ตู้ควบคุมอุณหภูมิสำหรับเด็กแรกเกิดด้วยคอมพิวเตอร์

PC - CONTROLLED INFANT INCUBATOR

โดย

นายเฉลิมพล คำอิน

นายดิเรก ปัทมะสุคนธ์

ว่าที่ร้อยตรีเดชฤทธ์ จันทร์แก้ว

ปริญญานิพนธ์ฉบับนี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิศวกรรมศาสตรบัณฑิต

สาขาวิชาอิเล็กทรอนิกส์

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

ปีการศึกษา 2551

ตู้ควบคุมอุณหภูมิสำหรับเด็กแรกเกิดด้วยคอมพิวเตอร์

PC - CONTROLLED INFANT INCUBATOR

โดย

นายเฉลิมพล คำอิน รหัสประจำตัวนักศึกษา 49015188

นายดิเรก ปัทมะสุคนธ์ รหัสประจำตัวนักศึกษา 49015191

ว่าที่ร้อยตรีเดชฤทธ์ จันทร์แก้ว รหัสประจำตัวนักศึกษา 49015193

ปริญญานิพนธ์สำหรับวิศวกรรมศาสตรบัณฑิต

สาขาวิชาอิเล็กทรอนิกส์

คณะวิศวกรรมศาสตร์

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

ปีการศึกษา 2551

ปริญญาปีการศึกษา 2551

ภาควิชา อิเล็กทรอนิกส์

คณะวิศวกรรมศาสตร์ สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

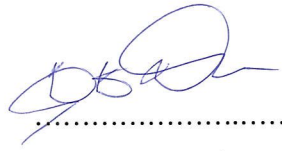
เรื่อง ตู้ควบคุมอุณหภูมิสำหรับเด็กแรกเกิดด้วยคอมพิวเตอร์

ผู้จัดทำ

นายเฉลิมพล คำอิน

นายดิเรก ปัทมะสุคนธ์

ว่าที่ร้อยตรีเดชฤทธิ์ จันทร์แก้ว



..... อาจารย์ที่ปรึกษา

(รศ.ดร. ชูชาติ ปิณฑวิรุจน์)

23/3/52

ตู้ควบคุมอุณหภูมิสำหรับเด็กทารกแรกเกิดด้วยคอมพิวเตอร์

นายเฉลิมพล คำอิน รหัส 49015188
นายดิเรก ปัทมะสุคนธ์ รหัส 490151191
ว่าที่ร้อยตรีเดชฤทธิ์ จันทร์แก้ว รหัส 49015193
รศ.ดร. ชูชาติ ปิณฑวิรุจน์
ปีการศึกษา 2551

บทคัดย่อ

รายงานฉบับนี้มีวัตถุประสงค์ เพื่อศึกษาและพัฒนาคุณภาพของตู้ควบคุมอุณภูมิสำหรับเด็กทารกแรกเกิด (Infant Incubator) ถือเป็นเครื่องมือทางการแพทย์ที่สำคัญอย่างหนึ่ง ที่ช่วยปรับสภาพอุณหภูมิสภาวะแวดล้อมให้เหมาะสมกับทารกแรกเกิดที่ร่างกายไม่สมบูรณ์แข็งแรงดีพอ ไม่สามารถปรับตัวให้เข้ากับสภาวะแวดล้อมปกติได้

โครงงานนี้ได้นำเอาเทคโนโลยีของไมโครคอนโทรลเลอร์ในตระกูลM8(PSoCMicrocontroller) มาประยุกต์ใช้ในระบบควบคุมทั้งหมดของตู้อบ สามารถควบคุมอุณหภูมิภายในตู้อบได้ และสั่งงานผ่าน Computer โดยนำอัลกอรึทึมของระบบควบคุมอัตโนมัติแบบพีไอดี (PID Control) มาใช้ในการเขียนโปรแกรมควบคุมค่าอุณหภูมิ ซึ่งใช้ตัวเซ็นเซอร์ SHT 15 เป็นอุปกรณ์ตรวจวัดค่าอุณหภูมิและความชื้น การแสดงผลออกทางจอ graphic LCD display ผู้ใช้งานสามารถเห็นอุณหภูมิ ความชื้นและสัญญาณเตือนต่างๆ เช่น ระดับน้ำ และ ระดับแบตเตอรี่ ที่ตู้ควบคุมได้อย่างชัดเจนตลอดเวลา

PC- controlled Infant Incubator

Mr.Chaloempon Kam-In ID. 49015188

Mr.Direk Pattamasukhon ID. 490151191

Mr.Detrid Chankhoaw ID. 49015193

Assoc. Prof. Dr. Chuchart Pintavirooj Advisor

Educational Year 2008

Abstract

This report concerns about the design and construction of infant incubator, which is the essential medical device for a newborn where the controlled atmosphere is needed, which is used for a newborn whose frail health is caused by premature birth.

The operation of the incubator is based on using the Programmable System on Chip. The input of it includes the sending data of temperature and humidity is controlled of Computer, where the output of the microcontroller is the Touch screen graphic display and the proportionally controlled signal for the heating element. The sending data of temperature and humidity are detected by SHT15 Chip. Then show result graphic LCD display, the user can observe humidity temperature and a signal warns all such as the water level and battery level at the incubator can control all the time clearly.

กิตติกรรมประกาศ

รายงานฉบับนี้สำเร็จลุล่วงอย่างดีด้วยความกรุณาจากอาจารย์ที่ปรึกษา รศ.ดร.ชูชาติ ปิณฑวิรุจน์ ในการให้คำแนะนำในการแก้ปัญหาต่างๆ ตลอดจนการให้ความรู้ในทุกๆด้านจนทำให้ปริญญานิพนธ์ฉบับนี้สำเร็จลุล่วงตามวัตถุประสงค์

ขอขอบคุณ พี่ปริญญาโทที่ให้คำปรึกษาและความช่วยเหลืออย่างดีตลอดปีการศึกษา

สุดท้ายนี้ขอกราบขอบพระคุณ บิดา มารดา และครอบครัวที่เป็นกำลังใจ และให้การสนับสนุนในทุกเรื่องๆ จนสามารถทำให้ปริญญานิพนธ์ฉบับนี้สำเร็จลุล่วงด้วยดี

นายเฉลิมพล คำอิน
นายดิเรก ปัทมะสุคนธ์
ว่าที่ร้อยตรีเดชฤทธ์ จันทร์แก้ว

สารบัญ

สารบัญ (ต่อ)

สารบัญตาราง

สารบัญรูป

สารบัญรูป(ต่อ)

สารบัญรูป(ต่อ)

หน้า

บทที่1
บทนำ

เนื่องจากทารกแรกเกิดมีสภาพร่างกายที่อ่อนไหวต่ออุณหภูมิภายนอก เป็นผลให้ทารกแรกเกิดมีอัตราการเกิดโรค (Morbidity) และอัตราตาย (mortality) เพิ่มมากขึ้น โดยเฉพาะอย่างยิ่งในทารกที่น้ำหนักตัวน้อย ส่วนหนึ่งคงจะเกิดจากการขาดแคลนบุคลากร เช่นแพทย์ พยาบาล อีกส่วนหนึ่งที่สำคัญมากคือ ปริมาณและประสิทธิภาพทางด้านเครื่องมือต่างๆ การปรับปรุงและพัฒนาเครื่องมือต่างๆจะช่วยเพิ่มประสิทธิภาพในการรักษามากยิ่งขึ้น

เพราะฉะนั้นจึงต้องมีการควบคุมอุณหภูมิร่างกายของเด็กแรกเกิดให้มีความเหมาะสม ตู้อบเด็กทารกแรกเกิดจึงจัดเป็นเครื่องมือทางแพทย์ที่สำคัญ สำหรับช่วยทารกแรกเกิดให้รอดชีวิตและเจริญเติบโตได้อย่างปกติ

ตู้อบเด็กทารกแรกเกิดได้พัฒนาจากเครื่องที่ควบคุมด้วยระบบปฏิบัติงานเอง(manual) ถึงระบบควบคุมกึ่งอัตโนมัติ จนกระทั่งเป็นระบบควบคุมอัตโนมัติ(Automatic) การควบคุมด้วยระบบอัตโนมัติทำให้ประสิทธิภาพการทำงานของตู้อบดีขึ้น และสะดวกง่ายต่อการใช้งาน

เครื่องที่ควบคุมด้วยระบบอัตโนมัติ ได้นำเอาเทคโนโลยีไมโครโปรเซสเซอร์มาประยุกต์ใช้ควบคุมการทำงานต่างๆของระบบ เพื่อให้ตู้อบเด็กทำงานอย่างมีประสิทธิภาพสูงสุด

วัตถุประสงค์

1. เพื่อศึกษาระบบควบคุมอัตโนมัติที่มีตัวไมโครโปรเซสเซอร์ ควบคุมระบบ
2. เพื่อออกแบบระบบวัดอุณหภูมิ และความชื้น
3. เพื่อออกแบบระบบควบคุมอุณหภูมิ และความชื้นแบบอัตโนมัติ
4. เพื่อพัฒนาประสิทธิภาพตู้อบเด็กแบบเก่าให้มีประสิทธิภาพการทำงานมากขึ้น
5. เพื่อช่วยชีวิตทารกแรกเกิดให้รอดชีวิต และเจริญเติบโตได้อย่างปกติ

แนวคิดและขอบเขตของโครงงาน

โครงงานนี้จัดทำขึ้นเพื่อศึกษาและพัฒนาตู้อบเด็กทารกแรกเกิด (Infant Incubator) รุ่นเก่าให้มีประสิทธิภาพในการทำงานสูงขึ้น

ส่วนควบคุมอัตโนมัติประกอบด้วย ส่วนเซนเซอร์วัดอุณหภูมิและความชื้น ซึ่งจะทำการวัดอุณหภูมิและความชื้นตลอดเวลา และส่งค่าไปยังไมโครคอนโทรลเลอร์เพื่อประมวลผลเปรียบเทียบข้อมูลระหว่างค่าที่วัดได้กับค่าที่ต้องการควบคุม โดยค่าที่ต้องการควบคุมจะสั่งงาน

ผ่านทาง Computer ถ้าหากค่าที่วัดได้ไม่ตรงกับค่าที่ต้องการควบคุม จะสั่งการให้อุปกรณ์ควบคุมอุณหภูมิทำงาน เมื่ออุณหภูมิมีค่าตรงกับค่าที่ตั้งไว้ อุปกรณ์ควบคุมอุณหภูมิจะหยุดทำงาน และได้นำ Lab View มาใช้ในการรับและส่งข้อมูลแบบ data locker และสามารถที่จะตั้งค่าอุณหภูมิได้

ส่วนแสดงผล มีการติดตั้งจอ Graphic LCD 128 x 240 แทนจอรุ่นเก่าที่เป็น LCD Display 16x2 เพื่อแสดงค่าอุณหภูมิและความชื้นที่วัดได้ ณ เวลานั้นๆ นอกจากนี้ ยังมีระบบสัญญาณเตือนอัตโนมัติทั้งแสงและเสียง ในกรณีที่ตู้อบเด็กทารกมีความขัดข้องเกิดขึ้น เช่น ระบบไฟฟ้าขัดข้อง, อุณหภูมิเกิน, น้ำแห้ง เป็นต้น ทั้งนี้เพื่อประสิทธิภาพในการทำงานของตู้อบเด็กทารกแรกเกิด

โครงสร้างเนื้อหาของโครงงาน

บทที่ 1 บทนำ กล่าวถึง ความเป็นมาของโครงงานฯ วัตถุประสงค์ แนวคิดและขอบเขต โครงสร้างเนื้อหาโครงงาน และประโยชน์ที่คาดว่าจะได้รับจากการทำโครงงาน

บทที่ 2 ทฤษฎีที่เกี่ยวข้อง

บทที่ 3 ขั้นตอนการดำเนินงาน การออกแบบ และการสร้างระบบ

บทที่ 4 การทดลอง และผลการทดลอง

บทที่ 5 สรุป วิจารณ์ผลการทดลอง และข้อเสนอแนะ

ประโยชน์ที่คาดว่าจะได้รับ

- สามารถนำเทคโนโลยีไมโครคอนโทรลเลอร์มาประยุกต์ใช้กับชิ้นงานได้
- เกิดประสบการณ์เรียนรู้ และเข้าใจ การทำงานหลักของตู้อบเด็กทารกแรกเกิด
- สามารถนำระบบควบคุมแบบ PID มาประยุกต์ใช้กับการควบคุมอุณหภูมิ และความชื้น
- สามารถออกแบบระบบวัดต่างๆ และควบคุมอุณหภูมิ ความชื้นเพื่อใช้งานกับระบบกึ่งอัตโนมัติหรือแบบอัตโนมัติให้มากที่สุดเท่าที่ จะเป็นได้
- ลดอัตราการเกิดโรค (Morbidity) และอัตราตาย (mortality) ในทารกแรกเกิด
- เป็นข้อมูลสนับสนุนผู้วิจัย เพื่อใช้ประโยชน์ในการพัฒนาเทคโนโลยีทางด้านเครื่องมือและอุปกรณ์ทางการแพทย์

บทที่ 2
ทฤษฎี

ทฤษฎีแบ่งออกเป็น 2 ส่วนดังนี้

2.1 ทฤษฎีเกี่ยวกับทารกแรกเกิด

2.1.1 สรีระวิทยาของทารกแรกเกิด

2.1.2 ระบบหายใจ

2.1.3 การพยาบาลที่จำเป็นจะต้องปฏิบัติต่อทารกแรกเกิด

2.1.4 การดูแลทารกแรกเกิด

2.1.5 การควบคุมอุณหภูมิกายทารกแรกเกิด

2.1.6 การเลือกใช้ตู้อบ

2.1.7 อุณหภูมิตู้อบที่เหมาะสมกับอายุและน้ำหนักของทารกแรกเกิด Neutral thermal environmental temperature

2.2 ระบบควบคุมของตู้อบเด็กทารกแรกเกิด

2.2.1 เซนเซอร์ตรวจจับอุณหภูมิและความชื้น

2.2.2 การควบคุมอุณหภูมิ ความชื้นและการแสดงผล

2.2.3 ระบบควบคุมแบบอัตโนมัติชนิด PID

2.2.4 เฟสคอลโทรล Phase Control IC

2.2.5 Graphic LCD ABG240128

2.1 ทฤษฎีเกี่ยวกับทารกแรกเกิด

ทารกแรกเกิด (Newborn Infant) หมายถึง ทารกแรกเกิดจนถึงอายุ 1 เดือน ซึ่งถือว่าเป็นช่วงที่เสี่ยงอันตรายมาก แบ่งเป็น 3 ประเภทดังนี้

- ทารกที่คลอดครบกำหนด (Full Term Baby) หมายถึง ทารกที่คลอดจากครรภ์มารดา ในระหว่าง 38-42 สัปดาห์ ทารกประเภทนี้จะเติบโตเต็มที่และสามารถมีชีวิตอยู่ได้

- ทารกที่คลอดก่อนกำหนด (Pre Term Baby) หมายถึง ทารกที่คลอดจากครรภ์มารดาก่อนการตั้งครรภ์ครบ 37 สัปดาห์ และมีน้ำหนักตัวแรกคลอดต่ำกว่า 2500 กรัม

- ทารกคลอดเกินกำหนด (Post Term Baby) หมายถึง ทารกที่คลอดจากครรภ์มารดาภายหลังการตั้งครรภ์ 42 สัปดาห์

ทารกทั้ง 3 ประเภทนี้มีความแตกต่างกันทั้งรูปร่าง ลักษณะ การเจริญเติบโต และความต้าน

ทานโรค โดยเฉพาะทารกที่คลอดเกินกำหนด จะมีความต้านทานโรคน้อย มีอันตราย (Mortality rate) สูงกว่า และมีอันตรายระหว่างคลอด (Birth Injury) ได้มากกว่าทารกที่คลอดก่อนกำหนด

2.1.1 สรีระวิทยาของทารกแรกเกิด

ทารกเมื่ออยู่ในครรภ์มารดาจะถูกห่อหุ้มเลี้ยงดูโดยไม่ต้องหายใจและกินอาหาร เนื่องจากได้รับอาหารและออกซิเจนผ่านทางสายสะดือ ทันทีที่ทารกคลอดพ้นครรภ์มารดาสู่โลกภายนอก ซึ่งสภาพแวดล้อมต่างกัน ทารกจำเป็นต้องปรับตัวเพื่อให้มีชีวิตอยู่รอด

2.1.2 ระบบหายใจ

เป็นระบบแรกที่ทำงานทันทีเมื่อคลอด ปอดของทารกขณะอยู่ในครรภ์มารดามีลักษณะแข็งแฟบไม่ทำงาน ถุงลมปอดไม่พองตัว การหายใจของทารกจะเกิดขึ้นเมื่อร่างกายขาดออกซิเจน และมีคาร์บอนไดออกไซด์คั่งในกระแสโลหิต และไปกระตุ้นศูนย์ควบคุมการหายใจให้เริ่มหายใจ

นอกจากนี้การที่บริเวณทรวงอกของทารกถูกบีบรัดขณะคลอด และสัมผัสเสียดสีบริเวณแขนขา มือ เท้า จะเป็นการกระตุ้นให้ทารกหายใจและร้องไห้ การร้องไห้จะช่วยให้ปอดขยายตัวดีขึ้น อัตราการหายใจเมื่อแรกคลอดใหม่ๆ ประมาณ 35-50 ครั้งต่อนาที เฉลี่ย 40 ครั้งต่อนาที ถ้าเกิน 60 ครั้งต่อนาทีแสดงว่าผิดปกติ

2.1.3 การพยาบาลที่จำเป็นจะต้องปฏิบัติต่อทารกแรกเกิด

การพยาบาลที่จำเป็นจะต้องปฏิบัตินี้ เป็นสิ่งที่ผู้ให้การพยาบาลจำเป็นจะต้องปฏิบัติเพื่อช่วยเหลือให้ทารกสามารถมีชีวิตอยู่รอด และเจริญเติบโตได้ตามปกติ โดยผู้ปฏิบัติจะต้องยึดหลักต่อไปนี้

1. ช่วยให้ทารกหายใจได้สะดวก และรับออกซิเจนเพียงพอโดย

ช่วยให้ระบบทางเดินหายใจของทารกปราศจากสิ่งอุดตัน โดยการดูดจมูก เมือก หรือโลหิต ที่อุดตันอยู่ในปาก คอ จมูก ของทารกออกได้หมด อาจใช้ลูกสูบยางดูดออก หรือใช้สายสอดลงไปดูดในรายที่อยู่ลึก การจับให้ศีรษะของทารกห้อยต่ำลงก็เป็นการช่วยให้ของเหลวที่อุดตันไหลออกจากปากและจมูกได้สะดวก (ยกเว้นในรายที่ทารกได้รับความกระทบกระเทือนที่ศีรษะ หรือมีการตกโลหิตในสมอง ไม่ควรจับศีรษะห้อยต่ำลง)

กระตุ้นให้ทารกร้อง เพื่อช่วยให้ถุงลมของปอดขยายตัวทำให้ทารกหายใจโดยการเขี่ยเบาๆที่ฝ่าเท้า หรือตบเบาๆที่ก้น

ทารกบางรายที่ไม่หายใจ เนื่องจากศูนย์ควบคุมการหายใจทำงานไม่ปกติทารกหยุดหายใจ อาจให้การช่วยเหลือโดยใส่ท่อเอนโดเครเทียล (Endotracheal Tube) และผายปอดโดยใช้ความกดดันลงในปอดทารกโดยตรง (Positive Pressure) ใช้แรงดันประมาณ 15-20 เซนติเมตรน้ำ ไม่ควรใช้แรงดันที่สูงเกินไป เพราะจะทำให้ปอดฉีกขาด การผายปอดโดยวิธีเป่าลมเข้าทางปากทารก (Mouth to Mouth) ก็เป็นวิธีที่ใช้ได้ผลดีในทารกใหม่ ควรหลีกเลี่ยงการใช้ยากระตุ้นหายใจ แต่อาจใช้ยาจำพวกกระตุ้นการไหลเวียนโลหิตได้บ้าง

การให้ออกซิเจน ทารกคลอดใหม่ที่มีอาการแสดงว่าขาดออกซิเจน เช่น หายใจลำบาก ปากเขียว เล็บเขียว ผิวคล้ำ หรือซีดขาว ควรให้ออกซิเจนไว้จนกว่าจะดีขึ้นคือผิวหน้าของทารกมีสีแดง การหายใจสม่ำเสมอเป็นปกติ วิธีให้ออกซิเจนทารกคลอดใหม่อาจจะให้ทางสายยางหรือใช้หน้ากากครอบจมูก แต่วิธีที่ให้ทางสายยางถ้าให้ออกซิเจนแก่ทารกเกินกว่า 40 เปอร์เซ็นต์ เพราะอาจทำให้ทารกตาบอดได้ โดยเฉพาะในทารกคลอดก่อนกำหนด

2. การให้ความอบอุ่น ทารกหลังคลอดแล้วอุณหภูมิของร่างกายจะลดลงทันที และจะกลับคืนสู่ปกติประมาณ 8 ชั่วโมงหลังคลอด เพื่อช่วยให้ทารกสามารถปรับอุณหภูมิของร่างกายได้ปกติ ควรห่อหุ้มทารกให้อบอุ่นด้วยผ้านุ่มๆ ทันทีหลังคลอด อุณหภูมิของห้องคลอดควรประมาณ 36.6 องศาเซลเซียส

3. การแยกเด็กและติดป้ายชื่อ ทารกคลอดใหม่ก่อนที่จะแยกออกไปจากเตียงคลอดต้องติดป้ายชื่อมารดาและทารก ที่ป้ายชื่อจะต้องบอก ชื่อ นามสกุล เพศ น้ำหนักทารกวันเวลาคลอดและควรอุ้มทารกให้มารดาได้เห็นหน้าและเพศอย่างชัดเจนอีกครั้ง ก่อนที่จะแยกทารกไปจากมารดาและตรวจดูป้ายชื่อทั้งมารดาและทารกให้ตรงกันอีกครั้ง

4. ป้องกันการตกโลหิตในทารกคลอดใหม่ ซึ่งบางแห่งจะฉีดวิตามินเค 0.5 มิลลิกรัมเข้ากล้ามเนื้อให้ทารกแรกคลอด

2.1.4 การดูแลทารกแรกเกิด

หลักการดูแลที่สำคัญมี 7 ประการ ดังนี้

1. การดูแลอุณหภูมิกายทางทวารหนักให้อยู่ที่ 37 °C
2. การดูแลทางเดินหายใจให้โล่ง และออกซิเจนในเลือดปกติ
3. การป้องกันการติดเชื้อ
4. การให้อาหาร ได้แก่ นมแม่ หรือสารน้ำในกรณีที่รับนมไม่ได้
5. การรักษาเฉพาะโรค เมื่อมีการเจ็บป่วย
6. การส่งเสริมการสร้างสายสัมพันธ์ (maternal-infant bonding)
7. การดูแลด้านพัฒนาการ (development care)

2.1.5 การควบคุมอุณหภูมิร่ากายทารกแรกเกิด

การพยาบาลมีบทบาทสำคัญในการป้องกันการสูญเสียความร้อน และควบคุมอุณหภูมิกาย ของทารก โดยมีหลักปฏิบัติดังนี้

1. การพยาบาลทั่วไป

ทันทีที่คลอดรีบเช็ดผิวให้แห้ง ใส่เสื้อผ้า ห่มผ้า และให้ทารกนอนในเตียงที่อบอุ่น

วัดอุณหภูมิกายทารกทุก15 นาที จนกว่าอุณหภูมิจะปกติ และเปลี่ยนวัดทุก 4 ชั่วโมง

ถ้าอุณหภูมิกายทารกต่ำกว่าปกติคือ ต่ำกว่า36.5องศาเซลเซียส ไม่ต้องอาบน้ำทารก ควร ทำความสะอาดร่างกายเฉพาะที่ เช่น หน้าและก้นก็พอ และห่มผ้าให้อบอุ่น

ถ้าอุณหภูมิกายทารกยังต่ำ อยู่พิจารณาใช้ผ้าห่มไฟฟ้าหรือโคมไฟฟ้า ตั้งส่องที่ทารก ระวัง อันตรายจากความร้อน (Burn) และแสงไฟเข้าตาโดยตรง จะเกิดระคายเคืองต่อตาทารกได้

ถ้าอุณหภูมิห้องเย็นมากโดยเฉพาะฤดูหนาวต้องใช้เครื่องทำความร้อน(Heater) ช่วย

ถ้าทารกยังไม่สามารถควบคุมอุณหภูมิกายได้ควรให้อยู่ในตู้อบจนกว่าอุณหภูมิจะปกติ

2. การใช้ตู้อบเด็ก ทารกแรกเกิดที่จำเป็นต้องนำเข้าตู้อบเด็ก ได้แก่

- น้ำหนักแรกเกิดน้อยกว่า 2,500 กรัม หรือ มากกว่า 4,000 กรัม
- เกิดก่อนอายุครรภ์ 37 สัปดาห์ หรือหลังอายุครรภ์ 42 สัปดาห์
- น้ำหนักผิดปกติเมื่อเทียบกับอายุครรภ์
- มีภาวะขาดออกซิเจน หายใจเร็ว หรือตัวเขียว
- พิการแต่กำเนิดรุนแรง
- ภาวะซีด ภาวะเม็ดเลือดแดงเกิน จ้ำเลือด

2.1.6 การเลือกใช้ตู้อบ

การเลือกใช้ตู้อบ มีหลักควรพิจารณาคุณสมบัติใหญ่ๆ คือ

- สามารถควบคุมอุณหภูมิและความชื้นให้คงที่ได้ง่าย
- ให้ออกซิเจนได้ง่าย
- ฝาครอบใสมองเห็นทารกได้ชัดเจนจากภายนอกตู้อบ
- มีสัญญาณเตือน เมื่อภายในตู้อบร้อนเกินไป หรือมีการถ่ายเทอากาศไม่เพียงพอ
- สามารถปรับระดับของที่นอนได้

2.1.7 อุณหภูมิตู้อบที่เหมาะสมกับอายุและน้ำหนักของทารกแรกเกิด ดังตารางที่ 2.1

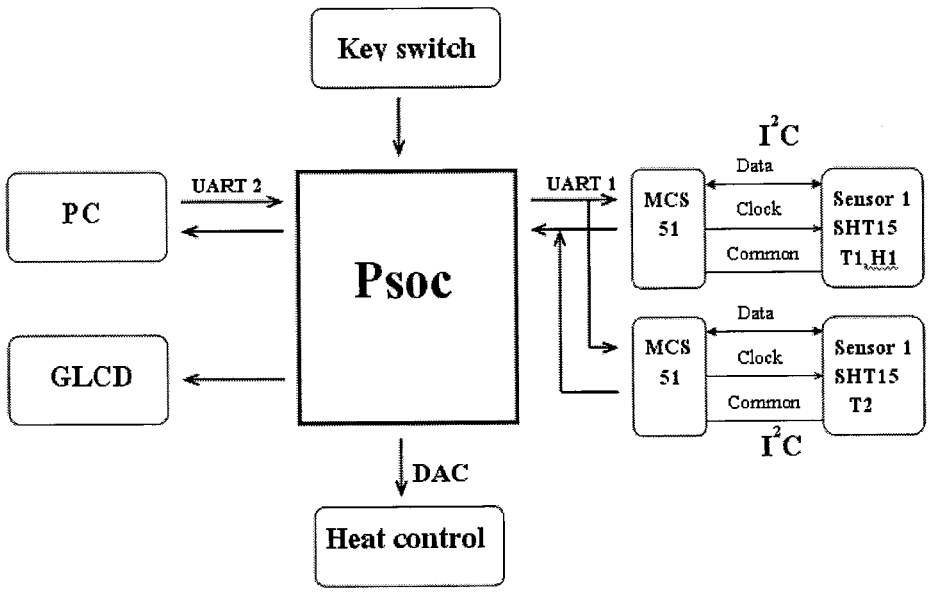
ตารางที่ 2.1 แสดงค่าอุณหภูมิสิ่งแวดล้อมที่เหมาะสมกับอายุและน้ำหนักของทารกแรกเกิด

อายุ/น้ำหนัก (กรัม)	อุณหภูมิ (°C)	อายุ/น้ำหนัก (กรัม)	อุณหภูมิ (°C)
0-6 ชั่วโมง		72-96 ชั่วโมง	
ต่ำกว่า 1200	34.0-35.4	ต่ำกว่า 1200	34.0-35.0
1200-1500	33.9-34.4	1200-1500	33.0-34.0
1501-2500	32.8-33.8	1501-2500	33.1-33.2
เกิน 2500 (และ> 36 สัปดาห์)	32.0-33.8	เกิน 2500 (และ> 36 สัปดาห์)	29.8-32.8
6-12 ชั่วโมง		4-12 วัน	
ต่ำกว่า 1200	34.0-35.4	ต่ำกว่า 1500	33.0-34.0
1200-1500	33.5-34.4	1501-2500	31.0-33.2
1501-2500	32.2-33.8	เกิน 2500 (และ> 36 สัปดาห์)	
เกิน 2500 (และ> 36 สัปดาห์)	31.4-33.8	4-5 วัน	29.5-32.6
12-24 ชั่วโมง		5-6 วัน	29.4-32.3
ต่ำกว่า 1200	34.0-35.4	6-8 วัน	29.0-32.2
1200-1500	33.3-34.3	8-10 วัน	29.0-31.8
1501-2500	31.8-33.8	10-12 วัน	29.0-31.4
เกิน 2500 (และ> 36 สัปดาห์)	31.0-33.7	12-14 วัน	
24-36 ชั่วโมง		ต่ำกว่า 1500	32.6-34.0
ต่ำกว่า 1200	34.0-35.0	1501-2500	31.0-33.2
1200-1500	33.1-34.2	เกิน 2500 (และ> 36 สัปดาห์)	29.0-30.8
1501-2500	31.6-33.6	2-3 สัปดาห์	
เกิน 2500 (และ> 36 สัปดาห์)	30.7-33.5	ต่ำกว่า 1500	32.2-34.0
36-48 ชั่วโมง		1501-2500	30.5-33.0
ต่ำกว่า 1200	34.0-35.0	3-4 สัปดาห์	
1200-1500	33.0-34.1	ต่ำกว่า 1500	31.6-33.6
1501-2500	31.4-33.5	1501-2500	30.0-32.7
เกิน 2500 (และ> 36 สัปดาห์)	30.5-33.3	4-5 สัปดาห์	
48-72 ชั่วโมง		ต่ำกว่า 1500	31.2-33.0
ต่ำกว่า 1200	34.0-35.0	1501-2500	29.5-32.2
1200-1500	33.0-34.0	5-6 สัปดาห์	
1501-2500	31.2-33.4	ต่ำกว่า 1500	30.6-32.3
เกิน 2500 (และ> 36 สัปดาห์)	30.1-33.2	1501-2500	29.0-31.8

การพิจารณาว่าทารกสามารถออกจากตู้อบเด็กมาอยู่ในอุณหภูมิห้องได้ พิจารณาจากการสังเกตอุณหภูมิตู้อบที่ทารกต้องการเพื่อรักษาอุณหภูมิกายให้ปกติ หากสามารถลดอุณหภูมิตู้อบลงจนใกล้เคียงกับอุณหภูมิห้องที่ทารกจะอยู่หลังออกจากตู้อบ และทารกยังมีอุณหภูมิกายปกติ ให้นำทารกออกจากตู้อบ สวมเสื้อผ้าและห่มผ้า ต่อจากนั้นให้ติดตามอุณหภูมิร่างกาย อย่างใกล้ชิด โดยวัดอุณหภูมิกายทุกครึ่งชั่วโมง จนแน่ใจว่าทารกสามารถรักษาอุณหภูมิกายได้เอง จึงติดตามอุณหภูมิกายตามปกติ

2.2 ระบบควบคุมของตู้อบเด็กทารกแรกเกิด

ผังการทำงานหลักของตู้อบเด็กทารกแรกเกิด แสดงได้ดังนี้



รูปที่ 2.1 บล็อกไดอะแกรมของระบบ

ตู้อบเด็กทารกแรกเกิด ประกอบด้วย

- ส่วนตัวประมวลผลควบคุม

เป็นส่วนที่ทำหน้าที่ควบคุมการทำงานทุกส่วนของตู้อบเด็กทารกแรกเกิด ตั้งแต่การรับข้อมูลจากส่วนอินพุทของตัวตู้เข้ามายังตัวประมวลผล และทำการส่งข้อมูลที่ทำการประมวลผลแล้วส่งออกไปยังส่วนแสดงผล

- ส่วนวัดอุณหภูมิและความชื้น

ตรวจจับค่าอุณหภูมิและความชื้นโดยใช้เซนเซอร์ SHT15 ซึ่งสามารถวัดได้ทั้งอุณหภูมิและความชื้นในตัวเดียวกัน โดยให้เอาท์พุทเป็นสัญญาณดิจิตอลส่งผ่านมายังตัวประมวลผลเพื่อประมวลผล

- ส่วนควบคุม

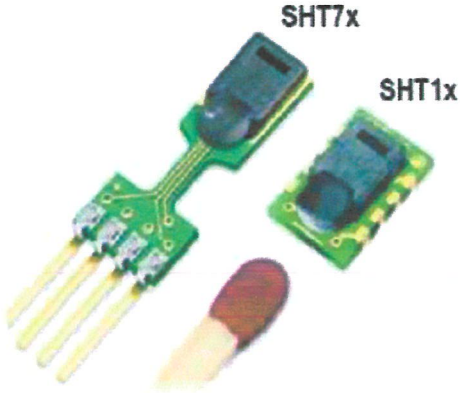
รับการตั้งค่าอุณหภูมิที่กำหนดการทำงานให้กับตู้อบเด็กทารกแรกเกิด เพื่อส่งไปยังตัวประมวลผลควบคุมการทำงานของตู้อบ

- ส่วนแสดงผล

รับค่าที่ได้จากส่วนประมวลผลส่งไปแสดงผ่านทางจอ Touch screen ของคอมพิวเตอร์

2.2.1 เซนเซอร์วัดอุณหภูมิและความชื้น

ส่วนวัดผลทางอุณหภูมิจะวัดค่าอุณหภูมิโดยใช้เซนเซอร์ SHT15 ซึ่ง SHT15 สามารถใช้เป็นเซนเซอร์วัดได้ทั้งอุณหภูมิและความชื้น มีตัวแปลงสัญญาณอนาลอกไปเป็นสัญญาณดิจิตอล (A/D) ภายในตัว จึงให้ค่าเอาท์พุทเป็นข้อมูลดิจิตอล ซึ่งค่าที่ได้ต้องนำมาผ่านการคำนวณตามสมการมาตรฐานของอุปกรณ์ จึงจะได้ค่าเอาท์พุทที่แท้จริงเป็นค่าของอุณหภูมิและความชื้น

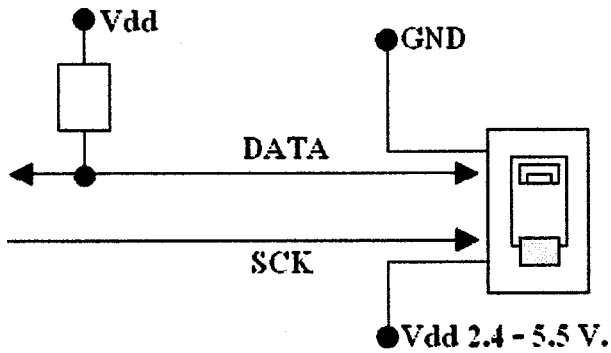


รูปที่ 2.2 รูป IC SHT1X และ SHT7X

ในโครงงานนี้เราเลือกใช้ SHT15 มีคุณสมบัติดังนี้

- มี Package แบบ LCC (Leadless Chip Carrier)
- สามารถวัดได้ทั้งอุณหภูมิและความชื้น

- สามารถวัดอุณหภูมิได้ตั้งแต่ 0-125°C ความละเอียดในการวัด 0.1°C
- สามารถวัดความชื้นสัมพัทธ์ได้ตั้งแต่1-99.9%RH ความละเอียดในการวัด0.1%RH
- ใช้แหล่งจ่ายไฟ +5 V กินกระแสต่ำ
- ใช้สัญญาณในการควบคุม 2 เส้น คือ DATA และ CLOCK
- มีความแม่นยำในการวัดอุณหภูมิที่ 0.5°C และความชื้นที่ 2.0%RH



รูปที่ 2.3 แสดงลักษณะของ SHT15

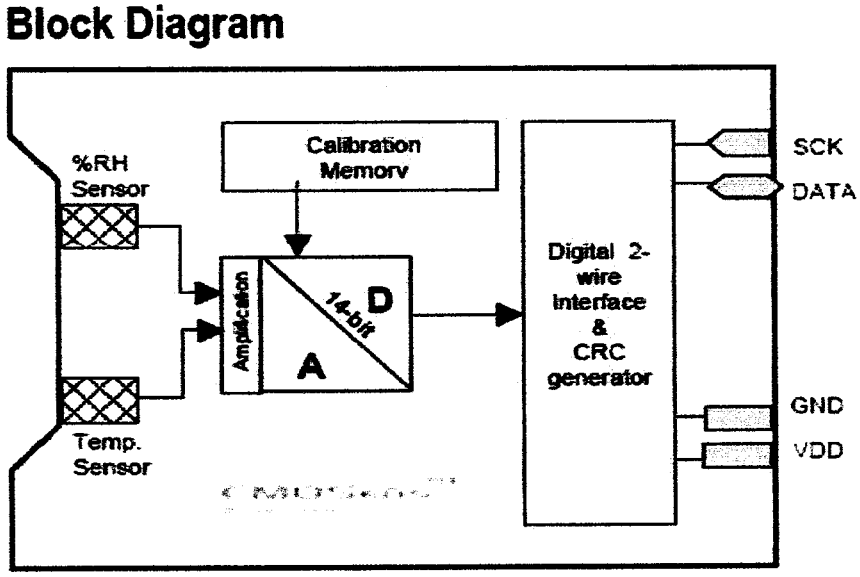
ตารางที่ 2.2 แสดงคุณสมบัติ DC ของ SHT15

Parameter	**Conditions**	**Min.**	**Type**	**Max.**	**Units.**
Power supply DC		2.4	5	5.5	V
Supply current	Measuring		550		µA
	Average	2[2]	28[3]		µA
	Sleep		0.3	1	µA
Low level output voltage		0		20%	Vdd
High level output voltage		75%		100%	Vdd
Low level input voltage	Negative going	0		20%	Vdd
High level input voltage	Positive going	80%		100%	Vdd
Input current on pads				1	µA
Input peak current	On			4	mA
	Tractate (off)		10		µA

ตารางที่ 2.3 แสดงขาสัญญาณของ SHT15

Pin	Name	Comment
1	GND	Ground
2	DATA	Serial data , bi-directional
3	SCK	Serial clock , input
4	VDD	Supply 2.4 – 5.5 V
	NC	Remaining pins must be left unconnected

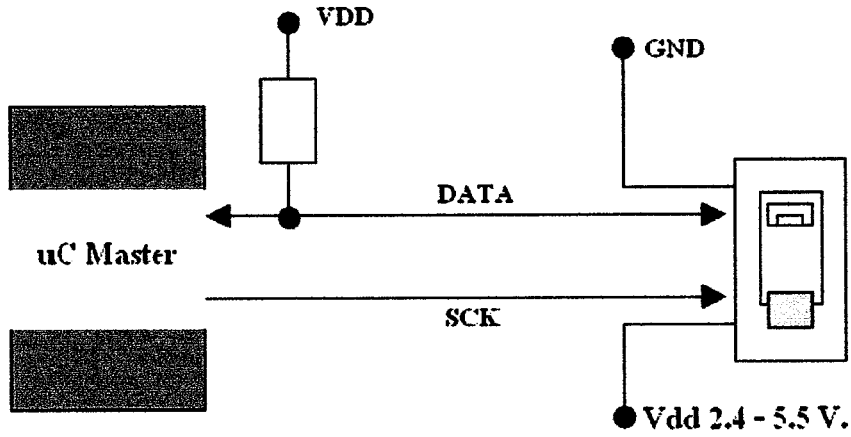
ลักษณะการทำงานของ SHT15 จะแสดงได้ดังรูปที่ 2.4



รูปที่ 2.4 แสดง Block Diagram ของ SHT15

- การต่อขา Vcc กับ GND ต้องต่อไฟเลี้ยงให้อยู่ระหว่าง 2.4- 5.5 V แล้วหลังจาก ที่จ่ายไฟเข้าที่ตัว SHT15 แล้ว SHT15 จะใช้เวลาประมาณ 11ms เพื่อเข้าสู่โหมด Sleep ดังนั้นต้องส่งข้อมูลก่อนที่ IC จะเข้าสู่โหมด Sleep ในการต่อ Vcc กับ GND ควรที่จะต่อ C 100 nF คร่อมระหว่างขา Vcc กับ GND

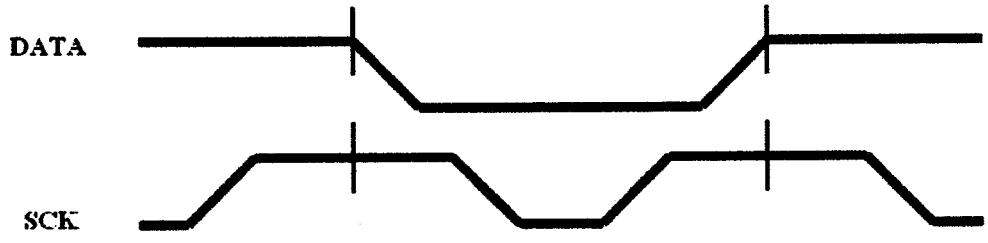
- การต่อขา DATA และขา SCK จะต่อแบบ Serial Interface (Bi-directional 2 –wire) ซึ่งจะเป็นการต่อในลักษณะที่คล้ายกับ I^2C แต่ไม่เหมือน I^2C โดยทั่วไป การนำไปใช้งานจะต่อวงจรลักษณะดังรูป 2.5



รูปที่ 2.5 แสดงการต่อ SHT15 กับ ไมโครคอนโทรเลอร์

- การต่อ ขา SCK จะเป็นการต่อแบบตรงระหว่างไมโครคอนโทลเลอร์กับSHT15

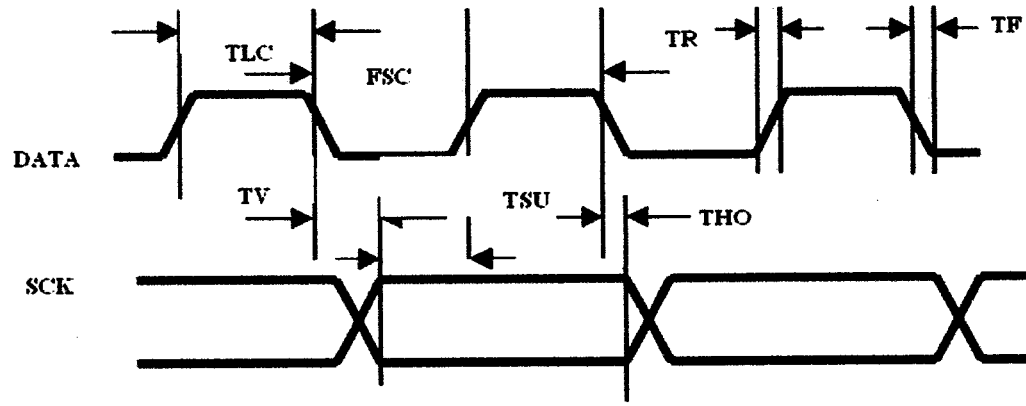
- การต่อขา DATA จะมีการต่อ Pull-up เพื่อให้ได้สัญญาณที่มีค่าสูง ซึ่งการต่อ Pull-up จะต่อกันบ่อยๆในการใช้ ไมโครคอนโทลเลอร์ และในช่วงที่ทำการส่ง DATAจำเป็นที่จะต้องทำให้ DATA มีความเสถียรในขณะที่ SCK high ซึ่งแสดง Timing Diagram ได้ดังรูป2.6



รูปที่ 2.6 แสดง Timing Diagram ในช่วงการส่งข้อมูลของ SHT15

ตารางที่2.4 แสดงคุณสมบัติของสัญญาณ SCK และ DATA

	Parameters	Conditions	Min.	Type.	Max.	Units.
F_{SCK}	SCK frequency	VDD> 4.5 V			10	MHz
		VDD< 4.5 V			0	MHz
T_{RFO}	DATA fall time	Output load 5 pF	3.5	10	20	ns
		Output load100 pF	30	40	200	ns
T_{CLx}	SCK h/l time		100			ns
T_V	DATA valid time			250		ns
T_{SU}	DATA setup time		100			ns
T_{HO}	DATA hold time		0	10		ns
T_R/T_F	SCK rise/fall time			200		ns



รูปที่ 2.7 แสดงคุณสมบัติของTiming Diagram ของสัญญาณ DATA และ SCK ของSHT15

PIN Assignment

1). Ground (GND)

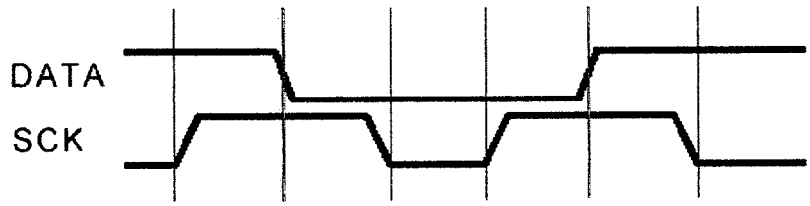
2). Serial data DATA (DATA)

3). Serial Clock Input (SCK) - สัญญาณนาฬิกาสำหรับ synchronizes กันระหว่างตัวแม่ กับ sensor โดยจะอ่านข้อมูลที่ขอบขาขึ้นของสัญญาณ

4). VDD (+2.4-5.5V) - เป็นสายข้อมูล โดยขานี้จะต้องต่อกับ R pull up 4.7K-10K

Command sequence

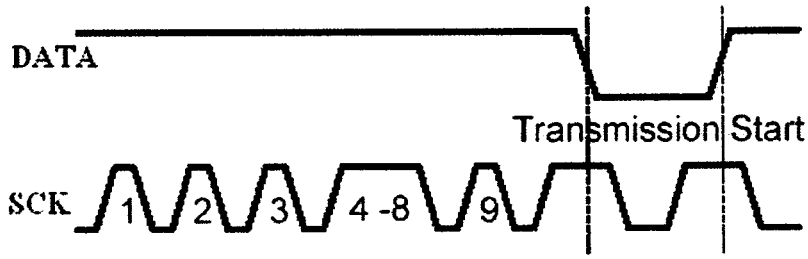
- Transmission Start



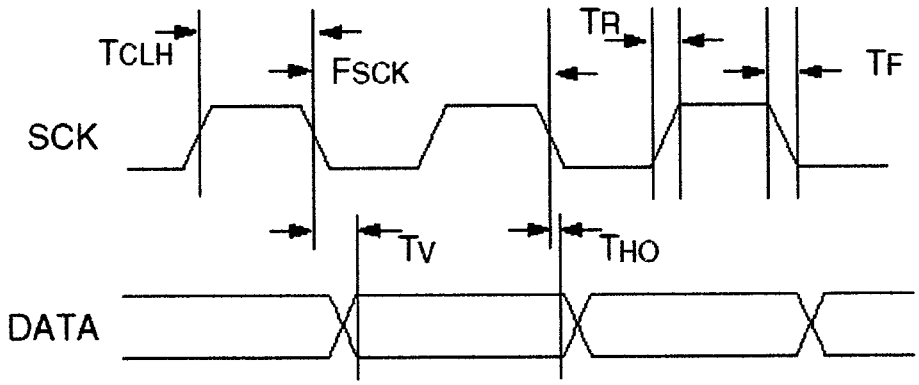
รูปที่ 2.8 แสดงลักษณะเงื่อนไขของสัญญาณมีลักษณะเป็นดังนี้

- Data เปลี่ยนจาก 1 เป็น 0 ขณะที่ SCK ลูกแรก เป็น 1
- Data เปลี่ยนจาก 0 เป็น 1 ขณะที่ SCK ลูกที่2 เป็น 1

- Connection reset sequence



รูปที่ 2.9 การแสดงเมื่อขาดการติดต่อกับอุปกรณ์ ให้ส่ง SCK ไปอย่างน้อย 9 ลูก ขณะที่ Data เป็น 1แล้วตามด้วย Transmission Start



รูปที่ 2.10 แสดงส่วนของ I/O Characteristics

ตารางที่ 2.5 แสดงค่าพารามิเตอร์ต่างๆ

	Parameter	Conditions	Min	Typ.	Max.	Unit
FSCK	SCK frequency	VDD > 4.5 V			10	MHz
		VDD < 4.5 V			1	MHz
TRFO	DATA fall time	Output load 5 pF	3.5	10	20	ns
		Output load 100 pF	30	40	200	ns
TCLH	SCK high time		100			ns
TCLL	SCK low time		100			ns
TV	DATA valid from			50		ns
THO	Output hold time		0	10		ns
TR/TF	SCK rise/fall time				200	ns

ตารางที่ 2.6 แสดงค่า List of commands

Command	Code	Describtion
Reserved	0000x	Reserved
Measure Temperature	00011	Temperature measurement
Measure Humidity	00101	Humidity measurement
Status Register Read	00111	Read access to the status register (see application note)
Status Register Write	00110	Write access to the status register (see application note)
Reserved	0101x-1110x	Reserved
Soft reset	11110	resets the chip, clears the status register to default values
		wait 11ms before next command

SHT1x list of commands

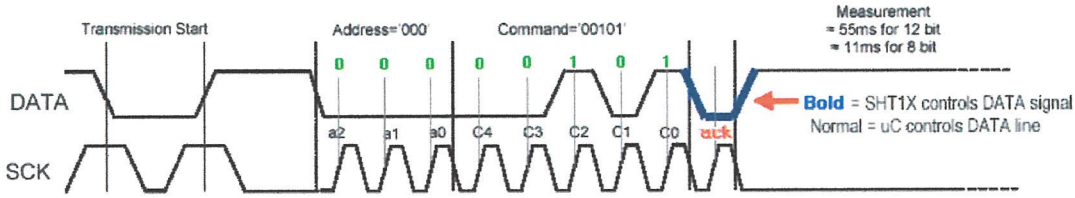
การอ่านข้อมูลจาก Sensor

รูปที่ 2.11 ชุดคำสั่งประกอบด้วย Transmission Start+Address+Command

โดยที่ Address= 000 3 +Command 5 bit

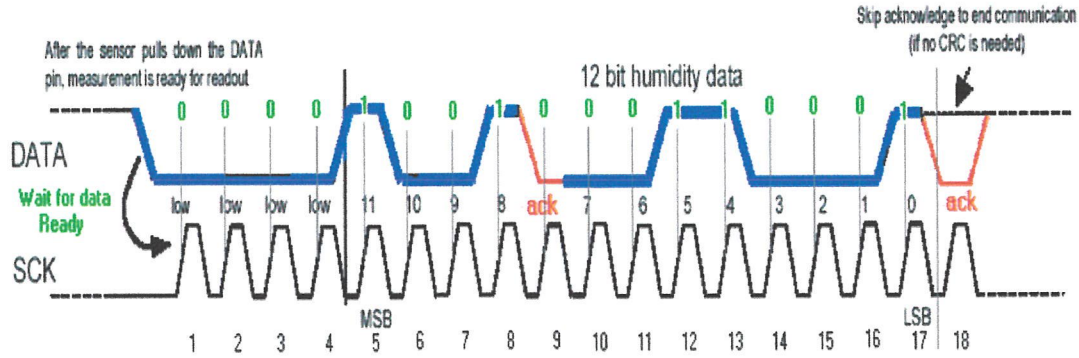
- Measurement sequence (T and RH)

การอ่านค่าอุณหภูมิและความชื้นสัมพัทธ์ ตัวอย่างเมื่อไมโครฯ ต้องการอ่านค่า Humidity จาก sensor ซึ่งมี address =000 และคำสั่ง=00101 จะมีชุดคำสั่งดังนี้



รูปที่ 2.12 การแสดงลักษณะสัญญาณในการอ่านข้อมูลจาก Sensor

เมื่อ Sensor รับทราบคำสั่งแล้วจะส่ง acknowledge (ACK) ด้วยการดึงขา Data ลงเป็น 0 (เส้นทึบ)



รูปที่ 2.13 แสดงลักษณะสัญญาณในการอ่านข้อมูลT/H 2 byte จาก Sensor

-ข้อมูลของ temperature มีขนาด 14bit และ humidity มีขนาด 12bit (สามารถเปลี่ยนได้เป็น 12 และ 8 bit โดย status register.)

