในอุปกรณ์แต่ละตัวมีการกำหนดข้อมูลควบคุมที่แตกต่างกันไป ยกตัวอย่าง ไอซีขยายพอร์ตมีข้อมูลควบคุมที่ใช้กำหนดว่า บิตใดเป็นอินพุต บิตใดเป็นเอาต์พุต ในขณะที่ไอซี ADC/DAC ต้องการข้อมูลควบคุมเพื่อกำหนดให้ทำงานเป็นวงจร ADC หรือ DAC เป็นต้น

ข้อมูลในไบต์ต่อมาคือ ข้อมูลที่ทำการถ่ายทอดจริง (data)

หลังจากที่มีการถ่ายทอดข้อมูลในแต่ละไบต์ อุปกรณ์สเลฟที่ได้รับการติดต่อต้องส่งสัญญาณรับรู้ตอบกลับมาด้วยทุกครั้ง เพื่อให้กระบวนการถ่ายทอดข้อมูลสามารถดำเนินต่อไปได้

2.6.7 การอ้างถึงแบบ 10 บิต

ในการอ้างอิงถึงแบบนี้ ยังคงใช้รูปแบบข้อมูลอนุกรมที่เหมือนกัน 7 บิต หากแต่จะมีข้อมูลเพิ่มเติมขึ้นมาเล็กน้อย โดยในข้อมูลไบต์แรกหลังจากเกิดสภาวะเริ่มต้น ต้องกำหนดให้ 5 บิตบนมีข้อมูลเป็น 11110 ส่วนอีก 2 บิตถัดมาเป็นบิตแอดเดรสของอุปกรณ์ที่ต้องการติดต่อ ในบิต LSB ของข้อมูลไบต์แรกยังคงเป็นการกำหนดว่าต้องการอ่านหรือเขียนข้อมูลกับอุปกรณ์สเลฟตัวที่ต้องการติดต่อด้วย ข้อมูลไบต์มาเป็นข้อมูลแอดเดรสในไบต์ที่ 2 ของอุปกรณ์ที่ต้องการติดต่อด้วย ข้อมูลไบต์ถัดไปจึงเป็นข้อมูลควบคุม ข้อมูลหลังจากนั้นก็จะเป็นข้อมูลจริงที่ใช้ในการติดต่อ

เช่นเดียวกับการอ้างถึงแบบ 7 บิต หลังจากถ่ายทอดข้อมูลครบทุกไบต์ ต้องมีสภาวะรับรู้เกิดขึ้น เพื่อให้กระบวนการถ่ายทอดข้อมูลสามารถดำเนินต่อไปได้ การต่ออุปกรณ์ระบบบัส I^2C กับไมโครคอนโทรลเลอร์ MCS-51

สามารถทำได้ง่ายมาก เพียงใช้ขาพอร์ต 2 ขา โดยกำหนดให้ขาหนึ่งเป็น SDA และอีกขาหนึ่งเป็น SCL และต่อตัวต้านทานค่าประมาณ 4.7 kΩ พูลอัปที่ขาพอร์ตทั้งสองขา

2.6.8 การเขียนโปรแกรมติดต่อบัส I^2C

เริ่มต้นด้วยการสร้างสภาวะมาตรฐานของบัส I^2C อันประกอบด้วย สภาวะเริ่มต้น, สภาวะสิ้นสุดการส่งข้อมูล, สภาวะหยุด, สัญญาณนาฬิกาบนขา SCL, การเขียนและอ่านข้อมูลกับอุปกรณ์บนระบบบัส I^2C

การสร้างสภาวะเริ่มต้น

1. เมื่อต้องการติดต่อกับบัส I^2C สิ่งแรกที่ต้องทำสำหรับไมโครคอนโทรลเลอร์ซึ่งถือว่าเป็นอุปกรณ์มาสเตอร์คือ การทำให้บัสว่างด้วยการกำหนดให้ขา SCL และขา SDA มีลอจิกเป็น "1" ทั้งคู่

2. จากนั้นทำให้ขา SDA มีลอจิก "0" โดยที่ขา SCL ยังคงเป็นลอจิก "1" อยู่

3. กำหนดให้ขา SCL มีลอจิก "0" ถึงตอนนี้ทั้ง SCL และ SDA มีลอจิกเป็น "0" ทั้งคู่ พร้อมที่จะติดต่อได้แล้ว

การสร้างสภาวะหยุด

1. เมื่อต้องการหยุดส่งข้อมูล ต้องส่งสภาวะหยุดออกไป โดยในตอนแรกต้องกำหนดให้ขา SCL และ SDA เป็นลอจิก "0" ทั้งคู่ก่อน

2. กำหนดให้ขา SCL มีลอจิกเป็น "1" โดย SDA ยังคงมีลอจิกเป็น "0"

3. ทำให้ขา SDA มีลอจิกเป็น "1" ทำให้กลับเข้าสู่บัสว่างอีกครั้ง พร้อมที่จะรับหรือส่งข้อมูลต่อไป

การส่งข้อมูลลอจิก "0" และลอจิก "1"

หลังจากส่งบิตเริ่มต้นแล้ว ลำดับต่อไปคือ ส่งข้อมูลควบคุมซึ่งจะเป็นขบวนของลอจิก "0" และ "1" ดังนี้

1. ทำให้ขา SDA เป็น "0" สำหรับการส่งข้อมูลลอจิก "0"

2. ทำให้ขา SCL เป็น "1" สำหรับการป้อนสัญญาณนาฬิกา ในขณะที่ขา SDA ยังคงเป็น "0"อยู่

3. จากนั้นทำให้ขา SCL กลับมามีสถานะเป็นลอจิก "0" เหมือนเดิม

ในขณะที่มีการส่งข้อมูลลอจิก "1" มีขั้นตอนดังนี้

4. ทำให้ขา SDA มีลอจิกเป็น "1" สำหรับการส่งข้อมูลลอจิก "1"

5. ทำให้ขา SCL เป็น “1” สำหรับการส่งสัญญาณนาฬิกา ในขณะที่ขา SDA ยังคงเป็น “1” อยู่

6. จากนั้นทำให้ขา SCL กลับมามีสถานะเป็นลอจิก “0” เหมือนเดิม

ข้อมูลที่ใช้ในการส่งไปยังขา SDA นั้นจะกำหนดที่แอกคิวมูเลเตอร์แล้วทำการส่งออกไปยังแฟลกทดด้วยการใช้คำสั่งหมุนข้อมูล (RLC A) เพื่อถ่ายทอดต่อไปยังขา SDA ต่อไป

บทที่ 3

การออกแบบการทดลอง

จากที่ได้กล่าวมาแล้วว่าชื้นของดินมีผลต่อค่าความต้านทาน ซึ่งเราก็ได้นำหลักการนี้มาใช้ในการวัดและควบคุมความชื้นในดิน ดังนั้นในบทนี้จะขอกล่าวถึงอุปกรณ์ที่ใช้ในการวัดและควบคุมความชื้นในดินดังต่อไปนี้

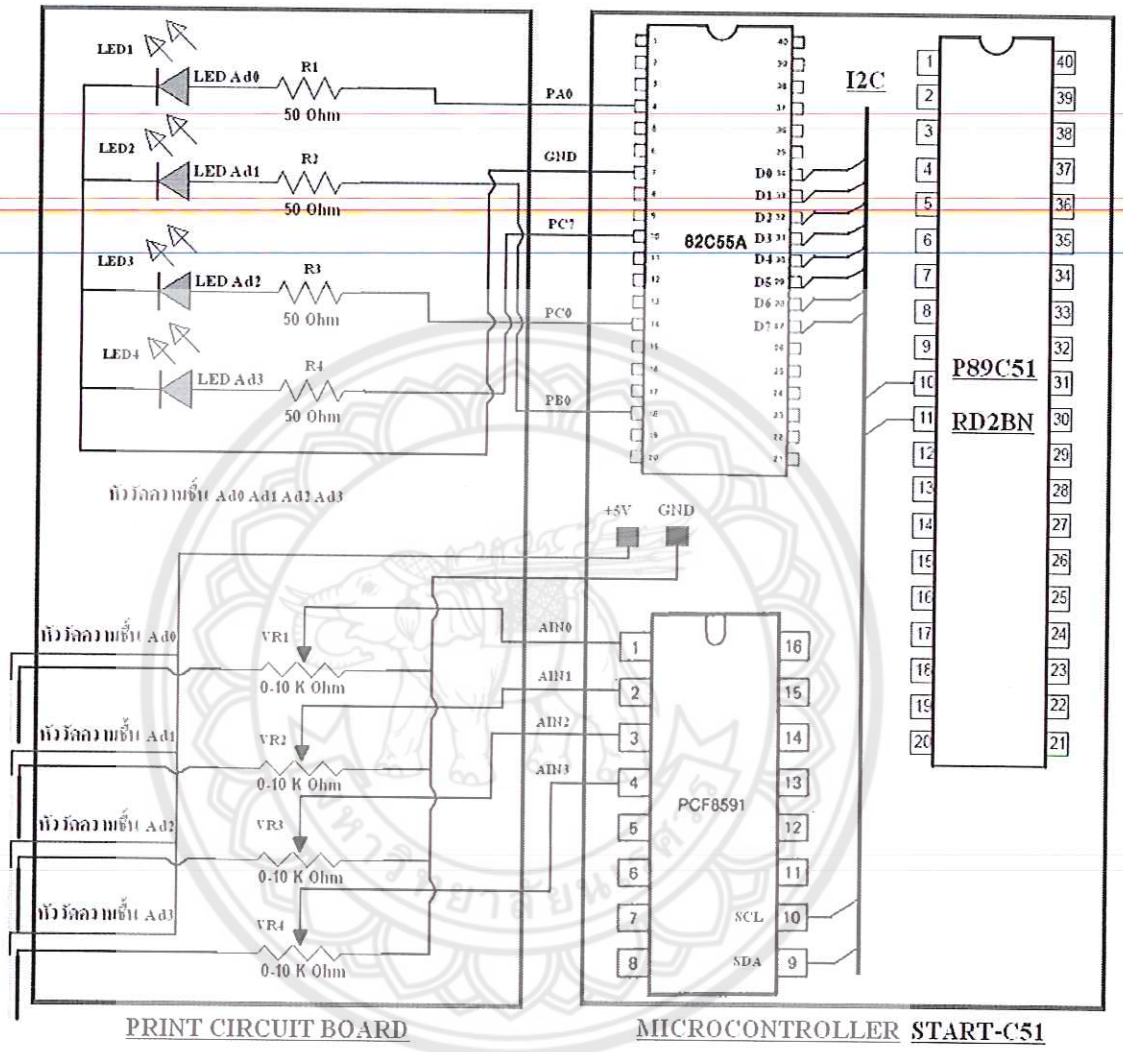
3.1 อุปกรณ์

1. แผ่นปริ้นส์อเนกประสงค์	1	แผ่น
2. ตัวต้านทานปรับค่าได้ 0- 10 กิโลโอห์ม	4	ตัว
3. ตัวต้านทาน 50 โอห์ม	4	ตัว
4. สายแพ 16 pin และ 26 pin อย่างละ	1	เส้น
5. แท่งทองแดง	8	แท่ง
6. สายไฟ	2	เมตร
7. เทอร์มินอล 2 pin	4	ตัว
8. เทอร์มินอล 3 pin	2	ตัว
9. pin คู่ 16 pin และ pin คู่ 26 pin อย่างละ	1	ตัว

3.2 การออกแบบวงจร

จะทำการต่อไฟกระแสตรง +5V จากบอร์ด START-C51 ผ่านแท่งทองแดงแท่งที่ 1 ที่เสียบอยู่ในดิน และไหลผ่านดินไปหาแท่งทองแดงอีกแท่งที่เสียบอยู่ห่างกันประมาณ 2-3 เซนติเมตร ซึ่งดินเปรียบเสมือนตัวต้านทานหนึ่งตัว โดยถ้าดินมีความชื้นมากค่าความต้านทานจะต่ำทำให้กระแสไหลได้ดี เป็นผลให้แรงดันมีค่ามาก แต่ถ้าดินมีค่าความต้านทานมากแสดงว่ามีค่าความชื้นในดินน้อยทำให้กระแสไหลได้น้อย ค่าแรงดันก็จะมีค่าน้อยตามไปด้วย ปัจจัยที่ทำให้ค่าความต้านทานมากหรือน้อยคือน้ำซึ่งอยู่ในดิน แท่งทองแดงแท่งที่สองจะต่อเข้ากับตัวต้านทานปรับค่าได้ แล้วต่อเข้าพอร์ต A/D ในที่นี้เราใช้หัววัดทั้งหมด 4 คู่ เพื่อให้สามารถใช้วัดได้ใน 4 สถานที่ที่แตกต่างกันไป โดยเราจะสามารถตั้งค่าระดับความชื้นได้ทั้งหมด 3 ระดับ หัววัดทั้งสี่คู่จะสามารถตั้งค่าความชื้นที่ต่างกันได้ เราไม่จำเป็นต้องใช้หัววัดทั้งหมด ถ้าหัววัดคู่ใดปล่อยลอยค่าแรงดันจะเป็นศูนย์ หัววัดทั้งสี่คู่จะมีการไหลของกระแสผ่านจากหัววัดหัวหนึ่งถึงหัววัดอีกหัว ผ่านตัวต้านทานปรับค่าได้แบบ trimmer ขนาด 0-10kΩ เข้าพอร์ต 4AD/1DA โดยหัววัดแต่ละคู่จะต่อเข้า AN0, AN1, AN2

และ AN3 คู่ละ 1 พอร์ต PCF8591 จะรับค่าแรงดันอะนาลอกมาเพื่อแปลงเป็นแรงดันดิจิตอลเพื่อส่งให้ MCU เพื่อทำการประมวลผลและขับออกพอร์ต 8255 ซึ่งวงจรทั้งหมดนี้จะทำงานตลอดเวลา โดยจะทำการเช็คค่าแรงดันและแสดงผล ซึ่งเราสามารถนำมาเขียนเป็นวงจรได้ดังรูป 3.1