- เมื่อได้รับ acknowledge แล้วให้ไมโครฯ รอสักครู่ประมาณ >210 ms เพื่อให้ Sensor พร้อม แล้วจึงส่งสัญญาณ SCK ต่อไปอีก 2 byte สำหรับรับข้อมูลและ 1 byte สำหรับ ข้อมูลตรวจสอบ ความผิดพลาด (CRC)

จากตัวอย่างจะสามารถอ่านข้อมูล 12 bit ของ Humidity (4 bit แรกเป็น 0 เสมอ) ได้เป็น 0000 1001 0011 0001= 2353 (dec)= 75.79%RH

เมื่อได้ข้อมูลครบแล้วหากต้องการ CRC ให้ตอบ acknowledge ด้วยการดึงขา Data ลงเป็น 0 หากไม่ต้องการก็ข้ามขั้นตอนนี้ไป

ตารางที่2.7 การหาค่า d1, d2 ขนาด 14 bit ที่แรงดัน 5V

	Celsius	
SOT	$d1$	$d2$
14bit5V	-40	0.01

- การเปลี่ยนข้อมูล digital

การเปลี่ยนข้อมูล Digital เป็นข้อมูล temperature ค่าของข้อมูล temperature มีลักษณะเป็นเชิงเส้นสามารถคำนวณได้โดยใช้สูตร

1) คำนวณหา Temperature โดยใช้สูตร

$$Temperature = d_1 + d_2 * SO_T \qquad (2.1)$$

โดยที่: SO_T คือ Serial Output Temperature และ d_1, d_2 คือ ค่าคงที่เชิงเส้น

เช่น อ่านข้อมูล Digital ขนาด 14 bit ได้ SO_T = 011010 11111111B = 06911 decimal ที่แรงดัน 5 V เมื่อต้องการอ่านค่าเป็นองศา Celsius จะได้ค่า d_1 = -40, d_2 = 0.01

ผลการคำนวณ Temperature = 29.11C

การเปลี่ยนข้อมูล Digital เป็นข้อมูล humidity ค่าของข้อมูล humidity มีลักษณะไม่เป็นเชิงเส้น สามารถคำนวณได้ตามขั้นตอนดังนี้

2) คำนวณหา RH-Linear โดยใช้สูตร

$$RH_{linear} = c_1 + c_2 \cdot SO_{RH} + c_3 \cdot SO_{RH}^{\ 2} \qquad (2.2)$$

โดยที่: RH คือ Serial Output humidity แบบ Linear และ c_1, c_2, c_3 มีค่าเป็น

$c_1 = -4$, $c_2 = 0.0405$, $c_3 = -2.8 \times 10^{-6}$ สำหรับ 12 bit SO_{RH}

3) เมื่อได้ ค่า RH-linear และ Tc แล้ว หาค่า RH true โดยใช้สูตร

$$RH_{true} = (T_{oc} - 25) \cdot (t_1 + t_2 \cdot SO_{RH}) + RH_{linear} \qquad (2.3)$$

โดยที่: t_1 = 0.01, t_2 = 0.00008, t_2 = 0.00128 สำหรับ 8 bit SO_{RH}

เช่น อ่านข้อมูล Digital ขนาด 12 bit ได้ RH-linear = 0110 11111111B =1791 dec ที่แรงดัน 5 V อ่านค่า

Tc = 29.11C (จาก 3.3.1) เมื่อต้องการหาค่า humidity จะได้ค่า C1= -4 ,C2 = 0.0405 ,

C3 = -0.0000028

ผลการคำนวณค่า Humidity = 60.18%

2.2.2 ส่วนควบคุมอุณหภูมิและความชื้น

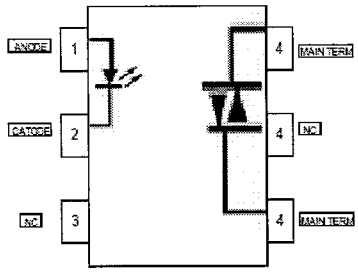
ขดลวดความร้อน(Heater) ที่นำมาใช้จะกินกำลังงาน 1000 วัตต์ การทำงานของขดลวดจะเป็นการควบคุมทางเฟส ซึ่งขดลวดที่นำมาใช้จะเป็นขดลวดนิโครมที่ให้ความร้อนได้เร็วทั้งยังสามารถที่จะระบายความร้อนได้ดีกว่าขดลวดรุ่นเก่าทำให้เราสามารถที่จะควบคุมอุณหภูมิได้ง่าย

โดยระบบการทำงานของตัวไมโครคอนโทรลเลอร์คือส่งสัญญาณไปควบคุมเฟสของการทำงานขดลวดความร้อน ซึ่งวงจรควบคุมการทำงานของขดลวดความร้อนประกอบไปด้วย

1. MOC3021
2. BTA24 600B

MOC3021

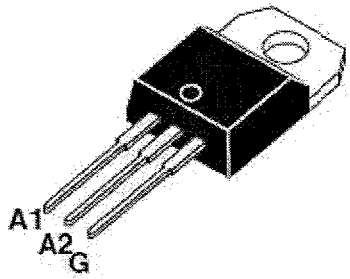
เป็นอุปกรณ์ Isolation ป้องกันแรงดันไฟกระแสสลับไหลเข้าวงจรในส่วนควบคุม

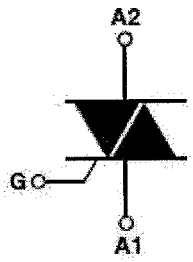


รูปที่ 2.14 แสดงลักษณะของ MOC3021

BTA26

เป็นอุปกรณ์ที่ใช้ควบคุมการเปิดการทำงานของวงจรกระแสสลับ ในการควบคุมเฟสการทำงานของขดลวดความร้อน ซึ่งใช้ต่อเป็นวงจรร่วมกับ MOC 3021 Zero crossing





TO-220AB Insulated
(BTA24)

รูปที่ 2.15 แสดงอุปกรณ์ BTA26

2.2.3 ระบบควบคุมแบบอัตโนมัติชนิด PID

ระบบควบคุมแบบอัตโนมัติเป็นระบบควบคุมแบบป้อนกลับที่ใช้กันอย่างแพร่หลาย ซึ่งเป็นระบบควบคุมขั้นสูงระบบนี้จะให้ความถูกต้องที่สูงกว่าในกระบวนการทำงาน บางระบบจะสามารถทำได้โดยเพียงใช้ Proportional (P) หรือ Proportional Integral (PI) ก็เพียงพอสำหรับระบบแล้วแต่เพื่อให้ได้ระบบที่สร้างขึ้นแล้วใกล้เคียงกับความเป็นจริงมากที่สุด โดยระบบส่วนมากจะต้องมีความสามารถในการคาดการสมรรถภาพของระบบ ซึ่งระบบดังกล่าวสามารถทำได้โดยเพิ่ม Derivative (D)เข้าไปในระบบ Derivative (D)ที่เพิ่มเข้าไปในระบบป้อนกลับจะป้อนกลับค่าเอาท์พุทที่กลับมายังที่Loop PI ซึ่งจะกลายมาเป็นระบบที่เรียกว่าProportional Integrate Derivative (PID) ระบบนี้จะให้ความแม่นยำและความถูกต้องสูงเหมาะกับระบบหรือกระบวนการที่มีความซับซ้อนสูงในยุคปัจจุบัน

แบบแผนของ PID controller เป็นสิ่งที่มีการรวมกันของค่าที่ปรับเปลี่ยนตามค่าที่แปลผันตรง (MV)

$$MV(t) = P_{out} + I_{out} + D_{out} \qquad (2.4)$$

โดยที่: P_{out}, I_{out}, และ D_{out} เป็นการสนับสนุนทาง Output จาก PID controller จากทั้ง 3 เทอมนี้ ตามคำนิยามด้านล่าง

Proportional term

ในเทอมของ Proportional ทำให้เกิดการเปลี่ยนแปลงด้านเอาต์พุตนั้นคือสัดส่วนค่าที่ผิดพลาดในปัจจุบัน สัดส่วนที่ตอบสนองสามารถปรับเปลี่ยนโดยผลคูณที่เกิดผิดพลาดโดยใช้ค่าคงที่ Kp ที่เรียกว่าอัตราขยายสัดส่วน

ในเทอมของ Proportional ได้สมการดังนี้

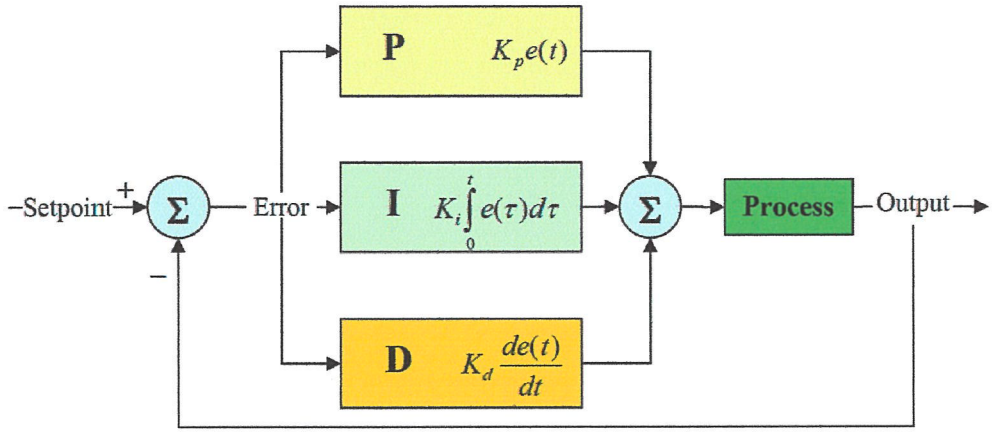
$$P_{out} = K_p e(t) \qquad (2.5)$$

โดยที่ : Pout : เทริมอัตราสัดส่วนด้านเอาต์พุต

Kp : อัตราขยายสัดส่วน การปรับค่าพารามิเตอร์

e: Error = SP – PV

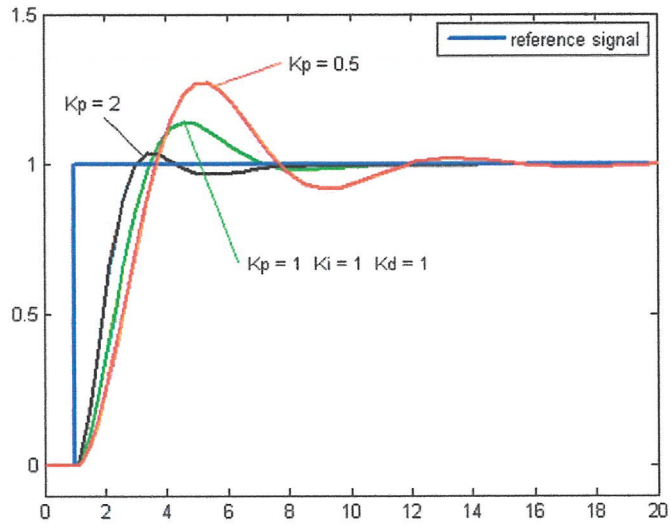
t: เวลา หรือ ค่าเวลาชั้วขณะ



รูปที่ 2.16 รูปบล็อกไดอะแกรมระบบสัญญาณที่เป็นผลการรวมกันของ PID ทางด้านเอาต์พุตของ ระบบ controller

Proportional (P)

ระบบรับสัญญาณเข้ามาและเมื่อผ่านระบบก็จะถูกป้อนกลับมายังตัวคอนโทรลเลอร์ ซึ่งจะทำให้เกิดค่าผิดพลาด (Error) คือผลต่างระหว่างเอาท์พุทกับจุด Set point จากการรับสัญญาณจากภายนอกเข้ามาในระบบทำให้เกิด Displacement error โดย Proportional filter จะทำหน้าที่ลดค่าผิดพลาดนี้อัตราเร็วในการเข้าสู่จุด Set point จะเป็นสัดส่วนเชิงเส้นกับค่าผิดพลาด การปรับค่าอัตราส่วนนี้ทำได้โดยปรับค่า Proportional gain คือค่า Kp นั่นเอง ซึ่งถ้าค่า Kp ต่ำอัตราเร็วในการเข้าสู่จุด Set pointจะต่ำ ทำให้ผลตอบสนองของระบบช้า แต่ถ้าค่า Kp สูงอัตราในการเข้าสู่จุด Set point สูง ผลตอบสนองของระบบทำให้เกิด Overshoot และ Undershoot ขึ้น



รูปที่ 2.17 แสดงการเข้าสู่จุด Set point ของ Proportional (P)

Integration (I)

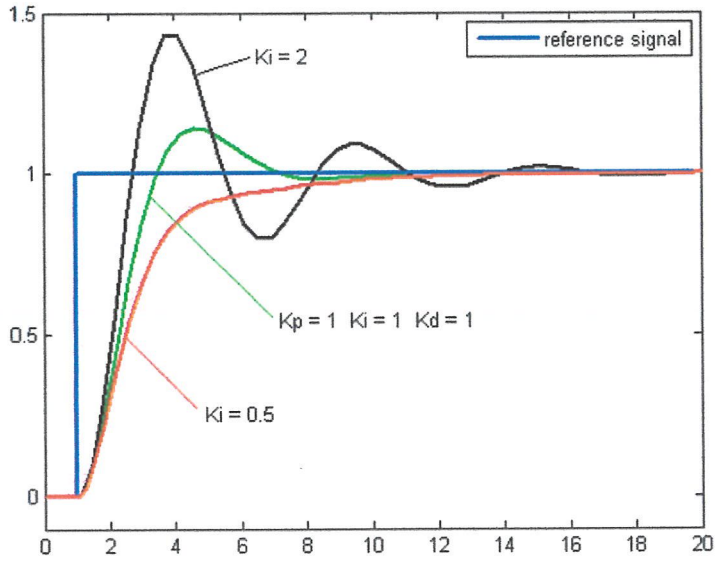
การกำจัด Tracking error ค่า Integral จะเป็นสัดส่วนโดยตรงกับค่า Tracking error และเพิ่มอย่างเป็นเชิงเส้นกับเวลา การปรับค่าสามารถปรับได้โดยการปรับ ค่า Integral gain คือค่า Ki ซึ่งถ้าค่า Ki มากจะเกิดการชดเชยอย่างรวดเร็ว แต่จะเพิ่มOvershoot และค่า Ringing ปกติค่า Ki ควรตั้งให้มีค่าสอดคล้องเพื่อทำให้คุณสมบัติของทั้ง 3 ระบบขอเชยซึ่งกันและกัน

จะได้ค่าอัตราขยาย Ki ในเทอมอินทิกรัล

$$= K_i \int_0^t e(T)dT \qquad (2.6)$$

โดยที่ : จะมีการ Integration (เมื่อการเพิ่มขึ้นของ Proportional) ทำให้เร่งกระบวนการเซ็ตค่าและกำจัดค่าerror ของสถานะคงที่ ที่เหลือนั้นปรากฏว่า Proportional เป็นเพียงตัวที่ควบคุม อย่างไรก็ตาม Integration ก็คือการตอบสนองที่มี error ที่เพิ่มขึ้นจากอดีต มันสามารถขึ้นอยู่กับค่าปัจจุบันที่ overshoot ค่าที่ เซ็ตได้

สำหรับผลที่บันทึกพิจารณาใน Integration ที่รวมกัน และควบคุมความเป็น เสถียรภาพจะพบในส่วนของลูปที่ทำการปรับให้เข้ากันแล้ว



รูปที่ 2.18 แสดงการเข้าสู่จุด Set point ของ Integration (I)

Derivative (D)

การกำจัด Oscillation ของระบบและลด Overshoot กับ Ringing สามารถปรับค่าได้โดยการปรับค่า Derivative gain คือค่า Kd ซึ่งค่า Kd จะเป็นสัดส่วนโดยตรงกับอัตราการเปลี่ยนแปลงของ Tracking error ค่า Kd จะทำให้เกิดเสถียรภาพของทั้งระบบและยังเพิ่มความสามารถในการคาดการผลตอบสนองของระบบ และยังสามารถทำให้เกิดความถูกต้องก่อนที่จะเกิด Tracking error ได้ ความสามารถในการคาดการนี้จะเกิดขึ้นได้จากผลตอบสนอง ซึ่งเกิดจากการกระทำของระบบในช่วงเวลานั้นๆ

Derivative(D) ได้สมการดังนี้

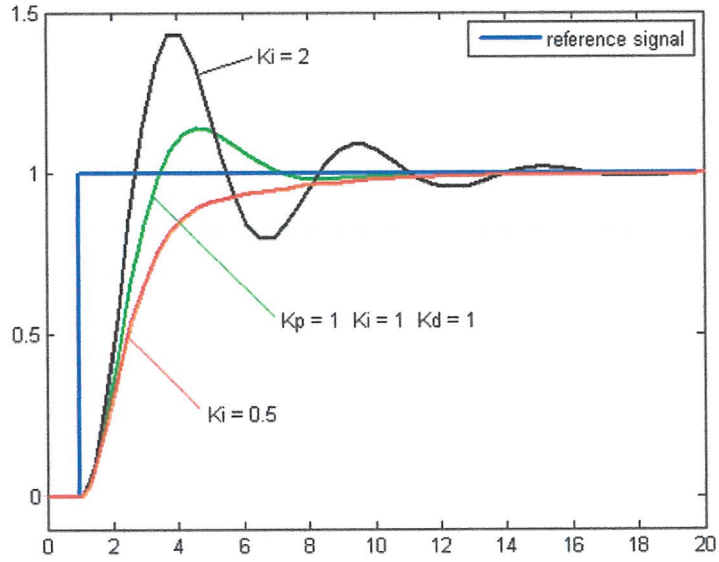
$$D_{out} = K_d \frac{de}{dt} \qquad (2.7)$$

โดยที่: Dout : เทริม Derivative ของเอาต์พุต

Kd : อัตราขยาย Derivative พารามิเตอร์ที่ปรับให้เข้ากัน

e : Error = SP – PV

t : เวลา หรือ เวลาชั่วขณะหนึ่ง (ในปัจจุบัน)



รูปที่ 2.19 แสดงการเข้าสู่จุด Set point ของ Derivative (D)

สรุป

เทริม Proportional, Integral และ Derivation เป็นผลการรวมกันของการบวกค่าการคำนวณเอาต์พุตของ PID controller

คำนิยาม u(t) เป็นเอาต์พุต controller สมการท้ายสุดคือ

$$u(t) = MV(t) = K_p e(t) + K_i \int_0^t e(T)dT + K_d \frac{de}{dt} \qquad (2.8)$$

และการปรับค่าพารามิเตอร์คือ

1. Kp : Proportional ค่า Kp ที่มีขนาดมากกว่าหมายความว่าจะมีการตอบสนองที่เร็วกว่าแต่ error ก็จะมากกว่าด้วย การชดเชยในเทริมสัดส่วนจะมีจำนวนมากกว่า อัตราขยายสัดส่วนที่มีจำนวนมากเกินไปจะนำไปใช้ในวิธีการไม่เสถียรภาพ และเกิดการออสซิลเลชั่น

2. Ki : Integral ค่าKi ที่มีขนาดมากกว่าแสดงนัยว่า error ที่แสดงสถานะคงที่จะถูกกำจัดได้เร็วกว่า การแลกเปลี่ยนคือ overshoot ที่มากกว่า ค่า error ทางด้านลบจะ Integral ระหว่างความตอบสนองแบบ transient ต้อง Integral ช่วง error ด้านบวกก่อนเราจึงจะได้ขอบเขต steady state

3. Kd : Derivation ค่าส่วนมาก Kd จะลดค่า Overshoot แต่ลดค่าตอบสนองเกี่ยวกับ transient ลง และอาจจะทำให้ไม่เสถียรภาพเพราะว่า สัญญาณรบกวนที่ขยายมีความแตกต่างของ error

Loop Tuning

ถ้าพารามิเตอร์ PID controller ถูกเลือกอย่างไม่ถูกต้องแล้ว สำหรับด้านอินพุตที่ควบคุมอยู่จะไม่เสถียรภาพ และเอาต์พุตของมันแตกต่างกัน หรือไม่ก็ไม่เกิด ออสซิลเลชั่น และถูกจำกัดเพียงแค่การอิ่มตัว หรือ การหยุดตัวลง การปรับลูปควบคุมคือปรับค่าพารามิเตอร์ให้ได้ค่าที่ได้ผลดีที่สุดสำหรับ ความต้องการ ของการตอบสนองในการควบคุม

จุดที่ทำให้ได้ผลที่ดีที่พบในวิธีนี้ คือการปรับเปลี่ยนเซ็ตค่า ซึ่งขึ้นอยู่กันการประยุกต์ใช้ สำหรับวิธีการนี้ต้องไม่มีการปล่อยให้เกิด overshoot ของการเปลี่ยนแปลงในทางปฏิบัติในวิธีนี้นอกจากการเซ็ตค่า สำหรับ ตัวอย่างเกิดความไม่ปลอดภัย ในวิธีการอื่นต้องใช้ค่าพลังงานที่นำในการปล่อยให้เซ็ตค่าใหม่โดยทั่วไปแล้วความเสถียรภาพของการตอบสนอง และวิธีการที่ใช้ต้องไม่เกิดการออสซิลเลเตอร์สำหรับการรวมกันของการปรับสภาพของวิธีการที่ใช้เซ็ตค่า ในหลายวิธีการที่ใช้มีความไม่เป็นเชิงเส้นและพารามิเตอร์ในการทำงานที่ดีที่การใช้โหลดแบบเต็มที่จะไม่สามารถปรับสภาพการทำงานได้เมื่อวิธีการทำงานเริ่มต้นจากไม่มีดหลด ในส่วนนี้จะบรรยายเกี่ยวกับสิ่งที่เป็นวิธีการที่สืบทอดจากการกระทำปรับให้อยู่ในลูป

วิธีที่หลากหลายในการปรับลูป PID ให้เข้ากัน วิธีการที่มีผลกระทบมากที่สุดเป็นการแก้ปัญหาโดยทั่วไปในการพัฒนาของต้นแบบวิธีการที่เป็นรูปแบบจากนั้นการเลือก P, I และ D เป็น

พื้นฐานทางด้านแบบทางรูปแบบที่เป็นค่าพารามิเตอร์ของ dynamic วิธีการปรับให้เข้ากับทางปฏิบัติสามารถใช้เทียบเคียงก็ยังไม่เพียงพอ

ทางเลือกของวิธีการจะขึ้นอยู่กับส่วนใหญ่หรือแม้ว่าจะไม่ทำการ offline สำหรับการปรับให้ตรงกันและเวลาในการตอบสนองของระบบ ถ้าระบบสามารถ offline ได้ วิธีการปรับให้ตรงกันบ่อยครั้งจะแก้ไขในตัวระบบเป็นการเปลี่ยนแปลงในด้านอินพุตการวัดเอาต์พุตในฟังก์ชั่นของเวลาในการใช้การตอบสนองได้ให้ค่าพารามิเตอร์ในการควบคุม

ตารางที่ 2.8 แสดงการเลือกวิธี Tuning

การเลือกวิธี **Tuning**		
วิธีการ	ข้อดี	ข้อเสีย
Manual Tuning	ไม่ต้องมีใช้การคำนวณ ใช้วิธี Online	ต้องการผู้ที่มีประสบการณ์
Ziegler-Nichols	ใช้วิธีการพิสูจน์ ใช้วิธี Online	จะเกิดปัญหาที่กระบวนการ และบางครั้งต้องทดลองซ้ำแล้วซ้ำอีก เกิดการรบกวนเมื่อ tuning
Software Tools	มีการ tuning ที่ไม่เปลี่ยนแปลง วิธีการ Online หรือ Offline	เกี่ยวกับราคา และ การฝึกฝน
Cohen-Coon	มีรูปแบบกระบวนการที่ดี	เกี่ยวกับการคำนวณ วิธี Offline ใช้ได้ผลดีกับ first-order เท่านั้น

Manual Tuning

ระบบต้องยังคง online วิธีการ **Tuning** อย่างหนึ่งลำดับแรกตั้งค่า I และ D เป็น 0 เพิ่ม P จนกระทั่ง output ของลูปเกิดออสซิลเล็ต จากนั้น P ควรจะทำการประมาณค่าในการเซ็ตเพียงครึ่งเดียวสำหรับ " การลดขนาดลงมา 1 ใน 4 " จากนั้นเพิ่ม I จนกระทั่งเกิด offset ที่ถูกต้องมีเวลาเพียงพอสำหรับกระบวนการที่ใช้ อย่างไรก็ตามถ้าปริมาณมากเกินไป I จะมีผลให้เกิดความไม่เถียรภาพ เพิ่ม D ขึ้น ถ้าต้องการจนกระทั่งลูปได้ยอมรับจะเกิดการตอบสนองมากเกินไป และ เกิด overshoot การ tuning ลูป PID เร็วเกินไปมักจะเกิด Overshoot อย่างช้าๆแล้วขึ้นไปที่จุด set point ที่เร็วมากขึ้น อย่างไรก็ตามบางระบบไม่ยอมรับ การ Overshoot ในกรณีนี้ในระบบปิดต้องการ Overdamp เป็นสิ่งซึ่งต้องการ การเซ็ต P เป็นสิ่งจำเป็นต้องเซ็ตให้น้อยกว่าครึ่งหนึ่งเพราะว่าการเกิดออสซิลเล็ต

ตารางที่ 2.9 ผลกระทบของค่าพารามิเตอร์ที่เพิ่มขึ้น

ผลกระทบของค่าพารามิเตอร์ที่เพิ่มขึ้น				
พารามิเตอร์	Rise Time	Overshoot	Settling Time	S.S. Error
Kp	ลดลง	เพิ่มขึ้น	เปลี่ยนแปลงเล็กน้อย	ลดลง
Ki	ลดลง	เพิ่มขึ้น	เพิ่มขึ้น	ถูกกำจัด
Kd	ลดลงเล็กน้อย	ลดลง	ลดลง	ไม่เปลี่ยนแปลง

Ziegler-Nichols Method:

1.ลำดับแรกไม่ว่าจะเป็นอัตราขยายควบคุมสัดส่วนที่ต้องการไม่ว่าเป็นด้านบวกหรือด้านลบสามารถทำได้ ก้าวแรกโดยการเพิ่ม อินพุตเล็กน้อย ภายใต้การใช้คู่มือการคอนโทรลพบว่าถ้าผลที่ได้มีค่าเป็น steady state ของกระบวนการทางเอาต์พุตที่สามารถเพิ่มขึ้นได้ จากนั้นอัตราการขยายกระบวนการของ steady state ที่ด้านบวกและอัตราขยายควบคุมสัดส่วนที่ต้องการ Kc จะมีการตอบสนองด้านบวกได้ดี

2.สามารถปิดคอนโทรลเลอร์ใน mode P ท่านั้น นั้นคือการปิดทั้งอินทิกรัลและดิริฟีทีฟ mode

3.อัตราขยายคอนโทรลเลอร์ เพิ่มขึ้นอย่างช้าๆ (เพิ่มด้านบวกให้มากขึ้นถ้า Kc ถูกตัดสินให้นำไปใช้ในก้าวแรก ไม่เช่นนั้นค่าด้านลบที่มากขึ้นถ้า Kc ถูกพบต้องการด้านลบในก้าวแรก) และสังเกตการณ์ตอบสนองด้านเอาต์พุตสิ่งนั้นต้องการเปลี่ยน Kc ในก้าวแรกที่เพิ่มขึ้นและการรอสำหรับ steady state ในทางด้านเอาต์พุตก่อนที่มีการเปลี่ยนแปลง Kc ที่ถูกทำให้ได้ผลดังกล่าว

4. เมื่อค่าของ Kc ผลที่ได้รักษาคาบเวลาการออสซิลเลชั่นในทางเอาต์พุต (หรือเข้าใกล้ค่าที่ต้องการมากที่สุด) กำหนดค่าวิกฤตของ Kc Ku ประมาณอัตราขยายการวัดคาบเวลา

ของออสซิลเลชั่น Pu อ้างอิงประมาณคาบเวลา

5.การใช้ค่าของอัตราขยายพื้นฐาน Ku และคาบเวลาพื้นฐาน Pu คำแนะนำของ Ziegler และ Nichols ทำตามค่า Ke, tr และ tp ขึ้นอยู่กับชนิดของ controller ที่ต้องการ

ตารางที่ 2.10 Ziegler-Nichols Tuning Chart

	K_c	□$_I$	□$_D$
P control	$K_u/2$		
PI control	$K_u/2.2$	$P_u/1.2$	
PID control	$K_u/1.7$	$P_u/2$	$P_u/8$

ตารางตัวเลือกจากตารางด้านบน ค่าที่ปรับแต่งต่างๆ นิยามโดย Tyreus และ Luyblen สำหรับ PI และ PID บ่อยครั้งถูกเรียกว่า TLC กฎการปรับแต่ให้เข้ากันค่าเหล่านี้มีแนวโน้มที่ลดผลกระทบของออสซิลเลเตอร์ และปรับปรุงให้มั่นคง

Cohen-Coon Tuning Method

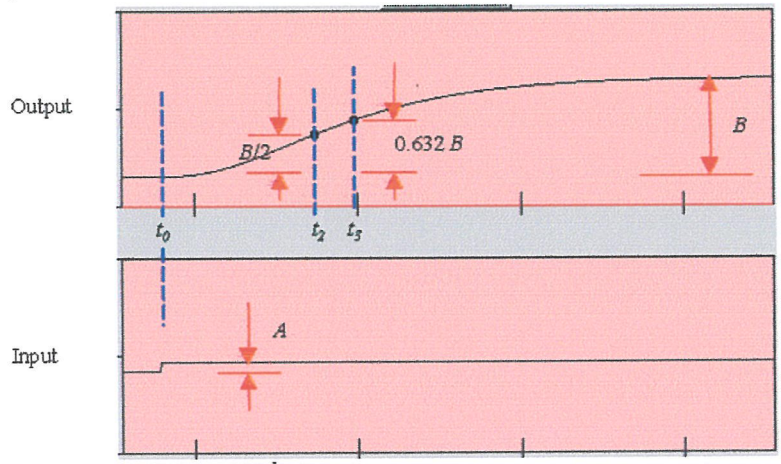
1. รอจนกระทั้งกระบวนการเกิด Steady state.
2. จากนั้นเริ่มเปลี่ยน input
3. อ้างอิงทาง output ทำการประมาณกระบวนการ first order ด้วยค่าเวลาคงที่ τ ถูกหน่วงเวลาโดย τ_{DEL} เมื่อทาง input ถูกอ้างอิง

ค่าของ τ และ τ_{DEL} สามารถใช้ค่าเวลาคงที่ตามนี้

t_0 = เวลาเริ่มต้น

t_2 = เวลาเมื่อที่ปรากฏครึ่งหนึ่ง

t_3 = เวลาที่ปรากฏที่ 63.2%



รูปที่ 2.20 แสดง Input และ Output ของ Cohen-Coon Tuning Method

4. จากการวัดจะได้พื้นฐานที่ทดสอบจะได้ t_0, t_2, t_3, A และ B ค่าที่ได้ตามมาของพารามิเตอร์ตามกระบวนการดังกล่าว

$$t_1 = (t_2 - In(2)t_3)/(1 - Ln(2)) \qquad (2.9)$$

โดยที่ : $\tau = t_3 - t_1$

$\tau_{DEL} = t_1 - t_0$

$K = B / A$

PID tuning software

Software เหล่านี้จะรวบรวมข้อมูล รูปแบบการพัฒนากระบวนการ อ้างอิงการเปลี่ยนแปลงทางด้านคณิตศาสตร์ได้นำ PID loop ใช้ด้านระบบ impulse และใช้ควบคุมระบบตอบสนองด้านความถี่ ออกแบบค่าของ PID loop ในลูปมีการตอบสนองเวลาที่แปลเปลี่ยนใช้ทางคณิตศาสตร์เข้ามาช่วยในการรับรองเพราะว่าการทำซ้ำหลายครั้งหลายหนในการจดบันทึกหลายวันเพียงเพื่อหาค่าเสถียรภาพ เป็นสิ่งที่ยากมากลูป digital อ้างถึงการปรับแต่งค่าด้วยตัวเองเป็นการเซ็ตค่าที่เปลี่ยนแปลงเล็กน้อยที่ส่งไปให้กระบวนการนั้น การอนุญาตให้คำนวณค่าด้วยตัวมันเองในการปรับค่า สูตรปรับเปลี่ยนไปตามความแต่งต่างของที่ทดลองเป็นบรรทัดฐาน

รูปแบบที่มาตรฐานของPID ที่เปรียบเทียบกันในอุดมคติ

$$MV(t) = K_p\left(e(t) + \frac{1}{T_i}\int_0^t e(T)dT + Td\frac{de}{dt}\right)\Bigg] \qquad (2.10)$$

โดยที่: T_i คือค่า **Integral Time**

T_d คือค่า **Derivative Time**

อีกด้านหนึ่งจะได้สมการ

$$MV(t) = K_p e(t) + K_i\int_0^t e(T)dT + K_d\frac{de}{dt} \qquad (2.11)$$

โดยที่: อัตราขยายพารามิเตอร์ที่มีความสัมพันกันในรูปมาตรฐานจาก

$$K_i = \frac{K_p}{T_i} \qquad K_d = K_p T_d$$

รูปแบบ Laplace ของ PID controller

รูปแบบของสมการที่ได้

$$G(s) = K_p + \frac{K_i}{s} + K_d s = \frac{K_d s^2 + K_p s + Ki}{s} \qquad (2.12)$$

ถ้ามีการเขียน PID controller ในรูป Laplace ทำให้ส่งผ่านทาง function ของระบบคอนโทรล ทำให้ง่ายในการลูปปิด function ของระบบ

รูปแบบที่มีผลต่อกันได้สมการดังนี้

$$G(s) = K_c \frac{(T_i s + 1)}{T_i s}(T_d s + 1) \tag{2.13}$$

โดยที่: มีค่าพารามิเตอร์ที่มีความสัมพันกันที่มีรูปมาตรฐานคือ

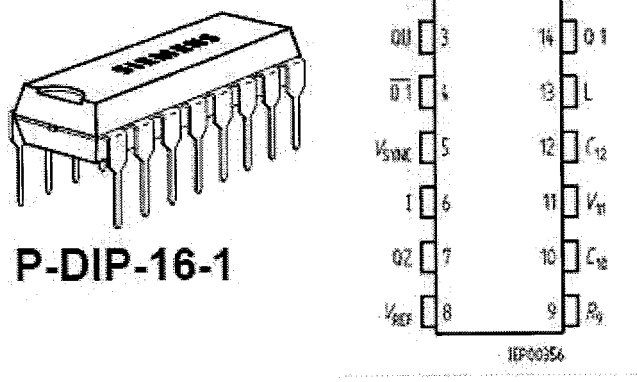
$$K_p = K_c \cdot \alpha \tag{2.14}$$

$$T_d = \frac{T_d}{\alpha} = 1 + \frac{T_d}{T_i} \cdot \alpha$$

สิ่งนี้เป็นพื้นฐานประกอบกับ PD และ PI ในรูปแบบอนุกรม และเป็นสิ่งที่ง่ายกว่าในการควบคุม แบบ analog เมื่อมีการเปลี่ยนเป็น digital จะมีรูปแบบที่ส่งผ่านระหว่างกัน

2.2.4 เฟสคอนโทรล

Phase Control IC



รูปที่ 2.21 แสดงตัวถังและขาต่างๆของ IC Phase Control

ลักษณะที่สำคัญ

-มีความจำในตัวที่เชื่อถือได้

-การประยุกต์ใช้ได้หลากหลาย

-การถูกใช้เป็นจุดเปลี่ยน Zero

-การเข้ากันได้กับ LSL

-การใช้งานกับ 3 เฟส (ICs)

-กระแส Output 250mA

-ความกว้างของย่านกระแส Ramp

-ย่านความกว้างของอุณหภูมิ

ตารางที่2.11 แสดงคำนิยามและหน้าการทำงานของขาต่างๆ

Pin	Symbol	Function
1	GND	Ground
2 3 4	$\overline{Q2}$ Q U $\overline{Q2}$	Output 2 inverted Output U Output 1 inverted
5	V_{SYNC}	Synchronous voltage
6 7	I Q Z	Inhibit Output Z
8	V_{REF}	Stabilized voltage
9 10	R_9 C_{10}	Ramp resistance Ramp capacitance
11	V_{11}	Control voltage
12	C_{12}	Pulse extension
13	L	Long pulse
14 15	Q 1 Q 2	Output 1 Output 2
16	V_S	Supply voltage

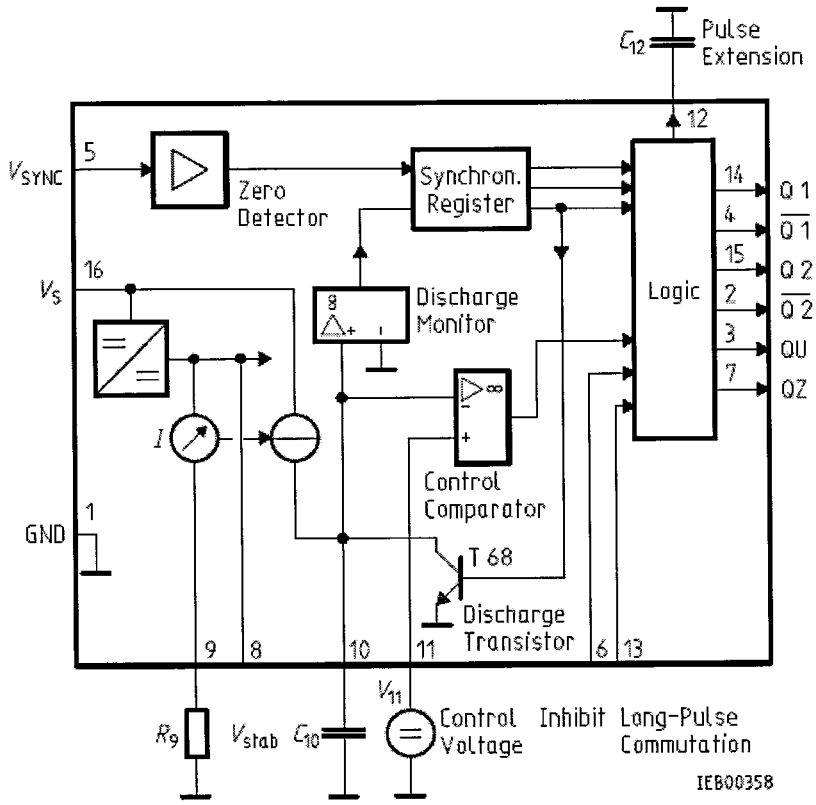
วงจรจุดชนวนเกตด้วยวงจรรวมสำหรับการควบคุมเฟสเบอร์ TA785

1. คุณสมบัติของวงจรรวม TCA785 วงจรรวมเบอร์ TCA785 ใช้ในงานควบคุมเฟสโดยเฉพาะหรือใช้ได้กับวงจรLine commutated converter ใช้ได้กับระบบไฟฟ้า 1 เฟส 3 เฟส สามารถปรับมุมจุดชนวนปรับมุมจุดชนวนเกตได้ตั้งแต่ 0 ° ถึง 180 ° และสามารถสร้างลักษณะของพัลส์ได้หลายรูปแบบ ตามลักษณะของโหลด ใช้งานได้ดีในย่านความถี่10 Hz ถึง 500 Hz ต้องการไฟเลี้ยงกระแสตรงขนาด 8 Vdc ถึง 18 Vdc ใช้งานได้ดีในย่านอุณหภูมิ -25 °C ถึง +85 °C กินกระแสไฟฟ้าประมาณ 4.5 mA – 10 mA อินเตอร์เฟสได้โดยตรงกับลอจิกเกตที่ใช้แรงดัน + 15 Vdc จ่ายกระแสไฟฟ้าด้านออกของวงจรที่ขา 14 และ 15 ได้ประมาณ 250 mA สัญญาณด้านออกมีทั้งแบบปกติและแบบกลับสัญญาณ มี Inhibit function ควบคุมการกำเนิดสัญญาณ และสามารถใช้งานในลักษณะ Zero crossing ได้ด้วย

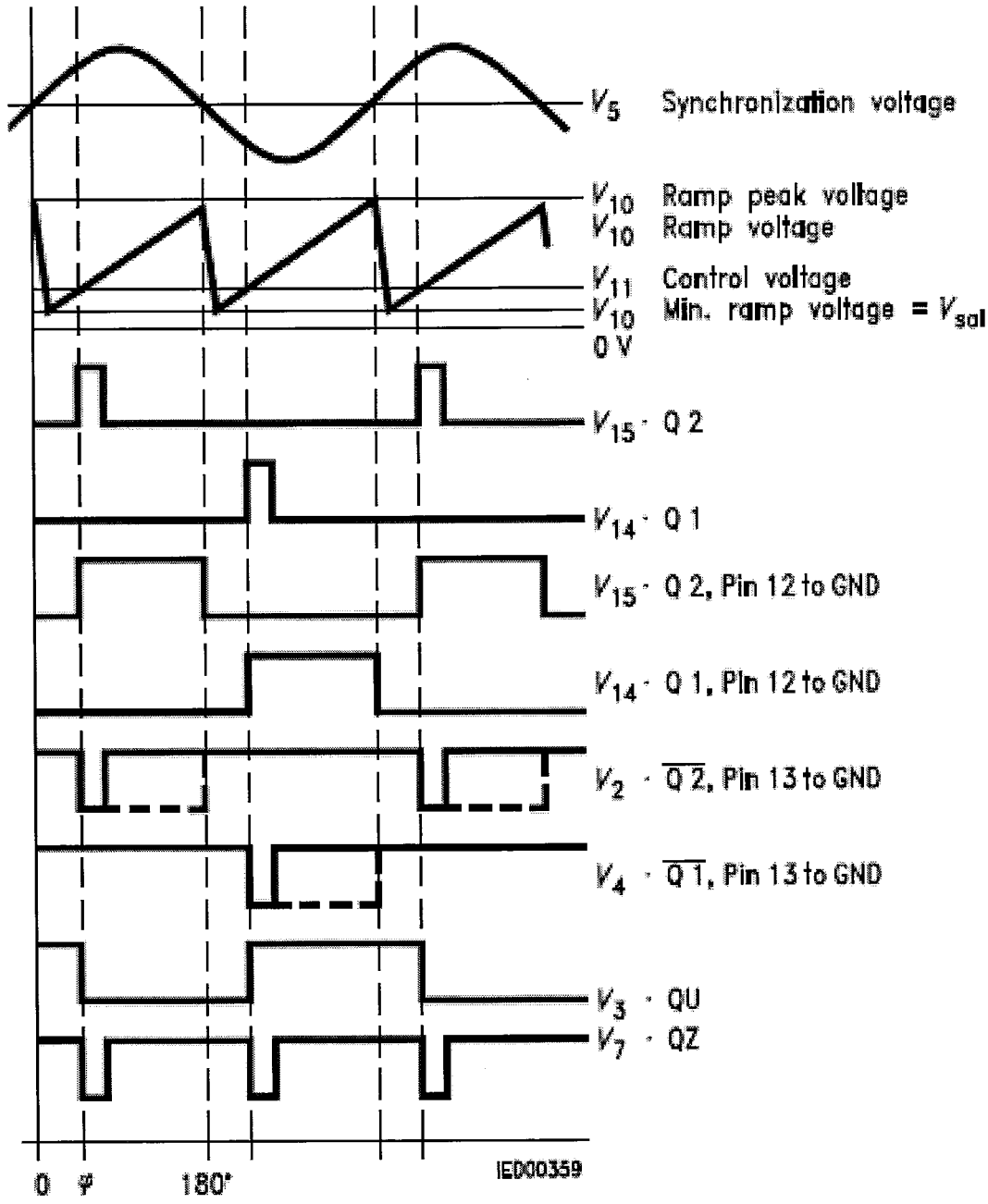
2.การทำงานของวงจรรวม TCA785 พิจารณาแผนภาพกรอบแสดงโครงสร้างภายในของ TCA785 ซึ่งเป็นวงจรรวมแบบ LSI มี 16 ขา บรรจุแบบตัวถังพลาสติกขา 16 คือ + Vs รับแรงดันไบอัสในย่าน +8 Vdc ถึง +18 Vdc โดยขา1เป็นกราวด์ แรงดันควบคุมภายในเป็นแรงดันอ้างอิง คื

Vreff $\approx$ 3.1 V วัดได้ที่ขา 8 โดยตัว C8 ทำหน้าที่ป้องกันสัญญาณรบกวน ขา 5 คือขาที่รับแรงดันซิงโครไนซ์ (Vsyn) ซึ่งเป็นแรงดันไฟสลับจากแหล่งจ่ายไฟสลับที่ต่อกับวงจรภาคกำลัง โดยต่อนิวตรอนกับขา 1, R9 คือตัวต้นทานสร้างสัญญาณลาดเอียง และ C10 คือตัวเก็บประจุสัญญาณลาดเอียงต่อกับขา9 และขา10 ตามลำดับ ค่าของ C10 มีค่าในย่าน 500 pF ถึง 1uF และ R9 มีค่าเหมาะสมในย่าน 3 kΩ-300kΩ

ค่า R9 และ C10 จะเป็นตัวกำหนดขนาดของสัญญาณลาดเอียง (V10) ถ้า R9 และ C10 มีค่ามาก ความลาดเอียงของ V10 จะมีค่ามากตามไปด้วย ขา 11 ของ TCA785 คือขาที่ต่อแรงดันควบคุมเป็นแรงดันไฟตรงปรับแรงดันไฟตรงปรับค่าได้แรงดันควบคุม (V11) นี้จะป้อนเข้าขาบวก (+) ของออปแอมป์เปรียบเทียบสัญญาณควบคุมในรูป โดยเทียบกับแรงดัน V10 เพื่อกำหนดขนาดของมุมจุดชนวนที่ด้านออกของวงจร ดังแสดง จะเห็นว่าสัญญาณด้านออกของวงจรอยู่ที่ขา 14 และ 15 โดยพัลส์จะทำงานช่วงเวลา $\omega t = 0^{o}$ -180^{o} และ พัลส์ที่ขา 14 จะทำงานที่ = 180° - 360° ขนาดความกว้างของพัลส์ด้านออกคือ β ปกติถ้าไม่ต่อ C12 จะมีค่า β = 30µS แต่สามารถเปลี่ยนค่า β = 30µS แต่สามารถเปลี่ยนค่า β ได้โดยใช้ค่า C12 ต่อเข้าที่ขา12 ดังแสดงความสัมพันธ์ของค่า C12 กับค่า β



รูปที่ 2.22 แสดง Block Diagram วงจรจุดชนวนเกตด้วยวงจรรวมสำหรับการควบคุมเฟส



รูปที่ 2.23 แสดงสัญญาณที่ได้จากวงจรจุดชนวนด้วยวงจรรวมสำหรับการควบคุมเฟส

วงจรจุดชนวนด้วยวงจรรวมสำหรับการควบคุมเฟส TCA785

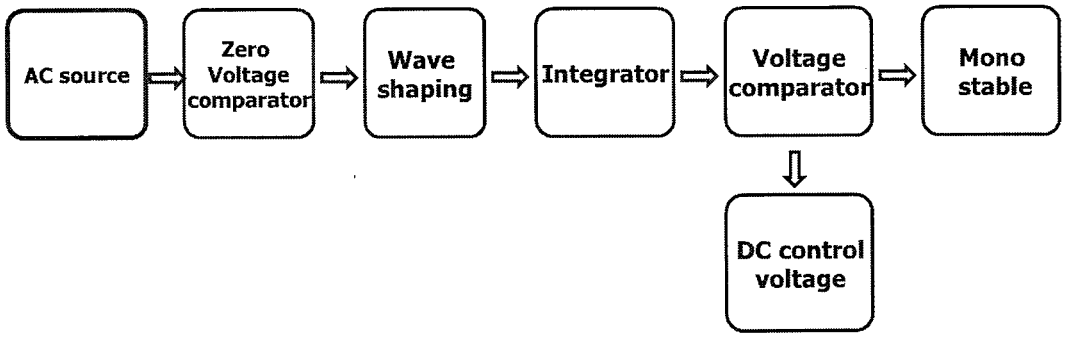
1.คุณสมบัติของวงจรรวม TCA785 ใช้ได้กับระบบไฟฟ้า 1 เฟส และ 3 เฟส สามารถปรับควบคุมมุมจุดชนวนเกตได้ตั้งแต่ 0- 180 องศา และสามารถสร้างลักษณะของพัลส์ได้หลายรูปแบบตามลักษณะ ของโหลด ใช้งานได้ดีในย่านความถี่ 10Hz ถึง 500Hz ต้องการไฟเลี้ยงกระแสตรงขนาด 8 Vdc ถึง 18 Vdc ใช้งานได้ดีในย่านอุณหภูมิ -25 องศาC ถึง +85องศาC กินกระแสไฟฟ้าประมาณ 4.5 mA-10mA อินเตอร์เฟสได้โดยตรงกับลอจิกเกตที่ใช้แรงดัน + 15Vdc (เช่น CMOS) สามารถจ่ายกระแสไฟฟ้าด้านออกของวงจรที่ขา 14 และ 15 ได้ประมาณ250mA สัญญาณด้านออกมีทั้งแบบปกติและแบบกลับเฟสสัญญาณ มี Inhibit function ควบคุมการกำเนิดสัญญาณ และสามารถใช้งานในลักษณะ Zero crossing ได้ด้วย

2.การทำงานของวงจร TCA785 พิจารณาแผนภาพกรอบแสดงโครงสร้างภายในของ TCA85 ซึ่งเป็นวงจรรวมแบบ LSI มี 16 ขา บรรจุแบบตัวถังพลาสติก 16 คือ + Vs รับแรงดันไบแอสในย่าน +8Vdc ถึง +18Vdc โดยขาที่1เป็นจุดดิน แรงดันควบคุมภายในเป็นแรงดันอ้างอิงคือ Vreff≈ 3.1 V วัดได้ที่ได้ขา 8 โดยตัว C8 ทำหน้าที่ป้องกันสัญญาณรบกวน ขา 5 คือขาที่รับแรงดันซิงโครไนซ์(Vsyn) ซึ่งเป็นแรงดันไฟสลับจากแหล่งจ่ายไฟสลับที่ต่อกับวงจรภาคกำลัง โดยต่อนิวตรอนกับขา 1, R9 คือตัวต้านทานสร้างสัญญาณลาดเอียง และ C10 คือตัวเก็บประจุสัญญาณลาดเอียงต่อขา9 และขา10 ตามลำดับค่าของ C10 มีค่าในย่าน 500 pF ถึง uF และ R9 มีค่าเหมาะสมในย่าน 3kΩ-300kΩ

ค่า Rġ และ C10 จะเป็นตัวกำหนดขนาดของสัญญาณลาดเอง (V10) ถ้า R9 และ R10 มีค่ามากความลาดเอียงของ V10 จะมีค่ามากตามไปด้วย ขา 11 ของ TCA785 คือขาที่ต่อแรงดันควบคุมเป็นแรงดันไฟตรงปรับค่าได้ แรงดันควบคุม (V11) นี้จะป้อนเข้าขาบวก (+) ของออปแอมป์เปรียบเทียบสัญญาณควบคุมในรูป8.22 โดยเทียบกับแรงดัน V10 เพื่อกำหนดขนาดของมุมจุดชนวนที่ด้านออกของวงจร

วงจรจุดชนวนเกตไทริสเตอร์ที่ใช้ออฟแอมป์

เนื่องจากออปแอมป์เป็นวงจรรวมที่มีประสิทธิภาพสูง ทำงานได้หลายหน้าที่ วงจรจุดชนวนเกตของไทริสเตอร์ มีหลักการดังแสดงในภาพกรอบรูป



รูปที่ 2.24 แบบภาพกรอบแสดงวงจรจุดชนวนเกตของไทริสเตอร์ที่ใช้ในวงจรรวมออปแอมป์

1.**Ac source** คือแหล่งจ่ายไฟสลับชุดเดียวกับแหล่งจ่ายไฟสลับที่จ่ายกำลังไฟฟ้าให้กับวงจรกำลังที่ไทริสเตอร์นั้นต่อทำงานอยู่ แต่แหล่งจ่ายไฟสลับที่จะจ่ายให้กับวงจรจุดชนวนเกตควรเป็นแรงดันไฟสลับแรงดันต่ำ เช่น 24Vac, 18Vac หรือ 12Vac โดยใช้หม้อแปลงแรงดันไฟฟ้าเป็นตัวลดระดับแรงดัน หรือใช้ตัวต้านทานที่ทนสูงๆ เป็นตัวแบ่งแรงดัน

2. **Zero voltage comparator** คือวงจรเปรียบเทียบแรงดันศูนย์ หรือวงจร Zero crossing detector ทำหน้าที่รับแรงดันไฟสลับ 12Vac เข้ามาทางขาบวกของIC1 เปรียบเทียบกับแรงดันศูนย์โวลต์ที่เข้ามาทางขาลบของ IC1 สำหรับ R2 ทำหน้าที่ลดแรงดันออฟเซตที่ด้านเข้าของ IC1 ลักษณะของวงจรในส่วนที่1 และ2 แสดงในรูป คลื่นแรงดันขาออกจาก IC1 จะเป็นรูปสี่เหลี่ยมที่มีระดับแรงดันสูงสุดด้านบวกเท่ากับ 0.8(Vcc) และแรงดันสูงสุดด้านลบเท่ากับ 0.8 (-Vcc) ค่าแรงดัน Vcc และ –Vcc ขึ้นอยู่กับแหล่งจ่ายไฟฟ้ากระแสตรงที่ไบอัสออปแอมป์

3.**Wave shaping** คือ วงจร ตัดแต่งรูปคลื่นในวงจรนี้คือ วงจรขริบ (clipper) โดยการใช้ไดโอด สวิตชิง (d1) และตัวต้านทาน R4 (ตัวต้านทาน R3 คือ RL ของออปแอมป์ (IC1)) จะตัดรูปคลื่นแรงดันที่จุดที่2 ให้มีเฉพาะด้านบวกเท่านั้น

4.**Integrator** คือวงจรออปแอมป์ ทำหน้าที่อินทิเกรตสัญญาณด้านเข้า เมื่อสัญญาณด้านเข้าเป็นคลื่นสี่เหลี่ยม สัญญาณด้านออกของวงจรอินทิเกรเตอร์ (IC2) จะเป็นสัญญาณลาดเอียวคล้ายคลื่นสามเหลี่ยมหรือสัญญาณลาดเอียง

5.**DC control voltage** คือระดับแรงดันไฟตรงที่ใช้ในการควบคุมมุมจุดชนวนเกตของสัญญาณพัลส์ด้านนอก โดยใช้ตัวต้านทาน R7 ต่อกับแหล่งจ่ายไฟตรง เพื่อทำหน้าที่เป็นวงจรแบ่งแรงดันที่ปรับค่าได้โดยการปรับค่าของ R7 ลักษณะของรูปคลื่น

6. **Voltage comparator** คือ วงจรออปแอมป์ที่ใช้เปรียบเทียบสัญญาณด้านเข้าสัญญาณหนึ่ง (Input signal) กับแรงดันอ้างอิง (reference voltage) สัญญาณด้านเข้าคือสัญญาณลาดเอียง และสัญญาณอ้างอิงคือแรงดันไฟตรงที่ปรับค่าได้ในจุด โดยสัญญาณที่จุดที่ ป้อนเข้าขาบวกของออปแอมป์ (IC3) และแรงดันอ้างอิงจุดที่ ป้อนเข้าขาลบของ (IC3) เมื่อปรับค่า R7 แรงดันอ้างอิงที่จุดที่ จะเปลี่ยนแปลงไป ทำให้ความกว้างของพัลส์ด้านออก IC3 เปลี่ยนแปลงไปด้วย

7. **Mono stable** คือวงจรกำเนิดสัญญาณพัลส์สี่เหลี่ยมที่ควบคุมความกว้างของพัลส์ได้ด้วยขนาดของ R8 และ C2 ดังนั้นสัญญาณด้านขาออกของออปแอมป์ (IC4) จะเป็นพัลส์สี่เหลี่ยมขนาดเล็กที่มีความกว้างของพัลส์คงที่โดยปกติควรให้ความกว้างของพัลส์นี้มีขนาดเล็กที่มีความกว้างของพัลส์คงที่ โดยปกติควรให้ความกว้างของพัลส์นี้มีขนาดโตกว่าขนาด t(on) ของไทริสเตอร์ที่ต้องการจุดชนวน

8.**Driver & Isolator** คือวงจรภาคขับและทำหน้าที่เป็นฉนวน กั้นระหว่างวงจรจุดชนวนกับวงจรจุดชนวนกับวงจรกำลังที่ไทริสเตอร์ต่ออยู่ โดยใช้ทรานซิสเตอร์ชนิด NPN(Q1) เป็นตัวสวิตซิ่ง โดย R9 เป็นตัวต้านทานเบส คำนาณค่า R9 จากขนาดของแรงดันที่จุด และ T1 คือ พัลส์หม้อแปลงที่ทำหน้าที่แยกสัญญาณที่เบสของทรานซิสเตอร์ (Q1) กับสัญญาณ Vgk ที่ด้านออกของวงจร เพื่อไม่ให้มีการเชื่อมโยงทางไฟฟ้าถึงกันระหว่างภาคควบคุมกับภาคกำลัง เป็นการป้องกันวงจรควบคุมไม่ให้เกิดอันตราย ขณะที่วงจรกำลังเกิดลัดวงจรขึ้น ขนาดอัตราส่วนของหม้อแปลงพัลส์ขึ้นอยู่กับขนาดแรงดัน Vgk

การแนะสำหรับการประยุกต์สำหรับอุปกรณ์ภายนอก

		min	max	
Ramp capacitance	C_{10}	500 pF	1 µF[1)]	The minimum and maximum values of I_{10} are to be observed
Triggering point	$t_{Tr} = \frac{V_{11} \times R_9 \times C_{10}}{V_{REF} \times K}$		[2)]	
Charge current	$I_{10} = \frac{V_{REF} \times K}{R_9}$		[2)]	Ramp voltage $V_{10\,max} = V_S - 2\,V$ $V_{10} = \frac{V_{REF} \times K \times t}{R_9 \times C_{10}}$ [2)]

ตารางที่ 2.12 ตารางการแสดงคุณสมบัติ ค่าพารามิเตอร์วงจรจุดชนวนเกตไทรริสเตอร์

Characteristics
$8 \le V_S \le 18$ V; -25 °C $\le T_A \le 85$ °C; $f = 50$ Hz

Parameter	Symbol	Limit Values			Unit	Test Circuit
		min.	typ.	max.		
Supply current consumption S1 ... S6 open $V_{11} = 0$ V $C_{10} = 47$ nF; $R_9 = 100$ kΩ	I_S	4.5	6.5	10	mA	1
Synchronization pin 5 Input current R_2 varied Offset voltage	$I_{S\,rms}$ ΔV_5	30	 30	200 75	µA mV	1 4
Control input pin 11 Control voltage range Input resistance	V_{11} R_{11}	0.2	 15	$V_{10\,peak}$	V kΩ	1 5

3.ลักษณะของพัลส์ด้านออกชนิดต่าง การสร้างพัลส์ด้านออกให้มีรูปร่างแตกต่างกันนั้น ขึ้นอยู่กับชนิดของโหลดบนวงจรกำลังที่ไทริสเตอร์นั้นต่อควบคุมโหลดอยู่ ลักษณะของพัลส์มีหลายแบบดังรูป

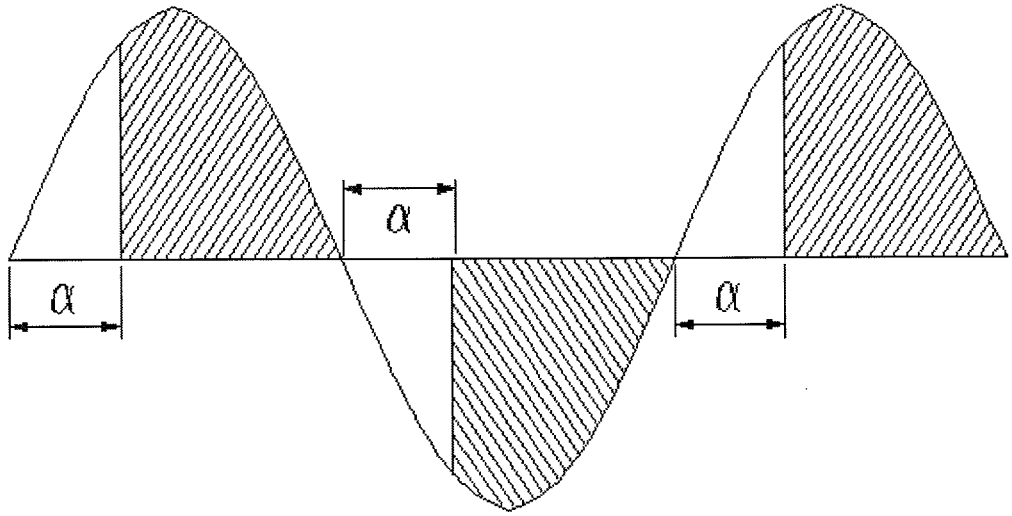
- Short pulse ($\beta = 30 - 100\mu S$) สำหรับโหลดตัวต้านทาน

-Long pulse($\beta = 100\mu S$-1mS) สำหรับโหลดตัวต้านทานที่กินกระแสสูงๆหรือโหลดตัวนำ

-Continuous pulse($\beta = 180^{\circ}-\alpha$) สำหรับโหลดตัวเหนี่ยวนำที่มีค่า L สูงมากๆ

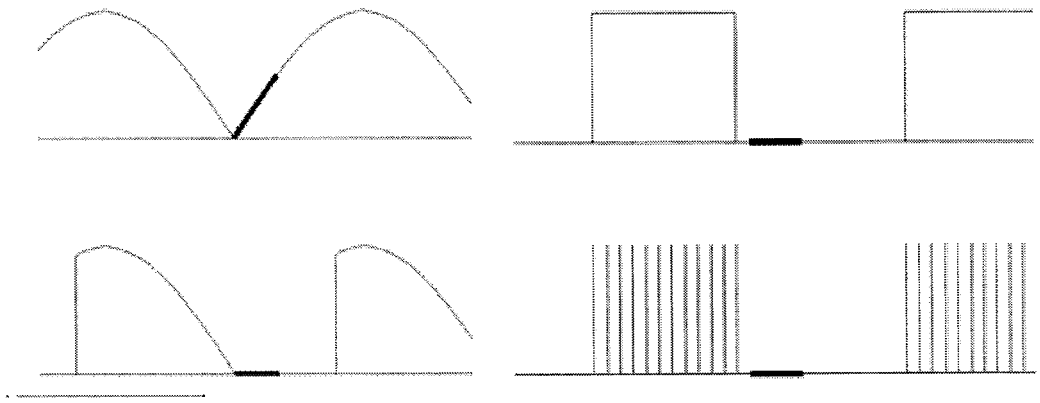
-Combined pulse สำหรับวงจรที่มีการเปลี่ยนแปลงของกระเกต (di/dt = 1 ถึง 3 A/µS) หรือสำหรับจุดชนวนเกตไทริสเตอร์ที่ต่ออนุกรมกัน

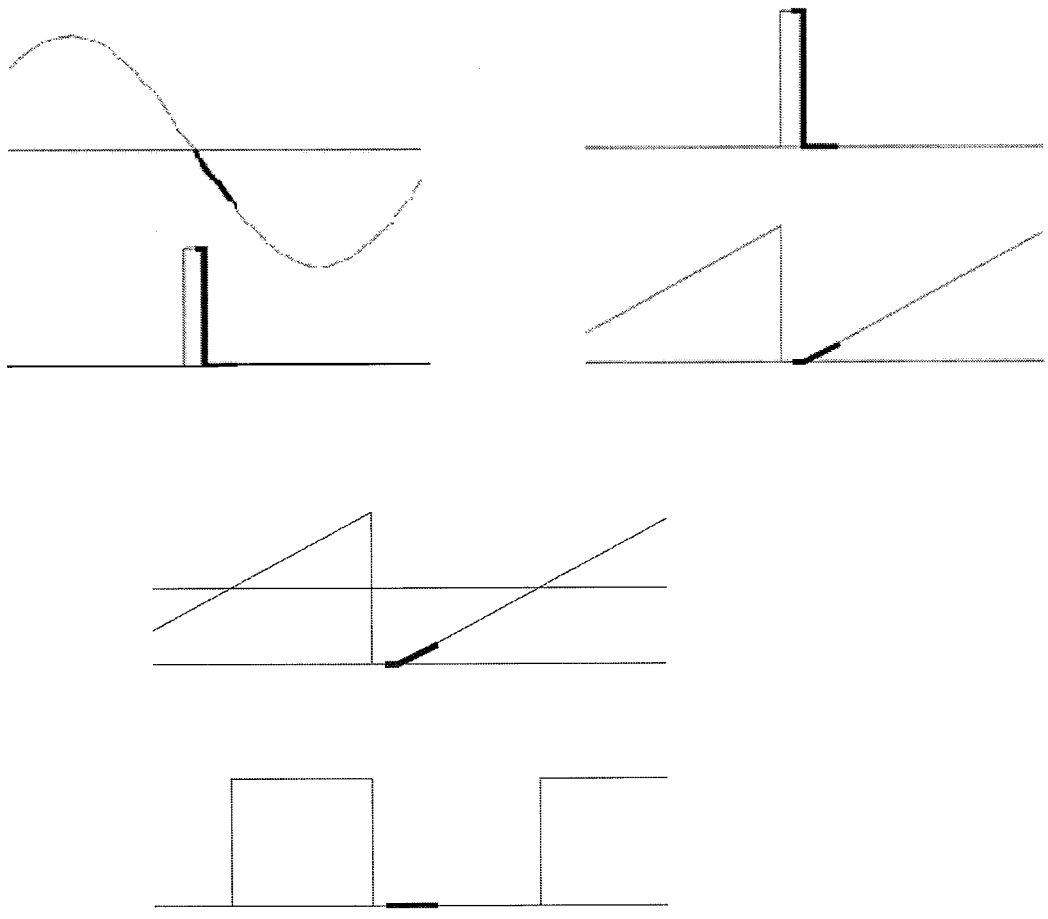
-Double pulse ใช้สำหรับวงจรจุดชนวนไทริสเตอร์ในวงจร 3 เฟส บริดจ์



รูปที่ 2.25 สัญญาณที่ได้จากวงจรคอนโทรลเฟสแบบครึ่งคลื่นด้วยการทริก

วงจร 3 เฟสที่ควบคุมด้วย TCA785 วงจรเรียงกระแส 3 เฟสที่ควบคุมด้วย TCA785 เช่น วงจรเรียงกระแส 3 เฟส ครึ่งคลื่น จะใช้วงจรจุดชนวนวงจรเดียวกัน เพราะมีไทริสเตอร์ทำงานบนสายไฟสลับ 3 เฟส เฟสละ 1 ตัว ลักษณะของวงจรจุดชนวนเกตจะใช้วงจรรวม TCA 785 3 ตัว แต่ละตัวทำงานควบคุมไทริสเตอร์ตัวละ 1 เฟส โดย TCA785 ที่รับแรงดัน Vsyn จากสายเฟส R จะจ่ายพัลส์จุดชนวนให้กับไทริสเตอร์ที่ทำงานบนเฟส R และสำหรับเฟส S และ T ก็เช่นเดียวกัน





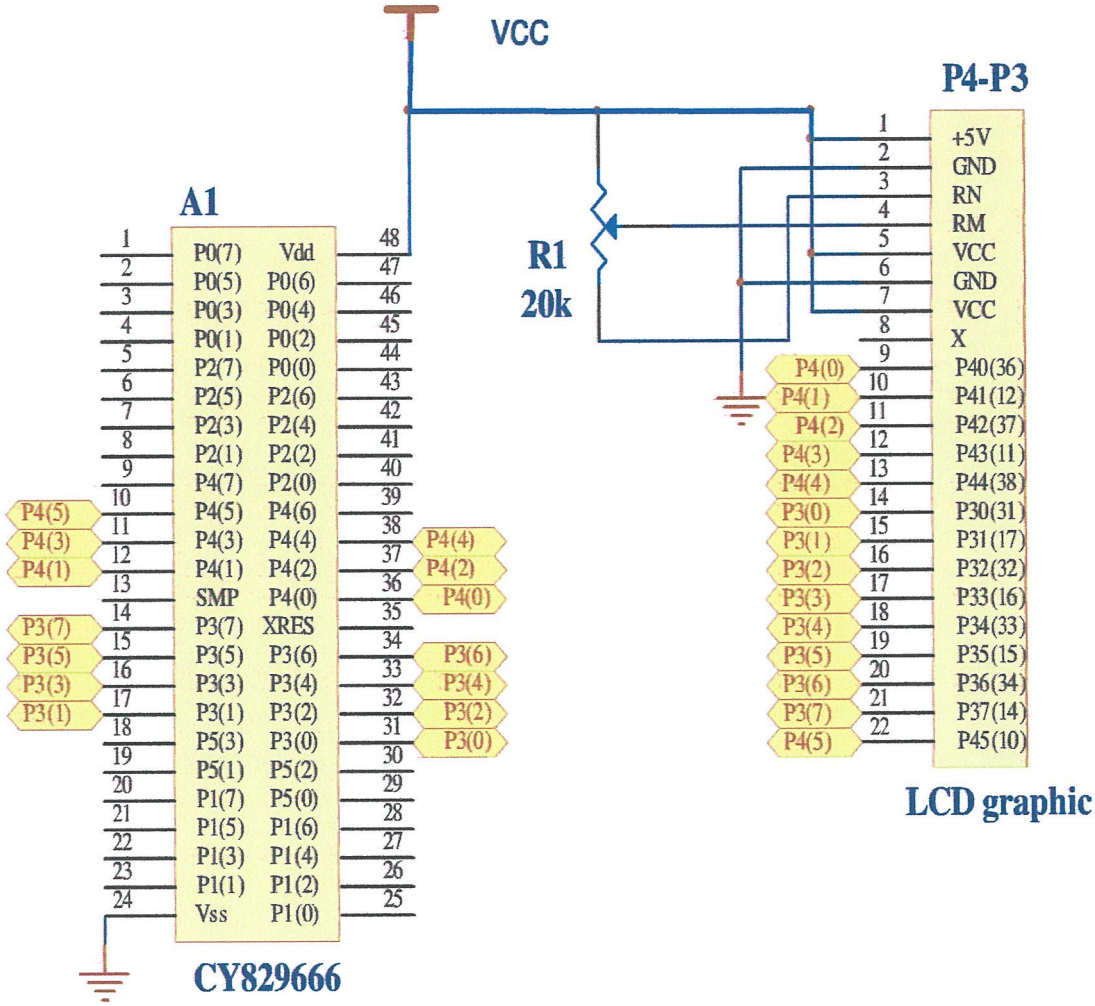
รูปที่ 2.26 รูปสัญญาณที่ได้จากวงจรคอนโทรลเฟส

2.2.5 Graphic LCD ABG240128

ABG240128 เป็น Graphic LCD Module ขนาด 240x128 Pixels ของ AV DISPLAY มีโหมดการแสดงผล 2 โหมดคือ โหมด Graphic กับ โหมด Text สามารถนำข้อมูลของ Text กับ Graphic มาทำการ OR, AND, EXOR กันได้ โดยการตั้งค่า Mode set ABG240128 ใช้ SAP1024 เป็น LCD Controller สำหรับรายละเอียดการต่อใช้งานของขาต่างๆ ของ LCD Module มีดังนี้

- ขา 1A เป็นขาไฟเลี้ยงสำหรับ LED Backlight ต่อกับไฟเลี้ยง +5V
- ขา 2 K เป็นขาไฟเลี้ยงสำหรับ LED Backlight ต่อลง Ground
- ขา 3 Vout เป็นขาจ่ายไฟ -15V
- ขา 4 V0 เป็นขาสำหรับปรับความเข้มของจอ
- ขา 5 PD เป็นขาควบคุมการทำงานของ Negative Converter ต่อเข้ากับ +5V
- ขา 6 GND ต่อลง Ground
- ขา 7 VDD ต่อไฟเลี้ยง +5V
- ขา 8 VEE ไฟเลี้ยงจอ LCD (ไม่ต่อใช้งาน)
- ขา 9 /WR เป็นขาสำหรับเขียนข้อมูลไปยัง LCD (Active low)
- ขา 10 /RD เป็นขาสำหรับอ่านข้อมูลจาก LCD (Active low)
- ขา 11 /CE เป็นขาสั่งให้ LCD ทำงาน (Active low)
- ขา 12 C/D เป็นขาเลือกว่าจะ อ่าน/เขียน ข้อมูล หรือ คำสั่ง ("1"=ข้อมูล, "0"=คำสั่ง)
- ขา 13 /RST เป็นขาสำหรับรีเซ็ต LCD
- ขา 14-21 DB0~DB7 เป็นบัสข้อมูล
- ขา 22 FS เป็นขาสำหรับเลือกขนาดของตัวอักษร ("1"=6x8, "0"=8x8)

การเชื่อมต่อกับ Graphic LCD ต้องใช้ขาข้อมูลทั้งทั้งหมด 8 ขา ไม่สามารถใช้การเชื่อมต่อแบบ 4-bits ได้ ดังนั้นการเชื่อมต่อกับ Graphic LCD จะเปลืองขาของไมโครคอนโทรลเลอร์พอสมควร



รูปที่ 2.27 ใช้ในการเชื่อมต่อ 240128 กับ CY8C29666

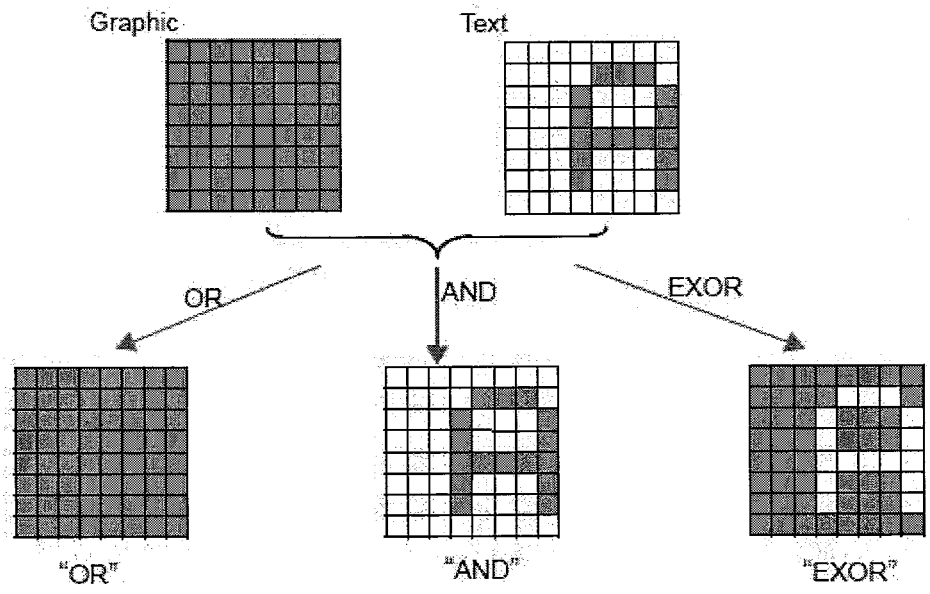
สำหรับการปรับความเข้มของจอ LCD นั้น ตัว LCD Module เองจะมีไฟเลี้ยง -15V ออกมาให้ที่ขา 3 เพื่อนำไปต่อเข้ากับ ตัวต้านทานปรับค่าได้ เอาไว้สำหรับปรับความเข้มของจอ LCD ดังรูปวงจรด้านบน

ตารางที่ 3.10 แสดงขั้นตอน ของการใช้ Initial

คำสั่ง	C/D	D7	D6	D5	D4	D3	D2	D1	D0	Hex	รายละเอียด
Power on	Power on										
Hard reset	ขา /RST="0" อย่างน้อย 1ms										
Control word set Graphic Home address set	0	0	0	0	0	0	0	0	0	00H	ตั้งค่าตำแหน่งเริ่มต้นของ Graphic = 0000H
	0	0	0	0	0	0	0	0	0	00H	
	1	0	1	0	0	0	0	1	0	42H	
Control word set Graphic Home address set	0	0	0	0	1	1	1	1	0	1EH	ตั้งค่า Graphic area = 001EH หรือ 240/8 = 30-bytes
	0	0	0	0	0	0	0	0	0	00H	
	1	0	1	0	0	0	0	1	1	43H	
Control word set Text Home address set	0	0	0	0	0	0	0	0	0	00H	ตั้งค่าตำแหน่งเริ่มต้นของ Text = 0F00H หรือ 128x30 = 3840-bytes
	0	0	0	0	0	1	1	1	1	0FH	
	1	0	1	0	0	0	0	0	0	40H	
Control word set Text Home address set	0	0	0	1	0	1	0	0	0	28H	ตั้งค่า Text area = 0028H หรือ 240/6 = 40-bytes
	0	0	0	0	0	0	0	0	0	00H	
	1	0	1	0	0	0	0	0	1	41H	
Mode set	1	1	0	0	0	0	0	0	1	81H	XOR mode ถ้าเปิดทั้ง Text และ Graphic จะเอาข้อมูลของ Text มา XOR กับ Graphic
Initial end											

การส่งข้อมูลคำสั่งที่มี 2-bytes นั้นจะส่งข้อมูลคำสั่ง low byte ไปก่อนแล้วตามด้วย high byte ตัวอย่าง การตั้งค่าตำแหน่งเริ่มต้น Text ข้อมูลของคำสั่ง เป็น 0F00H เราต้องส่ง 00H ไปก่อน ตามด้วย 0FH แล้วตามด้วย 40H คือคำสั่งการตั้งค่าตำแหน่งเริ่มต้นของ Text สำหรับคำสั่งทั้งหมดดูได้ใน Datasheet

สำหรับการตั้งค่า Mode set นั้น มีให้เลือกใช้งาน 3 โหมด คือ OR, AND และ EXOR การตั้งค่า Mode set นั้น เป็นการเลือกว่าจะเอาข้อมูลของ Text กับ Graphic มาทำการ OR, AND หรือ EXOR กัน ซึ่งการ OR, AND หรือ EXOR นั้นจะมีผลก็ต่อเมื่อมีการเปิดการแสดงผลแบบ Graphic และ Text พร้อมกันเท่านั้น ถ้าเปิดการใช้งานเฉพาะ Text หรือ Graphic การตั้งค่า Mode set จะไม่มีผล



รูปที่ 2.28 แสดงตัวอย่างของการตั้งค่า Mode set ในแบบต่างๆ

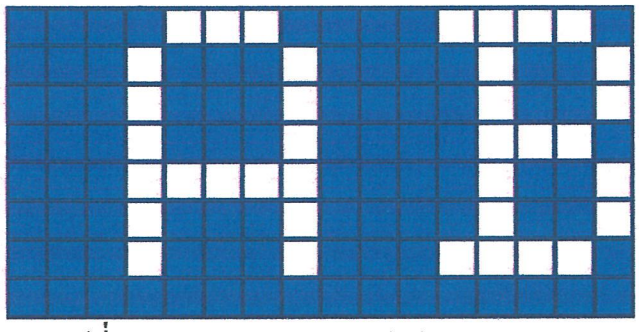
ตารางที่ 2.13 การเขียนข้อมูลลงไปใน Graphic area มีขั้นตอนดังนี้

ผคำสั่ง	C/D	D7	D6	D5	D4	D3	D2	D1	D0	Hex	รายละเอียด
Select font 8x8	ขา FS="0"										
Address pointer set	0	0	0	0	0	0	0	0	0	00H	ตั้งค่า Pointer ให้ชี้ไปที่ตำแหน่งเริ่มต้นของ Graphic = 0000H
	0	0	0	0	0	0	0	0	0	00H	
	1	0	0	1	0	0	1	0	0	24H	
Data write (Graphic)	0	x	x	x	x	x	x	x	x	xxH	เขียนข้อมูลลงในตำแหน่งของ Graphic Address pointer จะเพิ่มขึ้นอัตโนมัติ
	1	1	1	0	0	0	0	0	0	C0H	

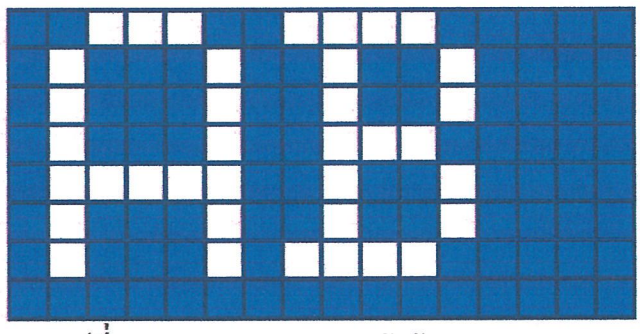
ตารางที่ 2.14 การเขียนข้อมูลลงไปใน Text area มีขั้นตอนดังนี้

คำสั่ง	C/D	D7	D6	D5	D4	D3	D2	D1	D0	Hex	รายละเอียด
Select font 6x8	ขา FS="1"										
Address pointer set	0	0	0	0	0	0	0	0	0	00H	ตั้งค่า Pointer ให้ชี้ไปที่ตำแหน่งเริ่มต้นของ Text = 0F00H
	0	0	0	0	0	1	1	1	1	0FH	
	1	0	0	1	0	0	1	0	0	24H	
Data write (Text)	0	x	x	x	x	x	x	x	x	xxH	เขียนข้อมูลลงในตำแหน่งของ Text Address pointer จะเพิ่มขึ้นอัตโนมัติ
	1	1	1	0	0	0	0	0	0	C0H	

สำหรับขา FS นั้นจะเป็นขาเลือกขนาดของตัวอักษร ขานี้จะมีผลต่อการแสดงผลในโหมด Graphic ด้วยถ้าหากแสดงผล Graphic อยู่ แล้วเลือกขนาดตัวอักษรเป็น 6x8 ภาพที่ได้จะไม่เต็มจอ ส่วนความแตกต่างระหว่างขนาดของตัวอักษร 6x8 กับ 8x8 ใน Text mode นั้นจะต่างกันที่ระยะห่างของตัวอักษร ดังรูปตัวอย่างด้านล่าง



รูปที่ 2.29 การแสดง FS="0" ตัวอักษรขนาด 8x8

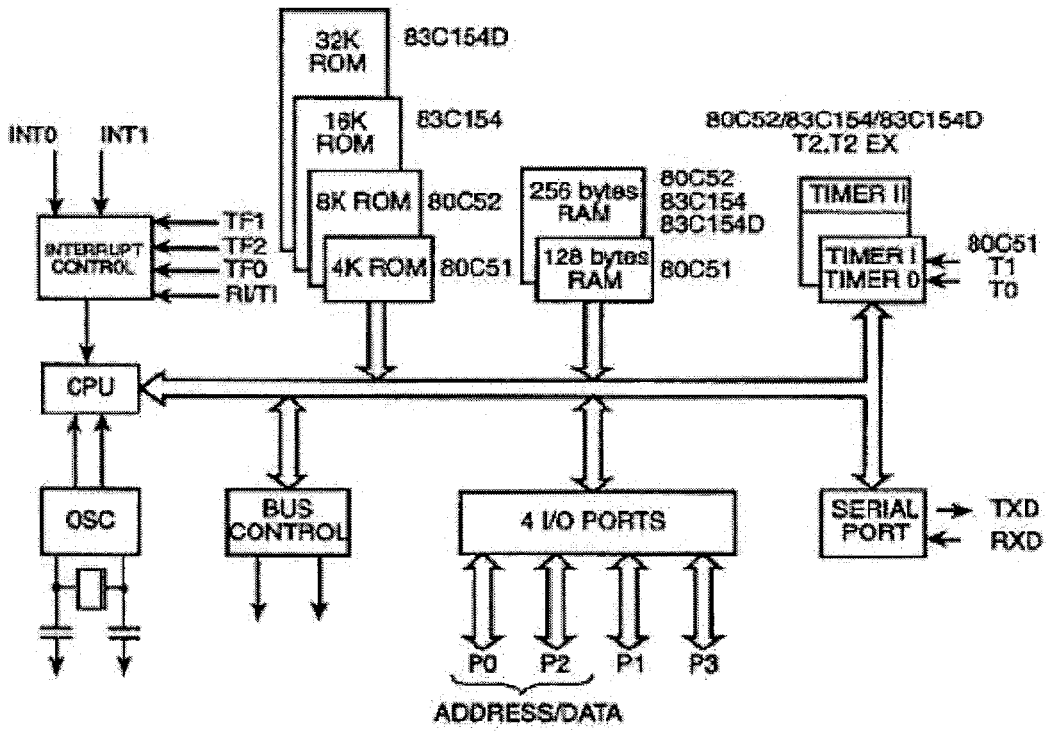


รูปที่ 2.30 การแสดง FS="1" ตัวอักษรขนาด 6x8

2.2.6 ตัวประมวลผลข้อมูลเซ็นเซอร์ (Microcontroller)

สำหรับในโครงงานตู้อบเด็กทารกนี้เราใช้ไมโครคอนโทรลเลอร์เป็นตัวทำงานประมวลผลควบคุม ซึ่งจะทำหน้าที่ในการควบคุมระบบการทำงานของตู้อบเด็กทารกทั้งระบบ

โครงสร้างของไมโครคอนโทรลเลอร์ MCS 51



รูปที่ 2.31 โครงสร้างของไมโครคอนโทรลเลอร์

คุณสมบัติเบื้องต้นของ Micro controller ในตระกูล MCS-51 จะมีดังนี้

-มี Core CPU ที่เป็น 8-Bit และชุดคำสั่งที่เหมาะในงานควบคุม และสามารถประมวลผลทาง Logicกับข้อมูลในระดับ BIT ได้

-มีหน่วยความจำโปรแกรม 4K ภายใน และรองรับการใช้งานของหน่วยความจำโปรแกรมได้ 64K

-มีหน่วยความจำ ข้อมูล (RAM) 128 Bytes ภายใน และรองรับการใช้งานของหน่วยความจำ ข้อมูลได้ ถึง 64K

-มี Port ที่เป็นได้ทั้ง I / O ทั้งหมด 4 port และสามารถใช้งานได้ในระดับ BIT

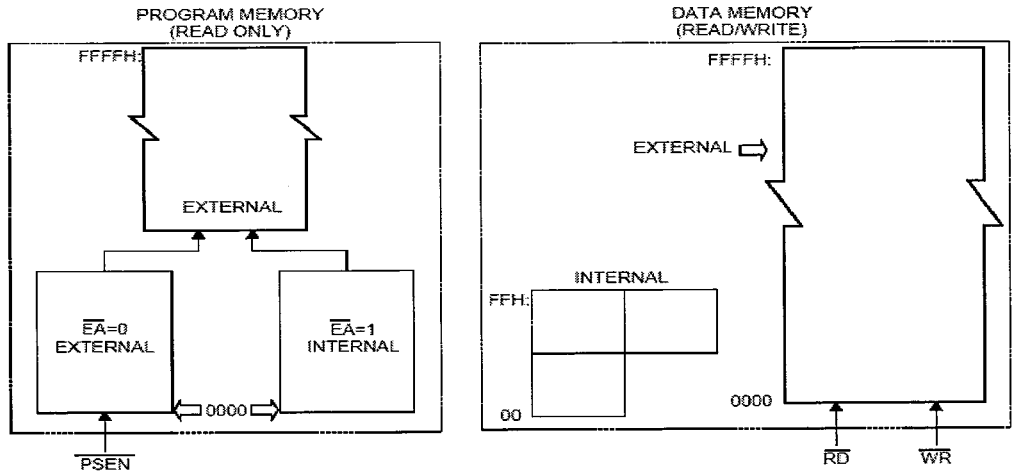
-มีส่วน Timer / Counter ขนาด 16 Bit สองชุด สำหรับใช้ในการจับเวลา หรือนับจำนวน

-มี Full duplex UART สำหรับใช้ รับ/ส่ง ข้อมูลแบบอนุกรม รับ Interrupt ได้จาก 6 แหล่งกำเนิดโดย มี 5 ตำแหน่งของ ISR และการ Interrupt โดยสามารถจัดระดับความสำคัญได้ 2 ระดับ

-มีตัวกำเนิดความถี่ Clock ภายใน

โครงสร้างของหน่วยความจำใน MCS-51Logical separation of program and data memory

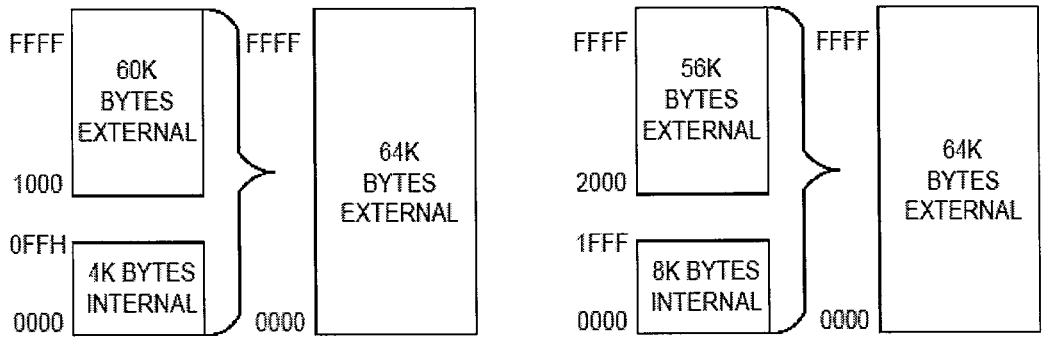
สำหรับ Micro controller MCS-51 นั้นถูกออกแบบมาให้มีหน่วยความจำสำหรับเก็บโปรแกรม (Op-code) และหน่วยความจำสำหรับเก็บข้อมูลที่แยกออกจากกันดังรูป