รูปที่ 3.1 แสดงวงจรวัดและควบคุมความชื้น

3.3 การออกแบบโปรแกรม

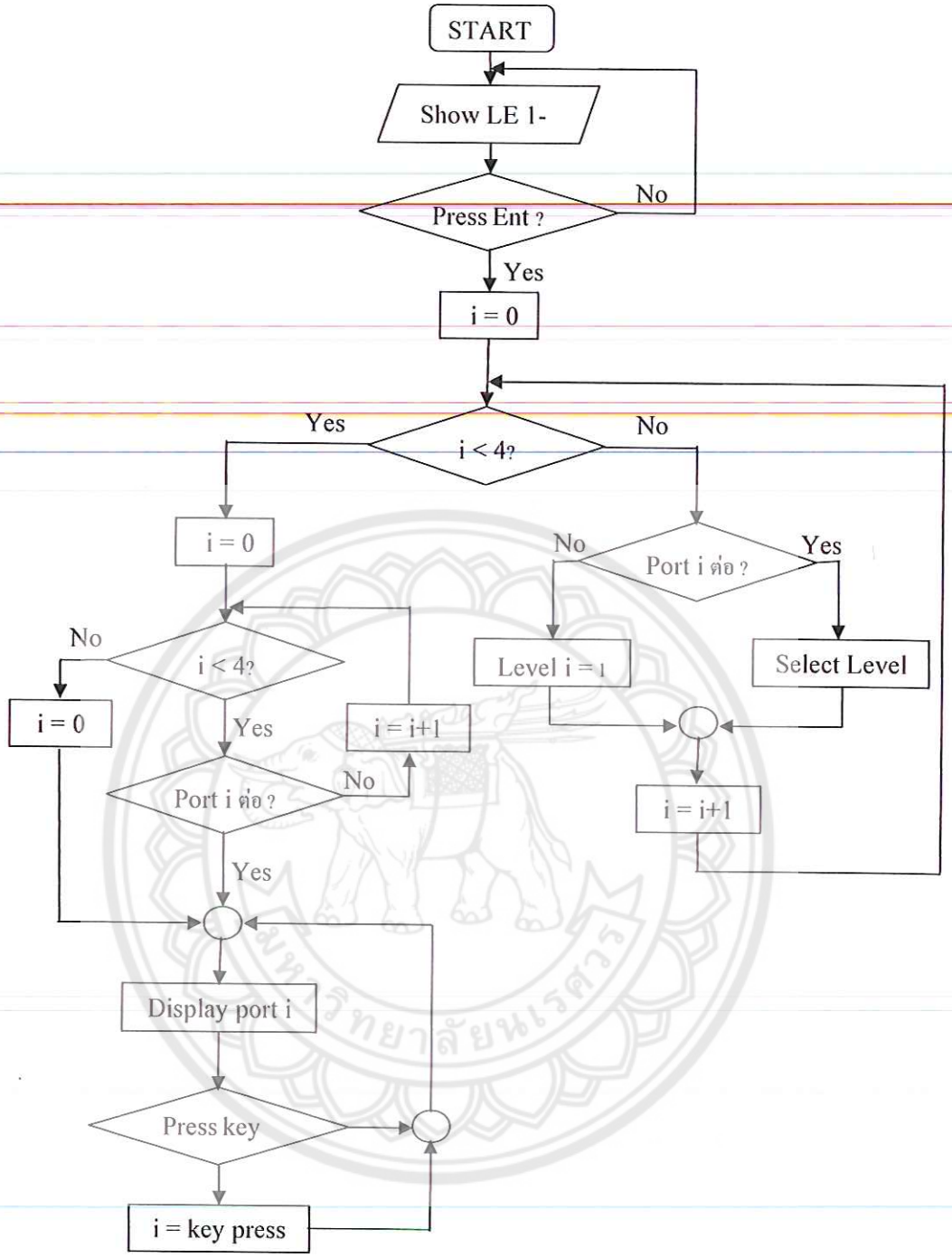
เมื่อรันโปรแกรมขึ้นมาครั้งแรกจะให้ 7-Segment แสดงข้อความ LE 1-3 เพื่อบอกให้เราทราบว่าเราจะสามารถเลือกระดับของความชื้นได้ ตั้งแต่ 1-3 โดยระดับความชื้น 1 , 2 และ 3 ได้ตั้งค่าระดับแรงดันไฟฟ้าเป็น 50H , A0H และ F0H ตามลำดับ ให้กด ENT เพื่อเข้าสู่ขั้นตอนการใส่ค่าระดับความชื้นที่เราต้องการให้แต่ละพอร์ต ซึ่งจะสามารถใส่ค่าได้เฉพาะพอร์ตที่มีการต่อใช้งานอยู่เท่านั้นโดยโปรแกรมจะทำการตรวจสอบพอร์ตของ A/D ว่ามีพอร์ตใดต่อใช้งานอยู่บ้าง เรียงจากพอร์ต 0- 3 โดย 7-Segment จะขึ้นชื่อพอร์ต เพื่อบอกให้เราใส่ค่าระดับความชื้นของพอร์ตนั้น โดย

การกดคีย์ตั้งแต่ 1-3 โดยระดับ 1 เป็นระดับความชื้นที่น้อยที่สุด และระดับ 3 เป็นระดับความชื้นที่มากที่สุด ซึ่งหากเราเลือกตัวเลขที่นอกเหนือจาก 1-3 จะไม่มีผลใดๆ ทั้งสิ้น เมื่อกดคีย์ระดับแล้วจะแสดงระดับที่เรากดที่ digit สุดท้ายของ 7-Segment จนกว่าเราจะกด ENT เมื่อเรากด ENT แล้ว 7-Segment จะแสดงชื่อพอร์ตต่อไปเพื่อให้เราใส่ค่าระดับความชื้นอีก เป็นเช่นนี้จนกว่าจะครบทุกพอร์ตที่ต่อใช้งาน ซึ่งโปรแกรมจะทำการจดจำค่าระดับความชื้นของแต่ละพอร์ต เพื่อนำไปใช้เปรียบเทียบกับค่าความชื้นที่มีอยู่ในดิน พอร์ตที่ไม่ได้ต่อใช้งานโปรแกรมจะไม่ให้ใส่ค่าระดับความชื้นแต่จะข้ามไปพอร์ตถัดไป เมื่อใส่ค่าครบทุกพอร์ตแล้ว จะเป็นการแสดงผลการทำงานของอุปกรณ์ต่างๆ ซึ่งจากข้อมูลเบื้องต้นเราสามารถนำมาเขียนเป็นแผนภูมิได้ดังรูปที่ 3.2

7-Segment มีทั้งหมด 6 digit ในที่นี้จะนับตัวหน้าสุดเป็น digit ที่ 1 การใช้งานนั้น 3 digit แรกจะแสดงชื่อว่าเป็นพอร์ต A/D พอร์ตที่เท่าไร digit ที่ 4 บอกค่าระดับความชื้นว่าเราตั้งค่าความชื้นที่พอร์ตนั้นไว้ระดับไหน ส่วน 2 digit สุดท้ายจะแสดงค่าแรงดันขณะนั้นหรือค่าความชื้นในดินขณะนั้นนั่นเอง โดยจะแสดงเป็นเลขฐานสิบหก ซึ่งการแสดงค่าข้อมูลแต่ละชุดจะถูกคั่นด้วยจุดไข่ปลา (.) พอร์ตที่จะแสดงผลเป็นพอร์ตแรกคือ พอร์ตที่ถูกต่อไว้พอร์ตแรก เช่นถ้าเราไม่ได้ต่อพอร์ต 0 ไว้ ส่วนพอร์ต 1 ถูกต่อใช้งานอยู่ มันจะแสดงผลการทำงานของพอร์ต 1 ในการแสดงผลเนื่องจากเราต่อใช้งานพอร์ต A/D ทั้งสี่พอร์ตจึงไม่สามารถแสดงผลทั้งสี่พอร์ตพร้อมกันได้ ดังนั้นเราจะใช้วิธีการเลือกดูผลพอร์ตใดพอร์ตหนึ่ง โดยการกดคีย์ตามหมายเลขพอร์ตนั้น เช่น เราจะดูพอร์ต 0 ก็ให้กดคีย์ 0 บนคีย์บอร์ด และเราสามารถเลือกดูผลของแต่ละพอร์ตได้ตลอดเวลาโดยไม่ต้องรีเซตโปรแกรมใหม่ หากกรณีที่เรากดคีย์ตรงกับพอร์ตที่ไม่ได้ต่อไว้ให้แสดงผลค่า ระดับความชื้นและแรงดันเป็น "–"และเมื่อเรากดคีย์ในส่วนที่นอกเหนือจาก 0-3 เช่นเรากดคีย์ 5 ซึ่งเราไม่มีพอร์ต A/D พอร์ต 5 7-Segment จะแสดงผลเป็น nO USE ในกรณีที่เราใช้งานพอร์ตไม่ครบทั้ง 4 พอร์ต และต้องการต่อใช้งานเพิ่ม หากเราต้องการจะตั้งระดับความชื้นเราจะต้องทำการรีเซตโปรแกรมโดยการกดปุ่ม reset บนบอร์ด START-C51 ทุกครั้ง ไม่เช่นนั้นแล้วโปรแกรมจะตั้งค่าระดับความชื้นเป็น 1 โดยอัตโนมัติ

LED P3.2, P3.3, P3.4 และ P3.5 (สีเหลือง) จะใช้ในการเช็คสถานะการต่อพอร์ตใช้งานของพอร์ต A/D 0, 1, 2 และ 3 ตามลำดับ โดยหากพอร์ตใดมีการต่อใช้งาน LED ตรงนั้นจะสว่าง ถ้ามีการเลิกใช้งาน หรือเกิดการชำรุดอย่างใดอย่างหนึ่งซึ่งทำให้เกิดการหยุดไหลของกระแสจะทำให้ LED ดับ

PORT 8255 ในที่นี้เราจะนำมาขับ LED ซึ่งใช้แทนก๊อกน้ำมาจำลองเพื่อทดสอบการทำงาน โดยหากพอร์ต A/D พอร์ตใดได้รับค่ามาต่ำกว่าค่าระดับความชื้นที่ตั้งไว้ LED ตรงนั้นจะสว่างโดยในการขับ LED เราจะใช้พอร์ตของ 8255 ครบทั้ง 3 พอร์ต คือ พอร์ต A, พอร์ต B และพอร์ต C ทั้งนี้เราสามารถขับออกพอร์ตอื่นๆ ได้อีก เช่น 12 Bit port เป็นต้น



รูปที่ 3.2 แสดงผังงานการทำงานของโปรแกรมโดยรวม

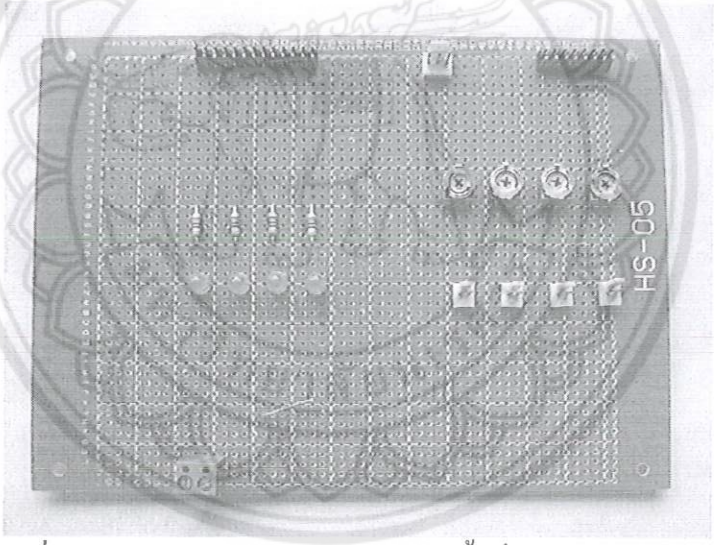
บทที่ 4

การทดลอง

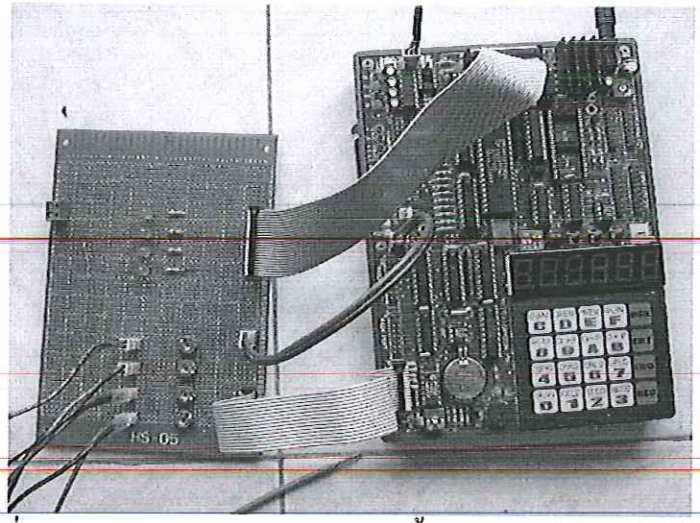
4.1 ขั้นตอนการทดลอง

4.1.1 การจัดเตรียมอุปกรณ์

รูปที่ 4.1 แสดงวงจรวัดและควบคุมความชื้นในดินที่ประกอบเสร็จแล้ว ให้ทำการต่อวงจรวัดและควบคุมความชื้นในดินกับไมโครคอนโทรลเลอร์ ดังรูป 4.2 ใช้แท่งทองแดงแต่ละคู่ที่จะใช้วัดเสียบลงไปในดินลึกประมาณ 5-6 เซนติเมตร หรือลึกกว่านั้นก็ได้แล้วแต่ชนิดของพืชว่ารากอยู่ลึกระดับใด เสียบอแดปเตอร์เพื่อป้อนไฟให้กับบอร์ด จากนั้นเซตจัมพ์เปอร์ของบอร์ดให้อยู่ในโหมดของการ run หรือจะปล่อยลอยก็ได้ ซึ่งจะอยู่ในโหมดของการรันเช่นกัน จากนั้นกดรีเซตหนึ่งครั้งเพื่อเริ่มการรันโปรแกรม