รูปที่ 2.32 โครงสร้างของหน่วยความจำใน MCS-51

การออกแบบของ Data memory ที่แยกออกมานี้ จะทำให้สามารถเรียกใช้งานได้โดยใช้ Address เพียง 8 Bit เท่านั้น ซึ่งจะทำได้อย่างรวดเร็วใน CPU ที่เป็น 8 Bit แต่การใช้ Address เพียง 8 Bit นี้ จะทำให้อ้างถึงตำแหน่งของหน่วยความจำได้เพียง 256 ตำแหน่งเท่านั้น (00h – FFh) ซึ่งก็เพียงพอสำหรับการอ้างถึงตำแหน่งของ Internal Data Memory อย่างไรก็ดีการอ้างถึงตำแหน่ง Data memoryโดยใช้ Address แบบ 16 Bit สำหรับ External data memory ก็สามารถทำได้ โดยใช้ DPTR: Data pointer (Data memory address register)

ส่วนของ Program memory จะเป็นหน่วยความจำที่อ่านได้เพียงอย่างเดียว และสามารถมีได้ทั้งหมด 64K ตำแหน่ง

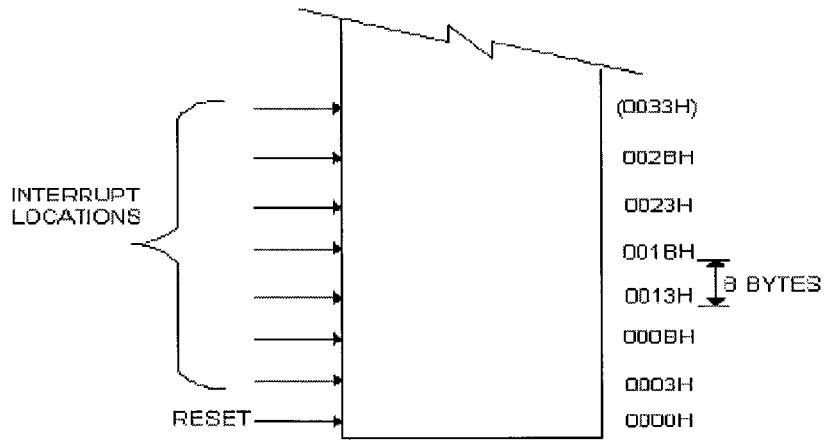


รูปที่ 2.33 โปรแกรม memory ของ MCS 51

สำหรับ MCS-51 ในแบบที่มี Program memory ภายใน ก็จะมีพื้นที่ในการเก็บโปรแกรมภายใน IC เอง 4K, 8K, 16K หรือ 32K Address (ตามเบอร์ของ IC ที่ใช้) สำหรับ Address ที่มากกว่านี้ ก็จะเป็น Program memory ภายนอก ส่วน MCS-51 ที่ไม่มี internal program memory ส่วนของ Program memory ทั้งหมดจะอยู่ภายนอก

สำหรับการอ่าน External program memory นั้น MCS-51 จะใช้ขาสัญญาณ PSEN (Program Store Enable)

สำหรับ Data memory ซึ่งสามารถที่จะอ่านหรือเขียนข้อมูลไปได้ ก็จะมี Address ที่แยกออกจาก Program memory และมีตำแหน่งของ External data memory ได้ทั้งหมด 64K Address ในการติดต่อกับ External data memory นั้น MCS-51 จะใช้ขาสัญญาณ RD และ WR Program memory



รูปที่ 2.34 การต่อ External program memory และ External data memory ร่วมกัน

ในกรณีที่ต้องการ ส่วนของ External program memory และ External data memory ร่วมกันนั้น จะทำได้โดยการนำสัญญาณ PSEN และ RD มารวมกัน โดยใช้ AND gate ก็จะทำให้ได้สัญญาณที่เป็นการอ่าน External Program/Data memory

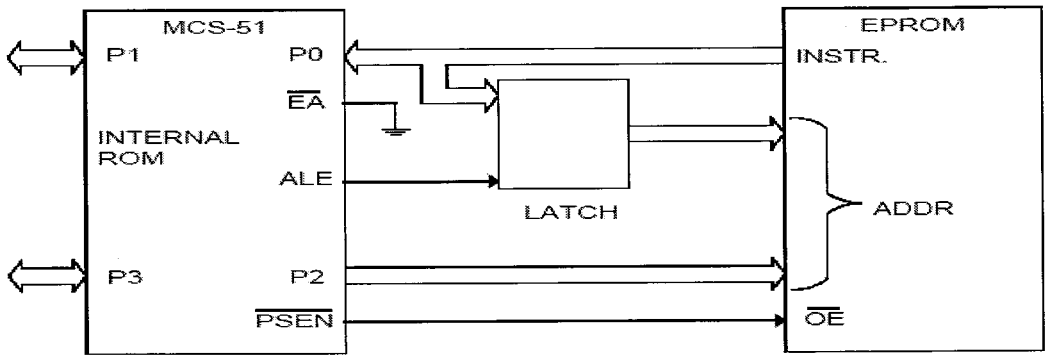
รูปแสดงส่วนของ Program memory ในตำแหน่งเริ่มต้น ซึ่งเมื่อ CPU เริ่มการทำงานหลังจากการ Reset ก็จะเริ่มการทำงานตามคำสั่งที่ Address 0000h และสำหรับ Address ที่แสดงต่อมานั้น จะตำแหน่งที่อยู่ของ ISR: Interrupt Service Routine โดยการทำงานของ Interrupt ใน MCS-51 เมื่อเกิดสัญญาณ Interrupt เข้ามา มันก็จะกระโดดการทำงานมายังโปรแกรมในตำแหน่งที่กำหนดนี้ นั่นเองตัวอย่างเช่น สำหรับ Interrupt 0 จากภายนอก เมื่อ MCS-51 ได้รับสัญญาณ

Interrupt นี้ มันก็จะกระโดดการทำงานมายังโปรแกรมใน Address 0003h และสำหรับ Interrupt ที่เกิดจาก Timer 0 ก็จะกระโดดการทำงานมายังโปรแกรมใน Address 000Bh และ Interrupt 0 จากภายนอก ก็จะกระโดดการทำงานมายังโปรแกรมใน Address 0013h . ในแต่ละ Address กำหนดให้สำหรับ ISR นั้นจะมีพื้นในการเก็บโปรแกรมได้ 8 Address ซึ่งถ้า ISR ที่ต้องการ เป็นโปรแกรมที่สั้นๆ ก็จะสามารถใส่เข้าไปได้ แต่ถ้าเป็น ISR ที่ยาวมากแล้วก็จะทำได้ โดยการใช้คำสั่ง Jump ไปยังโปรแกรมที่ต้องการอีกที

External Program Memory

สำหรับ MCS-51 ที่มี Internal program memory นั้น ผู้ใช้สามารถที่จะเลือกได้ว่า จะใช้งานของ Internal program memory นั้น หรือไม่ โดยการต่อของขาสัญญาณEA: External access เข้ากับ VCC หรือ GNDเช่นถ้า MCS-51 มี Internal program memory 4K (0000h-0FFFh) แล้วต่อขาสัญญาณ EA นี้เข้ากับ VCC การ Fetch คำสั่งที่ Address น้อยกว่า 0FFFh ก็จะได้จาก Internal program memory และถ้าเป็น Address ตั้งแต่ 1000h ก็ จะเป็นการอ่านจาก External program memory นั่นเอง แต่ถ้าต่อขา EA เข้ากับ GND การ Fetch คำสั่งทั้งหมดจะกระทำกับ External program memory สำหรับ MCS-51 ที่ไม่มี internal program memory แล้ว ขา EA จะต้องต่อ GND เสมอ สัญญาณ PSEN ซึ่งเป็นสัญญาณที่ใช้ในการ Fetch คำสั่งจาก External program memory นั้นจะไม่ทำงานเมื่อเป็นการ Fetch คำสั่งจาก internal program memory

การต่อ External program



รูปที่ 2.35 การต่อ External program memory

MCS-51 จะใช้ขาสัญญาณ ของ I/O port 16 bit (Port 0 และ Port 2) มาทำหน้าที่เป็น Bus ของระบบ โดยจะใช้งานของ Port 0 ทำหน้าที่เป็น Address และ Data bus สลับกัน คือส่งค่าของ Low byteของ Program counter (PCL) ออกมาที่ Port 0 นี้ หลังจากส่งค่าของ PCL ออกมาแล้วมันจะเข้าสู่สภาวะ Float เพื่อรอรับคำสั่งที่จะอ่านได้จาก External program memory ในระหว่างที่

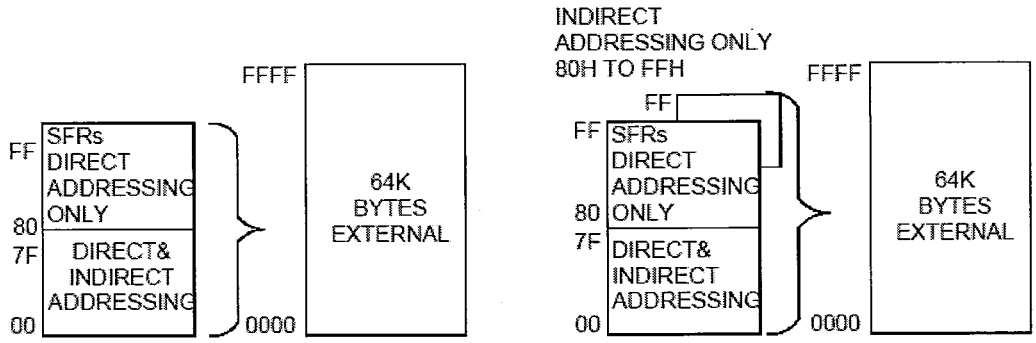
ค่าของ PCL ออกมาที่ P0 นี้ สัญญาณ ALE: Address Latch Enable จะไปทำให้ค่าของ PCL ถูกเก็บเข้าที่ LATCH และในเวลานั้น ค่าของ PCH ก็ถูกส่งออกมาที่ Port 2 เพื่อสร้างเป็น Address ขนาด 16 bit แล้ว สัญญาณ PSEN ก็จะเป็นตัวอ่านข้อมูลจาก Memory ที่ต้องการ

Data memory

Data memory ของ MSC-51 นั้นจะแบ่งออกเป็น Internal data memory และ External data memory โดยการใช้งานของ Data memory ทั้งสองส่วนนี้จะมี Address ที่แยกจากกันด้วย

MCS-51 with 128 bytes Internal Data memory (ซ้าย)

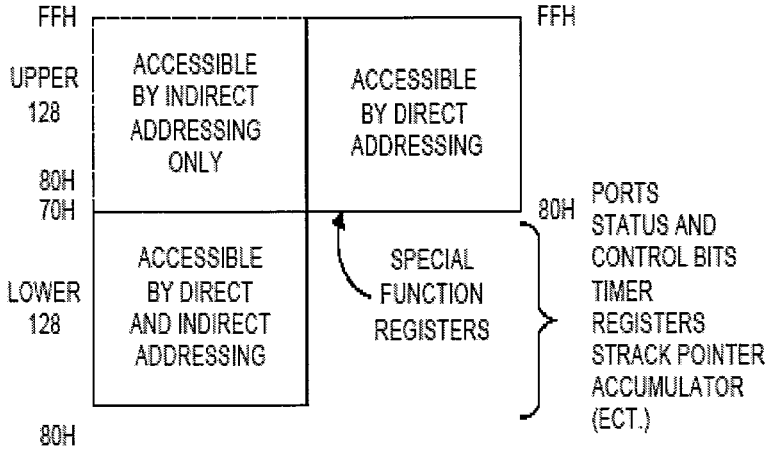
MCS-51 with 256 bytes Internal Data memory (ขวา)



รูปที่ 2.36 Data memory ของ MSC-51

Internal Data Memory

ผังการจัดแบ่งพื้นที่ของ Internal data memory จะเป็นดังรูป ซึ่งมันจะถูกแบ่งออกเป็นสามส่วนด้วยกัน คือ Lower 128, Upper 128 และ SFRและจากการที่ Internal data memory นั้นมีเพียง 256 ตำแหน่งเท่านั้น ทำให้การอ้าง Address สามารถทำได้ โดยใช้เพียง 8 Bit และจากการที่ Internal data memoryนั้นมีเพียง 256 ตำแหน่งเท่านั้น ทำให้การอ้าง Address สามารถทำได้ โดยใช้เพียง 8 Bit



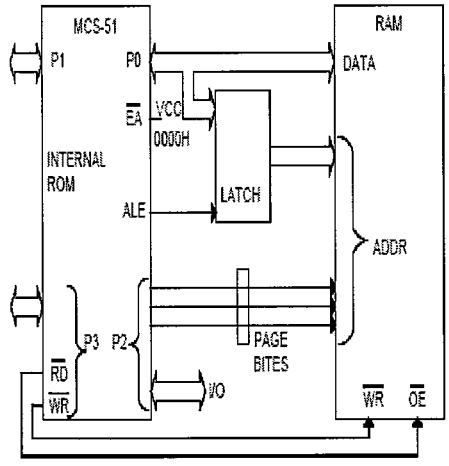
รูปที่ 2.37 Internal Data Memory

แต่ด้วยเทคนิคของการอ้างถึงตำแหน่งข้อมูล (Addressing mode) ทำให้ได้ตำแหน่งของ Internal data memory ทั้งหมด 384 bytes

โดยสำหรับข้อมูลตั้งแต่ Address 80h – FFh ถ้าอ้างถึงข้อมูลที่ Address นั้นๆ ด้วยวิธีของ Direct addressing ก็จะได้ข้อมูลที่มาจากคนละส่วน กับการอ้างถึงข้อมูลที่ Address เดียวกันนั้น ด้วยวิธีของ Indirect addressing จากรูปจะเห็นได้ว่า Memory ในส่วนของ Upper 128 ก็จะมี Address เดียวกับ Memory ในส่วนของ SFR แต่จะใช้วิธีการเข้าถึงข้อมูลที่แตกต่างกันนั่นเอง

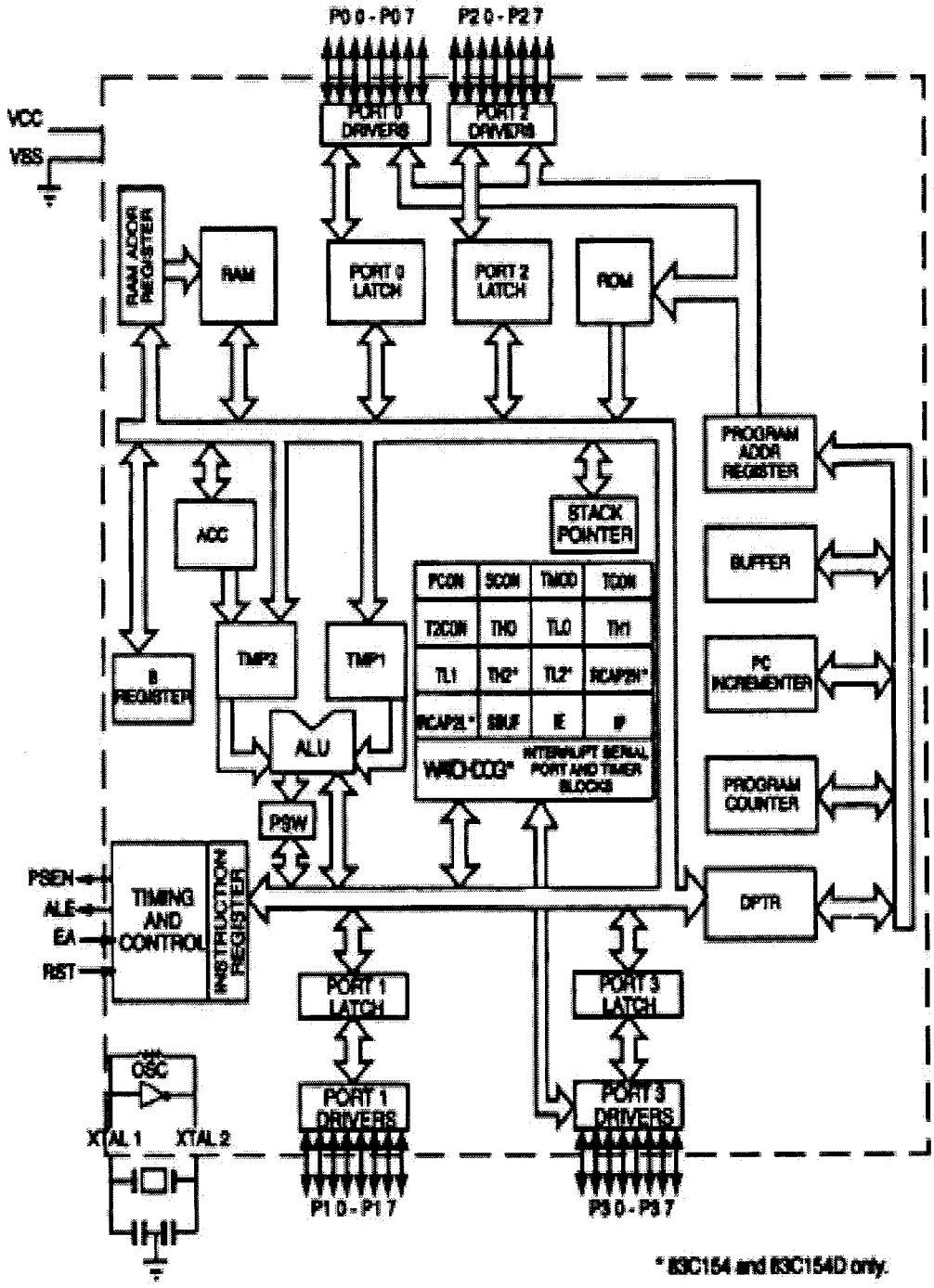
External Data Memory

การต่อใช้งานของ External data memory จะทำได้ดังตัวอย่าง ซึ่งเป็นการต่อใช้งานของ RAMขนาด 2K โดยให้ MCS-51 นี้ ทำงานจาก internal program memory



รูปที่ 2.38 การต่อใช้งานของ External data memoryโดยให้ MCS-51 นี้ ทำงานจาก Internal Program memory

โครงสร้างภายในของ Micro controller MCS-51



รูปที่ 2.39โครงสร้างภายในของ Micro controller MCS-51

จะเห็นได้ว่าตำแหน่งต่างๆ ใน SFR จะไม่ได้ถูกใช้ทั้งหมด การอ่านข้อมูลจากตำแหน่งที่ยังไม่ได้ กำหนดนี้ จะทำให้ได้ค่าสุ่มออกมา ส่วนการเขียนข้อมูลเข้าไปจะไม่มีผลใดๆ แต่อย่างไรก็ดี เราไม่ควรเขียนข้อมูล '1' เข้าไปในตำแหน่งที่ไม่ได้กำหนดนี้ เนื่องจากมันอาจถูกใช้งานในการควบคุม ฟังก์ชันการทำงานที่จะเพิ่มเข้ามาในอนาคต

Accumulator

ACC จะคือ accumulator ซึ่งจะมีตำแหน่งในหน่วยความจำภายในเท่ากับ 0E0H มีขนาด 8 Bit เป็น Register ที่ใช้กันมาก ในรหัสคำสั่งช่วยจำจะใช้อักษร A

B Register

จะมีตำแหน่งในหน่วยความจำภายในเท่ากับ 0F0H มีขนาด 8 Bit จะใช้ในการคูณและการหารเท่านั้น หรืออาจใช้ในการเก็บข้อมูลอื่นๆ ที่ต้องการก็ได้

Program status word (PSW)

จะมีตำแหน่งในหน่วยความจำภายในเท่ากับ D0H มีขนาด 8 Bit แต่ละ Bit จะบอกถึงสภาวะต่างๆ ในการทำงานของ CPU (Flag) ซึ่งแต่ละ Bit ของ PSW สามารถกำหนดให้เป็น 1 หรือ 0 ได้โดยคำสั่ง SETB หรือ CLRB ตามลำดับค่าตำแหน่ง Bit Address 0 ถึง Bit 7 ของ PSW เท่ากับ D0h ถึง D7h

Stack Pointer

จะมีตำแหน่งในหน่วยความจำภายในเท่ากับ 081H มีขนาด 8 Bit Register นี้ใช้ชี้ตำแหน่งในหน่วยความจำภายใน 8051 ที่จะใช้สร้างเป็น Stack ในการทำงานของ MCS-51 ค่าของ SP นี้จะมีค่าที่เพิ่มขึ้น ก่อนที่จะมีการเก็บข้อมูลเข้าไปด้วยคำสั่ง PUSH หรือ CALL

การกำหนดตำแหน่งของ Memory ที่จะสร้างเป็น Stack นั้นสามารถกำหนดให้เป็นที่ใดก็ได้ใน Internal data memory และเมื่อทำการ Reset ค่าเริ่มต้นของ SP จะค่าเป็น 07h ซึ่งจะทำให้การเก็บข้อมูลในตำแหน่งแรกของ Stack เริ่มที่ Internal data memory ตำแหน่งที่ 08h

Data Pointer Register

Data Pointer (DPTR) จะอยู่ในตำแหน่งหน่วยความจำภายในเท่ากับ 82h และ 83h DPTR นี้ประกอบไปด้วย Register ขนาด 8 Bit 2 ตัวคือ DPH และ DPL ซึ่ง DPTR นี้ จะใช้ในการชี้ตำแหน่งของข้อมูลของ External data memory แบบ 16 Bit ในการแก้ไขข้อมูลใน Register DPTR จะทำได้ทีละ 16 Bit หรือกระทำทีละ 8 Bit ก็ได้ (DPH, DPL)

PORT 0 ถึง 3

จะตำแหน่งในหน่วยความจำภายในเท่ากับ 80h, 90h, 0A0h, 0B0h เป็น Registerขนาด 8 Bit การเขียนข้อมูลไปยังหน่วยความจำแต่ละตำแหน่งเป็นการส่งข้อมูลไปยังพอร์ทนั้นๆ ของ MCS-51 ข้อมูลที่เขียนออกไปจะถูก Latch ค้างไว้ที่ Register นี้ และปรากฏแต่ละ Bit ของ Port เช่นถ้าเขียนข้อมูล 18h ไปที่หน่วยความจำตำแหน่ง 80h ก็จะปรากฏ Logic 0001 1000 ที่ขา 7 ถึง 0 ของ Port 0 ในการอ่านข้อมูลจาก Register แต่ละตัวจะเป็นการอ่านสภาวะ Logic ที่มีปรากฏอยู่แต่ละขาของ Port นั้นๆ

Serial Data Buffer (SBUF)

จะตำแหน่งหน่วยความจำภายในเท่ากับ 99H มีขนาด 8 Bit แต่จากโครงสร้างภายในแล้วมันคือ Register 2 ตัวที่มีชื่อเดียวกัน ตัวหนึ่งสำหรับเก็บข้อมูลที่จะส่งแบบอนุกรม และอีกตัวหนึ่งสำหรับรับข้อมูลแบบอนุกรมที่เข้ามา ในการเขียนข้อมูลเข้าที่ SBUF มันจะถูกเขียนไปยังที่สำหรับเก็บข้อมูลสำหรับส่ง และเริ่มต้นการส่งข้อมูล ส่วนการอ่านข้อมูลจาก SBUF ก็จะเป็นการอ่านค่าของข้อมูลที่รับเข้ามาได้

TIMER Register

คู่ของ Register (TH0, TL0) (TH1, TL1) และ (TH2, TL2) ซึ่งอยู่ในตำแหน่งหน่วยความจำภายใน (8Ch, 8Ah) (8Dh, 8Bh) และ (0CDh, 0CCh) ตามลำดับ ซึ่งจะใช้ในการเก็บค่าของการนับแบบ 16 Bit ในการใช้งานเป็น Timer หรือ Counter ใน 80C51 จะมี Timer อยู่ 2 ชุดคือ Timer 0 และ Timer 1 ใน Timer แต่ละชุดจะมี Register ขนาด 8 Bit อยู่ 2 ตัว เพื่อเก็บค่าการนับของ Timer ได้สูงสุดถึง 16 Bit

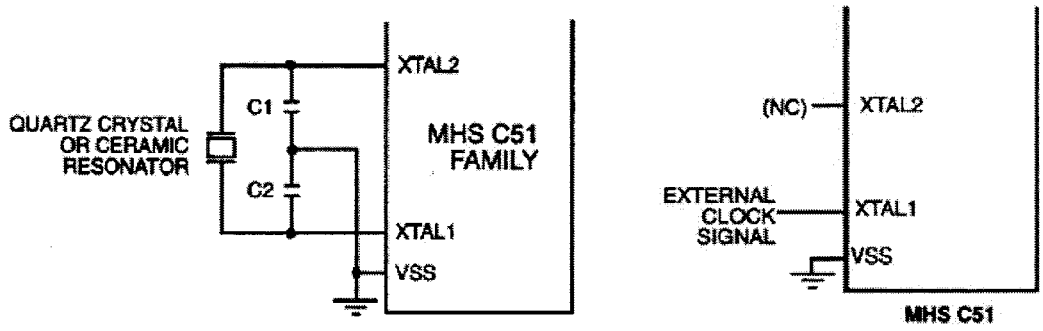
การกำหนดการทำงานของวงจร Timer ในโมด Timer หรือ Counter ทำได้โดยการกำหนดใน Register TMOD (Timer/Counter Mode Control Register) การทำงานเป็น Timer จะให้ Register ใน Timer 0 หรือ 1 ทำการนับจำนวนไซเคิลของสัญญาณนาฬิกา ในการให้วงจร Timer ทำงานเป็น Counter คือการใช้ Register THx และ TLx ทำการนับจำนวนไซเคิลของสัญญาณที่เข้ามาทางขา T0 หรือ T1

Control Register

SFR ที่ชื่อ IP, IE จะสำหรับกำหนดรูปแบบการทำงาน และสถานะของการ Interrupt TMOD, TCON, T2CON จะสำหรับกำหนดรูปแบบการทำงานของ Timer/Counter และ SCON จะสำหรับกำหนดรูปแบบการทำงานของ Serial port และ PCON จะสำหรับกำหนดรูปแบบการใช้พลังงานของตัว CPU เอง

CPU Timing

ใน MCS-51 จะมีส่วนของวงจรกำเนิด Clock อยู่ภายในแล้ว ซึ่งสามารถใช้เป็นสัญญาณนาฬิกาให้กับ CPU ได้ ในการใช้งานของ On-Chip Oscillator นั้น ก็เพียงต่อ Crystal หรือ Ceramic resonator ที่ขา XTAL1 กับ XTAL2 และตัวเก็บประจุ ดังแสดงในรูปที่ 2.41



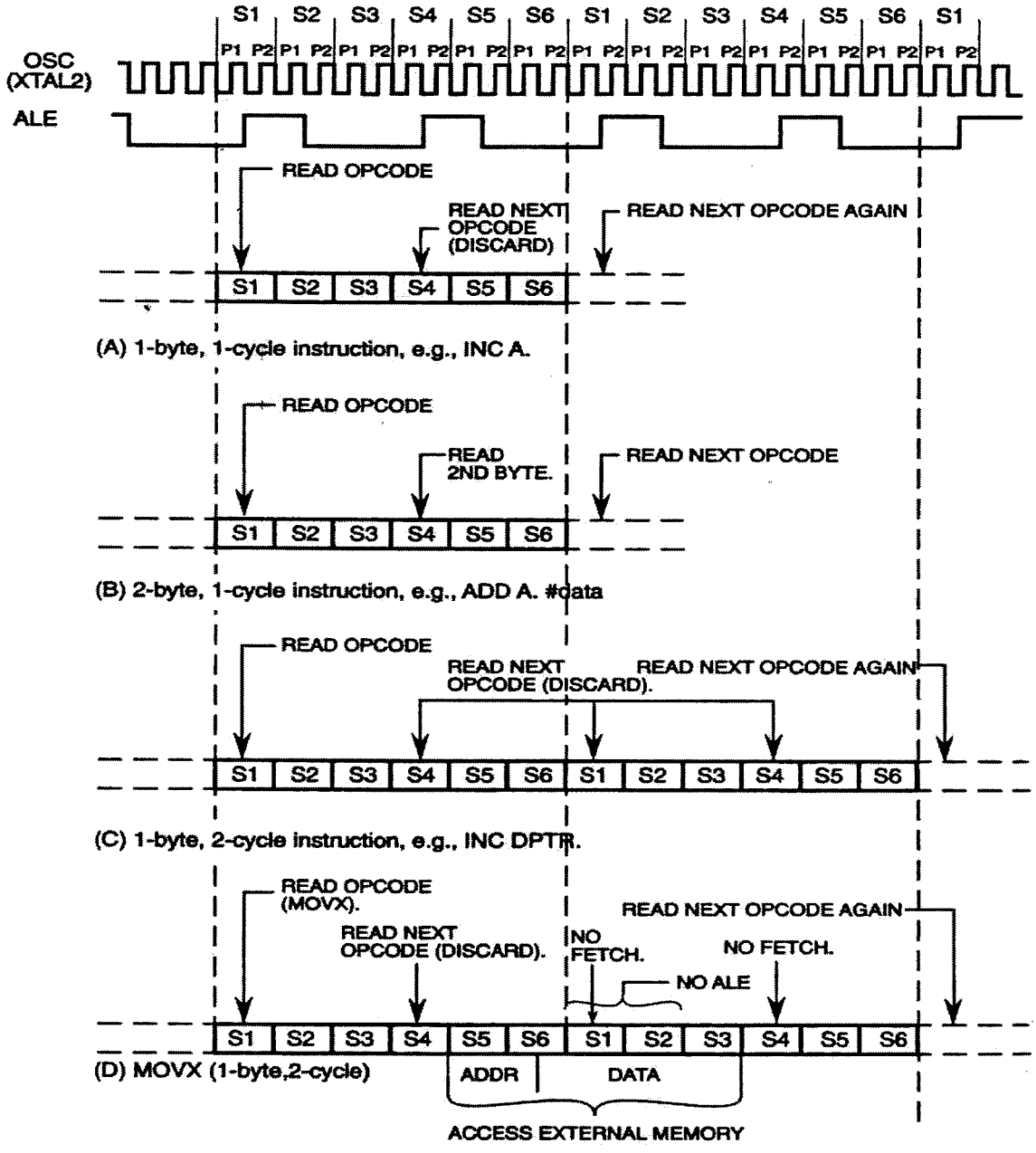
รูปที่ 2.40 การต่อ CPU Timing

อย่างไรก็ดี ถ้าต้องการใช้สัญญาณ Clock จากภายนอก ก็จะทำได้โดยการต่อสัญญาณ Ext. Clock เข้าที่ขา XTAL1 ดังรูป

Machine Cycles

สำหรับแต่ละ Machine cycle ของ MCS-51 จะประกอบด้วยการทำงาน 6 States (S1 – S6) ซึ่งในแต่ละ State ของการทำงาน จะใช้เวลา 2 Clocks ดังนั้นถ้าใช้ Clock 12 MHz ก็จะได้เวลาในการทำงานของ 1 Machine cycle คือ 1 μS

ดังแสดงในรูป จะเป็นตัวอย่างของการ Fetch / Execute ของคำสั่ง ซึ่งจะในแต่ละ Machine cycle จะมีการ Fetch รหัสคำสั่ง 2 ครั้ง ถึงแม้ว่าคำสั่งนั้นๆ จะเป็นคำสั่งแบบ 1 Byte ก็ตาม ในกรณีที่คำสั่งนั้นๆ ไม่ต้องการข้อมูล Byte ที่สอง CPU ก็เพียงแต่ไม่สนใจข้อมูลที่ Fetch ได้เกินมา และค่าของ PC ก็จะไม่เพิ่มขึ้น

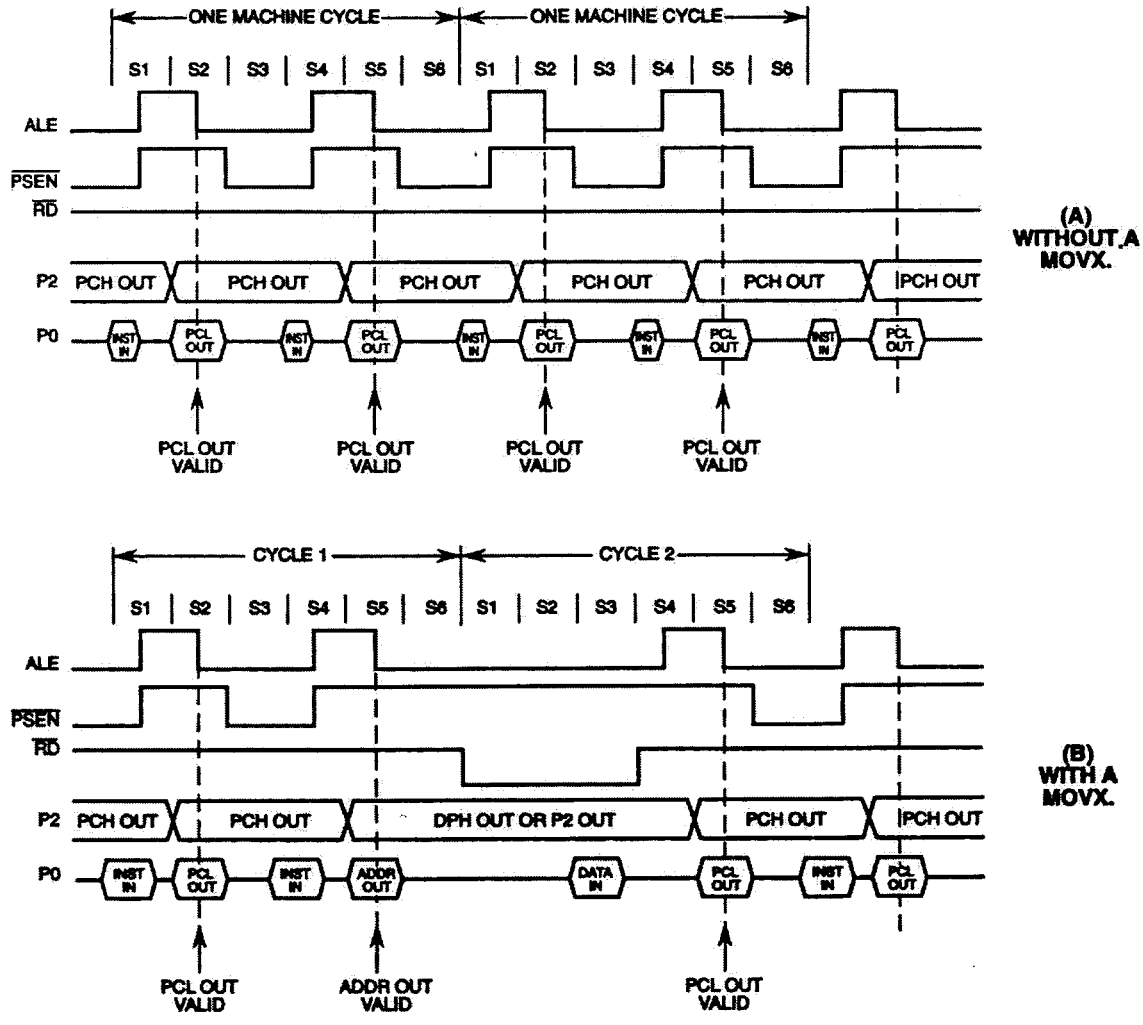


รูปที่ 2.41 Machine Cycles

จากรูป a และ b ซึ่งเป็นคำสั่งที่ใช้การทำงาน 1 Machine cycle ที่ S1 CPU ก็จะอ่าน Op-code ที่ต้องการเข้ามายัง Instruction register และที่ S4 จะมีการ Fetch ครั้งที่สองเกิดขึ้น ซึ่งในรูป a การ Fetch ครั้งที่สองนี้ CPU จะไม่สนใจข้อมูลที่ได้มา ส่วนในรูป b ข้อมูลที่ได้มาก็จะเป็น Byte ที่สองของคำสั่งนั่นเอง

จากรูป 2.42 ซึ่งเป็นการทำงานของคำสั่งที่ใช้การทำงาน 2 Machine cycles CPU จะไม่สนใจข้อมูลที่ได้จากการ Fetch 3 ครั้งด้วยกันและในรูป d ซึ่งเป็นการทำงานของคำสั่ง MOVX ซึ่งเป็นการติดต่อกับ External data memory ในกรณีนี้ จะไม่มีการ Fetch เกิดขึ้น เนื่องจาก CPU จะต้องใช้ระบบ BUS เดียวกันนี้ ในการติดต่อกับหน่วยความจำภายนอก

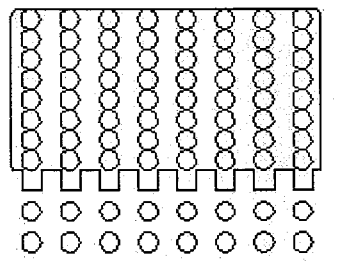
ขบวนการ Fetch / Execute นี้จะเหมือนกัน ไม่ว่าจะเป็นการทำงานของโปรแกรมที่อยู่ใน Internal หรือ External program memory ดังนั้นแล้วมันจะใช้เวลาในการทำงานของคำสั่งต่างๆ เท่ากัน



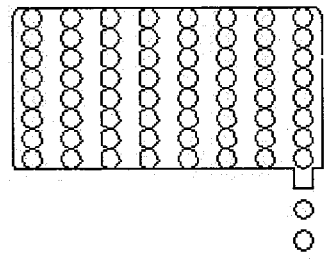
รูปที่ 2.42 ขบวนการ Fetch / Execute

การส่งผ่านข้อมูลแบบพล็อตขนาน

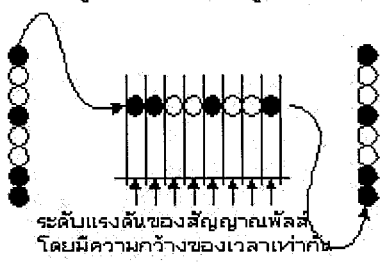
จากรูป a จะเป็นการแสดงถึงจังหวะการทำงานของสัญญาณต่างๆ ของการ Fetch เมื่อคำสั่งนั้นอยู่ใน External program memory ซึ่งสัญญาณ จะเกิดขึ้น 2 ครั้งต่อ Machine cycle ในรูป b จะแสดงสัญญาณต่างๆ ที่เกิดขึ้นเมื่อกระทำคำสั่งที่มีการติดต่อกับ External data memory การ Fetch จะถูกข้ามไป 2 ครั้ง และจะเห็นว่าจังหวะเวลาของการอ่านข้อมูลจาก External data memory นั้นจะใช้เวลาที่มากกว่าการอ่านข้อมูลจาก Program memory ข้อมูลในไมโครคอนโทรลเลอร์ที่เราใช้ศึกษาอยู่นี้ จะเป็นข้อมูลที่มีความยาวขนาด 1 ไบต์ หรือ 8 บิตซึ่งโดยปกติถ้าเราจะให้ส่ง ข้อมูลพร้อมๆกันไป 8 บิตจะเป็นวิธีการส่งข้อมูลแบบขนาน แสดงได้ดังรูป 11ก จะเป็นการส่งข้อมูลขนาด 8 บิตพร้อมกันไปยังอุปกรณ์ภายนอก และจะต้องมีจำนวนของสายสัญญาณจำนวน 8 เส้น เพื่อให้พอดีกับจำนวนของบิตที่ต้องการจะส่ง การส่งข้อมูลแบบขนานจึงทำให้มีการส่งข้อมูลที่มีความรวดเร็ว แต่ถ้าหากมีการสื่อสารข้อมูลในระยะไกล ก็จะต้องใช้จำนวนของสาย และระยะทางของสายมากขึ้นจึงทำให้มีการสิ้นเปลืองค่าใช้จ่ายสูง



รูป 11ก การส่งข้อมูลแบบขนาน



รูป 11ข การส่งข้อมูลแบบอนุกรม



รูป 11ค การเปลี่ยนสัญญาณจากขนานเป็นอนุกรม และจากอนุกรมเป็นขนาน

รูปที่ 2.43 การส่งผ่านข้อมูลแบบพล็อตขนาน

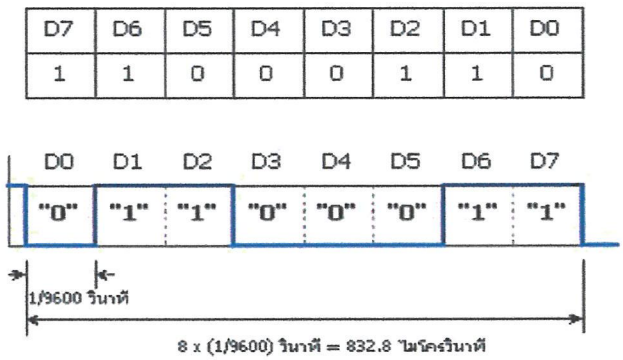
แสดงการส่งข้อมูลแบบขนานและแบบอนุกรม

ดังนั้นการสื่อสารข้อมูลแบบอนุกรมจึงถูกนำมาใช้ ในการสื่อสาร โดยจะใช้สายเพียงเส้นเดียวในการส่งข้อมูล หรือรับข้อมูล (คำว่าเส้นเดียวหมายความว่าสายส่ง(TxD) 1 เส้น สายรับ(RxD) 1 เส้น และสายกราวด์ร่วม(Ground) 1 เส้น) นำมาใช้สื่อสารข้อมูลกับอุปกรณ์ภายนอกในระยะทางที่ไกล ดังในรูป 1ข ถ้าหากต้องการส่งข้อมูลขนาด 8 บิต ก็จะทำการส่งข้อมูลออกไปทีละบิตเป็นลำดับไป จนกว่าจะครบจำนวนทั้ง 8 บิต ดังในรูป 1ค จะแสดงการเปลี่ยนข้อมูลแบบขนานให้เป็นแบบอนุกรม ข้อมูลจะถูกส่งไปตามสายสัญญาณที่ละบิต

ตามจังหวะเวลาที่กำหนด เป็นความกว้างของพัลส์ โดยจังหวะเวลาที่กล่าวนี้จะต้องมีมาตรฐาน ของฝ่ายส่ง และฝ่ายรับด้วย ในการรับสัญญาณที่ส่งมาทีละบิต จะทำการตรวจสอบระดับแรงดันของสัญญาณที่เข้ามาเพื่อแปลงเป็นลอจิก "1" หรือ "0" เมื่อรับข้อมูลเข้ามาครบใน 1 ไบต์ที่กำหนดไว้ ก็จะถูกเปลี่ยนให้อยู่ในรูปแบบของข้อมูลแบบขนานเหมือนเดิม

จังหวะเวลาของการสื่อสารข้อมูลอนุกรม

ในการสื่อสารข้อมูลแบบอนุกรม เพื่อรับหรือส่งข้อมูล จะเป็นลักษณะของกลุ่มข้อมูล ดังนั้นอัตราความเร็วจะต้องมีค่าเท่ากันระหว่างการรับและการส่งโดยทั่วไปเรา จะระบุความเร็วของจำนวนบิตในการรับและส่งข้อมูล เป็นจำนวนของบิตที่จะส่งใน 1 วินาที โดยเรียกความเร็วในการส่งข้อมูลว่า อัตราบอด(Baud Rate) ซึ่งมีหน่วยเป็นบิตต่อวินาที เช่น 300, 1,200, 2,400, 4,800 และ 9,600 บิตต่อวินาที ในรูป 12 ถ้าหากมีการส่งข้อมูลด้วยความเร็ว 9600 บิตต่อวินาที จะใช้เวลาในการรับส่งข้อมูลหนึ่งบิตมีค่าเท่ากับ 1/9600 หรือ 104.1 ไมโครวินาที และเวลาในการรับส่งข้อมูลทั้ง 8 บิตจะมีค่าเท่ากับ 8 x 104.1 หรือ832.8 ไมโครวินาที



รูปที่ 2.44 แสดงการส่งข้อมูลแบบอนุกรมด้วยความเร็ว 9600 บิตต่อวินาที

รูปแบบของการสื่อสารข้อมูลอนุกรม

การสื่อสารข้อมูลอนุกรมแบบอะซิงโคนัส เป็นวิธีการรับและส่งข้อมูลโดยไม่ต้องอาศัยสัญญาณนาฬิกาส่งร่วมไปด้วย แต่จะใช้อัตราความเร็วของจำนวนข้อมูลต่อวินาที และจะทำการเพิ่มบิตข้อมูลบางอย่างร่วมไปกับการส่งข้อมูลจริง เพื่อจะได้ทำการตรวจสอบข้อมูลได้อย่างถูกต้องมากยิ่งขึ้นแสดงดังรูปที่ 13 ซึ่งประกอบด้วยกัน 4 ส่วนคือ

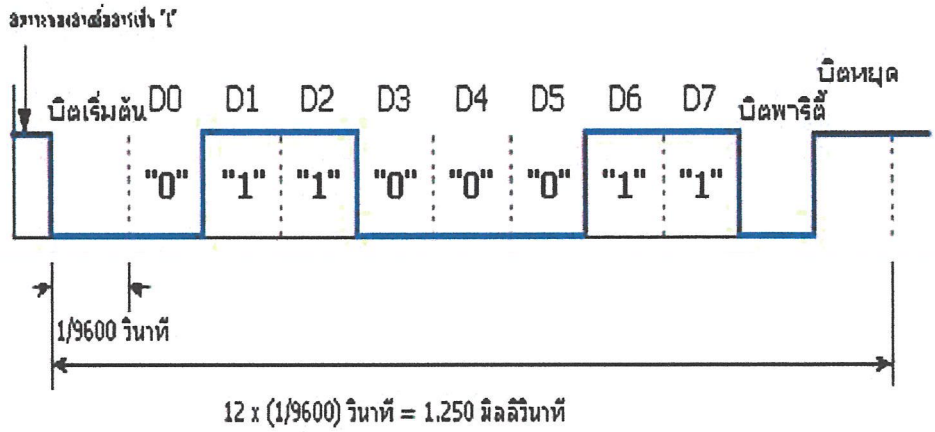
1 บิตเริ่มต้น (Start bit) จะมีขนาด 1 บิต จะเป็นละดับลอจิกตรงกันข้ามกับระดับลอจิกของสภาวะสายสื่อสาร ขณะที่ยังไม่มีการส่งข้อมูล

2 บิตข้อมูล (Data bit) จะเริ่มจากบิตที่มีนัยสำคัญต่ำสุดก่อนหรือ บิต LSB ก่อน โดย

ข้อมูลที่จะส่งอาจจะมีขนาด 5 ,6, 7 หรือ 8 บิตก็ได้

3 บิตแสดงสภาวะเลขคู่หรือเลขคี่ (Parity bit) มีขนาด 1 บิตโดยบิตนี้จะนำไปต่อท้ายกับบิตข้อมูล ค่าของบิตนี้ขึ้นอยู่กับจำนวนค่าของข้อมูลที่เป็น "1" โดยเลือกการส่งข้อมูลเป็นแบบ พาริตี้คู่ หรือ พาริตี้คี่ ตัวอย่าง ถ้ากำหนดให้มีการส่งข้อมูลแบบพาริตี้คู่ แต่ข้อมูลมีเลข 1 เป็นจำนวนคี่ ก็จะให้บิตพาริตี้นี้เป็น "1"เพื่อจะได้จำนวนเลข "1" เป็นคู่นั้นเอง ทำนองเดียวกันทางด้านรับเองก็ต้องมีการตรวจสอบจำนวนข้อมูลที่รับเข้ามาเป็น "1" รวมทั้งบิตพาริตี้ 1 บิต ถ้ามีค่า "1"เป็นจำนวนคู่ แสดงว่าข้อมูลที่รับเข้ามาถูกต้อง * สามารถกำหนดการรับและส่งข้อมูลเป็นแบบ NONE โดยไม่ต้องมีการตรวจสอบพาริตี้บิตก็ได้4 บิตสุดท้ายหรือบิตหยุด (Stop bit) เป็นการระบุถึงขอบเขตของการสิ้นสุดข้อมูล โดยจะทำให้ขาข้อมูลมีสถานะ ลอจิกเป็น "1" ซึ่งอาจมีจำนวนมากกว่า หนึ่งบิตก็ได้ เช่น 1 บิต 1.5 บิต หรือ 2 บิต

D7	D6	D5	D4	D3	D2	D1	D0
1	1	0	0	0	1	1	0



รูปที่ 2.45 แสดงการส่งข้อมูลขนาด 8 บิตแบบอนุกรมพร้อมด้วย บิตเริ่มต้น, บิตพาริตี้, บิตหยุด ด้วยความเร็ว 9600 บิตต่อวินาที

โครงสร้างของไมโครคอนโทรลเลอร์ PSoC

ระบบไมโครคอนโทรลเลอร์เดิม ซึ่งสามารถรองรับการทำงานในรูปแบบเฉพาะสัญญาณทางดิจิตอล จึงมีการพัฒนาชิพไมโครคอนโทรลเลอร์ขึ้นเพื่อลดปัญหาและข้อจำกัดของระบบไมโครคอนโทรลเลอร์แบบเดิมตามคอนเซ็ปที่ว่า PSoC หรือ Programmable System On Chip ซึ่งรวมเอาการทำงานทางด้านอนาล็อกเข้ามาภายในชิพเดียวจึงถือว่าเป็นประโยชน์ต่อการพัฒนา และลดความยุ่งยากในการจัดทำวงจรอินเตอร์เฟสเพิ่มเติม

คุณสมบัติสำคัญของ PSoC

1) มีการสร้างระบบภายในแบบ Harvard Architecture ด้วยหน่วยประมวลผลแบบ M8C และสามารถทำงานได้ที่ความถี่สูงถึง 24 MHz

2) มีวงจรคูณเลขภายในแบบ 8X8 Multiply (32 Bit Accumulate)

3) สามาทำงานแรงดันไฟต่ำได้ 3 – 5 โวลต์

4) มีโหมดการทำงานแบบ Switch Mode Pump (SMP) ซึ่งช่วยให้ระบบทำงานในสภาวะแรงดันที่ต่ำถึง 1 โวลต์

5) ทำงานในช่วงอุณหภูมิ -40 ถึง 85 องศาเซลเซียส

6) วงจรกำเนิดสัญญาณนาฬิกาภายในที่มีความเที่ยงตรงสูง เท่ากับ 24/48 MHz และยังทำงานร่วมกับ External Oscillator ได้ที่ความถี่สูงถึง 24 MHz

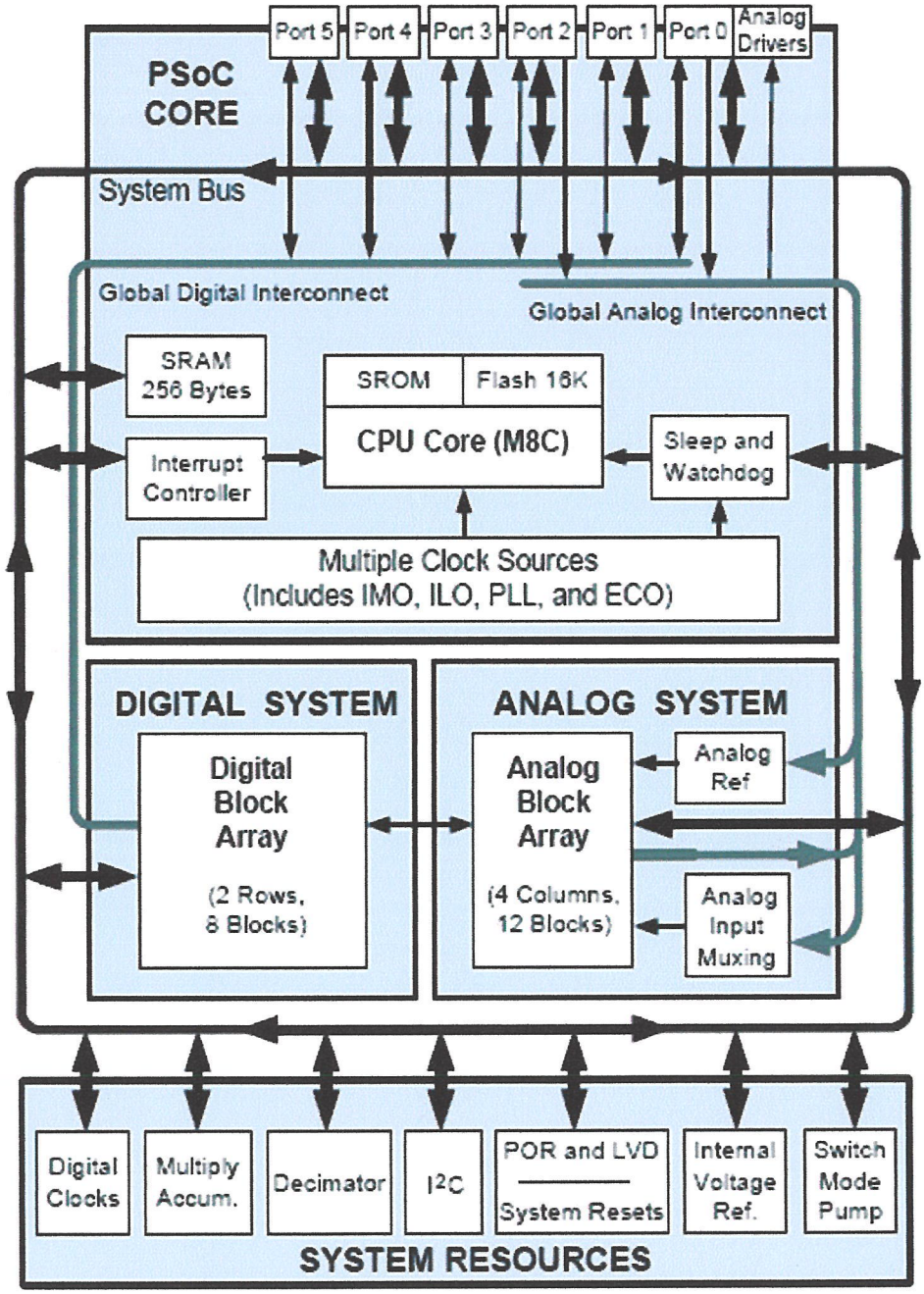
7) มีหน่วยความจำภายในที่ยืดหยุ่นสูง

8) สามารถโปรแกรมฟังก์ชั่นการทำงานให้กับขาต่างๆของไมโครคอนโทรลเลอร์ได้ และสามารถขับกระแสได้ 25 mA ทุกขาในโหมด GPIO

9) และมีทรัพยากรเพิ่มเติมที่มีอยู่ภายในต่างๆ เช่น I2C Slave Master Watchdog sleep timer และมีวงจรกำเนิดแรงดันอ้างอิงภายในที่มีความเที่ยงตรงสูง

10) มีซอฟต์แวร์สำหรับใช้ในการพัฒนาการใช้งานได้ทั้ง C และ Assembly

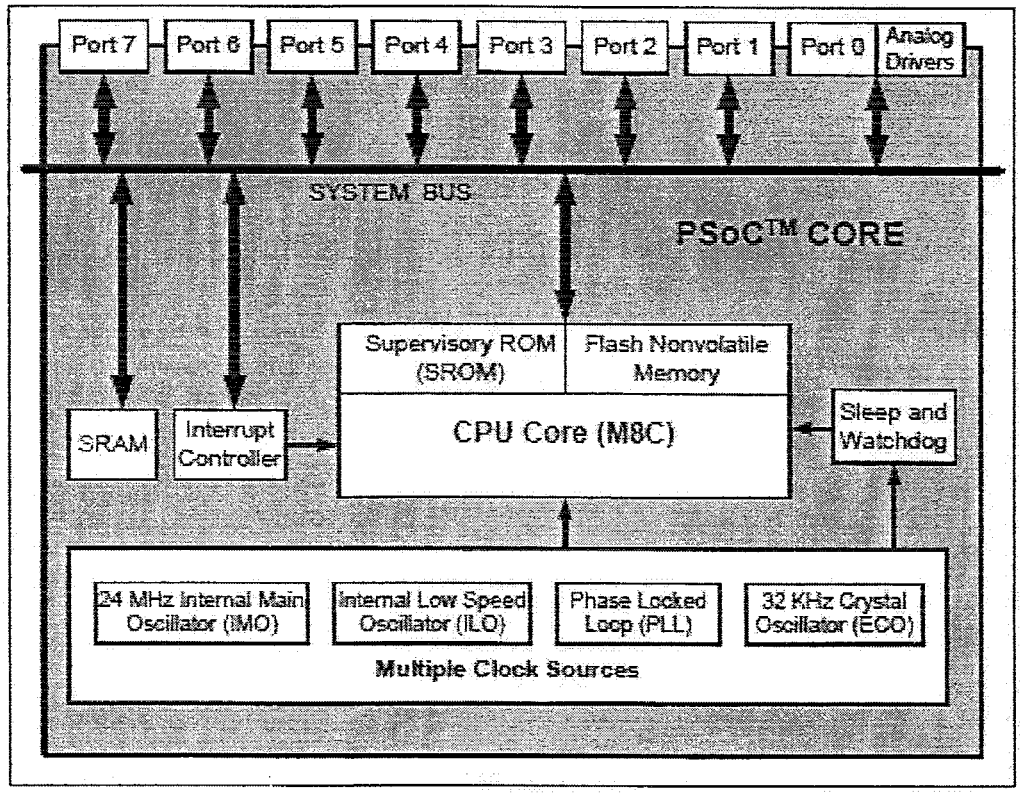
การศึกษาและใช้งานไมโครคอนโทรลเลอร์ให้เกิดประโยชน์และประสิทธิภาพสูงสุด ผู้ใช้จะต้องควรทราบถึงองค์ประกอบและความสามารถภายในตัวชิพ เพื่อสามารถนำไปประยุกต์ใช้งานได้อย่างถูกต้องและเหมาะสม สำหรับ PSoC มีรูปแบโครงสร้างของระบบภายในดังรูป 2.17



รูปที่ 2.46 บล็อกไดอะแกรมไมโครคอนโทรลเลอร์ PSoC

PSoC Core

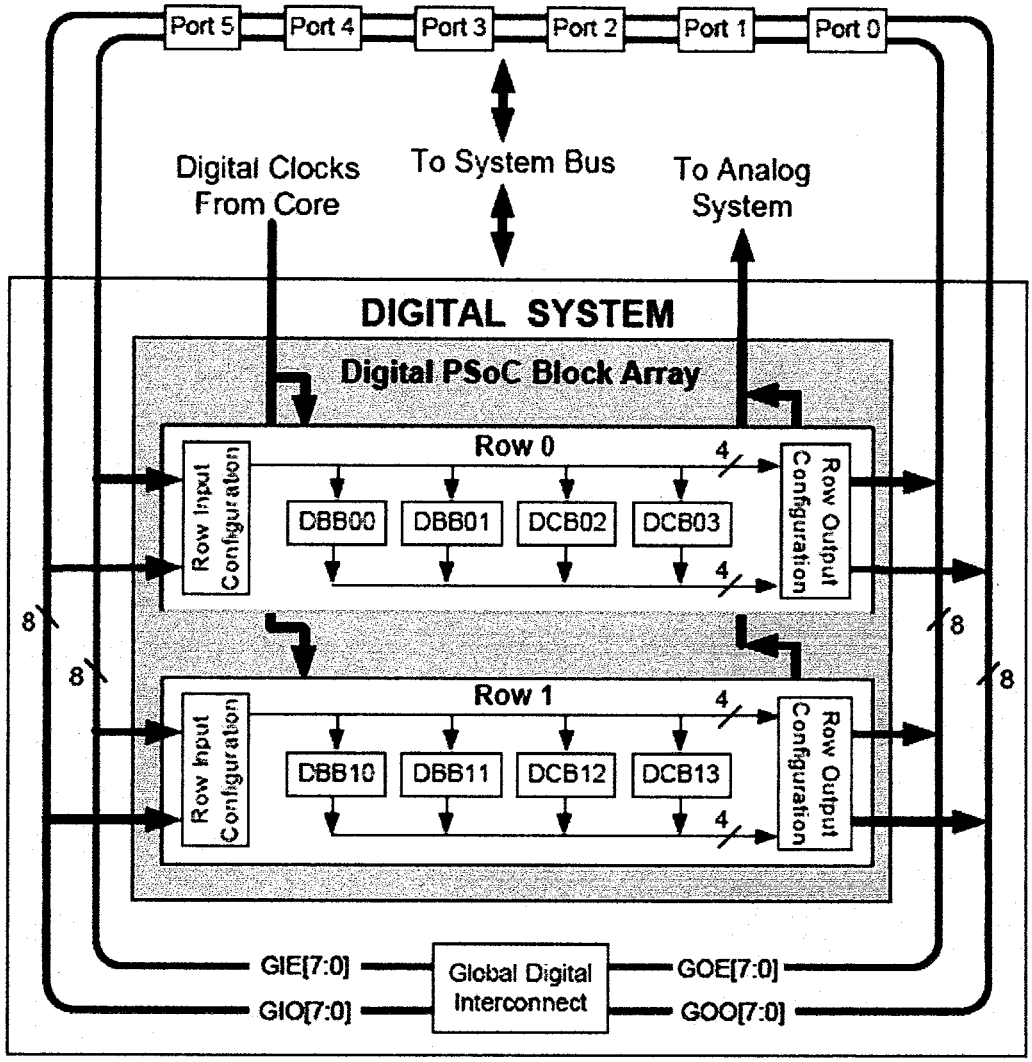
เป็นส่วนของแกนหลักในการประมวลผลและควบคุมการทำงานภายในทั้งหมด อันประกอบด้วย หน่วยประมวลผลแบบ M8C



รูปที่ 2.47 PSoC Core

Digital System

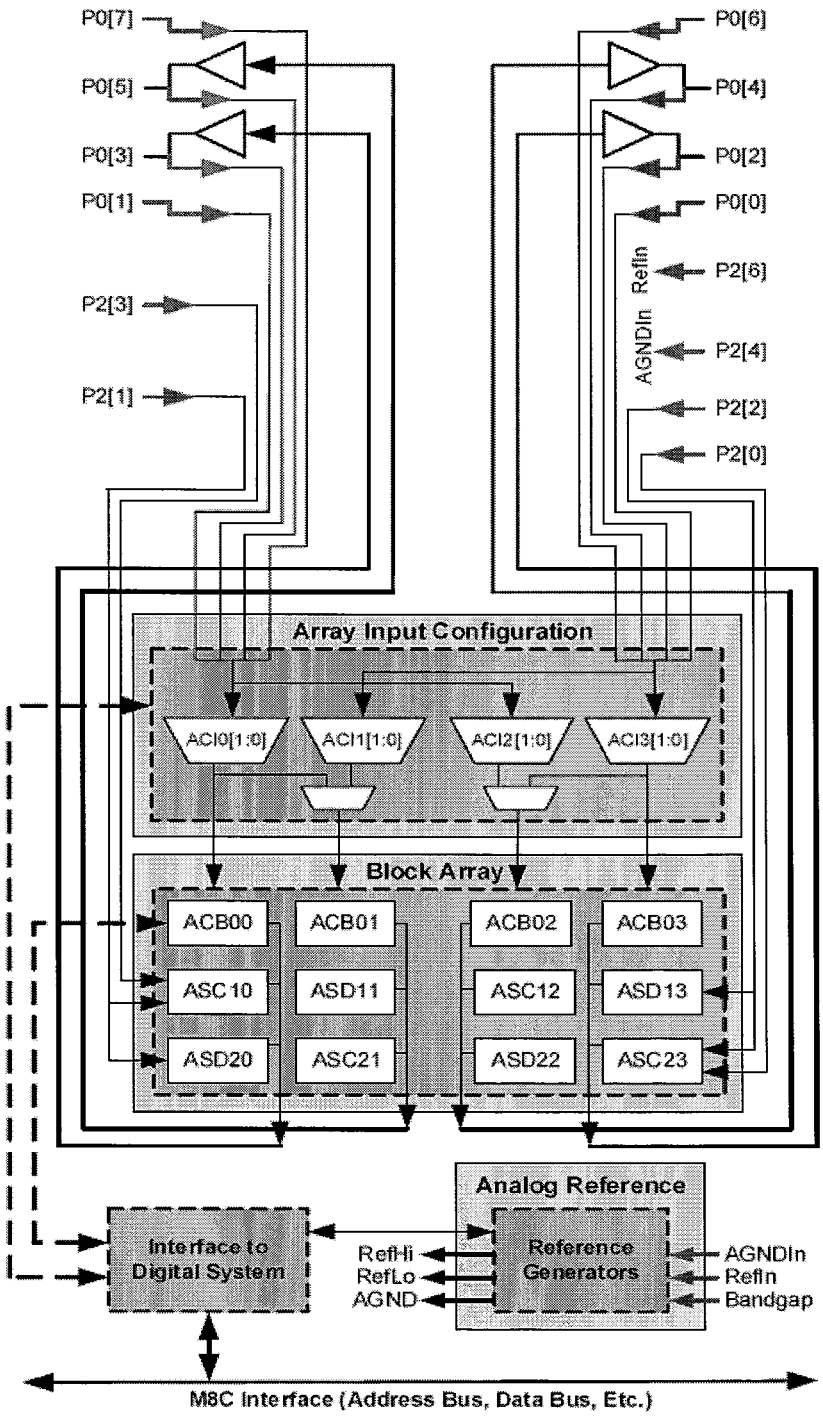
เป็นพื้นที่การทำงานของระบบดิจิตอลโดยเป็นส่วนการทำงานทางด้าน Hardware ที แยกเป็นอิสระจาก PCoS Core โครงสร้างส่วนนี้เองที่ผู้สร้างสามารถกำหนดคุณสมบัติทางด้านดิจิตอลลงบนชิพเองได้ เช่น Timer Counter PWM I2C และ UART เป็นต้นเพื่อให้ชิพมีคุณสมบัติทางดิจิตอลตามที่ต้องการ สำหรับชิพเบอร์ CY29666 มีให้ใช้งานได้ 16 Digital Block และแต่ละบล็อกมีข้อมูลขนาด 8 บิท



รูปที่ 2.48 Digital Systems

Analog System

เป็นพื้นที่การทำงานของระบบอนาล็อกโดยเป็นส่วนการทำงานทางด้าน Hardware ที แยกเป็นอิสระจาก PCoS Core และ Digital system โดยโครงสร้างส่วนนี้เองที่ผู้สร้างสามารถกำหนดคุณสมบัติทางด้านอนาล็อกลงบนชิพเองได้ เช่น Amplifier ADC DAC เป็นต้น สำหรับชิพเบอร์ CY29666 มีให้ใช้งานได้ 12 Analog Block



รูปที่ 2.49 Analog Systems

System Resource

เป็นส่วนของทรัพยากรรวมภายใน ซึ่งละส่วนของระบบไมโครคอนโทรลเลอร์สามารถติดต่อถึงกันได้ผ่านซิสเต็มบัส (System Bus) อันประกอบด้วย

Digital Clocks สำหรับควบคุมการหารความถี่สัญญาณนาฬิกา

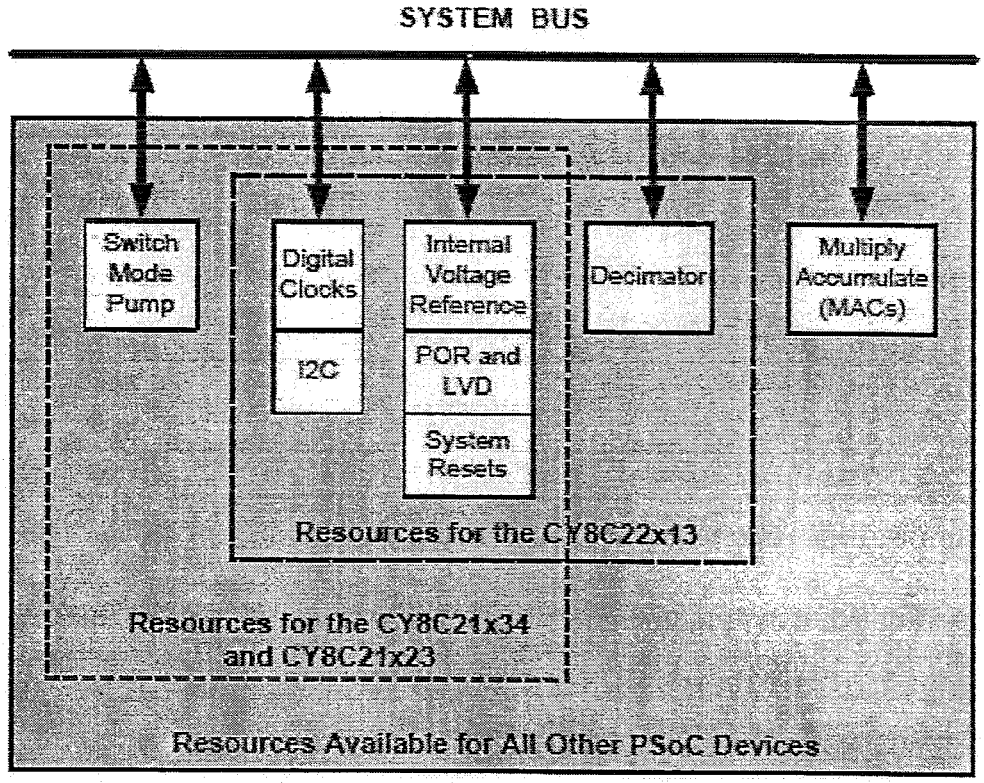
Multiply Accumulate (MAC)

Decimator

I2C สำหรับการสื่อสารด้วยรูปแบบ I2C

POR and LVD สำหรับควบคุมระบบ Reset และระบบตรวจสอบแรงดันไฟเลี้ยงต่ำกว่ากำหนด

Internal Voltage Reference แรงดันอ้างอิงภายในสามารถกำหนดเป็นแรงดันอ้างอิงให้แก่ ADC หรือส่งค่าแรงดันอ้างอิงออกสู่ขาสัญญาณเพื่อนำออกไปใช้งานภายนอกได้ Switch Mode Pump เป็นโหมดการทำงานเพื่อบูทแรงดันไฟเลี้ยงที่ต่ำให้มีแรงดันที่สูงขึ้นและเพียงพอสำหรับการทำงานของระบบไมโครคอนโทรลเลอร์ที่ประยุกต์ใช้กับแบตเตอรี่



รูปที่ 2.50 System Resources

PORT

เป็นขาสัญญาณต่างๆ สำหรับการอินเตอร์เฟสไปยังวงจรต่างๆ โดยจำนวนของพอร์ต จะขึ้นอยู่กับเบอร์ของชิพ สำหรับเบอร์ CY29666 ที่ใช้จะมี 48 ขาให้ได้เลือกใช้งาน ซึ่งขาสัญญาณของ PSoC มีลักษณะคล้ายกับไมโครคอนโทรลเลอร์เบอร์อื่นๆ คือมีทั้งขาสัญญาณอินพุต เอาต์พุต ซึ่งในบางขาอาจจะทำหน้าที่มากกว่าหนึ่งหน้าที่ หน้าที่การทำงานของขาสัญญาณต่างๆของ PSoC สามารถสรุปได้ดังตาราง

ตารางที่ 2.15 แสดงหน้าที่การทำงานของขาต่างๆ

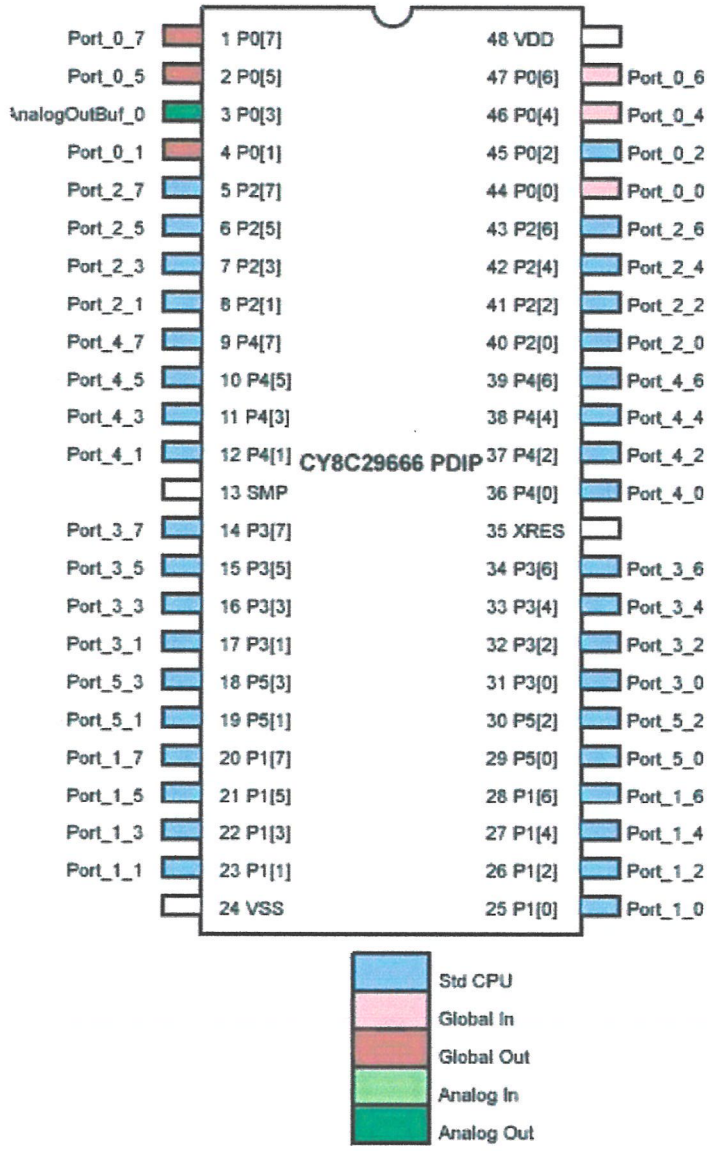
Table 1-3. 48-Pin Part Pinout (SSOP)

Pin No.	Type		Pin Name	Description
	Digital	Analog		
1	IO	I	P0[7]	Analog column mux input.
2	IO	IO	P0[5]	Analog column mux input and column output.
3	IO	IO	P0[3]	Analog column mux input and column output.
4	IO	I	P0[1]	Analog column mux input.
5	IO		P2[7]	
6	IO		P2[5]	
7	IO	I	P2[3]	Direct switched capacitor block input.
8	IO	I	P2[1]	Direct switched capacitor block input.
9	IO		P4[7]	
10	IO		P4[5]	
11	IO		P4[3]	
12	IO		P4[1]	
13	Power		SMP	Switch Mode Pump (SMP) connection to external components required.
14	IO		P3[7]	
15	IO		P3[5]	
16	IO		P3[3]	
17	IO		P3[1]	
18	IO		P5[3]	
19	IO		P5[1]	
20	IO		P1[7]	I2C Serial Clock (SCL)
21	IO		P1[5]	I2C Serial Data (SDA)
22	IO		P1[3]	
23	IO		P1[1]	Crystal (XTALin), I2C Serial Clock (SCL)
24	Power		Vss	Ground connection.
25	IO		P1[0]	Crystal (XTALout), I2C Serial Data (SDA)
26	IO		P1[2]	
27	IO		P1[4]	Optional External Clock Input (EXTCLK)
28	IO		P1[6]	
29	IO		P5[0]	
30	IO		P5[2]	
31	IO		P3[0]	
32	IO		P3[2]	
33	IO		P3[4]	
34	IO		P3[6]	
35	Input		XRES	Active high external reset with internal pull down.
36	IO		P4[0]	
37	IO		P4[2]	
38	IO		P4[4]	
39	IO		P4[6]	
40	IO	I	P2[0]	Direct switched capacitor block input.
41	IO	I	P2[2]	Direct switched capacitor block input.
42	IO		P2[4]	External Analog Ground (AGND)
43	IO		P2[6]	External Voltage Reference (VREF)
44	IO	I	P0[0]	Analog column mux input.
45	IO	IO	P0[2]	Analog column mux input and column output.
46	IO	IO	P0[4]	Analog column mux input and column output.
47	IO	I	P0[6]	Analog column mux input.
48	Power		Vdd	Supply voltage.

LEGEND: A = Analog, I = Input, and O = Output.

นอกจากการใช้งานของขาพอร์ตต่างๆเป็นพอร์ตอินพุต/เอาต์พุตทั่วไปแล้ว ขาพอร์ตของขายังมีหน้าที่เฉพาะอย่างดังต่อไปนี้

- VDD เป็นขาสัญญาณไฟเลี้ยง ต่อกับไฟ 5โวลต์
- VSS เป็นขากราวด์ ต่อกับไฟเลี้ยง 0 โวลต์
- XRES เป็นขาสำหรับรีเซต เมื่อนี้มีลอจิกเป็น "1" CPU จะถูกรีเซต
- P0[2]-P0[5] เป็นสำหรับรับสัญญาณทางอนาล็อกเข้าภายในเพื่อทำการประมวลผล นอกจากนี้แล้วยังสามารถส่งสัญญาณอนาล็อกออกไปทางขาเหล่านี้ได้อีกด้วย
- P0[6]-P0[7] เป็นขาสำหรับรับสัญญาณทางอนาล็อกเข้าภายในเพื่อทำการประมวลผล แต่ไม่สามารถส่งสัญญาณอนาล็อกออกไปทางขาเหล่านี้ได้
- P0[0] เป็นขา XTAL out ใช้สำหรับต่อกับ XTAL เพื่อสร้างสัญญาณให้กับ PSoC (ใช้งานร่วมกับ P0[1])
- P0[1] เป็นขา XTAL out ใช้สำหรับต่อกับ XTAL เพื่อสร้างสัญญาณให้กับ PSoC (ใช้งานร่วมกับ P0[0])
- P1[4] เป็นขาสำหรับรับสัญญาณจากภายนอก
- P1[5] เป็นขารับ/ส่งข้อมูลของ I2C ซึ่งจะเรียกว่าขา SDA (Serial Data)
- P1[7] เป็นขารับสัญญาณนาฬิกาในการรับ/ส่งของมูล I2C เพื่อให้ด้านส่งและด้านรับทำการรับข้อมูลได้อย่างสอดคล้องกัน ซึ่งเรียกว่า SCL (Serial Clock)
- P2[0]- P2[3] เป็นขารับสัญญาณอนาล็อกแบบ Non – Multiplexed
- P2[6] เป็นขารับสัญญาณอ้างอิงจากภายนอก



รูปที่ 2.51 PSoC เบอร์ CY8C29666

2.2.7 Introduction To Lab VIEW

Lab VIEW เป็นโปรแกรมที่ผลิตขึ้นมาเพื่อใช้ประโยชน์ในด้านการวัดและเครื่องมือวัด สำหรับงานทางวิศวกรรม Lab VIEW ย่อมาจาก Laboratory Virtual Instrument Engineering Workbench ซึ่งหมายความว่าเป็นโปรแกรมที่สร้างเครื่องมือวัดเสมือนจริงในห้องปฏิบัติการทางวิศวกรรม ดังนั้นจุดประสงค์หลักของการทำงานของโปรแกรมนี้คือการจัดการในด้านการวัด และเครื่องมือวัดอย่างมีประสิทธิภาพ และในตัวของโปแกรมจะประกอบไปด้วยฟังก์ชันที่ใช้ช่วยในการวัดมากมาย และแน่นอนที่สุดโปรแกรมนี้จะมีประโยชน์อย่างสูงเมื่อใช้ร่วมกับเครื่องมือวัดทางวิศวกรรมต่าง ๆ

Lab VIEW เป็นโปรแกรมที่ใช้รูปภาพหรือสัญลักษณ์แทนการเขียนด้วยตัวอักษรเหมือนโปรแกรมปกติทั่วไป ซึ่งข้อดีข้อแรกก็คือการลดความผิดพลาดด้านการสะกดผิดหรือพิมพ์ผิดออกไป ข้อแตกต่างอีกประการหนึ่งที่สำคัญของการเขียนโปรแกรมแบบ G กับการเขียนด้วยตัวหนังสือก็คือ การเขียนด้วยภาษา G นี้เป็นการเขียนโดยใช้หลักการของ Data Flow ซึ่งเมื่อเริ่มส่งข้อมูลเข้าสู่โปรแกรม เราจะต้องกำหนดทิศทางไหลของข้อมูลว่าจะไปที่ส่วนใด ผ่านการประเมินผลและคำนวณในส่วนใดบ้าง และจะให้แสดงผลอย่างไร ซึ่งลักษณะการเขียนภาษา G หรือ Data Flow นี้จะมีลักษณะเหมือนกับการเขียน Block Diagram ซึ่งทำให้ผู้เขียนโปรแกรมสามารถให้ความสนใจกับการเคลื่อนที่และเปลี่ยนแปลงข้อมูลได้โดยไม่ต้องจดจำรูปแบบคำสั่งที่ยุ่งยาก

เนื่องจาก Lab VIEW ใช้ลักษณะการเขียนแบบ Block Diagram ซึ่งวิศวกรส่วนใหญ่มีความคุ้นเคยอยู่แล้ว จึงเป็นการง่ายที่จะทำความเข้าใจและนำไปพัฒนาใช้ต่อไปได้ และถ้าหากเราจำได้ถึงขั้นตอนการเขียนโปรแกรมว่าก่อนที่จะเขียนโปรแกรม เราควรจะต้องเขียน Flow Chart ให้เสร็จสิ้นเสียก่อน หลักจากตรวจสอบ Flow Chart เรียบร้อยแล้วเราจึงนำไปเขียนโปรแกรม ดังนั้นเราจะมีความสะดวกมากขึ้นถ้าหากการเขียน Flow Chart ของ Lab VIEW ก็คือการเขียนโปรแกรมนั่นเอง ซึ่งเป็นการลดขั้นตอนการทำงานลงไปได้เป็นจำนวนมาก

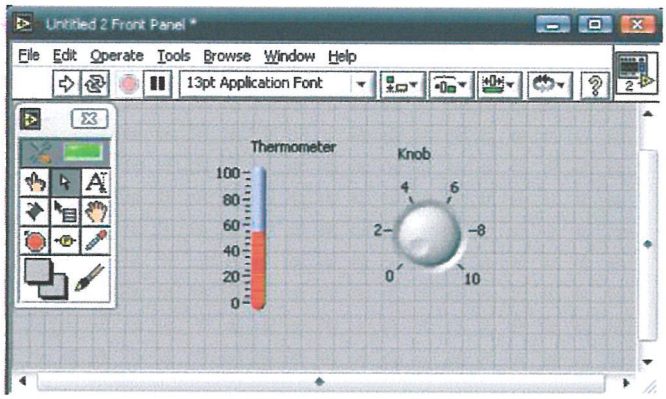
แม้ว่าการเขียนโปรแกรมใน Lab VIEW ไม่จำเป็นต้องมีความรู้ด้านการเขียนโปรแกรมใด ๆ มาก่อนเลย แต่การมีความรู้ด้านการเขียนโปรแกรมหรือใช้โปรแกรมสำเร็จรูปอื่น ๆ จะสามารถนำมาใช้ประโยชน์ได้เป็นอย่างดี

ส่วนประกอบต่างๆของ Lab VIEW

1. Front Panels and Block Diagrams

Front Panel หรือ หน้าปัทม์คือส่วนที่ผู้ใช้จะใช้ติดต่อกับโปรแกรม ในขณะที่เราให้ VI ทำงานอยู่นั้นหน้าปัทม์นี้จะต้องทำงานร่วมอยู่ด้วย เพื่อให้ผู้ใช้หรือผู้ควบคุมสามารถให้ข้อมูลเข้าสู่

โปรแกรม และเมื่อข้อมูลได้รับการประมวลผลแล้วก็จะแสดงออกมาทาง Front Panel นี้ ดังนั้นหากจะเปรียบกับโปรแกรมสำเร็จรูปอื่นๆและ Front Panel นี้ก็คือ Graphic User Interface (GUI) ของ Lab VIEW นั่นเอง ตัวอย่างของลักษณะของ Front Panel ใน Lab VIEW เป็นไปตามรูป ซึ่งในขั้นแรกนี้ผู้ที่ยังไม่มีความคุ้นเคยกับโปรแกรมนี้อาจมองดูว่าการสร้างองค์ประกอบต่างๆคงจะมีความยุ่งยาก แต่ถ้าหากเราเริ่มทำความเข้าใจกับ Lab VIEW แล้วเราจะพบว่าการเขียน Front Panel ในลักษณะในรูปนี้ไม่ใช่สิ่งที่ยุ่งยากหรือสิ้นเปลืองเวลาในการเขียนเลย

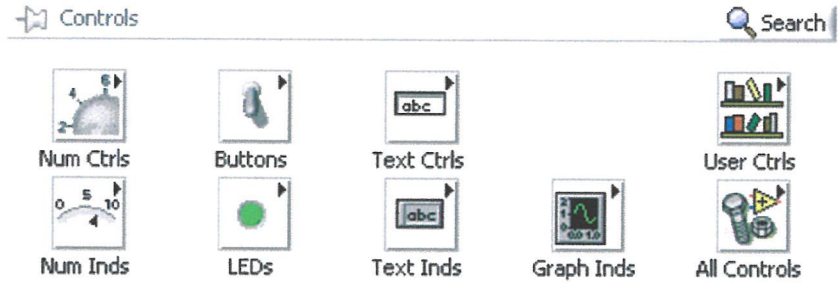


รูปที่ 2.52 แสดงรูปหน้าต่าง Front Panel

หากเราสังเกตจากรูปนี้เราจะพบว่าบนหน้าปัทม์ Front Panel ของ Lab VIEW จะมีส่วนประกอบที่สำคัญ 2 แบบ คือ ตัวควบคุม (Controlled) และ ตัวแสดงผล (Indicator) ซึ่งส่วนประกอบทั้ง 2 จะมีการทำงานต่างกันและหน้าที่ตรงกันข้ามกัน ดังมีรายละเอียดต่อไปนี้

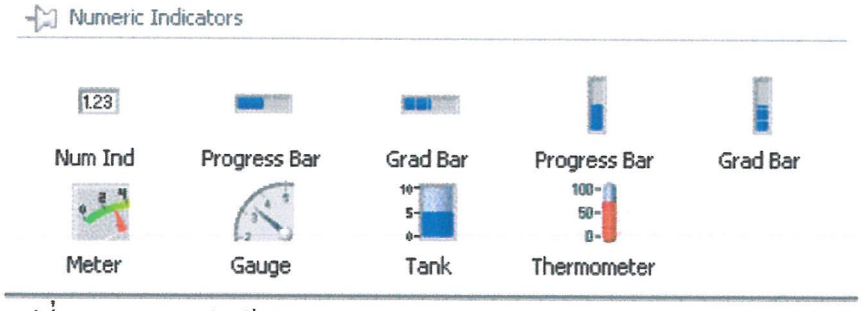
- **Controls** มีหน้าที่เป็นตัวควบคุมคือให้ค่าหรือ Input จากผู้ใช้ ลักษณะของ Controls เช่น ปุ่มปรับค่า, สะพานปิด – เปิดไฟ, แท่งเลื่อนเพื่อปรับค่า, การให้ค่าด้วยตัวเลข Digital หรืออื่นๆ ดังนั้นจากหลักการของ Controls ก็หมายความว่าเป็นการกำหนดค่าหรือแหล่ง (source) ของข้อมูล โดยปกติเราจะไม่สามารถนำข้อมูลมาแสดงผลที่ Controls ได้ และถ้าหากเราพยายามที่จะให้ Control แสดงผลข้อมูลก็จะเกิดความผิดพลาดขึ้นใน VI ของเราทันที

ตัวอย่างของ Object ที่ปกติแล้วจะทำหน้าที่เป็น Controls บน Front Panel บางประเภท จะแสดงในรูปต่อไปนี้ เราจะสังเกตเห็นว่าหากเปรียบเทียบกับในอุปกรณ์เครื่องมือวัดจริงแล้ว อุปกรณ์เหล่านี้จะได้รับการกำหนดค่าจากผู้ใช้ ดังนั้นจะเห็นว่า Lab VIEW พยามทำให้เราได้รู้สึกว่าใช้งานกับเครื่องมือจริงๆอยู่



รูปที่ 2.53 แสดงรูปหน้าต่าง Controls

- **Indicator** มีหน้าที่เป็นตัวแสดงผลเพียงอย่างเดียวโดยจะรับค่าที่ได้จากแหล่งข้อมูลมาแสดงผลซึ่งอาจปรากฏในรูปของกราฟ, เข็มชี้, ระดับของเหลว หรืออื่นๆ Indicator นี้เปรียบเสมือน output เพื่อให้ผู้ใช้ได้ทราบค่าสิ่งที่เรากำลังวิเคราะห์อยู่ และผู้ใช้ไม่สามารถปรับค่าบน indicator ได้ โดยตรงแต่จะต้องมีแหล่งข้อมูลที่ส่งให้กับ Indicators เหล่านี้ ดังนั้นเราอาจมอง Indicator ว่าเป็นเหมือน Sink ของข้อมูล ตัวอย่างของ object ที่ปกติแล้วจะมีเป็น Indicator บางชนิดได้แสดงในรูปต่อไปนี้

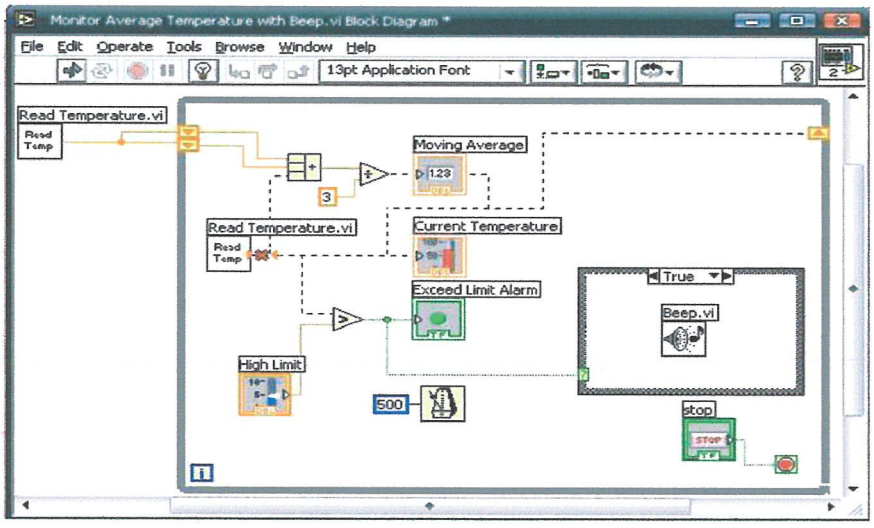


รูปที่ 2.54 แสดงรูปหน้าต่าง Indicators

ในการเขียน VI อันดับแรกคือการเขียนหน้าปัทม์ซึ่งผู้ใช้จะต้องออกแบบส่วนนี้จะจัดวางให้เหมาะสม ซึ่งจะกล่าวถึงในบทต่อไป สำหรับในขั้นนี้เราเพียงแต่เน้นว่าบนหน้าปัทม์ Front Panel จะประกอบด้วยสองส่วนและการที่เราจะเลือก Controls และ Indicators เป็นเรื่องสำคัญเพราะทั้งสองนี้ไม่สามารถแทนกันได้ นั่นคือเราไม่สามารถกำหนดค่าให้ Controls แสดงค่าได้และไม่สามารถนำค่าจาก Indicators ออกไปเป็นข้อมูลของระบบได้

Block Diagrams

ที่ผ่านมาเราจะพบว่าเราสามารถสร้าง Front Panel ได้ให้เป็นไปตามต้องการของเรา ซึ่งไม่ใช่สิ่งที่ยากมากนักสำหรับ Lab VIEW แต่สิ่งที่จะยุ่งยากมากกว่าคือการกำหนดให้สิ่งต่างๆ หรือที่เราเรียกว่า Object นั้นให้มีขั้นตอนหรือมีกระบวนการของการวิเคราะห์ต่างๆ ตามที่เราต้องการ เพราะเราจะต้องกำหนดการทำงานที่เกิดขึ้นหลังฉาก Front Panel เหล่านั้น นั่นคือหลังจากการที่เราออกแบบ GUI เรียบร้อยแล้วขั้นต่อไปก็คือการกำหนดการทำงานของ GUI เหล่านั้นนั่นเอง และส่วนที่มีหน้าที่นั้นคือ Block Diagram ในเบื้องต้นนี้เราอาจมอง Block Diagram ว่าเป็น Data Flow Chart และตัวโปรแกรมหรือ code ของ Lab VIEW ก็ได้ การเขียน Block Diagram ก็คือการเขียน code ในภาษา G นั่นเอง ซึ่งก็เหมือนกับการเขียน code ในภาษา C หรือ FORTRAN นั่นเอง ความแตกต่างที่สำคัญระหว่าง code ในภาษา C หรือ FORTRAN กับ Block Diagram ใน Lab VIEW ก็คือ Block Diagram นั้นพร้อมที่จะ execute หรือทำการประมวลผลตลอดเวลา นั่นคือในระหว่างที่เราสร้าง Block Diagram อยู่ Lab VIEW จะตรวจสอบการทำงานของ VI อยู่ตลอดเวลา ตัวอย่างของ Block Diagram เป็นไปตามรูป



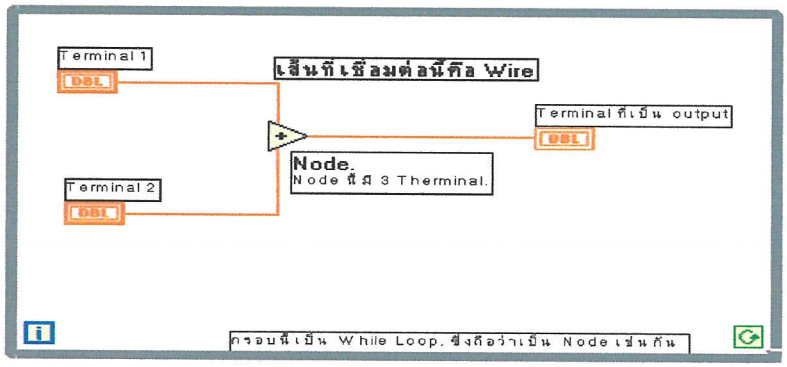
รูปที่ 2.55 รูปภายใน Block Diagrams

ถ้าหากเราพิจารณาองค์ประกอบใน Block Diagram เราจะพบว่าในส่วนของ Block Diagram จะมีส่วนประกอบที่สำคัญ 3 ส่วน คือ Terminal, Node และWire ทั้งสามส่วนจะมีหน้าที่หลักคือการควบคุมการส่งผ่านหรือเราอาจเรียกว่าการไหลของข้อมูล (Data Flow) และกำหนดถึงวิธีการประมวลผลข้อมูล

- **Terminal** ทุกครั้งที่เราสร้าง Control หรือ Indicator บน Front Panel ใน Window ของ Block Diagram จะปรากฏ Terminal ขึ้น Terminal ก็คล้ายกับสถานีของข้อมูลคือจะเป็นทั้งสถานีต้นทางของข้อมูลถ้า Terminal นั้นเป็น Terminal ของ Controls และขณะเดียวกันจะเป็นทั้งสถานีปลายทางของข้อมูลถ้า Terminal นั้นเป็น Terminal ของ Indicator

ข้อที่ควรเข้าใจอย่างหนึ่งก็คือ Object นี้เกิดขึ้นจากการเขียนขึ้นบน Front Panel ดังนั้นเมื่อเราจะไม่สามารถลบ Terminal ออกจาก Block Diagram ได้ และถ้าหากเราจะลบ Control หรือ Indicator นั้นออกไปจาก Front Panel แล้ว Terminal เหล่านี้ก็จะหายไปจาก Block Diagram เช่นกัน สำหรับลักษณะของตัวแปรต่างๆเราจะกล่าวในบทต่อไป