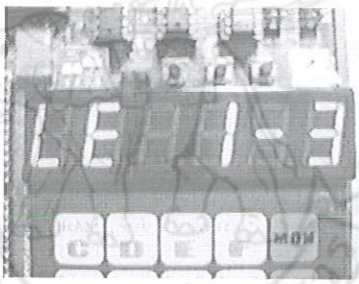
รูปที่ 4.1 แสดงวงจรวัดและควบคุมความชื้นที่ประกอบเสร็จแล้ว



รูปที่ 4.2 การต่อวงจรวัดและควบคุมความชื้นกับไมโครคอนโทรลเลอร์

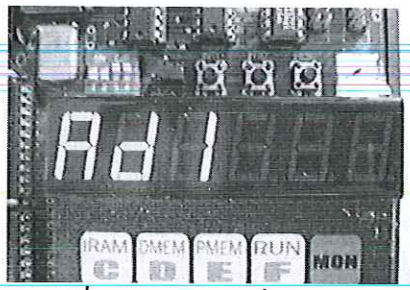
4.1.2 การรันโปรแกรม

เมื่อโปรแกรมรันขึ้นมาครั้งแรกจะแสดงผลดังรูป 4.3



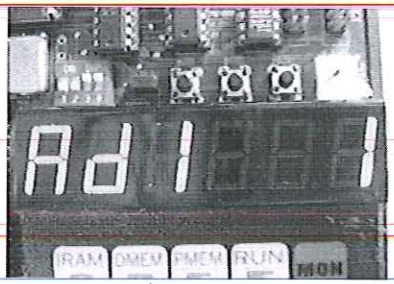
รูปที่ 4.3 การแสดงผลเมื่อรันโปรแกรมครั้งแรก

เมื่อกด ENT ครั้งแรก 7-Segment จะขึ้นชื่อพอร์ตแรกที่ต่อไว้เพื่อให้ใส่ค่าระดับความชื้นดังรูปที่ 4.4 พอร์ตแรกที่เราต่อไว้คือพอร์ต 1 ซึ่งถ้าเรากดคีย์ที่นอกเหนือจากคีย์ 1-3 แล้วจะไม่มีการตอบสนองใดๆ จากโปรแกรมจนกว่าเราจะกดคีย์ 1-3 เท่านั้น



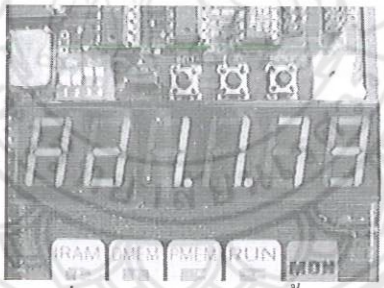
รูปที่ 4.4 แสดงผลเมื่อกด ENT

เมื่อใส่ระดับความชื้นที่ต้องการ แล้วคีย์ที่เรากดจะแสดงดังรูปที่ 4.5 หลังจากนั้นเมื่อกด ENT ก็จะขึ้นชื่อพอร์ตให้เราใส่เหมือนเดิมก็ให้ใส่ไปแล้วกด ENT ทำเช่นนี้จนกว่า จะครบทุกพอร์ตที่มีการต่อใช้งาน

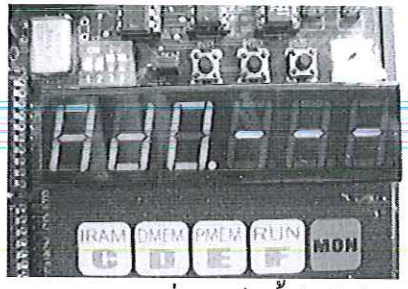


รูปที่ 4.5 แสดงผลเมื่อกดคีย์ใส่ระดับความชื้นเป็น 1

เมื่อกด ENT ครั้งสุดท้ายแล้วจะแสดงผลพอร์ตแรกที่เราต่อใช้งานดังรูปที่ 4.6 ซึ่งพอร์ตแรกที่ต่อใช้งานคือพอร์ต 1 ซึ่ง 7-Segment แต่ละตัวจะแสดงค่าต่างๆ ดังต่อไปนี้ 3 digit แรกจะแสดงชื่อพอร์ตว่าเป็นพอร์ต Ad อะไรแล้วคั่นด้วยจุด (.) ส่วน digit ที่ 4 จะแสดงระดับความชื้นว่าเป็น 1, 2 หรือ 3 แล้วคั่นด้วยจุดอีก ส่วนสอง digit หลังแสดงค่าแรงดันไฟฟ้าซึ่งจะแสดงเป็นเลขฐานสิบหก หากพอร์ตไหนไม่มีการต่อใช้งาน 3 digit หลังจะแสดงเป็น '-' ดังรูปที่ 4.7

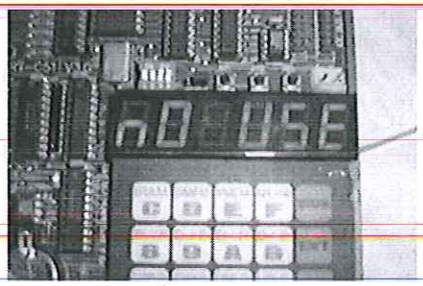


รูปที่ 4.6 แสดงผลเมื่อใส่ค่าระดับความชื้นตัวสุดท้ายแล้วกด ENT



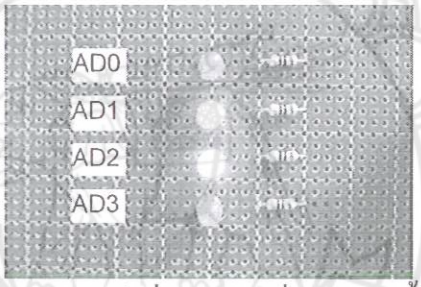
รูปที่ 4.7 แสดงผลเมื่อพอร์ตนั้นไม่ได้ต่อใช้งาน

หากเราต้องการดูผลของพอร์ตอื่นๆ อีก ให้กดคีย์ของพอร์ตนั้น แต่ต้องกดเฉพาะคีย์ 0-3 เท่านั้น เพราะเราสามารถต่อใช้งานได้แค่สี่พอร์ตนี้เท่านั้น หากกดคีย์หมายเลขพอร์ตผิดจะแสดงผล ดังรูปที่ 4.8



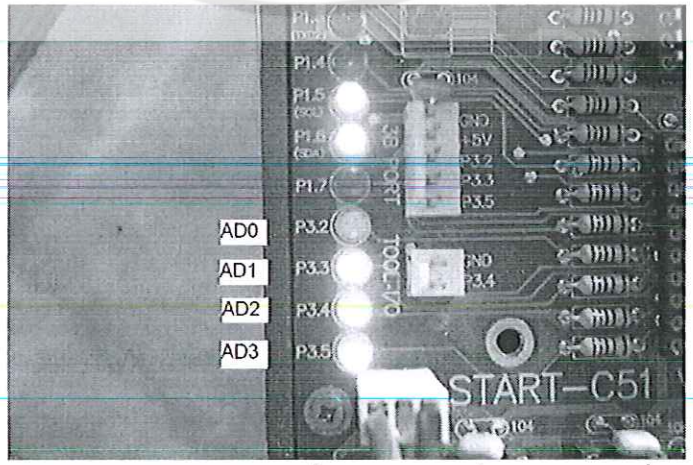
รูปที่ 4.8 แสดงผลเมื่อมีการกดหมายเลขพอร์ตผิด

หากพอร์ตใดมีค่าระดับความชื้นในดินน้อยกว่าระดับความชื้นที่ตั้งไว้ LED ตรงพอร์ตนั้นจะสว่าง ดังรูปที่ 4.9



รูปที่ 4.9 แสดง LED สีเขียวพอร์ตที่ 1 และ 2 เมื่อค่าความชื้นในดินต่ำกว่าที่ตั้งไว้

ส่วน LED สีเหลืองที่อยู่บนบอร์ด หากมีการต่อใช้งานพอร์ตไหน LED ตรงนั้นจะสว่าง ดังรูปที่ 4.10



รูปที่ 4.10 แสดง LED สีเหลืองเมื่อมีการต่อพอร์ต1, 2 และ 3 เพื่อใช้งาน

หากเราถอดหัววัดความชื้น ในขณะที่วงจรกำลังทำงานอยู่แล้ว LED สีเหลือง และสีเขียว ตรงตำแหน่งของพอร์ตนั้นจะดับ เพื่อบอกผู้ใช้ทราบว่าพอร์ตนั้นไม่ได้มีการใช้งานแล้ว ที่เป็นเช่นนี้เพราะเราจะได้ทราบว่าหัววัดความชื้นเกิดการชำรุดหรือไม่

4.1.3 การตั้งค่าระดับความชื้นที่ต้องการ

ก่อนที่จะนำวงจรวัดและควบคุมความชื้นในดินไปใช้งาน เราต้องทำการตั้งค่าความต้านทานปรับค่าได้ให้สัมพันธ์กันระหว่างระดับความชื้นในดินที่พืชต้องการกับระดับความชื้นในดินที่ต้องทำการควบคุมให้ตรงกัน ว่าจะให้ตัวควบคุมความชื้นในดินทำงานหรือหยุดทำงาน เพื่อใช้งานได้ตรงตามที่เราต้องการได้

ขั้นตอนต่อไปนี้สามารถใช้ได้กับทุกระดับความชื้นและดินทุกประเภท ก่อนที่จะนำวงจรวัดและควบคุมความชื้นในดินไปใช้งาน

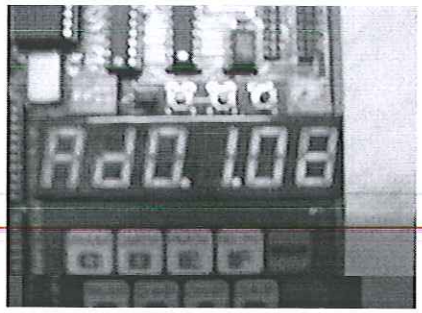
1. นำดิน ณ จุดที่เราต้องการนำวงจรวัดและควบคุมความชื้นในดินไปใช้งานมาทำเป็นดินเป็นตัวอย่าง
2. ให้ทำการรดน้ำดินจนชื้นเท่าที่เราต้องการให้กับพืชนั้น ๆ
3. นำหัววัดความชื้นไปจิ้มลงในดินตัวอย่าง แล้วใส่ RUN โปรแกรม
4. จากนั้นต้องทำการปรับค่าความต้านทานปรับค่าได้ (VR) โดยสังเกตที่ LED ของวงจรควบคุมความชื้นโดยไม่ต้องสนใจว่าดับหรือติดอยู่ แล้วทำการหมุนตัวต้านทานเพื่อหาจุดที่ LED ของวงจรควบคุมความชื้นดับพอดี
5. ณ จุดนี้เราจะได้ค่าความชื้นในดินที่วงจรควบคุมความชื้นหยุดทำงานเมื่อถึงค่าระดับความชื้นที่เกิดขึ้นจริงกับดินในแปลงพื้นที่ใช้งาน
6. จากนั้นก็สามารถนำวงจรวัดและควบคุมความชื้นในดินไปใช้งานจริง

4.1.4 ตัวอย่างผลการทดลอง

กรณีต่อไปนี้เป็นตัวอย่างผลการทดลองของพอร์ต 0 ซึ่งใช้ดินตัวอย่างปริมาตร 150 ลูกบาศก์เซนติเมตร ที่ตัวต้านทานปรับค่าได้มีค่าความต้านทาน 3.525 กิโลโอห์ม ตั้งค่าระดับความชื้นเป็นระดับ 1 และทดลองเติมน้ำในปริมาตรต่างๆ กันดังนี้

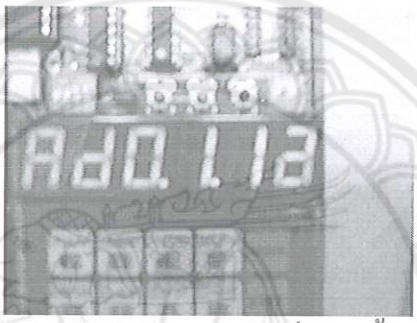
กรณีที่ 1 ดินตากแห้ง ผลที่ได้คือ เกิดการเปลี่ยนแปลงค่าของแรงดันไฟฟ้าตลอดเวลาระหว่างค่า 00H-03H ทำให้ LED ทั้งสีเหลือง (บอกสถานะของการใช้งานพอร์ต) และสีเขียว (บอกว่าพอร์ตไหนมีค่าแรงดันต่ำกว่าค่าแรงดันที่ตั้งไว้) เกิดการกระพริบอยู่ตลอดเวลา

กรณีที่ 2 ดินแห้งที่ไม่ได้ผ่านการตากแดด จะได้ค่าระดับแรงดันเป็น 08H ดังรูป 4.11 ไฟ LED สีเหลือง และสีเขียวที่ตำแหน่งพอร์ต 0 ติดทั้งคู่



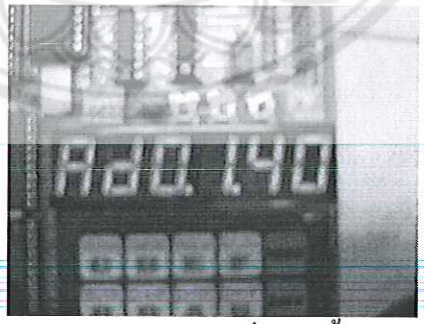
รูปที่ 4.11 ค่าระดับแรงดันไฟฟ้าของดินแห้งที่ไม่ผ่านการตากแดด

กรณีที่ 3 เติมน้ำ 2 ลูกบาศก์เซนติเมตรในดินแห้ง ค่าระดับแรงดันไฟฟ้าเป็น 12H ดังรูป 4.12 ไฟ LED สีเหลืองและสีเขียวที่ตำแหน่งพอร์ต 0 ติดทั้งคู่