- **Node** เป็นคำที่ใช้เรียก object ที่ทำกรรมวิธีใดๆ เพื่อประมวลข้อมูลใน Block Diagram เช่นเดียวกับที่เราเขียน Flow Chart แล้วใช้สัญลักษณ์ต่างๆแทนวิธีการวิเคราะห์ข้อมูล เมื่อมีข้อมูลเข้าสู่ node สิ่งที่เกิดขึ้นภายใน node ก็จะขึ้นอยู่กับว่าจะกำหนดให้ข้อมูลที่ส่งเข้าไปนั้น จะมีการประมวลผลอย่างไร ซึ่งอาจจะเป็นการบวก ลบ คูณ หาร หาราก ยกกำลัง หรือเป็นประเภทการเปรียบเทียบข้อมูล ว่ามากหรือน้อยกว่า หรืออื่นๆ ซึ่งจะเป็นการประมวลผลทางคณิตศาสตร์ทั่วไป นอกเหนือจากนั้น node นี้จะมีส่วนที่เรียกว่า Function แบบต่างๆ ซึ่งจะเหมือนกับ Function สำเร็จรูปเช่น sine, cosine, log เป็นต้น ซึ่งก็จะเหมือนกับในภาษาที่เป็นตัวอักษรทั่วๆ ไป



รูปที่ 2.56 รูปแสดงถึงลักษณะของ Node และ Terminal ที่บรรจุอยู่ภายใน Block Diagram ของ Lab VIEW

นอกเหนือจากการประมวลผลทางคณิตศาสตร์แล้ว เรายังมี node ประเภท Structure หรือ Control Flow อีกด้วย (ในภาษาตัวอักษร Structure Command จะเป็น คำสั่งจำพวก IF...THEN, FOR..., WHILE... เป็นต้น)

- **Wires** ขณะที่เรามีที่มาของข้อมูล ส่วนประมวลหรือปรับแต่งข้อมูล และส่วนแสดงผลข้อมูลเรียบร้อยแล้ว ขั้นต่อไปคือเราจะต้องสามารถควบคุมการส่งผ่านข้อมูลให้เป็นไปตามที่เราต้องการ อุปกรณ์ที่ใช้ใน Lab VIEW ก็คือ การต่อสาย หรือ Wires ซึ่งจะเป็นการเชื่อมการส่งข้อมูลระหว่าง terminal หรือ node ต่างๆที่มีใน Block Diagram นี้เข้าด้วยกัน โดย wires นี้จะเป็นการกำหนดเส้นทางของข้อมูลว่าเมื่อออกจาก terminal หนึ่งแล้ว จะกำหนดการไหลไปที่ node ใดบ้าง มีลำดับเป็นอย่างไร และสุดท้ายจะให้แสดงผลที่ terminal ใดนั่นเอง ซึ่งการเชื่อมต่อสายนี้จะทำให้เราเข้าใจถึงหลักการของ Data Flow Programming ได้ดีขึ้น

เนื่องจากข้อมูลนั้นมีหลายแบบไม่ว่าจะเป็นเลขทศนิยม, เลขจำนวนจริง, ตัวอักษร หรือค่าจริง-เท็จ (Boolean) ดังนั้นเพื่อแสดงถึงความแตกต่างของข้อมูลแต่ละแบบ Lab VIEW จึงได้กำหนดให้ลักษณะของ wires สำหรับข้อมูลแต่ละแบบมีลักษณะของเส้นและสีที่แตกต่างกัน นอกจากนั้นข้อมูลแต่ละแบบดังกล่าวยังอาจมีลักษณะเป็น scalars, 1-D array, 2-D array ได้ซึ่งลักษณะของเส้นของข้อมูลแต่ละแบบก็จะแตกต่างออกไปอีก

ตารางที่ 2.16 แสดงถึงลักษณะของเส้น wire เมื่อใช้เชื่อมต่อระหว่างข้อมูลแต่ละชนิด

	Scalars	1-D Array	2-D Array	สี
เลขทศนิยม				ส้ม
เลขจำนวนเต็ม				น้ำเงิน
Boolean				เขียว
ตัวอักษร				ชมพู

โดยปกติการต่อข้อมูลระหว่าง Node หรือ Terminal จะกำหนดแน่นอนว่าต้องการข้อมูลลักษณะใด เช่นถ้า Node ต้องการตัวเลข เราจำเป็นต้องต่อสายตัวเลขเข้ากับ Node นั้น ถ้าหากเราต่อสายจากตัวอักษรเข้าใน terminal ที่ต้องการตัวเลข สายต่อนั้นจะกลายเป็นสายต่อเสีย หรือ Bad Wire และโปรแกรมก็จะไม่สามารถทำงานต่อไปได้

เราจะเข้าใจหลักการของ Data Flow Programming ได้ในขณะที่เราต่อสายระหว่าง terminal และ Node ต่างๆ เข้าด้วยกัน ซึ่งหลักการทำงานของ Data Flow Programming จะต่างจากการเขียนโปรแกรมโดยใช้ตัวอักษร เพราะในโปรแกรมตัวอักษรคำสั่งจะถูกส่งเข้าสู่ส่วนประมวลผลทีละบรรทัด เพื่อการคำนวณตามลำดับบรรทัด แต่ใน Data Flow นั้น โปรแกรมจะคำนวณเมื่อมีข้อมูลส่งเข้ามาถึง Input ของ Node นั้นๆ ครบ และเมื่อข้อมูลส่งครบเข้าถึง Node นั้นแล้ว จะมีการประมวลผลและส่งค่าที่ได้ออกไปตามการต่อเชื่อมสาย การคำนวณจะเสร็จสิ้นในแต่ละรอบเมื่อข้อมูลส่งข้ามไปถึง Terminal สุดท้าย การประมวลผลไม่ได้เป็นไปตามลำดับการจัดวางคือไม่ได้ทำ

จากซ้ายไปขวาหรือบนลงล่าง แต่เป็นไปตามขั้นตอนการเดินทางของข้อมูลซึ่งในการเขียนโปรแกรมในเบื้องต้นผู้ที่คุ้นเคยกับภาษาตัวหนังสือโดยทั่วไปอาจจะต้องใช้เวลาสักระยะเพื่อที่จะทำความเข้าใจการทำงานของ Lab VIEW

Icon และ Connector

ถ้าเราคุ้นเคยกับการเขียนโปรแกรมเป็นตัวอักษร เราคงทราบว่าเราสามารถเขียนโปรแกรมย่อยขึ้นมาเพื่อใช้ร่วมกับโปรแกรมหลัก โดยการเขียนโปรแกรมย่อยหรือ Subroutine นี้จะมีประโยชน์มากในกรณีที่จะต้องทำการประมวลผลบ่อย ในภาษารูปภาพก็เช่นกันเราสามารถที่จะสร้าง Subroutine ขึ้นมาได้ ซึ่งเราจะเรียกว่า subVI โดย สำหรับข้อดีของการเขียนด้วยภาษารูปภาพนี้ก็คือ ทุก VI ที่เราเขียนขึ้นมาสามารถทำหน้าที่เป็น subVI ได้ แต่เนื่องจากเราจำเป็นจะต้องกำหนดลักษณะของ subVI ให้เป็นรูปภาพ และมีช่องที่จะต้องส่งเข้าสู่ subVI นั้น เราจึงได้กำหนด Icon และ Connector ขึ้น

หากเราจะกล่าวคร่าวๆ Icon นี้ก็อาจจะมองว่าเป็น node ในอีกรูปแบบหนึ่งก็ได้ โดย Icon จะหมายถึง Node ของ subVI ในทุกครั้งที่เราเขียน VI เราจะพบว่า Lab VIEW จะให้ VI นั้นสามารถทำงานเป็น subVI ได้ถ้าหากเราต้องการ โดยที่ Lab VIEW จะกำหนด icon ให้กับทุก VI ที่เขียนขึ้น ซึ่งเราสามารถเปลี่ยนแปลงรูปแบบของ Icon ที่ Lab VIEW เขียนขึ้นได้

หากเราเปิด Lab VIEW ใหม่ขึ้นมาเราจะสามารถสังเกตรูปของ Icon ที่ Lab VIEW กำหนดขึ้นมาได้ โดยรูปของ icon จะปรากฏอยู่ที่มุมบนด้านซ้ายใต้ Tittle Bar ของทั้งหน้าต่าง Block Diagram และ Front Panel เมื่อเรานำ VI นี้ ไปใช้เพื่อเป็น subVI เราสามารถที่จะกำหนดให้มี Input และ Output ที่จะเข้าและออกจาก VI ของเราได้ ซึ่งการเข้าและออกของข้อมูลสู่ subVI นั้นจะผ่านทาง Connector ซึ่งหากจะเปรียบเทียบกับภาษาตัวอักษรที่เราคุ้นเคยแล้ว การใช้ Connector ก็เหมือนกับการใช้คำสั่ง parameter ในภาษา C หรือการใช้คำสั่ง Function ในภาษา FORTRAN นั่นคือการกำหนดว่าข้อมูลใดเป็นข้อมูลที่ส่งไปสู่ Subroutine และข้อมูลใดเป็นข้อมูลที่จะรับกลับออกมาจาก Subroutine ทำนองเดียวกันกับในLab VIEW คือเราจะส่งข้อมูลเข้าสู่ Icon หรือ subVI โดยผ่านทาง Input connector เมื่อข้อมูลได้รับการประมวลใน subVI แล้วก็ส่งกลับมาทาง Output Connector Terminal

โดยปกติแล้ว Connector จะถูกบังอยู่ด้านหลังของรูป Icon เราสามารถแสดงให้เห็น Connector ได้โดยการใช้คำสั่ง Show Connector ซึ่งรายละเอียดเหล่านี้จะกล่าวในภายหลัง ในรูปต่อไปนี้เป็นการแสดง Icon และ Connector ของ VI หนึ่งซึ่งเป็นส่วนที่มาพร้อมกับ Lab VIEW ส่วนที่เป็นรูปตรงกลางเราเรียก Icon และส่วนที่เป็นสายต่อต่างๆ เราเรียก Connector

2. Menu and Palette

- **Menu Bar** จะประกอบด้วยเมนูต่างๆ คล้ายกับโปรแกรมอื่นใน Windows คือ จะประกอบด้วย File, Edit, Operated, Project, Windows และ Help ซึ่งการเลือกใช้ใดก็ได้เรา Click ที่เมนูนั้น

หากเราลอง Click เมนูทีละส่วนตามไปเราจะได้ Pull Down Menu ออกมาจากแต่ละ menu หลัก ซึ่งในแต่ละ menu มีคำสั่งในแต่ละลักษณะต่อไปนี้

File Menu จะเป็นคำสั่งที่จะเปิด, เก็บ, พิมพ์ VI

Edit Menu จะเป็นคำสั่งที่จะแก้ไข เช่น undo (เริ่มมีใน Lab VIEW 5.0) , cut, copy, Preference เป็นต้น

Operate Menu จะเป็นชุดคำสั่งเพื่อให้ Lab VIEW ทำงานเช่น run, stop และอื่นๆ

Project Menu จะเป็นชุดคำสั่งเพื่อบังคับการทำงานของ Lab VIEW เมนูนี้จะมีประโยชน์เมื่อเราทำงานกับ VI ขนาดใหญ่และมี subVI หลายๆ ชุดใน VI หลัก

Windows Menu จะเป็นการบังคับให้เปิดหน้าต่างที่เราต้องการ เช่น Front Panel หรือ Block Diagram รวมทั้งแสดง Palette ต่างๆ ด้วย

Help menu จะเป็นการใช้เมื่อต้องการคำอธิบายหรือความช่วยเหลือต่างๆใน

Lab VIEW

เนื่องจาก Lab VIEW มีคำสั่งเป็นจำนวนมากในเอกสารนี้ไม่ได้มีจุดมุ่งหมายที่จะให้ผู้อ่านเข้าใจคำสั่งทุกคำสั่งแต่เป็นการอธิบายเพื่อให้เราสามารถเริ่มใช้ Lab VIEW ได้ง่ายขึ้น สำหรับรายละเอียดของทุกคำสั่ง ผู้อ่านคงจะต้องอ่านจากเอกสารที่มาพร้อมกับ Lab VIEW

- **Toolbar** บน Toolbar ของ Lab VIEW นี้ก็จะคล้ายกับ Toolbar ของโปรแกรมต่างๆ ที่ทำงานบน Windows นั่นคือเป็นการรวบรวมคำสั่งที่ใช้อยู่ประจำให้มาอยู่ในรูปของ button เพื่อสะดวกในการใช้ สำหรับ toolbar ของ block diagram จะมีมากกว่า ของ front panel อยู่เล็กน้อย

สำหรับแต่ละ button บน toolbar จะมีชื่อและหน้าที่ดังนี้

 Run Button มีลักษณะเป็นลูกศร ใช้สั่งการให้ VI ทำงานเมื่อเรา dick ที่ปุ่มนี้ลักษณะของ Run button จะเปลี่ยนไปตามการทำงานของ VI ในขณะนั้นคือ จะเปลี่ยนลักษณะเป็นลูกศรวิ่ง

 เมื่อโปรแกรมกำลังทำงาน และถ้าหากว่า Run button ปรากฏเป็นรูปลูกศรขาด  แสดงว่า VI นั้นยังไม่พร้อมทำงานหรือมีข้อผิดพลาดอยู่ใน VI และคำว่า BROKEN VI จะเป็นศัพท์ที่นิยมใช้เมื่อโปรแกรมหรือ VI นั้นเกิดความผิดพลาดขึ้นแล้วทำให้ลูกศรขาด

 Continuous Run Button มีลักษณะเป็นลูกศรวน เนื่องจากการทำงานของ Data Flow จะเริ่มต้นเมื่อได้รับข้อมูล และสิ้นสุดลงเมื่อเสร็จสินการประเมินผลข้อมูล อย่างไรก็ตามในกรณีที่เราต้องการให้ VI ทำงานซ้ำต่อไปเรื่อยๆ นั่นคือ เมื่อจบการทำงานในครั้งแรกก็ให้กลับไปเริ่มต้นทำใหม่ไปเรื่อยๆ เราสามารถจะใช้ปุ่มนี้เป็นเครื่องมือทำคำสั่งนี้ได้ และเมื่อ VI ทำงานอย่างต่อเนื่องปุ่มนี้จะมีลักษณะเป็นลูกศรวนสีดำ 

 หรือ  Abort Button มีลักษณะเป็นเครื่องหมายจราจรให้หยุด ก่อนที่เราจะสั่งให้ VI ทำงานปุ่มนี้จะใช้งานไม่ได้ (Inactive และจะมีสีเทา) แต่เมื่อเราให้โปรแกรมทำงานด้วย Run หรือ Continuous Run ก็ตาม ปุ่มนี้จะปรากฏให้ใช้งานได้ (active) หากเรากดปุ่มนี้จะเป็นการยกเลิกการทำงานของ VI

ข้อควรระวังอย่างหนึ่งก็คือ LabVIEW ไม่แนะนำให้ใช้ Abort button เพื่อใช้เป็นการหยุดการทำงานในสภาพปกติเพราะจะทำให้ข้อมูลบางส่วนสูญหายหรือค้างอยู่ในหน่วยความจำได้ เพราะกระบวนการคำนวณยังไม่สิ้นสุด LabVIEW แนะนำว่าในโปรแกรมเราควรสร้าง คำสั่ง ขึ้นมาหยุดการทำงานของโปรแกรมที่เหมาะสม ซึ่งเราจะแสดงให้เห็นต่อไป

 Pause Button ปุ่มนี้เหมือนกับปุ่มบนเครื่องเสียงหรือ VCR ทั่วไป คือสั่งให้โปรแกรมหยุดการทำงานชั่วคราว เพื่อเราจะสามารถแก้ไขโปรแกรมได้ เช่น สั่งให้ข้ามบางขั้นตอนหรือออกจากบางขั้นตอน (step out) เป็นต้น

 Single Step Button ประกอบด้วยปุ่ม Step Into, Step Over, Step Out เป็นการบังคับให้กับ VI ทำงานตามขั้นตอนที่เราต้องการ เราจะกล่าวถึงรายละเอียดของปุ่มเหล่านี้ ในภายหลัง

 Execution Highlighting Button จะสั่งให้ VI ทำการ Highlight การไหลผ่านของข้อมูลเมื่อผ่านไปบนส่วนต่างๆ ของ diagram และเราจะเห็นว่ามีการแสดงค่าของข้อมูลเมื่อผ่านส่วนต่างๆ ในขณะนั้นไปด้วย

 Warning Button เป็นการเตือนการทำงานของ LabVIEW ถ้าหากว่าเรากำหนดให้ LabVIEW มีการเตือนและเมื่อปุ่มนี้ปรากฏเราสามารถให้แสดงข้อมูลที่มีการเตือนโดยการ click ที่

ปุ่มนี้ การปรากฏปุ่มเตือนนี้ไม่ให้แสดงว่าเกิดการผิดพลาดเพียงแต่เราเตือนว่ามีสิ่งผิดปกติเกิดขึ้นในโปรแกรมของเรา

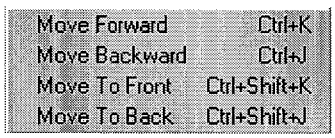
Front Ring เราสามารถเปลี่ยนรูปแบบตัวหนังสือที่ปรากฏบน VI ได้โดยใช้ Front Ring นี้บน toolbar ซึ่งก็จะเหมือนกับ Front Ring บนโปรแกรม word Programming ทั่วๆ ไป ที่ทำงานบน Windows


คำสั่งเหล่านี้จะเป็นการจัดวางตำแหน่งของส่วนประกอบต่างๆ ใน Front Panel และ Block Diagram ให้ตรงกันหรือมีระยะห่างที่เราต้องการซึ่งเป็นคำสั่งเพื่อเพิ่มความสวยงามให้กับ VI ของเรา


Alignment Ring ใช้เมื่อต้องการวางแนวของ Object ต่างๆ ให้อยู่ในลักษณะแนวที่เราต้องการ


Distribution Ring ใช้เมื่อต้องการจัดระยะระหว่าง Object ตามแนวต่างๆ ให้เป็นไปตามที่เราต้องการ


Reorder Ring เป็นส่วนที่ใช้จัดอันดับบน Front Panel และ Block Diagram ว่าวัตถุที่วางอันใดจะอยู่หน้าสุด อันใดจะอยู่หลังสุดใหม่ เพราะปกติ LabVIEW จะให้วัตถุที่วางลงอันแรกอยู่ล่างสุด และที่วางหลังจะทับอันแรกไปเรื่อย เราสามารถจัดลำดับได้โดยเลือกตัวเลือกต่อไปนี้ ซึ่งเราคงคุ้นเคยกับโปรแกรมวาดรูปต่างๆ ดีอยู่แล้ว



- โหมดการทำงานของ VI

เมื่อเปิด VI ขึ้นมาครั้งแรก VI นั้นจะอยู่ในโหมดการแก้ไข (Edit Mode) นั่นคือเราสามารถแก้ไข เปลี่ยนแปลงค่าต่างๆ ที่เราต้องการได้ แต่เมื่อเราให้ VI ทำงาน VI จะเข้าสู่โหมดการทำงาน (Run Mode) ซึ่งหมายความว่าเราไม่สามารถแก้ไขโปรแกรมได้ในขณะนี้ แต่อาจเปลี่ยนแปลงค่าของ Control ต่างๆ บน Front Panel ได้ โดยทั่วไปแล้ว VI จะอยู่ในโหมดการทำงานจนกว่าจะทำงานเสร็จหรือเราสั่งให้หยุดการทำงาน และเมื่อ VI หยุดการทำงานในโหมดการทำงานแล้ว VI จะกลับเข้ามาอยู่ในโหมดการแก้ไข

- Pop-Up Menus ในการเขียน VI จะมีเมนูอีกประเภทหนึ่ง ซึ่งเรียกว่า Pop-up menu ซึ่งเมนูประเภทนี้อาจจะใช้มากกว่าเมนูแบบแรกในการเขียน VI

การที่เราจะเรียก Pop-up menu ขึ้นมาใช้วิธีการง่ายที่สุดใน Windows 95, Windows 98 หรือ Windows NT ก็คือ Click บน Object นั้นๆ ด้วยเมาส์ปุ่มขวา ลักษณะของ Pop-up menu ของแต่ละ Object จะแตกต่างกันออกไป แล้วแต่ชนิดของ Object นั้นๆ

ส่วนประกอบหลักๆ ใน Pop-up menu ใน Object ต่างๆ จะมีดังนี้

- Change to Control หรือ Change to Indicator เป็นการเปลี่ยนชนิดของ Control และ Indicator เช่น ถ้าหากว่า object ของเราเป็น Indicator อยู่ใน Pop-up Menu จะแสดงคำสั่ง Change to Control หากเราเลือกคำสั่งนี้ Indicator ของเราจะเปลี่ยนเป็น Control ทันทีและในทางกลับกันกับคำสั่ง Change to Indicator ข้อควรระวังในการใช้คำสั่งนี้ก็คือ Control Object และ Indicator Object มีหน้าที่ตรงกันข้ามกันและไม่สามารถที่จะทำทำงานแทนกันได้ หากเราเปลี่ยนโดยไม่ระวังเราอาจเกิดความผิดพลาดขึ้นมาใน VI ของเราได้

- Find Terminal และ Find Control/Indicator ถ้าหากเลือก Find Terminal จาก Pop-up menu ของ Front Panel เราจะพบว่า LabVIEW จะแสดงและเน้นสีกับ terminal ของวัตถุนั้นของ Object ใน Block Diagram และถ้าหากเราเลือก Find Control/Indicator เราจะพบว่า LabVIEW จะแสดงและเน้นสี Object นั้นบน Front Panel คำสั่งทั้ง 2 นี้ จะปรากฏบน Pop-up menu เฉพาะในกรณีที่มี Object อยู่บนทั้ง Front Panel และ Block Diagram เท่านั้น เช่น เราจะไม่พบคำสั่งนี้ถ้าหากเรา Pop-up menu ของ Node

- Show หลายๆ object จะมี Show menu และมักจะเป็น Hierarchical menu คือจะเป็นการกำหนดให้แสดงส่วนต่างๆ ของ Object นั้น เช่นแสดงชื่อ (Label) หน่วย เป็นต้น

- Data Operation เมนูนี้เป็นเมนูที่จะปรากฏขึ้นเฉพาะขณะที่ VI อยู่ในโหมดการทำงานเท่านั้น ซึ่งจะใช้กับ Control และ Indicator โดยจะมีเมนูย่อยต่อไปนี้

Reinitialize to Default จะเป็นการกำหนดค่าของ Object ให้กลับไปที่ค่าเริ่มต้น (Default) ที่ LabVIEW กำหนดมาให้ครั้งแรก Make Current Value Default จะกำหนดค่าในขณะนั้นให้เป็นค่าเริ่มต้นเมื่อเริ่มให้ VI ทำงาน Cut Data, Copy Data และ Paste Data เป็นการตัด, คัดลอก หรือใส่ข้อมูล ออกจากหรือลงใน Control หรือ Indicator

- Description... เมนูนี้จะเป็นการนำ Dialog Box หรือช่วงข้อความขึ้นมาเพื่อเราจะได้ใส่หรืออ่านข้อความที่ Object นั้นๆ ใช้ ในโหมดการทำงานเราสามารถดูข้อความที่เกี่ยวข้องกับ Object นั้น ได้เท่านั้นไม่สามารถแก้ไขได้

- Show หรือ Hide Control/Indicator เป็นการสั่งให้แสดงหรือไม่แสดง Front Panel Terminals ของ Object นั้นเพื่อผู้ใช้จะไม่สามารถเห็น Object นั้นใน Front Panel แต่ยังจะปรากฏอยู่ใน Diagram ซึ่งเราอาจจำเป็นต้องใช้ในบางกรณี

- Create... เป็นคำสั่งที่ใช้สร้าง Attribute Node หรือ Local Variable ของ Object นั้นซึ่งจะกล่าวต่อไปในภายหลัง

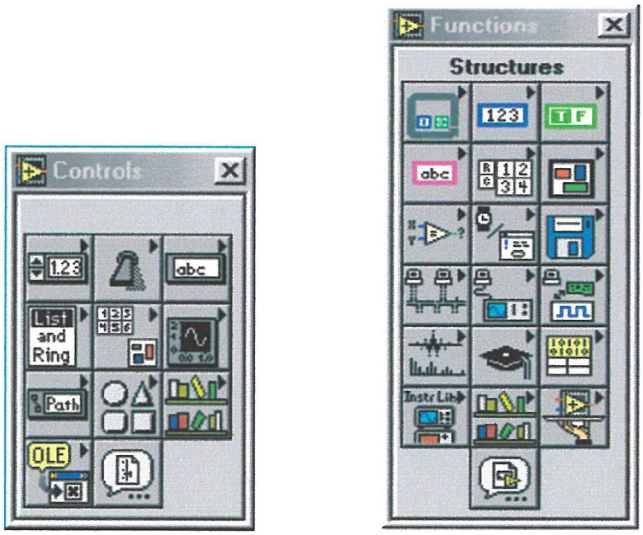
- Key Navigation ใช้ Key Navigation... เพื่อสั่งการโดยผ่านแป้นพิมพ์ (Keyboard) โดยการกดปุ่มหลายๆ ปุ่ม ขณะที่แสดง Front Panel อยู่ เมื่อผู้ใช้กดปุ่มตามที่ตั้งไว้ขณะที่ VI กำลังทำงาน LabVIEW จะทำงานเหมือนกับ ผู้ใช้ใช้เมาส์กดบนวัตถุนั้น จะทำให้วัตถุนั้นเป็น Key Focus คือ Active หรือพร้อมทำงานใน Object นั้น

- Online Help ขอความช่วยเหลือในส่วนที่เกี่ยวข้องกับ Object หรือวัตถุนั้น

- Replace เป็นการแทนที่ Object นั้นด้วย Object อื่น เมื่อเราเลือกตัวเลือกนี้ เราจะพบว่าเราจะเข้าสู่ Controls หรือ Functions Palette ขึ้นอยู่กับว่าวัตถุนั้นเป็นวัตถุใดหรือเราอยู่บนหน้าต่างใด และจะทำให้เราสามารถแทน Object เพิ่มด้วย Object ใหม่ได้ขณะที่สายเชื่อมเดิมยังคงอยู่

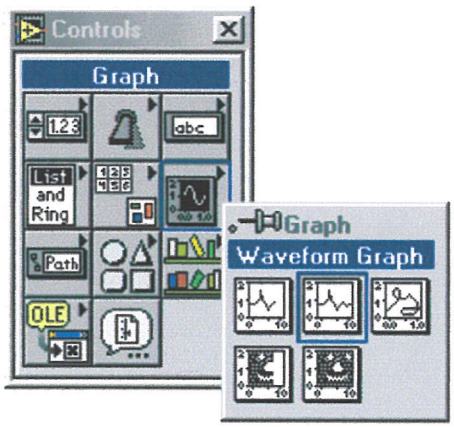
- Contorts และ Function Palettes

แม้ว่าเมื่อเราเปิด VI ขึ้นมาจะมี Controls และ Functions Palettes ปรากฏขึ้นแต่ทั้งคู่จะใช้งานไม่พร้อมกันคือControls Palettes จะปรากฏขึ้นให้เห็นได้ก็ต่อเมื่อ Front Panel อยู่ในสภาพพร้อมทำงานอยู่เท่านั้น ถ้าหาก Front Panel ไม่อยู่ในสภาพพร้อมทำงานส่วนของ Controls Palettes จะหายไป หรือเราอาจปิดเฉพาะ Controls palettes โดยใช้เมาส์กดที่ Exit Button บน Title bar ก็ได้ Function Palettes จะปรากฏขึ้นให้เห็นได้ก็ต่อเมื่อ Block Diagram อยู่ในสภาพพร้อมทำงานอยู่เท่านั้นถ้าหาก Block Diagram ไม่อยู่ในสภาพพร้อมทำงาน ส่วนของ Tools Palettes นี้จะหายไปหรือเราปิด Palette นี้เอง



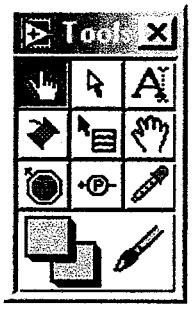
รูปที่ 2.57 รูปลักษณะของ Controls และ Functions Palette

Palettes ทั้งสองจะมี Sub palette บรรจุอยู่ถ้าหากว่าเราเลื่อนลูกศรของเมาส์ไปที่ปุ่มของ Sub palette ในชื่อของ Sub palette นั้นจะปรากฏขึ้น และหากเรากดลงไปที่ปุ่มของ Sub palette ใด และกดค้างไว้เราจะพบว่าจะมีการแสดงส่วนประกอบของ Sub palette นั้นต่อออกไปอีก และในหลายกรณีใน Sub palette ก็จะแยกย่อยเป็น Sub palette ต่อไปอีกดังที่แสดงต่อไปในรูป



รูปที่ 2.58 รูปแสดงส่วนประกอบของ Sub palette

- **Tools Palette** คือหน้าที่พิเศษของตัวชี้ของเมาส์ เราจะใช้ Tools เพื่อจะให้ทำงานในการแก้ไขหรือปฏิกิริยาหน้าที่ซึ่งเราต้องการ คล้ายกันกับที่เราใช้ในโปรแกรมการวาดรูปทั่วๆ ไป ใน Tools Palette จะประกอบด้วยปุ่มที่มีหน้าที่ต่างๆ ดังนี้



รูปที่ 2.59 รูปแสดงส่วนประกอบ ของ Tools Palette

ตารางที่ 2.17 สำหรับหน้าที่ของอุปกรณ์ต่างๆ บน Tools Palette นี้สรุปได้ตามตารางต่อไปนี้

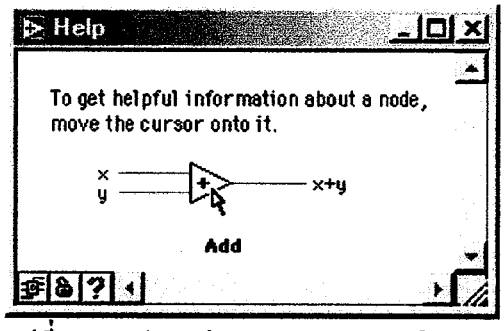
ลักษณะปุ่ม	ชื่อ	หน้าที่
	Operating Tool	ช่วยให้เราเปลี่ยนค่าของ control บน front panel ขณะที่ VI ทำงานหรือเปลี่ยนค่าอื่นๆ ในขณะที่ VI อยู่ในโหมดแก้ไข
	Positioning Tool	ช่วยปรับขนาด, เคลื่อนย้ายที่ หรือเลือก object
	Labeling Tool	สร้างหรือแก้ไข Text
	Wiring Tool	ใช้ในการต่อเชื่อมสายใน bock diagram เข้าด้วยกัน
	Pop-up Menu Tool	ถ้าตัวชี้นี้ชี้ไปที่ object ใด ก็จะเกิด Pop-up Menu ของ object นั้นขึ้น ใช้แทนการกดเมาส์ปุ่มขวาที่ Object ได้

ลักษณะปุ่ม	ชื่อ	หน้าที่
	Scroll Tool	เลื่อนภาพบนหน้าต่างที่กำลัง active อยู่ไปในทิศทางที่ต้องการ
	Breakpoint Tool	เป็นการใส่ตำแหน่งหยุดลงใน block diagram เมื่อข้อมูลเดินทางมาถึงจุดนี้ การประมวลผลจะชะลอชั่วคราว เพื่อให้เราตรวจสอบ และแก้ไขการทำงาน
	Probe Tool	สร้างเครื่องวัดลงบนเส้นเชื่อมเพื่อแสดงค่าข้อมูลในขณะผ่านเครื่องวัดนั้นๆ
	Color Copy Tool	ใช้ในการคัดลอกสีจาก object ที่เราต้องการเพื่อสามารถปรับแก้สีที่ object อื่นให้เหมือน object นั้น
	Color Tool	ใช้ในการปรับแต่งสีของ VI ให้เป็นไปตามต้องการ

- **HELP!** ในการจัดทำเอกสารเพื่อแนะนำการใช้โปรแกรม LabVIEW ขั้นพื้นฐานนี้ คงเป็นไปไม่ได้ที่เราจะบรรจุรายละเอียดทั้งหมดของ LabVIEW ลงในเอกสารนี้ ดังนั้นวิธีการหนึ่งที่จะช่วยเราได้คือการใช้ help ที่บรรจุมากับ LabVIEW โดย LabVIEW ได้บรรจุคำอธิบายการใช้งานของโปรแกรมมาให้ในหลายลักษณะ

เมื่อต้องการความช่วยเหลือหรือคำอธิบายใดๆ เราสามารถสั่งให้ LabVIEW แสดง Help ของ object นั้นๆ ได้โดยจาก Help menu เลือก Show Help หน้าต่างของ Help ก็จะปรากฏขึ้น หลังจากนั้นถ้าเรานำเมาส์ไปชี้ที่ Object ใด Help ของ Object นั้นก็จะปรากฏขึ้นบน Windows โดยปกติ Help ของ Object นั้นก็จะปรากฏขึ้นบน Windows โดยปกติ Help ของ Object นั้นจะบอกถึงหน้าที่ของ Object นั้น

การต่อเชื่อมสายหรือแสดง Connection ของ Object นั้นและอาจมีข้อมูลหรือข้อควรระวังของ Object นั้นๆ



รูปที่ 2.60 รูปตัวอย่างของ Help แสดงในภาพ

จากในรูปเราจะเห็นว่าที่มุมล่างซ้ายของหน้าต่าง Help จะพบปุ่มเล็กๆ 3 ปุ่ม แต่ละปุ่มมีหน้าที่ดังนี้

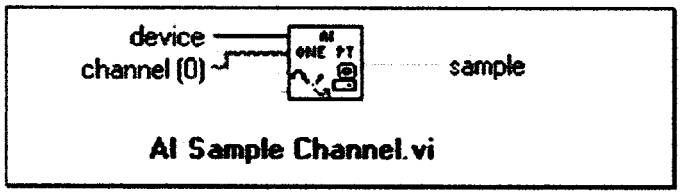
สลับระหว่าง Simple Help คือบอกความช่วยเหลืออย่างคร่าวๆ และ Detailed Help คือบอกรายละเอียดต่างๆมากขึ้น ใน Simple Help จะเป็นการบอกรายละเอียดเฉพาะส่วนที่จำเป็น เช่น สายต่อที่จำเป็นเข้าสู่ Icon เป็นต้น เราจะอธิบายในรายละเอียดต่อไป

Lock Help คือได้แสดง Help ของ Object นั้นต่อไปไม่ว่าเราจะเลื่อนลูกศรของเมาส์ไปที่ใดก็ตามบน Front Panel หรือ Block Diagram

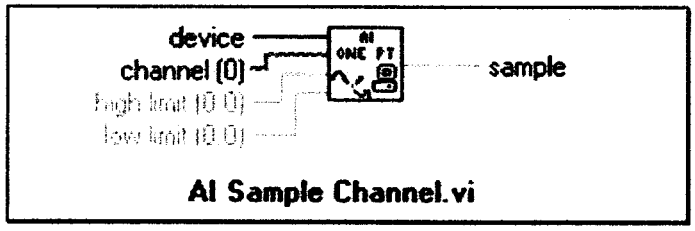
Online Help เป็นการแสดงการทำงานของ Object นั้นอย่างละเอียดและอาจมีตัวอย่างการทำงานของ object บางแบบด้วย

- Simple และ Detailed Help

สำหรับจุดเชื่อมหรือ Connection ของ Object นั้นบาง Connector อาจจะเป็น Connector ที่จำเป็นต้องมีค่าเข้าสู่ Object หรือ Node นั้น ในขณะที่บาง Connector อาจเป็นเพียงตัวเลือกว่าจะมหรือไม่ก็ได้ หากเราไม่ต่อสายที่ถูกต้องเข้ากับ Connector ที่มีความจำเป็นต้องใช้เข้าสู่ VI ของเราจะไม่อยู่ในสภาพที่พร้อมทำงาน และเครื่องหมาย RUN จะเป็นลูกศรขาด รูปต่อไปนี้แสดงความแตกต่างของการแสดง Help ทั้งสองแบบ



รูปที่ 2.61 รูปตัวอย่างของ **Simple Help**



รูปที่ 2.62 รูปตัวอย่างของ **Detail Help**

การใช้ Detailed Help จะบอกว่า Connector ตัวใดมีความต้องการข้อมูลในลักษณะใด และมีความจำเป็นหรือไม่ ส่วน Simple Help จะเป็นการแสดงเฉพาะ Connector ที่จำเป็นต้องใช้ในการนำข้อมูลไปใช้ในการประมวลผลข้อมูลเท่านั้น โดยทั่วไป

- Connection ที่จำเป็นจะแสดงด้วยตัวหนา
- Connection ที่แนะนำให้ต่อ (แต่ไม่จำเป็น) จะเป็นตัวหนังสือปกติ
- Connection ที่เป็นตัวเลือกจะปรากฏเป็นตัวหนังสือสีจางและจะไม่แสดงใน Simple Help

ถ้า Input ของ Function ใดไม่จำเป็นต้องเชื่อมต่อ ซึ่งอาจเป็นในกรณีของแนะนำให้ต่อหรือเป็นตัวเลือกก็ตาม ค่าเริ่มต้นจะปรากฏอยู่ในวงเล็บต่อจากชื่อของ Connection นั้น

- Online Help

Online Help จะเป็นการอธิบายการทำงานของ LabVIEW อย่างละเอียดซึ่ง Help ในส่วนนี้จะอธิบายขั้นตอน และหน้าที่ของแต่ละ Object อย่างละเอียด ซึ่งเราจะไม่ขอกล่าวในที่นี้เพราะการทำงานของ Online Help ของ LabVIEW ก็จะเหมือนกับ Help ของ Program อื่นๆ บน Windows

บทที่ 3
ขั้นตอนการดำเนินงาน

สามารถแบ่งขั้นตอนการดำเนินงานได้ดังนี้คือ

3.1 การออกแบบวงจรต่างๆ ภายในตู้อบเด็ก

3.1.1 วงจรแหล่งจ่ายไฟของระบบควบคุม

3.1.2 ส่วนตรวจจับอุณหภูมิที่ใช้ SHT15

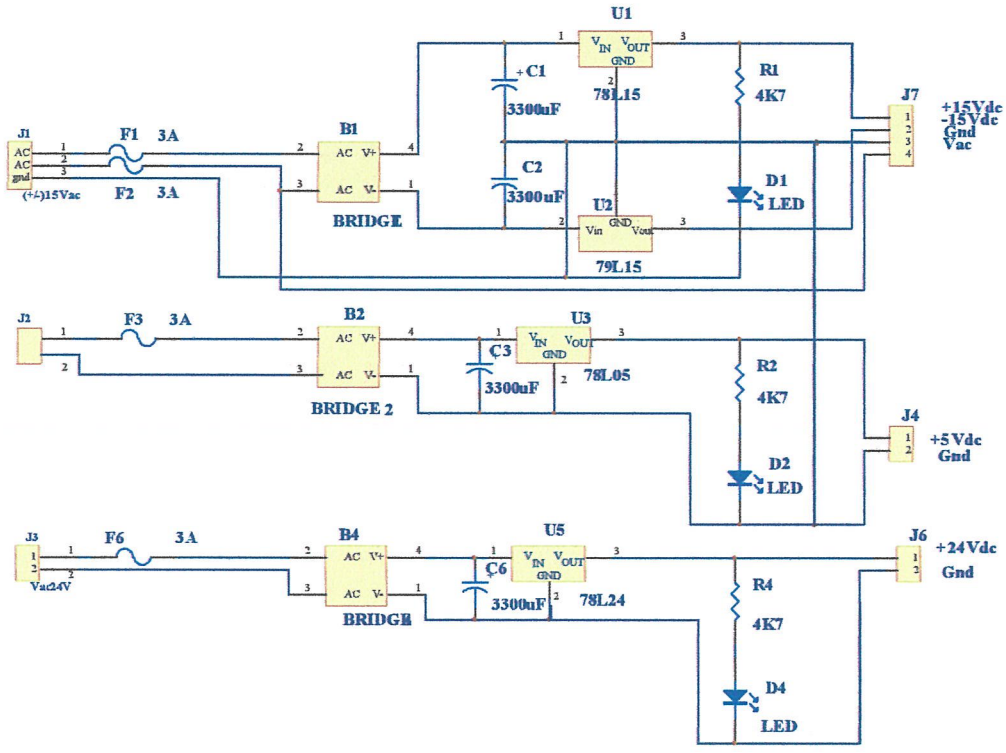
3.1.3 วงจรทริกและแยกโหลด

3.1.4 วงจรควบคุมการทำงานหลัก

3.1 การออกแบบวงจรส่วนต่างๆ ภายในตู้อบเด็ก

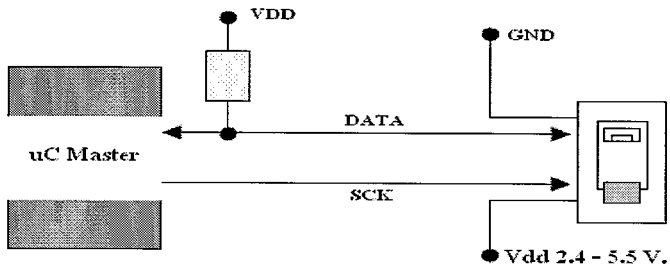
3.1.1 วงจรแหล่งจ่ายไฟกระแสตรง

วงจรจ่ายไฟกระแสตรง มีการรักษาระดับแรงดันคงที่ การออกแบบจะใช้ไอซีแบบประกอบด้วยขาอินพุท เอาท์พุท และขากราวนด์ ซึ่งตัวเลขที่บอกเบอร์ไอซีจะเป็นตัวบอกขนาดของแรงดันเอาท์พุท เบอร์ที่ให้แรงดันไฟบวกคือเบอร์ 78xx และให้แรงดันไฟลบคือเบอร์ 79xx



รูปที่ 3.1 วงจรแหล่งจ่ายไฟของระบบควบคุม

3.1.2 ส่วนตรวจจับอุณหภูมิที่ใช้ SHT15



รูปที่ 3.2 ตัวอย่างการต่อใช้งาน SHT15

- การต่อขา Vcc กับ GND ต้องต่อไฟเลี้ยงให้อยู่ระหว่าง 2.4- 5.5 V แล้วหลังจากที่จ่ายไฟเข้าที่ตัว SHT15 แล้ว SHT15 จะใช้เวลาประมาณ 11ms เพื่อเข้าสู่โหมด Sleep ดังนั้นต้องส่งข้อมูลก่อนที่ IC จะเข้าสู่โหมด Sleep ในการต่อ Vcc กับ GND ควรที่จะต่อ C 100 nF คร่อมระหว่างขา Vcc กับ GND

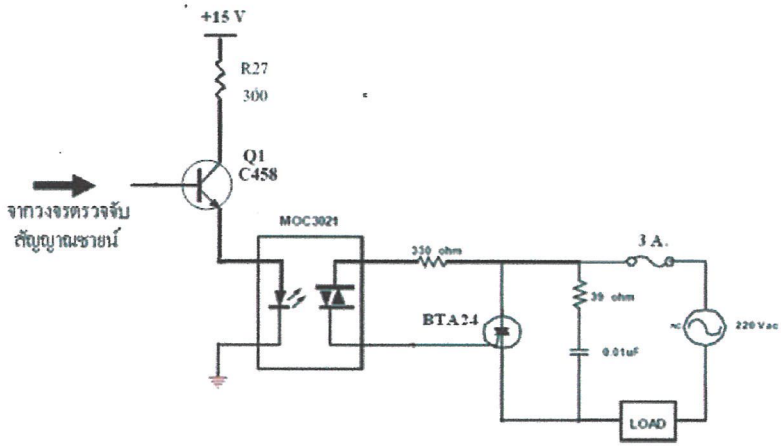
- การต่อขา DATA และขา SCK จะต่อแบบ Serial Interface(Bidirectional 2 –wire) ซึ่งจะเป็นการต่อในลักษณะที่คล้ายกับ I2C แต่ไม่เหมือน I2C โดยทั่วไป

- การต่อขา SCK จะเป็นการต่อแบบตรงระหว่างไมโครคอนโทลเลอร์ กับSHT15

- การต่อขา DATA จะมีการต่อ Pull-up เพื่อให้ได้สัญญาณที่มีค่าสูง ซึ่งการต่อ Pull-up จะ ต่อกันบ่อยๆในการใช้ไมโครคอนโทลเลอร์ และในช่วงที่ทำการส่ง DATAจำเป็นที่จะต้องทำให้ DATA มีความเสถียรในขณะที่ SCK high ซึ่งแสดง Timing Diagram ได้

3.1.3 วงจรทริกและแยกโหลด

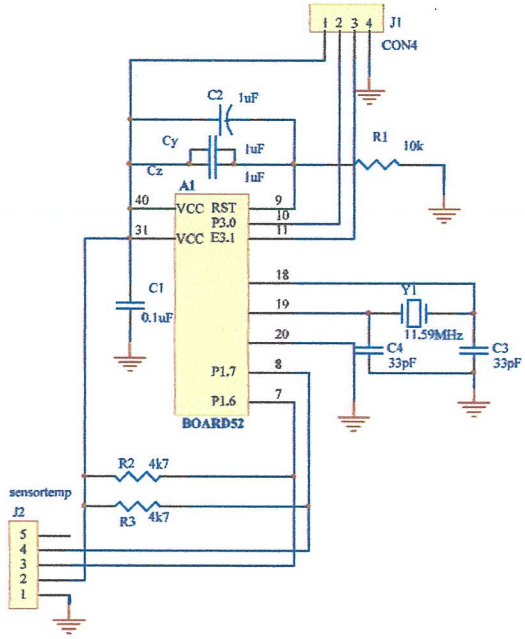
เป็นส่วนที่แยกวงจร High voltage ออกจากวงจรควบคุมต่างๆ เพื่อป้องกันแรงดัน 220 Vac ไหลเข้าสู่วงจรส่วนอื่นๆ