รูปที่ 4.12 ค่าระดับแรงดันไฟฟ้าของดินแห้ง เมื่อเติมน้ำ 2 ลูกบาศก์เซนติเมตร

กรณีที่ 4 เติมน้ำ 4 ลูกบาศก์เซนติเมตรในดินแห้ง ค่าระดับแรงดันไฟฟ้าเป็น 40H ดังรูป 4.13 ไฟ LED สีเหลืองและสีเขียวที่ตำแหน่งพอร์ต 0 ติดทั้งคู่

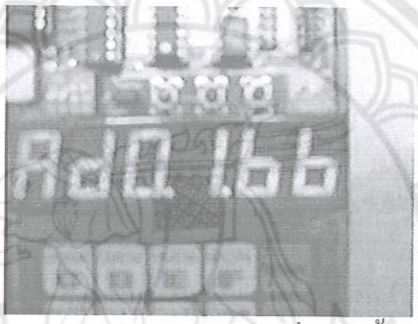


รูปที่ 4.13 ค่าระดับแรงดันไฟฟ้าของดินแห้ง เมื่อเติมน้ำ 4 ลูกบาศก์เซนติเมตร

กรณีที่ 5 เติมน้ำ 6 ลูกบาศก์เซนติเมตรในดินแห้ง ค่าระดับแรงดันไฟฟ้าเป็น A0H ดังรูป 4.14 ไฟ LED สีเหลืองที่ตำแหน่งพอร์ต 0 ติด และ LED สีเขียวที่ตำแหน่งพอร์ต 0 ดับ เนื่องจากแรงดันไฟฟ้ามากกว่าระดับที่กำหนดไว้แล้ว (มากกว่า 50H)

รูปที่ 4.14 ค่าระดับแรงดันไฟฟ้าของดินแห้ง เมื่อเติมน้ำ 6 ลูกบาศก์เซนติเมตร

กรณีที่ 6 เติมน้ำ 8 ลูกบาศก์เซนติเมตรในดินแห้ง ค่าระดับแรงดันไฟฟ้าเป็น BBH ดังรูป 4.15 ไฟ LED สีเหลืองที่ตำแหน่งพอร์ต 0 ติด และ LED สีเขียวที่ตำแหน่งพอร์ต 0 ดับ เนื่องจากแรงดันไฟฟ้ามากกว่าระดับที่กำหนดไว้แล้ว (มากกว่า 50H)

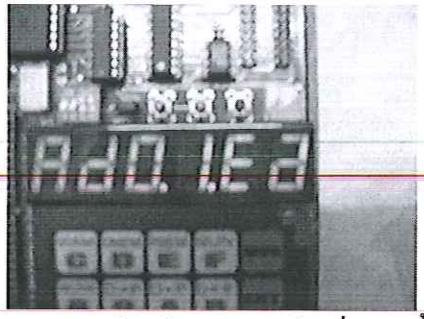


รูปที่ 4.15 ค่าระดับแรงดันไฟฟ้าของดินแห้ง เมื่อเติมน้ำ 8 ลูกบาศก์เซนติเมตร

กรณีที่ 7 เติมน้ำ 10 ลูกบาศก์เซนติเมตรในดินแห้ง ค่าระดับแรงดันไฟฟ้าเป็น D9H ดังรูป 4.16 ไฟ LED สีเหลืองที่ตำแหน่งพอร์ต 0 ติด และ LED สีเขียวที่ตำแหน่งพอร์ต 0 ดับ เนื่องจากแรงดันไฟฟ้ามากกว่าระดับที่กำหนดไว้แล้ว (มากกว่า 50H)

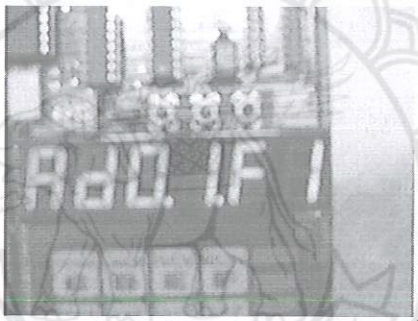
รูปที่ 4.16 ค่าระดับแรงดันไฟฟ้าของดินแห้ง เมื่อเติมน้ำ 10 ลูกบาศก์เซนติเมตร

กรณีที่ 8 เติมน้ำ 12 ลูกบาศก์เซนติเมตรในดินแห้ง ค่าระดับแรงดันไฟฟ้าเป็น E2H ดังรูป 4.17 ไฟ LED สีเหลืองที่ตำแหน่งพอร์ต 0 ติด และ LED สีเขียวที่ตำแหน่งพอร์ต 0 ดับ เนื่องจากแรงดันไฟฟ้ามากกว่าระดับที่กำหนดไว้แล้ว (มากกว่า 50H)



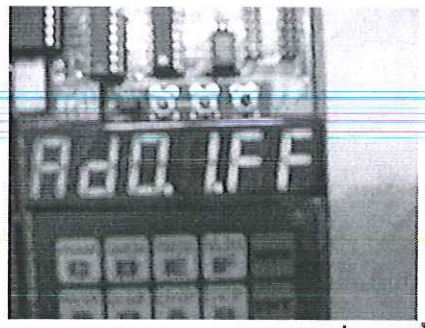
รูปที่ 4.17 ค่าระดับแรงดันไฟฟ้าของดินแห้ง เมื่อเติมน้ำ 12 ลูกบาศก์เซนติเมตร

กรณีที่ 9 เติมน้ำ 14 ลูกบาศก์เซนติเมตรในดินแห้ง ค่าระดับแรงดันไฟฟ้าเป็น F1H ดังรูป 4.18 ไฟ LED สีเหลืองที่ตำแหน่งพอร์ต 0 ติด และ LED สีเขียวที่ตำแหน่งพอร์ต 0 ดับ เนื่องจากแรงดันไฟฟ้ามากกว่าระดับที่กำหนดไว้แล้ว (มากกว่า 50H)



รูปที่ 4.18 ค่าระดับแรงดันไฟฟ้าของดินแห้ง เมื่อเติมน้ำ 14 ลูกบาศก์เซนติเมตร

กรณีที่ 10 เติมน้ำ 16 ลูกบาศก์เซนติเมตรในดินแห้ง ค่าระดับแรงดันไฟฟ้าเป็น FFH ดังรูป 4.19 ไฟ LED สีเหลืองที่ตำแหน่งพอร์ต 0 ติด และ LED สีเขียวที่ตำแหน่งพอร์ต 0 ดับ เนื่องจากแรงดันไฟฟ้ามากกว่าระดับที่กำหนดไว้แล้ว (มากกว่า 50H)



รูปที่ 4.19 ค่าระดับแรงดันไฟฟ้าของดินแห้ง เมื่อเติมน้ำ 16 ลูกบาศก์เซนติเมตร

จากผลการทดลองเราสามารถสรุปค่าแรงดันไฟฟ้ากับปริมาณน้ำที่เติมลงไปได้ดังตาราง 4.1

ตารางที่ 4.1 ความสัมพันธ์ระหว่างดินประเภทต่างๆ และระดับแรงดันไฟฟ้า

ประเภทของดิน	ระดับแรงดันไฟฟ้า (HEX)
ดินแห้ง (ตากแดด)	00H-03H
ดินแห้ง	08H
ดินแห้งเติมน้ำ 2 ลบ.ซม.	12H
ดินแห้งเติมน้ำ 4 ลบ.ซม.	40H
ดินแห้งเติมน้ำ 6 ลบ.ซม.	A0H
ดินแห้งเติมน้ำ 8 ลบ.ซม.	BBH
ดินแห้งเติมน้ำ 10 ลบ.ซม.	D9H
ดินแห้งเติมน้ำ 12 ลบ.ซม.	E2H
ดินแห้งเติมน้ำ 14 ลบ.ซม.	F1H
ดินแห้งเติมน้ำ 16 ลบ.ซม.	FFH

บทที่ 5
สรุป

5.1 สรุปผลการดำเนินโครงงาน

จากการทำโครงงาน การประยุกต์การใช้งานตัวแปลงสัญญาณแบบอะนาลอกไปสู่แบบดิจิตอล (PCF8591) เพื่อใช้วัดและควบคุมความชื้นในดิน สำหรับใช้ในการดูแลเรื่องการให้น้ำของพืชได้

5.2 ข้อเสนอแนะ

ในบทที่ 4 เราจะเห็นว่าเมื่อเราต้องการเสียบใช้งานพอร์ตเพิ่ม เราจะต้องทำการรีเซตใหม่ทุกครั้ง เพราะถ้าไม่มีการรีเซตแล้วโปรแกรมจะตั้งค่าระดับความชื้นที่ต่ำที่สุดให้โดยอัตโนมัติซึ่งถือเป็นข้อจำกัดข้อหนึ่งของโปรแกรม

ส่วน 7-Segment ที่ใช้ในการแสดงผลนั้นจะแสดงผลเป็นแรงดันที่ได้รับมาจากพอร์ต A/D ไม่ได้แสดงเป็นเปอร์เซ็นต์ความชื้นเนื่องจากเราไม่ได้เน้นในเรื่องของความชื้น แต่เน้นเรื่องของการควบคุมมากกว่าจึงใช้การตั้งค่าระดับความชื้นที่กำหนดนั้นเป็นไปเพียงเพื่อการทดลองเท่านั้น ไม่ได้ตั้งตามความต้องการน้ำของพืชที่แท้จริง แต่เมื่อนำไปใช้งานจริงๆ เราจะต้องศึกษาว่าพืชชนิดนั้นต้องการน้ำในดิน หรือความชื้นกี่เปอร์เซ็นต์จึงจะดีที่สุด

ในการแสดงผลออกทาง 7-Segment นั้นมีข้อจำกัดบางประการเช่น ไม่สามารถแสดงตัวอักษรบางตัวแบบสมบูรณ์ได้ การแสดงผลด้วย LCD จะให้ความยืดหยุ่นในการแสดงผลมากกว่าแต่ต้องซื้อมาต่อเพิ่มเองและมีราคาที่ค่อนข้างแพง

ดินแต่ละประเภทจะมีการเรียงตัวของเม็ดดินต่างกัน ซึ่งทำให้การนำไฟฟ้าของดินประเภทนั้นๆ ไม่เท่ากัน การนำวงจรวัดและควบคุมความชื้นไปใช้งาน เราจึงจำเป็นต้องปรับค่า ให้เหมาะสมที่ตัวต้านทานปรับค่าได้ (VR) ในส่วนของวงจรวัดความชื้นได้

เอกสารอ้างอิง

[1] กฤตกร หิรัญเขว้า. ศรีสัชน์ อนุพันธ์. **"การวัดความชื้นของดินเพื่อใช้ในการเกษตรกรรมโดยวิธีวัดความจุไฟฟ้า"** วิทยานิพนธ์วิศวกรรมศาสตรบัณฑิต สาขาวิชาวิศวกรรมไฟฟ้า บัณฑิตวิทยาลัย. มหาวิทยาลัยขอนแก่น. 2546.

[2] กองบรรณาธิการวารสารอิเล็กทรอนิกส์แฮนด์บุ๊ค. **"เครื่องวัดความชื้นในดิน "** ทำเล่นให้เป็นจริง 9. พิมพ์ครั้งที่ 1. ปทุมธานี : สถาบันอิเล็กทรอนิคส์กรุงเทพรังสิต 2545. หน้า 80-87.

[3] รศ.ธีรวัฒน์ ประกอบผล. ภาษาแอสเซมบลี สำหรับ MCS-51. พิมพ์ครั้งที่ 2. กรุงเทพมหานคร: สำนักพิมพ์ ส.ส.ท. 2547.

[4] พนัส นัถฤทธิ์. **"การเชื่อมต่อระหว่างไมโครโปรเซสเซอร์กับเซนเซอร์ชนิดต่างๆ"** เอกสารประกอบการสอน วิชา Microcontroller and Microcomputer Interfacing . 2549

[5] วรพจน์ กรแก้ววัฒนกุล. ชัยวัฒน์ ลิ้มพรจิตรวิไล. เรียนรู้ปฏิบัติการไมโครคอนโทรลเลอร์ MCS-51 แบบแฟลช. พิมพ์ครั้งที่ 4. กรุงเทพมหานคร: บริษัท อินโนเวตีฟ เอ็กเพอริเมนต์ จำกัด. 2539.

[6] R Nave. "Successive Approximation ADC" [Online]. Available: http://hyperphysics.phy-astr.gsu.edu/hbase/electronic/adc.html#c3.

[7] Doug Gingrich. "Successive-Approximation ADC" [Online]. Available: http://www.phys.ualberta.ca/~gingrich/phys395/notes/node166.html. 1999.

[8] SILA RESEARCH CO.,LTD. **"ALL IN ONE."** [CD]. Bangkok: SILA RESEARCH CO.,LTD. 2545.

[9] SILA RESEARCH CO.,LTD. **"MCS-51 SUMMARY."** [Manual]. Bangkok: SILA RESEARCH CO.,LTD. 2545.

[10] SILA RESEARCH CO.,LTD. **"MCS-51."** [Manual]. Bangkok: SILA RESEARCH CO.,LTD. 2545.

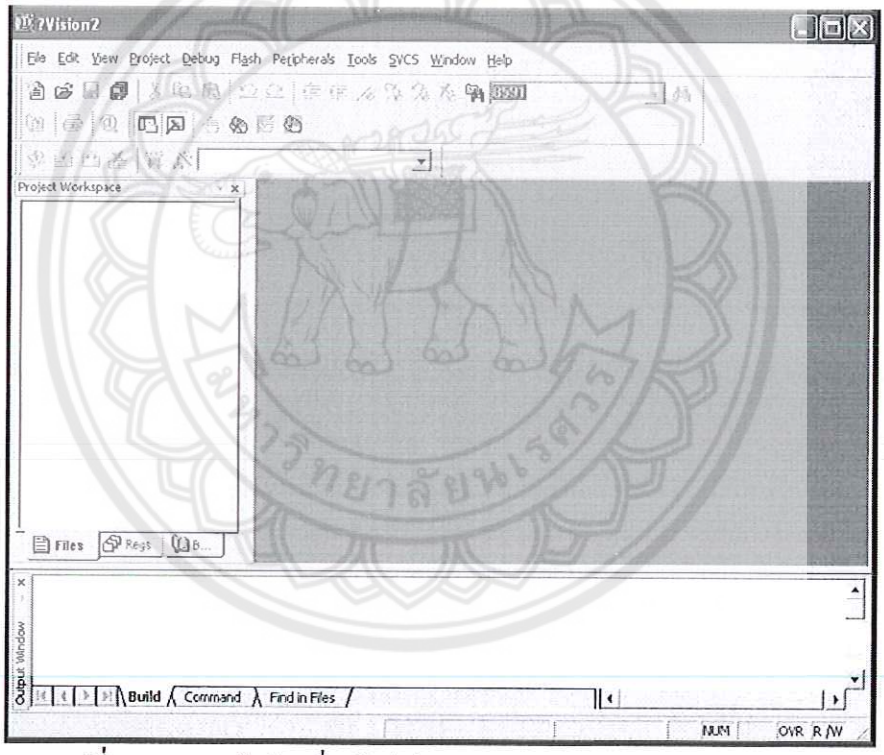
ภาคผนวก ก

การเขียนโปรแกรม

และการดาวน์โหลดโปรแกรมลงบอร์ด

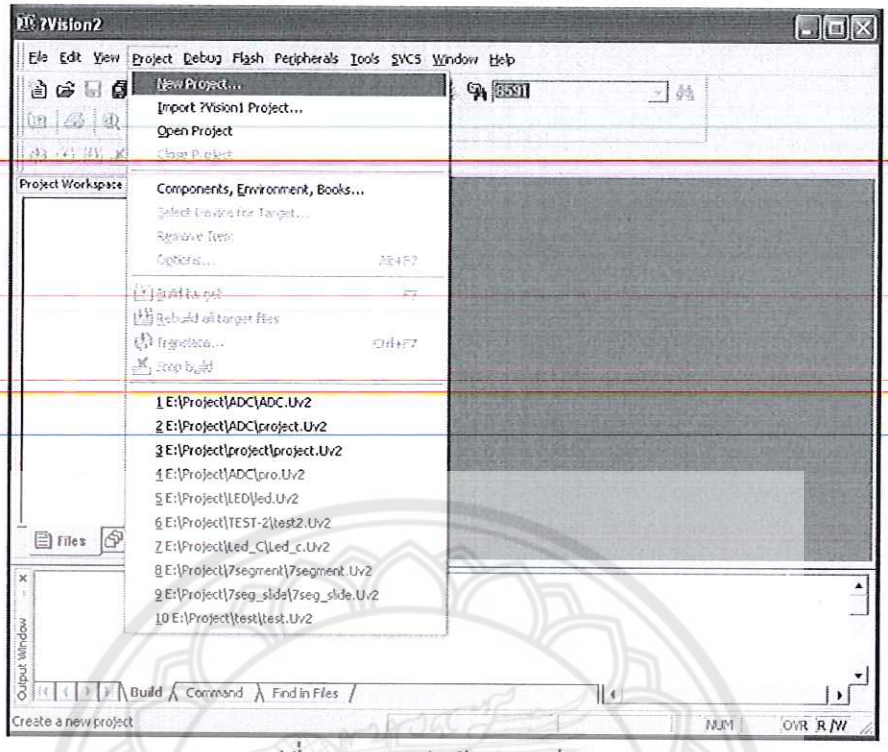
ในการเขียนโปรแกรมเพื่อติดต่อกับไมโครคอนโทรลเลอร์ในการทดลองนี้จะใช้ภาษาซี ซึ่งถือเป็นภาษาพื้นฐานของแทบทุกวงการที่เกี่ยวข้องกับระบบคอมพิวเตอร์ทั้ง Windows, Unix, Linux หรือแม้ระบบขนาดใหญ่อย่าง Mainframe ในที่นี้เราจะใช้ตัวแปร CA51 Compiler ของ Keil Evaluation Version 7.0 ที่ทำงานบน Windows โดยจะมีขั้นตอนและวิธีการดังนี้

เมื่อเปิดโปรแกรม Keil Evaluation Version 7.0 ขึ้นมาจะปรากฏหน้าต่างดังรูปที่ 1



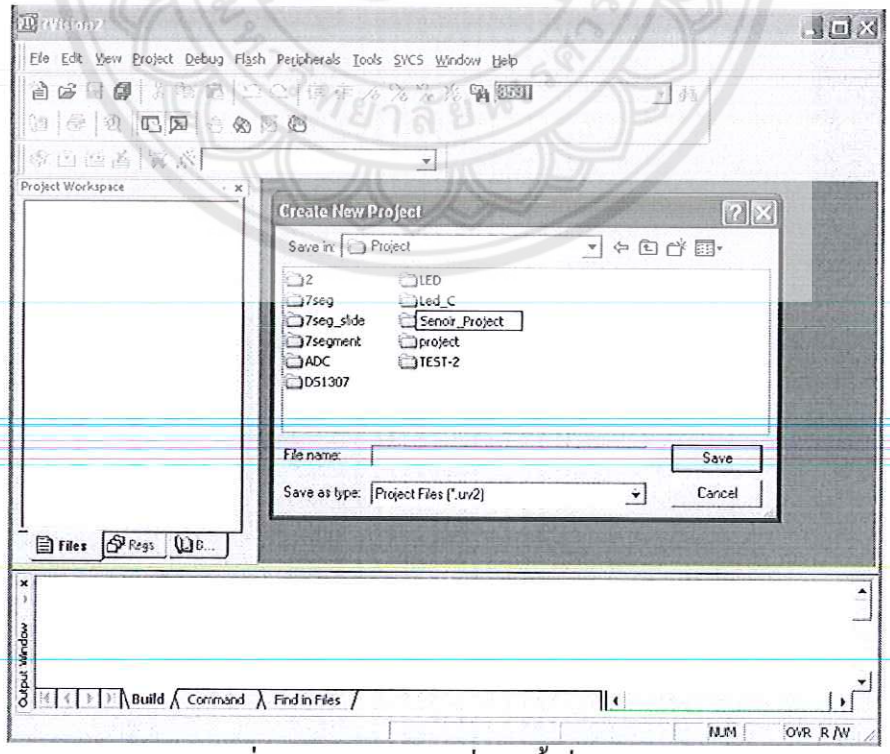
รูปที่ 1 แสดงหน้าต่างเมื่อเปิดโปรแกรม Keil Evaluation Version 7.0

ทำการสร้างโปรเจ็คใหม่โดยคลิกที่ Project> New Project… ดังรูปที่ 2



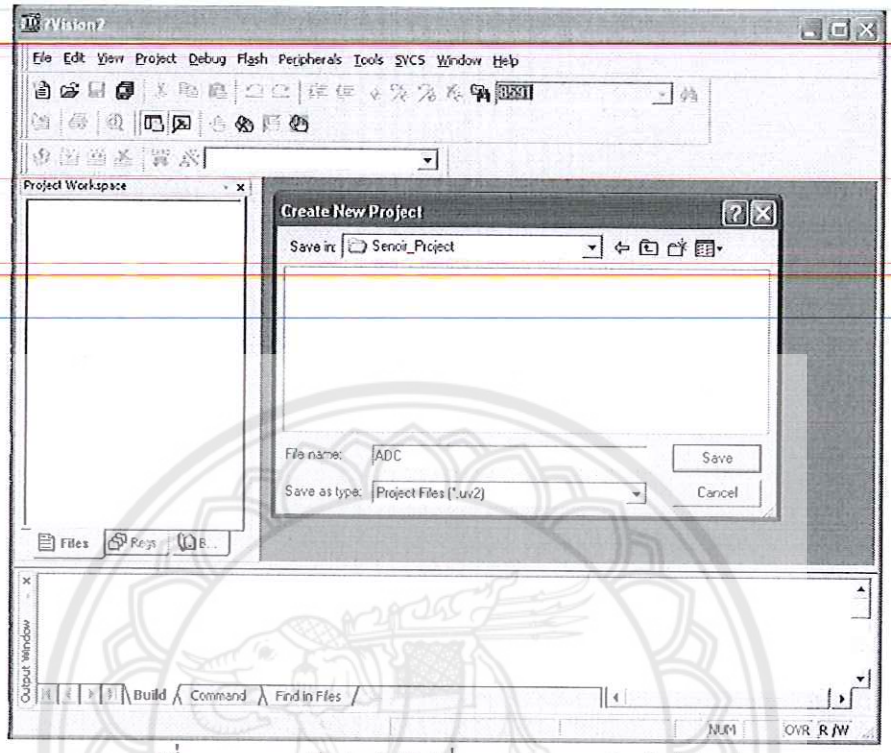
รูปที่ 2 แสดงวิธีสร้างโปรเจ็คใหม่

จะปรากฏหน้าต่างให้เราตั้งชื่อโปรเจ็คดังรูปที่ 3



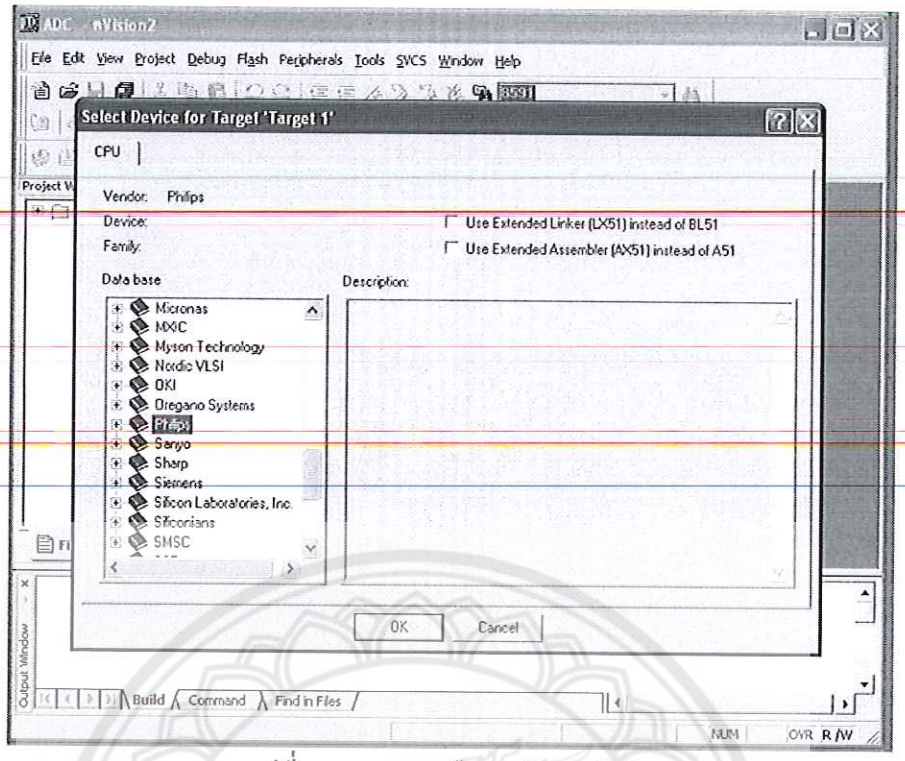
รูปที่ 3 แสดงหน้าต่างเพื่อให้ตั้งชื่อโปรเจ็ค

ให้ใส่ชื่อโปรเจ็คและคลิก Save ดังรูปที่ 4



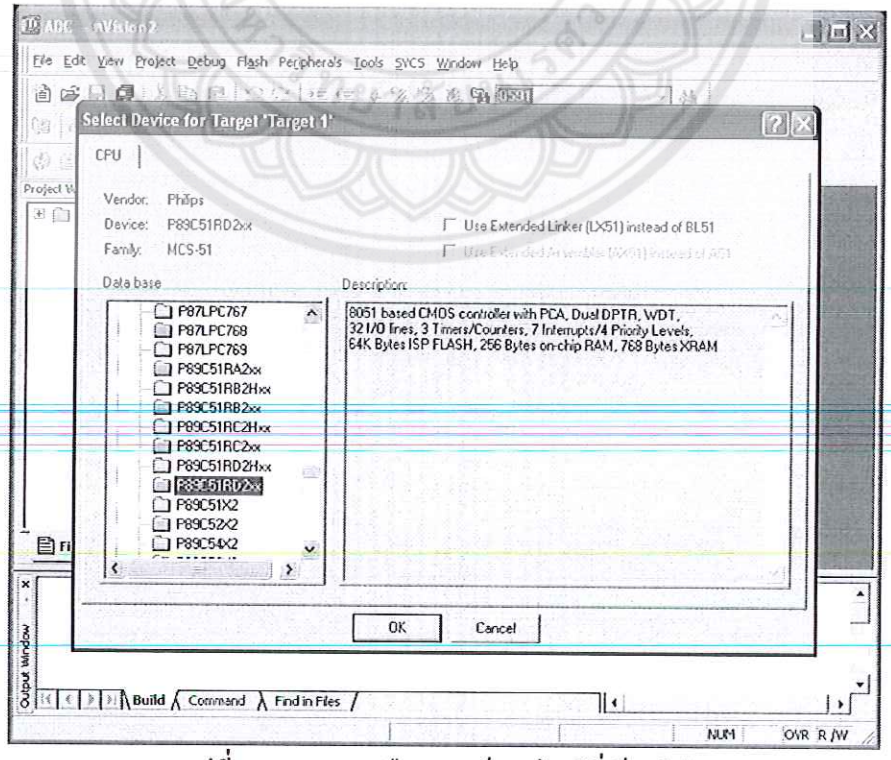
รูปที่ 4 แสดงหน้าต่างให้ใส่ชื่อและทำการบันทึกโปรเจ็ค

หลังจากนั้นจะปรากฏหน้าต่าง Select Device for Target 'Target 1' เพื่อให้เราเลือกชนิดของซีพียู ในช่อง Data base ให้เลือกเป็น Philips ดังรูปที่ 5