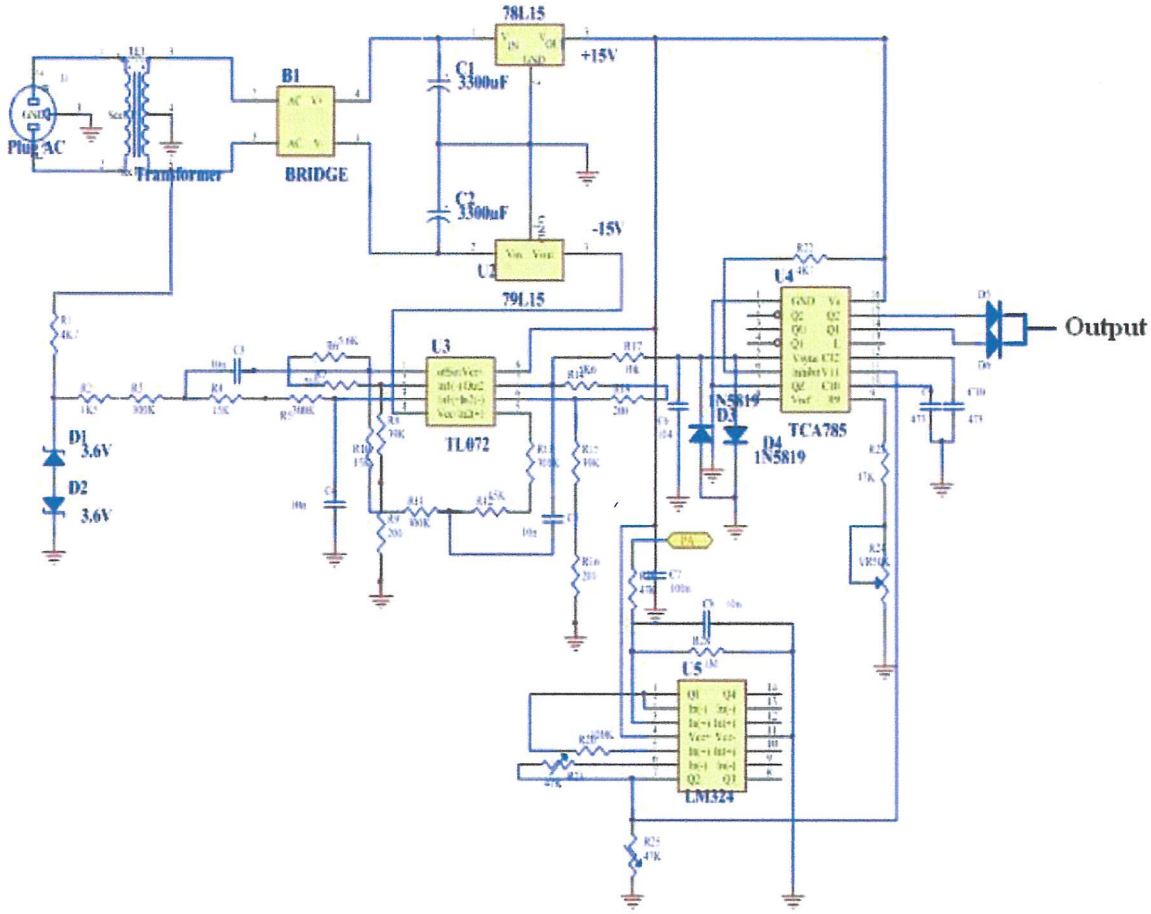
รูปที่ 3.3 แสดงวงจรทริกและแยกโหลด

3.1.4 วงจรควบคุมการทำงาน

เป็นวงจรที่ควบคุมการทำงานของตู้อบเด็ก มีไมโครคอนโทลเลอร์เป็นตัวประมวลผลควบคุมการทำงานทั้งหมด ประกอบด้วยส่วนที่ใช้ในการวัดและประมวลผลอุณหภูมิและความชื้น โดยใช้ SHT15 ในการตรวจจับ

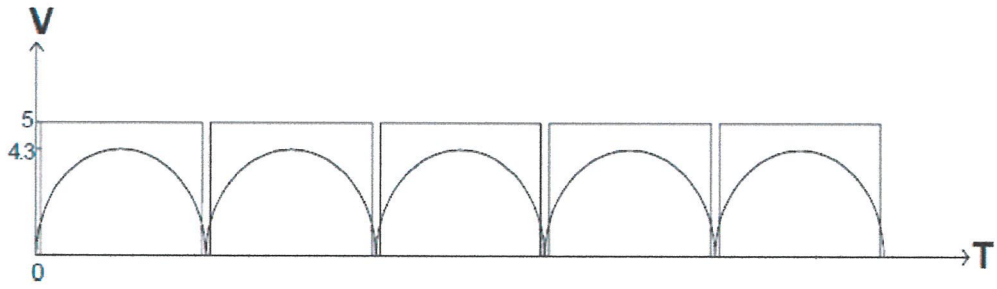


รูปที่ 3.4 วงจรส่วนที่ใช้ในการวัดและประมวลผลอุณหภูมิและความชื้น



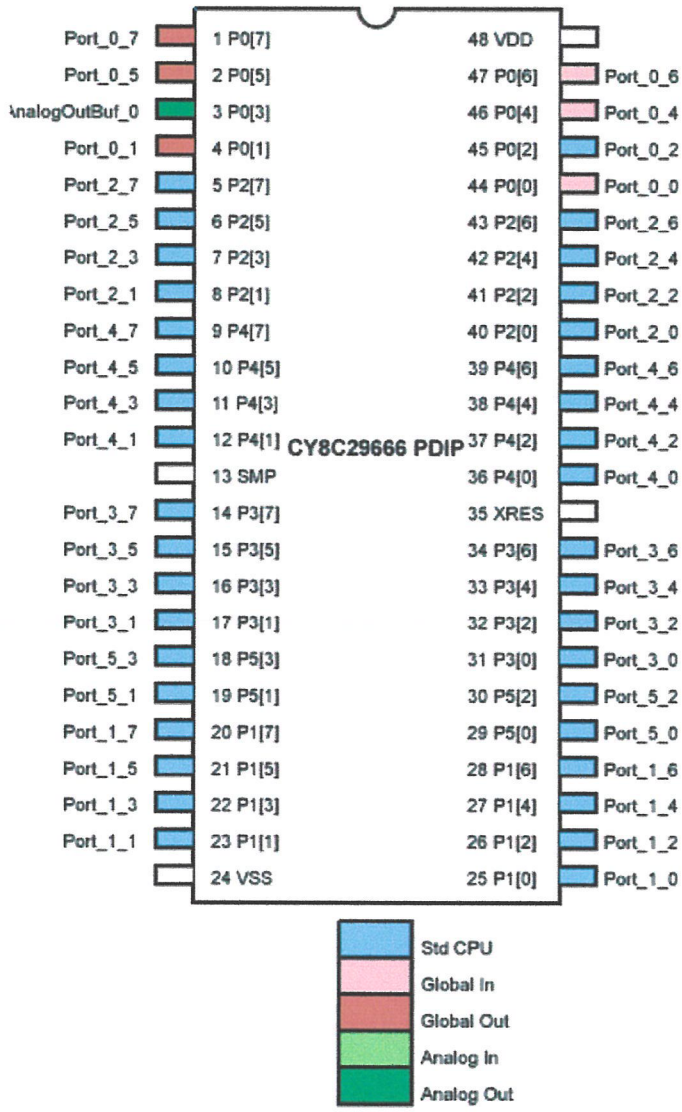
รูปที่ 3.5 วงจรตรวจจับสัญญาณซายน์

สัญญาณซายน์ 5 โวลล์เมื่อผ่านวงจรบริดจ์เรคติไฟเออร์จะให้สัญญาณเอาท์พุท full wave นำไปเข้าวงจรคอมพาราเตอร์โดยใช้ LM339 และแรงดันเปรียบเทียบสามารถปรับได้โดยใช้ VR10Kสามารถปรับช่วงที่เป็น 0 ของสัญญาณพัลล์เอาต์พุตได้ นำสัญญาณเอาต์พุตที่ได้จากวงจรคอมพาราเตอร์ไปเข้าขา INT0 ของไมโครคอนโทรลเลอร์เพื่อเป็นสัญญาณการอินเตอร์รัพท์ในการเริ่มดีเลย์

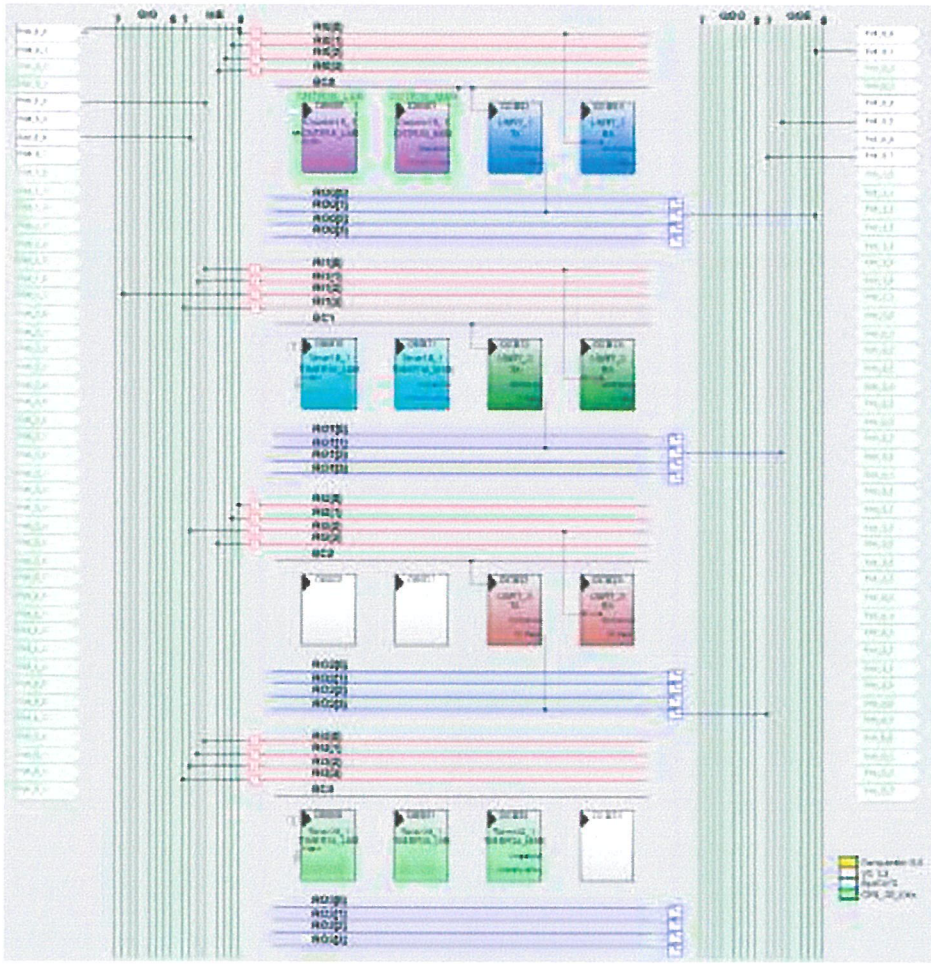


รูปที่ 3.6 สัญญาณซายน์เมื่อวงจรคอมพาราเตอร์ที่ใช้ LM 339

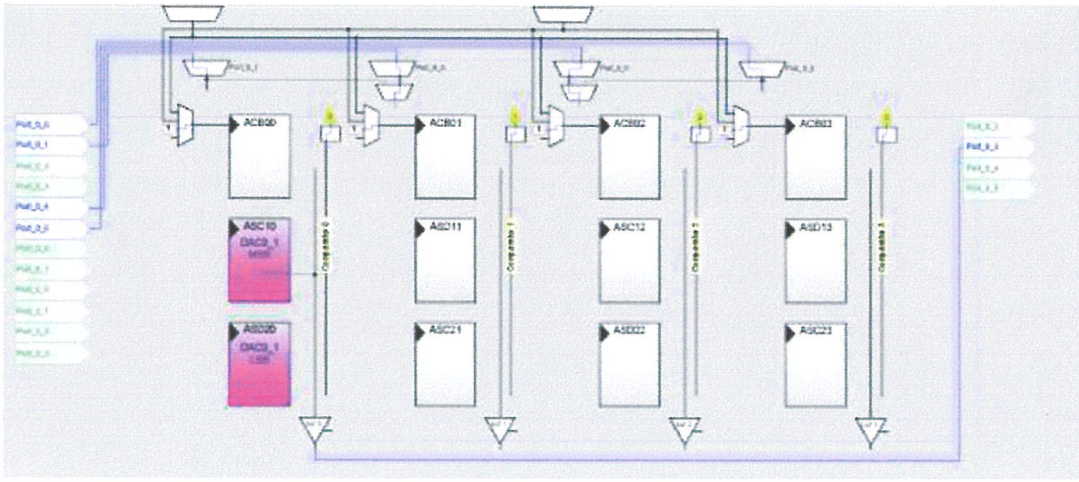
ตัวอย่างเมื่อไมโครฯ ต้องการอ่านค่า Humidity จาก sensor ซึ่งมี address =000 และคำสั่ง=00101 จะมีชุดคำสั่งดังนี้



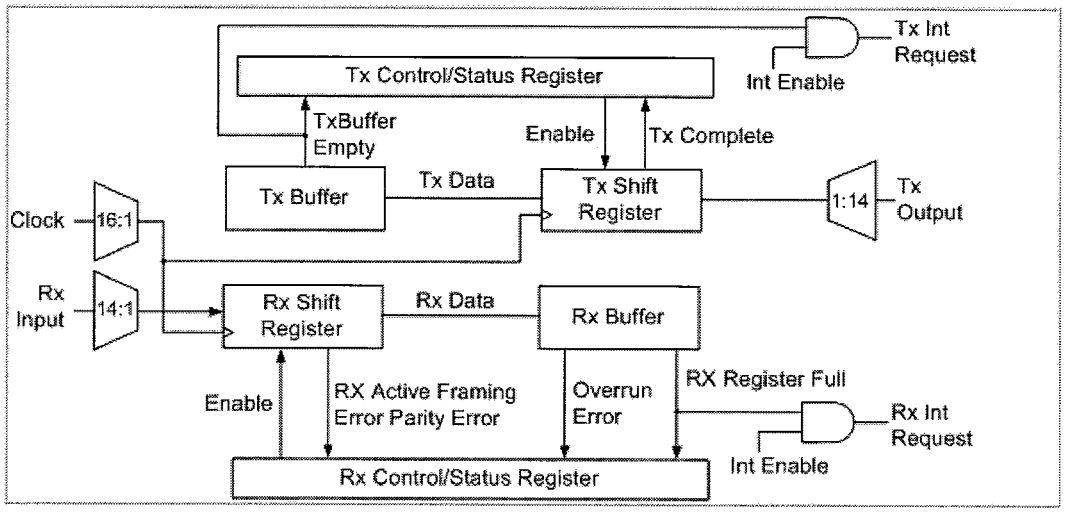
รูปที่ 3.7 PSoC เบอร์ CY8C29666



รูปที่ 3.8 การแสดงค่าการ set ค่าภายในของ โมดูล Psoc



รูปที่ 3.9 บล็อกแสดงค่าภายในของการต่อใช้งาน ของบล็อกโมดูลของ Psoc



รูปที่ 3.10 แสดงบล็อกไดอะแกรมภายใน User Modul Detail UART_(UART) Psoc

บทที่ 4
ขั้นตอนการทดลองและผลการทดลอง

4.1 ผลการทดลอง การ Control

ค่าจริงของ DAC 9 Bit ที่ใช้อยู่ในช่วง 0 – 511 ค่าที่ใช้ในการ control คือ 24 – 510 ซึ่งเปรียบเทียบกับแรงดัน Output ของ Psoc ดังนี้

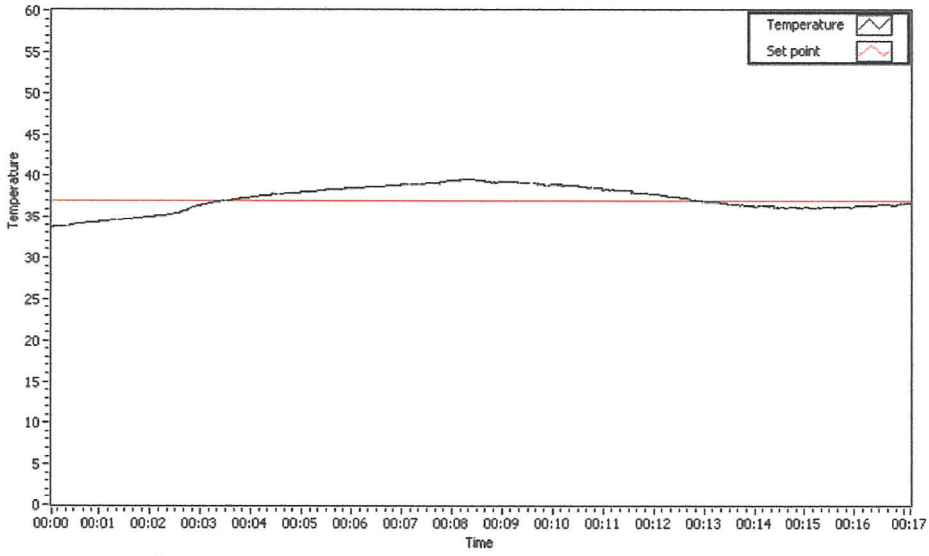
ตารางที่ 4.0 แสดงค่าของ DAC 9 Bit 24 – 510

Output จาก Psoc		จาก DAC
0.2 V.	≈	24
1 V.	≈	100
2 V.	≈	205
3 V.	≈	305
4 V.	≈	415
4.7 V.	≈	510

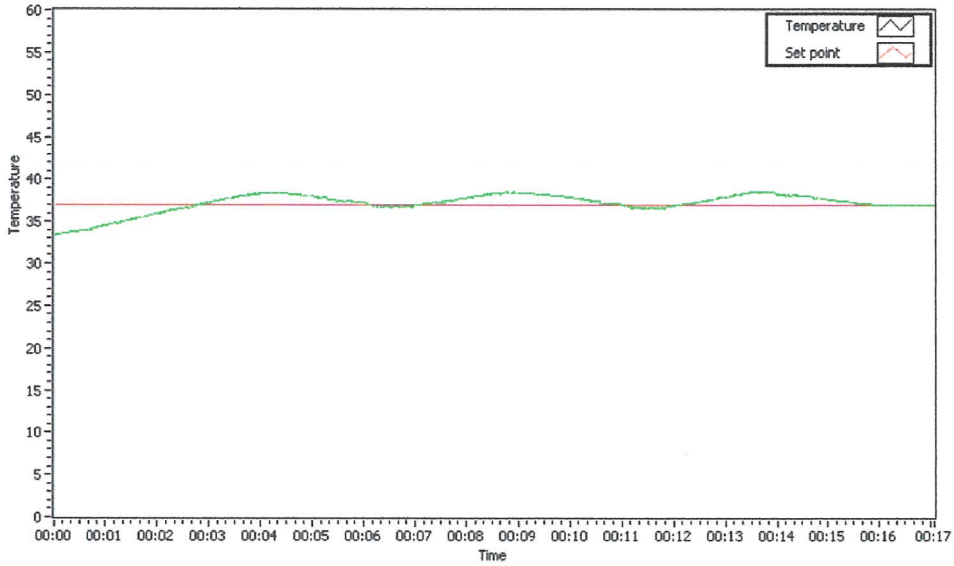
แรงดัน Output ที่ออกจาก Psoc ทำให้เกิดการในย่าน 0-5 V และค่าที่ได้ แสดงออกมาเป็นตัวเลขที่สามารถนำมาคอนโทรลซึ่งแปลผันตรงกับค่าแรงดันได้

4.2 ปรับค่าคงที่(K) ต่างๆ ของระบบพีไอดี (PID) ที่เหมาะสมกับระบบมากที่สุด

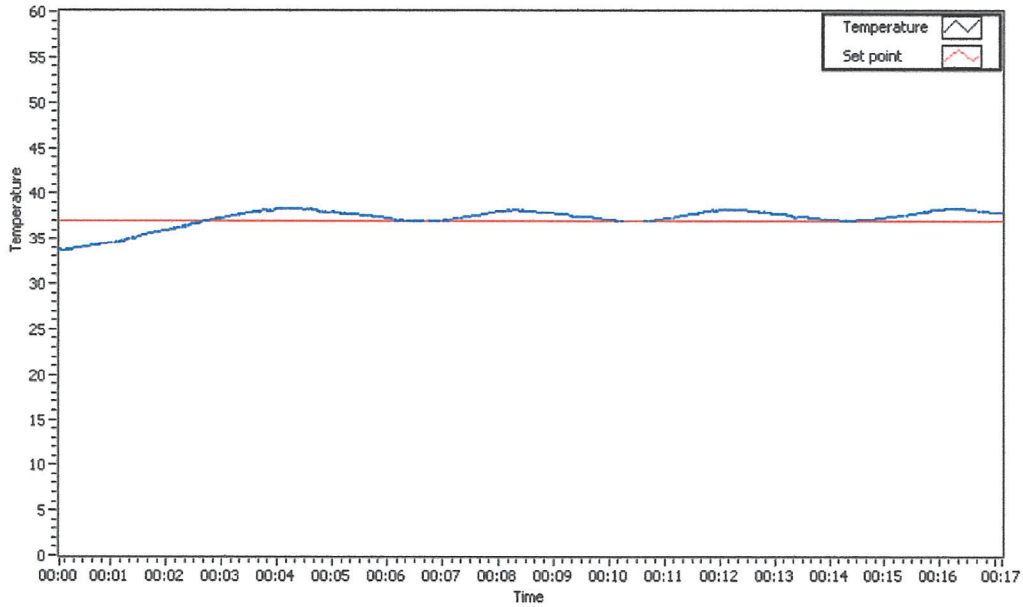
โดยเริ่มจากการปรับค่าค่าคงที่ของพี (Kp) ก่อนเป็นอันดับแรก ซึ่งจะต้องทำการสุ่มเปลี่ยนค่าไปเรื่อยๆจนได้ค่าที่เข้าใกล้จุดกำหนดมากที่สุดซึ่งค่าคงที่ดังนี้ 130, 120, 100, 80



รูปที่ 4.0 กราฟแสดงการปรับค่าคงที่ของพี (Kp=130) ค่า Offset ที่ (37 °C) Output จาก Psoc ที่ 0.2 V.

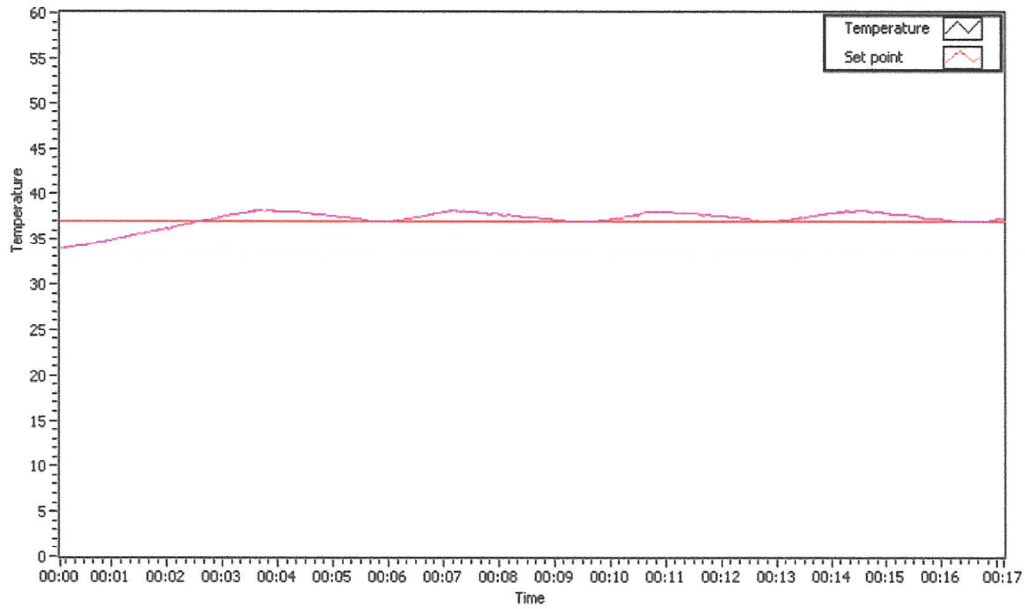


รูปที่ 4.1 กราฟแสดงการปรับค่าคงที่ของพี (Kp=120) ค่า Offset ที่ (37°C) Output จาก Psoc ที่ 0.2 V.



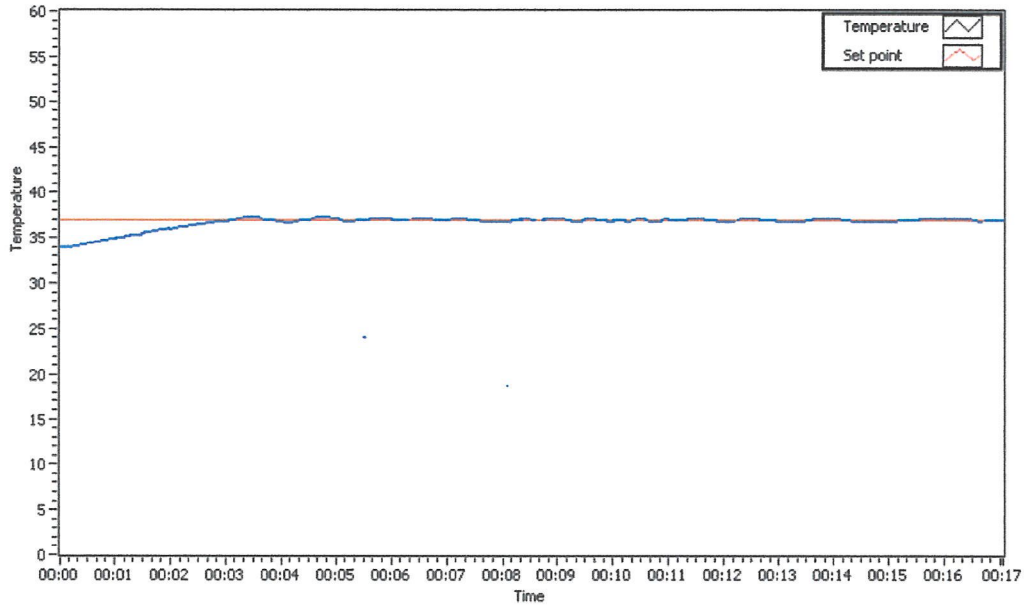
รูปที่ 4.2 กราฟแสดงการปรับค่าคงที่ของพี (Kp=100) ค่า Offset ที่ (37°C)

Output จาก Psoc ที่ 0.2 V.

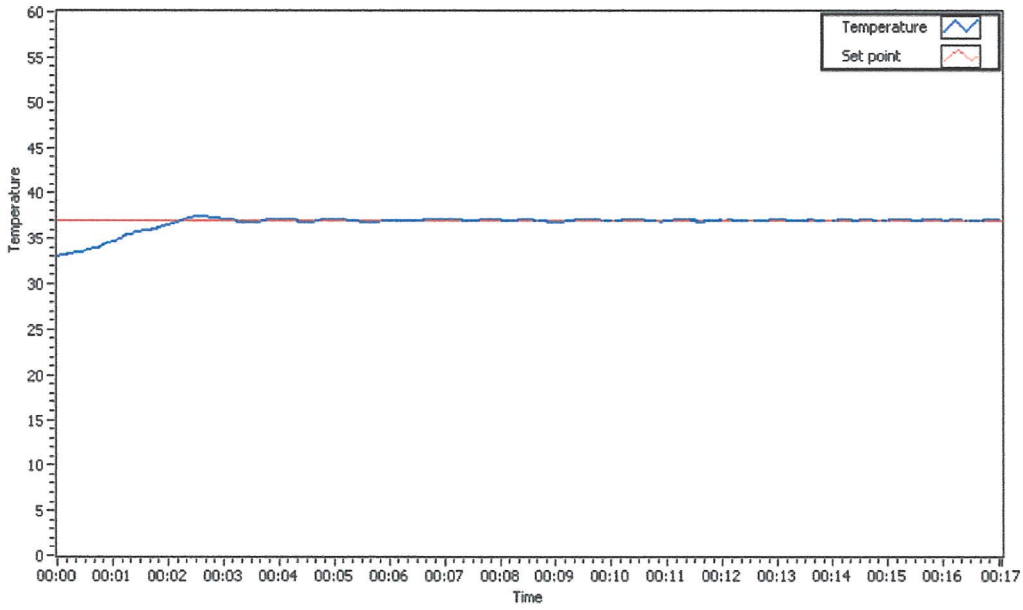


รูปที่ 4.3 กราฟแสดงการปรับค่าคงที่ของพี (Kp=80) ค่า Offset ที่ (37°C)

Output จาก Psoc ที่ 0.2 V.



รูปที่ 4.4 กราฟแสดงการปรับค่าคงที่ของพี (Kp=80), (Ki=0.001) ค่า Offset ที่ (37°C) Output จาก Psoc ที่ 0.2 V.



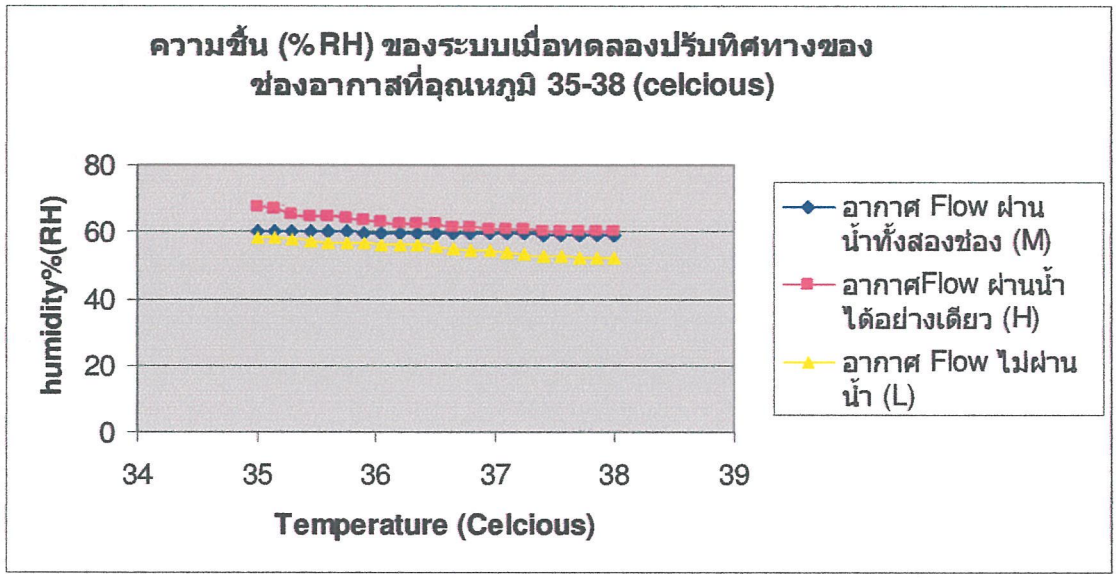
รูปที่ 4.5 กราฟแสดงการปรับค่าคงที่ของ PID (K_p=100), (K_i=0.4), (K_d=0.001) ค่า Offset ที่ (37°C)Output จาก Psoc ที่ 0.2 V.

-ในค่าแรกที่ทำการปรับค่าคงที่ ($k_p = 130$) ทำให้ได้กราฟที่มีการแกว่งของค่าที่ออกมาทางเอาท์พุต ไม่เข้าใกล้จุดที่เซ็ตค่าที่ 37°C จะใช้เวลานานมากกว่าจะเข้าใกล้ค่าที่ตั้งไว้

-ในการสุ่มค่าปรับค่าคงที่ ($k_p = 80$) จะพบว่าทำให้ได้ค่ากราฟที่เข้าใกล้จุดตั้งค่าของอุณหภูมิในเวลาที่สั้นมากขึ้น และใช้เวลาน้อยสุดในการเข้าใกล้ค่าคงที่ที่ตั้งไว้ที่ 37°C

-ในการทำการปรับค่า คงที่ k_p, k_I และ k_d ค่า ทำให้ได้กราฟที่เข้าใกล้จุดที่ตั้งค่าของอุณหภูมิในเวลาที่สั้นลงมากขึ้นและเข้าใกล้ค่าที่ตั้งไว้ 37°C

4.3 การทดลองปรับทิศทางของช่องอากาศที่อุณหภูมิ 35 – 38 °C



รูปที่ 4.6 ผลที่ได้จากความสัมพันธ์กันระหว่างความชื้นและอุณหภูมิ

-ในขั้นแรกเราทำปรับให้อากาศ Flow ผ่านน้ำทั้งสองช่อง (M)

- ต่อมาทำการปรับให้อากาศ Flow ผ่านน้ำได้อย่างเดียว (H)

- ขั้นสุดท้ายทำการปรับให้อากาศ Flow ไม่ผ่านน้ำ (L)

. เมื่อทำการหาความชื้น (%RH) ของระบบเมื่อทดลองปรับทิศทางของช่องอากาศที่อุณหภูมิ 35-38 °C เรามี 3 โหมดด้วยกัน คือ High (H) Medium (M) และ Low(L) จะพบว่าจากกราฟที่ได้พบว่า เมื่อทำการ Flow อากาศ แล้ว การ Flow ผ่านน้ำได้อย่างเดียวในตอนเริ่มต้นจะทำให้มีความชื้นสูงและค่อยๆ ลดลงมาเมื่อ อุณหภูมิเพิ่มขึ้น เมื่อทำการ Flow ผ่านทั้งสองช่องจะทำให้ความชื้นอยู่ที่ 60%(RH) เมื่อเวลาผ่านไปจะลดลงเล็กน้อย

บทที่ 5

สรุป วิจารณ์ผลการทดลอง และข้อเสนอแนะ

สรุป และวิจารณ์ผลการทดลอง

โครงงานนี้เป็นการออกแบบตู้อบเด็กทารกแรกเกิดในระบบควบคุมโดยใช้ไมโครคอนโทรลเลอร์ Psoc เป็นตัวประมวลผลเพื่อควบคุมการทำงาน เพื่อแสดงผลผ่านทาง Graphic LCD และได้นำระบบควบคุมแบบพีไอดี มาใช้ในการเขียนโปรแกรมประมวลผล และ ได้นำ Lab View มาใช้ในการรับและส่งข้อมูลแบบ data locker และสามารถที่จะตั้งค่าอุณหภูมิได้ในภาคการศึกษาที่ 2 นี้ ได้ทำการทดลองหาค่าคงที่ของสมการ การควบคุมแบบพีไอดี โดยใช้ขดลวดความร้อนเป็นโหลด ทำหน้าที่ให้ความร้อน จากผลการทดลองที่ได้สามารถสรุปได้ดังนี้

1. สามารถควบคุมอุณหภูมิในช่วง 35°C-38°C

2. ค่า K_p , K_i และK_d มีผลต่อความเสถียรภาพของระบบ โดยที่ K_p จะมีผลเมื่ออุณหภูมิเกิดการเปลี่ยนแปลงโดยดูได้จากสมการในระบบ PID และค่า K_i จะมีผลต่อการแกว่งของระบบ ส่วนค่า K_d มีผลต่อการเข้าสู่จุดกำหนด (set point) ของอุณหภูมิที่ใกล้เคียงและมีเสถียรภาพของระบบ

โดยค่าที่เลือกใช้ เป็นดังนี้ $K_p = 80$

$K_i = 0.001$

3. เมื่อทำการหาความชื้น (%RH) ของระบบเมื่อทดลองปรับทิศทางของช่องอากาศที่อุณหภูมิ 35-38 °C เรามี 3 โหมดด้วยกัน คือ High (H) Medium (M) และ Low(L) จะพบว่าจากกราฟที่ได้พบว่า เมื่อทำการ Flow อากาศ แล้ว จะพบได้ว่าเมื่อ อุณหภูมิต่ำลง เราควรใช้ mode L ปรับค่าเพื่อทำให้ความชื้นสูง และ เมื่อทำให้อุณหภูมิและความชื้นอยู่ในช่วงอุณหภูมิประมาณ 37°C ควรใช้ mode M เพื่อให้ความชื้นมีค่าที่เข้าใกล้ 60%(RH) และเมื่ออุณหภูมิให้มีค่าสูงขึ้นควรปรับค่า% ความชื้นเพื่อให้ค่าความชื้นสูงขึ้นเนื่องจากว่าค่าอุณหภูมิสูงขึ้นความชื้นจะลดลง

ปัญหาของโครงงาน

ส่วนที่ผิดพลาดจะเป็นการเซ็ต และการส่งข้อมูลให้กับ Lab view ในการประเมินค่าส่งและรับ

ข้อเสนอแนะ

ในการทดลองเพื่อหาค่าคงที่ที่เหมาะสมนั้นจะต้องทำในระบบปิด และต้องเป็นระบบที่เหมือนกันทุกประการ ทั้งอากาศภายนอก และตำแหน่งของอุปกรณ์ต่างๆ ที่ใช้ทำการทดลอง เพื่อจะได้สามารถเปรียบเทียบผลการทดลองได้อย่างถูกต้องแม่นยำมากที่สุด

เอกสารอ้างอิง

1. เกรียงศักดิ์ จีระแพทย์, "แนวทางการพัฒนาระบบเพื่อลดอัตราตายของทารกแรกเกิด", หนังสือวันอนามัยโลก, 2541 , หน้า 69-74

2. เกรียงศักดิ์ จีระแพทย์, "ทารกกลุ่มเสี่ยง", กุมารเวชศาสตร์, เล่ม 1, 2540, หน้า 228-236

3. เกรียงศักดิ์ จีระแพทย์, "เกณฑ์การประเมินการดูแลทารกแรกเกิด โครงการลูกเกิดรอดแม่ปลอดภัย", กรมอนามัย กระทรวงสาธารณสุข, 2541

4. เกรียงศักดิ์ จีระแพทย์, "การดูแลระบบการหายใจในทารกแรกเกิด", 2536

5. ประพุทธ ศิริปุณย์ และอุรพล บุญประกอบ, "ทารกแรกเกิด", โครงการตำรา-ศิริราช คณะแพทย์ศาสตร์ศิริราชพยาบาล มหาวิทยาลัยมหิดล, 2533

6. ชัยวัตน์ ลิ้มตระกูลจิตร วิไล และวรพจน์ กราบแก้ววัฒนกุล, "เรียนรู้และปฏิบัติการไมโคร-คอนโทรลเลอร์ MCS-51", โนเวทีฟเอ็กพอริเมนต์ กรุงเทพ

7. สมยศ จุณณะปิยะ, "การประยุกต์ใช้งานไมโครคอนโทรลเลอร์", สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง, 2543

8. คู่มือการใช้งาน V-85 atom infant incubator

9. สำเนาเอกสาร "ระบบการควบคุมป้อนกลับ"

ภาคผนวก

โปรแกรมกำหนดค่า Delay และแสดงผลอุณหภูมิ (PSoC)

```
#include <m8c.h>        // part specific constants and macros
#include "PSoCAPI.h"    // PSoC API definitions for all User Modules
#include<ioport.h>
#include<delay.h>
#pragma interrupt_handler PSoC_GPIO_ISR
char Rx ;
//BYTE  vol[];
//BYTE  voll[];
void PSoC_GPIO_ISR(void)
        {
        ClrBit1_1;
        Delay100uS(XXX);            // XXX คือค่า Delay ที่ใช้ปรับอุณหภูมิ
        SetBit1_1;
        }

void main()
{
  BYTE i;
        LCD_1_Start();
        M8C_EnableIntMask(INT_MSK0, INT_MSK0_GPIO);
        UART_1_CmdReset();
        UART_1_IntCntl(UART_1_ENABLE_RX_INT);
        UART_1_Start(UART_1_PARITY_NONE);
        M8C_EnableGInt;
        LCD_1_Position(0,0);
        LCD_1_PrCString("TEMP:");
        LCD_1_Position(1,0);
        LCD_1_PrCString("HUMI:");
        while (1)
```

```
    {

        if (UART_1_bCmdCheck()!=0)
        {
            LCD_1_Position(0,5);
            LCD_1_PrCString("        ");

            LCD_1_Position(0,5);
            for (i=0;i<7;i++)
            {
                vol[i]=UART_1_aRxBuffer[i];
                LCD_1_WriteData(UART_1_aRxBuffer[i]);
            }
            LCD_1_Position(1,5);
            LCD_1_PrCString("        ");

            LCD_1_Position(1,5);
            for (i=7;i<13;i++)
            {
                voll[i-6]=UART_1_aRxBuffer[i];
                LCD_1_WriteData(UART_1_aRxBuffer[i]);
            }
            //for
            UART_1_PutCRLF();
            UART_1_CmdReset();

        }
    }

}
```

ส่วนรับข้อมูลจากSensor SHT15 และส่งข้อมูลไปยัง PsoC โดยส่งแบบอนุกรม

(MCS-51 AT89C52)

```
#include<AT89x52.h>
#include <intrins.h> //Keil library (is used for _nop()_ operation)
#include <math.h> //Keil library
#include <stdio.h> //Keil library


typedef union
{ unsigned int i;
float f;
} value;
value humi_val,temp_val;
float Humi,Temp;
unsigned char round,count;

enum {TEMP,HUMI};
#define DATA P1_6
#define SCK  P1_7
#define noACK 0
#define ACK 1

#define STATUS_REG_W 0x06 //000 0011 0
#define STATUS_REG_R 0x07 //000 0011 1
#define MEASURE_TEMP 0x03 //000 0001 1
#define MEASURE_HUMI 0x05 //000 0010 1
#define RESET 0x1e //000 1111 0


void start (void)
```

```
{
SCON = 0x52; // set speacial function register
TMOD = 0x20;
TH1 = 0xfd; // Baud Rate = 9600
TCON =0x69;
TR1 = 1;
}

char s_write_byte(unsigned char value)
{
unsigned char i,error=0;
for (i=0x80;i>0;i/=2) //shift bit for masking
    {
    if (i & value) DATA=1; //masking value with i , write to SENSI-BUS
        else DATA=0;
            SCK=1; //clk for SENSI-BUS
            _nop_();_nop_();_nop_(); //pulswith approx. 5 us
            SCK=0;
    }
    DATA=1; //release DATA-line
    SCK=1; //clk #9 for ack
    error=DATA; //check ack (DATA will be pulled down by SHT11)
    SCK=0;
    return error; //error=1 in case of no acknowledge
}

char s_read_byte(unsigned char ack)
{
unsigned char i,val=0;
DATA=1; //release DATA-line
for (i=0x80;i>0;i/=2) //shift bit for masking
    {
```

```
        SCK=1; //clk for SENSI-BUS
        if (DATA) val=(val | i); //read bit
        SCK=0;
        }
        DATA=!ack; //in case of "ack==1" pull down DATA-Line
        SCK=1; //clk #9 for ack
        _nop_();_nop_();_nop_(); //pulswith approx. 5 us
        SCK=0;
        DATA=1; //release DATA-line
        return val;
    }

void s_transstart(void)
    {
    DATA=1; SCK=0; //Initial state
    _nop_();
    SCK=1;
    _nop_();
    DATA=0;
    _nop_();
    SCK=0;
    _nop_();_nop_();_nop_();
    SCK=1;
    _nop_();
    DATA=1;
    _nop_();
    SCK=0;
    }

void s_connectionreset(void)
    {
    unsigned char i;
```

```
	DATA=1; SCK=0; //Initial state
	for(i=0;i<9;i++) //9 SCK cycles
		{
		SCK=1;
		SCK=0;
		}
	s_transstart(); //transmission start
	}

char s_measure(unsigned char *p_value, unsigned char *p_checksum, unsigned char mode)
	{
	unsigned error=0;
	unsigned int i;
	s_transstart(); //transmission start
	switch(mode)
		{
		case TEMP : error+=s_write_byte(MEASURE_TEMP); break;
		case HUMI : error+=s_write_byte(MEASURE_HUMI); break;
		default : break;
		}
		for (i=0;i<65535;i++) if(DATA==0) break; //wait until sensor has finished the
measurement
			if(DATA) error+=1; // or timeout (~2 sec.) is reached
			*(p_value) =s_read_byte(ACK); //read the first byte (MSB)
			*(p_value+1)=s_read_byte(ACK); //read the second byte (LSB)
			*p_checksum =s_read_byte(noACK); //read checksum
			return error;
	}

void calc_sth15(float *p_humidity ,float *p_temperature)
	{
	const float C1=-4.0; // for 12 Bit
```

```
const float C2=+0.0405; // for 12 Bit
const float C3=-0.0000028; // for 12 Bit
const float T1=+0.01; // for 14 Bit @ 5V
const float T2=+0.00008; // for 14 Bit @ 5V

float rh=*p_humidity; // rh: Humidity [Ticks] 12 Bit
float t=*p_temperature; // t: Temperature [Ticks] 14 Bit
float rh_lin; // rh_lin: Humidity linear
float rh_true; // rh_true: Temperature compensated humidity
float t_C; // t_C : Temperature [?C]

t_C=t*0.01 - 40; //calc. temperature from ticks to [?C]
rh_lin=C3*rh*rh + C2*rh + C1; //calc. humidity from ticks to [%RH]
rh_true=(t_C-25)*(T1+T2*rh)+rh_lin; //calc. temperature compensated humidity [%RH]
if(rh_true>100)rh_true=100; //cut if the value is outside of
if(rh_true<0.1)rh_true=0.1; //the physical possible range
*p_temperature=t_C; //return temperature [?C]
*p_humidity=rh_true; //return humidity[%RH]
}

void main(void)
{
unsigned char error,checksum;
unsigned int i;
start ();
s_connectionreset();

while(1)
    {
    error=0;
    error+=s_measure((unsigned char*) &humi_val.i,&checksum,HUMI);
    error+=s_measure((unsigned char*) &temp_val.i,&checksum,TEMP);
```

```
            if(error!=0) s_connectionreset(); //in case of an error: connection reset
                    else
                    {
                    humi_val.f=(float)humi_val.i; //converts integer to float
                    temp_val.f=(float)temp_val.i; //converts integer to float
                    calc_sth15(&humi_val.f,&temp_val.f); //calculate humidity,
                    printf("%3.3f %3.3f\r",temp_val.f,humi_val.f);
                    }
            for(i=0;i<20000;i++);
            }
    }
```