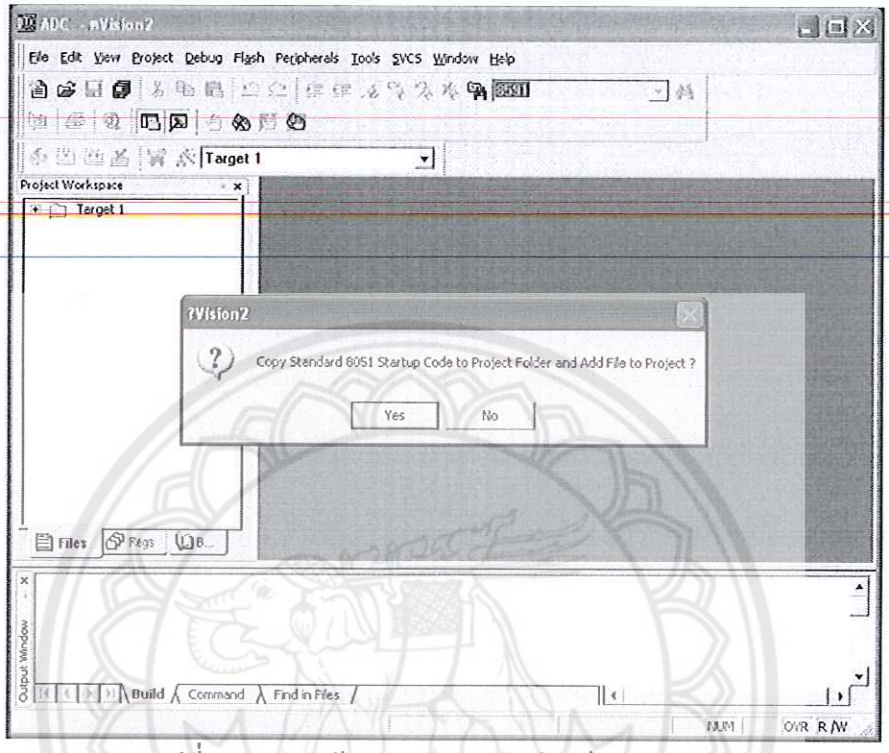
รูปที่ 5 แสดงการเลือกชนิดของซีพียู

และเลือกเบอร์ของไอซีเป็น P89C51RD2xx ดังรูปที่ 6 ซึ่งเป็นไอซีที่อยู่บนบอร์ดที่เราใช้งาน จากนั้นคลิกเลือก ok



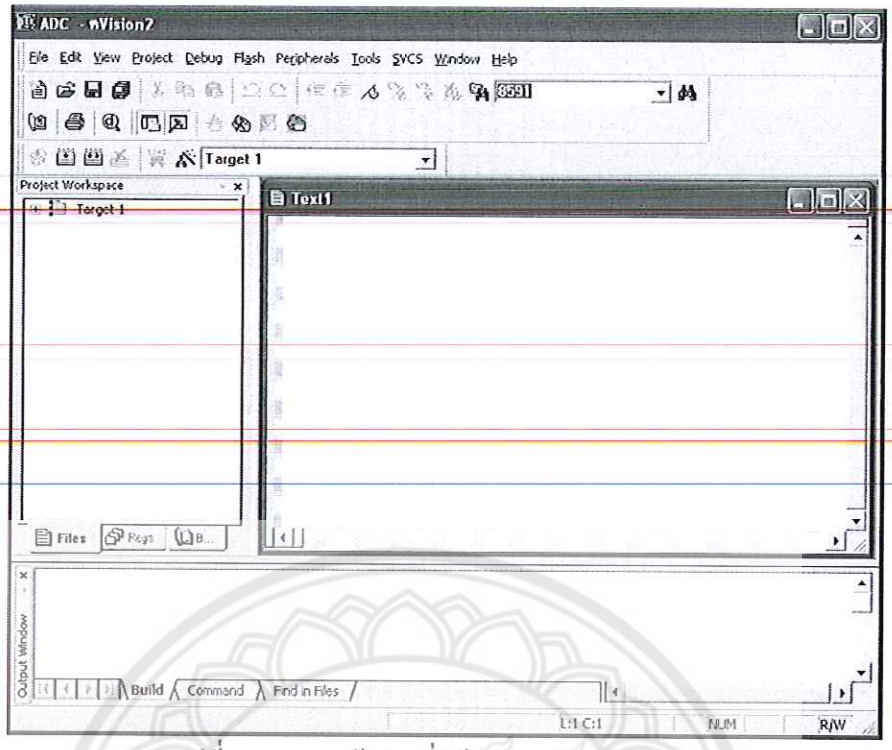
รูปที่ 6 แสดงการเลือกเบอร์ของไอซีที่เป็นซีพียู

จะปรากฏหน้าต่างดังรูปที่ 7 เพื่อถามว่าเราต้องการให้เรียกไฟล์ Startup.A51 มาด้วยหรือไม่ให้คลิก No เพื่อปฏิเสธ



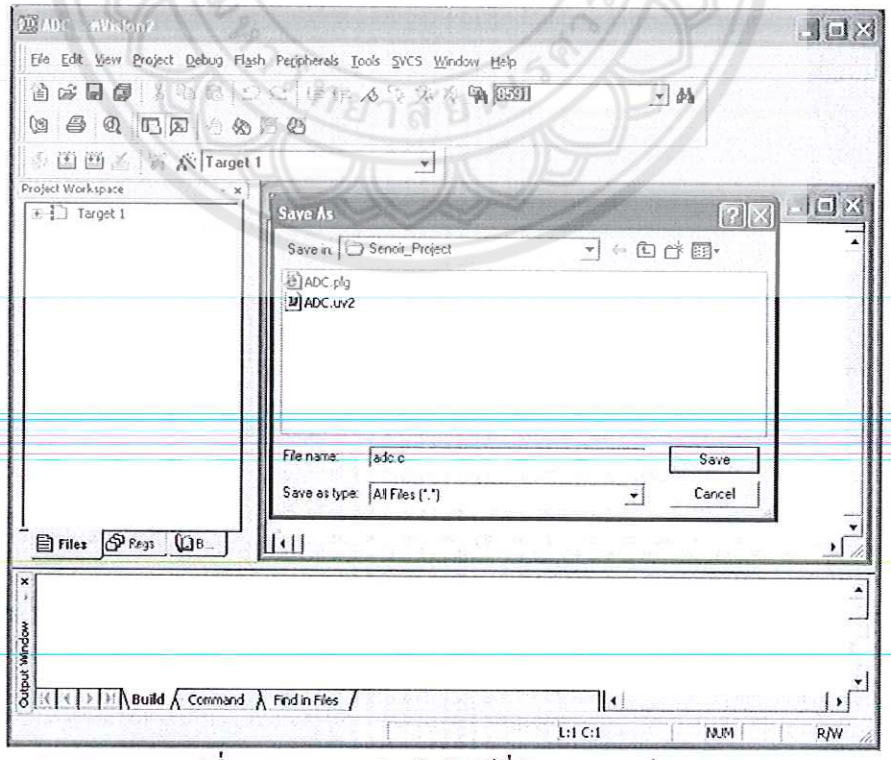
รูปที่ 7 แสดงหน้าต่างถามการเรียกไฟล์ Startup.A51

หลังจากนั้นให้ทำการสร้างไฟล์ที่นามสกุลเป็น .C เพื่อใช้ในการเขียนโปรแกรมโดยคลิกที่ File >New จะปรากฏหน้าต่างให้เราเขียนโปรแกรมดังรูปที่ 8



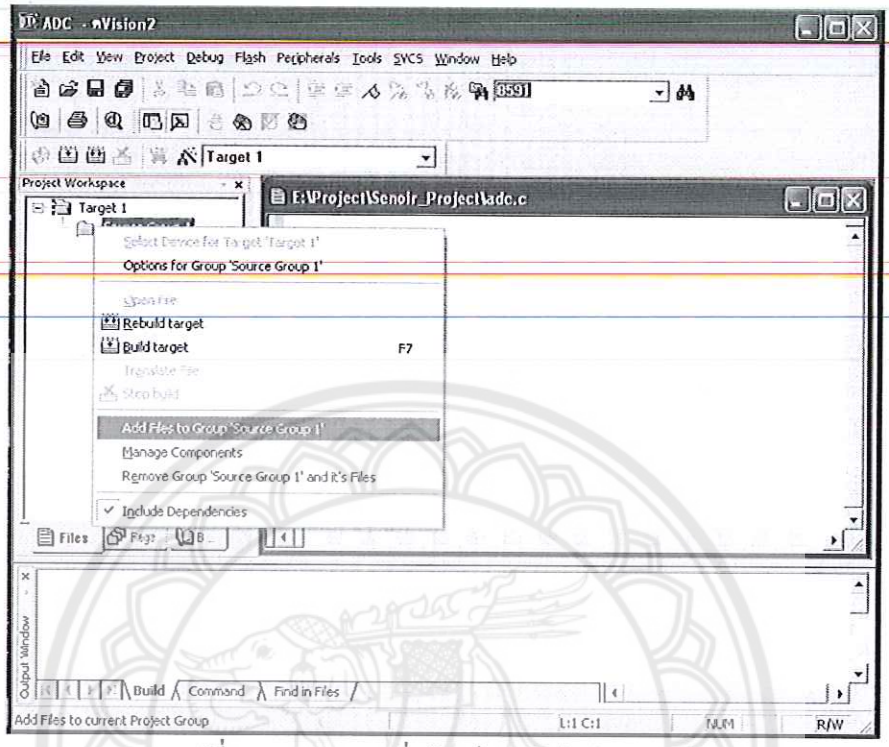
รูปที่ 8 แสดงหน้าต่างที่ใช้ในการเขียนโปรแกรม

หลังจากเขียนโปรแกรมเสร็จแล้วให้เราทำการบันทึกโปรแกรมให้ โดยคลิกที่ File > Save จะปรากฏหน้าต่างให้ใส่ชื่อไฟล์ซึ่งต้องใส่นามสกุลเป็น .C ดังรูปที่ 9 ให้ใส่ชื่อแล้วคลิก Save



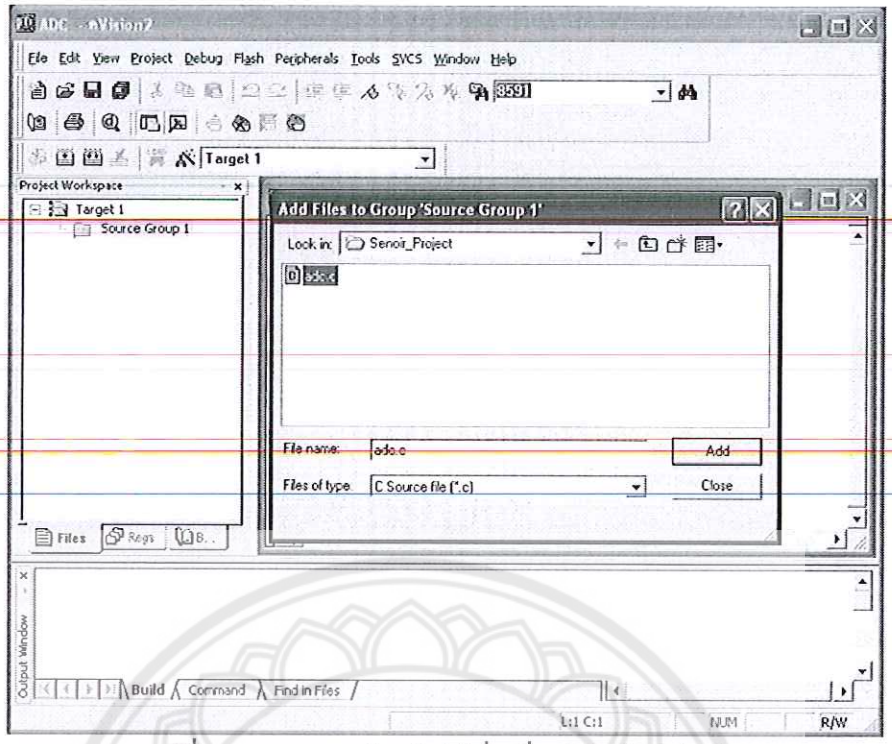
รูปที่ 9 แสดงการบันทึกไฟล์ที่มีนามสกุลเป็น .C

จากนั้นเราจะทำการเพิ่มไฟล์ .C ของเราเข้าไปในโปรเจ็คโดยคลิกขวาที่ Source Group 1 คลิกที่ Add Files to Group 'Source Group 1' ดังรูปที่ 10



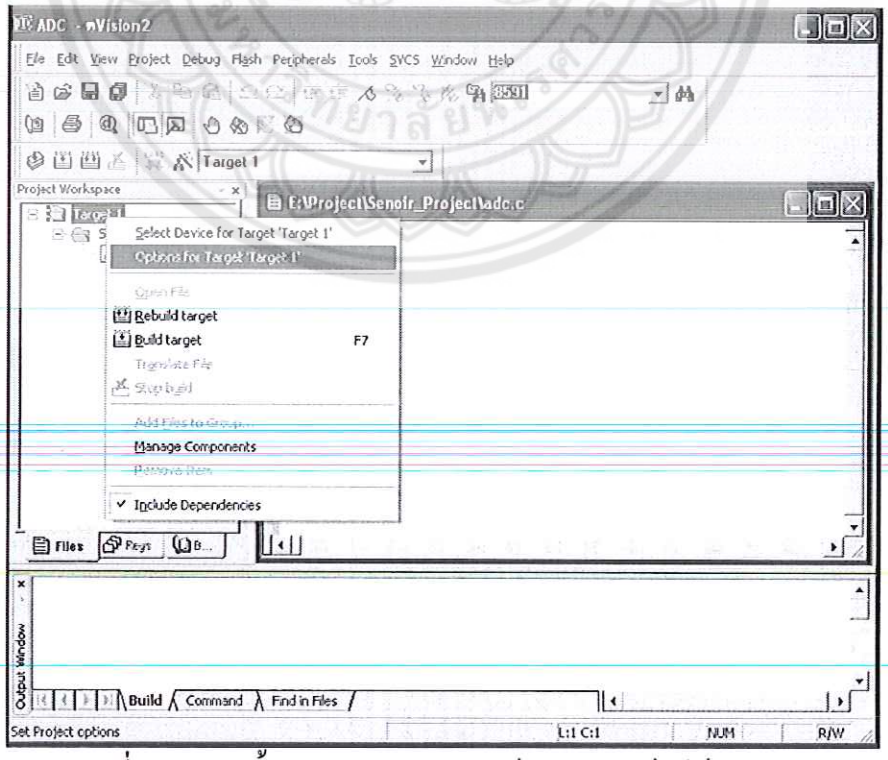
รูปที่ 10 แสดงการเพิ่มไฟล์ .C เข้าไปในโปรเจ็ค

จะปรากฏหน้าต่าง Add Files to Group 'Source Group 1' ดังรูปที่ 11 ให้เราเลือกไฟล์ .C ที่เราได้ทำการบันทึกไว้เมื่อสักครู่ และคลิกที่ Add และ Close ตามลำดับ



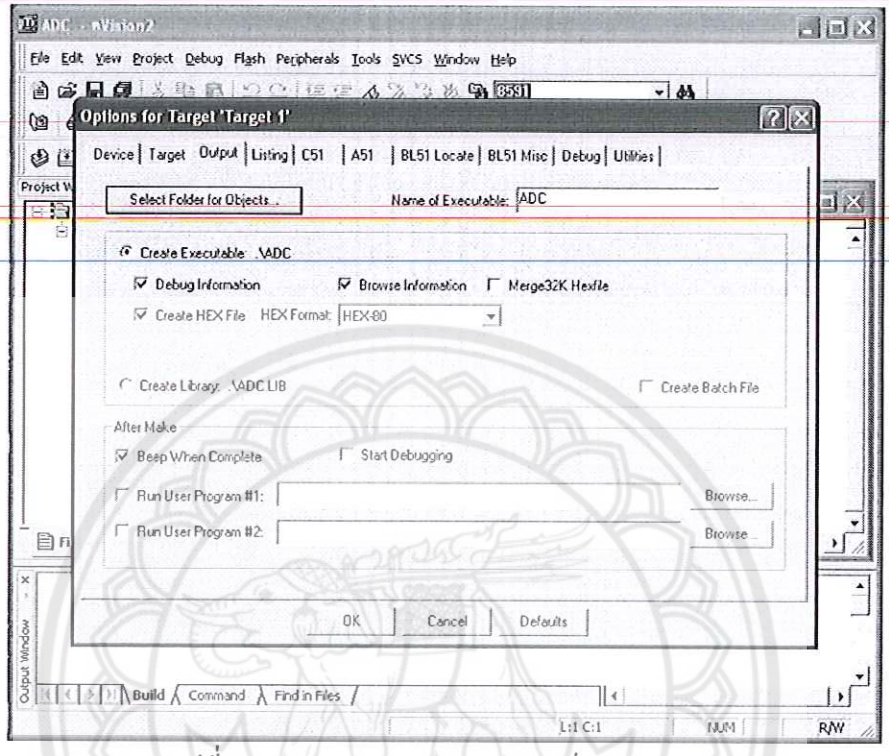
รูปที่ 11 แสดงการเลือกไฟล์เพื่อเพิ่มเข้าไปในโปรเจ็ค

หลังจากนั้นเราจะทำการกำหนดเอาท์พุตของโปรแกรมที่เราได้เขียนไว้ ให้เป็น .HEX โดยคลิกขวาที่โฟลเดอร์ Target 1 และเลือก Options for Target 'Target 1' ดังรูปที่ 12



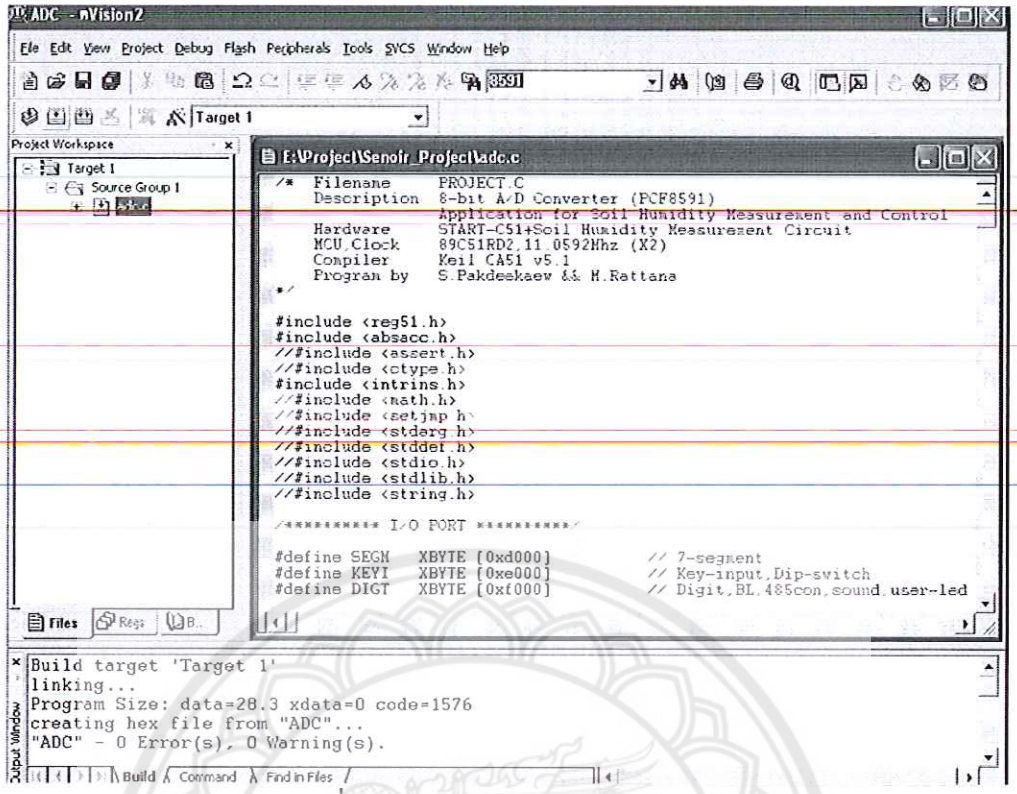
รูปที่ 12 แสดงขั้นตอนการกำหนดเอาท์พุตของไฟล์ให้เป็น .HEX

จะปรากฏหน้าต่าง Options for Target 'Target 1' ดังรูปที่ 13 ให้คลิกเลือก Output และคลิกเครื่องหมายถูกหน้า Create Hex File จากนั้นคลิกที่ OK



รูปที่ 13 แสดงการกำหนด Output เป็น HEX File

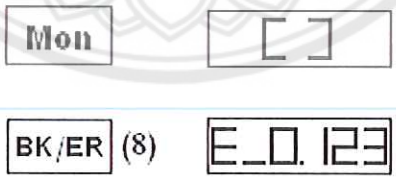
จากนั้นเราจะทำการคอมไพล์ไฟล์ .C โดยคลิกที่  ซึ่งผลลัพธ์ที่ได้จะอยู่ที่หน้าต่าง Output Window ซึ่งอยู่ด้านล่างดังรูปที่ 14



รูปที่ 14 แสดงการคอมไพล์โปรแกรม

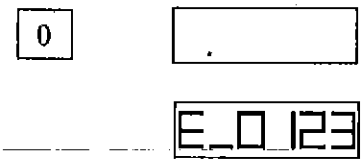
วิธีการตั้งค่าบอร์ด MCS-51 ก่อนทำการดาวน์โหลด

กด MON เพื่อเข้าสู่หน้าจอปกติทำการตรวจสอบหน่วยความจำในส่วนของ Program Memory ซึ่งเราจะโหลดโปรแกรมลง โดยกด BK/ER ว่าเป็น blank หรือไม่ ถ้าหมายเลขใดแสดงจุดไว้ แสดงว่ายังไม่เป็น blank ดังรูปที่ 15



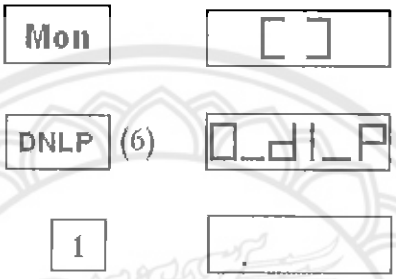
รูปที่ 15 แสดงการตรวจสอบหน่วยความจำภายในบอร์ด

หาก block ใดมีจุดให้กดคีย์หมายเลขนั้นๆ เพื่อทำการล้างหน่วยความจำตรงนั้นให้เป็น blank จากรูปที่ 15 เมื่อเรากดคีย์ 0 บอร์ดจะใช้เวลาสักครู่ในการล้าง จากนั้นจะกลับมาที่เดิม โดยจะเห็นว่าจุดที่หมายเลขหายไปแล้ว ดังรูปที่ 16



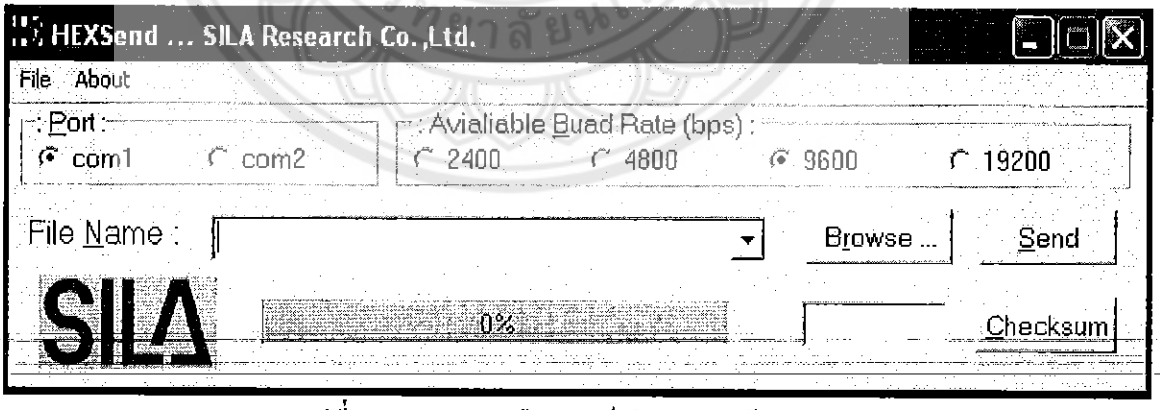
รูปที่ 16 แสดงการล้างหน่วยความจำ

จากนั้นจะทำการตั้งค่าเพื่อรอการดาวน์โหลด โดยการกด MON เพื่อเข้าสู่หน้าจอปกติ กดคีย์ DNLP เพื่อเลือกโหมดการดาวน์โหลด กด 1 เพื่อเลือกดาวน์โหลดโปรแกรม ดังรูปที่ 17



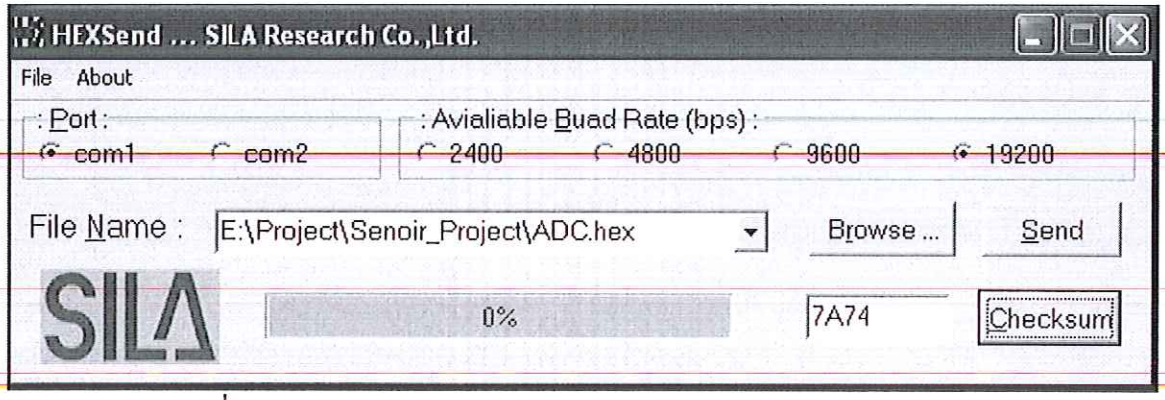
รูปที่ 17 แสดงขั้นตอนการตั้งค่าบอร์ดเพื่อดาวน์โหลดโปรแกรม

จากนั้นเปิดโปรแกรม HEXSend ซึ่งเป็นโปรแกรมที่ใช้ในการส่งข้อมูลระหว่างคอมพิวเตอร์กับบอร์ดไมโครคอนโทรลเลอร์ขึ้นมาและเลือกพอร์ตเป็น com1 ดังรูปที่ 18



รูปที่ 18 แสดงการเลือกพอร์ตในการส่งข้อมูล

หลังจากนั้นตั้งเลือกอัตราการส่งข้อมูลเป็น 19200 บิตต่อวินาที แล้วคลิก Browse เพื่อเลือกไฟล์ .HEX ที่เราต้องการโหลดลงบอร์ด จากนั้นทำการเช็คจำนวนบิตของข้อมูลโดยคลิกที่ Checksum และคลิก Send เพื่อส่งข้อมูลลงบอร์ด ดังรูปที่ 19