โปรแกรมส่วนข้อมูลหลัก

```
//#include <REGX52.H>
#include<AT89x52.h>
#include <intrins.h> //Keil library (is used for _nop()_ operation)
#include <math.h> //Keil library
#include <stdio.h> //Keil library


typedef union
{ unsigned int i;
float f;
} value;
value humi_val,temp_val;
float Humi,Temp;
//unsigned char round,count;
unsigned char c;

enum {TEMP,HUMI};
#define DATA P1_6
#define SCK  P1_7
#define noACK 0
#define ACK 1

#define STATUS_REG_W 0x06 //000 0011 0
#define STATUS_REG_R 0x07 //000 0011 1
#define MEASURE_TEMP 0x03 //000 0001 1
#define MEASURE_HUMI 0x05 //000 0010 1
#define RESET 0x1e //000 1111 0
```

```
start ()
        {
        //IE=0x90; // Set EA = 1, ES = 1
        SCON=0x52; // Serial Port working in mode 1
        PCON=0; // Serial port working in normal mode
        T2MOD=0x00; //
        RCAP2H=0xFF; // Set Baud rate to 9600 bps
        RCAP2L=0xDC; // Set Baud rate to 9600 bps
        T2CON=0x34; // Enable Timer2 TR2 = '1' and use overflow of Timer 2 to
generate clock signal for both Tx & Rx of serial
        }

char s_write_byte(unsigned char value)
        {
        unsigned char i,error=0;
        for (i=0x80;i>0;i/=2) //shift bit for masking
                {
                if (i & value) DATA=1; //masking value with i , write to SENSI-BUS
                        else DATA=0;
                                SCK=1; //clk for SENSI-BUS
                                _nop_();_nop_();_nop_(); //pulswith approx. 5 us
                                SCK=0;
                }
                DATA=1; //release DATA-line
                SCK=1; //clk #9 for ack
                error=DATA; //check ack (DATA will be pulled down by SHT11)
                SCK=0;
                return error; //error=1 in case of no acknowledge
        }

char s_read_byte(unsigned char ack)
        {
        unsigned char i,val=0;
        DATA=1; //release DATA-line
        for (i=0x80;i>0;i/=2) //shift bit for masking
                {
                SCK=1; //clk for SENSI-BUS
                if (DATA) val=(val | i); //read bit
                SCK=0;
                }
                DATA=!ack; //in case of "ack==1" pull down DATA-Line
                SCK=1; //clk #9 for ack
                _nop_();_nop_();_nop_(); //pulswith approx. 5 us
                SCK=0;
                DATA=1; //release DATA-line
                return val;
        }

void s_transstart(void)
        {
```

```
        DATA=1; SCK=0; //Initial state
        _nop_();
        SCK=1;
        _nop_();
        DATA=0;
        _nop_();
        SCK=0;
        _nop_();_nop_();_nop_();
        SCK=1;
        _nop_();
        DATA=1;
        _nop_();
        SCK=0;
        }

void s_connectionreset(void)
        {
        unsigned char i;
        DATA=1; SCK=0; //Initial state
        for(i=0;i<9;i++) //9 SCK cycles
                {
                SCK=1;
                SCK=0;
                }
        s_transstart(); //transmission start
        }

char s_measure(unsigned char *p_value, unsigned char *p_checksum, unsigned char
mode)
        {
        unsigned error=0;
        unsigned int i;
        s_transstart(); //transmission start
        switch(mode)
                {
                case TEMP : error+=s_write_byte(MEASURE_TEMP); break;
                case HUMI : error+=s_write_byte(MEASURE_HUMI); break;
                default : break;
                }
                for (i=0;i<65535;i++) if(DATA==0) break; //wait until sensor has
finished the measurement
                        if(DATA) error+=1; // or timeout (~2 sec.) is reached
                        *(p_value) =s_read_byte(ACK); //read the first byte (MSB)
                        *(p_value+1)=s_read_byte(ACK); //read the second byte (LSB)
                        *p_checksum =s_read_byte(noACK); //read checksum
                        return error;
        }

void calc_sth15(float *p_humidity ,float *p_temperature)
        {
```

```
        const float C1=-4.0; // for 12 Bit
        const float C2=+0.0405; // for 12 Bit
        const float C3=-0.0000028; // for 12 Bit
        const float T1=+0.01; // for 14 Bit @ 5V
        const float T2=+0.00008; // for 14 Bit @ 5V

        float rh=*p_humidity; // rh: Humidity [Ticks] 12 Bit
        float t=*p_temperature; // t: Temperature [Ticks] 14 Bit
        float rh_lin; // rh_lin: Humidity linear
        float rh_true; // rh_true: Temperature compensated humidity
        float t_C; // t_C : Temperature [?C]

        t_C=t*0.01 - 40; //calc. temperature from ticks to [?C]
        rh_lin=C3*rh*rh + C2*rh + C1; //calc. humidity from ticks to [%RH]
        rh_true=(t_C-25)*(T1+T2*rh)+rh_lin; //calc. temperature compensated
humidity [%RH]
        if(rh_true>100)rh_true=100; //cut if the value is outside of
        if(rh_true<0.1)rh_true=0.1; //the physical possible range
        *p_temperature=t_C; //return temperature [?C]
        *p_humidity=rh_true; //return humidity[%RH]
        }



void main(void)
        {
        unsigned char error,checksum;
        start ();
        s_connectionreset();

        while(1)
                {
                error=0;
                error+=s_measure((unsigned char*) &humi_val.i,&checksum,HUMI);
                error+=s_measure((unsigned char*) &temp_val.i,&checksum,TEMP);
                if(error!=0) s_connectionreset(); //in case of an error: connection reset
                        else
                        {
                        humi_val.f=(float)humi_val.i; //converts integer to float
                        temp_val.f=(float)temp_val.i; //converts integer to float
                        calc_sth15(&humi_val.f,&temp_val.f); //calculate humidity,
                        if(RI==1)
                                {
                                c=SBUF;
                                RI=0;
                                TI=0;
                                switch(c)
                                        {
                                        case 0x33 : SBUF=c;printf("%3.3f\n
",temp_val.f); break;
```

```
                                        case 0x34 : SBUF=c;printf("%3.3f\n
",humi_val.f); break;
                                        default:break;
                                        }
                                }
                        }

                }
        }
```

PC-Control Infant Incubator

Hardware and Software User's Manual

ผู้จัดทำ

นายเฉลิมพล คำอิน รหัส 49015188

นายดิเรก ปัทมะสุคนธ์ รหัส 49015191

ว่าที่ ร.ต. เดชฤทธิ์ จันทร์แก้ว รหัส 49015193

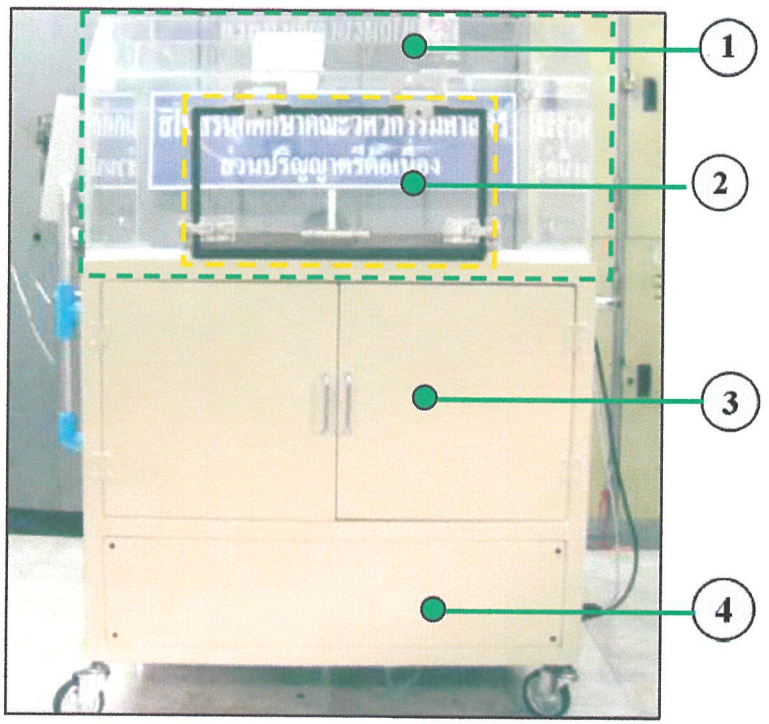
อาจารย์ที่ปรึกษา

รศ.ดร. ชูชาติ ปิณฑวิรุจน์

Table of Contents

1 Knowing the Parts

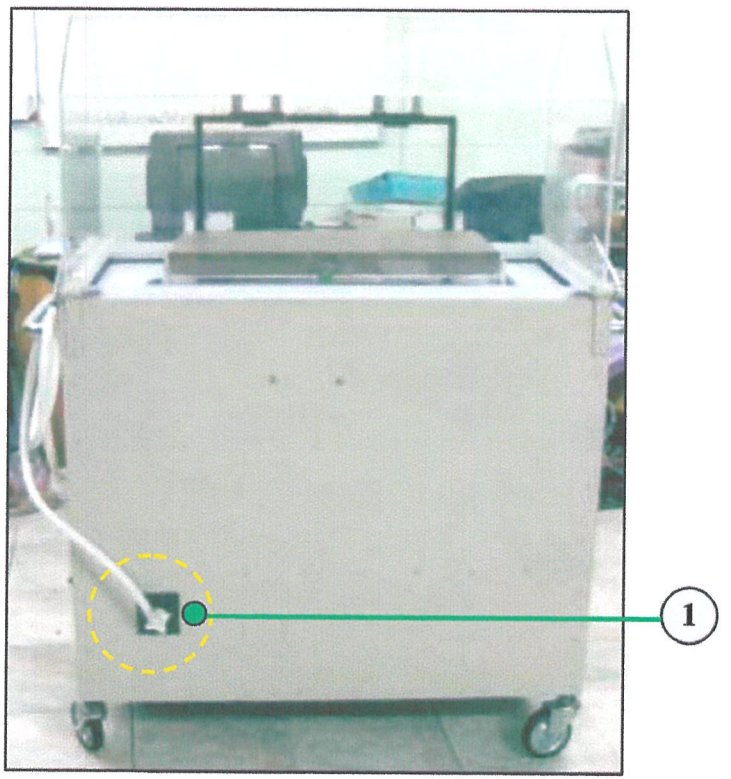
Front Side



1. ห้องนอนควบคุมอุณหภูมิและความชื้น
2. ประตูปิด เปิด สำหรับทำงานในตู้
3. ห้องเก็บของ
4. ห้องติดตั้งวงจร

1 Knowing the Parts

Back Side



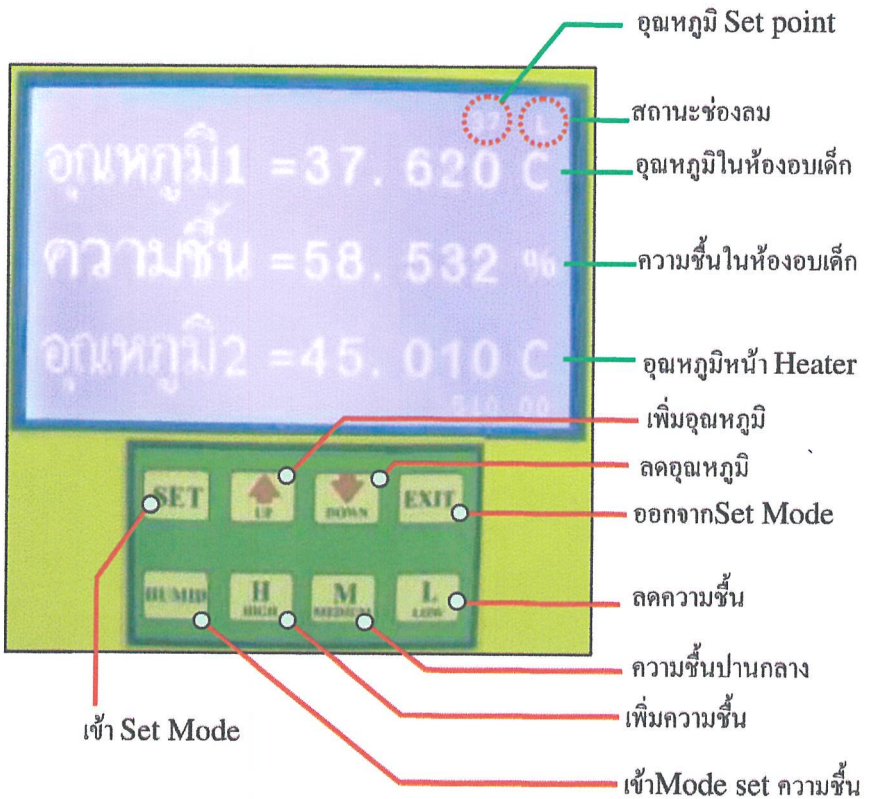
1. สายน้ำเข้า

1 Knowing the Parts

Left Side

1. Display Panel and Control

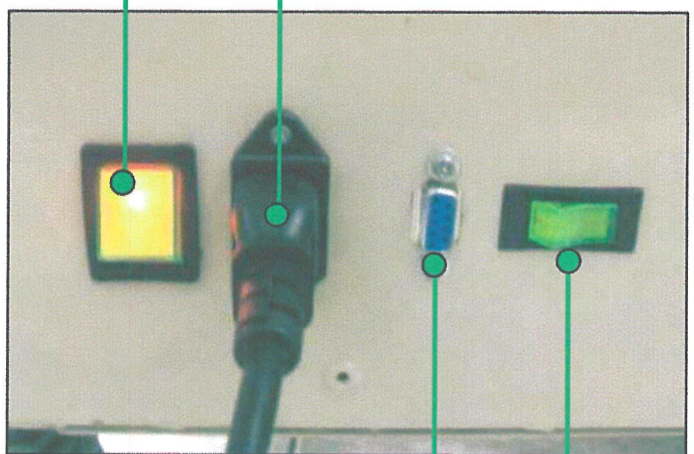
Display Panel and Control



1 Knowing the Parts

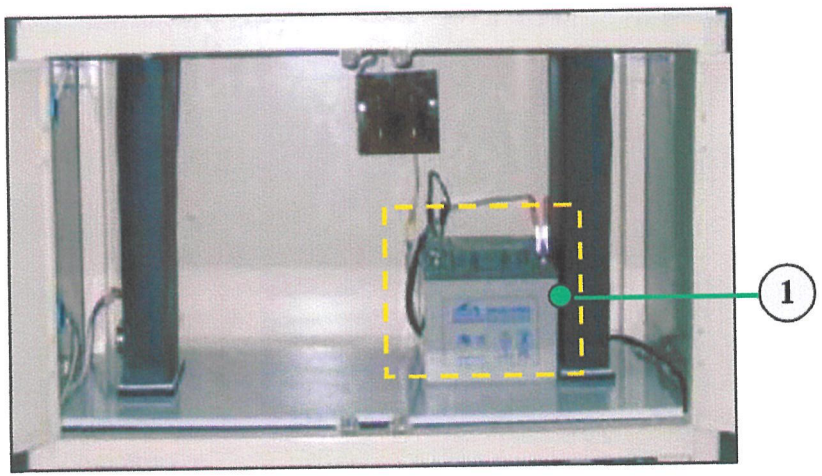
Right Side

1. AC 220 V. , สวิตท์ ปิด เปิดระบบ และ serial port



1 Knowing the Parts

In Side



1. Battery Back up 12 V. (แดง = + , ดำ = -)

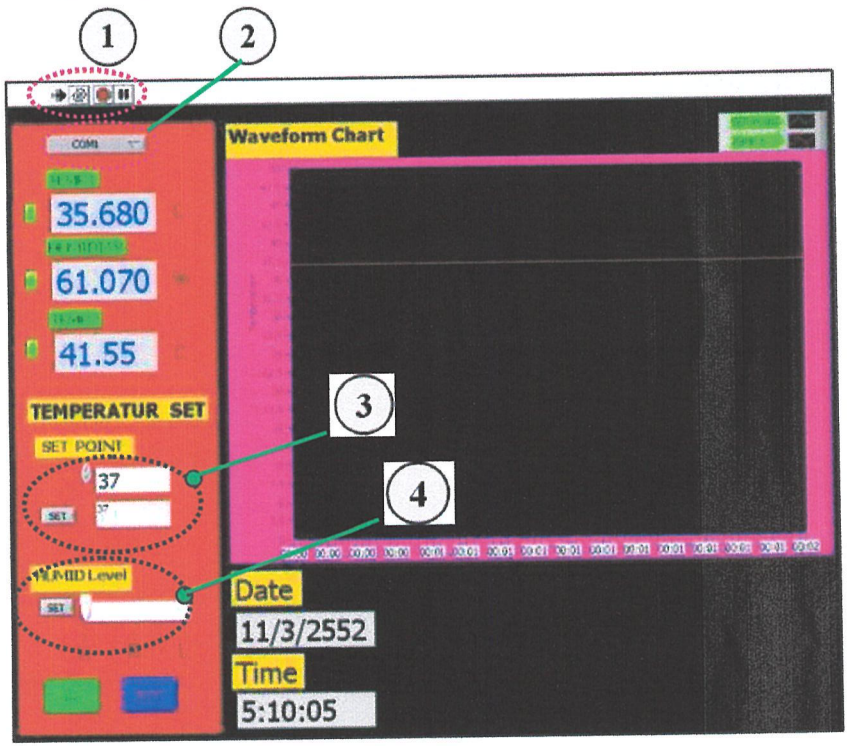
2 Using the PC-Control Infant Incubator

2.1 เปิดระบบ ขั้นแรกคีบสาย Battery Back up 12 V. (แดง = + , ดำ = -)ในชั้นเก็บของ >>> เสียบสายไฟ AC 220V. >>> เปิด Switch 220V. >>> เปิด Switch Battery Back up ระบบเริ่มทำงาน หน้าจอมีการแสดงผล โดยอุณหภูมิเริ่มต้นจะถูก Set ไว้ที่ 37 องศาเซียลเซียส

2.2 Setระบบ กด SET ที่หน้าปัดจะเข้า Mode อุณหภูมิ กด UP สำหรับเพิ่มอุณหภูมิ กด DOWN สำหรับลดอุณหภูมิ เมื่อ Set ค่าได้ตามต้องการแล้วถึงจะกด HUMID เพื่อเข้า Mode การ Set ความชื้น กด H HIGH เพิ่มความชื้น กด L LOW ลดความชื้น กด M MEDIUM เลือกความชื้นปานกลาง และกด EXIT เพื่อออกจาก Set Mode

หมายเหตุ: เมื่อเปิดระบบมาครั้งแรกและเข้า Mode ความชื้นจะไม่สามารถกด M MEDIUM ได้ต้องกด H HIGH หรือ L LOW ก่อนถึงจะกด M MEDIUM ได้ ระยะห่างระหว่างการกดเลือกความชื้นประมาณ 20 วินาที

2.3 Interface กับ Computer(LabVIEW)



>>> ต่อสาย RS 232จากComputer เข้ากับตู้ >>>

1. กด ให้โปรแกรมทำงาน (กด หยุดการทำงานของโปรแกรมทั้งหมด)
2. เลือก Port >>> กด START เพื่อRun โปรแกรม(กด STOP หยุดRun)
3. การ Set อุณหภูมิ ใส่ค่าในช่องด้านล่าง แล้วกดปุ่ม SET โดย LabVIEW จะส่งค่าไปให้ PSoC ทำการ Set อีกที
4. การ Set ความชื้นคือเลือกค่าในช่อง แล้วกดปุ่ม SET โดย LabVIEW จะส่งค่าไปให้ PSoC ทำการ Set อีกที

หมายเหตุ: ระยะห่างระหว่างการกดเลือกความชื้นประมาณ 20 วินาที

5.ส่วนของ Data logger

	A	B	C	D	E	F
1	35	31.88	74.176	42.71	11/3/2552	5:12:05
2	35	31.88	74.176	42.71	11/3/2552	5:12:06
3	35	31.88	74.149	42.71	11/3/2552	5:12:27
4	35	31.99	74.134	42.19	11/3/2552	5:12:28
5	35	31.99	74.134	41.92	11/3/2552	5:12:49
6	35	32.04	74.134	41.92	11/3/2552	5:12:50
7	35	32.04	74.134	41.92	11/3/2552	5:12:51

A . อุณหภูมิ Set point

B . อุณหภูมิในตู้อบ

C . ความชื้นในตู้อบ

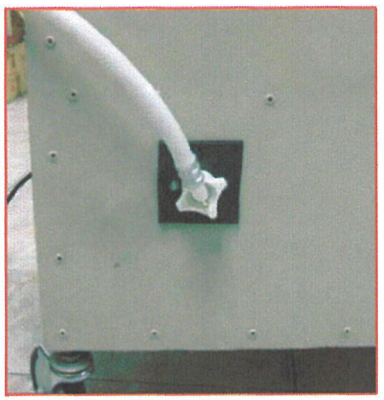
D . อุณหภูมิหน้า Heater

E . วัน-เดือน-ปี

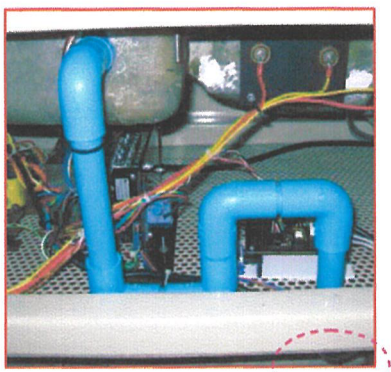
F . เวลา (ชั่วโมง-นาที-วินาที)

ข้อมูลของ Data logger ถูก Set ให้ Saveไว้ที่ Drive C:\ มีชื่อตามวันที่ที่เปิด Runโปรแกรม บันทึกเป็นไฟล์ Excel (เช่น 11-3-2552)

2.4 Water system



สายยางจะนำน้ำเข้าระบบโดยมีการปิดเปิดอัตโนมัติ เมื่อเติมน้ำได้ที่ระบบจะตัด Valve อัตโนมัติ หรือถ้าน้ำลดต่ำลง ระบบจะต่อ Valve ให้น้ำเข้า



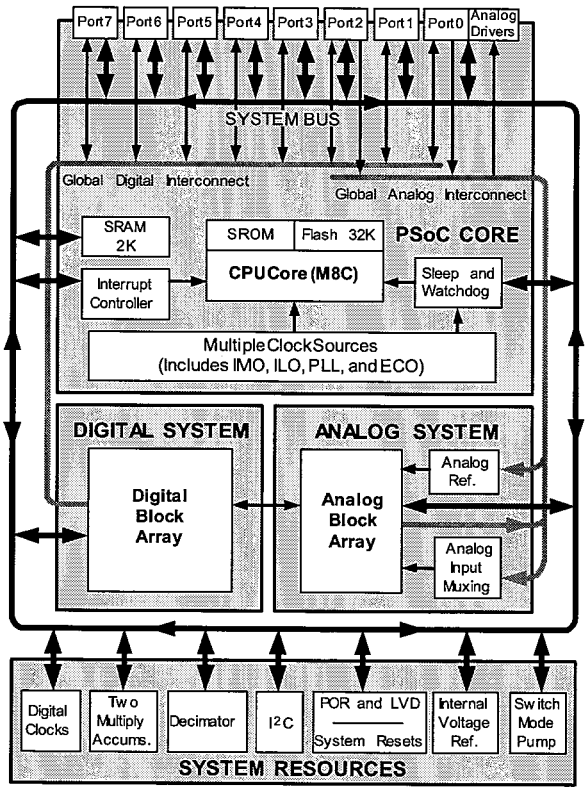
กรณี Sensor มีปัญหา ไม่ตัดการทำงานของ Valve น้ำจะไหลออกตามท่อ

บริเวณน้ำล้น

CY8C29466, CY8C29566, CY8C29666, and CY8C29866

Features

- **Powerful Harvard Architecture Processor**
  - M8C Processor Speeds to 24 MHz
  - Two 8x8 Multiply, 32-Bit Accumulate
  - Low Power at High Speed
  - 3.0V to 5.25V Operating Voltage
  - Operating Voltages Down to 1.0V Using On-Chip Switch Mode Pump (SMP)
  - Industrial Temperature Range: -40°C to +85°C
- **Advanced Peripherals (PSoC Blocks)**
  - 12 Rail-to-Rail Analog PSoC Blocks Provide:
    - Up to 14-Bit ADCs
    - Up to 9-Bit DACs
    - Programmable Gain Amplifiers
    - Programmable Filters and Comparators
  - 16 Digital PSoC Blocks Provide:
    - 8- to 32-Bit Timers, Counters, and PWMs
    - CRC and PRS Modules
    - Up to 4 Full-Duplex UARTs
    - Multiple SPI™ Masters or Slaves
    - Connectable to all GPIO Pins
  - Complex Peripherals by Combining Blocks
- **Precision, Programmable Clocking**
  - Internal ±2.5% 24/48 MHz Oscillator
  - 24/48 MHz with Optional 32.768 kHz Crystal
  - Optional External Oscillator, up to 24 MHz
  - Internal Oscillator for Watchdog and Sleep
- **Flexible On-Chip Memory**
  - 32K Bytes Flash Program Storage 50,000 Erase/Write Cycles
  - 2K Bytes SRAM Data Storage
  - In-System Serial Programming (ISSP™)
  - Partial Flash Updates
  - Flexible Protection Modes
  - EEPROM Emulation in Flash
- **Programmable Pin Configurations**
  - 25 mA Sink on all GPIO
  - Pull up, Pull down, High Z, Strong, or Open Drain Drive Modes on all GPIO
  - Up to 12 Analog Inputs on GPIO
  - Four 40 mA Analog Outputs on GPIO
  - Configurable Interrupt on all GPIO
- **Additional System Resources**
  - I²C™ Slave, Master, and Multi-Master to 400 kHz
  - Watchdog and Sleep Timers
  - User-Configurable Low Voltage Detection
  - Integrated Supervisory Circuit
  - On-Chip Precision Voltage Reference
- **Complete Development Tools**
  - Free Development Software (PSoC™ Designer)
  - Full-Featured, In-Circuit Emulator and Programmer
  - Full Speed Emulation
  - Complex Breakpoint Structure
  - 128K Bytes Trace Memory
  - Complex Events
  - C Compilers, Assembler, and Linker



PSoC™ Functional Overview

The PSoC™ family consists of many *Mixed-Signal Array with On-Chip Controller* devices. These devices are designed to replace multiple traditional MCU-based system components with one, low cost single-chip programmable device. PSoC devices include configurable blocks of analog and digital logic, as well as programmable interconnects. This architecture allows the user to create customized peripheral configurations that match the requirements of each individual application. Additionally, a fast CPU, Flash program memory, SRAM data memory, and configurable IO are included in a range of convenient pinouts and packages.

The PSoC architecture, as illustrated on the left, is comprised of four main areas: PSoC Core, Digital System, Analog System, and System Resources. Configurable global busing allows all the device resources to be combined into a complete custom system. The PSoC CY8C29x66 family can have up to eight IO ports that connect to the global digital and analog interconnects, providing access to 16 digital blocks and 12 analog blocks.

The PSoC Core

The PSoC Core is a powerful engine that supports a rich feature set. The core includes a CPU, memory, clocks, and configurable GPIO (General Purpose IO).

The M8C CPU core is a powerful processor with speeds up to 24 MHz, providing a four MIPS 8-bit Harvard architecture microprocessor. The CPU utilizes an interrupt controller with 25 vec-

tors, to simplify programming of real time embedded events. Program execution is timed and protected using the included Sleep and Watch Dog Timers (WDT).

Memory encompasses 32 KB of Flash for program storage, 2 KB of SRAM for data storage, and up to 2 KB of EEPROM emulated using the Flash. Program Flash utilizes four protection levels on blocks of 64 bytes, allowing customized software IP protection.

The PSoC device incorporates flexible internal clock generators, including a 24 MHz IMO (internal main oscillator) accurate to 2.5% over temperature and voltage. The 24 MHz IMO can also be doubled to 48 MHz for use by the digital system. A low power 32 kHz ILO (internal low speed oscillator) is provided for the Sleep timer and WDT. If crystal accuracy is desired, the ECO (32.768 kHz external crystal oscillator) is available for use as a Real Time Clock (RTC) and can optionally generate a crystal-accurate 24 MHz system clock using a PLL. The clocks, together with programmable clock dividers (as a System Resource), provide the flexibility to integrate almost any timing requirement into the PSoC device.

PSoC GPIOs provide connection to the CPU, digital and analog resources of the device. Each pin's drive mode may be selected from eight options, allowing great flexibility in external interfacing. Every pin also has the capability to generate a system interrupt on high level, low level, and change from last read.

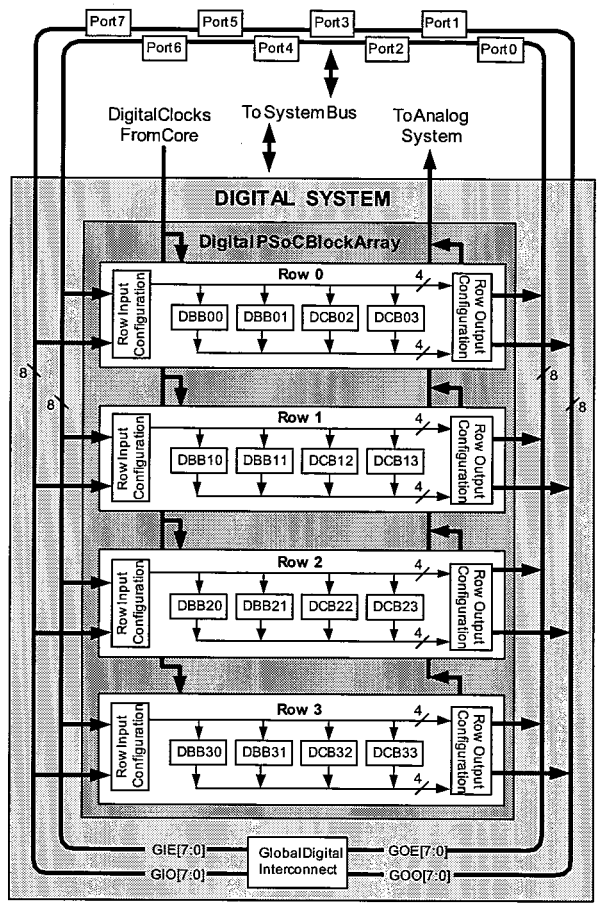
The Digital System

The Digital System is composed of 16 digital PSoC blocks. Each block is an 8-bit resource that can be used alone or combined with other blocks to form 8, 16, 24, and 32-bit peripherals, which are called user module references. Digital peripheral configurations include those listed below.

- PWMs (8 to 32 bit)
- PWMs with Dead band (8 to 32 bit)
- Counters (8 to 32 bit)
- Timers (8 to 32 bit)
- UART 8 bit with selectable parity (up to 4)
- SPI master and slave (up to 4 each)
- I2C slave and multi-master (1 available as a System Resource)
- Cyclical Redundancy Checker/Generator (8 to 32 bit)
- IrDA (up to 4)
- Pseudo Random Sequence Generators (8 to 32 bit)

The digital blocks can be connected to any GPIO through a series of global buses that can route any signal to any pin. The buses also allow for signal multiplexing and for performing logic operations. This configurability frees your designs from the constraints of a fixed peripheral controller.

Digital blocks are provided in rows of four, where the number of blocks varies by PSoC device family. This allows you the optimum choice of system resources for your application. Family resources are shown in the table titled "PSoC Device Characteristics" on page 3.



Digital System Block Diagram

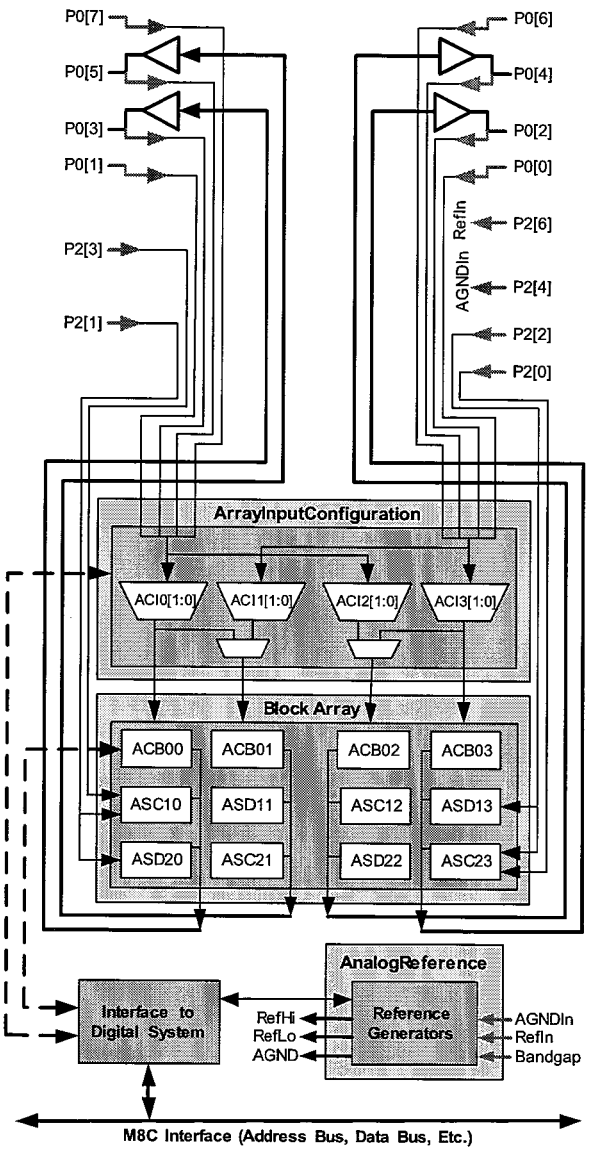
The Analog System

The Analog System is composed of 12 configurable blocks, each comprised of an opamp circuit allowing the creation of complex analog signal flows. Analog peripherals are very flexible and can be customized to support specific application requirements. Some of the more common PSoC analog functions (most available as user modules) are listed below.

- Analog-to-digital converters (up to 4, with 6- to 14-bit resolution, selectable as Incremental, Delta Sigma, and SAR)
- Filters (2, 4, 6, or 8 pole band-pass, low-pass, and notch)
- Amplifiers (up to 4, with selectable gain to 48x)
- Instrumentation amplifiers (up to 2, with selectable gain to 93x)
- Comparators (up to 4, with 16 selectable thresholds)
- DACs (up to 4, with 6- to 9-bit resolution)
- Multiplying DACs (up to 4, with 6- to 9-bit resolution)
- High current output drivers (four with 40 mA drive as a Core Resource)

- 1.3V reference (as a System Resource)
- DTMF Dialer
- Modulators
- Correlators
- Peak Detectors
- Many other topologies possible

Analog blocks are provided in columns of three, which includes one CT (Continuous Time) and two SC (Switched Capacitor) blocks, as shown in the figure below.



Analog System Block Diagram

Additional System Resources

System Resources, some of which have been previously listed, provide additional capability useful to complete systems. Additional resources include a multiplier, decimator, switch mode pump, low voltage detection, and power on reset. Brief statements describing the merits of each system resource are presented below.

- Digital clock dividers provide three customizable clock frequencies for use in applications. The clocks can be routed to both the digital and analog systems. Additional clocks can be generated using digital PSoC blocks as clock dividers.
- Two multiply accumulates (MACs) provide fast 8-bit multipliers with 32-bit accumulate to assist in both general math as well as digital filters.
- The decimator provides a custom hardware filter for digital signal, processing applications including the creation of Delta Sigma ADCs.
- The I2C module provides 100 and 400 kHz communication over two wires. Slave, master, and multi-master modes are all supported.
- Low Voltage Detection (LVD) interrupts can signal the application of falling voltage levels, while the advanced POR (Power On Reset) circuit eliminates the need for a system supervisor.
- An internal 1.3 voltage reference provides an absolute reference for the analog system, including ADCs and DACs.
- An integrated switch mode pump (SMP) generates normal operating voltages from a single 1.2V battery cell, providing a low cost boost converter.

PSoC Device Characteristics

Depending on your PSoC device characteristics, the digital and analog systems can have 16, 8, or 4 digital blocks and 12, 6, or 4 analog blocks. The following table lists the resources available for specific PSoC device groups. The PSoC device covered by this data sheet is highlighted below.

PSoC Device Characteristics

PSoC Device Group	Digital IO (max)	Digital Rows	Digital Blocks	Analog Inputs	Analog Outputs	Analog Columns	Analog Blocks	Amount of SRAM	Amount of Flash
CY8C29x66	**44**	**4**	**16**	**12**	**4**	**4**	**12**	**2K**	**32K**
CY8C27x43	44	2	8	12	4	4	12	256 Bytes	16K
CY8C24794	56	1	4	48	2	2	6	1K	16K
CY8C24x23	24	1	4	12	2	2	6	256 Bytes	4K
CY8C24x23A	24	1	4	12	2	2	6	256 Bytes	4K
CY8C21x34	28	1	4	28	0	2	4[a]	512 Bytes	8K
CY8C21x23	16	1	4	8	0	2	4[a]	256 Bytes	4K

a. Limited analog functionality.

Getting Started

The quickest path to understanding the PSoC silicon is by reading this data sheet and using the PSoC Designer Integrated Development Environment (IDE). This data sheet is an overview of the PSoC integrated circuit and presents specific pin, register, and electrical specifications. For in-depth information, along with detailed programming information, reference the *PSoC™ Mixed-Signal Array Technical Reference Manual.*

For up-to-date Ordering, Packaging, and Electrical Specification information, reference the latest PSoC device data sheets on the web at http://www.cypress.com/psoc.

Development Kits

Development Kits are available from the following distributors: Digi-Key, Avnet, Arrow, and Future. The Cypress Online Store at http://www.onfulfillment.com/cypressstore/ contains development kits, C compilers, and all accessories for PSoC development. Click on *PSoC (Programmable System-on-Chip)* to view a current list of available items.

Tele-Training

Free PSoC "Tele-training" is available for beginners and taught by a marketing or application engineer over the phone. Five training classes are available to accelerate the learning curve including introduction, designing, debugging, advanced design, advanced analog, as well as application-specific classes covering topics like PSoC and the LIN bus. For days and times of the tele-training, see http://www.cypress.com/support/training.cfm.

Consultants

Certified PSoC Consultants offer everything from technical assistance to completed PSoC designs. To contact or become a PSoC Consultant, go to the following Cypress support web site: http://www.cypress.com/support/cypros.cfm.

Technical Support

PSoC application engineers take pride in fast and accurate response. They can be reached with a 4-hour guaranteed response at http://www.cypress.com/support/login.cfm.

Application Notes

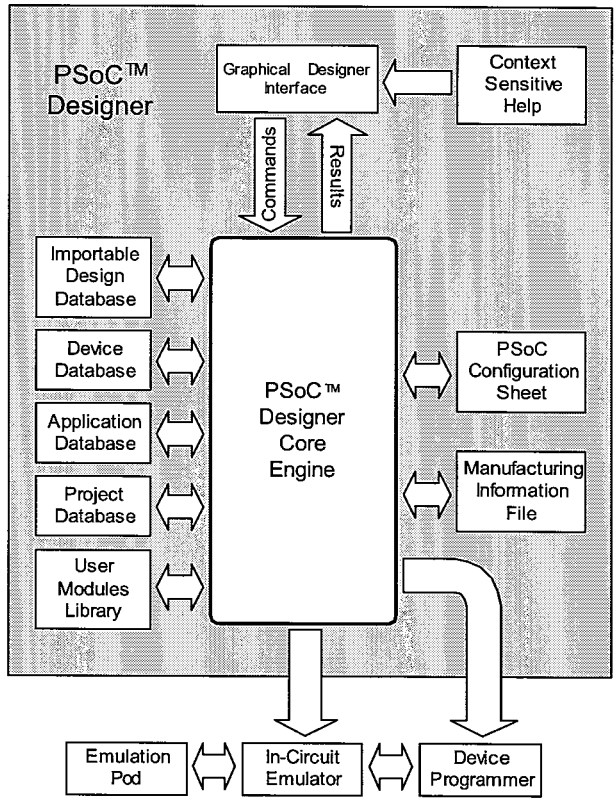
A long list of application notes will assist you in every aspect of your design effort. To locate the PSoC application notes, go to http://www.cypress.com/design/results.cfm.

Development Tools

The Cypress MicroSystems PSoC Designer is a Microsoft® Windows-based, integrated development environment for the Programmable System-on-Chip (PSoC) devices. The PSoC Designer IDE and application runs on Windows NT 4.0, Windows 2000, Windows Millennium (Me), or Windows XP. (Reference the PSoC Designer Functional Flow diagram below.)

PSoC Designer helps the customer to select an operating configuration for the PSoC, write application code that uses the PSoC, and debug the application. This system provides design database management by project, an integrated debugger with In-Circuit Emulator, in-system programming support, and the CYASM macro assembler for the CPUs.

PSoC Designer also supports a high-level C language compiler developed specifically for the devices in the family.



PSoC Designer Subsystems

PSoC Designer Software Subsystems

Device Editor

The Device Editor subsystem allows the user to select different onboard analog and digital components called user modules using the PSoC blocks. Examples of user modules are ADCs, DACs, Amplifiers, and Filters.

The device editor also supports easy development of multiple configurations and dynamic reconfiguration. Dynamic configuration allows for changing configurations at run time.

PSoC Designer sets up power-on initialization tables for selected PSoC block configurations and creates source code for an application framework. The framework contains software to operate the selected components and, if the project uses more than one operating configuration, contains routines to switch between different sets of PSoC block configurations at run time. PSoC Designer can print out a configuration sheet for a given project configuration for use during application programming in conjunction with the Device Data Sheet. Once the framework is generated, the user can add application-specific code to flesh out the framework. It's also possible to change the selected components and regenerate the framework.

Design Browser

The Design Browser allows users to select and import preconfigured designs into the user's project. Users can easily browse a catalog of preconfigured designs to facilitate time-to-design. Examples provided in the tools include a 300-baud modem, LIN Bus master and slave, fan controller, and magnetic card reader.

Application Editor

In the Application Editor you can edit your C language and Assembly language source code. You can also assemble, compile, link, and build.

Assembler. The macro assembler allows the assembly code to be merged seamlessly with C code. The link libraries automatically use absolute addressing or can be compiled in relative mode, and linked with other software modules to get absolute addressing.

C Language Compiler. A C language compiler is available that supports Cypress MicroSystems' PSoC family devices. Even if you have never worked in the C language before, the product quickly allows you to create complete C programs for the PSoC family devices.

The embedded, optimizing C compiler provides all the features of C tailored to the PSoC architecture. It comes complete with embedded libraries providing port and bus operations, standard keypad and display support, and extended math functionality.

Debugger

The PSoC Designer Debugger subsystem provides hardware in-circuit emulation, allowing the designer to test the program in a physical system while providing an internal view of the PSoC device. Debugger commands allow the designer to read and program and read and write data memory, read and write IO registers, read and write CPU registers, set and clear breakpoints, and provide program run, halt, and step control. The debugger also allows the designer to create a trace buffer of registers and memory locations of interest.

Online Help System

The online help system displays online, context-sensitive help for the user. Designed for procedural and quick reference, each functional subsystem has its own context-sensitive help. This system also provides tutorials and links to FAQs and an Online Support Forum to aid the designer in getting started.

Hardware Tools

In-Circuit Emulator

A low cost, high functionality ICE (In-Circuit Emulator) is available for development support. This hardware has the capability to program single devices.

The emulator consists of a base unit that connects to the PC by way of the USB port. The base unit is universal and will operate with all PSoC devices. Emulation pods for each device family are available separately. The emulation pod takes the place of the PSoC device in the target board and performs full speed (24 MHz) operation.

Designing with User Modules

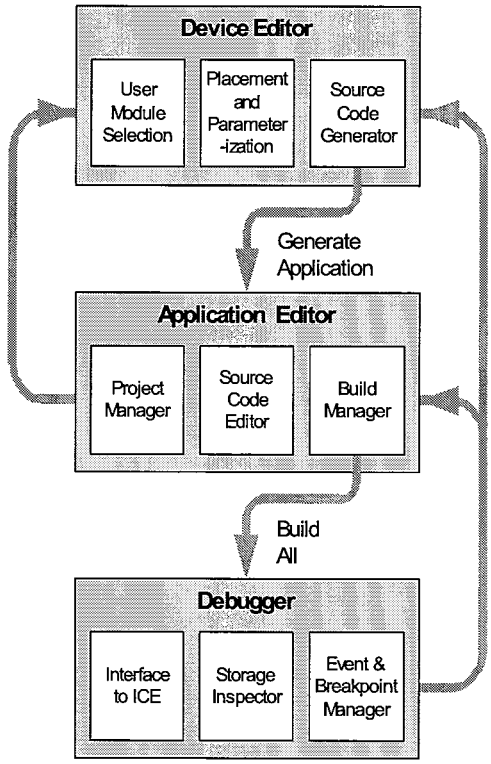
The development process for the PSoC device differs from that of a traditional fixed function microprocessor. The configurable analog and digital hardware blocks give the PSoC architecture a unique flexibility that pays dividends in managing specification change during development and by lowering inventory costs. These configurable resources, called PSoC Blocks, have the ability to implement a wide variety of user-selectable functions. Each block has several registers that determine its function and connectivity to other blocks, multiplexers, buses, and to the IO pins. Iterative development cycles permit you to adapt the hardware as well as the software. This substantially lowers the risk of having to select a different part to meet the final design requirements.

To speed the development process, the PSoC Designer Integrated Development Environment (IDE) provides a library of pre-built, pre-tested hardware peripheral functions, called "User Modules." User modules make selecting and implementing peripheral devices simple, and come in analog, digital, and mixed signal varieties. The standard User Module library contains over 50 common peripherals such as ADCs, DACs Timers, Counters, UARTs, and other not-so common peripherals such as DTMF Generators and Bi-Quad analog filter sections.

Each user module establishes the basic register settings that implement the selected function. It also provides parameters that allow you to tailor its precise configuration to your particular application. For example, a Pulse Width Modulator User Module configures one or more digital PSoC blocks, one for each 8 bits of resolution. The user module parameters permit you to establish the pulse width and duty cycle. User modules also provide tested software to cut your development time. The user module application programming interface (API) provides high-level functions to control and respond to hardware events at run-time. The API also provides optional interrupt service routines that you can adapt as needed.

The API functions are documented in user module data sheets that are viewed directly in the PSoC Designer IDE. These data sheets explain the internal operation of the user module and provide performance specifications. Each data sheet describes the use of each user module parameter and documents the setting of each register controlled by the user module.

The development process starts when you open a new project and bring up the Device Editor, a graphical user interface (GUI) for configuring the hardware. You pick the user modules you need for your project and map them onto the PSoC blocks with point-and-click simplicity. Next, you build signal chains by interconnecting user modules to each other and the IO pins. At this stage, you also configure the clock source connections and enter parameter values directly or by selecting values from drop-down menus. When you are ready to test the hardware configuration or move on to developing code for the project, you perform the "Generate Application" step. This causes PSoC Designer to generate source code that automatically configures the device to your specification and provides the high-level user module API functions.



User Module and Source Code Development Flows

The next step is to write your main program, and any sub-routines using PSoC Designer's Application Editor subsystem. The Application Editor includes a Project Manager that allows you to open the project source code files (including all generated code files) from a hierarchal view. The source code editor provides syntax coloring and advanced edit features for both C and assembly language. File search capabilities include simple string searches and recursive "grep-style" patterns. A single mouse click invokes the Build Manager. It employs a professional-strength "makefile" system to automatically analyze all file dependencies and run the compiler and assembler as necessary. Project-level options control optimization strategies used by the compiler and linker. Syntax errors are displayed in a console window. Double clicking the error message takes you directly to the offending line of source code. When all is correct, the linker builds a HEX file image suitable for programming.

The last step in the development process takes place inside the PSoC Designer's Debugger subsystem. The Debugger downloads the HEX image to the In-Circuit Emulator (ICE) where it runs at full speed. Debugger capabilities rival those of systems costing many times more. In addition to traditional single-step, run-to-breakpoint and watch-variable features, the Debugger provides a large trace buffer and allows you define complex breakpoint events that include monitoring address and data bus values, memory locations and external signals.

1.1.3 48-Pin Part Pinouts

Table 1-3. 48-Pin Part Pinout (SSOP)

Pin No.	Type		Pin Name	Description
	Digital	Analog		
1	IO	I	P0[7]	Analog column mux input.
2	IO	IO	P0[5]	Analog column mux input and column output.
3	IO	IO	P0[3]	Analog column mux input and column output.
4	IO	I	P0[1]	Analog column mux input.
5	IO		P2[7]	
6	IO		P2[5]	
7	IO	I	P2[3]	Direct switched capacitor block input.
8	IO	I	P2[1]	Direct switched capacitor block input.
9	IO		P4[7]	
10	IO		P4[5]	
11	IO		P4[3]	
12	IO		P4[1]	
13	Power		SMP	Switch Mode Pump (SMP) connection to external components required.
14	IO		P3[7]	
15	IO		P3[5]	
16	IO		P3[3]	
17	IO		P3[1]	
18	IO		P5[3]	
19	IO		P5[1]	
20	IO		P1[7]	I2C Serial Clock (SCL).
21	IO		P1[5]	I2C Serial Data (SDA).
22	IO		P1[3]	
23	IO		P1[1]	Crystal (XTALin), I2C Serial Clock (SCL).
24	Power		Vss	Ground connection.
25	IO		P1[0]	Crystal (XTALout), I2C Serial Data (SDA).
26	IO		P1[2]	
27	IO		P1[4]	Optional External Clock Input (EXTCLK).
28	IO		P1[6]	
29	IO		P5[0]	
30	IO		P5[2]	
31	IO		P3[0]	
32	IO		P3[2]	
33	IO		P3[4]	
34	IO		P3[6]	
35	Input		XRES	Active high external reset with internal pull down.
36	IO		P4[0]	
37	IO		P4[2]	
38	IO		P4[4]	
39	IO		P4[6]	
40	IO	I	P2[0]	Direct switched capacitor block input.
41	IO	I	P2[2]	Direct switched capacitor block input.
42	IO		P2[4]	External Analog Ground (AGND).
43	IO		P2[6]	External Voltage Reference (VREF).
44	IO	I	P0[0]	Analog column mux input.
45	IO	IO	P0[2]	Analog column mux input and column output.
46	IO	IO	P0[4]	Analog column mux input and column output.
47	IO	I	P0[6]	Analog column mux input.
48	Power		Vdd	Supply voltage.

LEGEND: A = Analog, I = Input, and O = Output.

CY8C29666 48-Pin PSoC Device