รูปที่ 19 แสดงการเลือกอัตราการส่งข้อมูลและการส่งข้อมูลลงบอร์ด

ภาคผนวก ข

Source Code

```
/* Filename    PROJECT.C
   Description  8-bit A/D Converter (PCF8591)  Application for Soil Humidity Measurement and
                Control
   Hardware     START-C51+Soil Humidity Measurement Circuit
   MCU,Clock    89C51RD2,11.0592Mhz (X2)
   Compiler     Keil CA51 v5.1
   Program by   S.Pakdeekaew && M.Rattana
*/
#include <reg51.h>
#include <absacc.h>
#include <assert.h>
#include <ctype.h>
#include <intrins.h>
#include <math.h>
#include <setjmp.h>
#include <stdarg.h>
#include <stddef.h>
#include <stdio.h>
#include <stdlib.h>
#include <string.h>

/********** I/O PORT **********/
#define SEGM   XBYTE [0xd000]          // 7-segment
#define KEYI   XBYTE [0xe000]          // Key-input,Dip-switch
#define DIGT   XBYTE [0xf000]          // Digit,BL,485con,sound,user-led
```

```
/********** 8255 **********/
#define USRA   XBYTE [0x9000]        // 8255 user
#define USRB   XBYTE [0x9001]
#define USRC   XBYTE [0x9002]
#define USRP   XBYTE [0x9003]

/************ I2C **********/
sbit   IPSCL  = P1^5;                // I2C I/O Bit
sbit   IPSDA  = P1^6;

sbit   AD0 = P3^2;
sbit   AD1 = P3^3;
sbit   AD2 = P3^4;
sbit   AD3 = P3^5;

/********** INT-RAM WORKING AREA **********/
bit         KEYFAG;                  // key press flag
unsigned char L[4];                  // Level Buffer
unsigned char DISBUF[6];             // display buffer (reserve for 6 digit)
unsigned char HEXBUF[3];             // hex buffer
unsigned char DIGMEM;                // DIGT port buffer

/********** BASIC FUNCTION **********/
/************ D E L A Y ************/
void dmsec (unsigned int count)      // mSec Delay
{
```

```
    unsigned int i;              // Keil CA51 (x2)
    while (count)
          {
      i = 230; while (i>0) i--;
      count--;
    }
}

unsigned char code SEGTAB[16] = {0x3F,0x06,0x5B,0x4F,0x66,0x6D,0x7D,0x07,
                                 0x7F,0x6F,0x77,0x7C,0x39,0x5E,0x79,0x71};
                                    // standard segment code

unsigned char htosx (unsigned char hex)   // change hex to segment
{
  return (SEGTAB[hex]);
}

void htos (void)
{
  DISBUF[0] = htosx ((HEXBUF[0] & 0xf0) >> 4);    // change hex to segment
  DISBUF[1] = htosx (HEXBUF[0] & 0x0f);                    // hex[3] to disbuf[6]
  DISBUF[2] = htosx ((HEXBUF[1] & 0xf0) >> 4);
  DISBUF[3] = htosx (HEXBUF[1] & 0x0f);
  DISBUF[4] = htosx ((HEXBUF[2] & 0xf0) >> 4);
  DISBUF[5] = htosx (HEXBUF[2] & 0x0f);
}

unsigned char code KEYTAB[] = {0x70,0x71,0x72,0x73,  // for all 20 key table
```

```
                0xb0,0xb1,0xb2,0xb3,
                0xd0,0xd1,0xd2,0xd3,
                0xe0,0xe1,0xe2,0xe3,
                0x74,0xb4,0xd4,0xe4};

unsigned char scan (void)              // scan display & get key
{
  unsigned char i,j,x;
  bit pressok;
  pressok = 0;
  for (i=0;i<=5;i++)
      {
    DIGMEM = (DIGMEM & 0xf0) | i;  // out digit
    DIGT = DIGMEM;
    SEGM = DISBUF[i];              // out segment
    for (j=0;j<=200;j++);          // delay
    SEGM = 0;
    x = KEYI&0x0f;                 // check key
    if (x!=0x0f)
          {
      pressok = 1;
      if (!KEYFAG)                 // new press
            {
        x = (x << 4) | i;
        for (j=0;j<=19;j++)
              {
          if (x==KEYTAB[j])
                {
```

```
                KEYFAG = 1;
                return (j);
              }
            }
          }
        }
      }
      if (!pressok) KEYFAG = 0;          // No any key press , clear KEYFAG (release)
      return (0xff);
}

void ipdel (void)                        // I2C delay
{
  _nop_ ();
  _nop_ ();
  _nop_ ();
  _nop_ ();
  _nop_ ();
  _nop_ ();
  _nop_ ();
  _nop_ ();
}

void ipchigh (void)        // I2C clock high
{
  IPSCL = 1;
  ipdel ();
}
```

```
void ipclow (void)                    // I2C clock low
{
   IPSCL = 0;
   ipdel ();
}

void ipstart (void)          // start condition
{
   IPSDA = 1;
   IPSCL = 1;
   IPSDA = 0;
   ipdel ();
   IPSCL = 0;
   IPSDA = 1;
}

void ipstop (void)                    // stop condition
{
   IPSDA = 0;
   IPSCL = 1;
   ipdel ();
   IPSDA = 1;
}

bit ipwrbyte (unsigned dat)          // write one byte
{
```

```
    unsigned char i;          // return 0 = ok
    bit outbit;                          // return 1 = error
    for (i=1;i<=8;i++)
          {
      outbit = dat & 0x80;
      IPSDA = outbit;
      dat = dat << 1;
      ipchigh ();
      ipclow ();
    }
    IPSDA = 1;
    ipchigh ();
    outbit = IPSDA;
    ipclow ();
    return (outbit);
}

unsigned char iprdbyte ()          // read last byte
{
    unsigned char i,dat;
    bit inbit;
    dat = 0;
    for (i=1;i<=8;i++)
          {
      ipchigh ();
      inbit = IPSDA;
      dat = dat << 1;
      dat = dat | inbit;
```

```
        ipclow ();
    }
    IPSDA = 1;
    ipchigh ();
    ipclow ();
    return (dat);
}

unsigned char ipad (unsigned char addr,ch)          // read A/D
{
    unsigned char dat;
    bit err;
    ipstart ();                    // **** set control ****
    addr = ((addr << 1) & 0x0e) | 0x90;
    err = ipwrbyte (addr);
    if (!err)
        {
        ipwrbyte (0x40 | ch);          // control byte
        ipstop ();
        dmsec (1);
        ipstart ();                // **** read a/d ****
        addr = addr | 0x1;             // change r/w
        err = ipwrbyte (addr);
        if (!err) dat = iprdbyte ();
    }
    ipstop ();
    return (dat);
}
```

```
unsigned char slpad(unsigned char a) //select level humidity
{
  unsigned char x;
        DISBUF[0] = 0x77;                    // display Ad[channel]
  DISBUF[1] = 0x5e;
  DISBUF[2] = SEGTAB[a];
  DISBUF[3] = 0;
  DISBUF[4] = 0;
  DISBUF[5] = 0;
  while(((x=scan())>0x3)||(x==0));//wait press key 1-3
  DISBUF[5] = htosx(x & 0x0f);//display press key
  while(scan()!=0x12);                  //wait press key ENT
  return(x);                                    //return number level
}


unsigned char compare(unsigned char x)  //compare level humidity and voltage
{
  unsigned char voltcomp;
  switch(x)
        {
          case 0x1: voltcomp = 0x50; break;
          case 0x2: voltcomp = 0xA0; break;
          case 0x3: voltcomp = 0xfe; break;
  }
        return(voltcomp); //return voltage compare
}
```

```
unsigned char error(void){          //display when key press not Ad 0-3
unsigned char x;
  do
        {
    DISBUF[0] = 0x54; //display nO USE
        DISBUF[1] = 0x3f;
        DISBUF[2] = 0;
        DISBUF[3] = 0x3e;
        DISBUF[4] = 0x6d;
        DISBUF[5] = 0x79;
  }
        while((x=scan())==0xff);
  return (x);
}

void leduse(void)                   //display led && 8255
{
   ipad(0,0);                       //port 0
   AD0 = (ipad(0,0)>0)? 0:1;
   ipad(0,1);                       //port 1
   AD1 = (ipad(0,1)>0)? 0:1;
   ipad(0,2);                       //port 2
   AD2 = (ipad(0,2)>0)? 0:1;
   ipad(0,3);                       //port 3
   AD3 = (ipad(0,3)>0)? 0:1;
}
```

```
void porta(void){
        unsigned char x0;
        ipad(0,0);
        x0=ipad(0,0);
        switch (USRA){
                case 0x00:
                        if((x0<compare(L[0]))&&(x0>0))  USRA=0x01;
                case 0x01:
                        if((x0>=compare(L[0]))&&(x0>0)) USRA=0x00;
                        else if (x0<=0) USRA=0x00;
        }
}

void portb(void){
        unsigned char x1;
        ipad(0,1);
        x1=ipad(0,1);
        switch (USRB){
                case 0x00:
                        if((x1<compare(L[1]))&&(x1>0)) USRB=0x01;
                case 0x01:
                        if((x1>=compare(L[1]))&&(x1>0))USRB=0x00;
                        else if (x1<=0) USRB=0x00;
        }
}

void portc(void){
```

```
unsigned char x2,x3;
ipad(0,2);
x2=ipad(0,2);

ipad(0,3);
x3=ipad(0,3);

if (x2>0&&x3>0){
        switch (USRC){
                case 0x00:
                        if((x3<compare(L[3]))&&(x2<compare(L[2])))
USRC=0x81;
                        else if((x3<compare(L[3]))&&(x2>=compare(L[2])))
USRC=0x80;
                        else if((x3>=compare(L[3]))&&(x2<compare(L[2])))
USRC=0x01;
                case 0x01:
                        if((x3<compare(L[3]))&&(x2<compare(L[2])))
USRC=0x81;
                        else if((x3<compare(L[3]))&&(x2>=compare(L[2])))
USRC=0x80;
                        else if((x3>=compare(L[3]))&&(x2>=compare(L[2])))
USRC=0x00;
                case 0x80:
                        if((x3>=compare(L[3]))&&(x2<compare(L[2])))
USRC=0x01;
                        else if((x3>=compare(L[3]))&&(x2>=compare(L[2])))
USRC=0x00;
                        else if((x3<compare(L[3]))&&(x2<compare(L[2])))
USRC=0x81;
                case 0x81:
                        if((x3>=compare(L[3]))&&(x2<compare(L[2])))
USRC=0x01;
```

```
                    else if((x3>=compare(L[3]))&&(x2>=compare(L[2])))
USRC=0x00;
                    else if((x3<compare(L[3]))&&(x2>=compare(L[2])))
USRC=0x80;
        }
}

else if ((x2>0)&&(x3<=0)){
        switch (USRC){
                case 0x00:
                        if(x2<compare(L[2]))    USRC= 0x01;
                case 0x01:
                        if(x2>=compare(L[2]))   USRC= 0x00;
                case 0x80:
                        if(x2<compare(L[2]))    USRC= 0x01;
                case 0x81:
                        if(x2<compare(L[2]))    USRC= 0x00;
        }
}

else if (x2<=0&&x3>0){
        switch (USRC){
                case 0x00:
                        if(x3<compare(L[3]))    USRC= 0x80;
                case 0x01:
                        if(x3>=compare(L[3]))   USRC= 0x80;
                case 0x80:
                        if(x3<compare(L[3]))    USRC= 0x00;
                case 0x81:
```

```
                    if(x3<compare(L[3]))     USRC= 0x00;
                }
            }

            else      USRC=0x00;

    }

    void drive8255(void)
    {
            porta();
            portb();
            portc();
    }

    void start(void)
    {
       do
            {
        DISBUF[0] = 0x38; // Display LE 1-3 wait press key    (Level)
        DISBUF[1] = 0x79;
        DISBUF[2] = 0;
        DISBUF[3] = 0x06;
        DISBUF[4] = 0x40;
        DISBUF[5] = 0x4f;
       }
            while (scan ()!=0x12);
    }
```

```
unsigned char display(unsigned char l,p)   //display when runing program
{
   unsigned char x;
   do
        {
              ipad(0,p);
              if (ipad(0,p)>0)
              {                                        //check using A/D port
                    HEXBUF[2] = ipad (0,p);          // display voltage using port
              htos ();
              }
              else
              {
              DISBUF[4] = 0x40;                  // display voltage not using port (--)
              DISBUF[5] = 0x40;
              }
              DISBUF[0] = 0x77;                  //display A/D
              DISBUF[1] = 0x5e;
              DISBUF[2] = SEGTAB[p]|0x80; //display number port
              DISBUF[3] = (ipad(0,p)>0)? (htosx(l & 0x0f)|0x80):0x40; //display level humidity
              leduse();                                    //led on when using port
              drive8255();
   }
   while ((x=scan ())==0xff);
   return (x);    // return next port display
}
```

```
void run(void)
{
  unsigned char x,i;
 USRP=0x80;
  for(i=0;i<4;i++)              // check using port and input level humidity port
       {
    ipad(0,i);
    if(ipad(0,i)>0)
                {
                      L[i]=slpad(i);
                }
    else
                      L[i]=1;
  }
  for(i=0;i<4;i++)              //check first port used for display on 7-segment
       {
    ipad(0,i);
    if(ipad(0,i)>0)
                {
                    x = display(L[i],i);break;
                }
  }

  while(1)              //check press key and display on 7-segment,led,8255 port
       {

            switch(x)
            {
```

```
                case 0x0: x = display(L[0],0); break;        //press key 0
                case 0x1: x = display(L[1],1); break;        //press key 1
                case 0x2: x = display(L[2],2); break;        //press key 2
                case 0x3: x = display(L[3],3); break;        //press key 3
                default:x = error();         //press other key
            }
        }
    }

void main()
{
    start();
    run();
}
```

ประวัติผู้เขียนโครงงาน

ชื่อ นายมานพ รัตนะ

ภูมิลำเนา 34 หมู่ 6 ซ. แสงศิริ ต. ทุ่งสมอ อ. เขาค้อ จ. เพชรบูรณ์ 67270

ประวัติการศึกษา

- จบการศึกษาจากโรงเรียนแคมป์สนวิทยาคม
- ปัจจุบันกำลังศึกษาในระดับปริญญาตรีชั้นปีที่ 4
  สาขาวิศวกรรมคอมพิวเตอร์ คณะวิศวกรรมศาสตร์
  มหาวิทยาลัยนเรศวร

E – mail : deaw_boy_cpex@hotmail.com

ชื่อ นางสาวศุลีพร ภักดีแก้ว

ภูมิลำเนา 295 หมู่ 8 ต.แม่เปิน กิ่งอ.แม่เปิน จ.นครสวรรค์ 60150

ประวัติการศึกษา

- จบการศึกษาจากโรงเรียนประชามงคล
- ปัจจุบันกำลังศึกษาในระดับปริญญาตรีชั้นปีที่ 4
  สาขาวิศวกรรมคอมพิวเตอร์ คณะวิศวกรรมศาสตร์
  มหาวิทยาลัยนเรศวร

E – mail : sucenu@hotmail.com